Deep Learning

LM Computer Science, Data Science, Cybersecurity

2nd semester - 6 CFU

Luca Pasa, Nicolò Navarin & Alessandro Sperduti

Introduction to Machine Learning Basics

What Is Machine Learning?

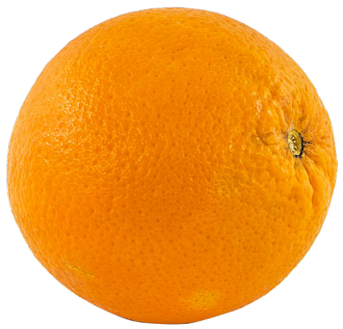
"A computer program is said to learn from **experience E** with respect to some class of **tasks T** and **performance measure P** if its performance at tasks in T, as measured by P, improves with experience E"

When to use Machine Learning

- Difficult to formalize the problem, easy to provide examples
- Presence of noise

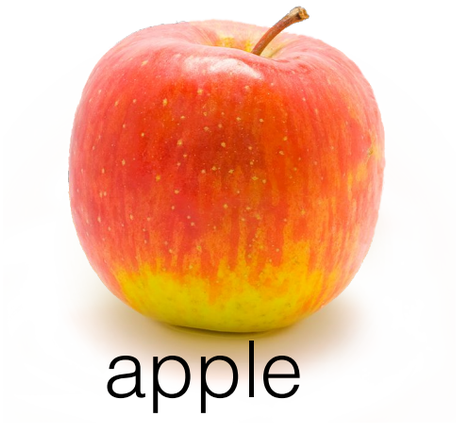
Learning Algorithm:

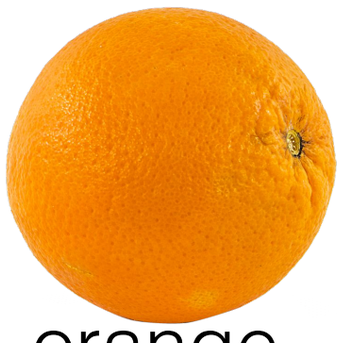
- an algorithm that is able to *learn* from *data*



The Task

- A task is usually described in terms of how the machine learning algorithm should process an *example (i.e. what the output should be)*


apple


orange


mango

- How is an example represented ?
  as a collection of features
  (that can be measured)

The Task

- Many different tasks, e.g.
  - Classification (with missing data),
  - Regression,
  - Machine translation,
  - Structured output,
  - Anomaly detection,
  - Synthesis and sampling,
  - Imputation of missing values,
  - Denoising,
  - Density estimation or probability mass function estimation

The Performance Measure

- How good is the learning algorithm ?
- We need to measure its performance, i.e. how accurate is the function/model returned by it!
- The performance measure depends on the task, e.g.:
  - **Classification** -> ***accuracy***, proportion of examples for which the model produces the correct output

accuracy: 0.7

classification

mango	apple	orange	mango	orange	apple	orange	orange	apple	orange
correct!	*correct!*	*correct!*	*correct!*	*correct!*	*wrong!*	*wrong!*	*correct!*	*correct!*	*wrong!*

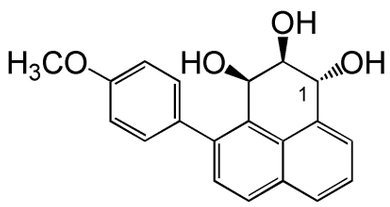
Alternative: 0-1 Loss (error rate -> 0, no error; 1 error)

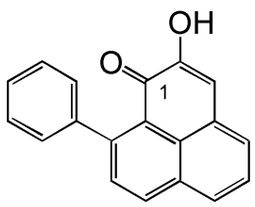
0	0	0	0	0	1	1	0	0	1

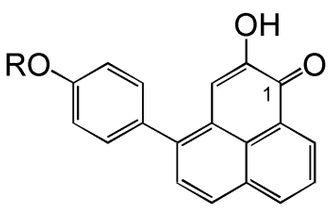
error rate: 0.3

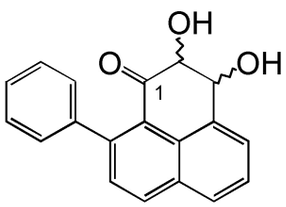
The Performance Measure

- How good is the learning algorithm ?
- We need to measure its performance, i.e. how accurate is the function/model returned by it!
- The performance measure depends on the task, e.g.:
  - **Regression** -> ***mean squared error (MSE)***, the *average* of the squares of the *errors*









predicted toxicity index →	1.2	0.9	0.75	1.1
real toxicity index →	1.0	1.1	0.8	1.2
squared error →	0.04	0.04	0.0025	0.01

MSE: 0.023125

The Experience

- The dataset
- Which kind of data ?
  - real-valued features
  - discrete features
  - mixed features
- How do we get data ?
  - obtained once for all (batch learning)
  - acquired incrementally (on-line learning)
    - by interacting with the environment
- How can data be used ?
  - Learning paradigms

Main Learning Paradigms

Different paradigms

- Supervised Learning
- Unsupervised learning
- Reinforcement learning
- .. and many others.

Main Learning Paradigms

Supervised learning:

The most common learning paradigm.

- Task **T**: predicting a dependent variable from one or more independent variables
- Performance Measure **P**: error on the predictions
- Training Experience **E**: a training dataset, with values for both independent and dependent variables

Independent variable

Dependent variable

Years of experience	Salary in 1000$
2	15
3	28
5	42
13	64
8	50
16	90
11	58

Main Learning Paradigms

Supervised learning (more formally):

Assumption: $t^i = f\left(x^{(i)}\right) + \epsilon^{(i)}$

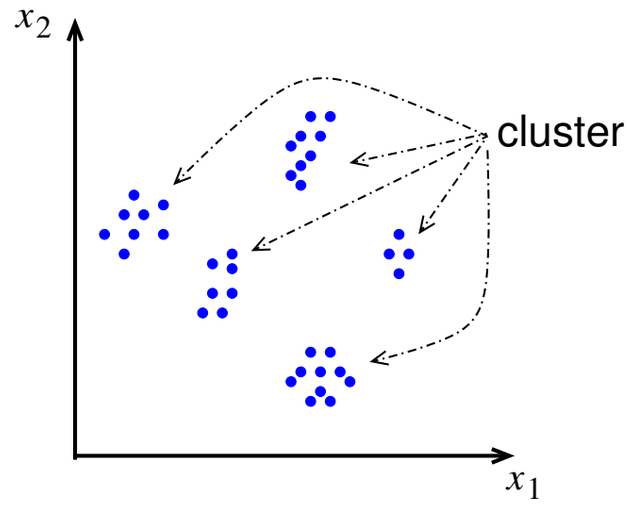
- Given pre-classified examples, $Tr = \left\{\left(x^{(i)}, t^{(i)}\right)\right\}$, learn a general description which captures the information content of the examples (rules working for the whole input domain)
- The model will learn a function h such that $h\left(x^{(i)}\right) \approx f(x^{(i)})$ for all the examples in Tr
- It should be possible to use this description in a predictive way (given a new input $\hat{x}$, predict the associated output $\mathbf{h}(\hat{x})$)

Note: It is assumed that an expert (or teacher) provides the supervision (i.e. the values of $t^{(i)}$corresponding to the training instances)

Main Learning Paradigms

Unsupervised Learning:

- given a set of examples $Tr = \{x^{(i)}\}$, discover regularities and/or patterns
  (true on the whole input domain)
- there is no expert (or teacher) to help us (i.e., no supervision!)



