GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Deep Learning

Lecture 3: Machine learning basics

Alexander Ecker
Institut für Informatik, Uni Göttingen

alxecker

https://alexanderecker.wordpress.com

– Credit: slides largely based on Ian Goodfellow's and Chris Bishop's slides –

Homework assignments

First homework assignment will be on Stud.IP on Friday Nov. 29 (tomorrow)

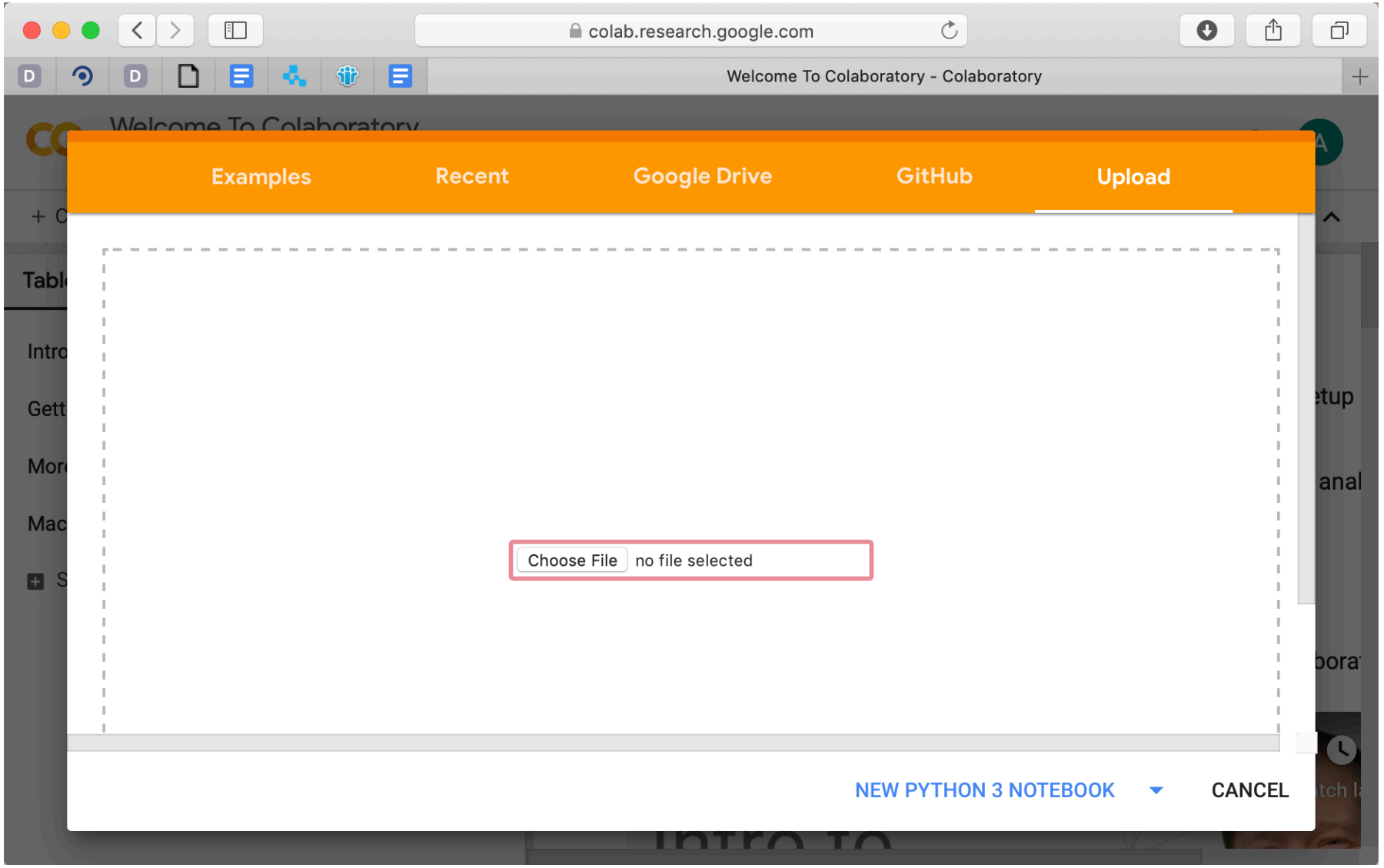
Form teams of 2–3 students

- No individuals please
- You'll present your solution to the tutors in tutorial (5 min)
- It's ok to split work, but everybody needs to be able to explain everything

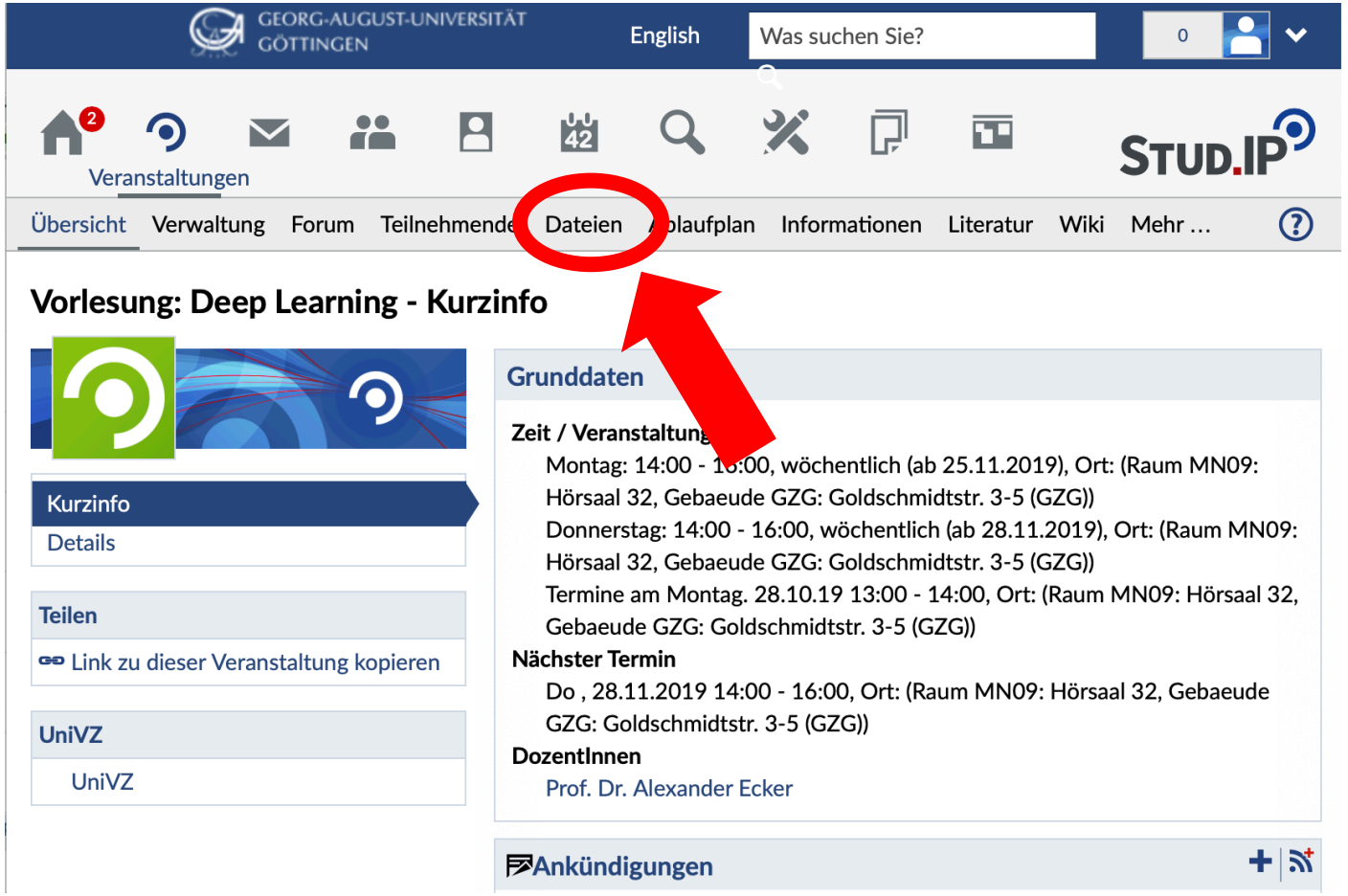
Submission

- Upload complete Jupyter Notebook including output on Stud.IP (see following slides)
- Deadline for upload: Dec. 16 at noon

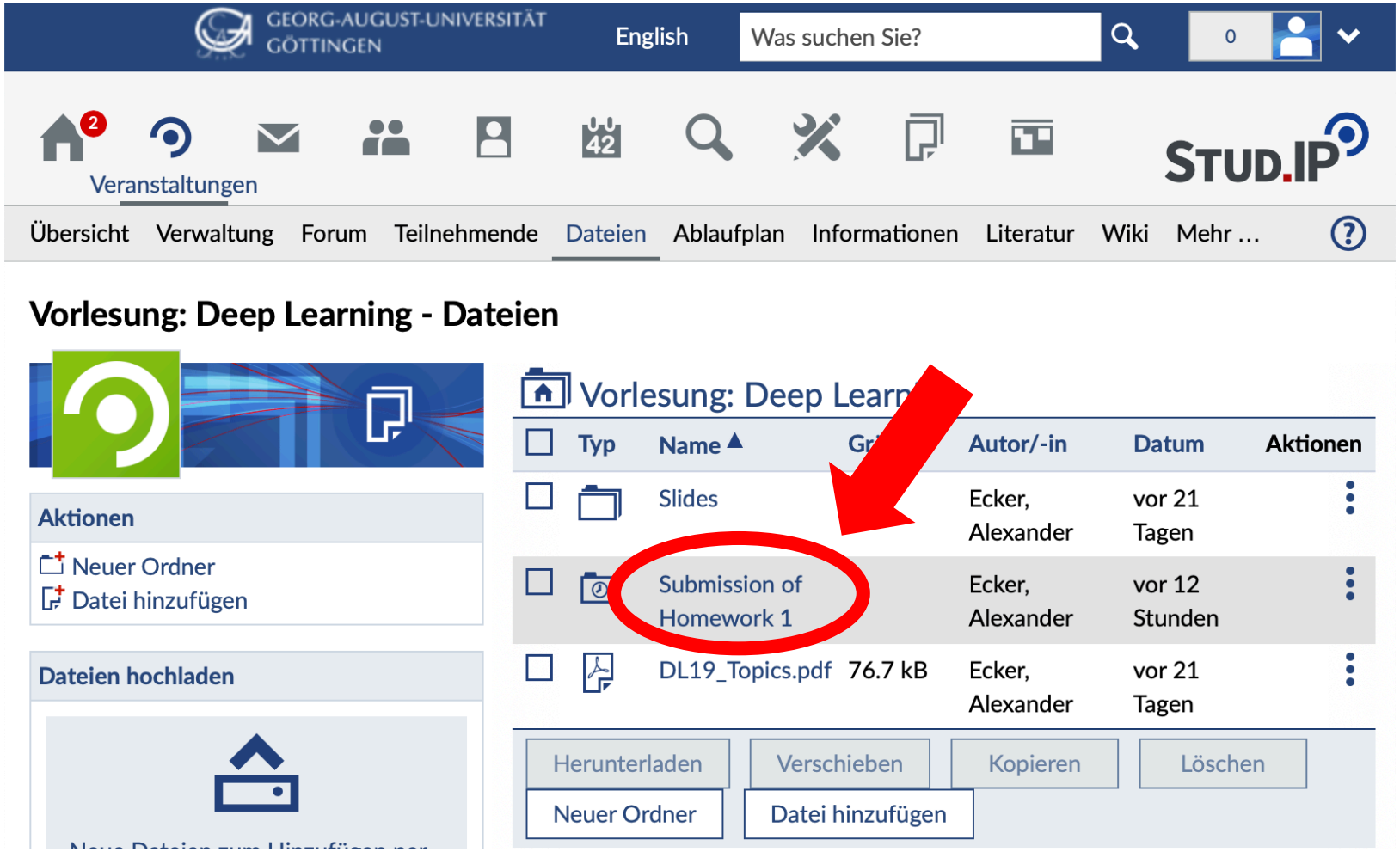
Google colab



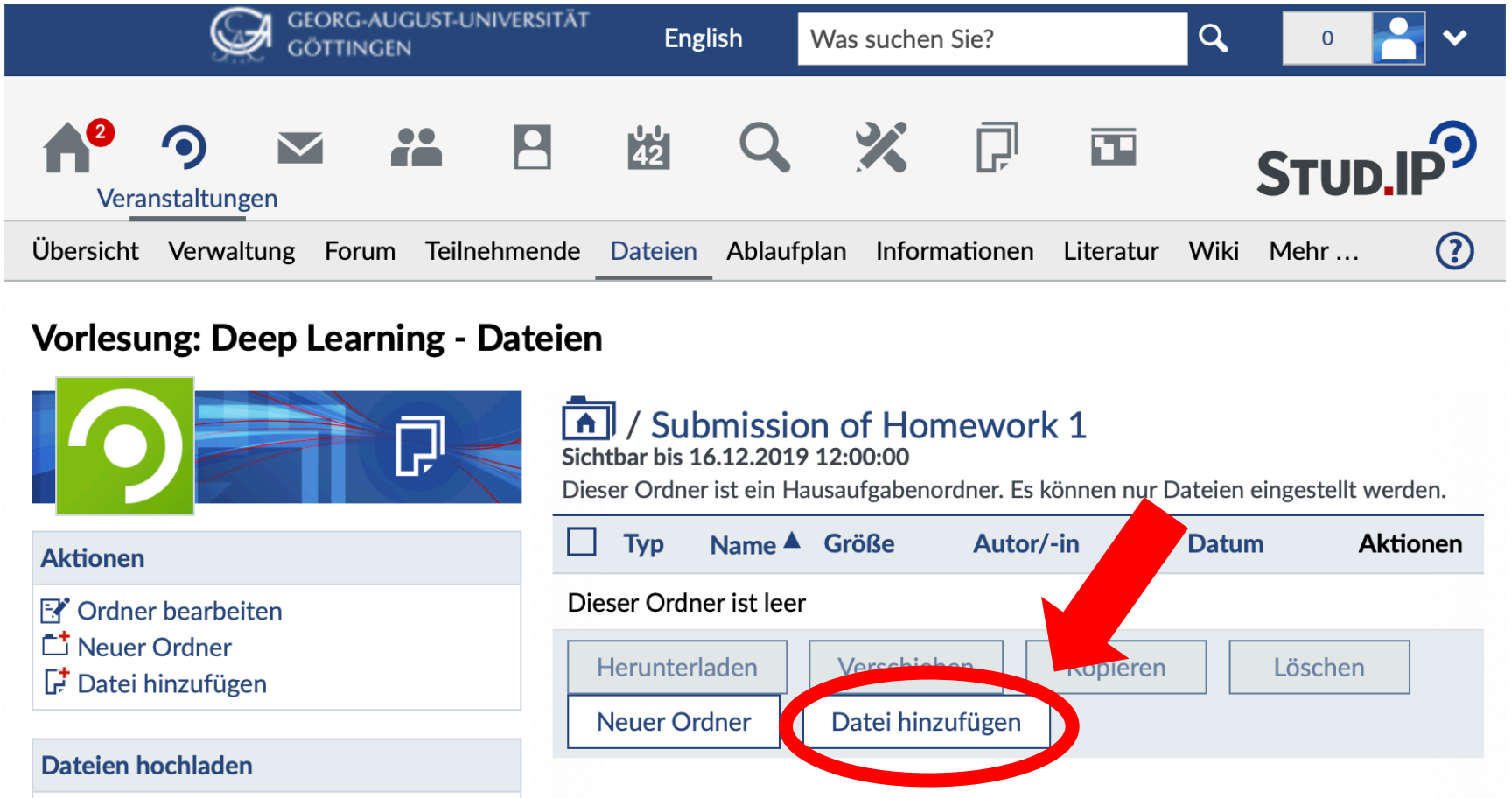
Uploading your solution on Stud.IP (1/4)



Uploading your solution on Stud.IP (1/4)



Uploading your solution on Stud.IP (1/4)



Uploading your solution on Stud.IP (4/4)

Important!

