The Perceptron

Volker Tresp

Summer 2014

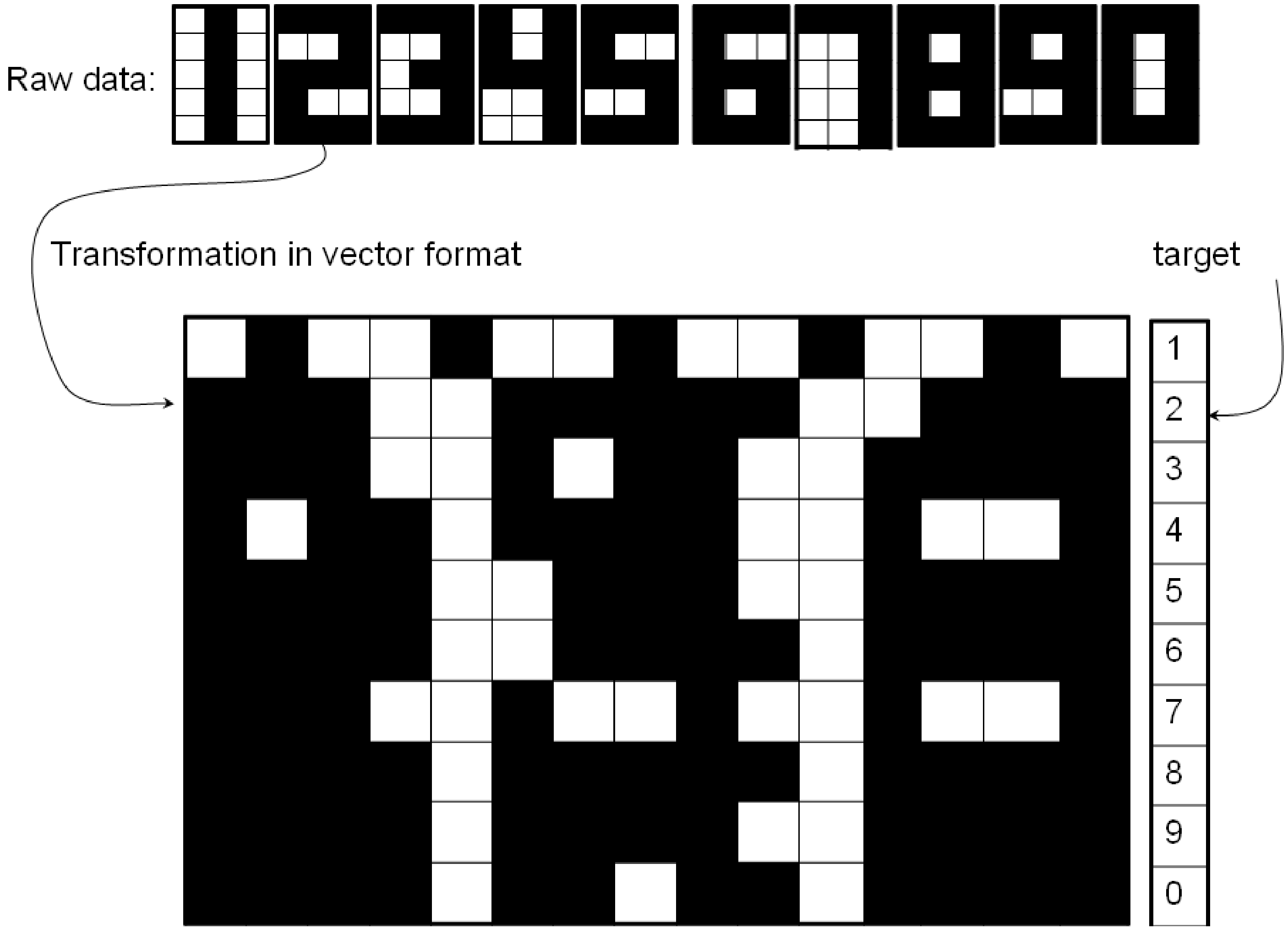
Introduction

- One of the first serious learning machines
- Most important elements in learning tasks
  - Collection and preprocessing of training data
  - Definition of a class of learning models. Often defined by the free parameters in a learning model with a fixed structure (e.g., a Perceptron)
  - Selection of a cost function
  - Learning rule to find the best model in the class of learning models. Often this means the learning of the optimal parameters

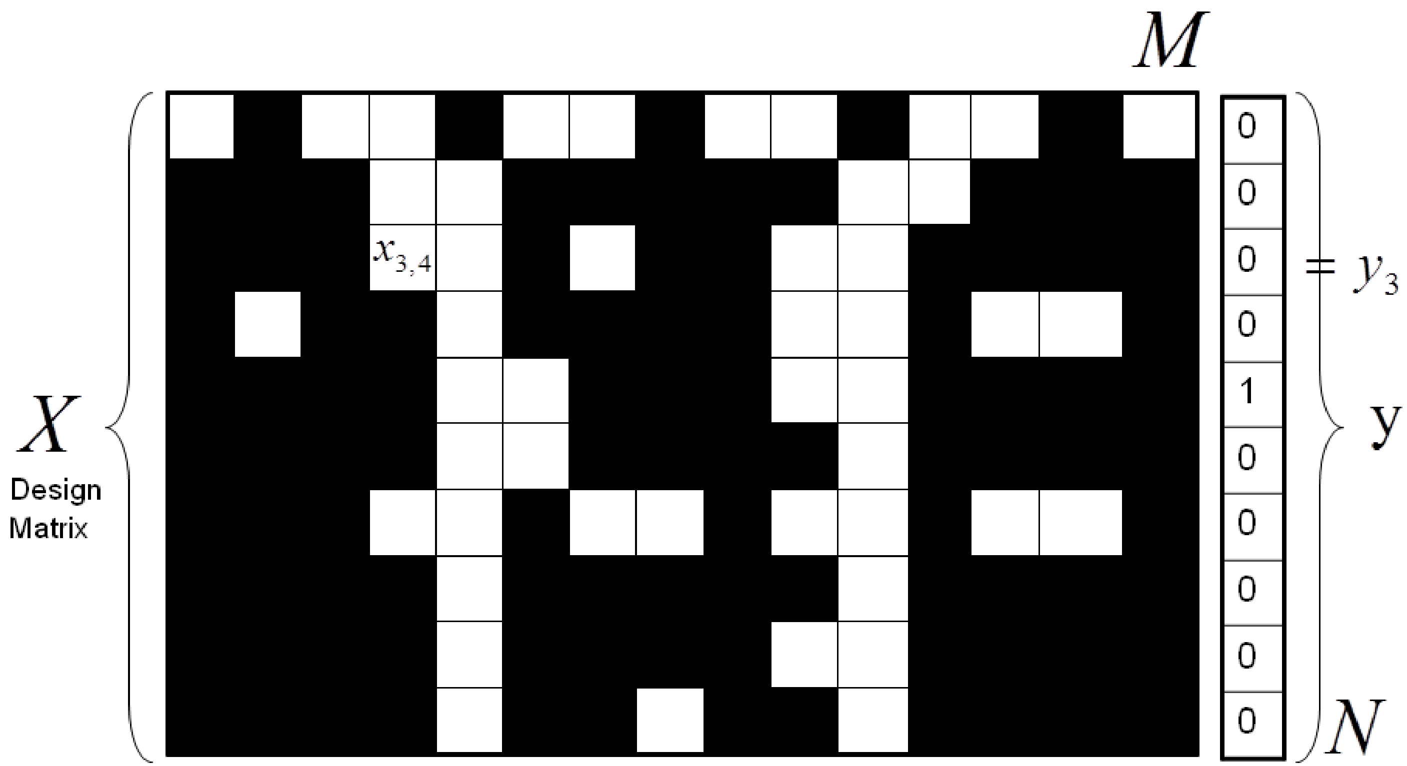
Prototypical Learning Task

- Classification of printed or handwritten digits
- Application: automatic reading of Zip codes
- More general: OCR (*optical character recognition*)

Transformation of the Raw Data (2-D) into Pattern Vectors (1-D) as part of a Learning Matrix



