European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Michail G. Lagoudakis
Intelligent Systems Laboratory
School of ECE
Technical University of Crete
Chania, Crete, Greece

Spring 2023

Acknowledgment of Support

"Support for the Internationalization of Higher Education, School of Electrical and Computer Engineering, Technical University of Crete" (MIS 5150766), under the call for proposals "Support of the Internationalization of Higher Education – Technical University of Crete" (EDULLL 153).

The project is co-financed by Greece and the European Union (European Social Fund – ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning 2014-2020".

European Union
European Social Fund

Operational Programme
Human Resources Development, Education and Lifelong Learning
Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Today

- **Machine Learning**
  - supervised
  - unsupervised
  - reinforcement
  - theory
- **Examples**
  - hand-written digit recognition
  - polynomial curve-fitting
  - …

Machine Learning

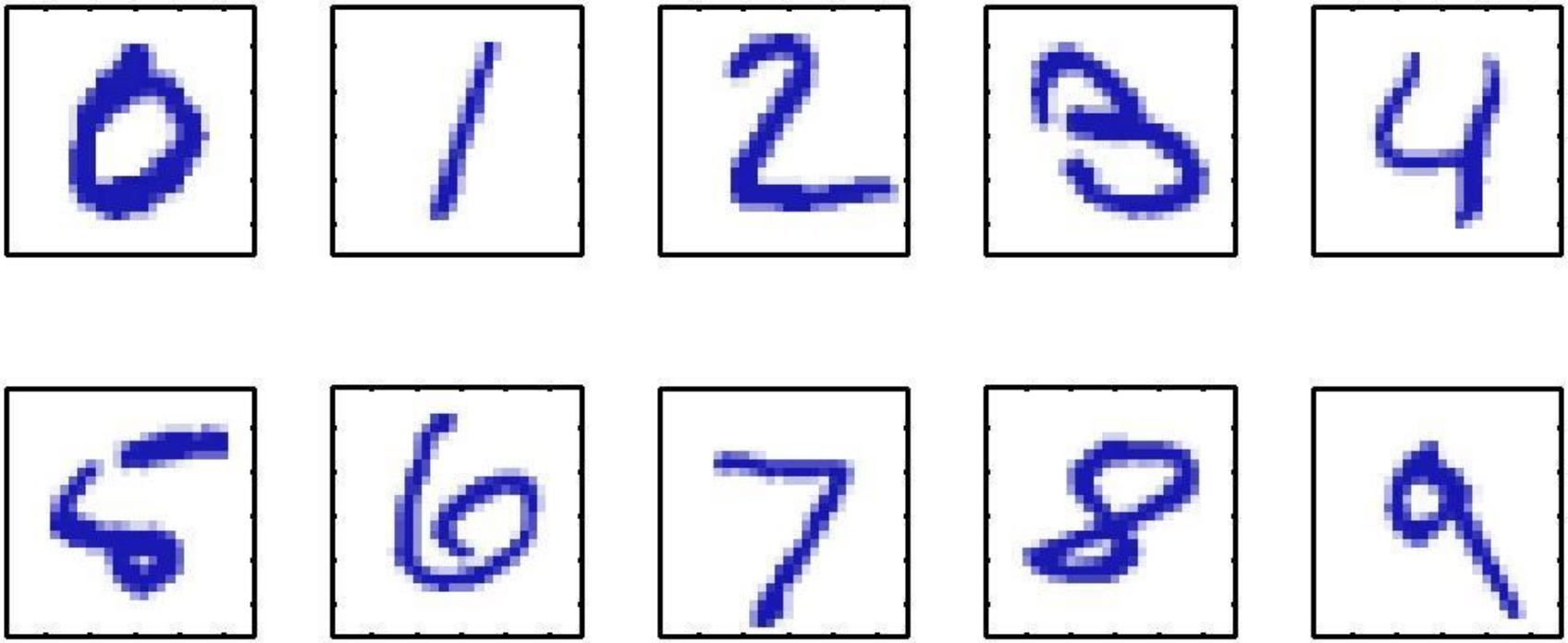
Machine Learning

- **Supervised Learning**
  - set of training data with inputs and targets
  - classification, regression, …
- **Unsupervised Learning**
  - set of training data with inputs, but without targets
  - clustering, density estimation, dimensionality reduction, …
- **Reinforcement Learning**
  - set of training trials of interaction with feedback by a critic
  - value function, decision policy, exploration vs. exploitation, …
- **Learning Theory**
  - theoretical investigations: what can be learned? how fast?

Examples

Hand-Written Digit Recognition

input: digital image (28x28 pixels)



target: label 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9

Hand-Written Digit Recognition (cnt'd)

- **Data**
  - training: set of hand-written digit images with correct labels
  - test: examples similar to the training set
- **Model Selection**
  - input x, output y(x)
  - generalization
- **Learning**
  - training phase
  - test phase
  - cross-validation

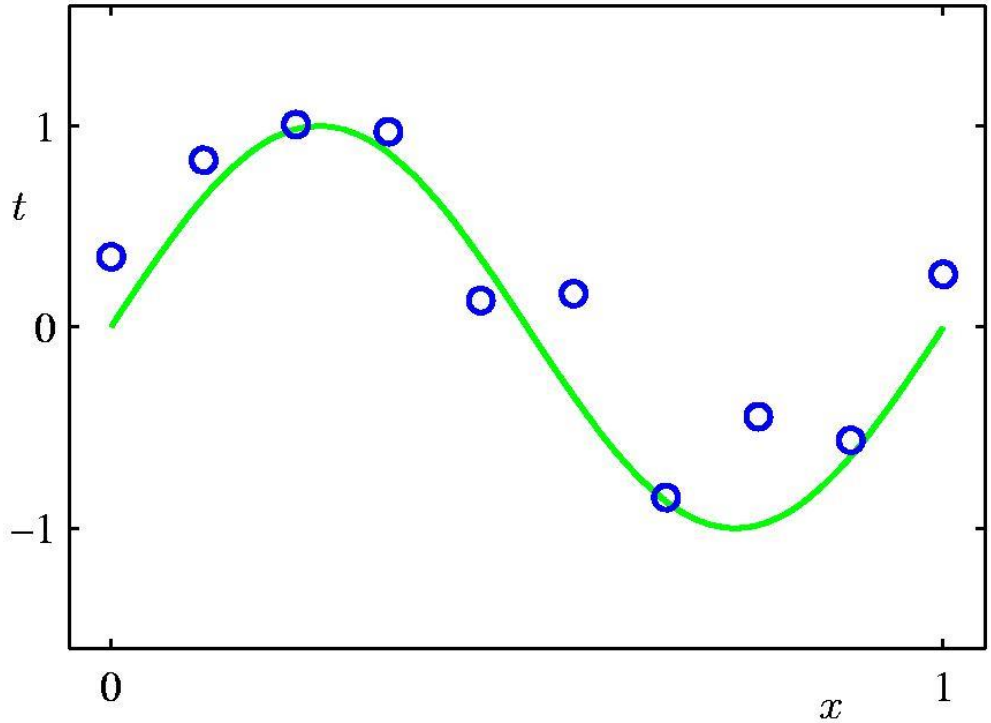
Hand-Written Digit Recognition (cnt'd)

- **MINST Data Set**
  - 60000 training examples
  - 10000 test examples
  - normalized to 20x20
  - fit to 28x28 square

Hand-Written Digit Recognition (cnt'd)

- **Feature Extraction/Selection**
  - preprocessing: translation and scaling to fit into a fixed box
  - reduction of the variability within each class
  - transformation to a space where the problem may be easier
- **Dimensionality Reduction**
  - reduce the dimensionality of the problem for speed-up
  - a set of carefully selected features instead of images
  - critical information may be discarded!

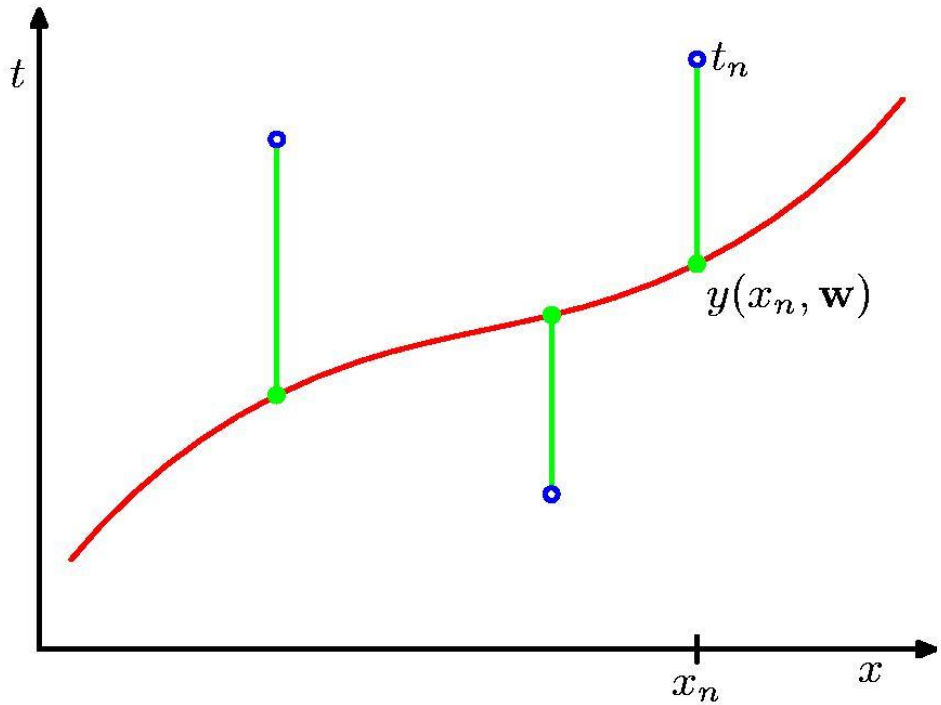
Polynomial Curve Fitting



$$t = \sin(2\pi x)$$

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

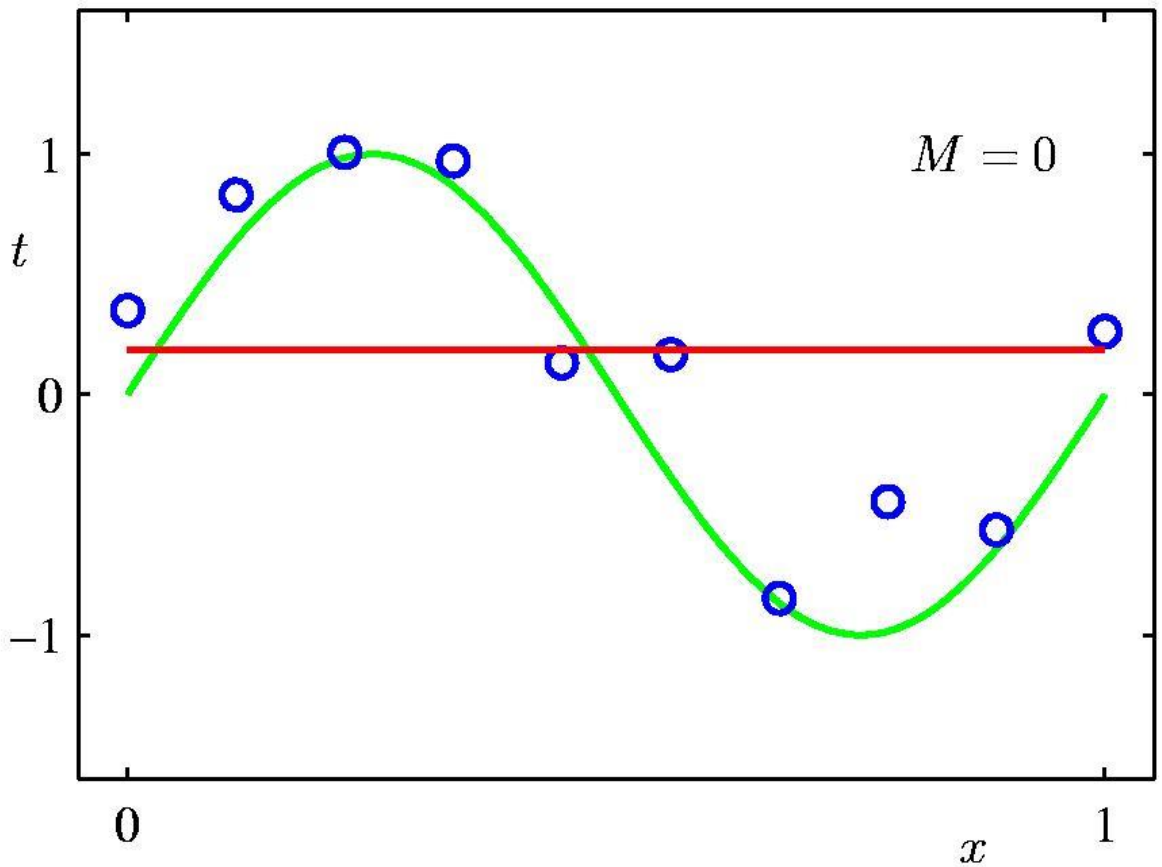
Sum-of-Squares Error Function



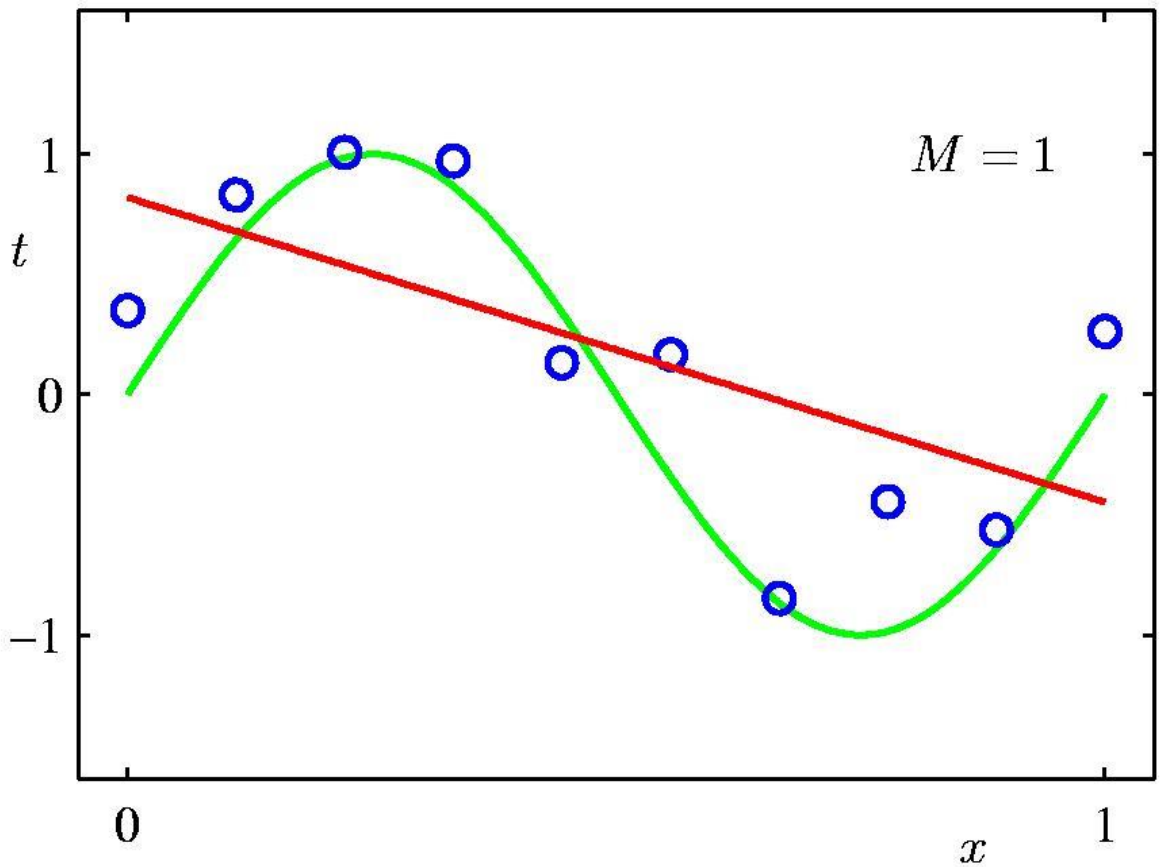
$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

It is a function minimization problem! Easy, because the derivative is a linear function of w!

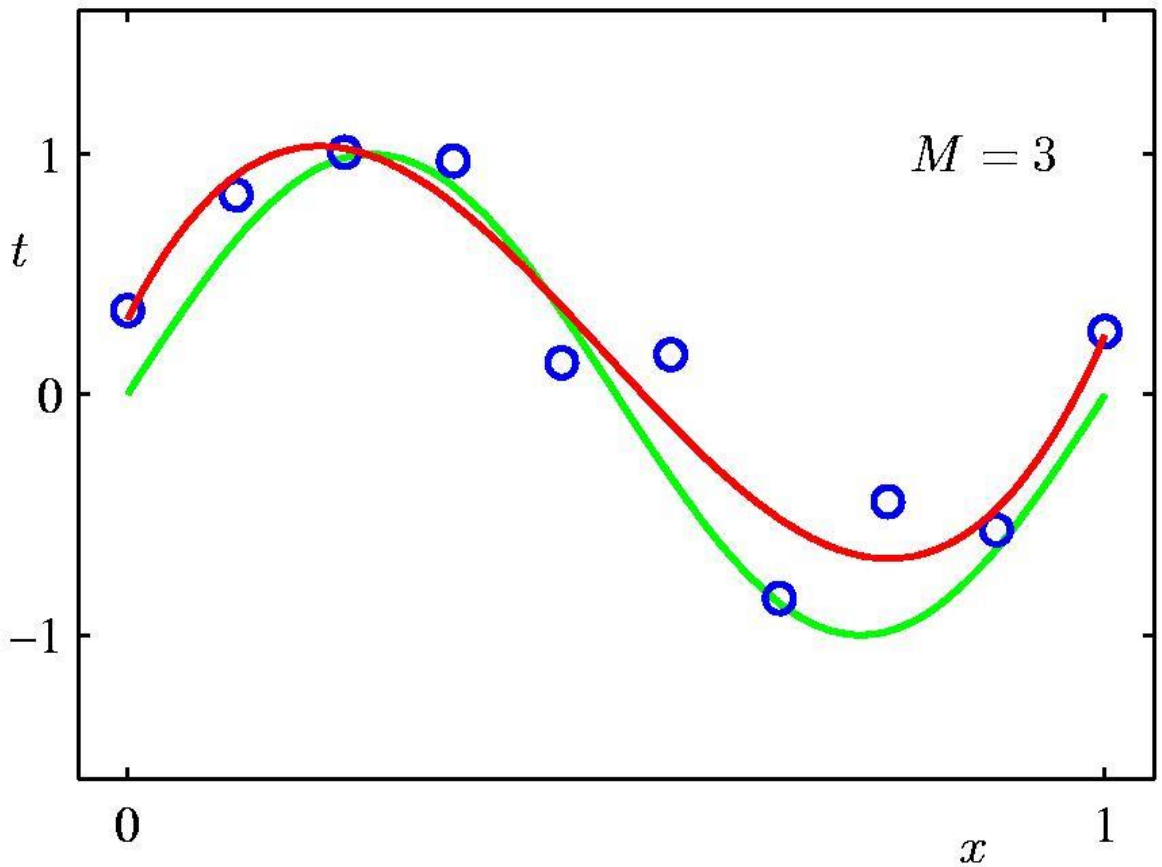
0th Order Polynomial



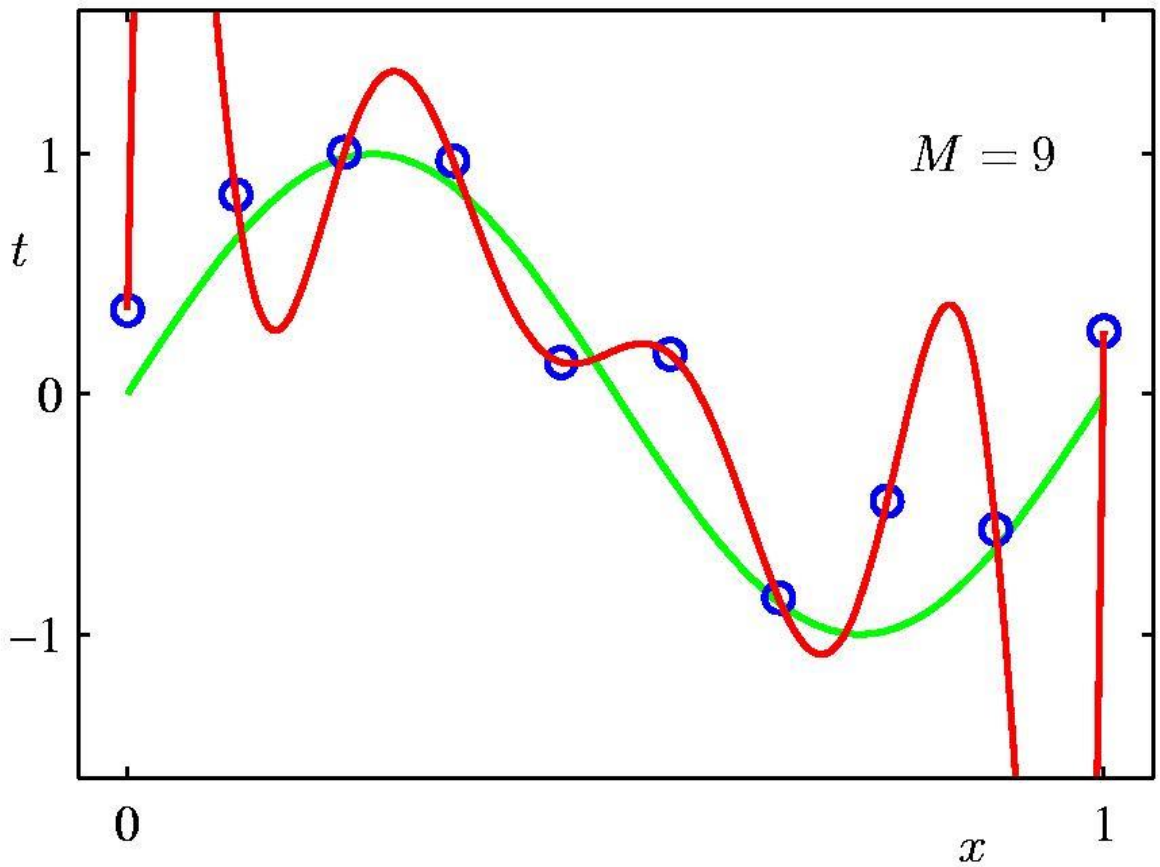
1st Order Polynomial



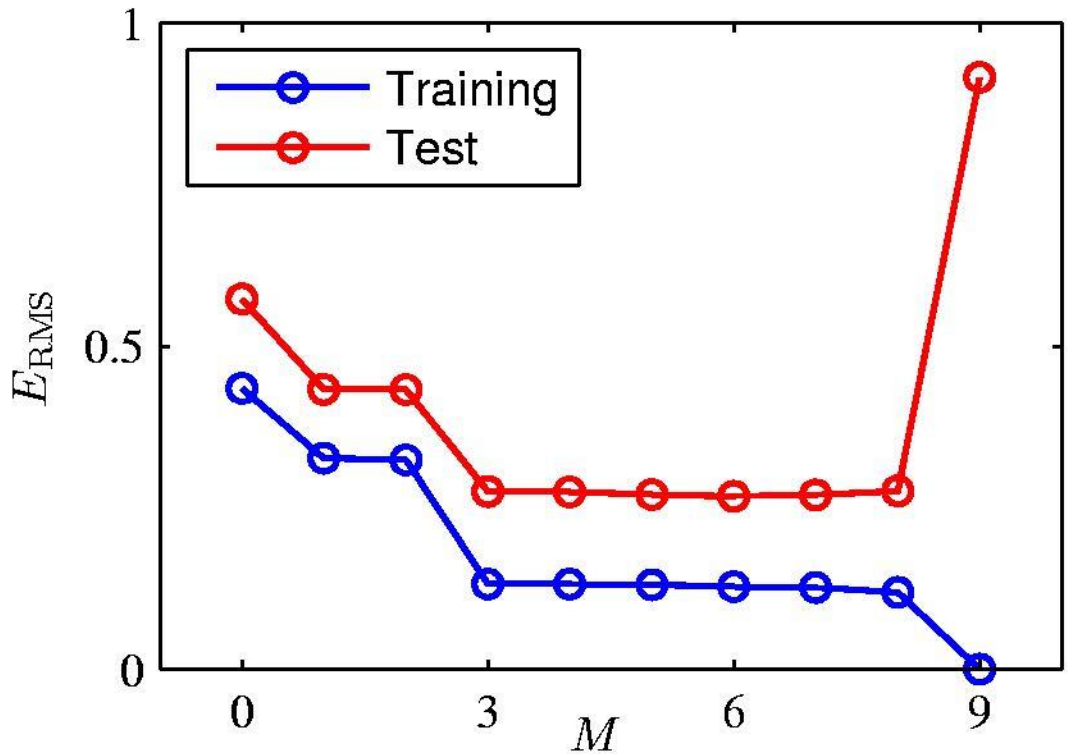
3rd Order Polynomial



9th Order Polynomial



Over-Fitting



|training| = 10 examples

|test|=100 examples

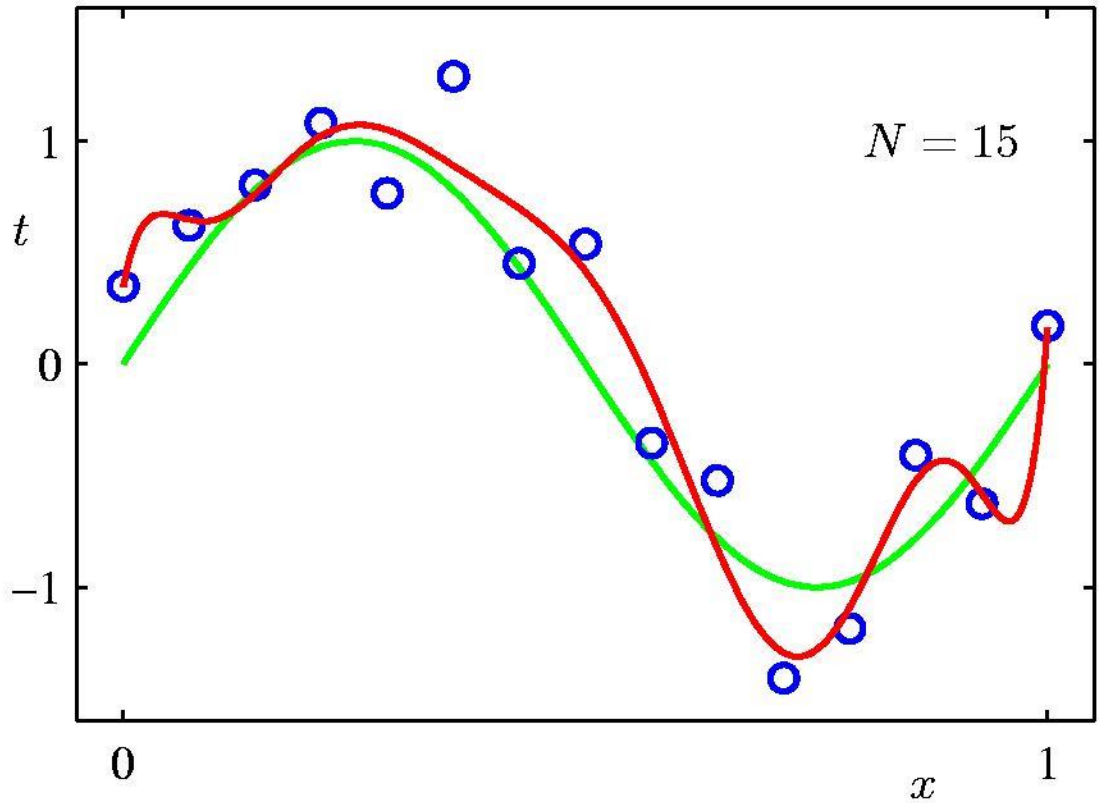
Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

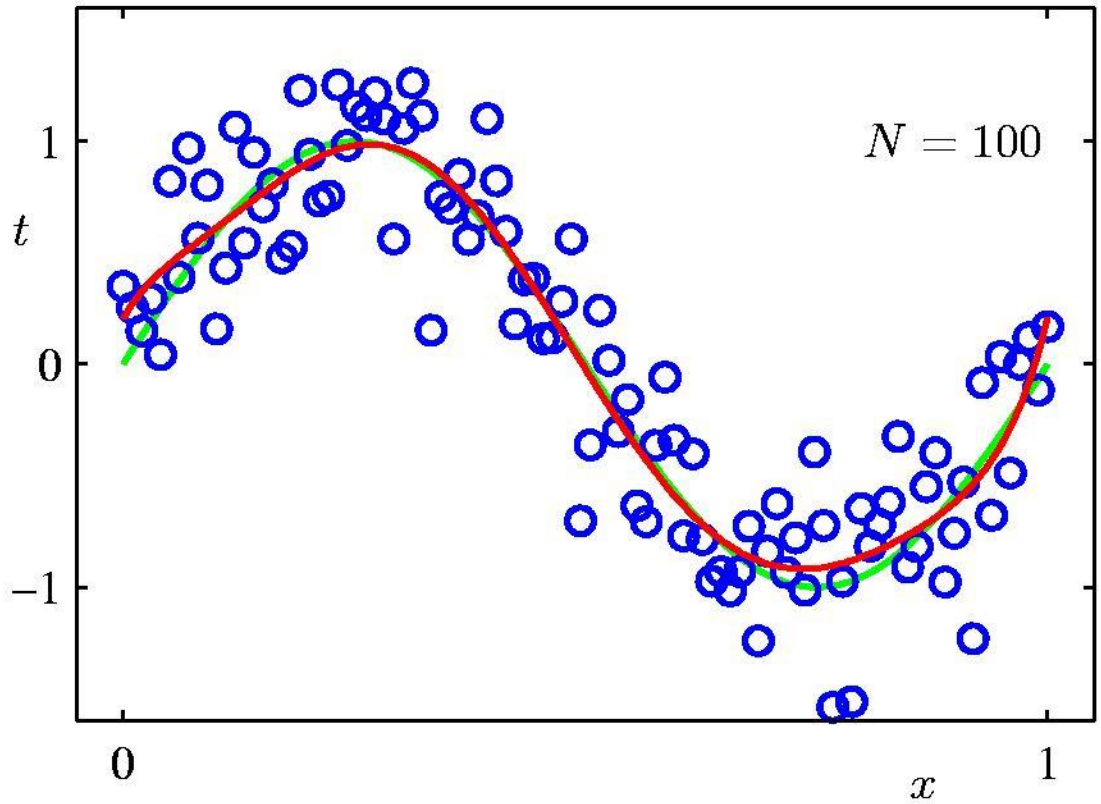
Data Set Size: $N = 15$

9th Order Polynomial



Data Set Size: $N = 100$

9th Order Polynomial



Regularization

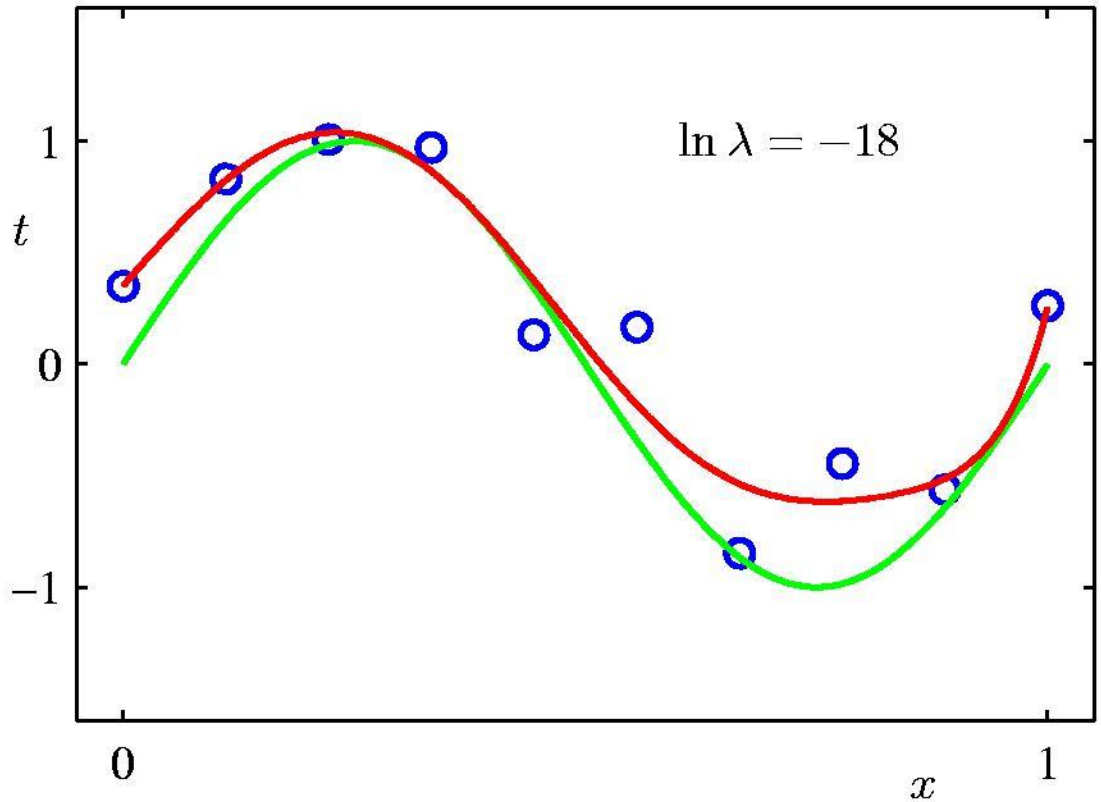
- Penalize large coefficient values

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

- where

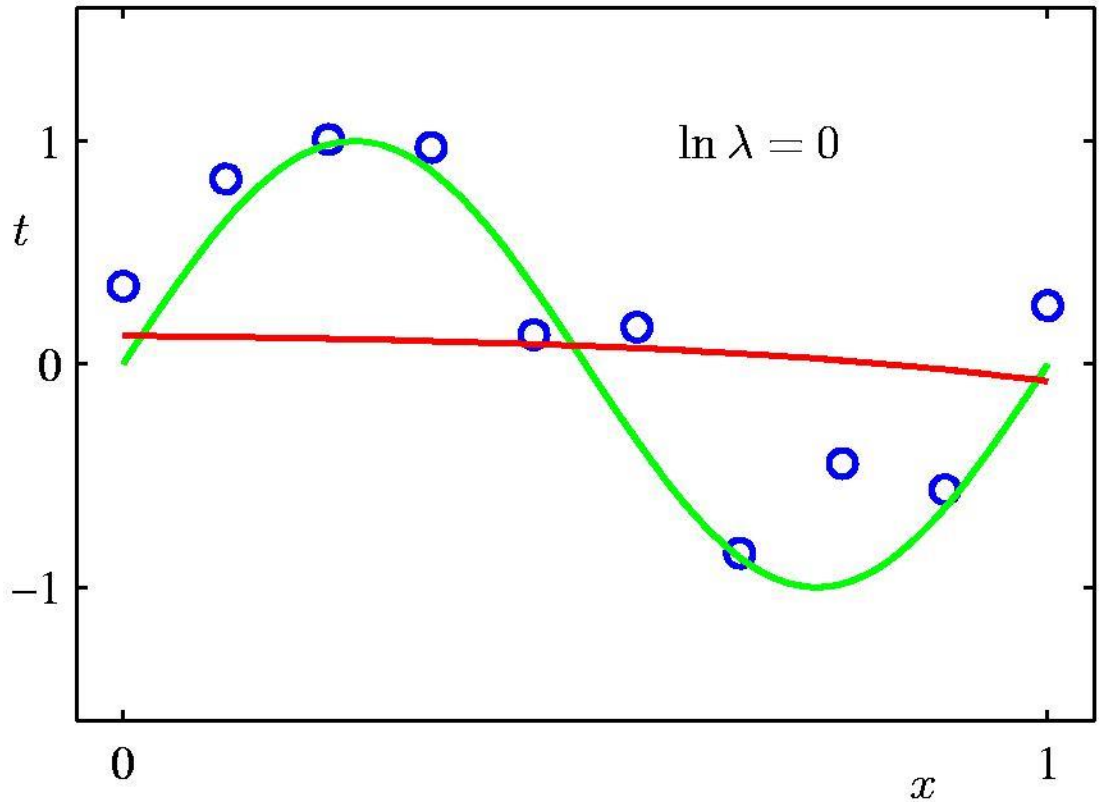
$$\|\mathbf{w}\|^2 = \mathbf{w}^\top \mathbf{w} = w_0^2 + w_1^2 + \ldots + w_n^2$$

Regularization: $\ln \lambda = -18$



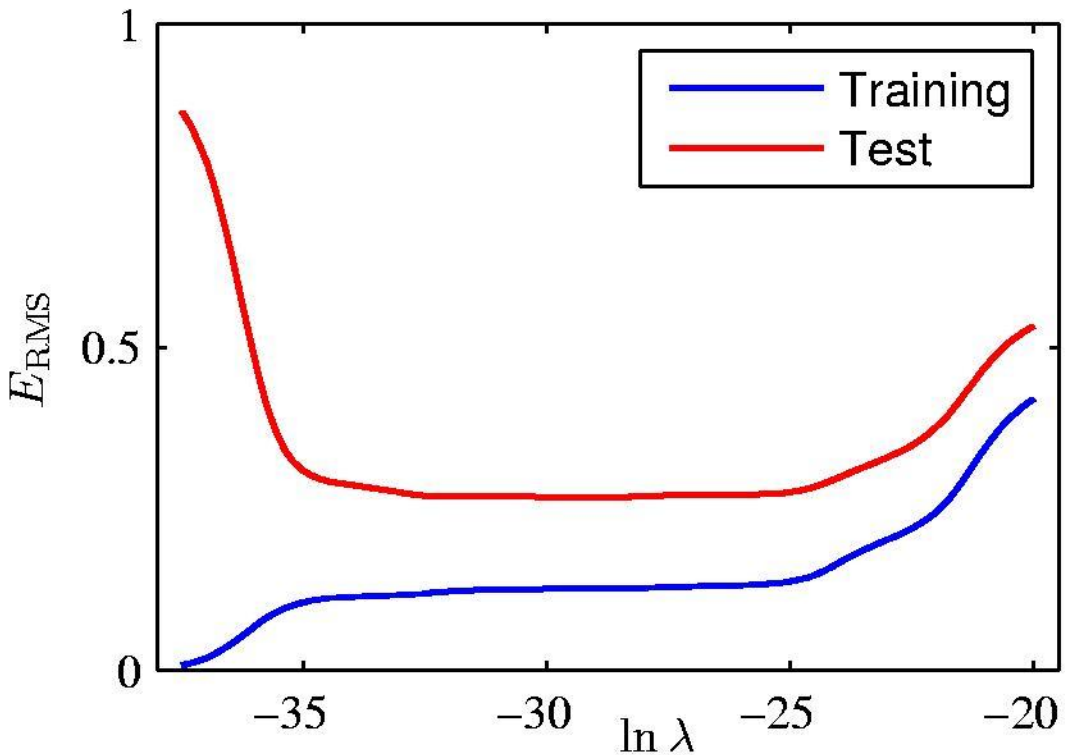
$\lambda \approx 10^{-8}$

Regularization: $\ln \lambda = 0$



$\lambda = 1$

Regularization: E_{RMS} vs. $\ln \lambda$

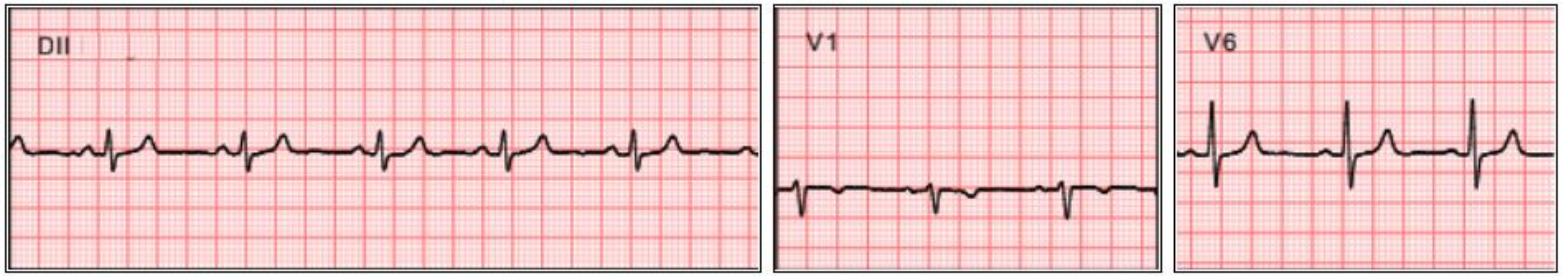


Polynomial Coefficients

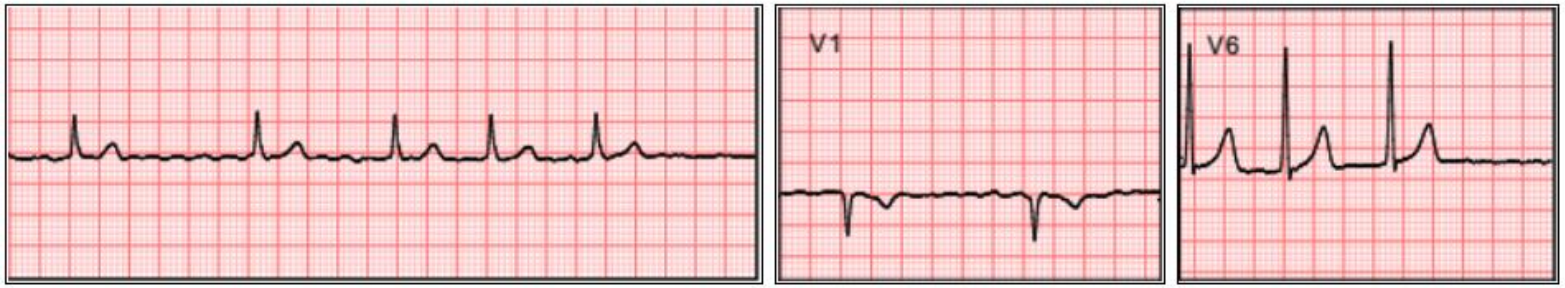
	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Heart Abnormalities

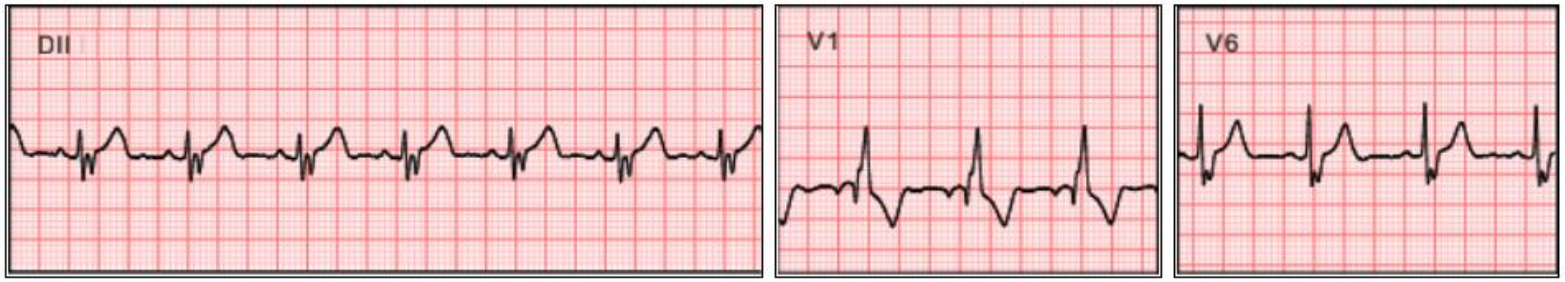
No abnormalities



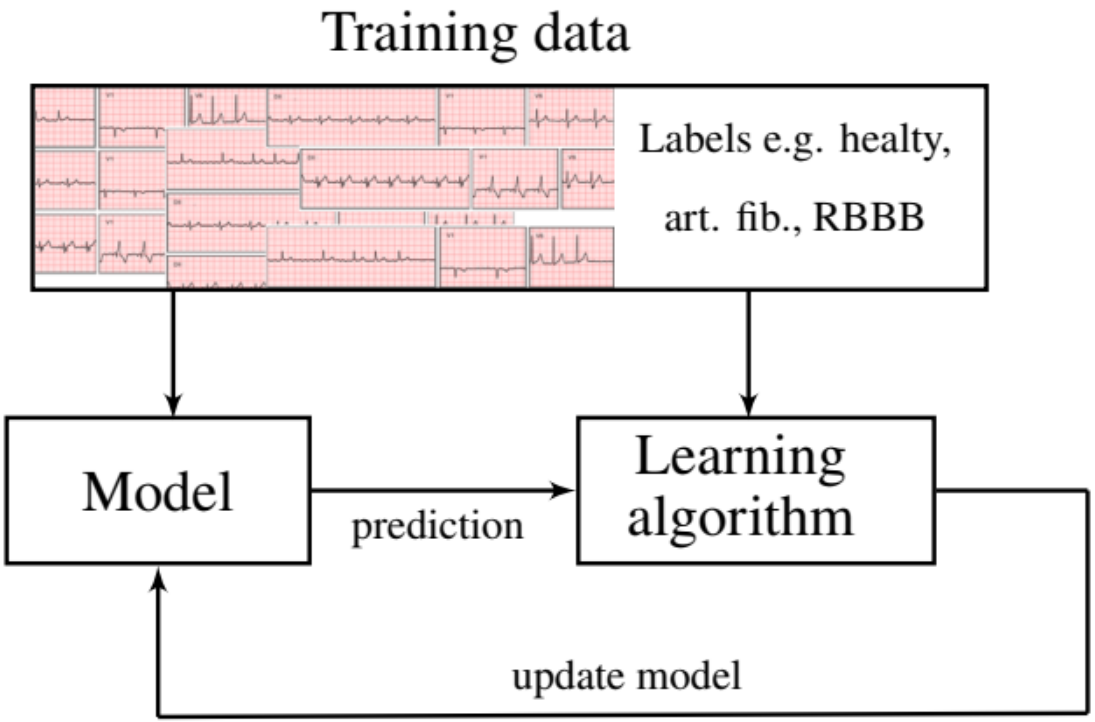
Atrial fibrillation

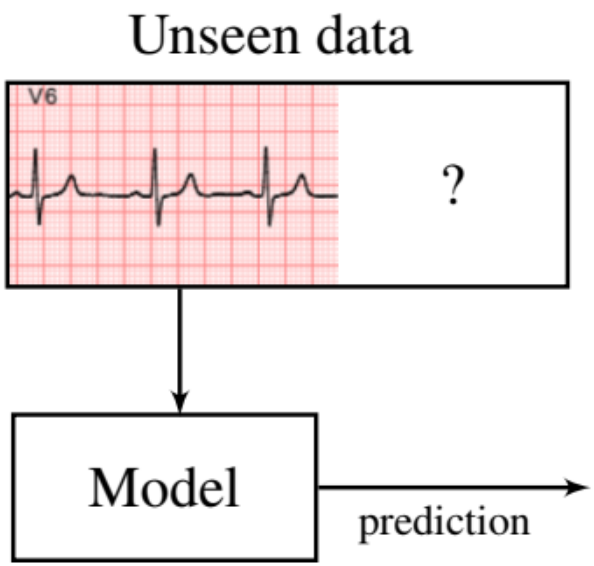


Right bundle branch block

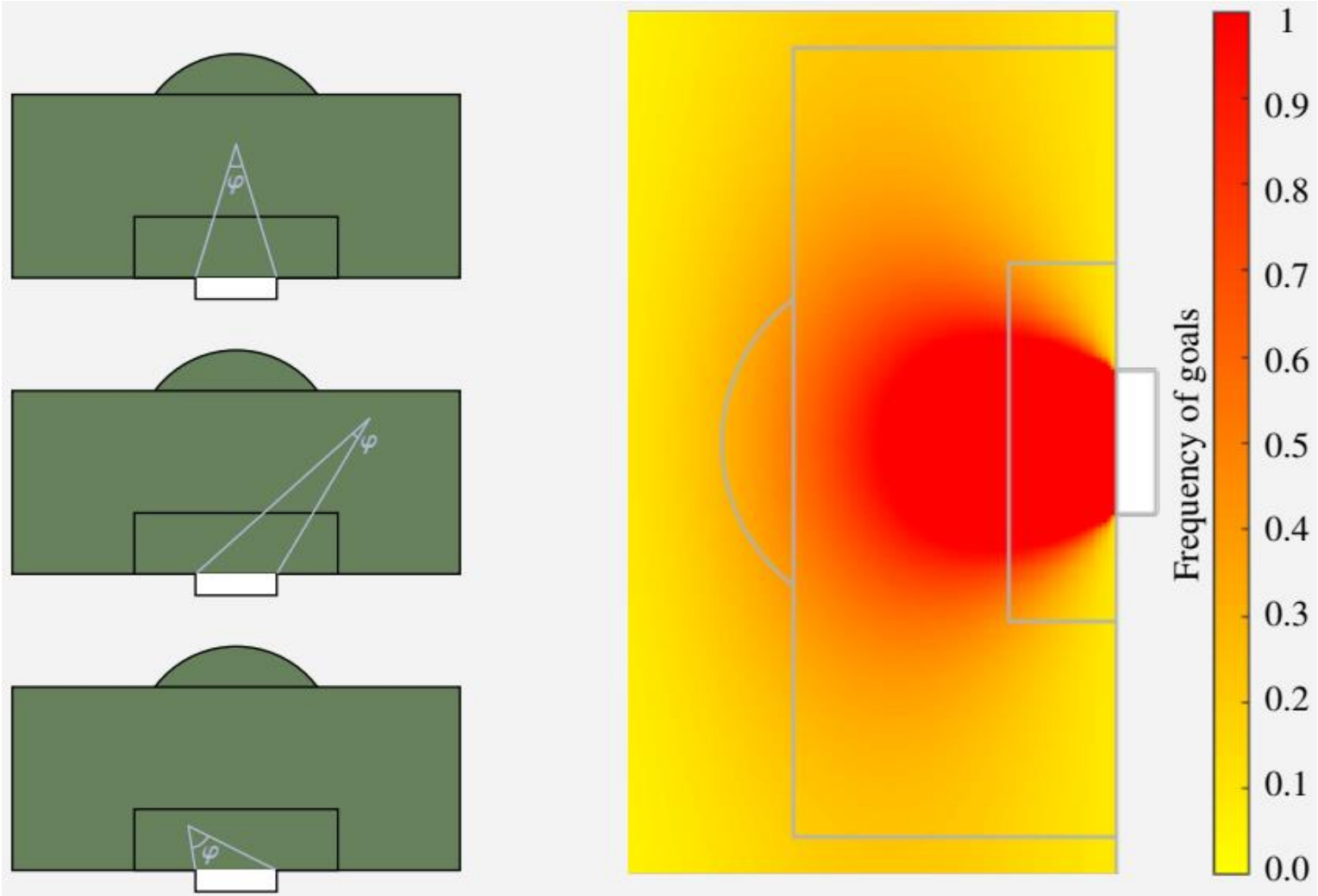


Machine Learning Approach





Predicting the Scoring of Goals



Real-Time Image Segmentation

Segmented Image (cars, signs, road, …)

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 02

Linear Regression

TUC ECE, Spring 2023

Today

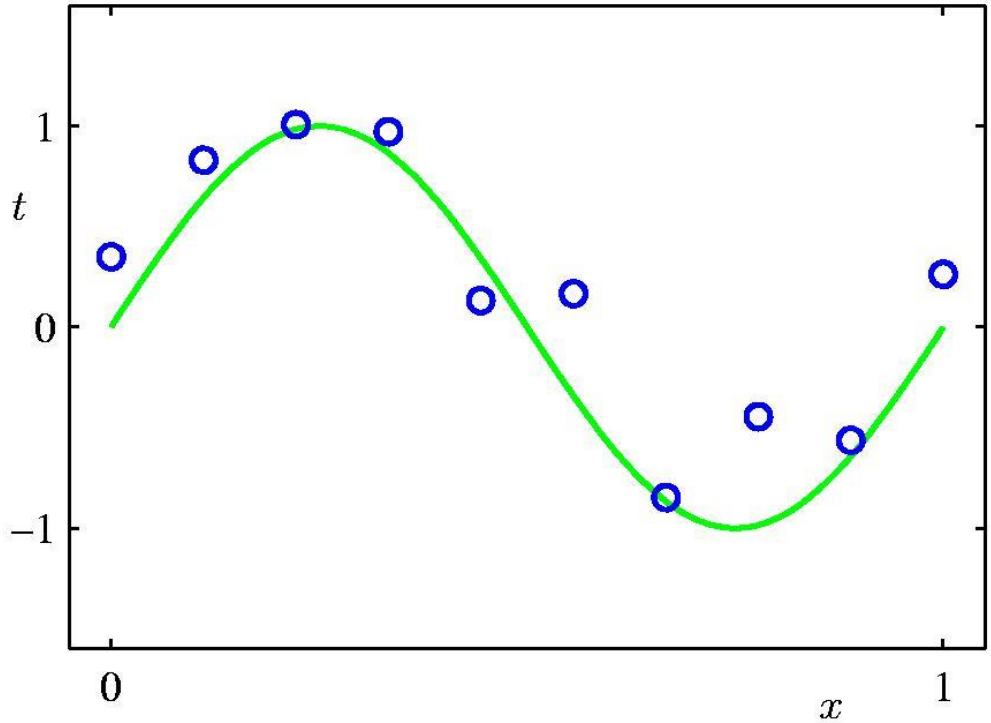
- **Regression**
- **Basis functions**
  - polynomial, Gaussian, sigmoidal
- **ML Least-Squares**
- **Sequential learning**
- **Regularization**
- **Multi-Dimensional Output**

Regression

Regression

- **Given**
  - a training data set of N observations of $\mathbf{x}$: $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$
  - together with the corresponding target values: $\{t_1, t_2, \ldots, t_N\}$
- **Goal**
  - a function $y(\mathbf{x})$ that predicts the target value t for any input $\mathbf{x}$
  - a predictive distribution $p(t|\mathbf{x})$ over values of t for any input $\mathbf{x}$
- **Objective**
  - minimization of a loss function (e.g. the squared loss)
- **Common model choice**
  - linear combinations of (non-linear) basis functions
  - limited in high dimensions, but with nice analytical properties

Example: Polynomial Curve Fitting



$$t = \sin(2\pi x)$$

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Basis Functions

Basis Function Models

- **Linear model**

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

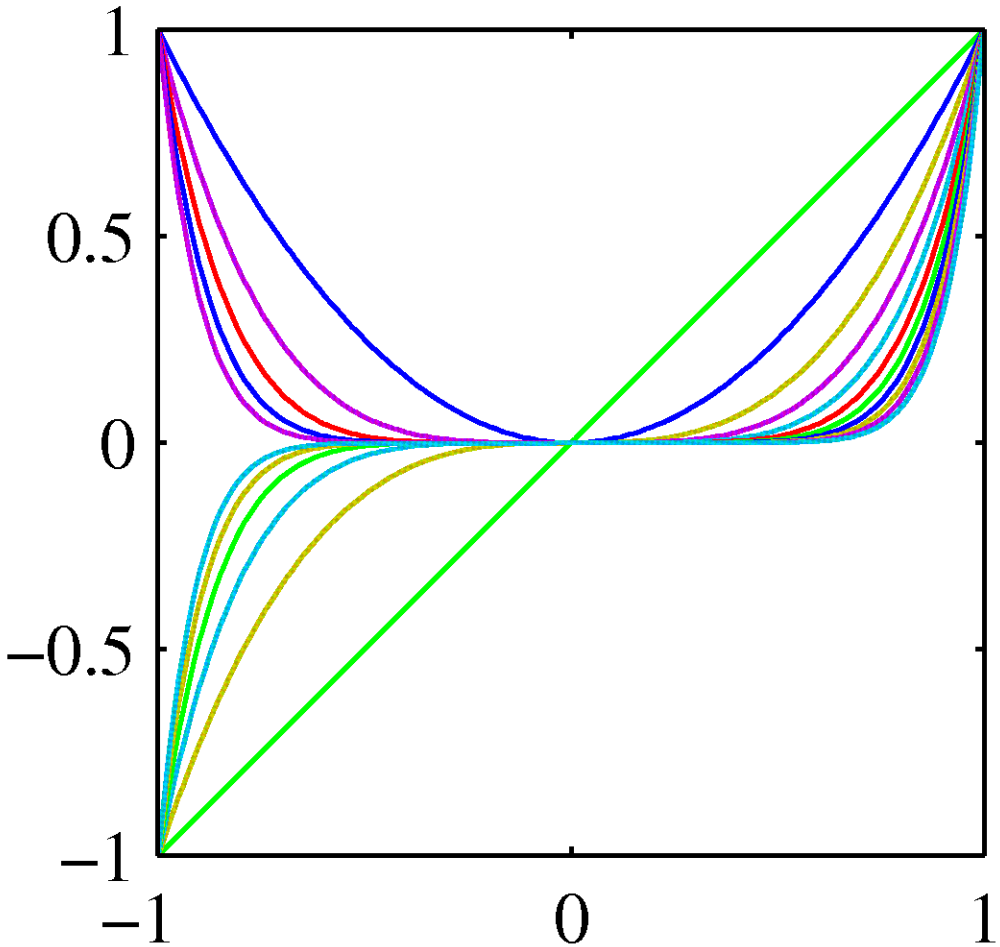
  - $\phi_j(\mathbf{x})$ are known as *basis functions* (may be non-linear)
  - w_j are known as *parameters* or *weights*
- **Bias parameter**
  - typically, $\phi_0(\mathbf{x}) = 1$, so that w_0 acts as a bias (DC component)
- **Simplest case**
  - linear basis functions, one per dimension: $\phi_d(\mathbf{x}) = x_d$
  - imposes significant limitations on the model

Polynomial Basis Functions

$$\phi_j(x) = x^j$$

global: a small change in x affects all basis functions

spline functions: separate regions with different polynomials in each region



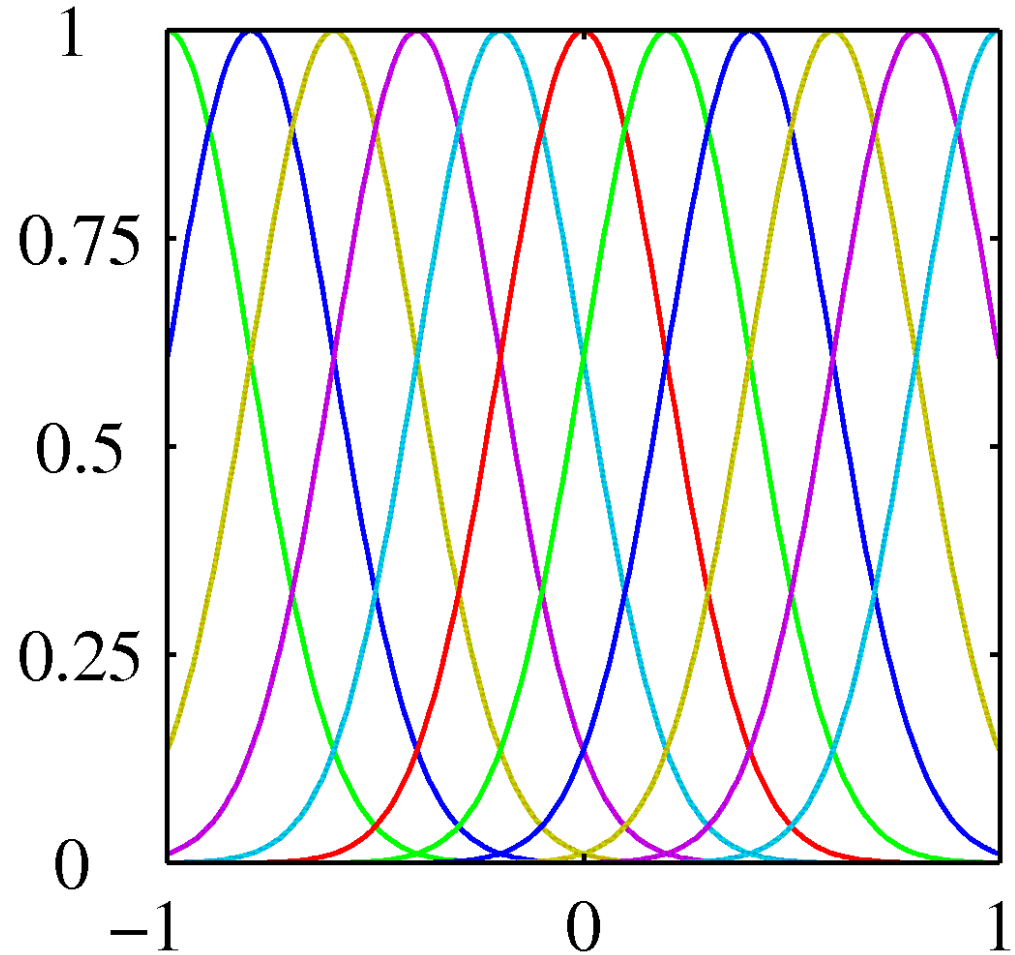
Gaussian (Radial) Basis Functions

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

local: a small change in x only affects nearby basis functions

μ_j's control location

s controls scale (width)



Sigmoidal Basis Functions

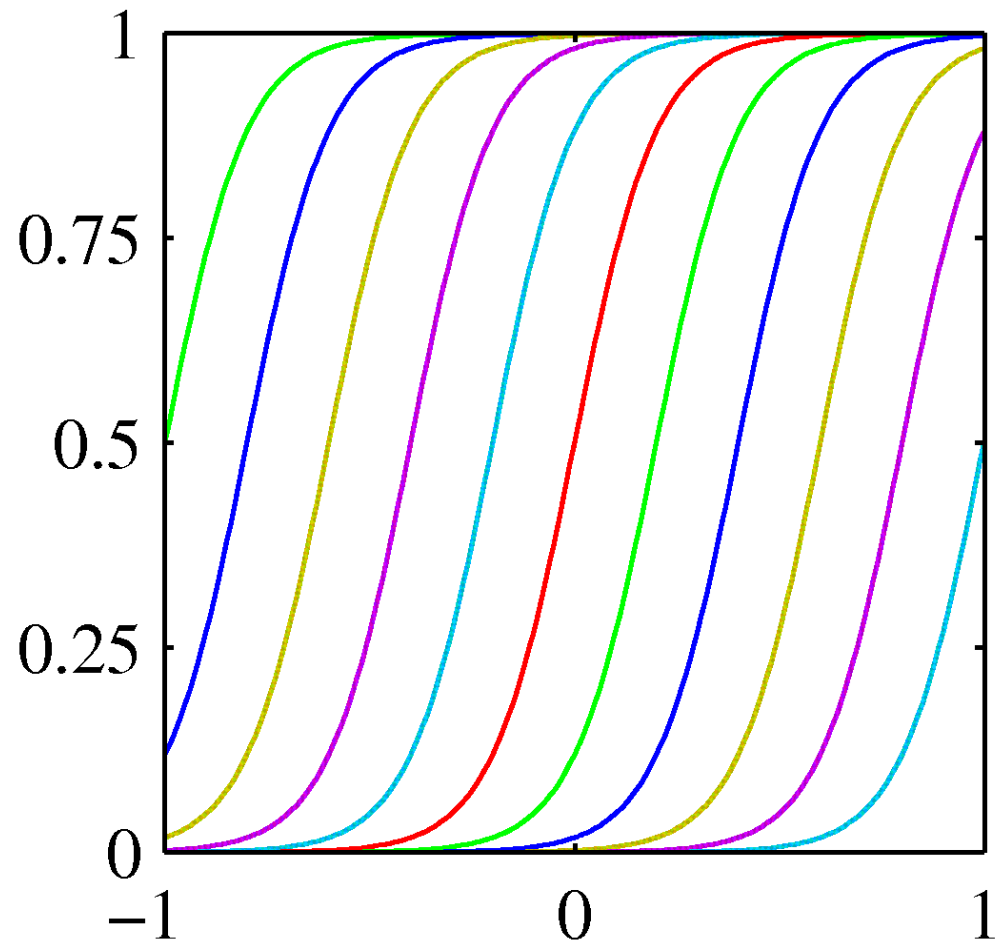
$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

local: a small change in x only affects nearby basis functions

μ_j's control location

s controls scale (slope)

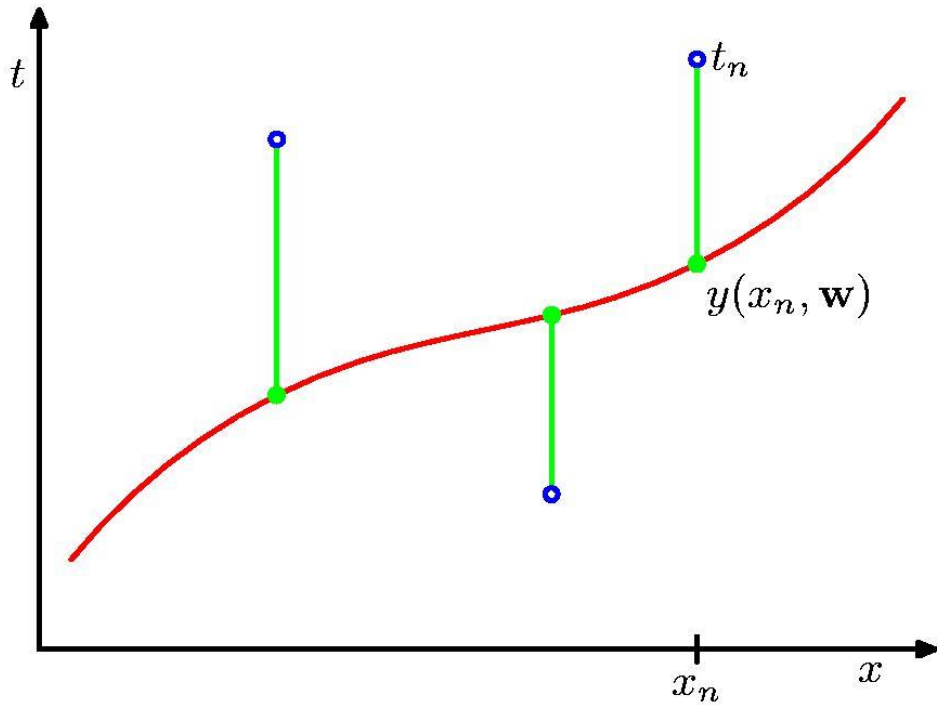


Other Basis Functions

- **Fourier series**
  - expansion in sinusoidal functions
  - each basis function represents a specific frequency
  - each basis function has an infinite spatial extent
- **Wavelets**
  - sinusoidal basis functions
  - localized in both space and frequency
  - mutually orthogonal
  - applicable mainly when the input lives on a regular lattice
    - successive time points in a temporal sequence
    - image pixels

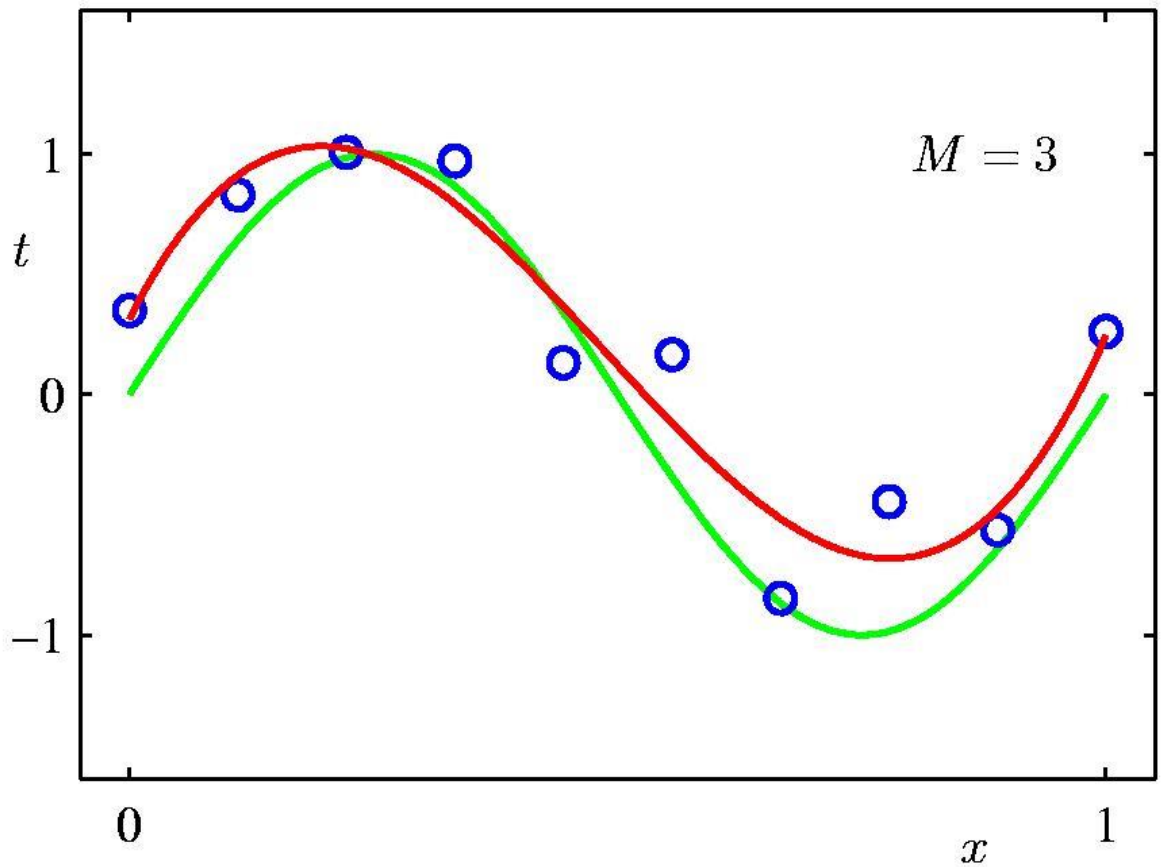
ML Least-Squares

Sum-of-Squares Error Function



$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

Fitting a 3rd Order Polynomial



	$M = 3$
$w_0^\star$	0.31
$w_1^\star$	7.99
$w_2^\star$	-25.43
$w_3^\star$	17.37

Likelihood

- **Observations**
  - assume deterministic function with added Gaussian noise

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \qquad \text{where} \qquad p(\epsilon|\beta) = \mathcal{N}(\epsilon|0, \beta^{-1})$$

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

- **Given**
  - observed inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
  - observed targets $\mathbf{t} = [t_1, \ldots, t_N]^{\mathrm{T}}$
  - basis functions $y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$
- **Likelihood**

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1})$$

Log Likelihood

- taking the logarithm, we get

$$\begin{aligned}
\ln p(\mathbf{t}|\mathbf{w},\beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n),\beta^{-1}) \\
&= \sum_{n=1}^{N} \ln \left(\frac{1}{(2\pi\beta^{-1})^{1/2}} \exp\left\{ -\frac{(t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n))^2}{2\beta^{-1}} \right\} \right) \\
&= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w})
\end{aligned}$$

- where

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

- is the sum-of-squares error

Maximization for w

- setting the gradient to zero

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{w}, \beta) = \beta \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right\} \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} = \mathbf{0}$$

- solving for $\mathbf{w}$

$$\mathbf{w}_{\mathrm{ML}} = \overbrace{\left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}}}^{\text{The Moore-Penrose pseudo-inverse, } \boldsymbol{\Phi}^{\dagger}} \mathbf{t}$$

- design matrix

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

Maximization for the Bias

- maximizing with respect to the bias, w_0, alone

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w_0 - \sum_{j=1}^{M-1} w_j\phi_j(\mathbf{x}_n)\}^2$$

- setting the derivative to zero and solving for w_0

$$\begin{aligned} w_0 &= \bar{t} - \sum_{j=1}^{M-1} w_j\overline{\phi_j} \\ &= \frac{1}{N}\sum_{n=1}^{N} t_n - \sum_{j=1}^{M-1} w_j \frac{1}{N}\sum_{n=1}^{N}\phi_j(\mathbf{x}_n) \end{aligned}$$

- difference between the average of the target values and the weighted sum of the averages of the basis function values

Maximization for β

– maximizing with respect to β

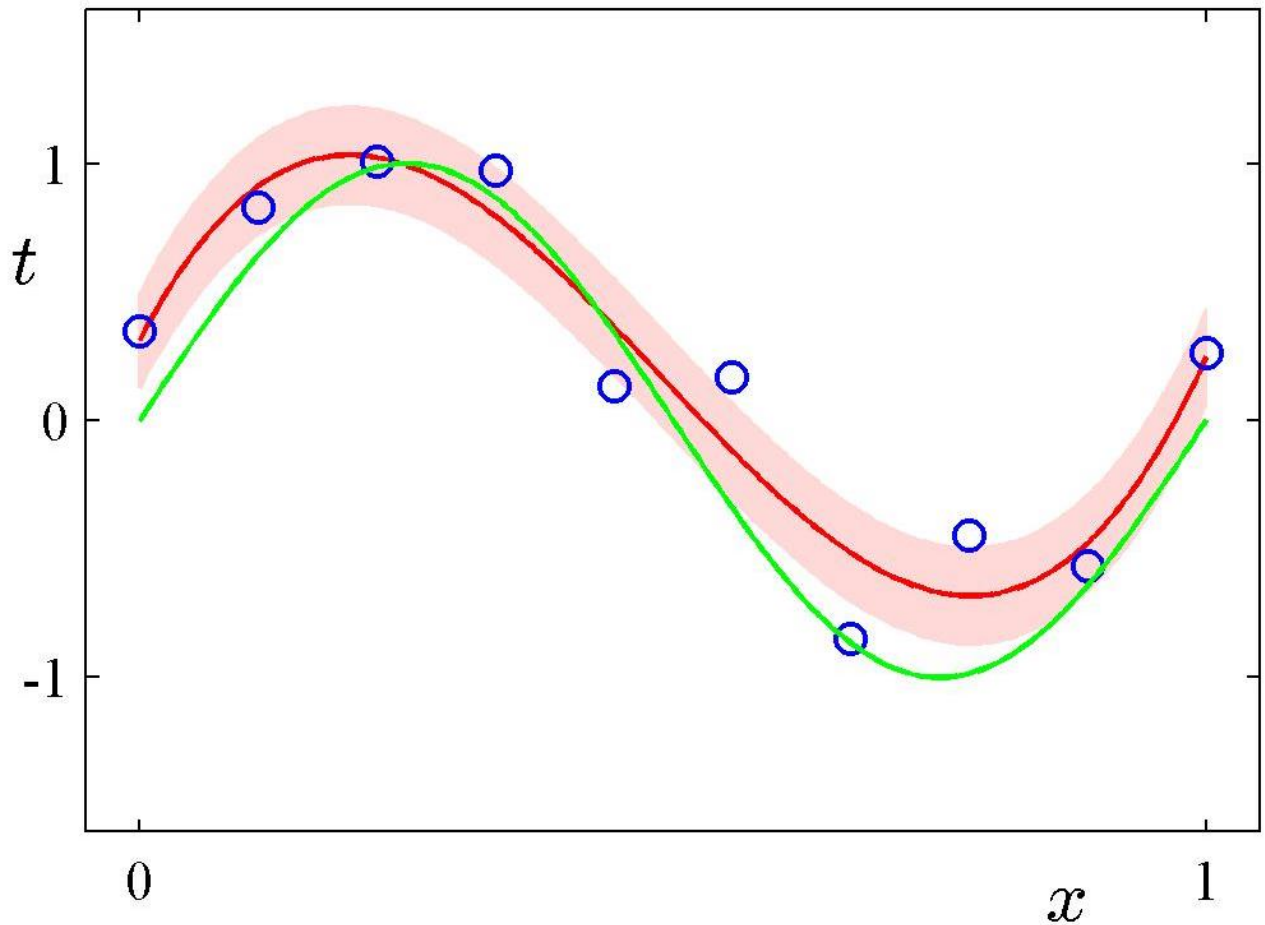
$$\begin{aligned}
\nabla_\beta \ln p(\mathbf{t}|\mathbf{w},\beta) &= \nabla_\beta \left(\frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w}_{\mathrm{ML}}) \right) \\
&= \frac{N}{2\beta} - E_D(\mathbf{w}_{\mathrm{ML}}) \\
&= \frac{N}{2\beta} - \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}_{\mathrm{ML}}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2
\end{aligned}$$

$$\frac{N}{2\beta_{\mathrm{ML}}} - \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}_{\mathrm{ML}}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 = 0$$

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N}\sum_{n=1}^{N}\{t_n - \mathbf{w}_{\mathrm{ML}}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

Predictive Distribution

$$p(t|x, \mathbf{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}}) = \mathcal{N}\left(t|y(x, \mathbf{w}_{\mathrm{ML}}), \beta_{\mathrm{ML}}^{-1}\right)$$



Geometry of Least Squares

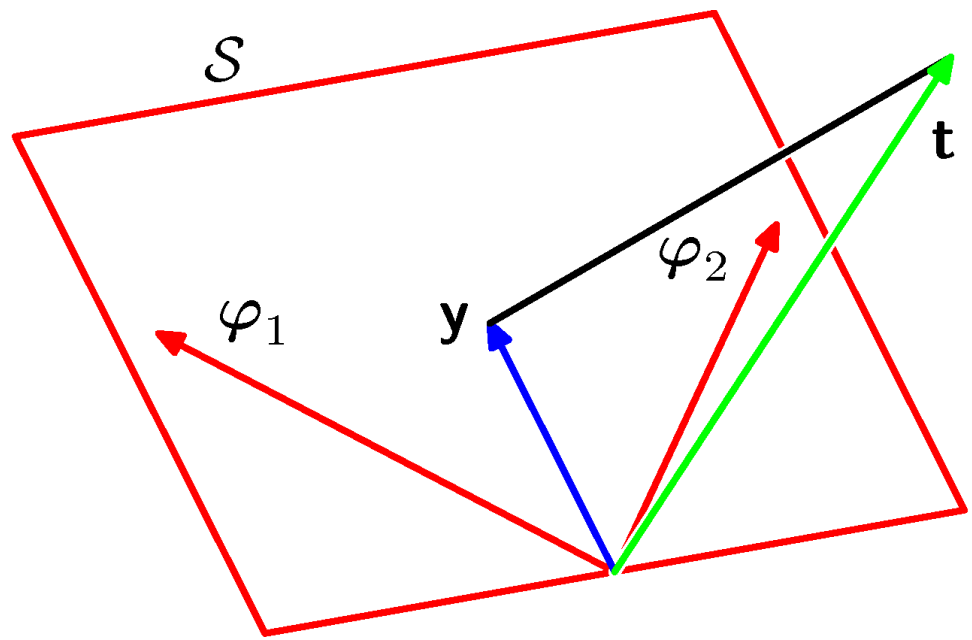
Consider

$$\mathbf{y} = \mathbf{\Phi}\mathbf{w}_{\mathrm{ML}} = [\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_M]\,\mathbf{w}_{\mathrm{ML}}$$

$$\mathbf{y} \in \mathcal{S} \subseteq \mathcal{T} \qquad \mathbf{t} \in \mathcal{T}$$

N-dimensional

M-dimensional



$\mathcal{S}$ is spanned by $\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_M$

$\mathbf{w}_{\mathrm{ML}}$ minimizes the distance between **t** and its orthogonal projection on $\mathcal{S}$, i.e. **y**

Sequential Learning

Sequential Learning

- **Sequentiality**
  - data items considered one at a time (online learning)
- **Approach**
  - use stochastic (sequential) gradient descent (with error E_n)

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla_{\mathbf{w}} E_n$$

  - for the sum-of-squares error function

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta \left(t_n - \mathbf{w}^{(\tau)\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right) \boldsymbol{\phi}(\mathbf{x}_n)$$

  - known as the *least-mean-squares (LMS) algorithm*
- **Issues**
  - iterations? convergence? how to choose ´?

Regularization

Regularized Least Squares

- **Regularized error function**

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

Data term + Regularization term

- **Squared error**
  - sum-of-squares error function
  - quadratic regularizer (weight decay)

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

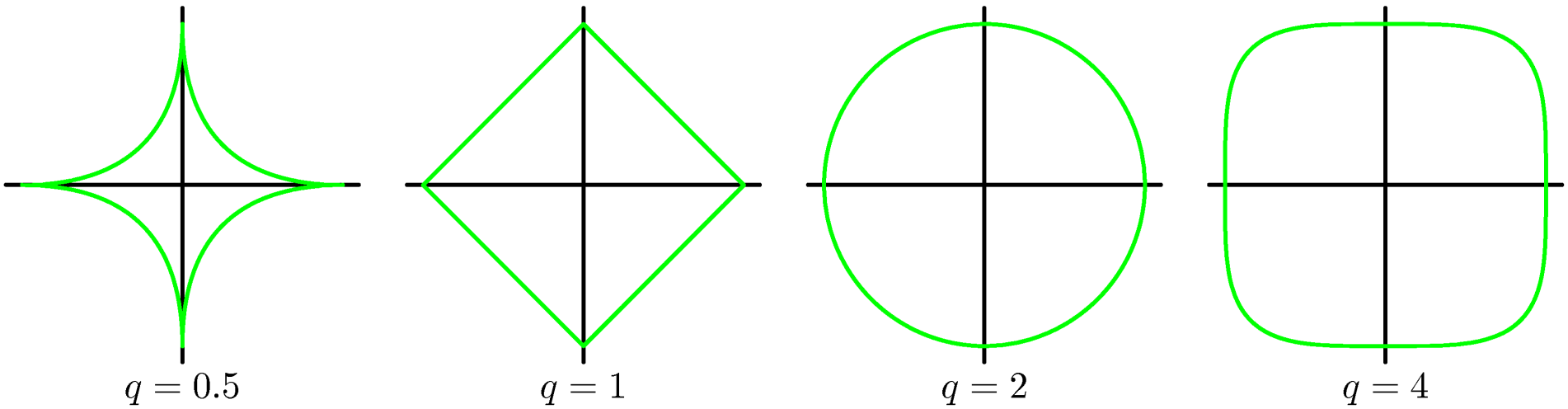
λ is called the regularization coefficient

- **Least squares solution**

$$\mathbf{w} = \left(\lambda\mathbf{I} + \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}$$

General Regularizer

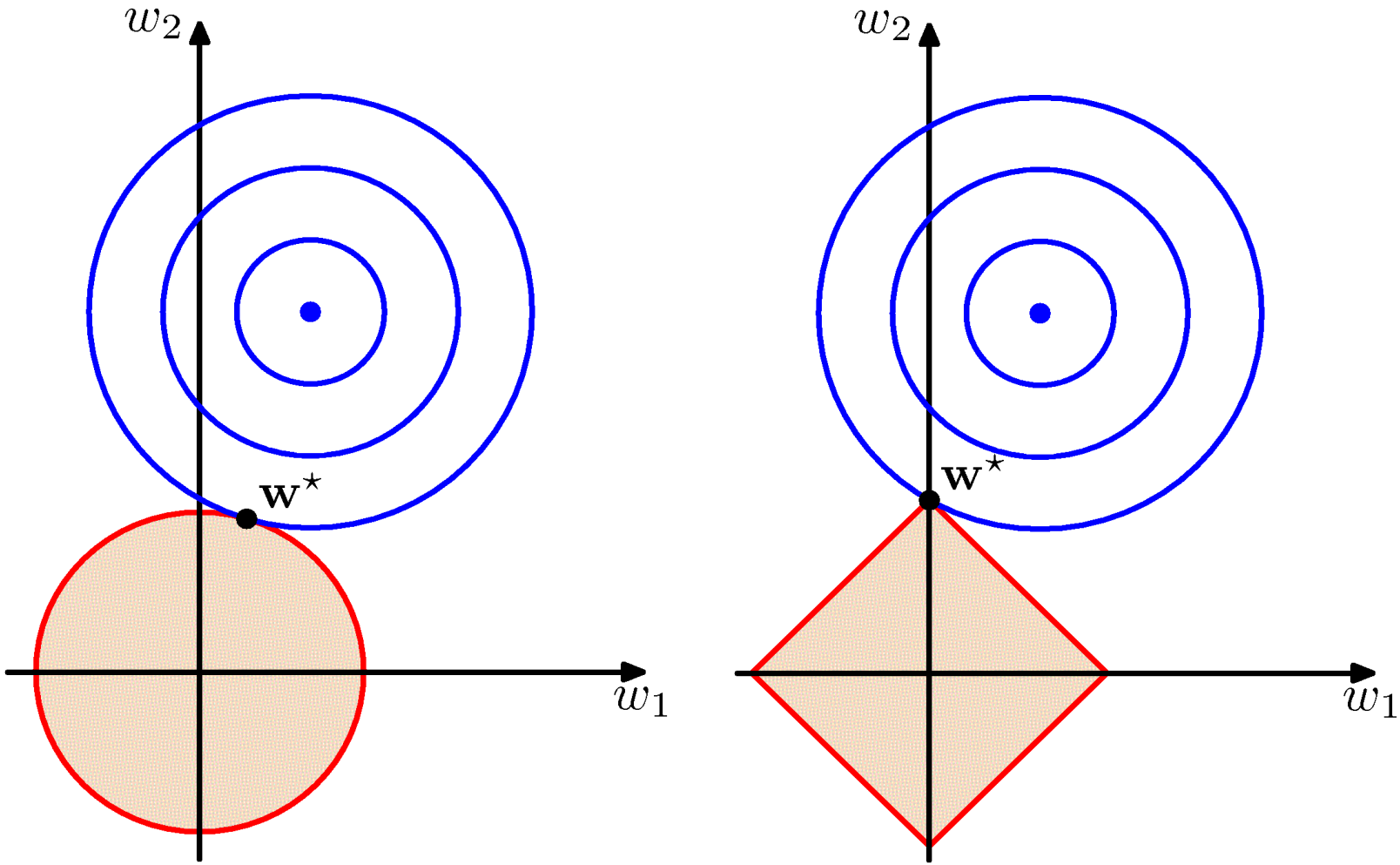
$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$



Lasso

Quadratic

Lasso Sparsity



Multi-Dimensional Output

Multi-Dimensional Output

- **Isotropic Gaussian in K dimensions**

$$p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \beta) = \mathcal{N}(\mathbf{t}|\mathbf{y}(\mathbf{x}, \mathbf{W}), \beta^{-1}\mathbf{I})$$

- **Given**
  - observed inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
  - observed targets $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]^{\mathrm{T}}$
  - basis functions $y(\mathbf{x}, \mathbf{W}) = \mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$
- **Log likelihood function**

$$\begin{aligned}\ln p(\mathbf{T}|\mathbf{X}, \mathbf{W}, \beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(\mathbf{t}_n|\mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}\mathbf{I}) \\ &= \frac{NK}{2}\ln\left(\frac{\beta}{2\pi}\right) - \frac{\beta}{2}\sum_{n=1}^{N}\left\|\mathbf{t}_n - \mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right\|^2\end{aligned}$$

Multi-Dimensional Maximum Likelihood

- **Maximization for $\mathbf{W}$**

$$\mathbf{W}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{T}$$

- **Single target variable t_k**

$$\mathbf{w}_k = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}_k = \boldsymbol{\Phi}^{\dagger}\mathbf{t}_k$$

$$\mathbf{t}_k = [t_{1k}, \ldots, t_{Nk}]^{\mathrm{T}}$$

- **Observations**
  - the solution decouples between the different target variables
  - similarly, for general Gaussian noise distributions

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 03

Bias-Variance Decomposition

TUC ECE, Spring 2023

Today

- **Bias-Variance**
- **Decomposition**

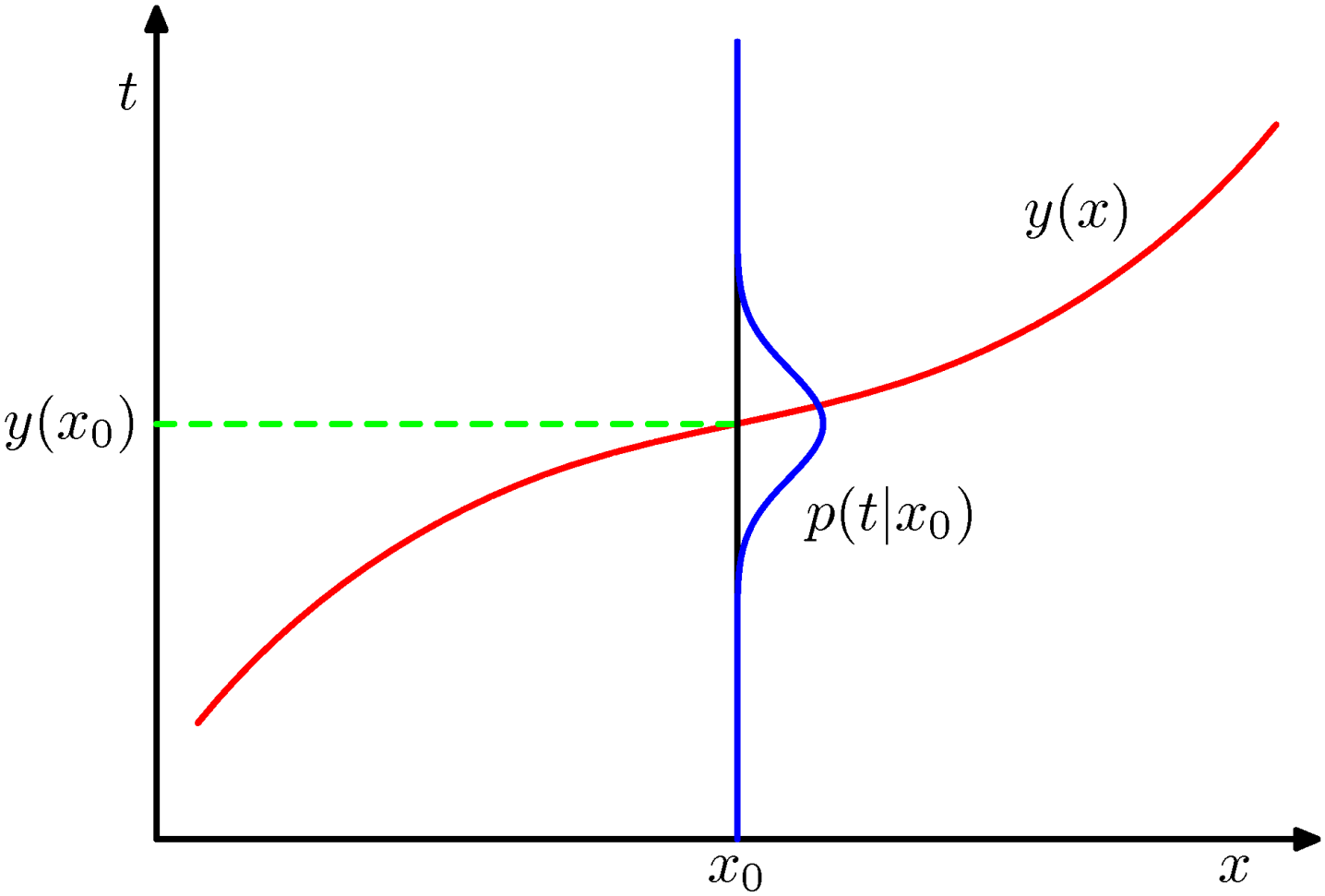
Bias-Variance

Decision Theory for Regression

- **Inference Step**
  - determine $p(\mathbf{x},t)$
- **Decision Step**
  - for any given $\mathbf{x}$, make optimal prediction $y(\mathbf{x})$
- **Loss**
  - loss function $L(t,y(\mathbf{x}))$
- **Expected Loss**

$$\mathbb{E}[L] = \iint L(t, y(\mathbf{x}))p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t$$

Regression Function



The Squared Loss Function

$$\mathbb{E}[L] = \iint \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\, \mathrm{d}\mathbf{x}\, \mathrm{d}t$$

– algebraic manipulation of the square

$$\begin{aligned}\{y(\mathbf{x}) - t\}^2 &= \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}] + \mathbb{E}[t|\mathbf{x}] - t\}^2 \\ &= \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 + 2\{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}\{\mathbb{E}[t|\mathbf{x}] - t\} + \{\mathbb{E}[t|\mathbf{x}] - t\}^2\end{aligned}$$

– substituting back and integrating over t (cross-term vanishes)

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x})\, \mathrm{d}\mathbf{x} + \int \mathrm{var}\,[t|\mathbf{x}]\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

– only the first term depends on $y(\mathbf{x})$; for minimization of loss:

$$y(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}]$$

The Squared Loss Function

$$\mathbb{E}[L] = \iint \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\, \mathrm{d}\mathbf{x}\, \mathrm{d}t$$

$$\begin{aligned}\{y(\mathbf{x}) - t\}^2 &= \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}] + \mathbb{E}[t|\mathbf{x}] - t\}^2 \\ &= \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 + 2\{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}\{\mathbb{E}[t|\mathbf{x}] - t\} + \{\mathbb{E}[t|\mathbf{x}] - t\}^2\end{aligned}$$

$$\mathbb{E}[L] = \underbrace{\int \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x})\, \mathrm{d}\mathbf{x}}_{\text{dependent on prediction}} + \underbrace{\iint \{\mathbb{E}[t|\mathbf{x}] - t\}^2 p(\mathbf{x}, t)\, \mathrm{d}\mathbf{x}\, \mathrm{d}t}_{\text{independent from prediction}}$$

A Closer Look

- **Expected Squared Loss**

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} + \underbrace{\iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t}$$

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x})\,\mathrm{d}t$$

  - The second term of $\mathbb{E}[L]$ corresponds to the noise inherent in the random variable t, is independent of $y(\mathbf{x})$, and represents the minimum achievable expected loss value.
  - $h(\mathbf{x})$ is the optimal prediction (the conditional expectation)
  - What about the first term?

Expectation over Multiple Data Sets

- suppose we were given multiple data sets, each of size N
- any particular data set, $\mathbf{D}$, gives a particular function $y(\mathbf{x};\mathbf{D})$
- let's expand the integrand in the first term

$$
\begin{aligned}
&\{y(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2 \\
&= \{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] + \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 \\
&= \{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2 + \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 \\
&\quad +2\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}
\end{aligned}
$$

- taking the expectation over $\mathbf{D}$, the last term vanishes

$$
\begin{aligned}
&\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2\right] \\
&= \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right]}_{\text{variance}}
\end{aligned}
$$

Bias-Variance Decomposition

The Bias-Variance Decomposition

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

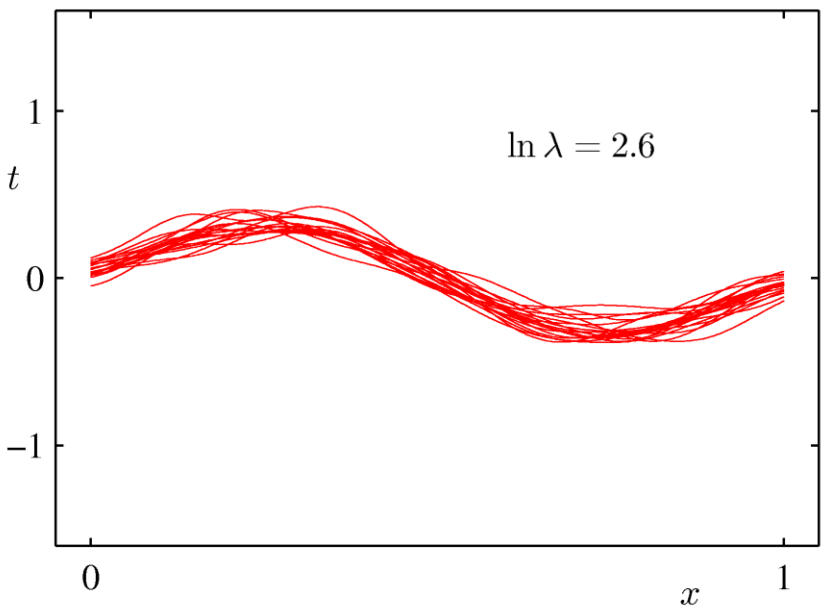
$$\begin{aligned}
(\text{bias})^2 &= \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\
\text{variance} &= \int \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\
\text{noise} &= \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t
\end{aligned}$$

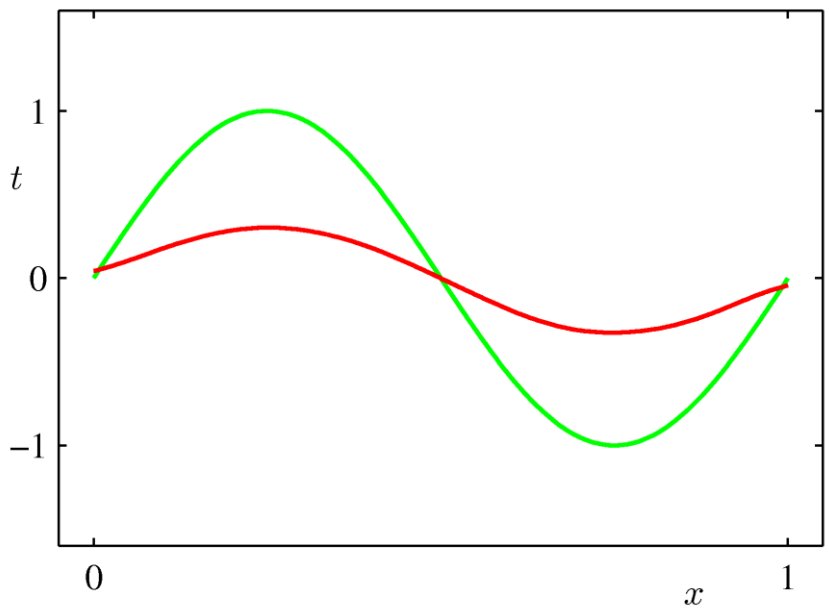
- **bias**: difference of average prediction over all data sets from best
- **variance**: variability of individual set predictions around average

- **Trade-off λ between bias and variance**
  - flexible models: low bias and high variance
  - rigid models: high bias and low variance

Bias-Variance Example (rigid)

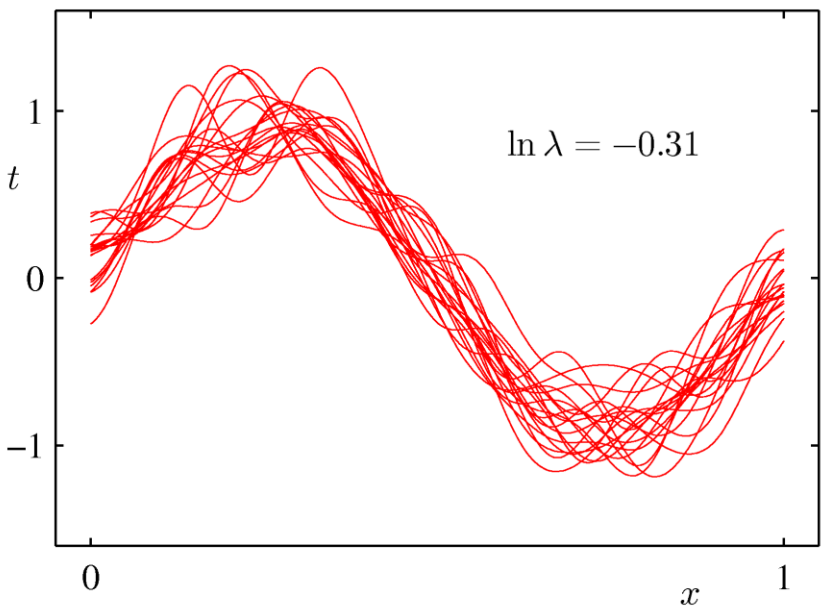
- 100 data sets from $h(x)=\sin(2\pi x)$ with 25 data points each
- model: 25 basis functions (24 Gaussian + 1 bias)
- varying the degree of regularization λ: rigid model

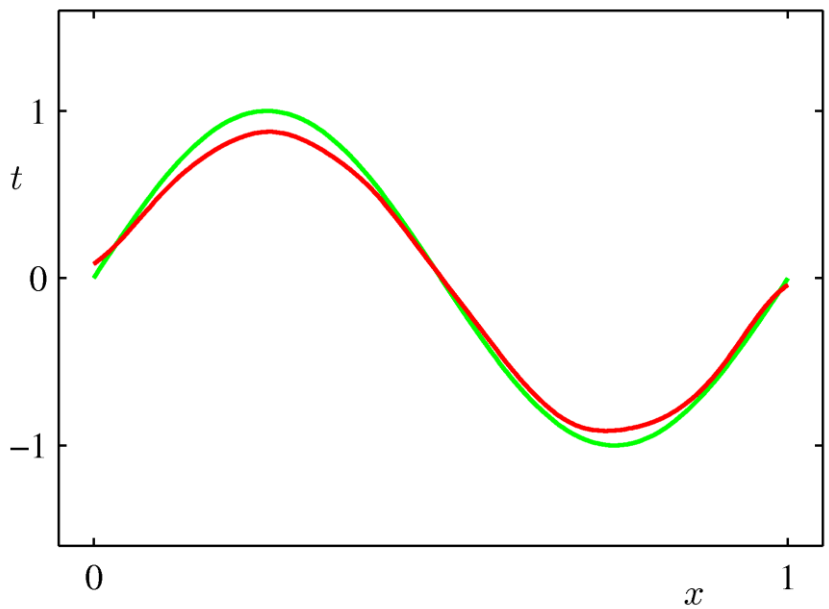




Bias-Variance Example (balanced)

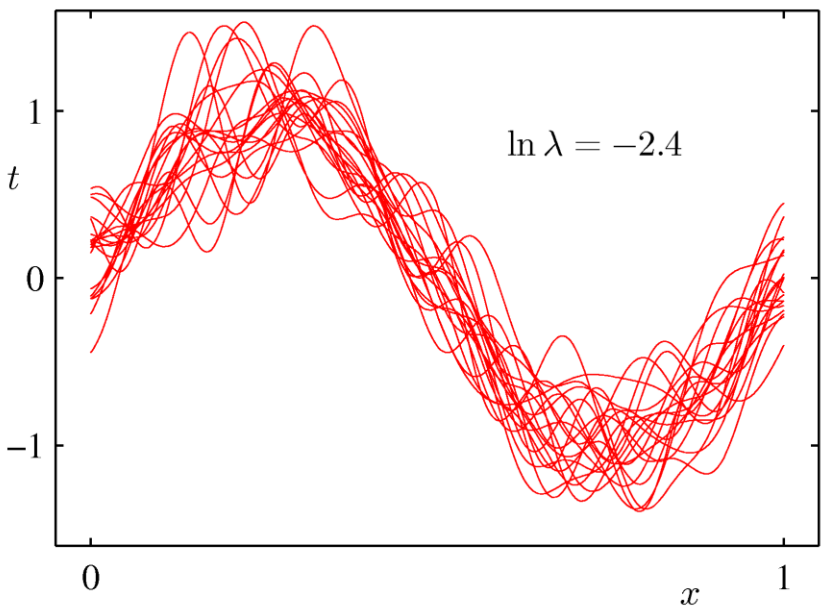
- 100 data sets from $h(x)=\sin(2\pi x)$ with 25 data points each
- model: 25 basis functions (24 Gaussian + 1 bias)
- varying the degree of regularization λ: balanced model

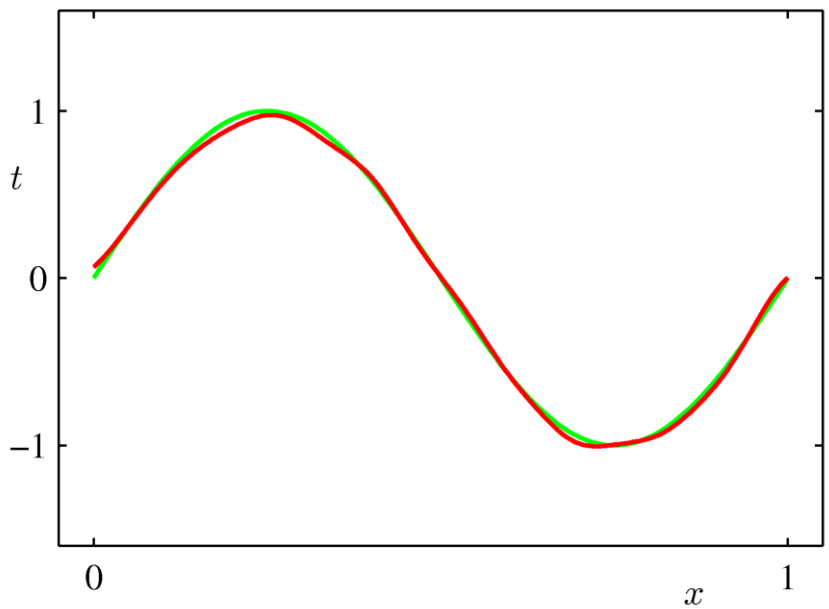




Bias-Variance Example (flexible)

- 100 data sets from $h(x)=\sin(2\pi x)$ with 25 data points each
- model: 25 basis functions (24 Gaussian + 1 bias)
- varying the degree of regularization λ: flexible model





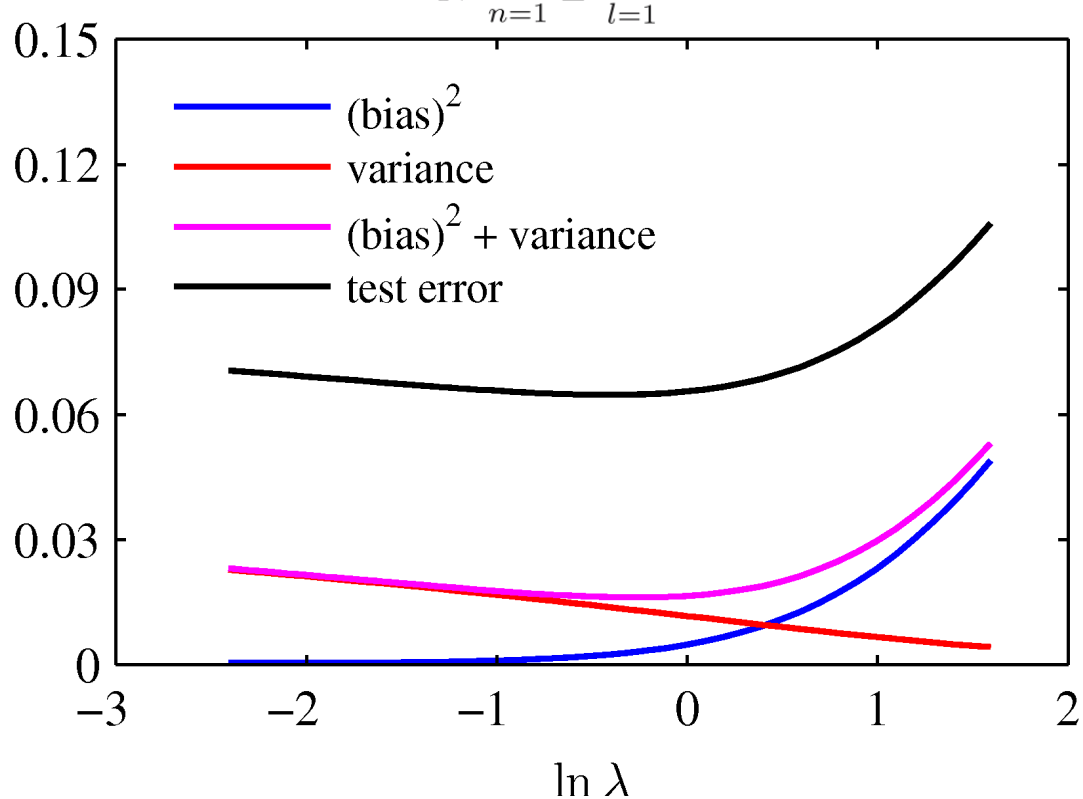
The Bias-Variance Trade-off

$$\overline{y}(x) = \frac{1}{L}\sum_{l=1}^{L} y^{(l)}(x)$$

$$(\text{bias})^2 = \frac{1}{N}\sum_{n=1}^{N}\{\overline{y}(x_n) - h(x_n)\}^2$$

$$\text{variance} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{L}\sum_{l=1}^{L}\{y^{(l)}(x_n) - \overline{y}(x_n)\}^2$$

- **over-regularized model**
  - large λ
  - high bias
  - low variance
- **under-regularized model**
  - small λ
  - low bias
  - high variance



European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 04

Bayesian Linear Regression

TUC ECE, Spring 2023

Today

- **Bayesian Probability**
  - frequentist vs. Bayesian
  - ML Gaussian density estimation
  - ML polynomial curve fitting
  - Bayesian polynomial curve fitting
- **Bayesian Linear Regression**
  - Gaussian prior
  - Bayesian linear regression example
  - Bayesian linear regression predictive distribution

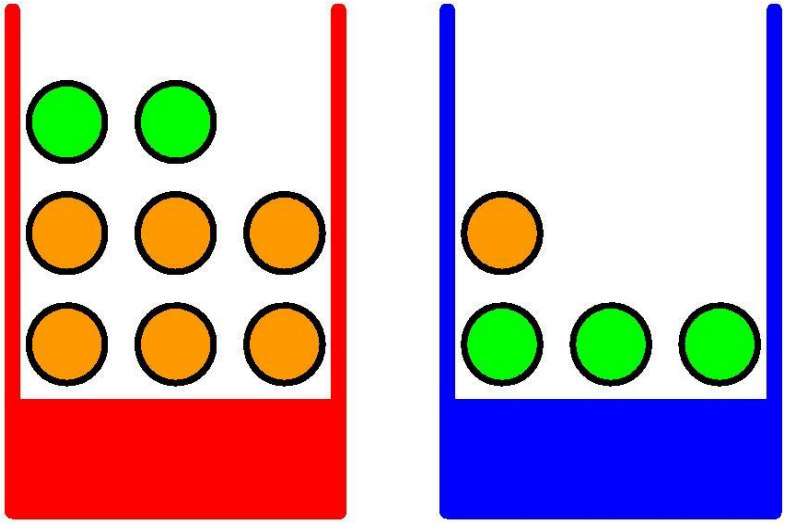
Bayesian Probability

Towards Bayesian Probability

- **Frequentist probability**
  - frequencies of random, repeatable events
- **Considerations**
  - can we reason about rare, non-repeatable events?
  - example: will the Arctic ice cap disappear in 1000 years?
  - what if we have some idea about the rate of ice melting?
  - what if this knowledge is revised after 100 years?
  - need to quantify uncertainty
  - need to make revisions in light of new evidence
- **Bayesian probability**
  - quantification and revision of uncertainty

Bayesian Thinking

Apples and Oranges

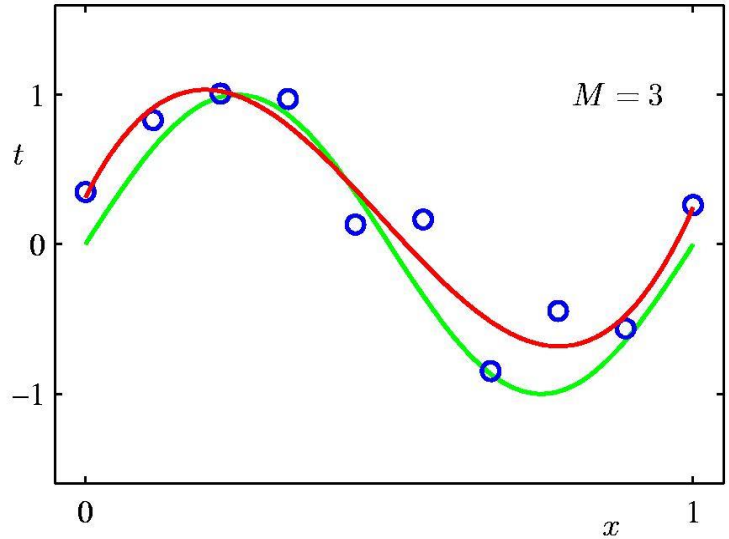


- blue box most likely chosen, before looking at fruit (prior)
- red box most likely chosen, after picking an orange (posterior)

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

Bayesian Thinking

Polynomial Curve Fitting



- what if we start with a belief about **w**? (prior)
- what if we update our belief, after given the data? (posterior)

$$p(\mathbf{w}|D) = \frac{p(D|\mathbf{w})p(\mathbf{w})}{p(D)}$$

posterior $\propto$ likelihood × prior

Likelihood Function

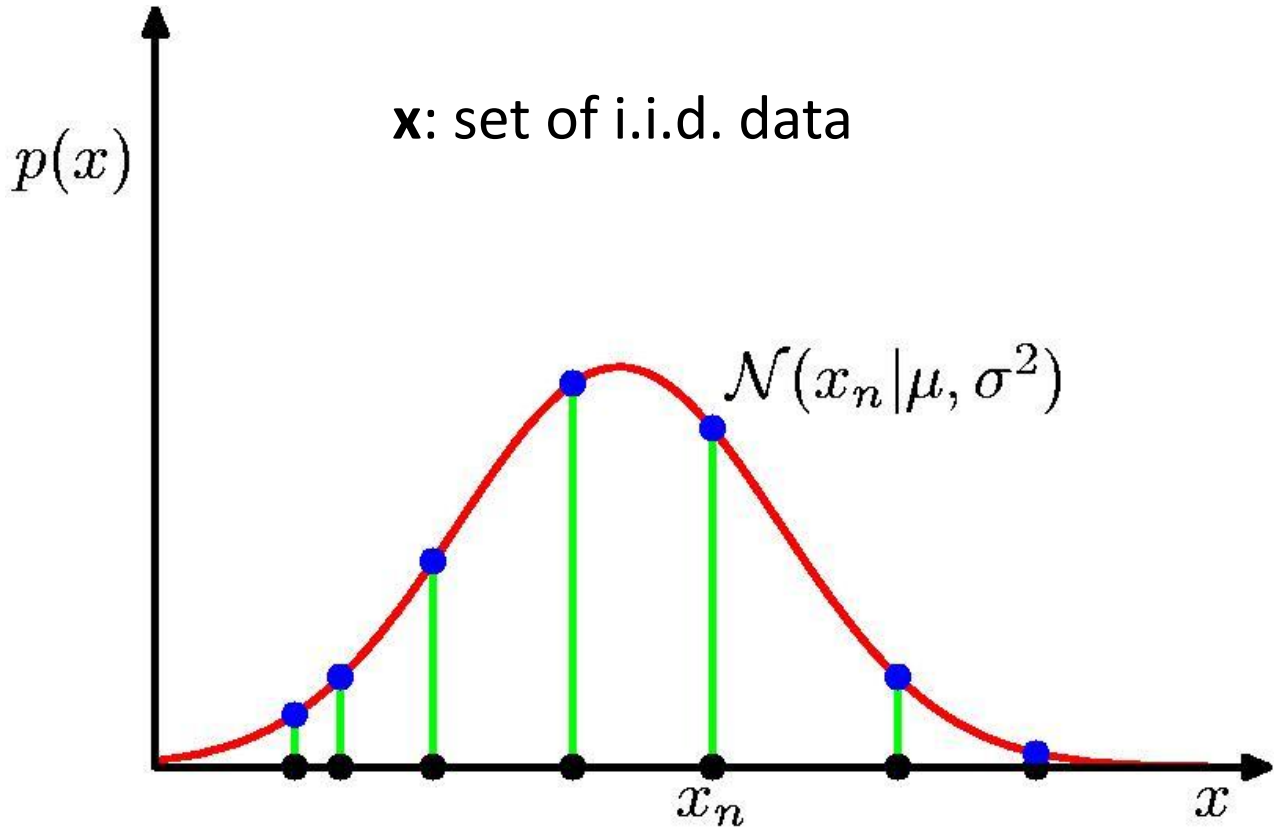
$$p(D|\mathbf{w})$$

- **Likelihood function**
  - how like are some data *D*, given the parameters **w**
  - not a probability distribution over **w**!
- **Frequentist view**
  - **w** is a fixed parameter, whose value is estimated
  - uncertainty as error bars from possible data sets *D*
- **Bayesian view**
  - the observed data set *D* is the most probable data set
  - uncertainty as a probability distribution over **w**

Frequentists vs. Bayesians

- **Frequentist approach**
  - maximum likelihood estimator for maximizing likelihood of data
  - or minimizing error function (negative log of the likelihood)
  - bootstrap method for variability in parameter estimation
  - predictions may be misled (e.g. three coin tosses, all heads)
- **Bayesian approach**
  - makes smoother predictions (e.g. three coin tosses, all heads)
  - incorporates knowledge in the prior (truth or convenience?)
  - reducing dependence on the prior (non-informative priors)
  - difficulty: marginalization over the entire parameter space
  - recent advancement: sampling/approximation methods

Gaussian Parameter Estimation



x: set of i.i.d. data

Likelihood function

$$p(\mathbf{x}|\mu, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}\left(x_n|\mu, \sigma^2\right)$$

Maximum (Log) Likelihood

- **Likelihood**

$$p(\mathbf{x}|\mu,\sigma^2) = \prod_{n=1}^{N} \mathcal{N}\left(x_n|\mu,\sigma^2\right) \qquad \mathcal{N}\left(x|\mu,\sigma^2\right) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

- **Log likelihood**

$$\ln p\left(\mathbf{x}|\mu,\sigma^2\right) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)$$

  – differentiate with respect to μ and σ and set to 0 for maximum

$$\mu_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N} x_n \qquad \sigma^2_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu_{\mathrm{ML}})^2$$

the sample mean — the sample variance

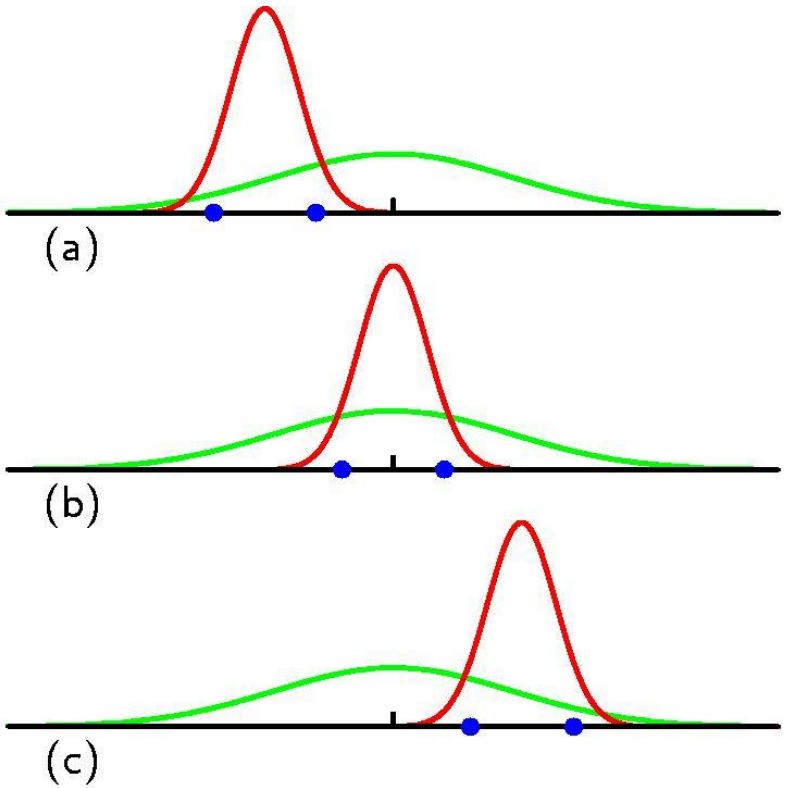
Properties of μ_{ML} and σ^2_{ML}

$$\mathbb{E}[\mu_{\mathrm{ML}}] = \mu$$

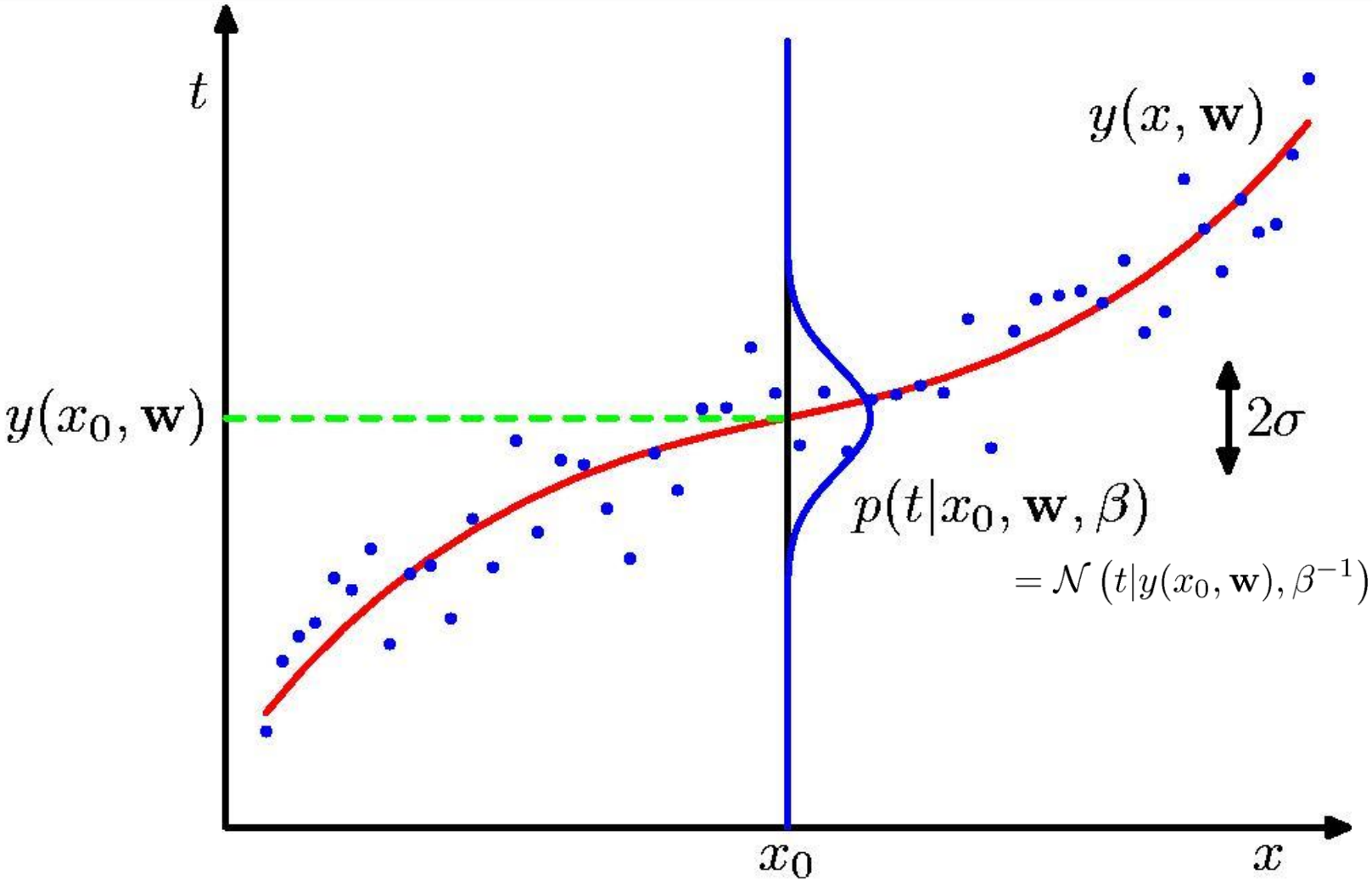
$$\mathbb{E}[\sigma^2_{\mathrm{ML}}] = \left(\frac{N-1}{N}\right)\sigma^2$$

$$\begin{aligned}\widetilde{\sigma}^2 &= \frac{N}{N-1}\sigma^2_{\mathrm{ML}} \\ &= \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \mu_{\mathrm{ML}})^2\end{aligned}$$

Unbiased variance estimate



Curve Fitting Re-visited



ML Curve Fitting

- **Likelihood**

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n | y(x_n, \mathbf{w}), \beta^{-1}\right)$$

- **Log likelihood**

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \underbrace{-\frac{\beta}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2}_{\beta E(\mathbf{w})} + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi)$$

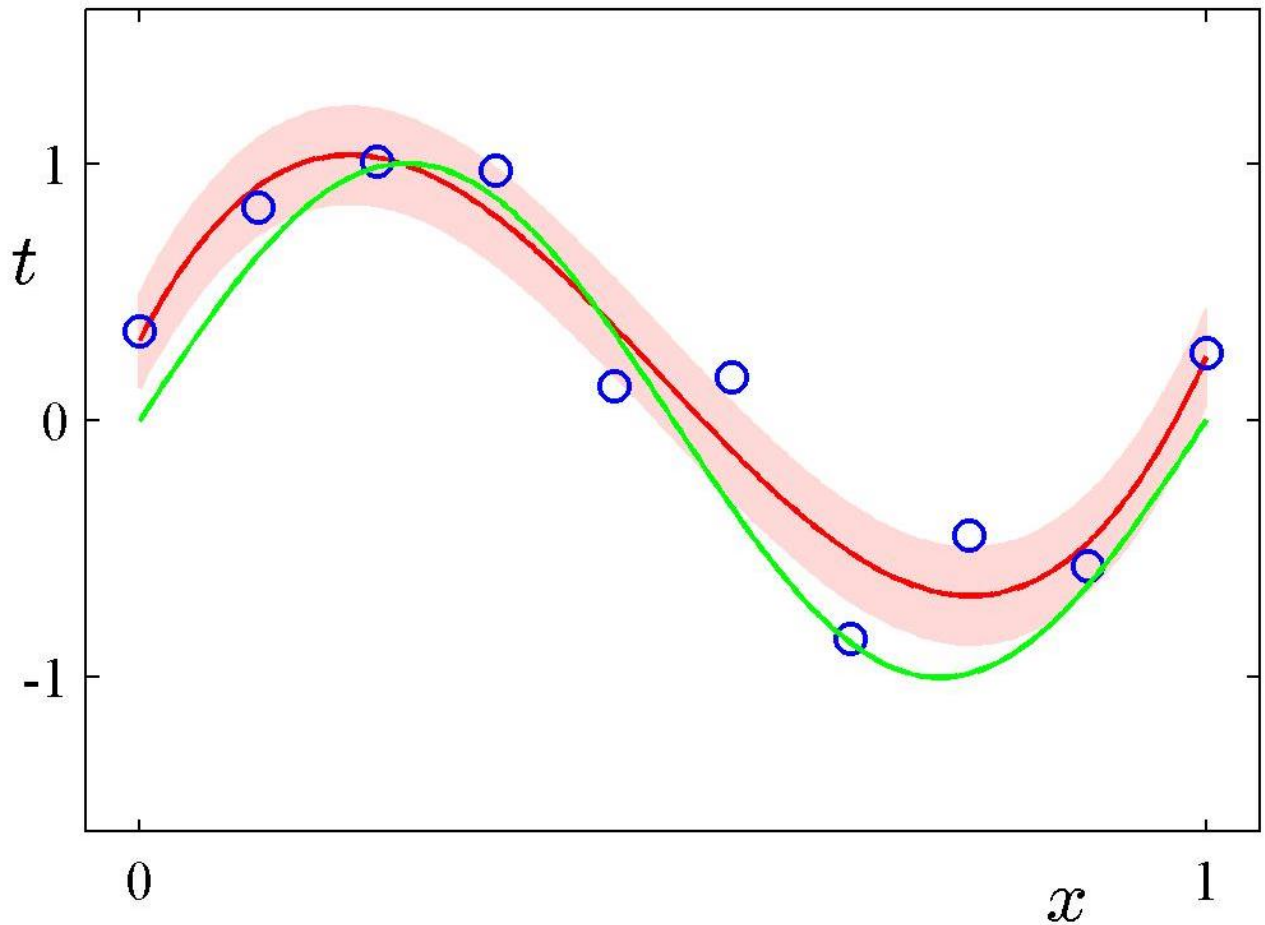
Determine $\mathbf{w}_{\mathrm{ML}}$ by minimizing sum-of-squares error, $E(\mathbf{w})$

Replace $\mathbf{w}_{\mathrm{ML}}$ and determine β_{ML}

$$\mathbf{w}_{\mathrm{ML}} = \arg\min_{\mathbf{w}} E(\mathbf{w}) \qquad \frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}_{\mathrm{ML}}) - t_n\}^2$$

Predictive Distribution

$$p(t|x, \mathbf{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}}) = \mathcal{N}\left(t|y(x, \mathbf{w}_{\mathrm{ML}}), \beta_{\mathrm{ML}}^{-1}\right)$$



MAP: A Step towards Bayes

- **Prior**

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp\left\{-\frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}\right\}$$

- **Likelihood**

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|y(x_n, \mathbf{w}), \beta^{-1}\right)$$

- **Posterior**

$$p(\mathbf{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)$$

- **Maximum posterior**
  - equivalently, minimize the negative log of the posterior

$$\beta\widetilde{E}(\mathbf{w}) = \frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

Determine $\mathbf{w}_{\mathrm{MAP}}$ by minimizing the regularized sum-of-squares error, $\widetilde{E}(\mathbf{w})$

Bayesian Curve Fitting

Given: points **x**, targets **t** Goal: predict t at point x

Sum out **w** (to avoid a point estimate) for a fully Bayesian treatment

$$p(t|x, \mathbf{x}, \mathbf{t}, \alpha, \beta) = \int p(t|x, \mathbf{w}, \beta) p(\mathbf{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta)\, \mathrm{d}\mathbf{w} = \mathcal{N}\left(t|m(x), s^2(x)\right)$$

The predictive distribution is Gaussian and can be found analytically

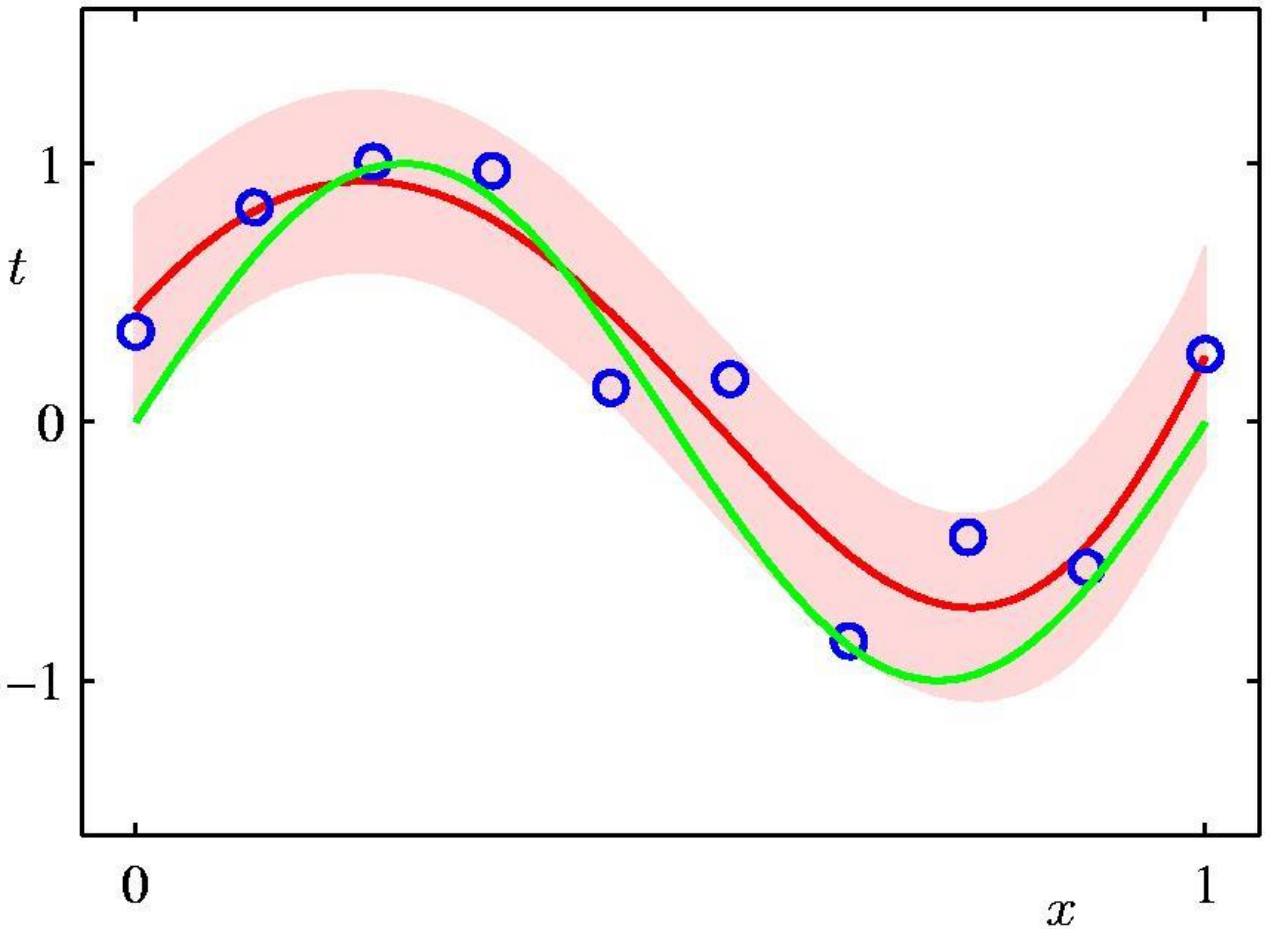
$$m(x) = \beta \boldsymbol{\phi}(x)^{\mathrm{T}} \mathbf{S} \sum_{n=1}^{N} \boldsymbol{\phi}(x_n) t_n \qquad s^2(x) = \beta^{-1} + \boldsymbol{\phi}(x)^{\mathrm{T}} \mathbf{S} \boldsymbol{\phi}(x)$$

$$\mathbf{S}^{-1} = \alpha \mathbf{I} + \beta \sum_{n=1}^{N} \boldsymbol{\phi}(x_n) \boldsymbol{\phi}(x_n)^{\mathrm{T}} \qquad \boldsymbol{\phi}(x_n) = \left(x_n^0, \ldots, x_n^M\right)^{\mathrm{T}}$$

Both the mean and the variance depend in the query point x

Bayesian Predictive Distribution

$$p(t|x, \mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}\left(t|m(x), s^2(x)\right)$$



$M = 9$

$\alpha = 5 \times 10^{-3}$

$\beta = 11.1$

Bayesian Linear Regression

Bayesian Linear Regression (general)

- **Conjugate prior over w**

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$$

- **Likelihood**

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) = \mathcal{N}(\mathbf{t}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I})$$

- **Posterior**

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

$$\begin{aligned} \mathbf{m}_N &= \mathbf{S}_N \left(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}\right) \\ \mathbf{S}_N^{-1} &= \mathbf{S}_0^{-1} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} \end{aligned}$$

- **Reminder: Bayes' Theorem for Gaussians**

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right) \\ p(\mathbf{y}|\mathbf{x}) &= \mathcal{N}\left(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}\right) \end{aligned}$$

$$\begin{aligned} p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \\ p(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \\ \boldsymbol{\Sigma} &= (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1} \end{aligned}$$

Bayesian Linear Regression (specific)

- **Common conjugate prior over w**

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

- **Likelihood**

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) = \mathcal{N}(\mathbf{t}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I})$$

- **Posterior**

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

$$\begin{aligned} \mathbf{m}_N &= \beta\mathbf{S}_N\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha\mathbf{I} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} \end{aligned}$$

- **Reminder: General case**

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

$$\begin{aligned} \mathbf{m}_N &= \mathbf{S}_N\left(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}\right) \\ \mathbf{S}_N^{-1} &= \mathbf{S}_0^{-1} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} \end{aligned}$$

Bayesian Linear Regression Example

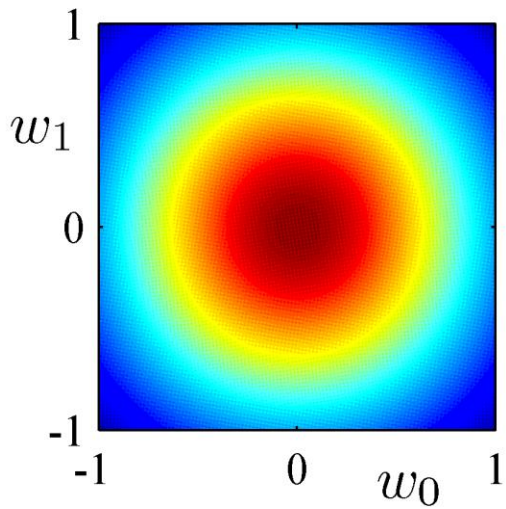
$$y(x, \mathbf{w}) = w_0 + w_1 x$$

0 data points observed

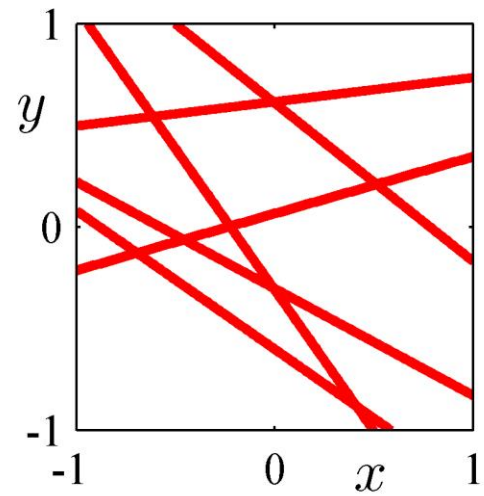
$$t_i = -0.3 + 0.5x_i + \mathcal{N}(0, 0.2)$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$
$$= \mathcal{N}(\mathbf{w}|\mathbf{0}, \frac{1}{2}\mathbf{I})$$

