



university of
 groningen

Supervised Learning

Michael Biehl

Bernoulli Institute for Mathematics,
Computer Science and Artificial Intelligence
University of Groningen

www.cs.rug.nl/biehl

background: Statistical Physics, Computational Physics
Neural Networks, Stat. Phys. of Learning
Monte Carlo simulations of non-equilibrium processes

“recent”:
Machine Learning: theory, algorithm development
mostly in supervised learning, classification
application areas: biomedical data ...
astrophysics (very recently)



THIS IS YOUR MACHINE LEARNING SYSTEM?

YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.

WHAT IF THE ANSWERS ARE WRONG?

JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.



unfortunate, risky tendency:

- “blind” application of very complex tools (*SVM, DNN,...*)

Here, emphasis on:

- theoretical background
- mathematical foundations
- basic *classical* algorithms and methods
- how and why do they work?
- pitfalls and challenges
- tutorials: implementation, hands-on experiments



Start simple:

- e.g. K-NN classifier, linear regression, PCA, k-means
- compare to baseline algorithms
- increase level of sophistication if necessary / promising
- the latest trend is not necessarily the best for your problem

Accuracy is not enough:

- try to obtain insight
- employ interpretable models/systems, vsualization
- proper testing/validation with respect to suitable measures
- beware of artefacts, biased data sets ...

Introduction:

- supervised learning, classification, regression
- machine learning “vs.” statistical modeling
- **K-means and Vector Quantization**

Early algorithms (yet still important)

- linear threshold classifier, Rosenblatt’s *Perceptron*
- adaptive linear neuron, Widrow and Hoff’s *Adaline*

From Perceptron to Support Vector Machine

- large margin classification
- beyond linear separability

Distance-based systems

- ~~prototypes: K-means and Vector Quantization~~
- from K-Nearest_Neighbors to Learning Vector Quantization
- adaptive distance measures and relevance learning



classification / regression

frequent workflow:

example data **training** → model

linear regression

machine learning vs. statistical modelling

linear threshold classifiers

from the perceptron to support vector machines

prototypes and distance-based classifiers

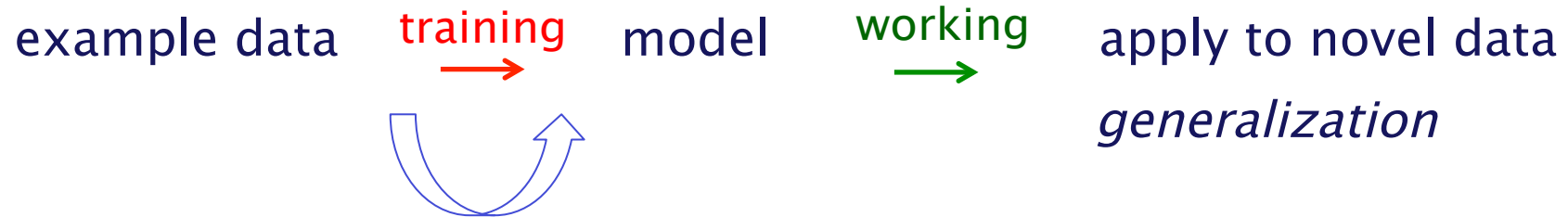
from K-means to Learning Vector Quantization

non-linear regression

(shallow) neural networks

classification / regression

frequent workflow:



validation

estimation of expected performance
avoid **over-fitting**, facilitate **model selection**
cope with **bias and variance dilemma**

input
(data)

'4'	'13'	'6'	'11'	'8'
A	B	A	B	A

output
(class label)

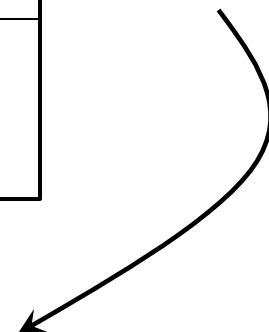
question

'7'
?

answer

consistent hypotheses:

odd → B

 ≤ 10 → A

input (data)	'4'	'13'	'6'	'11'	'8'	'2'
output (class label)	A	B	A	B	A	B






question	'7'
answer	?

consistent hypotheses:

~~odd → B~~
~~< 10 → A~~
 prime → B


space of consistent hypotheses shrinks with increasing # of examples

input
(data)

				
pear	apple	apple	pear	apple

output
(class label)







question


apple

answer


selected features: ~ color

input
(data)

					
pear	apple	apple	pear	apple	apple

output
(class label)

question


pear

answer

selected features: ~ shape

selection of possible features
for the representation of data

- is an important step in model design
- can be part of the training process

assign an observation (data) to one of C classes/categories
here, we consider exclusively numerical, 'vectorial' data

- character/digit/speech recognition
- medical diagnosis
- pixel-wise segmentation in image processing
- object recognition/scene analysis
- galaxy classification
-

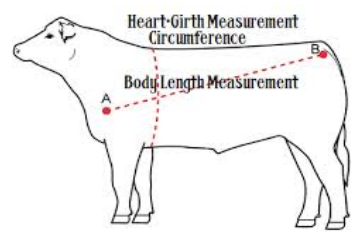
machine learning approach:

extract information from **example data**

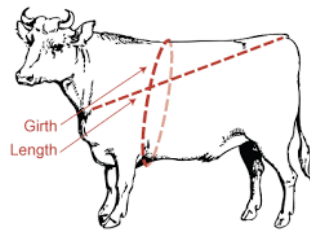
parameterized in a learning system (neural network, LVQ, SVM...)

working phase: application to novel data

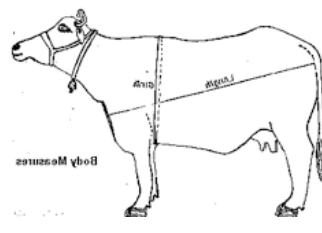
“input” (example data: images, measured quantities, numerical features)



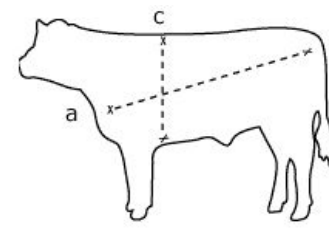
912 kg



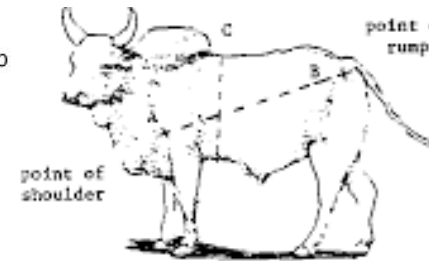
617 kg



489 kg



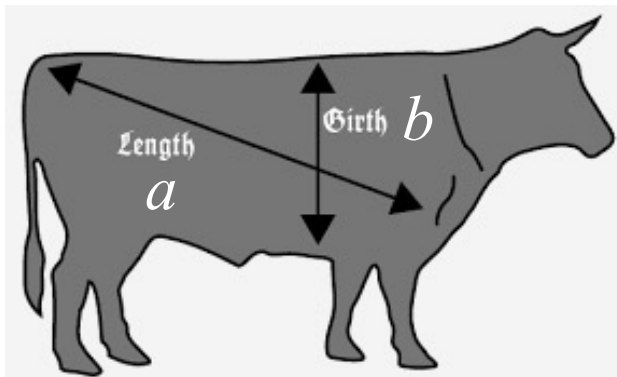
820 kg



591 kg

“output” (weight)

prediction of weight ?



a, b [inch], m [lb]

explicit mathematical/statistical model
e.g.

$$m = \gamma (a^\alpha \cdot b^\beta)$$

with adaptive parameters α, β, γ
inferred from example data (= learning)

$$\alpha = 1, \beta = 2, \gamma = 1/300$$

theory: **expanding universe**
velocity of far away galaxies
is proportional to their
distance:

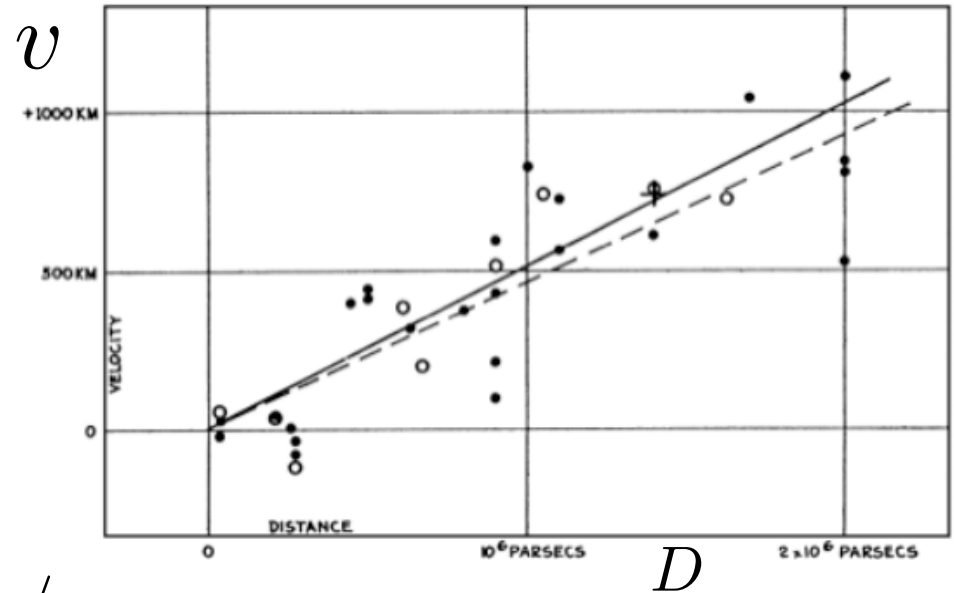
$$v = H_o D$$

Hubble constant

1929: $H_o = 500 \frac{km/s}{Mpc}$

2018: $H_o = 73.52(\pm 1.62) \frac{km/s}{Mpc}$

[Edwin Hubble, PNAS 15(3): 168–73 1929]



(more & better data)

more general problem:

N -dimensional arguments

$$\xi \in \mathbb{R}^N$$

multiple linear regression

real valued targets

$$y \in \mathbb{R}$$



“The short answer is that there is no difference”

“ML is simply statistics, the rest is marketing”

“All ML systems are heuristic black-boxes”

“Machine learning is the new statistics”

“Statistics is only for small data sets, ML is for big data”

“Statistical modeling has led to irrelevant theory and questionable conclusions”

“Whatever machine learning will look like in ten years, I’m sure statisticians will be whining that they did it earlier and better”



Machine Learning and Statistical Modeling

- *extract* information from data/observations
- formalize/parameterize the obtained information
- create and fit ‘models’
- often employ closely related methods from different perspectives

Statistical Modeling

emphasis: **inference**

- explain/understand data
in terms of **explicit models**
- verify assumptions/hypotheses,
significance/confidence
- can be used for *prediction*
e.g. in Bayesian classification

Machine Learning

emphasis: **prediction**

- discover **patterns** in the data
with no/few explicit assumptions
- achieve good performance
w.r.t. novel data
- may also aim at *understanding*
e.g. in relevance learning



given.: $\mathcal{D} = \{\boldsymbol{\xi}^\mu, y^\mu\}_{\mu=1}^P$ hypothesis: (approx.) linear relation

$$\boldsymbol{\xi}^\mu \in \mathbb{R}^N \quad y^\mu \in \mathbb{R} \quad f_H(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\xi} \quad \mathbf{w}, \boldsymbol{\xi} \in \mathbb{R}^N$$

remark: constant term formally via ‘clamped input component’, e.g.

$$\tilde{\boldsymbol{\xi}}^\mu = (\xi_1^\mu, \xi_2^\mu, \dots, \xi_N^\mu, -1)^\top \in \mathbb{R}^{N+1} \quad \Rightarrow \quad \tilde{\mathbf{w}}^\top \tilde{\boldsymbol{\xi}} = \mathbf{w}^\top \boldsymbol{\xi} - \theta$$

$$\tilde{\mathbf{w}} = (w_1, w_2, \dots, w_N, \theta)^\top \in \mathbb{R}^{N+1}$$

heuristics, direct treatment:

coefficients with minimal

quadratic deviation (Mean Square Error)

(no explicit model of the deviations)

$$E = \frac{1}{2} \sum_{\mu=1}^P \left[\mathbf{w}^\top \boldsymbol{\xi}^\mu - y^\mu \right]^2$$

$$E = \frac{1}{2} \sum_{\mu=1}^P \left[\mathbf{w}^\top \boldsymbol{\xi}^\mu - y^\mu \right]^2$$

$$\nabla_{\mathbf{w}} E = \sum_{\mu=1}^P \left[\mathbf{w}^\top \boldsymbol{\xi}^\mu - y^\mu \right] \boldsymbol{\xi}^\mu$$

$$Y = [y^1, y^2, \dots, y^P]^\top \in \mathbb{R}^P$$

$$= X^\top (X\mathbf{w} - Y) = 0 ?$$

$$X = [\boldsymbol{\xi}^1, \boldsymbol{\xi}^2, \dots, \boldsymbol{\xi}^P]^\top \quad (P \times N)$$

explicit, formal solution

$$\mathbf{w}^* = \underbrace{[X^\top X]^{-1}}_{N \times N} X^\top Y \quad N \times P$$

if $[X^\top X]^{-1}$ exists

(Moore–Penrose) pseudo-inverse
of rectangular matrix X

(necessary condition: $P \geq N$)

$$\mathbf{w}^* = [X^\top X]^{-1} X^\top Y \quad \text{if } [X^\top X]^{-1} \text{ exists}$$

modification for singular matrices $X^\top X$

(e.g. too few data points and/or dependencies/correlations)

$$\mathbf{w}^* = [X^\top X + \lambda I]^{-1} X^\top Y \quad N\text{-dimensional identity } I$$

$\lambda > 0$ guarantees existence of the inverse,
resulting \mathbf{w}^* corresponds to the minimum of

$$E_\lambda = \frac{1}{2} \sum_{\mu=1}^P \left[\mathbf{w}^\top \boldsymbol{\xi}^\mu - y^\mu \right]^2 + \frac{1}{2} \lambda \mathbf{w}^2 \quad \text{penalty for large norm of weights}$$

an example of *regularization* – a general concept in supervised ML
limiting the flexibility of an ML system can be beneficial

available set of data $\mathcal{ID} = \{\boldsymbol{\xi}^\mu, y^\mu\}_{\mu=1}^P$

model: explain observed data as (independently) generated by

$$p(y \mid \boldsymbol{\xi}, \mathbf{w}) = \mathcal{N}(y \mid \mathbf{w}^\top \boldsymbol{\xi}, \sigma^2) \propto \exp \left[-\frac{1}{2\sigma^2} (y - \mathbf{w}^\top \boldsymbol{\xi})^2 \right]$$

i.e. $y^\mu = \mathbf{w}^\top \boldsymbol{\xi}^\mu + \eta^\mu$ deviations η^μ according to $\mathcal{N}(0, \sigma^2)$

likelihood of the observed data, ...

$$p(\mathcal{ID} \mid \mathbf{w}) = \prod_{\mu=1}^P p(y^\mu \mid \boldsymbol{\xi}^\mu, \mathbf{w})$$

available set of data $\mathcal{ID} = \{\boldsymbol{\xi}^\mu, y^\mu\}_{\mu=1}^P$

model: explain observed data as (independently) generated by

$$p(y \mid \boldsymbol{\xi}, \mathbf{w}) = \mathcal{N}(y \mid \mathbf{w}^\top \boldsymbol{\xi}, \sigma^2) \propto \exp \left[-\frac{1}{2\sigma^2} (y - \mathbf{w}^\top \boldsymbol{\xi})^2 \right]$$

i.e. $y^\mu = \mathbf{w}^\top \boldsymbol{\xi}^\mu + \eta^\mu$ deviations η^μ according to $\mathcal{N}(0, \sigma^2)$

log-likelihood of the observed data ...

$$\log p(\mathcal{ID} \mid \mathbf{w}) = \sum_{\mu=1}^P \log p(y^\mu \mid \boldsymbol{\xi}^\mu, \mathbf{w}) = -\frac{P}{2} \log(2\pi\sigma^2) - \frac{1}{\sigma^2} \sum_{\mu=1}^P E(\mathbf{w})$$

... is maximized when E is minimized w.r.t. the weight vector

Maximum Likelihood estimate corresponds to MSE solution!



prior probability of weights, e.g. $p_o(\mathbf{w}) \propto \exp \left[-\frac{1}{2\tau^2} \mathbf{w}^2 \right]$

expresses *belief/knowledge* that large $|\mathbf{w}|$ are less likely

posterior probability (given data): $p(\mathbf{w}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{w}) p_o(\mathbf{w})$

$$\log [p(\mathbf{w}|\mathcal{D})] \sim -E(\mathbf{w}) - \frac{1}{2} \lambda \mathbf{w}^2 + \text{const.}$$

with appropriate λ

$$\mathbf{w}^* = [X^\top X + \lambda I]^{-1} X^\top Y$$

maximum a posteriori (MAP) solution

corresponds to **weight decay** (Tikhonov-, L2-regularization,)

instead of one specific solution (*Max. Likelihood, MAP*), consider all:

$$p(y|\boldsymbol{\xi}, \mathcal{ID}) = \int p(y|\boldsymbol{\xi}, \mathbf{w}, \sigma^2) \underbrace{p(\mathbf{w}, \sigma^2 | \mathcal{ID})}_{p(\mathcal{ID}|\mathbf{w}, \sigma^2) p(\mathbf{w}, \sigma^2)} d^N w d[\sigma^2]$$

*probabilistic
response*

model

*data
likelihood*

prior

*all possible
settings*

frequent practices:

- convenient choices (e.g. prior) in order to achieve simple schemes
- restrict optimization to subset of parameters, e.g. \mathbf{w} for fixed σ
- approximations often yield (heuristic) machine learning methods
- incremental inference schemes for sequence of observations



Vector Quantization and density estimation

- was planned as introduction to ‘prototype-based learning’
- links to Dalya Baron’s lectures on unsupervised learning
- provides another example for the relation of ‘heuristic’ machine learning and statistical modelling

Possible aims of unsupervised learning:

- represent a large data set by a few *prototypes*
- identify structures (e.g. clusters) in a given data set
- estimate an underlying probability density of data



Vector Quantization: identify (few) typical representatives of data
which capture essential properties

VQ system: set of **prototypes**

$$\mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^K \quad \mathbf{w}^k \in \mathbb{R}^N$$

data: set of **feature vectors**

$$\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^P \quad \mathbf{x}^\mu \in \mathbb{R}^N$$

(sorry, slight change of notation...)

assignment to prototypes:

based on dis-similarity/distance measure $d(\mathbf{w}, \mathbf{x}) \geq 0$

given vector \mathbf{x}^μ , determine *winner* $\mathbf{w}^* = \operatorname{argmin}_j \{d(\mathbf{w}^j, \mathbf{x}^\mu)\}$

→ assign \mathbf{x}^μ to prototype \mathbf{w}^*

one popular example: (squared) **Euclidean distance**

$$d(\mathbf{w}, \mathbf{x}) = \sum_{n=1}^N (w_n - x_n)^2$$

quantization error (sum over all data points)

measures the quality of the representation

defines a (one) criterion to evaluate / compare
the quality of different prototype configurations

$$H_{VQ} = \sum_{j=1}^K \sum_{\mu=1}^P \underbrace{(\mathbf{x}^\mu - \mathbf{w}^j)^2}_{d_j^\mu} \underbrace{\prod_{k \neq j} \Theta(d_k^\mu - d_j^\mu)}_{\mathbf{w}^j \text{ is the winner!}}$$

here:

Euclidean distance

$$\Theta(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{else} \end{cases}$$

- assign each data point to closest prototype
- measure the corresponding (squared) distance

optimization of quantization error:

- K-means algorithm
- unsupervised competitive learning



There is nothing objective about objective functions

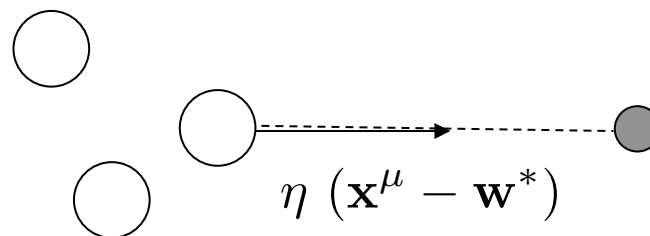
Jay McClelland

initially: *randomized* \mathbf{w}^k , random sequence of single data:

... *the winner takes it all*: $\mathbf{w}^* \rightarrow \mathbf{w}^* + \eta (\mathbf{x}^\mu - \mathbf{w}^*)$

competition for updates

learning rate / step size $\eta < 1$



η (<1): learning rate, step size

update along the negative gradient of H_{VQ} (contribution of a single data point)

repeated presentation of available data:

- sequential presentation of single data points
- e.g. random selection from the given set (with replacement)
- sweeps through data sets (“epochs”)

(0) initialization, for instance:

place vectors $w_k(t=0)$ at randomly selected data points

(1) assignment of each data point to nearest prototype/center

$$k^\mu \text{ with } \left(\mathbf{w}_{k^\mu}(t) - \mathbf{x}^\mu \right)^2 \leq \left(\mathbf{w}_j(t) - \mathbf{x}^\mu \right)^2 \text{ for all } j$$

(2) re-compute the centers as means over the assigned data:

$$\mathbf{w}_j(t+1) = \frac{\sum_{\mu=1}^P \delta_{j,k^\mu} \mathbf{x}^\mu}{\sum_{\mu=1}^P \delta_{j,k^\mu}}$$



comparison:

K-means: updates all prototypes, considers all data at a time
(batch- or offline-optimization)

VQ-*alg.* : updates only the winner, random sequential presentation of single examples (stochastic gradient descent)

both find (local) minimum of the quantization error

Gaussian mixture model:

- explain observed data as a set of independently generated vectors, drawn from a superposition of Gaussian densities:

$$P(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^K p_k (2\pi\sigma_k^2)^{-N/2} \exp \left[-\frac{1}{2\sigma_k^2} (\mathbf{x} - \mathbf{w}_k)^2 \right]$$

parameters: $\boldsymbol{\theta} = \{ \mathbf{w}_k, \sigma_k, p_k \}_{k=1}^K$

- centers of Gaussian contributions
- width of the Gaussians (here: isotropic)
- weight of Gaussians in the mixture