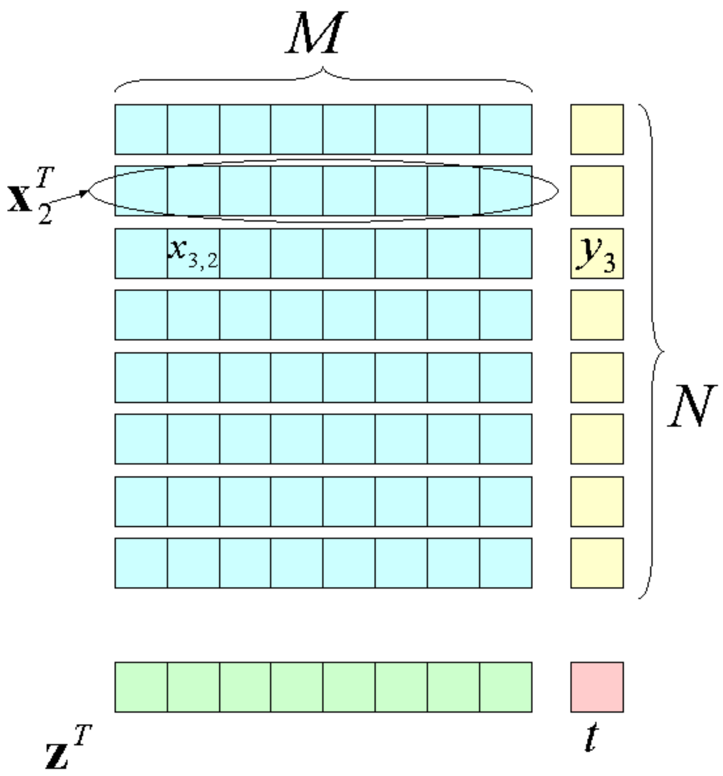
Binary Classification



target - 1: pattern belongs to class 5

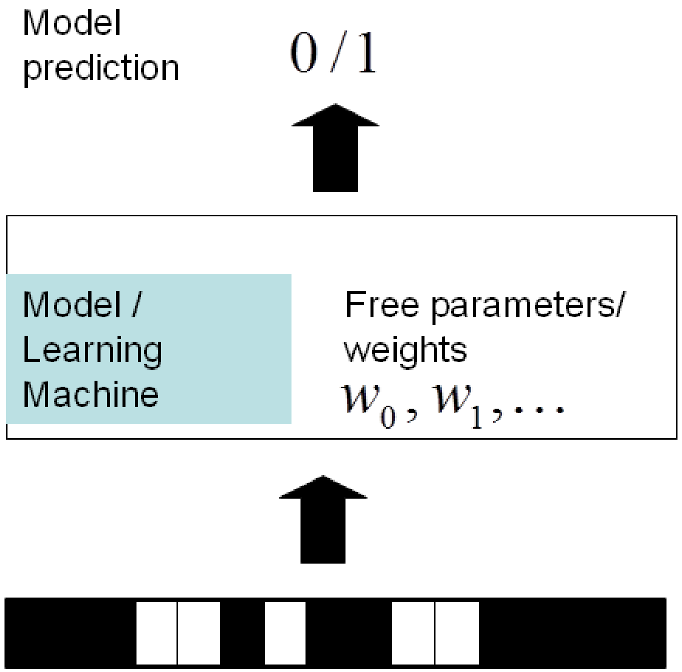
target - 0: pattern does not belong to class 5

Data Matrix for Supervised Learning



M	number of inputs
N	number of training patterns
$\mathbf{x}_i$	$= (x_{i,0}, \ldots, x_{i,M-1})^T$ i-th input
$x_{i,j}$	j-th component of $\mathbf{x}_i$
$\mathbf{X}$	$= (x_1, \ldots, x_N)^T$ design matrix
y_i	i-th target for $\mathbf{x}_i$
$\mathbf{y}$	$= (y_1, \ldots, y_N)^T$ Vector of targets
$\hat{y}_i$	prediction for $\mathbf{x}_i$
$\mathbf{d}_i$	$= (x_{i,0}, \ldots, x_{i,M-1}, y_i)^T$ i-th pattern
D	$= \{\mathbf{d}_1, \ldots, \mathbf{d}_N\}$ training data
$\mathbf{z}$	test input
t	unknown target for $\mathbf{z}$

Model



Adaptation of

$$w_0, w_1, \ldots$$

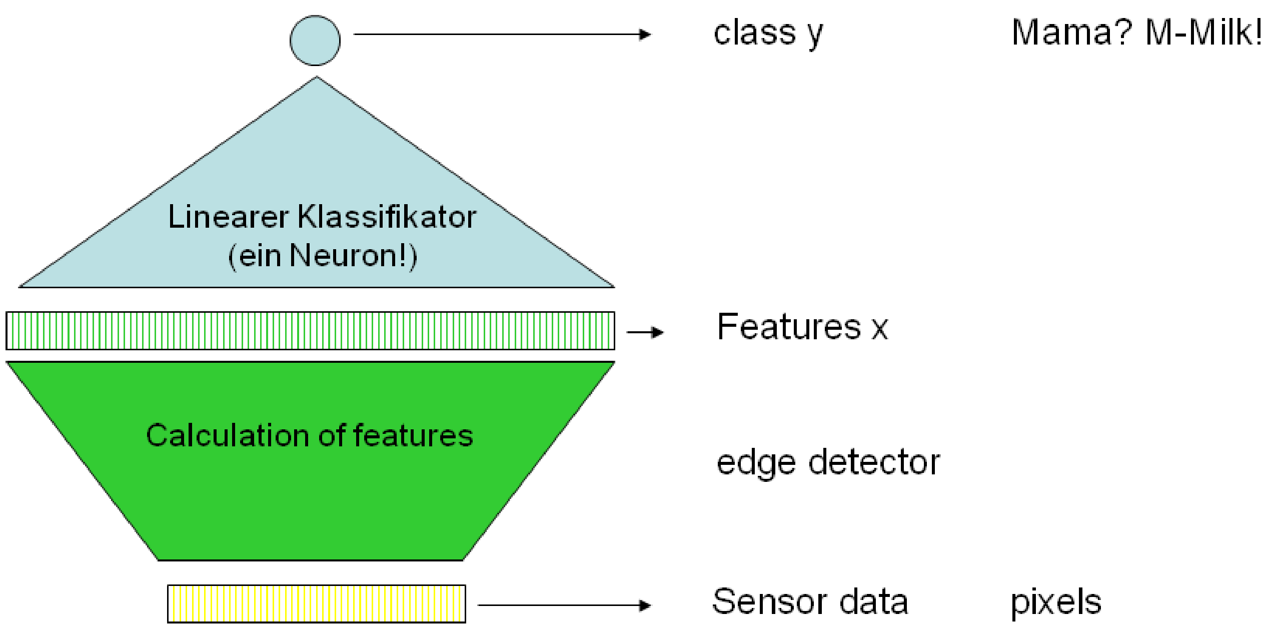
for good performance on training data

$$\mathrm{d}_i = (\mathrm{x}_i, y_i)$$

Real goal: good performance on test data

$$(\mathrm{z}, t_i)$$

A Biologically Motivated Model



Input-Output Models

- A biological system needs to make a decision, based on available senor information
- An OCR system classifies a hand written digit
- A prognostic system predicts tomorrow's energy consumption

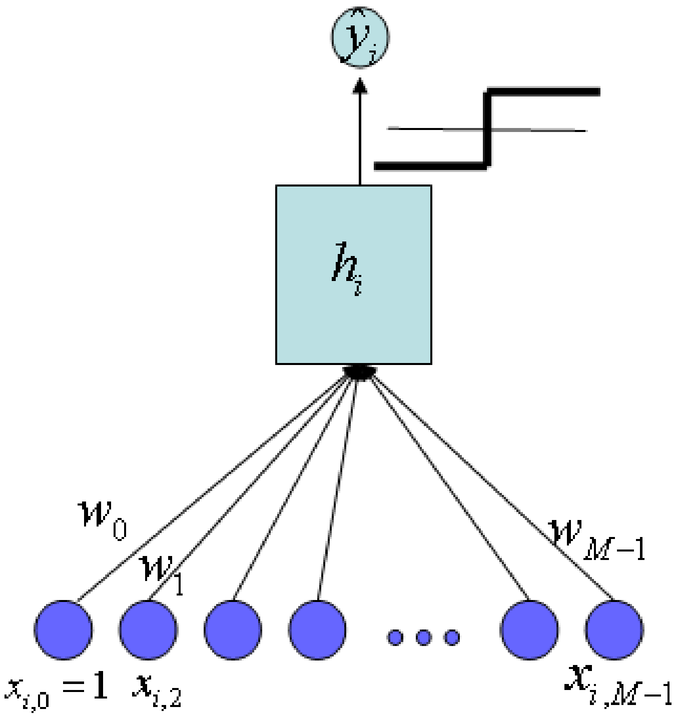
Supervised Learning

- In supervised learning one assumes that in training both inputs and outputs are available
- For example, an input pattern might reflect the attributes of an object and the target is the class membership of this object
- The goal is the correct classification for new patterns
- Linear classifier: one of the simplest but surprisingly powerful classifiers
- A linear classifier is particularly suitable, when the number of inputs M is large; if this is not the case, one can transform the input data into a high-dimensional space, where a linear classifier might be able to solve the problem; this idea is central to a large portion of the lecture (basis functions, neural networks, kernel models)
- A linear classifier can be realized through a Perceptron, a single formalized neuron!

Supervised Learning and Learning of Decisions

- One might argue that learning is only of interest if it changes (future) behavior; at least for a biological system
- Many decisions can be reduced to a supervised learning problem: if I can read a Zip code correctly, I know where the letter should be sent
- Decision tasks can often be reduced to an intermediate supervised learning problem
- But who produces the targets for the intermediate task? For biological systems a hotly debated issue: is supervised learning biologically relevant? Is only reinforcement learning, based on rewards and punishment, biologically plausible?

The Perceptron: A Learning Machine



- The activation function of the Perceptron is weighted sum of inputs

$$h_i = \sum_{j=0}^{M-1} w_j x_{i,j}$$

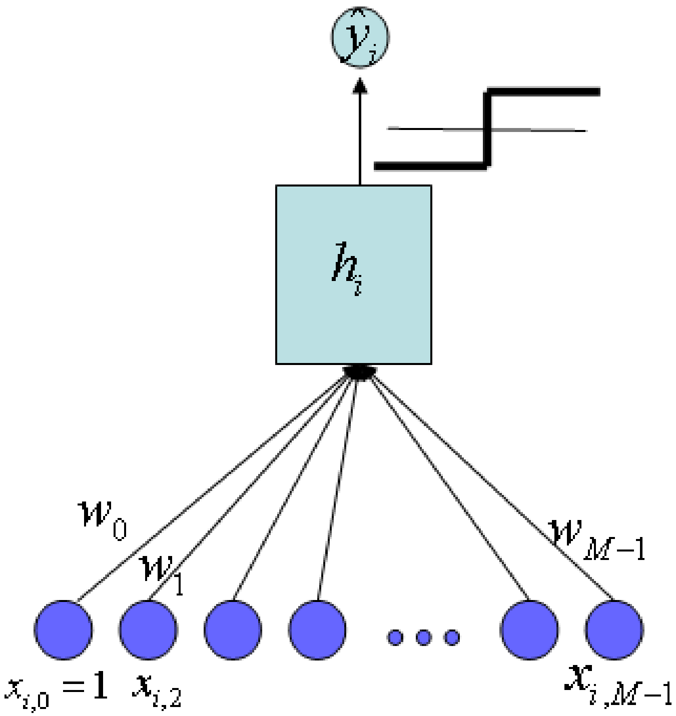
(Note: $x_{i,0} = 1$ is a constant input, such that w_0 can be though of as a bias