Prior

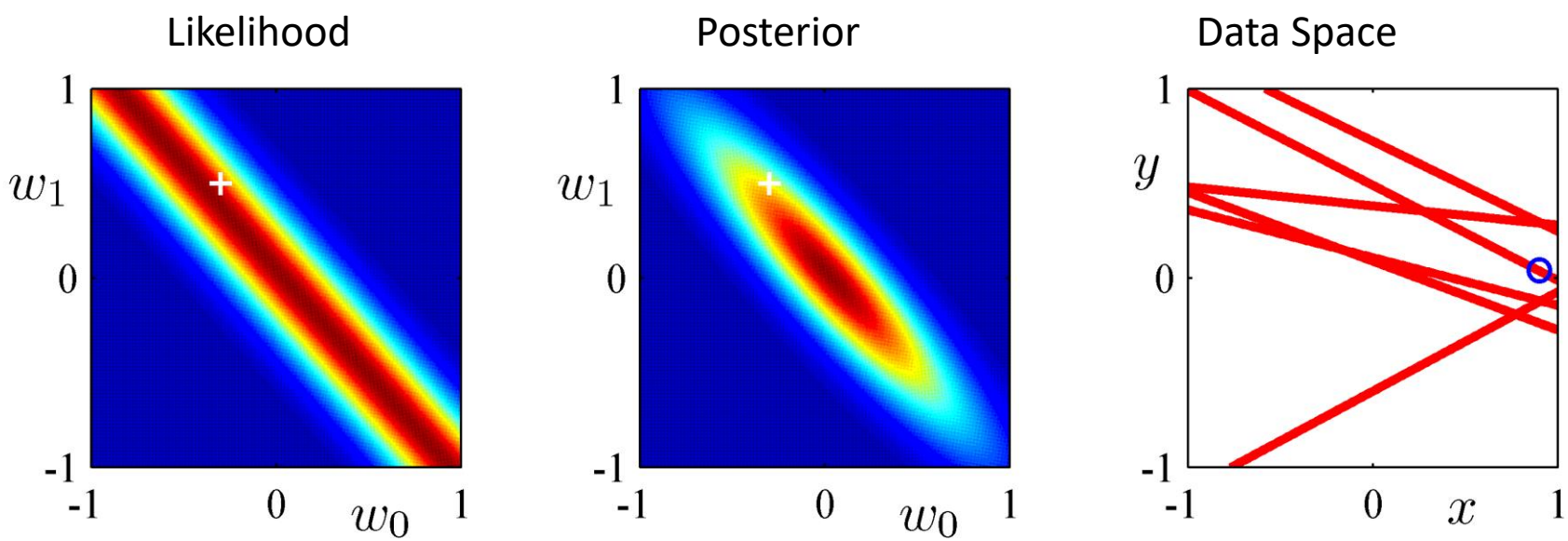


Data Space



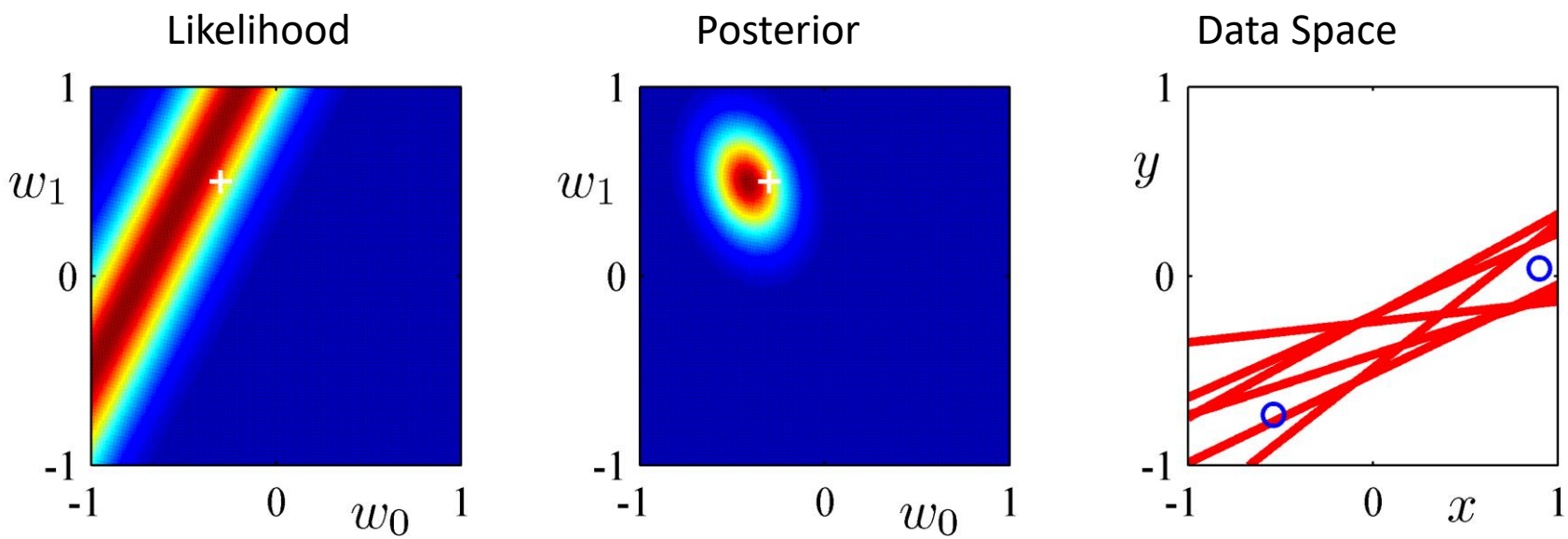
Bayesian Linear Regression Example

1 data point observed



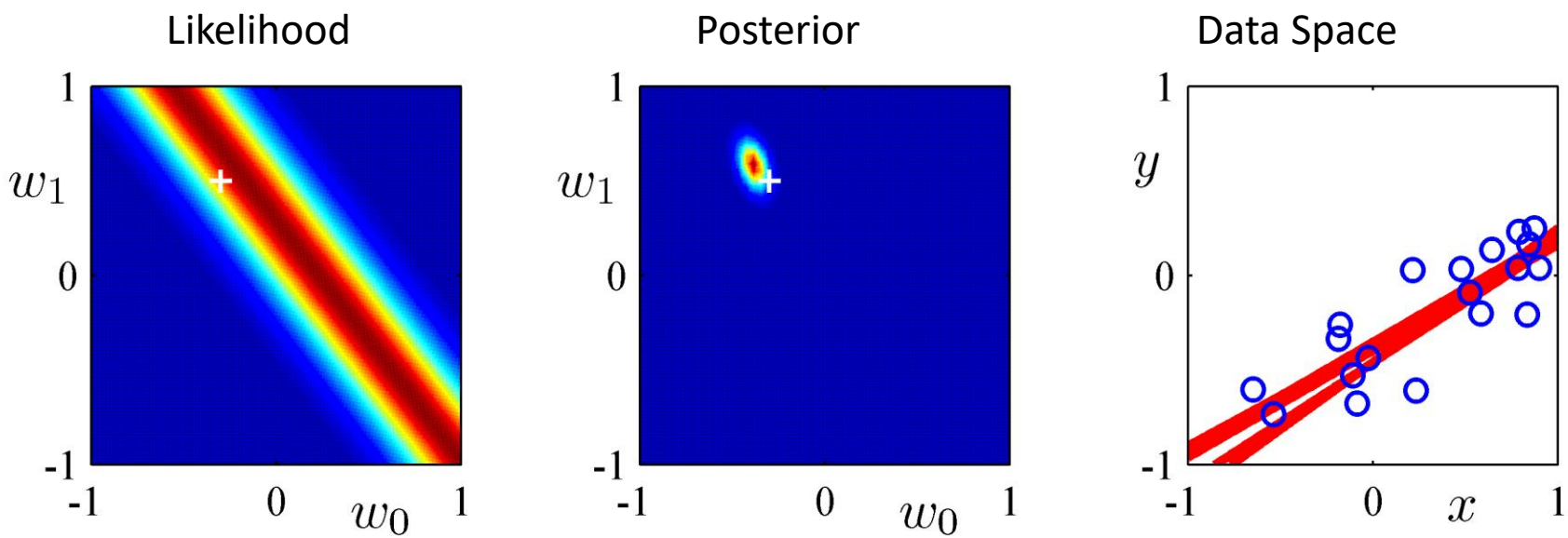
Bayesian Linear Regression Example

2 data points observed

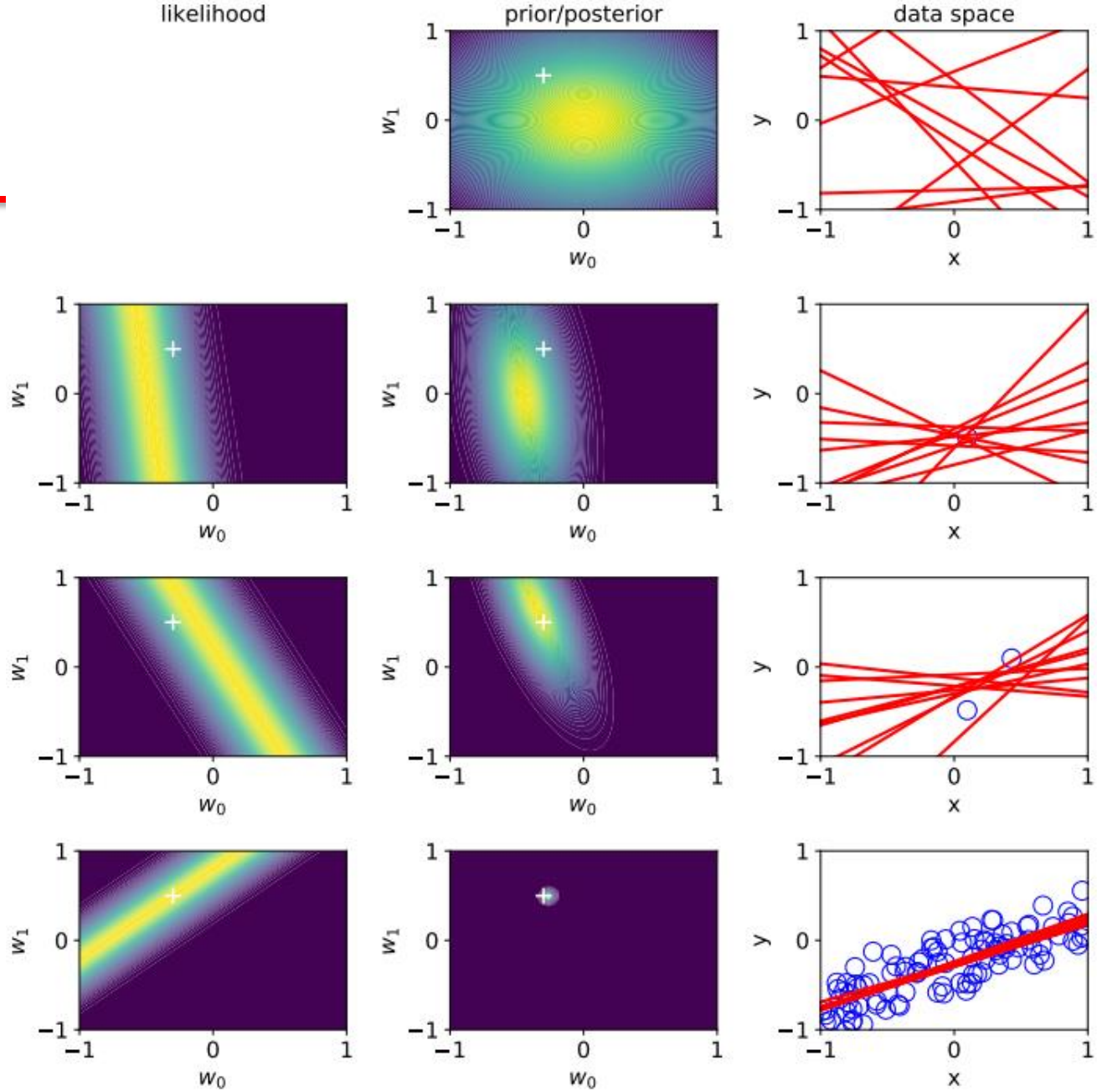


Bayesian Linear Regression Example

20 data points observed



Another View of the Bayesian Linear Regression Example



Predictive Distribution

- **Goal**
  - predict t for new values of $\mathbf{x}$ by integrating over $\mathbf{w}$

$$
\begin{aligned}
p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) &= \int p(t|\mathbf{x}, \mathbf{w}, \beta) p(\mathbf{w}|\mathbf{t}, \alpha, \beta)\, \mathrm{d}\mathbf{w} \\
&= \mathcal{N}(t|\mu_N(\mathbf{x}), \sigma_N^2(\mathbf{x}))
\end{aligned}
$$

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

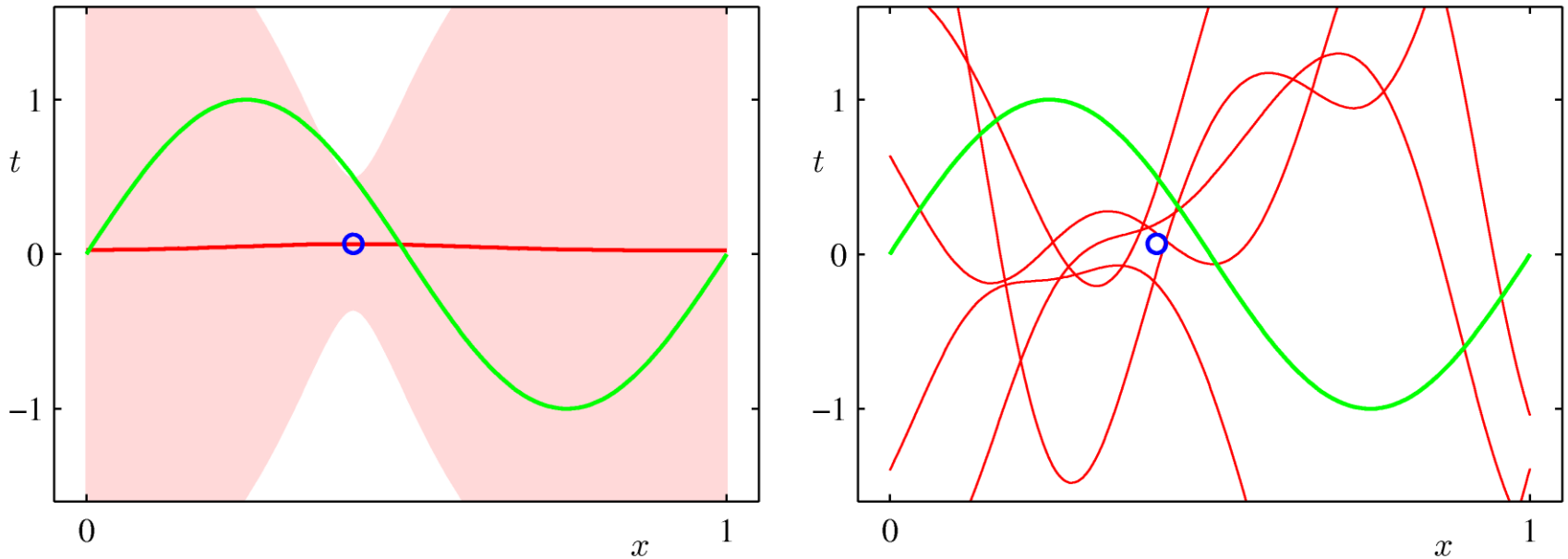
$$p(\mathbf{w}|\mathbf{t}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

$$
\begin{aligned}
\mathbf{m}_N &= \beta \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t} \\
\mathbf{S}_N^{-1} &= \alpha \mathbf{I} + \beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}
\end{aligned}
$$

$$\mu_N(\mathbf{x}) = y(\mathbf{x}, \mathbf{m}_N) = \mathbf{m}_N^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}) \qquad \sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x})$$

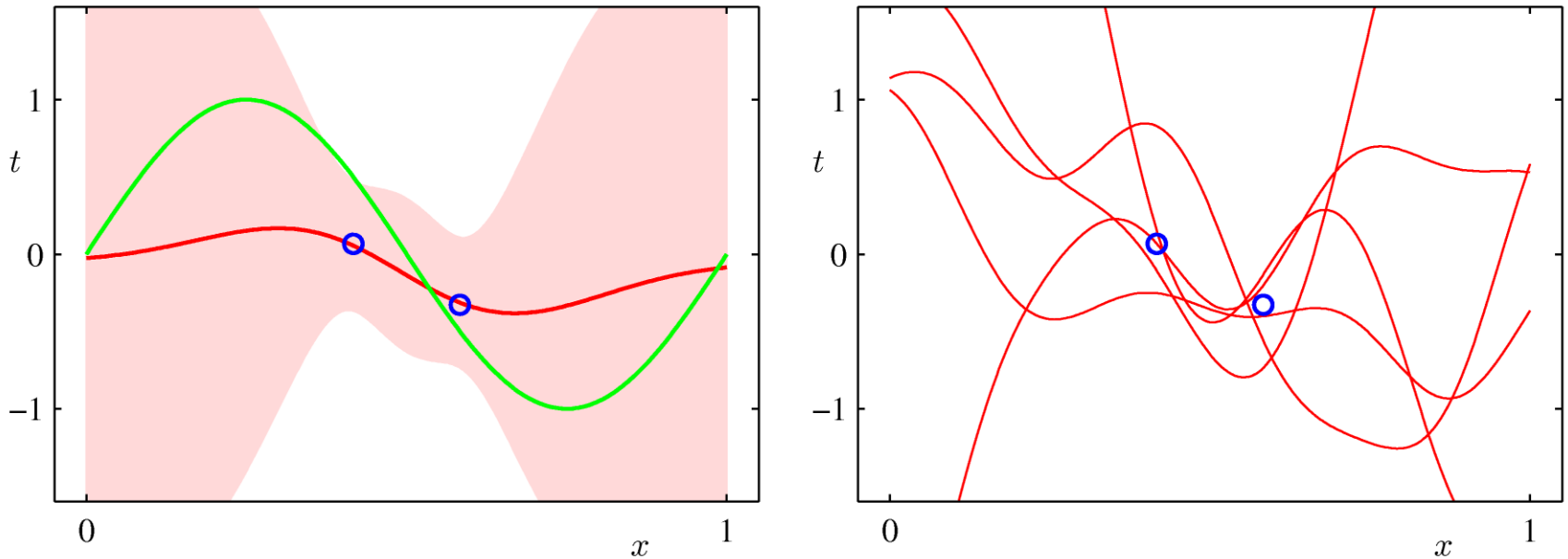
Predictive Distribution Example

– sinusoidal data, 9 Gaussian basis functions, 1 data point



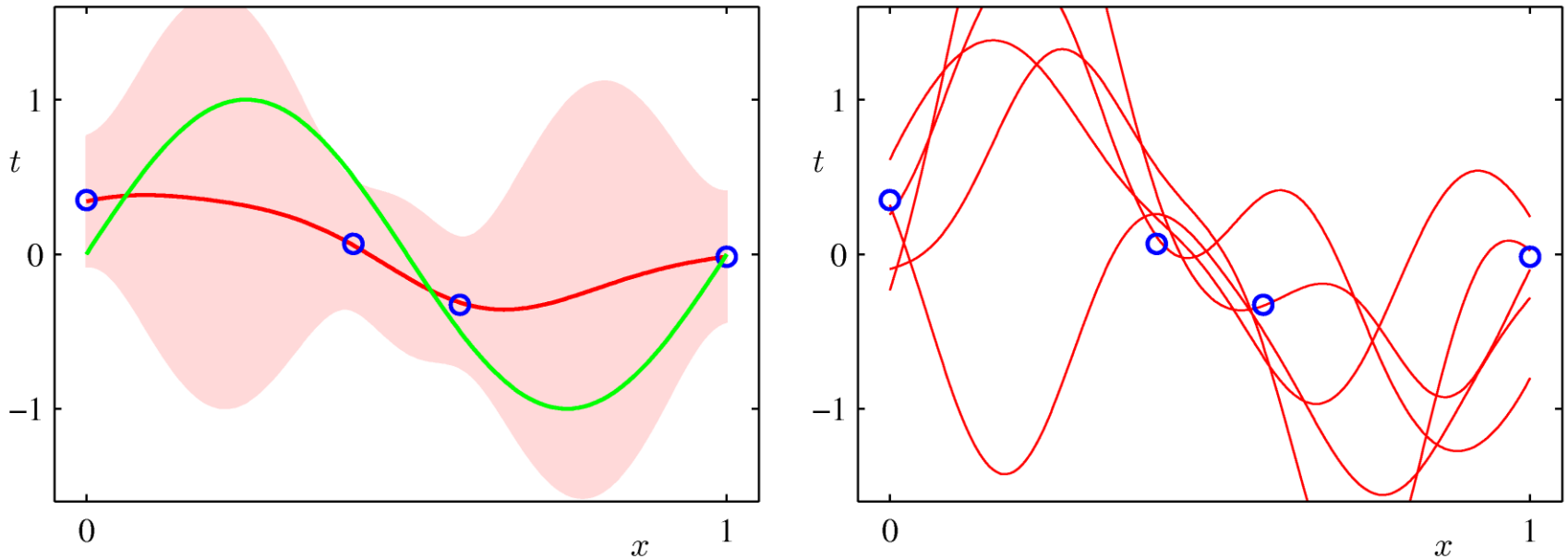
Predictive Distribution Example

– sinusoidal data, 9 Gaussian basis functions, 2 data points



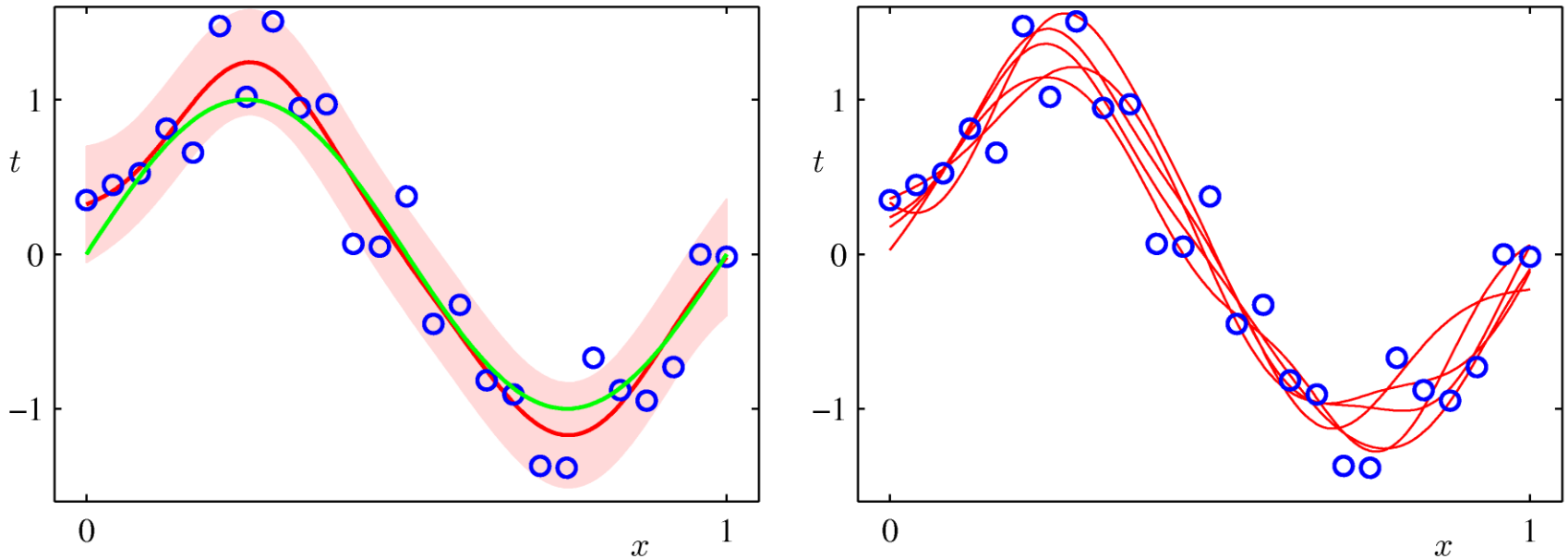
Predictive Distribution Example

– sinusoidal data, 9 Gaussian basis functions, 4 data points



Predictive Distribution Example

– sinusoidal data, 9 Gaussian basis functions, 25 data points



European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 05

Bayesian Model Comparison
The Evidence Approximation

TUC ECE, Spring 2023

Today

- **Bayesian Model Comparison**
  - model evidence
  - model trade-off
- **The Evidence Approximation**
  - maximization of the evidence function
  - effective number of parameters

Bayesian Model Comparison

Model Comparison

- **Question**
  - how to choose the right model for training?
  - polynomial? Gaussian? sigmoidal? order? number?
- **Considerations**
  - overfitting vs. generalization, bias vs. variance
- **Classical Approach**
  - cross-validation: multiple runs with different subsets of data
- **Alternative Approach**
  - marginalization over model parameters
  - avoids point estimates of model parameters
  - avoids multiple runs; models compared directly on the data

Bayesian Model Comparison

- **Question**
  - how do we choose the right model?
- **Problem**
  - compare models $\mathcal{M}_i$, $i=1, \ldots, L$, using data set $\mathcal{D}$

$$p(\mathcal{M}_i|\mathcal{D}) \propto p(\mathcal{M}_i)p(\mathcal{D}|\mathcal{M}_i)$$

Posterior Prior *model evidence* or *marginal likelihood*

  - *Bayes factor*: ratio of model evidence for two models

$$\frac{p(\mathcal{D}|\mathcal{M}_i)}{p(\mathcal{D}|\mathcal{M}_j)}$$

Bayesian Model Comparison

- **Predictive distribution**
  - having computed the posterior $p(\mathcal{M}_i|\mathcal{D})$, …
  - … we can compute the predictive (mixture) distribution

$$p(t|\mathbf{x}, \mathcal{D}) = \sum_{i=1}^{L} p(t|\mathbf{x}, \mathcal{M}_i, \mathcal{D}) p(\mathcal{M}_i|\mathcal{D})$$

- **Simple approximation**
  - known as *model selection*
  - simply use the model with the highest posterior

Model Evidence

- **Model evidence**
  - given a model with parameters $\mathbf{w}$, …
  - … we get the model evidence by marginalizing over $\mathbf{w}$

$$p(\mathcal{D}|\mathcal{M}_i) = \int p(\mathcal{D}|\mathbf{w}, \mathcal{M}_i) p(\mathbf{w}|\mathcal{M}_i) \, \mathrm{d}\mathbf{w}$$

- **Observation**
  - the model evidence is the normalizer in Bayes posterior

$$p(\mathbf{w}|\mathcal{D}, \mathcal{M}_i) = \frac{p(\mathcal{D}|\mathbf{w}, \mathcal{M}_i) p(\mathbf{w}|\mathcal{M}_i)}{p(\mathcal{D}|\mathcal{M}_i)}$$

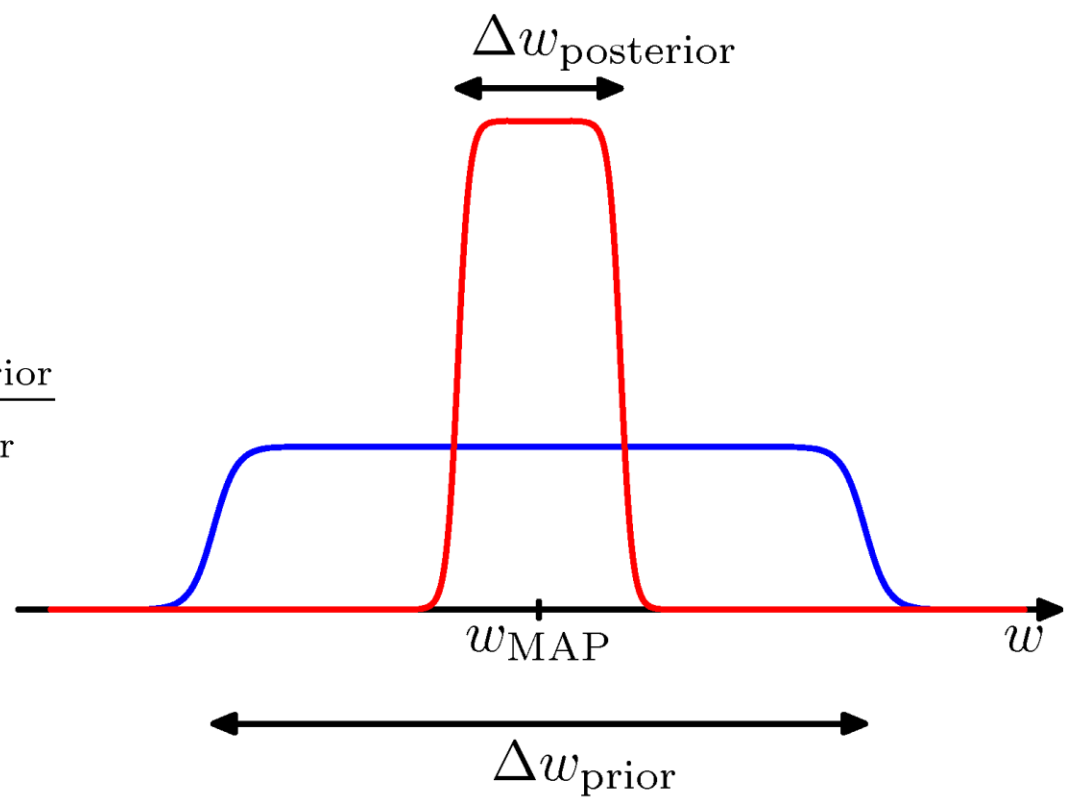
Approximating the Posterior

For a given model with a single parameter, w, consider the approximation

$$p(\mathcal{D}) = \int p(\mathcal{D}|w)p(w)\,\mathrm{d}w$$

$$\simeq p(\mathcal{D}|w_{\mathrm{MAP}})\frac{\Delta w_{\mathrm{posterior}}}{\Delta w_{\mathrm{prior}}}$$

where the prior is assumed to be flat and the posterior is assumed to be sharply peaked around w_{MAP}.



Model Trade-Off

- **Log Posterior**
  - taking the logarithm of the approximation

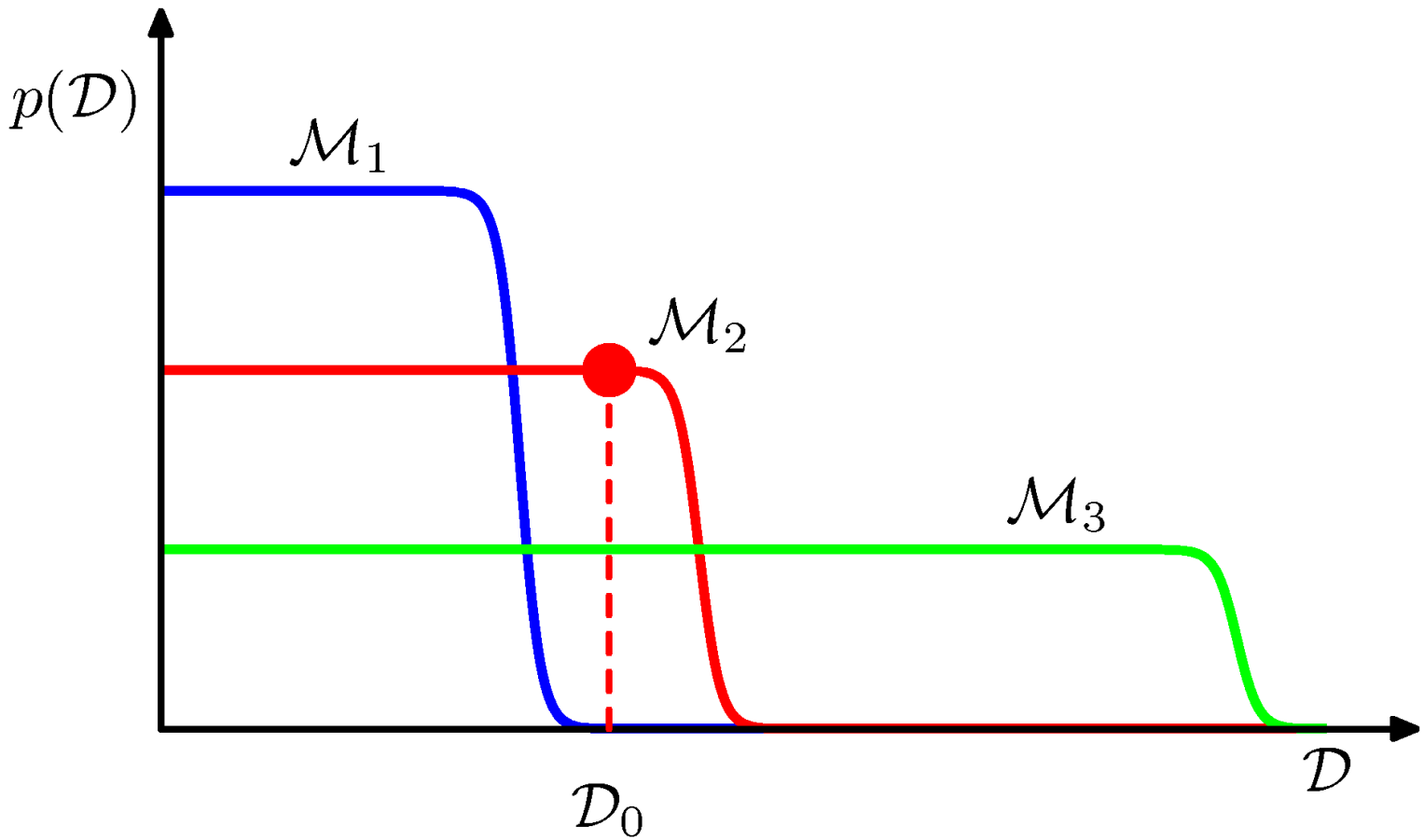
$$\ln p(\mathcal{D}) \simeq \ln p(\mathcal{D}|w_{\mathrm{MAP}}) + \underbrace{\ln\left(\frac{\Delta w_{\mathrm{posterior}}}{\Delta w_{\mathrm{prior}}}\right)}_{\text{negative}}$$

  - first term: fit to the data, given the most probable parameters
  - second term: penalty according to model complexity
- **Multiple parameters**
  - for M parameters, having the same ratio $\Delta w_{\mathrm{posterior}}/\Delta w_{\mathrm{prior}}$

$$\ln p(\mathcal{D}) \simeq \ln p(\mathcal{D}|\mathbf{w}_{\mathrm{MAP}}) + \underbrace{M \ln\left(\frac{\Delta w_{\mathrm{posterior}}}{\Delta w_{\mathrm{prior}}}\right)}_{\text{negative and linear in } M}$$

Trade-Off in Model Evidence

- matching data and model complexity



Bayesian Model Comparison

- **Assumption**
  - the true data distribution is contained in considered models
- **Principle**
  - Bayesian model comparison will favor the correct over others
- **Bayes factor**
  - for a single data set, it may be larger for some incorrect model
  - however, the expected Bayes factor will favor the correct one

$$\int p(\mathcal{D}|\mathcal{M}_1) \ln \frac{p(\mathcal{D}|\mathcal{M}_1)}{p(\mathcal{D}|\mathcal{M}_2)} \, \mathrm{d}\mathcal{D}$$

  - an example of the Kullback-Leibler (KL) divergence

The Evidence Approximation

Recall: Bayesian Linear Regression

- assuming basis $\phi(\mathbf{x})$ and zero-mean, isotropic prior over $\mathbf{w}$…
- … the predictive distribution over t for new values of $\mathbf{x}$ is

$$
\begin{aligned}
p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) &= \int p(t|\mathbf{x}, \mathbf{w}, \beta) p(\mathbf{w}|\mathbf{t}, \alpha, \beta)\, \mathrm{d}\mathbf{w} \\
&= \mathcal{N}(t|\mu_N(\mathbf{x}), \sigma_N^2(\mathbf{x}))
\end{aligned}
$$

$$
\mu_N(\mathbf{x}) = y(\mathbf{x}, \mathbf{m}_N) = \mathbf{m}_N^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})
$$

$$
\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x})
$$

$$
\begin{aligned}
\mathbf{m}_N &= \beta \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t} \\
\mathbf{S}_N^{-1} &= \alpha \mathbf{I} + \beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}
\end{aligned}
$$

The Evidence Approximation

- **Fully Bayesian predictive distribution**

$$p(t|\mathbf{t}) = \iiint p(t|\mathbf{w},\beta)p(\mathbf{w}|\mathbf{t},\alpha,\beta)p(\alpha,\beta|\mathbf{t})\,\mathrm{d}\mathbf{w}\,\mathrm{d}\alpha\,\mathrm{d}\beta$$

  – all parameters marginalized, but the integral is intractable

- **Evidence approximation**

$$p(t|\mathbf{t}) \simeq p\left(t|\mathbf{t},\widehat{\alpha},\widehat{\beta}\right) = \int p\left(t|\mathbf{w},\widehat{\beta}\right) p\left(\mathbf{w}|\mathbf{t},\widehat{\alpha},\widehat{\beta}\right)\,\mathrm{d}\mathbf{w}$$

  – where $\left(\widehat{\alpha},\widehat{\beta}\right)$ is the mode (MAP) of the posterior $p(\alpha,\beta|\mathbf{t})$
  – the posterior is assumed to be sharply peaked
  – fixed values obtained by maximizing the marginal likelihood
  – *empirical Bayes, type II* or *generalized maximum likelihood*

Maximum Marginal Likelihood

- from Bayes' theorem

$$p(\alpha, \beta|\mathbf{t}) \propto p(\mathbf{t}|\alpha, \beta)p(\alpha, \beta)$$

- assume $p(\alpha,\beta)$ to be flat

$$\begin{aligned} p(\alpha, \beta|\mathbf{t}) &\propto p(\mathbf{t}|\alpha, \beta) \\ &= \int p(\mathbf{t}|\mathbf{w}, \beta)p(\mathbf{w}|\alpha)\,\mathrm{d}\mathbf{w} \end{aligned}$$

- using properties of Gaussian integrals
- using linear basis functions

$$\ln p(\mathbf{t}|\alpha, \beta) = \frac{M}{2}\ln\alpha + \frac{N}{2}\ln\beta - E(\mathbf{m}_N) + \frac{1}{2}\ln|\mathbf{S}_N| - \frac{N}{2}\ln(2\pi)$$

- [derivation on the next slide]

Derivation

- manipulate

$$p(\mathbf{t}|\alpha,\beta) = \int p(\mathbf{t}|\mathbf{w},\beta)p(\mathbf{w}|\alpha)\,\mathrm{d}\mathbf{w} = \int \mathcal{N}(\mathbf{t}|\mathbf{\Phi}\mathbf{w},\beta^{-1})\mathcal{N}(\mathbf{w}|\mathbf{0},\alpha^{-1}\mathbf{I})\,\mathrm{d}\mathbf{w}$$

$$= \left(\frac{\beta}{2\pi}\right)^{N/2}\left(\frac{\alpha}{2\pi}\right)^{M/2}\int \exp\left\{-\frac{\beta}{2}\parallel \mathbf{t}-\mathbf{\Phi}\mathbf{w}\parallel^2 - \frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}\right\}\mathrm{d}\mathbf{w}$$

- complete the square

$$-\frac{\beta}{2}\parallel \mathbf{t}-\mathbf{\Phi}\mathbf{w}\parallel^2 - \frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} = -E(\mathbf{m}_N) - \frac{1}{2}(\mathbf{w}-\mathbf{m}_N)^{\mathrm{T}}\mathbf{S}_N^{-1}(\mathbf{w}-\mathbf{m}_N)$$

$$E(\mathbf{m}_N) = \frac{\beta}{2}\parallel \mathbf{t}-\mathbf{\Phi}\mathbf{m}_N\parallel^2 + \frac{\alpha}{2}\mathbf{m}_N^{\mathrm{T}}\mathbf{m}_N \qquad \mathbf{m}_N = \beta\mathbf{S}_N\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} \qquad \mathbf{S}_N^{-1} = \alpha\mathbf{I} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$$

- integrate

$$p(\mathbf{t}|\alpha,\beta) = \left(\frac{\beta}{2\pi}\right)^{N/2}\left(\frac{\alpha}{2\pi}\right)^{M/2}\int \exp\{-E(\mathbf{m}_N)\}\exp\left\{-\frac{1}{2}(\mathbf{w}-\mathbf{m}_N)^{\mathrm{T}}\mathbf{S}_N^{-1}(\mathbf{w}-\mathbf{m}_N)\right\}\mathrm{d}\mathbf{w}$$

$$= \left(\frac{\beta}{2\pi}\right)^{N/2}\left(\frac{\alpha}{2\pi}\right)^{M/2} e^{-E(\mathbf{m}_N)}(2\pi)^{M/2}|\mathbf{S}_N|^{1/2} = \alpha^{M/2}\beta^{N/2}e^{-E(\mathbf{m}_N)}|\mathbf{S}_N|^{1/2}(2\pi)^{-N/2}$$

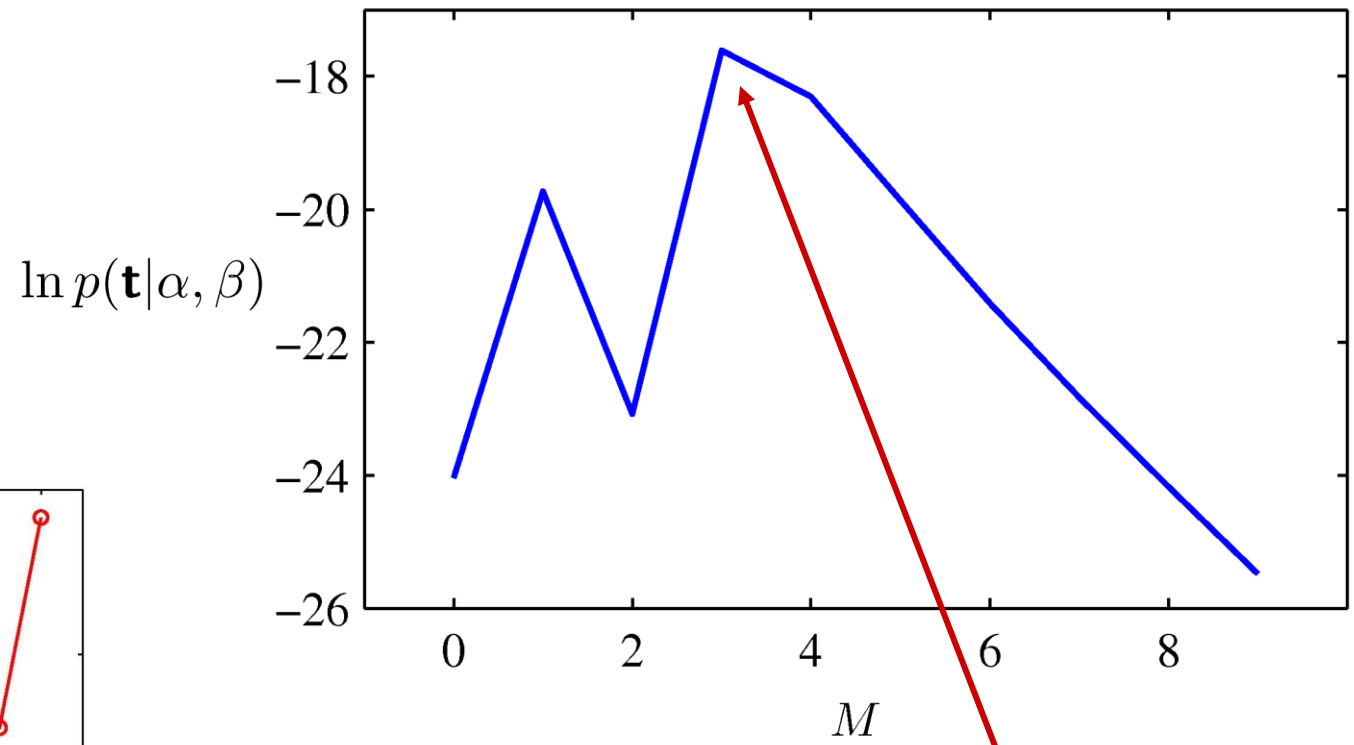
- take the logarithm

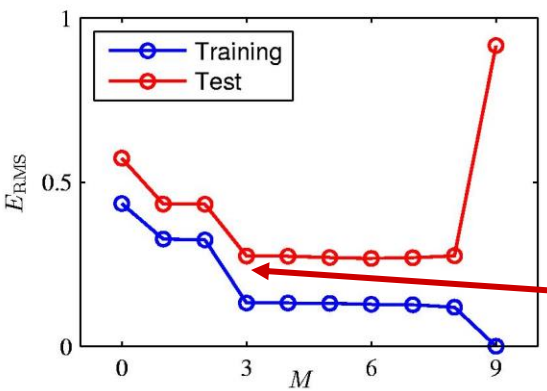
$$\ln p(\mathbf{t}|\alpha,\beta) = \frac{M}{2}\ln\alpha + \frac{N}{2}\ln\beta - E(\mathbf{m}_N) + \frac{1}{2}\ln|\mathbf{S}_N| - \frac{N}{2}\ln(2\pi)$$

Maximum Marginal Likelihood Example

- sinusoidal data, basis functions: M^{th} degree polynomial

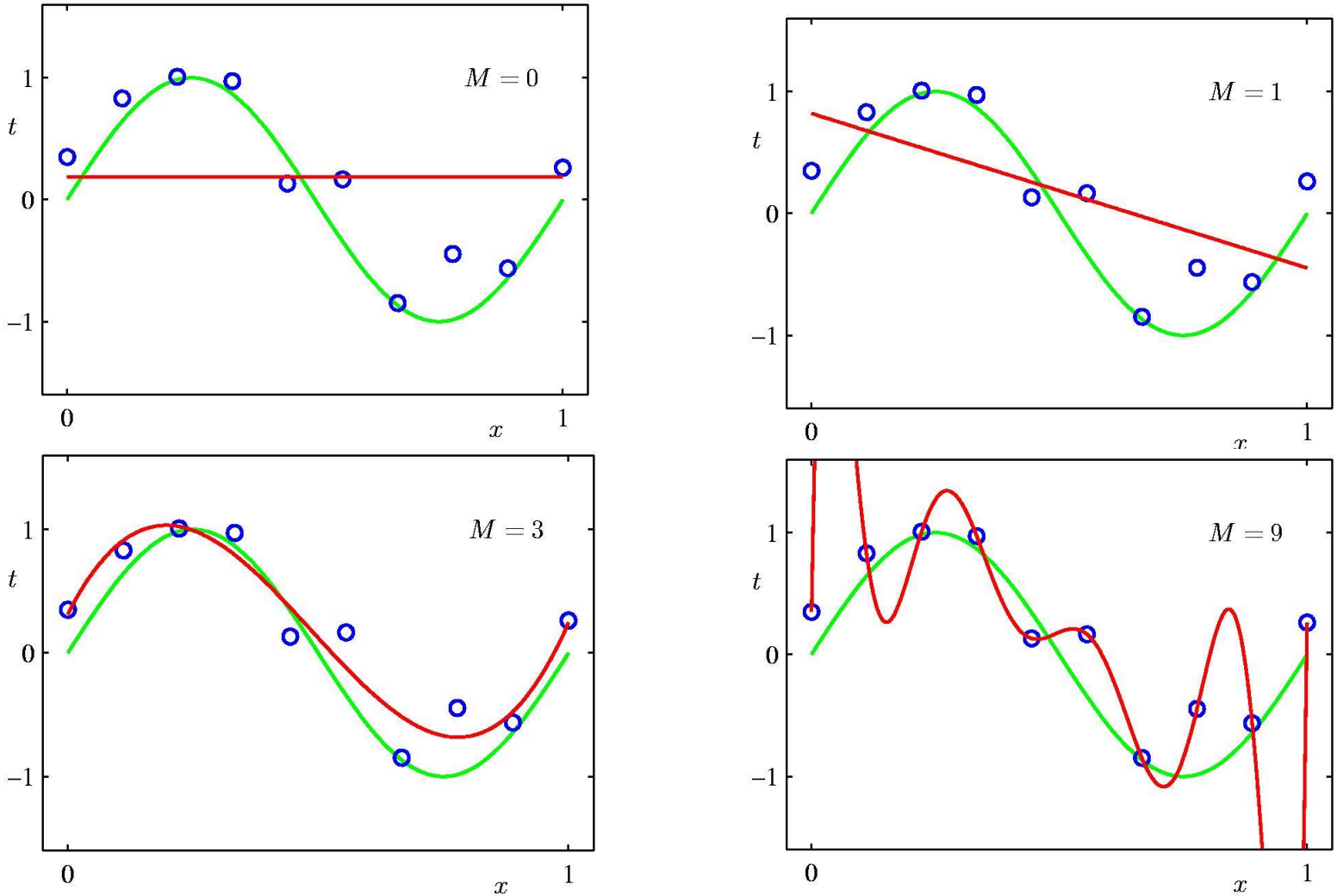
$$\alpha = 5 \times 10^{-3}$$
$$\beta = 11.1$$
$$N = 10$$





best model!

Recall: Polynomial Curve Fitting



Maximizing the Evidence Function

$$\ln p(\mathbf{t}|\alpha,\beta) = \frac{M}{2}\ln\alpha + \frac{N}{2}\ln\beta - E(\mathbf{m}_N) + \frac{1}{2}\ln|\mathbf{S}_N| - \frac{N}{2}\ln(2\pi)$$

- **Maximization (MAP)**
  - to maximize $\ln p(\mathbf{t}|\alpha,\beta)$ with respect to α and β, …
  - … differentiate $\ln p(\mathbf{t}|\alpha,\beta)$ w.r.t. α and β, …
  - … and set the results to zero
- **Consideration**
  - define the eigenvector equation $\left(\beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)\mathbf{u}_i = \lambda_i\mathbf{u}_i$
  - then the matrix

$$\mathbf{A} = \mathbf{S}_N^{-1} = \alpha\mathbf{I} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$$

  - has eigenvalues $\lambda_i + \alpha$

Maximizing over α

$$\ln p(\mathbf{t}|\alpha,\beta) = \frac{M}{2}\ln\alpha + \frac{N}{2}\ln\beta - E(\mathbf{m}_N) - \frac{1}{2}\ln|A| - \frac{N}{2}\ln(2\pi)$$

$$E(\mathbf{m}_N) = \frac{\beta}{2}\parallel \mathbf{t} - \mathbf{\Phi}\mathbf{m}_N \parallel^2 + \frac{\alpha}{2}\mathbf{m}_N^{\mathrm{T}}\mathbf{m}_N$$

– differentiate $\ln p(\mathbf{t}|\alpha,\beta)$ w.r.t. α and set to zero

$$\frac{d}{d\alpha}\ln|\mathbf{A}| = \frac{d}{d\alpha}\ln\prod_i(\lambda_i+\alpha) = \frac{d}{d\alpha}\sum_i\ln(\lambda_i+\alpha) = \sum_i\frac{1}{\lambda_i+\alpha}$$

$$\frac{d}{d\alpha}\ln p(\mathbf{t}|\alpha,\beta) = 0 \quad\Longrightarrow\quad \frac{M}{2\alpha} - \frac{1}{2}\mathbf{m}_N^{\mathrm{T}}\mathbf{m}_N - \frac{1}{2}\sum_{i=1}^{M}\frac{1}{\lambda_i+\alpha} = 0$$

$$\Longrightarrow\quad \alpha\mathbf{m}_N^{\mathrm{T}}\mathbf{m}_N = M - \sum_{i=1}^{M}\frac{\alpha}{\lambda_i+\alpha} = \gamma$$

$$\alpha = \frac{\gamma}{\mathbf{m}_N^{\mathrm{T}}\mathbf{m}_N} \qquad \gamma = \sum_i\frac{\lambda_i}{\alpha+\lambda_i}$$

γ depends on both α and β

Maximizing over β

$$\ln p(\mathbf{t}|\alpha,\beta) = \frac{M}{2}\ln\alpha + \frac{N}{2}\ln\beta - E(\mathbf{m}_N) - \frac{1}{2}\ln|A| - \frac{N}{2}\ln(2\pi)$$

$$E(\mathbf{m}_N) = \frac{\beta}{2}\parallel \mathbf{t} - \mathbf{\Phi}\mathbf{m}_N \parallel^2 + \frac{\alpha}{2}\mathbf{m}_N^{\mathrm{T}}\mathbf{m}_N$$

- differentiate $\ln p(\mathbf{t}|\alpha,\beta)$ w.r.t. β and set to zero
- the eigenvalues λ_i are proportional to β : $\left(\beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)\mathbf{u}_i = \lambda_i\mathbf{u}_i$

$$\frac{d}{d\beta}\ln|\mathbf{A}| = \frac{d}{d\beta}\sum_i \ln(\lambda_i+\alpha) = \frac{1}{\beta}\sum_i \frac{\lambda_i}{\lambda_i+\alpha} = \frac{\gamma}{\beta}$$

$$\frac{d}{d\beta}\ln p(\mathbf{t}|\alpha,\beta) = 0 \quad\Longrightarrow\quad \frac{N}{2\beta} - \frac{1}{2}\sum_{i=1}^{N}\left(t_n - \mathbf{m}_N^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right)^2 - \frac{\gamma}{2\beta} = 0$$

$$\Longrightarrow\quad \frac{1}{\beta} = \frac{1}{N-\gamma}\sum_{n=1}^{N}\left(t_n - \mathbf{m}_N^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right)^2$$

Maximizing the Evidence Function

- **Iterative maximization**
  - give arbitrary values to α and β and iterate until convergence
  - step I: given α and β, compute γ and $\mathbf{m}_N$

$$\mathbf{m}_N = \beta \mathbf{S}_N \mathbf{\Phi}^{\mathrm{T}} \mathbf{t}$$

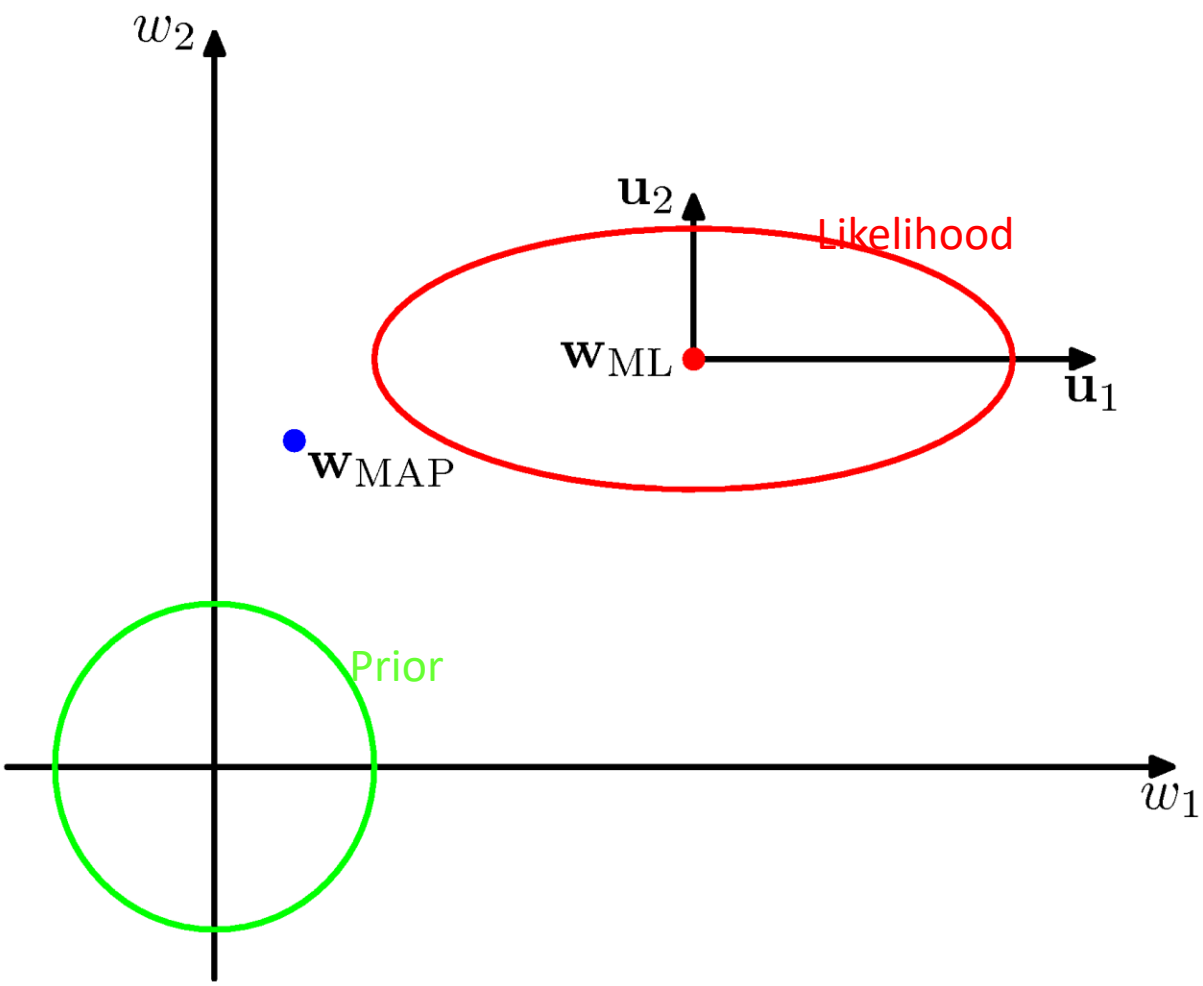
$$\gamma = \sum_i \frac{\lambda_i}{\alpha + \lambda_i}$$

  - step II: given γ and $\mathbf{m}_N$, compute α and β

$$\alpha = \frac{\gamma}{\mathbf{m}_N^{\mathrm{T}} \mathbf{m}_N}$$

$$\frac{1}{\beta} = \frac{1}{N - \gamma} \sum_{n=1}^{N} \left(t_n - \mathbf{m}_N^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right)^2$$

Effective Number of Parameters



$$\gamma = \sum_i \frac{\lambda_i}{\alpha + \lambda_i}$$

$\lambda_1 \ll \alpha$ (ratio in $\gamma \cong 0$)
w_1 is not well determined by the likelihood

$\lambda_2 \gg \alpha$ (ratio in $\gamma \cong 1$)
w_2 is well determined by the likelihood

γ is the number of well determined parameters

Bias Correction in the Mean

- **Bayesian estimation**

$$\frac{1}{\beta} = \frac{1}{N-\gamma}\sum_{n=1}^{N}\left(t_n - \mathbf{m}_N^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right)^2$$

excludes the number of effective parameters in the normalization!

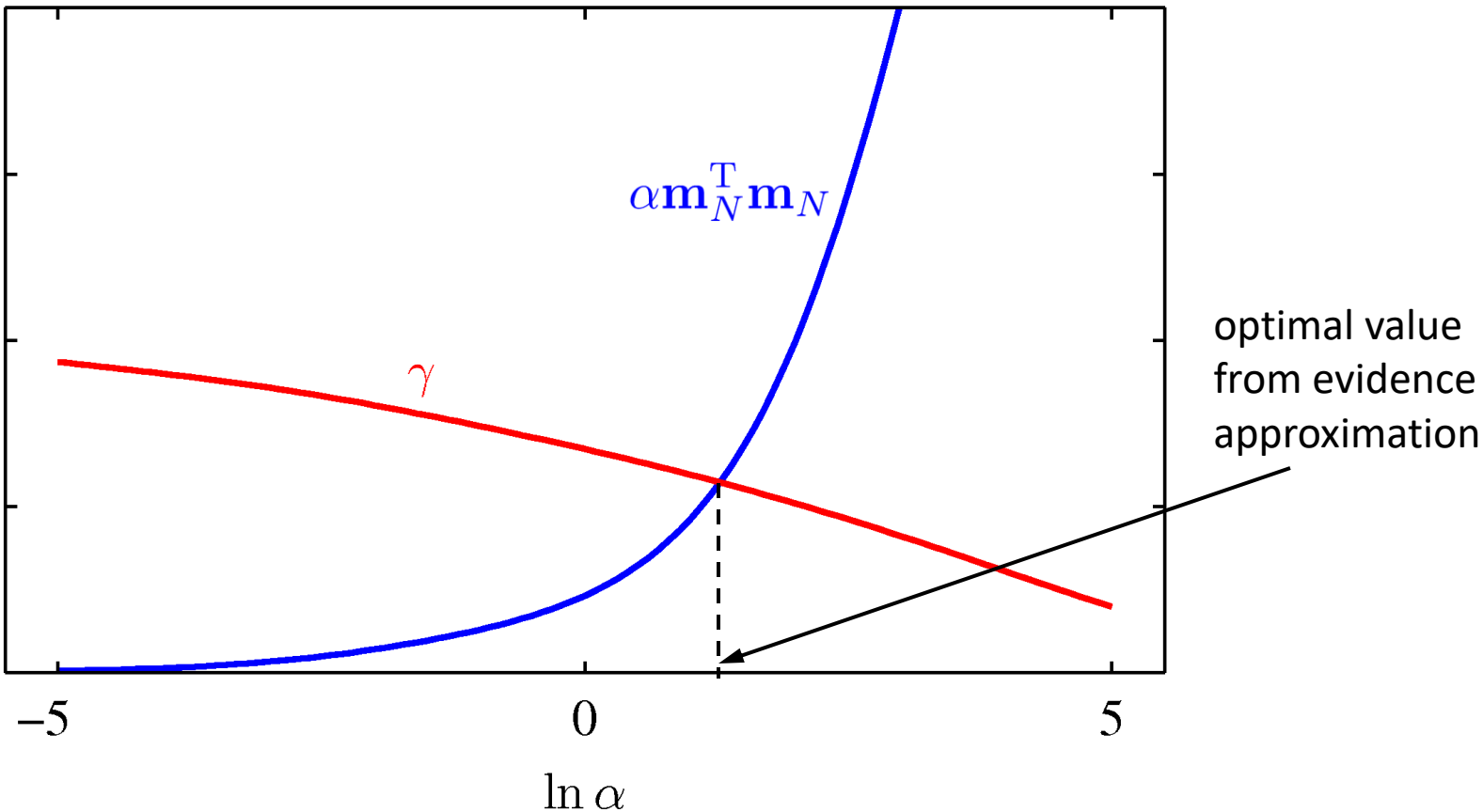
- **ML estimation**

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N}\sum_{n=1}^{N}\{t_n - \mathbf{w}_{\mathrm{ML}}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

- **Recall: Gaussian variance estimation**
  - biased $\sigma^2_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{\mathrm{ML}})^2$
  - unbiased $\sigma^2_{\mathrm{MAP}} = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \mu_{\mathrm{ML}})^2$

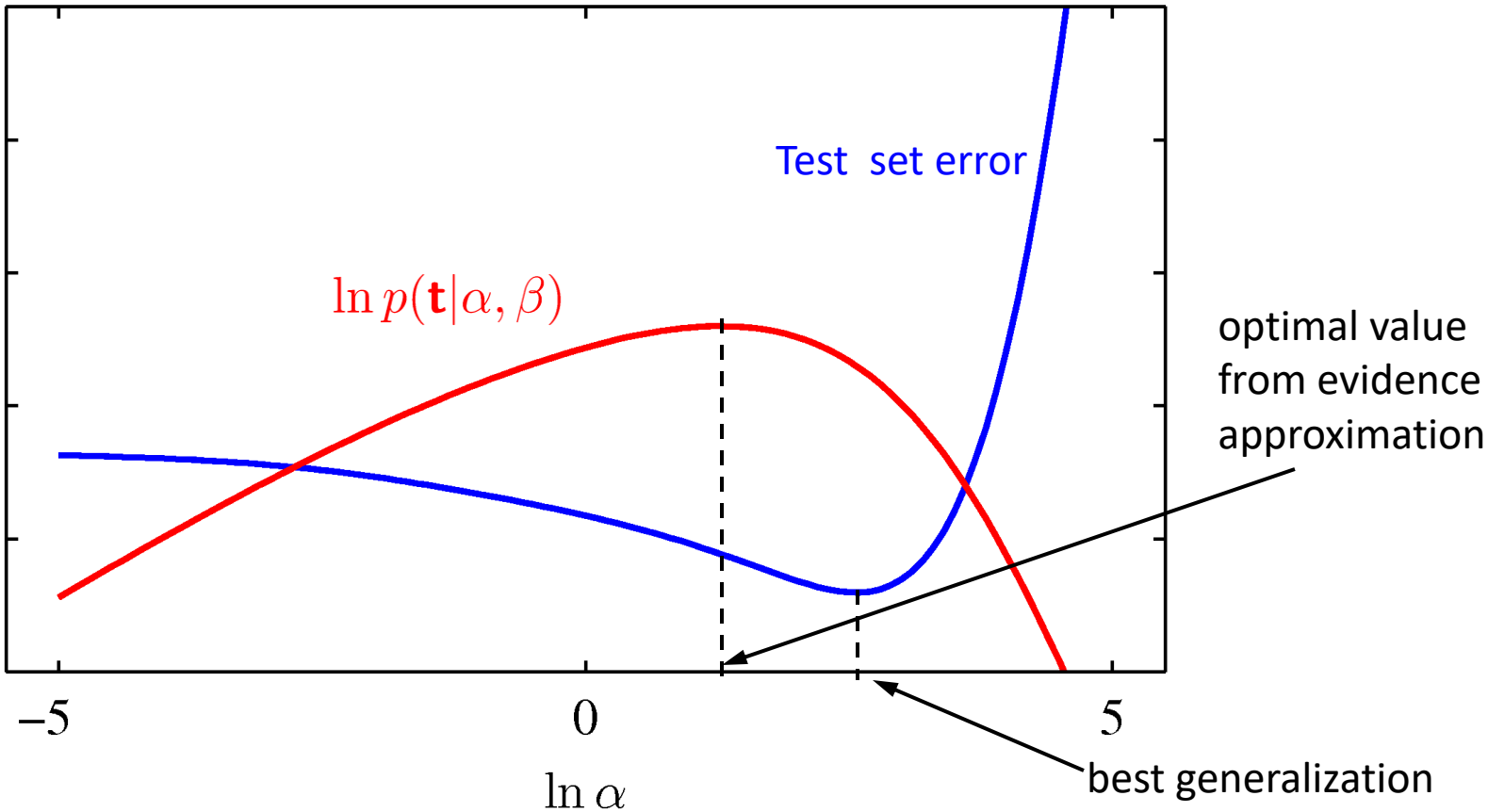
Effective Number of Parameters Example

- sinusoidal data, 9 Gaussian basis functions and bias (M=10)
- $\beta = 11.1$ (true value)



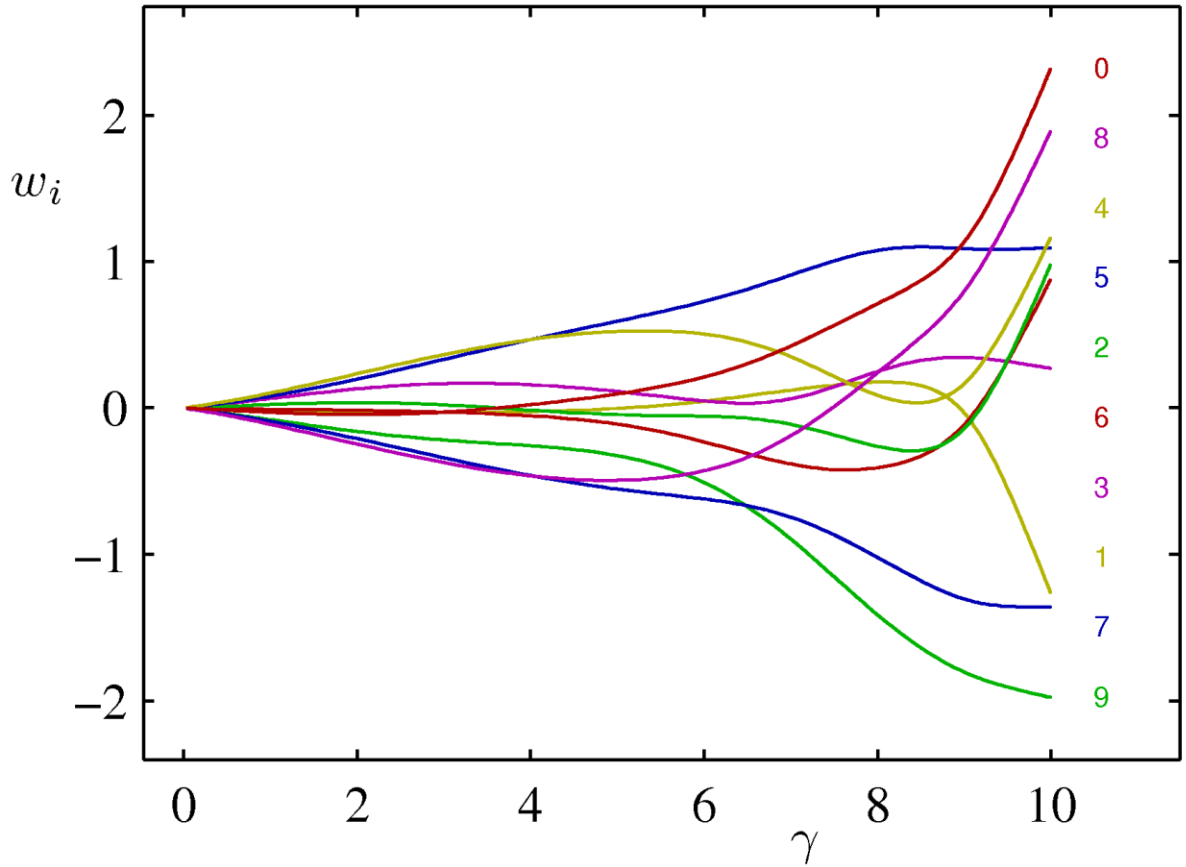
Effective Number of Parameters Example

- sinusoidal data, 9 Gaussian basis functions and bias (M=10)
- $\beta = 11.1$ (true value)



Effective Number of Parameters Example

- sinusoidal data, 9 Gaussian basis functions and bias (M=10)
- $\beta = 11.1$ (true value), $0 \leq \alpha \leq +\infty$ which implies that $0 \leq \gamma \leq M$



Evidence Approximation in Practice

- **Large data sets**
  - in the limit $N \gg M$, $\gamma = M$ because of large eigenvalues
  - we can consider using the easy-to-compute approximation

$$\alpha = \frac{M}{\mathbf{m}_N^{\mathrm{T}}\mathbf{m}_N}$$

$$\frac{1}{\beta} = \frac{1}{N}\sum_{n=1}^{N}\left\{t_n - \mathbf{m}_N^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right\}^2$$

  - no evaluation of the eigenvalue spectrum required

Limitations of Fixed Basis Functions

- **Curse of dimensionality**
  - M basis functions along each dimension
  - a D-dimensional input space requires M^D basis functions!
- **Considerations**
  - data typically lie in a non-linear manifold of lower dimension
  - targets occasionally depend only on a few input variables
- **Question**
  - can we choose fewer basis functions using the training data?
  - can we actively localize the basis functions in input space?

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 06

Gaussians

Derivations and Proofs

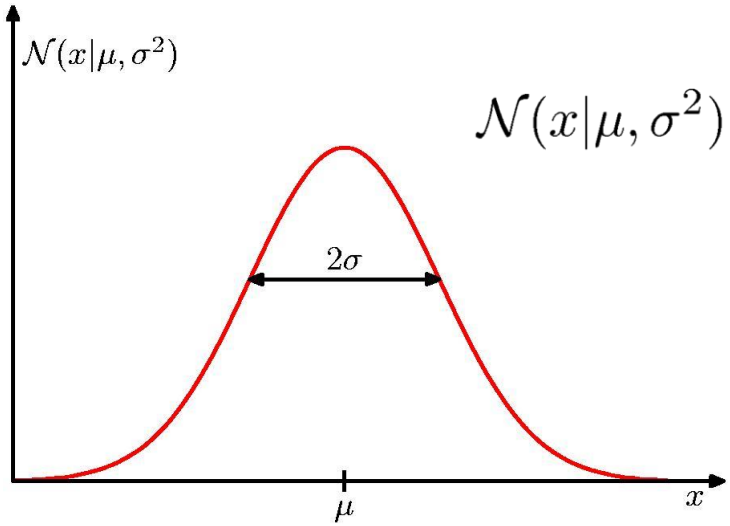
TUC ECE, Spring 2023

Today

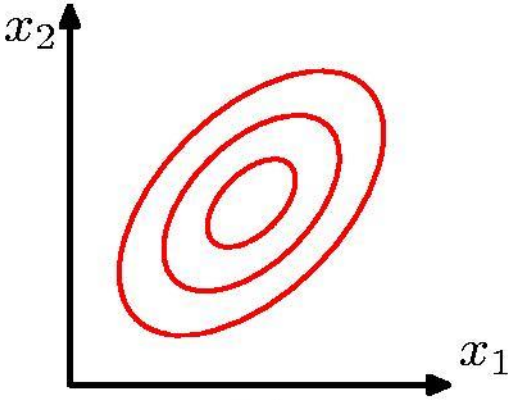
- **Gaussian**
  - geometry, properties, forms
- **Partitioned Gaussian**
  - conditional and marginal
- **Bayes' Theorem**
  - for Gaussian variables
- **Application**
  - Bayesian Linear Regression
  - Predictive Distribution

Gaussian Distributions

The Gaussian (Normal) Distribution



$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$



$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

Geometry of the Multivariate Gaussian

$$\Delta^2 = (\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})$$

$$\boldsymbol{\Sigma}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

$$\boldsymbol{\Sigma} = \sum_{i=1}^{D} \lambda_i \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}}$$

$$\boldsymbol{\Sigma}^{-1} = \sum_{i=1}^{D} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}}$$

$$\Delta^2 = \sum_{i=1}^{D} \frac{y_i^2}{\lambda_i}$$

$$y_i = \mathbf{u}_i^{\mathrm{T}} (\mathbf{x} - \boldsymbol{\mu})$$

$$\mathbf{y} = \mathbf{U}(\mathbf{x} - \boldsymbol{\mu})$$

$$\mathbf{U}\mathbf{U}^{\top} = \mathbf{U}^{\top}\mathbf{U} = \mathbf{I}$$

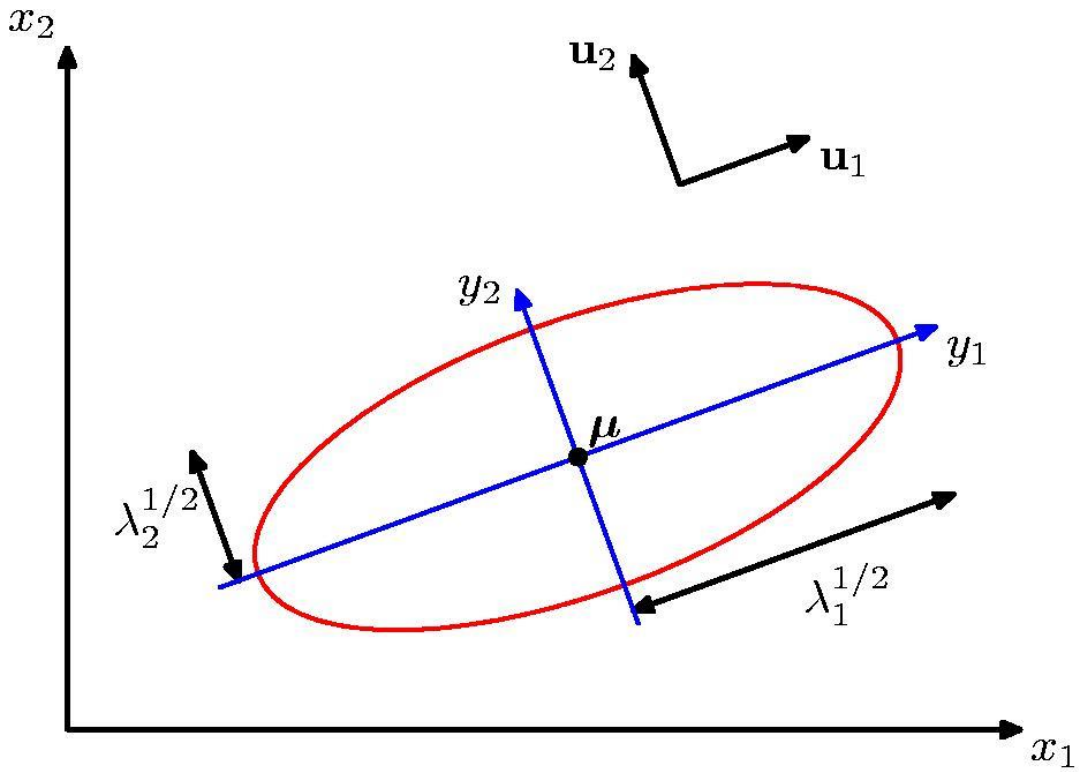
Δ : the Mahalanobis distance from $\boldsymbol{\mu}$ to $\mathbf{x}$

D : the number of dimensions, i = 1, 2, …, D

$\boldsymbol{\Sigma}$: symmetric matrix with real eigenvalues λ_i

$\mathbf{u}_i$: eigenvectors of $\boldsymbol{\Sigma}$ (orthonormal basis)

$\mathbf{U}$: orthogonal matrix, eigenvectors in rows



Properties of the Multivariate Gaussian

new coordinate system $\mathbf{y} = \mathbf{U}(\mathbf{x} - \boldsymbol{\mu})$ $\quad \mathbf{x} = \boldsymbol{\mu} + \mathbf{U}^{\mathrm{T}}\mathbf{y}$ $\quad x_i = \mu_i + \mathbf{U}_i^{\mathrm{T}}\mathbf{y}$

Jacobian matrix $\mathbf{J}$ $\quad J_{ij} = \dfrac{\partial x_i}{\partial y_j} = U_{ij}^{\mathrm{T}}$ $\quad \mathbf{J} = \mathbf{U}^{\mathrm{T}} \quad |\mathbf{J}| = 1$ $\quad |\boldsymbol{\Sigma}|^{1/2} = \prod_{j=1}^{D} \lambda_j^{1/2}$

in the old coordinates

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right\}$$

in the new coordinates

$$p(\mathbf{y}) = p(\mathbf{x})|\mathbf{J}| = \prod_{j=1}^{D} \frac{1}{(2\pi\lambda_j)^{1/2}} \exp\left\{-\frac{y_j^2}{2\lambda_j}\right\}$$

normalized distribution

$$\int p(\mathbf{y})\,\mathrm{d}\mathbf{y} = \prod_{j=1}^{D} \int_{-\infty}^{+\infty} \frac{1}{(2\pi\lambda_j)^{1/2}} \exp\left\{-\frac{y_j^2}{2\lambda_j}\right\} \mathrm{d}y_j = 1$$

Moments of the Multivariate Gaussian

$$\mathbb{E}[\mathbf{x}] = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\int \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}\mathbf{x}\,\mathrm{d}\mathbf{x}$$

$$= \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\int \exp\left\{-\frac{1}{2}\mathbf{z}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{z}\right\}(\mathbf{z}+\boldsymbol{\mu})\,\mathrm{d}\mathbf{z} \qquad (\mathbf{z}=\mathbf{x}-\boldsymbol{\mu})$$

$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}$ (thanks to anti-symmetry of **z**, the **z**-term vanishes)

$$\mathbb{E}[\mathbf{x}\mathbf{x}^{\mathrm{T}}] = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\int \exp\left\{-\frac{1}{2}\mathbf{z}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{z}\right\}(\mathbf{z}+\boldsymbol{\mu})(\mathbf{z}+\boldsymbol{\mu})^{\mathrm{T}}\,\mathrm{d}\mathbf{z}$$

(the two **z**-terms vanish)

(the $\boldsymbol{\mu}\boldsymbol{\mu}^{\mathrm{T}}$-term is constant)

$$\frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\int \exp\left\{-\frac{1}{2}\mathbf{z}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{z}\right\}\mathbf{z}\mathbf{z}^{\mathrm{T}}\,\mathrm{d}\mathbf{z}$$ (the $\mathbf{z}\mathbf{z}^{\mathrm{T}}$-term)

$$= \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\sum_{i=1}^{D}\sum_{j=1}^{D}\mathbf{u}_i\mathbf{u}_j^{\mathrm{T}}\int \exp\left\{-\sum_{k=1}^{D}\frac{y_k^2}{2\lambda_k}\right\}y_i y_j\,\mathrm{d}\mathbf{y}$$

$$\mathbf{z} = \sum_{j=1}^{D}\mathbf{u}_j^{\mathrm{T}}\mathbf{z}\mathbf{u}_j = \sum_{j=1}^{D}y_j\mathbf{u}_j = \sum_{i=1}^{D}y_i\mathbf{u}_i$$

$$= \sum_{i=1}^{D}\mathbf{u}_i\mathbf{u}_i^{\mathrm{T}}\lambda_i = \boldsymbol{\Sigma}$$ (the $y_i y_j$-terms vanish, unless i=j)

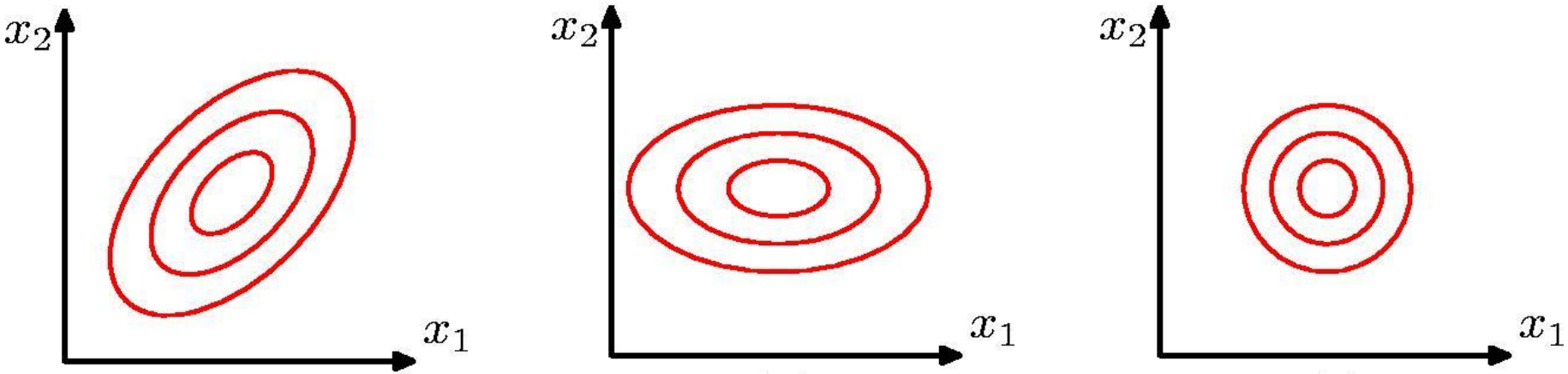
$$\mathbb{E}[\mathbf{x}\mathbf{x}^{\mathrm{T}}] = \boldsymbol{\mu}\boldsymbol{\mu}^{\mathrm{T}} + \boldsymbol{\Sigma}$$

$$\mathrm{cov}[\mathbf{x}] = \mathbb{E}\left[(\mathbf{x}-\mathbb{E}[\mathbf{x}])(\mathbf{x}-\mathbb{E}[\mathbf{x}])^{\mathrm{T}}\right] = \boldsymbol{\Sigma}$$

Forms of the Multivariate Gaussian

- D in $\boldsymbol{\mu}$ and $D(D+1)/2$ in $\boldsymbol{\Sigma}$, a total of $D(D+3)/2$ parameters
- many parameters for large D and still unimodal

- **Diagonal $\boldsymbol{\Sigma}$**
  - D in $\boldsymbol{\mu}$ and D in $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_i^2)$, a total of $2D$ parameters
- **Isotropic $\boldsymbol{\Sigma}$**
  - D in $\boldsymbol{\mu}$ and 1 in $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$, a total of $D+1$ parameters



Partitioned Gaussian Distributions

Partitioned Gaussian Distributions

- if two sets of variables are jointly Gaussian …
- … so is the conditional distribution of one on the other
- … so is the marginal distribution of either set

joint set $$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

partitioning $$\mathbf{x} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{pmatrix} \qquad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}$$

$$(\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^{\mathrm{T}}) \implies (\boldsymbol{\Sigma}_{aa} = \boldsymbol{\Sigma}_{aa}^{\mathrm{T}}) \text{ and } (\boldsymbol{\Sigma}_{bb} = \boldsymbol{\Sigma}_{bb}^{\mathrm{T}}) \text{ and } (\boldsymbol{\Sigma}_{ba} = \boldsymbol{\Sigma}_{ab}^{\mathrm{T}})$$

precision matrix $$\boldsymbol{\Lambda} \equiv \boldsymbol{\Sigma}^{-1} \qquad \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}$$

$$(\boldsymbol{\Lambda} = \boldsymbol{\Lambda}^{\mathrm{T}}) \implies (\boldsymbol{\Lambda}_{aa} = \boldsymbol{\Lambda}_{aa}^{\mathrm{T}}) \text{ and } (\boldsymbol{\Lambda}_{bb} = \boldsymbol{\Lambda}_{bb}^{\mathrm{T}}) \text{ and } (\boldsymbol{\Lambda}_{ba} = \boldsymbol{\Lambda}_{ab}^{\mathrm{T}})$$

Exponent of the Gaussian

completing the square!

$$-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) = \underbrace{-\frac{1}{2}\mathbf{x}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{x}}_{\text{2nd order}} + \underbrace{\mathbf{x}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}_{\text{1st order}} \underbrace{-\frac{1}{2}\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}_{\text{constant}}$$

$$\begin{aligned}
-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) &= -\frac{1}{2}(\mathbf{x}_a-\boldsymbol{\mu}_a)^{\mathrm{T}}\boldsymbol{\Lambda}_{aa}(\mathbf{x}_a-\boldsymbol{\mu}_a) \\
&\quad -\frac{1}{2}(\mathbf{x}_a-\boldsymbol{\mu}_a)^{\mathrm{T}}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b-\boldsymbol{\mu}_b) \\
&\quad -\frac{1}{2}(\mathbf{x}_b-\boldsymbol{\mu}_b)^{\mathrm{T}}\boldsymbol{\Lambda}_{ba}(\mathbf{x}_a-\boldsymbol{\mu}_a) \\
&\quad -\frac{1}{2}(\mathbf{x}_b-\boldsymbol{\mu}_b)^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}(\mathbf{x}_b-\boldsymbol{\mu}_b)
\end{aligned}$$

2nd order $\mathbf{x}_a$: $\ldots -\frac{1}{2}\mathbf{x}_a^{\mathrm{T}}\underbrace{\boldsymbol{\Lambda}_{aa}}_{\boldsymbol{\Sigma}_{a|b}^{-1}}\mathbf{x}_a\ldots$ 1st order $\mathbf{x}_a$: $\ldots\mathbf{x}_a^{\mathrm{T}}\underbrace{\left(\boldsymbol{\Lambda}_{aa}\boldsymbol{\mu}_a-\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b-\boldsymbol{\mu}_b)\right)}_{\boldsymbol{\Sigma}_{a|b}^{-1}\boldsymbol{\mu}_{a|b}}\ldots$

Partitioned Conditional Gaussian

$$p(\mathbf{x}_a|\mathbf{x}_b) = \mathcal{N}(\mathbf{x}_a|\boldsymbol{\mu}_{a|b}, \boldsymbol{\Sigma}_{a|b}) = \frac{1}{(2\pi)^{M/2}} \frac{1}{|\boldsymbol{\Sigma}_{a|b}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}_a - \boldsymbol{\mu}_{a|b})^{\mathrm{T}} \boldsymbol{\Sigma}_{a|b}^{-1} (\mathbf{x}_a - \boldsymbol{\mu}_{a|b})\right\}$$

$$\begin{aligned}
\boldsymbol{\Sigma}_{a|b} &= \boldsymbol{\Lambda}_{aa}^{-1} = \boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba} \\
\boldsymbol{\mu}_{a|b} &= \boldsymbol{\Sigma}_{a|b}\left\{\boldsymbol{\Lambda}_{aa}\boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b)\right\} \\
&= \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b) \\
&= \boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b)
\end{aligned}$$

$$\begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}$$

$$\begin{aligned}
\boldsymbol{\Lambda}_{aa} &= (\boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba})^{-1} \\
\boldsymbol{\Lambda}_{ab} &= -(\boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba})^{-1}\boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}
\end{aligned}$$

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{M} & -\mathbf{M}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}\mathbf{M} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}\mathbf{M}\mathbf{B}\mathbf{D}^{-1} \end{pmatrix}$$

(Schur complement)
$\mathbf{M} = (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}$

Exponent of the Gaussian (again)

completing the square!

$$-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) = \underbrace{-\frac{1}{2}\mathbf{x}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{x}}_{\text{2nd order}} + \underbrace{\mathbf{x}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}_{\text{1st order}} - \underbrace{\frac{1}{2}\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}_{\text{constant}}$$

$$\begin{aligned}
-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) = & -\frac{1}{2}(\mathbf{x}_a-\boldsymbol{\mu}_a)^{\mathrm{T}}\boldsymbol{\Lambda}_{aa}(\mathbf{x}_a-\boldsymbol{\mu}_a) \\
& -\frac{1}{2}(\mathbf{x}_a-\boldsymbol{\mu}_a)^{\mathrm{T}}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b-\boldsymbol{\mu}_b) \\
& -\frac{1}{2}(\mathbf{x}_b-\boldsymbol{\mu}_b)^{\mathrm{T}}\boldsymbol{\Lambda}_{ba}(\mathbf{x}_a-\boldsymbol{\mu}_a) \\
& -\frac{1}{2}(\mathbf{x}_b-\boldsymbol{\mu}_b)^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}(\mathbf{x}_b-\boldsymbol{\mu}_b)
\end{aligned}$$

2nd order $\mathbf{x}_b$: $\ldots -\frac{1}{2}\mathbf{x}_b^{\mathrm{T}}\underbrace{\boldsymbol{\Lambda}_{bb}}_{\boldsymbol{\Sigma}_{b|a}^{-1}}\mathbf{x}_b\ldots$ 1st order $\mathbf{x}_b$: $\ldots\mathbf{x}_b^{\mathrm{T}}\underbrace{\left(\boldsymbol{\Lambda}_{bb}\boldsymbol{\mu}_b-\boldsymbol{\Lambda}_{ba}(\mathbf{x}_a-\boldsymbol{\mu}_a)\right)}_{\boldsymbol{\Sigma}_{b|a}^{-1}\boldsymbol{\mu}_{b|a}}\ldots$

Partitioned Marginal Gaussian

$$p(\mathbf{x}_a) = \int p(\mathbf{x}_a, \mathbf{x}_b)\, \mathrm{d}\mathbf{x}_b$$ (aim to integrate out $\mathbf{x}_b$)

$$-\frac{1}{2}\mathbf{x}_b^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}\mathbf{x}_b + \mathbf{x}_b^{\mathrm{T}}\Big(\boldsymbol{\Lambda}_{bb}\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_{ba}(\mathbf{x}_a - \boldsymbol{\mu}_a)\Big)$$ (terms with $\mathbf{x}_b$)

$$= -\frac{1}{2}\mathbf{x}_b^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}\mathbf{x}_b + \mathbf{x}_b^{\mathrm{T}}\mathbf{m}$$ $\mathbf{m} = \boldsymbol{\Lambda}_{bb}\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_{ba}(\mathbf{x}_a - \boldsymbol{\mu}_a)$

$$= -\frac{1}{2}\mathbf{x}_b^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}\mathbf{x}_b + \frac{1}{2}\mathbf{x}_b^{\mathrm{T}}\mathbf{m} + \frac{1}{2}\mathbf{m}^{\mathrm{T}}\mathbf{x}_b$$

$$= -\frac{1}{2}\mathbf{x}_b^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}\mathbf{x}_b + \frac{1}{2}\mathbf{x}_b^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}\boldsymbol{\Lambda}_{bb}^{-1}\mathbf{m} + \frac{1}{2}\mathbf{m}^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}^{-1}\boldsymbol{\Lambda}_{bb}\mathbf{x}_b - \frac{1}{2}\mathbf{m}^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}^{-1}\mathbf{m} + \frac{1}{2}\mathbf{m}^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}^{-1}\mathbf{m}$$

$$= -\frac{1}{2}\left(\mathbf{x}_b^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}\mathbf{x}_b - \mathbf{x}_b^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}\boldsymbol{\Lambda}_{bb}^{-1}\mathbf{m} - \mathbf{m}^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}^{-1}\boldsymbol{\Lambda}_{bb}\mathbf{x}_b + \mathbf{m}^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}^{-1}\mathbf{m}\right) + \frac{1}{2}\mathbf{m}^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}^{-1}\mathbf{m}$$

$$= -\frac{1}{2}(\mathbf{x}_b - \boldsymbol{\Lambda}_{bb}^{-1}\mathbf{m})^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}(\mathbf{x}_b - \boldsymbol{\Lambda}_{bb}^{-1}\mathbf{m}) + \frac{1}{2}\mathbf{m}^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}^{-1}\mathbf{m}$$

(still have to consider this, depends only on $\mathbf{x}_a$, not $\mathbf{x}_b$)

$$\int \exp\left\{-\frac{1}{2}(\mathbf{x}_b - \boldsymbol{\Lambda}_{bb}^{-1}\mathbf{m})^{\mathrm{T}}\boldsymbol{\Lambda}_{bb}(\mathbf{x}_b - \boldsymbol{\Lambda}_{bb}^{-1}\mathbf{m})\right\}\mathrm{d}\mathbf{x}_b$$

(unnormalized Gaussian, integral independent of the mean and thus of $\mathbf{x}_a$, evaluates to the reciprocal of the normalizer)

Partitioned Marginal Gaussian

$$p(\mathbf{x}_a) = \int p(\mathbf{x}_a, \mathbf{x}_b)\, \mathrm{d}\mathbf{x}_b = \frac{1}{(2\pi)^{M/2}} \frac{1}{|\boldsymbol{\Sigma}_a|^{1/2}} \exp\left\{ -\frac{1}{2}(\mathbf{x}_a - \boldsymbol{\mu}_a)^{\mathrm{T}} \boldsymbol{\Sigma}_a^{-1} (\mathbf{x}_a - \boldsymbol{\mu}_a) \right\}$$

$$\frac{1}{2}\mathbf{m}^{\mathrm{T}} \boldsymbol{\Lambda}_{bb}^{-1} \mathbf{m} \qquad \mathbf{m} = \boldsymbol{\Lambda}_{bb}\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_{ba}(\mathbf{x}_a - \boldsymbol{\mu}_a)$$

(from previous page)

completing the square now for $\mathbf{x}_a$!

$$\frac{1}{2}\left[\boldsymbol{\Lambda}_{bb}\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_{ba}(\mathbf{x}_a - \boldsymbol{\mu}_a)\right]^{\mathrm{T}} \boldsymbol{\Lambda}_{bb}^{-1} \left[\boldsymbol{\Lambda}_{bb}\boldsymbol{\mu}_b - \boldsymbol{\Lambda}_{ba}(\mathbf{x}_a - \boldsymbol{\mu}_a)\right]$$

$$-\frac{1}{2}\mathbf{x}_a^{\mathrm{T}} \boldsymbol{\Lambda}_{aa} \mathbf{x}_a + \mathbf{x}_a^{\mathrm{T}} (\boldsymbol{\Lambda}_{aa}\boldsymbol{\mu}_a + \boldsymbol{\Lambda}_{ab}\boldsymbol{\mu}_b) + \mathrm{const}$$

(+ other terms with $\mathbf{x}_a$)

$$= \quad -\frac{1}{2}\mathbf{x}_a^{\mathrm{T}} (\boldsymbol{\Lambda}_{aa} - \boldsymbol{\Lambda}_{ab}\boldsymbol{\Lambda}_{bb}^{-1}\boldsymbol{\Lambda}_{ba}) \mathbf{x}_a$$

(2nd order term of $\mathbf{x}_a$)

$$+\mathbf{x}_a^{\mathrm{T}} (\boldsymbol{\Lambda}_{aa} - \boldsymbol{\Lambda}_{ab}\boldsymbol{\Lambda}_{bb}^{-1}\boldsymbol{\Lambda}_{ba}) \quad \boldsymbol{\mu}_a + \mathrm{const}$$

(1st order term of $\mathbf{x}_a$)

$$\boldsymbol{\Sigma}_a = (\boldsymbol{\Lambda}_{aa} - \boldsymbol{\Lambda}_{ab}\boldsymbol{\Lambda}_{bb}^{-1}\boldsymbol{\Lambda}_{ba})^{-1} = \boldsymbol{\Sigma}_{aa} \qquad \boldsymbol{\Sigma}_a (\boldsymbol{\Lambda}_{aa} - \boldsymbol{\Lambda}_{ab}\boldsymbol{\Lambda}_{bb}^{-1}\boldsymbol{\Lambda}_{ba})\boldsymbol{\mu}_a = \boldsymbol{\mu}_a$$

Partitioned Gaussians Summary

Given a joint Gaussian distribution $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\Lambda} \equiv \boldsymbol{\Sigma}^{-1}$ and

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{pmatrix}, \qquad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix}$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}, \qquad \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}$$

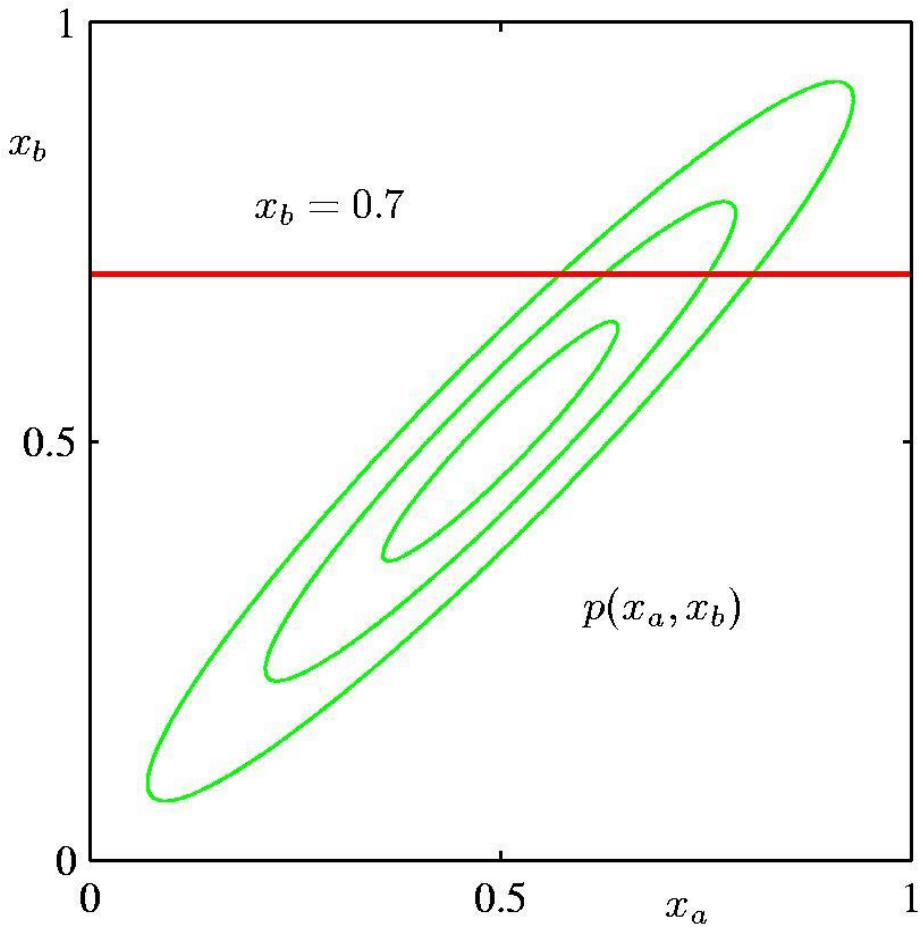
Conditional distribution:

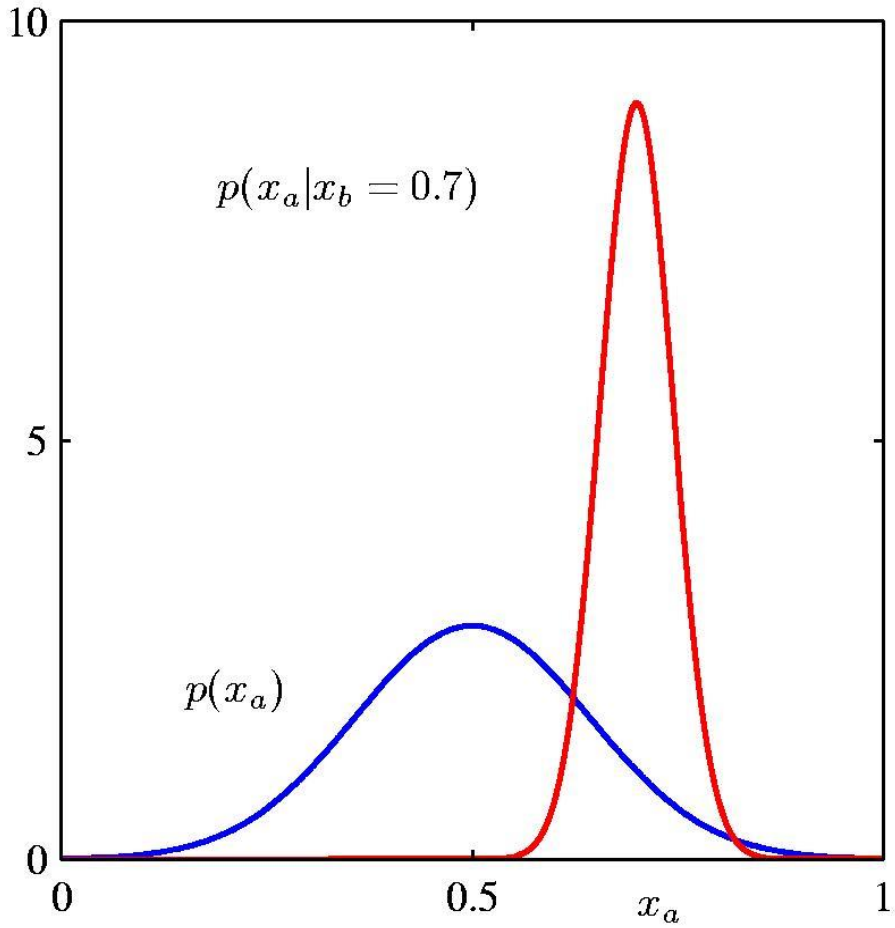
$$\begin{aligned} p(\mathbf{x}_a|\mathbf{x}_b) &= \mathcal{N}(\mathbf{x}_a|\boldsymbol{\mu}_{a|b}, \boldsymbol{\Lambda}_{aa}^{-1}) \\ \boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b). \end{aligned}$$

Marginal distribution:

$$p(\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_a|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_{aa}).$$

Partitioned Gaussians Example





Bayes' Theorem for Gaussians

Linear Gaussian Model

- **Given**

$$
\begin{aligned}
p(\mathbf{x}) &= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right) \\
p(\mathbf{y}|\mathbf{x}) &= \mathcal{N}\left(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}\right)
\end{aligned}
$$

  - *M*-dimensional $\mathbf{x}$ and *D*-dimensional $\mathbf{y}$
  - $\mathbf{A}$ is a *D*×*M* matrix and $\mathbf{b}$ is *D*-dimensional
- **Goal**
  - find the marginal distribution $p(\mathbf{y})$
  - find the conditional distribution $p(\mathbf{x}|\mathbf{y})$

Joint Distribution

$$\mathbf{z} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$$

$$\begin{aligned} \ln p(\mathbf{z}) &= \ln p(\mathbf{x}) + \ln p(\mathbf{y}|\mathbf{x}) \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Lambda}(\mathbf{x}-\boldsymbol{\mu}) - \frac{1}{2}(\mathbf{y}-\mathbf{A}\mathbf{x}-\mathbf{b})^{\mathrm{T}}\mathbf{L}(\mathbf{y}-\mathbf{A}\mathbf{x}-\mathbf{b}) + \text{constant} \end{aligned}$$

quadratic function of the components of $\mathbf{z}$, hence $p(\mathbf{z})$ is Gaussian

recall "completing the square"!

$$-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) = \underbrace{-\frac{1}{2}\mathbf{x}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{x}}_{\text{2nd order}} + \underbrace{\mathbf{x}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}_{\text{1st order}} - \underbrace{\frac{1}{2}\boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}_{\text{constant}}$$

Covariance of the Joint Distribution

$$\begin{aligned}\ln p(\mathbf{z}) &= \ln p(\mathbf{x}) + \ln p(\mathbf{y}|\mathbf{x}) \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Lambda}(\mathbf{x}-\boldsymbol{\mu}) - \frac{1}{2}(\mathbf{y}-\mathbf{A}\mathbf{x}-\mathbf{b})^{\mathrm{T}}\mathbf{L}(\mathbf{y}-\mathbf{A}\mathbf{x}-\mathbf{b}) + \text{constant}\end{aligned}$$

considering the second-order terms

$$\begin{aligned}&-\frac{1}{2}\mathbf{x}^{\mathrm{T}}(\boldsymbol{\Lambda}+\mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})\mathbf{x} - \frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{L}\mathbf{y} + \frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{L}\mathbf{A}\mathbf{x} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{y} \\ &= -\frac{1}{2}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix}^{\mathrm{T}}\begin{pmatrix}\boldsymbol{\Lambda}+\mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A} & -\mathbf{A}^{\mathrm{T}}\mathbf{L}\\ -\mathbf{L}\mathbf{A} & \mathbf{L}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} = -\frac{1}{2}\mathbf{z}^{\mathrm{T}}\mathbf{R}\mathbf{z}\end{aligned}$$

$$\mathbf{R} = \begin{pmatrix}\boldsymbol{\Lambda}+\mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A} & -\mathbf{A}^{\mathrm{T}}\mathbf{L}\\ -\mathbf{L}\mathbf{A} & \mathbf{L}\end{pmatrix} \qquad \mathrm{cov}[\mathbf{z}] = \mathbf{R}^{-1} = \begin{pmatrix}\boldsymbol{\Lambda}^{-1} & \boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}\\ \mathbf{A}\boldsymbol{\Lambda}^{-1} & \mathbf{L}^{-1}+\mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}\end{pmatrix}$$

$$\begin{pmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{pmatrix}^{-1} = \begin{pmatrix}\mathbf{M} & -\mathbf{M}\mathbf{B}\mathbf{D}^{-1}\\ -\mathbf{D}^{-1}\mathbf{C}\mathbf{M} & \mathbf{D}^{-1}+\mathbf{D}^{-1}\mathbf{C}\mathbf{M}\mathbf{B}\mathbf{D}^{-1}\end{pmatrix} \qquad \mathbf{M} = (\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}$$

Mean of the Joint Distribution

$$\begin{aligned}\ln p(\mathbf{z}) &= \ln p(\mathbf{x}) + \ln p(\mathbf{y}|\mathbf{x}) \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Lambda}(\mathbf{x}-\boldsymbol{\mu}) - \frac{1}{2}(\mathbf{y}-\mathbf{A}\mathbf{x}-\mathbf{b})^{\mathrm{T}}\mathbf{L}(\mathbf{y}-\mathbf{A}\mathbf{x}-\mathbf{b}) + \text{constant}\end{aligned}$$

considering the first-order terms

$$\mathbf{x}^{\mathrm{T}}\boldsymbol{\Lambda}\boldsymbol{\mu} - \mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{b} + \mathbf{y}^{\mathrm{T}}\mathbf{L}\mathbf{b} = \begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix}^{\mathrm{T}}\begin{pmatrix}\boldsymbol{\Lambda}\boldsymbol{\mu} - \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{b}\\ \mathbf{L}\mathbf{b}\end{pmatrix}$$

$$\mathbb{E}[\mathbf{z}] = \mathbf{R}^{-1}\begin{pmatrix}\boldsymbol{\Lambda}\boldsymbol{\mu} - \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{b}\\ \mathbf{L}\mathbf{b}\end{pmatrix}$$

$$\mathbf{R}^{-1} = \begin{pmatrix}\boldsymbol{\Lambda}^{-1} & \boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}\\ \mathbf{A}\boldsymbol{\Lambda}^{-1} & \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}\end{pmatrix}$$

$$\mathbb{E}[\mathbf{z}] = \begin{pmatrix}\boldsymbol{\mu}\\ \mathbf{A}\boldsymbol{\mu} + \mathbf{b}\end{pmatrix}$$

Recall: Partitioned Gaussians

Given a joint Gaussian distribution $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\Lambda} \equiv \boldsymbol{\Sigma}^{-1}$ and

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{pmatrix}, \qquad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix}$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}, \qquad \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}$$

Conditional distribution:

$$\begin{aligned} p(\mathbf{x}_a|\mathbf{x}_b) &= \mathcal{N}(\mathbf{x}_a|\boldsymbol{\mu}_{a|b}, \boldsymbol{\Lambda}_{aa}^{-1}) \\ \boldsymbol{\mu}_{a|b} &= \boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b). \end{aligned}$$

Marginal distribution:

$$p(\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_a|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_{aa}).$$

Marginal and Conditional

$$\mathbf{z} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \qquad \mathbb{E}[\mathbf{z}] = \begin{pmatrix} \boldsymbol{\mu} \\ \mathbf{A}\boldsymbol{\mu} + \mathbf{b} \end{pmatrix} \qquad \mathrm{cov}[\mathbf{z}] = \mathbf{R}^{-1} = \begin{pmatrix} \boldsymbol{\Lambda}^{-1} & \boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}} \\ \mathbf{A}\boldsymbol{\Lambda}^{-1} & \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}} \end{pmatrix}$$

$$\mathbf{R} = \begin{pmatrix} \boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A} & -\mathbf{A}^{\mathrm{T}}\mathbf{L} \\ -\mathbf{L}\mathbf{A} & \mathbf{L} \end{pmatrix}$$

now, apply the results for partitioned Gaussians

$$\begin{aligned} \mathbb{E}[\mathbf{y}] &= \mathbf{A}\boldsymbol{\mu} + \mathbf{b} \\ \mathrm{cov}[\mathbf{y}] &= \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}} \end{aligned}$$

$$\begin{aligned} \mathbb{E}[\mathbf{x}|\mathbf{y}] &= (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}\left\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\right\} \\ \mathrm{cov}[\mathbf{x}|\mathbf{y}] &= (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}. \end{aligned}$$

$$\boldsymbol{\Sigma}_{\mathbf{x}|\mathbf{y}} = \mathbf{R}_{\mathbf{xx}}^{-1} = (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1} \qquad \boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}} = \boldsymbol{\mu}_{\mathbf{x}} - \mathbf{R}_{\mathbf{xx}}^{-1}\mathbf{R}_{\mathbf{xy}}(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}}) = \boldsymbol{\mu} - (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}(-\mathbf{A}^{\mathrm{T}}\mathbf{L})(\mathbf{y} - \mathbf{A}\boldsymbol{\mu} - \mathbf{b})$$

$$= (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}\left\{(\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})\boldsymbol{\mu} - \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A}\boldsymbol{\mu} + \mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b})\right\} = (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}\left\{\boldsymbol{\Lambda}\boldsymbol{\mu} + \mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b})\right\}$$

Bayes' Theorem for Gaussian Variables

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y})$$

- **Given**

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right) \\ p(\mathbf{y}|\mathbf{x}) &= \mathcal{N}\left(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}\right) \end{aligned}$$

- **we have**

$$\begin{aligned} p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \\ p(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \end{aligned}$$

- **where**

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}$$

Application

Bayesian Linear Regression

- **Conjugate prior over parameters $\mathbf{w}$**

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$$

- **Data $\{\mathbf{x}_i, t_i\}$ Likelihood**

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) = \mathcal{N}(\mathbf{t}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I})$$

- **Find posterior**

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{t}|\mathbf{w})p(\mathbf{w})$$

- **Reminder: Bayes' Theorem for Gaussians**

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$$

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right) \\ p(\mathbf{y}|\mathbf{x}) &= \mathcal{N}\left(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}\right) \end{aligned}$$

$$\begin{aligned} p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \\ p(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \\ \boldsymbol{\Sigma} &= (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1} \end{aligned}$$

Bayesian Linear Regression (apply)

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0) \qquad p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{t}|\mathbf{w})p(\mathbf{w})$$

$$p(\mathbf{t}|\mathbf{w}) = \mathcal{N}(\mathbf{t}|\mathbf{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I})$$

$$\begin{aligned} p(\mathbf{w}|\mathbf{t}) &= \mathcal{N}\left(\mathbf{w}|\mathbf{\Sigma}\left\{\mathbf{\Phi}^{\mathrm{T}}(\beta^{-1}\mathbf{I})^{-1}(\mathbf{t}-\mathbf{0}) + \mathbf{S}_0^{-1}\mathbf{m}_0\right\}, \mathbf{\Sigma}\right) \\ &= \mathcal{N}\left(\mathbf{w}|(\mathbf{S}_0^{-1} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\left\{\beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} + \mathbf{S}_0^{-1}\mathbf{m}_0\right\}, (\mathbf{S}_0^{-1} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\right) \end{aligned}$$

$$\mathbf{\Sigma} = (\mathbf{S}_0^{-1} + \mathbf{\Phi}^{\mathrm{T}}(\beta^{-1}\mathbf{I})^{-1}\mathbf{\Phi})^{-1} = (\mathbf{S}_0^{-1} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}$$

- **Reminder: Bayes' Theorem for Gaussians**

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$$

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, \mathbf{\Lambda}^{-1}\right) \\ p(\mathbf{y}|\mathbf{x}) &= \mathcal{N}\left(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}\right) \end{aligned}$$

$$\begin{aligned} p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\mathbf{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \\ p(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\mathbf{x}|\mathbf{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y}-\mathbf{b}) + \mathbf{\Lambda}\boldsymbol{\mu}\}, \mathbf{\Sigma}) \\ \mathbf{\Sigma} &= (\mathbf{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1} \end{aligned}$$

Bayesian Linear Regression (result)

- **Conjugate prior over parameters $\mathbf{w}$**

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$$

- **Data $\{\mathbf{x}_i, t_i\}$ Likelihood**

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) = \mathcal{N}(\mathbf{t}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I})$$

- **Find posterior**

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{t}|\mathbf{w})p(\mathbf{w})$$

- **Result**

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

$$\begin{aligned} \mathbf{m}_N &= \mathbf{S}_N\left(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}\right) \\ \mathbf{S}_N^{-1} &= \mathbf{S}_0^{-1} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} \end{aligned}$$

Bayesian Linear Regression (specific)

- **Conjugate prior over parameters $\mathbf{w}$**

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

- **Data $\{\mathbf{x}_i, t_i\}$ Likelihood**

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) = \mathcal{N}(\mathbf{t}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I})$$

- **Find posterior**

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{t}|\mathbf{w})p(\mathbf{w})$$

- **Result**

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

$$\begin{aligned} \mathbf{m}_N &= \beta\mathbf{S}_N\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha\mathbf{I} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} \end{aligned}$$

Predictive Distribution

- **Conjugate prior over parameters $\mathbf{w}$**

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

- **Data $\{\mathbf{x}_i, t_i\}$ Likelihood**

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) = \mathcal{N}(\mathbf{t}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I})$$

- **Posterior**

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

$$\begin{aligned} \mathbf{m}_N &= \beta\mathbf{S}_N\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha\mathbf{I} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} \end{aligned}$$

- **Predictive Distribution**

$$p(t|\mathbf{t}, \mathbf{x}) = \int p(t|\mathbf{w}, \mathbf{x})p(\mathbf{w}|\mathbf{t})\,\mathrm{d}\mathbf{w} = \int \mathcal{N}(t|\boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{w}, \beta^{-1})\mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)\,\mathrm{d}\mathbf{w}$$

Predictive Distribution (apply)

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

$$p(t|\mathbf{w}, \mathbf{x}) = \mathcal{N}(t|\boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{w}, \beta^{-1})$$

$$\begin{aligned} p(\mathbf{w}) &= \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \\ \mathbf{m}_N &= \beta\mathbf{S}_N\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha\mathbf{I} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} \end{aligned}$$

$$\begin{aligned} p(t|\mathbf{t}, \mathbf{x}) &= \int \mathcal{N}(t|\boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{w}, \beta^{-1})\mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)\,\mathrm{d}\mathbf{w} \\ &= \mathcal{N}\left(t|\boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{m}_N, \beta^{-1} + \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{S}_N\boldsymbol{\phi}(\mathbf{x})\right) \\ &= \mathcal{N}\left(t|\boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\beta\mathbf{S}_N\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}, \beta^{-1} + \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}(\alpha\mathbf{I} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\phi}(\mathbf{x})\right) \end{aligned}$$

- **Reminder: Bayes' Theorem for Gaussians**

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$$

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right) \\ p(\mathbf{y}|\mathbf{x}) &= \mathcal{N}\left(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}\right) \end{aligned}$$

$$\boxed{\begin{aligned} p(\mathbf{y}) &= \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \end{aligned}}$$

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \\ \boldsymbol{\Sigma} &= (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1} \end{aligned}$$

Predictive Distribution (result)

- **Conjugate prior over parameters $\mathbf{w}$**

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

- **Data $\{\mathbf{x}_i, t_i\}$ Likelihood**

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) = \mathcal{N}(\mathbf{t}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I})$$

- **Posterior**

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

$$\begin{aligned} \mathbf{m}_N &= \beta\mathbf{S}_N\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha\mathbf{I} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} \end{aligned}$$

- **Predictive Distribution**

$$\begin{aligned} p(t|\mathbf{t}, \mathbf{x}) &= \int p(t|\mathbf{w}, \mathbf{x})p(\mathbf{w}|\mathbf{t})\,\mathrm{d}\mathbf{w} = \int \mathcal{N}(t|\boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{w}, \beta^{-1})\mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)\,\mathrm{d}\mathbf{w} \\ &= \mathcal{N}\left(t|\boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{m}_N, \beta^{-1} + \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{S}_N\boldsymbol{\phi}(\mathbf{x})\right) = \mathcal{N}\left(t|\mu_N(\mathbf{x}), \sigma_N^2(\mathbf{x})\right) \end{aligned}$$

$$\mu_N(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{m}_N \qquad \sigma_N^2(\mathbf{x}) = \beta^{-1} + \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{S}_N\boldsymbol{\phi}(\mathbf{x})$$

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 07

Linear Classification

Linear Discriminant Functions

TUC ECE, Spring 2023

Today

- **Classification**
  - linear models for classification
- **Linear Discriminant Functions**
  - two classes
  - multiple classes

Classification

Classification

- **Classification**
  - assign input vector **x** to one of K discrete classes (C_1, C_2, …,C_K)
  - the input space is partitioned in decision regions …
  - … defined by the decision boundaries or decision surfaces
- **Notation**
  - input: D-dimensional vector **x**
  - output (2 classes): binary variable t
  - output (K classes): K-dimensional vector **t** (1-of-K coding)
- **Linear classification**
  - decision surfaces are linear functions of the input vector **x**
  - (D-1)-dimensional hyperplanes in D-dimensional input space

Linear Models for Classification

- **Generalized linear model**

$$y(\mathbf{x}) = f(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0)$$

  - *f*( ) is a non-linear activation function, typical range: [0,1]
  - non-linear in the parameters due to the activation function
  - decision surfaces (linear in the input): $\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 = \mathrm{constant}$

- **Features of the input**

$$y(\mathbf{x}) = f(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) + w_0)$$

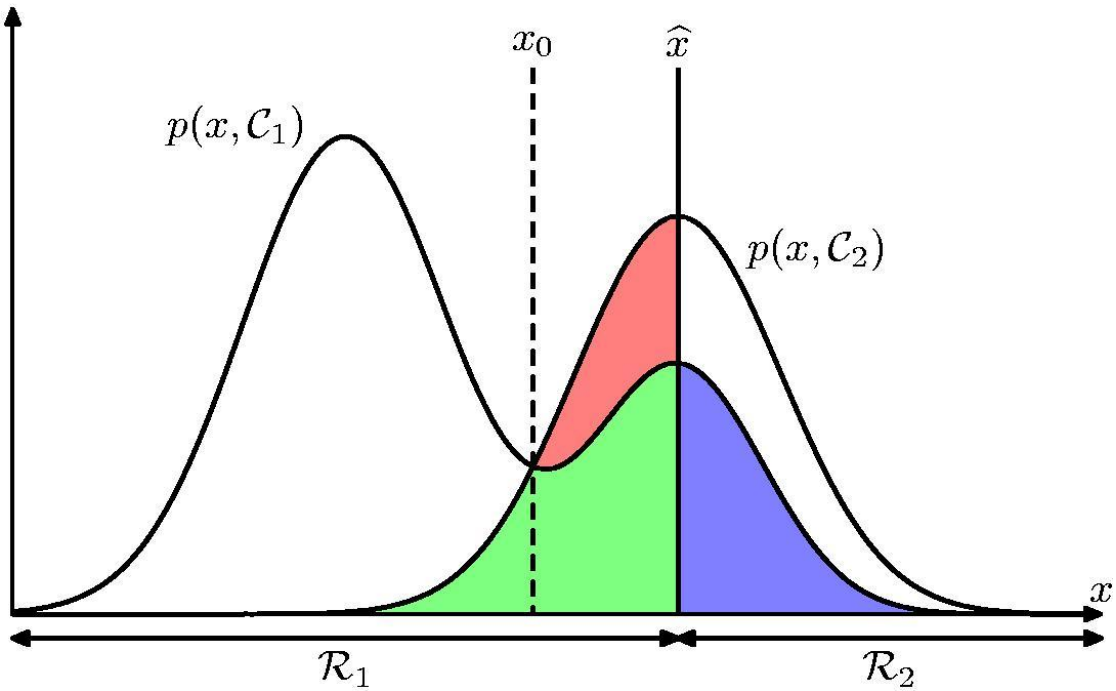
  - *φ*( ) is a vector of (non-linear) basis functions (features)
  - decision surfaces are now non-linear in the input
  - decision surfaces are still linear in the features/parameters

Inference and Decision

- **Inference**
  - input ***x***, target ***t***
  - determine either p(***x***,***t***) or p(***t***|***x***)
  - *example*: probability of cancer, given an x-ray image
- **Decision**
  - for given ***x***, determine optimal decision based on predicted ***t***
  - *example*: give cancer treatment, given an x-ray image
  - the decision step is easy, if we have solved the inference step
- **Example: Classification Rule**
  - decision region R_k over **x**
  - corresponding to class C_k (decision)

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$$

Minimum Misclassification Rate



$$\mathcal{C}^* = \arg\max_k p(\mathbf{x}, \mathcal{C}_k)$$

$$p(\mathbf{x}, \mathcal{C}_k) = p(\mathcal{C}_k|\mathbf{x})p(\mathbf{x})$$

$$\mathcal{C}^* = \arg\max_k p(\mathcal{C}_k|\mathbf{x})$$

Choose the class with the highest posterior probability

$$\begin{aligned} p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\ &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2)\,\mathrm{d}\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1)\,\mathrm{d}\mathbf{x}. \end{aligned}$$

Minimum Expected Loss

- **Loss Function L_{kj}**
  - associates decisions with costs
  - depends on decision and truth

$$\begin{array}{cc} & \text{Decision} \\ & \begin{array}{cc} \text{cancer} & \text{normal} \end{array} \\ \text{Truth} \begin{array}{c} \text{cancer} \\ \text{normal} \end{array} & \begin{pmatrix} 0 & 1000 \\ 1 & 0 \end{pmatrix} \end{array}$$

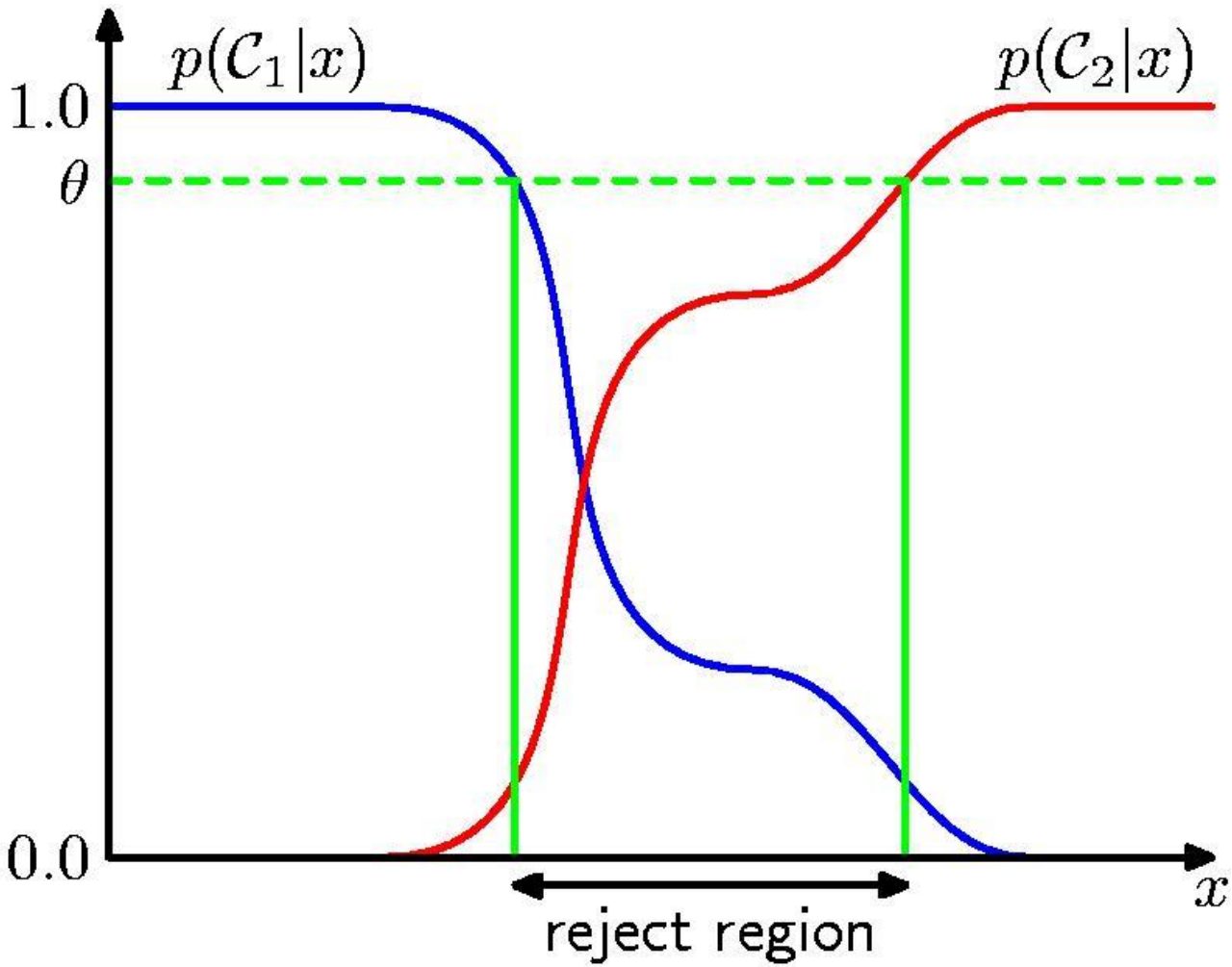
- **Minimum Expected Loss**
  - expected cost due to mistakes

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, \mathcal{C}_k) \, \mathrm{d}\mathbf{x}$$

  - choose regions that minimize the expected loss
- **Decision**
  - choose class that minimizes loss

$$C^* = \arg\min_j \sum_k L_{kj} p(\mathcal{C}_k | \mathbf{x})$$

Reject Option



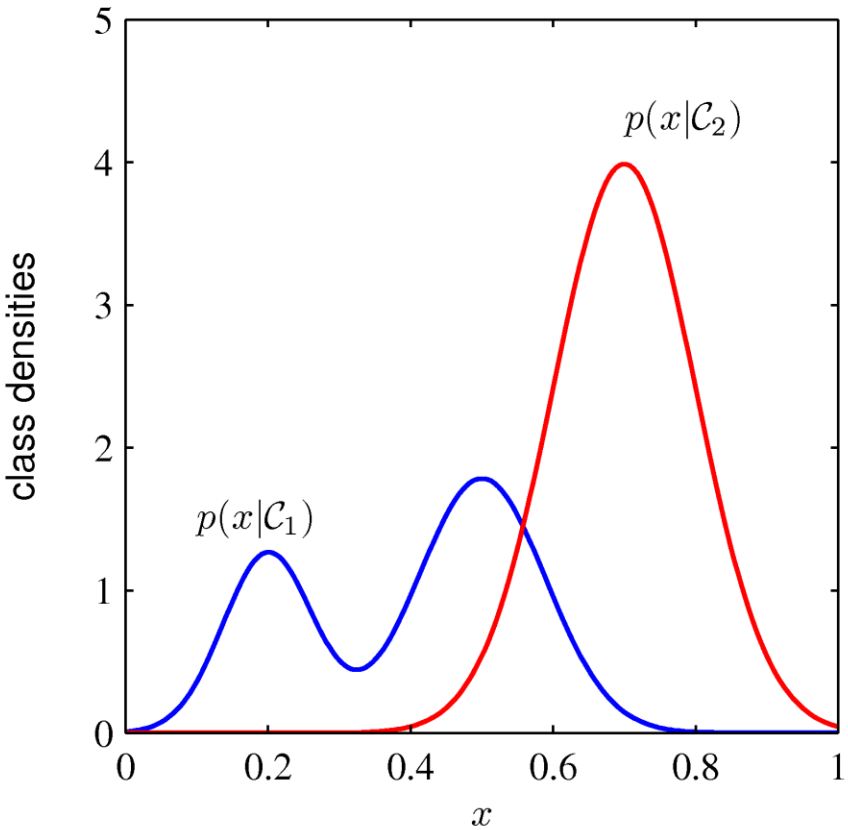
Classification Decision Problem

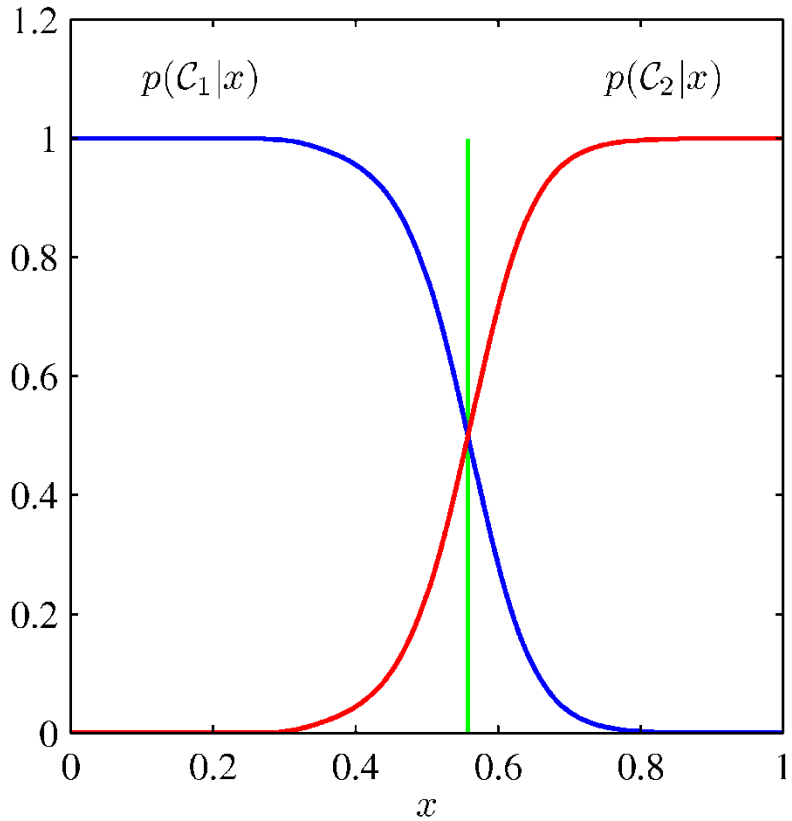
- **Approach I: generative models**
  - determine the class-conditional densities $p(\mathbf{x}|C_k)$
  - infer the posterior class probabilities $p(C_k|\mathbf{x})$ (through Bayes)
  - use decision theory to make decision
- **Approach II: discriminative models**
  - determine the posterior class probabilities $p(C_k|\mathbf{x})$
  - use decision theory to make decision
- **Approach III: discriminant functions**
  - determine a function that maps inputs to classes directly
  - use the discriminant function to make decision

Pros and Cons

- **Approach I: generative models**
  - pros: can be used to generate synthetic data
  - cons: solves a much bigger problem, quite demanding
  - pros: can be used for outlier/novelty detection
- **Approach II: discriminative models**
  - pros: makes no waste of resources, avoids complex models
  - cons: solves a slightly bigger problem
- **Approach III: discriminant functions**
  - pros: a single (learning) problem
  - pros (cons?): may not use probabilities at all!
  - cons: black-box approach, provides no intuition

$p(\mathbf{x}|C_k)$ vs. $p(C_k|\mathbf{x})$ vs. $\mathbf{x}^*$





Separating Inference and Decision

- **Minimizing risk**
  - loss matrix may change over time
  - changes can be easily introduced in the minimum risk criterion
- **Reject option**
  - posterior probabilities allow for a rejection criterion
- **Unbalanced class priors**
  - posterior probabilities can be transformed to other populations
- **Combining models**
  - split the problem, solve separately, model for each sub-problem
  - combine sub-models using probability theory to solve problem
  - exploit conditional independence

Linear Discriminant Functions

Linear Discriminants

- **Linear discriminant function**

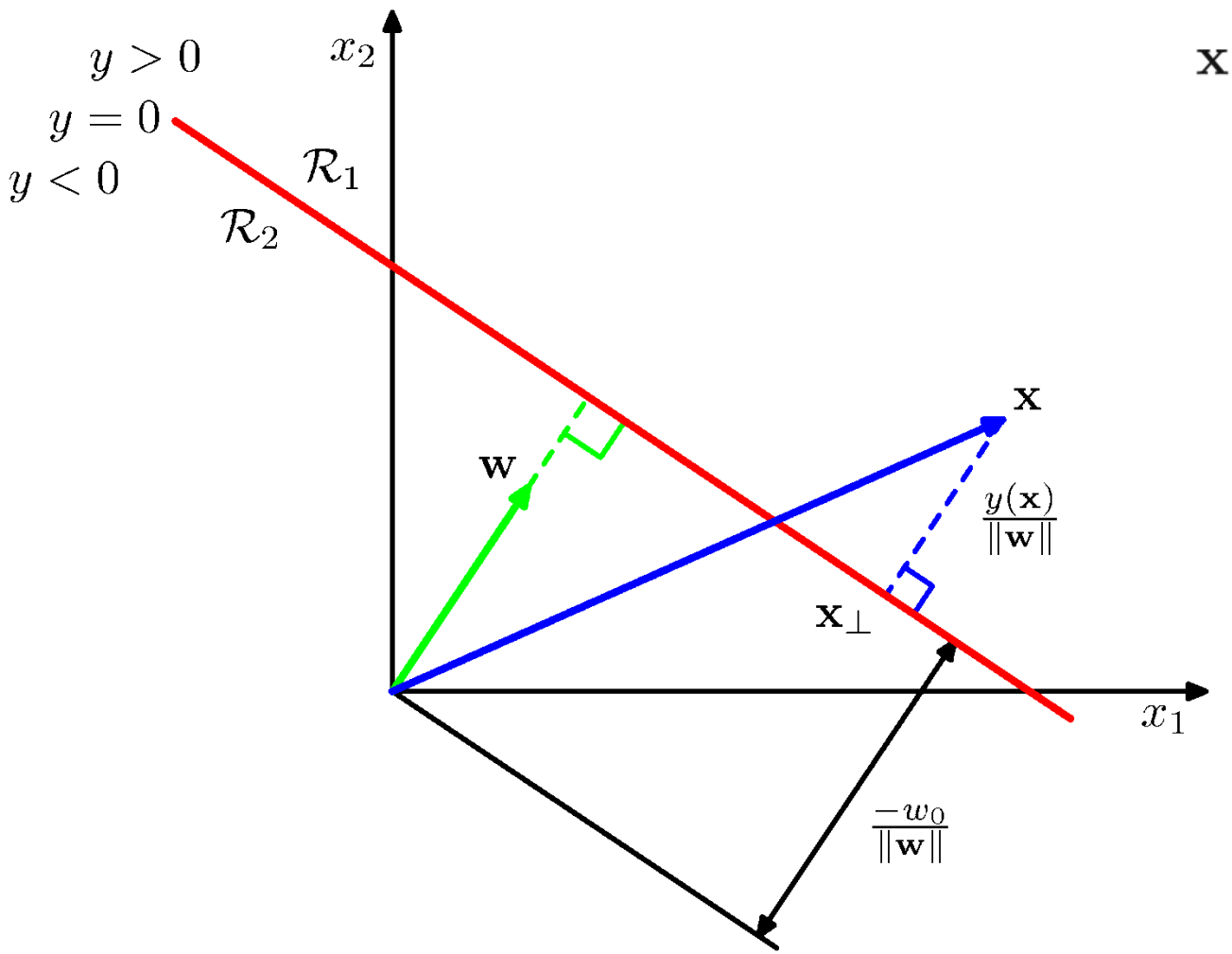
$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

  - $\mathbf{w}$ is the *weight* (parameter) vector
  - w_0 is the *bias* parameter (its negative is the *threshold*)
- **Binary classification decision**
  - choose class C_1, if $y(\mathbf{x}) \geq 0$; choose class C_2, otherwise
- **Decision surface**

$$y(\mathbf{x}) = 0 \Leftrightarrow \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 = 0$$

  - $\mathbf{w}$ defines orientation: $y(\mathbf{x}_A) = y(\mathbf{x}_B) = 0 \implies \mathbf{w}^{\mathrm{T}}(\mathbf{x}_A - \mathbf{x}_B) = 0$
  - w_0 defines location: $y(\mathbf{x}) = 0 \implies \dfrac{\mathbf{w}^{\mathrm{T}}\mathbf{x}}{\| \mathbf{w} \|} = -\dfrac{w_0}{\| \mathbf{w} \|}$