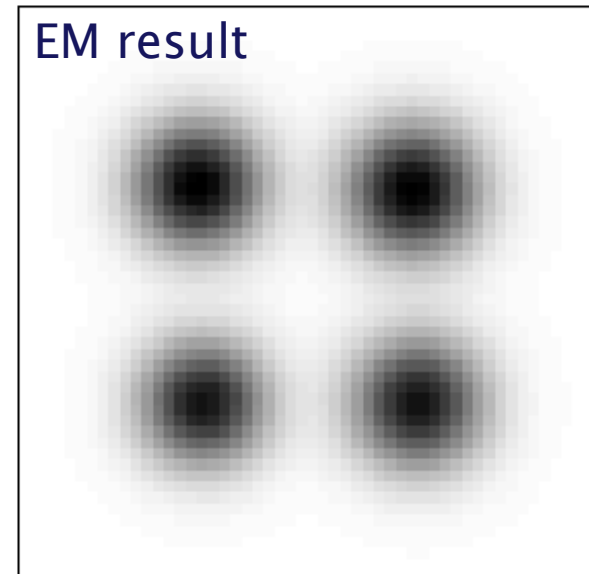
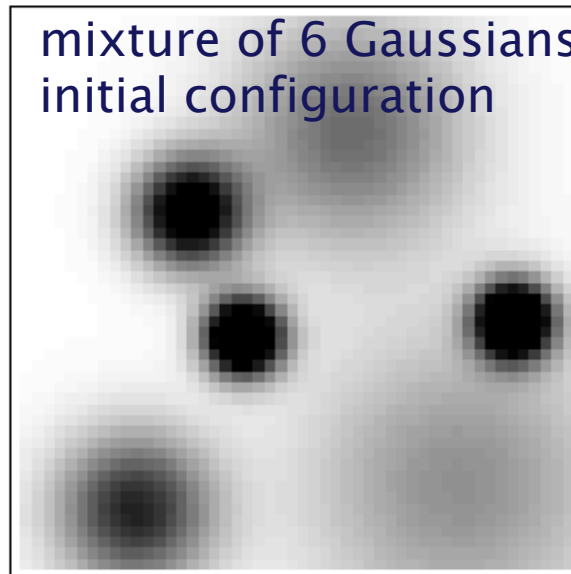
Maximum Likelihood approach to determine single, best model fit

- maximize (log-) Likelihood of observed data w.r.t. θ

$$L(\theta) = \prod_{\mu=1}^P P(\mathbf{x}^\mu | \theta) = \prod_{\mu=1}^P \left(\sum_k p_k (2\pi)^{-N/2} \sigma_k^{-N} \exp \left[-\frac{1}{2\sigma_k^2} (\mathbf{x}^\mu - \mathbf{w}_k)^2 \right] \right)$$

$$l(\theta) = \log L(\theta) = \sum_{\mu=1}^P \ln \left(\sum_k p_k \sigma_k^{-N} \exp \left[-\frac{1}{2\sigma_k^2} (\mathbf{x}^\mu - \mathbf{w}_k)^2 \right] \right) + \text{const.}$$

- Expectation–Maximization (EM) scheme [Dempster et al. 1977]



$\sigma_k = \sigma$ EM-Iteration:

- probabilistic assignment of data points to current centers:

$$Q_k^\mu(t) = \frac{p_k(t) \exp \left[-\frac{1}{2\sigma^2} (\mathbf{x}^\mu - \mathbf{w}_k(t))^2 \right]}{\sum_l p_l(t) \exp \left[-\frac{1}{2\sigma^2} (\mathbf{x}^\mu - \mathbf{w}_l(t))^2 \right]}$$

- re-compute centers (weighted means)

$$\mathbf{w}_k(t+1) = \sum_{\mu=1}^P \left(\frac{Q_k^\mu(t)}{\sum_{\nu=1}^P Q_k^\nu(t)} \right) \mathbf{x}^\mu$$

total weight of k-th Gaussian $p_k(t+1) = \frac{1}{P} \sum_{\mu=1}^P Q_k^\mu(t)$

limiting case: deterministic assignment to closest center

$$\lim_{\sigma \rightarrow 0} Q_k^\mu = \begin{cases} 1 & \text{if } (\mathbf{x}^\mu - \mathbf{w}_k)^2 \leq (\mathbf{x}^\mu - \mathbf{w}_j)^2 \text{ for all } j \\ 0 & \text{else} \end{cases}$$

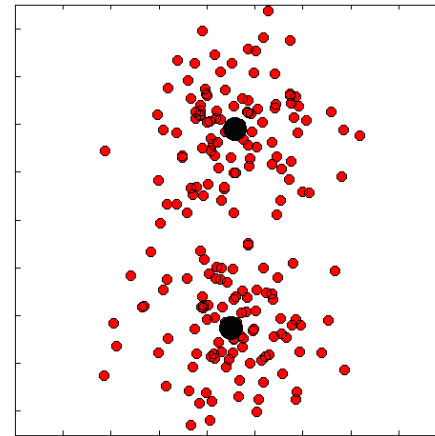
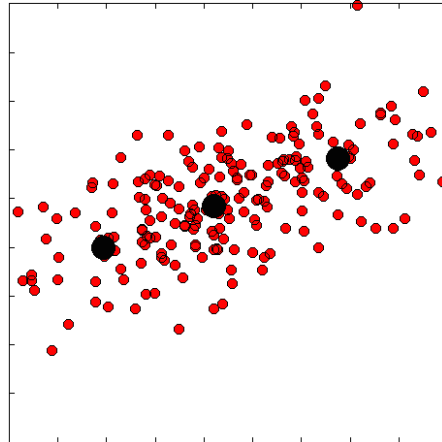
K-means algorithm recovered!

Remark 1: VQ \neq clustering

(“K-means clustering” ?)

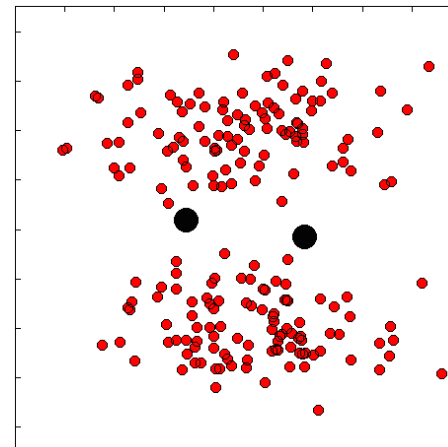
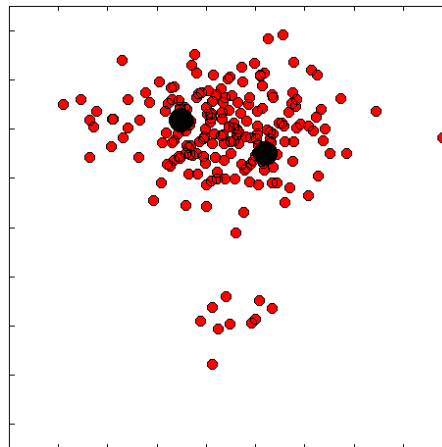
minimal quantization error (Euclidean distance)

in general:
representation
of observations
in feature space



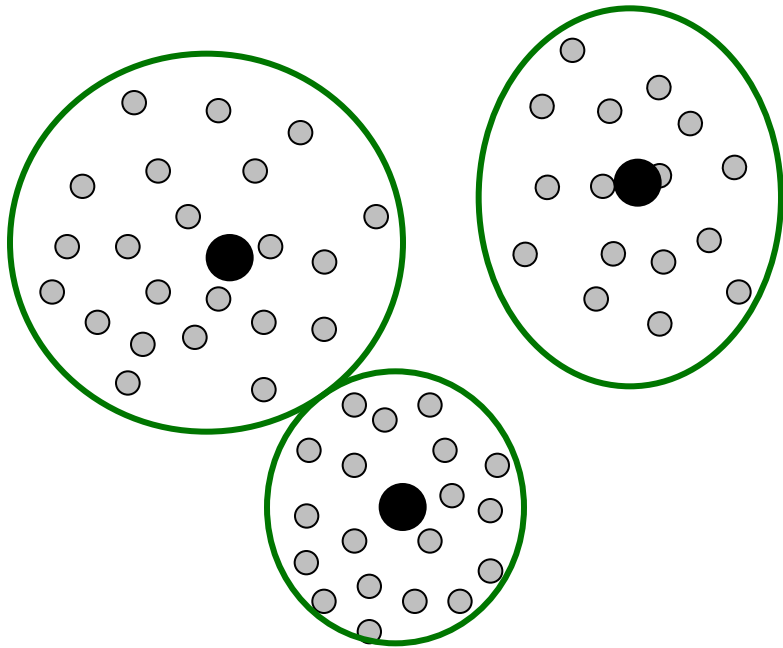
ideal clustering
scenario:
well-separated,
‘spherical’ clusters

small clusters
irrelevant with
respect to quan-
tization error



sensitive to
cluster shape,
coordinate
transformations
(even linear)

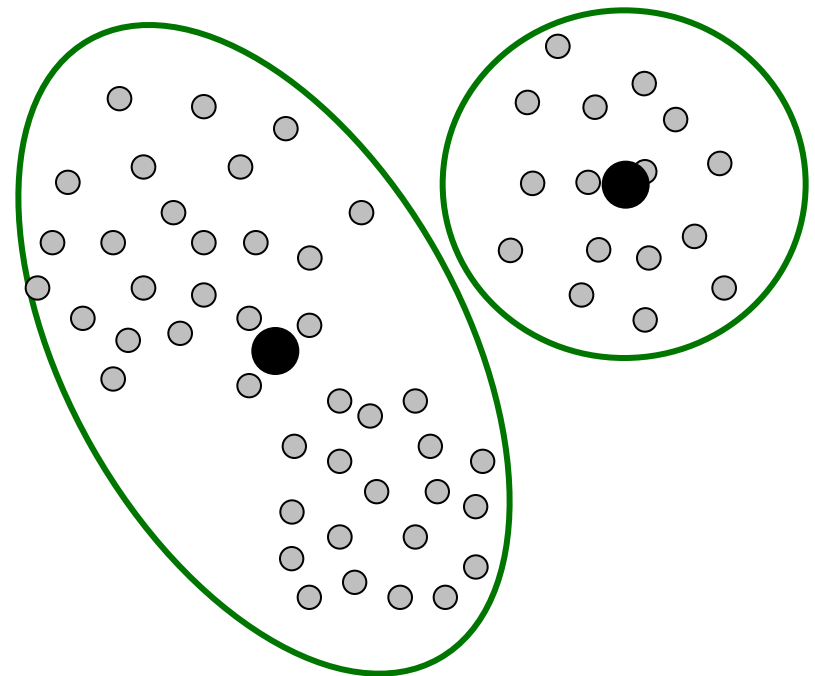
Remark 2: clustering is an ill-defined problem



“obviously three clusters”

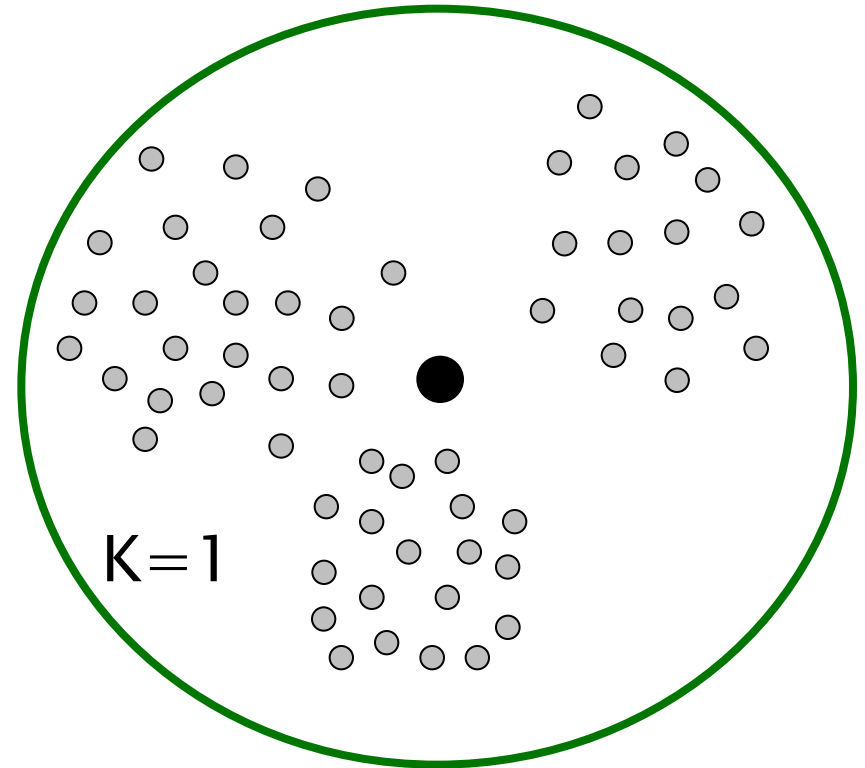
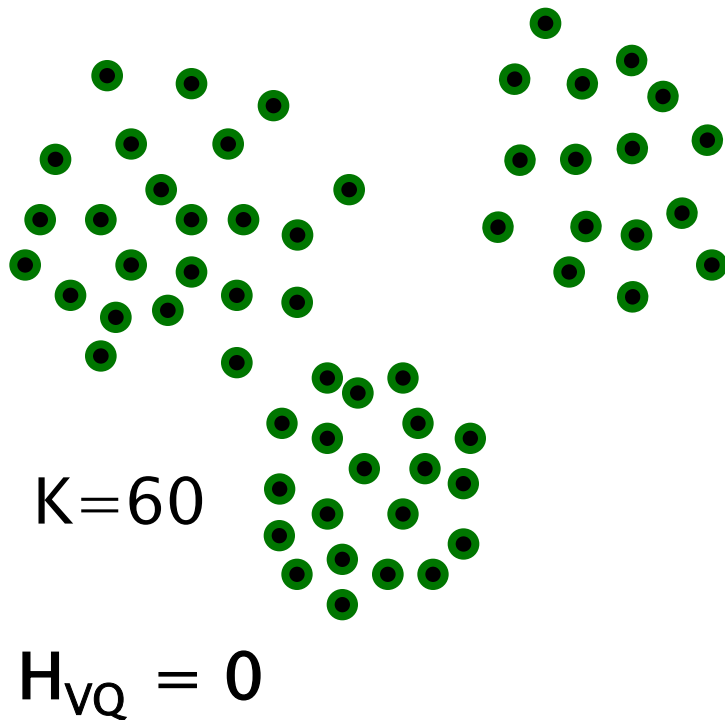
our criterion: lower H_{VQ}

→ “better clustering” ???



“well, maybe only two?”

higher H_{VQ}



→ “ the best clustering ” ?

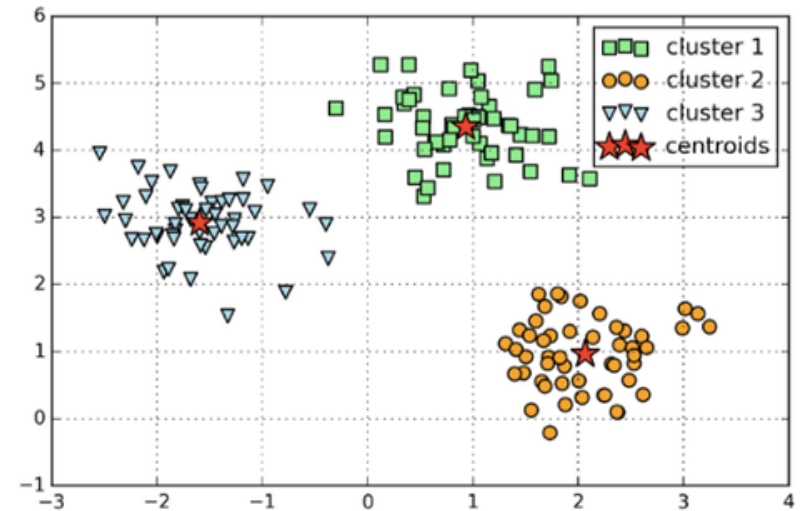
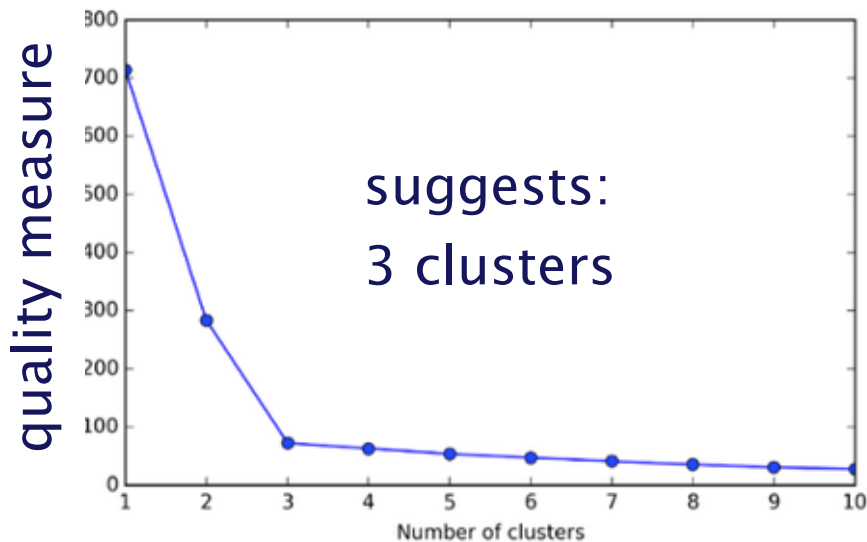
the simplest clustering ...

H_{VQ} (and similar criteria) allow only to compare VQ with the same K !
more general: heuristic compromise between “error” and “simplicity”

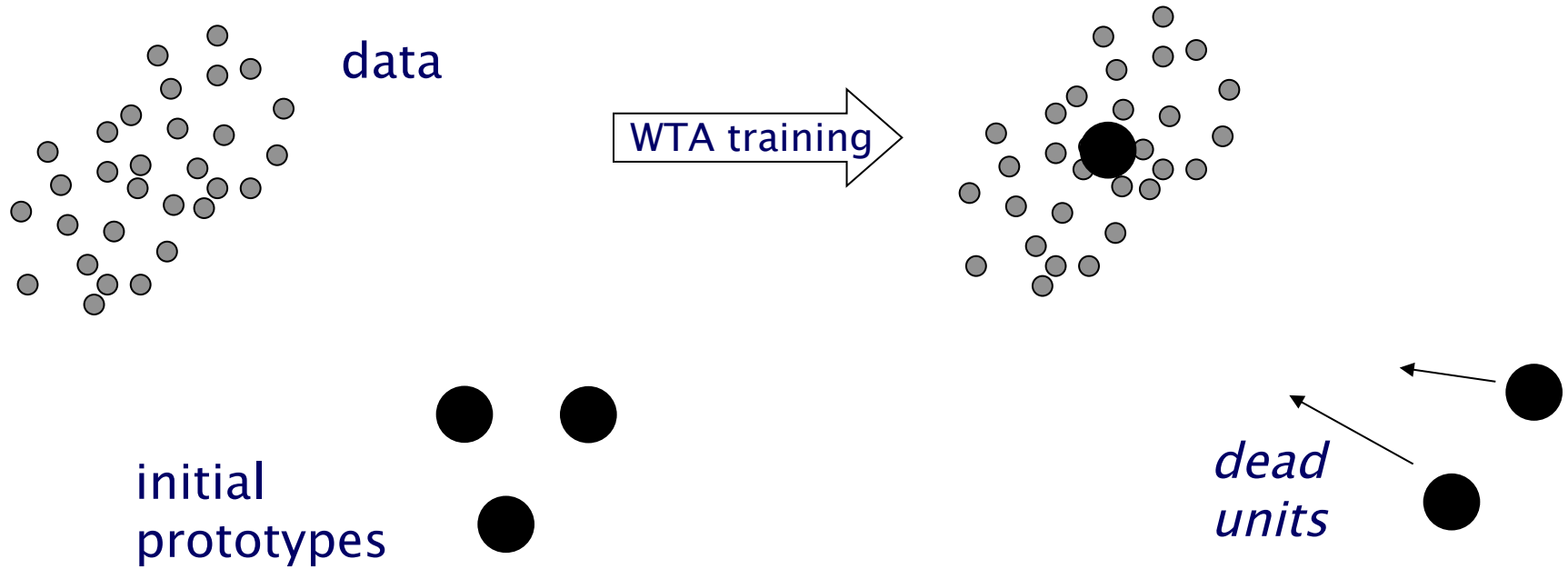
popular heuristics: elbow method

- run VQ for different K to convergence
- determine quantization error as a function of K
- identify “elbow” or other characteristic value of K

suggestive example: (in practice usually less pronounced)



practical issues of VQ training:



possible solution: rank-based updates (winner, second, third, ...)

in general: local minima of the quantization error,
initialization-dependent outcome of training

favorable initialization: *k-means++* [Arthur & Vassilvitskii, 2007]

many prototypes (*gas*) to represent the density of observed data

[Martinetz, Berkovich, Schulten, IEEE Trans. Neural Netw. 1993]

introduce rank-based *neighborhood cooperativeness*:

upon presentation of \mathbf{x}^μ :

- determine the rank (w.r.t. distance) of the prototypes

$$k_j \left(\mathbf{x}^\mu, \{\mathbf{w}_k\}_{k=1}^K \right) = \sum_{l=1}^K \Theta [d(\mathbf{w}_j, \mathbf{x}^\mu) - d(\mathbf{w}_l, \mathbf{x}^\mu)]$$

- update all prototypes: $\mathbf{w}_j \rightarrow \mathbf{w}_j + \eta h_\lambda(k_j) (\mathbf{x}^\mu - \mathbf{w}_j)$

with neighborhood function $h_\lambda(k_j) = \exp(-k_j/\lambda)$

and range λ in terms of rank (independent of overall scale)

- potential *annealing* of λ from large to smaller values with time



Machine Learning and Statistical Modeling

are **different** / complementary as they ...

- start from different formal perspectives
- emphasize different goals

are **similar** / closely related as they ...

