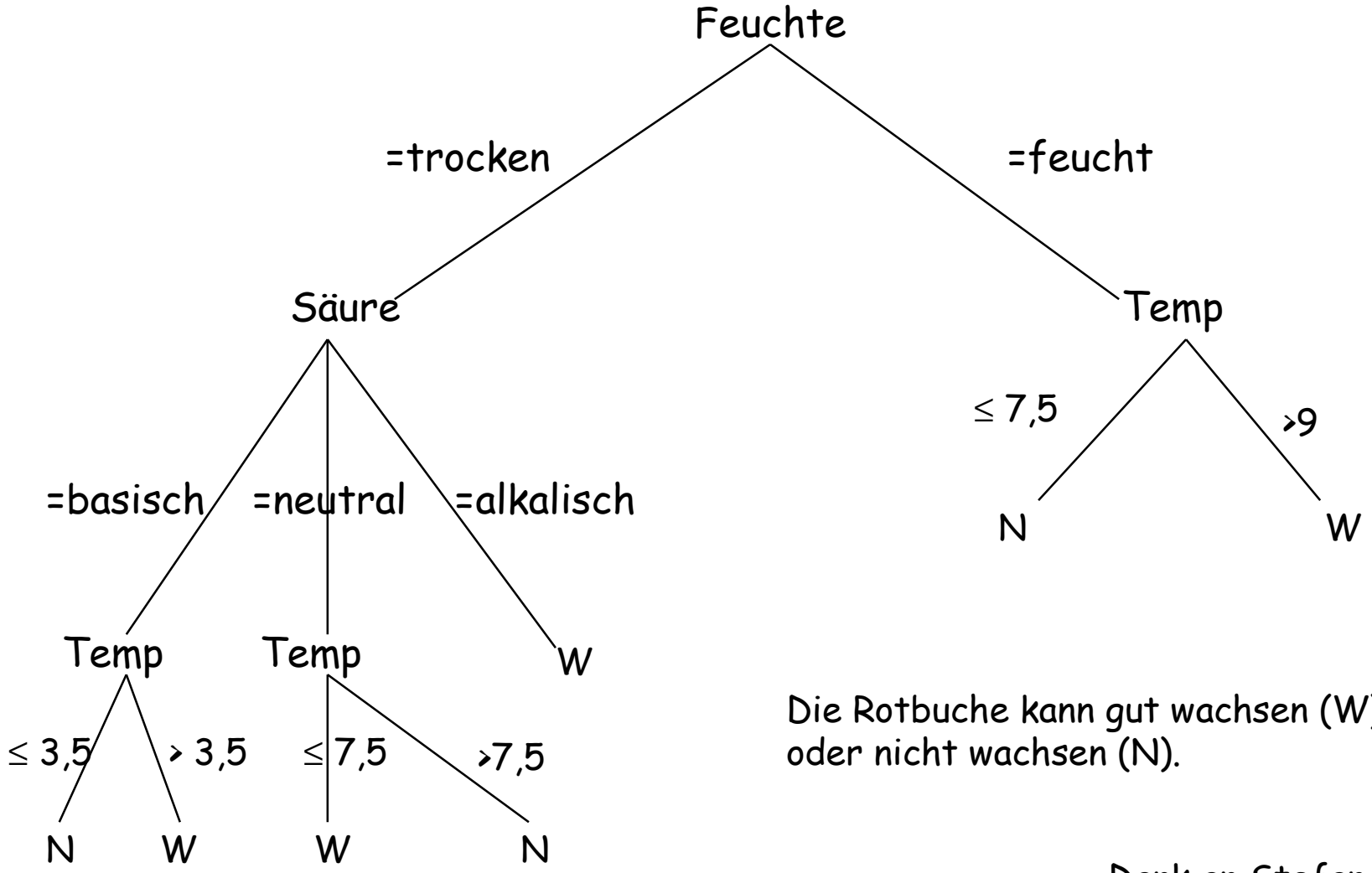
Was kommt jetzt?

Lernen als probabilistische Inferenz – in anderen Worten Statistisches Lernen.

- Entscheidungsbäume Lernaufgabe Begriffslernen
- Stützvektormethode Lernaufgabe Funktionsapproximation
  - Klassifikation (binäre Funktion)
  - Regression (reellwertige Funktion)

Entscheidungsbaum



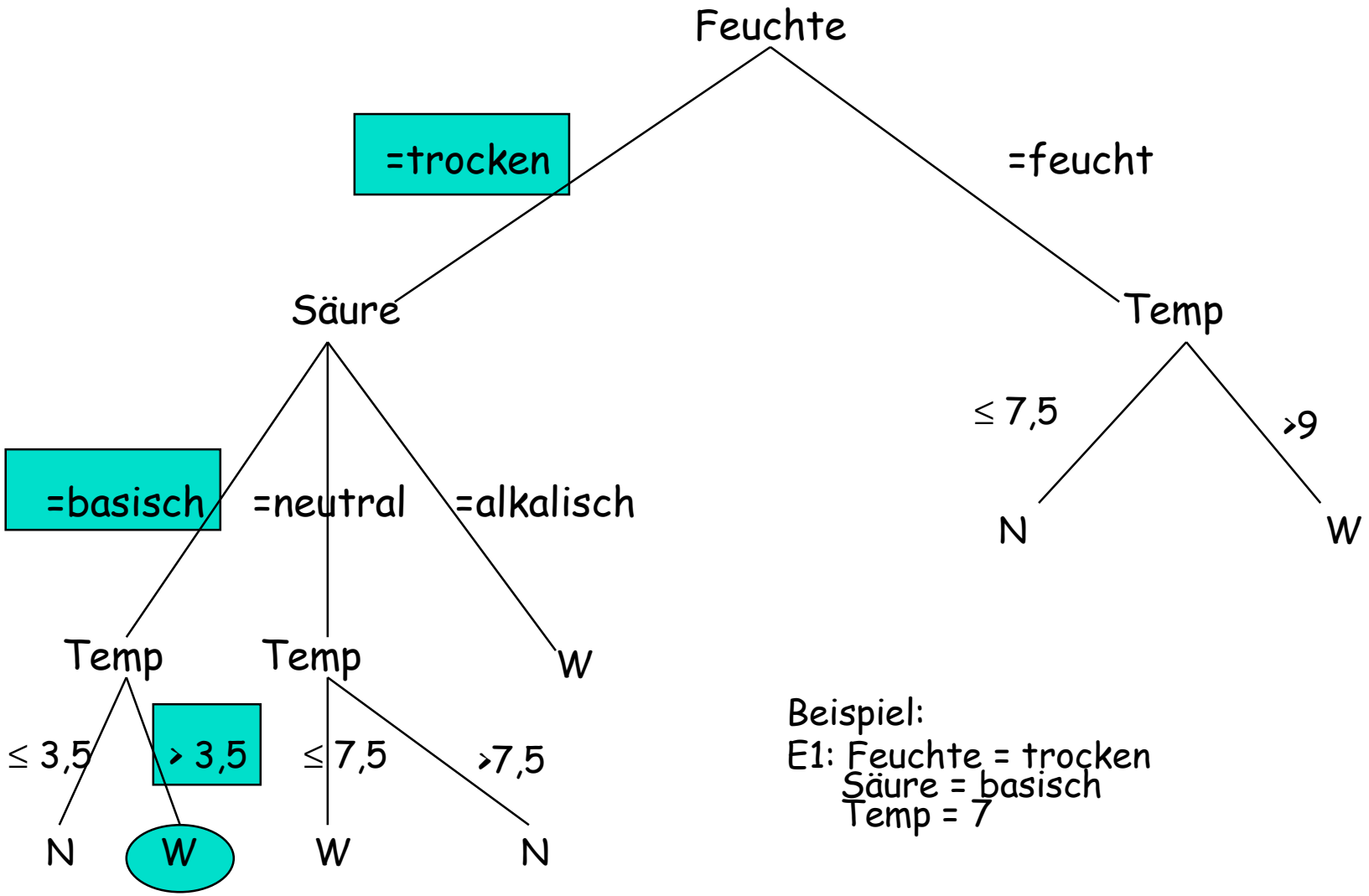
Die Rotbuche kann gut wachsen (W) oder nicht wachsen (N).

Dank an Stefan Wrobel!

Entscheidungsbäume

- Ein Entscheidungsbaum dient der Klassifikation von Beispielen.
- Ausgehend von der Wurzel des Baums werden die Attribute, deren Werte von einem Knoten zum Nachfolgeknoten führen, mit den Attributwerten des Beispiels verglichen und der entsprechende Pfad verfolgt.
- Wenn ein Blatt des Baums erreicht wird, gibt es die Klassifikation des Beispiels an.

Beispiel



Beispiel:
E1: Feuchte = trocken
Säure = basisch
Temp = 7

Lernen von Entscheidungsbäumen

Gegeben:

LE: Beispiele in Form von Attributen und Werten,

wobei ein binäres Attribut die Klasse des Beispiels angibt.

LH: Alle aus den gegebenen Attributen mit ihren Werten konstruierbare Entscheidungsbäume.

Ziel:

Ein Entscheidungsbaum, der Beispiele mit minimalem Fehler klassifiziert.

Beispiele

ID	Feuchte	Säure	Temp	Klasse
1	trocken	basisch	7	W
2	feucht	neutral	8	N
3	trocken	neutral	7	W
4	feucht	alkalisch	5	N
5	trocken	neutral	8	N
6	trocken	neutral	6	W
7	trocken	neutral	11	N
8	trocken	neutral	9	N
9	trocken	alkalisch	9	W

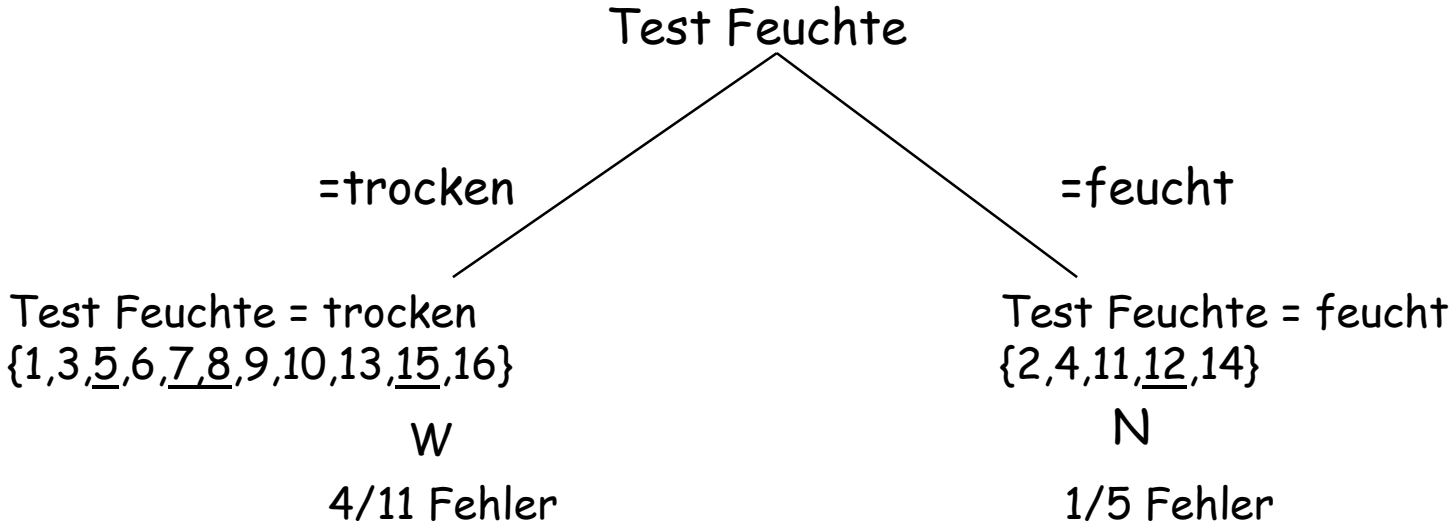
ID	Feuchte	Säure	Temp	Klasse
10	trocken	alkalisch	8	W
11	feucht	basisch	7	N
12	feucht	neutral	10	W
13	trocken	basisch	6	W
14	feucht	alkalisch	7	N
15	trocken	basisch	3	N
16	trocken	basisch	4	W

