technische universität dortmund

# Maschinelles Lernen

Skript zur Vorlesung

Katharina Morik

September 17, 2013

LS 8 Künstliche Intelligenz
Fakultät für Informatik
Technische Universität Dortmund

# Inhaltsverzeichnis

# Kapitel 1

# Einführung

## 1.1 Anwendungen maschinellen Lernens

**Bekannte Anwendungen**

- Google ordnet die Suchergebnisse nach der Anzahl der auf sie verweisenden Hyperlinks an.
- Amazon empfiehlt einem Kunden, der A gekauft hat, das Produkt B, weil alle (viele) Kunden, die A kauften, auch B kauften.
- Die Post sortiert handbeschriftete Briefe per Schrifterkennung.
- Firmen platzieren ihre Werbung gemäß der Vorlieben, die aus Suchanfragen, Bewegungsmustern,.. gelernt wurden.
- Industrie 4.0 integriert die Datenanalyse aus Prozessdaten in die Fertigung.

**Big Data**

- WWW ist die größte Datensammlung
- Tweets, Nachrichten, Videos, Fotos, Blogs,
- Einkaufstransaktionen, Verkehrsdaten, Mobilfunk-Daten,
- Sensor-Daten in Autos, Flugzeugen, Smartphones
- Physikalische Experimente produzieren Petabyte

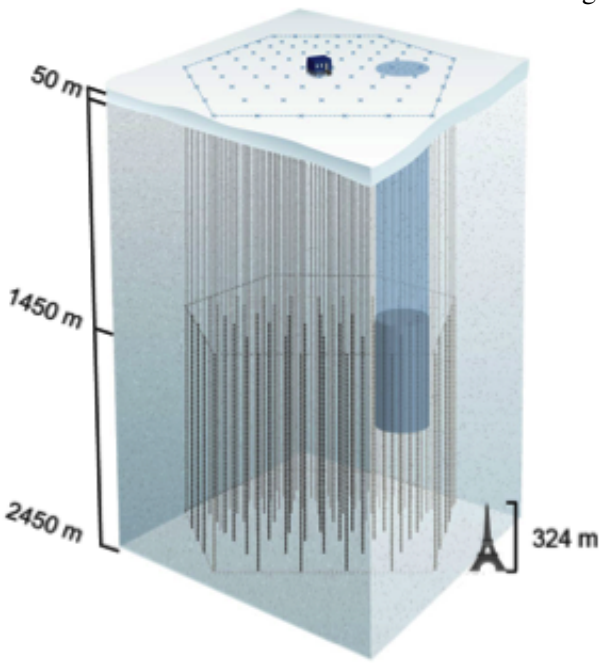
$$1Peta = 1000000000000000 = 10^{15} = 1TausendTera = 1MillionGiga$$

**WWW**


**Astrophysik**

Daten werden schneller per Schiff transportiert als per Satellit übertragen!

Neutrino-Beobachtung am Südpol zur Entdeckung astrophysikalischer Ereignisse produziert 1 TB pro Tag, 365 TB pro Jahr. Der Satellit braucht etwa 10 Jahre zur Übertragung vom Südpol nach Wisconsin,




das Schiff 28 Tage.

**Big Data als Vorteil**

Im WWW oder in den Petabyte an Messwerten etwas zu finden ist schwierig. Manchmal macht die Datenfülle es aber einfach, ein Muster zu erkennen. Beispiel: Pizzaauslieferung Freitag nachts in Manhattan.


**Produktion**

Embedded Analytics kann Messwerte ausnutzen, um bei der Produktion Ressourcen zu schonen!

## 1.2 Lernen beim Menschen

**Was ist Lernen beim Menschen?**

Menschen lernen durch:

- Auswendig lernen.
- Einüben. (Fertigkeiten)
- Logisch schließen:

  - Alle Menschen sind sterblich.
    Sokrates ist ein Mensch.
    Sokrates ist sterblich. (Deduktion)
  - Sokrates, Uta, Udo, Veronika, Volker, ... sind Menschen.
    Sokrates, Uta, Udo, Veronika, Volker, ... sind sterblich.
    Alle Menschen sind sterblich. (Induktion)

- Begriffe bilden.
- Grammatiken lernen.
- Gesetze entdecken.
- Theorien entwickeln. (Wissen)

### 1.2.1 Begriffsbildung

**Begriffsbildung**

- Eins von diesen Dingen gehört nicht zu den anderen!

**Clustering**
*Kategorisierung*

- Alle Beobachtungen, die sich ähneln, werden zu einer Gruppe zusammengefasst.
- Auf diese Weise strukturiert man die vielen Beobachtungen.
- Von den vielen Merkmalen wählt man zur Ähnlichkeitsbestimmung eine möglichst kleine Anzahl aus.
- Die ausgewählten Merkmale sind immer erkennbar (operational).

Die Kategorisierung ordnet jede Beobachtung mindestens einer Gruppe zu. Die Gruppen können sich überlappen. Menschen kategorisieren immer, ob sie wollen oder nicht! Es ist ein unbewusster kognitiver Prozess.

**Einige Gründe für die Kategorisierung**

- Handlungen können nicht auf der Gesamtheit der Beobachtungen ausgeführt werden. Menschen haben eine beschränkte Wahrnehmungs- und Aktionskapazität.
  - Menschen können nur 5-7 kognitive Objekte gleichzeitig beachten (ansehen, hören, merken).
  - Hände können nur eine begrenzte Anzahl physikalischer Objekte fassen.
  - Deshalb muss eine große Grundgesamtheit für Menschen in kleine, wahrnehmbare, handhabbare Untermengen aufgeteilt werden.
- Es gibt schon ein Wort dafür.
  - Jemand nennt ein Objekt x *Tasse*.
  - Alle Objekte, die von jemandem als *Tasse* bezeichnet wurden, gehören in eine Gruppe mit dem Titel *Tasse*.

**Positive Beispiele**

- Dies sind Tassen.

**Negative Beispiele**

- Dies sind keine Tassen.

**Klassifikation**

- Eine Funktion ordnet einer Wahrnehmung eine Klasse zu.
  - Dem Wort *Tasse* entspricht eine Erkennungsfunktion, die jeder Wahrnehmung die Klasse *Tasse* oder *Nicht-Tasse* zuordnet.
- Die einfachste Funktion ist das Aufzählen. Dies begrenzt aber die Klassifikation auf bereits gesehene Objekte.
- Als Wissenschaftler verwenden Menschen gern numerische Funktionen.
- Besonders verständlich sind logische Funktionen. Dies sind meist Definitionen.

**Definitionen**

Eine Definition ist eine Erkennungs- und Ergänzungsfunktion (hinreichende und notwendige Bedingungen).

**Definition:** Eine Tasse ist ein Behälter mit flachem Boden und einem Henkel an der Seite.

**Erkennungsfunktion:** Aha, konkav und undurchlässig, flacher Boden, Henkel an der Seite – eine Tasse!
$konkav(x), opak(x), hatBoden(x, y), flach(y), hatHenkel(x, z) \rightarrow tasse(x)$

**Ergänzungsfunktion:** Kann ich eine Tasse hinstellen? – Ja, denn eine Tasse hat einen flachen Boden und Objekte mit flachem Boden stehen sicher!
$tasse(x) \rightarrow kannStehen(x)$

**Ein Begriff erleichtert oft die Definition anderer Begriffe.**

- Wer nicht weiß, was ein *Boden* oder ein *Henkel* ist, hat Probleme, eine *Tasse* zu definieren.
- Die Definition für *Boden* und *Henkel*
  ... $\rightarrow hatBoden(x, y)$
  ... $\rightarrow hatHenkel(x, z)$
  erlaubt die Definition von *Tasse*:
  $konkav(x), opak(x), hatBoden(x, y), flach(y), hatHenkel(x, z) \rightarrow tasse(x)$

**Menschliches Lernen**

- Die kognitive Psychologie untersucht das menschliche Lernen.
- Die Entwicklungspsychologie untersucht das Lernen über die Alterstufen hinweg [12].
- Einflüsse auf das Lernen werden untersucht:
  - Reihenfolge der Beobachtungen oder Lernschritte [10]
  - Umgebung beim Lernen [2]
  - Soziale Zusammenarbeit (kollaboratives Lernen) [4]
  - ...

## 1.3 Maschinelle Lernaufgaben

**Maschinelles Lernen – generische Aufgabe**

**Population:** Eine Menge von Objekten, um die es geht.

**Merkmale:** Eine Menge von Merkmalen (quantitativ oder qualitativ) beschreibt die Objekte.

**Ausgabe:** Ein quantitativer Wert (Messwert) oder ein qualitativer (label, z.B. *Tasse*) gehört zu jeder Beobachtung.

**Ein Lernverfahren** findet eine Funktion, die Objekten einen Ausgabewert zuordnet. Oft *minimiert* die Funktion einen *Fehler*.

**Modell:** Das Lernergebnis (die gelernte Funktion) wird auch als *Modell* bezeichnet.

**Notation**

ExampleSet

Meta Data View | Data View | Plot View

ExampleSet (14 examples, 1 special attribute, 4 regular attributes)

| row no. | Play | Outlook | Temperat... | Humidity | Wind |
|---|---|---|---|---|---|
| 1 | no | sunny | 85 | 85 | false |
| 2 | no | sunny | 80 | 90 | true |
| 3 | yes | overcast | 83 | 78 | false |
| 4 | yes | rain | 70 | 96 | false |
| 5 | yes | rain | 68 | 80 | false |
| 6 | no | rain | 65 | 70 | true |
| 7 | yes | overcast | 64 | 65 | true |
| 8 | no | sunny | 72 | 95 | false |
| 9 | yes | sunny | 69 | 70 | false |
| 10 | yes | rain | 75 | 80 | false |
| 11 | yes | sunny | 75 | 70 | true |
| 12 | yes | overcast | 72 | 90 | true |
| 13 | yes | overcast | 81 | 75 | false |
| 14 | no | rain | 71 | 80 | true |

- Der Raum möglicher Beobachtungen wird als p-dimensionale Zufallsvariable $X$ geschrieben.
- Jede Dimension der Beobachtungen wird als $X_i$ notiert (Merkmal).
- Die einzelnen Beobachtungen werden als $\vec{x_1}, ..., \vec{x_N}$ notiert.
- Die Zufallsvariable $Y$ ist die Ausgabe (label).
- $N$ Beobachtungen von Vektoren mit $p$ Komponenten ergeben also eine $N \times p$-Matrix.

**Lernaufgabe Clustering**

Gegeben

- eine Menge $\mathcal{T} = \{\vec{x_1}, ..., \vec{x_N}\} \subset X$ von Beobachtungen,
- eine Anzahl $K$ zu findender Gruppen $C_1, ..., C_K$,
- eine Abstandsfunktion $d(\vec{x}, \vec{x'})$ und
- eine Qualitätsfunktion.

Finde

- Gruppen $C_1, ..., C_K$, so dass
- alle $\vec{x} \in X$ einer Gruppe zugeordnet sind und
- die Qualitätsfunktion optimiert wird: Der Abstand zwischen Beobachtungen der selben Gruppe soll minimal sein; der Abstand zwischen den Gruppen soll maximal sein.

**Lernaufgabe Klassifikation**

Gegeben

- Klassen $Y$, oft $y \in \{+1, -1\}$,
- eine Menge $\mathcal{T} = \{(\vec{x_1}, y_1), ..., (\vec{x_N}, y_N)\} \subset X \times Y$ von Beispielen,
- eine Qualitätsfunktion.

Finde

- eine Funktion $f : X \to Y$, die die Qualitätsfunktion optimiert.

**Lernaufgabe Regression**

Gegeben

- Zielwerte $Y$ mit Werten $y \in \mathcal{R}$,
- eine Menge $\mathcal{T} = \{(\vec{x_1}, y_1), ..., (\vec{x_N}, y_N)\} \subset X \times Y$ von Beispielen,
- eine Qualitätsfunktion.

Finde

- eine Funktion $f : X \to Y$, die die Qualitätsfunktion optimiert.

**Funktionsapproximation**

Wir schätzen die wahre, den Beispielen unterliegende Funktion. Gegeben

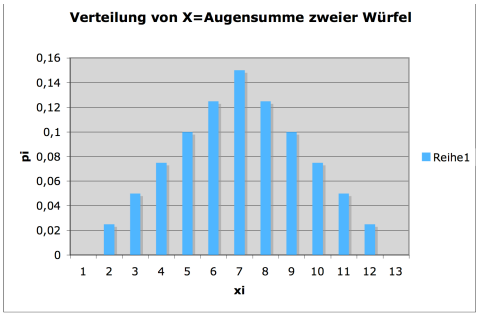
- eine Menge von Beispielen $\mathcal{T} = \{(\vec{x_1}, y_1), ..., (\vec{x_N}, y_N)\} \subset X \times Y$,
- eine Klasse zulässiger Funktionen $f_\theta$ (Hypothesensprache),
- eine Qualitätsfunktion,
- eine feste, unbekannte Wahrscheinlichkeitsverteilung $P(X)$.

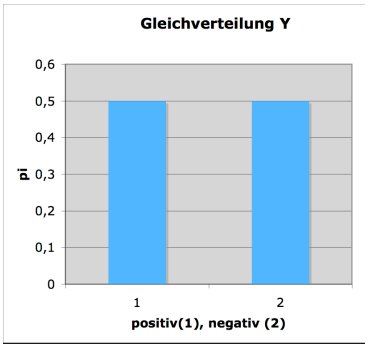
Finde

- eine Funktion $f_\theta : X \to Y$, die die Qualitätsfunktion optimiert.

**Zur Erinnerung: Verteilung**

Eine Zufallsvariable $X$ heißt *diskret*, wenn sie nur endlich oder abzählbar unendlich viele Werte $x_1, ..., x_m$ annehmen kann. Zu jedem Wert gehört ein Ereignis, das mit der Wahrscheinlichkeit $P(X = x_i)$ eintreten kann. Die Realisationen $x_i$ gemeinsam mit den zugehörigen Wahrscheinlichkeiten heißen *(Wahrscheinlichkeits-)Verteilung* von $X$.



**Verteilungsfunktion**

Sei $X$ eine diskrete oder stetige Zufallsvariable. Die Funktion
$D(x) = P(X \leq x), x \in \mathcal{R}$
heißt *Verteilungsfunktion* von $X$.
Bei diskreten Zufallsvariablen gilt: $D(x) = \sum_{i:x_i \leq x} p_i$
Eine Zufallsvariable heißt *stetige Zufallsvariable*, wenn ihre Verteilungsfunktion stetig ist.

**Dichtefunktion**

Die Ableitung $D'(x)$ wird *Dichtefunktion* genannt. Umgekehrt erhält man die Verteilungsfunktion durch Integration der Dichtefunktion: $D(x) = \int_{-\infty}^{x} h(t)dt$
Funktionen, die eine Dichte haben, sind absolut stetig.
Die Gesamtfläche unter dem Graphen von $h$ ist gleich 1.

**Wenn wir die Verteilung kennen, können wir eine gute Prognose machen!**

- Wenn wir wissen, dass $p_i = 0,01$ ist, dann ist es nicht so schlimm, wenn wir uns bei $x_i$ irren – wir irren uns dann selten.
- Wenn wir wissen, dass $P(Y = +1) = 0,99$ ist, dann sagen wir immer +1 voraus und sind in 99% der Fälle richtig. Wir haben nur ein Risiko von 1%, uns zu irren.

**Qualitätsfunktion – Fehlerfunktion**

**Fehlerrisiko:**

$$R(Y, f(X)) = \sum_{i=1}^{N} Q(y_i, \vec{x_i}) p(\vec{x_i}) \tag{1.1}$$

wobei $p(\vec{x_i})$ die Wahrscheinlichkeit ist, dass das Beispiel $\vec{x_i}$ aus $X$ gezogen wird.

**Mittlerer Quadratischer Fehler:**

$$MSE(Y, f(X)) = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(\vec{x_i}))^2 \tag{1.2}$$

**Mittlerer 0-1-Verlust:** $Q(Y, f(X)) = \frac{1}{N} \sum_{i=1}^{N} Q(\vec{x_i}, f)$, wobei

$$Q(y_i, f(\vec{x_i})) = \begin{cases} 0, falls & f(\vec{x_i}) = y \\ 1, falls & f(\vec{x_i}) \neq y \end{cases}$$

**Problem**

- Wir haben nur eine endliche Menge von Beispielen. Alle Funktionen, deren Werte durch die Beispiele verlaufen, haben einen kleinen Fehler.
- Wir wollen aber für *alle* Beobachtungen das richtige $y$ voraussagen. Dann sind nicht mehr alle Funktionen, die auf die Beispiele gepasst haben, gut.
- Wir kennen nicht die wahre Verteilung der Beispiele.
- Wie beurteilen wir da die Qualität unseres Lernergebnisses?

**Lern- und Testmenge**
Wir teilen die Daten, die wir haben, auf:

**Lernmenge:** Einen Teil der Daten übergeben wir unserem Lernalgorithmus. Daraus lernt er seine Funktion $f(x) = \hat{y}$.

**Testmenge:** Bei den restlichen Daten vergleichen wir $\hat{y}$ mit $y$.

**Aufteilung in Lern- und Testmenge**

- Vielleicht haben wir zufällig aus lauter Ausnahmen gelernt und testen dann an den normalen Fällen. Um das zu vermeiden, verändern wir die Aufteilung mehrfach.

  **leave-one-out:** Der Algorithmus lernt aus $N - 1$ Beispielen und testet auf dem ausgelassenen. Dies wird $N$ mal gemacht, die Fehler addiert.

- Aus Zeitgründen wollen wir den Algorithmus nicht zu oft anwenden.

  **Kreuzvalidierung:** Die Lernmenge wird zufällig in $n$ Mengen aufgeteilt. Der Algorithmus lernt aus $n - 1$ Mengen und testet auf der ausgelassenen Menge. Dies wird $n$ mal gemacht.

**Kreuzvalidierung**

- Man teile alle verfügbaren Beispiele in $n$ Mengen auf. z.B. $n = 10$.
- Für i=1 bis i=n:
  - Wähle die i-te Menge als Testmenge,
  - die restlichen $n - 1$ Mengen als Lernmenge.
  - Messe die Qualität auf der Testmenge.
- Bilde das Mittel der gemessenen Qualität über allen $n$ Lernläufen. Das Ergebnis gibt die Qualität des Lernergebnisses an.

**Fragestellungen des maschinellen Lernens**

- Welche Zusicherungen kann ich meinen Kunden geben? (Fehlerschranken)
- Wieviele Beispiele brauche ich?
- Welche Eigenschaften sollen die Beispiele haben, um gut vorherzusagen und wie finde (erzeuge) ich sie?
- Welche Modellklasse soll ich wählen?
- Welcher Algorithmus wird mit vielen Beispielen und vielen Dimensionen in kurzer Zeit fertig?

## Zusammenfassung

**Was wissen Sie jetzt?**

- Sie haben Clustering (Kategorisierung) und Klassifikation als menschliches Lernen gesehen.
- Die Lernaufgaben *Clustering, Klassifikation, Regression* haben Sie auch als Aufgaben des maschinellen Lernens gesehen.
- Sie wissen, was die *Kreuzvalidierung* ist.

**Was wissen Sie noch nicht?**

- Es gibt viele verschiedene *Modellklassen*. Damit werden die Lernaufgaben spezialisiert.
- Es gibt unterschiedliche *Qualitätsfunktionen*. Damit werden die Lernaufgaben als Optimierungsaufgaben definiert.
- Die *Algorithmen* zur Lösung der Lernaufgaben werden Sie in der Vorlesung kennenlernen und ihre Kernmethoden in den Übungen *selbst implementieren*.

## 1.4 Themen, Übungen, Scheine

**Themen**

- lineare Modelle und k neirest Neighbor und das Problem von *bias* und *variance*
- Stützvektormethode (SVM) und strukturelle Risikominimierung mit verschiedenen Algorithmen zur Lösung des Optimierungsproblems
  - Klassifikation von Texten
- Entscheidungsbäume
- Merkmalsselektion
- Graphische Modelle
  - Informationsextraktion aus Texten
- K-Means Clustering
- Tag Clustering
  - Clustering von Texten nach ihren Annotationen

**Grundidee der Vorlesung**

Die Vorlesung behandelt die Themen unter drei Aspekten:

- Theorie: abstrakte Darstellung der Lernaufgabe, ihrer Annahmen, Eigenschaften. Dies gründet sich auf die statistische Lerntheorie [8]. Als Mathe-Buch kann man dazu verwenden [14] und [6].
- Algorithmik: wie löst man nun also die Lernaufgabe?

  Verschiedene Algorithmen am Beispiel der SVM
- Praxis: Realistische Anwendungen werden bearbeitet, vom Datensatz zum Report.

**Übungen**

Wir verwenden das System RapidMiner und können damit

- (fast) alle Lernverfahren und Transformationen der Daten durchführen
- den Kern bestimmter Lernverfahren selbst implementieren und in der RapidMiner-Umgebung ablaufen lassen.

Wir gehen die Fragestellungen bei Anwendungen durch und Sie erwerben *know-how*, das nicht in Büchern steht.

**Wofür bekommen Sie einen Schein?**

- Kommen Sie in jede Vorlesung – dann können Sie auch das Tempo bestimmen und Fragen stellen.
- Gehen Sie in die Übungsgruppe!
- Lösen Sie jede Übungsaufgabe: Werden 80% der Punkte erreicht, bekommt man einen Schein.
- Nutzen Sie die Vorlesung/Übung zur Vorbereitung auf eine Fachprüfung!

**Wir sehen uns...**

In der ersten Übung wird RapidMiner vorgestellt. Sie findet statt:
Am Donnerstag 17.10.2011
um 14 Uhr
in OH 14 Raum 104
Wenn Sie einen Laptop haben, bringen Sie ihn mit. Sie können auch schon bei
http://rapid-i.com/
kostenlos RapidMiner herunterladen.

# Kapitel 2

# Ein erster Lernalgorithmus: $k$-Nearest-Neighbors

## 2.1 kNN zur Klassifikation, Regression

### Globale und lokale Modelle

- Lineare Modelle finden eine trennende Hyperebene.
- Die Ebene trennt alle Beispiele auf einmal. Es ist ein *globales Modell*.
- Klassifiziert man ein Beispiel nur anhand der Beispiele seiner *Umgebung*, spricht man von einem *lokalen Modell*.
- Nächste Nachbarn sind ein lokales Modell.

### Nächste Nachbarn

- Das $k$NN-Modell betrachtet nur die $k$ nächsten Nachbarn eines Beispiel $\vec{x}$:

$$\hat{f}(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i \tag{2.1}$$

- Die Nachbarschaft $N_k(\vec{x})$ wird durch ein Abstandsmaß, z.B. den Euklidschen Abstand bestimmt.
- Es gibt maximal $\frac{N}{k}$ Nachbarschaften und in jeder bestimmen wir den Durchschnitt (2.1).

### Regression und Klassifikation

**Regression** $f : X \to Y$ mit $y \in \mathcal{R}$

Gleichung (2.1) gibt als Regressionsfunktion den Mittelwert der $y_i$ zurück.

$$\hat{f}(\vec{x}) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})} y_i$$

**Klassifikation** $f : X \to Y$ mit $y \in \{+1, -1\}$

Durch einen Schwellwert können wir aus der Regression eine Klassifikation machen:

$$\hat{y} = \begin{cases} 1, & falls\ \hat{f}(\vec{x}) \geq 0,5 \\ 0, & sonst \end{cases}$$

**Die grünen und roten Datenpunkte werden in Nachbarschaften gruppiert**

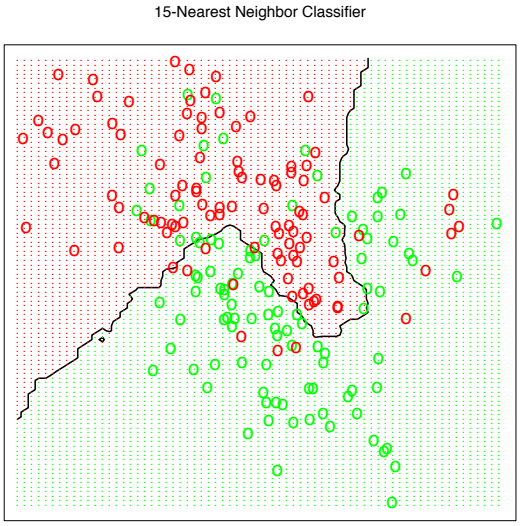



Figure 2.2: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

**Bei k=1 wird nur auswendig gelernt.**

- Falls $\vec{x} = \vec{x}' \to y = y'$, gibt es bei $k = 1$ keinen Trainingsfehler.
- Wenn allein der Trainingsfehler das Optimierungskriterium ist, würden wir stets $k = 1$ nehmen und nur auswendig lernen.
- Vermutlich ergibt das auf den Testdaten einen großen Fehler!

**Overfitting**

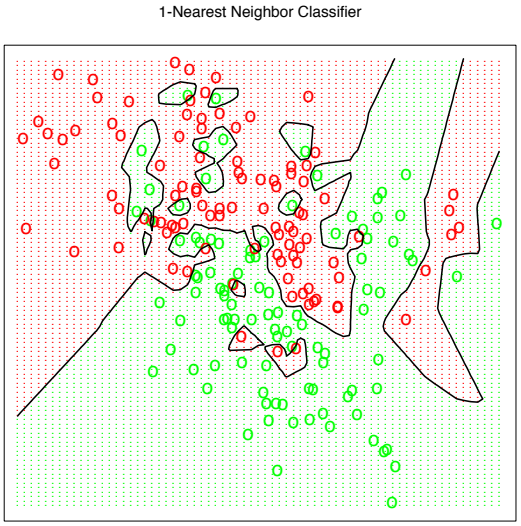



Figure 2.3: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

**Training- und Testfehler bei verschiedenen $k$**

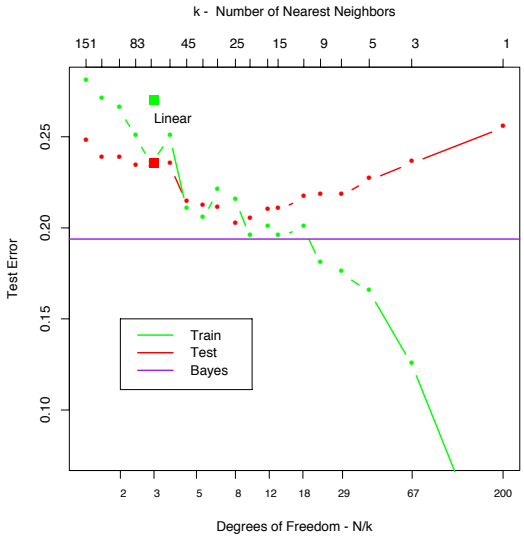



Figure 2.4: *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size* 200 *was used, and a test sample of size* $10,000$. *The red curves are test and the green are training error for k-nearest-neighbor classification. The results for linear regression are the bigger green and red dots at three degrees of freedom. The purple line is the optimal Bayes Error Rate.*

### 2.1.1 Bias und Varianz bei kNN

**Erwartungswert von $Y$ bei $k$ Nächsten Nachbarn**

- Der *Erwartungswert* von $Y$, $E(Y) = \sum_{i=1}^{N} y_i p_i$, geht bei up in den Fehler ein: $EPE(\hat{f}) = E(Y - \hat{f}(X))^2$.
- $k$NN verwendet den Erwartungswert von $Y$ direkt zur Vorhersage, allerdings beschränkt auf die Nachbarschaft $E(Y) = \frac{1}{k} \sum_{\vec{x}_i \in N_k(\vec{x})}^{k} y_i$.
- Für die Vorhersage sind wir an bedingten Wahrscheinlichkeiten interessiert $P(Y|X = \vec{x})$.
- Bei $k$NN wird die *bedingte Wahrscheinlichkeit* auf die Nachbarschaft begrenzt $E(Y|\vec{x}_i \in N_k(\vec{x}))$.
- Gerechnet wird dies mit Hilfe Gleichung (2.1).

**Asymptotisches Ergebnis zu $k$NN**

- Wenn $k/N$ gegen 0 und $N, k$ gegen $\infty$ konvergieren, konvergiert auch $\hat{f}(x)$ gegen $E(Y|X = x)$. (Hastie/etal/2001, S. 19)
- Haben wir also den perfekten Lernalgorithmus gefunden?

### Fluch der hohen Dimension bei $k$NN

- Die Dichte der Beispiele ist proportional zu $N^{\frac{1}{p}}$.
- Schon bei $p = 10$ brauchen wir 80% der möglichen Werte jedes Attributs $X_i$, um wenigstens 10% der Daten in einer Nachbarschaft gesehen zu haben!
- Die Dichte der Datenpunkte in der Nachbarschaft ist bei hoher Dimension furchtbar spärlich.
  - $N^{\frac{1}{p}}$ ist bei 100 Beispielen und $p = 10$ nur $100^{1/10} = \sqrt[5]{10}$.
  - Wenn 100 Beispiele bei $p = 1$ einen dichten Raum ergeben, muss man für die selbe Dichte bei $p = 10$ schon $100^{10}$ Beispiele sammeln: $100^{1/1} = 100, 100^{10 \cdot \frac{1}{10}} = 100$

### Bias und Varianz bei $k$NN

- Wenn man die richtige, dicht besetzte Nachbarschaft hat, verzerrt $k$NN die Vorhersage nicht (*kleiner Bias*).
- Wenn - wie bei hohen Dimensionen - die Nachbarschaft wild variiert, schwankt auch die Güte der Vorhersage (*große Varianz*).

### Bias und Varianz – bildlich

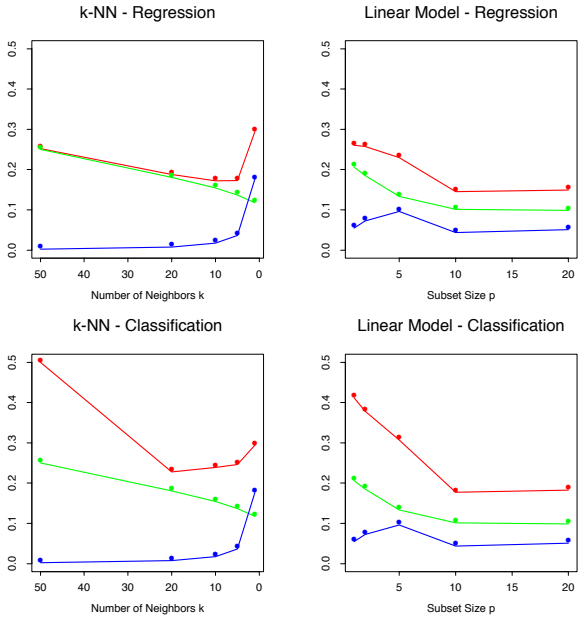



Figure 7.3: *Prediction error (red), squared bias (green) and variance (blue) for a simulated example. The top row is regression with squared error loss; the bottom row is classification with* 0–1 *loss. The models are* $k$*-nearest neighbors (left) and best subset regression of size* $p$ *(right). The variance and bias curves are the same in regression and classification, but the prediction error curve is different.*

**Bias, Varianz und Modellkomplexität – bildlich**

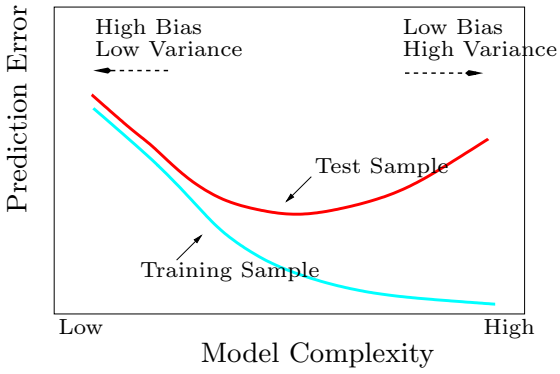



Figure 7.1: *Behavior of test sample and training sample error as the model complexity is varied.*

- Dieser Zusammenhang von Training-/Testfehler und Komplexität bestimmt alle Lernverfahren.
- Kreuzvalidierung lasst abschätzen, wie gut das Modell zu den Daten passt (Modellselektion).

**Was wissen Sie jetzt?**

- Sie kennen den Fluch der hohen Dimension bei kNN: kleiner Bias, aber hohe Varianz.
- Bei linearen Modellen ist es umgekehrt: kleine Varianz, aber hoher Bias (falls die Annahme des linearen Zusammenhangs von $X, Y$ nicht stimmt).

## 2.2 kNN implementieren

**Rahmen – Instanzbasiertes Lernen**

- Alle Beispiele werden gespeichert.
  - Geschickt indexieren?
  - Typische Beispiele auswählen?
- Zu einer neuen Beobachtung $\vec{x}$ werden die $k$ ähnlichsten Beispiele $\vec{x}'$ gefunden: $N_k(\vec{x})$.
  - Ähnlichkeitsmaß $Sim(\vec{x}, \vec{x}')$?
- und daraus gemäß einer Entscheidungsfunktion der $y$-Wert ermittelt.
  - Maximum, Mehrheit, Mittelwert?

**Mögliche Datenstrukturen**

- Alle Beispiele in ein $double[]values$ mit allen Attributwerten und dem Label kopieren, damit man schnell iterieren kann;
- $BoundedPriorityQueue$, die das kleinste Element vorn hat
- Tupel (Distanz zum zu klassifizierenden Beispiel, Beispiel)
- Counter für jedes Label, um die Häufigkeit zu zählen

**Entscheidungsfunktionen zur Klassifikation**

- Mehrheitsentscheidung (Gleichung (2.1)):

$$f(\vec{x}) = \frac{1}{k} \sum_{\vec{x_i} \in N_k(\vec{x})} y_i$$

- Gewichtete Entscheidung nach tatsächlicher Ähnlichkeit $w_i = Sim(\vec{x}, \vec{x_i})$:

$$f(\vec{x}) = \frac{1}{k} \sum_{\vec{x_i} \in N_k(\vec{x})} w_i \cdot y_i$$

- Bewahrt ein Schwellwert $Sim(\vec{x}, \vec{x_i}) \leq \theta \rightarrow \vec{x_i} \notin N_k(\vec{x})$ vor großer Varianz?
- Probieren Sie und vergleichen Sie die Ergebnisse der Kreuzvalidierung!

### 2.2.1 Ähnlichkeitsmaße

**Ähnlichkeit – Maße**

- Ähnlichkeit oder Distanz sollte stets Werte in $[0, 1]$ haben.
- $dist(\vec{x}_1, \vec{x}_2) = 1 - sim(\vec{x}_1, \vec{x}_2)$
- Eine Metrik erfüllt die Bedingungen
  1. $Metrik(x, x) = 0$
  2. $Metrik(x_1, x_2) = Metrik(x_2, x_1)$
  3. $Metrik(x_1, x_3) \leq Metrik(x_1, x_2) + Metrik(x_2, x_3)$

**$sim$: Ähnlichkeit für einzelne Attribute**

**Numerische Attribute:** Sei $max_j$ der höchste Wert von $X_j$ und $min_j$ der niedrigste, sei $x_{i,j}$ der Wert des $j$-ten Attributs in der $i$-ten Beobachtung, dann ist z.B.

$$sim_j(x_{1,j}, x_{2,j}) = 1 - \frac{|x_{1,j} - x_{2,j}|}{max_j - min_j}$$

ein Ähnlichkeitsmaß für $X_j$.

**Nominale Attribute:** Ganz einfach:

$$sim_j(x_{1,j}, x_{2,j}) = \begin{cases} 1 & falls \quad x_{1,j} = x_{2,j} \\ 0 & sonst \end{cases}$$

**$Sim$: Ähnlichkeit der Beispiele als Kombination der Attributähnlichkeiten**

Im einfachsten Fall mitteln wir die Einzelähnlichkeiten:

$$Sim(\vec{x_1}, \vec{x_2}) = \frac{1}{p} \sum_{j=1}^{p} sim(x_{1,j}, x_{2,j})$$

Vielleicht sind einige Attribute wichtiger als andere?

$$Sim(\vec{x_1}, \vec{x_2}) = \frac{\sum_{j=1}^{p} w_j sim(x_{1,j}, x_{2,j})}{\sum_{j=1}^{p} w_j}$$

Vielleicht ist der quadratische Abstand besser?

$$Sim(\vec{x_1}, \vec{x_2}) = 1 - \sum_{j=1}^{p} w_j (x_{1,j} - x_{2,j})^2$$

Wie bestimmt man $w_j$?

**Spielräume**

Sie sehen, es gibt viele Varianten in einer *Klasse* von Verfahren, die instanzbasiert oder nächste Nachbarn heißt. Das bedeutet für

**die Implementierung:** Es sollte eine Rahmenklasse geben, der Ähnlichkeits- und Fehlermaße übergeben werden – nicht für jedes Kriterium ein neues Programm!

**die Anwendung:** Bei jedem neuen Datensatz müssen die Kriterien wohl überlegt werden – Modellselektion!

**die Theorie:** Wir wollen Eigenschaften ermitteln, die für alle Verfahren einer Klasse gültig sind, z.B. Bias und Varianz.

# Kapitel 3

# Modellselektion

## 3.1 Funktionsapproximation

**Funktionsapproximation**

Allgemeine Aussagen über Modelle – Fehler oder Likelihood

- Gegeben sind die Trainingsbeispiele $\mathcal{T}$, gesucht ist eine Funktion

$$f_\theta(\vec{x}) = \sum_{k=1}^{K} h_k(\vec{x})\theta_k.$$

- Dabei gibt es Parameter $\theta$, die abzuschätzen sind, bei den linearen Modellen ist dies $\beta$.
- Darüber hinaus können die Daten transformiert werden in einen Raum, der für das Lernen besser geeignet ist: $h_k(\vec{x})$.
- Optimiert wird ein Qualitätskriterium, z.B. wird eine Verlustfunktion minimiert oder die Wahrscheinlichkeit maximiert.

**Wege der Funktionsapproximation**

**Verlustfunktion:** Fehler minimieren als Abstand zwischen wahrem Wert und Ergebnis der gelernten Funktion, z.B. $RSS(\theta)$ minimieren. Das kommt nächstes Mal.

**Likelihood:** Wahrscheinlichkeit der wahren Werte maximieren! Das schauen wir uns jetzt an.

### 3.1.1 Likelihood

**Maximum Likelihood**

Gegeben eine Verteilung $Pr_\theta(y)$ und eine Stichprobe dieser Verteilung $y_1, ...., y_N$, ist die logarithmierte Wahrscheinlichkeit:

$$L(\theta) = \sum_{i=1}^{N} \log Pr_\theta(y_i) \tag{3.1}$$

Genau das $\theta$, das $y_i$ am wahrscheinlichsten macht, ist gut – $L(\theta)$ maximieren!

- Wir können dafür eine Verteilung annehmen, da wir die wahre Verteilung nicht kennen.
- Meist ist die Normalverteilung eine gute Annahme.

**Normalverteilung $\mathcal{N}(\mu, \sigma)$**

**normalverteilt**

Eine Zufallsvariable $X$ heißt **normalverteilt** mit den Parametern $\mu, \sigma$, wenn sie die Dichtefunktion

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}((x-\mu)/\sigma)^2} \tag{3.2}$$

besitzt.

**Normalverteilung**

Die zugehörige Wahrscheinlichkeitsverteilung $X \sim \mathcal{N}(\mu, \sigma^2)$ heißt **Normalverteilung**, der Graph ihrer Dichtefunktion wird Gaußsche Glockenkurve genannt.

**Log-likelihood bei nominalem Y ist Entropie**

**Cross-Entropie**

Sei $Y$ eine Zufallsvariable, die als Werte die Namen von $K$ verschiedenen Klassen annimmt.
$Pr(Y = y_k|X = \vec{x}) = p_{k,\theta}(\vec{x}), k = 1, ..., K$

$$L(\theta) = \sum_{i=1}^{N} log(p_{y_i,\theta}(\vec{x}_i)) \tag{3.3}$$

Wenn man $L(\theta)$ maximiert, passt $\theta$ gut zu den Daten im Sinne der Likelihood.

## 3.2 Modellselektion

**Verfahren zur Modellselektion**
Wie wählen wir Modelle aus?

- Kreuzvalidierung für verschiedene Modelle – das mit dem geringsten durchschnittlichen Fehler nehmen! (Minimierung der Verlustfunktion jetzt auf der Ebene der Modelle)
- Direkt anhand der a posteriori Wahrscheinlichkeit Modelle vergleichen. (Maximierung der Wahrscheinlichkeit jetzt auf der Ebene der Modelle)
  - Bayes Information Criterion
  - Minimum Description Length

### Kreuzvalidierung zur Modellselektion

### Kreuzvalidierung zur Modellselektion

Gegeben eine Klasse von Modellen $f(\vec{x}, \alpha)$, wobei $\alpha$ ein Modell der Klasse indiziert, eine Verlustfunktion $L(y, f(\vec{x}, \alpha))$, $N$ Beispiele und eine Aufteilung der Beispiele in $K$ Partitionen mit der Indexfunktion $\kappa : \{1, ..., N\} \rightarrow \{1, ..., K\}$, die für jede Beobachtung die zugehörige Partition angibt.

**Kreuzvalidierung für alle Modelle:**

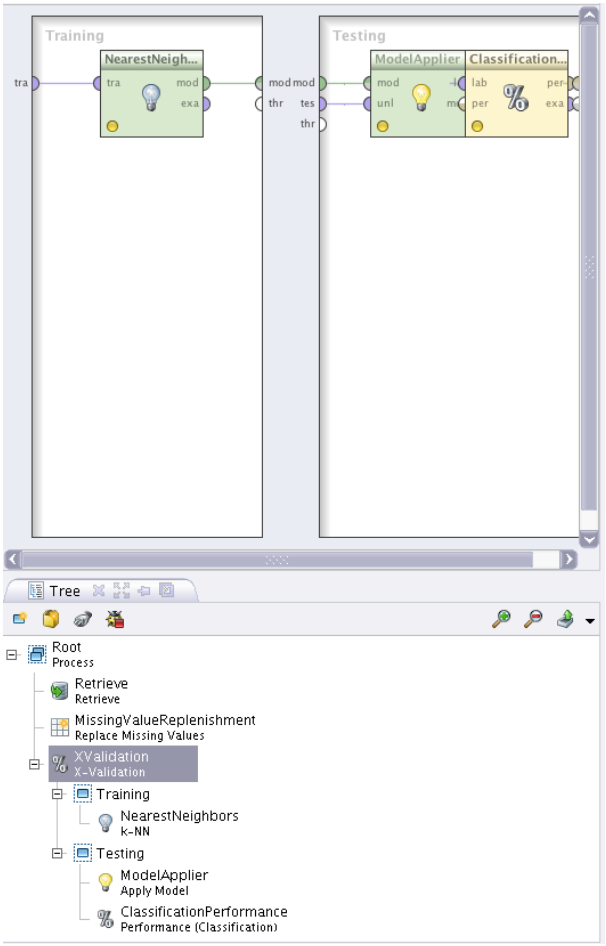
- Lasse die $\kappa(i)$-te Partition aus,
- lerne das $\alpha$-te Modell: $\hat{f}^{-\kappa(i)}(\vec{x}, \alpha)$.
- rechne den Fehler aus:

$$CV(\alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(\vec{x}_i, \alpha))$$

- Minimiere $CV(\alpha)$, wähle also das Modell mit dem geringsten Verlust.

### Modellselektion über Kreuzvalidierung praktisch

In RapidMiner wird die Kreuzvalidierungsschleife schon angeboten.




Es geht aber auch anders...

### Bayes Kriterien zur Modellselektion

### Bayes Statistik

**A posteriori Wahrscheinlichkeit**

Gegeben eine beliebige Einteilung von $X$ in Klassen $y_1, y_2, ..., y_K$ und eine Beobachtung $\vec{x} \in X$. Die Wahrscheinlichkeit von $y_j$ unter der Bedingung, dass $\vec{x}$ beobachtet wird, ist

$$Pr(y_j|\vec{x}) = \frac{Pr(y_j)Pr(\vec{x}|y_j)}{Pr(\vec{x})} \tag{3.4}$$

$Pr(y_j)$ ist die **a priori** Wahrscheinlichkeit der Klasse. $Pr(y_j|\vec{x})$ ist die **a posteriori** Wahrscheinlichkeit der Klasse.

### Bayes Modellselektion

Gegeben eine Menge von Modellen $\mathcal{M}_m, m = 1, ..., M$ mit entsprechenden Parametern $\theta_m$, Trainingsdaten $\mathcal{T}$ und eine Verteilung $Pr(\theta_m|\mathcal{M}_m)$, dann ist die a posteriori Wahrscheinlichkeit eines Modells

$$Pr(\mathcal{M}_m|\mathcal{T}) \sim Pr(\mathcal{M}_m) \cdot Pr(\mathcal{T}|\mathcal{M}_m)$$

Gegeben dass $Pr(\mathcal{M}_l|\mathcal{T}) \neq 0, Pr(\mathcal{T}|\mathcal{M}_l) \neq 0, Pr(\mathcal{M}_l) \neq 0$:
Zum Vergleich zweier Modelle $\mathcal{M}_j, \mathcal{M}_l$ berechnen wir den Quotienten:

$$\frac{Pr(\mathcal{M}_m|\mathcal{T})}{Pr(\mathcal{M}_l|\mathcal{T})} = \frac{Pr(\mathcal{M}_m)}{Pr(\mathcal{M}_l)} \cdot \frac{Pr(\mathcal{T}|\mathcal{M}_m)}{Pr(\mathcal{T}|\mathcal{M}_l)}$$

Ist das Ergebnis > 1, nehmen wir $\mathcal{M}_m$, sonst $\mathcal{M}_l$.

**Approximieren der a posteriori Wahrscheinlichkeit**
Wenn alle Modelle a priori gleich wahrscheinlich sind, müssen wir nur $Pr(\mathcal{T}|\mathcal{M}_i)$ approximieren.

- Mit Maximum Likelihood schätzen wir $\hat{\theta}_i$.
- Die Anzahl freier Parameter in $\mathcal{M}_i$ nennen wir $d_i$. Das ist z.B. die Dimension der Beispiele, kann aber wegen $h_k(\vec{x})$ oder einiger Eigenschaften des Lernverfahrens auch etwas anderes sein.
- Als Wahrscheinlichkeit nähern wir an:

$$log\ Pr(\mathcal{T}|\mathcal{M}_i) = log\ Pr(\mathcal{T}|\hat{\theta}_i, \mathcal{M}_i) - \frac{d_i}{2} \cdot log\ N + O(1) \tag{3.5}$$

**Maximale a posteriori Wahrscheinlichkeit und BIC**

**Bayes Informationskriterium**

Sei $d$ die Anzahl der Parameter eines Modells und $N$ die Anzahl der Beispiele, dann ist das Bayes Informationskriterium BIC

$$BIC = -2\ loglik + (log\ N) \cdot d \tag{3.6}$$

Dabei ist $loglik = \sum_{i=1}^{N} \log Pr_{\hat{\theta}}(y_i)$.

BIC als Qualitätskriterium bei Likelihood Maximierung wählt eher einfache Modelle. Unter einer Gaußschen Verteilung und bei bekannter Varianz $\sigma^2$ rechnen wir

$$-2\ loglik \sim \sum_i \frac{(y_i - \hat{y}_i)^2}{\sigma^2}$$

Die Wahl des Modells mit kleinstem BIC entspricht der
Wahl des Modells mit größter a posteriori Wahrscheinlichkeit.

**Relative Qualität der Modelle per BIC**

- Die Wahl des Modells mit kleinstem BIC ist zuverlässig. Gegeben eine Familie von Modellen, darunter das richtige, konvergiert die Wahrscheinlichkeit, dass BIC das richtige wählt, gegen 1, wenn die Anzahl der Beispiele gegen $\infty$ konvergiert.
- Wenn wir für jedes Modell $\mathcal{M}_m, m = 1, ..., M$ den BIC ausrechnen, können wir (wie bei Kreuzvalidierung auch) die Modelle relativ zueinander bewerten, hier:

$$\frac{e^{-\frac{1}{2}\cdot BIC_m}}{\sum_{l=1}^{M} e^{-\frac{1}{2}\cdot BIC_l}} \tag{3.7}$$

**Minimum Description Length**

Ein Modell *kodiert* eine Menge von Beispielen. Wir können Nachrichten so kodieren, dass keine Nachricht Präfix einer anderen ist, z.B.

| Nachricht | z1 | z2 | z3 | z4 |
|---|---|---|---|---|
| Code | 0 | 10 | 110 | 111 |

Wir wollen den kürzesten Code für die häufigste Nachricht. Der Code des Beispiels ist optimal, wenn $Pr(z1) = 1/2$, $Pr(z2) = 1/4$, $Pr(z3) = 1/8$, $Pr(z4) = 1/8$.

Wieso das?

**Shannon/Weaver Theorem**

**Code-Länge als Entropie**

Wählen wir die Code-Länge $l_i$ einer Nachricht $z_i$ als

$$l_i = -log_2 Pr(z_i)$$

so ist die durchschnittliche Nachrichtenlänge

$$length \geq -\sum Pr(z_i) log_2(Pr(z_i)) \tag{3.8}$$

Wenn $p_i = A^{-l_i}$, wobei $A$ die Anzahl der verwendeten Zeichen ist, gilt sogar die Gleichheit (s. Beispiel): $Pr(z_1) = 1/2 = 2^{-1} = A^{-l_1}, A = 2, l_1 = 1$

**Minimum Description Length zur Modellselektion**

Gegeben ein Modell $\mathcal{M}$ mit Parametern $\theta$ und Beispiele $\mathcal{T} = (\mathbf{X}, \mathbf{y})$, der Empfänger kennt alle $\mathbf{X}$ und soll die $\mathbf{y}$ empfangen. Dazu müssen wir den Unterschied zwischen Modell und wahren Werten sowie die Modellparameter übermitteln.

**Prinzip der Minimum Description Length MDL**

Wähle immer das Modell mit der kürzesten Nachrichtenlänge!

$$length = -log\ Pr(\mathbf{y}|\theta, \mathcal{M}, \mathbf{X}) - log\ Pr(\theta|\mathcal{M}) \tag{3.9}$$

**Eigenschaften von MDL**

- Bei normalverteilten $y, \theta$, wenn wir $\mathbf{X}$ zur Einfachheit weglassen, sehen wir den Einfluss von $\sigma$:

$$length = log\ \sigma + \frac{(y-\theta)^2}{\sigma^2} + \frac{\theta^2}{2}$$

- Je kleiner $\sigma$ desto kürzer die Nachricht und einfacher das Modell!

**Bezug zwischen MDL und BIC**

- Wenn wir die Länge (Gleichung 3.9) minimieren

$$length = -log\ Pr(\mathbf{y}|\theta, \mathcal{M}, \mathbf{X}) - log\ Pr(\theta|\mathcal{M})$$

  maximieren wir auch die a posteriori Wahrscheinlichkeit (vgl. Gleichung 3.4) $Pr(\mathbf{y}|\mathbf{X})$.

- Mit Hilfe des BIC haben wir Modelle für die Funktionsapproximation durch Maximum Likelihood ausgewählt: das Modell mit dem kleinsten BIC entspricht dem Modell mit größter a posteriori Wahrscheinlichkeit.

- Also kann man das Modell mit der kleinsten Code-Länge (MDL-Prinzip) auch durch die Minimierung des BIC finden.

**Was wissen Sie jetzt?**

- Funktionsapproximation optimiert eine *Qualitätsfunktion*.
  - *Fehlerminimierung*, z.B. MSE
  - *Maximierung der Likelihood*, z.B. durch Approximation der a posteriori Wahrscheinlichkeit
    - Maximum Likelihood ergibt bei Mehrklassenproblemen die maximale Cross-Entropie (Gleichung 3.3).
- Für die *Modellselektion* kann man
  - die Kreuzvalidierung mit Fehlerminimierung und
  - die Kriterien nach Bayes (BIC, MDL) nutzen.

# Kapitel 4

# Lineare Modelle, Bias und Varianz

## 4.1 Lineare Modelle zur Klassifikation und Regression

**Grundlagen**

Sei $X = \{X_1, \ldots, X_p\}$ eine Menge von Zufallsvariablen und $Y \neq \emptyset$ eine Menge.

Ein *Beispiel* (oder *Beobachtung*) $\vec{x}$ ist ein konkreter $p$-dimensionaler Vektor über diese Zufallsvariablen.

Eine *Menge von $n$ Beispielen* $\mathbf{X} = \{\vec{x}_1, \ldots, \vec{x}_N\}$ können wir dann als $(N \times p)$-Matrix auffassen:

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \ldots & x_{1,p} \\ x_{2,1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \ldots & x_{N,p} \end{pmatrix}$$

Dabei entspricht jede Zeile $\vec{x}_i$ der Matrix $\mathbf{X}$ einem Beispiel.

### 4.1.1 Klassifikation und Regression

**Klassifikation und Regression**

Beim *überwachten Lernen* (darum geht es hier), ist zusätzlich zu jeder Beobachtung $\vec{x}$ ein *Label* (*Klasse*) $y$ gegeben, d.h. wir haben Beobachtungen $(\vec{x}, y) \in X \times Y$.

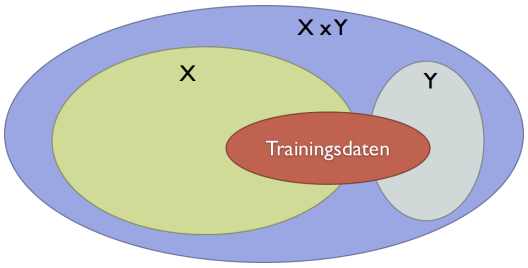
$Y$ kann sowohl eine *qualitative*, als auch eine *quantitative* Beschreibung von $\vec{x}$ sein.

Für den quantitativen Fall ist z.B. $Y = \mathbb{R}$ und wir versuchen für unbekanntes $\vec{x}$ den Wert $y$ vorherzusagen *Regression*.

Im Falle qualitativer Beschreibungen ist $Y$ eine diskrete Menge und wir nutzen $f$ zur *Klassifikation*.

**Lernen auf Trainingsdaten**

Wovon gehen wir also aus? Was ist unser Ziel?




- Wir suchen *die wahre Funktion* $f : X \to Y$ mit

$$f(\vec{x}) = y \quad \forall\, (\vec{x}, y) \in X \times Y$$

- Wir haben jedoch nur eine Teilmenge der Beobachtungen gegeben (Trainingsdaten)

**Klassifikation und Regression**

Auf Grundlage der Trainingsdaten suchen wir eine möglichst gute Annäherung $\hat{f}$ an die *wahre Funktion* $f$.

Die Funktion $\hat{f}$ bezeichnen wir auch als das gelernte *Modell*.

Haben wir ein Modell $\hat{f}$ gelernt, so liefert uns dieses Modell mit

$$\hat{y} = \hat{f}\,(\vec{x})$$

für *neue Daten* $\vec{x} \in X$ eine Vorhersage $\hat{y} \in Y$.

**Klassifikation und Regression**

Im Falle der *Regression* läßt sich so für zuvor unbekannte $\vec{x} \in X$ der Wert

$$\hat{y} = \hat{f}\,(\vec{x})$$

mit $\hat{y} \in \mathbb{R}$ vorhersagen.

Dieses Modell $\hat{f}$ läßt sich auch für die Klassifikation nutzen, bei der z.B. $\hat{y} \in \{-1, +1\}$ vorhergesagt werden sollen:

$$\hat{y} = \begin{cases} +1, & \text{falls } \hat{f}\,(\vec{x}) \geq \theta \\ -1, & \text{sonst} \end{cases}$$

Hier ist $\theta$ ein vorgegebener Schwellwert.

**Beispiel**

Gegeben seien Gewicht ($X_1$) und Größe ($X_2$) einiger Personen und ein Label $y \in \{m, w\}$:

| | $X_1$ | $X_2$ | $Y$ |
|---|---|---|---|
| $x_1$ | 91 | 190 | m |
| $x_2$ | 60 | 170 | w |
| $x_3$ | 41 | 160 | w |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |

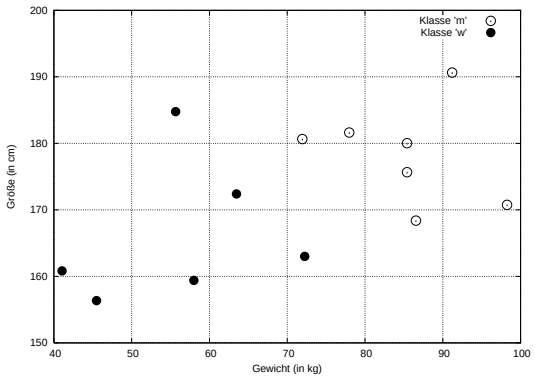
Die Tabelle enthält die zur Verfügung stehenden Trainingsdaten, also

$$\mathbf{X} = \begin{pmatrix} 91 & 190 \\ 60 & 170 \\ 41 & 160 \\ \vdots & \vdots \end{pmatrix}$$

**Beispiel**

Es wird nun eine Funktion $\hat{f}$ gesucht, die für neue Daten $\vec{x}$ das Attribut $Y$ (Geschlecht) voraussagt, also

$$\hat{y} = \begin{cases} \text{m, falls } \hat{f}(x) > \theta \\ \text{w, sonst} \end{cases}$$




### 4.1.2 Lineare Modelle

**Lineare Modelle**

Welche Art von Funktionen sind denkbar?

*Lineare Funktionen* als einfachste Funktionenklasse:

$$y = f(x) = mx + b \quad \text{Gerade im } \mathbb{R}^2$$

Allerdings betrachten wir als Beispielraum den $\mathbb{R}^p$, d.h. wir brauchen eine verallgemeinerte Form:

$$y = f(\vec{x}) = \sum_{i=1}^{p} \beta_i x_i + \beta_0 \quad \text{mit } \beta_0 \in \mathbb{R}, \vec{x}, \vec{\beta} \in \mathbb{R}^p \tag{4.1}$$

Die Funktion $f$ wird also durch $\vec{\beta}$ und $\beta_0$ festgelegt und sagt uns für ein gegebenes $\vec{x}$ das entsprechende $y$ voraus

**Notation, Vereinbarungen**

Bei genauerer Betrachtung von Formel (4.1) läßt sich $\sum_{i=1}^{p} \beta_i x_i$ als Matrizenmultiplikation oder Skalarprodukt schreiben, also

$$y = \sum_{i=1}^{p} \beta_i x_i + \beta_0 = \vec{x}^T \vec{\beta} + \beta_0 = \left\langle \vec{x}\,,\, \vec{\beta} \right\rangle + \beta_0$$

Zur einfacheren Darstellung von $f$, wird $\beta_0$ in den Vektor $\vec{\beta}$ codiert, indem jedes Beispiel $x = (x_1, \ldots, x_p)$ aufgefasst wird als $(p+1)$-dimensionaler Vektor

$$(x_1, \ldots, x_p) \mapsto (1, x_1, \ldots, x_p)$$

Dies ermöglicht die Darstellung von $f$ als:

$$y = f(\vec{x}) = \sum_{i=0}^{p} \beta_i x_i = \vec{x}^T \vec{\beta} = \left\langle \vec{x}\,,\, \vec{\beta} \right\rangle$$

**Was haben wir nun gemacht?**

Wir haben (bei der Beschränkung auf lineare Modelle) nun eine Darstellung für das, was wir *lernen* wollen:

$$y = \hat{f}(\vec{x}) = \vec{x}^T \vec{\beta}$$

Wir haben die Zielfunktion $\hat{f}$ in Abhängigkeit von $\vec{\beta}$ geschrieben und müssen *nur noch* das passende $\vec{\beta}$ finden.

### 4.1.3 Geometrie linearer Modelle: Hyperebenen

**Veranschaulichung**

Bevor wir uns an die Wahl des passenden $\vec{\beta}$ machen, zunächst einige Vorüberlegungen.
Betrachten wir dazu die binäre Klassifikation ($Y = \{-1, +1\}$):

- Was passiert dabei eigentlich anschaulich?
- Wie klassifiziert unser $\hat{f}$ die Daten?
- Wie wirkt sich die Wahl von $\vec{\beta}$ aus?

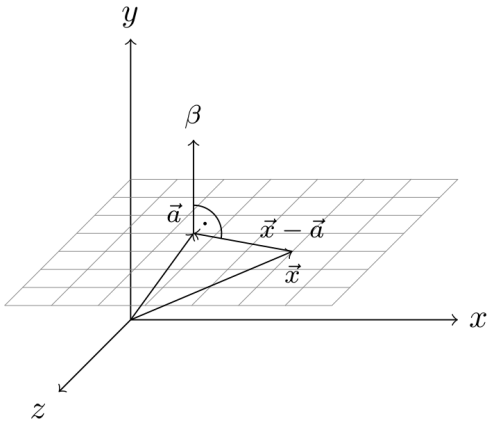
**Zur Erinnerung: Ebenengleichung**

Sei $V = \mathbb{R}^p$ ein Vektorraum, dann ist eine Hyperebene $H$ ein $(p-1)$-dimensionaler affiner Untervektorraum.

$H$ lässt sich über einen Stützvektor $\vec{a}$ und einen Normalenvektor $\vec{\beta}$ mit der Ebenengleichung schreiben

$$H = \left\{ x \in \mathbb{R}^p \mid \left\langle \vec{\beta}\,,\, \vec{x} - \vec{a} \right\rangle = 0 \right\}$$

**Beispiel Ebene, Stützvektor, Normalenvektor**




(Hyper-) Ebene im $\mathbb{R}^3$ mit Normalenvektor $\vec{\beta}$ und Stützvektor $\vec{a}$. Falls $\langle \vec{\beta}, \vec{x} - \vec{a} \rangle = 0$, also $\vec{\beta}$ und $\vec{x} - \vec{a}$ orthogonal zueinander, befindet sich $\vec{x}$ auf der Ebene.

**Hesse Normalform**

Multiplizieren wir die Ebenengleichung aus und setzen $\beta_0 = \left\langle \vec{\beta}\,,\, \vec{a} \right\rangle$, dann ist

$$\left\langle \vec{\beta}\,,\, \vec{x} \right\rangle - \beta_0 = 0$$

in *Hesse Normalform*, falls $||\vec{\beta}|| = 1$.

**Zur Erinnerung: Skalarprodukt**

Das Skalarprodukt ist definiert durch

$$\langle \vec{v}\,,\, \vec{w} \rangle = \vec{v}^T \vec{w} = \sum_{i=1}^{p} v_i w_i$$

aber auch durch den Kosinus mit

$$\langle \vec{v}, \vec{w} \rangle = \|\vec{v}\| \cdot \|\vec{w}\| \cdot \cos(\measuredangle(\vec{v}, \vec{w}))$$

*Beispiel:*

| | $\vec{w}$ : |
|---|---|
| | 4 |
| | 5 |
| | 6 |
| $\vec{v}^T$ : 1 2 3 | $1 \cdot 4 + 2 \cdot 5 + 3 \cdot 6 = 32$ |

**Zur Erinnerung: Euklidsche Länge**

**Euklidsche Länge oder Norm**

$$\| \vec{\beta} \| = \sqrt{\sum_{i=1}^{p} \beta_i^2} = \sqrt{\vec{\beta}^T \vec{\beta}} = \sqrt{\left\langle \vec{\beta}\,,\, \vec{\beta} \right\rangle}$$

weil $\| \vec{\beta} \|^2 = x_1^2 + ...x_p^2$ (Pythagoras)

**Beispiel:** $\vec{\beta} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \parallel \vec{\beta} \parallel = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14}$

**Normiert** heißt ein Vektor, wenn er die (Euklidsche) Länge 1 hat.

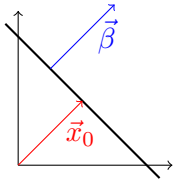
**Abstandsberechnung durch Hesse Normalform**

Sei $\vec{x}_0$ der Vektor, dessen Länge der Abstand vom Ursprung zur Ebene in *Hesse Normalform* ist. Dieser muss orthogonal zur Ebene liegen und somit parallel zu $\vec{\beta}$. Seien nun $\vec{\beta}$ und $\vec{x}_0$ gleichgerichtet, dann gilt

$$\cos(\measuredangle(\vec{\beta}, \vec{x}_0)) = 1$$

und $\|\beta\| = 1$ und somit

$$\begin{array}{rcl} \left\langle \vec{\beta}\,,\, \vec{x}_0 \right\rangle - \beta_0 & = & 0 \\ \Leftrightarrow \quad \|\vec{\beta}\| \cdot \|\vec{x}_0\| \cdot \cos(\measuredangle(\vec{\beta}, \vec{x}_0)) & = & \beta_0 \\ \Leftrightarrow \quad \|\vec{x}_0\| & = & \beta_0 \end{array}$$




Daraus folgt, dass $\beta_0$ der Abstand der Ebene zum Ursprung ist.

**Hesse Normalform**

Für die Hesse Normalform muss $\|\vec{\beta}\| = 1$ gelten, damit der Abstand zum Ursprung leicht abgelesen werden kann. Wir normieren den Normalenvektor auf die Euklidsche Länge 1

$$\vec{\beta} = \frac{\vec{\beta}'}{\|\vec{\beta}'\|}$$

und erhalten die Ebenengleichung in *Hesse Normalform*

$$\left\langle \vec{\beta}\,,\, \vec{x} \right\rangle - \beta_0 = 0 \tag{4.2}$$

wobei

$$\beta_0 = \left\langle \vec{\beta}\,,\, \vec{a} \right\rangle > 0$$

Dann ist $\beta_0$ der Abstand zum Ursprung.

**Beispiel Normalisierung**

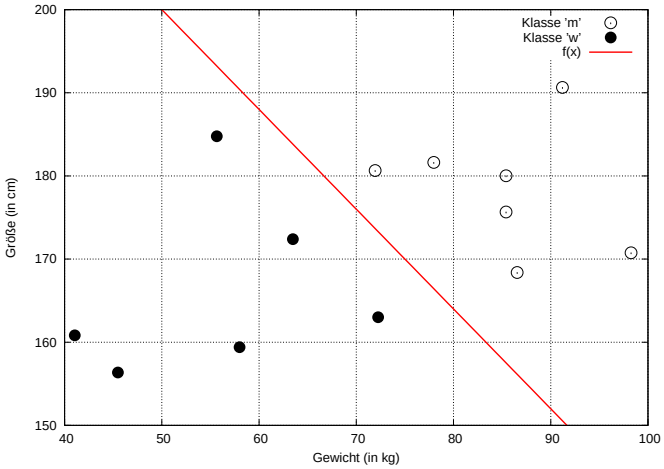
Sei $\vec{a} = \begin{pmatrix} 4 \\ 0 \\ 0 \end{pmatrix}$ und $\vec{\beta'} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ dann ist die Ebenengleichung nicht in Hesse Normalform, weil $\|\vec{\beta'}\| = \sqrt{14} \neq 1$. Wir normalisieren

$$\vec{\beta} = \frac{\vec{\beta'}}{\|\vec{\beta'}\|} \qquad \frac{1}{\sqrt{14}} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$\left\langle \vec{\beta}, \vec{x} \right\rangle - \beta_0 = 0 \quad \frac{1}{\sqrt{14}} x_1 + \frac{1}{\sqrt{14}} x_2 + \frac{1}{\sqrt{14}} x_3 - \frac{4}{\sqrt{14}} = 0$$

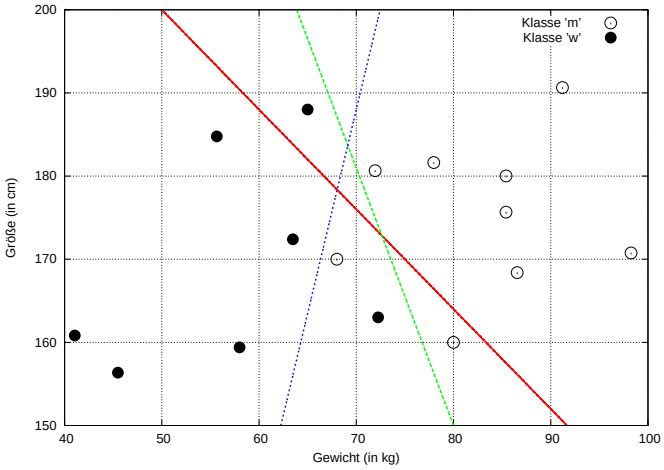
Jetzt ist $\beta_0 = \frac{-4}{\sqrt{14}}$ der Abstand der Ebene zum Ursprung.

**Beispiel: Ein mögliches $\vec{\beta}$**




$$f(\vec{x}) = \vec{x}^T \hat{\vec{\beta}} \quad \text{mit } \hat{\vec{\beta}} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 260 \\ 1 \\ 1.2 \end{pmatrix} \theta = 550$$

**Es ist nicht garantiert, dass $\vec{\beta}$ immer passt!**




**Modell-Anpassung**

Unsere linearen Modelle sind durch $\vec{\beta}$ parametrisiert, das Lernen eines Modells haben wir also auf die Wahl eines $\vec{\beta}$ abgewälzt.

Das wirft eine Reihe von Fragen auf:

- Was ist ein gutes $\vec{\beta}$?
- Gibt es ein optimales $\vec{\beta}$?
- Welche Möglichkeiten haben wir, unser Modell zu beurteilen?

Eine Möglichkeit: Berechne den *Trainingsfehler*

$$Err(\vec{\beta}) = \sum_{i=1}^{N} |y_i - \hat{f}(\vec{x}_i)| = \sum_{i=1}^{N} |y_i - x_i^T \vec{\beta}|$$

**Modell-Anpassung**

Häufig wird als Fehlerfunktion die *quadratische Fehlersumme* (RSS) verwendet:

$$\begin{aligned} RSS(\vec{\beta}) &= \sum_{i=1}^{N} (y_i - \vec{x_i}^T \vec{\beta})^2 \\ &= (\vec{y} - \mathbf{X}\vec{\beta})^T (\vec{y} - \mathbf{X}\vec{\beta}) \end{aligned}$$

Wir wählen jetzt $\vec{\beta}$ derart, dass der Fehler minimiert wird:

$$\min_{\vec{\beta} \in \mathbb{R}^p} RSS(\vec{\beta}) \tag{4.3}$$

$\Rightarrow$ Konvexes Minimierungsproblem!