- often yield very similar / identical methods
- ML methods limits or approximations of stat. mod.
- frequently can be used interchangeably

- it is useful to know and take advantage of both worlds
- don't be religious about making choices



Springer Series in Statistics

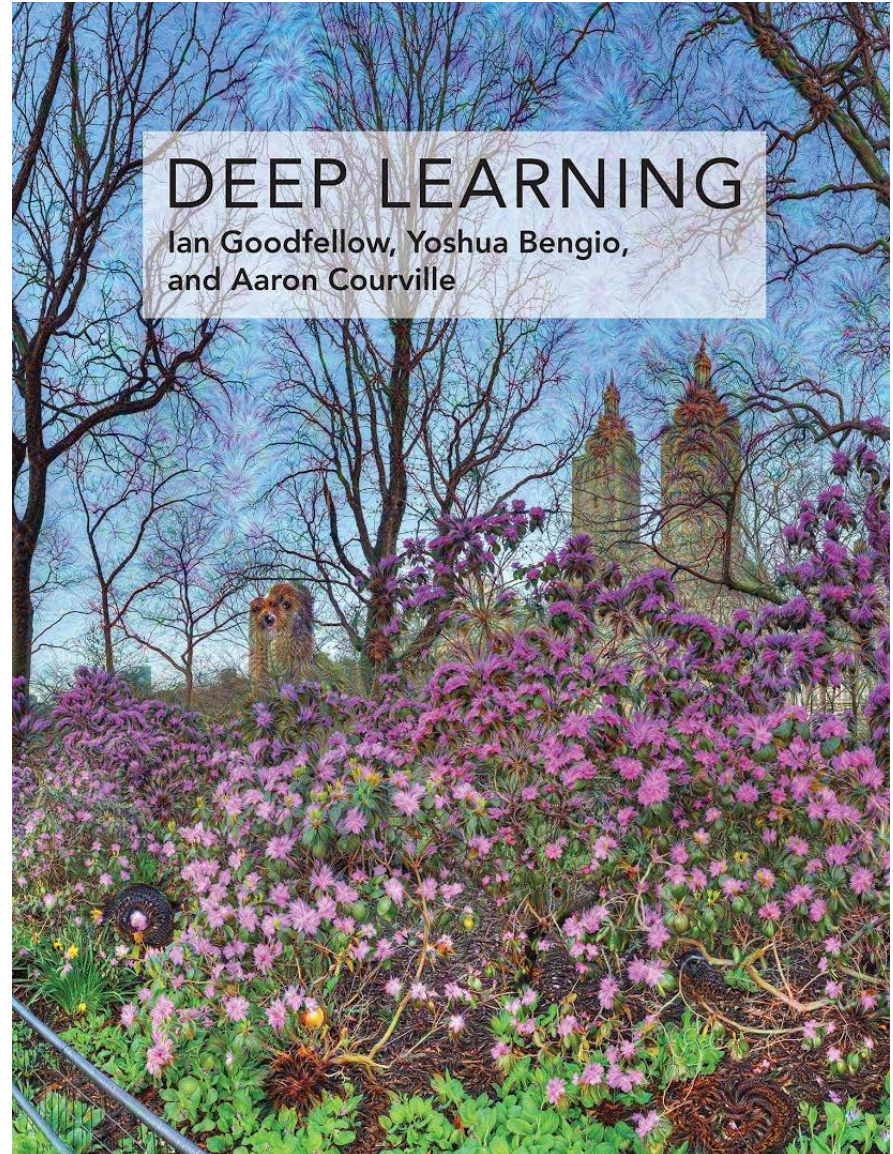
Trevor Hastie
 Robert Tibshirani
 Jerome Friedman

The Elements of Statistical Learning

Data Mining, Inference, and Prediction

Second Edition

 Springer



Introduction:

- supervised learning, classification, regression
- machine learning “vs.” statistical modeling

Early (important!) approaches:

- linear threshold classifier, Rosenblatt’s *Perceptron*
- adaptive linear neuron, Widrow and Hoff’s *Adaline*

From Perceptron to Support Vector Machine

- large margin classification
- beyond linear separability

Distance-based systems

- prototypes: K-means and Vector Quantization
- from K-Neares_Neighbors to Learning Vector Quantization
- adaptive distance measures and relevance learning



The New York Times:

“The embryo of an electronic computer that ... will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.”

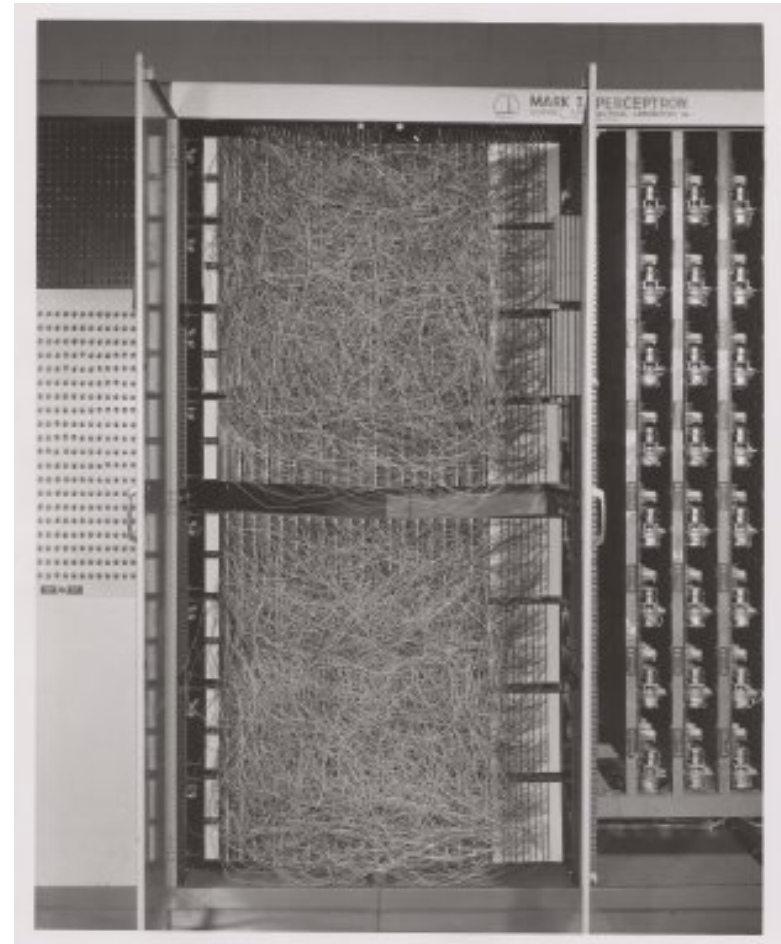
2018 ?

The New York Times:

“The embryo of an electronic computer that ... will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.”

The perceptron (1957)

Frank Rosenblatt
Cornell Aeronautical Laboratory
& Office of Naval Research



Mark 1 perceptron:

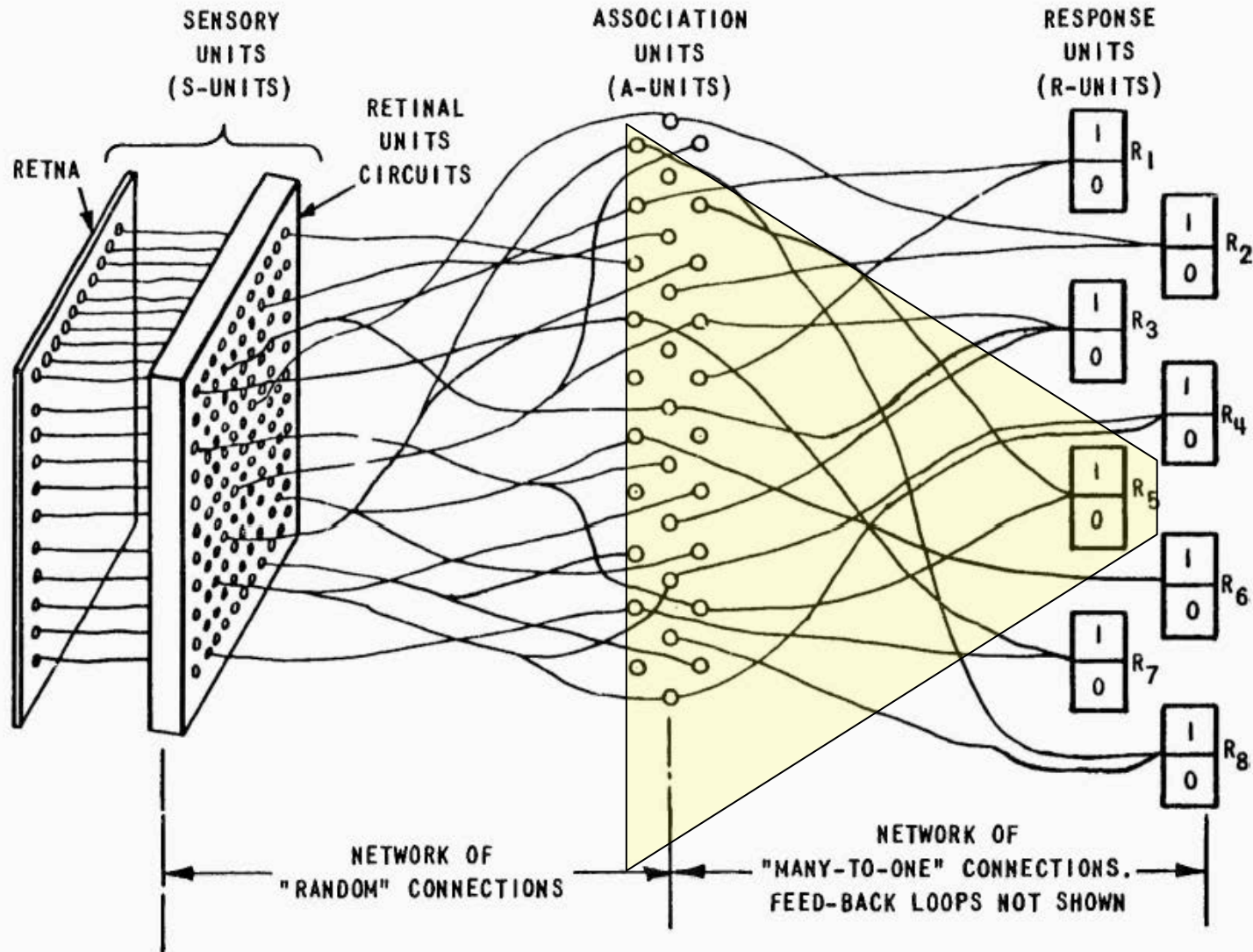
sensors: 400 photocells

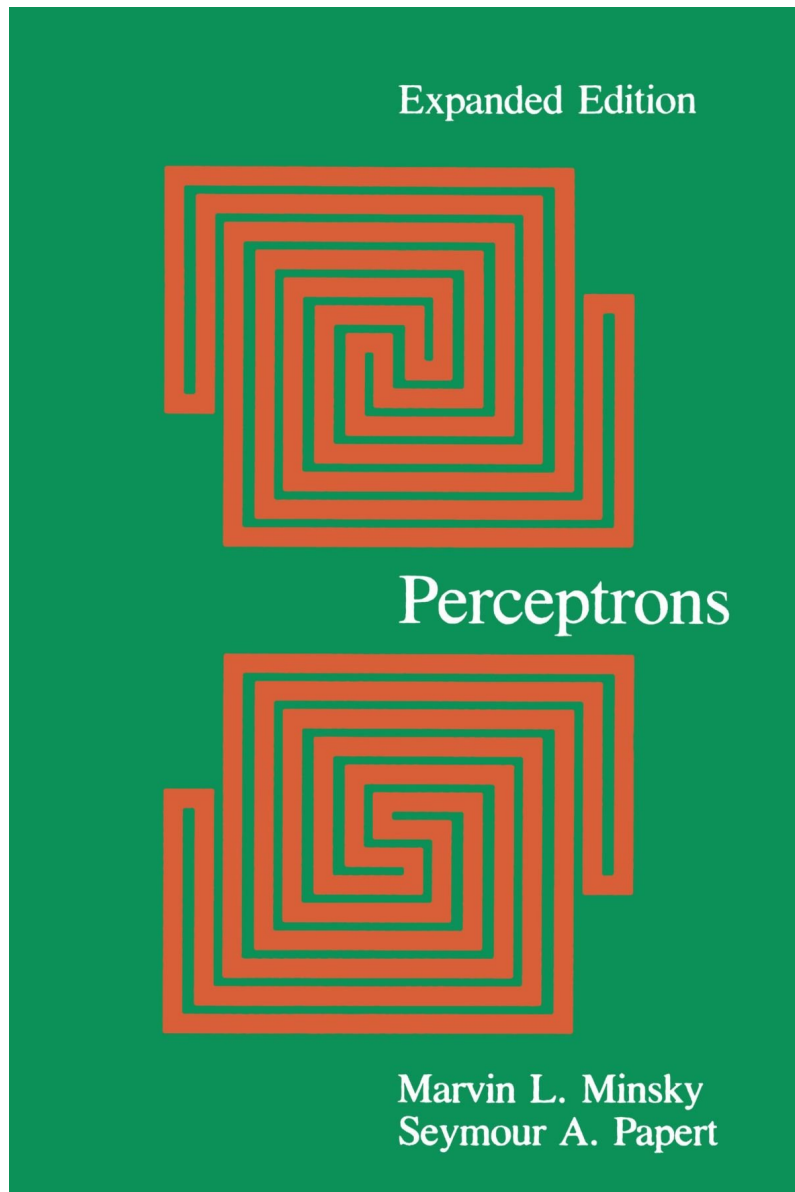
weights: potentiometers

changed by electric motors

youtube video on the Perceptron (historical document 😊)

http://www.youtube.com/watch?v=cNxadbrN_al





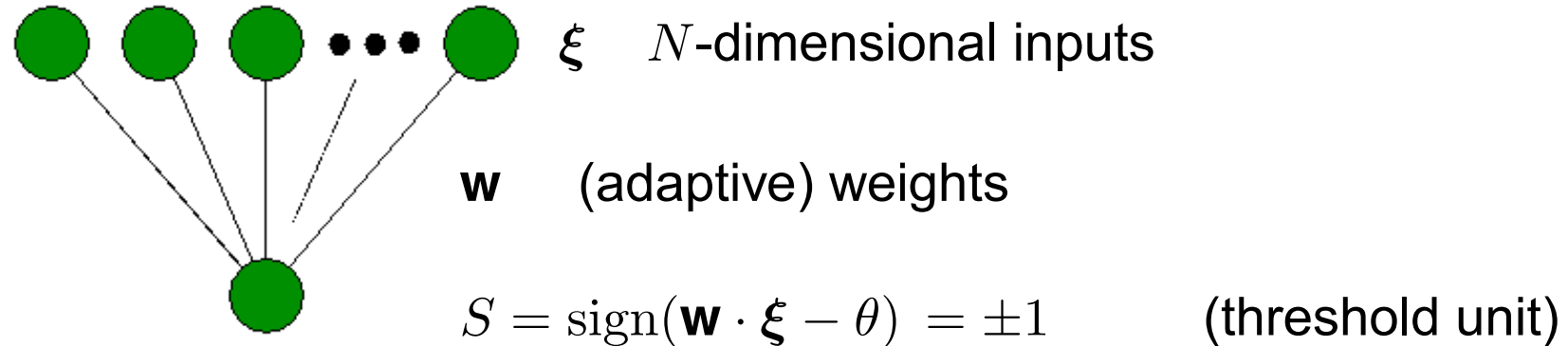
Perceptrons

An introduction to computational geometry (1969)

- an excellent mathematical analysis of the perceptron
- pointed out limitations and restrictions
- essentially stalled the field of machine learning (perceived as only negative)

The Perceptron architecture

building block and simple feed-forward "network":



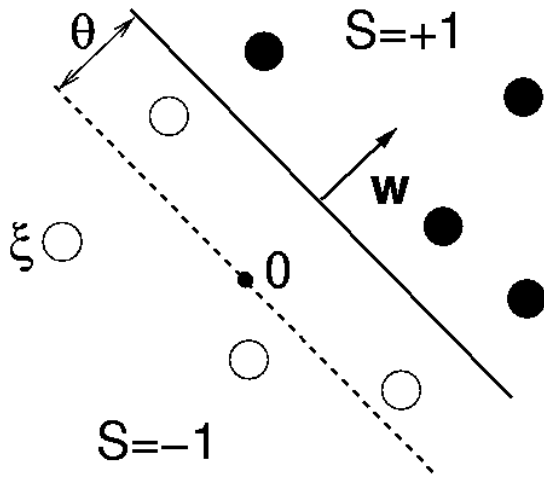
Perceptron Convergence Theorem: Rosenblatt (1958)

Capacity: Winder (1963), Cover (1965), Schläfli (**1852**)

Perceptrons, Minsky and Papert (1969)

Statistical physics theory of perceptron weights: Gardner (1988)

Support Vector Machines: Vapnik (1995)



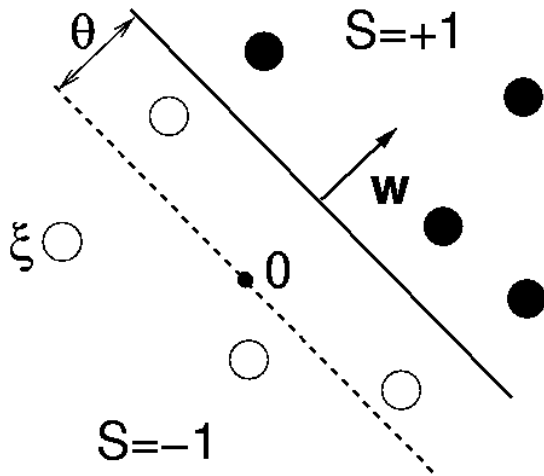
geometrical interpretation:

\mathbf{w} defines a **hyperplane** in \mathbb{R}^N

θ is the off-set from the origin (e.g. $|\mathbf{w}| = 1$)

linearly separable (l.s.) classification

of all possible inputs



geometrical interpretation:

w defines a **hyperplane** in \mathbb{R}^N

θ is the off-set from the origin (e.g. $|\mathbf{w}| = 1$)

linearly separable (l.s.) classification

of all possible inputs

A function $S(\xi)$ is called

- **homogeneously l.s.** if a vector $\mathbf{w} \in \mathbb{R}^N$ exists with $\text{sign}(\mathbf{w} \cdot \xi) = S(\xi)$ for all ξ
- **inhomogeneously l.s.** if a pair $\mathbf{w} \in \mathbb{R}^N, \theta \in \mathbb{R}$ exists with $\text{sign}(\mathbf{w} \cdot \xi - \theta) = S(\xi)$

formally:

$$\tilde{\xi} = (\xi_1, \xi_2, \dots, \xi_N, \xi_{N+1} = -1)^T \in \mathbb{R}^{N+1} \quad (\text{clamped extra input dimension})$$

$$\tilde{\mathbf{w}} = (w_1, w_2, \dots, w_N, w_{N+1} = \theta)^T \in \mathbb{R}^{N+1} \quad (\text{extra weight})$$

$$\tilde{\mathbf{w}} \cdot \tilde{\xi} = \mathbf{w} \cdot \xi - \theta \quad \rightarrow \quad S(\xi) \text{ inhom. l.s. in } \mathbb{R}^N \Leftrightarrow S(\tilde{\xi}) \text{ hom. l.s. in } \mathbb{R}^{N+1}$$

The perceptron (storage) problem

given a **dichotomy** of data $D_N^P = \{ \boldsymbol{\xi}^\mu \in \mathbb{R}^N, S^\mu \in \{-1, +1\} \}_{\mu=1,2,\dots,P}$

find a vector $\mathbf{w} \in \mathbb{R}^N$, such that $\text{sign}(\mathbf{w} \cdot \boldsymbol{\xi}^\mu) = S^\mu$ for all μ .

Questions:

- When is a given dichotomy linearly separable (l.s.)?
- How many l.s. D_N^P exist? (*The capacity of a hyperplane*)
- If it exists, how can we find a perceptron vector \mathbf{w} ?
- If there are several/many solutions, which is *best*?
- What can we do for non-separable D_N^P ?

