COMPUTATIONAL INTELLIGENCE

(INTRODUCTION TO MACHINE LEARNING) SS17

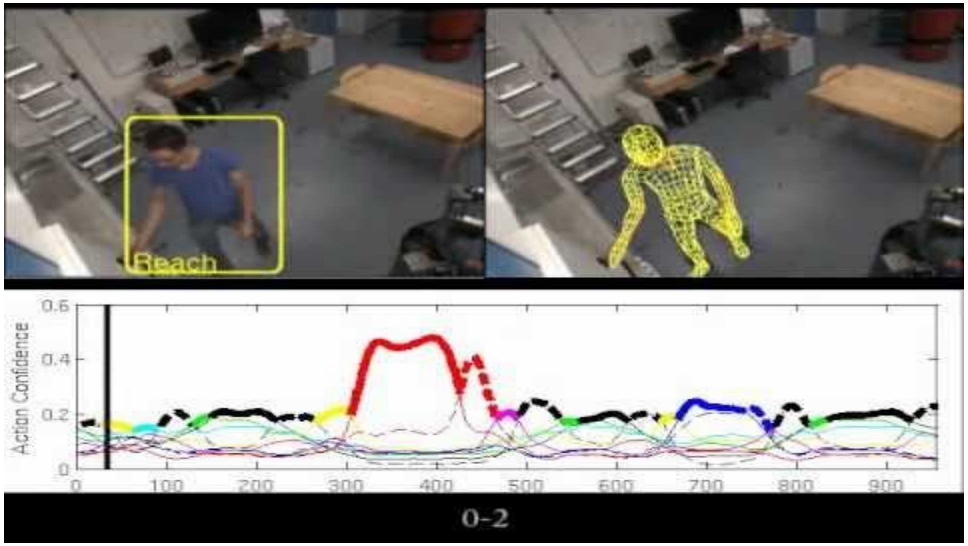
Lecture 3:

- Classification with Logistic Regression
- Advanced optimization techniques
- Underfitting & Overfitting
- Model selection (Training- & Validation- & Test set)

CLASSIFICATION WITH LOGISTIC REGRESSION

Logistic Regression

- **Classification** and not regression
- Classification = recognition



Logistic Regression

- "The" default **classification** model
  - Binary classification
  - Extensions to multi-class later in the course

- Simple classification algorithm
  - **Convex** cost - unique local optimum
  - Gradient descent
  - No more parameter than with linear regression

- Interpretability of parameters

- Fast evaluation of hypothesis for making predictions

LOGISTIC REGRESSION

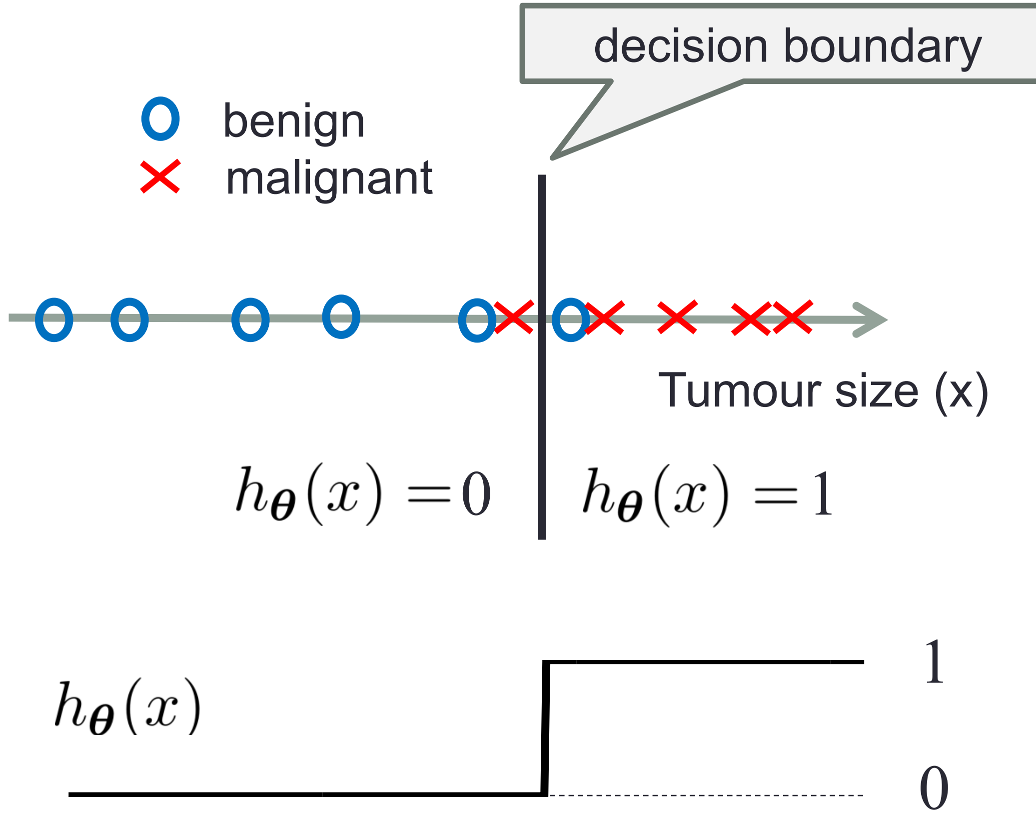
Hypothesis

Example (step function hypothesis)

"labelled data"

i	Tumour size (mm) x	Malignant ? y
1	2.3	**0 (N)**
2	5.1	**1 (Y)**
3	1.4	**0 (N)**
4	6.3	**1 (Y)**
5	5.3	**1 (Y)**
	…	…

labels

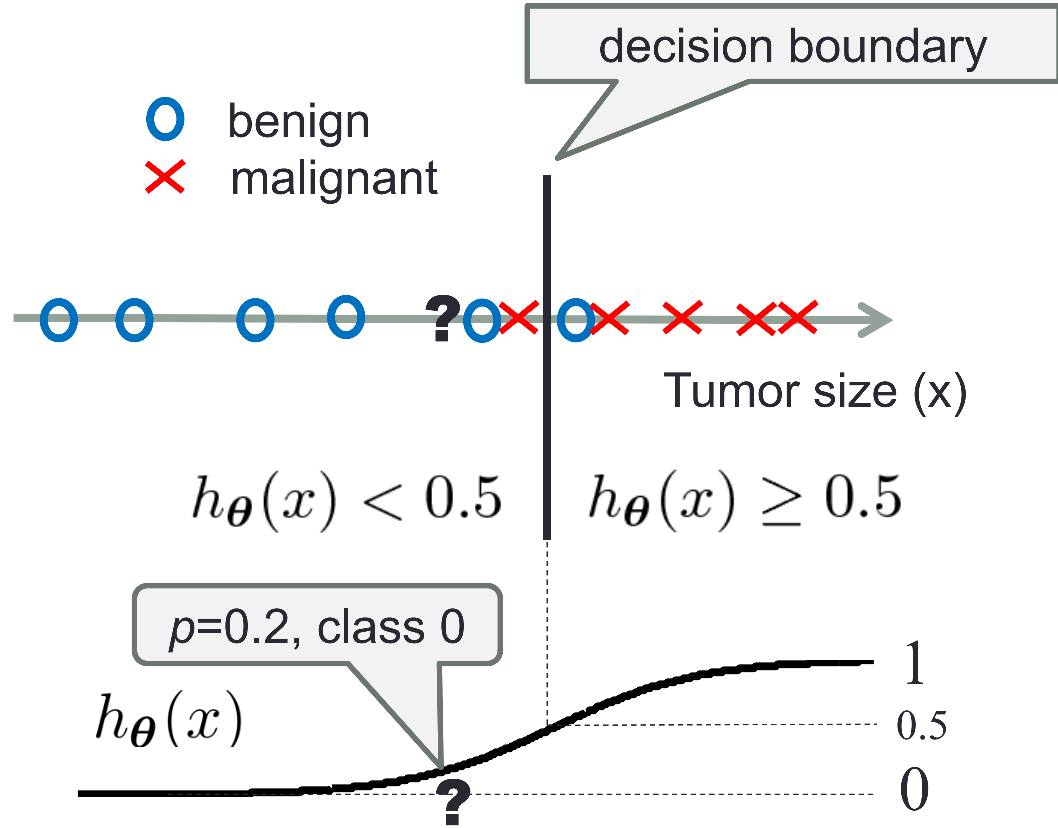


Example (logistic function hypothesis)

„labeled data“

i	Tumor size (mm) x	Malignant ? y
1	2.3	**0 (N)**
2	5.1	**1 (Y)**
3	1.4	**0 (N)**
4	6.3	**1 (Y)**
5	5.3	**1 (Y)**
	…	…

↑
labels



Hypothesis: Tumor is malignant with **probability p**

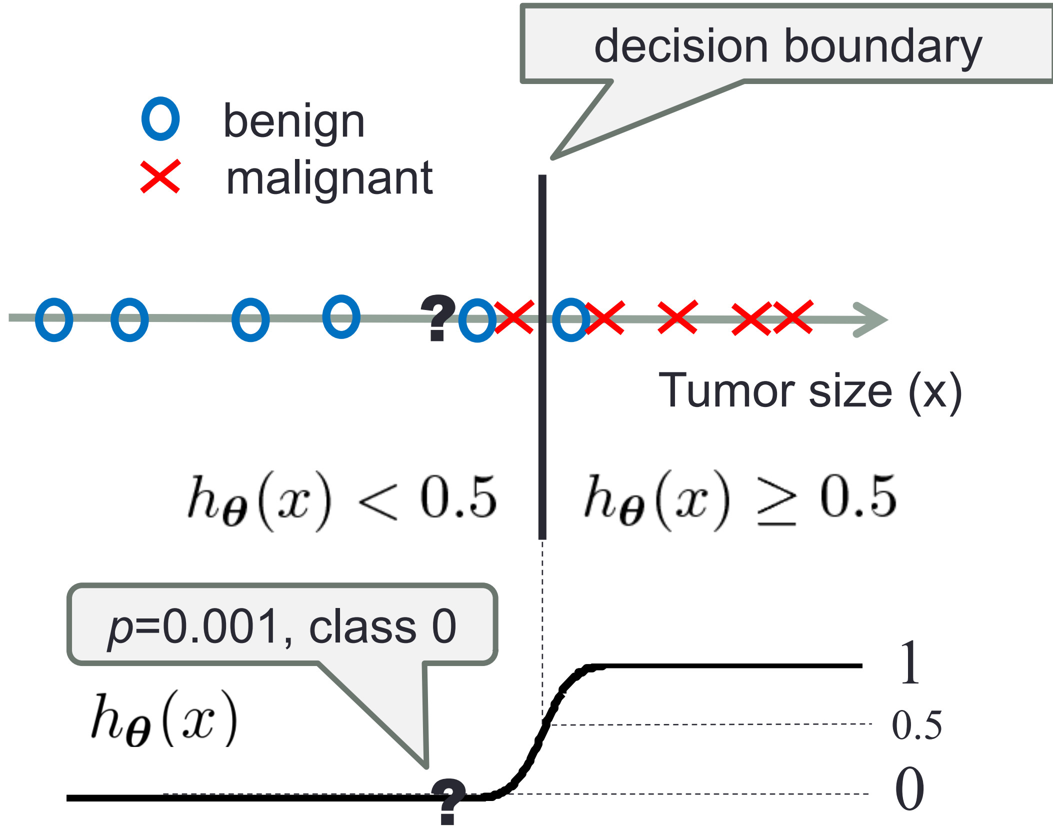
Classification: if $p < 0.5$: 0
if $p \geq 0.5$: 1

Example (logistic function hypothesis)

„labeled data“

i	Tumor size (mm) x	Malignant ? y
1	2.3	0 (N)
2	5.1	1 (Y)
3	1.4	0 (N)
4	6.3	1 (Y)
5	5.3	1 (Y)
	…	…

↑
labels



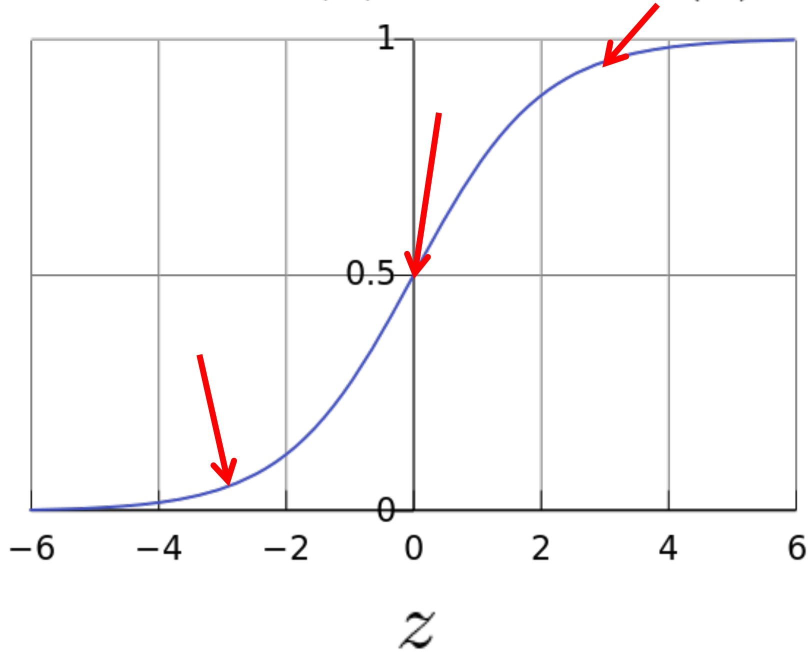
Hypothesis: Tumor is malignant with **probability p**

Classification: if $p < 0.5$: 0
if $p \geq 0.5$: 1

Logistic (Sigmoid) function

$$\sigma(-3) \approx 0.05 \quad \sigma(0) = 0.5 \quad \sigma(3) \approx 0.95$$

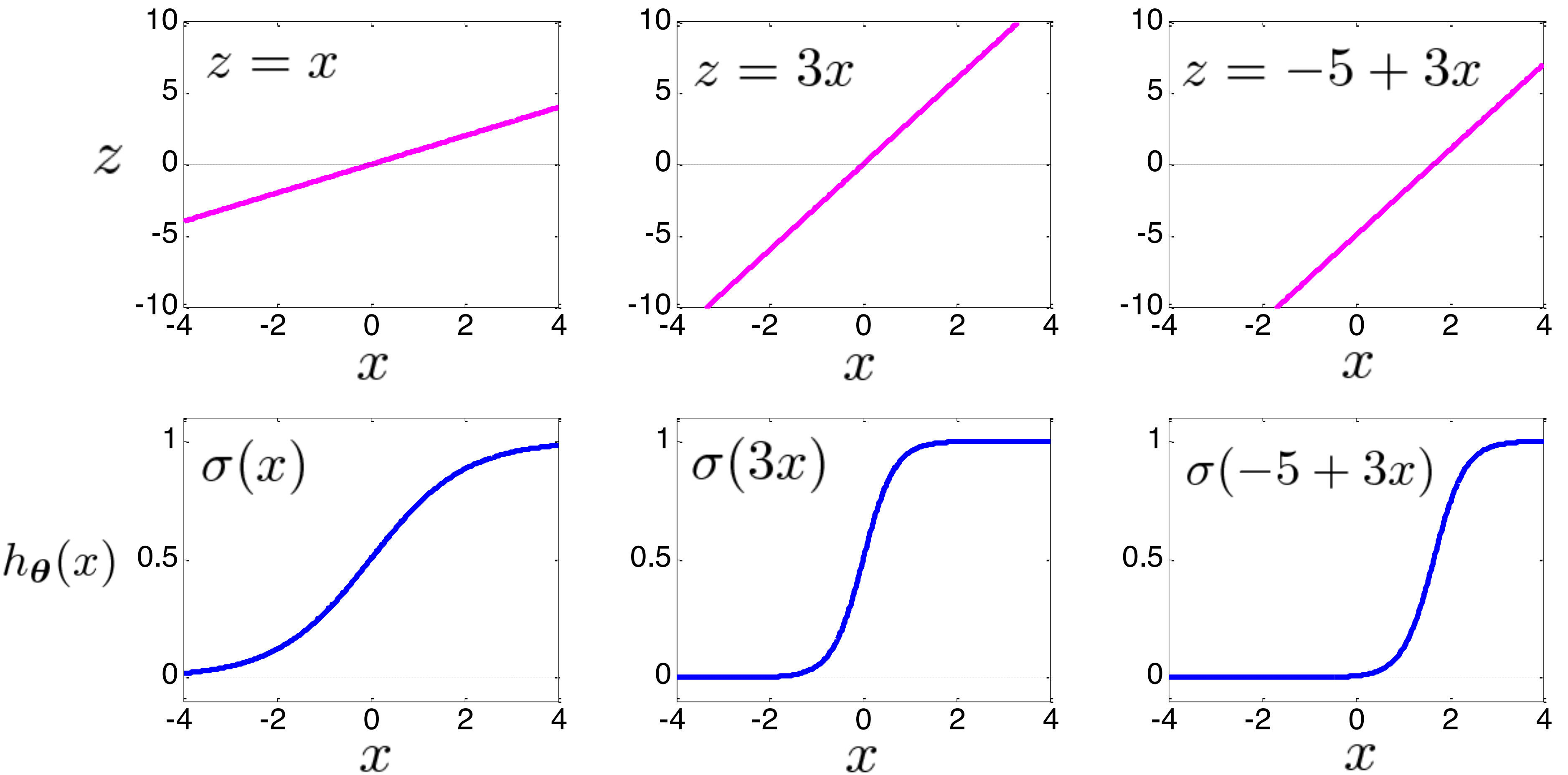
$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



- Advantages over step function for classification:
  - Differentiable → (**gradient** descent)
  - Contains additional information (how certain is the prediction?)

