Classification

Bayes Classifiers

Bayes Classifiers

Probabilistic Classification and Bayes' Rule

Naive Bayes Classifiers

- Derivation of the classification formula
- Probability estimation and Laplace correction
- Simple examples of naive Bayes classifiers
- A naive Bayes classifier for the Iris data

Full Bayes Classifiers

- Derivation of the classification formula
- Comparison to naive Bayes classifiers
- A simple example of a full Bayes classifier
- A full Bayes classifier for the Iris data

Summary

Probabilistic Classification

A classifier is an algorithm that assigns a class from a predefined set to a case or object, based on the values of descriptive attributes.

An optimal classifier maximizes the probability of a correct class assignment.

Let C be a class attribute with $\mathrm{dom}(C) = \{c_1, \ldots, c_{n_C}\}$,
which occur with probabilities p_i, $1 \leq i \leq n_C$.

Let q_i be the probability with which a classifier assigns class c_i.
($q_i \in \{0, 1\}$ for a deterministic classifier)

The probability of a correct assignment is

$$P(\text{correct assignment}) = \sum_{i=1}^{n_C} p_i q_i.$$

Therefore the best choice for the q_i is

$$q_i = \begin{cases} 1, & \text{if } p_i = \max_{k=1}^{n_C} p_k, \\ 0, & \text{otherwise.} \end{cases}$$

Probabilistic Classification

Consequence: An optimal classifier should assign the **most probable class**.

This argument does not change if we take descriptive attributes into account.

Let $U = \{A_1, \ldots, A_m\}$ be a set of descriptive attributes
with domains $\mathrm{dom}(A_k)$, $1 \leq k \leq m$.

Let $A_1 = a_1, \ldots, A_m = a_m$ be an instantiation of the descriptive attributes.

An optimal classifier should assign the class c_i for which

$$P(C = c_i \mid A_1 = a_1, \ldots, A_m = a_m) =$$
$$\max_{j=1}^{n_C} P(C = c_j \mid A_1 = a_1, \ldots, A_m = a_m)$$

Problem: We cannot store a class (or the class probabilities) for every possible instantiation $A_1 = a_1, \ldots, A_m = a_m$ of the descriptive attributes. (The table size grows exponentially with the number of attributes.)

Therefore: **Simplifying assumptions are necessary.**

Bayes' Rule and Bayes' Classifiers

Bayes' rule is a formula that can be used to "invert" conditional probabilities: Let X and Y be events, $P(X) > 0$. Then

$$P(Y \mid X) = \frac{P(X \mid Y) \cdot P(Y)}{P(X)}.$$

Bayes' rule follows directly from the definition of conditional probability:

$$P(Y \mid X) = \frac{P(X \cap Y)}{P(X)} \qquad \text{and} \qquad P(X \mid Y) = \frac{P(X \cap Y)}{P(Y)}.$$

Bayes' classifiers: Compute the class probabilities as

$$P(C = c_i \mid A_1 = a_1, \ldots, A_m = a_m) =$$
$$\frac{P(A_1 = a_1, \ldots, A_m = a_m \mid C = c_i) \cdot P(C = c_i)}{P(A_1 = a_1, \ldots, A_m = a_m)}.$$

Looks unreasonable at first sight: Even more probabilities to store.

Naive Assumption:
The descriptive attributes are conditionally independent given the class.

Bayes' Rule:

$$P(C = c_i \mid \omega) = \frac{P(A_1 = a_1, \ldots, A_m = a_m \mid C = c_i) \cdot P(C = c_i)}{P(A_1 = a_1, \ldots, A_m = a_m)}$$

$\leftarrow p_0$
abbrev. for the
normalizing constant

Chain Rule of Probability:

$$P(C = c_i \mid \omega) = \frac{P(C = c_i)}{p_0} \cdot \prod_{k=1}^{m} P(A_k = a_k \mid A_1 = a_1, \ldots, A_{k-1} = a_{k-1}, C = c_i)$$

Conditional Independence Assumption:

$$P(C = c_i \mid \omega) = \frac{P(C = c_i)}{p_0} \cdot \prod_{k=1}^{m} P(A_k = a_k \mid C = c_i)$$

Reminder: Chain Rule of Probability

Based on the **product rule** of probability:

$$P(A \wedge B) = P(A \mid B) \cdot P(B)$$

(Multiply definition of conditional probability with $P(B)$.)

Multiple application of the product rule yields:

$$\begin{aligned}
P(A_1, \ldots, A_m) &= P(A_m \mid A_1, \ldots, A_{m-1}) \cdot P(A_1, \ldots, A_{m-1}) \\
&= P(A_m \mid A_1, \ldots, A_{m-1}) \\
&\cdot \; P(A_{m-1} \mid A_1, \ldots, A_{m-2}) \cdot P(A_1, \ldots, A_{m-2}) \\
&= \vdots \\
&= \prod_{k=1}^{m} P(A_k \mid A_1, \ldots, A_{k-1})
\end{aligned}$$

The scheme works also if there is already a condition in the original expression:

$$P(A_1, \ldots, A_m \mid C) = \prod_{i=1}^{m} P(A_k \mid A_1, \ldots, A_{k-1}, C)$$

Conditional Independence

Reminder: **stochastic independence** (unconditional)

$$P(A \wedge B) = P(A) \cdot P(B)$$

(Joint probability is the product of the individual probabilities.)

Comparison to the **product rule**

$$P(A \wedge B) = P(A \mid B) \cdot P(B)$$

shows that this is equivalent to

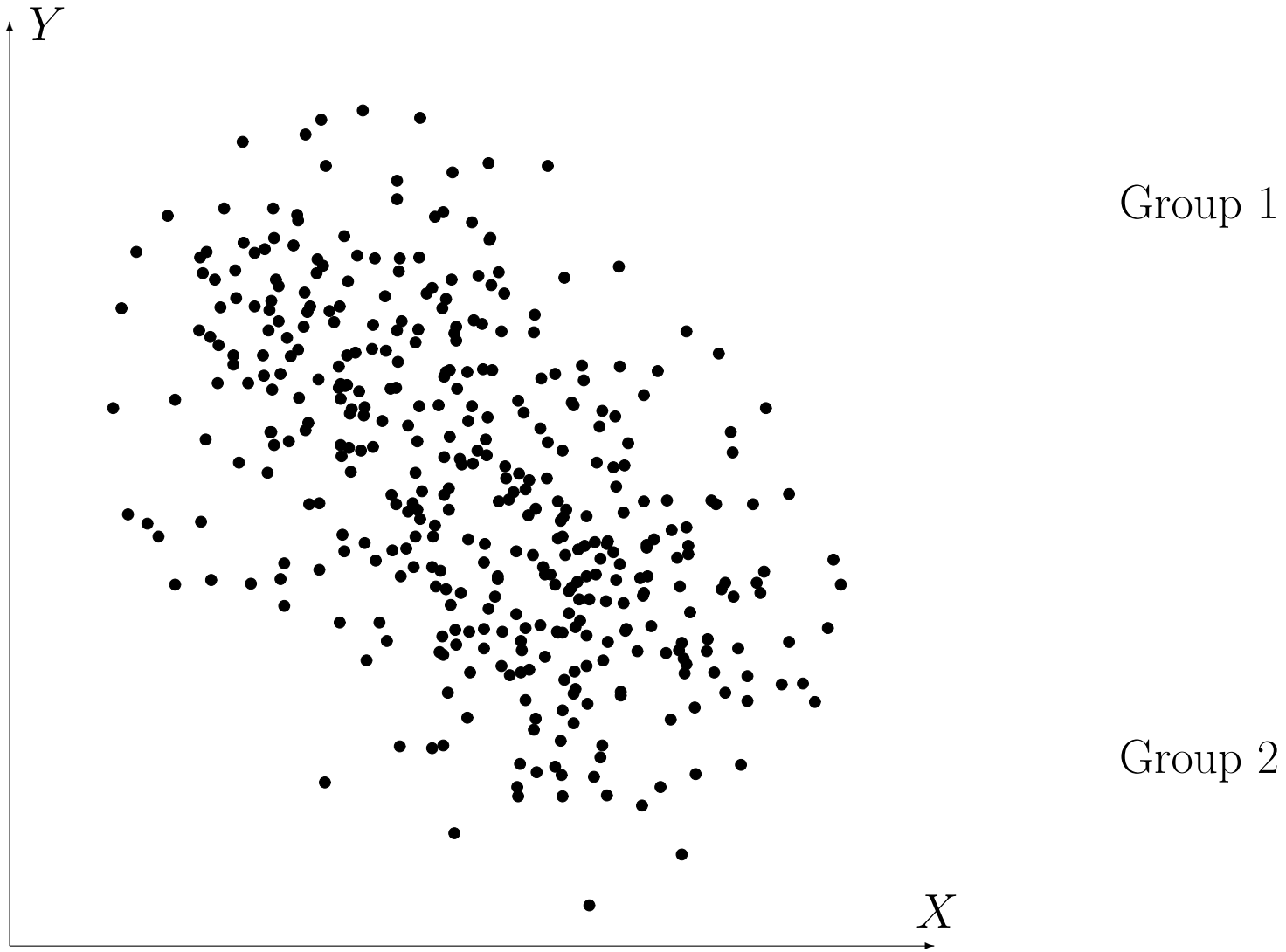
$$P(A \mid B) = P(A)$$

The same formulae hold conditionally, i.e.

$$\begin{aligned} P(A \wedge B \mid C) &= P(A \mid C) \cdot P(B \mid C) \qquad \text{and} \\ P(A \mid B, C) &= P(A \mid C). \end{aligned}$$

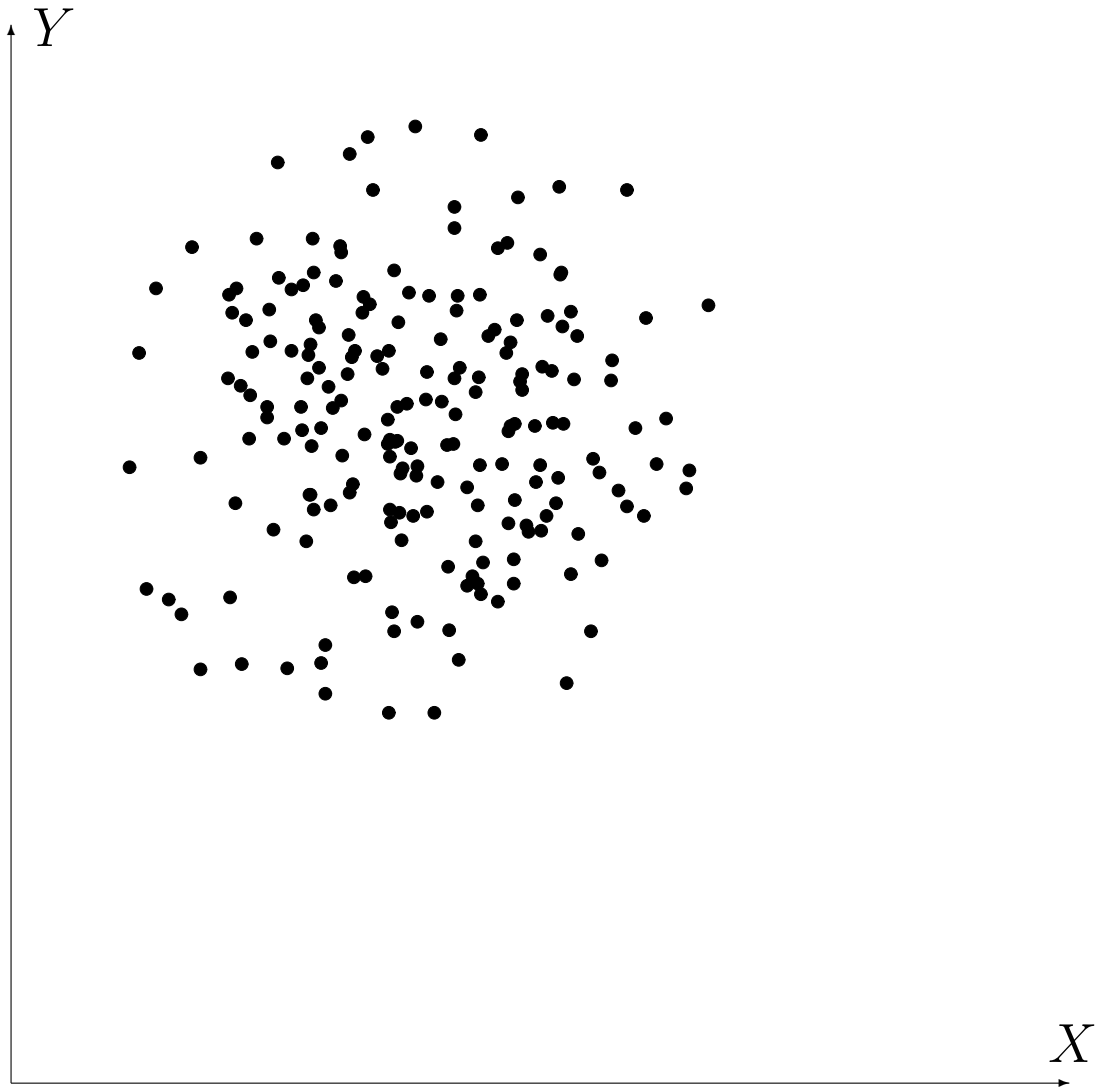
Conditional independence allows us to cancel some conditions.

Conditional Independence: An Example



(Weak) Dependence in the entire dataset: X and Y dependent.

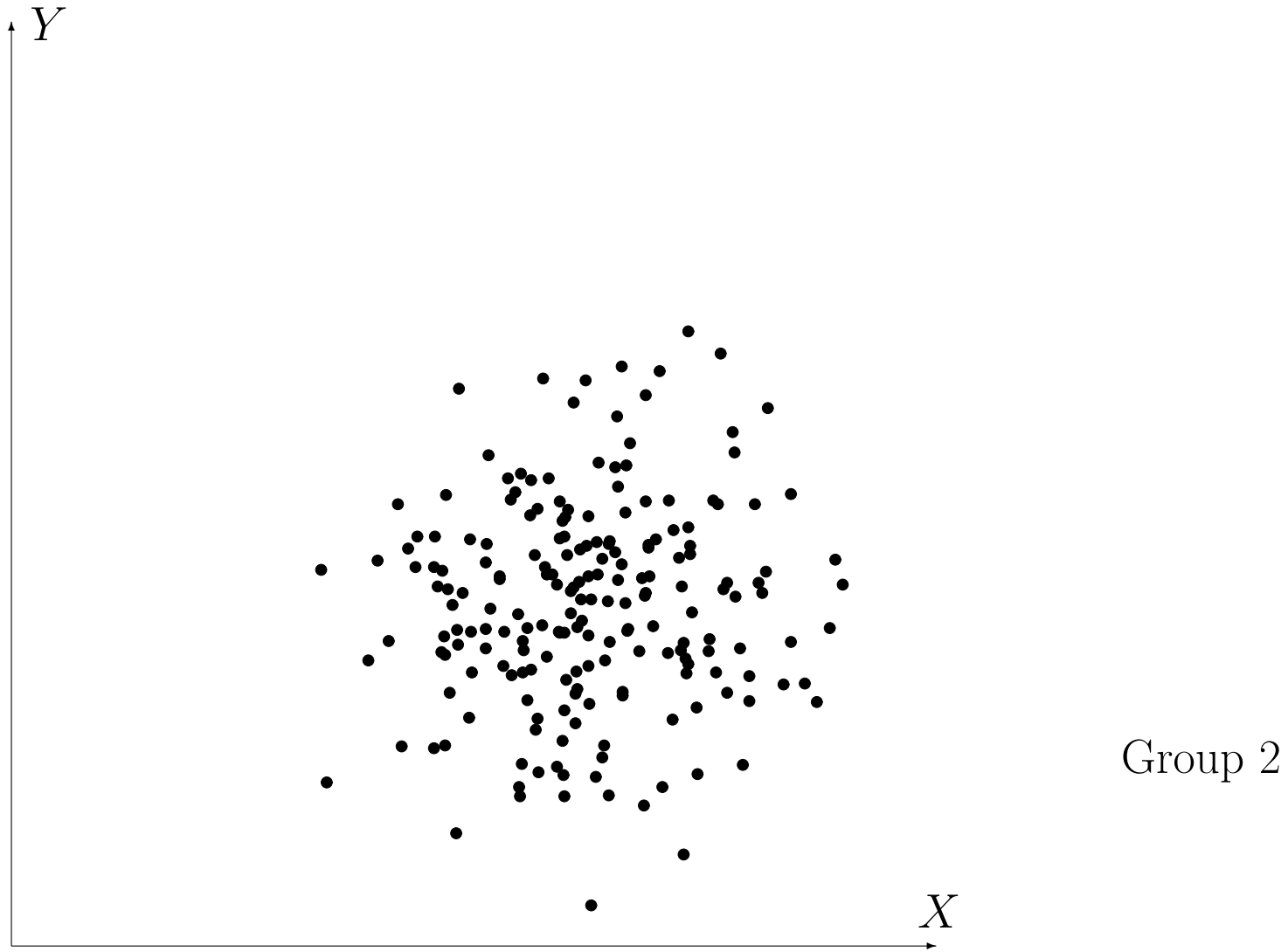
Conditional Independence: An Example



Group 1

No Dependence in Group 1: X and Y conditionally independent given Group 1.

Conditional Independence: An Example



No Dependence in Group 2: X and Y conditionally independent given Group 2.

Marginal & Conditional Independence

The next table shows four 3-dimensional probability distributions (one per row).

The (in)dependencies are always w. r. t. A and B.

The condition variable is C.

		a_1				a_2			
		b_1		b_2		b_1		b_2	
marginal	conditional	c_1	c_2	c_1	c_2	c_1	c_2	c_1	c_2
indep.	indep.	0.03	0.01	0.27	0.09	0.006	0.054	0.054	0.486
dep.	dep.	0.01	0.03	0.126	0.234	0.1275	0.3825	0.0315	0.0585
dep.	indep.	0.12	0.085	0.18	0.015	0.024	0.459	0.036	0.081
indep.	dep.	0.008	0.032	0.144	0.216	0.018	0.042	0.054	0.486

All combinations are possible.

Consequence: Manageable amount of data to store.
Store distributions $P(C = c_i)$ and $\forall 1 \leq k \leq m : P(A_k = a_k \mid C = c_i)$.

It is not necessary to compute p_0 explicitely, because it can be computed implicitly by normalizing the computed values to sum 1.

Estimation of Probabilities:

Nominal/Symbolic Attributes

$$\hat{P}(A_k = a_k \mid C = c_i) = \frac{\#(A_k = a_k, C = c_i) + \gamma}{\#(C = c_i) + n_{A_k}\gamma}$$

γ is called **Laplace correction**: Assume for every class c_i some number of hypothetical samples for every value of A_k to prevent the estimate to be 0 if $\#(A_k = a_k, C = c_i) = 0$.

$\gamma = 0$: Maximum likelihood estimation.

Common choices: $\gamma = 1$ or $\gamma = \frac{1}{2}$.

Estimation of Probabilities:

Metric/Numeric Attributes: Assume a normal distribution.

$$P(A_k = a_k \mid C = c_i) = \frac{1}{\sqrt{2\pi}\sigma_k(c_i)} \exp\left(-\frac{(a_k - \mu_k(c_i))^2}{2\sigma_k^2(c_i)}\right)$$

Estimate of mean value

$$\hat{\mu}_k(c_i) = \frac{1}{\#(C = c_i)} \sum_{j=1}^{\#(C=c_i)} a_k(j)$$

Estimate of variance

$$\hat{\sigma}_k^2(c_i) = \frac{1}{\xi} \sum_{j=1}^{\#(C=c_i)} (a_k(j) - \hat{\mu}_k(c_i))^2$$

$\xi = \#(C = c_i)$: Maximum likelihood estimation
$\xi = \#(C = c_i) - 1$: Unbiased estimation

Naive Bayes Classifiers: Simple Example 1

No	Sex	Age	Blood pr.	Drug
1	male	20	normal	A
2	female	73	normal	B
3	female	37	high	A
4	male	33	low	B
5	female	48	high	A
6	male	29	normal	A
7	female	52	normal	B
8	male	42	low	B
9	male	61	normal	B
10	female	30	normal	A
11	female	26	low	B
12	male	54	high	A

$P(\text{Drug})$	A	B
	0.5	0.5

$P(\text{Sex} \mid \text{Drug})$	A	B
male	0.5	0.5
female	0.5	0.5

$P(\text{Age} \mid \text{Drug})$	A	B
μ	36.3	47.8
σ^2	161.9	311.0

$P(\text{Blood Pr.} \mid \text{Drug})$	A	B
low	0	0.5
normal	0.5	0.5
high	0.5	0

A simple database and estimated (conditional) probability distributions.

Naive Bayes Classifiers: Simple Example 1

$$
\begin{aligned}
&P(\text{Drug A} \mid \text{male, 61, normal}) \\
&\quad = \quad c_1 \cdot P(\text{Drug A}) \cdot P(\text{male} \mid \text{Drug A}) \cdot P(61 \mid \text{Drug A}) \cdot P(\text{normal} \mid \text{Drug A}) \\
&\quad \approx \quad c_1 \cdot 0.5 \cdot 0.5 \cdot 0.004787 \cdot 0.5 \quad = \quad c_1 \cdot 5.984 \cdot 10^{-4} \quad = \quad 0.219
\end{aligned}
$$

$$
\begin{aligned}
&P(\text{Drug B} \mid \text{male, 61, normal}) \\
&\quad = \quad c_1 \cdot P(\text{Drug B}) \cdot P(\text{male} \mid \text{Drug B}) \cdot P(61 \mid \text{Drug B}) \cdot P(\text{normal} \mid \text{Drug B}) \\
&\quad \approx \quad c_1 \cdot 0.5 \cdot 0.5 \cdot 0.017120 \cdot 0.5 \quad = \quad c_1 \cdot 2.140 \cdot 10^{-3} \quad = \quad 0.781
\end{aligned}
$$

$$
\begin{aligned}
&P(\text{Drug A} \mid \text{female, 30, normal}) \\
&\quad = \quad c_2 \cdot P(\text{Drug A}) \cdot P(\text{female} \mid \text{Drug A}) \cdot P(30 \mid \text{Drug A}) \cdot P(\text{normal} \mid \text{Drug A}) \\
&\quad \approx \quad c_2 \cdot 0.5 \cdot 0.5 \cdot 0.027703 \cdot 0.5 \quad = \quad c_2 \cdot 3.471 \cdot 10^{-3} \quad = \quad 0.671
\end{aligned}
$$

$$
\begin{aligned}
&P(\text{Drug B} \mid \text{female, 30, normal}) \\
&\quad = \quad c_2 \cdot P(\text{Drug B}) \cdot P(\text{female} \mid \text{Drug B}) \cdot P(30 \mid \text{Drug B}) \cdot P(\text{normal} \mid \text{Drug B}) \\
&\quad \approx \quad c_2 \cdot 0.5 \cdot 0.5 \cdot 0.013567 \cdot 0.5 \quad = \quad c_2 \cdot 1.696 \cdot 10^{-3} \quad = \quad 0.329
\end{aligned}
$$