Notebook must be executable from top to bottom without errors

File name: names of the authors, e.g. `ecker_smith_mueller.ipynb`

Upload only one file per team

Topics today

Review of ML basics – case study: polynomial curve fitting

Logistic regression

Gradient descent

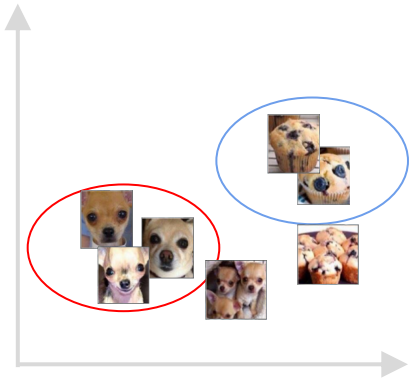
Machine learning

Three main forms of machine learning

Supervised Learning

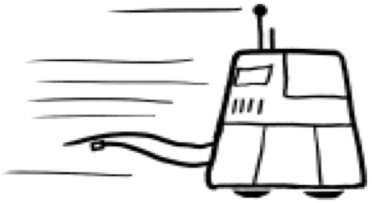
Dog or muffin?

Unsupervised Learning



Model P(**x**)

Reinforcement Learning

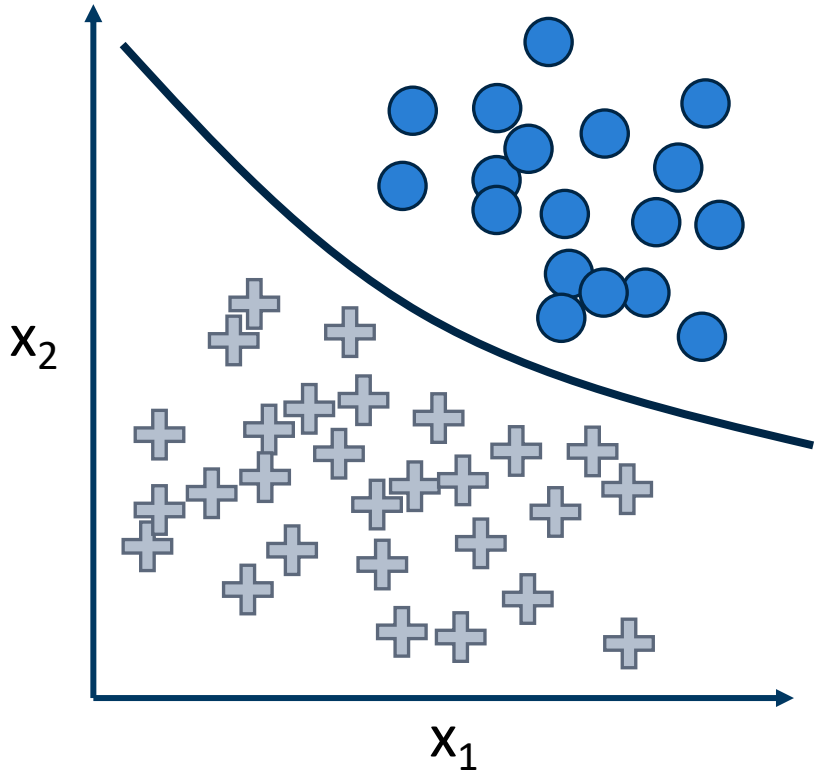


Learn to act

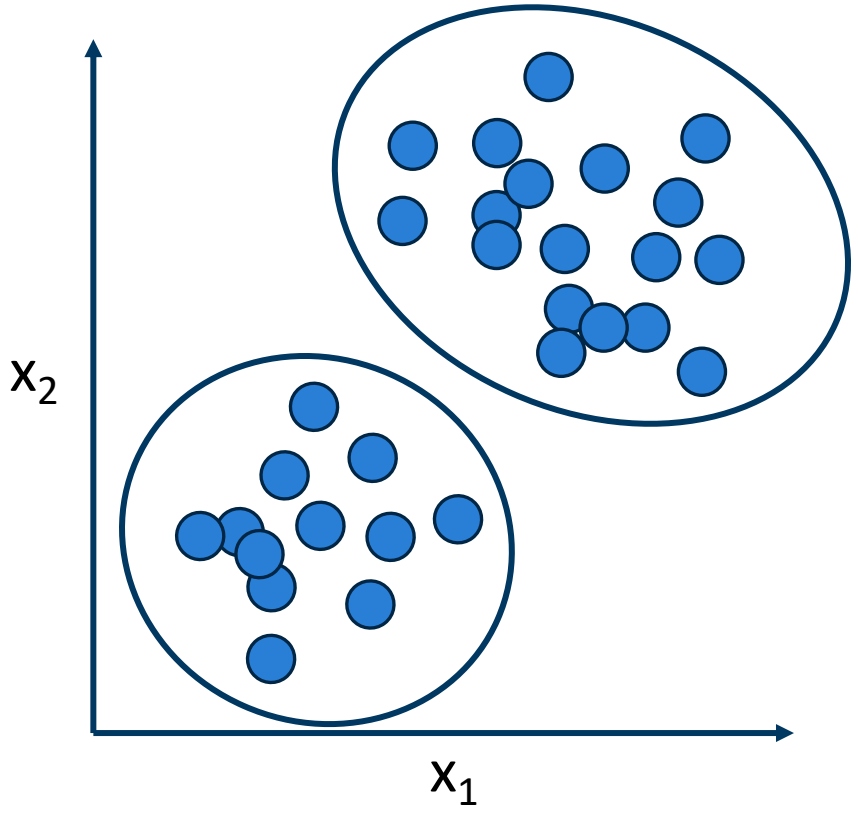
Most significant advances with deep learning