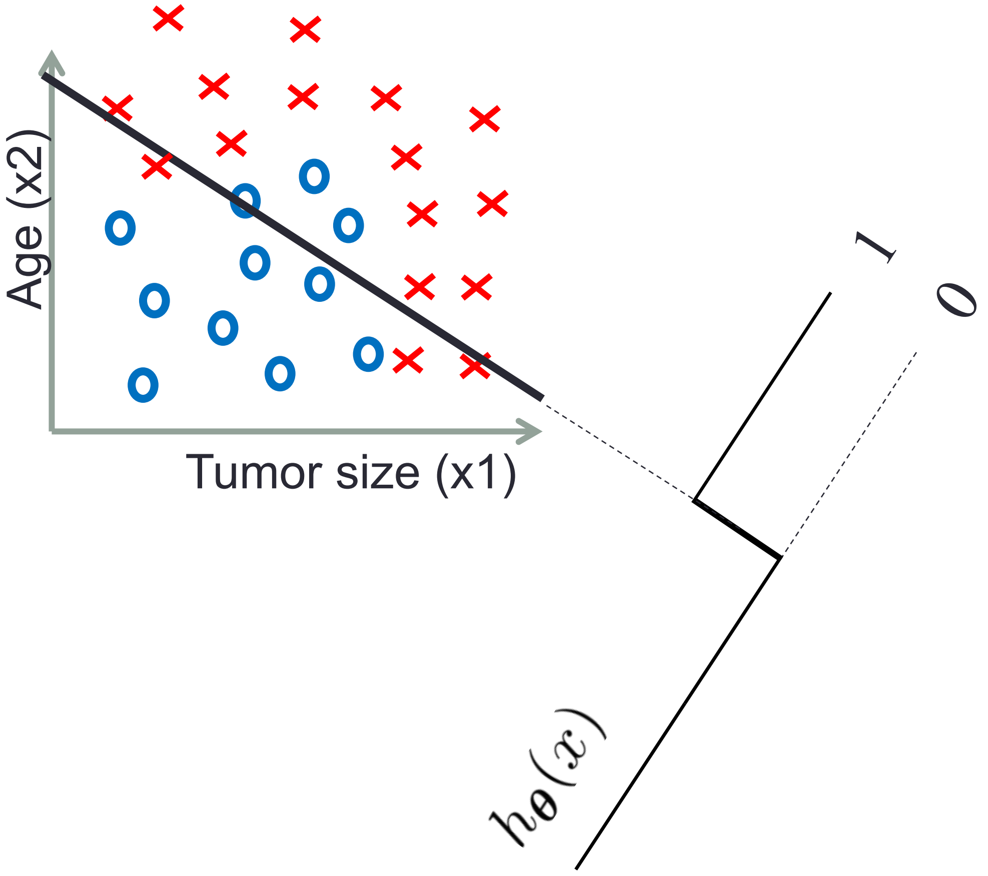
Logistic regression hypothesis (one input)

$$h_{\boldsymbol{\theta}}(x) = \sigma(z) = \sigma(\theta_0 + \theta_1 \cdot x)$$



Classification with multiple inputs

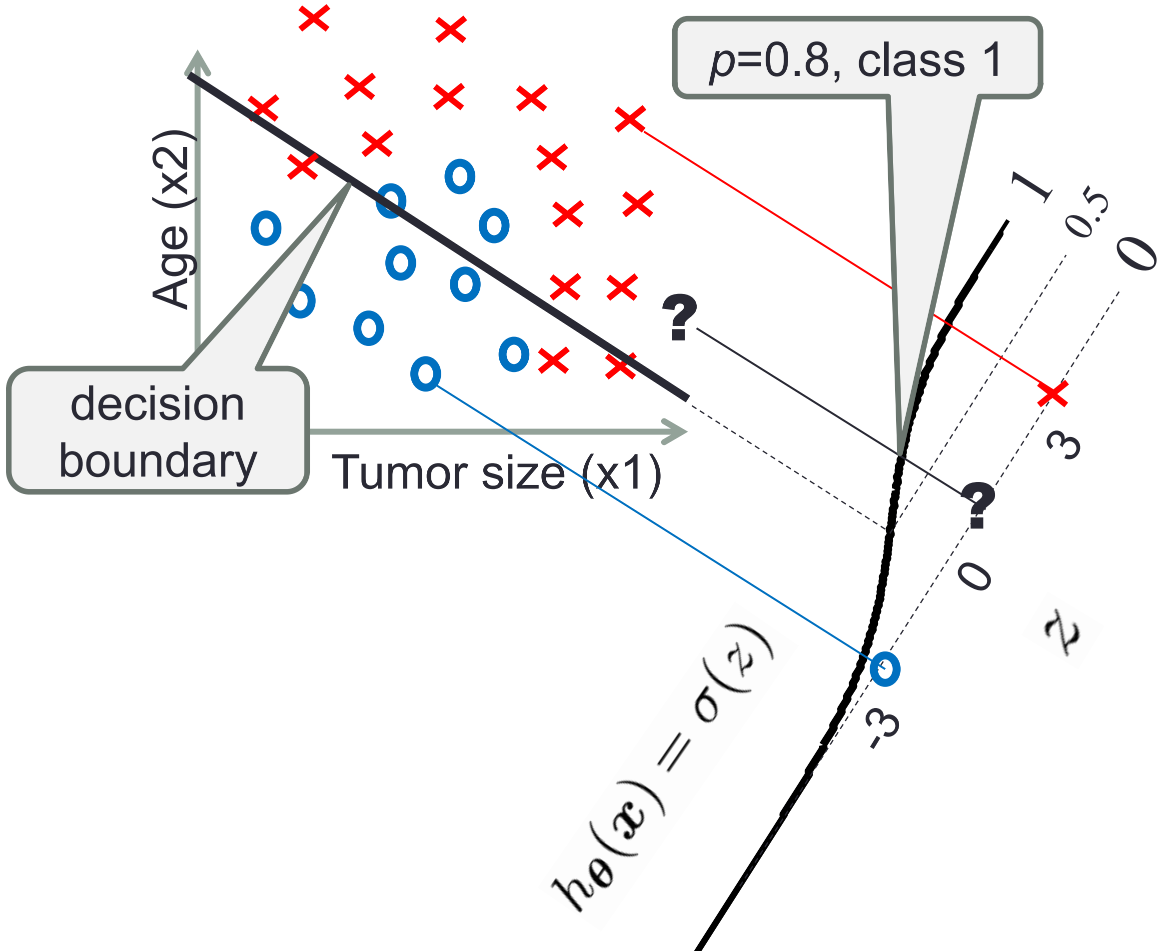
i	Tumor size (mm) x1	Age x2	Malignant? y
1	2.3	25	**0 (N)**
2	5.1	62	**1 (Y)**
3	1.4	47	**0 (N)**
4	6.3	39	**1 (Y)**
5	5.3	72	**1 (Y)**
	…		…



Multiple inputs and logistic hypothesis

i	Tumor size (mm) x1	Age x2	Malignant? y
1	2.3	25	**0 (N)**
2	5.1	62	**1 (Y)**
3	1.4	47	**0 (N)**
4	6.3	39	**1 (Y)**
5	5.3	72	**1 (Y)**
	…		…

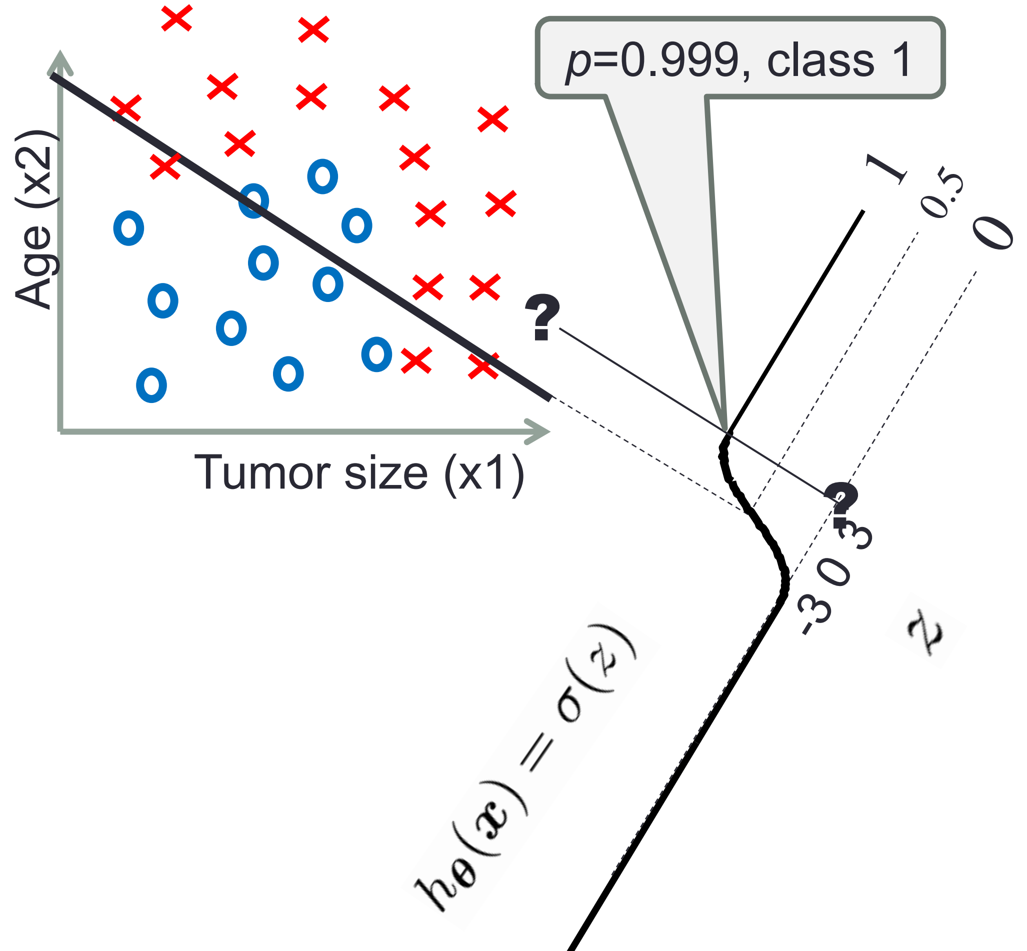
1. Reduce point in high-dimensional space to a scalar z
2. Apply logistic function



Classification with multiple inputs

1. Reduce point in high-dimensional space to a scalar z
2. Apply logistic function

i	Tumor size (mm) x1	Age x2	Malignant? y
1	2.3	25	**0 (N)**
2	5.1	62	**1 (Y)**
3	1.4	47	**0 (N)**
4	6.3	39	**1 (Y)**
5	5.3	72	**1 (Y)**
	…		…



Logistic regression hypothesis

1. Reduce high-dimensional input $\boldsymbol{x}$ to a scalar

$$\begin{aligned} z &= \boldsymbol{x}^T \boldsymbol{\theta} \\ &= \theta_0 + \theta_1 \cdot x_1 + \cdots + \theta_n \cdot x_n \end{aligned}$$

2. Apply logistic function

$$\begin{aligned} h_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \sigma(\boldsymbol{x}^T \boldsymbol{\theta}) \\ &= \sigma(\theta_0 + \theta_1 \cdot x_1 + \cdots + \theta_n \cdot x_n) \end{aligned}$$

3. Interpret output $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ as probability and predict class:

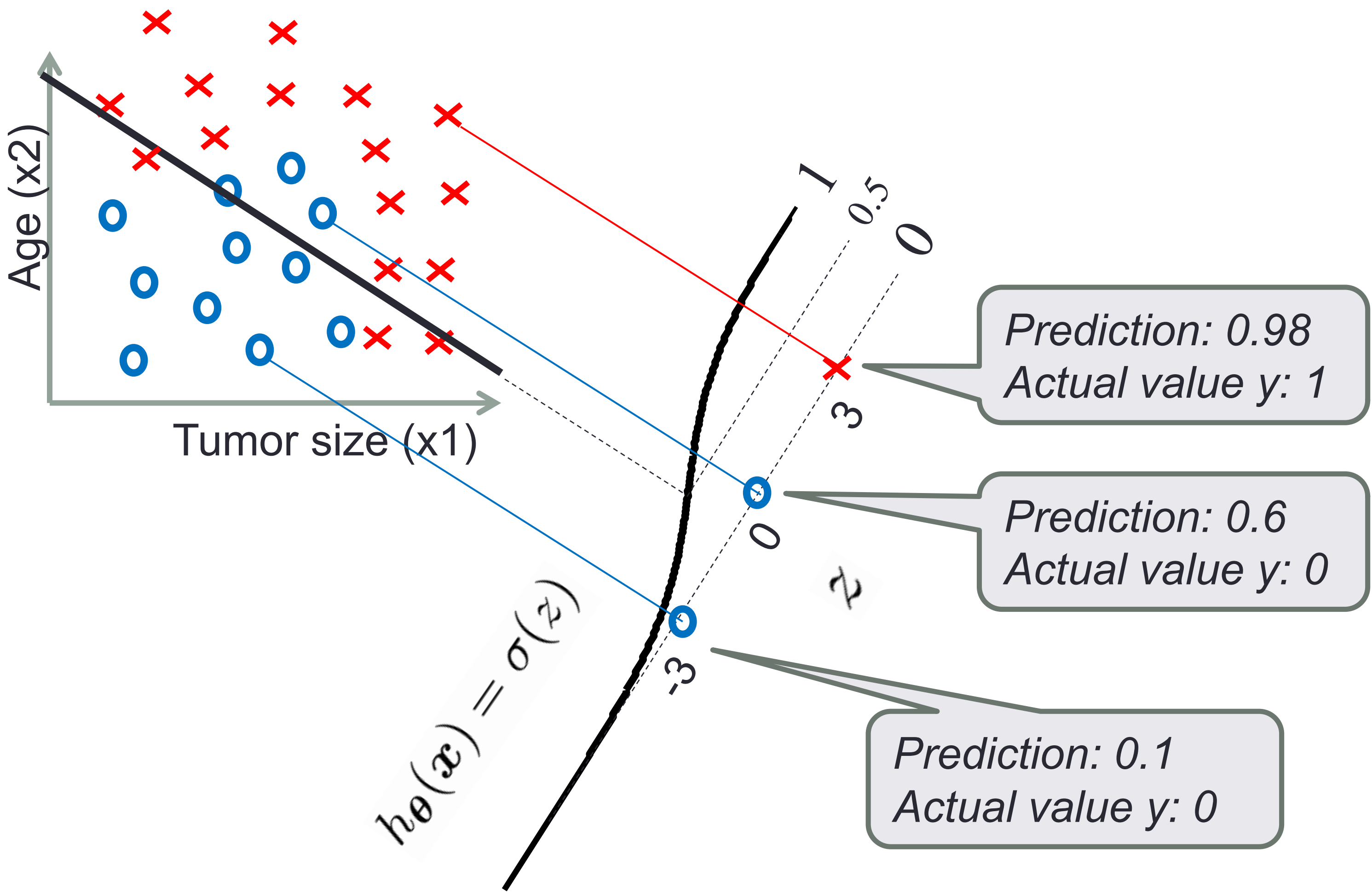
Class = $\begin{cases} 0 & \text{if } h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5 \\ 1 & \text{if } h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5 \end{cases}$

LOGISTIC REGRESSION

Cost function

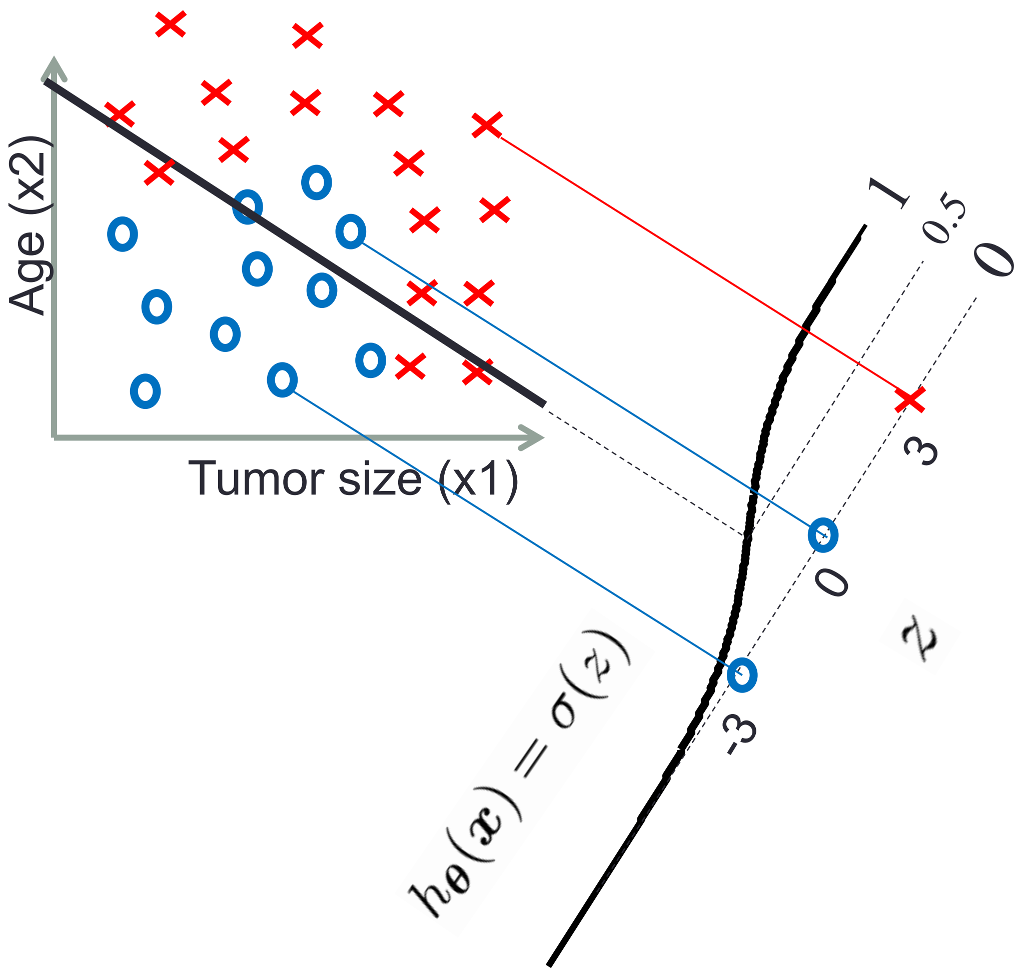
Logistic regression cost function

- How well does the hypothesis $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{x}^T\boldsymbol{\theta})$ fit the data?