**Minimierung von RSS($\vec{\beta}$)**

Um $RSS(\vec{\beta})$ zu minimieren, bilden wir die partielle Ableitung nach $\vec{\beta}$:

$$\frac{\partial RSS(\vec{\beta})}{\partial \beta} = \mathbf{X}^T(\mathbf{y} - \mathbf{X}\vec{\beta})$$

Notwendige Bedingung für die Existenz eines (lokalen) Minimums von $RSS$ ist

$$\frac{\partial RSS(\vec{\beta})}{\partial \beta} = \mathbf{X}^T(\mathbf{y} - \mathbf{X}\vec{\beta}) = 0$$

Ist $\mathbf{X}^T\mathbf{X}$ regulär, so erhalten wir

$$\hat{\vec{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{4.4}$$

**Reguläre Matrix**

Wenn es zu einer quadratischen Matrix $\mathbf{X}$ eine Matrix $\mathbf{X^{-1}}$ gibt mit

$$\mathbf{X}\mathbf{X^{-1}} = \mathbf{X^{-1}}\mathbf{X} = \mathbf{I}$$

Einheitsmatrix

$$I = \begin{pmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ . & . & ... & 0 \\ . & . & ... & 0 \\ 0 & 0 & ... & 1 \end{pmatrix}$$

dann ist die Matrix $\mathbf{X}$ invertierbar oder *regulär*, sonst *singulär*.

**Optimales $\hat{\vec{\beta}}$?**

Mit Hilfe der Minimierung der (quadratischen) Fehlerfunktion $RSS$ auf unseren Trainingsdaten haben wir ein (bzgl. RSS) optimales $\hat{\vec{\beta}}$ gefunden.

Bei einem konvexen Problem ist das lokale auch das globale Minimum.

Damit liefert unser Modell Voraussagen $\hat{y}$ für $\vec{x} \in X$:

$$\hat{y} = \hat{f}(\vec{x}) = x^T\hat{\vec{\beta}}$$

**Sind wir schon fertig?**

- Schön wär's!
- Aber drei Gründe sprechen für weitere Arbeit:
  1. Es ist nicht immer so einfach, z.B. dann nicht, wenn wir viele Dimensionen haben (Fluch der hohen Dimension).
  2. Vielleicht lassen sich die Beispiele nicht linear trennen!
  3. Nur den Fehler zu minimieren reicht nicht aus, wir suchen noch nach weiteren Beschränkungen, die zu besseren Lösungen führen.
- Also schauen wir uns den Fehler noch einmal genauer an, stoßen auf Bias und Varianz und merken, dass wir noch keine perfekte Lösung haben.

## 4.2 Bias-Varianz

**Fehler**

- Bisher haben wir mit RSS die Fehler einfach summiert.
- Wir wollen aber einbeziehen, wie wahrscheinlich der Fehler ist – vielleicht ist er ja ganz unwahrscheinlich! Das machen wir über den *Erwartungswert*.
- Wir können sehr unterschiedliche Stichproben als Beispielmenge haben. Der Fehler soll sich auf *alle möglichen Trainingsmengen* beziehen – nicht nur eine, zufällig günstige!

### 4.2.1 Exkurs:Erwartungswert

**Zur Erinnerung: Erwartungswert**

**Erwartungswert**

Sei $X$ eine *diskrete Zufallsvariable*, mit Werten $x_1, \ldots, x_n$ und $p_i$ die Wahrscheinlichkeit für $x_i$. Der Erwartungswert von $X$ ist

$$E(X) = \sum_i x_i p_i = \sum_i x_i P(X = x_i)$$

Ist $X$ eine *stetige Zufallsvariable* und $f$ die zugehörige Wahrscheinlichkeitsdichtefunktion, so ist der Erwartungswert von $X$

$$E(X) = \int_{-\infty}^{\infty} x\, f(x) dx$$

**Erwartungswert (Eigenschaften)**

**Eigenschaften**

Seien $X, Y$ und $X_1, \ldots, X_n$ Zufallsvariablen, dann gilt:

- Der Erwartungswert ist additiv, d.h. es gilt

$$E\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} E(X_i) \tag{4.5}$$

- Ist $Y = kX + d$, so gilt für den Erwartungswert

$$E(Y) = E(kX + d) = kE(X) + d \tag{4.6}$$

- Sind die Zufallsvariablen $X_i$ *stochastisch unabhängig*, gilt

$$E\left(\prod_{i=1}^{n} X_i\right) = \prod_{i=1}^{n} E(X_i)$$

### Varianz und Standardabweichung

Über den Erwartungswert einer Zufallsvariablen $X$ sind mehrere Eigenschaften von $X$ definiert, die helfen, $X$ zu charakterisieren:

**Varianz**

Sei $X$ eine Zufallsvariable mit $\mu = E(X)$. Die *Varianz* $Var(X)$ ist definiert als

$$Var(X) := E\left((X - \mu)^2\right).$$

Die Varianz wird häufig auch mit $\sigma^2$ bezeichnet.

**Standardabweichung**

Die *Standardabweichung* $\sigma$ einer Zufallsvariable $X$ ist definiert als

$$\sigma := \sqrt{Var(X)}$$

### Varianz und Standardabweichung

**Verschiebungssatz**

Sei $X$ eine Zufallsvariable, für die Varianz gilt

$$Var(X) = E(X - E(X))^2 = E(X^2) - (E(X))^2$$

### Bias

Eine weitere Charakteristik, die häufig zur Beschreibung von erwarteten Fehlern verwendet wird, ist die Verzerrung:

**Verzerrung (Bias)**

Sei $Y$ eine Zufallsvariable, dann ist die Verzerrung definiert als der erwartete Schätzfehler für $Y$ also wie im Durchschnitt die Schätzungen vom wahren Mittelwert abweichen

$$Bias(\hat{y}) = E(Y - \hat{y}) = E(Y) - \hat{y}$$

### Fehler der Regression

- Fehlerfunktion $L(y, \hat{y})$ für gelernte Modelle $\hat{f}$
  - absolut $\sum(y_i - \hat{y_i})$
  - quadratisch $\sum(y_i - \hat{y_i})^2$

  - 0,1-Fehler $\sum \delta_i$, $\delta = 1$, falls $y = \hat{y}$, sonst 0.

- Es geht um $Y$. Wir unterscheiden
  - das wahre $y$,
  - das in der Beispielmenge genannte $y$,
  - das vom Modell vorhergesagte $\hat{y}$
- Wir wollen den Erwartungswert des Fehlers minimieren.
- Wir mitteln über alle möglichen Beispielmengen $\mathcal{T}$.

**Erwartungswert des Fehlers einer Regression minimieren!**

**Erwarteter quadratischer Vorhersagefehler:** Gelernte Funktion $\hat{f} : X \to Y$, der Erwartungswert ihres Fehlers ist:

$$EPE(f) = E(Y - \hat{f}(X))^2 \tag{4.7}$$

**Optimierungsproblem:** Wähle $\hat{f}$ so, dass der erwartete Fehler minimiert wird!

$$\hat{f}(x) = argmin_c E_{Y|X}((Y - c)^2 | X = x) \tag{4.8}$$

**Lösung (Regressionsfunktion):** $\hat{f}(x) = E(Y|X = x)$

**Bias und Varianz**

Zwei Aspekte machen den erwarteten Fehler aus, die Verzerrung (Bias) und die Varianz. Wir wollen den Fehler an einem Testpunkt $x_0 = 0$ angeben und mitteln über allen Trainingsmengen $\mathcal{T}$.

- Wir gehen davon aus, dass die Angaben in den Daten nicht immer ganz stimmen, so dass es einen Messfehler $\epsilon$ gibt, dessen Erwartungswert aber 0 ist.
- Der Bias ist unabhängig vom Beispielsatz und 0 bei einem perfekten Lerner.
- Die Varianz ist unabhängig vom wahren Wert $y$ und 0 bei einem Lerner, der bei allen Beispielsätzen dasselbe ausgibt.

**MSE Dekomposition in Bias und Varianz**

Wir nehmen für unser Modell an,
dass $Y = f(x) + \epsilon$ und $E(\epsilon) = 0$.

$$\begin{aligned} EPE(x_0) = & \quad E_{Y,\mathcal{T}}((Y - \hat{y_0})^2 | x_0) & \\ = & \quad E_Y((Y - f(x_0))^2 | x_0) + & \sigma^2 Rauschen \\ & E_{\mathcal{T}}((f(x_0) - E_{\mathcal{T}}(\hat{y_0}))^2 | x_0) + & Bias^2 \\ & E_{\mathcal{T}}((E_{\mathcal{T}}(\hat{y_0}) - \hat{y_0})^2 | x_0) & Varianz \end{aligned}$$

Wie das?!

Haupttrick: kreatives Einfügen von Termen, $+a-a$, die nichts ändern, aber Umformungen erlauben.

**Herleitung der Dekomposition - 1**

$$\begin{aligned}
EPE(x_0) = & \quad E_{Y,\mathcal{T}}((Y-\hat{y_0})^2|x_0) & \\
= & \quad E_{Y,\mathcal{T}}(((Y-f(x_0))+(f(x_0)-\hat{y_0}))^2|x_0) & kreativ \\
= & \quad E_{Y,\mathcal{T}}((Y-f(x_0))^2+(f(x_0)-\hat{y_0})^2+ & binomisch \\
& \quad 2(Y-f(x_0))(f(x_0)-\hat{y_0})|x_0) & \\
= & \quad E_Y((Y-f(x_0))^2|x_0)+ & herausziehen \\
& \quad E_{Y,\mathcal{T}}((f(x_0)-\hat{y_0})^2|x_0)+ & s.Formel(4.5) \\
& \quad 2E_Y(Y-f(x_0)|x_0)E_{Y,\mathcal{T}}(f(x_0)-\hat{y_0}|x_0) & E(2)=2
\end{aligned}$$

Jetzt $E_Y(Y-f(x_0)|x_0) = E_Y(Y|x_0)-f(x_0) = 0$ wegen der Modellannahme $E_Y(Y|x_0) = f(x_0)$.

**Herleitung der Dekomposition - 2**

$$\begin{aligned}
EPE(x_0) = & \quad E_Y((Y-f(x_0))^2|x_0)+ & Var_\epsilon \\
& \quad E_{Y,\mathcal{T}}((f(x_0)-\hat{y_0})^2|x_0) & \\
= & \quad Var_\epsilon + E_{Y,\mathcal{T}}((f(x_0)-\hat{y_0})^2|x_0) & \\
= & \quad Var_\epsilon + E_{Y,\mathcal{T}}(((f(x_0)-E_{Y,\mathcal{T}}(\hat{y_0}))+ & kreativ \\
& \quad (E_{Y,\mathcal{T}}(\hat{y_0})-\hat{y_0}))^2|x_0) & \\
= & \quad Var_\epsilon + E_{Y,\mathcal{T}}((f(x_0)-E_{Y,\mathcal{T}}(\hat{y_0}))^2+ & binomisch \\
& \quad (E_{Y,\mathcal{T}}(\hat{y_0})-\hat{y_0})^2+ & \\
& \quad 2(f(x_0)-E_{Y,\mathcal{T}}(\hat{y_0}))(E_{Y,\mathcal{T}}(\hat{y_0})-\hat{y_0}|x_0)) & \\
= & \quad Var_\epsilon + E_{Y,\mathcal{T}}((f(x_0)-E_{Y,\mathcal{T}}(\hat{y_0}))^2|x_0)+ & herausziehen \\
& \quad E_{Y,\mathcal{T}}((E_{Y,\mathcal{T}}(\hat{y_0})-\hat{y_0})^2|x_0)+ & (4.5) \\
& \quad 2E_{Y,\mathcal{T}}((f(x_0)-E_{Y,\mathcal{T}}(\hat{y_0})) & \\
& \quad (E_{Y,\mathcal{T}}(\hat{y_0})-\hat{y_0})|x_0) &
\end{aligned}$$

**Herleitung der Dekomposition - 3**

Rauschen haben wir schon gefunden. Bias und Varianz auch!

$$\begin{aligned}
EPE(x_0) = & \quad Var_\epsilon+ & \sigma^2 Rauschen \\
& \quad E_{Y,\mathcal{T}}((f(x_0)-E_{Y,\mathcal{T}}(\hat{y_0}))^2|x_0)+ & Bias^2 \\
& \quad E_{Y,\mathcal{T}}((E_{Y,\mathcal{T}}(\hat{y_0})-\hat{y_0})^2|x_0)+ & Varianz \\
& \quad 2E_{Y,\mathcal{T}}(f(x_0)-E_{Y,\mathcal{T}}(\hat{y_0})) & Konstanten \\
& \quad (E_{Y,\mathcal{T}}(E_{Y,\mathcal{T}}\hat{y_0}-\hat{y_0}|x_0)) & =0 \\
= & \quad E_{Y,\mathcal{T}}((Y-\hat{y_0})^2|x_0) & q.e.d.
\end{aligned}$$

Wir betrachten den Fehler also als zusammengesetzt aus Rauschen, Verzerrung und Varianz. Die Dekomposition des MSE in Bias und Varianz abstrahiert so, dass wir besser über Modelle nachdenken können.

### 4.2.2 Bias und Varianz bei linearen Modellen

**Bias und Varianz bei linearen Modellen**

Das lineare Modell wird an die Daten angepasst durch

$$\hat{f}_p(\vec{x}) = \hat{\beta}^T \vec{x}$$

Der Fehler ist dann für ein beliebiges $\vec{x}$:

$$\begin{aligned} Err(\vec{x_0}) &= E[(Y - \hat{f}_p(\vec{x_0}))^2 | X = \vec{x_0}] && (4.9) \\ &= \sigma_\epsilon^2 + Var(\hat{f}_p(\vec{x_0})) + \left[ f(\vec{x_0}) - E\hat{f}_p(\vec{x_0}) \right]^2 && (4.10) \end{aligned}$$

Die Anpassung des linearen Modells geht über alle $N$ Beispiele und gewichtet alle $p$ Merkmale (s. (4.4)).

Diese Varianz ist von $x_i$ zu $x_i$ verschieden. Im Mittel über allen $\vec{x_i}$ ist $Var(\hat{f}_p) = (p/N)\sigma_\epsilon^2$.

**Zusammenhang zwischen Anzahl der Beispiele, der Attribute und erwartetem Fehler**

Modellkomplexität $(p, N)$ und Varianz der Schätzungen bei unterschiedlichen Trainingsmengen hängen bei linearen Modellen direkt zusammen.

Gemittelt über alle $x_i$ ist der Trainingsfehler linearer Modelle:

$$\frac{1}{N} \sum_{i=1}^{N} Err(x_i) = \sigma_\epsilon^2 + \frac{p}{N}\sigma_\epsilon^2 + \frac{1}{N} \sum_{i=1}^{N} \left[ f(\vec{x_i}) - E\hat{f}(\vec{x_i}) \right]^2 \quad (4.11)$$

Wir haben also wieder das Rauschen, die Varianz, die die Schwankungen der Schätzungen angibt, und den Bias, der sich auf die Differenz von Schätzung und Wahrheit bezieht (in-sample error).

**Fluch der hohen Dimension bei linearen Modellen**

- Leider mussten wir annehmen, dass das Modell genau passt, um den erwarteten Fehler klein zu halten.
- Wir wissen aber nicht, welche Art von Funktion gut zu unseren Daten passt! *Modellselektion* ist schwierig!
- Das Modell muss immer komplizierter werden, je mehr Dimensionen es gibt.
- Bei linearen Modellen entspricht die Komplexität des Modells direkt $p$, denn $\beta$ hat so viele Komponenten wie $p$ bzw. $p + 1$.

**Lineare Modelle**

Die grünen und roten Datenpunkte werden durch eine Ebene getrennt.

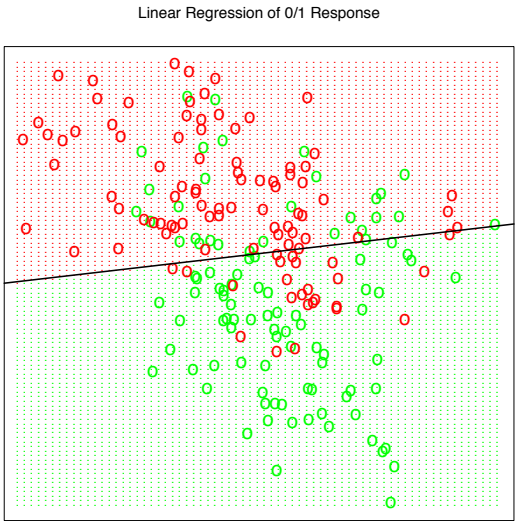



Figure 2.1: *A classification example in two dimensions. The classes are coded as a binary variable—* GREEN $= 0$, RED $= 1$*—and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$*. The red shaded region denotes that part of input space classified as* RED*, while the green region is classified as* GREEN*.*

**Was wissen Sie jetzt?**

- Sie haben theoretisch lineare Modelle für Klassifikation und Regression kennengelernt.
- Sie kennen das *Optimierungsproblem* der kleinsten Quadrate RSS (Gleichung 4.3) für lineare Modelle (Gleichung 4.4).
- Sie kennen den erwarteten Fehler EPE bei linearen Modellen (Gleichung 4.7).
- Sie kennen den *Fluch der hohen Dimension* bei linearen Modellen: Komplexität und Varianz hängen an der Dimension! Der Bias kann sehr hoch sein, wenn die Beispiele tatsächlich nicht linear separierbar sind.

**Bis zum nächsten Mal...**

- Gehen Sie alle Folien noch einmal in Ruhe durch.
- Vertiefen Sie sich noch einmal in die Ebenengleichung (4.2)! Die lineare Algebra wird immer wieder vorkommen. Sie können auch die partiellen Ableitungen für RSS mit der Normalengleichung vornehmen.

- Rechnen Sie mal ein Beispiel durch mit Gleichung zur Optimierung linearer Modelle (4.4), der Minimierung des Trainingsfehlers (4.11)...
- Diskutieren Sie, warum Bias und Varianz so wichtig sind!
- Probieren Sie lineare Regression in RapidMiner aus!

# Kapitel 5

# Support Vector Machines

## 5.1 Hinführungen zur SVM

**Übersicht über die Stützvektormethode (SVM)**
Eigenschaften der Stützvektormethode (SVM) (Support Vector Machine)

- Maximieren der Breite einer separierenden Hyperebene – maximum margin method – ergibt eindeutige, optimale trennende Hyperebene.
- Transformation des Datenraums durch Kernfunktion behandelt Nichtlinearität.
- Regularisierung minimiert nicht nur den Fehler, sondern auch die Komplexität des Modells.

**Einführende Literatur**

- Vladimir Vapnik "The Nature of Statistical Learning Theory" Springer Vg. 1995
- W.N. Wapnik, A. Tscherwonenkis "Theorie der Zeichenerkennung" Akademie Vg. 1979
- Christopher Burges "A Tutorial on Support Vector Machines for Pattern Recognition" in: Data Mining and Knowledge Discovery 2, 1998, 121-167

Vertiefung: Bernhard Schölkopf, Alexander Smola "Learning with Kernels", MIT Press, 2002

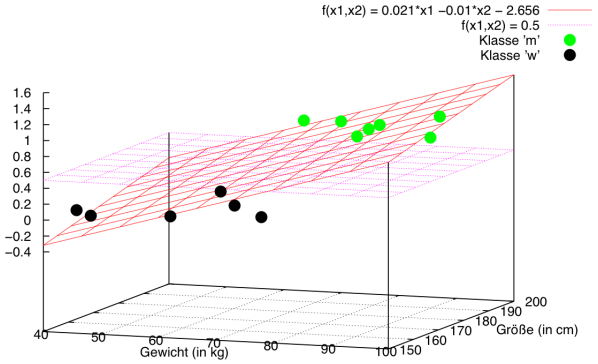
**Probleme der Empirischen Risikominimierung**
*Empirische Risikominimierung:* Bisher haben wir lineare Modelle

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$$

auf die Fehlerminimierung hin optimiert:

$$RSS(\hat{\vec{\beta}}) = \sum_{i=1}^{N} (y_i - \vec{x}_i^T \hat{\beta})^2$$

**Wo trennen wir die Daten?**




*Problem:* Mehrere Funktionen mit minimalem Fehler existieren. Welche wählen?

- *1. Ausweg:* Verbessertes Kriterium: *maximum margin*.
- *2. Ausweg:* Zusätzliches Kriterium: möglichst geringe Komplexität des Modells (*Regularisierung*)

**Klassifikationsproblem**

Gegeben sei ein Klassifikationsproblem mit $Y = \{-1; +1\}$ und $\mathbf{X} \subseteq \mathbb{R}^p$.
Sei $\mathbf{X} = C_+ \dot\cup C_-$ die Menge der Trainingsbeispiele mit

$$C_+ = \{(\vec{x}, y) \mid y = +1\} \quad \text{und} \quad C_- = \{(\vec{x}, y) \mid y = -1\}$$

Zur Klassifikation ist nun eine Hyperebene

$$H = \left\{ \vec{x} \mid \beta_0 + \left\langle \vec{x}\,,\, \vec{\beta} \right\rangle = 0 \right\}$$

gesucht, die die Mengen $C_+$ und $C_-$ *bestmöglichst* trennt

Für eine gegebene Hyperebene $H$ erfolgt die Klassifikation dann durch

$$\hat{y} = \text{sign}\left(\beta_0 + \left\langle \vec{x}\,,\, \vec{\beta} \right\rangle\right)$$

**Notationen...**

**Und warum jetzt $\left\langle \vec{x}\,,\, \vec{\beta} \right\rangle$ statt $\vec{x}^T\vec{\beta}$?**

Wir bewegen uns derzeit in einem $\mathbb{R}$-Vektorraum der Beispiele mit dem Standardskalarprodukt

$$\left\langle \vec{x}\,,\, \vec{\beta} \right\rangle = \underbrace{\vec{x}^T\vec{\beta}}_{Matrixmultiplikation} = \underbrace{\vec{x}\vec{\beta}}_{ImplizitesSkalarprodukt}$$

**Und warum jetzt $\beta_0 + \left\langle \vec{x}\,,\, \vec{\beta} \right\rangle$ statt $\left\langle \vec{x}\,,\, \vec{\beta} \right\rangle - \beta_0$?** Warum nicht? Vorher $\beta_0 = \left\langle \vec{\beta}\,,\, \vec{a} \right\rangle > 0$, es geht auch $\beta_0 < 0$.

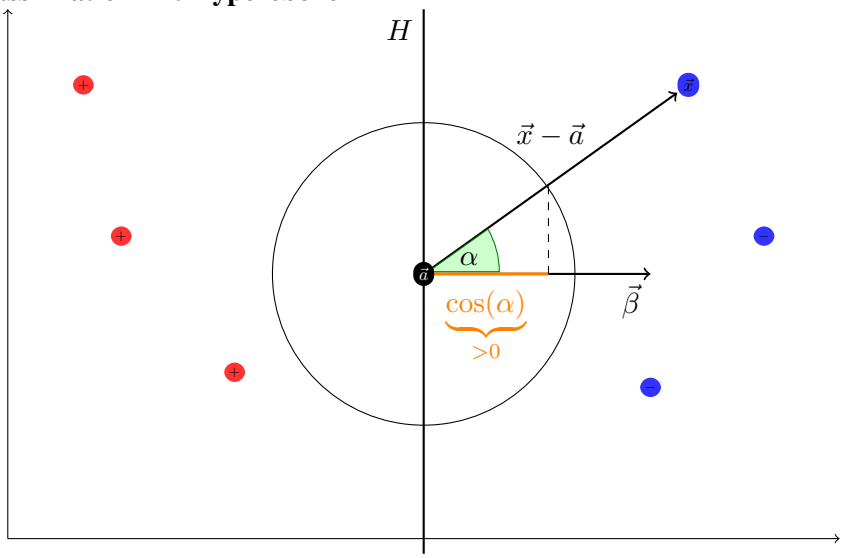
**Klassifikation mit Hyperebenen**

Die vorzeichenbehaftete Distanz eines Punktes $\vec{x}$ zu einer Hyperebene $H$ mit dem Stützvektor $\vec{a}$ und Normalenvektor $\vec{\beta}$ ist

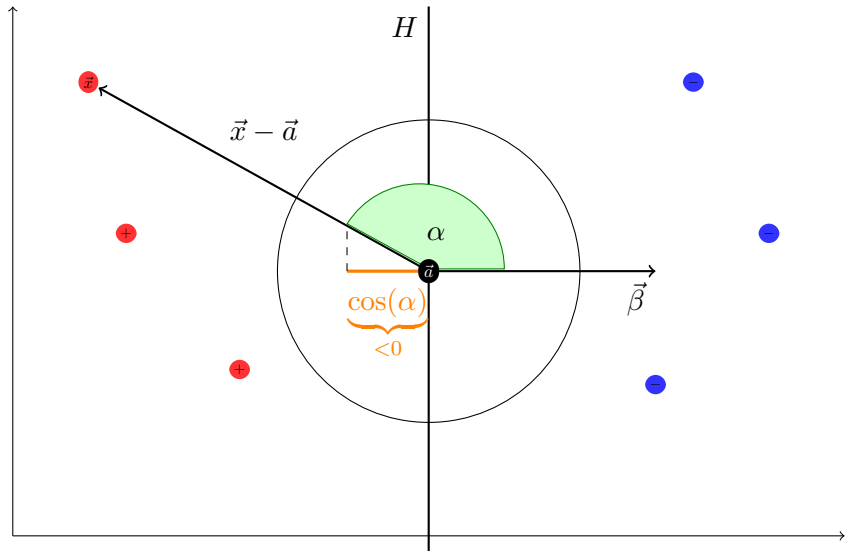
$$
\begin{aligned}
d(\vec{x}, H) &= \left\langle \vec{x}\,,\, \vec{\beta} \right\rangle - \beta_0 && (5.1)\\
&= \left\langle \vec{x}\,,\, \vec{\beta} \right\rangle - \left\langle \vec{a}\,,\, \vec{\beta} \right\rangle && (5.2)\\
&= \left\langle \vec{x} - \vec{a}\,,\, \vec{\beta} \right\rangle && (5.3)\\
&= \underbrace{\|\vec{x} - \vec{a}\| \cdot \|\vec{\beta}\|}_{>0} \cdot \cos(\measuredangle(\vec{x} - \vec{a}, \vec{\beta})) && (5.4)
\end{aligned}
$$

Nur $\cos(\measuredangle(\vec{x} - \vec{a}, \vec{\beta}))$ kann negativ werden und bestimmt die Klassifizierung.

**Klassifikation mit Hyperebenen**




**Klassifikation mit Hyperebenen**




**Klassifikation mit Hyperebenen**

Die vorzeichenbehaftete Distanz $d(\vec{x}, H)$ drückt aus

1. den Abstand $|d(\vec{x}, H)|$ von $\vec{x}$ zu Ebene $H$
2. die Lage von $\vec{x}$ relativ zur Orientierung ($\vec{\beta}$) von $H$, d.h.

$$\text{sign}\left(d(\vec{x}, H)\right) = \begin{cases} +1 & d(\vec{x}, H) > 0, \cos \measuredangle(\vec{x}, \vec{\beta}) > 0 \\ -1 & d(\vec{x}, H) < 0, \cos \measuredangle(\vec{x}, \vec{\beta}) < 0 \end{cases}$$

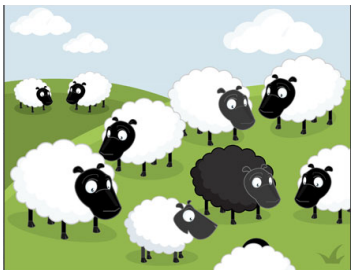
Auf diese Weise lassen sich die Punkte klassifizieren mit

$$\hat{y} = \text{sign}\left(\beta_0 + \left\langle \vec{x}\,,\, \vec{\beta} \right\rangle\right)$$

Bei $y = -1$ liegen die Punkte $\vec{x_i}$ im Halbraum des Ursprungs.

**Einführung von Schölkopf/Smola**

Gegeben eine Menge von Schafen, packe immer die ähnlichen zusammen! Vorgehen: Schafe vergleichen!



**Einfacher Ansatz nach Schölkopf/Smola**

Ein einfacher Ansatz zu einer separierenden Hyperebene zu kommen, geht über die Zentroiden von $C_+$ und $C_-$:

Seien

$$\vec{c}_+ := \frac{1}{|C_+|} \sum_{(\vec{x},y)\in C_+} \vec{x} \quad \text{und} \quad \vec{c}_- := \frac{1}{|C_-|} \sum_{(\vec{x},y)\in C_-} \vec{x}$$

Wähle nun

$$\vec{a} := \frac{\vec{c}_+ + \vec{c}_-}{2} \quad \text{und} \quad \vec{\beta} := \vec{c}_+ - \vec{c}_-$$

als Hyperebene mit Normalenvektor $\vec{\beta}$ durch den Punkt $\vec{x}_0$

**Separierende Hyperebene über Zentroiden**

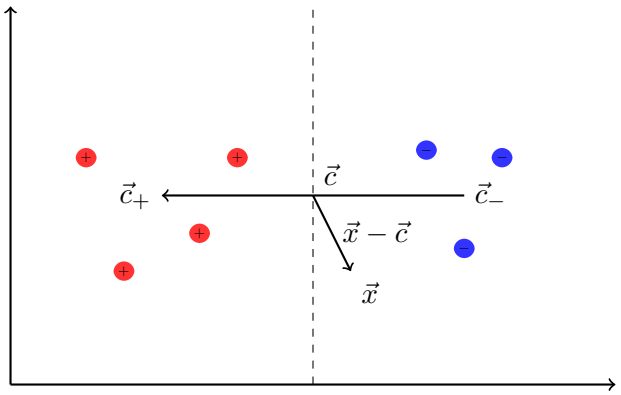
Durch $\vec{\beta}$ und $\vec{a}$ ist die Hyperebene gegeben als

$$\tilde{H} = \left\{\vec{x} \mid \left\langle \vec{x} - \vec{a}\,,\, \vec{\beta} \right\rangle = 0\right\} = \left\{\vec{x} \mid \left\langle \vec{x}\,,\, \vec{\beta} \right\rangle - \underbrace{\left\langle \vec{a}\,,\, \vec{\beta} \right\rangle}_{=:-\beta_0} = 0\right\}$$

Damit erfolgt die Klassifikation durch

$$\begin{aligned} \hat{y} &= \operatorname{sign}\left(\left\langle \vec{x} - \vec{c}\,,\, \vec{\beta} \right\rangle\right) \\ &= \operatorname{sign}\left(\langle \vec{x}\,,\, \vec{c}_+ \rangle - \langle \vec{x}\,,\, \vec{c}_- \rangle + \beta_0\right) \end{aligned}$$

**Lernalgorithmus im Bild**

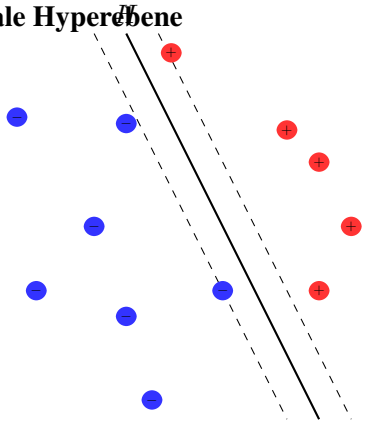



**Fast...**

... wäre das schon die Stützvektormethode. Aber:

- Einfach den Mittelpunkt der Beispiele einer Klasse zu berechnen ist zu einfach, um ein ordentliches $\vec{\beta}$ zu bekommen.
- Man erhält so nicht die optimale Hyperebene.

**Die optimale Hyperebene**




Eine Menge von Beispielen heißt *linear trennbar*, falls es eine Hyperebene $H$ gibt, die die positiven und negativen Beispiele trennt.

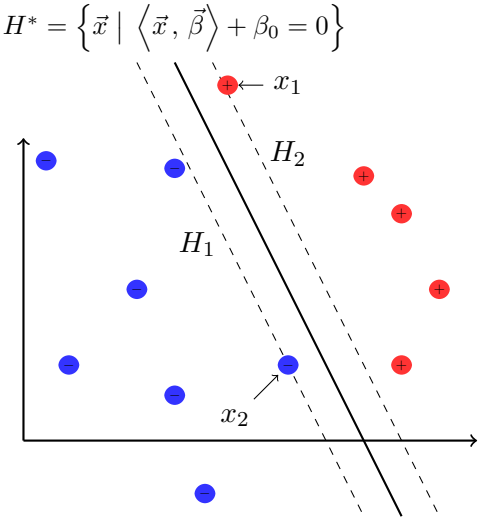
**5.1: Optimale Hyperebene**

Eine separierende Hyperebene $H$ heißt *optimal*, wenn ihr minimaler Abstand $d$ zu allen Beispielen maximal ist.

**5.2: Satz (Eindeutigkeit)**

Es existiert eine eindeutig bestimmte optimale Hyperebene.

**Bild**




Nach 5.1 wird die optimale Hyperebene durch die nächstliegenden Punkte aus $C_+$ und $C_-$ bestimmt.
Skalierung von $\vec{\beta}$ und $\beta_0$, so dass für die nächstliegenden Punkte $x_i$ zu $H^*$ gilt:

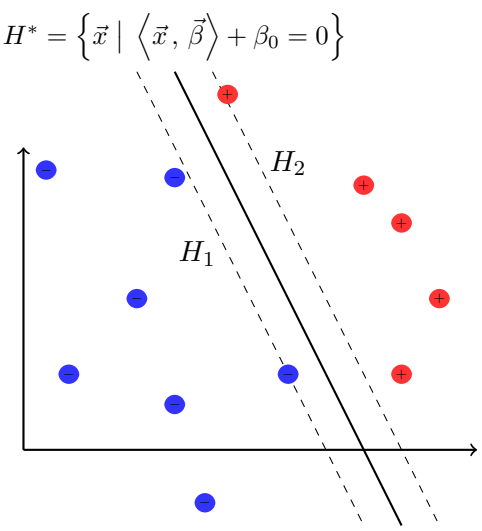
$$|\left\langle \vec{\beta}, \vec{x}_i \right\rangle + \beta_0| = 1$$

Die Beispiele am nächsten zur Hyperebene liefern die beiden Hyperebenen $H_1$ und $H_2$

$$H_j = \left\{\vec{x} \mid \left\langle \vec{x}, \vec{\beta} \right\rangle + \beta_0 = (-1)^j\right\}$$

## 5.2 Maximum Margin Methode

**Abstand der Hyperebenen zum Ursprung**



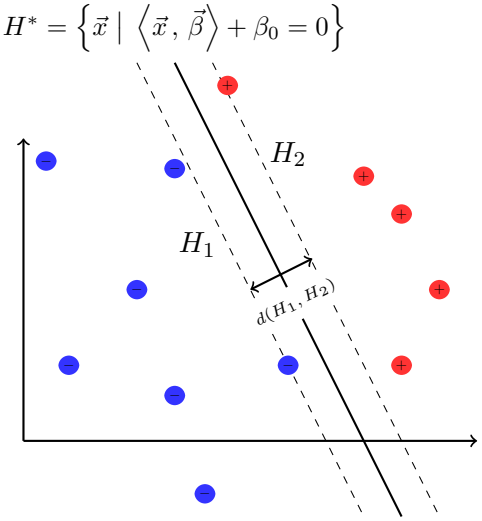

Der Abstand der mittleren Ebene $H^*$ zum Ursprung beträgt

$$d(\vec{0}, H^*) = \frac{\beta_0}{||\vec{\beta}||}$$

Der Abstand zwischen den Ebenen $H_1$ und $H_2$ ist

$$\begin{aligned} d(H_1, H_2) &= \frac{\beta_0+1}{||\vec{\beta}||} - \frac{\beta_0-1}{||\vec{\beta}||} \\ &= \frac{\beta_0-\beta_0+1+1}{||\vec{\beta}||} \\ &= \frac{2}{||\vec{\beta}||} \end{aligned}$$

**Margin**




Nach Konstruktion liegt kein Beispiel zwischen $H_1$ und $H_2$, d.h.

$$\langle \vec{x}, \vec{\beta} \rangle + \beta_0 \geq +1 \; \forall \vec{x} \in C_+ \quad (5.5)$$
$$\langle \vec{x}, \vec{\beta} \rangle + \beta_0 \leq -1 \; \forall \vec{x} \in C_- \quad (5.6)$$

Der Abstand

$$d(H_1, H_2) = \frac{2}{||\vec{\beta}||}$$

heißt *Margin* und soll maximiert werden!

**Maximimum Margin**

Mit der Maximierung des Margin finden wir eine *optimale Hyperebene* innerhalb der Menge der möglichen trennenden Hyperebenen.

Konvexes, quadratisches Optimierungsproblem:

- Es existiert eine eindeutig bestimmte, optimale Hyperebene

$$H^* = \left\{\vec{x} \mid \left\langle \vec{x}, \vec{\beta} \right\rangle + \beta_0 = 0\right\}$$

- unter der Bedingung, dass $\frac{1}{2}||\vec{\beta}||^2$ minimal ist.

Das Optimierungsproblem läßt sich in Zeit $O(N^3)$ lösen.

**Optimierungsaufgabe**

Nach diesen Vorüberlegungen haben wir also (nur noch) die folgende Optimierungsaufgabe zu lösen:

**Optimierungsaufgabe**

Minimiere

$$\frac{1}{2}||\vec{\beta}||^2$$

unter den Nebenbedingungen

$$\left\langle \vec{x}\,,\,\vec{\beta} \right\rangle + \beta_0 \geq +1 \quad \forall\, \vec{x} \in C_+$$
$$\left\langle \vec{x}\,,\,\vec{\beta} \right\rangle + \beta_0 \leq -1 \quad \forall\, \vec{x} \in C_-$$

Die Nebenbedingungen lassen sich zusammenfassen zu

$$y(\left\langle \vec{x}\,,\,\vec{\beta} \right\rangle + \beta_0) - 1 \geq 0 \quad \forall (\vec{x}, y) \in \mathbf{X} \tag{5.7}$$

**Optimierung mit Nebenbedingungen**

Sei die optimierende Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ gegeben als

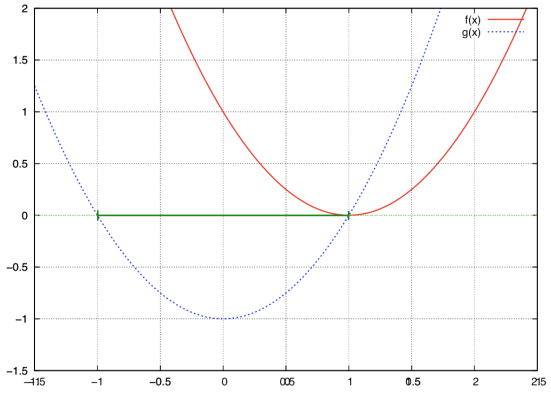
$$f(x) = (x-1)^2$$

unter der einzigen Nebenbedingung

$$g(x) = x^2 - 1,$$

d.h. für die möglichen Lösungen $\tilde{x}$ muss gelten

$$\tilde{x} \in \{x \in \mathbb{R} \mid g(x) \leq 0\}$$



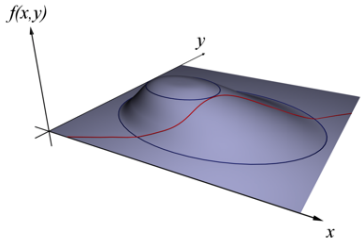

### 5.2.1 Lagrange-Optimierung

**Beispiel Lagrange Multiplikatoren zur Optimierung**

Gegeben: Funktion $f(x, y)$, Nebenbedingung $g(x, y) = c$, Optimierungsziel: maximiere $c$.
Notwendige Bedingung: $f(x, y) = c$ und $g(x, y) = c$.
Lagrangefunktion

$$L(x, y, \lambda) = f(x, y) + \lambda(g(x, y) - c)))$$

http://de.wikipedia.org/wiki/Lagrange-Multiplikator




**Optimierung mit Lagrange**

Die Optimierung nach Lagrange formuliert die Optimierung einer Funktion $f(x)$ unter Nebenbedingungen um in eine Optimierung ohne Nebenbedingungen.

Mit der Lagrange-Methode lassen sich Nebenbedingungen $g_i$ und $h_j$ der Art

$$g_i(x) \leq 0 \quad \text{und} \quad h_j(x) = 0$$

in die zu optimierende Funktion $f$ hinzufügen, im Falle eines Minimierungsproblems als

$$\min f(x) + \sum_i \alpha_i g_i(x) + \sum_j \mu_j h_j(x) \quad \text{mit } \alpha_i, \mu_j \geq 0 \; \forall \, i, j$$

Die $\alpha_i$ und $\mu_j$ heißen auch *Lagrange-Multiplikatoren*.

**Lagrange-Funktion**

Die Umformung der Nebenbedingungen (5.7) erlaubt nun die Anwendung von Lagrange (nur Ungleichheitsbedingungen):

**Lagrange-Funktion**

Sei das Optimierungsproblem gegeben, $f(\vec{\beta})$ zu minimieren unter den Nebenbedingungen $g_i(\vec{\beta}) \geq 0, i = 1, ..., m$ dann ist die Lagrange-Funktion:

$$L\left(\vec{\beta}, \vec{\alpha}\right) = f(\vec{\beta}) - \sum_{i=1}^{m} \alpha_i g_i(\vec{\beta}) \tag{5.8}$$

Dabei muss gelten $\alpha_i \geq 0$, Gleichheitsbedingungen sind nicht gegeben.

**SVM Optimierungsfunktion als Lagrange**

Die Nebenbedingungen $g_i$ sind gegeben durch

$$g_i(\vec{\beta}, \beta_0) = y_i \left( \left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle + \beta_0 \right) - 1 \geq 0 \; \forall \, \vec{x_i} \in \mathbf{X}$$

Die Formulierung des Optimierungsproblems nach Lagrange wird auch als *Primales Problem* bezeichnet:

**Primales Problem**

Die Funktion

$$L_P(\vec{\beta}, \beta_0, \vec{\alpha}) = \frac{1}{2}\|\vec{\beta}\|^2 - \sum_{i=1}^{N} \alpha_i \left( y_i \left( \left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle + \beta_0 \right) - 1 \right) \tag{5.9}$$

soll $L_P$ bezüglich $\vec{\beta}$ und $\beta_0$ *minimiert* und bezüglich $\vec{\alpha}$ *maximiert* werden!

**Karush-Kuhn-Tucker Bedingungen**

Durch die partiellen Ableitung nach $\vec{\beta}$ und $\beta_0$ erhalten wir

$$\frac{\partial}{\partial \vec{\beta}} L_P(\vec{\beta}, \beta_0, \vec{\alpha}) = \vec{\beta} - \sum_i \alpha_i y_i \vec{x}_i \quad \text{und} \quad \frac{\partial}{\partial \beta_0} L_P(\vec{\beta}, \beta_0, \vec{\alpha}) = -\sum_i \alpha_i y_i$$

Nullsetzen der Ableitungen und die Berücksichtigung der Nebenbedingungen führt zu den KKT-Bedingungen für eine Lösung für $L_P$:

$$\vec{\beta} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i \quad \text{und} \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{5.10}$$

$$\alpha_i \geq 0 \; \forall \, i = 1, \ldots, N \tag{5.11}$$

$$\alpha_i \left( y_i \left( \left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle + \beta_0 \right) - 1 \right) = 0 \; \forall \, i = 1, \ldots, N \tag{5.12}$$

**Duales Problem**

Das primale Problem soll bezüglich $\vec{\beta}$ und $\beta_0$ minimiert und bezüglich $\vec{\alpha}$ maximiert werden:
Mit den Bedingungen aus $\frac{\partial L_P}{\partial \vec{\beta}}$ und $\frac{\partial L_P}{\partial \beta_0}$ erhalten wir den *dualen Lagrange-Ausdruck* $L_D(\vec{\alpha})$

- Der duale Lagrange-Ausdruck $L(\vec{\alpha})$ soll maximiert werden.
- Das Minimum des ursprünglichen Optimierungsproblems tritt genau bei jenen Werten von $\vec{\beta}$,$\beta_0$,$\vec{\alpha}$ auf wie das Maximum des dualen Problems.

**Umformung des primalen in das duale Problem**

$$\begin{aligned}
& \frac{1}{2}||\vec{\beta}||^2 - \sum_{i=1}^{N} \alpha_i && \left[ y_i \left( \left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle + \beta_0 \right) - 1 \right] && \\
=& \frac{1}{2}||\vec{\beta}||^2 - \sum_{i=1}^{N} \alpha_i && y_i \left( \left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle + \beta_0 \right) && + \sum_{i=1}^{N} \alpha_i \\
=& \frac{1}{2}||\vec{\beta}||^2 - \sum_{i=1}^{N} \alpha_i y_i \left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle && - \sum_{i=1}^{N} \alpha_i y_i \beta_0 && + \sum_{i=1}^{N} \alpha_i \\
\overset{(5.10)}{=}& \frac{1}{2}||\vec{\beta}||^2 - \sum_{i=1}^{N} \alpha_i y_i \left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle && && + \sum_{i=1}^{N} \alpha_i
\end{aligned}$$

**Umformung II**

Einsetzen von $\vec{\beta} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i$ führt zu

$$\begin{aligned}
&\frac{1}{2}||\vec{\beta}||^2 && - \sum_{i=1}^{N} \alpha_i y_i \left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle && + \sum_{i=1}^{N} \alpha_i \\
=&\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \left\langle \vec{x}_i \, , \, \vec{x}_j \right\rangle && - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \left\langle \vec{x}_i \, , \, \vec{x}_j \right\rangle && + \sum_{i=1}^{N} \alpha_i \\
=&+ \sum_{i=1}^{N} \alpha_i && - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \left\langle \vec{x}_i \, , \, \vec{x}_j \right\rangle
\end{aligned}$$

unter den Nebenbedingungen $0 = \sum_{i=1}^{N} \alpha_i y_i$ und $\alpha_i \geq 0 \; \forall i$

**SVM Optimierungsproblem (Duales Problem)**

Die Umformungen führen nach Einsetzen der KKT-Bedingungen zum *dualen Problem*:

**Duales Problem**

Maximiere

$$L_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i \, , \, \vec{x}_j \right\rangle \tag{5.13}$$

unter den Bedingungen

$$\alpha_i \geq 0 \; \forall \, i = 1, \ldots, N \quad \text{und} \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$

**Stützvektoren**

Die Lösung $\vec{\alpha}^*$ des dualen Problems

$$L_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i \, , \, \vec{x}_j \right\rangle$$

muss die KKT-Bedingungen erfüllen, d.h. es gilt unter anderem

$$\alpha_i \left( y_i \left( \left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle + \beta_0 \right) - 1 \right) = 0 \; \forall \, i = 1, \ldots, N$$

$\vec{\alpha}^*$ enthält für jedes Beispiel $\vec{x}_i$ genau ein $\alpha_i$ mit

$$\begin{aligned}
\alpha_i = 0 \quad &, \quad \text{falls } \vec{x}_i \text{ im richtigen Halbraum liegt} \\
\alpha_i > 0 \quad &, \quad \text{falls } \vec{x}_i \text{ auf der Hyperebene } H_1 \text{ oder } H_2 \text{ liegt}
\end{aligned}$$

Ein Beispiel $\vec{x}_i$ mit $\alpha_i > 0$ heißt Stützvektor.

**Optimale Hyperebene**

Haben wir das optimale $\vec{\alpha}^*$ bestimmt, erhalten wir unsere optimale Hyperebene:
Nach (5.10) gilt

$$\vec{\beta} = \sum \alpha_i y_i \vec{x}_i$$

d.h. der optimale Normalenvektor $\vec{\beta}$ ist eine Linearkombination von Stützvektoren.

Um $\beta_0$ zu bestimmen können wir

$$\alpha_i \left( y_i \left( \left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle + \beta_0 \right) - 1 \right) = 0$$

für ein beliebiges $i$ und unser berechnetes $\vec{\beta}$ nutzen.

**Berechnung der $\alpha_i$?**

Das prinzipielle Vorgehen ist bei der SVM wie bei anderen Lernverfahren auch:

- Parametrisierung der Modelle, hier über Umwege durch $\vec{\alpha}$
- Festlegung eines Optimalitätskriteriums, hier: *Maximum Margin*
- Formulierung als Optimierungsproblem

Das finale Optimierungsproblem läßt sich mit unterschiedlichen Ansätzen lösen

- Numerische Verfahren (*quadratic problem solver*)
- *Sequential Minimal Optimization* (SMO, [J. C. Platt, 1998])
- Evolutionäre Algorithmen (EvoSVM, [I. Mierswa, 2006])

**Zusammenfassung der Lagrange-Optimierung für SVM**

Das Lagrange-Optimierungs-Problem (5.9) ist definiert als:

$$L_P = \frac{1}{2}\|\vec{\beta}\|^2 - \sum_{i=1}^{N} \alpha_i \left[ y_i (\left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle + \beta_0) - 1 \right]$$

mit den *Lagrange-Multiplikatoren* $\vec{\alpha}_i \geq 0$.

Notwendige Bedingung für ein Minimum liefern die Ableitungen nach $\vec{\beta}$ und $\beta_0$

$$\frac{\partial L_P}{\partial \vec{\beta}} = \vec{\beta} - \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i \quad \text{und} \quad \frac{\partial L_P}{\partial \beta_0} = \sum_{i=1}^{N} \alpha_i y_i$$

Diese führen zum *dualen Problem* (5.13)

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} \left\langle \vec{x}_i \, , \, \vec{x}_{i'} \right\rangle$$

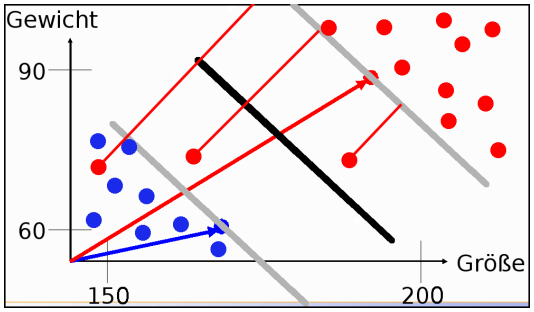
**Was wissen wir jetzt?**

- Maximieren des Margins einer Hyperebene ergibt eine eindeutige Festlegung der optimalen trennenden Hyperebene.
- Dazu minimieren wir die Länge des Normalenvektors $\vec{\beta}$
  - Formulierung als Lagrange-Funktion
  - Formulierung als duales Optimierungsproblem
- Das Lernergebnis ist eine Linearkombination von Stützvektoren.
- Mit den Beispielen müssen wir nur noch das Skalarprodukt rechnen.

## 5.3 Weich trennende SVM

**SVM mit Ausnahmen**

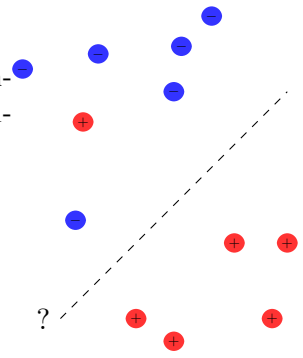
- Was passiert, wenn die Beispiele nicht komplett trennbar sind?




**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.




**SVM mit Ausnahmen**

Ein anderer Ansatz basiert wieder auf einer Relaxation:

- Punkte, die nicht am Rand oder auf der richtigen Seite der Ebene liegen, bekommen einen Strafterm $\xi_j > 0$.

- Korrekt klassifizierte Punkte erhalten eine Variable $\xi_j = 0$.

Dies führt zu folgenden Minimierungsproblem

$$\frac{1}{2}\|\vec{\beta}\|^2 + C\sum_{j=1}^{N}\xi_j \quad \text{für ein festes } C \in \mathbb{R}_{>0} \tag{5.14}$$

Daraus folgt insbesondere

$$0 \leq \alpha_i \leq C$$

**Weich trennende Hyperebene**

**Relaxiertes Optimierungsproblem**

Sei $C \in \mathbb{R}$ mit $C > 0$ fest. Minimiere

$$||\vec{\beta}||^2 + C\sum_{i=1}^{N}\xi_i$$

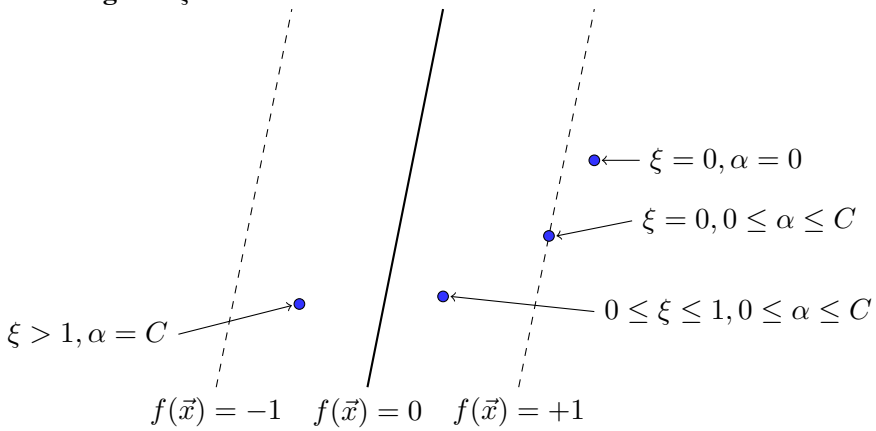
unter den Nebenbedingungen

$$\begin{aligned} \left\langle \vec{x}_i\,,\,\vec{\beta}\right\rangle + \beta_0 &\geq +1 - \xi_i \quad \text{für } \vec{y}_i = +1 \\ \left\langle \vec{x}_i\,,\,\vec{\beta}\right\rangle + \beta_0 &\leq -1 + \xi_i \quad \text{für } \vec{y}_i = -1 \end{aligned}$$

Durch Umformung erhalten wir wieder Bedingungen für die Lagrange-Optimierung:

$$y_i(\left\langle \vec{x}_i\,,\,\vec{\beta}\right\rangle + \beta_0) \geq 1 - \xi_i \quad \forall\, i = 1, \ldots, N$$

**Bedeutung von $\xi$ und $\vec{\alpha}$**



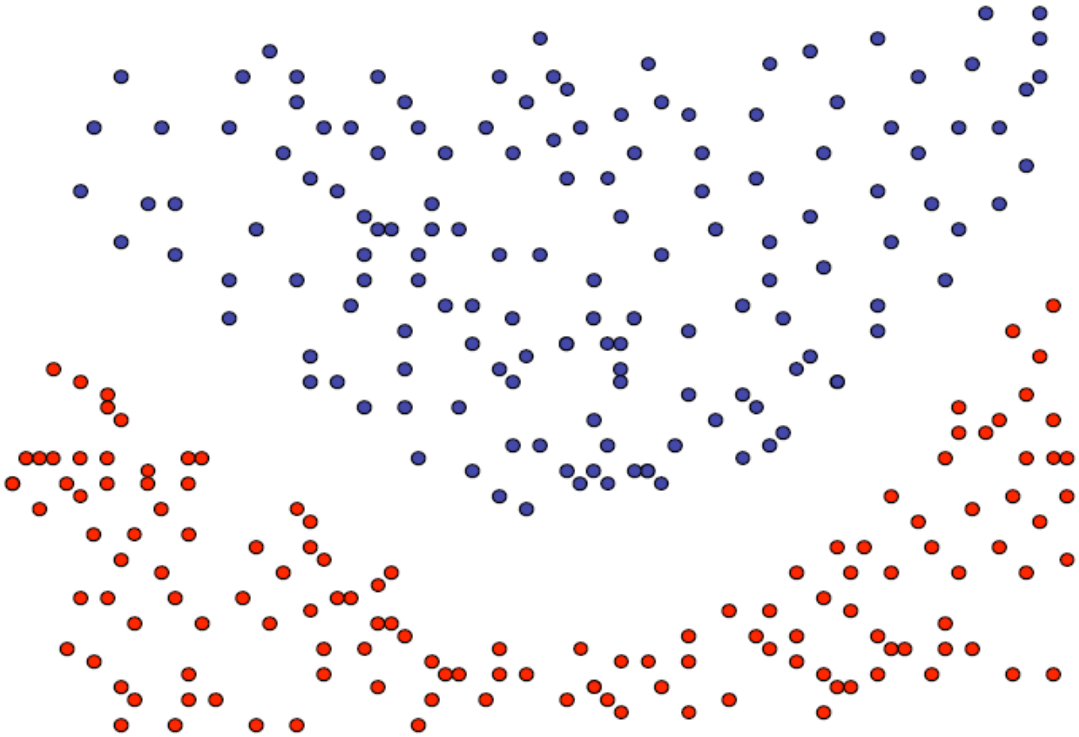

Beispiele $\vec{x}_i$ mit $\alpha_i > 0$ sind Stützvektoren.

**Wo sind wir?**

- Maximieren der Breite einer separierenden Hyperebene (*maximum margin method*) ergibt eindeutige, optimale trennende Hyperebene.
- Transformation des Datenraums durch Kernfunktion behandelt Nichtlinearität.
  - Das kam nur einmal am Rande vor. Wir sehen es nachher genauer.
- Regularisierung minimiert nicht nur den Fehler, sondern auch die Komplexität des Modells.
  - Später!

## 5.4 Kernfunktionen

**Nicht-lineare Daten**

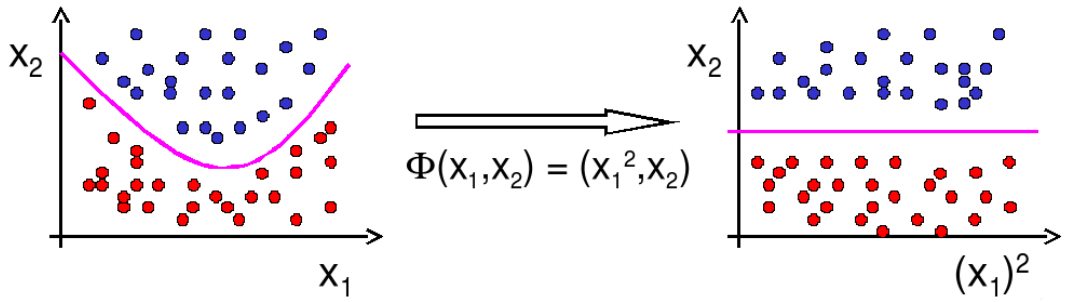


**Nicht-lineare Daten**

- 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!



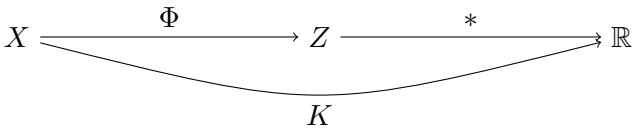

**Kernfunktionen**

- Erinnerung:

$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \vec{x_i}\,,\, \vec{x_j} \rangle$$

$$f(\vec{x}) = \sum \alpha_i y_i \langle \vec{x_i}\,,\, \vec{x} \rangle + \beta_0$$

- SVM hängt von $\vec{x}$ nur über Skalarprodukt $\left\langle \vec{x}\,,\, \vec{x'} \right\rangle$ ab.
- Ersetze Transformation $\Phi$ und Skalarprodukt durch Kernfunktion $K(\vec{x_1}, \vec{x_2}) = \langle \Phi(\vec{x_1})\,,\, \Phi(\vec{x_2}) \rangle$



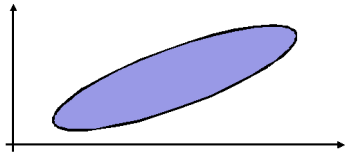

**Kernfunktionen II**

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

**Polynom-Kernfunktionen**

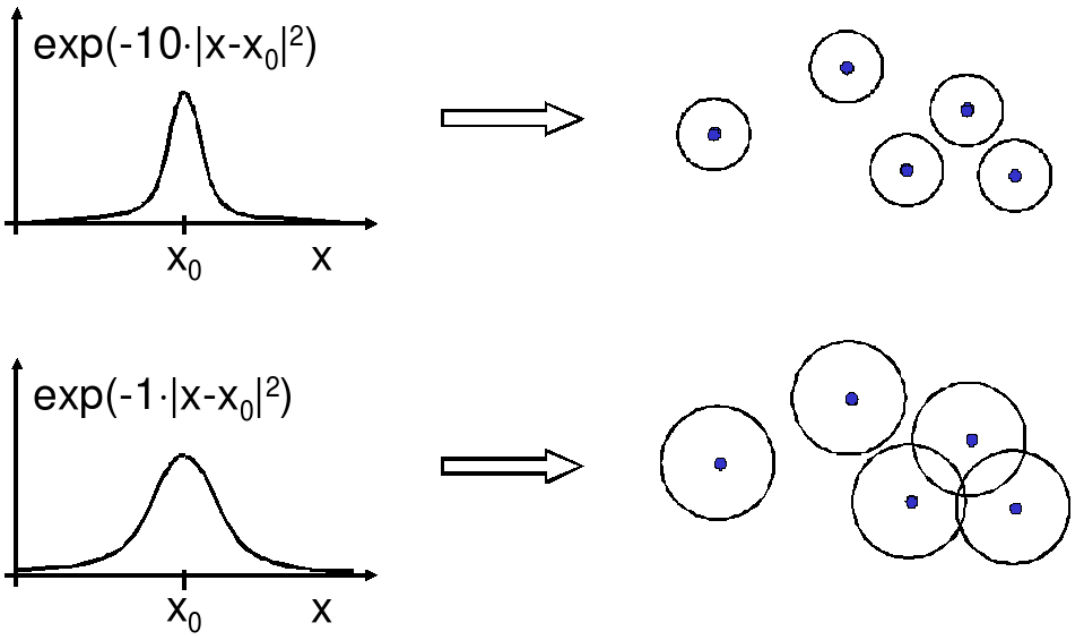
- $K_d(\vec{x_i}, \vec{x_j}) = \langle \vec{x_i}\,,\, \vec{x_j} \rangle^d$

- Beispiel: $d = 2, \vec{x_i}, \vec{x_j} \in \mathbb{R}^2$.



$$
\begin{aligned}
K_2(\vec{x_i}, \vec{x_j}) &= \langle \vec{x_i}\,,\, \vec{x_j} \rangle^2 \\
&= ((x_{i_1}, x_{i_2}) * (x_{j_1}, x_{j_2}))^2 = (x_{i_1} x_{j_1} + x_{i_2} x_{j_2})^2 \\
&= x_{i_1}^2 x_{j_1}^2 + 2 x_{i_1} x_{j_1} x_{i_2} x_{j_2} + x_{i_2}^2 x_{j_2}^2 \\
&= (x_{i_1}^2, \sqrt{2} x_{i_1} x_{i_2}, x_{i_2}^2) * (x_{j_1}^2, \sqrt{2} x_{j_1} x_{j_2}, x_{j_2}^2) \\
&=: \langle \phi(\vec{x_i})\,,\, \phi(\vec{x_j}) \rangle
\end{aligned}
$$

**RBF-Kernfunktion**




**Kernfunktionen**

- Die Kernfunktionen werden nicht als Vorverarbeitungsschritt durchgeführt.
- Man muss lediglich bei der Berechnung des Skalarprodukts die Kernfunktion berücksichtigen.
- Allerdings kann $\vec{\beta}$ jetzt nicht mehr so einfach interpretiert werden als Bedeutung der Variablen (Merkmale) $X_i$.

**Interpretation der SVM**

- Wenn das Skalarprodukt als Kernfunktion gewählt wird, entspricht jede Komponente des $\vec{\beta}$ einem Gewicht des Merkmals und jedes $\alpha$ dem Gewicht eines Beispiels $\vec{x}$, $\phi(\vec{x}) = \vec{x}$.
- Wenn nicht, wie finden wir zu jedem $\phi(\vec{x})$ den Ursprung $\vec{x}$?

$$\begin{aligned} f(\vec{x}) &= \sum_{i=1}^{N} \alpha_i K(\vec{x_i}, \vec{x}) \\ &= \sum_{i=1}^{s} \alpha_i \phi(\vec{x_i}) \cdot \phi(\vec{x}) \\ &= \left( \sum_{i=1}^{N} \alpha_i \phi(\vec{x_i}) \right) \cdot \phi(\vec{x}) \\ &=: \vec{\beta} \cdot \phi(\vec{x}) \end{aligned}$$

**Pre-Image Problem von Mika et al. 1998**

Mika, Schölkopf, Smola, Müller, Scholz, Rätsch (1998) Kernel PCA and de-noising in feature spaces, in: NIPS, vol 11.

Rüping (2006) Learning Interpretable Models, Diss. TU Dortmund

**Pre-Image Problem**

Gegeben die Abbildung $\phi : X \to \mathcal{X}$ und ein Element aus dem Merkmalsraum, $\vec{\beta} \in \mathcal{X}$,
finde ein $\vec{x} \in X$, so dass $\phi(\vec{x}) = \vec{\beta}$.

**Approximatives Pre-Image Problem**

Gegeben die Abbildung $\phi : X \to \mathcal{X}$ und ein Element aus dem Merkmalsraum, $\vec{\beta} \in \mathcal{X}$,
finde ein $\vec{x} \in X$ mit minimalem Fehler $\| \vec{\beta} - \phi(\vec{x}) \|^2$.

**Den Ursprung im Merkmalsraum suchen**

Weil wir die genaue Abbildung $\phi$ nicht kennen, müssen wir den quadratischen Fehler im Merkmalsraum minimieren, um $\vec{x}$ zu finden.

$$\begin{aligned} \vec{x} &= argmin \| \vec{\beta} - \phi(\vec{x}) \|^2 \\ &= argmin \langle \beta , \beta \rangle - \langle 2\beta , \phi(\vec{x}) \rangle + \langle \phi(\vec{x}) , \phi(\vec{x}) \rangle \\ &= argmin \langle \beta , \beta \rangle - 2f(\vec{x}) + K(\vec{x}, \vec{x}) \end{aligned}$$

Minimum von $K(\vec{x}, \vec{x}) - 2f(\vec{x})$ (Gradientenabstieg) kann das Pre-Image von $\vec{\beta}$ liefern (oder ein lokales Minimum).

**Pre-Images lernen!**

Wenn wir häufiger für den selben Merkmalsraum den Ursprung $\vec{x}$ von $\phi(\vec{x})$ bestimmen wollen, dann lohnt es sich, die umgekehrte Abbildung $\Gamma : \mathcal{X} \to X$ zu lernen.

Allerdings müssen wir dann für den Merkmalsraum eine geeignete Basis finden, z.B. durch eine Hauptkomponentenanalyse mit Kernfunktion.

Auf dieser Basis wird dann für eine kleinere Menge $\vec{x_i}$ die Abbildung $\Gamma$ approximiert.

Bair, Weston, Schölkopf (2003) Learning to find pre-images, in: NIPS, vol. 16

**Reduced Set Problem**

**Reduced Set Problem**

Gegeben die Abbildung $\phi : X \to \mathcal{X}$ und eine natürliche Zahl $s$,
finde $\vec{z_1}, ..., \vec{z_s} \in X$ und Koeffizienten $\gamma_1, ..., \gamma_s$
so dass $\| \vec{\beta} - \sum_{i=1}^{s} \gamma_i \phi(\vec{z_i}) \|^2$ minimal ist.

Das gelernte $\vec{\beta} = \sum_{i=1}^{N} \alpha_i \phi(\vec{x_i})$ ist eine Linearkombination der Stützvektoren. Diese sind die erste Lösung des Problems.

Wir wollen aber nicht alle Daten bearbeiten, sondern nur $s << N$!

Wir wollen $\vec{\gamma}$ aus weniger Beispielen lernen. Das ist möglich, weil hier nicht die Nebenbedingungen gelten wie bei dem Optimierungsproblem der SVM.

**Neues Optimierungsproblem**

SVM liefert $\vec{\beta} = \sum_{i}^{N} \alpha_i \phi(\vec{x_i})$

Wir wollen die Distanz der Approximation zur originalen SVM minimieren:

$$\| \sum_{i=1}^{N} \alpha_i \phi(x_i) - \sum_{i=1}^{N} \gamma_i \phi(x_i) \|^2 + \lambda \sum |\gamma_i|$$

und $\gamma$ soll spärlich besetzt sein. $\lambda > 0$ gewichtet die Spärlichkeit gegen die Präzision.

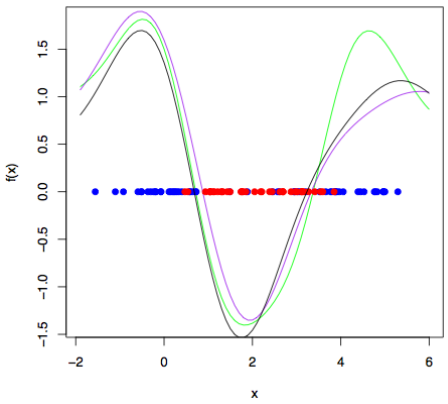
Schölkopf, Mika, Burges, Knirsch, Müller, Rätsch, Smola (1999) Input space versus feature space in kernel-based methods. IEEE Neural Networks Vol.10, No. 5

**Iterativer Algorithmus – Skizze**

1. k=:0; $Z_k =: \{\}$
2. finde $z_k$, so dass gilt $\mathbf{K}^{zx}\alpha = \mathbf{K}^{z}\gamma$ Kernmatrix $\mathbf{K}^{zx} = \langle \phi(\vec{z_i}), \phi(\vec{x_i}) \rangle$, Kernmatrix $\mathbf{K}^{z} = \langle \phi(\vec{z_i}), \phi(\vec{z_j}) \rangle$ der neue Punkt $\vec{z}$ mit $\gamma$ verhält sich wie mit $\beta$ bei allen Beispielen.
3. k=: k+1; $Z_k =: Z_{k-1} \cup z_k$
4. berechne $\gamma = (\mathbf{K}^{z})^{-1}\mathbf{K}^{zx}\alpha$
5. Wenn $\| \sum_{i=1}^{N} \alpha_i \phi(x_i) - \sum_{i=1}^{N} \gamma_i \phi(x_i) \|^2 + \lambda \sum |\gamma_i| < \theta$, stop, sonst Schritt 2!

