Machine Learning Algorithms
LVA 389.203

Basics

Markus Rupp

3.9.2020

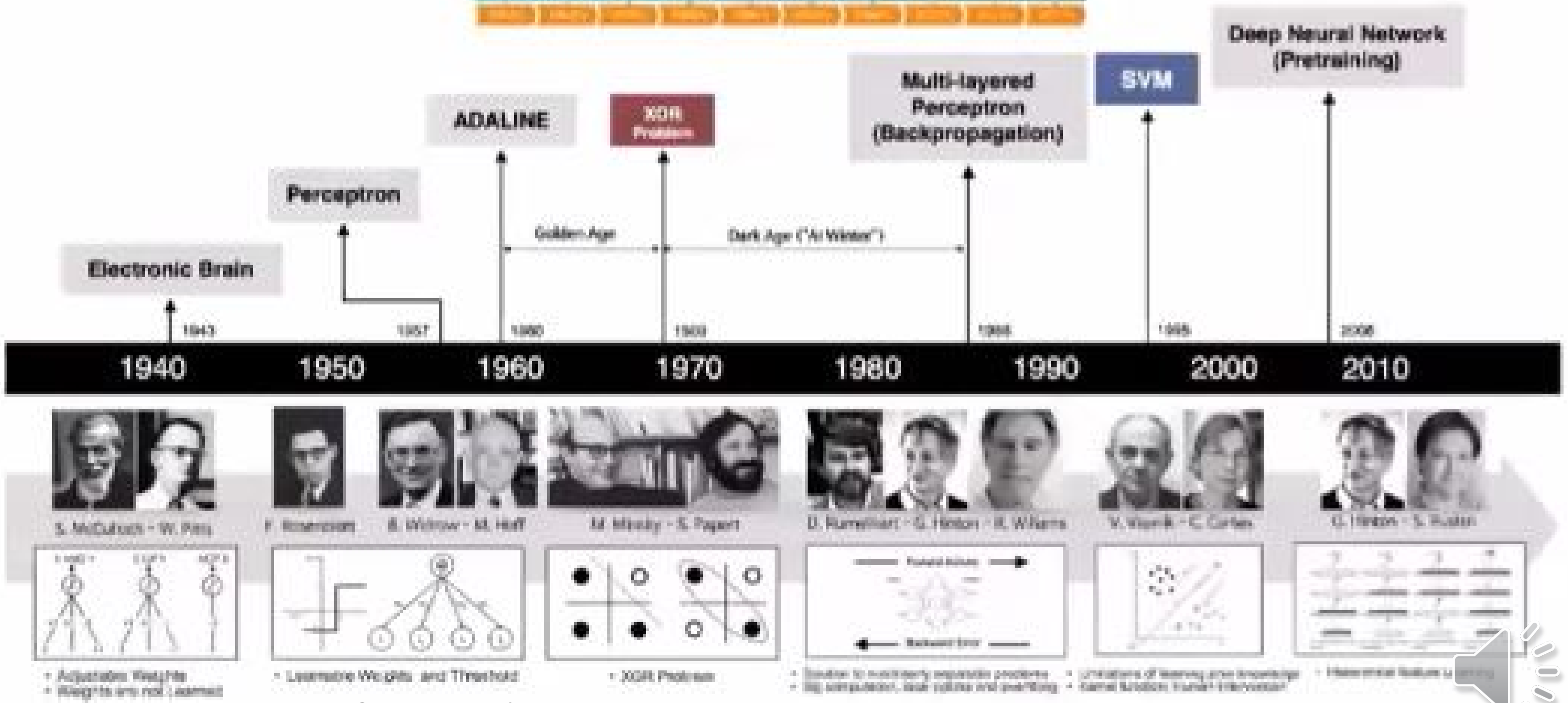
Literature

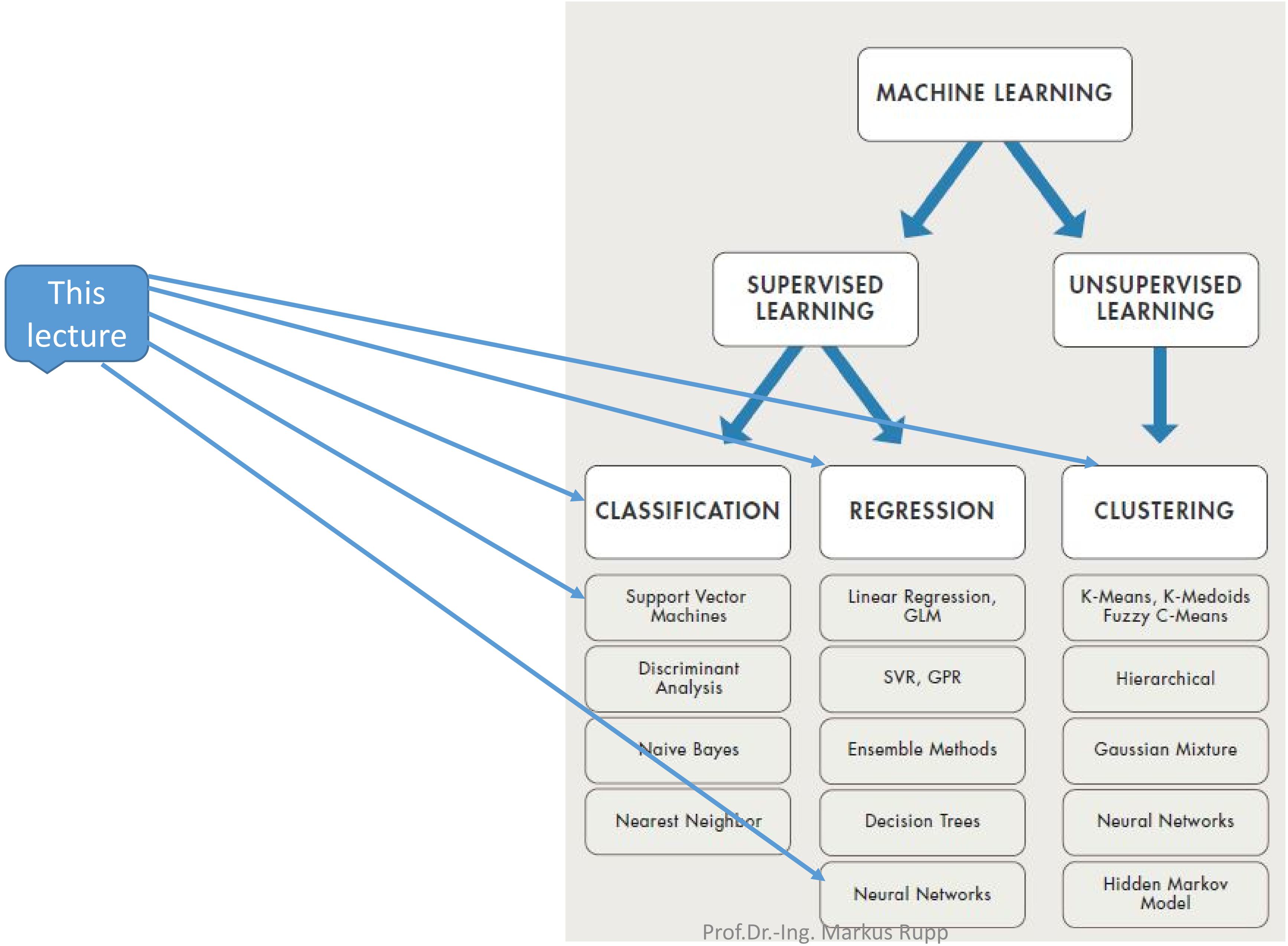
- Watt, Borhani, Katsegallos, „Machine Learning Refined“ (1.st and 2nd edition)
- M.Rupp, „Script to Adaptive filtering course“