Mostly focus in this lecture

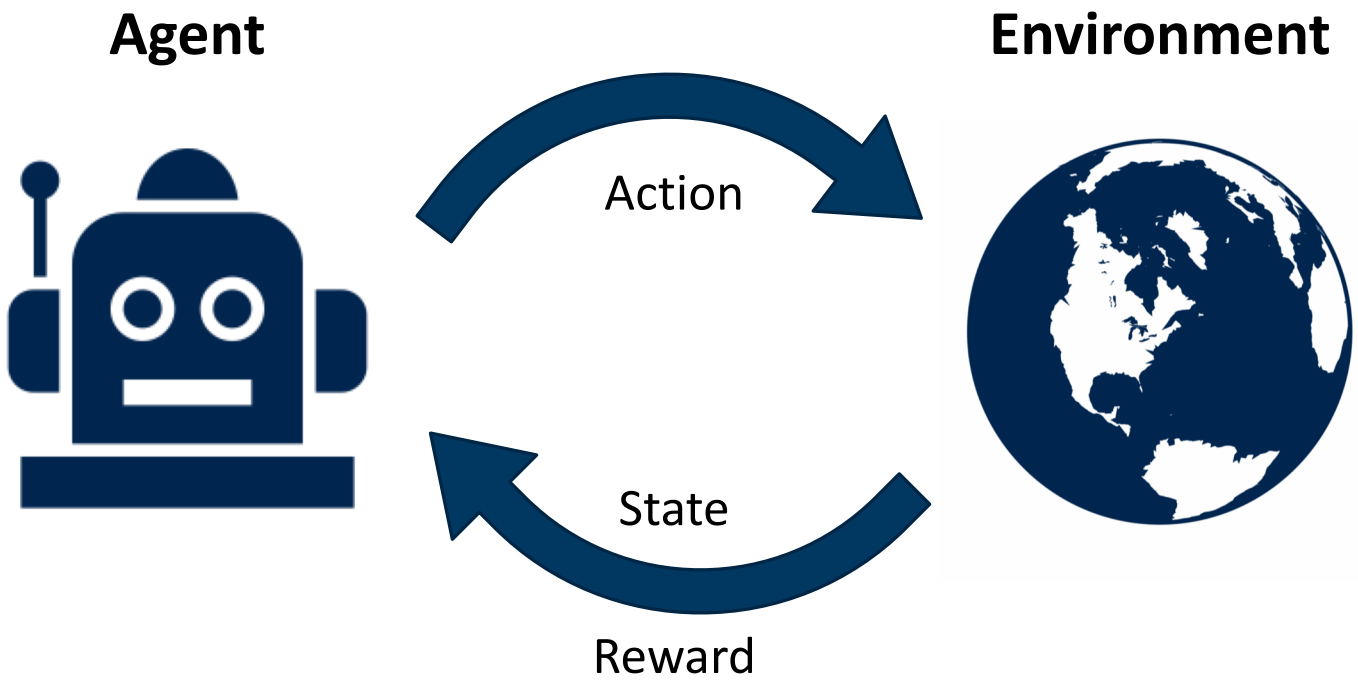
Supervised learning



Unsupervised learning



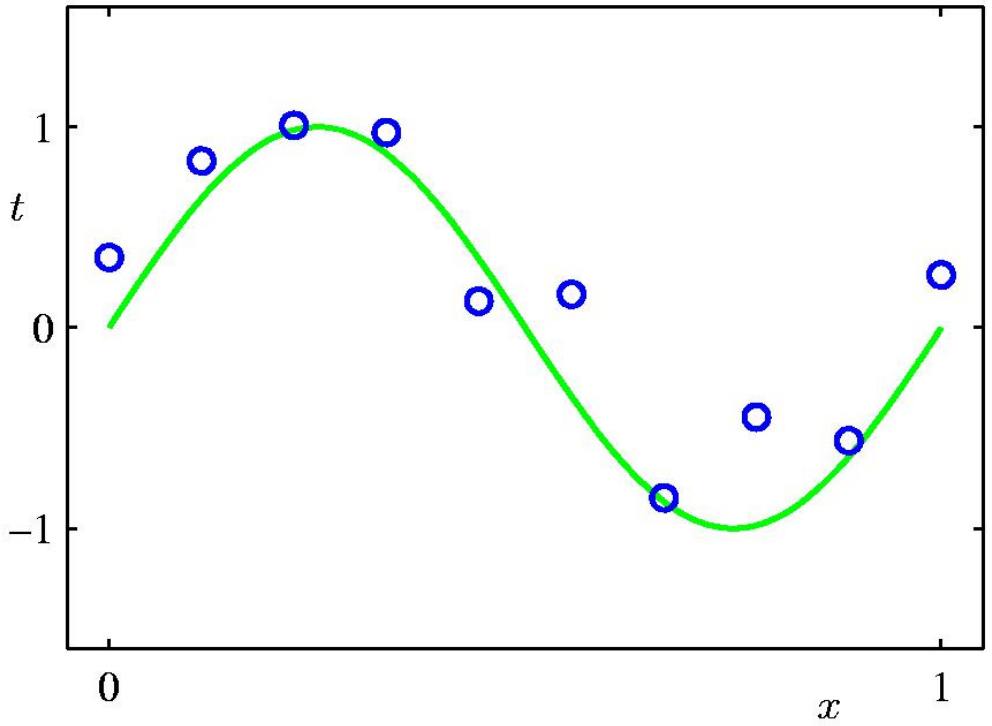
Reinforcement learning



Case study

POLYNOMIAL CURVE FITTING

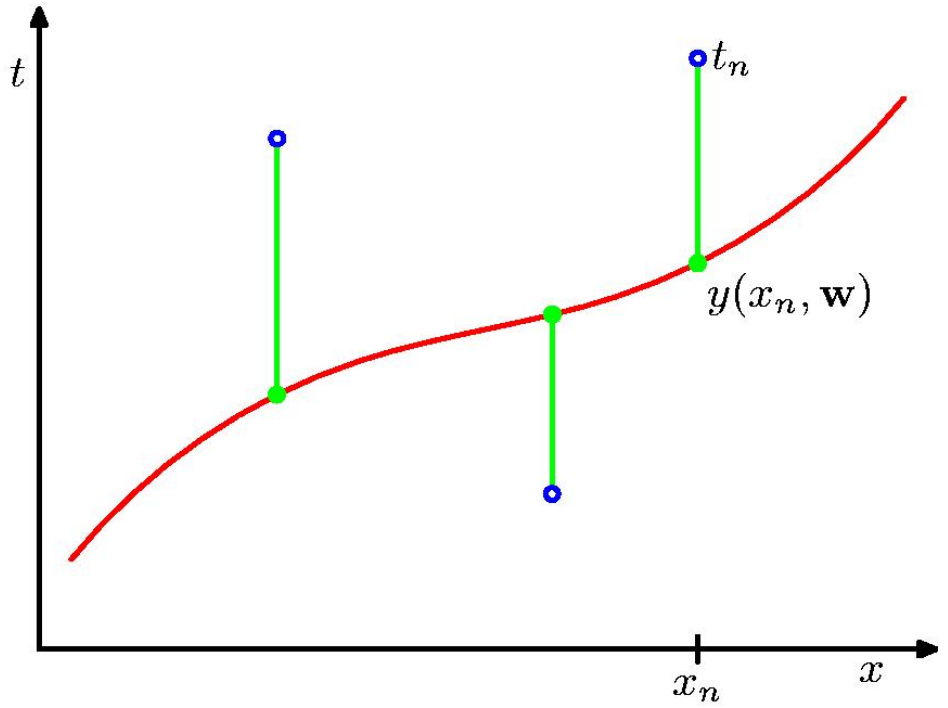
Polynomial Curve Fitting



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

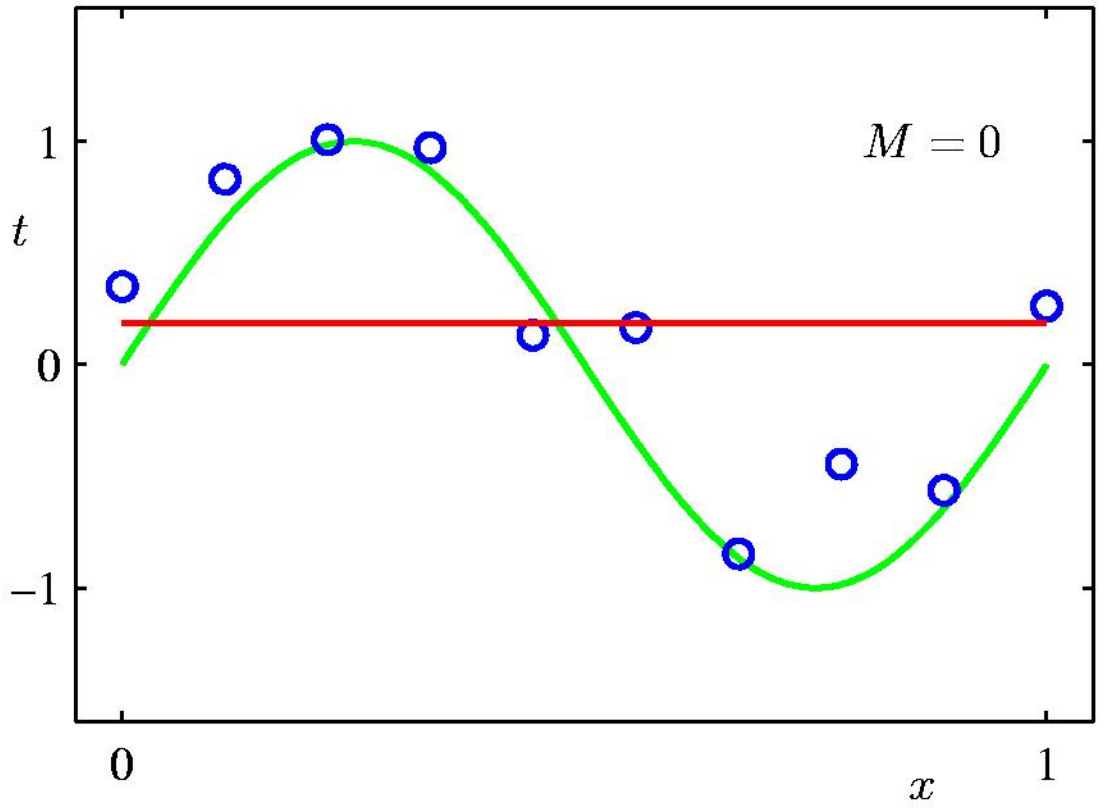
Bishop: Pattern Recognition and Machine Learning

Sum-of-Squares Error Function



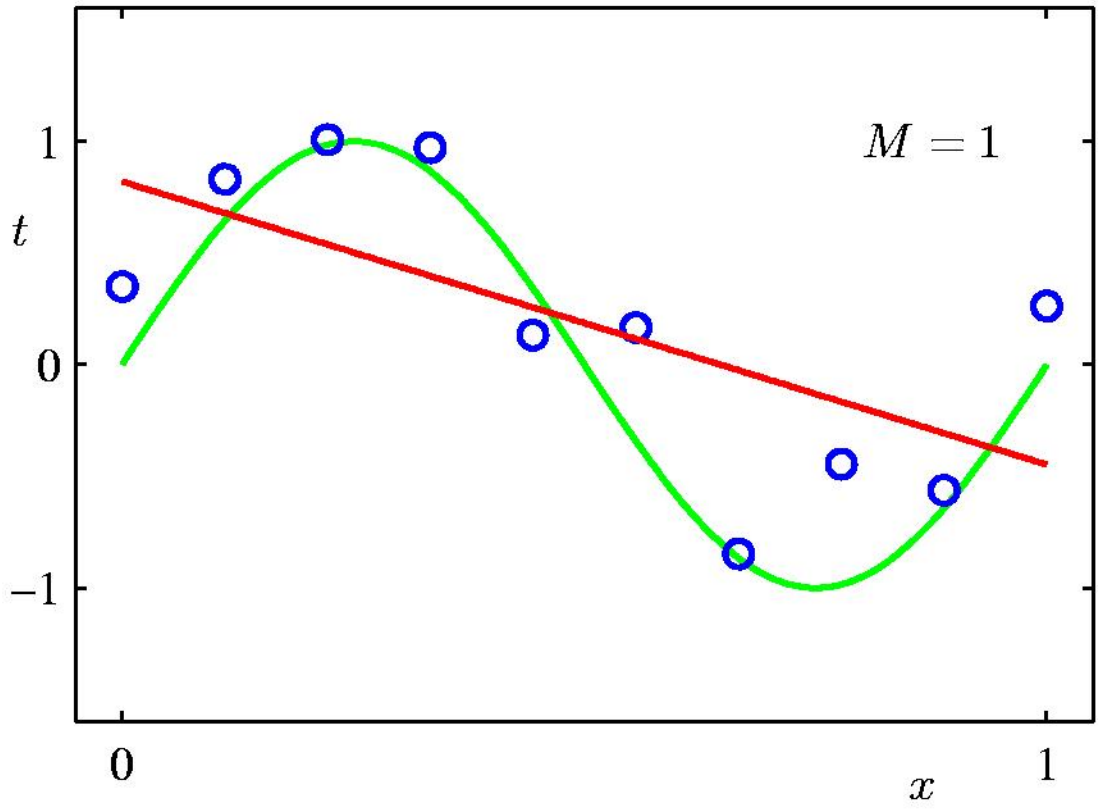
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

0^{th} Order Polynomial



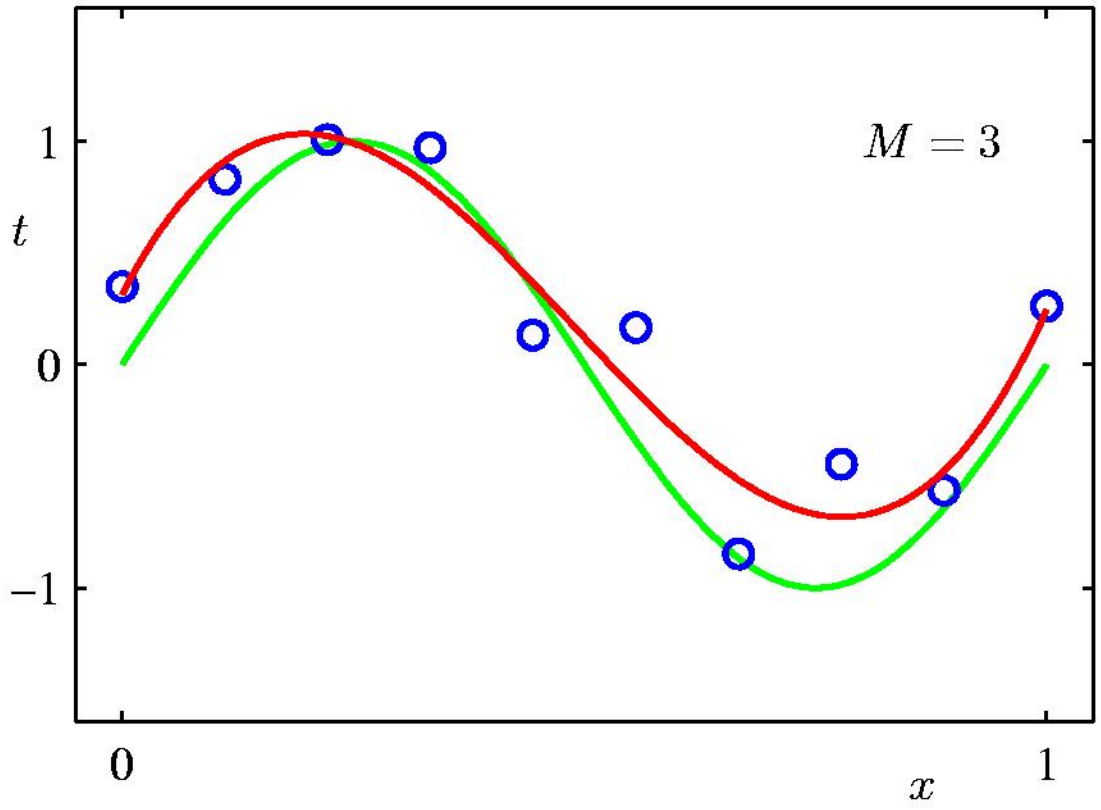
Bishop: Pattern Recognition and Machine Learning

1st Order Polynomial



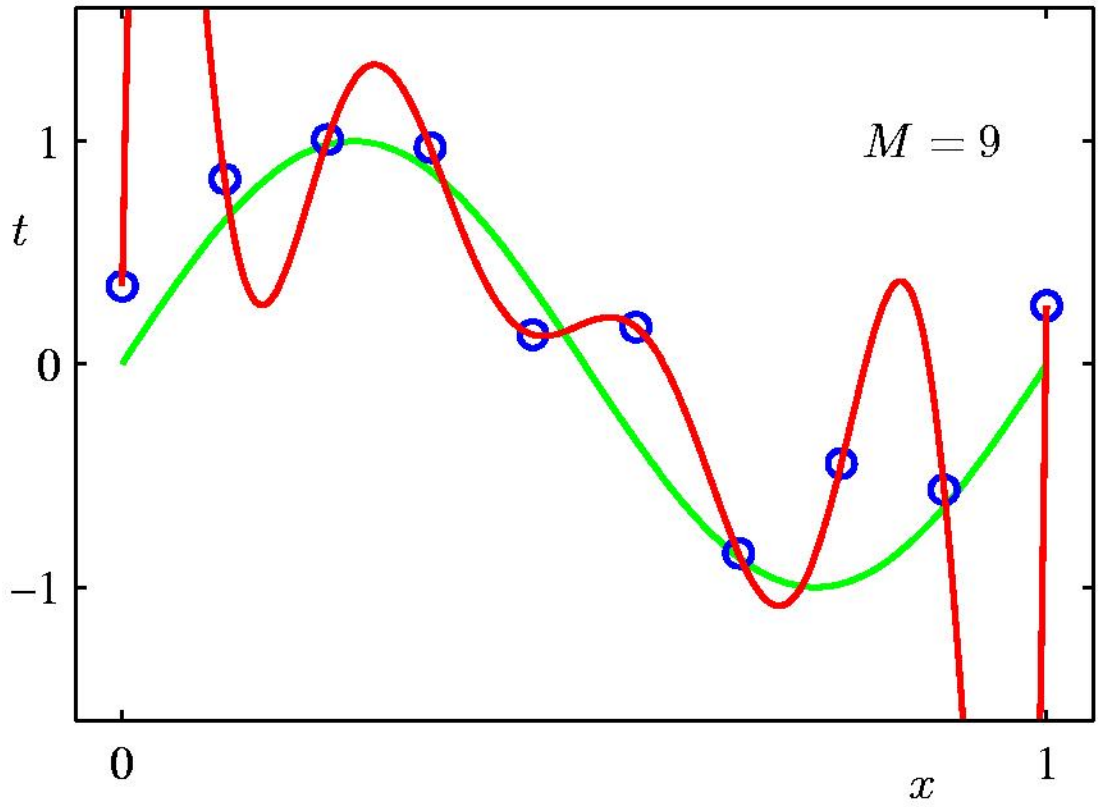
Bishop: Pattern Recognition and Machine Learning

