Machine Learning 1 – Fundamentals

Learning Theory

Prof. Dr. J. M. Zöllner, M.Sc. Nikolai Polley, M.Sc. Marcus Fechner

Overview

- Motivation
- Is learning equivalent to optimization?
  - Can learning be described formally?
  - Error minimization for empirical and real error
  - Hypothesis quality, model selection
  - Boosting, Ensembles
- Learnability and Capacity of Learning Machines
  - VC – Dimension

Principle – Ockham’s Razor

- **William of Ockham**: Important medieval philosopher and theologian, born in Ockham in 1287 and died in Munich in 1347
  - Accused of heresy
    - Revocation of teaching license
- **Occam‘s razor**
  - **Latin**: *„Entia non sunt multiplicanda sine necessitate“*
  - **German**: *„Löse nie ein Problem komplizierter als nötig, denn die einfachste, richtige Erklärung, ist die Beste“*
  - **English**: *„One should not use more entities than necessary“*
- Oftentimes interpreted as:
  “other things being equal, simpler explanations are generally better than more complex ones“
- Learning theory and successful practice aims to formalize and explain the problem and its solution. Additionally, it tries to explain "simple" and "better"

Learning System

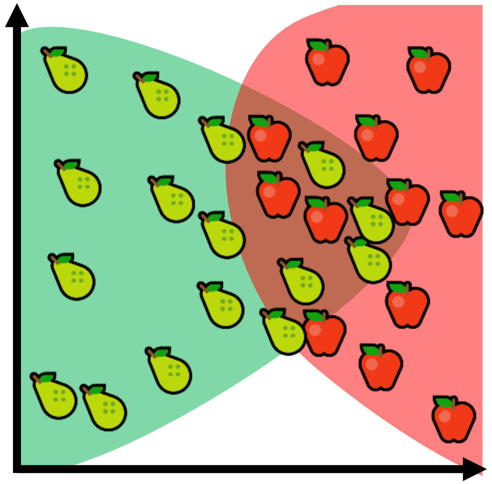
- **Learning System**: A learning system is defined by …
  - **Hypothesis Space** H with hypotheses $h_\theta \in H$, where θ represents parameters
    - **Example**: Linear model $h_\theta(x) = sgn(mx + c)$, with $\theta = (m, c)$ whereas $m \in M$ und $c \in C$
      - The set of all parameter combinations $M \times C$ spans the hypothesis space H_{lin}
      - A specific combination of parameters θ is considered a hypothesis $h_\theta \in H_{lin}$
  - **Learning method:** Find an optimal hypothesis $h_{opt} \in H$ with the help of learning examples (requires error function, optimization method, ...)

最后复习的时候对这些问题自己做个解答

- **Challenges**: How to choose a suitable learning system?
  - Which hypothesis space? Linear? Non-linear? Parametric? Non-parametric? ...
  - What learning method? What optimization? What error function? ...
  - What defines a good / optimal hypothesis? With which metrics can we measure it?

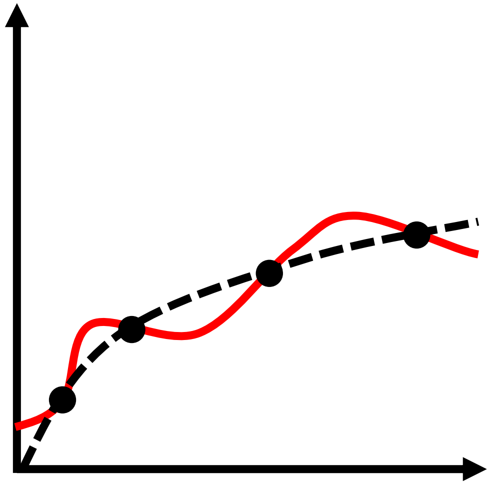
Examples

- **Example: Classification**
  - Underlying data is in $\mathbb{R}^2$
    - e.g. size and color of apples and pears
  - Non-linear separable (complexity)
  - No strict class-separation in training data (representation)
  - Good model for given data



Classification

- **Example: Regression**
  - Regression for black data points
    - Dashed black curve should be learned
    - Non-linear solvable
  - Bad model for given data (red line), too high complexity, too high polynomial



Regression

Overview

- Motivation
- Is learning equivalent to optimization?
  - Can learning be described formally?
  - Error minimization for empirical and real error
  - Hypothesis quality, model selection
  - Boosting, Ensembles
- Learnability and Capacity of Learning Machines
  - VC – Dimension

Formal Description – Supervised Learning

- **Learning**: Find optimal hypothesis $h_{opt}: X \rightarrow Y$ in hypothesis space H
  - Training dataset $\{(x_1, y_1), \ldots, (x_N, y_N)\}$
    - Input data $x_i \in X$
    - Output data $y_i \in Y$
    - Taken from (unknown) probability density function $p(x, y)$
  - Learning method: Optimization

- **Types of supervised Learning**: defined by space $X \times Y$
  - $\mathbb{R}^n \times \{y_1, \ldots, y_k\}$ - **Classification**: Output data is discrete
  - $\mathbb{R}^n \times \mathbb{R}^m$ - **Regression**: Output data is continuous
  - $\{Atr_1, .., Atr_n\}^n \times \{true, false\}$ - **Concept**: Input is discrete, output true/false
  - ... (see Deep Learning book by Ian Goodfellow])

Error Minimization

- **Goal**: Find optimal hypothesis h_{opt} by minimizing cost function $\mathcal{L}(h_\theta)$

- **Loss function**: $\ell(h_\theta(x_i), y_i)$
  - Error/mismatch between predicted output $h_\theta(x_\mathrm{i})$ of hypothesis and the target output y_i for single instance in dataset

- **Cost function / Risk:**

$$\mathcal{L}(h_\theta) = \mathbb{E}_{(x,y)\sim p}[\ell(h_\theta(x), y)] = \int \ell(h_\theta(x), y)\, p(x, y)\, dx\, dy$$

  - Expected loss of **all data** in (unknown) probability density function $p(x, y)$
  - Also called *generalization error* or *generalization risk*

Error Minimization

- **Challenge:** what kind of loss functions ℓ or cost function $\mathcal{L}$ should be used?
  - Misclassification $\ell(h_\theta(\mathrm{x}), y) = \begin{cases} 1 & \text{if } h_\theta(x) \neq y \\ 0 & \text{else} \end{cases}$
  - Absolute Error $\ell(h_\theta(\mathrm{x}), y) = |h_\theta(\mathrm{x}) - y|$
  - Quadratic Error $\ell(h_\theta(\mathrm{x}), y) = (h_\theta(\mathrm{x}) - y)^2$
  - (Binary) Cross Entropy $\ell(h_\theta(x), y) = -[y \cdot \log h_\theta(x) + (1-y)\log(1-h_\theta(x))]$
  - Rule of thumb:
    - Loss is always a single positive number greater or equals zero: $\ell(h_\theta(x), y) \geq 0$;
    - Classification tasks: usually cross entropy is used
    - Regression tasks: usually quadratic error is used

Error Minimization

- **Problem**: Usually $\mathcal{L}(h_\theta)$ can‘t be calculated!
  - To calculate we need all instances in probability function $p(x, y)$
    - Quick example: Image classification
      - $p(x, y)$ contains all images that could ever exist.
      - For images with resolution 1920×1080 there are $\sim 10^{15{,}000{,}000}$ unique images in probability function
        - We need all these images and their corresponding y...
  - Generally, it is impossible to calculate $\mathcal{L}(h_\theta)$ in finite time
  - Our limited size training dataset defines the empirical distribution $\hat{p}(x, y)$, and is only a very small subset of true probability function $p(x, y)$

- **Approximate $\mathcal{L}(h_\theta)$!** ➡ **Empirical Risk Minimization** 经验风险

Empirical Risk Minimization

- Empirical distribution $\hat{p}(x,y)$ is defined by limited dataset $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$
- **Empirical Risk / Empirical Error**:

$$\hat{\mathcal{L}}_{\mathrm{D}}(h_\theta) = \mathbb{E}_{(x,y)\sim\hat{p}}[\ell(h_\theta(x), y)] = \frac{1}{|\mathrm{D}|} \sum_{(x,y)\in D} \ell(h_\theta(\mathrm{x}), y)$$

  - Calculate **expected error of estimated data distribution** $\hat{p}(x,y)$ **instead of actual error** of underlying distribution $p(x,y)$!

- **Consequently:** instead of directly minimizing the real risk $\mathcal{L}(h_\theta)$, we minimize the empirical risk $\hat{\mathcal{L}}_{\mathrm{D}}(h_\theta)$ and <u>hope</u> that the $\mathcal{L}(h_\theta)$ decreases to the same extent

- The difference $\mathcal{L}(h_\theta) - \hat{\mathcal{L}}_{\mathrm{D}}(h_\theta)$ is called **generalization gap** 泛化误差
  - Cannot be calculated because $\mathcal{L}(h_\theta)$ is still unknown
- Approximate generalization gap by splitting dataset D into training, validation and test

Data Splits: Training, Validation, Test

All Data

Data used for finding optimal hypothesis | Unseen data

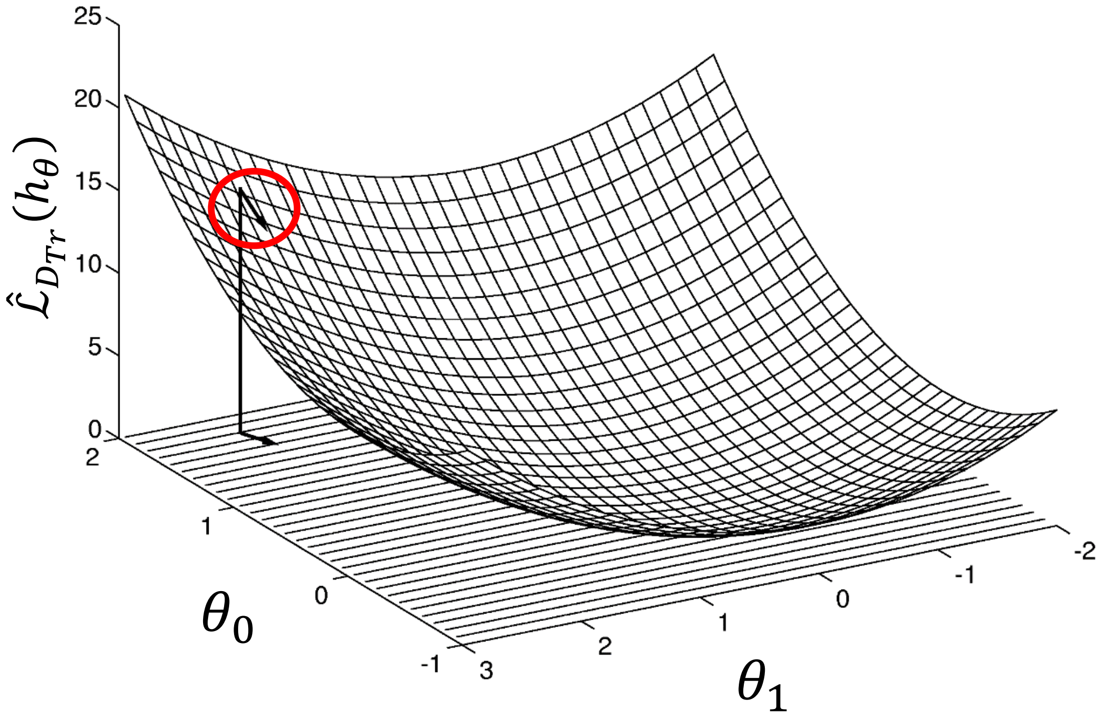
Training Dataset D_T	Validation Dataset D_V	Test D_T
Model training: find best hypothesis/parameters	Evaluate different training-hypotheses	Final model evaluation
ca. 70% of data	ca. 20% of data	ca. 10% of data
Train Error	Validation Error	Test Error

- **Usually:** Use training error to calculate empirical risk and use difference between test- and training error to calculate generalization gap
- **Problem**: Empirical risk and generalization gap is dependent of the specific split of the data. Different splits result in different values, especially with small datasets.