- The binary classification $y_i \in \{1, -1\}$ is calculated as

$$\hat{y}_i = \text{sign}(h_i)$$

- The linear classification boundary (*separating hyperplane*) is defined as $h_i = 0$

Perceptron as a Weighted Voting machine



- The Perceptron is often displayed as a graphical model with one input node for each input variable and with one output node for the target
- The bias w_0 determines the class when all inputs are zero
- When $x_{i,j} = 1$ the j-th input votes with weight $|w_j|$ for class $\text{sign}(w_j)$
- *Thus, the response of the Perceptron can be thought of as a weighted voting for a class.*

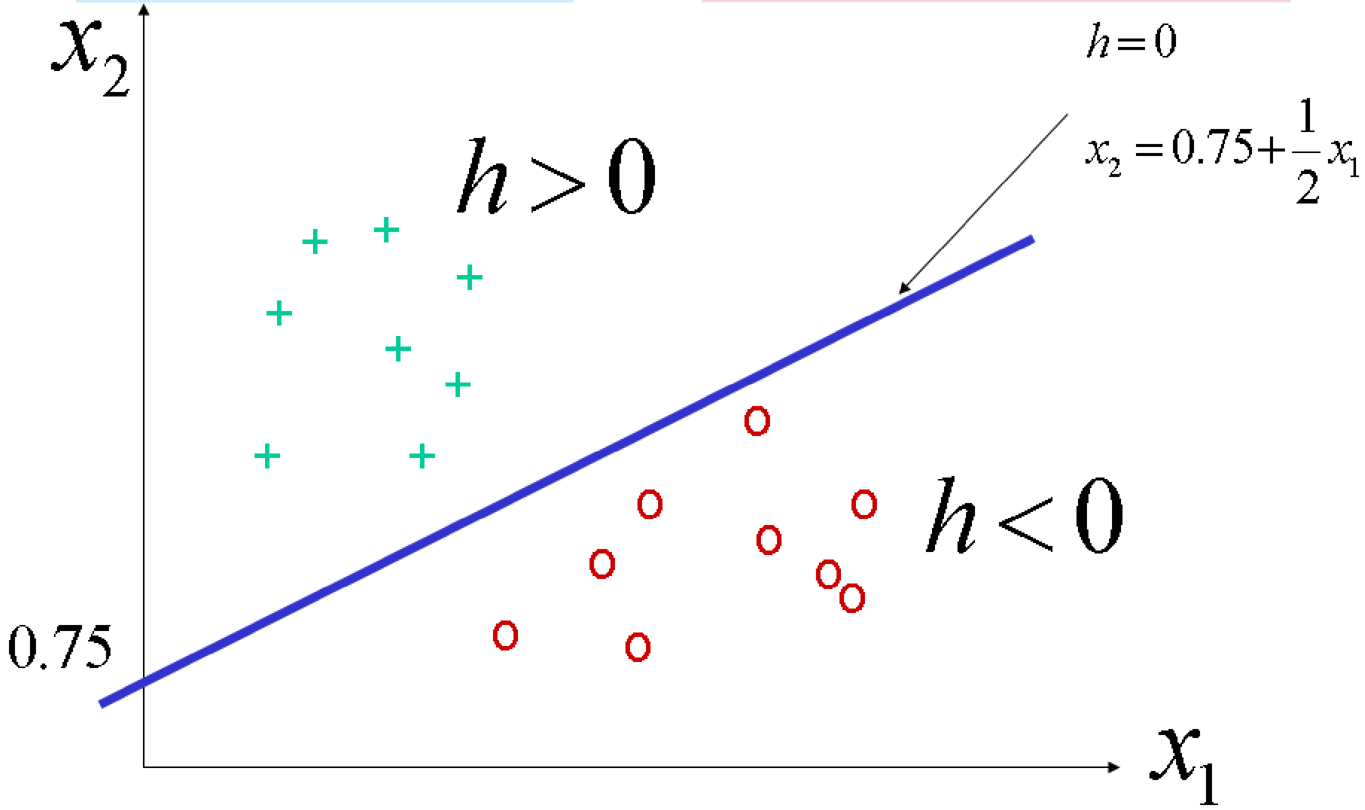
2-D Representation of the Decision Boundary

- The class boundaries are often displayed graphically with $M = 3$ (next slide)
- This provides some intuition
- But note, that this 2-D picture can be misleading, since the Perceptron is typically employed in high-dimensional problems ($M >> 1$)

Two classes that are Linearly Separable

$$h = -0.75 - \frac{1}{2}x_1 + x_2$$

$$h = 0 \Rightarrow x_2 = 0.75 + \frac{1}{2}x_1$$



Perceptron Learning Rule

- We now need a learning rule to find optimal parameters $w_0, \ldots, w_{M-1}$
- We define a cost function that is dependent on the training data and the parameters
- In the learning process (training), one attempts to find parameters that minimize the cost function

The Perceptron Cost Function

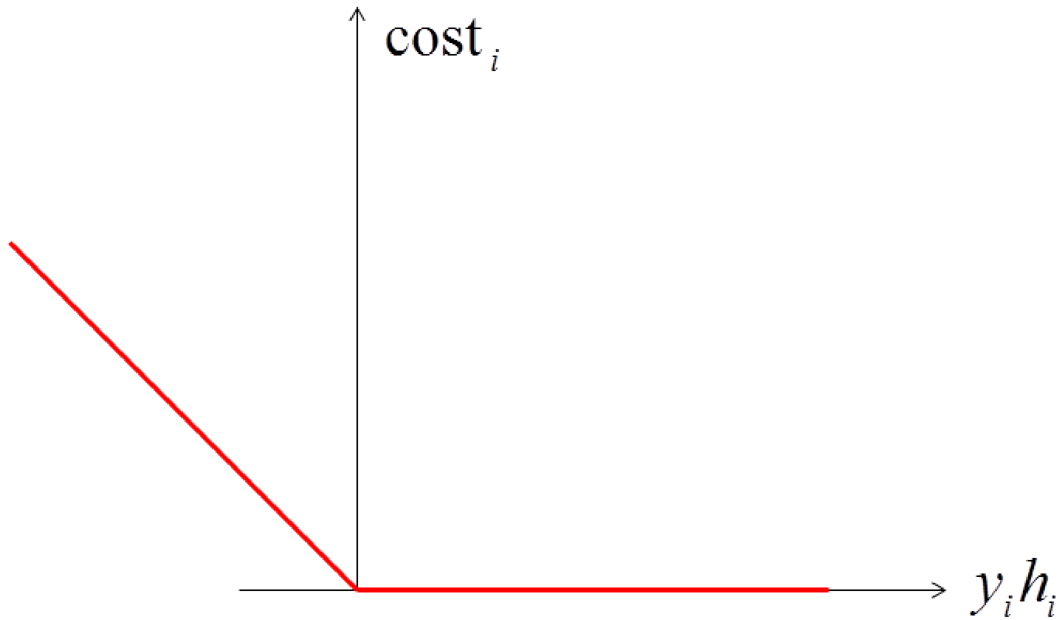
- Goal: correct classification of the N training samples $\{y_1, \ldots, y_N\}$
- The Perceptron cost function is

$$\text{cost} = -\sum_{i \in \mathcal{M}} y_i h_i = \sum_{i=1}^{N} |-y_i h_i|_+$$

where $\mathcal{M} \subseteq \{1, \ldots, N\}$ is the index set of the currently misclassified patterns and $x_{i,j}$ is the value of the j-th input in the i-th pattern. $|arg|_+ = \max(arg, 0)$.

- Obviously, we get $\text{cost} = 0$ only, when all patterns are correctly classified (then $\mathcal{M} \subseteq \emptyset$); otherwise $\text{cost} > 0$, since y_i and h_i have different signs for misclassified patterns

Contribution to the Cost Function of one Data Point



Gradient Descent

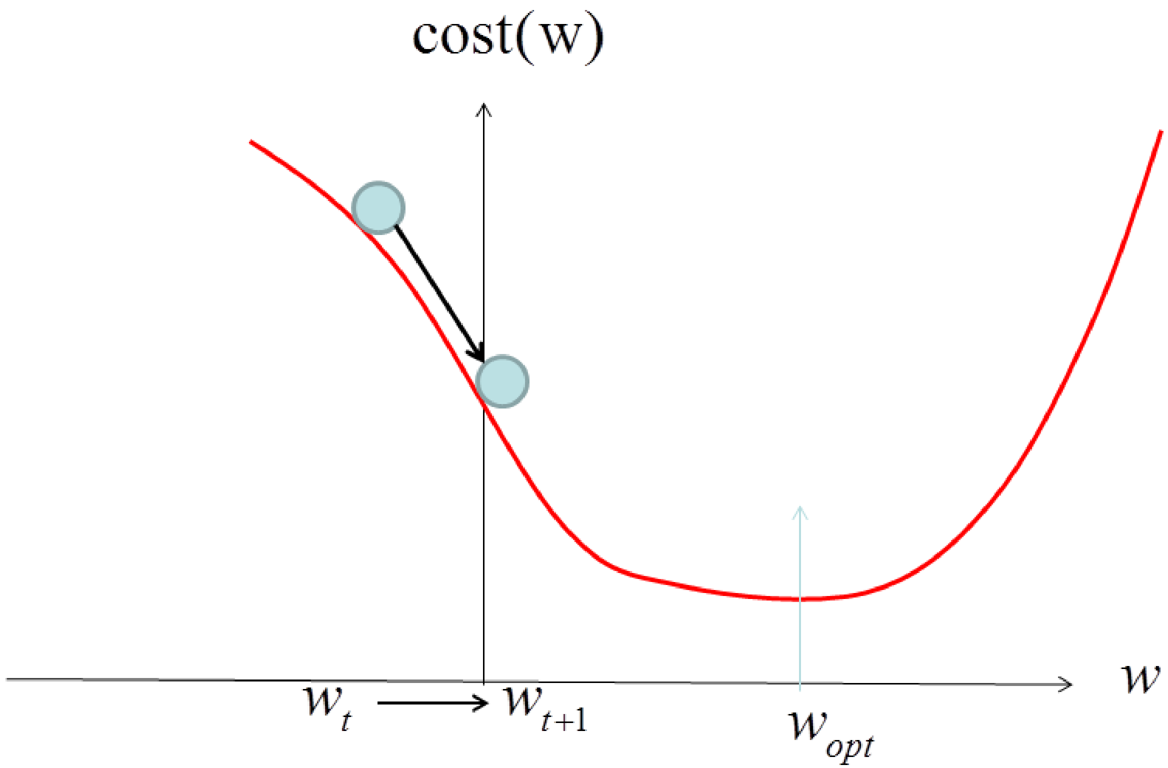
- Initialize parameters (typically small random values)
- In each learning step, change the parameters such that the cost function decreases
- Gradient decent: adapt the parameters in the direction of the negative gradient
- The partial derivative of the weights with respect to the parameters is (Example: w_j)

$$\frac{\partial \text{cost}}{\partial w_j} = - \sum_{i \in \mathcal{M}} y_i x_{i,j}$$