Logistic regression cost function

- **Probabilistic model**: y is 1 with probability: $$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{x}^T\boldsymbol{\theta})$$



Logistic regression cost function

$$\langle x^{(1)}, y^{(1)} \rangle \; ... \; \langle x^{(m)}, y^{(m)} \rangle$$

- The "likelihood" of data:

$$p(y = (y_1, \ldots y_n), X = (x_1 \ldots x_n) | \theta)$$

  i.e the probability of the given data as a function of parameters

  - For logistic regression we care about:

$$p(y = (y_1, \ldots y_n) | X = (x_1 \ldots x_n); \theta)$$

- We want to maximize the likelihood of data
- We usually maximize the log likelihood instead
  - i.e $\log p(y|X;\ \theta)$
  - (or minimize the negative log-likelihood)

- Because logarithm:
  - Is monotonically increasing
  - And products become sums
  - more numerically stable for small numbers (like probabilities)

Logistic regression cost function

Probabilistic model: y is 1 with probability: $p(C_1|X) = h_\theta(\boldsymbol{x}) = \sigma(x^T\theta)$

The parameters should maximize the **log-likelihood** of the data

$$\max_\theta \log p(y|X;\ \theta)$$

If data points are independent $p(y_i,\ y_j|X;\ \theta) = p(y_j|X;\ \theta)\, p(y_i|X;\ \theta)$

$$\max_\theta \sum_i \log p(y_i|X;\ \theta)$$

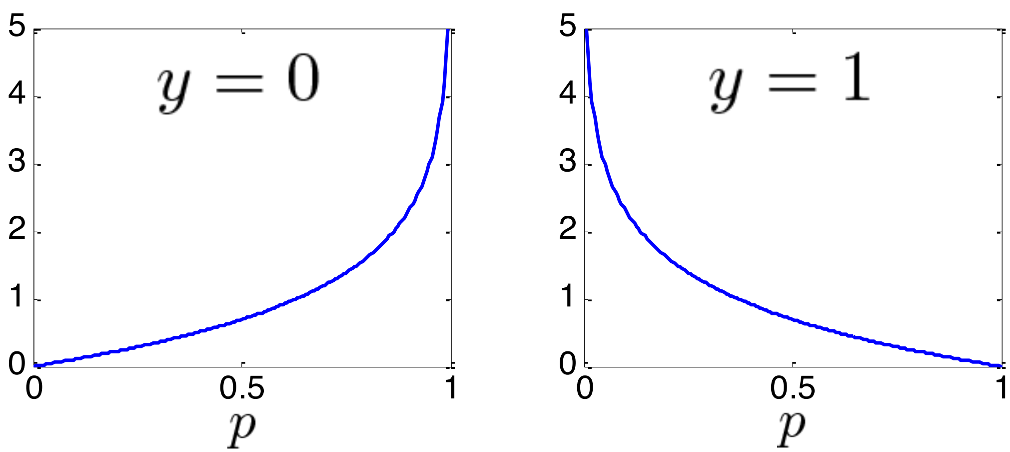
Separating positive and negative examples

$$\max_\theta \sum_{t_i=1} \log p(\underbrace{y_i = 1|X;\ \theta}_{\sigma(x^T\theta)}) + \sum_{t_i=0} \log p(\underbrace{y_i = 0|X;\ \theta}_{1-\sigma(x^T\theta)})$$

Logistic regression cost function

- How well does the hypothesis $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{x}^T\boldsymbol{\theta})$ fit the data?
- „Cost“ for predicting probability *p* when the real value is *y*:

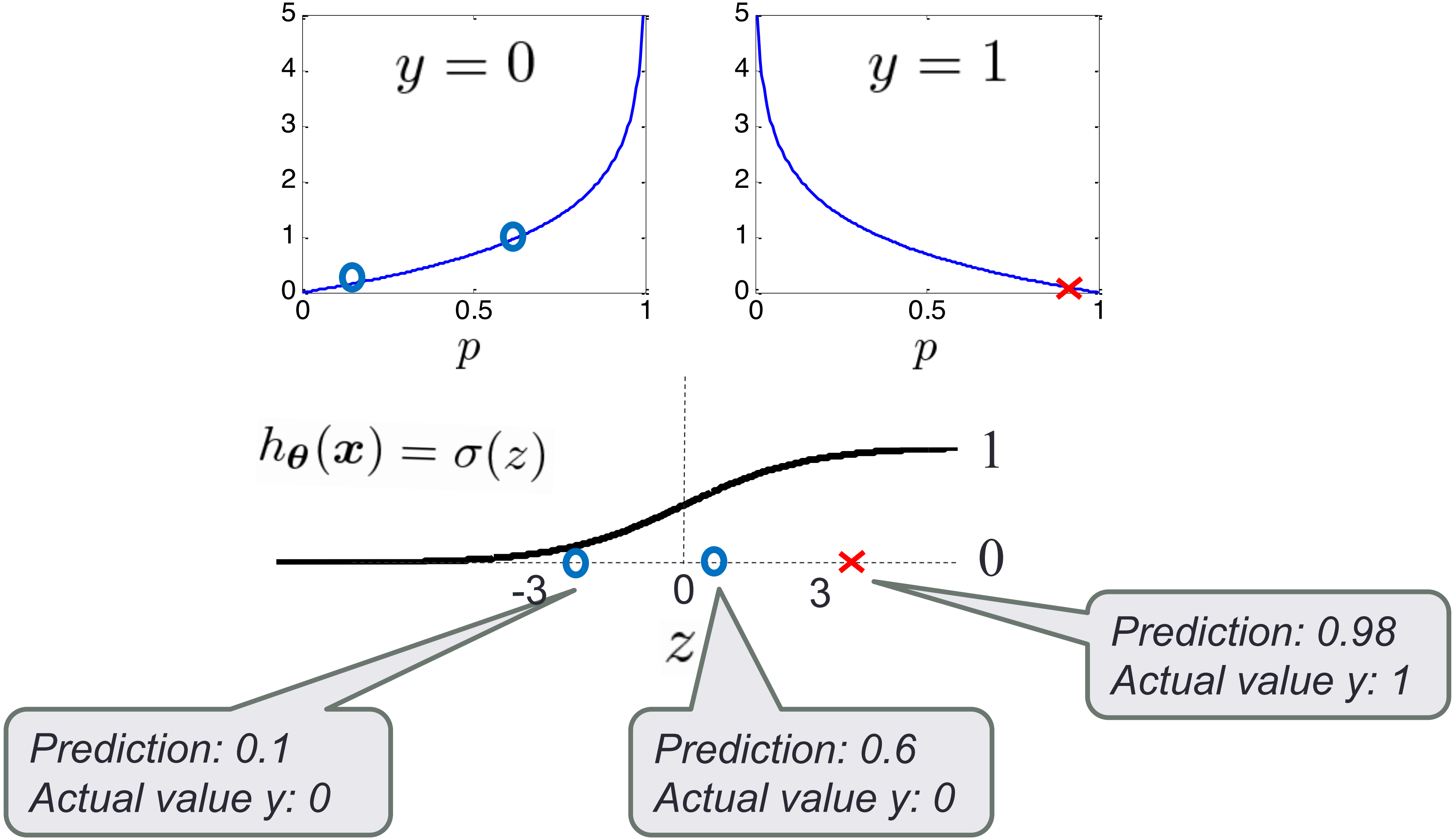
$$\mathrm{Cost}(p,\ y) = \begin{cases} -\log(1-p) & \text{if } y = 0 \ , \\ -\log(p) & \text{if } y = 1 \ . \end{cases}$$



- Mean over all training examples: $J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}),\ y^{(i)})$

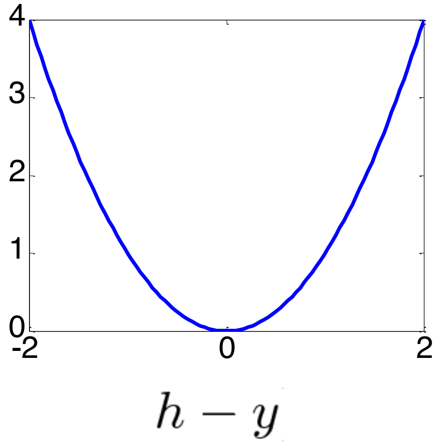
Multiple inputs and logistic hypothesis

- How well does the hypothesis $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{x}^T\boldsymbol{\theta})$ fit the data?



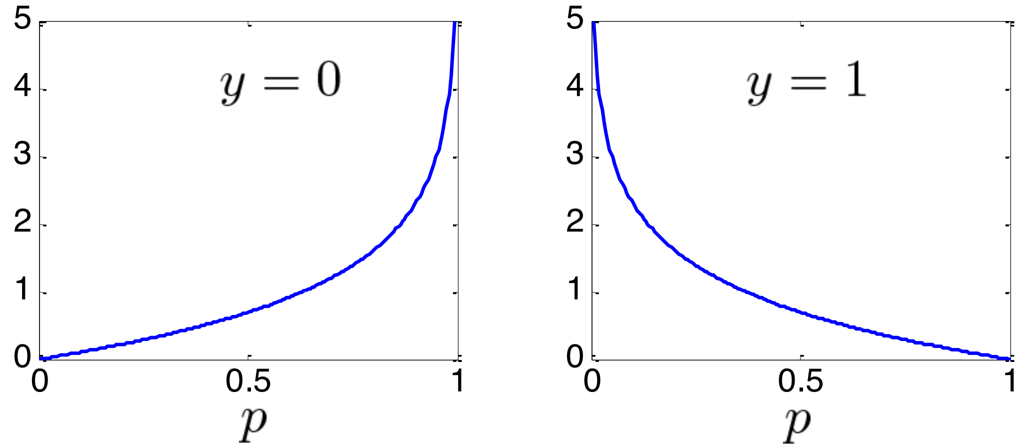
Comparison cost functions

Linear regression



$$\mathrm{Cost}(h,\ y) = (h-y)^2$$

Logistic regression



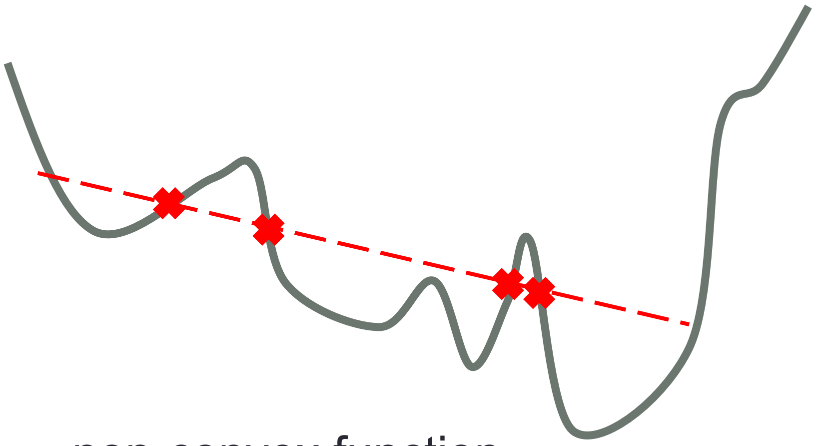
$$\mathrm{Cost}(p,\ y) = \begin{cases} -\log(1-p) & \text{if } y = 0\ , \\ -\log(p) & \text{if } y = 1\ . \end{cases}$$

Mean over all training examples:

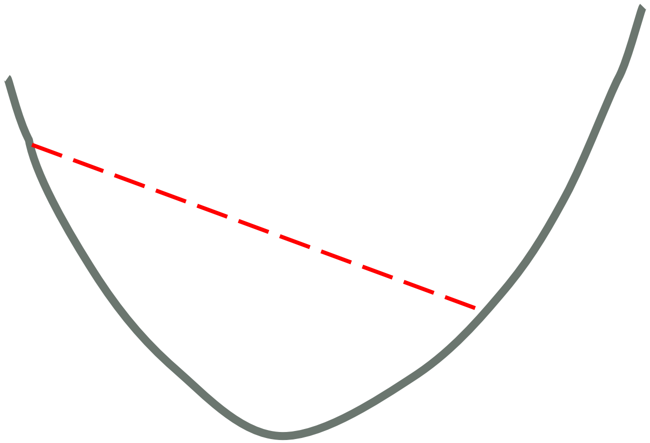
$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}),\ y^{(i)})$$

Why not mean squared error (MSE) again?

- **MSE** with logistic hypothesis is **non-convex** (many local minima)
- Logistic regression **is convex** (unique minimum)
- Cost function can be derived from statistical principles („**maximum likelihood**“)



non-convex function



convex function

LOGISTIC REGRESSION

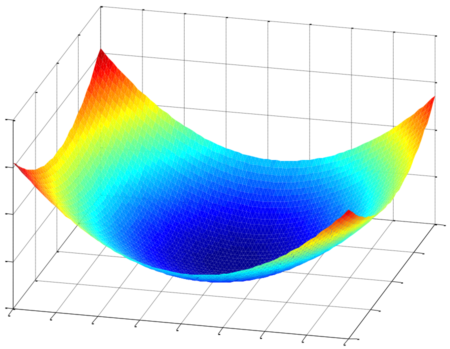
Learning from data

Minimizing the cost via gradient descent

- Gradient descent

$$\theta_j := \theta_j - \eta \cdot \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

(simultaneous update for j=0…n)

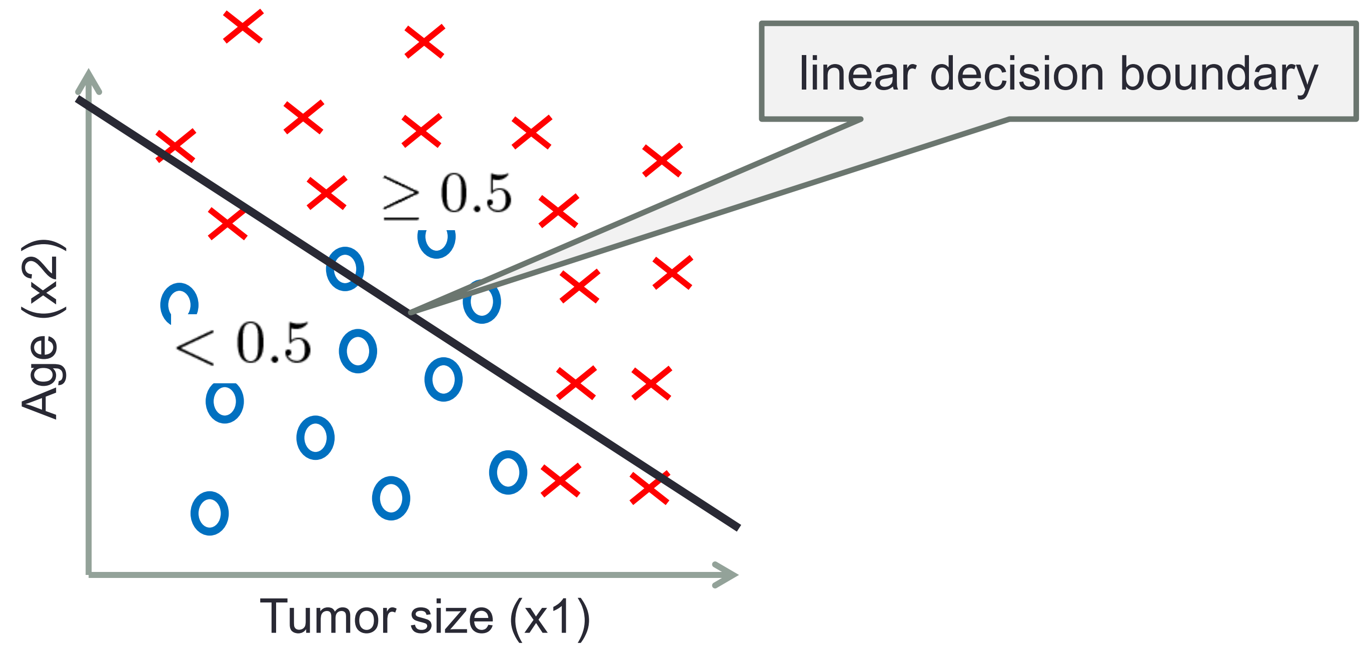


- Gradient of logistic regression cost:

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \Big(\underbrace{h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)}}_{\text{„error“}} \Big) \cdot \underbrace{x_j^{(i)}}_{\text{„input“}}$$

(for j=0: $x_0^{(i)} = 1$)

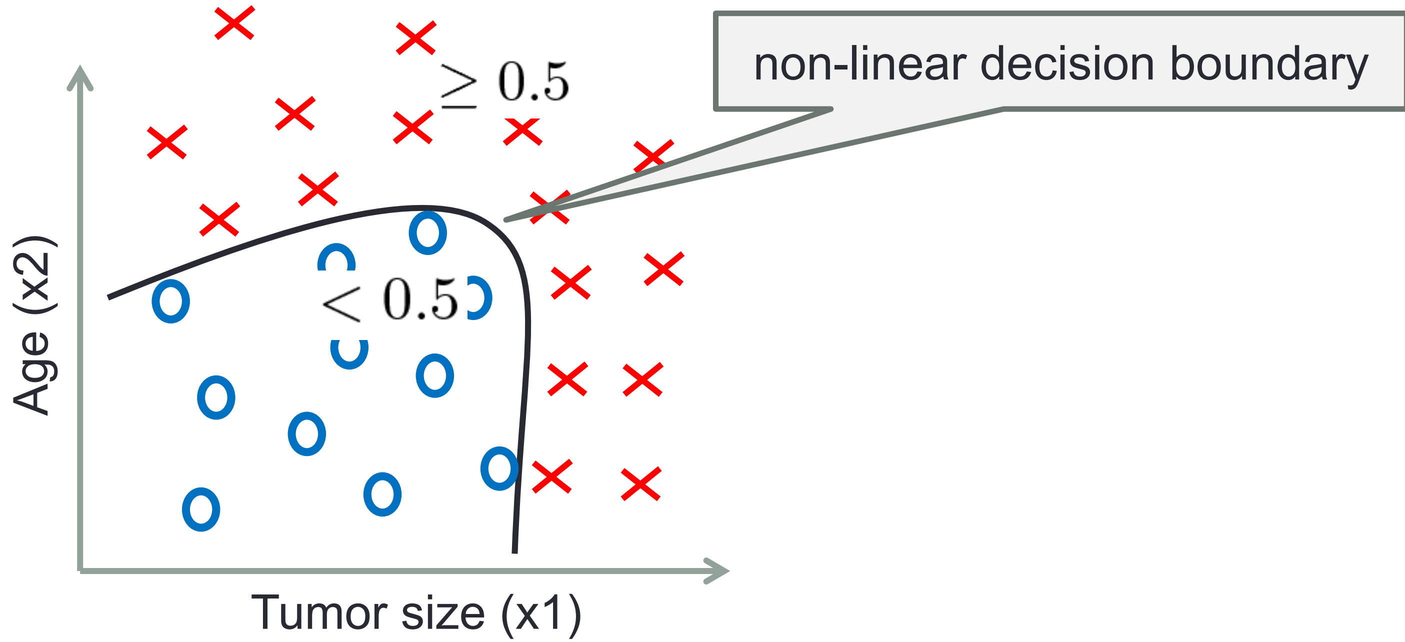
Linear features



$$x_1 = \text{Tumor Size},\ x_2 = \text{Age}$$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(-10 + 2 \cdot x_1 + 0.05 \cdot x_2)$$

Non-linear features

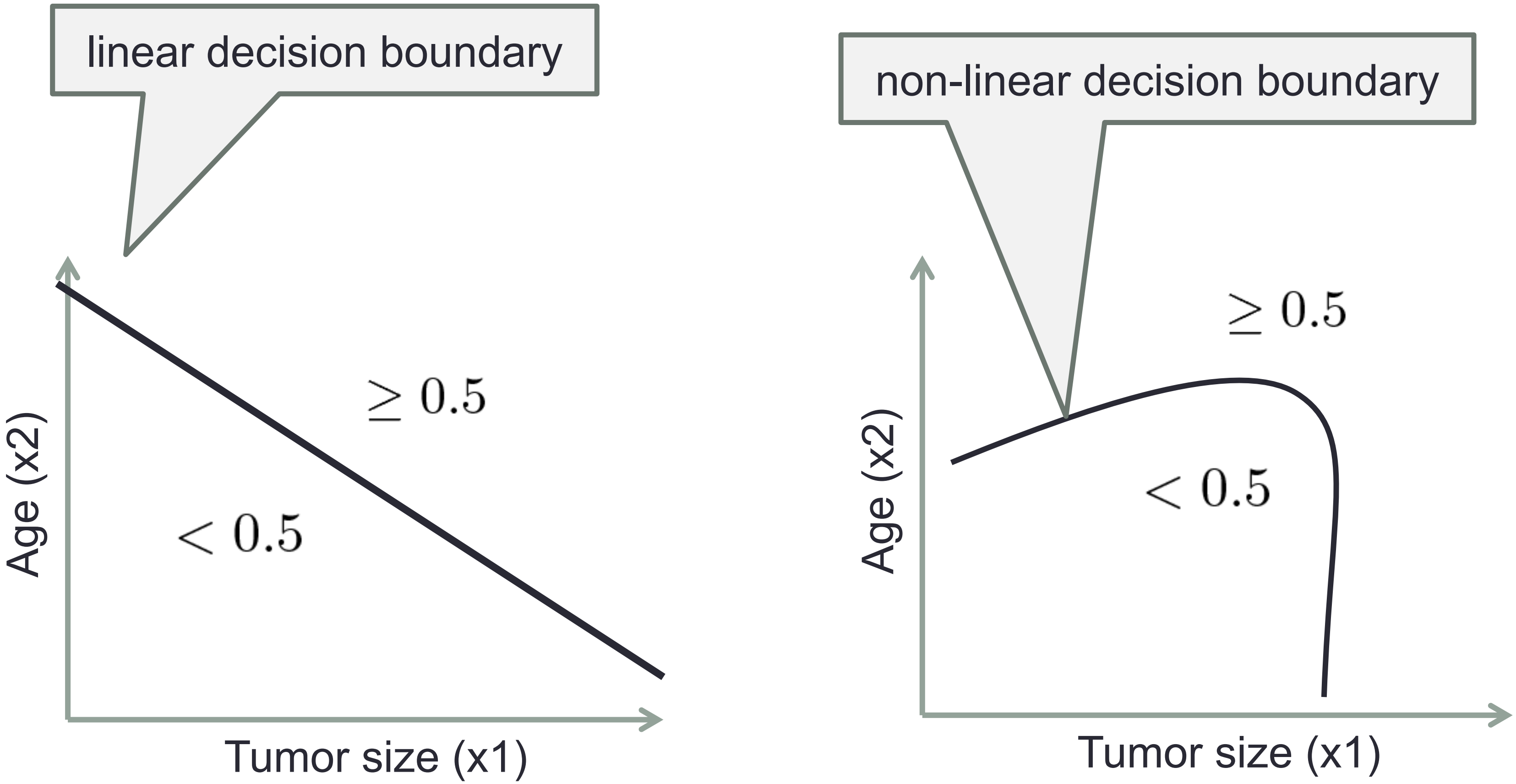


$$\phi_1 = \text{Tumor Size},\ \phi_2 = \text{Age},\ \phi_3 = \text{Tumor Size}^2,$$

$$\phi_4 = \text{Age}^2,\ \phi_5 = \text{Tumor Size} \cdot \text{Age},\ \ldots$$

$$h_{\boldsymbol{\theta}}(\boldsymbol{\phi}) = \sigma(-3 + 1.2 \cdot \phi_1 + 0.07 \cdot \phi_2 - 0.9 \cdot \phi_3 + \ldots)$$

Decision boundaries



$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(-10 + 2 \cdot x_1 + 0.05 \cdot x_2) \qquad h_{\boldsymbol{\theta}}(\boldsymbol{\phi}) = \sigma(-3 + 1.2 \cdot \phi_1 + 0.07 \cdot \phi_2 - 0.9 \cdot \phi_3 + \dots)$$

Decision boundary is a property of hypothesis, not of data!

Linear vs. Logistic Regression

Linear Regression

- Regression
- Hypothesis $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{\theta}$
- Cost for one training example:

$$\mathrm{Cost}(h,\ y) = (h - y)^2$$

- Gradient

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{2}{m} \sum_{i=1}^{m} \underbrace{\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)} \right)}_{\text{„error“}} \cdot \underbrace{x_j^{(i)}}_{\text{„input“}}$$

- Analytical:

$$\boldsymbol{\theta}^* = \left(\boldsymbol{X}^T \boldsymbol{X} \right)^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

Logistic Regression

- Binary classification (!)
- Hypothesis $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{x}^T \boldsymbol{\theta})$
- Cost for one training example:

$$\mathrm{Cost}(p,\ y) = \begin{cases} -\log(1-p) & \text{if } y = 0\ , \\ -\log(p) & \text{if } y = 1\ . \end{cases}$$

- Gradient

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \underbrace{\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)} \right)}_{\text{„error“}} \cdot \underbrace{x_j^{(i)}}_{\text{„input“}}$$

- No analytical solution!

EVALUATION OF HYPOTHESIS

Training and Test set

Training and Test set

- **Training set**: used by learning algorithm to fit parameters and find a hypothesis.

- **Test set**: independent data set, used after learning to estimate the performance of the hypothesis on **new (unseen) test examples.**

$$\langle \boldsymbol{x}^{(1)}, y^{(1)} \rangle \qquad \langle \boldsymbol{x}_{\text{test}}^{(1)}, y_{\text{test}}^{(1)} \rangle$$

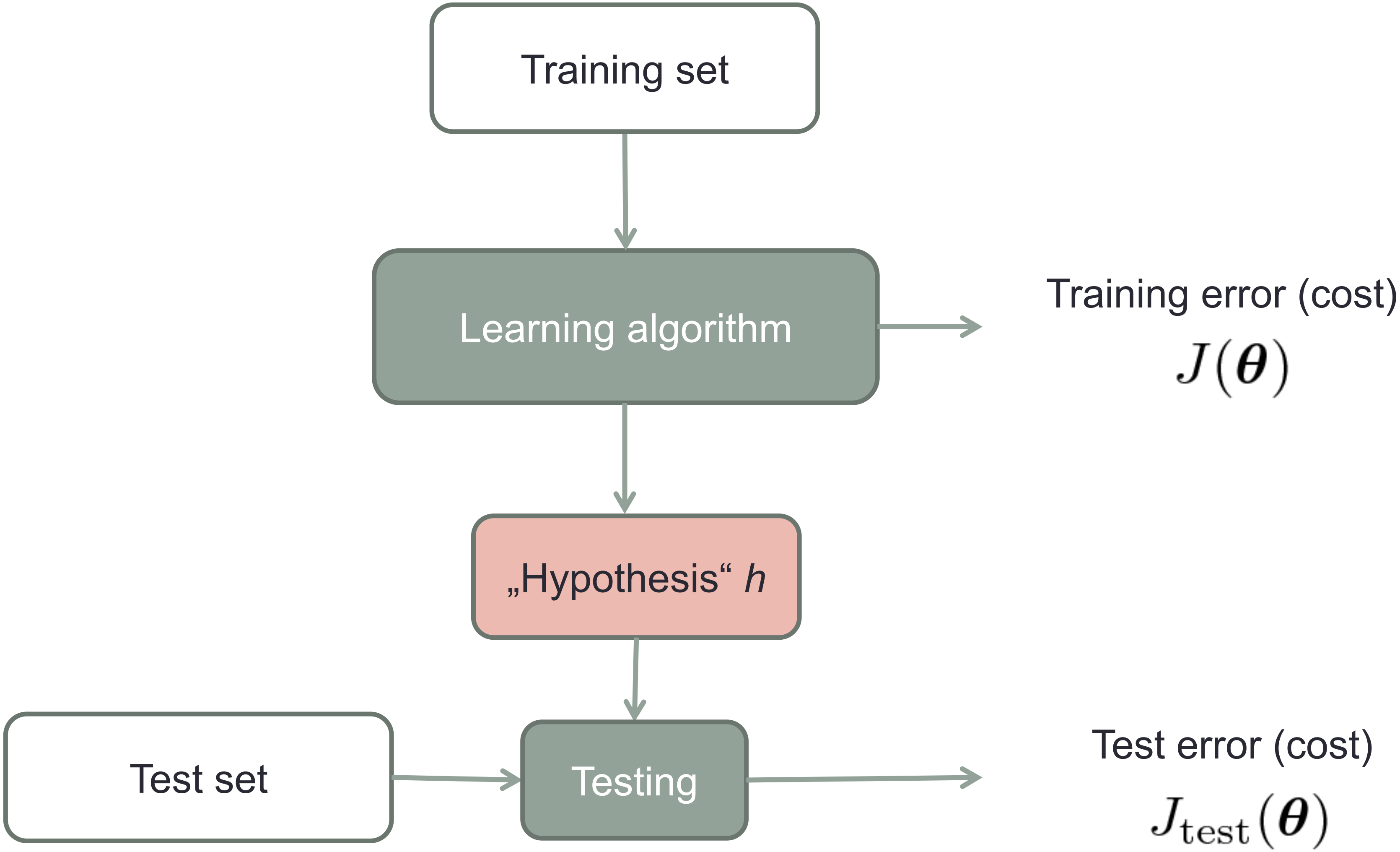
$$\langle \boldsymbol{x}^{(2)}, y^{(2)} \rangle \qquad \langle \boldsymbol{x}_{\text{test}}^{(2)}, y_{\text{test}}^{(2)} \rangle$$

$$\vdots \qquad \vdots$$

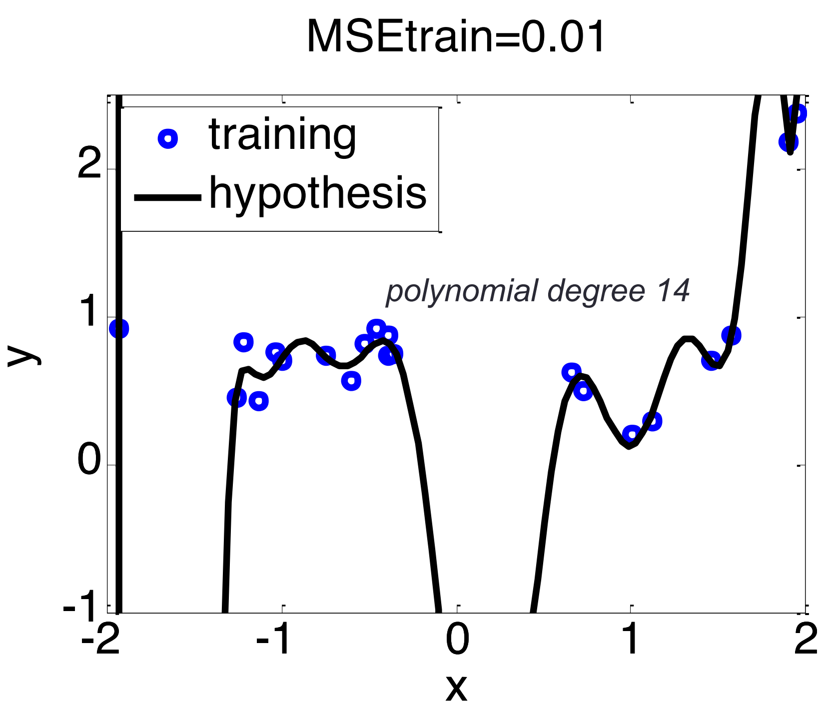
$$\langle \boldsymbol{x}^{(m)}, y^{(m)} \rangle \qquad \langle \boldsymbol{x}_{\text{test}}^{(m_{\text{test}})}, y_{\text{test}}^{(m_{\text{test}})} \rangle$$

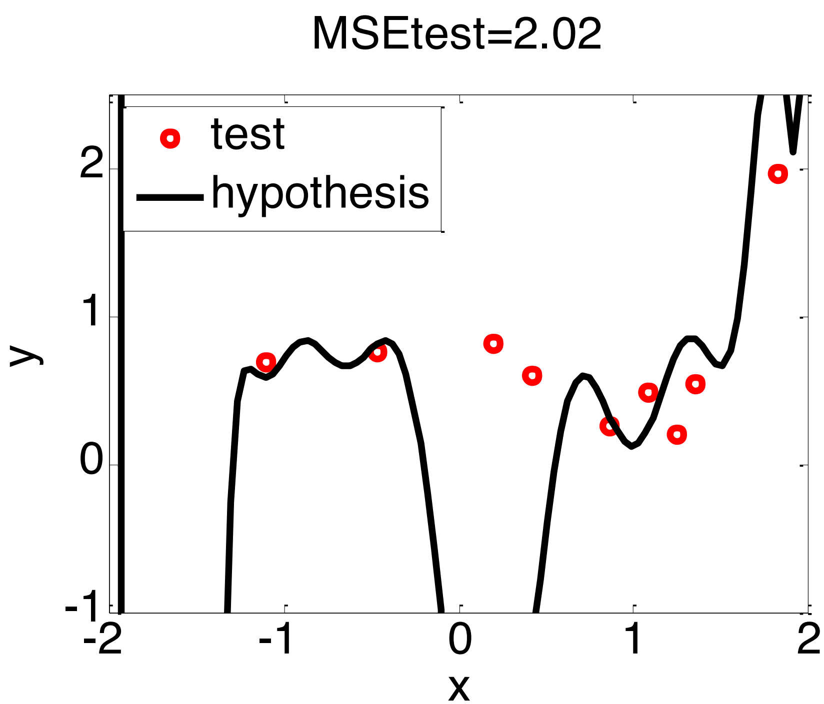
E.g. 80% randomly chosen examples from dataset are training examples, the remaining 20% are test examples. Must be **disjoint subsets**!