Learning: Minimizing Errors with Optimization

One Solution:

- Define initial (random) hypothesis h_θ
- Find best θ_{opt} via iterative minimization of empirical train error $\hat{\mathcal{L}}_{D_{Tr}}(h_\theta)$
- e.g.: gradient descent on empirical cost function of training dataset



Gradient: $\nabla \hat{\mathcal{L}}_{D_{Tr}}(\theta) \equiv \left[\frac{\partial \hat{\mathcal{L}}_{D_{Tr}}(h_\theta)}{\partial \theta_1}, \ldots, \frac{\partial \hat{\mathcal{L}}_{D_{Tr}}(h_\theta)}{\partial \theta_n}\right]$ D

Error Minimization with Gradient Descent

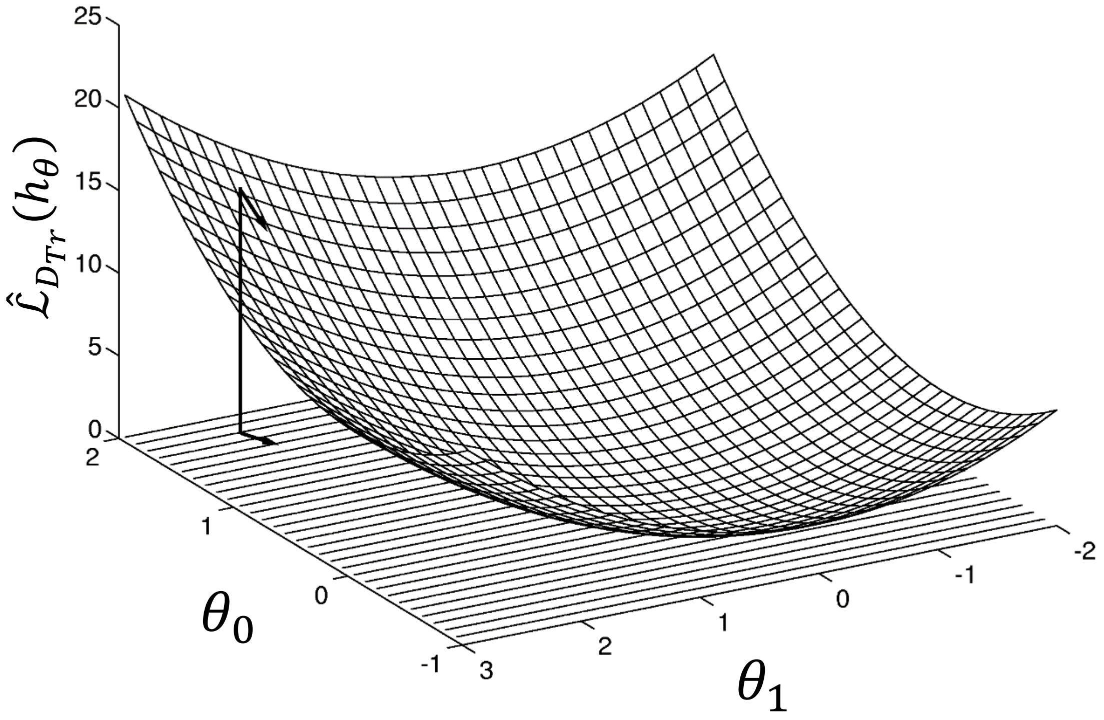
- Adjustment of parameters using gradient descent:

  $$\theta \leftarrow \theta + \Delta\theta$$

  $$\Delta\theta \approx -\eta \nabla \hat{\mathcal{L}}_{D_{Tr}}(\theta) \qquad \eta = learning\ rate$$

  - Hypothesis needs to be differentiable to calculate the gradient
- Does this work?

  WELL… YES … at least oftentimes
- Remember: We minimize $\hat{\mathcal{L}}_{D_{Tr}}(\theta)$ and ***hope*** that $\mathcal{L}(h_\theta)$ is also minimized simultaneously.
  - If this is not the case → large generalization gap
    - Observed with $\hat{\mathcal{L}}_{D_{Test}}(h_\theta) > \hat{\mathcal{L}}_{D_{Tr}}(h_\theta)$
  - An often-occurring problem for minimization is overfitting! (See later)



Learning Challenges

Statistical Problem

- The learning system considers a hypothesis space that is „too large“ in terms of the amount of training data.
- Based on the training data, several hypotheses are equally suitable.

Complexity Problem

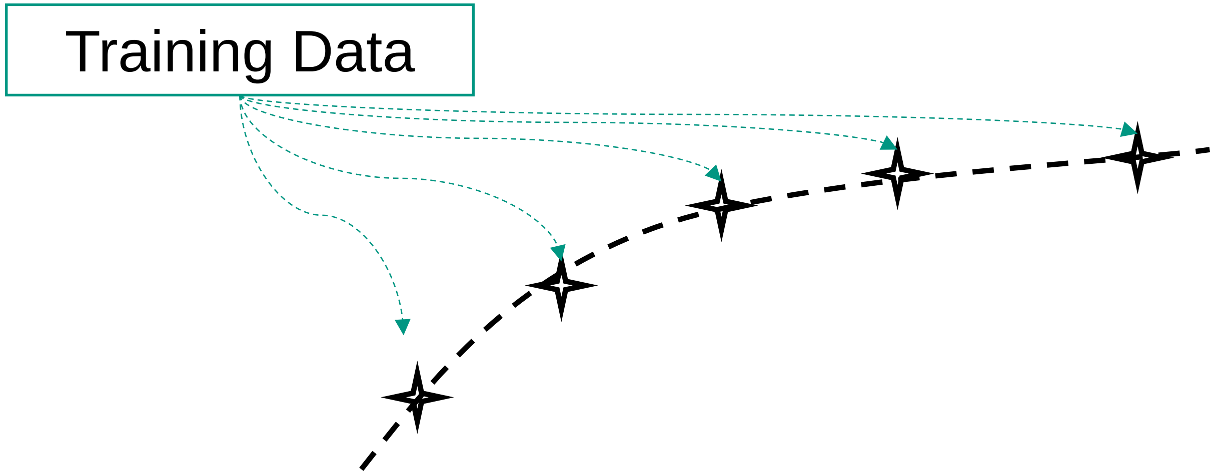
- Due to the complexity of the problem, the learning process cannot find an optimal solution within the hypothesis space, although it theoretically exists in hypothesis space.
- Risk of a suboptimal solution.

Representation Problem

- The hypothesis space does not contain sufficiently good approximations of the objective function/concept etc...
- The learning method cannot provide the desired degree of approximation.

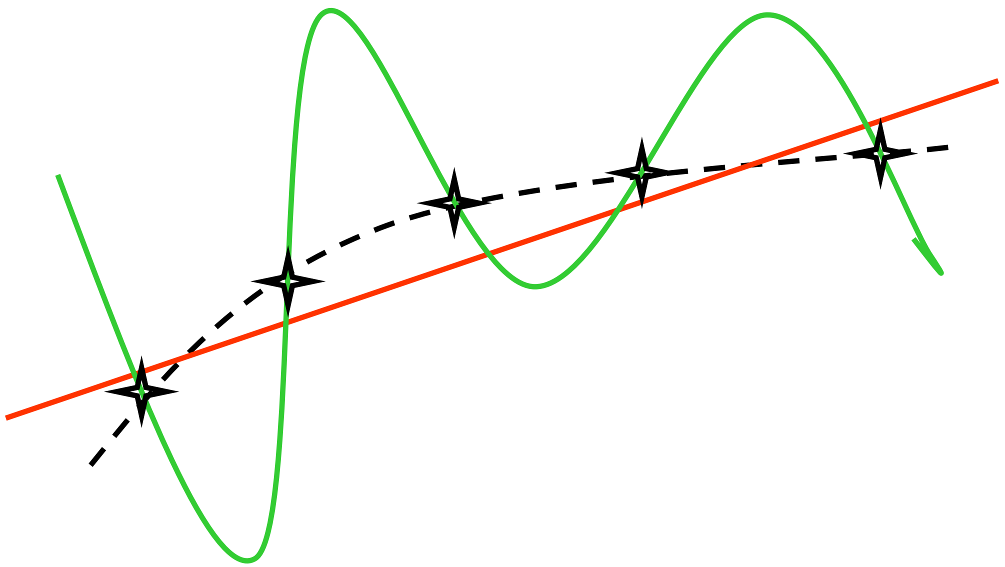
Complexity / Capacity

- Example (Regression model): Optimal hypothesis is dashed line.



- Learning system should find/approximate the optimal hypothesis given the training data.

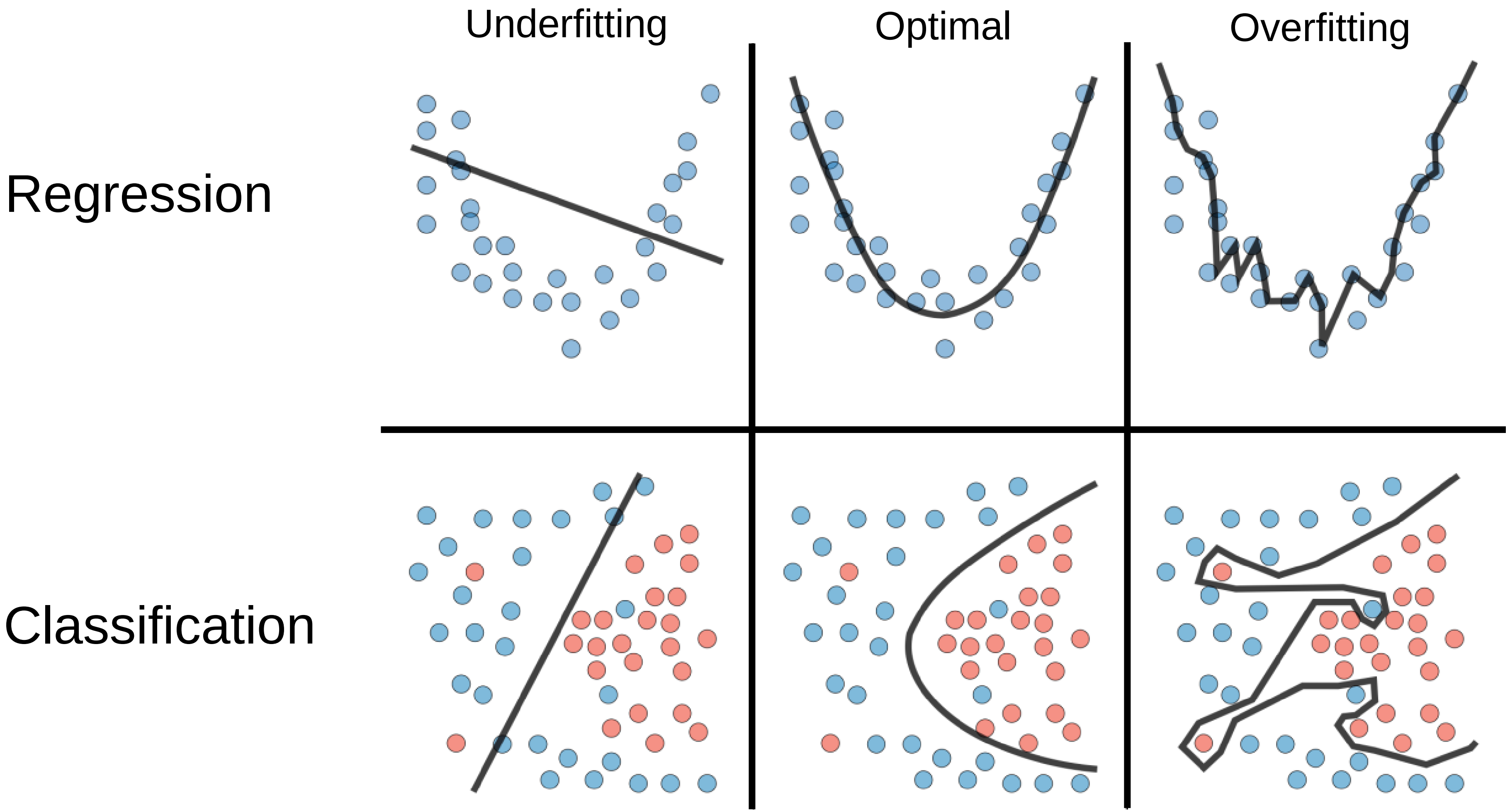
Complexity / Capacity



- System with high capacity
- Hypothesis space spanned by „many parameters“.
  - Can sometimes be calculated with VC Dimension
- Unsuitable model (why?): **Overfitting**

- System with low capacity
- Hypothesis space spanned by „few parameters“
- Unsuitable Model (why?): **Underfitting**

Overfitting - Underfitting



Overfitting formal

- **Definition:** A hypothesis overfits the training examples, if some other hypothesis, that fits the training examples less well, -performs better over the entire distribution of instances
  - Learning system memorizes training data instead of learning the underlying structure

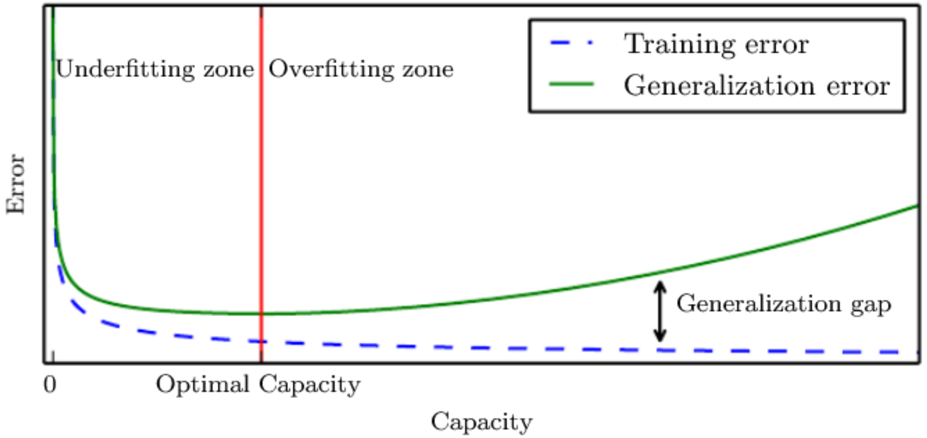
- Formal definition:

$$h \in H \text{ overfitting} \Leftrightarrow \ \exists h' \in H \text{ such that given } D_{Tr} \text{ and } D_V$$
$$\hat{\mathcal{L}}_{D_{Tr}}(h) < \hat{\mathcal{L}}_{D_{Tr}}(h') \wedge \hat{\mathcal{L}}_{D_V}(h) > \hat{\mathcal{L}}_{D_V}(h')$$