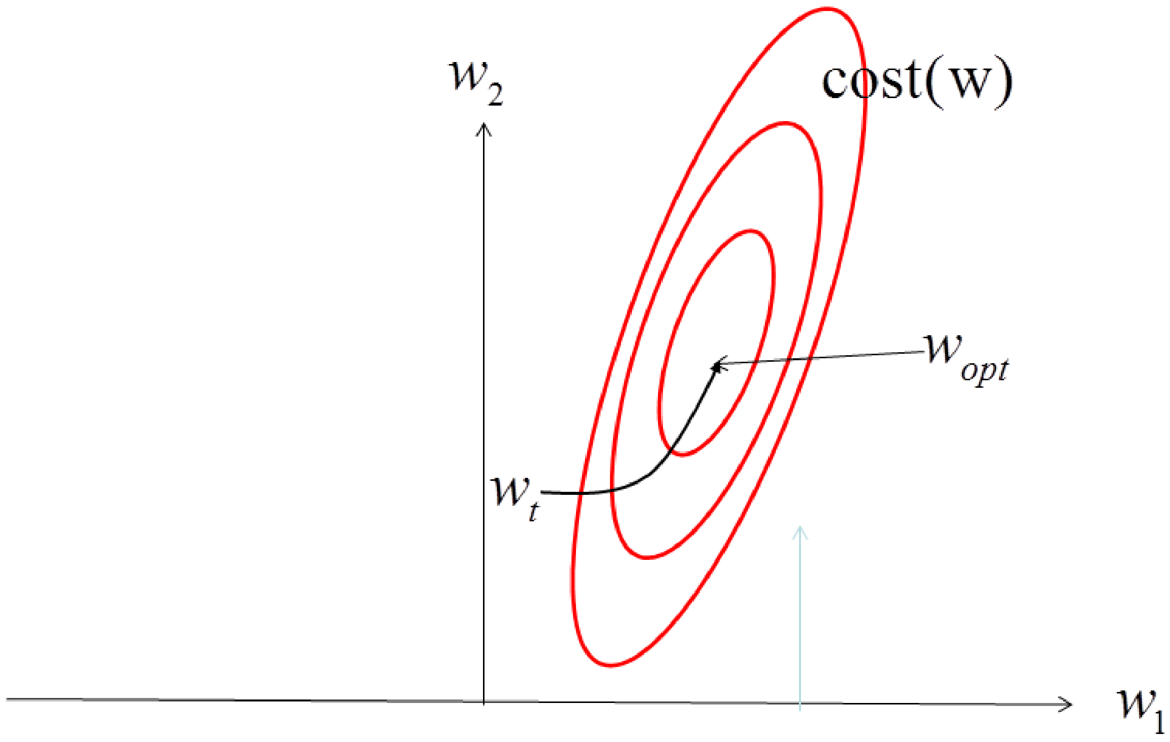
- Thus, a sensible adaptation rule is

$$w_j \longleftarrow w_j + \eta \sum_{i \in \mathcal{M}} y_i \mathbf{x}_{i,j}$$

Gradient Descent with One Parameter (Conceptual)



Gradient Descent with Two Parameters (Conceptual)



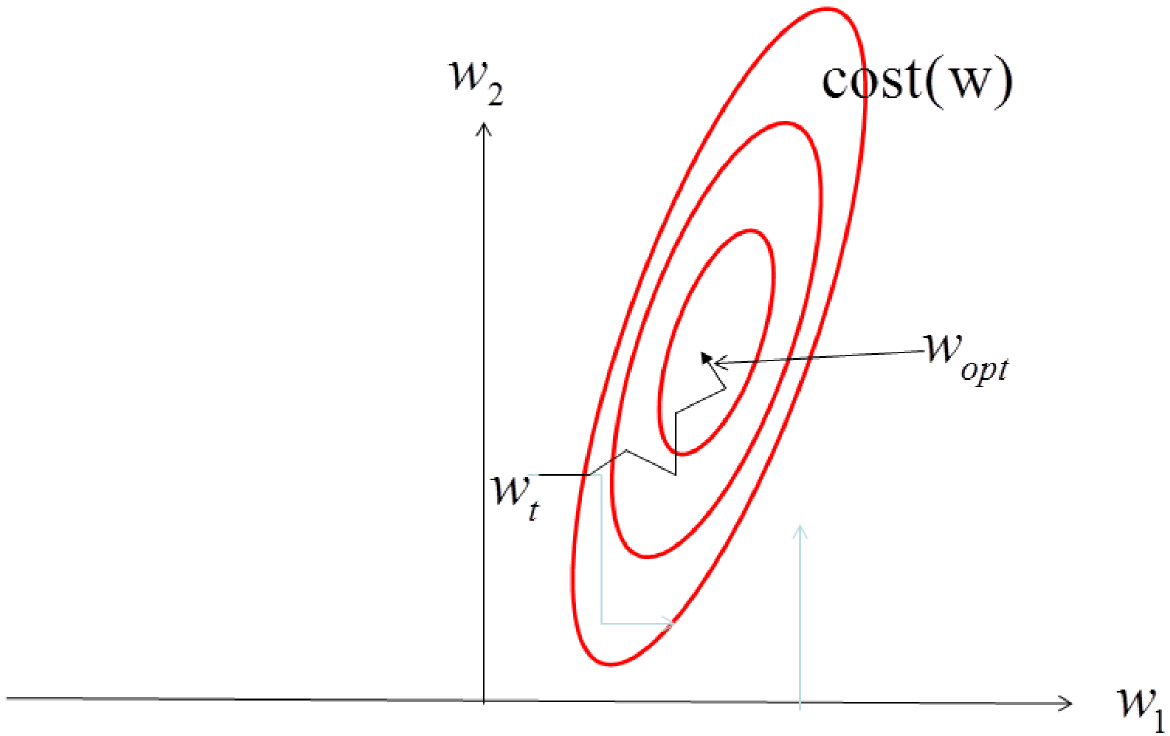
The Perceptron-Learning Rule

- In the actual Perceptron learning rule, one presents randomly selected currently misclassified patterns and adapts with only that pattern. This is biologically more plausible and also leads to faster convergence. Let $\mathbf{x}_t$ and y_t be the training pattern in the t-th step. One adapts $t = 1, 2, \ldots$

$$w_j \longleftarrow w_j + \eta y_t x_{t,j} \quad j = 1, \ldots, M$$

- A weight increases, when (postsynaptic) $y(t)$ and (presynaptic) $x_j(t)$ have the same sign; different signs lead to a weight decrease (compare: **Hebb Learning**)
- $\eta > 0$ is the learning rate, typically $0 < \eta << 1$
- Pattern-based learning is also called stochastic gradient descent (SGD)

Stochastic Gradient Descent (Conceptual)



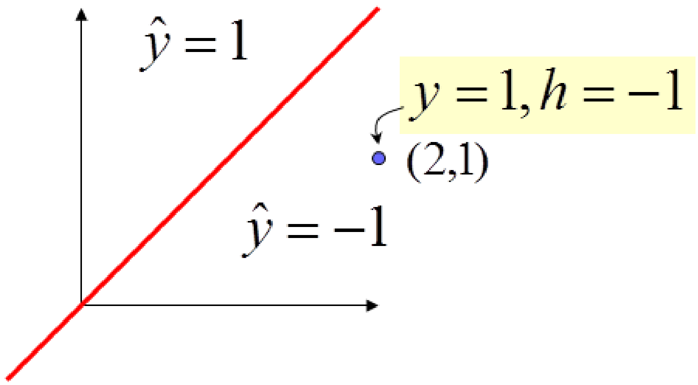
Comments

- Convergence proof: with sufficiently small learning rate η and when the problem is linearly separable, the algorithm converges and terminates after a finite number of steps

- If classes are not linearly separable and with finite η there is no convergence

Example: Perceptron Learning Rule, $\eta = 0.1$

$$h = 0 \times 1 - 1 \times x_1 + 1 \times x_2$$



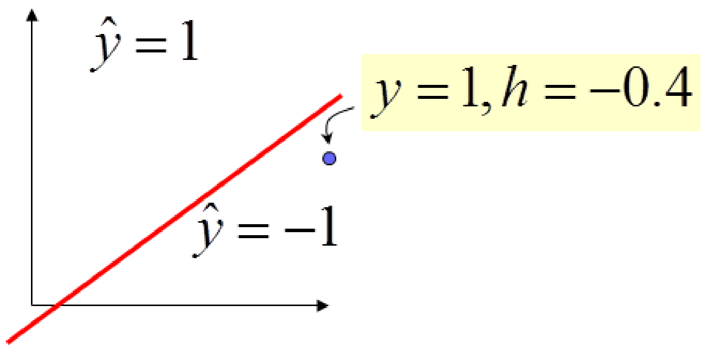
Separation plane prior to adaptation step: $x_2 = x_1$