Training and Test set workflow



Linear regression training vs. Test error

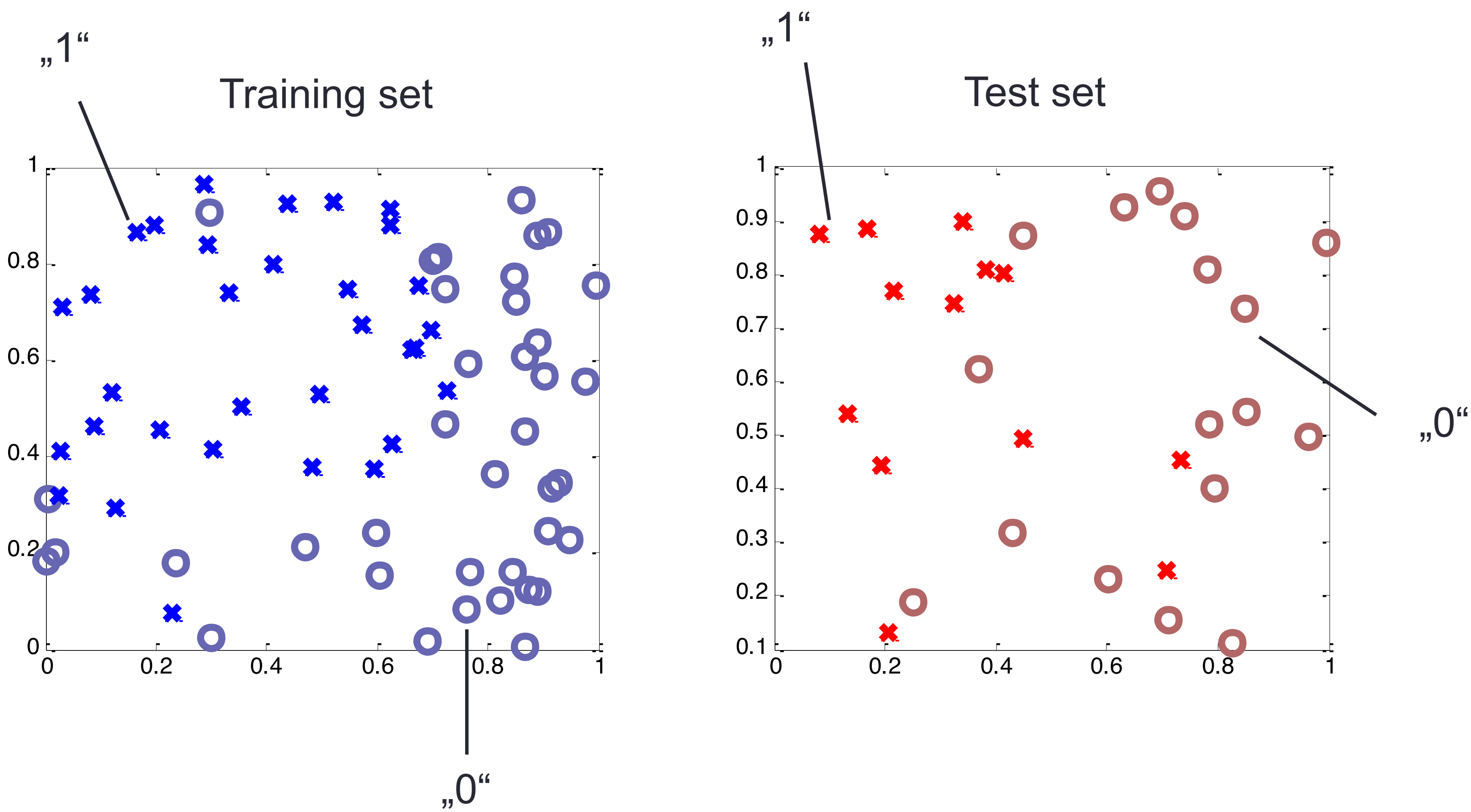




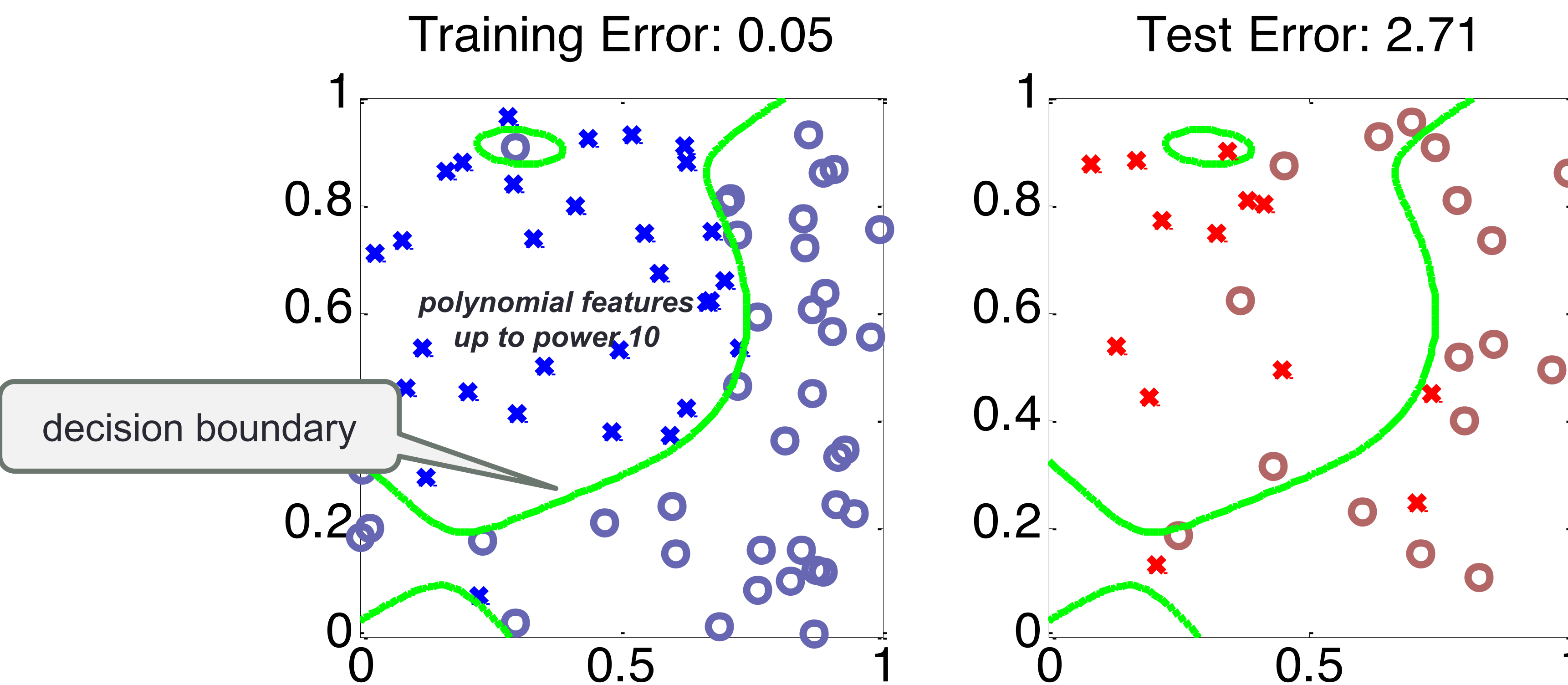
$$J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2$$

$$J_{\text{test}}(\boldsymbol{\theta}) = \frac{1}{m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}_{\text{test}}^{(i)} \right) - y_{\text{test}}^{(i)} \right)^2$$

Classification Training / Test set



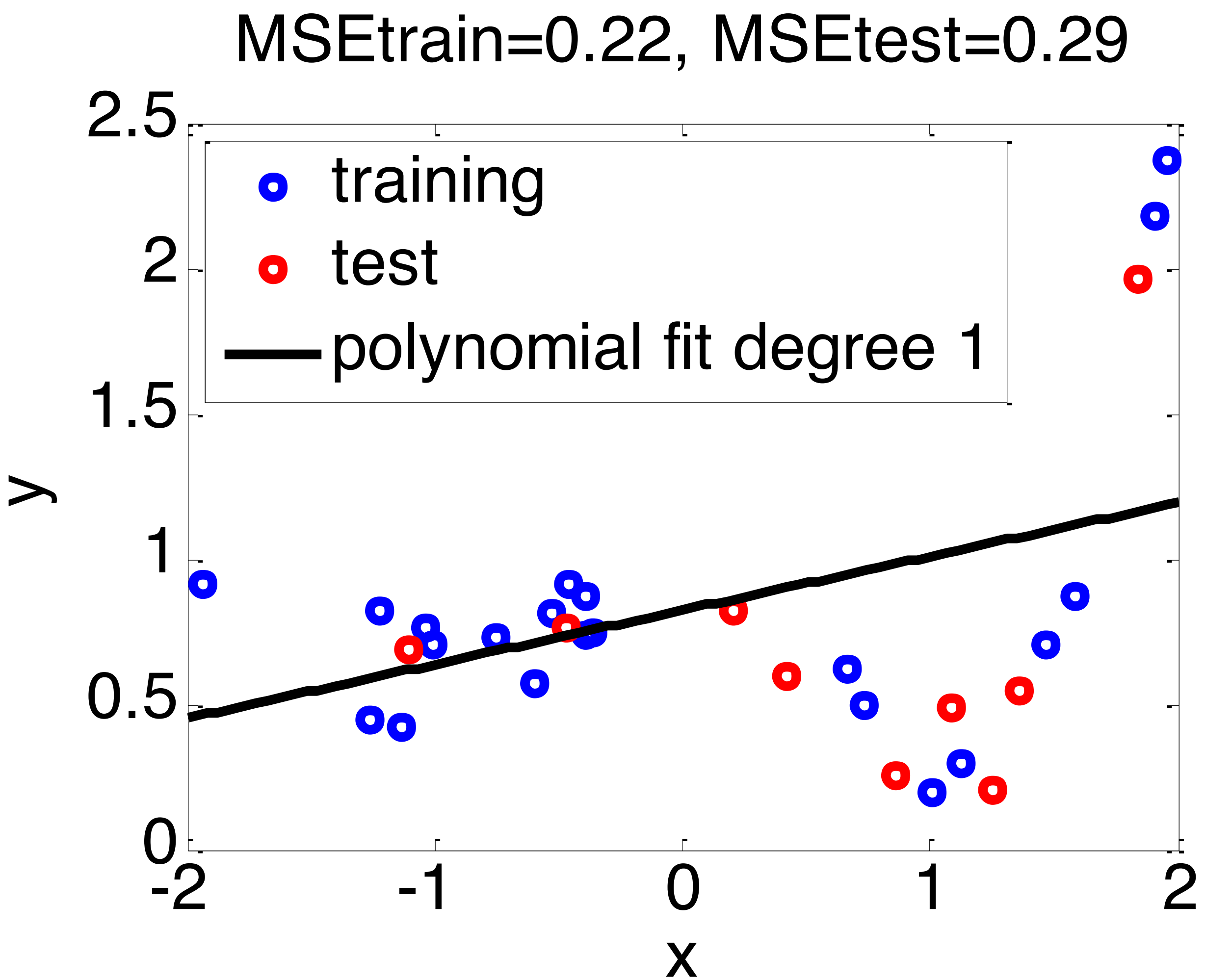
Logistic regression training vs. test error



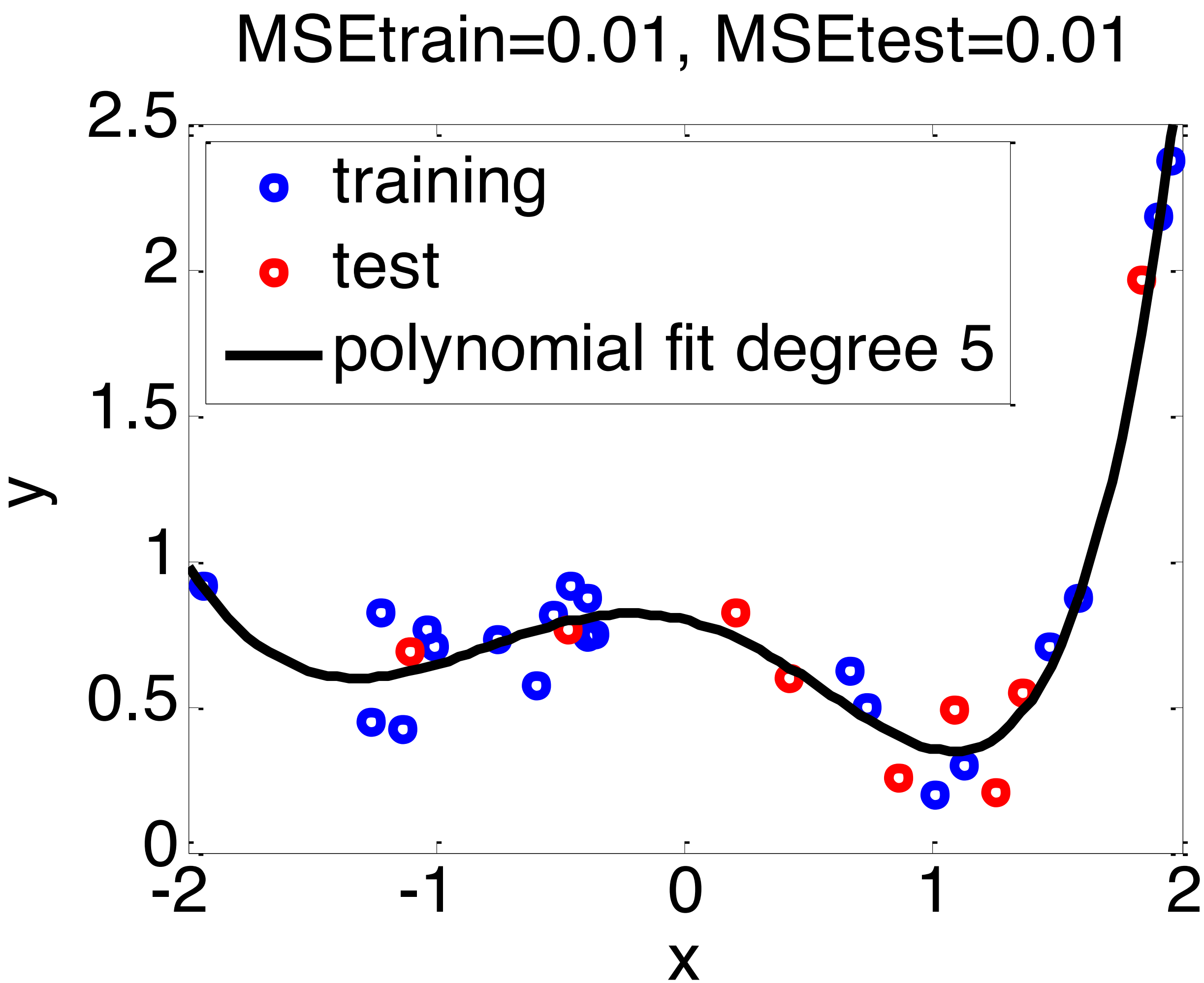
$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}),\ y^{(i)}) \qquad J_{\text{test}}(\boldsymbol{\theta}) = -\frac{1}{m_{\text{test}}}\sum_{i=1}^{m_{\text{test}}} \mathrm{Cost}(h_{\boldsymbol{\theta}}(\boldsymbol{x}_{\text{test}}^{(i)}),\ y_{\text{test}}^{(i)})$$