Beispiele nach Attributen sortiert

Feuchte: trocken	Feuchte: feucht	Säure: basisch	Säure: neutral	Temp: >6,5	Temp: ≤6,5
1 W	2 N	1 W	2 N	1 W	4 N
3 W	4 N	11 N	3 W	2 N	6 W
5 N	11 N	13 W	5 N	3 W	13 W
6 W	12 W	15 N	6 W	5 N	15 N
7 N	14 N	16 W	7 N	10 W	16 W
8 N		W:3,N:2	8 N	11 N	
9 W		Total: 5	12 W	12 W	
10 W		Säure: alkalisch		14 N	
13 W		4 N			
15 N		9 W			
16 W		10 W			
		14 N			
W:7, N: 4	W: 1, N: 4	W:2, N:2	W:3,N:4	W:4, N:4	W: 3, N:2
Total: 11	Total: 5	Total: 4	Total: 7	Total: 8	Total: 5

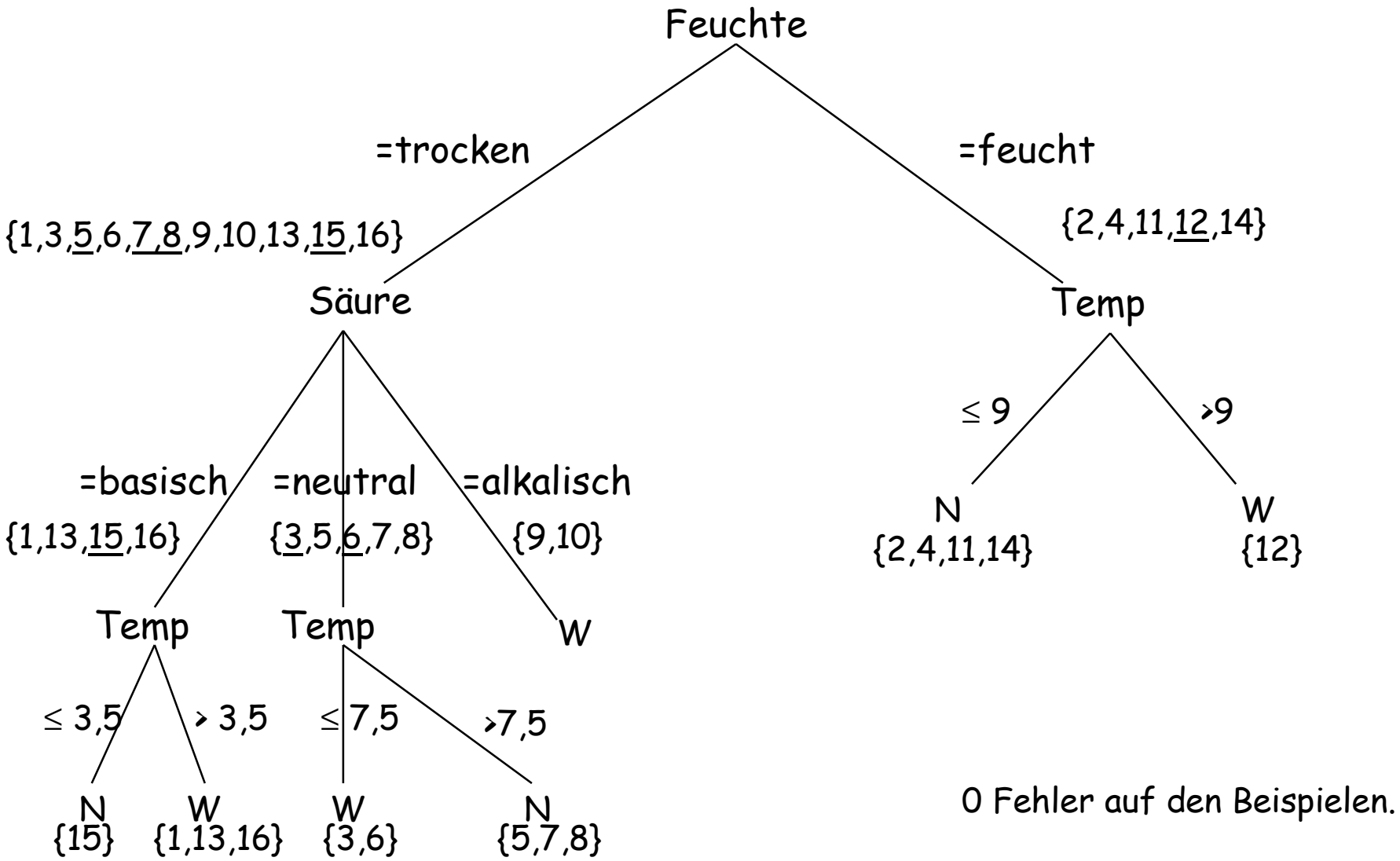
Rekursiver Ansatz

- Die Klasse, die am häufigsten bei einem Attributwert vorkommt, wird vorausgesagt, wenn der Attributwert im Beispiel vorkommt.
- Es wird das Attribut mit dem kleinsten Fehler auf den Beispielen gewählt.



- Die Knoten werden wieder nach Attributen aufgeteilt, wenn es noch einen Fehler auf den Beispielen gibt.

Beispiel



0 Fehler auf den Beispielen.

TDIDT - Algorithmus

TDIDT (E, Tests)

Falls E nur Beispiele einer Klase enthält, liefere einen Blattknoten mit dieser Klasse zurück. Sonst

Für jeden Test in Tests, berechne Qualität (Test, E).

Wähle den Test mit der höchsten Qualität für den aktuellen Knoten aus.

Teile E anhand des Testes gemäß der Attributwerte in Mengen $E_1, ..., E_k$ auf.

Für i = 1, ..., k rufe rekursiv auf: TDIDT(E_i, Tests\{Test}).

Liefere den aktuellen Knoten mit den darunter liegenden Teilbäumen zurück.

Qualitätsmaß

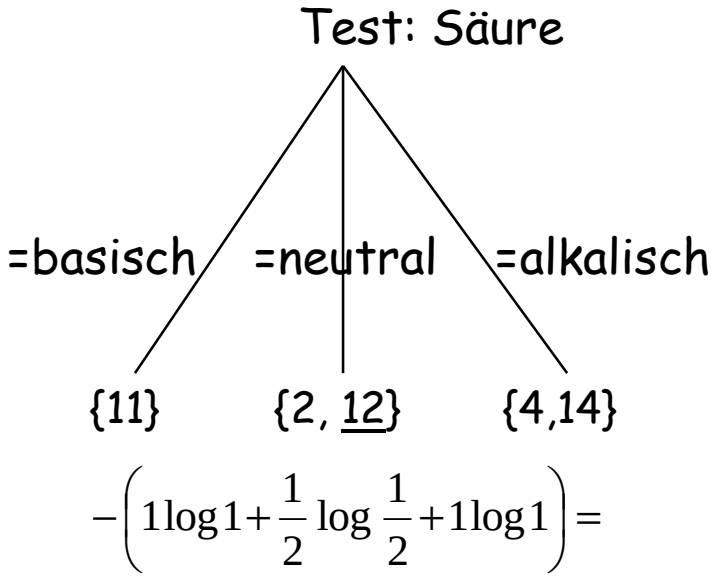
- Information: wie viele ja-/nein-Fragen braucht man, um ein Element einer Menge zu identifizieren? → Bits
- Je nach Wahrscheinlichkeit p_i des Elements sind es weniger oder mehr Fragen.
- Informationsgehalt einer Menge mit m Elementen(Entropie): wie viele Fragen braucht man durchschnittlich, um jedes Element zu identifizieren.

$$-\sum_{i=1}^{m} p_i \log p_i$$

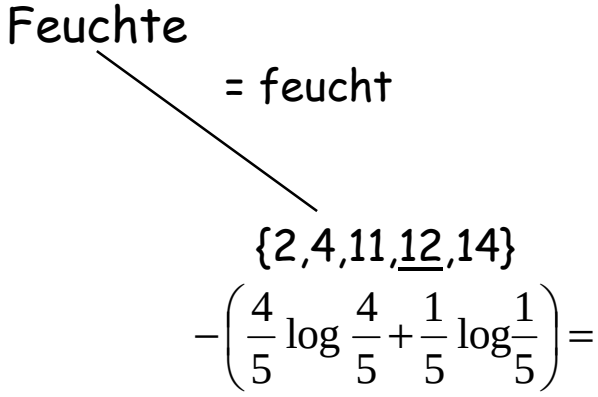
Informationsgewinn

- Hier sind die zu identifizierenden Elemente die Klassen.
- Als Annäherung an die Wahrscheinlichkeit nehmen wir die Häufigkeiten in den Beispielen.
- Wir wollen den Test auswählen, der den Informationsgehalt der durch ihn entstehenden Beispielmengen am meisten reduziert.

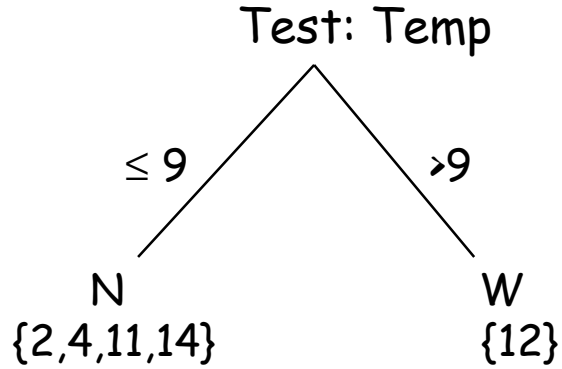
Beispiel



$$-\left(1\log 1+\frac{1}{2}\log\frac{1}{2}+1\log 1\right)=$$



$$-\left(\frac{4}{5}\log\frac{4}{5}+\frac{1}{5}\log\frac{1}{5}\right)=$$



$$1\log 1+1\log 1=0$$

Der Gewinn ist bei Temp größer
als bei Säure.
Es wird der Test nach dem Attribut
Temp gewählt.

Probleme

- Überanpassung (overfitting): Da der Trainingsfehler minimiert wird, kann sich der Entscheidungsbaum zu stark an die Beispiele anpassen, so dass er auf den Testdaten falsch entscheidet.
  Ausweg: einen Teil der Traininsdaten zum Test des vorläufigen Ergebnisses beiseite legen (Kreuvalidierung). Den Baum zur Optimierung auf den Validierungsdaten stutzen.
- Identische Testfolgen können an verschiedenen Stellen des Baumes auftreten.

Was wissen Sie jetzt?

- TDIDT ist ein effektiver und effizienter Algorithmus. Sie kennen seine Arbeitsweise, wissen,
  - dass Tests an den Knoten Beispielmengen zurückliefern
  - dass die Tests nach einem Qualitätskriterium ausgewählt werden (automatische Merkmalsselektion) und
  - dass meist der Informationsgewinn als Kriterium gewählt wird.
- C4.5 (bei weka J48) ist das am häufigsten verwendete Lernverfahren.

Erinnerung: Funktionslernen

Gegeben:

Beispiele X in LE

- die anhand einer Wahrscheinlichkeitsverteilung P auf X erzeugt wurden und
- mit einem Funktionswert Y = t(X) versehen sind (alternativ: Eine Wahrscheinlichkeitsverteilung P(Y|X) der möglichen Funktionswerte - verrauschte Daten).