Adaptation:

$$w_0 \leftarrow 0 + 0.1 \times (1 \times 1) = 0.1$$
$$w_1 \leftarrow -1 + 0.1 \times (1 \times 2) = -0.8$$
$$w_2 \leftarrow 1 + 0.1 \times (1 \times 1) = 1.1$$

"Hebb": all parameters grow

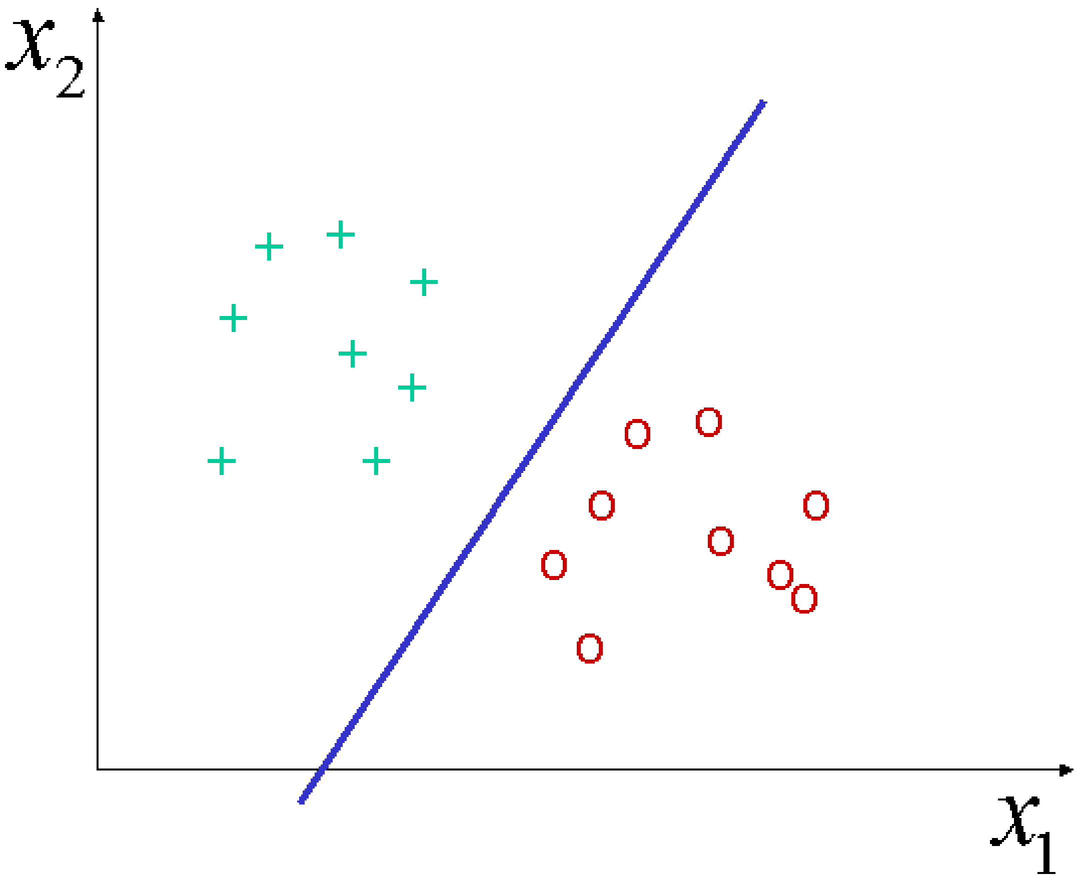
$$h = 0.1 \times 1 - 0.8 \times x_1 + 1.1 \times x_2$$



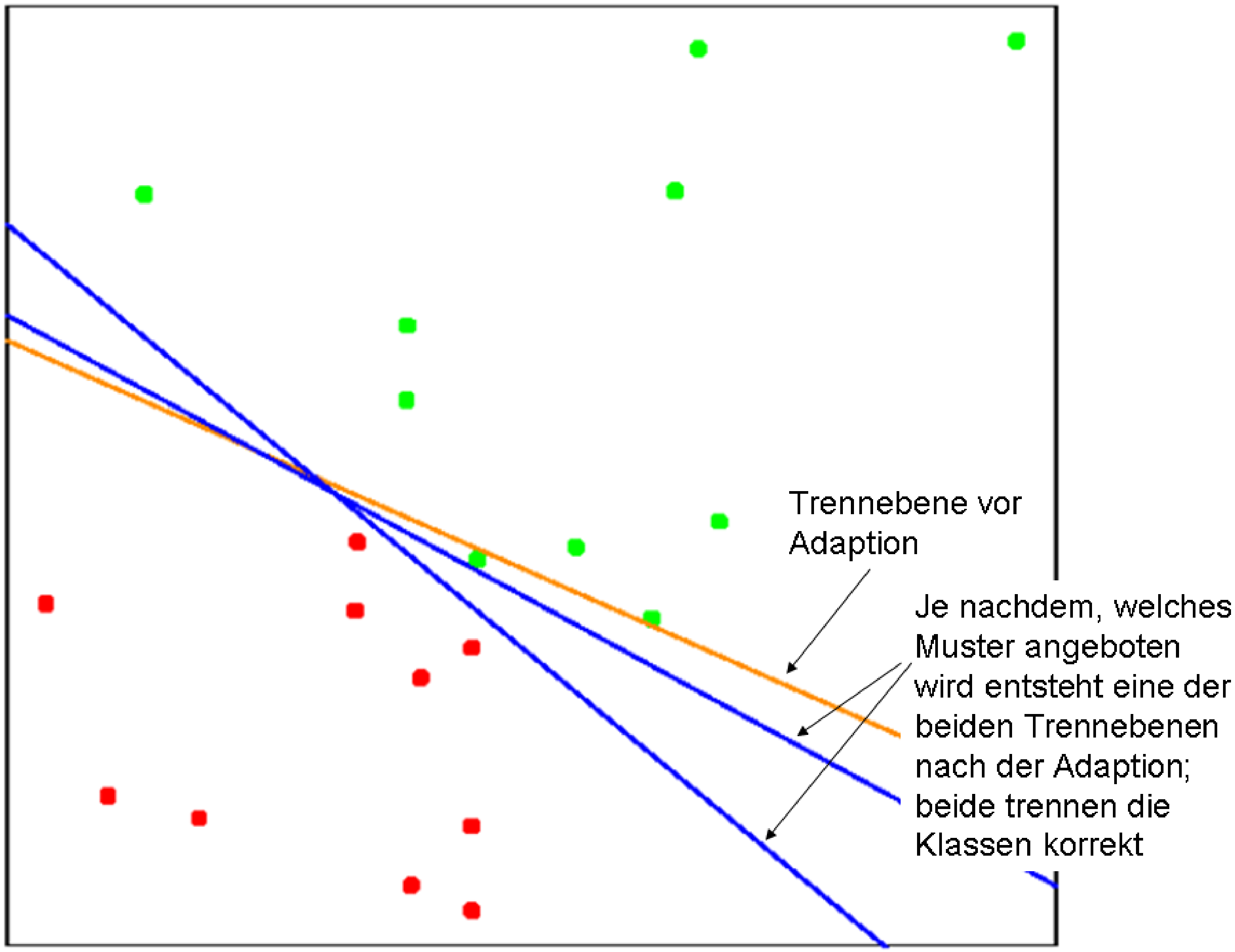
Separating plane after adaptation:

$$x_2 = -0.09 + 0.72 \times x_1$$

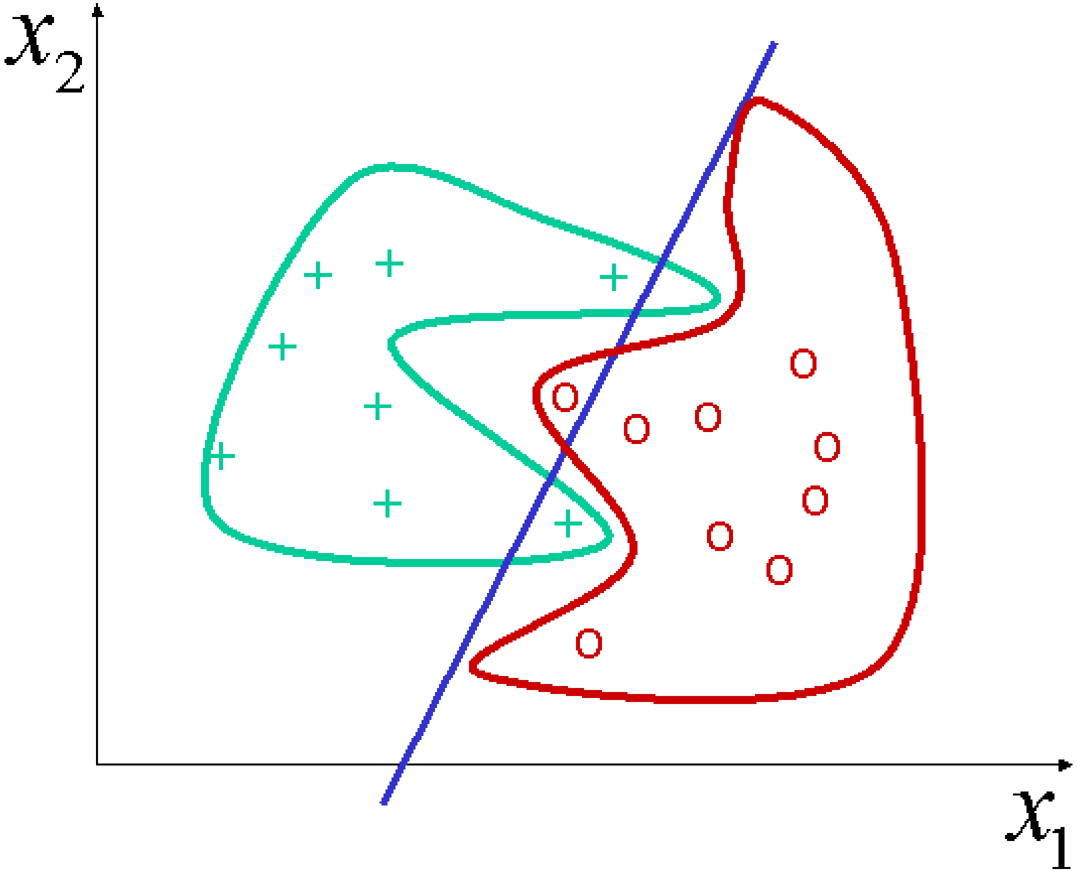
Linearly Separable Classes



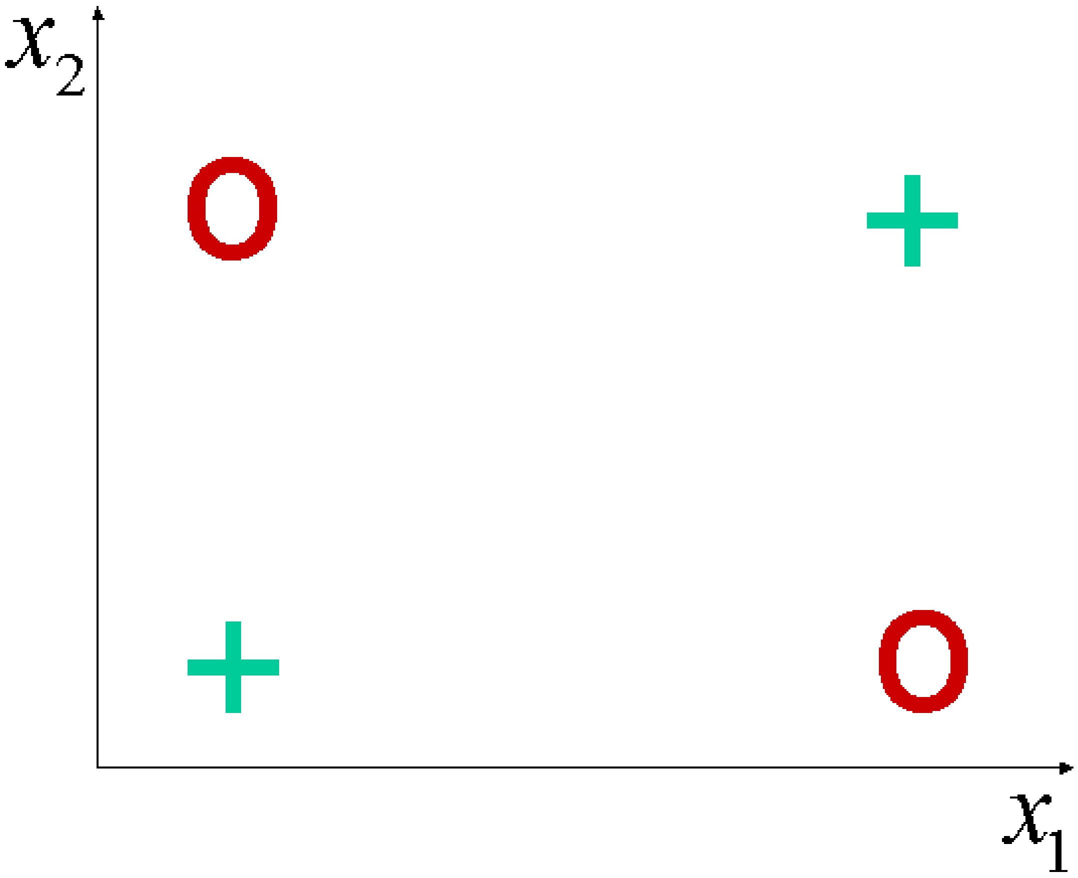
Convergence and Degenerativity



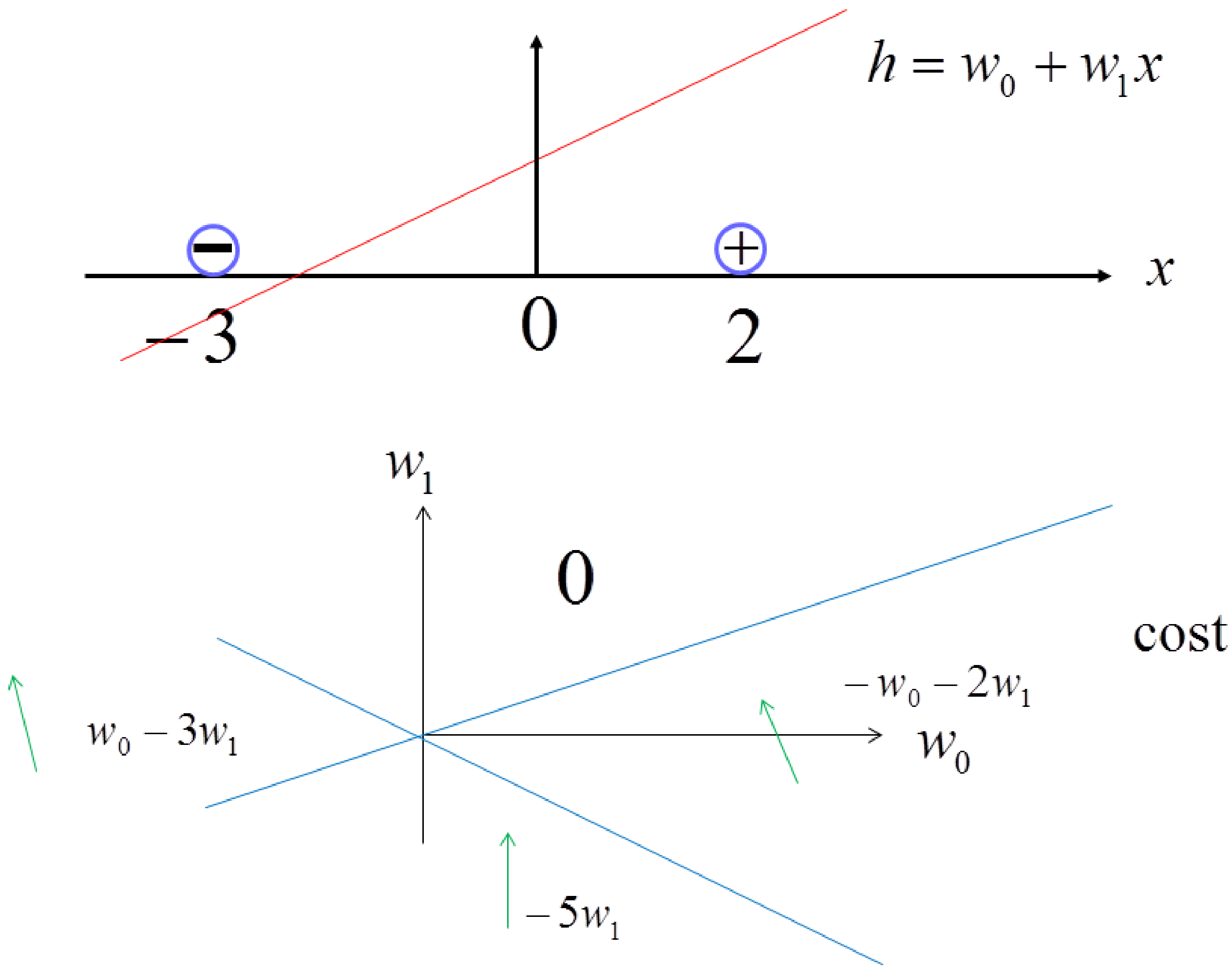
Classes that Cannot be Separated with a Linear Classifier



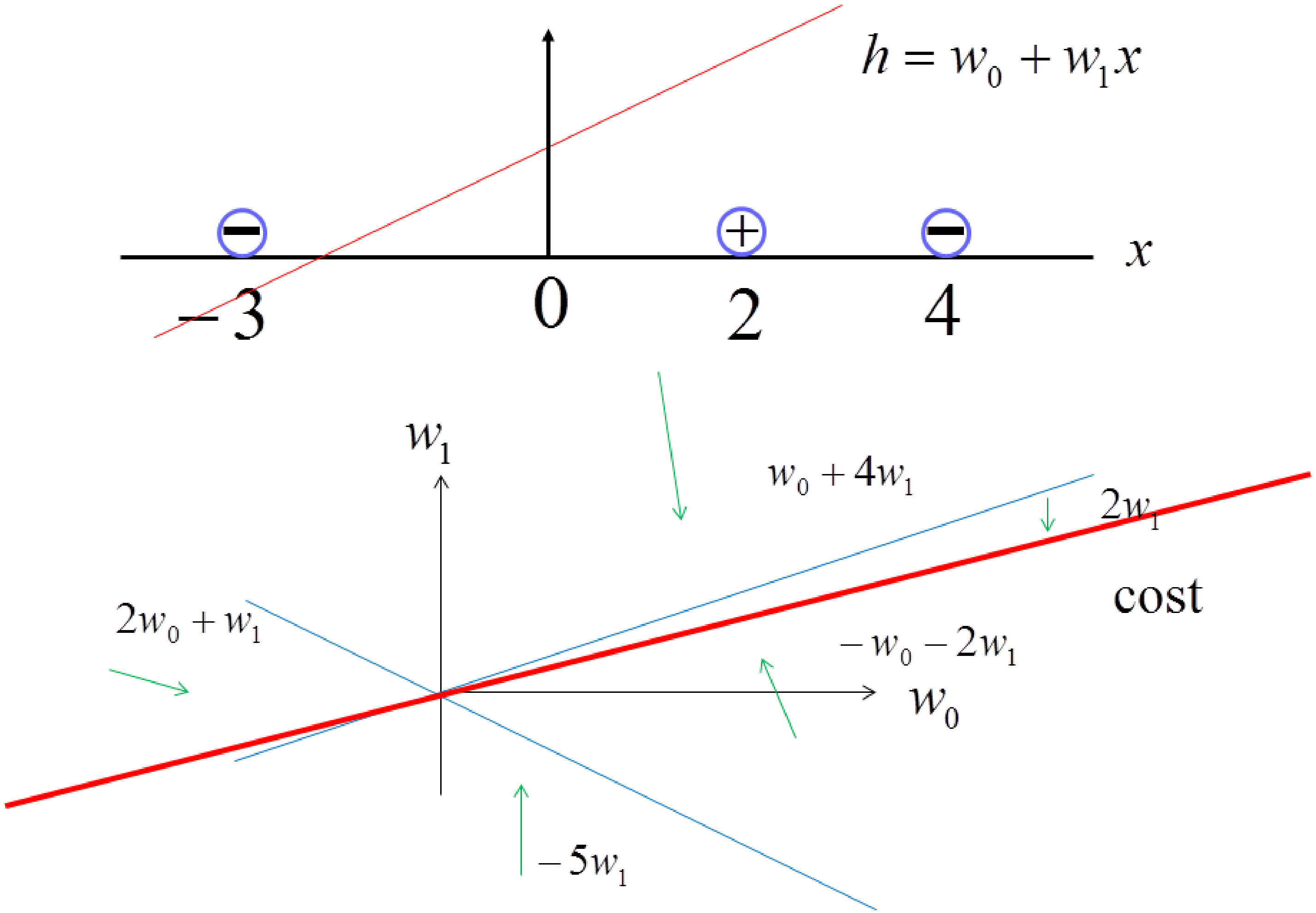
The classical Example for Linearly Non-Separable Classes: XOR