Linear Discriminant Geometry



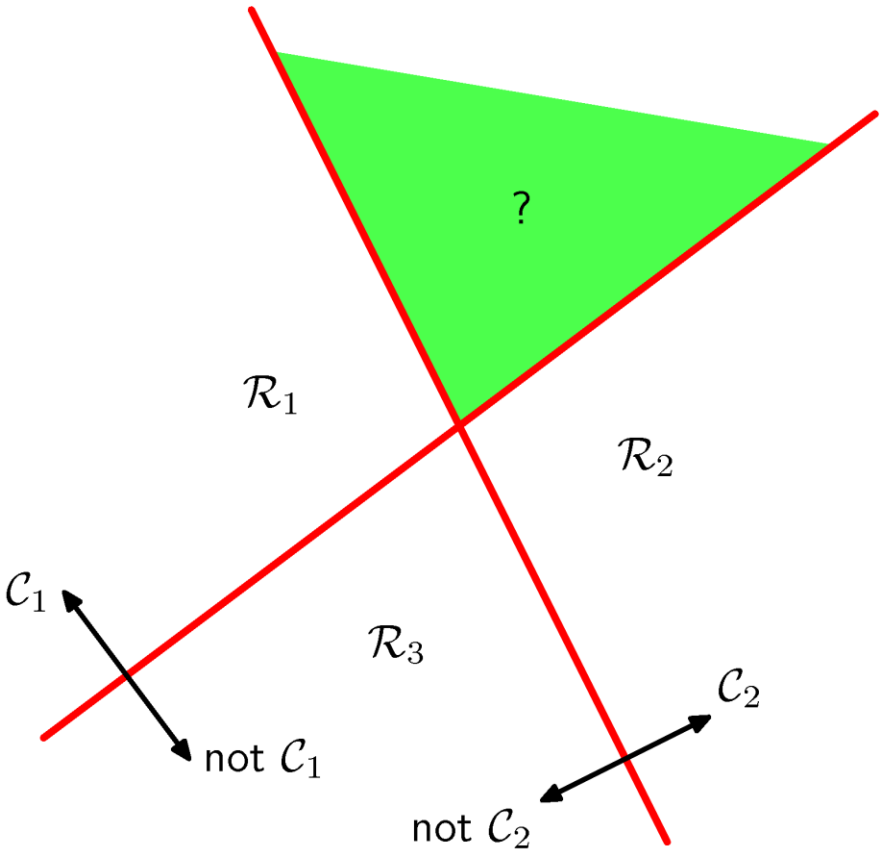
$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

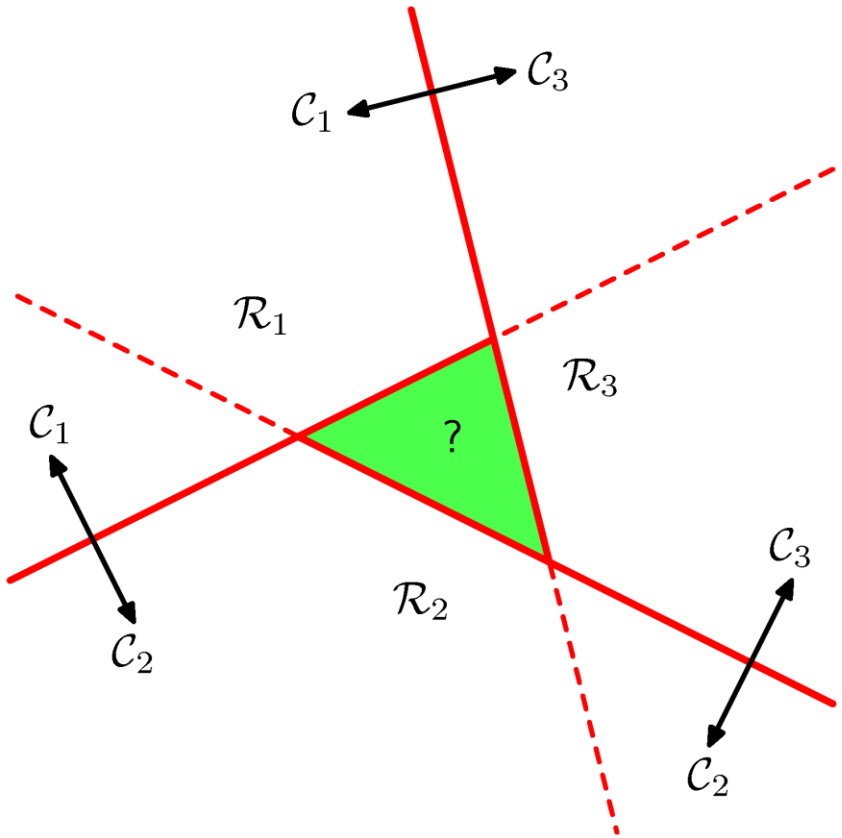
$$r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}$$

Multiple Classes

- **One-versus-the-rest**
  - K-1 binary discriminant functions
  - one for each of the first K-1 classes (within or outside class)
  - assign to last class, if not assigned to any of the previous K-1
  - problem: some regions may be ambiguously classified
- **One-versus-one**
  - $K(K-1)/2$ binary discriminant functions
  - one for each pair of classes (competitive comparison)
  - assign according to a majority vote amongst the discriminants
  - problem: some regions may be ambiguously classified

Three Classes Examples





K-Class Discriminant Function

- **K linear discriminant functions**

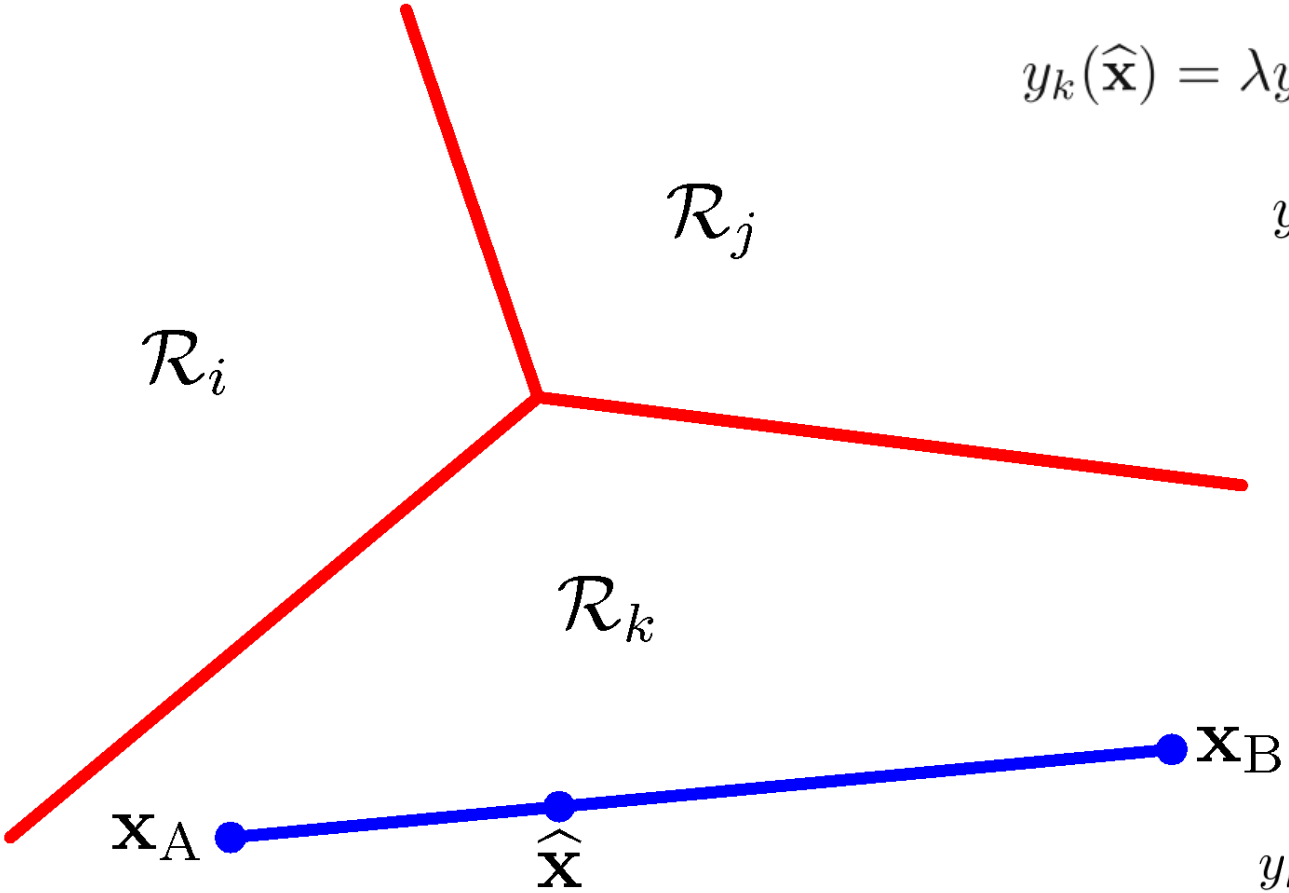
$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$$

  – 1-of-*K* binary coding scheme

- **Classification decision**
  – assign input **x** to class C_k, if $y_k(\mathbf{x}) > y_j(\mathbf{x}),\ \forall j \neq k$

- **Decision boundaries**
  – boundary between classes C_k and C_j : $y_k(\mathbf{x}) = y_j(\mathbf{x})$
  – (*D*-1)-dimensional hyperplane $(\mathbf{w}_k - \mathbf{w}_j)^{\mathrm{T}}\mathbf{x} + (w_{k0} - w_{j0}) = 0$
  – singly connected and convex decision regions

Decision Regions



$$y_k(\widehat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_\mathrm{A}) + (1-\lambda) y_k(\mathbf{x}_\mathrm{B})$$

$$y_k(\widehat{\mathbf{x}}) > y_j(\widehat{\mathbf{x}}), \ \forall j \neq k$$

$$\widehat{\mathbf{x}} \in \mathcal{R}_k$$

$$y_k(\mathbf{x}_A) > y_j(\mathbf{x}_A), \ \forall j \neq k$$

$$\widehat{\mathbf{x}} = \lambda \mathbf{x}_\mathrm{A} + (1-\lambda)\mathbf{x}_\mathrm{B} \quad 0 \leqslant \lambda \leqslant 1$$

$$y_k(\mathbf{x}_B) > y_j(\mathbf{x}_B), \ \forall j \neq k$$

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 08

Linear Discriminant Estimation

TUC ECE, Spring 2023

Today

- **Linear Discriminant Estimation**
  - least-squares
  - Fisher discriminant
  - perceptron

Least-Squares Discriminant

Recall: K-Class Discriminant Function

- **K linear discriminant functions**

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$$

  – 1-of-*K* binary coding scheme

- **Classification decision**
  – assign input **x** to class C_k, if $y_k(\mathbf{x}) > y_j(\mathbf{x}),\ \forall j \neq k$

- **Decision boundaries**
  – boundary between classes C_k and C_j : $y_k(\mathbf{x}) = y_j(\mathbf{x})$
  – (*D*-1)-dimensional hyperplane $(\mathbf{w}_k - \mathbf{w}_j)^{\mathrm{T}}\mathbf{x} + (w_{k0} - w_{j0}) = 0$
  – singly connected and convex decision regions

Notation

- **K-Class Linear Discriminant**

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$$

  – 1-of-*K* binary coding scheme

- **Matrix notation**

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}}$$

  – augmented input $\widetilde{\mathbf{x}}$ with dummy x_0 = 1 for the bias weight w_{k0}
  – columns of (*D*+1)×*K* matrix $\widetilde{\mathbf{W}}$: the *K* augmented weight vectors
  – assign input to class corresponding to largest-valued output

- **Given data**
  – input $\widetilde{\mathbf{X}}$ and target $\mathbf{T}$ matrices with inputs and targets in rows

Least-Squares Discriminant

- **Sum-of-squares error**

$$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\left\{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^{\mathrm{T}}(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})\right\}$$

- **Pseudoinverse solution**

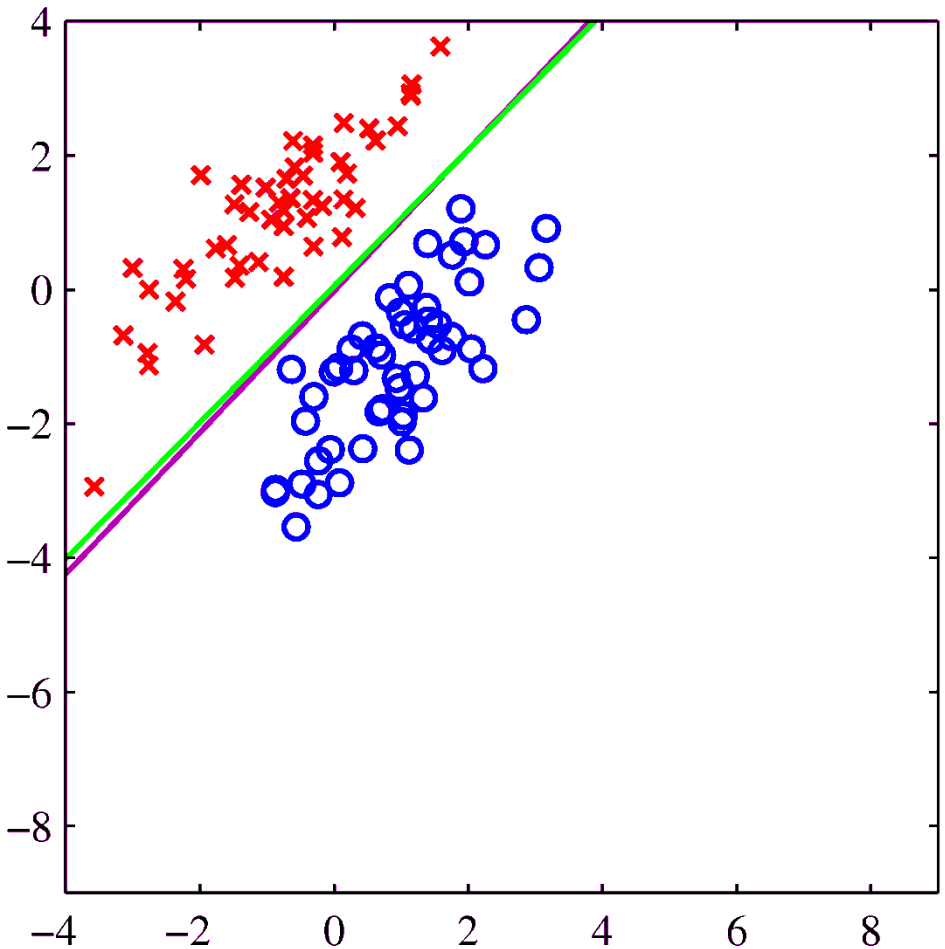
$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^{\mathrm{T}}\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^{\mathrm{T}}\mathbf{T} = \widetilde{\mathbf{X}}^{\dagger}\mathbf{T}$$

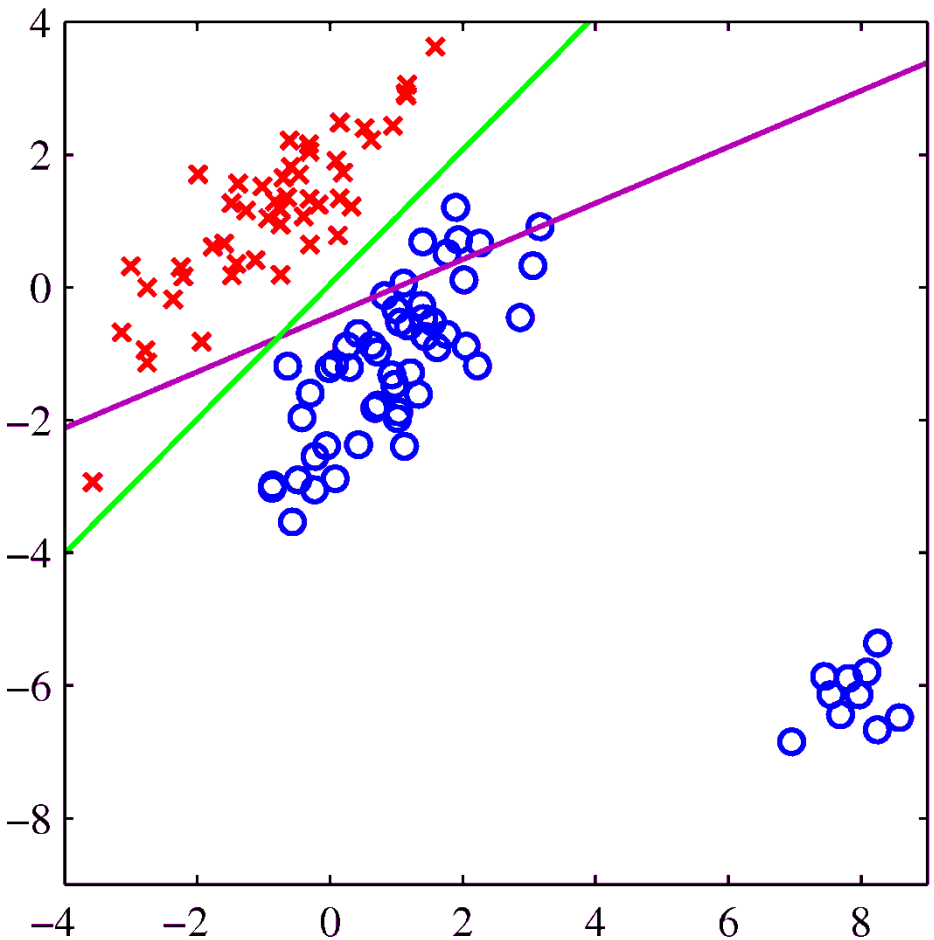
- **Linear discriminant function**

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}} = \mathbf{T}^{\mathrm{T}}\left(\widetilde{\mathbf{X}}^{\dagger}\right)^{\mathrm{T}}\widetilde{\mathbf{x}}$$

- **Properties**
  - $\mathbf{a}^{\mathrm{T}}\mathbf{t}_n + b = 0$ implies $\mathbf{a}^{\mathrm{T}}\mathbf{y}(\mathbf{x}) + b = 0$, thus output sums up to 1
  - … but each individual element may be outside [0,1]

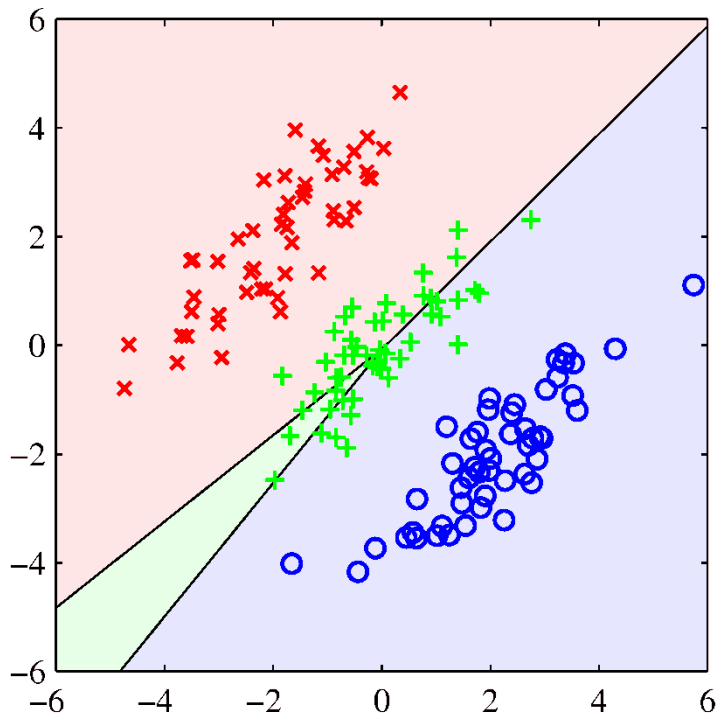
Least-Squares Discriminant Example





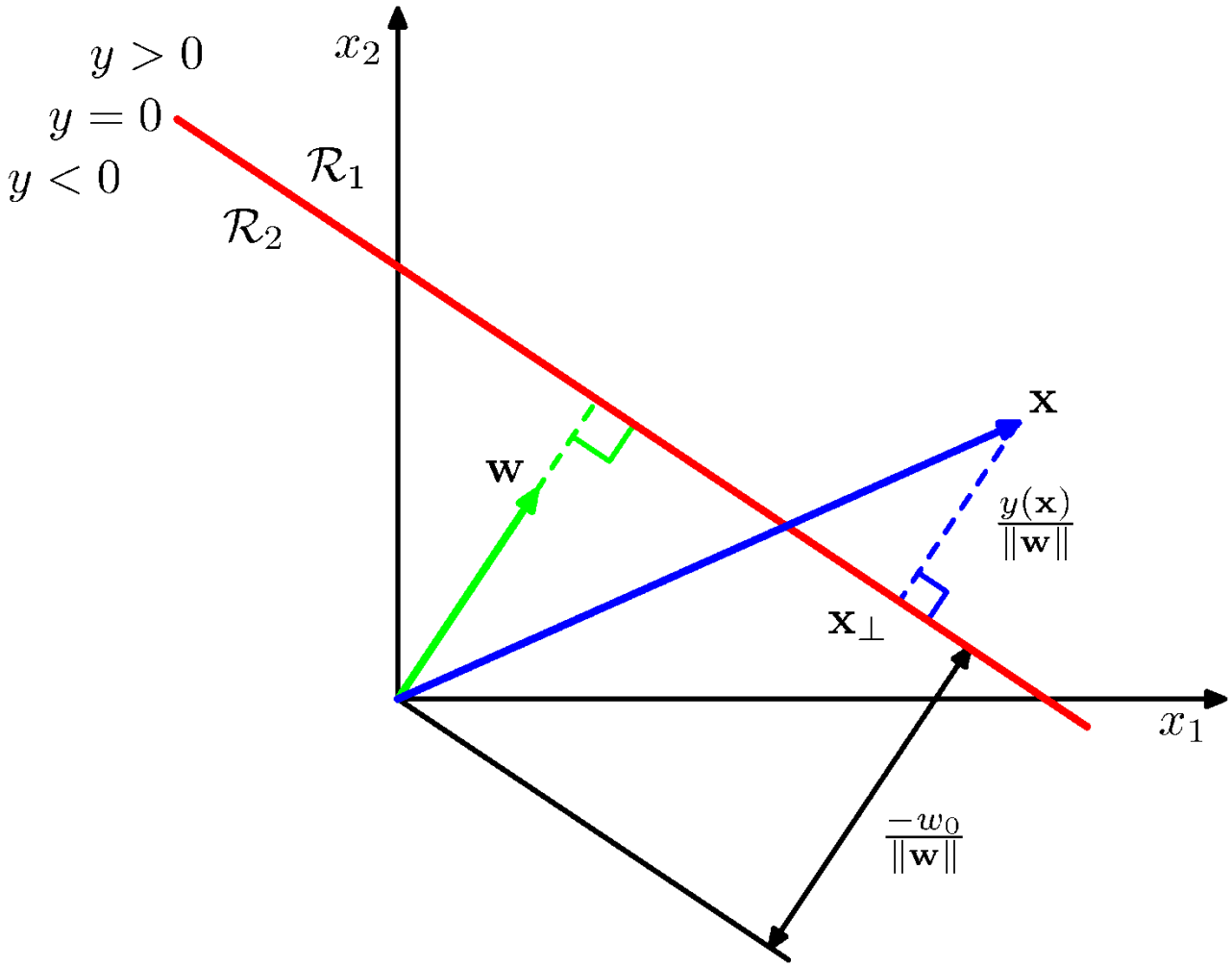
Least-Squares Discriminant Properties

- **Outliers**
  - suffers from lack of robustness to outliers
  - the sum-of-squares function penalizes “too-correct” predictions
- **Assumptions**
  - least-squares assumes Gaussian conditional distribution
  - binary target vectors are far from having a Gaussian distribution



Fisher’s Linear Discriminant

Recall Linear Discriminant Geometry



Classification as Projection

- **Consideration**
  - view classification in terms of dimensionality reduction
- **Observation**
  - linear discriminant amounts to …
  - … projection down to one dimension using $\mathbf{w}$
  - … and thresholding using w_0
- **Idea**
  - choose a projection that maximizes class separation!
  - adjust $\mathbf{w}$ to achieve the desired projection

Maximum Separation of Means

- **Class means**

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n, \qquad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$$

- **Projection**

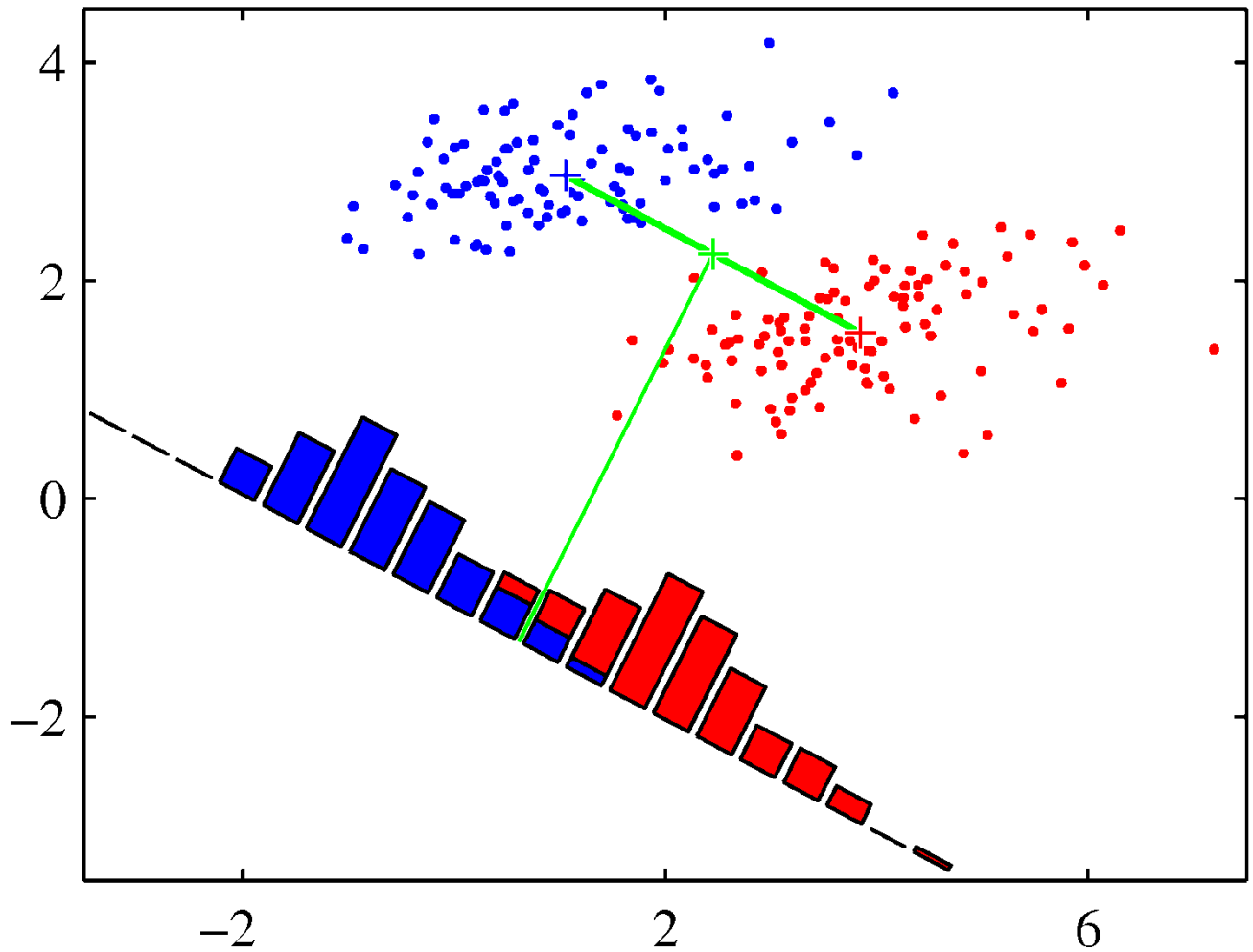
$$m_1 = \mathbf{w}^{\mathrm{T}} \mathbf{m}_1 \qquad m_2 = \mathbf{w}^{\mathrm{T}} \mathbf{m}_2$$

- **Separation**

$$m_2 - m_1 = \mathbf{w}^{\mathrm{T}} (\mathbf{m}_2 - \mathbf{m}_1)$$

- **Constraint**
  - constrain $\mathbf{w}$ to have unit length, using a Lagrange multiplier
- **Solution**
  - project on the line connecting the means $\mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1)$

Maximum Separation of Means Example



Fisher's Criterion

- **Idea**
  - find projection that maximizes class separation …
  - … and minimizes the within class variance (and class overlap)
- **Within-class variance**

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2 = \sum_{n \in \mathcal{C}_k} \left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n - \mathbf{w}^{\mathrm{T}}\mathbf{m}_k\right)^2 = \sum_{n \in \mathcal{C}_k} \left(\mathbf{w}^{\mathrm{T}}(\mathbf{x}_n - \mathbf{m}_k)\right)^2$$

- **Fisher's criterion**
  - maximize the ratio of between-class and within-class variance

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

Fisher's Linear Discriminant

- **Fisher's criterion rewrite**

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\left(\mathbf{w}^{\mathrm{T}}(\mathbf{m}_2 - \mathbf{m}_1)\right)^2}{\sum_{n \in \mathcal{C}_1} \left(\mathbf{w}^{\mathrm{T}}(\mathbf{x}_n - \mathbf{m}_1)\right)^2 + \sum_{n \in \mathcal{C}_2} \left(\mathbf{w}^{\mathrm{T}}(\mathbf{x}_n - \mathbf{m}_2)\right)^2} = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w}}$$

$$\mathbf{S}_{\mathrm{B}} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^{\mathrm{T}} \qquad \mathbf{S}_{\mathrm{W}} = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^{\mathrm{T}} + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^{\mathrm{T}}$$

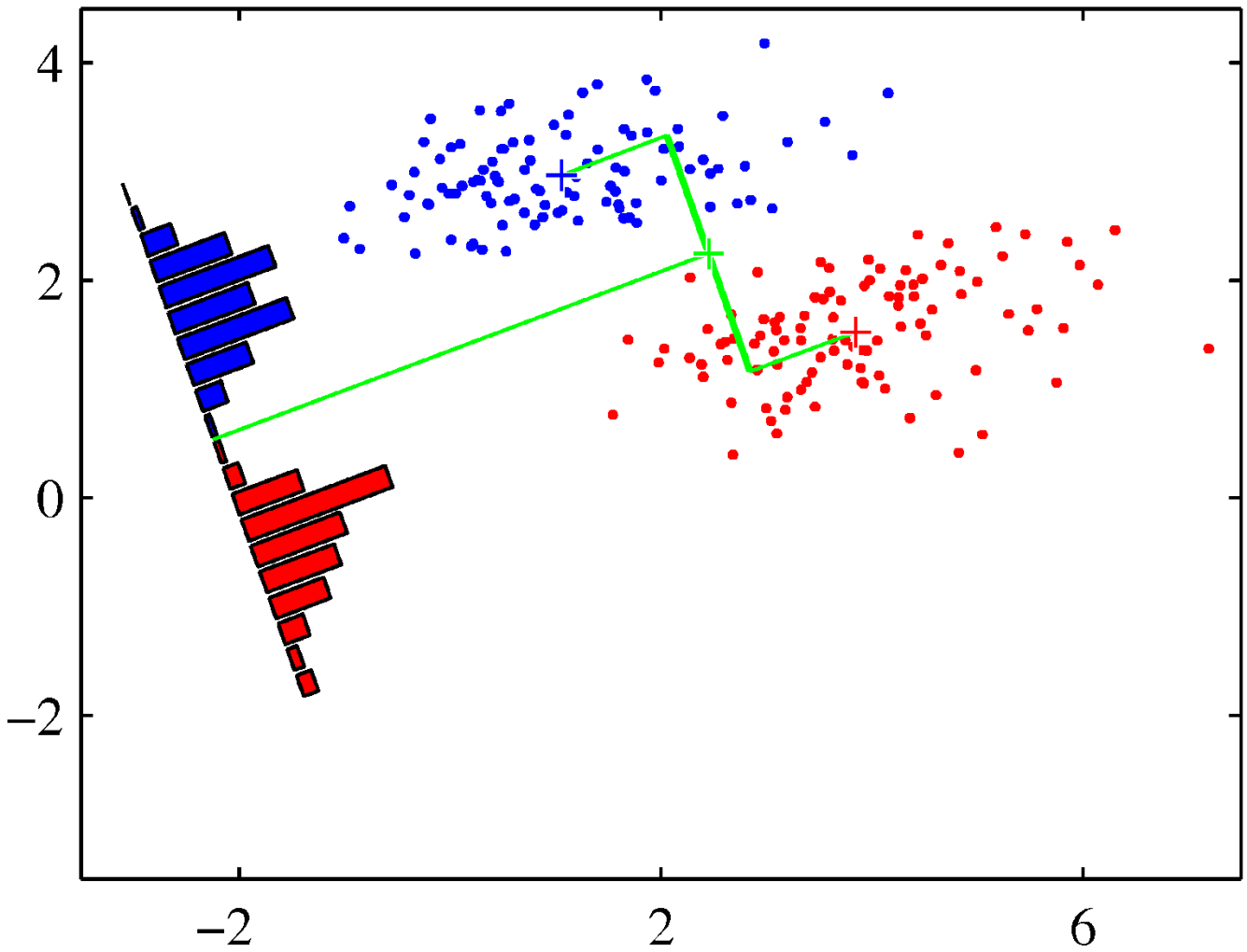
- **Maximization**
  - differentiate with respect to $\mathbf{w}$ and set to zero

$$(\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w})\mathbf{S}_{\mathrm{W}}\mathbf{w} = (\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w})\mathbf{S}_{\mathrm{B}}\mathbf{w}$$

  - drop the scaling factor (the magnitude of $\mathbf{w}$ is irrelevant)
  - multiply both sides by $\mathbf{S}_{\mathrm{W}}^{-1}$

$$\mathbf{w} \propto \mathbf{S}_{\mathrm{W}}^{-1}\mathbf{S}_{\mathrm{B}}\mathbf{w} = \mathbf{S}_{\mathrm{W}}^{-1}(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^{\mathrm{T}}\mathbf{w} \propto \mathbf{S}_{\mathrm{W}}^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

Fisher's Linear Discriminant Example



Fisher's Criterion and Least Squares

- **Target coding**
  - for class C_1: N/N_1, for class C_2: $-N/N_2$ (N data points)
  - approximates the reciprocal of the prior class probability
- **Minimization of the sum-of-squares error**

$$E = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + w_0 - t_n\right)^2$$

$$\sum_{n=1}^{N}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + w_0 - t_n\right) = 0$$

$$\sum_{n=1}^{N}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + w_0 - t_n\right)\mathbf{x}_n = 0$$

$$\sum_{n=1}^{N} t_n = N_1\frac{N}{N_1} - N_2\frac{N}{N_2} = 0$$

$$\left(\mathbf{S}_{\mathrm{W}} + \frac{N_1 N_2}{N}\mathbf{S}_{\mathrm{B}}\right)\mathbf{w} = N(\mathbf{m}_1 - \mathbf{m}_2)$$

$$\mathbf{m} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n = \frac{1}{N}(N_1\mathbf{m}_1 + N_2\mathbf{m}_2)$$

$$\mathbf{S}_{\mathrm{B}} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^{\mathrm{T}}$$

$$w_0 = -\mathbf{w}^{\mathrm{T}}\mathbf{m}$$

$$\mathbf{w} \propto \mathbf{S}_{\mathrm{W}}^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

$$\boxed{\text{class } \mathcal{C}_1 \text{ if } y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}(\mathbf{x} - \mathbf{m}) > 0 \text{ and class } \mathcal{C}_2 \text{ otherwise}}$$

Multi-Class Fisher's Discriminant

- *D* dimensions, *K* classes, *D*' linear features $y_k = \mathbf{w}_k^{\mathrm{T}}\mathbf{x}$, (*D*×*D*') $\mathbf{W}$
- **Projection**
  - dimensionality reduction from D to D' dimensions: $\mathbf{y} = \mathbf{W}^{\mathrm{T}}\mathbf{x}$
- **Co-variances in input space**

$$\mathbf{S}_{\mathrm{W}} = \sum_{k=1}^{K} \mathbf{S}_k$$

$$\mathbf{S}_{\mathrm{T}} = \sum_{n=1}^{N} (\mathbf{x}_n - \mathbf{m})(\mathbf{x}_n - \mathbf{m})^{\mathrm{T}}$$

$$\mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^{\mathrm{T}}$$

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$$

$$\mathbf{m} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n = \frac{1}{N} \sum_{k=1}^{K} N_k \mathbf{m}_k$$

$$\mathbf{S}_{\mathrm{B}} = \sum_{k=1}^{K} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^{\mathrm{T}}$$

$$\mathbf{S}_{\mathrm{T}} = \mathbf{S}_{\mathrm{W}} + \mathbf{S}_{\mathrm{B}}$$

Multi-Class Fisher's Discriminant

- **Co-variances in projected space**

$$\mathbf{s}_{\mathrm{W}} = \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} (\mathbf{y}_n - \boldsymbol{\mu}_k)(\mathbf{y}_n - \boldsymbol{\mu}_k)^{\mathrm{T}} \qquad \mathbf{s}_{\mathrm{B}} = \sum_{k=1}^{K} N_k (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^{\mathrm{T}}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{y}_n, \qquad \boldsymbol{\mu} = \frac{1}{N} \sum_{k=1}^{K} N_k \boldsymbol{\mu}_k$$

- **Multi-class Fisher's criterion**

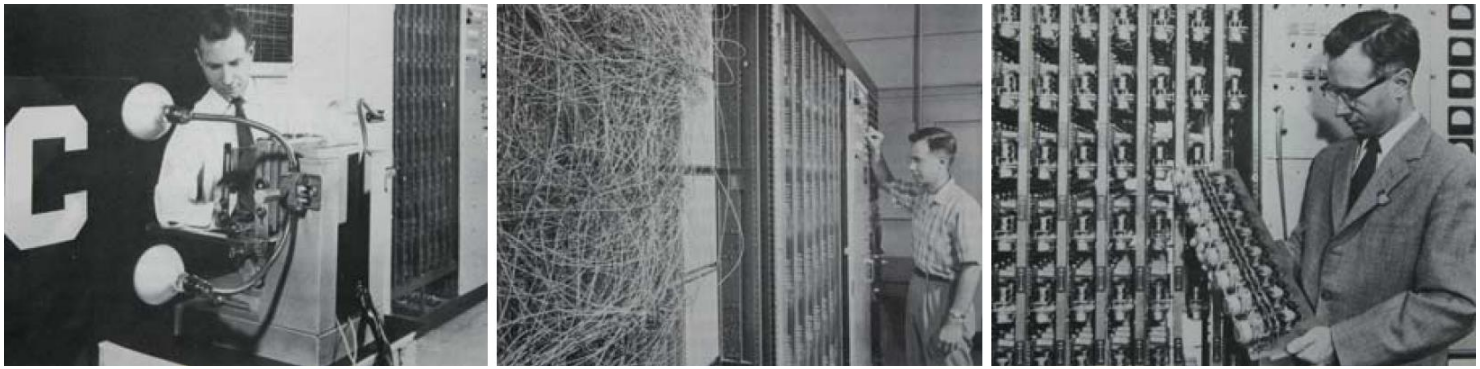
$$J(\mathbf{W}) = \mathrm{Tr}\{\mathbf{s}_{\mathrm{W}}^{-1} \mathbf{s}_{\mathrm{B}}\} = \mathrm{Tr}\left\{(\mathbf{W}\mathbf{S}_{\mathrm{W}}\mathbf{W}^{\mathrm{T}})^{-1}(\mathbf{W}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{W})\right\}$$

- **Solution**
  - weights determined by the *D*' most-important eigenvectors
- **Observation**
  - *D*' can effectively be at most (*K*-1), due to the rank of $\mathbf{S}_{\mathrm{B}}$

The Perceptron Algorithm

Perceptron

- **History**
  - a first type of neural network (Rosenblatt, 1962)



- **Generalized linear model**

$$y(\mathbf{x}) = f\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})\right) \qquad f(a) = \begin{cases} +1, & a \geqslant 0 \\ -1, & a < 0 \end{cases} \qquad \phi_0(\mathbf{x}) = 1$$

- **Target coding**
  - for class C_1: +1, for class C_2: -1

Perceptron Criterion

- **Perceptron classification**
  - $\mathbf{x}_n$ in class C_1, if $\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) > 0$; $\mathbf{x}_n$ in class C_2, if $\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) < 0$
- **Perceptron criterion**
  - for correct classification, training examples $(\mathbf{x}_n, t_n)$ must satisfy

$$\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)t_n \;>\; 0$$

  - if example is correctly classified, no penalty
  - if example is misclassified, a penalty of $-\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)t_n$
- **Perceptron error function**

$$E_{\mathrm{P}}(\mathbf{w}) = -\sum_{n\in\mathcal{M}} \mathbf{w}^{\mathrm{T}}\phi_n t_n$$

  - $\mathcal{M}$ is the set of misclassified examples

Perceptron Learning

- **Update rule**

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_{\mathrm{P}}(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \phi_n t_n$$

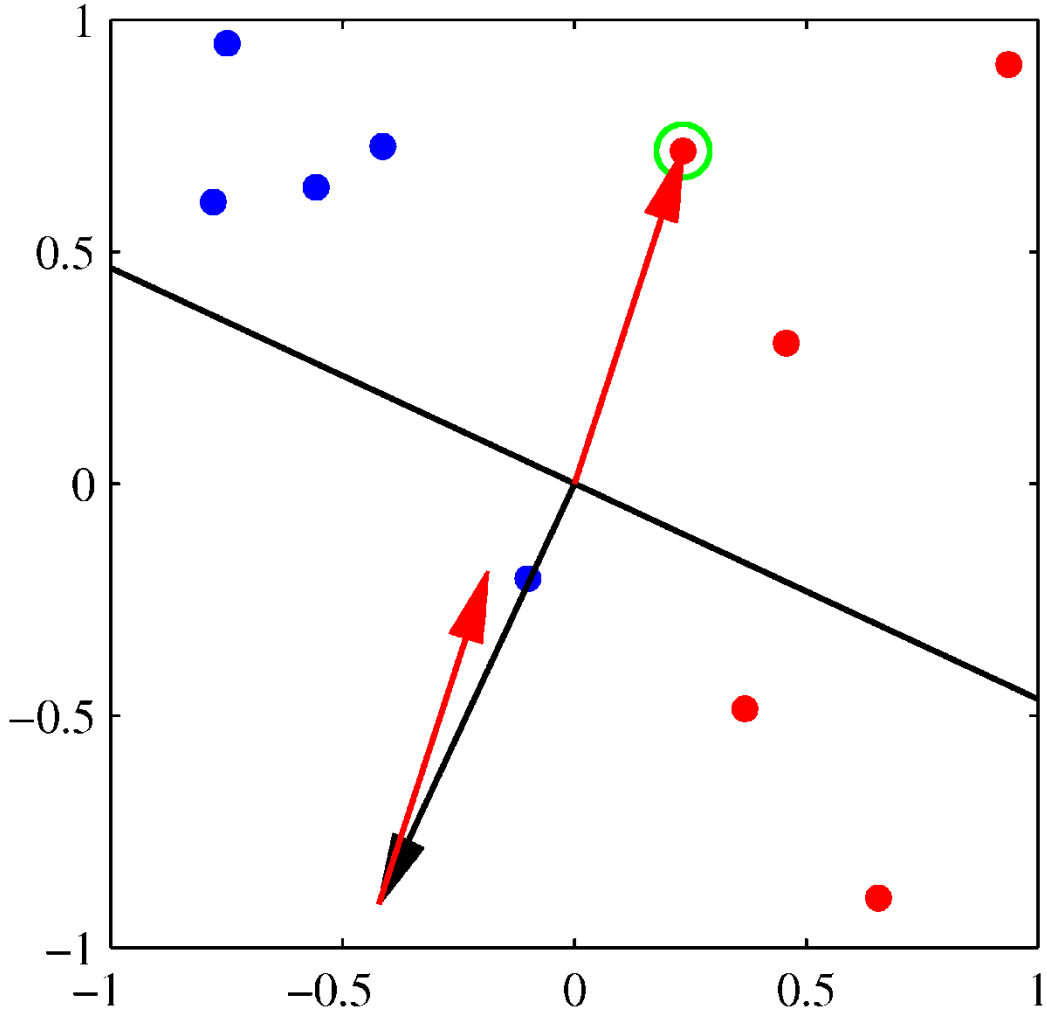
  - where η is the learning rate; it can be set to 1 (scaling of $\mathbf{w}$)
  - if example is correctly classified, no change
  - if misclassified, then we add or subtract $\phi(\mathbf{x}_n)$ to the weights

- **Convergence**
  - error reduction from one example only, after update

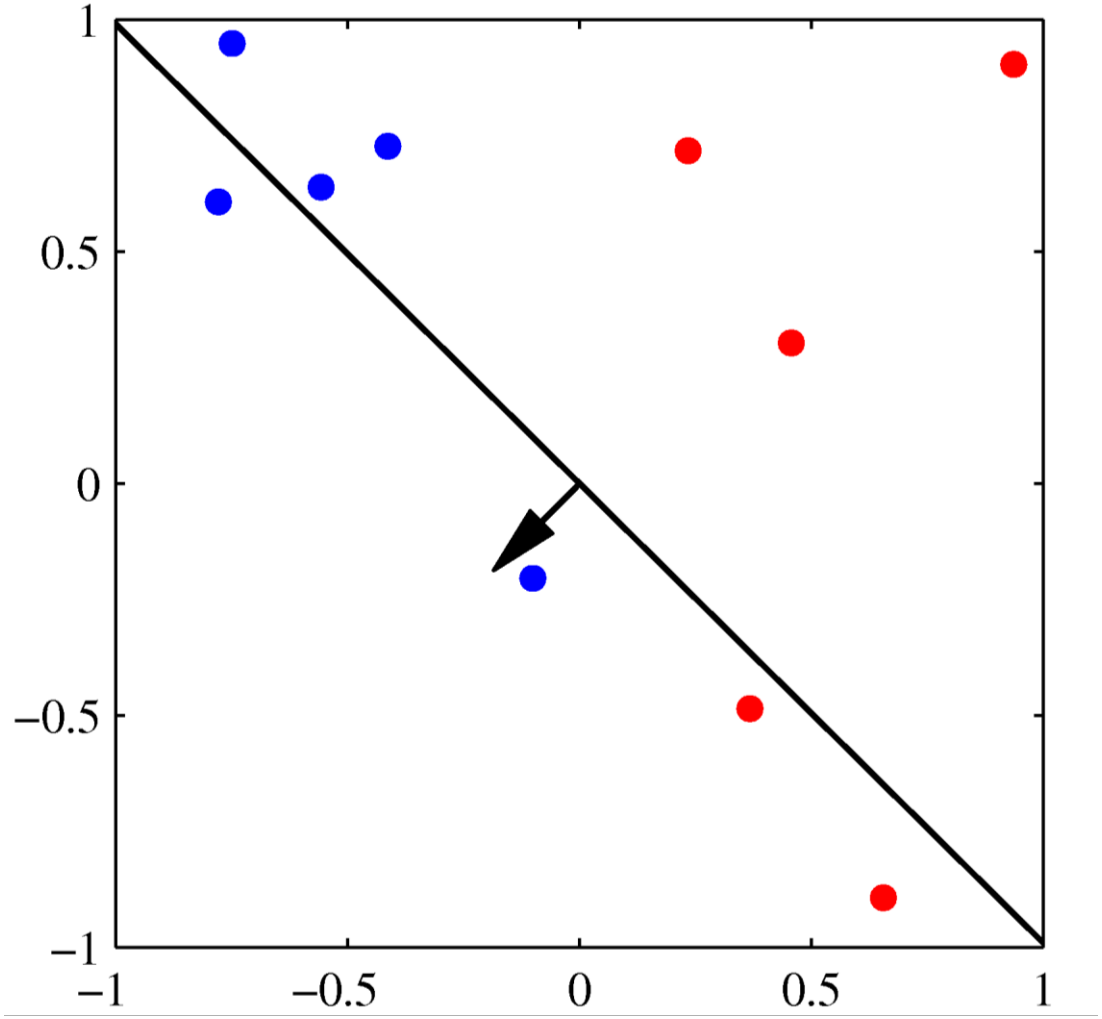
$$-\mathbf{w}^{(\tau+1)\mathrm{T}} \phi_n t_n = -\mathbf{w}^{(\tau)\mathrm{T}} \phi_n t_n - (\phi_n t_n)^{\mathrm{T}} \phi_n t_n < -\mathbf{w}^{(\tau)\mathrm{T}} \phi_n t_n$$

  - if data are linearly separable, the perceptron will converge
    - may take many steps; dependence on initialization and order
  - if data are not linearly separable, it will not converge

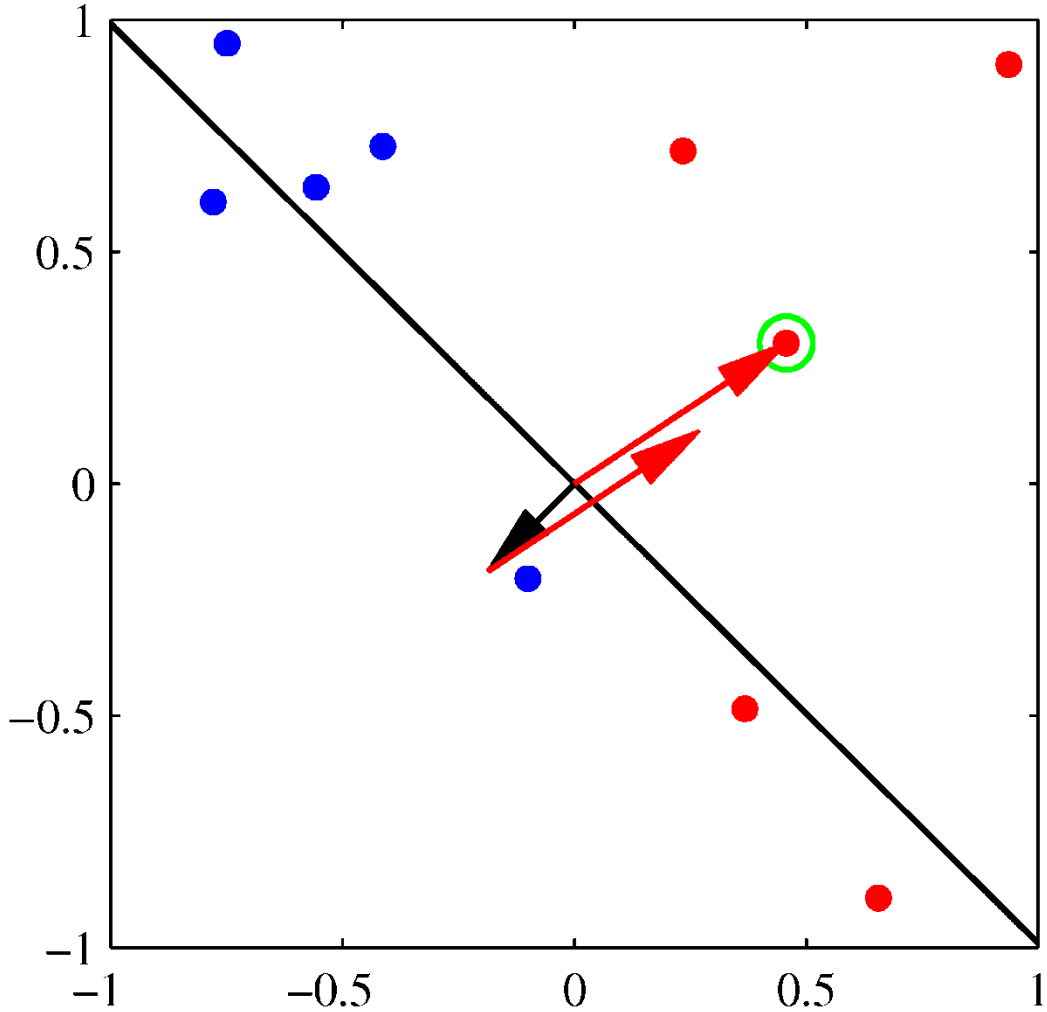
Perceptron Learning Example (1)



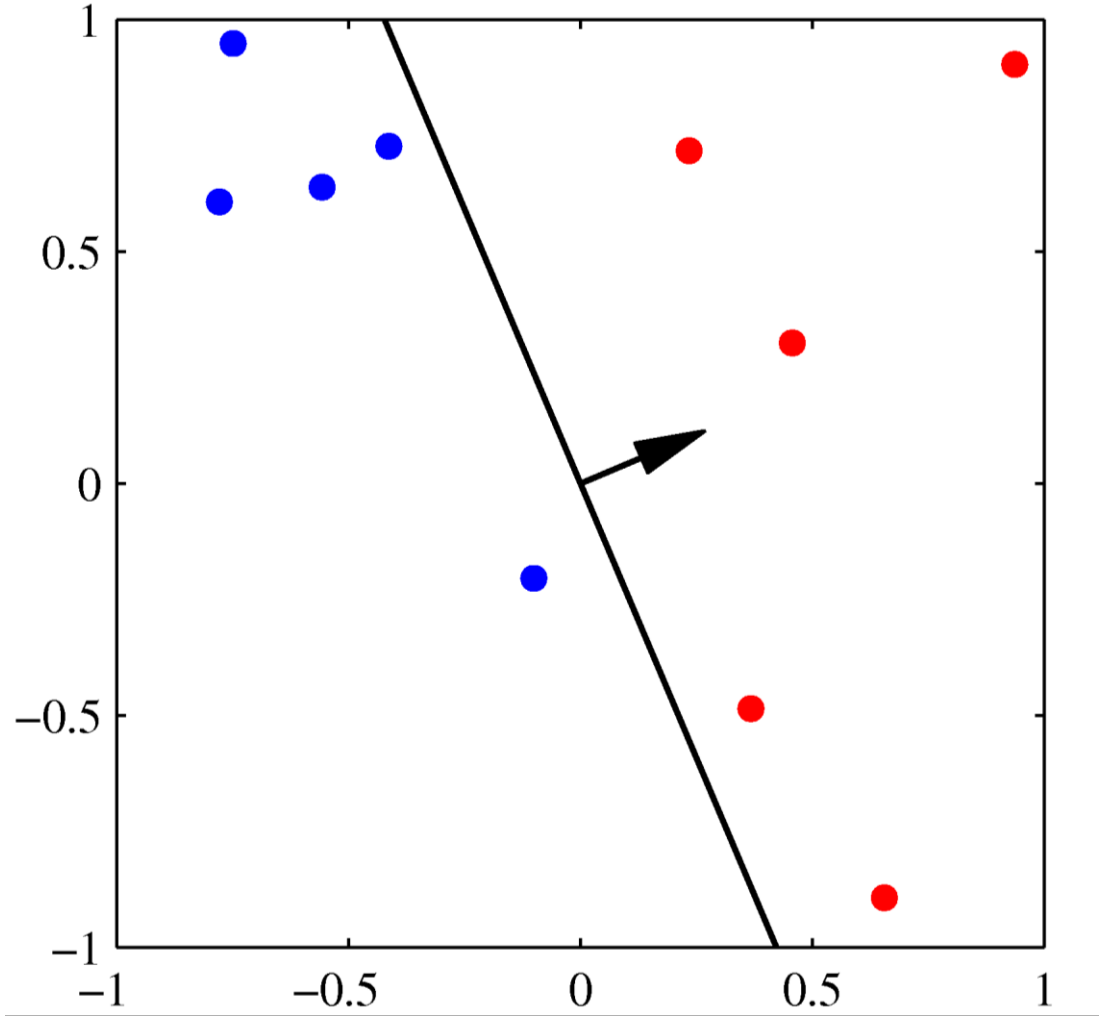
Perceptron Learning Example (2)



Perceptron Learning Example (3)



Perceptron Learning Example (4)



Perceptron Properties

- **Limitations**
  - binary classification problems
  - data must be linearly separable
- **Criticism**
  - perceptrons cannot learn the XOR function!
  - "*Perceptrons*" by Marvin Minsky and Seymour Papert (1969)
  - turned down all funding in neural computing
- **Related models**
  - *adaline*: adaptive linear element (Widrow and Hoff, 1960)
  - same functional form, different training method

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 09

Linear Generative Classification

TUC ECE, Spring 2023

Today

- **Linear Generative Classification**
  - generative approach
  - generative models
  - ML parameter estimation
  - discrete inputs
  - exponential family

Recall: Classification Decision Problem

- **Approach I: generative models**
  - determine the class-conditional densities $p(\mathbf{x}|C_k)$
  - infer the posterior class probabilities $p(C_k|\mathbf{x})$ (through Bayes)
  - use decision theory to make decision
- **Approach II: discriminative models**
  - determine the posterior class probabilities $p(C_k|\mathbf{x})$
  - use decision theory to make decision
- **Approach III: discriminant functions**
  - determine a function that maps inputs to classes directly
  - use the discriminant function to make decision

Linear Generative Classification

Generative Models for Classification

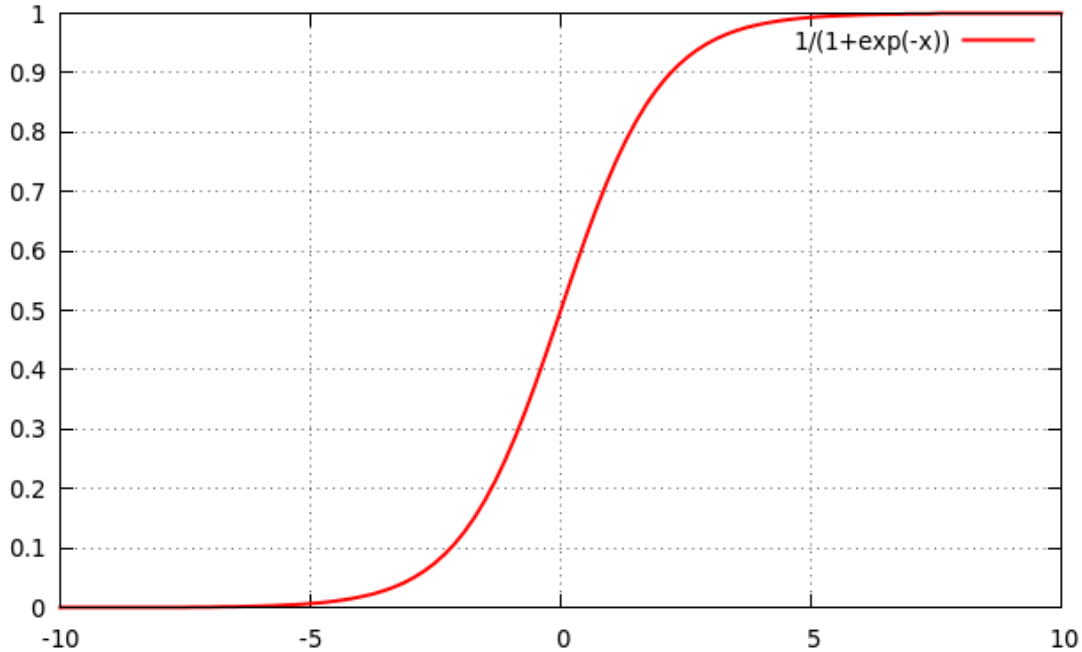
- **Generative models approach**
  - determine the class-conditional densities $p(\mathbf{x}|C_k)$
  - infer the posterior class probabilities $p(C_k|\mathbf{x})$ (through Bayes)
  - use decision theory to make decision
- **Simplest case: two classes**
  - posterior probability for the first class

$$
\begin{aligned}
p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\
&= \frac{1}{1+\exp(-a)} = \sigma(a) \qquad\qquad a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}
\end{aligned}
$$

Logistic Sigmoid Function

- **Definition**
  - squashing function
  - maps the real axis into a finite interval

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$



- **Symmetry**

$$\sigma(-a) = 1 - \sigma(a)$$

- **Inverse**
  - logit function (log odds)

$$a = \ln\left(\frac{\sigma}{1-\sigma}\right)$$

- **Derivative**

$$\sigma'(a) = \sigma(a)\big(1-\sigma(a)\big)$$

Generative Models for Classification

- **Generalization: multiple classes**
  - posterior probability for each class

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

$$a_k = \ln\left(p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)\right)$$

  - known as the normalized exponential
  - multiclass generalization of the sigmoid function
  - also known as the softmax function

$$a_k \gg a_j, \quad \forall j \neq k \implies p(\mathcal{C}_k|\mathbf{x}) \approx 1 \text{ and } p(\mathcal{C}_j|\mathbf{x}) \approx 0$$

Continuous Inputs, Two Classes

- **Gaussian class-conditional densities**
  - same covariance matrix $\boldsymbol{\Sigma}$

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}$$

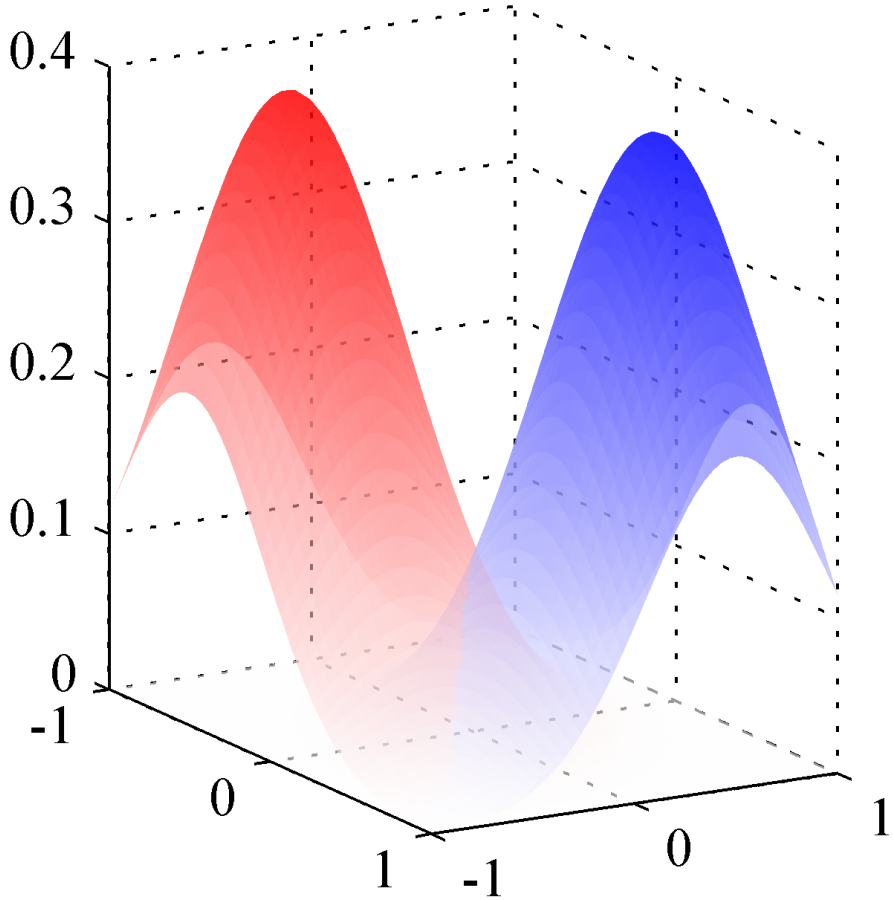
- **Posterior probability**

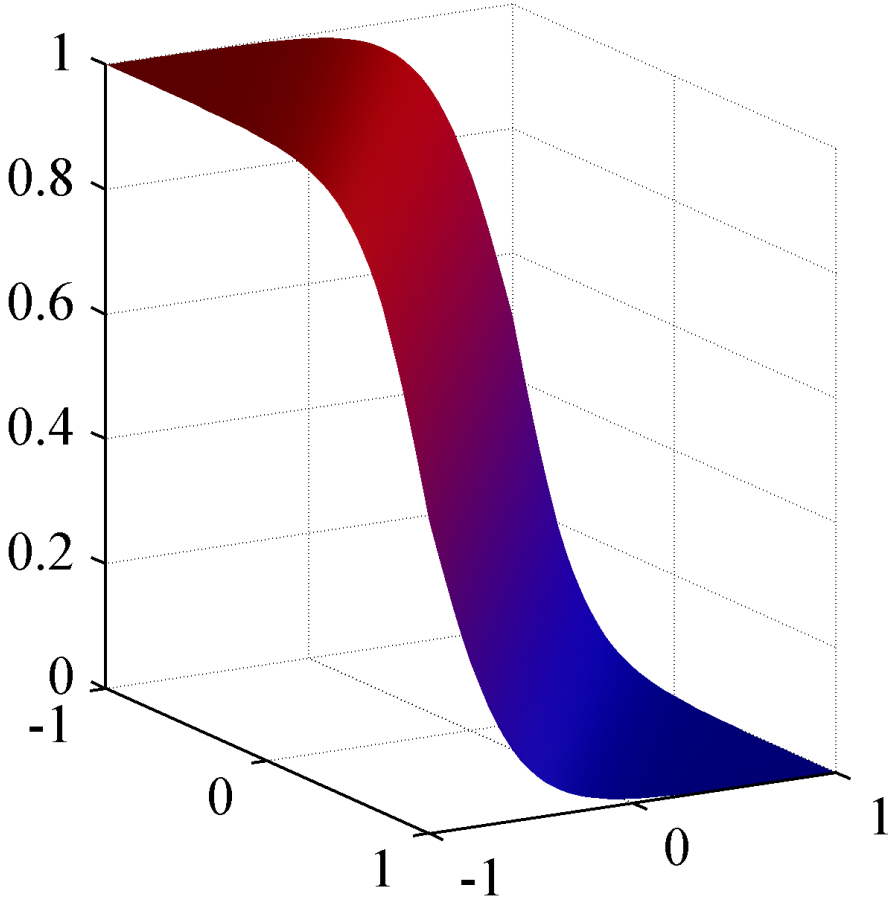
$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0)$$

$$\begin{aligned} \mathbf{w} &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)} \end{aligned}$$

  - quadratic terms cancel out (due to common covariance)
  - a sigmoid of a linear function of the input $\mathbf{x}$
  - linear boundary in input space, priors only shift the boundary

Continuous Inputs, Two Classes





Continuous Inputs, Multiple Classes

- **Gaussian class-conditional densities**
  - same covariance matrix Σ

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}$$

- **Posterior probability**

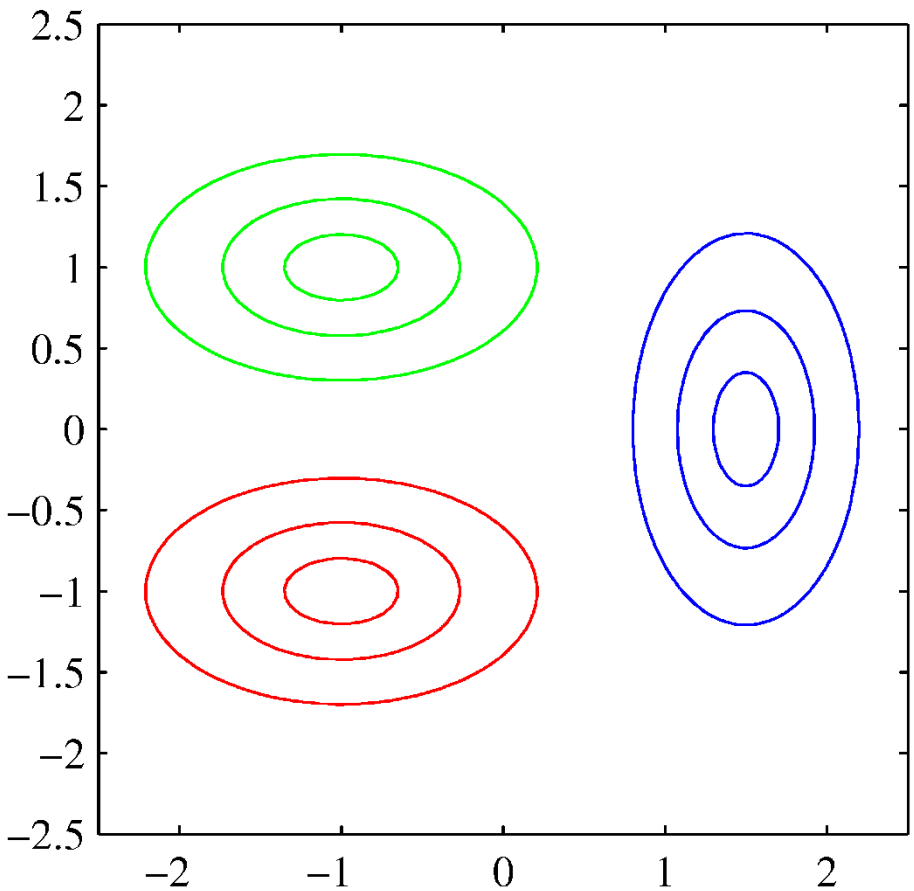
$$p(\mathcal{C}_k|\mathbf{x}) = \frac{\exp\left(a_k(\mathbf{x})\right)}{\sum_j \exp\left(a_j(\mathbf{x})\right)}$$

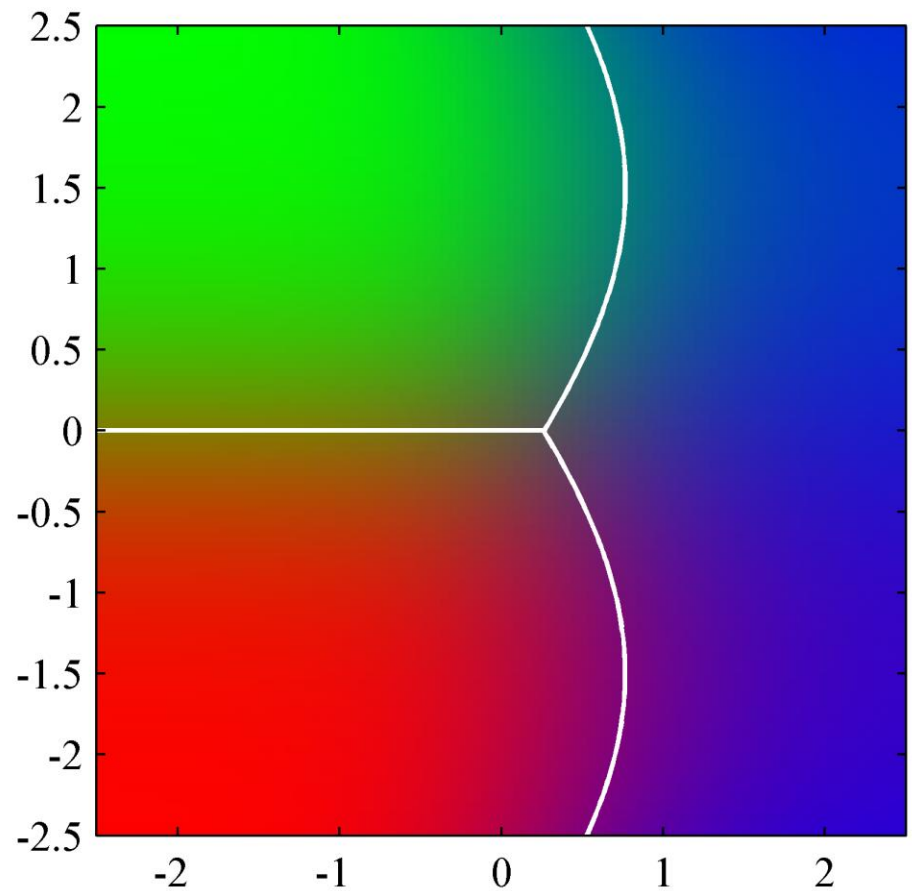
$$a_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$$

$$\begin{aligned} \mathbf{w}_k &= \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k \\ w_{k0} &= -\frac{1}{2}\boldsymbol{\mu}_k^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln p(\mathcal{C}_k) \end{aligned}$$

  - quadratic terms cancel out (due to common covariance)
  - for different covariance matrices Σ_k: a quadratic function of $\mathbf{x}$

Continuous Inputs, Multiple Classes





ML Parameter Estimation

- **Given**
  - data set $\{\mathbf{x}_n, t_n\}$, where $t_n = 1$ for C_1 and $t_n = 0$ for C_2
  - prior probabilities: $p(C_1) = \pi$ and $p(C_2) = 1 - \pi$
  - Gaussian class-conditional densities, same covariance matrix
- **Joint densities**

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$$

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1-\pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

- **Likelihood**

$$p(\mathbf{x}, \mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})\right]^{t_n} \left[(1-\pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})\right]^{1-t_n}$$

  - maximize log likelihood with respect to $\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}$

ML Estimation of π

- **Terms containing π**

$$\sum_{n=1}^{N} \{t_n \ln \pi + (1 - t_n) \ln(1 - \pi)\}$$

- **Differentiating and setting to zero**

$$\pi = \frac{1}{N} \sum_{n=1}^{N} t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$$

  - N_j: total number of data points in class C_j
  - ML estimate for prior: fraction of data in class
  - generalizes easily to multiple classes

ML Estimation of μ_1 and μ_2

- **Terms containing μ_1**

$$\sum_{n=1}^{N} t_n \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = -\frac{1}{2} \sum_{n=1}^{N} t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) + \text{const.}$$

- **Differentiating and setting to zero**

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^{N} t_n \mathbf{x}_n$$

$$\boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n=1}^{N} (1 - t_n) \mathbf{x}_n$$

  - ML estimate of $\boldsymbol{\mu}_j$: mean of data points assigned to class C_j
  - generalizes easily to multiple classes

ML Estimation of Σ

- **Terms containing Σ**

$$-\frac{1}{2}\sum_{n=1}^{N} t_n \ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N} t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1)$$

$$-\frac{1}{2}\sum_{n=1}^{N} (1-t_n) \ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N} (1-t_n)(\mathbf{x}_n - \boldsymbol{\mu}_2)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_2)$$

$$= -\frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{N}{2}\mathrm{Tr}\left\{\boldsymbol{\Sigma}^{-1}\mathbf{S}\right\}$$

$$\begin{aligned}
\mathbf{S} &= \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2 \\
\mathbf{S}_1 &= \frac{1}{N_1}\sum_{n\in\mathcal{C}_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}} \\
\mathbf{S}_2 &= \frac{1}{N_2}\sum_{n\in\mathcal{C}_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^{\mathrm{T}}
\end{aligned}$$

- **Solution**
  - $\Sigma = \mathrm{S}$
  - ML estimate of Σ: weighted average of class covariances
  - generalizes easily to multiple classes

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 10

Linear Discriminative Models

TUC ECE, Spring 2023

Today

- **Linear Discriminative Models**
  - two-class
  - iteratively reweighted least squares
  - multi-class

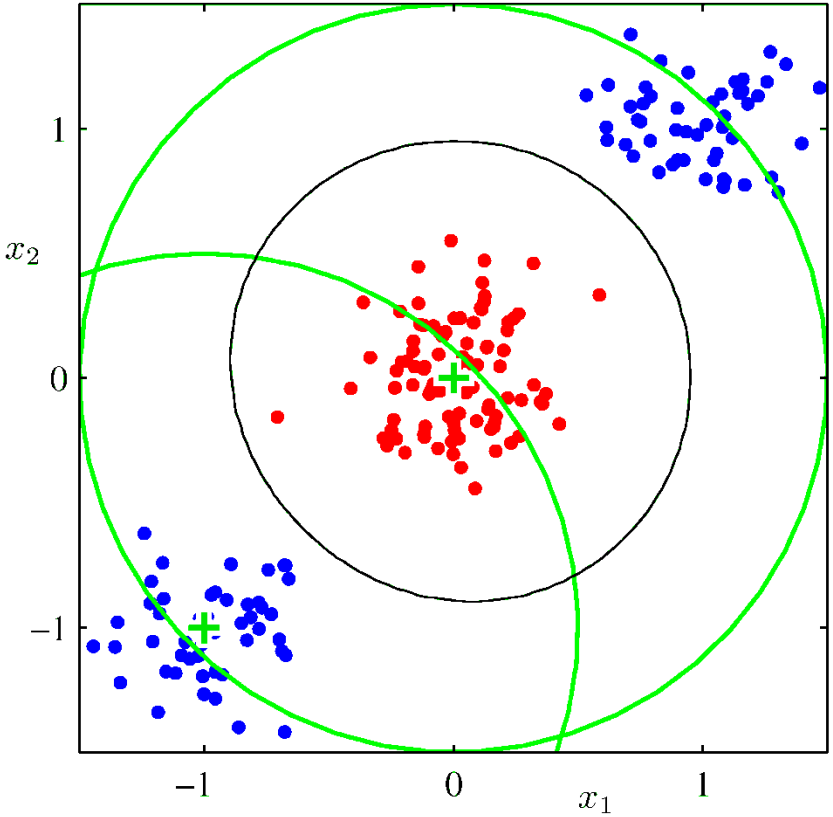
Linear Discriminative Models

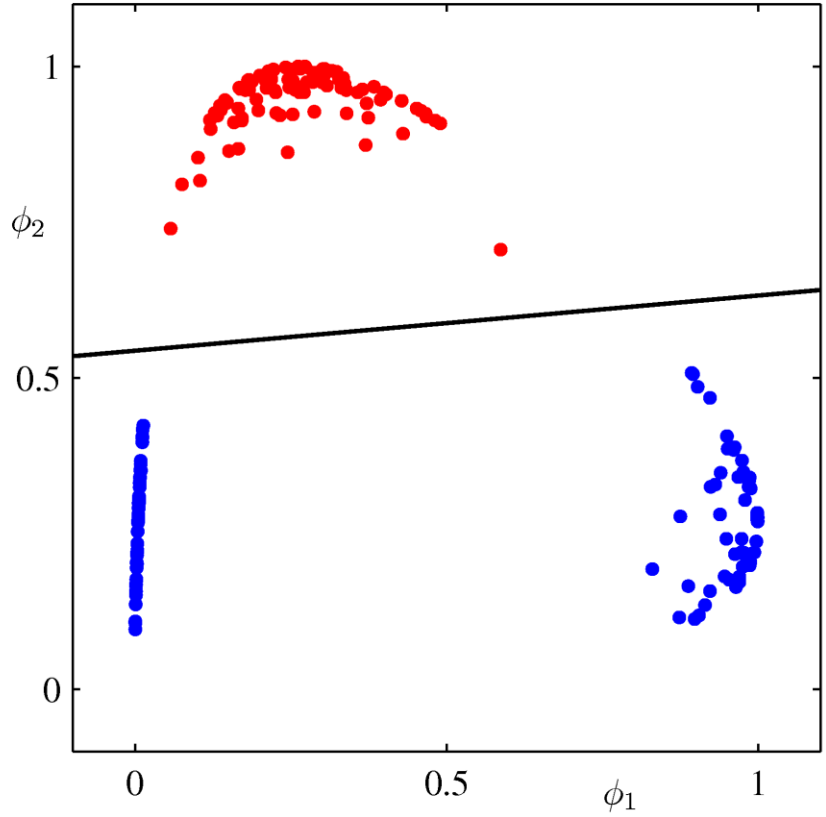
Recall: Classification Decision Problem

- **Approach I: generative models**
  - determine the class-conditional densities $p(\mathbf{x}|C_k)$
  - infer the posterior class probabilities $p(C_k|\mathbf{x})$ (through Bayes)
  - use decision theory to make decision
- **Approach II: discriminative models**
  - determine the posterior class probabilities $p(C_k|\mathbf{x})$
  - use decision theory to make decision
- **Approach III: discriminant functions**
  - determine a function that maps inputs to classes directly
  - use the discriminant function to make decision

Input vs. Feature Space

- classification models: in the original *or* in the feature space
- simplest feature space: a set of fixed basis functions





Binary Discriminative Models

Generative vs. Discriminative Models

- **Recall: two-class generative classification**
  - models for the class-conditional densities p($\mathbf{x}$|C_k) (and priors)
  - Gaussian class-conditional densities, same covariance matrix
  - assume M-dimensional feature space $\phi(\mathbf{x})$ (or input space $\mathbf{x}$)
  - $1+M+M+M(M+1)/2 = (M^2+5M+2)/2$ parameters: $\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}$
  - posterior: a sigmoid of a linear function of $\phi(\mathbf{x})$ (or $\mathbf{x}$)
- **Idea: two-class discriminative classification**
  - models for the posterior class probabilities p(C_k|$\mathbf{x}$)

$$p(\mathcal{C}_1|\boldsymbol{\phi}(\mathbf{x})) = \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})) \qquad p(\mathcal{C}_2|\boldsymbol{\phi}(\mathbf{x})) = 1 - p(\mathcal{C}_1|\boldsymbol{\phi}(\mathbf{x}))$$

  - a total of M parameters ($\mathbf{w}$) – linear vs. quadratic growth!

ML Parameter Estimation

- **Given**
  - a set of M fixed basis functions $\phi(\mathbf{x})$ (features)
  - posterior model: a sigmoid of a linear function of $\phi(\mathbf{x})$
  - data set $\{\phi_n, t_n\}$, where $\phi_n = \phi(\mathbf{x}_n)$, $t_n = 1$ for C_1 and $t_n = 0$ for C_2
- **Likelihood**

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} \{1 - y_n\}^{1-t_n} \qquad \mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}} \qquad y_n = \sigma(\mathbf{w}^{\mathrm{T}}\phi_n)$$

- **Cross-entropy error function**

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

  - optimize with respect to the parameters $\mathbf{w}$

ML Parameter Estimation

- **Cross-entropy error function**

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

$$y_n = \sigma(\mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}_n)$$

- **Differentiation**

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n) \boldsymbol{\phi}_n$$

- **Sequential gradient descent**

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla_{\mathbf{w}} E_n = \mathbf{w}^{(\tau)} - \eta (y_n - t_n) \boldsymbol{\phi}_n$$

- **Observations**
  - for linearly-separable data, there is an infinity of solutions
  - regularization or MAP estimation to avoid singularity

Newton-Raphson ML Estimation

- **ML parameter estimation**
  - unlike least-squares linear regression with Gaussian noise, closed-form solution is not possible due to non-linearity of σ
  - *idea*: use of Newton-Raphson iterative optimization scheme
  - based on a local quadratic approximation of the log likelihood
- **Newton-Raphson**
  - minimize error function $E(\mathbf{w})$
  - iterative update

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - \mathbf{H}^{-1}\nabla E(\mathbf{w})$$

  - $\mathbf{H}$ is the Hessian matrix (second derivatives of $E(\mathbf{w})$)

Newton-Raphson on Sum-of-Squares

- **Sum-of-squares error function**

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

- **Gradient and Hessian**

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}_n - t_n)\boldsymbol{\phi}_n = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{w} - \boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N}\boldsymbol{\phi}_n\boldsymbol{\phi}_n^{\mathrm{T}} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}$$

- **Newton-Raphson update**

$$\mathbf{w}^{(\mathrm{new})} = \mathbf{w}^{(\mathrm{old})} - (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\left\{\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{w}^{(\mathrm{old})} - \boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}\right\} = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}$$

  – least-squares solution in one step due to quadratic error

Newton-Raphson on Cross-Entropy

- **Cross-entropy error function**

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

- **Gradient and Hessian**

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\phi_n = \mathbf{\Phi}^{\mathrm{T}}(\mathbf{y}-\mathbf{t})$$

$$y_n = \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}_n)$$

$$R_{nn} = y_n(1-y_n)$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1-y_n)\phi_n\phi_n^{\mathrm{T}} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{\Phi}$$

- **Newton-Raphson update**

$$\begin{aligned}
\mathbf{w}^{(\mathrm{new})} &= \mathbf{w}^{(\mathrm{old})} - (\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}(\mathbf{y}-\mathbf{t}) \\
&= (\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{\Phi})^{-1}\left\{\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{\Phi}\mathbf{w}^{(\mathrm{old})} - \mathbf{\Phi}^{\mathrm{T}}(\mathbf{y}-\mathbf{t})\right\} \\
&= (\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{z}
\end{aligned}$$

$$\mathbf{z} = \mathbf{\Phi}\mathbf{w}^{(\mathrm{old})} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})$$

Iteratively Reweighted Least Squares

- **Iteratively Reweighted Least Squares (IRLS)** **[Rubin, 1983]**

$$\mathbf{w}^{(\text{new})} = (\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{z} \qquad \mathbf{z} = \mathbf{\Phi}\mathbf{w}^{(\text{old})} - \mathbf{R}^{-1}(\mathbf{y} - \mathbf{t}) \qquad R_{nn} = y_n(1 - y_n)$$

  - normal equations for a weighted least-squares problem
  - weights $\mathbf{R}$ and "targets" $\mathbf{z}$ depend on the parameter vector $\mathbf{w}$
  - the weights can be seen as the variance in the targets

$$\begin{aligned} \mathbb{E}[t] &= \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) = y \\ \mathrm{var}[t] &= \mathbb{E}[t^2] - \mathbb{E}[t]^2 = \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) - \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi})^2 = y(1 - y) \end{aligned}$$

- **IRLS interpretation**
  - solving the linearized problem in the space of $a = \mathbf{w}^{\mathrm{T}}\phi$
  - the effective target z_n is a localized linear approximation of σ

$$a_n(\mathbf{w}) \simeq a_n(\mathbf{w}^{(\text{old})}) + \left.\frac{\mathrm{d}a_n}{\mathrm{d}y_n}\right|_{\mathbf{w}^{(\text{old})}} (t_n - y_n) = \boldsymbol{\phi}_n^{\mathrm{T}}\mathbf{w}^{(\text{old})} - \frac{(y_n - t_n)}{y_n(1 - y_n)} = z_n$$

Multi-Class Discriminative Models

Multi-Class ML Parameter Estimation

- **Given**
  - a set of *M* fixed basis functions $\phi(\mathbf{x})$ (features)
  - posterior model per class: a softmax of a linear function of $\phi(\mathbf{x})$

$$p(\mathcal{C}_k|\boldsymbol{\phi}) = y_k(\boldsymbol{\phi}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \qquad a_k = \mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}$$

  - a total of $M \times K$ parameters for *K* classes
  - data set $\{\boldsymbol{\phi}_n, \mathbf{t}_n\}$, where $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$, $\mathbf{t}_n$ is a 1-of-*K* coding label
- **Likelihood**

$$p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(\mathcal{C}_k|\boldsymbol{\phi}_n)^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}} \qquad \mathbf{T} = \{t_{nk}\} \qquad y_{nk} = y_k(\boldsymbol{\phi}_n)$$

Multi-Class ML Parameter Estimation

- **Cross-entropy error function**

$$E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

- **Differentiation**

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \sum_{n=1}^{N} (y_{nj} - t_{nj})\, \boldsymbol{\phi}_n$$

$$\frac{\partial y_k}{\partial a_j} = y_k(I_{kj} - y_j) \qquad \sum_k t_{nk} = 1$$

- **Sequential gradient descent**

$$\mathbf{w}_k^{(\tau+1)} = \mathbf{w}_k^{(\tau)} - \eta \nabla_{\mathbf{w}_k} E_n = \mathbf{w}_k^{(\tau)} - \eta(y_{nk} - t_{nk})\boldsymbol{\phi}_n$$

Multi-Class Newton-Raphson

- **Cross-entropy error function**

$$E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

- **Gradient and Hessian**

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \sum_{n=1}^{N} (y_{nj} - t_{nj})\, \boldsymbol{\phi}_n$$

$$\nabla_{\mathbf{w}_k} \nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \sum_{n=1}^{N} y_{nk}(I_{kj} - y_{nj})\boldsymbol{\phi}_n \boldsymbol{\phi}_n^{\mathrm{T}}$$

- **IRLS**

$$\mathbf{w}_k^{(\mathrm{new})} = \mathbf{w}_k^{(\mathrm{old})} - \left(\nabla_{\mathbf{w}_k} \nabla_{\mathbf{w}_j} E\right)^{-1} \nabla_{\mathbf{w}_k} E$$

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 11

Bayesian Logistic Regression

TUC ECE, Spring 2023

Today

- **The Laplace approximation**
  - one-dimensional
  - multi-dimensional
  - application to model comparison
- **Bayesian logistic regression**
  - approximate posterior
  - approximate predictive distribution

The Laplace Approximation

1-D Laplace Approximation

- **Laplace density approximation**
  - approximate an arbitrary probability density with a Gaussian
  - Gaussian centered on a mode of the approximated density
- **Density**
  - probability density over some variable z

$$p(z) = \frac{1}{Z} f(z) \qquad\qquad Z = \int f(z)\,\mathrm{d}z$$

  - normalization constant Z can be unknown
- **Mode**
  - the derivative of the density is zero at a mode z_0 : $\left.\frac{df(z)}{dz}\right|_{z=z_0} = 0$

1-D Laplace Approximation

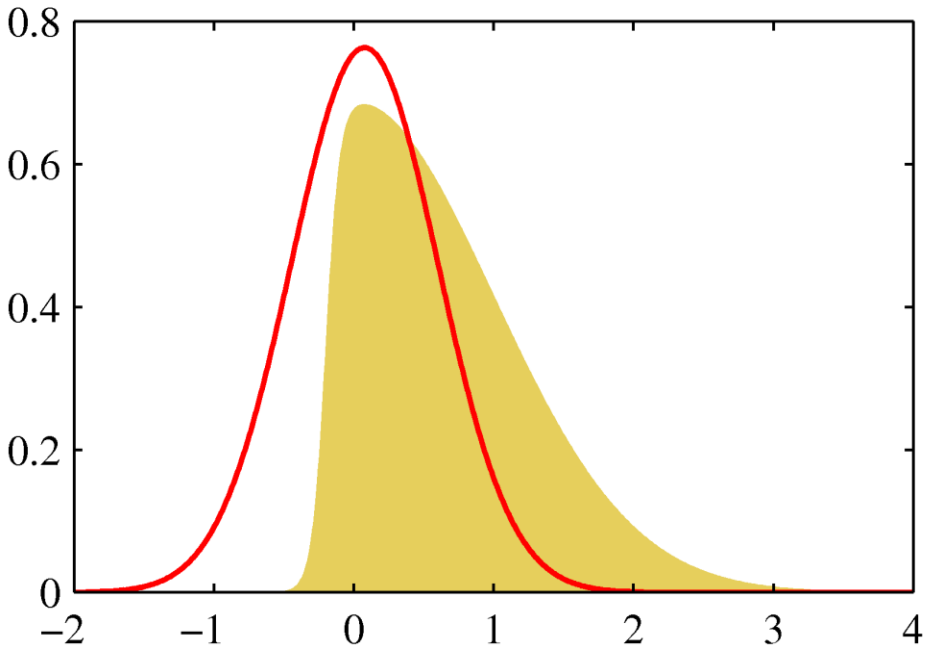
- **Taylor expansion** $g(x) = g(a) + g'(a)(x-a) + \frac{g''(a)}{2!}(x-a)^2 + \ldots + \frac{g^{(n)}(a)}{n!}(x-a)^n + \ldots$
  - Taylor expansion of the density logarithm about the mode z_o

$$\ln f(z) \simeq \ln f(z_0) - \frac{1}{2}A(z-z_0)^2 \qquad A = -\left.\frac{d^2}{dz^2}\ln f(z)\right|_{z=z_0}$$

  - the first-order term vanishes due to z_o being a mode
- **Gaussian approximation**
  - the logarithm of a Gaussian is a quadratic function
  - so is the Taylor expansion of the density logarithm!
  - take exponential to form a Gaussian and normalize (need *A*>0)

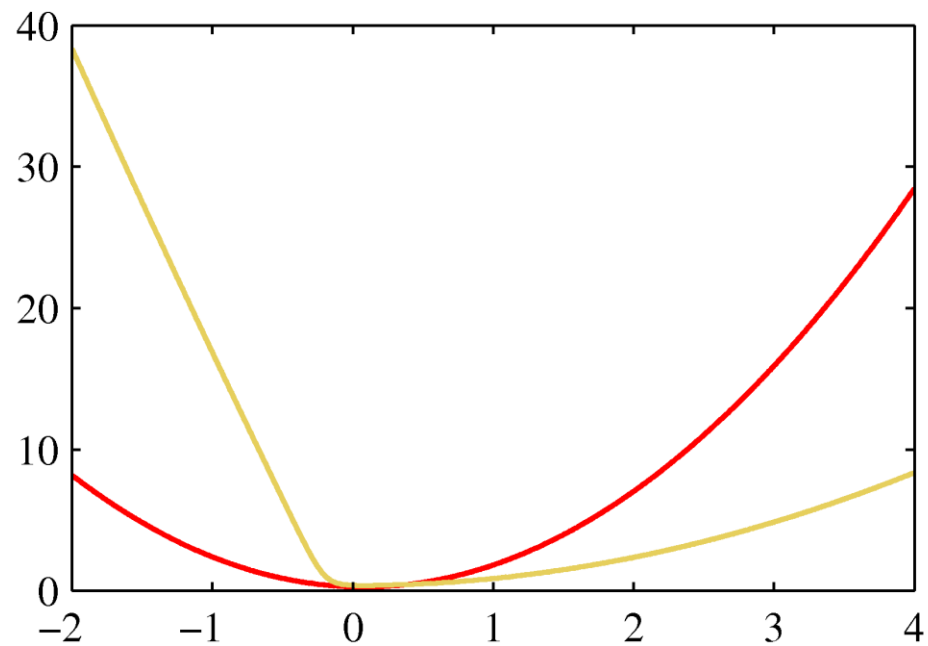
$$f(z) \simeq f(z_0)\exp\left\{-\frac{A}{2}(z-z_0)^2\right\} \qquad q(z) = \left(\frac{A}{2\pi}\right)^{1/2}\exp\left\{-\frac{A}{2}(z-z_0)^2\right\}$$

1-D Laplace Approximation Example



normalized distributions

$$p(z) \propto \exp(-z^2/2)\sigma(20z+4)$$



negative logarithm curves

M-D Laplace Approximation

- **Density**
  - density over *M*-dimensional variable $\mathbf{z}$: $p(\mathbf{z}) = f(\mathbf{z})/Z$
- **Taylor expansion**

$$\ln f(\mathbf{z}) \simeq \ln f(\mathbf{z}_0) - \frac{1}{2}(\mathbf{z}-\mathbf{z}_0)^{\mathrm{T}}\mathbf{A}(\mathbf{z}-\mathbf{z}_0) \qquad \mathbf{A} = -\nabla\nabla \ln f(\mathbf{z})|_{\mathbf{z}=\mathbf{z}_0}$$

- **Gaussian approximation**

$$f(\mathbf{z}) \simeq f(\mathbf{z}_0)\exp\left\{-\frac{1}{2}(\mathbf{z}-\mathbf{z}_0)^{\mathrm{T}}\mathbf{A}(\mathbf{z}-\mathbf{z}_0)\right\}$$

$$q(\mathbf{z}) = \frac{|\mathbf{A}|^{1/2}}{(2\pi)^{M/2}}\exp\left\{-\frac{1}{2}(\mathbf{z}-\mathbf{z}_0)^{\mathrm{T}}\mathbf{A}(\mathbf{z}-\mathbf{z}_0)\right\} = \mathcal{N}(\mathbf{z}|\mathbf{z}_0, \mathbf{A}^{-1})$$

  - $\mathbf{A}$ must be positive definite for $\mathbf{z}_0$ to be a mode

Laplace Approximation in Practice

- **Procedure**
  - find a mode: numerical optimization (multiple solutions)
  - evaluate the Hessian at the mode: must be positive definite
- **Application**
  - approximating posteriors (tend to be Gaussian for lots of data)
- **Weaknesses**
  - the Gaussian distribution extends over the entire real axis
  - the Gaussian distribution is inherently unimodal
  - the approximation is based on local aspects around the mode
  - the approximation may fail to capture global properties
- **Trick**
  - transformation: Laplace approximation of $\ln x$ for $0 \leq x < \infty$

Approximating the Normalizer

- **Idea**
  - integrate the Laplace approximation instead of the density
- **Approximate normalizer**

$$
\begin{aligned}
Z &= \int f(\mathbf{z})\,\mathrm{d}\mathbf{z} \\
&\simeq f(\mathbf{z}_0) \int \exp\left\{-\frac{1}{2}(\mathbf{z}-\mathbf{z}_0)^{\mathrm{T}}\mathbf{A}(\mathbf{z}-\mathbf{z}_0)\right\}\mathrm{d}\mathbf{z} \\
&= f(\mathbf{z}_0)\frac{(2\pi)^{M/2}}{|\mathbf{A}|^{1/2}}
\end{aligned}
$$

- **Application**
  - compute approximation to model evidence

Application to Model Comparison

- **Bayesian model comparison**
  - given: data set D, set of models $\{M_i\}$ with parameters $\{\boldsymbol{\theta}_i\}$
  - define likelihood $p(D|\boldsymbol{\theta}_i, M_i)$, prior $p(\boldsymbol{\theta}_i|M_i)$ over parameters
  - find the model evidence $p(D|M_i)$ for each model M_i
- **Approach**

$$p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} \qquad \begin{array}{c} f(\boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \\ Z = p(\mathcal{D}) \end{array} \qquad Z = f(\mathbf{z}_0)\frac{(2\pi)^{M/2}}{|\mathbf{A}|^{1/2}}$$

$$\ln p(\mathcal{D}) \simeq \ln p(\mathcal{D}|\boldsymbol{\theta}_{\mathrm{MAP}}) + \underbrace{\ln p(\boldsymbol{\theta}_{\mathrm{MAP}}) + \frac{M}{2}\ln(2\pi) - \frac{1}{2}\ln|\mathbf{A}|}_{\text{Occam factor}}$$

  - $\boldsymbol{\theta}_{\mathrm{MAP}}$ is the mode of the posterior
  - $\mathbf{A}$ is the Hessian $\mathbf{A} = -\nabla\nabla \ln p(\mathcal{D}|\boldsymbol{\theta}_{\mathrm{MAP}})p(\boldsymbol{\theta}_{\mathrm{MAP}}) = -\nabla\nabla \ln p(\boldsymbol{\theta}_{\mathrm{MAP}}|\mathcal{D})$

Bayesian Information Criterion (BIC)

- **Model evidence**

$$\ln p(\mathcal{D}) \simeq \ln p(\mathcal{D}|\boldsymbol{\theta}_{\mathrm{MAP}}) + \underbrace{\ln p(\boldsymbol{\theta}_{\mathrm{MAP}}) + \frac{M}{2}\ln(2\pi) - \frac{1}{2}\ln|\mathbf{A}|}_{\text{Occam factor}}$$

- **Assumptions**
  - a broad Gaussian prior distribution over the parameters
  - the Hessian is full rank
- **Bayesian Information or Schwarz Criterion**

$$\ln p(\mathcal{D}) \simeq \ln p(\mathcal{D}|\boldsymbol{\theta}_{\mathrm{MAP}}) - \frac{1}{2}M\ln N$$

  - 1st term: log likelihood evaluated at the optimized parameters
  - 2nd term: penalty for model complexity

Bayesian Logistic Regression

Bayesian Logistic Regression

- **Bayesian treatment**
  - introduce a (conjugate) prior over the parameters
  - define the likelihood of the data given the parameters
  - compute the posterior over the parameters
  - infer the predictive distribution by integrating over parameters
- **Exact Bayesian logistic regression**
  - finding the posterior or the predictive distribution is intractable!
  - posterior: product of prior and likelihood (product of sigmoids)
  - predictive: integration of a product of sigmoids
  - idea: apply Laplace approximation!

Approximate Bayesian Logistic Regression

- **Gaussian prior**

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$$

- **Posterior**

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{w})p(\mathbf{t}|\mathbf{w})$$

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} \{1 - y_n\}^{1-t_n}$$

$$\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$$

- **Log posterior**

$$\begin{aligned} \ln p(\mathbf{w}|\mathbf{t}) &= -\frac{1}{2}(\mathbf{w} - \mathbf{m}_0)^{\mathrm{T}}\mathbf{S}_0^{-1}(\mathbf{w} - \mathbf{m}_0) \\ &\quad + \sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\} + \text{const} \end{aligned}$$

$$y_n = \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}_n)$$

- **Laplace approximation**

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\mathrm{MAP}}, \mathbf{S}_N)$$

$$\mathbf{S}_N^{-1} = -\nabla\nabla \ln p(\mathbf{w}|\mathbf{t}) = \mathbf{S}_0^{-1} + \sum_{n=1}^{N} y_n(1 - y_n)\boldsymbol{\phi}_n\boldsymbol{\phi}_n^{\mathrm{T}}$$

Predictive Distribution

- **Predictive distribution for input $\phi(\mathrm{x})$**
  - class C_1: $p(\mathcal{C}_1|\boldsymbol{\phi}, \mathbf{t}) = \int p(\mathcal{C}_1|\boldsymbol{\phi}, \mathbf{w}) p(\mathbf{w}|\mathbf{t})\, \mathrm{d}\mathbf{w} \simeq \int \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) q(\mathbf{w})\, \mathrm{d}\mathbf{w}$
  - class C_2: $p(\mathcal{C}_2|\boldsymbol{\phi}, \mathbf{t}) = 1 - p(\mathcal{C}_1|\boldsymbol{\phi}, \mathbf{t})$
- **Algebraic manipulation**

$$a = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}$$

$$\sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) = \int \delta(a - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) \sigma(a)\, \mathrm{d}a$$

$$\int \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) q(\mathbf{w})\, \mathrm{d}\mathbf{w} = \int \sigma(a) p(a)\, \mathrm{d}a \qquad p(a) = \int \delta(a - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) q(\mathbf{w})\, \mathrm{d}\mathbf{w}$$

  - $\delta()$ is the Dirac delta function

Predictive Distribution

- **Evaluation**

$$p(a) = \int \delta(a - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) q(\mathbf{w}) \, \mathrm{d}\mathbf{w}$$

  - the delta function imposes a linear constraint on $\mathbf{w}$
  - leads to a marginal distribution of the joint distribution $q(\mathbf{w})$
  - obtained by integrating out all directions orthogonal to $\boldsymbol{\phi}$
  - the marginal must be Gaussian, since $q(\mathbf{w})$ is Gaussian

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\mathrm{MAP}}, \mathbf{S}_N)$$

$$\mu_a = \mathbb{E}[a] = \int p(a) a \, \mathrm{d}a = \int q(\mathbf{w}) \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi} \, \mathrm{d}\mathbf{w} = \mathbf{w}_{\mathrm{MAP}}^{\mathrm{T}} \boldsymbol{\phi}$$

$$\begin{aligned} \sigma_a^2 &= \mathrm{var}[a] = \int p(a) \left\{ a^2 - \mathbb{E}[a]^2 \right\} \mathrm{d}a \\ &= \int q(\mathbf{w}) \left\{ (\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi})^2 - (\mathbf{m}_N^{\mathrm{T}}\boldsymbol{\phi})^2 \right\} \mathrm{d}\mathbf{w} = \boldsymbol{\phi}^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi} \end{aligned}$$

$$\mathbf{m}_N = \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$$

Approximate Predictive Distribution

- **Variational approximation**

$$p(\mathcal{C}_1|\mathbf{t}) = \int \sigma(a)p(a)\,\mathrm{d}a = \int \sigma(a)\mathcal{N}(a|\mu_a, \sigma_a^2)\,\mathrm{d}a$$

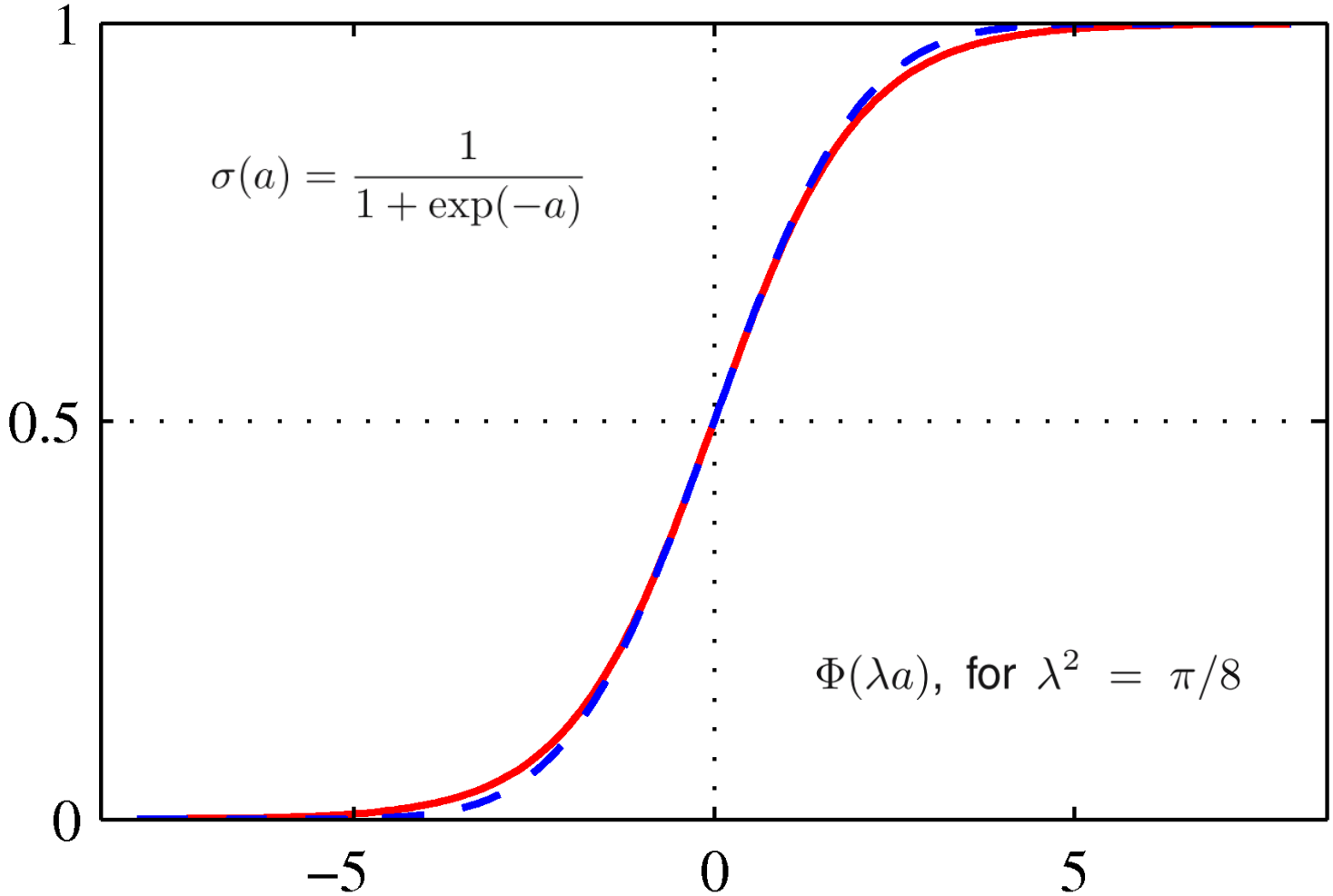
  - convolution of a logistic sigmoid with a Gaussian
  - cannot be evaluated analytically
  - approximate the logistic sigmoid with a probit function

$$\Phi(a) = \int_{-\infty}^{a} \mathcal{N}(\theta|0,1)\,\mathrm{d}\theta \qquad \sigma(a) \simeq \Phi(\lambda a)$$

  - rescale the probit to have identical slope at 0 $\qquad \lambda^2 = \pi/8$
  - the convolution of a probit and a Gaussian is another probit

$$\int \Phi(\lambda a)\mathcal{N}(a|\mu, \sigma^2)\,\mathrm{d}a = \Phi\left(\frac{\mu}{(\lambda^{-2} + \sigma^2)^{1/2}}\right)$$

Sigmoid (red) and Probit (blue)



Approximate Predictive Distribution

$$\int \Phi(\lambda a)\mathcal{N}(a|\mu,\sigma^2)\,\mathrm{d}a = \Phi\left(\frac{\mu}{(\lambda^{-2}+\sigma^2)^{1/2}}\right)$$

- **Approximate convolution**

$$\int \sigma(a)\mathcal{N}(a|\mu,\sigma^2)\,\mathrm{d}a \simeq \sigma\left(\kappa(\sigma^2)\mu\right) \qquad \kappa(\sigma^2) = (1+\pi\sigma^2/8)^{-1/2}$$

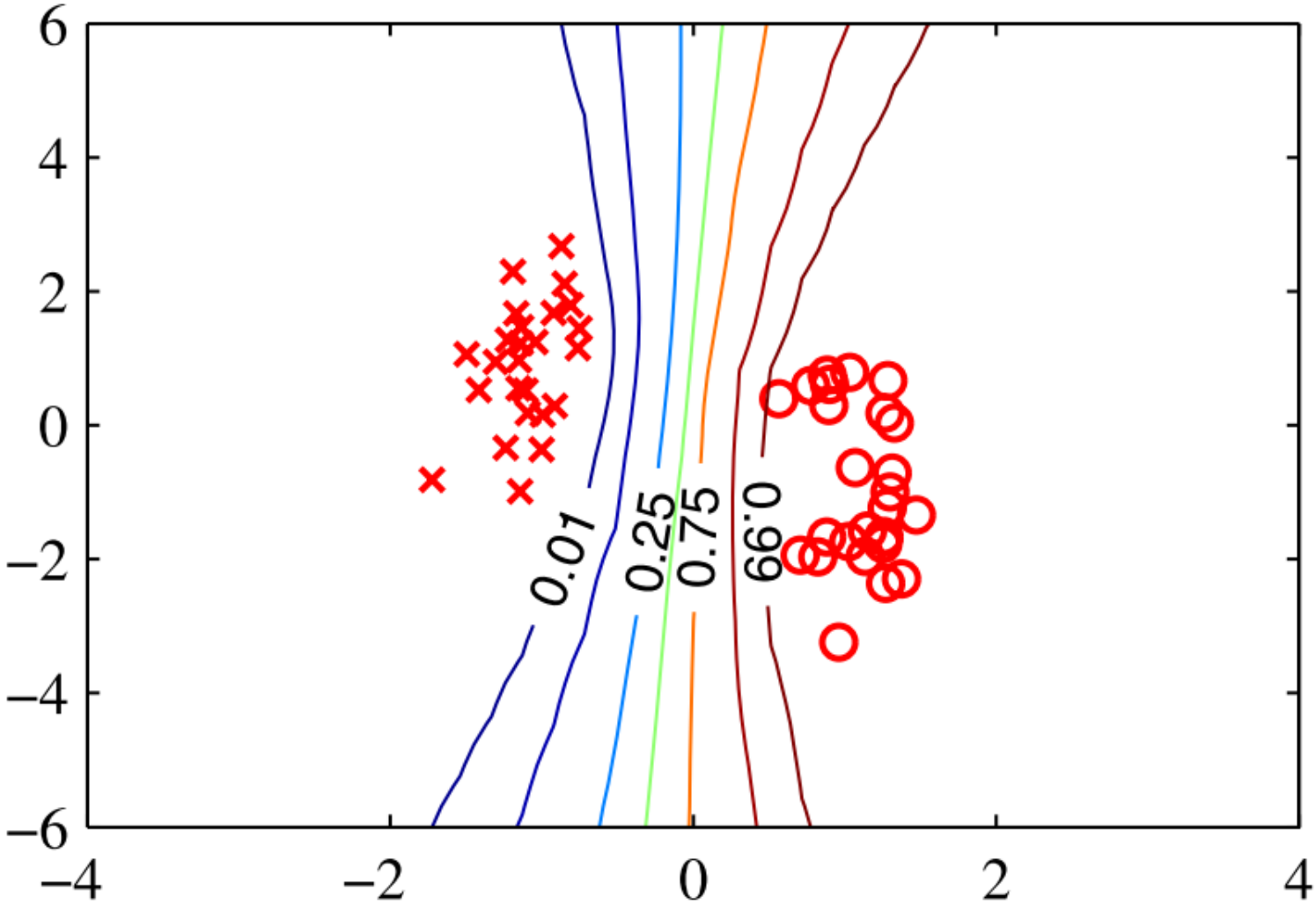
- **Approximate predictive distribution**

$$p(\mathcal{C}_1|\mathbf{t}) = \int \sigma(a)p(a)\,\mathrm{d}a = \int \sigma(a)\mathcal{N}(a|\mu_a,\sigma_a^2)\,\mathrm{d}a$$

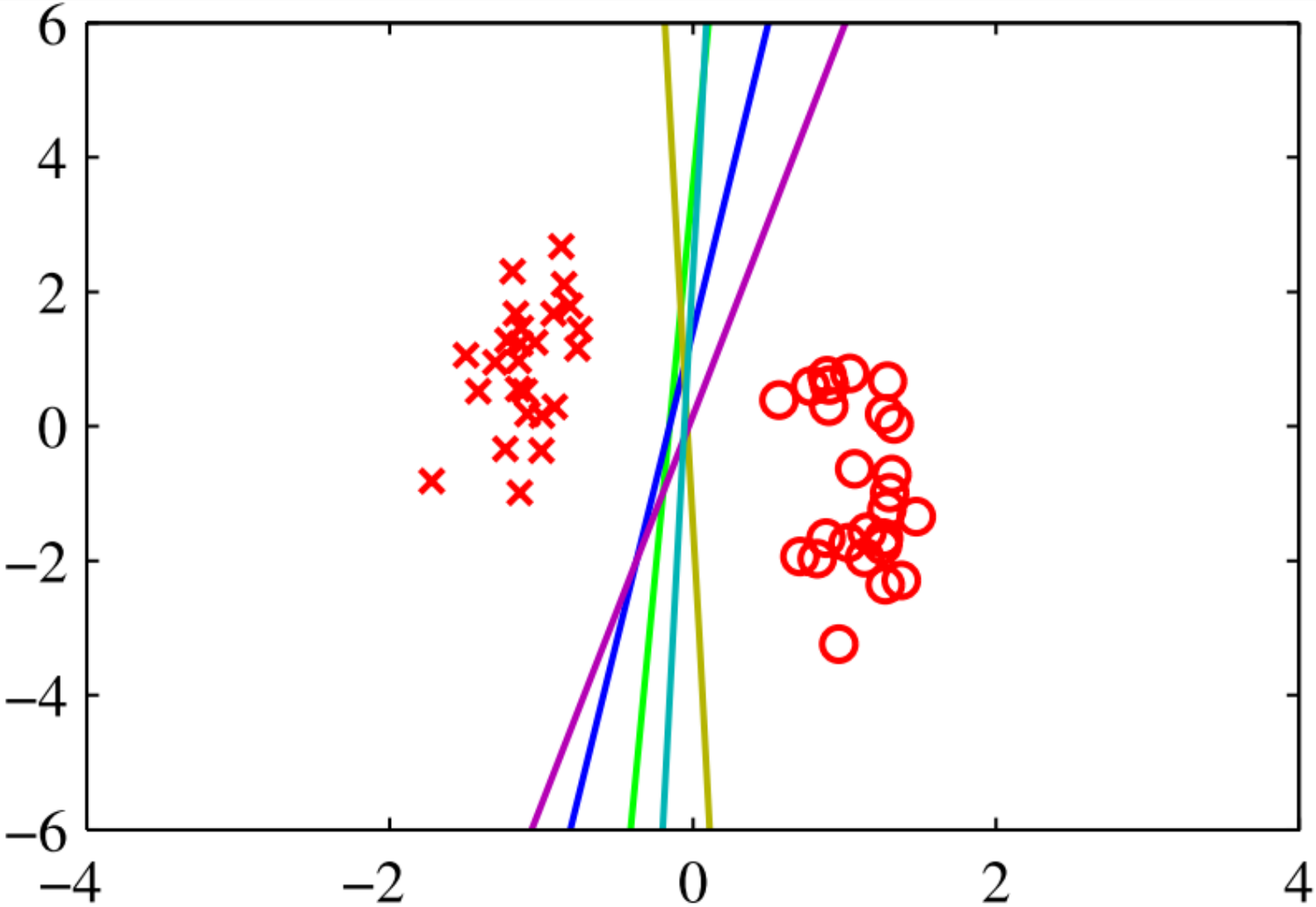
$$p(\mathcal{C}_1|\boldsymbol{\phi},\mathbf{t}) = \sigma\left(\kappa(\sigma_a^2)\mu_a\right) \qquad \mu_a = \mathbf{w}_{\mathrm{MAP}}^{\mathrm{T}}\boldsymbol{\phi} \qquad \sigma_a^2 = \boldsymbol{\phi}^{\mathrm{T}}\mathbf{S}_N\boldsymbol{\phi}$$

  - decision boundary $p(C_1|\phi,\mathbf{t})$=0.5 is given by μ_a=0 (same as MAP)
  - for minimizing misclassification with equal priors, no change
  - for other decision criteria, marginalization makes a difference

Predictive Distribution Example



Sampled ($\mathbf{w}$) Decision Boundaries



European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 12

Kernel Methods

TUC ECE, Spring 2023

Today

- **Kernel Methods**
- **Dual Representation**
- **Kernel Construction**
- **Kernel Regression**

Kernel Methods

Parametric vs. Non-Parametric

- **Parametric**
  - model is governed by a vector of adjustable parameters
  - *learning*: obtain a point estimate or a posterior distribution
  - training data are discarded after learning
  - predictions are based on the learned parameters
- **Non-parametric**
  - there are no adjustable parameters in the model
  - *learning*: select which of the training data to use and how
  - training data (or a subset of) are kept after learning
  - predictions are based on the kept training data
  - memory-based methods: fast learning, slow prediction
  - examples: Parzen probability density, nearest neighbors, …

Kernel Functions

- **Kernel function**
  - inner product of feature vectors (fixed nonlinear features ϕ)

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}} \phi(\mathbf{x}')$$

  - symmetric function over its arguments: $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$
  - the definition can be generalized even to discrete objects
- **Kernel substitution (kernel trick)**
  - formulate an algorithm so that $\mathbf{x}$ enters only in scalar products
  - replace the scalar products with a kernel
- **Types of kernels**
  - stationary (invariant to translations): $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$
  - homogeneous (depend on distance): $k(\mathbf{x}, \mathbf{x}') = k(\| \mathbf{x} - \mathbf{x}' \|)$

Kernel Methods

- **Kernel methods**
  - methods in which predictions are based on combinations of kernel functions evaluated between the input and data points
  - need a dual representation of the problem for kernelization
- **Kernelized algorithms**
  - nearest-neighbor classifiers
  - Fisher discriminant
  - support vector machines
  - relevance vector machines
  - principal component analysis
  - …

Dual Representation

Dual Representation (1)

- **Duality**
  - many linear models can be reformulated using a dual representation, where the kernel function arises naturally
- **Regularized Sum-of-Squares**

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} \qquad \lambda \geq 0$$

- **Solution**
  - setting the gradient to zero reveals the form of the solution

$$J'(\mathbf{w}) = \sum_{n=1}^{N}\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\}\boldsymbol{\phi}(\mathbf{x}_n) + \lambda\mathbf{w} = 0 \qquad a_n = -\frac{1}{\lambda}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}$$

$$\mathbf{a} = (a_1, \ldots, a_N)^{\mathrm{T}}$$

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) - t_n\right\}\boldsymbol{\phi}(\mathbf{x}_n) = \sum_{n=1}^{N} a_n\boldsymbol{\phi}(\mathbf{x}_n) = \boldsymbol{\Phi}^{\mathrm{T}}\mathbf{a}$$

linear combination of feature vectors!

Dual Representation (2)

- **Reformulation**
  - reformulate in terms of the new *N*-dim parameter vector $\mathbf{a}$

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{\Phi w} - \mathbf{t})^{\mathrm{T}}(\mathbf{\Phi w} - \mathbf{t}) + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$
$$= \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi w} - \mathbf{w}^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

  - substitute $\mathbf{w} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{a}$ into $J(\mathbf{w})$

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{\Phi\Phi}^{\mathrm{T}}\mathbf{\Phi\Phi}^{\mathrm{T}}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{\Phi\Phi}^{\mathrm{T}}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{\Phi\Phi}^{\mathrm{T}}\mathbf{a}$$

- **Kernelization**
  - $N \times N$ Gram matrix $\mathbf{K} = \mathbf{\Phi\Phi}^{\mathrm{T}}$ $\quad K_{nm} = \phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{KKa} - \mathbf{a}^{\mathrm{T}}\mathbf{Kt} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{Ka}$$

Dual Representation (3)

- **Solution**
  - setting the gradient of $J(\mathbf{a})$ to zero

$$J'(\mathbf{a}) = 0 \implies \mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{K}\mathbf{t} + \lambda\mathbf{K}\mathbf{a} = 0 \implies \mathbf{K}(\mathbf{K}\mathbf{a} + \lambda\mathbf{a} - \mathbf{t}) = 0$$

  - or, alternatively, substituting $\mathbf{w} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{a}$ into the solution for $\mathbf{w}$

$$\mathbf{w} = -\tfrac{1}{\lambda}\mathbf{\Phi}^{\mathrm{T}}(\mathbf{\Phi}\mathbf{w} - \mathbf{t}) \implies -\lambda\mathbf{\Phi}^{\mathrm{T}}\mathbf{a} = \mathbf{\Phi}^{\mathrm{T}}(\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{a} - \mathbf{t}) \implies \mathbf{\Phi}^{\mathrm{T}}(\mathbf{K}\mathbf{a} + \lambda\mathbf{a} - \mathbf{t}) = 0$$

  - solving for $\mathbf{a}$ leads to an $N\times N$ inversion (vs. $M\times M$ for $\mathbf{w}$, $M \ll N$)

$$\mathbf{K}\mathbf{a} + \lambda\mathbf{a} - \mathbf{t} = 0 \implies \mathbf{a} = (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}$$

- **Prediction**

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) = \phi(\mathbf{x})^{\mathrm{T}}\mathbf{w} = \phi(\mathbf{x})^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{a} = \mathbf{k}(\mathbf{x})^{\mathrm{T}}(\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}$$

  - need N-dim vector $\mathbf{k}(\mathbf{x}) = \mathbf{\Phi}\phi(\mathbf{x}) \quad k_n(\mathbf{x}) = \phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}) = k(\mathbf{x}_n, \mathbf{x})$
  - the prediction is a function of the data, no parameters!
  - no explicit computation of $\phi(\mathbf{x})$, computation of kernels only

Kernel Construction

Kernel Definition Approaches

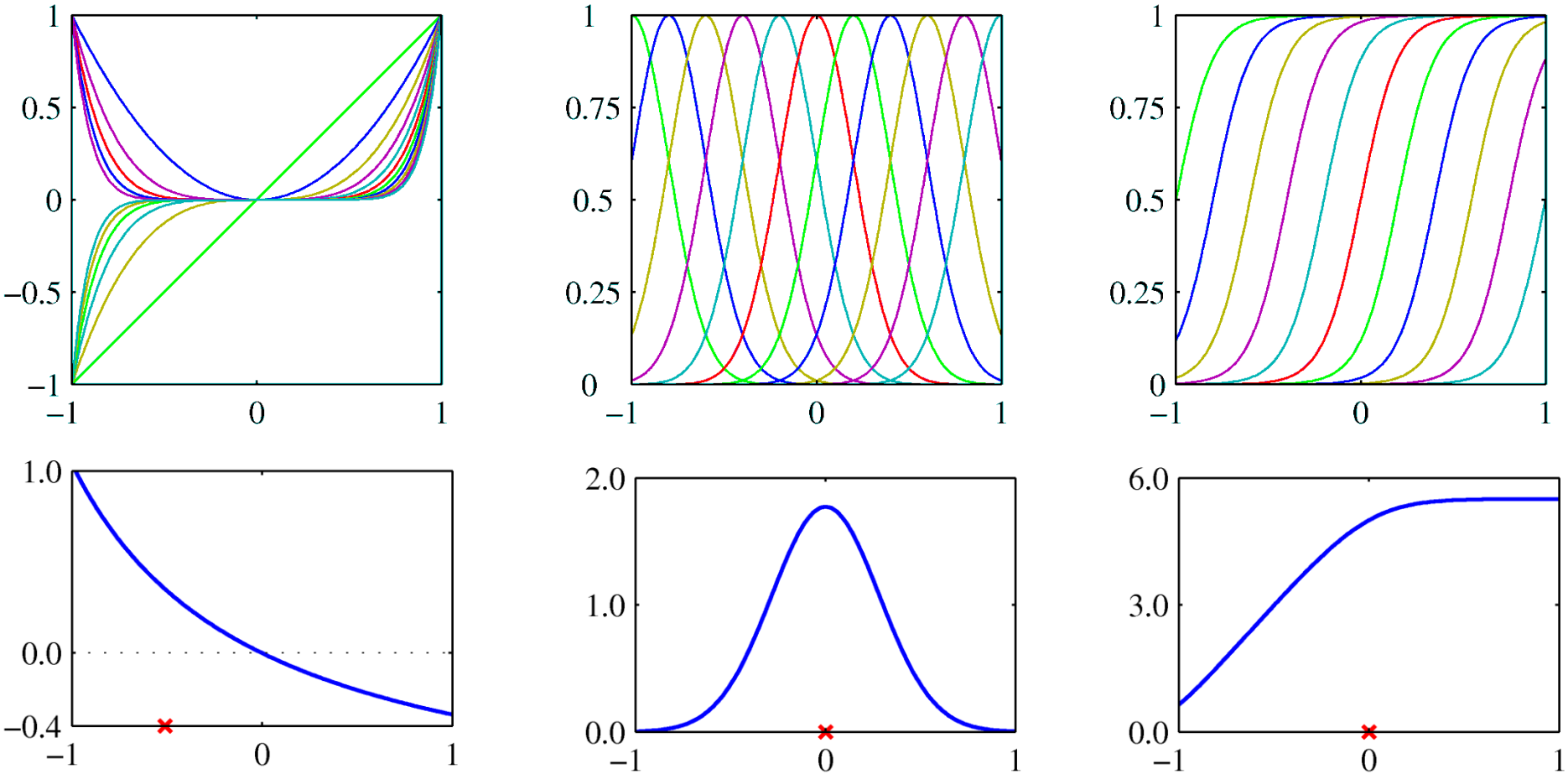
- **Indirect**
  - choose feature space $\boldsymbol{\phi}(\mathbf{x})$ first and then construct the kernel

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}') = \sum_{i=1}^{M} \phi_i(\mathbf{x}) \phi_i(\mathbf{x}')$$

- **Direct**
  - choose the kernel directly, but ensure it is a valid one
  - must correspond to a scalar product in some feature space
  - the Gram matrix $\mathbf{K}$ must be positive semidefinite for all $\{\mathbf{x}_n\}$
  - the dimension *M* of the feature space can be arbitrary

Indirect Approach Example

feature spaces and kernels (x' at red mark) for polynomial, Gaussian, logistic sigmoid basis



Direct Approach Example

- **Kernel function**

$$k(\mathbf{x}, \mathbf{z}) = \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2$$

- **Two-dimensional input space** $\mathbf{x} = (x_1, x_2) \quad \mathbf{z} = (z_1, z_2)$

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2 = (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= (x_1^2, \sqrt{2} x_1 x_2, x_2^2)(z_1^2, \sqrt{2} z_1 z_2, z_2^2)^{\mathrm{T}} \\ &= \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{z}). \end{aligned}$$

  – it is a valid kernel!

- **Feature space**

$$\boldsymbol{\phi}(\mathbf{x}) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)^{\mathrm{T}}$$

Structural Kernel Definition

– given valid kernels $k_1(\mathbf{x},\mathbf{x}')$ and $k_2(\mathbf{x},\mathbf{x}')$, the following are also valid:

$k(\mathbf{x},\mathbf{x}') = ck_1(\mathbf{x},\mathbf{x}')$	$c > 0$ is a constant
$k(\mathbf{x},\mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x},\mathbf{x}')f(\mathbf{x}')$	$f(.)$ is any function
$k(\mathbf{x},\mathbf{x}') = q\left(k_1(\mathbf{x},\mathbf{x}')\right)$	$q(.)$ is a polynomial with coeff ≥ 0
$k(\mathbf{x},\mathbf{x}') = \exp\left(k_1(\mathbf{x},\mathbf{x}')\right)$	
$k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}') + k_2(\mathbf{x},\mathbf{x}')$	
$k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}')k_2(\mathbf{x},\mathbf{x}')$	
$k(\mathbf{x},\mathbf{x}') = k_3\left(\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{x}')\right)$	$\boldsymbol{\phi}(\mathbf{x})$ is a feature space mapping
$k(\mathbf{x},\mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}'$	$\mathbf{A}$ is symmetric positive semidefinite
$k(\mathbf{x},\mathbf{x}') = k_a(\mathbf{x}_a,\mathbf{x}_a') + k_b(\mathbf{x}_b,\mathbf{x}_b')$	$\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$
$k(\mathbf{x},\mathbf{x}') = k_a(\mathbf{x}_a,\mathbf{x}_a')k_b(\mathbf{x}_b,\mathbf{x}_b')$	$k_a(\mathbf{x}_a,\mathbf{x}_a')$ and $k_b(\mathbf{x}_b,\mathbf{x}_b')$ are valid

Kernel Examples

- **Polynomial**
  - $k(\mathbf{x},\mathbf{x}') = (\mathbf{x}^{\mathrm{T}}\mathbf{x}')^2$ contains only terms of degree 2
  - $k(\mathbf{x},\mathbf{x}') = (\mathbf{x}^{\mathrm{T}}\mathbf{x}'+c)^2$, $c > 0$, contains terms of degree up to 2
  - $k(\mathbf{x},\mathbf{x}') = (\mathbf{x}^{\mathrm{T}}\mathbf{x}')^M$ contains only terms of degree M
  - $k(\mathbf{x},\mathbf{x}') = (\mathbf{x}^{\mathrm{T}}\mathbf{x}'+c)^M$, $c > 0$, contains terms of degree up to M
- **Gaussian**
  - $k(\mathbf{x},\mathbf{x}') = \exp(-\|\mathbf{x}-\mathbf{x}'\|^2/2\sigma^2)$, infinite dimensional feature space
- **Sigmoidal**
  - $k(\mathbf{x},\mathbf{x}') = \tanh(a\mathbf{x}^{\mathrm{T}}\mathbf{x}'+b)$, Gram may not be positive semidefinite
- **Non-vectorial**
  - $k(\mathrm{A}_1,\mathrm{A}_2) = 2^{|\mathrm{A}_1\cap\mathrm{A}_2|}$ over subsets of a set

The Gaussian Kernel

- defining

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2\right)$$

- expanding the square

$$\|\mathbf{x} - \mathbf{x}'\|^2 = \mathbf{x}^{\mathrm{T}}\mathbf{x} + (\mathbf{x}')^{\mathrm{T}}\mathbf{x}' - 2\mathbf{x}^{\mathrm{T}}\mathbf{x}'$$

- substituting

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\mathbf{x}^{\mathrm{T}}\mathbf{x}/2\sigma^2\right) \exp\left(\mathbf{x}^{\mathrm{T}}\mathbf{x}'/\sigma^2\right) \exp\left(-(\mathbf{x}')^{\mathrm{T}}\mathbf{x}'/2\sigma^2\right)$$

- using valid linear kernel and kernel properties

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{x}'$$

$$k(\mathbf{x}, \mathbf{x}') = c k_1(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right)$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}')$$

Kernel Regression

Application to Regression

- **Given**
  - training set $\{\mathbf{x}_n, t_n\}$ of inputs and targets
- **Joint distribution**
  - Parzen density estimator for the joint distribution $p(\mathbf{x}, t)$

$$p(\mathbf{x}, t) = \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{x} - \mathbf{x}_n, t - t_n)$$

  - component density functions $f(\mathbf{x}, t)$ centered on data
- **Goal**
  - an expression for the regression function $y(\mathbf{x}) = p(t|\mathbf{x})$
  - conditional average of target t conditioned on input $\mathbf{x}$

Kernel Regression

$$y(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int_{-\infty}^{\infty} t p(t|\mathbf{x})\,\mathrm{d}t = \frac{\int t p(\mathbf{x},t)\,\mathrm{d}t}{\int p(\mathbf{x},t)\,\mathrm{d}t} = \frac{\sum_n \int t f(\mathbf{x}-\mathbf{x}_n, t-t_n)\,\mathrm{d}t}{\sum_m \int f(\mathbf{x}-\mathbf{x}_m, t-t_m)\,\mathrm{d}t}$$

- zero mean components and change of variables

$$\int_{-\infty}^{\infty} f(\mathbf{x},t)t\,\mathrm{d}t = 0 \qquad g(\mathbf{x}) = \int_{-\infty}^{\infty} f(\mathbf{x},t)\,\mathrm{d}t$$

- Nadaraya-Watson model (kernel regression)

$$y(\mathbf{x}) = \frac{\sum_n g(\mathbf{x}-\mathbf{x}_n)t_n}{\sum_m g(\mathbf{x}-\mathbf{x}_m)} = \sum_n k(\mathbf{x},\mathbf{x}_n)t_n$$

$$k(\mathbf{x},\mathbf{x}_n) = \frac{g(\mathbf{x}-\mathbf{x}_n)}{\sum_m g(\mathbf{x}-\mathbf{x}_m)}$$

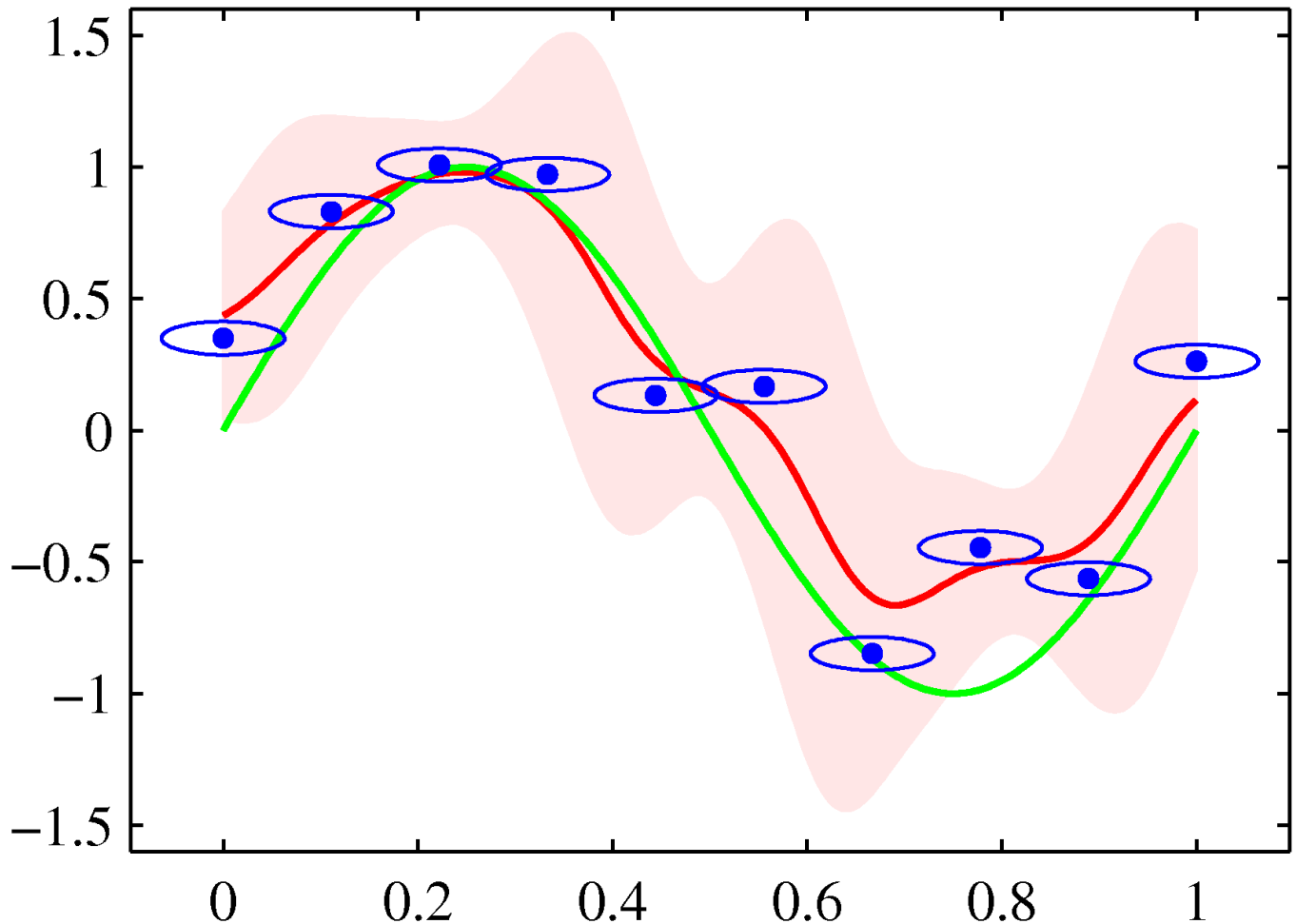
$$\sum_{n=1}^{N} k(\mathbf{x},\mathbf{x}_n) = 1$$

- full conditional distribution

$$p(t|\mathbf{x}) = \frac{p(t,\mathbf{x})}{\int p(t,\mathbf{x})\,\mathrm{d}t} = \frac{\sum_n f(\mathbf{x}-\mathbf{x}_n, t-t_n)}{\sum_m \int f(\mathbf{x}-\mathbf{x}_m, t-t_m)\,\mathrm{d}t}$$

Kernel Regression Example

- **Densities** $f(x,t)$
  - zero-mean
  - isotropic
  - Gaussian
- **Legend**
  - truth (green)
  - data (blue)
  - regression (red)
  - variance (shade)



European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 13

Gaussian Processes for Regression

TUC ECE, Spring 2023

Today

- **Gaussian Processes**
- **Gaussian Process Regression**
- **Learning Hyperparameters**

Gaussian Processes

Motivation

- **Bayesian linear regression**
  - parametric linear model in a (non-linear) feature space
  - prior distribution over the parameters …
  - … induces prior distribution over regression functions
  - given training set, posterior distribution over the parameters …
  - … yields a posterior distribution over regression functions
  - … and implies a predictive distribution (with addition of noise)
- **Gaussian processes**
  - directly define a prior distribution over regression functions
  - … then infer the posterior distribution over regression functions
  - *problem*: uncountably infinite space of regression functions!
  - *idea*: can focus only on their values at the data points (finite)

Linear Regression Revisited

- **Model**
  - linear combination of *M* fixed features $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$
- **Prior**
  - isotropic Gaussian $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$
- **Distributions**
  - each value of $\mathbf{w}$ defines some regression function $y(\mathbf{x})$
  - prior over $\mathbf{w}$ induces distribution over regression functions y
  - for data points $\mathbf{x}_1$, $\mathbf{x}_2$, … $\mathbf{x}_N$, we care about $y(\mathbf{x}_1)$, $y(\mathbf{x}_2)$, … $y(\mathbf{x}_N)$
  - what is the joint distribution of the regression function values?