  - Whereas:
    - D_{Tr} – Training Data
    - D_V – Validation Data

Reasons for Overfitting

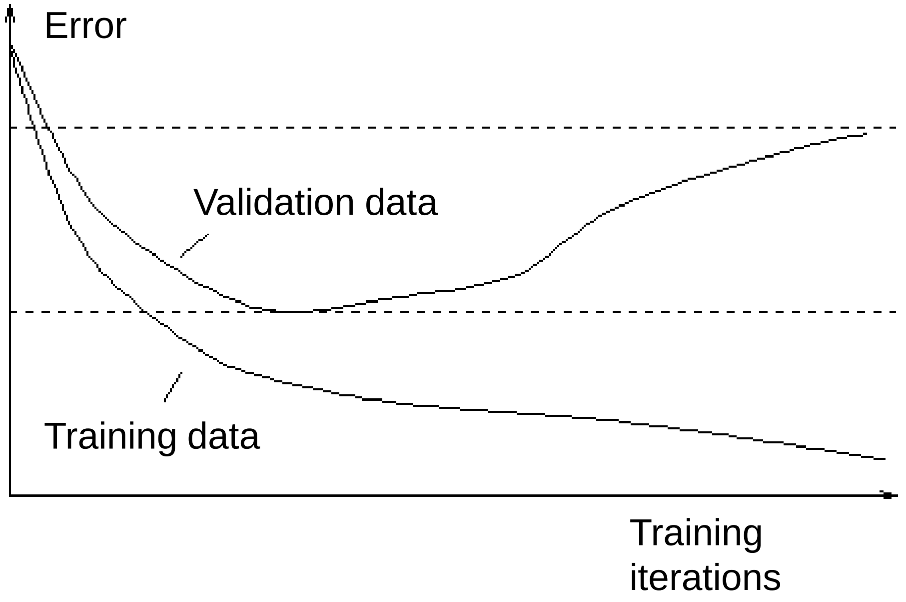
- Model capacity is too large
  - Hypothesis space contains hypotheses that perfectly fit training data but do not correlate to underlying structure
  - Even if a generalized hypothesis exists in hypothesis space, the overfit hypothesis is preferred by learning system because it additionally fits noisy/wrong data.



- Model is trained for too many iterations
  - After a while, the system optimizes the loss by finding (wrong) reasons why noisy/mislabeld/outlier data should be fitted a certain way.

Detect Overfitting

- Track error during training for training / validation dataset
  - Initially: both errors decrease
  - After some iterations: training error keeps decreasing while validation error increases
  - **→ Model generalization capability decreases**

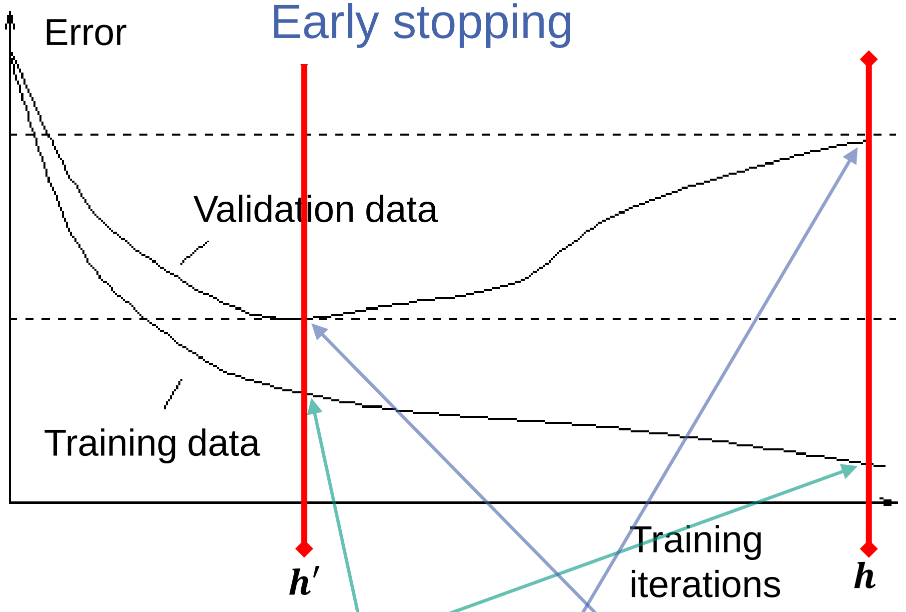


Explanation

Data is different in training and validation set → Different hypotheses minimize $\hat{\mathcal{L}}_{D_{Tr}}$ and $\hat{\mathcal{L}}_{D_V}$. In training, only $\hat{\mathcal{L}}_{D_{Tr}}$ is minimized and at a certain point it is easier for the model to start memorizing the training data (including outlier/noisy/mislabeled datapoints) instead of learning the underlying structure which in turn increases $\hat{\mathcal{L}}_{D_V}$

Solutions for Overfitting

- Representative instances in training dataset (increase number and types of instances)
- Steer learning with validation error, e.g. early stopping.
  - (Afterwards, validation can no longer be used for performance ~ generalization quality)
- Decrease model capacity
- Correct choice and search for optimal hypothesis h_θ



$h \in H$ overfitting $\Leftrightarrow \exists h' \in H$ such that given D_{Tr} and D_V
$\hat{\mathcal{L}}_{D_{Tr}}(h) < \hat{\mathcal{L}}_{D_{Tr}}(h') \wedge \hat{\mathcal{L}}_{D_V}(h) > \hat{\mathcal{L}}_{D_V}(h')$

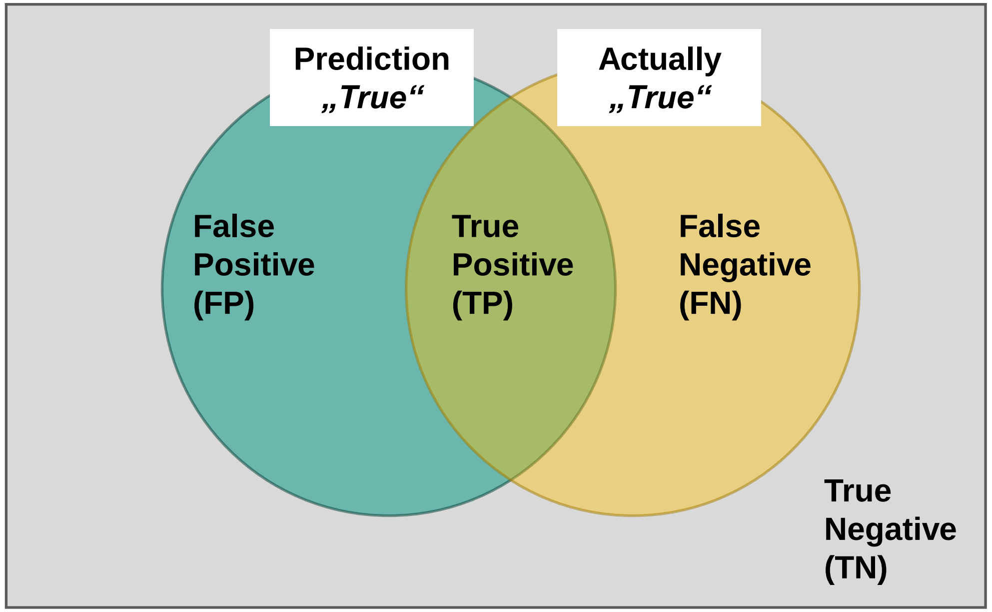
Determining Model Quality

- Metrics: To describe how (in)accurate a system is suited to a task
  → not necessarily identical to error $\hat{\mathcal{L}}$
  → choose carefully

- Different methods depending on the task at hand
  - Classification: e.g. How often is something classified right or wrong?
  - Regression: e.g. How close are you to what you want to approximate?
  - Unsupervised learning: e.g., how well is the data mapped?
    - Generally difficult to measure
  - Reinforcement Learning: Indirectly via the process, How well does the sequence of actions fit?
    - Generally, no general metrics, for individual components (see above)
  - ...

Classification: True vs. False Positive vs. Negative

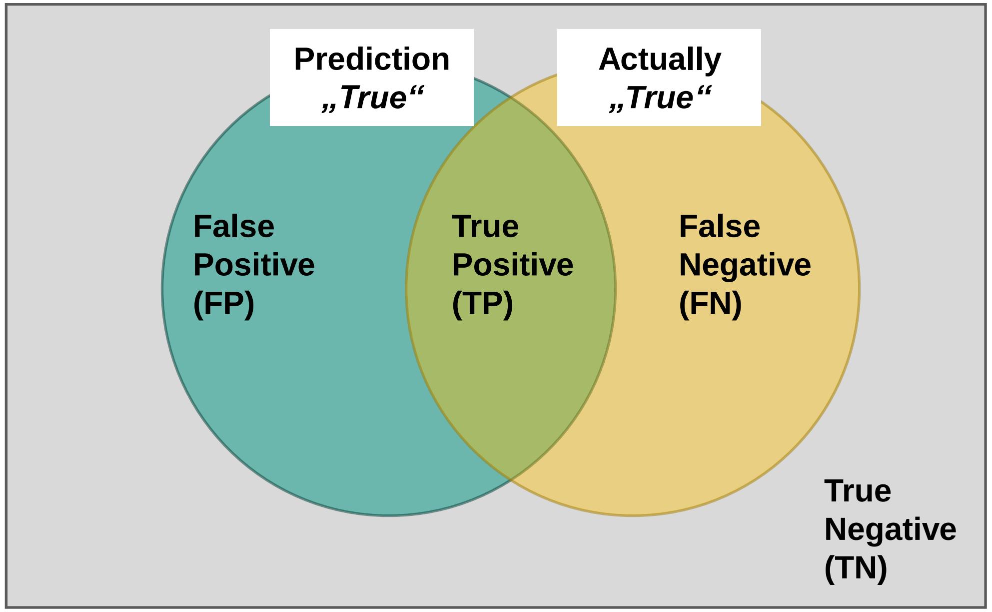
- Differentiation between 4 classification outcomes
  - Correct classification of positive instances
    True Positive (TP)
  - Correct classification of negative instances
    True Negative (TN)
  - False classification of positive instances:
    False Positive (FP)
  - False classification of negative instances
    False Negative (FN)



Classification: True vs. False Positive vs. Negative

- Example: The boy who cried wolf…
  - *wolf present*: positive class
  - *wolf absent*: negative class

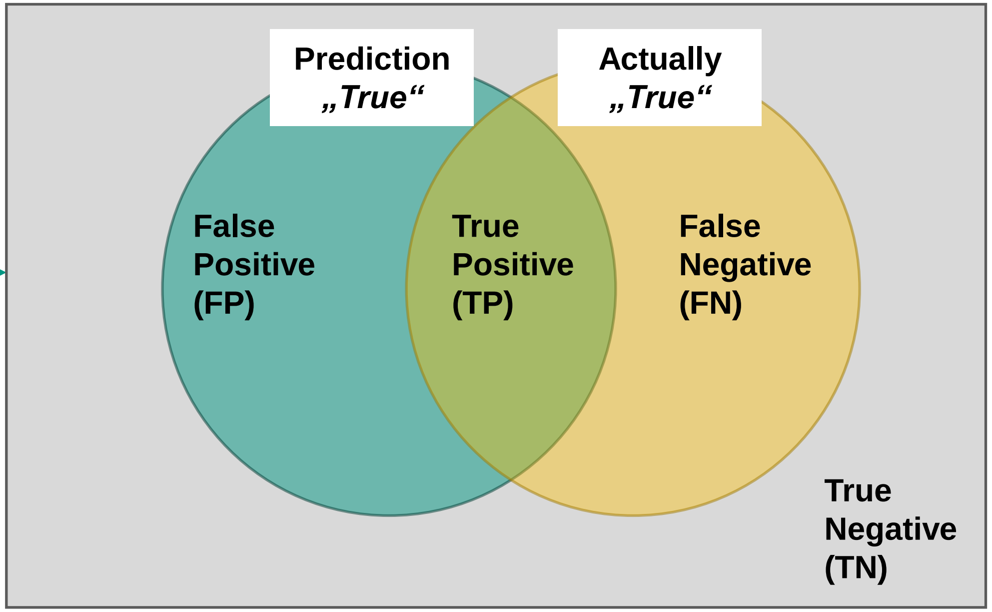
  - TP: Boy cries wolf, wolf is present
    - Sheeps are save
  - TN: Boy doesn‘t cry wolf, wolf is absent
    - Sheeps are save
  - FP: Boy cries wolf, wolf is absent
    - Neighbours are angry
  - FN: Boy doesn‘t cry wolf, wolf is present
    - Sheeps are eaten



