Ludwig Maximilians Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme

Skript zur Vorlesung
Knowledge Discovery in Databases II
im Wintersemester 2012/13

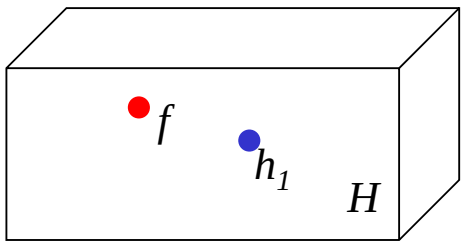
Kapitel 7: Ensemble Learning und Multi-Repräsentierte Daten

http://www.dbs.ifi.lmu.de/Lehre/KDD_II

Übersicht

- Annahme: Elemente *x* aus einem Raum *D* gehören zu einer Klasse c_i aus einer Menge von möglichen Klassen *C*.
- Es gibt eine Funktion $f: D \rightarrow C$, die einen eindeutigen Zusammenhang zwischen einem gegebenen Element *x* und seiner Klasse c_i beschreibt.
- Aufgabe eines Lern-Algorithmus' ist es, diesen Zusammenhang zu "lernen".
- Im Allgemeinen stellt ein Klassifikator (das Ergebnis eines Lern-Algorithmus') eine Approximation der Funktion *f* dar, auch eine "Hypothese" genannt.

- Die "wahre" Funktion *f* ist unbekannt.
- Es gibt nur eine Menge von Beispielen: Tupel $(x, c_i) \in f \subseteq D \times C$, die Trainingsdaten.
- Ein konkreter Lernalgorithmus sucht diejenige Hypothese h_i als Klassifikator aus einem Raum $H \subseteq D \times C$ möglicher Hypothesen, die optimal zu den Trainingsdaten passt.



- Achtung: die Zielfunktion *f* ist nicht zwangsläufig Element von *H*!

- Ein Klassifikator (eine erlernte Hypothese *h*) kann auf Elemente $x \in D$ angewendet werden, um die Klasse $c_i = f(x)$ vorherzusagen.
- Die Genauigkeit eines Klassifikators ist die Wahrscheinlichkeit (oder statistisch gemessen: die Häufigkeit), mit der seine Vorhersage korrekt ist.

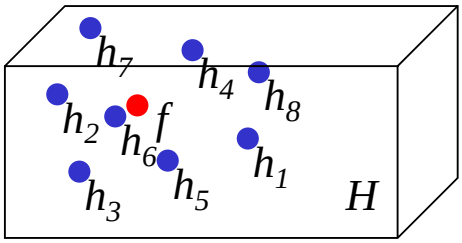
$$Acc(h) = P(h(x)=f(x))$$

- Entsprechend ist die Fehlerrate das Komplement:

$$Err(h) = P(h(x)\neq f(x))=1-Acc(h)$$

- Idee der Ensemble-Technik: Reduktion der Häufigkeit von Fehlurteilen durch Bilden einer “Jury von Experten” und Abstimmung über die richtige Vorhersage.

- mathematisch: bessere Approximation von *f* durch Mittelung über mehrere Hypothesen



- Einfacher Abstimmungsmodus für ein Zwei-Klassen-Problem mit *C={-1,1}*:
  - Bilde Menge von Hypothesen $\{h_1,\ldots,h_k\}$ mit Gewichten $\{w_1,\ldots,w_k\}$.
  - Ensemble-Klassifikator $\hat{h}$ ist gegeben durch

$$\hat{h}(x) = \begin{cases} w_1 h_1(x) + \ldots + w_k h_k \geq 0 \rightarrow 1 \\ w_1 h_1(x) + \ldots + w_k h_k < 0 \rightarrow -1 \end{cases}$$

- Häufig $w_1=\ldots=w_k=1$ (bzw. ungewichtete Abstimmung).
- Gewichte können aber auch auf der (gemessenen) Zuverlässigkeit der einzelnen Klassifikatoren (Hypothesen) basieren.
- Komplexeres Abstimmungsverhalten möglich (und bei mehr als zwei Klassen auch nötig) → verschiedene Ensemble-Methoden

$$\hat{h}(x) = \begin{cases} w_1 h_1(x) + \ldots + w_k h_k \geq 0 \rightarrow 1 \\ w_1 h_1(x) + \ldots + w_k h_k < 0 \rightarrow -1 \end{cases}$$

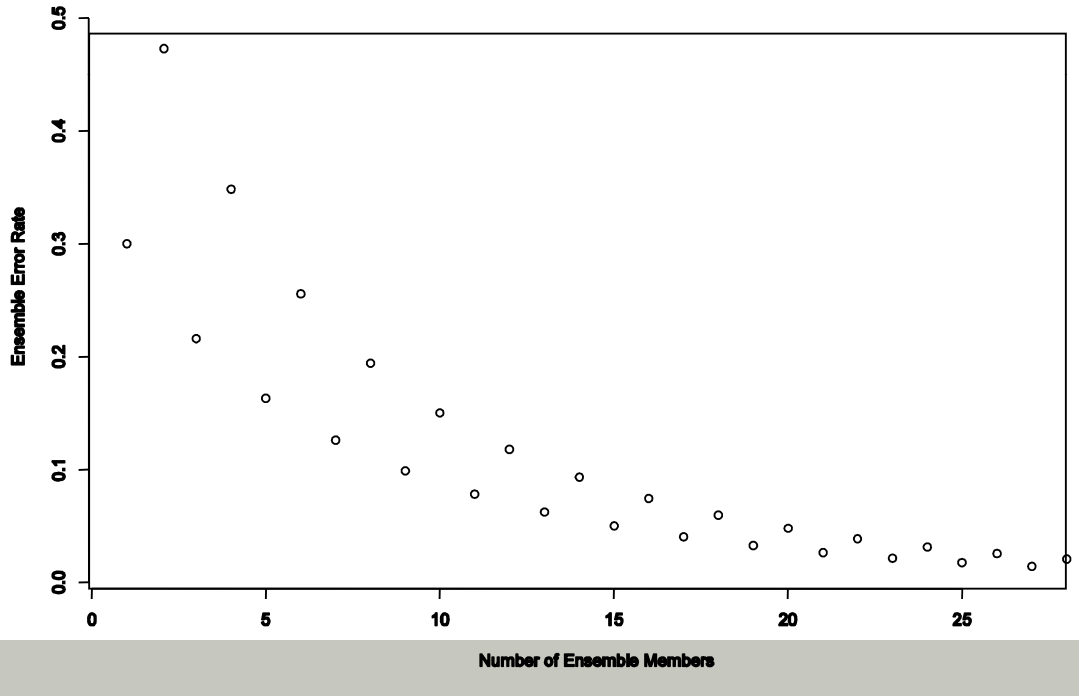
- Error-Rate eines Ensembles abhängig von der Error-Rate der Base-Classifier und ihrer Anzahl:

  die Häufigkeit, mit der mindestens die Hälfte der Ensemble-Mitglieder falsch abstimmt:

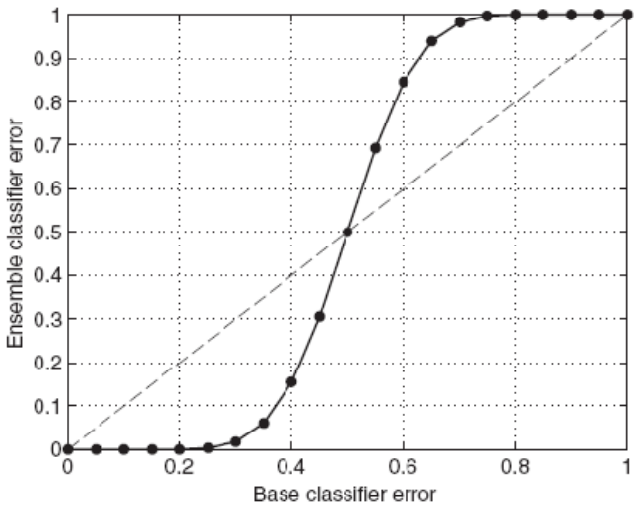
$$Err(\hat{h}) = \sum_{i=\left\lceil \frac{k}{2} \right\rceil}^{k} \binom{k}{i} e^i (1-e)^{k-i}$$

- (Annahme: $Err(h_1)=\ldots=Err(h_k)=e$)

- Abhängigkeit der Gesamt-Error-Rate von der Anzahl der Base-Classifier (bei Fehlerrate der Base-Classifier von 0,3):



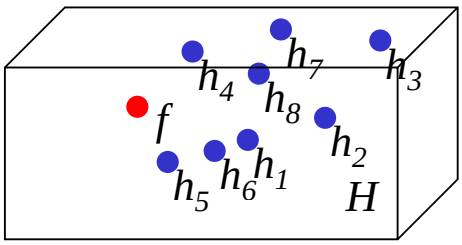
- Error-Rate für ein einfaches Abstimmungs-Ensemble mit 25 Basis-Klassifikatoren:



(aus: Tan, Steinbach, Kumar: Introduction to Data Mining)

- Notwendige Annahme für diese Verbesserung: Unabhängigkeit der Fehler der einzelnen Base-Classifier

$$Err(\hat{h}) = \sum_{i=\left\lceil \frac{k}{2} \right\rceil}^{k} \binom{k}{i} e^i (1-e)^{k-i}$$



- einseitige Fehler: keine oder nur wenig Verbesserung durch Ensemble

- Schlussfolgerung:

> Notwendige Bedingungen für Verbesserung der Gesamt-Fehlerrate:
>
> 1. Alle Base-Classifier sind "genau" (accurate).
> 2. Die einzelnen Base-Classifier sind "unterschiedlich" (diverse).

- Genauigkeit: milde Bedingung (besser als Zufall)
- Diversität: keine (oder wenigstens keine starke) Korrelation der Vorhersagen
- Ist gleichzeitige Optimierung von Genauigkeit und Diversität möglich?

Aspekte der Diversität

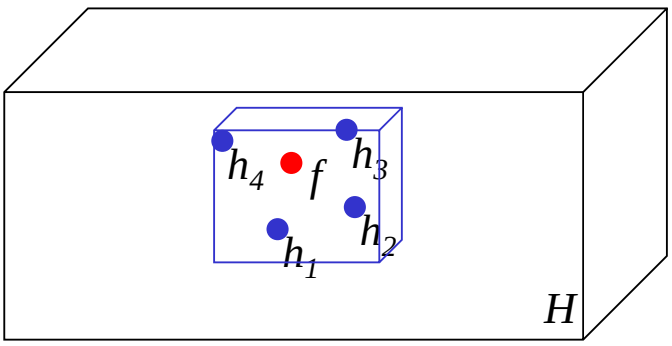
- Gründe für die Diversität von Classifiern für das selbe Klassifikationsproblem:
  - Statistische Varianz
  - Berechnungs-Varianz
  - Darstellungsproblem

DATABASE SYSTEMS GROUP

Aspekte der Diversität

LMU

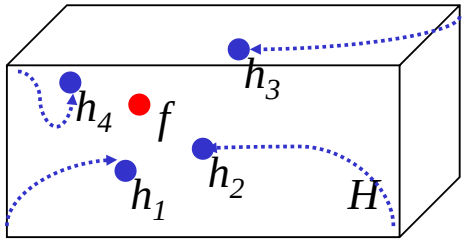
- Statistische Varianz:
  - Der Raum möglicher Hypothesen ist zu groß, um anhand der begrenzten Trainingsdaten eine beste Hypothese zu bestimmen.



  - Kombination mehrerer Hypothesen reduziert das Risiko, sehr stark daneben zu liegen.

Aspekte der Diversität

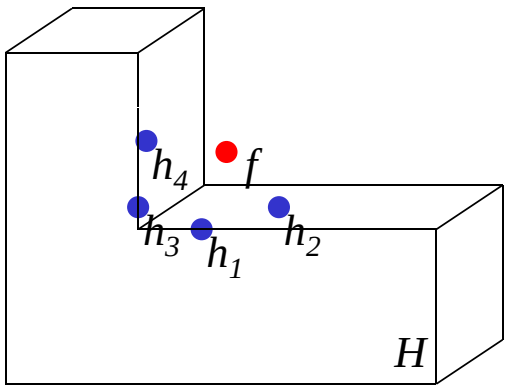
- Berechnungs-Varianz:
  - Manche Lern-Algorithmen können nicht garantieren, die beste Hypothese aus dem Raum möglicher Hypothesen zu finden, da dies zu Berechnungsaufwändig wäre.
  - Z.B. werden beim Lernen Heuristiken verwendet, die in lokalen Optima gefangen bleiben können.



  - Kombination mehrerer Hypothesen reduziert das Risiko, das falsche (lokale) Optimum gewählt zu haben.

Aspekte der Diversität

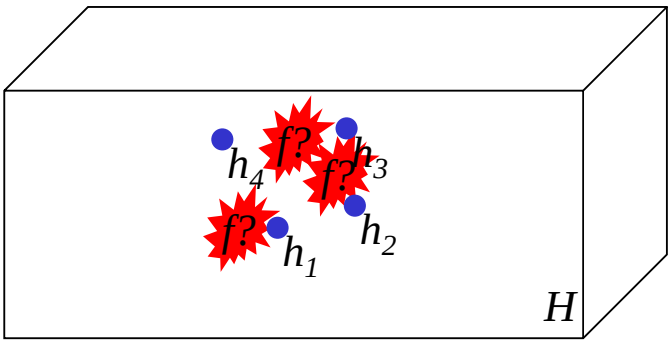
- Darstellungsproblem:
  - Der Hypothesenraum enthält gar keine guten Approximationen an die "wahre" Funktion f.



  - Kombination mehrerer Hypothesen kann den Raum darstellbarer Hypothesen erweitern.