Naive Bayes Classifiers: Simple Example 2

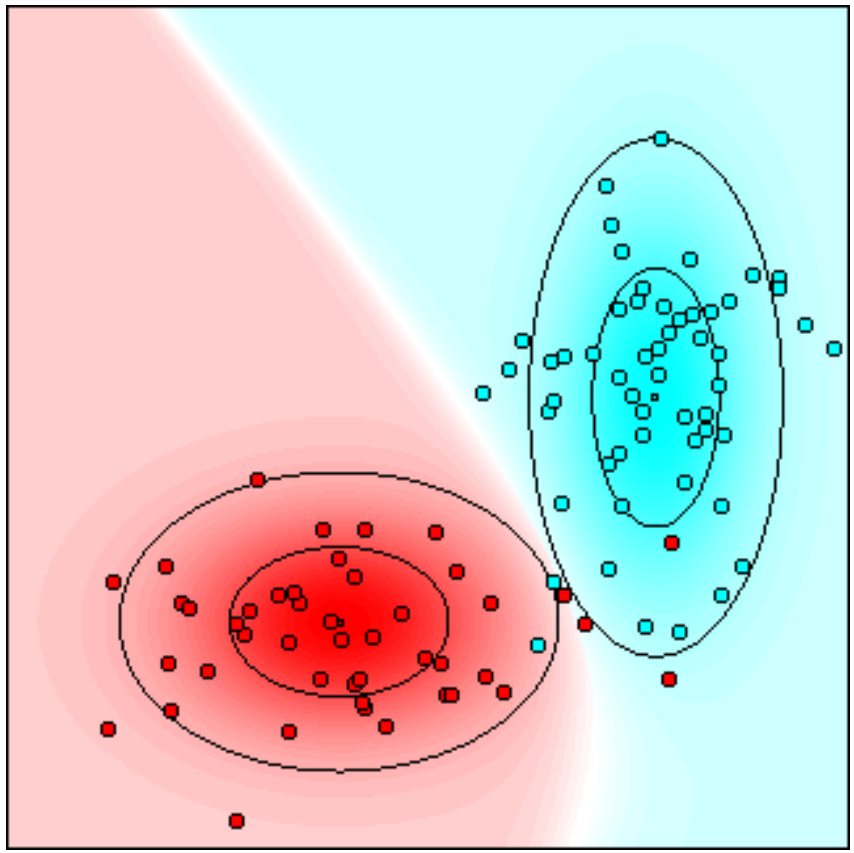
100 data points, 2 classes

Small squares: mean values

Inner ellipses:
one standard deviation

Outer ellipses:
two standard deviations

Classes overlap:
classification is not perfect



Naive Bayes Classifier

Naive Bayes Classifiers: Simple Example 3

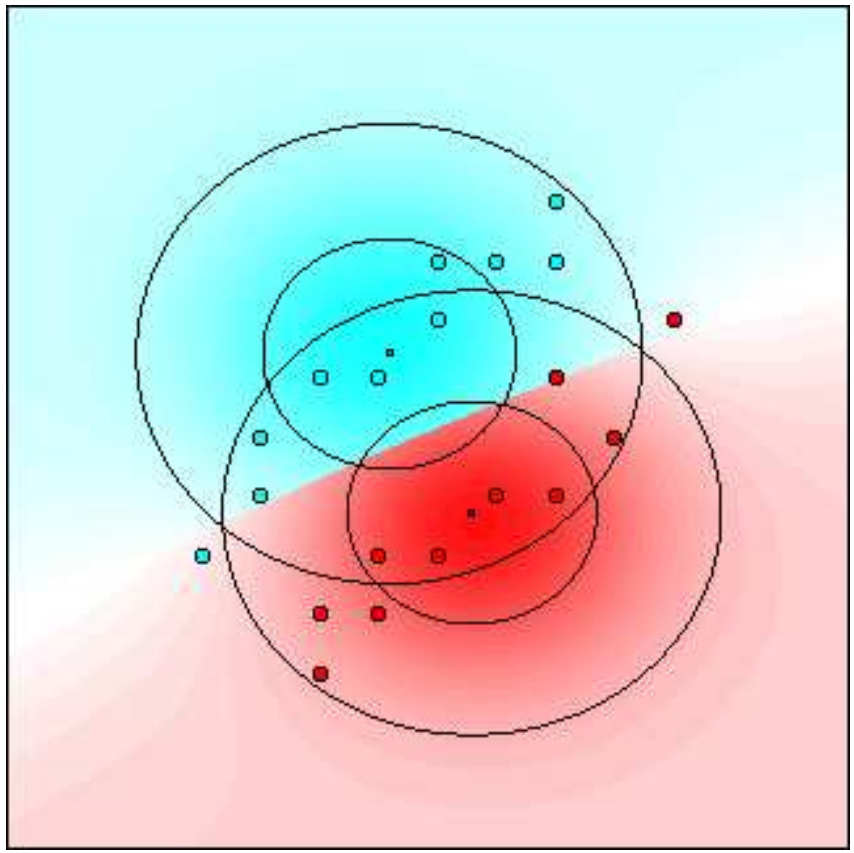
20 data points, 2 classes

Small squares: mean values

Inner ellipses:
one standard deviation

Outer ellipses:
two standard deviations

Attributes are not conditionally independent given the class



Naive Bayes Classifier

Naive Bayes Classifiers: Iris Data

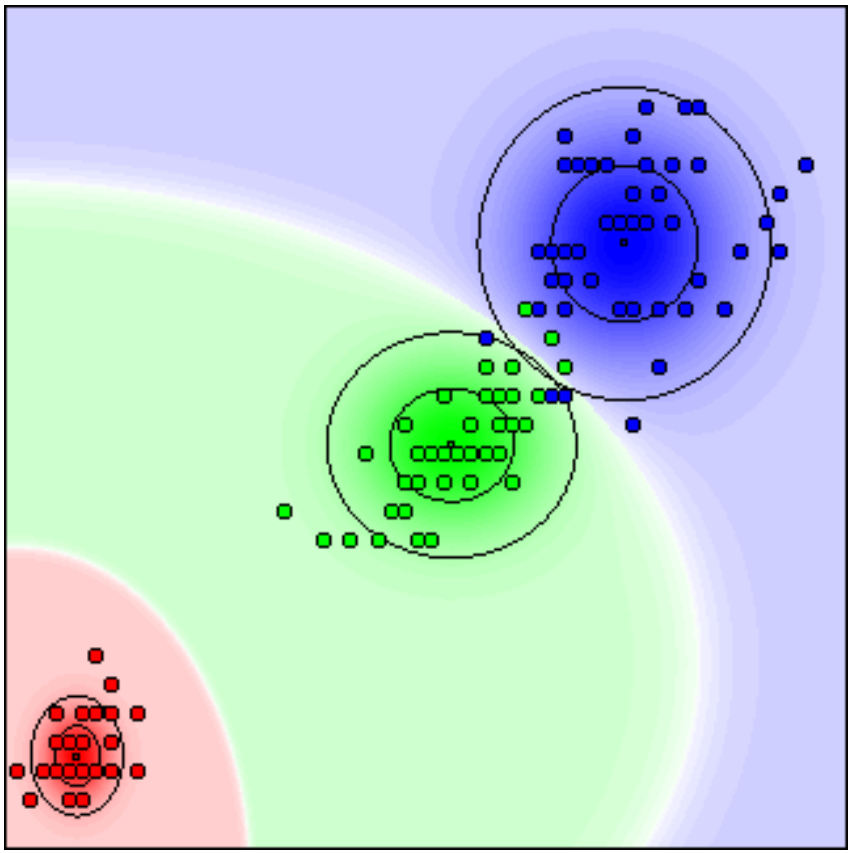
150 data points, 3 classes

Iris setosa (red)
Iris versicolor (green)
Iris virginica (blue)

Shown: 2 out of 4 attributes

sepal length
sepal width
petal length (horizontal)
petal width (vertical)

6 misclassifications
on the training data
(with all 4 attributes)



Naive Bayes Classifier

Full Bayes Classifiers

Restricted to metric/numeric attributes (only the class is nominal/symbolic).

Simplifying Assumption:
Each class can be described by a multivariate normal distribution.

$$f(A_1 = a_1, \ldots, A_m = a_m \mid C = c_i)$$
$$= \frac{1}{\sqrt{(2\pi)^m |\boldsymbol{\Sigma}_i|}} \exp\left(-\frac{1}{2}(\vec{a} - \vec{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\vec{a} - \vec{\mu}_i)\right)$$

$\vec{\mu}_i$: mean value vector for class c_i

$\boldsymbol{\Sigma}_i$: covariance matrix for class c_i

Intuitively: Each class has a bell-shaped probability density.

Naive Bayes classifiers: Covariance matrices are diagonal matrices.
(Details about this relation are given below.)

Estimation of Probabilities:

Estimate of mean value vector

$$\hat{\vec{\mu}}_i = \frac{1}{\#(C = c_i)} \sum_{j=1}^{\#(C=c_i)} \vec{a}(j)$$

Estimate of covariance matrix

$$\widehat{\mathbf{\Sigma}}_i = \frac{1}{\xi} \sum_{j=1}^{\#(C=c_i)} \left(\vec{a}(j) - \hat{\vec{\mu}}_i\right) \left(\vec{a}(j) - \hat{\vec{\mu}}_i\right)^\top$$

$\xi = \#(C = c_i)$: Maximum likelihood estimation
$\xi = \#(C = c_i) - 1$: Unbiased estimation

$\vec{x}^\top$ denotes the transpose of the vector $\vec{x}$.

$\vec{x}\vec{x}^\top$ is the so-called **outer product** or **matrix product** of $\vec{x}$ with itself.

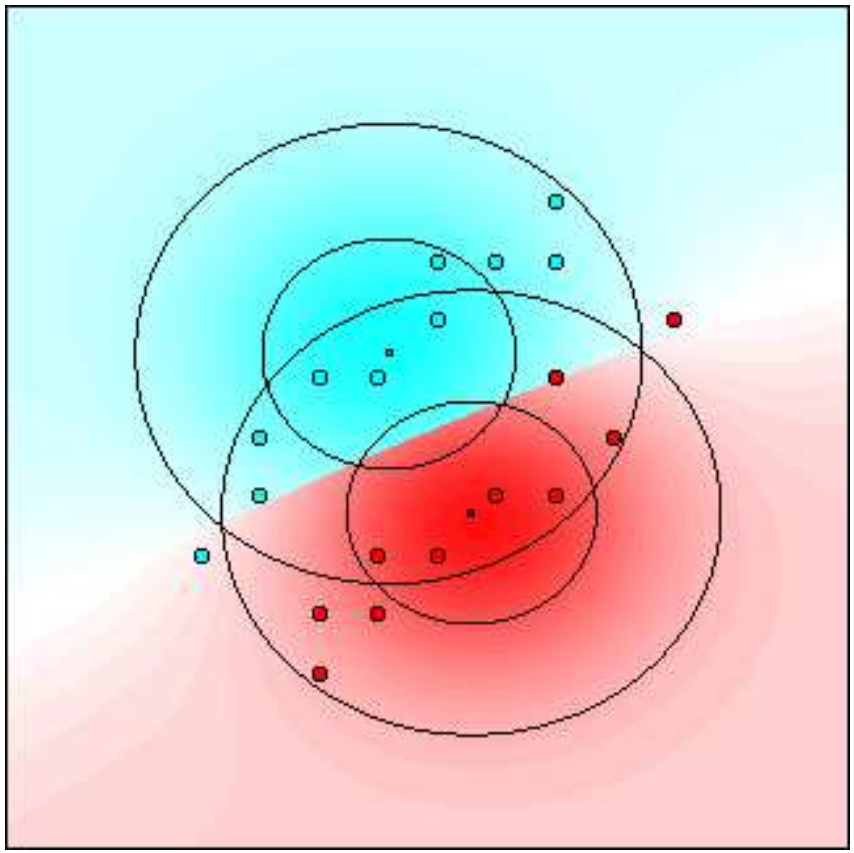
Comparison of Naive and Full Bayes Classifiers

Naive Bayes classifiers for metric/numeric data are equivalent
to full Bayes classifiers with diagonal covariance matrices:

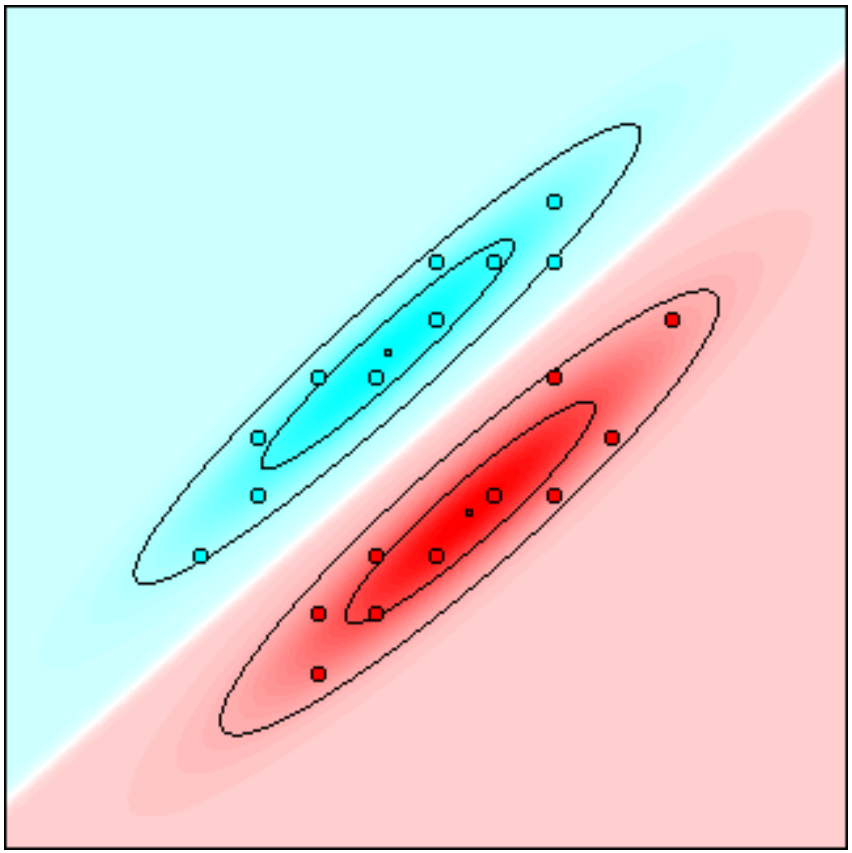
$$
\begin{aligned}
& f(A_1 = a_1, \ldots, A_m = a_m \mid C = c_i) \\
&= \frac{1}{\sqrt{(2\pi)^m |\boldsymbol{\Sigma}_i|}} \cdot \exp\Big(-\frac{1}{2}(\vec{a} - \vec{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1} (\vec{a} - \vec{\mu}_i)\Big) \\
&= \frac{1}{\sqrt{(2\pi)^m \prod_{k=1}^m \sigma_{i,k}^2}} \cdot \exp\Big(-\frac{1}{2}(\vec{a} - \vec{\mu}_i)^\top \operatorname{diag}\Big(\sigma_{i,1}^{-2}, \ldots, \sigma_{i,m}^{-2}\Big)\,(\vec{a} - \vec{\mu}_i)\Big) \\
&= \frac{1}{\prod_{k=1}^m \sqrt{2\pi\sigma_{i,k}^2}} \cdot \exp\left(-\frac{1}{2}\sum_{k=1}^m \frac{(a_k - \mu_{i,k})^2}{\sigma_{i,k}^2}\right) \\
&= \prod_{k=1}^m \frac{1}{\sqrt{2\pi\sigma_{i,k}^2}} \cdot \exp\left(-\frac{(a_k - \mu_{i,k})^2}{2\sigma_{i,k}^2}\right) \quad \widehat{=} \quad \prod_{k=1}^m f(A_k = a_k \mid C = c_i),
\end{aligned}
$$

where $f(A_k = a_k \mid C = c_i)$ are the density functions used by a naive Bayes classifier.

Comparison of Naive and Full Bayes Classifiers



Naive Bayes Classifier



Full Bayes Classifier

Full Bayes Classifiers: Iris Data

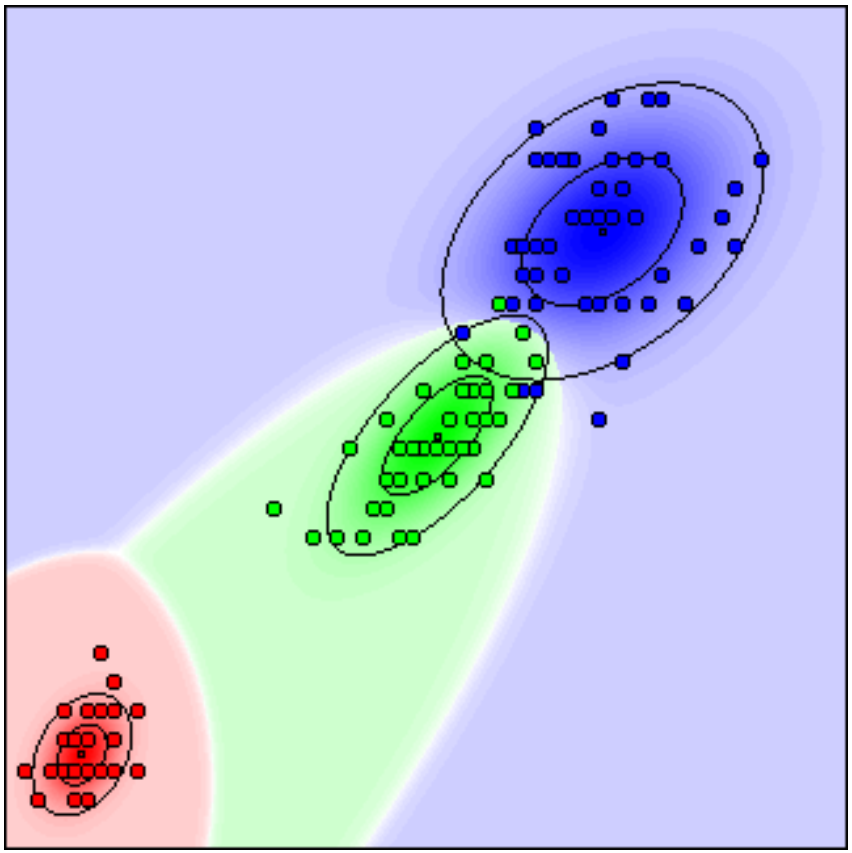
150 data points, 3 classes

Iris setosa (red)
Iris versicolor (green)
Iris virginica (blue)

Shown: 2 out of 4 attributes

sepal length
sepal width
petal length (horizontal)
petal width (vertical)

2 misclassifications
on the training data
(with all 4 attributes)



Full Bayes Classifier

Summary Bayes Classifiers

Probabilistic Classification: Assign the most probable class.

Bayes' Rule: "Invert" the conditional class probabilities.

Naive Bayes Classifiers

Simplifying Assumption:
Attributes are conditionally independent given the class.

Can handle nominal/symbolic as well as metric/numeric attributes.

Full Bayes Classifiers

Simplifying Assumption:
Each class can be described by a multivariate normal distribution.

Can handle only metric/numeric attributes.

Decision Trees

Classification with a Decision Tree

Top-down Induction of Decision Trees

- A simple example
- The general algorithm
- Attribute selection measures
- Treatment of numeric attributes and missing values

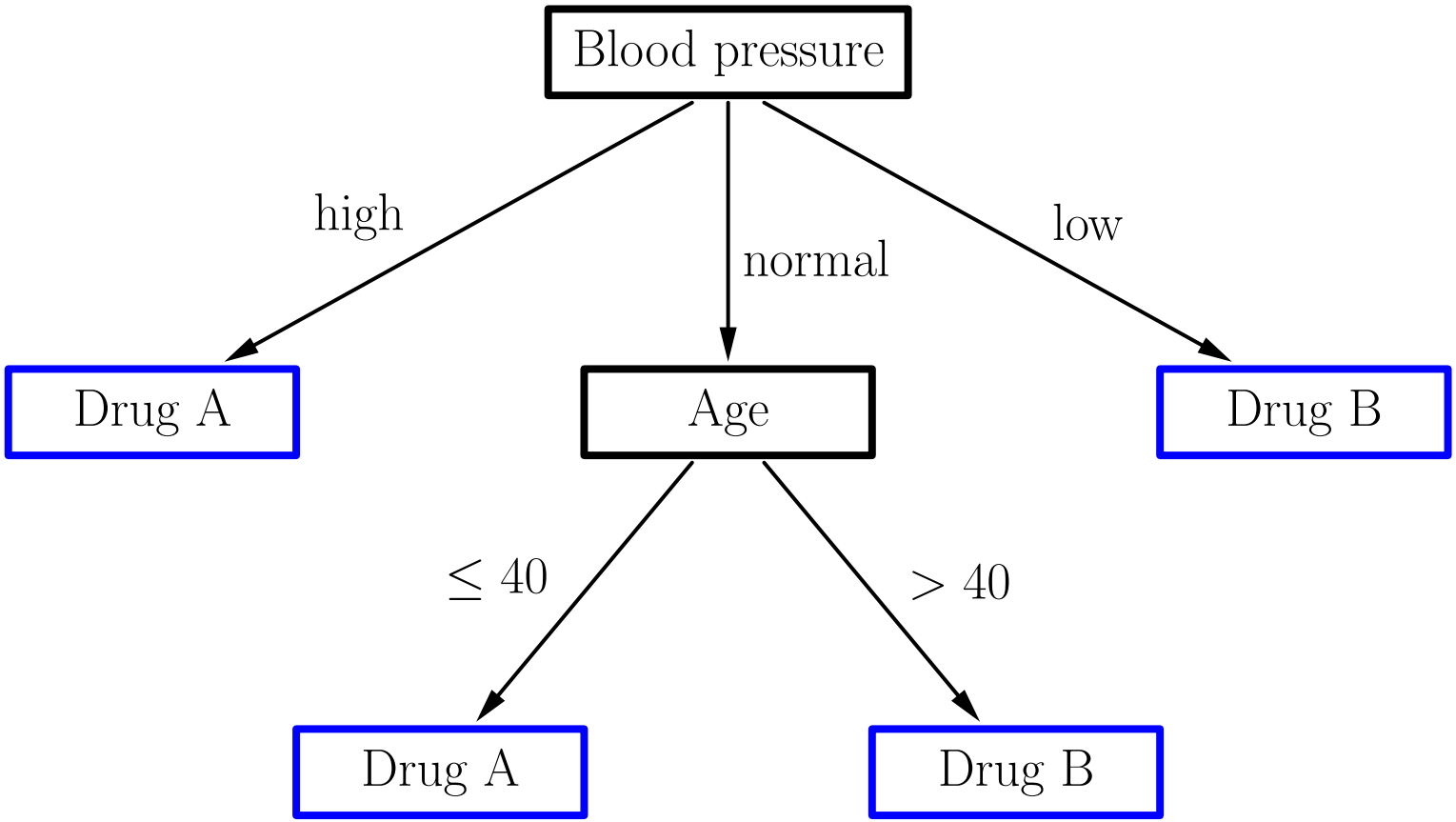
Pruning Decision Trees

- General approaches
- A simple example

Summary

A Very Simple Decision Tree

Assignment of a drug to a patient:



Classification with a Decision Tree

Recursive Descent:

Start at the root node.

If the current node is an **leaf node**:

Return the class assigned to the node.

If the current node is an **inner node**:

Test the attribute associated with the node.

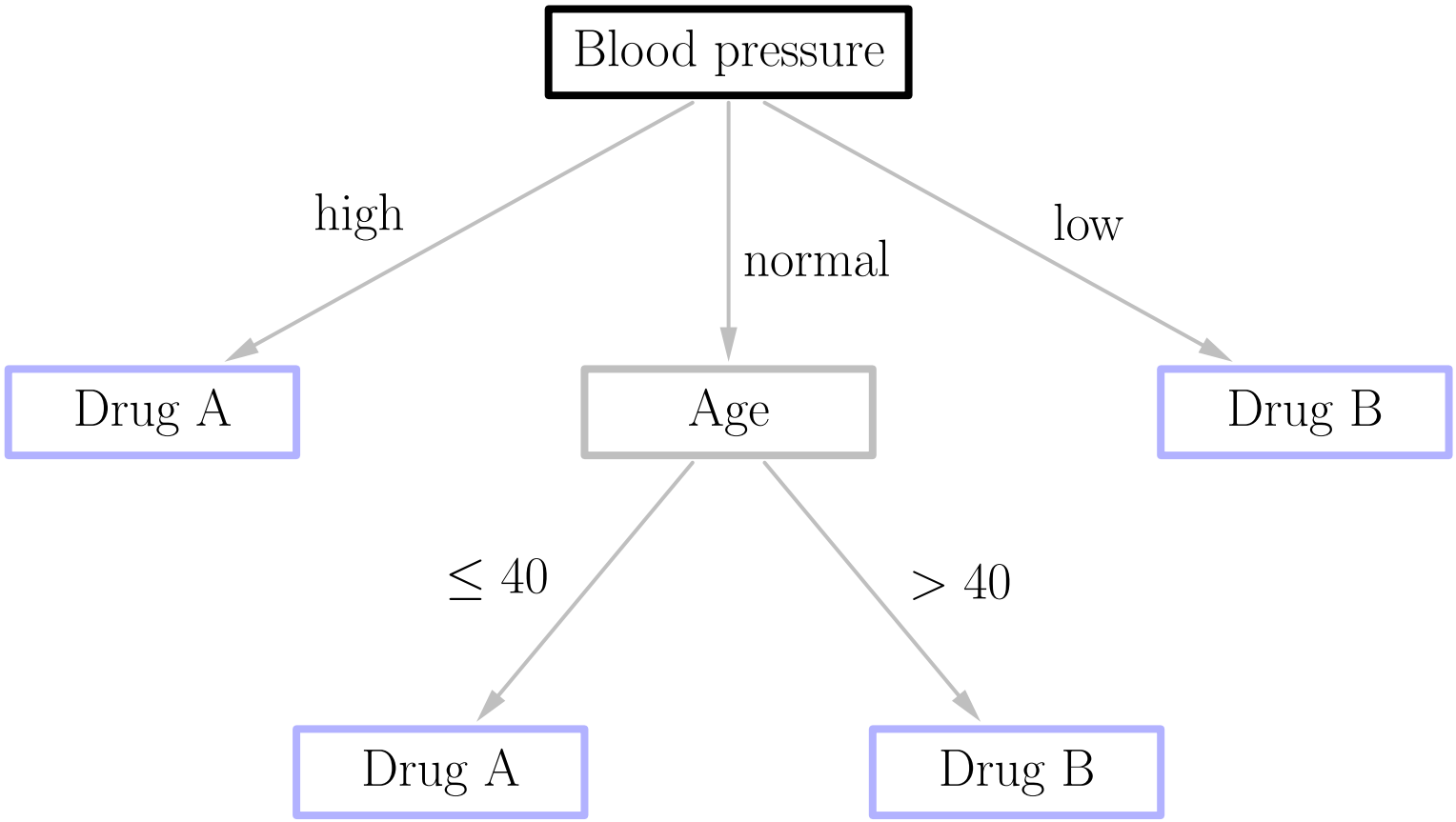
Follow the branch labeled with the outcome of the test.