- Sergios Theodoridis, **Machine Learning: A Bayesian and Optimization Perspective, 2nd edition**
- Ali Sayed, „Fundamentals of adaptive Filtering“

Machine Learning (ML)

- A rapidly growing technology




This lecture
MACHINE LEARNING
SUPERVISED LEARNING
UNSUPERVISED LEARNING
CLASSIFICATION
REGRESSION
CLUSTERING
Support Vector Machines
Discriminant Analysis
Naive Bayes
Nearest Neighbor
Linear Regression, GLM
SVR, GPR
Ensemble Methods
Decision Trees
Neural Networks
K-Means, K-Medoids Fuzzy C-Means
Hierarchical
Gaussian Mixture
Neural Networks
Hidden Markov Model

Machine Learning

- Not entirely new topic, well researched in 80-90ies
- But some important breakthroughs in recent years.

- Interpretations based on statistics or pure data
- Strong relation to „Big Data“

Data Type	Feature Selection Task	Techniques
Sensor data	Extract signal properties from raw sensor data to create higher-level information	**Peak analysis** – perform an fft and identify dominant frequencies **Pulse and transition metrics** – derive signal characteristics such as rise time, fall time, and settling time **Spectral measurements** – plot signal power, bandwidth, mean frequency, and median frequency
Image and video data	Extract features such as edge locations, resolution, and color	**Bag of visual words** – create a histogram of local image features, such as edges, corners, and blobs **Histogram of oriented gradients (HOG)** – create a histogram of local gradient directions **Minimum eigenvalue algorithm** – detect corner locations in images **Edge detection** – identify points where the degree of brightness changes sharply
Transactional data	Calculate derived values that enhance the information in the data	**Timestamp decomposition** – break timestamps down into components such as day and month **Aggregate value calculation** – create higher-level features such as the total number of times a particular event occurred

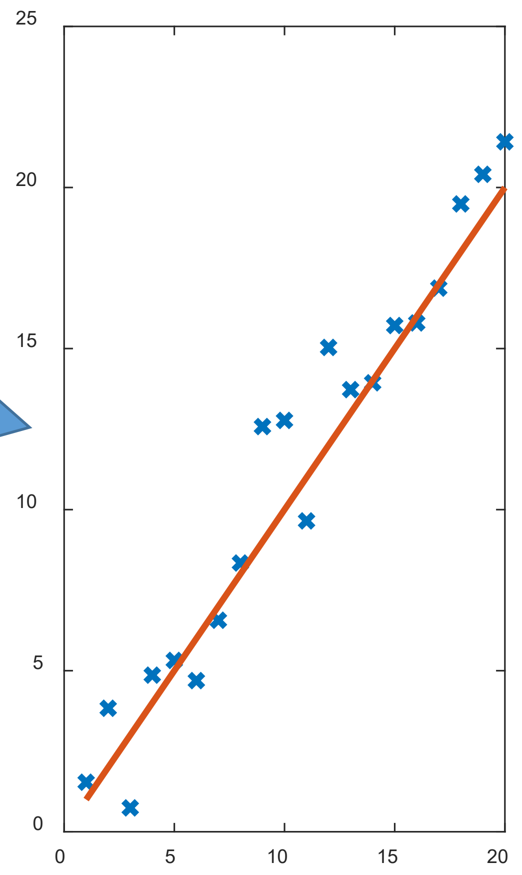
How do we learn?

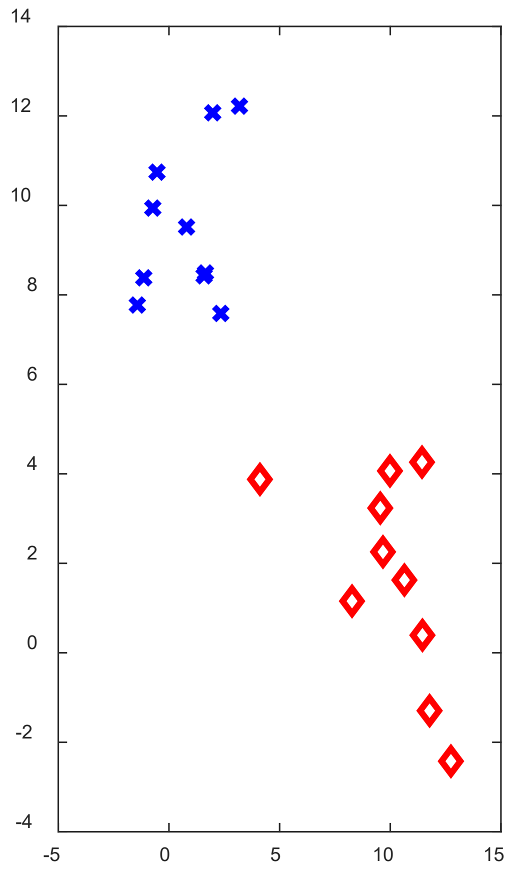
- Learning by understanding: proof a theorem
- Learning by heart: learn a poem
- Conceptual learning: how to drive a car

- Machine learning = categorizing
  - Supervised learning: categories a priori known
  - Unsupervised learning: algorithm has to find categories

Regression or classification?