Classification: Confusion Matrix

- Confusion matrix for 2 classes

		Prediction	
		Yes	**No**
Class	**Yes**	TP	FN
	No	FP	TN

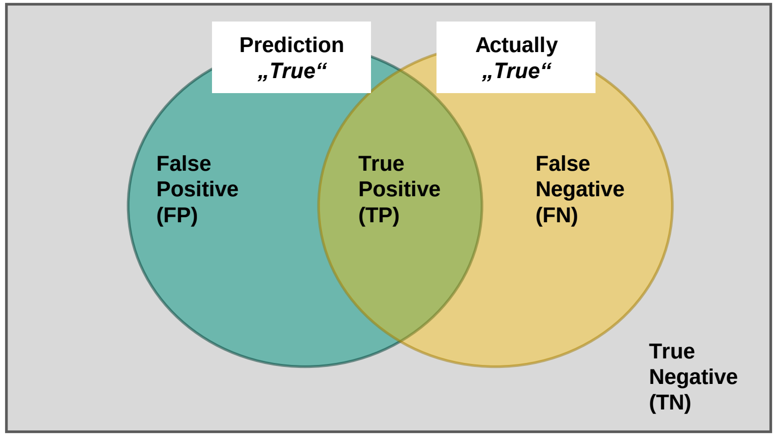


- Generalization to multiclasses possible
- Desired Result:
  - High values on the diagonal
  - Low FP and FN

Metrics: Accuracy and Error Rate

- **Accuracy** *(as high as possible)*

$$\text{Acc} = \frac{correct}{total} = \frac{TP + TN}{TP + FN + FP + TN}$$



- False positive rate **FPR**, *(as small as possible)*

$$FPR = \frac{FP}{FP + TN}$$

- False negative rate **FNR** *(as small as possible)*

$$FNR = \frac{FN}{TP + FN} = 1 - TPR$$

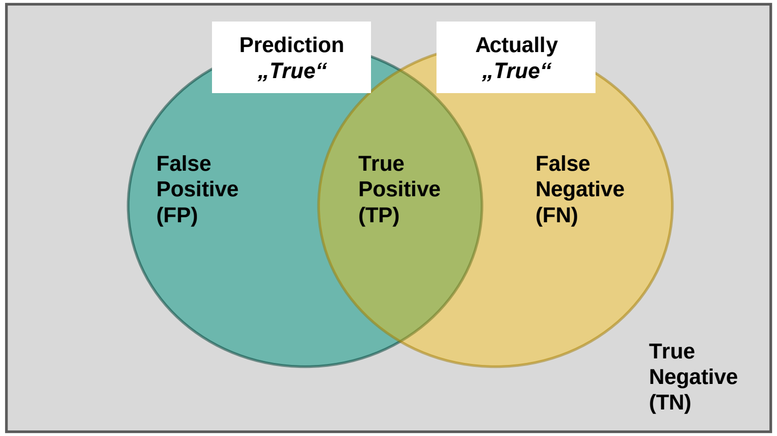
Metrics: Recall und Precision

- **Precision** *(as high as possible)*

$$P = \frac{TP}{TP + FP}$$

- **Recall** *(as high as possible)*

$$TPR = R = \frac{TP}{TP + FN}$$



- **F1 – Score**: harmonic mean of precision and recall *(as high as possible)*

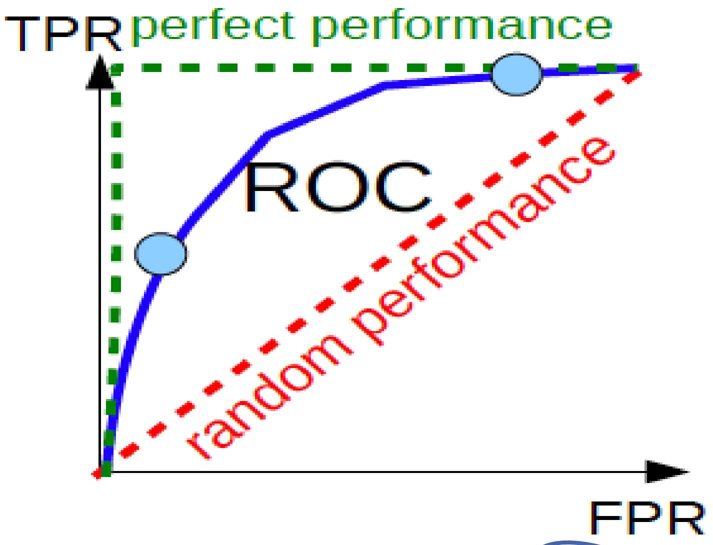
$$F_1 = \frac{2}{\frac{1}{R} + \frac{1}{P}}$$

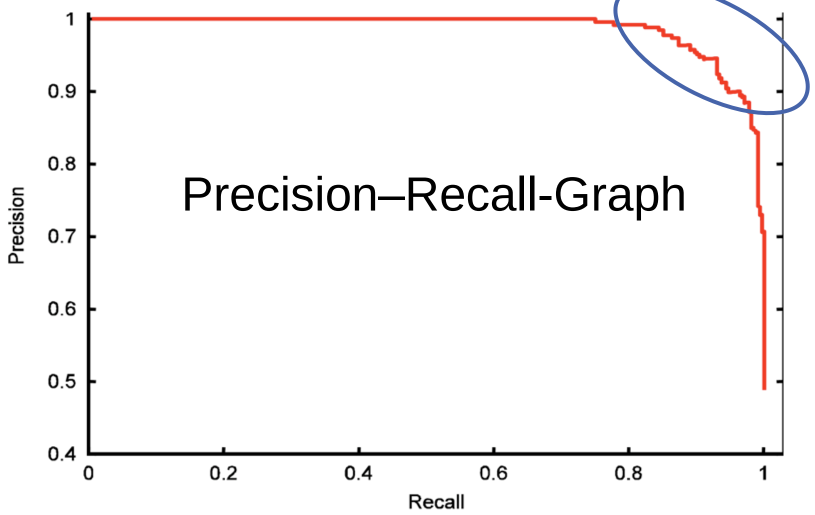
We won't explicitly ask these formulas in the exam, but nonetheless it doesn't hurt to know them

Find Best Model of a Learning System

- Combine multiple metrics to determine best hypothesis from multiple hypotheses of learning system

- Approaches
  - TPR/FPR-Graph
    (Receiver-Operating Characteristic, ROC)
  - Precision-Recall-Graph





Determine Model Quality – Data-Dependent

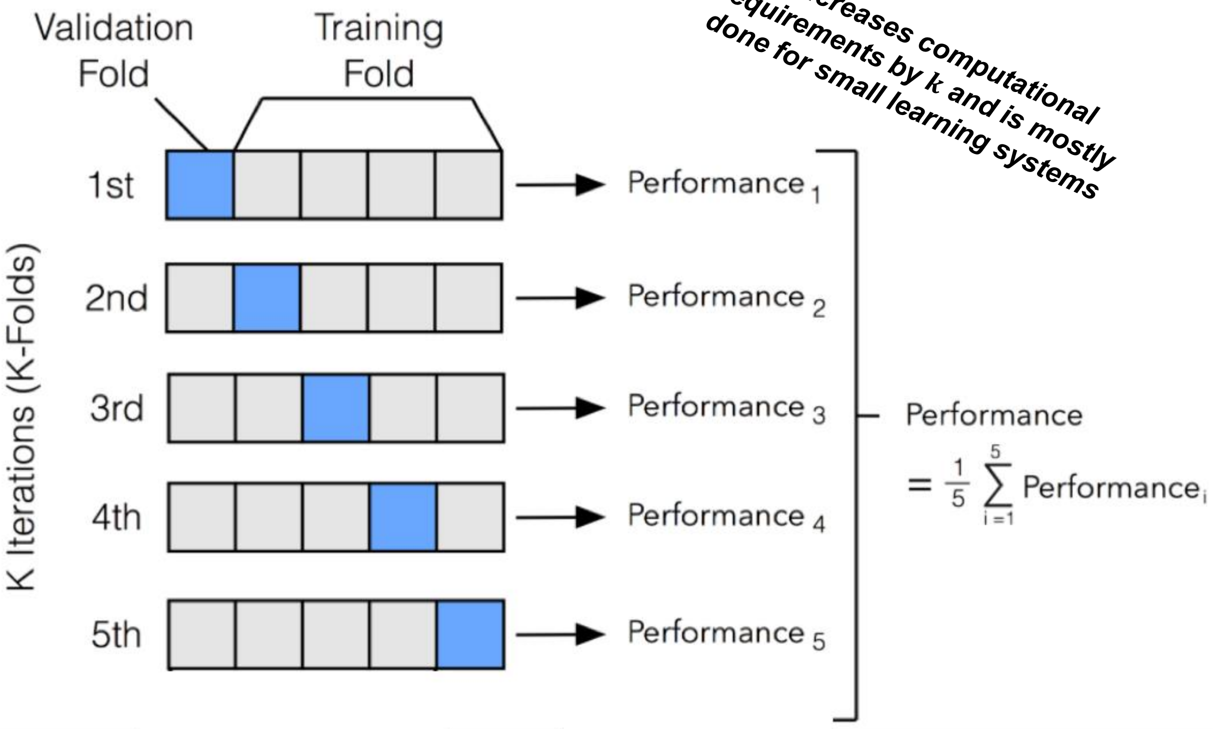
- Splitting learning data into three datasets drastically reduces the data we can use to train/evaluate the system
  - Validation and test dataset can not be used for training
  - A learning systems might have large variances in performance, depending on the specific split of the data

- One Solution: Cross-Validation
  - Repeatedly partition data into training and validation data
  - Then determine good hypotheses (or parameters of the learning machine)
  - Calculate respective metrics ("generalization")
  - Repeat

Determine Model Quality – Data-Dependent

- Forms:
  - K-fold cross-validation
    - Decompose the learning data into k folds
    - Train on $k-1$ folds, and validate each on remaining fold
      - Measure variance between validation folds?
  - Leave-one-out cross-validation (special case)
    - k-fold whereas k = size of dataset $|D|$
      - Validation fold only contains single instance of dataset
      - Train model $|D|$-times and measure average evaluation error

This increases computational requirements by k and is mostly done for small learning systems



Bootstrap

- **Fundamental idea**: "How can you achieve more with simple procedures?"

- Learning
  - Draw randomly **(with layback)** $|D|$ instances k-times from dataset D
  - Learn model/parameters
  - Repeat
  - Determine the mean, variance,... of the model's metrics

➔Analysis of quality / stability

➔Use to find a higher quality model

Bagging: Bootstrap Aggregating

- Variant of bootstrap
- Use multiple different learning systems
  - Use bootstrap principle
  - draw $n = |D|$ instances (with layback)
  - Determine the respective models
- Combine the learning machines, e.g. weighted sum
  - Higher quality of the resulting model

Bagging: Bootstrap Aggregating

- Given : $\mathcal{D} = \{ (\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)\}$

- For $k = 1, \dots, k_{max}$
  - Draw new dataset $\mathcal{D}_k$ with layback from $\mathcal{D}$ with size n
  - Train model M_k with $\mathcal{D}_k$

- Combine models:
  - Regression: use average of trained models:

$$M(\vec{x}) = \frac{1}{k_{max}} \sum_{k=1}^{k_{max}} M_k(\vec{x})$$

  - Classification: Each model votes for the most likely class. Choose class that received the most votes.

