

# Lerntheorie, Algorithmenunabhängige Verfahren (für überwachtes induktives Lernen)

Prof. Dr.-Ing. J. Marius Zöllner



Forschungszentrum Karlsruhe  
in der Helmholtz-Gemeinschaft



Universität Karlsruhe (TH)  
Forschungsuniversität • gegründet 1825

# Charakterisierung von Lernmaschinen

- Motivation:
  - Wieso brauchen wir immer neue Lernverfahren?
- Lernen = Optimierung?
  - Kann man Lernen formal beschreiben?
  - Fehler? Empirisch und Real?
  - Modellauswahl
  - Ensembles
- Lernbarkeit, Kapazität von Lernmaschinen
  - PAC – Lernbarkeit
  - Aussagen über Lernbarkeit, Stichprobenkomplexität
  - VC – Dimension
- Korrektes Lernen ?



**William of Occam**  
(1280/88-1347/49)  
Franziskaner

- Anklage wegen Ketzerei
- Entzug der Lehrerlaubnis

Razor principle (Rasiermesser - Prinzip) :

*Entia non sunt multiplicanda sine necessitate*  
(Entities should not be multiplied beyond necessity)



Löse nie ein Problem komplizierter als nötig,  
denn die einfachste, richtige Erklärung ist die Beste

## Definition

- Eine lernende Maschine wird bestimmt durch:
  - Hypothesenraum  $\{h_\alpha : \alpha \in A\}$
  - Lernverfahren: die Methode um  $\alpha_{opt}$  mit Hilfe von Lernbeispielen zu finden  
(benötigt Fehlerfunktion, Optimierungsmethode)
- Das resultierende (Entscheidungs) Modell  $M_{opt}$ 
  - ist gegeben durch die Auswertung der optimalen Hypothese  $h_{\alpha_{opt}}$ , die durch die lernende(n) Maschine(n) bestimmt wird

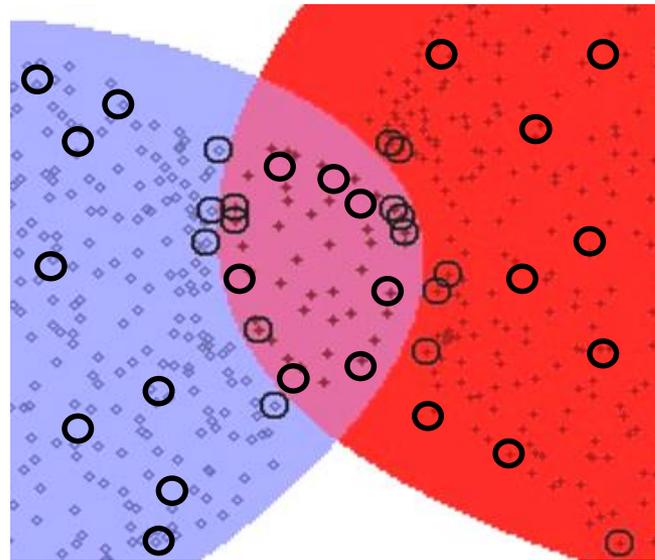
Problem: Welche Maschine ist zu wählen? (Welches Lernverfahren?)  
Welches Modell soll gewählt werden?

- Statistisches Problem: Das Verfahren betrachtet einen – gemessen an der Menge von Trainingsdaten – „zu großen“ Hypothesenraum
  - Auf Basis der Trainingsdaten eignen sich mehrere Hypothesen gleichermaßen gut
- Komplexitätsproblem: Aufgrund der Komplexität des Problems kann das Lernverfahren nicht das Finden einer optimalen Lösung innerhalb des Hypothesenraumes garantieren.
  - Bei Verwendung von speziellen Verfahren oder Heuristiken besteht die Gefahr einer suboptimalen Lösung
- Repräsentationsproblem: Der Hypothesenraum enthält keine ausreichend gute Approximationen der Zielfunktion/Konzept etc...
  - Das Lernverfahren kann einen gewünschten Approximationsgrad nicht liefern.

# Zwei einfache Beispiele

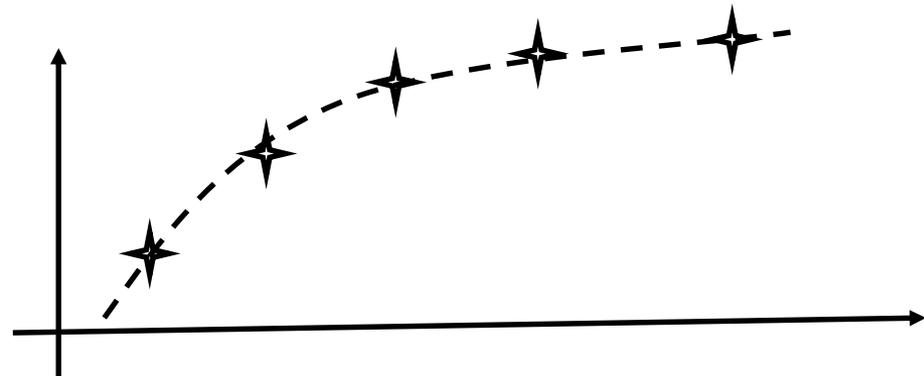
## Beispiel 1

- Klassifikation  $R^2$



## Beispiel 2

- Regression in  $R$   
- gestrichelte Kurve soll  
eingelernt werden



Was sieht man hier? Diskussion: Daten, Komplexität,  
Repräsentation  $\leftrightarrow$  Ergebnis, Fehler, Güte

# Überwachtes Lernen aus Beispielen

Lernen = Schätzen einer Abbildung (Hypothese) :

$$h_{\alpha}(\vec{x}) = y, \quad \alpha - \text{Parameter}$$

mit Hilfe von (bekannten) Beispielen (Lernbeispiele)

$$(\vec{x}_1, y_1) \dots (\vec{x}_n, y_n) \in X \times Y$$

generiert von einem (nicht bekannten) System  
mit der Wahrscheinlichkeit

$$P(\vec{x}, y)$$

Lernproblem = definiert durch  $X \times Y$

$\{Atr_1, \dots, Atr_n\} \times \{true, false\}$  - Konzept

$R^N \times \{Klasse_1, \dots, Klasse_n\}$  - Klassifikation (numerisch)

$R^N \times R$  - Regression/Prognose (numerisch)

# Lernen? Fehlerdefinition

Schätze  $h$  so, dass „der Fehler“  $E(h)$  minimal ist

$$E(h_\alpha) = \text{Func} ( h_\alpha(\vec{x}) \overset{\approx}{\longleftrightarrow} y ) , P(\vec{x}, y)$$

wobei  $\overset{\approx}{\longleftrightarrow}$  ein Vergleich zw. den Termen ist

Kann  $E$  berechnet werden?? → NEIN, unvollständige Daten

Kann  $E$  geschätzt werden ?? → JA mit verschiedenen Methoden

Lernen = Wie kann  $E$  effektiv minimiert werden?  
Abhängigkeiten? Methode? Maschine?

Problem: Schätzen oder Abschätzen von  $E(h)$

Distanz  $E(h) = \int l(h(\vec{x}), y) dP(\vec{x}, y)$

Probabilistisch  $E(h) \equiv P(h(\vec{x}) \neq y)$

→ Welche Distanzfunktion  $l$ ?

Beispiele für  $l$

# Missklassifik.  $l(h(\vec{x}), y) = \begin{cases} 1, & \text{wenn } h(\vec{x}) \neq y \\ 0, & \text{sonst} \end{cases}$

Abs. Fehler  $l(h(\vec{x}), y) = |h(\vec{x}) - y|$

Quad. Fehler  $l(h(\vec{x}), y) = (h(\vec{x}) - y)^2$

# Realer & Empirischer Fehler

Der reale Fehler

$$E(h) = \int l(h(\vec{x}), y) dP(\vec{x}, y)$$

bezüglich aller real vorkommenden Daten

ist leider nicht berechenbar → gute Schätzung nötig

Empirische Fehler

$$E_D(h) = \frac{1}{|D|} \sum_{(\vec{x}, y) \in D} l(h(\vec{x}), y)$$

wobei  $D$ : Lerndaten

→ Lernfehler

Verifikationsdaten

→ Verifikationsfehler

Testdaten

→ Generalisierungsfehler

# Lernen: Fehlerminimierung

Eine Lösung:

Definiere  $h_a$  und  
Finde beste  $\alpha$

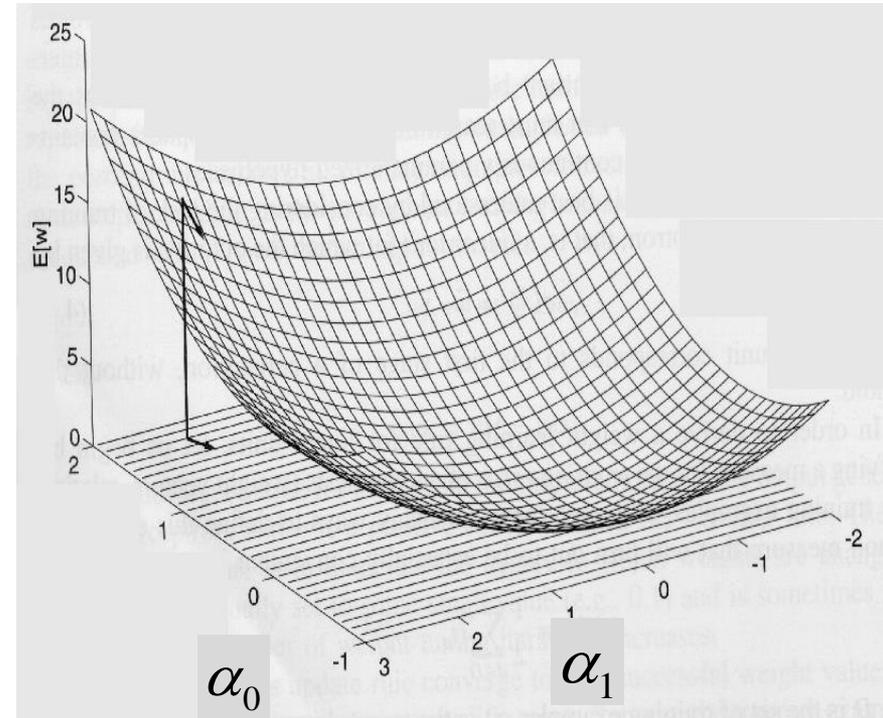
Durch iterative Minimierung  
des empirischen Fehlers

$$E_D(h_a)$$

z.B.: Gradientenabstieg auf  
der Fehlerfunktion

Gradient

$$\nabla E_D(\vec{\alpha}) \equiv \left[ \frac{\partial E_D(h_a)}{\partial \alpha_1}, \dots, \frac{\partial E_D(h_a)}{\partial \alpha_n} \right]$$

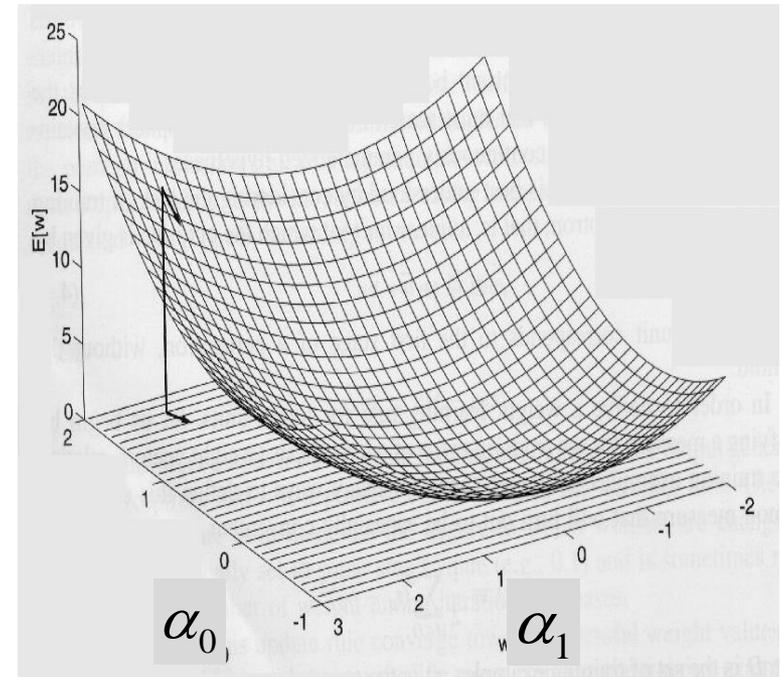


# Fehlerminimierung als Gradientenabstieg

Anpassung der Parameter

$$\vec{\alpha} \leftarrow \vec{\alpha} + \Delta\vec{\alpha}$$

$$\Delta\vec{\alpha} \approx -\eta\nabla E_D(\vec{\alpha}) \quad \eta - \text{Lernrate}$$



Frage: ist das korrekt?

Nun JA ..... ZUMINDEST oft hinreichend

# Overfitting

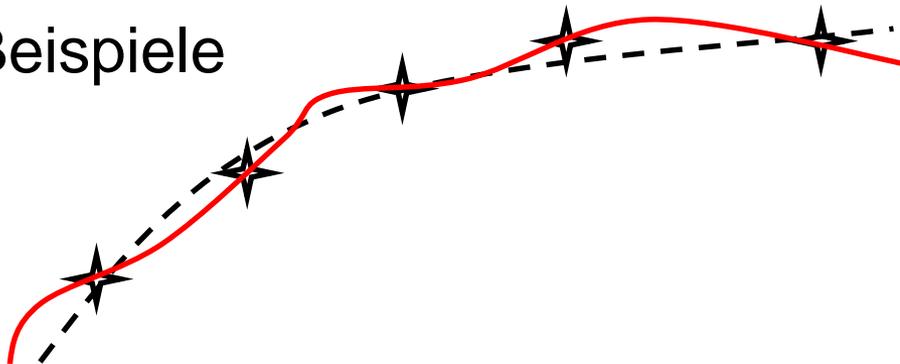
## ■ Definition:

Die Tendenz der Maschine sich beim Lernen auf die Lernbeispiele zu spezialisieren (Auswendig Lernen)

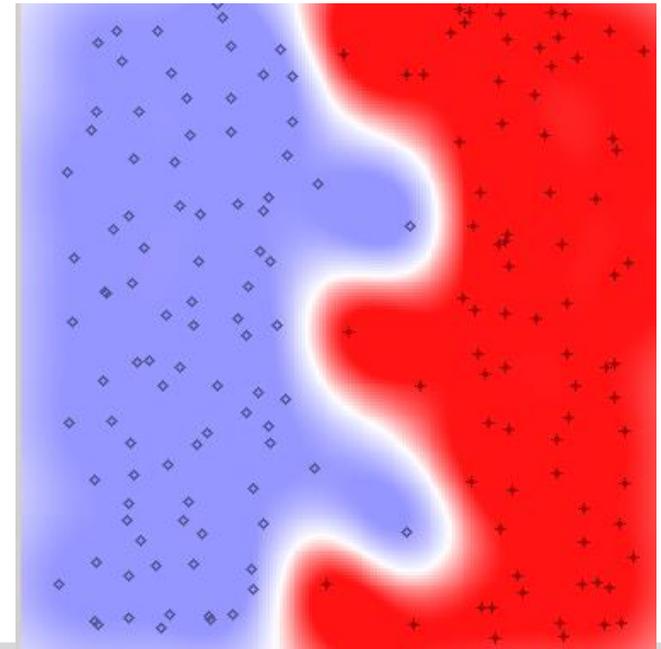
## ■ Formal: $h \in H$ – *overfit* $\Leftrightarrow \exists h' \in H$ s.d. $\forall \text{Testdaten}$

$$E_{\text{Lerndaten}}(h) < E_{\text{Lerndaten}}(h') \wedge E_{\text{Testdaten}}(h) > E_{\text{Testdaten}}(h')$$

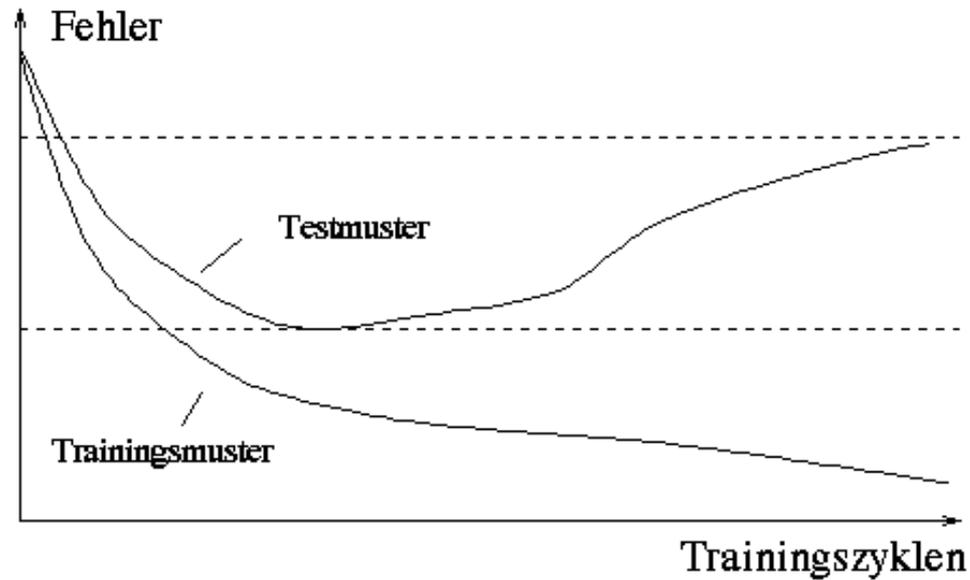
## ■ Beispiele



- Problematisch wenn verrauschte Lerndaten



→ Lernfehler fällt , Testfehler steigt, **Generalisierung fällt**



Erklärung:

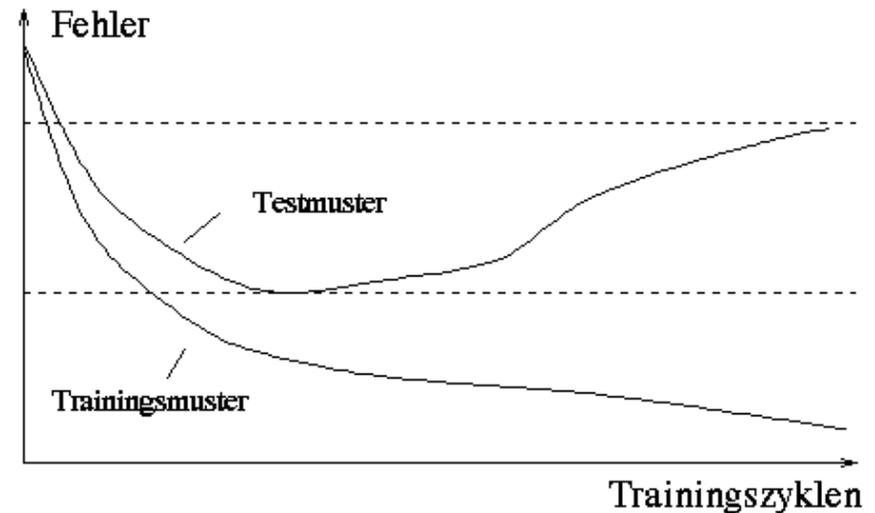
Lerndatenmenge und Testdatenmenge unterschiedlich (je nach Komplexität der Hypothese) → unterschiedlicher Verlauf von:

$$E_{Lern}(h) \quad \text{und} \quad E_{Verifikation}(h)$$

# Overfitting

Lösung:

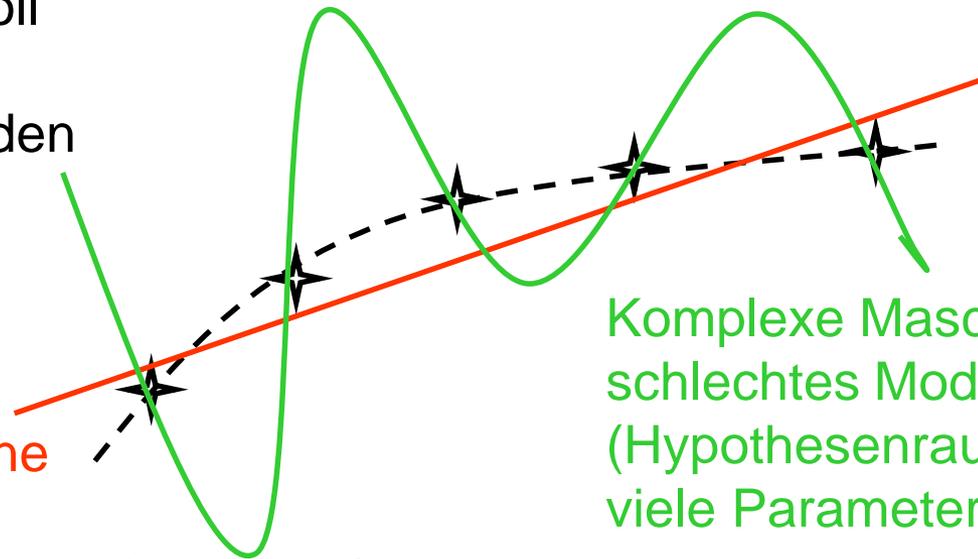
- Repräsentative Beispiele (Anzahl und Art/Verteilung)
- Lernprozess durch den Verifikationsfehler steuern (Verifikationsdatensatz kann anschließend nicht für die Gütebestimmung genutzt werden)



- Richtige Wahl und Suche der optimalen Hypothesen  $h_a$

Beispiel (Regressionsmodell):  
Die gestrichelte Kurve soll  
eingelernt werden.  
Es soll ein Modell gefunden  
werden.

wenig komplexe Maschine  
schlechtes Modell  
(Hypothesenraum: oft wenig Parameter)



Komplexe Maschine  
schlechtes Modell  
(Hypothesenraum: oft  
viele Parameter)

# Modellgüte - „Validierung“

- Eine Lösung: Cross – Validierung
  - Teile die Daten wiederholt in Lern- und Validierungsdaten
  - Bestimme darauf verschiedene Hypothesen (bzw. deren Parameter)
  - Berechne jeweils Generalisierung
  - Wiederhole
- n-fold-crossvalidation:
  - Zerlege die Menge der Lerndaten in n Mengen
  - Trainiere jew. auf n-1 Mengen, Teste auf 1 Menge
  - Wiederhole
- Leave One Out (Spezialfall)
  - Jeweils ein Beispiel für das Lernen weglassen
  - Addiere Fehler für weggelassene Beispiele
  - Wiederhole

➔ statistische Auswertung (Mittelwert, Varianz ...) auf verschiedenen Modellen auch mit rel. wenig Lerndaten evaluieren

- Grundgedanke: „Wie kann man mit einfachen Verfahren mehr erreichen?“
- In der Literatur oft - allgemeiner anekdotischer Hintergrund (Paradoxon)
  - Baron Münchhausen: am eigenen Schopf aus dem Sumpf gezogen
  - (für Informatiker) Computer startet zunächst ein einfaches Programm um dann „vereinfachte“ Programme zu starten → mächtige Leistung
- Und beim Lernen
  - Ziehe zufällig (mit Zurücklegen) aus  $D$  jeweils  $|D|$  Beispiele  $m$  Mal
  - Bestimme jeweils die Modell – Parameter
  - Wiederhole
  - Bestimme den Mittelwert, Varianz,... der Parameter des Modells

→ Analyse der Güte / Stabilität,

→ mit weiteren Ansätzen höhere Güte des Modells erreichen

# Bagging = Bootstrap aggregation

Variante des Bootstrap

+

Verwende mehrere Modelle

Verwende Bootstrap – Prinzip

ziehe  $n < |D|$  Beispiele (mit Zurücklegen)

Bestimme die jeweiligen Modelle

+

Kombiniere die Modelle, z.B. gewichtete Summe

→ Höhere Güte des Modells

→ Aussagen über die Stabilität des Modells  
(große Abweichungen in den einzelnen  
Modellen = Unstabil)

# Boosting für Klassifikation – ursprünglich Schapire 1990

Kombiniere „schwache“ Modelle um  
ein gutes Modell zu erhalten

Idee: Zerlege  $D$  in mehrere Datensätze, z.B. in  $D_1, D_2, D_3$

1. Wähle  $D_1$  und bestimme das Modell  $M_1$
2. Wähle für  $D_2$  aus  $D$  neue Beispiele s.d. 50% durch  $M_1$  korrekt geschätzt werden und erstelle damit  $M_2$
3. Wähle für  $D_3$  Beispiele bei denen  $M_1$  und  $M_2$  gegensätzlich sind und bestimme  $M_3$
4. Kombiniere die Modelle

$$M = \begin{cases} M_1, & \text{wenn } M_1 = M_2 \\ M_3, & \text{sonst} \end{cases}$$

# AdaBoost - Adaptive Boosting [Freund, Schapire 1996]

- Ausgangspunkt Lernmenge  $D$  mit  $n$  Beispielen
- Iteratives Erstellen eines komplexen Klassifikators in  $k$  Stufen (aus  $k$  zusammengesetzten Klassifikatoren)
- Ziehen von Lernbeispielen entsprechend definierter Gewichte
- Gewichtung der Lernbeispiele entsprechend dem Klassifikationsergebnis des zuletzt generierten „schwachen“ Klassifikators
  - Verringerung des Gewichts von korrekt klassifizierten Beispielen
  - Erhöhung des Gewichts von falsch klassifizierten Beispielen
- Mit der neuen Lernmenge wird mit Hilfe eines Lernverfahrens der nächste Klassifikator bestimmt.
- Die Klassifikatoren  $1, \dots, k$  werden zu einem Ensemble zusammengefasst und legen durch gewichteten Mehrheitsentscheid die Klasse eines Beispiels fest.