Apply the algorithm recursively.

Intuitively: Follow the path corresponding to the case to be classified.

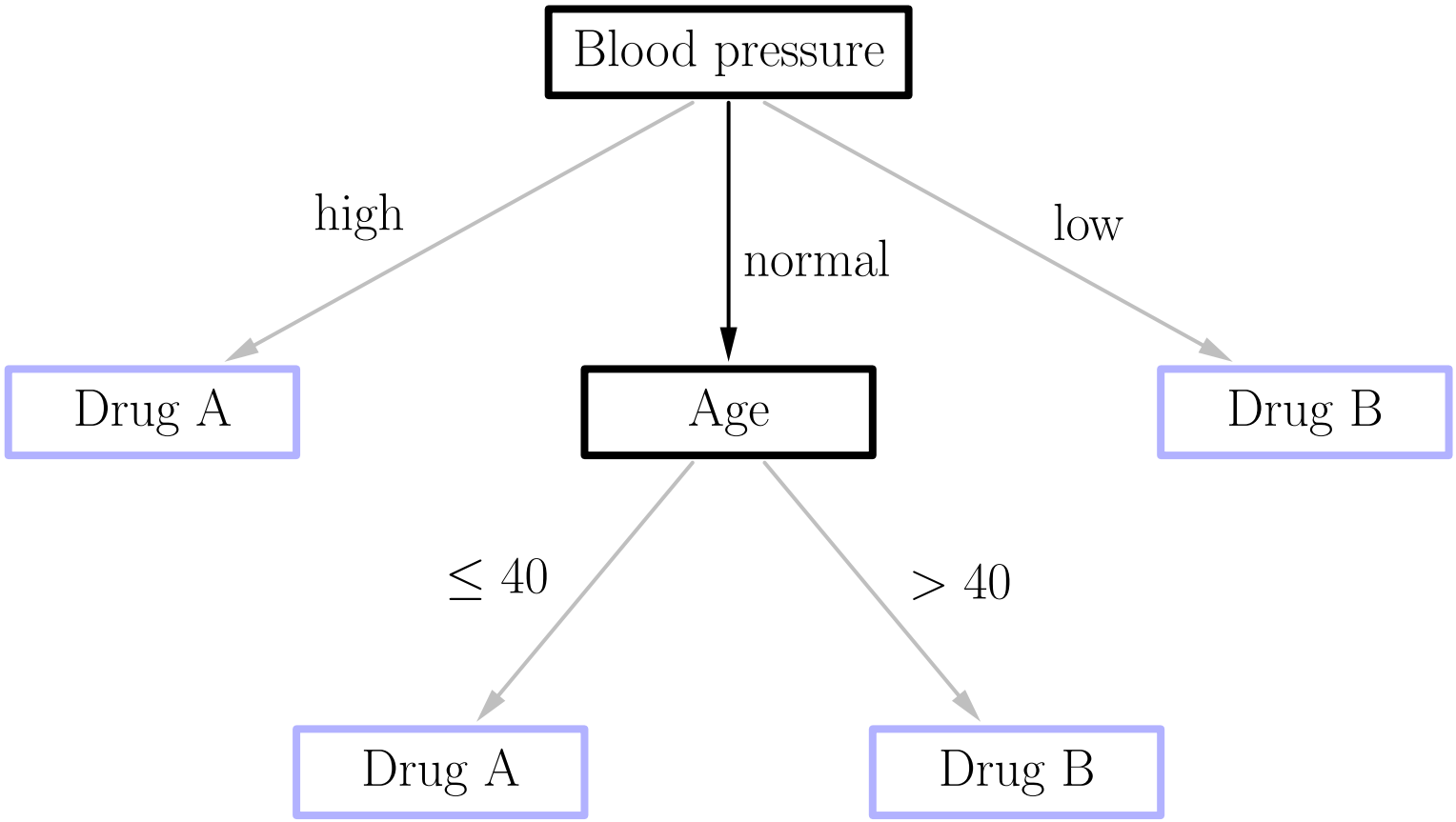
Classification in the Example

Assignment of a drug to a patient:



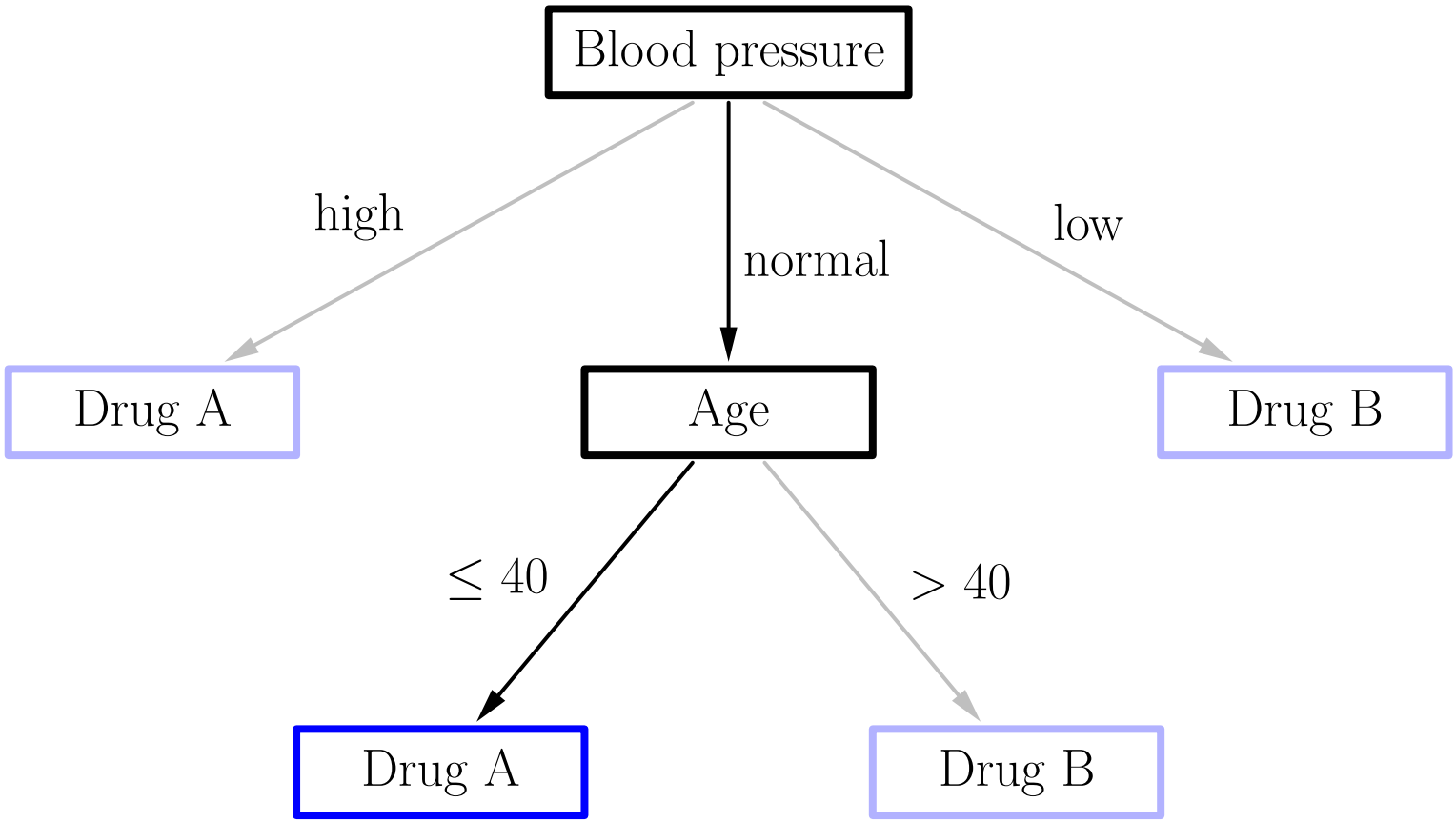
Classification in the Example

Assignment of a drug to a patient:



Classification in the Example

Assignment of a drug to a patient:



Induction of Decision Trees

Top-down approach

- Build the decision tree from top to bottom
  (from the root to the leaves).

Greedy Selection of a Test Attribute

- Compute an evaluation measure for all attributes.
- Select the attribute with the best evaluation.

Divide and Conquer / Recursive Descent

- Divide the example cases according to the values of the test attribute.
- Apply the procedure recursively to the subsets.
- Terminate the recursion if
  - all cases belong to the same class
  - no more test attributes are available

Induction of a Decision Tree: Example

Patient database

12 example cases

3 descriptive attributes

1 class attribute

Assignment of drug

(without patient attributes)

always drug A or always drug B:

50% correct (in 6 of 12 cases)

No	Sex	Age	Blood pr.	Drug
1	male	20	normal	A
2	female	73	normal	B
3	female	37	high	A
4	male	33	low	B
5	female	48	high	A
6	male	29	normal	A
7	female	52	normal	B
8	male	42	low	B
9	male	61	normal	B
10	female	30	normal	A
11	female	26	low	B
12	male	54	high	A

Induction of a Decision Tree: Example

Sex of the patient

Division w.r.t. male/female.

Assignment of drug

male:	50% correct	(in 3 of 6 cases)
female:	50% correct	(in 3 of 6 cases)
total:	**50% correct**	(in 6 of 12 cases)

No	Sex	Drug
1	male	A
6	male	A
12	male	A
4	male	B
8	male	B
9	male	B
3	female	A
5	female	A
10	female	A
2	female	B
7	female	B
11	female	B

Induction of a Decision Tree: Example

Age of the patient

Sort according to age.

Find best age split.
here: ca. 40 years

Assignment of drug

≤ 40:	A	67% correct	(in 4 of 6 cases)
> 40:	B	67% correct	(in 4 of 6 cases)
total:		**67% correct**	(in 8 of 12 cases)

No	Age	Drug
1	20	A
11	26	B
6	29	A
10	30	A
4	33	B
3	37	A
8	42	B
5	48	A
7	52	B
12	54	A
9	61	B
2	73	B

Induction of a Decision Tree: Example

Blood pressure of the patient

Division w.r.t. high/normal/low.

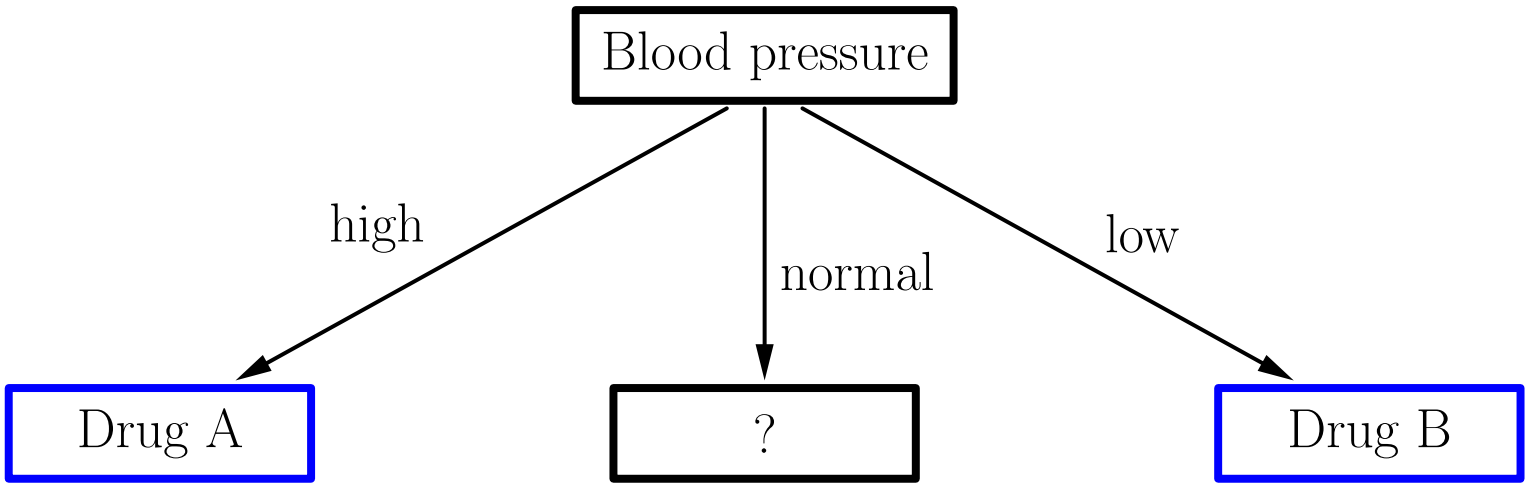
Assignment of drug

high:	A	100% correct	(in 3 of 3 cases)
normal:		50% correct	(in 3 of 6 cases)
low:	B	100% correct	(in 3 of 3 cases)
total:		**75% correct**	(in 9 of 12 cases)

No	Blood pr.	Drug
3	high	A
5	high	A
12	high	A
1	normal	A
6	normal	A
10	normal	A
2	normal	B
7	normal	B
9	normal	B
4	low	B
8	low	B
11	low	B

Induction of a Decision Tree: Example

Current Decision Tree:



Induction of a Decision Tree: Example

Blood pressure and sex

Only patients
with normal blood pressure.

Division w.r.t. male/female.

Assignment of drug

male:	A	67% correct	(2 of 3)
female:	B	67% correct	(2 of 3)
total:		**67% correct**	(4 of 6)

No	Blood pr.	Sex	Drug
3	high		A
5	high		A
12	high		A
1	normal	male	A
6	normal	male	A
9	normal	male	B
2	normal	female	B
7	normal	female	B
10	normal	female	A
4	low		B
8	low		B
11	low		B

Induction of a Decision Tree: Example

Blood pressure and age

Only patients
with normal blood pressure.

Sort according to age.

Find best age split.
here: ca. 40 years

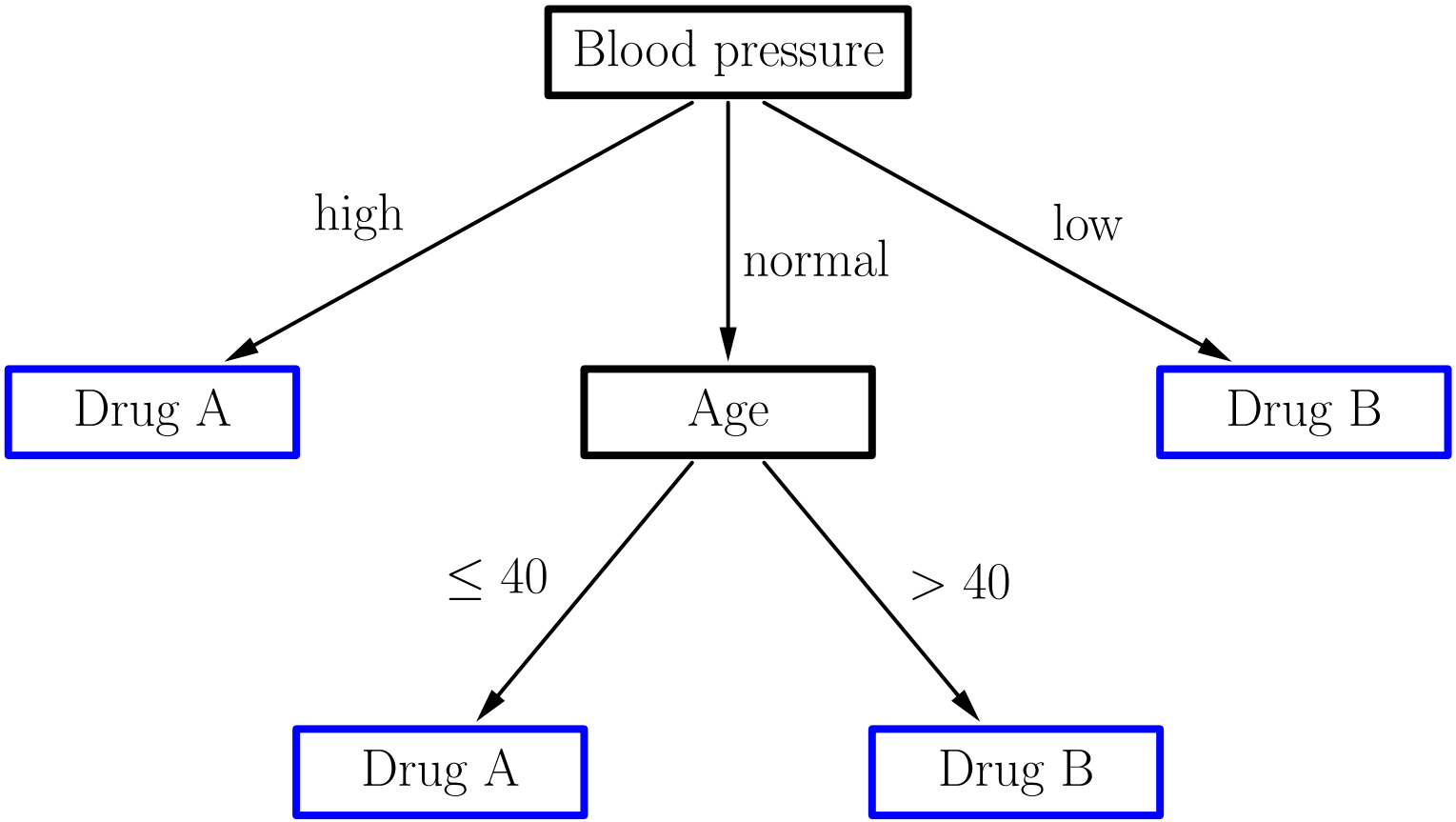
Assignment of drug

≤ 40:	A	100% correct	(3 of 3)
> 40:	B	100% correct	(3 of 3)
total:		**100% correct**	(6 of 6)

No	Blood pr.	Age	Drug
3	high		A
5	high		A
12	high		A
1	normal	20	A
6	normal	29	A
10	normal	30	A
7	normal	52	B
9	normal	61	B
2	normal	73	B
11	low		B
4	low		B
8	low		B

Result of Decision Tree Induction

Assignment of a drug to a patient:



Decision Tree Induction: Notation

S	a set of case or object descriptions
C	the class attribute
$A^{(1)}, \ldots, A^{(m)}$	other attributes (index dropped in the following)
$\mathrm{dom}(C)$	$= \{c_1, \ldots, c_{n_C}\}$, n_C: number of classes
$\mathrm{dom}(A)$	$= \{a_1, \ldots, a_{n_A}\}$, n_A: number of attribute values
$N_{\cdot\cdot}$	total number of case or object descriptions i.e. $N_{\cdot\cdot} = \lvert S \rvert$
$N_{i\cdot}$	absolute frequency of the class c_i
$N_{\cdot j}$	absolute frequency of the attribute value a_j
N_{ij}	absolute frequency of the combination of the class c_i and the attribute value a_j. It is $N_{i\cdot} = \sum_{j=1}^{n_A} N_{ij}$ and $N_{\cdot j} = \sum_{i=1}^{n_C} N_{ij}$.
$p_{i\cdot}$	relative frequency of the class c_i, $p_{i\cdot} = \frac{N_{i\cdot}}{N_{\cdot\cdot}}$
$p_{\cdot j}$	relative frequency of the attribute value a_j, $p_{\cdot j} = \frac{N_{\cdot j}}{N_{\cdot\cdot}}$
p_{ij}	relative frequency of the combination of class c_i and attribute value a_j, $p_{ij} = \frac{N_{ij}}{N_{\cdot\cdot}}$
$p_{i\lvert j}$	relative frequency of the class c_i in cases having attribute value a_j, $p_{i\lvert j} = \frac{N_{ij}}{N_{\cdot j}} = \frac{p_{ij}}{p_{\cdot j}}$

Decision Tree Induction: General Algorithm

function grow_tree (S : set of cases) : node;
begin
    $best\_v$:= WORTHLESS;
    for all untested attributes A **do**
        compute frequencies N_{ij}, $N_{i.}$, $N_{.j}$ for $1 \leq i \leq n_C$ and $1 \leq j \leq n_A$;
        compute value v of an evaluation measure using N_{ij}, $N_{i.}$, $N_{.j}$;
        if $v > best\_v$ **then** $best\_v := v$; $best\_A := A$; **end**;
    end
    if $best\_v$ = WORTHLESS
    then create leaf node x;
        assign majority class of S to x;
    else create test node x;
        assign test on attribute $best\_A$ to x;
        for all $a \in \mathrm{dom}(best\_A)$ **do** x.child[a] := grow_tree($S|_{best\_A=a}$); **end**;
    end;
    return x;
end;

Evaluation Measures

Evaluation measure used in the above example:
rate of correctly classified example cases.

Advantage: simple to compute, easy to understand.

Disadvantage: works well only for two classes.

If there are more than two classes, the rate of misclassified example cases **neglects a lot of the available information**.

Only the majority class—that is, the class occurring most often in (a subset of) the example cases—is really considered.

The distribution of the other classes has no influence. However, a good choice here can be important for deeper levels of the decision tree.

Therefore: Study also other evaluation measures. Here:

Information gain and its various normalizations.

χ^2 **measure** (well-known in statistics).

An Information-theoretic Evaluation Measure

Information Gain (Kullback and Leibler 1951, Quinlan 1986)

Based on Shannon Entropy $H = -\sum_{i=1}^{n} p_i \log_2 p_i$ (Shannon 1948)

$$I_{\text{gain}}(C, A) = \overbrace{H(C)}^{} - \overbrace{H(C|A)}^{}$$

$$= \overbrace{-\sum_{i=1}^{n_C} p_{i.} \log_2 p_{i.}}^{} - \overbrace{\sum_{j=1}^{n_A} p_{.j} \left(-\sum_{i=1}^{n_C} p_{i|j} \log_2 p_{i|j} \right)}^{}$$

$H(C)$	Entropy of the class distribution (C: class attribute)
$H(C\|A)$	*Expected entropy* of the class distribution if the value of the attribute A becomes known
$H(C) - H(C\|A)$	Expected entropy reduction or *information gain*

Inducing the Decision Tree with Information Gain

Information gain for drug and sex:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Sex}) = \frac{1}{2}\underbrace{\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)}_{H(\text{Drug}|\text{Sex=male})} + \frac{1}{2}\underbrace{\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)}_{H(\text{Drug}|\text{Sex=female})} = 1$$

$$I_{\text{gain}}(\text{Drug}, \text{Sex}) = 1 - 1 = 0$$

No gain at all since the initial the uniform distribution of drug is splitted into two (still) uniform distributions.

Inducing the Decision Tree with Information Gain

Information gain for drug and age:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Age}) = \frac{1}{2}\underbrace{\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right)}_{H(\text{Drug}|\text{Age}\leq 40)} + \frac{1}{2}\underbrace{\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right)}_{H(\text{Drug}|\text{Age}>40)} \approx 0.9183$$

$$I_{\text{gain}}(\text{Drug}, \text{Age}) = 1 - 0.9183 = 0.0817$$

Splitting w. r. t. age can reduce the overall entropy.

Inducing the Decision Tree with Information Gain

Information gain for drug and blood pressure:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Blood\_pr}) = \frac{1}{4}\cdot 0 + \frac{1}{2}\Bigg(\underbrace{-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}}_{H(\text{Drug}|\text{Blood\_pr=normal})} \Bigg) + \frac{1}{4}\cdot 0 = 0.5$$

$$I_{\text{gain}}(\text{Drug}, \text{Blood\_pr}) = 1 - 0.5 = 0.5$$

Largest information gain, so we first split w. r. t. blood pressure (as in the example with misclassification rate).

Inducing the Decision Tree with Information Gain

Next level: Subtree blood pressure is normal.

Information gain for drug and sex:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Sex}) = \frac{1}{2}\underbrace{\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right)}_{H(\text{Drug}|\text{Sex=male})} + \frac{1}{2}\underbrace{\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right)}_{H(\text{Drug}|\text{Sex=female})} = 0.9183$$

$$I_{\text{gain}}(\text{Drug}, \text{Sex}) = 0.0817$$

Entropy can be decreased.

Inducing the Decision Tree with Information Gain

Next level: Subtree blood pressure is normal.

Information gain for drug and age:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Age}) = \frac{1}{2}\cdot 0 + \frac{1}{2}\cdot 0 = 0$$

$$I_{\text{gain}}(\text{Drug}, \text{Age}) = 1$$

Maximal information gain, that is we result in a perfect classification (again, as in the case of using misclassification rate).

Interpretation of Shannon Entropy

Let $S = \{s_1, \ldots, s_n\}$ be a finite set of alternatives having positive probabilities $P(s_i)$, $i = 1, \ldots, n$, satisfying $\sum_{i=1}^{n} P(s_i) = 1$.

