Kapitel ML:IX (Fortsetzung)

IX. Combined Models and Meta Learning

- Motivating Ensemble Classification
- Bagging
- Boosting
- Cascading
- □ Ensemble Classifier

ML:IX-1 Meta Learning © LETTMANN 2007-2015

Motivating Ensemble Classification

Generalisierungsfähigkeit von Klassifikatoren

 \Box *Err**(y) (wahre Missklassifikationsrate)

$$\Box$$
 $\textit{Err}(y, D_{tr}) = \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D_{tr} : c(\mathbf{x}) \neq y(\mathbf{x})\}|}{|D_{tr}|}$ (Missklassifikationsrate auf Trainingsmenge)

$$\Box$$
 $\textit{Err}(y, D_{ts}) = \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D_{ts} : c(\mathbf{x}) \neq y(\mathbf{x})\}|}{|D_{ts}|}$ (Missklassifikationsrate auf Testmenge)

 $Err(y, D_{ts})$ ermöglicht auch den Vergleich von Lernverfahren.

→ Gibt es ein bestes Lernverfahren?

ML:IX-2 Meta Learning © LETTMANN 2007-2015

Motivating Ensemble Classification

No Free Lunch Theorems

"There Ain't No Such Thing As A Free Lunch."

[Heinlein, 1966]

In der Optimierung:

"[...] all algorithms that search for an extremum of a cost function perform exactly the same, when averaged over all possible cost functions."

[Wolpert/Macready, 1995]

Im Maschinellen Lernen:

"Some of those theorems show, loosely speaking, that for any two algorithms A and B, there are "as many" targets for which algorithm A has lower expected OTS [off-training set sampling] error than algorithm B as vice-versa (whether one averages over training sets or not)."

[Wolpert 1996]

ML:IX-3 Meta Learning © LETTMANN 2007-2015

Motivating Ensemble Classification

Instabilität von Lernverfahren

Heuristische Formulierung:

Ein Lernverfahren heißt instabil, wenn eine kleine Veränderung in den Trainingsdaten eine große Veränderung im gelernten Klassifikator bewirkt.

Instabile Lernverfahren:

- Neuronale Netze (mit fixer Anzahl von hidden Units)
- Entscheidungsbäume (mit fixer Anzahl von Blattknoten)

Stabile Lernverfahren:

- Bayes
- □ k Nearest Neighbor

ML:IX-4 Meta Learning ©LETTMANN 2007-2015

Motivation Ensemble-Klassifikation

Zusammenfassung

- Statistisches Problem.
 - Verfahren betrachten eine gemessen an der Menge von Trainingsdaten zu große Menge von Hypothesen.
 - → Auf Basis der Trainingsdaten eignen sich mehrere Hypothesen gleichermaßen gut als Klassifizierer.
- Rechentechnisches Problem.
 - Aufgrund der Komplexität des Problems kann das Lernverfahren nicht das Finden einer besten Lösung innerhalb der Hypothesenmenge garantieren.
 - → Bei Verwendung von Heuristiken of nur suboptimale Lösung.
- □ Repräsentationsproblem:
 - Die Kandidatenmenge der Hypothesen enthält keine ausreichend guten Approximationen des Zielkonzeptes.
 - → Lernverfahren kann den gewünschten Approximationsgrad nicht liefern.

ML:IX-5 Meta Learning © LETTMANN 2007-2015

Bootstrap Aggregating

Idee:

Eine Gruppe von Klassifikatoren, die gemeinsam klassifizieren, kann die Nachteile einzelner Klassifikatoren aufwiegen.

Problem:

Das Lernverfahren braucht verschiedene Trainingsmengen, um verschiedene Klassifikatoren zu bestimmen.

Lösung:

Generierung von ähnlichen Trainingsmengen durch Bootstrapping (vgl. auch Kreuzvalidierung).

ML:IX-6 Meta Learning ©LETTMANN 2007-2015

Remarks:

Lernverfahren, die sich anderer Lernverfahren bedienen, um einen Klassifizierer zu bilden
heißen auch Meta-Lernverfahren.

Aufgrund der untergeordneten Rolle der einzelnen Klassifizierer beim Bagging, besonders aber aufgrund ihrer recht schwachen Klassifikationsvermögens im Kontext des Boosting oder des Cascading, nennen wir diese auch schwache Klassifizierer.