# AdaBoost – adaptive Boosting

Erweiterung des allgemeinen Boosting,  
Gewichtete Einzelmaschinen, Gewichtung Lernbeispiele

*begin* :  $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)\}$ ,  $W_1(i) = 1/n$  Gewicht pro Bsp.,  $k = 0$

1. *do* :  $k \leftarrow k + 1$

2. Trainiere  $M_k$  auf  $D_k$  ( $|D_k| = n$ , Bsp. gewählt abh. von  $W_k(i)$ )

3.  $E_k \leftarrow$  emp. Fehler von  $M_k$  (gewichtet bzgl.  $W_k(i)$ ,  $E_k < 0,5$  sonst Stopp)  
 Konzentration auf Fehler

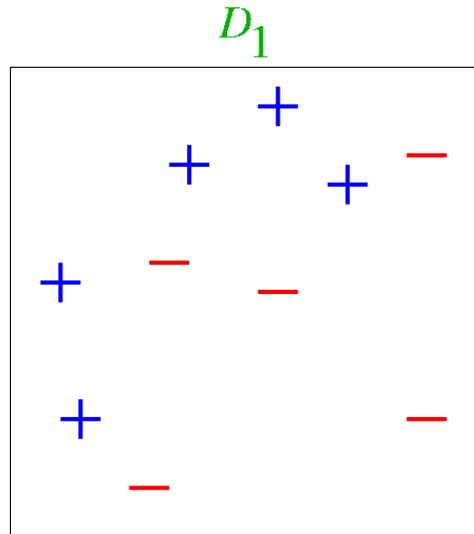
4.  $\alpha_k \leftarrow \frac{1}{2} [\ln((1 - E_k) / E_k)]$   
 1. Fehler  $\rightarrow$  größeres Gewicht und  
2. Größerer Fehler  $\rightarrow$  kleinere Erhöhung der Gewichte für Fehlerdaten

5.  $W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} \begin{cases} e^{-\alpha_k}, & \text{wenn } h_k(\vec{x}_i) = y_i \\ e^{\alpha_k}, & \text{wenn } h_k(\vec{x}_i) \neq y_i \end{cases}$ ,  $Z_k$  – Normalisierungskonst.

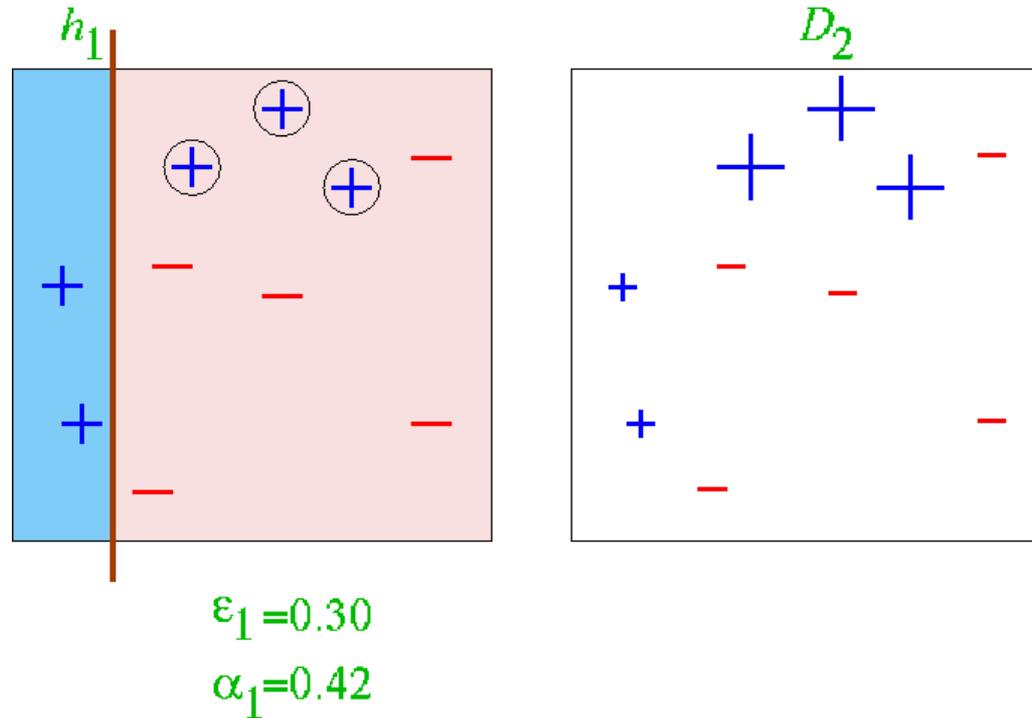
6. *until* :  $k = k_{\max}$

*Ergebnis* : Nutze  $M_k$  und  $\alpha_k$ , z.B. Entscheidung :  $\vec{x} \rightarrow \text{sign}(\sum_k \alpha_k h_k(\vec{x}))$