Regression:
Output continuous
Can be used for
prediction





Classification:
Output discrete
Can be used for
discrimination

(linear) Regression

- Assume a set of pairs ($\underline{x}_i$,y_i) i=1,2,….,N in which we call
  - $\underline{x}_i$: Independent variables, regressor, input
  - y_i: Dependent variables, regressand, observation
- We like to find a linear combination such that

$$y_i = c_0 + c_1 x_{i,1} + \ldots + c_M x_{i,M} + v_i$$
$$= \left[1, \underline{x}_i^H\right] \underline{c} + v_i = \underline{\widehat{x}}_i^H \underline{c} + v_i$$

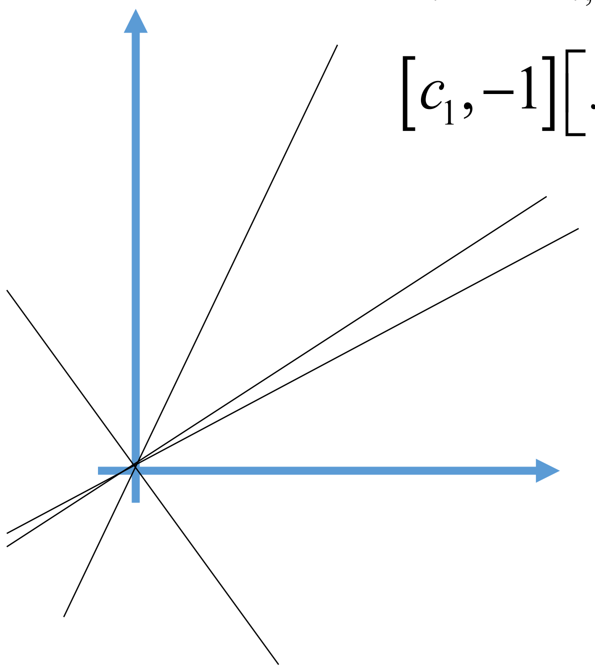
  - v_i : disturbance, perturbation, error

for all i=1,2,…,N

Why do we need a Bias term?

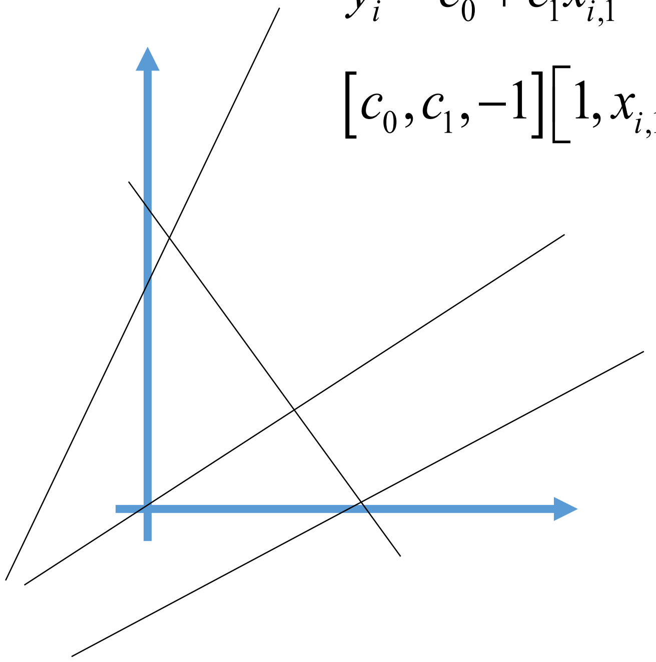
$$y_i = c_1 x_{i,1}$$

$$[c_1, -1][x_{i,1}, y_i]^T = 0$$



$$y_i = c_0 + c_1 x_{i,1}$$

$$[c_0, c_1, -1][1, x_{i,1}, y_i]^T = 0$$



(linear) Regression

- We like to minimize all error terms v_i in a LS manner, i.e.,

$$\min \sum_{i=1}^{N} |v_i|^2 = \min \|\underline{v}\|^2 = \min_{\underline{c}} \sum_{i=1}^{N} \left| y_i - \underline{\widehat{x}}_i^H \underline{c} \right|^2$$

$$= \min_{\underline{c}} \left\| \underline{y} - X \underline{c} \right\|^2$$

$$\underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}; \qquad X = \begin{bmatrix} \underline{\widehat{x}}_1^H \\ \underline{\widehat{x}}_2^H \\ \vdots \\ \underline{\widehat{x}}_N^H \end{bmatrix} \in C^{N \times (M+1)}$$

(linear) Regression

- The LS solution is simply given by

$$\underline{c}_{LS} = \begin{cases} \left[X^H X \right]^{-1} X^H \underline{y} & ; N \geq M+1 \\ X^H \left[X X^H \right]^{-1} \underline{y} & ; else \end{cases}$$

(Non)-linear Regression

- By extending the basis towards nonlinear elements, it is straightforward to find a **nonlinear representation**
- For example,

$$
\begin{aligned}
y_i = c_0 + c_1 x_{i,1} + \ldots + c_M x_{i,M} &+ \\
+ c_{M+1} x_{i,1}^2 + \ldots + c_{2M} x_{i,M}^2 &+ \\
+ \ldots & \\
+ c_{KM+1} x_{i,1}^{K+1} + \ldots + c_{(K+1)M} x_{i,M}^{K+1} &+ v_i \\
= \left[1, \tilde{\underline{x}}_i^H\right] \underline{c} + v_i = \widehat{\underline{x}}_i^H \underline{c} + v_i &
\end{aligned}
$$

(Non)-linear Regression

- Note that the formulation remains linear in the coefficients and thus an ordinary LS problem.
- Strictly speaking, this is still a linear regression problem as the regressor remains linear in the coefficients.
- In practice, it is recommended to use an orthogonal polynomial family to avoid numerical problems with the Gramian.

Classification

Classification

In English grammar, a *split infinitive* is a construction in which one or more words come between the infinitive marker *to* and the verb (as in "*to really try* my best"). Also called a *cleft infinitive*.

- - **The English-Speaking World**
  "The English-speaking world may be divided into
- (1) those who neither know nor care what a split infinitive is;
- (2) those who do not know, but care very much;
- (3) those who know and condemn;
- (4) those who know and approve;
- (5) those who know and distinguish."
  (H.W. Fowler and Ernest Gowers, *A Dictionary of Modern English Usage*, 2nd ed. Oxford Univ. Press, 1965)

Classification

- Gallia est omnis divisa in partes tres, quarum unam incolunt Belgae, aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur.

- J. Cesar, de Bello Gallico

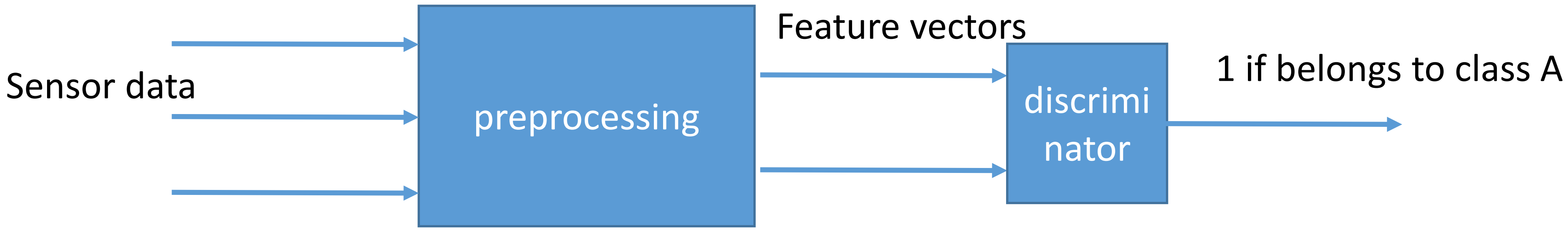
- All Gaul is divided into three parts, one of which the Belgae inhabit, the Aquitani another, those who in their own language are called Celts, in our Gauls, the third.