Bootstrap

Aggregating

Boosting for classification – originally Schapire 1990

- Idea: Combine "weak" models to get a good model

- Basic Approach:
  - Get initial dataset $\mathcal{D}_1$ from $\mathcal{D}$
  - Use it to create model M_1
  - Choose $\mathcal{D}_2$ from $\mathcal{D}$ in such a way, so that 50% of its instances are classified correctly by M_1
  - Use D_2 to create model M_2
  - For $\mathcal{D}_3$, choose instances for which M_1 and M_2 are contradictory and create M_3
  - Combine models

$$M = \begin{cases} M_1, \ if \ M_1 = M_2 \\ M_3, \ else \end{cases}$$

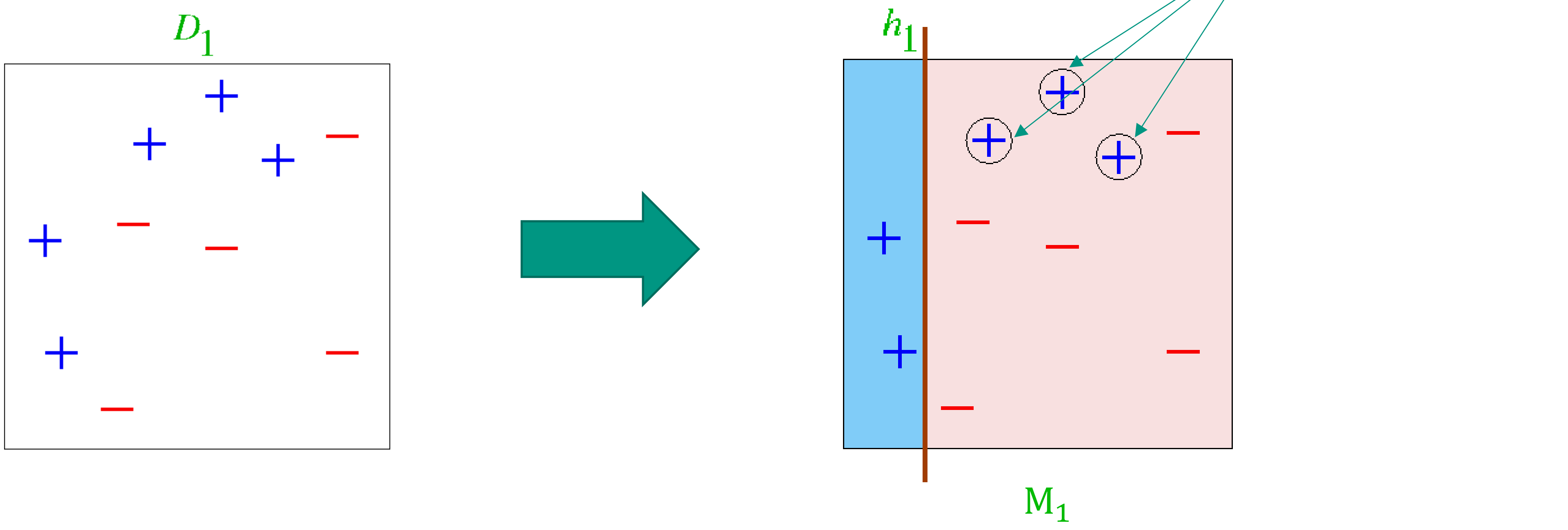
AdaBoost – adaptive Boosting

- Given: $\mathcal{D} = \{ (\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)\}$
- Initialization: Fixed weight for all learning instances $W_i = \frac{1}{n}$
- For $k = 1, \dots, k_{max}$
  - Train M_k with $\mathcal{D}_k$ ($|\mathcal{D}_k| = n$) with weighted learning instances W_i
  - How much to trust M_k?
    - Assign weights α_k to M_k
  - Increase weight of false classified learning instances
    - Calculate weights for all learning instances W_i
  - Normalize weights W_i
- Final Model Prediction:

$$\hat{y} = sign\left(\sum_{k=1}^{k_{max}} \alpha_k \, \mathrm{M}_k(x)\right)$$

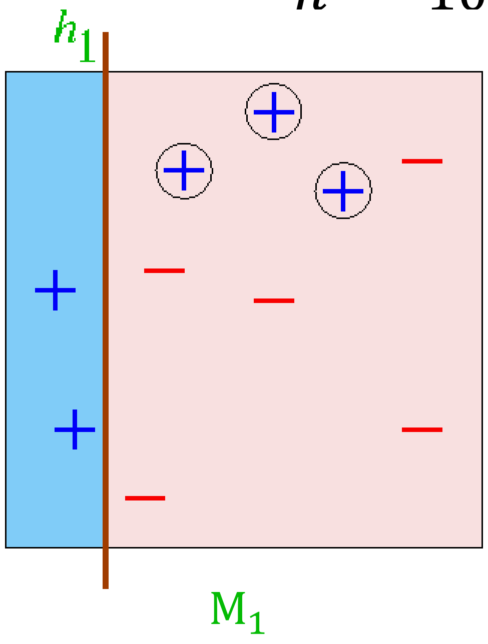
Adaptive Boosting: Example

- $k = 1$: Learning a classifier on original data



Adaptive Boosting: Example

- Calculate α_1 with weights of M_1 using weighted error E_k
- 10 samples: $W_i = \frac{1}{n} = \frac{1}{10}$



If E_k large → don't trust M_k
→ α_k should be small

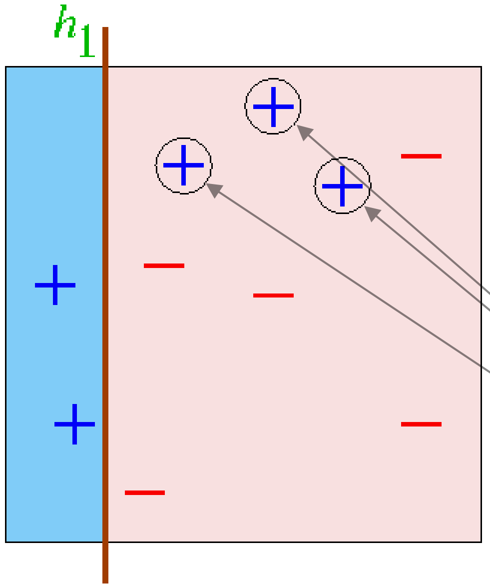
$$\boxed{\alpha_k = \frac{1}{2}\ln\left(\frac{1-E_k}{E_k}\right)}$$

$$E_1 = \frac{3}{10} = 0.3$$

$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-0.3}{0.3}\right) = 0.42$$

Adaptive Boosting: Example

- Updating W_i:
  - If instance $\vec{x}_i$ is correctly classified → decrease its W_i
  - If instance $\vec{x}_i$ is falsely classified → increase W_i

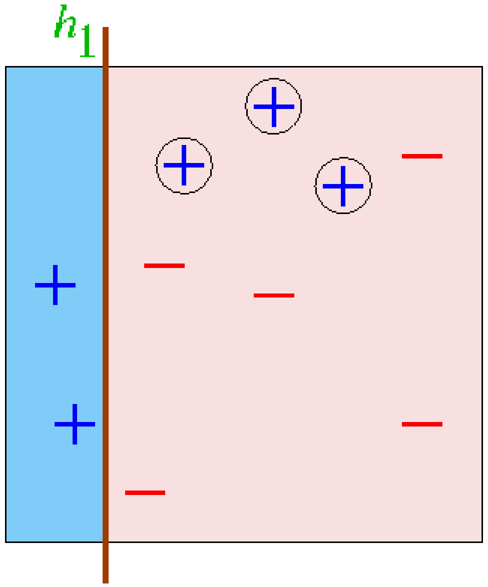


$$W_i \leftarrow \begin{cases} W_i\, e^{-\alpha_1}, & if\ M_1(\vec{x}_i) = y_i \\ W_i\, e^{+\alpha_1}, & if\ M_1(\vec{x}_i) \neq y_i \end{cases}$$

$$W_i \leftarrow \begin{cases} \frac{1}{10}\, e^{-0,42} = 0.065, & if\ M_1(\vec{x}_i) = y_i \\ \frac{1}{10}\, e^{+0,42} = 0.152, & if\ M_1(\vec{x}_i) \neq y_i \end{cases}$$

Adaptive Boosting: Example

- Normalize W_i,
  - Important for numerical stability.

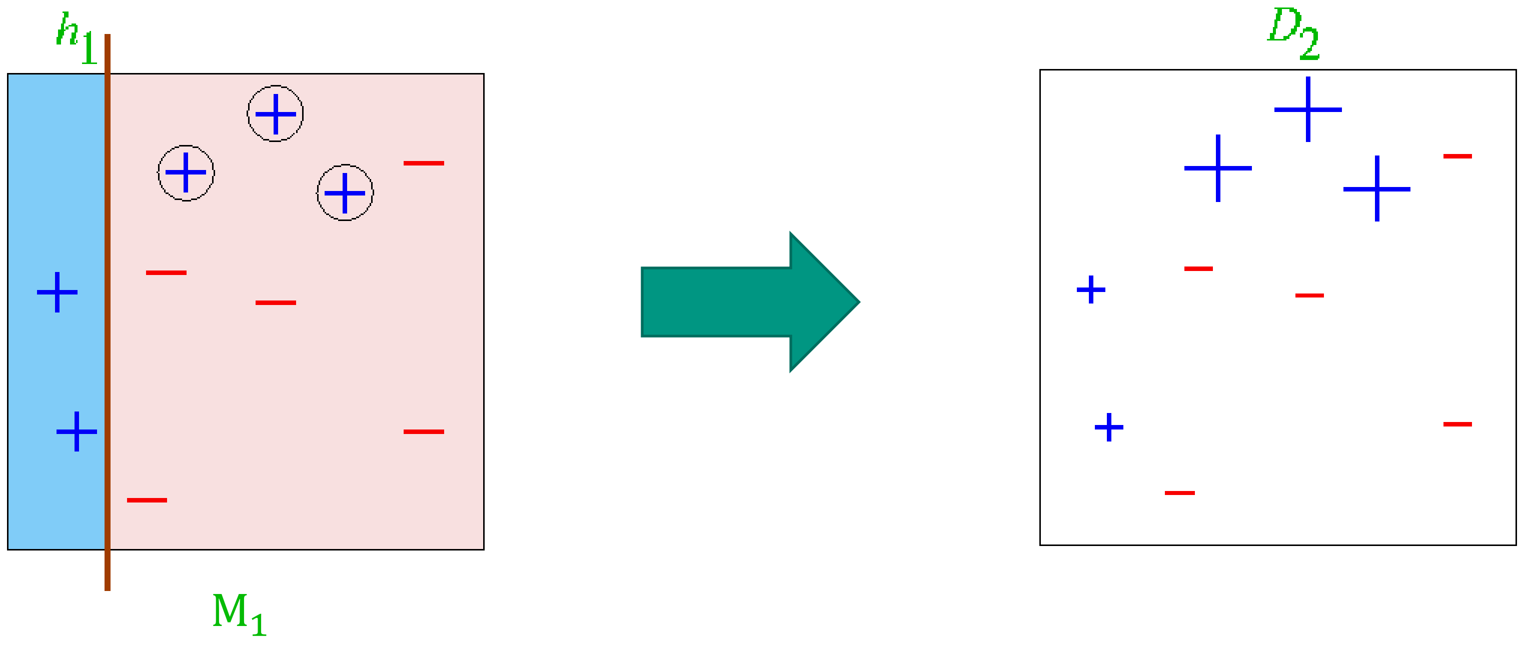


$$W_i \leftarrow \begin{cases} \dfrac{0.065}{3 \cdot 0.152 + 7 \cdot 0.065} = 0.071 & \text{if } M_1(\vec{x}_i) = y_i \\ \dfrac{0.0152}{3 \cdot 0.152 + 7 \cdot 0.065} = 0.166 & \text{if } M_1(\vec{x}_i) \neq y_i \end{cases}$$