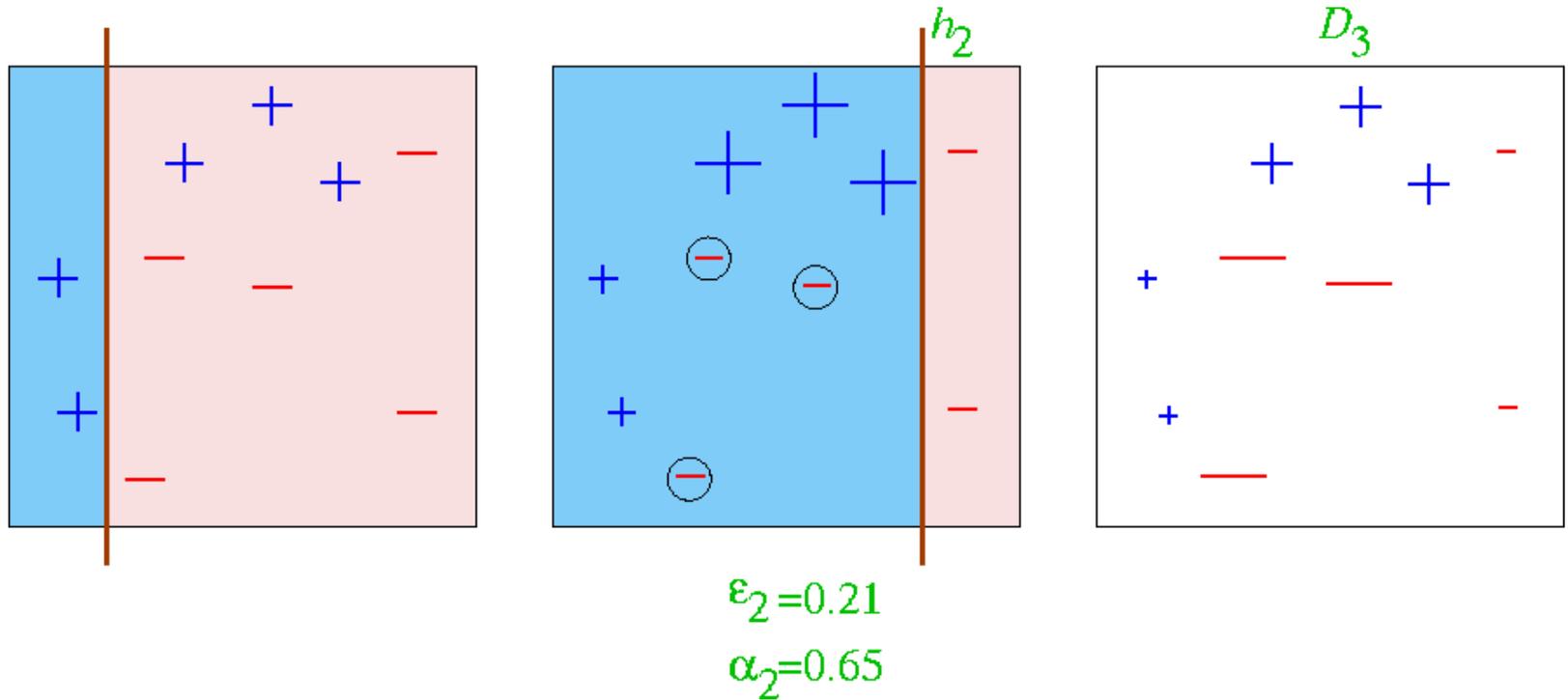
# Adaptive Boosting Beispiel



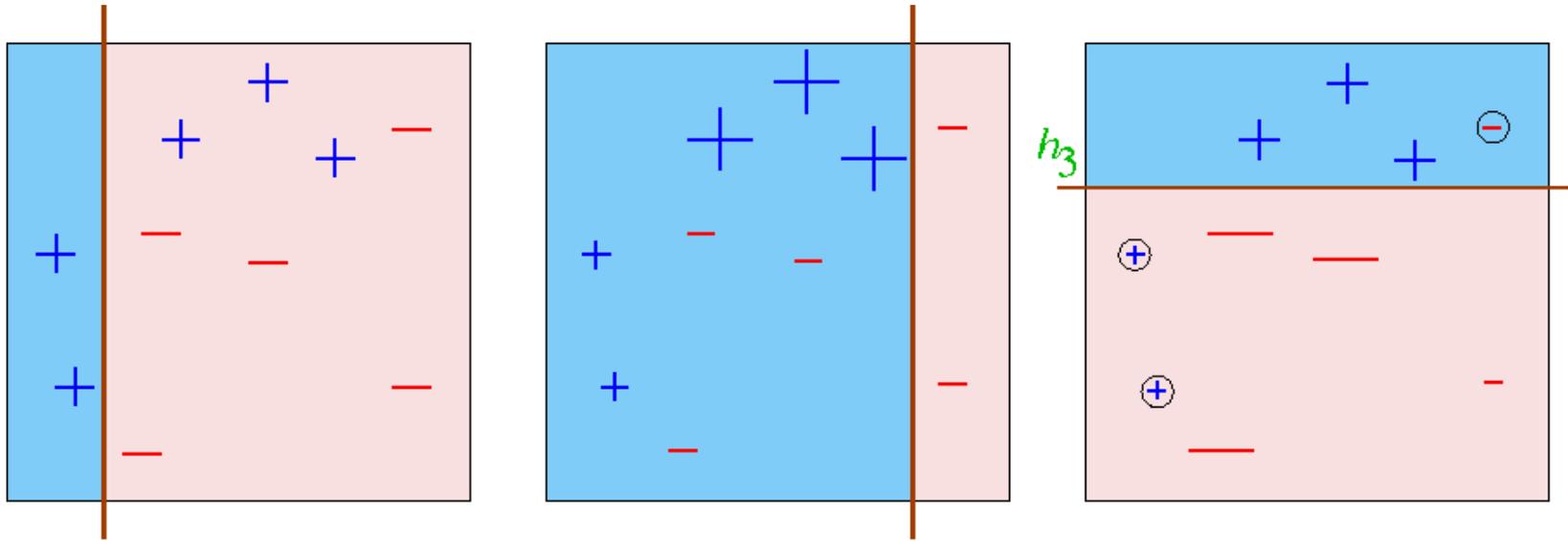
# Adaptive Boosting Beispiel – erster Klassifikator



# Adaptive Boosting Beispiel – zweiter Klassifikator



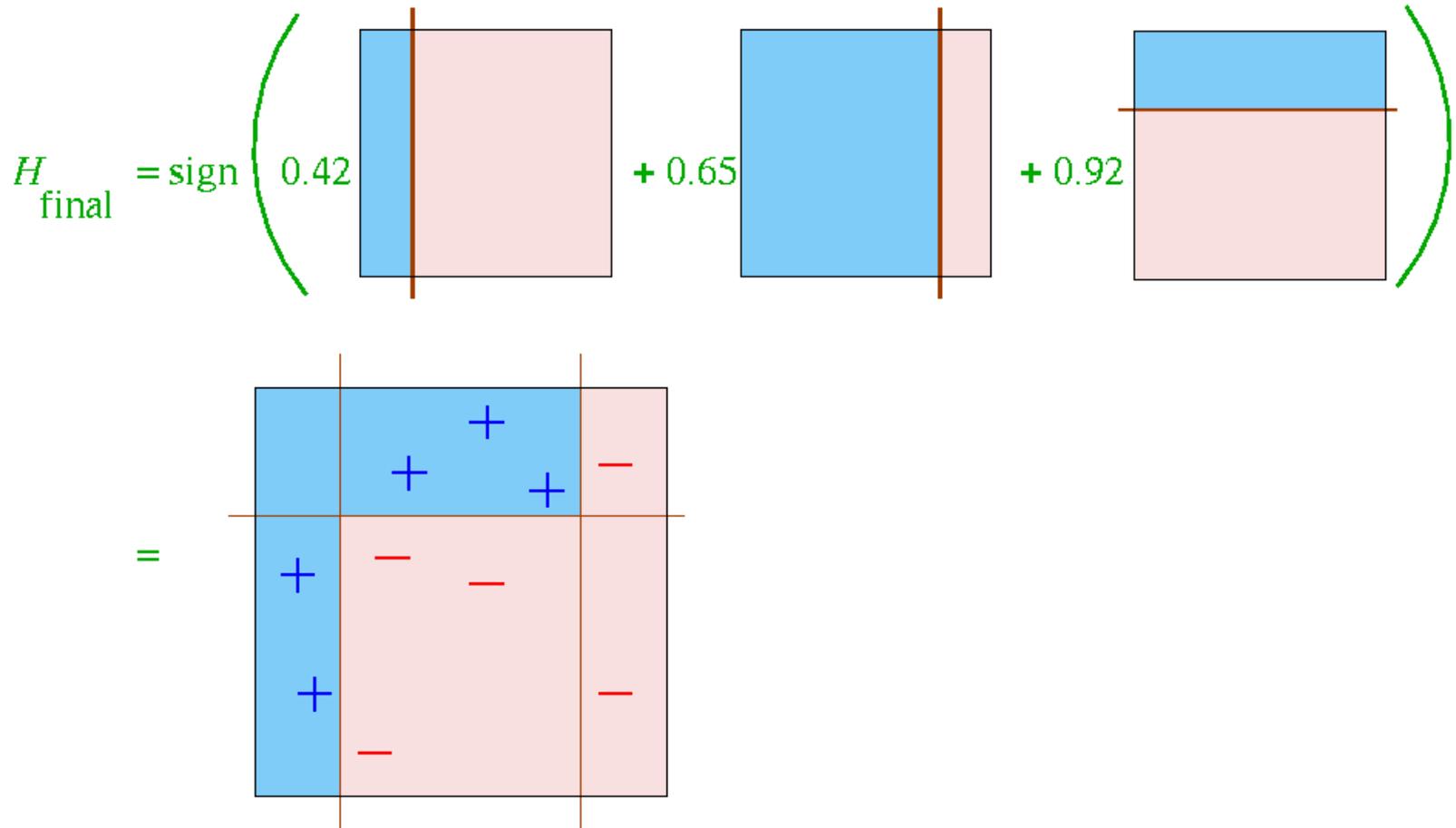
# Adaptive Boosting Beispiel – dritter Klassifikator



$$\epsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

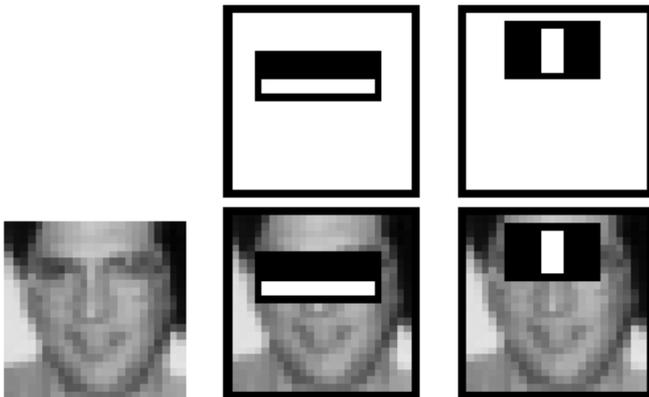
# Adaptive Boosting Beispiel – finaler Klassifikator