Solving the perceptron storage problem

re-write the problem ...

consider a given data set $ID = \{\xi^\mu, S_R^\mu\}$

... find a vector \mathbf{w} with $S_H^\mu = \text{sign}(\mathbf{w} \cdot \xi^\mu) = S_R^\mu$ for all μ

Note: $\text{sign}(\mathbf{w} \cdot \xi^\mu) = S_R^\mu \Leftrightarrow \text{sign}(\mathbf{w} \cdot \xi^\mu S_R^\mu) = 1 \Leftrightarrow E^\mu = \mathbf{w} \cdot \xi^\mu S_R^\mu > 0$
(local potentials E^μ)

Solving the perceptron storage problem

re-write the problem ...

consider a given data set $ID = \{\xi^\mu, S_R^\mu\}$

... find a vector \mathbf{w} with $S_H^\mu = \text{sign}(\mathbf{w} \cdot \xi^\mu) = S_R^\mu$ for all μ

Note: $\text{sign}(\mathbf{w} \cdot \xi^\mu) = S_R^\mu \Leftrightarrow \text{sign}(\mathbf{w} \cdot \xi^\mu S_R^\mu) = 1 \Leftrightarrow E^\mu = \mathbf{w} \cdot \xi^\mu S_R^\mu > 0$
(local potentials E^μ)

equivalent problem: solve a set of **linear inequalities** (in \mathbf{w})

... find a vector \mathbf{w} with $E^\mu = \mathbf{w} \cdot \xi^\mu S_R^\mu \geq c > 0$ for all μ

Note that the actual value of $c > 0$ is irrelevant:

$$\left(\mathbf{w}_1 \text{ satisfies } \{E_1^\mu \geq c\}_{\mu=1}^P \right) \Leftrightarrow \left(\mathbf{w}_2 = \lambda \mathbf{w}_1 \text{ satisfies } \{E_2^\mu \geq \lambda c\}_{\mu=1}^P \right) \quad (\lambda > 0)$$

consider **iterative algorithms**

- **sequential presentation of data** $\{ \xi^{\mu(t)}, S_R^{\mu(t)} \}$, e.g.

learning steps (*time*) $t = 0, 1, 2, 3, \dots$

number of example $\nu(t) = 1, 2, 3, \dots, P, 1, 2, 3, \dots$

- **update of vectors** ($\mathbf{w}(0) = 0$) $E^{\nu(t)} = \mathbf{w}(t) \cdot \xi^{\nu(t)} S_R^{\nu(t)}$

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \frac{1}{N} f(E^{\nu(t)}) \xi^{\nu(t)} S_R^{\nu(t)} \quad (f \text{ defines the actual algorithm})$$

\mathbf{w} accumulates *Hebbian terms* $\xi^{\mu} S_R^{\mu}$ (*input* \times *output*)

consider **iterative algorithms**

- **sequential presentation of data** $\{ \xi^{\mu(t)}, S_R^{\mu(t)} \}$, e.g.

learning steps (*time*) $t = 0, 1, 2, 3, \dots$

number of example $\nu(t) = 1, 2, 3, \dots, P, 1, 2, 3, \dots$

- **update of vectors** ($\mathbf{w}(0) = 0$) $E^{\nu(t)} = \mathbf{w}(t) \cdot \xi^{\nu(t)} S_R^{\nu(t)}$

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \frac{1}{N} f(E^{\nu(t)}) \xi^{\nu(t)} S_R^{\nu(t)} \quad (f \text{ defines the actual algorithm})$$

\mathbf{w} accumulates *Hebbian terms* $\xi^{\mu} S_R^{\mu}$ (*input* \times *output*)

- \rightarrow **general form of the result** $\mathbf{w}(t) = \frac{1}{N} \sum_{\mu=1}^P x^{\mu}(t) \xi^{\mu} S_R^{\mu}$

x^{μ} is called the **embedding strength** of example μ

Rosenblatt's perceptron algorithm (Rosenblatt, 1958)

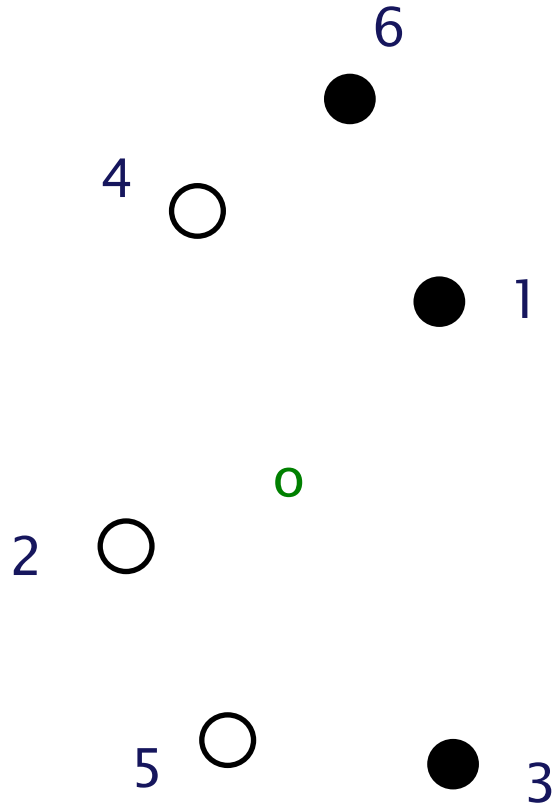
$$\mathbf{w}(t+1) = \mathbf{w}(t) + \frac{1}{N} \Theta [c - E^{\nu(t)}] \boldsymbol{\xi}^{\nu(t)} S_R^{\nu(t)} \quad (\text{initial weights } w(0) = 0)$$

learning from mistakes $f(E^\mu) = \Theta[c - E^\mu] = \begin{cases} 1 & \text{if } E^\mu < c \\ 0 & \text{if } E^\mu \geq c \end{cases}$

integer embedding strengths $0 \leq x^\mu(t+1) = \begin{cases} x^\mu(t) + 1 & \text{if } E^\mu < c \\ x^\mu(t) & \text{if } E^\mu \geq c \end{cases} \quad (\text{for } \mu = \nu(t))$

$$\mathbf{w}(0) = 0$$

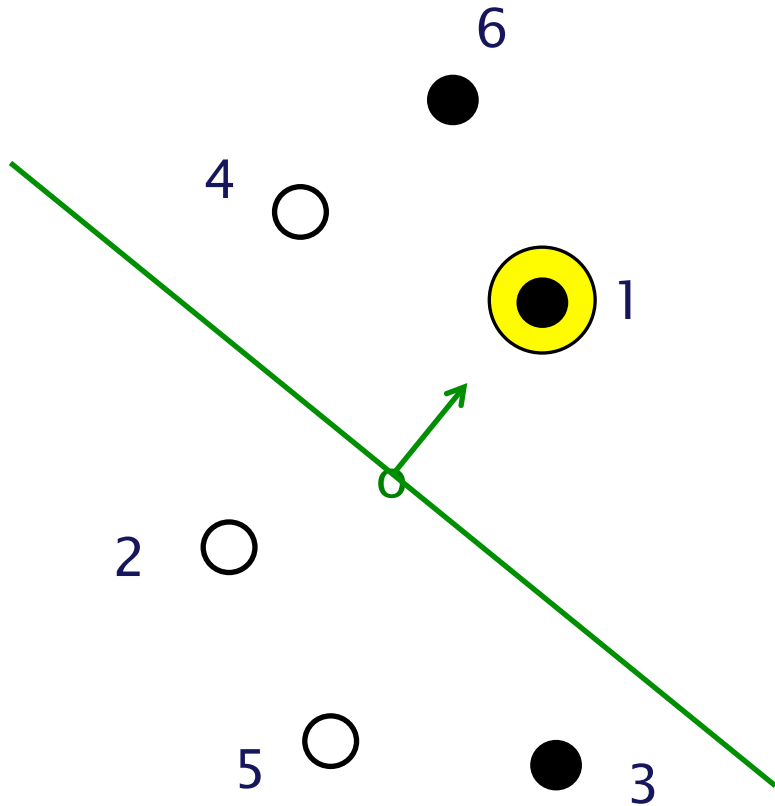
$$\mathbf{w}(t) = \frac{1}{N} \sum_{\mu=1}^P x^{\mu}(t) \boldsymbol{\xi}^{\mu} S_R^{\mu}$$



μ	x^{μ}
1	0
2	0
3	0
4	0
5	0
6	0

$$\mathbf{w}(1) = \frac{1}{N} \boldsymbol{\xi}^1 \cdot (+1)$$

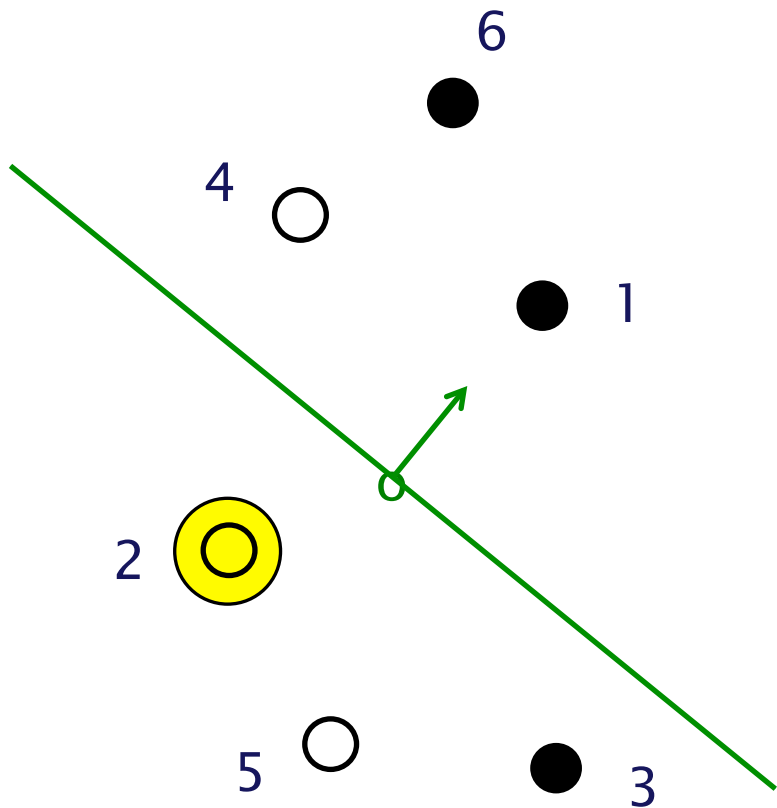
$$\mathbf{w}(t) = \frac{1}{N} \sum_{\mu=1}^P x^{\mu}(t) \boldsymbol{\xi}^{\mu} S_R^{\mu}$$



μ	x^{μ}
1	1
2	0
3	0
4	0
5	0
6	0

$$\mathbf{w}(2) = \mathbf{w}(1)$$

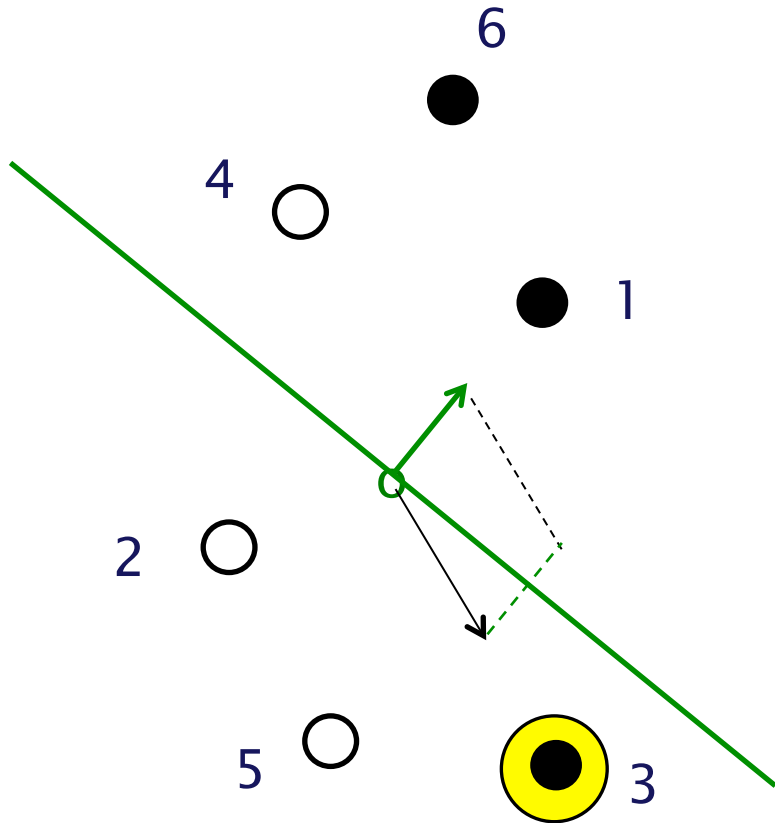
$$\mathbf{w}(t) = \frac{1}{N} \sum_{\mu=1}^P x^{\mu}(t) \boldsymbol{\xi}^{\mu} S_R^{\mu}$$



μ	x^{μ}
1	1
2	0
3	0
4	0
5	0
6	0

$$\mathbf{w}(3) = \mathbf{w}(2) + \frac{1}{N} \boldsymbol{\xi}^3 \cdot (+1)$$

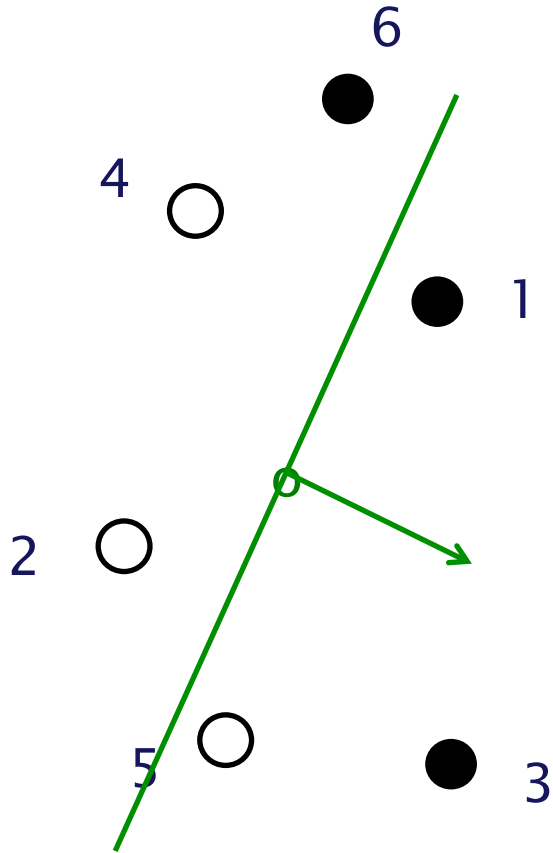
$$\mathbf{w}(t) = \frac{1}{N} \sum_{\mu=1}^P x^\mu(t) \boldsymbol{\xi}^\mu S_R^\mu$$



μ	x^μ
1	1
2	0
3	0
4	0
5	0
6	0

$$\mathbf{w}(3) = \mathbf{w}(2) + \frac{1}{N} \boldsymbol{\xi}^3 \cdot (+1)$$

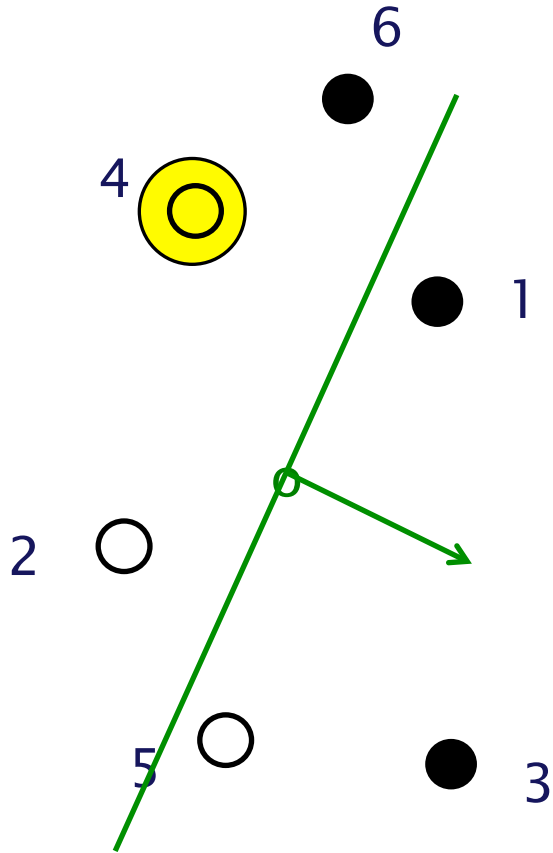
$$\mathbf{w}(t) = \frac{1}{N} \sum_{\mu=1}^P x^\mu(t) \boldsymbol{\xi}^\mu S_R^\mu$$



μ	x^μ
1	1
2	0
3	1
4	0
5	0
6	0

$$\mathbf{w}(4) = \mathbf{w}(3)$$

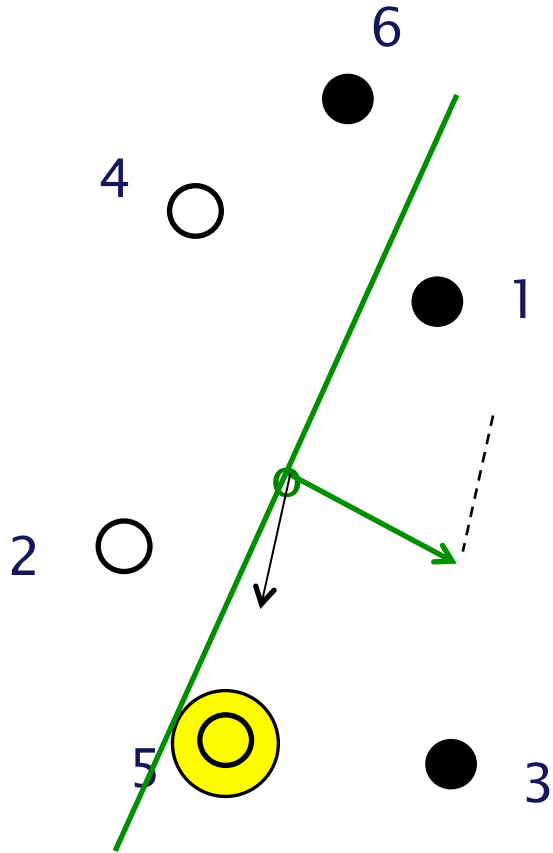
$$\mathbf{w}(t) = \frac{1}{N} \sum_{\mu=1}^P x^{\mu}(t) \boldsymbol{\xi}^{\mu} S_R^{\mu}$$



μ	x^{μ}
1	1
2	0
3	1
4	0
5	0
6	0

$$\mathbf{w}(5) = \mathbf{w}(4) + \frac{1}{N} \boldsymbol{\xi}^5 \cdot (-1)$$

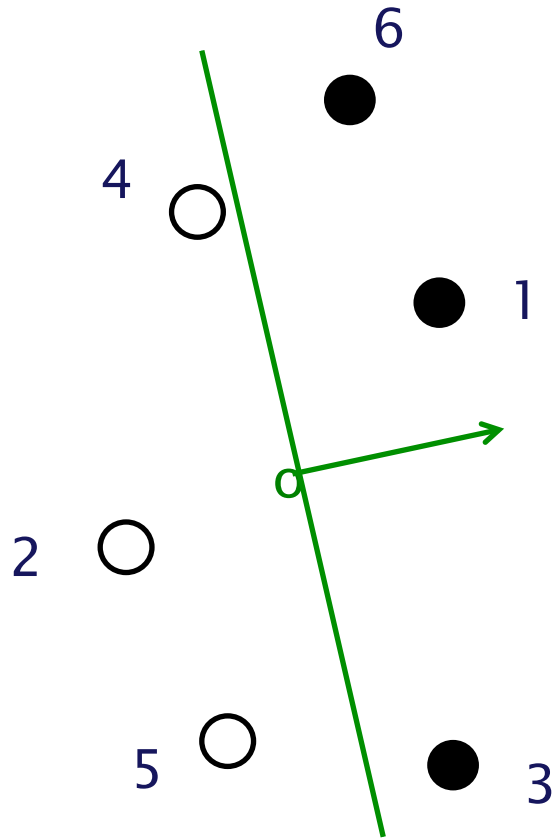
$$\mathbf{w}(t) = \frac{1}{N} \sum_{\mu=1}^P x^\mu(t) \boldsymbol{\xi}^\mu S_R^\mu$$



μ	x^μ
1	1
2	0
3	1
4	0
5	0
6	0

$$\mathbf{w}(5) = \mathbf{w}(4) + \frac{1}{N} \boldsymbol{\xi}^5 \cdot (-1)$$

$$\mathbf{w}(t) = \frac{1}{N} \sum_{\mu=1}^P x^\mu(t) \boldsymbol{\xi}^\mu S_R^\mu$$



μ	x^μ
1	1
2	0
3	1
4	0
5	1
6	0

solution found! (here: in the first epoch already)

Proof of convergence



- assume a vector \mathbf{w}^* exists with $\{E^{\mu*} = \mathbf{w}^* \cdot \boldsymbol{\xi}^\mu S_R^\mu \geq c > 0\}_{\mu=1}^P$
- note that for any vector \mathbf{w} : $0 \leq \frac{(\mathbf{w}^* \cdot \mathbf{w})^2}{|\mathbf{w}^*|^2 |\mathbf{w}|^2} = \cos^2 \angle(\mathbf{w}, \mathbf{w}^*) \leq 1$

Proof of convergence

- assume a vector \mathbf{w}^* exists with $\{E^{\mu*} = \mathbf{w}^* \cdot \boldsymbol{\xi}^\mu S_R^\mu \geq c > 0\}_{\mu=1}^P$
- note that for any vector \mathbf{w} : $0 \leq \frac{(\mathbf{w}^* \cdot \mathbf{w})^2}{|\mathbf{w}^*|^2 |\mathbf{w}|^2} = \cos^2 \angle(\mathbf{w}, \mathbf{w}^*) \leq 1$
- weight vector after t learning steps $\mathbf{w}(t) = \frac{1}{N} \sum_{\mu=1}^P x^\mu(t) \boldsymbol{\xi}^\mu S_R^\mu$ $\mathbf{w}(0)=0$

$x^\mu(t)$ number of non-zero learning steps for example μ

$$M(t) = \sum_{\mu=1}^P x^\mu(t) \quad \text{total number of non-zero steps}$$

Proof of convergence

- assume a vector \mathbf{w}^* exists with $\{E^{\mu*} = \mathbf{w}^* \cdot \boldsymbol{\xi}^\mu S_R^\mu \geq c > 0\}_{\mu=1}^P$
- note that for any vector \mathbf{w} : $0 \leq \frac{(\mathbf{w}^* \cdot \mathbf{w})^2}{|\mathbf{w}^*|^2 |\mathbf{w}|^2} = \cos^2 \angle(\mathbf{w}, \mathbf{w}^*) \leq 1$
- weight vector after t learning steps $\mathbf{w}(t) = \frac{1}{N} \sum_{\mu=1}^P x^\mu(t) \boldsymbol{\xi}^\mu S_R^\mu$ $\mathbf{w}(0)=0$

$x^\mu(t)$ number of non-zero learning steps for example μ

$$M(t) = \sum_{\mu=1}^P x^\mu(t) \quad \text{total number of non-zero steps}$$

- projection $R(t) = \mathbf{w}^* \cdot \mathbf{w}(t)$

$$R(t) = \frac{1}{N} \sum_{\mu=1}^P x^\mu(t) (\mathbf{w}^* \cdot \boldsymbol{\xi}^\mu) S_R^\mu = \frac{1}{N} \sum_{\mu=1}^P x^\mu(t) \underbrace{E^{\mu*}}_{\geq c} \geq \frac{1}{N} c M(t)$$