3rd Order Polynomial



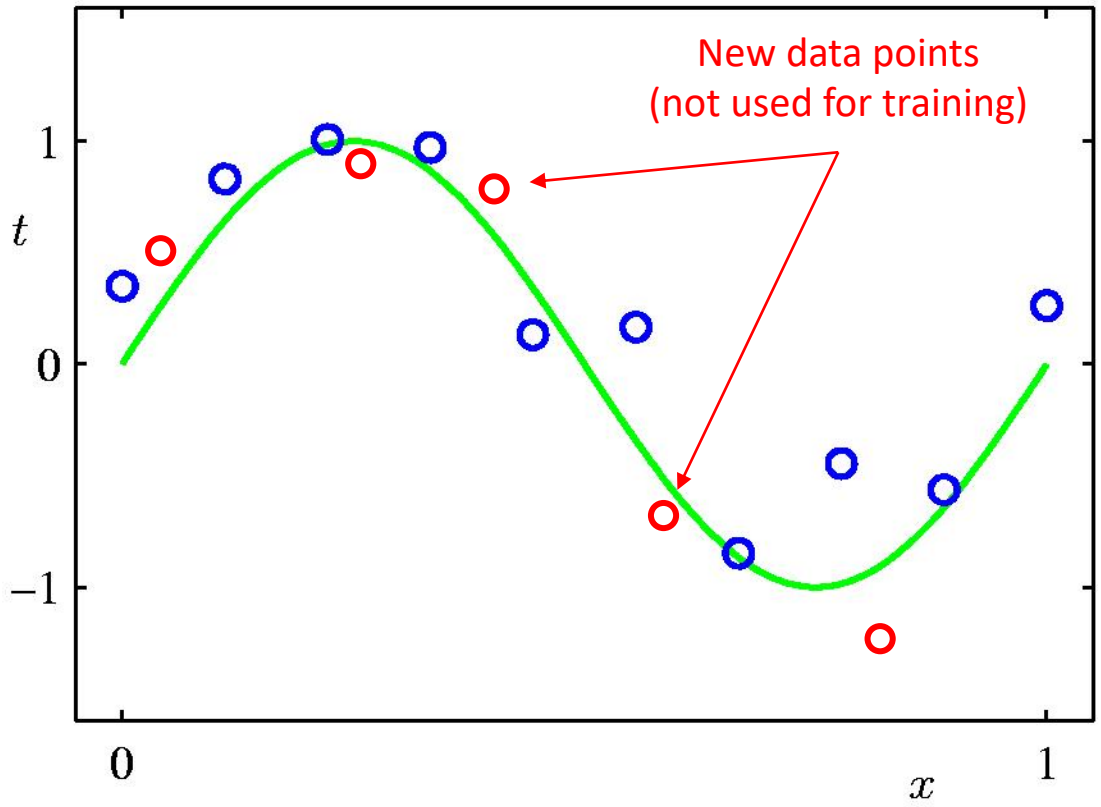
Bishop: Pattern Recognition and Machine Learning

9th Order Polynomial



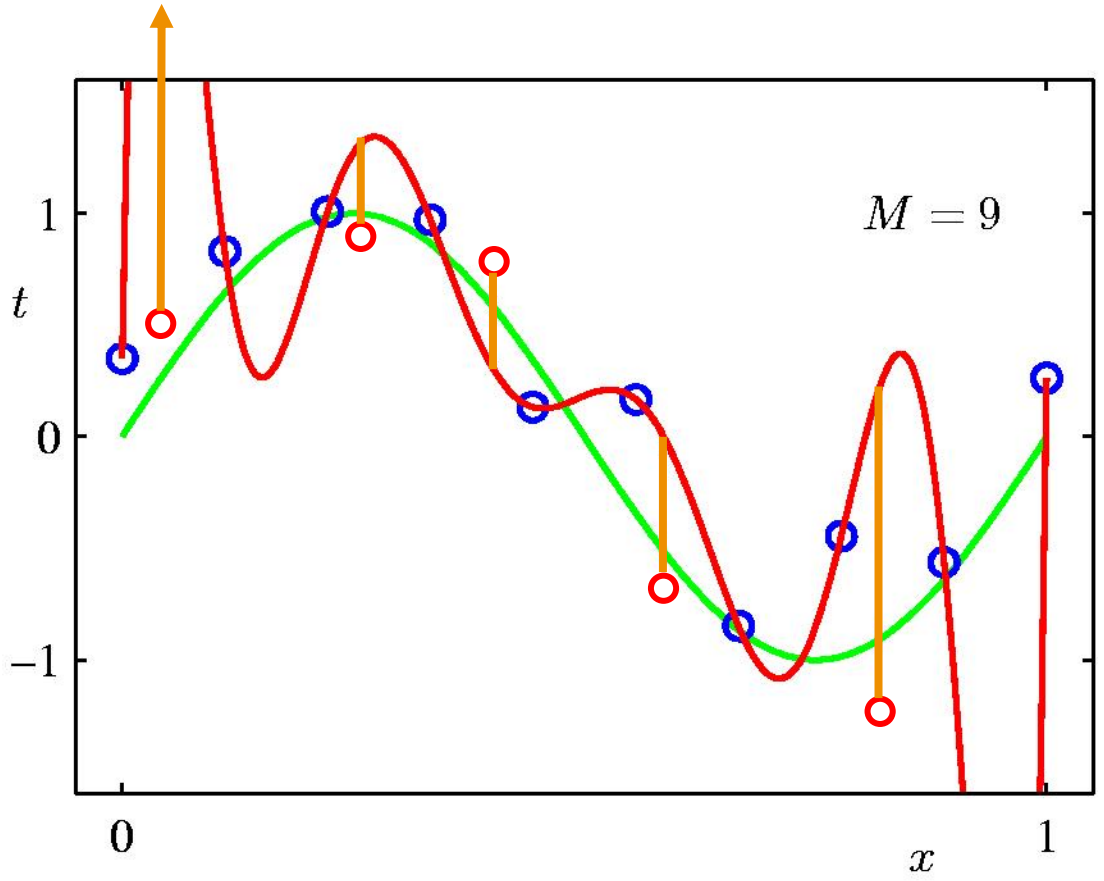
Bishop: Pattern Recognition and Machine Learning

Goal: Generalization to new data



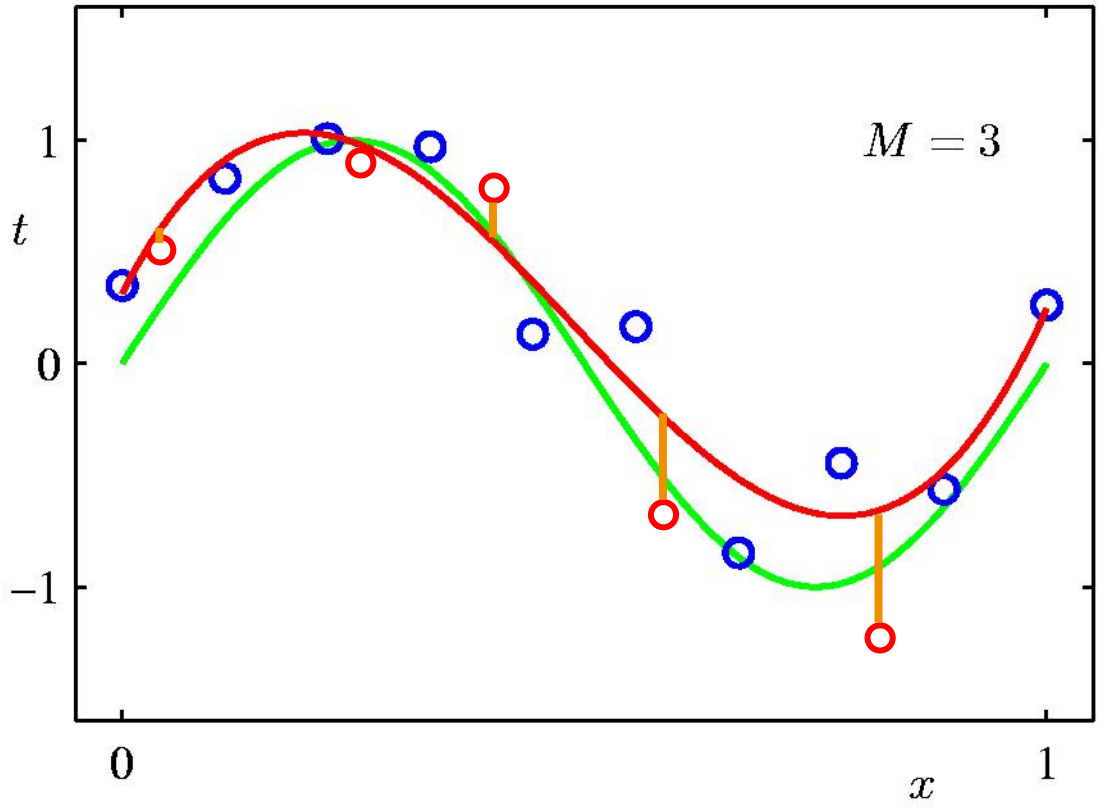
Bishop: Pattern Recognition and Machine Learning

9th Order Polynomial



Training set: perfect fit
Test set: **large errors!**

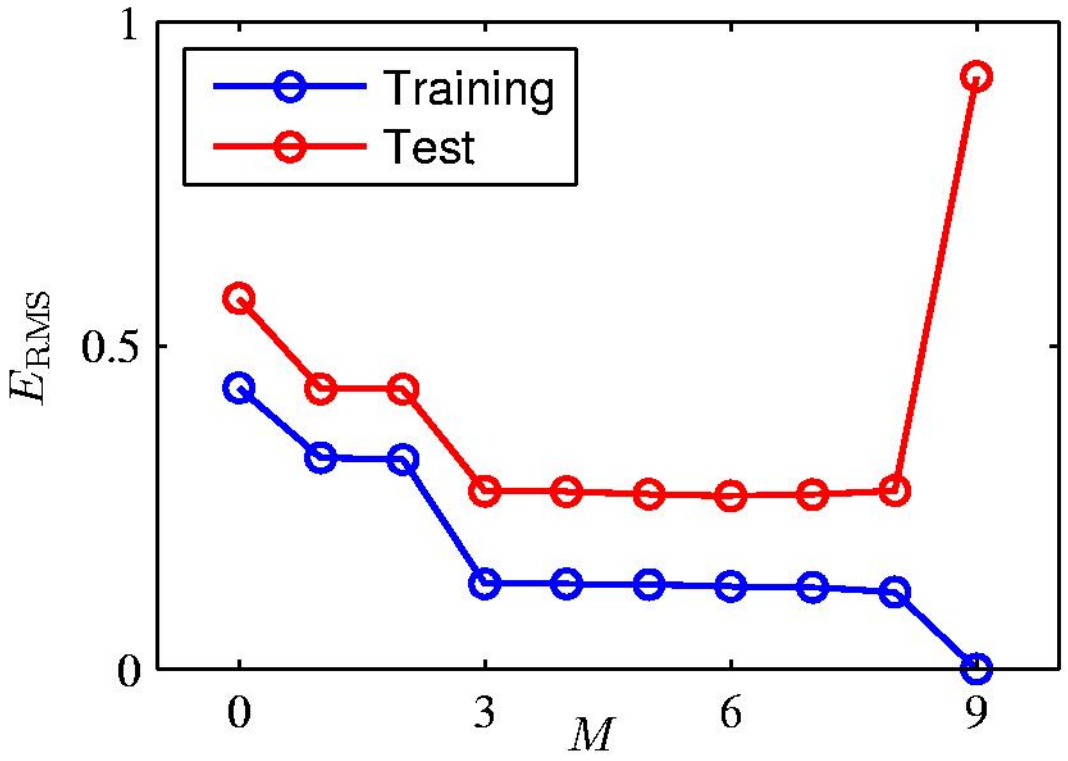
3rd Order Polynomial



Training set: small errors
Test set: **small errors**

Bishop: Pattern Recognition and Machine Learning

Overfitting



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

Goal of ML: risk minimization

Assuming

- **Hypothesis** $h: \mathcal{X} \to \mathcal{Y}$ from a hypothesis space $\mathcal{H}$
- **Prediction** $\hat{y} = h(x)$
- **Training samples** (x_n, y_n) drawn independently and identically from joint probability density $P(\boldsymbol{x}, y)$
- **Loss function** $L(\hat{y}, y): \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$
- **Risk** R: $\mathcal{H} \mapsto \mathbb{R}$, the expected loss $R(h) = \mathbb{E}[L(h(x), y)]$

Goal of supervised learning:

Risk minimization:

$$\arg\min_{h \in \mathcal{H}} R(h)$$

Empirical risk minimization

In practice, risk $R(h)$ cannot be computed!

The distribution $P(x, y)$ is unknown to the learning algorithm

Typical solution:

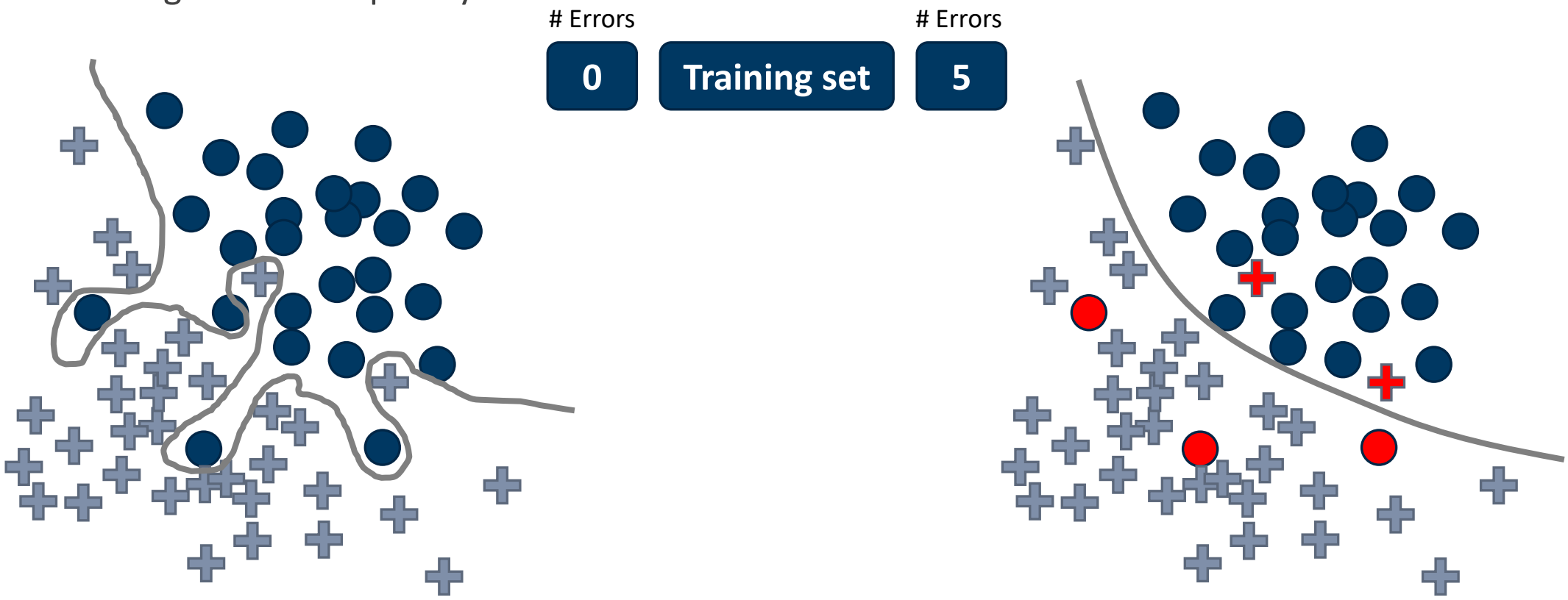
Empirical risk minimization (ERM):