Linear Regression Revisited

- **Output**
  - vector **y** with elements $y_n = y(\mathbf{x}_n)$
  - can be written as **y** = $\mathbf{\Phi w}$, $\mathbf{\Phi}$ is the design matrix $\Phi_{nk} = \phi_k(\mathbf{x}_n)$
- **Distribution**
  - **y** is a linear combination of Gaussian distributed variables ($\mathbf{w}$)
  - hence, the distribution over **y** must be Gaussian
  - only need to find mean and covariance

$$\begin{aligned} \mathbb{E}[\mathbf{y}] &= \mathbf{\Phi}\mathbb{E}[\mathbf{w}] = \mathbf{0} \\ \mathrm{cov}[\mathbf{y}] &= \mathbb{E}\left[\mathbf{y}\mathbf{y}^{\mathrm{T}}\right] = \mathbf{\Phi}\mathbb{E}\left[\mathbf{w}\mathbf{w}^{\mathrm{T}}\right]\mathbf{\Phi}^{\mathrm{T}} = \frac{1}{\alpha}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} = \mathbf{K} \end{aligned}$$

This is an example of a Gaussian process!

  - $\mathbf{K}$ is the Gram matrix $K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha}\phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m)$

Gaussian Stochastic Processes

- **Definition**
  - a probability distribution over functions $y(\mathbf{x})$, so that the values of $y(\mathbf{x})$ at points $\mathbf{x}_1$, $\mathbf{x}_2$, … $\mathbf{x}_N$ jointly have a Gaussian distribution
  - for two dimensional input, it is known as Gaussian random field
  - specified by the joint probability distribution over $\mathbf{y}_1$, $\mathbf{y}_2$, …, $\mathbf{y}_N$
  - being Gaussian, it can be specified by second-order statistics
  - mean is commonly taken to be zero, thus need only covariance
  - covariance of $y(\mathbf{x})$ at any two points $\mathbf{x}_n$, $\mathbf{x}_m$ given by a kernel

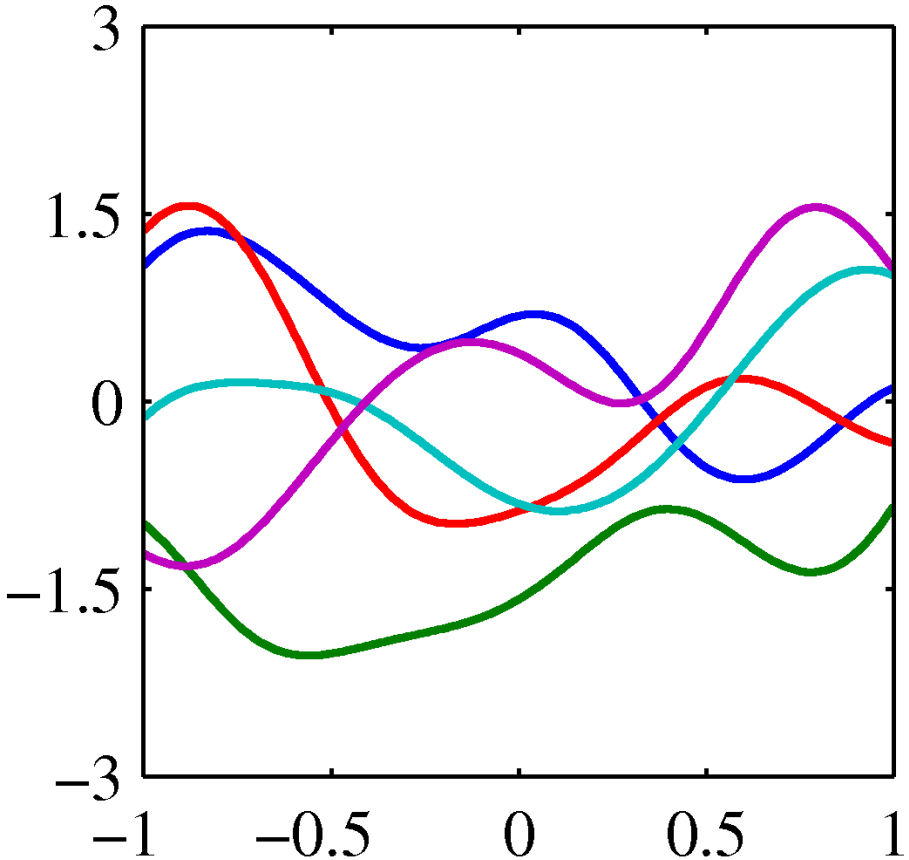
$$\mathbb{E}\left[y(\mathbf{x}_n)y(\mathbf{x}_m)\right] = k(\mathbf{x}_n, \mathbf{x}_m)$$

  - the kernel function can be defined indirectly (feature vector) …
  - … or the kernel function can be defined directly (no features)

Functions from Gaussian Processes

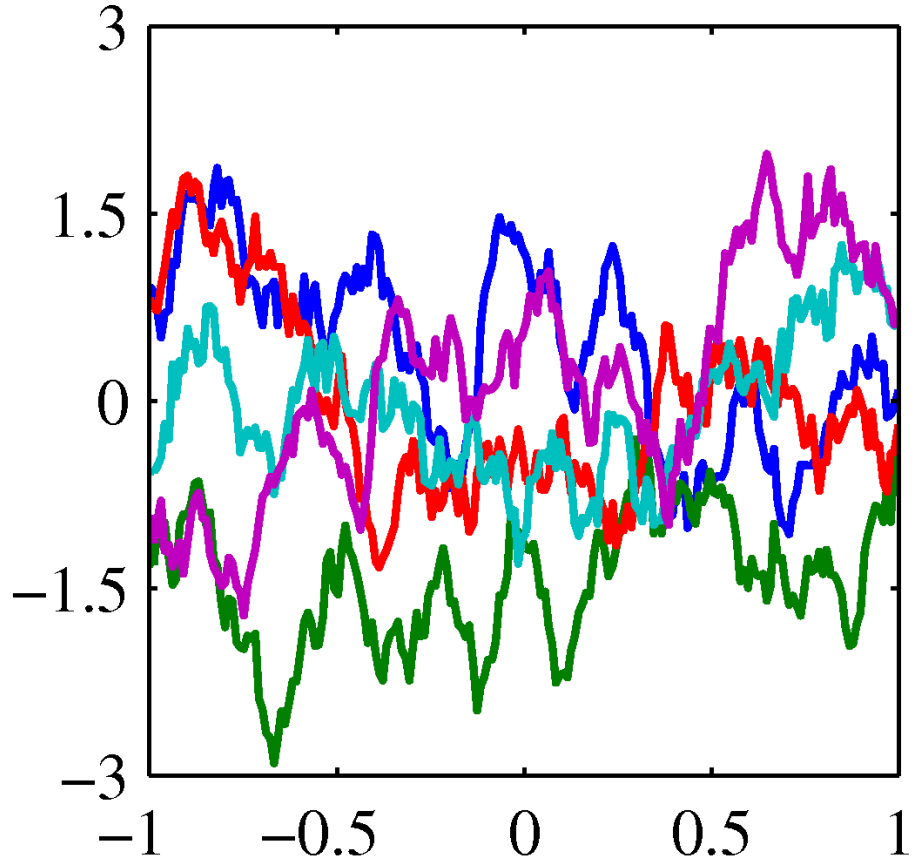
Gaussian kernel

$$k(x, x') = \exp(-\|x - x'\|^2/2\sigma^2)$$



exponential kernel

$$k(x, x') = \exp\left(-\theta\,|x - x'|\right)$$



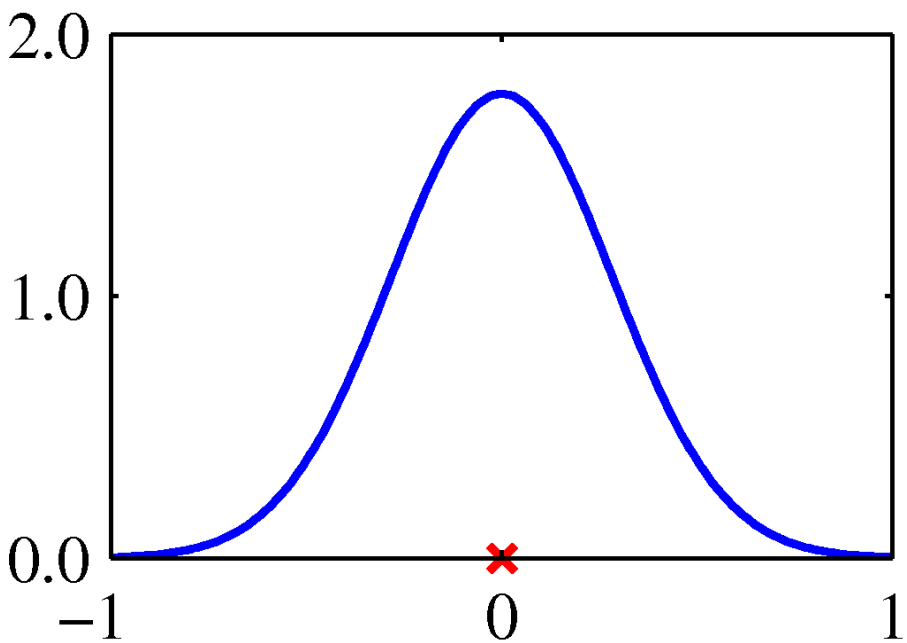
Functions from Gaussian Processes

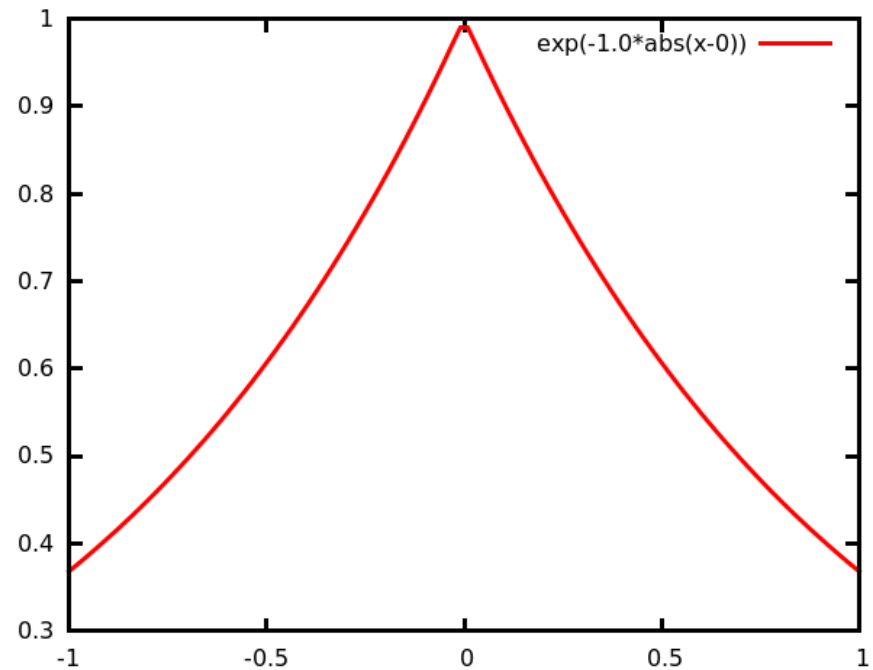
Gaussian kernel

$$k(x, x') = \exp(-\|x - x'\|^2/2\sigma^2)$$

exponential kernel

$$k(x, x') = \exp\left(-\theta\,|x - x'|\right)$$





Gaussian Process Regression

Gaussian Process for Regression

- **Output noise**
  - for $y_n = y(\mathbf{x}_n)$ and random noise ϵ_n, the target is $t_n = y_n + \epsilon_n$
  - for Gaussian noise with precision β: $p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1})$
- **Joint distribution**

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I}_N) \qquad \mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}) \qquad \mathbf{y} = (y_1, \ldots, y_N)^{\mathrm{T}}$$

  - for similar inputs $\mathbf{x}_n$, $\mathbf{x}_m$ the outputs $\mathbf{y}_n$, $\mathbf{y}_m$ will be correlated
- **Marginal distribution**

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y})\,\mathrm{d}\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C})$$

$$\mathbf{C} = \mathbf{K} + \beta^{-1}\mathbf{I}$$

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}\left(\mathbf{t}|\mathbf{A}\mathbf{y} + \mathbf{b}, \mathbf{L}^{-1}\right)$$

$$p(\mathbf{y}) = \mathcal{N}\left(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right)$$

$$p(\mathbf{t}) = \mathcal{N}(\mathbf{t}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}})$$

  - independent Gaussian randomness, so covariances add

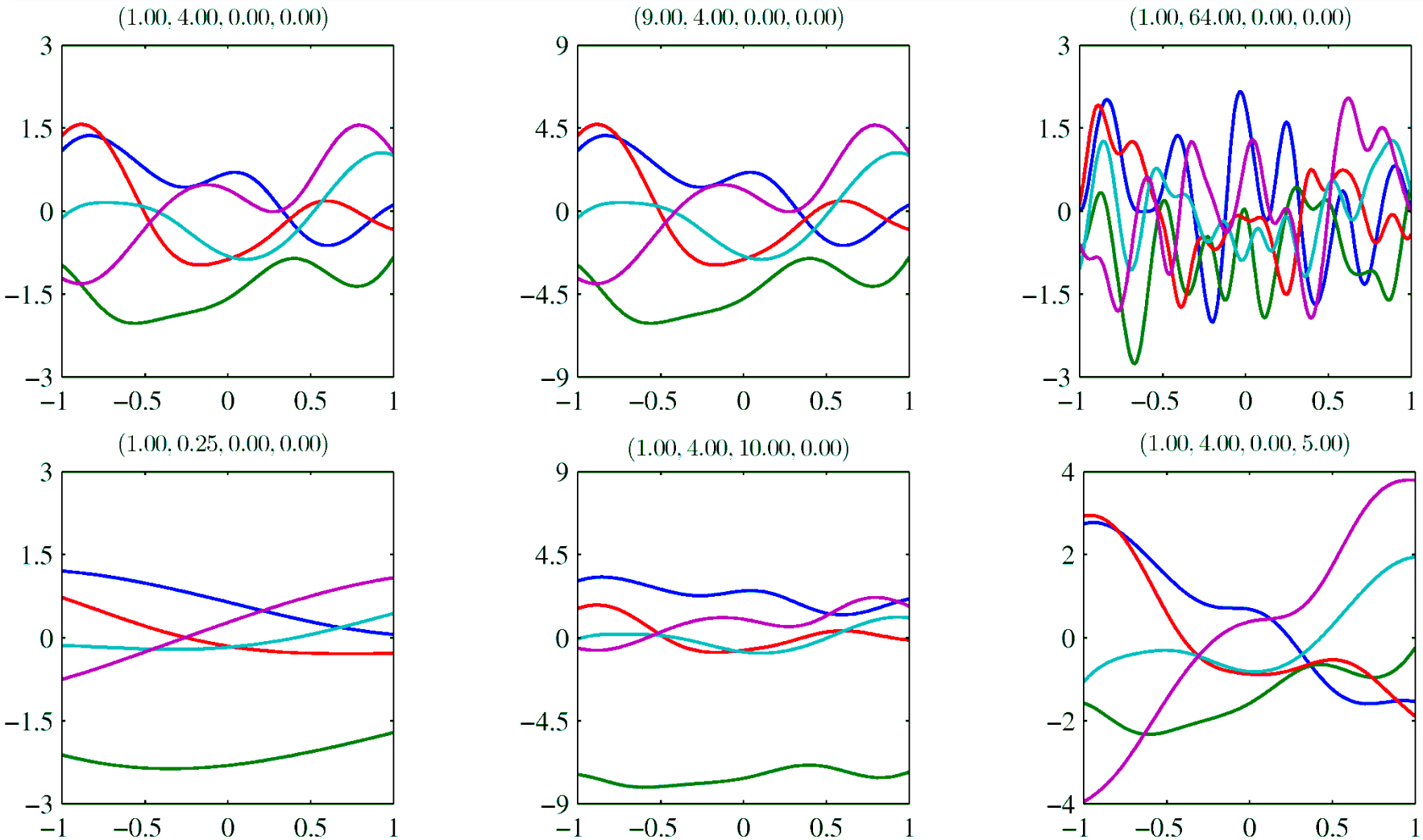
Example Kernel for Regression

- **Kernel**
  - exponential of quadratic form with linear and constant terms

$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp\left\{-\frac{\theta_1}{2}\|\mathbf{x}_n - \mathbf{x}_m\|^2\right\} + \theta_2 + \theta_3 \mathbf{x}_n^{\mathrm{T}} \mathbf{x}_m$$

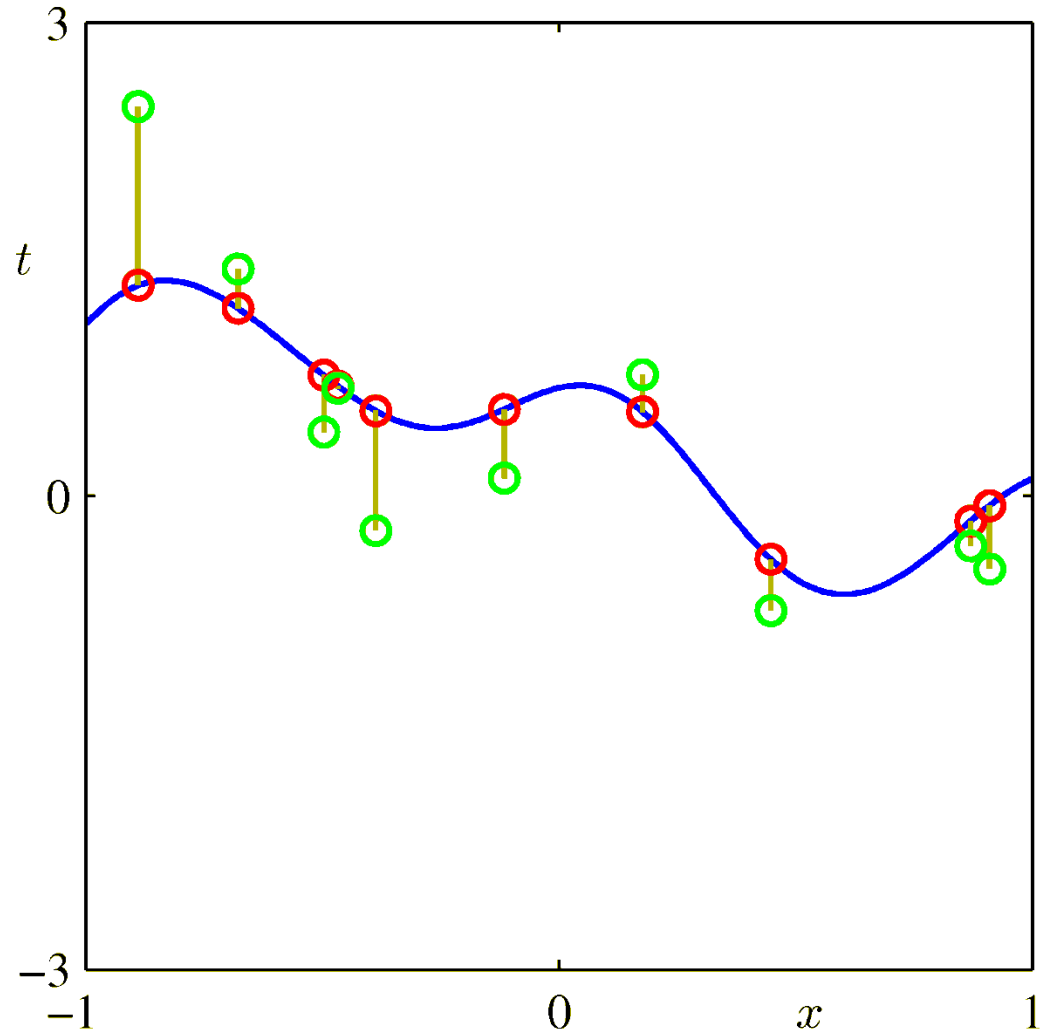
- **Hyperparameters** $(\theta_0, \theta_1, \theta_2, \theta_3)$
  - θ_0 : weight on quadratic term
  - θ_1 : scaling of exponential
  - θ_2 : weight on constant term
  - θ_3 : weight on linear term

Regression Function Samples $(\theta_0, \theta_1, \theta_2, \theta_3)$

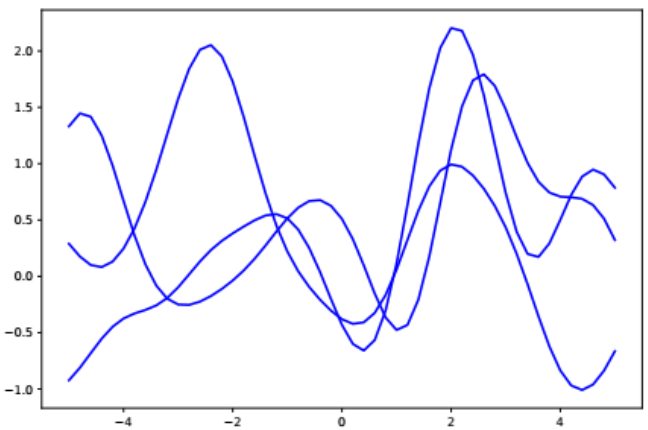


Sampling of Data Points Example

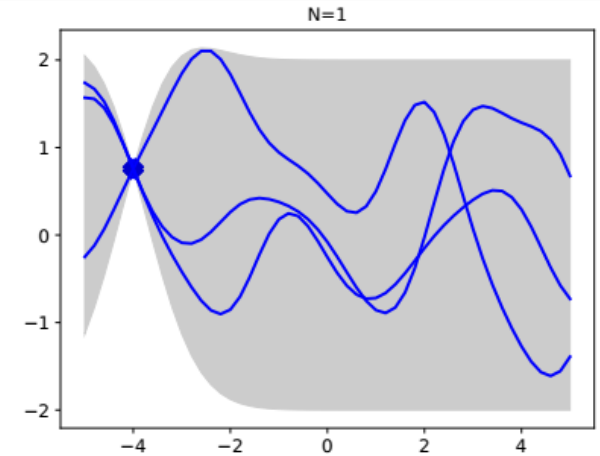
- **Kernel**
  - exponential of quadratic …
  - … plus linear
  - … plus constant
- **Legend**
  - regression function (blue)
  - 10 data points
  - sample outputs (red)
  - additive noise (yellow)
  - sample targets (green)



Data Conditioning Example

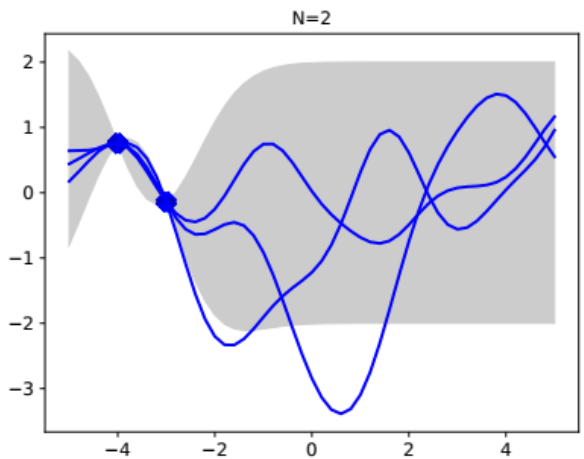


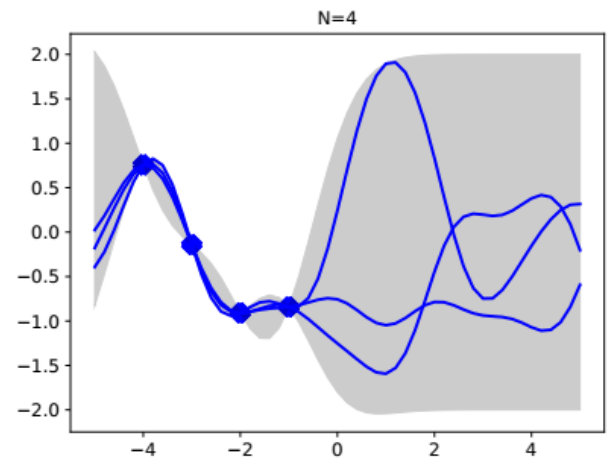
(a)



(b)

$$\mathbb{E}\left[f(\boldsymbol{x})\right] \pm 2\mathrm{std}\left[f(\boldsymbol{x})\right]$$





Making Predictions

- **Prediction**
  - so far, only model of the joint distribution over data points
  - regression is about making predictions at new data points
  - given a training set: targets $\mathbf{t}_N = (t_1, t_2, \ldots, t_N)^{\mathrm{T}}$ for $\mathbf{x}_1$, $\mathbf{x}_2$, … $\mathbf{x}_N$
  - … we need to predict target t_{N+1} for new input $\mathbf{x}_{N+1}$
- **Gaussian process prediction**
  - we need to evaluate the predictive distribution $p(t_{N+1} | \mathbf{t}_N)$
  - conditioned also on $\mathbf{x}_1$, $\mathbf{x}_2$, … $\mathbf{x}_N$, $\mathbf{x}_{N+1}$, but dropped for clarity
  - obtain conditional distribution from the joint one for N+1

Gaussian Process Prediction

- **Joint distribution**

$$p(\mathbf{t}_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1})$$

$$\mathbf{t}_{N+1} = (\underbrace{t_1, t_2, \ldots, t_N}_{\mathbf{t}_N^{\mathrm{T}}}, t_{N+1})^{\mathrm{T}}$$

$$\mathbf{C}_{N+1} = \mathbf{K}_{N+1} + \beta^{-1}\mathbf{I}_{N+1}$$

$$\mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^{\mathrm{T}} & c \end{pmatrix}$$

$$\mathbf{k} = \{k(\mathbf{x}_n, \mathbf{x}_{N+1})\}, n = 1, 2, \ldots, N$$

$$c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$$

- **Conditional distribution**
  - $p(t_{N+1}|\mathbf{t}_N)$ is also Gaussian
  - mean $m(\mathbf{x}_{N+1}) = \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t}_N$
  - variance $\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}$
  - dependence on $\mathbf{x}_{N+1}$ through $\mathbf{k}$

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{pmatrix} \quad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}$$

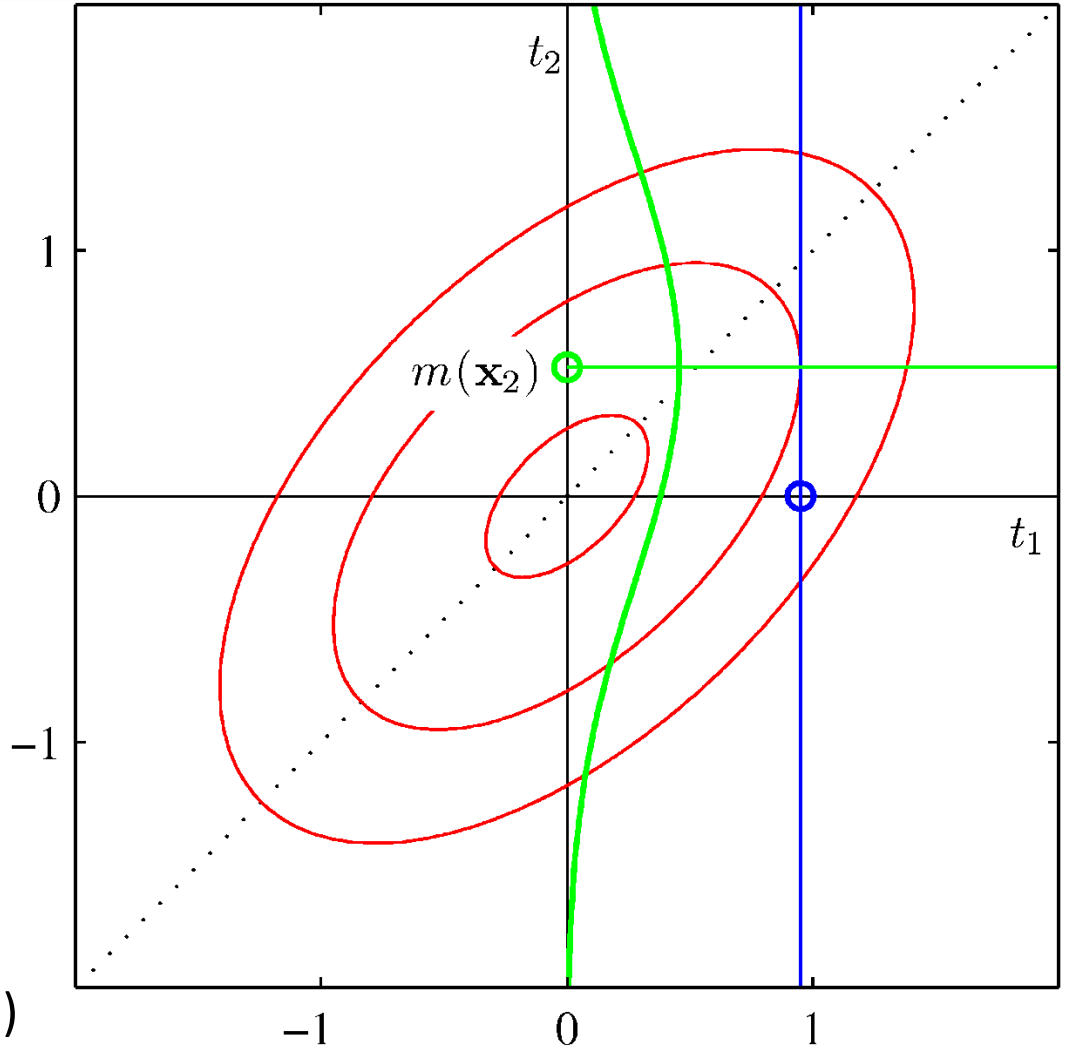
$$p(\mathbf{x}_b|\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_b|\boldsymbol{\mu}_{b|a}, \boldsymbol{\Sigma}_{b|a})$$

$$\boldsymbol{\mu}_{b|a} = \boldsymbol{\mu}_b + \boldsymbol{\Sigma}_{ba}\boldsymbol{\Sigma}_{aa}^{-1}(\mathbf{x}_a - \boldsymbol{\mu}_a)$$

$$\boldsymbol{\Sigma}_{b|a} = \boldsymbol{\Sigma}_{bb} - \boldsymbol{\Sigma}_{ba}\boldsymbol{\Sigma}_{aa}^{-1}\boldsymbol{\Sigma}_{ab}$$

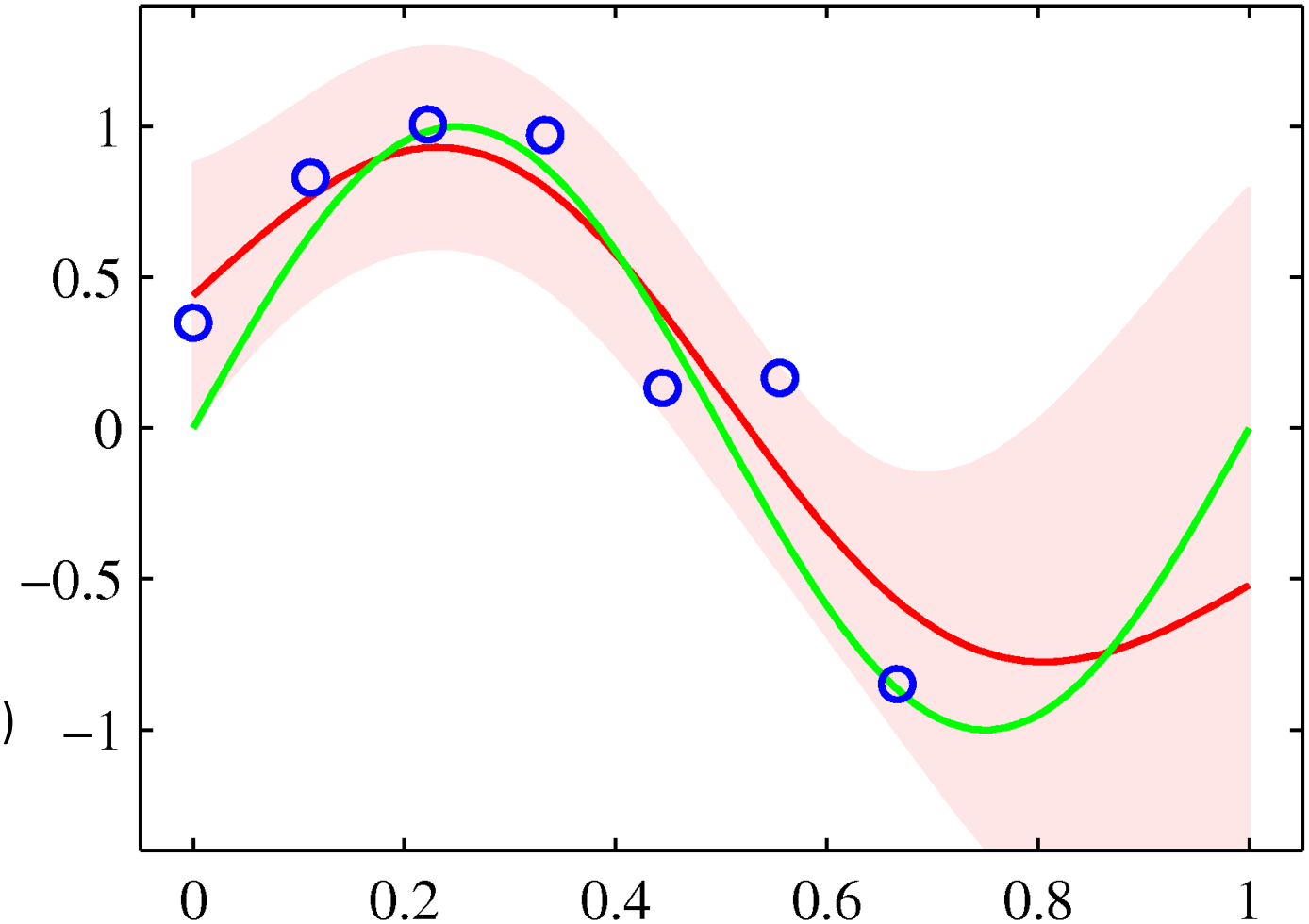
Conditional Distribution Example

- **Kernel**
  - exponential of quadratic …
  - … plus linear
  - … plus constant
- **Legend**
  - training data point (blue)
  - test data point (green)
  - joint distribution (red)
    - 2D zero-mean Gaussian
  - conditioning (blue line)
  - conditional (green curve)
    - 1D Gaussian
  - conditional mean (green line)



Gaussian Process Regression Example

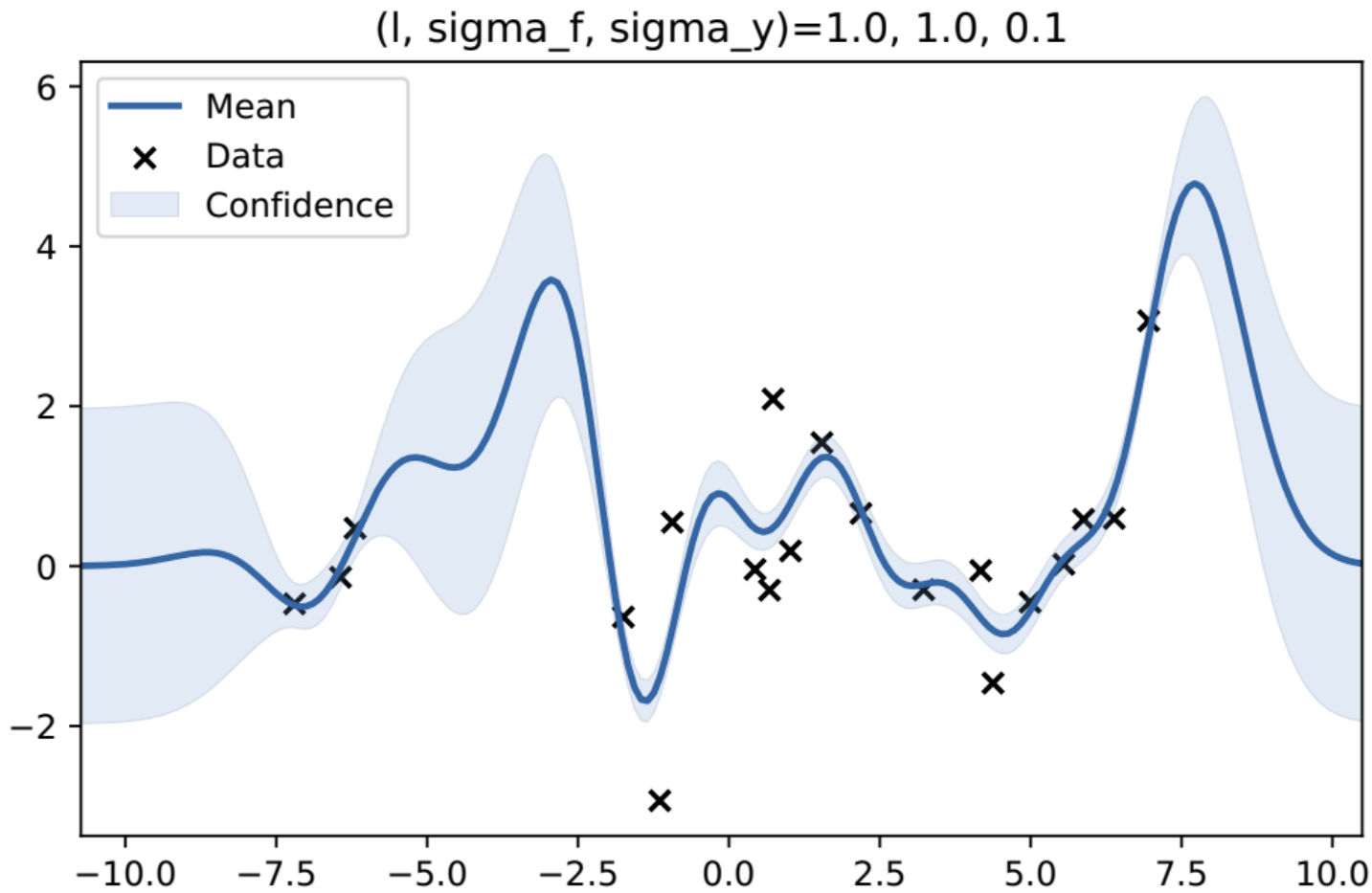
- **Kernel**
  - exponential of quadratic …
  - … plus linear
  - … plus constant
- **Legend**
  - truth (green)
  - data (blue)
  - regression (red)
  - variance (shade)



GP Regression Example 1

$$\mathcal{K}(x_p, x_q) = \sigma_f^2 \exp(-\frac{1}{2\ell^2}(x_p - x_q)^2)$$

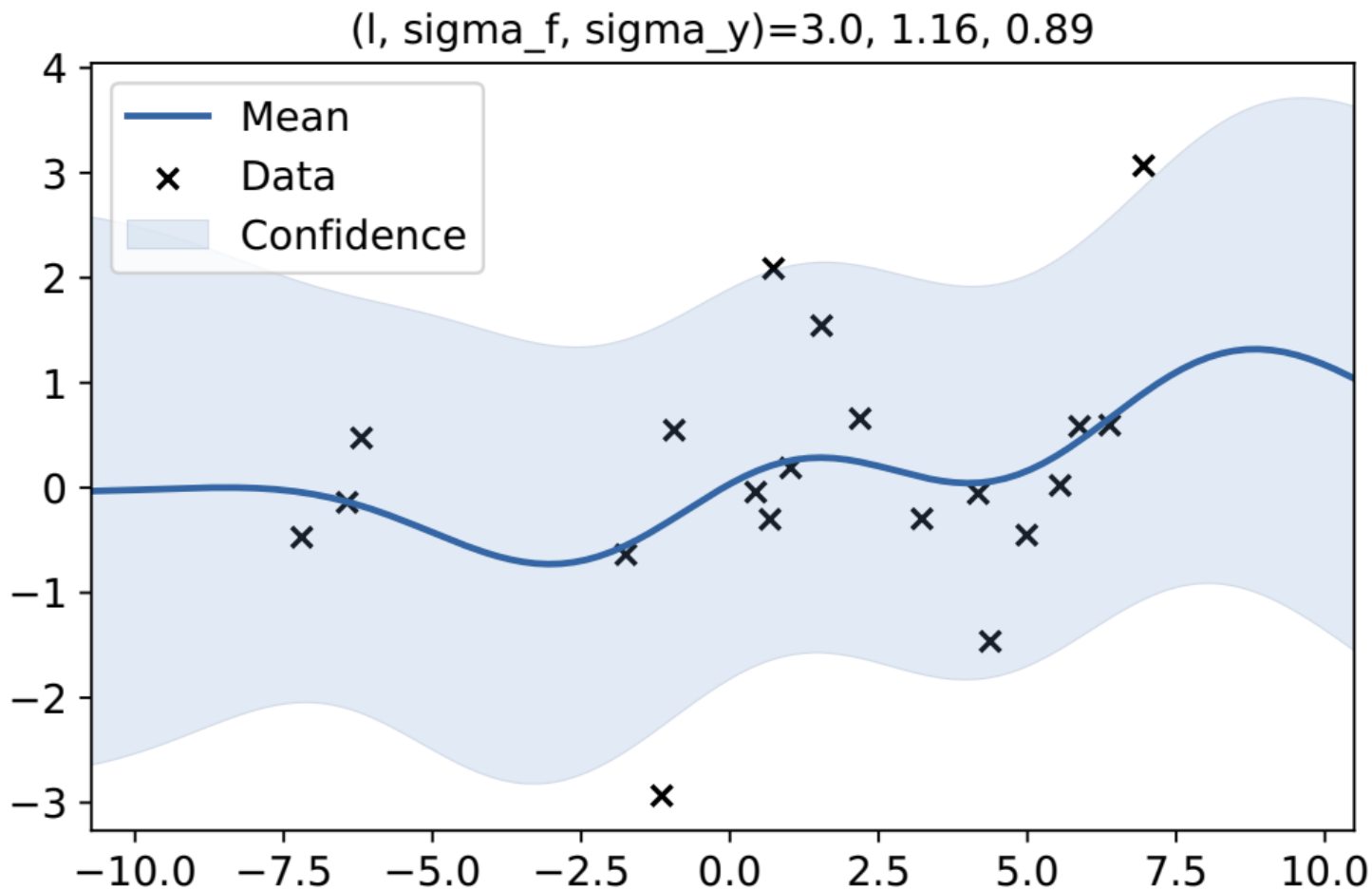
- **Kernel**
  - exponential of quadratic
  - params: σ_f , l
- **Legend**
  - data (x)
  - regression (dark blue)
  - variance (blue shade)
  - observation noise σ_y



GP Regression Example 2

$$\mathcal{K}(x_p, x_q) = \sigma_f^2 \exp(-\frac{1}{2\ell^2}(x_p - x_q)^2)$$

- **Kernel**
  - exponential of quadratic
  - params: σ_f , l
- **Legend**
  - data (x)
  - regression (dark blue)
  - variance (blue shade)
  - observation noise σ_y



Constructing Kernels for Regression

- **Requirements**
  - covariance matrix $\mathbf{C}$ must be positive definite
  - if λ_i is an eigenvalue of $\mathbf{K}$, then $\lambda_i + \beta^{-1}$ is an eigenvalue of $\mathbf{C}$
  - since $\beta > 0$, it suffices $\lambda_i \geq 0$ or that $\mathbf{K}$ is positive semidefinite
- **Kernel construction**
  - same, typical, requirements for kernel construction
- **Expansion**
  - mean of predictive dist $m(\mathbf{x}_{N+1}) = \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t}_N = \sum_{n=1}^{N} a_n k(\mathbf{x}_n, \mathbf{x}_{N+1})$
  - a_n is the n-th component of $\mathbf{C}_N^{-1}\,\mathbf{t}_N$
  - for homogeneous kernel function, it is a radial basis expansion
  - in this case, kernels can be constructed indirectly (features)

Computational Complexity

- **Basis functions linear regression**
  - inversion of $M \times M$ matrix S_N, $O(M^3)$ time, once per training set
  - $M \times M$ matrix-vector multiplication, $O(M^2)$ time per prediction
- **Gaussian process regression**
  - inversion of $N \times N$ matrix C_N, $O(N^3)$ time, once per training set
  - $N \times N$ matrix-vector multiplication, $O(N^2)$ time per prediction
- **Comparison**
  - if M much smaller than N, then basis functions are preferred
  - GPs consider covariance functions expressible by infinite BFs
  - for large data sets, there are approximation schemes for GPs

Learning Hyperparameters

Learning

- **Learning hyperparameters**
  - parameters $\boldsymbol{\theta}$ of the kernel; determine the covariance function
  - instead of fixing their values, infer them from the data
- **Log likelihood learning**
  - maximization of the log likelihood $p(\mathbf{t}|\boldsymbol{\theta})$ for a point estimate
  - log likelihood may be non-convex and may have multiple maxima
  - log likelihood: $\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2}\ln|\mathbf{C}_N| - \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} - \frac{N}{2}\ln(2\pi)$
  - gradient: $\frac{\partial}{\partial\theta_i}\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2}\mathrm{Tr}\left(\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta_i}\right) + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{C}_N^{-1}\frac{\partial\mathbf{C}_N}{\partial\theta_i}\mathbf{C}_N^{-1}\mathbf{t}$
- **Bayesian setting**
  - introduce a prior over $\boldsymbol{\theta}$ and maximize the log posterior
  - evaluate marginals over $\boldsymbol{\theta}$ weighted by prior, likelihood (approx)

Automatic Relevance Determination

- **ARD**
  - introduce a separate parameter η_i for each input dimension i
  - optimize using maximum likelihood
  - infer the relative importance of different inputs from data
  - a small value of η_i implies insensitivity to input dimension i
- **ARD examples**
  - exponential-quadratic kernel in two dimensions

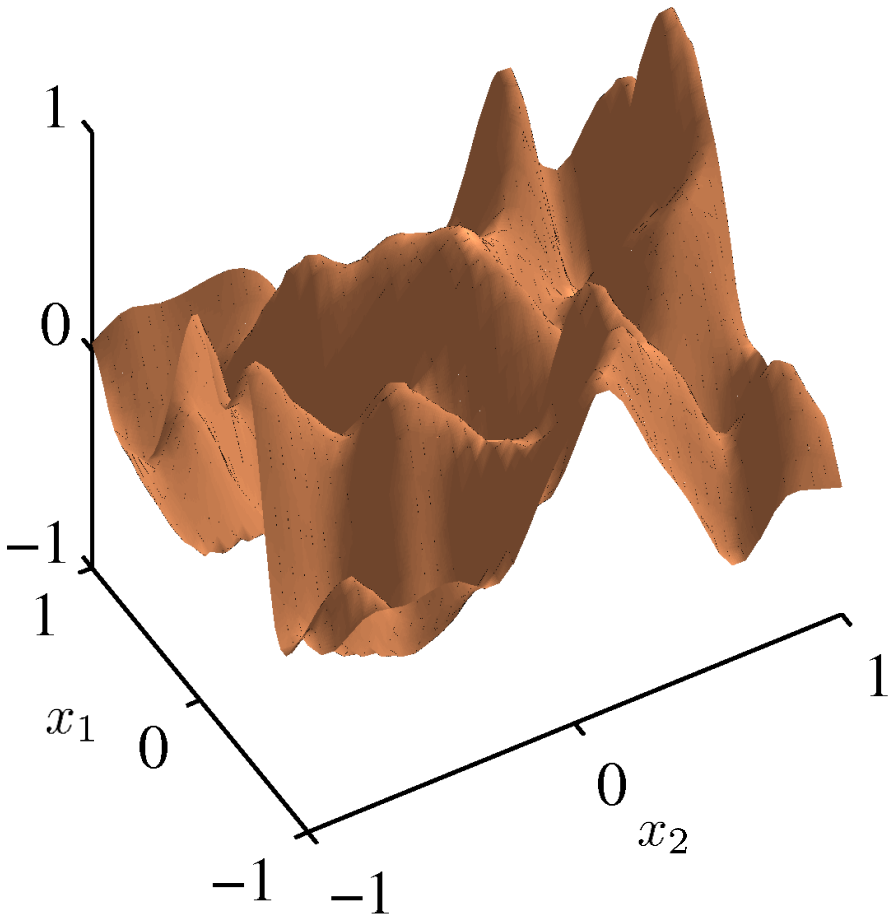
$$k(\mathbf{x}, \mathbf{x}') = \theta_0 \exp\left\{ -\frac{1}{2} \sum_{i=1}^{2} \eta_i (x_i - x_i')^2 \right\}$$

  - extended exponential-quadratic kernel in *D* dimensions

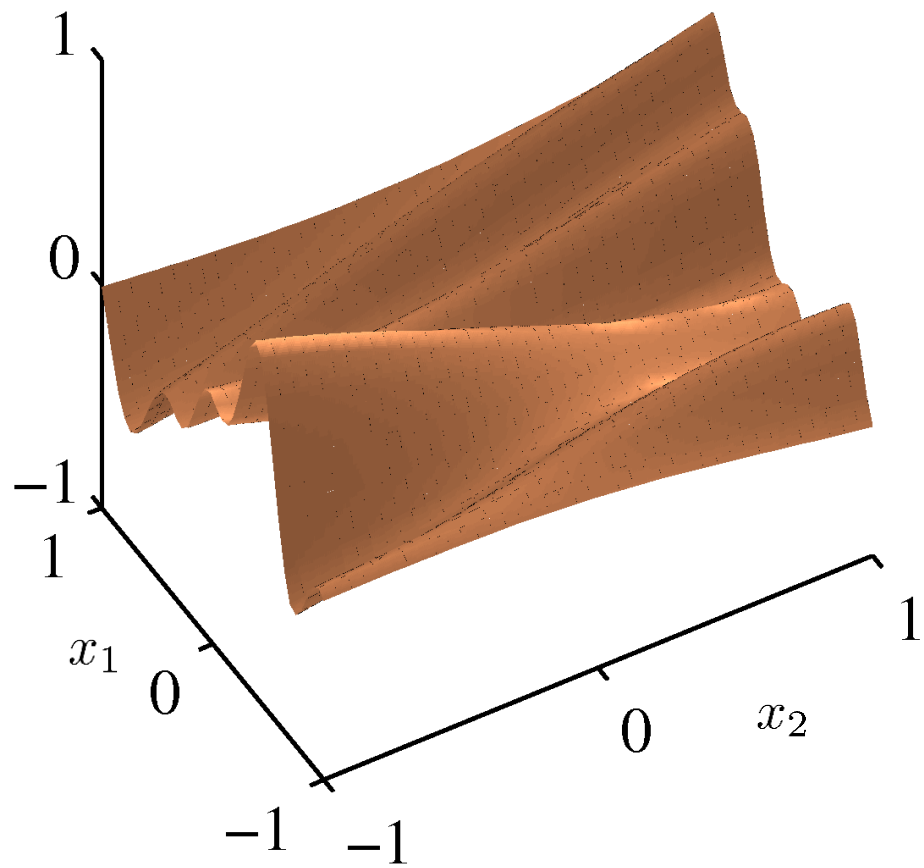
$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp\left\{ -\frac{1}{2} \sum_{i=1}^{D} \eta_i (x_{ni} - x_{mi})^2 \right\} + \theta_2 + \theta_3 \sum_{i=1}^{D} x_{ni} x_{mi}$$

ARD Impact Examples

$\eta_1 = \eta_2 = 1$

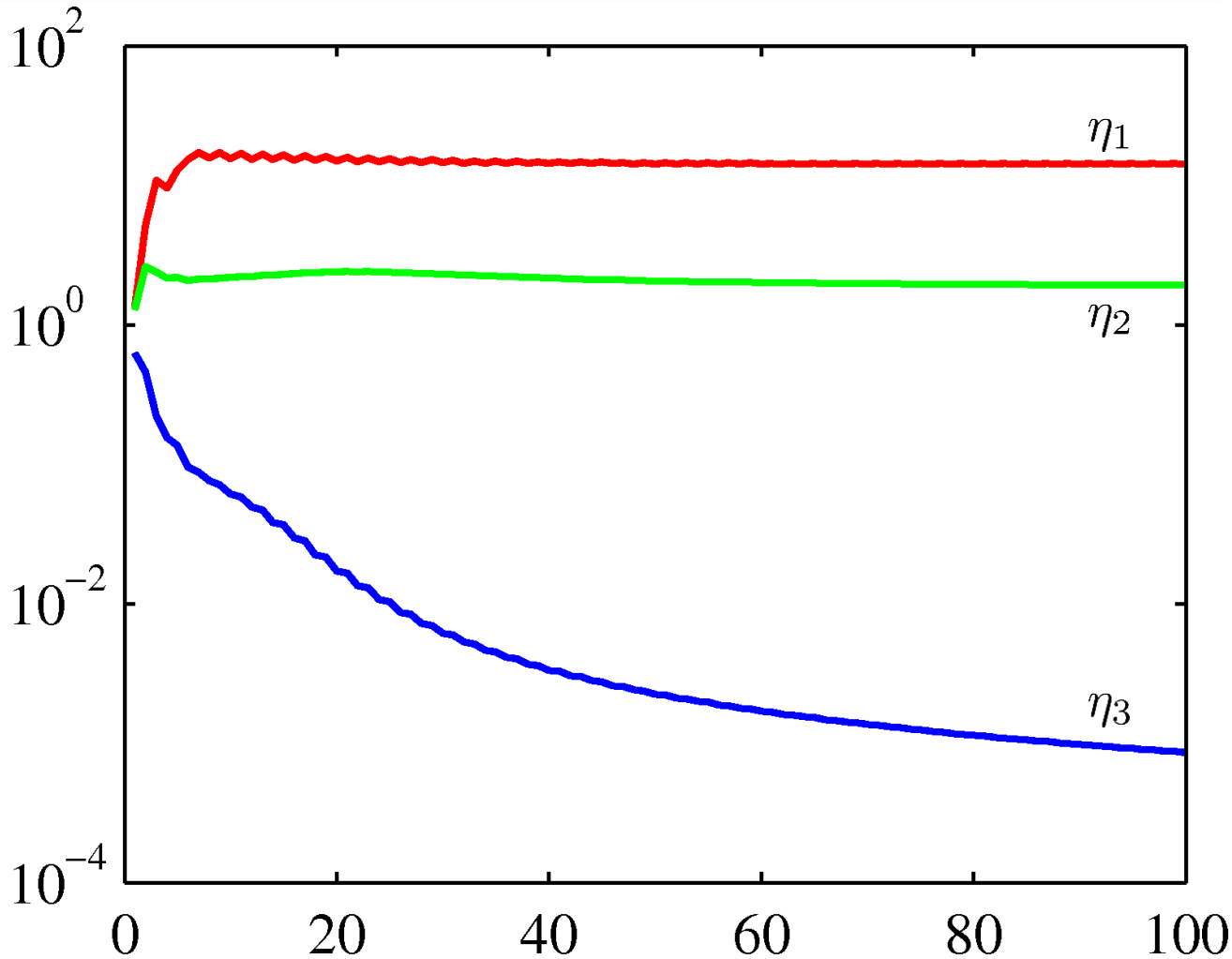


$\eta_1 = 1, \eta_2 = 0.01$



ARD Learning

- x_1 **(red)**
  - samples from the target function
- x_2 **(green)**
  - copy of x_1 with extra noise added
- x_3 **(blue)**
  - samples from an independent Gaussian



European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 14

Gaussian Processes for Classification

TUC ECE, Spring 2023

Today

- **Gaussian Process Classification**
  - using Laplace approximation
  - using Iteratively Reweighted Least-Squares

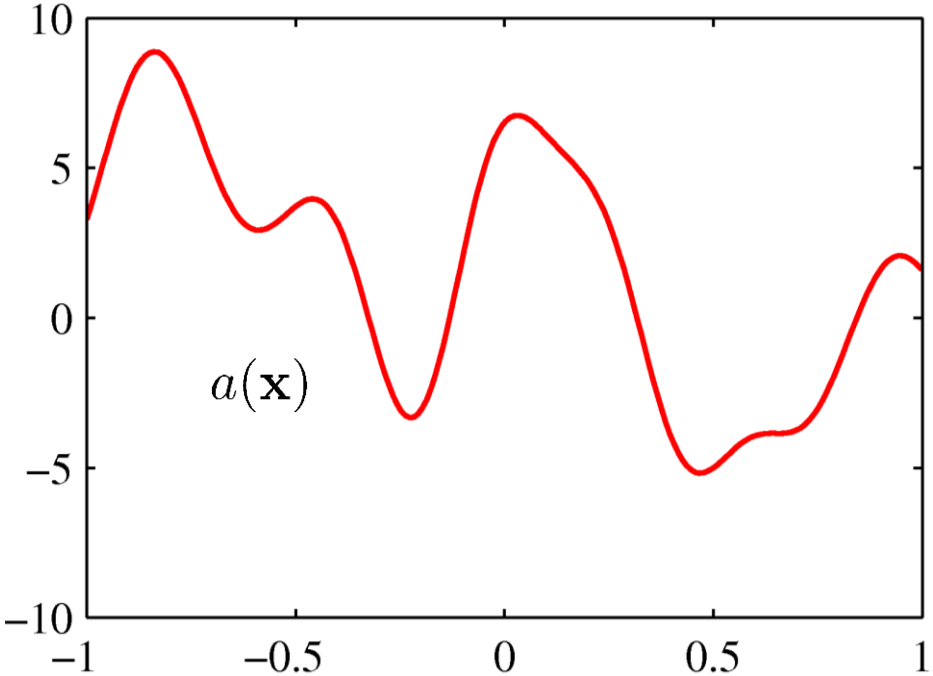
Gaussian Process Classification

Motivation

- **From regression to classification**
  - probabilistic classification: predict posterior class probabilities
  - Gaussian processes make predictions on the entire real axis
  - *idea*: use a non-linear activation function to limit output to (0,1)
- **Binary classification**
  - two-class classification problem with targets in {0,1}
  - define a Gaussian process whose (real) output is $a(\mathbf{x})$
  - transform (real) output to probability using a sigmoid, $y = \sigma(a)$
  - *result*: a non-Gaussian process over functions $y(\mathbf{x})$, $y \in (0,1)$
  - take Bernoulli probability distribution over the target variable t

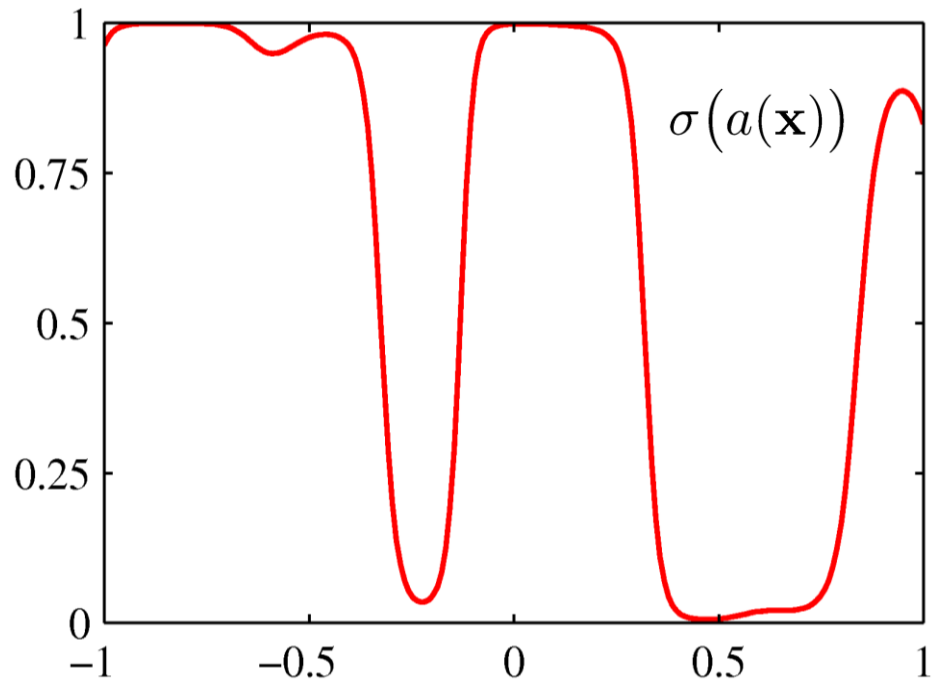
$$p(t|a) = \sigma(a)^t (1 - \sigma(a))^{1-t}$$

Output Transformation Example



output of Gaussian process

output of activation function



Formulation

- **Given**
  - inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N$ with observed targets $\mathbf{t}_N = (t_1, t_2, \ldots, t_N)^{\mathrm{T}}$
  - a single test input $\mathbf{x}_{N+1}$ whose target is denoted as t_{N+1}
- **Goal**
  - determine predictive distribution $p(t_{N+1} | \mathbf{t}_N)$
  - conditioned also on $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N, \mathbf{x}_{N+1}$, but dropped for clarity
- **Approach**
  - a Gaussian process prior over $\mathbf{a}_{N+1} = \Big(a(\mathbf{x}_1), a(\mathbf{x}_2), \ldots, a(\mathbf{x}_{N+1})\Big)^{\mathrm{T}}$
  - a non-Gaussian process prior over $\mathbf{t}_{N+1} = (t_1, t_2, \ldots, t_N, t_{N+1})^{\mathrm{T}}$
  - marginalization by conditioning on the training data $\mathbf{t}_N$

Gaussian Process Classification

- **Gaussian process**

$$p(\mathbf{a}_{N+1}) = \mathcal{N}(\mathbf{a}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1}) \qquad \mathbf{C}_{N+1} = \mathbf{K}_{N+1} + \nu \mathbf{I}_{N+1}$$

  - no need for output noise parameter β, since targets are "clean"
  - still, a small value ν is used to ensure that $\mathbf{C}$ is positive definite
  - can use any positive semidefinite kernel (parameterized by $\boldsymbol{\theta}$)
  - prediction $p(a_{N+1}|\mathbf{a}_N) = \mathcal{N}\left(a_{N+1} \mid \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{a}_N,\ c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}\right)$

$$\mathbf{k} = \{k(\mathbf{x}_n, \mathbf{x}_{N+1})\}, n = 1, 2, \ldots, N$$

$$c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \nu$$

- **Predictive distribution**
  - suffices to predict $p(t_{N+1}=1|\mathbf{t}_N)$; complementary $p(t_{N+1}=0|\mathbf{t}_N)$

$$p(t_{N+1} = 1|\mathbf{t}_N) = \int p(t_{N+1} = 1|a_{N+1})p(a_{N+1}|\mathbf{t}_N)\,\mathrm{d}a_{N+1}$$

$$p(t_{N+1} = 1|a_{N+1}) = \sigma(a_{N+1})$$

Approximations …

- the integral of the predictive distribution is intractable!

- **Approximating the integral**
  - 1) Monte Carlo sampling methods
  - 2) approximation for convolution of sigmoid with Gaussian
  - the latter requires a Gaussian approximation to $p(a_{N+1}|\mathbf{t}_N)$
- **Approximating the posterior**
  - 1) variational inference using variational bound on logistic
  - 2) expectation propagation due to unimodality
  - 3) Laplace approximation (our lovely choice!)

Laplace Approximation

- **Using Bayes theorem**

$$
\begin{aligned}
p(a_{N+1}|\mathbf{t}_N) &= \int p(a_{N+1}, \mathbf{a}_N|\mathbf{t}_N)\,\mathrm{d}\mathbf{a}_N \\
&= \frac{1}{p(\mathbf{t}_N)} \int p(a_{N+1}, \mathbf{a}_N) p(\mathbf{t}_N|a_{N+1}, \mathbf{a}_N)\,\mathrm{d}\mathbf{a}_N \\
&= \frac{1}{p(\mathbf{t}_N)} \int p(a_{N+1}|\mathbf{a}_N) p(\mathbf{a}_N) p(\mathbf{t}_N|\mathbf{a}_N)\,\mathrm{d}\mathbf{a}_N \\
&= \int p(a_{N+1}|\mathbf{a}_N) p(\mathbf{a}_N|\mathbf{t}_N)\,\mathrm{d}\mathbf{a}_N
\end{aligned}
$$

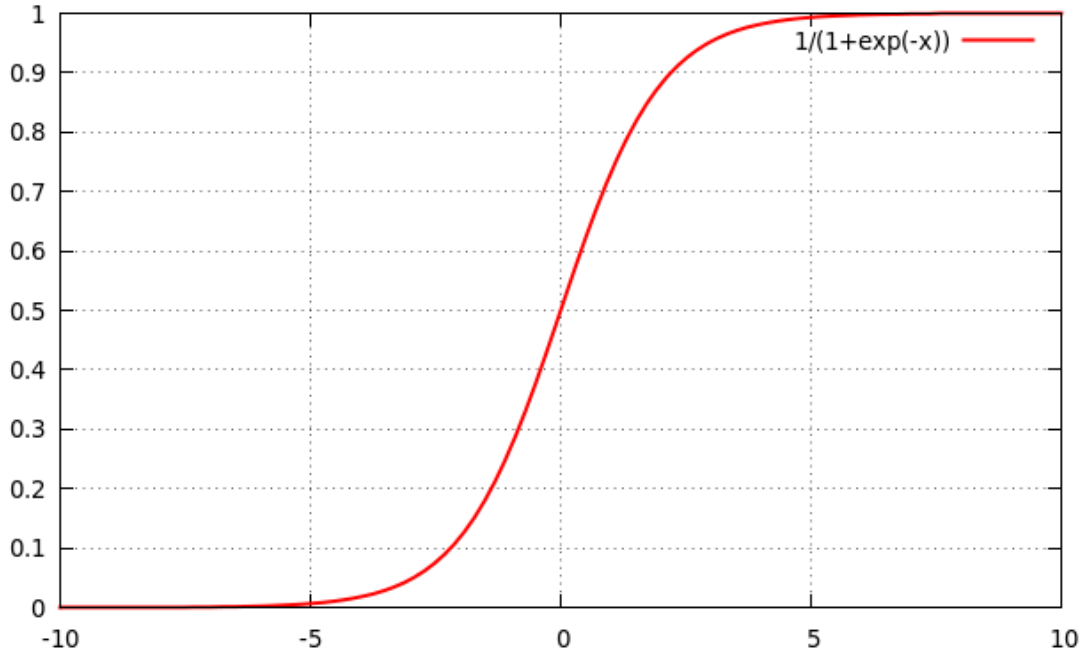
$$p(\mathbf{t}_N|a_{N+1}, \mathbf{a}_N) = p(\mathbf{t}_N|\mathbf{a}_N)$$

  - $p(a_{N+1}|\mathbf{a}_N)$ is Gaussian $p(a_{N+1}|\mathbf{a}_N) = \mathcal{N}\left(a_{N+1} \mid \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{a}_N,\ c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}\right)$
  - Laplace approximation for the posterior $p(\mathbf{a}_N|\mathbf{t}_N)$
  - then, use known result for convolution of two Gaussians

Recall: Logistic Sigmoid Function

- **Definition**
  - squashing function
  - maps the real axis into a finite interval

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$



- **Symmetry**

$$\sigma(-a) = 1 - \sigma(a)$$

- **Inverse**
  - logit function (log odds)

$$a = \ln\left(\frac{\sigma}{1-\sigma}\right)$$

- **Derivative**

$$\sigma'(a) = \sigma(a)\big(1-\sigma(a)\big)$$

Recall: *M*-D Laplace Approximation

- **Density**
  - density over *M*-dimensional variable $\mathbf{z}$: $p(\mathbf{z}) = f(\mathbf{z})/Z$
- **Taylor expansion**

$$\ln f(\mathbf{z}) \simeq \ln f(\mathbf{z}_0) - \frac{1}{2}(\mathbf{z}-\mathbf{z}_0)^{\mathrm{T}}\mathbf{A}(\mathbf{z}-\mathbf{z}_0) \qquad \mathbf{A} = -\nabla\nabla \ln f(\mathbf{z})|_{\mathbf{z}=\mathbf{z}_0}$$

- **Gaussian approximation**

$$f(\mathbf{z}) \simeq f(\mathbf{z}_0)\exp\left\{-\frac{1}{2}(\mathbf{z}-\mathbf{z}_0)^{\mathrm{T}}\mathbf{A}(\mathbf{z}-\mathbf{z}_0)\right\}$$

$$q(\mathbf{z}) = \frac{|\mathbf{A}|^{1/2}}{(2\pi)^{M/2}}\exp\left\{-\frac{1}{2}(\mathbf{z}-\mathbf{z}_0)^{\mathrm{T}}\mathbf{A}(\mathbf{z}-\mathbf{z}_0)\right\} = \mathcal{N}(\mathbf{z}|\mathbf{z}_0, \mathbf{A}^{-1})$$

  - $\mathbf{A}$ must be positive definite for $\mathbf{z}_0$ to be a mode

Laplace Approximation

- the prior $p(\mathbf{a}_N)$ is zero-mean Gaussian with covariance $\mathbf{C}_N$
- likelihood of targets

$$p(\mathbf{t}_N|\mathbf{a}_N) = \prod_{n=1}^{N} \sigma(a_n)^{t_n}(1-\sigma(a_n))^{1-t_n} = \prod_{n=1}^{N} e^{a_n t_n}\sigma(-a_n)$$

- Taylor expansion of the logarithm of $p(\mathbf{a}_N|\mathbf{t}_N) \propto p(\mathbf{t}_N|\mathbf{a}_N)p(\mathbf{a}_N)$

$$\begin{aligned}\Psi(\mathbf{a}_N) &= \ln p(\mathbf{a}_N) + \ln p(\mathbf{t}_N|\mathbf{a}_N) \\ &= -\frac{1}{2}\mathbf{a}_N^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{a}_N - \frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln|\mathbf{C}_N| + \mathbf{t}_N^{\mathrm{T}}\mathbf{a}_N - \sum_{n=1}^{N}\ln(1+e^{a_n}) + \text{const.}\end{aligned}$$

- first and second derivatives

$$\nabla\Psi(\mathbf{a}_N) = \mathbf{t}_N - \boldsymbol{\sigma}_N - \mathbf{C}_N^{-1}\mathbf{a}_N \qquad \nabla\nabla\Psi(\mathbf{a}_N) = -\mathbf{W}_N - \mathbf{C}_N^{-1}$$

$$\boldsymbol{\sigma}_N = \Big(\sigma(a_1), \sigma(a_2), \ldots, \sigma(a_N)\Big)^{\mathrm{T}} \qquad \mathbf{W}_N = \mathrm{diag}\Big\{\sigma(a_n)\big(1-\sigma(a_n)\big)\Big\}$$

- Hessian $\mathbf{A} = -\nabla\nabla\Psi(\mathbf{a}_N)$ is positive definite (sum of positive definite)
- the posterior $p(\mathbf{a}_N|\mathbf{t}_N)$ is log convex and thus has a single mode

Iteratively Reweighted Least-Squares

- the mode (global maximum) cannot be found by setting the gradient to zero, due to nonlinear dependence of $\boldsymbol{\sigma}_N$ on $\mathbf{a}_N$
- thus, iterative scheme based on Newton-Raphson (and IRLS)

$$\mathbf{a}_N^{\text{new}} = \mathbf{C}_N(\mathbf{I} + \mathbf{W}_N\mathbf{C}_N)^{-1}\left\{\mathbf{t}_N - \boldsymbol{\sigma}_N + \mathbf{W}_N\mathbf{a}_N\right\}$$

- the iteration converges to $\mathbf{a}^*_N$, where the gradient vanishes

$$\nabla\Psi(\mathbf{a}_N) = \mathbf{t}_N - \boldsymbol{\sigma}_N - \mathbf{C}_N^{-1}\mathbf{a}_N = 0 \implies \mathbf{a}_N^\star = \mathbf{C}_N(\mathbf{t}_N - \boldsymbol{\sigma}_N)$$

- now, we can evaluate the Hessian

$$\mathbf{H} = -\nabla\nabla\boldsymbol{\Psi}(\mathbf{a}_N^*) = \mathbf{W}_N^* + \mathbf{C}_N^{-1} \qquad \mathbf{W}_N^* = \operatorname{diag}\Big\{\sigma(a_n^*)\big(1 - \sigma(a_n^*)\big)\Big\}$$

- to obtain the final Gaussian approximation to $p(\mathbf{a}_N|\mathbf{t}_N)$

$$p(\mathbf{a}_N|\mathbf{t}_N) \approx q(\mathbf{a}_N) = \mathcal{N}(\mathbf{a}_N|\mathbf{a}_N^*, \mathbf{H}^{-1})$$

Back to our Approximation

- final Gaussian approximation to $p(a_{N+1}|\mathbf{t}_N)$

$$p(a_{N+1}|\mathbf{t}_N) = \int p(a_{N+1}|\mathbf{a}_N)p(\mathbf{a}_N|\mathbf{t}_N)\,\mathrm{d}\mathbf{a}_N$$

$$p(a_{N+1}|\mathbf{a}_N) = \mathcal{N}\Big(a_{N+1} \mid \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{a}_N,\ c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}\Big)$$

$$p(\mathbf{a}_N|\mathbf{t}_N) \simeq q(\mathbf{a}_N) = \mathcal{N}(\mathbf{a}_N|\mathbf{a}_N^*, \mathbf{H}^{-1})$$

$$\mathbf{a}_N^\star = \mathbf{C}_N(\mathbf{t}_N - \boldsymbol{\sigma}_N) \quad \mathbf{H} = -\nabla\nabla\boldsymbol{\Psi}(\mathbf{a}_N^*) = \mathbf{W}_N^* + \mathbf{C}_N^{-1}$$

$$\mathbf{W}_N^* = \mathrm{diag}\Big\{\sigma(a_n^*)\big(1 - \sigma(a_n^*)\big)\Big\}$$

- use known result for convolution of two Gaussians (next slide)

$$\begin{aligned}\mathbb{E}[a_{N+1}|\mathbf{t}_N] &= \mathbf{k}^{\mathrm{T}}(\mathbf{t}_N - \boldsymbol{\sigma}_N)\\ \mathrm{var}[a_{N+1}|\mathbf{t}_N] &= c - \mathbf{k}^{\mathrm{T}}(\mathbf{W}_N^{-1} + \mathbf{C}_N)^{-1}\mathbf{k}\end{aligned}$$

Back to our Approximation (Proof)

- **Reminder: Bayes' Theorem for Gaussians**

$$p(\mathbf{x},\mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$$

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Lambda}^{-1}\right) \\ p(\mathbf{y}|\mathbf{x}) &= \mathcal{N}\left(\mathbf{y}|\mathbf{A}\mathbf{x}+\mathbf{b},\mathbf{L}^{-1}\right) \end{aligned}$$

$$\boxed{\begin{aligned} p(\mathbf{y}) &= \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu}+\mathbf{b},\mathbf{L}^{-1}+\mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \end{aligned}}$$

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y}-\mathbf{b})+\boldsymbol{\Lambda}\boldsymbol{\mu}\},\boldsymbol{\Sigma}) \\ \boldsymbol{\Sigma} &= (\boldsymbol{\Lambda}+\mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1} \end{aligned}$$

$$p(\mathbf{a}_N|\mathbf{t}_N) \simeq \mathcal{N}(\mathbf{a}_N|\mathbf{C}_N(\mathbf{t}_N-\boldsymbol{\sigma}_N),(\mathbf{W}_N^*+\mathbf{C}_N^{-1})^{-1})$$

$$p(a_{N+1}|\mathbf{a}_N) = \mathcal{N}\Big(a_{N+1} \mid \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{a}_N,\ c-\mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}\Big)$$

$$\begin{aligned} p(a_{N+1}|\mathbf{t}_N) &= \int p(a_{N+1}|\mathbf{a}_N)p(\mathbf{a}_N|\mathbf{t}_N)\,\mathrm{d}\mathbf{a}_N \\ &= \mathcal{N}\Big(\mathbf{y} \,\Big|\, \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{C}_N(\mathbf{t}_N-\boldsymbol{\sigma}_N),\ c-\mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}+\mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}(\mathbf{W}_N^*+\mathbf{C}_N^{-1})^{-1}\mathbf{C}_N^{-1}\mathbf{k}\Big) \\ &= \mathcal{N}\Big(\mathbf{y} \,\Big|\, \mathbf{k}^{\mathrm{T}}(\mathbf{t}_N-\boldsymbol{\sigma}_N),\ c-\mathbf{k}^{\mathrm{T}}(\mathbf{C}_N^{-1}-\mathbf{C}_N^{-1}(\mathbf{W}_N^*+\mathbf{C}_N^{-1})^{-1}\mathbf{C}_N^{-1})\mathbf{k}\Big) \\ &= \mathcal{N}\Big(\mathbf{y} \,\Big|\, \mathbf{k}^{\mathrm{T}}(\mathbf{t}_N-\boldsymbol{\sigma}_N),\ c-\mathbf{k}^{\mathrm{T}}\big(\mathbf{C}_N+(\mathbf{W}_N^*)^{-1}\big)^{-1}\mathbf{k}\Big) \end{aligned}$$

$$(\mathbf{A}+\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}+\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}$$

Derive the Predictive Distribution

- **Reminder: Convolution of Sigmoid with Gaussian**

$$\int \sigma(a)\mathcal{N}(a|\mu,\sigma^2)\,\mathrm{d}a \simeq \sigma\left(\kappa(\sigma^2)\mu\right) \qquad \kappa(\sigma^2) = (1+\pi\sigma^2/8)^{-1/2}$$

- **Predictive Distribution**

$$p(t_{N+1}=1|\mathbf{t}_N) = \int p(t_{N+1}=1|a_{N+1})p(a_{N+1}|\mathbf{t}_N)\,\mathrm{d}a_{N+1}$$

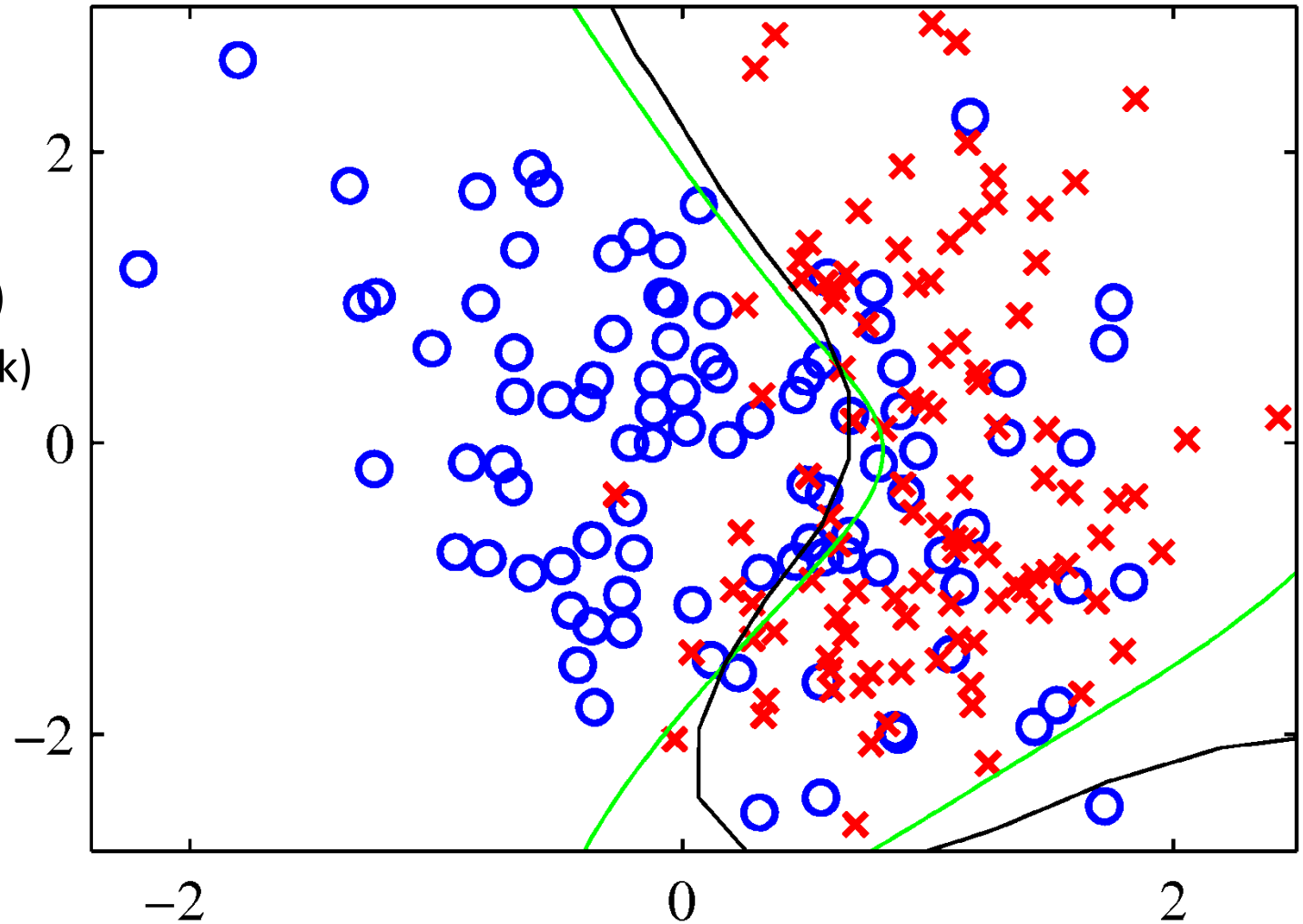
$$p(t_{N+1}=1|a_{N+1}) = \sigma(a_{N+1})$$

$$p(a_{N+1}|\mathbf{t}_N) = \mathcal{N}\Big(\mathbf{y}\;\Big|\;\mathbf{k}^{\mathrm{T}}(\mathbf{t}_N-\boldsymbol{\sigma}_N),\;c-\mathbf{k}^{\mathrm{T}}\big(\mathbf{C}_N+(\mathbf{W}_N^*)^{-1}\big)^{-1}\mathbf{k}\Big)$$

$$\begin{aligned} p(t_{N+1}=1|\mathbf{t}_N) &\simeq \sigma\Big(\kappa(\sigma^2)\mathbf{k}^{\mathrm{T}}(\mathbf{t}_N-\boldsymbol{\sigma}_N)\Big) \\ \kappa(\sigma^2) &= \sqrt{1+0.125\pi\sigma^2} \\ \sigma^2 &= c-\mathbf{k}^{\mathrm{T}}\big(\mathbf{C}_N+(\mathbf{W}_N^*)^{-1}\big)^{-1}\mathbf{k} \end{aligned}$$

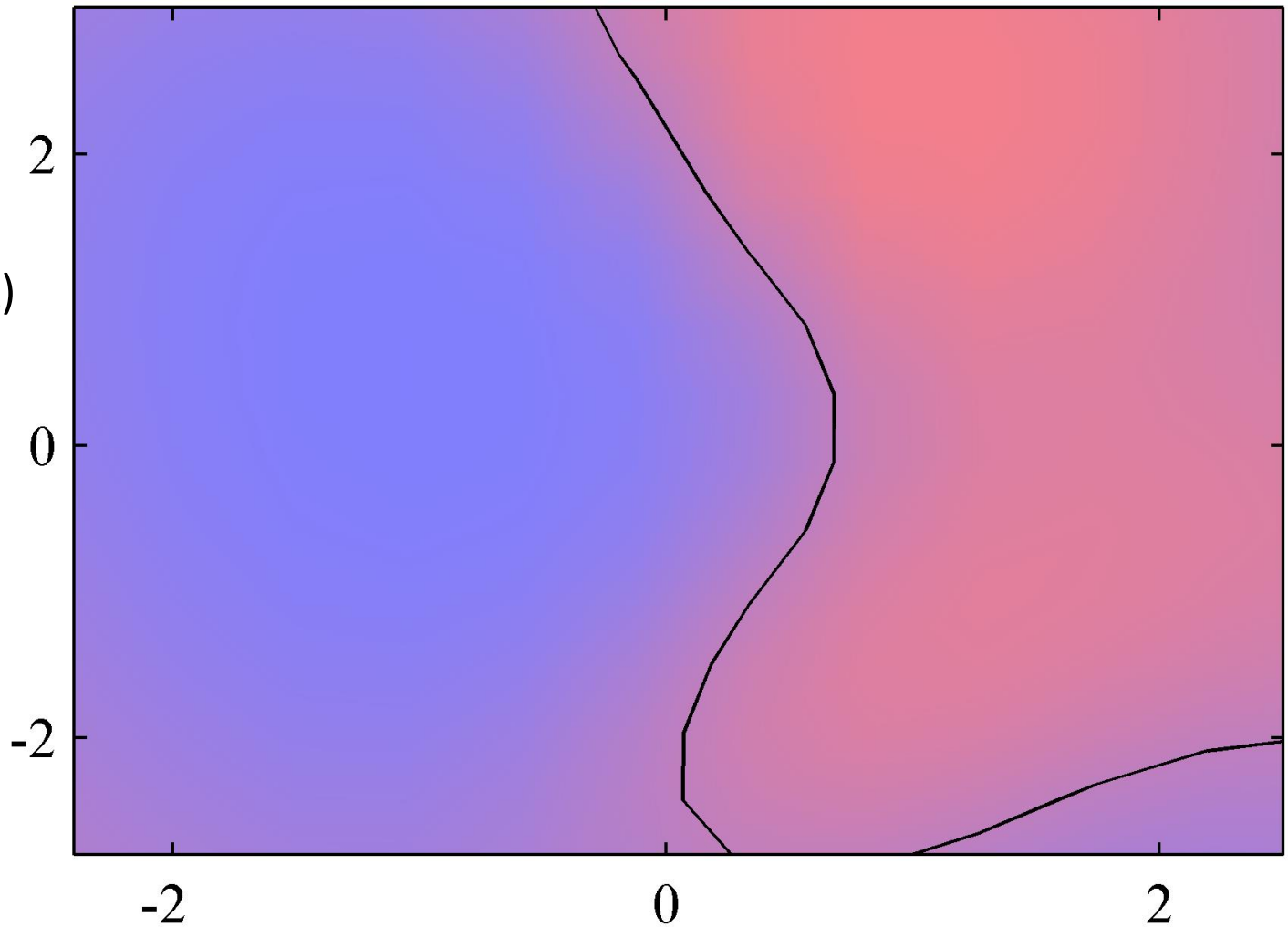
GP Binary Classification Example

- **Legend**
  - class 0 (red)
  - class 1 (blue)
  - optimal (green)
  - boundary (black)



GP Binary Classification Probabilities

- **Legend**
  - class 0 (red)
  - class 1 (blue)
  - boundary (black)



European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 15

Maximum Margin Machines

TUC ECE, Spring 2023

Today

- **Sparse Kernel Machines**
- **Maximum Margin Criterion**
- **Lagrange Multipliers**

Sparse Kernel Methods

- **Kernel Methods**
  - non-parametric methods, rely on kernel, no parameters
  - limitation: the kernel function must be evaluated over all pairs
  - predictions are computationally expensive
  - *idea*: can we evaluate the kernel function only over a subset?
- **Sparse Kernel Machines**
  - select a subset of training points that determine the outcome
  - use only the selected subset for prediction
  - Support Vector Machines (SVMs) [discriminant functions]
  - Relevance Vector Machines (RVMs) [discriminant models]

Maximum Margin Criterion

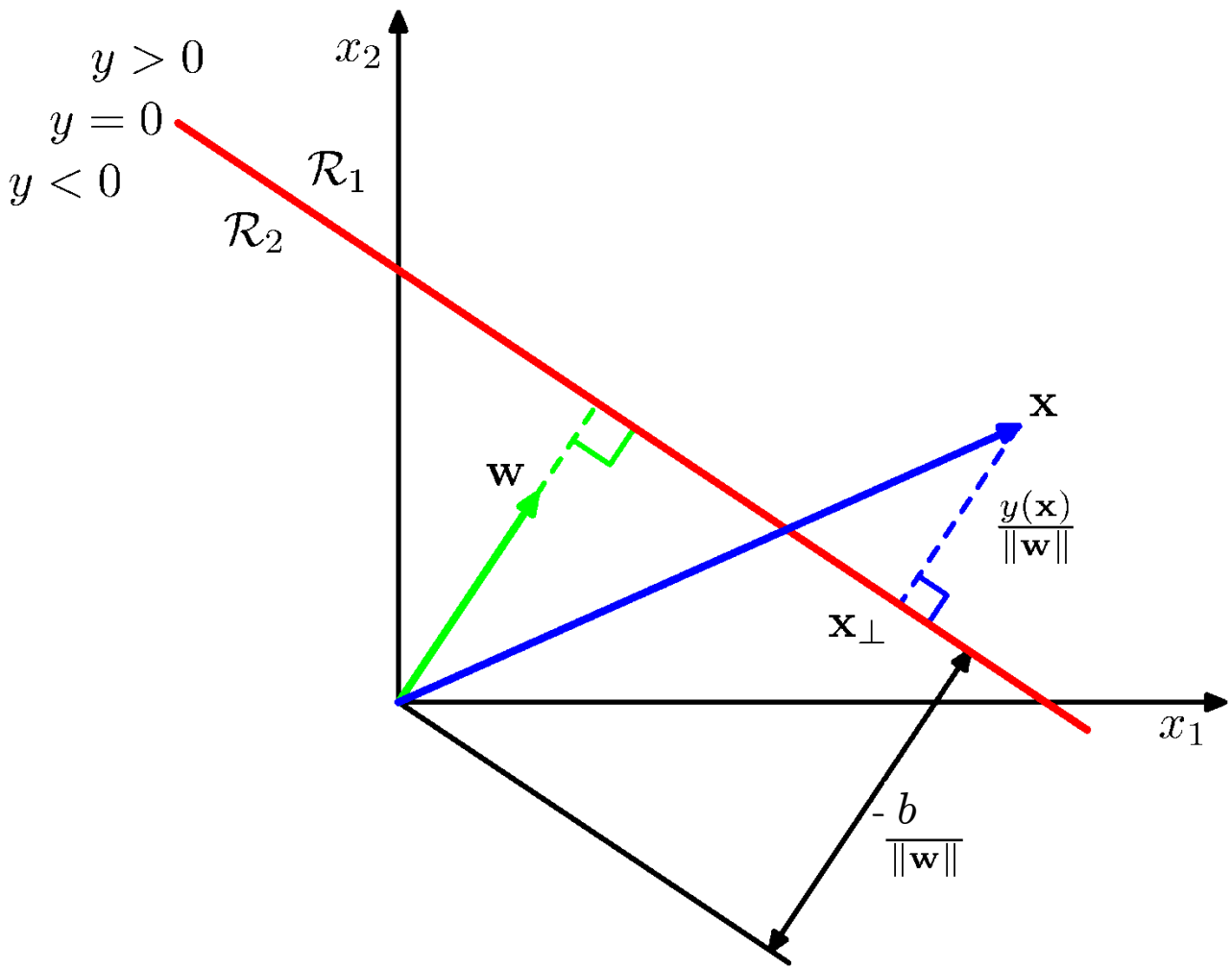
Recall: Linear Discriminants

- **Linear discriminant function**

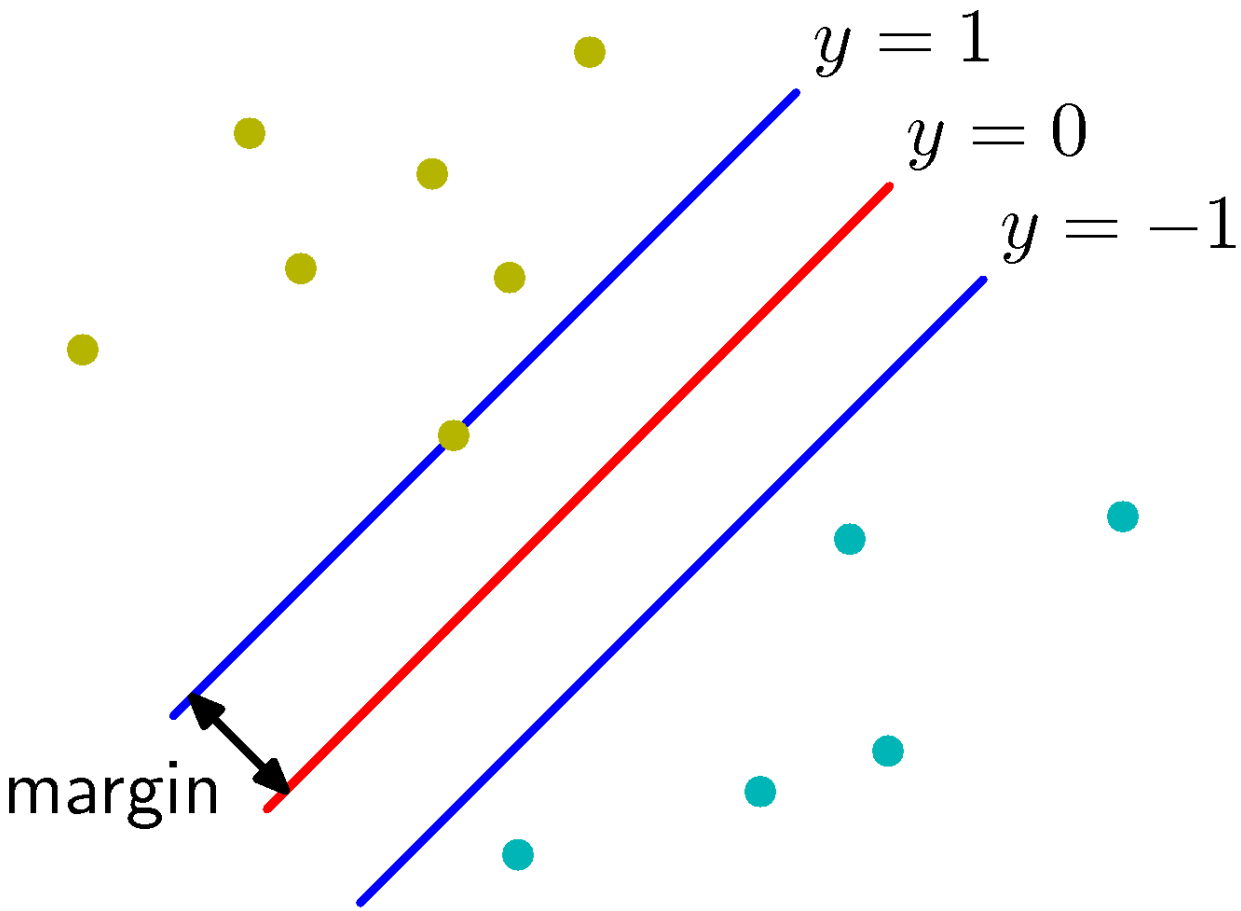
$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b$$

  - ϕ is a set of features
  - $\mathbf{w}$ is the *weight* (parameter) vector
  - b is the *bias* parameter (its negative is the *threshold*)
- **Binary classification**
  - data: inputs $\mathbf{x}_1$, $\mathbf{x}_2$, … $\mathbf{x}_N$ with targets t_1, t_2, … t_N
  - target coding: for class C_1: +1, for class C_2: -1
- **Decision surface** $y(\mathbf{x}) = 0 \Leftrightarrow \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b = 0$
  - $\mathbf{w}$ defines orientation and b defines location
  - decision: sign of $y(\mathbf{x})$; if separable, for all points: $t_n y(\mathbf{x}_n) > 0$

Linear Discriminant Geometry

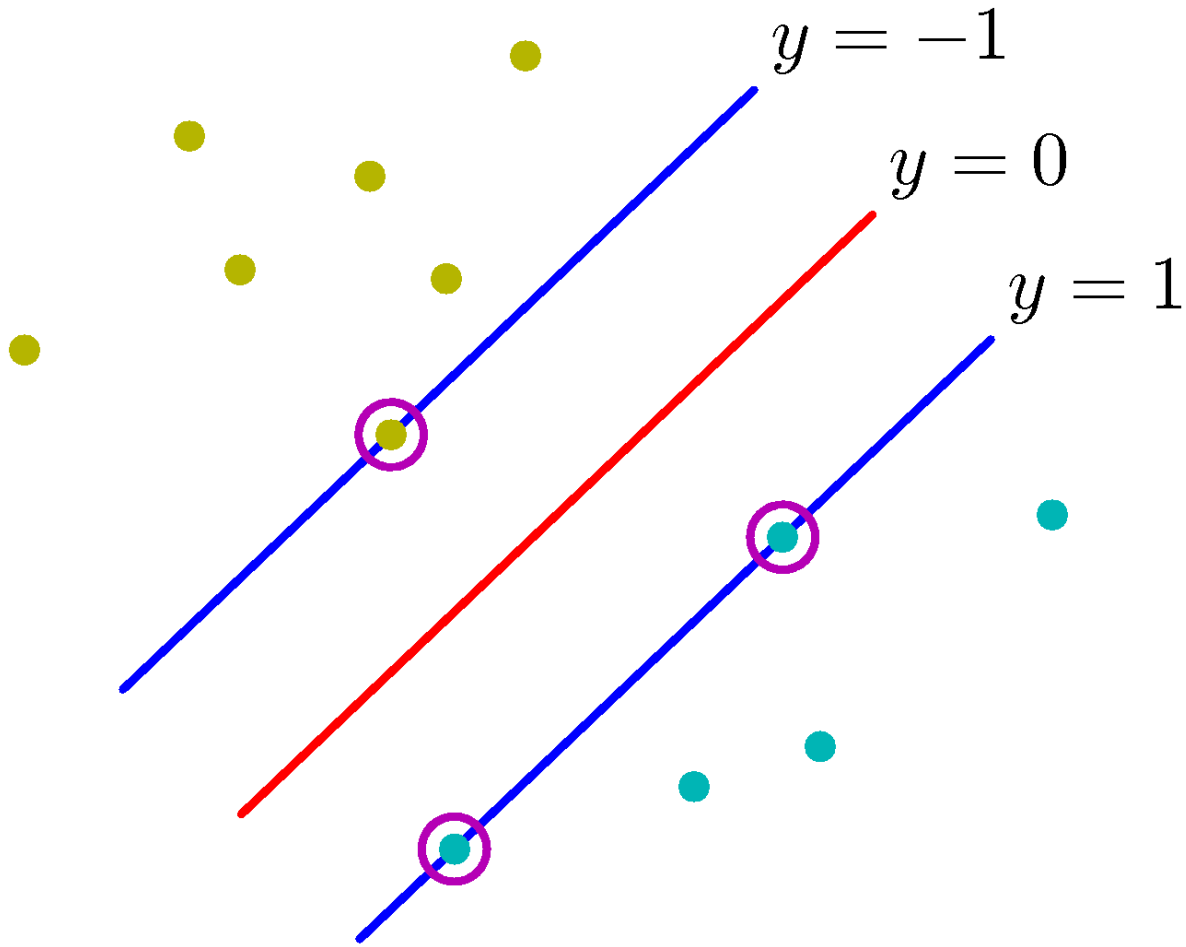


Margin Geometry



margin: perpendicular distance between decision boundary and closest point

Maximum Margin Geometry



maximum margin: determined by a subset of points (support vectors)

Max Margin and Support Vectors

- **(Unsigned) distance to margin**

$$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$$

- **Maximum margin solution**

$$\arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n \left[t_n \left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b \right) \right] \right\}$$

  - hard to solve problem
  - points defining the min in the solution are the support vectors

Canonical Representation

- **Scale invariance** $\mathbf{w} \rightarrow \kappa \mathbf{w}$ and $b \rightarrow \kappa b$
  - does not affect distances to margin and the decision boundary
  - therefore, there exist multiple solutions for $\mathbf{w}$ and b
- **Canonical representation**
  - for points closest to the margin (active)

$$t_n \left(\mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) + b \right) = 1$$

  - for all points, including those away (inactive)

$$t_n \left(\mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) + b \right) \geqslant 1$$

- **Constrained optimization**
  - maximize $\|\mathbf{w}\|^{-1}$ (or minimize $\|\mathbf{w}\|^2$)
  - quadratic programming problem

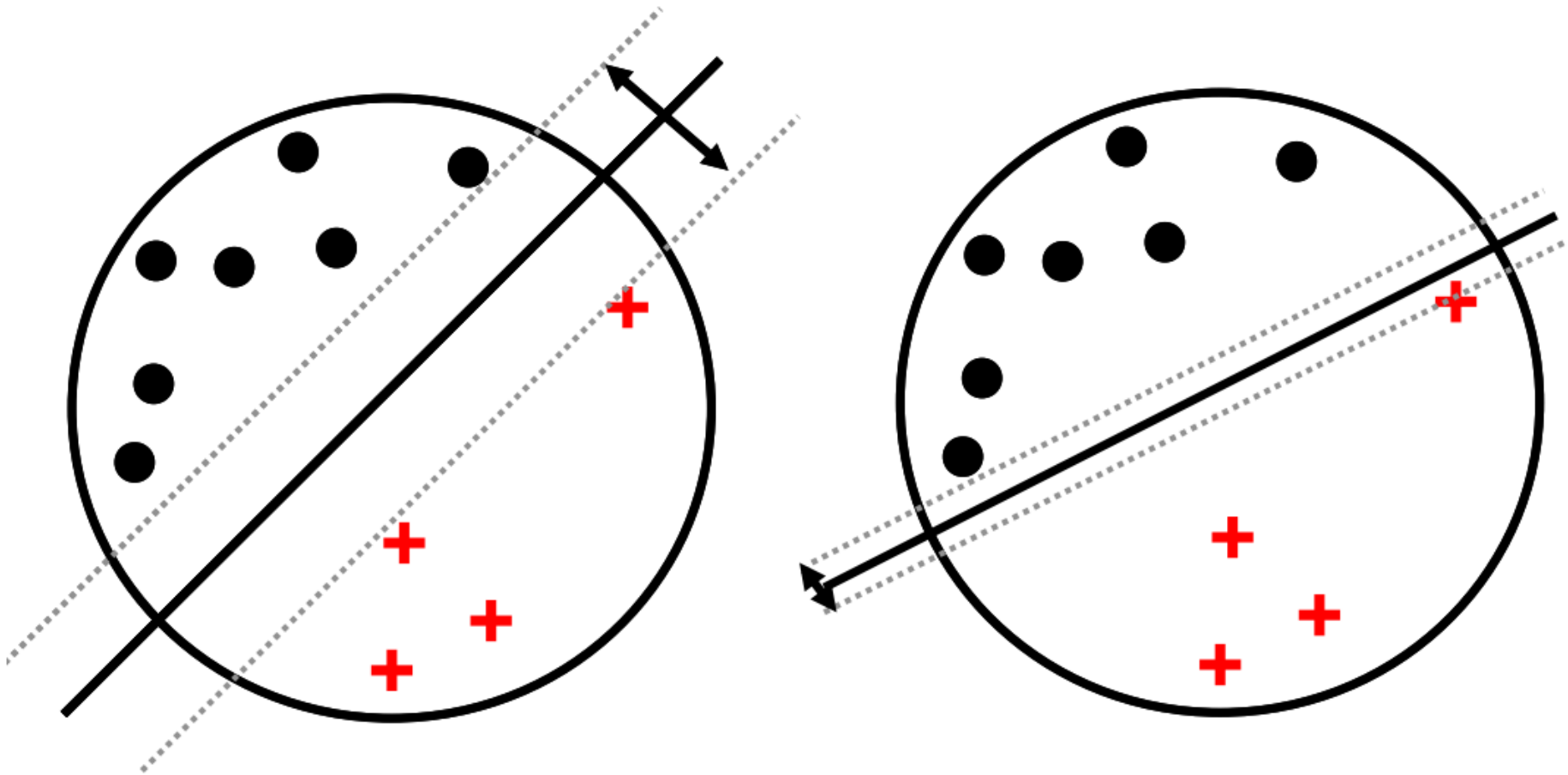
$$\arg\min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2$$

subject to

$$t_n \left(\mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) + b \right) \geqslant 1$$

$$n = 1, \ldots, N$$

Maximum Margin Example



large margin (left, 3 active points) and **small margin** (right, 2 active points)

Lagrange Multipliers

Constrained Optimization

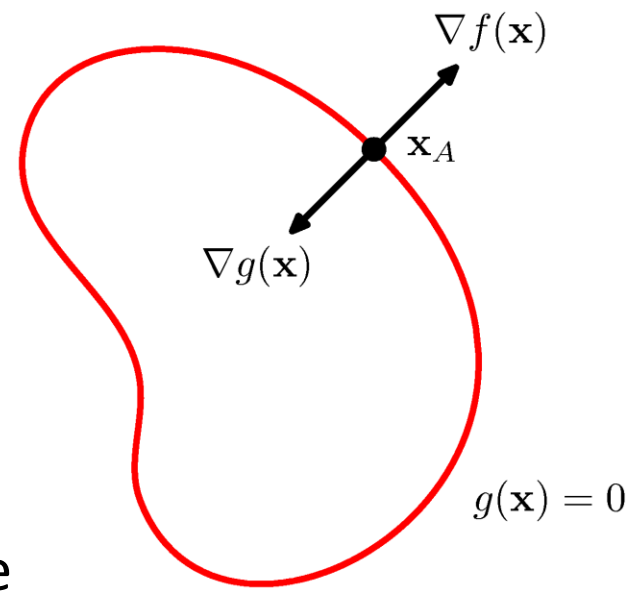
- **Problem**
  - find the stationary points of a function of several variables …
  - … subject to one or more constraints on the variables
- **Example**
  - find the maximum of $f(x_1,x_2)$ subject to constraint $g(x_1,x_2) = 0$
- **Approach**
  - solve $g(x_1,x_2) = 0$ to express x_2 as a function of x_1: $x_2 = h(x_1)$
  - substitute x_2 into $f(x_1,x_2)$ to obtain $f(x_1, h(x_1))$
  - differentiate, set to zero, and solve for x_1 to obtain x_1^*
  - obtain $x_2^* = h(x_1^*)$
- **Properties**
  - analytical solution may be difficult; also, breaks the symmetry

Optimization Geometry - Equality

- D-dimensional optimization
  - maximize $f(\mathbf{x})$ subject to $g(\mathbf{x}) = 0$
  - $g(\mathbf{x})$=0: (D-1)-dimensional surface on $\mathbf{x}$
- Property
  - $\nabla g(\mathbf{x})$ is orthogonal to the surface
- Proof
  - Taylor expansion around $\mathbf{x}$ on the surface

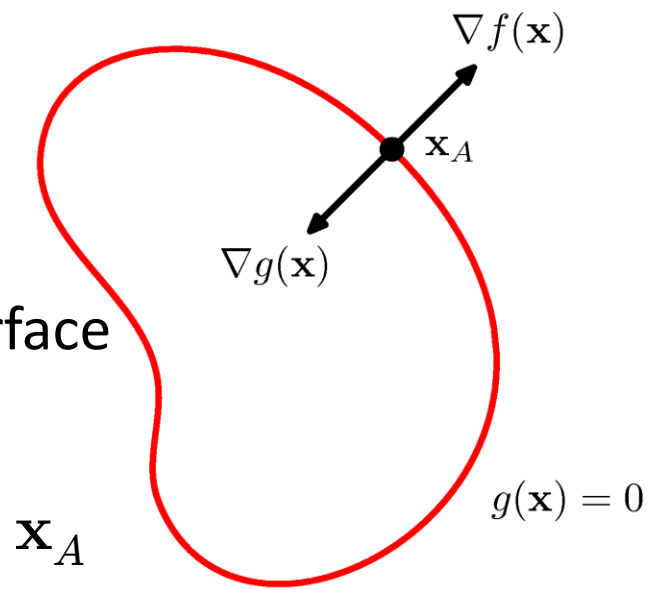
$$g(\mathbf{x}+\boldsymbol{\epsilon}) \simeq g(\mathbf{x}) + \boldsymbol{\epsilon}^{\mathrm{T}} \nabla g(\mathbf{x})$$

  - both $\mathbf{x}$ and $\mathbf{x}+\boldsymbol{\epsilon}$ lie on the surface: $g(\mathbf{x}) = g(\mathbf{x}+\boldsymbol{\epsilon}) = 0$
  - therefore, $\boldsymbol{\epsilon}^{\mathrm{T}} \nabla g(\mathbf{x}) \simeq 0$ and when $\|\boldsymbol{\epsilon}\| \to 0$: $\boldsymbol{\epsilon}^{\mathrm{T}} \nabla g(\mathbf{x}) = 0$
  - since $\boldsymbol{\epsilon}$ is parallel to the surface, ∇g is normal to the surface



Optimization Geometry - Equality

- **Property**
  - $\nabla f(\mathbf{x})$ is also orthogonal to the surface
  - … at the point $\mathbf{x}_A$ of the surface
  - … where $f(\mathbf{x})$ is maximized along the surface



- **Proof**
  - if it was not orthogonal to the surface at $\mathbf{x}_A$
  - … then we could increase the value of $f(\mathbf{x})$
  - … by moving a short distance along the surface
- **Property**
  - vectors $\nabla f(\mathbf{x})$ and $\nabla g(\mathbf{x})$ are either parallel or anti-parallel at $\mathbf{x}_A$

$$\nabla f(\mathbf{x}_A) + \lambda \nabla g(\mathbf{x}_A) = 0, \quad \lambda \neq 0$$

Lagrangian Optimization - Equality

- **Lagrangian Function**

$$L(\mathbf{x}, \lambda) \equiv f(\mathbf{x}) + \lambda g(\mathbf{x})$$

  - $\lambda \neq 0$ is known as the *Lagrange* (or *undetermined*) *multiplier*
  - stationary points of $L(\mathbf{x},\lambda)$ also solve the constrained problem

$$\nabla_{\mathbf{x}} L(\mathbf{x}, \lambda) = 0 \iff \nabla f(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) = 0$$

$$\partial L(\mathbf{x}, \lambda)/\partial \lambda = 0 \iff g(\mathbf{x}) = 0$$

- **Lagrangian Optimization**
  - to maximize $f(\mathbf{x})$ subject to $g(\mathbf{x})$ = 0, equivalently …
  - … find the stationary point of the Lagrangian function $L(\mathbf{x},\lambda)$
  - yields D+1 equations that determine both $\mathbf{x}_A$ and λ

Lagrangian Optimization Example

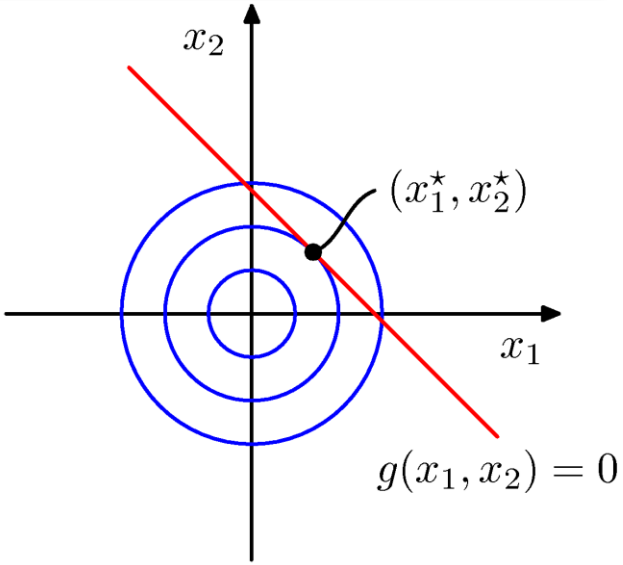
- **2-dimensional optimization**
  - maximize $f(x_1, x_2) = 1 - x_1^2 - x_2^2$
  - subject to $g(x_1, x_2) = x_1 + x_2 - 1 = 0$
- **Lagrangian**

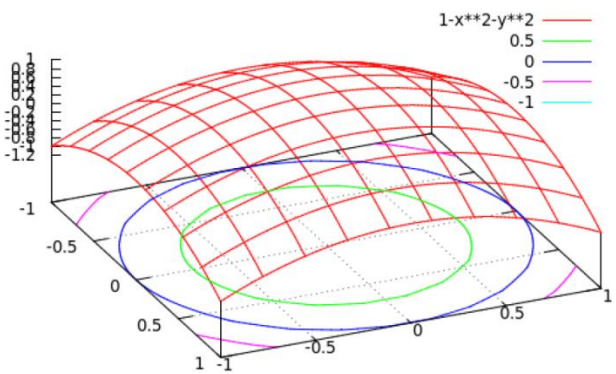
$$L(x_1, x_2, \lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1)$$

- **Stationarity conditions**

$$\partial L(x_1, x_2, \lambda)/\partial x_1 = 0 \iff -2x_1 + \lambda = 0$$
$$\partial L(x_1, x_2, \lambda)/\partial x_2 = 0 \iff -2x_2 + \lambda = 0$$
$$\partial L(x_1, x_2, \lambda)/\partial \lambda = 0 \iff x_1 + x_2 - 1 = 0$$

- **Solution**
  - $(x_1^*, x_2^*) = (0.5, 0.5)$ and $\lambda = 1$





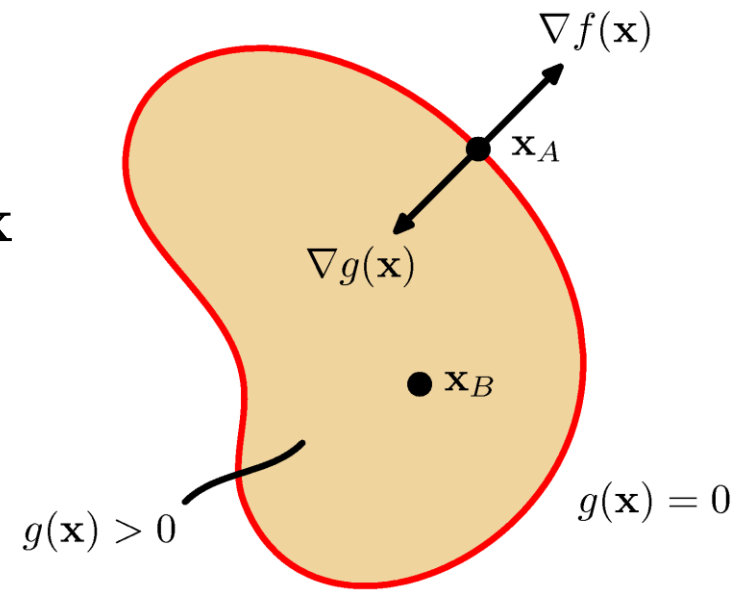
Optimization Geometry - Inequality

- D-dimensional optimization
  - maximize $f(\mathbf{x})$ subject to $g(\mathbf{x}) \geq 0$
  - the constraint defines a region over $\mathbf{x}$
- **Solution $\mathbf{x}_B$ inside region**
  - the constraint is inactive

$$g(\mathbf{x}_B) > 0, \quad \nabla f(\mathbf{x}_B) = 0, \quad \lambda = 0$$

- **Solution $\mathbf{x}_A$ on boundary**
  - the constraint is active
  - vectors $\nabla f(\mathbf{x})$ and $\nabla g(\mathbf{x})$ must be anti-parallel at $\mathbf{x}_A$

$$g(\mathbf{x}_A) = 0, \quad \nabla f(\mathbf{x}_A) = -\lambda \nabla g(\mathbf{x}_A), \quad \lambda > 0$$



- **Property**
  - $\mathbf{x}_A$ and $\mathbf{x}_B$ are stationary points of $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$, if $\lambda g(\mathbf{x}) = 0$

Lagrangian Optimization - Inequality

- **Lagrangian maximization**
  - to maximize $f(\mathbf{x})$ subject to $g(\mathbf{x}) \geq 0$, equivalently …
  - … find the stationary point of the Lagrangian function $L(\mathbf{x},\lambda)$

$$L(\mathbf{x}, \lambda) \equiv f(\mathbf{x}) + \lambda g(\mathbf{x})$$

  - subject to

$$\begin{aligned} g(\mathbf{x}) &\geq 0 \\ \lambda &\geq 0 \\ \lambda g(\mathbf{x}) &\geq 0 \end{aligned}$$

  - known as Karush-Kuhn-Tucker (KTT) conditions
- **Lagrangian minimization**
  - to minimize $f(\mathbf{x})$ subject to $g(\mathbf{x}) \geq 0$, simply …
  - … change the Lagrangian function to $L(\mathbf{x}, \lambda) \equiv f(\mathbf{x}) - \lambda g(\mathbf{x})$

Lagrange Multipliers - Summary

- **Maximization**
  - to maximize $f(\mathbf{x})$ subject to $g_j(\mathbf{x}) = 0$ and $h_k(\mathbf{x}) \geq 0$, …
  - … use Lagrange multipliers λ_j and μ_k and Lagrangian function

$$L(\mathbf{x}, \{\lambda_j\}, \{\mu_k\}) \equiv f(\mathbf{x}) + \sum_{j=1}^{J} \lambda_j g_j(\mathbf{x}) + \sum_{k=1}^{K} \mu_k h_k(\mathbf{x})$$

  - subject to $h_k(\mathbf{x}) \geq 0, \quad \mu_k \geq 0, \quad \mu_k h_k(\mathbf{x}) \geq 0$

- **Minimization**
  - to minimize $f(\mathbf{x})$ subject to $g_j(\mathbf{x}) = 0$ and $h_k(\mathbf{x}) \geq 0$, …

$$L(\mathbf{x}, \{\lambda_j\}, \{\mu_k\}) \equiv f(\mathbf{x}) + \sum_{j=1}^{J} \lambda_j g_j(\mathbf{x}) - \sum_{k=1}^{K} \mu_k h_k(\mathbf{x})$$

  - subject to $h_k(\mathbf{x}) \geq 0, \quad \mu_k \geq 0, \quad \mu_k h_k(\mathbf{x}) \geq 0$

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 16

Support Vector Machines

TUC ECE, Spring 2023

Today

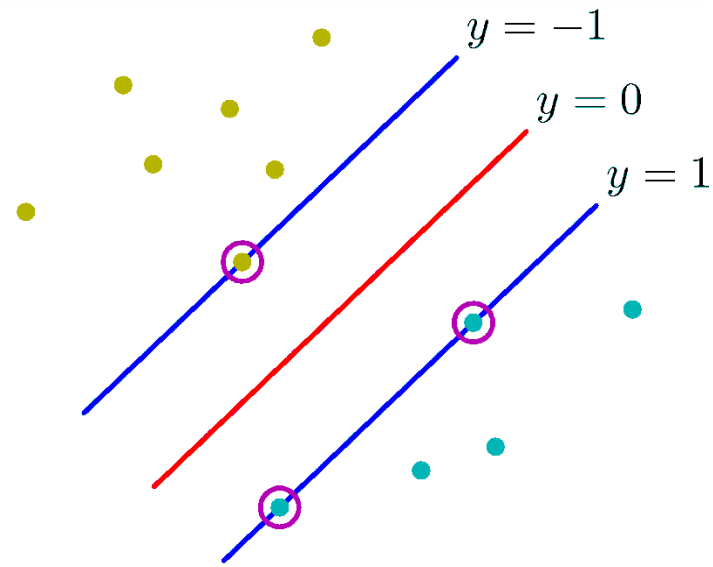
- **SVMs for Non-Overlapping Classes**
- **SVMs for Overlapping Classes**
- **SVMs for Multiple Classes**
- **SVMs for Regression**

SVMs for Non-Overlapping Classes

Maximum Margin Solution

- **Maximum margin**

$$\arg\max_{\mathbf{w},b}\left\{\frac{1}{\|\mathbf{w}\|}\min_{n}\left[t_n\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)+b\right)\right]\right\}$$



- **Constrained optimization**
  - quadratic programming problem
  - given a data set { $(\mathbf{x}_n, t_n)$ } …
  - … and a linear discriminant function $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) + b$

$$\arg\min_{\mathbf{w},b}\frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad t_n\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)+b\right)\geq 1, \quad n=1,2,\ldots,N$$

  - can be solved using the method of Lagrange multipliers

Lagrange Multipliers for Max Margin

- **Problem**

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad t_n\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)+b\right) \geq 1, \quad n=1,2,\ldots,N$$

- **Recall**
  - to minimize $f(\mathbf{x})$ subject to $g_j(\mathbf{x}) = 0$ and $h_k(\mathbf{x}) \geq 0$, …

$$L(\mathbf{x},\{\lambda_j\},\{\mu_k\}) \equiv f(\mathbf{x}) + \sum_{j=1}^{J}\lambda_j g_j(\mathbf{x}) - \sum_{k=1}^{K}\mu_k h_k(\mathbf{x})$$

  - subject to $h_k(\mathbf{x}) \geq 0, \quad \mu_k \geq 0, \quad \mu_k h_k(\mathbf{x}) \geq 0$

- **Lagrangian**

$$L(\mathbf{w},b,\mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n\left\{t_n(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)+b)-1\right\}$$

  - s.t. $t_n\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)+b\right)-1 \geq 0, \; a_n \geq 0, \; a_n\left(t_n\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)+b\right)-1\right) \geq 0$

Dual Formulation

- **Primal**

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b) - 1 \right\}$$

  – s.t. $t_n \left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b\right) - 1 \geq 0, \;\; a_n \geq 0, \;\; a_n \left(t_n \left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b\right) - 1\right) \geq 0$

- **Derivatives**

  – stationarity

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b, \mathbf{a}) = 0 \iff \mathbf{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n)$$

$$\frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial b} = 0 \iff 0 = \sum_{n=1}^{N} a_n t_n$$

- **Dual**

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \qquad k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}')$$

  – subject to $a_n \geq 0, \quad n = 1, 2, \ldots, N, \quad \sum_{n=1}^{N} a_n t_n = 0$

Dual Formulation Properties

- **Dual**

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \qquad k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}')$$

  – subject to $a_n \geq 0, \quad n = 1, 2, \ldots, N, \quad \sum_{n=1}^{N} a_n t_n = 0$

- **Properties**
  – it is a kernelized approach
  – bounded above for positive definite kernel function
  – well-defined, quadratic, convex optimization problem
  – optimal $\mathbf{a}$ can be found by a variety of methods
  – complexity: from $O(M^3)$ in primal to $O(N^3)$ in dual, but …
  – … now can utilize high-dimensional feature spaces!

Prediction

- **Output** $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) + b$ $\qquad \mathbf{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n)$

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n)^T \boldsymbol{\phi}(\mathbf{x}) + b = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x}) + b$$

- **Sparsity**
  - from KKT conditions, …
  - … either $t_n y(\mathbf{x}_n) = 1$ (support vectors)
  - … or must be $a_n = 0$ (most data points)
  - need to keep and use only the support vectors for prediction

$$\begin{aligned} a_n &\geqslant 0 \\ t_n y(\mathbf{x}_n) - 1 &\geqslant 0 \\ a_n \{t_n y(\mathbf{x}_n) - 1\} &= 0 \end{aligned}$$

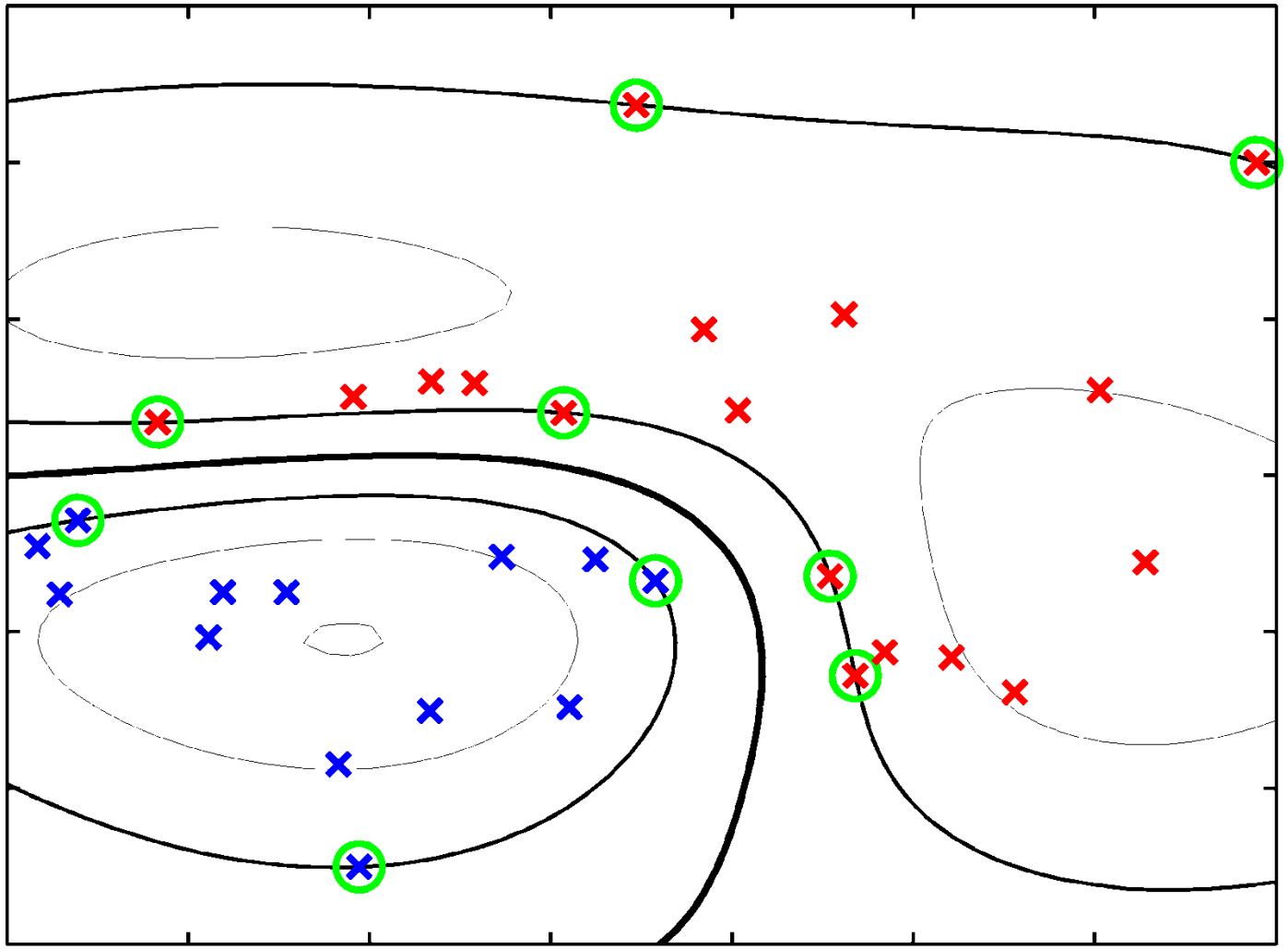
- **Estimating b from support vectors S**

$$t_n \left(\sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1$$

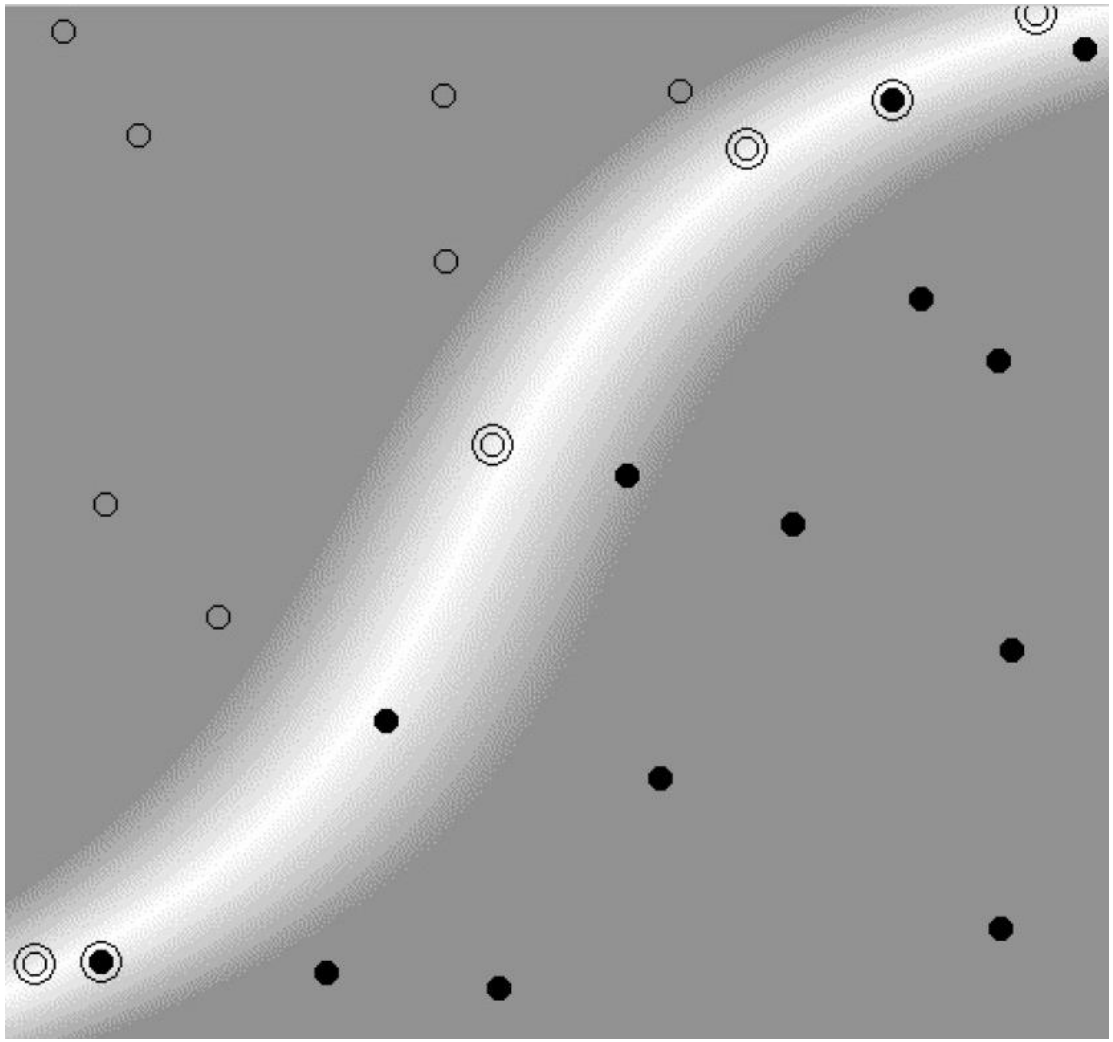
multiply both sides by t_n $\quad t_n^2 = 1$

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right)$$

SVM with Gaussian Kernel



SVM with Cubic Polynomial Kernel



SVMs for Overlapping Classes

Separable vs. Non-Separable Data

- **Separable data**
  - objective $\text{minimize } \frac{1}{2}||\mathbf{w}||^2 \text{ subject to } \forall n : y(\mathbf{x}_n)t_n \geq 1$
  - error function $\sum_{n=1}^{N} E_\infty(y(\mathbf{x}_n)t_n - 1) + \lambda\|\mathbf{w}\|^2$
  - infinite error to misclassified data points $E_\infty(z) = \infty \text{ for } z < 0$
  - zero error to correctly classified data points $E_\infty(z) = 0 \text{ for } z \geq 0$
- **Non-separable data**
  - must allow for some misclassification of data points
  - define penalty for being on the wrong side of the boundary
  - penalty should increase with distance from the boundary
  - for convenience, penalty is a linear function of the distance

Slack Variables

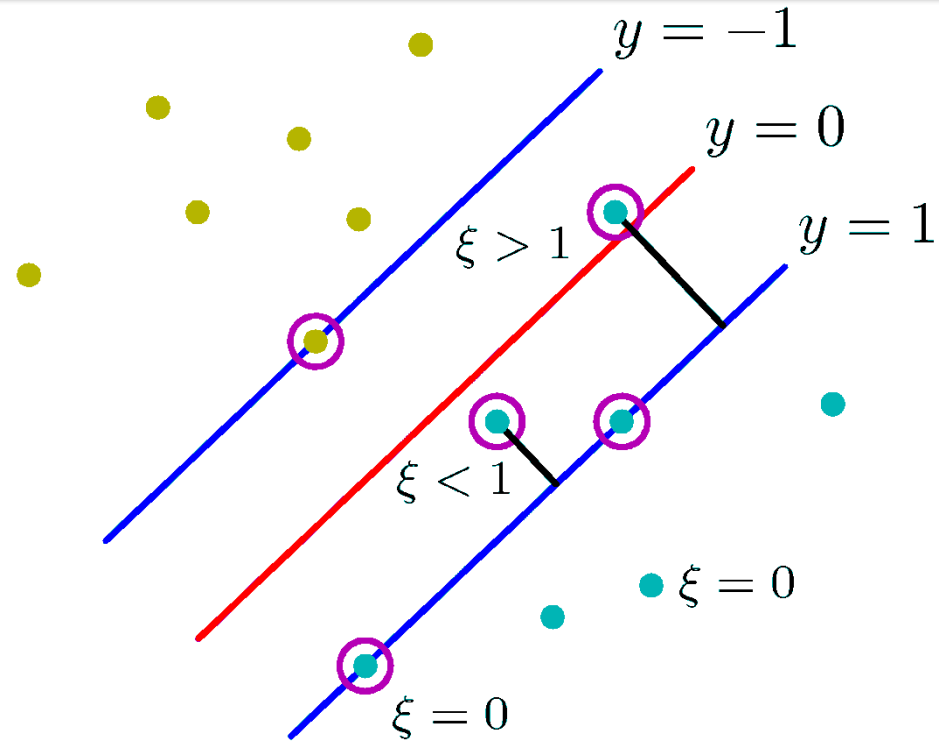
- **Constraints**

$$\forall n : y(\mathbf{x}_n)t_n \geq 1$$

- **Relaxation**

$$\forall n : y(\mathbf{x}_n)t_n \geq 1 - \xi_n$$

- **Slack variables**
  - one per data point
  - constraint $\xi_n \geq 0$
  - if $\xi_n = 0$, $\mathbf{x}_n$ is correctly classified (original problem)
  - if $0 < \xi_n \leq 1$, $\mathbf{x}_n$ is within the margin, but correctly classified
  - if $1 < \xi_n$, $\mathbf{x}_n$ is on the wrong side of the boundary, misclassified



SVM Optimization Problem

- **Original**

$$\text{minimize } \frac{1}{2}||\mathbf{w}||^2 \qquad \text{subject to } \forall n : y(\mathbf{x}_n)t_n \geq 1$$

- **New**

$$\text{minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{n=1}^{N}\xi_n \text{ subject to } \forall n : y(\mathbf{x}_n)t_n \geq 1 - \xi_n, \xi_n \geq 0$$

  - the parameter C>0 controls the trade-off between penalties
  - $\sum_n \xi_n$ is an upper bound to the number of misclassifications
  - C is analogous to (the inverse of) a regularization coefficient
  - C trades off misclassification and model complexity
  - when $C \rightarrow \infty$, the original problem (separable data) is recovered

SVM Lagrangian

- **Lagrangian**

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \mathbf{a}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n \left\{t_n y(\mathbf{x}_n) - 1 + \xi_n\right\} - \sum_{n=1}^{N}\mu_n\xi_n$$

- **Karush-Kuhn-Tucker (KKT) conditions**

$n = 1, \ldots, N$

$$\begin{aligned} a_n &\geqslant 0 \\ t_n y(\mathbf{x}_n) - 1 + \xi_n &\geqslant 0 \\ a_n\left(t_n y(\mathbf{x}_n) - 1 + \xi_n\right) &= 0 \\ \mu_n &\geqslant 0 \\ \xi_n &\geqslant 0 \\ \mu_n\xi_n &= 0 \end{aligned}$$

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) + b$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n)$$

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{n=1}^{N} a_n t_n = 0$$

$$\frac{\partial L}{\partial \xi_n} = 0 \quad \Rightarrow \quad a_n = C - \mu_n.$$

$$\mu_n \geqslant 0 \quad \Rightarrow \quad a_n \leqslant C$$

SVM Dual Lagrangian

- **Maximize**

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

- **subject to**

$$0 \leqslant a_n \leqslant C \qquad n = 1, \ldots, N$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

  - quadratic optimization problem
  - almost identical to the separable case, except *C* !
  - the N constraints on the a_n's are known as box constraints

SVM Prediction

- **Predictive model**

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

  - a_n = 0 : no contribution to prediction, data can be discarded
  - a_n > 0 : support vectors *S*, the sum is taken over them only
  - a_n < *C* : on the margin (μ_n > 0 and ξ_n = 0), correctly classified
  - a_n = *C* : within the margin, correct ($\xi_n \leq 1$) or misclassified ($\xi_n > 1$)
- **Threshold b**
  - *M*: support vectors with $0 < a_n < C$ have ξ_n = 0 and $t_n y(\mathbf{x}_n)$=1, thus

$$t_n \left(\sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1 \qquad b = \frac{1}{N_{\mathcal{M}}} \sum_{n \in \mathcal{M}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right)$$

v-SVM Dual Lagrangian

- **Maximize**

$$\widetilde{L}(\mathbf{a}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

- **subject to**

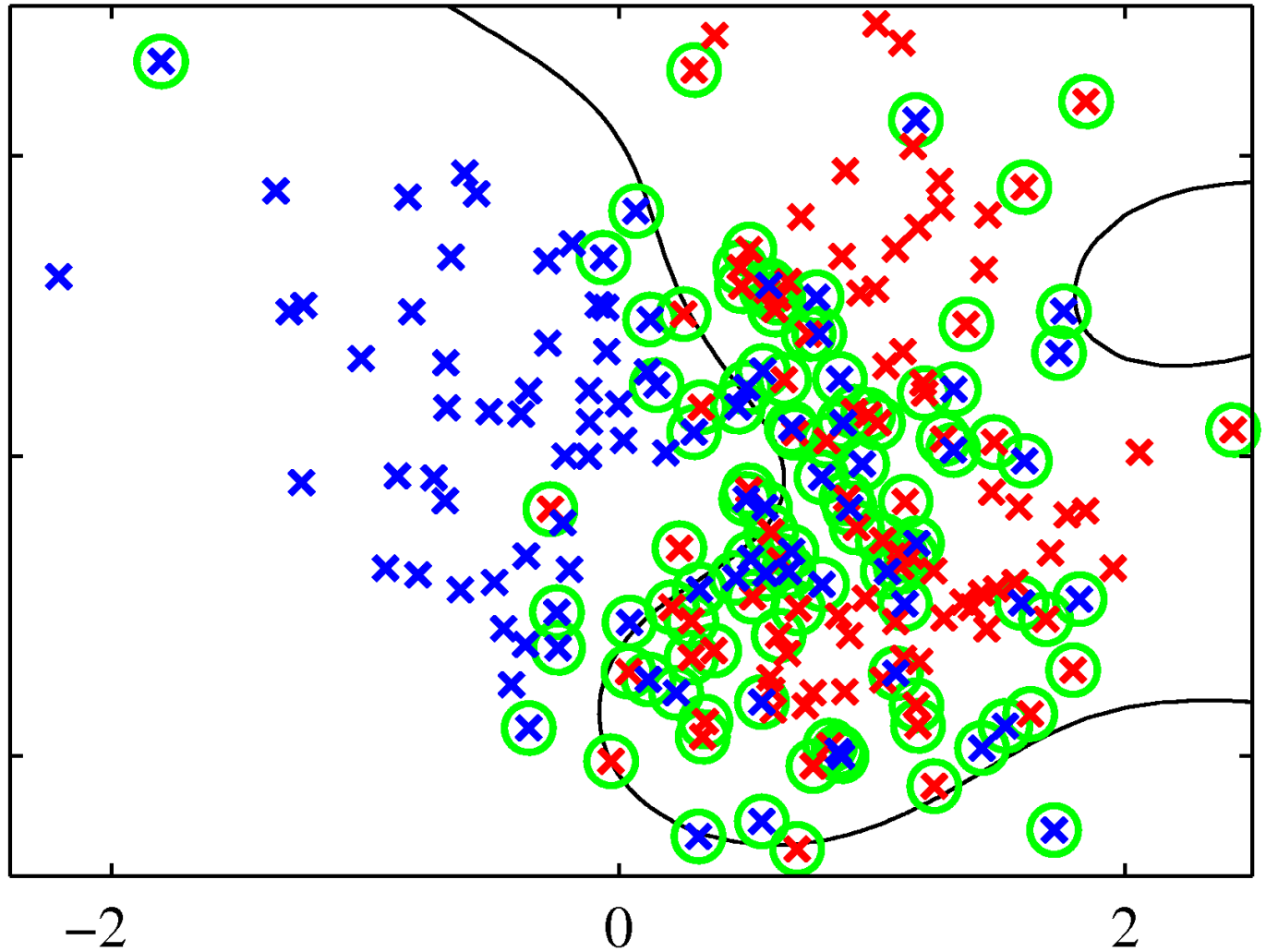
$$0 \leqslant a_n \leqslant 1/N \qquad n = 1, \ldots, N$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

$$\sum_{n=1}^{N} a_n \geqslant \nu.$$

  - quadratic optimization problem
  - *v*: bound, upper on margin errors and lower on support vectors

v-SVM Classification Example

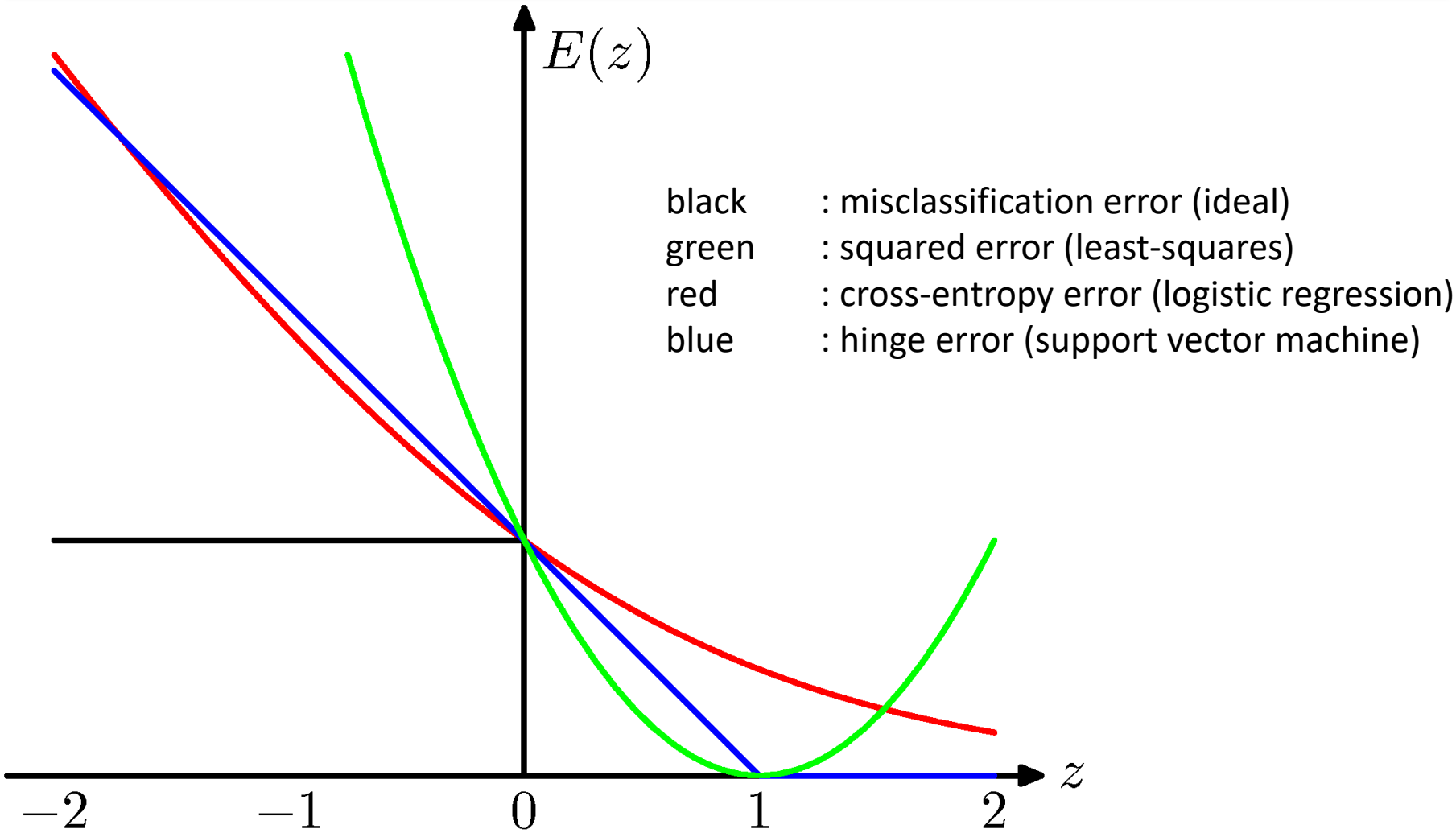


Gaussian kernels with γ=0.45

Quadratic Programming Problem

- **Optimization problem**
  - objective function is quadratic
  - linear constraints define a convex region
  - any local optimum is also a global optimum
- **Approaches**
  - chunking (Vapnik, 1982)
  - protected conjugate gradients (Burges, 1998)
  - decomposition methods (Osuna et al., 1996)
  - sequential minimal optimization (SMO) (Platt, 1999)

Loss Functions (scaled)



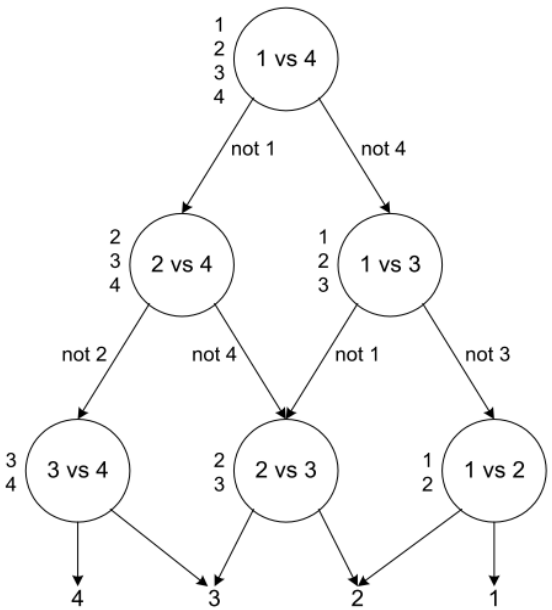
SVMs for Multiple Classes

One-vs-Rest

- **Training**
  - *K* classifiers, one for each class
  - positive examples from class *k*
  - negative examples from all other classes
- **Prediction**
  - if no unique classification, use maximization
- **Problems**
  - imbalanced training sets
    - variant: set target of negative class to -1/(*K*-1)
  - inconsistent results
    - variant: train *K* SVMs simultaneously (large problem!)