Aspekte der Diversität

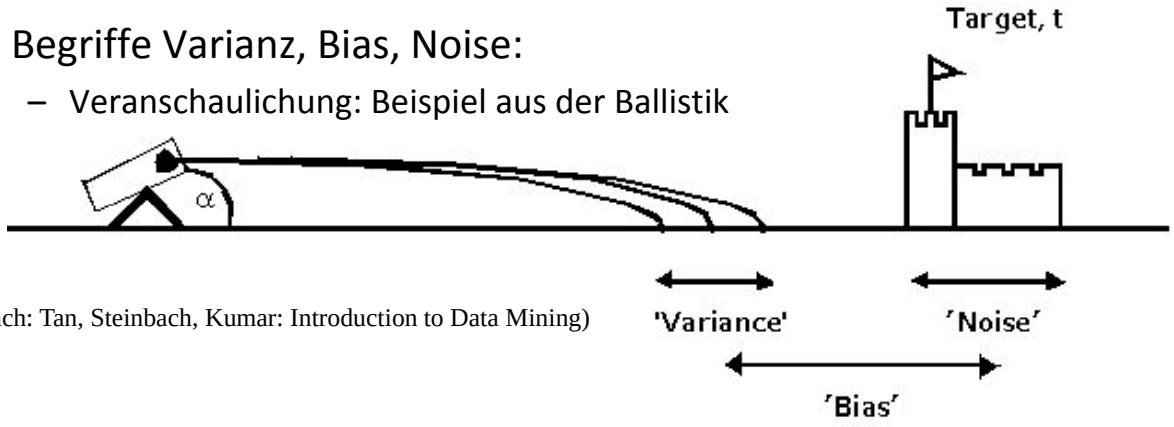
- Unscharfe Zielfunktion:
  - Die Lernbeispiele (Trainingsdaten) erlauben keine eindeutigen Rückschlüsse auf die Zielfunktion (z.B. wegen widersprüchlicher Beispiele oder nicht-deterministischer Klassenzugehörigkeit).



  - Kombination mehrerer Hypothesen reduziert das Risiko, eine fehlerhafte Zielfunktion zu approximieren.

Aspekte der Diversität

- Begriffe Varianz, Bias, Noise:
  - Veranschaulichung: Beispiel aus der Ballistik



(nach: Tan, Steinbach, Kumar: Introduction to Data Mining)

  - Varianz, Bias und Noise sind verschiedene Komponenten des Fehlers

$$err = Bias_{\alpha} + Variance_{f} + Noise_{t}$$

  - Varianz: abhängig von der aufgewendeten Kraft f
  - Noise: Unschärfe des Ziels
  - Bias: abhängig vom Abschusswinkel

Aspekte der Diversität

- Begriffe Varianz, Bias, Noise in der Klassifikation:
  - Varianz:

    Abhängig von Variationen in den Trainingsdaten oder der Parametrisierung des Klassifikators werden unterschiedliche Hypothesen gebildet.
  - Noise:

    Klassenzugehörigkeit ist nicht deterministisch oder anderweitig uneindeutig (z.B. widersprüchliche Trainingsbeispiele).
  - Bias:

    Ein bestimmter Lernalgorithmus hat immer auch bestimmte Annahmen über das zu erlernende Konzept (z.B. Annahme der Möglichkeit linearer Trennbarkeit verschiedener Klassen).

    Ein Lernen ohne jede konzeptionelle Annahme wäre nur ein Auswendiglernen ➔ "Bias-free learning is futile."

Aspekte der Diversität

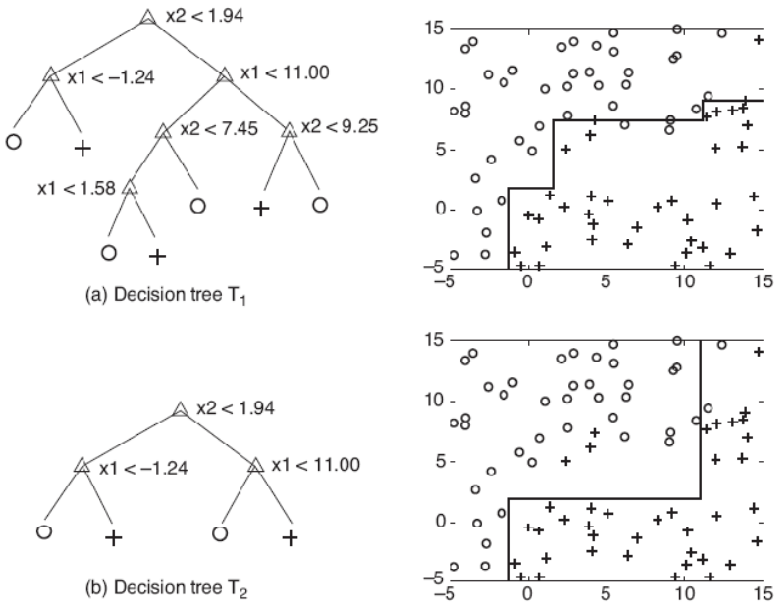


Figure 5.33. Two decision trees with different complexities induced from the same training data.

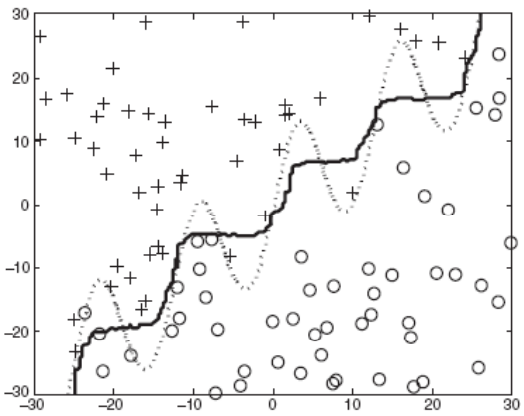
(aus: Tan, Steinbach, Kumar: Introduction to Data Mining)

Aspekte der Diversität

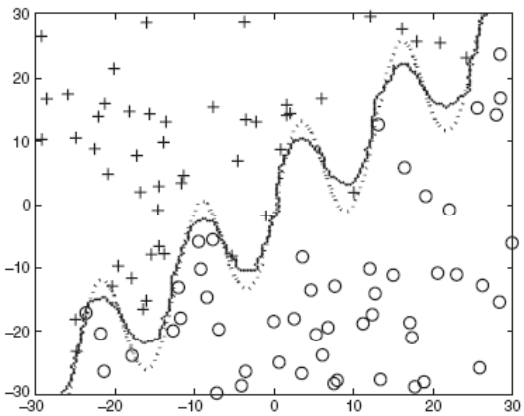
- Bias am Beispiel von Decision Trees:
  - T_1 und T_2 wurden auf den gleichen Daten trainiert
  - T_2 wurde durch Pruning auf maximale Tiefe 2 aus T_1 erzeugt
  - T_2 hat stärkere Annahmen bezüglich der Trennbarkeit der Klassen, also stärkeren Bias

Aspekte der Diversität

- relativer Beitrag von Bias und Variance zum Error ist unterschiedlich für verschiedene Klassifikationsmethoden



(a) Decision boundary for decision tree.



(b) Decision boundary for 1-nearest neighbor.

Figure 5.34. Bias of decision tree and 1-nearest neighbor classifiers.