$$R_{\text{emp}}(h) = \frac{1}{N} \sum_{n=1}^{N} L(h(x_n), y_n)$$

$$\hat{h} = \arg\min_{\text{h} \in \mathcal{H}} R_{\text{emp}}(h)$$

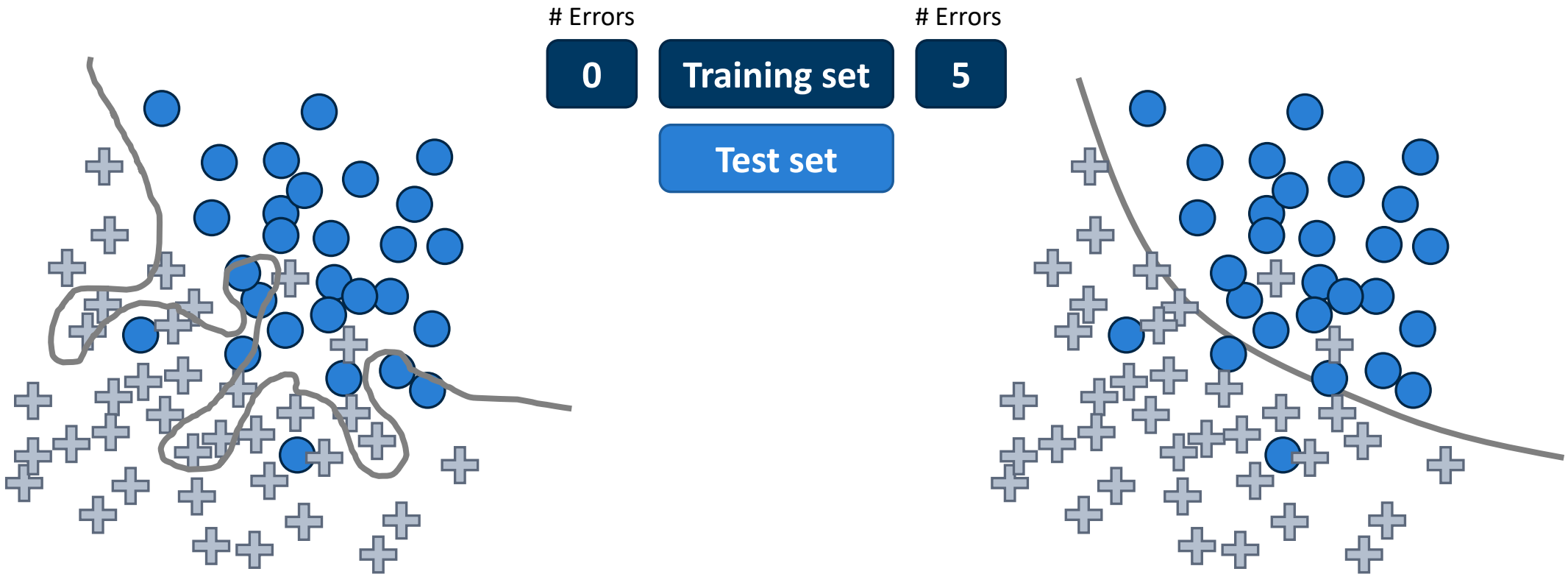
Problem with Empirical Risk Minimization

If hypothesis space $\mathcal{H}$ is not constrained, empirical risk R can be made arbitrarily small by increasing model complexity



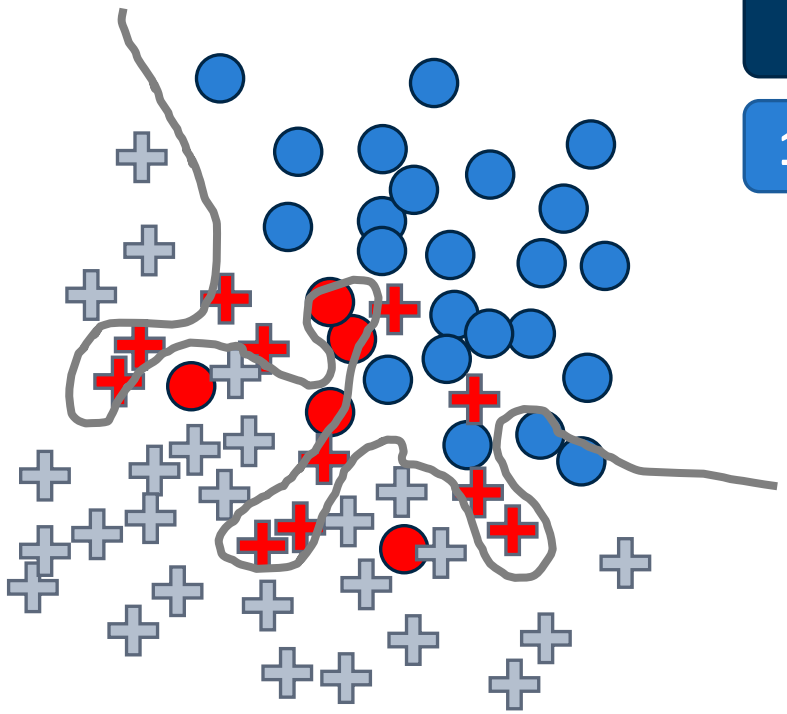
Problem with Empirical Risk Minimization

What happens when we deploy our classifier and test it on new data?

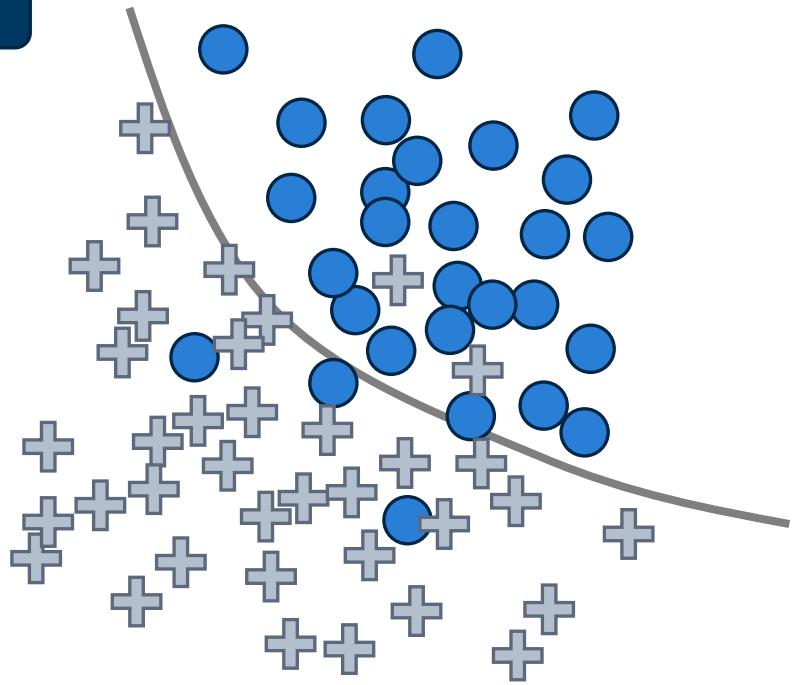


Problem with Empirical Risk Minimization

Does not generalize!

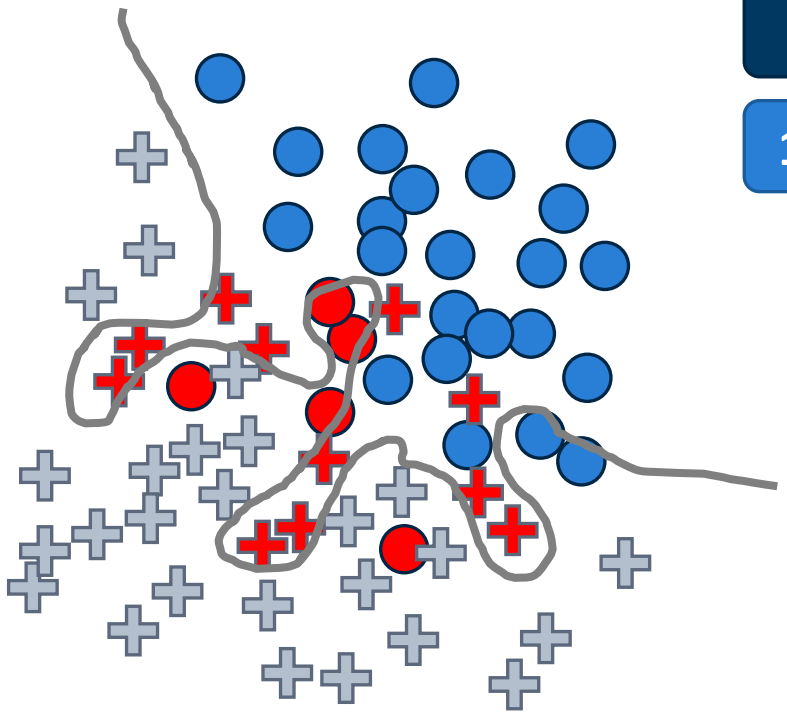


# Errors		# Errors
0	Training set	5
16	Test set	



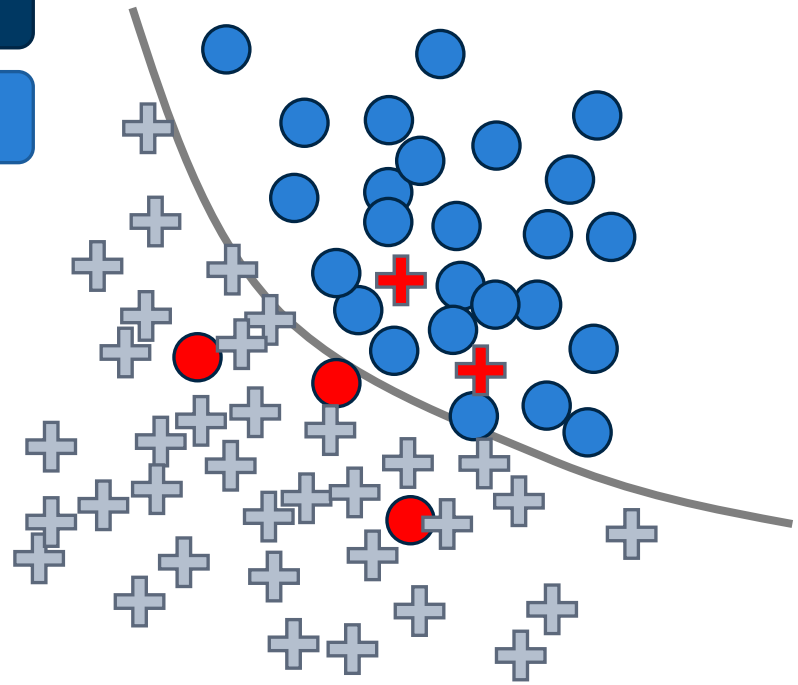
Problem with Empirical Risk Minimization

Does not generalize!



# Errors		# Errors
0	Training set	5
16	Test set	5

Generalizes well



What does this mean in practice?

Goal: Generalization

Machine learning is about generalization, not about optimization

CONTROL MODEL COMPLEXITY

A number of techniques exist:

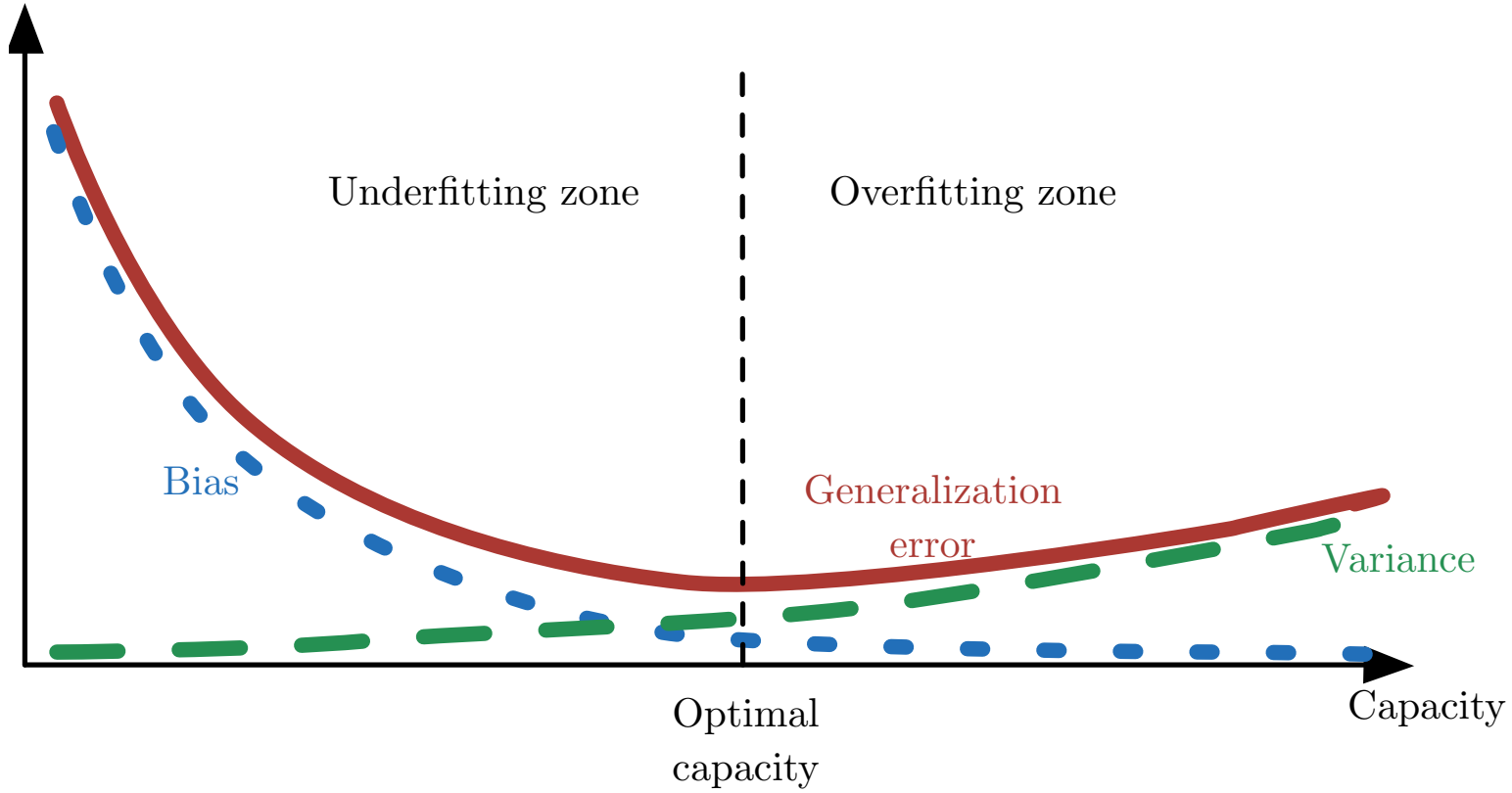
- Reduce number of parameters
- Regularization
- Early stopping