One-vs-One

- **Training**
  - train $K(K\text{-}1)/2$ binary classifiers, one for each pair of classes
- **Prediction**
  - select class with max number of votes
- **Problems**
  - inconsistent answers
  - huge training cost
  - huge prediction cost
    - variant: DAGSVM
  - extension: error-correcting codes



SVMs for Regression

SVM Regression Error Function

- **Linear regression regularized error function**

$$\frac{1}{2}\sum_{n=1}^{N}\{y_n - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

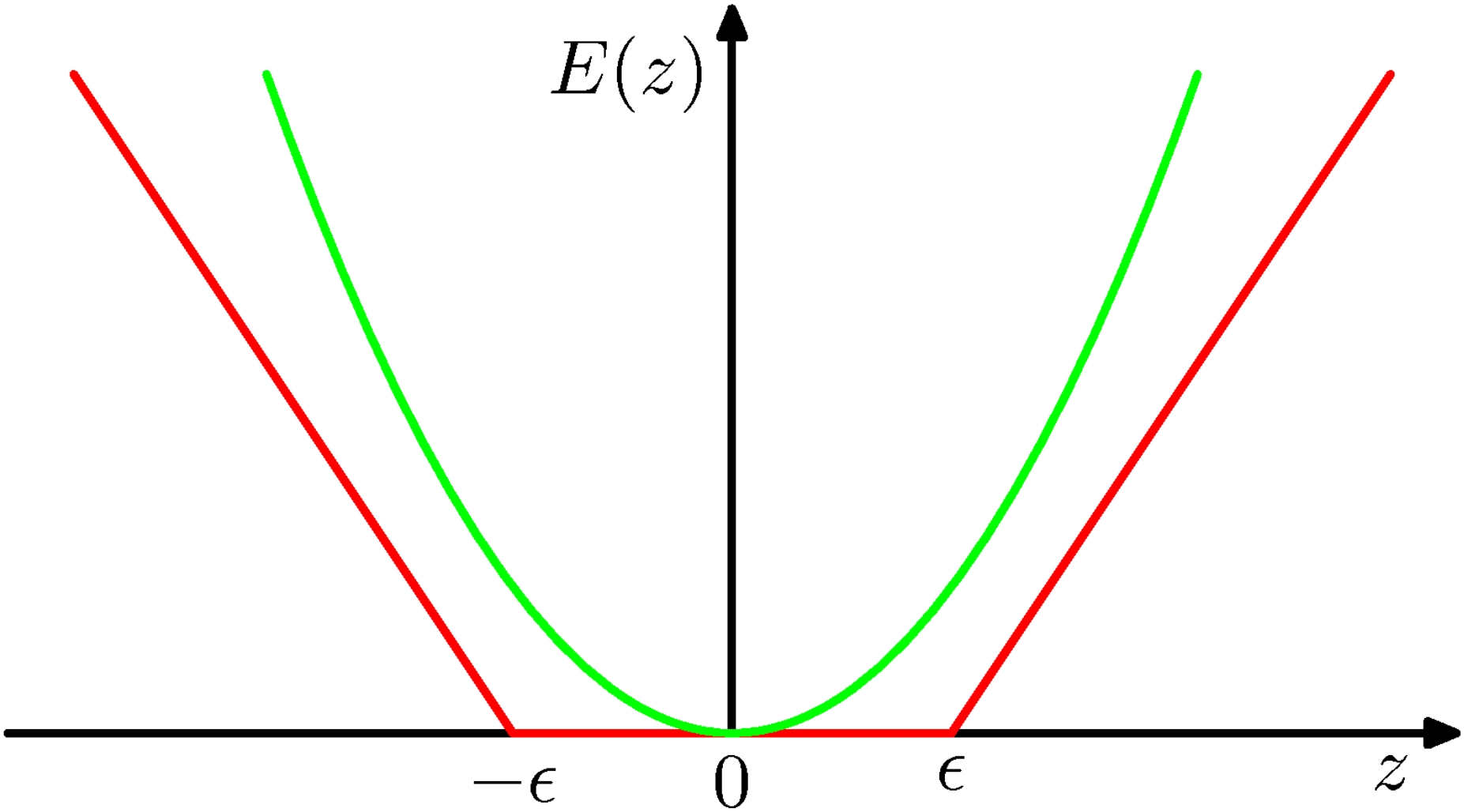
- **ϵ-insensitive error function**

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon; \\ |y(\mathbf{x}) - t| - \epsilon, & \text{otherwise} \end{cases}$$

- **ϵ-insensitive regularized error function**

$$C\sum_{n=1}^{N} E_\epsilon(y(\mathbf{x}_n) - t_n) + \frac{1}{2}\|\mathbf{w}\|^2$$

ϵ-Insensitive vs. Quadratic Error



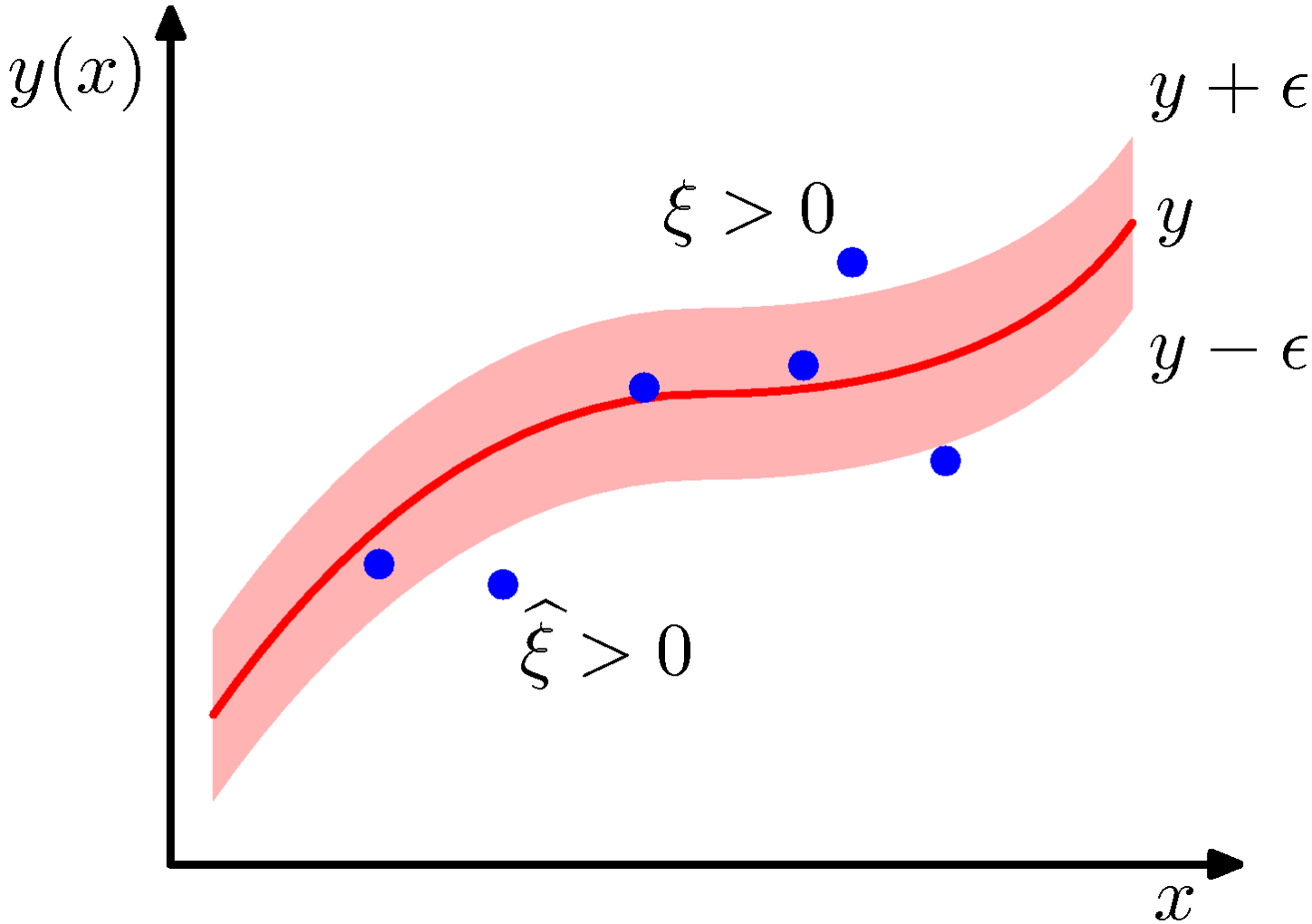
SVMs Regression Formulation

- **Slack variables**
  - two slack variables for each data point
- **ϵ-tube**
  - targets must lie with the ϵ-tube (plus/minus slack)

$$\begin{aligned} t_n &\leqslant y(\mathbf{x}_n) + \epsilon + \xi_n \\ t_n &\geqslant y(\mathbf{x}_n) - \epsilon - \widehat{\xi}_n \end{aligned}$$

- **SVM quadratic programming**
  - minimize $$C \sum_{n=1}^{N} (\xi_n + \widehat{\xi}_n) + \frac{1}{2} \|\mathbf{w}\|^2$$
  - subject to $\xi_n \geqslant 0$ and $\widehat{\xi}_n \geqslant 0$ $$\begin{aligned} t_n &\leqslant y(\mathbf{x}_n) + \epsilon + \xi_n \\ t_n &\geqslant y(\mathbf{x}_n) - \epsilon - \widehat{\xi}_n \end{aligned}$$

ϵ-Tube Example



SVM Lagrangian

- **SVM Lagrangian**

$$L = C\sum_{n=1}^{N}(\xi_n + \widehat{\xi}_n) + \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N}(\mu_n\xi_n + \widehat{\mu}_n\widehat{\xi}_n)$$
$$- \sum_{n=1}^{N} a_n(\epsilon + \xi_n + y_n - t_n) - \sum_{n=1}^{N} \widehat{a}_n(\epsilon + \widehat{\xi}_n - y_n + t_n)$$

- **Gradients to zero**

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{n=1}^{N}(a_n - \widehat{a}_n)\boldsymbol{\phi}(\mathbf{x}_n)$$

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{n=1}^{N}(a_n - \widehat{a}_n) = 0$$

$$\frac{\partial L}{\partial \xi_n} = 0 \quad \Rightarrow \quad a_n + \mu_n = C$$

$$\frac{\partial L}{\partial \widehat{\xi}_n} = 0 \quad \Rightarrow \quad \widehat{a}_n + \widehat{\mu}_n = C.$$

SVM Dual Lagrangian

- **SVM dual Lagrangian**
  - maximize

$$\widetilde{L}(\mathbf{a}, \widehat{\mathbf{a}}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(a_n - \widehat{a}_n)(a_m - \widehat{a}_m)k(\mathbf{x}_n, \mathbf{x}_m) - \epsilon\sum_{n=1}^{N}(a_n + \widehat{a}_n) + \sum_{n=1}^{N}(a_n - \widehat{a}_n)t_n$$

  - subject to

$$0 \leqslant a_n \leqslant C$$
$$0 \leqslant \widehat{a}_n \leqslant C$$

  - box constraints!

SVM Regression Prediction

- **Prediction**

$$y(\mathbf{x}) = \sum_{n=1}^{N} (a_n - \widehat{a}_n) k(\mathbf{x}, \mathbf{x}_n) + b$$

- **Observations**
  - from KKT conditions
  - non-zero coefficients
    for points outside the tube
    or on the tube boundary
  - for each data point,
    one coefficient is zero (or both)

$$\begin{aligned} a_n(\epsilon + \xi_n + y_n - t_n) &= 0 \\ \widehat{a}_n(\epsilon + \widehat{\xi}_n - y_n + t_n) &= 0 \\ (C - a_n)\xi_n &= 0 \\ (C - \widehat{a}_n)\widehat{\xi}_n &= 0 \end{aligned}$$

v-SVM for Regression

- ***v*-SVM dual Lagrangian**
  - maximize

$$\widetilde{L}(\mathbf{a}, \widehat{\mathbf{a}}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(a_n - \widehat{a}_n)(a_m - \widehat{a}_m)k(\mathbf{x}_n, \mathbf{x}_m) + \sum_{n=1}^{N}(a_n - \widehat{a}_n)t_n$$

  - subject to

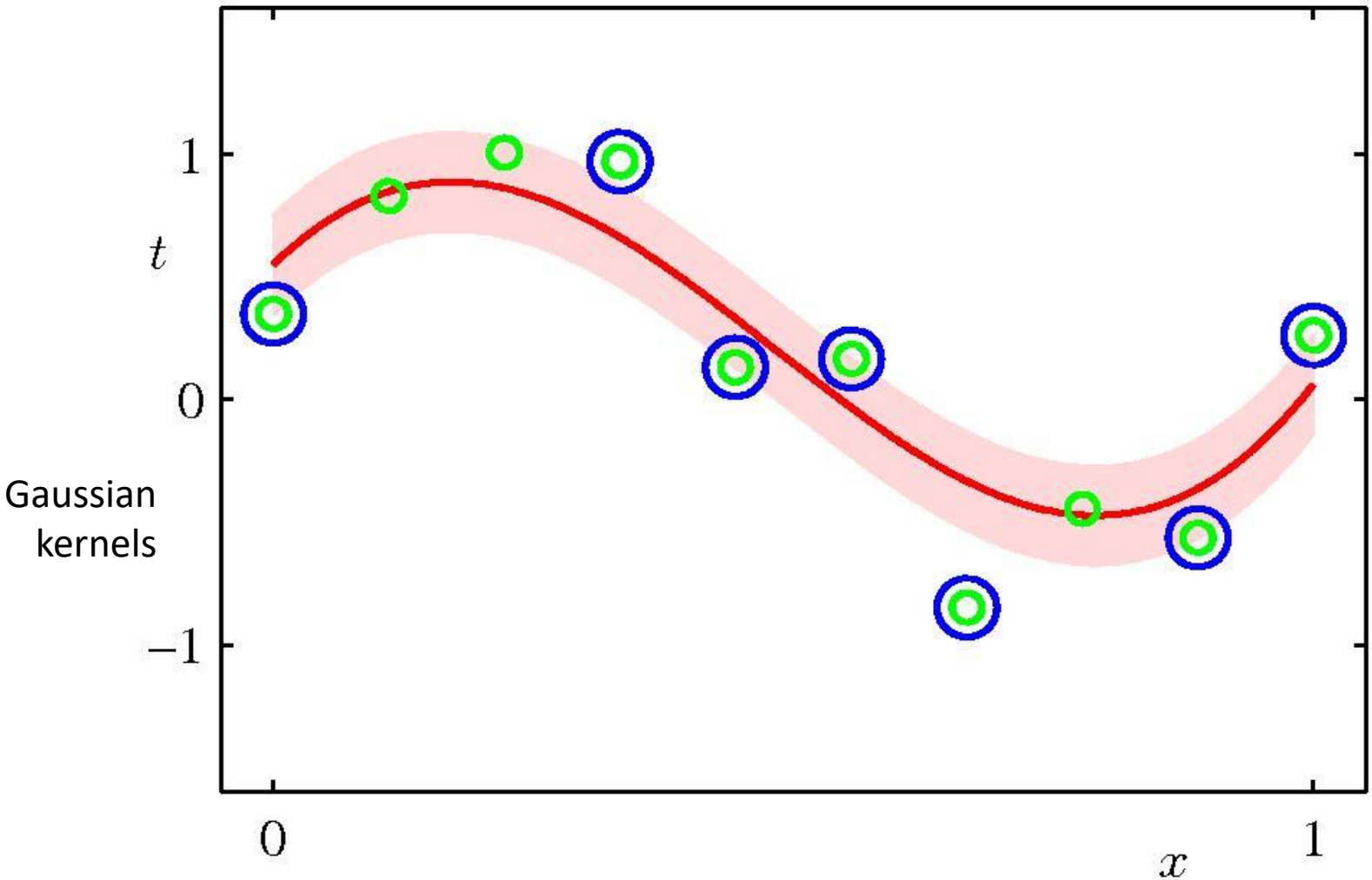
$$\sum_{n=1}^{N}(a_n - \widehat{a}_n) = 0$$

$$0 \leqslant a_n \leqslant C/N$$

$$0 \leqslant \widehat{a}_n \leqslant C/N$$

$$\sum_{n=1}^{N}(a_n + \widehat{a}_n) \leqslant \nu C$$

SVM Regression Example



Gaussian kernels

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 17

Relevance Vector Machines

TUC ECE, Spring 2023

Today

- **Relevance Vector Machines**
- **RVMs for Regression**
- **RVMs for Classification**

Relevance Vector Machines

From SVMs to RVMs

- **SVM limitations**
  - no probabilistic interpretation of the output
  - no natural extension to multiple classes
  - difficulty in tuning key parameters (C, v, ϵ)
  - kernel functions must be positive definite
  - solutions are not so sparse
- **Relevance Vector Machines (RVMs)**
  - Bayesian sparse kernel technique
  - applies to both regression and classification
  - overcomes the principal SVM limitations
  - leads to sparser solutions with comparable generalization

RVMs for Regression

RVM Regression Model

- **Linear regression model**
  - for some noise parameter β and a set of basis functions $\boldsymbol{\phi}$, the predictive distribution is

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$$

- **RVM regression model**

$$y(\mathbf{x}) = \sum_{n=1}^{N} w_n k(\mathbf{x}, \mathbf{x}_n) + b$$

  - a total of $M = N+1$ parameters ($w_M = b$)
  - no restriction to positive definite kernels
  - kernels are utilized as basis functions (features)
  - no necessity for centering on the data points

Recall: Bayesian Linear Regression

- **Common conjugate prior over $\mathbf{w}$**

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

- **Likelihood**

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) = \mathcal{N}(\mathbf{t}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I})$$

- **Posterior**

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

$$\begin{aligned} \mathbf{m}_N &= \beta\mathbf{S}_N\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} \\ \mathbf{S}_N^{-1} &= \alpha\mathbf{I} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} \end{aligned}$$

- **Reminder: General case**

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

$$\begin{aligned} \mathbf{m}_N &= \mathbf{S}_N\left(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}\right) \\ \mathbf{S}_N^{-1} &= \mathbf{S}_0^{-1} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} \end{aligned}$$

RVM Probabilities

- **Likelihood**

$$p(\mathbf{t}|\mathbf{X},\mathbf{w},\beta) = \prod_{n=1}^{N} p(t_n|\mathbf{x}_n,\mathbf{w},\beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n),\beta^{-1})$$

- **ARD Prior**

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=1}^{M} \mathcal{N}(w_i|0,\alpha_i^{-1})$$

  – zero-mean Gaussian with a different precision for each weight
  – most precisions go to infinity, giving zero weights (sparsity)

- **Posterior**

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m},\mathbf{S}) \qquad \begin{aligned} \mathbf{m} &= \beta\mathbf{S}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} \\ \mathbf{S}^{-1} &= \mathbf{A} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} \end{aligned}$$

  – $\boldsymbol{\Phi}$ is the $N\times M$ design matrix with Φ_{nM}=1 (could be $\mathbf{K}$), $\mathbf{A}$ = diag(α_i)

Recall: Maximizing the Evidence

- **Iterative maximization**
  - give arbitrary values to α and β and iterate until convergence
  - step I: given α and β, compute γ and $\mathbf{m}_N$

$$\mathbf{m}_N = \beta \mathbf{S}_N \mathbf{\Phi}^{\mathrm{T}} \mathbf{t}$$

$$\gamma = \sum_i \frac{\lambda_i}{\alpha + \lambda_i}$$

  - step II: given γ and $\mathbf{m}_N$, compute α and β

$$\alpha = \frac{\gamma}{\mathbf{m}_N^{\mathrm{T}} \mathbf{m}_N}$$

$$\frac{1}{\beta} = \frac{1}{N - \gamma} \sum_{n=1}^{N} \left(t_n - \mathbf{m}_N^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right)^2$$

Evidence Approximation for Parameters

- **Marginal likelihood**

$$p(\mathbf{t}|\mathbf{X}, \boldsymbol{\alpha}, \beta) = \int p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) p(\mathbf{w}|\boldsymbol{\alpha}) \, \mathrm{d}\mathbf{w}$$

- **Log marginal likelihood**

$$\begin{aligned} \ln p(\mathbf{t}|\mathbf{X}, \boldsymbol{\alpha}, \beta) &= \ln \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C}) \\ &= -\frac{1}{2} \left\{ N \ln(2\pi) + \ln |\mathbf{C}| + \mathbf{t}^{\mathrm{T}} \mathbf{C}^{-1} \mathbf{t} \right\} \end{aligned}$$

$$\mathbf{C} = \beta^{-1} \mathbf{I} + \boldsymbol{\Phi} \mathbf{A}^{-1} \boldsymbol{\Phi}^{\mathrm{T}}$$

- **Iterative estimation**

$$\begin{aligned} \alpha_i^{\mathrm{new}} &= \frac{\gamma_i}{m_i^2} \\ (\beta^{\mathrm{new}})^{-1} &= \frac{\|\mathbf{t} - \boldsymbol{\Phi}\mathbf{m}\|^2}{N - \sum_i \gamma_i} \end{aligned}$$

$$\begin{aligned} \mathbf{m} &= \beta \mathbf{S} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t} \\ \mathbf{S}^{-1} &= \mathbf{A} + \beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \\ \gamma_i &= 1 - \alpha_i S_{ii} \end{aligned}$$

RVM Predictive Distribution

- **Prediction**
  - given the optimized hyperparameters $\boldsymbol{\alpha}^*$, β^*

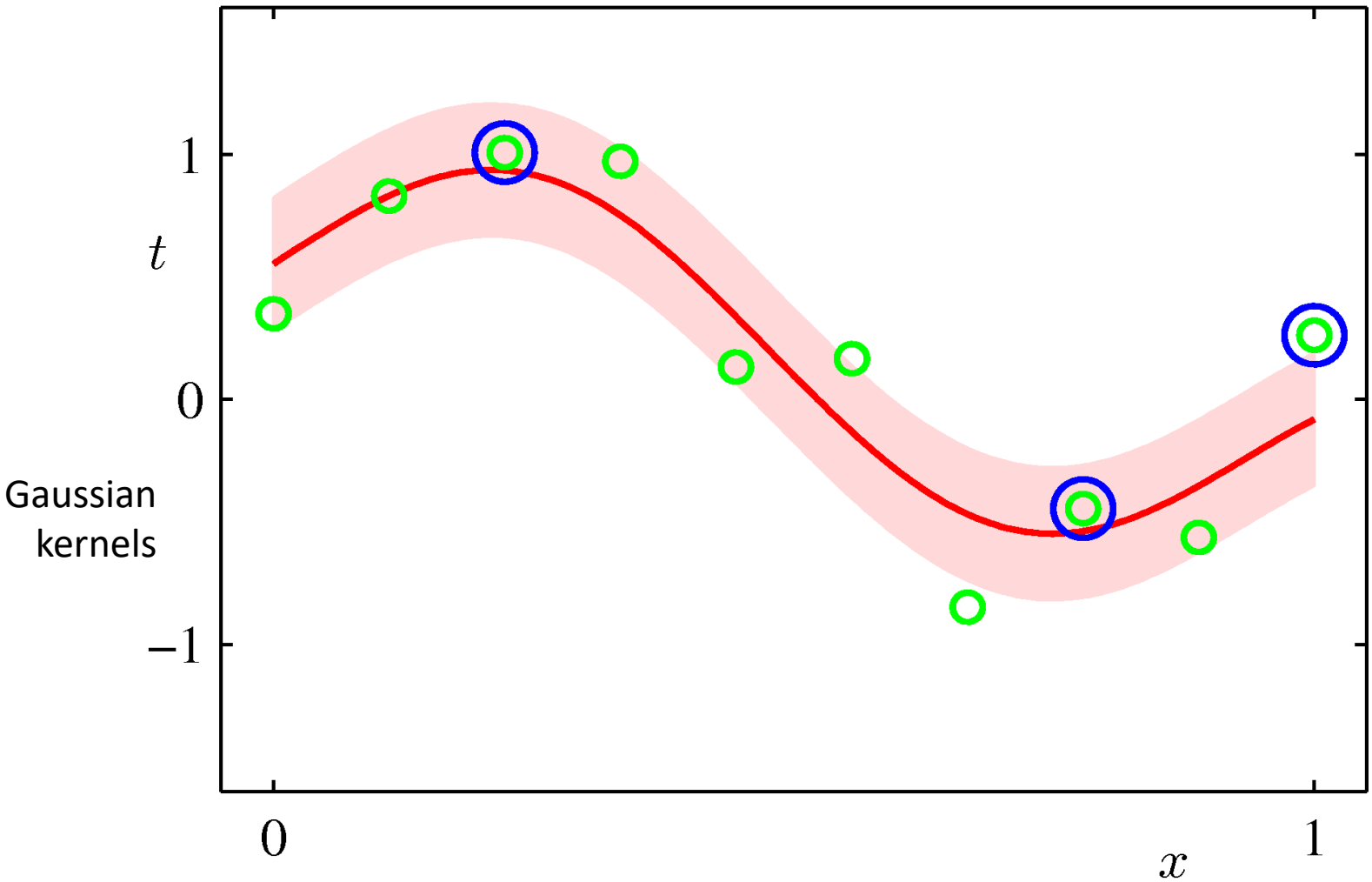
$$
\begin{aligned}
p(t|\mathbf{x}, \mathbf{X}, \mathbf{t}, \boldsymbol{\alpha}^\star, \beta^\star) &= \int p(t|\mathbf{x}, \mathbf{w}, \beta^\star) p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \boldsymbol{\alpha}^\star, \beta^\star)\, \mathrm{d}\mathbf{w} \\
&= \mathcal{N}\left(t|\mathbf{m}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}), \sigma^2(\mathbf{x})\right).
\end{aligned}
$$

  - where

$$
\begin{aligned}
\mathbf{m} &= \beta^\star \mathbf{S}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} \\
\mathbf{S} &= \left(\mathbf{A}^\star + \beta^\star \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1} \\
\sigma^2(\mathbf{x}) &= (\beta^\star)^{-1} + \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{S}\boldsymbol{\phi}(\mathbf{x})
\end{aligned}
$$

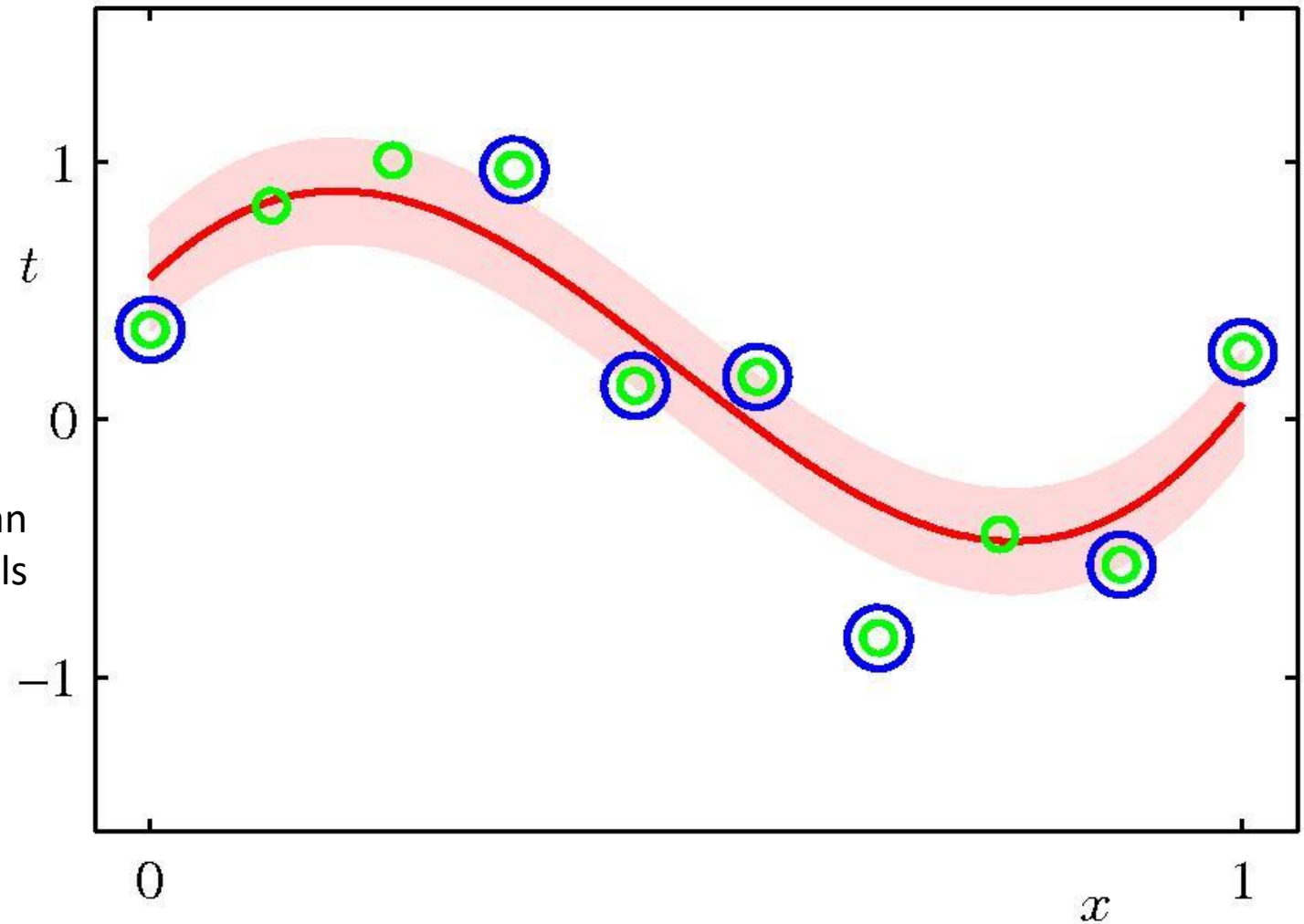
  - many points have zero weight and can be omitted
  - points with non-zero weight are called *relevance vectors*

RVM Regression Example

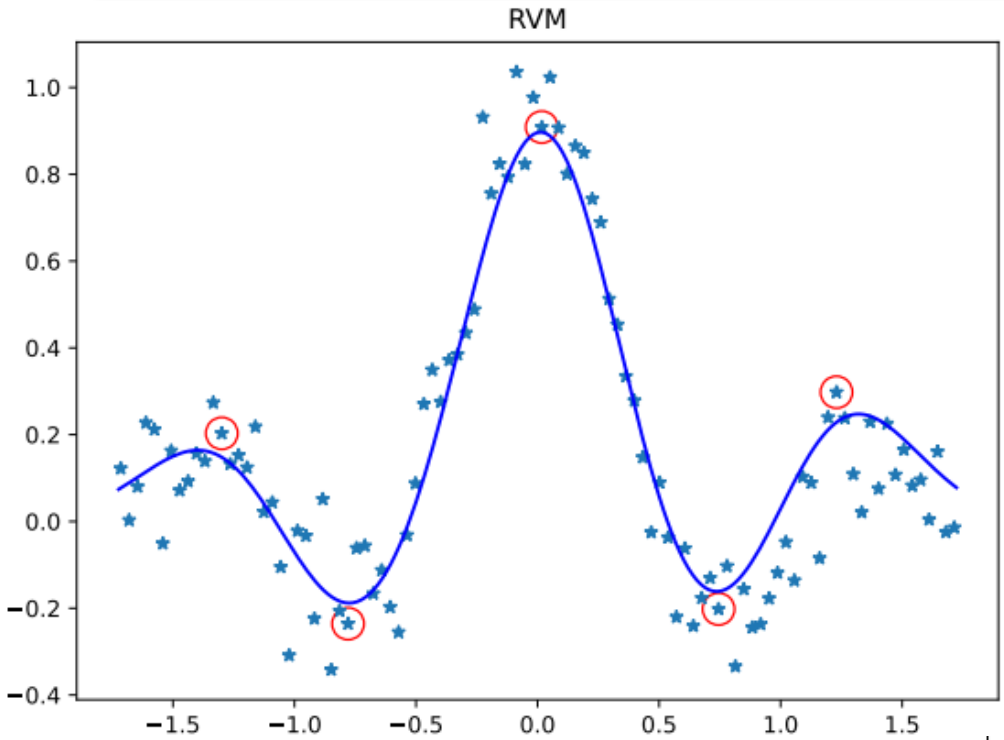


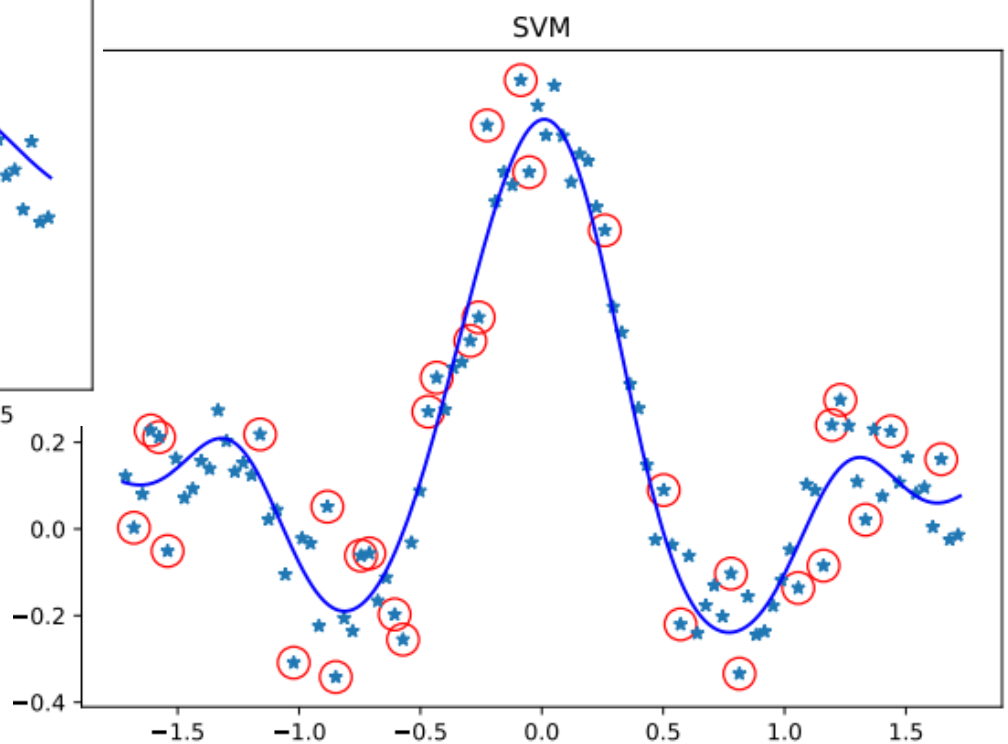
Comparison to SVM Regression

Gaussian kernels



Comparison: RVM vs. SVM





Gaussian kernels with γ=0.30

RVMs for Classification

RVM Logistic Regression Model

- **Logistic regression model**
  - for targets t in {0,1}

$$p(\mathcal{C}_1|\mathbf{x},\mathbf{w}) = y(\mathbf{x},\mathbf{w}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})\right) = \frac{1}{1+\exp\left(-\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})\right)}$$

- **RVM logistic regression model**

$$y(\mathbf{x},\mathbf{w}) = \sigma\left(\sum_{n=1}^{N} w_n k(\mathbf{x},\mathbf{x}_n) + b\right)$$

  - a total of M = N+1 parameters (w_M = b)
  - kernels are utilized as basis functions (features)

- **ARD Prior**

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=1}^{M} \mathcal{N}(w_i|0,\alpha_i^{-1}) = \mathcal{N}(\mathbf{w}|\mathbf{0},\mathbf{A}^{-1})$$

Posterior

- **Posterior**
  - integrating out $\mathbf{w}$ for the predictive distribution is intractable!
  - idea: apply Laplace approximation!
- **Log posterior**

$$\ln p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha}) = \ln\{p(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})\} - \ln p(\mathbf{t}|\boldsymbol{\alpha}) \qquad y_n = \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}_n)$$

$$= \sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} - \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{A}\mathbf{w} + \mathrm{const}$$

- **IRLS for the mode $\mathbf{w}$* and the negative Hessian**

$$\nabla \ln p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha}) = \boldsymbol{\Phi}^{\mathrm{T}}(\mathbf{t}-\mathbf{y}) - \mathbf{A}\mathbf{w} \qquad \mathbf{w}^{\star} = \mathbf{A}^{-1}\boldsymbol{\Phi}^{\mathrm{T}}(\mathbf{t}-\mathbf{y})$$

$$\nabla\nabla \ln p(\mathbf{w}|\mathbf{t},\boldsymbol{\alpha}) = -\left(\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{B}\boldsymbol{\Phi} + \mathbf{A}\right) \qquad \boldsymbol{\Sigma} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{B}\boldsymbol{\Phi} + \mathbf{A}\right)^{-1}$$

  - B is a $N \times N$ diagonal matrix with elements $b_n = y_n(1-y_n)$

Evidence Approximation

- **Laplace approximation**

$$p(\mathbf{w}|\boldsymbol{\alpha}) \approx \mathcal{N}(\mathbf{w}|\mathbf{w}^\star, \Sigma)$$

$$\begin{aligned} \mathbf{w}^\star &= \mathbf{A}^{-1}\boldsymbol{\Phi}^{\mathrm{T}}(\mathbf{t}-\mathbf{y}) \\ \boldsymbol{\Sigma} &= \left(\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{B}\boldsymbol{\Phi}+\mathbf{A}\right)^{-1} \end{aligned}$$

- **Marginal likelihood**

$$\begin{aligned} p(\mathbf{t}|\boldsymbol{\alpha}) &= \int p(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})\,\mathrm{d}\mathbf{w} \\ &\simeq p(\mathbf{t}|\mathbf{w}^\star)p(\mathbf{w}^\star|\boldsymbol{\alpha})(2\pi)^{M/2}|\boldsymbol{\Sigma}|^{1/2} \end{aligned}$$

- **Iterative estimation**

$$\alpha_i^{\mathrm{new}} = \frac{\gamma_i}{(w_i^\star)^2} \qquad \gamma_i = 1 - \alpha_i \Sigma_{ii}$$

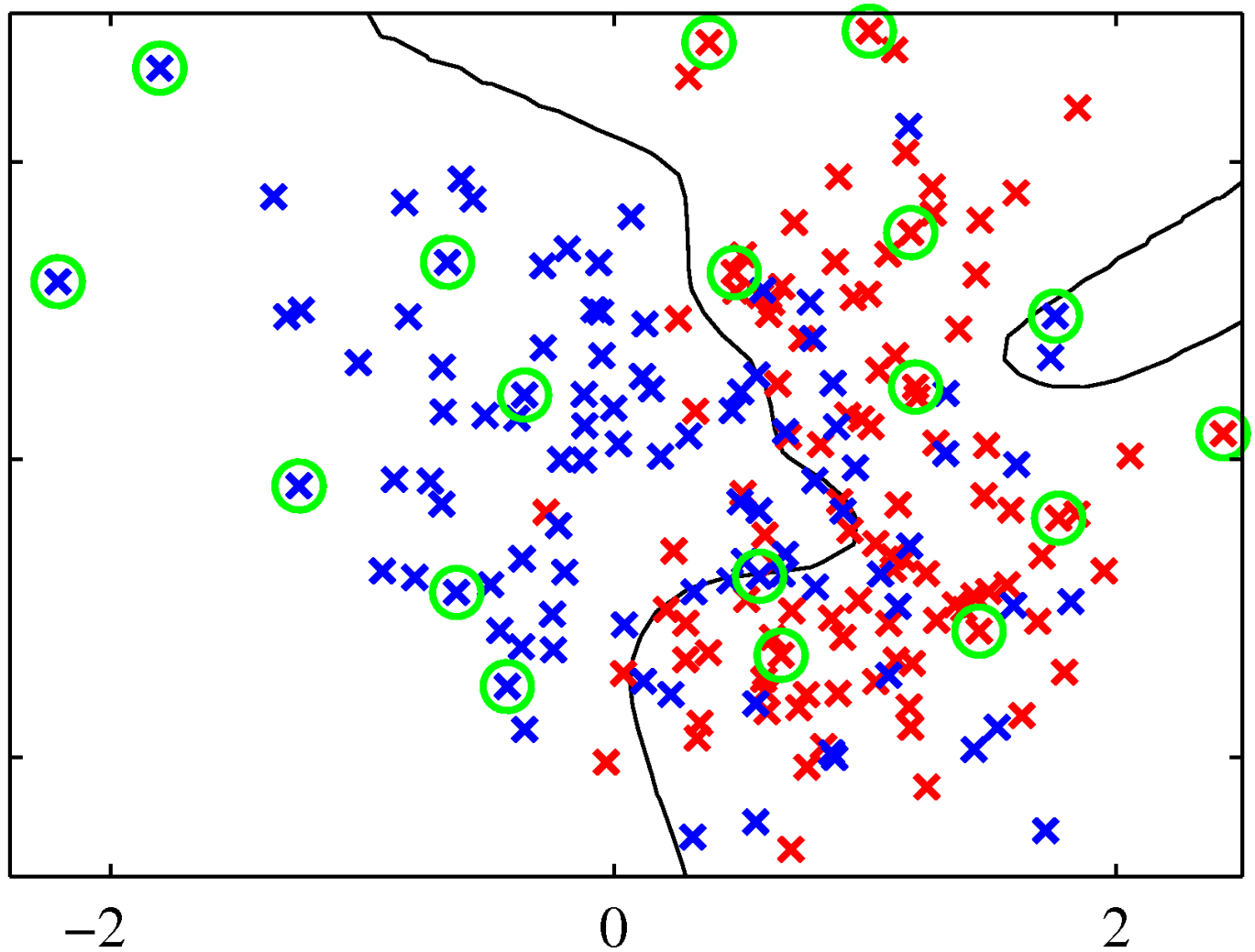
- **Log marginal likelihood**

$$\ln p(\mathbf{t}|\boldsymbol{\alpha}) = -\frac{1}{2}\left\{N\ln(2\pi) + \ln|\mathbf{C}| + (\widehat{\mathbf{t}})^{\mathrm{T}}\mathbf{C}^{-1}\widehat{\mathbf{t}}\right\}$$

$$\widehat{\mathbf{t}} = \boldsymbol{\Phi}\mathbf{w}^\star + \mathbf{B}^{-1}(\mathbf{t}-\mathbf{y})$$

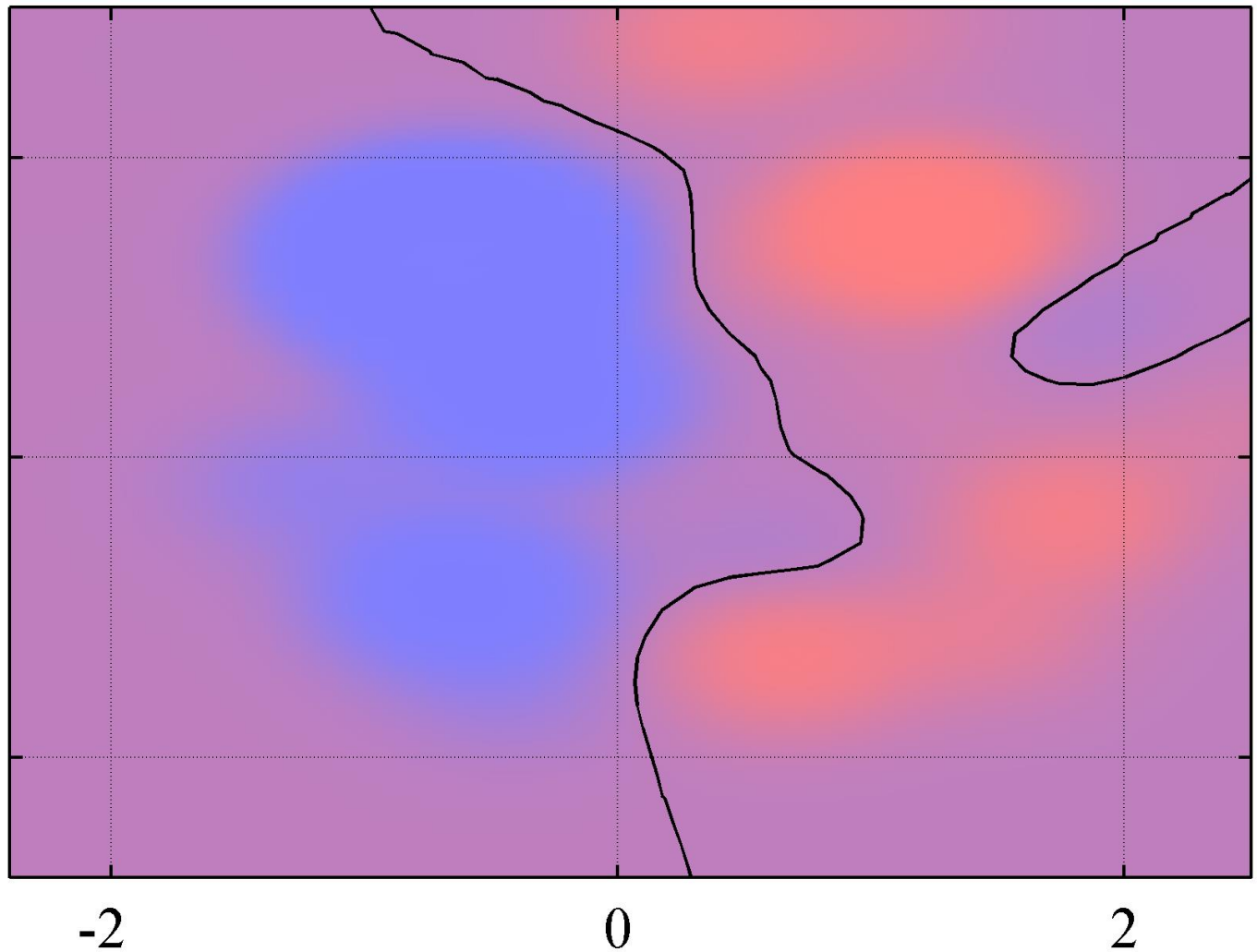
$$\mathbf{C} = \mathbf{B} + \boldsymbol{\Phi}\mathbf{A}\boldsymbol{\Phi}^{\mathrm{T}}$$

RVM Classification Example



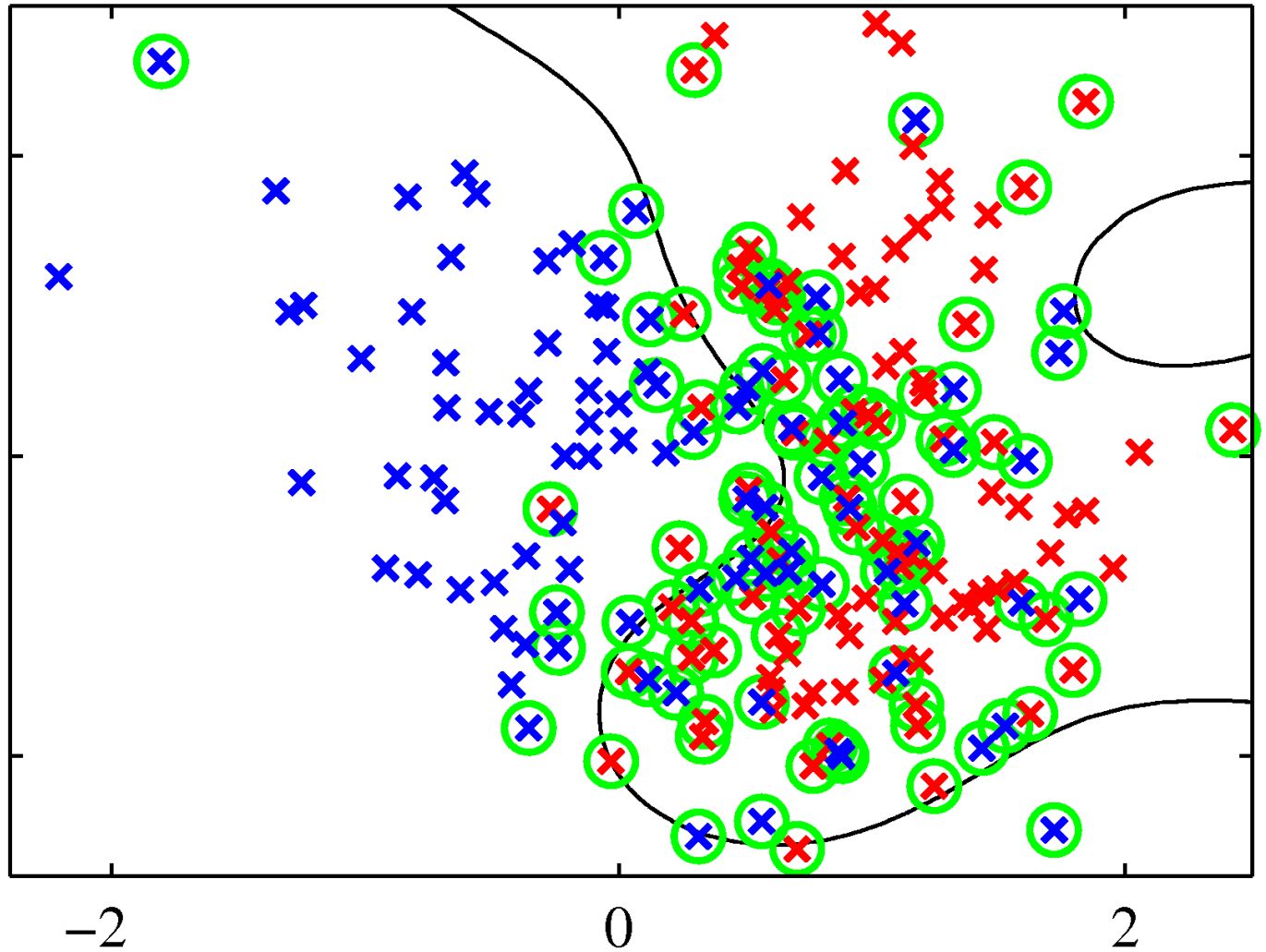
Gaussian kernels with γ=0.45

RVM Posterior Probabilities



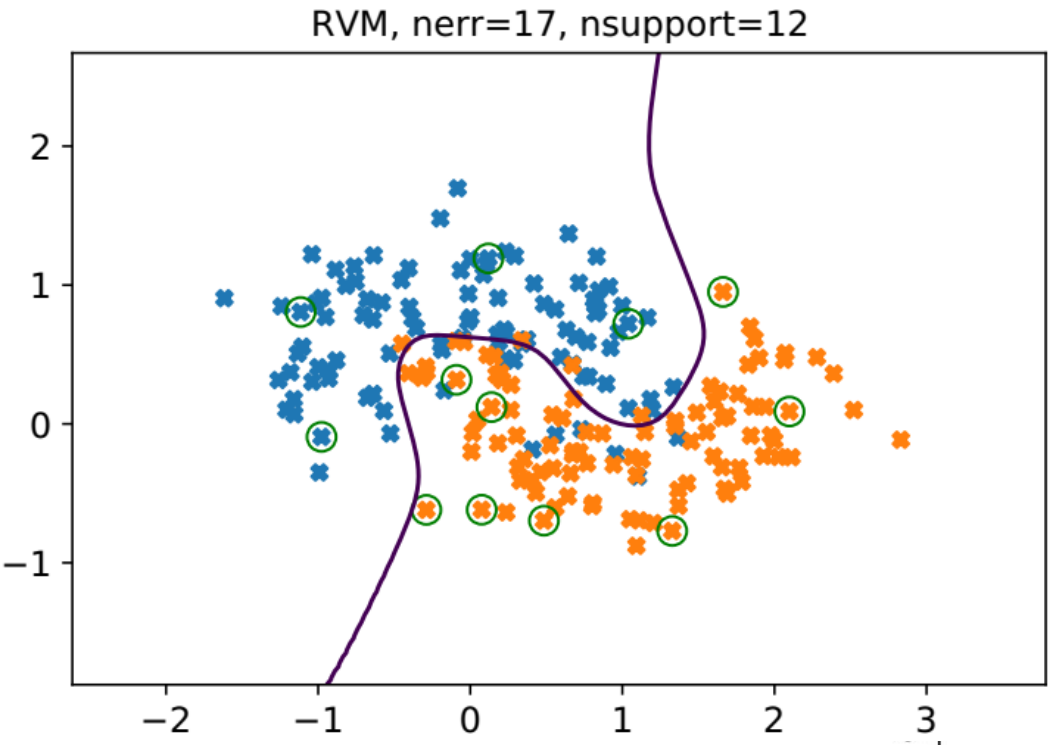
Gaussian kernels with γ=0.45

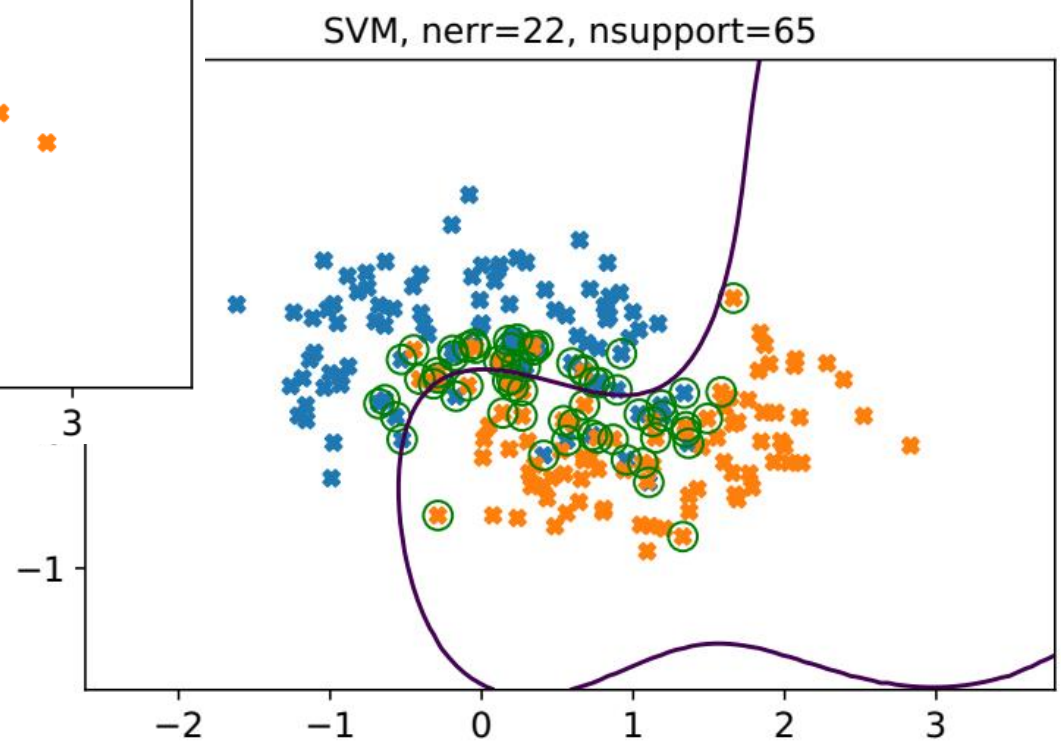
Comparison to *v*-SVM Classification



Gaussian kernels with γ=0.45

Comparison: RVM vs. SVM





Gaussian kernels with γ=0.30

RVM Multi-Class Classification

- **Prediction**
  - *K* models, one for each class, combine using softmax

$$y_k(\mathbf{x}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \qquad a_k = \mathbf{w}_k^{\mathrm{T}}\mathbf{x}$$

- **Log likelihood**

$$\ln p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

  - 1-of-*K* coding for each data point
  - Laplace approximation to optimize the hyperparameters
  - mode and $MK \times MK$ Hessian found through IRLS
  - cons: additional factor of K^3 to computational cost of training
  - pros: sparser models, faster prediction, no cross-validation

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 18

Principal Component Analysis

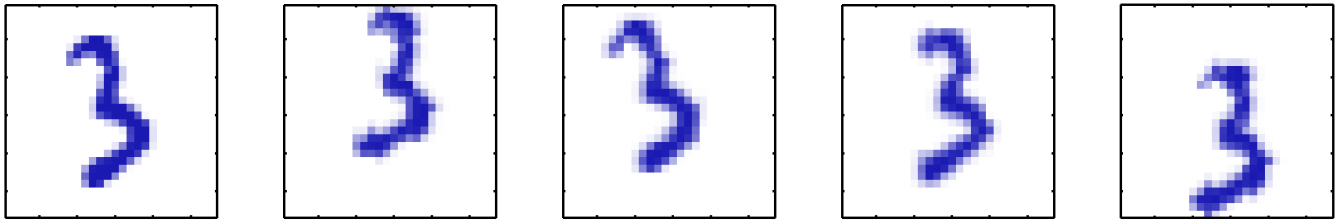
TUC ECE, Spring 2023

Today

- **Principal Component Analysis**
- **PCA Applications**
- **PCA for High-Dimensional Data**

Motivation

- **Synthetic data**
  - embed a 64×64 image of a handwritten digit into 100×100
  - random choices in x-translation, y-translation, θ-rotation



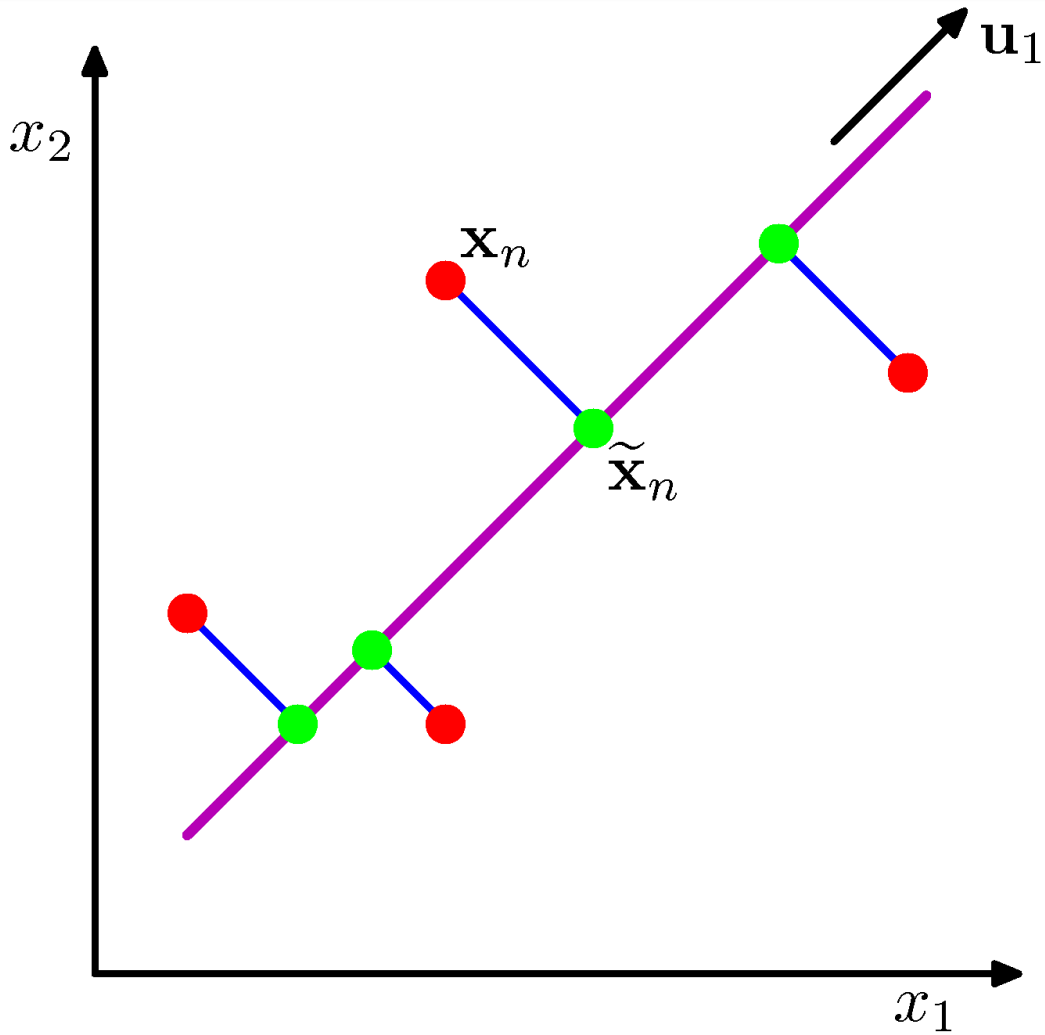
- **Dimensionality**
  - each resulting image is a point in a 10,000-dimensional space
  - however, there are only three degrees of freedom of variability
  - data live in a non-linear manifold of intrinsic dimensionality 3
  - question: is it possible to identify these intrinsic dimensions?

Principal Components Analysis

Principal Components Analysis (PCA)

- **PCA**
  - technique for dimensionality reduction, lossy compression, feature extraction, data visualization, …
  - also known as the Karhunen-Loeve transform
- **Definitions**
  - orthogonal projection of the data onto the lower-dimensional principal subspace, so that variance in projection is maximized
  - linear projection that minimizes the average projection cost (mean squared distance between data and their projections)
- **Data and Goal**
  - $\{\mathbf{x}_n\}$, n=1,…,N, of dimension D projected to M<D dimensions

PCA Projection Example



Maximum Variance Formulation

- for M=1, select a D-dimensional unit vector $\mathbf{u}_1$ ($\mathbf{u}_1{}^{\mathrm{T}}\mathbf{u}_1$=1)

- **Means and variance**
  - the projection of data point $\mathbf{x}_n$ onto vector $\mathbf{u}_1$ is $\mathbf{u}_1{}^{\mathrm{T}}\mathbf{x}_n$
  - sample mean $\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$ and projection mean $\mathbf{u}_1^{\mathrm{T}}\overline{\mathbf{x}}$
  - variance of projected data

$$\frac{1}{N}\sum_{n=1}^{N}\left\{\mathbf{u}_1^{\mathrm{T}}\mathbf{x}_n - \mathbf{u}_1^{\mathrm{T}}\overline{\mathbf{x}}\right\}^2 = \mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1 \qquad \mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}}$$

- **Optimization**
  - form Lagrangian: $\mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1 + \lambda_1\left(1 - \mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1\right)$
  - set derivative to zero: $\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1$ and multiply by $\mathbf{u}_1{}^{\mathrm{T}}$: $\mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1 = \lambda_1$
  - variance maximized for $\mathbf{u}_1$ = eigenvector of largest eigenvalue

Maximum Variance Formulation

- **Generalization**
  - incrementally …
  - … choose a new orthogonal direction compared to previous
  - … maximizing the variance in the projection
- **Algorithm**
  - compute mean and covariance matrix $\mathbf{S}$ of the data
  - find the M eigenvectors of $\mathbf{S}$ from M largest eigenvalues
- **Complexity**
  - full eigenvector decomposition of $D \times D$ matrix : $O(D^3)$
  - power method (only largest M eigenvalues/vectors): $O(MD^2)$

Minimum Error Formulation

- complete, orthonormal *D*-dimensional basis $\{\mathbf{u}_i\}$ ($\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_j=\delta_{ij}$)

- **Projection**
  - data can be re-written as $\mathbf{x}_n = \sum_{i=1}^{D} \alpha_{ni}\mathbf{u}_i$ where $\alpha_{nj} = \mathbf{x}_n^{\mathrm{T}}\mathbf{u}_j$
  - approximate the full projection using only first *M* dimensions

$$\mathbf{x}_n = \sum_{i=1}^{D} \left(\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_i\right)\mathbf{u}_i \qquad \widetilde{\mathbf{x}}_n = \sum_{i=1}^{M} z_{ni}\mathbf{u}_i + \sum_{i=M+1}^{D} b_i\mathbf{u}_i$$

  - $\{z_{ni}\}$ for each data point, $\{b_i\}$ constants common for all data

- **Optimization**
  - squared error: $J = \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{x}_n - \widetilde{\mathbf{x}}_n\|^2$
  - setting derivative wrt z_{nj} to zero: $z_{nj} = \mathbf{x}_n^{\mathrm{T}}\mathbf{u}_j$
  - setting derivative wrt b_j to zero: $b_j = \overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_j$

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$$

Minimum Error Formulation

- **Substitution**

$$\mathbf{x}_n - \widetilde{\mathbf{x}}_n = \sum_{i=M+1}^{D} \left\{ (\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}} \mathbf{u}_i \right\} \mathbf{u}_i$$

$$J = \frac{1}{N} \sum_{n=1}^{N} \|\mathbf{x}_n - \widetilde{\mathbf{x}}_n\|^2 = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=M+1}^{D} \left(\mathbf{x}_n^{\mathrm{T}} \mathbf{u}_i - \overline{\mathbf{x}}^{\mathrm{T}} \mathbf{u}_i \right)^2 = \sum_{i=M+1}^{D} \mathbf{u}_i^{\mathrm{T}} \mathbf{S} \mathbf{u}_i$$

  - must minimize wrt $\{\mathbf{u}_i\}$ under the orthogonality constraints

- **Optimization**
  - Lagrangian for two dimensions: $\widetilde{J} = \mathbf{u}_2^{\mathrm{T}} \mathbf{S} \mathbf{u}_2 + \lambda_2 \left(1 - \mathbf{u}_2^{\mathrm{T}} \mathbf{u}_2\right)$
  - set derivative to zero: $\mathbf{S}\mathbf{u}_2 = \lambda_2 \mathbf{u}_2$ in general: $\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i$
  - squared projection error: $J = \sum_{i=M+1}^{D} \lambda_i$
  - minimized when choosing the *D-M* smallest eigenvalues

PCA Applications

PCA Approximate Representation

- **Data approximation**
  - each data point $\mathbf{x}_n$ can be approximated as

$$\begin{aligned}\widetilde{\mathbf{x}}_n &= \sum_{i=1}^{M}(\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_i)\mathbf{u}_i + \sum_{i=M+1}^{D}(\overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_i)\mathbf{u}_i \\ &= \overline{\mathbf{x}} + \sum_{i=1}^{M}\left(\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_i - \overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_i\right)\mathbf{u}_i\end{aligned}$$

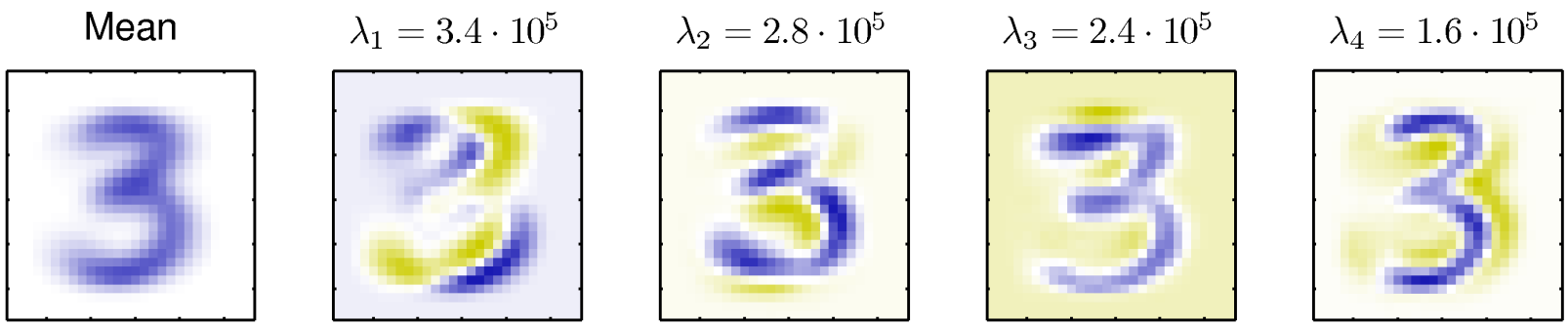
$$\boxed{\overline{\mathbf{x}} = \sum_{i=1}^{D}\left(\overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_i\right)\mathbf{u}_i}$$

  - approximate representation is *M*-dimensional
  - need to keep the mean of the data and the *M* eigenvectors
  - O(*NM*+*D*+*MD*) space for *N* data points
  - compare to original O(*ND*) space requirement

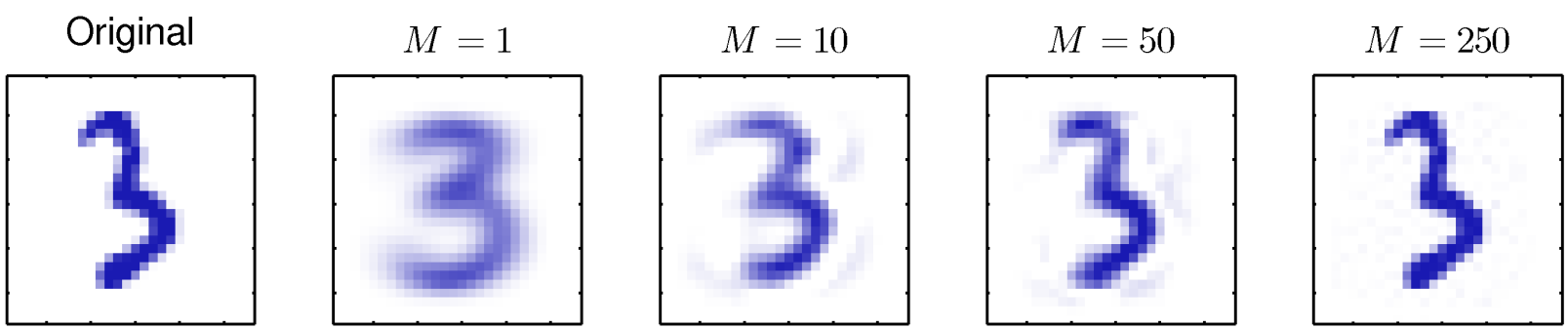
PCA Compression Example

– data set: images (28×28=768 pixels) of the handwritten digit 3

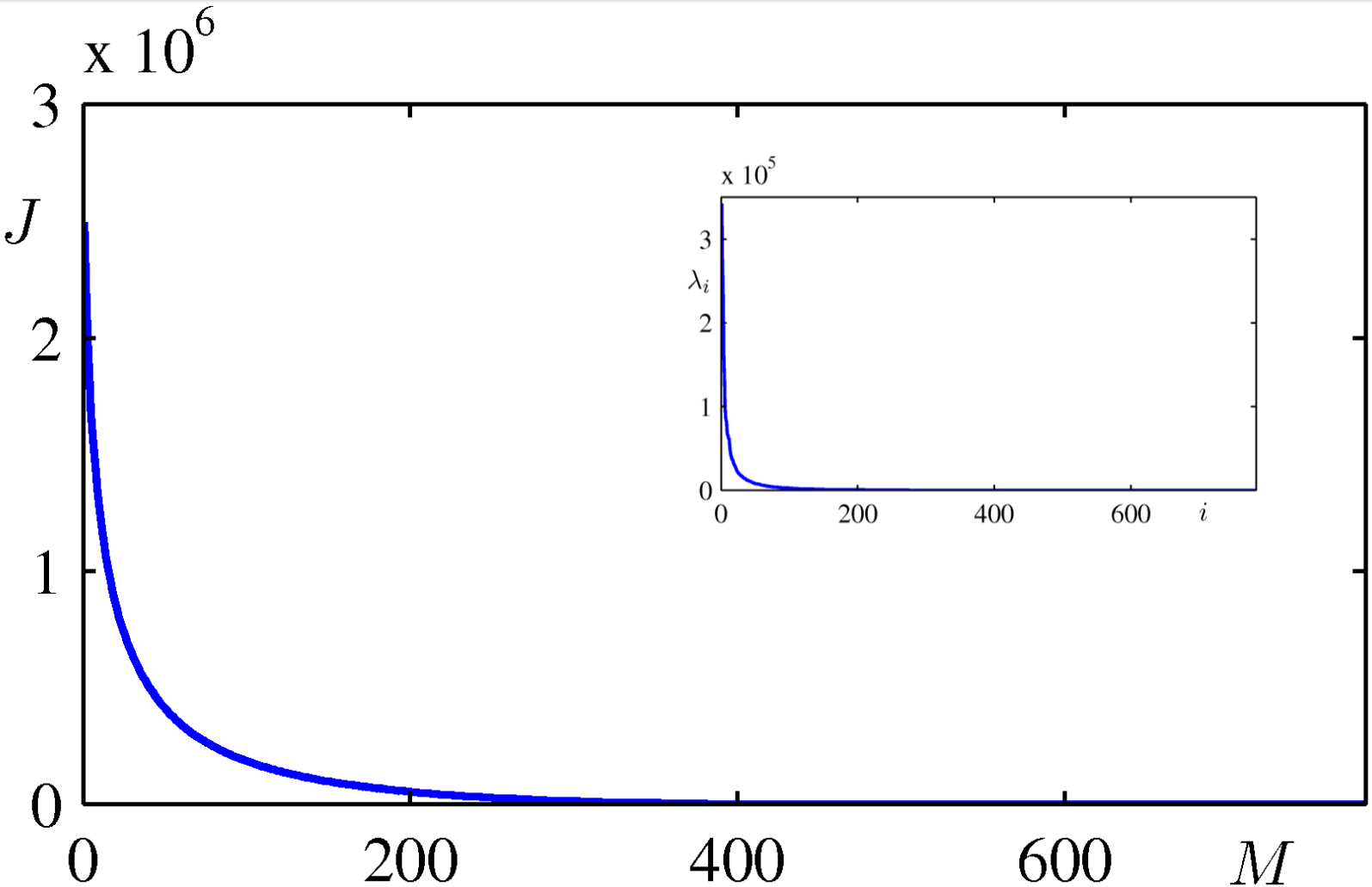
- **Principal components**



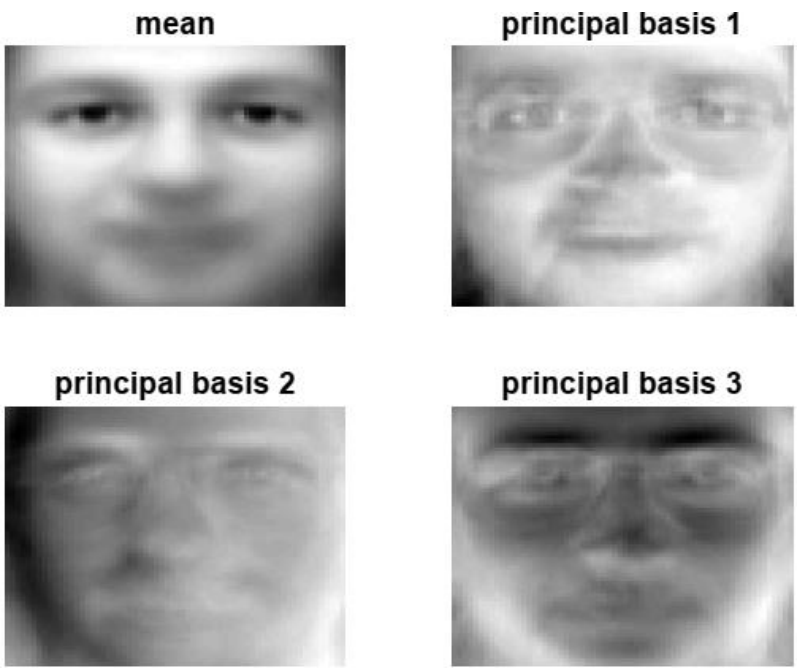
- **Reconstruction**



PCA Compression Example Error



PCA on Human Faces



PCA Data Preprocessing

- **Standardizing the data**
  - linear scaling: each variable has zero mean and unit variance
  - standardized data covariance matrix

$$\rho_{ij} = \frac{1}{N}\sum_{n=1}^{N}\frac{(x_{ni}-\overline{x}_i)}{\sigma_i}\frac{(x_{nj}-\overline{x}_j)}{\sigma_j}$$

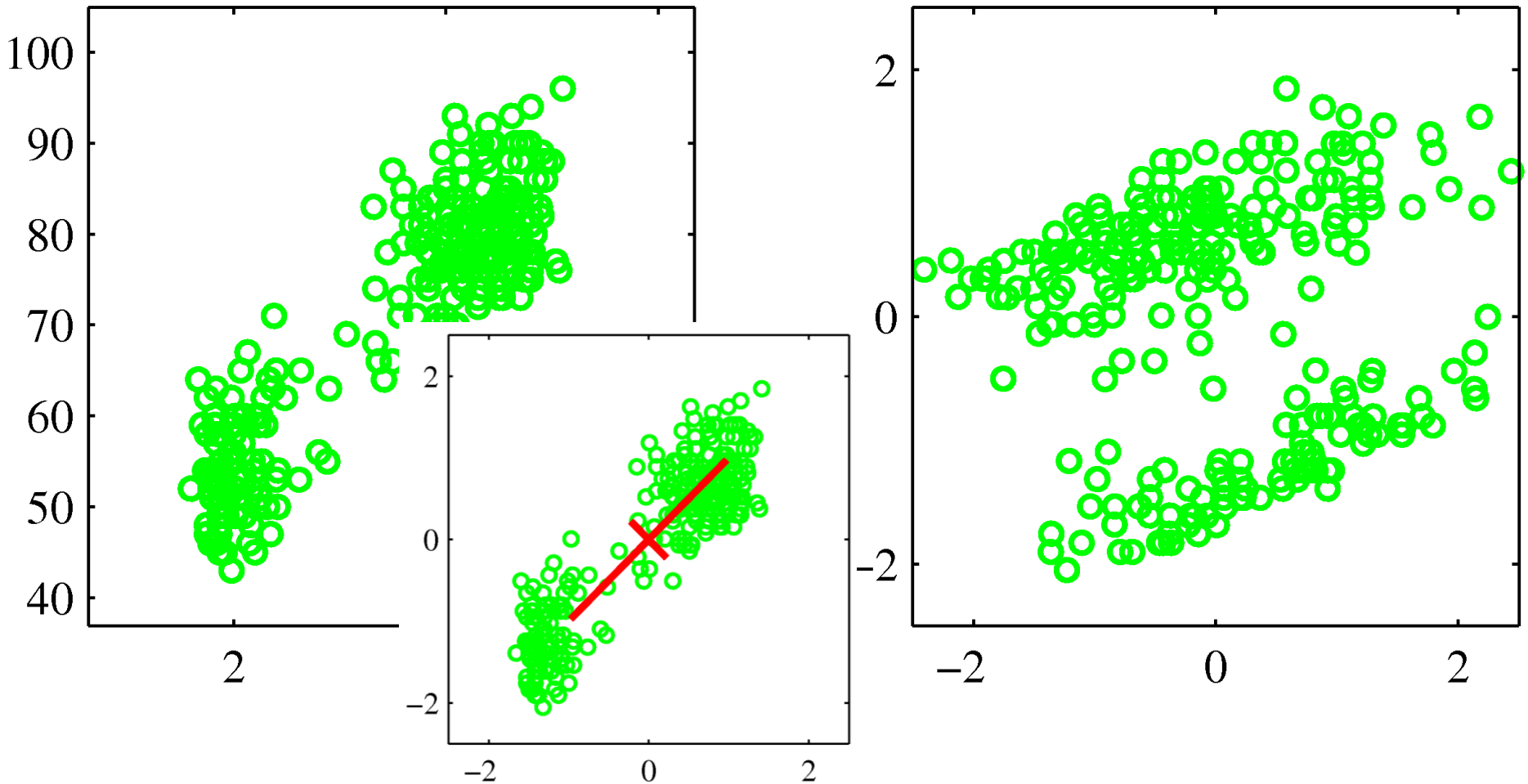
  - ρ_{ij} = 1 for perfect correlation, ρ_{ij} = 0 for no correlation
- **PCA standardization ("data whitening")**
  - standardizing data + different variables become decorrelated

$$\mathbf{SU} = \mathbf{UL}$$

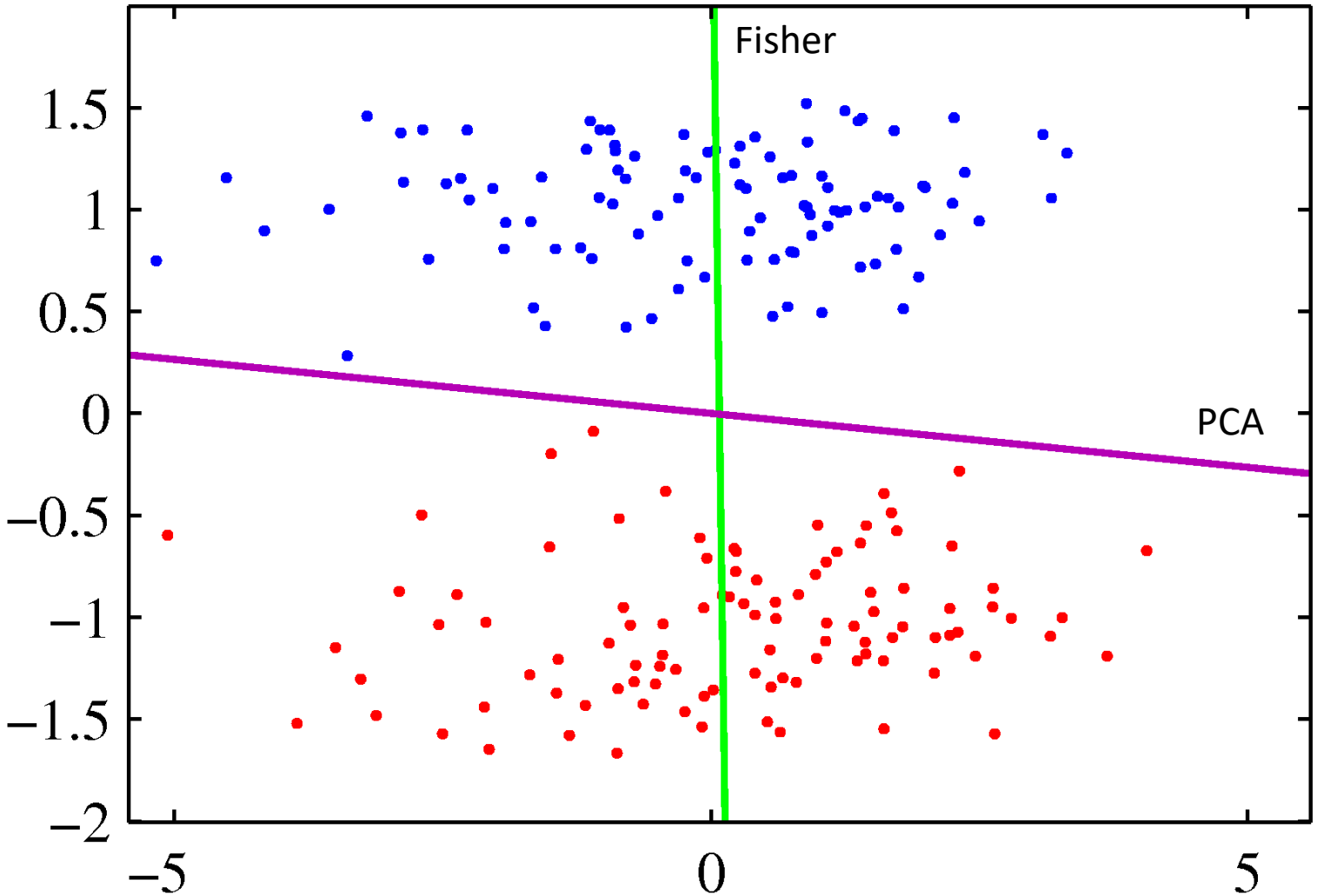
$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\mathbf{y}_n\mathbf{y}_n^{\mathrm{T}} &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{L}^{-1/2}\mathbf{U}^{\mathrm{T}}(\mathbf{x}_n-\overline{\mathbf{x}})(\mathbf{x}_n-\overline{\mathbf{x}})^{\mathrm{T}}\mathbf{U}\mathbf{L}^{-1/2} \\ &= \mathbf{L}^{-1/2}\mathbf{U}^{\mathrm{T}}\mathbf{S}\mathbf{U}\mathbf{L}^{-1/2} = \mathbf{L}^{-1/2}\mathbf{L}\mathbf{L}^{-1/2} = \mathbf{I}.\end{aligned}$$

$$\mathbf{y}_n = \mathbf{L}^{-1/2}\mathbf{U}^{\mathrm{T}}(\mathbf{x}_n-\overline{\mathbf{x}})$$

PCA Data Preprocessing Example



PCA vs. Fisher's Discriminant



PCA for High-Dimensional Data

PCA for High-Dimensional Data

- **Less data than dimensions**
  - data sets with fewer data points (N) than dimensions (D)
  - example: a few hundred high-resolution color images
  - $N<D$ defines a linear subspace with dimensionality at most N-1
  - in this case, at least D-N+1 eigenvalues are zero!
  - but … the naïve computational cost of $O(D^3)$ is prohibitive!
- **Reducing complexity**
  - define $N \times D$ matrix $\mathbf{X}$ with rows $(\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}}$
  - covariance matrix $\mathbf{S} = N^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{X}$ $\qquad \mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}}$
  - eigenvector equation $\frac{1}{N}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{u}_i = \lambda_i \mathbf{u}_i \Longrightarrow \frac{1}{N}\mathbf{X}\mathbf{X}^{\mathrm{T}}(\mathbf{X}\mathbf{u}_i) = \lambda_i(\mathbf{X}\mathbf{u}_i)$

PCA for High-Dimensional Data

$$\frac{1}{N}\mathbf{X}\mathbf{X}^{\mathrm{T}}(\mathbf{X}\mathbf{u}_i) = \lambda_i(\mathbf{X}\mathbf{u}_i)$$

- **Subspace**
  - define $\mathbf{v}_i = \mathbf{X}\mathbf{u}_i$
  - then $\frac{1}{N}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{v}_i = \lambda_i\mathbf{v}_i$
  - eigenvector equation for the $N\times N$ matrix $N^{-1}\mathbf{X}\mathbf{X}^{\mathrm{T}}$
  - retrieve the original eigenvectors $\boxed{\left(\frac{1}{N}\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)}\boxed{(\mathbf{X}^{\mathrm{T}}\mathbf{v}_i)} = \lambda_i\boxed{(\mathbf{X}^{\mathrm{T}}\mathbf{v}_i)}$
    - multiply from left with $\mathbf{X}^{\mathrm{T}}$
  - also, need to normalize to unit length $\mathbf{u}_i = \frac{1}{(N\lambda_i)^{1/2}}\mathbf{X}^{\mathrm{T}}\mathbf{v}_i$
- **Approach**
  - find eigenvalues and eigenvectors of matrix $N^{-1}\mathbf{X}\mathbf{X}^{\mathrm{T}}$
  - retrieve the eigenvectors of matrix $\mathbf{S}$
  - computational cost $O(N^3)$ instead of $O(D^3)$

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 19

Probabilistic PCA

Kernel PCA

TUC ECE, Spring 2023

Today

- **Probabilistic PCA**
- **Bayesian PCA**
- **Kernel PCA**

Probabilistic PCA

Probabilistic vs. Conventional PCA

- **Conventional PCA**
  - linear projection of data onto subspace of lower dimensionality
- **Probabilistic PCA**
  - maximum likelihood solution of a probabilistic latent model
- **Advantages of probabilistic PCA**
  - can be viewed as a constrained form of Gaussian distribution
  - a computationally efficient EM algorithm can be derived
  - missing values in the data set can be treated
  - mixtures of probabilistic PCA can be formulated
  - can be used to model class-conditional densities (classification)
  - can be used as a generative model to provide samples

Probabilistic PCA Formulation

- **Assumption**
  - all marginal and conditional distributions are Gaussian
- **Principal subspace**
  - define latent variable $\mathbf{z}$ for the principal component subspace
  - define a Gaussian prior $p(\mathbf{z})$ over the latent variable $\mathbf{z}$

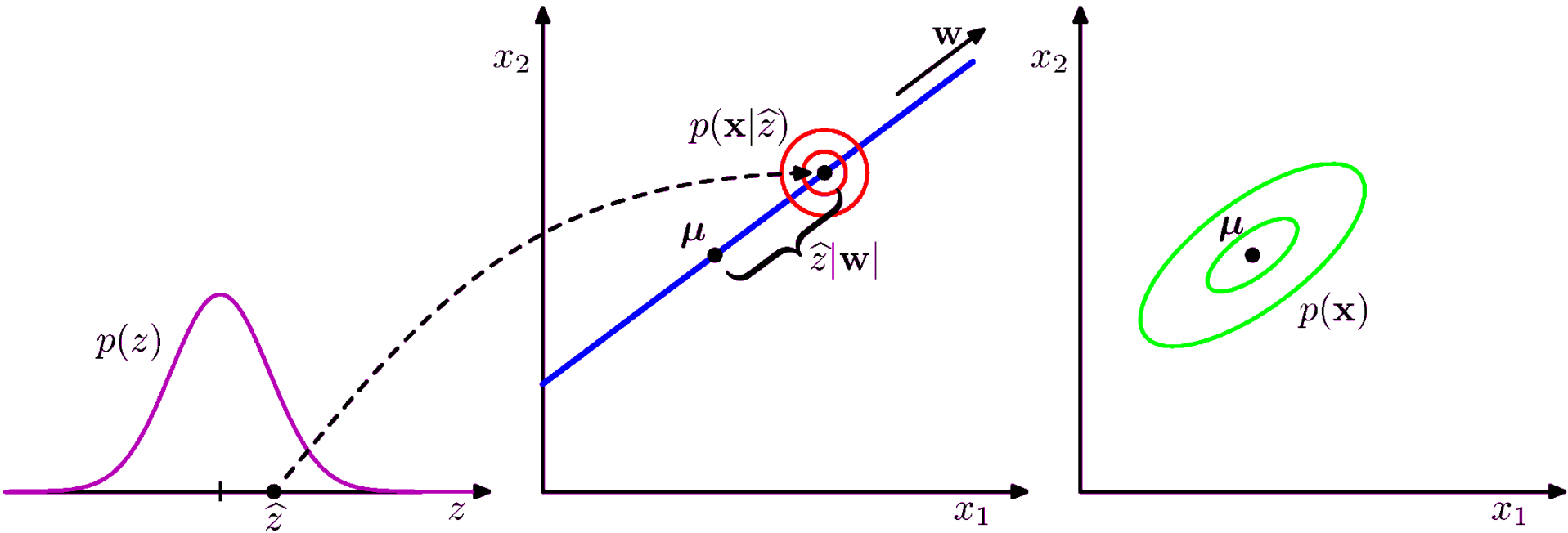
$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$$

  - define a Gaussian conditional $p(\mathbf{x}|\mathbf{z})$ for observed variable $\mathbf{x}$

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I})$$

  - the columns of $\mathbf{W}$ define the principal linear subspace
  - the scalar σ^2 governs the variance of the conditional
  - generative view (*D*-dim $\mathbf{x}$ from *M*-dim $\mathbf{z}$): $\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}$

Probabilistic PCA Illustration



Recall Bayes' Theorem for Gaussians

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y})$$

- **Given**

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right) \\ p(\mathbf{y}|\mathbf{x}) &= \mathcal{N}\left(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}\right) \end{aligned}$$

- **we have**

$$\begin{aligned} p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \\ p(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \end{aligned}$$

- **where**

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}$$

Bayes' Theorem Application

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z}|\mathbf{x})p(\mathbf{x}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$$

- **Given**

$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \\ p(\mathbf{x}|\mathbf{z}) &= \mathcal{N}\left(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I}\right) \end{aligned}$$

- **we have**

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{0} + \boldsymbol{\mu}, \sigma^2\mathbf{I} + \mathbf{W}\mathbf{I}\mathbf{W}^{\mathrm{T}}) \\ p(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z}|\boldsymbol{\Sigma}\{\mathbf{W}^{\mathrm{T}}\sigma^{-2}\mathbf{I}(\mathbf{x} - \boldsymbol{\mu}) + \mathbf{I}\mathbf{0}\}, \boldsymbol{\Sigma}) \end{aligned}$$

- **where**

$$\boldsymbol{\Sigma} = (\mathbf{I} + \mathbf{W}^{\mathrm{T}}\sigma^{-2}\mathbf{I}\mathbf{W})^{-1}$$

Bayes' Theorem Application

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z}|\mathbf{x})p(\mathbf{x}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$$

- **Given**

$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \\ p(\mathbf{x}|\mathbf{z}) &= \mathcal{N}\left(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I}\right) \end{aligned}$$

- **we have**

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \sigma^2\mathbf{I} + \mathbf{W}\mathbf{W}^{\mathrm{T}}) \\ p(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z}|\sigma^{-2}\boldsymbol{\Sigma}\mathbf{W}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}), \boldsymbol{\Sigma}) \end{aligned}$$

- **where**

$$\boldsymbol{\Sigma} = (\mathbf{I} + \sigma^{-2}\mathbf{W}^{\mathrm{T}}\mathbf{W})^{-1}$$

Towards Maximum Likelihood

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

- **Predictive distribution**
  - marginal of the observed variable $\mathbf{x}$: $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\,\mathrm{d}\mathbf{z}$
  - Gaussian due to assumptions

$$\mathbb{E}[\mathbf{x}] = \mathbb{E}[\mathbf{W}\mathbf{z}+\boldsymbol{\mu}+\boldsymbol{\epsilon}] = \boldsymbol{\mu}$$
$$\mathrm{cov}[\mathbf{x}] = \mathbb{E}\left[(\mathbf{W}\mathbf{z}+\boldsymbol{\epsilon})(\mathbf{W}\mathbf{z}+\boldsymbol{\epsilon})^{\mathrm{T}}\right] = \mathbb{E}\left[\mathbf{W}\mathbf{z}\mathbf{z}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\right] + \mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^{\mathrm{T}}] = \mathbf{W}\mathbf{W}^{\mathrm{T}}+\sigma^2\mathbf{I}$$

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\mathbf{C}) \qquad \mathbf{C} = \mathbf{W}\mathbf{W}^{\mathrm{T}}+\sigma^2\mathbf{I}$$

  - rotation redundancy

$$\widetilde{\mathbf{W}} = \mathbf{W}\mathbf{R} \qquad \mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I} \qquad \widetilde{\mathbf{W}}\widetilde{\mathbf{W}}^{\mathrm{T}} = \mathbf{W}\mathbf{R}\mathbf{R}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}} = \mathbf{W}\mathbf{W}^{\mathrm{T}}$$

  - inversion $(\mathbf{A}+\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}+\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}$

$$\mathbf{C}^{-1} = \sigma^{-2}\mathbf{I}-\sigma^{-2}\mathbf{W}\mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}} \qquad \mathbf{M} = \mathbf{W}^{\mathrm{T}}\mathbf{W}+\sigma^2\mathbf{I}$$

- **Posterior distribution**
  - Gaussian $p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\sigma^{-2}\boldsymbol{\Sigma}\mathbf{W}^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu}),\boldsymbol{\Sigma}) \quad \boldsymbol{\Sigma} = (\mathbf{I}+\sigma^{-2}\mathbf{W}^{\mathrm{T}}\mathbf{W})^{-1} = \sigma^2\mathbf{M}^{-1}$

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu}),\sigma^2\mathbf{M}^{-1})$$

posterior covariance independent of x

Maximum Likelihood PCA

- data set $\mathbf{X} = \{\mathbf{x}_n\}$ of observed data points

$$\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}, \mathbf{C}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\mathbf{C}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{x}_n - \boldsymbol{\mu})\right\}$$

- **Log likelihood**

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \mathbf{W}, \sigma^2) = \sum_{n=1}^{N} \ln p(\mathbf{x}_n|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) = \sum_{i=1}^{N} \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}, \mathbf{C})$$

$$= -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\mathbf{C}| - \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{x}_n - \boldsymbol{\mu})$$

- **Optimization**
  - setting the derivative wrt $\boldsymbol{\mu}$ to 0: $\boldsymbol{\mu}_{\mathrm{ML}} = \bar{\mathbf{x}}$ $\quad \bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$
  - backsubstitution: $\ln p(\mathbf{X}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) = -\frac{N}{2}\left\{D\ln(2\pi) + \ln|\mathbf{C}| + \mathrm{Tr}\left(\mathbf{C}^{-1}\mathbf{S}\right)\right\}$
  - setting the derivative wrt $\mathbf{W}$ to 0: $\mathbf{W}_{\mathrm{ML}} = \mathbf{U}_M(\mathbf{L}_M - \sigma^2\mathbf{I})^{1/2}\mathbf{R}$
    - $\mathbf{L}_M$: $M\times M$ max eigenvalues of $\mathbf{S}$, $\mathbf{U}_M$: $D\times M$ eigenvectors, $\mathbf{R}$: $M\times M$ rotation
  - setting the derivative wrt σ^2 to 0: $\sigma^2_{\mathrm{ML}} = \frac{1}{D-M}\sum_{i=M+1}^{D}\lambda_i$
    - average variance of discarded dimensions

Observations

- **Rotational invariance**
  - the predictive density is independent of latent space rotations
  - for $\mathbf{R}$=$\mathbf{I}$, columns of $\mathbf{W}$ are eigenvectors scaled by $\sqrt{\lambda_i - \sigma^2}$
  - additive variances, for convolution of independent Gaussians
  - variance in direction of eigenvector $\mathbf{u}_i$ has two contributions
    - a component from the projection of a unit-variance latent subspace
    - an isotropic contribution added to all directions
- **Data covariance** $\mathbf{C} = \mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I}$ $\mathbf{W}_{\mathrm{ML}} = \mathbf{U}_M(\mathbf{L}_M - \sigma^2\mathbf{I})^{1/2}\mathbf{R}$
  - variance of predictive distribution along unit vector $\mathbf{v}$: $\mathbf{v}^{\mathrm{T}}\mathbf{C}\mathbf{v}$
    - orthogonal to latent subspace: $\mathbf{v}^{\mathrm{T}}\mathbf{U} = \mathbf{0}$ $\mathbf{v}^{\mathrm{T}}\mathbf{C}\mathbf{v} = \sigma^2$
    - parallel to retained eigenvector: $\mathbf{v} = \mathbf{u}_i$ $\mathbf{v}^{\mathrm{T}}\mathbf{C}\mathbf{v} = (\lambda_i - \sigma^2) + \sigma^2 = \lambda_i$
  - extreme *M*=*D*: $\mathbf{C} = \mathbf{U}(\mathbf{L} - \sigma^2\mathbf{I})^{1/2}\mathbf{R}\mathbf{R}^{\mathrm{T}}(\mathbf{L} - \sigma^2\mathbf{I})^{1/2}\mathbf{U}^{\mathrm{T}} + \sigma^2\mathbf{I} = \mathbf{U}\mathbf{L}\mathbf{U}^{\mathrm{T}} = \mathbf{S}$

Observations

- **Projection**
  - conventional PCA projects points from *D*-space to *M*-space
  - probabilistic PCA can reverse the mapping (Bayes' theorem)
  - points in data space can be summarized by
    - mean in latent space $\mathbb{E}[\mathbf{z}|\mathbf{x}] = \mathbf{M}^{-1}\mathbf{W}_{\mathrm{ML}}^{\mathrm{T}}(\mathbf{x} - \overline{\mathbf{x}})$ $\mathbf{M} = \mathbf{W}^{\mathrm{T}}\mathbf{W} + \sigma^2\mathbf{I}$
    - covariance in latent space $\sigma^2\mathbf{M}^{-1}$
  - in limit $\sigma^2 \to 0$, orthogonal projection: $(\mathbf{W}_{\mathrm{ML}}^{\mathrm{T}}\mathbf{W}_{\mathrm{ML}})^{-1}\mathbf{W}_{\mathrm{ML}}^{\mathrm{T}}(\mathbf{x} - \overline{\mathbf{x}})$
  - $\sigma^2 \to 0$: the standard PCA model, $\sigma^2 > 0$: shifted towards origin
- **Multivariate Gaussians**
  - in general, *D* parameters for mean, *D*(*D*+1)/2 for covariance
  - full independence: *D* for mean, *D* for covariance, but loss!
  - probabilistic PCA: *DM* + 1 – *M*(*M*-1)/2 degrees of freedom

Bayesian PCA

Choosing *M*

- **Ad-hoc**
  - M=2 for visualization
  - check the eigenvalue spectrum and choose cut-off
- **Cross validation**
  - try various values of M and evaluate using cross validation
  - select M giving the largest log likelihood on validation set
- **Bayesian approach**
  - marginalize out the model parameters $\boldsymbol{\mu}$, $\mathbf{W}$, and σ^2
  - variational framework to approximate hard marginalizations
  - evidence approximation, for large data set and peaked posterior

Automatic Relevance Determination

- **ARD prior**
  - independent Gaussian prior over each column of $\mathbf{W}$
  - each Gaussian has its own precision hyperparameter α_i

$$p(\mathbf{W}|\boldsymbol{\alpha}) = \prod_{i=1}^{M} \left(\frac{\alpha_i}{2\pi}\right)^{D/2} \exp\left\{-\frac{1}{2}\alpha_i \mathbf{w}_i^{\mathrm{T}} \mathbf{w}_i\right\}$$

- **Optimization**
  - values of α_i are found iteratively
  - maximizing marginal likelihood after integrating $\mathbf{W}$ out
  - most α_i are driven to infinity and corresponding $\mathbf{w}_i$ to zero
  - sparsity: number of finite-valued α_i gives value for *M*

Laplace Approximation

- **Marginal likelihood**

$$p(\mathbf{X}|\boldsymbol{\alpha}, \boldsymbol{\mu}, \sigma^2) = \int p(\mathbf{X}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) p(\mathbf{W}|\boldsymbol{\alpha}) \, \mathrm{d}\mathbf{W}$$

  - no prior for $\boldsymbol{\mu}$ and σ^2, parameters to be estimated

- **Iterative optimization (EM)**

$$\begin{aligned} \mathbb{E}[\mathbf{z}_n] &= \mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}}(\mathbf{x}_n - \overline{\mathbf{x}}) \\ \mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}] &= \sigma^2\mathbf{M}^{-1} + \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^{\mathrm{T}} \end{aligned}$$

E-step

$$\alpha_i^{\mathrm{new}} = \frac{D}{\mathbf{w}_{\mathrm{new}i}^{\mathrm{T}}\mathbf{w}_{\mathrm{new}i}}$$

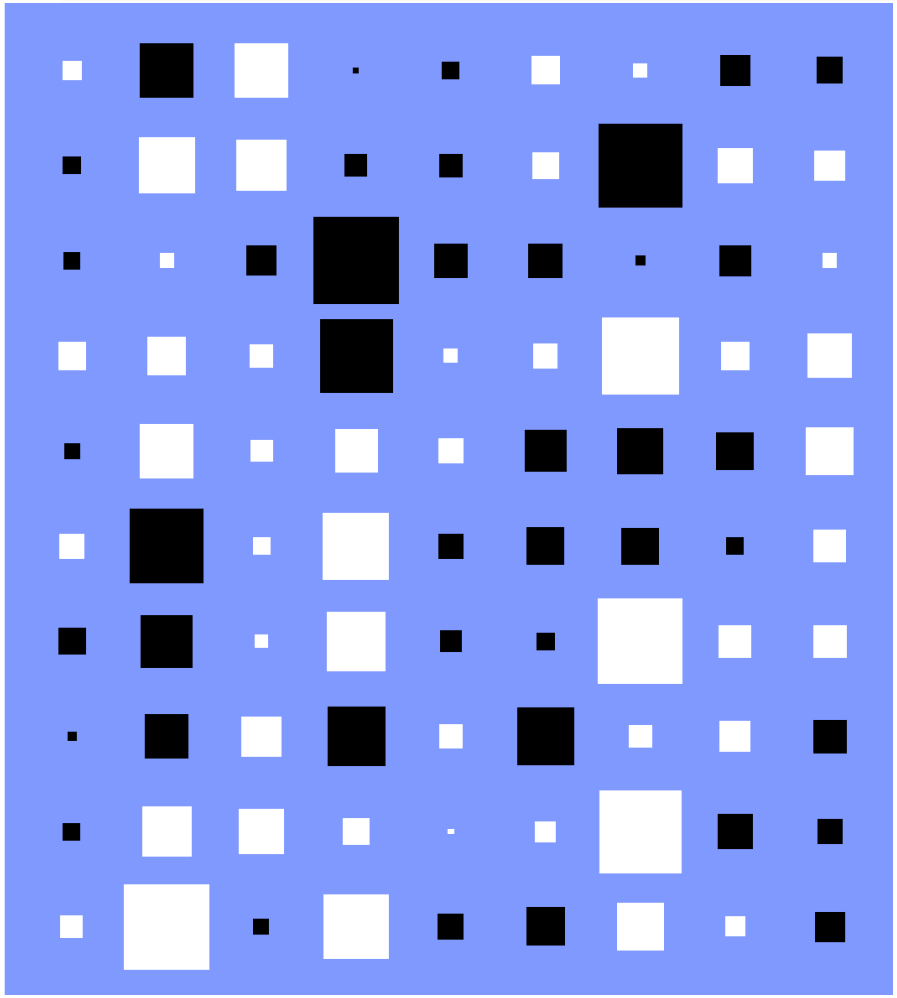
$$\mathbf{A} = \mathrm{diag}(\alpha_i)$$

$$\mathbf{W}_{\mathrm{new}} = \left[\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})\mathbb{E}[\mathbf{z}_n]^{\mathrm{T}}\right]\left[\sum_{n=1}^{N}\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}] + \sigma^2\mathbf{A}\right]^{-1}$$

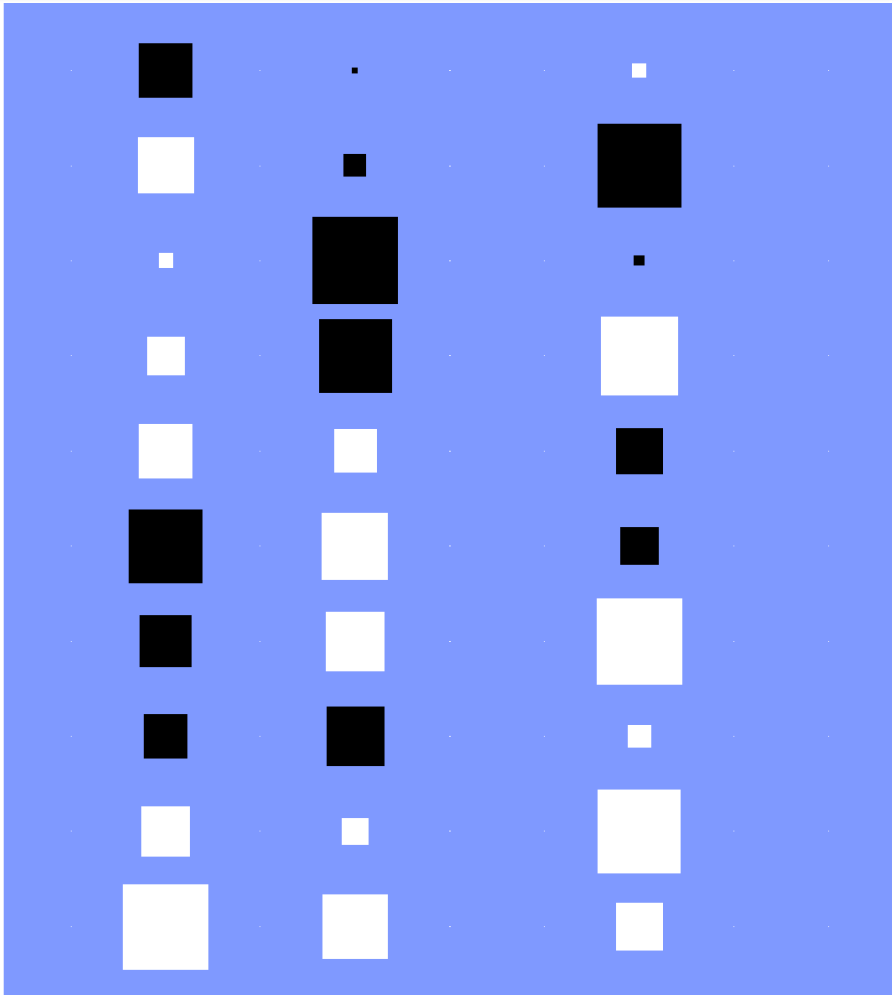
M-step

$$\sigma^2_{\mathrm{new}} = \frac{1}{ND}\sum_{n=1}^{N}\left(||\mathbf{x}_n - \overline{\mathbf{x}}||^2 - 2\mathbb{E}[\mathbf{z}_n]^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}(\mathbf{x}_n - \overline{\mathbf{x}}) + \mathrm{Tr}(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}]\mathbf{W}^{\mathrm{T}}\mathbf{W})\right)$$

Bayesian PCA Example (W)



ML probabilistic PCA



Bayesian PCA

Kernel PCA

From Conventional to Kernel PCA

- data set $\mathbf{X} = \{\mathbf{x}_n\}$ of observed data points in *D* dimensions
- assume that the sample mean of the data is zero: $\sum_n \mathbf{x}_n = \mathbf{0}$

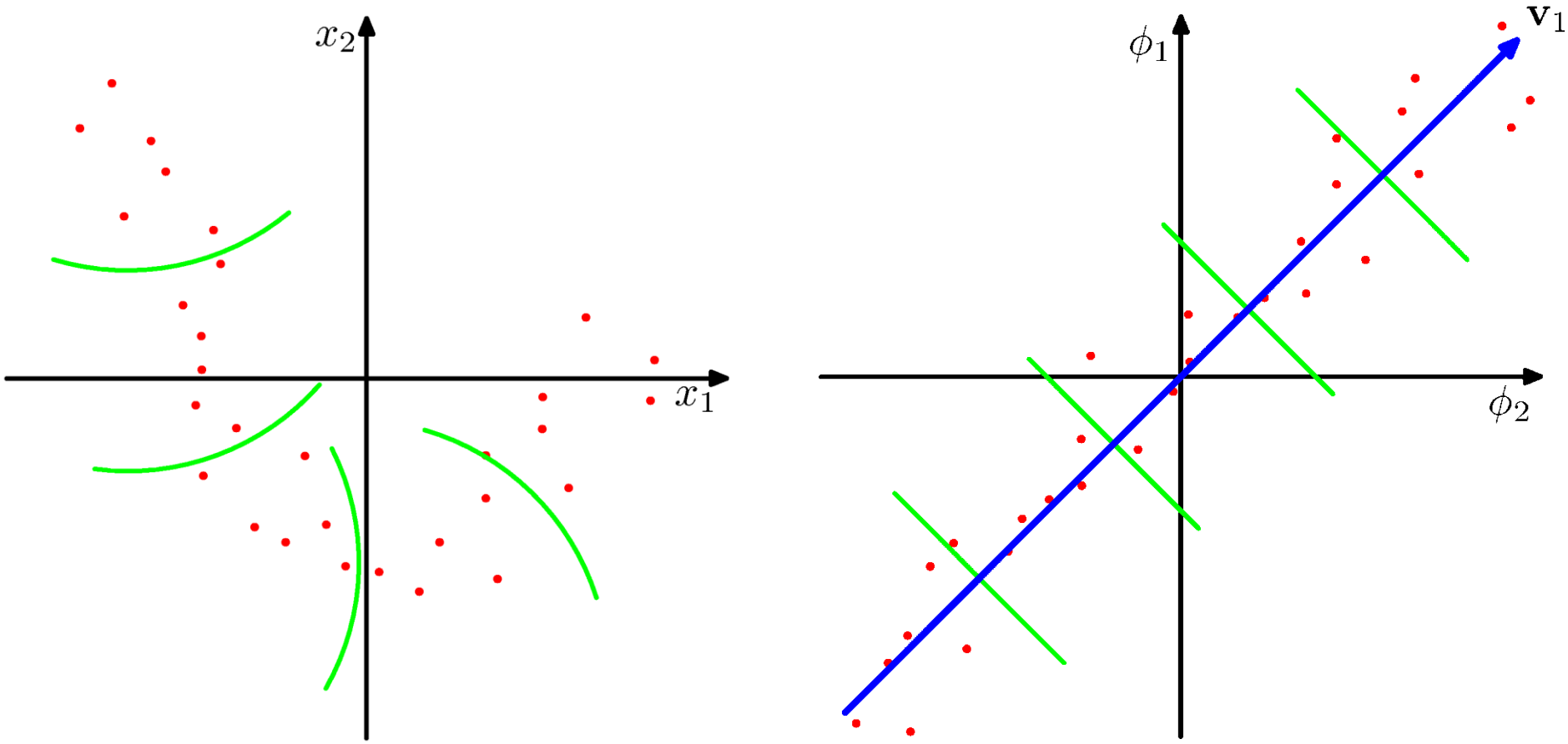
- **Conventional PCA**
  - principal components defined by the eigenvectors $\mathbf{u}_i$ of $\mathbf{S}$

$$\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_i = 1 \qquad \mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i \qquad \mathbf{S} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}}$$

- **Kernel PCA**
  - non-linear transformation $\phi(\mathbf{x})$ to *M*-dimensional feature space
  - assume that sample mean of projection is zero: $\sum_n \phi(\mathbf{x}_n) = \mathbf{0}$
  - standard PCA in feature gives nonlinear components in input

$$\mathbf{v}_i^{\mathrm{T}}\mathbf{v}_i = 1 \qquad \mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i \qquad \mathbf{C} = \frac{1}{N}\sum_{n=1}^{N} \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^{\mathrm{T}}$$

Kernel PCA Illustration



Kernelization

$$\mathbf{v}_i^{\mathrm{T}}\mathbf{v}_i = 1 \qquad \mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i \qquad \mathbf{C} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}$$

- the *M* eigenvectors $\mathbf{v}_i$ of $\mathbf{C}$

$$\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n)\left\{\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\mathbf{v}_i\right\} = \lambda_i \mathbf{v}_i \qquad \mathbf{v}_i = \sum_{n=1}^{N} a_{in}\boldsymbol{\phi}(\mathbf{x}_n) \qquad \lambda_i > 0$$

$$\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} \sum_{m=1}^{N} a_{im}\boldsymbol{\phi}(\mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in}\boldsymbol{\phi}(\mathbf{x}_n)$$

- kernel function $k(\mathbf{x}_n, \mathbf{x}_m) = \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m)$

$$\frac{1}{N}\sum_{n=1}^{N} k(\mathbf{x}_l, \mathbf{x}_n) \sum_{m=1}^{m} a_{im} k(\mathbf{x}_n, \mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in} k(\mathbf{x}_l, \mathbf{x}_n)$$

- *N*×*N* Gram matrix $\mathbf{K}$ $\qquad \mathbf{K}^2\mathbf{a}_i = \lambda_i N \mathbf{K}\mathbf{a}_i$
- vectors $\mathbf{a}_i$ with a_{in}, n=1,…,*N* $\qquad \mathbf{K}\mathbf{a}_i = \lambda_i N \mathbf{a}_i$

Principal Component Projection

- **Coefficient normalization**

$$1 = \mathbf{v}_i^{\mathrm{T}}\mathbf{v}_i = \sum_{n=1}^{N}\sum_{m=1}^{N} a_{in}a_{im}\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m) = \mathbf{a}_i^{\mathrm{T}}\mathbf{K}\mathbf{a}_i = \lambda_i N \mathbf{a}_i^{\mathrm{T}}\mathbf{a}_i$$

- **Projection**

$$y_i(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) = \sum_{n=1}^{N} a_{in}k(\mathbf{x},\mathbf{x}_n)$$

  - at most *D* linear principal components in input space
  - at most *M* linear principal components in feature space
  - the dimensions of feature space (*M*) may exceed *D* (infinite?)
  - the number of non-zero eigenvalues cannot exceed *N*
  - thus, at most max{*M,N*} principal components in feature space

Zero-Mean Assumption in Feature

$$\widetilde{\phi}(\mathbf{x}_n) = \phi(\mathbf{x}_n) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_l)$$

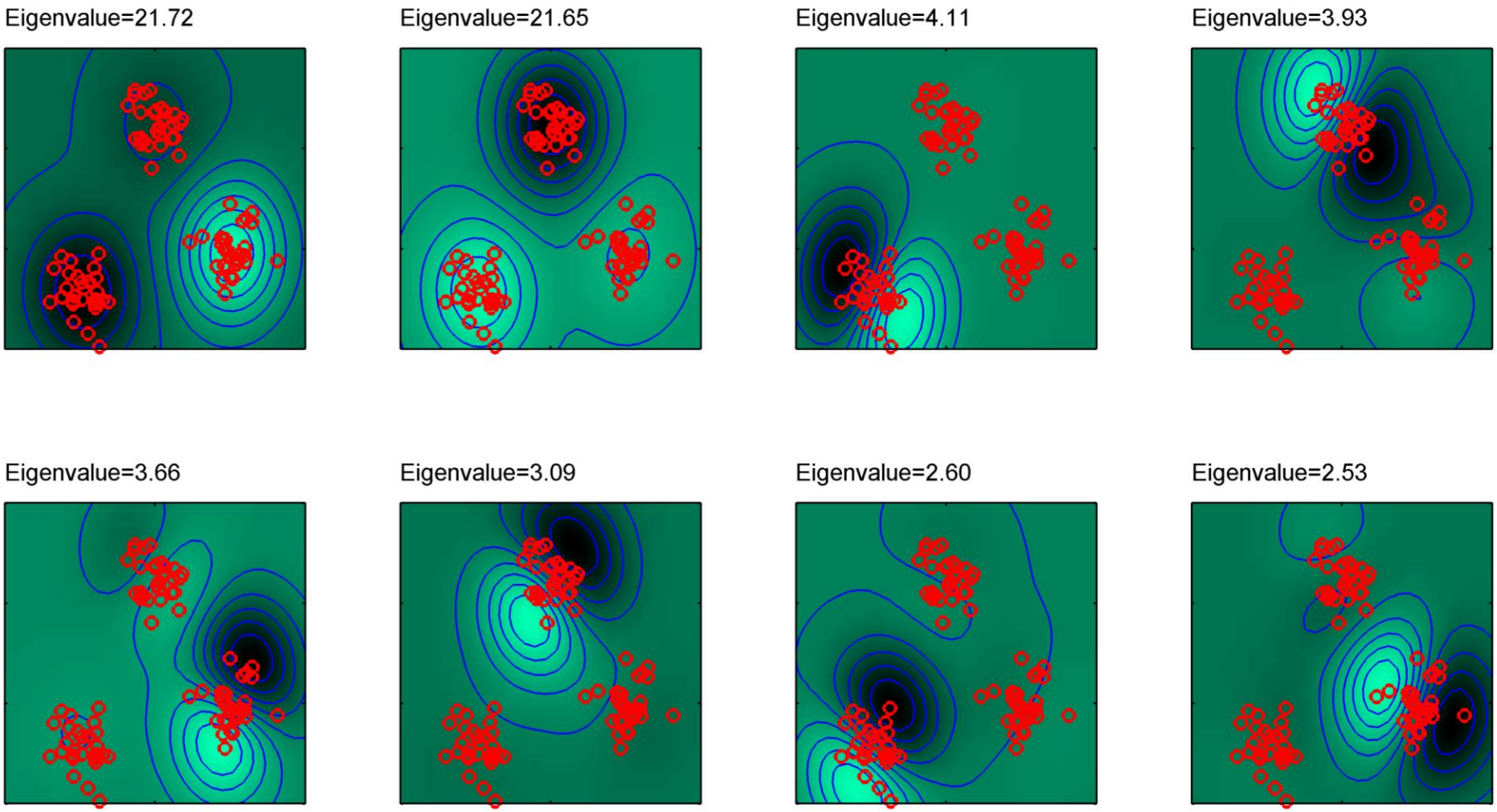
$$\begin{aligned}
\widetilde{K}_{nm} &= \widetilde{\phi}(\mathbf{x}_n)^{\mathrm{T}}\widetilde{\phi}(\mathbf{x}_m) \\
&= \phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_l) \\
&\quad -\frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_l)^{\mathrm{T}}\phi(\mathbf{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}\phi(\mathbf{x}_j)^{\mathrm{T}}\phi(\mathbf{x}_l) \\
&= k(\mathbf{x}_n, \mathbf{x}_m) - \frac{1}{N}\sum_{l=1}^{N}k(\mathbf{x}_l, \mathbf{x}_m) \\
&\quad -\frac{1}{N}\sum_{l=1}^{N}k(\mathbf{x}_n, \mathbf{x}_l) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}k(\mathbf{x}_j, \mathbf{x}_l).
\end{aligned}$$

Evaluate the new Gram matrix and perform PCA on the new Gram matrix!

$$\widetilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N\mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N\mathbf{K}\mathbf{1}_N$$

$\mathbf{1}_N$: $N \times N$ matrix with elements $1/N$

Kernel (Gaussian) PCA Example



Properties

- **Pros**
  - use of a large library of kernel functions
  - non-linear projections in input space
- **Cons**
  - performing PCA on a $N \times N$ matrix, as opposed to a $D \times D$ matrix
  - in practice, approximations are used
- **Compression?**
  - PCA: approximate with $L<D$ eigenvectors $\widehat{\mathbf{x}}_n = \sum_{i=1}^{L} \left(\mathbf{x}_n^{\mathrm{T}} \mathbf{u}_i\right) \mathbf{u}_i$
  - $\phi(\mathbf{x})$ maps $\mathbf{x}$ in D-dimensional manifold in M dimensions
  - projection of points in feature space onto linear PCA subspace may not lie on that D-dimensional manifold! no pre-image $\mathbf{x}$!

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 20

Reinforcement Learning

Decision Making under Uncertainty

TUC ECE, Spring 2023

Recall: Machine Learning

- **Supervised Learning**
  - set of training data with inputs and targets
  - classification, regression, …
- **Unsupervised Learning**
  - set of training data with inputs, but without targets
  - clustering, density estimation, dimensionality reduction, …
- **Reinforcement Learning**
  - set of training trials of interaction with feedback by a critic
  - value function, decision policy, exploration vs. exploitation, …
- **Learning Theory**
  - theoretical investigations: what can be learned? how fast?

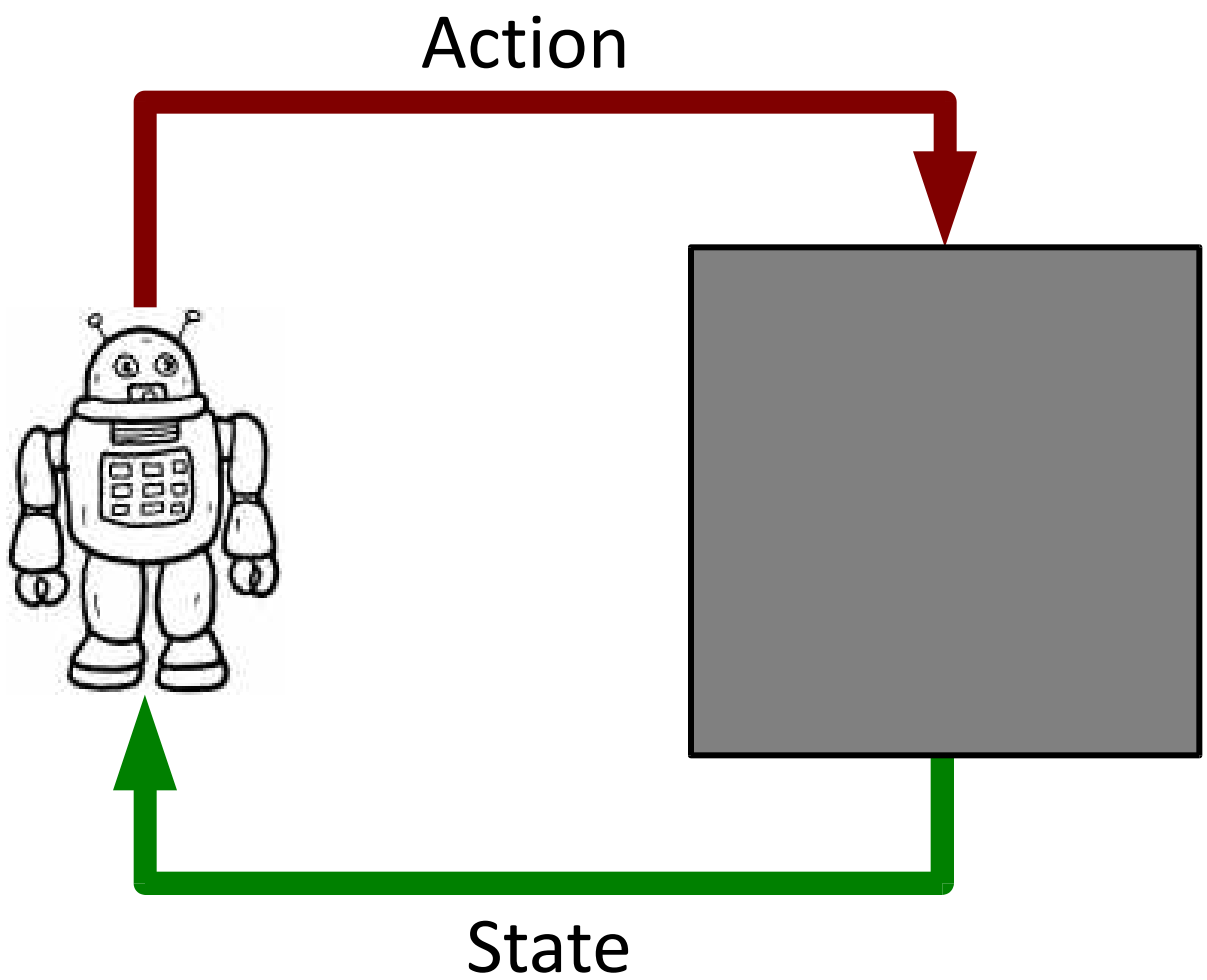
Today

- **Decision Making under Uncertainty**
  - Sequential Decision Making
  - Markov Decision Process (MDP)

Sequential Decision Making

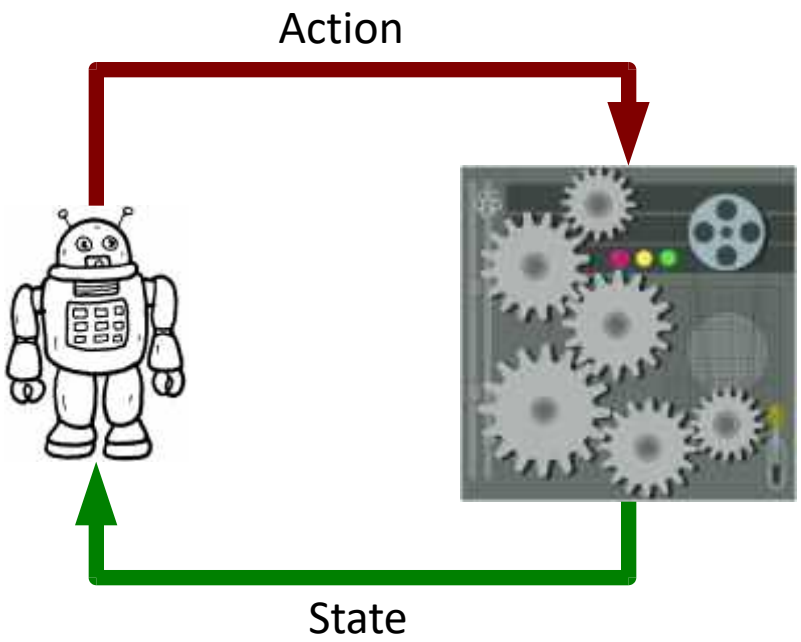
Decisions after decisions …

Sequential Decision Making



Planning or Learning?