# Speziell: Viola & Jones Objekterkennung

## [~ 2001 - 2003]

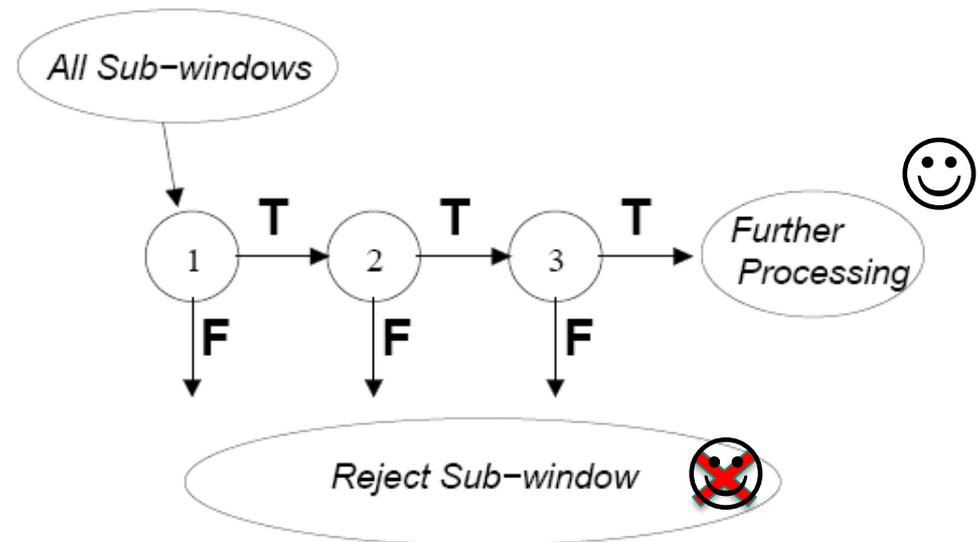
- Erkennen von Gesichtern in Bildern
- Sliding Window: Entscheiden für Ausschnitte (z.B. 24 x 24 Pixel) → Gesicht oder kein Gesicht
- Klassifikation: Merkmalsbasiert (Haar)



- naiv
  - z.B. 180000 Merkmale pro Ausschnitt
  - z.B.: 0.7 Sec pro Klassifikation pro Bild (384 x 288) bei 700MHz
- deutlich effizienter und definierbar gut
  - Trick 1: Kaskaden von Klassifikatoren einstellbarer „Güte“
  - Trick 2: Güte ausgewählter Merkmale einstellen durch Adaboost
  - .....

# Kaskadierung Viola & Jones 2001

- 2-Klassen-Problem
- Trick 1: Kaskade
  - Gesucht wird eine Folge von Klassifikatoren mit steigender Komplexität
  - Vorgehen: Aus der Datenmenge wird schrittweise eine Teilmenge verworfen (F - kein Gesicht), die von der bisherigen Kaskade mit T (potentiell Gesicht) klassifiziert wurde.

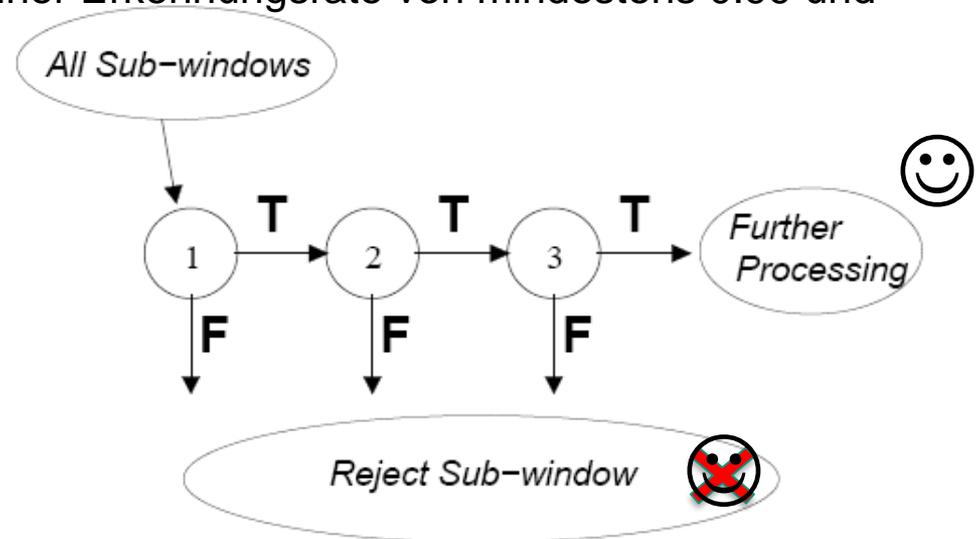


# Kaskadierung Viola & Jones 2001

- Für jeden Klassifikator definiere Mindestbedingungen
  - Erkennungsrate (hier: reales Gesicht in T)
  - Falsch Positiv Rate\* (hier: nicht Gesicht in T)
- Kaskade → Stufenweise Reduktion der Falsch Positiv Rate (aber Erhöhung der Falsch Negativ Rate und Reduktion der Erkennungsrate)
  - Für 10 Stufen in der Kaskade, einer Erkennungsrate von mindestens 0.99 und einer Falsch Positiv Rate von höchstens 0.3

erhält man für die Kaskade:

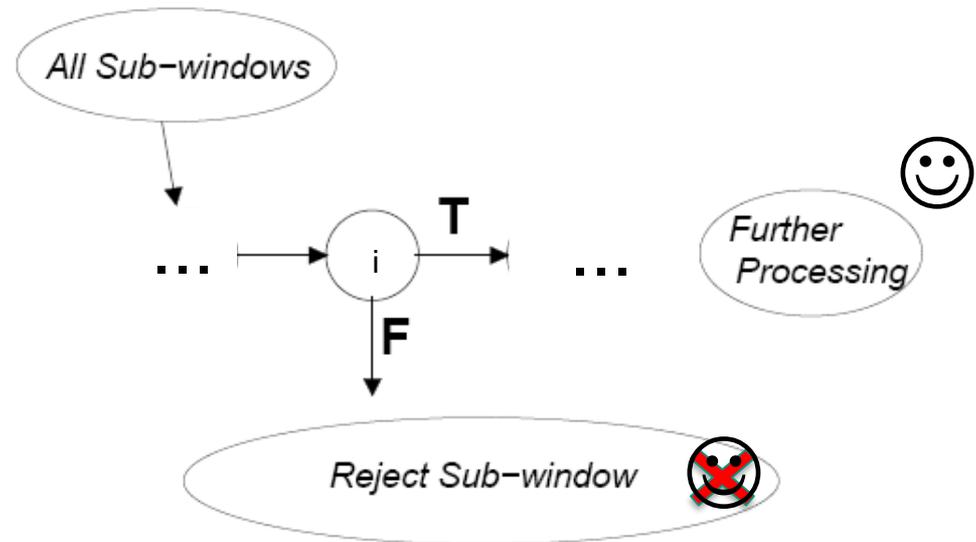
- eine Erkennungsrate von  $0.99^{10} \approx 0.9$  und
- eine Falsch Positiv Rate von höchstens  $0.3^{10} \approx 0.000006$



(\* siehe Folien ML2)

# Kaskadierung Viola & Jones 2001

- Für jeden Klassifikator definiere Mindestbedingungen
  - Erkennungsrate (hier: reales Gesicht in T)
  - Falsch Positiv Rate (hier: nicht Gesicht in T)
- Trick 2: Baue die einzelnen Klassifikatoren s.d. die Bedingungen jeweils erfüllt sind
- Methode: Adaboost (iterativ)
  - Kombiniere  $n$  einfache Schwellwert-Klassifikatoren anhand der Haarmerkmale
  - Welches  $n$  ? (Iteration)
  - Welches sind die besten Merkmale? (Adaboost)



# Kaskadierung – nach Viola & Jones

*begin* :  $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)\},$

*wähle*  $f, F_{soll}$  – Einzel u. Gesamt – Falsch positiv Rate,

*d* Detektionsrate (Erkennungsrate)

*while*  $F_i > F_{soll}$       ← Kaskade

$i = i + 1$

*while*  $F_i > fF_{i-1}$       ← AdaBoost-Klassifikator

$n = n + 1$

*trainiere*  $h_i$  mit  $n$  Merkmalen (AdaBoost)

*bestimme*  $F_i, D_i$

*(passe Anzahl und Schwellwerte  $\theta$  von  $h_i$  an*

*um für den Kaskadenklassifikator  $H_i$ ,*

*die Detektionsrate  $dD_{i-1}$  zu erreichen)*

# AdaBoost – nach Viola & Jones

*begin*:  $D = \{(\bar{x}_1, y_1), \dots, (\bar{x}_n, y_n)\}$ ,  $W_1(i) = \frac{1}{|P|} \cdot \frac{1}{|N|}$  Gewicht pro Bsp.,  $k = 0$

1. *do*:  $k \leftarrow k + 1$

2. *Trainiere für jedes Merkmal  $f$  eine Maschine auf  $D_k$*

( $|D_k| = n$ , Bsp. gewählt abh. von  $W_k(i)$ )

$$h(x) = \begin{cases} 1, & pf(x) > p\theta \\ 0 & \text{sonst} \end{cases}, \quad \theta - \text{Schwellwert}, \quad p - \text{Polarität } (\pm 1)$$

3. *Wähle  $M_k$  mit kleinstem  $E_k \leftarrow$  emp. Fehler von  $M_k$*

(gewichtet bzgl.  $W_k(i)$ )