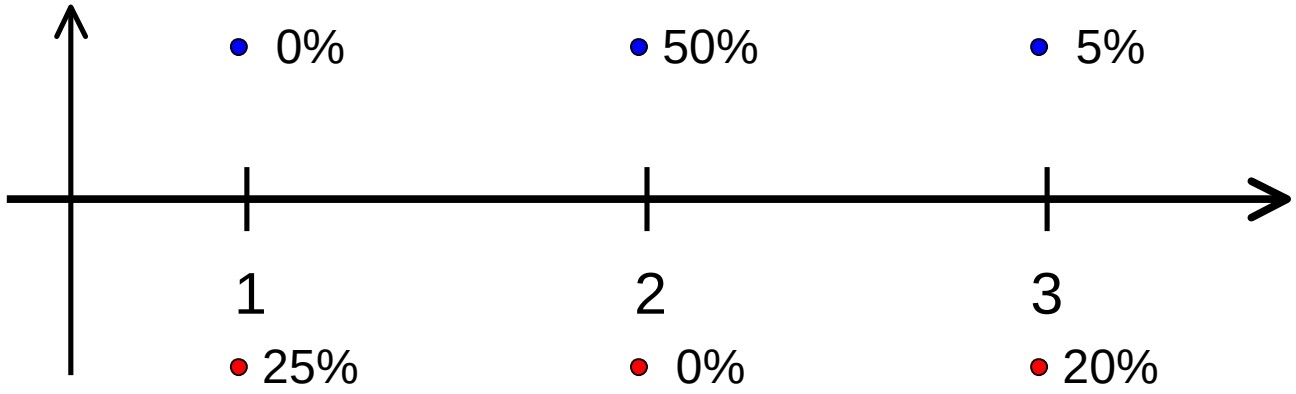
H die Menge von Funktionen in LH.

Ziel: Eine Hypothese h(X) $\in$ H, die das erwartete Fehlerrisiko R(h) minimiert.

Risiko:

$$R(h) = \sum_{x} Q(x,h)P(x)$$

Beispiel: Funktionenlernen



- $H = \{ f_a \mid f_a(x) = 1, \text{für } x \geq a, f_a(x) = -1 \text{ sonst}, a \in \Re \}$
- $R(f_0) = 0{,}25 + 0 + 0{,}20 = 0{,}45$
- $R(f_{1,5}) = 0 + 0 + 0{,}20 = 0{,}20$
- $R(f_{3,5}) = 0 + 0{,}5 + 0{,}05 = 0{,}55$

Dank an Stefan Rüping!

Reale Beispiele

- Klassifikation: $Q(x,h) = 0$, falls $t(x) = h(x)$, 1 sonst
  - Textklassifikation (x = Worthäufigkeiten)
  - Handschriftenerkennung (x = Pixel in Bild)
  - Vibrationsanalyse in Triebwerken (x = Frequenzen)
  - Intensivmedizinische Alarmfunktion (x = Vitalzeichen)
- Regression: $Q(x,h) = (t(x)-h(x)))^2$
  - Zeitreihenprognose (x = Zeitreihe, $t(x)$ = nächster Wert)

Erinnerung: Minimierung des beobachteten Fehlers

Funktionslernaufgabe nicht direkt lösbar. Problem:

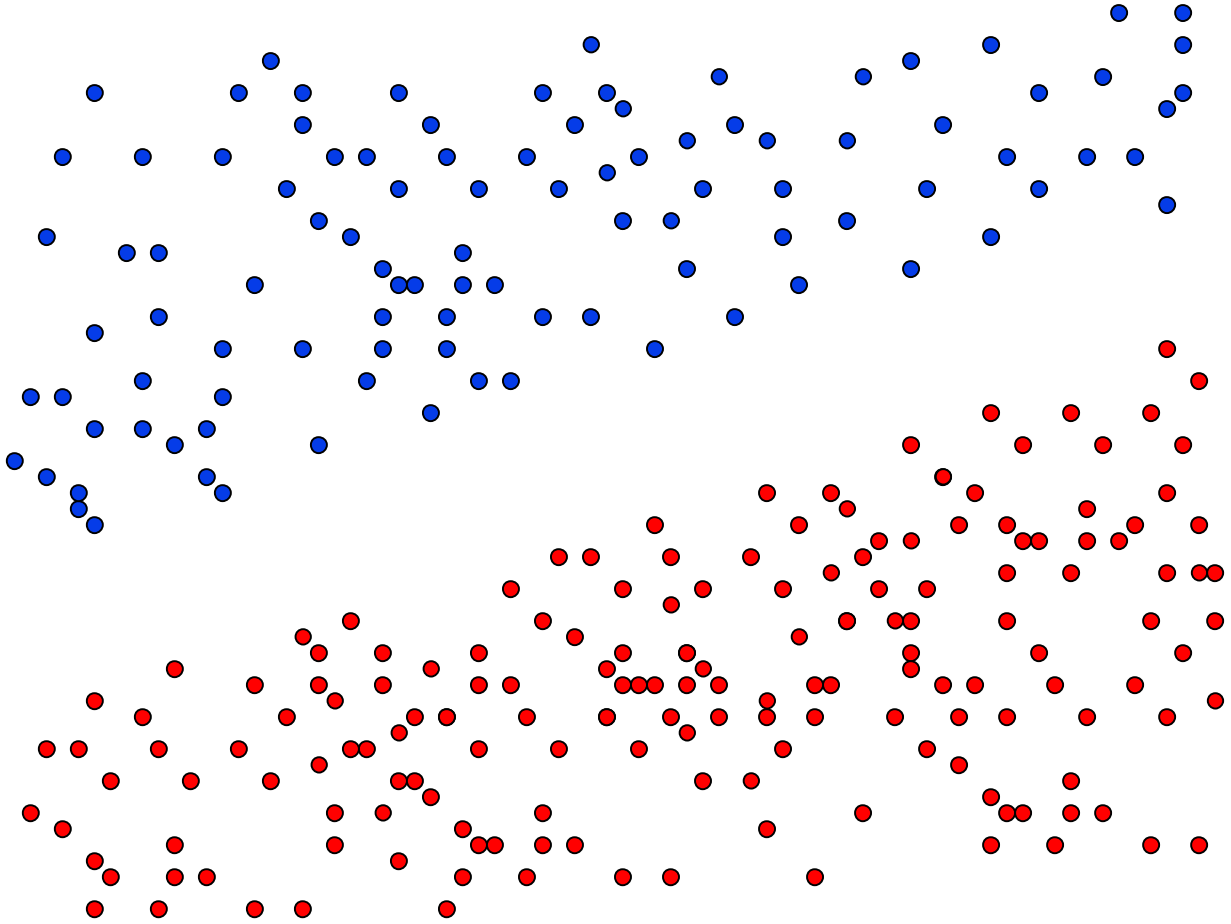
- Die tatsächliche Funktion t(X) ist unbekannt.
- Die zugrunde liegende Wahrscheinlichkeit ist unbekannt.

Ansatz:

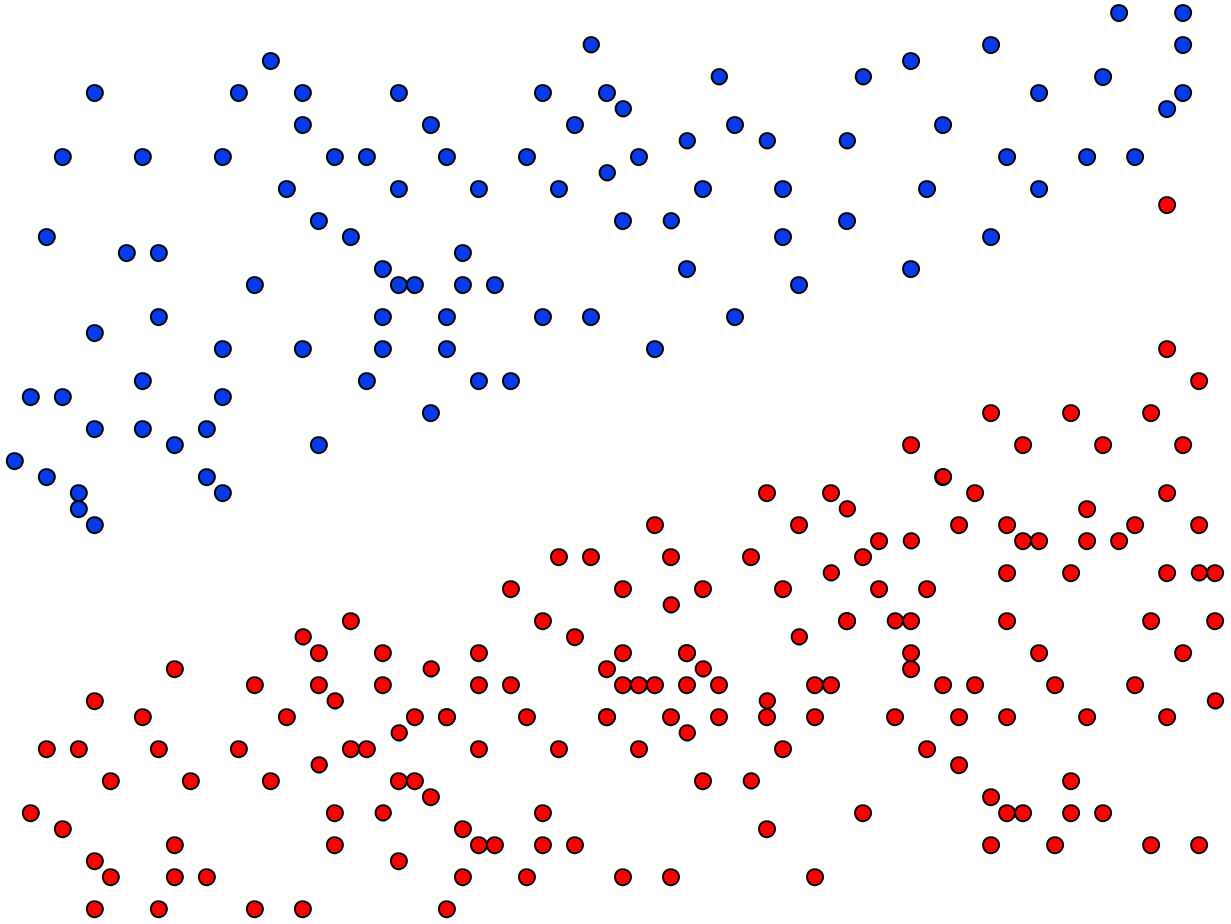
- eine hinreichend große Lernmenge nehmen und für diese den Fehler minimieren.

$\Rightarrow$ Empirical Risk Minimization

Beispiel



Beispiel II

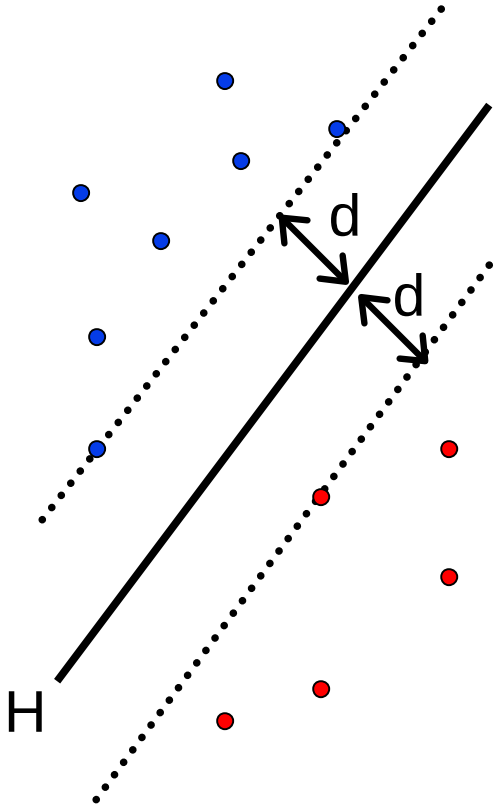


Probleme der ERM

- Aufgabe ist nicht eindeutig beschrieben: Mehrere Funktionen mit minimalem Fehler existieren. Welche wählen?
- Overfitting: Verrauschte Daten und zu wenig Beispiele führen zu falschen Ergebnissen.