ML:IX-7 Meta Learning © LETTMANN 2007-2015

Bootstrap Aggregating [Breiman 1994]

- $lue{}$ Ausgangspunkt Lernmenge D mit n Beispielen
- \Box Für $t=1,\ldots,T$: Ziehe aus D insgesamt n Beispiele mit Zurücklegen und bilde daraus die Lernmenge D_t
- \square Mit der Lernmenge D_t, \ldots, D_T werden mit Hilfe eines Lernverfahrens die einzelne Klassifikatoren y_t bestimmt.
- Die Klassifikatoren y_1, \ldots, y_T werden zu einem Ensemble zusammengefasst und legen durch Mehrheitsentscheid die Klasse eines Beispiels fest:

$$y(x) := \underset{j \in \{1,...,J\}}{\operatorname{argmax}} |\{t \in \{1,...,T\} : y_t(x) = j\}|$$

ML:IX-8 Meta Learning ©LETTMANN 2007-2015

Algorithm: Bagging

Input: Lernbeispiele $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \dots, (\mathbf{x}_n, c(\mathbf{x}_n))\}, n \in \mathbf{N}$

mit $\mathbf{x}_i \in X$ und $c(\mathbf{x}_i) \in \{1, \dots, J\}$ für $1 \le i \le n$;

Anzahl T mit $T \in \mathbb{N}$ für die Anzahl der Klassifizierer.

Output: Klassifizierer y für X.

1. Für t = 1, ..., T:

- (a) Bestimme Lernmenge D_t durch Bootstrapping aus D (Ziehen von n Beispielen aus D mit Zurücklegen)
- (b) Trainiere einen schwachen Klassifikator y_t , d.h. $y_t: X \to \mathbf{R}$ mit der Beispielmenge D_t .
- 2. Ergebnis ist der Klassifikator

$$y(x) := \underset{j \in \{1, \dots, J\}}{\operatorname{argmax}} \left| \{ t \in \{1, \dots, T\} : y_t(x) = j \} \right|$$

ML:IX-9 Meta Learning ©LETTMANN 2007-2015

Leistungsfähigkeit von Bootstrap Aggregating

- Die Wahrscheinlichkeit, dass ein Beispiel mindestens einmal gezogen wird, ist $1 (1 1/n)^n$.
- □ Für n groß, gilt $1 (1 1/n)^n \approx 1 1/e \approx 0.632$.
- In jeder Lernmenge sind etwa 63.2% der Beispiele in D.
- □ Verbesserungen der Fehlerrate von 20% bis 47% bei Anwendung mit Entscheidungsbäumen wurden beobachtet.

ML:IX-10 Meta Learning © LETTMANN 2007-2015

Boosting Weak Classifiers [Freund/Schapire 1995]

Idee:

Eine Gruppe von Klassifikatoren, die gemeinsam klassifizieren, kann die Nachteile einzelner Klassifikatoren aufwiegen.

Problem:

Das Lernverfahren braucht verschiedene Trainingsmengen, um verschiedene Klassifikatoren zu bestimmen.

Lösung:

Verschränkung von Lernalgorithmus und Generierung von Lernmengen: Gewichtung der Lernbeispiele (Änderung der relativen Häufigkeiten) aufgrund der Auswertung des vorherigen Klassifikators.

ML:IX-11 Meta Learning ©LETTMANN 2007-2015

AdaBoost, Adaptive Boosting [Freund/Schapire 1995]

- $lue{}$ Ausgangspunkt Lernmenge D mit n Beispielen
- 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.
- \Box Die Klassifikatoren y_1, \ldots, y_T werden zu einem Ensemble zusammengefasst und legen durch gewichteten Mehrheitsentscheid die Klasse eines Beispiels fest.

→ Anwendung z.B. mit teilweise gelernten Entscheidungsbäumen

ML:IX-12 Meta Learning © LETTMANN 2007-2015

Algorithm: AdaBoost.M1

Input: Lernbeispiele $(\mathbf{x}_1, c(\mathbf{x}_1)), \dots, (\mathbf{x}_n, c(\mathbf{x}_n)), n \in \mathbf{N}$

mit $\mathbf{x}_i \in X$ und $c(\mathbf{x}_i) \in \{1, \dots, J\}$ für $1 \le i \le n$;

Anzahl T mit $T \in \mathbb{N}$ für die Anzahl der Runden.

Output: Klassifizierer y für X.