**Reduced Set Approximation – Bild**

Bei einem eindimensionalen Datensatz mit Klassen blau und rot, sieht man die Funktionswerte der tatsächlichen Funktion (grün), die Approximation lt. Schölkopf et al (1999) (lila) und die Approximation lt. Rüping (2006) (schwarz):




**Was wissen Sie jetzt?**

- Lineare SVM sind leicht zu interpretieren: $\alpha$ gewichtet Beispiele, $\beta$ gewichtet Merkmale.
- Bei Kernfunktionen wissen wir für gegebene Wert $\phi(\vec{x})$ nicht, welches $\vec{x}$ dahinter steht.
- Ansatz: zu einer SVM noch eine Approximation der SVM lernen!
  - Die gelernte SVM klassifiziert mit max margin.
  - Die Approximation gibt eine Vorstellung von der Funktion.
  - Das Reduced Set Problem findet eine Approximation für wenige Beispiele mit $\gamma$ statt $\beta$ auf der Grundlage eines gelernten Modells.

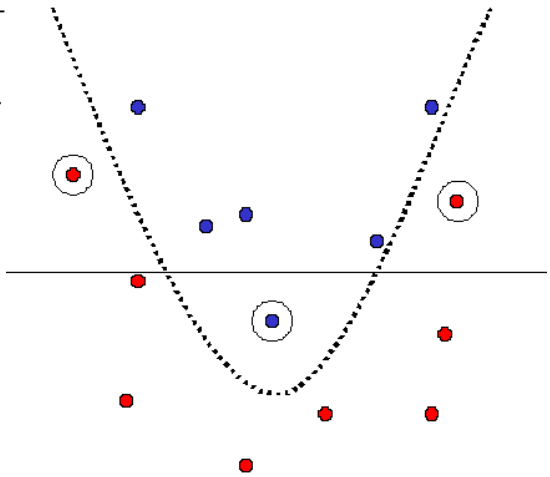
## 5.5 Bias und Varianz bei SVM

**Was ist gutes Lernen?**

- Fauler Botaniker: “klar ist das ein Baum - ist ja grün.”
  - Übergeneralisierung
  - Wenig Kapazität
  - Bias
- Botaniker mit fotografischem Gedächtnis: “nein, dies ist kein Baum, er hat 15 267 Blätter und kein anderer hatte genau so viele.”
  - Overfitting
  - Viel Kapazität
  - Varianz
- Kontrolle der Kapazität!

**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: $R(\alpha) \leq_\eta R_{emp}(\alpha) + Var(\alpha)$



**Risikoschranke nach Vapnik**

**Strukturelles Risiko**

Gegeben eine unbekannte Wahrscheinlichkeitsverteilung $P(\vec{x}, y)$, nach der Daten gezogen werden. Die Abbildungen $\vec{x} \rightarrow f(\vec{x}, \vec{\alpha})$ werden dadurch gelernt, dass $\vec{\alpha}$ bestimmt wird. Mit einer Wahrscheinlichkeit $1 - \mu$ ist das Risiko $R(\vec{\alpha})$ nach dem Sehen von $N$ Beispielen beschränkt:

$$R(\vec{\alpha}) \leq R_{emp}(\vec{\alpha}) + \underbrace{\sqrt{\frac{\eta\left(\log\left(\frac{2N}{\eta}\right)+1\right) - \log\left(\frac{\mu}{4}\right)}{N}}}_{\text{VC confidence}}$$

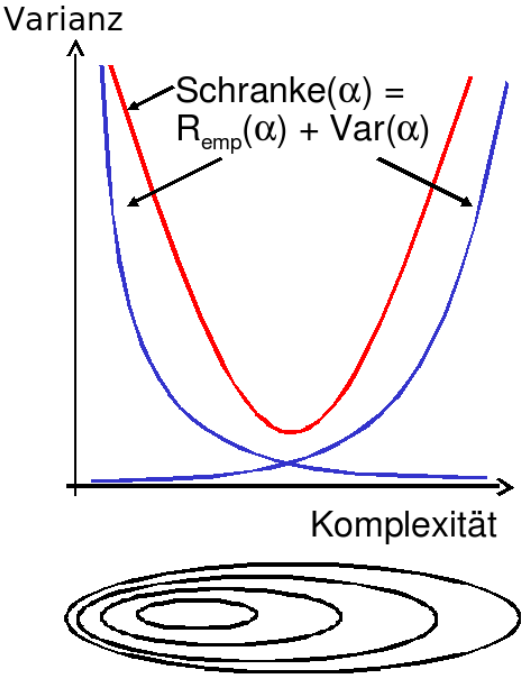
Bevor wir $\eta$ ergründen (Vapnik-Chervonenkis-Dimension), erst einmal festhalten, was die Bedeutung dieser Schranke ist!

**Strukturelle Risikoschranke**

- Unabhängig von einer Verteilungsannahme. Alles, was die Schranke braucht, ist, dass Trainings- und Testdaten gemäß der selben Wahrscheinlichkeitsverteilung gezogen werden.
- Das tatsächliche Risiko können wir nicht berechnen.
- Die rechte Seite der Ungleichung können wir berechnen, sobald wir $\eta$ kennen, die Vapnik-Chervonenkis-Dimension.
- Gegeben eine Menge Hypothesen für $f(\vec{x}, \vec{\alpha})$, wähle immer die mit dem niedrigsten Wert für die rechte Seite der Schranke ($R_{emp}$ oder VC confidence niedrig).

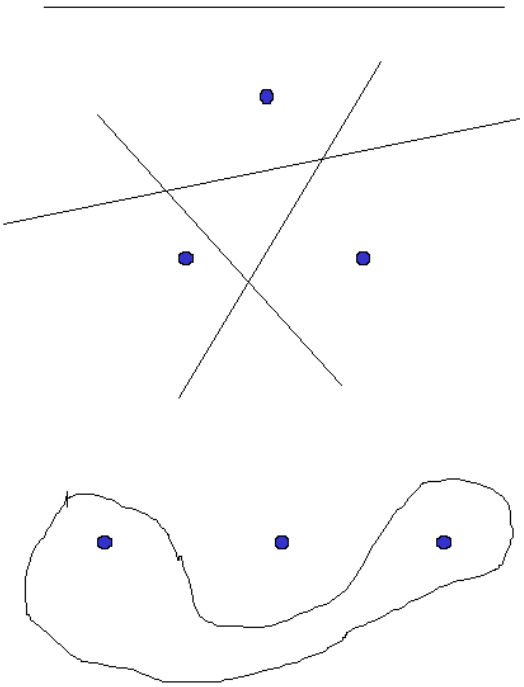
**Strukturelle Risikominimierung**

1. Ordne die Hypothesen in Teilmengen gemäß ihrer Komplexität.
2. Wähle in jeder Teilmenge 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.
- Eine Menge von 3 Punkten kann von geraden Linien zerschmettert werden, keine Menge von 4 Punkten kann von geraden Linien zerschmettert werden.



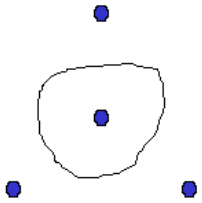
**ACHTUNG**

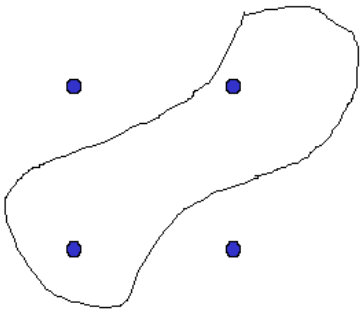
- Für eine Klasse von Lernaufgaben gibt es mindestens eine Menge $E$, die zerschmettert werden kann - NICHT jede Menge $E$ kann zerschmettert werden!
- Zum Beweis der VC Dimension $n$ muss man also zeigen:
  - Es gibt eine Menge $E$ aus $n$ Punkten, die von $H$ zerschmettert werden kann. $VCdim(H) \geq n$
  - Es kann keine Menge $E'$ aus $n+1$ Punkten geben, die von $H$ zerschmettert werden könnte. $VCdim(H) \leq n$

**VC-Dimension von Hyperebenen**

Satz: Die VC-Dimension der Hyperebenen im $R^p$ ist $p+1$.
Beweis:

- $VCdim(R^p) \geq p+1$ : Wähle $\vec{x_0} = 0$ und $\vec{x_i} = (0, \ldots, 0, 1, 0, \ldots 0)$. Für eine beliebige Teilmenge $A$ von $(\vec{x_0}, \ldots, \vec{x_n})$ setze $y_i = 1$, falls $\vec{x_i} \in A$,
  sonst $y_i = -1$.
  Definiere $\vec{\beta} = \sum y_k \vec{x_k}$ und $\beta_0 = \frac{y_0}{2}$. Dann gilt $\vec{\beta}\vec{x_0} + \beta_0 = \frac{y_0}{2}$ und $\vec{\beta}\vec{x_i} + \beta_0 = y_i + \frac{y_0}{2}$.
  Also: $\vec{\beta}\vec{x} + \beta_0$ trennt $A$.
- $VCdim(R^p) \leq p+1$ : Zurückführen auf die beiden Fälle rechts.





**VCdim misst Kapazität**

- Eine Funktion mit nur 1 Parameter kann unendliche $VCdim$ haben: $H$ kann Mengen von $n$ Punkten zerschmettern, egal wie groß $n$ ist.

- $H$ kann unendliche $VCdim$ haben und trotzdem kann ich eine kleine Zahl von Punkten finden, die $H$ nicht zerschmettern kann.
- $VCdim$ ist also nicht groß, wenn die Anzahl der Parameter bei der Klasse von Funktionen $H$ groß ist.

**VC-Dimension der SVM**

- Gegeben seien Beispiele $\vec{x_1}, \ldots, \vec{x_N} \in \mathcal{R}^p$ mit $\|\vec{x_i}\| < D$ für alle $i$. Für die VC-Dimension der durch den Vektor $\vec{\beta}$ gegebenen optimalen Hyperebene $H$ gilt:

$$VCdim(H) \leq \min\left\{D^2\|\vec{\beta}\|^2, p\right\} + 1$$

- Die Komplexität einer SVM ist auch durch die Struktur der Lösung begrenzt!
- Die SVM minimiert nicht nur das empirische Risiko, sondern auch das strukturelle – Regularisierung.

**Zusicherungen**

- Strukturelle Risikominimierung garantiert, dass die einfachste Hypothese gewählt wird, die noch an die Daten anpassbar ist.
- Strukturelle Risikominimierung kontrolliert die Kapazität des Lernens (weder fauler noch fotografischer Botaniker).
- Die Strukturen von Klassen von Funktionen werden durch die $VCdim$ ausgedrückt. Große $VCdim \rightarrow$ große VC-confidence.
- Wir haben nun also ein Verfahren, dass ohne zusätzlichen Aufwand die Komplexität regularisiert, wie wir es bei der *Modellselektion* für lineare und lokale Modelle mal wollten.

**Performanzschätzer**

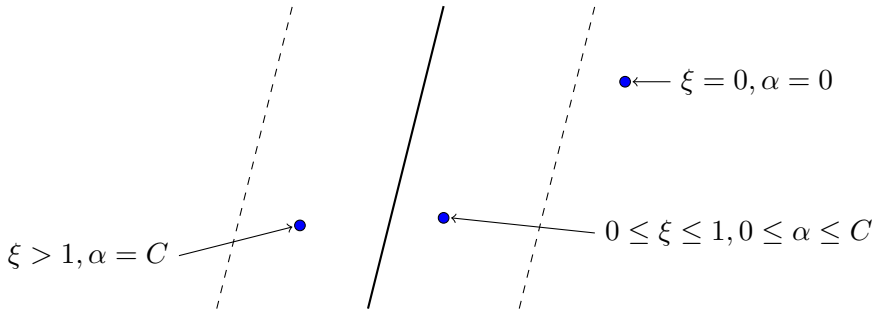
- Welches erwartete Risiko $R(\alpha)$ erreicht SVM?
- $R(\vec{\alpha})$ selbst nicht berechenbar
- Trainingsfehler (zu optimistisch - Overfitting)
- Obere Schranke mittels VC-Dimension (zu locker)
- Kreuzvalidierung / Leave-One-Out-Schätzer (ineffizient)

**Performanzschätzer II**

- Satz: Der Leave-One-Out-Fehler einer SVM ist beschränkt durch $R_{l1o} \leq \frac{|SV|}{N}$
- Beweis (Skizze):
  - Falsch klassifizierte Beispiele werden Stützvektoren (SV).
  - Also: Nicht-Stützvektoren werden korrekt klassifiziert. Weglassen eines Nicht-Stützvektors ändert die Hyperebene nicht, daher wird es auch beim $l1o$-Test richtig klassifiziert.
  - Nur der Anteil der Stützvektoren an den Beispielen macht den Fehler aus.

**Performanzschätzer III**

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




**Was wissen wir jetzt?**

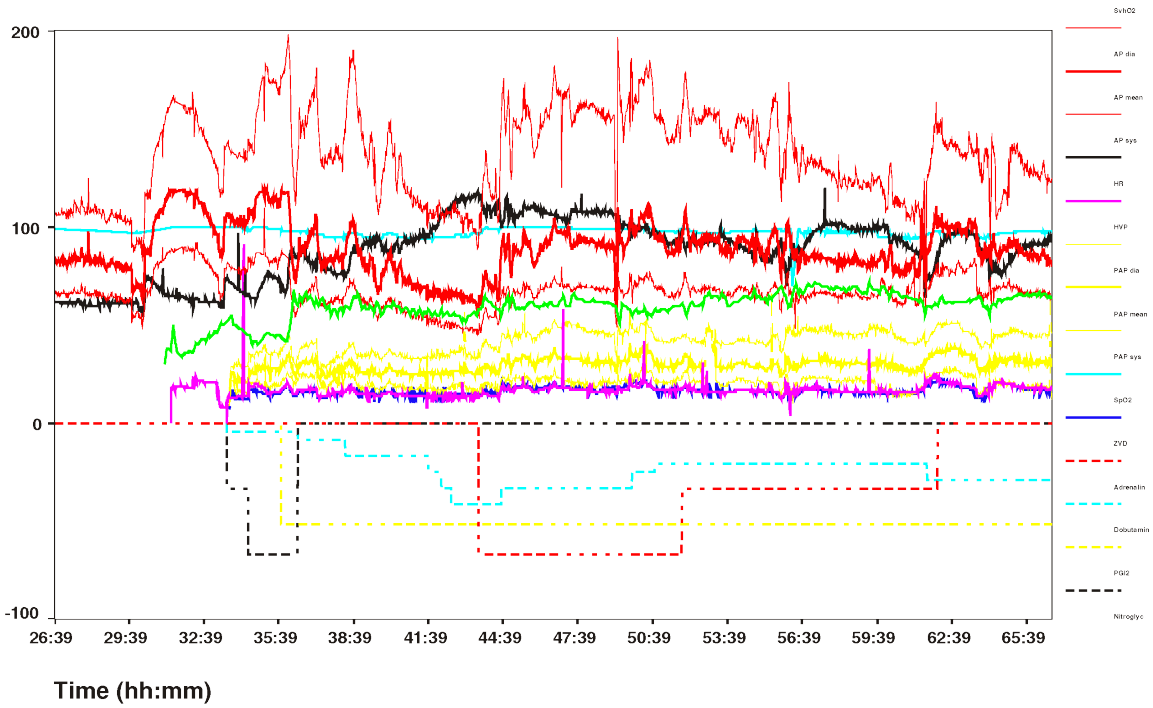
- Kernfunktionen - eine Transformation, die man nicht erst durchführen und dann mit ihr rechnen muss, sondern bei der nur das Skalarprodukt gerechnet wird.
- Idee der Regularisierung:
  - obere Schranke für das Risiko
  - Schrittweise Steigerung der Komplexität
- Formalisierung der Komplexität: VC-Dimension
- Regularisierung als strukturelle Risikominimierung der SVM
- Garantie für die Korrektheit der Lernstrategie

## 5.6 Anwendungen

**Fallstudie Intensivmedizin**

- Städtische Kliniken Dortmund, Intensivmedizin 16 Betten, Prof. Dr. Michael Imhoff (Ruhr-Universität Bochum)
- Hämodynamisches Monitoring, minütliche Messungen
  - Diastolischer, systolischer, mittlerer arterieller Druck
  - Diastolischer, systolischer, mittlerer pulmonarer Druck
  - Herzrate
  - Zentralvenöser Druck
- Therapie, Medikamente:
  - Dobutamine, adrenaline, glycerol trinitrate, noradrenaline, dopamine, nifedipine

**Patient G.C., male, 60 years old - Hemihepatektomie right**




**Wann wird Medikament gegeben?**

- Mehrklassenproblem in mehrere 2-Klassen-Probleme umwandeln:
  - Für jedes Medikament entscheide, ob es gegeben werden soll oder nicht.
  - Positive Beispiele: alle Minuten, in denen das Medikament gegeben wurde
  - Negative Beispiele: alle Minuten, in denen das Medikament nicht gegeben wurde

  Parameter: Kosten falscher Positiver = Kosten falscher Negativer

  Ergebnis: Gewichte der Vitalwerte $\vec{\beta}$, so dass positive und negative Beispiele maximal getrennt werden (SVM).

**Beispiel: Intensivmedizin**

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

**Wie wird ein Medikament dosiert ?**

- Mehrklassenproblem in mehrere 2 Klassenprobleme umwandeln: für jedes Medikament und jede Richtung (increase, decrease, equal), 2 Mengen von Patienten-daten:
  - Positive Beispiele: alle Minuten, in denen die Dosierung in der betreffenden Richtung geändert wurde
  - Negative Beispiele: alle Minuten, in denen die Dosierung nicht in der betreffenden Richtung geändert wurde.

**Steigern von Dobutamine**

Vektor $\vec{\beta}$ für $p$ Attribute

| | |
|---|---|
| $ARTEREN$: | -0.05108108119 |
| $SUPRA$ : | 0.00892807538657973 |
| $DOBUTREX$ : | -0.100650806786886 |
| $WEIGHT$ : | -0.0393531801046265 |
| $AGE$ : | -0.00378828681071417 |
| $ARTSYS$ : | -0.323407537252192 |
| $ARTDIA$ : | -0.0394565333019493 |
| $ARTMN$ : | -0.180425080906375 |
| $HR$ : | -0.10010405264306 |
| $PAPSYS$ : | -0.0252641188531731 |
| $PAPDIA$ : | 0.0454843337112765 |
| $PAPMN$ : | 0.00429504963736522 |
| $PULS$ : | -0.0313501236399881 |

**Anwendung des Gelernten für Dobutamin**

- Patientwerte
  pat46, artmn 95, min. 2231
  ...
  pat46, artmn 90, min. 2619
- Gelernte Gewichte $\beta_i$:
  $artmn - 0,18$
  ...

$$svm\_calc = \sum_{i=1}^{p} \beta_i x_i$$

$$decision = sign(svm\_calc + \beta_0)$$

- $svm\_calc(pat46, dobutrex, up, min.2231, 39)$
- $svm\_calc(pat46, dobutrex, up, min.2619, 25)$
- $\beta_0 = -26$, i.e. increase in minute 2231, not increase in minute 2619.

**Steigern von Glyceroltrinitrat (nitro)**

$$f(x) = \left[ \begin{pmatrix} 0.014 \\ 0.019 \\ -0.001 \\ -0.015 \\ -0.016 \\ 0.026 \\ 0.134 \\ -0.177 \\ -9.543 \\ -1.047 \\ -0.185 \\ 0.542 \\ -0.017 \\ 2.391 \\ 0.033 \\ 0.334 \\ 0.784 \\ 0.015 \end{pmatrix} \begin{pmatrix} artsys = 174.00 \\ artdia = 86.00 \\ artmn = 121.00 \\ cvp = 8.00 \\ hr = 79.00 \\ papsys = 26.00 \\ papdia = 13.00 \\ papmn = 15.00 \\ nifedipine = 0 \\ noradrenaline = 0 \\ dobutamie = 0 \\ dopamie = 0 \\ glyceroltrinitrate = 0 \\ adrenaline = 0 \\ age = 77.91 \\ emergency = 0 \\ bsa = 1.79 \\ broca = 1.02 \end{pmatrix} - 4.368 \right]$$

- Jedes Medikament hat einen Dosierungsschritt. Für Glyceroltrinitrat ist es 1, für $Suprarenin$ (adrenalin) 0,01. Die Dosis wird um einen Schritt erhöht oder gesenkt.
- Vorhersage: $pred\_interv$ $(pat49, min.32,$ $nitro, 1, 0)$

**Evaluierung**

- Blind test über 95 noch nicht gesehener Patientendaten.
  - Experte stimmte überein mit tatsächlichen Medikamentengaben in 52 Fällen
  - SVM Ergebnis stimmte überein mit tatsächlichen Medikamentengaben in 58 Fällen

| *Dobutamine* | *Actual up* | *Actual equal* | *Actual down* |
|---|---|---|---|
| *Predicted up* | **10 (9)** | 12 (8) | 0 (0) |
| *Predicted equal* | 7 (9) | **35 (31)** | 9 (9) |
| *Predicted down* | 2 (1) | 7 (15) | **13 (12)** |

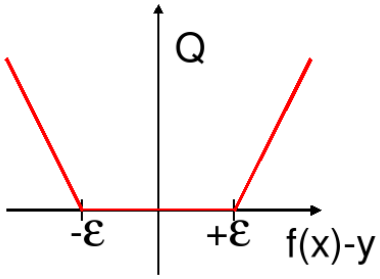
**SVMs für Regession**

Durch Einführung einer anderen *Loss-Funktion* läßt sich die SVM zur Regression nutzen. Sei $\varepsilon \in \mathbb{R}_{>0}$ und

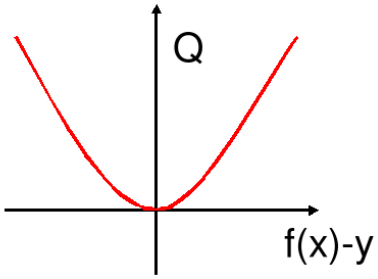
$$L_k(y, f(\vec{x}, \alpha)) = \begin{cases} 0 & \text{, falls } y - f(\vec{x}, \alpha) \leq \varepsilon \\ (y - f(\vec{x}, \alpha) - \varepsilon)^k & \text{, sonst} \end{cases}$$

Die *Loss-Funktion* $L_1$ gibt den Abstand der Funktion $f$ von den Trainingsdaten an, alternativ quadratische Loss-Funktion $L_2$:

lineare Verlustfunktion




quadratische Verlustfunktion




**SVMs für Regression**

Dadurch ergibt sich das Optimierungsproblem:

**Regressions-SVM**

Minimiere

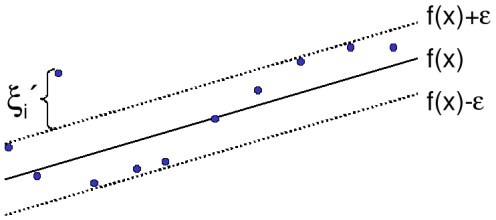
$$\|\vec{\beta}\|^2 + C\left(\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\xi_i'\right)$$

unter den Nebenbedingungen

$$\begin{aligned} f(\vec{x_i}) &= \left\langle \vec{\beta}\,,\,\vec{x_i}\right\rangle + \beta_0 \leq y_i + \epsilon + \xi_i' \\ f(\vec{x_i}) &= \left\langle \vec{\beta}\,,\,\vec{x_i}\right\rangle + \beta_0 \geq y_i - \epsilon - \xi_i \end{aligned}$$

**SVMs für Regression**

Die $\xi_i$ bzw. $\xi_i'$ geben für jedes Beispiel Schranken an, innerhalb derer der vorhergesagte Funktionswert für jedes Beispiel liegen soll:




Bei der Lösung des Optimierungsproblems mit Lagrange führt dies zu *zwei* $\alpha$-Werten je Beispiel!

**SVMs für Regression**

Das duale Problem enthält für jedes $\vec{x}_i$ je zwei $\alpha$-Werte $\alpha_i$ und $\alpha_i'$, je einen für $\xi_i$ und $\xi_i'$, d.h.

**Duales Problem für die Regressions-SVM**

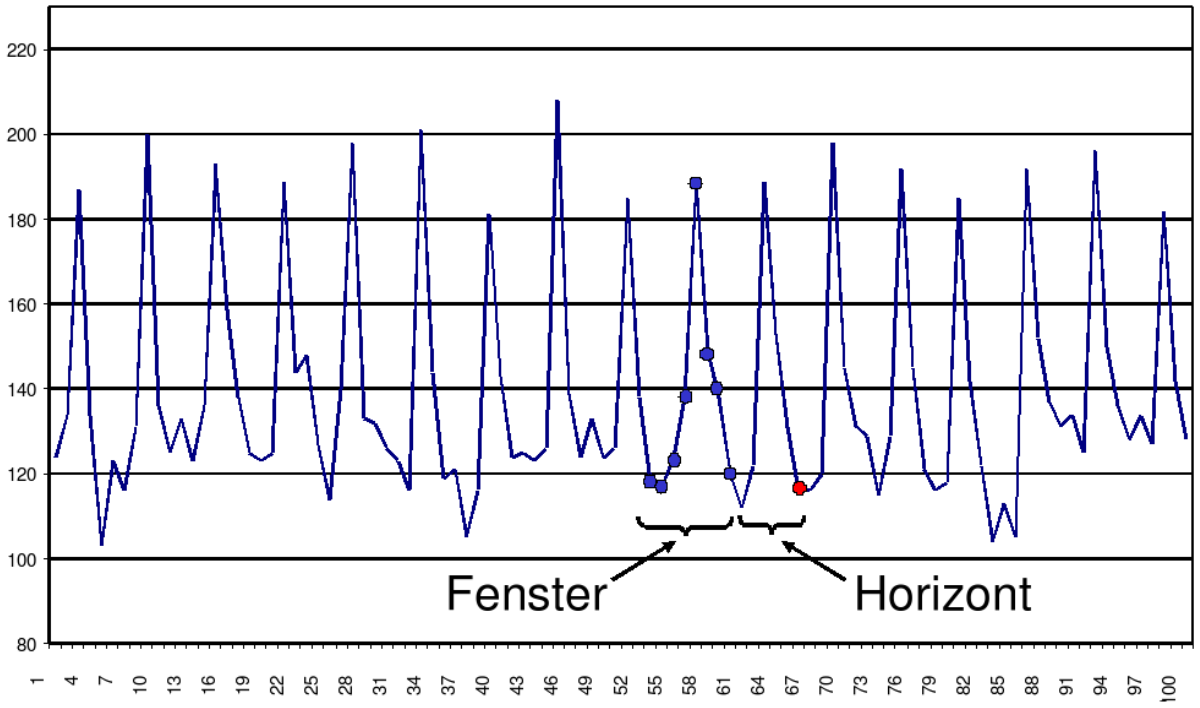
Maximiere

$$L_D(\vec{\alpha}, \vec{\alpha}') = \sum_{i=1}^{N} y_i \left(\alpha_i' - \alpha_i\right) - \epsilon \sum_{i=1}^{N} y_i \left(\alpha_i' - \alpha_i\right) - \frac{1}{2} \sum_{i,j=1}^{n} y_i \left(\alpha_i' - \alpha_i\right) \left(\alpha_j' - \alpha_j\right) K(\vec{x_i}, \vec{x_j})$$

unter den Nebenbedingungen

$$0 \leq \alpha_i, \alpha_i' \leq C \,\forall\, i = 1, \ldots, N \quad \text{und} \quad \sum_{i=1}^{N} \alpha_i' = \sum_{i=1}^{N} \alpha_i$$

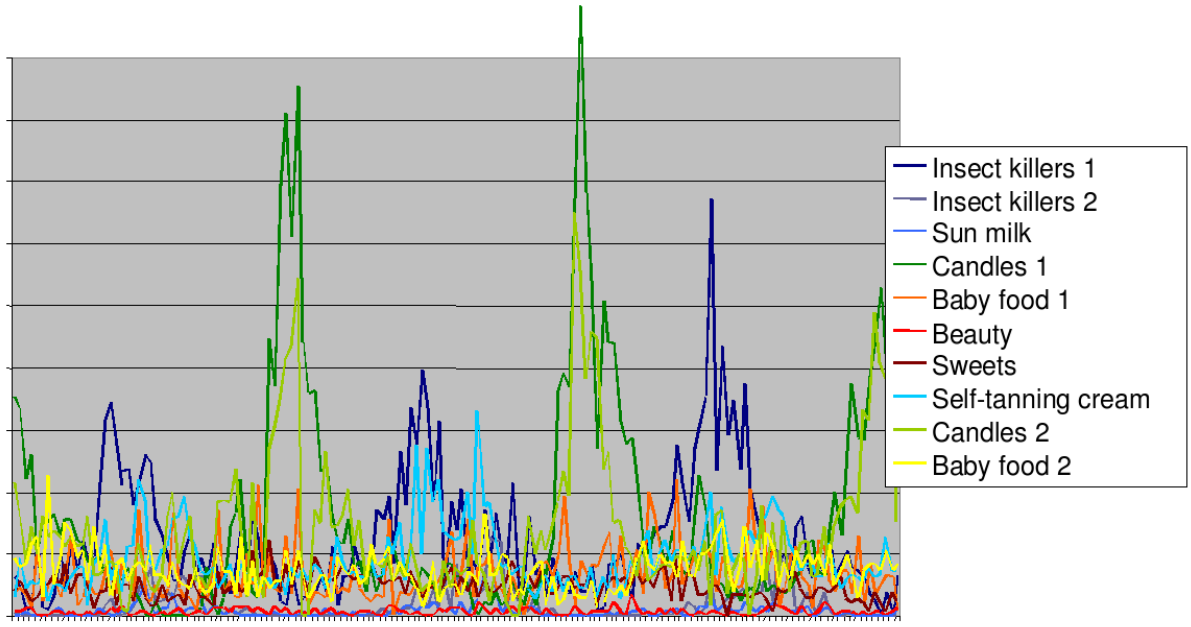
**Beispiel: Prognose von Zeitreihen**




**Prognose von Zeitreihen**

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

**Abverkauf Drogerieartikel**




**Vorhersage Abverkauf**

Gegeben Verkaufsdaten von 50 Artikeln in 20 Läden über 104 Wochen

Vorhersage Verkäufe eines Artikels, so dass

- Die Vorhersage niemals den Verkauf unterschätzt,
- Die Vorhersage überschätzt weniger als eine Faustregel.

Beobachtung 90% der Artikel werden weniger als 10 mal pro Woche verkauft.

Anforderung Vorhersagehorizont von mehr als 4 Wochen.

**Verkaufsdaten – multivariate Zeitreihen**

| Shop | Week | Item1 | ... | Item50 |
|---|---|---|---|---|
| Dm1 | 1 | 4 | ... | 12 |
| Dm1 | ... | ... | ... | ... |
| Dm1 | 104 | 9 | ... | 16 |
| Dm2 | 1 | 3 | ... | 19 |
| ... | ... | ... | ... | ... |
| Dm20 | 104 | 12 | ... | 16 |

**Vorverarbeitung: multivariat nach univariat**

*Quasi-SQL*:
For all shops for all items: Create view Univariate as Select shop, week, $item_i$
Where shop="$dm_j$"
From Source;

| Shop_Item | Week | Sale | Week | Sale |
|---|---|---|---|---|
| Dm1_Item1 | 1 | 4... | 104 | 9 |
| ... | | | | |
| Dm1_Item50 | 1 | 12... | 104 | 16 |
| ... | | | | |
| Dm20_Item50 | 1 | 14... | 104 | 16 |

- Multiples Lernen für alle univariaten Zeitreihen

**Vorverarbeitung II**

- Problem: eine Zeitreihe ist nur 1 Beispiel!
- Das ist für das Lernen zu wenig.
- Lösung: Viele Vektoren aus einer Reihe gewinnen durch Fenster der Breite (Anzahl Zeitpunkte) $w$, bewege Fenster um $m$ Zeitpunkte weiter.

| Shop_Item_Window | Week | Sale | Week | Sale |
|---|---|---|---|---|
| Dm1_Item1_1 | 1 | 4... | 5 | 7 |
| Dm1_Item1_2 | 2 | 4... | 6 | 8 |
| ... | ... | ... | ... | ... |
| Dm1_Item1_100 | 100 | 6... | 104 | 9 |
| ... | ... | ... | ... | ... |
| Dm20_Item50_100 | 100 | 12... | 104 | 16 |

**SVM im Regressionfall**

- Multiples Lernen: für jeden Laden und jeden Artikel, wende die SVM an. Die gelernte Regressionsfunktion wird zur Vorhersage genutzt.
- Asymmetrische Verlustfunktion :
  - Unterschätzung wird mit 20 multipliziert, d.h. 3 Verkäufe zu wenig vorhergesagt – 60 Verlust
  - Überschätzung zählt unverändert, d.h. 3 Verkäufe zu viel vorhergesagt – 3 Verlust

  (Diplomarbeit Stefan Rüping 1999)

**Vergleich mit Exponential Smoothing**

| Horizont | SVM | exp. smoothing |
|---|---|---|
| 1 | 56.764 | 52.40 |
| 2 | 57.044 | 59.04 |
| 3 | 57.855 | 65.62 |
| 4 | 58.670 | 71.21 |
| 8 | 60.286 | 88.44 |
| 13 | 59.475 | 102.24 |

Verlust, nicht normiert auf $[0, 1]$!

**Was wissen wir jetzt?**

- Anwendung der SVM für die Medikamentenverordnung
- Idee der Regressions-SVM
- Anwendung der SVM für die Verkaufsvorhersage
  - Umwandlung multivariater Zeitreihen in mehrere univariate
  - Gewinnung vieler Vektoren durch gleitende Fenster
  - Asymmetrische Verlustfunktion

## 5.7 Web Mining

**World Wide Web**

- Seit 1993 wächst die Anzahl der Dokumente – 12,9 Milliarden Seiten (geschätzt für 2005)
- Ständig wechselnder Inhalt ohne Kontrolle, Pflege
  - Neue URLs
  - Neue Inhalte
  - URLs verschwinden
  - Inhalte werden verschoben oder gelöscht
- Verweisstruktur der Seiten untereinander
- Verschiedene Sprachen
- Unstrukturierte Daten

**Aufgaben**

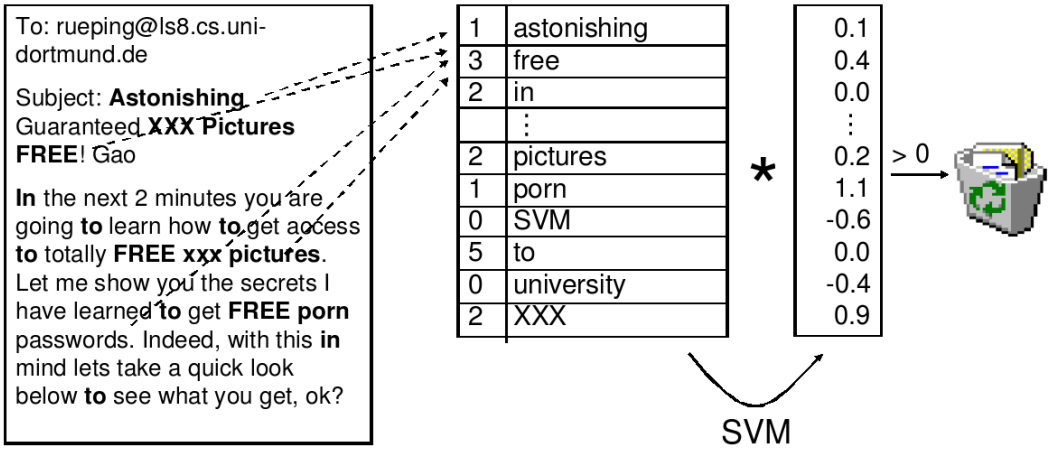
- Indexierung möglichst vieler Seiten (Google)
- Suche nach Dokumenten, ranking der Ergebnisse z.B. nach Häufigkeit der Verweise auf das Dokument (PageLink – Google)
- Kategorisierung (Klassifikation) der Seiten manuell (Yahoo), automatisch
- Strukturierung von Dokumentkollektionen (Clustering)
- Personalisierung:
  - Navigation durch das Web an Benutzer anpassen
  - Ranking der Suchergebnisse an Benutzer anpassen
- Extraktion von Fakten aus Texten

### 5.7.1 Information Retrieval

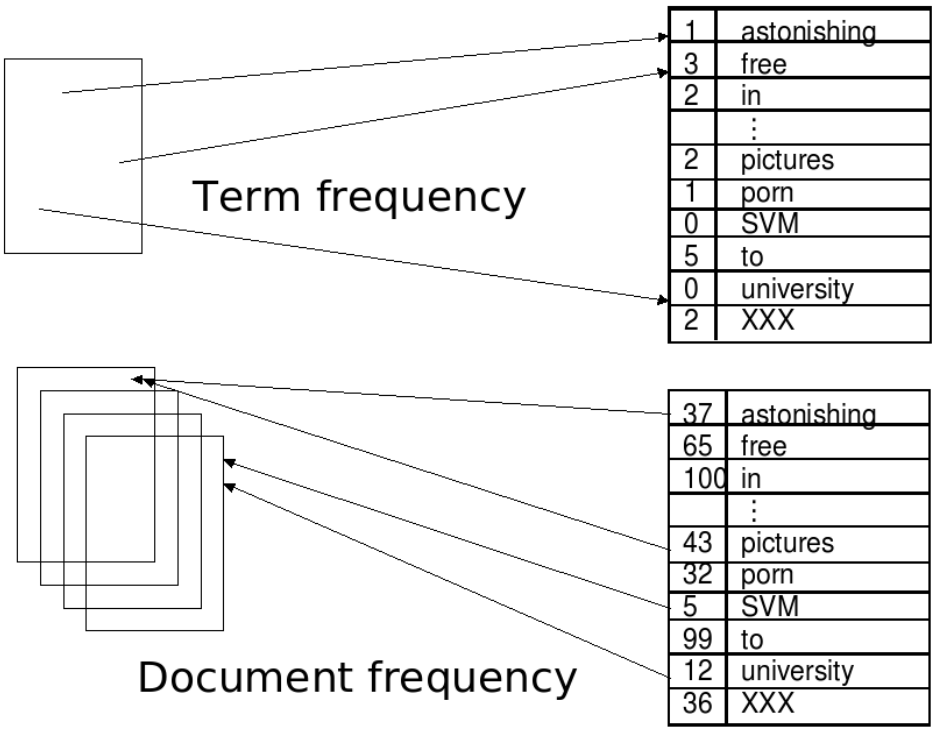
**Information Retrieval**

- Ein Dokument besteht aus einer Menge von Termen (Wörtern)
  - Bag of words: Vektor, dessen Komponenten die Häufigkeit eines Wortes im Dokument angeben.
- Für alle Dokumente gibt es eine Termliste mit Verweis auf die Dokumente.
  - Anzahl der Dokumente, in denen das Wort vorkommt.

**Beispiel zur Klassifikation**




**Texte als Daten**




**TFIDF**

- Term Frequenz: wie häufig kommt ein Wort $w_i$ in einem Dokument $d$ vor? $TF(w_i, d)$
- Dokumentenfrequenz: in wie vielen Dokumenten einer Kollektion $D$ kommt ein Wort $w_i$ vor? $DF(w_i)$
- Inverse Dokumentenfrequenz:

$$IDF(D, w_i) = \log \frac{|D|}{DF(w_i)}$$

- Bewährte Repräsentation:

$$TFIDF(w_i, D) = \frac{TF(w_i, d)IDF(w_i, D)}{\sqrt{\sum_j [TF(w_j, d)IDF(w_j, D)]^2}}$$

## 5.8 Textklassifikation

**Textklassifikation**

- Thorsten Joachims “The Maximum-Margin Approach to Learning Text Classifiers Kluwer", 2001
- Modell der Textklassifikation TCat
- Verbindung zur SVM-Theorie

→ theoretisch begründete Performanzabschätzung

**Eigenschaften der Textklassifikation 1**

- Hochdimensionaler Merkmalsraum
  - Reuters Datensatz mit 9603 Dokumenten: verschiedene Wörter
    $$V = 27658$$
  - Heapes Gesetz: Anzahl aller Wörter
    $$(s)V = ks^{\beta}$$
  - Beispiel:
    * Konkatenieren von 10 000 Dokumenten mit je 50 Wörtern zu einem,
    * $k = 15$ und $\beta = 0,5$
    * ergibt $V = 35000 \rightarrow$ stimmt!

**Eigenschaften der Textklassifikation 2**

- Heterogener Wortgebrauch
  - Dokumente der selben Klasse haben manchmal nur Stoppwörter gemeinsam!
  - Es gibt keine relevanten Terme, die in allen positiven Beispielen vorkommen.
  - Familienähnlichkeit (Wittgenstein): A und B haben ähnliche Nasen, B und C haben ähnliche Ohren und Stirn, A und C haben ähnliche Augen.

**Eigenschaften der Textklassifikation 3**

- Redundanz der Merkmale
  - Ein Dokument enthält mehrere die Klasse anzeigende Wörter.
  - Experiment:
    * Ranking der Wörter nach ihrer Korrelation mit der Klasse.
    * Trainieren von Naive Bayes für Merkmale von Rang

      1 - 200 (90% precision/recall)
      201 - 500 (75%)
      601 - 1000 (63%)
      1001- 2000 (59%)
      2001- 4000 (57%)
      4001- 9947 (51%) – zufällige Klassifikation (22%)

**Eigenschaften der Textklassifikation 4**

- Dünn besetzte Vektoren
- Reuters Dokumente durchschnittlich 152 Wörter lang
  - mit 74 verschiedenen Wörtern
  - also meist bei etwa 78 Wörtern 0
- Euklidsche Länge der Vektoren klein!

**Eigenschaften der Textklassifikation 5**

- Zipfs Gesetz: Verteilung von Wörtern in Dokumentkollektionen ist ziemlich stabil.
  - Ranking der Wörter nach Häufigkeit $(r)$
  - Häufigkeit des häufigsten Wortes $(max)$
  - $\frac{1}{r}max$ häufig kommt ein Wort des Rangs $r$ vor.
- Generalisierte Verteilung von Häufigkeit nach Rang (Mandelbrot): $v$ ist Größe der Dokumentkollektion in Wortvorkommen

$$\frac{v}{(k+r)^{\phi}}$$

**Plausibilität guter Textklassifikation durch SVM**

- $R$ sei Radius des Balles, der die Daten enthält. Dokumente werden auf einheitliche Länge normiert, so dass $R = 1$
- Margin sei $\delta$, so dass großes $\delta$ kleinem $\frac{R^2}{\delta^2}$ entspricht.

| Reuters | $\frac{R^2}{\delta^2}$ | $\sum_{i=1}^{n}\xi$ | Reuters | $\frac{R^2}{\delta^2}$ | $\sum_{i=1}^{n}\xi$ |
|---|---|---|---|---|---|
| Earn | 1143 | 0 | trade | 869 | 9 |
| acquisition | 1848 | 0 | interest | 2082 | 33 |
| money-fx | 1489 | 27 | ship | 458 | 0 |
| grain | 585 | 0 | wheat | 405 | 2 |
| crude | 810 | 4 | corn | 378 | 0 |

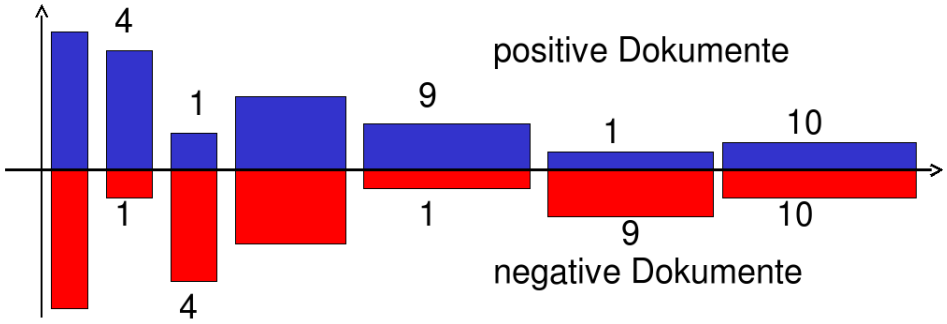
**TCat Modell – Prototyp**

- Hochdimensionaler Raum: $V = 11100$ Wörter im Lexikon
- Dünn besetzt: Jedes Dokument hat nur 50 Wörter, also mindestens 11050 Nullen
- Redundanz: Es gibt 4 mittelhäufige und 9 seltene Wörter, die die Klasse anzeigen
- Verteilung der Worthäufigkeit nach Zipf/Mandelbrot.
- Linear separierbar mit $\beta_0 = 0$, $\sum_{i=1}^{11100}\beta_i x_i$

$$\beta_i = \begin{cases} 0,23 & \text{für mittelhäufige Wörter in } POS, \\ -0,23 & \text{für mittelhäufige Wörter in } NEG, \\ 0,04 & \text{für seltene Wörter in } POS, \\ -0,04 & \text{für seltene Wörter in } NEG, \\ 0 & \text{sonst} \end{cases}$$

**TCat im Bild**

- 20 aus 100 Stoppwörtern, 5 aus 600 mittelhäufigen und 10 aus seltenen Wörtern kommen in $POS$- und $NEG$-Dokumenten vor; 4 aus 200 mittelhäufigen Wörtern in $POS$, 1 in $NEG$, 9 aus 3000 seltenen Wörtern in $POS$, 1 in $NEG$ (Es müssen nicht immer die selben Wörter sein!)




**TCat**

The TCat concept

$$TCat\left([p_1 : n_1 : f_1], \ldots, [p_s : n_s : f_s]\right)$$

describes a binary classification task with $s$ sets of disjoint features. The i-th set includes $f_i$ features. Each positive example contains $p_i$ occurences of features from the respective set and each negative example contains $n_i$ occurrences. The same feature can occur multiple times in one document. (Joachims 2002)

**TCat zum Bild**

7 disjunkte Wortmengen; bei einem zur Klasse gehörigen Dokument kommt 20 mal eines der 100 Wörter der ersten Wortmenge vor, 4 mal eines der 200 Wörter der zweiten Wortmenge, ...; bei einem nicht zur Klasse gehörigen Dokument gibt es 20 Auftreten von Wörtern aus der ersten Wortmenge,... Es sind also nicht bestimmte Wörter, die die Klassenzugehörigkeit anzeigen!

$$TCat( \underbrace{[20:20:100]}_{\text{sehr häufig}} \underbrace{[4:1:200][1:4:200][5:5:600]}_{\text{mittel häufig}} \underbrace{[9:1:3000][1:9:3000][10:10:4000]}_{selten})$$

**Lernbarkeit von TCat durch SVM**

(Joachims 2002) Der erwartete Fehler einer SVM ist nach oben beschränkt durch:

$$\frac{R^2}{n+1}\frac{a+2b+c}{ac-b^2}$$

$$a = \sum_{i=1}^{s} \frac{p_i^2}{f_i}$$

$$b = \sum_{i=1}^{s} \frac{p_i^2 n_i}{f_i}$$

$$c = \sum_{i=1}^{s} \frac{n_i^2}{f_i}$$

$$R^2 = \sum_{r=1}^{d} \left(\frac{v}{(r+k)^\phi}\right)^2$$

Es gibt $l$ Wörter, $s$ Merkmalsmengen, für einige $i$: $p_i \neq n_i$ und die Termhäufigkeit befolgt Zipfs Gesetz. Wähle $d$ so, dass:

$$\sum_{r=1}^{d} \frac{v}{(r+k)^\phi} = l$$

**Was wissen Sie jetzt?**

- Die automatische Klassifikation von Texten ist durch das WWW besonders wichtig geworden.
- Texte können als Wortvektoren mit TFIDF dargestellt werden. Die Formel für TFIDF können Sie auch!
- Textkollektionen haben bzgl. der Klassifikation die Eigenschaften: hochdimensional, dünn besetzt, heterogen, redundant, Zipfs Gesetz.
- Sie sind mit breitem margin linear trennbar.
- Das TCat-Modell kann zur Beschränkung des erwarteten Fehlers eingesetzt werden. Die Definition von TCat kennen Sie mindestens, besser wäre noch die Fehlerschranke zu kennen.

## 5.9 Verwendung des Modells zur Textklassifikation für zeitgestempelte Daten

**Verwendung des TCat Modells für zeitgestempelte Daten**

Und jetzt wenden wir das Gelernte auf ein Gebiet fernab von Texten an!

**Lokale Muster**

- Lokale Muster beschreiben seltene Ereignisse.
- Gegeben ein Datensatz, für den ein globales Modell bestimmt wurde, weichen lokale Muster davon ab.
  - Lokale Muster beschreiben Daten mit einer internen Struktur, z.B. Redundanz, Heterogenität

**Zeit-gestempelte Daten**

- Zeit-gestempelte Daten können transformiert werden in:
  - Eine Menge von Ereignissen,
  - Zeitintervalle,
  - Zeitreihen.

**Klassische Methoden**

- Zeitreihenanalyse für Vorhersage, Trend und Zyklus Erkennung
- Indexing und clustering von Zeitreihen (time warping)
- Segmentierung (motif detection)
- Entdeckung von Episoden
  - frequent sets,
  - chain logic programs (grammars)
- Regression

**Beispielrepräsentation**

- Die Beispielrepräsentation $X$ bestimmt die Anwendbarkeit der Methoden: welche Variablen, was sind Beispiele?
- Bedeutung der Repräsentation lange unterschätzt.
- Suche nach guter Repräsentation ist aufwändig.
- Transformieren der Rohdaten in die Repräsentation auch.

**Einige Repräsentationen für zeitgestempelte Daten**

- Schnappschuss: ignoriere Zeit, nimm nur den aktuellen Zustand. (So war es bei der Intensivmedizin-Anwendung.)
- Ereignisse mit Zeitintervallen: aggregiere Zeitpunkte zu Intervallen, wende frequent set mining an. (Das machen wir in dieser Vorlesung nicht.)
- Generierte Merkmale: hier: transformiere Zeitinformation in Häufigkeitsmerkmale!

**Häufigkeitsmerkmale für Zeitaspekte**

- Term frequency: wie oft änderte Attribut $A$ seinen Wert $a_i$ für ein Objekt $c_j$.

$$tf(a_i, c_j) = \| \{x \in timepoints | a_i \text{ of } c_j \text{ changed } \} \|$$

- Document frequency: in wie vielen Objekten $c_j$ änderte Attribut $A$ seinen Wert $a_i$.

$$df(a_i) = \| \{c_j \in C | a_i \text{ of } c_j \text{ changed } \} \|$$

- TF/IDF:

$$tfidf(a_i) = tf(a_i, c_j) \log \frac{\|C\|}{df(a_i)}$$

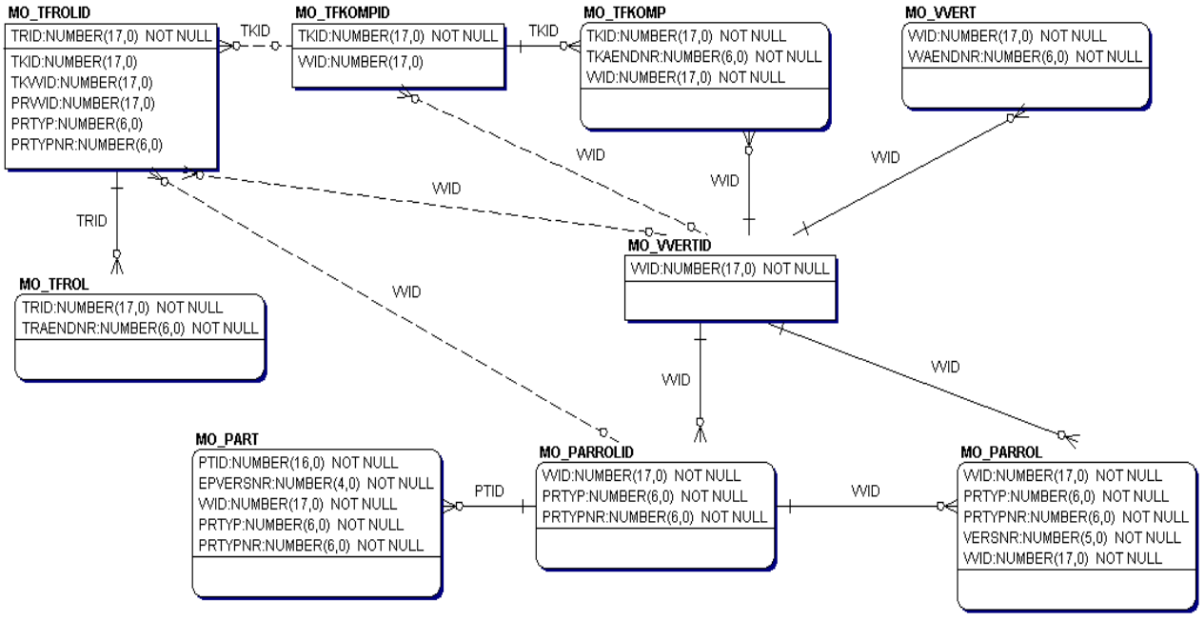
### Fallstudie SwissLife

- Lokale Muster
  - Seltenes Ereignis der Kündigung
  - Lokales Muster weicht ab vom generellen Modell
  - Interne Struktur in lokalen Mustern
- Zeit-gestempelte Daten
  - Schnappschuss
  - Zeitintervall
  - Generierte Merkmale: $TFIDF$

### Lokale Muster in Versicherungsdaten

- Nur 7.7% der Verträge enden vorzeitig (customer churn).
- Für einige Attribute weicht die likelihood in der churn-Klasse von der globalen ab.
- Interne Struktur:
  - Überlappung: häufige Mengen in churn Verträgen sind auch häufig in fortgesetzten Verträgen.
  - Redundanz: in jedem Vertrag gibt es mehrere Attribute, die auf Fortsetzung oder Kündigung hinweisen.
  - Heterogenität: Es gibt gekündigte Verträge, die nicht ein einziges Attribut gemeinsam haben.

### Database



**Contract Table**

| VVID | VVAENDNR | VVWIVON | VVWIBIS | VVAENDAT | VVAENDART | ... |
|---|---|---|---|---|---|---|
| 16423 | 1 | 1946 | 1998 | 1946 | 1000 | |
| 16423 | 2 | 1998 | 1998 | 1998 | 27 | |
| 16423 | 3 | 1998 | 1998 | 1998 | 4 | |
| 16423 | 4 | 1998 | 1998 | 1998 | 54 | |
| 16423 | 5 | 1998 | 1998 | 1998 | 4 | |
| 16423 | 6 | 1998 | 9999 | 1998 | 61 | |
| 5016 | 1 | 1997 | 1999 | 1997 | 33 | |
| 5016 | 2 | 1999 | 2001 | 1999 | 33 | |
| 5016 | 3 | 2001 | 2001 | 2001 | 33 | |
| 5016 | 4 | 2001 | 2001 | 2001 | 33 | |
| 5016 | 5 | 2001 | 2002 | 2001 | 81 | |
| 5016 | 6 | 2002 | 9999 | 2001 | 94 | |
| ... | ... | ... | ... | ... | ... | ... |

**Datensatz**

- Tabellen enthalten Informationen über
  - 217586 Komponenten and
  - 163745 Kunden
- Attribute:
  - 14 Attributes ausgewählt
  - Eines der Attribute gibt den Grund an für einen Wechsel. Es gibt 121 Gründe. Daraus werden 121 Boolean Attribute.
  - 134 Attribute mit $TFIDF$ Werten.

**Erste Experimente**

- Bei SwissLife wurde die Abweichung der Wahrscheinlichkeit bestimmter Attributwerte in gekündigten und fortgesetzten Verträgen festgestellt anhand der Schnappschussrepräsentation $\rightarrow$ keine operationale Vorhersage.

**Calculating Term Frequency**

| VVID | ... | VVSTACD | VVPRFIN | VVPRZA | VVINKZWEI | VVBEG | VVEND | VVINKPRL | ... |
|---|---|---|---|---|---|---|---|---|---|
| 16423 | | 4 | 1 | 2 | 2 | 1946 | 1998 | 295.29 | |
| 16423 | | 4 | 1 | 2 | 2 | 1946 | 1998 | 295.29 | |
| 16423 | | 4 | 5 | 2 | 0 | 1946 | 2028 | 0 | |
| 16423 | | 5 | 3 | 2 | 0 | 1946 | 2028 | 0 | |
| 16423 | | 4 | 1 | 2 | 2 | 1946 | 1998 | 295.29 | |
| 16423 | | 5 | 3 | 2 | 0 | 1946 | 1998 | 0 | |

| | |
|---|---|
| 3 | VVSTACD |
| 4 | VVPRFIN |
| 0 | VVPRZA |
| 3 | VVINKZWEI |
| 0 | VVBEG |
| 2 | VVEND |
| 3 | VVINKPRL |

**Experimente mit der TFIDF Repräsentation**

- Vergleich der originalen Repräsentation und der TFIDF
  - 10fold cross validation
    - Apriori mit Konklusion 'churn'
    - Entscheidungsbaumlerner J4.8
    - Naive Bayes
    - mySVM mit linearem Kern
  - F-measure balanciert precision und recall gleich.

Alle Lernalgorithmen werden besser mit der $TFIDF$- Repräsentation.

**Resultate (F-measure)**

| Lerner | TF/IDF repr. | Original repr. |
|---|---|---|
| Apriori | 63.35 | 30.24 |
| J4.8 | 99.22 | 81.21 |
| Naive Bayes | 51.8 | 45.41 |
| mySVM | 97.95 | 16.06 |

**Erklärung?**

- TF/IDF stammt aus Lernen über Texten.
- Dazu gibt es eine Theorie – TCat.
- Können wir die auch hier einsetzen??

**Datenbeschreibung im TCat Modell**

$$TCat(\underbrace{[2:0:2],[1:4:3],}_{\text{high frequency}}\underbrace{[3:1:3],[0:1:4],}_{\text{medium frequency}}\underbrace{[1:0:19],[0:1:64],}_{\text{low frequency}}\underbrace{[1:1:39]))}_{\text{rest}}$$

$[1:4:3]$ : Aus der Menge von 3 Merkmale finden wir ein Auftreten in positiven und 4 in negativen Beispielen.

**Learnability of TCat**

Error bound (Joachims 2002)

$$\frac{R^2}{n+1}\frac{a+2b+c}{ac-b^2}$$

$$\begin{aligned} a &= \textstyle\sum_{i=1}^{s} \frac{p_i^2}{f_i} && = 5.41 \\ b &= \textstyle\sum_{i=1}^{s} \frac{p_i n_i}{f_i} && = 2.326 \\ c &= \textstyle\sum_{i=1}^{s} \frac{n_i^2}{f_i} && = 5.952 \\ R^2 &= \textstyle\sum_{r=1}^{d} \left(\frac{c}{(r+k)^{\phi}}\right)^2 && \leq 37 \end{aligned}$$

Nach 1000 Beispielen erwarteter Fehler $\leq 2.2\%$ Tatsächlicher Fehler $2.05\%$

**Experimente zu lokalen Mustern**

- Durch TCat-Konzepte Daten künstlich generieren.
- Lokale Muster als seltene Ereignisse mit interner Struktur.

**Lokale Muster: Verzerrte Verteilung**

- 10 000 Beispiele mit 100 Attributen
- SVM runs mit 10 fold cross validation

| *Repr.* | $Targetconcept:$ | Verzerrung: |
|---|---|---|
| TF/IDF | 1. change of a particular attribute | 50%, 25%, |
| Boolean | 2. frequency of changes | 12.5%, 6.25% |

**Lokale Muster: Strukturen**

- 10 000 Beispiele mit 100 Attributen
- 20 Attribute wechseln pro Beispiel (dünn besetzt)
- Variieren:
  - Heterogenität: $\frac{f_i}{p_i}$ Beispiele der selben Klasse haben kein gemeinsames Attribut $4, 5, 10, 20$
  - Redundanz: $\frac{p_i}{f_i}$ oder $\frac{n_i}{f_i}$ für die Redundanz innerhalb einer Klasse $0.5, 0.2, 0.1$
  - Überlappung: einige Attribute sind häufig in beiden Klassen $0.25, 0.66$

**Resultate**

- Für alle Kombinationen ohne Überlappung sind die Lernergebnisse 100% in Boolean und im TF/IDF- Format.
- Mehr Überlappung verschlechtert das Lernen bei Boolean auf 68.57% F-measure.
- Für alle Kombinationen (auch mit großer Überlappung) erreicht das Lernen mit TF/IDF Daten 100% precision und recall.

**Navigation im Raum der Beispiele**

- Zunehmende Größe des Datensatzes zeitgestempelter Daten: Schnappschuss < Intervalle < Boolean < TF/IDF
- TF/IDF ist günstig für lokale Muster, wenn diese Redundanz, Heterogenität als Eigenschaft aufweisen.
- Berechnung des TCat Modells für gegebene Daten implementiert $\rightarrow$ Fehlerschranke angebbar.

**Was wissen Sie jetzt?**

- Lokale Muster haben manchmal die typische TCat-Struktur.
- Sie haben gesehen, wie manche zeitgestempelte Datenbanken in TCat-Modelle transformiert werden können.
- Die Lernbarkeit mit linearer SVM der so transformierten Daten können Sie ausrechnen.

## 5.10 Lösung des Optimierungsproblems mit SMO

**Optimierungsproblem der SVM**

Die Lösung $\vec{\alpha}^*$ des dualen Problems

$$L_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i \, , \, \vec{x}_j \right\rangle$$

muss die KKT-Bedingungen erfüllen, d.h. es gilt unter anderem

$$\alpha_i \left( y_i \left( \left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle + \beta_0 \right) - 1 \right) = 0 \; \forall \, i = 1, \ldots, N$$

$\vec{\alpha}^*$ enthält für jedes Beispiel $\vec{x}_i$ genau ein $\alpha_i$ mit

$$\begin{aligned} \alpha_i = 0 \quad &, \quad \text{falls } \vec{x}_i \text{ im richtigen Halbraum liegt} \\ \alpha_i > 0 \quad &, \quad \text{falls } \vec{x}_i \text{ auf der Hyperebene } H_1 \text{ oder } H_2 \text{ liegt} \end{aligned}$$

Ein Beispiel $\vec{x}_i$ mit $\alpha_i > 0$ heißt Stützvektor.

**Optimierungsproblem für weiche Trennung**

Sei $C \in \mathbb{R}$ mit $C > 0$ fest. Minimiere

$$||\vec{\beta}||^2 + C \sum_{i=1}^{N} \xi_i$$

unter den Nebenbedingungen

$$\begin{aligned} \left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle + \beta_0 &\geq +1 - \xi_i \quad \text{für } y_i = +1 \\ \left\langle \vec{x}_i \, , \, \vec{\beta} \right\rangle + \beta_0 &\leq -1 + \xi_i \quad \text{für } y_i = -1 \end{aligned}$$

**Optimierungsproblem zur Minimierung**

- Erst minimierten wir $\vec{\beta}$ (primales Problem), dann maximierten wir $\alpha$ (duales Problem), jetzt minimieren wir das duale Problem, indem wir alles mit $-1$ multiplizieren...
- Minimiere $L'_D(\alpha)$

  $$\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j K(x_i, x_j) \alpha_i \alpha_j - \sum_{i=1}^{m} \alpha_i$$

  unter den Nebenbedingungen $0 \leq \alpha_i \leq C$

  $$\sum_{i=1}^{m} y_i \alpha_i = 0$$

**Algorithmus?**

- Berechnen wir $L'_D(\alpha)$ durch Gradientensuche!
  - Naiver Ansatz berechnet Gradienten an einem Startpunkt und sucht in angegebener Richtung ... bis kleinster Wert gefunden ist. Dabei wird immer die Nebenbedingung eingehalten. Bei $m$ Beispielen hat $\alpha$ $m$ Komponenten, nach denen es optimiert werden muss. Alle Komponenten von $\alpha$ auf einmal optimieren? $m^2$ Terme! Zu aufwändig.
  - Eine Komponente von $\alpha$ ändern? Nebenbedingung verletzt.
  - Zwei Komponenten $\alpha_1$, $\alpha_2$ im Bereich $[0, C] \times [0, C]$ verändern!

**Sequential Minimal Optimization**

- Wir verändern $\alpha_1$, $\alpha_2$ , lassen alle anderen $\alpha_i$ fest. Die Nebenbedingung wird zu:

  $$\alpha_1 y_i + \alpha_2 y_2 = -\sum_{i=3}^{m} \alpha_i y_i$$

- Zulässige $\alpha_1$, $\alpha_2$ liegen im Bereich $[0, C] \times [0, C]$ auf der Geraden $W = \alpha_1 y_1 + \alpha_2 y_2$ äquivalent $\alpha_1 + s\alpha_2$ mit $s = \frac{y_2}{y_1}$

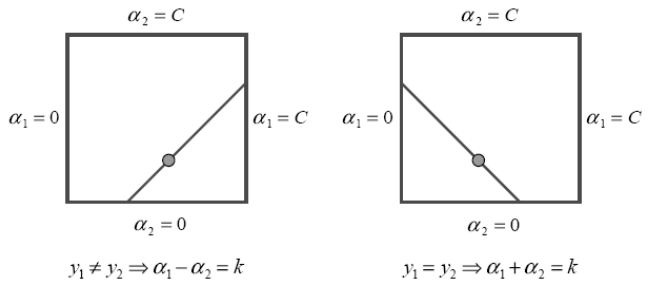
- Wir optimieren $\alpha_2$
- Aus dem optimalen $\hat{\alpha}_2$ können wir das optimale $\hat{\alpha}_1$ herleiten:

$$\hat{\alpha}_1 = \alpha_1 + y_1 y_2(\alpha_2 - \hat{\alpha}_2)$$

- Dann kommen die nächsten zwei $\alpha_i$ dran...

**Ermitteln der $\alpha s$ im Bild**

- Alle $\alpha s$ zu optimieren ist zu komplex.
- Nur ein $\alpha$ zur Zeit zu optimieren, verletzt $0 = \sum_{i=1}^{N} \alpha_i y_i$
- Also: zwei $\alpha s$ gleichzeitig optimieren!
- Man optimiert beide innerhalb eines Quadrates...




**$\alpha_2$ optimieren**

- Maximum der Funktion $L'_D(\alpha)$ entlang der Geraden $s\alpha_2 + \alpha_1 = d$.
- Wenn $y_1 = y_2$ ist $s = 1$, also steigt die Gerade. Sonst $s = -1$, also fällt die Gerade.
- Schnittpunkte der Geraden mit dem Bereich $[0, C] \times [0, C]$:
  - Falls s steigt: $\max(0; \alpha_2 + \alpha_1 - C)$ und $\min(C; \alpha_2 + \alpha_1)$
  - Sonst: $\max(0; \alpha_2 - \alpha_1)$ und $\min(C; \alpha_2 - \alpha_1 + C)$
  - Optimales $\alpha_2$ ist höchstens $\max$-Term, mindestens $\min$-Term.

**Bestimmen der $\alpha s$**

- $k = \alpha_1^{old} + s\alpha_2^{old} = \alpha_1^{new} + s\alpha_2^{new}$
- Mit Hilfe der Optimierungsquadrate lassen sich untere und obere Schranken für $\alpha_2$ bestimmen:
  - $y_1 = y_2 : L = max(0, \alpha_1^{old} + \alpha_2^{old} - C)\ H = min(C, \alpha_1^{old} + \alpha_2^{old})$
  - $y_1 \neq y_2 : L = max(0, \alpha_2^{old} - \alpha_1^{old})\ H = min(C, C + \alpha_2^{old} - \alpha_1^{old})$

- Ableiten des Dualen Problems nach $\alpha_2$ ergibt das Optimum für $\alpha_2^{new}$
  - $\alpha_2^{new} = \alpha_2^{old} + \frac{y_2((f(\vec{x_1})-y_1)-(f(\vec{x_2})-y_2))}{\eta}$
  - $= \alpha_2^{old} + \frac{y_2(E_1-E_2)}{\eta}$
  - $\eta = x_1^T x_1 + x_2^T x_2 - 2x_1^T x_2$
  -

**Optimales $\alpha_2$**

- Sei $\alpha = (\alpha_1, \ldots, \alpha_N)$ eine Lösung des Optimierungsproblems. Wir wählen zum update:

$$\hat{\alpha}_2 = \alpha_2 + \frac{y_2\left((f(x_1) - y_1) - (f(x_2) - y_2)\right)}{K(x_1, x_1) - 2K(x_1, x_2) + K(x_2, x_2)}$$

- Optimales $\hat{\alpha}_1 = \alpha_1 + y_1 y_2 \left(\alpha_2 - \hat{\alpha}_2\right)$
- Prinzip des Optimierens: Nullsetzen der ersten Ableitung...

**SMO und mehr**

- SMO hat Laufzeit $O(N^{2,2})$.
- Der SMO nimmt jeweils ein Paar $\alpha_1, \alpha_2$.
- Man kann aber auch gleich eine Menge von $\alpha_i$ für die Bearbeitung auswählen.