Classes are Separable (Convergence)



Classes are not Separable (no Convergence)



Comments on the Perceptron

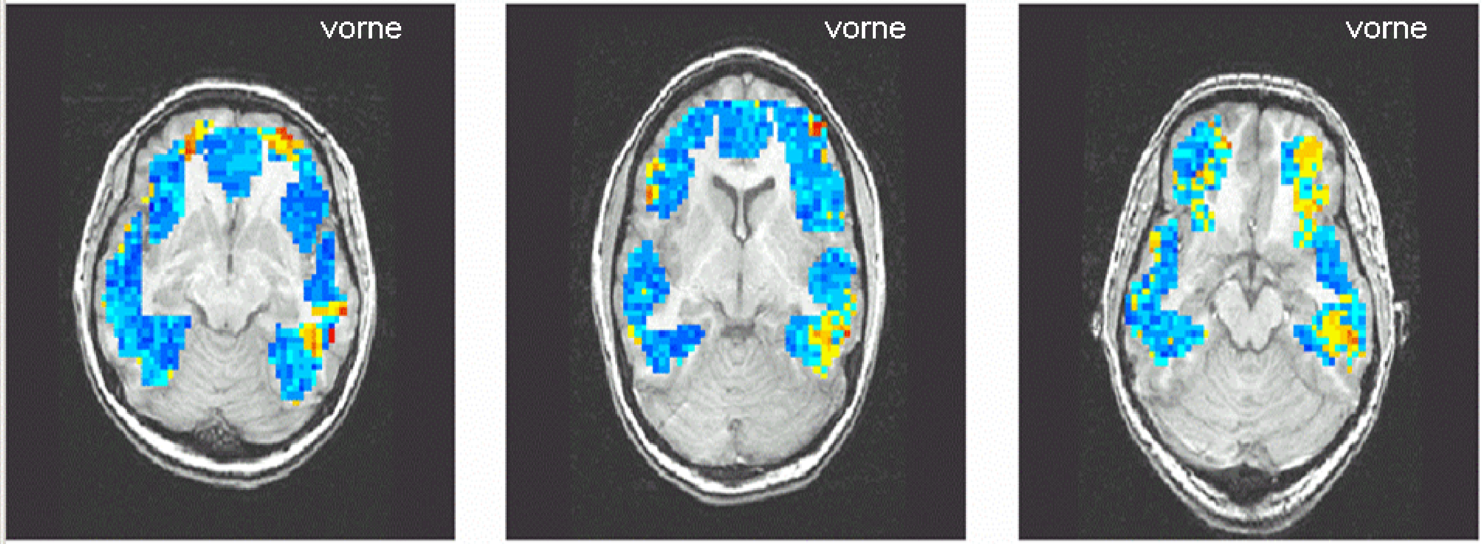
- Convergence can be very fast
- A linear classifiers is a very important basic building block: with $M \to \infty$ most problems become linearly separable!
- In some case, the data are already high-dimensional with $M > 10000$ (e.g., number of possible key words in a text)
- In other cases, one first transforms the input data into a high-dimensional (sometimes even infinite) space and applies the linear classifier in that space: kernel trick, Neural Networks
- Considering the power of a single formalized neuron: how much computational power might 100 billion neurons posses?
- Are there *grandmother cells* in the brain? Or grandmother areas?

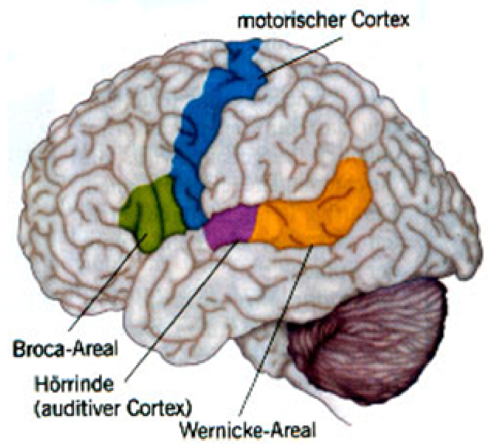
Comments on the Perceptron (cont'd)

- The Perceptron learning rule is not used much any more
  - No convergence, when classes are not separable
  - Classification boundary is not unique
- Alterbatvie learning rules:
  - Linear Support Vector Machine
  - Fisher Linear Discriminant
  - Logistic Regression

Application for a Linear Classifier; Analysis of fMRI Brain Scans (Tom Mitchel et al., CMU)

- Goal: based on the image slices determine if someone thinks of tools, buildings, food, or a large set of other semantic concepts
- The trained linear classifier is 90% correct and can. e.g., predict if someone reads about tools or buildings
- The figure shows the voxels, which are most important for the classification task. All three test persons display similar regions


vorne
vorne
vorne


motorischer Cortex
Broca-Areal
Hörrinde
(auditiver Cortex)
Wernicke-Areal

Pattern Recognition Paradigm

- von Neumann: *... the brain uses a peculiar statistical language unlike that employed in the operation of man-made computers...*
- A classification decision is done in done by considering the complete input pattern, and neither as a logical decision based on a small number of attributes nor as a complex logical programm
- The linearly weighted sum corresponds more to a voting: each input has either a positive or a negative influence on the classification decision
- Robustness: in high dimensions a single, possible incorrect, input has little influence

Afterword

Why Pattern Recognition?

- Alternative approach to pattern recognition: learning of simple close-to deterministic rules (naive expectation)
- One of the big mysteries in machine learning is why rule learning is not very successful
- Problems: the learned rules are either trivial, known, or extremely complex and very difficult to interpret
- This is in contrast to the general impression that the world is governed by simple rules
- Also: computer programs, machines ... follow simple deterministic if-then-rules?