- 1. Initialisiere Gewichte für alle Beispiele durch $w_1(i) = 1/n$ für $1 \le i \le n$.
- 2. Für t = 1, ..., T führe folgende Schritte aus:
 - (a) Trainiere einen schwachen Klassifikator y_t , d.h. $y_t: X \to \mathbf{R}$, mit den durch w_t gewichteten Beispielen.
 - (b) Sei $\varepsilon_t = \sum_{i=1}^n w_t(i) \cdot (1 \delta(y_t(\mathbf{x}_i), c(\mathbf{x}_i))) = \sum_{\{i | y_t(\mathbf{x}_i) \neq c(\mathbf{x}_i)\}} w_t(i)$. (δ Kronecker-Funktion, d.h. $\delta(u, v) = 1$ für u = v und $\delta(u, v) = 0$ sonst.)
 - (c) Setze $\beta_t = \frac{\varepsilon_t}{(1-\varepsilon_t)}$.
 - (d) Setze $w_{t+1}(i) = \begin{cases} w_t(i) \cdot \beta_t \cdot 1/z_t & \text{falls } y_t(\mathbf{x}_i) = c(\mathbf{x}_i) \\ w_t(i) \cdot 1/z_t & \text{sonst} \end{cases}$ für $1 \leq i \leq n$. z_t ist Normalisierungsfaktor, durch den das Gesamtgewicht aller Beispiele den Wert 1 erhält, also eine Verteilung widerspiegelt.
- 3. Ergebnis ist der Klassifikator

$$y(\mathbf{x}) = \operatorname*{argmax}_{j \in \{1, \dots, J\}} \sum_{\{t \mid y_t(\mathbf{x}) = j\}} \log \frac{1}{\beta_t}$$

ML:IX-13 Meta Learning © LETTMANN 2007-2015

Leistungsfähigkeit von AdaBoost.M1

Der Klassifikator y gewichtet die Entscheidungen der einzelnen Klassifikatoren stärker, wenn ihr Fehler klein ist.

Wenn die einzelnen Klassifikatoren eine bessere Fehlerrate als 1/2 haben, dann fällt der Fehler von y exponentiell in T gegen 0.

Satz 1

Falls für die Fehlerraten ε_t während des Ablaufs von Algorithmus AdaBoost.M1 gilt $\varepsilon_t \leq 1/2$, so folgt für den trainierten Klassifizierer y

$$\frac{1}{n} \cdot |\{i : y(\mathbf{x}_i) \neq c(\mathbf{x}_i)\}| \leq \exp\left(-2\sum_{t=1}^T \left(\frac{1}{2} - \varepsilon_t\right)^2\right)$$

ML:IX-14 Meta Learning ©LETTMANN 2007-2015

Remarks:

- \Box Problem: Fehlerrate der im Fall von J Klassen mit J > 2 nicht so einfach erreichbar.
- \Box Spezialfall j=2: Klassifikationsproblem mit genau 2 Klassen, ein schwacher Klassifizierer muss nur geringfügig besser sein als Raten.

 \Box Betrachtung der Klassen $\{-1, +1\}$ erlaubt einfachere Schreibweisen.

ML:IX-15 Meta Learning ©LETTMANN 2007-2015

Algorithm: Discrete AdaBoost

Input: Lernbeispiele $(\mathbf{x}_1, c(\mathbf{x}_1)), \dots, (\mathbf{x}_n, c(\mathbf{x}_n)), n \in \mathbf{N}$

 $\mathsf{mit}\; \mathbf{x}_i \in X \; \mathsf{und}\; c(\mathbf{x}_i) \in \{-1, +1\} \; \mathsf{für}\; 1 \leq i \leq n;$

Anzahl T mit $T \in \mathbb{N}$ für die Anzahl der Runden.

Output: Klassifizierer y für X.

- 1. Initialisiere Gewichte für alle Beispiele durch $w_1(i) = 1/n$ für $1 \le i \le n$.
- 2. Für t = 1, ..., T führe folgende Schritte aus:
 - (a) Trainiere einen schwachen Klassifikator c_t , d.h. $y_t: X \to \{-1, +1\}$, mit den durch w_t gewichteten Beispielen.
 - (b) Sei $\varepsilon_t = \sum_{i=1}^n w_t(i) \cdot 1/2 \cdot |y_t(\mathbf{x}_i) c(\mathbf{x}_i)| = \sum_{\{i | y_t(\mathbf{x}_i) \neq c(\mathbf{x}_i)\}} w_t(i)$.
 - (c) Setze $\beta_t = \frac{\varepsilon_t}{(1-\varepsilon_t)}$.
 - (d) Setze $w_{t+1}(i) = \left\{ \begin{array}{ll} w_t(i) \cdot \beta_t \cdot 1/z_t & \text{falls } y_t(\mathbf{x}_i) = c(\mathbf{x}_i) \\ w_t(i) \cdot 1/z_t & \text{sonst} \end{array} \right.$ für $1 \leq i \leq n$.