  Eigentlich soll man die gesamte Kernmatrix für die Optimierung verwenden, man kann aber eine Auswahl von Beispielen, die die KKT-Bedingungen verletzen, in einen *working set* aufnehmen.
- Die Kalkulationen in diesem working set werden gespeichert (caching).
- Beispiele, die nicht mehr die KKT-Bedingunge verletzen, werden aus dem working set gelöscht (shrinking).
- Mit caching und shrinking und Heuristiken hat Thorsten Joachims die erste effiziente SVM implementiert, SVM_light.

**Optimierungsalgorithmus**

1: g = Gradient von $L'_D(\alpha)$
2: WHILE nicht konvergiert($g$)
3: $WS$=working set($g$)
4: $\alpha'$=optimiere($WS$)
5: g=aktualisiere($g, \alpha'$)

1: $g_i = \sum \alpha_k y_k y_i \langle x_k\,,\, x_i \rangle - 1$
2: auf $\epsilon$ genau
3: suche $k$ “gute” Variablen
4: $k$ neue $\alpha$-Werte (update)
5: $g$ = Gradient von $L'_D(\alpha')$

- Gradientensuchverfahren
- Stützvektoren allein definieren die Lösung
- Tricks: Caching von $\langle x_i\,,\,x_j\rangle$
- Wir arbeiten nicht über der gesamten $N \times N$ Kernmatrix, sondern nur über dem working set.

**Was wissen wir jetzt?**

- Der SMO-Algorithmus ist *einer* der Optimierungsalgorithmen für das duale Problem.
- Man kann auch z.B. per Evolutionsalgorithmus optimieren (Mierswa 2006).
- Oder die lineare SVM mit der *cutting plane* Methode bearbeiten (Kelley 1960) (Joachims 2006)
- ...

# Kapitel 6

# Erweiterungen der SVM

## 6.1 Minimal Enclosing Ball, Core Vector Machine, Ball Vector Machine

**SVM mal anders**

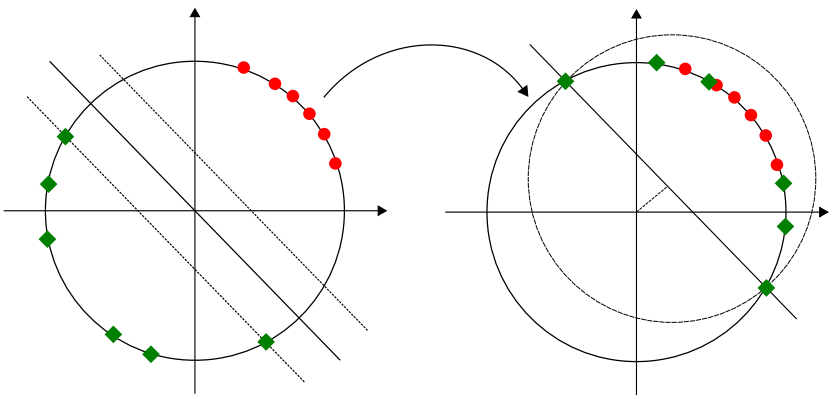
**Semi-überwachtes Lernen** : Nicht alle Beispiele haben ein label $y_i$, aber einige. Schon die Lage, wo es Beobachtungen gibt (und wo nicht) ist informativ.

**Transduktion** : Nicht alle Beispiele haben ein label $y_i$, aber alle $\vec{x_i}$ sind gegeben. Die Klassifikationsaufgabe ist vereinfacht: sie muss nur für die gegebenen Beobachtungen gut sein. Nützlich z.B. bei großen Bestandssammlungen.

**1-Klassen-SVM** : Hyperebene, die alle Beispiele mit maximalem Abstand vom Ursprung trennt. Gibt einen Eindruck von der Verteilung der Daten.

**SVDD zu Daten-Beschreibung** : Beschreibung der Daten durch eine sie umfassende Kugel. Nützlich zur Ausreißererkennung.

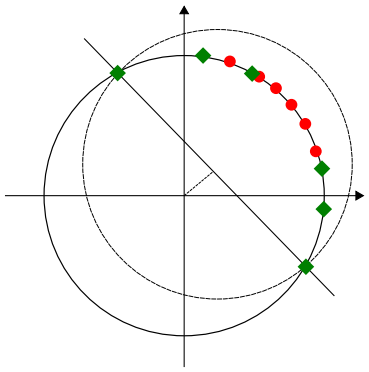
**Anschaulich: 2-Klassen-SVM und 1-Klassen-SVM**



Hyperebene durch den Ursprung – projizieren auf eine Seite – max. Abstand vom Ursprung.

**1-Klassen-SVM und SVDD – anschaulich**

Wenn die Kern-Funktion $K(x,x) = \kappa, \forall x \in S$ konstant ist, werden die Daten aus $S$ auf einen Ball $K_S$ abgebildet. Dann sind die SVDD und die 1-Klassen-SVM gleich.



So beim RBF-Kern, $K(\vec{x},\vec{x}) = \exp(-\gamma\|\vec{x}-\vec{x}\|^2) = 1, \forall x \in S$. Bei einem Poly-Kern ist dies nicht der Fall. Da muss erst *normalisiert* werden: $K(x,y) = \frac{K(x,y)}{\sqrt{K(x,x)}\sqrt{K(y,y)}}$

**Beispielrechnung**

$$\begin{aligned} K_2(\vec{x_i},\vec{x_i}) &= \langle \vec{x_i}\,,\,\vec{x_i}\rangle^2 \\ &= x_{i1}^2 x_{i1}^2 + 2x_{i1}x_{i1}x_{i2}x_{i2} + x_{i2}^2 x_{i2}^2 \\ \langle (2,2)\,,\,(2,2)\rangle^2 &= 2^2\cdot 2^2 + 2^5 + 2^2\cdot 2^2 = 64 \\ normal(\langle (2,2)\,,\,(2,2)\rangle^2) &= \frac{64}{8\cdot 8} = 1 \end{aligned}$$

Allgemein:

$$\begin{aligned} \langle x\,,\,y\rangle &= ||x||\cdot||y|| cos\angle(x,y) \\ \frac{\langle x\,,\,y\rangle}{\sqrt{\langle x\,,\,x\rangle}\sqrt{\langle y\,,\,y\rangle}} &= \frac{||x||\cdot||y||}{||x||\cdot||y||} cos\angle(x,y) \\ \frac{\langle x\,,\,x\rangle}{\sqrt{\langle x\,,\,x\rangle}\sqrt{\langle x\,,\,x\rangle}} &= cos\angle(x,x) = 1 \end{aligned}$$

**1-Klassen-SVM – formal**

Die 1-Klassen-SVM schiebt die Hyperebene so, dass +1 für (fast) alle Beispiele gilt.

$\rho$ gibt den Abstand zum Ursprung an. Da es in der Zielfunktion aufgeführt ist, wird es (mit) optimiert.

**Primales Problem der 1-Klassen-SVM**

$$\min_{\beta,\rho,\xi} = \| \vec{\beta} \|^2 - 2\rho + C\sum \xi^2$$

so dass $\vec{\beta}'\phi(\vec{x_i}) \geq \rho - \xi, \forall i$,
wobei $\rho = \vec{\beta}'\phi(\vec{x_i})$

**Merkmalsabbildung $\phi$**

- Der Abstand $\rho$ zum Ursprung bezieht sich auf den Merkmalsraum, der durch $\phi$ aufgespannt wird.
- Die Kernfunktion $k(\vec{x_i}, \vec{x_j})$ ist bei der 1-Klassen-SVM erweitert zu $\tilde{k}(\vec{x_i}, \vec{x_j})$, so dass

$$\forall i = 1, ..., N : \tilde{k}(\vec{x_i}, \vec{x_i}) = \kappa + \frac{1}{C} = \tilde{\kappa}$$

- Entsprechend $\left\langle \tilde{\phi}(\vec{x_i}) \, , \, \tilde{\phi}(\vec{x_j}) \right\rangle = \tilde{k}(\vec{x_i}, \vec{x_j})$ mit nicht-linearer Abbildung

$$\tilde{\phi}(\vec{x}) = \begin{bmatrix} \phi(\vec{x}) \\ \frac{1}{\sqrt{C}}\vec{e_i} \end{bmatrix}$$

  wobei $\vec{e_i}$ nur an $i$-ter Stelle eine 1, sonst 0 hat.

**SVDD – formal**

**Primales Problem der SVDD**

$$\min_{R,c,\xi} = R^2 + C\sum_{i=1}^{N} \xi_i$$

so dass $\| \vec{x_i} - c \|^2 \leq R^2 + \xi_i, \xi_{\geq} 0, \forall i$

**Duales Problem der SVDD**

$$\max_{\alpha} = \sum_{i=1}^{N} \alpha_i k(\vec{x_i}, \vec{x_i}) - \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j k(\vec{x_i}, \vec{x_j}) \text{ so dass}$$
$$\alpha_i \geq 0, i = 1, ..., N$$
$$\sum \alpha_i = 1$$

**SVM Betrachtung als Hyperkugel**

Die 1-Klassen-SVM und die SVDD betrachten die Daten in einer umschließenden Kugel. Wenn wir diese Kugel effizient berechnen, lösen wir damit auch das Optimierungsproblem von 1-Klassen-SVM und SVDD und sogar das der 2-Klassen-SVM.

Wir lernen hier zwei Methoden kennen:

- Core Vector Machine
- Ball Vector Machine

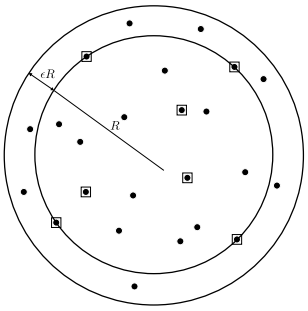
Aber zunächst das Ausgangsproblem: Minimum Enclosing Ball.

**Minimum Enclosing Ball**

**MEB Problem**

Gegeben eine Menge $S = \{x_1, ..., x_N\}$, finde den *minimum enclosing ball* $MEB(S) = B(c, R)$, der alle Daten enthält, mit $c$ Zentrum und $R$ minimalem Radius.

$B(c, (1+\epsilon)R)$ ist eine $1+\epsilon$-Approximation, $\epsilon > 0$, wenn $R \leq MEB(S), S \subset B(c, (1+\epsilon)R)$.




Bernd Gärtner (1999) Fast and Robust Smallest Enclosing Balls, in: Europ. Symposion on Algorithms

**Core Sets, Algorithmus – Skizze**

**Core Set**

Eine Teilmenge aller Beispiele $Q \subseteq S$ heißt *Core Set* mit $MEB(Q) = B(c, r)$, wenn $S$ in dem um $\epsilon$ erweiterten Ball enthalten ist, $S \subset B(c, (1+\epsilon)R)$.

In jeder Iteration wird ein Punkt dem Core Set hinzugefügt, der am weitesten vom Zentrum entfernt ist, bis kein Punkt aus S mehr außerhalb $B(c, (1+\epsilon)R)$ liegt.

Tsang, Kwok, Cheung (2005) Core Vector Machines: Fast SVM Training on Very Large Data Sets, JMLR 6 (http://jmlr.csail.mit.edu/)

**Core Vector Machine**

Die SVM wird stets approximativ gelöst (working set). Ein *Core Set* ist eine Menge, mit der man ein fast so gutes Ergebnis erzielt wie mit allen Beispielen. Die Core Vector Machine nutzt diese Approximation, um z.B. Intrusion Detection über 5.000.000 Beispielen in 1,4 Sekunden zu lernen!

Wir wollen das SVM-Problem über das MEB-Problem lösen. Wir wollen Kernfunktionen einbeziehen. Diese werden normalisiert (s.o. Beispielrechnung). Kernmethode, als MEB formuliert

- Transformierter Kern-Funktion $\tilde{k}$
- Merkmalsraum $\tilde{\mathbb{F}}$ mit Featuremap $\tilde{\varphi}$
- Konstante $\tilde{\kappa} = \tilde{k}(z, z)$

**Core Vector Machine - Algorithmus**

**Core Vector Machine Algorithmus**

1. Initialisiere $S_0, c_0$ und $R_0$.
2. Terminiere, wenn es keinen Trainingspunkt $z$ mehr gibt, der mit $\tilde{\varphi}(z)$ außerhalb der Kugel $B(c_t, (1+\epsilon)R)$ liegt.

3. Finde $z$, so dass $\tilde{\varphi}(z)$ am weitesten entfernt von $c_t$ ist. Setze $S_{t+1} = S_t \cup z$.
4. Berechne den neuen $MEB(S_{t+1})$.
5. Setze $t = t + 1$ und gehe zu (2).

**Schritt 1: Initalisierung**

- Starte mit einem zufälligen Punkt $z \in S$ und finde den von $z$ entferntesten Punkt $z_a \in S$.
- Finde einen anderen von $z_a$ entferntesten Punkt $z_b \in S$.
- CoreSet $S_0 := \{z_a, z_b\}$.
- Mittelpunkt $c_0 = \frac{1}{2}(\tilde{\varphi}(z_a) + \tilde{\varphi}(z_b))$
- Radius $R_0 = \frac{1}{2}\,||\tilde{\varphi}(z_a) - \tilde{\varphi}(z_b)|| = \frac{1}{2}\sqrt{2\tilde{\kappa} - 2\tilde{k}(z_a, z_b)}$
- Bei der binären Klassifikation sollten $z_a$ und $z_b$ aus verschiedenen Klassen kommen.

**Schritt 2, 3: Abstandsberechnung**

Der Abstand $||c_t - \tilde{\varphi}(z_l)||$ muss in Schritt 2 (Terminierung) und Schritt 3 (Finden eines entferntesten Punktes) berechnet werden.

$$||\vec{c_t} - \tilde{\varphi}(z_l)||^2 = \sum_{z_i, z_j \in S_t} \alpha_i \alpha_j \tilde{k}(z_i, z_j) - 2 \sum_{z_i \in S_t} \alpha_i \tilde{k}(z_i, z_l) + \tilde{k}(z_l, z_l)$$

Random Sampling $S' \subset S$, so dass der Abstand zum Mittelpunkt nur für $S'$ berechnet wird, beschleunigt die Laufzeit auf $O(|S_t|^2)$ in der $t$-ten Iteration.

**Schritt 4: MEB berechnen**

Wir haben in Iteration $t$ den Core Set $S_t$ und ermitteln aus diesem den $MEB(S_t)$.

- Die Core Vektoren $S_t$ erfüllen die (weichen) KKT-Bedingungen.
- Die Nicht-Core-Vektoren sind innerhalb der Kugel, $\alpha = 0$, also gelten die (weichen) KKT-Bedingungen.
- Wenn die CVM terminiert, erfüllen alle Beispiele die KKT-Bedingungen mit nur minimaler Abweichung.

Wir lösen das duale Problem über dieser kleineren Menge von Beispielen mit SMO.

Mit den Core Vektoren bestimmen wir den working set.

Bei kleinem $\epsilon$ wird nähert sich die Lösung der exakten, dafür dauert es länger.

Laufzeit: $\frac{1}{\epsilon^4}$

**Core Vector Machine**

Die Core Vector Machine löst das Optimierungsproblem auf dem Core Set in jeder Iteration (Schritt 4):

$$\max_{\alpha} = -\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j k(\vec{x_i}, \vec{x_j}) \text{ so dass}$$
$$\alpha_i \geq 0, i = 1, ..., N$$
$$\sum \alpha_i = 1$$

Das Lösen von Optimierungsproblemen ist aufwändig.

Bei großen Datenmengen kann auch das Core Set groß werden.

Eine Heuristik kann vielleicht ähnlich gute Ergebnisse liefern?

**Ball Vector Machine**

Heuristischer Ansatz vereinfacht das MEB Problem durch einen festen Radius:

**Enclosing Ball Problem**

Gegeben der Radius $r > R^*$,

finde den Ball B(c, r), der alle Punkte $\phi(x) \in S$ umschließt:

$$||c - \phi(\vec{x}_i)||^2 \leq r^2$$

- Dazu gibt es einen Approximationsalgorithmus, der ausnutzt, dass der Radius fest ist. Welche Punkte liegen außerhalb von $r(1 + \epsilon)$?
- Der Radius wird gewählt als $\sqrt{\tilde{\kappa}}$.
- Der Ball wird in jeder Iteration verschoben. Das Zentrum wird in jeder Iteration neu berechnet: minimiere $\|c - c_t\|^2$ so dass $r^2 \geq \|c - \phi(x_t)\|^2$

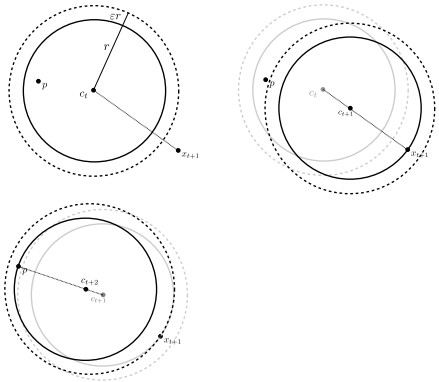
**Algorithmus anschaulich**

Man findet einen Punkt außerhalb, verschiebt das Zentrum,...

Möglicherweise oszilliert der Prozess und die Lösung verbessert sich nicht.

**Was wissen Sie jetzt?**

- Sie kennen die Aufgabenstellungen:
  - Semi-überwachtes Lernen
  - Transduktion
  - 1-Klassen-SVM
  - SVDD zu Daten-Beschreibun
- Sie haben eine Vorstellung, wie man SVM-Probleme als Minimum Enclosing Ball Problem sehen kann.



- Sie kennen die Methode, den working set einer SVM durch den Core Set zu bestimmen.
- Sie kennen die Heuristik, mit festem Radius das SVM-Problem durch das MEB Problem zu lösen.

## 6.2 Überblick Lernaufgaben

**Jenseits des Bag of Words**

- Bisher haben wir Texte als Anzahl und Häufigkeit von Wörtern repräsentiert.
- Damit haben wir die Struktur der Sprache ignoriert.
  - Grammatik
  - Koreferenz
  - Eigennamen
  - Semantische Relationen
- Es gibt eine ganze Reihe von Ansätzen des maschinellen Lernens, um (sprachliche) Strukturen zu behandeln.
- Wir besprechen hier nur die SVM bezogenen Ansätze.

**Lernaufgabe Named Entity Recognition**

- Wortfolgen, die sich auf ein individuelles Objekt beziehen, werden *Named Entities* (NE) genannt.
- Eigennamen, Ortsnamen, Firmennamen sind z.B. NEs.
- Gegeben Beispiele von Sätzen, in denen NEs annotiert sind, lerne die Entscheidungsfunktion, die für jedes Wort angibt, ob es zu einer bestimmten NE gehört, oder nicht.
- Beispiel:

| Johann | Sebastian | Bach | publiziert | im | Henle | Verlag | München. |
|---|---|---|---|---|---|---|---|
| Per | Per | Per | 0 | 0 | Org | Org | Place |

**Anwendungen**

Wenn wir in Dokumenten die NEs automatisch annotieren,

- können wir sie im Text markieren, so dass die Benutzer schneller interessante Stellen auffinden;
- können wir alle Sätze zu einer Person, Firma, einem Ort herausschreiben und so eine Zusammenfassung für einen Text erstellen;
- eine weitere Lernaufgabe aufsetzen: *Relationen* zwischen NEs lernen, z.B. Fusion von Firmen $fusion(Org_1, Org_2)$, Mitglied im Aufsichtsrat $aufsicht(Org, Per)$.

Letztlich erstellen wir eine Datenbank aus einer Dokumentensammlung. Auf diese Datenbank wenden wir dann unsere Lernverfahren wie gehabt an.

**Part of Speech Tagging**

- Unter *Part-of-speech Tagging* versteht man die Zuordnung von Wörtern eines Textes zu Wortarten (engl.: part of speech).
- Beispiel:

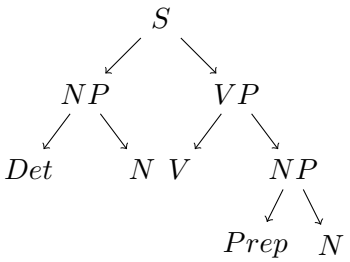
| Die | Noten | erscheinen | bei | Henle. |
|---|---|---|---|---|
| Det | N | V | Prep | N |

**Full Parsing**

- Syntaxregeln produzieren einen Syntaxbaum für einen Satz, dessen Wurzel das Startsymbol $S$ ist und die Blätter sind die Wortarten (präterminalen Knoten), denen dann die Wörter zugeordnet sind. *Full Parsing* erstellt Syntaxbäume für Sätze.
- Beispiel:

  $x$: Die Noten erscheinen bei Henle.

  $y$:




$S \rightarrow NP, VP$
$VP \rightarrow V, NP$
$NP \rightarrow Det, N$
$NP \rightarrow Prep, N$

**Lernaufgaben Part of Speech Tagging, Full Parsing**

- *Part of Speech Tagging*: Gegeben eine Menge von Sätzen, bei denen zu jedem Wort die Wortart angegeben ist, lerne eine Entscheidungsfunktion, die bei beliebigen Sätzen jedem Wort eine Wortart zuordnet.
- *Full Parsing Learning*: Gegeben eine Menge von Sätzen, eine Menge von Syntaxregeln und die Anzahl von Regelanwendungen für jeden Satz, lerne eine Entscheidungsfunktion, die beliebigen Sätzen die Anzahl von Regelanwendungen zuordnet (discriminant model).

- *Zur Abgrenzung*: Es sind

  Gegeben eine Menge von syntaktisch korrekten Sätzen (positive Beispiele) und eine Menge von synaktisch falschen Sätzen (negative Sätze), bei denen jeweils die Wortarten annotiert sind, lerne Syntaxregeln, die gerade die syntaktisch korrekten Sätze produzieren (generative model).

**Support Vector Machines für alles...**

Bisher haben wir zwei Lernaufgaben für *Support Vector Machines* betrachtet:

- Binäre Klassifikation
- SVM zur Regression

Aktuelles Übungsblatt: *Mehrklassen-Problem*

**Jetzt:** Lernen von Funktionen für beliebige strukturelle Ausgaben (Bäume, Graphen) mit Hilfe der *SVMstruct*.

**SVMstruct**

**Strukturelle Modelle**

Sei $X$ die Menge der Beispiele. Ist die Ausgabe-Variable $Y$ nicht ein Skalar, sondern eine Struktur (z.B. eine Folge, ein Baum, ein Graph), so heißt das Modell

$$f : X \to Y$$

*strukturelles Modell*.

- *Large Margin Methods for Structured and Interdependent Output Variables*, I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, J. of Machine Learning Research, Vol. 6, p. 1453 – 1484, 2005
- *Training Linear SVMs in Linear Time*, Thorsten Joachims, Proc. KDD 2006

**Lernaufgabe der SVMstruct**

**Lernaufgabe SVMstruct**

Sei $Y = Y_1 \times \ldots \times Y_q$ eine Menge und $\vec{y} \in Y$ eine *Konfiguration* in $Y$ und

$$\mathcal{T} = \{(\vec{x}_1, \vec{y}_1), \ldots, (\vec{x}_n, \vec{y}_n)\} \subset X \times Y$$

eine Menge von Beobachtungen.

**Ziel:** Finde eine Funktion $f$ mit

$$f : X \to Y$$

**SVMstruct**

Es sei eine $\vec{\beta}$-parametrisierte Schar von Funktionen

$$F_{\vec{\beta}} : X \times Y \to \mathbb{R}$$

gegeben, die die Ähnlichkeit zwischen $\vec{x}$ und $\vec{y}$ ausdrücken.

Gesucht ist nun ein $\vec{\beta}^*$ derart, daß

$$f(x, \vec{\beta}^*) = \arg\max_{\vec{y} \in Y} F_{\vec{\beta}^*}(\vec{x}, \vec{y})$$

jeweils zu einem $\vec{x}$, das am besten passende $\vec{y} \in Y$ liefert.

**Hinweis:** In der Literatur wird für $F_{\vec{\beta}}(\vec{x}, \vec{y})$ meist $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ geschrieben.

Wir brauchen also eine Art Kostenfunktion, die uns die Ähnlichkeit zwischen $\vec{x}$ und $\vec{y}$ liefert. Sei dazu $\Psi$ eine Abbildung

$$\Psi : X \times Y \to \mathcal{F}$$

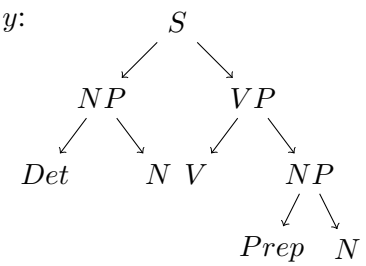
von $(\vec{x}, \vec{y})$ auf eine gemeinsame Repräsentation $\Psi(\vec{x}, \vec{y})$.

Die Abbildung $\Psi$ hängt dabei vom konkreten Problem ab, eine Darstellung für Parse-Trees liefert z.B.

$x$: Die Noten erscheinen bei Henle.

$f : X \to Y$

$y$:



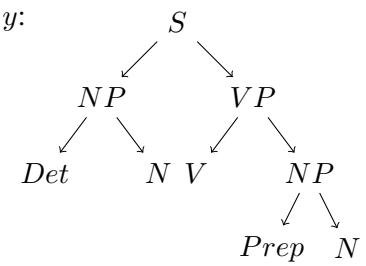

$$\Psi(\vec{x}, \vec{y}) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \quad \begin{matrix} \text{S} \to \text{NP, VP} \\ \text{NP} \to \text{Det, N} \\ \text{NP} \to \text{Prep, N} \\ \text{NP} \to \text{Adj, N} \\ \vdots \\ \text{Prep} \to \text{in} \\ \text{Prep} \to \text{bei} \end{matrix}$$

**Merkmalsabbildung am Beispiel der ParseTrees**

$x$: Die Noten erscheinen bei Henle.

$f : X \to Y$

$y$:




$$\Psi(\vec{x}, \vec{y}) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \quad \begin{matrix} \text{S} \to \text{NP, VP} \\ \text{NP} \to \text{Det, N} \\ \text{NP} \to \text{Prep, N} \\ \text{NP} \to \text{Adj, N} \\ \vdots \\ \text{Prep} \to \text{in} \\ \text{Prep} \to \text{bei} \end{matrix}$$

Wir nehmen im Folgenden an, daß $F$ linear in $\Psi(\vec{x}, \vec{y})$ ist, d.h.

$$F_{\vec{\beta}}(\vec{x}, \vec{y}) = \left\langle \vec{\beta}\,,\, \Psi(\vec{x}, \vec{y}) \right\rangle$$

Wir lernen also über input-/output-Kombinationen die Ranking-Funktion

$$F : X \times Y \to \mathbb{R}.$$

## 6.3 Primales Problem

**Was hat das mit SVM zu tun?**

Wo ist jetzt die SVM-Idee hier?

Mit der Funktion $F_{\vec{\beta}^*}$ haben wir eine Ordnung auf den Paaren $(\vec{x}, \vec{y})$, so daß wir jedem neuen $\vec{x}'$ ein $\vec{y}'$ zuordnen können:

$$f(\vec{x}', \vec{\beta}^*) = \vec{y}' = \arg\max_{\vec{y} \in Y} F_{\vec{\beta}^*}(\vec{x}', \vec{y})$$

Wir wählen also immer das am besten bzgl. $F_{\vec{\beta}^*}$ passende $\vec{y}$!

Lernen von $\vec{\beta}^*$ über Optimierung mit der SVM: *Maximiere den Abstand (Margin) zwischen der besten und der zweit-besten Lösung!*

**Primales Problem**

Wir suchen also eine Hyperebene, die das Beste von allen anderen Beispielen trennt.

Dazu soll $f(\vec{x}_i, \vec{\beta})$ für $\vec{x}_i \in \mathcal{T}$ jeweils das "richtige $\vec{y}_i$" vorhersagen, d.h.

$$\begin{aligned} & \forall i \ : \ \max_{\vec{y} \in Y \setminus \vec{y}_i} \left\{ \left\langle \vec{\beta}, \ \Psi(\vec{x}_i, \vec{y}) \right\rangle \right\} < \left\langle \vec{\beta}, \ \Psi(\vec{x}_i, \vec{y}_i) \right\rangle \\ \Rightarrow \quad & \forall i, \forall \vec{y} \in Y \setminus \vec{y}_i \ : \ \left\langle \vec{\beta}, \ \Psi(\vec{x}_i, \vec{y}) \right\rangle < \left\langle \vec{\beta}, \ \Psi(\vec{x}_i, \vec{y}_i) \right\rangle \\ \Rightarrow \quad & \forall i, \forall \vec{y} \in Y \setminus \vec{y}_i \ : \ \left\langle \vec{\beta}, \ \Psi(\vec{x}_i, \vec{y}_i) - \Psi(\vec{x}_i, \vec{y}) \right\rangle > 0 \end{aligned}$$

**Primales Problem**

Sind die Nebenbedingungen

$$\forall i, \forall \vec{y} \in Y \setminus \vec{y}_i \ : \ \left\langle \vec{\beta}, \ \Psi(\vec{x}_i, \vec{y}_i) - \Psi(\vec{x}_i, \vec{y}) \right\rangle > 0$$

erfüllbar, führt dies typischerweise zu einer Menge optimaler Lösungen.

Eine eindeutige Lösung läßt sich jetzt finden, indem man das *Maximum Margin*-Prinzip der SVM nutzt und $\vec{\beta}$ mit $||\vec{\beta}|| \leq 1$ so wählt, dass der Abstand zum nächstbesten $\vec{\beta}'$ maximal wird, also

$$\hat{\vec{y}}_i = \arg\max_{\vec{y} \neq \vec{y}_i} \left\langle \vec{\beta}, \ \Psi(\vec{x}_i, \vec{y}) \right\rangle$$

**Primales Problem**

Schließlich führt das zum primalen Problem der *SVMstruct*:

**Primales Problem**

Minimiere

$$L_P(\vec{\beta}) = \frac{1}{2}||\vec{\beta}|| \tag{6.1}$$

unter den Nebenbedingungen

$$\forall i, \forall \vec{y} \in Y \setminus \vec{y}_i \ : \ \left\langle \vec{\beta}, \ \delta\Psi_i(\vec{y}_i) \right\rangle \geq 1 \tag{6.2}$$

mit $\delta\Psi_i(\vec{y}) = \Psi(\vec{x}_i, \vec{y}_i) - \Psi(\vec{x}_i, \vec{y})$

Die SVMstruct optimiert also unter Beachtung von $n|Y| - n$ Nebenbedingungen.

**SVMstruct mit Ausnahmen – slack rescaling**

Ein Strafterm $C$ für alle Beispiele $\vec{x}$, bei denen die Nebenbedingungen verletzt sind, und die Relaxierung durch $\xi$ führt zu dem Minimierungsproblem:

**Slack Rescaling SVM**

$$SVM_1 : \quad \min_{\vec{\beta},\xi} \ \frac{1}{2}\|\vec{\beta}\|^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i \tag{6.3}$$

unter den Bedingungen

$$\forall i, \forall \vec{y} \in Y \backslash \vec{y_i} : \langle \vec{\beta}, \delta\Psi_i(\vec{y}) \rangle \geq 1 - \xi_i, \xi_i \geq 0 \tag{6.4}$$

$C$ ist linear in den $\xi_i$.

**SVMstruct mit Ausnahmen – margin rescaling**

Verletzungen der Nebenbedingungen können auch durch einen quadratischen Term bestraft werden.

**Margin rescaling SVM**

$$SVM_2 : \quad \min_{\vec{\beta},\xi} \ \frac{1}{2}\|\vec{\beta}\|^2 + \frac{C}{2N}\sum_{i=1}^{N}\xi_i^2 \tag{6.5}$$

unter den Bedingungen

$$\forall i, \forall \vec{y} \in Y \backslash \vec{y_i} : \langle \vec{\beta}, \delta\Psi_i(\vec{y}) \rangle \geq 1 - \xi_i, \xi_i \geq 0 \tag{6.6}$$

**Empirisches Risiko und Slack Rescaling**

Auch bei strukturellen Modellen geht es darum, den Fehler zu minimieren. Der erwartete Fehler bei irgend einer Verlustfunktion $\Delta$ ist für eine Menge von Beispielen $\mathcal{T}$

$$R_{\mathcal{T}}(f) = \frac{1}{2}\sum_{i=1}^{N}\Delta(\vec{y_i}, f(\vec{x_i})) \tag{6.7}$$

Um die Verletzung der Nebenbedingung für ein $\vec{y} \neq \vec{y_i}$ bei großem Verlust $\Delta(\vec{y_i}, \vec{y})$ stärker zu bestrafen als bei geringerem, passen wir die Nebenbedingungen an:

$$SVM_1 : \quad \min_{\vec{\beta},\xi} \ \frac{1}{2}\|\vec{\beta}\|^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i$$

$$\forall i, \forall \vec{y} \in Y \backslash \vec{y_i} : \langle \vec{\beta}, \delta\Psi_i(\vec{y}) \rangle \geq 1 - \frac{\xi_i}{\Delta(\vec{y_i}, \vec{y})} \tag{6.8}$$

$$SVM_2 : \quad \min_{\vec{\beta},\xi} \ \frac{1}{2}\|\vec{\beta}\|^2 + \frac{C}{2N}\sum_{i=1}^{N}\xi_i^2$$

$$\forall i, \forall \vec{y} \in Y \backslash \vec{y_i} : \langle \vec{\beta}, \delta\Psi_i(\vec{y}) \rangle \geq 1 - \frac{\xi_i}{\sqrt{\Delta(\vec{y_i}, \vec{y})}} \tag{6.9}$$

**Obere Schranke des empirischen Fehlers**

**Satz**

Seien $\xi_i^*(\vec{\beta})$ die optimalen Schlupfvariablen für ein gegebenes $\vec{\beta}$, dann ist

$$R_{\mathcal{T}}(\vec{\beta}) \leq \frac{1}{N} \sum_{i=1}^{N} \xi_i^*$$

**Obere Schranke des empirischen Fehlers**

**Beweis:**

Wir wissen:

$$\xi_i^* = \max \left\{ 0, \max_{\vec{y} \neq \vec{y_i}} \left\{ \Delta(\vec{y_i}, \vec{y}) \left[ 1 - \langle \vec{\beta}, \delta\Psi_i(\vec{y}) \rangle \right] \right\} \right\}$$

Sei $\vec{y}^* = f(\vec{x}_i, \vec{\beta})$, es ergeben sich zwei Fälle

$$\begin{array}{lll} \text{1. Fall: } \vec{y}^* = \vec{y} & : & \text{Es ist } \Delta(\vec{y_i}, f(\vec{x}_i, \vec{\beta})) = 0 \leq \xi_i^* \\ \text{2. Fall: } \vec{y}^* \neq \vec{y} & : & \Rightarrow \langle \vec{y_i}\,,\, \delta\Psi_i(\vec{y}^*) \rangle \leq 0 \\ & & \Rightarrow \frac{\xi_i^*}{\Delta(\vec{y_i}, \vec{y})} \geq 1 \Leftrightarrow \Delta(\vec{y_i}, \vec{y}) \leq \xi_i^* \end{array}$$

Da die Schranke für jedes Beispiel gilt, gilt sie auch für den Durchschnitt. □

## 6.4 Duales Problem

**Hinführung zum dualen Problem**

- Wir wollen auch hier das duale Problem formulieren.
- Bisher war es:

$$L_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \vec{x}_i\,,\, \vec{x}_j \rangle$$

- Jetzt sind $\vec{y_i}, \vec{y_j}$ nicht mehr einfach Werte, sondern Strukturen.
- Für jedes der $N$ Beispiele $\vec{x_i}$ mit jedem der $\mid Y \mid$ möglichen $\vec{y_i}$ müssen wir feststellen, wie groß der Abstand zu allen anderen $\Psi(\vec{x_i}, \vec{y})$ ist.
- Wir haben also nicht mehr ein $\alpha$ je Beispiel in $X$, sondern ein $\alpha$ für jedes Paar in $X \times Y$.
- Erst sehen wir uns den Raum an, in dem optimiert wird, dann die $\alpha$.

**Raum, in dem SVMstruct optimiert**

- Bei der klassischen SVM haben wir beim dualen Problem für $i, j = 1, ..., N$ das Skalarprodukt $\langle \vec{x_i}, \vec{x_j} \rangle$ gerechnet.
- Jetzt müssen wir für $i, j = 1, ..., N$ rechnen $J_{(i\vec{y})(j\vec{y'})} \equiv \langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$.

| | $\vec{x_1}$ | ... | $\vec{x_j}$ | ... | $\vec{x_N}$ |
|---|---|---|---|---|---|
| $\vec{x_1}$ | $-$ | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ... |
| $\vec{x_i}$ | ... | ... | $\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$ | ... | ... |
| ... | ... | ... | ... | ... | ... |
| $\vec{x_N}$ | ... | ... | ... | ... | $-$ |

- Dabei ist $\langle \delta\Psi_j(\vec{y}), \delta\Psi_i(\vec{y'}) \rangle$ wieder eine Matrix!

**Matrix J**

- In der $N \times N$ Matrix sind die Einträge $\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$.
- Wir erinnern uns: $\delta\Psi_i(\vec{y}) \equiv \Psi(\vec{x_i}, \vec{y_i}) - \Psi(\vec{x_i}, \vec{y})$
- Statt einer einfachen Matrix haben wir einen Tensor, d.h. der Eintrag in die $N \times N$ Matrix ist eine $\mid Y \mid \times \mid Y \mid$ Matrix **J**:

| | $\vec{y_1}$ | ... | $y_{\vec{\mid Y \mid}}$ |
|---|---|---|---|
| $\vec{y_1}$ | $-$ | ... | $\Psi(\vec{x}, \vec{y_1}) - \Psi(\vec{x}, y_{\vec{\mid Y \mid}})$ |
| ... | ... | ... | ... |
| $y_{\vec{\mid Y \mid}}$ | $\Psi(\vec{x}, y_{\vec{\mid Y \mid}}) - \Psi(\vec{x}, \vec{y_1})$ | ... | $-$ |

- **J** ist eine Kernfunktion über $X \times Y$: $J_{(i\vec{y})(j\vec{y'})} = \langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$

**$\alpha_{i\vec{y}}$ und optimales $\beta$**

- Statt $\alpha_i$ für $\vec{x_i}$, haben wir $\alpha_{ij}$ mit $j = 1, ..., \mid Y \mid$

$$\begin{pmatrix} \alpha_{i1} \\ ... \\ \alpha_{im} \end{pmatrix}$$

- Das optimale $\vec{\beta}$ ist

$$\begin{aligned} \hat{\vec{\beta}} &= \sum_{i=1}^{N} \sum_{\vec{y} \neq \vec{y_i}}^{|Y|} \alpha_{(i\vec{y})} (\Psi(\vec{x_i}, \vec{y_i}) - \Psi(\vec{x_i}, \vec{y})) \\ &= \sum_{i=1}^{N} \sum_{\vec{y} \neq \vec{y_i}}^{|Y|} \alpha_{(i\vec{y})} \delta\Psi_i(\vec{y}) \end{aligned} \tag{6.10}$$

**Duales Problem der SVMstruct**

- SVMstruct bei linear separierbaren Beispielen:

$$L_D(\alpha) = -\frac{1}{2}\sum_{i,\vec{y}\neq\vec{y}_i}^{N}\sum_{j,\vec{y'}\neq\vec{y}_i}^{N}\alpha_{i\vec{y}}\alpha_{j,\vec{y'}}J_{(i\vec{y})(j\vec{y'})} + \sum_{i,\vec{y}\neq\vec{y}_i}^{N}\alpha_{i\vec{y}} \tag{6.11}$$

- Für die Slack Rescaling $SVM_1$ mit Ausnahmen muss zusätzlich gelten:

$$\sum_{\vec{y}\neq\vec{y}_i}^{N}\alpha_{i\vec{y}} \leq \frac{C}{N}, \forall i = 1, ..., N$$

- Für die Margin Rescaling $SVM_2$ mit Ausnahmen wird $J_{(i\vec{y})(j\vec{y'})}$ unter Verwendung der Indikatorfunktion $I(a, b) = 1$ falls $a = b$, sonst 0 zu:

$$\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'})\rangle + I(i, j)\frac{N}{C}$$

- Immer soll $\vec{\alpha}$ maximiert werden.

## 6.5 Optimierung der SVMstruct

**Die SVMstruct stellt ein schwieriges Optimierungsproblem!**

- Bei $N \mid Y \mid -N$ Nebenbedingungen und vermutlich sehr großem $\mid Y \mid$ ist normale Optimierung durch quadratische Programmierung nicht möglich.
- Es sollen nun deutlich weniger Nebenbedingungen wirklich bearbeitet werden.
- Beobachtung: Es gibt immer eine **Teilmenge** von Nebenbedingungen, so dass die damit errechnete Lösung auch **alle** Nebenbedingungen erfüllt mit einer Ungenauigkeit von nur $\epsilon$.

**SVMstruct: Algorithmus zum Optimieren – Idee**

- Für jedes Beispiel $\vec{x_i}$ gibt es einen working set $S_i$, in dem die verletzten Nebenbedingungen gespeichert sind. Zunächst sind $S_i$ leer, das Problem unbeschränkt.
- Für jedes Beispiel $\vec{x_i}$ wird die am schlimmsten verletzte Nebenbedingung bzgl. $\vec{y_i^*}$ festgestellt und $S_i$ hinzugefügt. Das Problem wird zunehmend stärker beschränkt.
- Optimiere $\alpha$
  - bezüglich aller working sets gemeinsam oder
  - nur für ein $S_i$, wobei die $\alpha_{j\vec{y}}$ mit $j \neq i$ eingefroren werden.
- Wenn kein $S_i$ mehr verändert wurde, STOP.

**SVMstruct: Algorithmus zum Optimieren**

1. Input: $\mathcal{T} = \{(\vec{x_1}, \vec{y_1}), ..., (\vec{x_N}, \vec{y_N})\}, C, \epsilon$
2. $S_i := \{\}$ für alle $i = 1, ..., N$
3. Solange ein $S_i$ sich in der Iteration ändert:
4. **for** $i = 1, ..., N$ **do**
5. Kosten: $H(\vec{y}) \begin{cases} 1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle & SVM_0 \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_1 \ (s.6.8) \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_2 \ (s.6.9) \end{cases}$ wobei $\vec{\beta} \equiv \sum_j \sum_{\vec{y'} \in S_j} \alpha_{j\vec{y'}} \delta\Psi_j(\vec{y'})$
6. $\vec{y^*} := \arg\max_{\vec{y} \in Y} H(\vec{y})$ – schwieriger Schritt!
7. $\xi_i := max\{0, max_{\vec{y} \in S_i} H(\vec{y})\}$
8. **if** $H(\vec{y^*}) > \xi_i + \epsilon$ **then**
9. $S_i := S_i \cup \{\vec{y^*}\}$
10. $\alpha_S :=$optimiere duales Problem für $S = \cup S_i$

## 6.6 Anwendungen

**Full Parsing Learning mit SVMstruct**

- Probabilistische kontextfreie Grammatik: Regeln $n_l[C_i \to C_j, C_k], \beta_l$. Dabei gibt $\beta_l$ die logarithmierte Wahrscheinlichkeit dafür an, dass ein Knoten $C_i$ mit Regel $n_l$ expandiert wird.
- Lernaufgabe: Gegeben Paare $(\vec{x}, \vec{y})$, wobei $\vec{x} = x_1, ..., x_p$ ein Satz (Kette von Wortarten) ist und $\vec{y}$ ein Syntaxbaum, lerne $X \to Y$, wobei nur in den Beispielen vorkommende Regeln verwendet werden.
- Formuliert als Maximierungsproblem:

$$h(\vec{x}) = \arg\max_{\vec{y} \in Y} P(\vec{y}|\vec{x}) = argmax_{\vec{y} \in Y} \left\{ \sum_{n_l \in rules(\vec{y})} \beta_l \right\}$$

  $rules(\vec{y})$ ist die Menge der Regeln, die in $\vec{y}$ verwendet sind.

**Full Parsing Learning mit SVMstruct**

- Es ergibt sich: $\langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \rangle = \sum_{n_l \in rules(\vec{y})} \beta_l$
- Den schwierigen Schritt $\vec{y^*} := argmax_{\vec{y} \in Y} \langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \rangle$ löst nun ein Parser, der sowohl den besten als auch den zweitbesten Syntaxbaum für $\vec{x}$ liefert. Somit können die *Beispiele* bei der Optimierung (Schritt 6) effizient bearbeitet werden.
- Das Lernergebnis ordnet *bisher nicht gesehenen Sätzen* $\vec{x}$ die richtigen Syntaxbäume zu. Dabei erweitert es die Fähigkeit der Grammatik – nicht die Menge der Regeln.

**Experiment**

- Trainingsmenge: 4098 Sätze mit maximal $p = 10$ Wörtern (dargestellt durch ihre Wortart)
- Testmenge: 163 Sätze mit maximal $p = 10$
- Maximum likelihood zum Lernen ergibt: $86,8\%$ precision, $85,2\%$ recall, $86\% F_1$ measure
- $SVM_2$ mit slack rescaling ergibt: $88,9\%$ precision, $88,1\%$ recall, $88,5\% F_1$ measure
- Der Unterschied des F-measures ist signifikant.
- $SVM_2$ hat in 12 Iterationen insgesamt 8043 Nebenbedingungen behandelt.
- Das Lernen dauerte insgesamt 3,4 Stunden, wovon die $SVM_2$ $10,5\%$ verwendete.

**Andere Anwendungen der SVMstruct**

- Wenn man die SVMstruct anwenden will, muss man
  - die Merkmalsabbildung $\Psi(\vec{x}, \vec{y})$ definieren und ggf. implementieren
  - die Verlustfunktion implementieren $\Delta(\vec{y_i}, \vec{y})$
  - die Selektion verletzter Bedingungen (Schritt 6 des Algorithmus') implementieren.
- Klassifikation mit Taxonomien
- Named Entity Recognition
- Mehrklassen-Klassifikation
- ...

**Was wissen Sie jetzt?**

- Sie wissen, was strukturelle Modelle sind: $Y$ kann mehr sein als nur ein Wert.
- $\Psi(\vec{x}, \vec{y})$ erweitert die üblichen Beispiele so, dass nun wieder ein Skalarprodukt $\langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \rangle$ gerechnet werden kann.
- Das Problem sind die $N \times \mid Y \mid -N$ Nebenbedingungen, weil wir jedes Beispiel mit jedem anderen nicht nur bezüglich eines $y$, sondern bezüglich der $\mid Y \mid$ möglichen $\vec{y}$ vergleichen müssen.
- Dabei wird dieser Vergleich als *joint kernel* aufgefasst: $\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$. Es gibt noch viele andere Arbeiten zu *string kernels, tree kernels*, die Sie hier nicht kennen gelernt haben.

**Sie wissen noch mehr!**

- Der Ansatz von Joachims besteht darin,
  - dass als margin der Abstand zwischen der besten und der zweitbesten Lösung maximiert wird,
  - dass nur wenige der Nebenbedingungen wirklich behandelt werden müssen,
  - dass beliebige Verlustfunktionen $\Delta$ in den Nebenbedingungen und in der Auswahl der am stärksten verletzten Nebenbedingung verwendet werden können.

# Kapitel 7

# Additive Modelle

**Ausgangspunkt: Funktionsapproximation**

- Die bisher vorgestellten Lernverfahren, sind Instanzen der Funktionsapproximation.
- Gegeben sind die Trainingsbeispiele $\mathcal{T}$, gesucht ist eine Funktion

$$f_\theta(x) = \sum_{m=1}^{M} h_m(x)\theta_m$$

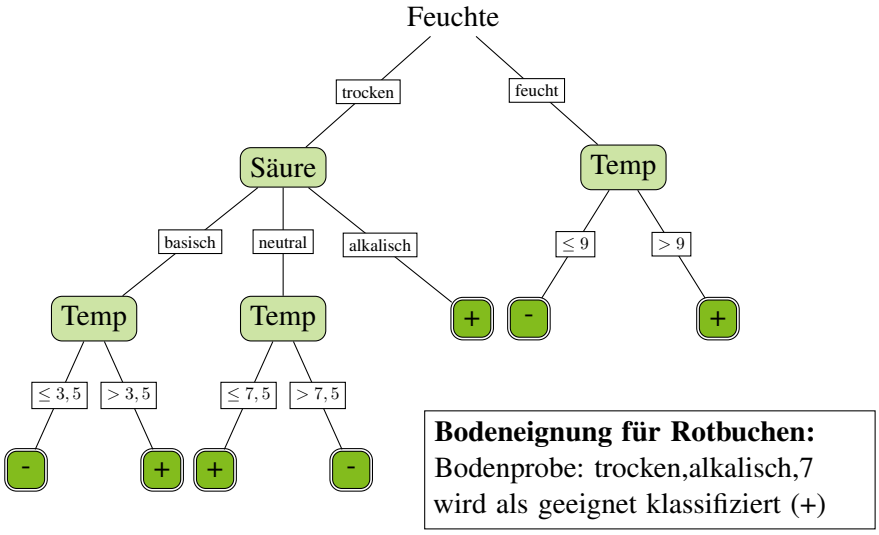
- Dabei gibt es Parameter $\theta$, die abzuschätzen sind, bei den linearen Modellen ist dies $\hat{\beta}$.
- $h_m(x)$ ist eine Transformation der Beispiele.

## 7.1 Baumlerner

**Aufteilen der Beispiele und Modellierung jeder Region**

- Lineare Modelle passen die Parameter für den gesamten Raum der Beispiele an, der evtl. durch eine implizite Transformation (Kernfunktionen) oder explizite Transformationen (Vorverarbeitung) in einen Merkmalsraum überführt wurde.
- *Baumlerner* teilen den Merkmalsraum in Rechtecke auf und passen in jedem ein Modell an. Dabei wird die Wahl des Merkmals in der rekursiven Aufteilung automatisch bestimmt.
- kNN teilt den Raum der Beispiele bei einer Anfrage $x$ in die Nachbarschaft von $x$ und den Rest auf.

**Klassifizieren mit Entscheidungsbäumen**

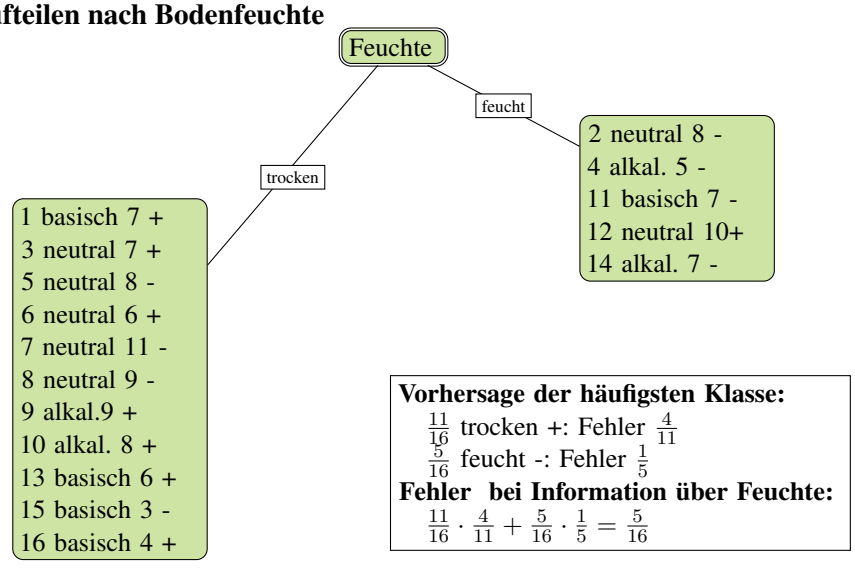



**Lernen aus Beispielen**

| + | | | | - | | | |
|---|---|---|---|---|---|---|---|
| ID | Feuchte | Säure | Temp | ID | Feuchte | Säure | Temp |
| 1 | trocken | basisch | 7 | 2 | feucht | neutral | 8 |
| 3 | trocken | neutral | 7 | 4 | feucht | alkal. | 5 |
| 6 | trocken | neutral | 6 | 5 | trocken | neutral | 8 |
| 9 | trocken | alkal. | 9 | 7 | trocken | neutral | 11 |
| 10 | trocken | alkal. | 8 | 8 | trocken | neutral | 9 |
| 12 | feucht | neutral | 10 | 11 | feucht | basisch | 7 |
| 13 | trocken | basisch | 6 | 14 | feucht | alkal. | 7 |
| 16 | trocken | basisch | 4 | 15 | trocken | basisch | 3 |

Ohne weiteres Wissen können wir als Vorhersage immer - sagen. Der Fehler ist dann 8/16.

**Aufteilen nach Bodenfeuchte**



### 7.1.1 Merkmalsauswahl

**Bedingte Wahrscheinlichkeit**

- Wahrscheinlichkeit, dass ein Beispiel zu einer Klasse gehört, gegeben der Merkmalswert
$$P(Y|X_j) = P(Y \cap X_j)/P(X_j)$$
- Annäherung der Wahrscheinlichkeit über die Häufigkeit
- Gewichtung bezüglich der Oberklasse
- Beispiel: $Y = \{+, -\}, X_j = \{feucht, trocken\}$

  $P(+|feucht) = 1/5,\ P(-|feucht) = 4/5$ gewichtet mit $5/16$

  $P(+|trocken) = 7/11,\ P(-|trocken) = 4/11$ gewichtet mit $11/16$

  Wahl des Merkmals mit dem höchsten Wert (kleinsten Fehler)

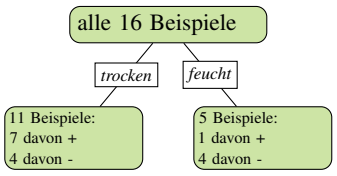
**Information eines Merkmals**

- Wir betrachten ein Merkmal als Information.
- Wahrscheinlichkeit $p_+$, dass das Beispiel der Klasse + entstammt. $I(p_+, p_-) = (-p_+ \log p_+) + (-p_- \log p_-)$ Entropie
- Ein Merkmal $X_j$ mit k Werten teilt eine Menge von Beispielen $\mathbf{X}$ in k Untermengen $\mathbf{X}_1, ..., \mathbf{X}_k$ auf. Für jede dieser Mengen berechnen wir die Entropie.
$$Information(X_j, \mathbf{X}) := -\sum_{i=1}^{k} \frac{|\mathbf{X}_i|}{|\mathbf{X}|} I(p_+, p_-)$$
- Der *Informationsgewinn* ist die Differenz zwischen der Entropie der Beispiele mit und ohne die Aufteilung durch $X_j$.

**Feuchte**

[T]

Güte des Attributs Feuchte mit den 2 Werten *trocken* und *feucht*:




$$= -\left[\underbrace{\frac{11}{16} \cdot I(+,-)}_{trocken} + \underbrace{\frac{5}{16} \cdot I(+,-)}_{feucht}\right]$$

$$= -\left[\underbrace{\frac{11}{16} \cdot \left(-\frac{7}{11} \cdot \log\left(\frac{7}{11}\right) - \frac{4}{11} \cdot \log\left(\frac{4}{11}\right)\right)}_{trocken} + \underbrace{\frac{5}{16}\left(-\frac{1}{5} \cdot \log\left(\frac{1}{5}\right) - \frac{4}{5} \cdot \log\left(\frac{4}{5}\right)\right)}_{feucht}\right] = -0,27$$

**Säure**
[T]

Güte des Attributs Säure mit den 3 Werten basisch, neutral und alkalisch:

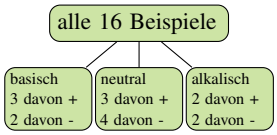
$$-\left(\underbrace{\frac{5}{16}\cdot I(+,-)}_{\text{basisch}}+\underbrace{\frac{7}{16}\cdot I(+,-)}_{\text{neutral}}+\underbrace{\frac{4}{16}\cdot I(+,-))}_{\text{alkalisch}}\right)$$

$$= -0,3$$

**basisch** $-\frac{3}{5}\cdot\log\left(\frac{3}{5}\right)+-\frac{2}{5}\cdot\log\left(\frac{2}{5}\right)$

**neutral** $-\frac{3}{7}\cdot\log\left(\frac{3}{7}\right)+-\frac{4}{7}\cdot\log\left(\frac{4}{7}\right)$

**alkalisch** $-\frac{2}{4}\cdot\log\left(\frac{2}{4}\right)+-\frac{2}{4}\cdot\log\left(\frac{2}{4}\right)$



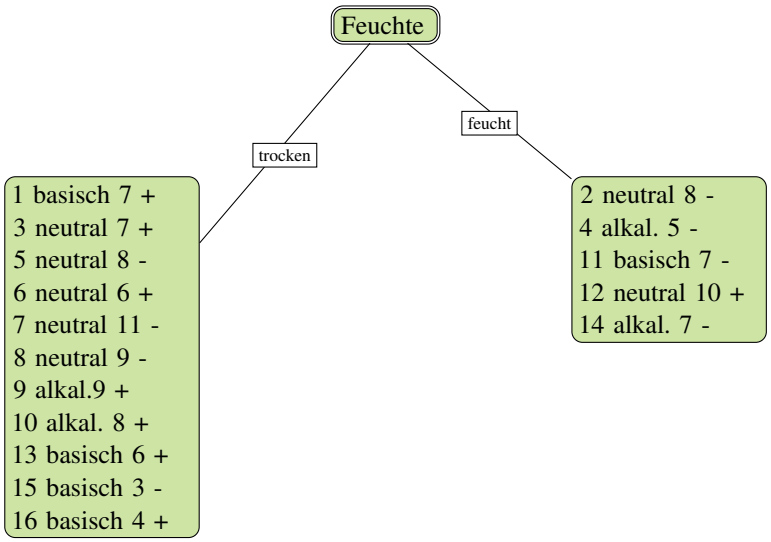

**Temperatur**

- Numerische Merkmalswerte werden nach Schwellwerten eingeteilt.
  - 9 verschiedene Werte in der Beispielmenge, also 8 Möglichkeiten zu trennen.
  - Wert mit der kleinsten Fehlerrate bei Vorhersage der Mehrheitsklasse liegt bei 7.
  - 5 Beispiele mit Temp < 7, davon 3 in +, 11 Beispiele Temp $\gtrsim$ 7, davon 6 in -.
- Die Güte der Temperatur als Merkmal ist $-0,29$.

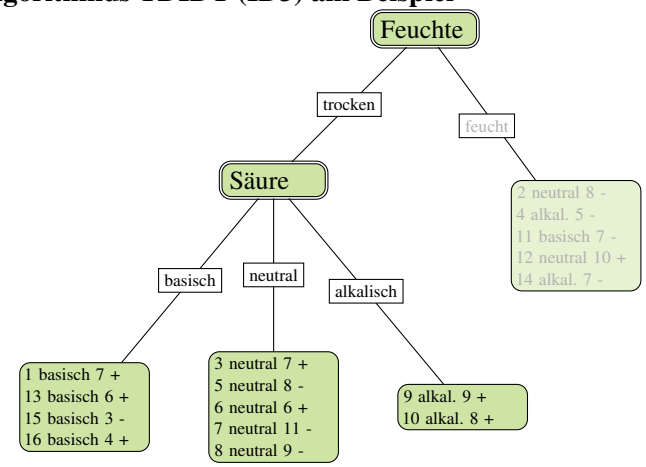
**Merkmalsauswahl**

- Gewählt wird das Merkmal $X_j$, dessen Werte am besten in (Unter-)mengen $\mathbf{X}_i$ aufteilen, die geordnet sind.
- Das Gütekriterium *Information* (Entropie) bestimmt die Ordnung der Mengen.
- Im Beispiel hat *Feuchte* den höchsten Gütewert.

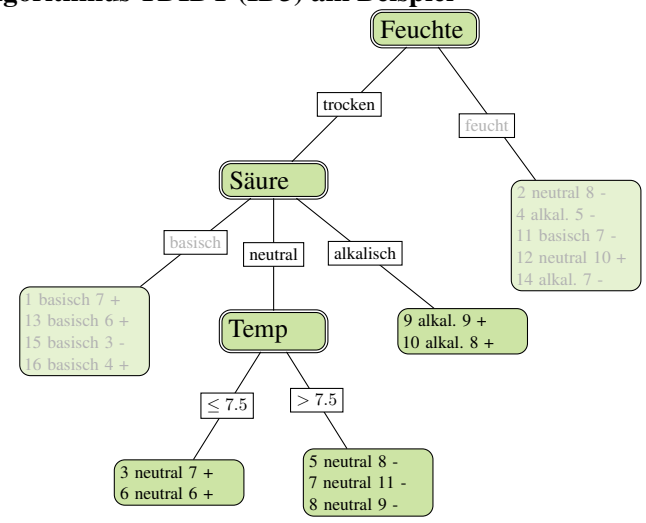
**Algorithmus Top Down Induction of Decision Trees (TDIDT, hier: ID3) am Beispiel**



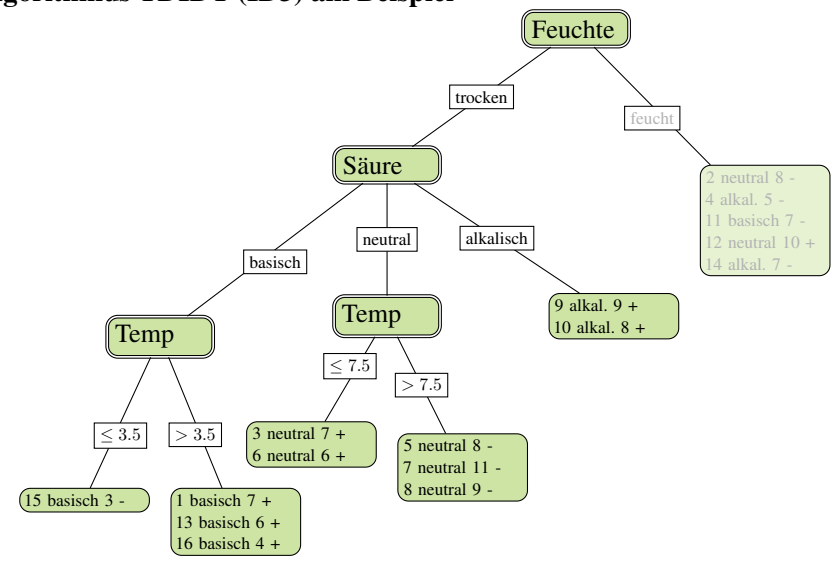

**Algorithmus TDIDT (ID3) am Beispiel**




**Algorithmus TDIDT (ID3) am Beispiel**



**Algorithmus TDIDT (ID3) am Beispiel**




**Algorithmus ID3 (TDIDT)**

Rekursive Aufteilung der Beispielmenge nach Merkmalsauswahl:

1. $TDIDT(\mathbf{X}, \{X_1, ...X_p\})$
2. $\mathbf{X}$ enthält nur Beispiele einer Klasse $\rightarrow$ fertig
3. $\mathbf{X}$ enthält Beispiele verschiedener Klassen:
   - $Güte(X_1, .., X_p, \mathbf{X})$
   - Wahl des besten Merkmals $X_j$ mit $k$ Werten
     - Aufteilung von $\mathbf{X}$ in $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_k$
     - für $i = 1, ..., k$: $TDIDT(\mathbf{X}_i, \{X_1, ..., X_p\} \backslash X_j)$
   - Resultat ist aktueller Knoten mit den Teilbäumen $T_1, ..., T_k$

**Komplexität TDIDT ohne Pruning**

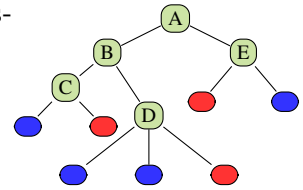
Rekursive Aufteilung der Beispielmenge nach Merkmalsauswahl:

- Bei $p$ (nicht-numerischen) Merkmalen und $N$ Beispielen ist die Komplexität $\mathcal{O}(pN \log N)$
  - Die Tiefe des Baums sei in $\mathcal{O}(\log N)$.
  - $\mathcal{O}(N \log N)$ alle Beispiele müssen "in die Tiefe verteilt" werden, also: $\mathcal{O}(N \log N)$ für ein Merkmal.
  - $p$ mal bei $p$ Merkmalen!

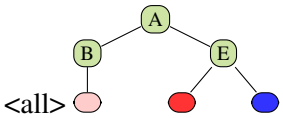
### 7.1.2 Gütemaße und Fehlerabschätzung

**Stutzen**
[T]

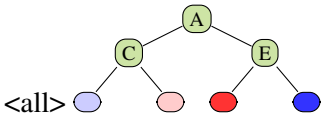
- Überanpassung des Baums an die Trainingsdaten verringern!
- Verständlichkeit erhöhen!
- Stutzen (Pruning):
  a) Knoten an Stelle eines Teilbaums setzen
  b) Einen Teilbaum eine Ebene höher ziehen
- Schätzen, wie sich der wahre Fehler beim Stutzen entwickelt.




a) Knoten an Stelle eines Teilbaums setzen




b) Einen Teilbaum eine Ebene höher ziehen




**Stutzen durch Fehlerschätzen**

- Wenn der Fehler eines Knotens kleiner ist als die Summe der Fehler seiner Unterknoten, können die Unterknoten weggestutzt werden.
- Dazu müssen wir (bottom-up) die Fehler an allen Knoten schätzen.
- Obendrein sollten wir berücksichtigen, wie genau unsere Schätzung ist. Dazu bestimmen wir ein Konfidenzintervall.
- Wenn die obere Schranke der Konfidenz in den Fehler beim oberen Knoten kleiner ist als bei allen Unterknoten zusammen, werden die Unterknoten gestutzt.

**Was ist ein Konfidenzintervall?**

**Konfidenzintervall**

Vorgegeben eine tolerierte Irrtumswahrscheinlichkeit $\alpha$, gibt das Konfidenzintervall

$$P(u \leq X \leq o) = 1 - \alpha$$

an, dass $X$ mit der Wahrscheinlichkeit $1 - \alpha$ im Intervall [u, o] liegt und mit der Wahrscheinlichkeit $\alpha$ nicht in [u, o] liegt.

Meist wird das Konfidenzintervall für den Erwartungswert gebildet. Beispiel $\alpha = 0,1$: Mit 90% iger Wahrscheinlichkeit liegt der Mittelwert $\bar{X}$ im Intervall [u, o], nur 10% der Beobachtungen liefern einen Wert außerhalb des Intervalls.

**z-Transformation in eine standard-normalverteilte Zufallsvariable**

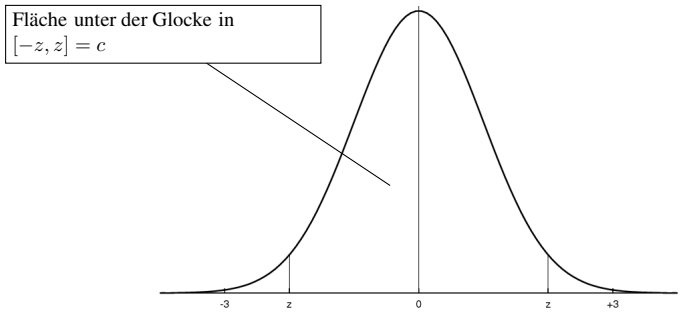
Die Zufallsvariable $X$ wird bezüglich ihres Mittelwerts $\bar{X}$ standardisiert unter der Annahme einer Normalverteilung:

$$Z = \frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{N}}} \sim \mathcal{N}(0;1)$$

Die Wahrscheinlichkeit dafür, dass der Mittelwert im Intervall liegt, ist nun:

$$P\left(-z\left(1-\frac{\alpha}{2}\right) \leq \frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{N}}} \leq z\left(1-\frac{\alpha}{2}\right)\right) = 1 - \alpha$$

**Verteilung mit z-Werten**




- $P(-z \leq X \leq z) = 1 - \alpha$ Konfidenzniveau Wahrscheinlichkeit, dass $X$ mit Mittelwert $0$ im Intervall der Breite $2z$ liegt ist $1 - \alpha$.
- $z$ kann nachgeschlagen werden (z.B. Bronstein), wobei wegen Symmetrie nur angegeben ist: $P(X \geq z)$

**Rechnung für reellwertige Beobachtungen und Mittelwert**

Wir wollen ein bestimmtes Konfidenzniveau erreichen, z.B. 0,8.

- $P(X \geq -z)$ $P(X \leq z)$ ist dann $(1 - 0,8)/2 = 0,1$.
- Der $z$-Wert, für den die Fläche der Glockenkurve zwischen $-z$ und $z$ genau $1 - \alpha = 0,8$ beträgt, ist das $(1 - \frac{\alpha}{2})$-Quantil der Standardnormalverteilung, hier: $1,28$ (nachschlagen).
- Das standardisierte Stichprobenmittel liegt mit der Wahrscheinlichkeit 0,8 zwischen -1,28 und +1,28.

$$\begin{aligned} 0,8 &= P(-1,28 \leq \frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{N}}} \leq 1,28) \\ &= P(-1,28\frac{\sigma}{\sqrt{N}} \leq \bar{X} - \mu \leq 1,28\frac{\sigma}{\sqrt{N}}) \\ &= P(\bar{X} - 1,28\frac{\sigma}{\sqrt{N}} \leq \mu \leq \bar{X} - 1,28\frac{\sigma}{\sqrt{N}}) \end{aligned}$$

Das Intervall ist $[\bar{X} - 1,28\frac{\sigma}{\sqrt{N}}; \bar{X} + 1,28\frac{\sigma}{\sqrt{N}}]$.