- Clustering
- Rule Discovery

Main Learning Paradigms

Reinforcement Learning:

- agent which may
  - be in state s, and
  - execute an action a (chosen among the ones admissible in the current state)
- and operates in an environment e, which in response to action a in the state s returns
  - the next state, and
  - a reward r, which can be positive (+), negative (-), or neutral (0).

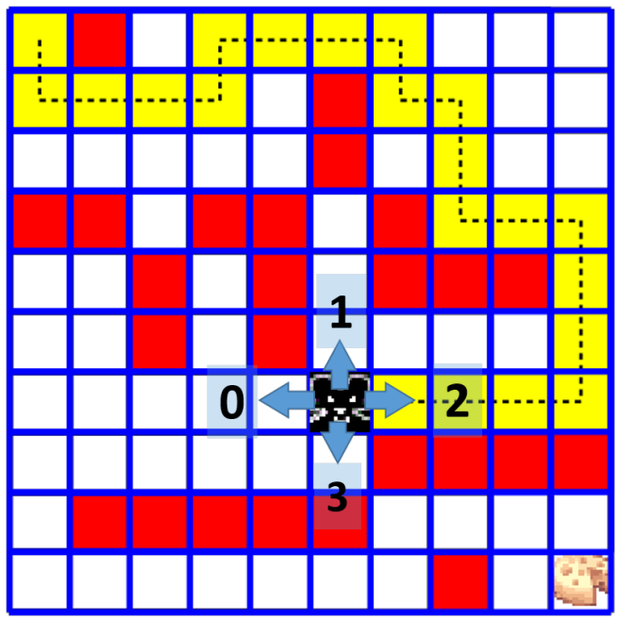
The goal of the agent is to maximize a function of the rewards (e.g. expected discounted sum of rewards: $\sum_{t=0}^{\infty} \gamma^t r_{t+1}$ where $0 \leq \gamma < 1$)

The reward is influenced by past actions (not only by the last one)

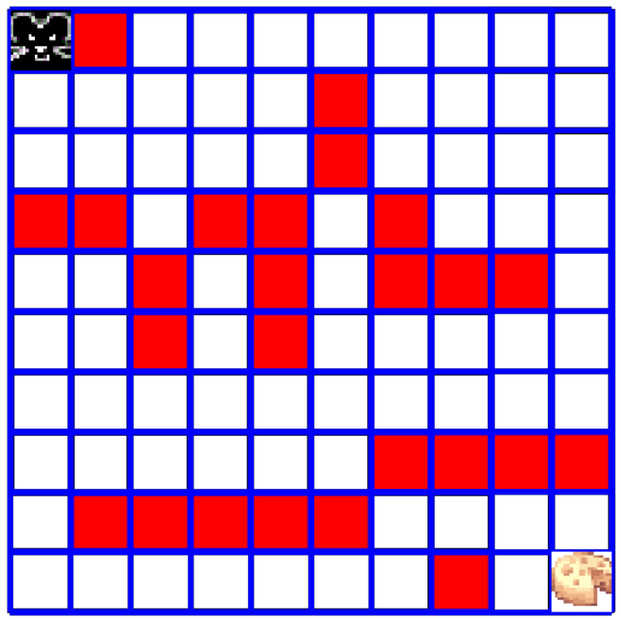
Example of Reinforcement Learning

- Example from HERE.

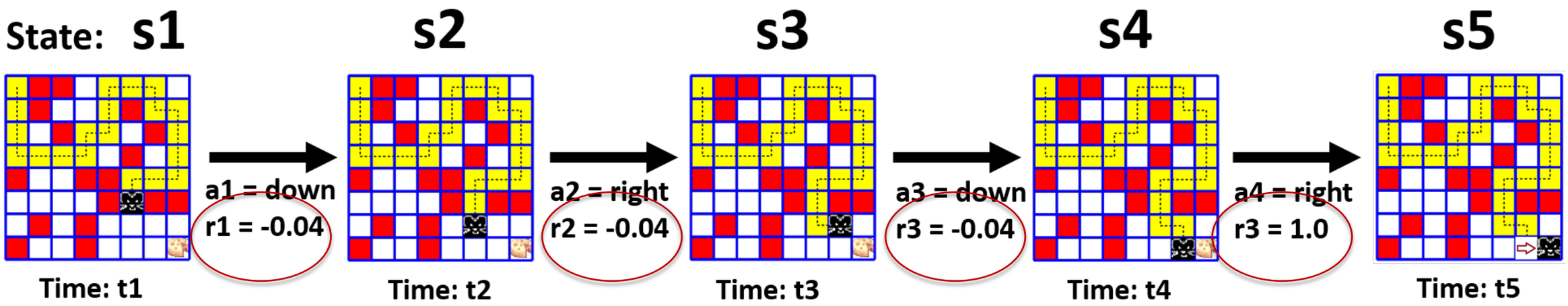
4 actions



An action takes the agent to a different state..