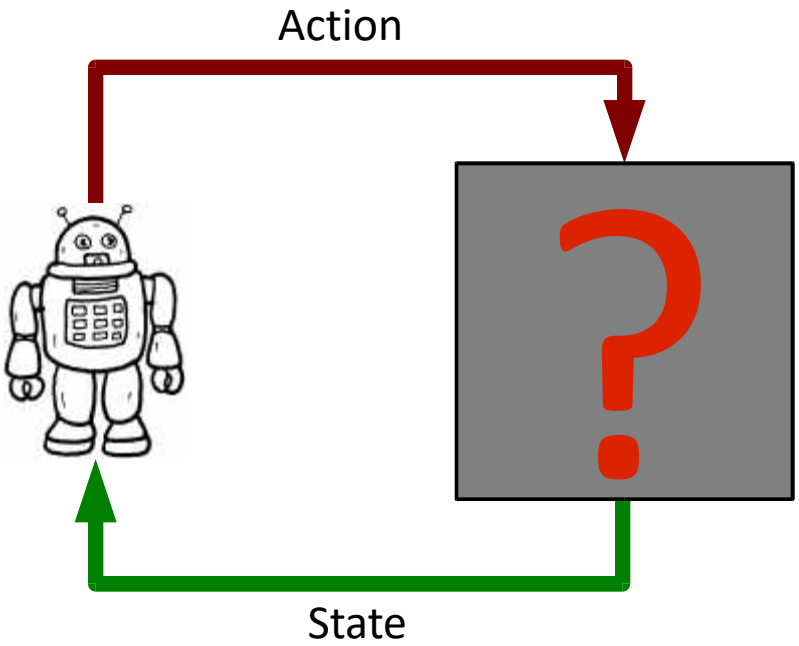
Planning



Known Model

Uncertainty **in** the process

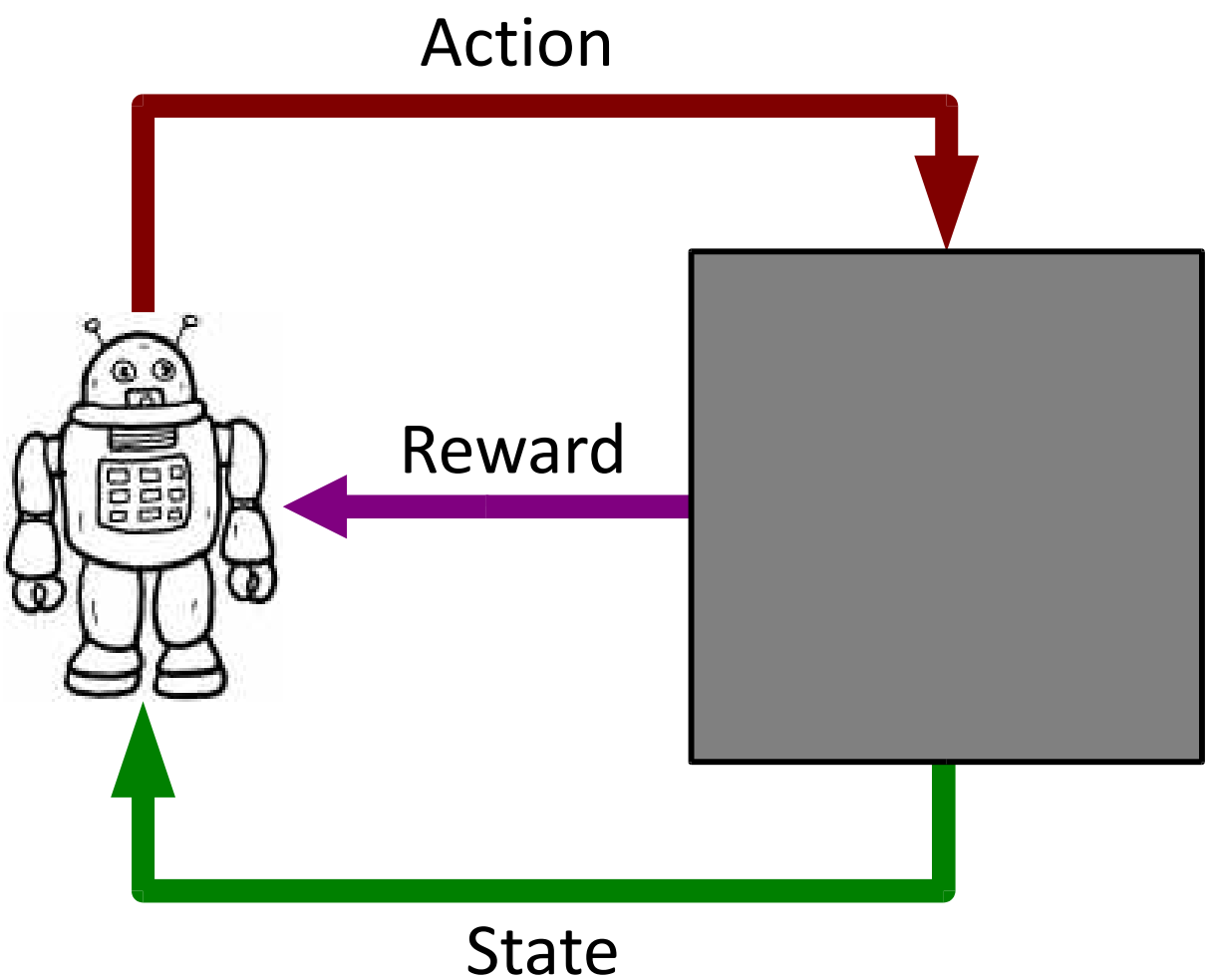
Learning



Unknown Model

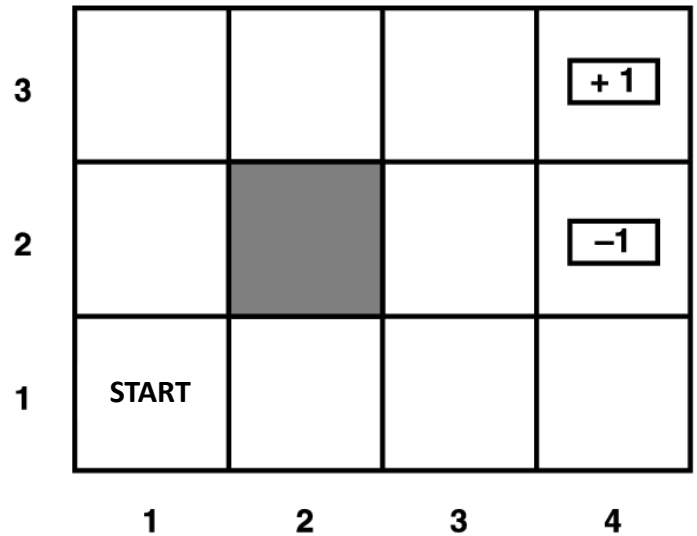
Uncertainty **in** the process
and **about** the process

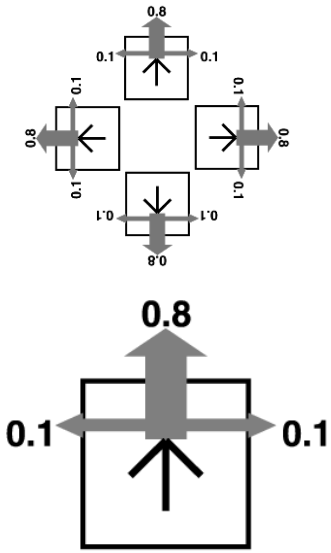
Decision Making for a Purpose



Navigation in the Grid World

- **Grid World**
  - 11 states
  - 4 actions
  - uncertainty
  - collisions
  - r = +1 terminal
  - r = –1 terminal
  - r = –0.04 / step





- **Classical planning**
  - deterministic plan: [*Up*, *Up*, *Right*, *Right*, *Right*]
  - probability of success: $0.8^5 + 0.1^4 \times 0.8 = 0.32776$

Plan Execution [↑,↑,→,→,→]

			+1
			−1
1.0			

			+1
0.8			−1
0.1	0.1		

0.64			+1
0.24			−1
0.02	0.09	0.01	

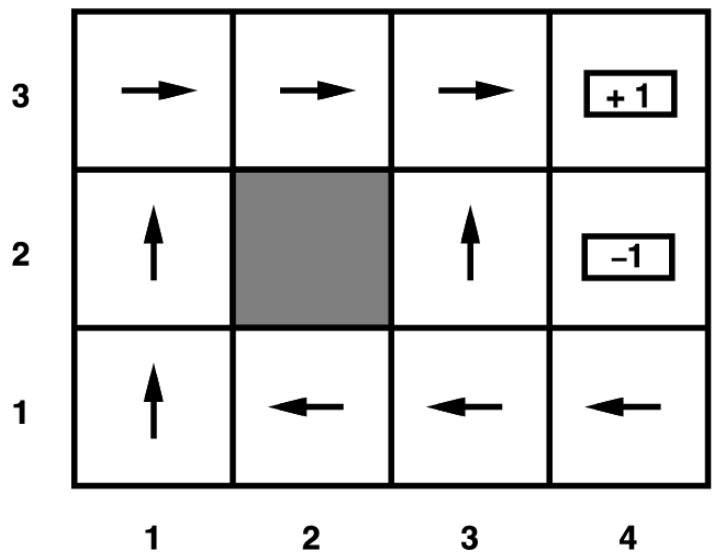
0.088	0.512		+1
0.258		0.001	−1
0.026	0.034	0.073	0.008

???	???	0.4097	+1
???		???	0.0016
???	???	???	???

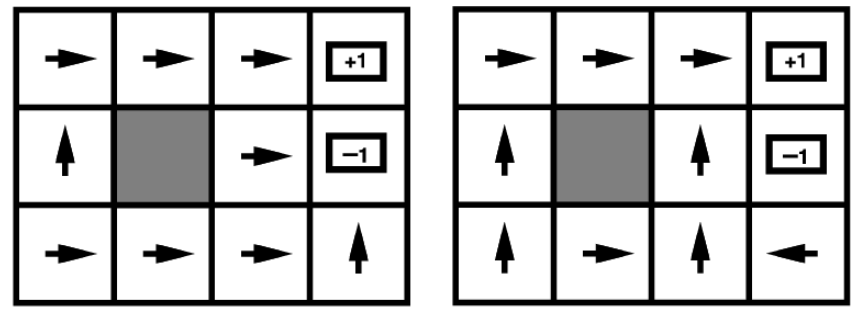
???	???	???	0.32776
???		???	???
???	???	???	???

Optimal Policies

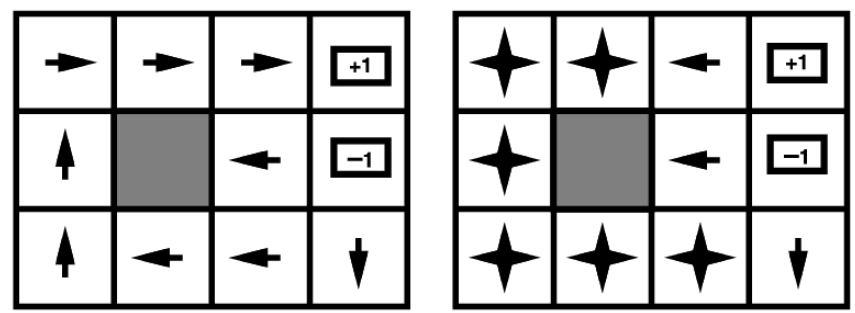
- **Policy π**
  - action choice (decision making) in every possible state
- **Optimal policy π***



$R(s)$=-0.04



$R(s) < -1.6284$ $\quad -0.4278 < R(s) < -0.0850$



$-0.0221 < R(s) < 0$ $\quad R(s) > 0$

Optimality Metrics

- **Horizon**
  - finite $\Rightarrow$ non-stationary policies
  - infinite $\Rightarrow$ stationary policies
- **Stationary metrics over state sequences**
  - additive rewards
    - $U_h([s_0, s_1, s_2 \ldots]) = R(s_0) + R(s_1) + R(s_2) + \ldots$
  - discounted rewards
    - $U_h([s_0, s_1, s_2 \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$
    - discount factor γ in (0,1]
  - average reward
    - $U_h([s_0, s_1, s_2 \ldots]) = \lim_{T \to \infty} [(R(s_0) + R(s_1) + R(s_2) + \ldots + R(s_T)) / T]$

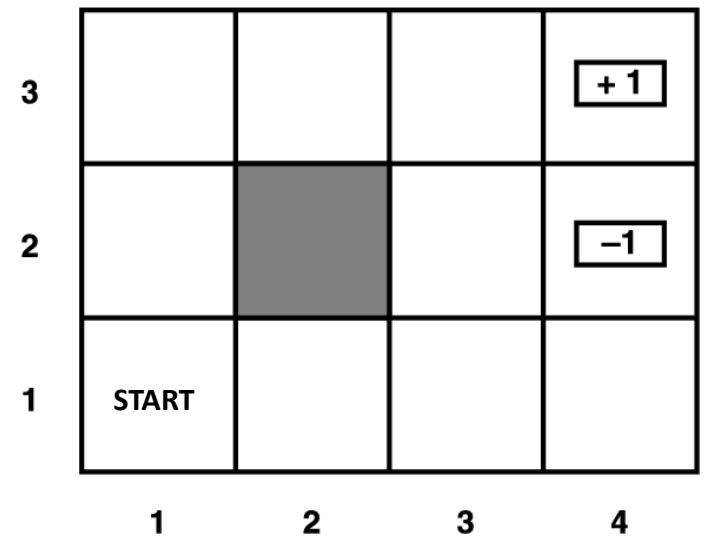
Markov Decision Process (MDP)

- **MDP (S, A, P, R, γ, D)**
  - S: state space of the process
  - A: action space of the process
  - P: transition model, $P(s' \mid s, a)$
  - R: reward model, $R(s, a)$
  - γ: discount factor, $0 < \gamma \leq 1$
  - D: initial state distribution
- **Markov property**
  - next state and reward are **independent** of history

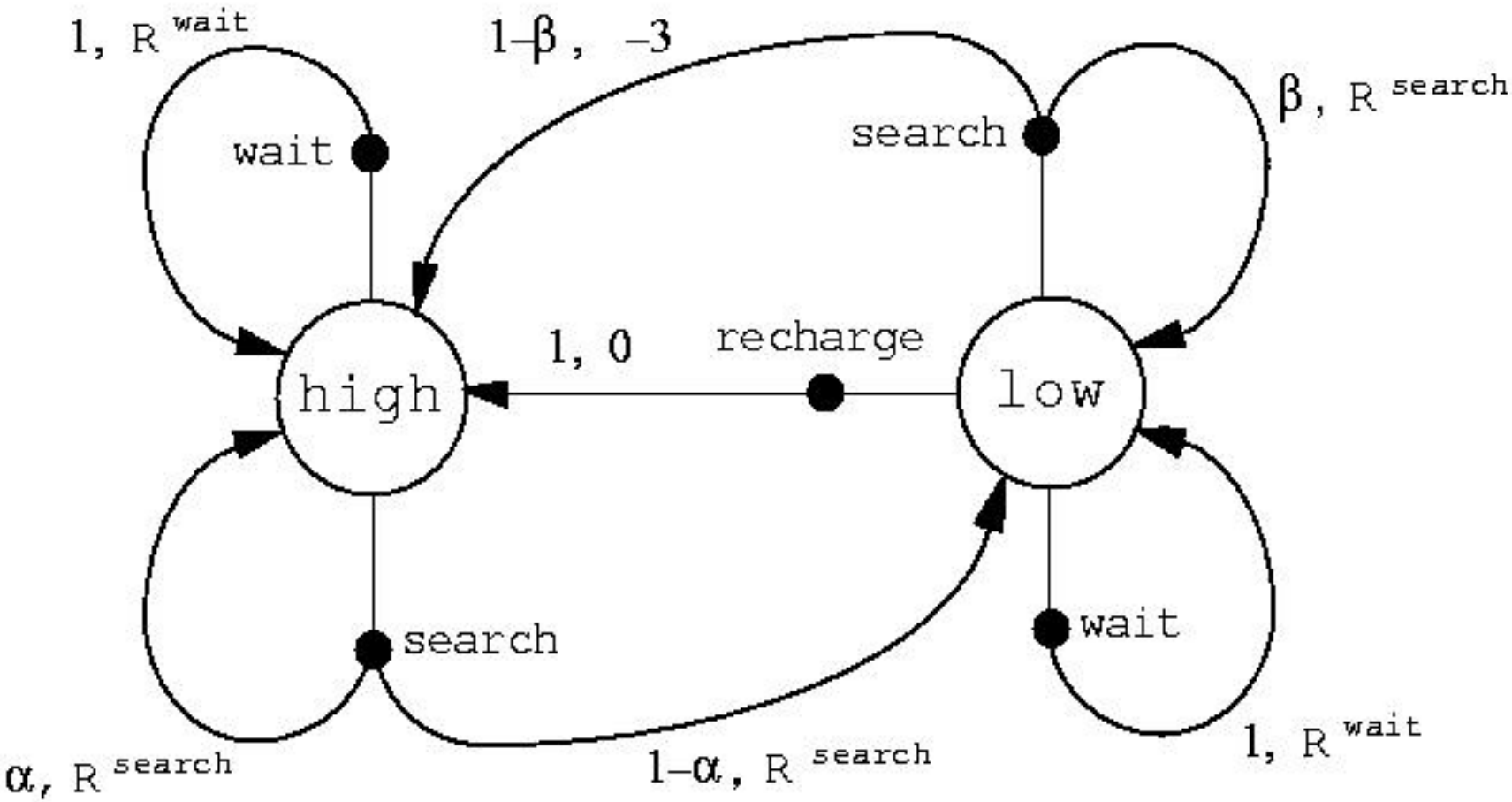
note: the reward model can also be defined as R(s) or R(s,a,s')

Simple MDP: Grid World

- **Grid World**
  - |S| = 11 (all cells except [2,2])
  - |A| = 4 ($\Uparrow, \Downarrow, \Leftarrow, \Rightarrow$)
  - P(s' | s, a)
    - non-zero probability of transition to at most 3 other states
    - zero probability to all others
  - R(s, a)
    - R = +1 or R = –1 in terminal states independently of the action
    - R = –0,04 in all other states independently of the action
  - $\gamma = 1$
  - D : probability 1 for [1,1], probability 0 for all others

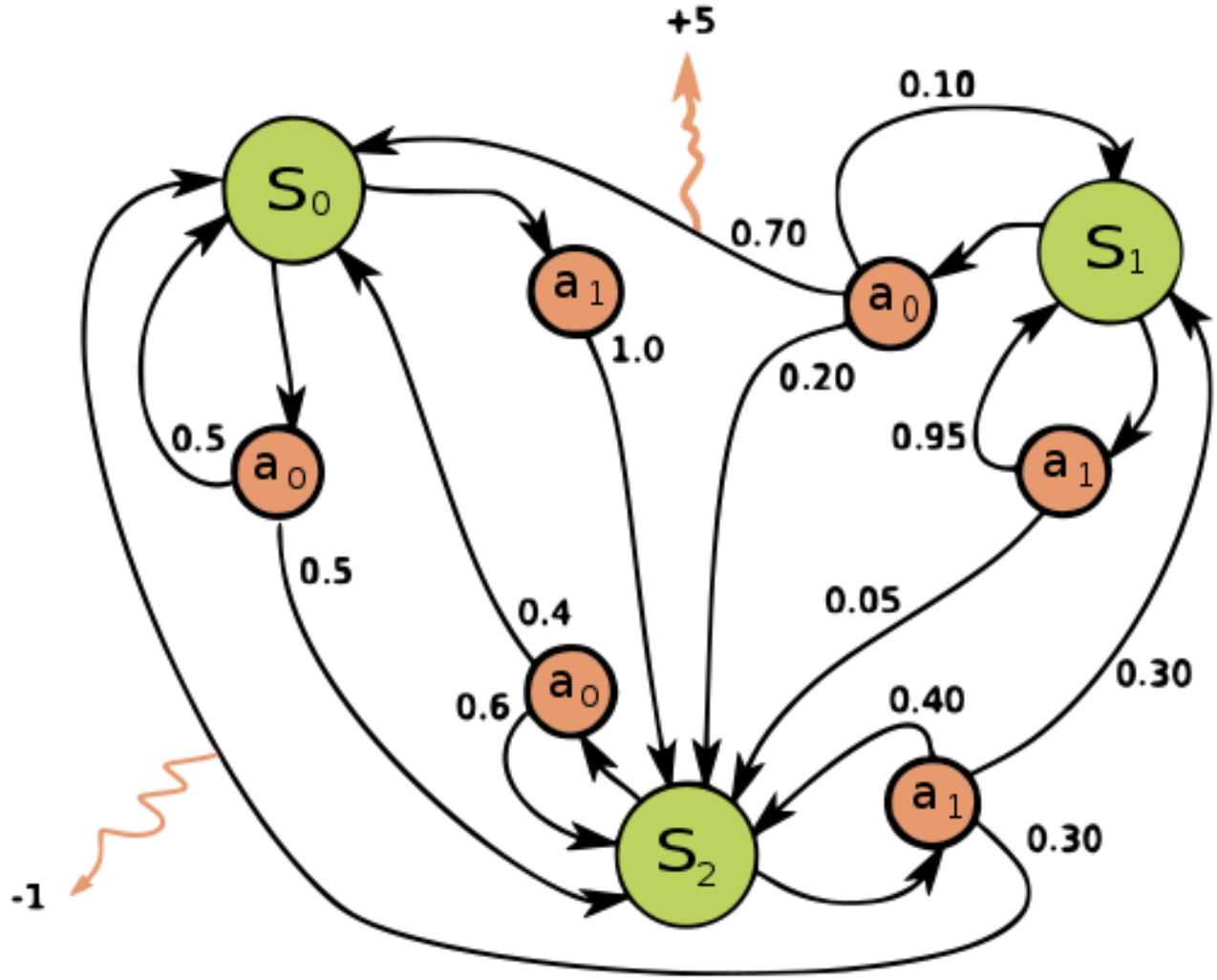


Simple MDP: Recycling Robot



[Sutton and Barto, 1998]

Yet Another Example MDP



MDPs

- **Episodes**

$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} s_3 \quad \dots \quad s_{h-1} \xrightarrow[r_{h-1}]{a_{h-1}} s_h$$

- **Expected total discounted reward (utility)**

$$E\left(r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \cdots + \gamma^h r_h\right)$$

- **Optimization goal**

$$\text{Maximize } E_{s\sim\mathcal{D};\, a_t\sim ?;\, s_t\sim\mathcal{P};\, r_t\sim\mathcal{R}}\left(\sum_{t=0}^{h} \gamma^t r_t \mid s_0 = s\right)$$

Policies

- **Deterministic policy**

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

- **Stochastic policy**

$$\pi : \mathcal{S} \mapsto \Omega(\mathcal{A})$$

- **Expected total discounted reward (utility)**

$$E_{s\sim\mathcal{D};\, a_t\sim\pi;\, s_t\sim\mathcal{P};\, r_t\sim\mathcal{R}}\left(\sum_{t=0}^{h}\gamma^t r_t \mid s_0 = s\right)$$

- **Optimal policy**

$$\pi^* = \arg\max_{\pi}\; E_{s\sim\mathcal{D};\, a_t\sim\pi;\, s_t\sim\mathcal{P};\, r_t\sim\mathcal{R}}\left(\sum_{t=0}^{h}\gamma^t r_t \mid s_0 = s\right)$$

Value Functions

- **State Value Function V**

$$V^{\pi}(s) = E_{a_t \sim \pi;\, s_t \sim \mathcal{P};\, r_t \sim \mathcal{R}} \left(\sum_{t=0}^{h} \gamma^t r_t \mid s_0 = s \right)$$

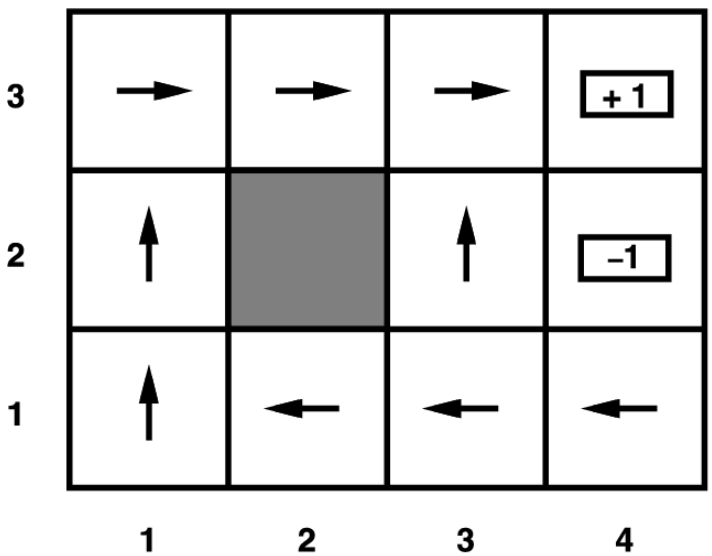
$$s \xrightarrow[r_0]{\pi(s)} s_1 \xrightarrow[r_1]{\pi(s_1)} s_2 \xrightarrow[r_2]{\pi(s_2)} s_3 \quad \dots \quad s_{h-1} \xrightarrow[r_{h-1}]{\pi(s_{h-1})} s_h$$

- **State-Action Value Function Q**

$$Q^{\pi}(s,a) = E_{a_t \sim \pi;\, s_t \sim \mathcal{P};\, r_t \sim \mathcal{R}} \left(\sum_{t=0}^{h} \gamma^t r_t \mid s_0 = s,\ a_0 = a \right)$$

$$s \xrightarrow[r_0]{a} s_1 \xrightarrow[r_1]{\pi(s_1)} s_2 \xrightarrow[r_2]{\pi(s_2)} s_3 \quad \dots \quad s_{h-1} \xrightarrow[r_{h-1}]{\pi(s_{h-1})} s_h$$

Example: Grid World



Policy π(s)

State Value Function $V^{\pi}(s)$

3	0.812	0.868	0.918	+1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388
	1	2	3	4

Bellman Equation for V

$$s \xrightarrow[r_0]{\pi(s)} s_1 \xrightarrow[r_1]{\pi(s_1)} s_2 \xrightarrow[r_2]{\pi(s_2)} s_3 \quad \dots \quad s_{h-1} \xrightarrow[r_{h-1}]{\pi(s_{h-1})} s_h$$

$$V^\pi(s) = \underbrace{\mathcal{R}(s, \pi(s))}_{\text{first step}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s, \pi(s), s')\, V^\pi(s')}_{\text{subsequent steps}}$$

$$V^\pi = \mathcal{R}^\pi + \gamma \mathbf{P}^\pi V^\pi$$

- a linear system of size ($|S| \times |S|$) with unknowns V^π
- can be solved directly or iteratively

Bellman Equation for Q

$$s \xrightarrow[\mathcal{R}(s,a)]{a} s' \xrightarrow[r_1]{\pi(s')} s_2 \xrightarrow[r_2]{\pi(s_2)} s_3 \quad \dots \quad s_{h-1} \xrightarrow[r_{h-1}]{\pi(s_{h-1})} s_h$$

$$Q^\pi(s,a) = \underbrace{\mathcal{R}(s,a)}_{\text{first step}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, Q^\pi(s', \pi(s'))}_{\text{subsequent steps}}$$

$$Q^\pi = \mathcal{R} + \gamma \mathbf{P} \mathbf{\Pi}_\pi Q^\pi$$

- a linear system of size (|S||A|×|S||A|) with unknowns Q^π
- can be solved directly or iteratively

Greedy Policy Improvement

- **Improved (greedy) policy over V**

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} \left\{ \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, V^{\pi}(s') \right\}$$

$$\forall s \in \mathcal{S}, \quad V^{\pi'}(s) \geq V^{\pi}(s)$$

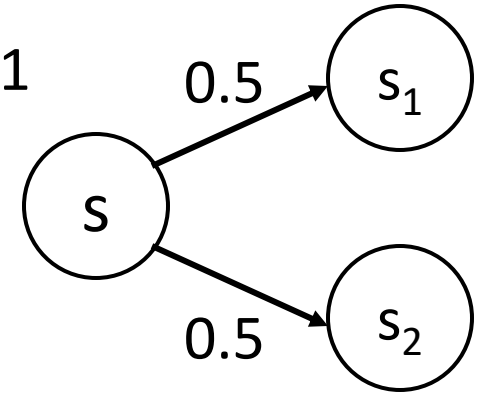
- **Improved (greedy) policy over Q**

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} Q^{\pi}(s,a)$$

$$\forall s \in \mathcal{S}, \quad Q^{\pi'}\big(s, \pi'(s)\big) \geq Q^{\pi}\big(s, \pi(s)\big)$$

Greedy Policy Improvement Example

$R(s, a_1)=1$

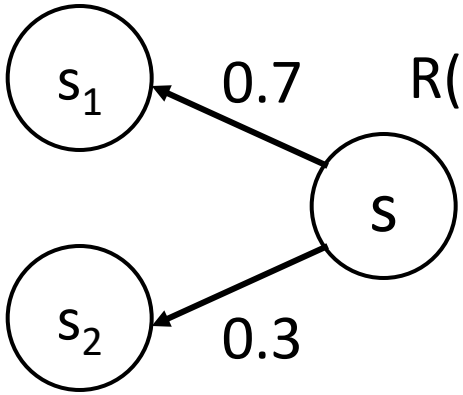


Action 1

$V^\pi(s_1) = 10$

$\gamma = 1$

$V^\pi(s_2) = 5$



$R(s, a_2)=0.5$

Action 2

$$\pi'(s) = \text{argmax}\{ 1 + 1 \times (0.5\times10+0.5\times5), 0.5 + 1 \times (0.7\times10+0.3\times5) \} = a_2$$

$Q^\pi(s, a_1) = 8.5$

$Q^\pi(s, a_2) = 9.0$

$$\pi'(s) = \text{argmax}\{ 8.5, 9.0 \} = a_2$$

Bellman Optimality Equation for V

$$s \xrightarrow[\mathcal{R}(s,a)]{a} s' \xrightarrow[r_1]{\pi^*(s')} s_2 \xrightarrow[r_2]{\pi^*(s_2)} s_3 \quad \dots \quad s_{h-1} \xrightarrow[r_{h-1}]{\pi^*(s_{h-1})} s_h$$

$$V^{\pi^*}(s) = \max_{a \in \mathcal{A}} \left\{ \underbrace{\mathcal{R}(s,a)}_{\text{first step}} + \gamma \underbrace{\sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s')\, V^{\pi^*}(s')}_{\text{subsequent steps}} \right\}$$

- a non-linear system of size ($|S| \times |S|$) with unknowns V^{π^*}
- can be solved iteratively

Bellman Optimality Equation for Q

$$s \xrightarrow[\mathcal{R}(s,a)]{a} s' \xrightarrow[r_1]{\pi^*(s')} s_2 \xrightarrow[r_2]{\pi^*(s_2)} s_3 \quad \dots \quad s_{h-1} \xrightarrow[r_{h-1}]{\pi^*(s_{h-1})} s_h$$

$$Q^{\pi^*}(s,a) = \underbrace{\mathcal{R}(s,a)}_{\text{first step}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \max_{a' \in \mathcal{A}} Q^{\pi^*}(s',a')}_{\text{subsequent steps}}$$

- a non-linear system of size (|S||A|×|S||A|) with unknowns Q^{π^*}
- can be solved iteratively

Variations of Bellman Equations

$$V^{\pi}(s) = \mathcal{R}(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,\pi(s))\, V^{\pi}(s')$$

$$\mathcal{R}(s)$$

$$V^{\pi^*}(s) = \mathcal{R}(s) + \gamma \max_{a \in \mathcal{A}} \left\{ \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, V^{\pi^*}(s') \right\}$$

$$Q^{\pi}(s,a) = \mathcal{R}(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, Q^{\pi}(s',\pi(s'))$$

$$Q^{\pi^*}(s,a) = \mathcal{R}(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \max_{a' \in \mathcal{A}} Q^{\pi^*}(s',a')$$

$$\mathcal{R}(s,a,s')$$

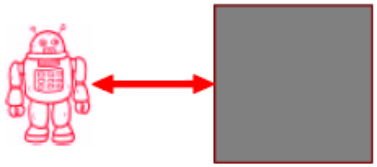
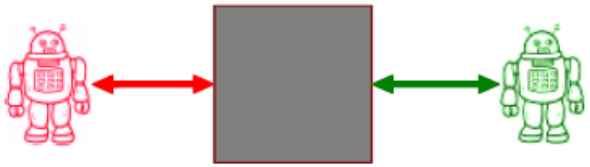
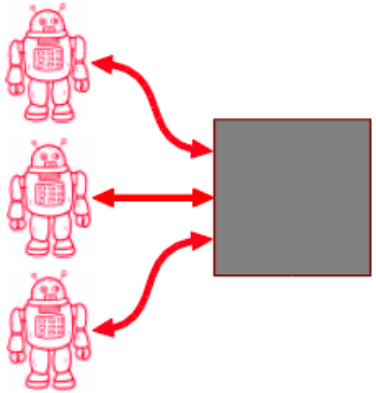
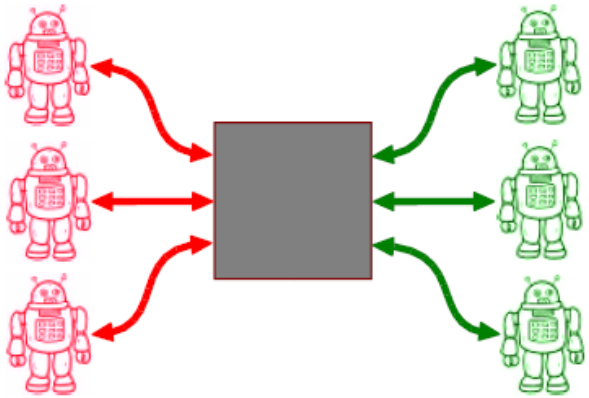
$$V^{\pi}(s) = \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,\pi(s)) \left(\mathcal{R}(s,a,s') + \gamma V^{\pi}(s')\right)$$

$$V^{\pi^*}(s) = \max_{a \in \mathcal{A}} \left\{ \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \left(\mathcal{R}(s,a,s') + \gamma V^{\pi^*}(s')\right) \right\}$$

$$Q^{\pi}(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \left(\mathcal{R}(s,a,s') + \gamma Q^{\pi}(s',\pi(s'))\right)$$

$$Q^{\pi^*}(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \left(\mathcal{R}(s,a,s') + \gamma \max_{a' \in \mathcal{A}} Q^{\pi^*}(s',a')\right)$$

Spectrum of Sequential Decision Making

	Collaboration	Competition
Single agent	 Markov Decision Process (MDP)	 Zero-Sum Markov Game
Multi agent	 Team MDP	 Team Zero-Sum Markov Game

Applications of MDPs

- **Economics/Operations Research**
  - Fleet maintenance (Howard, Rust)
  - Road maintenance (Golabi et al.)
  - Packet retransmission (Feinberg et al.)
  - Nuclear plant (Rothwell & Rust)
- **EE/Control**
  - Missile defense (Bertsekas et al.)
  - Inventory management (Van Roy et al.)
  - Football play selection (Patek & Bertsekas)
- **Agriculture**
  - Herd management (Kristensen, Toft)
- **AI/Computer Science**
  - Robot control (Koenig & Simmons, Thrun et al., Kaelbling et al., …)
  - Air campaign planning (Meuleau et al.)
  - Elevator control (Barto & Crites)
  - Computation scheduling (Zilberstein et al.)
  - Control and automation (Moore et al.)
  - Spoken dialogue management (Singh et al.)
  - Algorithm selection (Lagoudakis et al.)
- **Telecommunications**
  - Cellular channel allocation (Singh & Bertsekas)
  - Network routing (Boyan & Littman)

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 21

Reinforcement Learning
Planning under Uncertainty

TUC ECE, Spring 2023

Today

- **Planning under Uncertainty**
  - solving MPDs
  - value iteration
  - policy iteration
  - linear programming

Planning under Uncertainty

Finding Optimal Policies by Solving MDPs

Value Iteration

- **Idea**
  - iterative solution of the Bellman optimality equations
  - extraction of optimal policy through greedy improvement
- **Value Iteration for V**

$$V^{\pi^*}(s) = \max_{a \in \mathcal{A}} \left\{ \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, V^{\pi^*}(s') \right\}$$

- **Value Iteration for Q**

$$Q^{\pi^*}(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \max_{a' \in \mathcal{A}} Q^{\pi^*}(s',a')$$

Value Iteration for V

Value Iteration for V $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, V_0, \epsilon)$

// $\mathcal{S}$: States
// $\mathcal{A}$: Actions
// $\mathcal{P}$: Transition model
// $\mathcal{R}$: Reward model
// γ : Discount factor
// V_0 : Initial value function
// ϵ : Stopping criterion

$V \leftarrow V_0$
repeat
$V' \leftarrow V$
$$\forall s \in \mathcal{S}, V(s) = \max_{a \in \mathcal{A}} \left\{ \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, V(s') \right\}$$
until $(||V - V'|| < \epsilon)$
$$\forall s \in \mathcal{S},\ \pi(s) \leftarrow \arg\max_{a \in \mathcal{A}} \left\{ \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, V(s') \right\}$$
return π

Value Iteration for Q

Value Iteration for Q $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, Q_0, \epsilon)$

// $\mathcal{S}$: States
// $\mathcal{A}$: Actions
// $\mathcal{P}$: Transition model
// $\mathcal{R}$: Reward model
// γ : Discount factor
// Q_0 : Initial value function
// ϵ : Stopping criterion

$Q \leftarrow Q_0$
repeat
$Q' \leftarrow Q$
$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, Q(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \max_{a' \in \mathcal{A}} Q(s',a')$
until $(||Q - Q'|| < \epsilon)$
$\forall s \in \mathcal{S},\ \pi(s) \leftarrow \arg\max_{a \in \mathcal{A}} Q(s,a)$
return π

Grid World: Value Iteration

- γ=1 and $R(s)$=-0.04 for non-terminal states

- **Computation of V(s) for *s*=(1,1)**
  - $V(1, 1) = \max\{ -0.04 + \gamma[0.8V(1, 2) + 0.1V(2, 1) + 0.1V(1, 1)],$
    $-0.04 + \gamma[0.9V(1, 1) + 0.1V(1, 2)],$
    $-0.04 + \gamma[0.9V(1, 1) + 0.1V(2, 1)],$
    $-0.04 + \gamma[0.8V(2, 1) + 0.1V(1, 2) + 0.1V(1, 1)] \}$ ↑ ← ↓ →

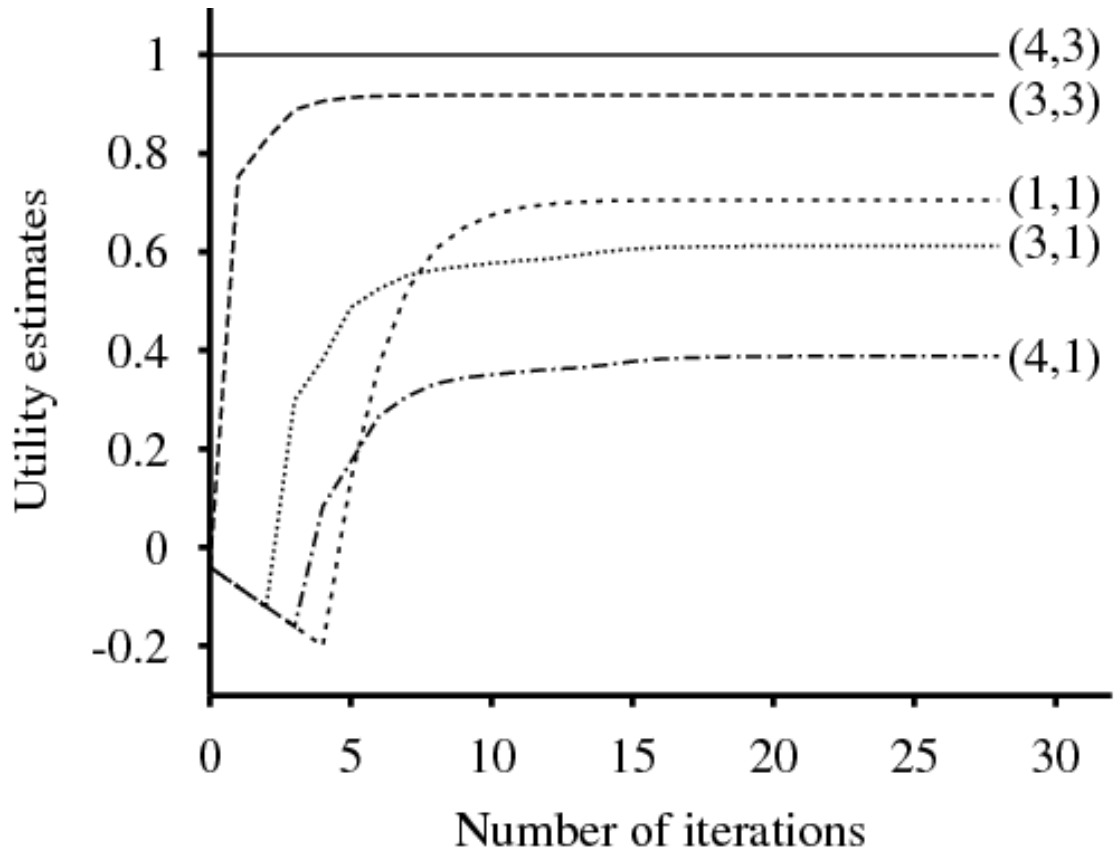
$$V^{\pi^*}(s) = \max_{a \in \mathcal{A}} \left\{ \underbrace{\mathcal{R}(s,a)}_{\text{first step}} + \gamma \underbrace{\sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s')\, V^{\pi^*}(s')}_{\text{subsequent steps}} \right\}$$

Best choice in state (1,1) is action UP

3	0.812	0.868	0.918	+1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388
	1	2	3	4

Grid World: Value Iteration Convergence

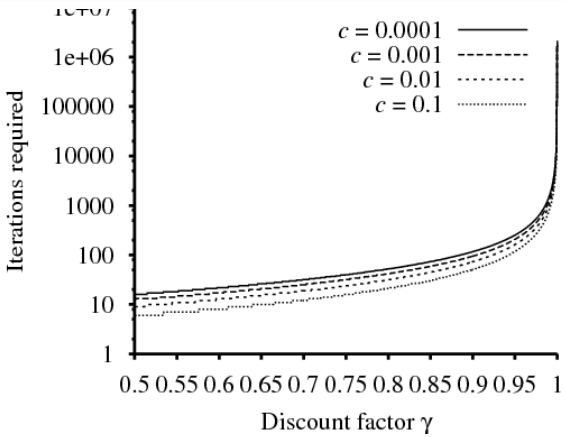
– Grid World with γ=1, $R(s)$=-0.04 for non-terminal states



Convergence of Value Iteration

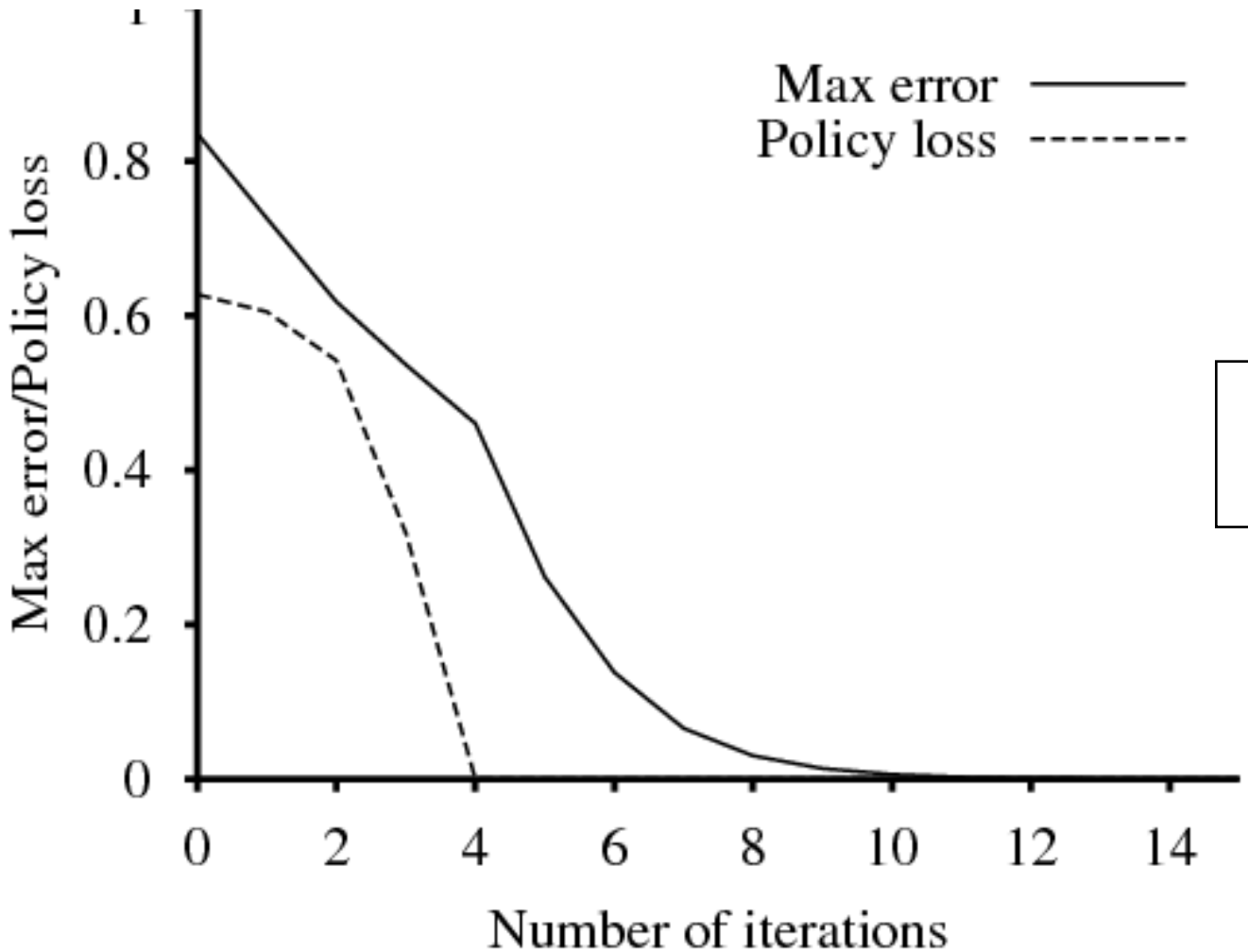
- **Contraction f**
  - the images of f are "closer" to each other than its arguments
  - contraction f: $\| f(x)-f(y) \| \leq \beta \| x-y \|$, $\beta<1$
- **Contraction properties**
  - every contraction has a unique fixed point
  - the image is closer to the fixed point than the argument
- **Value iteration**
  - Bellman*: right-hand side of the Bellman optimality equation
  - the Bellman* optimality operator is a contraction under $\| . \|_\infty$
  - $\| \text{Bellman*}(V)-\text{Bellman*}(V') \|_\infty \leq \gamma \| V-V' \|_\infty$

Errors in Value Iteration



- **Value error**
  - bounded reward function: $|R(s)| \leq R_{max}$
  - bounded value function: $|V(s)| \leq R_{max}/(1-\gamma)$
  - maximum initial error: $\|V-V^*\|_\infty \leq 2R_{max}/(1-\gamma)$
  - num of iterations N for error ε: $\lceil \log(2R_{max}/\varepsilon(1-\gamma))/\log(1/\gamma) \rceil$
  - N increases exponentially as γ goes to 1
  - termination condition: $\|V_{i+1}-V_i\|_\infty < \varepsilon(1-\gamma)/\gamma \Rightarrow \| V_{i+1} - V^*\|_\infty < \varepsilon$
- **Policy loss**
  - $\|V^{\pi_i}-V^*\|$ the largest loss if policy π_i is executed instead of π^*
  - $\|V-V^*\|_\infty < \varepsilon \Rightarrow \|V^{\pi_i}-V_i\|_\infty < 2\varepsilon\gamma/(1-\gamma)$
  - an optimal policy may be obtained before convergence

Grid World: Error in Value Iteration



Grid World
with γ=0,9

Policy Iteration

- **Idea: iterate**
  - policy evaluation: solution of the Bellman equations
  - policy improvement: greedy action selection
- **Policy Iteration for V**

$$V^\pi(s) = \mathcal{R}(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, \pi(s))\, V^\pi(s')$$

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} \left\{ \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, a)\, V^\pi(s') \right\}$$

- **Policy Iteration for Q**

$$Q^\pi(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, a)\, Q^\pi(s', \pi(s'))$$

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} Q^\pi(s, a)$$

Policy Iteration for V

Policy Iteration for V $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \pi_0)$

```
//   S   : States
//   A   : Actions
//   P   : Transition model
//   R   : Reward model
//   γ   : Discount factor
//   π0  : Initial policy
```

$\pi' \leftarrow \pi_0$

repeat

$\pi \leftarrow \pi'$

$V^\pi \leftarrow (\mathbf{I} - \gamma \mathcal{P} \mathbf{\Pi}_\pi)^{-1} \mathcal{R}$

$$\forall s \in \mathcal{S},\ \pi'(s) \leftarrow \arg\max_{a \in \mathcal{A}} \left\{ \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, V^\pi(s') \right\}$$

until $(\pi = \pi')$

return π

Policy Iteration for Q

Policy Iteration for Q $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \pi_0)$

```
//  S   : States
//  A   : Actions
//  P   : Transition model
//  R   : Reward model
//  γ   : Discount factor
//  π0  : Initial policy
```

$\pi' \leftarrow \pi_0$

repeat

$\pi \leftarrow \pi'$

$Q^\pi \leftarrow (\mathbf{I} - \gamma \mathcal{P} \mathbf{\Pi}_\pi)^{-1} \mathcal{R}$

$\forall s \in \mathcal{S},\ \pi'(s) \leftarrow \arg\max_{a \in \mathcal{A}} Q^\pi(s, a)$

until $(\pi = \pi')$

return π

Policy Iteration Extensions

- **Modified Policy Iteration**
  - policy evaluation (solution of linear system) is expensive, $O(|S|^3)$
  - idea: partial iterative computation of V^π or Q^π
  - small number k of Gauss-Seidel-type iterations, $kO(|S|^2)$

$$V^\pi(s) = \mathcal{R}(s,\pi(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,\pi(s),s')\, V^\pi(s')$$

$$Q^\pi(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s')\, Q^\pi(s',\pi(s'))$$

- **Asynchronous Policy Iteration**
  - idea: apply evaluation/improvement only to subsets of states
  - significant decrease in computational complexity
  - focus at points of interest

Linear Programming Approach

- **Idea**
  - optimize the values of V subject to some constraints
  - *constraint*: value V cannot be less than any Q value in each state
  - *objective*: minimize each V to match the best Q value
- **Linear program**
  - minimization of linear objective function (sum of $|S|$ variables)
  - subject to $|S||A|$ constraints (Bellman equation for V)
  - solution: optimal value function V^{π^*}
- **Complexity**
  - weakly polynomial (pseudo-polynomial) algorithm
  - polynomial in $|S|$, $|A|$, and the number of bits B for accuracy

Linear Programming MDP Algorithm

LP-MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ Computes the optimal policy

// $\mathcal{S}$: States
// $\mathcal{A}$: Actions
// $\mathcal{P}$: Transition model
// $\mathcal{R}$: Reward function
// γ : Discount factor

minimize

$$\sum_{s \in \mathcal{S}} V(s)$$

subject to

$$V(s) \geq \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') V(s'), \quad \forall s \in \mathcal{S}, \ \forall a \in \mathcal{A}$$

$$\forall s \in \mathcal{S}, \ \pi(s) \leftarrow \arg\max_{a \in \mathcal{A}} \left[\mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') V(s') \right]$$

return π

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 22

Reinforcement Learning
Fundamental Algorithms

TUC ECE, Spring 2023

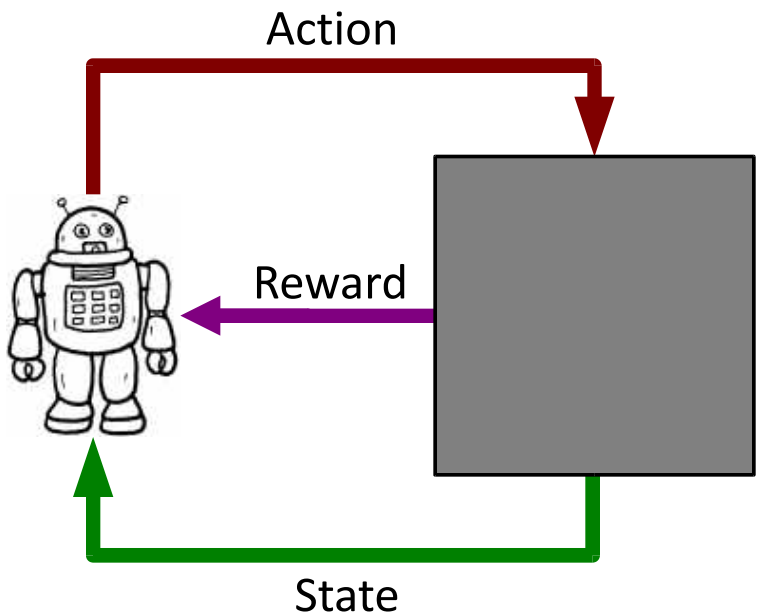
Today

- **Reinforcement Learning (RL)**
  - definition
  - process modeling
  - prediction and control
- **Prediction**
  - Adaptive Dynamic Programming (ADP)
  - Direct Utility Estimation (DUE)
  - Temporal Difference (TD) learning
- **Control**
  - SARSA
  - Q-learning

Reinforcement Learning

Learning from Mistakes!

Reinforcement Learning



Learn how to take actions in each state of the process so as to maximize in the long-term the cumulative reward!

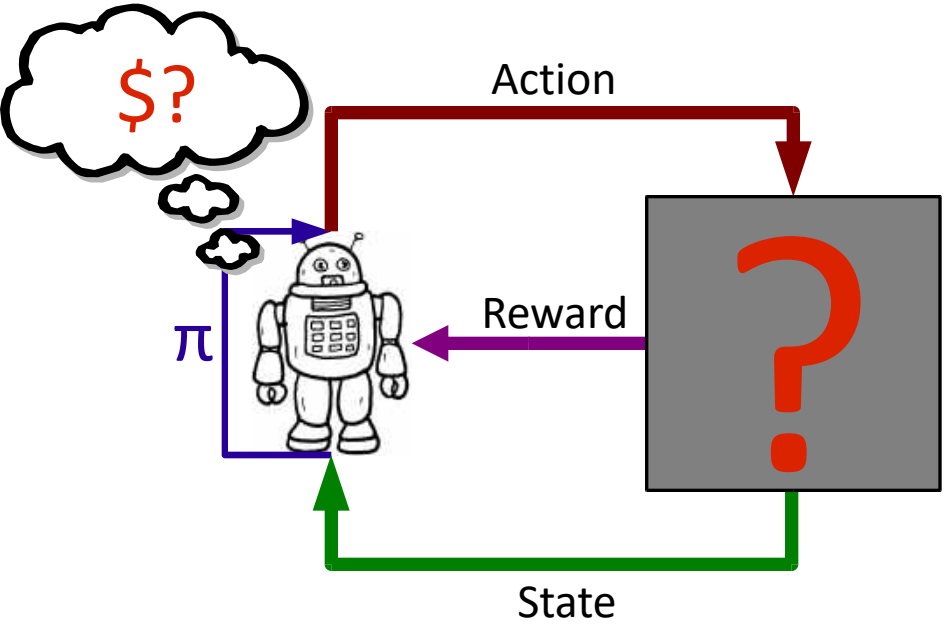
- reward reinforces good decisions (and penalizes the bad ones)
- learn from experience: (state, action, reward, next state)-samples
- samples taken from the process or from a generative model

Reinforcement Learning Setup

- **Known**
  - states, actions, rewards
- **Unknown**
  - transition model, reward model
- **Goal**
  - a good (or even optimal) policy
- **Significance**
  - learning without knowing what you are learning
  - generic approach for agent design
  - very hard problem

Reinforcement Learning Problems

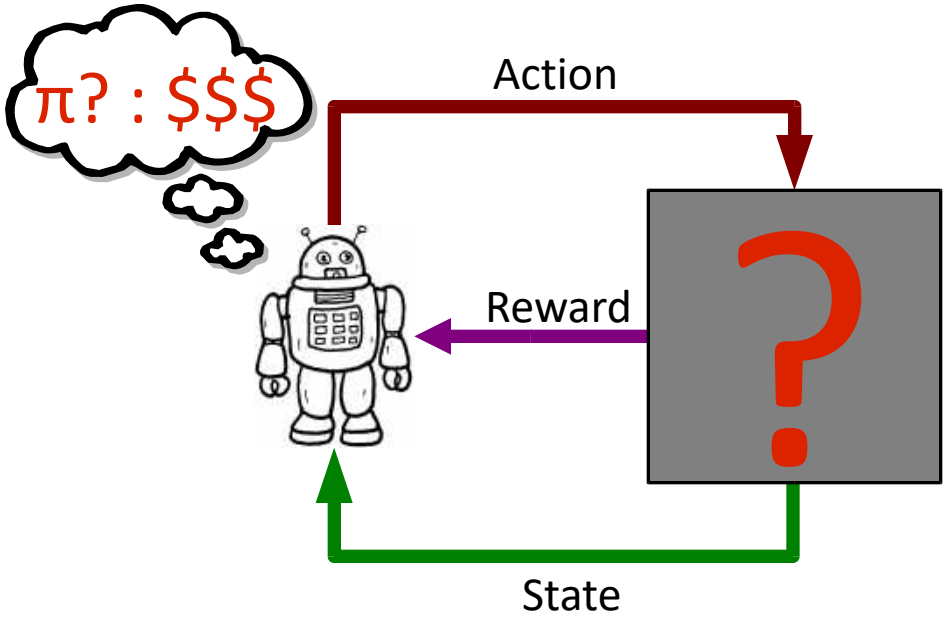
Prediction



Learn to predict the expected total reward for a fixed action policy

[Passive Reinforcement Learning]

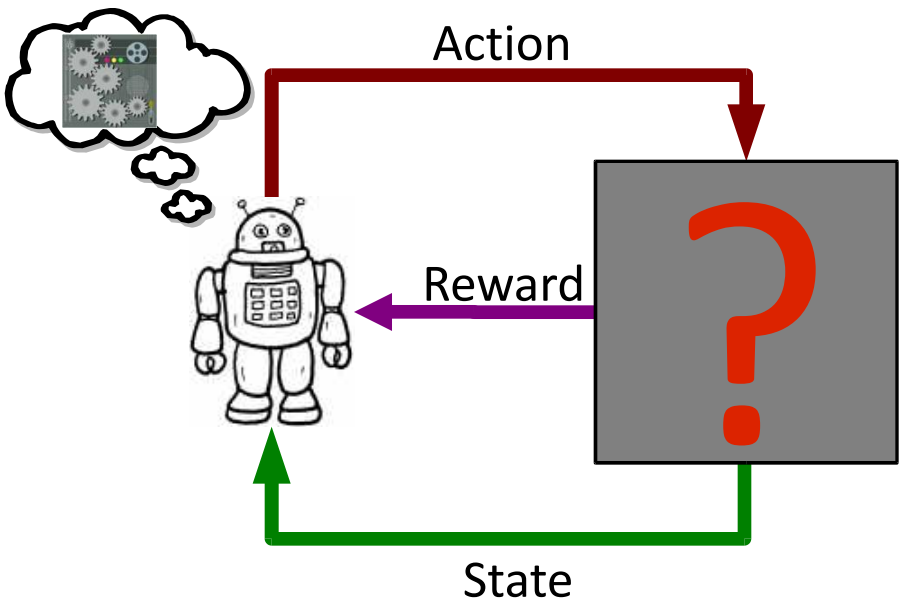
Control



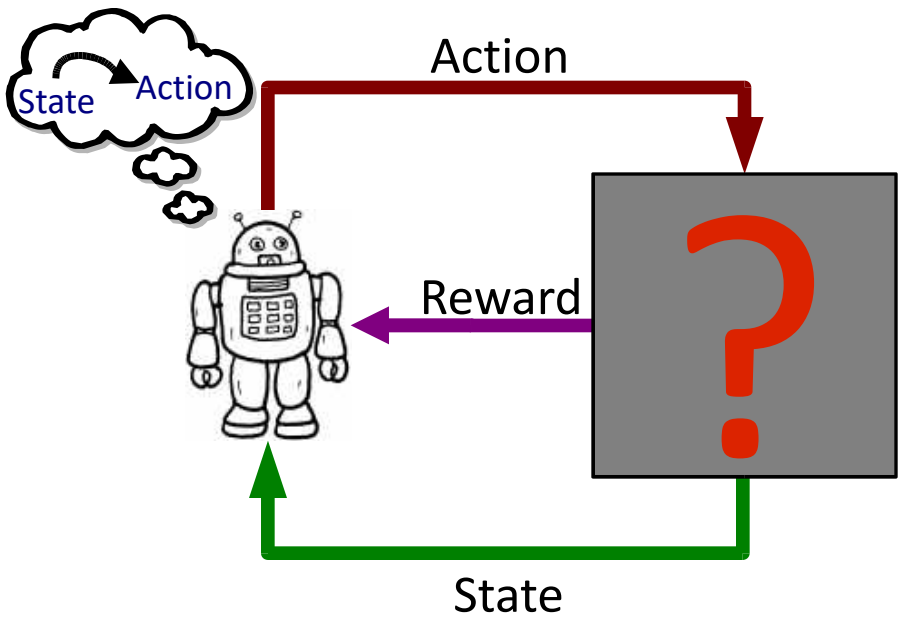
Learn to control the process to maximize the expected total reward

[Active Reinforcement Learning]

Reinforcement Learning Methodology

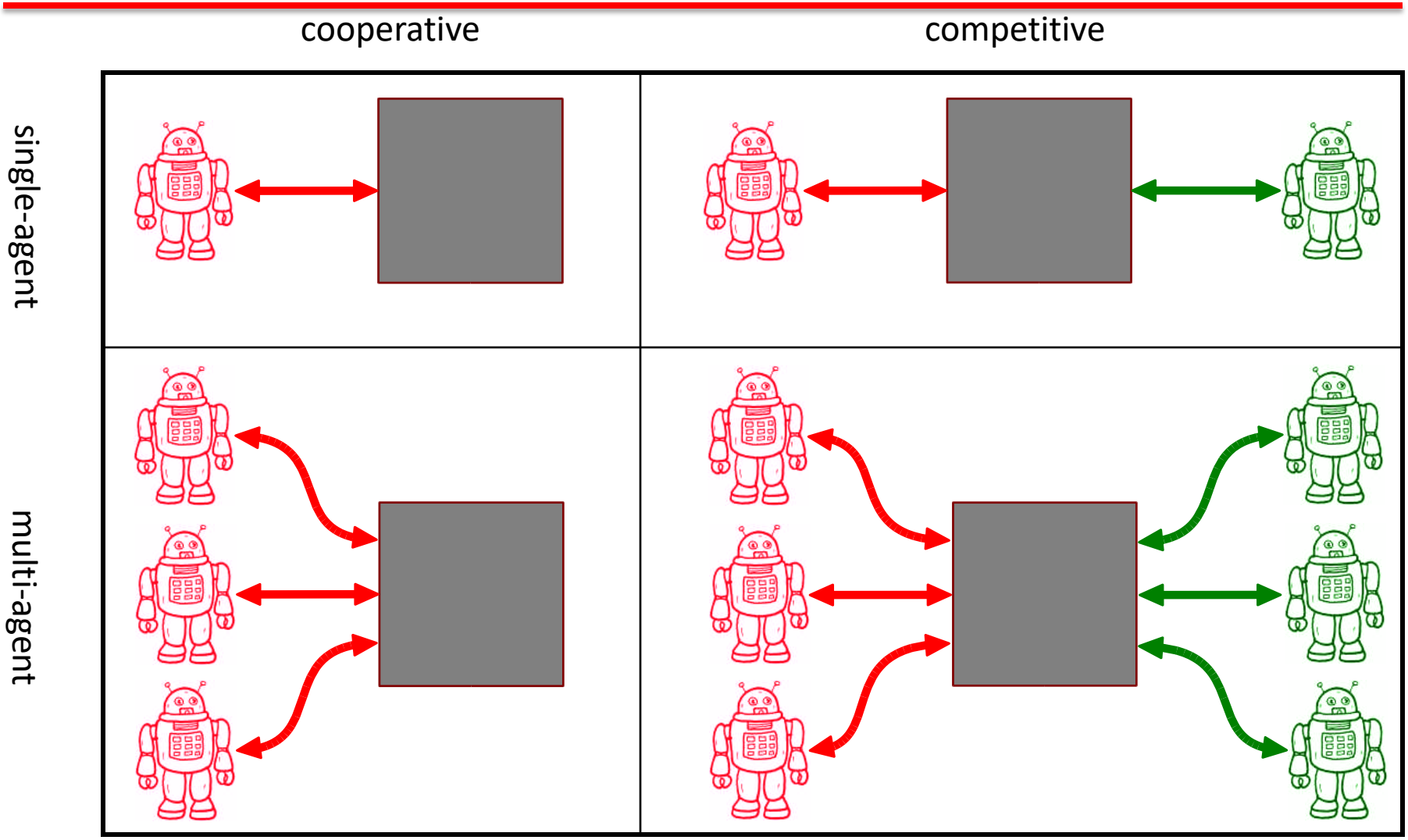


model-based learning



model-free learning

Reinforcement Learning Environment



Process Modeling

Markov Decision Processes

Markov Decision Process (MDP)

- **MDP (S, A, P, R, γ, D)**
  - S: state space of the process
  - A: action space of the process
  - P: transition model, $P(s' \mid s, a)$
  - R: reward model, $R(s, a)$
  - γ: discount factor, $0 < \gamma \leq 1$
  - D: initial state distribution
- **Markov property**
  - next state and reward are **independent** of history

Value Functions

- **State Value Function V**

$$V^{\pi}(s) = E_{a_t \sim \pi;\, s_t \sim \mathcal{P};\, r_t \sim \mathcal{R}} \left(\sum_{t=0}^{h} \gamma^t r_t \mid s_0 = s \right)$$

$$s \xrightarrow[r_0]{\pi(s)} s_1 \xrightarrow[r_1]{\pi(s_1)} s_2 \xrightarrow[r_2]{\pi(s_2)} s_3 \quad \dots \quad s_{h-1} \xrightarrow[r_{h-1}]{\pi(s_{h-1})} s_h$$

- **State-Action Value Function Q**

$$Q^{\pi}(s,a) = E_{a_t \sim \pi;\, s_t \sim \mathcal{P};\, r_t \sim \mathcal{R}} \left(\sum_{t=0}^{h} \gamma^t r_t \mid s_0 = s,\ a_0 = a \right)$$

$$s \xrightarrow[r_0]{a} s_1 \xrightarrow[r_1]{\pi(s_1)} s_2 \xrightarrow[r_2]{\pi(s_2)} s_3 \quad \dots \quad s_{h-1} \xrightarrow[r_{h-1}]{\pi(s_{h-1})} s_h$$

Value Function Representation

- **Exact**
  - a table with a distinct value/entry for each case
  - V : one entry for each s, $O(|S|)$ space
  - Q : one entry for each (s,a), $O(|S||A|)$ space
  - infeasible for realistic problems
- **Approximate**
  - approximate the value function with a function approximator
  - e.g. neural networks, polynomials, radial basis functions, ...
  - need only enough space to store the approximator parameters
  - equations and algorithms become harder to deal with
  - convergence properties are compromised

Linear Value Function Approximation

$$\widehat{V}^{\pi}(s) = \sum_{i=1}^{k} w_i^{\pi} \phi_i(s) = \phi(s)^{\top} w^{\pi}$$

$$\widehat{Q}^{\pi}(s,a) = \sum_{i=1}^{k} w_i^{\pi} \phi_i(s,a) = \phi(s,a)^{\top} w^{\pi}$$

- **Basis Functions (features) φ**
  - non-linear, in general
  - linearly independent
  - weights/parameters w^{π}
  - $k << |S|$ for V and $k << |S||A|$ for Q
  - properties: easy to design, engineer, interpret, modify, debug, …
  - examples: polynomials, radial basis functions, tile coding, …

Prediction

Passive Reinforcement Learning

The Prediction Problem

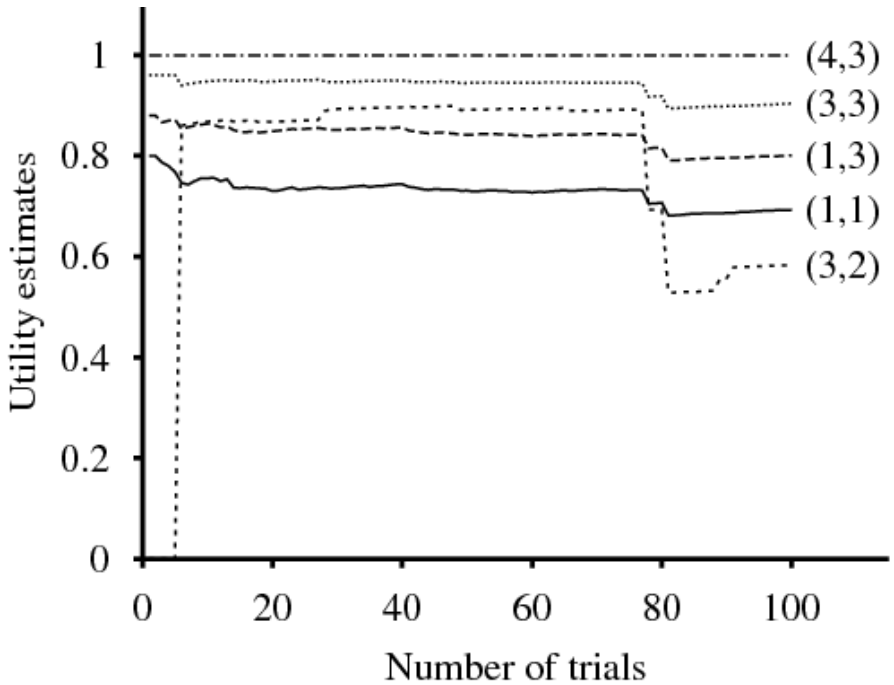
- **Given**
  - a fixed deterministic (or, stochastic) policy π
  - experience samples (s,a,r,s') [typically, a=π(s)]
- **Goal**
  - to predict the performance of policy π
  - to evaluate policy π
  - to learn the value function $V^{\pi}(s)$ of policy π
- **State value function**

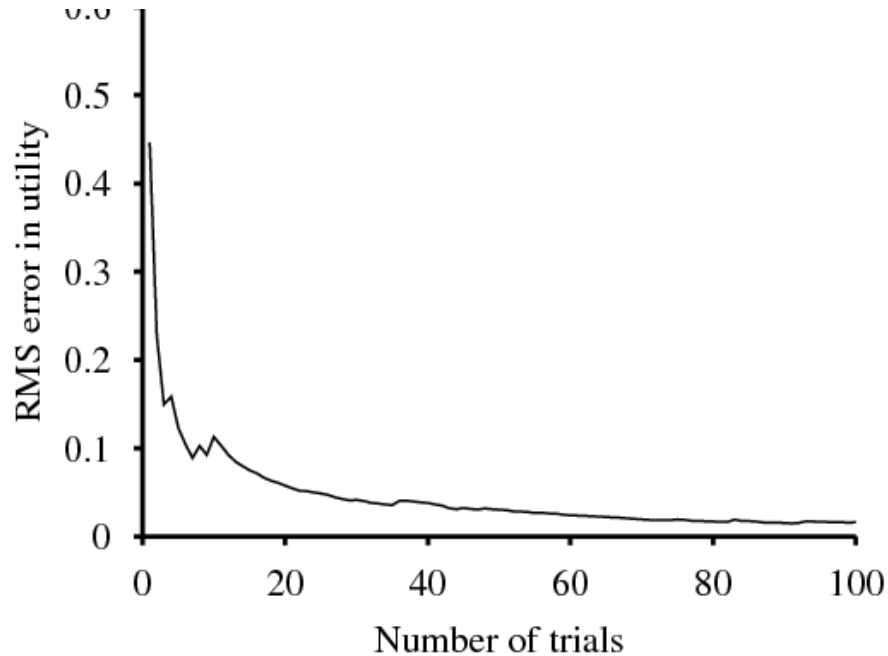
$$V^{\pi}(s) = E_{s_t \sim P;\ a_t \sim \pi;\ r_t \sim R}\left(\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right)$$

Adaptive Dynamic Programming

- **Adaptive Dynamic Programming (ADP)**
  - model-based learning
  - learns the transition model, $P(\, s' \mid s, \pi(s)\,)$
    - transition frequency counting
  - learns the reward model, $R(s)$
    - running average for each state
  - finds V^{π} through the Bellman equation and the full model
  - converges to the true model in the limit of infinite uniform samples
- **Properties**
  - huge space complexity
  - excellent use of samples

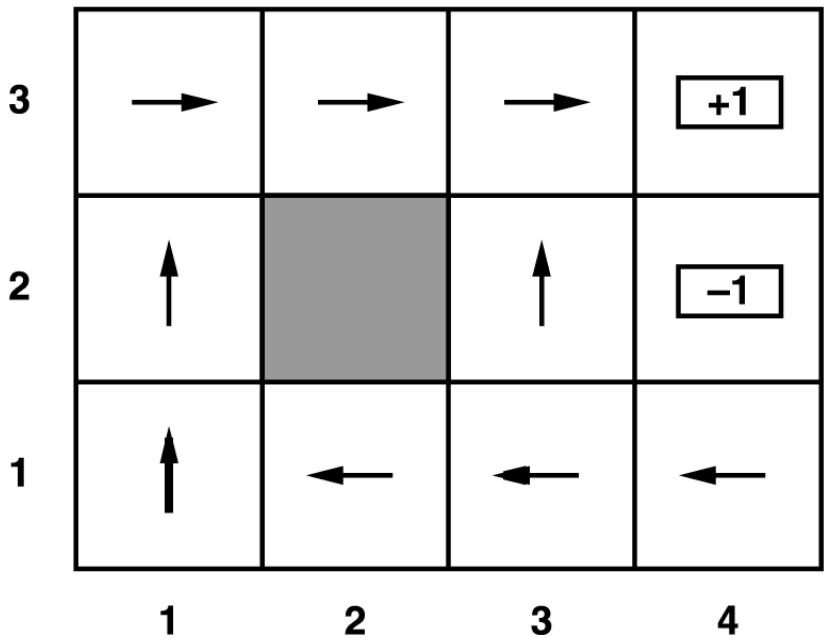
ADP Performance





- fast convergence
- each sample takes significant processing time

Trials in the Grid World



3	0.812	0.868	0.918	+ 1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388
	1	2	3	4

- **Sample trials**
  - $(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (4,3)_{+1}$
  - $(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (3,2)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (4,3)_{+1}$
  - $(1,1)_{-,04} \rightarrow (1,2)_{-,04} \rightarrow (1,3)_{-,04} \rightarrow (2,3)_{-,04} \rightarrow (3,3)_{-,04} \rightarrow (3,2)_{-,04} \rightarrow (4,2)_{-1}$

Monte-Carlo Learning

- **Direct Utility Estimation** [Widrow and Hoff, 1960]
  - utility = expected total (discounted) reward from state *s*
  - each episode (trial) gives one sample for each state visited
    - $(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (4,3)_{+1}$
    - ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
    - 0.72 0.76 0.80 0.84 0.88 0.92 0.96 1 (γ=1)
  - estimation: mean of all samples for each state
  - learning: maintaining a running mean for each state
  - convergence to the true values in the limit of infinitely-many trials
- **Properties**
  - ignores the dependencies between values (Bellman equation)
  - searches a larger space of functions and converges at a slow rate

Temporal Difference Learning

- **Basic idea**
  - local value update taking into account dependencies
  - (linear) Bellman equation $V^\pi(s) = \mathcal{R}(s,\pi(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,\pi(s),s')\, V^\pi(s')$
- **Temporal Difference (TD) Learning**
  - for each sample (s,a,r,s')

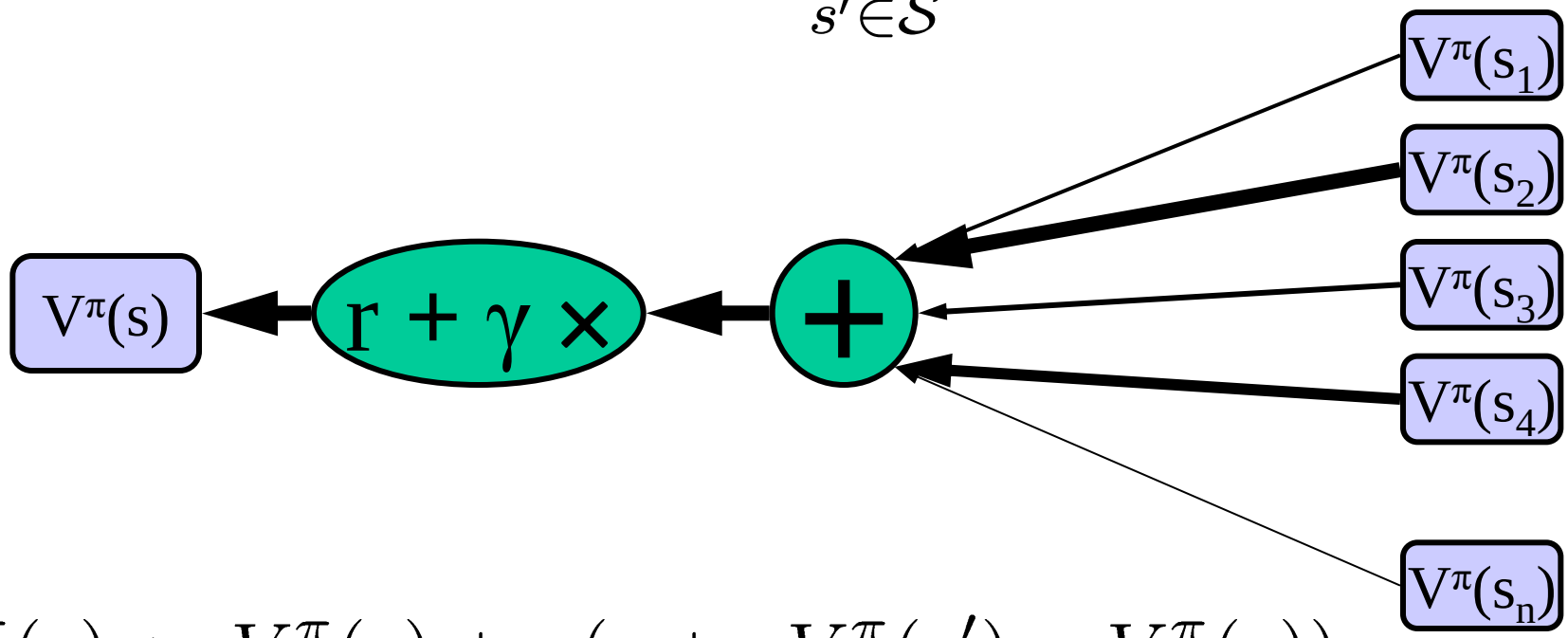
$$V^\pi(s) \leftarrow V^\pi(s) + \alpha(r + \gamma V^\pi(s') - V^\pi(s))$$

  - α = learning rate (decreased over time to avoid "oscillations")
  - if s' is a terminal state, we typically consider $V^\pi(s') = 0$

$$V^\pi(s) \leftarrow V^\pi(s) + \alpha\big(r - V^\pi(s)\big)$$

Temporal Difference Learning

$$V^\pi(s) = \mathcal{R}(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, \pi(s))\, V^\pi(s')$$



$$V^\pi(s) \leftarrow V^\pi(s) + \alpha(r + \gamma V^\pi(s') - V^\pi(s))$$

$$V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + \alpha(r + \gamma V^\pi(s'))$$

Temporal Difference Learning Algorithm

TD $(D,\ \pi,\ \gamma,\ V_0,\ \alpha_0,\ \sigma)$ // Learns V^π from samples

// D : Source of samples (s, a, r, s')
// π : Policy whose value function is sought
// γ : Discount factor
// V_0 : Initial value function
// α_0 : Initial learning rate
// σ : Learning rate schedule

$\widetilde{V} \leftarrow V_0;\ \alpha \leftarrow \alpha_0;\ t \leftarrow 0$

for each $(s, a, r, s') \in D(\pi)$

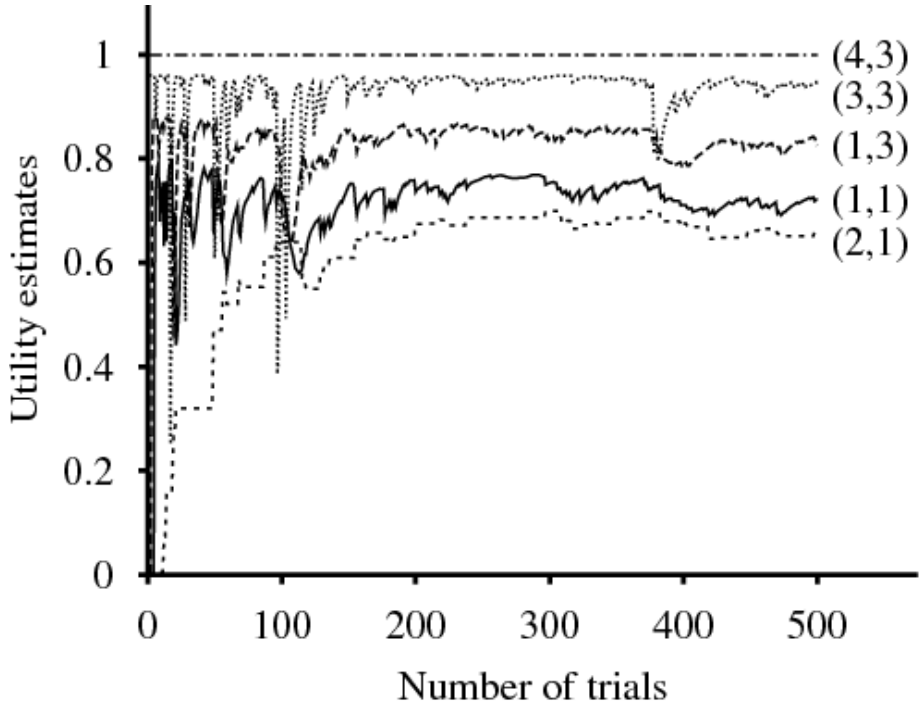
$\widetilde{V}(s) \leftarrow \widetilde{V}(s) + \alpha\Big(r + \gamma\widetilde{V}(s') - \widetilde{V}(s)\Big)$

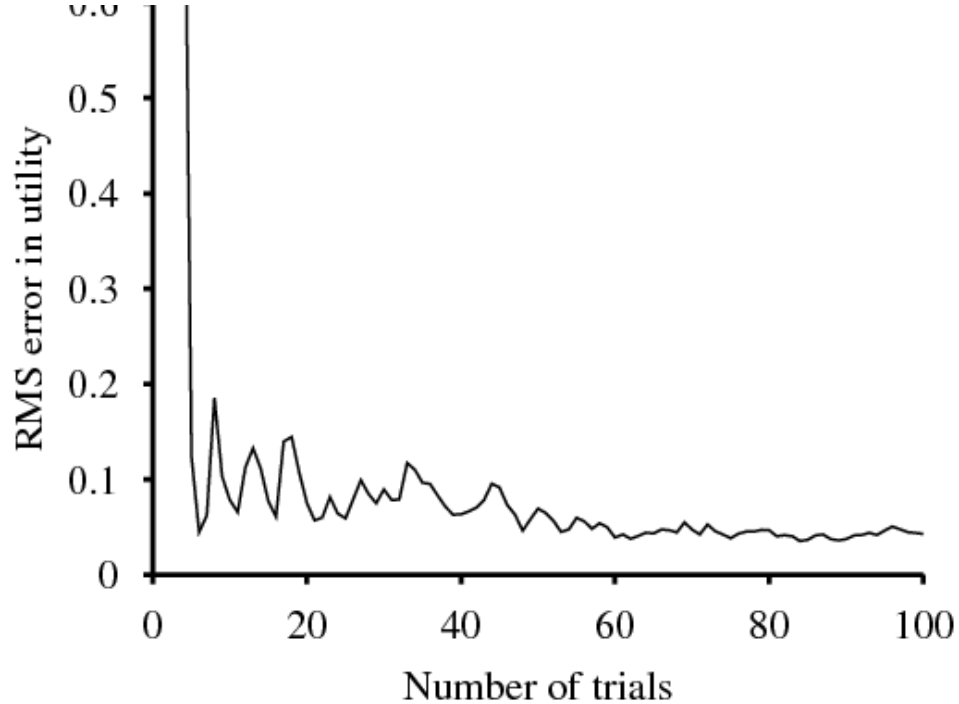
$\alpha \leftarrow \sigma(\alpha, \alpha_0, t)$

$t \leftarrow t + 1$

return $\widetilde{V}$

TD Performance





- faster than DUE, but more oscillatory convergence
- advantage: no need to wait until the end of the episode
- advantage: each sample is processed in little time

TD with Approximation

- **Generic approximation**
  - generic (non-linear) approximation, e.g. neural network
  - can only update the parameters of the approximator
  - update the parameters according to the temporal difference
  - use the gradient to determine the appropriate change

$$\widehat{V}(s;w^\pi) \qquad\qquad w^\pi = (w_1^\pi, w_2^\pi, \ldots, w_k^\pi)$$

$$update: w_i^\pi \leftarrow w_i^\pi + \alpha \frac{\partial \widehat{V}(s;w^\pi)}{\partial w_i^\pi}\Big(r + \gamma \widehat{V}(s';w^\pi) - \widehat{V}(s;w^\pi)\Big)$$

$$terminal: w_i^\pi \leftarrow w_i^\pi + \alpha \frac{\partial \widehat{V}(s;w^\pi)}{\partial w_i^\pi}\Big(r - \widehat{V}(s;w^\pi)\Big)$$

TD with Linear Approximation

- **Linear approximation**
  - linear combinations of (non-linear) basis functions
  - the gradient is easily computable

$$\widehat{V}(s; w^\pi) = \sum_{i=1}^{k} w_i^\pi \phi_i(s) = \phi(s)^\top w^\pi$$

$$update: w_i^\pi \leftarrow w_i^\pi + \alpha \phi_i(s) \Big(r + \gamma \phi(s')^\top w^\pi - \phi(s)^\top w^\pi \Big)$$

$$terminal: w_i^\pi \leftarrow w_i^\pi + \alpha \phi_i(s) \Big(r - \phi(s)^\top w^\pi \Big)$$

TD with Linear Approximation

TD-LA $(D,\ \pi,\ k,\ \phi,\ \gamma,\ w_0,\ \alpha_0,\ \sigma)$ // Learns $\widehat{V}^{\pi}$ from samples

// D : Source of samples (s, a, r, s')
// π : Policy whose value function is sought
// k : Number of basis functions
// ϕ : Basis functions
// γ : Discount factor
// w_0 : Initial parameters
// α_0 : Initial learning rate
// σ : Learning rate schedule

$\widetilde{w} \leftarrow w_0;\ \alpha \leftarrow \alpha_0;\ t \leftarrow 0$

for each $(s, a, r, s') \in D(\pi)$

$\widetilde{w} \leftarrow \widetilde{w} + \alpha\ \phi(s)\Big(r + \gamma\phi(s')^{\intercal}\widetilde{w} - \phi(s)^{\intercal}\widetilde{w}\Big)$

$\alpha \leftarrow \sigma(\alpha, \alpha_0, t)$

$t \leftarrow t + 1$

return $\widetilde{w}$

Control

Active Reinforcement Learning

The Control Problem

- **Given**
  - experience samples (*s*,*a*,*r*,*s*') from the unknown process
- **Goal**
  - to learn a good (optimal, if possible) policy π
- **Idea**
  - a better policy can be retrieved from a state-action value function
- **State-action value function**

$$Q^{\pi}(s,a) = E_{a_t \sim \pi;\ s_t \sim \mathcal{P};\ r_t \sim \mathcal{R}} \left(\sum_{t=0}^{h} \gamma^t r_t \mid s_0 = s,\ a_0 = a \right)$$

- **(Improved) policy**

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} Q^{\pi}(s,a)$$

Greedy Policy Improvement

- **Greedy (improved) policy over V**

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} \left\{ \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, V^{\pi}(s') \right\}$$

Inappropriate! Requires the model …

$$\forall s \in \mathcal{S}, \quad V^{\pi'}(s) \geq V^{\pi}(s)$$

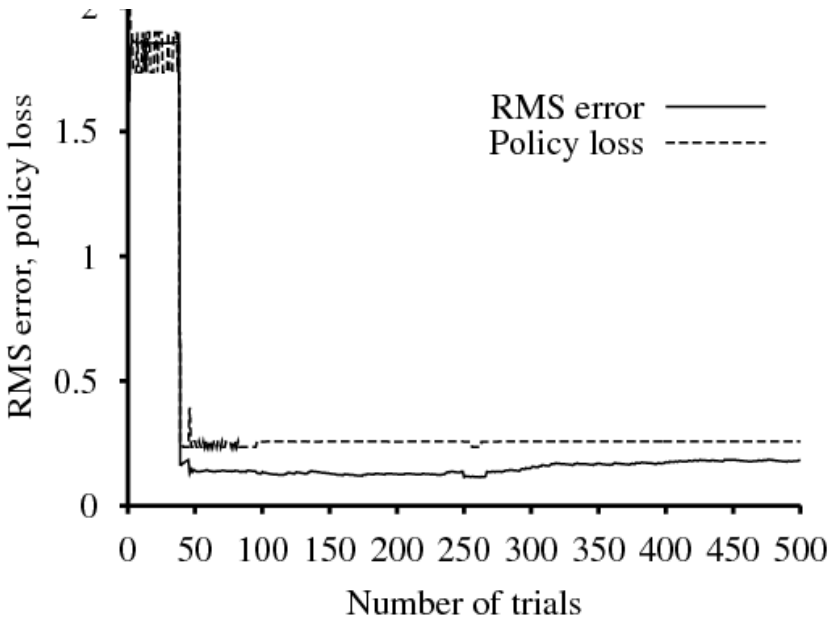
- **Greedy (improved) policy over Q**

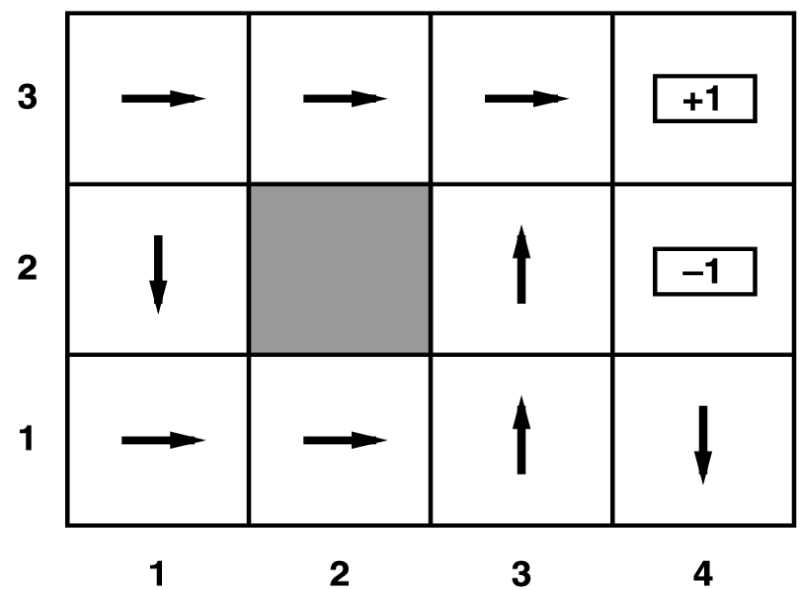
$$\pi'(s) = \arg\max_{a \in \mathcal{A}} Q^{\pi}(s,a)$$

$$\forall s \in \mathcal{S}, \quad Q^{\pi'}\big(s, \pi'(s)\big) \geq Q^{\pi}\big(s, \pi(s)\big)$$

Greedy Policy

- executes the best action according to the estimated value function
- non-optimal initial choices may disoriented exploration
- leaves areas of the state space unexplored!
- solution: optimistic initialization, exploration





Exploration vs. Exploitation

- **Exploitation**
  - use the greedy policy to maximize return in the short-term
- **Exploration**
  - choose random actions to discover things in the long-term
- **Exploration vs. Exploitation Dilemma**
  - exploration or exploitation?
- **Optimal balance**
  - Greedy in the Limit of Infinite Exploration (GLIE)

SARSA (s,a,r,s',a')

- **Main idea**
  - local update exploiting dependencies from the Bellman equation

$$Q^{\pi}(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, Q^{\pi}(s', \pi(s'))$$

- **SARSA** [Sutton, 1985]
  - for each sample (*s,a,r,s'*), where typically *α*=π(*s*):

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s', \pi(s')) - Q(s,a)\big)$$

  - policy π gradually becomes greedy (typically, 1-ε exploration)
  - if *s'* is terminal state, the update (typically) becomes:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r - Q(s,a)\big)$$

SARSA with Approximation

- **Generic approximation architecture**
  - generic (non-linear) architecture, e.g. neural network
  - only the parameters of the architecture can be updated
  - update the parameters based on the temporal difference
  - use the gradient to determine the appropriate change

$$\widehat{Q}(s,a;w) \qquad w = (w_1, w_2, \ldots, w_k)$$

$$update: w_i \leftarrow w_i + \alpha \frac{\partial \widehat{Q}(s,a;w)}{\partial w_i}\Big(r + \gamma \widehat{Q}(s', \pi(s'); w) - \widehat{Q}(s,a;w)\Big)$$

$$terminal: w_i \leftarrow w_i + \alpha \frac{\partial \widehat{Q}(s,a;w)}{\partial w_i}\Big(r - \widehat{Q}(s,a;w)\Big)$$

SARSA with Linear Approximation

- **Linear approximation architecture**
  - linear combination of basis functions
  - the gradient can be easily computed

$$\widehat{Q}(s,a;w) = \sum_{i=1}^{k} w_i \phi_i(s,a) = \phi(s,a)^\top w$$

$$update: w_i \leftarrow w_i + \alpha \phi_i(s,a)\Big(r + \gamma \phi(s', \pi(s'))^\top w - \phi(s,a)^\top w\Big)$$

$$terminal: w_i \leftarrow w_i + \alpha \phi_i(s,a)\Big(r - \phi(s,a)^\top w\Big)$$

SARSA Properties

- **Advantages**
  - processes each sample immediately
  - minimal update cost per sample
- **Disadvantages**
  - requires a huge number of samples
  - requires careful schedule for the learning rate
  - poses constraints on sample collection (on-policy)
  - requires careful handling on the policy greediness
  - makes minimal use of each sample
  - the ordering of samples influences the outcome
  - exhibits instabilities under approximate representations

Q-Learning

- **Main Idea**
  - local update exploiting dependencies from the optimality equation

$$Q^{\pi^*}(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \max_{a' \in \mathcal{A}} Q^{\pi^*}(s',a')$$

- **Q-Learning** [Watkins, 1989]
  - for each sample (*s,a,r,s'*):

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma \max_{a' \in \mathcal{A}} Q(s',a') - Q(s,a)\big)$$

  - if *s'* is a terminal state, the update (typically) becomes:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r - Q(s,a)\big)$$

Q-Learning with Approximation

- **Generic approximation architecture**
  - generic (non-linear) architecture, e.g. neural network
  - only the parameters of the architecture can be updated
  - update the parameters based on the temporal difference
  - use the gradient to determine the appropriate change

$$\widehat{Q}(s,a;w) \qquad w = (w_1, w_2, \ldots, w_k)$$

$$update: w_i \leftarrow w_i + \alpha \frac{\partial \widehat{Q}(s,a;w)}{\partial w_i}\left(r + \gamma \max_{a' \in \mathcal{A}} \widehat{Q}(s',a';w) - \widehat{Q}(s,a;w)\right)$$

$$terminal: w_i \leftarrow w_i + \alpha \frac{\partial \widehat{Q}(s,a;w)}{\partial w_i}\left(r - \widehat{Q}(s,a;w)\right)$$

Q-Learning with Linear Approximation

- **Linear approximation architecture**
  - linear combination of basis functions
  - the gradient can be easily computed

$$\widehat{Q}(s,a;w) = \sum_{i=1}^{k} w_i \phi_i(s,a) = \phi(s,a)^\top w$$

$$update: w_i \leftarrow w_i + \alpha \phi_i(s,a)\Big(r + \gamma \max_{a' \in \mathcal{A}} \phi(s',a')^\top w - \phi(s,a)^\top w\Big)$$

$$terminal: w_i \leftarrow w_i + \alpha \phi_i(s,a)\Big(r - \phi(s,a)^\top w\Big)$$

Q-Learning Properties

- **Advantages**
  - processes each sample immediately
  - minimal update cost per sample
  - poses no constraints on sample collection (off-policy)
- **Disadvantages**
  - requires a huge number of samples
  - requires careful schedule for the learning rate
  - makes minimal use of each sample
  - the ordering of samples influences the outcome
  - exhibits instabilities under approximate representations

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 23

Reinforcement Learning

Batch Algorithms, Experimentation

TUC ECE, Spring 2023

Today

- **Prediction**
  - least-squares TD (LSTD)
- **Control**
  - least-squares policy iteration (LSPI)
  - rollout classification policy iteration (RCPI)
  - extension to continuous action spaces
- **Experimentation**

Prediction

Passive Reinforcement Learning

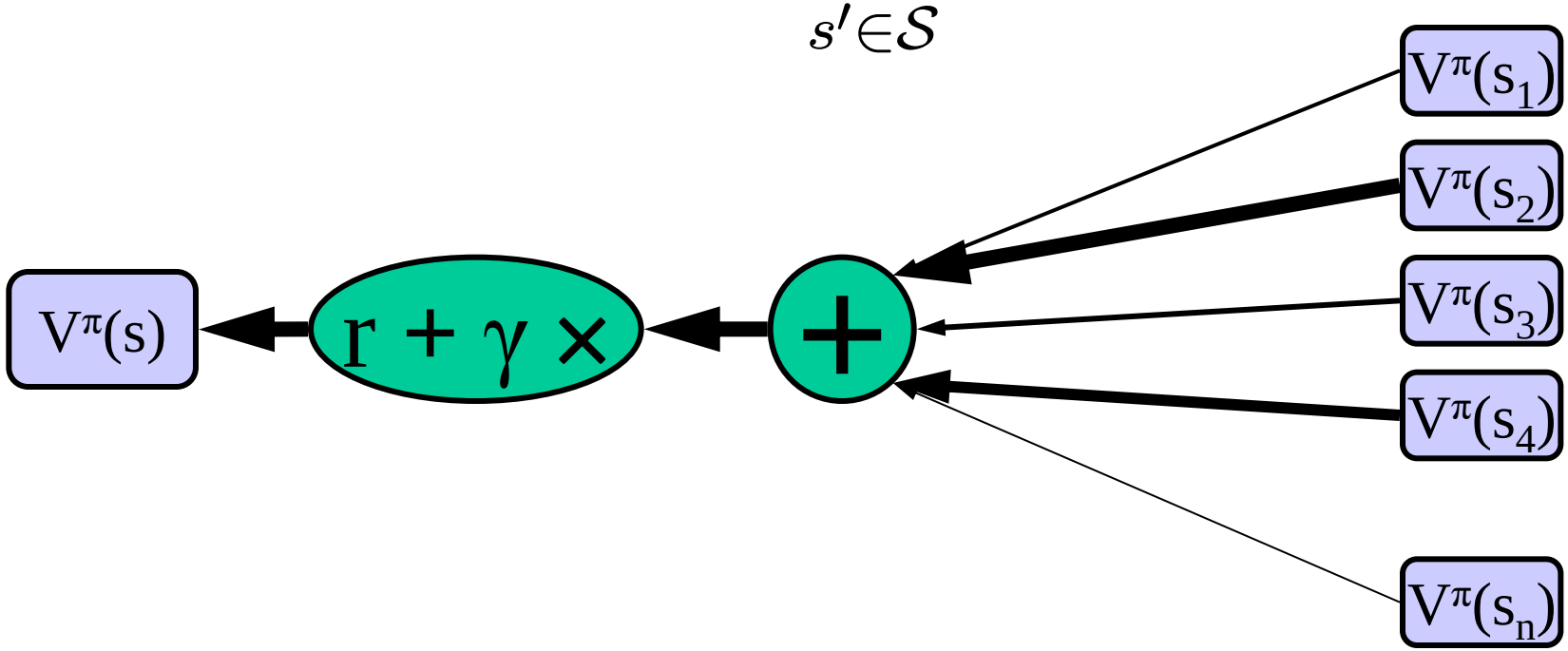
The Prediction Problem

- **Given**
  - a fixed deterministic (or, stochastic) policy π
  - experience samples (s,a,r,s') [typically, a=π(s)]
- **Goal**
  - to predict the performance of policy π
  - to evaluate policy π
  - to learn the value function $V^{\pi}(s)$ of policy π
- **State value function**

$$V^{\pi}(s) = E_{s_t \sim P;\ a_t \sim \pi;\ r_t \sim R}\left(\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right)$$

Temporal Difference Learning

$$V^\pi(s) = \mathcal{R}(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, \pi(s))\, V^\pi(s')$$



$$V^\pi(s) \leftarrow V^\pi(s) + \alpha(r + \gamma V^\pi(s') - V^\pi(s))$$

Least-Squares Temporal Difference

- TD is trying to solve a linear system (Bellman) incrementally

- **Idea**
  - collect all data and solve the (sampled) Bellman equation at once
  - the true value function satisfies the fixed point property
- **Linear architectures**
  - try to find the best point in the space of approximator parameters
  - enforce the fixed point property under orthogonal projection
  - solution is a fixed-point approximation to the true value function
- **Properties**
  - efficient use of all samples at once
  - elimination of learning rate, schedules, oscillations, ...

LSTD Algorithm

LSTD $(D,\ \pi,\ k,\ \phi,\ \gamma)$ // Learns $\widehat{V}^{\pi}$ from samples

// D : Source of samples (s, a, r, s')
// π : Policy whose value function is sought
// k : Number of basis functions
// ϕ : Basis functions
// γ : Discount factor

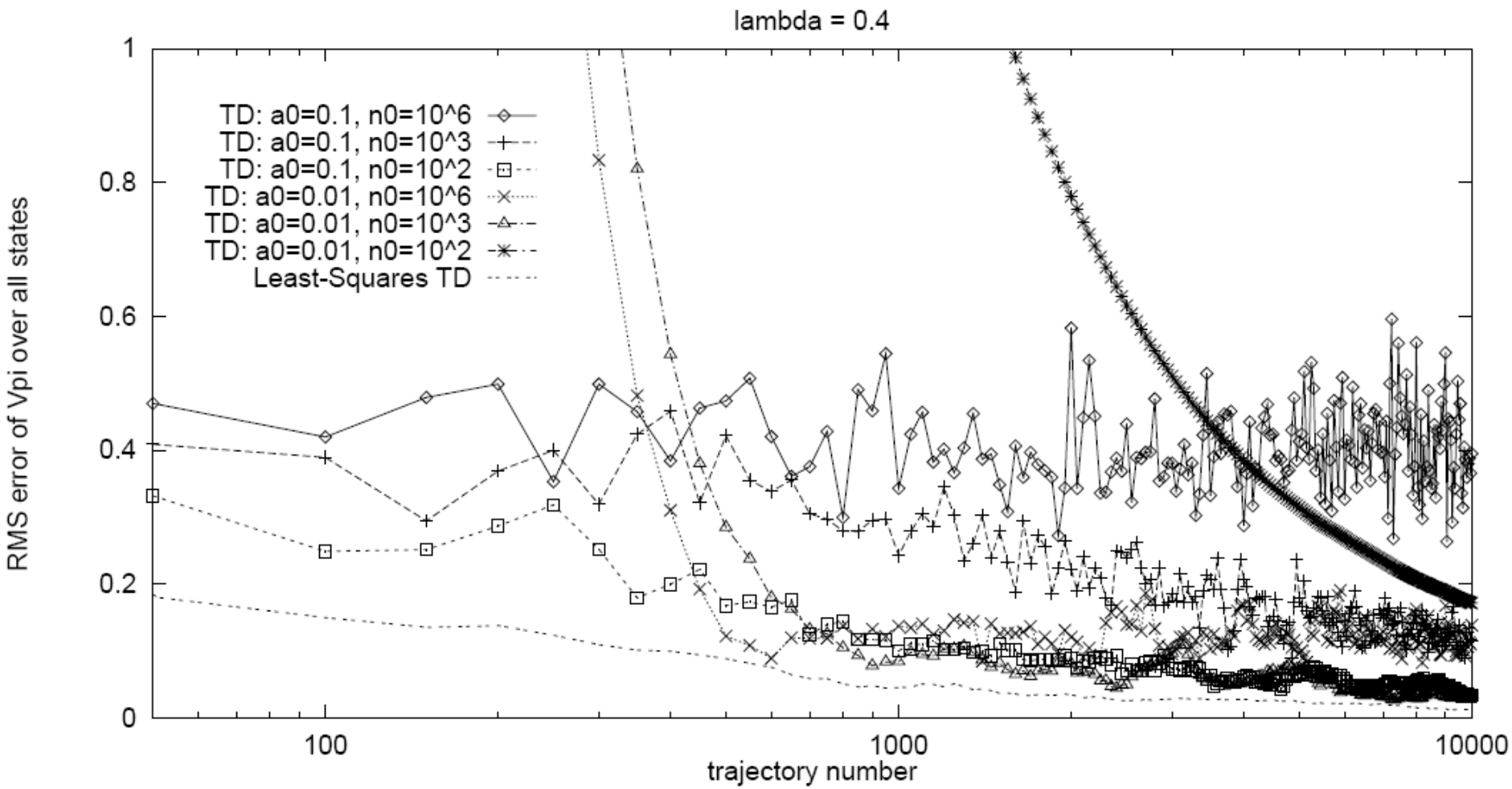
$\widetilde{\mathbf{A}} \leftarrow \mathbf{0}$ // $(k \times k)$ matrix
$\widetilde{b} \leftarrow \mathbf{0}$ // $(k \times 1)$ vector

for all $(s, a, r, s') \in D(\pi)$
$\widetilde{\mathbf{A}} \leftarrow \widetilde{\mathbf{A}} + \phi(s)\Big(\phi(s) - \gamma\phi(s')\Big)^{\mathsf{T}}$
$\widetilde{b} \leftarrow \widetilde{b} + \phi(s)r$

$\widetilde{w}^{\pi} \leftarrow \widetilde{\mathbf{A}}^{-1}\widetilde{b}$

return $\widetilde{w}^{\pi}$

LSTD Performance



Chain Walk from [Boyan, 2000]

Control

Active Reinforcement Learning

The Control Problem

- **Given**
  - experience samples (*s*,*a*,*r*,*s*') from the unknown process
- **Goal**
  - to learn a good (optimal, if possible) policy π
- **Idea**
  - a better policy can be retrieved from a state-action value function
- **State-action value function**

$$Q^{\pi}(s,a) = E_{a_t \sim \pi;\ s_t \sim \mathcal{P};\ r_t \sim \mathcal{R}} \left(\sum_{t=0}^{h} \gamma^t r_t \mid s_0 = s,\ a_0 = a \right)$$

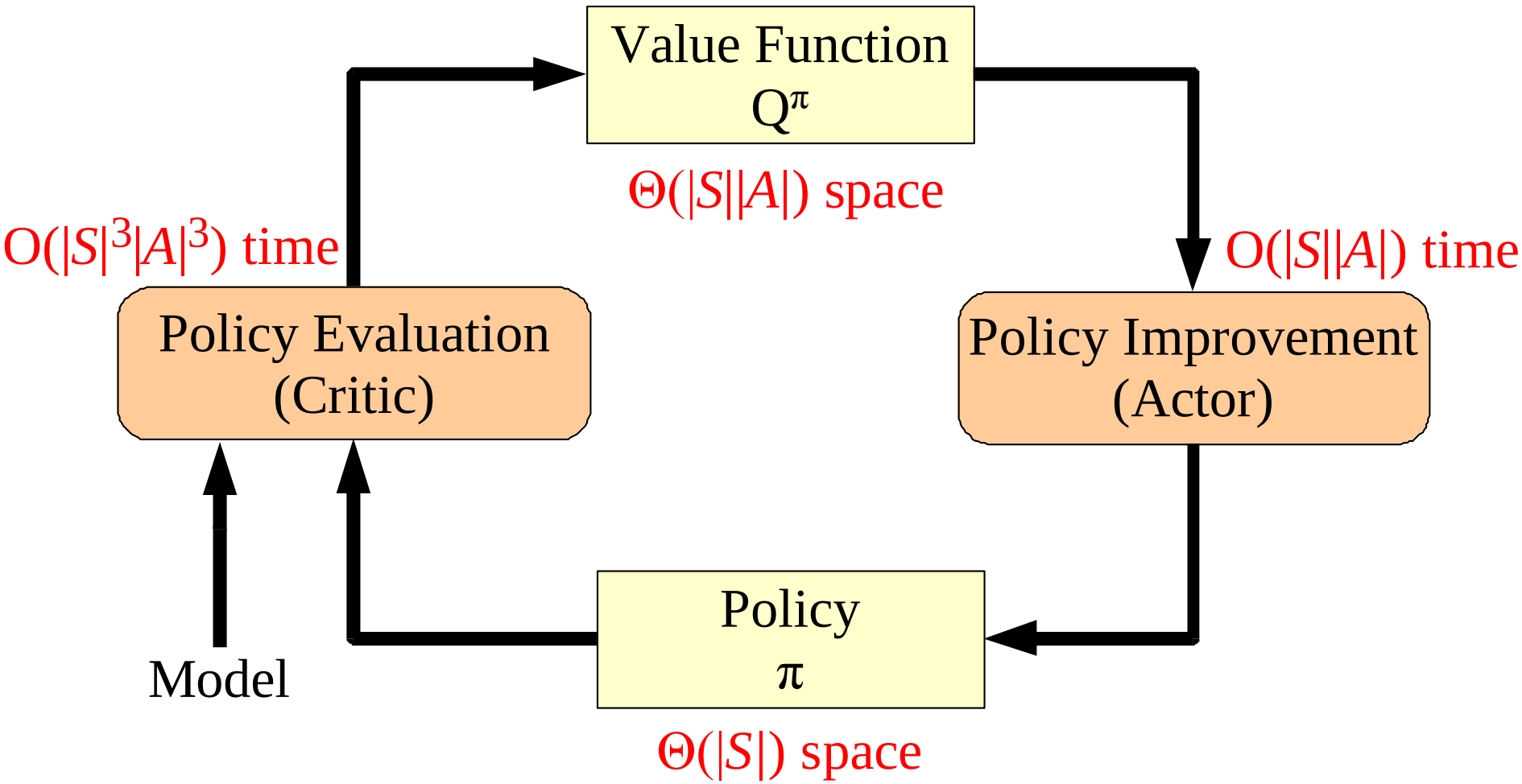
- **(Improved) policy**

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} Q^{\pi}(s,a)$$

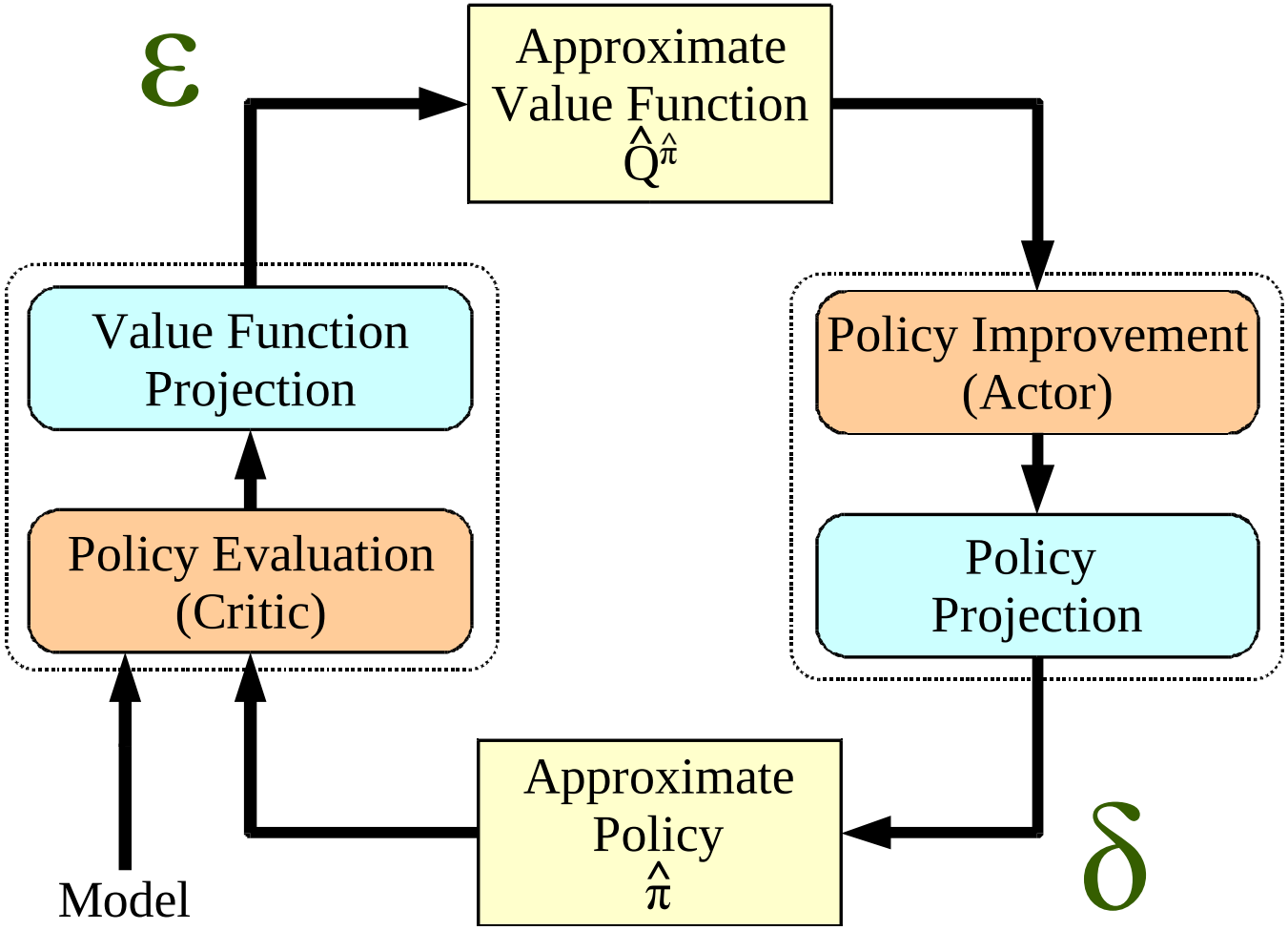
Least-Squares Policy Iteration

- **Problem**
  - SARSA attempts to solve a (changing) linear system incrementally
  - LSTD ideas cannot be applied because of changing policy
  - Q-Learning attempts to solve a non-linear system incrementally
  - LSTD ideas cannot be applied because of non-linearity
- **Idea**
  - exploit policy iteration (evaluation – improvement)
  - use a linear architecture for efficient representation
  - LSTD*Q*: variation of LSTD for efficient policy evaluation
  - implicit representation of improved policies (greedy improvement)
  - exploitation of the same sample set in all iterations

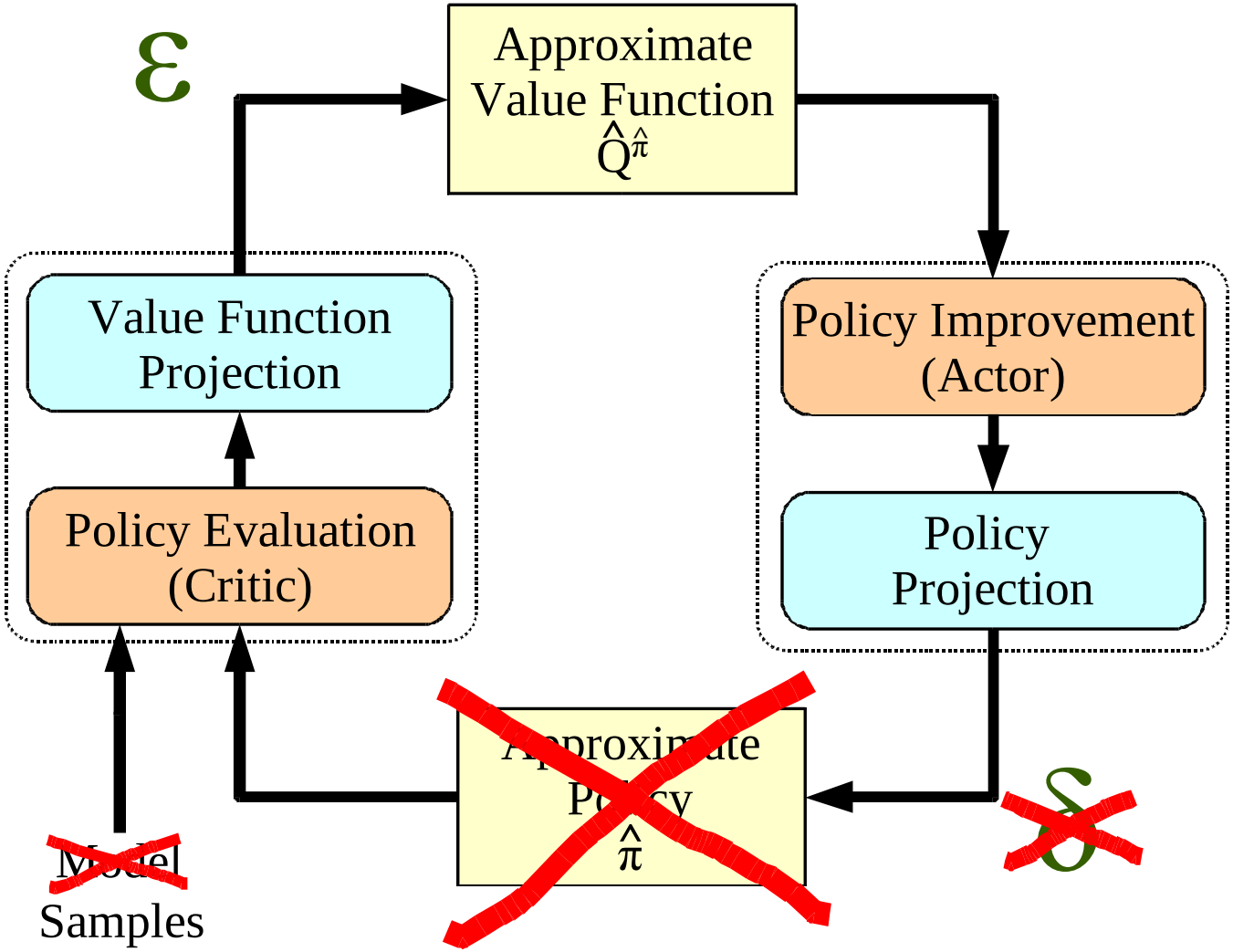
Policy Iteration



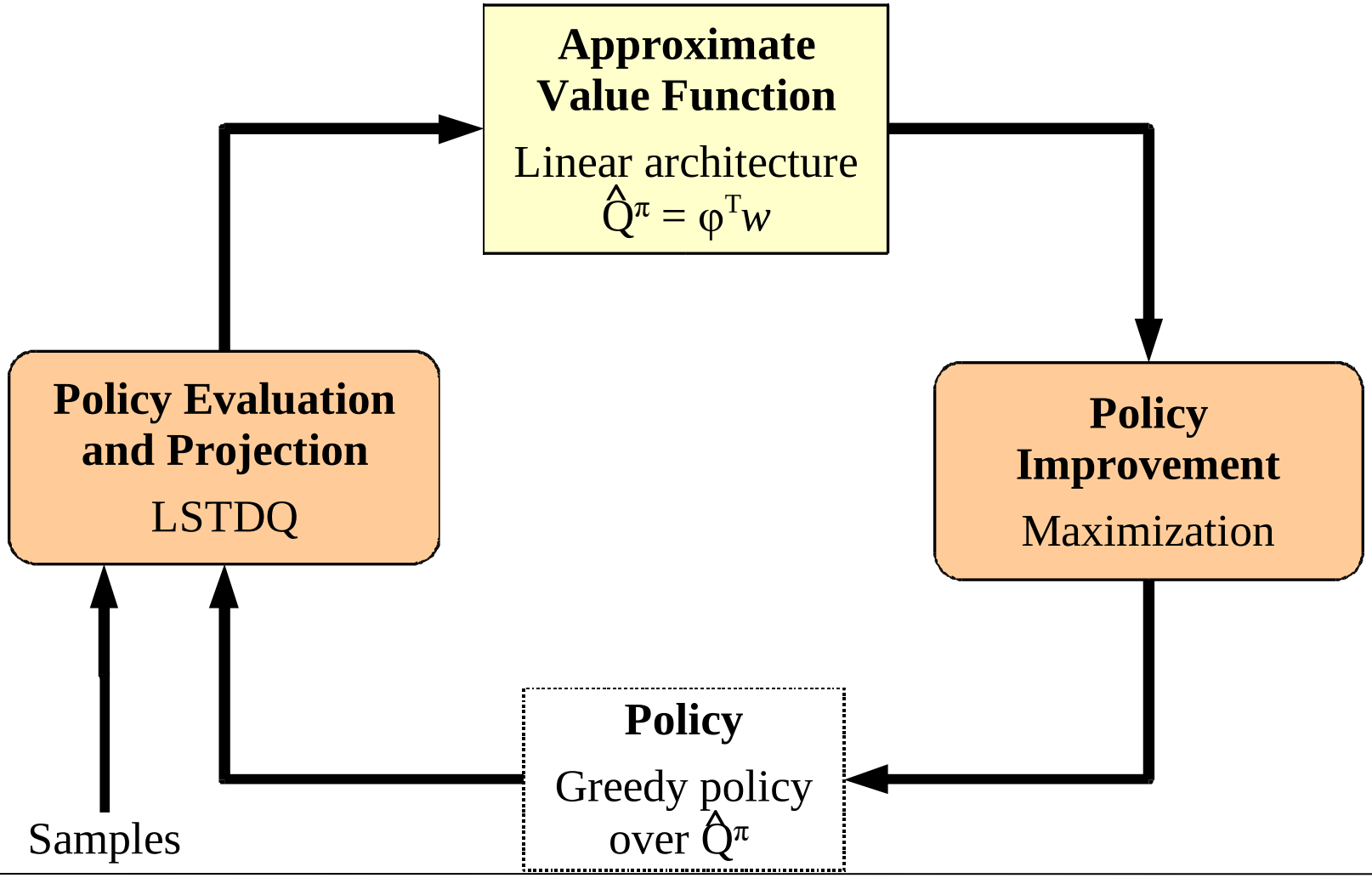
Approximate Policy Iteration



The Key Idea of LSPI



Least-Squares Policy Iteration



Fixed Point Approximation for Q

$$Q^\pi(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s')\, Q^\pi(s', \pi(s'))$$

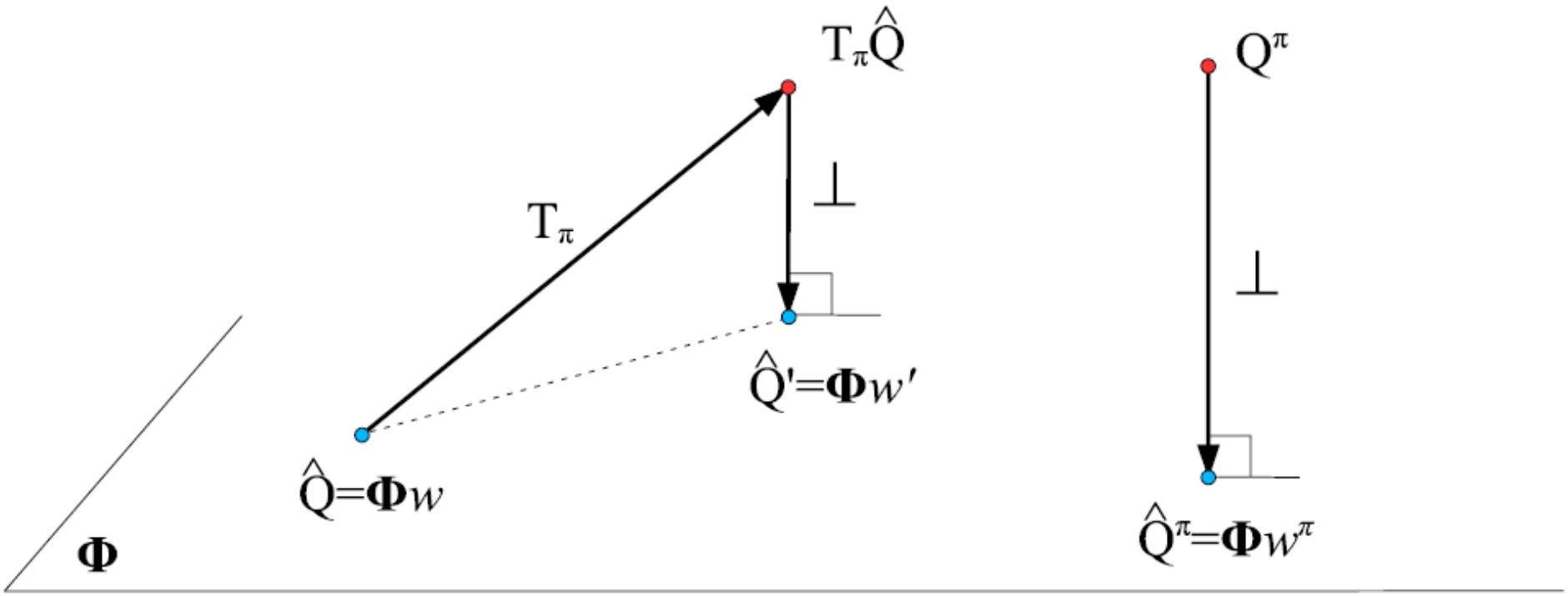
Fixed point property: $Q^\pi = T_\pi Q^\pi$

$$\widehat{Q}^\pi = \underbrace{\mathbf{\Phi}(\mathbf{\Phi}^\intercal \mathbf{\Phi})^{-1}\mathbf{\Phi}^\intercal}_{\text{Orthogonal Projection}} \underbrace{(T_\pi \widehat{Q}^\pi)}_{\text{Bellman Operator}}$$

$$\mathbf{\Phi} w^\pi = \mathbf{\Phi}(\mathbf{\Phi}^\intercal \mathbf{\Phi})^{-1}\mathbf{\Phi}^\intercal \quad (\mathcal{R} + \gamma \mathbf{P}\mathbf{\Pi}_\pi \mathbf{\Phi} w^\pi)$$

$$\underbrace{\mathbf{\Phi}^\intercal(\mathbf{\Phi} - \gamma \mathbf{P}\mathbf{\Pi}_\pi \mathbf{\Phi})}_{(k \times k)} w^\pi = \underbrace{\mathbf{\Phi}^\intercal \mathcal{R}}_{(k \times 1)}$$

Orthogonal Projection

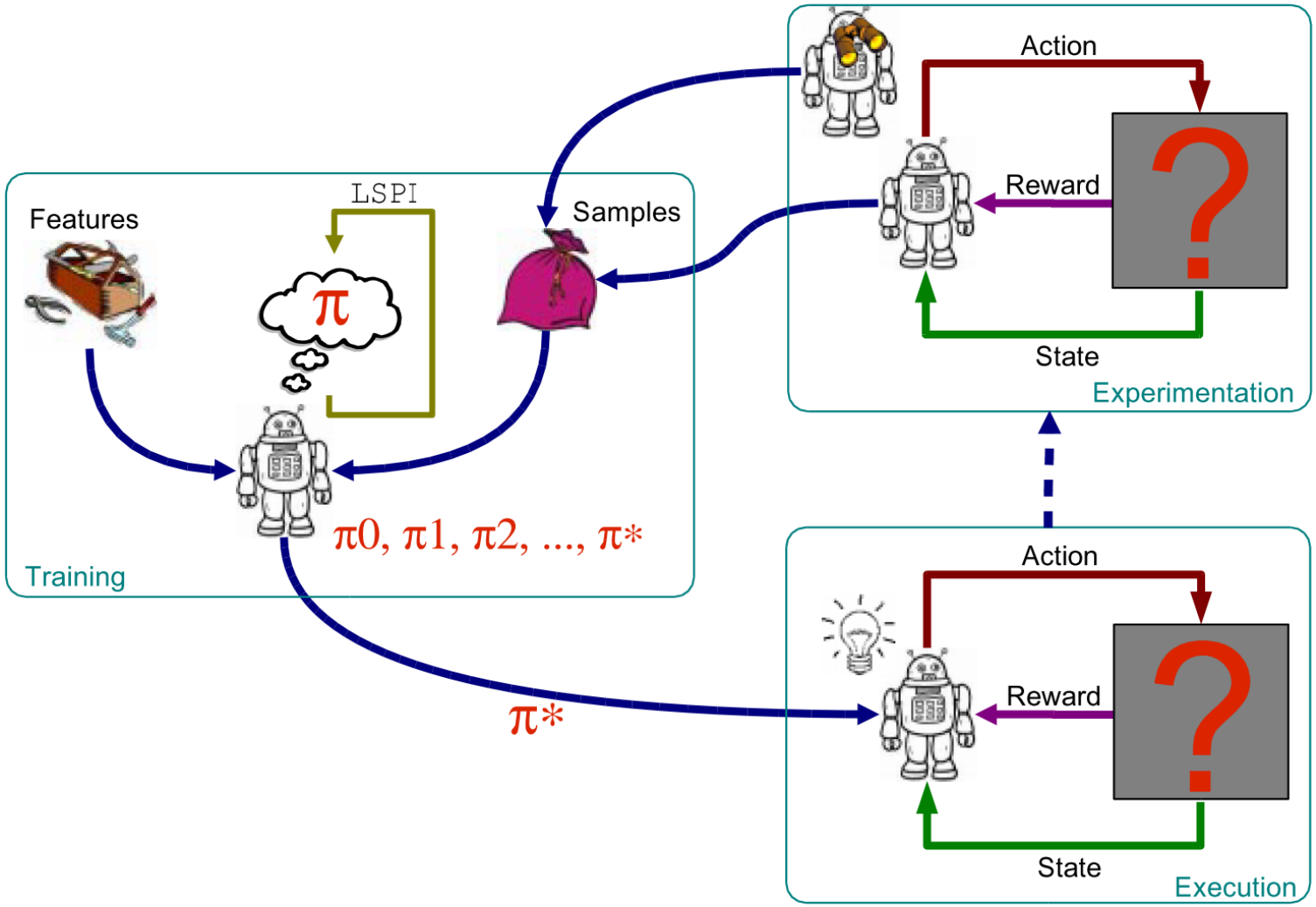


The LSPI Algorithm

LSPI(samples D, basis ϕ, discount γ, tol ϵ)

$w' \leftarrow \mathbf{0}$
repeat
  $w \leftarrow w'$, $\mathbf{A} \leftarrow \mathbf{0}$, $b \leftarrow \mathbf{0}$
  for each (s, a, r, s') in D **do**
    $a' = \arg\max_{a'' \in \mathcal{A}} \left\{ \phi(s', a'')^\top w \right\}$
    $\mathbf{A} \leftarrow \mathbf{A} + \phi(s, a)\Big(\phi(s, a) - \gamma\phi(s', a')\Big)^\top$
    $b \leftarrow b + \phi(s, a) r$
  end for
  $w' \leftarrow \mathbf{A}^{-1} b$
until $\big(\|w - w'\| < \epsilon\big)$
return w

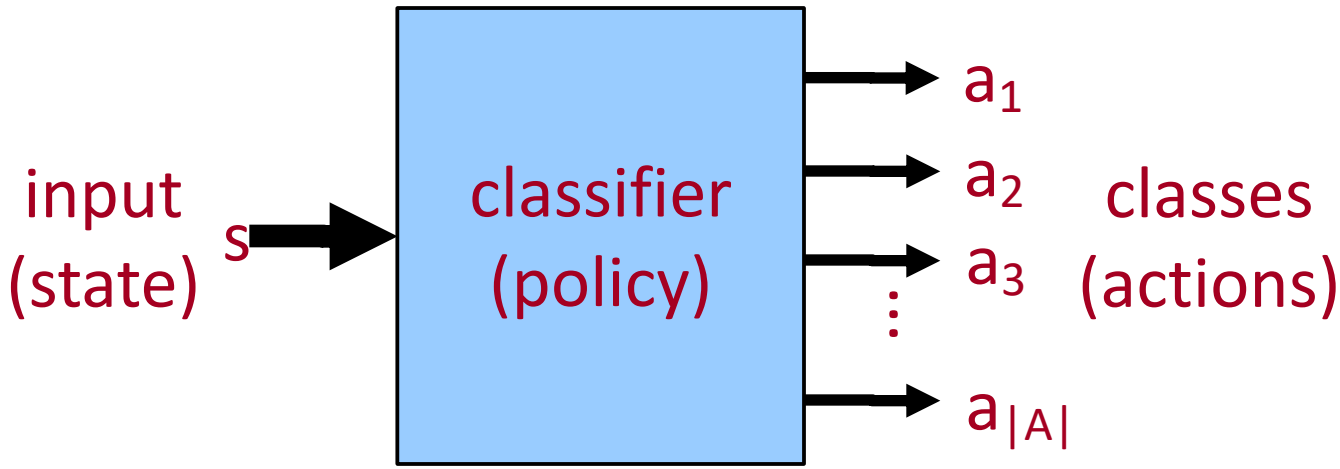
LSPI in a Nutshell



LSPI Properties

- **Properties**
  - learns policies of bounded quality
  - is stable; does not diverge
  - makes efficient use and reuse of training samples
  - handles successfully large scale problems
  - allows great flexibility in choosing/using basis functions
  - poses no restrictions on sample collection
  - it is simple and easy to implement
- **Limitations**
  - cannot guarantee convergence to the optimal solution
  - with badly distributed samples, the iteration may oscillate
  - with insufficient basis functions, it may converge to a poor policy

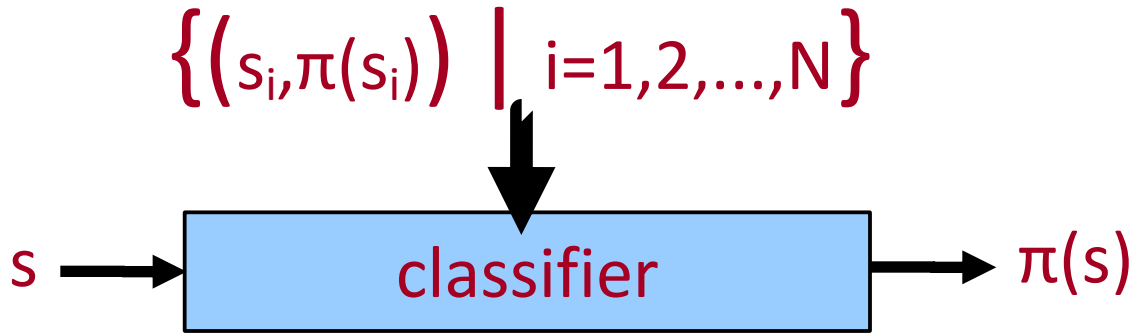
Policy as Classifier



- **Deterministic policy : maps states to actions**
- **Multi-class classifier : maps inputs to classes**

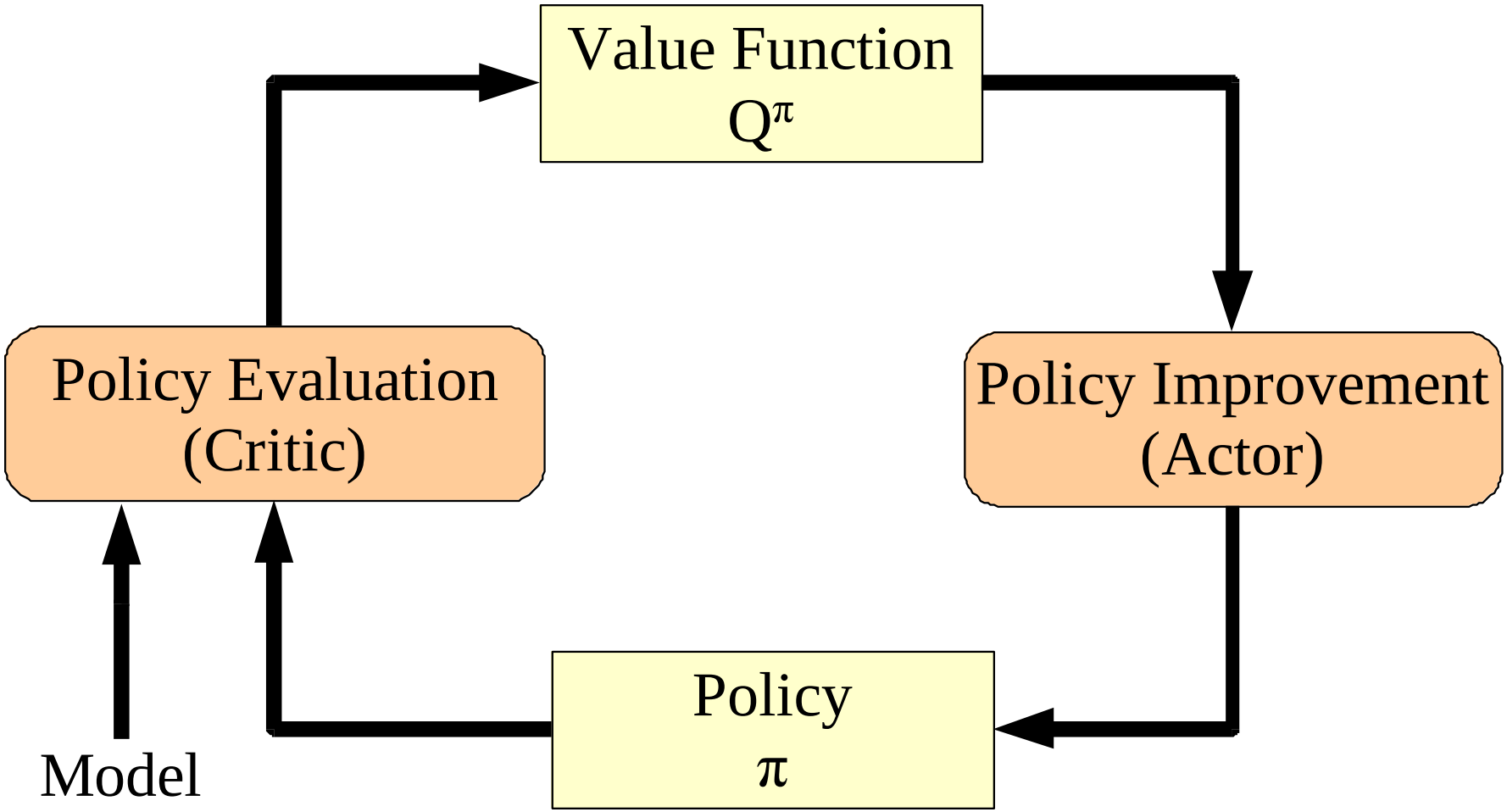
Any deterministic policy can be represented as a multi-class classifier

Policy Learning as Classifier Learning

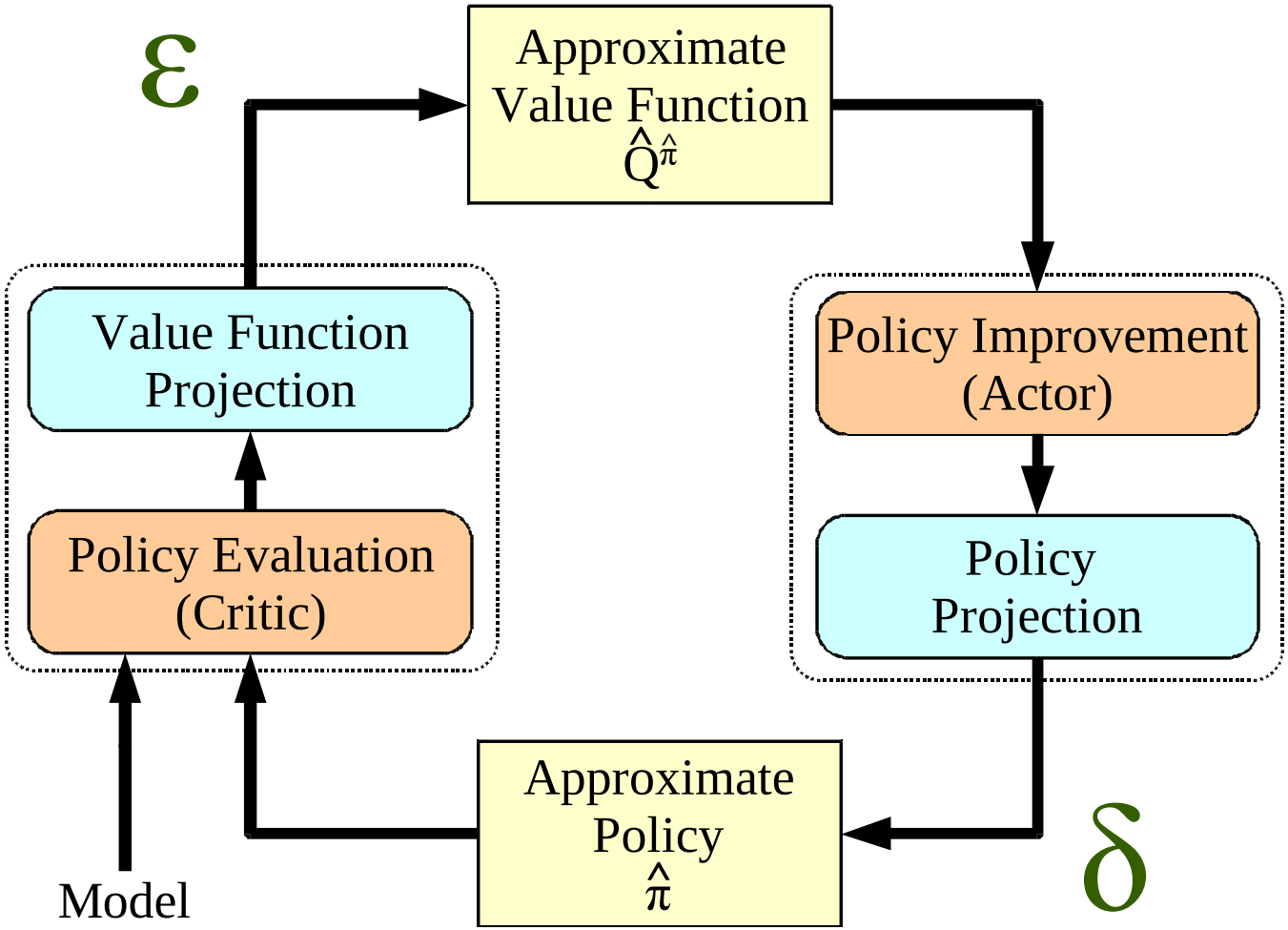


- **Input**
  - examples of the target policy at a subset of states
- **Learner**
  - your favorite classifier
- **Output**
  - generalization of the target policy over the entire state space

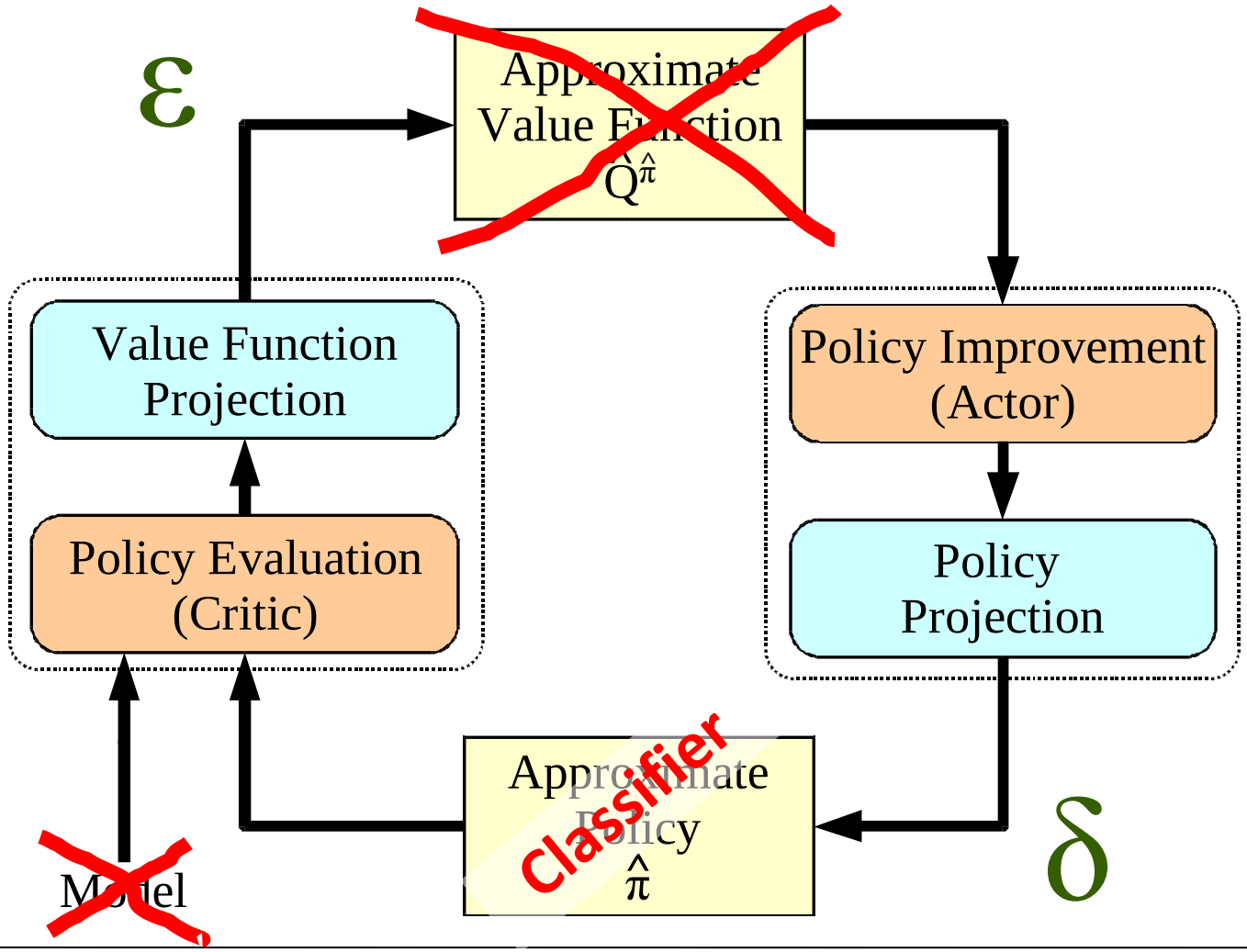
Policy Iteration



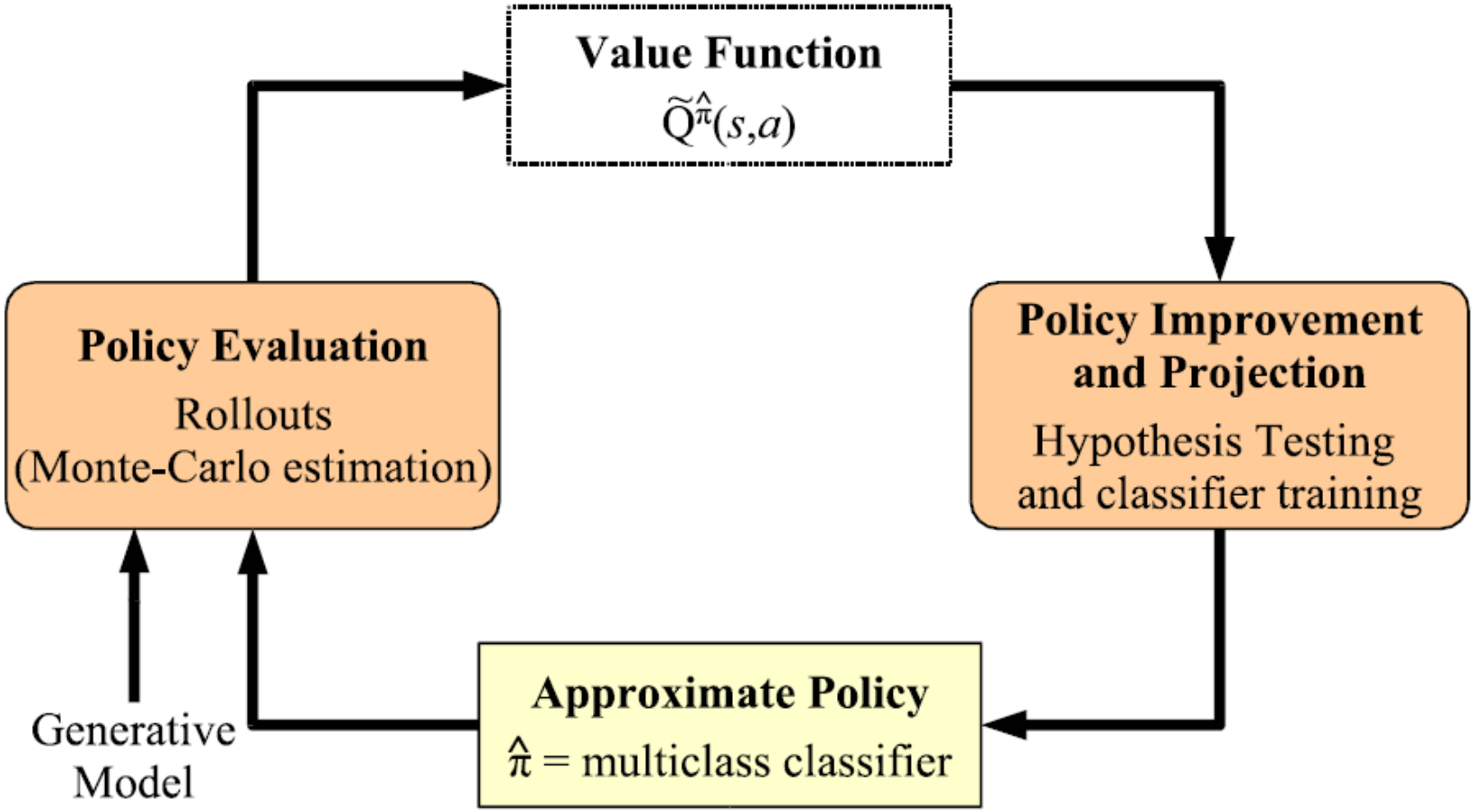
Approximate Policy Iteration



Approximate Policy Iteration



Rollout Classification Policy Iteration



Rollouts

Value function estimation

- Monte-Carlo estimation of $Q_\pi(\boldsymbol{s}, a)$ at any pair $(\boldsymbol{s}, a)$ for given π
- there is no full/explicit representation of a value function
- start in $\boldsymbol{s}$, choose a at first step, then choose according to π

Value from one trajectory

$$\widehat{Q}^i_\pi(\boldsymbol{s}, a) = \sum_{t=0}^{H} \gamma^t r_t \qquad H - \text{horizon (number of steps)}$$

Value from many trajectories

$$\widehat{Q}_\pi(\boldsymbol{s}, a) = \frac{1}{K} \sum_{i=1}^{K} \widehat{Q}^i_\pi(\boldsymbol{s}, a) \qquad K - \text{number of trajectories}$$

Action Domination

first, obtain $Q_\pi(\boldsymbol{s}, a)$ for all actions a in a given state $\boldsymbol{s}$ using rollouts:

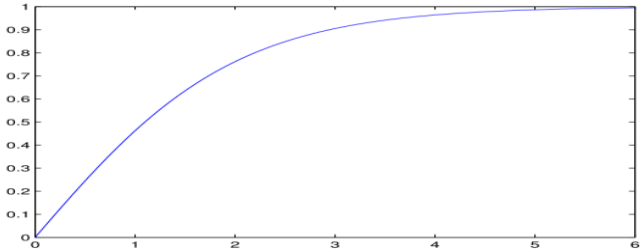
Maximum difference $f(\boldsymbol{s})$ of Q-values for state $\boldsymbol{s}$

$$f(\boldsymbol{s}) = \max_{a' \in \mathcal{A}} \left\{ \widehat{Q}_\pi(\boldsymbol{s}, a') \right\} - \min_{a'' \in \mathcal{A}} \left\{ \widehat{Q}_\pi(\boldsymbol{s}, a'') \right\}$$

Action advantage function $\Delta Q_\pi(\boldsymbol{s})$

$$\Delta Q(\boldsymbol{s}) = 2 \left(\frac{1}{1 + \exp(-f(\boldsymbol{s}))} - 0.5 \right)$$

$\Delta Q(\boldsymbol{s})$ is based on a scaled and shifted sigmoid



Filtering and Training Set

Removal of near zero ΔQ values

The purpose of filtering is to clear out noise in action domination.

$$\Delta Q(\boldsymbol{s}) \leq \epsilon$$

States with $\Delta Q_\pi(s) > \epsilon$ are included in the training set

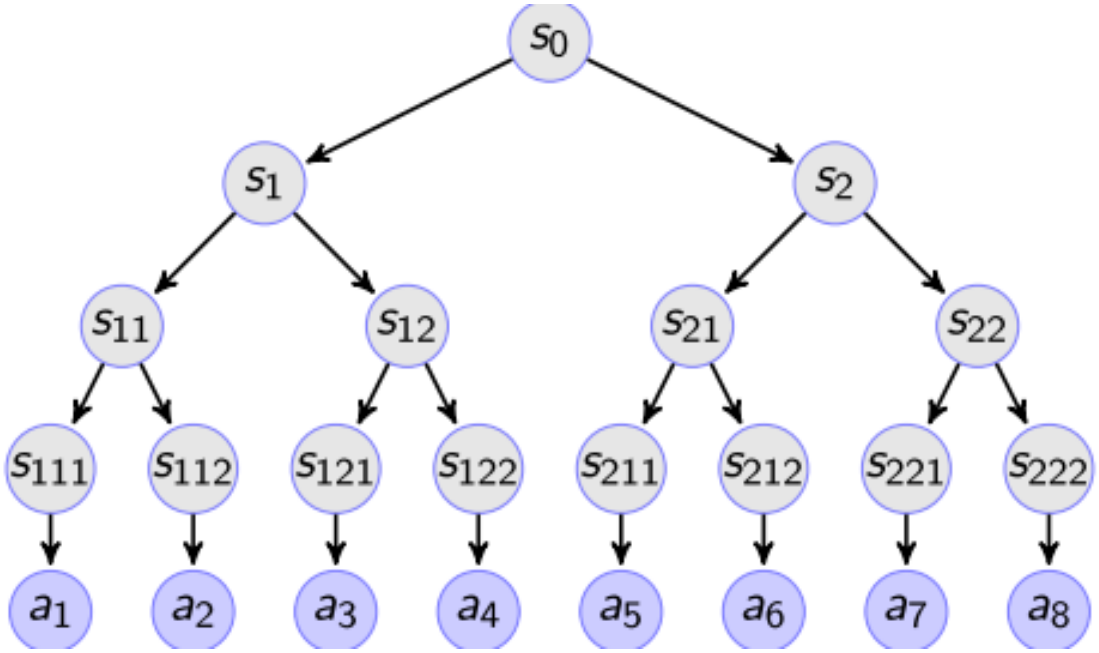
- any action $a^* = \arg\max_{a' \in \mathcal{A}} \{\widehat{Q}_\pi(\boldsymbol{s}, a')\}$ is *dominating*
  - yields a positive example $(\boldsymbol{s}, a^*)^+$
- any other action a is *dominated*
  - yields a negative example $(\boldsymbol{s}, a)^-$

Classifier-Based API Algorithms

- **RCPI-SVM**
  - SVMs for representing policies
  - support vectors direct the selection of rollout states
- **RCPI-RVM**
  - RVMs for representing policies (sparser)
  - RVM regression for advantage function
- **Localized Policy Iteration (not covered here)**
  - exploiting locality in policies
  - identification of the “ball” where action still dominates
  - ball-based classifier

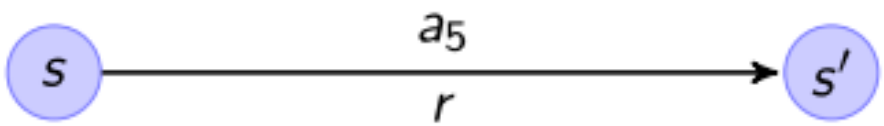
Extension to Continuous Action Spaces

- **Binary Action Selection**
  - fine discretization of continuous range and binary search
  - view of a continuous action as sequence of binary actions
  - generalizes to multi-dimensional action spaces

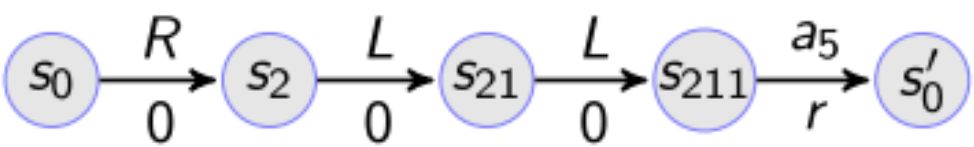


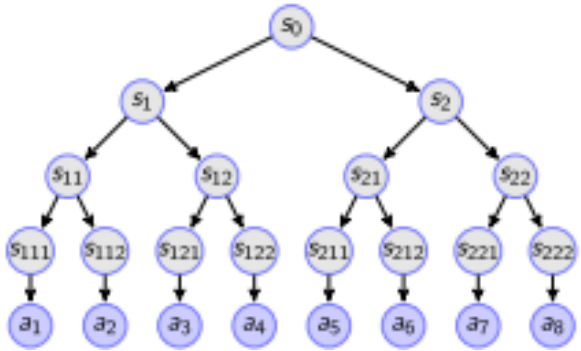
MDP Transformation

In the original MDP: (s, a, r, s')



In the transformed MDP:
$(s_0, R, 0, s_2), (s_0, R, 0, s_2), (s_2, L, 0, s_{21}), (s_{21}, L, 0, s_{211}), (s_{211}, a_5, r, s_0')$





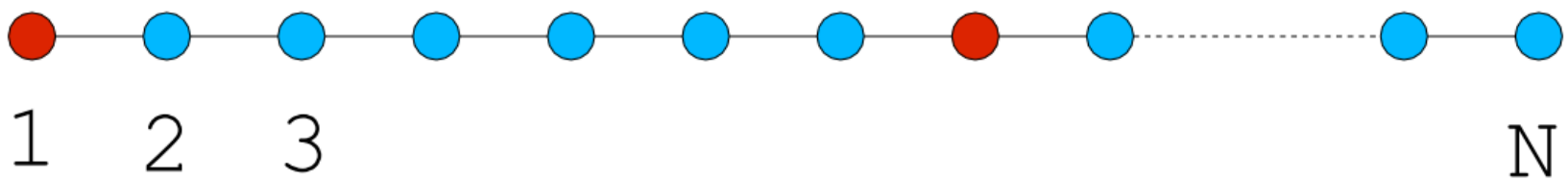
Experimentation

Put RL to work!