(aus: Tan, Steinbach, Kumar: Introduction to Data Mining)

Aspekte der Diversität

- Beispiel:
  - Durchschnittliche Entscheidungsgrenzen über 100 Klassifikatoren, trainiert auf 100 unterschiedlichen Trainingsdatensätzen mit jeweils 100 Beispielen.
  - gestrichelt: wahre Entscheidungsgrenze, die zur Erzeugung der Daten benutzt wurde
  - Beobachtung:
    - geringerer Abstand der gemittelten Entscheidungsgrenze von der wahren Entscheidungsgrenze bei 1-NN Klassifikatoren

      ➔ niedrigerer Bias
    - größere Variabilität der einzelnen Entscheidungsgrenzen innerhalb der 100 1-NN Klassifikatoren

      ➔ höhere Varianz

Übersicht

Methoden der Konstruktion von Ensembles

- Wie kann man Unterschiedlichkeit von Klassifikatoren erreichen?
  - Variieren des Training Sets
    - Methoden: Bagging und Boosting
  - Manipulieren der Input-Features
    - Lernen auf unterschiedlichen Unterräumen
    - Verwendung verschiedener Repräsentationen (MR-learning: nächstes Kapitel)
  - Manipulieren der Klassenlabel
    - Verschiedene Arten von Abbildungen auf Zwei-Klassen-Probleme
  - Manipulieren des Lernalgorithmus'
    - Einführen von Zufallselementen
    - Unterschiedliche Startkonfigurationen

Variieren der Trainings-Menge

- Eine wichtige Eigenschaft von Lernalgorithmen ist die Stabilität.
- Ein Lernalgorithmus ist umso stabiler, je weniger sich die auf unterschiedlichen Trainingsdaten (für das gleiche Klassifikationsproble) erzeugten Klassifikatoren unterscheiden.
- Bei einem instabilen Lernalgorithmus haben kleine Änderungen in der Trainingsmenge starke Änderungen der gelernten Hypothese zur Folge.
- Um Ensembles basierend auf Variationen der Trainingsmenge zu bilden, sind **instabile** Lernalgorithmen vorteilhaft, z.B.:
  - Decision Trees
  - Neuronale Netze
  - Regel-Lerner

Variieren der Trainings-Menge

- Bootstrap:

  bilden einer Trainingsmenge aus einer gegebenen Datenmenge durch Ziehen mit Zurücklegen.
  - jedes Sample hat die gleiche Größe wie die ursprüngliche Trainingsmenge
  - ein Sample enthält durchschnittlich 63% der Ausgangsbeispiele (einige mehrfach, etwa 37% gar nicht):
    - ein einzelnes Beispiel in einem Datensatz mit *n* Beispielen hat bei jedem Ziehen die Chance *1/n* gezogen zu werden, wird also mit Wahrscheinlichkeit *1-1/n* **nicht** gezogen
    - nach *n*-mal Ziehen ist ein bestimmtes Element mit Wahrscheinlichkeit $\left(1-\frac{1}{n}\right)^n$ nicht gezogen worden
    - für große *n* ist $\left(1-\frac{1}{n}\right)^n \approx e^{-1} \approx 0.368$
  - daher auch der Name “0.632 bootstrap” für diese Sampling-Methode (als solche auch eine Alternative zur Kreuzvalidierung)

Variieren der Trainings-Menge

- Bagging (**B**ootstrap **Agg**regat**ing**):

  bilden unterschiedlicher Trainingsmengen durch wiederholtes bootstrapping
- Bagging aggregiert mehrere Bootstraps (Samples nach obigem Muster) und trainiert auf jedem Bootstrap einen eigenen Classifier.
- Bei instabilen Lernalgorithmen werden hinreichend unterschiedliche Hypothesen erlernt.
- Ein neuer Datensatz wird durch einfache Abstimmung über alle erlernten Hypothesen klassifiziert.

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

Variieren der Trainings-Menge

- Während der 0.632 Bootstrap unter Gleichverteilung gezogen wird, weist **Boosting** jedem Datensatz ein Gewicht zu.
- Datenobjekte, die schwierig zu klassifizieren sind, erhalten ein höheres Gewicht.
- Verwendung der Gewichte:
  - Angabe der Ziehungswahrscheinlichkeit im bootstrap sample der nächsten Runde

    ➔ schwierige Beispiele sind in der nächsten Runde häufiger in der Trainingsmenge und erhalten daher automatisch ein höheres Gewicht beim Training des Klassifikators

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

  - Manche Lernalgorithmen können Gewichte von Datensätzen direkt benutzen

    ➔ Bias der erlernten Hypothese auf die höher gewichteten Beispiele hin

Manipulation der Input-Feature

- Manipulieren der Input-Features:
  - Lernen auf unterschiedlichen Unterräumen oder kombinierten Features
    - Beispiel: Random Forests

      Menge von Decision Trees, deren Training durch Zufallsvektoren bestimmt wird, z.B.:

      a) zufällige Auswahl von Features für den Split an jedem Knoten des Baumes

      b) an jedem Knoten Erzeugen eines neuen Features als Linearkombination einer zufällig ausgewählten Teilmenge der Features

      c) an jedem Knoten zufällige Auswahl aus den F besten Splits
  - Kombination von Klassifiern, die auf unterschiedlichen Repräsentationen der Daten trainiert wurden: siehe nächstes Kapitel

Manipulieren der Klassenlabel

- Zahlreiche Methoden bilden ein Multi-Klassen-Problem auf mehrere Zwei-Klassen-Probleme ab.

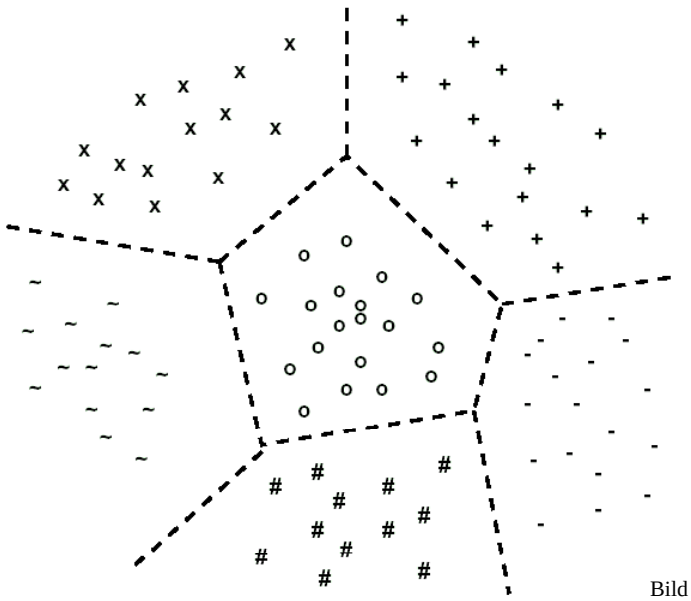


Bild aus: Fürnkranz 2002

DATABASE SYSTEMS GROUP

Manipulieren der Klassenlabel

LMU

- Die Entscheidungen der auf den einzelnen Zwei-Klassen-Problemen trainierten Klassifikatoren werden geeignet kombiniert, um auf die ursprüngliche Klasse zurückzuschließen.
- Dies entspricht dem Einführen von Unterschiedlichkeit in Klassifikatoren durch Manipulieren der Klassenlabel.
- Gängige Methoden:
  - one-versus-rest
  - all-pairs
  - error correcting output codes

Manipulieren der Klassenlabel

- *one-versus-rest* (auch: *one-versus-all*, *one-per-class*):

  Bei *n* Klassen, werden *n* Klassifikatoren trainiert, die jeweils eine Klasse von allen anderen unterscheiden sollen.