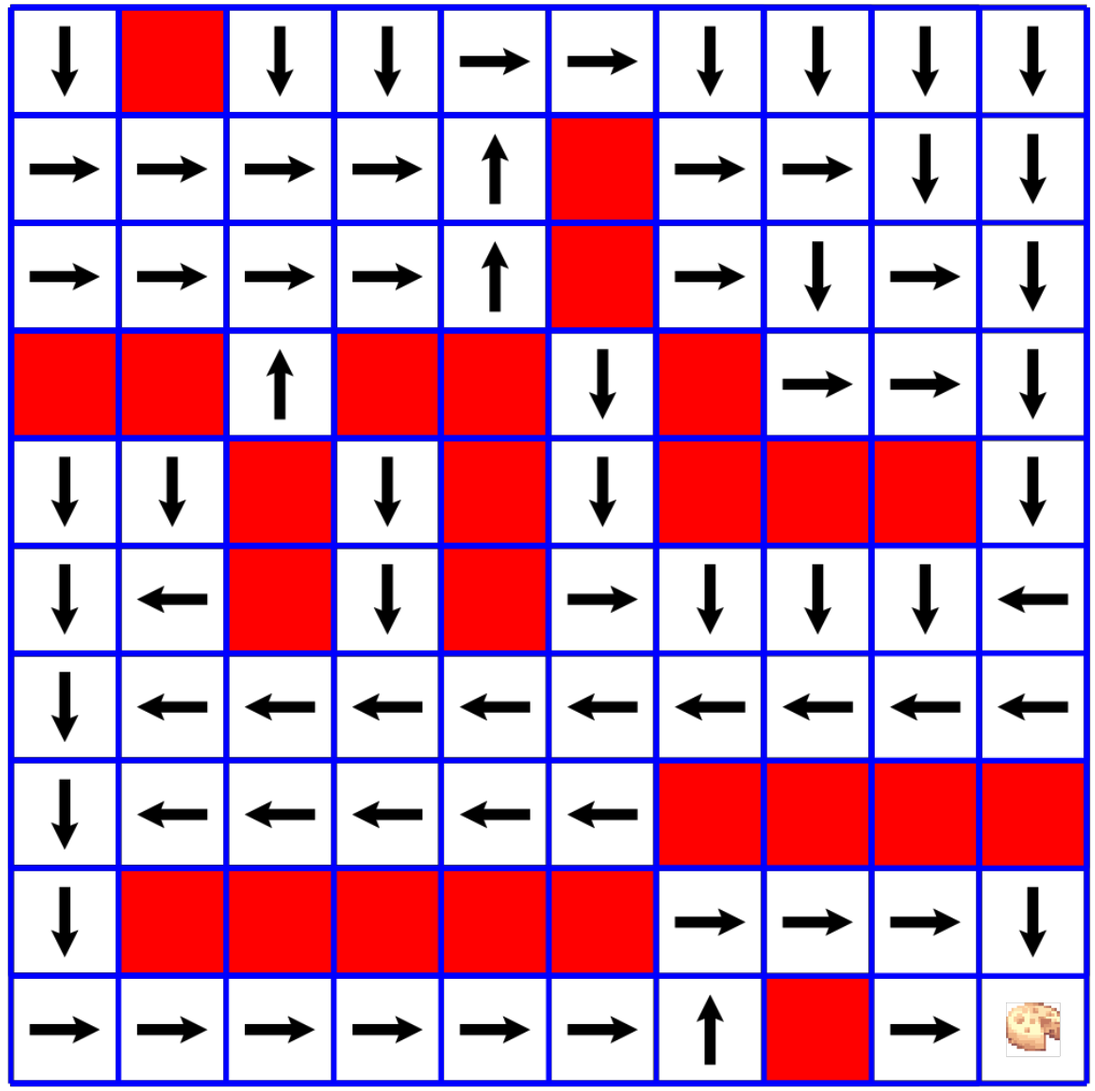


…and provides a reward



Example of RL 2

- The agent learns a policy: a mapping from states to actions, that maximizes the long-term reward



Hypothesis Space

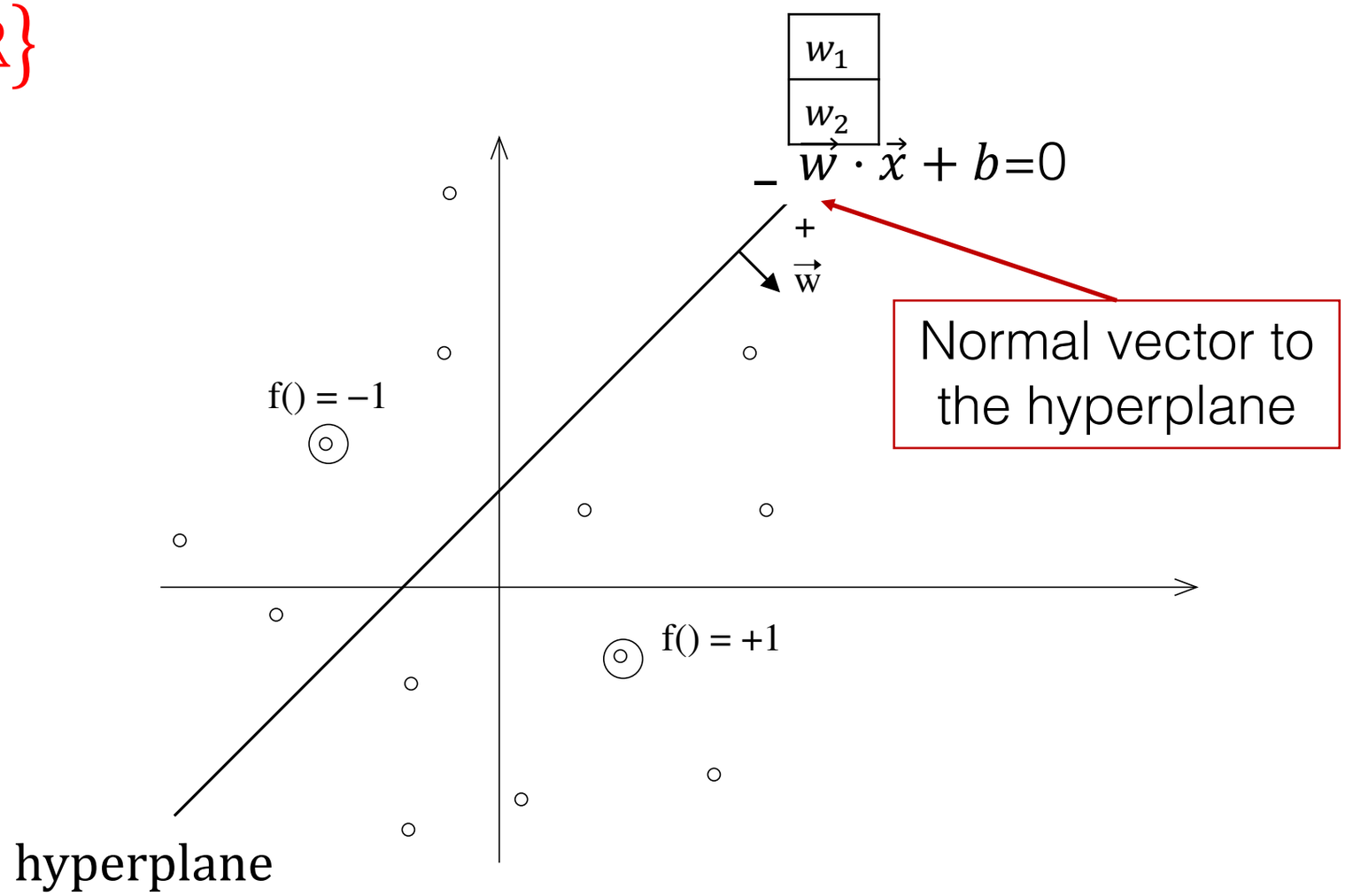
- Training Data (drawn from the Instance Space, X)
- Hypothesis Space, $\mathcal{H}$
  - it constitutes the set of functions which can be implemented by the machine learning system;
  - it is assumed that the function to be learned f may be represented by a hypothesis $h \in \mathcal{H}$... (the actual h is selected via the training data)
    - or that at least a hypothesis $h \in \mathcal{H}$ is "similar" to f (approximation);
- Search Algorithm into the Hypothesis Space, learning algorithm

WARNING: $\mathcal{H}$ cannot coincide with the set of all possible functions and the search (into $\mathcal{H}$) to be exhaustive $\rightarrow$ Learning is useless!!!
Inductive Bias: on the representation ($\mathcal{H}$) and/or on the search (learning algorithm)

Example of Hypothesis Space (for classification)

Lines in $\mathbb{R}^2$

- Instance space → points in the plane: $X = \{\vec{x} \in \mathbb{R}^2\}$
- Hypothesis Space → dichotomies induced by hyperplanes in $\mathbb{R}^2$ (lines): $\mathcal{H} = \{ f_{\vec{w},b}(\vec{x}) | f_{\vec{w},b}(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b), \vec{w} \in \mathbb{R}^2, b \in \mathbb{R}\}$



Example of Learning Algorithm: the Perceptron (Rosenblatt, 1958)