**Fehler oder Erfolg schätzen**

- Bei den Entscheidungsbäumen beobachten wir nur zwei Werte $Y \in \{+, -\}$.
- Wir haben eine Binomialverteilung mit wahrer Wahrscheinlichkeit $p_+$ für $y = +$ (Erfolg).
- Beobachtung der Häufigkeit $f_+$ bei $N$ Versuchen. Varianz:

$$\sigma^2 = \frac{f_+(1 - f_+)}{N}$$

  Erwartungswert:

$$E(p_+) = f_+/N$$

- In das allgemeine Konfidenzintervall $[\bar{X} - z(1-\alpha/2)\frac{\sigma}{\sqrt{N}}; \bar{X} + 1,28\frac{\sigma}{\sqrt{N}}]$ setzen wir diese Varianz ein und erhalten:

$$\left[ f_+ - z(1-\alpha/2)\frac{\sqrt{f_+(1-f_+)}}{N}; f_+ z(1-\alpha/2)\frac{\sqrt{f_+(1-f_+)}}{N} \right]$$

**Konfidenz bei Binomialverteilung**

Allgemein berechnet man die obere und untere Schranke der Konfidenz bei einer Binomialverteilung für ein Bernoulli-Experiment:

$$p_+ = \frac{f_+ + \frac{z^2}{2N} \pm z\sqrt{\frac{f_+}{N} - \frac{f_+^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}}$$

Hierzu muss lediglich die Häufigkeit $f_+$ gezählt werden, $N, z$ bekannt sein.
Diese Abschätzung für den Erfolg können wir symmetrisch für den Fehler ($p_-$) durchführen.

**Anwendung zum Stutzen**

- Für jeden Knoten nehmen wir die obere Schranke (pessimistisch):

$$p_- = \frac{f_- + \frac{z^2}{2N} + z\sqrt{\frac{f_-}{N} - \frac{f_-^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}}$$

- Wenn der Schätzfehler eines Knotens kleiner ist als die Kombination der Schätzfehler seiner Unterknoten, werden die Unterknoten weggestutzt. Die Kombination wird gewichtet mit der Anzahl der subsumierten Beispiele.

**Gütemaße**

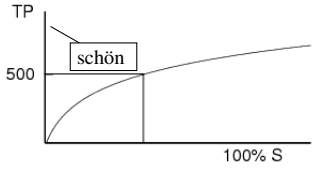
- Konfusionsmatrix:

| tatsächlich | Vorhergesagt $+$ | Vorhergesagt $-$ | |
|---|---|---|---|
| $+$ | True positives $TP$ | False negatives $FN$ | Recall: $TP/(TP+FN)$ |
| $-$ | False positives $FP$ | True negatives $TN$ | |
| | Precision: $TP/(TP+FP)$ | | |

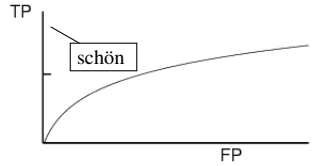
- Accuracy: $P(\hat{f}(x) = y)$ geschätzt als $(TP+TN)/total$

**Balance von FP und FN**

- F-measure: $\frac{\beta \cdot recall \cdot precision}{recall+precision} = \frac{\beta TP}{\beta TP+FP+FN}$
- Verlaufsformen:
  - Lift: $TP$ für verschiedene Stichprobengrößen $S$

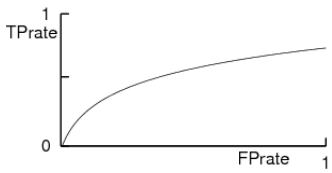



  - Receiver Operating Characteristic (ROC): für verschiedene $TP$ jeweils die $FP$ anzeigen




**ROC genauer**

- Statt der absoluten Anzahl $TP$ nimm die Raten von true oder false positives – ergibt eine glatte Kurve.
  - Für jeden Prozentsatz von falschen Positiven nimm eine Hypothese $h$, deren Extension diese Anzahl von $FP$ hat und zähle die $TP$.
  - $TP_{rate} := TP/P \sim recall$ bezogen auf eine Untermenge
  - $FP_{rate} := FP/N \sim FP/FP+TN$ bezogen auf Untermenge




**Kosten von Fehlern**

- Nicht immer sind FP so schlimm wie FN
  - medizinische Anwendungen: lieber ein Alarm zu viel als einen zu wenig!
- Gewichtung der Beispiele:
  - Wenn FN 3x so schlimm ist wie FP, dann gewichte negative Beispiele 3x höher als positive.
  - Wenn FP 10x so schlimm ist wie FN, dann gewichte positive Beispiele 10x höher als negative.
- Lerne den Klassifikator mit den gewichteten Beispielen wie üblich. So kann jeder Lerner Kosten berücksichtigen!

**Was wissen Sie jetzt?**

- Sie kennen den Algorithmus ID3 als Beispiel für TDIDT.
- Für das Lernen verwendet ID3 das Gütemaß des Informationsgewinns auf Basis der Entropie.
- Man kann abschätzen, wie nah das Lernergebnis der unbekannten Wahrheit kommt $\rightarrow$ Konfidenz
- Man kann abschätzen, wie groß der Fehler sein wird und dies zum Stutzen des gelernten Baums nutzen.
- Lernergebnisse werden evaluiert:
  - Einzelwerte: accuracy, precision, recall, F-measure
  - Verläufe: Lift, ROC

  Diese Evaluationsmethoden gelten nicht nur für Entscheidungsbäume!

# Kapitel 8

# Graphische Modelle

## 8.1 Einführung

**Graphische Modelle**
Lernen mit einer Struktur:

- Structural SVM
- Hidden Markov Models (HMM)
- Conditional Random Fields (CRF)
- Bayes Networks

Raum-zeitliche Verhältnisse

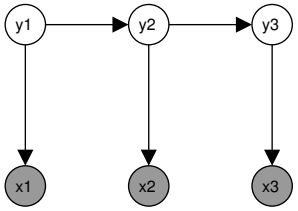
- Sprache
- Handlungen
- Bilder
- Geodaten

## 8.2 HMM

**Hidden Markov Models**
$\lambda = (Y, A, B, \pi, X)$

- Struktur gegeben durch einen Graph möglicher Zustände $y_1, ..., y_i, ...$(versteckte Knoten) $Y$
- Matrix $A$ der Übergangswahrscheinlichkeit von einem Zustand zum nächsten (Transition Features): $a_{ij} = p(y_i|y_j)$
- Wahrscheinlichkeit $B$, im Zustand $y_i$ die Beobachtung $x_j$ zu machen (State Features) $b_{ij} = p(y_i|x_j)$
- Anfangswahrscheinlichkeit, dass $y_i$ der Startzustand ist: $\pi(i) = p(y_1 = y_i)$
- Menge der möglichen Beobachtungen (beobachtete Knoten) $X$

**HMM Modell**




- Eine Beobachtung $x_t$ hängt nur vom Zustand $y_t$ ab: $p(x_t|y_t)$
- Ein Zustand $y_t$ hängt nur vom Vorgängerzustand ab: $p(y_t|y_{t-1})$
- Die Anfangswahrscheinlichkeit schreiben wir als $p(y_1|y_0)$, so dass es von $p(y_t|y_{t-1})$ mit abgedeckt ist.
- Die Wahrscheinlichkeit für die Zustandssequenz $\vec{y}$ und die Beobachtungssequenz $\vec{x}$ ist

$$p(\vec{y}, \vec{x}) = \prod_{t=1}^{T} p(y_t|y_{t-1})p(x_t|y_t)$$

**Aufgaben für Algorithmen**

- Lernen der Wahrscheinlichkeiten $a_{ij}, b_{ij}, \pi(i)$ (Training Problem)
- Annotieren einer Beobachtungssequenz durch das (gelernte) Modell (Decoding problem)

In der Literatur gibt es noch die Evaluierungsaufgabe, die die Wahrscheinlichkeit dafür angibt, dass eine bestimmte Beobachtungssequenz durch eine gegebene Sequenz versteckter Zustände zustande kommt [11]. Wir besprechen hier zwei Algorithmen, die immer wieder vorkommen:

- Forward Backward
- Viterbi

**Einführung in den Forward Backward Algorithmus**

Es gibt eine Illustration des Algorithmus mit Excel-Berechnungen [5], die in den Algorithmus einführen soll.

**Eiscreme und Klima (Eisner)**

Im Jahr 2799 will eine Forschergruppe den Klimawandel untersuchen. Es werden keine Aufzeichnungen des Wetters in Baltimore gefunden, aber das Tagebuch von Jason Eisner über 33 aufeinander folgende Tage im Jahr 2001, in dem er notiert hat, wie viele Kugeln Eis er aß: $X = \{1, 2, 3\}$.

Gesucht ist für diese Sequenz von Tagen jeweils die verborgene Temperatur, hier nur kalt oder heiß: $Y = \{C, H\}$ .

**Wahrscheinlichkeiten**

State

- Wahrscheinlichkeit an einem kalten Tag für eine Kugel $p(1|C) = 0,7$ und zwei $p(2|C) = 0,2$ und drei $p(3|C) = 0,1$.
- Wahrscheinlichkeit an einem heißen Tag
  $p(1|H) = 0,1$ und $p(2|H) = 0,2$ und $p(3|H) = 0,7$.

Transition

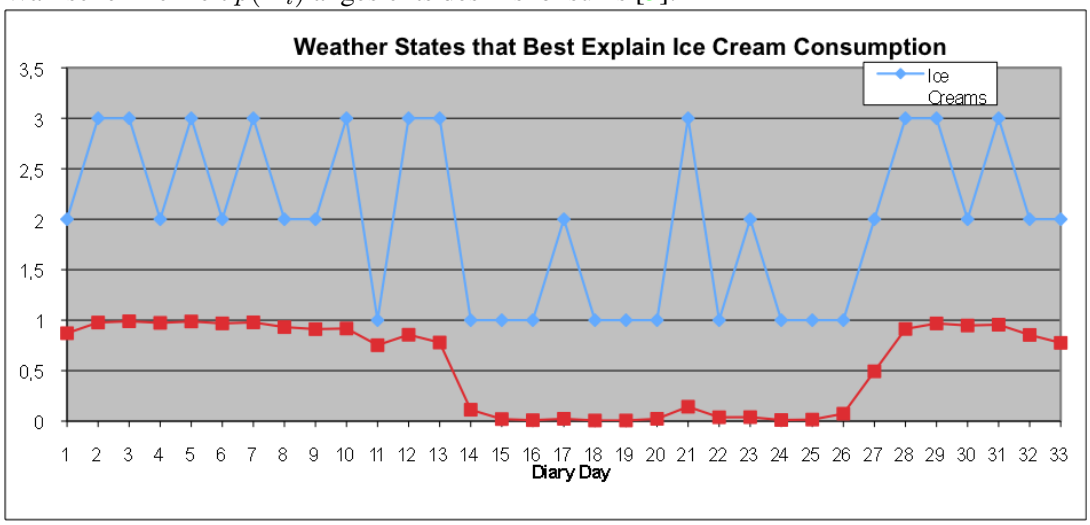
- Wahrscheinlichkeit von heiß nach kalt $p(C_{i+1}|H_i) = 0,1$ und von kalt nach kalt $p(C_{i+1}|C_i) = 0,8$
- Wahrscheinlichkeit von kalt nach heiß $p(H_{i+1}|C_i) = 0,1$ und von heiß nach heiß $p(H_{i+1}|H_i) = 0,8$

Boundary

- Wahrscheinlichkeit, das Tagebuch anzufangen/zu beenden an einem kalten/heißen Tag $p(C|start) = p(H|start) = 0,5$ und $p(stop|C) = p(stop|H) = 0,5$

**Wetterrekonstruktion**

Wahrscheinlichkeit $p(H_i)$ angesichts des Eiskonsums [5]:




**Rechnen der Wahrscheinlichkeit von Pfaden**

- Wahrscheinlichkeit für $start, H, H, C$ und $2, 3, 3$ Produkt von Übergangswahrscheinlichkeit und Zustandswahrscheinlichkeit für jeden Übergang $(0,5 \cdot 0,2) \cdot (0,8 \cdot 0,7) \cdot (0,1 \cdot 0,1) = 0,1 \cdot 0,56 \cdot 0,01 = 0,00056$
- Welche der 8 Zustandsfolgen ist die wahrscheinlichste bei der Beobachtung $2, 3, 3$? $P(start, H, H, H) = 0,1 \cdot 0,56 \cdot 0,56 = 0,0314$ – diese! $P(start, H, C, H) = 0,1 \cdot 0,01 \cdot 0,07 = 0,00007$ $P(start, C, H, H) = 0,1 \cdot 0,07 \cdot 0,56 = 0,0039$ $P(start, C, C, H) = 0,1 \cdot 0,08 \cdot 0,007 = 0,000056$
- $\alpha$: Summe der Wahrscheinlichkeiten aller Pfade von $start$ bis $t$, z.B. $y_t = H$

**Forward Variable $\alpha$**

Pfad vom Anfang bis Position $T$, wo der Zustand irgendein festes $y_i$ ist:

$$\alpha_t(i) = P(x_1, ..., x_T, Y_t = y_i|\lambda)$$

- Anfang: $\pi$ mal Wahrscheinlichkeit für erste Beobachtung

$$\alpha_1(i) = P(x_1, y_i|\lambda) = \pi_i b_i(x_1)$$

- Rekursion für $1 < t \leq T$:

$$\alpha_t(j) = P(x_1, ..., x_t, y_j|\lambda) = \sum_{i=1}^{|Y|} \alpha_{t-1}(i) a_{ij} b_j(x_t)$$

- Ende:

$$P(x_1, ..., x_t, y_i|\lambda) = \sum_{i=1}^{|Y|} \alpha_n(i)$$

**Rechnen des Beispiels**

Eisners Excel-Tabelle hat die Wahrscheinlichkeiten in den Matritzen A (Übergang) und B (Zustand) sowie Randverteilung bereits gegeben.

Wir schauen uns die Propagierung einmal an.

Danach betrachten wir CRFs und den Forward Backward Algorithmus für CRFs.

**Was wissen Sie jetzt?**

- Sie haben eine Formulierung für Sequenzen kennengelernt: HMM.
- Wir sind ein Beispiel durchgegangen, bei dem alleWahrscheinlichkeiten schon gegeben waren.
- Sie wissen noch nicht, wie man diese Wahrscheinlichkeiten aus Daten heraus schätzt!

## 8.3 CRF

**Conditional Random Fields**

CRF=(X,Y,A,B)

- Die Struktur ist (wie bei HMMs) durch die multivariaten Zufallsvariablen $X$ und $Y$ gegeben, aber ungerichtet.
- Zustandsübergänge $a_{i,j,k}$ in $A$ als *transition features* gewichtet: $\lambda_k f_k(y_{t-1}, y_t, x_t)$ wobei $\lambda \in [0, 1]$, $y_{t-1} = y_i, y_t = y_j, x_t = x_k$.
- Zustandsmerkmale $b_{i,k}$ in $B$ als *state features* gewichtet: $\lambda f_k(y_t, x_t)$ wobei $y_t = y_i, x_t = x_k$.
- Gewichtung von Start- und Endzuständen für $y_0 = start$ bzw. $y_{T+1} = stop$ sind ebenfalls in $A$ und $B$ enthalten.
- Wir vereinigen die Merkmale und erhalten den Gewichtungsvektor $\Theta$ für die $k = 1, ..., K$ Merkmale.

Die Beobachtungen müssen NICHT voneinander unabhängig sein!

### 8.3.1 Strukturen der CRF

**Beispiel**
Lexikon $X = \{Udo, goes, comes, to, from, Unna\}$
Zustände (Kategorien) $Y = \{Pos, Per, \emptyset\}$
Beobachtung: $x_1 = Udo$, $x_2 = goes$, $x_3 = to$, $x_4 = Unna$
$|X|$ Matritzen der Form $|Y| \times |Y|$

| $M_{Felix}$ | Pos | Per | $\emptyset$ |
|---|---|---|---|
| Pos | 1 | 2 | 3 |
| Per | 4 | 5 | 6 |
| $\emptyset$ | 7 | 8 | 9 |

2:
$exp[\lambda_1 f_1(Pos, Per, Udo) +$
$\lambda_2 f_2(Per, Udo) +$
$\lambda_3 f_3(Per, capitalletter)]$

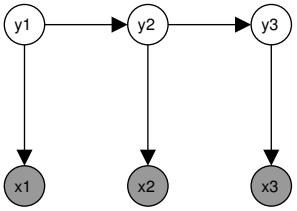
$$M_x = exp[\sum_k^K \lambda_k f_k(y_{t-1}, y_t, x_t) + \sum_k^K \lambda_k f_k(y_t, x_t)]$$

Potenzialfunktion rechnen: Für jedes $x_t$ der beobachteten Sequenz die gesamte Matrix $M_x$ ausrechnen!

**CRF vs. HMM**

HMM:

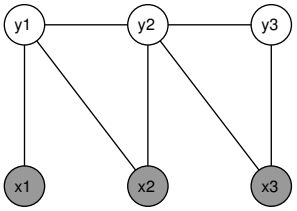
$p(\mathbf{y}, \mathbf{x}) = \prod_{t=1}^{3} p(y_t|y_{t-1})p(x_t|y_t)$




---

CRF:

$p(\mathbf{y}|\mathbf{x}) =$
$Z(\mathbf{x})^{-1} \prod_{t=1}^{3} \exp\left[a_{y_{t-1},y_t,x_t} + b_{y_t,x_t}\right]$




Bemerkung: $y_0 = start$, $p(y|start) = \pi(y)$

**CRF Modell**

Normalisieren, wenn wir Wahrscheinlichkeit wollen, etwa: $Z(x) = M_1(x) \cdot ... \cdot M_T(x)$

$$Z(x) = \sum_y^{|Y|} exp[\sum_k^K \lambda_k f_k(y_{t-1}, y_t, x_t)]$$

$$\frac{1}{Z(x)} \prod_{t=1}^{T} exp[\sum_k^K \lambda_k f_k(y_{t-1}, y_t, x_t) + \sum_k^K \lambda_k f_k(y_t, x_t)]$$

Warum Exponenzialfunktion?

**Exponenzialfunktionen**

**Hammersley Clifford 1990**

Eine lokale Verteilung mit Markov-Eigenschaft hat auch global die Markov Eigenschaft.
Eine Wahrscheinlichkeitsverteilung hat die Markov-Eigenschaft, gdw.

$$P(X) \propto exp(\sum Q(C, X))$$

wobei $C$ die Menge aller Cliquen im Graph ist und $Q$ irgendeine reellwertige Funktion, die die likelihood für bestimmte Ausprägungen der Zufallsvariablen im Graph ausdrückt.

Die Exponenzialfunktion liefert immer einen Wert > 0.

**Aufgaben für Algorithmen**

- Training:
  Wie oft kam $to$ in den Trainingsdaten mit $y = \emptyset$ vor?
  *Forward Backward Algorithmus* für *linear chain* CRF (i.e. $y_t$ sind linear verbunden)
  Gewichte lernen (parameter estimate) $\Theta = (\lambda_1, ..., \lambda_K)$
  *Maximum Likelihood*
- Annotierung einer Beobachtungssequenz durch das gelernte Modell
  *Viterbi Algorithmus* für die wahrscheinlichste Annotation

### 8.3.2 Forward Backward Wahrscheinlichkeiten rechnen

**Forward Backward Algorithmus**
Sequenz $1, ..., t, ..., T$ und festes $y_t = y_i$

**forward** Wahrscheinlichkeit $\alpha_t(i)$ für Sequenz $1, ..., t$

**backward** Wahrscheinlichkeit $\beta_t(i)$ für Sequenz $t, ..., T$

- Anfang: $\alpha_1(i) = P(y_0 = start, y_1 = y_i, x_1)$ $\beta_T(i) = P(y_{T+1} = stop, y_T = y_i, x_T)$
- Rekursion für 1 < t < T: $\alpha_t(x) = \alpha_{t-1} \cdot M_t(x) = \alpha_{t-1}(x) \cdot exp[\sum_k^K \lambda_k f_k(y_{t-1}, y_t, x_t) + \sum_k^K \lambda_k f_k(y_t, x_t)]$
  $\beta_t(x) = \beta_{t+1}(x) \cdot M_{t+1}(x) = \beta_{t+1}(x) \cdot exp[\sum_k^K \lambda_k f_k(y_t, y_{t+1}t, x_t) + \sum_k^K \lambda_k f_k(y_{t+1}, x_t)]$

**Algorithmus**

Für alle Beobachtungen t=1 bis T
  für alle Label i=1 bis $|Y|$
    berechne Forward $\alpha_t(y_i|x)$
berechne $Z(x)$
Für alle Beobachtungen t=T bis 1
  für alle Label i=1 bis $|Y|$
    berechne Backward $\beta_t(y_i|x)$

$$O(T|Y|^2)$$

Für jeden Schritt (Beobachtung) $|Y|$ Nachrichten $(\alpha, \beta)$ rechnen und jede Nachricht summiert über $|Y|$ Terme.

### 8.3.3 Viterbi Annotation einer Sequenz von Beobachtungen

**Annotation einer Sequenz**

Wir wollen die Sequenz der Label als Vektor erhalten, $\vec{y}_t$.
Forward $\alpha$ wurde eben als Summe von Übergangs- und Zustandswahrscheinlichkeiten gerechnet.

- Jetzt wird das maximale Label $y$ vorwärts propagiert!
  $\alpha^*(y|x) = \alpha_{t-1} \cdot max_y exp[\sum_k^K \lambda_k f_k(y_{t-1}, y_t, x_t) + \sum_k^K \lambda_k f_k(y_t, x_t)]$
- dabei der wahrscheinlichste Vorgänger bestimmt:
  $\delta_t(y|x) = \alpha_{t-1} \cdot max_y exp[\sum_k^K \lambda_k f_k(y_{t-1}, y_t, x_t) + \sum_k^K \lambda_k f_k(y_t, x_t)]$
- Vom maximal wahrscheinlichen Label für den letzten Knoten $\delta_T(y|x)$ rechnet man rekursiv zurück und erhält die Sequenz.

**Viterbi Algorithmus**

Für alle Beobachtungen t=1 bis T
  für alle Label i=1 bis $|Y|$
    berechne MaxForward $\alpha_t^*(y_i|x)$
    berechne wahrscheinlichsten Vorgänger: $\delta_t(y|x)$
$\vec{y}_T = max_y \alpha_t^*(y_i|x)$
Für alle Beobachtungen t=T-1 bis 1
    $\vec{y}_t = \delta_t(y|x, \vec{y}_{t+1})$

### 8.3.4 Bestimmen des Gewichtsvektors

**Bestimmen der Gewichte $\Theta$**

Gegeben: Trainingsmenge $\mathcal{T}$

$$\max_{A,B} \mathcal{L}(A, B|\mathcal{T}) \approx \max_{A,B} \prod_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} p(\mathbf{y}, \mathbf{x}), \text{ HMM}$$

$$\max_{A,B} \mathcal{L}(A, B|\mathcal{T}) \approx \max_{A,B} \prod_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} p(\mathbf{y}|\mathbf{x}), \text{ CRF}$$

Zielfunktion ist Konvex
$\Rightarrow$ Maximierung mit Gradientenabstieg oder Newton-Verfahren

**Maximierung der Likelihood**

Sei $\mathcal{T}(X_t = x)$ die Menge aller Trainingsbeispiele die an der $t$-ten Stelle der Sequenz die Beobachtung $x$ enthalten.

Sei $1_{Y_{t-1}=y',Y_t=y}$ eine Indikatorfunktion die den Wert 1 annimmt gdw. die Sequenz an Position $t$ im Zustand $y$ und an Position $t-1$ im Zustand $y'$ ist.

$$a_{y',y,x}^{(t+1)} = a_{y',y,x}^{(t)} + \eta^{(t)} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}(X_t=x)} \left[1_{Y_{t-1}=y',Y_t=y} - p\left(y', y \mid x\right)\right]$$

$\eta^{(t)}$ ist eine Schrittweite.
Die Optimierung konvergiert für $\eta^{(t)} = \mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$
Update für $b_{i,k}$ analog.

**Was wissen Sie jetzt?**

- CRFs können beliebige Graphen sein. Die Beobachtungen werden unabhängig von der Reihenfolge genutzt. Gesucht ist $p(y|x)$.
- Wir haben die Übergangswahrscheinlichkeiten in der Matrix A und die Zustandswahrscheinlichkeiten in der Matrix B gesehen. Übergänge und Zustände werden als Merkmale aufgefasst.
- Die Merkmale sind jeweils gewichtet. Der Vektor aller Gewichte ist $\Theta$ und wird nach Maximum Likelihood bestimmt.
- Sie kennen die Matritzen $Mx$ für jede Beobachtung $x$ mit allen Zustandsübergängen.
- Der Forward Backward-Algorithmus berechnet die Wahrscheinlichkeiten. Sie wissen auch, wie!
- Der Viterbi-Algorithmus maximiert, um für eine Beobachtungssequenz die Label-Sequenz zu finden.

# Kapitel 9

# Clustering

## 9.1 Lernaufgabe Cluster-Analyse

**Lernaufgabe Clustering**

Gegeben

- eine Menge $\mathcal{T} = \{\vec{x_1}, ..., \vec{x_N}\} \subset X$ von Beobachtungen,
- eine Anzahl $K$ zu findender Gruppen $C_1, ..., C_K$,
- eine Abstandsfunktion $d(\vec{x}, \vec{x'})$ und
- eine Qualitätsfunktion.

Finde

- Gruppen $C_1, ..., C_K$, so dass
- alle $\vec{x} \in X$ einer Gruppe zugeordnet sind und
- die Qualitätsfunktion optimiert wird: Der Abstand zwischen Beobachtungen der selben Gruppe soll minimal sein; der Abstand zwischen den Gruppen soll maximal sein.

**Bild**

Der Abstand wurde zum Cluster-Zentrum gemessen. Dadurch ergibt sich der grüne Punkt neben den roten.

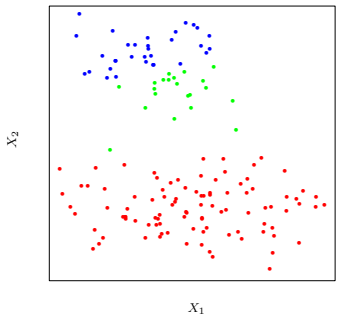



Figure 14.4: *Simulated data in the plane, clustered into three classes (represented by red, blue and green), by the K-means clustering algorithm*

- Könnte ein besseres Abstandsmaß den grünen Punkt dem roten Cluster zuweisen?
- Wenn nicht nur ein Punkt als Repräsentation eines Clusters gewählt wird, würde das Clustering dann besser?
- Wie kann man die Cluster verständlich beschreiben?
- Wäre $K = 2$ besser gewesen?

**Die Probleme der Cluster-Analyse**

1. Bestimmung des Abstandsmaßes
2. Formulierung des Optimierungsproblems
3. Repräsentation der Cluster
4. Bestimmung von $K$

### 9.1.1 Abstandsmaße

**Bestimmung des Abstandsmaßes**

- Ähnlichkeitsmaße haben wir schon bei kNN gesehen.
- Im Allgemeinen ist der Abstand invers zur Ähnlichkeit:

$$D(\vec{x_1}, \vec{x_2}) = 1 - Sim(\vec{x_1}, \vec{x_2})$$

- Man kann aber irgendeine geeignete monoton absteigende Funktion zur Überführung der Ähnlichkeiten in Abstände wählen.

**$sim$: Ähnlichkeit für einzelne Attribute (Erinnerung)**

**Numerische Attribute:** Sei $max_j$ der höchste Wert von $X_j$ und $min_j$ der niedrigste, sei $x_{ij}$ der Wert des $j$-ten Attributs in der $i$-ten Beobachtung, dann ist die normalisierte Ähnlichkeit:

$$sim_j(x_{1j}, x_{2j}) = 1 - \frac{|x_{1j} - x_{2j}|}{max_j - min_j}$$

**Nominale Attribute:** Ganz einfach:

$$sim_j(x_{1j}, x_{2j}) = \begin{cases} 1 & falls \quad x_{1j} = x_{2j} \\ 0 & sonst \end{cases}$$

**$d$: Abstand für einzelne Attribute**

**Numerische Attribute:** Ohne Normalisierung durch $max_j - min_j$ ist der Betrag der Differenz:

$$d_j(x_{ij}, x_{i'j}) = \mid x_{ij} - x_{i'j} \mid$$

Der quadratische Abstand zwischen Beobachtungen $x_i$ und $x'_i$ bezüglich des Merkmals $X_j$ gewichtet große Abstände stärker als kleine:

$$d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2 \tag{9.1}$$

**Nominale Attribute:** Man kann für jede Variable $X_j$ mit $M$ Attributwerten eine $M \times M$ Abstandsmatrix angeben oder einfach:

$$d_j(x_{1j}, x_{2j}) = \begin{cases} 1 & falls \quad x_{1j} \neq x_{2j} \\ 0 & sonst \end{cases}$$

**$Sim$: Ähnlichkeit der Beobachtungen als Kombination der Attributähnlichkeiten**

Im einfachsten Fall mitteln wir die Einzelähnlichkeiten:

$$Sim(\vec{x_1}, \vec{x_2}) = \frac{1}{p} \sum_{j=1}^{p} sim(x_{1j}, x_{2j})$$

Die *Korrelation* verwendet das Mittel $\bar{x}_i$ über allen $p$ Variablen:

$$Sim(\vec{x_1}, \vec{x_2}) = \frac{\sum_{j=1}^{p}(x_{1j} - \bar{x}_1)(x_{2j} - \bar{x}_2)}{\sqrt{\sum_{j=1}^{p}(x_{1j} - \bar{x}_1)^2 \sum_{j=1}^{p}(x_{2j} - \bar{x}_2)^2}} \tag{9.2}$$

Vielleicht sind einige Attribute wichtiger als andere?

$$Sim(\vec{x_1}, \vec{x_2}) = \frac{\sum_{j=1}^{p} w_j sim(x_{1,j}, x_{2,j})}{\sum_{j=1}^{p} w_j}$$

Wie bestimmt man $w_j$?

**Abstandsmaß**

- Verwendet wird eine $N \times N$ Matrix $\mathbf{D}$ für die $N$ Beobachtungen, wobei $d_{12}$ der Eintrag für $D(\vec{x_1}, \vec{x_2})$ ist.
- Die Matrix hat keine negativen Einträge.
- Die Diagonale der Matrix: $d_{ii} = 0$
- Der Abstand soll symmetrisch sein – falls nicht: $(\mathbf{D} + \mathbf{D}^T)/2$.

**$D$: Abstand der Beobachtungen als Kombination der Attributabstände**

- Gewichteter Durchschnitt:

$$D(\vec{x_1}, \vec{x_2}) = \sum_{j=1}^{p} w_j d_j(x_{1j}, x_{2j}); \sum_{j=1}^{p} w_j = 1 \tag{9.3}$$

- Bei quadratischem Abstand $d_{12}$ ergibt sich:

$$D(\vec{x_1}, \vec{x_2}) = \sum_{j=1}^{p} w_j (x_{1j} - x_{2j})^2 \tag{9.4}$$

- Man kann die Korrelation (Gleichung 9.2) verwenden:

$$1 - Sim(\vec{x_1}, \vec{x_2}) \tag{9.5}$$

### Einfluss einer Variablen auf das Clustering

- Wenn für alle Variablen $w_j = 1$ wäre, hätten doch nicht alle Variablen den gleichen Einfluss auf das Clustering!
- Der Einfluss einer Variable $X_j$ richtet sich vielmehr nach ihrer durchschnittlichen Unähnlichkeit:

$$\bar{d}_j = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} d_j(x_{ij}, x_{i'j}) \tag{9.6}$$

- Beim gewichteten quadratischen Abstand

$$\bar{d}_j = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} (x_{ij} - x_{i'j})^2 = 2 \cdot var_j \tag{9.7}$$

  wobei $var_j$ die anhand der Beobachtungmenge $\mathcal{T}$ geschätzte Varianz von $X_j$ ist.
- Der Einfluss einer Variablen auf das Clustering richtet sich also nach der Varianz! Der relative Einfluss ist $w_j \, \bar{d}_j$.

### Beispiel für Nachteil gleichen Einflusses der Variablen

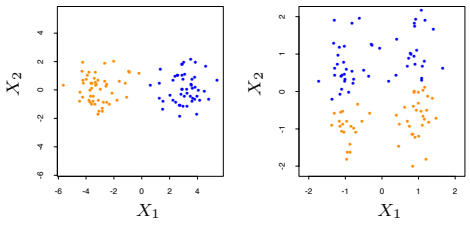



Figure 14.5: *Simulated data: on the left, K-means clustering (with K=2) has been applied to the raw data. The two colors indicate the cluster memberships. One the right, the features were first standardized before clustering. This is equivalent to using feature weights* $1/[2 \cdot \mathrm{var}(X_j)]$. *The standardization has obscured the two well-separated groups. Note that each plot uses the same units in the horizontal and vertical axes.*

- Alle Variablen haben den selben Einfluss auf das Clustering, wenn $w_j \sim 1/\bar{d}_j$.
- Wenn als Gewichte $w_j = \frac{1}{2 \cdot var_j}$ gewählt wird, hat man den Einfluss der Varianz ausgeschaltet und erhält manchmal keine gute Separierung mehr.

### Es hängt von der Anwendung ab, wie man $w_j$ wählt!

Für eine Anwendung kann man vor dem Clustern

1. gar nichts tun, d.h. die Rohdaten ohne Gewichtung und ohne Normalisierung clustern,
2. die Rohdaten *normalisieren* (Werte im selben Wertebereich, z.B. $[0, 1]$, oder jeweils $max_j - min_j$ in den Abständen),
3. $\bar{d}_j$ für jedes Merkmal berechnen (Varianz-Gleichung 9.7),

4. die Rohdaten *standardisieren*, so dass alle Variablen den gleichen Einfuss haben,
5. Gewichte $w_j$, die dem Sachbereich entsprechen könnten oder dem Clustering-Ziel, direkt auf die Daten als Transformation der Eingabe anzuwenden. (*Implizites* $w_j$!)
6. Dann die Ergebnisse vergleichen!

### 9.1.2 Optimierungsprobleme

**Qualitätsfunktionen**

Sei die Anzahl $K$ der Cluster gegeben und jedes Cluster durch eine ganze Zahl $k \in \{1, 2, ..., K\}$ eindeutig ausgezeichnet. Die Abbildung $C(i) = k$ weist der $i$-ten Beobachtung das $k$-te Cluster zu.

**Innerer Abstand Within:** Minimiert werden soll der Abstand innerhalb eines Clusters $C$:

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} D(\vec{x_i}, \vec{x_{i'}}) \tag{9.8}$$

**Zwischenunähnlichkeit Between:** Maximiert werden soll der Abstand zwischen Clustern:

$$B(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')\neq k} D(\vec{x_i}, \vec{x_{i'}}) \tag{9.9}$$

**Optimierungsproblem der Cluster-Analyse**

- Gegeben die Summe aller Abstände $T = \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} d_{ii'}$, ergänzen sich $W(C) + B(C) = T$, so dass die Minimierung von $W(C)$ der Maximierung von $B(C)$ entspricht.
- Man hat so nur *ein* Optimierungsproblem.
- Sei $\bar{x}_k = (\bar{x}_{1k}, ..., \bar{x}_{pk})$ der Vektor der Mittelwerte aller Variablen in Cluster $k$ und $N_k = \sum_{i=1}^{N} I(C(i) = k)$, dann ist das Optimierungsproblem:

$$C^* = min_C \sum_{k=1}^{K} N_k \sum_{C(i)=k} \parallel \vec{x_i} - \bar{x_k} \parallel^2 \tag{9.10}$$

## 9.2 K-Means

**Iteratives Lösen des Optimierungsproblems – K-Means**

**Algorithmus K-Means**$(\mathcal{T}, K)$

1. Wähle $K$ Beobachtungen aus $\mathcal{T}$ zufällig als Mittelpunkte $\vec{m_1}, ..., \vec{m_K}$ von Clustern aus.
2. Berechne das Clustering anhand der Mittelpunkte:

$$C(i) = argmin_{1 \leq k \leq K} \parallel \vec{x_i} - \vec{m_k} \parallel^2 \tag{9.11}$$

3. Berechne die Mittelpunkte entsprechend $C(i)$:

$$\vec{m}_i := argmin_m \sum_{i=1}^{N} \parallel \vec{x}_i - \vec{m} \parallel^2 \tag{9.12}$$

4. Wiederhole Schritt 2 und 3 bis die Zuweisungen sich nicht mehr ändern. Gib zurück $C(1), ..., C(K)$.

**K-Means im Bild**

**Eigenschaften von K-Means**

- K-Means ist für numerische Variablen gemacht.
- Als Abstandsmaß wird der quadratische Euklidsche Abstand genutzt.
  - Den größten Einfluss haben Datenpunkte mit dem größten Abstand.
  - Das Verfahren ist daher anfällig für Ausreißer.
- Der Aufwand ist proportional zu $N \cdot K$.
  - Für jedes Cluster wird der Mittelpunkt berechnet anhand der zugeordneten Beobachtungen. Ein Cluster ist also nur durch einen Punkt repräsentiert.
  - Für alle Beobachtungen wird der Abstand zu den $K$ Mittelpunkten berechnet.
- Es kann sein, dass die Lösung von K-Means nicht optimal ist (lokales Optimum).

**Repräsentation der Cluster**

- K-Means repräsentiert ein Cluster durch einen errechneten Punkt. Dies ist effizient.
- K-Medoid wählt eine Beobachtung als Repräsentation eines Clusters. Dafür muss über allen Punkten optimiert werden – ineffizient.
- Rajeev Rastogi hat vorgeschlagen *einige* Punkte als Repräsentation eines Clusters zu wählen (well scattered points).
- Oft möchte man eine interpretierbare Charaktierisierung der Cluster haben.
  - Aufgabe des *labeling*: finde eine (logische) Charakterisierung der Cluster. Man betrachtet die Cluster als Klassen und wendet z.B. Entscheidungsbaumlernen an.
  - Ryszard Michalski hat ein logisches Cluster-Verfahren vorgeschlagen, die Star-Methode (AQ-Algorithmus), bei dem direkt über den nominalen Werten der Beobachtungen gearbeitet wird.

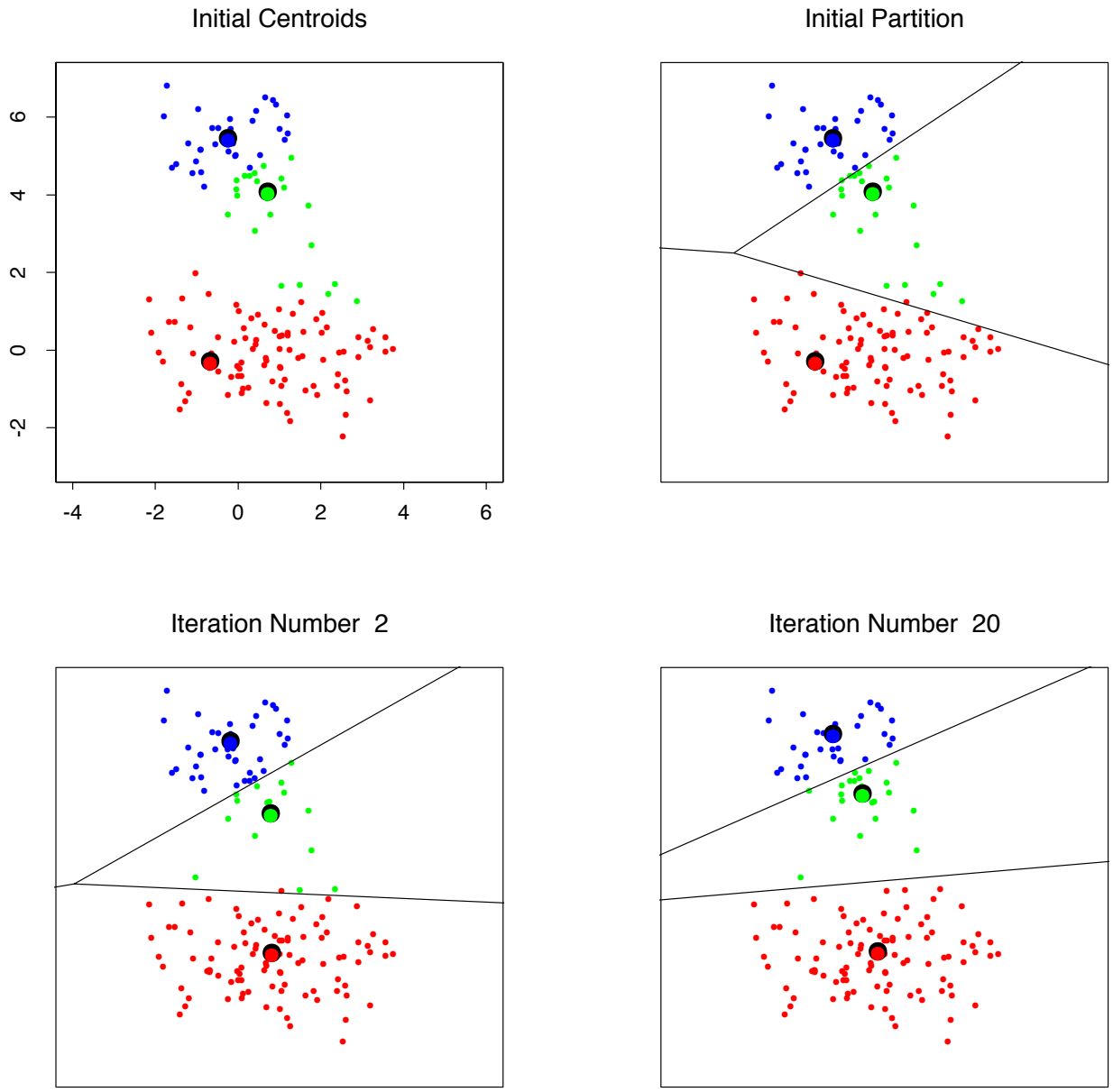



Figure 14.6: *Successive iterations of the $K$-means clustering algorithm for the simulated data of Figure 14.4.*

**Bestimmung der vorgegebenen Mittelpunkte**

Die Lösung von K-Means hängt von den gewählten Start- mittelpunkten ab. Dafür gibt es mindestens zwei Auswege:

- Mehrfach mit zufällig gewählten Startmittelpunkten den Algorithmus starten!
- Optimierungskriterium

$$min_{C,\{m_k\}_1^K} \sum_{k=1}^{K} N_k \sum_{C(i)=k} \parallel \vec{x_i} - m_k \parallel^2$$

  Für $k = 1, ..., K$: Wähle einen Mittelpunkt $i_k$ so, dass das Kriterium minimiert wird gegeben $i_1, ..., i_{k-1}$. Starte K-Means mit den so gefundenen $K$ Mittelpunkten.

### 9.2.1 Bestimmung von $K$

**Wie viele Cluster sollen gebildet werden?**

- Vielleicht geht aus der Anwendung hervor, wie viele Cluster nötig sind. Z.B. sollen Kunden so auf $K$ Vertriebsmitarbeiter aufgeteilt werden, dass ein Mitarbeiter ähnliche Fälle bearbeitet.
- Oft soll $K^*$ anhand der Daten so ermittelt werden, dass die Clustering-Qualität optimiert wird (Gleichung 9.8).

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} D(\vec{x_i}, \vec{x_{i'}})$$

  Man bestimmt $W_1, ..., W_{Kmax}$ für $K = 1, ..., Kmax$.

**Daten-gestützte Bestimmung von $K$**

- Wenn $K < K^*$, dann ist meist eine Teilmenge der Beobachtungen in einem Cluster schon richtig zugeordnet, das Cluster müsste aber weiter aufgeteilt werden.
  - $W_{K+1} << W_K$
- Wenn $K > K^*$, dann ist ein 'richtiges' Cluster zerteilt worden.
  - $W_{K+1} < W_K$.
- Man sucht also nach einem Knick in der Kurve der $W_1, ..., W_{Kmax}$-Werte und wählt als $K$ den Wert mit dem geringsten Abstieg $W_K - W_{K+1}$.
  - $\{W_K - W_{K+1} \mid K < K^*\} >> \{W_K - W_{K+1} \mid K \geq K^*\}$

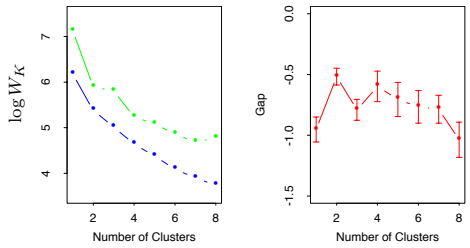



Figure 14.11: *Left panel: observed (green) and expected (blue) values of* $\log W_K$ *for the simulated data of Figure 14.4. Right panel: Gap curve, equal to the difference between the observed and expected values of* $\log W_K$*. The Gap estimate* $K^*$ *is the smallest* $K$ *producing a gap within one standard deviation of the maximum; here* $K^* = 2$*.*

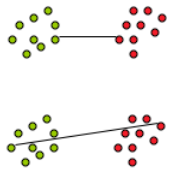
### Gap Heuristik

- Tibshirani et al. (2001) vergleichen die Kurve der anhand der Daten gemessenen $W$-Werte mit einer "normalen".
- Es werden $n$ Mal zufällig Datenpunkte erzeugt, die innerhalb einer Hülle um die Beobachtungen gleichmäßig verteilt sind.
- Für die simulierten Daten werden die $W$-Werte ausgerechnet und der Erwartungswert bestimmt.
- Die Kurven werden auf einer logarithmisierten Skala aufgetragen und verglichen: wo der Abstand zwischen den Kurven (gap) am größten ist, liegt das richtige $K^*$.

### Gap Heuristik im Bild

### Was wissen Sie jetzt?

- Sie haben die Abstandsmaße kennengelernt und sich dabei an die Ähnlichkeit bei $k$NN erinnert.
- Sie kennen das Optimierungsproblem des Clusterings (Gleichung 9.10).
- Sie kennen das Qualitätskriterium des inneren Abstands (Gleichung 9.8).
- Die Repräsentation eines Clusters kann durch alle zugeordneten Punkte, einige zugeordnete Punkte, einen zentralen zugeordneten Punkt oder ein berechnetes Zentrum sowie durch logische Formeln erfolgen.
- Zur Lösung des Optimierungsproblems kennen Sie K-Means: Euklidscher Abstand, Repräsentation durch berechnete Mittelpunkte, iteratives Vorgehen.
- Als Vorgehen zur Wahl der Anzahl $K$ und zur Initialisierung der $K$ Mittelpunkte haben Sie Heuristiken gesehen.



## 9.3 Hierarchisches Clustering

**Hierarchisches Clustering**

- Die Cluster sollen nicht auf einer Ebene liegen, sondern eine Taxonomie bilden.
- Die unterste Ebene enthält einzelne Beobachtungen.
- Jede Ebene enthält Cluster, die (zwei) Cluster der Ebene darunter subsummieren.
- Die oberste Ebene enthält ein Cluster mit allen Beobachtungen.
- Man unterscheidet ein Vorgehen bottom-up (agglomerativ) und top-down (aufteilend).

**Agglomeratives Clustering**

- Stufenweise werden Beobachtungen zu übergeordneten Clustern verschmolzen.
- Oft wird ein binärer Baum erzeugt, d.h. immer je 2 Cluster werden verschmolzen.
- Der Benutzer sucht die aussagekräftigste Ebene aus.
- Grundlage ist die *Unähnlichkeit von Clustern*: solche mit geringster Unähnlichkeit werden verschmolzen.
- Die Unähnlichkeit $d(G, H)$ der Cluster $G, H$ wird berechnet durch den Abstand $d_{gh} = D(\vec{x_g}, \vec{x_h})$, wobei $\vec{x_g} \in G, \vec{x_h} \in H$.
- Welche Beobachtungen genutzt werden, macht den Unterschied zwischen den 3 wichtigsten Maßen zur Cluster-Unähnlichkeiten aus.

**Single Linkage Clustering**

Die Unähnlichkeit zwischen Cluster $G$ und $H$ ist die Unähnlichkeit der nächsten Punkte.

$$\begin{aligned} d_{SL}(G, H) &= min_{\vec{x_g} \in G, \vec{x_h} \in H} D(\vec{x_g}, \vec{x_h}) \\ &= min_{g \in G, h \in H}\ d_{gh} \end{aligned}$$

- Problem: Single Linkage ergibt eventuell Cluster, die nicht kompakt sind mit großer Unähnlichkeit innerhalb eines Clusters.

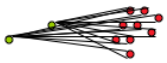


Figure 1.3: *DNA microarray data: expression matrix of 6830 genes (rows) and 64 samples (columns), for the human tumor data. Only a random sample of 100 rows are shown. The display is a heat map, ranging from bright green (negative, under expressed) to bright red (positive, over expressed). Missing values are gray. The rows and columns are displayed in a randomly chosen order.*

**Complete Linkage Clustering**

Die Unähnlichkeit zwischen Cluster $G$ und $H$ ist die Unähnlichkeit der entferntesten Punkte.

$$\begin{aligned} d_{CL}(G,H) &= max_{\vec{x_g} \in G, \vec{x_h} \in H} D(\vec{x_g}, \vec{x_h}) \\ &= max_{g \in G, h \in H}\ d_{gh} \end{aligned}$$

- Problem: Complete Linkage produziert kompakte Cluster, aber eventuell sind die Beobachtungen eines Clusters $G$ näher zu denen eines anderen $H$ als zu denen in $G$.

**Average Linkage Clustering**

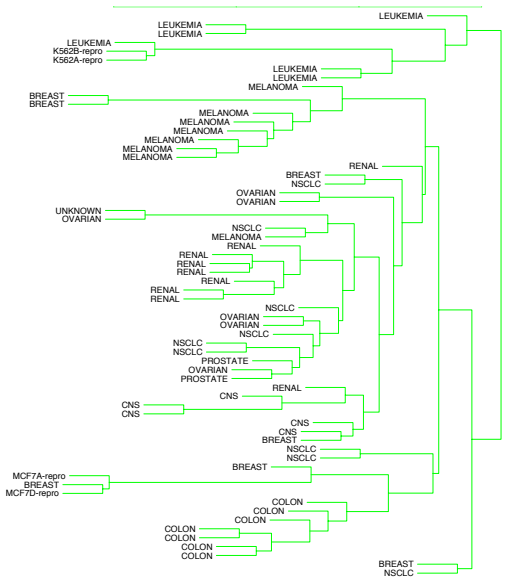
Die Unähnlichkeit zwischen Cluster $G$ und $H$ ist die durchschnittliche Unähnlichkeit aller Punkte in $G$ von allen in $H$.

$$d_{AL}(G,H) = \frac{1}{N_G N_H} \sum_{g \in G} \sum_{h \in H} d_{gh}$$

- Kompromiss zwischen Single und Complete Linkage: relativ kompakte Cluster, die relativ weit von einander entfernt sind.
- Problem: Eine strikt monoton aufsteigende Transformation des Abstandsmaßes $h(d_{gh})$ kann das Ergebnis stark verändern.

**Beispiel MicroArray-Daten über Krebs**

**Beispiel Average Linkage bei MicroArray-Daten über Krebs**




**Dendrogramme für agglomeratives Clustering der MicroArray-Daten über Krebs mit Average, Complete, Single Linkage**

### Dendogramme

- Monotonie: Die Unähnlichkeit steigt über die Ebenen von unten nach oben monoton an.
- Ein Dendogramm ist so angeordnet, dass die Höhe eines Knoten (Clusters) gerade proportional zur Unähnlichkeit zwischen den beiden Unterknoten ist.
- Deshalb kann der Benutzer eine Ebene auswählen, bei der die Unähnlichkeit zwischen Clustern einen Schwellwert übersteigt.

### Aufteilendes Clustering durch rekursives K-Means

- Die rekursive Anwendung von K-Means mit $K = 2$ ergibt ein aufteilendes Verfahren.
- Allerdings ist das Ergebnis dann kein Dendogramm, bei dem die Unähnlichkeit mit den Ebenen immer monoton ansteigt.
- Deshalb gibt es ein anderes Verfahren.

### Aufteilendes Clustering durch iteratives Verringern der Unähnlichkeit in einem Cluster

- Alle Beobachtungen sind im Wurzelknoten $G$.
- Aufteilung(G)
  1. Initialisierung:Wähle den Punkt $\vec{x_h}$ in $G$, der am unähnlichsten zu allen anderen ist. Dieser wird dem neuen Cluster $H$ zugeordnet.

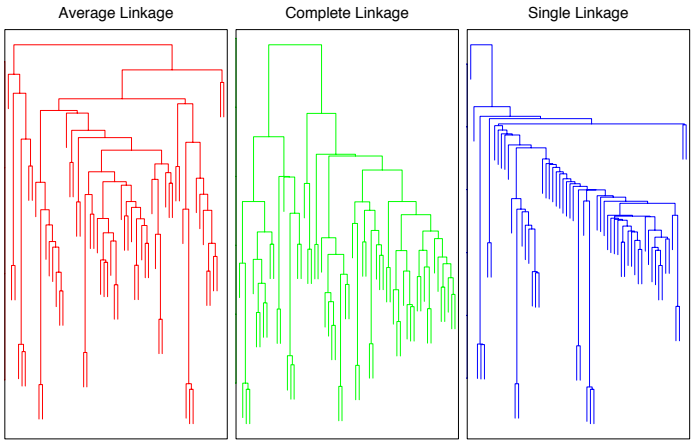



Figure 14.13: *Dendrograms from agglomerative hierarchical clustering of human tumor microarray data.*

2. Teile iterativ $G$ auf solange es ein $\vec{x_i} \in G$ gibt, das im Durchschnitt ähnlicher zu allen $\vec{x_j} \in H$ ist als zu allen $\vec{x_g} \in G$:$H := H \cup \{\vec{x_i}\}$; $G := G \backslash \{\vec{x_i}\}$;
3. Wähle Cluster zur Aufteilung aus:Solange $\mid G \mid > 1$ und $d_{ij} > 0$ für alle $\vec{x_i}, \vec{x_j} \in G$ Aufteilung(G). Solange $\mid H \mid > 1$ und $d_{ij} > 0$ für alle $\vec{x_i}, \vec{x_j} \in H$ Aufteilung(H).

**Was wissen Sie jetzt?**

- Top-down Clustering kann durch rekursives K-Means realisiert werden, ist aber aufwändig.
- Optimieren der Average Linkage $d_{AL}(G, H)$ für alle möglichen Aufteilungen wird angenähert durch ein iteratives Verfahren, bei dem in jeder Iteration eine Beobachtung von dem Ausgangscluster $G$ dem neuen Cluster $H$ zugeordnet wird.

**Cluster Ensembles**

- Lernaufgabe
  - Gegeben verschiedene clusterings eines Datensatzes
  - finde ein kombiniertes clustering.
- Wie müssen die Eingabe-clusterings beschaffen sein?
  - möglichst unterschiedlich
  - möglichst leicht/schnell zu berechnen
- Wie soll die Konsensusfunktion aussehen, damit das kombinierte clustering gut ist?
  - Gütemaß: inter cluster dissimilarity, intra cluster similarity
  - Kombiniertes clustering ist besser als irgend ein einzelnes.

**Cluster Ensembles: formale Aufgabenstellung**

**Gegeben:** $X$: Beobachtungen im $d$-dimensionalen Raum
$G$: Menge von clusterings $\{\varphi_1, \ldots, \varphi_H\}$

- Dabei: $\varphi_i = \{g_1^i, \ldots, g_K^i\}$, wobei $X = g_1^i \cup g_2^i \cup \cdots \cup g_K^i$

**Gesucht:** $\sigma = \{g_1, \ldots, g_K\}$, wobei

- $X = g_1 \cup g_2 \cup \cdots \cup g_k$ und
- $sim(x_1, x_2) > sim(x_1, x_3)$ gdw. $x_1, x_2 \in g_i, x_3 \notin g_i$

**Cluster-Zugehörigkeit als neues Merkmal [15]**

*Gegeben:* $X$: Beobachtungen im $d$-dimensionalen Raum
<all> $G$: Menge von clusterings $\{\varphi_1, \ldots, \varphi_H\}$ mit

- $\varphi_i = \{g_1^i, \ldots, g_K^i\}$
- $X = g_1^i \cup \cdots \cup g_K^i$

*Gesucht:* $\sigma = \{g_1, \ldots, g_K\}$, wobei

- $X = g_1 \cup \cdots \cup g_K$
- $sim(x_1, x_2) > sim(x_1, x_3)$ gdw. $x_1, x_2 \in g_i, x_3 \notin g_i$

- Mehrere clusterings über den Attributen $f_1, \ldots, f_d$ (Eingabe $G$)
- Ein clustering über den Attributen $\varphi_1, \ldots, \varphi_H$ (Konsensus $\sigma$)

| Attribut | $f_1$ | $\ldots$ | $f_d$ | $\varphi_1$ | $\ldots$ | $\varphi_H$ |
|---|---|---|---|---|---|---|
| Attributwerte | | | | $g_1^1$ | | $g_1^H$ |
| | | | | $\ldots$ | | $\ldots$ |
| | | | | $g_K^1$ | | $g_K^H$ |

**Konsensusfunktion Ko-Assoziation**

Ko-Assotiationsmatrix

| co | $\varphi_1$ | $\ldots$ | $\varphi_H$ |
|---|---|---|---|
| $x_1$ | $g_i$ | | $g_j$ |
| $\ldots$ | | | |
| $x_N$ | $g_i$ | | $g_j$ |

$\delta(x_1, x_N) = 1$ für $\varphi_1$
$\delta(x_1, x_N) = 0$ für $\varphi_H$

| $sim$ | $x_1$ | $\ldots$ | $x_N$ |
|---|---|---|---|
| $x_1$ | - | | $sim(x_1, x_N)$ |
| $\ldots$ | | | |
| $x_N$ | $sim(x_1, x_N)$ | | - |

- neue Attribute für jede Beobachtung in co-Matrix eintragen
- für jedes Paar $(x_i, x_j)$ und jedes Attribut Gleichheit der Attributwerte feststellen

$$d(a,b) = \begin{cases} 1, & \text{falls } a = b \\ 0, & \text{falls } a \neq b \end{cases}$$

- Zählen, wie oft $(x_i, x_j)$ den gleichen Attributwert haben!

$$sim(x,y) = \sum_{i=1} d(\varphi_i(x), \varphi_i(y))$$

**Algorithmus für den Konsens**

Beliebiger clustering Algorithmus verwendet $sim(x, y)$ als Ähnlichkeitsmaß und

- maximiert Ähnlichkeit aller $x_i$ im cluster und
- minimiert Ähnlichkeit aller $x_i$ zwischen clusters.

**Konsensusfunktion Nutzen**

Vergleich des Kandidaten-clusterings $\sigma = \{c_1, \ldots, c_K\}$ mit einem Eingabe-clustering $\varphi_i = \{g_1^i, \ldots, g_{K(i)}^i\}$: kann jedes cluster $g_i$ der Eingabe (jedes Attribut) besser vorhergesagt werden, wenn man $\sigma$ kennt?

$$U(\sigma, \varphi_i) = \sum_{r=1}^{K} p(c_r) \sum_{j=1}^{K(i)} p(g_j^i|c_r)^2 - \sum_{j=1}^{K}(i) p(g_j^i)^2$$

Nutzen von $\sigma$ für alle $\varphi$:

$$U(\sigma, G) = \sum_{i=1}^{H} U(\sigma, \varphi_i)$$

Wähle den Kandidaten mit dem größten Nutzen!

$$s_{\text{best}} = \arg\max_s U(s, G)$$

$$p(c_r) = \frac{|c_r|}{N}$$
$$p(g_j^i|c_r) = \frac{|g_j^i \cap c_r|}{|c_r|}$$
$$p(g_j^i) = \frac{|g_j^i|}{N}$$

**Preprocessing für $k$-Means**

- Wir können $k$-Means anwenden, wenn wir neue (numerische) Attribute formen! Wir müssen die Anzahl der Cluster vorgeben.
- Eigentlich haben wir die schon… für jede Eingabe $\varphi$: für jedes cluster $g$ ein binäres Attribut $\delta(g_1, \varphi_i(x)) = 1$, falls $\varphi_i(x) = g_1, \delta(g_1, \varphi_i(x)) = 0$ sonst
- Einträge sind $[-1, 0[$, falls $\varphi_i(x) \neq g_j$, sonst $]0, 1]$

| | $g_1$ | … | $g_K$ |
|---|---|---|---|
| $x_1$ | $\delta(g_1, \varphi_i(x_i)) - p(g_1)$ | | $\delta(g_K, \varphi_i(x_1)) - p(g_K)$ |
| … | | | |
| $x_N$ | $\delta(g_1, \varphi_i(x_N)) - p(g_1)$ | | $\delta(g_K, \varphi_i(x_N)) - p(g_K)$ |

**Fragen**

Kann aus mehreren sehr schwachen clusterings per Konsensusfunktion ein sehr gutes clustering gemacht werden?

- Einfachste Eingabe-clusterings – wieviele?
- Konsensusfunktionen – wann ist welche besser?
  - Ko-Assoziation
  - Nutzen

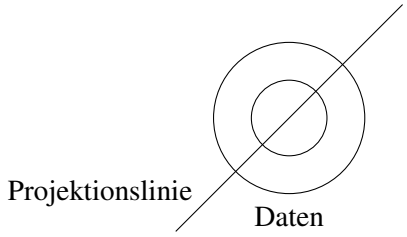
Fehlermaß bei klassifizierten Daten: wieviele falsch eingeordnet?

- Iris, 3 Klassen, 150 Beobachtungen
- 2 Spiralen, 2 Klassen, 200 Beobachtungen
- 2 Halbringe, 2 Klassen, 400 Beobachtungen (ungleich verteilt)
- Galaxie, 2 Klassen, 4190 Beobachtungen

**Experimente Ko-Assoziation**

Einfachste Eingabe-clusterings:

- Projektion der Beobachtungen auf 1 Dimension, zufällig ausgewählt
- $k$-Means auf den projizierten Daten (schnell)




Konsensusfunktion Ko-Assoziation mit

- single-link,

  i. e. $\min\{dist(a,b)\}, a \in c_2, b \in c_2$

  bester Alg. bei Spiralen, Halbringen, schlecht bei Iris
- complete link

  i. e. $\max\{dist(a,b)\}$

  gut bei Iris, schlecht ansonsten
- average link

  i. e. avg$\{dist(a,b)\}$

  gut bei Iris, schlecht ansonsten

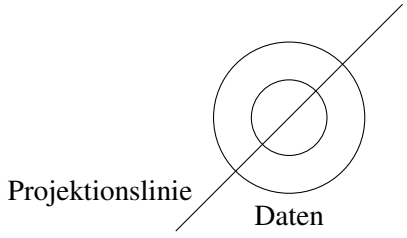
Bei Anzahl der Eingabe-clusterings $> 50$ sind die Ergebnisse besser als bestes einmal clustern der Originaldaten:

- z.B. Halbringe bestStandard 5,25% Fehler, Ensemble-Clustering 0 Fehler

**Experimente Nutzen**

Einfachste Eingabe-clusterings:

- Projektion der Beobachtungen auf 1 Dimension, zufällig ausgewählt
- $k$-Means auf den projizierten Daten (schnell)




Konsensusfunktion Nutzen

- $k$-Means
- bester Alg. bei Galaxie

Günstig bei

- vielen Beobachtungen (etwa $N > 150$)
- wenigen Eingabe-clusterings (etwa $H < 50$)

Laufzeit in $O(kNH)$

**Was wissen Sie jetzt?**

- Cluster Ensembles formen aus mehreren Eingabe-clusterings ein neues clustering
- Der Ansatz von [15] zeigt, dass
  - Ensembles von vielen, sehr einfachen clusterings besser sein können als ein „anspruchsvolles“ clustering.
  - Konsensusfunktionen wiederum cluster-Algorithmen sind, wenn man die Eingabe-clusterings zu Merkmalen macht.
- Sie wissen, wie man aus Eingabe-clusterings
  - Ko-Assoziationsmatrix oder
  - Matrizen mit binären Merkmalen macht (s. [9])

**Ansatz von [13]**

Informationstheoretische Konsensusfunktion

- Wechselseitige Information zweier clusterings  Normalized mutual information

$$NMI(\phi_a, \phi_b) = \frac{2}{N} \sum_{i=1}^{k(a)} \sum_{j=1}^{k(b)} |g_i^a \cap g_j^b| \log_{k(a)k(b)} \left( \frac{|g_i^a \cap g_j^b|}{|g_i^a||g_j^b|} \right)$$

- Der beste Konsens hat am meisten gemeinsam mit den meisten Eingabe-clusterings:

$$\sigma = \arg\max_{\sigma'} \sum_{i=1}^{r} NMI(\sigma', \phi_i)$$

- Alle möglichen clusterings $\sigma'$ aufzählen und testen? Zu aufwändig!

**Hypergraph als Repräsentation**

Ein Clustering $c$ ist ein Vektor  Jedes cluster von jedem clustering wird nun zu einer Hyperkante, die clusterings sind die Hyperknoten.

| | $c_1$ | $c_2$ | $c_3$ | $\Rightarrow$ | $g_1^1$ | $g_2^1$ | $g_3^1$ | $g_1^2$ | $g_2^2$ | $g_3^2$ | $g_1^3$ | $g_2^3$ | $g_3^3$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 1 | 2 | 1 | | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| $x_2$ | 1 | 2 | 1 | | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| $x_3$ | 1 | 2 | 2 | | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| $x_4$ | 2 | 3 | 2 | | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| $x_5$ | 2 | 3 | 3 | | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| $x_6$ | 3 | 1 | 3 | | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| $x_7$ | 3 | 1 | 3 | | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

**Subspace Clustering**

- Jedes clustering verwendet verschiedene Merkmale.
- Ensemble Clustering wird nun verwendet, um diese zusammenzuführen.

| Daten | Dimensionalität Subspace | Anzahl clusterings | Qualität Ensemble NMI($\sigma, \varphi_l$) | Beste NMI in einem Subspace |
|---|---|---|---|---|
| 2D2K | 1 | 3 | 0,68864 | 0,68864 |
| 8D5K | 2 | 5 | 0,98913 | 0,76615 |
| Pendig | 4 | 10 | 0,59009 | 0,53197 |
| Yahoo | 128 | 20 | 0,38167 | 0,21403 |

**Distributed Clustering**

Eingabe-clusterings stammen von verschiedenen Quellen.

- Horizontale Aufteilung: alle Quellen verwenden die selben Attribute, z.B. Filialen einer Firma.
- Vertikale Aufteilung: die Quellen verwenden unterschiedliche Attribute, z.B. private Benutzer in einem peer-to-peer Forum zu Austausch von Filmen.

Man kann Ensemble clustering direkt anwenden.

**Erstellen einer Ähnlichkeitsmatrix**

$N \times N$ mit Ähnlichkeitseinträgen ist auf der $H$ für $r$ Eingabe-clusterings jetzt leicht als Matrizenmultiplikation durchführbar:

$$Sim = \frac{1}{r} H H^T$$

- Beliebigen cluster-algorithmus anwenden.
- [13] haben zwei weitere spezielle Algorithmen für clustering von Hypergraphen entwickelt.

**Distributed Clustering**

Verschiedene Quellen liefern Eingabe-Clusterings.

- Horizontale Aufteilung: alle Quellen verwenden die selben Attribute, aber andere Beobachtungen, z.B. Filialen einer Firma.
- Vertikale Aufteilung: jede Quelle verwendet andere Attribute, z.B. Benutzer in einem peer-to-peer Netz über Filme.

Wir können direkt Ensemble Clustering verwenden. Aber:

- Aufwändig, alle Beobachtungen zusammen zu führen!
- Datenschutz (privacy)!

**Datenschutz**

- Es sollen nicht (alle) Daten bekannt gemacht werden, sondern nur die Verteilung in den Daten.
- Die Daten werden also durch lokale Modelle generalisiert. Das globale Modell wird aus diesen Modellen, nicht aus den ursprünglichen Daten gebildet.
- Nach den lokalen Verteilungen werden künstlich Beispiele erzeugt.
- Globales Modell = Mischung von Gauss-Modellen!

**Distributed Model-Based Clustering**

Gegeben:

- $\{X_i\}, i = 1, \ldots, n$ Datenquellen
- $\{\varphi_i\}m, i = 1, \ldots, n$ unterliegende (lokale) Modelle
- $\{v_i\}, i = 1, \ldots, n$ nicht-negative Gewichtung der Modell

Finde das optimale globale Modell $\sigma = \arg\min Q(\sigma')$, wobei $Q$ ein Fehlermaß (z.B. KL) und $\sigma'$ aus einer Familie von Modellen ist.

**Durchschnittsmodell finden**

Das DMC-Problem ist äquivalent zu dem, ein Modell zu finden, das nahe am Durchschnitt der lokalen Modelle bezüglich der KL-Divergenz ist.

- KL-Divergenz: $KL(x, y) = \sum_{i=1}^{d} x_i \log\left(\frac{x_i}{y_i}\right)$
- Durchschnitt aller lokalen Modelle so dass $p_\sigma(x) = \sum_{i=1}^{n} v_i p_{\phi_i}(x)$

Finde das Modell, das dem Durchschnitt der lokalen Modelle bzgl. KL-Divergenz am nächsten ist!

**Approximierender Algorithmus für DMC**

**Input** $\{\varphi_i\}, i = 1, \ldots, n$ mit $\{v_i\}, i = 1, \ldots, n$ Familie $F$: Gaussian Mixture Models

**Output** $\sigma_a \approx \underset{\sigma' \in F}{\arg\min} \sum_{i=1}^{n} v_i D_{KL}(\phi_i, \sigma')$

1. Erzeuge ein Durchschnittsmodell, so dass $p_{\bar{\sigma}}(x) = \sum_{i=1}^{n} v_i p_{\phi_i}(x)$
2. Ziehe daraus eine Stichprobe $X'$
3. Wende EM an, um $\sigma_a$ zu erhalten mit

$$\sigma_a = \underset{\sigma' \in F}{\arg\max} L(X', \sigma') = \arg\max \frac{1}{m} \sum_{j=1}^{m} \log\left(p_{\sigma'}(x_j)\right)$$

**Mixture Models**

Gewichtete Linearkombinationen von $K$ einzelnen Verteilungen (Dichten) mit Parametern $\theta$,

$$p(x) = \sum_{k=1}^{K} p_k \left(x \mid \theta_k\right) \pi_k$$

$\pi_k$: Die Wahrscheinlichkeit, dass eine zufällig gezogene Beobachtung aus der $k$-ten Verteilung kommt.