UNDERFITTING AND OVERFITTING

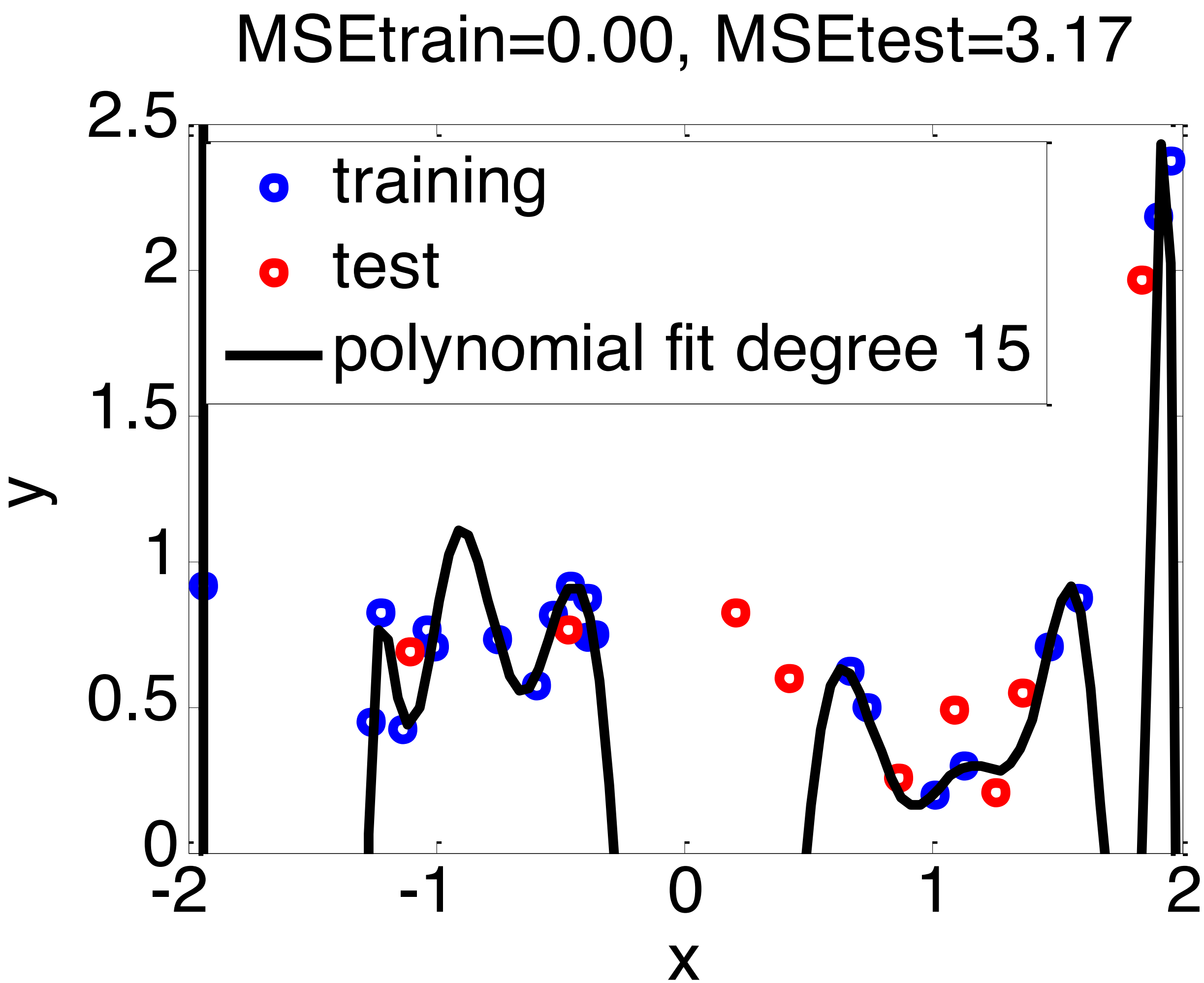
Polynomial regression under-/overfitting



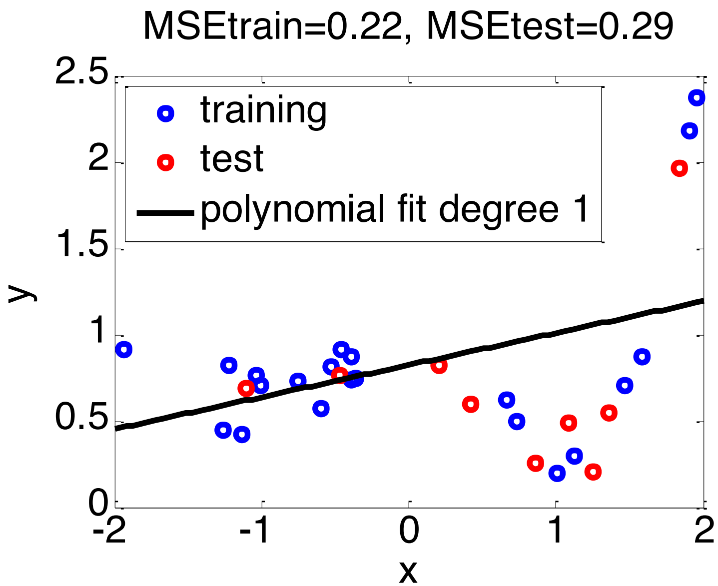
Polynomial regression under-/overfitting



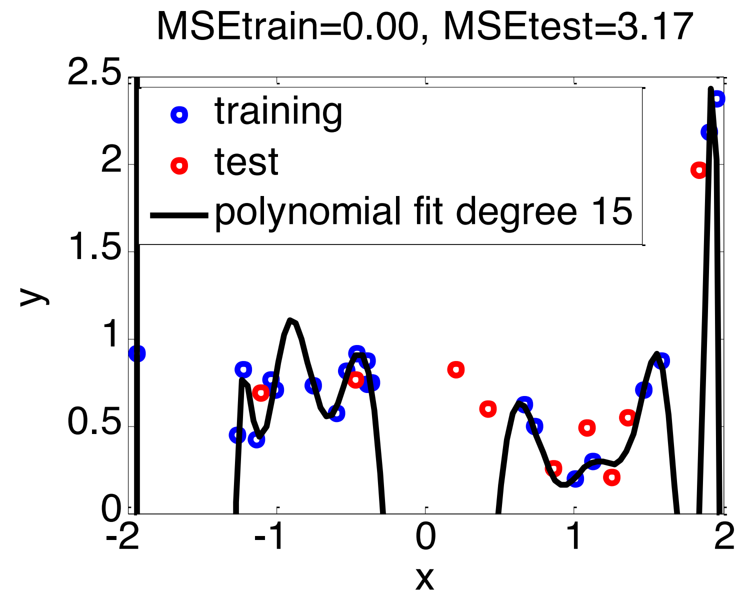
Polynomial regression under-/overfitting



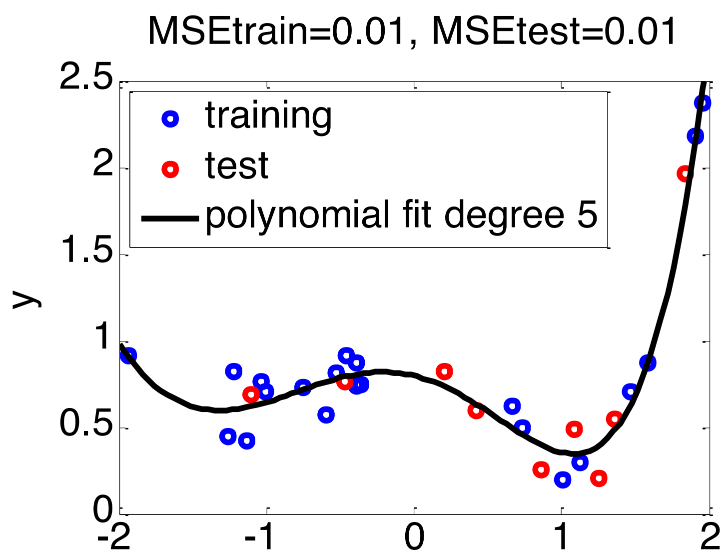
Polynomial regression under-/overfitting



underfitting

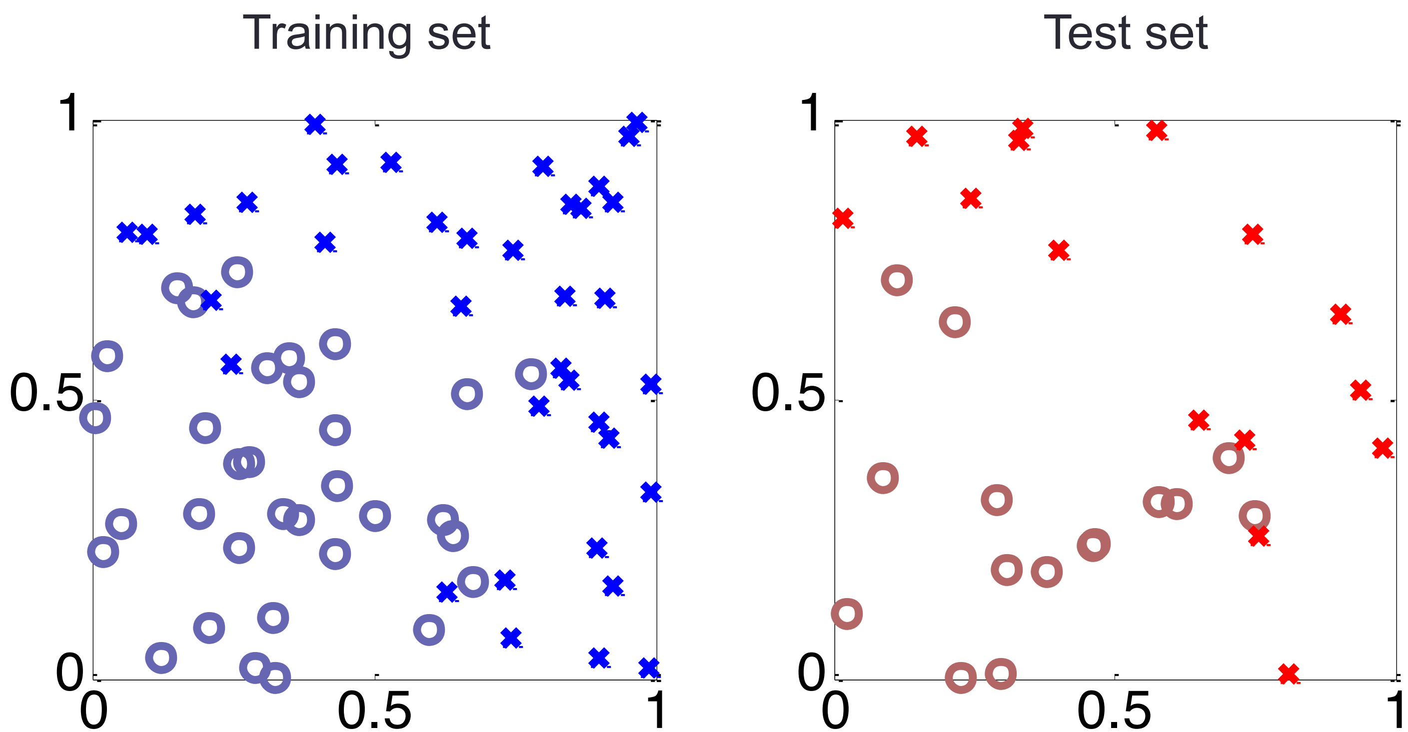


overfitting

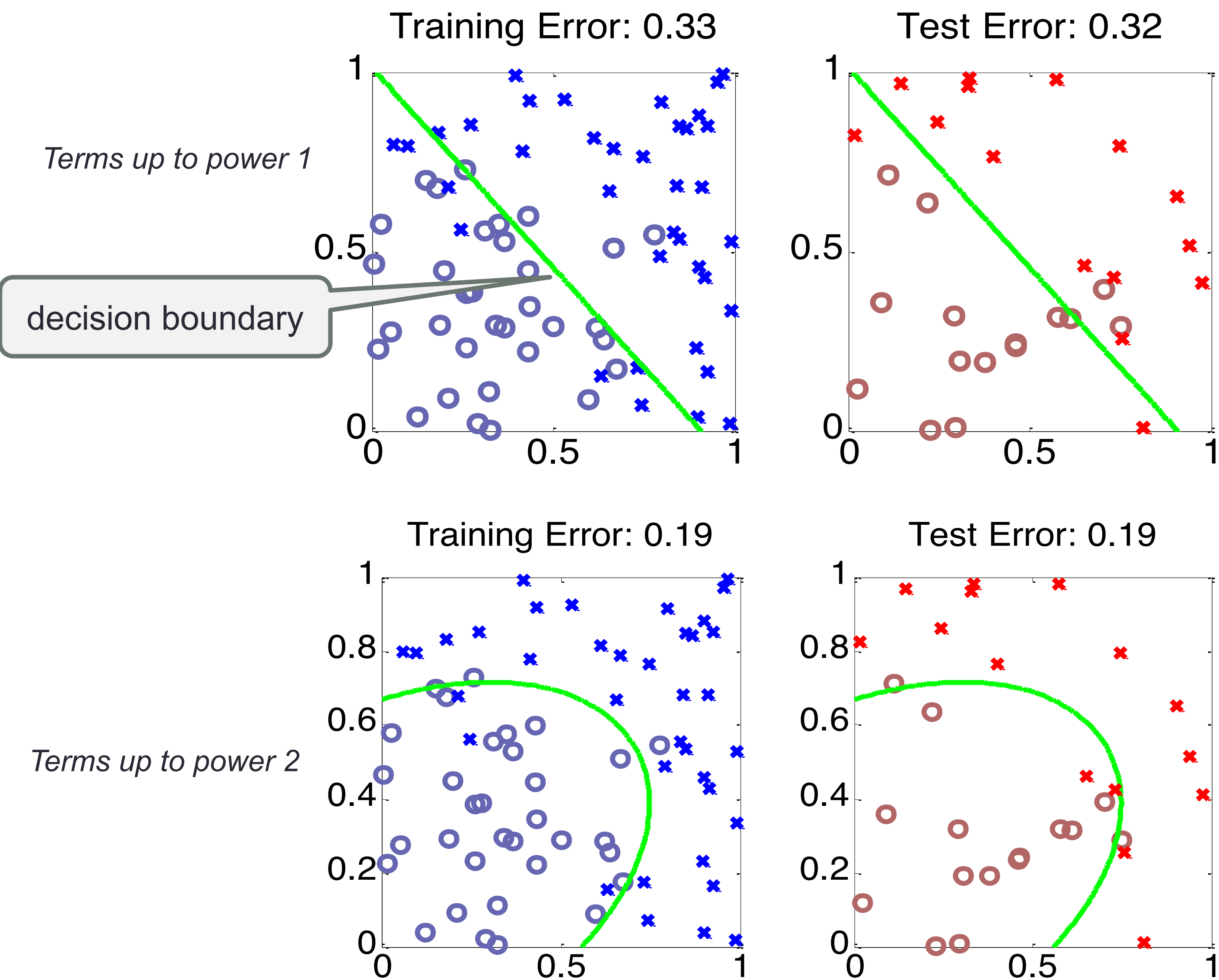


„just right“

Logistic regression with polynomial terms

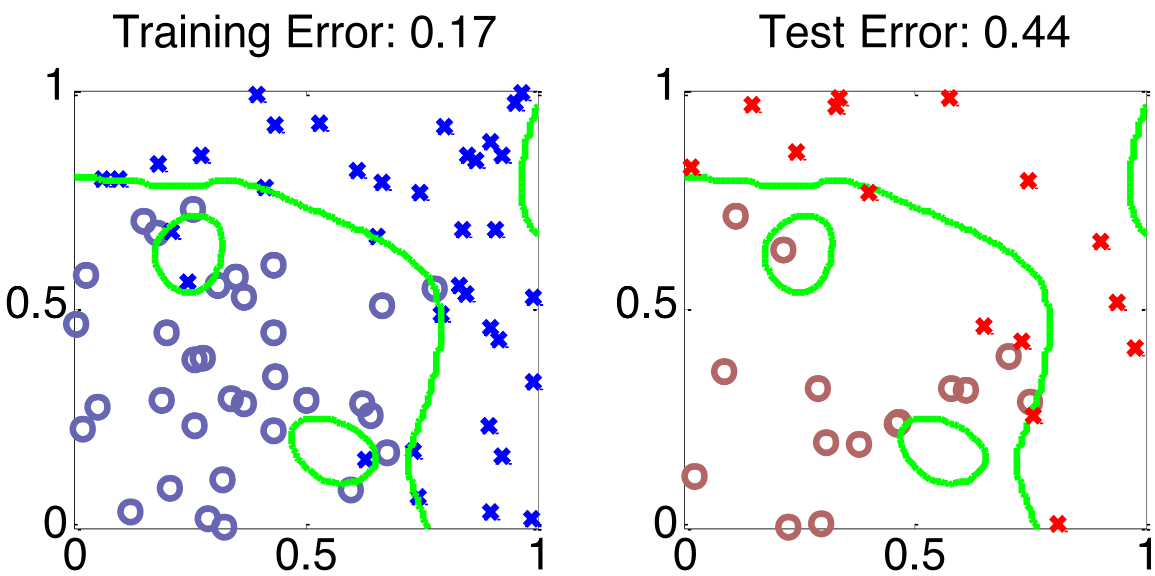


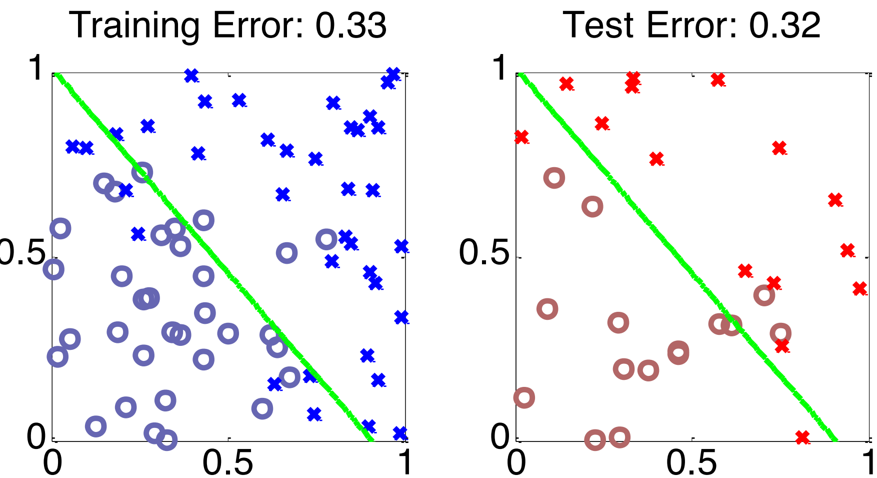
Logistic regression with polynomial terms