- Very simple (binary) classifier

$$out = sign(\vec{w} \cdot \vec{x} + b)$$

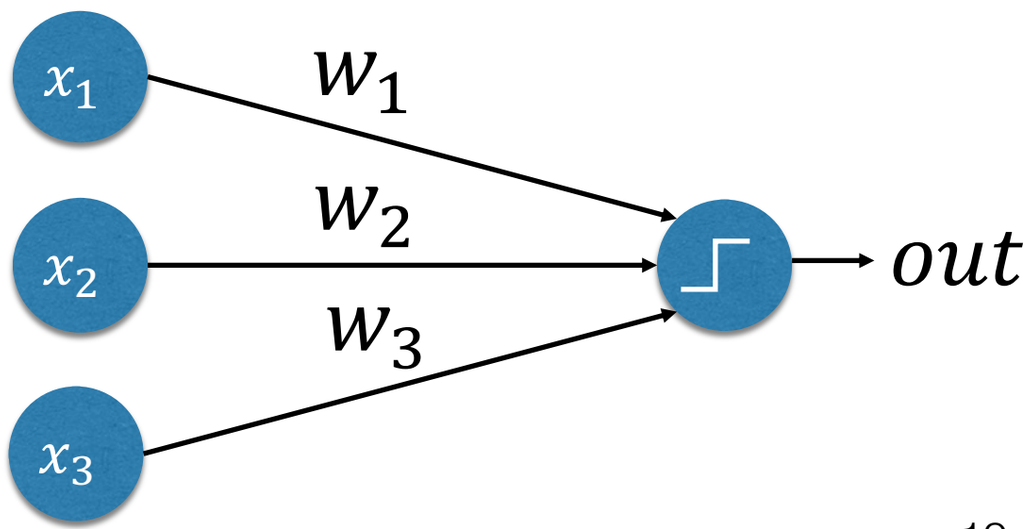
- Supervised Learning
- Consider the space of hyperplanes in $\mathbb{R}^n$

$$\mathcal{H} = \left\{ f_{\vec{w},b}(\vec{x}) | f_{\vec{w},b}(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b) \colon x, \vec{w} \in \mathbb{R}^n, b \in \mathbb{R} \right\}$$

- that we can rewrite as:

$$\mathcal{H} = \left\{ f_{\vec{w},b}(\vec{x}) | f_{\vec{w},b}(\vec{x}) = sign(\vec{w} \cdot \vec{x}) \colon x, \vec{w} \in \mathbb{R}^{n+1} \right\}$$

$$\vec{w}=[\vec{w}, b], \quad \vec{x}=[\vec{x}, 1]$$



Example of Learning Algorithm: the Perceptron (Rosenblatt, 1958)

Training algorithm for a Perceptron

input: training set $Tr = \{(\vec{x}, t)\}$, where $t \in \{-1, +1\}$ and $\vec{x} \in \mathbb{R}^n$

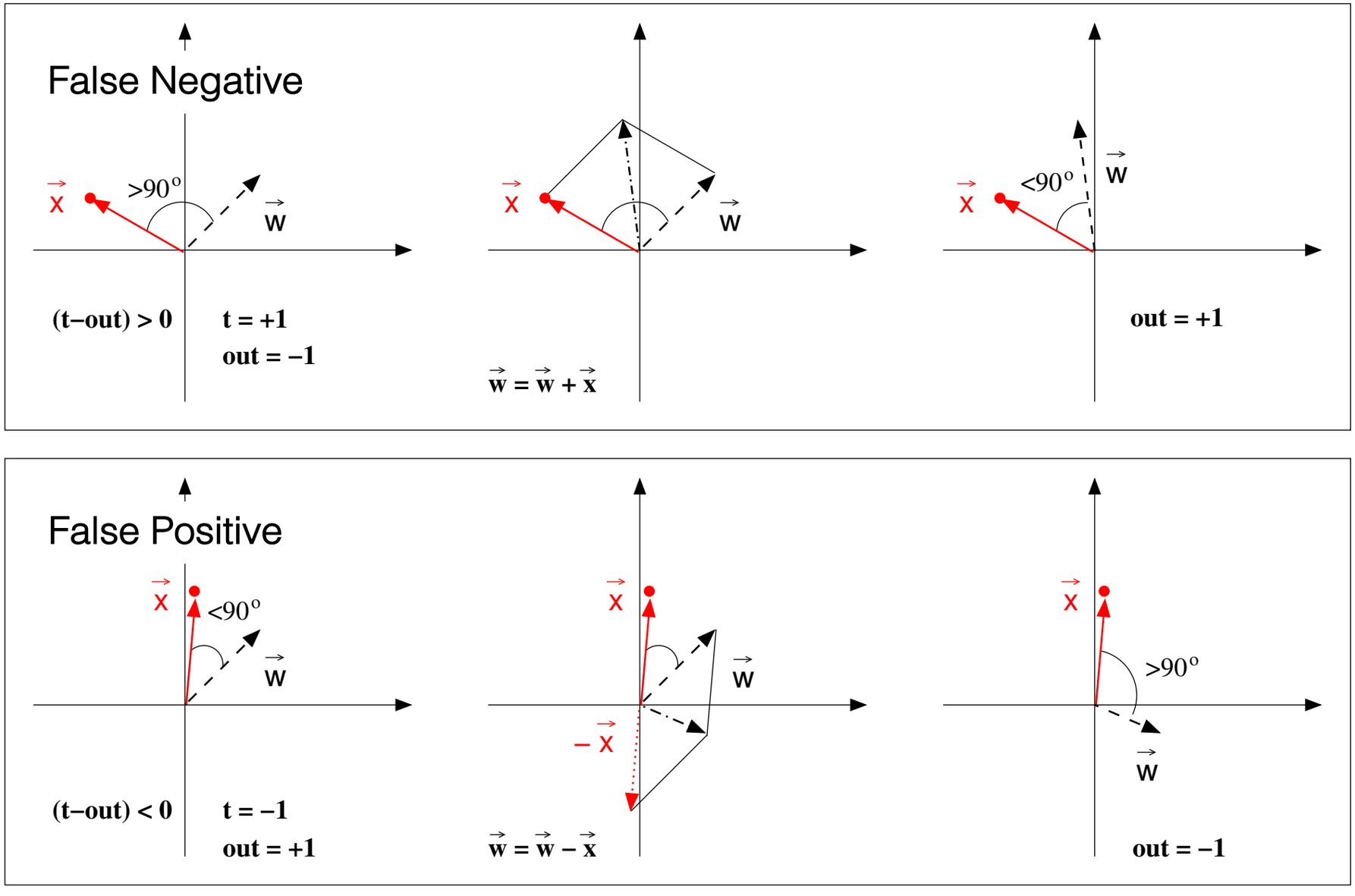
1. inizialize the weight vector $\vec{w}$ to the null vector (all components equal to 0);
2. repeat
   - 2.1 select (at random) one training example $(\vec{x}, t)$
   - 2.2 if $out = sign(\vec{w} \cdot \vec{x}) \neq t$ then

$$\vec{w} \leftarrow \vec{w} + \eta(t - out)\vec{x}$$

where η is the learning rate.