Classification

- Assume a set of pairs ($\underline{x}_i$,y_i) i=1,2,….,N in which we call
  - $\underline{x}_i$: input vector, observation
  - y_i: class, category
- Often a preprocessing of the input vectors is required to obtain a vector with desired features
  - Feature vectors: $f(\underline{x}_i) \rightarrow \underline{f}_i$
- The problem thus reads:
  - Given a set of pairs ($\underline{x}_i$,y_i) i=1,2,….,N for which a subset belongs to a class A (y_i=1) and the remaining part not (y_i=-1),
  - Find a hyperplane that divides the two sets

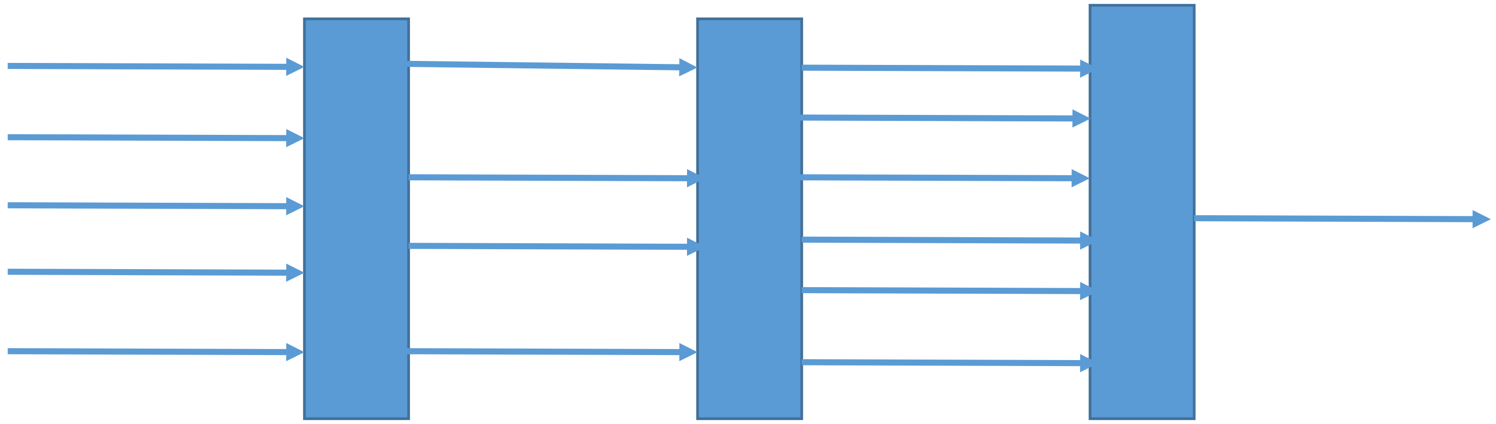
Architectures for Classification

- Simple classification network



Architectures for Classification

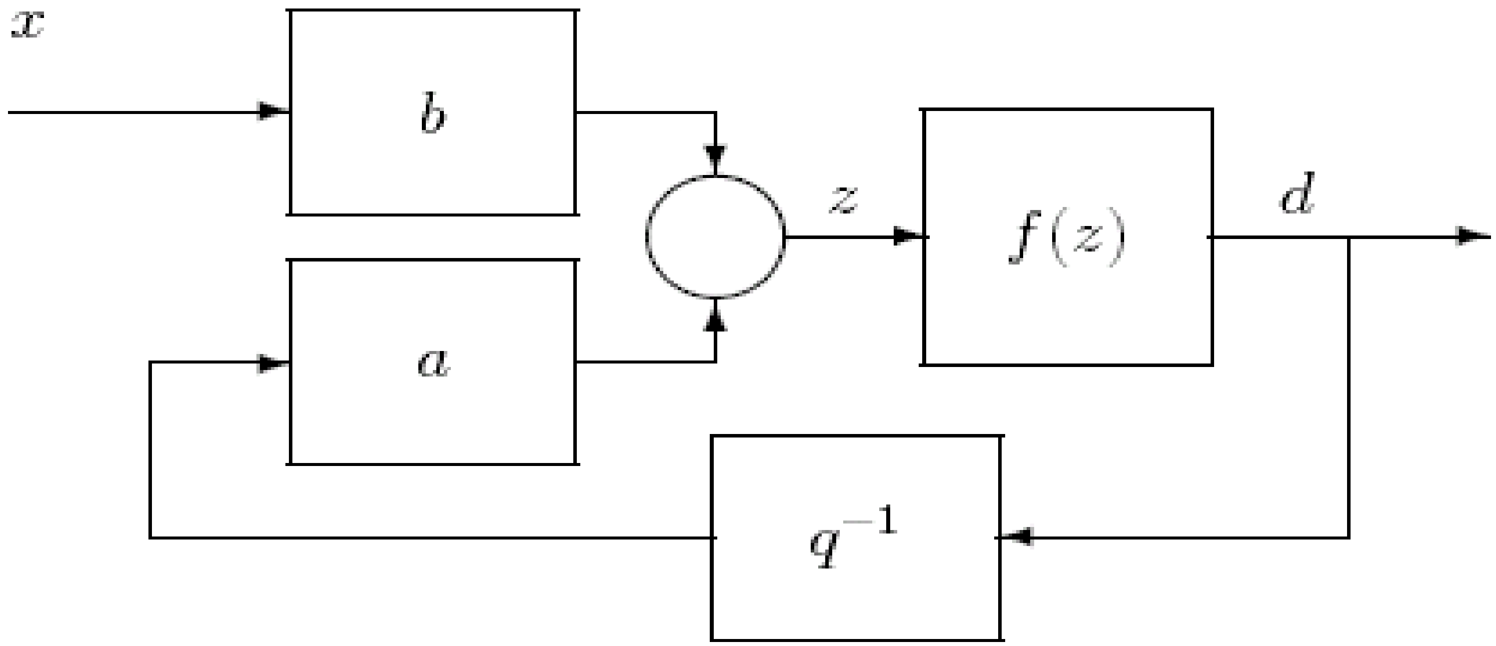
- Neural network
- Several layers of meshed and connected „neurons“



- Offers „complicated“ shapes in the feature space to distinguish

Architectures for Classification (RNN)