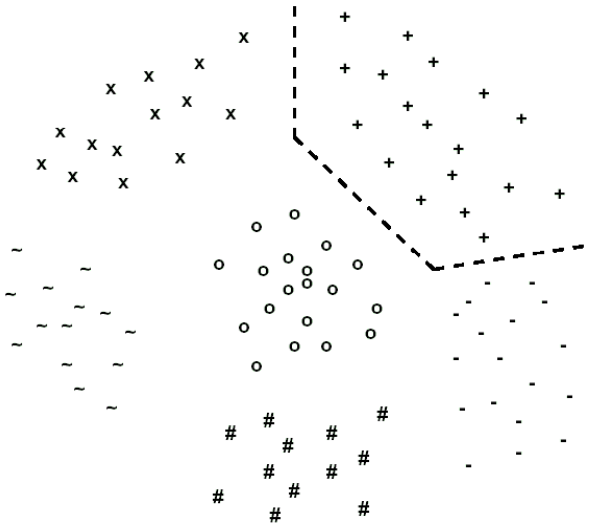


Bild aus: Fürnkranz 2002

Manipulieren der Klassenlabel

- *all-pairs* (auch: *all-versus-all*, *one-versus-one*, *round robin*, *pairwise*):

  Für jedes Paar von Klassen wird ein Klassifikator trainiert, der diese beiden Klassen unterscheiden soll.

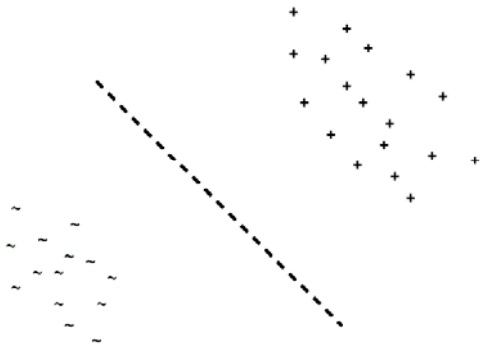


Bild aus: Fürnkranz 2002

Manipulieren der Klassenlabel

- Error Correcting Output Codes (ECOC):
  - Die Menge *C* der Klassen wird *k*-mal zufällig in zwei Teilmengen *A+B* aufgeteilt.
  - Datensätze, die zu Klasse *A* gehören, erhalten das neue Label *-1*, die anderen (Klasse *B*) das neue Label *1*.
  - Auf den entstehenden *k* Zwei-Klassen-Problemen werden *k* Klassifikatoren trainiert.
  - Stimmt Klassifikator *i* für Klasse *A*, erhalten alle Klassen aus *C*, die zu *A* gehören, eine Stimme.
  - Die Klasse $c \in C$, die die meisten Stimmen erhalten hat, ist die Klassifikationsentscheidung des Ensembles.

Manipulieren der Klassenlabel

- Beispiel: $C=\{c_1,c_2,c_3,c_4\}$, 7-bit Kodierung

Klasse	Code-Wort						
c_1	1	1	1	1	1	1	1
c_2	0	0	0	0	1	1	1
c_3	0	0	1	1	0	0	1
c_4	0	1	0	1	0	1	0

- Für jedes Bit der Code-Wörter wird ein Klassifikator trainiert, hier also 7 Klassifikatoren.
- Angenommen, ein Klassifikationsergebnis ist (0,1,1,1,1,1,1) – für welche Klasse entscheidet das Ensemble?

Manipulieren der Klassenlabel

- Der Name “Error Correcting Output Codes” steht für die Idee, dass beim Lernen eine gewisse Redundanz der Klassengrenzen eingeführt wird.
- Die “Code-Wörter”, die die Zugehörigkeit zu den Klassen binär codieren, können zufällig gewählt werden.
- Für eine gute Diversität sollten die Code-Wörter aber gut separieren:
  - Row-Separation: Jedes Paar von Code-Wörtern sollte eine große Hamming-Distanz (=Anzahl der unterschiedlichen Bits) aufweisen.
  - Column-Separation: Die einzelnen Binär-Klassifikatoren sollten unkorreliert sein.

Manipulieren der Klassenlabel

Klasse	Code-Wort						
c_1	1	1	1	1	1	1	1
c_2	0	0	0	0	1	1	1
c_3	0	0	1	1	0	0	1
c_4	0	1	0	1	0	1	0

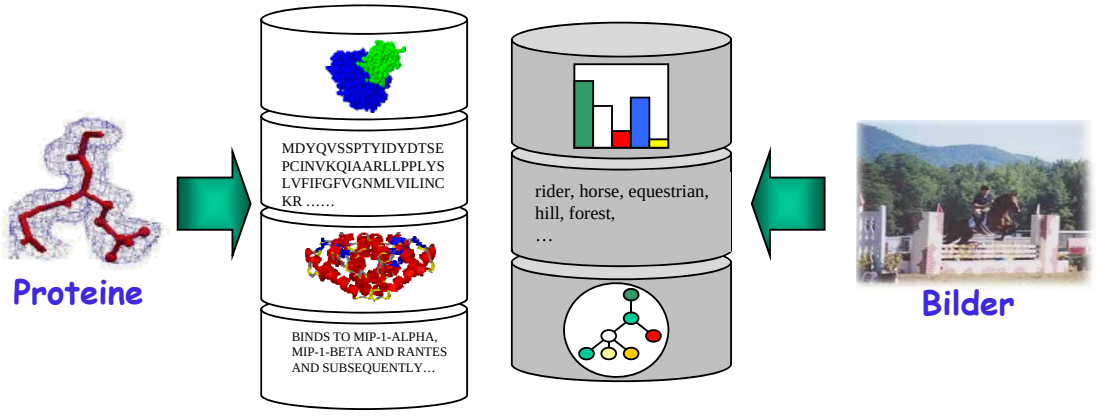
- Große Hamming-Distanz zwischen den Zeilen erlaubt möglichst eindeutige Klassifikationsentscheidung des Ensembles.
- Welche Hamming-Distanz weist das Klassifikationsergebnis (0,1,1,1,1,1,1) zu den Codes für c_1, c_2, c_3 und c_4 jeweils auf?

Manipulieren des Lernalgorithmus

- Manipulieren des Lernalgorithmus durch Zufallselemente:
  - Starten von unterschiedlichen Konfigurationen aus (z.B. Start-Gewichte für Backpropagation)
  - Randomisierte Entscheidungen in Decision Trees beim Split-Kriterum (vgl. Random Forests)

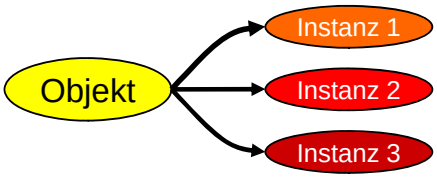
Übersicht

Grundsituation



Gründe für Multirepräsentierte Objekte:

- unterschiedliche Featuretransformationen
- unterschiedliche Messtechniken
- unterschiedliche Aspekte desselben Objekts



➡ Multirepräsentierte Objekte

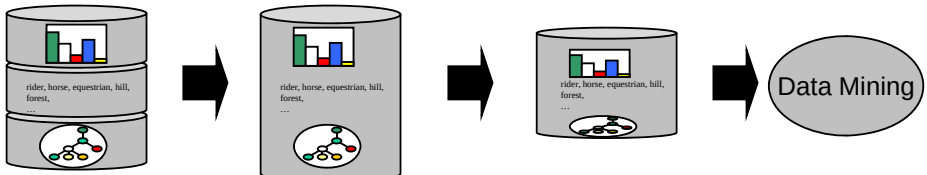
Probleme mehreren Repräsentationen

Grundproblem:

- alle notwendigen Informationen sollen dem Algorithmus zur Verfügung stehen => Verwende alle verfügbaren Informationen
- zu viele unnötige Features können das Ergebnis negativ beeinflussen => Verwende nur notwendige Features

Standard Lösungsansatz:

1. Bilde einen gemeinsamen Feature-Space aus allen Features jeder Repräsentation.
2. Benutze Feature-Reduktion oder Feature-Selektion.
3. Wende Data Mining auf reduzierten Feature-Raum an.