Die optimale Hyperebene

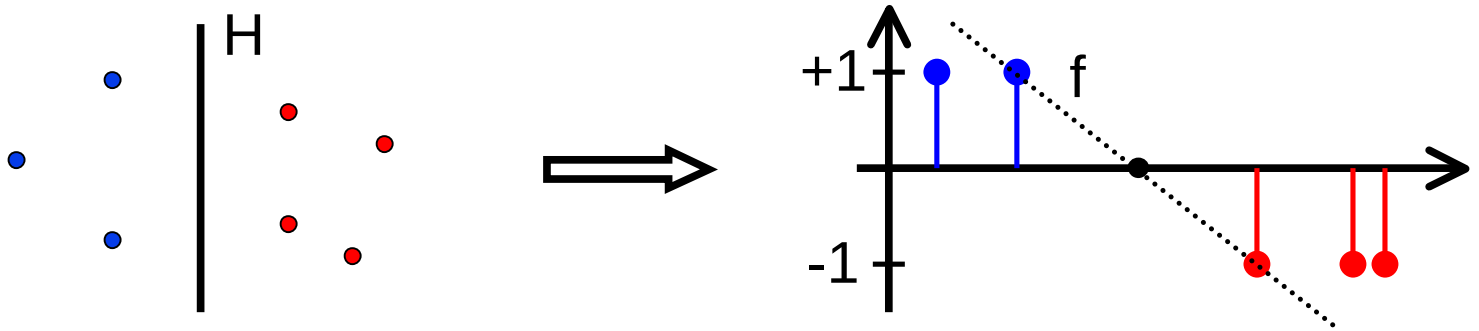
- Beispiele heißen linear trennbar, wenn es eine Hyperebene H gibt, die die positiven und negativen Beispiele voneinander trennt.
- H heißt optimale Hyperebene, wenn ihr Abstand d zum nächsten positiven und zum nächsten negativen Beispiel maximal ist.
- Satz: Es existiert eine eindeutig bestimmte optimale Hyperebene.



Berechnung der opt. Hyperebene

- Hyperebene
  $H = \{x \mid w^*x+b = 0\}$
- H trennt (x_i,y_i), $y_i \in \{\pm 1\}$
- H ist optimale Hyperebene

- Entscheidungsfunktion $f(x) = w^*x+b$
- $f(x_i) > 0 \Leftrightarrow y_i > 0$
- $||w||$ minimal und
  $f(x_i) \geq 1$, wenn $y_i = 1$
  $f(x_i) \leq -1$, wenn $y_i = -1$

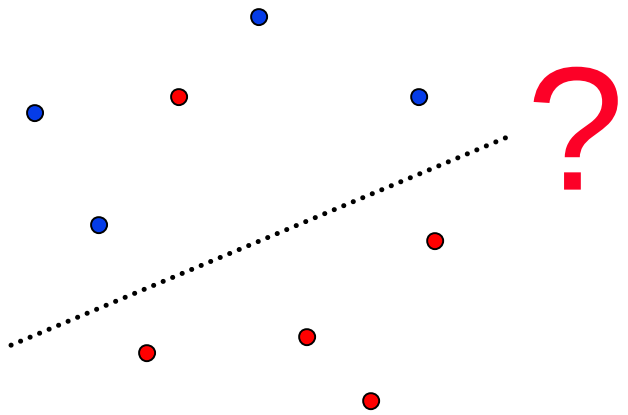


Optimierungsaufgabe der SVM

- Minimiere $||w||^2$
- so dass für alle i gilt:
  $f(x_i) = w*x_i+b \geq 1$ für $y_i = 1$ und
  $f(x_i) = w*x_i+b \leq -1$ für $y_i = -1$
- Äquivalente Nebenbedingung: $y_i*f(x_i) \geq 1$
- Konvexes, quadratisches Optimierungs-problem $\Rightarrow$ eindeutig in $O(n^3)$ lösbar.
- Satz: $||w|| = 1/d$, d = Abstand der optimalen Hyperebene zu den Beispielen.

Nicht linear trennbare Daten

- In der Praxis sind linear trennbare Daten selten.
- 1. Ansatz: Entferne eine minimale Menge von Datenpunkten, so dass die Daten linear trennbar werden (minimale Fehlklassifikation).
- Problem: Algorithmus wird exponentiell.

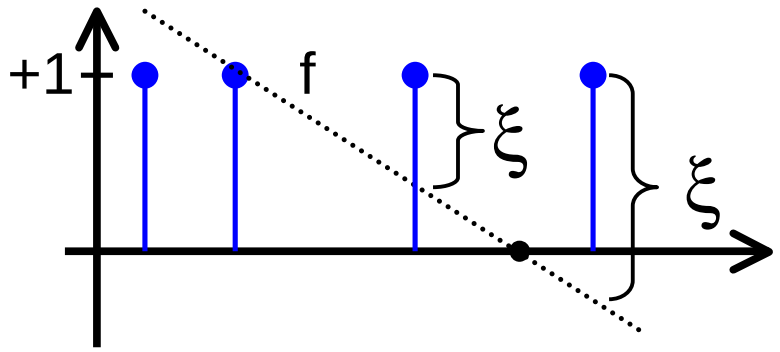


Weich trennende Hyperebene

- Wähle $C \in \Re_{>0}$ und minimiere $\|w\|^2 + C\sum_{i=1}^{n} \xi_i$
- so dass für alle i gilt:

  $f(x_i) = w*x_i+b \geq 1-\xi_i$ für $y_i = 1$ und

  $f(x_i) = w*x_i+b \leq -1+\xi_i$ für $y_i = -1$
- Äquivalent: $y_i*f(x_i) \geq 1- \xi_i$



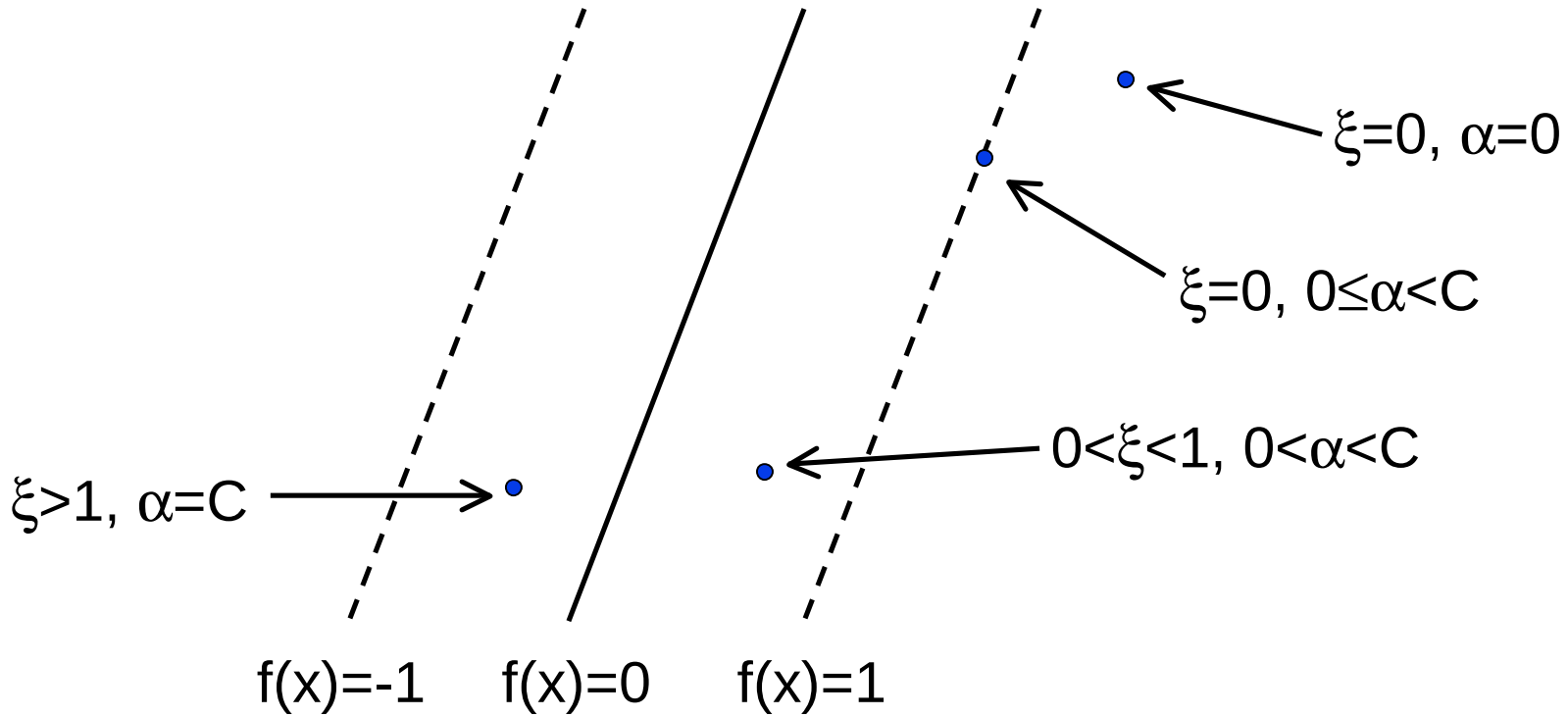
Duales Optimierungsproblem

- Umformung mit Lagrange-Multiplikatoren liefert einfacheres Optimierungsproblem:
- Maximiere

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \left(x_i * x_j \right)$$

- unter $0 \leq \alpha_i \leq C$ für alle i und $\sum \alpha_i y_i = 0$
- Es gilt $w = \sum \alpha_i y_i x_i$, also $f(x) = \sum \alpha_i y_i (x_i * x) + b$

Bedeutung von ξ und α



Beispiele x_i mit $\alpha_i>0$ heißen Stützvektoren $\Rightarrow$ SVM

Optimierungsalgorithmus

s = Gradient von W(α) // $s_i = \sum \alpha_j (x_j * x_i)$

while(nicht konvergiert(s)) // auf ε genau

    WS = working_set(s) // suche k „gute“ Variablen

    α‘ = optimiere(WS) // k neue α-Werte

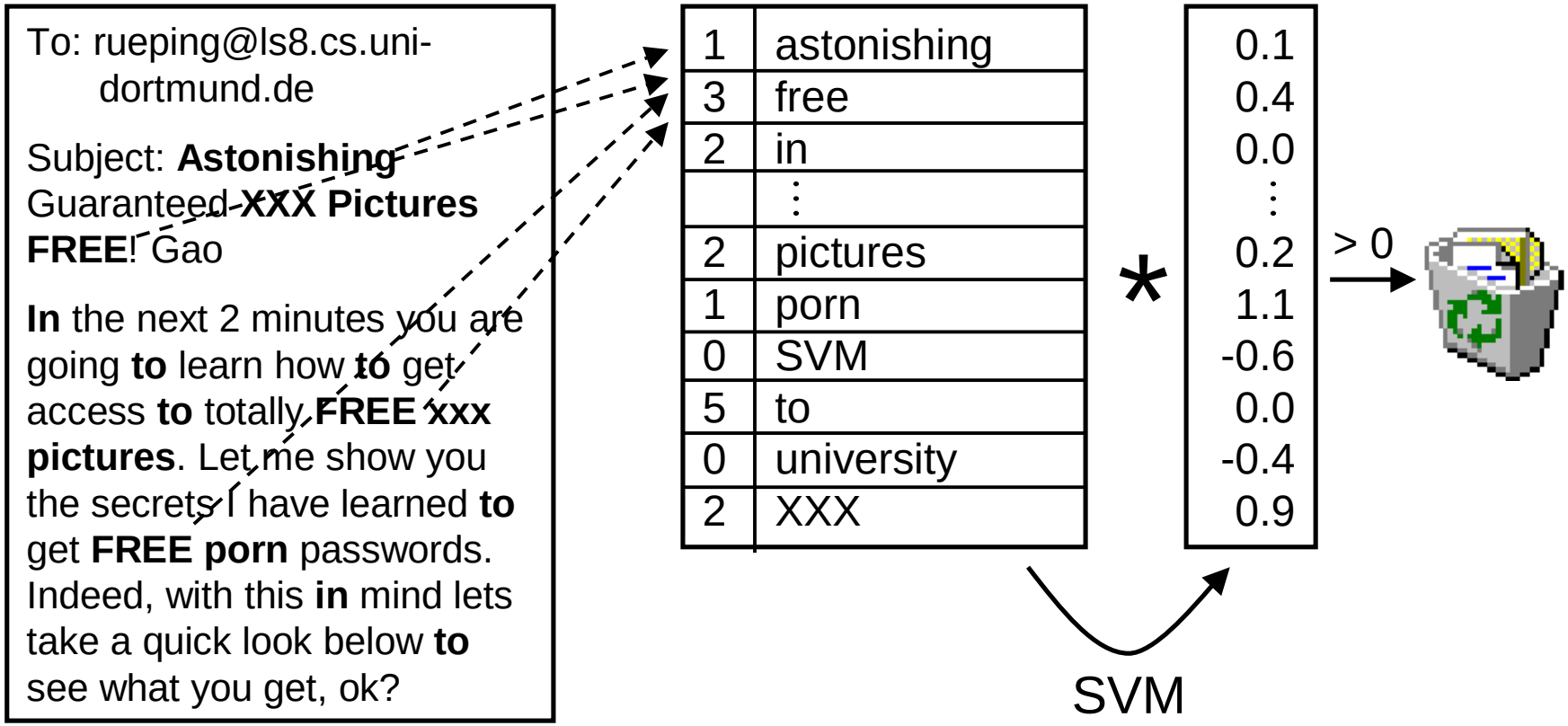
    s = update(s, α‘) // s = Gradient von W(α‘)