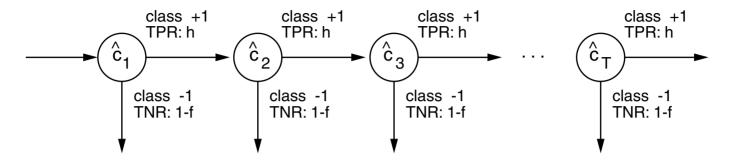
 z_t ist Normalisierungsfaktor, durch den das Gesamtgewicht aller Beispiele den Wert 1 erhält, also eine Verteilung widerspiegelt.

3. Ergebnis ist der Klassifikator

$$y(\mathbf{x}) = \operatorname{sign}\left(\sum_{t=1}^T \log \frac{1}{\beta_t} \cdot y_t(\mathbf{x})\right)$$

ML:IX-16 Meta Learning © LETTMANN 2007-2015

Cascades of Classifiers [Viola/Jones 2001]



- Gesucht wird eine Folge y_1, \ldots, y_T von Klassifikatoren mit steigender Komplexität für ein 2-Klassen-Problem.
- \Box In der Lernmenge D überwiegen die Negativbeispiele (Klasse -1).
- Aus der Lernmenge D wird eine Teilmenge D_t von Beispielen gezogen, die von der bisherigen Kaskade mit +1 klassifiziert werden. Das Verhältnis von Positivbeispielen und Negativbeispielen in D_t ist fest.
- \Box Mit D_t wird durch Boosting ein Ensemble y_t von schwachen Klassifizierern gelernt.
- Für 10 Stufen in der Kaskade, einer Hitrate von mindestens 0.99 und einer False Alarm Rate von höchstens 0.3 erhält man für die Kaskade eine Hitrate von $0.99^{10} \approx 0.9$ und eine False Alarm Rate von höchstens $0.3^{10} \approx 0.000006$.

ML:IX-17 Meta Learning © LETTMANN 2007-2015

Cascades of Classifiers (Fortsetzung)

In einem 2-Klassen-Problem wird eine Lern- oder Testmenge D durch einen Klassifikator c in vier disjunkte Teilmengen aufgeteilt:

- 1. D_h Menge der Beispiele \mathbf{x} mit Klasse +1 und $y(\mathbf{x}) = +1$ (Index h steht für *hits*, Menge der richtig positiven Beispiele, true positive),
- 2. D_f Menge der Beispiele \mathbf{x} mit Klasse -1 und $y(\mathbf{x}) = +1$ (Index f steht für *false alarms*, Menge der falsch positiven Beispiele, false positive, Fehler 1. Art),
- 3. D_m Menge der Beispiele \mathbf{x} mit Klasse +1 und $y(\mathbf{x}) = -1$ (Index m steht für *misses*, Menge der falsch negativen Beispiele, false negative, Fehler 2. Art),
- 4. D_z Menge der Beispiele \mathbf{x} mit Klasse -1 und $y(\mathbf{x}) = -1$ (Index z steht für zero, Menge der richtig negativen Beispiele, true negative).

(*richtig/falsch* bezieht sich auf Übereinstimmung von $c(\mathbf{x})$ und $y(\mathbf{x})$, *positiv/negativ* auf die Klassifikation durch y, also $y(\mathbf{x})$.) Diese Mengen werden häufig in einer Tabelle angeordnet:

	c = +1	c = -1	
y = +1	D_h	D_f	D_{pPos}
y = -1	D_m	D_z	D_{pNeg}
	D_{Pos}	D_{Neg}	D

 D_{pNeg} und D_{Neg} bezeichnen die Mengen der (predicted) Negatives, D_{pPos} und D_{Pos} bezeichnen die Mengen der (predicted) Positives.