Shannon Entropy:

$$H(S) = -\sum_{i=1}^{n} P(s_i) \log_2 P(s_i)$$

Intuitively: **Expected number of yes/no questions that have to be asked in order to determine the obtaining alternative.**

Suppose there is an oracle, which knows the obtaining alternative, but responds only if the question can be answered with "yes" or "no".

A better question scheme than asking for one alternative after the other can easily be found: Divide the set into two subsets of about equal size.

Ask for containment in an arbitrarily chosen subset.

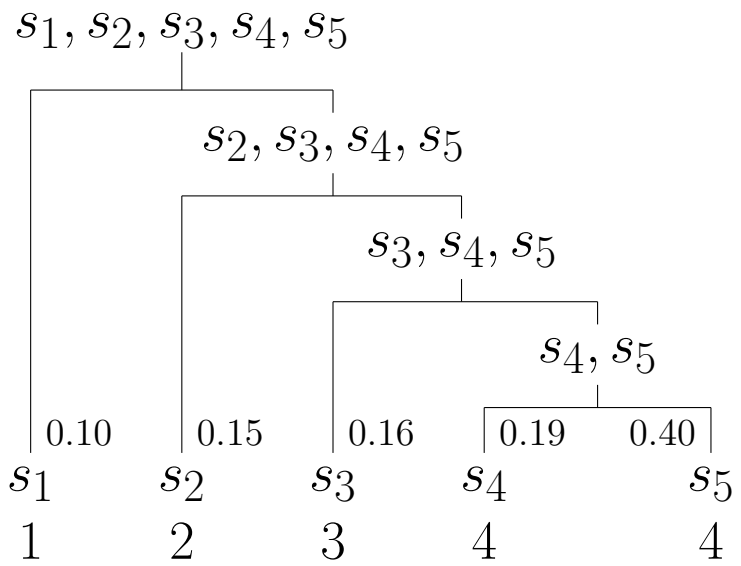
Apply this scheme recursively $\rightarrow$ number of questions bounded by $\lceil \log_2 n \rceil$.

Question/Coding Schemes

$$P(s_1) = 0.10, \quad P(s_2) = 0.15, \quad P(s_3) = 0.16, \quad P(s_4) = 0.19, \quad P(s_5) = 0.40$$

Shannon entropy: $-\sum_i P(s_i) \log_2 P(s_i) = 2.15$ bit/symbol

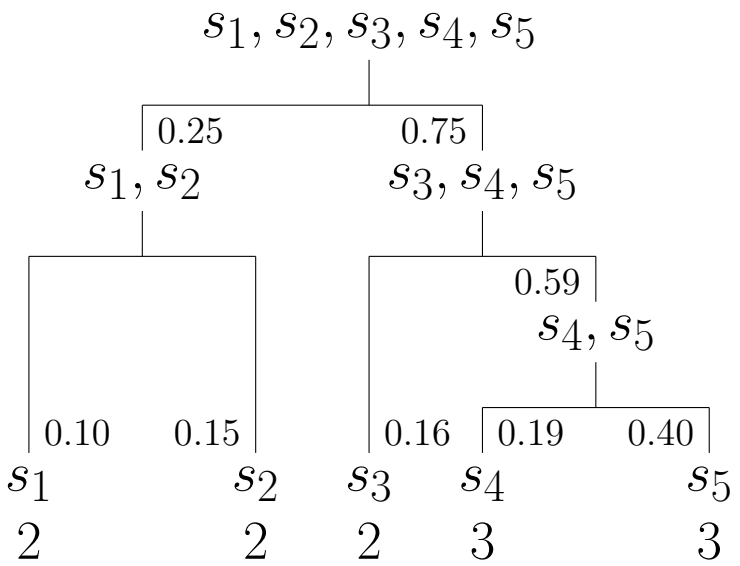
Linear Traversal



Code length: 3.24 bit/symbol

Code efficiency: 0.664

Equal Size Subsets



Code length: 2.59 bit/symbol

Code efficiency: 0.830

Question/Coding Schemes

Splitting into subsets of about equal size can lead to a bad arrangement of the alternatives into subsets $\rightarrow$ high expected number of questions.

Good question schemes take the probability of the alternatives into account.

Shannon-Fano Coding (1948)

Build the question/coding scheme top-down.

Sort the alternatives w.r.t. their probabilities.

Split the set so that the subsets have about equal *probability*
(splits must respect the probability order of the alternatives).

Huffman Coding (1952)

Build the question/coding scheme bottom-up.

Start with one element sets.

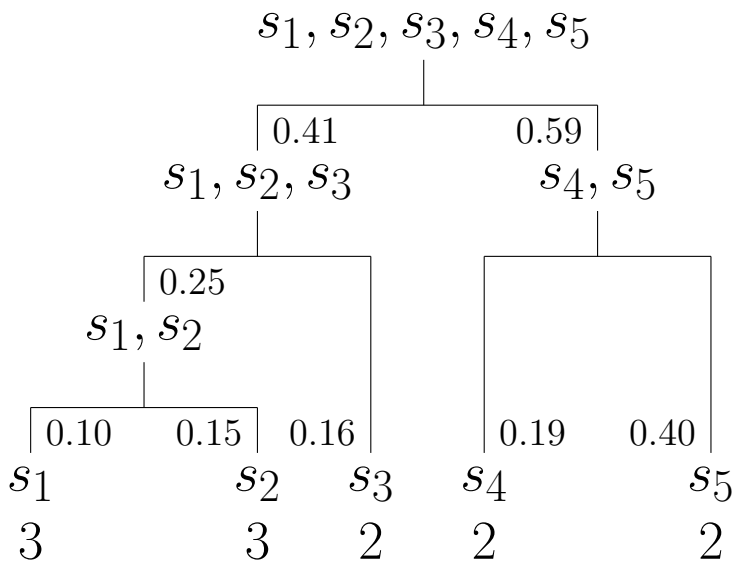
Always combine those two sets that have the smallest probabilities.

Question/Coding Schemes

$$P(s_1) = 0.10, \quad P(s_2) = 0.15, \quad P(s_3) = 0.16, \quad P(s_4) = 0.19, \quad P(s_5) = 0.40$$

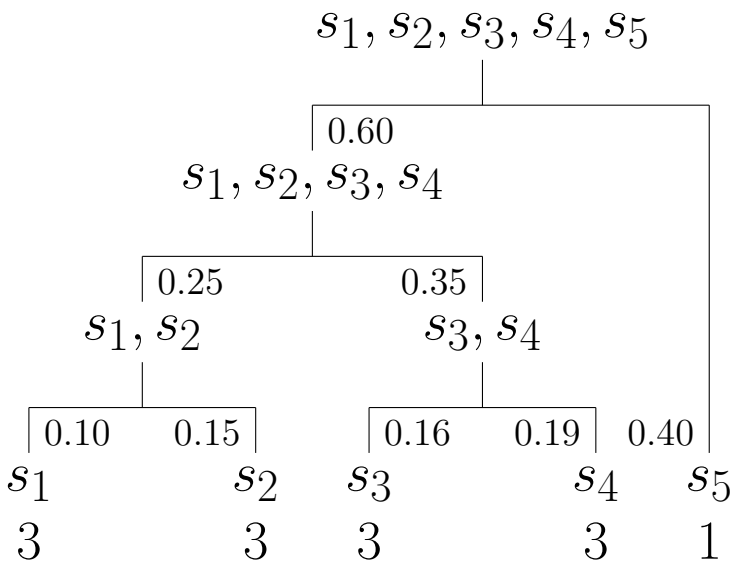
Shannon entropy: $-\sum_i P(s_i) \log_2 P(s_i) = 2.15$ bit/symbol

Shannon–Fano Coding (1948)



Code length: 2.25 bit/symbol
Code efficiency: 0.955

Huffman Coding (1952)



Code length: 2.20 bit/symbol
Code efficiency: 0.977

Question/Coding Schemes

It can be shown that Huffman coding is optimal if we have to determine the obtaining alternative in a single instance.
(No question/coding scheme has a smaller expected number of questions.)

Only if the obtaining alternative has to be determined in a sequence of (independent) situations, this scheme can be improved upon.

Idea: Process the sequence not instance by instance, but combine two, three or more consecutive instances and ask directly for the obtaining combination of alternatives.

Although this enlarges the question/coding scheme, the expected number of questions per identification is reduced (because each interrogation identifies the obtaining alternative for several situations).

However, the expected number of questions per identification cannot be made arbitrarily small. Shannon showed that there is a lower bound, namely the Shannon entropy.

Interpretation of Shannon Entropy

$$P(s_1) = \tfrac{1}{2}, \quad P(s_2) = \tfrac{1}{4}, \quad P(s_3) = \tfrac{1}{8}, \quad P(s_4) = \tfrac{1}{16}, \quad P(s_5) = \tfrac{1}{16}$$

Shannon entropy: $-\sum_i P(s_i) \log_2 P(s_i) = 1.875$ bit/symbol

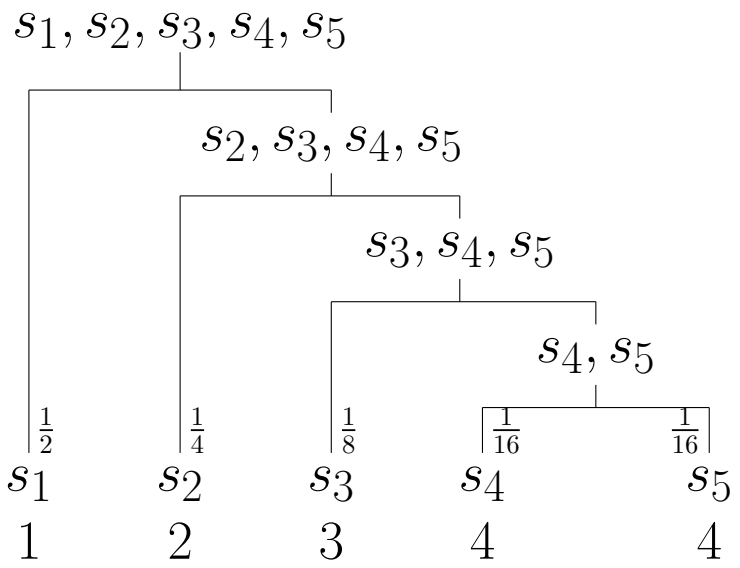
If the probability distribution allows for a perfect Huffman code (code efficiency 1), the Shannon entropy can easily be interpreted as follows:

$$-\sum_i P(s_i) \log_2 P(s_i)$$

$$= \sum_i \underbrace{P(s_i)}_{\substack{\text{occurrence} \\ \text{probability}}} \cdot \underbrace{\log_2 \frac{1}{P(s_i)}}_{\substack{\text{path length} \\ \text{in tree}}} .$$

In other words, it is the expected number of needed yes/no questions.

Perfect Question Scheme



Code length: 1.875 bit/symbol
Code efficiency: 1

Normalized Information Gain

Information gain is biased towards many-valued attributes.

Normalization removes / reduces this bias.

Information Gain Ratio (Quinlan 1986 / 1993)

$$I_{\text{gr}}(C, A) = \frac{I_{\text{gain}}(C, A)}{H_A} = \frac{I_{\text{gain}}(C, A)}{-\sum_{j=1}^{n_A} p_{\cdot j} \log_2 p_{\cdot j}}$$

Symmetric Information Gain Ratio (López de Mántaras 1991)

$$I^{(1)}_{\text{sgr}}(C, A) = \frac{I_{\text{gain}}(C, A)}{H_{AC}} \qquad \text{or} \qquad I^{(2)}_{\text{sgr}}(C, A) = \frac{I_{\text{gain}}(C, A)}{H_A + H_C}$$

Bias of Information Gain

Information gain is biased towards many-valued attributes,
i.e., of two attributes having about the same information content it tends to select the one having more values.

The reasons are quantization effects caused by the finite number of example cases (due to which only a finite number of different probabilities can result in estimations) in connection with the following theorem:

Theorem: Let A, B, and C be three attributes with finite domains and let their joint probability distribution be strictly positive, i.e., $\forall a \in \mathrm{dom}(A) : \forall b \in \mathrm{dom}(B) : \forall c \in \mathrm{dom}(C) : P(A = a, B = b, C = c) > 0$. Then

$$I_{\mathrm{gain}}(C, AB) \;\geq\; I_{\mathrm{gain}}(C, B),$$

with equality obtaining only if the attributes C and A are conditionally independent given B, i.e., if $P(C = c \mid A = a, B = b) = P(C = c \mid B = b)$.

(A detailed proof of this theorem can be found, for example, in [Borgelt and Kruse 2002], p. 311ff.)

χ^2 Measure

Compares the actual joint distribution
with a **hypothetical independent distribution**.

Uses absolute comparison.

Can be interpreted as a difference measure.

$$\chi^2(C, A) = \sum_{i=1}^{n_C} \sum_{j=1}^{n_A} N_{\cdot\cdot} \frac{(p_{i\cdot} p_{\cdot j} - p_{ij})^2}{p_{i\cdot} p_{\cdot j}}$$

Side remark: Information gain can also be interpreted as a difference measure.

$$I_{\text{gain}}(C, A) = \sum_{i=1}^{n_C} \sum_{j=1}^{n_A} p_{ij} \log_2 \frac{p_{ij}}{p_{i\cdot} p_{\cdot j}}$$

Treatment of Numeric Attributes

General Approach: Discretization

Preprocessing I

Form equally sized or equally populated intervals.

During the tree construction

Sort the example cases according to the attribute's values.

Construct a binary symbolic attribute for every possible split (values: "$\leq$ threshold" and "$>$ threshold").

Compute the evaluation measure for these binary attributes.

Possible improvements: Add a penalty depending on the number of splits.

Preprocessing II / Multisplits during tree construction

Build a decision tree using only the numeric attribute.

Flatten the tree to obtain a multi-interval discretization.

Treatment of Missing Values

Induction

Weight the evaluation measure with the fraction of cases with known values.

Idea: The attribute provides information only if it is known.

Try to find a surrogate test attribute with similar properties
(CART, Breiman *et al.* 1984)

Assign the case to all branches, weighted in each branch with the relative frequency of the corresponding attribute value (C4.5, Quinlan 1993).

Classification

Use the surrogate test attribute found during induction.

Follow all branches of the test attribute, weighted with their relative number of cases, aggregate the class distributions of all leaves reached, and assign the majority class of the aggregated class distribution.

Pruning Decision Trees

Pruning serves the purpose

to simplify the tree (improve interpretability),

to avoid overfitting (improve generalization).

Basic ideas:

Replace "bad" branches (subtrees) by leaves.

Replace a subtree by its largest branch if it is better.

Common approaches:

Reduced error pruning

Pessimistic pruning

Confidence level pruning

Minimum description length pruning

Reduced Error Pruning

Classify a set of new example cases with the decision tree.
(These cases must not have been used for the induction!)

Determine the number of errors for all leaves.

The number of errors of a subtree is the sum of the errors of all of its leaves.

Determine the number of errors for leaves that replace subtrees.

If such a leaf leads to the same or fewer errors than the subtree,
replace the subtree by the leaf.

If a subtree has been replaced,
recompute the number of errors of the subtrees it is part of.

Advantage: Very good pruning, effective avoidance of overfitting.

Disadvantage: Additional example cases needed.

Pessimistic Pruning

Classify a set of example cases with the decision tree.
(These cases may or may not have been used for the induction.)

Determine the number of errors for all leaves and
increase this number by a fixed, user-specified amount r.

The number of errors of a subtree is the sum of the errors of all of its leaves.

Determine the number of errors for leaves that replace subtrees
(also increased by r).

If such a leaf leads to the same or fewer errors than the subtree,
replace the subtree by the leaf and recompute subtree errors.

Advantage: No additional example cases needed.

Disadvantage: Number of cases in a leaf has no influence.

Confidence Level Pruning

Like pessimistic pruning, but the number of errors is computed as follows:

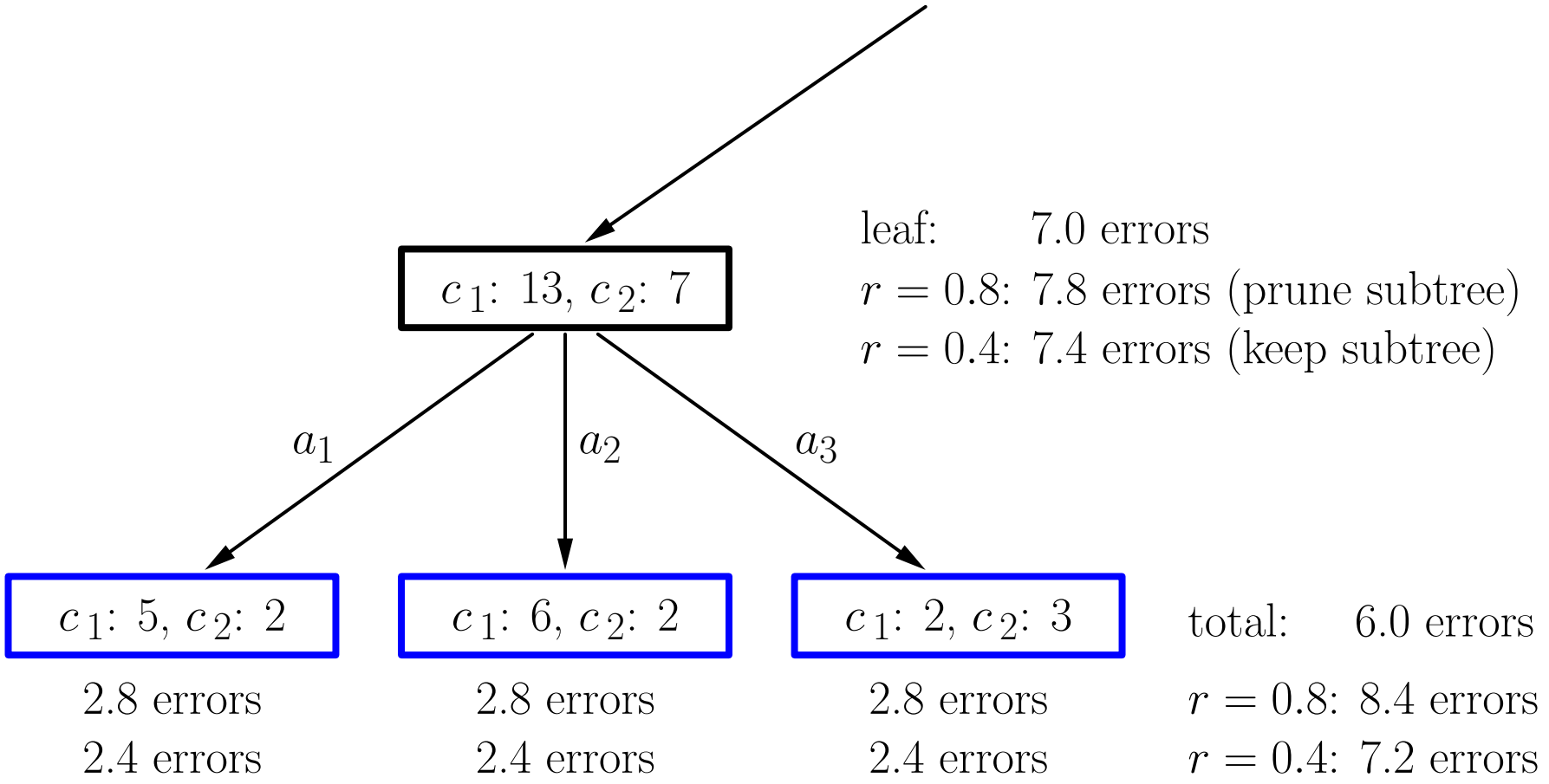
1. See classification in a leaf as a Bernoulli experiment (error / no error).
2. Estimate an interval for the error probability based on a user-specified confidence level α.
   (use approximation of the binomial distribution by a normal distribution)
3. Increase error number to the upper level of the confidence interval
   times the number of cases assigned to the leaf.
4. Formal problem: Classification is not a random experiment.

Advantage: No additional example cases needed, good pruning.

Disadvantage: Statistically dubious foundation.

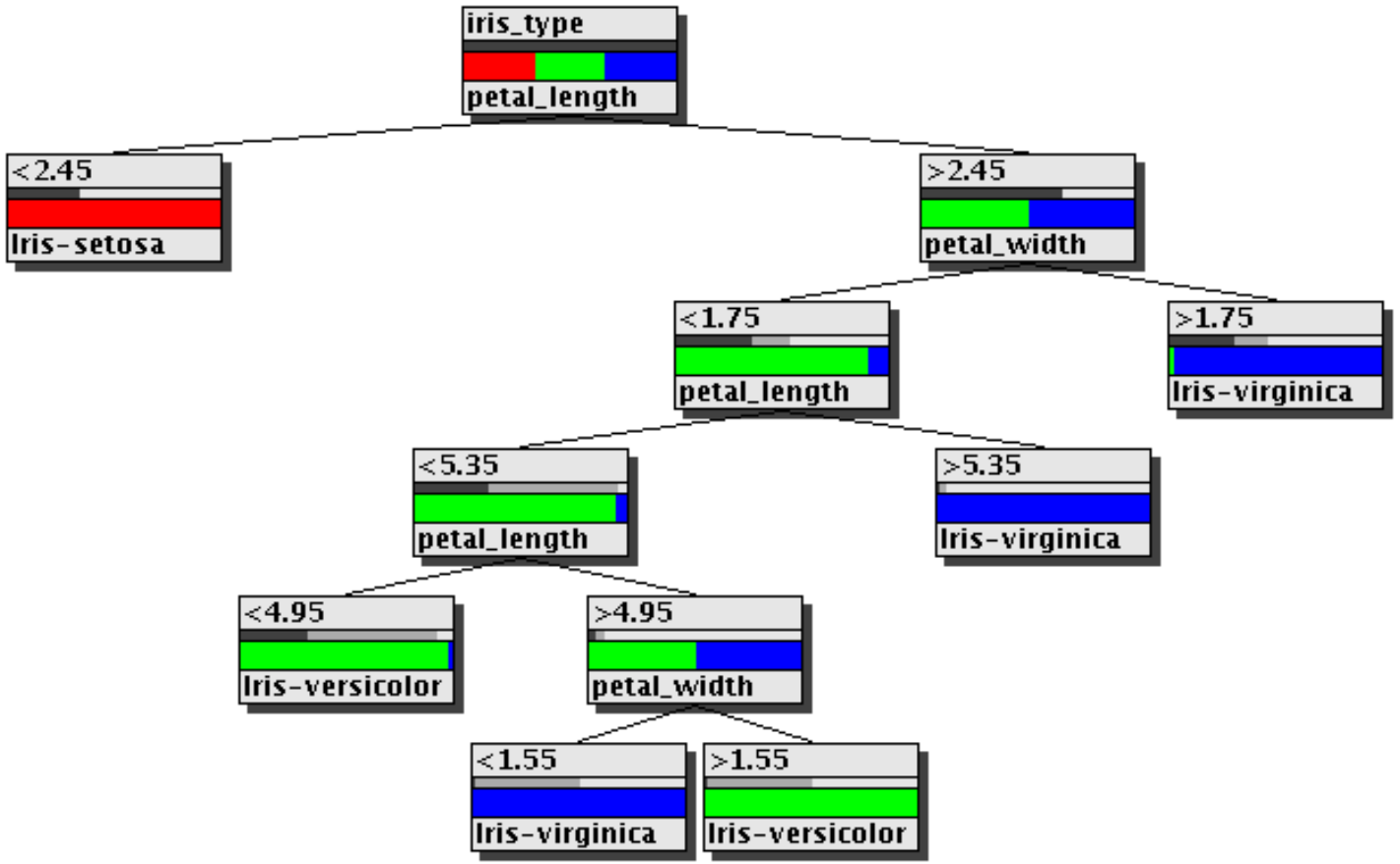
Pruning a Decision Tree: A Simple Example

Pessimistic Pruning with $r = 0.8$ and $r = 0.4$:



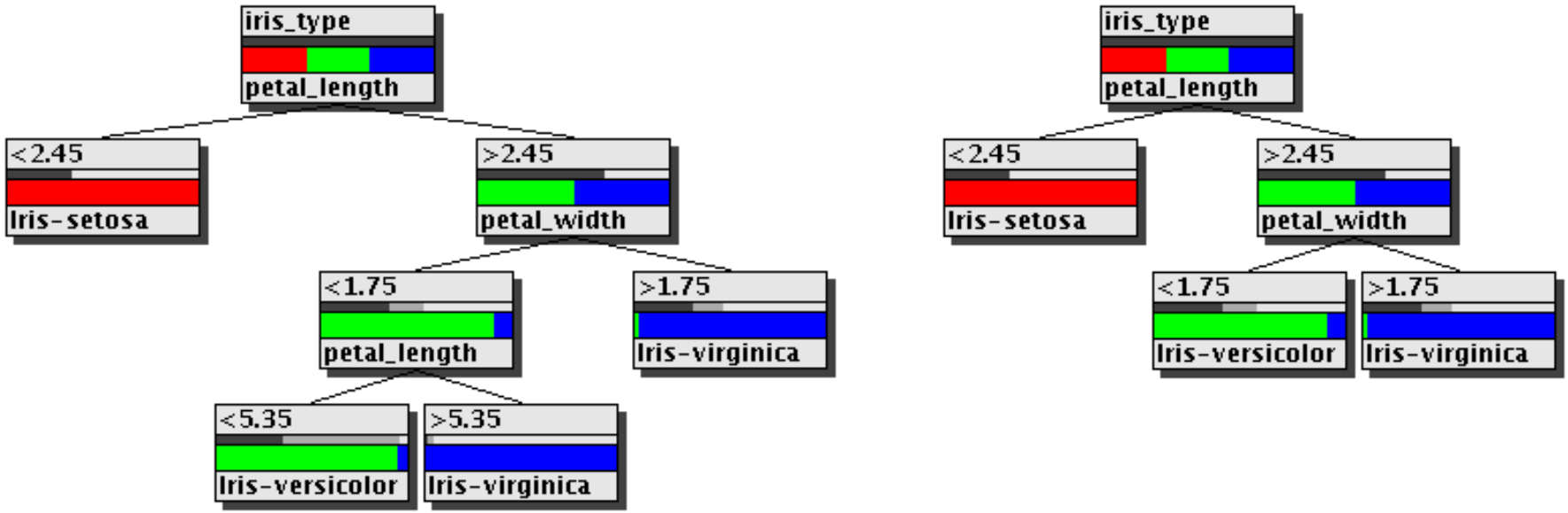
Decision Trees: An Example

A decision tree for the Iris data
(induced with information gain ratio, unpruned)



Decision Trees: An Example

A decision tree for the Iris data
(pruned with confidence level pruning, $\alpha = 0.8$, and pessimistic pruning, $r = 2$)



Left: 7 instead of 11 nodes, 4 instead of 2 misclassifications.