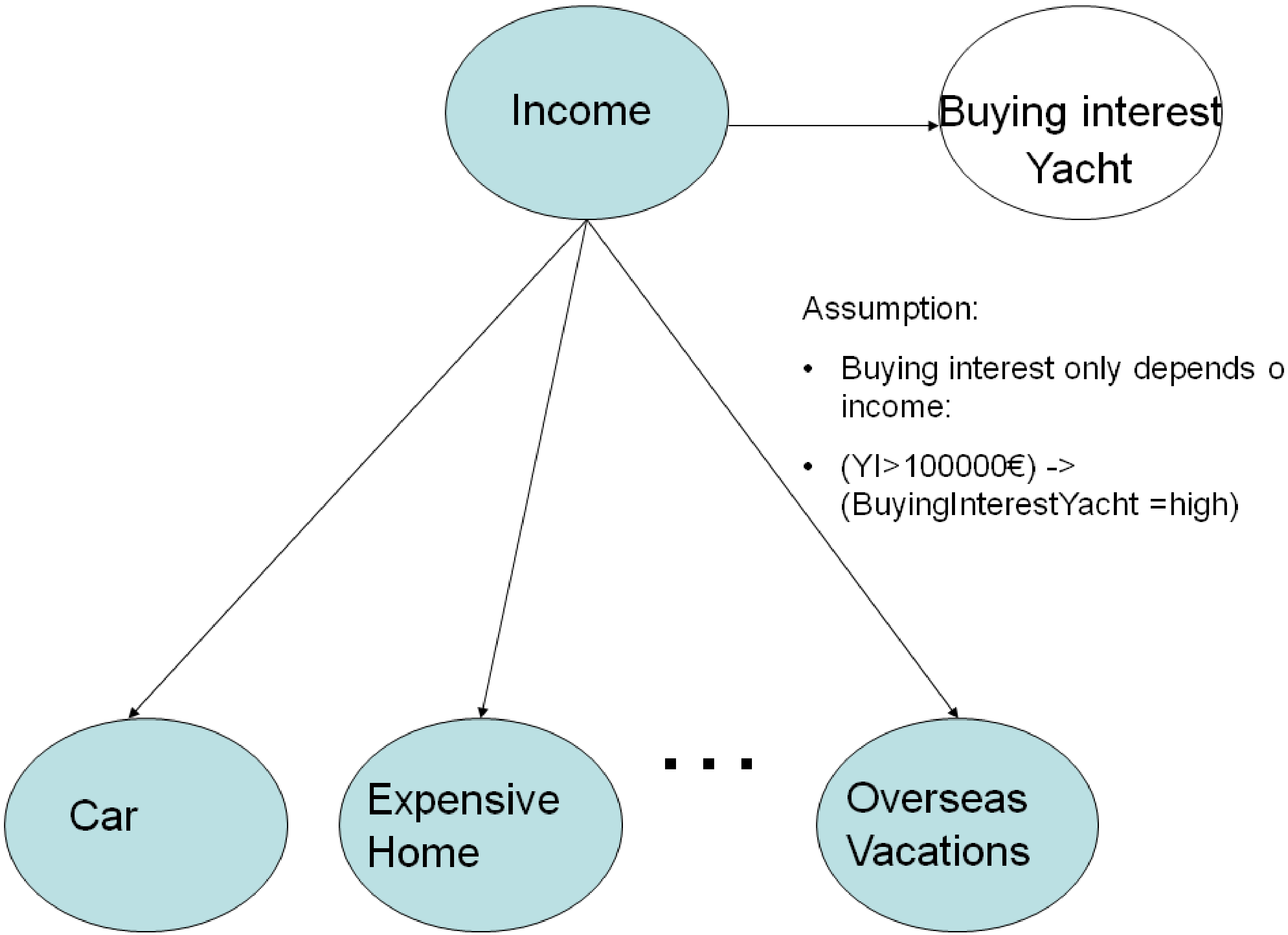
Example: Birds Fly

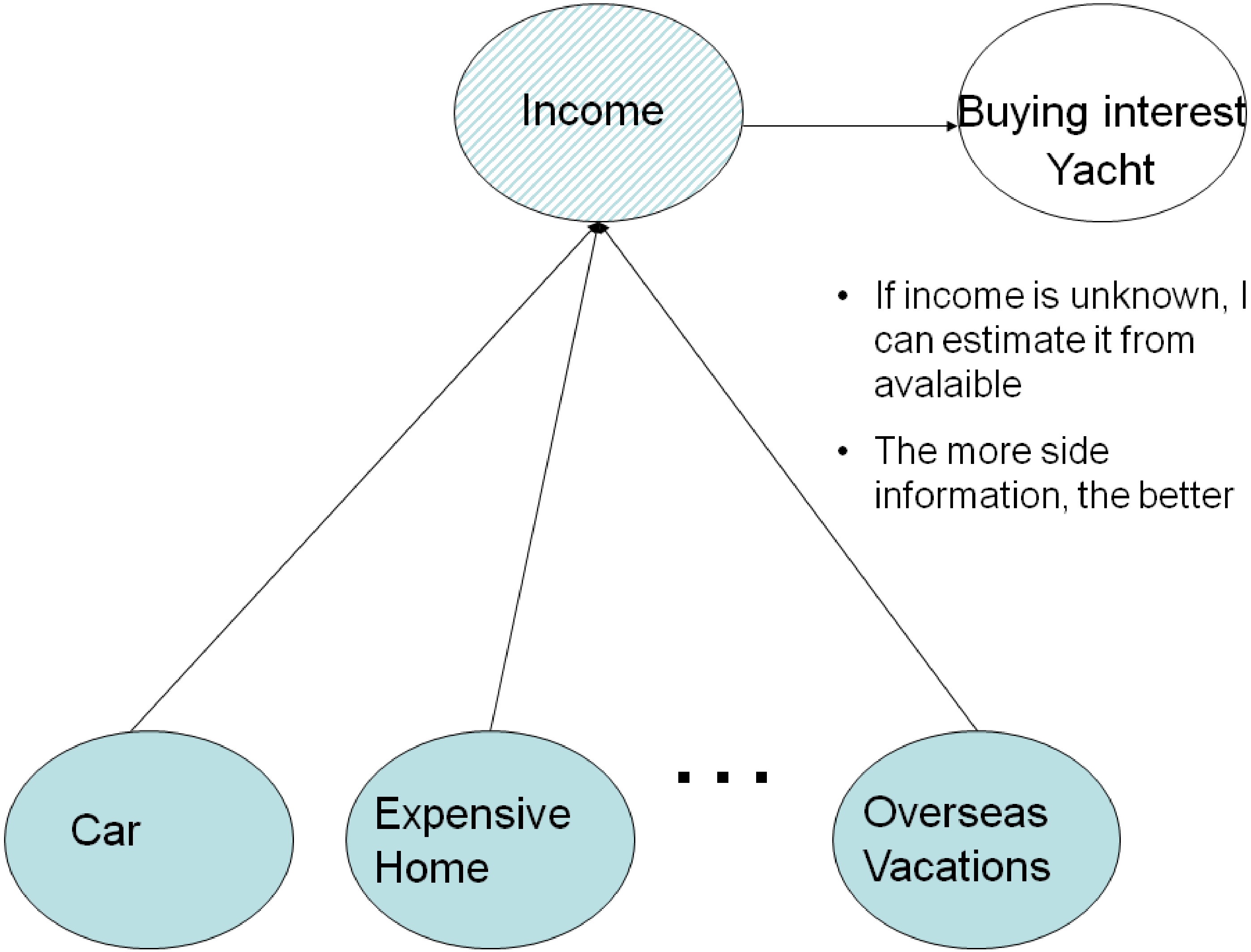
- Define flying: using its own force, at leat 20m, at leat 1m high, at least one every day in its adult life, ...
- A bird can fly if,
  - it is not a penguin, or ....
  - it is not seriously injured or dead
  - it is not too old
  - the wings have not been clipped
  - it does not have a number of diseases
  - it only lives in a stable
  - it carries heavy weights
  - ...

Pattern Recognition

- 90% of all birds fly
- Of all birds which do not belong to a flightless class 94% fly
- ... and which are not domesticated 96% ...
- Basic problem:
  - Complexity of the underlying (deterministic) system
  - Incomplete information
- Thus: success of statistical machine learning!

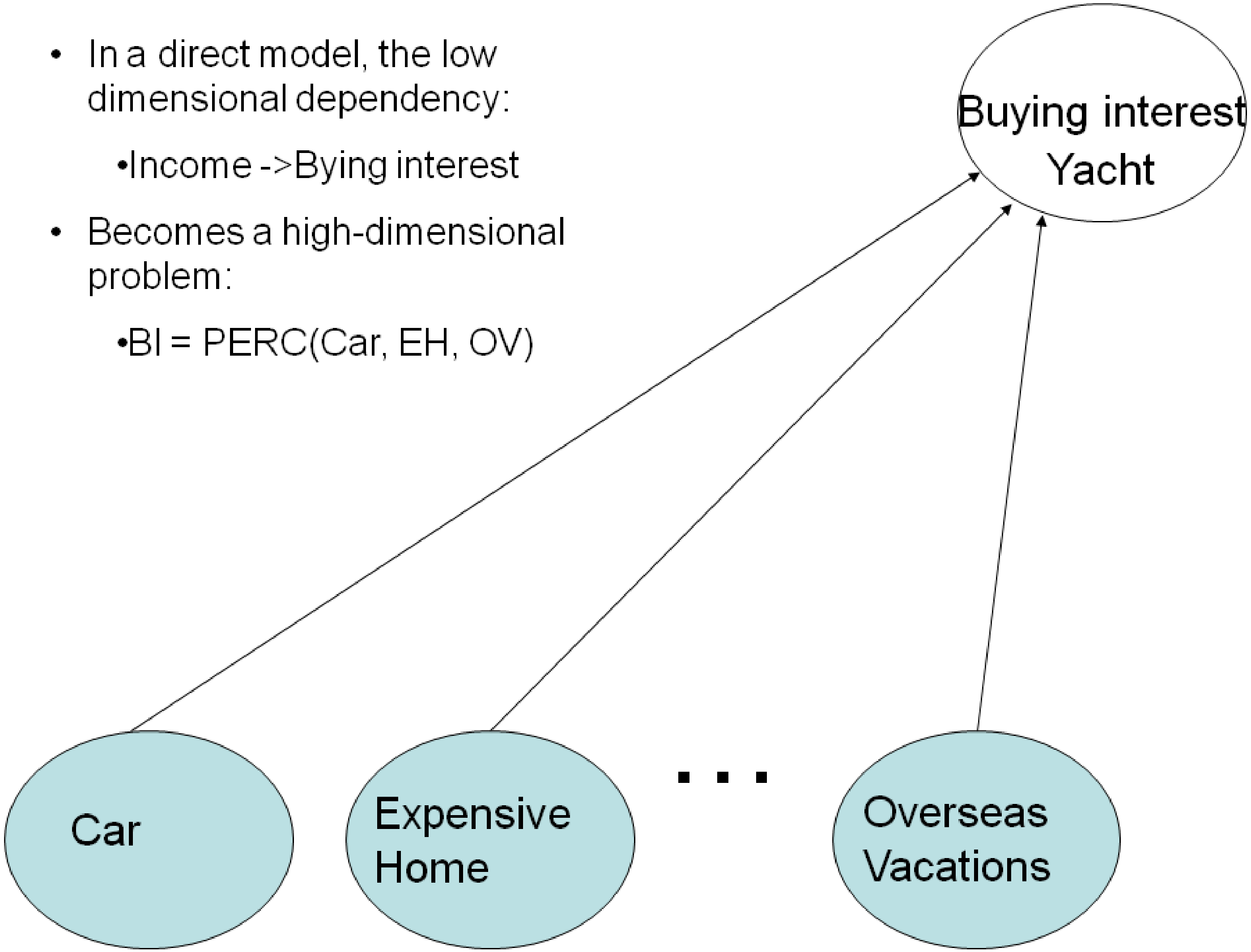
Example: Predicting Buying Pattern





- If income is unknown, I can estimate it from avalaible
- The more side information, the better

- In a direct model, the low dimensional dependency:
  - Income ->Bying interest
- Becomes a high-dimensional problem:
  - BI = PERC(Car, EH, OV)



Where Rule-Learning Works

- Technical human generated worlds ("Engine A always goes with transmission B").