ML:IX-18 Meta Learning © LETTMANN 2007-2015

Maßzahlen

Mit den Kardinalitäten dieser Mengen lassen sich verschiedene Maßzahlen definieren. Maßzahlen zur Beurteilung des Tests:

- extstyle ext
- \square $TNR = P(y(\mathbf{x}) = -1 | c(\mathbf{x}) = -1) \approx \frac{|D_f|}{|D_f| + |D_z|} = \frac{|D_f|}{|D_{Neg}|}$ Wahrscheinlichkeit einer richtig negativen Vorhersage (Richtignegativ-Rate, Correct Rejection Rate, True Negative Rate, Spezifität)

ML:IX-19 Meta Learning © LETTMANN 2007-2015

Maßzahlen (Fortsetzung)

Maßzahlen zur Beurteilung der Population:

- \square $NPV = P(c(\mathbf{x}) = -1|y(\mathbf{x}) = -1) \approx \frac{|D_z|}{|D_m| + |D_z|} = \frac{|D_z|}{|D_{pNeg}|}$ Wahrscheinlichkeit der Korrektheit bei negativer Vorhersage (negativ prädiktiver Wert, Negative Predictive Value, Segreganz, Trennfähigkeit)
- □ $FAR = P(c(\mathbf{x}) = -1|y(\mathbf{x}) = +1) \approx \frac{|D_f|}{|D_h| + |D_f|} = \frac{|D_f|}{|D_{pPos}|}$ Wahrscheinlichkeit der Nicht-Korrektheit bei positiver Vorhersage (False Alarm Ratio)
- $\Box \quad FCR \approx \frac{|D_f| + |D_m|}{|D_h| + |D_f| + |D_m|} = \frac{|D_f| + |D_m|}{|D|}$ Wahrscheinlichkeit einer falschen Klassifikation (Falschklassifikationsrate, False Classification Rate)

ML:IX-20 Meta Learning ©LETTMANN 2007-2015

Algorithm: Cascading

Input: Lernbeispiele $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \dots, (\mathbf{x}_N, c(\mathbf{x}_n))\}, n \in \mathbf{N}$

mit $\mathbf{x}_i \in X$ und $c(\mathbf{x}_i) \in \{+1, -1\}$ für $1 \le i \le n$; Anzahl T mit $T \in \mathbf{N}$ für die Anzahl der Stufen.

Output: Klassifizierer y für X.

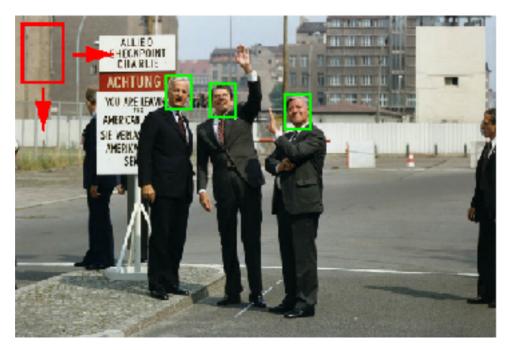
- 1. Für t = 1, ..., T führe folgende Schritte aus:
 - (a) Bestimme Lernmenge D_t durch Ziehen von n' Beispielen aus D, die unter der bisherigen Kaskade als +1 klassifiziert wurden.
 - (b) Trainiere einen Klassifikator y_t , d.h. $y_t: X \to \mathbf{R}$ mit der Beispielmenge D_t .
- 2. Ergebnis ist der Klassifikator

$$y(x) := \left\{ \begin{array}{ll} +1 & \text{falls } y_t(x) = +1 \text{ für alle } t \in \{1, \dots, T\} \\ -1 & \text{sonst} \end{array} \right.$$

ML:IX-21 Meta Learning ©LETTMANN 2007-2015

Anwendung: Face Detection (Main Loop)

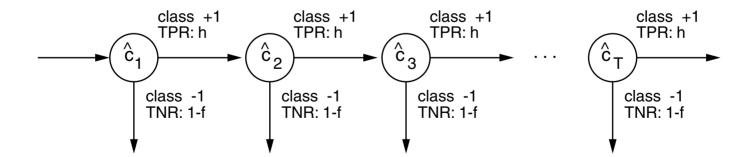
- Graustufenbilder
- \Box Suchbereich mit fester Größe 20×20 Pixel
- \Box verschiedene Skalierungsstufen des Ausgangsbildes: Faktor $1/1.1 \approx 0.91$
- □ zeilenweises Überstreichen mit Suchbereich
- Überprüfen des Ausschnitts auf Gesicht