- norm of the weight vector $Q(t) = \mathbf{w}(t)^2 \quad (\mathbf{w}^{*2} = Q^* = \text{const.})$

$$\begin{aligned}
 Q(t+1) &= \left(\mathbf{w}(t) + \frac{1}{N} \Theta [c - E^{\nu(t)}] \boldsymbol{\xi}^{\nu(t)} S_R^{\nu(t)} \right)^2 \\
 &= Q(t) + \frac{2}{N} \Theta [c - E^{\nu(t)}] E^{\nu(t)} + \frac{1}{N^2} \Theta^2 [c - E^{\nu(t)}] \left| \boldsymbol{\xi}^{\nu(t)} \right|^2
 \end{aligned}$$

- note: $\Theta^2[\dots] = \Theta[\dots]$

- norm of the weight vector $Q(t) = \mathbf{w}(t)^2 \quad (\mathbf{w}^{*2} = Q^* = \text{const.})$

$$Q(t+1) = \left(\mathbf{w}(t) + \frac{1}{N} \Theta [c - E^{\nu(t)}] \boldsymbol{\xi}^{\nu(t)} S_R^{\nu(t)} \right)^2$$

$$= Q(t) + \frac{2}{N} \Theta [c - E^{\nu(t)}] E^{\nu(t)} + \frac{1}{N^2} \Theta^2 [c - E^{\nu(t)}] \left| \boldsymbol{\xi}^{\nu(t)} \right|^2$$

- note: $\Theta^2[\dots] = \Theta[\dots]$

- among all inputs, one has the largest norm: $\max_{\mu} \{ |\boldsymbol{\xi}^{\mu}|^2 \} \equiv \Gamma N$

- zero learning step: $E^{\nu(t)} \geq c, \Theta = 0$

- non-zero learning step: $E^{\nu(t)} < c, \Theta = 1$

we replace E by c and
obtain an **upper bound**:

- norm of the weight vector $Q(t) = \mathbf{w}(t)^2 \quad (\mathbf{w}^{*2} = Q^* = \text{const.})$

$$Q(t+1) = \left(\mathbf{w}(t) + \frac{1}{N} \Theta [c - E^{\nu(t)}] \boldsymbol{\xi}^{\nu(t)} S_R^{\nu(t)} \right)^2$$

$$= Q(t) + \frac{2}{N} \Theta [c - E^{\nu(t)}] E^{\nu(t)} + \frac{1}{N^2} \Theta^2 [c - E^{\nu(t)}] \left| \boldsymbol{\xi}^{\nu(t)} \right|^2$$

- note: $\Theta^2[\dots] = \Theta[\dots]$

- among all inputs, one has the largest norm: $\max_{\mu} \{ |\boldsymbol{\xi}^{\mu}|^2 \} \equiv \Gamma N$

- zero learning step: $E^{\nu(t)} \geq c, \Theta = 0$

we replace E by c and
obtain an **upper bound**:

- non-zero learning step: $E^{\nu(t)} < c, \Theta = 1$

$$Q(t+1) \leq Q(t) + \frac{2}{N} c \Theta [c - E^{\nu(t)}] + \frac{1}{N} \Gamma \Theta [c - E^{\nu(t)}] \quad (Q(0)=0)$$

$$Q(t+1) \leq \frac{1}{N} (2c + \Gamma) M(t+1) \quad (M \text{ is the \# of non-zero steps so far})$$

we have the bounds: $R(t) \geq \frac{1}{N} c M(t)$ $Q(t) \leq \frac{1}{N} (2c + \Gamma) M(t)$

$$1 \geq \frac{R^2(t)}{Q^* Q(t)} \geq \frac{\frac{1}{N^2} c^2 M^2(t)}{Q^* \frac{1}{N} (2c + \Gamma) M(t)} = \frac{c^2}{Q^* N (2c + \Gamma)} M(t)$$

we have the bounds: $R(t) \geq \frac{1}{N} c M(t)$ $Q(t) \leq \frac{1}{N} (2c + \Gamma) M(t)$

$$1 \geq \frac{R^2(t)}{Q^* Q(t)} \geq \frac{\frac{1}{N^2} c^2 M^2(t)}{Q^* \frac{1}{N} (2c + \Gamma) M(t)} = \frac{c^2}{Q^* N (2c + \Gamma)} M(t)$$

the total number of non-zero steps is limited: $M(t) \leq \frac{(2c + \Gamma) N Q^*}{c^2}$

constants!

we have the bounds: $R(t) \geq \frac{1}{N} c M(t)$ $Q(t) \leq \frac{1}{N} (2c + \Gamma) M(t)$

$$1 \geq \frac{R^2(t)}{Q^* Q(t)} \geq \frac{\frac{1}{N^2} c^2 M^2(t)}{Q^* \frac{1}{N} (2c + \Gamma) M(t)} = \frac{c^2}{Q^* N (2c + \Gamma)} M(t)$$

the total number of non-zero steps is limited: $M(t) \leq \frac{(2c + \Gamma) N Q^*}{c^2}$

constants!

in one epoch pres. all data $\mu = 1, 2, \dots, P$

- either M does not change at all (all inputs classified correctly, done!)
- **or** M increases at least by one (at least one input was misclassified)

→ the total number of epochs is also limited
the **algorithm converges in finite time!**

we have the bounds: $R(t) \geq \frac{1}{N} c M(t)$ $Q(t) \leq \frac{1}{N} (2c + \Gamma) M(t)$

$$1 \geq \frac{R^2(t)}{Q^* Q(t)} \geq \frac{\frac{1}{N^2} c^2 M^2(t)}{Q^* \frac{1}{N} (2c + \Gamma) M(t)} = \frac{c^2}{Q^* N (2c + \Gamma)} M(t)$$

the total number of non-zero steps is limited: $M(t) \leq \frac{(2c + \Gamma) N Q^*}{c^2}$

constants!

in one epoch pres. all data $\mu = 1, 2, \dots, P$

- either M does not change at all (all inputs classified correctly, done!)
- **or** M increases at least by one (at least one input was misclassified)

→ the total number of epochs is also limited
the **algorithm converges in finite time!**

Remark: in the limit $c \rightarrow 0$, $\mathbf{w}^* \cdot \boldsymbol{\xi} > c$ implies that \mathbf{w}^* exists with $Q^* \propto c^2$

the bound for $M(t)$ remains finite

Perceptron Convergence Theorem

If the data set $\mathcal{D} = \{\xi^\mu, S_R^\mu\}_{\mu=1}^P$ is linearly separable,
the Rosenblatt Perceptron algorithm converges and yields a weight vector
 \mathbf{w} with $\mathbf{w} \cdot \xi^\mu S_R^\mu > 0$ for all $\mu = 1, \dots, P$

Remarks:

- this is one of the most fundamental results in the field
- we have assumed the existence of a solution
- it is difficult to *decide whether a given data set is linearly separable*
- the required number of steps is finite, but may be *large*, even if a solution exists

Learning a linearly separable rule from reliable examples

storage of a data set is not the primary goal of perceptron training

- assume: unknown lin. sep. function or **rule** $S_R(\xi) = \text{sign}(\mathbf{w}^* \cdot \xi)$
defines the correct classification for every possible input
(the *teacher perceptron* $\mathbf{w}^* \in \mathbb{R}^N$ parameterizes the rule)

- only available information: **example data**

$$\mathcal{ID} = \{\xi^\mu, S_R^\mu = S_R(\xi^\mu)\}_{\mu=1}^P$$

(correct labels S_R^μ provided by the teacher, absence of noise etc.)

Learning a linearly separable rule from reliable examples

storage of a data set is not the primary goal of perceptron training

- assume: unknown lin. sep. function or **rule** $S_R(\xi) = \text{sign}(\mathbf{w}^* \cdot \xi)$
defines the correct classification for every possible input
(the *teacher perceptron* $\mathbf{w}^* \in \mathbb{R}^N$ parameterizes the rule)

- only available information: **example data**

$$ID = \{\xi^\mu, S_R^\mu = S_R(\xi^\mu)\}_{\mu=1}^P$$

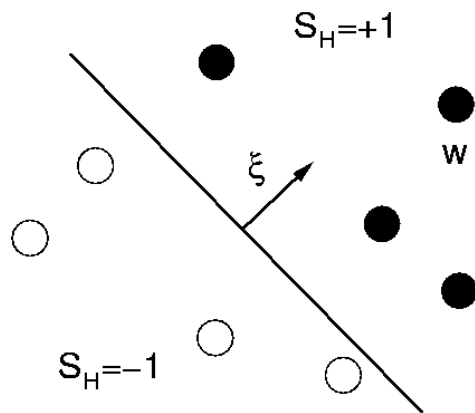
(correct labels S_R^μ provided by the teacher, absence of noise etc.)

- **training:** choice of *student* weights \mathbf{w} (w/o loss of generality: $\mathbf{w} \cdot \mathbf{w} = 1$)
parameterizes a **hypothesis** $S_H(\xi) = \text{sign}(\mathbf{w} \cdot \xi)$

extreme strategy: *zero training error*, learning in *version space*,

accept only hypotheses which are perfectly consistent with ID

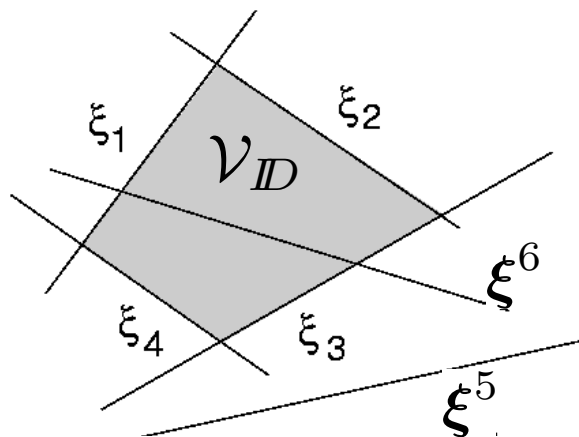
Learning in version space



dual geometrical interpretation:

each ξ, S^μ defines a **hyperplane in \mathbb{R}^N**

which separates the \mathbf{w} with $S_H(\xi)$ right/wrong
each example defines a correct half-space



set of data $ID \rightarrow$ **version space**

$$\mathcal{V}_{ID} = \left\{ \mathbf{w} \mid \left\{ \text{sign}(\mathbf{w} \cdot \xi^\mu) = S_R(\xi^\mu) \right\}_{\mu=1}^P \right\}$$

schematic example: $P = 4$, orientations S^μ not shown

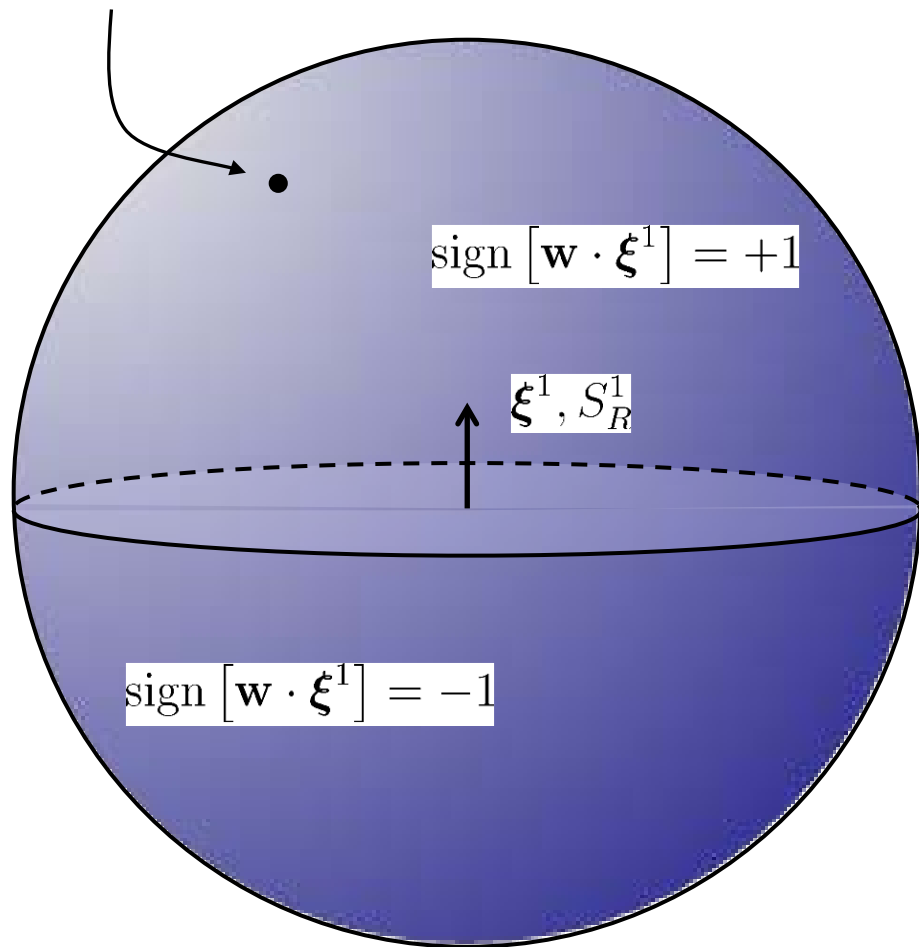
new data point ξ^5 *non-informative example:*
only one label possible

ξ^6 *informative example:*
version space shrinks!

illustration: version space

consider set of example data, normalized perceptron weight vectors

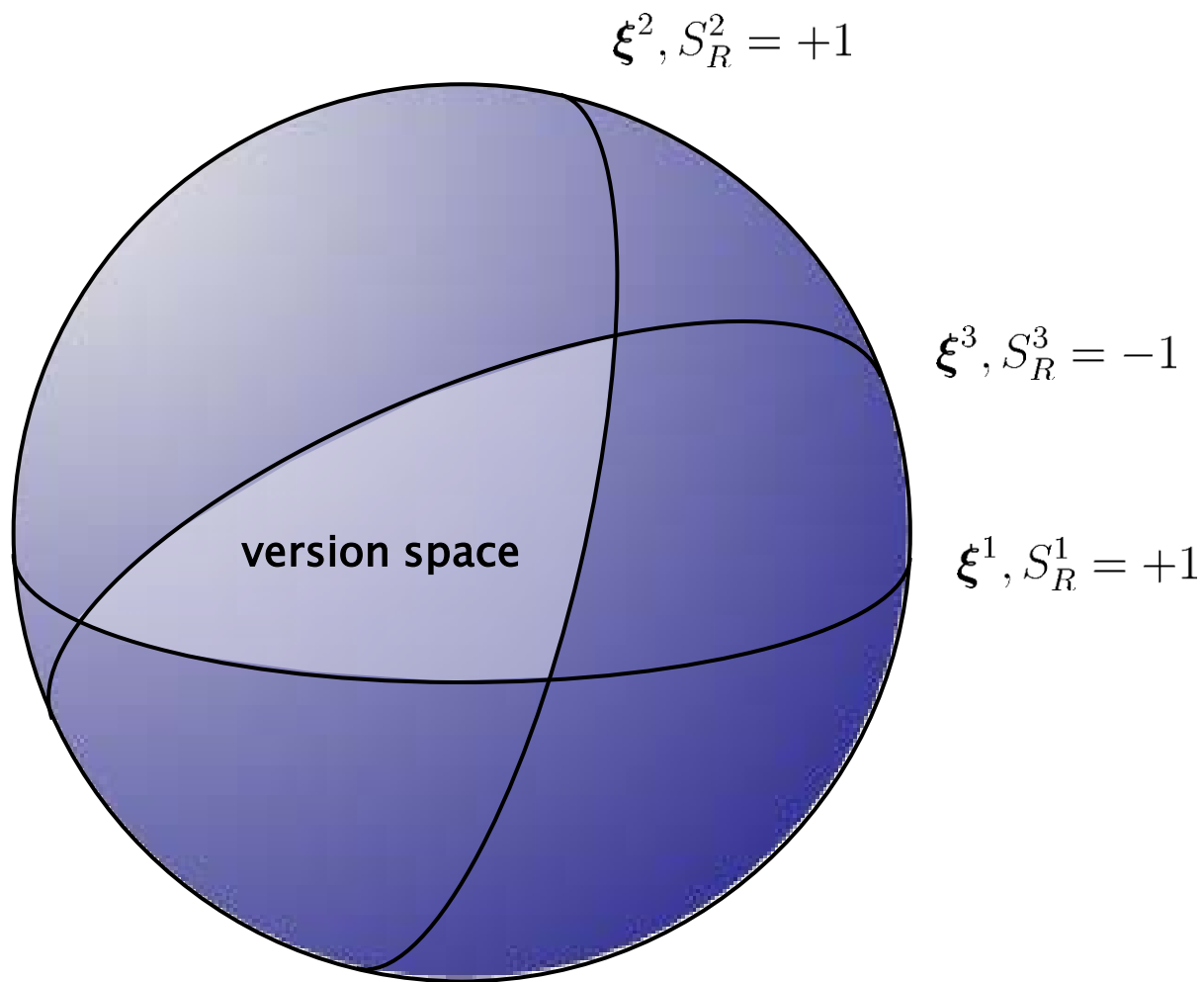
$$\mathbf{w} \in \mathbb{R}^N \text{ with } \mathbf{w} \cdot \mathbf{w} = |\mathbf{w}|^2 = \text{const.}$$



upper hemisphere:
correct weight vectors

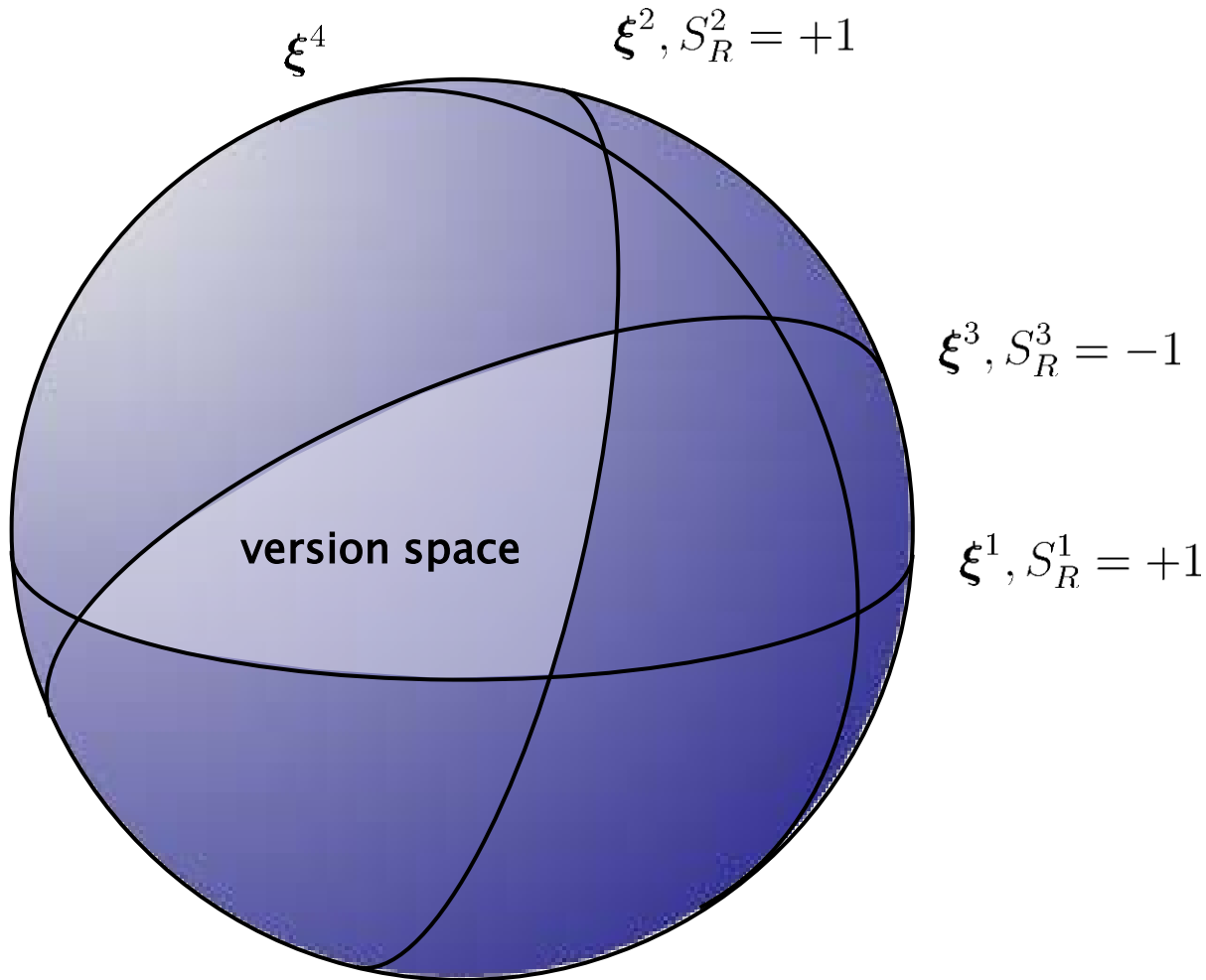
lower hemisphere:
incorrect weight vectors

a set of linearly separable examples defines “version space”
volume of all correct weight vectors