**Hintergrung**

Zur Erinnerung:

- Mixture Models
- log-likelihood schätzen für Mixture Models
- EM

**log-likelihood für Mixture Models**

Wahrscheinlichkeit für Daten $D$ gegeben Parameter $\theta$, wobei wir die Werte $H$ eines Attributs für die Beobachtungen nicht kennen.

$$\log p(D \mid q) = \log \sum_{H} p(G, H \mid \theta)$$

Wenn wir immerhin die Verteilung $Q(H)$ kennen:

$$\log \sum_{H} Q(H) \frac{p(D, H \mid \theta)}{(H)} \geq \sum_{H} Q(H) \log \frac{p(D, H \mid \theta)}{Q(H)} = BL(Q, \theta)$$

**Iteratives Vorgehen**

- likelihood($\theta$) bei unbekannten Werten $H$ schwer zu finden!
- $BL(Q, \theta)$ ist untere Schranke von likelihood($\theta$).
- Abwechselnd:
  - Maximieren von $BL(Q, \theta)$ bei festen Parametern $\theta$

$$Q^{k+1} = \arg\max_{Q} BL(Q^k, \theta^k)$$

  - Maximieren $BL(Q, \theta)$ bei fester Verteilung $Q = p(H)$

$$\theta^{k+1} = \arg\max_{\theta} BL(Q^k, \theta^k)$$

Das macht EM-Vorgehen.

**EM-Vorgehen [3]**

Estimation:

- Schätze Wahrscheinlichkeit, dass eine Beobachtung zu einem cluster gehört
- Speichere Matrix: Beobachtungen $X$ cluster

Maximization

- Passe clustering den Wahrscheinlichkeiten an.
- Bis es keine Verbesserung mehr ergibt: gehe zu E-Schritt.

**Approximierender Algorithmus für DMC**

**Input** $\{\varphi_i\}, i = 1, \ldots, n$ mit $\{v_i\}, i = 1, \ldots, n$ Familie $F$: Gaussian Mixture Models

**Output** $\sigma_a \approx \arg\min_{\sigma' \in F} \sum_{i=1}^{n} v_i D_{KL}(\phi_i, \sigma')$

1. Erzeuge ein Durchschnittsmodell, so dass $p_{\bar{\sigma}}(x) = \sum_{i=1}^{n} v_i p_{\phi_i}(x)$
2. Ziehe daraus eine Stichprobe $X'$
3. Wende EM an, um $\sigma_a$ zu erhalten mit

$$\sigma_a = \arg\max_{\sigma' \in F} L(X', \sigma') = \arg\max \frac{1}{m} \sum_{j=1}^{m} \log\left(p_{\sigma'}(x_j)\right)$$

**Privacy**

- Ein feines Gauss-Modell mit wenig Varianz, womöglich zentriert an jedem Datenpunkt, gibt dann doch genau die Daten wieder.
  - Auch wenn nur das Modell weitergegeben wird – keine Anonymität!
- Ein grobes Modell mit nur einem Gauss-Kegel und hoher Varianz hat hohe Anonymität – niedrige Wahrscheinlichkeit, die Daten aus dem Modell zu generieren.
- Immerhin wird das globale Modell auch aus mittelfeinen lokalen Gauss-Modellen noch ordentlich...

**Was wissen Sie jetzt?**

- Sie kennen das Kriterium der wechselseitigen Information.

$$NMI(\phi_a, \phi_b) = \frac{2}{n} \sum_{i=1}^{k(a)} \sum_{j=1}^{k(b)} |g_i^a \cap g_j^b| \log_{k(a)k(b)} \left( \frac{|g_j^a \cap g_j^b|}{|g_i^a||g_j^b|} \right)$$

- Das kann man zur Konsensusfunktion machen.

- Sie kennen die Hypergraph-Representation der Eingabe-clusterings.
- Ensemble clustering ist eine Art, clusterings mit unterschiedlichen Attributen zusammen zu führen. Es gibt noch andere Subspace clustering Algorithmen!
- Ensemble clustering ist eine Art, verteilte Datenbestände zu clustern. Es gibt noch andere!
- Wenn man nicht die Daten austauscht, sondern Modelle der Daten, schützt man die Daten und wahrt Anonymität.

## 9.4 Organisation von Sammlungen

**Organisation von Sammlungen**

Sammlungen von Fotos, Musik, Filmen bevölkern PCs und das Internet. Sie sind organisiert

- in Taxonomien nach vorgegebenen Kriterien
  - iTunes: Genre, Artist, Album, Jahr
- in Taxonomien nach eigenen Kriterien
  - flickR: Sammlung, Album, Gruppen – annotiert wird mit eigenen tags.
- einfache Dateien, evtl. mit Benutzeroberfläche
  - iPhoto: Ereignisse, jedes Bild kann annotiert werden.

**Wie organisieren Menschen Medien?**

- Studie von Jones, Cunningham, Jones (2004): Studenten wurden befragt, wie sie ihre CDs, DVDs, Bücher organisieren.
  - Nachttisch, spezieller Schrank, Auto, Küche
  - Gelegenheiten zur Nutzung
  - Aktualität, Anschaffungszeitpunkt
- Studie von Vignoli (2004): Ordnung digitaler Musik auf PCs wurden untersucht.
  - Meist wurden hierarchische Strukturen aufgebaut.
  - Es gibt immer einen Ordner mit nicht einsortierter Musik.
- Studie PG 461 "Kollaboratives Strukturieren von Multimediadaten für Peer-to-Peer-Netze"
  - Verschiedene Aspekte: Gelegenheiten ("beim Autofahren", "Dinner", ”Party"), Personen (”für Susie"), Erinnerungen (”Sommer03"), Stimmungen, Tempi, Genres
  - Wieder gibt es Ordner mit nicht einsortierter Musik.

**Automatisches Sortieren von Mediensammlungen**

- Medien sollen hierarchisch strukturiert werden.
- Die Taxonomien sollen personalisiert sein.
  - Die Bezeichner sind unterschiedlich: was dem einen ”fröhliche Tanzmusik”, gehört bei dem anderen unter ”Depression” (The Cure).
  - Bereiche, die einer fein strukturiert, fasst der andere zusammen.
  - Verschiedene Benutzer stellen verschiedene Mengen als ähnlich betrachteter Medien zusammen.
- Derselbe Benutzer verwendet mehrere, unterschiedliche Hierarchien (*Aspekte*), die teilweise gleiche Medien abdecken.
- Die Einsortierung neuer Medien soll automatisch erfolgen.
- Die Struktur soll automatisch erweitert werden, ohne den Benutzer zur bevormunden.

### 9.4.1 Web 2.0

**Web 2.0**

- Semantic Web:
  - Semantische Beschreibung
  - Vorgegebene, allgemeine Ontologie
  - Logische Beschreibungssprache
  - top-down Modellierung
- Web 2.0
  - Freies Tagging der Benutzer
  - Entstehende *Folksonomies*
  - Statistische Methoden
  - Empfehlungssysteme

**Sammlungen im Web 2.0**

- Verschiedene Benutzer laden ihre Medien hoch.
- Verschiedene Benutzer annotieren ihre Medien.
- Kollaborative Empfehlung:
  - Ein Benutzer sind einander ähnlich, wenn sie ähnliche Mengen von Medien ausgewählt haben.
  - Medien sind einander ähnlich, wenn sie in Sammlungen ähnlicher Benutzer vorkommen.
  - Meist werden nur flache Medienmengen betrachtet (Amazon, Last.fm). Es werden auch nur Listen von Medien empfohlen.
- Für die automatische Unterstützung der Strukturierung reicht das nicht.

### 9.4.2 Clustering verteilter Daten

**Clustering Mediensammlungen**

- **Ziel:** Hierarchisches Clustering erzeugt für einen Benutzer anhand seiner und der Clusterings anderer Benutzer je Aspekt mehrere Taxonomien zur Auswahl.
  - Wie kann das Benutzer gegebene Clustering beibehalten und nur ergänzt werden? –> Supervised Clustering
  - Wie kann ein Benutzer von den Strukturierungen anderer Benutzer profitieren? –> Distributed Clustering, Ensemble Clustering
  - Wie kann das Verfahren mehrere alternative Clusterings zur Auswahl anbieten? –> Nonredundant Clustering

**Supervised Clustering**

- **Constraint Clustering** (Cohn, Caruana, McCallum 2003) beachtet bei der Optimierung vom Benutzer gegebene Nebenbedingungen
  - $must - link(\vec{x_g}, \vec{x'_g})$, d.h. $\vec{x_g}, \vec{x'_g}$ müssen im selben Cluster sein;
  - $cannot - link(\vec{x_g}, \vec{x_h})$, d.h. $\vec{x_g}, \vec{x_h}$ dürfen nicht im selben Cluster sein.
- **Supervised Clustering** (Finley, Joachims 2005) beachtet bei der Optimierung als Nebenbedingungen, dass einige Cluster mit zugeordneten Beobachtungen vorgegeben sind: $C(i) = k$ für $\vec{x_i}$, i=1,...,M, M<N $C_k, k = 1, ..., L, L \leq K$
- Leider nur für flache Clusterings und nicht für mehrere, verteilte gegebene Clusterings!

**Distributed Clustering**

- Verteilte Daten sollen gruppiert werden.
- Horizontale Verteilung:
  - Alle Daten haben die selben Merkmale, sind aber auf verschiedene Rechner verteilt.
  - Kein Datum ist mehr als einem Rechner zugeordnet.
  - Typisches Beispiel: Filialen eines Geschäfts.
- Vertikale Verteilung:
  - Daten der verschiedenen Rechner haben unterschiedliche Merkmale.
  - Das selbe Objekt ist auf mehreren Rechnern zu finden.
  - Typisches Beispiel: Mediensammlungen Web 2.0.
- Ziel ist ein *Konsens-Modell* als gemeinsames Clustering für alle Daten.
- Das ist nicht das Ziel bei der Strukturierung persönlicher Mediensammlungen!

**Ensemble Clustering**

- Ensemble Clustering kombiniert eine Menge gegebener Clusterings (Strehl, Ghosh 2002).
- Alle Clusterings decken die selbe Menge von Beobachtungen ab.
  - Zusätzliches Ähnlichkeitsmaß: kommen gemeinsam in einem Cluster vor (Topchy, Jain, Punch 2003);
  - Zuordnung zu einem gegebenen Cluster als zusätzliches Merkmal einer Beobachtung – dann in diesem Raum k-Means anwenden!
- Wieder wird ein Konsens-Modell erzeugt!

**Nonredundant Clustering**

- Gegeben ein Clustering $C(i) = k$ für Beobachtungen $\vec{x_i}, i = 1, ..., N$ und Cluster $C_k, k = 1, ..., K$
- finde ein alternatives Clustering $C'$, das möglichst orthogonal zu $C$ ist. (Gondek, Hofmann 2004)
- Das Verfahren erhält keine gegebenen Strukturierungen, sondern bietet Alternativen zum gesamten Clustering an.

**Es gibt noch kein geeignetes Verfahren für das Strukturieren persönlicher Sammlungen im Web 2.0**

- Bisherige Ansätze reichen nicht aus:
  - Supervised clustering ist noch nicht geeignet für hierarchische Strukturen und die Eingabe mehrerer Clusterings.
  - Distributed clustering und Ensemble Clustering erstellen ein Konsens-Modell, das die eigene Annotation von Benutzern überschreiben würde.
  - Nonredundant clustering erhält in den Alternativen nicht das gegebene Clustering.
- Wir mussten also ein eigenes Verfahren entwickeln: Localized Alternative Clustering of Ensembles

## 9.5 LACE

**Lernaufgabe Localized Alternative Clustering of Ensembles**

- Wir sprechen jetzt statt von der Zuordnung $C(i) = k$ einer Beobachtung $\vec{x_i}$ zu einem Cluster $C_k$ von dem Clustering $\varphi_i$ von einer Menge von Beobachtungen $S_i$ auf ein Cluster $G_i$.
- Gegeben eine Menge $S \subseteq X$, eine Menge von Clusterings $I \subseteq \{\varphi_i : S_i \rightarrow G_i\}$ und eine Qualitätsfunktion

$$q : 2^{\Phi} \times 2^{\Phi} \times 2^{S} \rightarrow \mathcal{R} \tag{9.13}$$

*localized alternative clustering ensembles* findet Clusterings $O \subseteq \{\varphi_i | \varphi_i : S_i \rightarrow G_i\}$ so dass die Qualität $q(I, O, S)$ maximiert wird und für jedes $\varphi_i \in O$ gilt, dass $S$ Teil seines Ursprungsbereichs ist: $S \subseteq D_{\varphi_i}$.

**$\varphi$ als hierarchisches Clustering**

- Die Cluster sollen nicht auf einer Ebene liegen, sondern eine Taxonomie bilden.
- Die unterste Ebene enthält Mengen von Beobachtungen.
- Jede Ebene enthält Cluster, die die Cluster der Ebene darunter subsummieren: jeder Teilbaum von Clustern ist eine Taxonomie.
- Die oberste Ebene enthält ein Cluster mit allen Beobachtungen.
- Man unterscheidet ein Vorgehen bottom-up (agglomerativ) und top-down (aufteilend).
- $\varphi_i : S_i \to G_i$ soll die Menge $S_i$ hierarchisch aufteilen, d.h. $G_i$ soll eine Hierarchie von Clustern sein.

**Zur Erinnerung: Agglomeratives Clustering**

- Stufenweise werden Beobachtungen zu übergeordneten Clustern verschmolzen.
- Grundlage ist die *Unähnlichkeit von Clustern*: solche mit geringster Unähnlichkeit werden verschmolzen.
- Die Unähnlichkeit $d(G, H)$ der Cluster $G, H$ wird berechnet durch den Abstand $d_{gh} = D(\vec{x_g}, \vec{x_h})$, wobei $\vec{x_g} \in G, \vec{x_h} \in H$.
- Welche Beobachtungen genutzt werden, macht den Unterschied zwischen den 3 wichtigsten Maßen zur Cluster-Unähnlichkeiten aus.
  - Single Linkage Clustering: Die Unähnlichkeit zwischen Cluster $G$ und $H$ ist die Unähnlichkeit der nächsten Punkte.
  - Complete Linkage Clustering: Die Unähnlichkeit zwischen Cluster $G$ und $H$ ist die Unähnlichkeit der entferntesten Punkte.
  - Average Linkage Clustering: Die Unähnlichkeit zwischen Cluster $G$ und $H$ ist die durchschnittliche Unähnlichkeit aller Punkte in $G$ von allen in $H$.

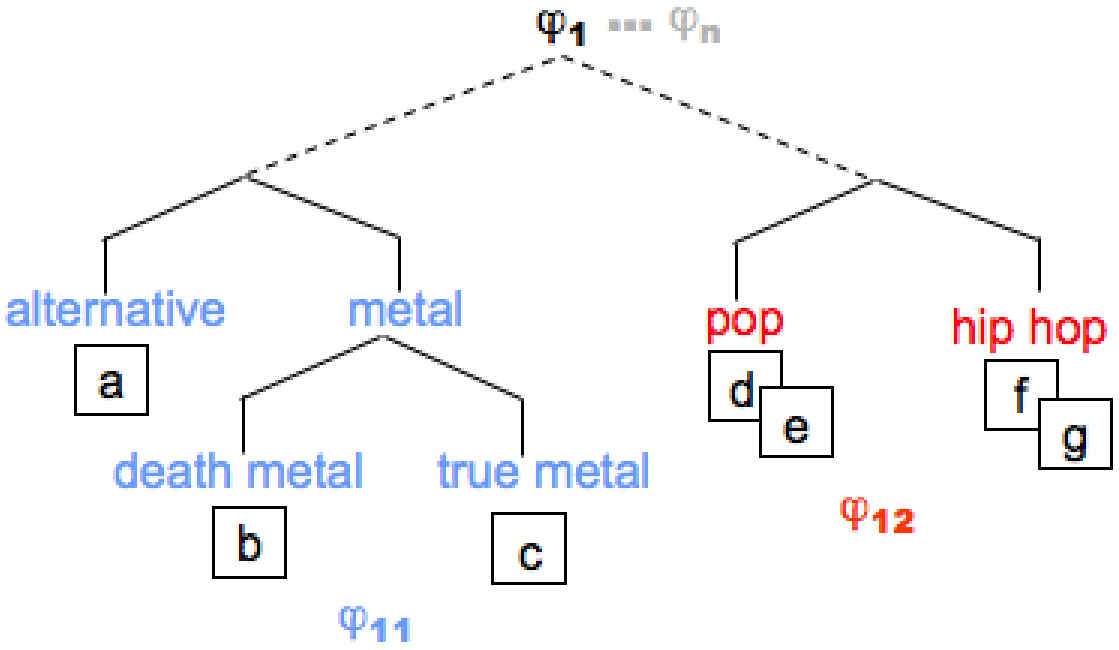
**Erweiterung eines Clustering**

Wir wollen ein gegebenes Clustering erweitern, d.h.:

- Bestehende Zuordnungen bleiben.
- Bisher abgedeckte Beobachtungen bleiben abgedeckt.
- Zusätzliche Beobachtungen werden abgedeckt.

**Erweiterte Funktion**

$\varphi'_i : S'_i \to G_i$ ist die *erweiterte Funktion* für $\varphi_i : S_i \to G_i$, wenn $S_i \subset S'_i$ und $\forall \vec{x} \in S_i : \varphi_i(\vec{x}) = \varphi'_i(\vec{x})$.




**Beutel von Clusterings**

Wir wollen die noch nicht strukturierten Beobachtungen in $S$ durch vorhandene Clusterings $\varphi_1, ..., \varphi_m$ abdecken.

**Beutel von Clusterings**

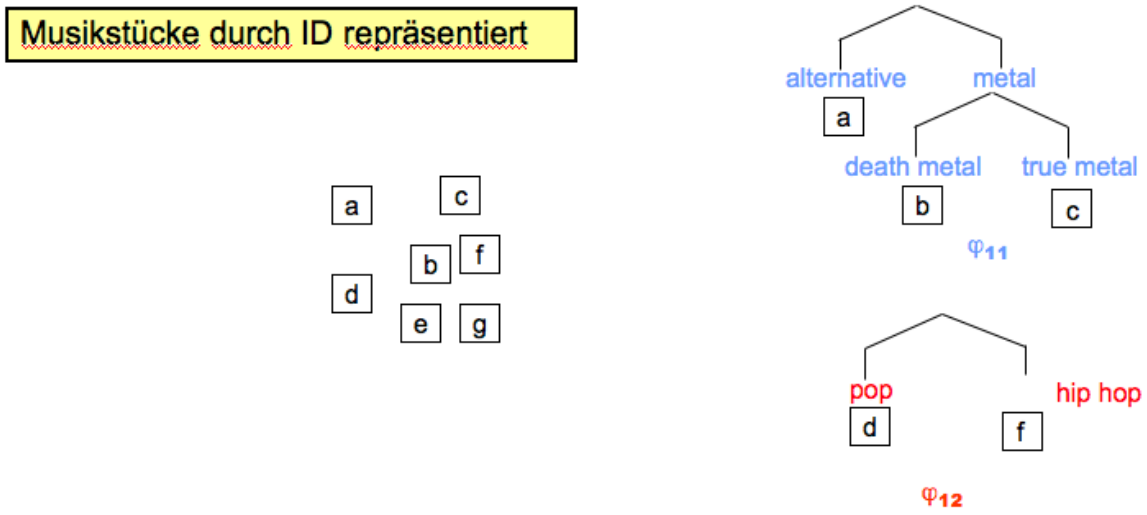
Sei $I$ eine Menge von Clusterings. Ein *Beutel von Clusterings* ist eine Funktion

$$\varphi_i(\vec{x}) = \begin{cases} \varphi'_{i1}(x), & \text{wenn } \vec{x} \in S'_{i1} \\ \vdots & \vdots \\ \varphi'_{ij}(x), & \text{wenn } \vec{x} \in S'_{ij} \\ \vdots & \vdots \\ \varphi'_{im}(x), & \text{wenn } \vec{x} \in S'_{im} \end{cases} \tag{9.14}$$

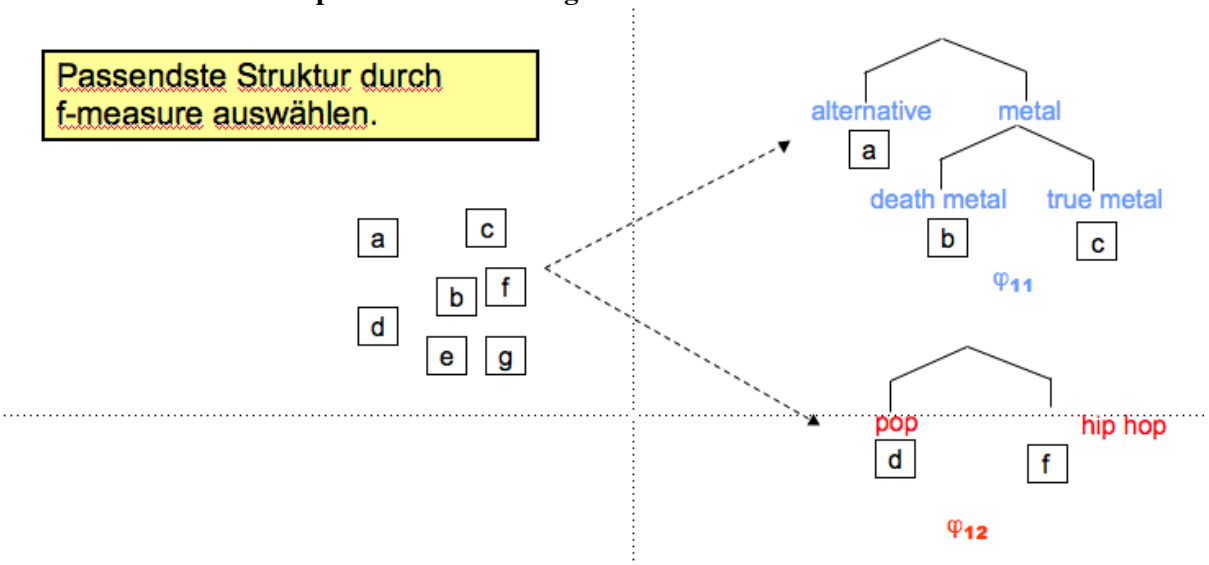
wobei jedes $\varphi'_{ij}$ eine Erweiterung eines $\varphi_{ij} \in I$ ist und $\{S'_{i1}, \ldots, S'_{im}\}$ ist eine Partitionierung von $S$.

**Beutel von Clusterings im Bild**

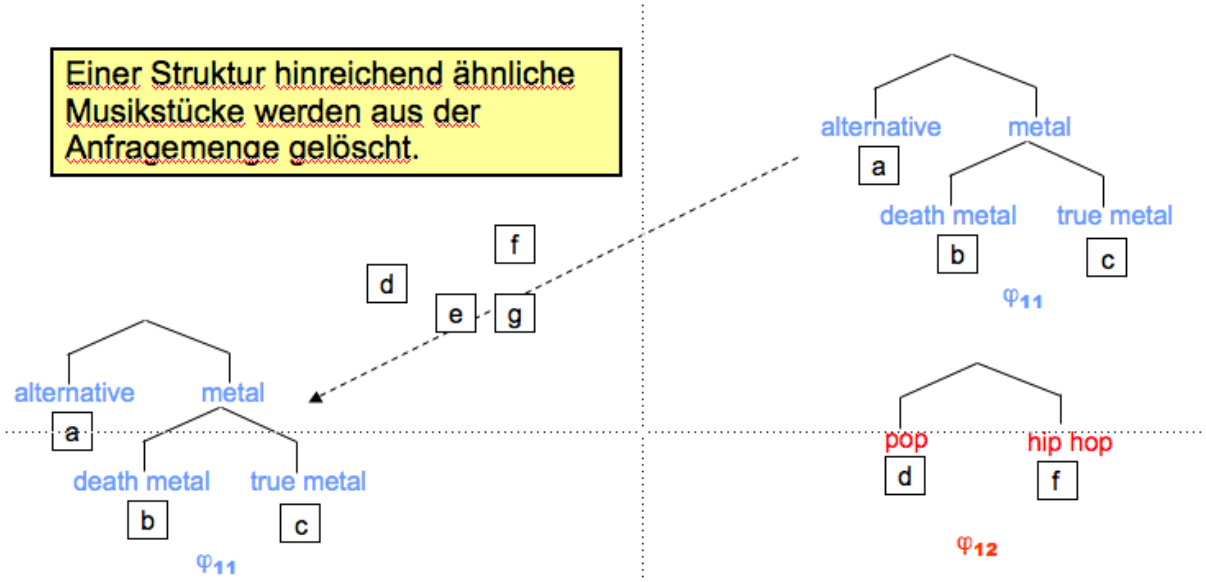
**LACE in Bildern - 1: Nicht eingeordnete Stücke, Clusterings anderer Benutzer**



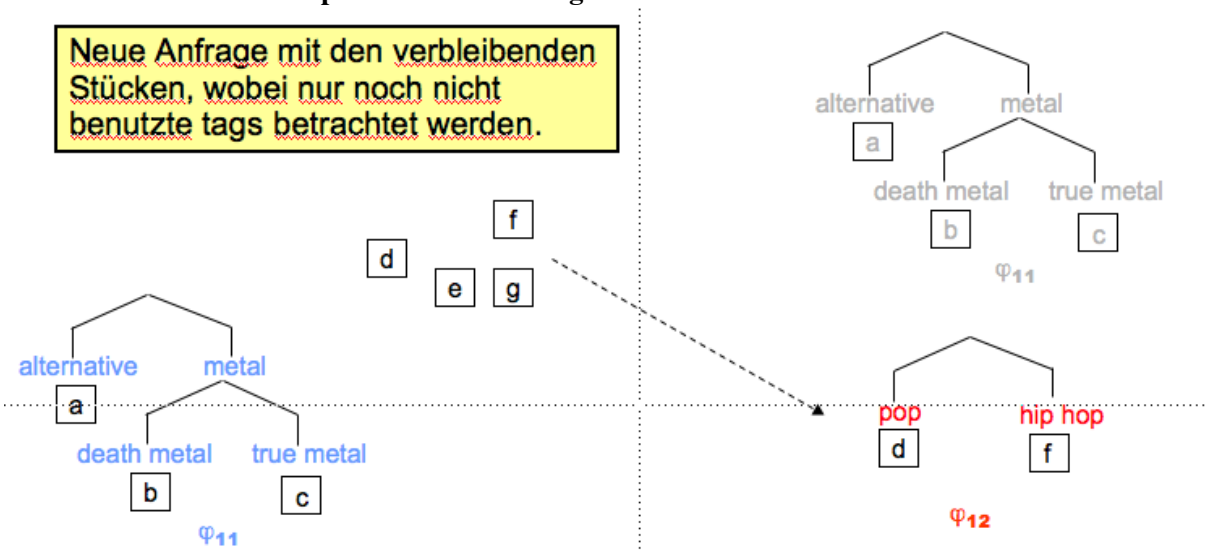

**LACE in Bildern - 2: Finden passender Clusterings**



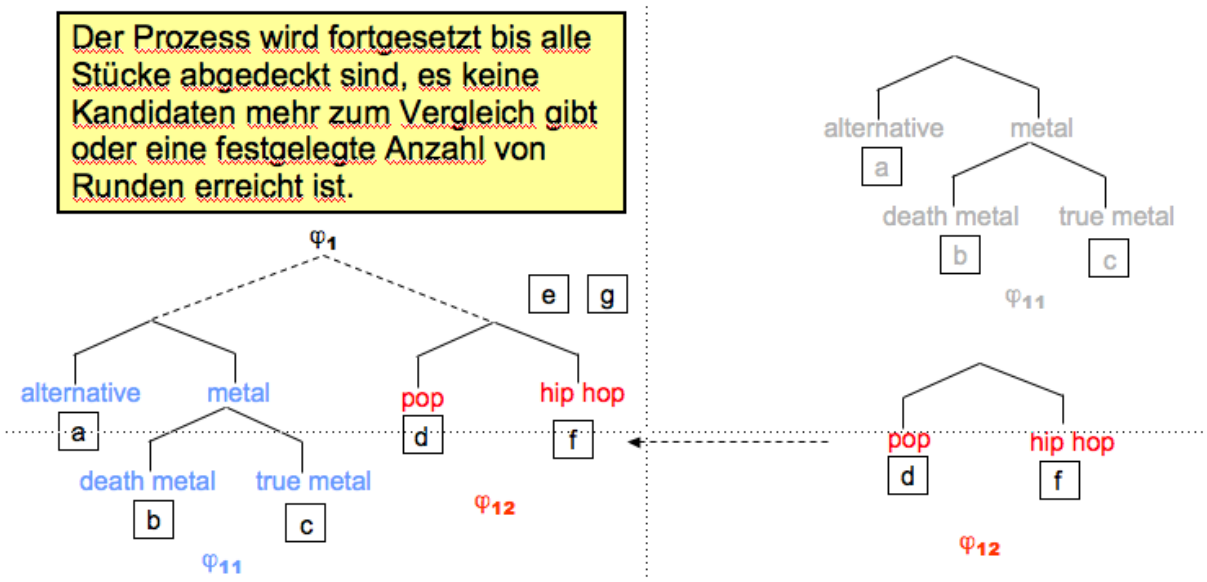

**LACE in Bildern - 3: Löschen abgedeckter Stücke**



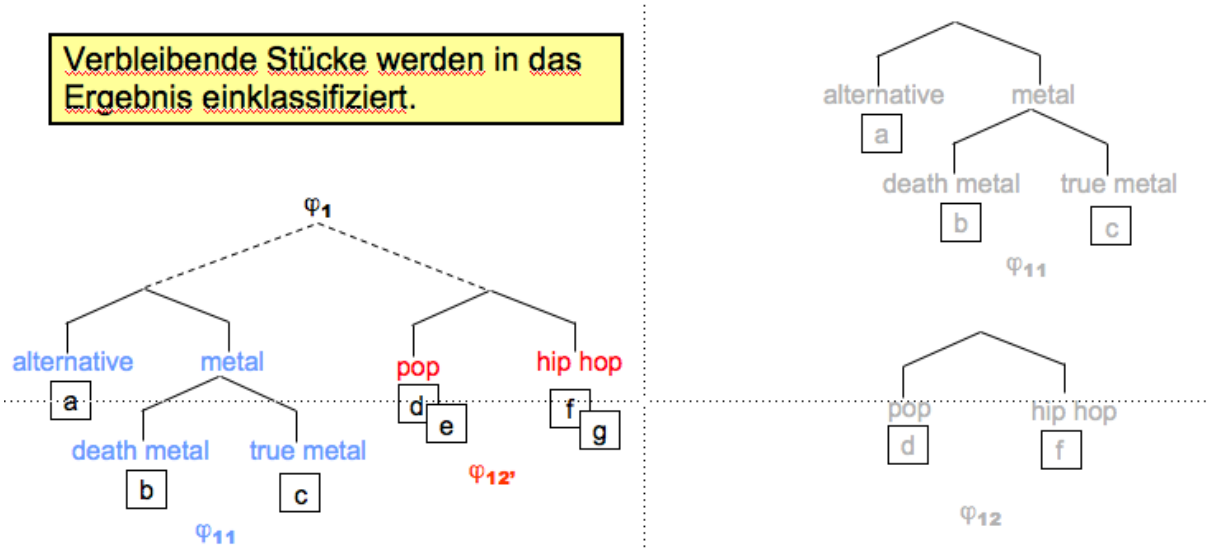

**LACE in Bildern - 4: Finden passender Clusterings für den Rest**



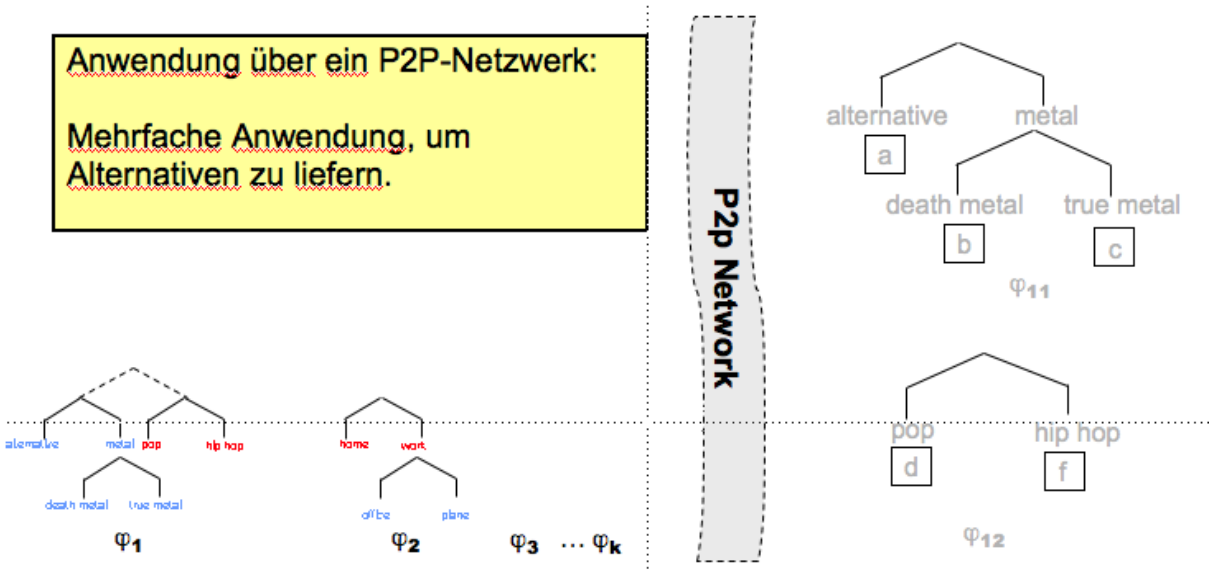

**LACE in Bildern - 5: Abbruchbedingung für das sequentielle Abdecken**



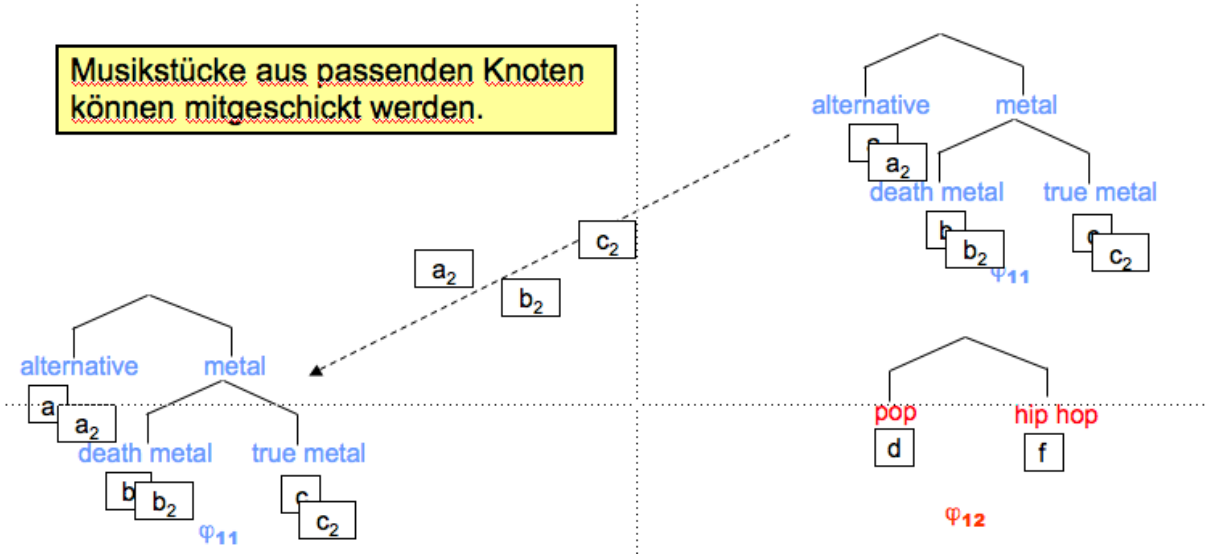

**LACE in Bildern - 6: Klassifkation von Stücken in neue Struktur**




**LACE in Bildern - 7: Posten der abzudeckenden Stücke ins P2P-Netz, Empfangen der passenden Clusterings**




**Personalisierte Empfehlungen**




**Qualitätsfunktion für Clustering und Menge von Objekten**

- Bei der Repräsentation eines Clusters durch *well-scattered points* ist $Z_{\varphi_i}$ die Menge von Beobachtungen, die $\varphi_i$ beschreibt. $\beta$ sei eine Gewichtung, die Precision und Recall ins Verhältnis setzt:
- Precision:

$$prec(Z_{\varphi_i}, S) = \frac{1}{|Z_{\varphi_i}|} \sum_{\vec{z} \in Z_{\varphi_i}} max\{sim(\vec{x}, \vec{z}) | \vec{x} \in S\}.$$

- Recall:
$$rec(Z_{\varphi_i}, S) = \frac{1}{|S|} \sum_{\vec{x} \in S} max \{sim(\vec{x}, \vec{z}) | \vec{z} \in Z_{\varphi_i}\}.$$

- F-Measure:
$$q_f^*(Z_{\varphi_i}, S) = \frac{(\beta^2 + 1) rec(Z_{\varphi_i}, S) prec(Z_{\varphi_i}, S)}{\beta^2 rec(Z_{\varphi_i}, S) + prec(Z_{\varphi_i}, S)}. \quad (9.15)$$

**Basisalgorithmus Sequenzielles Abdecken**

- $O = \emptyset$, $J = I$
- WHILE($|O| < max_{alt}$)
  - $S_u = S$, $B = \emptyset$, $step = 0$
  - WHILE($(S_u \neq \emptyset) \wedge (step < max_{steps})$)
    - $\varphi_i = \arg\max_{\varphi \in J} q_f^* (Z_\varphi, S_u)$
    - $S_u = S_u \setminus \{\vec{x} \in S_u | \vec{x} \sqsubset_\alpha \varphi_i\}$
    - $B = B \cup \{\varphi_i\}$
    - $step = step + 1$
  - $O = O \cup \{bag(B, S)\}$
- Wobei $max_{alt}$ die maximale Anzahl an Alternativen angibt, die Funktion $bag(B, S)$ einen Beutel von Clusterings angibt, der jedem Stück $\vec{x} \in S$ das Clustering $\varphi_i \in B$ zuweist, das die zu $\vec{x}$ ähnlichsten Objekte enthält.

**Hierarchisches Vorgehen: Rekursiv Precision und Recall berechnen!**

$$prec(Z_{\varphi_i}, S) = \begin{array}{ll} \frac{|Z^*_{\varphi_i}|}{|Z_{\varphi_i}|} prec(Z^*_{\varphi_i}, S) + & updateSchritt \\ \sum_{\varphi_j \prec \varphi_i} \frac{|Z_{\varphi_j}|}{|Z_{\varphi_i}|} prec(Z_{\varphi_j}, S) & direkterNachfolger \end{array}$$

wobei $Z^*_{\varphi_i} = Z_{\varphi_i} \setminus \bigcup_{\varphi_j \prec \varphi_i} Z_{\varphi_j}$ *nur Oberknoten.*

- Die hierarchischen Funktionen $\varphi_j$ und $\varphi_i$, sind in direkter Nachfolgerelation $\varphi_j \prec \varphi_i$, gdw.

$$G_j \subset G_i$$
$$\forall \vec{x} \in S_i : \varphi_j(\vec{x}) = \varphi_i(\vec{x}) \cap G_j$$
$$\neg \exists \varphi'_i : G_j \subset G'_j \subset G_i$$

- Wenn eine optimistische Schätzung des F-measure schon am Wurzelknoten schlechter als ein Schwellwert ist, muss das Clustering nicht weiter untersucht werden!

## 9.6 Experimente mit LACE

**Daten**

- $\varphi_1, ..., \varphi_{39}$ sind 39 Taxonomien für eine Musiksammlung von 1886 Stücken.
- Es wird immer eine Taxonomie weggelassen und auf die restlichen LACE angewandt.
- Das Ergebnis wird mit der weggelassenen Taxonomie verglichen. Differenz der absoluten Tree Distance zwischen zwei Beobachtungen in beiden Taxonomien:

| S | $x_1$ | $x_2$ | ... | $x_m$ | sum of differences |
|---|---|---|---|---|---|
| $x_1$ | - | $\varphi$:1;$\varphi'$:3 | | | 2+ |
| $x_2$ | | - | | $\varphi$:1;$\varphi'$:2 | 1+ |
| ... | | | - | | |
| $x_m$ | | | | - | |
| Total | | | | | 3+ |

**Andere Kriterien und Verfahren**

- Andere Kriterien: Korrelation zwischen den Tree Distances FScore:
  - Jedes Cluster der weggelassenen Taxonomie wird mit jedem Cluster der gelernten verglichen (Precision und Recall –> F-measure) und das jeweils beste ausgewählt. Der Durchschnitt ergibt den FScore.
- Single-linkage agglomeratives Clustering
- TD: Rekursives top-down K-Means (Guan, Kulis 2004)
- Mehrfaches Starten, um zu Ensembles zu kommen, von denen stets das beste ausgesucht wird.

**Ergebnisse**

| Method | Correlation | Absolute distance | FScore |
|---|---|---|---|
| LACE | 0.44 | 0.68 | 0.63 |
| TD ensemble | 0.23 | 2.5 | 0.55 |
| single-link ensemble | 0.17 | 9.9 | 0.60 |
| random | 0.09 | 1.8 | 0.5 |

| Representation | Correlation | Absolute distance | FScore |
|---|---|---|---|
| all points | 0.44 | 0.68 | 0.63 |
| $\|Z\| = 10$ | 0.44 | 0.68 | 0.63 |
| $\|Z\| = 5$ | 0.41 | 0.69 | 0.63 |
| $\|Z\| = 3$ | 0.40 | 0.69 | 0.62 |
| centroid | 0.19 | 1.1 | 0.42 |

**Was wissen Sie jetzt?**

- Sie haben das Feld der Strukturierung von Sammlungen im Web 2.0 kennen gelernt.
- Sie kennen eine neue Lernaufgabe: lokale alternative Cluster Ensembles und einen Algorithmus dazu.
- Insbesondere haben Sie dabei gesehen, dass man aus der unüberwachten Lernaufgabe des Clusterings manchmal eine halb-überwachte machen kann:
  - Für einzelne Beobachtungen ist angegeben, ob sie im selben oder in verschiedenen Clustern landen sollen (Constraint Clustering).
  - Es soll eine bestimmte Menge von Objekten abgedeckt (strukturiert) werden (LACE).
  - Es soll eine bestimmte Struktur erhalten, aber erweitert werden (Supervised Clustering, LACE).
- Und Sie haben gesehen, wie man Strukturen anderer Benutzer (über ein P2P Netz) nutzen kann.

## 9.7 Musik als Daten

**Technische Grundlagen**

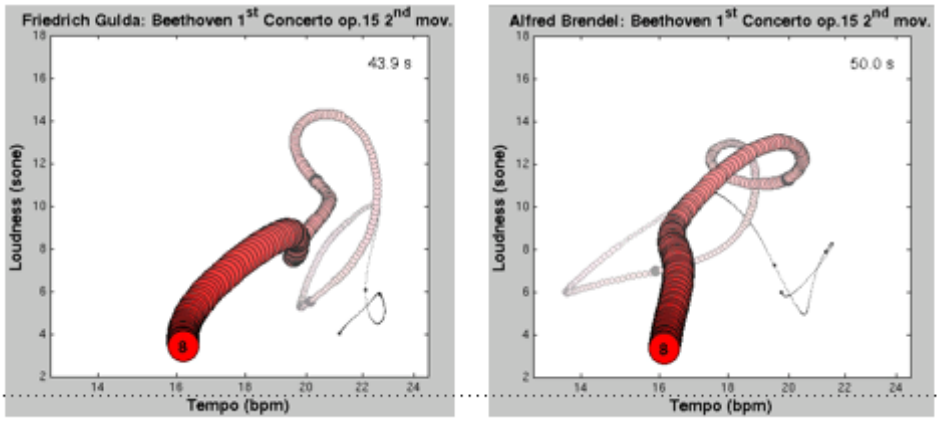
- Moving Pictures Expert Group Audio Layer 3 Karlheinz Brandenburg, TU Ilmenau, Fraunhofer Institut Standard für Musik und Filme, min. 1/12 komprimiert
- Tauschbörsen für Musik:
  - Napster 80 Mio. Benutzer, Nachfolger: Morpheus, Gnutella, KaZaA
  - KaZaA 500 Mio. Musikstücke
  - Privatsammlungen oft mehr als 10 000 Musikstücke
- Speichern, Abspielen, GUI zum Anbieten von Musik

**Arbeitsfelder – Musik**
Wissenschaftliche Untersuchung von Musik

**Arbeitsfelder – Music Information Retrieval**

- Anfragen: über ID3 tags (Metadaten), query by humming
- Indexierung: über Metadaten, über tags der Benutzer
- Navigation in Sammlungen gemäß Ähnlichkeit
- Klassifikation von Musik
- Empfehlungen

– Interpretation (Gerhard Widmer)
Der "Performance Worm": Eine Bewegung des Wurms nach rechts oben beschreibt ein gleichzeitiges Beschleunigen und Lauterwerden. Der dunkelste Punkt repräsentiert den gegenwärtigen Zeitpunkt, die Vergangenheit erscheint blasser. Typische Muster für Künstler finden.

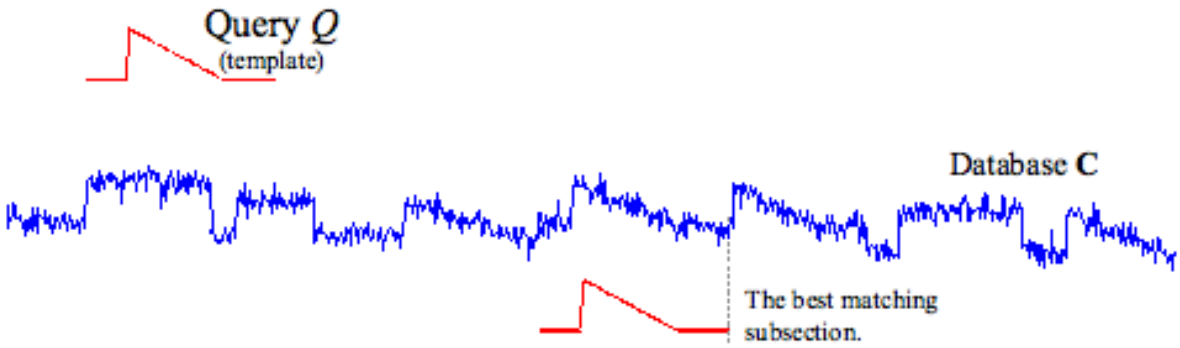



**Arbeitsfelder – Intelligente Systeme**

- Automatische Annotation von Musik
- Automatische Klassifikation von Musik nach
  - Genre (nur noch als Benchmark)
  - Benutzerpräferenzen
  - arbiträren tags (Aspekten)
- Automatische Organisation von Sammlungen
- Empfehlungen

**Technischer Kern**

- Musikdaten sind Zeitreihen der Elongation.
- Wir müssen Ähnlichkeiten von Zeitreihen erkennen. Das ist der technische Kern in fast allen Lernverfahren.
- Ähnlichkeit von Zeitreihen bisher:
  - Ähnlichkeit der Kurven
  - Dynamic Time Warping: Ähnlichkeit mit Verzerrung
- Achtung: Zeitreihenanalyse untersucht *eine* Zeitreihe und sagt neue Werte in der Zukunft voraus. Hier geht es aber um die Klassifikation oder das Clustering von *vielen* Zeitreihen. (Eamonn Keough)




**Ähnlichkeit von Zeitreihen**

- Gegeben eine Anfrage Q, eine Datenbank mit Zeitreihen C und ein Abstandsmaß,
- finde den Ort in einer Reihe in C, der Q am ähnlichsten ist.

**Dynamic Time Warping**

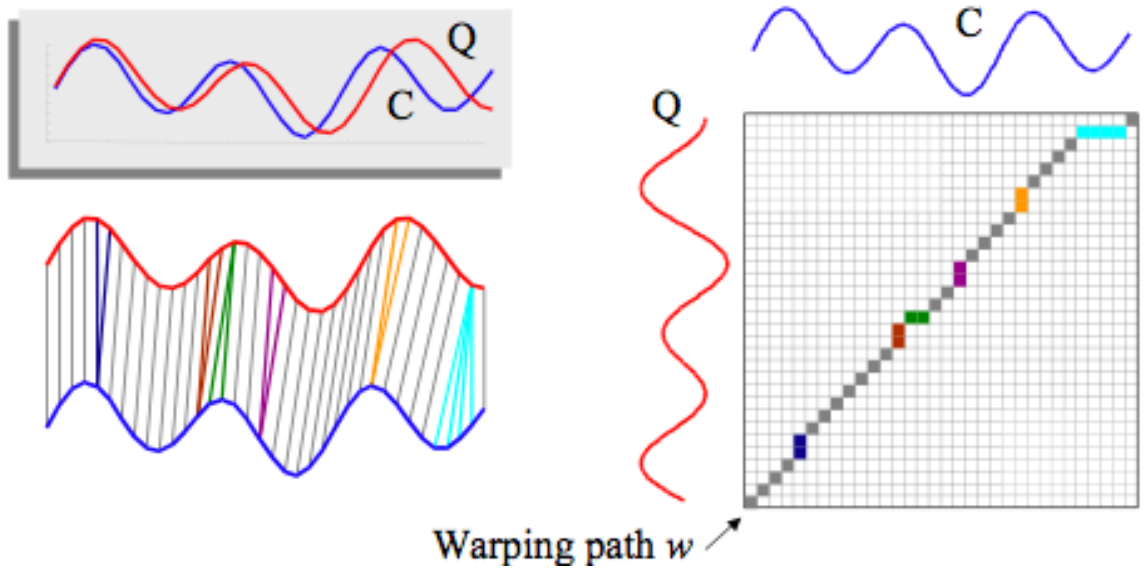
**So geht es nicht! Nötig ist die Merkmalsextraktion.**

- Musikdaten geben die Ähnlichkeit von Musik nicht wieder. Musik ist nicht ähnlich, wenn die Elongation ähnlich ist.
- Aus den Elongationsdaten müssen Merkmale extrahiert werden, nach denen die Ähnlichkeit bestimmt werden kann.
- Merkmalsextraktion ist die Voraussetzung für:
  - Annotation
  - Indexing
  - Clustering
  - Kassifikation

**Merkmalsextraktion**

- Eine Reihe von *low level descriptors* wird extrahiert:
  - Lautstärke
  - Peaks, Verhältnis vom höchsten zum zweithöchsten Peak, ...
  - Zero Crossing Rate
  - Spectral Centroid (Cepstral)
  - Mel Frequency Cepstral Coefficient (MFCC)
- Es gibt einen Merkmalssatz, der sich häufig bewährt: Tzanetakis, Dissertation 2002

$$\gamma(i,j) = d(q_i,c_j) + \min\{ \gamma(i-1,j-1) , \gamma(i-1,j) , \gamma(i,j-1) \}$$




**Ergebnis von Pohle et al. 2005: je Lernaufgabe ist anderer Merkmalssatz nötig!**

- Gegeben eine Menge low level descriptors, klassifiziere nach einem Aspekt
  - Genre
  - Stimmung
  - Tempo
  - Instrument vs. Gesang vs. beides
- Es gibt keine Menge von Merkmalen, die alle Klassifikationsaufgaben lösen hilft.
- Je Lernziel (Aspekt) ist ein anderer Merkmalssatz nötig.
- Tzanetakis' Merkmale sind immer einigermaßen gut.

**Mierswa Diplomarbeit 2004**

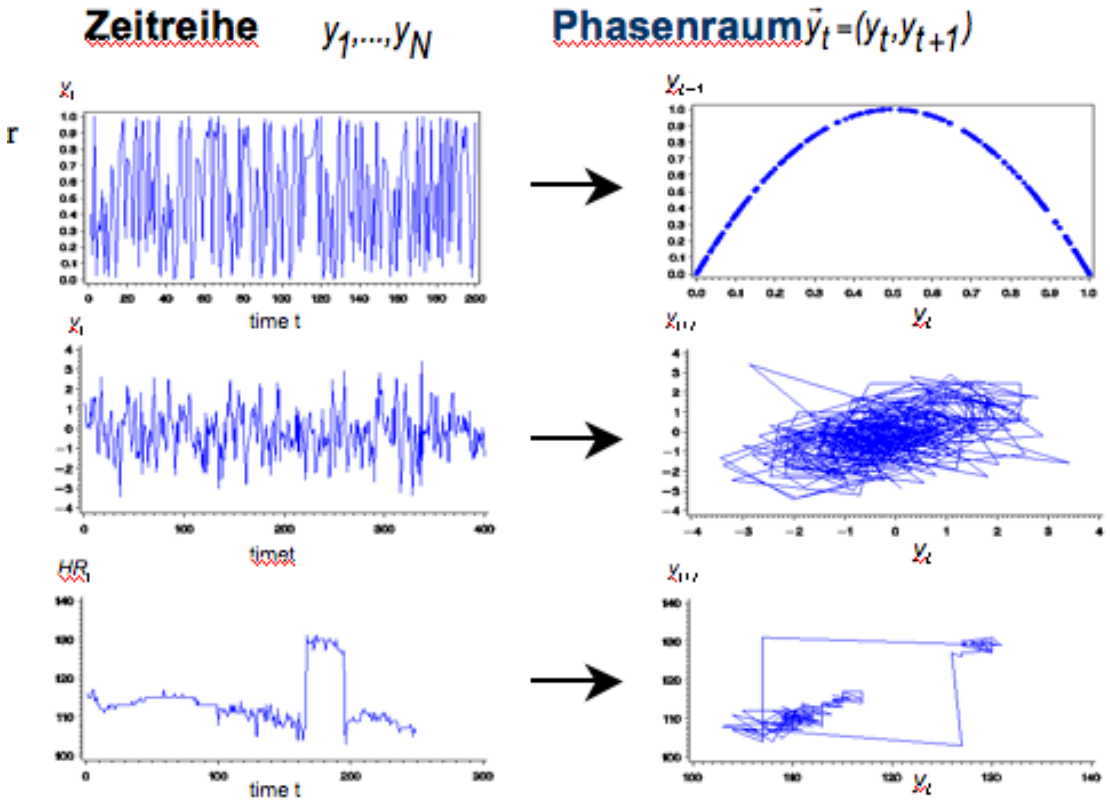
- Jeder Mensch achtet auf Unterschiedliches, um Musik zu beurteilen.
- Dieselbe abstrakte Eigenschaft wird anhand völlig unterschiedlicher Merkmale der physikalischen Ebene zugeschrieben.
- Für persönliche Empfehlungen sind auch persönliche Merkmale nötig.
- Also: lernende Merkmalsextraktion für automatische Klassifikation!

### 9.7.1 Lernende, adaptive Merkmalsextraktion

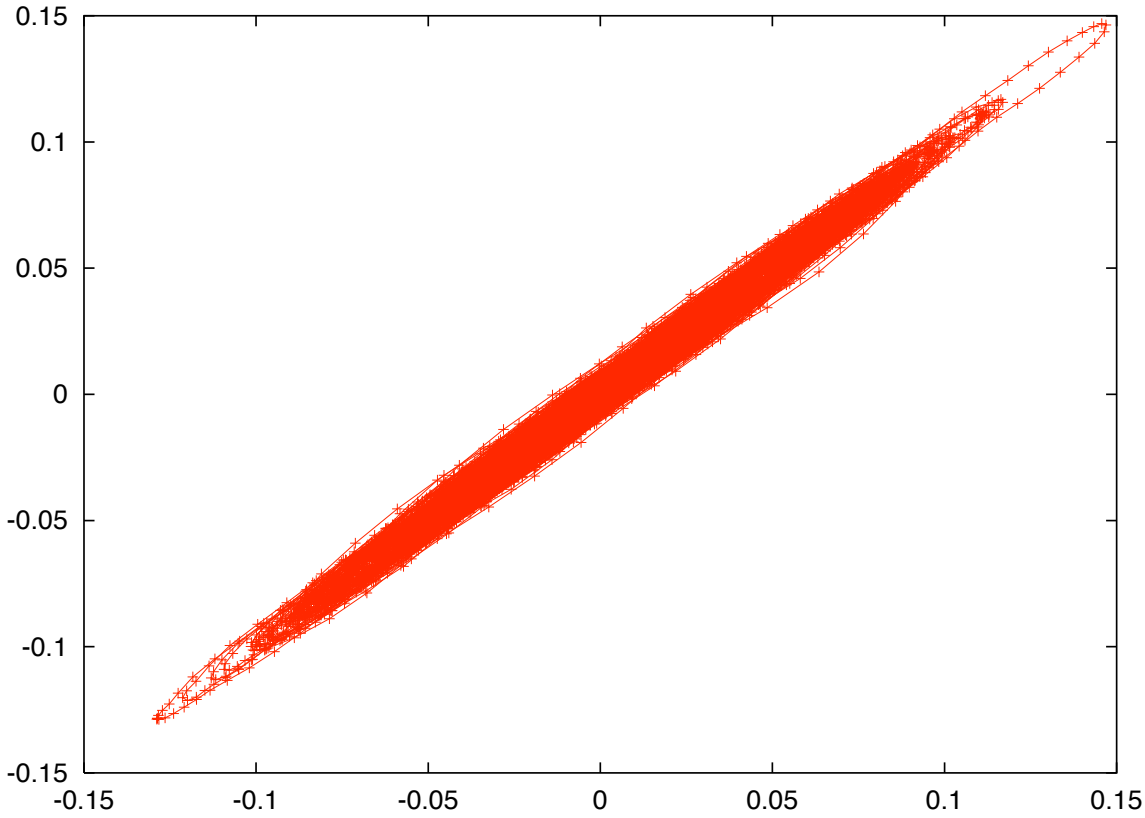
**Merkmalsraum strukturieren**

- Zeitraum (index)
  - Mittlere Lautstärke: $LS(\vec{x}) = \frac{1}{N} \sum_{i=1} \mid x_i \mid$
  - Tempobestimmung durch Autokorrelation verschobener Reihen: für alle Geschwindigkeiten 90 - 170 bpm: Verschiebung der Reihe um einen Takt, berechnen der Differenz zum Original, wenn die Differenz minimal ist, ist das richtige Tempo bestimmt.
- Frequenzraum
  - Für uns ist die diskrete Fourier-Transformation interessant, insbesondere die schnelle (FFT). Dafür muss die Anzahl der Abtastpunkte eine Zweierpotenz sein. Bei FFT geht die Information verloren, wann die Frequenzen auftreten. Also wird ein Zeitfenster über die Reihe verschoben, innerhalb dessen FFT angewandt wird.
- Phasenraum: gegeben die Messwerte $y_1, ...y_N$ für die Zeitpunkte $1, ..., N$, bilde eine neue Reihe mit den Werten $y_{i-1}$ für die Punkte $y_i$.

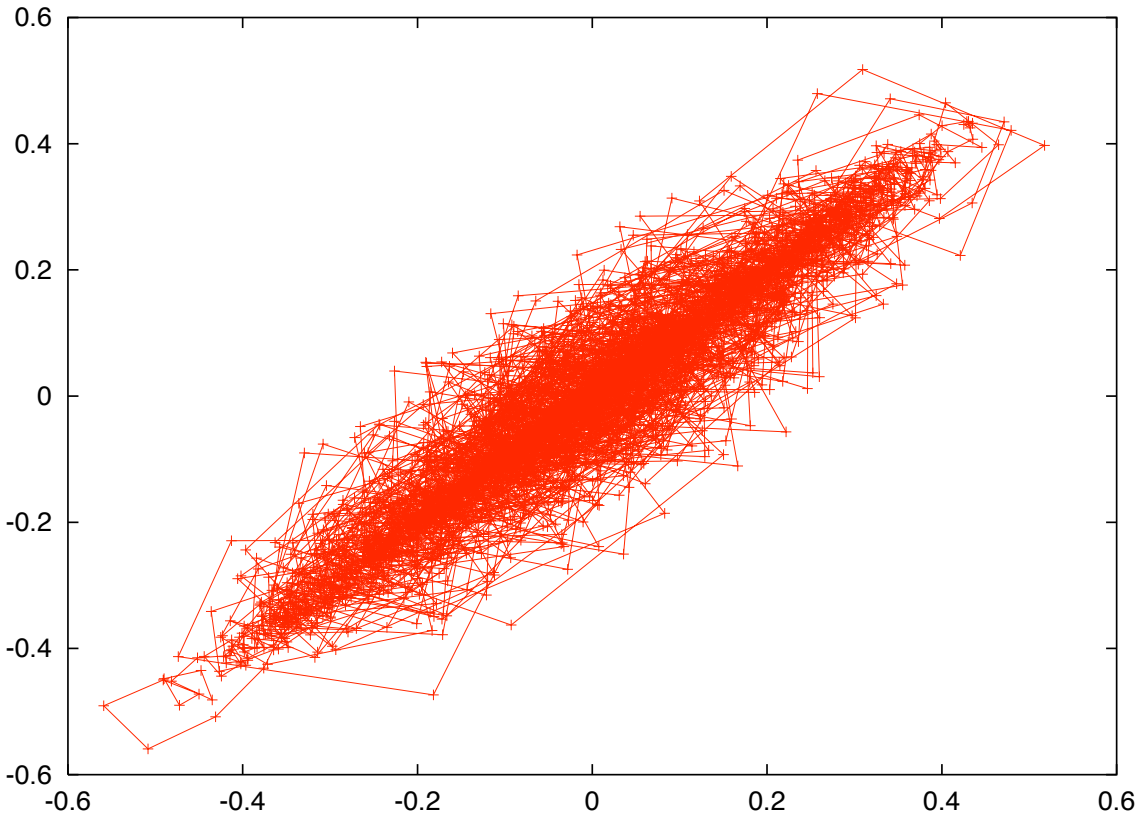
**Phasenraum**



**Phasenraum zur Klassifikation von Genre: Klassik**




**Phasenraum zur Klassifikation von Genre: Pop**




**Merkmalsraum weiter strukturieren**

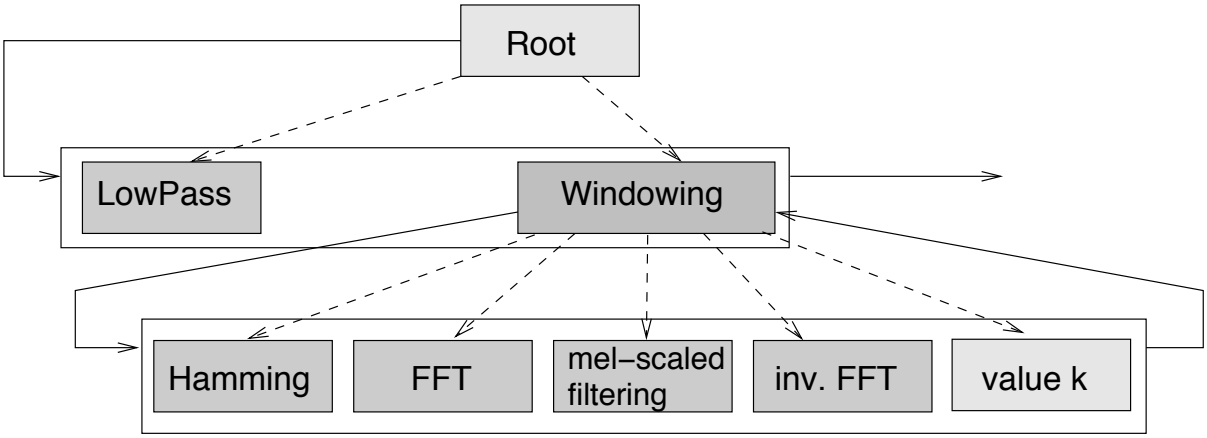
- Wir haben die Transformationen im Zeit-, Frequenz-, Phasenraum gesehen.
- Außerdem gibt es Filter und Annotationen von Segmenten.
- Das generalisierte Fenstern trennt die Funktion, die auf Messwerte in einem Fenster angewandt wird, von dem Fenstern selbst. Beim generalisierten Fenstern können beliebig viele beliebige Funktionen auf Werte in einem Fenster angewandt werden.
- Während bei allen vorigen Funktionen wieder eine Reihe zurückgegeben wird, liefert ein Funktional für eine Reihe nur einen Wert zurück.
- Aus diesen modularen Elementen können nun beliebige Merkmalsextraktionen zusammengestellt werden.

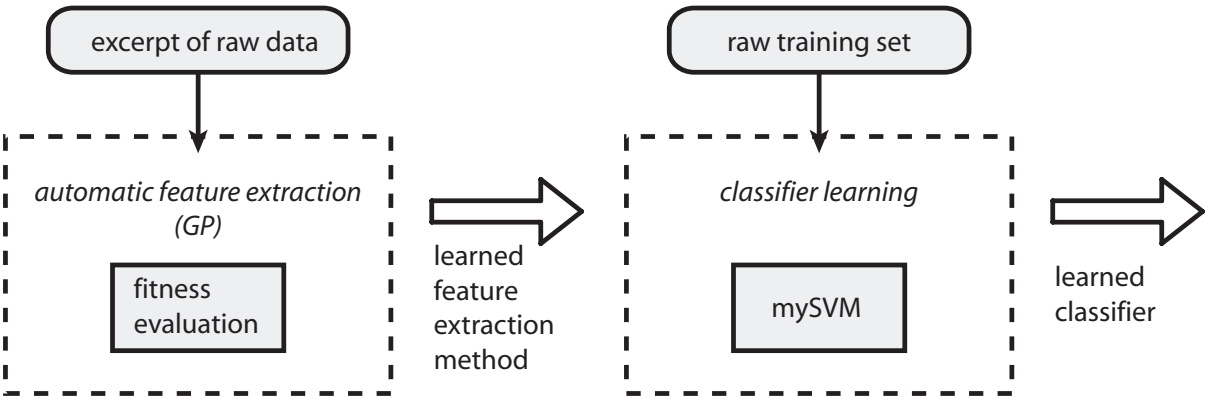
**Methodenbaum zur Extraktion von MFCC**

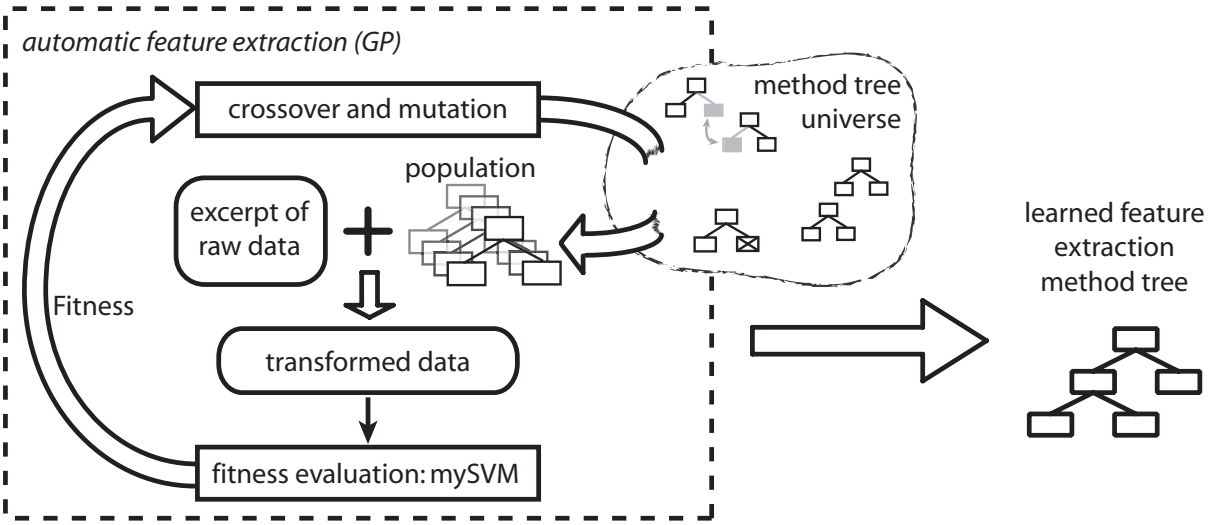
**Überblick über den Lernprozess**

Mierswa, Morik 2005

**Lernen von Methodenbäumen mit genetischer Programmierung**


Root
LowPass
Windowing
Hamming
FFT
mel–scaled filtering
inv. FFT
value k


excerpt of raw data
automatic feature extraction (GP)
fitness evaluation
learned feature extraction method
raw training set
classifier learning
mySVM
learned classifier


automatic feature extraction (GP)
crossover and mutation
method tree universe
population
excerpt of raw data
Fitness
transformed data
fitness evaluation: mySVM
learned feature extraction method tree

| | **Classic/pop** | **Techno/pop** | **Hiphop/pop** |
|---|---|---|---|
| Accuracy | 100% | 93.12% | 82.50% |
| Precision | 100% | 94.80% | 85.27% |
| Recall | 100% | 93.22% | 79.41% |
| Error | 0% | 6.88% | 17.50% |

Tabelle 9.1: Klassifikation (lineare SVM) mit gelernten Merkmalen.

| | **Classic/pop** | **Techno/pop** | **Hiphop/pop** |
|---|---|---|---|
| Accuracy | 96.50% | 64.38% | 72.08% |
| Precision | 94.12% | 60.38% | 70.41% |
| Recall | 95.31% | 64.00% | 67.65% |
| Error | 3.50% | 35.63% | 27.92% |

Tabelle 9.2: Klassifikation mit dem selben Merkmalssatz für alle Aufgaben (lineare SVM).