$$7 \cdot 0.071 + 3 \cdot 0.166 \approx 1$$

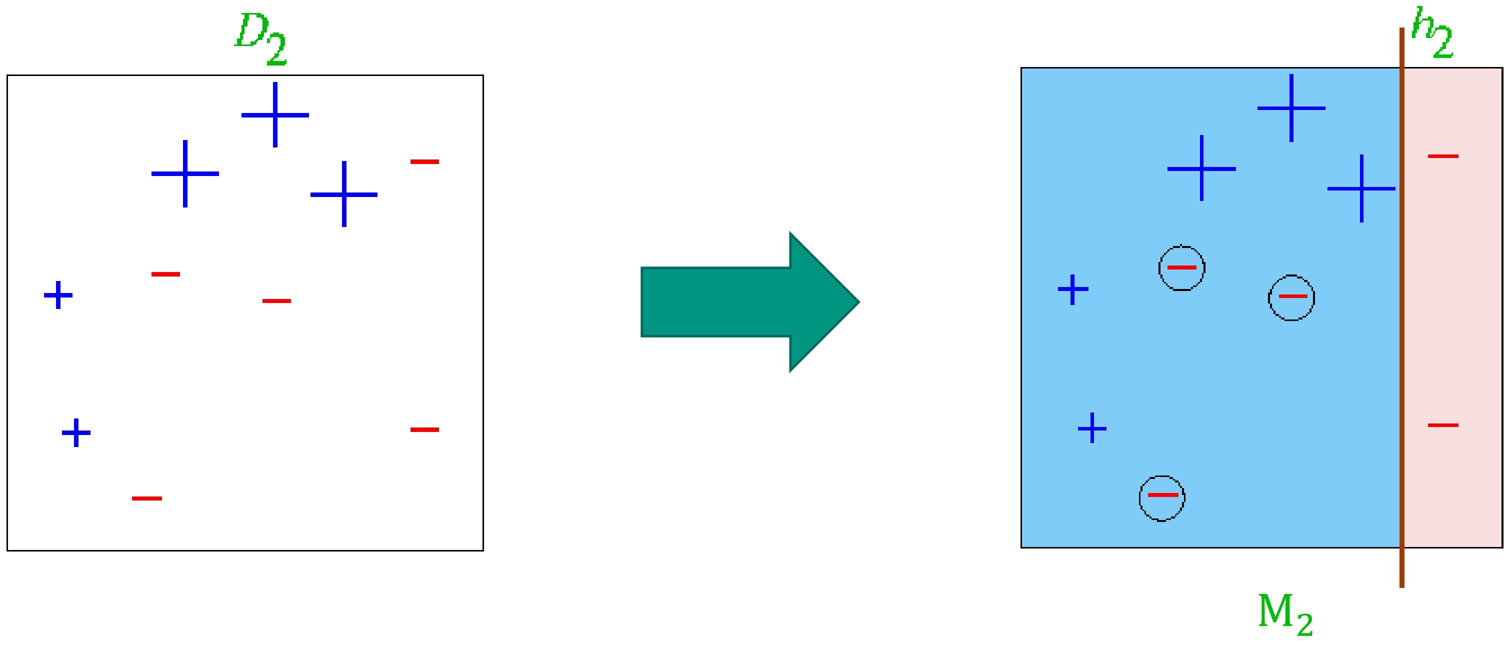
Adaptive Boosting: Example

- Update of W_i



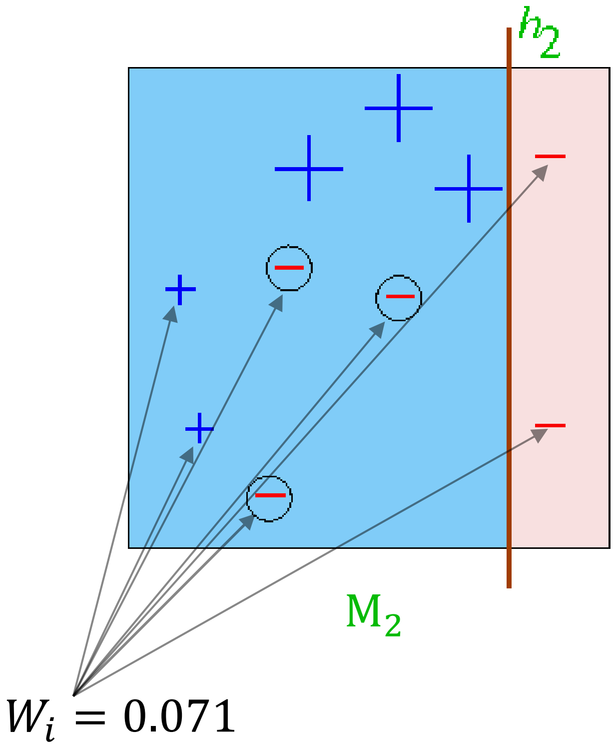
Adaptive Boosting: Example

- $k = 2$: Learn new classifier on weighted dataset



Adaptive Boosting: Example

- Calculate α_2 with weighted error of M_2



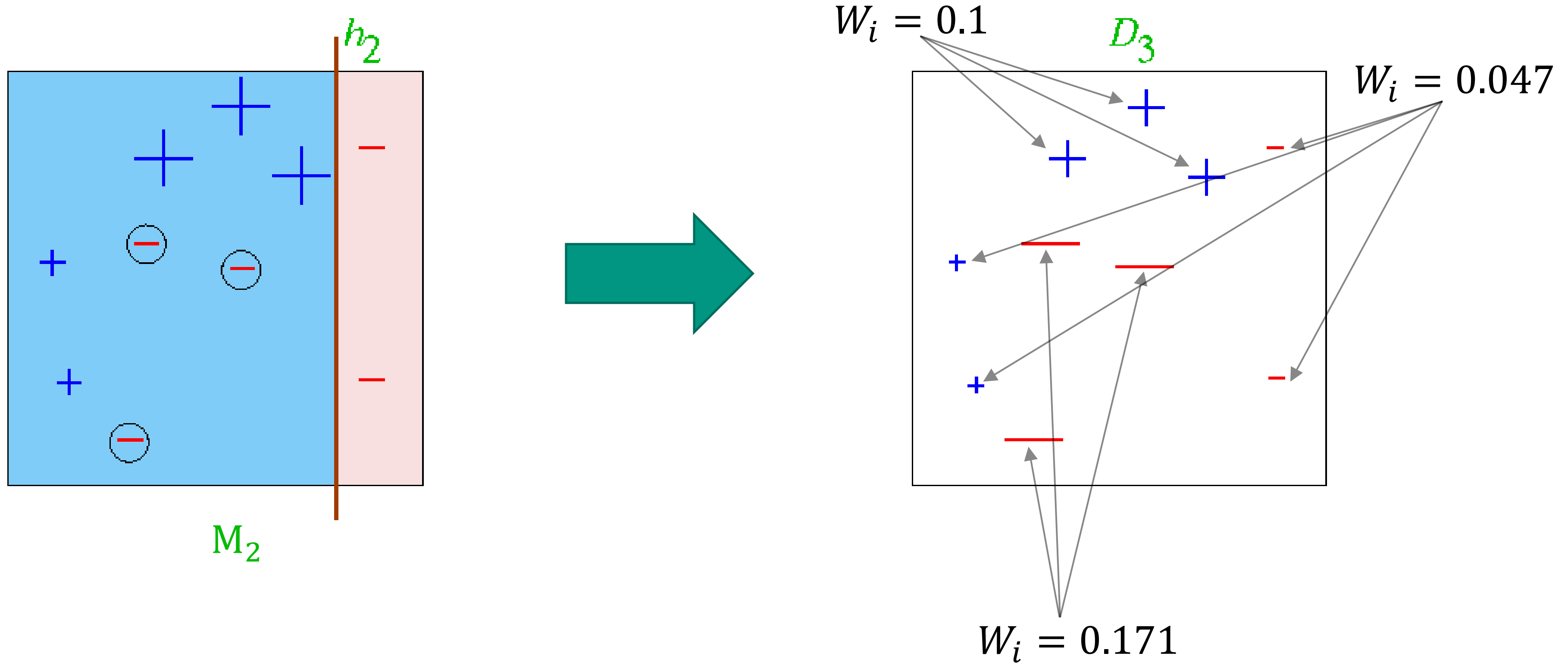
$$\alpha_2 = \frac{1}{2} \ln \left(\frac{1 - E_2}{E_2} \right)$$

$$E_2 = 3 \cdot 0{,}071 = 0.21$$

$$\alpha_2 = 0.65$$

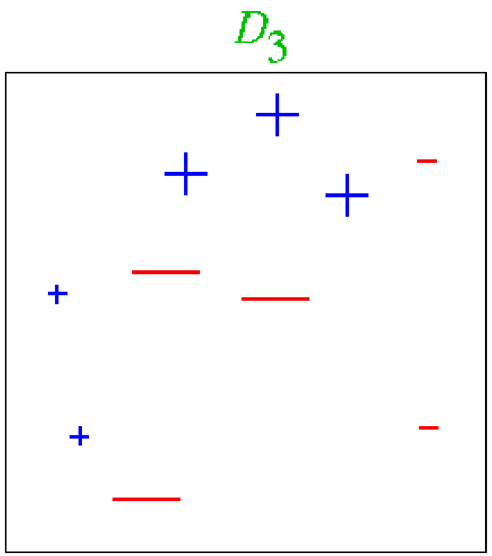
Adaptive Boosting: Example

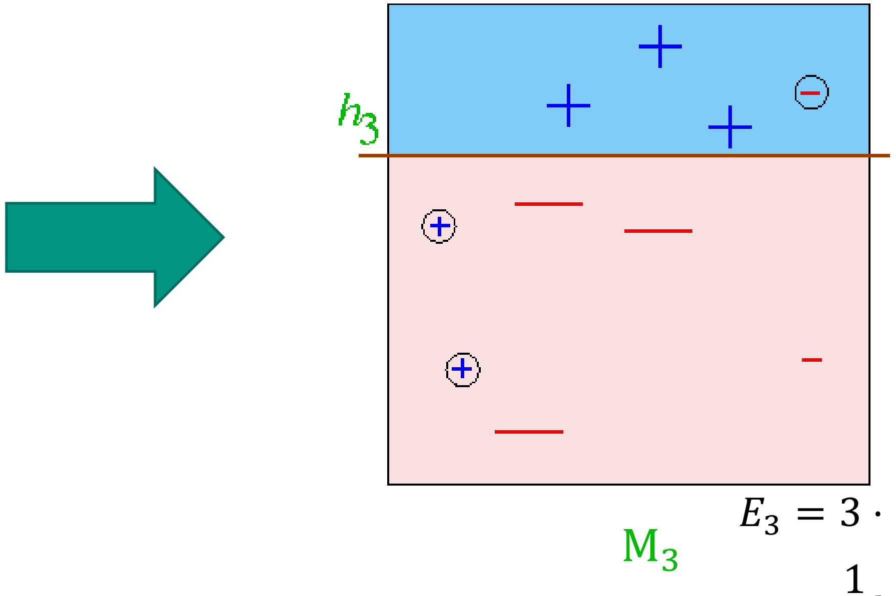
- Update α_2



Adaptive Boosting: Example

- $k = 3$ (k_{max}): learn last classifier

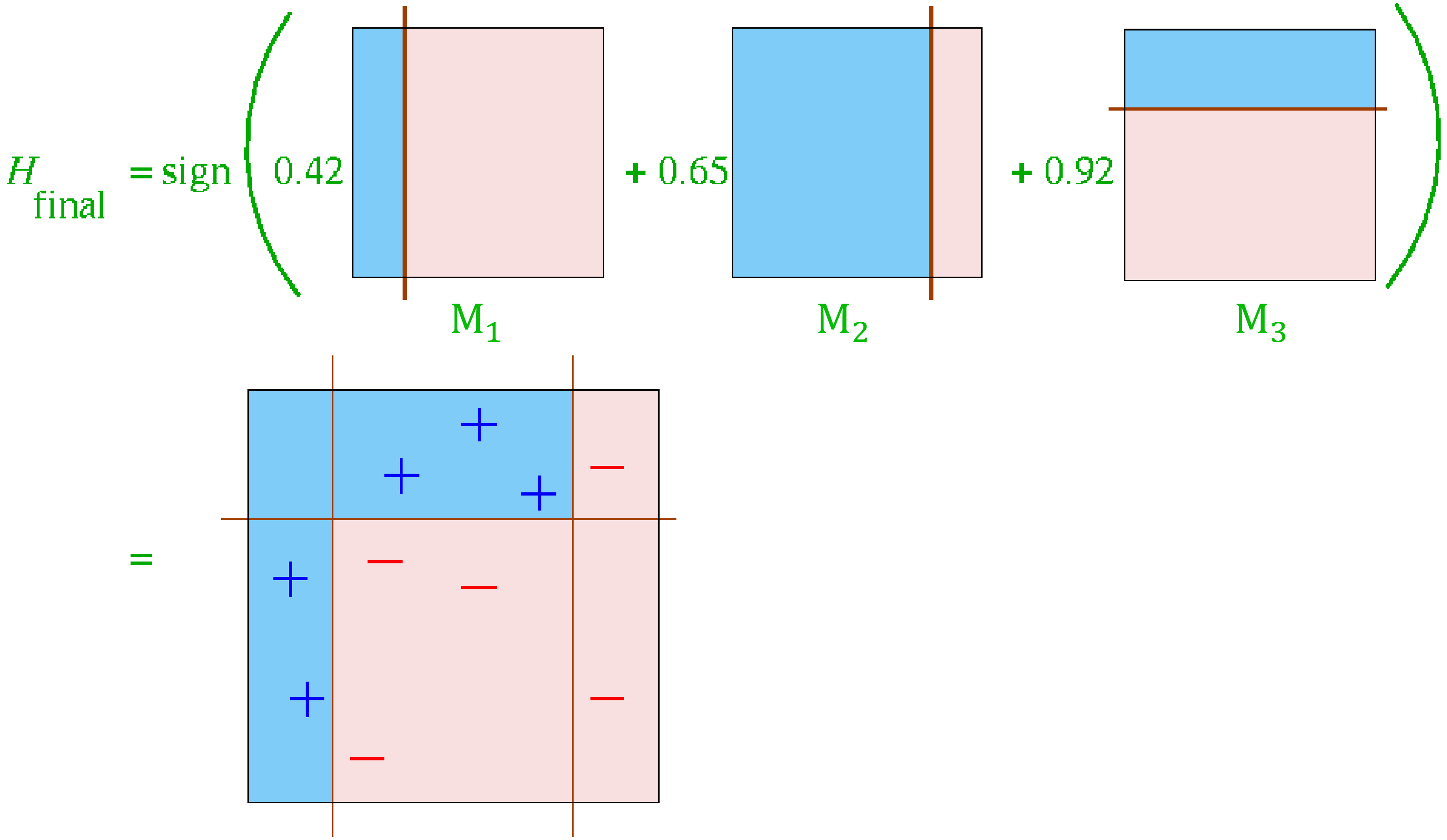




$$E_3 = 3 \cdot 0.047 = 0.141$$

$$\alpha_3 = \frac{1}{2}\ln\left(\frac{1-0.141}{0.141}\right) \approx 0.92$$

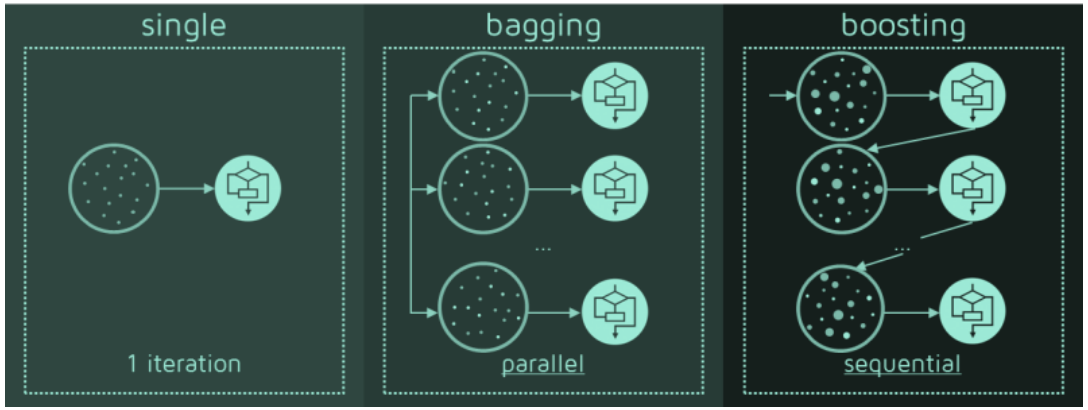
Adaptive Boosting: Example– final classification



Bagging vs. Boosting

- k_{max} models are used
- Bagging trains in parallel (models are independent of each other)
- Boosting creates new learning systems sequentially.