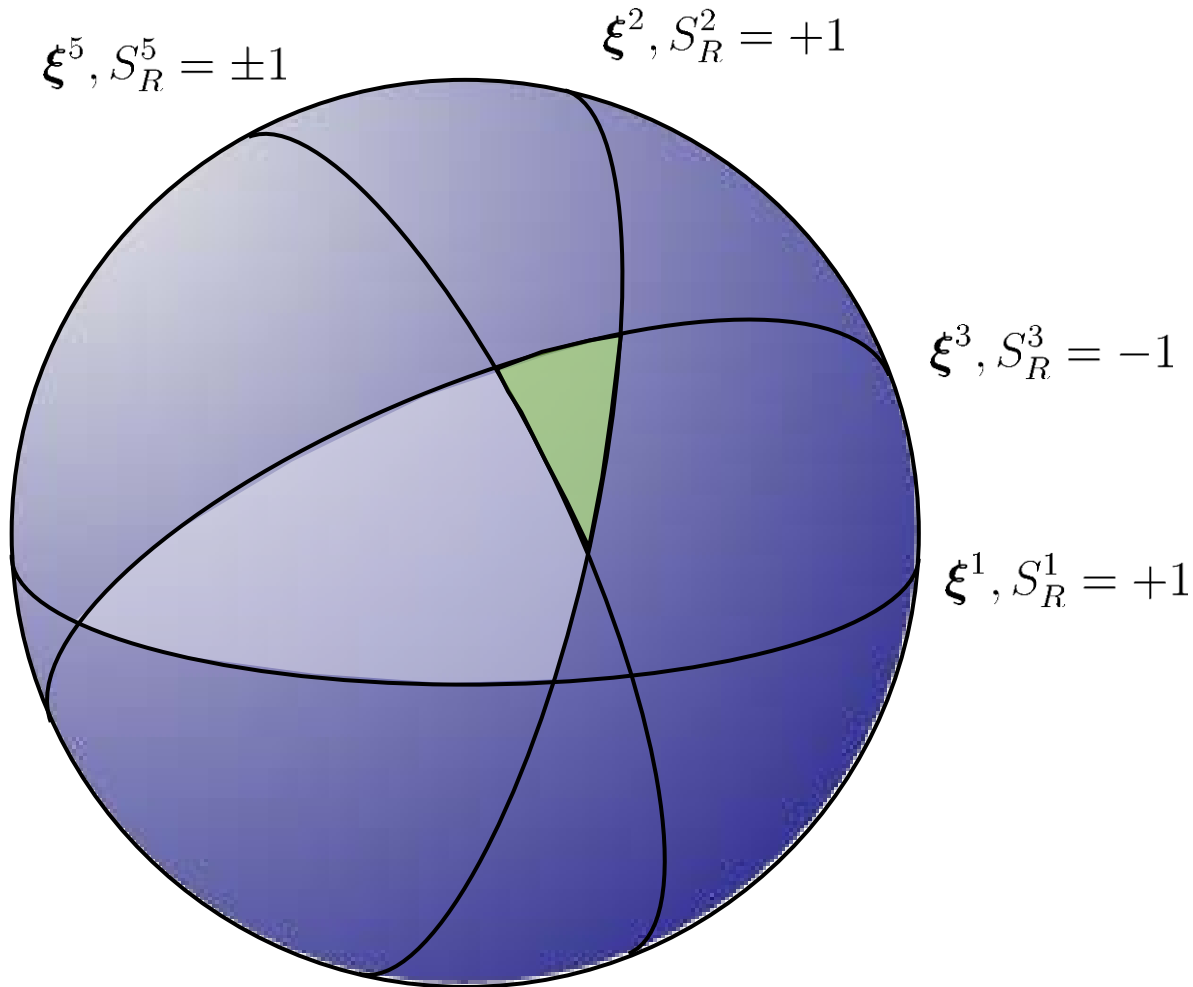


a “non-informative” example (non-ambiguous case)

does not shrink version space, only one label consistent with linear separability



an “informative” example (ambiguous case)
does shrink version space according to its label



Learning a linearly separable rule: generalization

- the unknown *teacher* \mathbf{w}^* define the (hom.) **lin. sep. rule** $S_R(\boldsymbol{\xi}) = \text{sign}[\mathbf{w}^* \cdot \boldsymbol{\xi}]$
- training process based on $ID = \{\boldsymbol{\xi}^\mu, S_R^\mu = S_R(\boldsymbol{\xi}^\mu)\}$ yields a *student* $S_H(\boldsymbol{\xi}) = \text{sign}(\mathbf{w} \cdot \boldsymbol{\xi})$ with $\mathbf{w} \in \mathbb{R}^N$

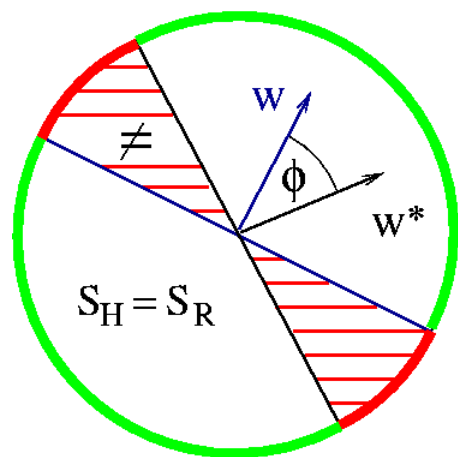
Learning a linearly separable rule: generalization

- the unknown *teacher* \mathbf{w}^* define the (hom.) **lin. sep. rule** $S_R(\boldsymbol{\xi}) = \text{sign}[\mathbf{w}^* \cdot \boldsymbol{\xi}]$
- training process based on $ID = \{\boldsymbol{\xi}^\mu, S_R^\mu = S_R(\boldsymbol{\xi}^\mu)\}$ yields a *student* $S_H(\boldsymbol{\xi}) = \text{sign}(\mathbf{w} \cdot \boldsymbol{\xi})$ with $\mathbf{w} \in \mathbb{R}^N$

consider a novel random input $\boldsymbol{\xi} \in \mathbb{R}^N$

generated with equal probability anywhere on the

N -sphere with $\boldsymbol{\xi}^2 = \Gamma$ (normalization)



probability for disagreement:

generalization error

$$\varepsilon_g = \frac{2\phi}{2\pi} = \frac{1}{\pi} \arccos \left[\frac{|\mathbf{w} \cdot \mathbf{w}^*|}{|\mathbf{w}| |\mathbf{w}^*|} \right]$$

(valid in arbitrary dimensions N)

learning in version space (consistent hypotheses, e.g. by Rosenblatt Perceptron)

growing number of examples $P = \alpha N$ with $\alpha \rightarrow \infty$

$$\frac{\mathbf{w} \cdot \mathbf{w}^*}{|\mathbf{w}| |\mathbf{w}^*|} \rightarrow 1 \quad \text{as } \mathbf{w} \text{ becomes parallel to } \mathbf{w}^*$$

for normalized \mathbf{w} : version space \mathcal{V} shrinks to a point

$$\epsilon_g(\alpha \rightarrow \infty) \rightarrow 0 \quad \text{zero generalization error}$$

The perceptron (storage) problem

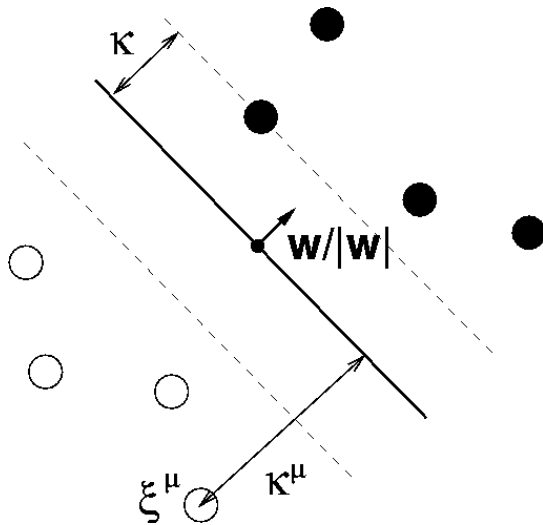
given a **dichotomy** of data $D_N^P = \{\boldsymbol{\xi}^\mu \in \mathbb{R}^N, S^\mu \in \{-1, +1\}\}_{\mu=1,2,\dots,P}$

find a vector $\mathbf{w} \in \mathbb{R}^N$, such that $\text{sign}(\mathbf{w} \cdot \boldsymbol{\xi}^\mu) = S^\mu$ for all μ .

Questions:

- When is a given dichotomy linearly separable (l.s.)?
- How many l.s. D_N^P exist? (*The capacity of a hyperplane*)
- If it exists, how can we find a perceptron vector \mathbf{w} ?
- If there are several/many solutions, which is *best*?
- What can we do for non-separable D_N^P ?

The perceptron of maximal stability

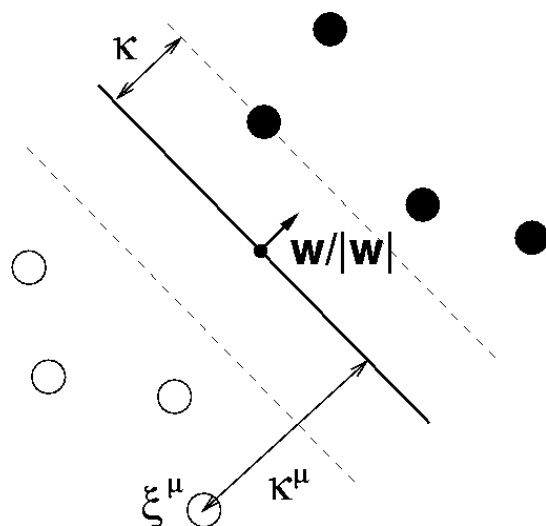


stability of example μ : $\kappa^\mu = \frac{E^\mu}{|\mathbf{w}|} = \frac{\mathbf{w} \cdot \xi^\mu S_R^\mu}{|\mathbf{w}|}$

measures the distance from the decision plane

stability of the perceptron $\kappa(\mathbf{w}) = \min_{\mu} \{\kappa^\mu\}$

The perceptron of maximal stability



stability of example μ : $\kappa^\mu = \frac{E^\mu}{|\mathbf{w}|} = \frac{\mathbf{w} \cdot \xi^\mu S_R^\mu}{|\mathbf{w}|}$

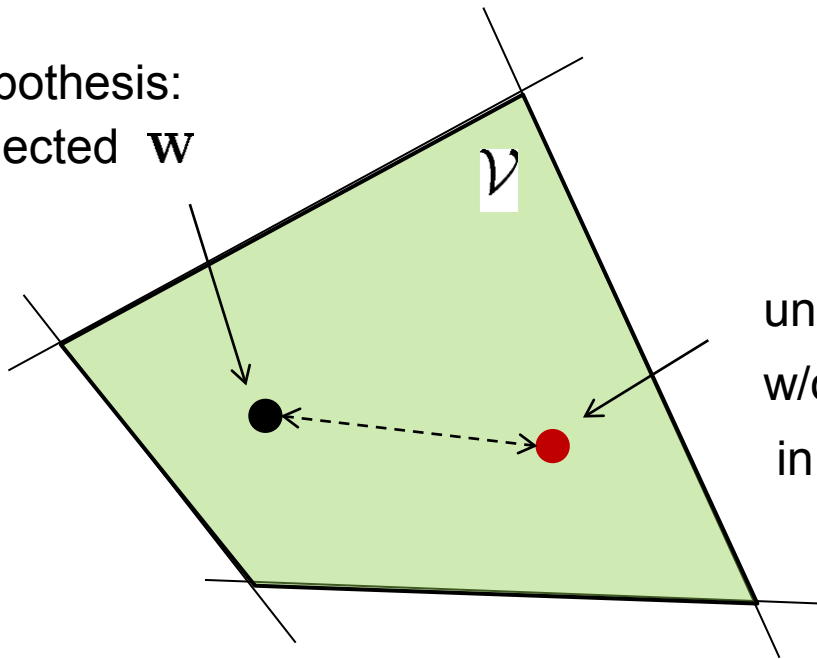
measures the distance from the decision plane

stability of the perceptron $\kappa(\mathbf{w}) = \min_{\mu} \{\kappa^\mu\}$

perceptron of maximal stability: $\mathbf{w}_{\max} = \operatorname{argmax}_{\mathcal{V}} \kappa(\mathbf{w})$:

- realize large separating gap between the two classes $\kappa(\mathbf{w}_{\max}) = \kappa_{\max}$
- classification is insensitive to small variations of ξ^μ e.g. due to *noise*
- corresponds to weight vector close to the center of version space in a l.s. rule
- yields (typically) good generalization ability

hypothesis:
selected \mathbf{w}



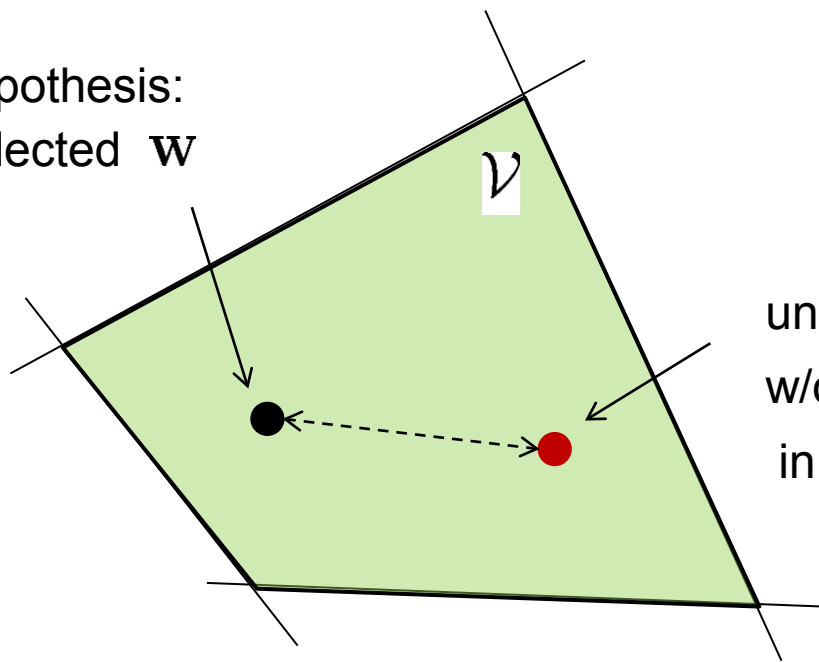
version space:

normalized vectors \mathbf{w}, \mathbf{w}^*

unknown rule, "teacher" \mathbf{w}^*

w/o additional knowledge: anywhere
in \mathcal{V} with equal probability

hypothesis:
selected \mathbf{w}



version space:

normalized vectors \mathbf{w}, \mathbf{w}^*

unknown rule, "teacher" \mathbf{w}^*

w/o additional knowledge: anywhere
in \mathcal{V} with equal probability

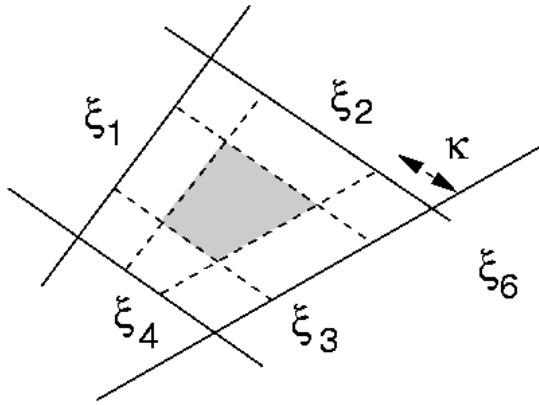
distance $d^2 = (\mathbf{w} - \mathbf{w}^*)^2 = \mathbf{w}^2 - 2\mathbf{w} \cdot \mathbf{w}^* + \mathbf{w}^{*2}$

small for large $\frac{\mathbf{w} \cdot \mathbf{w}^*}{|\mathbf{w}| |\mathbf{w}^*|}$, small ϵ_g

weight \mathbf{w} in the center (of mass) of version space:

lowest expectation value of distance d , best generalization error

on average over all possible positions of \mathbf{w}^*

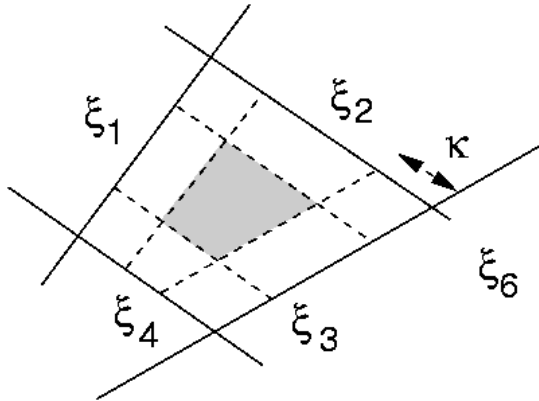


the restriction to $\mathbf{w} \in \mathcal{V}$ with

stability $\kappa(\mathbf{w}) \geq \kappa$ favors

vectors in the *center* of version space
(not quite the center of mass...)

shaded area shrinks to a point (\mathbf{w}_{\max}) as $\kappa \rightarrow \kappa_{\max}$



the restriction to $\mathbf{w} \in \mathcal{V}$ with

stability $\kappa(\mathbf{w}) \geq \kappa$ favors

vectors in the *center* of version space
(not quite the center of mass...)

shaded area shrinks to a point (\mathbf{w}_{\max}) as $\kappa \rightarrow \kappa_{\max}$

The minover algorithm

[Krauth and Mezard, 1989]

- iterative procedure, time steps $t = 0, 1, 2, 3, \dots$ $\mathbf{w}(0) = 0$
- given $\mathbf{w}(t)$, determine the example with lowest stability (minimal overlap)

$$\mu(t) \quad \text{with} \quad \kappa^{\mu(t)} = \min_{\nu} \left\{ \kappa^{\nu}(t) = \frac{\mathbf{w}(t) \boldsymbol{\xi}^{\nu} S_R^{\nu}}{|\mathbf{w}(t)|} \right\}$$

- update of the weight vector: $\mathbf{w}(t+1) = \mathbf{w}(t) + \frac{1}{N} \boldsymbol{\xi}^{\mu(t)} S_R^{\mu(t)}$

without proof:

If $\mathcal{D} = \{\xi^\mu, S_R^\mu\}_{\mu=1}^P$ is linearly separable, the *minover* algorithm converges and yields the perceptron weight vector \mathbf{W}_{max} of maximal stability

without proof:

If $\mathcal{D} = \{\xi^\mu, S_R^\mu\}_{\mu=1}^P$ is linearly separable, the *minover* algorithm converges and yields the perceptron weight vector \mathbf{w}_{max} of maximal stability

we show (only):

embedding strengths $\{x_{max}^\mu\}_{\mu=1}^P$ exist with $\mathbf{w}_{max} = \frac{1}{N} \sum_{\mu=1}^P x_{max}^\mu \xi^\mu S_R^\mu$ (#)

consider two perceptrons: $\mathbf{w}_1 = \frac{1}{N} \sum_{\mu=1}^P x_1^\mu \xi^\mu S_R^\mu$ and $\mathbf{w}_2 = \mathbf{w}_1 + \delta$
with $\delta \neq 0, \delta \perp \{\xi^\mu\}_{\mu=1}^P$

without proof:

If $ID = \{\xi^\mu, S_R^\mu\}_{\mu=1}^P$ is linearly separable, the *minover* algorithm converges and yields the perceptron weight vector \mathbf{W}_{max} of maximal stability

we show (only):

embedding strengths $\{x_{max}^\mu\}_{\mu=1}^P$ exist with $\mathbf{w}_{max} = \frac{1}{N} \sum_{\mu=1}^P x_{max}^\mu \xi^\mu S_R^\mu$ (#)

consider two perceptrons: $\mathbf{w}_1 = \frac{1}{N} \sum_{\mu=1}^P x_1^\mu \xi^\mu S_R^\mu$ and $\mathbf{w}_2 = \mathbf{w}_1 + \delta$
 with $\delta \neq 0, \delta \perp \{\xi^\mu\}_{\mu=1}^P$

$$\left. \begin{aligned} \mathbf{w}_2^2 &= \mathbf{w}_1^2 + \delta^2 + 2 \mathbf{w}_1 \cdot \delta \geq \mathbf{w}_1^2 \\ \mathbf{w}_2 \cdot \xi^\mu &= \mathbf{w}_1 \cdot \xi^\mu + \underbrace{\delta \cdot \xi^\mu}_0 = \mathbf{w}_1 \cdot \xi^\mu \end{aligned} \right\} \Rightarrow \kappa^\mu(\mathbf{w}_2) = \frac{\mathbf{w}_1 \cdot \xi^\mu S_R^\mu}{|\mathbf{w}_2|} < \kappa^\mu(\mathbf{w}_1)$$

$\Rightarrow \mathbf{W}_{max}$ can be written as (#),
 can be found by iterative training algorithms, potentially

Solving the perceptron storage problem

re-write the problem ...

consider a given data set $ID = \{\xi^\mu, S_R^\mu\}$

... find a vector \mathbf{w} with $S_H^\mu = \text{sign}(\mathbf{w} \cdot \xi^\mu) = S_R^\mu$ for all μ

Note: $\text{sign}(\mathbf{w} \cdot \xi^\mu) = S_R^\mu \Leftrightarrow \text{sign}(\mathbf{w} \cdot \xi^\mu S_R^\mu) = 1 \Leftrightarrow E^\mu = \mathbf{w} \cdot \xi^\mu S_R^\mu > 0$
(local potentials E^μ)

equivalent problem: solve a set of **linear inequalities** (in \mathbf{w})

... find a vector \mathbf{w} with $E^\mu = \mathbf{w} \cdot \xi^\mu S_R^\mu \geq c > 0$ for all μ

Instead of *inequalities*, try to solve P equations for N unknowns:

$$E^\mu = \sum_{j=1}^N w_j \xi_j^\mu S^\mu = 1 \quad \text{for all } \mu = 1, 2, \dots, P$$

(A) if no solution exists, find **approximation** by least square dev.:

$$\text{minimize } f = \frac{1}{2} \sum_{\mu=1}^P (1 - E^\mu)^2$$

minimization, e.g. by means of **gradient descent** with

$$\nabla_w f = - \sum_{\mu=1}^P (1 - E^\mu) \xi^\mu S^\mu$$

(B) system can be under-determined \rightarrow find a unique solution:

$$\text{minimize } \frac{1}{2} | \mathbf{w} |^2 \quad \text{under constraints } \{E^\mu = 1\}_{\mu=1}^P$$

Lagrange function

$$L = \frac{1}{2} | \mathbf{w} |^2 + \sum_{\mu=1}^P \lambda^\mu (1 - E^\mu)$$

necessary conditions for optimum:

$$\frac{\partial L}{\partial \lambda^\mu} = (1 - E^\mu) \stackrel{!}{=} 0$$

$$\nabla_w L = \mathbf{w} - \sum_{\mu=1}^P \lambda^\mu \boldsymbol{\xi}^\mu S^\mu \stackrel{!}{=} 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{\mu=1}^P \lambda^\mu \boldsymbol{\xi}^\mu S^\mu$$