**Aufgabenspezifisches Lernen der Merkmale verbessert das Ergebnis**

41 Merkmale wurden insgesamt gelernt.

**Klassifikation nach Benutzerpräferenz**

- 50 to 80 Stücke Lieblingsmusik
- Die selbe Anzahl negativer Beispiele.

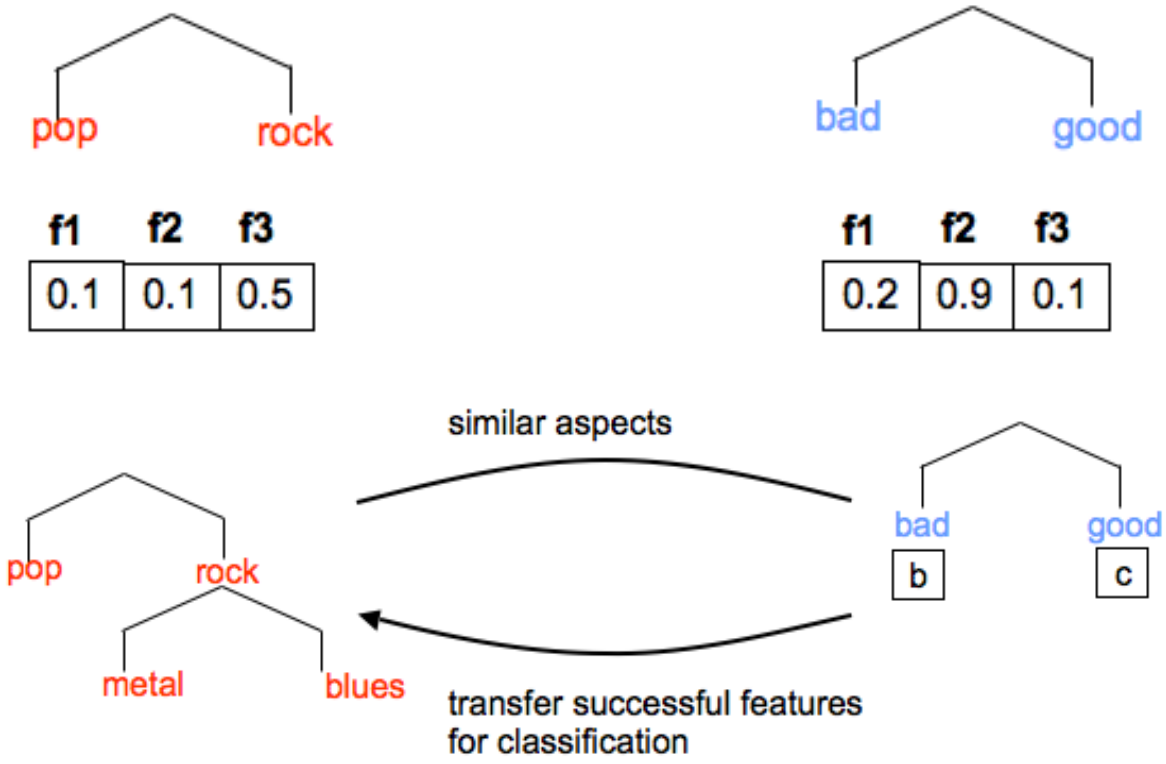
Alles implementiert im Value-Series Plugin von RapidMiner.

Verwendbar für alle Wertereihen!

**Eigenschaften lernender Merkmalsextraktion**

- Sehr gute Lernergebnisse
- Aufwand des Benutzers, die Beispielmengen zusammenzustellen → automatisch (aus Hörverhalten) extrahieren!
- Aufwand der genetischen Programmierung
- Merkmale werden aus einem Musikstück (Sample) extrahiert – funktioniert nicht inkrementell (online).

| | **User$_1$** | **User$_2$** | **User$_3$** | **User$_4$** |
|---|---|---|---|---|
| Accuracy | 95.19% | 92.14% | 90.56% | 84.55% |
| Precision | 92.70% | 98.33% | 90.83% | 85.87% |
| Recall | 99.00% | 84.67% | 93.00% | 83.74% |
| Error | 4.81% | 7.86% | 9.44% | 15.45% |




### 9.7.2 Merkmalsübertragung

**Merkmalsübertragung**

- Wenn das Trainieren der Merkmalsextraktion so lange dauert (1 Woche), sollte für ähnliche Lernaufgaben auch bereits gelernte Merkmalssätze verwendet werden (Mierswa/Wurst 2005, Wurst/Morik 2006).
- Charakterisierung einer Merkmalsmenge durch Gewichtung von Basismerkmalen.
- Feststellen der Eignung von Basismerkmalen für eine Klassifikationsaufgabe.
- Ähnliche Gewichte der Basismerkmale $\rightarrow$ ähnliche Lernaufgaben und Transfer des gesamten Merkmalssatzes.

**Merkmalstransfer im Bild**

**Eignung von Merkmalen für eine Lernaufgabe**

- Ein Merkmal $X_{ik}$ ist **irrelevant** für eine Klassifikationsaufgabe $t_i$, wenn es nicht mit $Y_i$ korreliert ist: $Pr(Y_i \mid X_{ik}) = Pr(Y_i)$. Die Menge irrelevanter Merkmale für $t_i$ ist $IF_i$.
- Zwei Merkmale $X_{ik}$ und $X_{ir}$ heißen **alternativ** bzgl. einer Lernaufgabe $t_i$, $X_{ik} \sim X_{ir}$, gdw. $X_{ir} = a + b \cdot X_{ik}, b > 0$. Die Menge alternativer Merkmale für $t_i$ ist $AF_i$.
- $X_B$ sei eine Menge von Basismerkmalen.

- Die Merkmale sollen nun so gewichtet werden, wie es ihrer Eignung für die Lösung einer Lernaufgabe enspricht $w : X_B \to \mathcal{R}$.

**Bedingungen für Merkmalsgewichtungen, die die Charakterisierung von Lernaufgaben erlauben**

1. $w(X_{ik}) = 0$, wenn $X_{ik} \in X_B$ irrelevant ist. Irrelevante Merkmale sind mit 0 gewichtet.
2. Für $AF_i \subseteq X_B$ gilt: $\forall S \subset AF_i, S \neq \{\} : \sum_{X_i \in S} w(X_k) = \sum_{X_k \in AF_i} w(X_k) = \hat{w}$

   Die Gewichtsumme alternativer Merkmale ist unabhängig von der Anzahl alternativer Merkmale.
3. $X_{ik} \sim X_{ir} \Rightarrow w(X_{ik}) = w(X_{ir})$ Alternative Merkmale sind gleich gewichtet.
4. $\forall X_{ik} \in AF_i : X_{ir} \in IF_i \vee \exists X_{ir} \in X_B : X_{ik}X_{ik} \sim X_{ir} \Rightarrow \forall X_{ir} \in X_B : \nexists X_{ik} \in AF_i : X_{ir} \sim X_{ik} \wedge w'(X_{ir}) = w(X_{ik})$ mit $w' : X_B \cup AF \to \mathcal{R}$. Eine Menge alternativer Merkmale ist nicht stärker gewichtet als ein einzelnes Merkmal.

**Die Bedingungen gelten nicht immer!**

- Alle Methoden der Merkmalsauswahl, die Merkmale binär gewichten, verletzen Bedingung 2 oder 3, sobald ein alternatives Merkmal hinzugefügt wird. $X'_B = X_B \cup \{X_{ir}\}, X_{ir} \sim X_{ik}, X_{ik} \in X_B \Rightarrow w'(X_{ir}) = w'(X_{ik}) = w(X_{il}) = 1$ weil ein ausgewähltes Merkmal in $X_B$ Gewicht 1 hat; Verletzung 2. Bedingung: die Summe wäre 2! oder $w'(X_{ir}) \neq w(X_{ik})$ Verletzung 3. Bedingung (Alternativen sind gleichgewichtet).
- Jede Methode, die die Merkmale unabhängig voneinander gewichtet, verletzt Bedingung 2. Bei $X'_B = X_B \cup \{X_{ir}\}$ bleiben alle Gewichte für Merkmale in $X_B$ gleich. Wenn $X_{ir} \sim X_{ik}, X_{ik} \in X_B$ verändert sich die Summe, so dass 2. Bedingung verletzt ist.

**Die lineare SVM erfüllt alle Bedingungen**

Die Merkmalsgewichtung durch die lineare SVM, $\vec{\beta}$, erfüllt alle Bedingungen.

- Bedingung 1: Die Euklidsche Länge von $\vec{\beta}$ soll minimiert werden, also werden möglichst Merkmale mit 0 gewichtet, wenn dadurch nicht der Fehler steigt. Also werden irrelevante Merkmale mit 0 gewichtet.
- Bedingung 2: Fügen wir einfach das selbe Merkmal mehrfach hinzu, so ergibt sich $(\beta_{i1}+...+\beta_{im})\vec{x}$ in $\vec{\beta}\vec{x} + \beta_0$. Die optimale Hyperebene ändert sich nicht und die Summe der Gewichte bei allen anderen Merkmalen bleibt unverändert.
- Bedingung 3: Die Summe der alternativen Merkmale verteilt sich gleichmäßig auf die Alternativen.
- Bedingung 4: Folglich ist die Menge alternativer Merkmale nicht stärker gewichtet als ein einzelnes Merkmal.

**Geeignete Abstandsmaße für die Gewichtung der Basismerkmale als Ähnlichkeit von Lernaufgaben**

Das Abstandsmaß $d : T \times T \to \mathcal{R}^+$ soll erfüllen:

1. $d(\vec{t_1}, \vec{t_2}) = 0 \Leftrightarrow \vec{t_1} = \vec{t_2}$
2. $d(\vec{t_1}, \vec{t_2}) = d(\vec{t_2}, \vec{t_1})$
3. $d(\vec{t_1}, \vec{t_2}) = d(\vec{t_1}', \vec{t_2}'), \vec{t_1}' = \vec{t_1}, \vec{t_1}' \in X_B^2 \cup IF_1^2$ und $\vec{t_2}' = \vec{t_2}, \vec{t_2}' \in X_B^2 \cup IF_2^2$ gleiche Gewichtsvektoren behalten im erweiterten Bereich gleichen Abstand.
4. $d(\vec{t_1}, \vec{t_2}) = d(\vec{t_1}', \vec{t_2}'), \vec{t_1}' = \vec{t_1}, \vec{t_1}' \in X_B^2 \cup AF_1^2$ und $\vec{t_2}' = \vec{t_2}, \vec{t_2}' \in X_B^2 \cup AF_2^2$ gleiche Gewichtsvektoren behalten im erweiterten Bereich gleichen Abstand.

**Die Bedingungen gelten nicht immer!**

Bei Euklidschem Abstand wird Bedingung 5 nicht eingehalten, d.h. das Hinzufügen alternativer Merkmale verändert den Abstand.

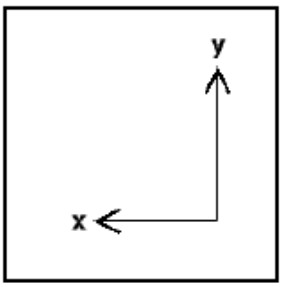
- Das alternative Merkmal $X_r$ wird $X_B$ hinzugefügt und ist alternativ zu $X_k \in X_B$. Wenn die Bedingungen an die Merkmalsgewichtung eingehalten sind, gilt: $w'(X_{sk}) = w'(X_{sr}) = \frac{w(X_{sk})}{2} = \frac{w(X_{sr})}{2}$ für $s = 1, 2$
- Seien alle anderen Merkmalsabstände $S$, dann ist

$$\begin{aligned} d(\vec{t_1}', \vec{t_2}') &= \sqrt{S + 2(w'(X_{ik}) - w'(X_{jk}))^2} \\ &= \sqrt{S + 2(\tfrac{w(X_{ik})}{2} - \tfrac{w(X_{jk})}{2})^2} \\ &= \sqrt{S + \tfrac{1}{2}(w(X_{ik}) - w(X_{jk}))^2} \\ &\neq \sqrt{S + (w(X_{ik}) - w(X_{jk}))^2} \\ &= d(\vec{t_1}, \vec{t_2}) \end{aligned}$$

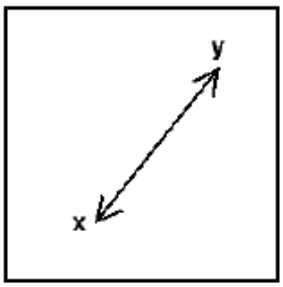
**Manhattan Abstand hält alle Bedingungen ein**

- Bedingungen 1 - 3 sind die einer Metrik.
- Bedingung 4: Wir fügen ein für beide Lernaufgaben $\vec{t_1}, \vec{t_2}$ irrelevantes Merkmal $X_{k+1}$ hinzu. Wenn die Bedingung 4 an die Gewichtung eingehalten ist, gilt: $\mid w'(X_{1,k+1}) - w'(X_{2,k+1}) \mid = 0$. Also:

$$\begin{aligned} d(\vec{t_1}', \vec{t_2}') &= \sum_{r=1}^{k} \mid w'(X_{1,r}) - w'(X_{2,r}) \mid + 0 \\ &= d(\vec{t_1}, \vec{t_2}) \end{aligned}$$




Manhattan




Euclidean

**Manhattan Fortsetzung**

- Bedingung 5: Das alternative Merkmal $X_{k+1}$ wird $X_B$ hinzugefügt und ist alternativ zu $X_k \in X_B$. Wenn die Bedingungen an die Merkmalsgewichtung eingehalten sind, gilt: $w'(X_{s,k+1}) = w'(X_{s,k}) = \frac{w(X_{s,k+1})}{2} = \frac{w(X_{s,k})}{2}$ für $s = 1, 2$

$$\begin{aligned} d(\vec{t_1}', \vec{t_2}') &= (\sum_{r=1}^{k-1} \mid w'(X_{1,r}) - w'(X_{2,r}) \mid) + \\ &\quad 2(\mid w'(X_{1,k+1}) - w'(X_{2,k+1}) \mid) \\ &= (\sum_{r=1}^{k-1} \mid w(X_{1,r}) - w(X_{2,r}) \mid) + \\ &\quad \mid w(X_{1,k}) - w(X_{2,k}) \mid \\ &= d(\vec{t_1}, \vec{t_2}) \end{aligned}$$

**Unterschied der Abstandsmaße Manhattan und Euklid**

$d(x, y)$

**Anwendung der Merkmalsübertragung**

- Gegeben die 39 Taxonomien zur Musikorganisation. Je Knoten sei die Lernaufgabe, in die Unterknoten zu klassifizieren.
- Wir optimieren Musikmerkmale für jede Lernaufgabe.
- Als Basismerkmale werden 10 gewählt, die für die meisten Lernaufgaben erzeugt wurden.
- Anwendung der linearen SVM auf jede Lernaufgabe liefert $\vec{\beta}$ und damit auch eine Gewichtung der Basismerkmale. $O(\mid X_B \mid\mid T \mid N^3)$
- Gemäß der gewichteten Basismerkmale wird die Ähnlichkeit der Lernaufgaben festgestellt. $O(\mid X_B \mid\mid T \mid^2)$
- Bei ähnlichen Lernaufgaben wird der komplette Merkmalssatz transferiert.

**Ergebnis des Merkmalsübertragung**

| | Accuracy | Time | Optimization cycles |
|---|---|---|---|
| base features | 0.79 | - | - |
| optimal features | 0.92 | 42s | 3970 |
| cbfc (k = 1) | 0.85 | 3s | 257 |
| cbfc (k = 3) | 0.88 | 5s | 389 |
| cbfc (k = 9) | 0.89 | 8s | 678 |

Tabelle 9.3: Durchschnittliche accuracy und Gesamtaufwand auf einem Testset von 11 Taxonomien für Lernen mit Basismerkmalen, optimierten Merkmalssätzen und Merkmalstransfer von den $k$ ähnlichsten Lernaufgaben (cbfc).

**Was wissen Sie jetzt?**

- Merkmale können aus Basisfunktionen und -transformationen per Genetischer Programmierung hergestellt werden, wobei die Qualität des Lernergebnisses optimiert wird.
- Merkmale werden von Entscheidungsbaumlernern und der SVM sehr unerschiedlich behandelt. Wichtigster Unterschied ist die Behandlung irrelevanter oder alternativer Merkmale.
- Nur die SVM-Merkmalsgewichtung im Zusammenhang mit der Manhattan-Distanz ermöglicht, anhand der Gewichtung von Basismerkmalen die Ähnlichkeit von Lernaufgaben festzustellen.
- Antrainierte Merkmalssätze können auf ähnliche Lernaufgaben übertragen werden und liefern dann mit viel weniger Aufwand fast gleich gute Ergebnisse.

# Kapitel 10

# Subgruppenentdeckung

## 10.1 Lernaufgabe Subgruppenentdeckung

**Lernaufgabe Subgruppenentdeckung**

- Gegeben
  - $X$ der Raum möglicher Beobachtungen mit einer Wahrscheinlichkeitsverteilung $D$,
  - $S \subseteq X$ eine gemäß $D$ gezogene Stichprobe,
  - $L_H$ der Raum möglicherweise gültiger Regeln, wobei jeder Regel $h \in L_H$ eine Extension zugeordnet ist: $ext(h) \subseteq X$ und
  - eine Qualitätsfunktion
$$q : L_H \rightarrow \mathcal{R}$$
- finde
  - eine Menge $H \subseteq L_H, \mid H \mid = k$
  - und es gibt keine $h' \in L_H \backslash H, h \in H$, für die gilt $q(h') \geq q(h)$
- Rule Discovery Tool RDT Jörg-Uwe Kietz, Stefan Wrobel (1992) Controlling the complexity of learning in logic through syntactic and task-oriented models. In: S. Muggleton (ed) Inductive Logic Programming.

**Beispiel der Subgruppenentdeckung**

Es werden Gruppen beschrieben, die sich abweichend von der Gesamtpopulation verhalten.

Es geht nicht notwendigerweise um Vorhersage, sondern um Beschreibung! Trotzdem ist meist eine Hypothese eine Abbildung $h : X \rightarrow Y$.

- Unter den alleinstehenden jungen Männern in ländlichen Regionen ist der Anteil an Lebensversicherungskinden signifikant niedriger als im gesamten Kundenbestand.
- Verheiratete Männer mit Pkws der Luxusklasse machen nur 2 Prozent der Kunden aus, erzeugen aber 14 Prozent der Lebensversicherungsabschlusssumme.

**Ansätze zur Subgruppenentdeckung**

- Aufzählend: vollständige Suche im strukturierten Raum $L_H$ mit Pruning – Garantie, dass die $k$ besten Regeln gefunden werden. Explora (Klösgen 1996), Midos (Wrobel 1997)
- Heuristisch: ein Entscheidungsbaumlerner wird so verändert, dass seine Qualitätsfunktion die der Subgruppenentdeckung wird und Beispiele ein veränderliches Gewicht erhalten – keinerlei Garantie. CN2-SD (Lavrac et al. 2004)
- *Probabilistisch*: Stichproben-bezogene Fehler werden während das Scans der Daten abgeschätzt – probabilistische Garantie, dass die $k$ besten Regeln gefunden werden. (Scheffer, Wrobel 2002)

**Modellselektion**

- Die Menge $H$ der gewählten Hypothesen kann auch als Modell betrachtet werden.
- Die Subgruppenentdeckung ist dann ein Problem der Modellselektion.
- Dabei geht es immer um Gütekriterien.
- Wir hatten ja schon:
  - Accuracy
  - Precision
  - Recall
  - Mittlerer quadratischer Fehler, quadratische Fehlersumme, erwarteter quadratischer Fehler, 0-1-Verlust
  - Entropie

### 10.1.1 Qualitätsfunktionen

**Erinnerung**

Bewertung anhand der Zielvariablen

| | predicted $Y$ | predicted $\neg Y$ | |
|---|---|---|---|
| true $Y$ | TP | FN | Recall: $\frac{TP}{TP+FN}$ |
| true $\neg Y$ | FP | TN | Specificity: $\frac{TN}{FP+TN}$ |
| | Precision: $\frac{TP}{TP+FP}$ | | |
| | Accuracy: $\frac{TP+TN}{TP+TN+FP+FN}$ | | |

**RDT: Bausteine von Regelbewertungen**

Für eine Regel $h : A \rightarrow Y$ zählt man (z.B. per SQL-Anfrage):

- $pos(h)$: Prämisse und Konklusion kommen gemeinsam vor
- $neg(h)$: Prämisse und Negation der Konklusion kommen gemeinsam vor
- $concl(h)$: Konklusion kommt vor
- $pred(h)$: Prämisse kommt vor, aber Konklusion (noch) nicht

- $unc(h)$: Instanzen der Konklusion, die von Prämisse nicht abgedeckt sind
- Regelbewertung z.B. precision: $\frac{pos(h)}{pos(h)+neg(h)}$ recall: $\frac{pos(h)}{concl(h)}$

**Qualitätskriterium Lift**

Probablistische Regelbewertung:

- Für eine Regel $h = A \rightarrow Y$, wobei $A$ eine Menge von Literalen ist und $Y = \{0, 1\}$ ist

$$Lift(A \rightarrow Y) = \frac{Pr[A, Y]}{Pr[Y]} = \frac{precision(A \rightarrow Y)}{Pr[Y]} \tag{10.1}$$

- Bei $Lift(A \rightarrow Y) = 1$ sind $A$ und $Y$ unabhängig.
- Bei $Lift(A \rightarrow Y) > 1$ steigt die bedingte Wahrscheinlichkeit für $Y$ gegeben $A$.
- Bei $Lift(A \rightarrow Y) < 1$ sinkt die bedingte Wahrscheinlichkeit für $Y$ gegeben $A$.
- Lift normalisiert die precision gegenüber einer verzerrten Verteilung der Klassen!

**Coverage und Bias von Regeln**

- Die Wahrscheinlichkeit, dass der Antezedens $A$ der Regel auf ein Beispiel zutrifft bei einer Verteilung $D$ der Beispiele ist:

$$Cov(A \rightarrow Y) = Pr[A]$$

- Die Differenz zwischen der bedingten Wahrscheinlichkeit von $Y$ gegeben $A$ und der a priori Wahrscheinlichkeit für $Y$ ist der Bias:

$$Bias(A \rightarrow Y) = Pr[Y \mid A] - Pr[Y] = Pr[Y] \cdot (Lift(A \rightarrow Y) - 1)$$

**Weighted relative accuracy WRAcc**

- Man kann Bias und Coverage für eine Anwendung mit einem Parameter $\alpha$ geeignet gewichten.
  - Vielleicht will man auf jeden Fall alles abdecken, weil man alle Beispiele irgendwie behandeln muss. Dann gewichtet man Coverage hoch.
  - Vielleicht findet man nur Abweichungen von der a priori Wahrscheinlichkeit interessant. Dann gewichtet man Bias hoch.
  - Bei gleichgewichteten Coverage und Bias $\alpha = 0,5$ erhält man das selbe Ergebnis wie beim binominalen Test, der die Nullhypothese ($A$ hat keinen Einfluss) testet.
- Für eine Regel $h$ und eine Gewichtung $\alpha \in [0, 1]$ ist

$$WRAcc(\alpha, h) = Cov(h) \cdot Bias(h)$$

Tabelle 10.1: $y = 1$ für Spam, Fehler insgesamt 9%

| | Predicted | |
|---|---|---|
| True | email | spam |
| email | 57,3 | 4,0 |
| spam | 5,3 | 33,4 |

**Wofür die Maße?**

- Jetzt wissen wir, wie wir Regeln auswählen können.
- Wir wollen aber auch noch wissen, wie gut das Modell, also die gesamten Regeln, ist.
- Dann können wir die Regelmenge auswählen, die am besten ist.

**Sensitivität und Spezifität – ROC**

- Sensitivität (Recall): Wahrscheinlichkeit, dass ein positives Beispiel auch als positiv erkannt wird. (TP: true positives)
- Spezifizität: Wahrscheinlichkeit, dass ein negatives Beispiel auch als negativ erkannt wird. (TN: true negatives)
- Die *Receiver Operator Characteristic (ROC)* Kurve setzt Sensitivität und Spezifizität in Beziehung für verschiedene Parameter. Je nach Benutzerinteresse (TP wichtiger? TF wichtiger? Beides?) wird das Modell gewählt.

**Beispiel**

Sensitivität (Recall) $\frac{TP}{TP+FN}$:

$$100 \cdot \frac{33,4}{33,4+5,3} = 86,3\%$$

Spezifizität $\frac{TN}{FP+TN}$:

$$100 \cdot \frac{57,3}{57,3+4,0} = 93,4\%$$

**ROC im Bild**

Ein Parameter wurde zwischen 0,1 und 10 variiert.

**Area Under the Curve**

**AUC**

Für $h : X \to Y, Y \in \{0, 1\}$ und $D : X \times Y \to \mathcal{R}^+$ ist die *Area Under the ROC Curve* (AUC) die Wahrscheinlichkeit

$$AUC(h) = Pr[h(\vec{x}) \geq_q h(\vec{x}') \mid y = 1, y' = 0]$$

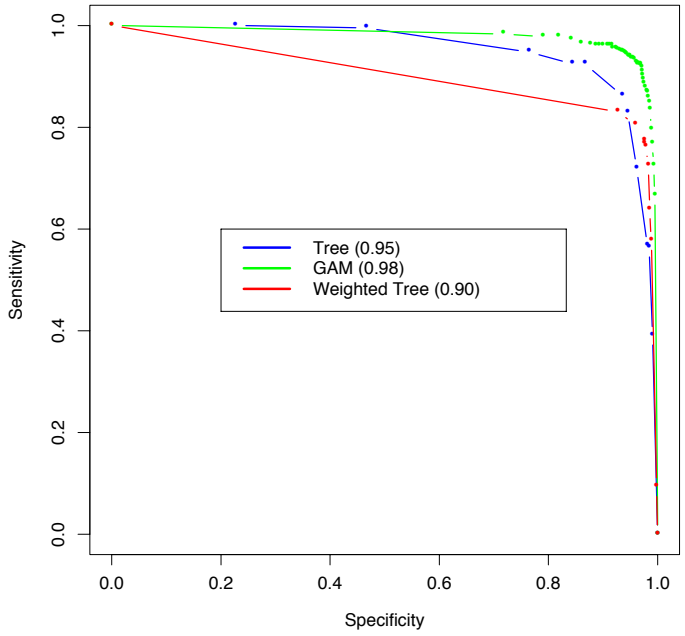



Figure 9.6: *ROC curves for the classification rules fit to the* `spam` *data. Curves that are closer to the north-east corner represent better classifiers. In this case the GAM classifier dominates the trees. The weighted tree achieves better sensitivity for higher specificity than the unweighted tree. The numbers in the legend represent the area under the curve.*

dass ein zufällig gezogenes positives Beispiel höher bewertet wird gemäß einer Qualitätsfunktion $q$ als ein zufällig gezogenes negatives Beispiel.

AUC ist invariant gegenüber monotonen Transformationen von $h$.

**Idee eines Algorithmus', der AUC berechnet**

- Schätze für jedes Beispiel in $S$ die Wahrscheinlichkeit, ein positives zu sein.
- Ordne die Beispiele nach dieser Wahrscheinlichkeit (ranking).
- Bewerte ein Lernergebnis nach der Anzahl $\Lambda(h, S)$ der notwendigen Vertauschungen der Reihenfolge (des rankings).
- Sei $S_+$ die Menge der positiven Beispiele, $S_-$ die Menge der negativen, dann ist

$$AUC(h, S) = \frac{\Lambda(h, S)}{\mid S_+ \mid \cdot \mid S_- \mid}$$

## 10.2 Sampling

**Abhängigkeit des Lernergebnisses von $S$**

- Eigentlich wollen wir ja ein optimales (oder wenigstens angenähert optimales) Lernergebnis auch für noch nicht gesehene Beispiele haben.
- Die ROC Kurve bezieht sich wie auch AUC nur auf die Stichprobe $S$.
- Meist sind die Datenmengen so groß, dass wir nur eine Stichprobe behandeln können.
- Wir wollen jetzt eine Stichprobe ziehen, die ungefähr so verteilt ist wie die Gesamtmenge.
- Leider haben wir keine Ahnung, was die wahre Verteilung ist!

**i.i.d. erhaltende Stichprobe**

- Die Daten insgesamt, $\mathbf{X}$, und die Stichprobe $S$ sollen i.i.d. verteilt sein.
- Folgen von Zufallsvariablen, die sowohl unabhängig als auch identisch verteilt sind werden üblicherweise mit i.i.d. (für independent and identically distributed) bezeichnet.
  - Beispiel dreimaliges Würfeln:
    - $X_1$ 1. Wurf, $X_2$ 2. Wurf, $X_3$ 3. Wurf sind i.i.d. verteilt.
    - $X_4 = X_1 + X_2$ und $X_5 = X_2 + X_3$ sind zwar identisch verteilt, aber nicht unabhängig.
    - $X_4$ und $X_3$ sind unabhängig, aber nicht identisch verteilt.
- Wenn die Daten in der Datenbank in zufälliger Reihenfolge gespeichert sind, ergibt das Ziehen der $m$ ersten Daten eine i.i.d. erhaltende Stichprobe.

### Ziehen der Stichprobe mit/ohne Zurücklegen

- Zufällig ein Beispiel ziehen ist Ziehen mit Zurücklegen. Dabei kann es Doppelte geben und damit eine Verzerrug (Bias). Die Wahrscheinlichkeit für Doppelte beim Ziehen von $m$ Beispielen aus $N$ ist:
$$p_m = \frac{N!}{(N-m)! \cdot N^m}$$
Also sinkt die Wahrscheinlichkeit, keine Doppelten zu haben, $1 - p_m$, exponentiell mit Steigen von $m$.
- Zufällig ein Beispiel ziehen und es nicht Zurücklegen verfälscht nicht: jedes Beispiel hat die selbe Wahrscheinlichkeit, gezogen zu werden $m/N$. Leider ist dies aufwändig: man muss prüfen, ob ein Beispiel der Datenbank schon gezogen wurde, logarithmische Laufzeit.

### Konfidenz

- Wir möchten gern wissen, bei wie vielen Beispielen wir wie sicher sein können, uns nicht zu verschätzen.
- Dazu nehmen wir einen Konfidenzwert $\delta$ und Schranken für die Wahrscheinlichkeit.
- Dann können wir nach und nach imer größere Stichproben ziehen, bis wir uns sicher genug sind. Und dann aufhören!

### Chernoff-Schranke

- Sei $p$ die Wahrscheinlichkeit, dass ein Beispiel gezogen wird, das von einer Regel $h$ korrekt klassifiziert wird.
- Bei i.i.d. Stichproben ist $p$ konstant für alle Regeln.
- Die Zufallsvariable $X_i$, mit $i = 1, ..., m$ sei 1 für die korrekte Klassifikation, 0 sonst.
- Der Erwartungswert für $\bar{Y} = 1/m \sum X_i$ ist gerade $p$: $E(\bar{X}) = p$
- Die Standardabweichung ist $\sigma(\bar{Y}) = \sqrt{\frac{p(1-p)}{m}}$
- Die *Chernoff-Schranke* sagt für beliebigen Parameter $\lambda$:
$$Pr[\bar{Y} \geq (1+\lambda)p] \leq exp(-\lambda^2 mp/3) \tag{10.2}$$
$$Pr[\bar{Y} \leq (1-\lambda)p] \leq exp(-\lambda^2 mp/2) \tag{10.3}$$

### Chernoff-Schranke zur Abschätzung der geeigneten Stichprobengröße – Beispiel

- Wie wahrscheinlich ist es, dass Regeln mit der wahren Accuracy $Acc = p = 75\%$ bei einer Stichprobe der Größe $m$ nicht besser als reiner Zufall abschneiden?
- Sei $\bar{Y} = \widehat{Acc}$ der Anteil korrekter Klassifikationen, sei der reine Zufall 50%. $\lambda = 1/3$, weil $(1 - 1/3) \cdot Acc = 50\%$.

- Wegen Gleichung (10.2) ergibt sich:

$$Pr[\widehat{Acc} \leq (1 - 1/3) \cdot Acc] \leq exp(-(1/3)^2 m \cdot Acc/2)$$
$$\Leftrightarrow Pr[\widehat{Acc} \leq 1/2] \leq exp(-1/9m3/8) = exp(-\frac{m}{24})$$

- Risiko $\leq \delta = 5\%$, dass bei $m \geq 72$ Beispielen ein 75% gutes $h$ die Hälfte falsch klassifiziert:

$$exp(-\frac{m}{24}) \leq \delta \Leftrightarrow -\frac{m}{24} \leq ln\ \delta = -ln\frac{1}{\delta} \Leftrightarrow m \geq 24ln\frac{1}{\delta} = 24ln20$$

**Hoeffding-Schranke**

- Die *Hoeffding-Schranke* ist unabhängig von $Acc$ definiert.

$$\begin{aligned} Pr[\bar{Y} - p \geq \epsilon] &\leq exp(-2\epsilon^2 m) \\ Pr[\bar{Y} - p \leq -\epsilon] &\leq exp(-2\epsilon^2 m) \\ Pr[\mid \bar{Y} - -p \mid \geq \epsilon] &\leq 2exp(-2\epsilon^2 m) \end{aligned} \quad (10.4)$$

- Die wahre $Acc$ soll um nicht mehr als 10% über- oder unterschätzt werden. Wegen Gleichung (10.4) ergibt sich:

$$Pr[\mid \widehat{Acc} - Acc \mid \geq 0,1] \leq 2exp(-2 \cdot (0,1)^2 m) \leq 2exp(0,02m)$$

- Risiko $\leq \delta = 5\%$ dafür bei $m \sim 184$ Beispielen:

$$2exp(-0,02m) \leq 0,05 \Leftrightarrow -0,02m \leq ln\frac{1}{40}$$
$$\Leftrightarrow 0,02m \geq ln\ 40 \Leftrightarrow m \geq 50ln\ 40 \sim 184$$

**Stichprobengröße für Subgruppenentdeckung**

- Sei Konfidenzparameter $\delta \in [0,1]$ und höchster geduldeter Fehler $\epsilon \in \mathcal{R}^+$, es sollen die $k$ besten Regeln $H \in L_H$ gemäß einer Qualitätsfunktion $q$ so gelernt werden, dass mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ eine i.i.d. Stichprobe $\mid S \mid = m$ die wahre Qualität $\hat{q}$ höchstens um $E(m, \delta)$ verfälscht.

- In (Scheffer, Wrobel 2002) wird für die verschiedenen Qualitätskriterien aufgeführt, was $E(m, \delta)$ ist.

- Für $Acc$ kann man die worst case Größenordnung der Stichprobe durch die Menge betrachteter Regeln $L_H$ angeben:

$$m = O(\frac{1}{\epsilon^2} log \frac{\mid L_H \mid}{\delta})$$

**Generic Sequential Sampling (Scheffer, Wrobel 2002)**

- Durchgehen der Beispiele (scan) bis höchstens $m = O(\frac{1}{\epsilon^2} log \frac{|L_H|}{\delta})$ betrachtet wurden;
  1. $Cov$ für positive und negative Beispiele bestimmen;
  2. Anordnen der Regeln nach dem Qualitätskriterium (ranking);
  3. Alle Regeln aussortieren aus dem Lernergebnis $H$, wenn sie häufiger als $\delta(2m \mid L_H \mid)$ falsch waren; die Wahrscheinlichkeit, eine gute Regel auszusortieren, ist dann höchstens $\delta/2$.
  4. Wenn $\mid H \mid \leq k$, wird $H$ ausgegeben und die Regeln sind mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ bis auf eine Abweichung von höchstens $\epsilon$ optimal.

**Stratifizierte Stichproben**

**Stratifizierte Dichtefunktion**

Für $D : X \times Y \rightarrow \mathcal{R}^+$ ist die stratifizierte Dichtefunktion $D'$ definiert als

$$D'(x, y) = \frac{D(x, y)}{\mid Y \mid \cdot Pr[y = y']}$$

und falls wir klassifizieren mit $f : X \rightarrow Y$ als

$$D'(x, y) = \frac{D(x)}{\mid Y \mid \cdot Pr[f(x)]}$$

.

Es wird also die gegebene Verteilung $D$ so geändert, dass die Verteilung der Klassen in $D'$ gleich ist.

**Ergebnis von Scholz 2005**

- Wenn stratifizierte Stichproben gezogen, d.h. die Verteilung ensprechend geändert wird, entspricht die Subgruppenentdeckung mit der Qualitätsfunktion $WRAcc$ genau einer Klassifikation mit der Gütefunktion $Acc$.
- Man kann also in Ruhe die Lernalgorithmen für Klassifikation verwenden und braucht keine neuen zu erfinden.
- Allerdings muss man eine Stratifizierung, also Veränderung der Verteilung algorithmisch formulieren.
- Idee: Das tut man beim Ziehen von Stichproben.
- Folge: das Lernen auch aus großen Datenmengen geht schnell!

## 10.3 Knowledge Based Sampling

**Knowledge-Based Sampling for Subgroup Discovery**

- Wir wollen Vorwissen berücksichtigen, insbesondere nicht redundante Regelmengen $H$ lernen. Dabei ist die Redundanz der Extension wichtig, nicht, dass sie durch verschiedene Merkmale ausgedrückt werden.
- Auch bereits gelernte Regeln $h \in H$ sind Vorwissen.
- Wir wollen wenig Beispiele bearbeiten müssen.
- Wir wollen vorhandene Algorithmen nutzen.
- Wir wollen diejenigen Subgruppen zurückliefern, die von der Allgemeinheit abweichen.
- Meist interessiert den Anwender die Extension einer solchen abweichenden Gruppe.

Martin Scholz *Scalable and Accurate Knowledge Discovery in Real-World Databases*, Dissertation am LS8, TU Dortmund, 2006

**Ansatz: die Verteilung verändern**

Die neue Verteilung $D'$ soll nichts Wesentliches verändern:

$$Pr_{D'}[x \mid A, Y] = Pr_D[x \mid A, Y] \tag{10.5}$$
$$Pr_{D'}[x \mid A, \neg Y] = Pr_D[x \mid A, \neg Y] \tag{10.6}$$
$$Pr_{D'}[x \mid \neg A, Y] = [x \mid \neg A, Y] \tag{10.7}$$
$$Pr_{D'}[x \mid \neg A, \neg Y] = [x \mid \neg A, \neg Y] \tag{10.8}$$

Die Beschränkungen (10.5 – 10.8) bestimmen die neue Verteilung $D' : X \to \mathcal{R}^+$ eindeutig:

$$Pr_{D'}(x) = Pr_D(x) \cdot (Lift_D(h, x))^{-1} \tag{10.9}$$

$Lift(h, x)$

Der Lift eines Beispiels $x \in X$ ist für eine Regel $A \to Y$:

$$Lift(A \to Y, x) = \begin{cases} Lift(A \to Y), falls & x \in ext(A) \cap ext(Y) \\ Lift(A \to \neg Y), falls & x \in ext(A) \cap ext(\neg Y) \\ Lift(\neg A \to Y), falls & x \in ext(\neg A) \cap ext(Y) \\ Lift(\neg A \to \neg Y), falls & x \in ext(\neg A) \cap ext(\neg Y) \end{cases} \tag{10.10}$$

Lift drückt genau aus, wie weit eine Gruppe $A$ von der allgemeinen Verteilung von $Y$ abweicht.

**Knowledge-Based Sampling für Subgruppenentdeckung**

Gegeben $X = \{(x_1, y_1), ..., (x_N, y_N)\}$ und $k$, finde eine Menge $H = \{h_1, ..., h_k\}$

1. Stelle die a priori Verteilung $\pi(y)$ für jedes $y \in Y$ fest.
2. Stratifizieren der Verteilung: $D_1(x_i) = \pi(y_i)^{-1}$ für $i = 1, ..., N$
3. für $t = 1$ bis $k$ do
   - $h_t = RegelLernen(D_t, X)$
   - Kontingenzmatrix für $h_t$ mit Gewichten gemäß $D_t$
   - Lift-Bewertung für $h_t$ gemäß der Kontingenzmatrix
   - $D_{t+1}(x_i) = D_t(x_i) \cdot (Lift_{Dt}(h_t, x))^{-1}$ für $i \in \{1, ..., N\}$
4. Ausgabe $\{h_1, ..., h_k\}$ mit $Lift(h_i)$ (Definition 10.1)

**Subgruppen für die Vorhersage**

- Die Regeln können mit ihrer Gewichtung zu einem Ensemble zusammengefasst werden.
- LiftRatio LR:

$$LR(A \to Y, x) = \begin{cases} \frac{Lift(A \to Y)}{Lift(A \to \neg Y)}, falls & x \in ext(A) \\ \frac{Lift(\neg A \to Y)}{Lift(\neg A \to \neg Y)}, falls & x \in ext(\neg A) \end{cases} \tag{10.11}$$

- Für alle Regeln, wobei $D_0$ die uniforme Verteilung über $X$ ist:

$$\hat{\beta}(x) = \frac{Pr_{D0}[Y]}{Pr_{D0}[\neg Y]} \cdot \prod_{1 \le i \le k} LR_{Di}[(A^i \to Y), x] \tag{10.12}$$

**Was wissen Sie jetzt?**

- Sie haben eine neue Lernaufgabe kennengelernt: Subgruppenendeckung.
- Sie haben neue Gütekritereien kennengelernt: Lift, WRAcc, Spezifizität und Sensitivität
- Für eine Reihe von Experimenten haben Sie ROC und AUC kennengelernt.
- Die Größe von Stichproben in Bezug auf das Risiko, dass das Lernergebnis falsch ist, wurde mit den Schranken von Chernoff und Hoeffding beschränkt.
- Zwei effiziente Ansätze zur Subgruppenentdeckung, von Wrobel und von Scholz, beruhen darauf, dass man nicht alle Beispiele zu betrachten braucht.
- Sie kennen Knowledge-Based Sampling für Subgruppenentdeckung und wie man das Ergebnis für die Klassifikation verwenden kann.

# Kapitel 11

# Häufige Mengen

## 11.1 Einführung

**Datenbanksysteme**

- Entwurf, z.B. mit Entity Relationship Model
- Deklaration
- Speichern der Daten
  - Hauptspeicher, Cache, virtueller Speicher, Platte
  - Indexierung, z.B. mit B-Bäumen
- Anfragen
  - Syntax in SQL, Semantik
  - Pläne zur Ausführung
  - Optimierung
- Änderungen (Transaktionen)

**Relationen**

| Titel | Jahr | Dauer | Foto |
|---|---|---|---|
| Star Wars | 1977 | 124 | farbig |
| Mighty Ducks | 1991 | 104 | farbig |
| Wayne's World | 1992 | 95 | farbig |

- Attribute (Spaltennamen): X=Titel, Jahr, Dauer,Foto
- Tupel (Zeilen): $t : X \to C$; $C$ ist die unendliche Menge der Konstanten
- Relation $r$: Menge von Tupeln mit gleichem Definitionsbereich

**Schema**

| Titel | Jahr | Dauer | Foto |
|---|---|---|---|
| Star Wars | 1977 | 124 | farbig |
| Mighty Ducks | 1991 | 104 | farbig |
| Wayne's World | 1992 | 95 | farbig |

- Name der Relation und Menge von Attributen mit Wertebereichen, hier: Filme(Titel:string, Jahr:integer, Dauer:integer, Foto:{farbig, sw})
- Wertebereiche sind einfache Datentypen: integer, string, date, Aufzählung

**Ein Schema erzeugen – DDL**

Data Definition Language, Teilmenge von SQL
CREATE TABLE FilmeTest (titel VARCHAR2(18), jahr NUMBER); DESCRIBE FilmeTest liefert:

| Name | Type |
|---|---|
| Titel | VARCHAR2(18) |
| Jahr | NUMBER |

**Instanz eines Schemas**

- Relationenschema der Form $R(A_1 : D_1, ...A_p : D_p)$
- $(d, r)$ ist eine Instanz von $R(A_1 : D_1, ..., A_p : D_p)$ gdw.
  - $d \subset C$, alle konkreten Werte sind Teilmengen der unendlichen Menge an Konstanten
  - $dom(r) = \{A_1, ..., A_p\}$, die Instanz hat die selben Attribute
  - $t(A_i) \in D_i$, die Attributwerte der Instanz entsprechen dem Wertebereich des jeweiligen Attributs

Eine Instanz heißt auch *Zustand*. Das Schema ändert sich (fast) nie, der Zustand sehr häufig.

**Eine Instanz erzeugen – DML**

Zeilenweise werden Tupel eingefügt.

- INSERT INTO FilmeTest (Titel, Jahr, Star) VALUES ('Star Wars', 1988, 'C. Fisher');

Wenn man die Daten aus einer anderen Tabelle bekommen kann, darf statt VALUES ein SELECT-Ausdruck stehen, der Tupel aus der gegebenen Tabelle holt.

**Datenbank und Datenbankschema**

- Eine Menge von Relationenschemata ist ein Datenbankschema. *Es gibt nicht nur EINE Tabelle!*
- Eine Menge von Instanzen von Relationenschemata ist ein Datenbankzustand (kurz: Datenbank). *Also lauter gefüllte Tabellen.*
- Metadaten beschreiben Daten. Ein Datenbankschema beschreibt eine Datenbank.

### Datenbankanfragen

**Algebra** Aus Operationen und Variablen oder Konstanten werden Ausdrücke gebildet, z.B. (Un-)Gleichungen.

**Relationenalgebra** Operationen bilden neue Relationen aus gegebenen Relationen.

- Variable und Konstante bezeichnen Relationen.
- Operationen sind
  - Vereinigung, Schnittmenge, Differenz
  - Projektion
  - Selektion
  - Kartesisches Produkt, natürlicher Verbund
- Die Ausdrücke heißen *Anfragen*.

### Projektion

- Eine Relation auf eine andere projizieren. $\pi A_1, ..., A_m(r)$ liefert nur die Attribute $A_1, ..., A_m$ von R. Beispiel: $\pi Titel, Jahr(Filme)$ liefert gerade FilmeTest.
- Eine Relation gemäß Bedingungen projizieren. $\pi_q(r)$, wobei $q$ definiert werden muss. Beispiel: $\pi_q(Filme)$ mit $q(Foto) := Farbe$ benennt das Attribut um; $\pi_q(Filme)$ mit $q := farbig$ ergibt die Relation $\frac{\text{farbig}}{\text{true}}$ Es gibt nur ein Tupel, denn alle Filme waren farbig.

### Selektion

Die Selektion $\sigma\ Cond(r)$ aus eine Relation r ergibt eine Relation mit der Teilmenge von Tupeln von r, die der Bedingung $Cond$ genügen. $Cond$ besteht aus Gleichheit und Vergleichsoperationen.

Die Ergebnisrelation hat dasselbe Schema wie r.

$\sigma\ Dauer(Filme) \geq 120(Filme)$ liefert:

| Titel | Jahr | Dauer | Foto |
|---|---|---|---|
| Star Wars | 1977 | 124 | farbig |

### Structured Query Language – SQL

- $\pi L(\sigma\ Cond(r))$
- SELECT L from r WHERE Cond;
- L ist eine Liste von Attributen
- r ist eine Relation
- Cond ist eine Bedingung
- SELECT Titel from Filme WHERE Jahr(Filme)=1977;

  $\frac{\text{Titel}}{\text{Star Wars}}$

**On-line Analytical Processing (OLAP)**
Ziel: Auffinden interessanter Muster in großen Datenmengen

- Formulierung einer Anfrage
- Extraktion der Daten
- Visualisierung der Ergebnisse
- Analyse der Ergebnisse und Formulierung einer neuen Anfrage

**OLAP-Werkzeuge**

- Datenmenge wird als $p$-dimensionaler Raum aufgefasst
- Identifizierung von „interessanten“ Unterräumen
- In relationalen Datenbanken werden $p$-dimensionale Daten als Relationen mit $p$ Attributen modelliert
- Dimensionsreduktion durch Aggregation der Daten entlang der weggelassenen Dimensionen

**Beispiel: Autoverkäufe**

| Modell | Jahr | Farbe | Anzahl |
|---|---|---|---|
| Opel | 1990 | rot | 5 |
| Opel | 1990 | weiß | 87 |
| Opel | 1990 | blau | 62 |
| Opel | 1991 | rot | 54 |
| Opel | 1991 | weiß | 95 |
| Opel | 1991 | blau | 49 |
| Opel | 1992 | rot | 31 |
| Opel | 1992 | weiß | 54 |
| Opel | 1992 | blau | 71 |
| Ford | 1990 | rot | 64 |
| Ford | 1990 | weiß | 62 |
| Ford | 1990 | blau | 63 |
| Ford | 1991 | rot | 52 |
| Ford | 1991 | weiß | 9 |
| Ford | 1991 | blau | 55 |
| Ford | 1992 | rot | 27 |
| Ford | 1992 | weiß | 62 |
| Ford | 1992 | blau | 39 |

## 11.2 Aggregation in SQL, GROUP BY

**Aggregation in SQL**

- **Aggregatfunktionen:**
  COUNT(), SUM(), MIN(), MAX(), AVG()

  *Beispiel*: SELECT **AVG**(Anzahl) FROM Autoverkäufe

- Aggregation nur über verschiedene Werte

  *Beispiel*: SELECT COUNT(**DISTINCT** Modell) FROM Autoverkäufe

- Aggregatfunktionen liefern einen einzelnen Wert
- Aggregation über mehrere Attribute mit **GROUP BY**

**GROUP BY**
**SELECT** Modell, Jahr, **SUM**(Anzahl)
**FROM** Autoverkäufe
**GROUP BY** Modell, Jahr

- Die Tabelle wird gemäß den Kombinationen der ausgewählten Attributmenge in Gruppen unterteilt
- Jede Gruppe wird über eine Funktion aggregiert
- Das Resultat ist eine Tabelle mit aggregierten Werten, indiziert durch die ausgewählte Attributmenge

**Beispiel: GROUP BY**

| Modell | Jahr | Farbe | Anzahl |
|---|---|---|---|
| Opel | 1990 | rot | 5 |
| Opel | 1990 | weiß | 87 |
| Opel | 1990 | blau | 62 |
| Opel | 1991 | rot | 54 |
| Opel | 1991 | weiß | 95 |
| Opel | 1991 | blau | 49 |
| Opel | 1992 | rot | 31 |
| Opel | 1992 | weiß | 54 |
| Opel | 1992 | blau | 71 |
| Ford | 1990 | rot | 64 |
| Ford | 1990 | weiß | 62 |
| Ford | 1990 | blau | 63 |
| Ford | 1991 | rot | 52 |
| Ford | 1991 | weiß | 9 |
| Ford | 1991 | blau | 55 |
| Ford | 1992 | rot | 27 |
| Ford | 1992 | weiß | 62 |
| Ford | 1992 | blau | 39 |

**SELECT** Modell, Jahr, **SUM**(Anzahl) **FROM** Autoverkäufe
**GROUP BY** Modell, Jahr

| Modell | Jahr | SUM(Anzahl) |
|---|---|---|
| Opel | 1990 | 154 |
| Opel | 1991 | 198 |
| Opel | 1992 | 156 |
| Ford | 1990 | 189 |
| Ford | 1991 | 116 |
| Ford | 1992 | 128 |

**Roll Up**

Gleiche Anfrage in unterschiedlichen Detailierungsgraden

- Verminderung des Detailierungsgrades = **Roll Up**
- Erhöhung des Detailierungsgrades = **Drill Down**

*Beispiel: Autoverkäufe*

- Roll Up über drei Ebenen
- Daten werden nach Modell, dann nach Jahr, dann nach Farbe aggregiert
- Die Verkaufszahlen werden zuerst für jedes Modell aus jedem Jahr in jeder Farbe aufgelistet, dann werden alle Verkaufszahlen des gleichen Modells und Jahres aufsummiert und daraus die Verkaufszahlen der Modelle berechnet

**GROUP BY: Roll Up**

| Modell | Jahr | Farbe | Anzahl nach Modell, Jahr, Farbe | Anzahl nach Modell, Jahr | Anzahl nach Modell |
|---|---|---|---|---|---|
| Opel | 1990 | rot | 5 | | |
| | | weiß | 87 | | |
| | | blau | 62 | | |
| | | | | 154 | |
| | 1991 | rot | 54 | | |
| | | weiß | 95 | | |
| | | blau | 49 | | |
| | | | | 198 | |
| | 1992 | rot | 31 | | |
| | | weiß | 54 | | |
| | | blau | 71 | | |
| | | | | 156 | |
| | | | | | 508 |

## 11.3 Probleme mit GROUP BY

### Probleme mit GROUP BY: Roll Up

- Tabelle ist nicht relational, da man wegen der leeren Felder (Null-Werte) keinen Schlüssel festlegen kann.
- Die Zahl der Spalten wächst mit der Zahl der aggregierten Attribute
- Um das exponentielle Anwachsen der Spaltenanzahl zu vermeiden, wird der ALL-Wert eingeführt.
- Der ALL-Wert repräsentiert die Menge, über die die Aggregation berechnet wird.

*Beispiel:* Ein ALL in der Spalte Farbe bedeutet, dass in der Anzahl dieser Zeile die Verkaufszahlen der roten, weißen und blauen Autos zusammengefasst sind.

**GROUP BY: Roll Up mit ALL**

| Modell | Jahr | Farbe | Anzahl |
|---|---|---|---|
| Opel | 1990 | rot | 5 |
| Opel | 1990 | weiß | 87 |
| Opel | 1990 | blau | 62 |
| Opel | 1990 | ALL | 154 |
| Opel | 1991 | rot | 54 |
| Opel | 1991 | weiß | 95 |
| Opel | 1991 | blau | 49 |
| Opel | 1991 | ALL | 198 |
| Opel | 1992 | rot | 31 |
| Opel | 1992 | weiß | 54 |
| Opel | 1992 | blau | 71 |
| Opel | 1992 | ALL | 156 |
| Opel | ALL | ALL | 506 |

Erzeugung der Tabelle mit SQL:

```
SELECT Modell, 'ALL', 'ALL', SUM(Anzahl)
    FROM Autoverkäufe
    WHERE Modell = 'Opel'
    GROUP BY Modell
  UNION
    SELECT Modell, Jahr, 'ALL', SUM(Anzahl)
    FROM Autoverkäufe
    WHERE Modell = 'Opel'
    GROUP BY Modell, Jahr
  UNION
    SELECT Modell, Jahr, Farbe, SUM(Anzahl)
    FROM Autoverkäufe
    WHERE Modell = 'Opel'
    GROUP BY Modell, Jahr, Farbe
```

### Probleme mit GROUP BY: Roll Up

- Beispiel war ein einfaches dreidimensionales Roll Up
- Eine Aggregation über $p$ Dimensionen erfordert $p$ Unions

- Roll Up ist asymmetrisch:
  Verkäufe sind nach Jahr, aber nicht nach Farbe aggregiert

**Kreuztabellen**

Symmetrische Darstellung mehrdimensionaler Daten und Aggregationen

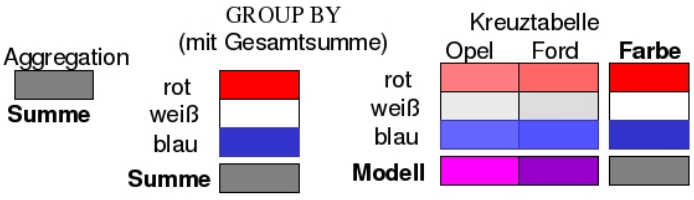
| Opel | 1990 | 1991 | 1992 | Total (ALL) |
|---|---|---|---|---|
| rot | 5 | 54 | 31 | 90 |
| weiß | 87 | 95 | 54 | 236 |
| blau | 62 | 49 | 71 | 182 |
| Total (ALL) | 154 | 198 | 156 | 508 |

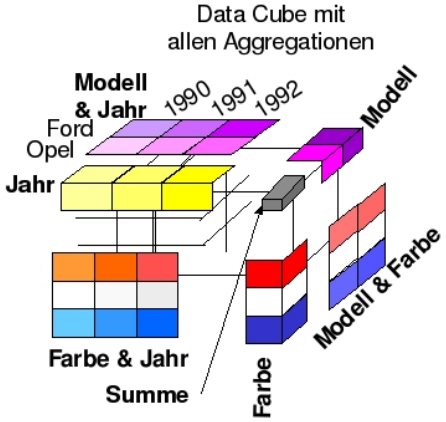
Diese Kreuztabelle ist eine zweidimensionale Aggregation
Nimmt man noch andere Automodelle hinzu, kommt für jedes Modell eine weitere Ebene hinzu
Man erhält eine dreidimensionale Aggregation

## 11.4 Der Cube-Operator

**Der CUBE-Operator**

$p$-dimensionale Generalisierung der bisher genannten Konzepte
Der 0D Data Cube ist ein Punkt
Der 1D Data Cube ist eine Linie mit einem Punkt
Der 2D Data Cube ist eine Kreuztabelle
Der 3D Data Cube ist ein Würfel mit drei sich überschneidenden Kreuztabellen
(Gray, Chaudhuri, Bosworth, Layman 1997)




**Der CUBE-Operator**

- Beispiel: **SELECT** Modell, Jahr, Farbe, **SUM**(Anzahl)
  **FROM** Autoverkäufe
  **GROUP BY CUBE** Modell, Jahr, Farbe

  (Der CUBE-Operator ist nicht für jede Datenbank direkt verfügbar)

- Der Cube-Operator erzeugt eine Tabelle, die sämtliche Aggregationen enthält

- Es werden GROUP BYs für alle möglichen Kombinationen der Attribute berechnet
- Die Erzeugung der Tabelle erfordert die Generierung der Potenzmenge der zu aggregierenden Spalten.
- Bei $p$ Attributen werden $2^p$ GROUP BYs berechnet
- Sei $C_1, C_2, \ldots, C_p$ die Kardinalität der $p$ Attribute, dann ist die Kardinalität der resultierenden Data Cube-Relation $\prod(C_i + 1)$

### Data Cube des Beispiels

| Modell | Jahr | Farbe | Anzahl |
|---|---|---|---|
| Opel | 1990 | rot | 5 |
| Opel | 1990 | weiß | 87 |
| Opel | 1990 | blau | 62 |
| Opel | 1991 | rot | 54 |
| Opel | 1991 | weiß | 95 |
| Opel | 1991 | blau | 49 |
| Opel | 1992 | rot | 31 |
| Opel | 1992 | weiß | 54 |
| Opel | 1992 | blau | 71 |
| Ford | 1990 | rot | 64 |
| Ford | 1990 | weiß | 62 |
| Ford | 1990 | blau | 63 |
| Ford | 1991 | rot | 52 |
| Ford | 1991 | weiß | 9 |
| Ford | 1991 | blau | 55 |
| Ford | 1992 | rot | 27 |
| Ford | 1992 | weiß | 62 |
| Ford | 1992 | blau | 39 |

### Data Cube des Beispiel

| Modell | Jahr | Farbe | Anzahl |
|---|---|---|---|
| Opel | 1990 | rot | 5 |
| Opel | 1990 | weiß | 87 |
| Opel | 1990 | blau | 62 |
| Opel | 1990 | ALL | 154 |
| Opel | 1991 | rot | 54 |
| Opel | 1991 | weiß | 95 |
| Opel | 1991 | blau | 49 |
| Opel | 1991 | ALL | 198 |
| Opel | 1992 | rot | 31 |
| Opel | 1992 | weiß | 54 |
| Opel | 1992 | blau | 71 |
| Opel | 1992 | ALL | 156 |
| Opel | ALL | rot | 90 |
| Opel | ALL | weiß | 236 |
| Opel | ALL | blau | 182 |
| Opel | ALL | ALL | 508 |
| Ford | 1990 | rot | 64 |
| Ford | 1990 | weiß | 72 |
| Ford | 1990 | blau | 63 |
| Ford | 1990 | ALL | 189 |
| Ford | 1991 | rot | 52 |
| Ford | 1991 | weiß | 9 |
| Ford | 1991 | blau | 55 |
| Ford | 1991 | ALL | 116 |

| Modell | Jahr | Farbe | Anzahl |
|---|---|---|---|
| Ford | 1992 | rot | 27 |
| Ford | 1992 | weiß | 62 |
| Ford | 1992 | blau | 39 |
| Ford | 1992 | ALL | 128 |
| Ford | ALL | rot | 143 |
| Ford | ALL | weiß | 133 |
| Ford | ALL | blau | 157 |
| Ford | ALL | ALL | 433 |
| ALL | 1990 | rot | 69 |
| ALL | 1990 | weiß | 149 |
| ALL | 1990 | blau | 125 |
| ALL | 1990 | ALL | 343 |
| ALL | 1991 | rot | 106 |
| ALL | 1991 | weiß | 104 |
| ALL | 1991 | blau | 104 |
| ALL | 1991 | ALL | 314 |
| ALL | 1992 | rot | 58 |
| ALL | 1992 | weiß | 116 |
| ALL | 1992 | blau | 110 |
| ALL | 1992 | ALL | 284 |
| ALL | ALL | rot | 233 |
| ALL | ALL | weiß | 369 |
| ALL | ALL | blau | 339 |
| ALL | ALL | ALL | 941 |

## 11.5 Implementierung des Data Cube

### Implementationsalternativen

- Physische Materialisierung des gesamten Data Cube:

  - beste Antwortzeit
  - hoher Speicherplatzbedarf
- Keine Materialisierung:
  - jede Zelle wird nur bei Bedarf aus den Rohdaten berechnet
  - kein zusätzlicher Speicherplatz
  - schlechte Antwortzeit
- Materialisierung von Teilen des Data Cube:
  - Werte vieler Zellen sind aus Inhalt anderer Zellen berechenbar
  - diese Zellen nennt man „abhängige“ Zellen
  - Zellen, die einen ALL-Wert enthalten, sind abhängig
  - Problem: Welche Zellen des Data Cube materialisieren?
  - Zellen des Data Cube entsprechen SQL Anfragen (Sichten)

**Abhängigkeit von Sichten**

Die Abhängigkeitsrelation $\leq$ zwischen zwei Anfragen $Q_1$ und $Q_2$

$Q_1 \leq Q_2$ gdw. $Q_1$ kann beantwortet werden, indem die Ergebnisse von $Q_2$ verwendet werden. $Q_1$ ist abhängig von $Q_2$

- Anfragen bilden einen Verband unter folgenden Voraussetzungen:
  1. $\leq$ ist eine Halbordnung und
  2. es gibt ein maximales Element (eine oberste Sicht)
- Der Verband wird durch eine Menge von Anfragen (Sichten) $L$ und der Abhängigkeitsrelation $\leq$ definiert und mit $\langle L, \leq \rangle$ bezeichnet
- Ein Verband wird dargestellt durch einen Graphen, in dem die Anfragen die Knoten sind und $\leq$ die Kanten.

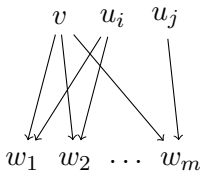
**Auswahl von Sichten zur Materialisierung**

- Optimierungsproblem, das unter folgenden Bedingungen gelöst werden soll:
  - Die durchschnittliche Zeit für die Auswertung der Anfragen soll minimiert werden.
  - Man beschränkt sich auf eine feste Anzahl von Sichten, die materialisiert werden sollen, unabhängig von deren Platzbedarf
- Das Optimierungsproblem ist NP-vollständig.
- Heuristiken für Approximationslösungen: Greedy-Algorithmus
- Der Greedy-Algorithmus verhält sich nie zu schlecht: Man kann zeigen, dass die Güte mindestens 63% beträgt (Harinayaran, Rajaraman, Ullman 1996).

**Der Greedy Algorithmus**

- Gegeben ein Verband mit Speicherkosten $C(v)$ für jede Sicht $v$
- Annahme: Speicherkosten = Anzahl der Reihen in der Sicht
- Beschränkung auf $k$ materialisierte Sichten
- Nach Auswahl einer Menge $S$ von Sichten wird der Nutzen der Sicht $v$ relativ zu $S$ mit $B(v, S)$ bezeichnet und wie folgt definiert:

1. Für jede Sicht $w \leq v$ wird $B_w$ berechnet:
   (a) Sei $u$ die Sicht mit den geringsten Kosten in $S$, so dass $w \leq u$
   (b) $B_w = \begin{cases} C(u) - C(v), & \text{falls } C(v) < C(u) \\ 0, & \text{sonst} \end{cases}$

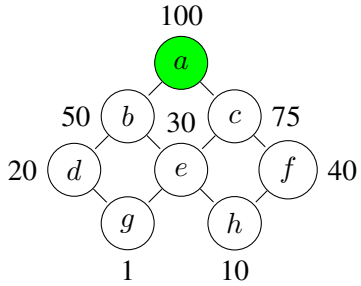



2. $B(v, S) = \sum_{w \leq v} B_w$

**Der Greedy Algorithmus**

1. $S = \{\text{oberste Sicht}\}$
2. for $i = 1$ to $k$ do begin
3. Wähle die Sicht $v \notin S$, so dass $B(v, S)$ maximal ist;
4. $S = S \cup \{v\}$
5. end;
6. return $S$;

**Beispiel**



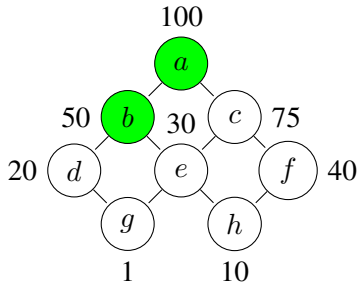

Für jede Sicht $w \leq v$ wird $B_w$ berechnet:

- Sei $u$ die Sicht mit den geringsten Kosten in $S$, so dass $w \leq u$
- $B_w = \begin{cases} C(u) - C(v), & \text{falls } C(v) < C(u) \\ 0, & \text{sonst} \end{cases}$

$B(v, S) = \sum_{w \leq v} B_w$

$S : \{a\}$
$v := b, u := a$
Knoten $w := b, d, g, e, h$ haben $C(a) - C(b) = 50$.
Greedy Auswahl: $b$ wird zusätzlich materialisiert

**Beispiel**




Für jede Sicht $w \leq v$ wird $B_w$ berechnet:

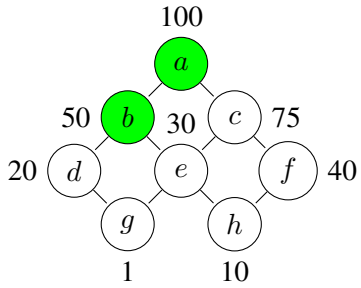
- Sei $u$ die Sicht mit den geringsten Kosten in $S$, so dass $w \leq u$
- $B_w = \begin{cases} C(u) - C(v), & \text{falls } \mathbf{C(v) < C(u)} \\ 0, & \text{sonst} \end{cases}$

$B(v, S) = \sum_{w \leq v} B_w$

$S : \{a, b\}$

- $v := d, u := b, w := d, g \rightarrow B_{d,g} = C(b) - C(d) = 30$
- $v := c, u := a, w := c, f \rightarrow B_{c,f} = C(a) - C(c) = 25$; $w := e, h, g \rightarrow B_{e,h,g} = 0$ weil $C(u) < C(v)$ statt $C(v) < C(u)$

**Beispiel**




Für jede Sicht $w \leq v$ wird $B_w$ berechnet:

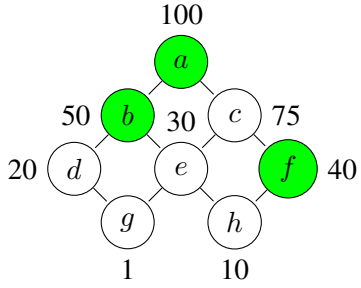
- Sei $u$ die Sicht mit den **geringsten Kosten** in $S$, so dass $w \leq u$
- $B_w = \begin{cases} C(u) - C(v), & \text{falls } C(v) < C(u) \\ 0, & \text{sonst} \end{cases}$

$B(v, S) = \sum_{w \leq v} B_w$

- $v := e, u := b, w := e, g, h \rightarrow B_{e,g,h} = C(b) - C(e) = 20$
- $v := f, u := a, w := f \rightarrow B_f = C(a) - C(f) = 60, u := b, w := h \rightarrow C(b) - C(f) = 10$

Gewählt wird $f$.