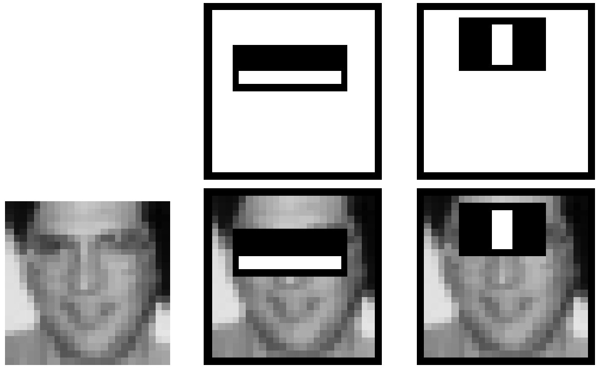
依次



Excursion: Viola & Jones Object Detection [~ 2001 – 2003]

Excursions are not relevant for exam

- Detection of faces in images
- Sliding Window: Create cutouts (e.g. 24 x 24 pixel) and decide if the contain face/ no face
- Classification: Characteristic-based (Haar-like Features)

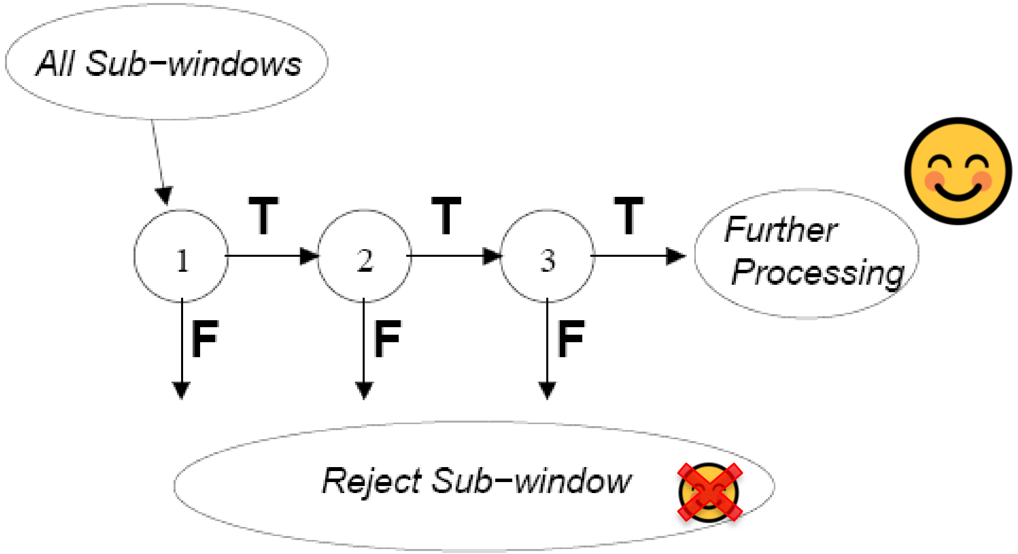


- Naïve → e.g. 180000 features per section then linear separation

- Much more efficient
  - Trick 1: Cascades of classificators with adjustable quality
  - Trick 2: Adjust quality of selected features with Adaboost
  - ......

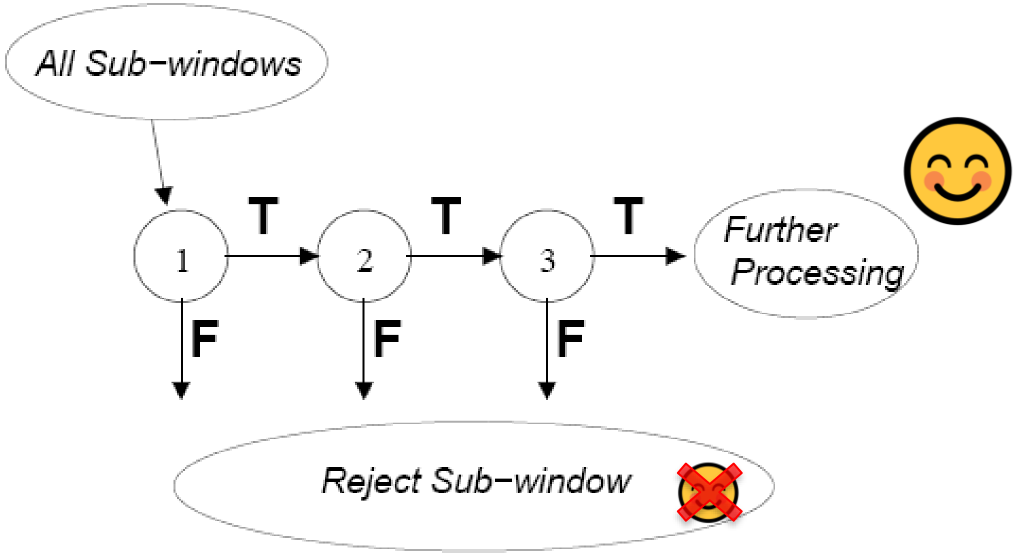
Excursion: Cascade Viola & Jones 2001

- 2-class problem
- Trick 1: Cascade
  - We are looking for a sequence of classifiers with increasing complexity
  - Procedure: A subset (**F** - no face) is gradually discarded from the dataset, which was classified by the previous cascade with **T** (potentially face).



Excursion: Cascade Viola & Jones 2001

- Define minimum conditions for each classifier
  - Recognition rate (here: real face in T)
  - False positive rate (here: no face in T)
- Cascade: Gradual reduction of the false positive rate
(but increasing the false negative rate and reducing the detection rate)
  - For 10 steps in the cascade, a detection rate of at least 0.99 and a false positive rate of 0.3 at the cascade will have at most:
  - a detection rate of
$0.99^{10} \approx 0.9$ and
  - a false positive rate of no more than
$0.3^{10} \approx 0.000006$

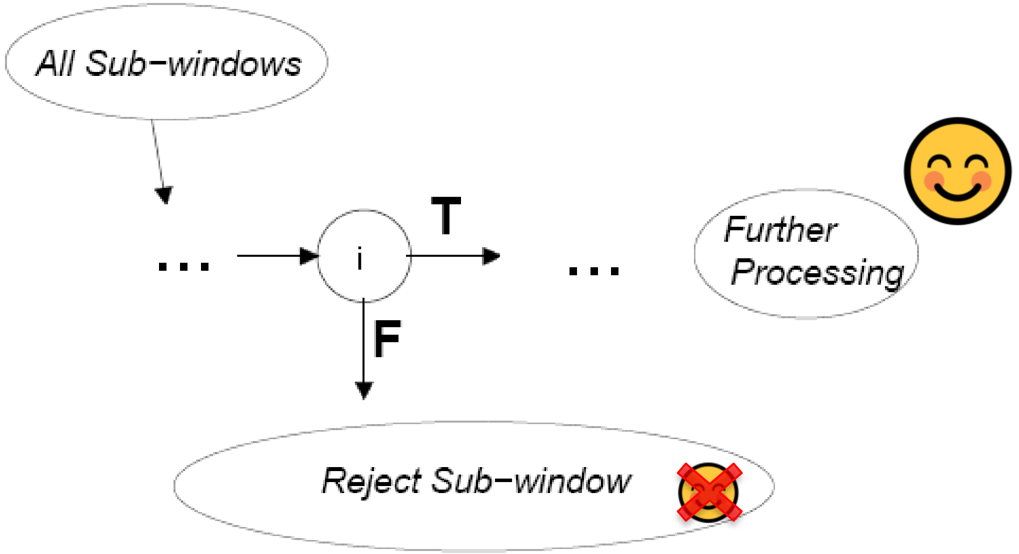


Excursion: Cascade Viola & Jones 2001

- Define minimum conditions for each classifier
  - Recognition rate (here: real face in T)
  - False positive rate (here: no face in T)

- Trick 2: Build individual classifiers such that the conditions are met

- Method: AdaBoost (iterative)
  - Combine simple threshold classifiers based on individual Haar-features
  - Which n?
    (→ Iteration)
  - What are the best features?
    (→ Adaboost)



Overview

- Motivation
- Is learning equivalent to optimization?
  - Can learning be described formally?
  - Error minimization for empirical and real error
  - Hypothesis quality, model selection
  - Boosting, Ensembles
- Learnability and Capacity of Learning Machines
  - VC – Dimension

Excursion: PAC – Learnability

- PAC = Probably Approximate Correct
- Given:
  - A set X of instances, each with length n
  - Concept C: $X \rightarrow \{true, false\}$
  - Hypotheses space H
  - A set of learning data D_L
- Can a correct hypothesis from H, $h(\vec{x}) = c(\vec{x}), \forall\vec{x} \in D_L$ be found?
  - No
  - But an ε - accurate: $\hat{\mathcal{L}}_{D_L}(h) \leq \varepsilon, 0 < \varepsilon < \frac{1}{2}$ Approximate Correct

Excursion: PAC – Learnability

- Can it be found always?
  - No
  - But with chosen probability

$$1-\delta, 0<\delta<\frac{1}{2}$$ Probably

- How is the problem (to find hypothesis) solvable?
  - in polynomial time depending on: $\frac{1}{\delta}, \frac{1}{\varepsilon}, n$
  - with memory complexity depending on: C

PAC – Sample Complexity (Exkurs)

- And the number of learning data needed (sample complexity) is:

$$m \geq \frac{1}{\varepsilon}(\ln(\frac{1}{\delta}) + \ln|H|)$$

- And what does that mean?
  - the higher the desired quality
  - the smaller the permissible error
  - the larger the hypothesis space

  → the greater the number of data required

Excursion: PAC – Example (Excursion)

- For hypotheses
  - that consist of conjunctions of
  - up to ten literals
  - can with 95% certainty
  - a hypothesis with error < 0.1 be found
- Given these prerequisites, a learning system requires at least:

$$m \geq \frac{1}{0{,}1}\left(ln\left(\frac{1}{0.05}\right) + 10ln|3|\right) = 140|H| = 3^{10} \text{ learning instances}$$

Unfortunately, not so easy for complex functions and machines

Vapnik-Chervonenkis (VC) Dimension

- How can we formally define what a learning system can achieve?

- A set of mappings (hypotheses)

$$\{h_\theta : \theta \in Parameter\ Space\}$$

  define the hypotheses space H

- **Definition** (for linear classification): The VC-Dimension $VC(h_\theta)$ of H is equal to the maximum number of data points (from set S) which can be arbitrarily separated from H

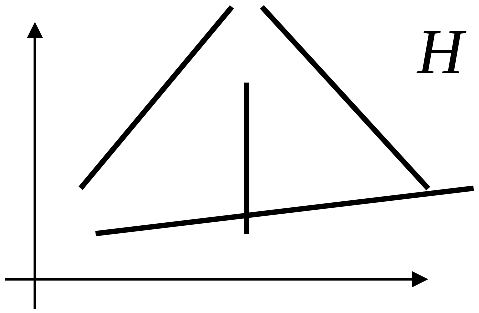
Vapnik-Chervonenkis (VC) Dimension

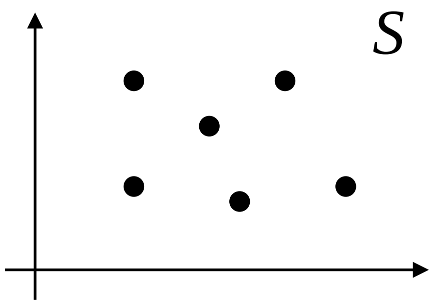
- **Definition** (for linear classification): A mapping (hypothesis) $h(x)$ *shatters* the data S if $h(x)$ creates subsets which can be defined :

$$\{x|h(x) = 0\} \quad \{x|h(x) = 1\}$$

- **Example**:

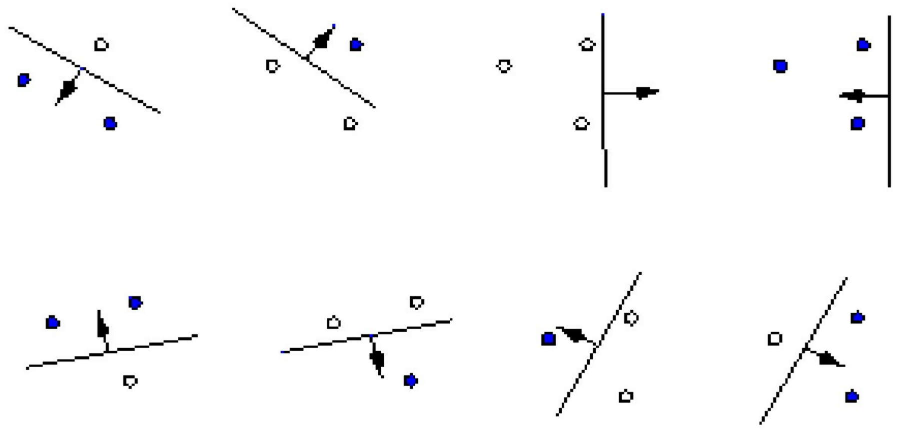
  Hypotheses space are hyperplanes in $\mathbb{R}^2$ and $S \subset \mathbb{R}^2$





VC Dimension Example

- **Assertion**:
  - A maximum of 3 data points can be separated from straight lines if all arbitrary splits are allowed



- General: Hyperplane in $R^n \Rightarrow n + 1$ separable values

VC Dimension – Use Case I

- Measure of data complexity of learning [Blumer et al 1988]
- Statements about PAC - sample complexity, number of learning examples m

$$m \geq \frac{1}{\varepsilon}\left(4\log_2(\frac{2}{\delta}) + 8VC(h)\log_2(\frac{13}{\varepsilon})\right)$$

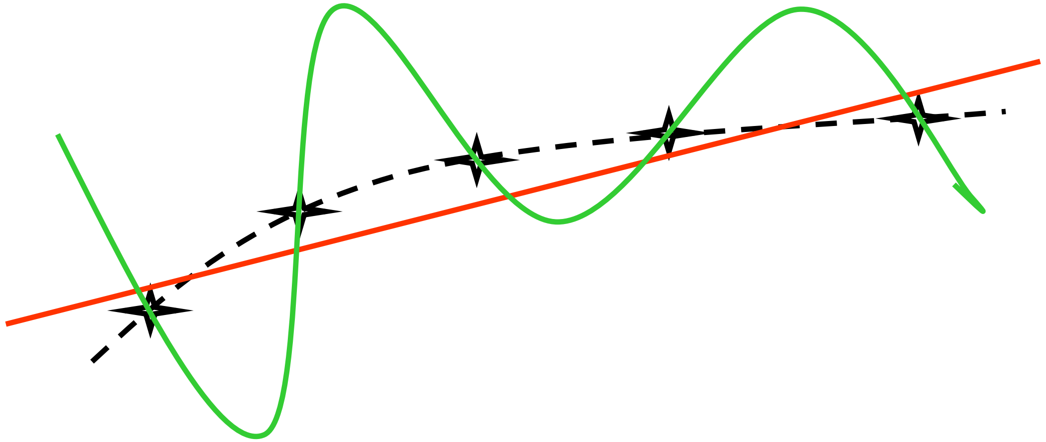
  - Significantly better estimation, which also includes the learning machine

- There are further restrictions for special machines……

VC Dimension – Use Case II

- $VC(h)$ is a measure of the capacity of learning systems
- Guess:
  The larger $VC(h)$, the better a system can learn to solve a problem

~~True~~ or False?



Schematics:

- $VC(h)$ ↘
- $VC(h)$ ↗
- Error, in this example, is greater for model with large VC-Dimension

Estimation of Test-Error

- According to Vapnik with probability η:

$$\mathcal{L}(h_\theta) \leq\approx \hat{\mathcal{L}}(h_\theta) + \sqrt{\ldots \frac{VC(h_\theta)}{N} \ldots}$$

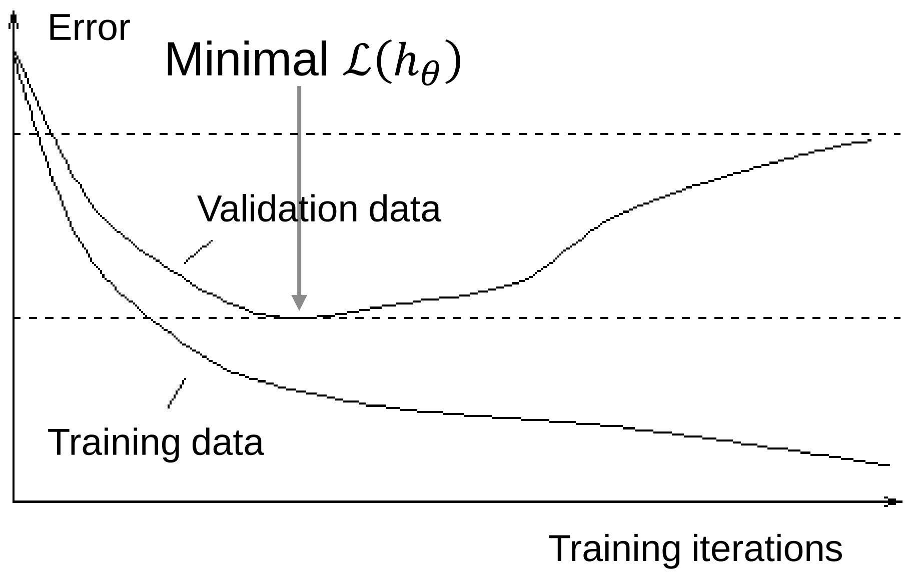
As small as possible

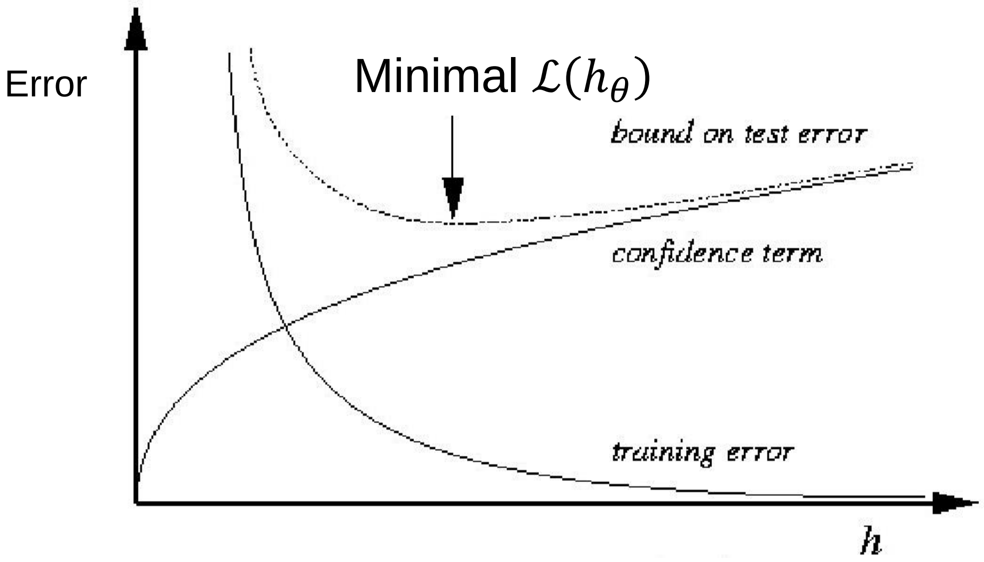
As large as possible

  - whereas:
    - $VC(h_\theta)$ – VC-dimension of learning system
    - N – number of learning instances
    - $\hat{\mathcal{L}}(h_\theta)$ – empirical error, dependent from $VC(h_\theta)$ und N
    - $\mathcal{L}(h_\theta)$ – real error

- Success of learning is dependent on:
  - Capacity of learning system (as small as possible)
  - Optimization method (as good as possible)
  - Learning instances in dataset (representative of real world and as many as possible)

Estimation of Test-Error





- Minimization of emp. error with:
  - Constant VC-dimension
    (e.g. keeping topology of model constant)
  - Constant number of training instances
  - Iterative optimization
- Relationship between empirical and real error
  - VC-dimension = changeable
  - Constant number of training instances.

Structural Risk Minimization

- **Goal**: find a solution for

$$\min_{H^n}\left(\hat{\mathcal{L}}(h_\theta) + \sqrt{\dots\frac{VC(h_\theta)}{N}\dots}\right)$$

→ find $VC(h_\theta)$ („learning system"), N („number instances"), and θ („minimum of empirical error")

- **Ideal solution (meta-algorithm) :** minimize whole sum, not individual summands
  - Use learning system that can create multiple hypotheses spaces with changing complexities
  - Structure hypothesis spaces with regards to their VC-dimension

$$H^1 \subset H^2 \subset \cdots \subset H^n, VC\left(h_\theta^i\right) \leq VC(h_\theta^{i+1})$$

  - Iterate over all hypothesis spaces H^i
  - Train learning system to receive optimal empirical error $\hat{\mathcal{L}}(h_\theta)$ for given hypothesis space
  - If sum is minimized → stop and use this hypothesis.

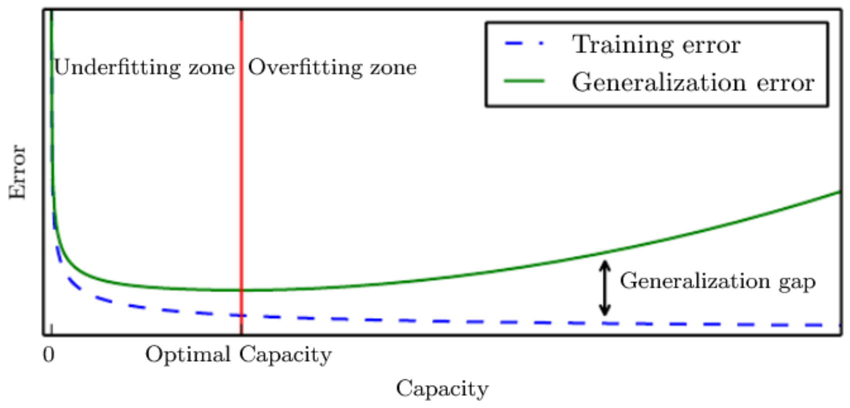
Structural Risk Minimization

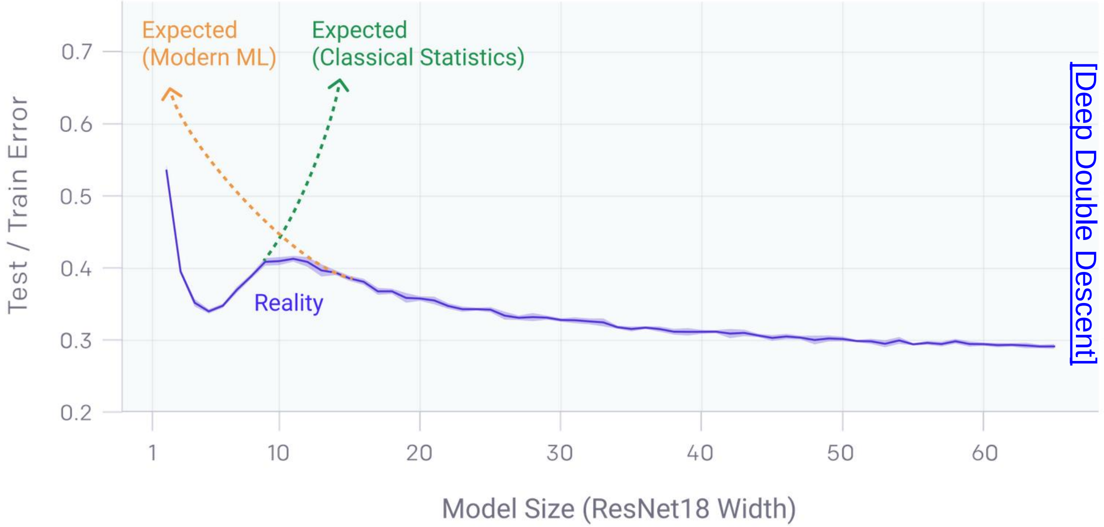
- Problems:
  - Calculation of VC-dimension is hard and resource intensive
    - Mostly unfeasible for currently used large neural network architectures
  - Not all models support the structure and for many models impossible

- **Correct Learning**: Find fitting realizations to get the optimum of the sum

Outlook: Is this always true?

- Are these learning theories from this lecture always applicable?
  - Well, in theory but reality shows exceptions
- Double Descent (short version)
  - With increasing complexity of hypothesis space we initially expect a decrease in test-error and then an increase in test-error (overfitting, classical statistics)
  - But recent observations have shown that increasing complexity even further will reduce test-error again
  - Until now: not fully understood why it is happening, one explanation from 2023: [here]





[Deep Double Descent]

Literature

- Tom M. Mitchell: Machine Learning, 1997
  - Online
- Ian Goodfellow and Yoshua Bengio and Aaron Courville: Deep Learning, MIT Press, 2016, https://www.deeplearningbook.org/
- V.N. Vapnik, 1997: Statistical Learning Theory
- Leo Breiman, 1996: Bagging Predictors
- Duda & Hart: Pattern Classification
- P. Viola, M. Jones, 2001: Rapid object detection using a boosted cascade of simple features