Multirepräsentierte Algorithmen

Möglichkeit zur Kombination mehrerer Repräsentationen:

1. **Kombination auf Feature-Ebene:**
   - unterschiedliche Merkmale werden aus verschiedenen Repräsentationen in einen Feature-Vektor vereint.
   - Feature-Selektion oder Selektion der Repräsentation sollen irrelevante Information ausschließen. Bereits behandelt in Kap.2

2. **Kombination der Distanzen und Ähnlichkeiten**:
   Bestimme Objektähnlichkeit in jeder Repräsentation
   und kombiniere Ähnlichkeitsaussagen.

3. **Kombination auf Muster-Ebene:**

- Bestimme Muster in jeder Repräsentation und kombiniere die Muster zu allgemeinen Mustern.
  Bsp: Kombination der Klassenwahrscheinlichkeiten aus mehreren Repräsentationen.

Multirepräsentierte Ähnlichkeits- und Distanzfunktionen

Integration der verschiedenen Repräsentationen über Kombination von Ähnlichkeitsmaßen oder Distanzen.

Idee: Erhalte die Trennung der einzelnen Repräsentationen bei und kombiniere auf Ebene der Ähnlichkeitsaussagen.

Beispiel: gewichtete Linear-Kombination

$d_i(o_1,o_2)$: lokale Metrik oder lokaler Kernel in R_i.

$$D_{kombi}(o_1,o_2) = \sum_{R_i \in R} w_i \cdot d_i(o_1,o_2)$$

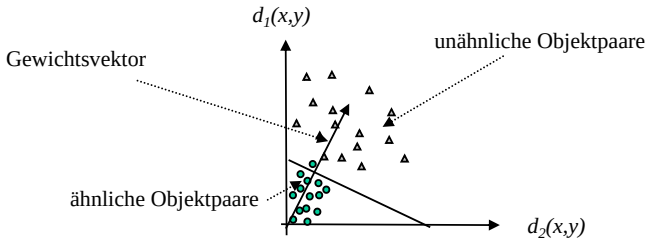
Lernen von Gewichten

Formuliere Ähnlichkeit als lineares Klassifikationsproblem:

- Normalvektor der trennenden Hyperebene setzt sich aus Gewichten zusammen
- Trainingsobjekte: Paare von ähnlichen und unähnlichen Objekten
- Klassen: {„ähnlich“,“unähnlich“}
- Feature-Space: Abstandsvektor, $v_i = d_i(x,y)$ für alle Represenation R_i $1 \leq i \leq n$

Vorgehen:

- Bestimme Abstandsvektoren auf DB-Sample
  (Vorsicht: Es gibt quadratisch viele Abstandsvektoren! => Sample)
- Trainiere linearen Klassifikator
- Bestimme Gewichtungsvektor aus Normalvektor der Trennebene (MMH).



Kombination von Distanzen/Ähnlichkeiten

Bemerkungen:

- Vorsicht: lineare Klassifikatoren garantieren keine positiven Gewichte für alle Repräsentationen!

- Alternativ kann auch der gelernte Klassifikator direkt zur Kombination der Ähnlichkeiten bzw. Distanzen verwendet werden. In diesem Fall ersetzt die Wahrscheinlichkeit für die Klasse „unähnlich“ die Distanz.

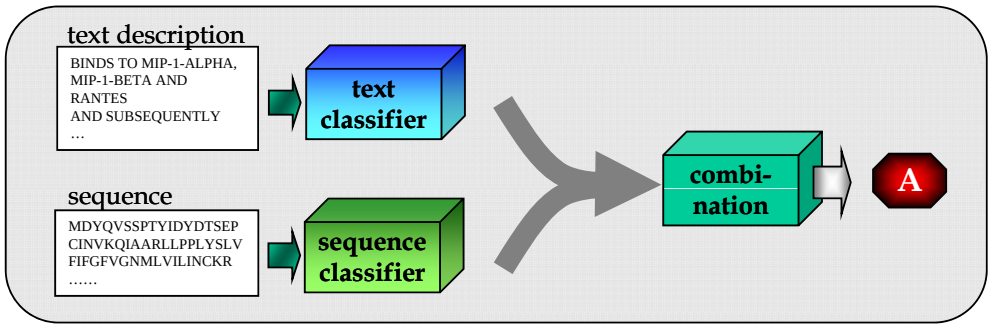
- Bei komplexeren Kombinationsregeln müssen die Metrik- bzw. Kernel-Eigenschaften erneut geprüft werden, falls das anschließende Data-Mining-Verfahren diese Eigenschaften benötigt.

Multirepräsentierte Klassifikation

Eingabe : $o \in R_1 \times .. \times R_n$, wobei R_i der Darstellungsraum für die i-te Repräsentation ist.

Kombination mehrerer Klassifikatoren (Classifier Combination):

1. Trainiere Klassifikator für jede Repräsentation getrennt.
2. Klassifiziere neues Objekte mit jedem Klassifikator
3. Kombiniere die Resultate der Klassifikatoren zur einer globalen Klassenvorhersage.



Kombination mehrerer Klassifikatoren

Wie kombiniert man Klassenvorhersagen so, dass die richtige Vorhersage bevorzugt wird?

1. Jeder Klassifikator gibt für jede Klasse *A* und ein Objekt *x* eine Vorhersagewahrscheinlichkeit c_A zurück.
   Für Konfidenzvektor $c^r(x)$ gilt: $\sum_{A \in C} c_A^r(x) = 1$

2. Klassifikation durch Kombintion der Konfidenzevektoren $c^r(x)$:

$$pred(X) = \underset{A \in C}{\mathbf{argmax}} \left(\Theta_{r \in R} \left(c_A^r \right) \right) \text{ mit } \Theta \in \left\{ \min, \max, \sum, \prod \right\}$$

Kombination mehrerer Klassifikatoren

Beispiel:

Gegeben: 2 Repräsentation für Bildobjekte: Farbhistogramme(R1) und Texturvektoren(R2).

Klassen = {„enthält Wasseroberfläche“=A, „keine Wasseroberfläche“=B}

Bayes Klassifikatoren K1 (für R1) und K2 (für R2)

Kombination mit Summe.

Klassifikation von Bild b:

K1(b)=c1 =(0.45, 0.55); K2(b) = c2 = (0.6, 0.4)

Kombination mit Durchschnitt (Summe):

c_{global} = (1.05, 0.95) * ½ = (0.525 , 0.475) und argmax(c_{global}) = A

Kombination mit Produkt:

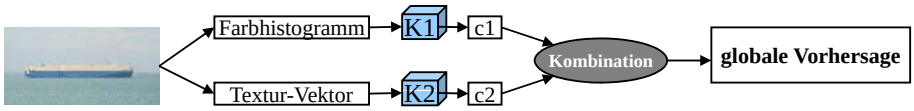
c_{global} = (0.27, 0.22) und argmax(c_{global}) = A

Kombination mit Maximum:

c_{global} = (0.6, 0.55) und argmax(c_{global}) = A

Kombination mit Mininum:

c_{global} = (0.45, 0.4) und argmax(c_{global}) = A



Co-Training

Multiple Repräsentationen können auch dazu verwendet werden eine Trainingsmenge zu erweitern.