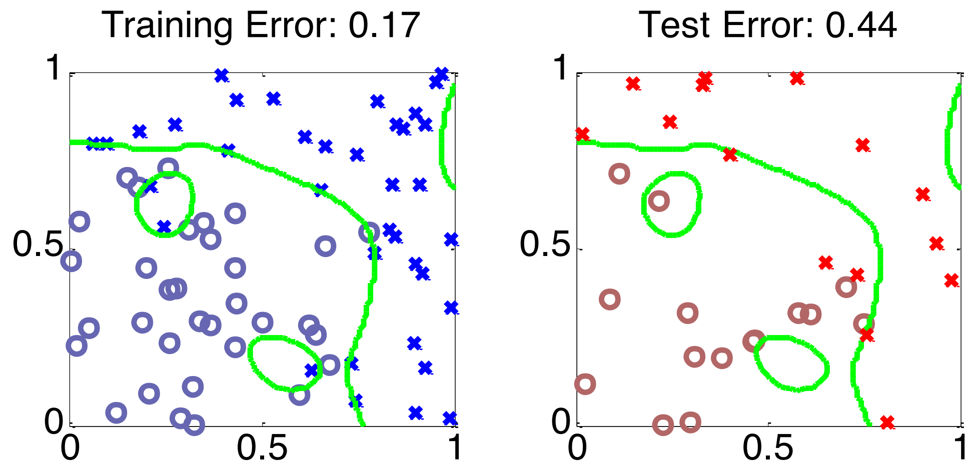
Logistic regression with polynomial terms

Terms up to power 20

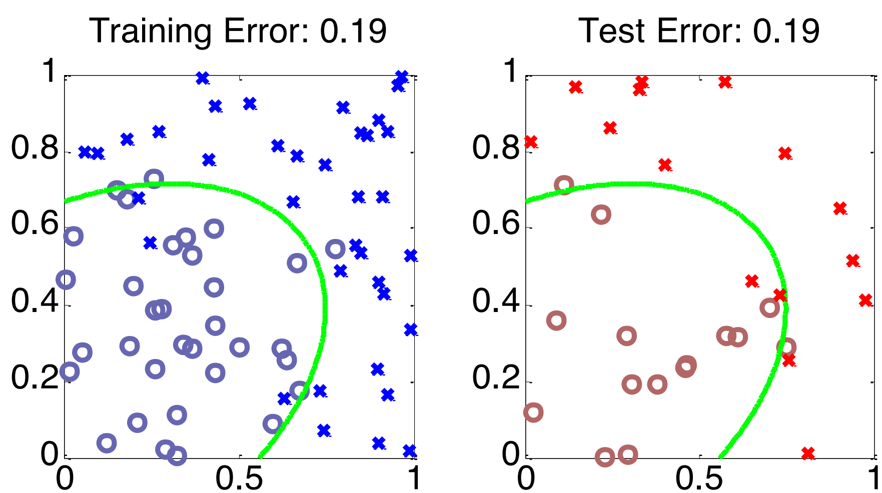




underfitting

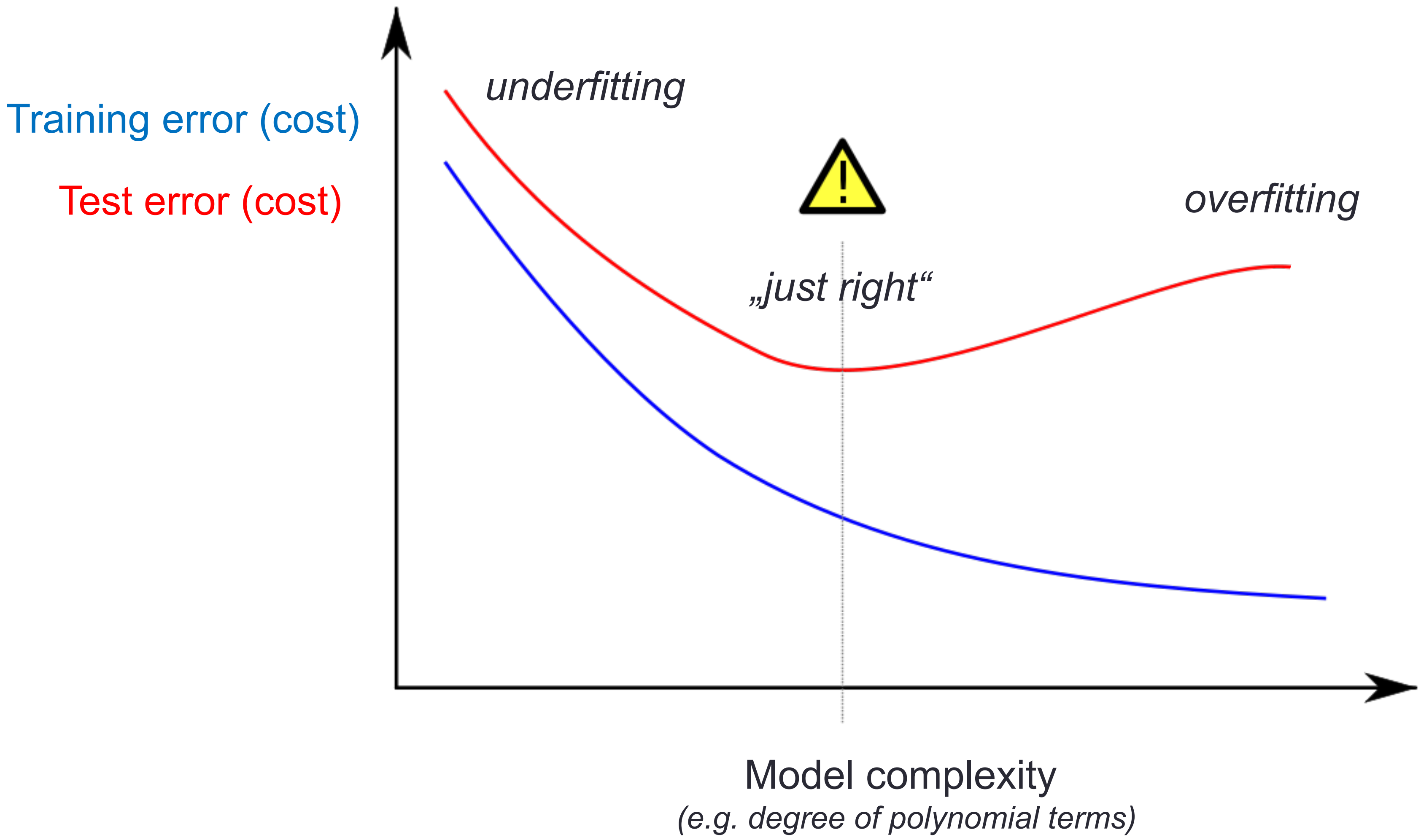


overfitting



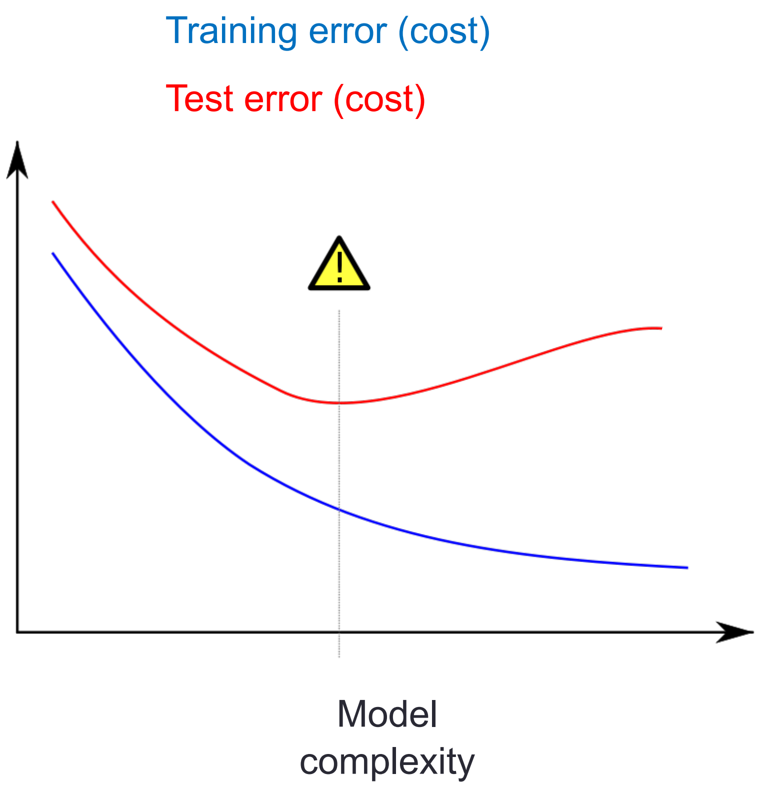
„just right“

Under-/ and Overfitting



Under- and Overfitting

- Underfitting:
  - Model is **too simple** (often: too few parameters)
  - **High training error, high test error**
- Overfitting:
  - Model is **too complex** (often: too many parameters relative to number of training examples)
  - Low training error, **high test error**
- In between:
  - Model has „right“ complexity
  - Moderate training error
  - **Lowest test error**



How to deal with overfitting

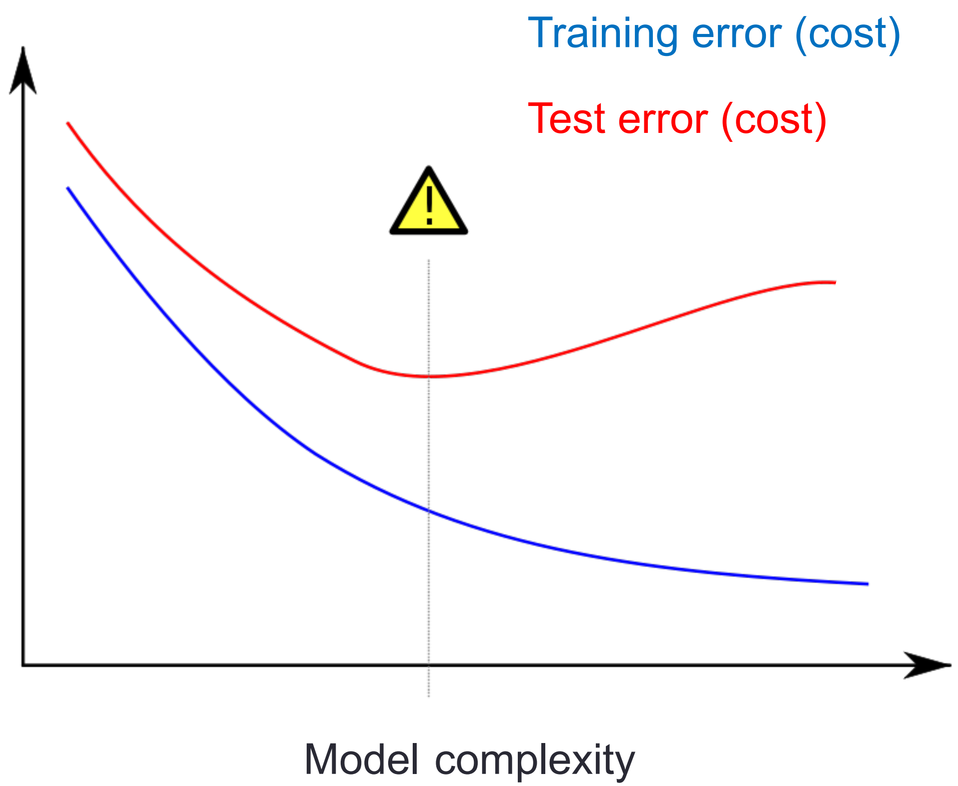
- Use **model selection** to automatically select the right model complexity
- Use **regularization** to keep parameters small *(other lecture…)*
- Collect more data
  (often not possible or inefficient)
- Manually throw out features which are unlikely to contribute
  (often hard to guess which ones, potentially throwing out the wrong ones)
- Find better features with less noise, more predictive of the output
  (often not possible or inefficient)

MODEL SELECTION

Training, Validation and Test sets

Model selection

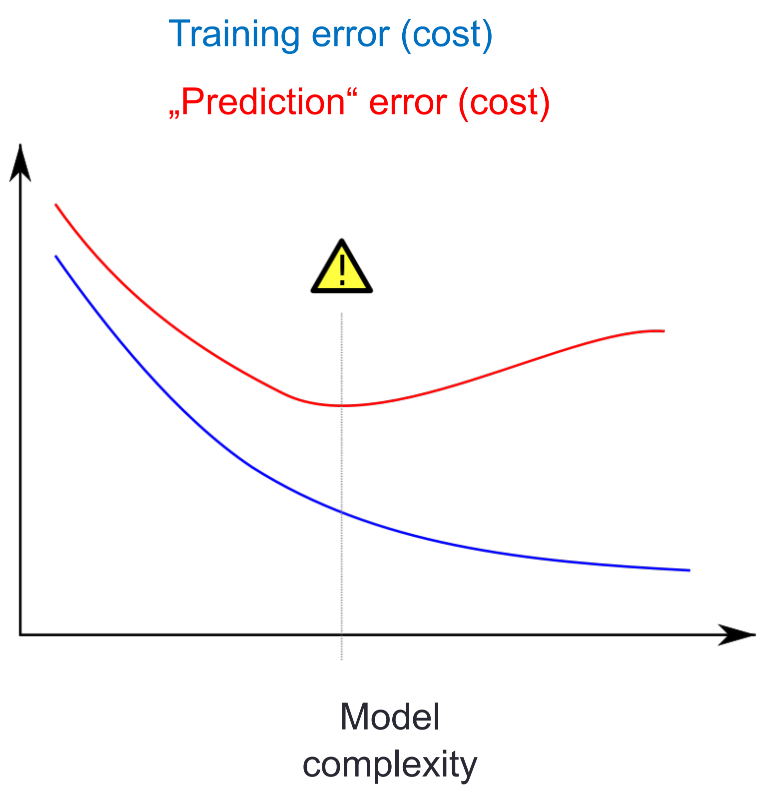
- Selection of learning algorithm and „hyperparameters“ (model complexity) that are **most suitable** for a given learning problem



Idea

- Try out different learning algorithms/variants
  - Vary degree of polynomial
  - Try different sets of features
  - …

- Select variant with best predictive performance

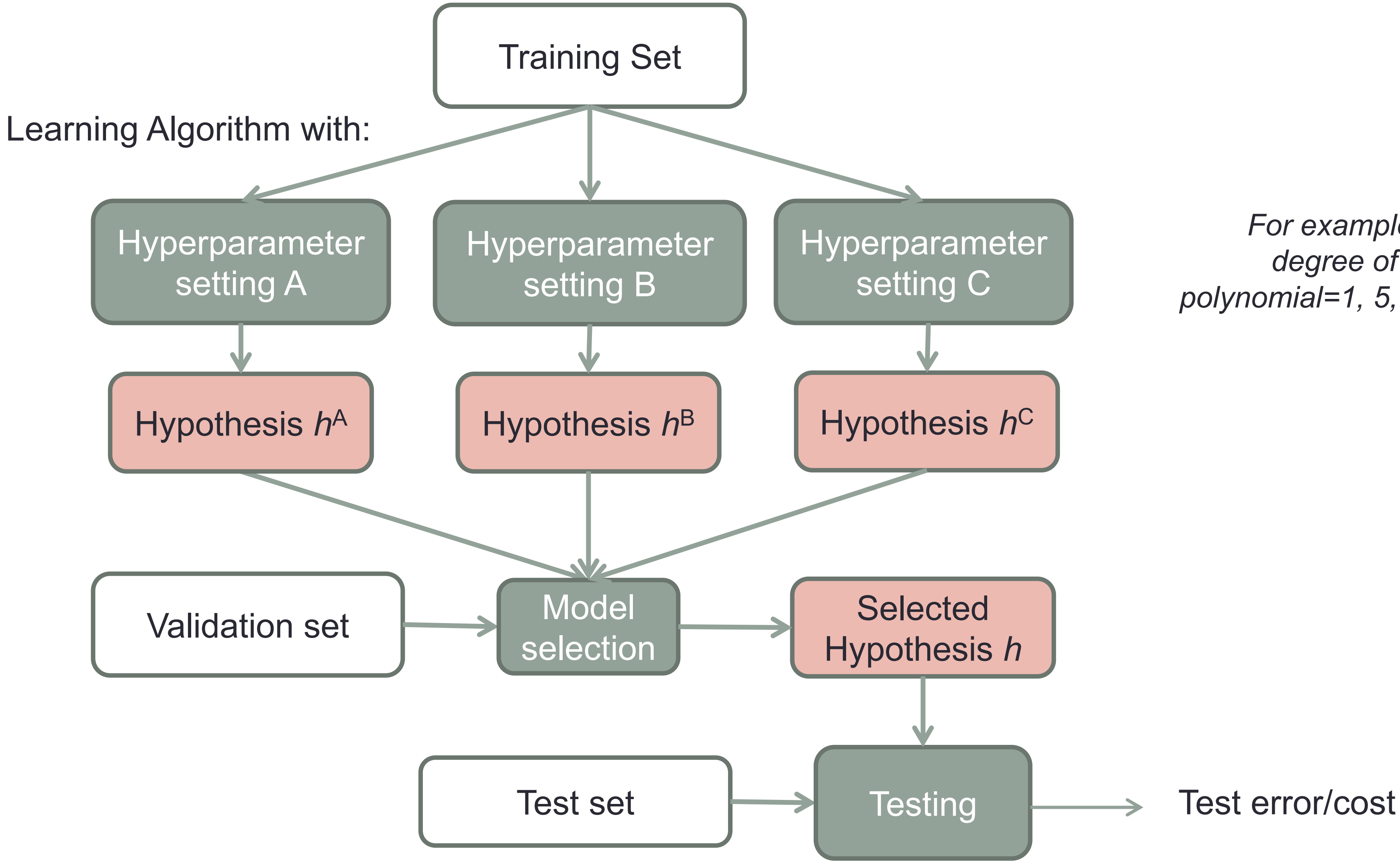


Training, Validation, Test set

- **Training set**: used by learning algorithm **to fit parameters** and find a hypothesis for each learning algorithm/variant.
- **Validation set**: used to estimate predictive performance of each learning algorithm/variant. The hypothesis with **lowest validation error** (cost) is selected.
- **Test set**: independent data set, used after learning and model selection to estimate the performance of the final (selected) hypothesis on **new (unseen) test examples**.