RL Experimentation

- **Application Domains**
  - chain walk
  - inverted pendulum
  - bicycle balancing and riding
  - mountain car
  - acrobot
  - tetris
  - othello/reversi
  - simulated soccer
  - load balancing
  - recursive algorithm selection

N-State Chain Walk



Walk along the chain and maximize the expected return!

- **S** = {1, 2, 3, …, N} discrete positions, linearly ordered
- **A** = {Left, Right} move left or move right
- **noise**: 90% action success, 10% failure (opposite direction)
- **reward**: +1 at selected (red) positions, 0 otherwise
- **γ** = 0,9

simple problem, fully solvable, ideal for comparisons

20-State Chain Walk: V Function

Samples

- 5000 random moves

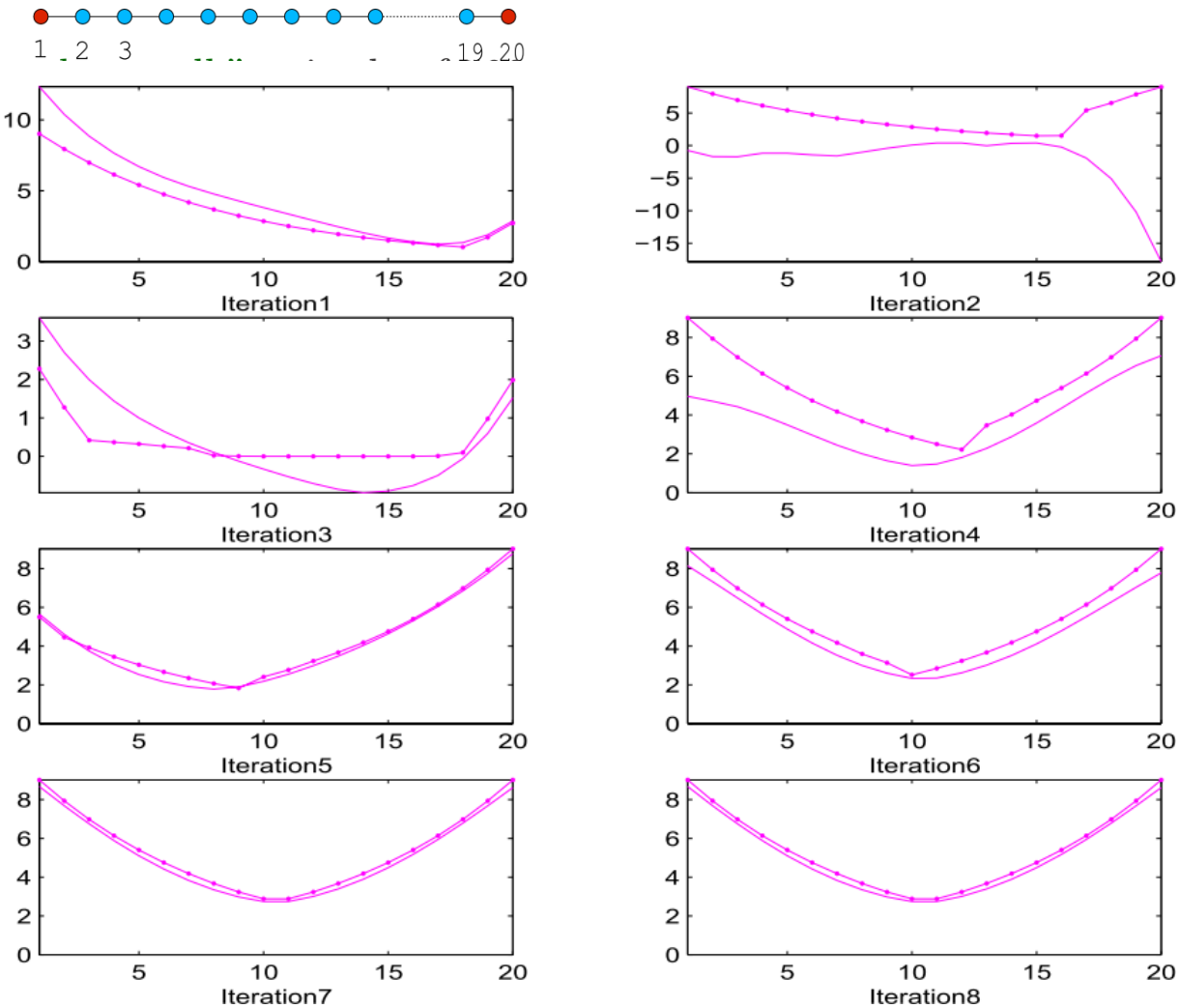
Approximation

- 10 basis functions
- 4th degree polynomial for each action

Results

- initial policy: Left
- V function over iterations
- convergence: 8 iterations

- exact: dotted line
- LSPI: solid line



20-State Chain Walk: Policies

Samples

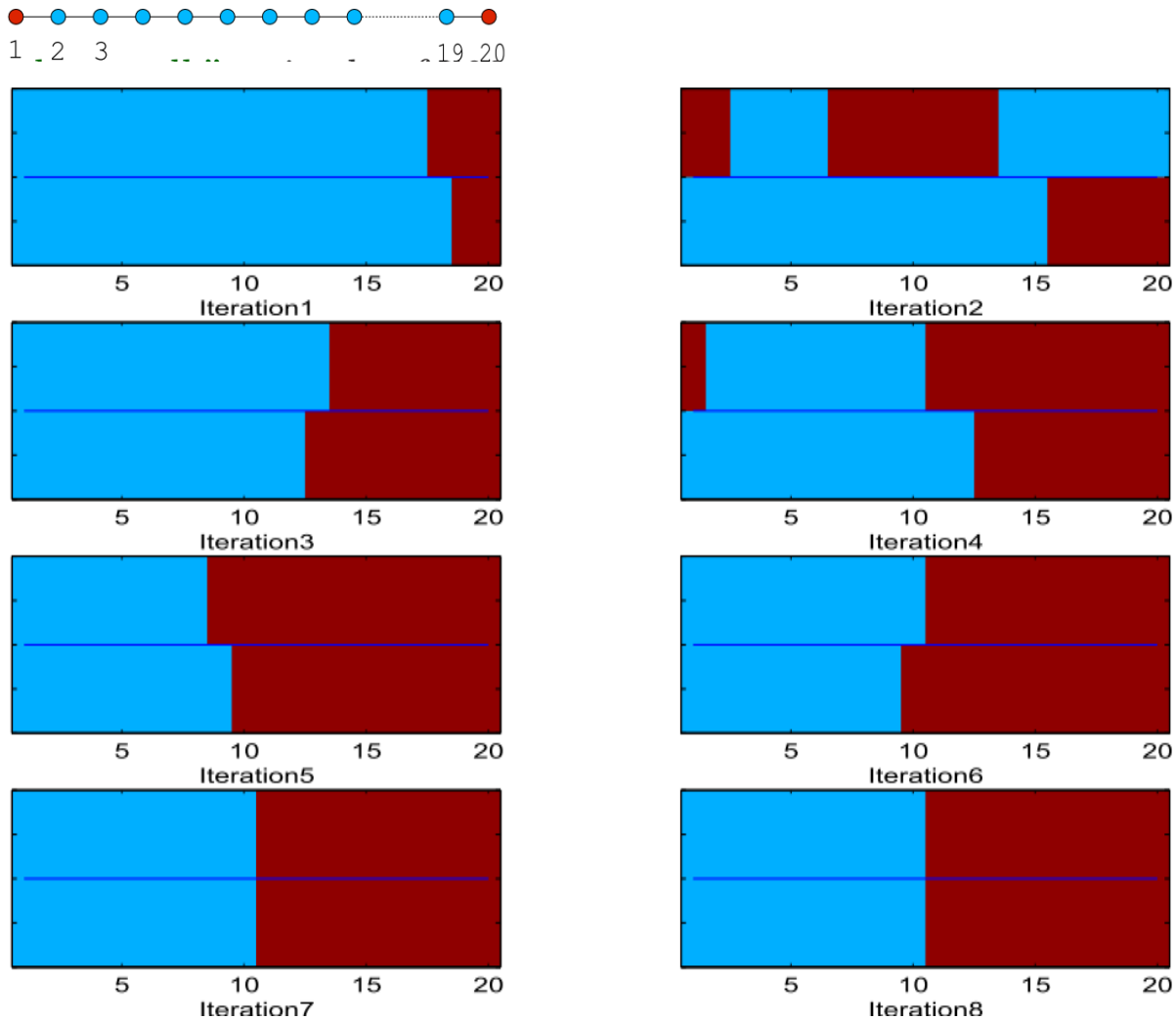
- 5000 random moves

Approximation

- 10 basis functions
- 4th degree polynomial for each action

Results

- initial policy: Left
- policies over iterations
- convergence: 8 iterations
- outcome: **optimal** policy
- Left: blue, Right: red
- exact: top, LSPI: bottom



50-State Chain Walk: Q Function

Samples

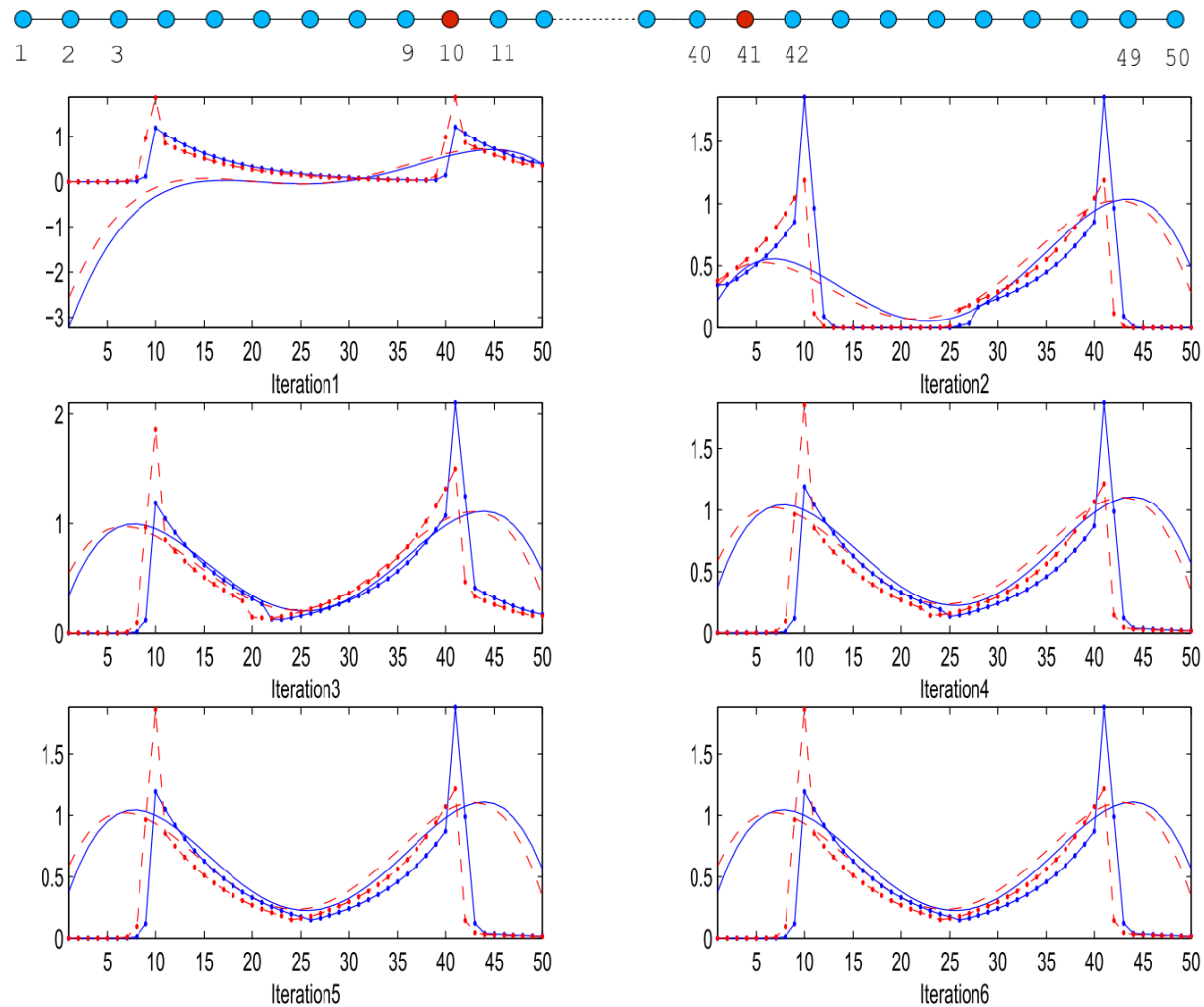
- 10000 random moves

Approximation

- 10 basis functions
- 4th degree polynomial for each action

Results

- initial policy: Left
- Q function over iterations
- convergence: 6 iterations
- exact: dotted line
- LSPI: solid line
- Left: blue, Right: red



50-State Chain Walk: Policies

Samples

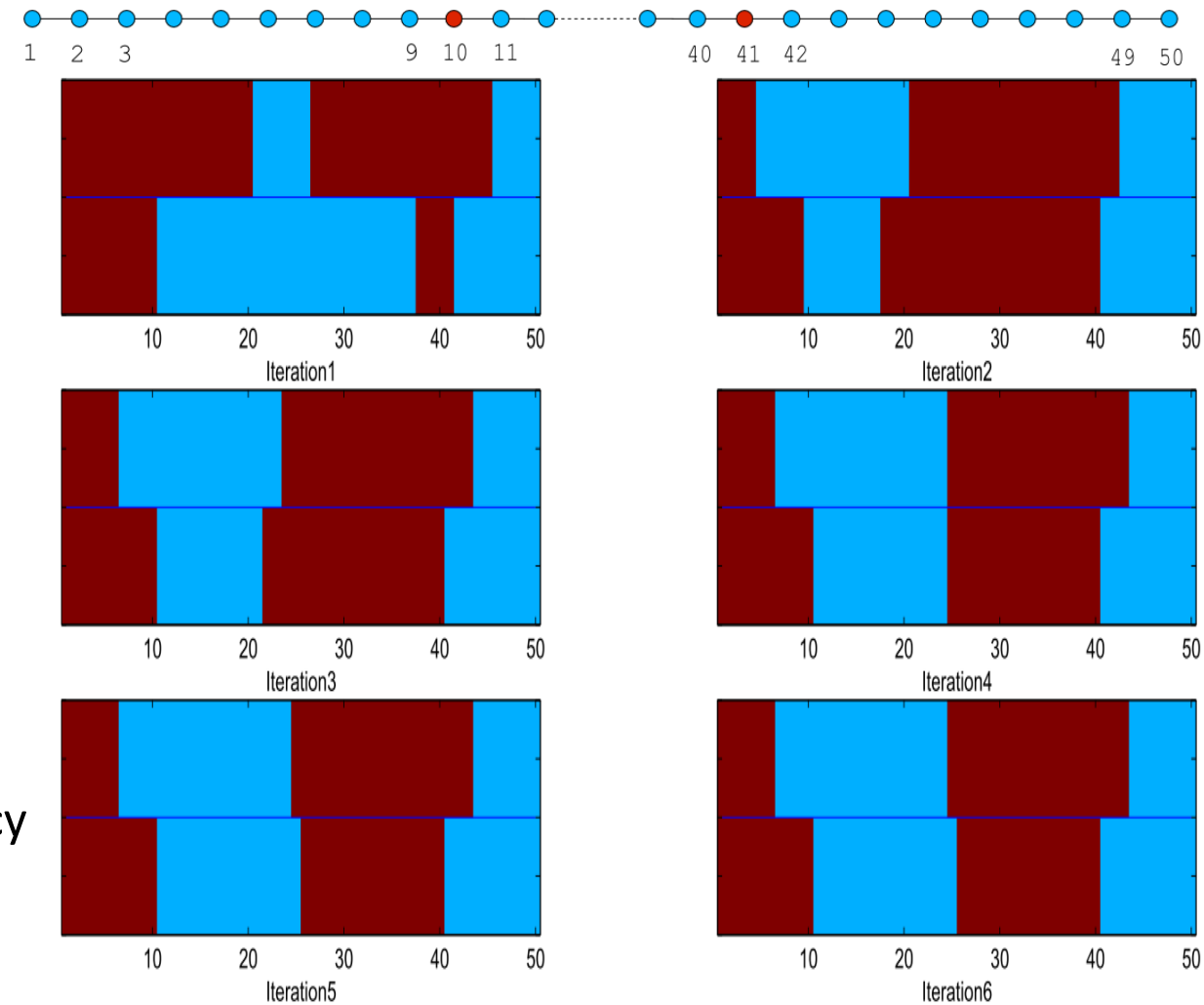
- 10000 random moves

Approximation

- 10 basis functions
- 4th degree polynomial for each action

Results

- initial policy: Left
- policies over iterations
- convergence: 6 iterations
- outcome: **suboptimal** policy
- Left: blue, Right: red
- exact: top, LSPI: bottom



50-State Chain Walk: Q Function

Samples

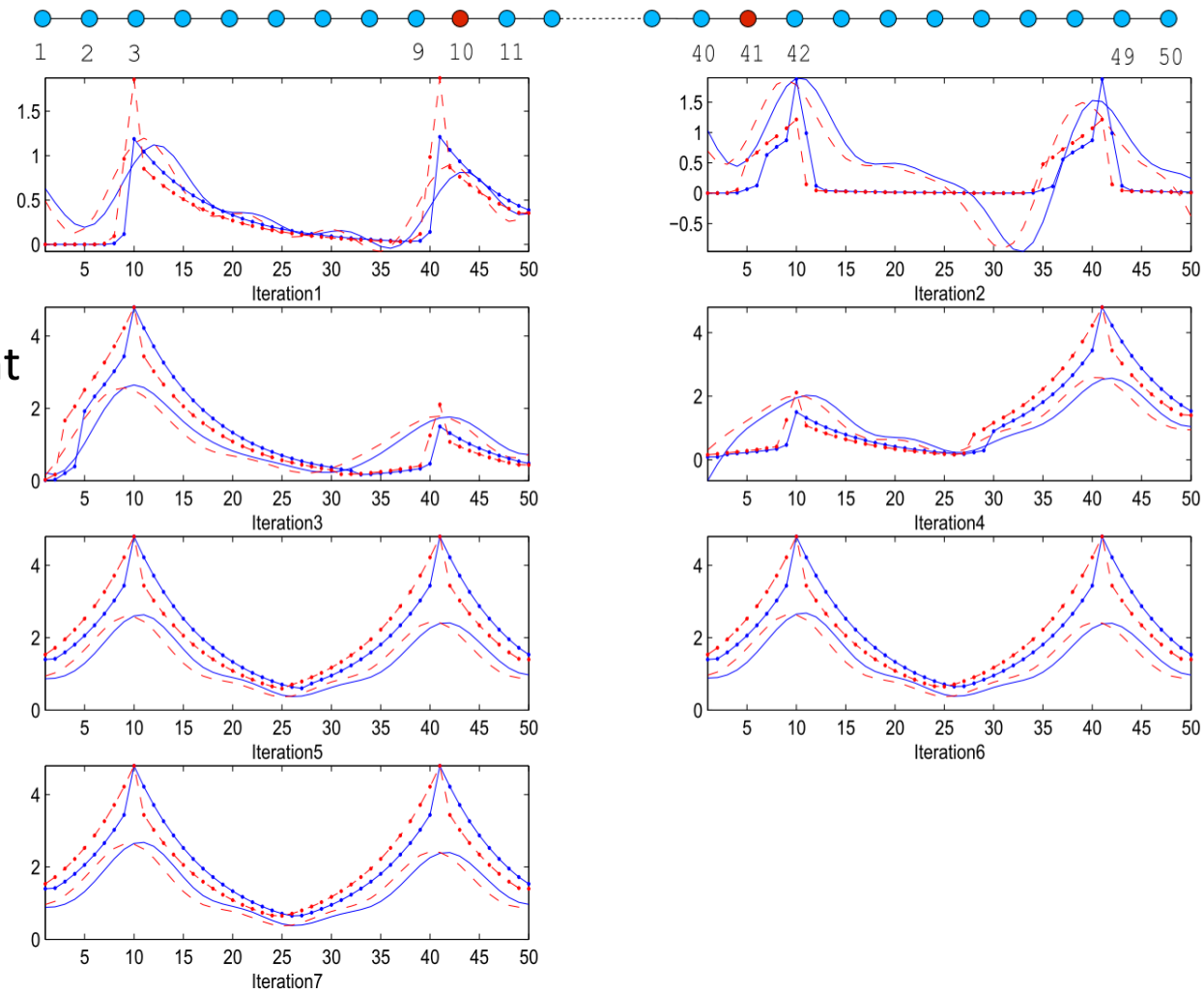
- 10000 random moves

Approximation

- 22 basis functions
- 10 uniform RBFs + constant for each action

Results

- initial policy: Left
- Q function over iterations
- convergence: 7 iterations
- exact: dotted line
- LSPI: solid line
- Left: blue, Right: red



50-State Chain Walk: Policies

Samples

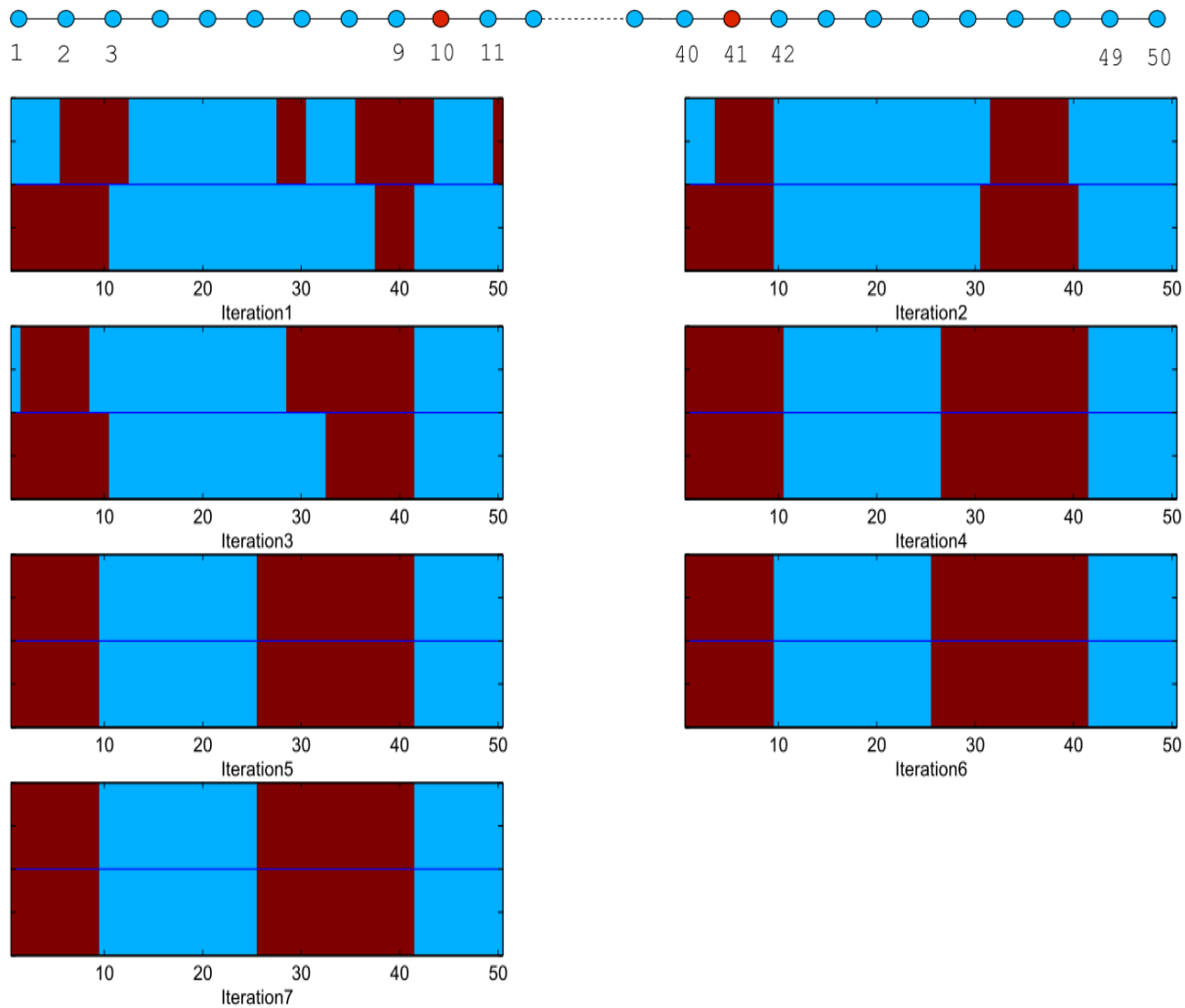
- 10000 random moves

Approximation

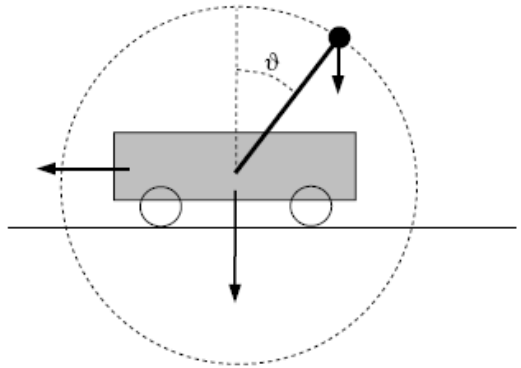
- 22 basis functions
- 10 uniform RBFs + constant for each action

Results

- initial policy: Left
- policies over iterations
- convergence: 7 iterations
- outcome: **optimal** policy
- Left: blue, Right: red
- exact: top, LSPI: bottom



Inverted Pendulum



Balance the pendulum at the upright position!

- $\mathcal{S} = \{(\text{angle } \theta, \text{angular velocity } \dot{\theta})\}$
- $\mathcal{A} = \{-50 \text{ N}, \ 0 \text{ N}, \ +50 \text{ N}\}$
- Noise: Input $u = a + 10n$ (Gaussian)
- Reward: -1 if $|\theta| > \frac{\pi}{2}$, 0 otherwise
- $\gamma = 0.9$
- Model: non-linear dynamics [Wang et al., 1996]

Pendulum: Learning Parameters

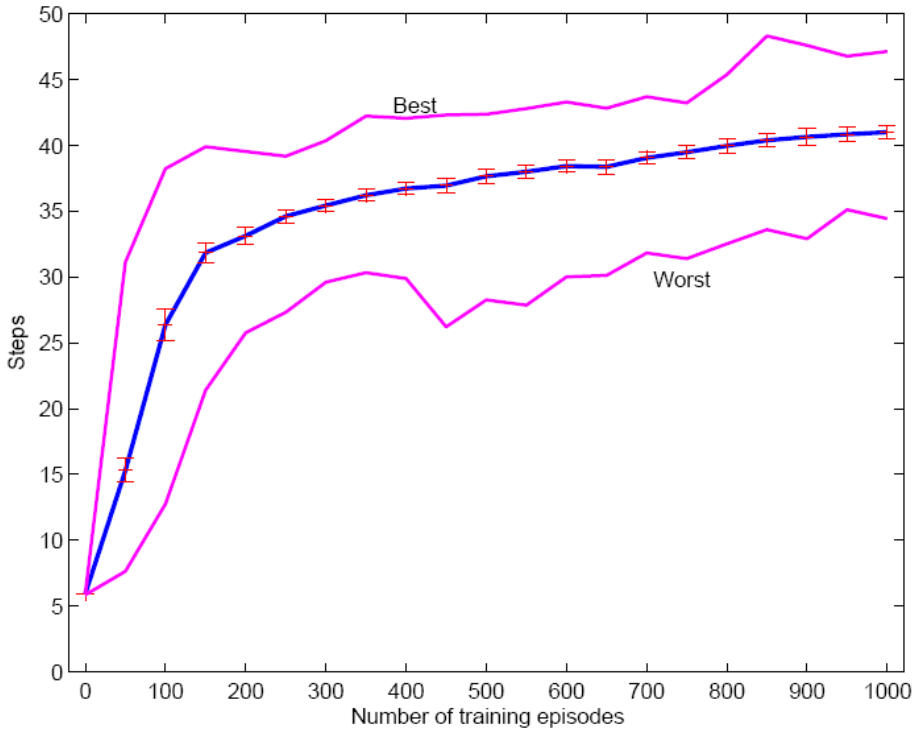
- Features: $k = 30$ (10 basis functions for each action)
  - a constant term
  - nine radial basis functions (Gaussians with $\sigma^2 = 1$)

    3×3 grid: $\mu_i \in \{-\frac{\pi}{4},\ 0,\ +\frac{\pi}{4}\} \times \{-1,\ 0,\ +1\}$

$$\left(1,\ e^{-\frac{||s-\mu_1||^2}{2\sigma^2}},\ e^{-\frac{||s-\mu_2||^2}{2\sigma^2}},\ e^{-\frac{||s-\mu_3||^2}{2\sigma^2}},\ \ldots,\ e^{-\frac{||s-\mu_9||^2}{2\sigma^2}} \right)$$

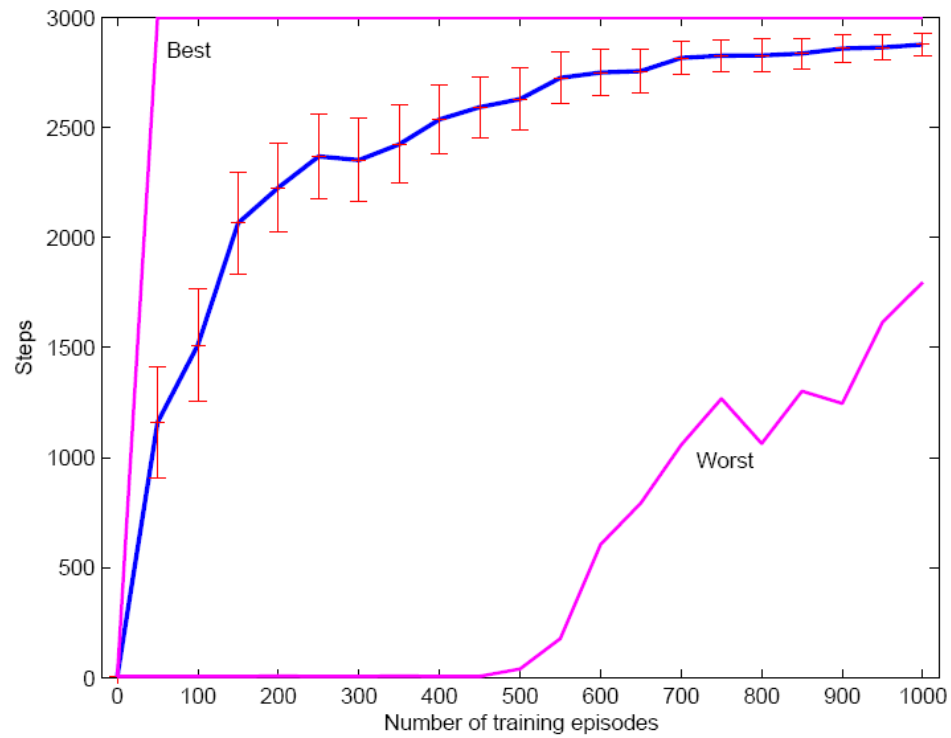
- Samples: Collected from random episodes:
  - starting at the upright position and ...
  - following a purely random policy ...
  - until the episode ends.

Pendulum: Results

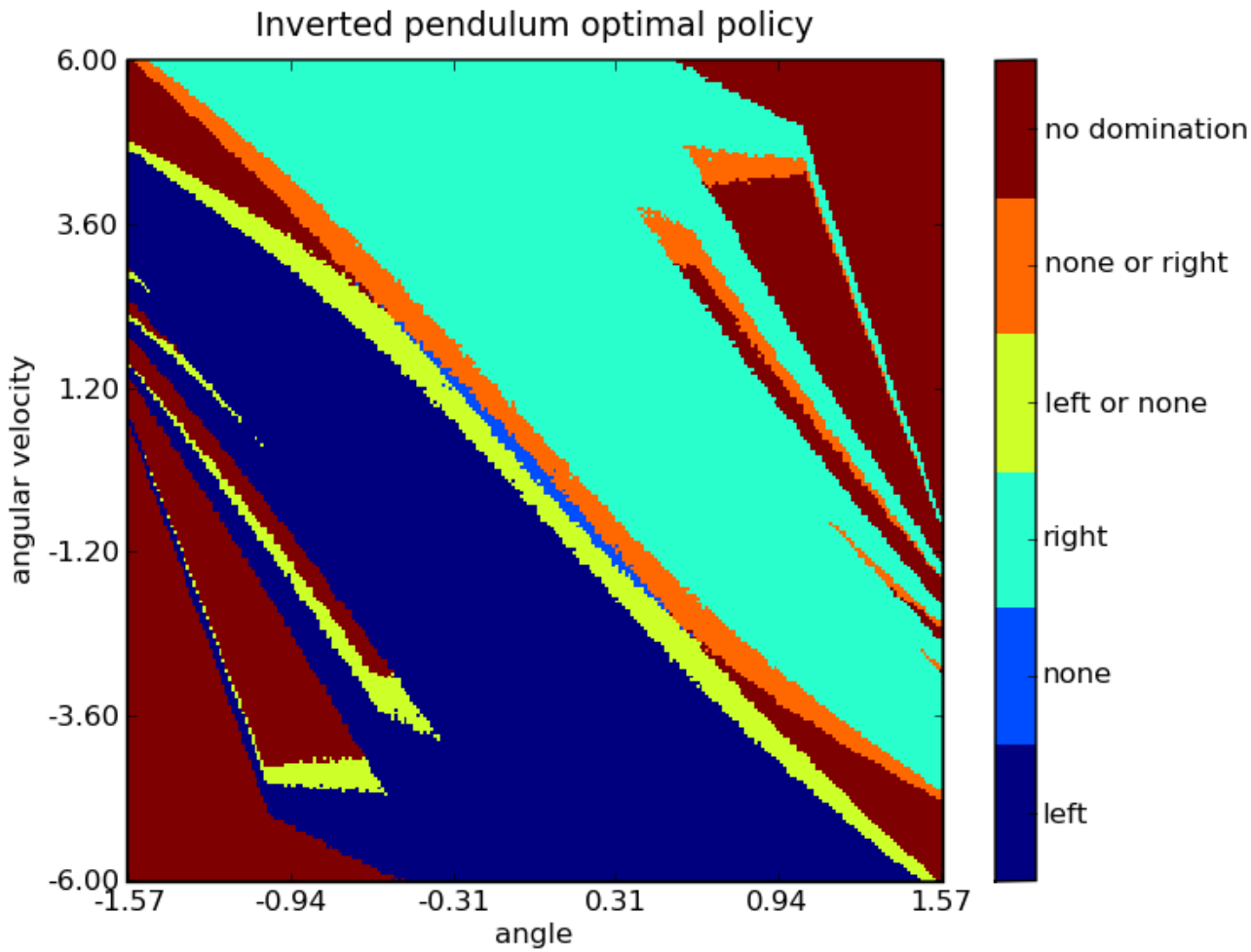


Q-learning

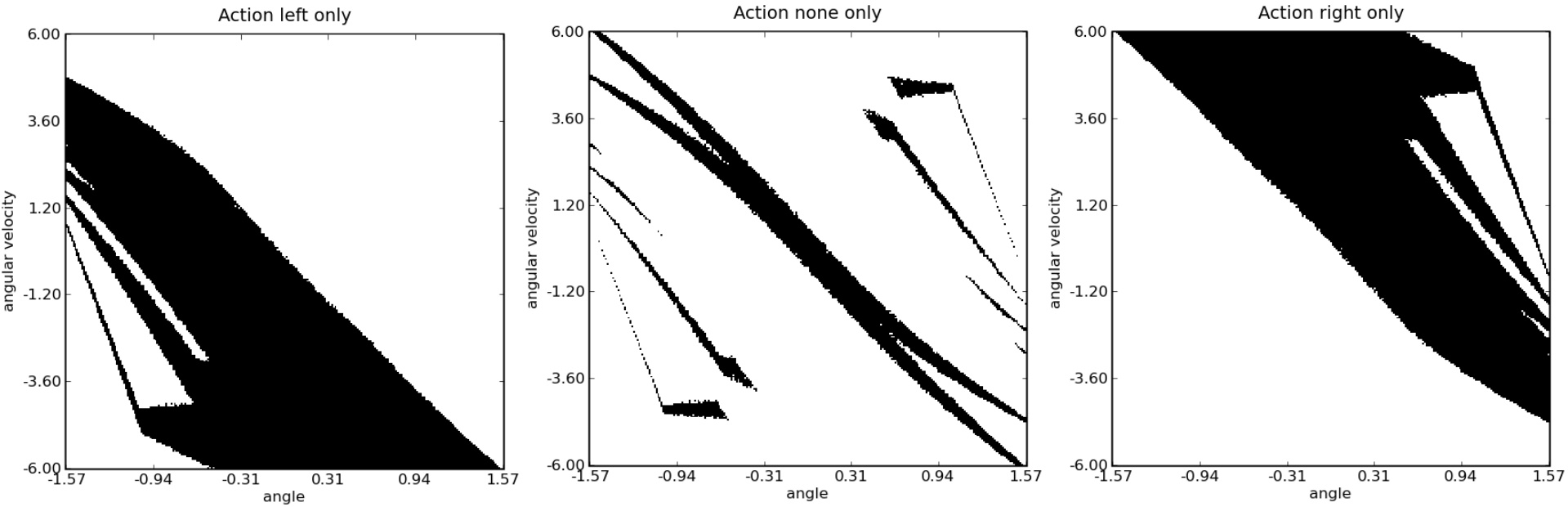
LSPI



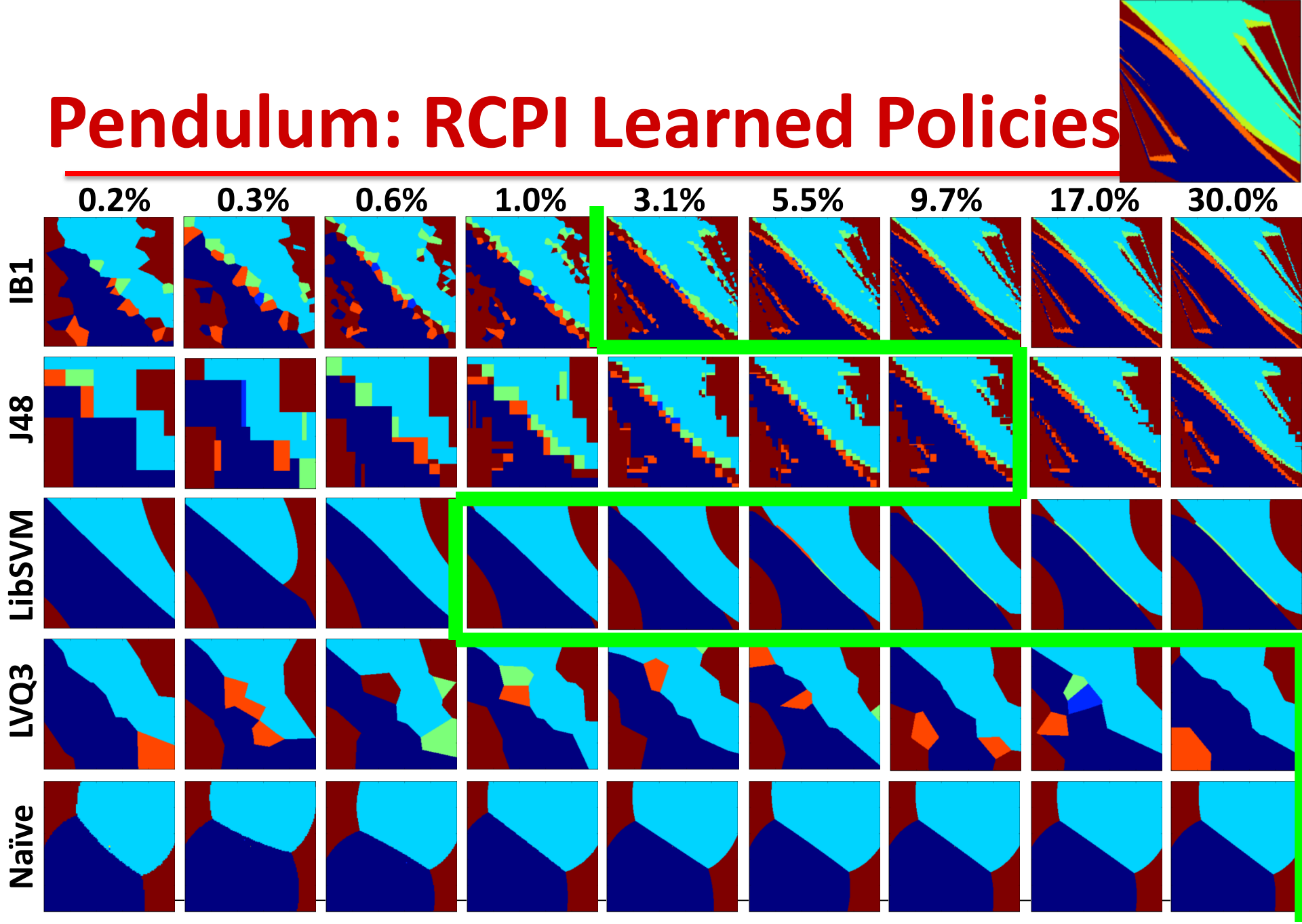
Pendulum: Optimal Policy



Pendulum: Optimal Actions



Pendulum: RCPI Learned Policies



Pendulum: Nao Robot Balancing

Bicycle Balancing and Riding

Balance and ride a bicycle at a target location 1 Km away!

- $\mathcal{S} = \{(\theta, \dot{\theta}, \omega, \dot{\omega}, \ddot{\omega}, \psi)\}$

  θ: angle of the handlebar, ω: vertical angle, ψ: angle to the goal

- $\mathcal{A} = \{(\tau, \upsilon)\}$

  $\tau \in \{-2, 0, +2\}$: torque, $\upsilon \in \{-0.02, 0, +0.02\}$: displacement

- Model: non-linear dynamical system [Randløv and Alstrøm, 1998]

- Noise: input $(\tau, \upsilon + n)$, $n \in [-0.02, +0.02]$

- Reward:

  – the net change in ω^2, plus ...

  – 1% of the net change in the distance to the goal

- $\gamma = 0.8$

Bicycle Learning Parameters

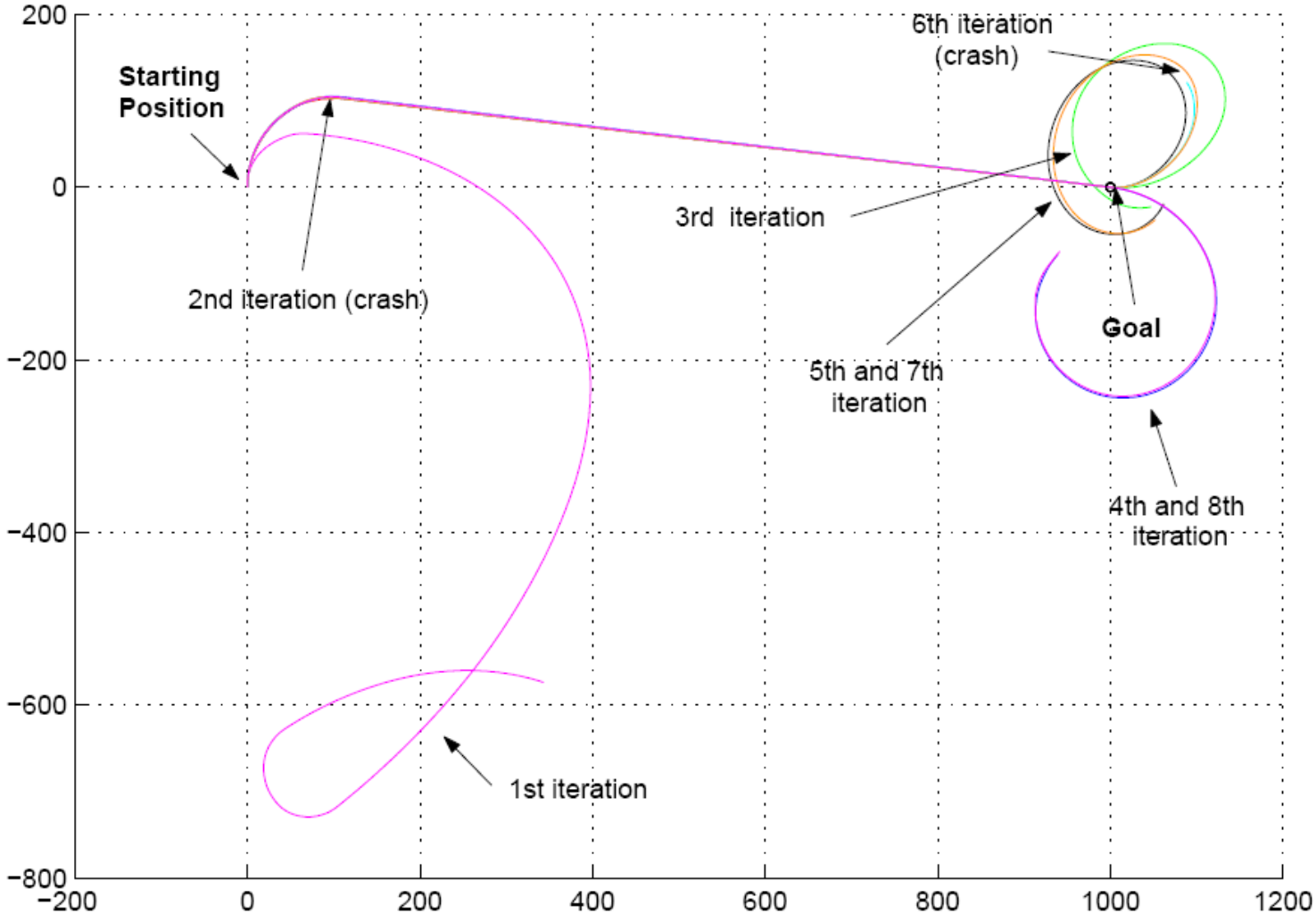
- **Features**
  - k=100 (20 basis functions for each action)

$$(1,\ \omega,\ \dot{\omega},\ \omega^2,\ \dot{\omega}^2, \omega\dot{\omega},\ \theta,\ \dot{\theta},\ \theta^2,\ \dot{\theta}^2,\ \theta\dot{\theta},\ \omega\theta,\ \omega\theta^2,\ \omega^2\theta,\ \psi,\ \psi^2,\ \psi\theta,\ \bar{\psi},\ \bar{\psi}^2,\ \bar{\psi}\theta)$$

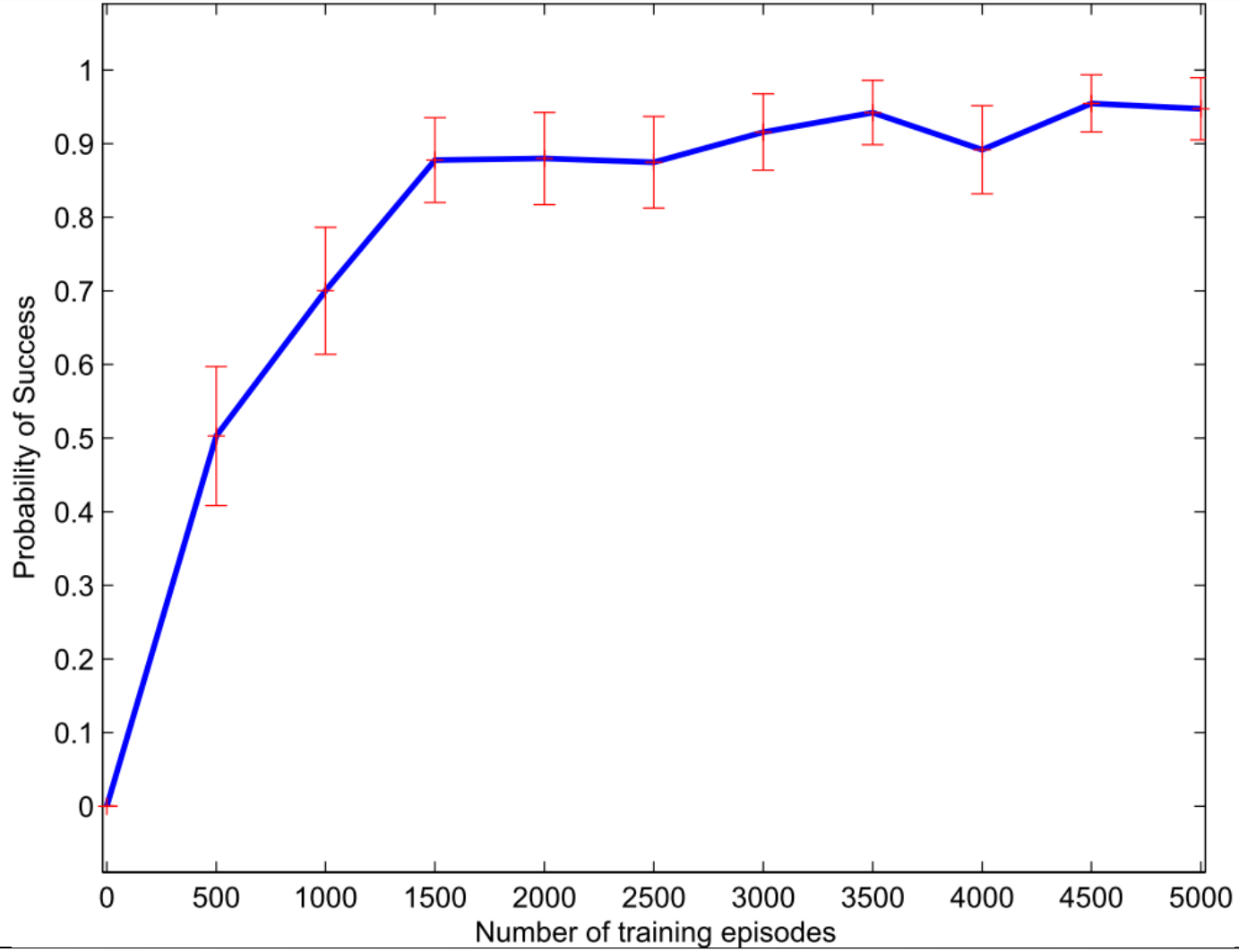
$$\bar{\psi} = \pi - \psi \ \text{ for } \ \psi > 0 \quad \text{and} \quad \bar{\psi} = -\pi - \psi \ \text{ for } \ \psi < 0$$

- **Samples**
  - collected from random episodes
    - starting at a random state around the initial position
    - following a purely random policy for only 20 steps
  - only 20 minutes worth of operating time!

Bicycle Learning Results



Bicycle Learning Performance



Tetris

Learn to play the game of Tetris!

- $\mathcal{S}$: $\approx 10^{61}$ states
- $\mathcal{A}$: ≈ 40 actions
- Noise: the next object is chosen randomly
- Reward: $+1$ for each completed row, 0 otherwise
- $\gamma = 1$
- Model: It is known!

Tetris: Learning Parameters

- Features: $k = 10$ basis functions defined over (s, a)

  1. the constant 1.0
  2. the number of rows completed if a is taken in s
  3. the maximum height
  4. the difference in #3 if a is taken in s
  5. the total number of "holes"
  6. the difference in #5
  7. the mean height of the board
  8. the difference in #7
  9. the sum of absolute height differences between adjacent columns
  10. the difference in #9

- Samples: Complete games of a handcoded player
  - Handcoded player: $w = [0, 2, 0, -1, 0, -4, 0, 0, 0, -1]$
  - Handcoded player scores about 675 points on average
  - Random player does not provide sufficient coverage

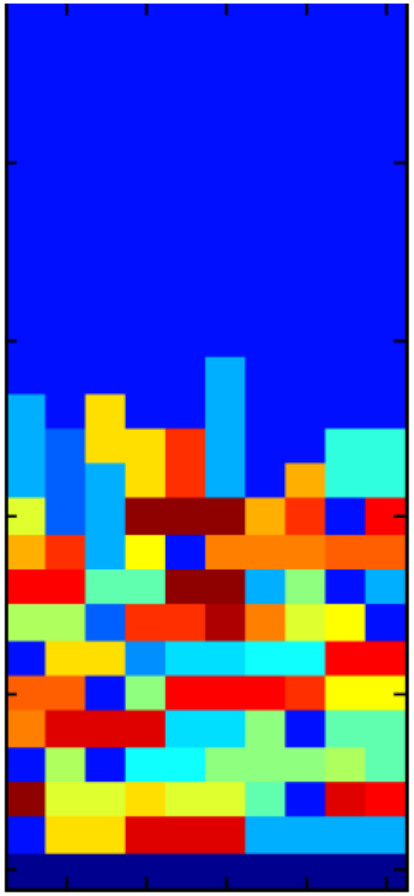
Tetris: Results

Handcoded Player

Average Score ≈ 675

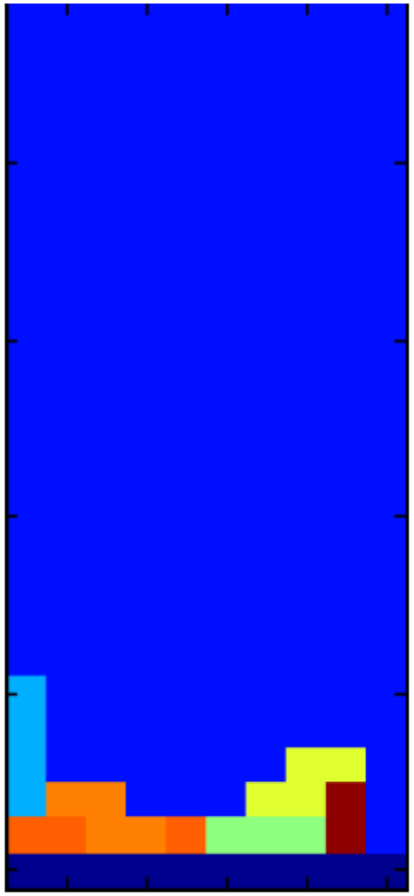
Score = 49

(after 150 steps)



Learned Player

Average Score ≈ 3000

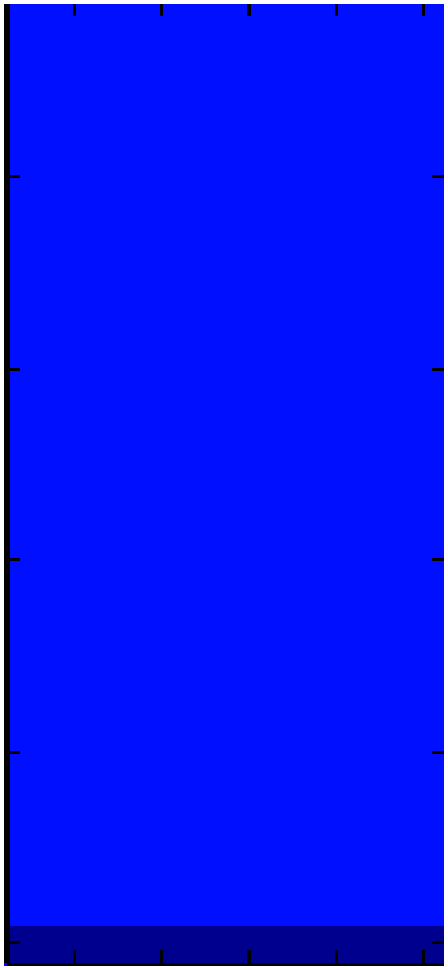


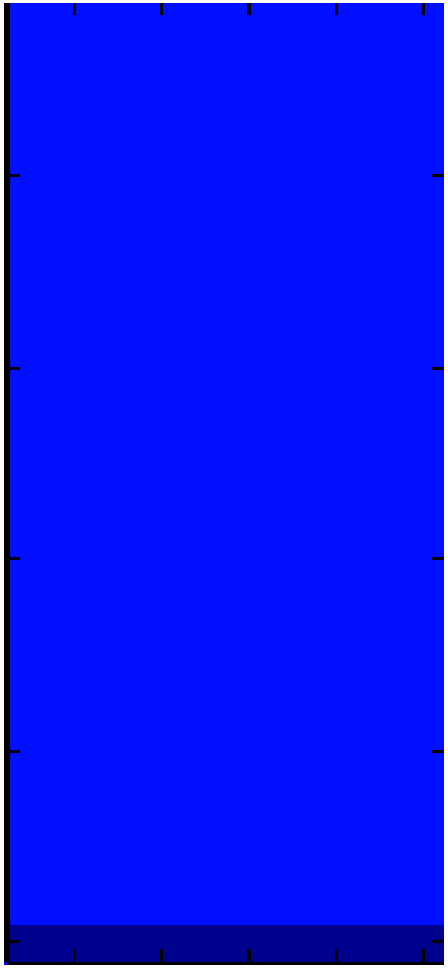
Score = 58

(after 150 steps)

Tetris: Handcoded and LSPI

Handcoded Video



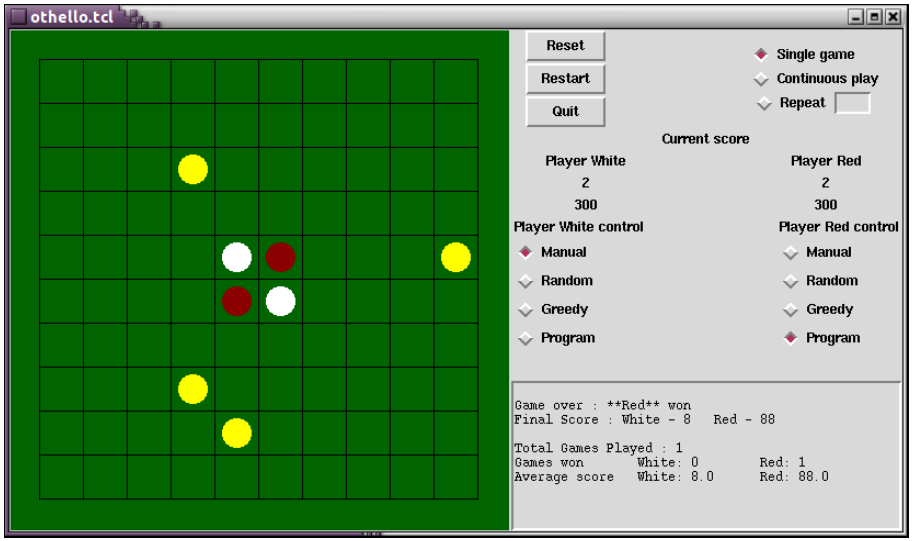


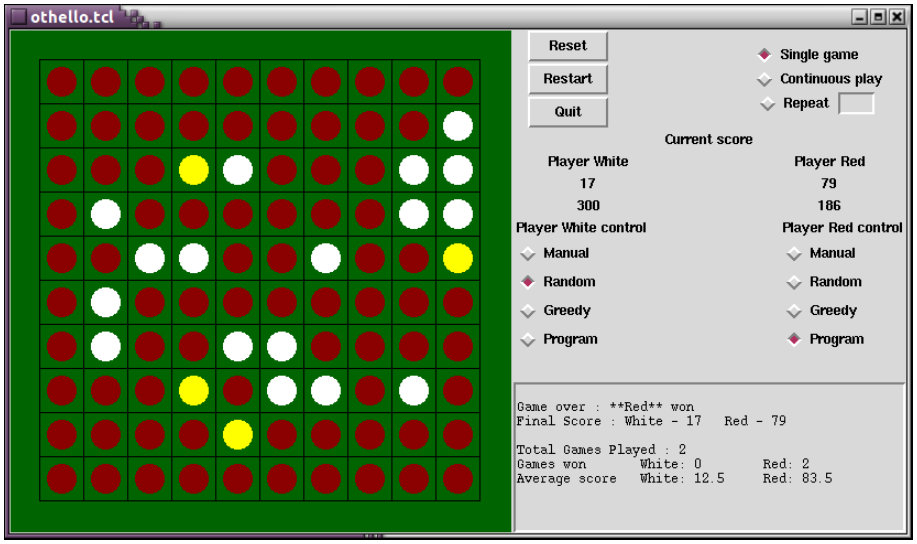
LSPI Video

Tetris: Learned Weights

#	Feature	Handcoded	Learned
1	constant 1.0	0	1946.70
2	number of rows completed	2	388.43
3	maximum height	0	-3.36
4	the difference in #3	-1	-4.82
5	total number of "holes"	0	-68.40
6	the difference in #5	-4	-111.74
7	mean height	0	-10.92
8	the difference in #7	0	379.08
9	sum of absolute height differences	0	-22.02
10	the difference in #9	-1	-20.79

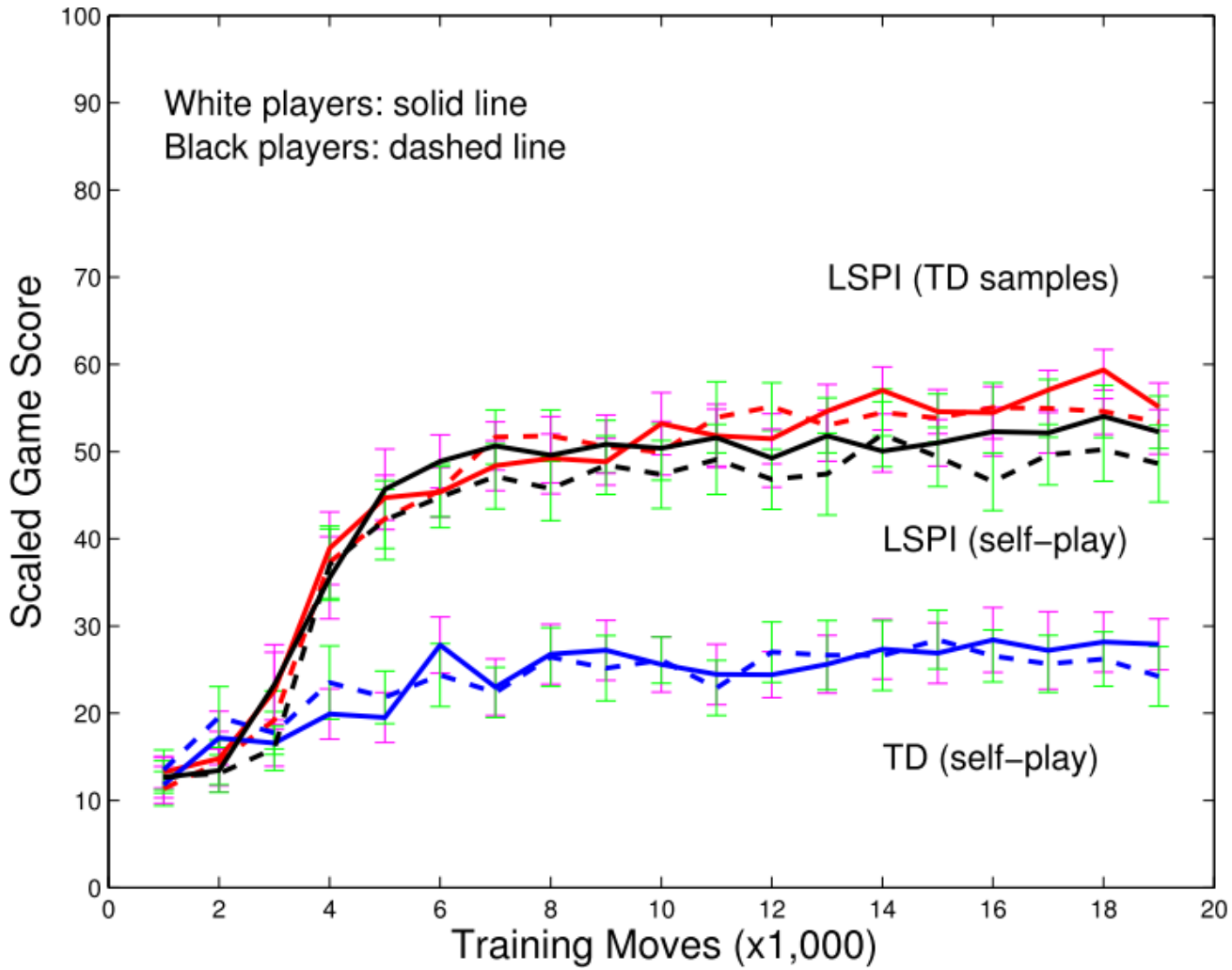
Othello/Reversi



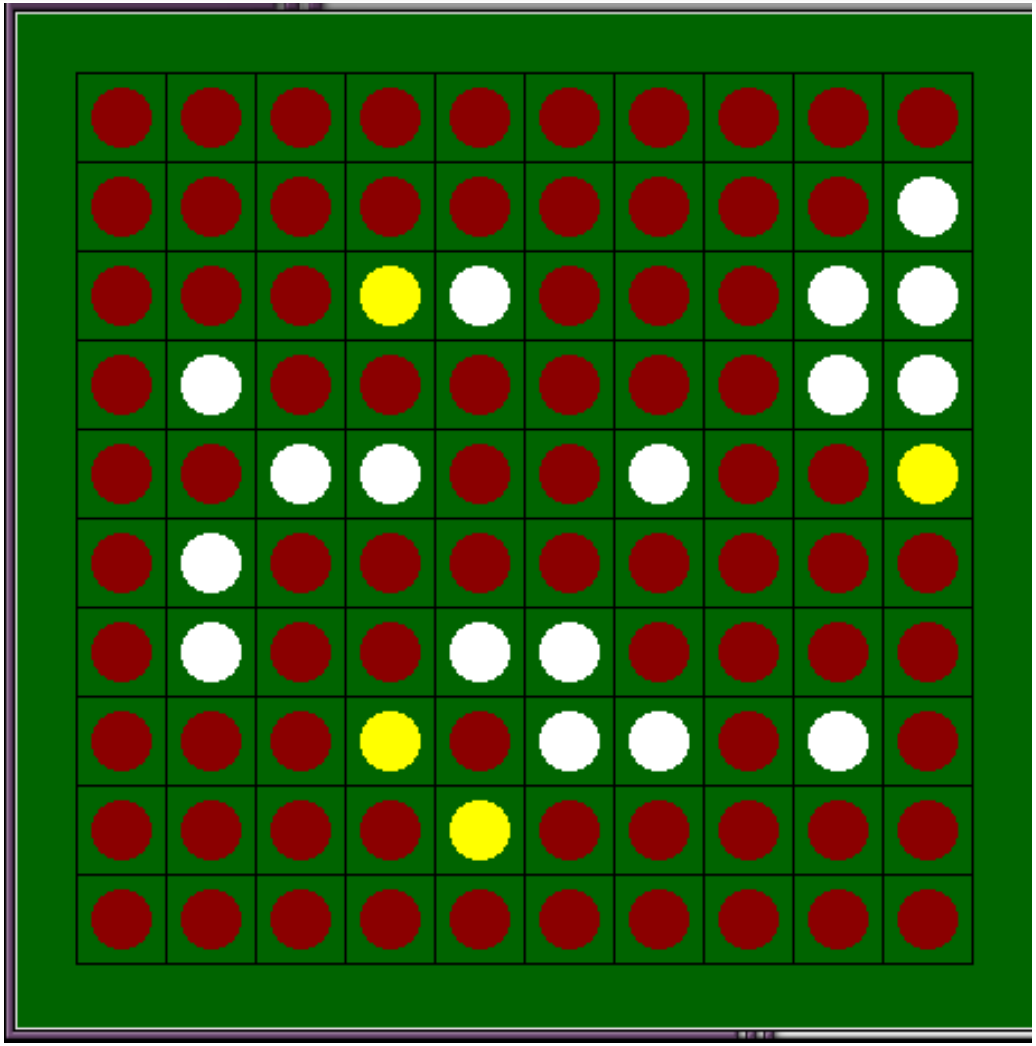


- states: ~10^{30} in 8x8 board, ~10^{47} in 10x10 board
- actions: from 0 to a few dozens
- noise: the unknown opponent, possibly random blocked cells
- reward: score at the end of the game
- known model: combine with minimax and α-β pruning

Othello/Reversi: Results



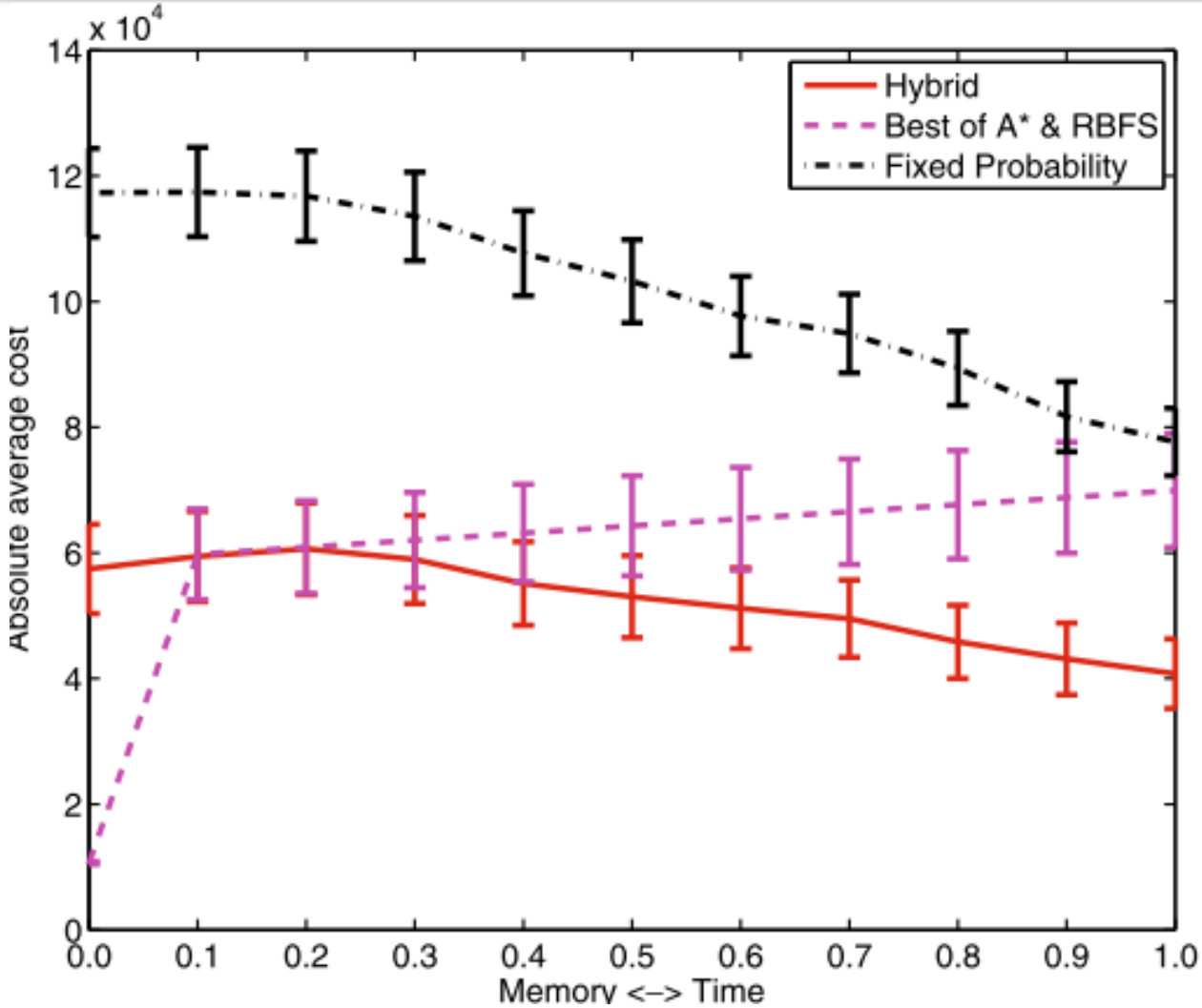
Othello/Reversi: Performance



Algorithm Optimization

- **Algorithm Selection**
  - which algorithm to choose (out of many) for a given instance?
- **Recursive Algorithm Selection**
  - recursive calls lead to a sequential (tree) decision problem
  - recursive algorithm selection: learn an algorithm selection policy
  - learn by solving a number of instances on the actual system
  - adapt to machine/cpu/memory/network (autonomic computing)
  - learned policy solves instances using combinations of algorithms
  - learned hybrid algorithm outperforms all individual algorithms
- **Problems**
  - sorting, order-statistic selections, satisfiability
  - systematic heuristic search (A*, RBFS)

Algorithm Optimization: Results



RL Connections

RL and (Un)Supervised Learning

- **Model-based RL**
  - density estimation for learning models
- **Value Functions**
  - dimensionality reduction for extracting features
  - feature spaces for approximating value functions
  - regression techniques for estimating values
  - …
- **Policies**
  - classifiers for representing policies
  - density estimation for representing policies
  - …

Study

- **Books**
  - R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge 1998
  - D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, Belmont, Massachusetts, 1996
- **Articles**
  - L. Kaelbling, M. Littman, A. Moore, *Reinforcement Learning: A Survey*, Journal of Artificial Intelligence Research **4**, 237–285, 1996
  - M. Lagoudakis, *Value Function Approximation*, in Encyclopedia of Machine Learning, Springer, 2010, pp. 1011-1021
  - S. Bradtke, A. Barto, *Linear Least-Squares Algorithms for Temporal Difference Learning*, Machine Learning, **22**: 1-3, 33-57, 1996
  - M. Lagoudakis and Ronald Parr, *Least-Squares Policy Iteration,* Journal of Machine Learning Research **4**, 1107-1149, 2003
  - M. Lagoudakis and Ronald Parr, *Reinforcement Learning as Classification: Leveraging Modern Classifiers*, Intl Conf on Machine Learning (ICML) 2003, p.424
  - V. Vasilikos and M. Lagoudakis, *Optimization of Heuristic Search using Recursive Algorithm Selection and Reinforcement Learning*, Annals of Mathematics and Artificial Intelligence, **60** (1-2), 2010.

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 24

Deep Learning

Neural Networks for Tabular Data

TUC ECE, Spring 2023

Today

- **From Perceptrons to MLPs**
- **Back Propagation**
- **Deep Considerations**

Neural Networks: the key idea

- **Linear models**
  - the input-output mapping is linear
- **Linear models in features spaces**
  - the input is transformed non-linearly in a feature space
  - the output is still linear in the parameters over the features
- **Beyond that?**
  - endow the feature extractor with its own parameters

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \mathbf{W}\boldsymbol{\phi}(\boldsymbol{x};\boldsymbol{\theta}_2) + \boldsymbol{b} \qquad \text{where } \boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) \text{ and } \boldsymbol{\theta}_1 = (\mathbf{W}, \boldsymbol{b})$$

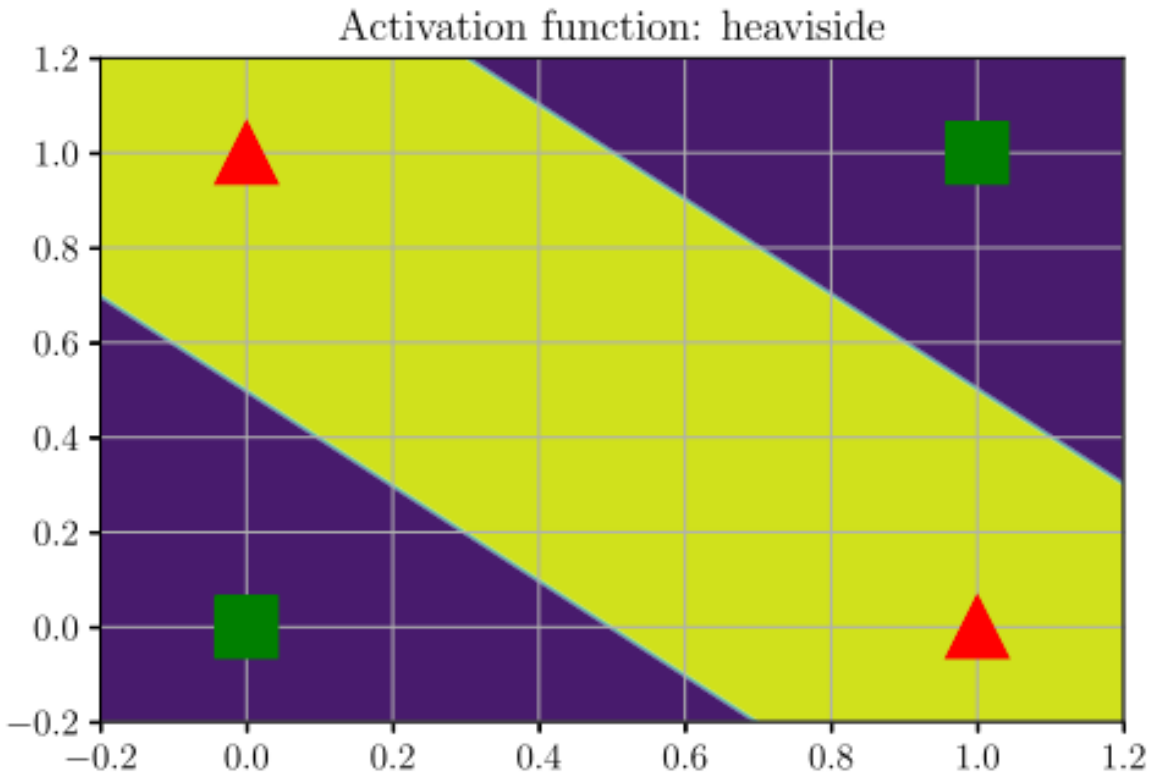
  - repeat this process recursively, to create complex functions

$$f(\boldsymbol{x};\boldsymbol{\theta}) = f_L(f_{L-1}(\cdots(f_1(\boldsymbol{x}))\cdots))$$

  - the key idea behind (deep) neural networks (NNs and DNNs)

Perceptron: XOR Function

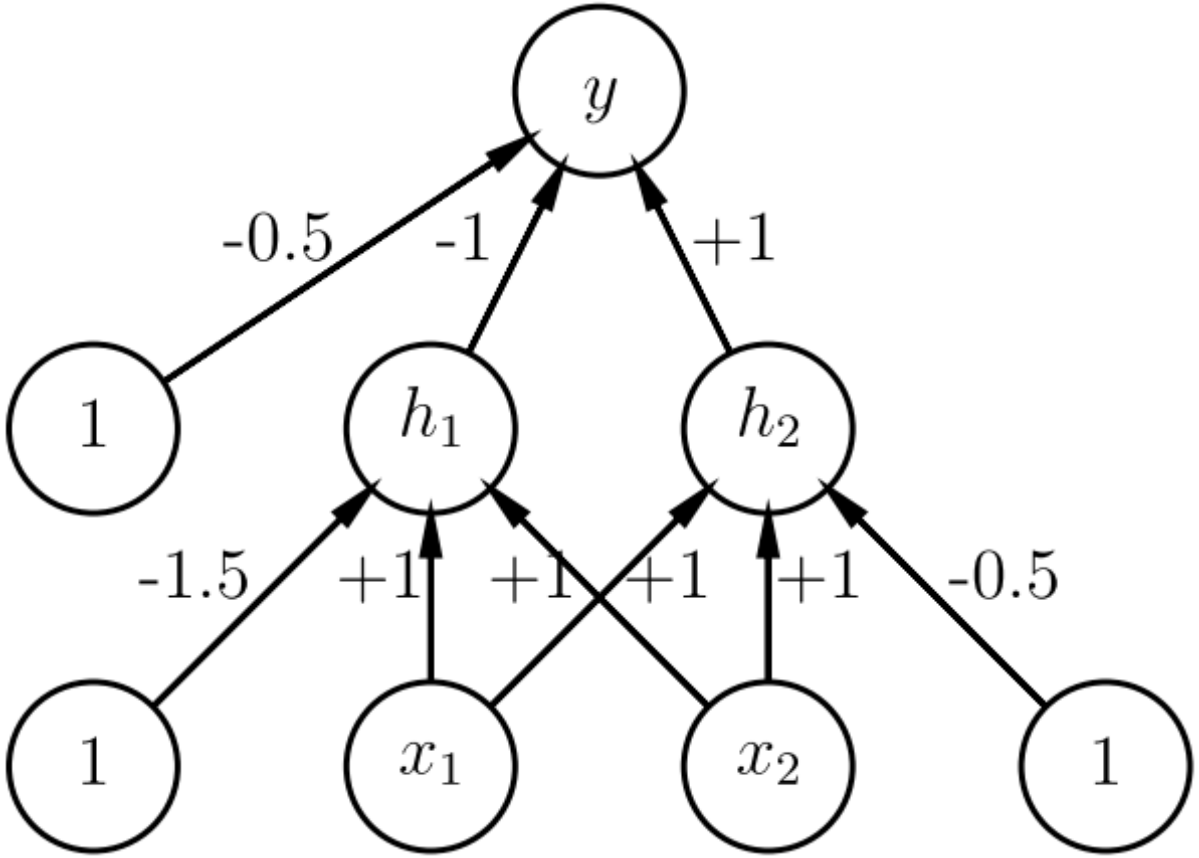
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



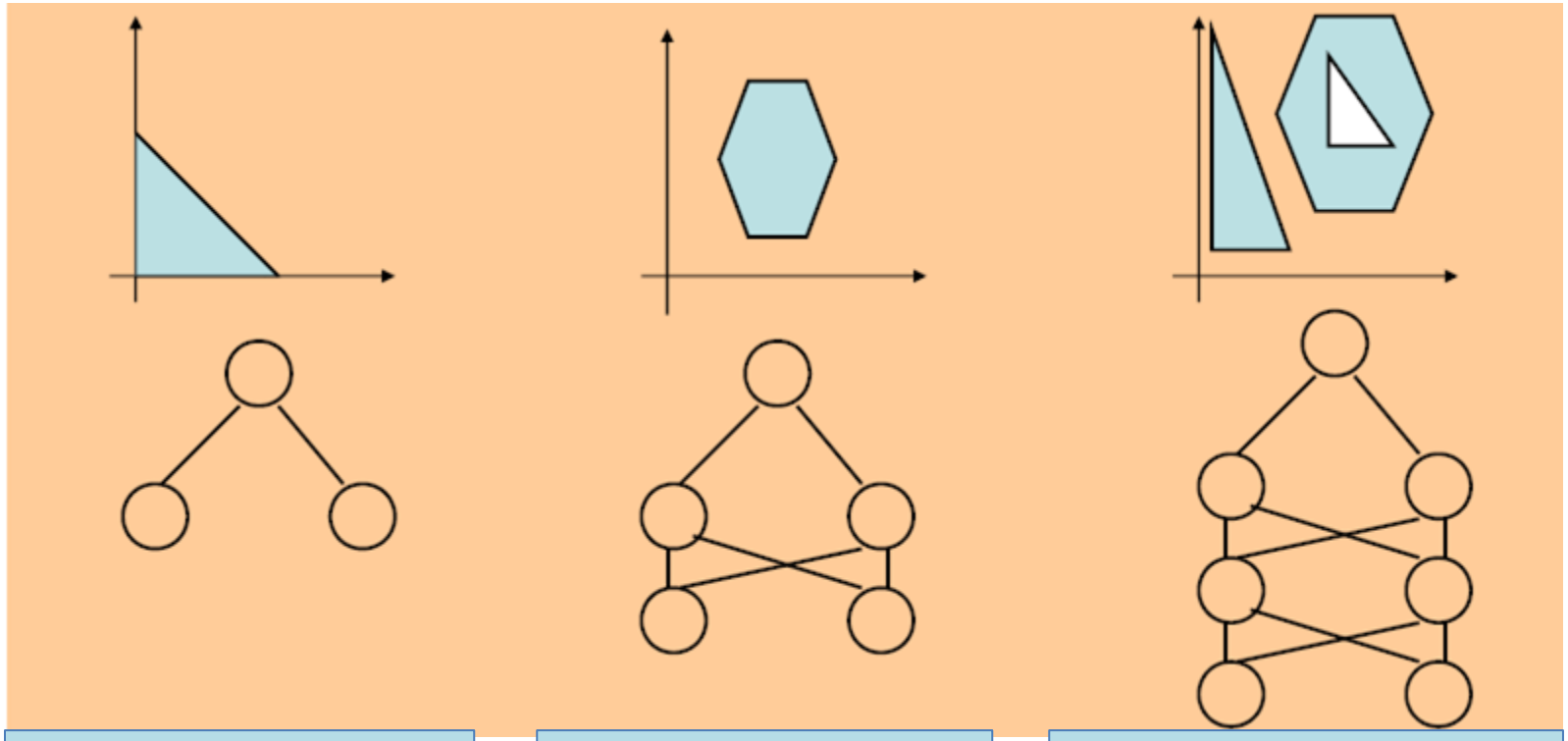
$$f(\boldsymbol{x};\boldsymbol{\theta}) = \mathbb{I}\left(\boldsymbol{w}^\mathsf{T}\boldsymbol{x} + b \geq 0\right) = H(\boldsymbol{w}^\mathsf{T}\boldsymbol{x} + b)$$

Two-Layer Perceptron: XOR Function

$$y = f(x_1, x_2) = \overline{(x_1 \wedge x_2)} \wedge (x_1 \vee x_2)$$



Why do we need hidden layers?



One Layer creates Linear boundaries	Two Layers combine Linear boundaries	Three or more Layers are able to create complex shapes

Activation Functions, Differentiable MLPs

$$\boldsymbol{z}_l = f_l(\boldsymbol{z}_{l-1}) = \varphi_l\left(\boldsymbol{b}_l + \mathbf{W}_l \boldsymbol{z}_{l-1}\right)$$

differentiable **activation function** $\varphi : \mathbb{R} \to \mathbb{R}$

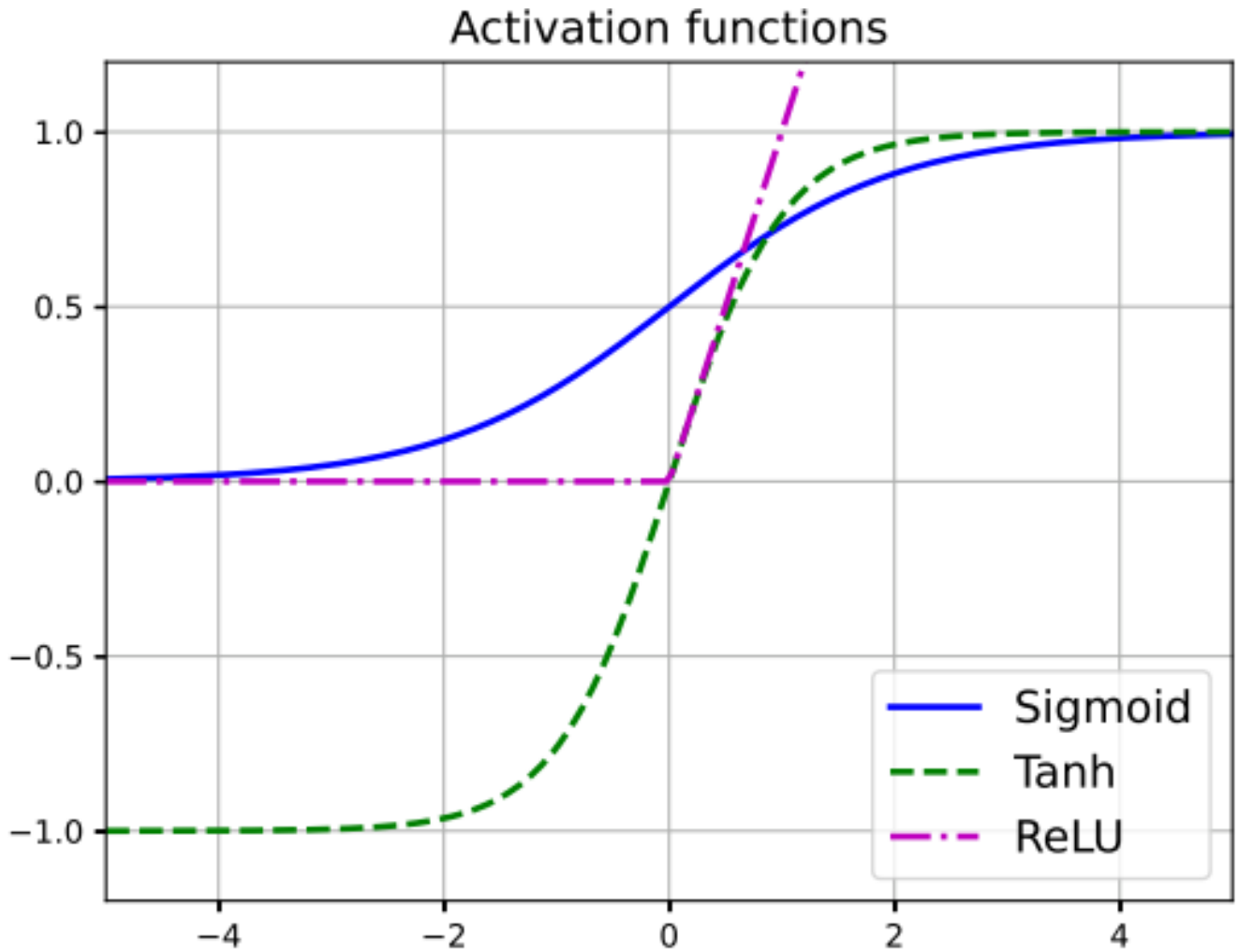
- the entire function is differentiable
- the composition of such functions is differentiable
- simply apply the chain rule repeatedly

- **Activation Functions**
  - typically, non-linear (otherwise, no gain)
  - sigmoid $\sigma(\alpha)$
  - tanh $f(x)$
  - ReLU (common choice)

$$\sigma(a) = \frac{1}{1+e^{-a}} \qquad f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$

$$\mathrm{ReLU}(a) = \max(a, 0) = a\mathbb{I}\left(a > 0\right)$$

Examples of Activation Functions



Softmax Activation Function

Activation Function:

$$\begin{bmatrix} 1.2 \\ 0.9 \\ 0.4 \end{bmatrix} \rightarrow \boxed{\text{Softmax}} \rightarrow \begin{bmatrix} 0.46 \\ 0.34 \\ 0.20 \end{bmatrix}$$

$$y_l(k) = \frac{e^{z_l(k)}}{\sum_{i=1}^{d_l} e^{z_l(i)}}$$

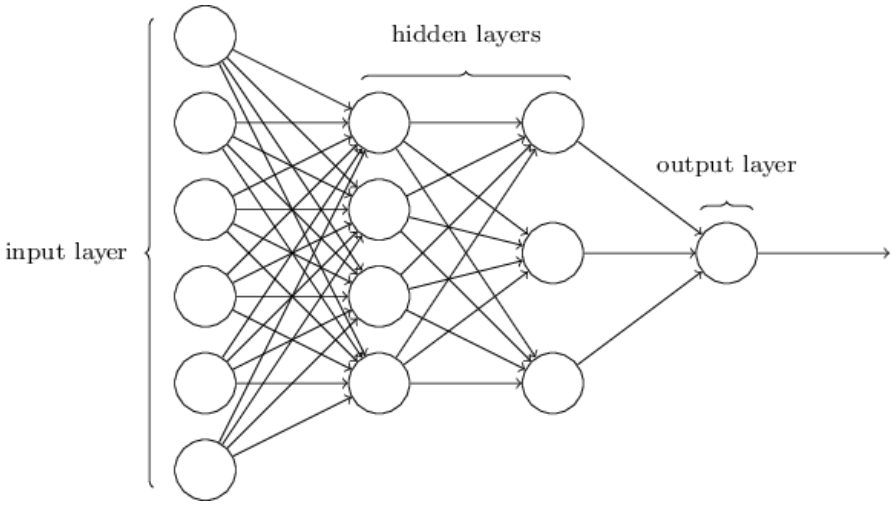
- d_l number of neurons

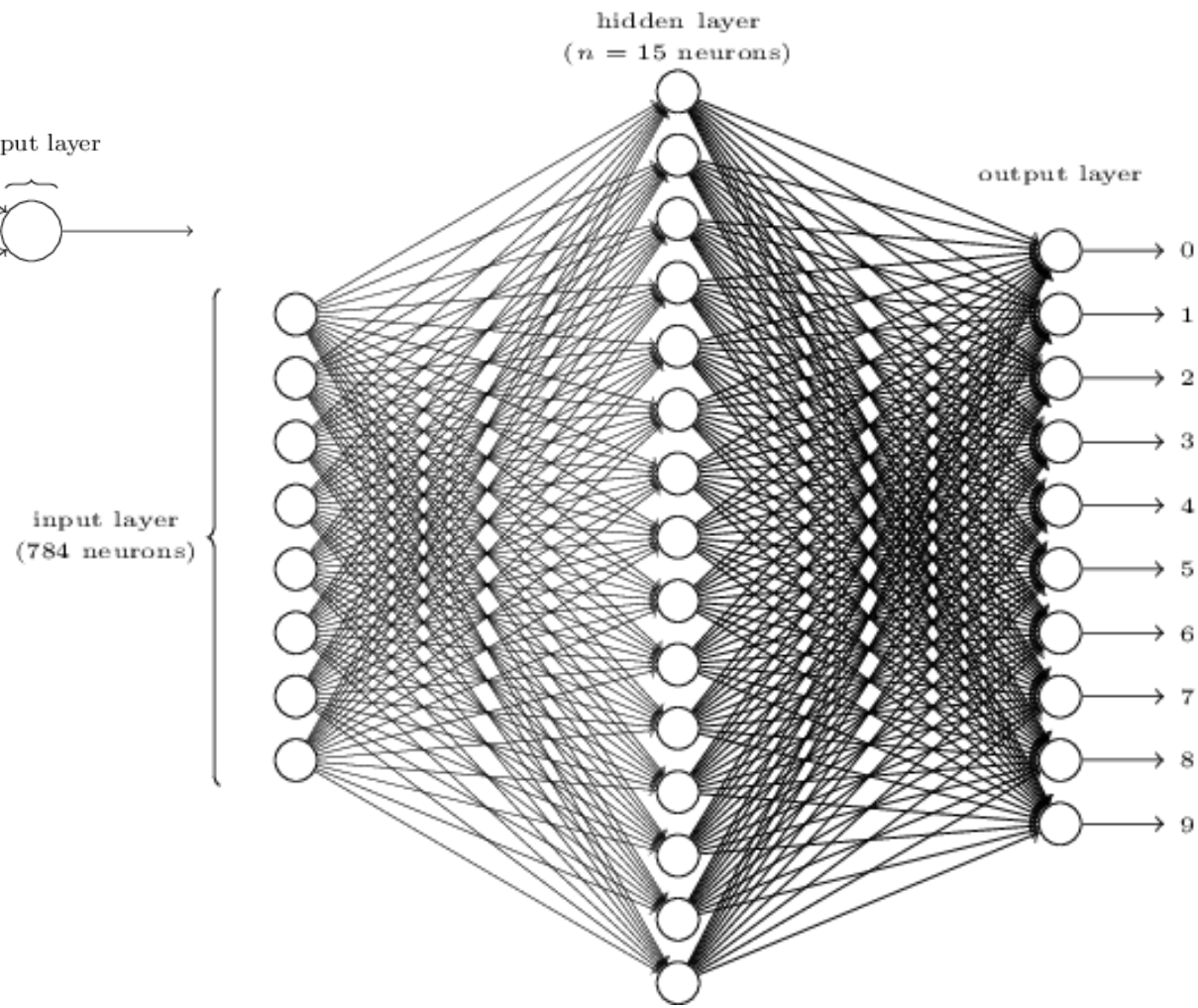
Usually used at the neurons of the output layer

Converts a vector of real numbers to a vector of posterior probabilities

- Output values are between 0 and 1 and sum to unit
  - The output of a neuron depends the outputs of all other neurons in its layer
- It is a measure of uncertainty of the output
  - Strong prediction ➔ single entry in the output vector close to 1
  - Weak prediction ➔ multiple entries almost equally likely

Multi-Layered Perceptrons



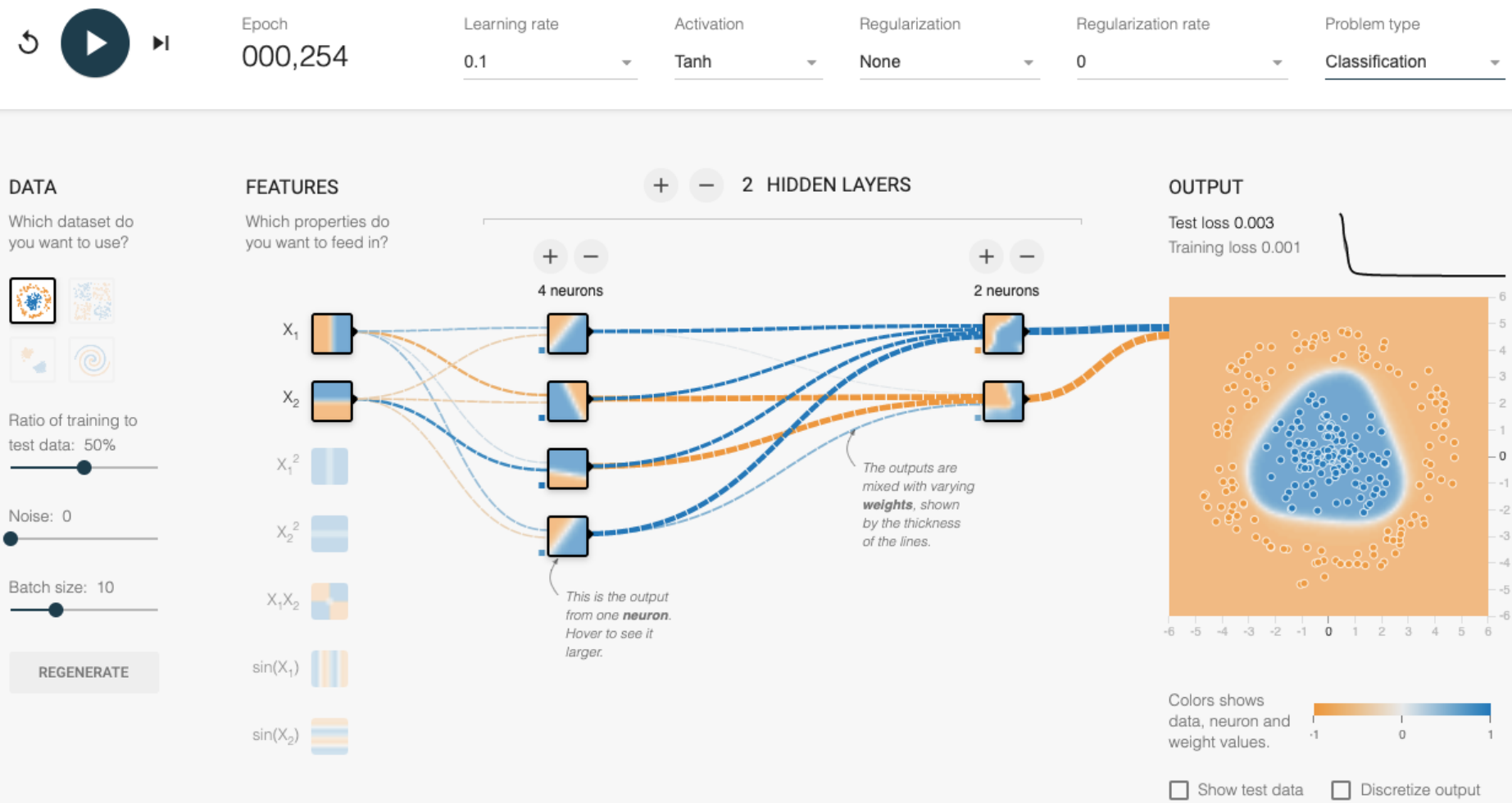


Learning Process of Neural Networks

- The NN structure is predefined (inference model)

- **Learning process**
  - The NN learns a specific task by adjusting its parameters (weights) from the correctly labeled training examples (supervised learning)
- **Adjusting the parameters**
  - randomly initialize the weights
  - obtain a predicted output based on this network
  - if the output matches the real label, do not change the weights
  - if the output is larger than the real label,
    adjust the weights that contribute to large output values
  - if the output is smaller than the real label,
    adjust the weights that contribute to small output values
  - repeat the previous steps, until the error converges to a minimum

http://playground.tensorflow.org



Backpropagation

Gradient Descent on the Cost Function

- **The key idea**
  - adjust a model's weights in response to the error it produces, until you can't reduce the error any more

- **"How do the errors vary as the weights are adjusted?"**
  - calculate how a change in *weight* affects a change in *Error*
  - first calculate how a change in *activation* affects a change in *Error*,
  - and then how a change in *weight* affects a change in *activation*

$$\frac{dError}{dweight} = \frac{dError}{dactivation} * \frac{dactivation}{dweight}$$

Gradient Descent on the Cost Function

- **Use a smooth cost function**
  - helps figure out how to make small changes in weights and biases to get an improvement in the cost

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

- **Perform stochastic gradient descent**
  - to guarantee reduction of cost

Xi ← training inputs as a mini-batch (small set of *m* training inputs)

A training epoch

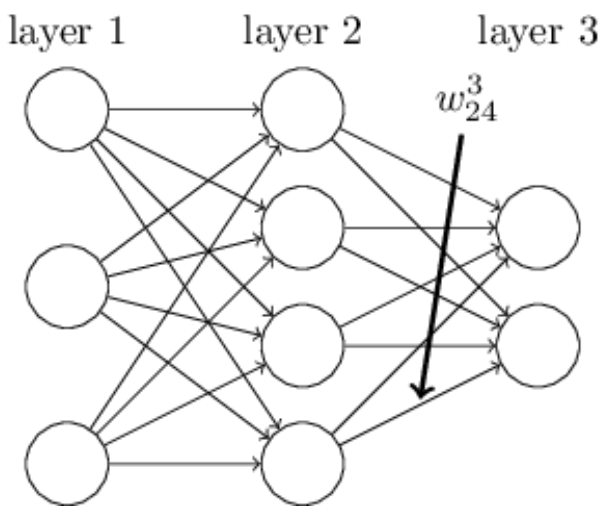
Train over a *randomly chosen* mini-batch, then on another, until you exhaust all mini-batches [repeat]

$$w_k \rightarrow w'_k = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

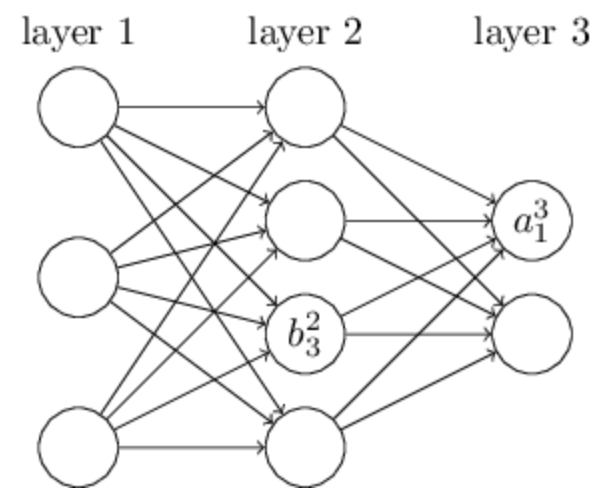
$$b_l \rightarrow b'_l = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l},$$

The Backpropagation Algorithm

- But…how to compute the gradient of the cost function?
- Use the backpropagation algorithm!



w^l_{jk} is the weight from the kth neuron in the $(l-1)^{\text{th}}$ layer to the jth neuron in the lth layer



$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right)$$

$$a^l = \sigma(w^l a^{l-1} + b^l)$$

$$a^l = \sigma(z^l)$$

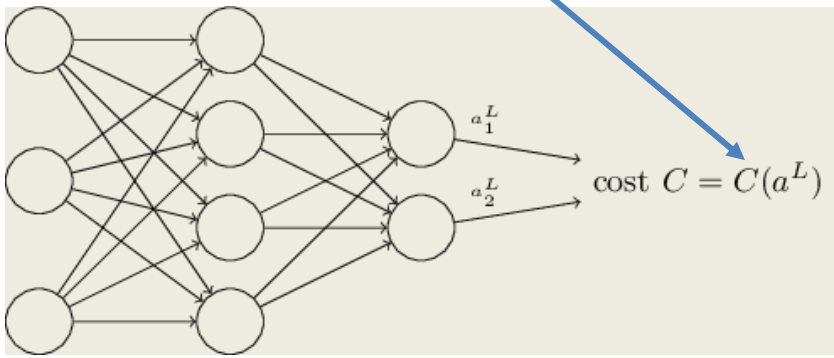
The Backpropagation Algorithm

$$C(w,b) \equiv \frac{1}{2n}\sum_x \|y(x)-a\|^2 \qquad C = \frac{1}{2n}\sum_x \|y(x)-a^L(x)\|^2$$

L: number of layers; *n* total number of training examples *x*; *y(x)* desired output; $a^L(x)$ vector of activations that are output from the network, when *x* is input

It holds: $C = \frac{1}{n}\sum_x C_x$ **required** for averaging over training examples

We also need *C* to be a function of the output activations, which we have



$$C = \frac{1}{2}\|y-a^L\|^2 = \frac{1}{2}\sum_j (y_j - a_j^L)^2$$

The Backpropagation Algorithm

the error δ^l_j of neuron j in layer l $\quad \delta^l_j \equiv \frac{\partial C}{\partial z^l_j}$

the error in the output layer $\quad \delta^L_j = \frac{\partial C}{\partial a^L_j}\sigma'(z^L_j) \qquad \delta^L = \nabla_a C \odot \sigma'(z^L)$

$s \odot t$ denotes the *elementwise* product of the two vectors

$$(s \odot t)_j = s_j t_j \qquad \begin{bmatrix} 1 \\ 2 \end{bmatrix} \odot \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 * 3 \\ 2 * 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix}$$

For the quadratic cost function, we have

$$\boxed{C = \frac{1}{2}\sum_j (y_j - a^L_j)^2}$$

So we can easily compute: $\boxed{\partial C / \partial a^L_j = (a^L_j - y_j)}$

The Backpropagation Algorithm

An equation for the error δ^l in terms of the error in the next layer

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

moves the error backward,

first assessing it on *l* layer's output, then to *l*'s weighted input via σ

the rate of change of the cost with respect to any bias in the network

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

the rate of change of the cost w.r.t. any weight in the network

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$

The Backpropagation Algorithm

1. **Input** x**:** Set the corresponding activation a^1 for the input layer.

2. **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$.

3. **Output error** δ^L**:** Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$.

4. **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.

5. **Output:** The gradient of the cost function is given by $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ and $\frac{\partial C}{\partial b^l_j} = \delta^l_j$.

The Backpropagation Algorithm

In other words:

- We compute the error vectors δ^l backward, starting from the final layer.

- The backward movement is a consequence of the fact that the cost is a function of outputs from the network.

In practice, we usually combine backpropagation with stochastic gradient descent, in which we compute the gradient for many training examples.

Gradient descent given a mini-batch of m training examples

1. **Input a set of training examples**

2. **For each training example** x**:** Set the corresponding input activation $a^{x,1}$, and perform the following steps:

   - **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^{x,l} = w^l a^{x,l-1} + b^l$ and $a^{x,l} = \sigma(z^{x,l})$.

   - **Output error** $\delta^{x,L}$**:** Compute the vector $\delta^{x,L} = \nabla_a C_x \odot \sigma'(z^{x,L})$.

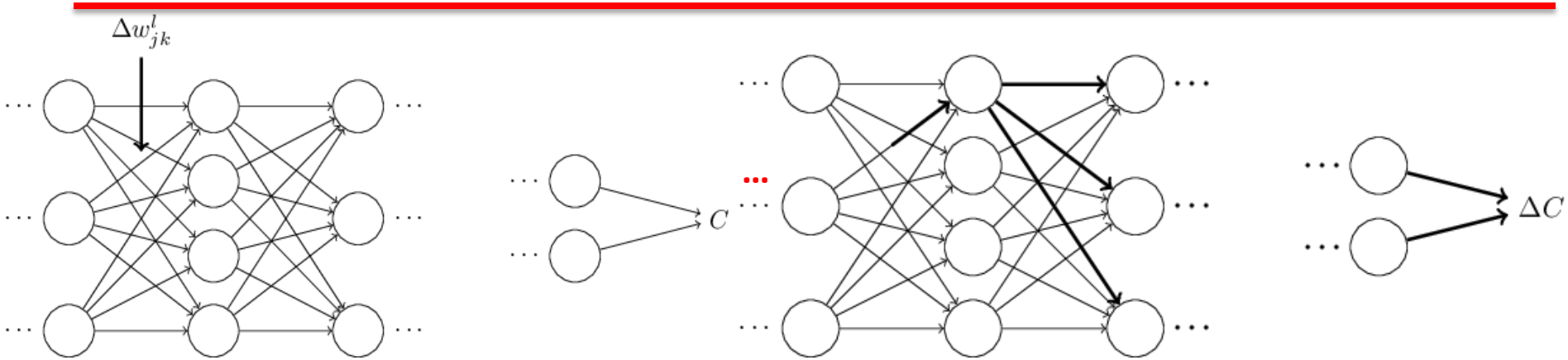
   - **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^{x,l} = ((w^{l+1})^T \delta^{x,l+1}) \odot \sigma'(z^{x,l})$.

3. **Gradient descent:** For each $l = L, L-1, \ldots, 2$ update the weights according to the rule $w^l \rightarrow w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$, and the biases according to the rule $b^l \rightarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$.

Backpropagation: More intuitions

- Every edge between two neurons in the network is associated with a rate factor, which is just the partial derivative of one neuron's activation with respect to the other neuron's activation.
- The rate factor for a path is just the product of the rate factors along the path.
- The total rate of change of *C* with respect to a weight in the network is just the sum of the rate factors of all paths from the initial weight to the final cost.
- The backpropagation algorithm is providing a way of computing the sum over the rate factor for all these paths.

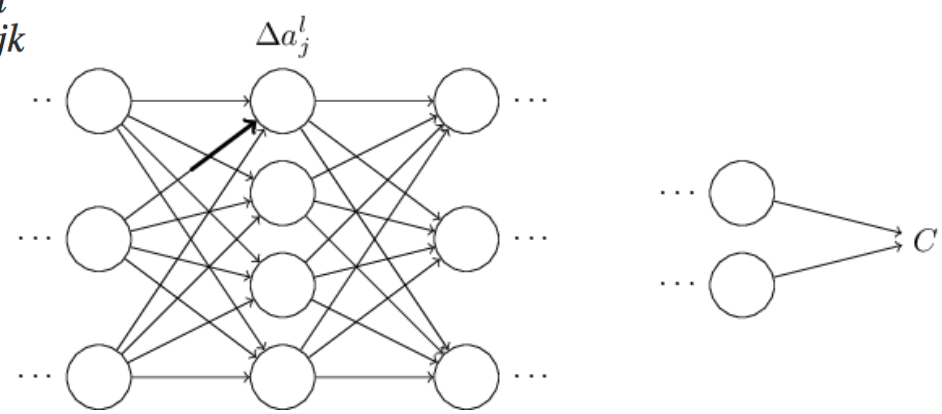
Backpropagation: More intuitions



Given the change, overall: $\Delta C \approx \dfrac{\partial C}{\partial w^l_{jk}} \Delta w^l_{jk}$

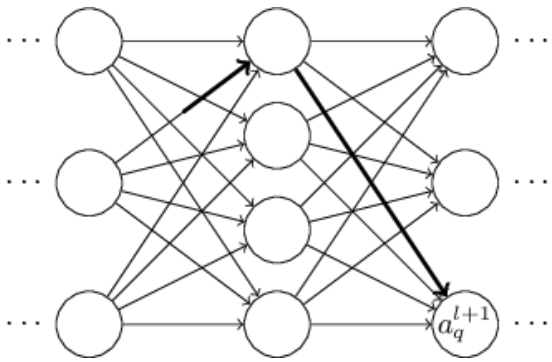
First, the change causes a small change in the activation of the *j*-th neuron in the *l*-th layer:

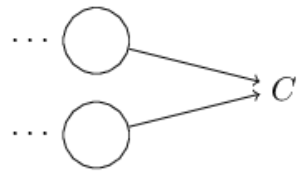
$$\Delta a^l_j \approx \frac{\partial a^l_j}{\partial w^l_{jk}} \Delta w^l_{jk}$$



This change in turn, will cause changes in *all* the activations in the next layer.

Backpropagation: More intuitions





$$\Delta a_q^{l+1} \approx \frac{\partial a_q^{l+1}}{\partial a_j^l} \Delta a_j^l$$

$$\Delta a_q^{l+1} \approx \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l$$

Imagining a path all the way to C :

$$\Delta C \approx \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l$$

Since the change propagates across all possible paths to affect C :

$$\Delta C \approx \sum_{mnp\ldots q} \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l$$

Backpropagation: More intuitions

$$\Delta C \approx \sum_{mnp\ldots q} \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l \qquad \Delta C \approx \frac{\partial C}{\partial w_{jk}^l} \Delta w_{jk}^l$$

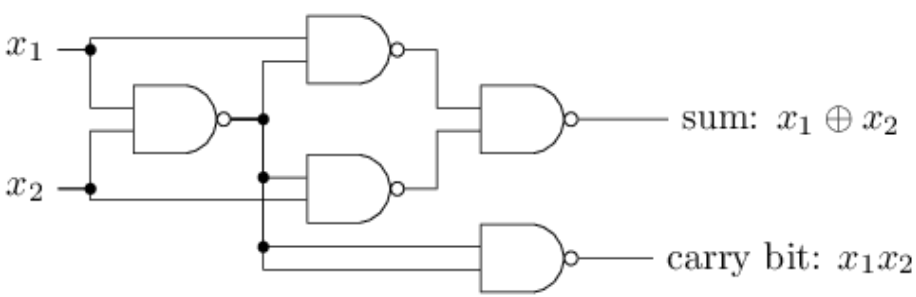
$$\frac{\partial C}{\partial w_{jk}^l} = \sum_{mnp\ldots q} \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l}$$

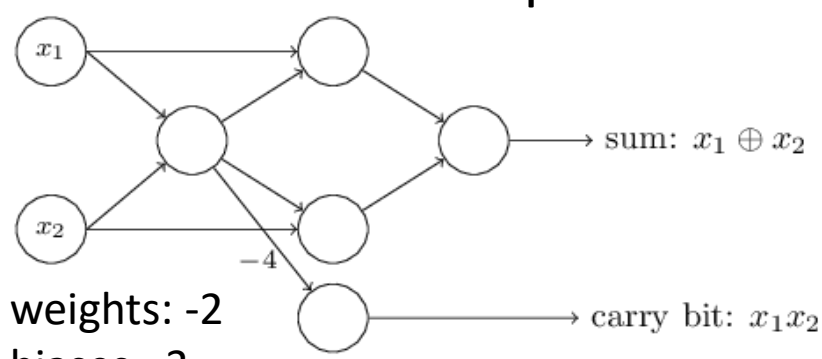
- We are computing the rate of change of *C* with respect to a weight in the network.
- This is just the sum of the rate factors of all paths from that weight to the final cost.
- The backpropagation algorithm computes the sum over the rate factor for all these paths.

Deep Considerations …

The Universal Approximation Theorem

- **"A feed-forward network with a single hidden layer containing a finite number of neurons, can approximate continuous functions on compact subsets of R^n, under mild assumptions on the activation function."**
  - shown, e.g., for sigmoid functions (Cybenko, 1989)
- **Intuition:** a network of perceptrons can simulate a circuit containing NAND gates. Since any boolean function can be implemented using NAND gates, it follows that perceptrons are universal computers!





weights: -2
biases= 3

The Universal Approximation Theorem

- **Implication**
  - Simple networks can represent essentially any function.
- **Thus, in theory,**
  - all we need is networks of perceptrons…
  - Unfortunately, this does not tell us anything on how to actually do the learning / computations…
  - in a sense, we just confirm that a network of perceptrons is a new type of NAND gate!
- **Practice + new theory shows that deep is good…**

Why Deep NN?

- **First … what does "deep" mean?**
- **Well, deep means more than two (hidden+output) layers!**
- **So, strictly speaking, it is not "a new thing"!**
- **Why "two"? With depth 2 we have a universal approximator.**
- **Recent theoretical results show the power of "deep".**
- **Deep architectures can represent a function with exponentially fewer units!**

Deep Learning and Deep Neural Networks

- **Why now (i.e., after 2010)?**
  - more powerful CPUs + appearance of GPUs + big data
  - high-quality software (tensorflow, PyTorch, MXNet, …)
  - new theoretical results that suggest their power
- **A Spiral Effect**
  - Deep NNs do well →
  - more and more people now get interested →
  - people get to learn how to tweak Deep NNs →
  - Deep NNs get even better in more domains →
  - [repeat]

Deep Neural Networks in a Nutshell

input layer + number of hidden layers + output layer

- Each unit's input: a weighted sum of outputs of units from previous layers (weights are on links between layers)
- Units' outputs: nonlinear transformation of weighted inputs by activation functions (tanh, logistic, rectified linear unit, …)
- Compute error derivatives and backpropagate gradients from the output layer towards the input layer.
- Update weights from gradients to optimize loss function.
- Repeat until convergence.