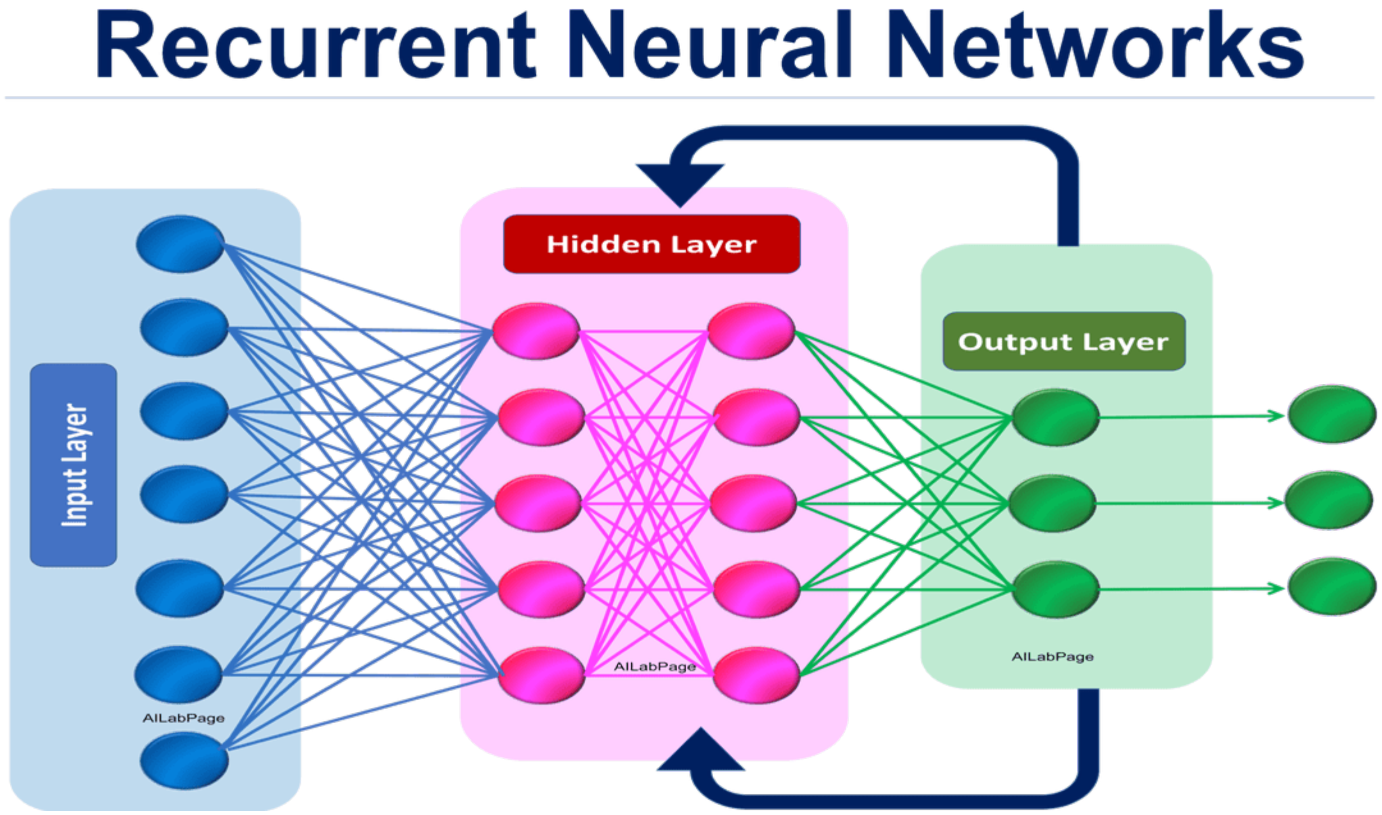
Neural networks also come with feedback, in particular if time series are processed (speech detector)



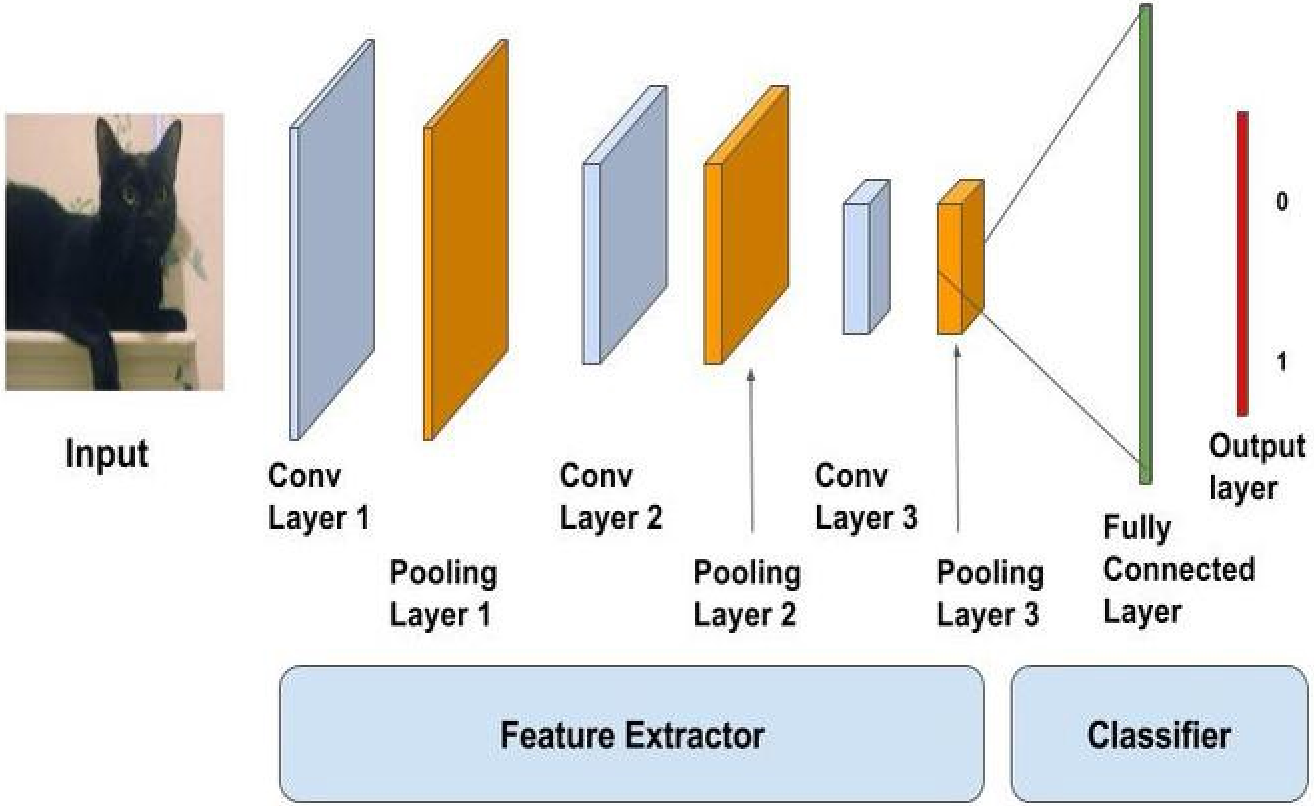
Architectures: Speed up Learning

- As learning can be very tedious and time consuming, in particular if many layers are needed, architectural structures may need to be selected depending on the problem to solve.
- While recurrent networks (RNN) are often useful in speech communications taking strong correlations into account, other architectures are useful in different contexts.
- Convolution neural networks (CNN) are particularly useful in the context of image and video understanding as predefined convolutions are able to preselect typical features as edge detections. (Le Cun 2010)