ML:IX-22 Meta Learning © LETTMANN 2007-2015

Anwendung: Face Detection (Gesamtklassifikator)

- \Box Kaskade mit T=22
- \Box False Alarm Rate pro Stufe: FPR = 0.5
- \Box True Positive Rate pro Stufe: TPR = 0.999
- □ False Alarm Rate gesamt: $\approx 0.5^{22} \approx 0.00000024$
- ☐ True Positive Rate Rate gesamt: $\approx 0.999^{22} \approx 0.978$
- $\ \square$ Trainingsbeispiele für Stage i+1 sind positiv klassifizierte Beispiele der Stage i



ML:IX-23 Meta Learning ©LETTMANN 2007-2015

Anwendung: Face Detection (Einzelklassifikator in Kaskade)

- Ensemble-Klassifizierer mit Discrete AdaBoost bestimmen
- □ Stage 1: drei schwache Klassifikatoren ... Stage 22: 213 schwache Klassifikatoren

Algorithm: Discrete AdaBoost

Input: Lernbeispiele $(\mathbf{x}_1, c(\mathbf{x}_1)), \dots, (\mathbf{x}_n, c(\mathbf{x}_n)), n \in \mathbf{N}$

 $\text{mit } \mathbf{x}_i \in X \text{ und } c(\mathbf{x}_i) \in \{-1, +1\} \text{ für } 1 \leq i \leq n;$

Anzahl T mit $T \in \mathbb{N}$ für die Anzahl der Runden.

Output: Klassifizierer y für X.

- 1. Initialisiere Gewichte für alle Beispiele durch $w_1(i) = 1/n$ für $1 \le i \le n$.
- 2. Für t = 1, ..., T führe folgende Schritte aus:
 - (a) Trainiere einen schwachen Klassifikator c_t , d.h. $y_t: X \to \{-1, +1\}$, mit den durch w_t gewichteten Beispielen.
 - (b) Sei $\varepsilon_t = \sum_{i=1}^n w_t(i) \cdot 1/2 \cdot |y_t(\mathbf{x}_i) c(\mathbf{x}_i)| = \sum_{\{i|y_t(\mathbf{x}_i) \neq c(\mathbf{x}_i)\}} w_t(i)$.
 - (c) Setze $\beta_t = \frac{\varepsilon_t}{(1-\varepsilon_t)}$.
 - $\text{(d)} \quad \text{Setze } w_{t+1}(i) = \left\{ \begin{array}{ll} w_t(i) \cdot \beta_t \cdot 1/z_t & \text{falls } y_t(\mathbf{x}_i) = y_i \\ w_t(i) \cdot 1/z_t & \text{sonst} \end{array} \right. \quad \text{für } 1 \leq i \leq n.$

 z_t ist Normalisierungsfaktor, durch den das Gesamtgewicht aller Beispiele den Wert 1 erhält, also eine Verteilung widerspiegelt.

3. Ergebnis ist der Klassifikator

$$y(\mathbf{x}) = \operatorname{sign}\left(\sum_{t=1}^T \log rac{1}{eta_t} \cdot y_t(\mathbf{x})
ight)$$

ML:IX-24 Meta Learning © LETTMANN 2007-2015

Anwendung: Face Detection (Einzelklassifikator in Ensemble)

- Schwacher Klassifikator: CART Klassifikator mit genau einem Knoten
- Gesucht: Threshold mit bestem Missklassifikationsgewicht für Feature
- Auswahl: Merkmal mit bestem Missklassifikationsgewicht
- Größe der Trainingsmenge: ca. 5000 Bespiele für Gesichter, ca. 3000 Beispiele ohne Gesichter pro Stufe



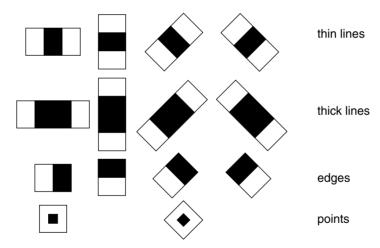
Welche Merkmale verwendet man für die Gesichtserkennung?

ML:IX-25 Meta Learning © LETTMANN 2007-2015

Anwendung: Face Detection (alle Features)

- \Box Verwendung der 20×20 Bildpunkte, also 400 Features mit je 256 Werten
- → Schwache Klassifikatoren verwenden einzelne Bildpunkte.