ESTIMATE GENERALIZATION ERROR

Split dataset into

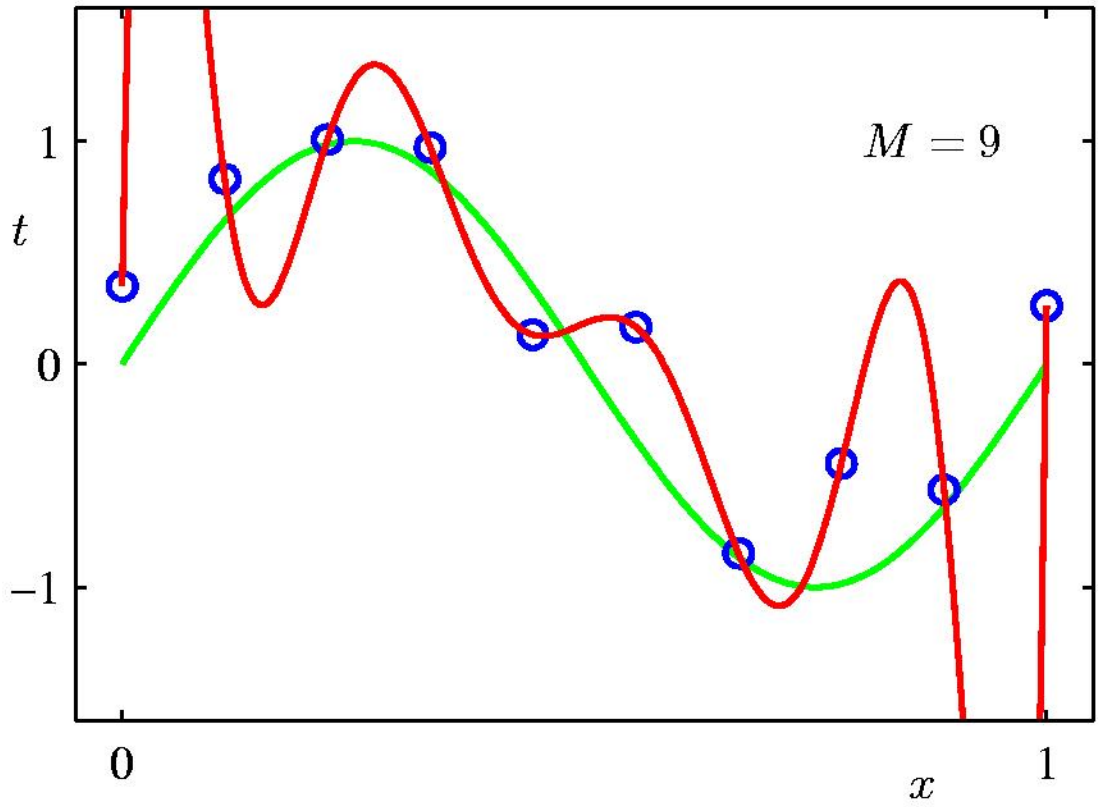
- **Training set** used to train the algorithm using empirical risk minimization (ERM)
- **Test set** used to estimate the true risk