RNN: Recurrent Neural Network



CNN: Convolutional Neural Network



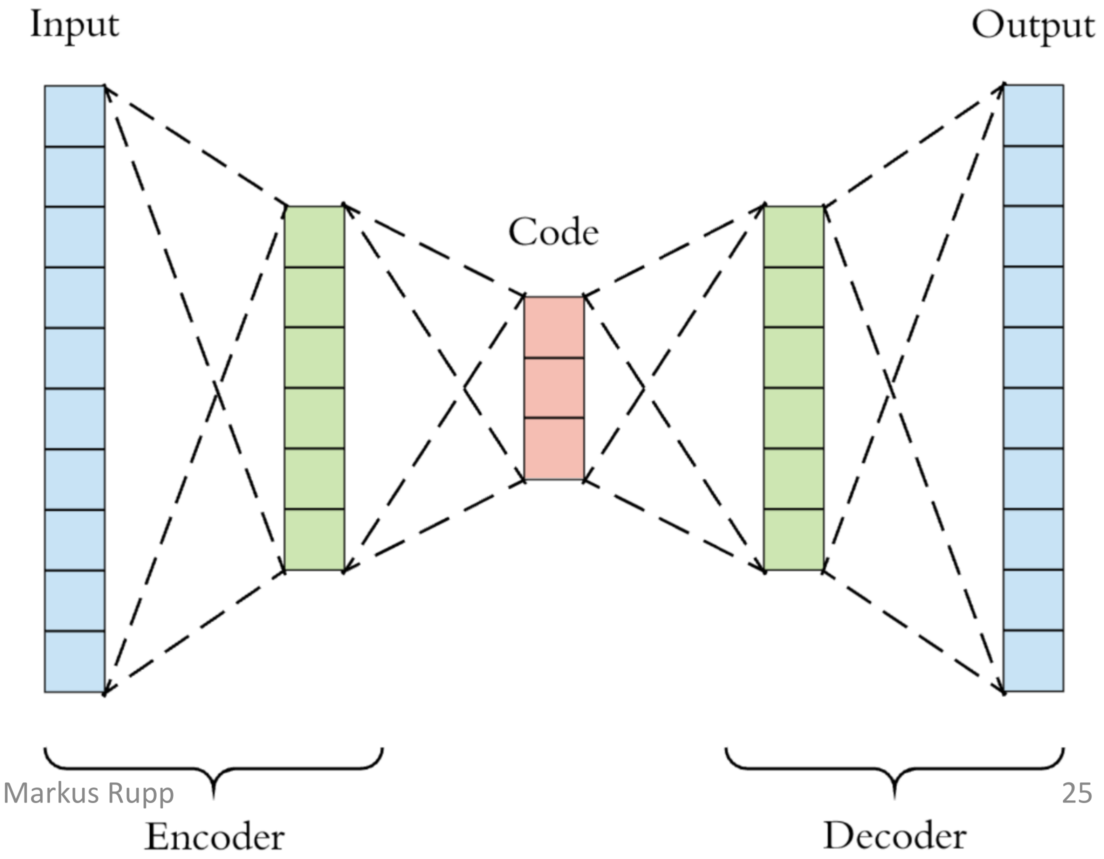
Architectures

- Essentially a neural network comprises of a number of layers with a selection of nonlinear functions in between. We could thus describe it parameterically.
- Specific to their associated problems it may make sense however to add preprocessors in between the layers, e.g. CNN with small convolutional units in between.
- To mimic an iterative method we can use „Unfolding“, i.e., rather than 6 iterations we add 6 layers, leading typically to deep neural networks (DNN). Recently many classic algorithms for communications have been described as DNN with specific architectural additions.

Architecture: Autoencoding

- While most neural nets are designed to be trained on input/output pairs of data, one structure only requires an input:

- Idea is to compress data!

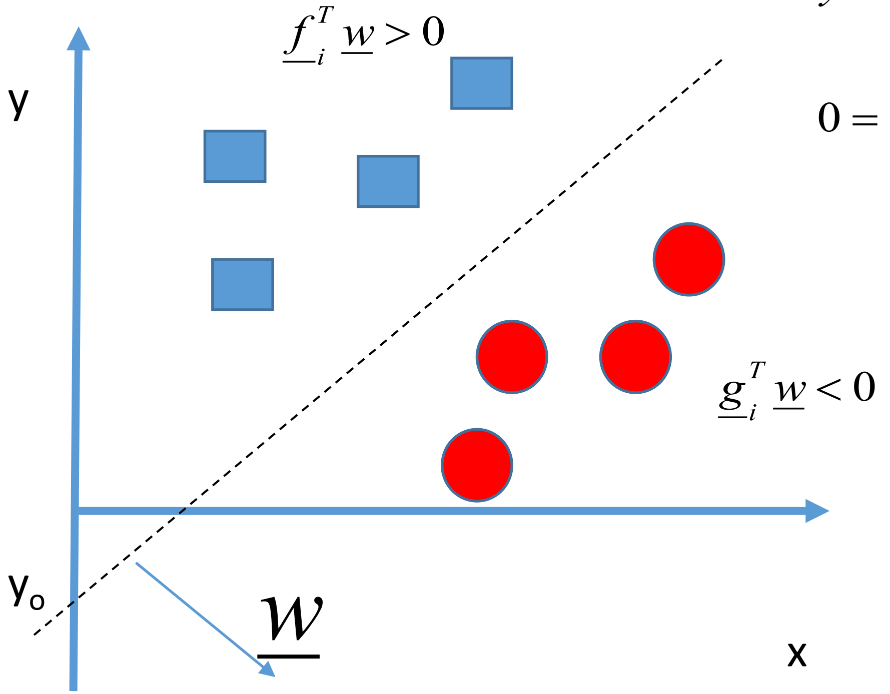


Classification

- Consider a set of objects S whose members either have a certain attribute A or not. We thus want to separate the set into two subsets S_1 and S_2.
- We assume each object s_i with attribute A has a certain set of features that are indicators for the desired attribute, organized in a vector $\underline{f}_i=[f_{i1},f_{i2},\dots,f_{iN}]$, while those objects that do not have such attribute have a similar vector $\underline{g}_i$.
- We want to find a vector $\underline{w}$ such that the inner product $<\underline{w},\underline{f}_i>$ is close to „A“ and $<\underline{w},\underline{g}_i>$ to „B=not A“.