Right: 5 instead of 11 nodes, 6 instead of 2 misclassifications.

The right tree is "minimal" for the three classes.

Summary Decision Trees

Decision Trees are Classifiers with Tree Structure

Inner node: Test of a descriptive attribute

Leaf node: Assignment of a class

Induction of Decision Trees from Data

(Top-Down Induction of Decision Trees, TDIDT)

Divide and conquer approach / *recursive descent*

Greedy selection of the test attributes

Attributes are selected based on an *evaluation measure*, e.g. information gain, χ^2 measure

Recommended: *Pruning* of the decision tree

Classification Evaluation: Cross Validation

General method to evaluate / predict the performance of classifiers.

Serves the purpose to estimate the error rate on new example cases.

Procedure of cross validation:

1. Split the given data set into n so-called *folds* of equal size (n-fold cross validation).
2. Combine $n - 1$ folds into a training data set, build a classifier, and test it on the n-th fold.
3. Do this for all n possible selections of $n - 1$ folds and average the error rates.

Special case: Leave-1-out cross validation.
(use as many folds as there are example cases)

Final classifier is learned from the full data set.

Support Vector Machines

Supervised Learning, Diagnosis System for Diseases

Training data: Expression profiles of patients with known diagnosis

The known diagnosis gives us a structure within the data, which we want to generalize for future data.

Learning/Training: Derive a decision rule from the training data which separates the two classes.

Ability for generalization: How useful is the decision rule when it comes to diagnosing patients in the future?

Aim: Find a decision rule with high ability for generalization!

Learning from Examples

Given: $X = \{x_i, y_i\}_{i=1}^n$, training data of patients with known diagnosis

consists of:

$$x_i \in \mathbb{R}^g \text{ (points, expression profiles)}$$

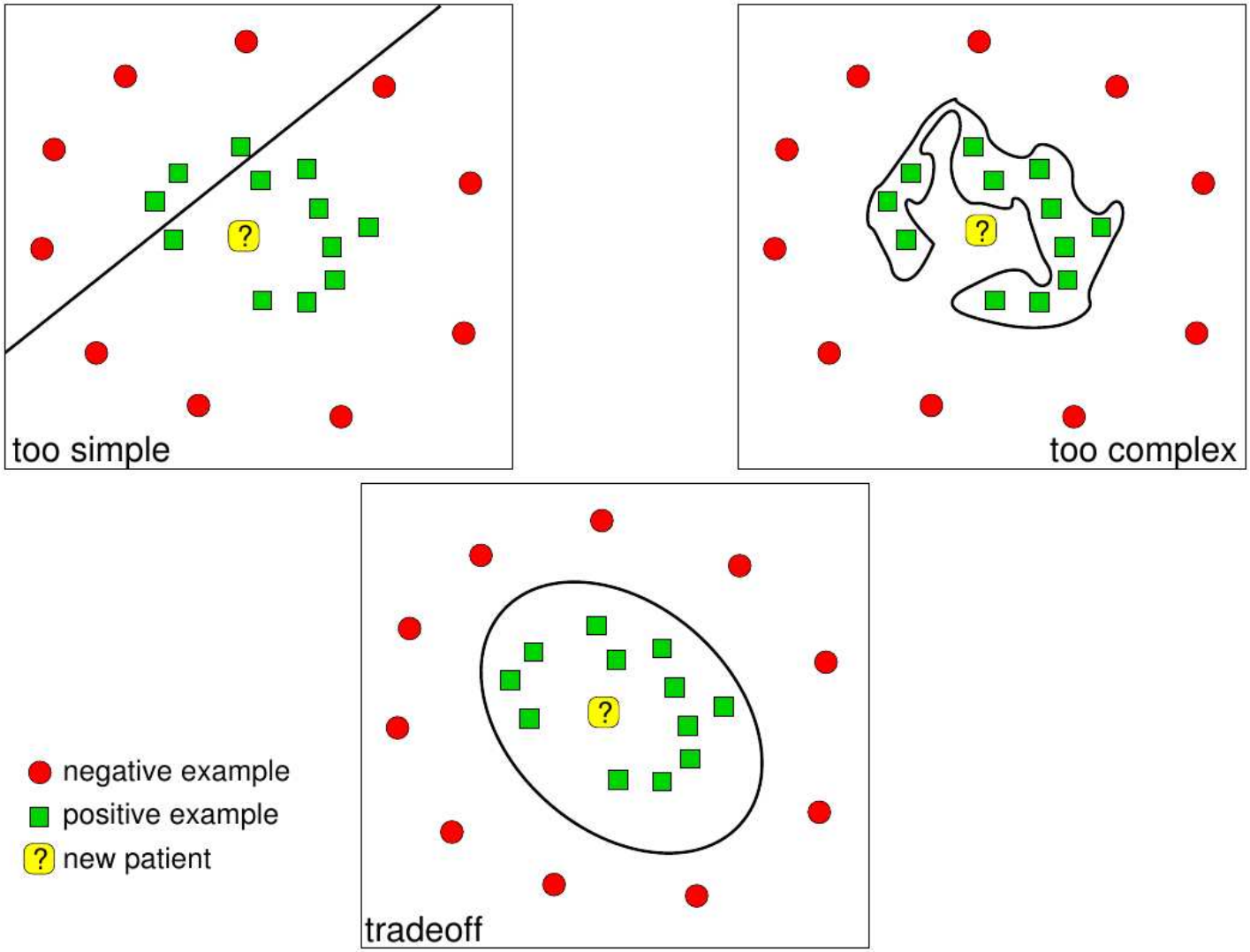
$$y_i \in \{+1, -1\} \text{ (classes, 2 kinds of cancer)}$$

Decision function:

$$f_X : \mathbb{R}^g \to \{+1, -1\}$$

$$\text{diagnosis} = f_X(\text{new patient})$$

Underfitting / Overfitting



Linear Separation of Training Data

Begin with linear separation and increase the complexity in a second step with a kernel function.

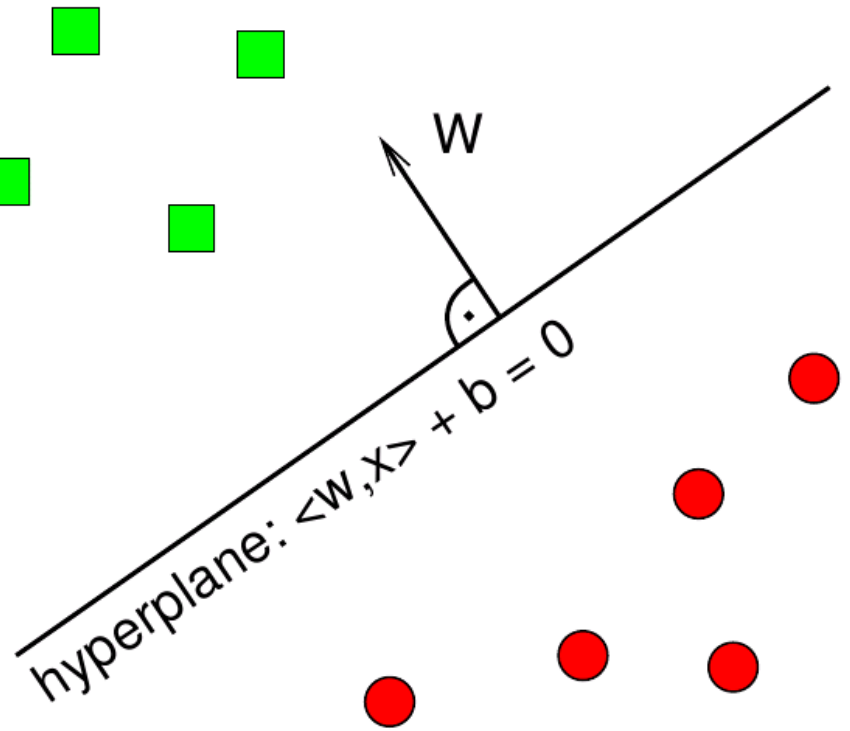
A separating hyperplane is defined by

a normal vector w and

an offset b:

Hyperplane $\mathcal{H} = \{x | \langle w, x \rangle + b = 0\}$

$\langle \cdot, \cdot \rangle$ is called the inner product
or scalar product.



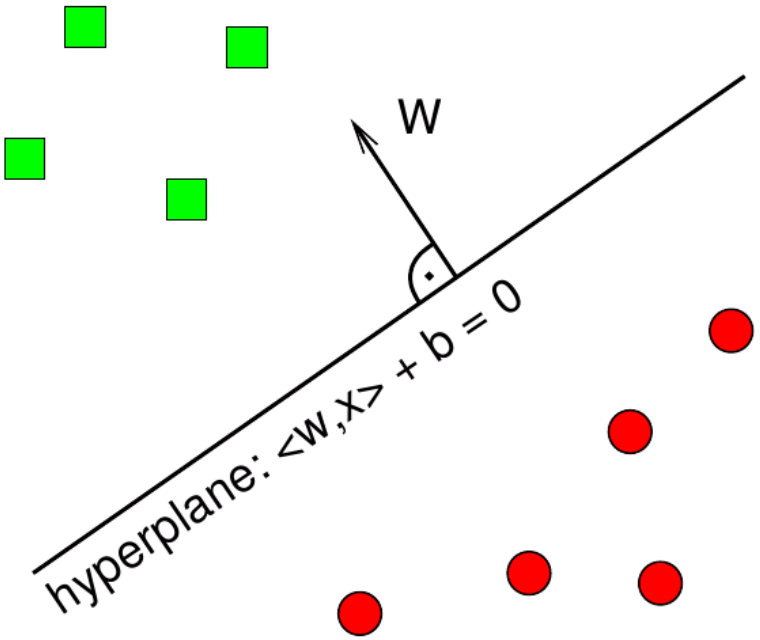
Predicting the class of a new point

Training: Choose w and b in such a way that the hyperplane separates the training data.

Prediction: Which side of the hyperplane is the new point located on?

Points on the side that the normal vector points at are diagnosed as POSITIVE.

Points on the other side are diagnosed as NEGATIVE.



Motivation

Origin in Statistical Learning Theory; class of optimal classifiers

Core problem of Statistical Learning Theory: Ability for generalization.
When does a low training error lead to a low real error?

Binary Class Problem:
Classification $\equiv$ mapping function $f(x, u): \ x \to y \in \{+1, -1\}$
x: sample from one of the two classes
u: parameter vector of the classifier

Learning sample with l observations $x_1, x_2, \ldots, x_l$
along with their class affiliation $y_1, y_2, \ldots, y_l$
$\to$ the empirical risk (error rate) for a given training dataset:

$$R_{emp}(u) \ = \ \frac{1}{2l} \sum_{i=1}^{l} |y_i - f(x_i, u)| \ \in [0, 1]$$

A lot of classifiers do minimize the empirical risk, e.g. Neural Networks.

Motivation

Expected value of classification error (expected risk):

$$R(u) = E\{R_{test}(u)\} = E\{\frac{1}{2}|y - f(x,u)|\} = \int \frac{1}{2}|y - f(x,u)|p(x,y)\, dxdy$$

$p(x,y)$: Distribution density of all possible samples x along with their class affiliation y (Can't evaluate this expression directly as $p(x,y)$ is not available.)

Optimal sample classification:
Search for deterministic mapping function $f(x,u) : x \to y \in \{+1,-1\}$ that minimizes the expected risk.

Core question of sample classification:

> How close do we get to the *real* error after we saw l training samples? How well can we estimate the real risk $R(u)$ from the empirical risk $R_{emp}(u)$? (Structural Risk Minimization instead of Empirical Risk Minimization)

The answer is given by Learning Theory of Vapnik-Chervonenkis $\to$ SVMs

SVMs for linear separable classes

Previous solution:

General hyperplane: $wx + b = 0$

Classification: $\text{sgn}(wx + b)$

Training, e.g. by perceptron-algorithm
(iterative learning, correction after every misclassification; no unique solution)

Reminder: Function Optimization

Task: Find values $\vec{x} = (x_1, \ldots, x_m)$ such that $f(\vec{x}) = f(x_1, \ldots, x_m)$ is optimal.

Often feasible approach:

A necessary condition for a (local) optimum (maximum or minimum) is that the partial derivatives w. r. t. the parameters vanish (Pierre Fermat).

Therefore: (Try to) solve the equation system that results from setting all partial derivatives w. r. t. the parameters equal to zero.

Example task: Minimize $f(x, y) = x^2 + y^2 + xy - 4x - 5y$.

Solution procedure:

Take the partial derivatives of the objective function and set them to zero:

$$\frac{\partial f}{\partial x} = 2x + y - 4 = 0, \qquad \frac{\partial f}{\partial y} = 2y + x - 5 = 0.$$

Solve the resulting (here: linear) equation system: $x = 1, \quad y = 2.$

Function Optimization with Constraints

Often a function has to be optimized subject to certain **constraints**.

Here: restriction to k **equality constraints** $C_i(\vec{x}) = 0$, $i = 1, \ldots, k$.

Note: the equality constraints describe a subspace of the domain of the function.

Problem of optimization with constraints:

> The gradient of the objective function f may vanish outside the constrained subspace, leading to an unacceptable solution (violating the constraints).
>
> At an optimum *in the constrained subspace* the derivatives need not vanish.

One way to handle this problem are **generalized coordinates**:

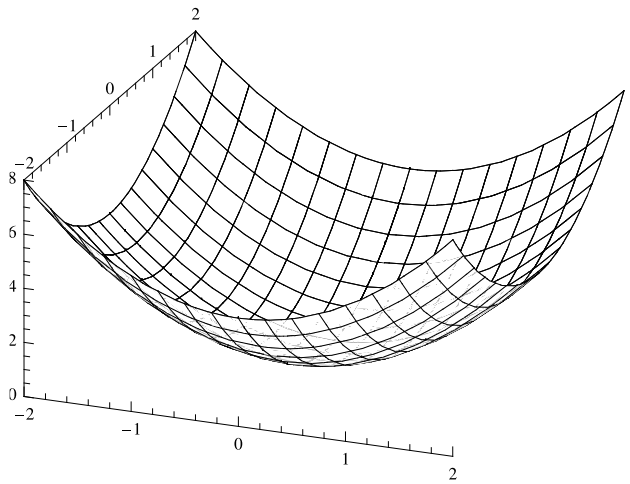
> Exploit the dependence between the parameters specified in the constraints to express some parameters in terms of the others and thus reduce the set $\vec{x}$ to a set $\vec{x}'$ of independent parameters (*generalized coordinates*).
>
> Problem: Can be clumsy and cumbersome, if possible at all, because the form of the constraints may not allow for expressing some parameters as proper functions of the others.

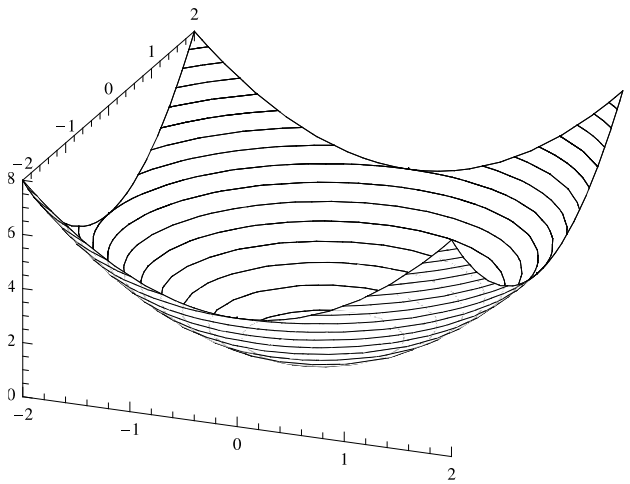
Contour Lines of a Function

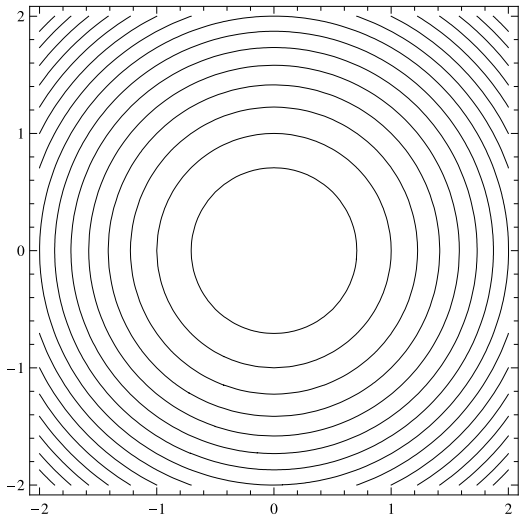
Contour Lines: Given a function $f : \mathbb{R}^2 \to \mathbb{R}$, the contour plot is obtained by drawing the contour sets for equidistant levels, i. e., plot the following sets of points:

$$M_{kc} = \{(x_1, x_2) \in \mathbb{R}^2 \mid f(x_1, x_2) = kc\}$$
$$\text{for } k \in \mathbb{N} \text{ and fixed } c \in \mathbb{R}_{\geq 0}$$

Example: $f(x_1, x_2) = x_1^2 + x_2^2$







Gradient Field of a Function

The gradient of a function $f\colon \mathbb{R}^n \to \mathbb{R}$ consists of the vector of its partial derivatives w. r. t. the arguments:

$$\nabla_{\vec{x}} f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)^\top$$

The gradient evaluated at a point $\vec{x}^*$, written as

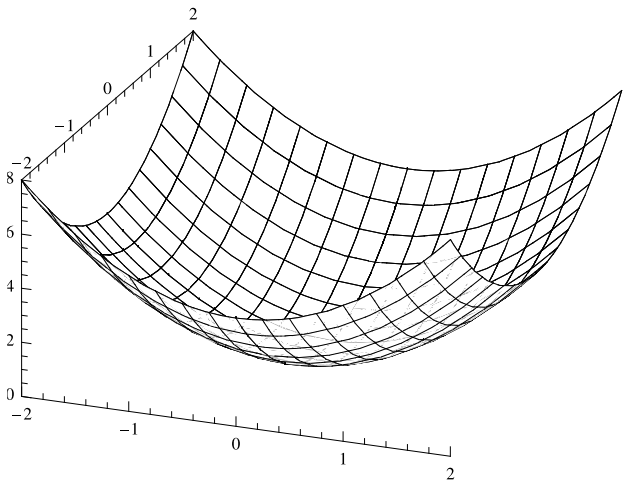
$$\nabla_{\vec{x}} f|_{\vec{x}^*} = \left(\frac{\partial f}{\partial x_1}\bigg|_{x_1^*}, \ldots, \frac{\partial f}{\partial x_n}\bigg|_{x_n^*} \right)^\top,$$

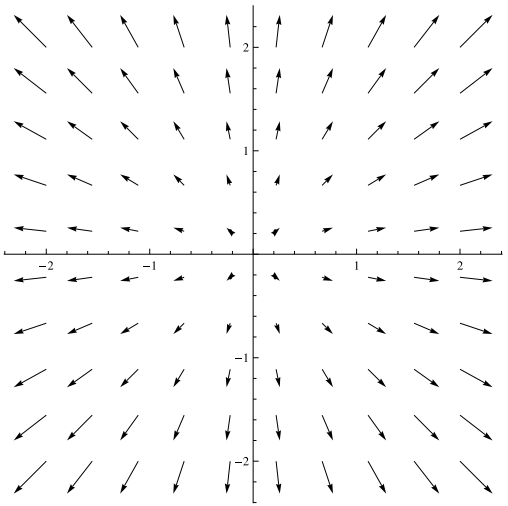
points into the direction of largest increase of f.

Formally, the gradient of a function with domain $\mathbb{R}^n$ has n dimensions although it is often depicted as an $n+1$-dimensional vector.

Gradient Field of a Function: Examples

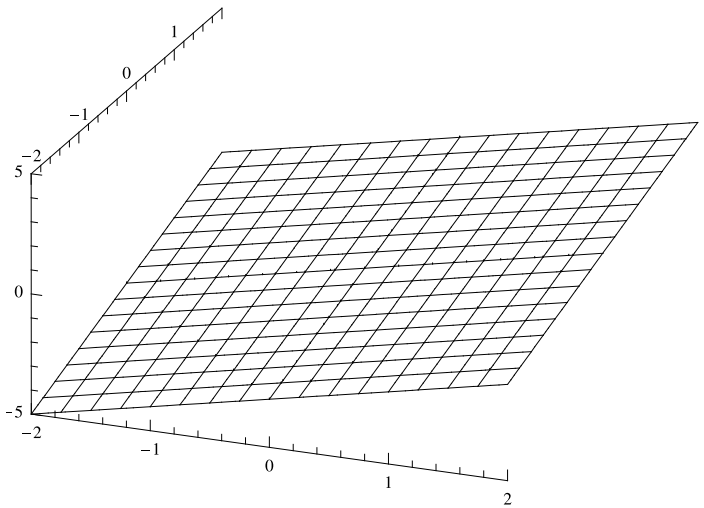
$$f_1(x_1, x_2) = x_1^2 + x_2^2$$

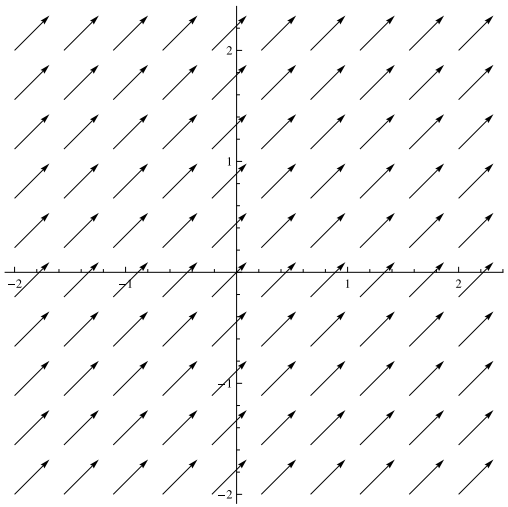




$$\nabla_{\vec{x}} f_1 = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix}$$

$$f_2(x_1, x_2) = x_1 + x_2 - 1$$





$$\nabla_{\vec{x}} f_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Function Optimization with Constraints

Problem: If the global optimum of f lies outside the feasible region the gradient does not vanish at the constrained optimum $\vec{x}^*$.

Which criteria do hold at the constrained optimum?

Assume we move $\vec{x}^*$ throughout the feasible region to find the optimum "manually". If we *cross* a contour line of f, the crossing point cannot be an optimum: because crossing a contour line means descending or ascending.

However, if we *touch* a contour line we have found an optimum because stepping backward or forward will increase (or decrease) the value.