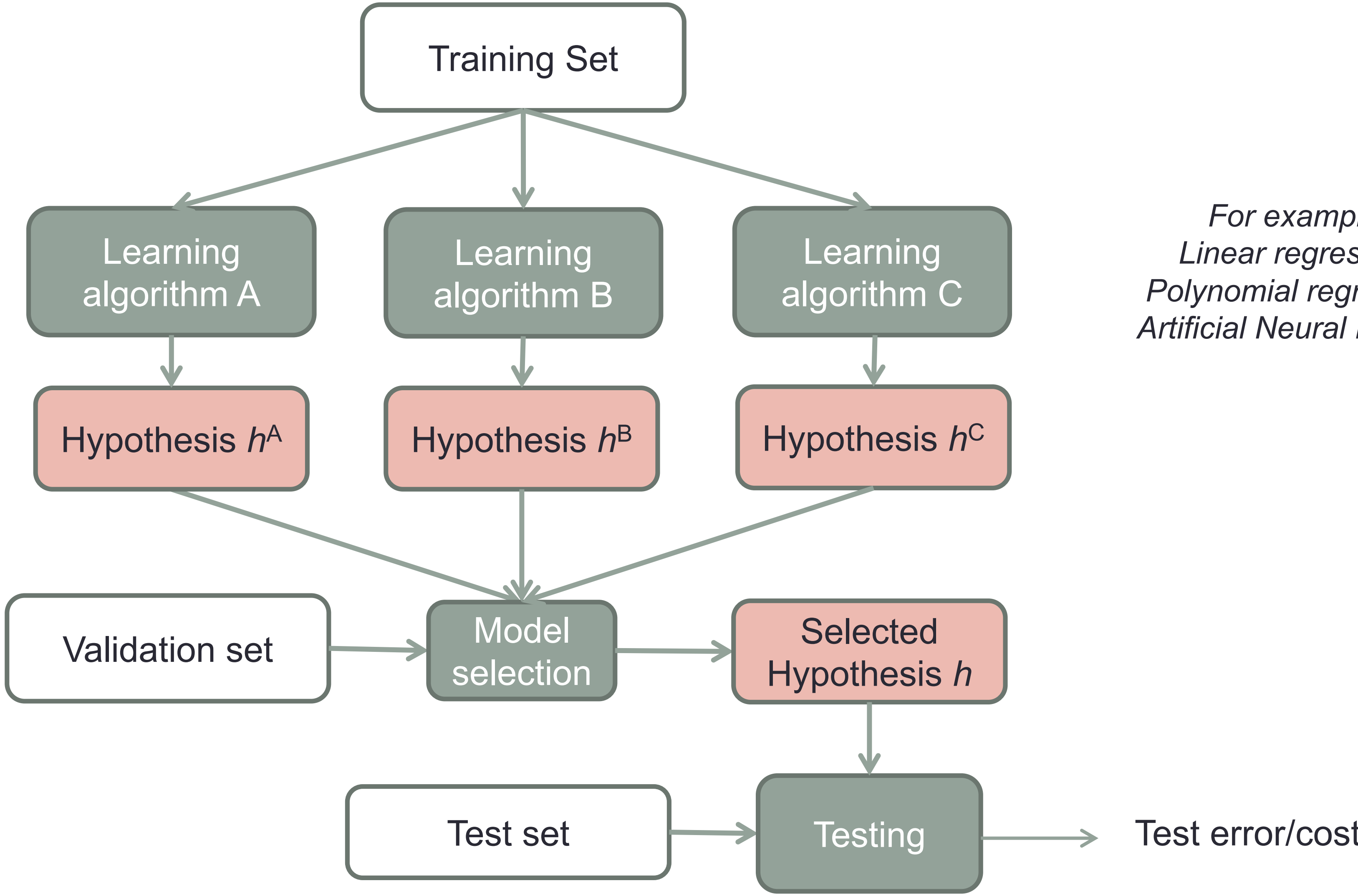
E.g. 60/20/20% randomly chosen examples from dataset. Must be disjoint subsets!

Training/Validation/Test set workflow



For example: degree of polynomial=1, 5, and 15

Training/Validation/Test set workflow



For example:
Linear regression,
Polynomial regression,
Artificial Neural Network

GRADIENT DESCENT TRICKS, AND MORE ADVANCED OPTIMIZATION TECHNIQUES

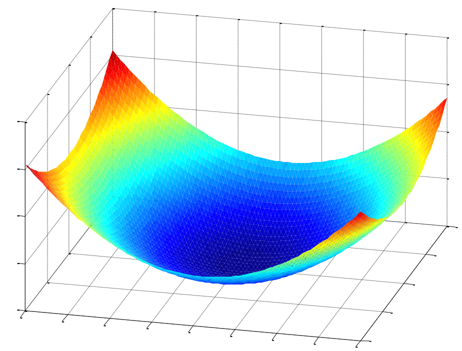
For linear regression, logistic regression, ….

Minimizing the cost via gradient descent

- Gradient descent

$$\theta_j := \theta_j - \eta \cdot \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

(simultaneous update for j=0…n)



- Gradient of logistic regression cost:

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \Big(\underbrace{h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)}}_{\text{„error“}} \Big) \cdot \underbrace{x_j^{(i)}}_{\text{„input“}}$$

(for j=0: $x_0^{(i)} = 1$)

GD trick #1: feature scaling

- Feature scaling and mean normalization
  - Bring all features into a similar range
  - E.g.: shift and scale each feature to have mean 0 and variance 1

Mean of unscaled feature

$$x_j \leftarrow \frac{x_j - \mu_j}{\sigma_j} \qquad \phi_j \leftarrow \frac{\phi_j - \mu_j}{\sigma_j}$$

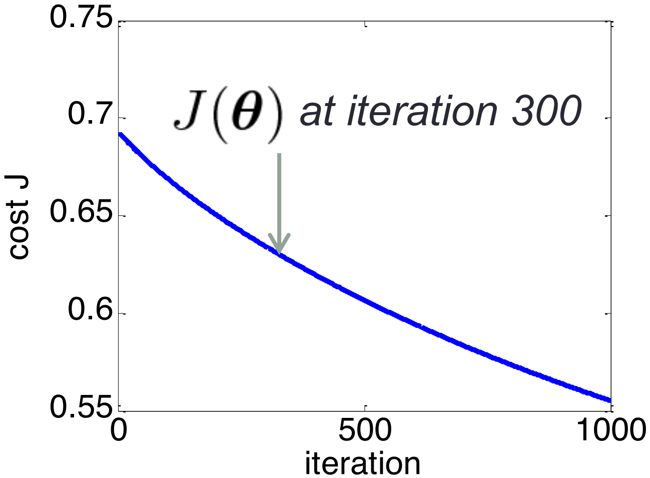
(when using non-linear features)

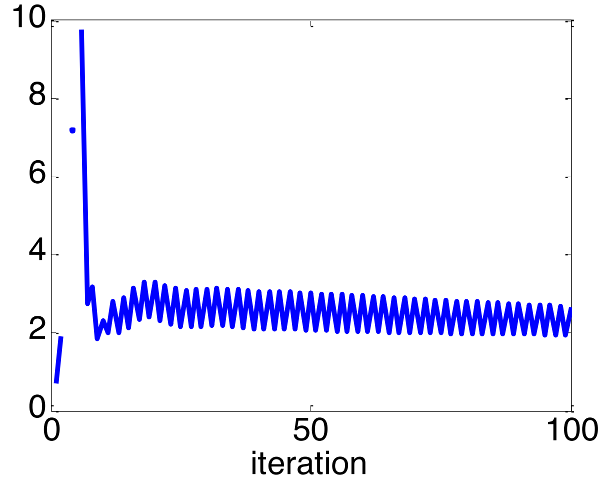
Standard deviation of unscaled feature

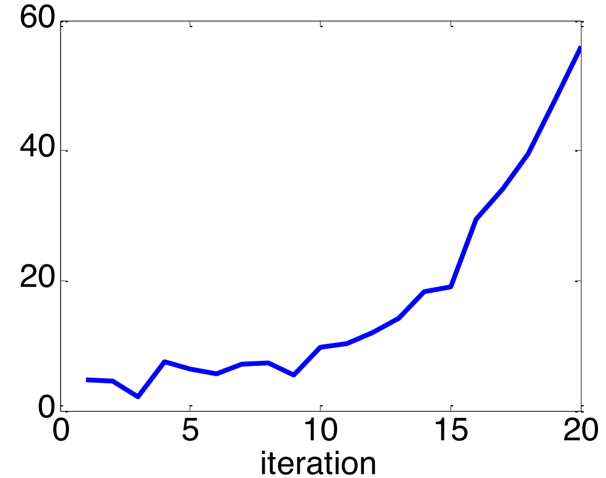
  - Do not apply to constant feature x_0 / ϕ_0 !
- Typically leads to much faster convergence

GD trick #2: monitoring convergence

- Diagnose typical issues with Gradient Descent:







… slow convergence
(increase learning rate?)

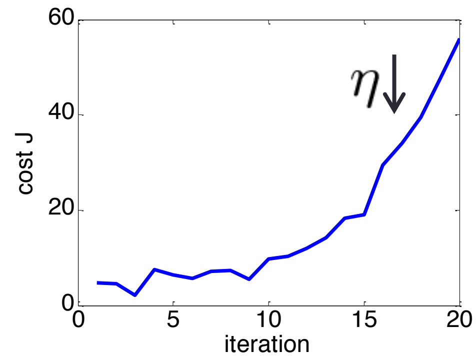
…oscillations
(decrease learning rate)

…divergence
(decrease learning rate)

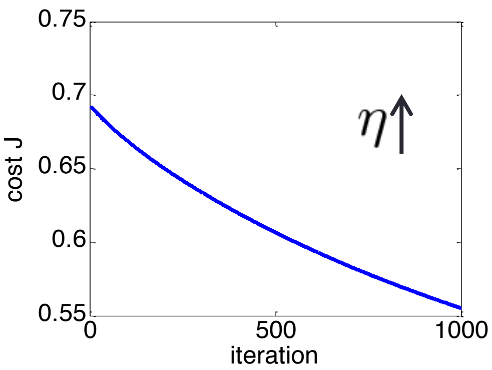
GD trick #3: adaptive learning rate

- At each iteration
  - Compare cost function value $J(\boldsymbol{\theta})$ before and after Gradient Descent update

- If cost increased:
  - Reject update (go back to previous parameters)
  - Multiply learning rate η by **0.7** (for example)



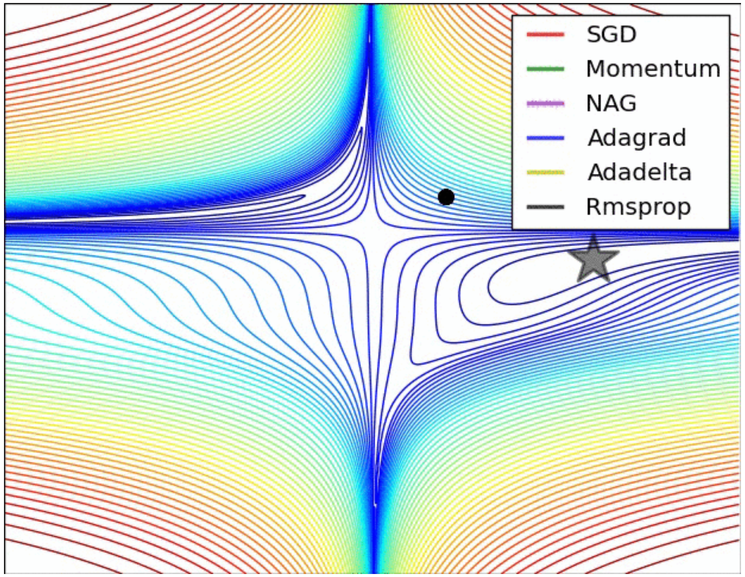
- If cost decreased:
  - Multiply learning rate η by **1.02** (for example)

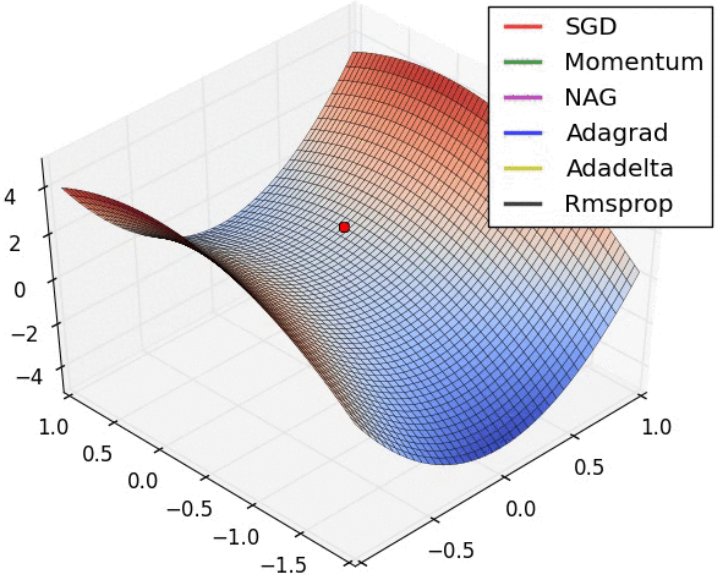


Often eliminates slow convergence and divergence issues

Variants of Gradient Descent

- SGD: Stochastic Gradient Descent
- Momentum: with momentum term
- RMSProp: adaptive learning rate

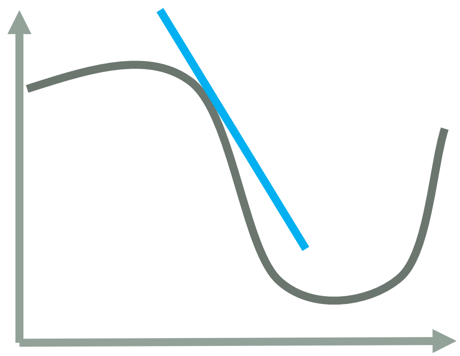


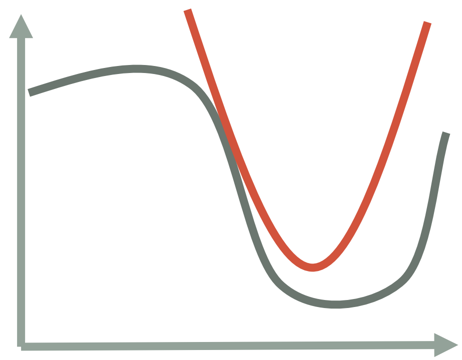


Source: http://sebastianruder.com/optimizing-gradient-descent/index.html

More advanced optimization methods

- Gradient methods = order 1 ⟶ Newton methods = order 2





  - Need **Hessian matrix** or approximations
  - Avoid choosing a learning rate
  - Conjugate gradient, BFGS, L-BFGS, …

- Tricky to implement (numerical stability, etc.)
  - Use available toolbox / library implementations!
    `scipy.optimize.minimize`
  - **Only use** when fighting for performance

SUMMARY

And questions

Some questions…

- Logistic regression is a method for … regression/classification?
- What is the hypothesis for Logistic regression?
- What‘s the cost function used for logistic regression?
- Is the cost function convex or non-convex?

- What is under-/overfitting?

- What is model selection?
- What are training, validation and test sets?
- How does model selection work (procedure)?

- What does “adaptive learning rate” mean in the context of gradient descent?

What is next?

- Neural Networks (Guillaume Bellec):
  - Perceptron
  - Feedforward Neural Network
  - Backpropagation