Example of Learning Algorithm

Geometric interpretation



Example of Execution

Tr:

ex.	$\vec{x}$	target
1	(4,5)	1
2	(6,1)	1
3	(4,1)	-1
4	(1,2)	-1

threshold $\longrightarrow$

ex.	$\vec{x}'$	target
1'	(1,4,5)	1
2'	(1,6,1)	1
3'	(1,4,1)	-1
4'	(1,1,2)	-1

weights vector: $\vec{w} = (w_0, w_1, w_2)$

assume to start with "random" weights: $\vec{w} = (0, 1, -1)$ and $\eta = \frac{1}{2}$

$$\vec{w} \leftarrow \vec{w} + \eta(t - out)\vec{x}$$

AL

weights	input	target	out	error	new weights
(0,1,-1)	(1,4,5)	1	-1	2	(1,5,4)
(1,5,4)	(1,6,1)	1			
	(1,4,1)	-1			
	(1,1,2)	-1			

Example of Execution

Tr:

ex.	$\vec{x}$	target
1	(4,5)	1
2	(6,1)	1
3	(4,1)	-1
4	(1,2)	-1

threshold $\longrightarrow$

ex.	$\vec{x}'$	target
1'	(1,4,5)	1
2'	(1,6,1)	1
3'	(1,4,1)	-1
4'	(1,1,2)	-1

weights vector: $\vec{w} = (w_0, w_1, w_2)$

assume to start with "random" weights: $\vec{w} = (0, 1, -1)$ and $\eta = \frac{1}{2}$

$$\vec{w} \leftarrow \vec{w} + \eta(t - out)\vec{x}$$

AL

weights	input	target	out	error	new weights
(0,1,-1)	(1,4,5)	1	-1	2	(1,5,4)
(1,5,4)	(1,6,1)	1	1	0	No changes
	(1,4,1)	-1			
	(1,1,2)	-1			

Example of Execution

Tr:

ex.	$\vec{x}$	target
1	(4,5)	1
2	(6,1)	1
3	(4,1)	-1
4	(1,2)	-1

threshold $\longrightarrow$

ex.	$\vec{x}'$	target
1'	(1,4,5)	1
2'	(1,6,1)	1
3'	(1,4,1)	-1
4'	(1,1,2)	-1

weights vector: $\vec{w} = (w_0, w_1, w_2)$

assume to start with "random" weights: $\vec{w} = (0, 1, -1)$ and $\eta = \frac{1}{2}$

$$\vec{w} \leftarrow \vec{w} + \eta(t - out)\vec{x}$$

AL

weights	input	target	out	error	new weights
(0,1,-1)	(1,4,5)	1	-1	2	(1,5,4)
(1,5,4)	(1,6,1)	1	1	0	No changes
(1,5,4)	(1,4,1)	-1	1	-2	(0,1,3)
	(1,1,2)	-1			

Example of Execution

Tr:

ex.	$\vec{x}$	target
1	(4,5)	1
2	(6,1)	1
3	(4,1)	-1
4	(1,2)	-1

threshold $\longrightarrow$

ex.	$\vec{x}'$	target
1'	(1,4,5)	1
2'	(1,6,1)	1
3'	(1,4,1)	-1
4'	(1,1,2)	-1

weights vector: $\vec{w} = (w_0, w_1, w_2)$

assume to start with "random" weights: $\vec{w} = (0, 1, -1)$ and $\eta = \frac{1}{2}$

$$\vec{w} \leftarrow \vec{w} + \eta(t - out)\vec{x}$$

AL

weights	input	target	out	error	new weights
(0,1,-1)	(1,4,5)	1	-1	2	(1,5,4)
(1,5,4)	(1,6,1)	1	1	0	No changes
(1,5,4)	(1,4,1)	-1	1	-2	(0,1,3)
(0,1,3)	(1,1,2)	-1	1	-2	(-1,0,1)

Example of Execution

1

weights $\vec{w}$	input $\vec{x}'$	target t	out out	error $(t-out)$	new weights $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t-out)\vec{x}'$
(0,1,-1)	(1,4,5)	1	-1	2	(1,5,4)
(1,5,4)	(1,6,1)	1	1	0	no change
(1,5,4)	(1,4,1)	-1	1	-2	(0,1,3)
(0,1,3)	(1,1,2)	-1	1	-2	(-1,0,1)

2

weights $\vec{w}$	input $\vec{x}'$	target t	out out	error $(t-out)$	new weights $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t-out)\vec{x}'$
(-1,0,1)	(1,4,5)	1	1	0	no change
(-1,0,1)	(1,6,1)	1	? (-1)	2	(0,6,2)
(0,6,2)	(1,4,1)	-1	1	-2	(-1,2,1)
(-1,2,1)	(1,1,2)	-1	1	-2	(-2,1,-1)

3

weights $\vec{w}$	input $\vec{x}'$	target t	out out	error $(t-out)$	new weights $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t-out)\vec{x}'$
(-2,1,-1)	(1,4,5)	1	-1	2	(-1,5,4)
(-1,5,4)	(1,6,1)	1	1	0	no change
(-1,5,4)	(1,4,1)	-1	1	-2	(-2,1,3)
(-2,1,3)	(1,1,2)	-1	1	-2	(-3,0,1)

4

weights $\vec{w}$	input $\vec{x}'$	target t	out out	error $(t-out)$	new weights $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t-out)\vec{x}'$
(-3,0,1)	(1,4,5)	1	1	0	no change
(-3,0,1)	(1,6,1)	1	-1	2	(-2,6,2)
(-2,6,2)	(1,4,1)	-1	1	-2	(-3,2,1)
(-3,2,1)	(1,1,2)	-1	1	-2	(-4,1,-1)

5

weights $\vec{w}$	input $\vec{x}'$	target t	out out	error $(t-out)$	new weights $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t-out)\vec{x}'$
(-4,1,-1)	(1,4,5)	1	-1	2	(-3,5,4)
(-3,5,4)	(1,6,1)	1	1	0	no change
(-3,5,4)	(1,4,1)	-1	1	-2	(-4,1,3)
(-4,1,3)	(1,1,2)	-1	1	-2	(-5,0,1)

6

weights $\vec{w}$	input $\vec{x}'$	target t	out out	error $(t-out)$	new weights $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t-out)\vec{x}'$
(-5,0,1)	(1,4,5)	1	? (-1)	2	(-4,4,6)
(-4,4,6)	(1,6,1)	1	1	0	no change
(-4,4,6)	(1,4,1)	-1	1	-2	(-5,0,5)
(-5,0,5)	(1,1,2)	-1	1	-2	(-6,-1,3)

7

weights $\vec{w}$	input $\vec{x}'$	target t	out out	error $(t-out)$	new weights $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t-out)\vec{x}'$
(-6,-1,3)	(1,4,5)	1	1	0	no change
(-6,-1,3)	(1,6,1)	1	-1	2	(-5,5,4)
(-5,5,4)	(1,4,1)	-1	1	-2	(-6,1,3)
(-6,1,3)	(1,1,2)	-1	1	-2	(-7,0,1)

8

weights $\vec{w}$	input $\vec{x}'$	target t	out out	error $(t-out)$	new weights $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t-out)\vec{x}'$
(-7,0,1)	(1,4,5)	1	-1	2	(-6,4,6)
(-6,4,6)	(1,6,1)	1	1	0	no change
(-6,4,6)	(1,4,1)	-1	1	-2	(-7,0,5)
(-7,0,5)	(1,1,2)	-1	1	-2	(-8,-1,3)

9

weights $\vec{w}$	input $\vec{x}'$	target t	out out	error $(t-out)$	new weights $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t-out)\vec{x}'$
(-8,-1,3)	(1,4,5)	1	1	0	no change
(-8,-1,3)	(1,6,1)	1	-1	2	(-7,5,2)
(-7,5,2)	(1,4,1)	-1	1	-2	(-8,1,3)
(-8,1,3)	(1,1,2)	-1	-1	0	no change