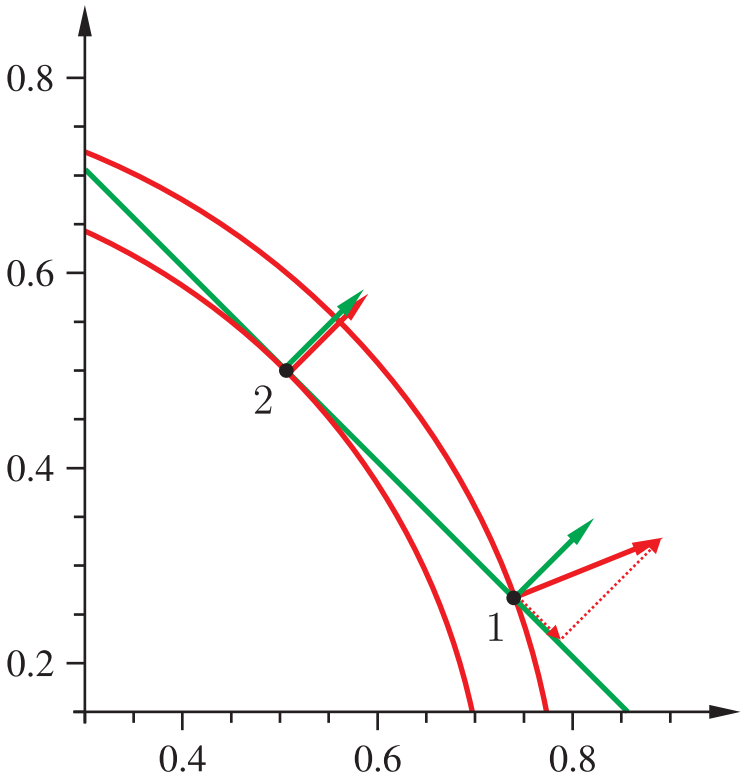
At the "touching point" $\vec{x}^*$ the gradient of f and the gradient of g are parallel.

$$\nabla f = \lambda \nabla g$$

We only need both gradients to be parallel. Since they can have opposite directions and different lengths λ is used to rescale ∇g.

Example

Task: Minimize $f(x_1, x_2) = x_1^2 + x_2^2$ subject to $g\colon x + y = 1$.



Crossing a contour line: Point 1 cannot be a constrained minimum because ∇f has a non-zero component in the constrained space. Walking in opposite direction to this component can further decrease f.

Touching a contour line: Point 2 is a constrained minimum: both gradients are parallel, hence there is no component of ∇f in the constrained space that might lead us to a lower value of f.

Function Optimization with Constraints

Therefore, at the constrained optimum $\vec{x}^*$ we require:

$$\nabla f(\vec{x}^*) = \lambda \nabla g(\vec{x}^*) \qquad \text{and} \qquad g(\vec{x}^*) = 0$$

More compact representation:

$$L(\vec{x}, \lambda) = f(\vec{x}) - \lambda g(\vec{x}) \qquad \text{and} \qquad \nabla L = 0$$

Taking the partial derivatives reveals the initial conditions:

$$\begin{aligned} \frac{\partial}{\partial \vec{x}} L(\vec{x}, \lambda) &= \nabla f(\vec{x}) - \lambda \nabla g(\vec{x}) = 0 \\ \nabla f(\vec{x}) &= \lambda \nabla g(\vec{x}) \end{aligned}$$

$$\frac{\partial}{\partial \lambda} L(\vec{x}, \lambda) = g(\vec{x}) = 0$$

The negative sign in the Lagrange function L can be incorporated into λ, i. e. we will from now on replace it by a positive sign.

Lagrange Theory: Example 1

Example task: Minimize $f(x, y) = x^2 + y^2$ subject to $x + y = 1$.

Solution procedure:

Rewrite the constraint, so that one side gets zero: $x + y - 1 = 0$.

Construct the Lagrange function by incorporating the constraint into the objective function with a Lagrange multiplier λ:

$$L(x, y, \lambda) = x^2 + y^2 + \lambda(x + y - 1).$$

Take the partial derivatives of the Lagrange function and set them to zero (necessary conditions for a minimum):

$$\frac{\partial L}{\partial x} = 2x + \lambda = 0, \qquad \frac{\partial L}{\partial y} = 2y + \lambda = 0, \qquad \frac{\partial L}{\partial \lambda} = x + y - 1 = 0.$$

Solve the resulting (here: linear) equation system:

$$\lambda = -1, \qquad x = y = \tfrac{1}{2}.$$

Summary: Function Optimization with Constraints

Let $\vec{x}^*$ be a (local) optimum of $f(\vec{x})$ *in the constrained subspace.* Then:

The gradient $\nabla_{\vec{x}} f(\vec{x}^*)$, if it does not vanish, must be **perpendicular** to the constrained subspace. (If $\nabla_{\vec{x}} f(\vec{x}^*)$ had a component in the constrained subspace, $\vec{x}^*$ would not be a (local) optimum in this subspace.)

The gradients $\nabla_{\vec{x}} g_j(\vec{x}^*)$, $1 \leq j \leq k$, must all be **perpendicular** to the constrained subspace, because they are constant, namely 0, in this subspace. Together they span the subspace perpendicular to the constrained subspace.

Therefore it must be possible to find values λ_j, $1 \leq j \leq k$, such that

$$\nabla_{\vec{x}} f(\vec{x}^*) + \sum_{j=1}^{s} \lambda_j \nabla_{\vec{x}} g_j(\vec{x}^*) = 0.$$

If the constraints (and thus their gradients) are linearly independent, the values λ_j are uniquely determined. This equation can be used to **compensate the gradient** of $f(\vec{x}^*)$ so that it vanishes at $\vec{x}^*$.

General Principle: Lagrange Theory

As a consequence of these insights we obtain the

Method of Lagrange Multipliers:

Given:
- a function $f(\vec{x})$, which is to be optimized,
- k equality constraints $g_j(\vec{x}) = 0$, $1 \leq j \leq k$.

Procedure:

Construct the so-called **Lagrange function** by incorporating the equality constraints g_i, $i = 1, \ldots, k$, with (unknown) **Lagrange multipliers** λ_i:

$$L(\vec{x}, \lambda_1, \ldots, \lambda_k) = f(\vec{x}) + \sum_{i=1}^{k} \lambda_i g_i(\vec{x}).$$

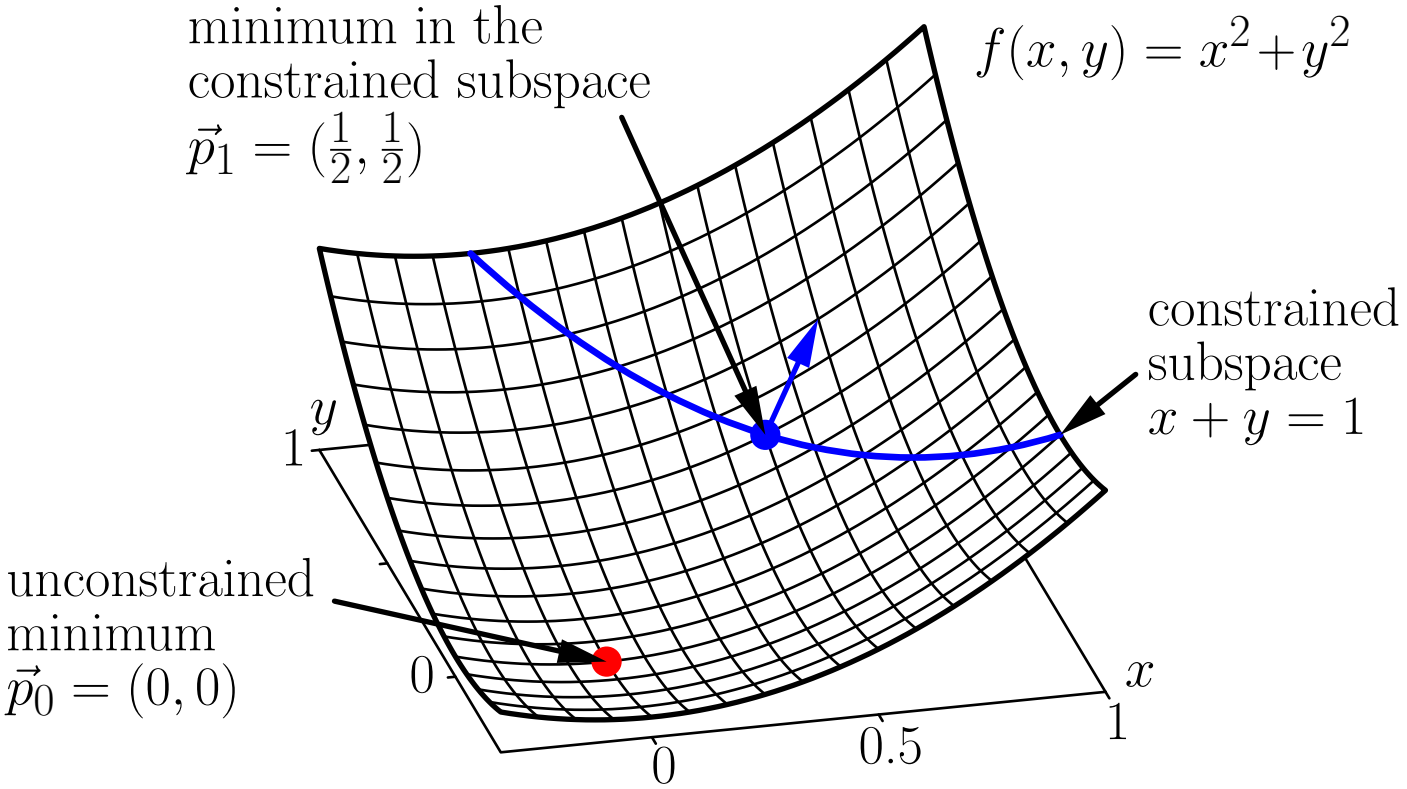
Set the partial derivatives of the Lagrange function equal to zero:

$$\frac{\partial L}{\partial x_1} = 0, \quad \ldots, \quad \frac{\partial L}{\partial x_m} = 0, \qquad \frac{\partial L}{\partial \lambda_1} = 0, \quad \ldots, \quad \frac{\partial L}{\partial \lambda_k} = 0.$$

(Try to) solve the resulting equation system.

Lagrange Theory: Revisited Example 1

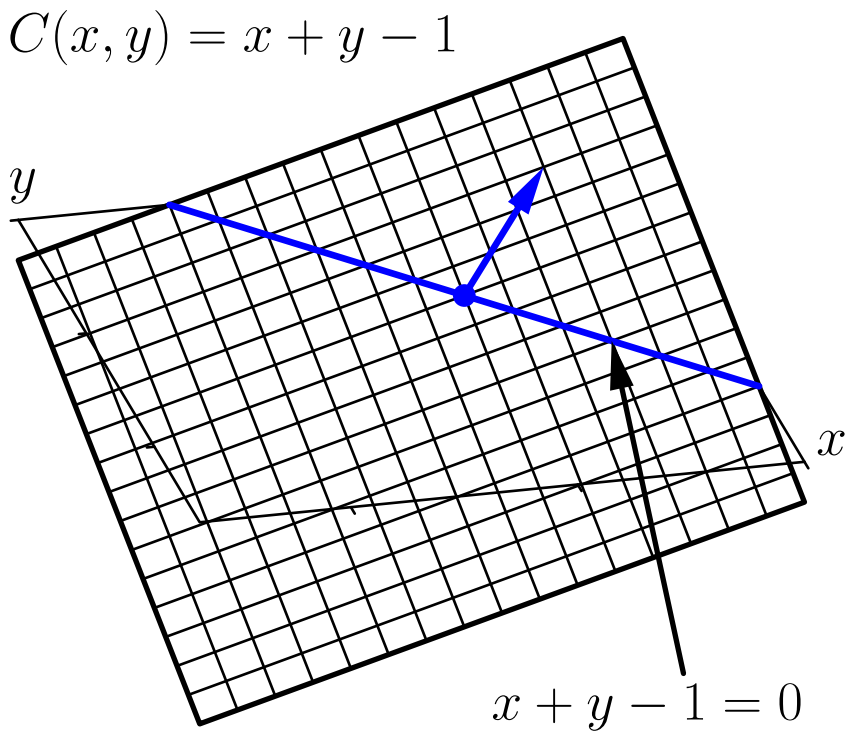
Example task: Minimize $f(x, y) = x^2 + y^2$ subject to $x + y = 1$.

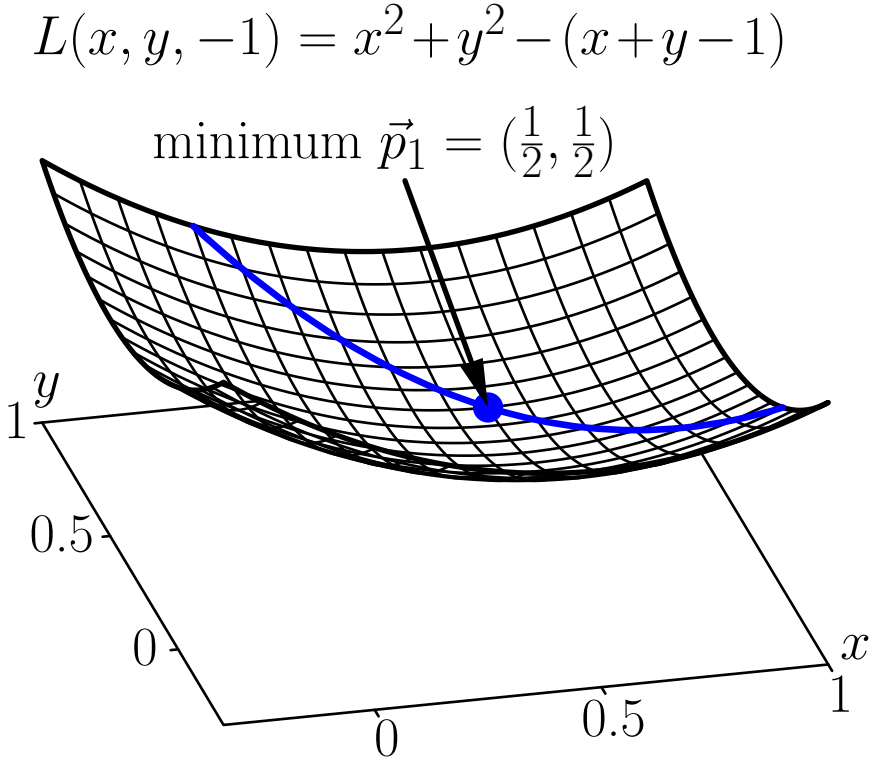


The unconstrained minimum is not in the constrained subspace, and at the minimum in the constrained subspace the gradient does not vanish.

Lagrange Theory: Revisited Example 1

Example task: Minimize $f(x, y) = x^2 + y^2$ subject to $x + y = 1$.





The gradient of the constraint is perpendicular to the constrained subspace.
The (unconstrained) minimum of the Lagrange function $L(x, y, \lambda)$
is the minimum of the objective function $f(x, y)$ in the constrained subspace.

Lagrange Theory: Example 2

Example task: Find the side lengths x, y, z of a box with maximum volume for a given area S of the surface.

Formally:

$$\begin{array}{ll} \text{Maximize} & f(x, y, z) = xyz \\ \text{subject to} & 2xy + 2xz + 2yz = S. \end{array}$$

Solution procedure:

The constraint is $C(x, y, z) = 2xy + 2xz + 2yz - S = 0$.

The Lagrange function is

$$L(x, y, z, \lambda) = xyz + \lambda(2xy + 2xz + 2yz - S).$$

Taking the partial derivatives yields (in addition to the constraint):

$$\frac{\partial L}{\partial x} = yz + 2\lambda(y+z) = 0, \quad \frac{\partial L}{\partial y} = xz + 2\lambda(x+z) = 0, \quad \frac{\partial L}{\partial y} = xy + 2\lambda(x+y) = 0.$$

The solution is: $\lambda = -\frac{1}{4}\sqrt{\frac{S}{6}}, \quad x = y = z = \sqrt{\frac{S}{6}}$ (i.e., the box is a cube).

Function Optimization: Lagrange Theory

Observations:

Due to the representation of the gradient of $f(\vec{x})$ at a local optimum $\vec{x}^*$ in the constrained subspace (see above) the gradient of L w.r.t. $\vec{x}$ vanishes at $\vec{x}^*$.

$\rightarrow$ The standard approach works again!

If the constraints are satisfied, the additional terms have no influence.

$\rightarrow$ The original task is not modified (same objective function).

Taking the partial derivative w.r.t. a Lagrange multiplier
reproduces the corresponding equality constraint:

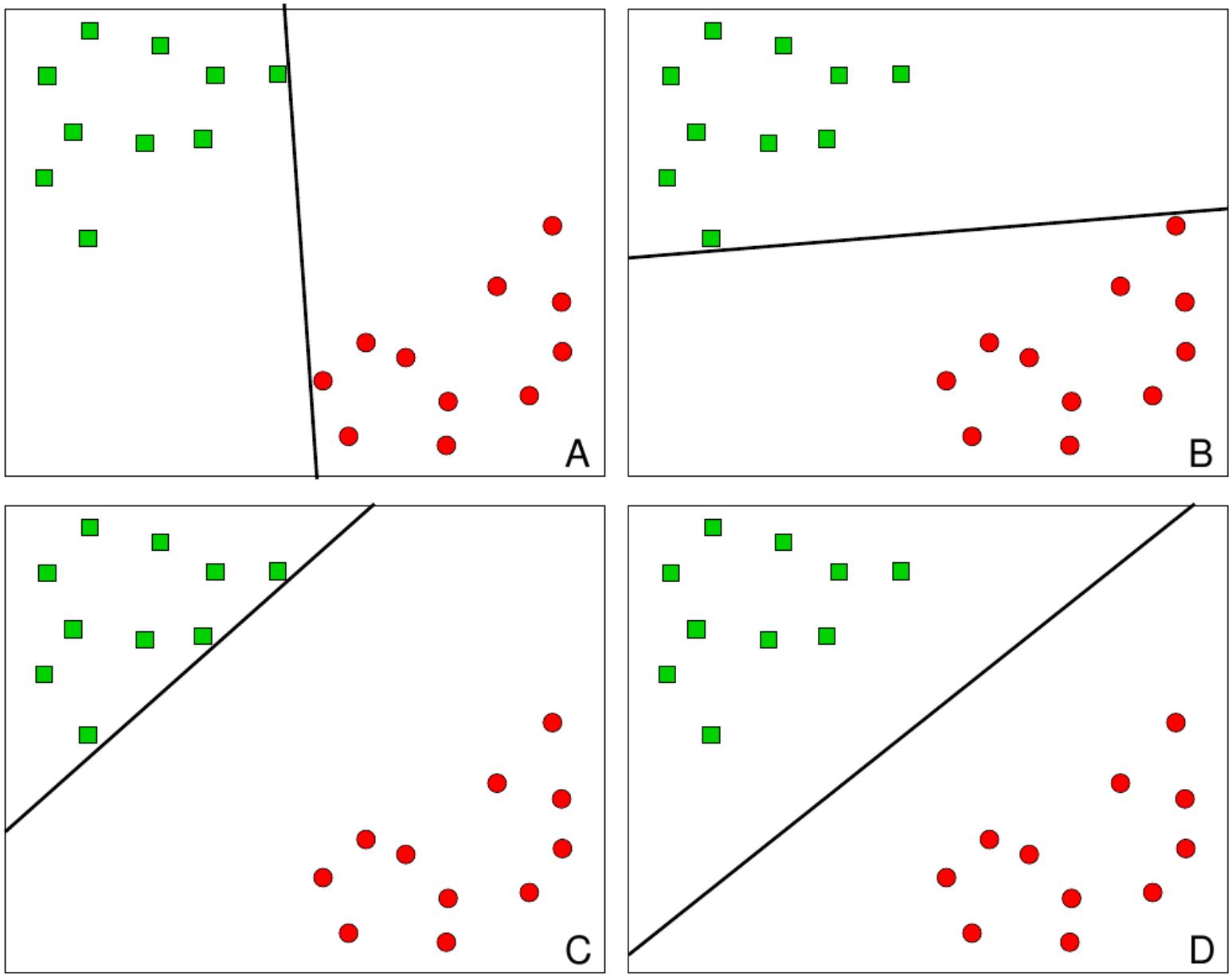
$$\forall j; 1 \leq j \leq k: \qquad \frac{\partial}{\partial \lambda_j} L(\vec{x}, \lambda_1, \ldots, \lambda_k) = C_j(\vec{x}),$$

$\rightarrow$ Constraints enter the equation system to solve in a natural way.

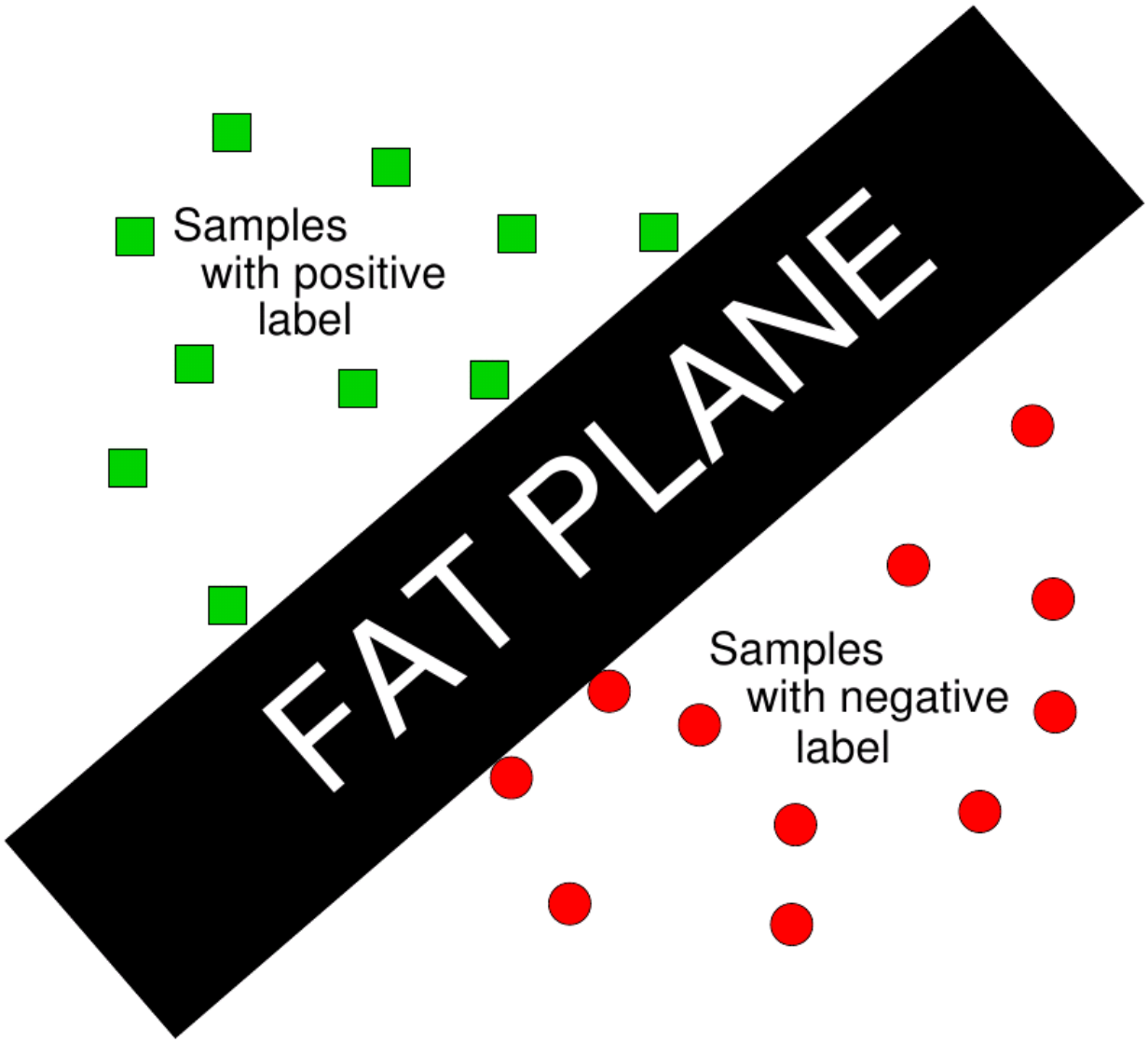
Remark:

Inequality constraints can be handled with the **Kuhn–Tucker theory**.