- Gradientensuchverfahren
- Trick: Stützvektoren allein definieren Lösung
- Weitere Tricks: Shrinking, Caching von $x_i * x_j$

Was wissen wir jetzt?

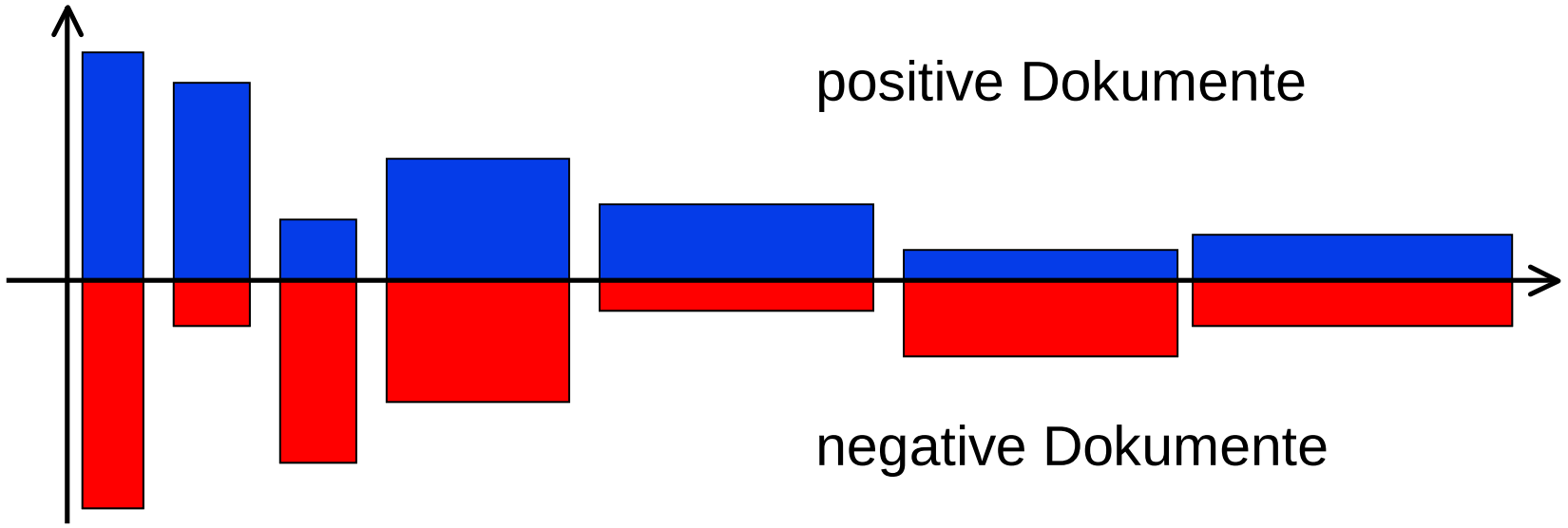
- Funktionslernen als allgemeine Lernaufgabe
- Minimierung des empirischen Risikos als Lösungsstrategie
- Optimale Hyperebene präzisiert die ERM
- Praxis: weich trennende Hyperebene
- Berechnung mittels SVM und dualem Problem
- **Offene Fragen:** Generelles Prinzip hinter der optimalen Hyperebene? Nicht lineare Daten?

Beispiel: Textklassifikation



TCat-Modell

- Typische Dimension: 10.000 – 100.000
- SVM lernt ohne Vorauswahl von Wörtern!
- **T**ext-**Cat**egorisierungs-Model:



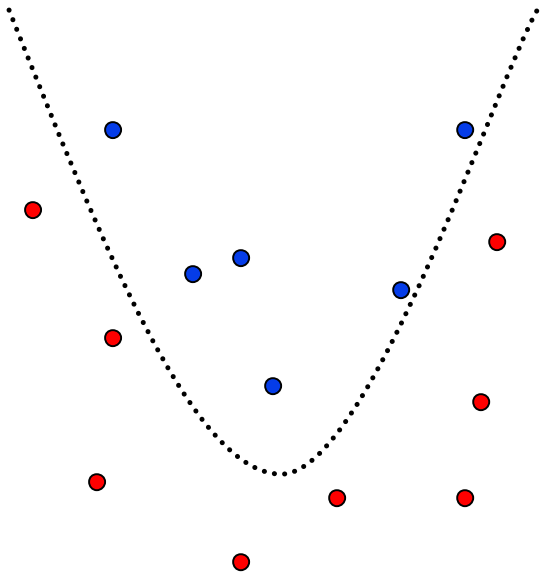
Beispiel: Intensivmedizin

$$f(x) = \left[\begin{pmatrix} 0.014 \\ 0.019 \\ -0.001 \\ -0.015 \\ -0.016 \\ 0.026 \\ 0.134 \\ -0.177 \\ \vdots \end{pmatrix} \begin{pmatrix} artsys = 174.00 \\ artdia = 86.00 \\ artmn = 121.00 \\ cvp = 8.00 \\ hr = 79.00 \\ papsys = 26.00 \\ papdia = 13.00 \\ papmn = 15.00 \\ \vdots \end{pmatrix} - 4.368 \right]$$

- Vitalzeichen von Intensivpatienten
- Alarm geben oder nicht?
- Hohe Genauigkeit
- Verständlichkeit?

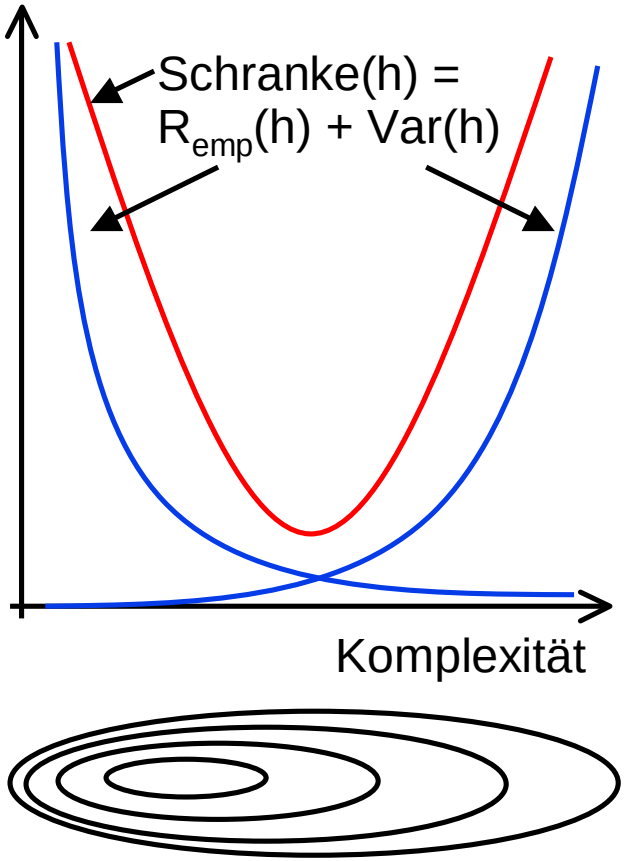
Bias-Varianz-Problem

- Zu kleiner Hypothesenraum: Zielfunktion nicht gut genug approximierbar (Bias)
- Zu großer Hypothesenraum: Zuviel Einfluß zufälliger Abweichungen (Varianz)
- Lösung: Minimiere obere Schranke des Fehlers:
  $R(h) \leq_{\eta} R_{emp}(h) + Var(h)$



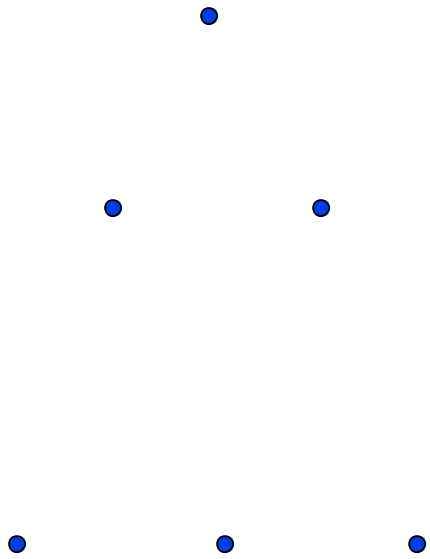
Strukturelle Risikominimierung

1. Ordne die Hypothesen in Teilmenge gemäß ihrer Komplexität
2. Wähle in jeder Teil-menge die Hypothese mit dem geringsten empirischen Fehler
3. Wähle insgesamt die Hypothese mit minimaler Risikoschranke



Vapnik-Chervonenkis-Dimension

- Definition: Eine Menge H von Hypothesen *zerschmettert* eine Menge E von Beispielen, wenn jede Teilmenge von E durch ein $h \in H$ abgetrennt werden kann.
- Definition: Die *VC-Dimension* einer Menge von Hypothesen H ist die maximale Anzahl von Beispielen E, die von H zerschmettert wird.



VC-Dimension von Hyperebenen

Satz: Die VC-Dimension der Hyperebenen im R^n ist n+1.

Beweis:

- VCdim(R^n) $\geq$ n+1: Wähle $x_0 = 0$ und $x_i = (0,\ldots,0,1,0,\ldots 0)$. Für eine beliebige Teilmenge A von $(x_0,\ldots,x_n)$ setze $y_i = 1$, falls $x_i \in A$ und $y_i = -1$ sonst. Definiere $w = \sum y_k x_k$ und $b = y_0/2$. Dann gilt $wx_0+b = y_0/2$ und $wx_i+b = y_i+y_0/2$. Also: wx+b trennt A.
- VCdim(R^n) $\leq$ n+1: Zurückführen auf die beiden Fälle rechts.