Lagrange parameters \sim embedding strengths λ^μ (rescaled with N)

eliminate weights:

$$E^\nu = \sum_{\mu=1}^P \frac{1}{N} \underbrace{\sum_{k=1}^N (\xi_k^\mu S^\mu) (\xi_k^\nu S^\nu)}_{\equiv C^{\nu\mu}} \lambda^\mu \quad \sum_{j=1}^N w_j^2 \propto \sum_{\mu,\nu} \lambda^\nu C^{\nu\mu} \lambda^\mu$$

simplified problem: $\max_{\lambda} L = -\frac{1}{2} \sum_{\mu,\nu} \lambda^\nu C^{\nu\mu} \lambda^\mu + \sum_{\mu} \lambda^\mu$

gradient ascent with: $\frac{\partial L}{\partial \lambda^\rho} = 1 - \sum_{\mu} C^{\rho\mu} \lambda^\mu = (1 - E^\rho)$

in terms of weights:
the same as in (A) !!!

$$\Delta \mathbf{w} \propto \sum_{\rho} (1 - E^\rho) \xi^\rho S^\rho$$

Adaptive Linear Neuron (Widrow and Hoff, 1960)

Adaline algorithm: $\mathbf{w}(t) = \mathbf{w}(t-1) + \eta \left(1 - E^{\mu(t)}\right) \xi^{\mu(t)} S^{\mu(t)}$

sequence $\mu(t)$
of examples $x^{\mu}(t) = x^{\mu}(t-1) + \eta \left(1 - E^{\mu(t)}\right)$

iteration of weights / embedding strengths

more general: training of a linear unit with continuous output

$$\text{minimize } f = \frac{1}{2} \sum_{\mu=1}^P (h^{\mu} - E^{\mu})^2 \quad \text{with } h^{\mu} \in \mathbb{R}, \mu = 1, 2, \dots, P$$

$$f = \frac{1}{2} \sum_{\mu=1}^P (y^{\mu} - \mathbf{w}^{\top} \xi^{\mu})^2 \quad \text{with } y^{\mu} = h^{\mu} S^{\mu}$$

gradient based learning for linear regression (MSE)

frequent strategy: regression as a proxy for classification

“Science in action” ca. 1960

youtube video “science in action” with Bernard Widrow

<http://www.youtube.com/watch?v=IEFRtz68m-8>

Introduction:

- supervised learning, classification, regression
- machine learning “vs.” statistical modeling

Early (important!) approaches:

- linear threshold classifier, Rosenblatt’s *Perceptron*
- adaptive linear neuron, Widrow and Hoff’s *Adaline*

From **Perceptron** to **Support Vector Machine**

- large margin classification
- beyond linear separability

Distance-based systems

- prototypes: K-means and Vector Quantization
- from K-Neares_Neighbors to Learning Vector Quantization
- adaptive distance measures and relevance learning

Optimal stability by quadratic optimization

storage problem: find \mathbf{w} such that $\text{sign}(\mathbf{w}^\top \boldsymbol{\xi}^\mu) = S_R^\mu$ for a given $\mathcal{D} = \{\boldsymbol{\xi}^\nu, S_R^\nu\}$

optimal stability: maximize $\kappa(\mathbf{w})$ where $\kappa(\mathbf{w}) = \min_{\mu} \left\{ \kappa^\mu = \frac{\mathbf{w}^\top \boldsymbol{\xi}^\mu S_R^\mu}{|\mathbf{w}|} \right\}$

alternative formulation:

minimize $\frac{1}{2} \mathbf{w}^2$ subject to inequality constraints $\{E^\mu = \mathbf{w}^\top \boldsymbol{\xi}^\mu S_R^\mu \geq 1\}_{\mu=1}^P$

Note: the solution \mathbf{w}_{max} of the problem yields stability $\kappa_{max} = \frac{1}{|\mathbf{w}_{max}|}$

we know: search can be restricted to \mathbf{w} of the form $\mathbf{w} = \frac{1}{N} \sum_{\mu=1}^P x^\mu \boldsymbol{\xi}^\mu S_R^\mu$

$$\Rightarrow \mathbf{w}^2 = \frac{1}{N^2} \sum_{\mu, \nu=1}^P x^\mu x^\nu S_R^\mu S_R^\nu \boldsymbol{\xi}^\mu \cdot \boldsymbol{\xi}^\nu, \quad E^\nu = \frac{1}{N} \sum_{\mu=1}^P x^\mu S_R^\mu S_R^\nu \boldsymbol{\xi}^\mu \cdot \boldsymbol{\xi}^\nu$$

Notation:

correlation matrix $C \in \mathbb{R}^{P \times P}$ (outputs incorporated)

with elements $C^{\mu\nu} = \frac{1}{N} S_R^\mu S_R^\nu \boldsymbol{\xi}^\mu \cdot \boldsymbol{\xi}^\nu = \frac{1}{N} S_R^\mu S_R^\nu \sum_{j=1}^N \xi_j^\mu \xi_j^\nu$

***P*-vectors:** $\vec{x} = (x^1, x^2, \dots, x^P)^\top \in \mathbb{R}^P, \quad \vec{E} = (E^1, E^2, \dots, E^P)^\top \in \mathbb{R}^P$

inequalities $\vec{a} \geq \vec{b}$ iff $a^\mu \geq b^\mu$ for all $\mu = 1, 2, \dots, P$

“one-vector”: $\vec{1} = (1, 1, \dots, 1)^\top \in \mathbb{R}^P$

$$\vec{E} = C \vec{x} \quad \text{with components} \quad E^\mu = \sum_{\nu=1}^P C^{\mu\nu} x^\nu = \left(\frac{1}{N} \sum_{\nu=1}^P x^\nu \xi^\nu S_R^\nu \right) \cdot \xi^\mu S_R^\mu$$

$$\mathbf{w}^2 = \frac{1}{N} \vec{x}^\top C \vec{x} \geq 0 \quad \text{quadratic form} \quad \mathbf{w}^2 = \frac{1}{N} \sum_{\mu=1}^P x^\mu E^\mu = \frac{1}{N} \sum_{\mu,\nu=1}^P x^\mu C^{\mu\nu} x^\nu$$

(C is positive semi-definite)

We can formulate optimal stability completely in terms of embedding strengths:

$$\underset{\vec{x}}{\text{minimize}} \quad \frac{1}{2} \vec{x}^\top C \vec{x} \quad \text{subject to linear constraints} \quad \vec{E} = C \vec{x} \geq \vec{1}$$

This is a special case of a standard problem in *Quadratic Programming*:
minimize a nonlinear function under linear inequality constraints

Optimization theory: **Kuhn–Tucker theorem**

see, e.g., R. Fletcher, Practical Methods of Optimization (Wiley, 1987)
or <http://wikipedia.org> “Karush-Kuhn-Tucker-conditions” for a quick start

necessary conditions for a local solution of a general
non-linear optimization problem with equality and inequality constraints
here: **only inequality constraints** (see literature for the mixed case)

minimize $_{\vec{x}}$ $f(\vec{x})$ subject to $c_i(\vec{x}) \geq 0$ for $i=1,2,\dots,k$

Lagrange function: $\mathcal{L}(\vec{x}, \lambda_1, \lambda_2, \dots, \lambda_k) = f(\vec{x}) - \sum_{i=1}^k \lambda_i c_i(\vec{x})$

necessary conditions for solutions:

$\nabla_{\vec{x}} \mathcal{L} = 0$ stationarity (zero gradient)

$c_i(\vec{x}) \geq 0$ inequality constraints ($i = 1, 2, \dots, k$)

$\lambda_i \geq 0$ non-negative Lagrange parameters

$\lambda_i c_i(\vec{x}) = 0$ **complementarity**

Max. stability: **Kuhn–Tucker theorem** for a special non-linear optimization problem

$$\text{minimize}_{\vec{x}} \frac{1}{2} \vec{x}^\top C \vec{x} \quad \text{subject to} \quad C \vec{x} \geq \vec{1}$$

$$\text{Lagrange function: } \mathcal{L}(\vec{x}, \vec{\lambda}) = \frac{1}{2} \vec{x}^\top C \vec{x} - \vec{\lambda}^\top (C \vec{x} - \vec{1})$$

necessary conditions for solution:

$$C \vec{x} = C \vec{\lambda} \quad \text{stationarity} \quad \text{note: } \begin{aligned} \nabla_{\vec{x}} (\vec{x}^\top C \vec{x}) &= 2 C \vec{x} \\ \nabla_{\vec{x}} (\vec{\lambda}^\top C \vec{x}) &= \vec{\lambda}^\top C = C \vec{\lambda} \end{aligned}$$

$$C \vec{x} \geq \vec{1} \quad \text{linear separability!}$$

$$\vec{\lambda} \geq 0 \quad \text{non-negative Lagrange parameters}$$

$$\lambda^\mu ([C \vec{x}]^\mu - 1) = 0 \quad \text{complementarity } (\mu = 1, 2, \dots, P)$$

$$C \vec{x} = C \vec{\lambda} \quad \text{does not necessarily imply } \vec{x} = \vec{\lambda}$$

but: $\vec{\lambda}$ satisfies also all conditions, so we can replace \vec{x} by $\vec{\lambda}$
(and rename it to \vec{x})

here: any solution can be represented by a **Kuhn-Tucker (KT) point** \vec{x}^* with:

$$\vec{x}^* \geq \vec{0} \quad (\vec{x}^* \neq \vec{0})$$

non-negative embedding strengths (\leftarrow minover)

$$C \vec{x}^* \geq \vec{1}$$

linear separability

$$x^{*\mu} (1 - [C \vec{x}^*]^\mu) = 0 \quad \text{for all } \mu$$

complementarity

$$\text{implies also: } \vec{x}^{*T} C \vec{x}^* = \vec{x}^{*T} \vec{E}^* = \vec{x}^{*T} \vec{1}$$

consider two KT-points \vec{x}_1 and \vec{x}_2

$$\begin{aligned} 0 &\leq (\vec{x}_1 - \vec{x}_2)^\top C (\vec{x}_1 - \vec{x}_2) = \vec{x}_1^\top C \vec{x}_1 + \vec{x}_2^\top C \vec{x}_2 - \vec{x}_1^\top C \vec{x}_2 - \vec{x}_2^\top C \vec{x}_1 \\ &\quad \left(\text{use: } \vec{x}^*{}^\top C \vec{x}^* = \vec{x}^*{}^\top \vec{1} \right) = \vec{x}_1^\top (\vec{1} - C \vec{x}_2) + \vec{x}_2^\top (\vec{1} - C \vec{x}_1) \leq 0 \\ &\quad \left(\text{note: } C \vec{x}^* \geq \vec{1} \text{ and } \vec{x}^* \geq \vec{0} \right) \end{aligned}$$

$$\begin{aligned} \Rightarrow 0 &= (\vec{x}_1 - \vec{x}_2)^\top C (\vec{x}_1 - \vec{x}_2) \propto \sum_{\mu, \nu} (x_1^\mu - x_2^\mu) S_R^\mu \xi^\mu \cdot \xi^\nu S_R^\nu (x_1^\nu - x_2^\nu) \\ &= \left[\sum_{\mu} (x_1^\mu - x_2^\mu) S_R^\mu \xi^\mu \right]^2 \propto [\mathbf{w}_1 - \mathbf{w}_2]^2 \end{aligned}$$

→ all KT-points yield the same **unique** perceptron weight vector

→ any local solution is **globally optimal**

Duality, theory of Lagrange multipliers → equivalent formulation (*Wolfe dual*):

$$\text{maximize}_{\vec{x}} \tilde{f} = -\frac{1}{2} \vec{x}^T C \vec{x} + \vec{x}^T \vec{1} \quad \text{subject to} \quad \vec{x} \geq 0$$

Adaline: $\max_{\lambda} L = -\frac{1}{2} \sum_{\mu, \nu} \lambda^{\nu} C^{\nu\mu} \lambda^{\mu} + \sum_{\mu} \lambda^{\mu}$ (unconstrained)

Duality, theory of Lagrange multipliers \rightarrow equivalent formulation (*Wolfe dual*):

$$\underset{\vec{x}}{\text{maximize}} \quad \tilde{f} = -\frac{1}{2} \vec{x}^T C \vec{x} + \vec{x}^T \vec{1} \quad \text{subject to} \quad \vec{x} \geq 0$$

AdaTron algorithm: (Adaptive PercepTron)

[Anlauf and Biehl, 1989]

- sequential presentation of examples $\mathcal{D} = \{ \xi^\mu, S^\mu \}$
- gradient ascent w.r.t. \tilde{f} , projected onto $\vec{x} \geq 0$

$$x^\mu \rightarrow \max \{ 0, x^\mu + \eta (1 - [C\vec{x}]^\mu) \} \quad (0 < \eta < 2)$$

$$\eta \overbrace{\left[\nabla_{\vec{x}} \tilde{f} \right]^\mu}$$

Duality, theory of Lagrange multipliers → equivalent formulation (*Wolfe dual*):

$$\underset{\vec{x}}{\text{maximize}} \quad \tilde{f} = -\frac{1}{2} \vec{x}^T C \vec{x} + \vec{x}^T \vec{1} \quad \text{subject to} \quad \vec{x} \geq 0$$

AdaTron algorithm: (Adaptive PercepTron)

[Anlauf and Biehl, 1989]

- sequential presentation of examples $D = \{ \xi^\mu, S^\mu \}$
- gradient ascent w.r.t. \tilde{f} , projected onto $\vec{x} \geq 0$
 $x^\mu \rightarrow \max \{ 0, x^\mu + \eta (1 - [C\vec{x}]^\mu) \} \quad (0 < \eta < 2)$

for the proof of convergence one can show:

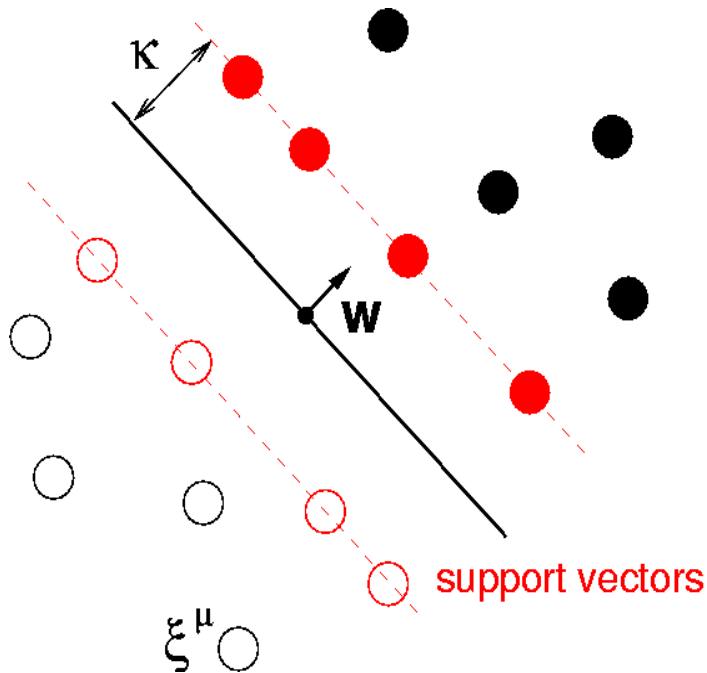
- for an arbitrary $\vec{x} \geq 0$ and a KT point \vec{x}^* : $\tilde{f}(\vec{x}^*) \geq \tilde{f}(\vec{x})$
- $\tilde{f}(x)$ is bounded from above in $\vec{x} \geq 0$
- $\tilde{f}(x)$ increases in every cycle through D , unless a KT point has been reached

Support Vectors

complementarity condition: $x^\mu (1 - E^\mu) = 0$ for all μ

i.e. either $\left\{ \begin{array}{l} E^\mu = 1 \\ x^\mu \geq 0 \end{array} \right\}$ or $\left\{ \begin{array}{l} E^\mu > 1 \\ x^\mu = 0 \end{array} \right\}$

examples ... have to be embedded or ... are stabilized “automatically”



the weights $w \propto \sum_{\mu} x^\mu \xi^\mu S^\mu$
depend (explicitly) only on a subset of \mathcal{ID}

if these support vectors were known
in advance, training could be restricted
to the subset

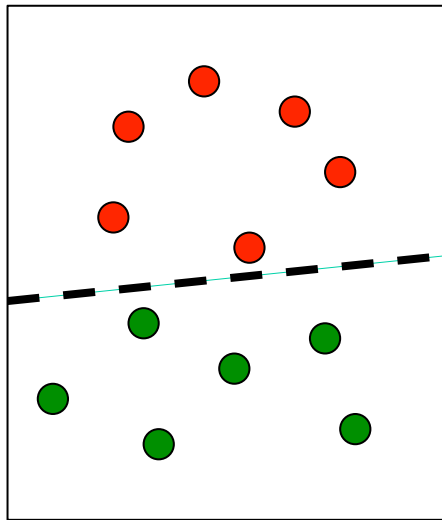
(unfortunately they are not...)

... is only possible if

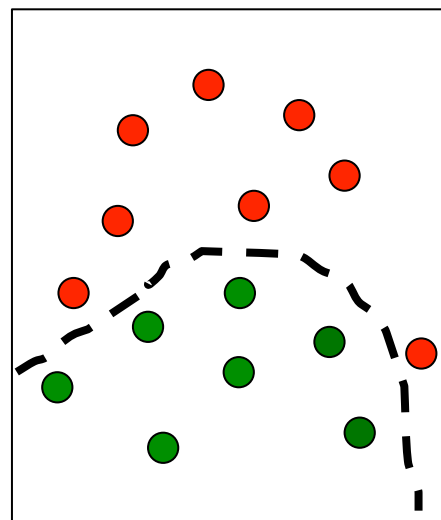
- the data set is linearly separable

... even then, it only makes sense if

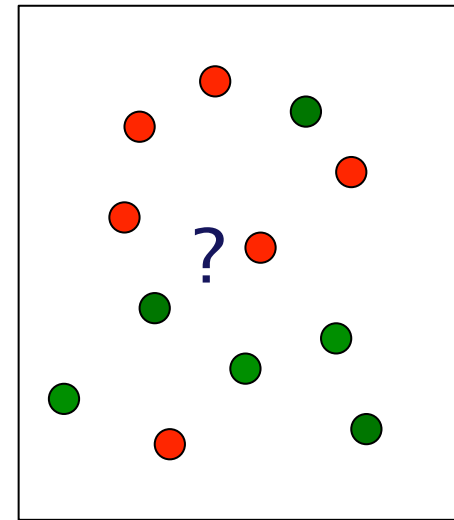
- the unknown rule is a linearly separable function
- the data set is reliable (*noise-free*)



lin. separable



nonlin. boundary



noisy data (?)

Classification beyond linear separability

assume $D = \{\xi^\mu, S^\mu\}$ is **not linearly** separable - what can we do?

potential reasons: noisy data, more complex problem

- accept an approximation by a linearly separable function → see “pocket algorithm”
and “large margin with errors”
- construct more complex architectures from perceptron-like units.
e.g. multilayer networks (universal classifiers, difficult training)

Classification beyond linear separability

assume $D = \{\xi^\mu, S^\mu\}$ is **not linearly** separable - what can we do?

potential reasons: noisy data, more complex problem

- accept an approximation by a linearly separable function

→ see “pocket algorithm”
and “large margin with errors”

- construct more complex architectures from perceptron-like units.
e.g. multilayer networks (universal classifiers, difficult training)

- consider *ensembles* of perceptrons

train several student perceptrons $S^{(j)} = \text{sign} [\mathbf{w}^{(j)} \cdot \xi]$

combine the $S^{(j)}$ into an *ensemble classifier*, e.g. by majority vote $S_H = \text{sign} \left[\sum_j S^{(j)} \right]$

competing aims:

- each student should make a *small* number of errors
- the perceptrons should differ significantly

see also: [Decision Trees and Forests \(lectures by Dalya Baron\)](#)

- employ a linear decision boundary, but after a non-linear transformation of the data to an M -dim. feature space ($M=N$ is possible, but not required)

$$S_H(\boldsymbol{\xi}) = \text{sign} [\underline{W} \cdot \underline{\Psi}(\boldsymbol{\xi})] \quad \text{with } \underline{W} \in \mathbb{R}^M \quad M\text{-dim. weight vector}$$

$$\underline{\Psi}(\boldsymbol{\xi}) \in \mathbb{R}^M \quad \text{non-linear transformation}$$

$$\mathbb{R}^N \rightarrow \mathbb{R}^M$$

for a given, explicit transformation $\underline{\Psi}(\boldsymbol{\xi})$, perceptron training can be applied in \mathbb{R}^M

- employ a linear decision boundary, but after a non-linear transformation of the data to an M -dim. feature space ($M=N$ is possible, but not required)

$$S_H(\underline{\xi}) = \text{sign} [\underline{W} \cdot \underline{\Psi}(\underline{\xi})] \quad \text{with } \underline{W} \in \mathbb{R}^M \quad M\text{-dim. weight vector}$$

$$\underline{\Psi}(\underline{\xi}) \in \mathbb{R}^M \quad \text{non-linear transformation}$$

$$\mathbb{R}^N \rightarrow \mathbb{R}^M$$

for a given, explicit transformation $\underline{\Psi}(\underline{\xi})$, perceptron training can be applied in \mathbb{R}^M

important examples:

- Rosenblatt's **perceptron with masks**, transformed features $\Psi_j(\underline{\xi})$
- **Support Vector Machines:** $M > N$ transformation to higher-dim. space
 $\underline{\Psi}(\underline{\xi})$ is defined only implicitly (kernel-trick)
 \underline{W} perceptron of optimal stability in M dimensions
- very frequent approach (e.g. multilayered feed-forward neural networks):
 replace classification by regression in the training phase

approximation: linearly separable function, accept errors, e.g.