Gegeben: 2 Repräsentationen für die sowohl gelabelte als auch nicht gelabelte Objekte vorhanden sind.

Idee: Benutze Klassifikator um neue Trainingsobjekte aus ungelabelten Datenobjekten zu erzeugen.

Aber: Warum braucht man dazu mehrere Repräsentationen ?

Generieren von Trainingsobjekte mit nur 1 Repräsentation

Versuch:

- Trainiere Klassifikator ***CL*** auf allen gelabelten Objekten
- klassifiziere *k* ungelabete Objekte und füge sie in die Trainingsmenge ein.
- Trainiere nächsten Klassifikator auf der neuen Trainingsmenge

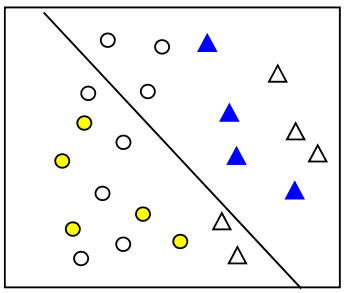
Problem:

- neue Daten werden mit dem Modell von ***CL*** gelabelt
- damit neue Trainingsobjekte CL verändern können, müssten sie aber Widersprüche zum bisherigen Modell enthalten
- => Generieren von Trainingsobjekten mit einer Repräsentation verstärkt nur die Schwächen des Klassifikators

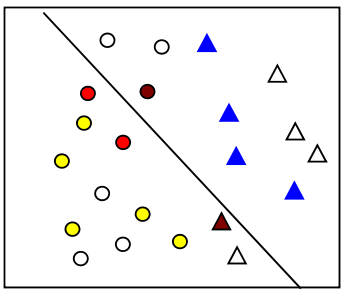
Generieren von Trainingsobjekte mit nur 1 Repräsentation

Beispiel:

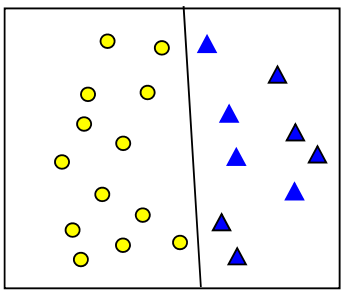
- blau = gelabelte Objekte Dreieck-Klasse
- gelb = gelabelte Objekte Kreis Klasse
- rot = relabelte Objekte mit CL_1



Training auf originalen Daten



Training mit relabelten Daten



optimale Lösung

Fazit:

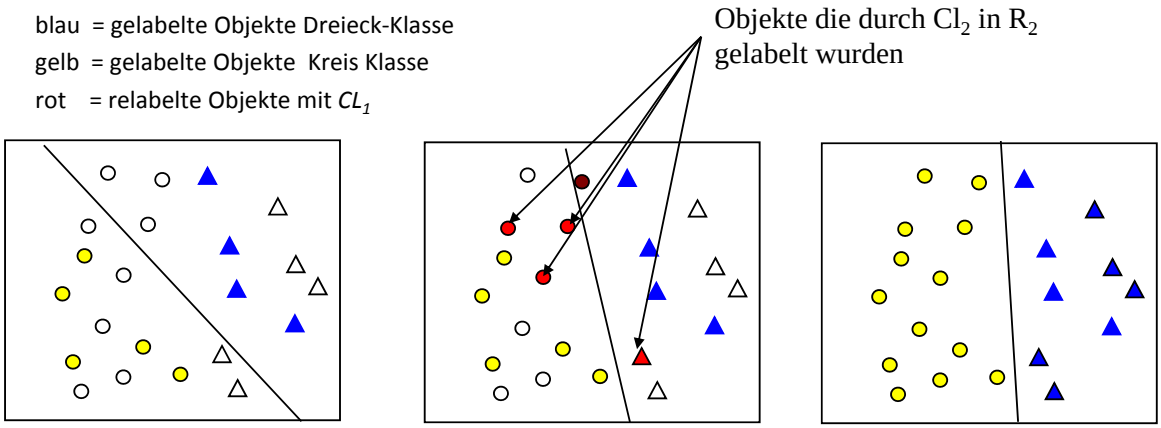
- Die roten Objekte bestätigen nur die Annahmen des Klassifikators, können diese aber nicht verbessern.
- Zur Verbesserung wären von CL unabhängig Informationen notwendig.

Co-Training

Idee: Klassifikatoren aus anderen Repräsentationen labeln Objekte, mit für diese Repräsentation neuen Informationen.

Beispiel:

- blau = gelabelte Objekte Dreieck-Klasse
- gelb = gelabelte Objekte Kreis Klasse
- rot = relabelte Objekte mit CL_1



- originaler Klassifikator
- Klassifikator nach unabhängigen Relabeling
- optimale Lösung

=> Durch neue unabhängig gelabelte Objekte kann sich ein Klassifikator verbessern.

Der Co-Training Algorithmus

Gegeben: 2 Mengen aus multirepräsentierten Objekten

TR = gelabelte Trainingsmenge, *U* = Menge ungelabelter Objekte.

Co-Training Algorithmus

For *k* times do

For each R_i Do

Trainiere CL_i für Repräsentation *i.*

Ziehe Sample aus *U*.

generiere neue Label mit CL_i.

füge neu gelabelte Objekte zu *TR* hinzu

Clustering Multirepräsentierter Objekte

Anforderungen an Clustering-Algorithmen für Multirepräsentierte Objekte:

- Integration aller Informationsquellen.
- Eigenschaften in unterschiedlichen Repräsentationen müssen unterschiedlich behandelt werden.
- spezialisierte Techniken für unterschiedliche Arten von Repräsentationen sollten verwendet werden.
  (Zugriffsmethoden, Indexstrukturen, Distanzmaße …).
- Der Aufwand sollte möglichst nur linear mit jeder Repräsentation ansteigen.

Vereinigungs-Methode

Idee: Ein Objekt ist in einem dichten Bereich, wenn *k* Nachbarn in allen Repräsentationen in der ε-Umgebung liegen.

Geeignet für : "sparse" Daten mit viel Rauschen.

Vereinigungs-Kernobjekt:
Sei $e_1, e_2, .., e_m \in \mathcal{R}^*$, $MinPts \in N$, $o \in O$ ist ein **Vereinigungs-Kernobjekt**, falls

$$\left| \bigcup_{R_i(o) \in o} N_{\varepsilon_i}^{R_i}(o) \right| \geq MinPts$$

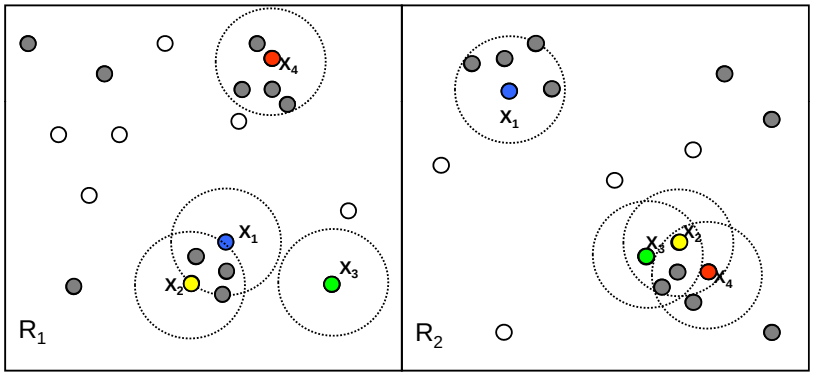
, wobei $N_{\varepsilon_i}^{R_i}(o)$ die lokale *e*-Nachbarschaft in Repr. *i* ist.