Bias and variance



Data Set Size: $N = 10$

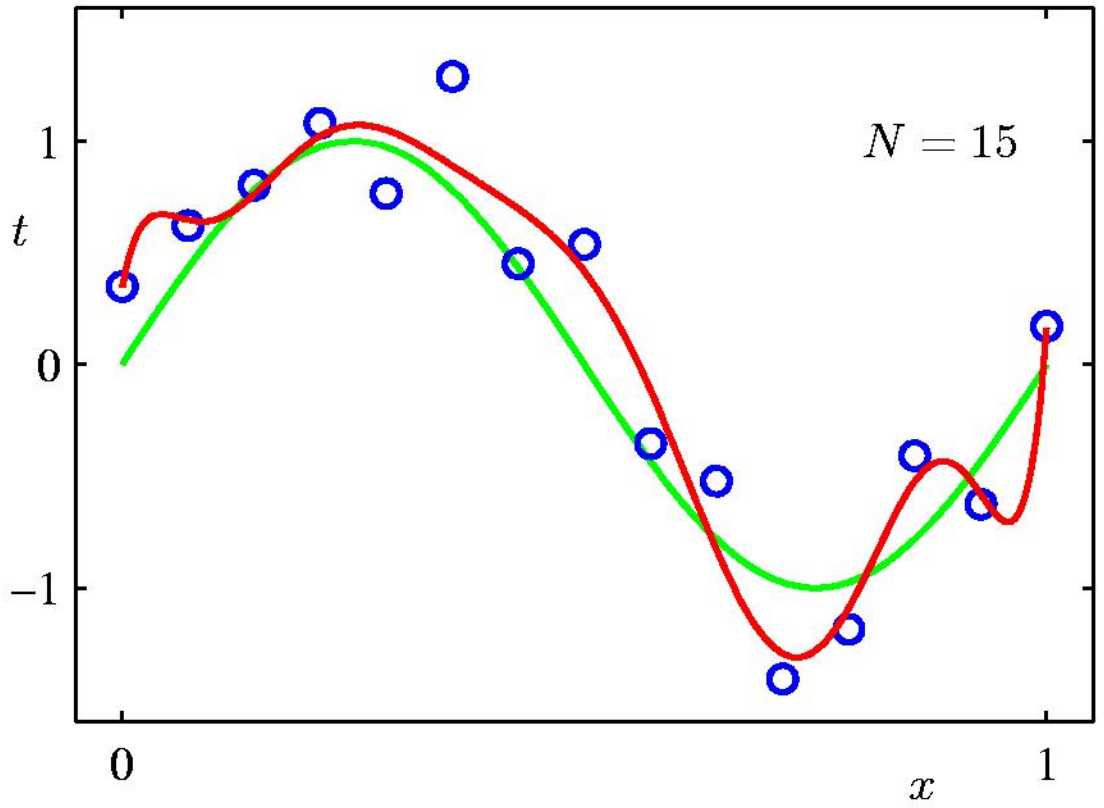
9th Order Polynomial



Bishop: Pattern Recognition and Machine Learning

Data Set Size: $N = 15$

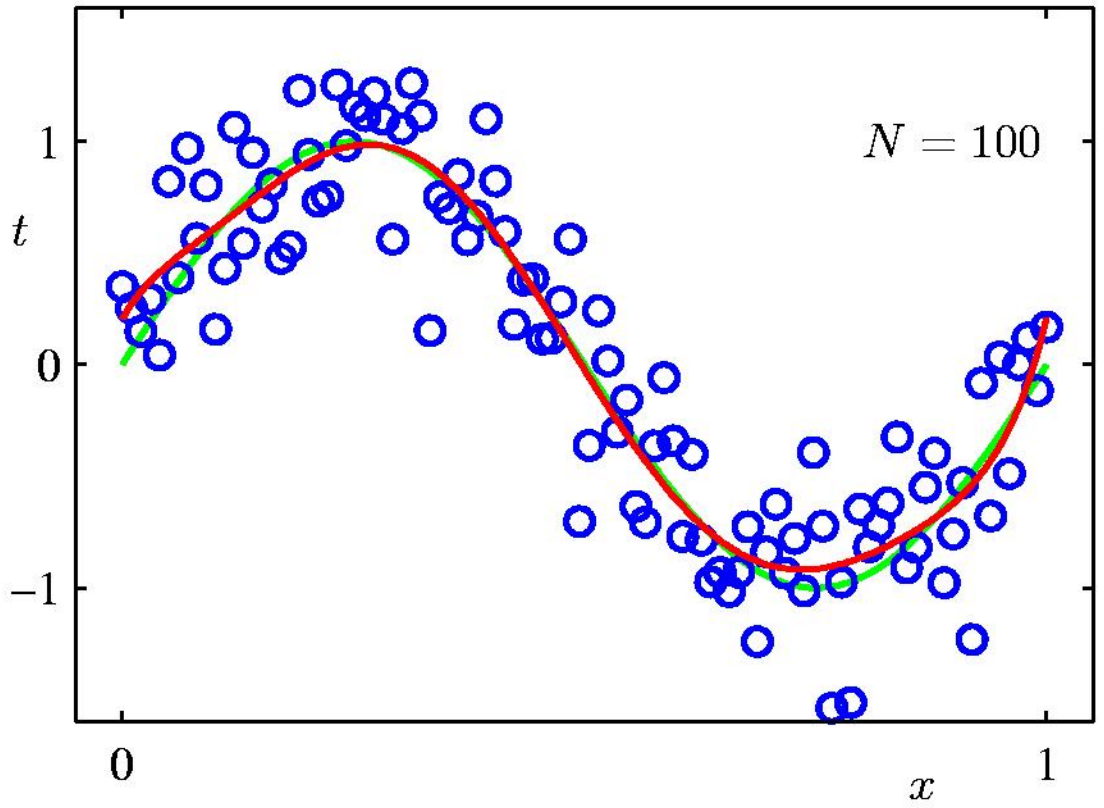
9th Order Polynomial



Bishop: Pattern Recognition and Machine Learning

Data Set Size: $N = 100$

9th Order Polynomial



Bishop: Pattern Recognition and Machine Learning

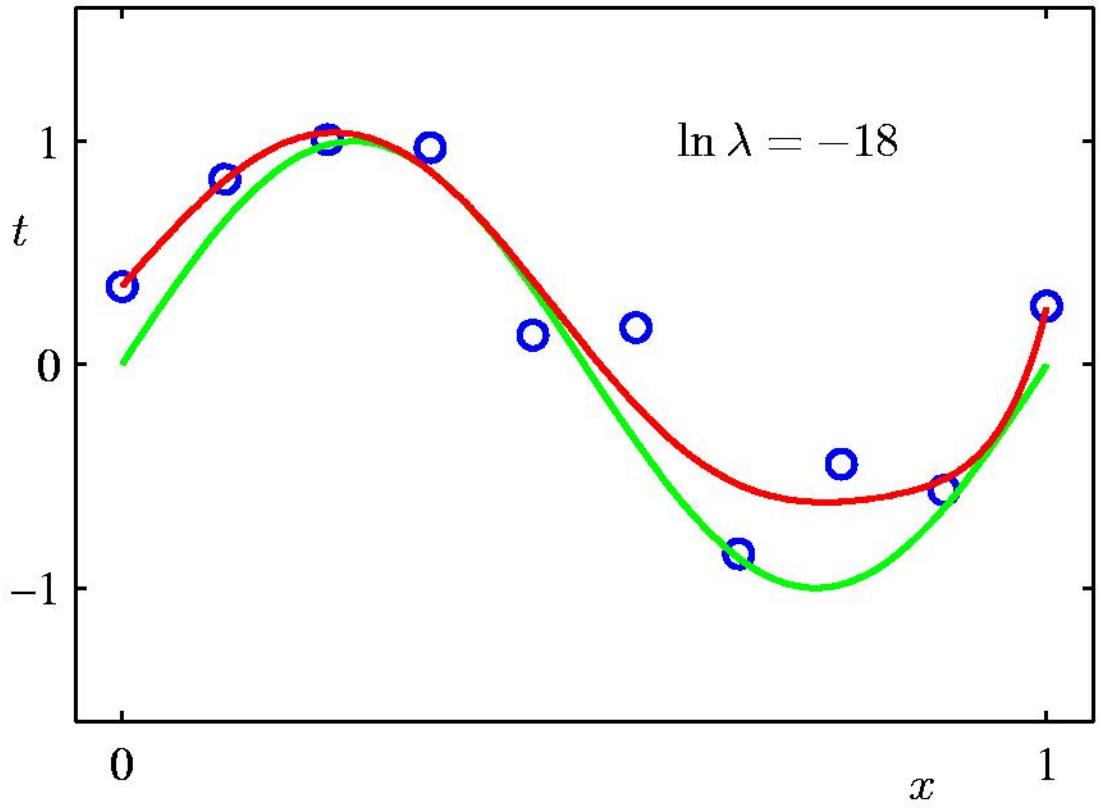
Regularization

Regularization

Penalize large coefficient values

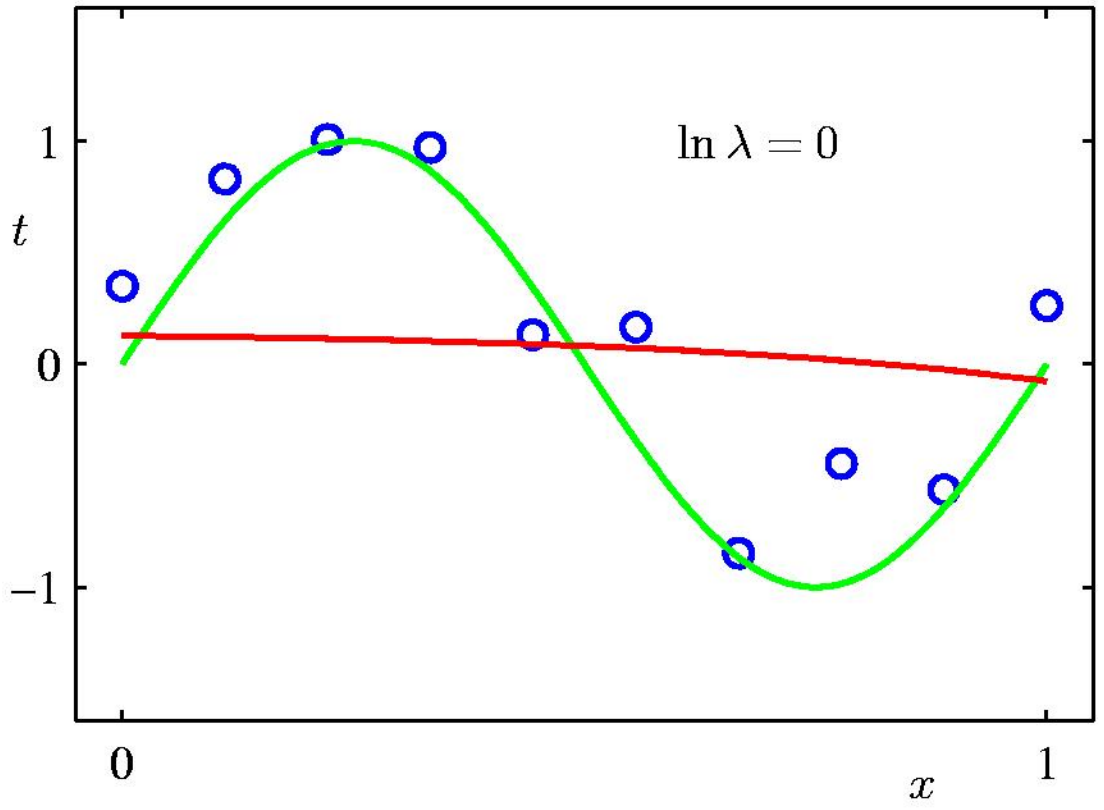
$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Moderate Regularization



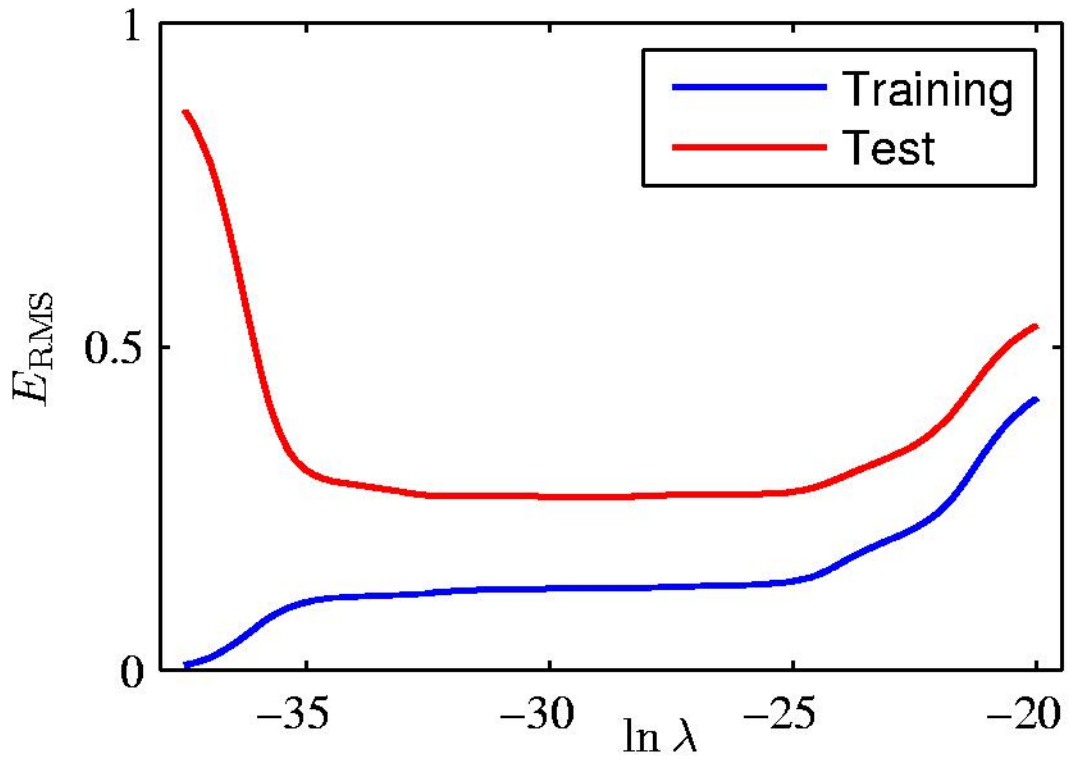
Bishop: Pattern Recognition and Machine Learning

Regularization too strong



Bishop: Pattern Recognition and Machine Learning

Squared error vs. regularization strength

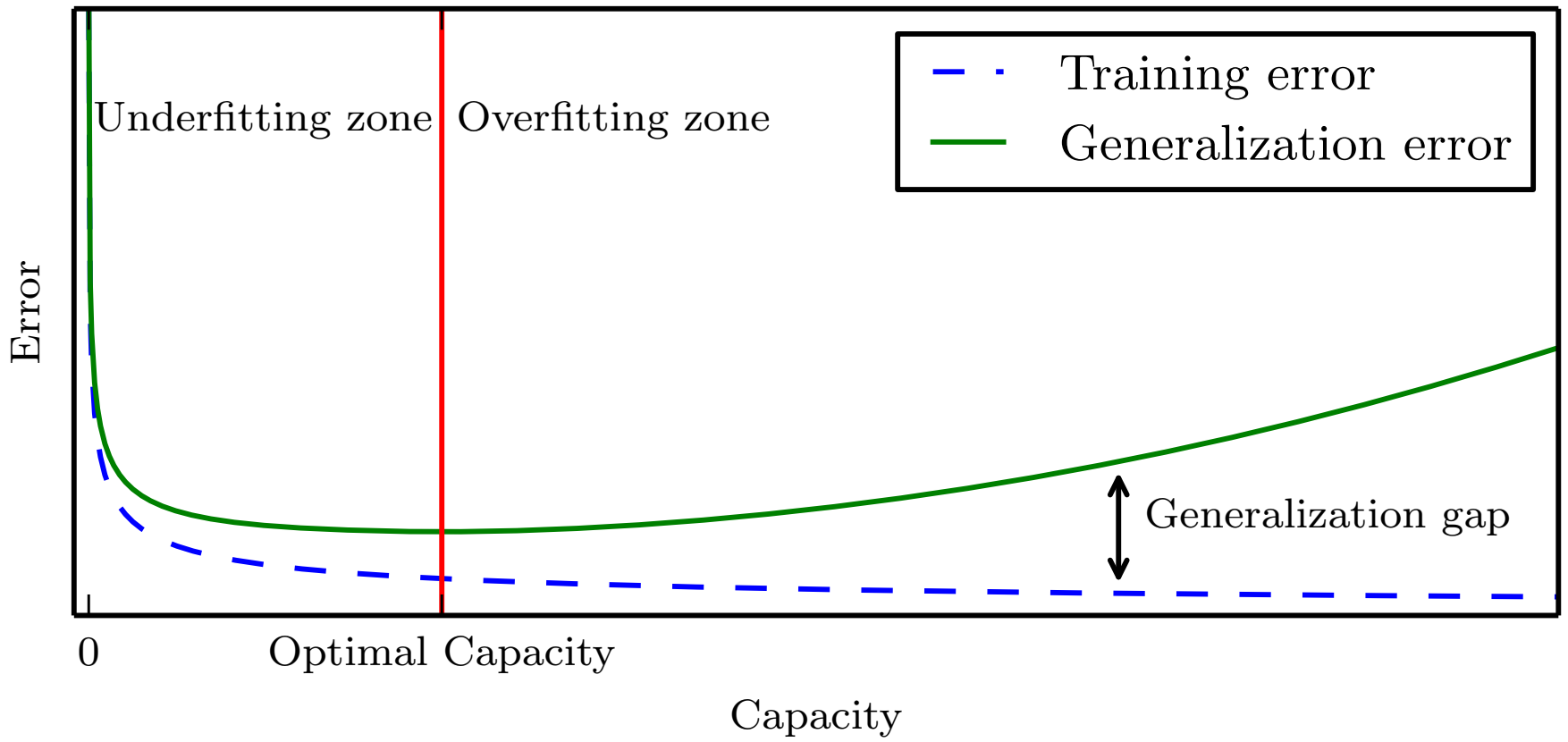


Bishop: Pattern Recognition and Machine Learning

Polynomial Coefficients

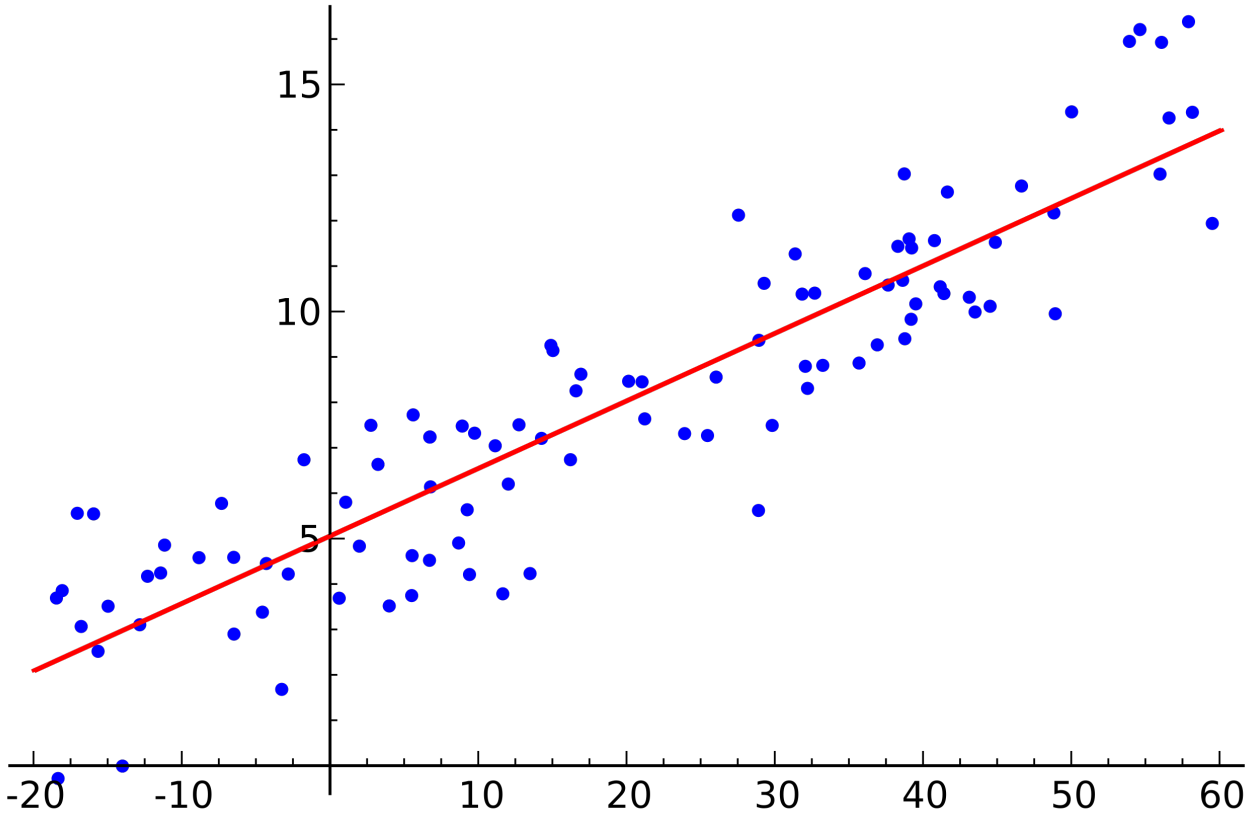
	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$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

Generalization and capacity



Linear regression

Linear regression



$$y = ax + b$$
$$y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$

$$\boldsymbol{w} = [a, b]^{\mathrm{T}}$$
$$\boldsymbol{x} = [x, 1]^{\mathrm{T}}$$

Maximum Likelihood and Least Squares (1)

Assume observations from a deterministic function $t = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + \epsilon$ with added Gaussian noise:

$$p(\epsilon|\beta) = \mathcal{N}(\epsilon|0, \beta^{-1})$$

which is the same as saying,

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

Given observed inputs, $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$, and targets, $\boldsymbol{t} = [t_1, \ldots, t_N]$, we obtain the likelihood function

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

Maximum Likelihood and Least Squares (2)

Taking the logarithm, we get

$$\log p(t|\boldsymbol{X}, \boldsymbol{w}, \beta) = \sum_{n=1}^{N} \mathcal{N}\left(t_n | \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n, \beta^{-1}\right)$$
$$= \frac{N}{2} \log \beta - \frac{N}{2} \log 2\pi - \beta L(\boldsymbol{w})$$

where

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

is the sum-of-squares error.

Maximum Likelihood and Least Squares (3)

Computing the gradient and setting it to zero yields

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

Solving for $\boldsymbol{w}$, we get

$$\boldsymbol{w}_{\mathrm{ML}} = \left(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^{\mathrm{T}} \boldsymbol{t}$$

The Moore-Penrose pseudo-inverse, $\boldsymbol{\Phi}^{\dagger}$.