Aussagefähigere Merkmale:



- Merkmale sind Helligkeitsunterschiede zwischen Flächen.
- \Box Merkmale: Typ \times Position \times Breite \times Höhe
- \Box Merkmale: Anzahl verschiedener Merkmale > 100.000

ML:IX-26 Meta Learning © LETTMANN 2007-2015

Anwendung: Face Detection (ausgewählte Features)

Haben die ausgewählten Merkmale eine Bedeutung für uns?



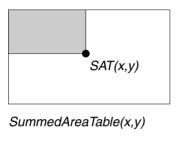
- Zusammenfassung der Merkmale der Stufen 1 bis 11
- Auswertung der Merkmale:
 - Ausgangspunkt graue Fläche (Wert 127)
 - dunkler Bereich des Merkmals verringert Farbwerte
 - heller Bereich des Merkmals erhöht Farbwerte
 - Betrag abhängig vom Threshold

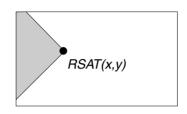
Nicht alle Merkmale sind anzeigbar.

ML:IX-27 Meta Learning © LETTMANN 2007-2015

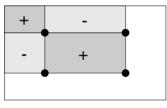
Anwendung: Face Detection (Effizienz)

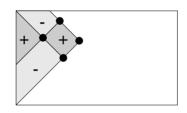
Wie lassen sich die Merkmale effizient berechnen?





RotatedSummedAreaTable(x,y)





Area pixel sum

Rotated area pixel sum

- Pass 1: $\mathsf{RSAT}(x,y) = \mathsf{RSAT}(x-1,y-1) + \mathsf{RSAT}(x-1,y) + I(x,y) \mathsf{RSAT}(x-2,y-1)$ $\mathsf{mit} \; \mathsf{RSAT}(-1,y) = \mathsf{RSAT}(-2,y) = \mathsf{RSAT}(x,-1) = 0$
- $\begin{tabular}{ll} \square & Pass 2: \\ & RSAT(x,y) = RSAT(x,y) + RSAT(x-1,y+1) RSAT(x-2,y) \\ \end{tabular}$

ML:IX-28 Meta Learning ©LETTMANN 2007-2015

Anwendung: Face Detection (Effizienz beim Training)



- Anzahl Trainingsbeispiele: > 10.000
- □ Anzahl Merkmale: > 100.000
- \Box Benötigter Hauptspeicher: 4 Byte \times 10.000 \times 100.000 > 3,7GByte

Merkmalwerte werden (zeilenweise) immer wieder neu berechnet. Mehrere Tage Trainingslaufzeit.

ML:IX-29 Meta Learning ©LETTMANN 2007-2015

Ensemble Classifier

Trainingsmengen

- Bagging
 - D_t wird aus D_t gebildet durch Ziehen mit Zurücklegen.
 - → Trainingsmengen voneinander unabhängig.
- Boosting
 - D_t wird aus D_{t-1} gebildet durch stärkere Gewichtung falsch klassifizierter Beispiele.
 - → Trainingsmengen bauen aufeinander auf, sind aber bis auf Gewichtung gleich.
- Cascading
 - D_t wird aus D_{t-1} gebildet durch Beschränkung auf korrekt klassifizierte Beispiele.
 - → Trainingsmengen bauen aufeinander auf, sind jeweils Teilmengen der vorherigen.

ML:IX-30 Meta Learning ©LETTMANN 2007-2015

Ensemble Classifier

Grundlage der Entscheidung

- Bagging
 Einfache Mehrheitsentscheidung durch schwache Klassifizierer.
- Boosting
 Gewichtete Mehrheitsentscheidung durch schwache Klassifizierer.
- Cascading
 Sequentielle Entscheidung durch schwache Klassifizierer: +1 nur bei Einstimmigkeit.

ML:IX-31 Meta Learning © LETTMANN 2007-2015

Ensemble Classifier

Literature

- □ P. Viola, M. Jones. *Rapid Object Detection using a Boosted Cascade of Simple Features*
- R. Lienhart, J. Maydt.
 An Extended Set of Haar-like Features for Rapid Object Detection
- □ R. Lienhart, A. Kuranov, V. Pisarevsky. *Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection*
- R. Lienhart, L. Liang, A. Kuranov.
 A Detector Tree of Boosted Classifiers for Real-time Object Detection and Tracking

ML:IX-32 Meta Learning © LETTMANN 2007-2015