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 25

Deep Learning

Neural Networks for Images

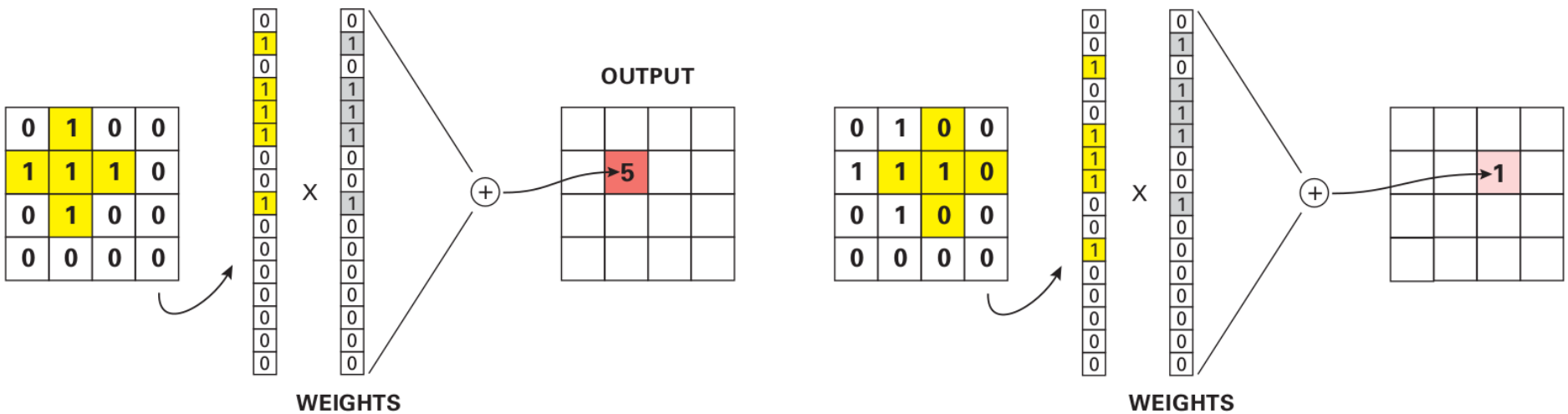
TUC ECE, Spring 2023

Today

- **Convolutional Neural Networks**
- **Trained Systems for Images**

Plain MLPs for Images?

- **Problems**
  - huge number of weights
  - spatial information is ignored
  - cannot account for translations

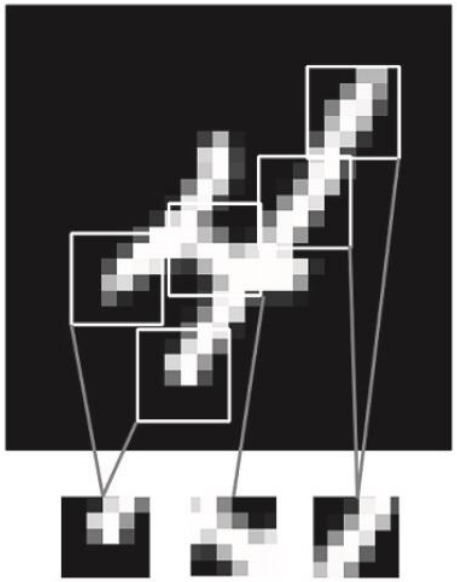


Convolutional Neural Networks

Convolutional Neural Networks

- **Solution**
  - convolution operations instead of matrix operations
  - divide the image into small overlapping patches
  - compare each patch against a template
  - templates (filters) are small (3x3, 4x4, …)
  - templates have a small weight matrix
  - translation invariance!

  - look for the relative locations of matches
  - good for character recognition



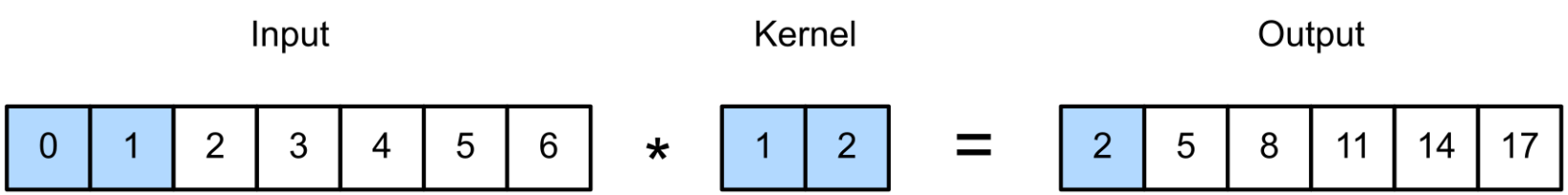
Convolutions in 1D

$$[f \circledast g](\boldsymbol{z}) = \int_{\mathbb{R}^D} f(\boldsymbol{u}) g(\boldsymbol{z} - \boldsymbol{u}) d\boldsymbol{u}$$

$$[\boldsymbol{w} \circledast \boldsymbol{x}](i) = \sum_{u=0}^{L-1} w_u x_{i+u}$$

-	-	1	2	3	4	-	-	
7	6	5	-	-	-	-	-	$z_0 = x_0 w_0 = 5$
-	7	6	5	-	-	-	-	$z_1 = x_0 w_1 + x_1 w_0 = 16$
-	-	7	6	5	-	-	-	$z_2 = x_0 w_2 + x_1 w_1 + x_2 w_0 = 34$
-	-	-	7	6	5	-	-	$z_3 = x_1 w_2 + x_2 w_1 + x_3 w_0 = 52$
-	-	-	-	7	6	5	-	$z_4 = x_2 w_2 + x_3 w_1 = 45$
-	-	-	-	-	7	6	5	$z_5 = x_3 w_2 = 28$

Discrete convolution of $\boldsymbol{x} = [1, 2, 3, 4]$ *with* $\boldsymbol{w} = [5, 6, 7]$

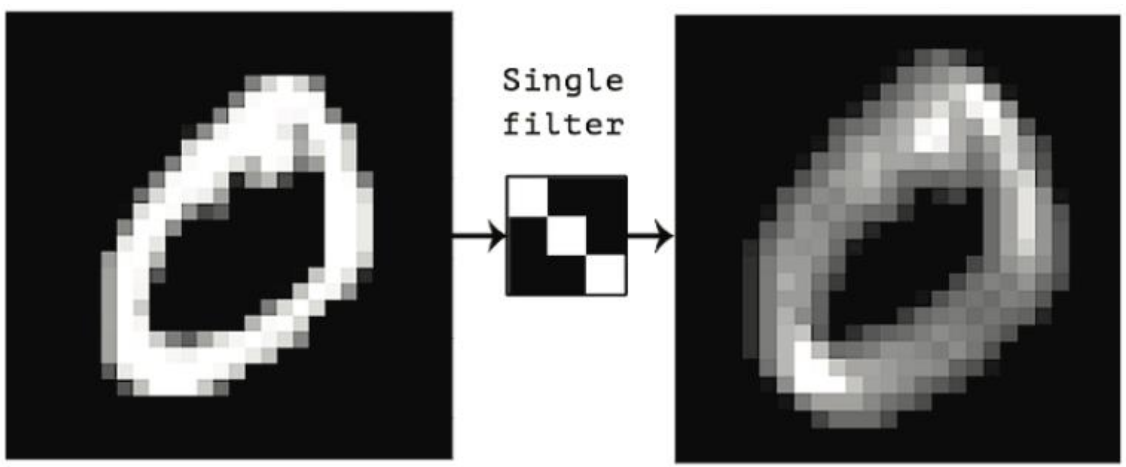


1d cross correlation

Convolutions in 2D

$$[\mathbf{W} \circledast \mathbf{X}](i,j) = \sum_{u=0}^{H-1} \sum_{v=0}^{W-1} w_{u,v} x_{i+u,j+v}$$

Input				Kernel			Output	
0	1	2	*	0	1	=	19	25
3	4	5		2	3		37	43
6	7	8						

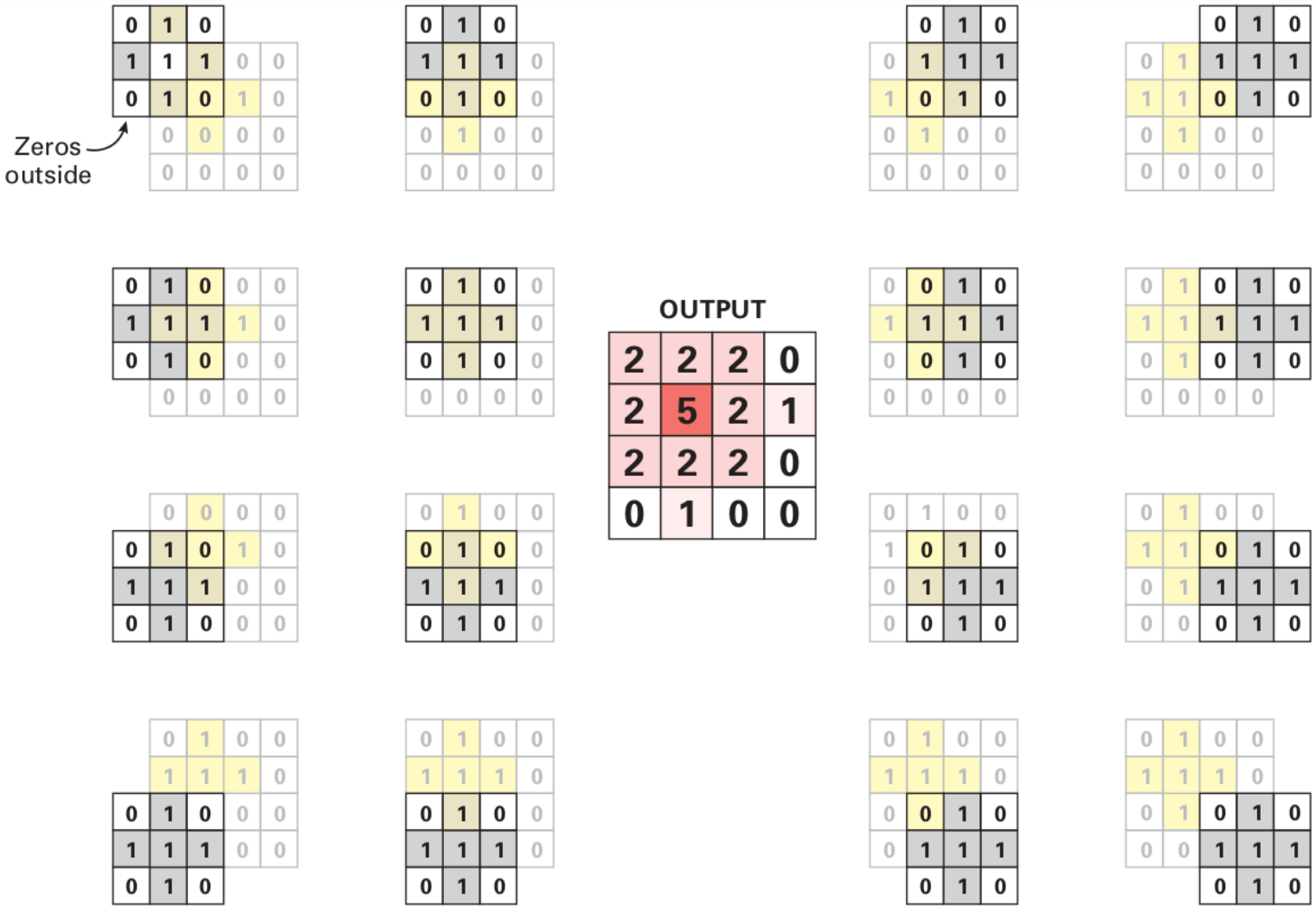


Convolution as Matrix Multiplication

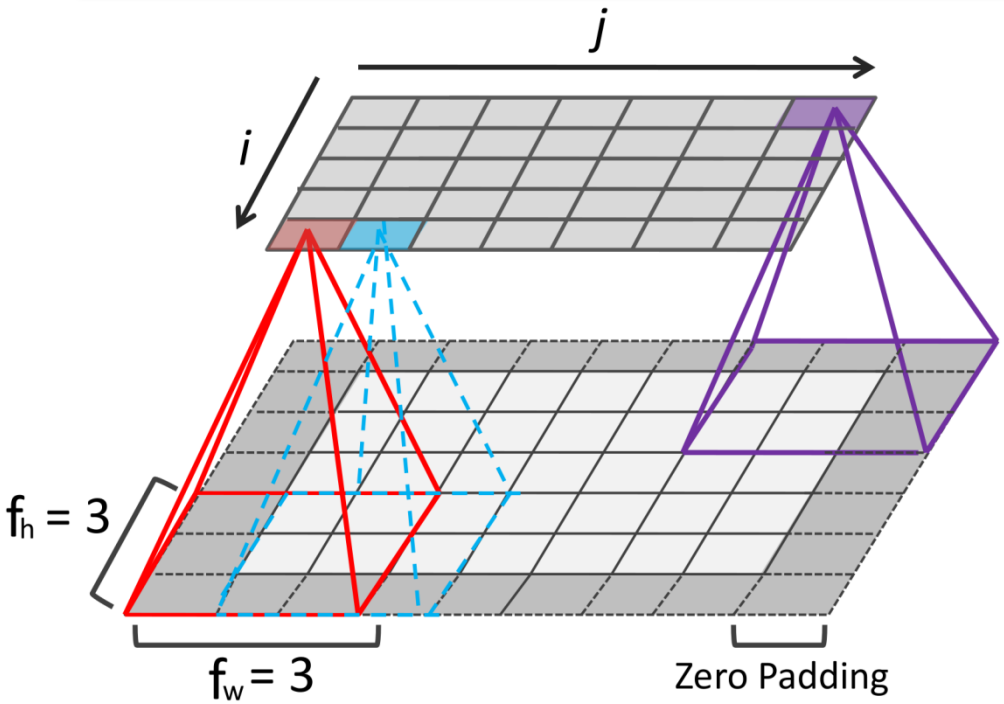
$$\boldsymbol{y} = \mathbf{C}\boldsymbol{x} = \left(\begin{array}{ccc|ccc|ccc} w_1 & w_2 & 0 & w_3 & w_4 & 0 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & 0 & w_3 & w_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & 0 & w_3 & w_4 & 0 \\ 0 & 0 & 0 & 0 & w_1 & w_2 & 0 & w_3 & w_4 \end{array}\right) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{pmatrix}$$

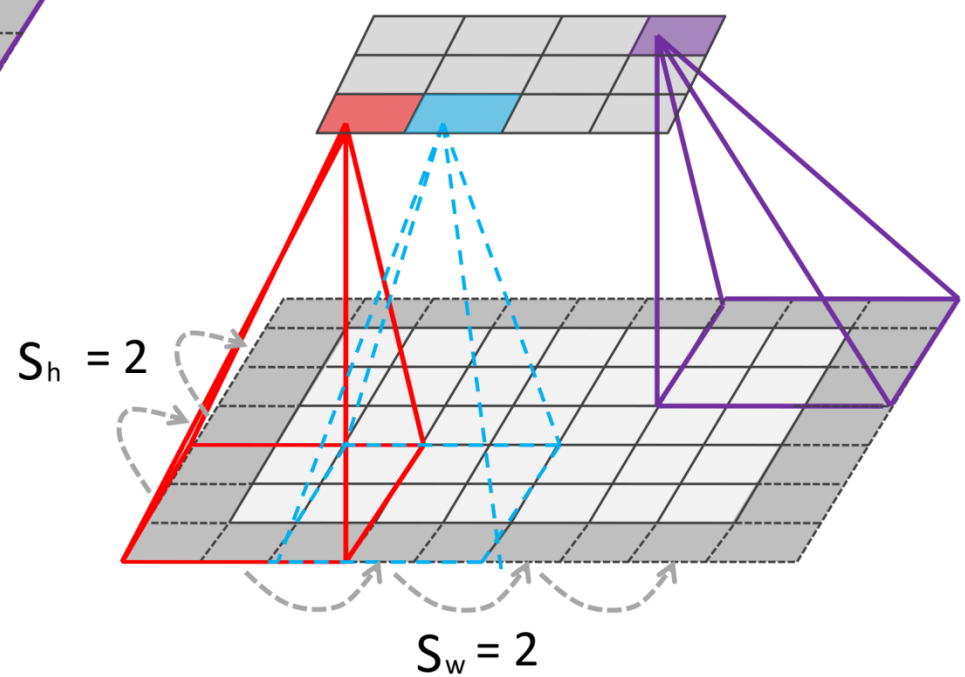
$$= \begin{pmatrix} w_1x_1 + w_2x_2 + w_3x_4 + w_4x_5 \\ w_1x_2 + w_2x_3 + w_3x_5 + w_4x_6 \\ w_1x_4 + w_2x_5 + w_3x_7 + w_4x_8 \\ w_1x_5 + w_2x_6 + w_3x_8 + w_4x_9 \end{pmatrix}$$

Zero-Padding (same Convolution)

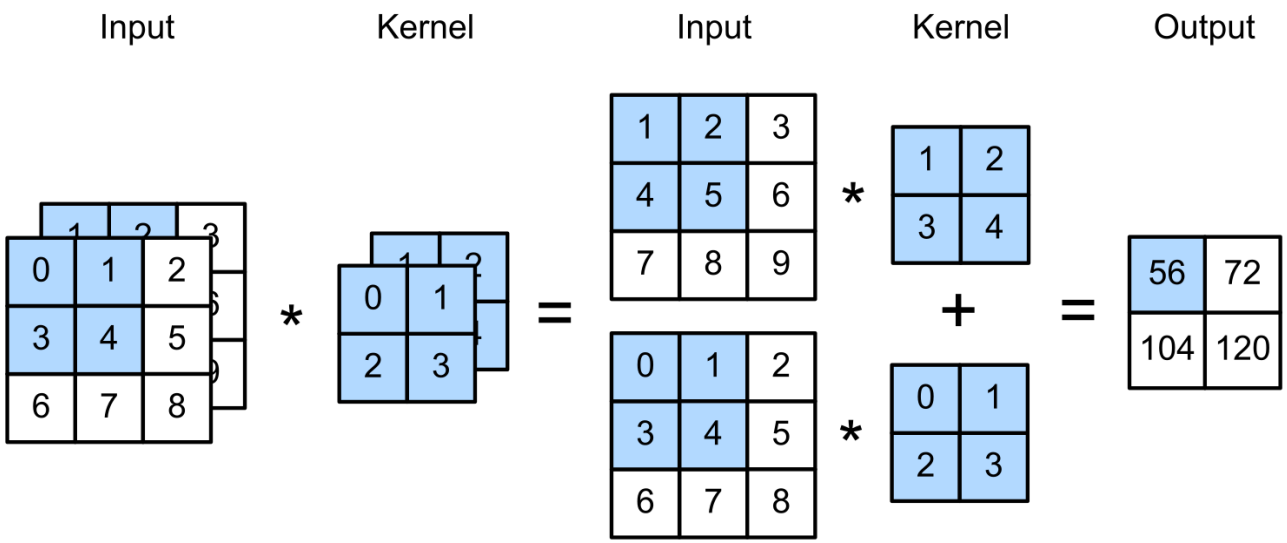


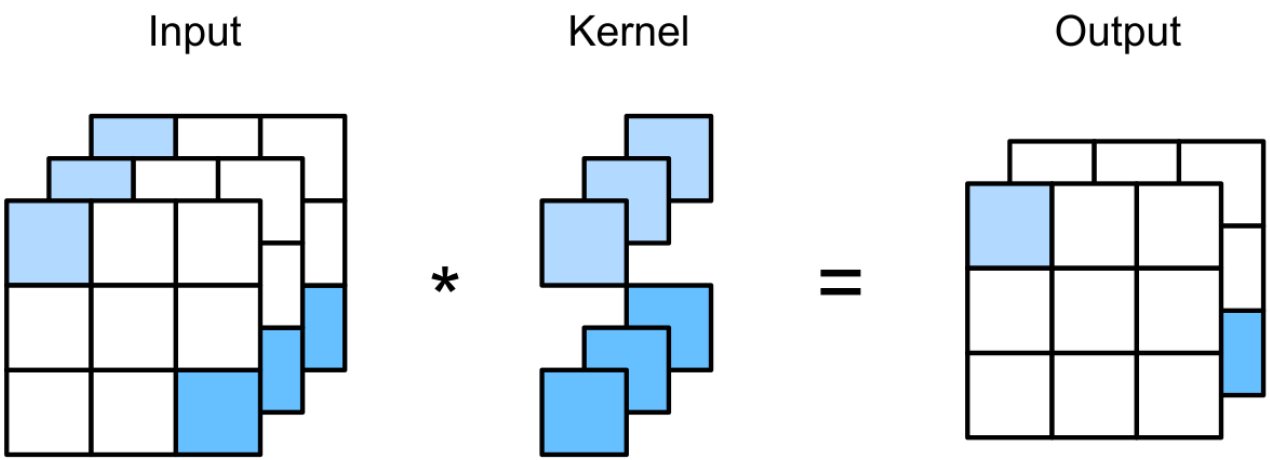
Strided Convolution



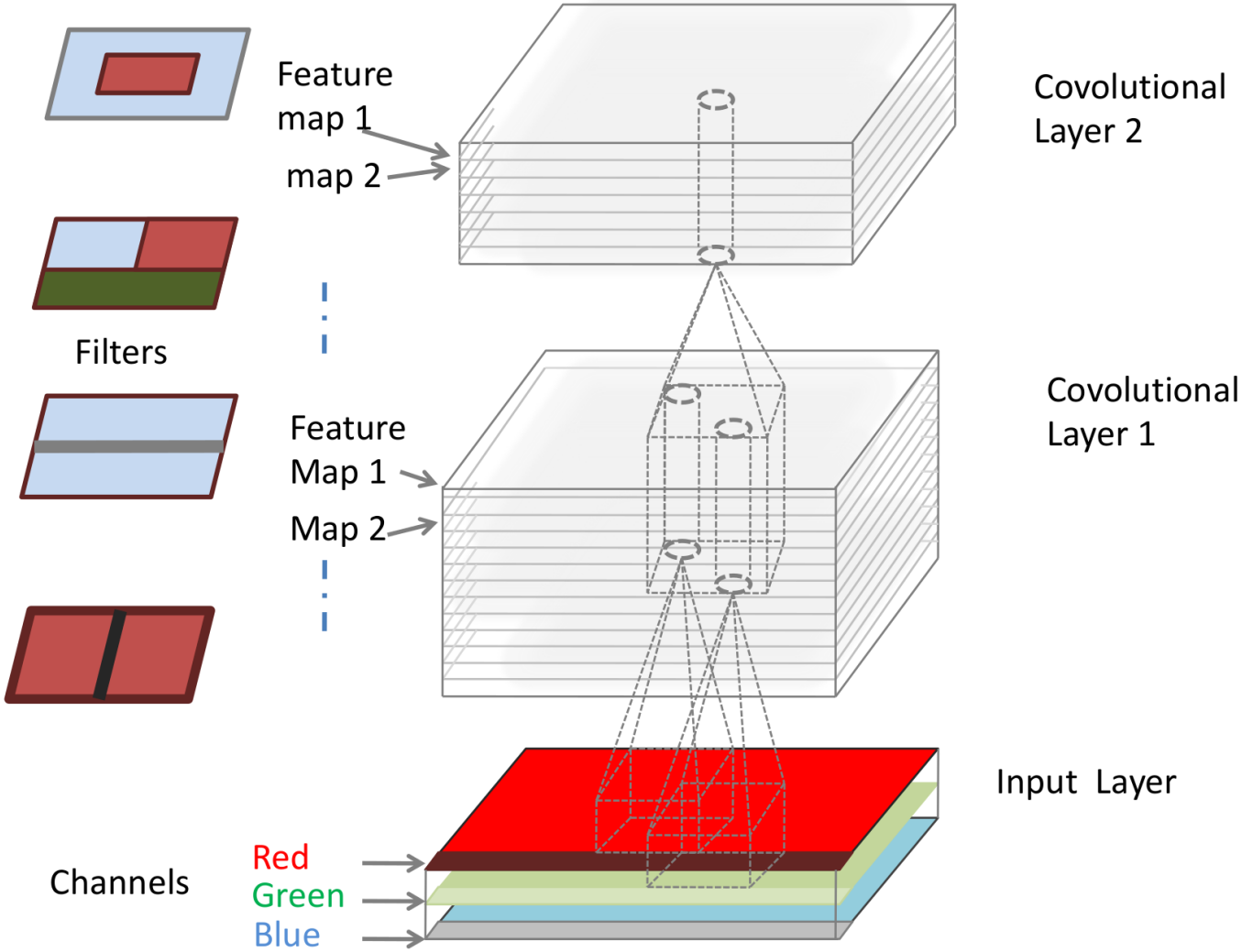


MIMO Convolutions

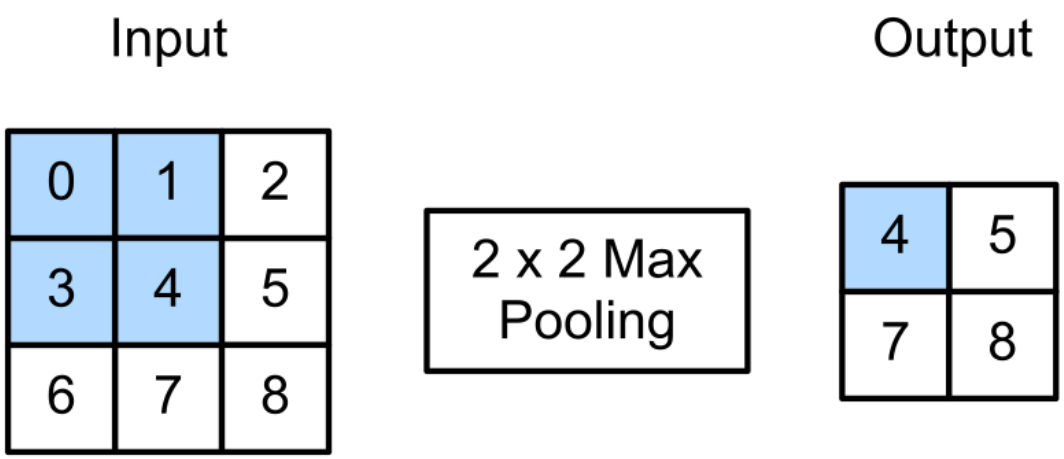




Convolutional Layers

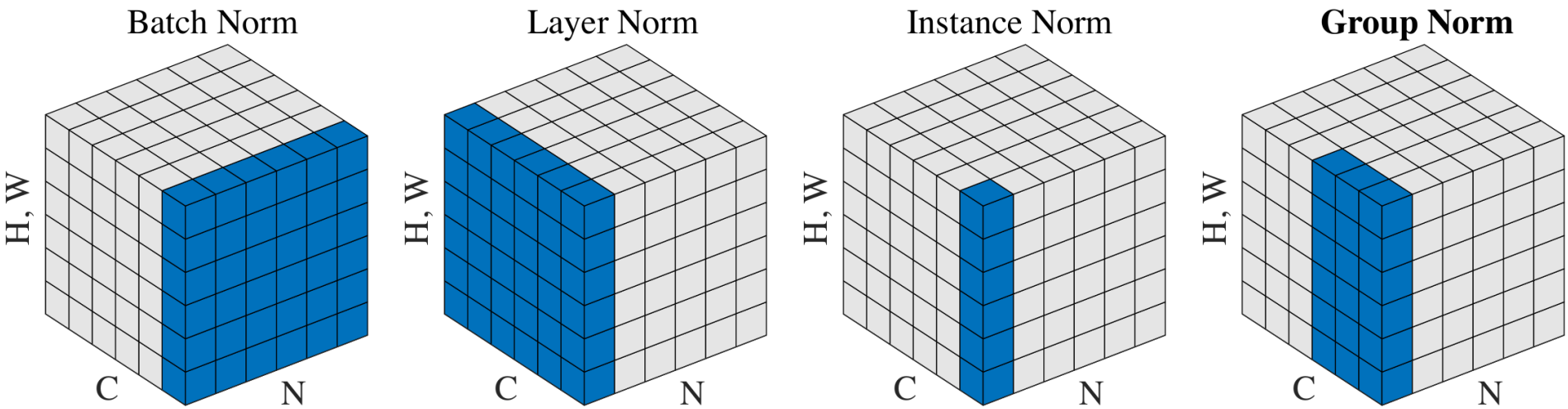


Pooling Layers

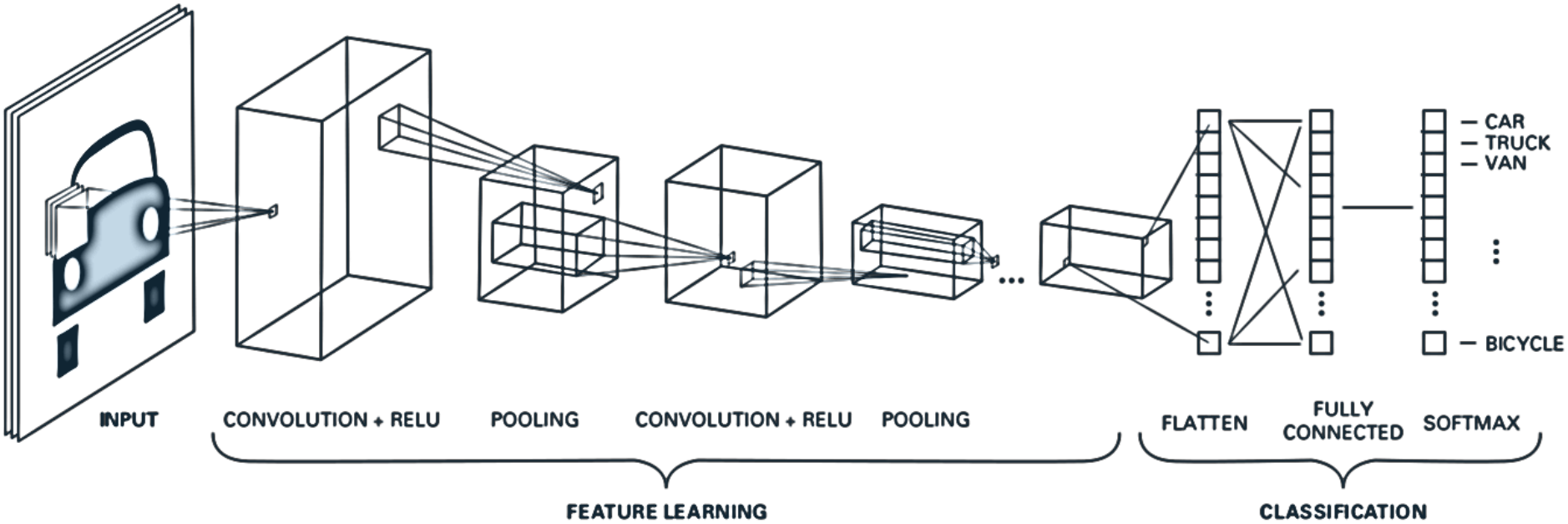


- **Max pooling**
- **Average pooling**
- **Global average pooling**

Normalization Layers

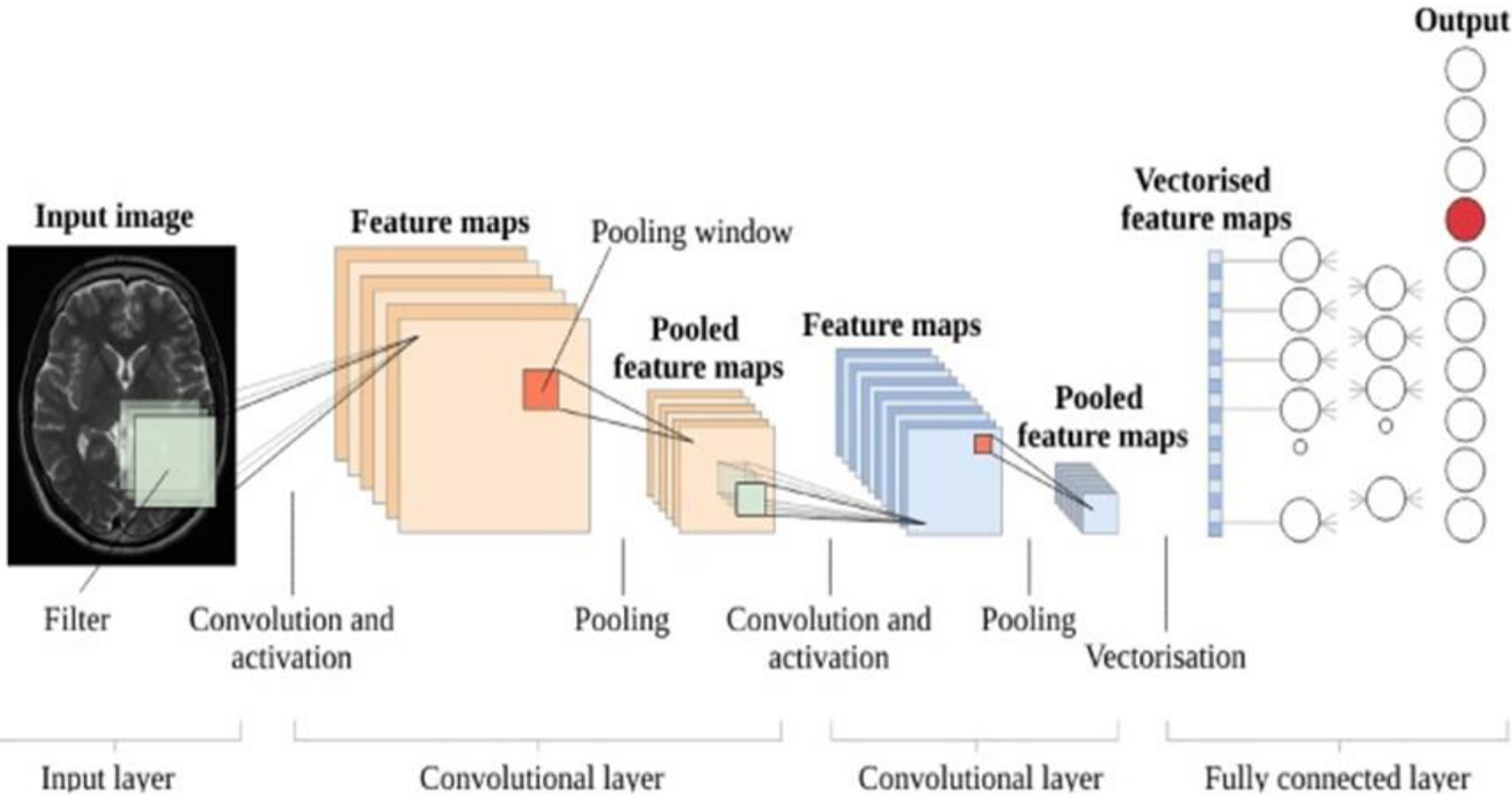


Convolutional NN for Classification

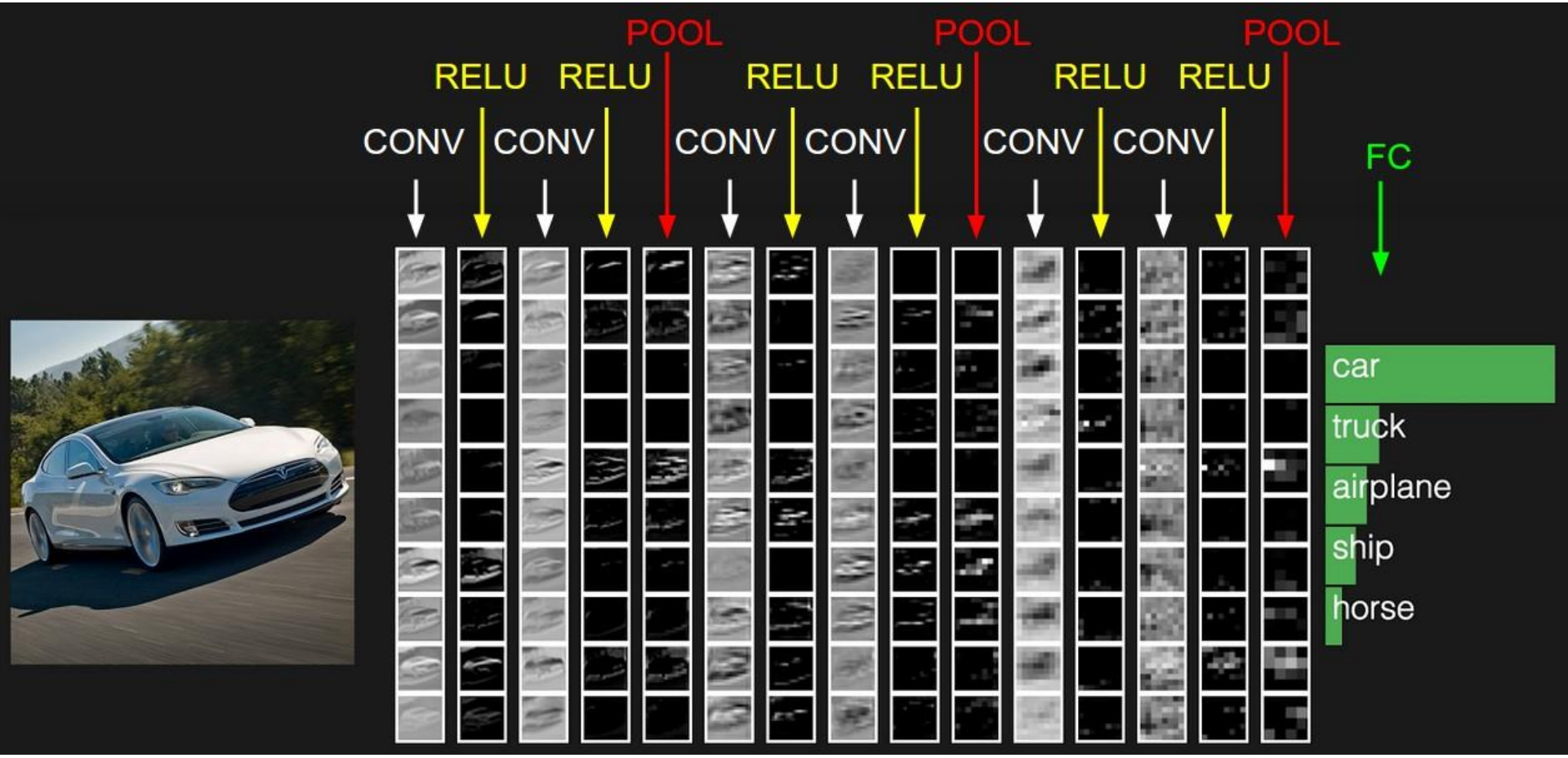


$$f : \mathbb{R}^{H \times W \times K} \rightarrow \{0, 1\}^C$$

Convolutional NN for Classification

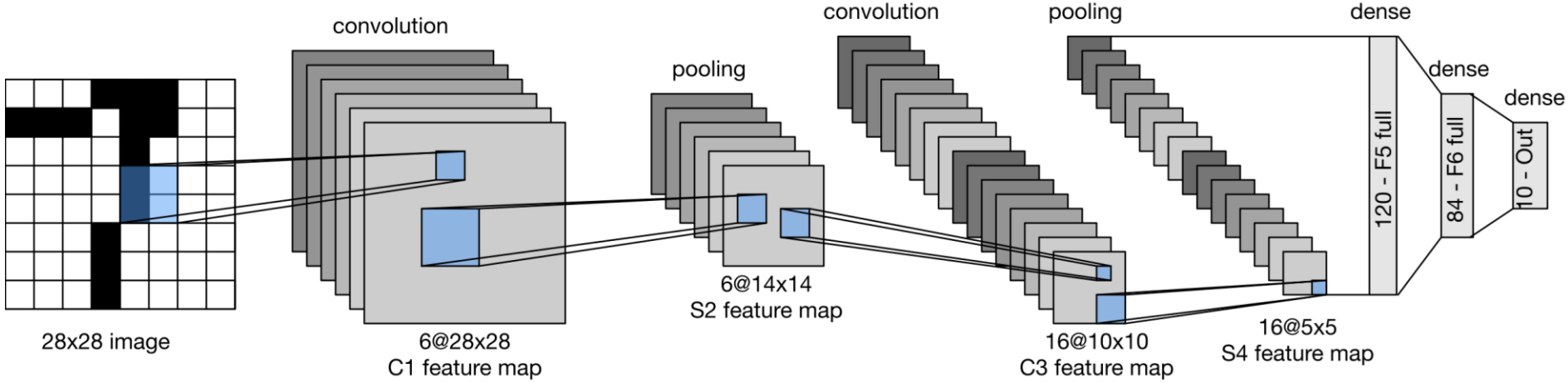


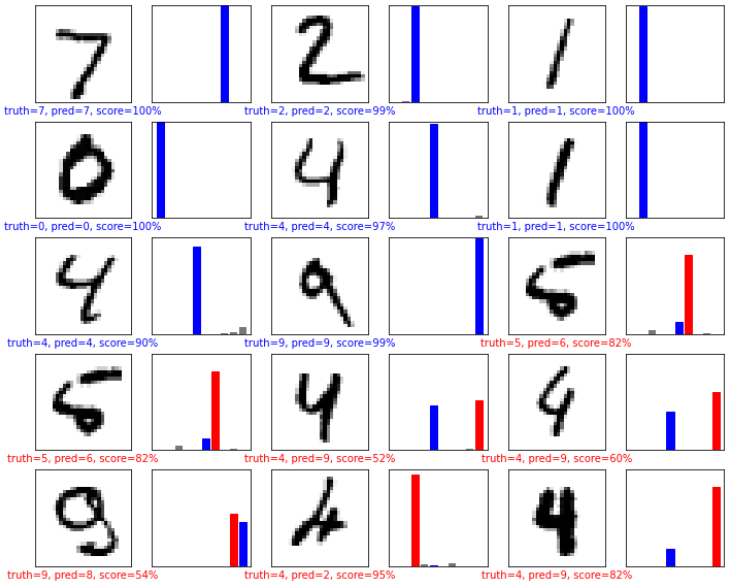
Example CNN Classification

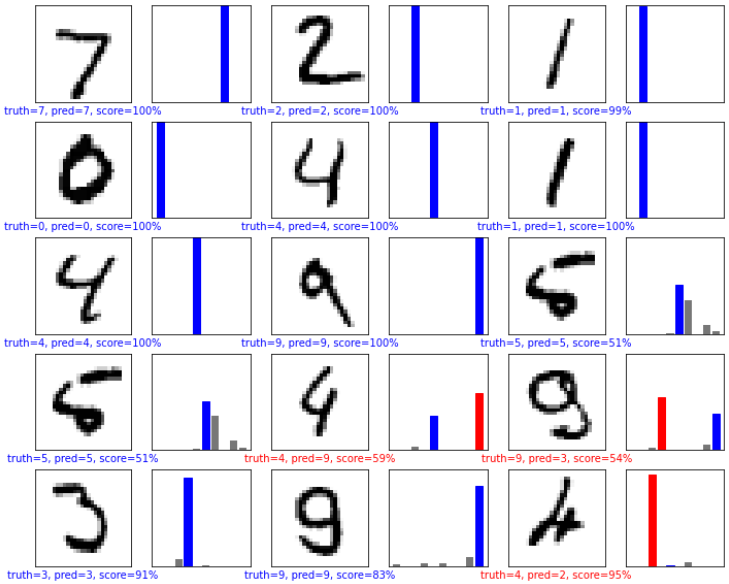


Trained Systems for Images

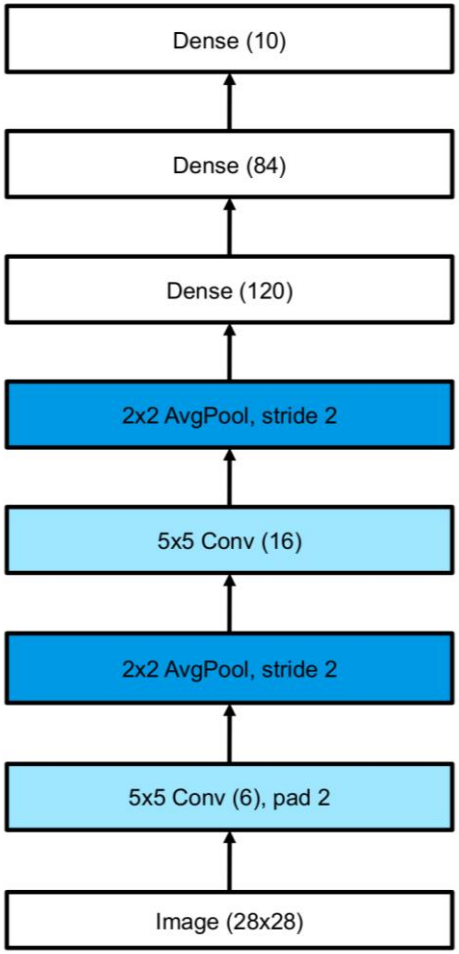
LeNet (Yann LeCun)

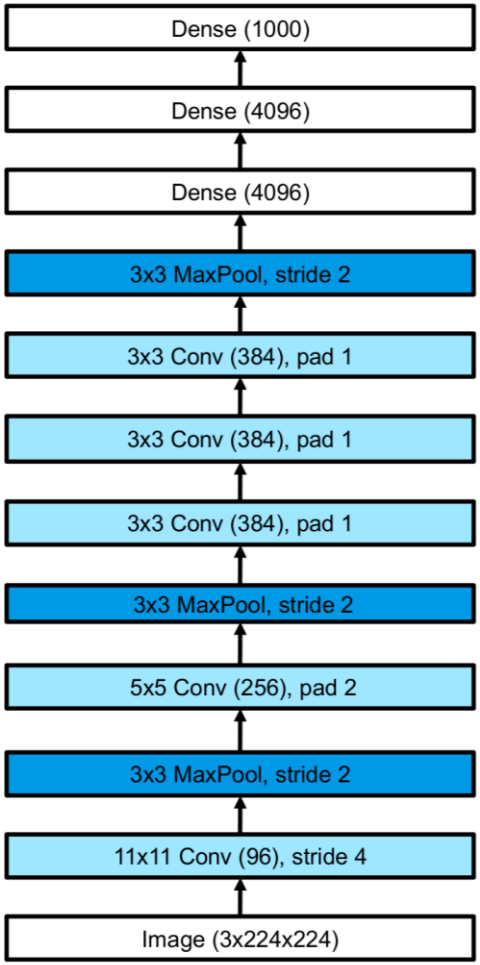




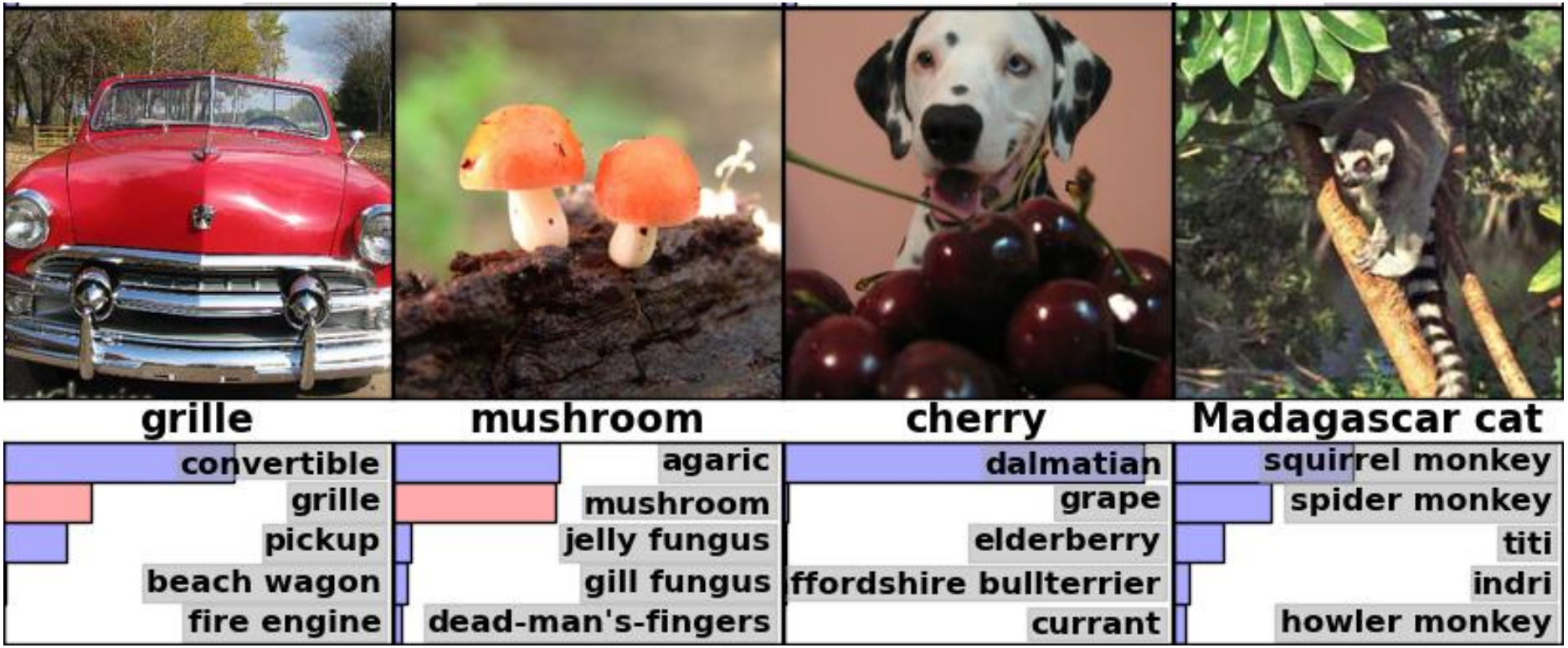


LeNet and AlexNet (Alex Krizhevsky)

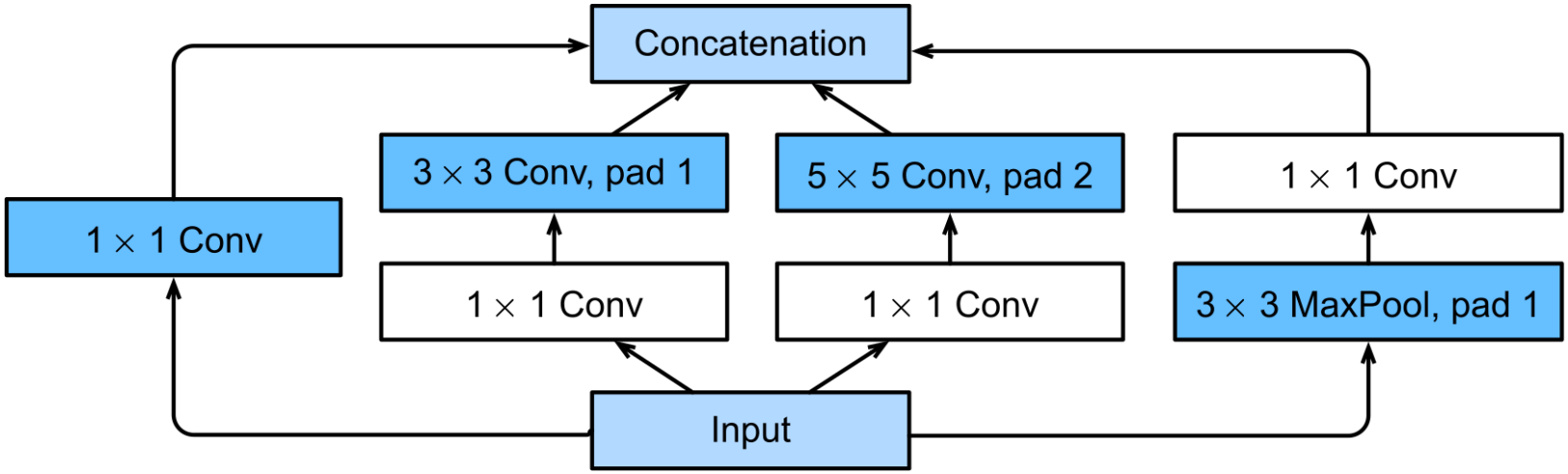


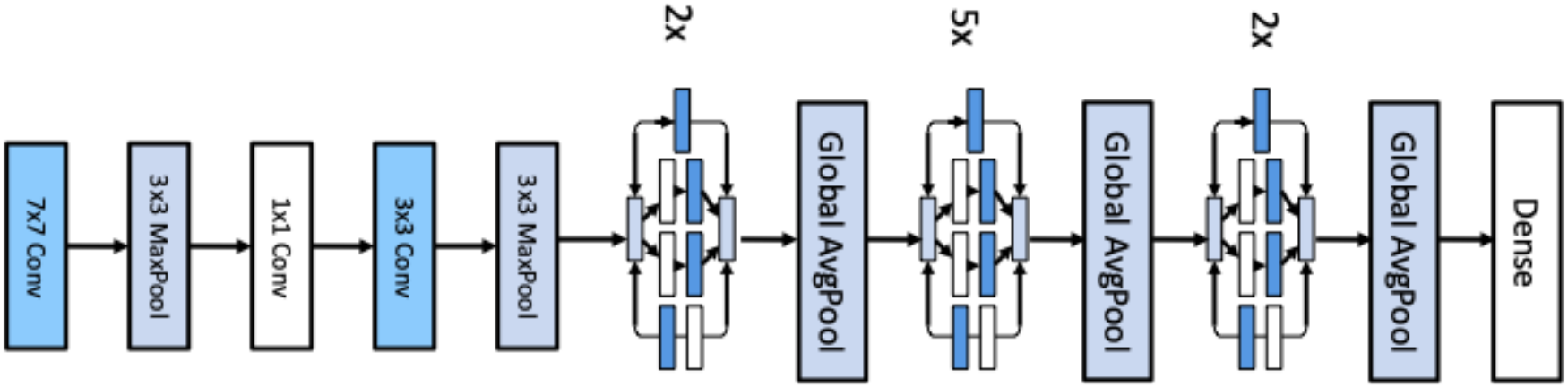


AlexNet Results

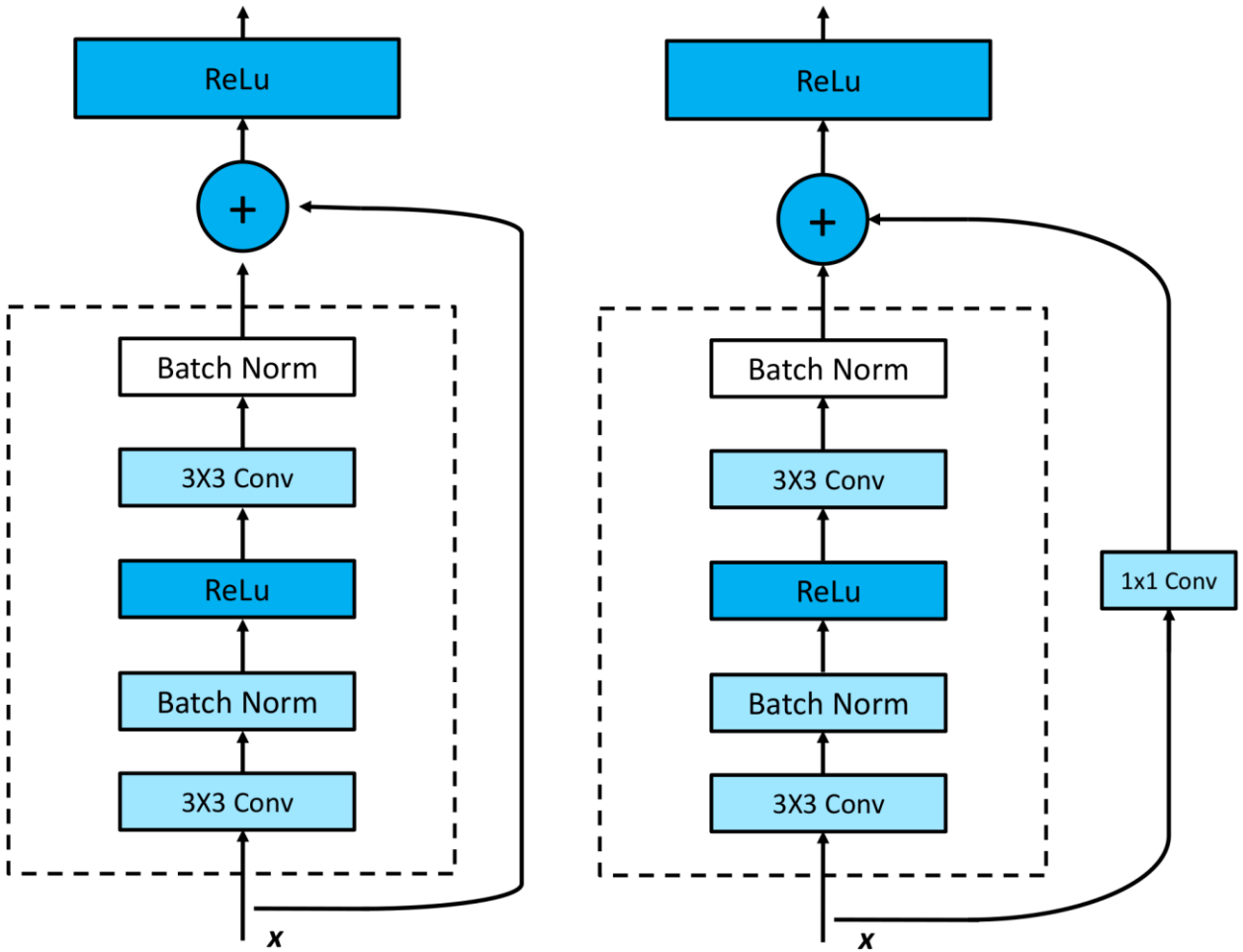


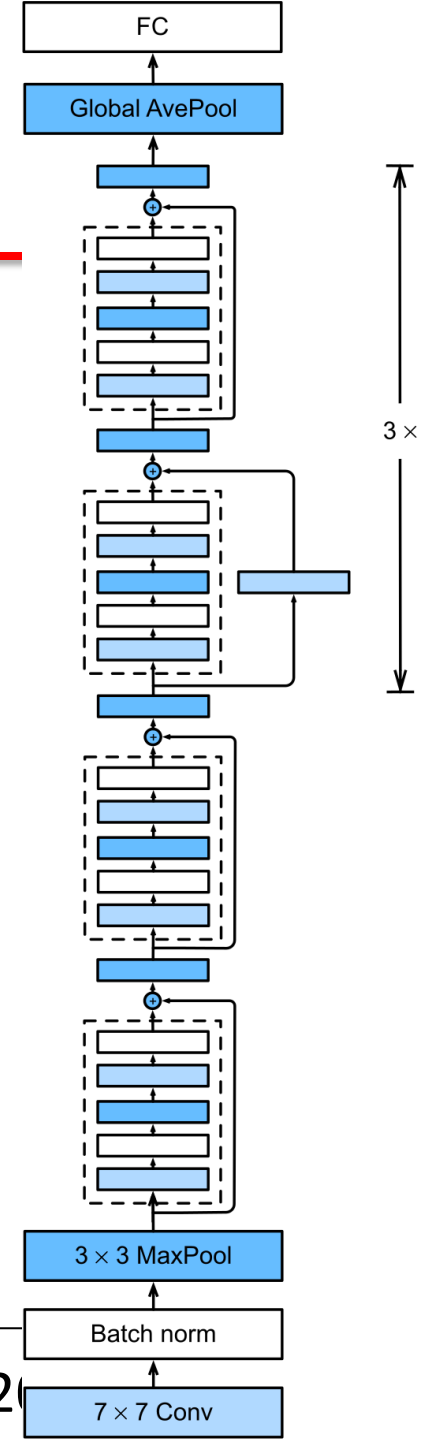
GoogLeNet (Inception Block)



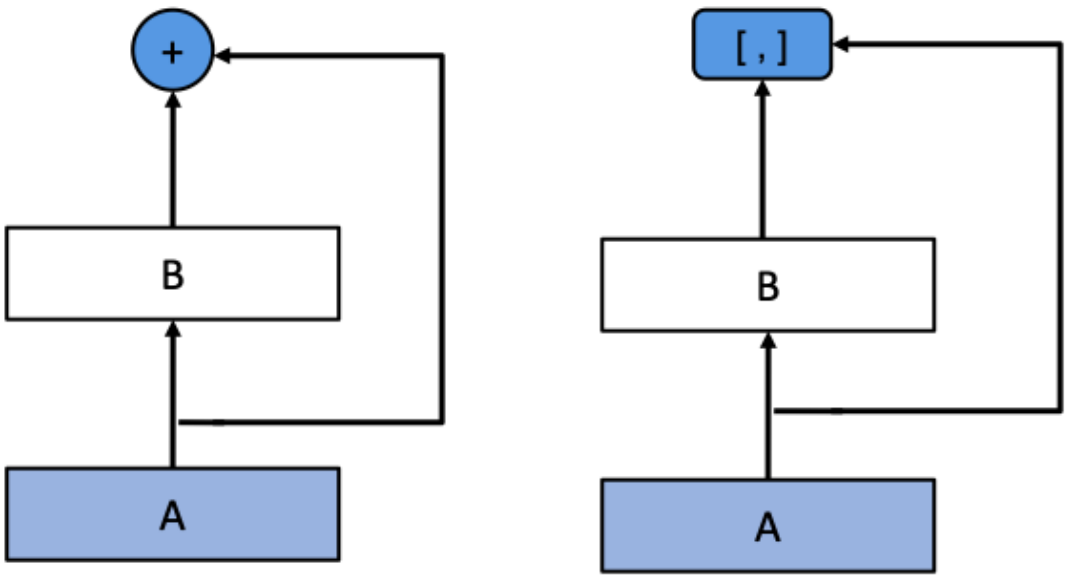


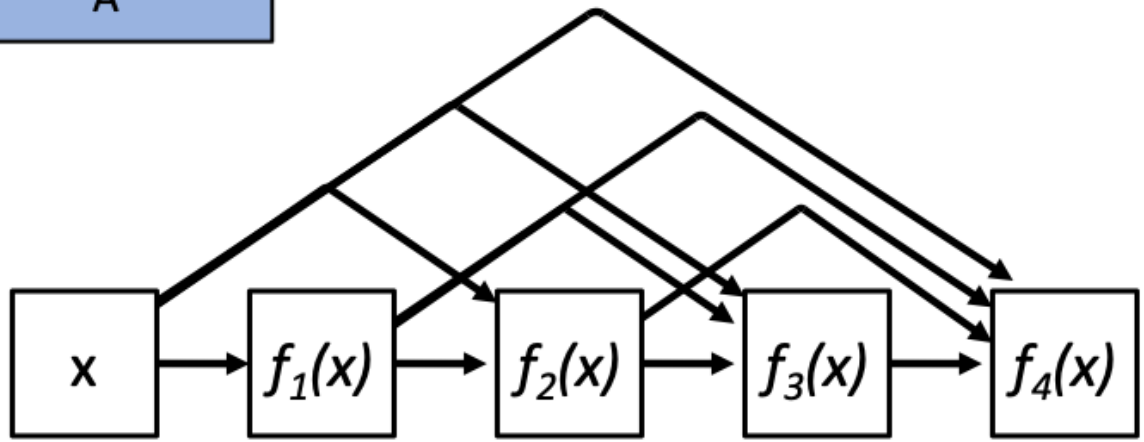
ResNet (Microsoft)





DenseNet





Object Detection (Anchor Boxes)

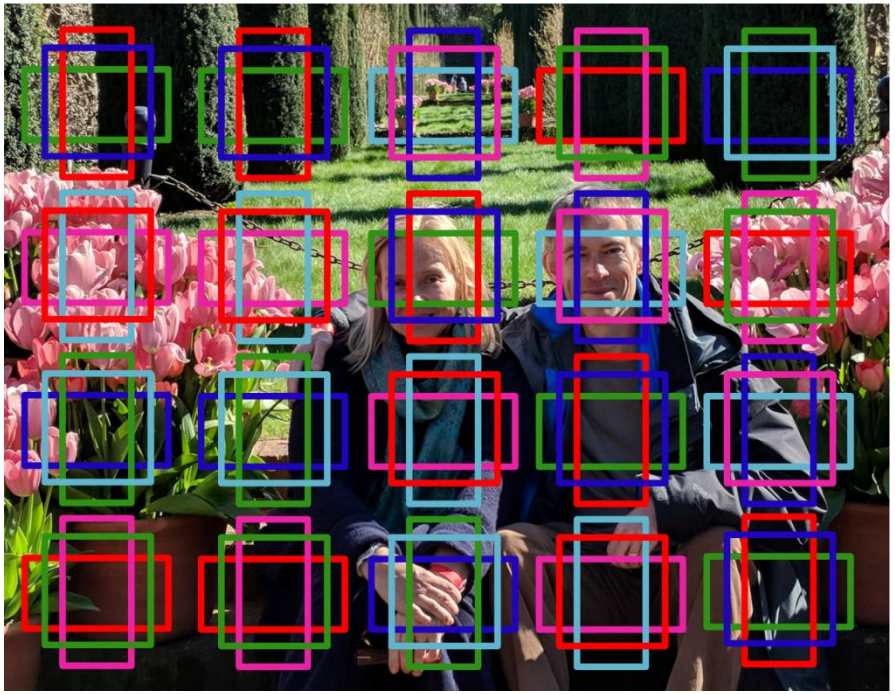
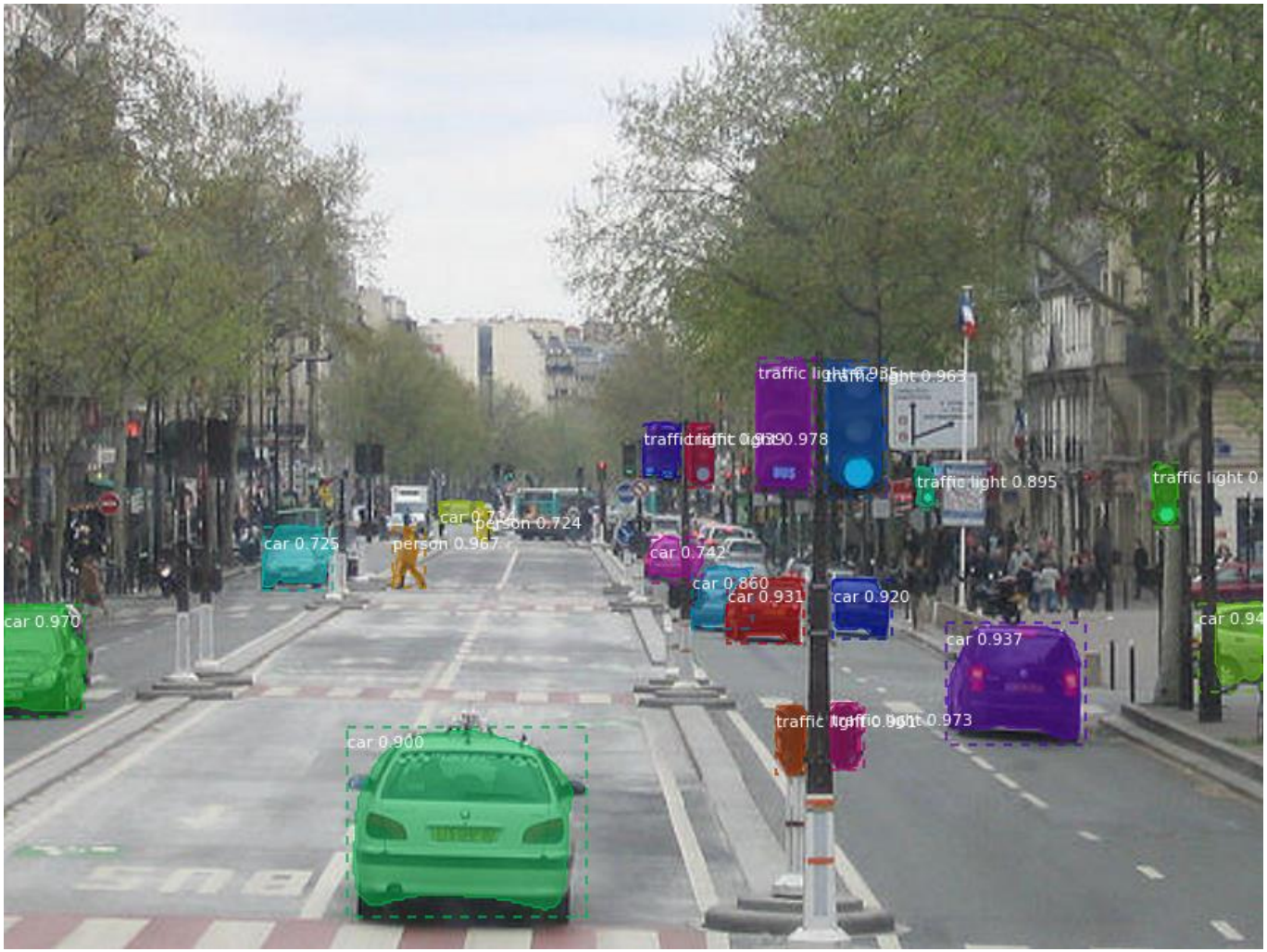
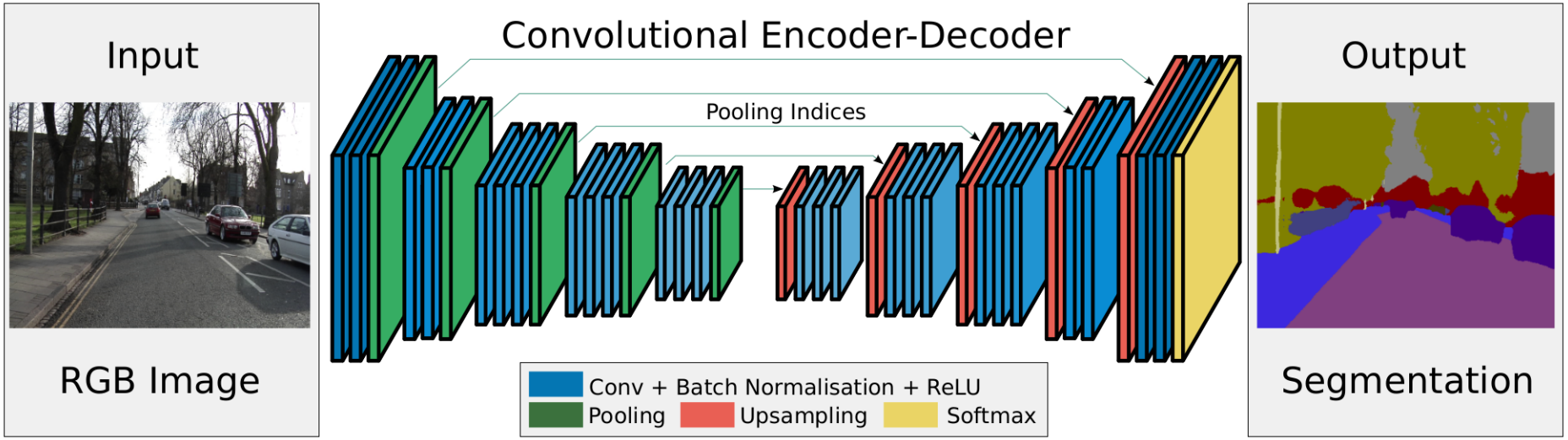


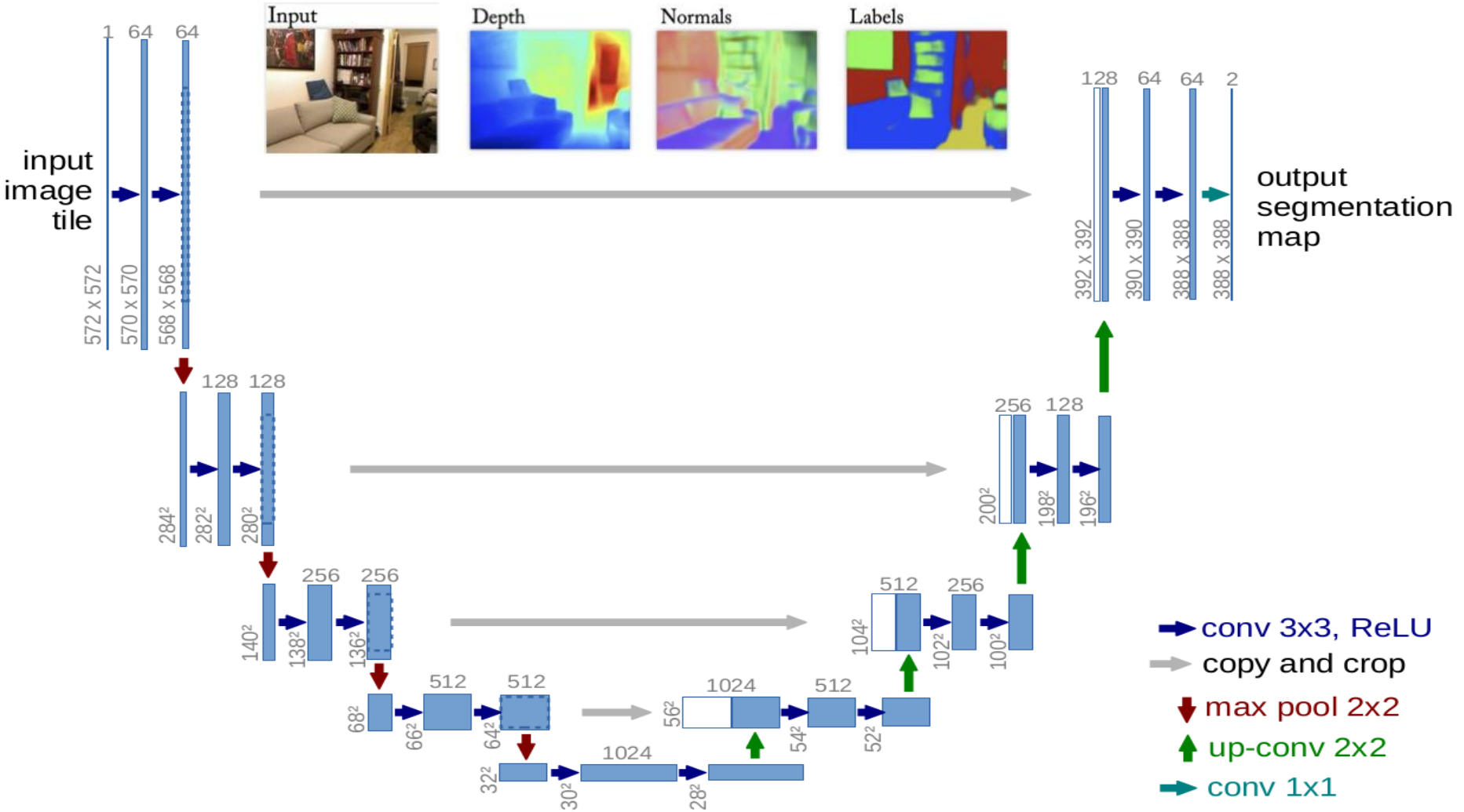
Image Segmentation



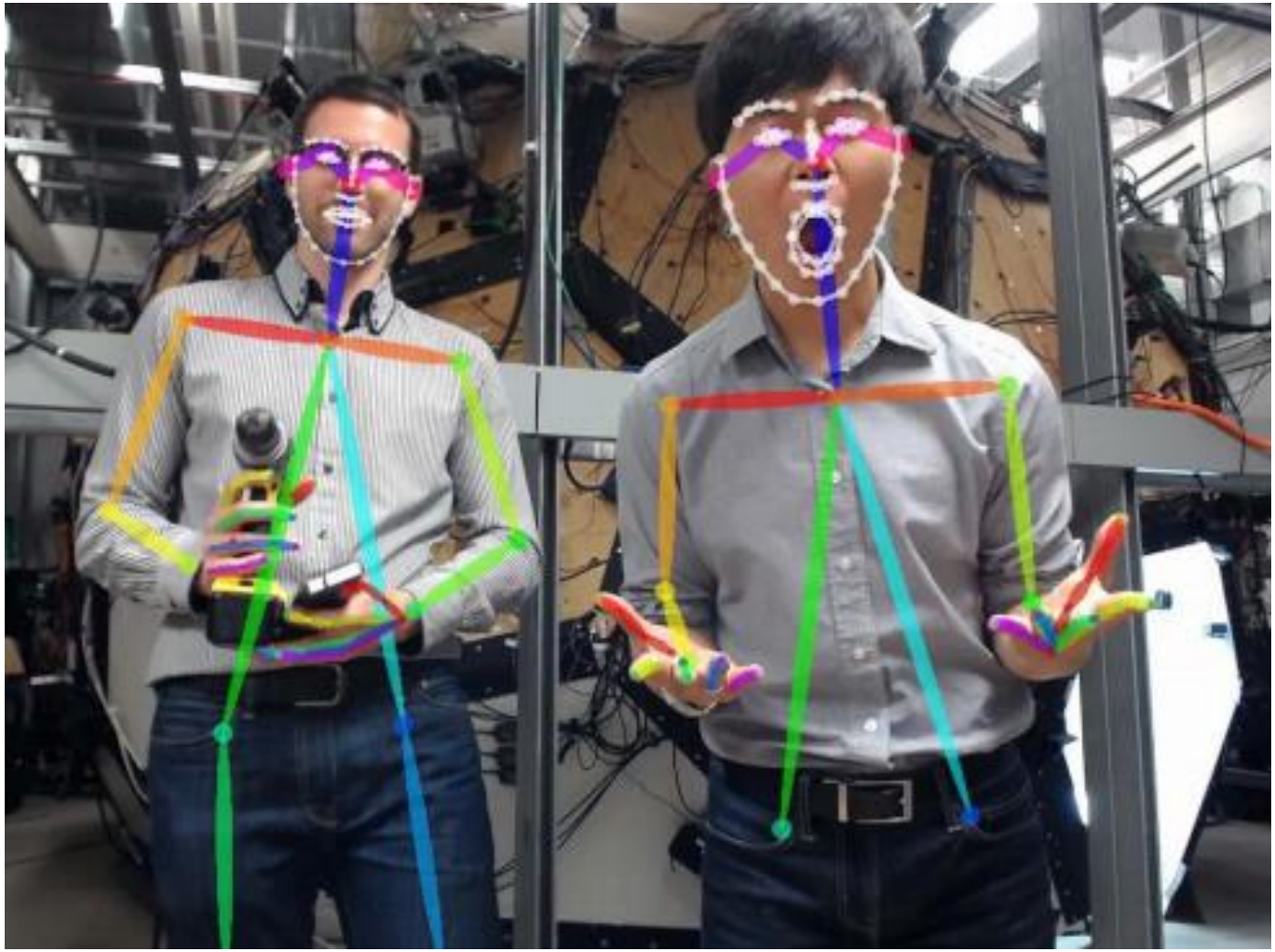
Encoder-Decoder



Semantic Segmentation



Human Pose Estimation



CNNs for Medical Imaging

- **BrainNet**: developed based on TensorFlow, aims to train deep neural networks to segment GM (Gray Matter) and WM (White Matter) from brain MR images
- **LiviaNet**: developed based on Theano, aims to train 3-D fully convolutional neural networks by using MR images to segment sub-cortical brain structures
- **DIGITS**: developed to rapidly train accurate deep neural networks for image segmentation, classification, and tissue detection tasks (e.g. Alzheimer's disease detection)
- **DeepMedic**: developed based on Theano, aims to train multi-scale 3-D convolutional neural networks for brain lesion segmentation from MR images (winner of the ISLES 2015 competition)

European Union
European Social Fund

Operational Programme
Human Resources Development,
Education and Lifelong Learning

Co-financed by Greece and the European Union

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη

Graduate Course on

Machine Learning

Lecture 26

Deep Learning

Neural Networks for Sequences

TUC ECE, Spring 2023

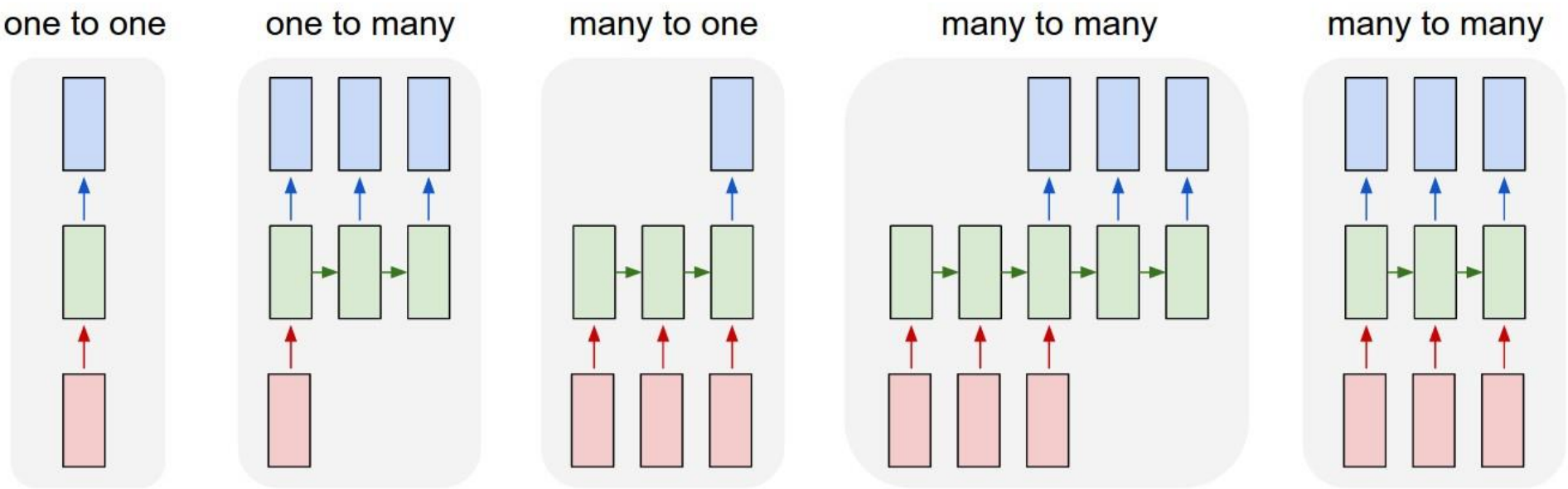
Today

- **Recurrent Neural Network (RNN)**
- **Gated Recurrent Units (GRU)**
- **Long Short Term Memory (LSTM)**
- **Considerations**
- **Attention**

Recurrent Neural Network (RNN)

- **RNNs**
  - map sequences to sequences using also internal state (memory)
  - instant output is **not** a function only of the instant input …
  - … but depends also on the **hidden state** …
  - … which is constantly **updated** over time
- **Applications**
  - sequence generation
  - sequence classification
  - sequence translation
  - …

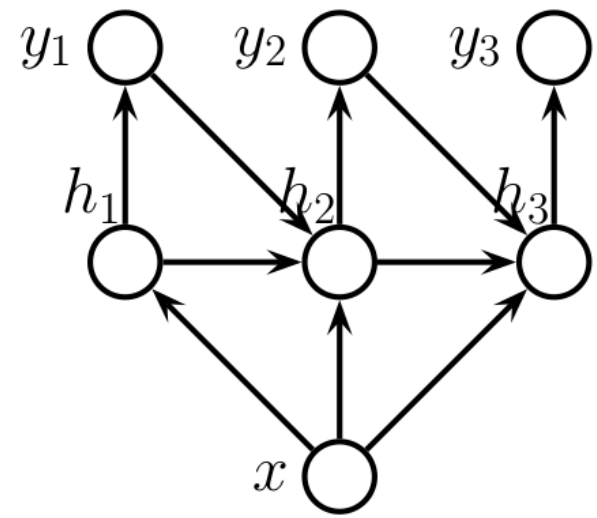
Types of RNNs



vec2seq (Sequence Generation)

$$f_{\boldsymbol{\theta}} : \mathbb{R}^D \rightarrow \mathbb{R}^{N_\infty C}$$

- *D*: size of the input vector
- output: arbitrary-length sequence of vectors
- *C*: size of each output vector
- output vectors generated one at a time
- output vectors are sampled
- hidden states are deterministic
- conditional generative model
- variation: variational RNNs

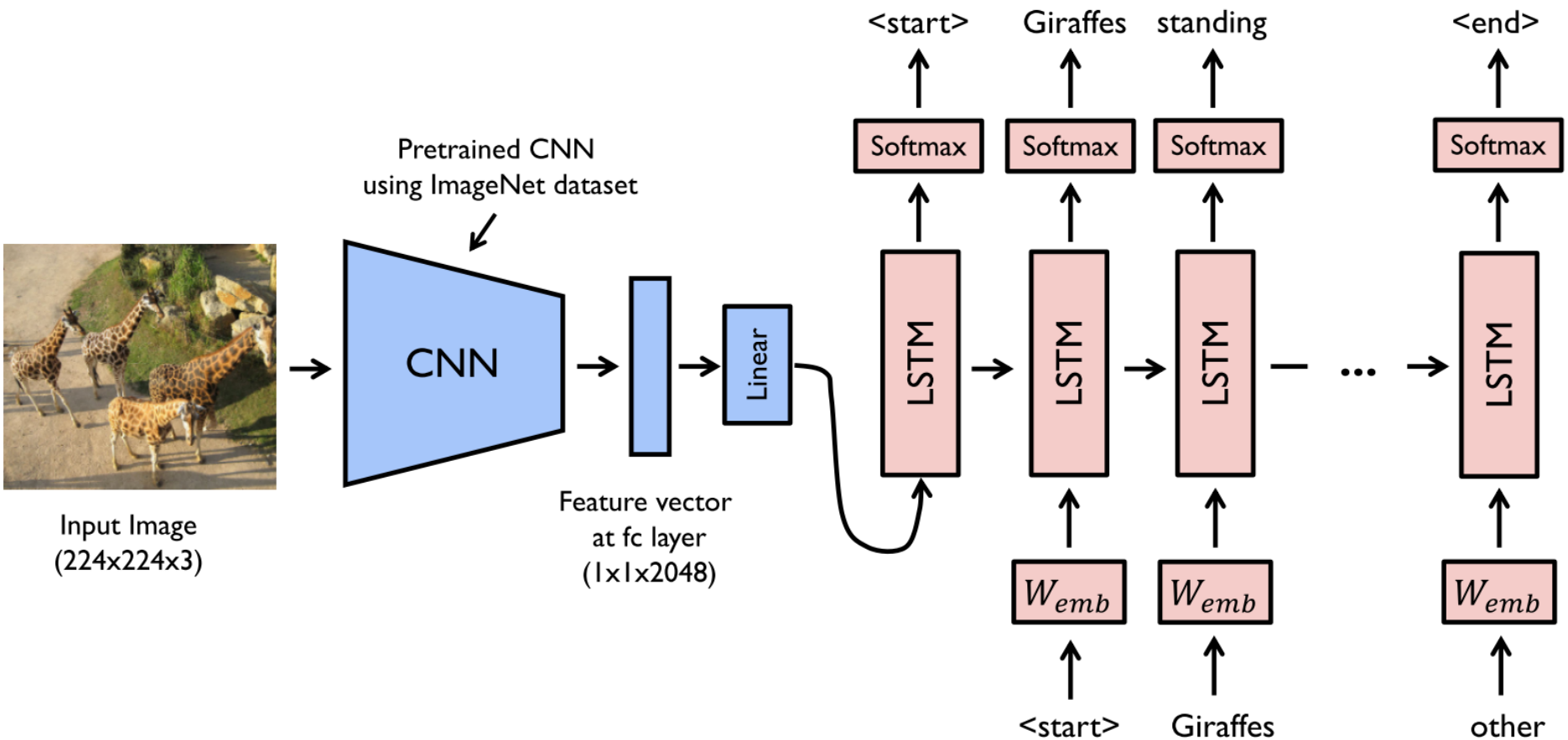


$$p(\boldsymbol{y}_{1:T}|\boldsymbol{x}) = \sum_{\boldsymbol{h}_{1:T}} p(\boldsymbol{y}_{1:T}, \boldsymbol{h}_{1:T}|\boldsymbol{x}) = \sum_{\boldsymbol{h}_{1:T}} \prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{h}_t) p(\boldsymbol{h}_t|\boldsymbol{h}_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{x})$$

vec2seq Applications

- **Language Modelling**
  - no input! just generation of output
- **Language Modelling example**
  - *the githa some thong the time traveller held in his hand was a glittering metallic framework scarcely larger than a small clock and very delicately made there was ivory in it and the latter than s bettyre tat howhong s ie time thave ler simk you a dimensions le ghat dionthat shall travel indifferently in any direction of space and time as the driver determines …*
- **Image Captioning**
  - image (or an embedding of) as input
  - a textual description of the image content as output

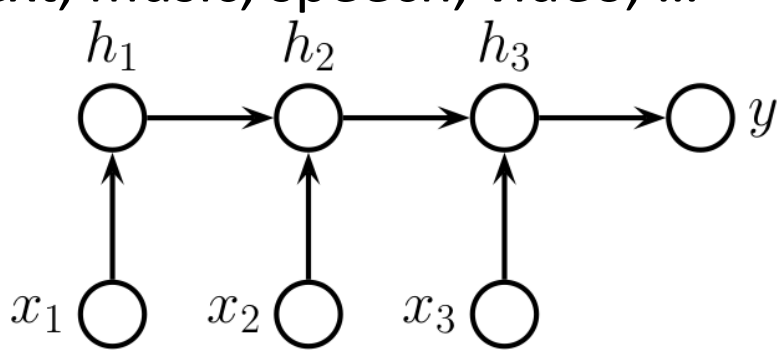
vec2seq: Image Captioning Example

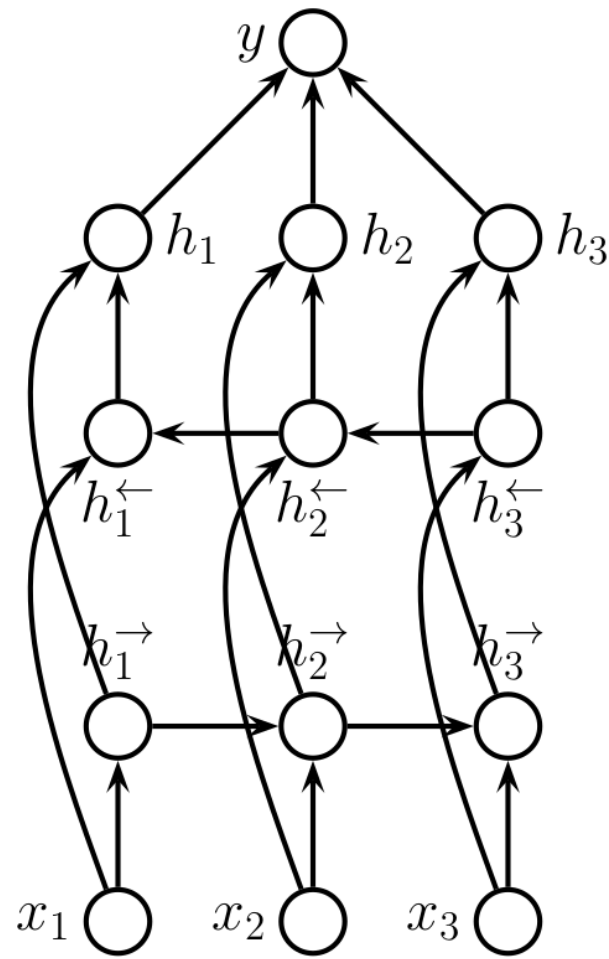


seq2vec (Sequence Classification)

$$f_{\boldsymbol{\theta}} : \mathbb{R}^{TD} \rightarrow \mathbb{R}^{C}$$

- *D*: size of each input vector
- input: length-*T* sequence of vectors
- *C*: size of the output vector
- variation: bidirectional RNNs
- application: classification of …
- … text, music, speech, video, …

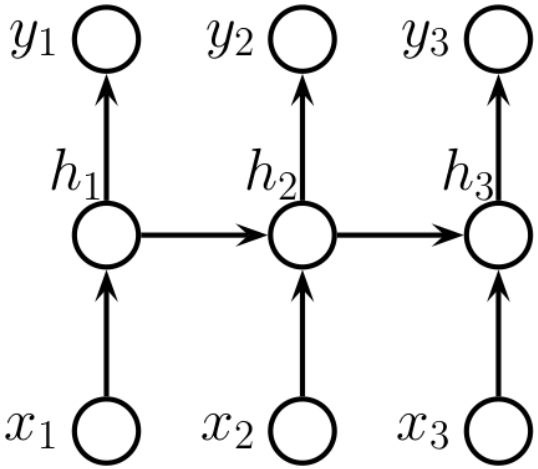


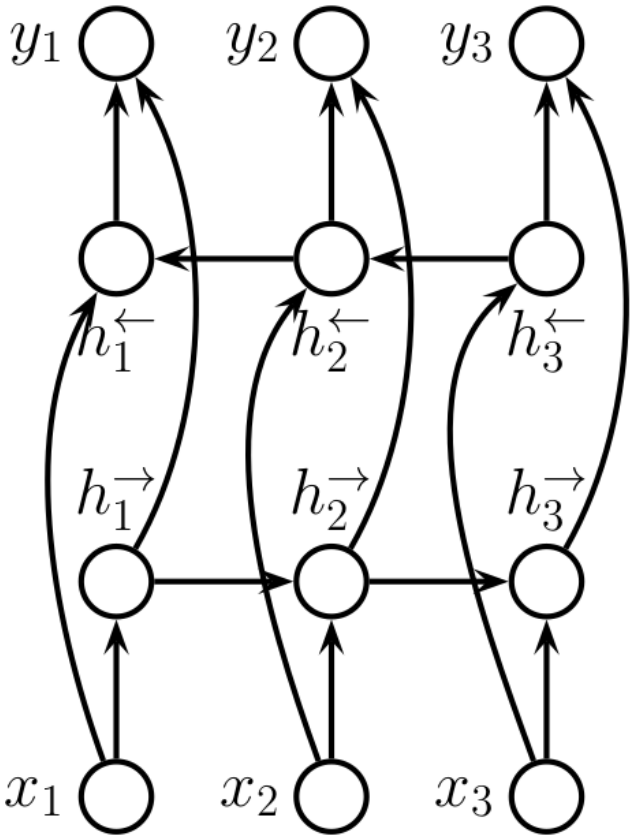


vec2vec (Sequence Translation)

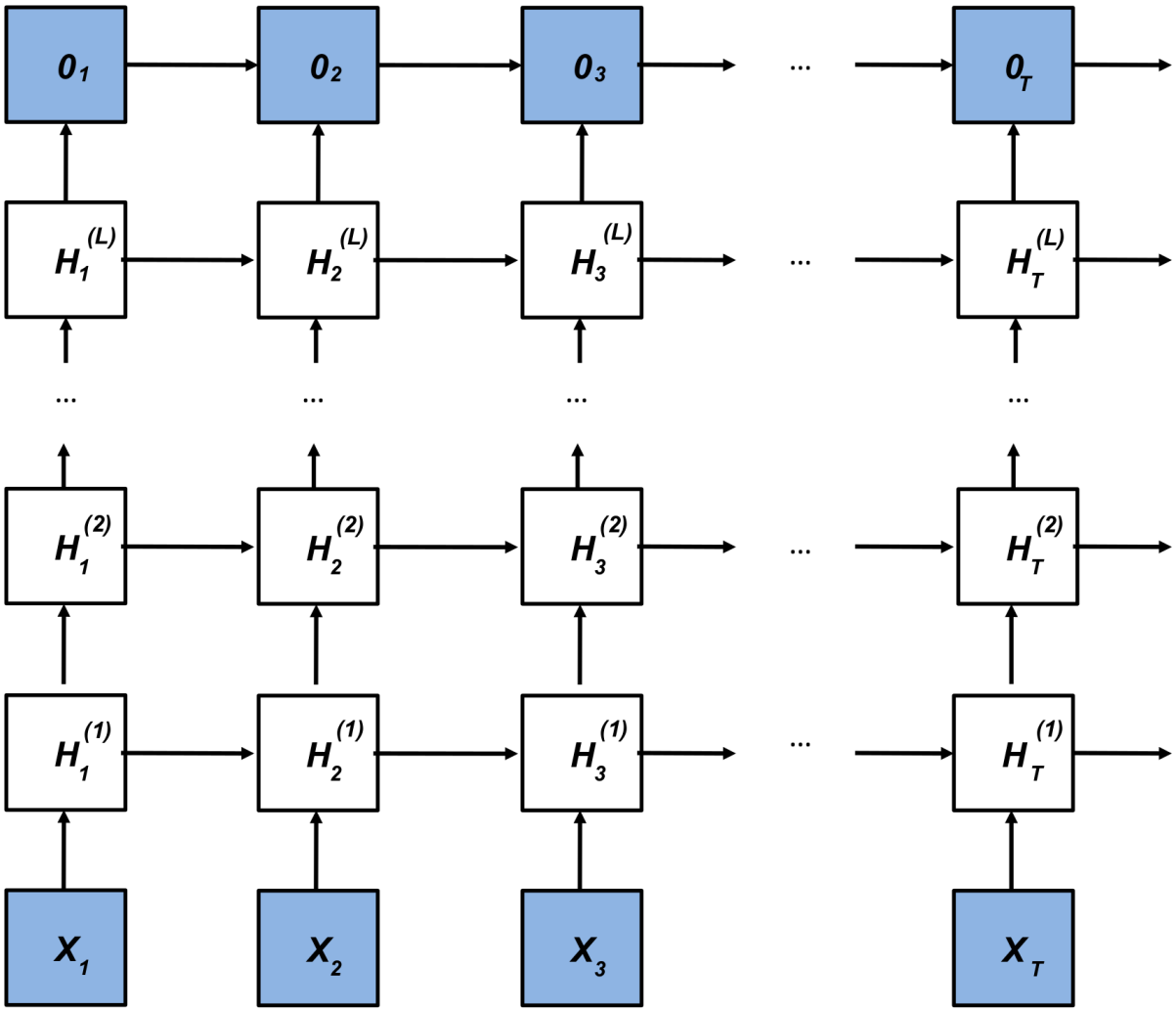
$$f_{\boldsymbol{\theta}} : \mathbb{R}^{TD} \rightarrow \mathbb{R}^{T'C}$$

- *D*: size of each input vector
- *T*, *T'*: length of input/output sequence
- *C*: size of the output vector
- *T*=*T'* → aligned case
  (dense sequence labeling)





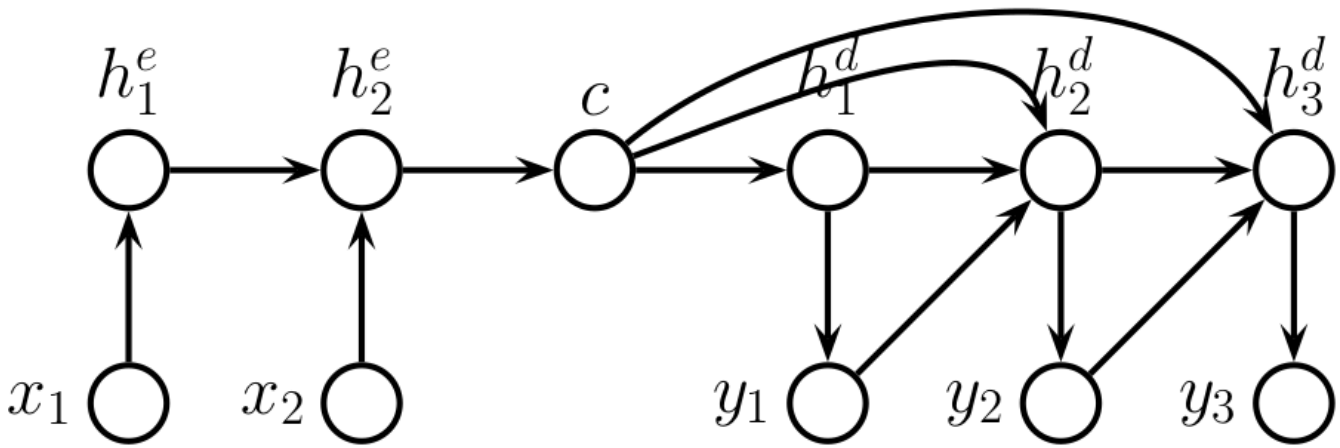
vec2vec Deep RNN



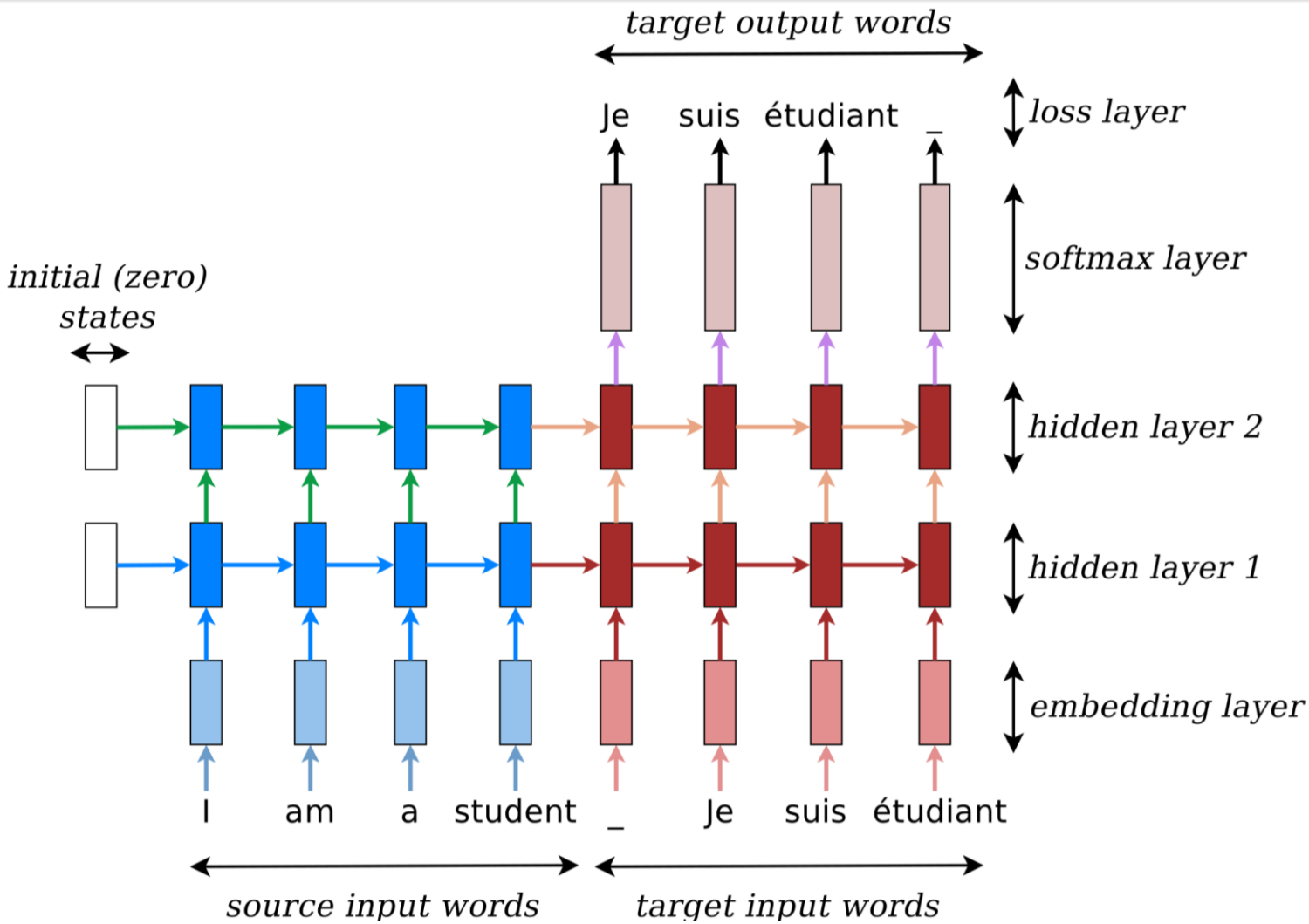
vec2vec (Sequence Translation)

$$f_{\boldsymbol{\theta}} : \mathbb{R}^{TD} \rightarrow \mathbb{R}^{T'C}$$

- *D*: size of each input vector
- *T*, *T'*: length of input/output sequence
- *C*: size of the output vector
- *T*≠*T'* → unaligned case
  (encoder-decoder architecture)



vec2vec Machine Translation



Gated Recurrent Units (GRU)

Gated Recurrent Units (GRU)

- **Key idea**
  - learn when to update the hidden state
  - selectively "remember" important info when first seen
  - learn when to reset the hidden state
  - thus, forget things that are no longer useful

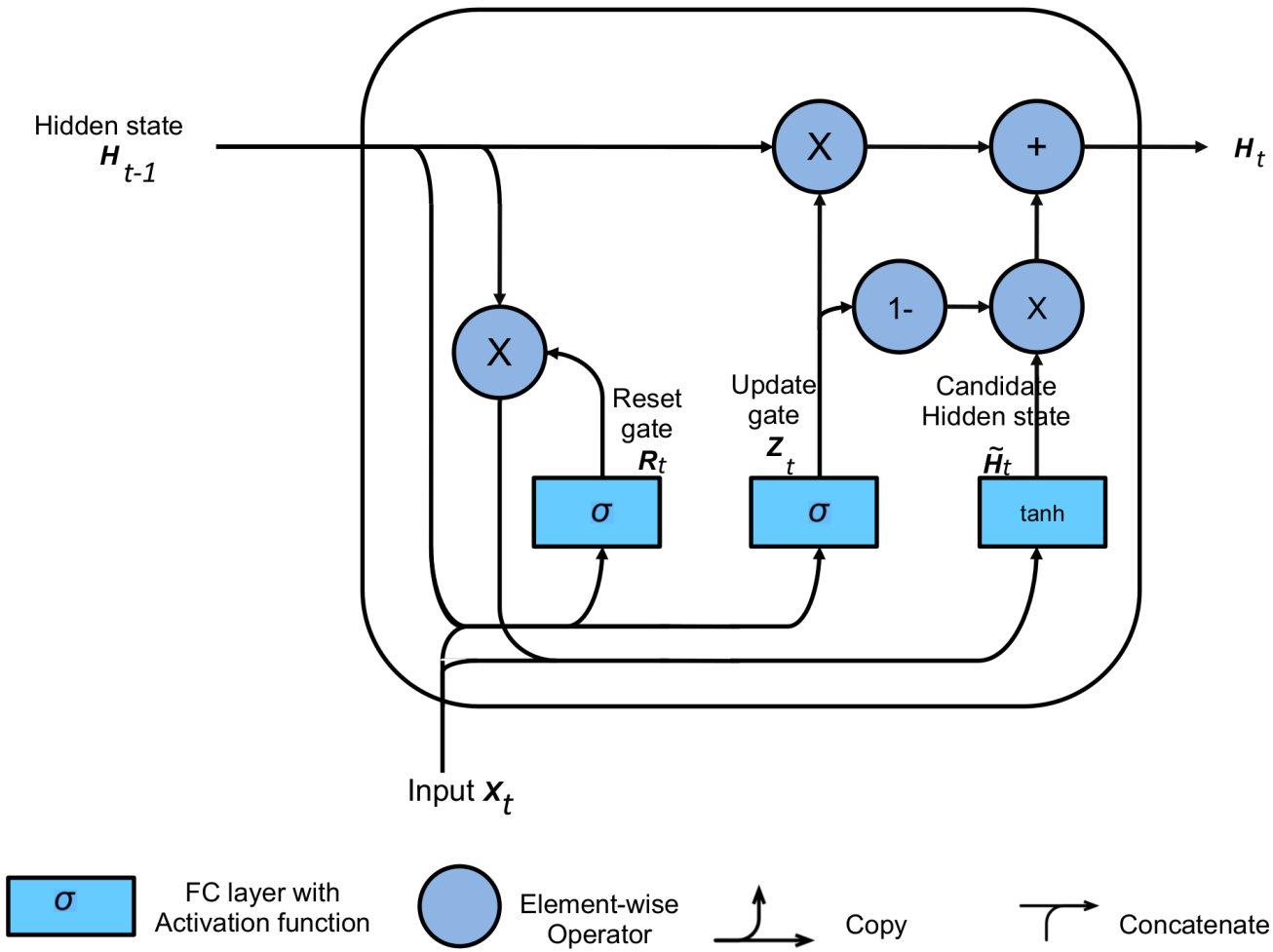
reset gate $\mathbf{R}_t \in \mathbb{R}^{N \times H}$ $\qquad \mathbf{R}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xr} + \mathbf{H}_{t-1} \mathbf{W}_{hr} + \boldsymbol{b}_r)$

update gate $\mathbf{Z}_t \in \mathbb{R}^{N \times H}$ $\qquad \mathbf{Z}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xz} + \mathbf{H}_{t-1} \mathbf{W}_{hz} + \boldsymbol{b}_z)$

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + (\mathbf{R}_t \odot \mathbf{H}_{t-1}) \mathbf{W}_{hh} + \boldsymbol{b}_h)$$

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t$$

GRU Illustrated



Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM)

- **Key idea**
  - similar to GRU, but more sophisticated
  - augment the hidden states with memory cells

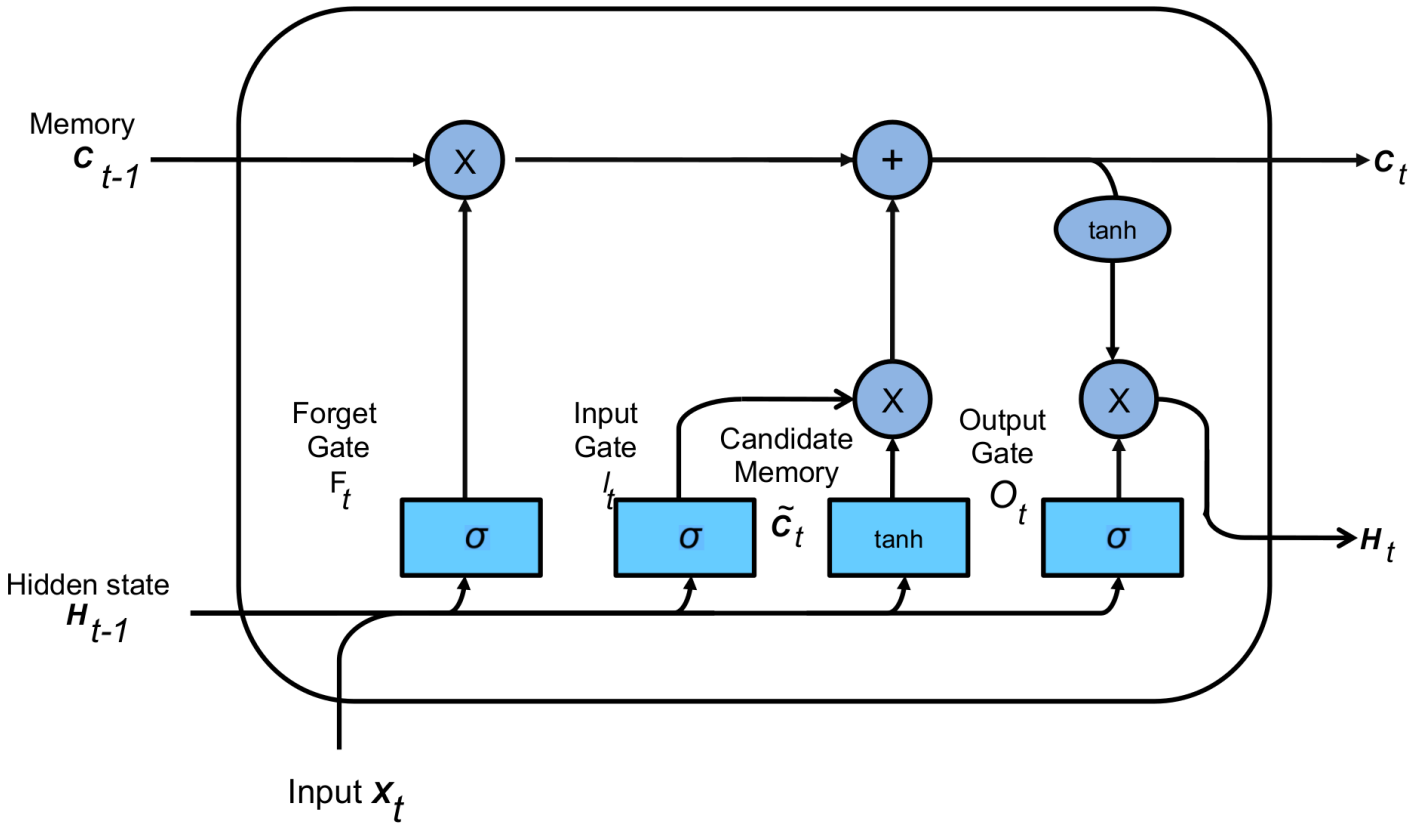
  - output gate $\mathbf{O}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xo} + \mathbf{H}_{t-1} \mathbf{W}_{ho} + \boldsymbol{b}_o)$
  - input gate $\mathbf{I}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xi} + \mathbf{H}_{t-1} \mathbf{W}_{hi} + \boldsymbol{b}_i)$
  - forget gate $\mathbf{F}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xf} + \mathbf{H}_{t-1} \mathbf{W}_{hf} + \boldsymbol{b}_f)$

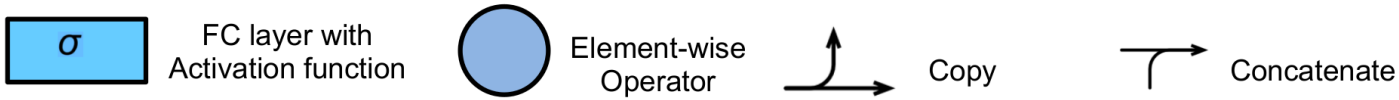
$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xc} + \mathbf{H}_{t-1} \mathbf{W}_{hc} + \boldsymbol{b}_c)$$

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t$$

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t)$$

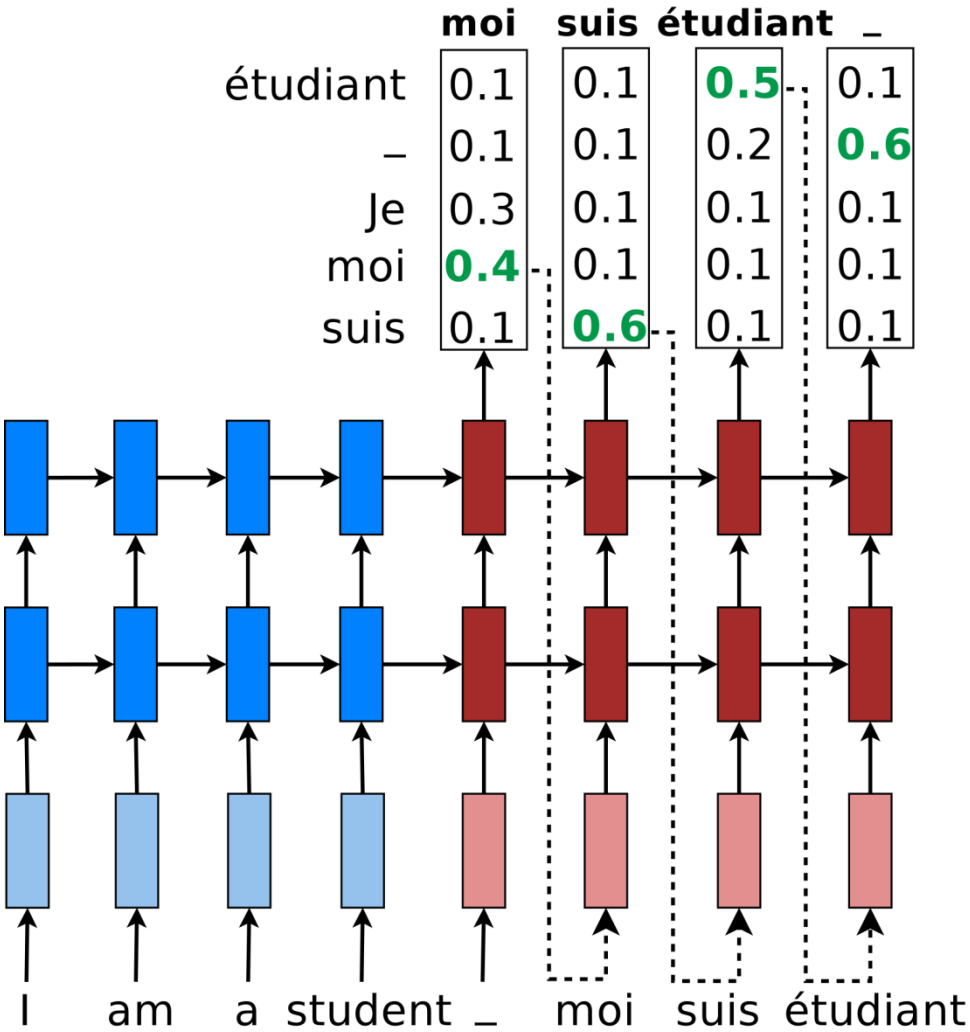
LSTM Illustrated





Considerations

Greedy Decoding



Examples

Time step	1	2	3	4
A	0.5	0.1	0.2	0.0
B	0.2	0.4	0.2	0.2
C	0.2	0.3	0.4	0.2
<eos>	0.1	0.2	0.2	0.6

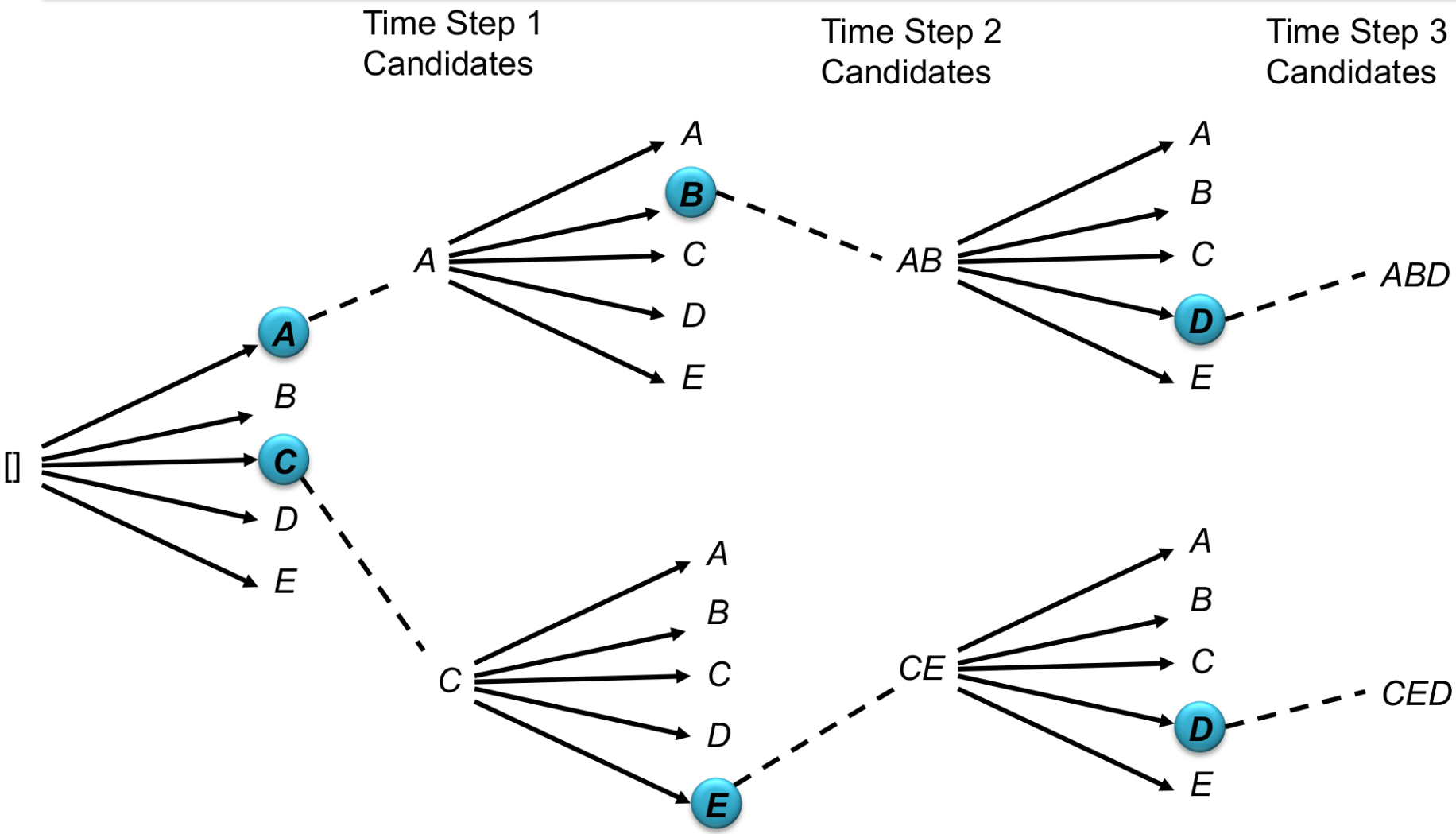
$$0.5 \times 0.4 \times 0.4 \times 0.6 = 0.048$$

Time step	1	2	3	4
A	0.5	0.1	0.1	0.1
B	0.2	0.4	0.6	0.2
C	0.2	0.3	0.2	0.1
<eos>	0.1	0.2	0.1	0.6

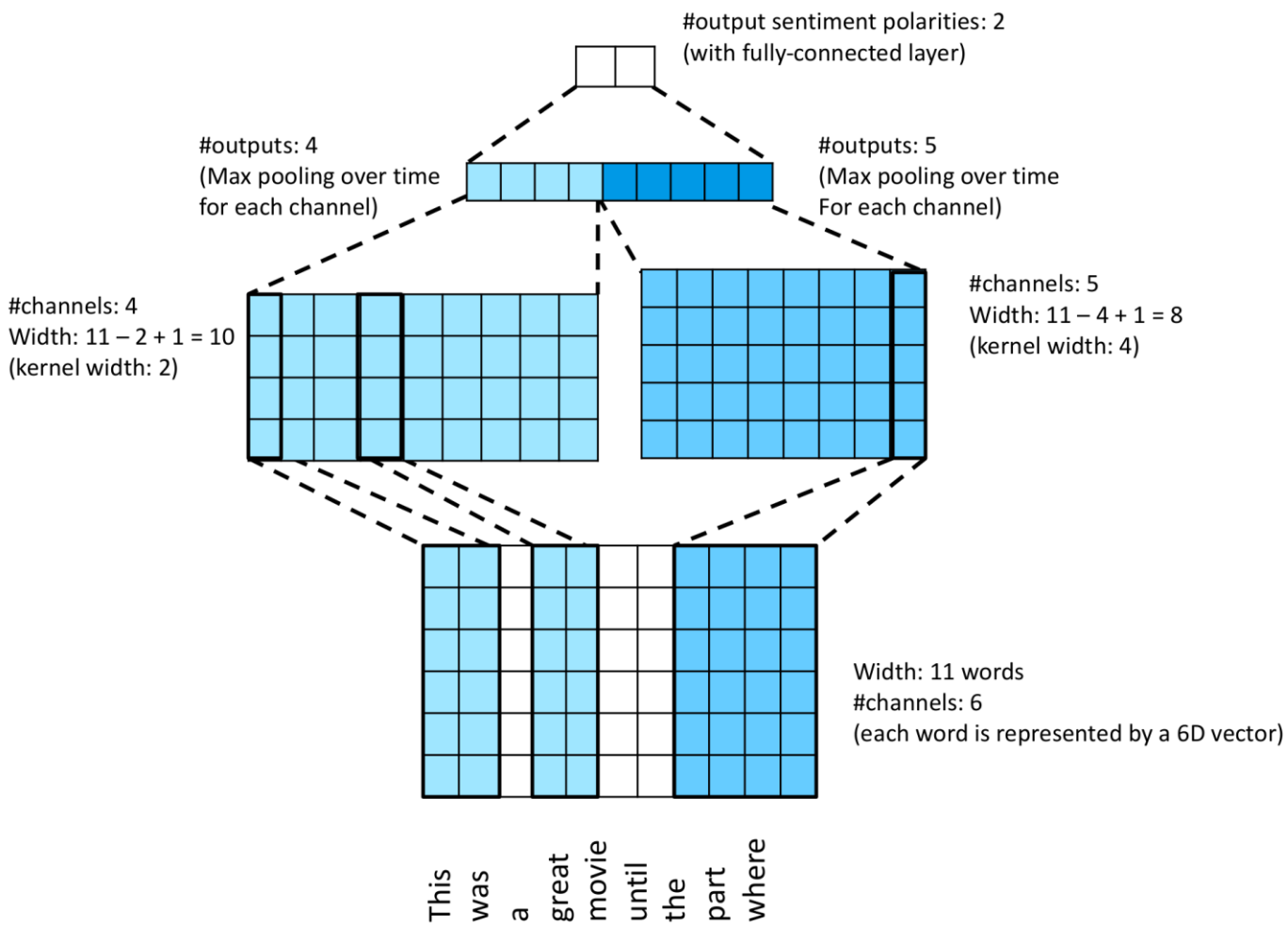
$$0.5 \times 0.3 \times 0.6 \times 0.6 = 0.054$$

- Viterbi: optimal, but expensive
- beam search: heuristic, but cheap

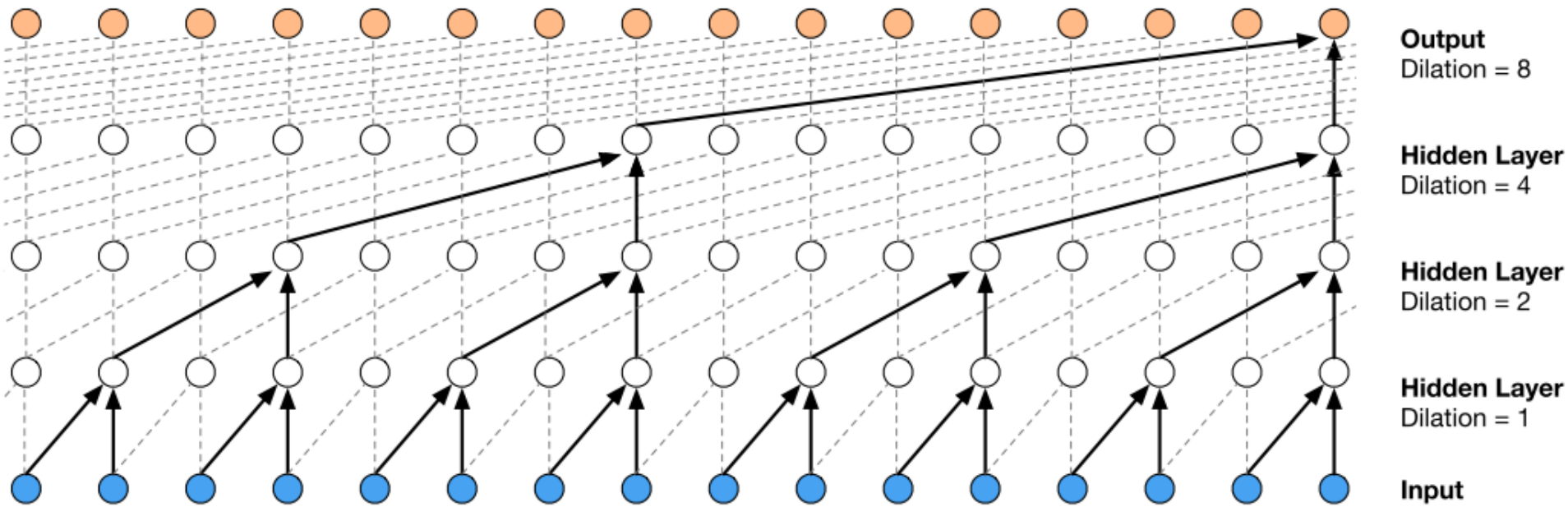
Beam Search


Time Step 1
Candidates
Time Step 2
Candidates
Time Step 3
Candidates
[]
A
B
C
D
E
A
A
B
C
D
E
AB
A
B
C
D
E
ABD
C
A
B
C
D
E
CE
A
B
C
D
E
CED

1d CNNs for Sequence Classification

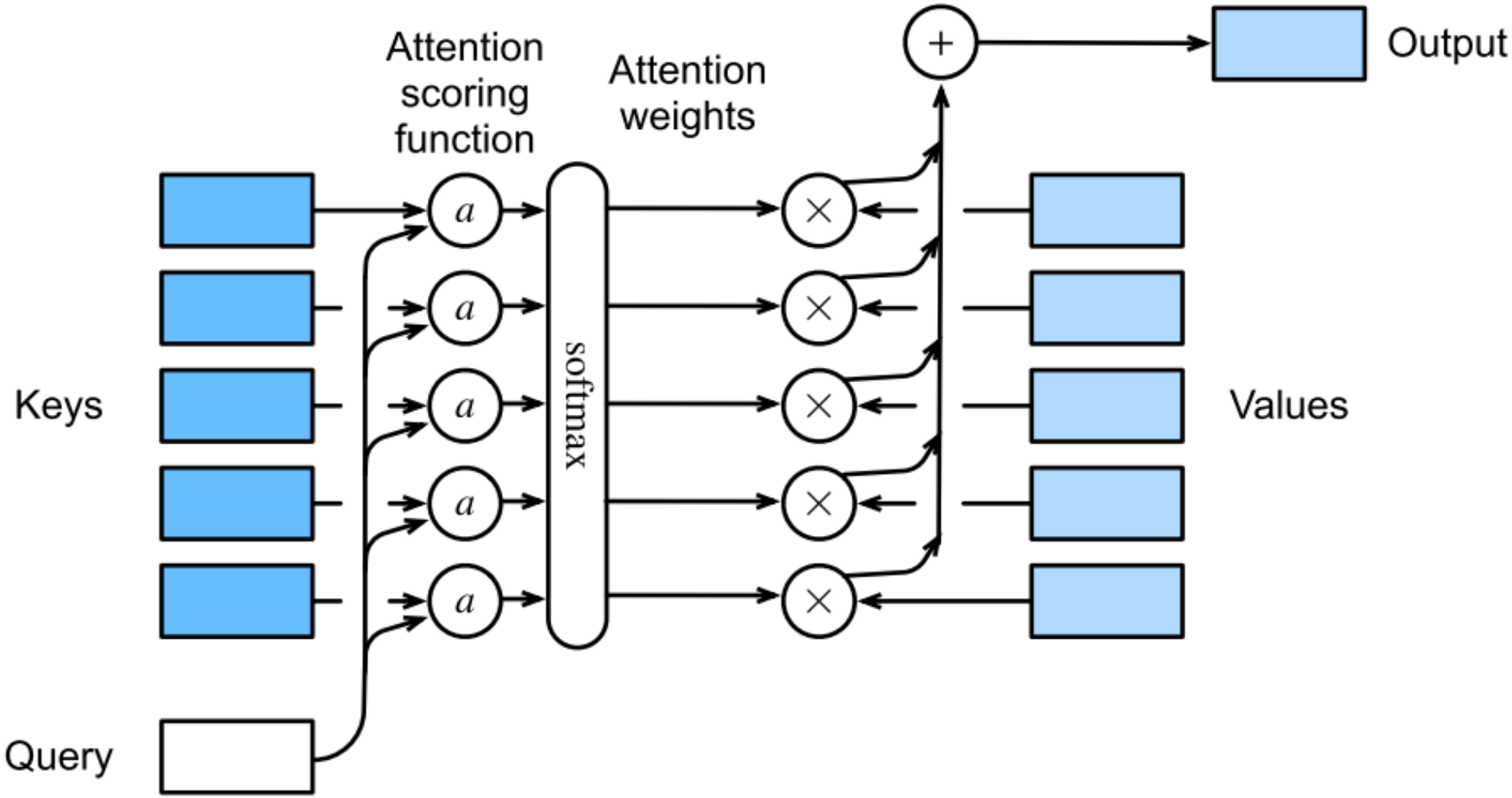


Causal 1d CNNs for Sequence Classification

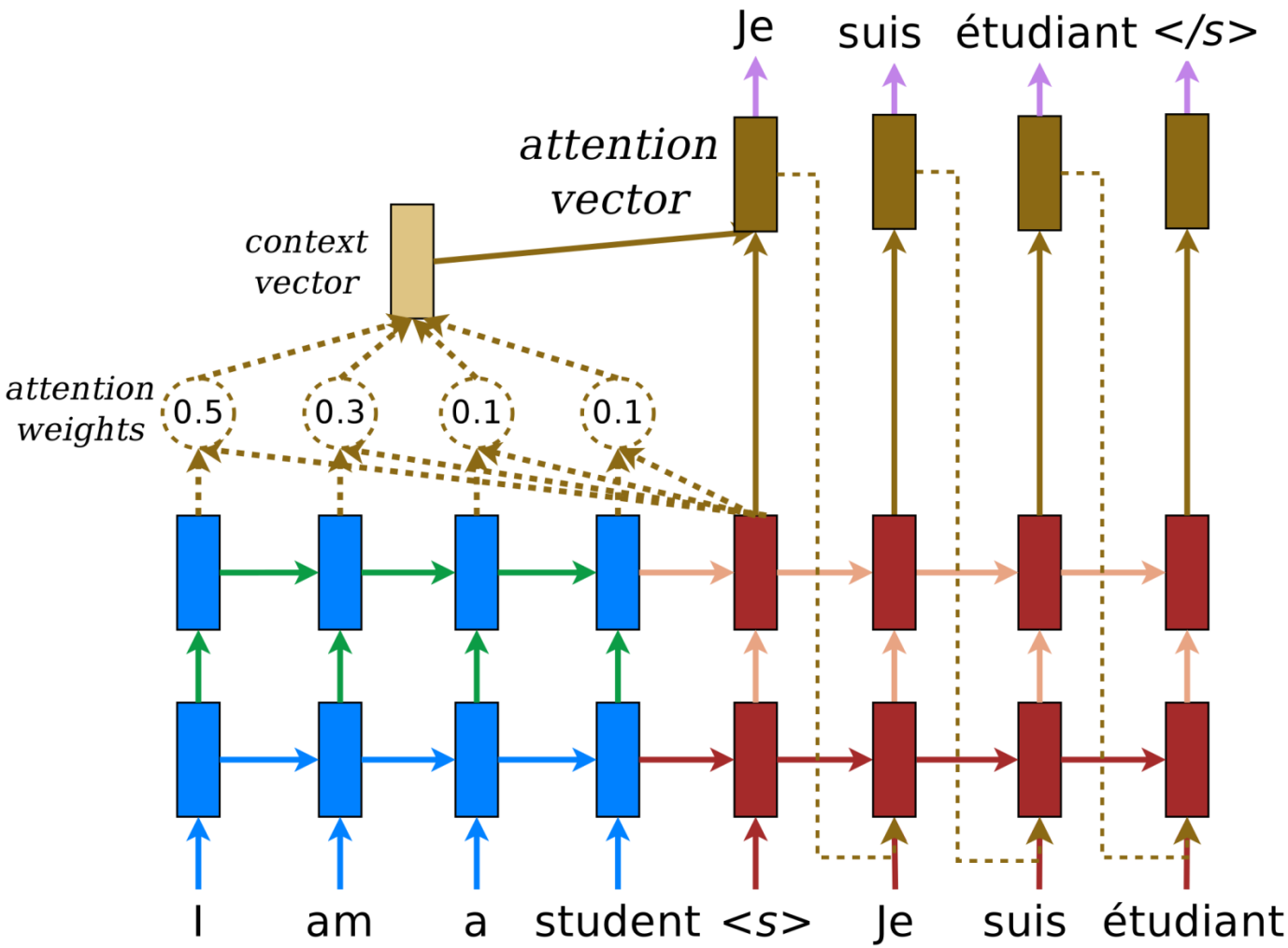


Attention

Attention

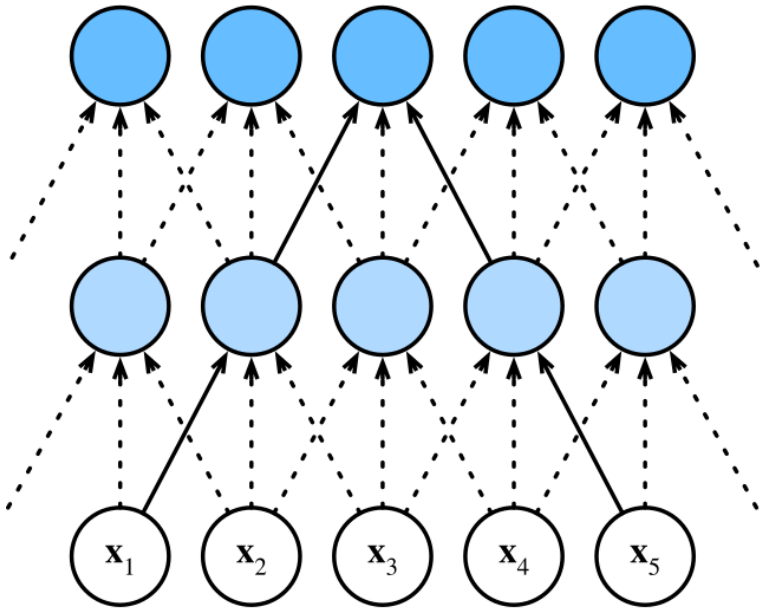

Attention scoring function
Attention weights
Output
Keys
Values
Query
a
softmax
+

Attention with seq2seq

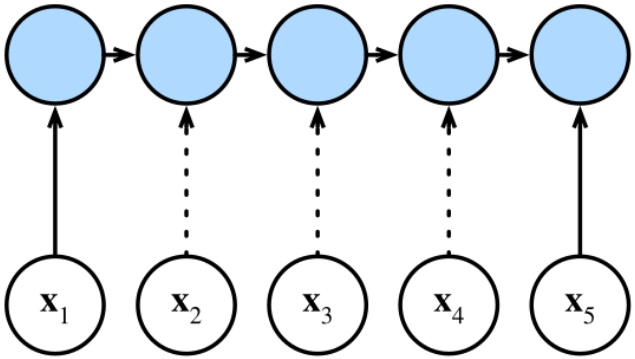


CNN, RNN, Attention

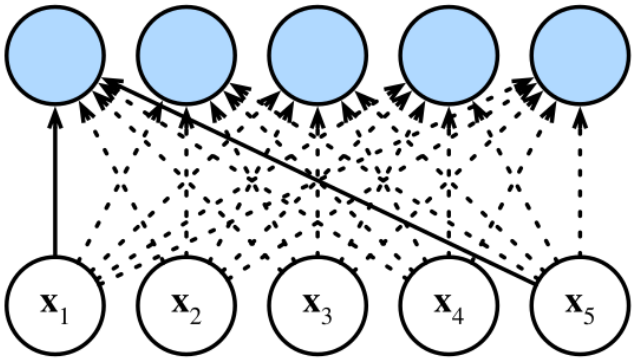
CNN



RNN



Self-attention



Layer type	Complexity	Sequential ops.	Max. path length
Self-attention	$O(n^2 d)$	$O(1)$	$O(1)$
Recurrent	$O(nd^2)$	$O(n)$	$O(n)$
Convolutional	$O(knd^2)$	$O(1)$	$O(\log_k n)$