Probabilistic view on regularization

Bayles rule

$$P(\boldsymbol{w}|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\boldsymbol{w})P(\boldsymbol{w})}{P(\boldsymbol{x})}$$

Maximum a-posteriori (MAP) estimate of $\boldsymbol{w}$:

$$\arg\max_{\boldsymbol{w}} P(\boldsymbol{w}|\boldsymbol{x})$$

$$P(\boldsymbol{w}|\boldsymbol{x}) \propto P(\boldsymbol{x}|\boldsymbol{w})P(\boldsymbol{w})$$

Linear regression with Gaussian prior on weights

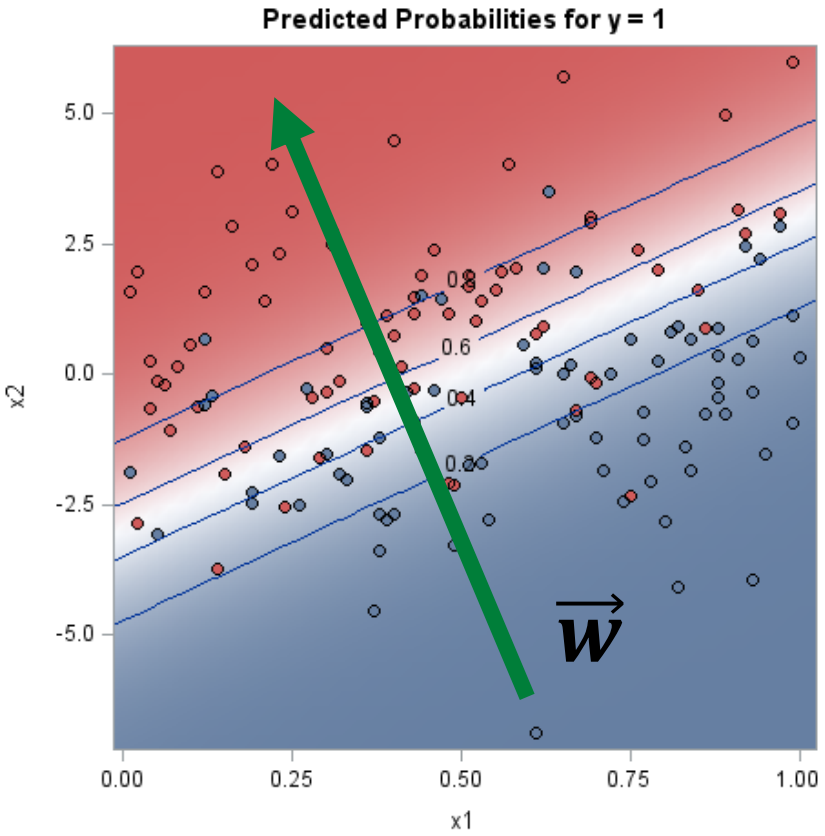
Suppose $P(\boldsymbol{w}) = \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$

Find MAP by minimizing $L(\boldsymbol{w}) - \log P(\boldsymbol{w}|\boldsymbol{x})$

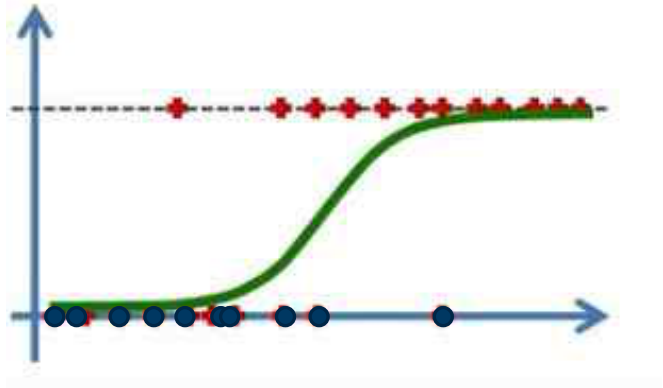
$$
\begin{aligned}
-\log P(\boldsymbol{w}|\boldsymbol{x}) &= -\log P(\boldsymbol{x}|\boldsymbol{w}) - \log \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}) \\
&= -\log P(\boldsymbol{x}|\boldsymbol{w}) - \log\left(\exp\left(-\frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}}{2\sigma^2}\right)\right) \\
&= -\frac{1}{2}\sum_{n=1}^{N}\{t_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n\}^2 - \frac{1}{2\sigma^2}\|\boldsymbol{w}\|_2^2
\end{aligned}
$$

Logistic regression

Logistic regression

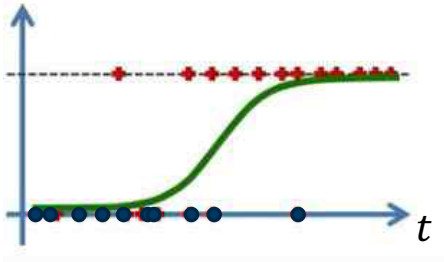


Project onto $\vec{w}$



Logistic regression model

Sigmoid
(logistic function)



$$f(t) = \frac{1}{1 + \exp(-t)}$$

Model the probability of **'on'** $(y = 1)$ as

$$P(y = 1|\boldsymbol{x}, \boldsymbol{w}) := \sigma(\boldsymbol{w}^T \boldsymbol{x}) := \frac{1}{1 + \exp(-\boldsymbol{w}^T \boldsymbol{x})} := p$$

Probability of **'off'** $(y = 0)$:

$$P(y = 0|\boldsymbol{x}, \boldsymbol{w}) = 1 - p = 1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}) = \sigma(-\boldsymbol{w}^T \boldsymbol{x})$$

Loss function (negative log-likelihood):

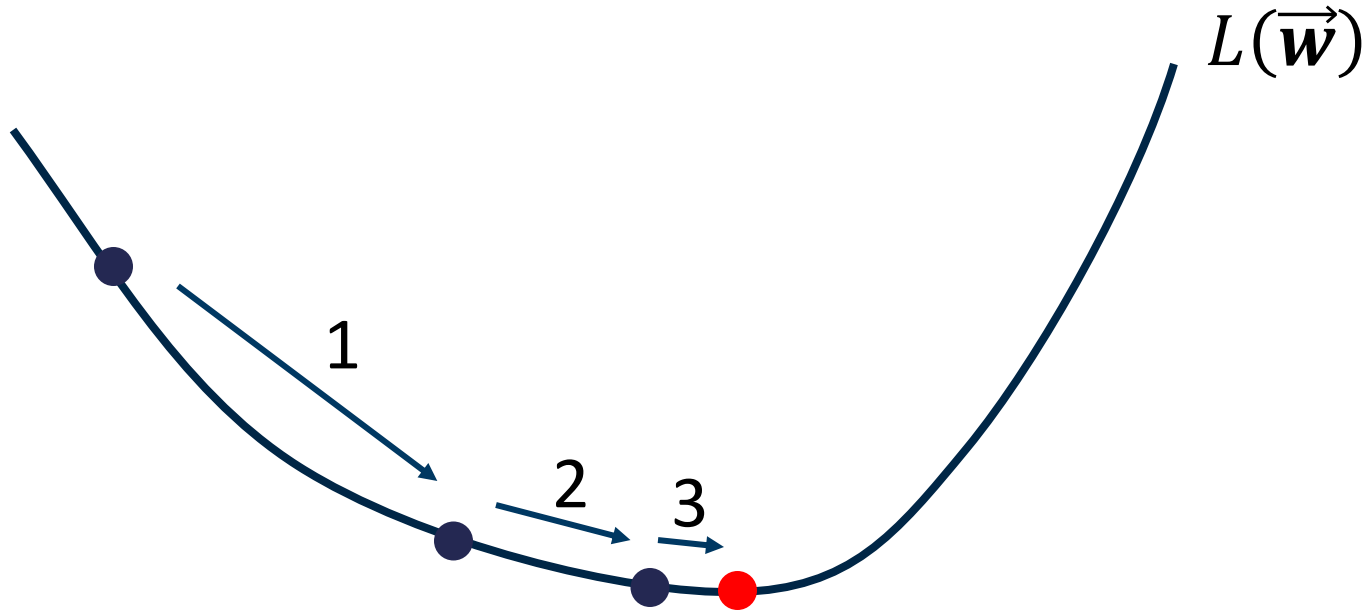
$$P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w}) = \prod_n p_n^{y_n} (1 - p_n)^{1 - y_n}$$

$$L = -\log P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w}) = -\sum_n y_n \log \sigma(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n) + (1 - y_n) \log \sigma(-\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n)$$

Solving logistic regression

Optimal solution ca be found via *iterative reweighted least squares*

Convex loss function → any gradient-based optimizer will find optimum



Gradient descent

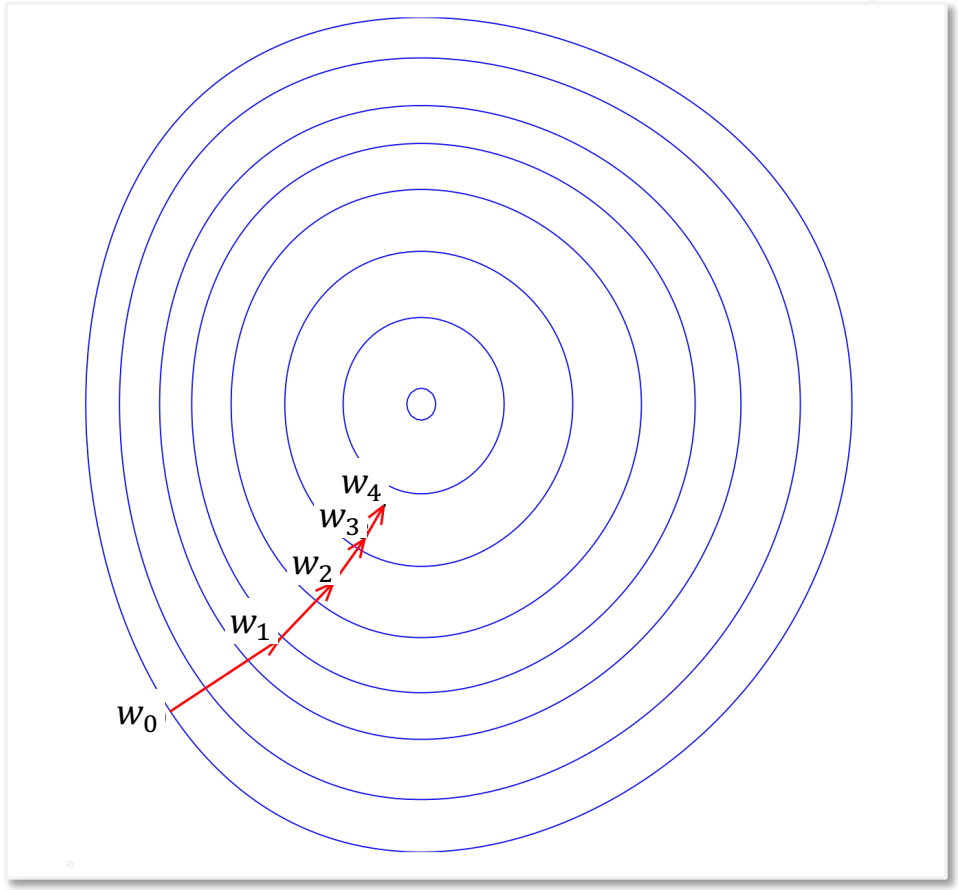
We use gradient descent to minimize the loss

Start with a random guess for the parameters $\boldsymbol{w}$

Iteratively update w by a small step in the direction of the gradient of the loss function

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \lambda \nabla_w L(\boldsymbol{w}_t)$$

Learning rate

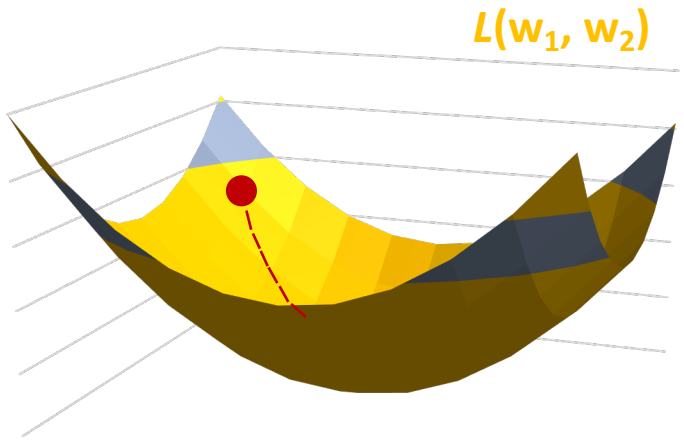


Source: Wikipedia

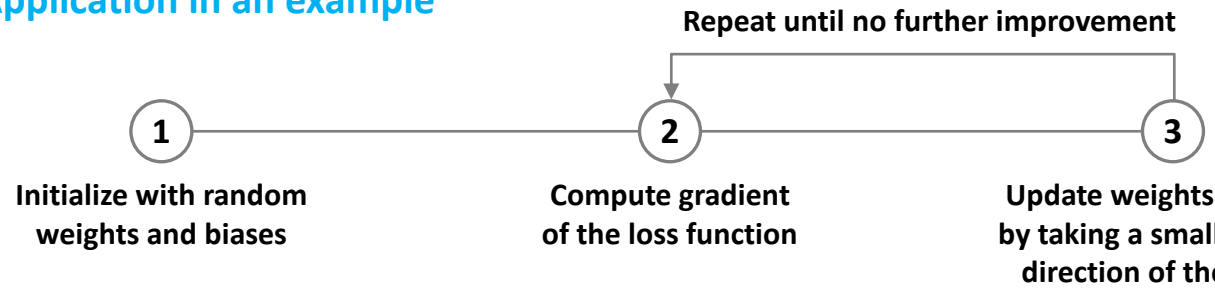
The three steps of gradient descent

Intuition

Iteratively move along the direction of steepest descent.



Application in an example



$$\boldsymbol{w} = \begin{bmatrix} 4{,}8 \\ -2{,}3 \\ 0{,}7 \\ \vdots \\ -0{,}2 \\ 1{,}5 \end{bmatrix} + -\nabla L(\boldsymbol{w}) = \begin{bmatrix} -0{,}7 \\ 1{,}2 \\ 5{,}3 \\ \vdots \\ -1{,}7 \\ -0{,}4 \end{bmatrix} = \boldsymbol{w} = \begin{bmatrix} 4{,}1 \\ -1{,}1 \\ 6{,}0 \\ \vdots \\ -1{,}9 \\ 1{,}1 \end{bmatrix}$$

Questions?