4.  $\alpha_k \leftarrow \frac{1}{2} [\ln((1 - E_k) / E_k)]$

5.  $W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} \begin{cases} e^{-\alpha_k}, & \text{wenn } h_k(\bar{x}_i) = y_i \\ e^{\alpha_k}, & \text{wenn } h_k(\bar{x}_i) \neq y_i \end{cases}, \quad Z_k - \text{Normalisierungskonst}$

6. *until*:  $k = k_{\max}$

*Ergebnis*:  $M_k$  und  $\alpha_k \Rightarrow$  Entscheidung:  $\bar{x} \rightarrow \begin{cases} 1 & \sum_k \alpha_k h_k(\bar{x}) \geq \frac{1}{2} \sum_i \alpha_k \\ 0 & \text{sonst} \end{cases}$

# (Frühe) Ergebnisse von Viola Jones

- Datensatz 4916 – Gesichter , 10000 – nicht Gesichter
- 32 (38) – Kaskadenklassifikatoren, 4297 (6060) – Merkmale
- Die Teilklassifikatoren der einzelnen Kaskadenstufen haben entsprechend nach AdaBoost – Training:
  - 2 Merkmale – 60% „nicht Gesichter“ erkannt, 100% Gesicht behalten  
→ 40% Falsch Positiv
  - 5 Merkmale – 80 % „nicht Gesichter“ erkannt, ....  
→ 20% Falsch Positiv
  - 20 Merkmale - ....
  - 50 .....
  - 100
- Trainingszeit – Wochen auf 466MHz, AlphaStation XP900
- Aber Klassifikation 0,067 sec pro Bild (Faktor 10x schneller und definierte „Güte“)

Heute Training innerhalb  
von Stunden (je nach  
Anwendung)

PAC = Probably Approximate Correct

Gegeben: eine Menge  $X$  von Instanzen der Länge  $n$   
ein Konzept  $C$   
ein Hypothesenraum  $H$   
eine Lerndatenmenge  $D$

Kann eine korrekte Hypothese aus  $H$ ,  $h(\vec{x}) = c(\vec{x}), \forall \vec{x} \in D$   
gefunden werden?

- Nein

- aber eine  $\varepsilon$  - genaue:  $E_D(h) \leq \varepsilon, 0 < \varepsilon < \frac{1}{2}$

Approximate Correct

Kann diese sicher gefunden werden?

- Nein
- aber mit beliebiger Wahrscheinlichkeit

$$1 - \delta, \quad 0 < \delta < \frac{1}{2} \quad \text{Probably}$$

Wie ist das Problem (Finden der Hypothese) lösbar?

- in polynomialer Zeit abh. von:  $\frac{1}{\delta}, \frac{1}{\epsilon}, n$
- mit Speicheraufwand abh. von:  $C$

Und die Anzahl der benötigten Lerndaten (Stichprobenkomplexität) ist:

$$m \geq \frac{1}{\varepsilon} \left( \ln\left(\frac{1}{\delta}\right) + \ln|H| \right)$$

Und was heißt das?

- je größer die gewünschte Sicherheit
- je kleiner der zulässige Fehler
- je größer der Hypothesenraum
- um so größer die Anzahl der benötigten Daten

Für Hypothesen

- die aus Konjunktionen von
- bis zu 10 Literalen bestehen
- kann mit 95% Sicherheit eine
- Hypothese mit Fehler  $< 0.1$  gefunden werden

Dafür benötigt eine Lernmaschine mindestens

$$m \geq \frac{1}{0,1} \left( \ln\left(\frac{1}{0.05}\right) + 10 \ln|3| \right) = 140 \quad |H| = 3^{10}$$

Lernbeispiele

**Leider für komplexe Maschinen nicht so leicht**

# Vapnik-Chervonenkis (VC) Dimension

Eine Menge von Abbildungen (Hypothesen)

$\{h_\alpha : \alpha \in A\}$  definieren den Hypothesenraum  $H^\alpha$

Definition (für die Klassifikation):

Die VC Dimension  $VC(h_\alpha)$  von  $H^\alpha$  ist gleich der maximalen Anzahl von Datenpunkten (aus einer Menge  $S$ ) die von  $H^\alpha$  beliebig separiert werden können

# Vapnik-Chervonenkis (VC) Dimension

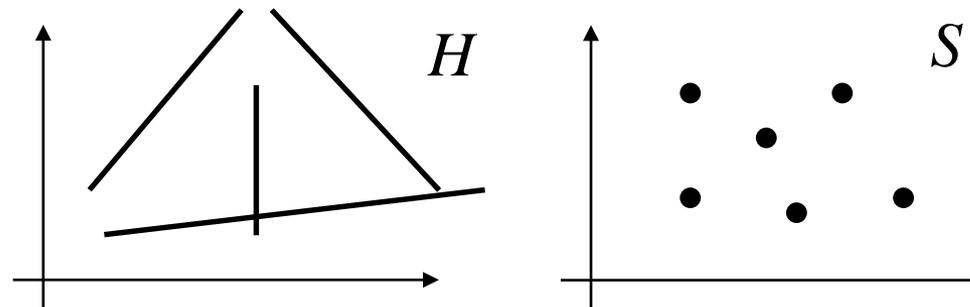
Definition (für die Klassifikation):

Eine Abbildung (Hypothese)  $h$  separiert die Daten aus  $S$  wenn durch  $h$  zwei Untermengen definiert werden :

$$\{x \mid h(x) = 0\} \quad \text{und} \quad \{x \mid h(x) = 1\}$$

Beispiel:

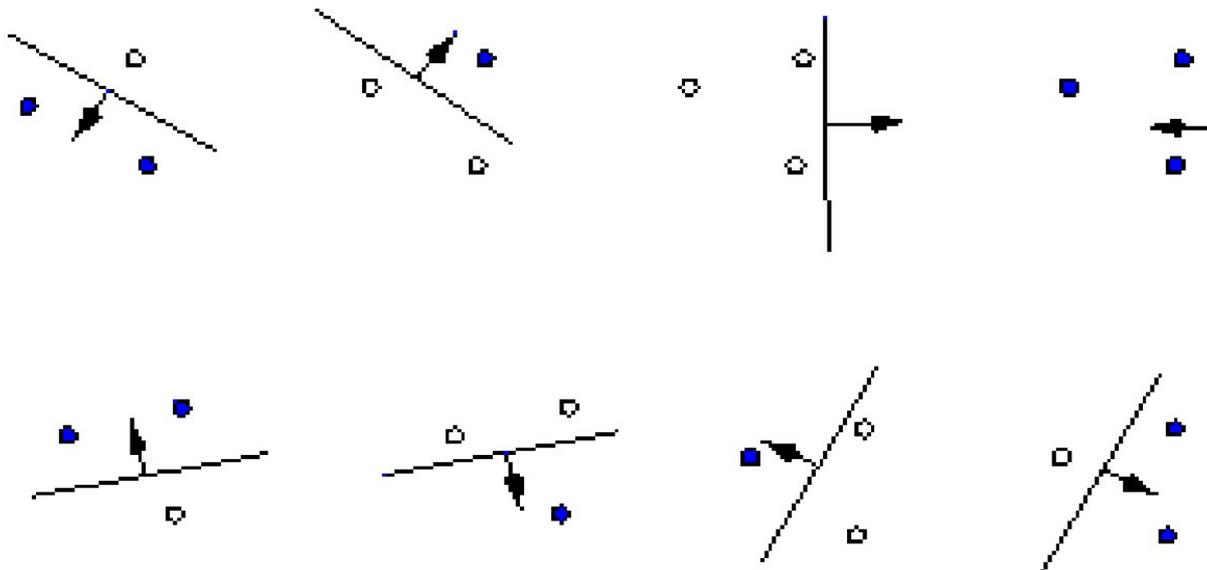
Hypothesenraum seien die Hyperebenen aus  $R^2$  (Geraden) und  $S \subset R^2$



# VC Dimension Beispiel

Behauptung:

Maximal 3 Werte können von Geraden separiert werden, wenn alle beliebigen Aufteilungen erlaubt sind



Allgemein: Hyperebene in  $R^n \Rightarrow n+1$  separierte Werte

VC Dimension auch erweiterbar auf Regressionsmodelle → Literatur

Maß für die Datenkomplexität des Lernens [Blumer et al 1988]

Aussagen über PAC - Stichprobenkomplexität, Anzahl von Lernbeispielen  $m$

$$m \geq \frac{1}{\varepsilon} \left( 4 \log_2 \left( \frac{2}{\delta} \right) + 8 VC(h) \log_2 \left( \frac{13}{\varepsilon} \right) \right)$$

Erheblich bessere Abschätzung, welche auch die Lernmaschine einbezieht

Es gibt weitere Beschränkungen für spezielle Maschinen .....