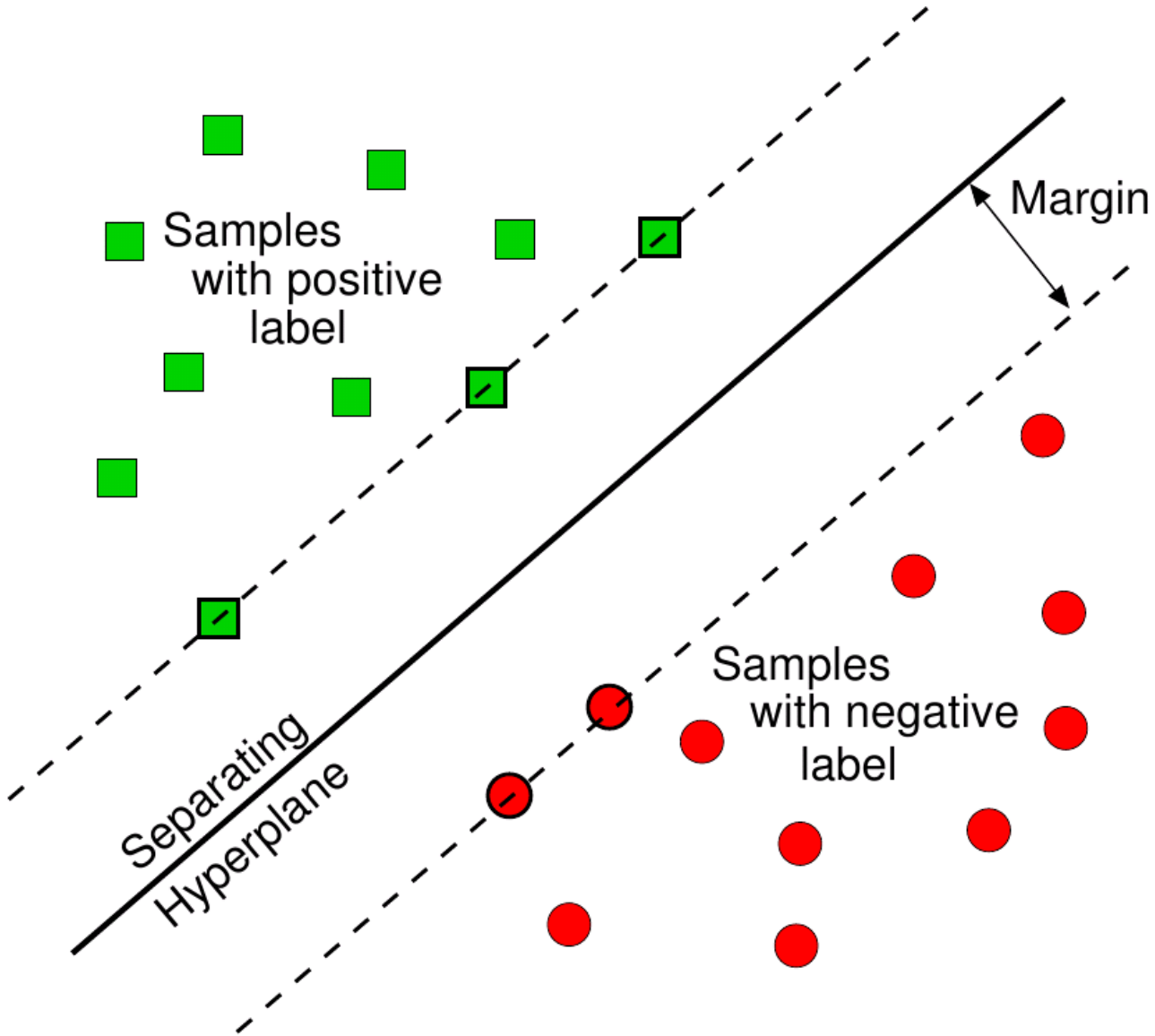
Which hyperplane is the best - and why?



No exact cut, but a . . .

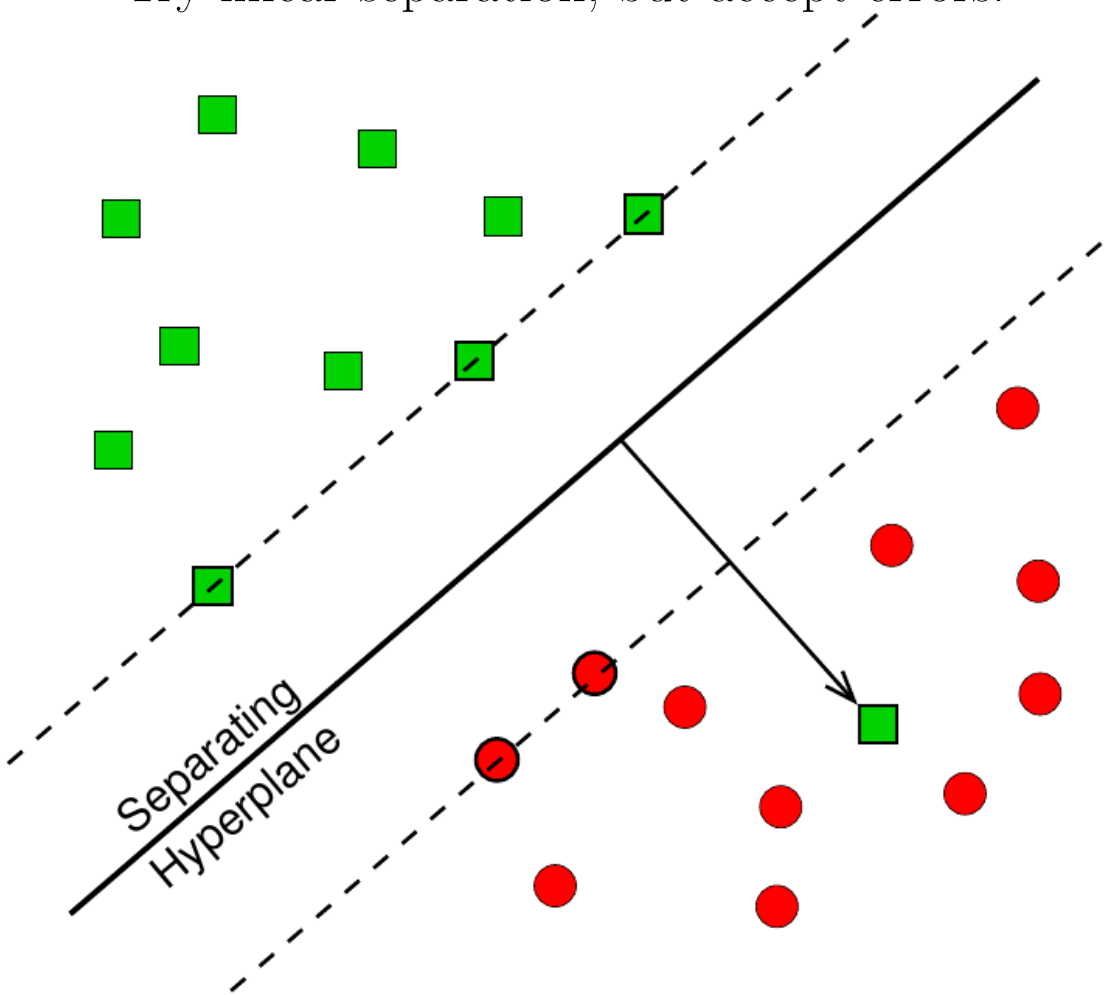


Separate the training data with maximal separation margin



Separate the training data with maximal separation margin

Try linear separation, but accept errors:



Penalty for errors: Distance to hyperplane times error weight C

SVMs for linearly separable classes

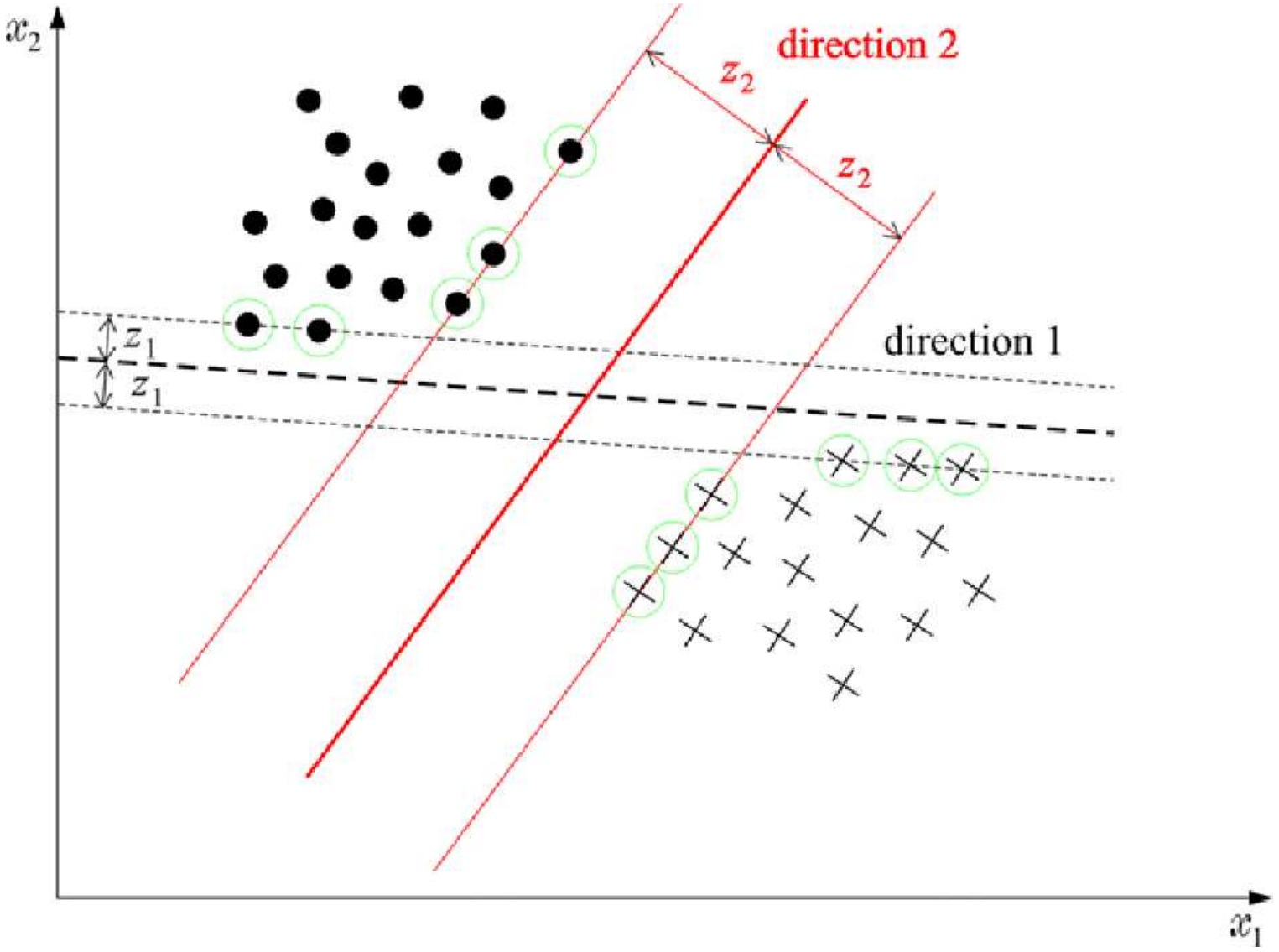
With SVMs we are searching for a separating hyperplane with maximal margin.
Optimum: The hyperplane with the highest 2δ of all possible separating hyperplanes.

This is intuitively meaningful
(At constant intra-class scattering, the confidence of right classification is growing with increasing inter-class distance)

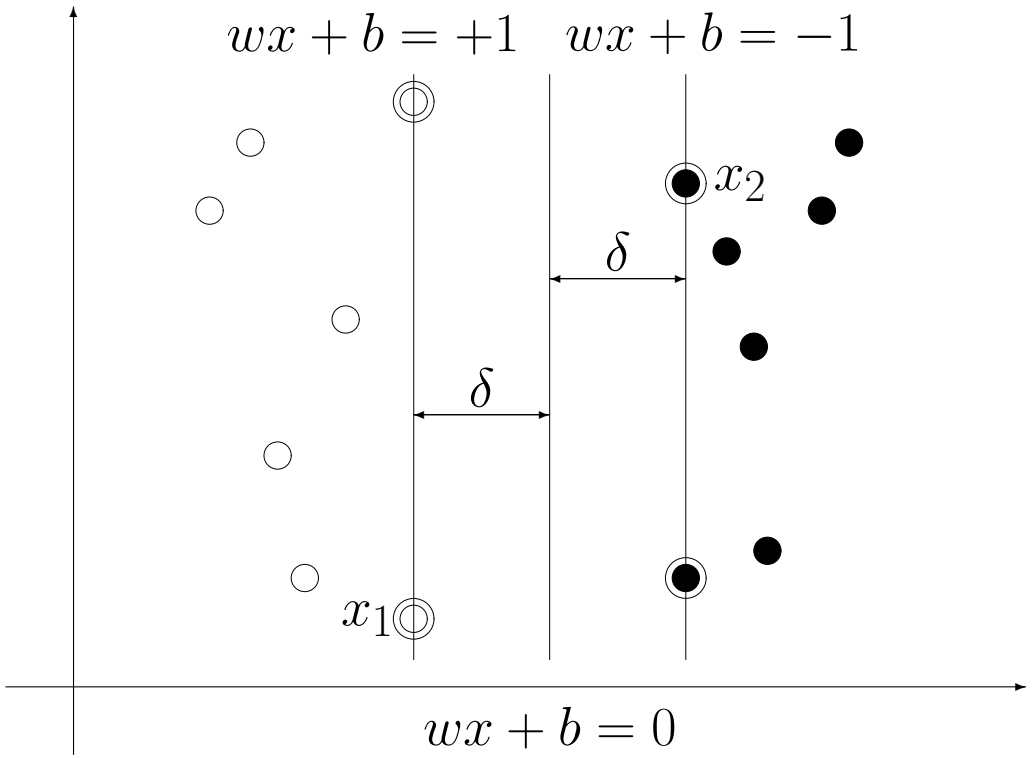
SVMs are theoretically justified by Statistical Learning Theory.

SVMs for linearly separable classes

Large-Margin Classifier: Separation line 2 is better than 1



SVMs for linearly separable classes



Training samples are classified correctly, if:

$$y_i(wx_i + b) > 0$$

Invariance of this expression towards a positive scaling leads to:

$$y_i(wx_i + b) \geq 1$$

with canonical hyperplanes:

$$\begin{cases} wx_i + b = +1; (\text{class with } y_i = +1) \\ wx_i + b = -1; (\text{class with } y_i = -1) \end{cases}$$

The distance between the canonical hyperplanes results from projecting $x_1 - x_2$ to the unit length normal vector $\frac{w}{||w||}$:

$$2\delta = \frac{2}{||w||}; \quad \text{d.h. } \delta = \frac{1}{||w||}$$

$\rightarrow$ maximizing $\delta \equiv$ minimizing $||w||^2$

SVMs for linearly separable classes

Optimal separating plane by minimizing a quadratic function to linear constraints:

Primal Optimization Problem:
minimize: $J(w,b) = \frac{1}{2}||w||^2$
to the constraints $\forall i \; [y_i(wx_i + b) \geq 1], \; i = 1, 2, \ldots, l$

Introducing a Lagrange-Function:

$$L(w,b,\alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \alpha_i [y_i(wx_i + b) - 1]; \quad \alpha_i \geq 0$$

leads to the *dual problem*:
maximize $L(w,b,\alpha)$ with respect to α, under the constraints:

$$\frac{\partial L(w,b,\alpha)}{\partial w} = 0 \quad \Longrightarrow \quad w = \sum_{i=1}^{l} \alpha_i y_i x_i$$

$$\frac{\partial L(w,b,\alpha)}{\partial b} = 0 \quad \Longrightarrow \quad \sum_{i=1}^{l} \alpha_i y_i = 0$$

Insert this terms in $L(w, b, \alpha)$:

$$
\begin{aligned}
L(w, b, \alpha) &= \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \alpha_i [y_i (w x_i + b) - 1] \\
&= \frac{1}{2} w \cdot w - w \cdot \sum_{i=1}^{l} \alpha_i y_i x_i - b \cdot \sum_{i=1}^{l} \alpha_i y_i + \sum_{i=1}^{l} \alpha_i \\
&= \frac{1}{2} w \cdot w - w \cdot w + \sum_{i=1}^{l} \alpha_i \\
&= -\frac{1}{2} w \cdot w + \sum_{i=1}^{l} \alpha_i \\
&= -\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j x_i x_j + \sum_{i=1}^{l} \alpha_i
\end{aligned}
$$

SVMs for linearly separable classes

Dual Optimization Problem:

$$\text{maximize: } L'(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j x_i x_j$$

$$\text{to the constraints } \alpha_i \geq 0 \text{ and } \sum_{i=1}^{l} y_i \alpha_i = 0$$

This optimization problem can be solved numerically with the help of standard quadratic programming techniques.

SVMs for linearly separable classes

Solution of the optimization problem:

$$
\begin{aligned}
w^* &= \sum_{i=1}^{l} \alpha_i y_i x_i = \sum_{x_i \in SV} \alpha_i y_i x_i \\
b^* &= -\frac{1}{2} \cdot w^* \cdot (x_p + x_m) \\
&\quad \text{for arbitrary } x_p \in SV,\ y_p = +1,\ \text{und } x_m \in SV,\ y_m = -1
\end{aligned}
$$

where

$$SV = \{x_i \mid \alpha_i > 0,\ i = 1, 2, \ldots, l\}$$

is the set of all support vectors.

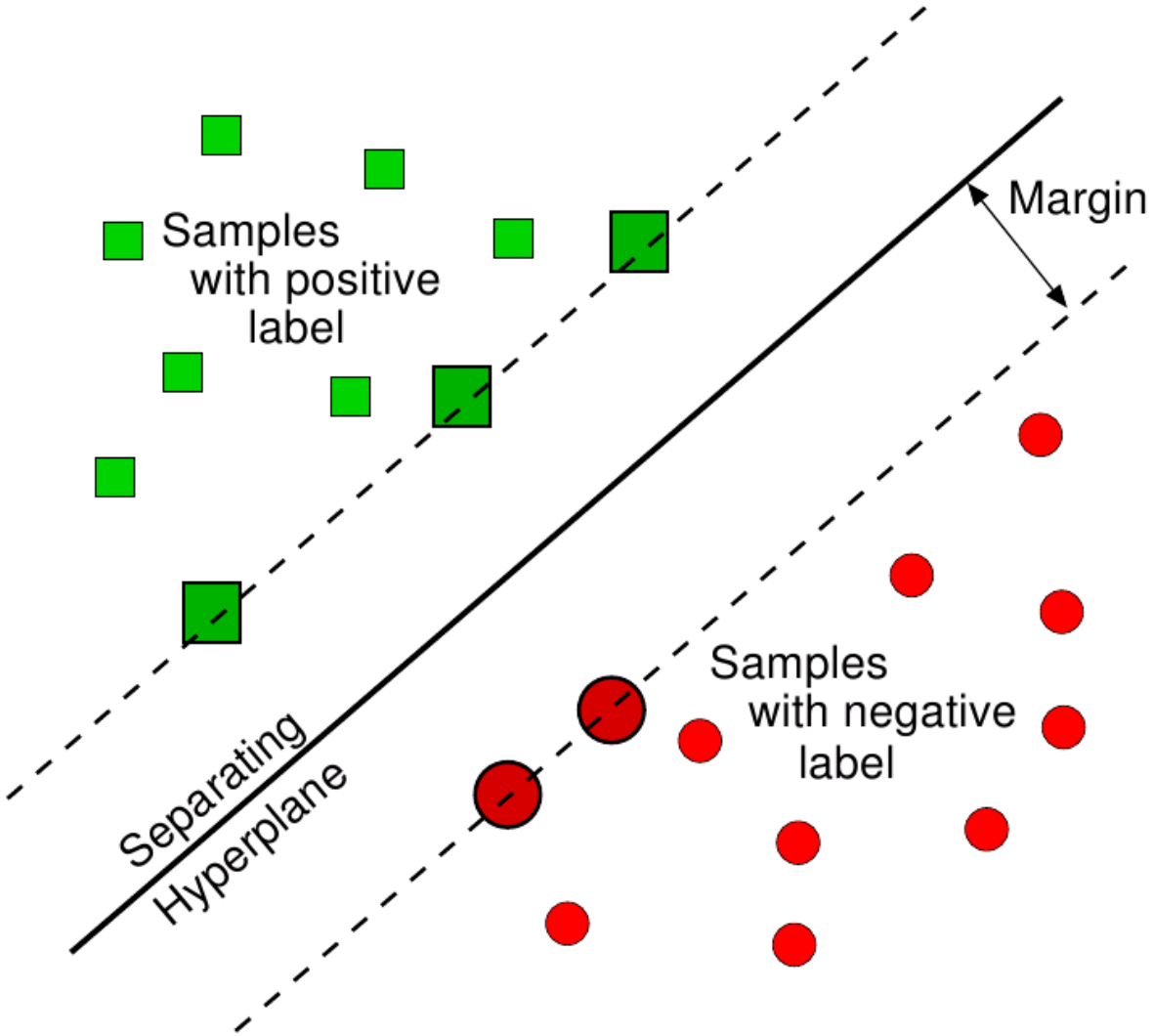
Classification rule:

$$\operatorname{sgn}(w^* x + b^*) = \operatorname{sgn}[(\sum_{x_i \in SV} \alpha_i y_i x_i) x + b^*]$$

The classification only depends on the support vectors!

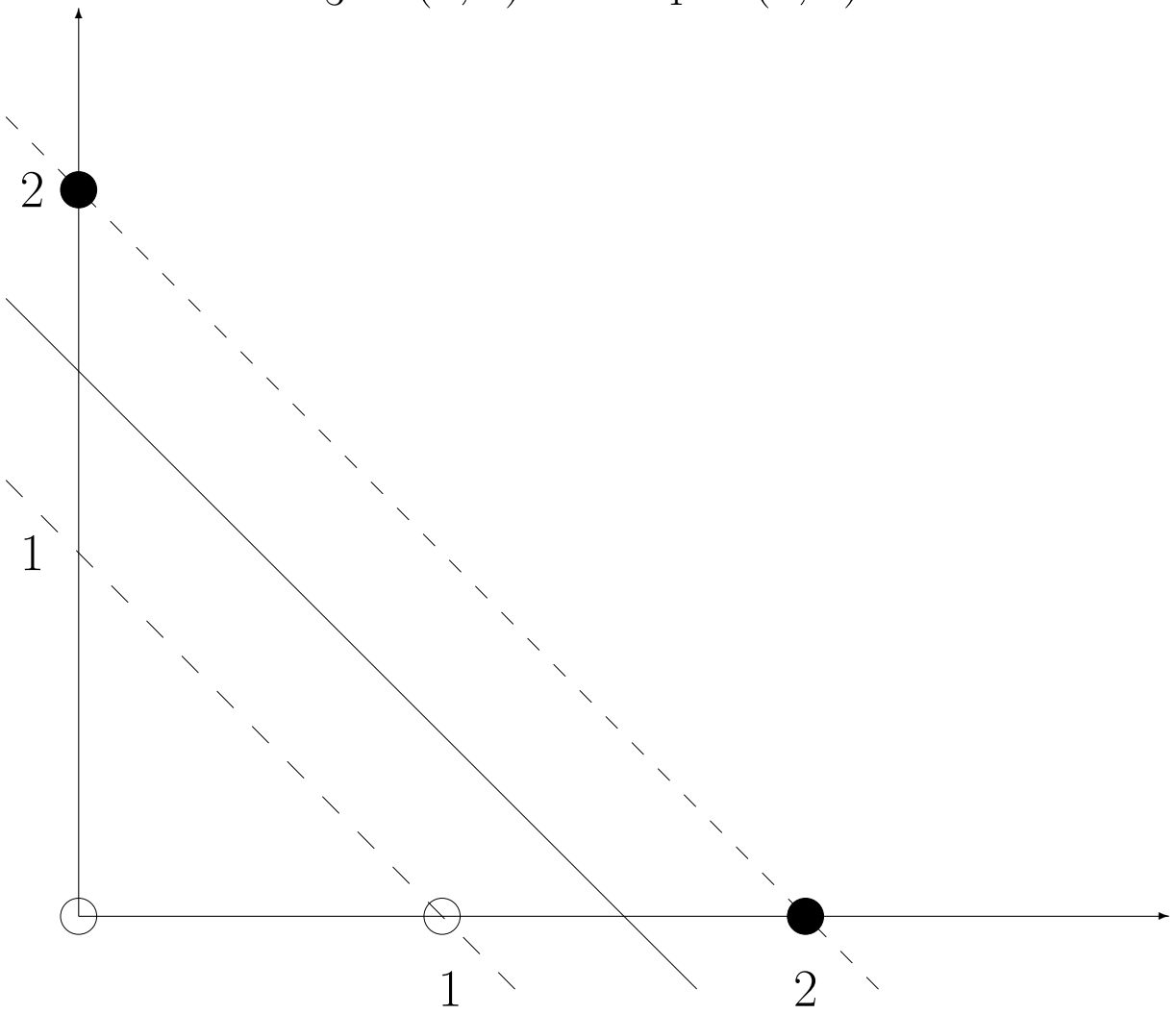
SVMs for linearly separable classes

Example: Support Vectors



SVMs for linearly separable classes

Example: class +1 contains $x_1 = (0, 0)$ and $x_2 = (1, 0)$;
class -1 contains $x_3 = (2, 0)$ and $x_4 = (0, 2)$



SVMs for linearly separable classes

The Dual Optimization Problem is:

maximize: $L'(\alpha) = (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4) - \frac{1}{2}(\alpha_2^2 - 4\alpha_2\alpha_3 + 4\alpha_3^2 + 4\alpha_4^2)$
to the constraints $\alpha_i \geq 0$ and $\alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0$