Classification

- Consider simple example



$$y = ax + y_0$$

$$0 = [1, x, y]\begin{bmatrix} y_0 \\ a \\ -1 \end{bmatrix} = \langle \underline{\hat{f}}, \underline{w} \rangle$$

Description of line by w is not unique

We can bring in additional constraints on w, such as min||w||

w is perpendicular to the hyperplane
w defines the hyperplane

Classification

- We can thus set:

$$\underline{w}_{opt} = \arg\min_{\underline{w}} \sum_{i=1}^{N_1} \left(1-\left\langle \underline{w}, \underline{f}_i \right\rangle\right)^2 + \sum_{i=1}^{N_2} \left(-1-\left\langle \underline{w}, \underline{g}_i \right\rangle\right)^2$$

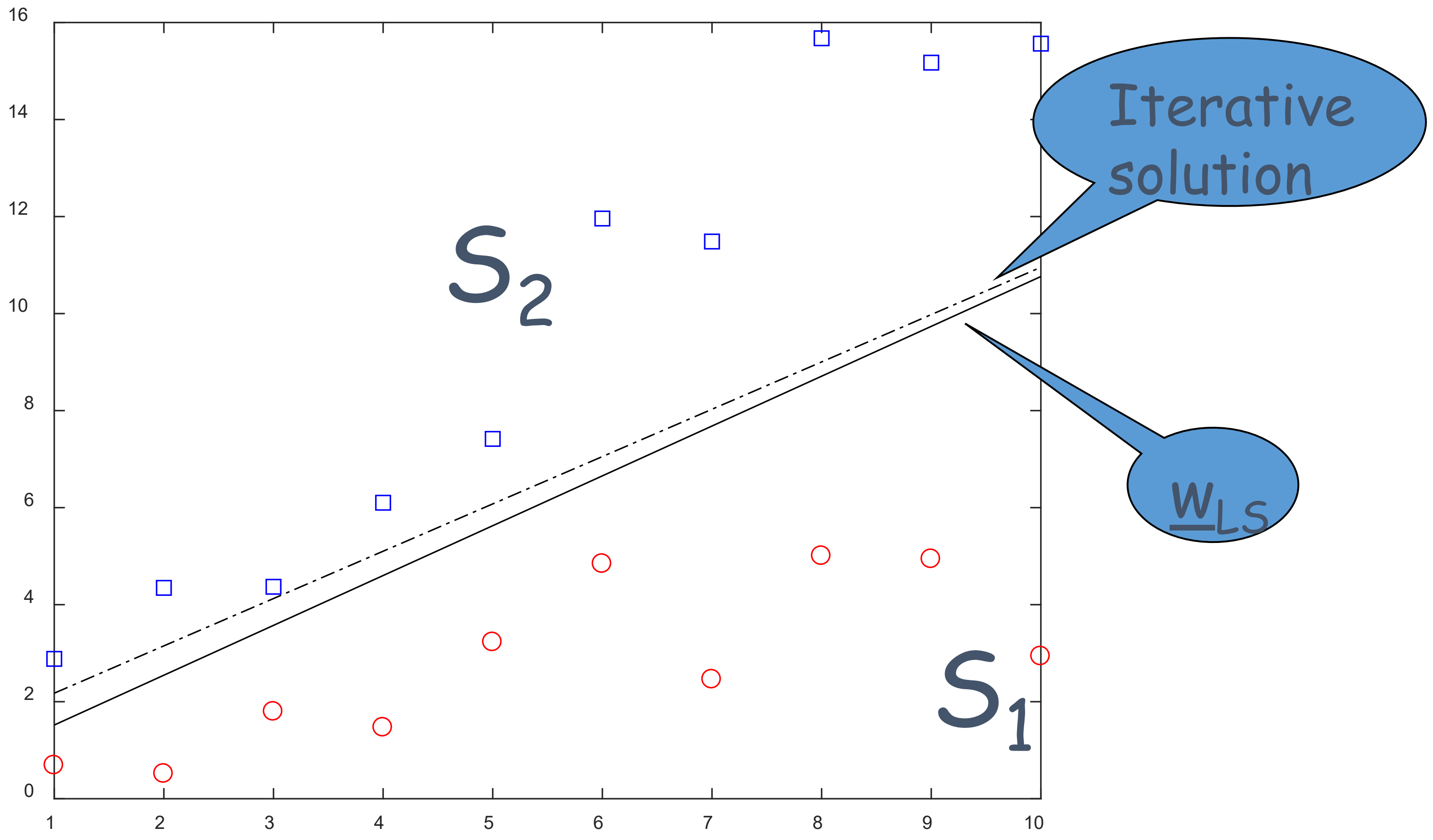
- which results in a hyperplane that separates both sets (if they are separable).
- Note however, that such a hyperplane should also be moved to the y direction (we did the same in the regression problem), which is not possible yet.
  Therefore, we add another degree of freedom:

$$\left\langle \left[w_0, \underline{w}\right], \left[1, \underline{f}_i\right] \right\rangle = \left\langle \widehat{\underline{w}}, \widehat{\underline{f}}_i \right\rangle$$