- large margins with errors

admit disagreements w.r.t. training data, but keep basic idea of optimal stability

$$\text{minimize}_{\mathbf{w}, \beta} \quad \frac{1}{2} \mathbf{w}^2 + \gamma \sum_{\mu=1}^P \beta^{\mu} \quad \text{subject to } E^{\mu} \geq 1 - \beta^{\mu}$$

and $\beta^{\mu} \geq 0$ for all μ

slack variables $\left\{ \begin{array}{l} \beta^{\mu} = 0 \leftrightarrow E^{\mu} \geq 1 \\ \beta^{\mu} > 0 \leftrightarrow E^{\mu} < 1 \end{array} \right.$ includes errors with $E^{\mu} < 0$

- large margins with errors

admit disagreements w.r.t. training data, but keep basic idea of optimal stability

$$\text{minimize}_{\mathbf{w}, \beta} \quad \frac{1}{2} \mathbf{w}^2 + \gamma \sum_{\mu=1}^P \beta^\mu \quad \text{subject to } E^\mu \geq 1 - \beta^\mu$$

and $\beta^\mu \geq 0$ for all μ

slack variables $\left\{ \begin{array}{l} \beta^\mu = 0 \leftrightarrow E^\mu \geq 1 \\ \beta^\mu > 0 \leftrightarrow E^\mu < 1 \end{array} \right.$ includes errors with $E^\mu < 0$

rewritten in terms of embedding strengths (see above for notation)

$$\text{minimize}_{\vec{x}, \vec{\beta}} \quad \frac{1}{2} \vec{x}^\top C \vec{x} + \gamma \vec{\beta} \cdot \vec{1} \quad \text{subject to } C \vec{x} \geq \vec{1} - \vec{\beta}$$

and $\vec{\beta} \geq 0$

dual problem: (elimination of slack variables!)

$$\text{maximize}_{\vec{x}} \quad -\frac{1}{2} \vec{x}^\top C \vec{x} + \vec{1} \cdot \vec{x} \quad \text{subject to} \quad 0 \leq \vec{x} \leq \gamma \vec{1}$$

positive and upper-bounded embedding strengths

parameter γ - limits the growth of x^μ for misclassified data points

- controls a compromise between aims of *large margin* and *low error*
 - has to be chosen appropriately, e.g. by validation methods (later chapter)
- note: even for lin. sep. data the optimum can include misclassifications!

dual problem: (elimination of slack variables!)

$$\text{maximize}_{\vec{x}} \quad -\frac{1}{2} \vec{x}^\top C \vec{x} + \vec{1} \cdot \vec{x} \quad \text{subject to} \quad 0 \leq \vec{x} \leq \gamma \vec{1}$$

positive and upper-bounded embedding strengths

parameter γ - limits the growth of x^μ for misclassified data points

- controls a compromise between aims of *large margin* and *low error*
 - has to be chosen appropriately, e.g. by validation methods (later chapter)
- note: even for lin. sep. data the optimum can include misclassifications!

Example algorithm:

AdaTron with errors (projected gradient ascent)

$\tilde{x}^\mu \leftarrow x^\mu + \eta (1 - [C\vec{x}]^\mu)$ gradient step

$\hat{x}^\mu \leftarrow \max \{0, \tilde{x}^\mu\}$ enforce non-negative embeddings

$x^\mu \leftarrow \min \{\gamma, \hat{x}^\mu\}$ limit embedding strengths to $x^\mu \leq \gamma$



- Perceptron of optimal stability: **support vectors**
- SVM: non-linear transformation to high-dim. feature space
- implicit kernel formulation, Mercer's theorem

history: www.svms.org

- Vapnik and Lerner (1963) introduce the Generalized Portrait algorithm
- Aizerman, Braverman and Rozonoer (1964) introduced the geometrical interpretation of the kernels
- Vapnik and Chervonenkis (1964) further develop the Generalized Portrait algorithm.
- Vapnik (1982) wrote an English translation of his 1979 book.
- SVMs close to their current form were first introduced with a paper at the COLT 1992 conference (Boser, Guyon and Vapnik 1992).
- In 1995 the soft margin classifier was introduced by Cortes and Vapnik (1995)

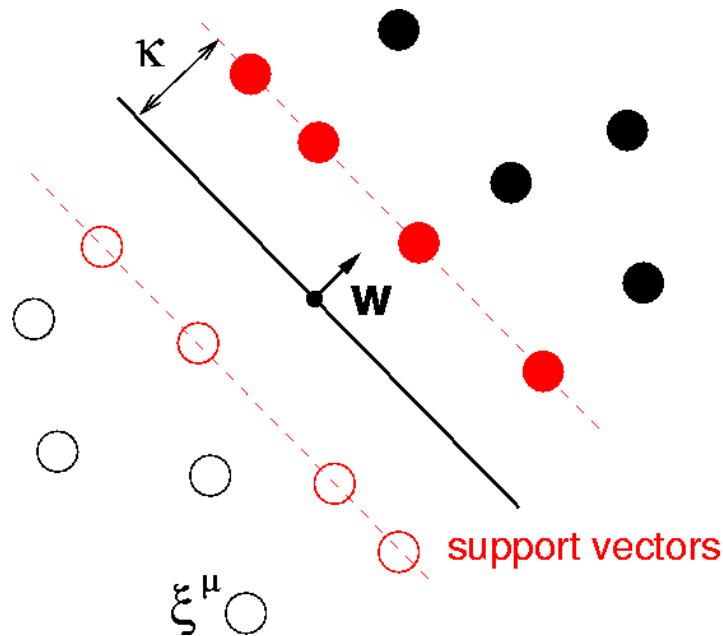
Support Vectors (linear separable case)

(reminder)

complementarity condition: $x^\mu (1 - E^\mu) = 0$ for all μ

i.e. either $\left\{ \begin{array}{l} E^\mu = 1 \\ x^\mu \geq 0 \end{array} \right\}$ or $\left\{ \begin{array}{l} E^\mu > 1 \\ x^\mu = 0 \end{array} \right\}$

examples ... have to be embedded or ... are stabilized “automatically”



the weights $\vec{w} \propto \sum_{\mu} x^\mu \xi^\mu S^\mu$

depend (explicitly) only on a subset of \mathcal{D}

if these support vectors were known
in advance, training could be restricted
to the subset

basic idea:

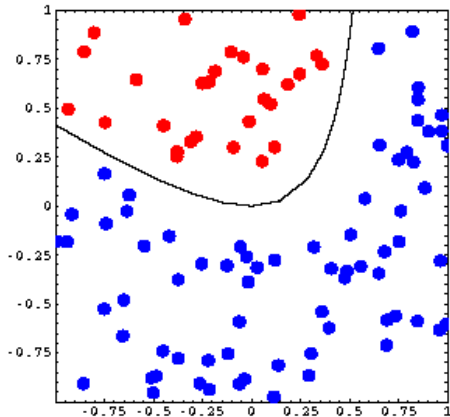
employ a linear decision boundary, but after a non-linear transformation of the data

$$S_H^\mu = \text{sign} [\underline{W} \cdot \underline{\Psi}(\xi^\mu)], \quad \xi \in \mathbb{R}^N \rightarrow \underline{\Psi}(\xi) \in \mathbb{R}^M \quad \text{with weights } \underline{W} \in \mathbb{R}^M$$

SVM: transformation with $M > N$ to high-dim. feature space

An illustrative example (c/o R. Dietrich, PhD thesis)

consider original, two-dimensional data (x_1, x_2)



basic idea:

employ a linear decision boundary, but after a non-linear transformation of the data

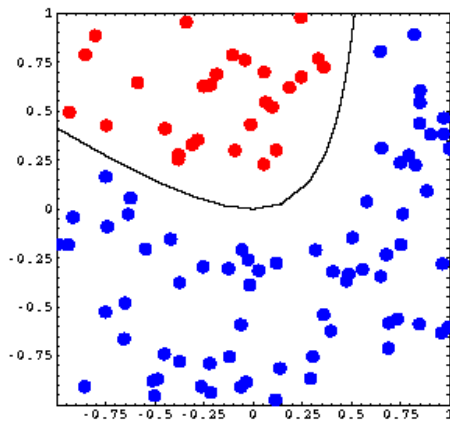
$$S_H^\mu = \text{sign} [\underline{W} \cdot \underline{\Psi}(\xi^\mu)], \quad \xi \in \mathbb{R}^N \rightarrow \underline{\Psi}(\xi) \in \mathbb{R}^M \quad \text{with weights } \underline{W} \in \mathbb{R}^M$$

SVM: transformation with $M > N$ to high-dim. feature space

An illustrative example (c/o R. Dietrich, PhD thesis)

consider original, two-dimensional data (x_1, x_2)

and the non-linear transformed data $\underline{\Psi}(x_1, x_2) = (x_1^2, \sqrt{2} x_1 x_2, x_2) \in \mathbb{R}^3$



basic idea:

employ a linear decision boundary, but after a non-linear transformation of the data

$$S_H^\mu = \text{sign} [\underline{W} \cdot \underline{\Psi}(\xi^\mu)], \quad \xi \in \mathbb{R}^N \rightarrow \underline{\Psi}(\xi) \in \mathbb{R}^M \quad \text{with weights } \underline{W} \in \mathbb{R}^M$$

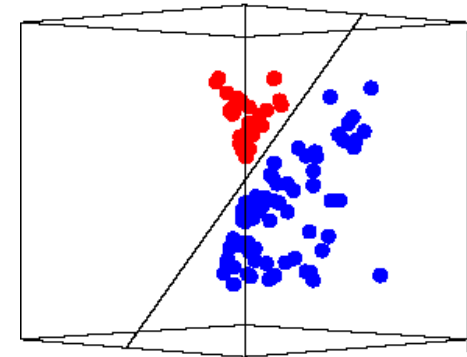
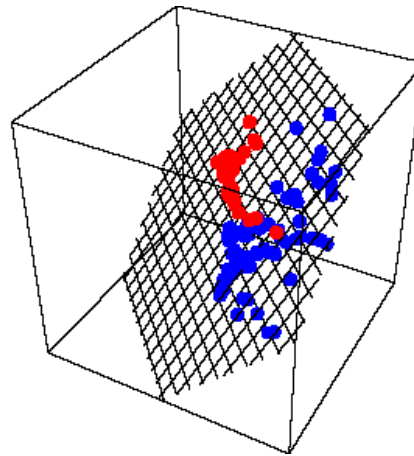
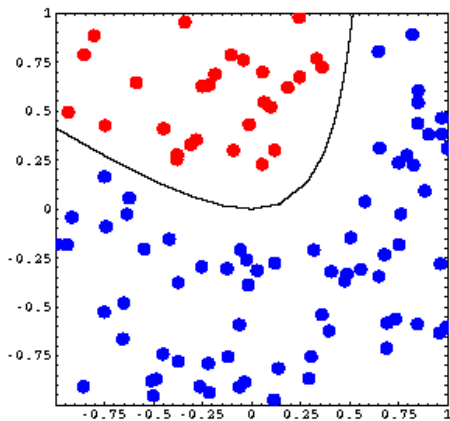
SVM: transformation with $M > N$ to high-dim. feature space

An illustrative example (c/o R. Dietrich, PhD thesis)

consider original, two-dimensional data (x_1, x_2)

and the non-linear transformed data $\underline{\Psi}(x_1, x_2) = (x_1^2, \sqrt{2} x_1 x_2, x_2) \in \mathbb{R}^3$

$$S^\mu = \text{sign} (\underline{W} \cdot \underline{\Psi}(x_1, x_2)) \quad \text{with } \vec{W} = (1, 1, -1)$$



the non-separable classification in \mathbb{R}^2 becomes linearly separable in \mathbb{R}^3

assume: transformation guarantees linear separability of $\{ \underline{\Psi}(\xi^\mu), S^\mu \}$

→ a vector \underline{W} exists with $S_H^\mu = \text{sign}(\underline{W} \cdot \underline{\Psi}(\xi^\mu))$ for all μ .

optimal stability:

$$\underset{\underline{W}}{\text{maximize}} \kappa(\underline{W}) \quad \text{where} \quad \kappa(\underline{W}) = \min_{\mu} \left\{ \kappa^\mu = \frac{\underline{W} \cdot \underline{\Psi}(\xi^\mu) S^\mu}{|\underline{W}|} \right\}$$

Exact same structure as the original perceptron problem – all above results from optimization theory apply accordingly

assume: transformation guarantees linear separability of $\{ \underline{\Psi}(\xi^\mu), S^\mu \}$

→ a vector \underline{W} exists with $S_H^\mu = \text{sign}(\underline{W} \cdot \underline{\Psi}(\xi^\mu))$ for all μ .

optimal stability:

$$\underset{\underline{W}}{\text{maximize}} \kappa(\underline{W}) \quad \text{where} \quad \kappa(\underline{W}) = \min_{\mu} \left\{ \kappa^\mu = \frac{\underline{W} \cdot \underline{\Psi}(\xi^\mu) S^\mu}{|\underline{W}|} \right\}$$

Exact same structure as the original perceptron problem – all above results from optimization theory apply accordingly

re-formulate:

$$\underset{\vec{X}}{\text{minimize}} \frac{1}{2} \vec{X}^T \Gamma \vec{X} \quad \text{subject to} \quad \Gamma \vec{X} \geq \vec{1}$$

here:

$$\underline{W} = \frac{1}{M} \sum_{\mu=1}^P X^\mu \underline{\Psi}(\xi^\mu) S^\mu \quad \Gamma^{\mu\nu} = \frac{1}{M} S^\mu \underline{\Psi}(\xi^\mu) \cdot \underline{\Psi}(\xi^\nu) S^\nu$$

$$\underline{W}^2 = \frac{1}{M} \vec{X}^T \Gamma \vec{X}$$

Kernel formulation

consider the function $K : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$ with $K(\xi^\mu, \xi^\nu) = \frac{1}{M} \underline{\Psi}(\xi^\mu) \cdot \underline{\Psi}(\xi^\nu)$

re-write in terms of this *kernel function*

- the classification scheme: $S_H(\xi) = \text{sign} (\underline{W} \cdot \underline{\Psi}(\xi))$

$$= \text{sign} \left(\sum_{\mu=1}^P X^\mu S^\mu \underline{\Psi}(\xi^\mu) \cdot \underline{\Psi}(\xi) \right) = \text{sign} \left(\sum_{\mu=1}^P X^\mu S^\mu K(\xi^\mu, \xi) \right)$$

Kernel formulation

consider the function $K : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$ with $K(\xi^\mu, \xi^\nu) = \frac{1}{M} \underline{\Psi}(\xi^\mu) \cdot \underline{\Psi}(\xi^\nu)$

re-write in terms of this *kernel function*

- the classification scheme: $S_H(\xi) = \text{sign} (\underline{W} \cdot \underline{\Psi}(\xi))$

$$= \text{sign} \left(\sum_{\mu=1}^P X^\mu S^\mu \underline{\Psi}(\xi^\mu) \cdot \underline{\Psi}(\xi) \right) = \text{sign} \left(\sum_{\mu=1}^P X^\mu S^\mu K(\xi^\mu, \xi) \right)$$

- training algorithms for the embedding strengths, just one example:

$$\textit{Kernel AdaTron} \quad X^\mu \rightarrow \max \left\{ 0, X^\mu + \eta \left(1 - S^\mu \sum_{\nu=1}^P S^\nu X^\nu K(\xi^\mu, \xi^\nu) \right) \right\}$$

Kernel formulation

consider the function $K : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$ with $K(\xi^\mu, \xi^\nu) = \frac{1}{M} \underline{\Psi}(\xi^\mu) \cdot \underline{\Psi}(\xi^\nu)$

re-write in terms of this *kernel function*

- the classification scheme: $S_H(\xi) = \text{sign} (\underline{W} \cdot \underline{\Psi}(\xi))$

$$= \text{sign} \left(\sum_{\mu=1}^P X^\mu S^\mu \underline{\Psi}(\xi^\mu) \cdot \underline{\Psi}(\xi) \right) = \text{sign} \left(\sum_{\mu=1}^P X^\mu S^\mu K(\xi^\mu, \xi) \right)$$

- training algorithms for the embedding strengths, just one example:

$$\textit{Kernel AdaTron} \quad X^\mu \rightarrow \max \left\{ 0, X^\mu + \eta \left(1 - S^\mu \sum_{\nu=1}^P S^\nu X^\nu K(\xi^\mu, \xi^\nu) \right) \right\}$$

– no explicit use of the transformed feature vectors $\underline{\Psi}(\xi)$

– only dot-products required, which can be expressed in terms of the *kernel*

so far: define non-linear $\underline{\Psi}(\xi) \in \mathbb{R}^M$, find corresponding kernel function $K(\xi^\mu, \xi^\nu)$

now: as we will never use $\underline{\Psi}(\xi)$ explicitly, why not start with defining a kernel function in the first place?

for practical purposes, we need not know $\underline{\Psi}$ nor its dimension M

Question: does a given kernel K correspond to some valid transformation $\underline{\Psi}$?

so far: define non-linear $\underline{\Psi}(\xi) \in \mathbb{R}^M$, find corresponding kernel function $K(\xi^\mu, \xi^\nu)$

now: as we will never use $\underline{\Psi}(\xi)$ explicitly, why not start with defining a kernel function in the first place?

for practical purposes, we need not know $\underline{\Psi}$ nor its dimension M

Question: does a given kernel K correspond to some valid transformation $\underline{\Psi}$?

Mercer's Theorem (sufficient condition)

a given kernel function K can be written as $K(\xi^\mu, \xi^\nu) = \underline{\Psi}(\xi^\mu) \cdot \underline{\Psi}(\xi^\nu)$, if

$$\int \int g(\xi^\mu) K(\xi^\mu, \xi^\nu) g(\xi^\nu) d^N \xi^\mu d^N \xi^\nu \geq 0 \quad \text{holds true}$$

$$\text{for all functions } g \text{ with finite norm } \int g(\xi)^2 d^N \xi < \infty$$

popular classes of kernels (which satisfy Mercer's condition)

- **polynomial kernels** of degree (up to) q , e.g.

$$K(\xi^\mu, \xi) = (1 + \xi^\mu \cdot \xi)^q \quad \text{yields} \quad S_H(\xi) = \text{sign} \left[\sum_{\mu=1}^P X^\mu S^\mu (1 + \xi^\mu \cdot \xi)^q \right]$$

linear kernel ($q = 1$)

$$K(\xi^\mu, \xi) = (1 + \xi^\mu \cdot \xi) \quad \text{yields} \quad S_H(\xi) = \text{sign} \left[\underbrace{\Theta}_{\sum_{\mu} X^\mu S^\mu} + \sum_{\mu=1}^P X^\mu S^\mu \xi^\mu \cdot \xi \right]$$

= perceptron with threshold in original space

popular classes of kernels (which satisfy Mercer's condition)

- **polynomial kernels** of degree (up to) q , e.g.

$$K(\xi^\mu, \xi) = (1 + \xi^\mu \cdot \xi)^q \quad \text{yields} \quad S_H(\xi) = \text{sign} \left[\sum_{\mu=1}^P X^\mu S^\mu (1 + \xi^\mu \cdot \xi)^q \right]$$

linear kernel ($q = 1$)

$$K(\xi^\mu, \xi) = (1 + \xi^\mu \cdot \xi) \quad \text{yields} \quad S_H(\xi) = \text{sign} \left[\underbrace{\Theta}_{\sum_{\mu} X^\mu S^\mu} + \sum_{\mu=1}^P X^\mu S^\mu \xi^\mu \cdot \xi \right]$$

= perceptron with threshold in original space

quadratic kernel ($q = 2$)

$$K(\xi^\mu, \xi) = (1 + \xi^\mu \cdot \xi)^2 = 1 + 2 \sum_j \xi_j^\mu \boxed{\xi_j} + \sum_{j,k} \xi_j^\mu \xi_k^\mu \boxed{\xi_j \xi_k}$$

-> perceptron with respect to feature vectors containing all single and products of 2 original features

$$(\xi_1, \xi_2, \dots, \xi_N, \xi_1 \xi_1, \xi_1 \xi_2, \dots, \dots, \xi_{N-1} \xi_N, \xi_N \xi_N)^T \quad \text{i.e.} \quad M = N + N(N - 1)/2$$

- **Radial basis function (RBF) kernel**

$$K(\xi^\mu, \xi) = \exp \left[-\frac{|\xi^\mu - \xi|^2}{2\sigma} \right]$$

involves all powers of the features, “ $M \rightarrow \infty$ ”

attractive aspects of the SVM approach:

- optimization problem is uniquely solvable (no local minima)
- efficient training algorithms are known
- maximum stability facilitates good generalization ability

... if the kernel (its parameters) is (are) appropriately chosen

in practice:

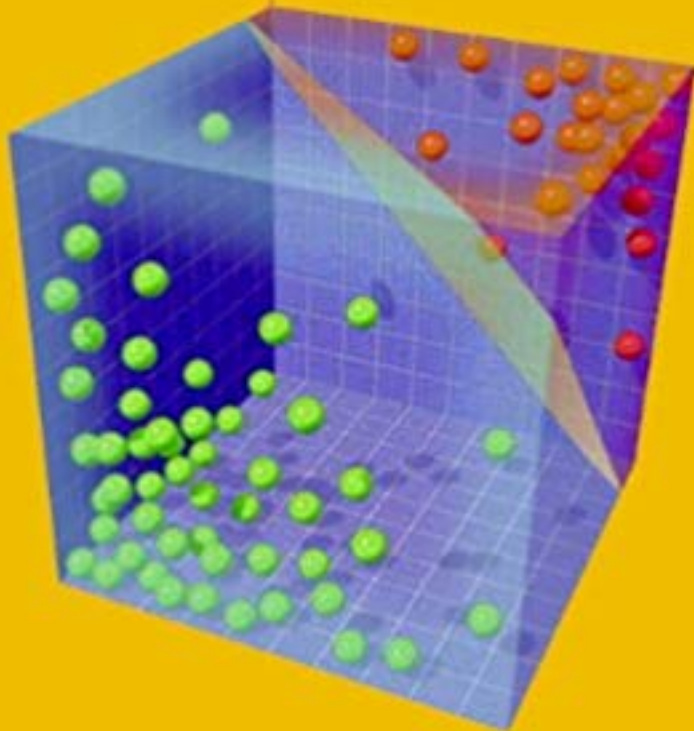
- select simple kernels, allow for violations of some of the linear constraints
by means of slack variables (e.g. kernel-version of Adatron with errors, see above)
- choose kernel (kernel parameters) by means of cross-validation procedures
- use approximate schemes for huge amounts of data (many support vectors)

Elisio Cristóbal
John Shawe-Taylor

An Introduction to

Support Vector Machines

and other kernel-based learning methods



Learning with Kernels

Support Vector Machines, Regularization,
Optimization and Beyond

Bernhard Schölkopf and Alexander Smola