VC-Dim. und Anzahl der Parameter

- Setze $f_\alpha(x) = \cos(\alpha x)$ und $x_i = 10^{-i}$, i=1...l. Wähle $y_i \in \{-1,1\}$. Dann gilt für $\alpha = \pi(\sum {}^1/_2 (1-y_i)10^i)$:

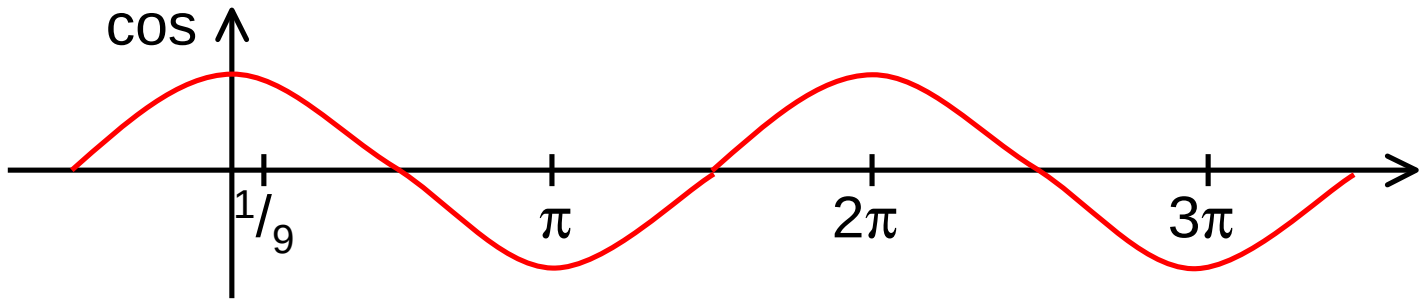
$$\alpha x_k = \pi\left(\sum_{i=1}^{l} \tfrac{1}{2}(1-y_i)10^i\right)10^{-k} = \pi\left(\sum_{i=1}^{l} \tfrac{1}{2}(1-y_i)10^{i-k}\right)$$

$$= \pi\left(\underbrace{\sum_{i=1}^{k-1} \tfrac{1}{2}(1-y_i)10^{i-k}}_{0 \le \sum\ldots \le 10^{-1}+10^{-2}+\ldots = 1/9} + \tfrac{1}{2}(1-y_k) + \underbrace{\sum_{i=k+1}^{l} \tfrac{1}{2}(1-y_i)10^{i-k}}_{\text{Vielfaches von 2}}\right)$$

(geometrische Reihe)

VC-Dim. und Anzahl der Parameter

$\Rightarrow$ $\cos(\alpha x_k)=\cos(\pi z)$ mit $z \in [0,{}^1/_9]$ für $y_k=1$ und $z \in [1,{}^{10}/_9]$ für $y_k=-1$



$\Rightarrow$ $\cos(\alpha x)$ zerschmettert $x_1,\ldots x_l$

$\Rightarrow$ $\cos(\alpha x)$ hat unendliche VC-Dimension

$\Rightarrow$ Die VC-Dimension ist unabhängig von der Anzahl der Parameter!

VC-Dimension der SVM

- Gegeben seien Beispiele $x_1,\ldots,x_l \in \Re^n$ mit $||xi|| < D$ für alle i. Für die VC-Dimension der durch den Vektor w gegebenen optimalen Hyperebene h gilt:

  $$VCdim(h) \leq \min\{D^2 ||w||^2, n\}+1$$

- Die Komplexität einer SVM ist nicht nur durch die Dimension der Daten beschränkt (Fluch der hohen Dimension), sondern auch durch die Struktur der Lösung!

Wozu die ganze Theorie?

Löse ich überhaupt das Lernproblem? SRM garantiert dies!

Empirisches Risiko $R_{emp}(h)$ $\xrightarrow[n \to \infty]{h \text{ fest}}$ Erwartetes Risiko $R(h)$

ERM
$h : R_{emp}(h) = \min_{h'} R_{emp}(h')$ $\not\xrightarrow[n \to \infty]{}$ Optimale Hypothese
$h : R(h) = \min_{h'} R(h')$

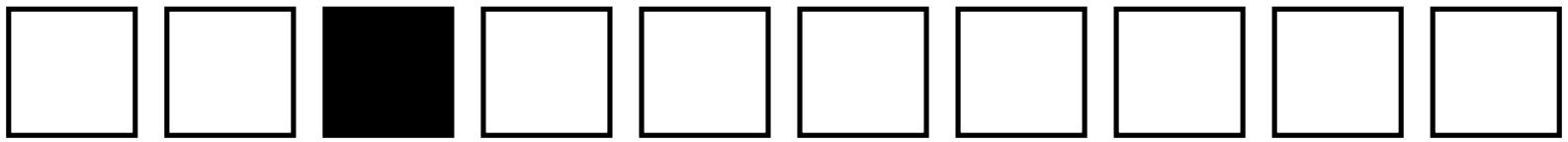
SRM
$h : R_{srm}(h) = \min_{h'} R_{srm}(h')$ $\xrightarrow[n \to \infty]{}$ Optimale Hypothese
$h : R(h) = \min_{h'} R(h')$

Was wissen wir jetzt?

- Idee der strukturellen Risikominimierung:
  - obere Schranke für das Risiko
  - Schrittweise Steigerung der Komplexität
- Formalisierung der Komplexität: VC-Dimension
- SRM als Prinzip der SVM
- Garantie für die Korrektheit der Lernstrategie
- Offene Frage: Nützlichkeit für die Praxis?

Performanzschätzer

- Welches erwartete Risiko R(h) erreicht SVM?
- R(h) selbst nicht berechenbar
- Trainingsfehler (zu optimistisch – Overfitting)
- Obere Schranke mittels VC-Dimension (zu locker)
- Kreuzvalidierung / Leave-One-Out-Schätzer (ineffizient)

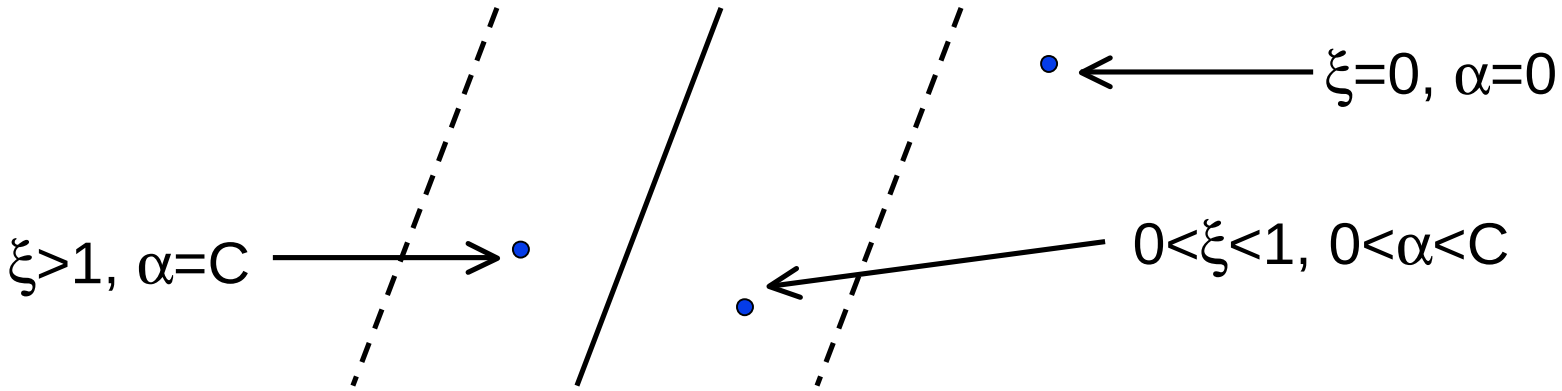


Performanzschätzer II

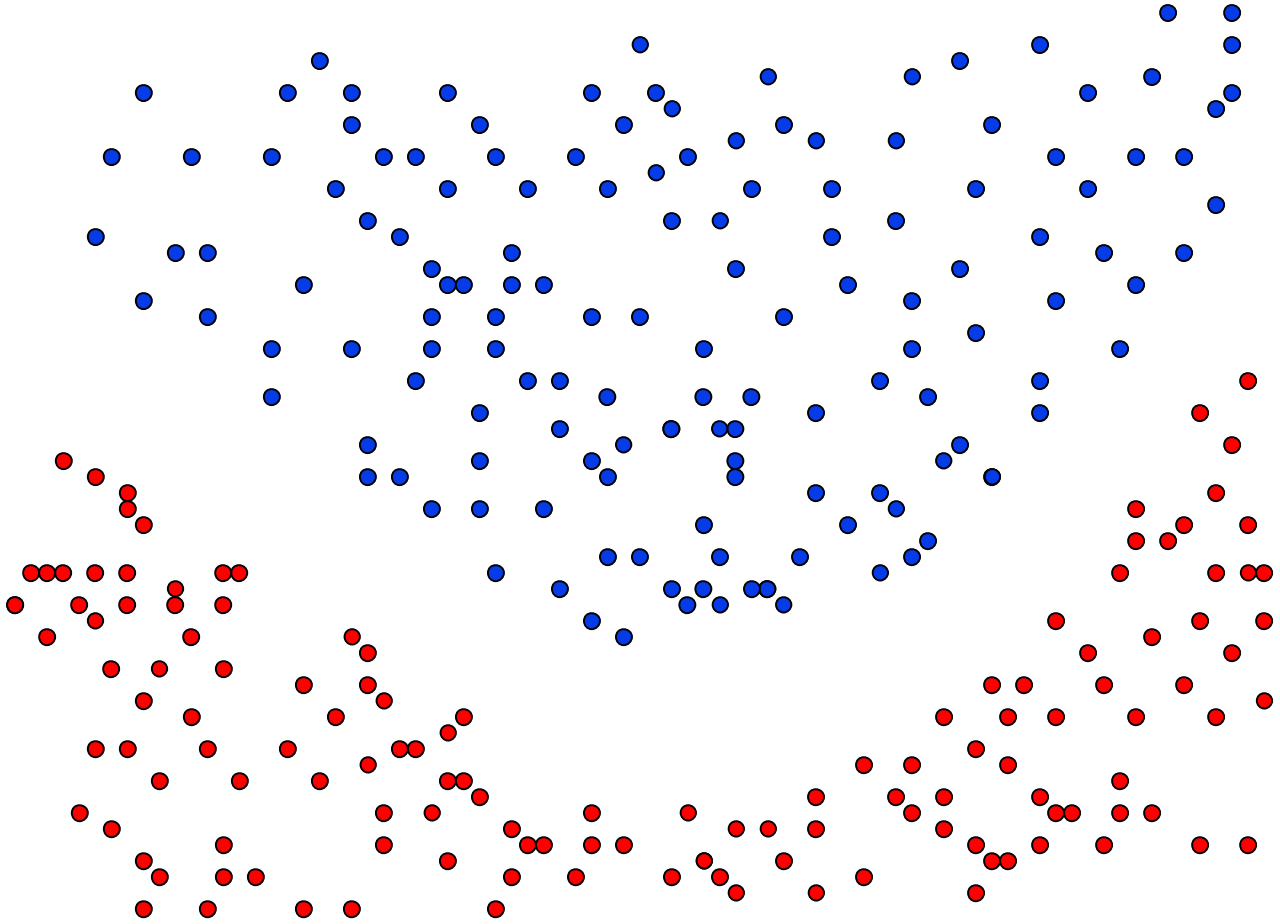
- Satz: Der Leave-One-Out-Fehler einer SVM ist beschränkt durch $R_{l1o} \leq |SV| / n$
- Beweis: Falsch klassifizierte Beispiele werden Stützvektoren. Also: Nicht-Stützvektoren werden korrekt klassifiziert. Weglassen eines Nicht-Stützvektors ändert die Hyperebene nicht, daher wird es auch beim l1o-Test richtig klassifiziert.