Solution:

$$\alpha_1 = 0, \quad \alpha_2 = 1, \quad \alpha_3 = \tfrac{3}{4}, \quad \alpha_4 = \tfrac{1}{4}$$
$$SV = \{(1,0),\ (2,0),\ (0,2)\}$$
$$w^* = 1 \cdot (1,0) - \tfrac{3}{4} \cdot (2,0) - \tfrac{1}{4} \cdot (0,2) = (-\tfrac{1}{2}, -\tfrac{1}{2})$$
$$b^* = -\tfrac{1}{2} \cdot (-\tfrac{1}{2}, -\tfrac{1}{2}) \cdot ((1,0) + (2,0)) = \tfrac{3}{4}$$

Optimal separation line: $x + y = \frac{3}{2}$

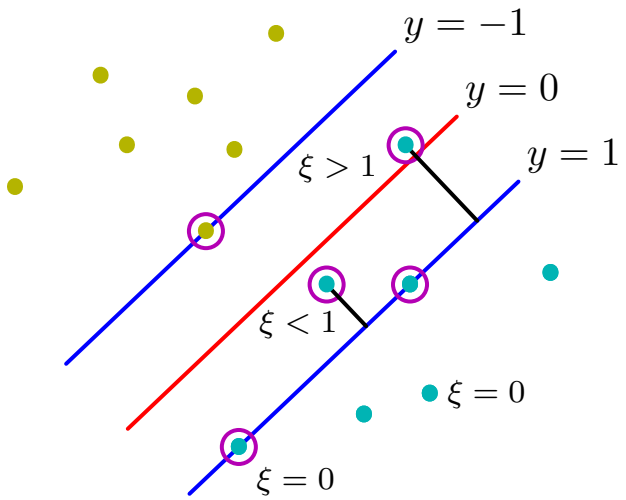
SVMs for linearly separable classes

Observations:

- For the Support Vectors holds: $\alpha_i > 0$
- For all training samples outside the margin holds: $\alpha_i = 0$
- Support Vectors form a $sparse$ representation of the sample; They are sufficient for classification.
- The solution is the global optima and unique
- The optimization procedure only requires scalar products $x_i x_j$

SVMs for non-linearly separable classes

In this example there is no separating line such as $\forall i \; [y_i(wx_i + b) \geq 1]$



All three cases can be interpreted as: $y_i(wx_i + b) \; \geq \; 1 - \xi_i$

A) $\xi_i = 0$

B) $0 < \xi_i \leq 1$

C) $\xi_i > 1$

Three possible cases:

A) Vectors beyond the margin, which are correctly classified, i.e.

$$y_i(wx_i + b) \; \geq \; 1$$

B) Vectors within the margin, which are correctly classified, i.e.

$$0 \; \leq \; y_i(wx_i + b) \; < \; 1$$

C) Vectors that are not correctly classified, i.e.

$$y_i(wx_i + b) \; < \; 0$$

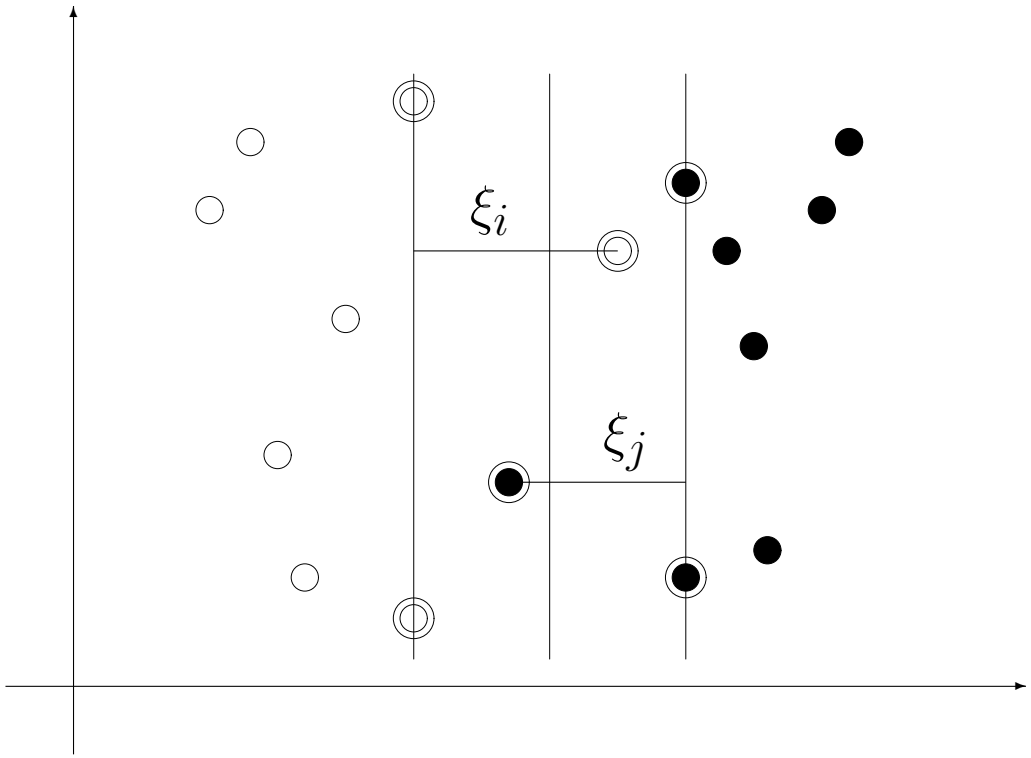
SVMs for non-linearly separable classes

Motivation for generalization:

Previous approach gives no solution for classes that are non-lin. separable.

Improvement of the generalization on outliers within the margin

Soft-Margin SVM: Introduce "slack"-Variables



SVMs for non-linearly separable classes

Penalty for outliers via "slack"-Variables

Primale Optimization Problem:

minimize: $J(w, b, \xi) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{l} \xi_i$

to the constraints $\forall i \ [y_i(wx_i + b) \geq 1 - \xi_i, \ \xi_i \geq 0]$

Dual Optimization Problem:

maximize: $L'(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j x_i x_j$

to the constraints $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{l} y_i \alpha_i = 0$

(Neither slack-Variables nor Lagrange-Multiplier occur in the dual optimization problem.)

The only difference compared to the linear separable case: Constant C in the constraints.

SVMs for non-linearly separable classes

Solution of the optimization problem:

$$\begin{aligned} w^* &= \sum_{i=1}^{l} \alpha_i y_i x_i = \sum_{x_i \in SV} \alpha_i y_i x_i \\ b^* &= y_k(1 - \xi_k) - w^* x_k; \quad k = \arg\max_i \alpha_i \end{aligned}$$

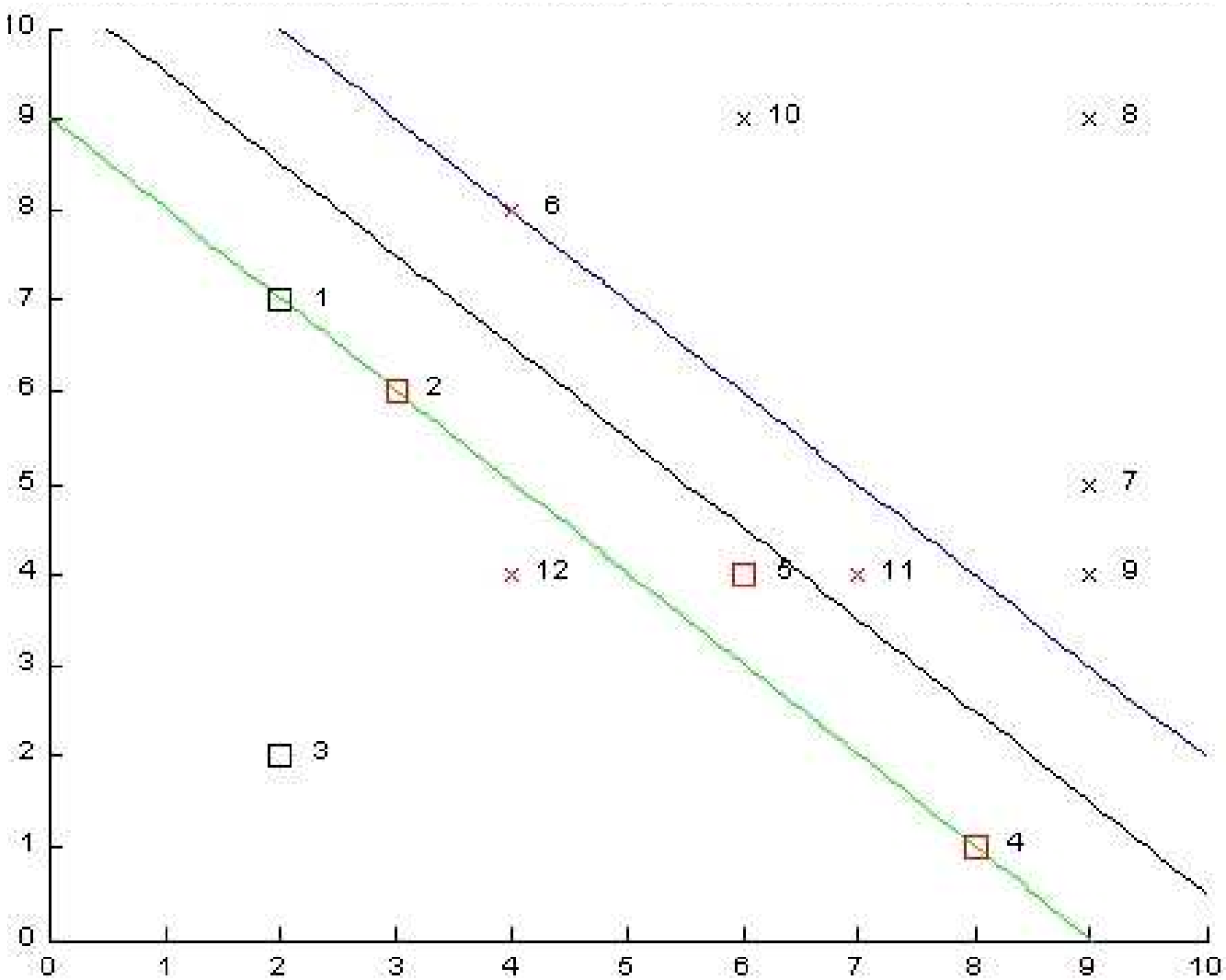
where

$$SV = \{x_i \mid \alpha_i > 0,\ i = 1, 2, \ldots, l\}$$

describes the set of all Support Vectors.

SVMs for non-linearly separable classes

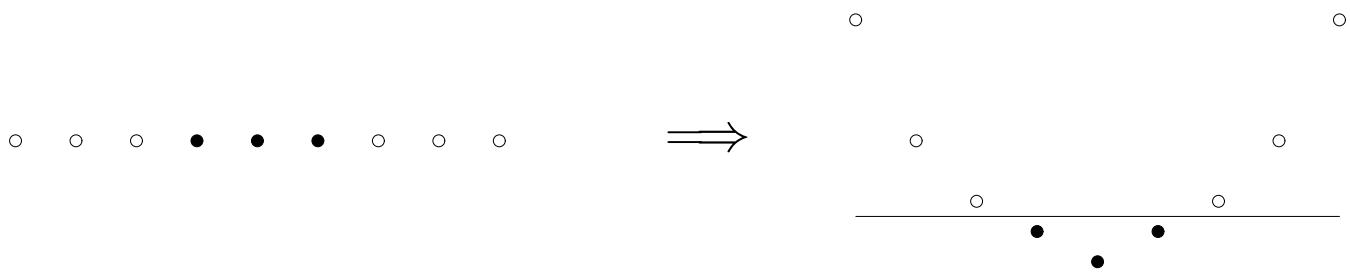
Example: non-linearly separable classes



Non-linear SVMs

Non-linear class boundaries $\rightarrow$ low precision

Example: Transformation $\Psi(x) = (x, x^2) \rightarrow C_1$ and C_2 linearly separable



Idea:

Transformation of attributes $x \in \Re^n$ in a higher dimensional space $\Re^m$, $m > n$ by

$$\Psi : \quad \Re^n \longrightarrow \Re^m$$

and search for an optimal linear separating hyperplane in this space.

Transformation Ψ increases linear separability.

Separating hyperplane in $\Re^m \equiv$ non-linear separating plane in $\Re^n$

Non-linear SVMs

Problem: High dimensionality of the attribute space $\Re^m$
E.g. Polynomes of p-th degree over $\Re^n \rightarrow \Re^m$, $m = O(n^p)$

Trick with kernel function:

Originally in $\Re^n$: only scalar products $x_i x_j$ required
new in $\Re^m$: only scalar products $\Psi(x_i)\Psi(x_j)$ required

Solution
No need to compute $\Psi(x_i)\Psi(x_j)$, but express them at reduced complexity with the kernel function

$$K(x_i, x_j) \;=\; \Psi(x_i)\Psi(x_j)$$

Example: For the transformation $\Psi : \Re^2 \longrightarrow \Re^6$

$$\Psi((y_1, y_2)) = (y_1^2,\ y_2^2,\ \sqrt{2}y_1,\ \sqrt{2}y_2,\ \sqrt{2}y_1y_2,\ 1)$$

the kernel function computes

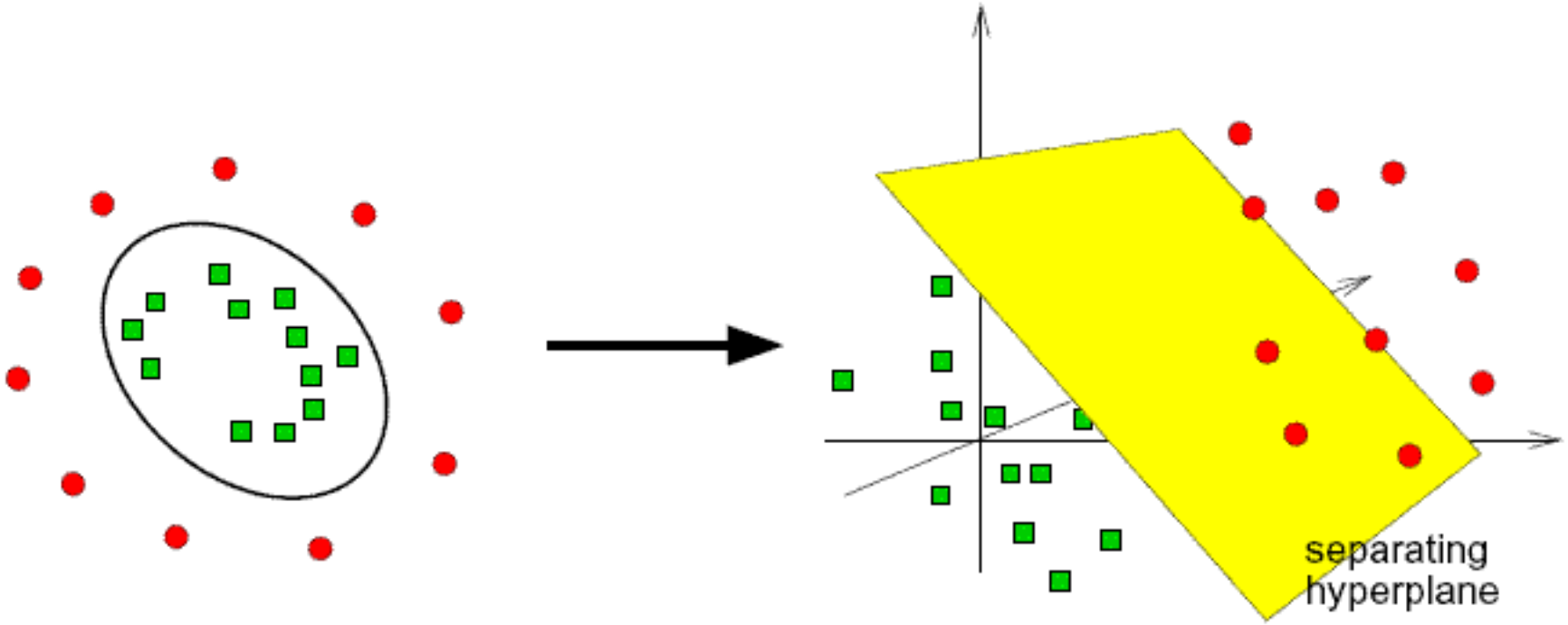
$$\begin{aligned} K(x_i, x_j) &= (x_i x_j + 1)^2 \\ &= ((y_{i1}, y_{i2}) \cdot (y_{j1}, y_{j2}) + 1)^2 \\ &= (y_{i1}y_{j1} + y_{i2}y_{j2} + 1)^2 \\ &= (y_{i1}^2, y_{i2}^2, \sqrt{2}y_{i1}, \sqrt{2}y_{i2}, \sqrt{2}y_{i1}y_{i2}, 1) \\ &\quad \cdot (y_{j1}^2, y_{j2}^2, \sqrt{2}y_{j1}, \sqrt{2}y_{j2}, \sqrt{2}y_{j1}y_{j2}, 1) \\ &= \Psi(x_i)\Psi(x_j) \end{aligned}$$

the scalar product in the new attribute space $\Re^6$

Non-linear SVMs

Example: $\Psi : \Re^2 \longrightarrow \Re^3$

$$\Psi((y_1, y_2)) = (y_1^2, \sqrt{2}y_1y_2, y_2^2)$$



The kernel function

$$K(x_i, x_j) = (x_i x_j)^2 = \Psi(x_i)\Psi(x_j)$$

computes the scalar product in the new attribute space $\Re^3$. It is possible to compute the scalar product of $\Psi(x_i)$ and $\Psi(x_j)$ without applying the function Ψ.

Nonlinear SVMs

Commonly used kernel functions:

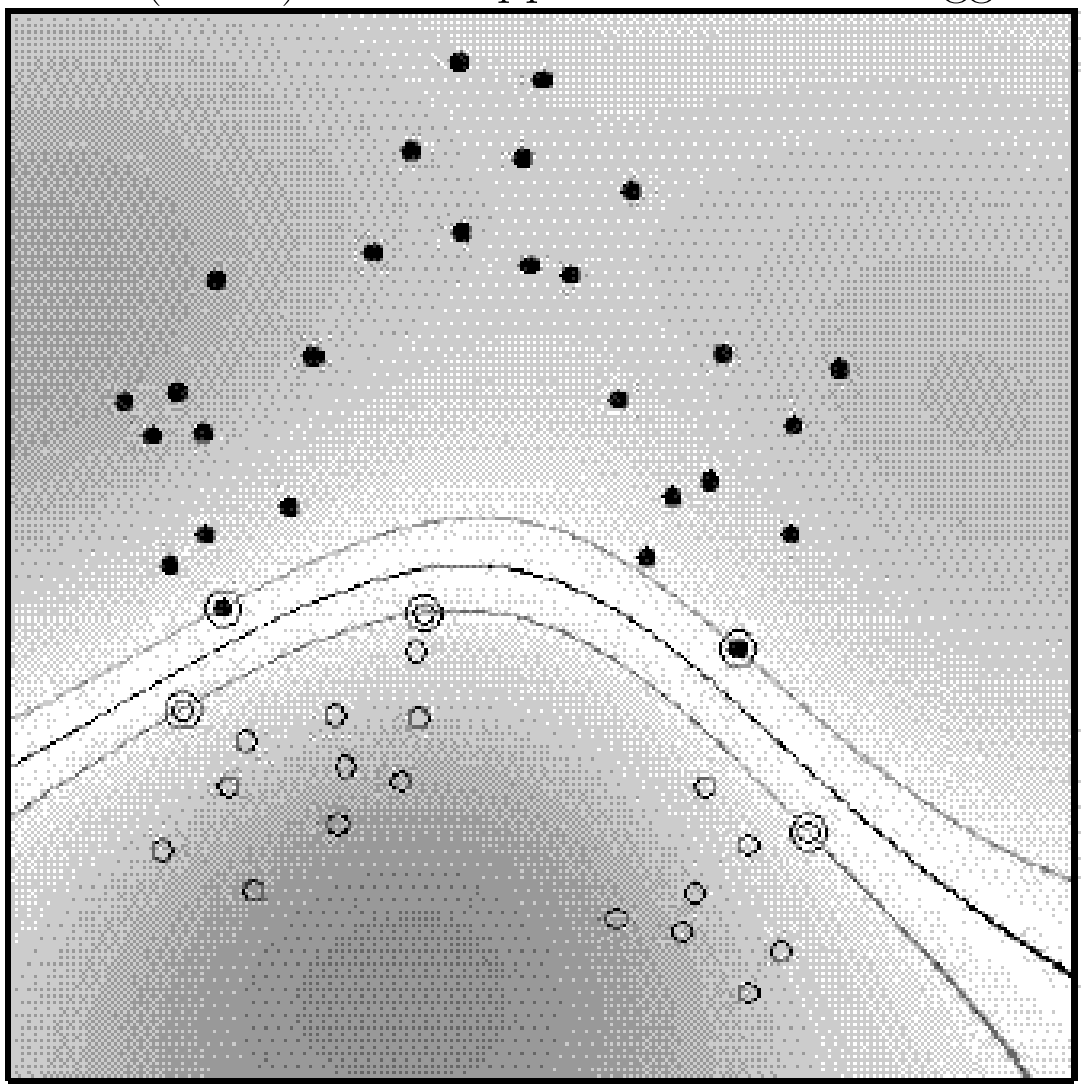
$$
\begin{array}{lrcl}
\text{Polynomial-Kernel:} & K(x_i, x_j) & = & (x_i x_j)^d \\
\text{Gauss-Kernel:} & K(x_i, x_j) & = & e^{-\frac{||x_i - x_j||^2}{c}} \\
\text{Sigmoid-Kernel:} & K(x_i, x_j) & = & \tanh(\beta_1 x_i x_j + \beta_2)
\end{array}
$$

Linear combination of valid kernels $\rightarrow$ new kernel functions

We do not need to know what the new attribute space $\Re^m$ looks like. The only thing we need is the kernel function as a measure for similarity.

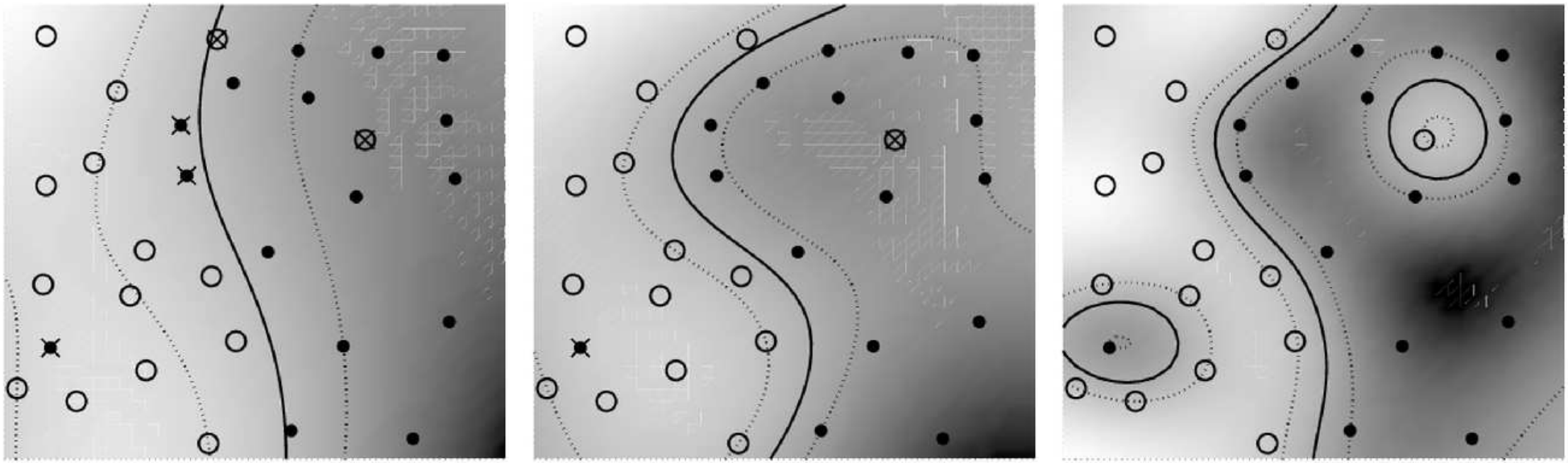
Non-linear SVMs

Example: Gauss-Kernel ($c = 1$). The Support Vectors are tagged by an extra circle.



Non-linear SVMs

Example: Gauss-Kernel ($c = 1$) for Soft-Margin SVM.



Final Remarks

Advantages of SVMs:

- According to current knowledge SVMs yield very good classification results; in some tasks they are considered to be the top-performer.
- Sparse representation of the solution by Support Vectors
- Easily practicable: few parameters, no need for a-priori-knowledge
- Geometrically intuitive operation
- Theoretical statements about results: global optima, ability for generalization

Disadvantages of SVMs

- Learning process is slow and in need of intense memory
- "Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner"

Final Remarks

List of SVM-implementations at
`http://www.kernel-machines.org/software`

The most common one is LIBSVM:
`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`