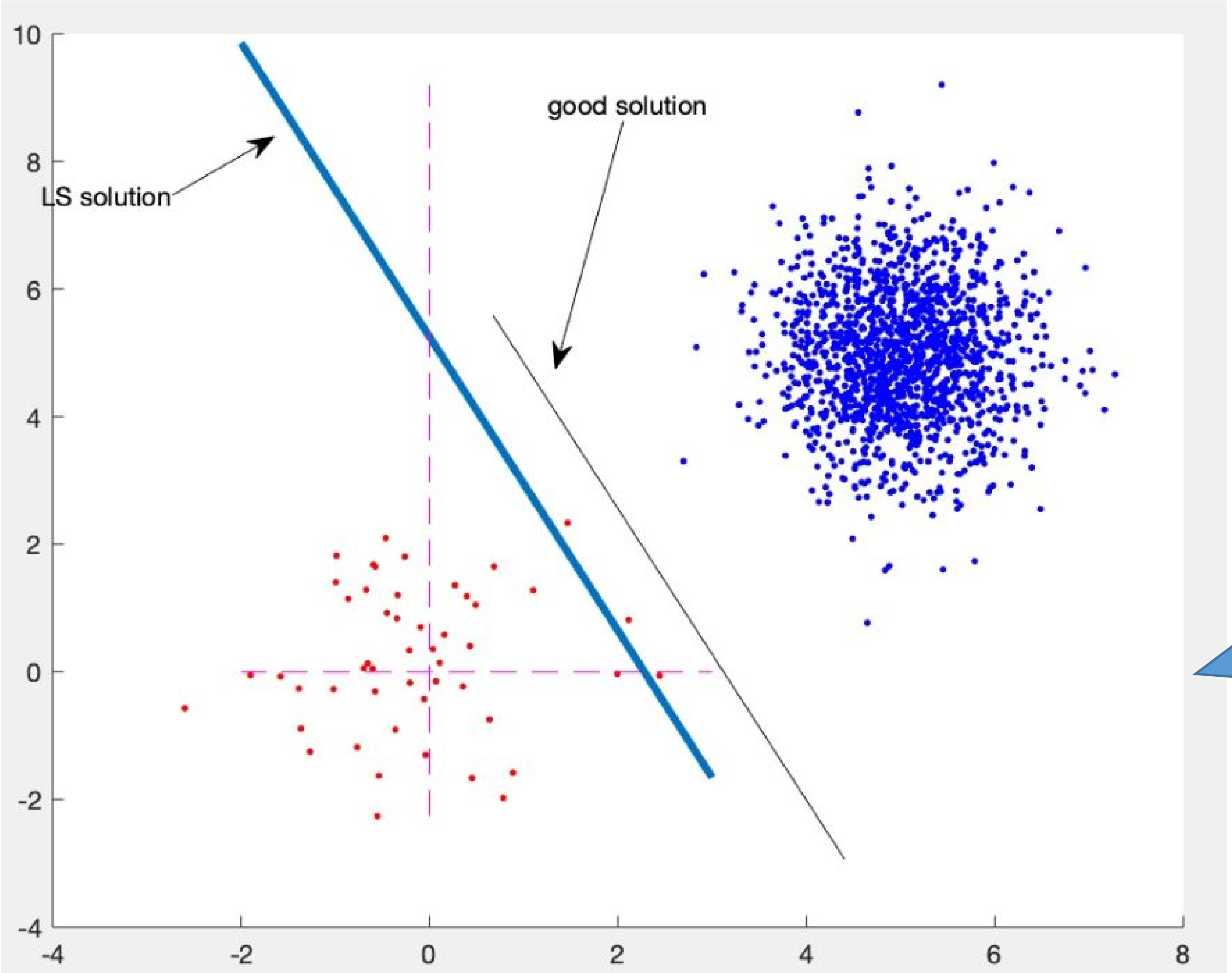
$$\underline{w}_{LS} = \arg\min_{\underline{w}} \sum_{i=1}^{N_1} \left(1-\left\langle \widehat{\underline{w}}, \widehat{\underline{f}}_i \right\rangle\right)^2 + \sum_{i=1}^{N_2} \left(-1-\left\langle \widehat{\underline{w}}, \widehat{\underline{g}}_i \right\rangle\right)^2$$

$$= \left[\sum_{i=1}^{N_1} \widehat{\underline{f}}_i \widehat{\underline{f}}_i^T + \sum_{i=1}^{N_2} \widehat{\underline{g}}_i \widehat{\underline{g}}_i^T \right]^{-1} \left(\sum_{i=1}^{N_1} \widehat{\underline{f}}_i - \sum_{i=1}^{N_2} \widehat{\underline{g}}_i \right)$$

Classification



LS can fail



Thanks to Aggelos Pikrakis

Classification

- Note that selecting „1“ and „-1“ was rather arbitrary.
- What happens if we select a and b?

$$\underline{w}_{opt} = \arg\min_{\underline{w}} \sum_{i=1}^{N_1} \left(a - \left\langle \underline{w}, \underline{f}_i \right\rangle\right)^2 + \sum_{i=1}^{N_2} \left(b - \left\langle \underline{w}, \underline{g}_i \right\rangle\right)^2$$

$$\left\langle \left[w_0, \underline{w}\right], \left[1, \underline{f}_i\right] \right\rangle = \left\langle \underline{\tilde{w}}, \underline{\tilde{f}}_i \right\rangle$$

$$\underline{w}_{LS} = \arg\min_{\underline{w}} \sum_{i=1}^{N_1} \left(a - \left\langle \underline{\tilde{w}}, \underline{\tilde{f}}_i \right\rangle\right)^2 + \sum_{i=1}^{N_2} \left(-b - \left\langle \underline{\tilde{w}}, \underline{\tilde{g}}_i \right\rangle\right)^2$$

$$= \left[\sum_{i=1}^{N_1} \underline{\tilde{f}}_i \underline{\tilde{f}}_i^T + \sum_{i=1}^{N_2} \underline{\tilde{g}}_i \underline{\tilde{g}}_i^T\right]^{-1} \left(b \sum_{i=1}^{N_2} \underline{\tilde{g}}_i + a \sum_{i=1}^{N_1} \underline{\tilde{f}}_i\right)$$

Classification

- Scaling (a,b)$\rightarrow \alpha$ (a,b) results in a scaled LS solution $\alpha \, \underline{w}_{LS}$.
- As all scaled versions of w describe the same line, it does not matter.
- The distance between a and b, however, matters.

Classification: LS or Iterative Solution?

- Our first (simple) problem allowed a formulation in LS and thus a direct solution by the computation of a pseudo inverse.
- Assuming a decent numerical posedness of the Gramian and a not too large number of variables, the ordinary LS solution would be the method of choice.
- However, often problems are formulated in nonlinear terms or at least one of the above problems occurs (in practice most times all of them occur at the same time!),
- Then **iterative schemes** are preferred.

Iterative Schemes

- Such algorithms always work based on the following idea
  - Have a starting value (in case you have a priori knowledge, zero otherwise)
  - Improve the first estimate/approximation by a simple operation that includes an error term
  - Continue doing so until the error term becomes zero (or practically close to zero)

$$\underline{\hat{w}}_{k+1} = \underline{\hat{w}}_k + \mu \frac{\partial}{\partial \underline{w}} g(e_k^2)$$

- Advantages:
  - simple, robust, efficient in implementation
- Drawbacks:
  - may require many iterations,
  - may get stuck in a local minimum

Iterative Schemes

- Although each iteration step often only offers a small improvement step, this property becomes very handy once new data arrive.
- Thus, with every new data pair arriving, the algorithm keeps improving its quality

$$\underline{\hat{w}}_{k+1} = \underline{\hat{w}}_k + \mu \frac{\partial}{\partial \underline{w}} g(e_k^2)$$

- However, it also begins to forget older achievements and gives the newer data more weight
- By this the iterative algorithm can perform tracking.
- If the class evolves over time, the iterative algorithm is able to follow such evolution.

Iterative Schemes

- **Gradient based approaches** use a gradient, derived from a (quadratic) error:

$$\frac{\partial}{\partial e} g(e^2) = 2\, e\, g'(e^2)$$

$$\frac{\partial}{\partial e}\left(g(e)\right)^2 = 2\, g(e)\, g'(e^2)$$

$$with\ g(0) = 0$$

- **Reinforcement learning approaches** use error indicators to train the unknown weights.

Iterative Schemes

- If the data set is fixed and we iterate along such data set, we call the algorithm **iterative or off-line algorithm**.

- If on the other hand, there is permanently new data to learn from, we call the adaptive algorithm **recursive or on-line algorithm.** Such algorithms are usually more of interest as they can potentially allow also tracking, that is to change their behaviour in case the environment changes.