Performanzschätzer III

- Satz: Der Leave-One-Out-Fehler einer SVM ist beschränkt durch $R_{l1o} \leq |\{i : (2\alpha_i D^2+\xi_i)\geq 1\}| / n$ (D = Radius des Umkreises um die Beispiele).
- Beweis: Betrachte folgende drei Fälle:



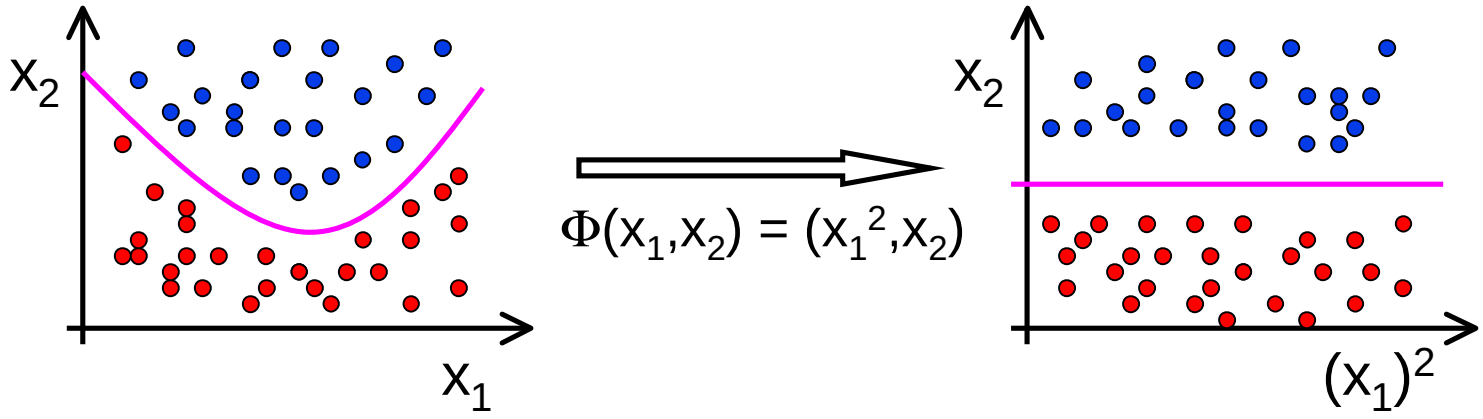
Nicht-lineare Daten



Nicht-lineare Daten

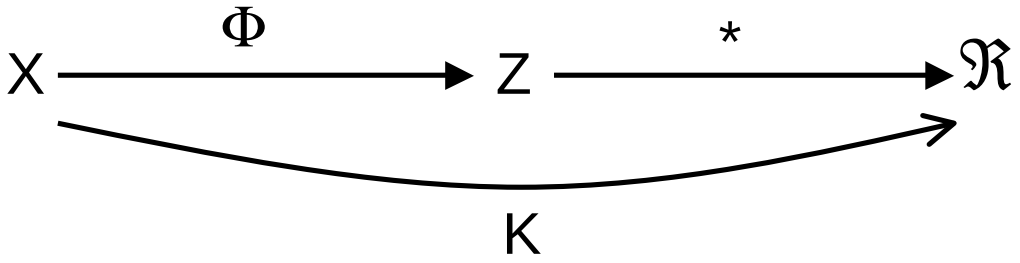
Was tun?

- Neue SVM-Theorie entwickeln? (Neeee!)
- Lineare SVM benutzen? (*„If all you've got is a hammer, every problem looks like a nail"*)
- Transformation in lineares Problem!



Kernfunktionen

- Erinnerung: $$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \left(x_i * x_j \right)$$
$$f(x) = \sum \alpha_i y_i (x_i * x) + b$$
- SVM hängt von x nur über Skalarprodukt x*x' ab.
- Ersetze Transformation Φ und Skalarprodukt * durch Kernfunktion $K(x_1,x_2) = \Phi(x_1)*\Phi(x_2)$

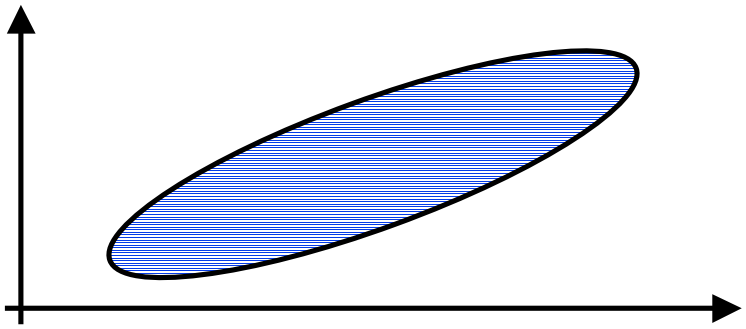


Kernfunktionen II

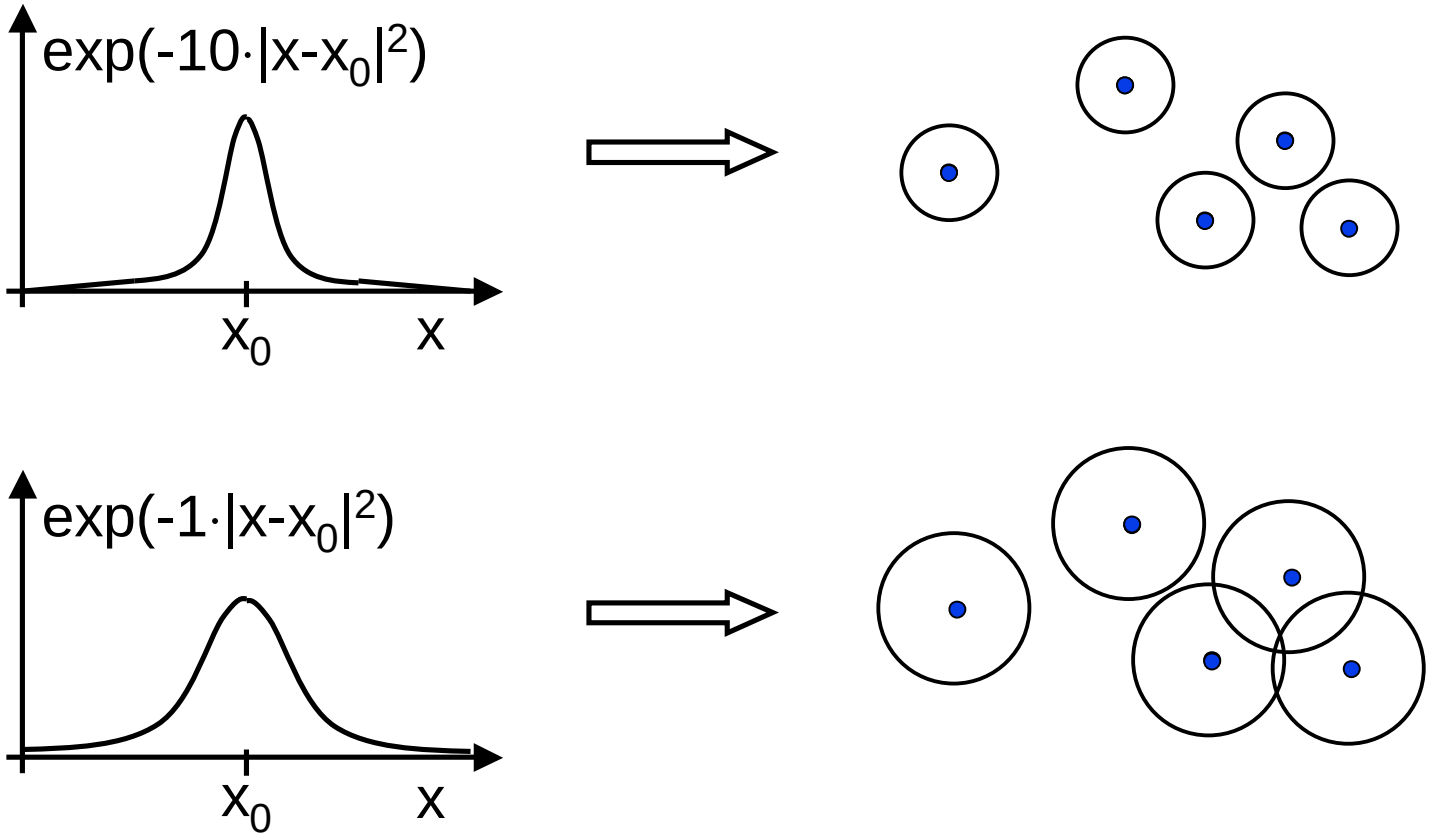
- Angabe von Φ nicht nötig, einzige Bedingung: Kernmatrix $(K(x_i,x_j))_{i,j=1...n}$ muss positiv definit sein.
- Radial-Basisfunktion: $K(x,y) = \exp(-\gamma||x-y||^2)$
- Polynom: $K(x,y) = (x*y)^d$
- Neuronale Netze: $K(x,y) = \tanh(a \cdot x*y+b)$
- Konstruktion von Spezialkernen durch Summen und Produkte von Kernfunktionen, Multiplikation mit positiver Zahl, Weglassen von Attributen

Polynom-Kernfunktionen

- $K_d(x,y) = (x*y)^d$
- Beispiel: d=2, $x,y \in \Re^2$. $K_2(x,y) = (x*y)^2$
  $= ((x_1,x_2)*(y_1,y_2))^2 = (x_1y_1+x_2y_2)^2$
  $= x_1^2y_1^2+2x_1y_1x_2y_2+x_2^2y_2^2$
  $= (x_1^2,\sqrt{2}x_1x_2,x_2^2)*(y_1^2,\sqrt{2}y_1y_2,y_2^2)$
  $=: \Phi(x)*\Phi(y)$



RBF-Kernfunktion

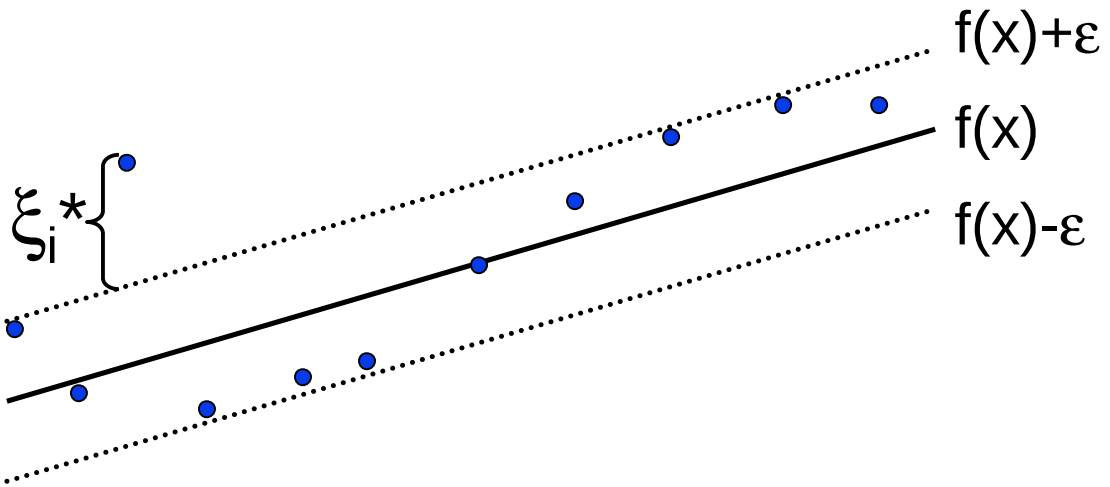


SVMs für Regression

- Minimiere $\|w\|^2 + C\left(\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\xi_i^*\right)$
- so dass für alle i gilt:

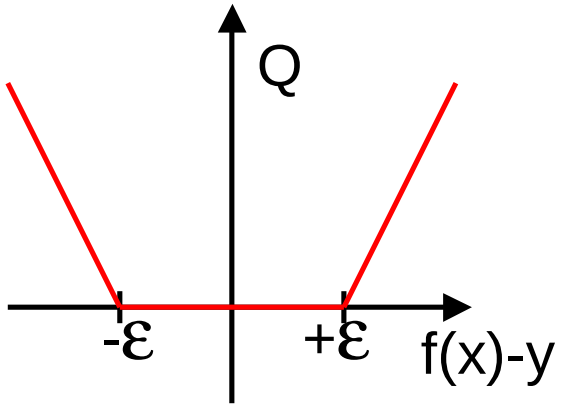
  $f(x_i) = w^*x_i+b \leq y_i +\varepsilon +\xi_i^*$ und