10

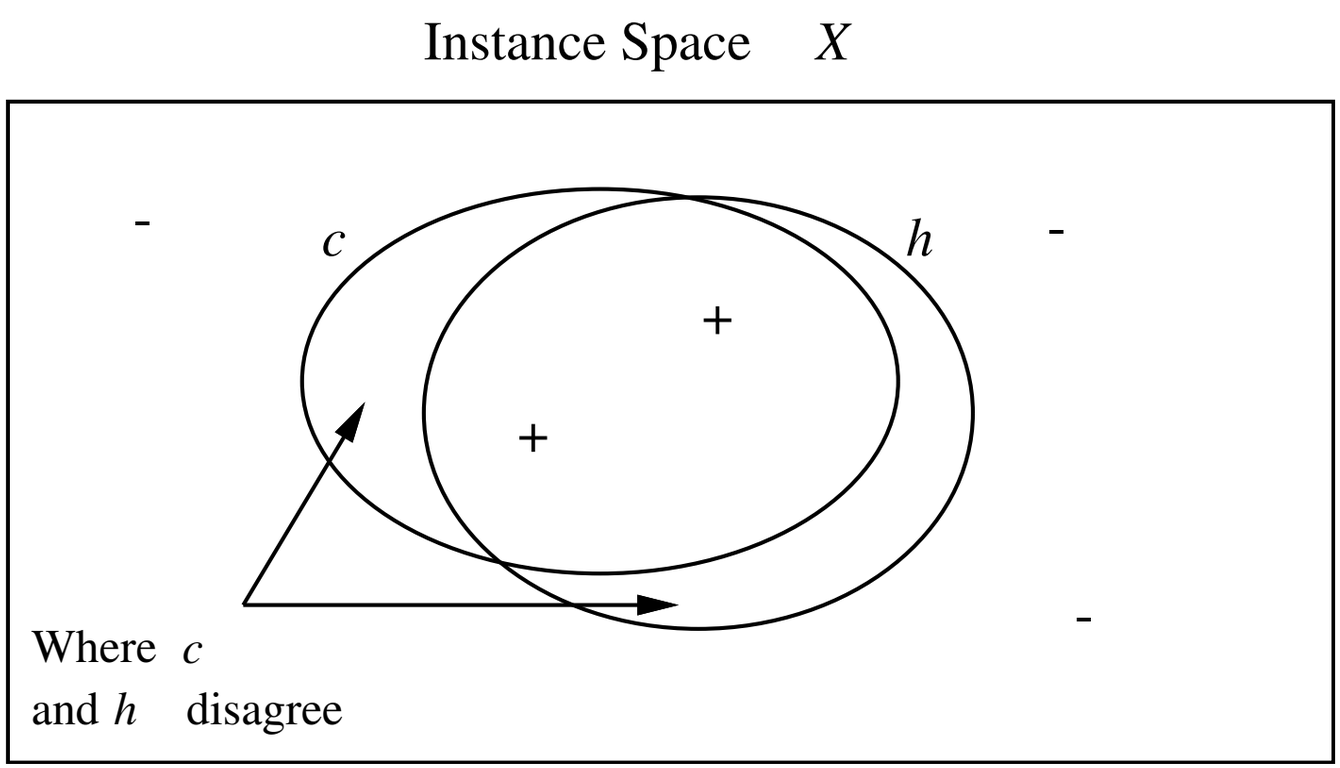
weights $\vec{w}$	input $\vec{x}'$	target t	out out	error $(t-out)$	new weights $\vec{w} \leftarrow \vec{w} + \frac{1}{2}(t-out)\vec{x}'$
(-8,1,3)	(1,4,5)	1	1	0	no change
(-8,1,3)	(1,6,1)	1	1	0	no change
(-8,1,3)	(1,4,1)	-1	-1	0	no change
(-8,1,3)	(1,1,2)	-1	-1	0	no change

WARNING: Perceptron can only learn linearly separable data!

Notions of Statistical Learning Theory

- The dataset we have is a random sample identically and independently distributed according to some probability distribution $\mathcal{D}$
- In general, we are interested in generalization!
- E.g. Emotion detection system from faces.
  - Training set: pictures of your faces expressing different emotions
  - Goal: classify emotions of other people!

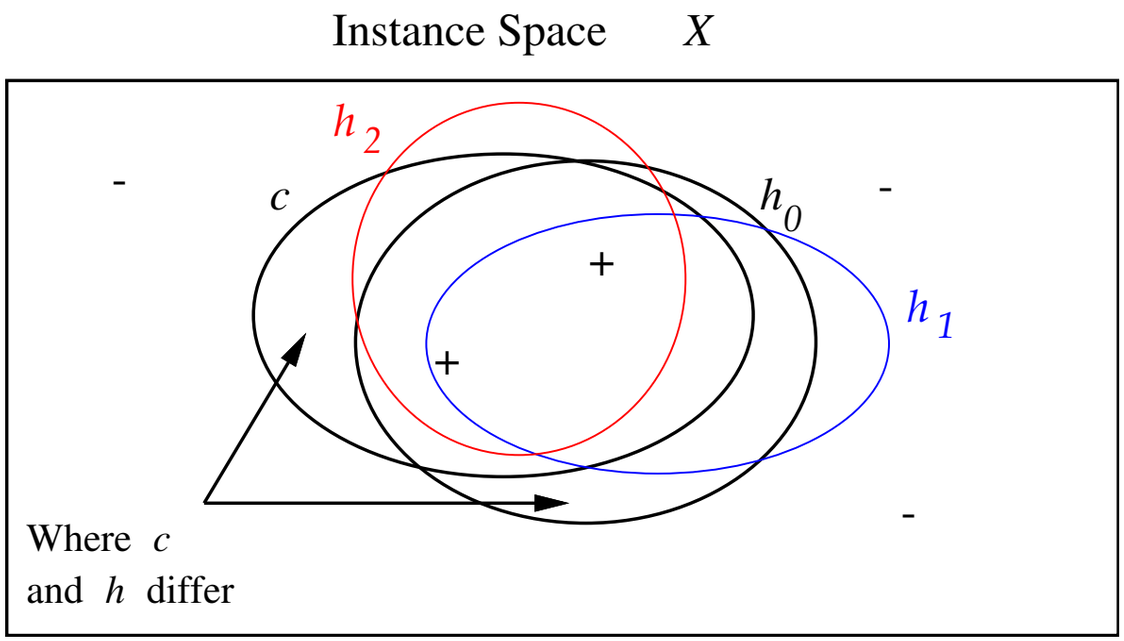
True Error



Def: The **True Error** ($error_{\mathcal{D}}(h)$) of hypothesis h with respect to target concept c and distribution $\mathcal{D}$ (to observe an input instance $x \in X$) is the probability that h will misclassify an instance drawn at random according to $\mathcal{D}$:

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)]$$

Empirical Error

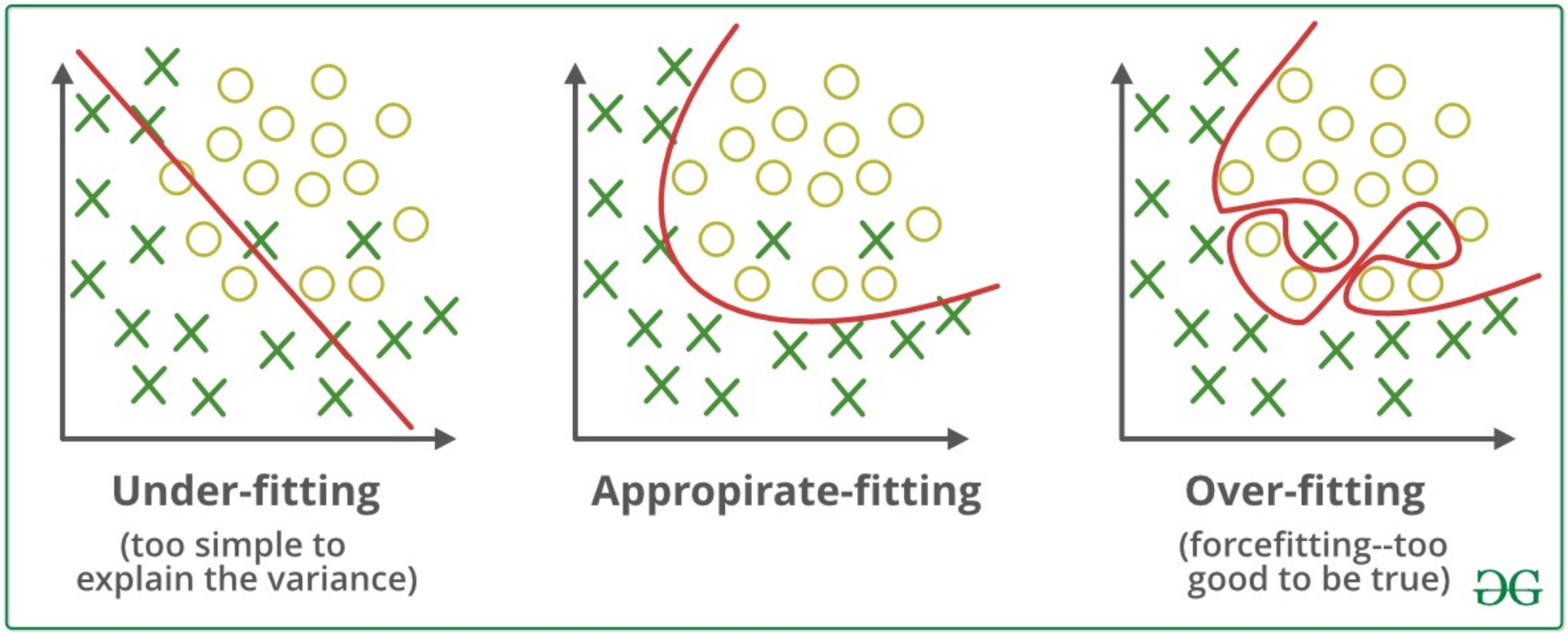


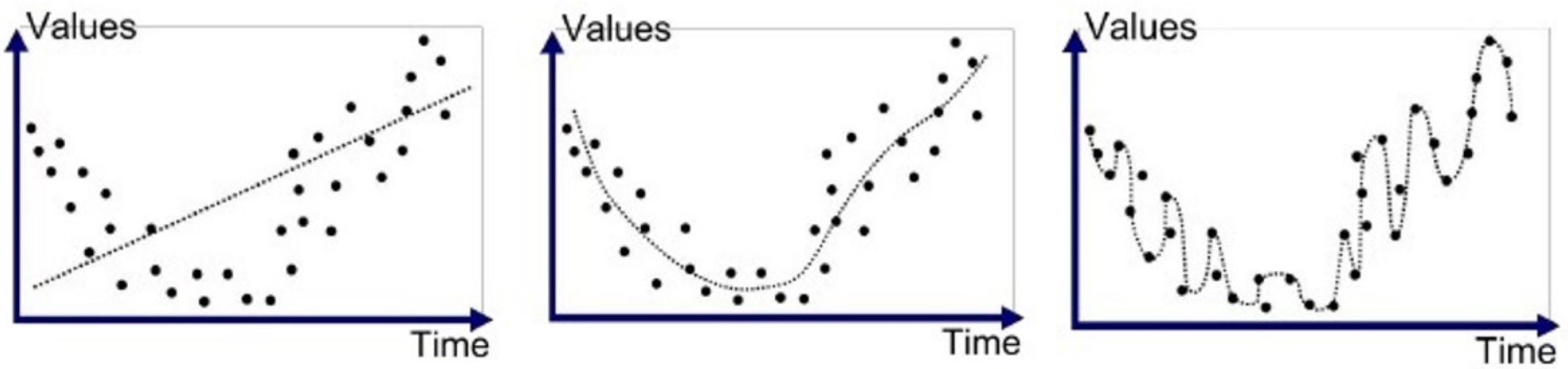
Def: The **Empirical Error** ($error_{Tr}(h)$) of hypothesis h with respect to Tr is the number of examples that h misclassifies:

$$error_{Tr}(h) = Pr_{(x,f(x))\in Tr}\left[f(x) \neq h(x)\right] = \frac{|\{(x, f(x)) \in Tr \,|f(x) \neq h(x)\}|}{|Tr|}$$

Def: $h \in \mathcal{H}$ overfits Tr if $\exists h' \in \mathcal{H}$ such that $error_{Tr}(h) < error_{Tr}(h')$, but $error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$.

Overfitting





Underfitted

Good Fit/Robust

Overfitted

Estimating the True error

- Minimizing the error on the training set (Empirical Risk Minimization) may not be the best option (see overfitting later)

We want to minimize the true error!

$$error_D(h) = error_{Tr}(h) + generalization(h)$$

2 ways: bounds and estimation

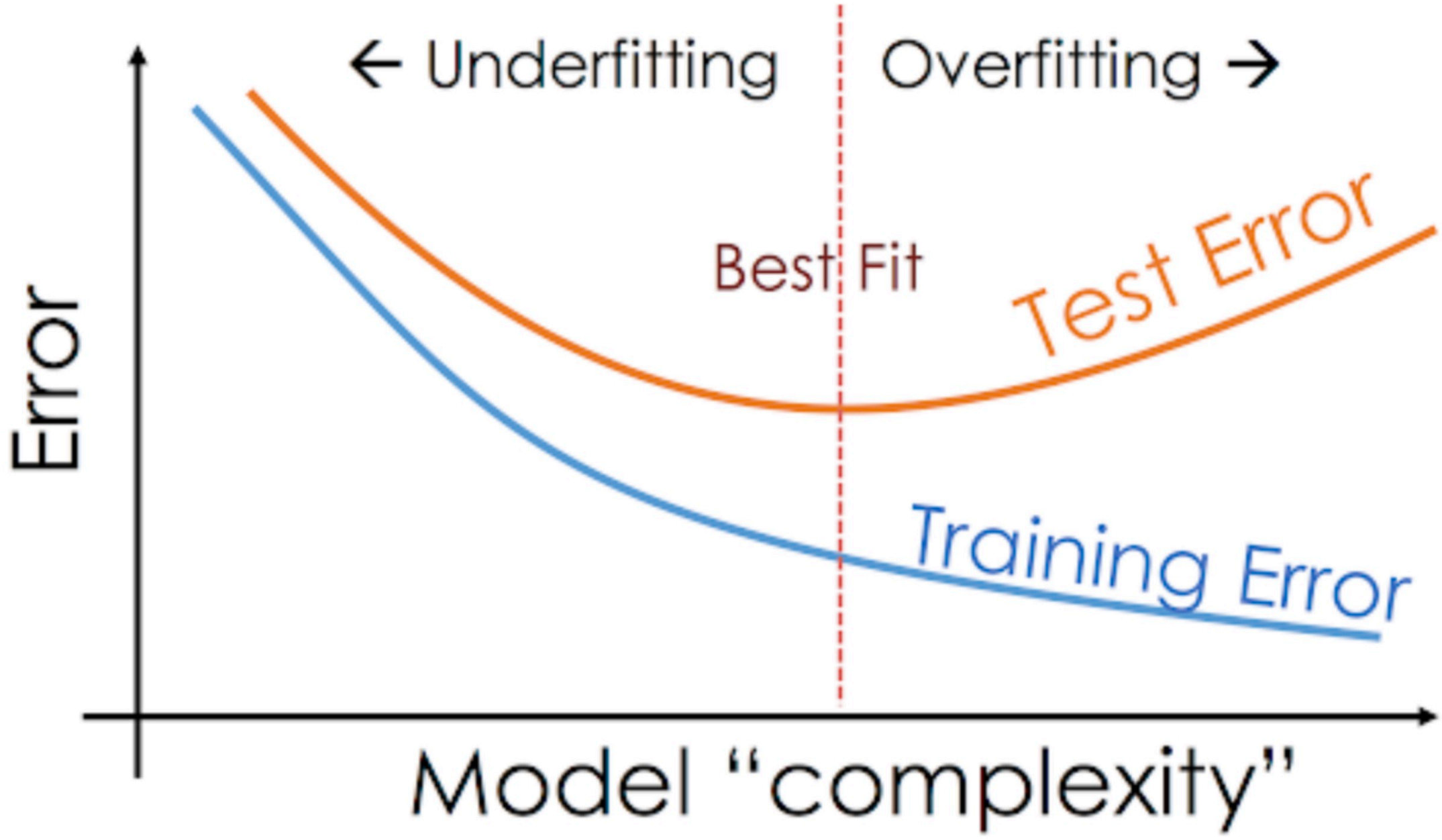
1. relate the Empirical error and the true error with generalization bounds:

   $$error_{\mathcal{D}}(h) \leq error_{Tr}(h) + complexityMeasure(\mathcal{H})$$

   with $\mathrm{h} \in \mathcal{H}$, exploiting some complexity measure of the hypothesis space

2. compute the error on unseen data (TEST set)

Overfitting - 2



Model Selection and Validation Set

We can hold out some of our original training data

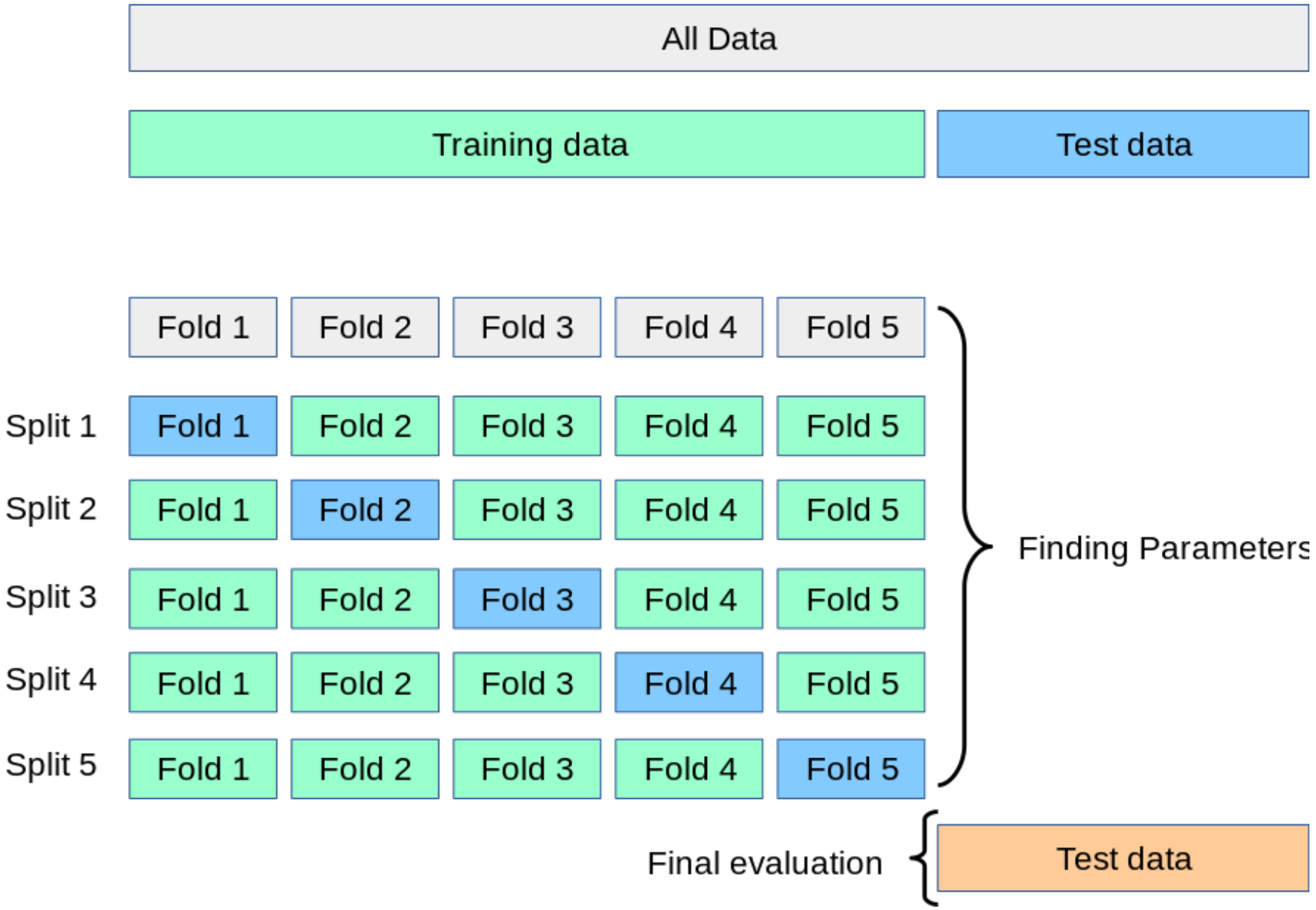
Hold-out procedure

1. A small subset of Tr, called the validation set (or hold-out set), denoted Va, is identified
2. A classifier/regressor is learnt using examples in $Tr - Va$
3. Step 2 is performed with different values of the parameter(s) (in our example, p), and tested against the hold-out sample

In an operational setting, after parameter optimization, one typically re-trains the classifier on the entire training corpus, in order to boost effectiveness (debatable step!)

It is possible to show that the evaluation performed in Step 2 gives an unbiased estimate of the error performed by a classifier learnt with the same parameter(s) and with training set of cardinality $|Tr| - |Va| < |Tr|$

K-fold Cross Validation



K-fold Cross Validation

An alternative approach to model selection (and evaluation) is the K-fold cross-validation method

K-fold CV procedure

1. K different classifiers/regressors $h_1, h_2, \ldots, h_k$ are built by partitioning the initial corpus Tr into k disjoint sets $Va_1, \ldots, Va_k$ and then iteratively applying the Hold-out approach on the k-pairs $(Tr_i = Tr - Va_i, Va_i)$
2. Final error is obtained by individually computing the errors of $h_1, \ldots, h_k$, and then averaging the individual results

The above procedure is repeated for different values of the parameter(s) and the setting (model) with smaller final error is selected

The special case $k = |Tr|$ of k-fold cross-validation is called **leave-one-out** cross-validation

Model selection and error estimation

- Training phase:
  - Select the appropriate set of hyperparameters (with corresponding hypothesis space)
  - fit the models
  - Model selection: select the best model estimating its true error (without looking at test data)
- Testing phase
  - performance assessment (error estimation)