Direkte Vereinigungserreichbarkeit:
Objekt $p \in O$ ist **direkt vereinigungserreichbar** von $q \in O$ bzgl. $e_1, e_2, .., e_m$ und *MinPts,* falls *q* ein Vereinigungs-Kernobjekt in *O* ist und es gilt:

$$\exists \; i \in \{1, \ldots, m\} : R_i(p) \in N_{\varepsilon_i}^{R_i}(q)$$

Vereiniguns-Methode

Clusterexpansion bei der Vereinigungsmethode



MinPts = 3

Schnitt-Methode

Idee: Ein Objekt ist in einem dichten Bereich, falls es *k* Objekte in den *e*-Nachbarschaften aller Repräsentationen gibt.

Geeignet für: dichte Repräsentationen and unzuverlässige lokale Feature-Vektoren.

Schnitt-Kernobjekt:
Sei $e_1, e_2, .., e_m \in \Re^+$, *MinPts* $\in N$. o $\in$ O ist ein **Schnitt-Kernobjekt**, falls

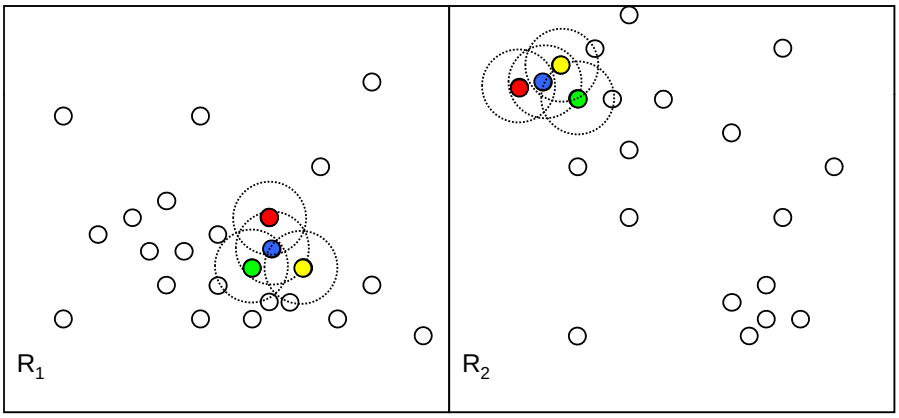
$\left| \bigcap_{R_i(o) \in o} N_{\varepsilon_i}^{R_i}(o) \right| \geq MinPts$, wobei $N_{\varepsilon_i}^{R_i}(o)$ die lokale *e*-Nachbarschaft in Repr. *i* ist.

Direkt schnitterreichbar:
Objekt $p \in O$ ist **direkt schnitterreichbar** von $q \in O$ bzgl.
$e_1, e_2, .., e_m$ und *MinPts,* falls *q* ein Schnitt-Kernobjekt in *O* ist und es gilt:

$$\forall \quad i \in \{1, \ldots, m\} : R_i(p) \in N_{\varepsilon_i}^{R_i}(q)$$

Schnitt-Methode

Clusterexpansion mit Schnitt-Methode



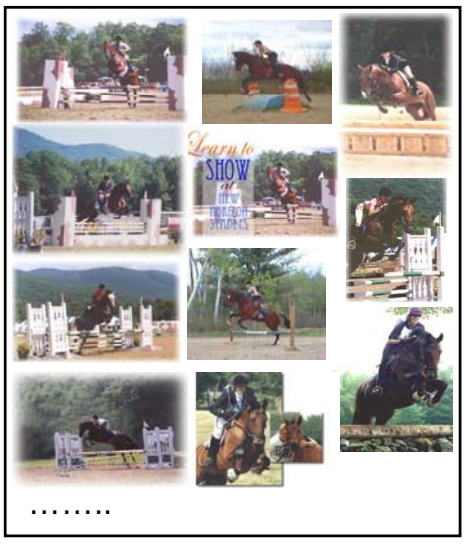
MinPts = 3

Beispiel-Ergebnisse auf Bilddaten

Cluster in den einzelnen Repr.

Beispiele für Bilder im Cluster IC 5
(nur Farbhistogramme)

Beispiele für Bilder im Cluster IC 5
(nur Segmentbäume)

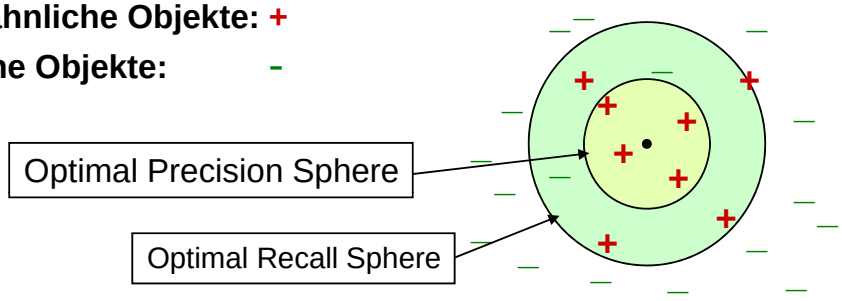


Cluster IC5 der auf beiden Repräsentationen mit der Intersectionmethode gebildet wurde.

Bedeutung der Repräsentationen

wirklich ähnliche Objekte: +

unähnliche Objekte: -



Möglichen Interpretationen der ε-Nachbarschaft:

hohe Precision- und Recall-Werte
=>1 Rep. lässt gutes Clustering zu

niedrige Precision- und Recall-Werte
=> alle Rep. lassen kein gutes Clustering zu

hohe Precision- aber niedrige Recall-Werte
$\Rightarrow$ benutze Vereinigungs-Methode

niedrige Precision- aber hohe Recall-Werte
$\Rightarrow$ verwende Schnitt-Methode

Literatur

- T. G. Dietterich: **Ensemble methods in machine learning.** In: Multiple Classier Systems, First International Workshop, MCS 2000, Cagliari, Italy, 2000.
- T. G. Dietterich: **Ensemble learning**. In: M. A. Arbib, editor, The Handbook of Brain Theory and Neural Networks. MIT Press 2003.
- J. Fürnkranz: **Round robin classification.** In: Journal of Machine Learning Research, 2:721-747, 2002.
- P.-N.Tan, M. Steinbach, and V. Kumar: **Introduction to Data Mining**, Addison-Wesley, 2006, Kapitel 5.6+5.8.
- G. Valentini and F. Masulli: **Ensembles of learning machines**. In: Neural Nets WIRN Vietri 2002.
- Kailing K., Kriegel H.-P.,Pryakhin A., Schubert M.: ***Clustering Multi-Represented Objects with Noise*** Proc. 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining **(PAKDD'04)**, 2004.
- Blum. A, Mitchell T.: ***Combining Labeled and Unlabeled Data with Co-Training***, Workshop on Computational Learning Theory (COLT 98),1998