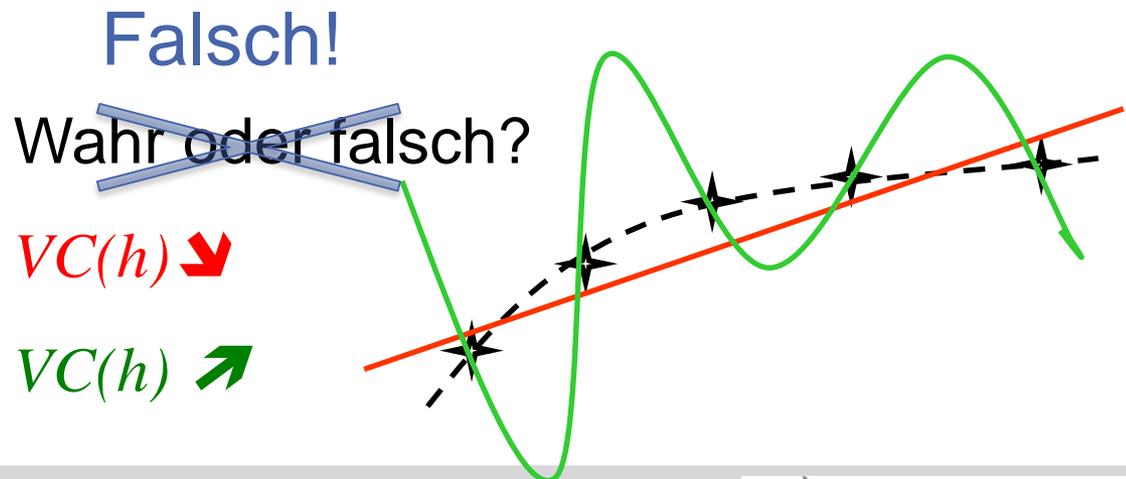
# VC Dimension – Nutzen II

$VC(h)$  = Maß für die Kapazität von lernenden Maschinen

Vermutung:

Je höher  $VC(h)$  um so besser kann die Maschine ein Problem einlernen.

Schematisch:



# Abschätzung des Testfehlers

Nach Vapnik gilt mit Wahrscheinlichkeit  $1 - \eta$

$$E(h_\alpha) \leq \approx E_{emp}(h_\alpha) + \sqrt{\dots \frac{VC(h_\alpha)}{N} \dots}$$

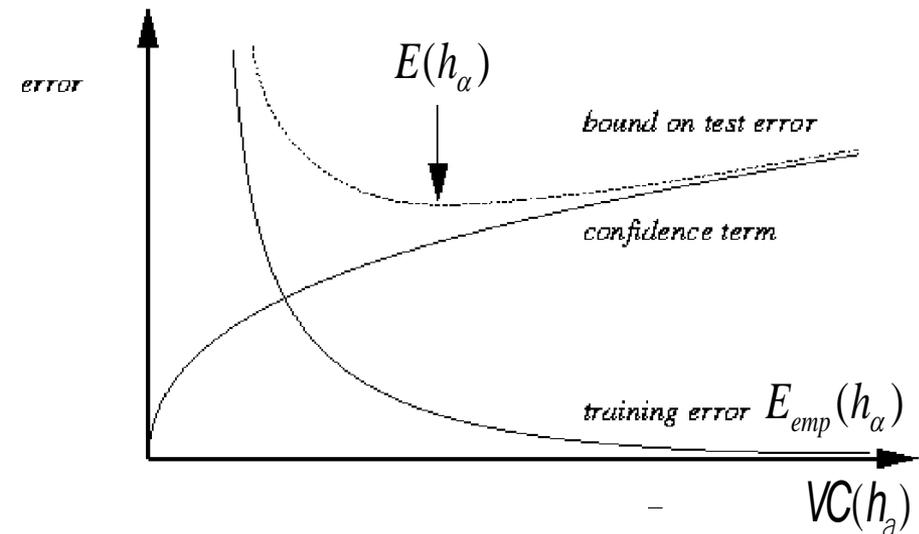
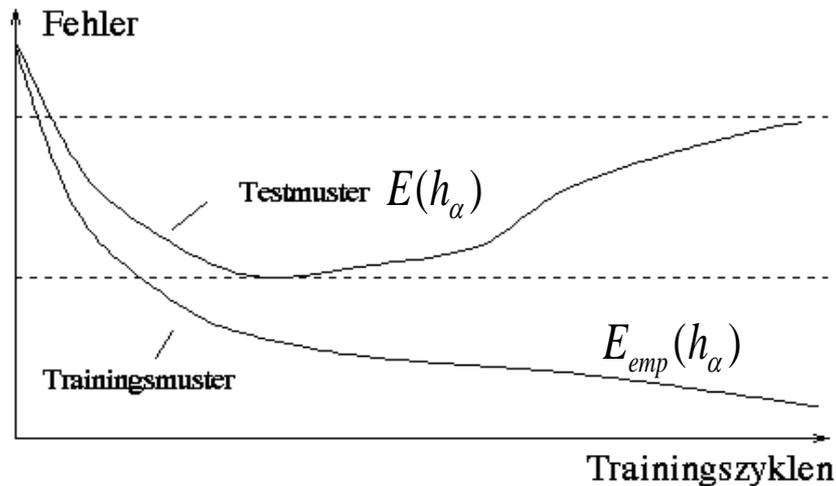
wobei:

- $VC(h_\alpha)$  – VC-Dimension der lernenden Maschine
- $N$  – Anzahl der Lernbeispiele
- $E_{emp}(h_\alpha)$  – empirischer Fehler abhängig von  $VC(h_\alpha)$  und  $N$
- $E(h_\alpha)$  – realer (zu minimierender) Fehler

Lernerfolg ist abhängig von:

- Kapazität der lernenden Maschine (so gering wie nötig)
- Optimierungsmethode (so gut wie möglich)
- Lernbeispiele (repräsentativ, so viele wie möglich)

# Abschätzung des Testfehlers



- Minimierung des emp. Fehlers bei:
- VC Dimension (z.B. Topologie bei NN)  
= konst.
  - Anzahl von Trainingsbeispielen  
= konst,
  - Iterative Optimierung

- Verhältnis: emp. und realer Fehler
- VC Dimension = veränderlich
  - Anzahl von Trainingsbeispielen  
= konst.

# Lösungsansatz: Structural Risk Minimization

Ziel: finde eine Lösung für

$$\min_{H_n} \left( E_{\text{emp}}(h_\alpha) + \sqrt{\dots \frac{VC(h_\alpha)}{N} \dots} \right)$$

↔ finde  $VC(h_\alpha)$  („Maschine“),  $N$  („Beispiele“),  
und  $\alpha$  („Minimum des emp. Fehlers“)

Ideale Lösung:

- Minimiere Summe (nicht Summanden)
- Strukturiere den Hypothesenraum

$$H^1 \subset H^2 \subset \dots \subset H^n, \quad VC(h_\alpha^i) \leq VC(h_\alpha^{i+1})$$

- Suche Optimum: das Minimum für  $E(h_\alpha)$

# Lösungsansatz: Structural Risk Minimization

## Probleme:

- Strukturierung:
  - Berechnung der VC Dimension schwer, rechenintensiv, für viele Hypothesenräume unmöglich
- Optimierung:
  - Finden, definieren der Hypothesenräume
  - große Kapazität → kleiner empirischer Fehler
  - geringe Kapazität → größerer Fehler
  - Minimierung von  $E_{emp}(h_\alpha^n), \quad \forall H^n$

Korrektes Lernen → Gesucht sind geeignete Realisierungen

- Definition des Lernens
- Fehler: empirischer & realer Fehler
- Overfitting
- Modellauswahl (Crossvalidation, Bootstrap, AdaBoost)
- Was ist PAC ?
- Approximation des realen Fehlers nach Vapnik und Chervonenkis
- Wie kann korrektes Lernen erfolgen?

# ... und wie geht es weiter in ML

Bestimme eine Hypothese  $h$  aus Beispielen  $\leftrightarrow$  Finde geeignete Maschine und bestimme optimales Modell  $M$

- Probabilistisch: Schätzen von Wahrscheinlichkeiten
  - Bayes
- im Wesentlichen Empirische Minimierung aber auch mit Struktureller Risikominimierung
  - Entscheidungsbäume
  - Neuronale Netze
- Strukturelle Risikominimierung
  - Support Vector Methoden

Tom M. Mitchell: Machine Learning

Tom M. Mitchell - Carnegie Mellon University (CMU)  
<http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/mlc/>

Duda & Hart: Pattern Classification

V.N. Vapnik: Statistical Learning Theory

P. Viola, M. Jones: Rapid object detection using a boosted cascade of simple features (2001 , ....)