  $f(x_i) = w^*x_i+b \geq y_i - \varepsilon - \xi_i$

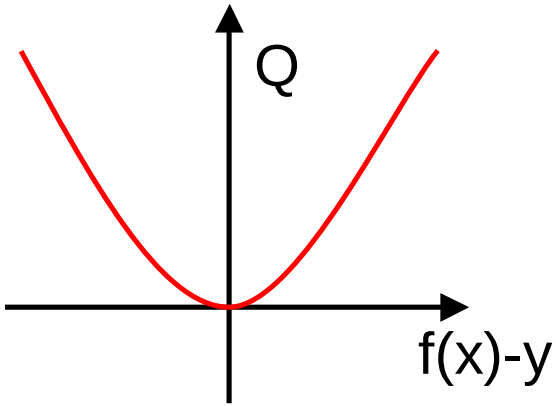


Verlustfunktion

lineare Verlustfunktion



quadratische Verlustfunktion



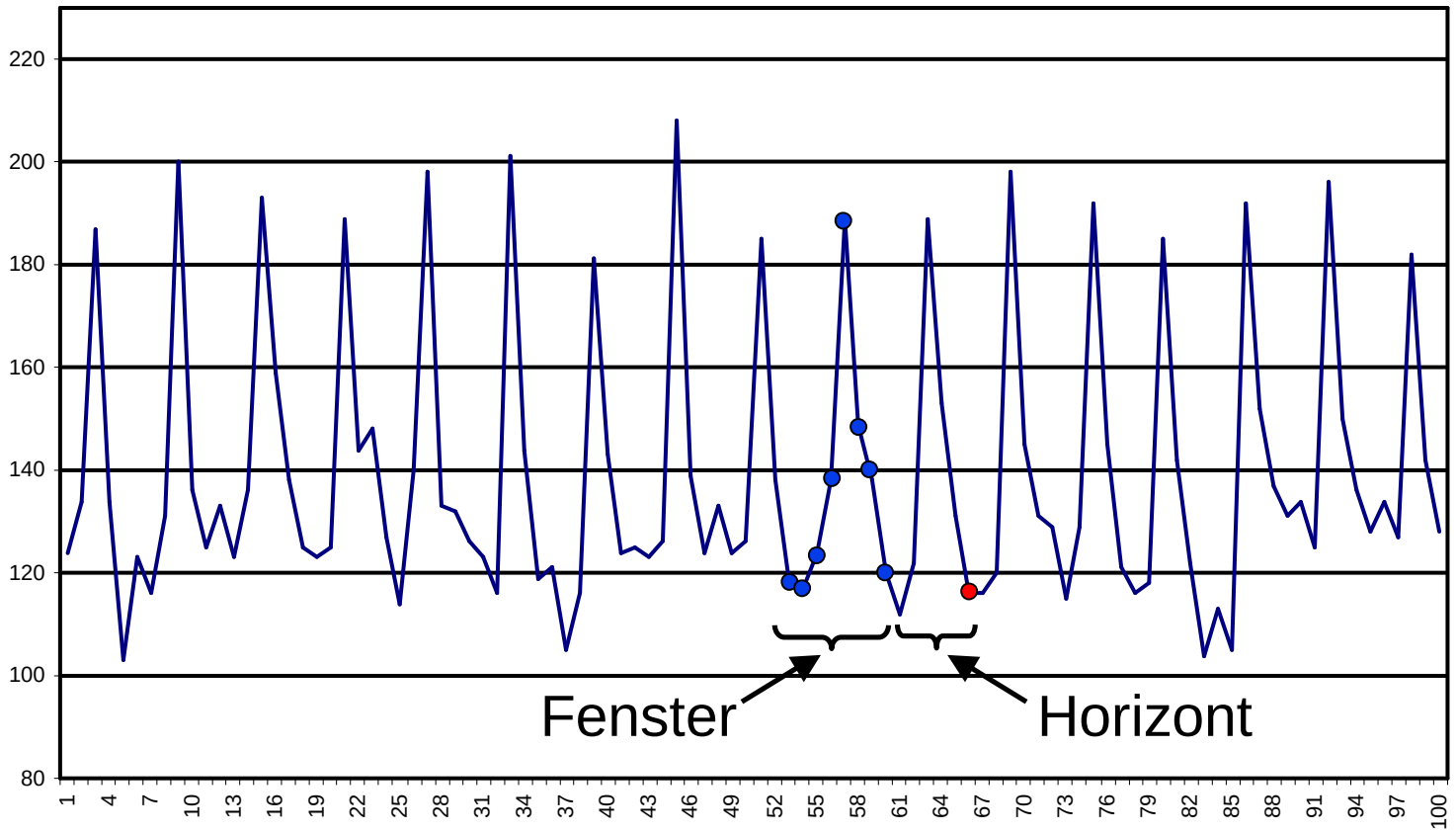
Duales Optimierungsproblem

- Maximiere

$$W(\alpha) = \sum_{i=1}^{n} y_i(\alpha_i^* - \alpha_i) - \varepsilon \sum_{i=1}^{n} (\alpha_i^* + \alpha_i) - \tfrac{1}{2} \sum_{i,j=1}^{n} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(x_i, x_j)$$

- unter $0 \leq \alpha_i, \alpha_i^* \leq C$ für alle i und $\sum \alpha_i^* = \sum \alpha_i$
- Mit $y_i \in \{-1,+1\}$, $\varepsilon=0$ und $\alpha_i=0$ für $y_i=1$ und $\alpha_i^*=0$ für $y_i=-1$, erhält man die Klassifikations-SVM!

Beispiel: Prognose von Zeitreihen



Prognose von Zeitreihen

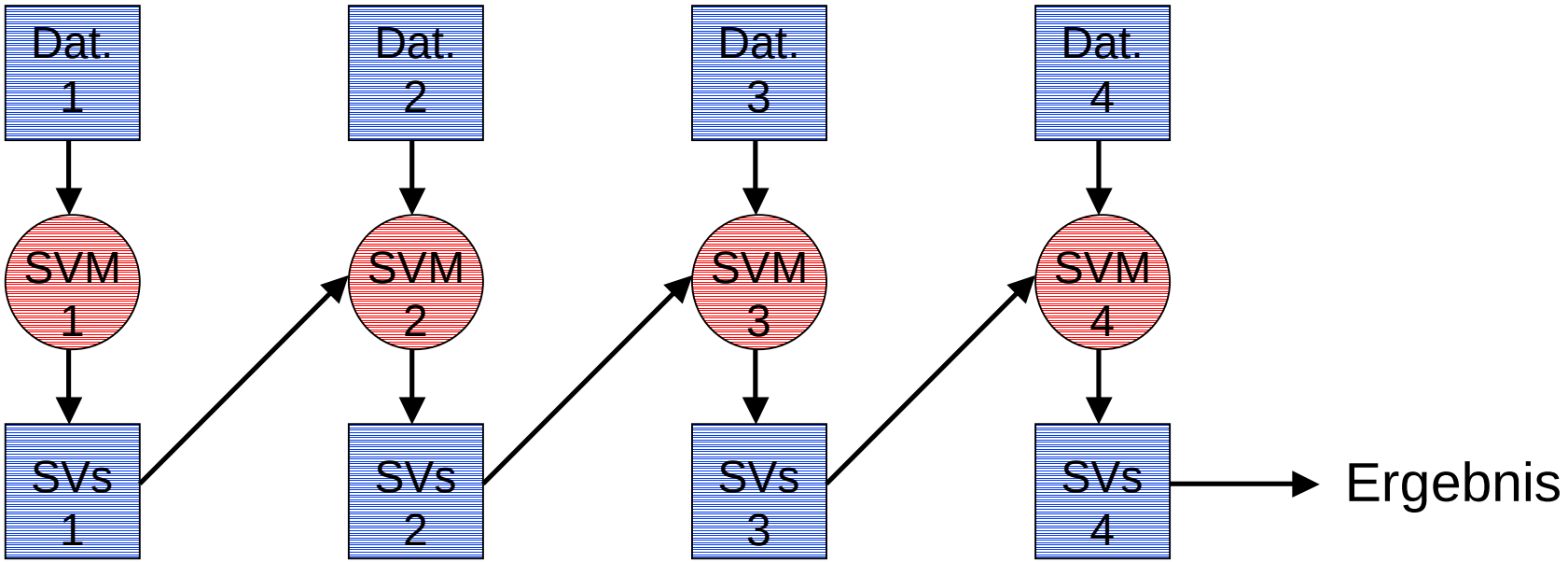
- Trend
- Zyklen
- Besondere Ereignisse (Weihnachten, Werbung, ...)
- Wie viele vergangene Beobachtungen?
- Ausreißer

SVMs und Datenbanken

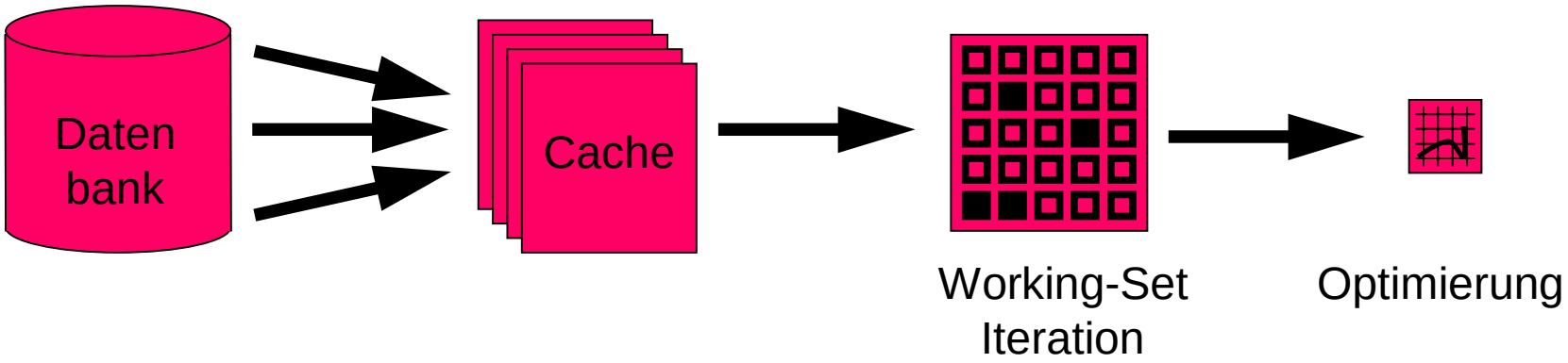
- Sehr große Datenmengen (mehrere GB)
- Datenbanken sind nicht auf numerische Operationen optimiert
- 3 Ansätze:
  - Daten aus der Datenbank herausziehen
  - Inkrementelles Lernen
  - Verarbeitung intern in Datenbank

Inkrementelle SVMs

- Platzbedarf der Kernmatrix quadratisch
- Idee: Schrittweises Lernen auf Teilmengen



SVMs in Datenbanken



- Zugriff auf Daten geschieht über Kernfunktion
- Cache der Zeilen der Kernmatrix
- Berechnung der Kernfunktion als SQL-Abfrage
- Beobachtung: Gebräuchliche Kernfunktionen basieren auf Skalarprodukt oder Abstand

Kernfunktion in SQL

- **SELECT**
  x1.att_1 * x2.att_1 + … + x1.att_d * x2.att_d
  FROM examples_table x1, examples_table x2
  WHERE x1.index = i and x2.index = j
- **SELECT <*kernel term*>**
  FROM examples_table x1, examples_table x2
  WHERE x1.index = I
- **SELECT <*kernel term*>**
  FROM examples_table x1, examples_table x2,
  free_examples f
  WHERE x1.index = i AND x2.index = f.index
- Weitere Optimierung durch Ausnutzen der relationalen Struktur im Cache möglich.

Was man über SVMs wissen muss

- Funktionenlernen – ERM – SRM
- Optimale Hyperebene: Definition, Berechnung, harte und weiche Trennung
- Nicht-Linearität durch Kernfunktionen
- Idee der SRM, VC-Dimension
- Schätzung der Performanz
- Idee der Regressions-SVM