**Beispiel**




Für jede Sicht $w \leq v$ wird $B_w$ berechnet:

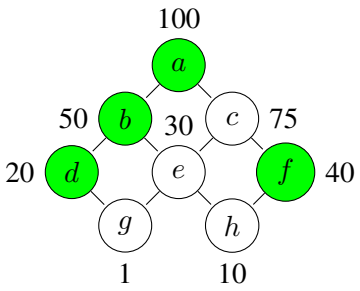
- Sei $u$ die Sicht mit den geringsten Kosten in $S$, so dass $w \leq u$
- $B_w = \begin{cases} C(u) - C(v), & \text{falls } C(v) < C(u) \\ 0, & \text{sonst} \end{cases}$

$S = \{a, b, f\}$

$B(v, S) = \sum_{w \leq v} B_w$

- $v := c, u := a, w := c \rightarrow B_c = C(a) - C(c) = 25$ $w := e, h, g \rightarrow B_{e,h,g} = 0$
- $v := d, u := b, w := d, g \rightarrow B_{d,g} = C(b) - C(d) = 30$
- $v := e, u := b, w := e, g \rightarrow B_{e,g} = C(b) - C(e) = 20$ $u := f, w := h \rightarrow B_h = C(f) - C(e) = 10$

**Beispiel**




| | Erste Wahl | Zweite Wahl | Dritte Wahl |
|---|---|---|---|
| b | 50 x 5 = 250 | | |
| c | 25 x 5 = 125 | 25 x 2 = 50 | 25 x 1 = 25 |
| d | 80 x 2 = 160 | 30 x 2 = 60 | 30 x 2 = 60 |
| e | 70 x 3 = 210 | 20 x 3 = 60 | 20 + 20 + 10 = 50 |
| f | 60 x 2 = 120 | 60 + 10 = 70 | |
| g | 99 x 1 = 99 | 49 x 1 = 49 | 49 x 1 = 49 |
| h | 90 x 1 = 90 | 40 x 1 = 40 | 30 x 1 = 30 |

$S : \{a\}, S : \{a, b\}, S : \{a, b, f\}, S : \{a, b, d, f\}$
Greedy Auswahl: $b, d, f$ werden zusätzlich materialisiert

## 11.6 Zusammenfassung

**Was wissen Sie jetzt?**

- Möglichkeiten und Grenzen der Aggregation in SQL
- Einführung von Data Cubes zur Unterstützung von Aggregationen über $p$ Dimensionen
- Greedy-Algorithmus zur Auswahl einer festen Anzahl von Sichten, die materialisiert werden

## 11.7 Apriori

**Lernen von Assoziationsregeln**

Gegeben:

- $R$ eine Menge von Objekten, die binäre Werte haben
- $t$ eine Transaktion, $t \subseteq R$
- $r$ eine Menge von Transaktionen
- $s_{min} \in [0, 1]$ die minimale Unterstützung,
- $conf_{min} \in [0, 1]$ die minimale Konfidenz

Finde alle Regeln $c$ der Form $X \rightarrow Y$, wobei $X \subseteq R, Y \subseteq R, X \cap Y = \{\}$

$$s(r,c) = \frac{|\{t \in r \mid X \cup Y \subseteq t\}|}{|r|} \geq s_{min} \tag{11.1}$$

$$conf(r,c) = \frac{|\{t \in r \mid X \cup Y \subseteq t\}|}{|\{t \in r \mid X \subseteq r\}|} \geq conf_{min} \tag{11.2}$$

**Binäre Datenbanken**

Sei $R$ eine Menge von Objekten, die binäre Werte haben, und $r$ eine Menge von Transaktionen, dann ist $t \in r$ eine Transaktion und die Objekte mit dem Wert 1 sind eine Teilmenge aller Objekte.

$$R = \{A, B, C\}$$
$$t = \{B, C\} \subseteq R$$

| A | B | C | ID |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 2 |
| 0 | 1 | 1 | 3 |
| 1 | 0 | 0 | 4 |

**Warenkorbanalyse**

| Aftershave | Bier | Chips | EinkaufsID |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 2 |
| 0 | 1 | 1 | 3 |
| 1 | 0 | 0 | 4 |

$\{\text{Aftershave}\} \rightarrow \{\text{Bier}\}$ $\quad s = \frac{1}{4}, conf = \frac{1}{2}$

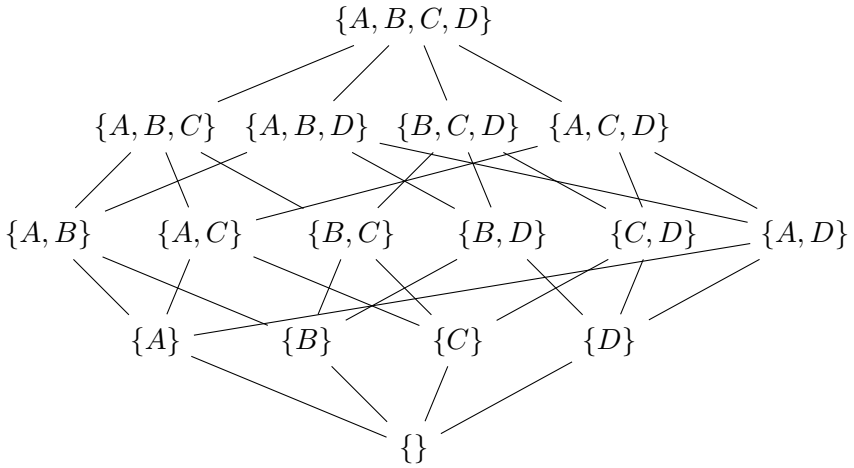
$\{\text{Aftershave}\} \rightarrow \{\text{Chips}\}$ $\quad s = 0$

$\{\text{Bier}\} \rightarrow \{\text{Chips}\}$ $\quad s = \frac{1}{2}, conf = \frac{2}{3}$ (zusammen anbieten?)

$\{\text{Chips}\} \rightarrow \{\text{Aftershave}\}$ $\quad s = 0$

$\{\text{Aftershave}\} \rightarrow \{\text{Bier, Chips}\}$ $\quad s = 0$

**Verband**




**Ordnungsrelation**

- Hier ist die Ordnungsrelation die Teilmengenbeziehung.
- Eine Menge $S_1$ ist größer als eine Menge $S_2$, wenn $S_1 \supseteq S_2$.
- Eine kleinere Menge ist allgemeiner.

**Assoziationsregeln**

$LH$: Assoziationsregeln sind keine logischen Regeln!

- In der Konklusion können mehrere Attribute stehen
- Attribute sind immer nur binär.
- Mehrere Assoziationsregeln zusammen ergeben kein Programm.

$LE$: Binärvektoren (Transaktionen)

- Attribute sind eindeutig geordnet.

Aufgabe:

- Aus häufigen Mengen Assoziationsregeln herstellen

**Apriori Algorithmus**

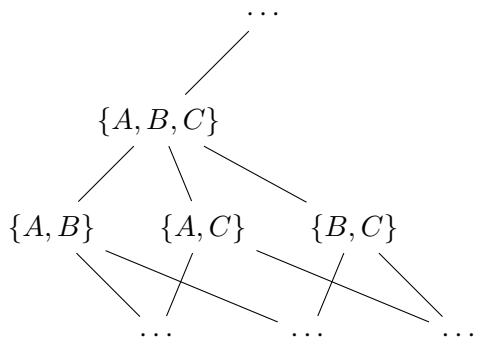
*(Agrawal, Mannila, Srikant, Toivonen, Verkamo 1996)*

$LH$ des Zwischenschritts: Häufige Mengen $L_k = X \cup Y$ mit $k$ Objekten (large itemsets, frequent sets)

- Wenn eine Menge häufig ist, so auch all ihre Teilmengen. (Anti-Monotonie)
- Wenn eine Menge selten ist, so auch all ihre Obermengen. (Monotonie)
- Wenn $X$ in $L_{k+1}$ dann alle $S_i \subseteq X$ in $L_k$ (Anti-Monotonie)
- Alle Mengen $L_k$, die $k-1$ Objekte gemeinsam haben, werden vereinigt zu $L_{k+1}$.

Dies ist der Kern des Algorithmus, die Kandidatengenerierung.

**Beispiel**




- Wenn $\{A, B, C\}$ häufig ist, dann sind auch $\{A, B\}$, $\{A, C\}$, $\{B, C\}$ häufig.
- Das bedeutet, daß$\{A, B\}$, $\{A, C\}$, $\{B, C\}$ ($k = 2$) häufig sein müssen, damit $\{A, B, C\}$ ($k + 1 = 3$) häufig sein *kann*.
- Also ergeben die häufigen Mengen aus $L_k$ die Kandidaten $C_{k+1}$

**Beispiel**

Gesucht werden Kandidaten mit $k + 1 = 5$

$L_4 = \{\{ABCD\}, \{ABCE\}, \{ABDE\}, \{ACDE\}, \{BCDE\}\}$

- $k - 1$ Stellen gemeinsam vereinigen zu:

$$l = \{ABCDE\}$$

- Sind alle $k$ langen Teilmengen von $l$ in $L_4$?
  $\{ABCD\}\{ABCE\}\{ABDE\}\{ACDE\}\{BCDE\}$ - ja!
- Dann wird $l$ Kandidat $C_5$.

$L4 = \{\{ABCD\}, \{ABCE\}\}$

$l = \{ABCDE\}$

- Sind alle Teilmengen von $l$ in $L_4$?
  $\{ABCD\}\{ABCE\}\{ABDE\}\{ACDE\}\{BCDE\}$ - nein!
- Dann wird $l$ nicht zum Kandidaten.

**Kandidatengenerierung**

- Erzeuge-Kandidaten($L_k$)
  - $C_{k+1} := \{\}$
  - For all $l_1, l_2$ in $L_k$, sodass

$$l_1 = \{i_1, \cdots, i_{k-1}, i_k\} \text{ und}$$

$$l_2 = \{i_1, \cdots, i_{k-1}, i'_k\} i'_k < i_k$$

    * $l := \{i_1, \cdots, i_{k-1}, i_k, i'_k\}$
    * if alle $k$-elementigen Teilmengen von $l$ in $L_k$ sind, then

$$C_{k+1} := C_{k+1} \cup \{l\}$$

    * return $C_{k+1}$

- Prune($C_{k+1}, r$) vergleicht Häufigkeit von Kandidaten mit $s_{min}$.

**Häufige Mengen**

- Häufige-Mengen($R$, $r$, $s_{min}$)
  - $C_1 := \cup_{i \in R} i$, $k = 1$
  - $L_1 :=$ Prune$(C_1)$
  - while $L_k \neq \{\}$
    - $C_{k+1} :=$ Erzeuge-Kandidaten$(L_k)$
    - $L_{k+1} :=$ Prune$(C_{k+1}, r)$
    - $k := k + 1$
    - return $\cup_{j=2}^{k} L_j$

**APRIORI**

- Apriori($R$, $r$, $s_{min}$, $conf_{min}$)
  - $L :=$Häufige-Mengen($R$, $r$, $s_{min}$)
  - $c :=$Regeln ($L$, $conf_{min}$)
  - return $c$

**Regelgenerierung**

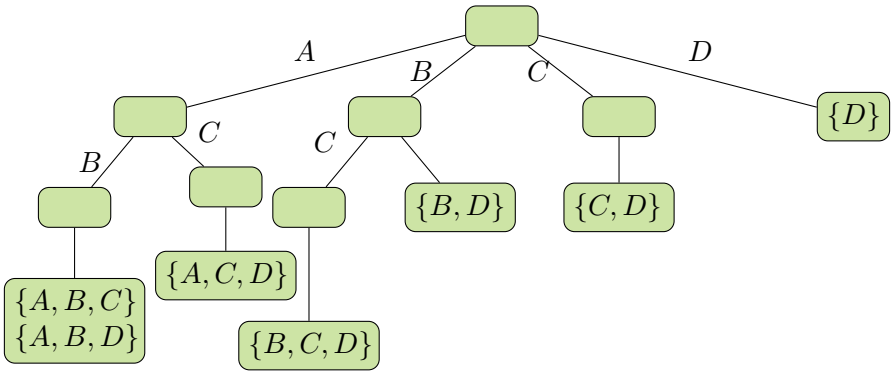
Aus den häufigen Mengen werden Regeln geformt. Wenn die Konklusion länger wird, kann die Konfidenz sinken. Die Ordnung der Attribute wird ausgenutzt:

$$\begin{array}{lll} l_1 = \{i_1, \cdots, i_{k-1}, i_k\} & c_1 = \{i_1, \cdots, i_{k-1}\} \rightarrow \{i_k\} & conf_1 \\ l_1 = \{i_1, \cdots, i_{k-1}, i_k\} & c_2 = \{i_1, \cdots\} \rightarrow \{i_{k-1}, i_k\} & conf_2 \\ \cdots & \cdots & \cdots \\ l_1 = \{i_1, \cdots, i_{k-1}, i_k\} & c_k = \{i_1\} \rightarrow \{\cdots, i_{k-1}, i_k\} & conf_k \end{array}$$

$$conf_1 \geq conf_2 \geq \cdots \geq conf_k$$

**Implementierung**

- Hash-Tree für den Präfixbaum, der sich aus der Ordnung der Elemente in den Mengen ergibt.
- An jedem Knoten werden Schlüssel und Häufigkeit gespeichert.
- Dynamischer Aufbau




**Was wissen Sie jetzt?**

- Assoziationsregeln sind keine logischen Regeln.
- Anti-Monotonie der Häufigkeit: Wenn eine Menge häufig ist, so auch all ihre Teilmengen.
- Man erzeugt häufige Mengen, indem man häufige Teilmengen zu einer Menge hinzufügt und diese Mengen dann auf Häufigkeit testet. Bottom-up Suche im Verband der Mengen.
- Monotonie der Seltenheit: Wenn eine Teilmenge selten ist, so auch jede Menge, die sie enthält.
- Man beschneidet die Suche, indem Mengen mit einer seltenen Teilmenge nicht weiter betrachtet werden.

**Probleme von Apriori**

- Im schlimmsten Fall ist Apriori exponentiell in $R$, weil womöglich alle Teilmengen gebildet würden. In der Praxis sind die Transaktionen aber spärlich besetzt. Die Beschneidung durch $s_{min}$ und $conf_{min}$ reicht bei der Warenkorbanalyse meist aus.
- Apriori liefert unglaublich viele Regeln.
- Die Regeln sind höchst redundant.
- Die Regeln sind irreführend, weil die Kriterien die a priori Wahrscheinlichkeit nicht berücksichtigen. Wenn sowieso alle Cornflakes essen, dann essen auch hinreichend viele Fußballer Cornflakes.

**Prinzipien für Regelbewertungen**

1. $RI(A \to B) = 0$, wenn $|A \to B| = \frac{(|A||B|)}{|r|}$
   $A$ und $B$ sind unabhängig.
2. $RI(A \to B)$ steigt monoton mit $|A \to B|$.
3. $RI(A \to B)$ fällt monoton mit $|A|$ oder $|B|$.

Also:

- $RI > 0$, wenn $|A \to B| > \frac{(|A||B|)}{|r|}$, d.h. wenn $A$ positiv mit $B$ korreliert ist.

- $RI < 0$, wenn $|A \to B| > \frac{(|A||B|)}{|r|}$, d.h. wenn $A$ negativ mit $B$ korreliert ist.

Wir wissen, dass immer $|A \to B| \leq |A| \leq |B|$ gilt, also

- $RI_{min}$, wenn $|A \to B| = |A|$ oder $|A| = |B|$
- $RI_{max}$, wenn $|A \to B| = |A| = |B|$

Piatetsky-Shapiro 1991

**Konfidenz**

- Die Konfidenz erfüllt die Prinzipien nicht! (Nur das 2.) Auch unabhängige Mengen $A$ und $B$ werden als hoch-konfident bewertet.
- Die USA-Census-Daten liefern die Regel

  $$\text{aktiv-militär} \to \text{kein-Dienst-in-Vietnam}$$

  mit 90% Konfidenz. Tatsächlich ist $s(\text{kein-Dienst-in-Vietnam}) = 95\%$ Es wird also wahrscheinlicher, wenn aktiv-militär gegeben ist!
- Gegeben eine Umfrage unter 2000 Schülern, von denen $60\%$ Basketball spielen, $75\%$ Cornflakes essen. Die Regel

  $$\text{Basketball} \to \text{Cornflakes}$$

  hat Konfidenz $66\%$ Tatsächlich senkt aber Basketball die Cornflakes Häufigkeit!

**Signifikanztest**

- Ein einfaches Maß, das die Prinzipien erfüllt, ist:

  $$|A \to B| - \frac{|A||B|}{|r|}$$

- Die Signifikanz der Korrelation zwischen $A$ und $B$ ist:

  $$\frac{|A \to B| - \frac{|A||B|}{|r|}}{\sqrt{|A||B|\left(1 - \frac{A}{r}\right)\left(1 - \frac{|B|}{|r|}\right)}}$$

**Sicherheitsmaß**

Shortliffe, Buchanan 1990 führten ein Sicherheitsmaß$CF$ ein (für Regeln in Wissensbasen)

- Wenn $conf(A \to B) > s(B)$
  $CF(A \to B) = conf(A \to B) - \frac{s(B)}{1-s(B)}$
- Wenn $conf(A \to B) < s(B)$
  $CF(A \to B) = conf(A \to B)$
- Sonst
  $CF(A \to B) = 0$

Das Sicherheitsmaßbefolgt die Prinzipien für Regelbewertung. Wendet man Signifikanztest oder Sicherheitsmaßan, erhält man weniger (irrelevante, irreführende) Assoziationsregeln.

**Was wissen Sie jetzt?**

- Sie haben drei Prinzipien für die Regelbewertung kennengelernt:
  - Unabhängige Mengen sollen mit 0 bewertet werden.
  - Der Wert soll höher werden, wenn die Regel mehr Belege hat.
  - Der Wert soll niedriger werden, wenn die Mengen weniger Belege haben.
- Sie haben drei Maße kennen gelernt, die den Prinzipien genügen:
  - Einfaches Maß
  - statistisches Maß und
  - Sicherheitsmaß

## 11.8 FP-Tree

# Literaturverzeichnis

[Jiawei Han and Micheline Kamber, 2000] Jiawei Han and Micheline Kamber Data Mining: Concepts and Techniques Slides for Textbook - Chapter 6 *Intelligent Database Systems Research Lab*. *School of Computing Science*. Simon Fraser University, Canada. http://www.cs.sfu.ca

**Mining Frequent Patterns Without Candidate Generation**

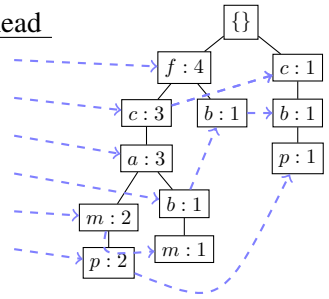
- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database scans
- Develop an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only!

**Construct FP-tree from a Transaction DB**

| TID | Items bought | (ordered) frequent items |
|---|---|---|
| 100 | $\{f, a, c, d, g, i, m, p\}$ | $\{f, c, a, m, p\}$ |
| 200 | $\{a, b, c, f, l, m, o\}$ | $\{f, c, a, b, m\}$ |
| 300 | $\{b, f, h, j, o\}$ | $\{f, b\}$ |
| 400 | $\{b, c, k, s, p\}$ | $\{c, b, p\}$ |
| 500 | $\{a, f, c, e, l, p, m, n\}$ | $\{f, c, a, m, p\}$ |

$support_{min} = 0.5$

| Item | *freq* | head |
|---|---|---|
| f | 4 | |
| c | 4 | |
| a | 3 | |
| b | 3 | |
| m | 3 | |
| p | 3 | |




1 Scan DB once, find frequent 1-itemset (single item pattern)

2 Order frequent items in frequency descending order

3 Scan DB again, construct FP-tree

**$FP$-Tree**

- Ein $FP$ Tree ist nach Häufigkeiten (von oben nach unten) geordnet.
- Ein $FP$ Tree fasst Transaktionen als Wörter auf und stellt gemeinsame Präfixe verschiedener Wörter dar.
- Für jede Transaktion lege einen Pfad im $FP$ Tree an:
  - Pfade mit gemeinsamem Präfix - Häufigkeit $+1$, Suffix darunter hängen.
  - Kein gemeinsamer Präfix vorhanden - neuen Zweig anlegen.
- Header Tabelle verweist auf das Vorkommen der items im Baum. Auch die Tabelle ist nach Häufigkeit geordnet.

**Benefits of the $FP$-tree Structure**

- Completeness:
  - never breaks a long pattern of any transaction
  - preserves complete information for frequent pattern mining
- Compactness:
  - reduce irrelevant information - infrequent items are gone
  - frequency descending ordering: more frequent items are more likely to be shared
  - never be larger than the original database (if not count node-links and counts)
  - Example: For Connect-4 DB, compression ratio could be over 100

**Mining Frequent Patterns Using $FP$-tree**

- General idea (divide-and-conquer)
  - Recursively grow frequent pattern path using the FP-tree
- Method
  - For each item, construct its *conditional pattern-base*, and then its *conditional $FP$-tree*
  - Repeat the process on each newly created conditional $FP$-tree
  - Until the resulting $FP$-tree is *empty*, or it contains *only one path* (single path will generate all the combinations of its sub-paths, each of which is a frequent pattern)

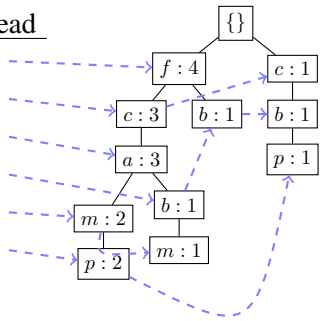
**Major Steps to Mine $FP$-tree**

1. Construct conditional pattern base for each node in the $FP$-tree
2. Construct conditional $FP$-tree from each conditional pattern-base
3. Recursively mine conditional $FP$-trees and grow frequent patterns obtained so far
   - If the conditional $FP$-tree contains a single path, simply enumerate all the patterns

**Step 1: From $FP$-tree to Conditional Pattern Base**

- Starting at the frequent header table in the $FP$-tree
- Traverse the $FP$-tree by following the link of each frequent item
- Accumulate all of transformed prefix paths of that item to form a conditional pattern base

| Item | $freq$ | head |
|---|---|---|
| f | 4 | |
| c | 4 | |
| a | 3 | |
| b | 3 | |
| m | 3 | |
| p | 3 | |




| Item | conditional pattern base |
|---|---|
| c | $f:3$ |
| a | $fc:3$ |
| b | $fca:1, f:1, c:1$ |
| m | $fca:2, fcab:1$ |
| p | $fcam:2, cb:1$ |

**Vom FP Tree zur Cond. Pattern Base**

- Die Header Tabelle von unten (selten) nach oben durchgehen. Die Verweise führen zu den Pfaden, in denen das item vorkommt.
  - Das item wird als Suffix betrachtet und alle Präfixe davon als Bedingungen für dies Suffix.
  - Die Häufigkeiten der Präfixe werden von unten nach oben propagiert.

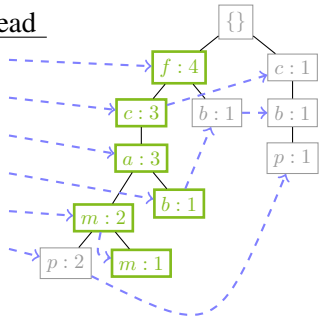
**Properties of $FP$-tree for Conditional Pattern Base Construction**

- Node-link property
  - For any frequent item $a_i$, all the possible frequent patterns that contain $a_i$ can be obtained by following $a_i$'s node-links, starting from $a_i$'s head in the $FP$-tree header
- Prefix path property
  - To calculate the frequent patterns for a node $a_i$ in a path $P$, only the prefix sub-path of $a_i$ in $P$ need to be accumulated, and its frequency count should carry the same count as node $a_i$.

**Step 2: Construct Conditional $FP$-tree**

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base

| Item | $freq$ | head |
|---|---|---|
| f | 4 | |
| c | 4 | |
| a | 3 | |
| b | 3 | |
| m | 3 | |
| p | 3 | |




$\Longrightarrow$

$m$-conditional pattern base:
$fca : 2, fcab : 1$

$m$-conditional $FP$-tree:

{}
|
$f : 3$
|
$c : 3$
|
$a : 3$

**$m$-conditional $FP$-tree**

$m$-conditional pattern base:
$fca : 2, fcab : 1$

$m$-conditional $FP$-tree:

{}
|
$f : 3$
|
$c : 3$
|
$a : 3$

$\Longrightarrow$

All frequent patterns concerning $m$

- $m$
- $fm, cm, am$
- $fcm, fam, cam$
- $fcam$

**Mining Frequent Patterns by Creating Conditional Pattern-Bases**

| Item | Conditional pattern-base | conditional $FP$-tree |
|---|---|---|
| p | $\{(fcam : 2), (cb : 1)\}$ | $\{(c : 3)\}\|p$ |
| m | $\{(fca : 2), (fcab : 1)\}$ | $\{(f : 3, c : 3, a : 3)\}\|m$ |
| b | $\{(fca : 1), (f : 1), (c : 1)\}$ | Empty |
| a | $\{(fc : 3)\}$ | $\{((f : 3, c : 3)\}\|a$ |
| c | $\{(f : 3)\}$ | $\{(f : 3)\}\|c$ |
| f | Empty | Empty |

**Cond. Pattern Base - Cond. $FP$ Tree**

- Präfixpfade eines Suffixes bilden die bedingte Basis.
- Diejenigen Präfixpfade, die häufiger als $support_{min}$ sind, bilden den bedingten $FP$ Tree.
- Falls mehrere dieser Präfixpfade zu einem Suffix gleich sind (vom Anfang bis zu einer bestimmten Stelle), wird daraus ein Pfad bis zu dieser Stelle und die ursprünglichen Häufigkeiten werden addiert.
- Ansonsten gibt es mehrere Pfade im bedingten Baum.

**Step 3: Recursively mine the conditional $FP$-tree**

| | Cond. pattern base of "$am$": $(fc:3)$ | Cond. pattern base of "$cm$": $(f:3)$ | Cond. pattern base of "$cam$": $(f:3)$ |
|---|---|---|---|
| $m$-conditional $FP$-tree: | $am$-conditional $FP$-tree: | $cm$-conditional $FP$-tree: | $cam$-conditional $FP$-tree: |
| $\{\}$ <br> \| <br> $f:3$ <br> \| <br> $c:3$ <br> \| <br> $a:3$ $\Longrightarrow$ | $\{\}$ <br> \| <br> $f:3$ <br> \| <br> $c:3$ | $\{\}$ <br> \| <br> $f:3$ | $\{\}$ <br> \| <br> $f:3$ |

**Single FP-tree Path Generation**

- Suppose an FP-tree $T$ has a single path $P$
- The complete set of frequent pattern of $T$ can be generated by enumeration of all the combinations of the sub-paths of $P$

$m$-conditional $FP$-tree:

$\{\}$ <br> \| <br> $f:3$ <br> \| <br> $c:3$ <br> \| <br> $a:3$

$\Longrightarrow$

All frequent patterns concerning $m$

- $m$
- $fm, cm, am$
- $fcm, fam, cam$
- $fcam$

**Cond. FP Tree - frequent sets**

- Alle Teilmuster im bedingten $FP$ Baum, der nur ein Zweig ist, und des Suffixes bilden die Menge häufiger Muster.
- Die gesuchte Menge der häufigen Mengen ist die Gesamtheit alles häufiger Muster aus allen bedingten $FP$ Bäumen.

**Principles of Frequent Pattern Growth**

- Pattern growth property
  - Let $\alpha$ be a frequent itemset in $DB$, $B$ be $\alpha$'s conditional pattern base, and $\beta$ be an itemset in $B$. Then $\alpha \cup \beta$ is a frequent itemset in $DB$ iff $\beta$ is frequent in $B$.
- "$abcdef$" is a frequent pattern, if and only if
  - "$abcde$" is a frequent pattern, and
  - "$f$" is frequent in the set of transactions containing "$abcde$"

**Algorithmus** $FP\_growth$
Input:

- $D$ eine Transaktionsdatenbank
- $support_{min}$ ein Schwellwert der Häufigkeit

1. Scan von $D$, Erstellen der Menge $F$ häufiger items und ihrer Häufigkeiten, Ordnen von $F$ in absteigender Häufigkeit.
2. Wurzel des $FP$ Trees ist Null. Für jede Transaktion Trans in $D$:
   nach Häufigkeit gemäß$F$ geordnete items in Trans werden zur Liste $[p|P]$, wobei $p$ das erste item und $P$ die restlichen sind. $insert\_tree([p|P], T)$
3. $FP\_growth(FP\_tree, null)$

$insert\_tree([p|P], T)$

- Wenn $T$ ein Kind $N$ hat mit $N.item\_name = p.item\_name$ dann erhöhe Häufigkeit von $N + 1$.
- Sonst bilde neuen Knoten $N$ mit Häufigkeit $= 1$ direkt unter $T$ und füge Knotenverweise zu den Knoten mit dem selben $item.name$ ein.
- Solange $P$ nicht $\{\}$ ist, $insert\_tree(P, N)$.

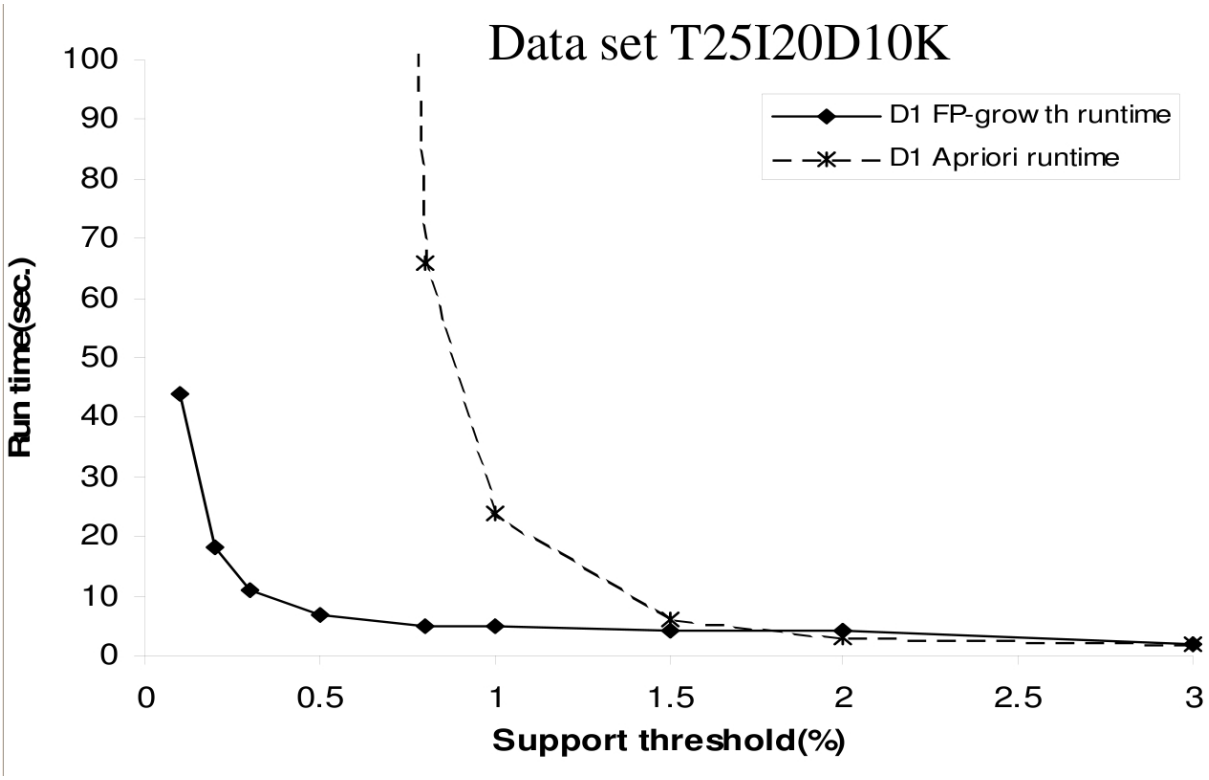
$fp\_growth(Tree, \alpha)$

- Wenn Tree ein einziger Pfad $P$ ist,
  - dann generiere für jede Kombination $\beta$ von Knoten in $P$ Muster $\beta \cup \alpha$ mit $support = support_{min}$ eines items in $\beta$.
- Sonst für jedes $a_i$ in header von Tree
  - generiere Muster $\beta = a_i \cup \alpha$ mit $s = a_i.s$
  - konstruiere $\beta$ cond. base und daraus $\beta$ cond. $FP$ tree $Tree_\beta$
  - Wenn $Tree_\beta$ nicht $\{\}$, dann $fp\_growth(Tree_\beta, \beta)$

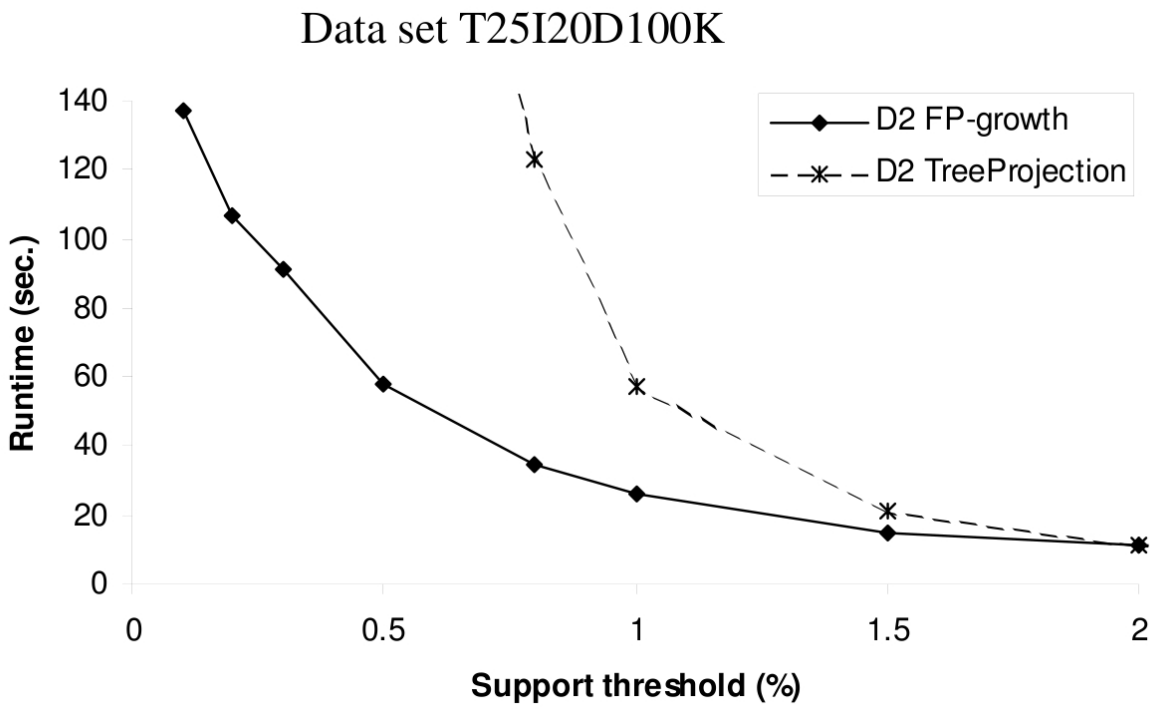
**Why Is Frequent Pattern Growth Fast?**

- Our performance study shows
  - $FP$-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection
- Reasoning
  - No candidate generation, no candidate test
  - Use compact data structure
  - Eliminate repeated database scan
  - Basic operation is counting and $FP$-tree building

***FP*-growth vs. Apriori: Scalability With the Support Threshold**




***FP*-growth vs. Apriori: Scalability With the Support Threshold**




**Was wissen wir jetzt?**

- $FP$-growth als Alternative zu Apriori
  - Schneller, weil keine Kandidaten generiert werden
  - Kompaktes Speichern
  - Basisoperation ist einfach Zählen.
- Der $FP$-Baum gibt Präfixbäume für ein Suffix an.
- Die Ordnungsrelation ist die Häufigkeit der items.
  - Der Baum wird vom häufigsten zum seltensten gebaut.
  - Die bedingte Basis wird vom seltensten Suffix zum häufigsten erstellt.

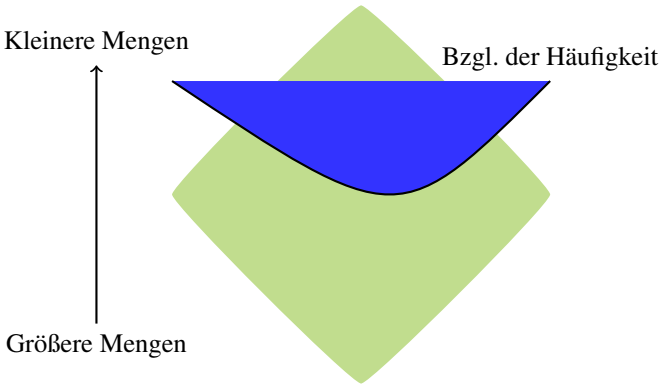
## 11.9 Closed Sets

**Zu viele Muster!**

- Es werden sehr viele häufige Mengen gefunden, die redundant sind.
- Wir müssen also aus den Beispielen
  - eine untere Grenze und
  - eine obere Grenze konstruieren.

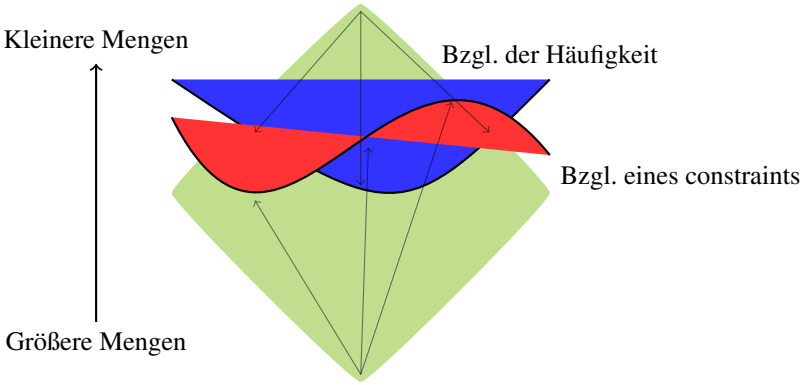
- Eine Halbordnung bzgl. Teilmengenbeziehung haben wir schon.
- Die Grenzen haben wir auch.
- Gemerkt?

### Untere Grenze




- Wenn eine Menge häufig ist, so auch all ihre Teilmengen. (Anti-Monotonie)
- Beschneiden der Ausgangsmengen für die Kandidatengenerierung gemäß dieser Grenze!

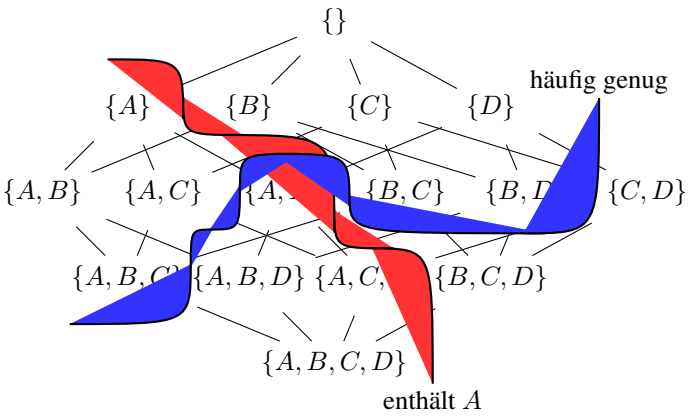
### Obere Grenze




- Monotonie der Seltenheit: Wenn eine Teilmenge selten ist, so auch jede Menge, die sie enthält. Seltenheit ist ein constraint.
- Beschneidung der Kandidatengenerierung nach der Monotonie.

### Beispiel mit Frequency threshold 0.3

| A | B | C | D |
|---|---|---|---|
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |




Dank an Jean-Francois Boulicaut!

**Kondensierte Repräsentationen**

- Statt Suche nach allen häufigen Mengen: Suche nach einer kondensierten Repräsentation,
  - die kleiner ist als die ursprüngliche Repräsentation und
  - aus der wir alle häufigen Mengen und ihre Häufigkeit ableiten können, ohne noch mal die Daten selbst anzusehen.
- Kondensierte Repräsentationen für Assoziationsregeln:
  - Closed item sets
  - Free sets
- Operator, der die Menge aller Assoziationsregeln ableitet:
  - Cover operator

**Closed Item Sets**

| A | B | C | D |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 |

- $closure(S)$ ist die maximale Obermenge (gemäß der Teilmengenbeziehung) von $S$, die noch genauso häufig wie $S$ vorkommt.
- $S$ ist ein closed item set, wenn $closure(S) = S$
- $support(S) = support(closure(S))$ (für alle $S$)
- Bei einem Schwellwert von $0.1$ sind alle Transaktionen häufig genug.
- Closed sind: $C, AC, BC, ABC, ABCD$
  - keine Obermenge von $C$ kommt auch 6 mal vor
  - $A$ kommt 5 mal vor, aber auch die Obermenge $AC$ und keine Obermenge von $AC$

**Kondensierte Repräsentation und Ableitung**

- Closed item sets sind eine kondensierte Repräsentation:
  - Sie sind kompakt.
  - Wenn man die häufigen closed item sets $C$ berechnet hat, braucht man nicht mehr auf die Daten zuzugreifen und kann doch alle häufigen Mengen berechnen.
- Ableitung:
  - Für jede Menge $S$ prüfen wir anhand von $C$: Ist $S$ in einem Element $X$ von $C$ enthalten?

    Nein , dann ist $S$ nicht häufig.

    Ja , dann ist die Häufigkeit von $S$ genau die der kleinsten solchen Obermenge $X$.

    $*$ Wenn es in mehreren Elementen von $C$ vorkommt, nimm die maximale Häufigkeit!

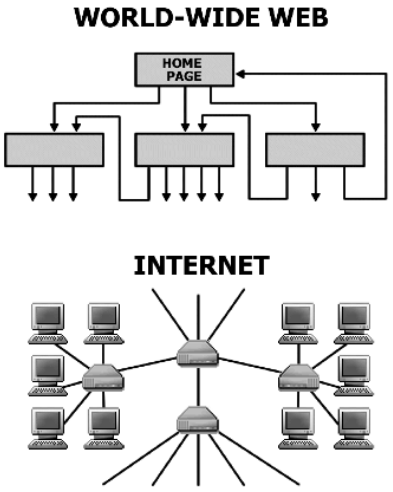
**Was wissen Sie jetzt?**

- Sie kennen eine Repräsentation, die weniger Elemente als häufig ausgibt, aus der aber alle häufigen Mengen hergeleitet werden können.
- Es gibt noch viele andere Methoden, um nur interessante Muster auszugeben, aber hier lernen Sie nur eine kennen.

## 11.10 Web Mining

**Das Web und das Internet als Graph**

Webseiten sind Knoten, verbunden durch Verweise. Router und andere Rechner sind Knoten, physikalisch verbunden.




Reka Albert, Albert-Laszlo Barabasi: Statistical Mechanics of Complex Networks, arXiv, 2006

**Eigenschaften des World Wide Web (WWW)**

Die Struktur des Webs wurde schon früh untersucht (http://arxiv.org/pdf/cond-mat/0106096v1.pdf)

- *Small Worlds:* Der Pfad zwischen zwei Knoten hat nur wenige Knoten. In einem Ausschnitt des WWW mit 200 Mio. Seiten fanden Broder at al (2000) durchschnittliche Pfadlängen von 16 Knoten.

**Eigenschaften des World Wide Web (WWW)**

- *Clustering Coefficient C:* Bei den meisten Knoten $i$ sind nicht alle ihre direkten Nachfolger $k_i$ miteinander verbunden, sondern es gibt nur $E_i$ Kanten zwischen ihnen. Wenn die Nachfolger streng zusammenhängend wären, gäbe es $k_i(k_i-1)/2$ Kanten zwischen ihnen.

$$C_i = \frac{2E_i}{k_i(k_i-1)}$$

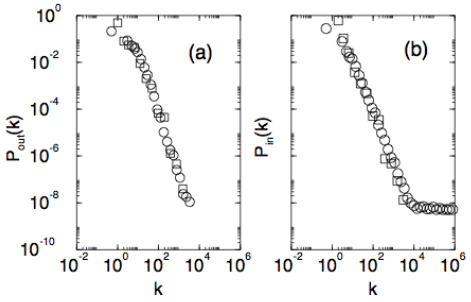
  Auf das WWW angewandt ignoriert man die Richtung der Kanten. Adamic (1999) fand bei 153.127 Web-Sites C=0,1078, bei einem Zufallsgraphen der selben Größe nur C'=0,00023.

**Eigenschaften des World Wide Web (WWW)**

- *Exponenzialverteilung:* Die Wahrscheinlichkeit, dass ein Knoten $k$ Kanten hat, folgt einer Exponenzialverteilung: $P(X = k) \sim k^{-\gamma}$

  Die bedingte Wahrscheinlichkeit hängt dann nicht von der Größe ab (*scale-free*): $P(X \geq k|X \geq m) = P(X \geq k)$

Mit hoher Wahrscheinlichkeit gehen nur wenige Kanten ab, kommen nur wenige Kanten an.




Verteilungen der Anzahl von Kanten: (a) ausgehende (b) eingehende.

**Web Mining**

Das WWW hat zu einer Menge interessanter Forschungsaufgaben geführt. Unter anderem gibt es:

- Indexieren von Web-Seiten für die Suche – *machen wir hier nicht*
- Analysieren von Klick-Strömen – web usage mining *kommt später*
- Co-Citation networks – *machen wir hier nicht*
- Finden häufiger Muster in vernetzten Informationsquellen
- Ranking von Web-Seiten nach Autorität

**Finden häufiger Muster in Graphen**

- Wir betrachten Web-Sites als Graphen.
  - Jede Web-Seite ist ein Knoten und die Knoten sind verbunden: das Klicken auf einer Seite führt zu einer anderen Seite.
  - Alternativ: die HTML-Struktur wird als Graph aufgefasst.
- Die Beobachtungsmenge beinhaltet viele Graphen und Muster sollen in vielen davon vorkommen.
- Alternativ: Ein Muster soll häufig in einem Graphen vorkommen.

### 11.10.1 Finden von häufigen Subgraphen

**Apriori-artiges Finden von Mustern**

- Wir betrachten Web-Sites als Graphen. Jede Web-Seite ist ein Knoten und die Knoten sind verbunden: das Klicken auf einer Seite führt zu einer anderen Seite.
- Die Beobachtungsmenge beinhaltet viele Graphen und die Muster sollen in vielen davon vorkommen.
- Analog zu Apriori werden häufige Subgraphen erweitert zur Kandidatengenerierung.
- Die Erweiterung ist jetzt etwas komplizierter.
- AGM-Algorithmus: A. Inokuchi, T. Washio, H. Motoda: An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data. PKDD Conference 2000.

**Beispiel für Subgraphen**

Die beiden linken 5-großen Graphen haben einen 4-großen Graphen gemeinsam.
Die beiden rechten 6-großen Graphen sind die Kandidaten.

Hong Cheng, Xifeng Yan, Jiawei Han: Mining Gaph Patterns. In: Charu Agharwal, Haixun Wang (eds): Managing and Mining Graph Data 2010

**Apriori-artiges Finden von Mustern: Notation**

- $V(g)$ sind die Knoten eines Graphen $g$;
- $E(g)$ sind die Kanten eines Graphen $g$;
- Annotation $l$ bildet ein Label auf einen Knoten oder eine Kante ab.

**Subgraph Isomorphie**

Für zwei annotierte Graphen $g, g'$ is die *Subgraph Isomorphie* die Einbettung $f : V(g) \rightarrow V(g')$, so dass

- $\forall v \in V(g) : l(v) = l'(f(v))$
- $\forall (u, v) \in E(g) : (f(u), f(v)) \in E(g') und l(u, v) = l'(f(u), f(v))$

wobei $l, l'$ die Annotationen von $g, g'$ sind.

**Häufige Graphen und Anti-Monotonie**

Wir können nun die Lernaufgabe definieren als das Finden aller häufiger Graphen.

**Häufiger Graph**

- Gegeben eine Menge annotierter Graphen $D = G_1, ..., G_n$ und ein Subgraph $g$, dann ist $support(g) = \frac{|D_g|}{|D|}$ und die Menge $D_g = \{G_i | g \subseteq G_i, G_i \in D\}$.
- Finde alle Graphen $g$, deren $support(g)$ nicht kleiner ist als $minsup$.

**Anti-Monotonie**

Eine Subgraph der mit $k$ Knoten bzw. Kanten ist nur dann häufig, wenn alle seine Subgraphen häufig sind.

$Apriori(D, minsup, S_k)$

**Input:** Graphen $D$, Schwellwert $minsup$, $k$ große Subgraphen $S_k$

**Output:** Die Menge aller $k + 1$-großen häufigen Subgraphen $S_{k+1}$

$$
\begin{array}{rcl}
S_{k+1} & := & \{\} \\
for & g_i \in S_k & do \\
 & for & g_j \in S_k do \\
 & for & g = g_i \odot g_j do \\
 & if & support(g) \geq minsup, g \notin S_{k+1} \\
 & then & S_{k+1} := S_{k+1} \cup g \\
if & S_{k+1} \neq \{\} & then \\
call & Apriori & (D, minsup, S_{k+1}) \\
return & &
\end{array}
$$

**Closed Subgraphs**

Analog zu den Closed Sets gibt es auch bei den Graphen eine Closed Subgraph Darstellung.

**Closed subgraph**

A subgraph $g$ is a *closed subgraph* in a graph set $D$, if

- $g$ is frequent in $D$ and
- there exists no proper supergraph $g'$ such that $g \subset g'$ and $g'$ is frequent in $D$.

**Was wissen Sie jetzt?**

- Man kann eine Web-site als Graph auffassen, bei dem die Seiten (Knoten) miteinander verbunden sind.
- Auch bei einer Menge von Graphen kann man häufige Muster (Teilgraphen) finden. Sie kennen den Apriori-Algorithmus für Graphen, der ein Muster durch Hinzunahme eines Knotens erweitert.
- Auch bei häufigen Mustern in Graphen gibt es eine aggregierte Darstellung und Sie kennen die Definition.

### 11.10.2 Ranking von Web-Seiten nach Autorität

**Ranking von Web-Seiten**

Was sind besonders *wichtige* Seiten?

- Eine Seite, von der besonders viele Links ausgehen, heißt *expansiv*.
- Eine Seite, auf die besonders viele links zeigen, heißt *beliebt*.
- Wie oft würde ein zufälliger Besucher auf eine Seite $i$ kommen? Zufällige Besuche von einer beliebigen Startseite aus:
  - Mit der Wahrscheinlichkeit $\alpha$ folgt man einer Kante der aktuellen Seite (Übergangswahrscheinlichkeit).

– Mit der Wahrscheinlichkeit $1 - \alpha$ springt man auf eine zufällige Seite, unter der Annahme, dass die Seiten gleich verteilt sind (Sprungwahrscheinlichkeit).

Der Rang einer Seite $PageRank(i)$ ist der Anteil von $i$ an den besuchten Knoten.

**Zufalls-Surfermodell: PageRank**

Matrix $M_{ij}$ für Kanten von Knoten $j$ zu Knoten $i$; $n(j)$ ist die Anzahl der von $j$ ausgehenden Kanten; $N$ Knoten insgesamt.

$$\begin{pmatrix} & 1 & \dots & N \\ 1 & 0 & \dots & M_{1N} \\ \vdots & \dots & M_{ij} = 1/n(j) & \dots \\ N & \dots & \dots & 0 \end{pmatrix}$$

Matrix $N \times N$ mit den Einträgen $1/N$ gibt die Gleichverteilung der Knoten an (Sprungwahrscheinlichkeit).

Die Wahrscheinlichkeit, die Seite zu besuchen, ist die Summe von Sprung- und Übergangswahrscheinlichkeit, angegeben in $N \times N$ Matrix $M'$:

$$M' = (1 - \alpha) \left[ \frac{1}{N} \right] + \alpha M \tag{11.3}$$

**PageRank**

Eigenvektoren von $M'$ geben den Rang der Knoten an.

Man kann das Gleichungssystem für $\alpha < 1$ lösen:

$$Rang_i = (1 - \alpha) \left[ \frac{1}{N} \right] + \alpha \sum_j M^{-1} ij$$

PageRank ist der rekursive Algorithmus:

$$Rang_i = \frac{1 - \alpha}{N} + \alpha \sum_{\forall j \in \{(i,j)\}} \frac{Rang_j}{n(j)} \tag{11.4}$$

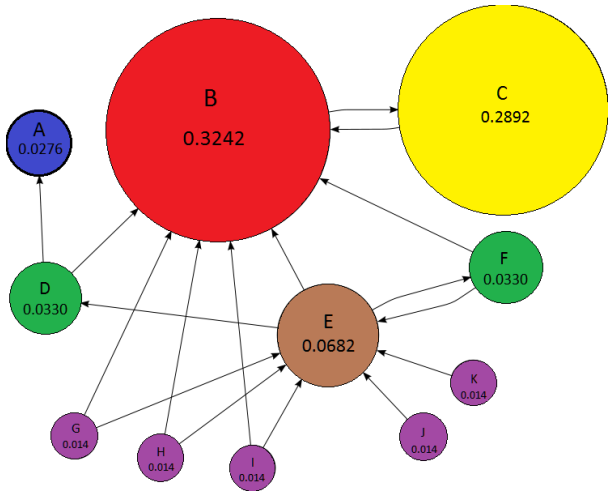
**PageRank Beispiel**

Mit $\alpha = 0,85$ hier ein kleines Beispiel (wikipedia). Die Größe der Kreise entspricht der Wahrscheinlichkeit, mit der ein Surfer auf die Seite kommt. Seite C wird nur von einer einzigen, aber gewichtigeren Seite verlinkt und hat also einen höheren PageRank als Seite E, obwohl E von sechs Seiten verlinkt wird.

**Alternativ: HIT – Hyperlinked-Induced Topic search**

Ähnlich ist eine Anfrage-orientierte Bewertung von Web-Seiten. Statt des Zufalls-Surfers wird eine Suchanfrage gestellt und nur die Menge $G$ der gelieferten Seiten mit allen Seiten, die auf diese verbunden sind, bewertet. Idee:

- Eine expansive Seite ist ein *hub* $h$. Sie soll auf beliebte Seiten zeigen. Die Expansionsbewertung aller Seiten ist ein Vektor $\vec{h}$.



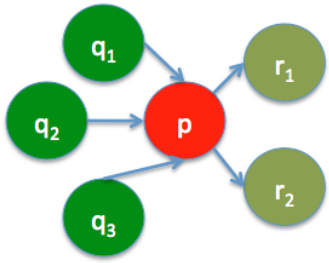

- Eine beliebte Seite ist eine *authority* $a$. Auf sie soll von beliebten Seiten aus verwiesen werden. Die Beliebtheitsbewertung aller Seiten ist ein Vektor $\vec{a}$.

Die Matrix der Kanten $M$ hat $M_{ij} = 1$, falls es eine Kante zwischen $I$ und $j$ gibt, 0 sonst.

$$\vec{h} = M\vec{a} \quad \vec{a} = M^T\vec{h} \tag{11.5}$$

Jon Kleinberg: Authorative sources in a hyperlinked environment. Journal of the ACM 46(5), 1999.

**Berechnung von $\vec{h}, \vec{a}$**




$k$ Iterationen

- Berechne $\vec{a}$ für alle Knoten $p \in G$:
  - $a_p := \sum h_q$;
  - $norm := \sqrt{\sum_{p \in G} a_p^2}$;
  - $a_p := a_p / norm$;
- Berechne $\vec{h}$ für alle Knoten $p \in G$:
  - $h_p := \sum a_r$;
  - $norm := \sqrt{\sum_{p \in G} h_p^2}$;
  - $h_p := h_p / norm$;

**HIT**

- HIT bewertet die Seiten, die eine Suchmaschine geliefert hat, nach Beliebtheit $h$ und Autorität $a$.
- Alle Suchergebnisse werden nach der Bewertung geordnet.
- HIT liefert Seiten mit großem $h$ und Seiten mit großem $a$.
- Probleme:
  - Wenn eine Seite verschiedene Inhalte hat, kann sie auch ausgegeben werden, wenn sie kein guter *hub* für die Anfrage ist: viele $r$ zu unterschiedlichen Themen!
  - Wenn viele Seiten einer Web-Site auf Seite zeigen, bekommt diese hohe Autorität (topic hijacking), obwohl die $q_i$ nicht unabhängig voneinander waren.
- Die Summen $a_p, h_p$ sollten gewichtet werden. PageRank tut das.

**Was wissen Sie jetzt?**

- Sie kennen jetzt die Grundlage des Ranking von Web-Seiten und einige Probleme. Schreiben Sie den Kern von PageRank und HIT in Matrix-Notation (Gleichungen 1, 2, 3).
- PageRank schätzt die Wahrscheinlichkeit ab, auf die Seite zu kommen, indem es Kanten folgt und zufällig auf Knoten springt. Dabei verwendet es die Wahrscheinlichkeit $\alpha$ als Gewicht der Übergangswahrscheinlichkeiten und $1 - \alpha$ als Gewicht der Sprungwahrscheinlichkeit.
- HIT summiert die Beliebtheit der abgehenden Seiten als Wert der Expansion (*hub*). HIT summiert die Expansion der eingehenden Seiten als Wert der Beliebtheit (*authority*).

**Häufige Mengen**

Grundalgorithmen

- Apriori
- FP Growth

Verbesserungen

- Kondensierte Repräsentationen
- Pushing Constraints into the algorithm
- Bessere Sifnigikanztests

**Häufige Mengen**
Erweiterungen zur Zeit

- Episodenlernen mit WINEPI
- Zeitreihen lernen nach Das
- Zeitintervallbeziehungen lernen nach Hoeppner

Privacy preserving data mining

- $K$-anonymity (pushing constraints into FSM)

Clustering anhand häufiger Mengen

$\rightarrow$ Ansatz von Ester

- Ansatz von Strehl und Ghosh
- ...

**Clustering von Dokumenten**

**Terme** $T$ Menge aller Wörter, die in irgend einem Dokument einer Sammlung $D$ vorkommen

$F = \{F_1, \ldots, F_k\}$ Alle Termmengen $F_i \subseteq T$, die häufiger als $smin$ in der Sammlung vorkommen.

**Dokument** $D_i$ Menge der Terme, die in diesem Dokument vorkommen ($D_i \subseteq T$)

**Abdeckung** Ein Dokument $D_i$ wird von einer häufigen Menge $F_i$ *abgedeckt*, wenn alle Terme in $F_i$ auch in dem Dokument vorkommen.

$$\mathsf{Cov}(F_i) = \{D_i \in D \mid F_i \subseteq D_i\}$$

**Cluster** Menge aller abgedeckten Dokumente $\mathsf{Cov}(F_i)$ und die häufigen Terme $F_i$ sind seine Beschreibung.

**Aufgabe** Finde Menge von clusters, die sich möglichst wenig überlappt.

**Frequent Term-Based Clustering**

**Gegeben** Datenbank von Dokumenten $D$

Berechnen aller häufigen Mengen mit Apriori oder Fpgrowth. Bei jeder häufigen Menge ist angegeben, welche Dokumente sie abdeckt (i.e. support ist hier cov).

**Aufgabe** Wähle diejenigen häufigen Mengen, deren cover sich möglichst wenig überdeckt.

Vergleiche [1, 7].

**Overlap**

$f$: Anzahl der Termmengen, die ein Dokument unterstützt.

Entropy Overlap eines clusters $C$:

$$EO(C) = \sum_{D_i \in C} \frac{-1}{f_i \ln(\frac{1}{f_i})}$$

Wenn alle Dokumente von $C$ keine andere Termmenge unterstützen, ist

$$EO(C) = 0, f_i = 1.$$

$D_1$

{sun}, {fun}, {surf},
{sun,fun}, {fun,surf}, $f_1 = 6$
{sun,fun,surf}

| {sun} | {beach} |
|---|---|
| $D_1, D_4, D_5, D_6$ | $D_7, D_{14}$ |
| $D_2, D_9, D_{13}$ | $D_2, D_9, D_{13}, D_8$ |
| $D_8, D_{10}, D_{11}, D_{15}$ | $D_{10}, D_{11}, D_{15}$ |

{sun, fun, beach}

$D_8, D_{10}, D_{11}, D_{15}$

**Algorithmus FTC**

**Gegeben:** $F$ häufige Mengen,
$n = |D|$ Anzahl Dokumente, select= {}
**while** $|\mathsf{cov}(\mathsf{select})| \neq n$ **do**
**for** each $C_i \in F$ calculate **do**
$EO(C_i)$
best := $C_i$ with minimal $EO(C_i)$
select := select $\cup$ best
$F := F \setminus \mathsf{best}$
$D := D \setminus \mathsf{cov}(\mathsf{best})$

**Ausgabe:** select, cov($C_i$), $C_i \in$ select.

Eine Iteration:
<all> $F$: {sun}, {fun}, {surf}, {beach}
<all> {sun, fun}, {fun, surf}, {sun, beach}
<all> {sun, fun, surf}, {sun, fun, beach}
<all> $n = 16$

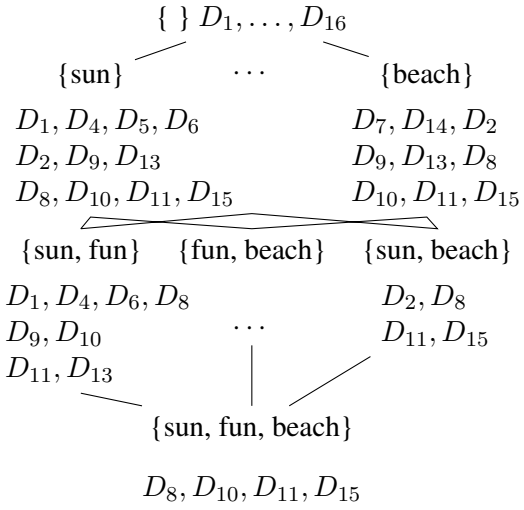
<all>

| | |
|---|---|
| $EO(\{\text{sun}\})$ | $= 2,98$ |
| $EO(\{\text{fun}\})$ | $= 3,0$ |
| $EO(\{\text{beach}\})$ | $= 2,85\ldots$ |
| $EO(\{\text{sun, fun}\})$ | $= 1,97$ |
| $EO(\{\text{sun, beach}\})$ | $= 1,72\ldots$ |
| $EO(\{\text{sun, fun, beach}\})$ | $= 0,9$ |

select := {sun, fun, beach}
<all> $F$: {sun}, {fun}, {surf}, {beach}
<all> {sun, fun}, {fun, surf}, {sun, beach}
<all> {sun, fun, surf}, {sun, fun, beach}

**Hierarchisches FTC**

- FTC-Algorithmus nimmt nicht $F$, sondern nur häufige Mengen der Länge $k$ als Eingabe
- Ein Cluster mit $F_i$ der Länge $k$ wird weiter aufgeteilt in cluster der Länge $k+1$, die $F_i$ enthalten.




**Anwendung für Tagging-Systeme**

- Web 2.0: tagging
- Verbesserung von HFTS durch explizite Kriterien
  - Vollständigkeit
  - Überlappungsfreiheit

  als multikriterielle Optimierung (Wurst/Kaspari in KDML 2007)
- Verbessertes HFTS für die Strukturierung von Tagsets

**Tagging**

- Del.icio.us tagging von Web-Seiten
- Bibsonomy tagging von Web-Seiten und Literatur
- Last.FM tagging von Musik
- FLICKR tagging von Bildern
- YouTube tagging von Videos
- Yahoo! PODCASTS tagging von Podcasts
- TECHNORATI tagging von eigenen Blogs
- Upcoming tagging von Veranstaltungen

**Folksonomy**

Eine *Folksonomy* ist ein Tupel $(U, T, R, Y)$ wobei

- $U$: Benutzer
- $T$: tags
- $R$: Ressourcen
- $Y \subseteq U \times T \times R$ und ein Tupel $(u, t, r) \in YY$ heißt, dass Benutzer $u$ der Ressource $r$ den tag $t$ zugewiesen hat.

| $U$ | $T$ | $R$ |
|---|---|---|
| User1 | tag1 | Res2 |
| User1 | tag2 | Res3 |
| User2 | tag3 | Res3 |
| User4 | tag3 | Res3 |

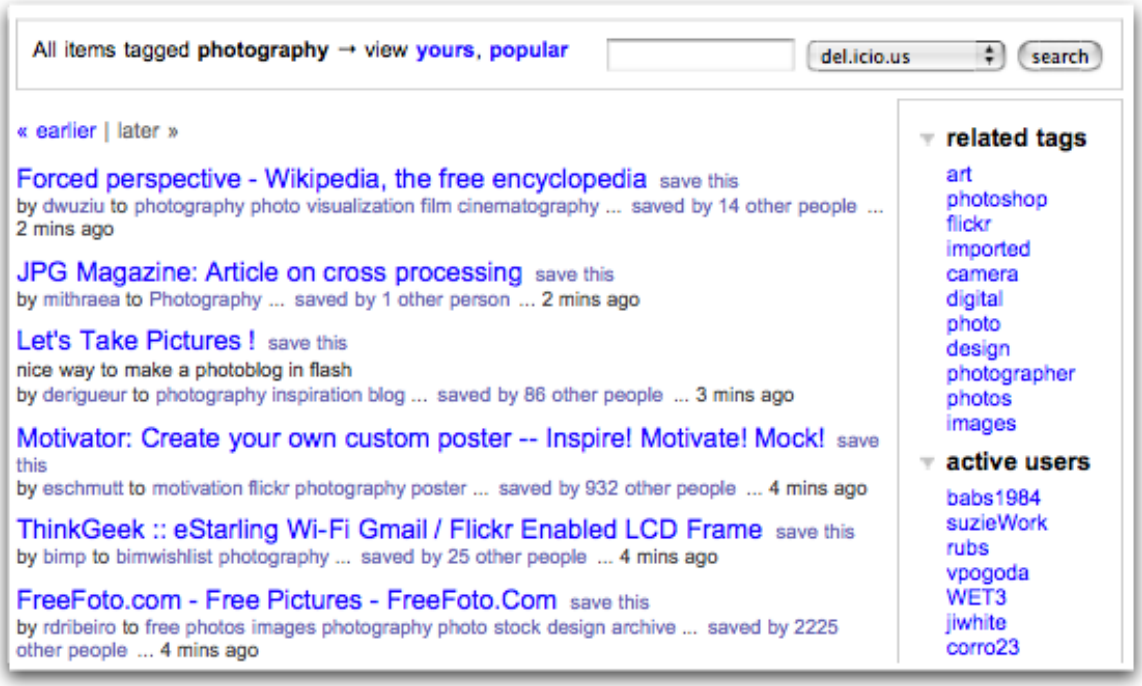
**Häufigkeit berücksichtigen**

.net advertising ajax apple architecture art article audio blog blogging blogs book books business comics community computer cooking cool css culture database design development diy download education english environment fashion fic finance firefox flash flickr fonts food free freeware fun funny furniture gallery game games google graphics green hardware health history home howto humor illustration imported inspiration internet java javascript jobs language library lifehacks linux mac magazine management maps marketing media microsoft mobile movies mp3 music network networking news online opensource osx phone photo photography photos photoshop php plugin politics portfolio productivity programming psychology python radio rails recipes reference research resource resources rss ruby rubyonrails science search security seo sga shop shopping slash social software tech technology tips tool tools toread travel tutorial tutorials tv twitter ubuntu video videos web web2.0 webdesign webdev wedding wiki windows wordpress work writing youtube

**Reicht das?**

- Kein Überblick
- Keine Gruppierung verwandter Begriffe
- Keine Navigationsmöglichkeit

**Tag „photography“ Del.icio.us**



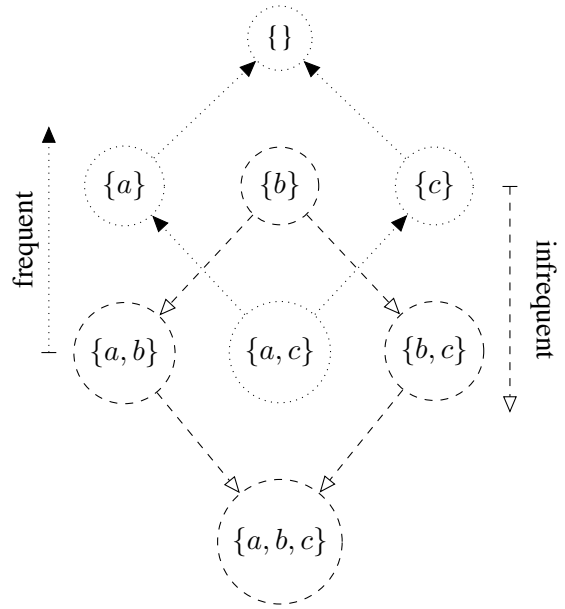

**Navigation**

Wie navigiert man in dieser Folksonomy?

- Von „photography“ kann man zu „art“ navigieren (related links).
- Man kann nicht von „photography“ zu „photography UND art“ navigieren!

Wie bietet man hierarchische Navigationsstrukturen an? Natürlich durch HTFC!

**Verband aller tagsets – häufige**




**Problem**

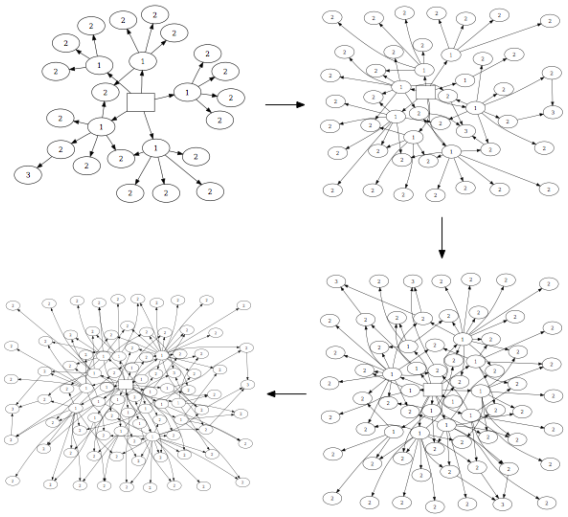
- Man erhält genau eine hierarchische Struktur häufiger tagsets.
- Verschiedene Benutzer möchten verschiedene Strukturen.
- Der Verband der häufigen tagsets wird nach verschiedenen Kriterien reduziert zum gewünschten clustering:
  - Vollständige Abdeckung von $R$ (completeness)
  - Überschneidungsfreie cluster (overlap)
  - Ressourcen nicht nur im Wurzelknoten, sondern gut verteilt (coverage)
  - Niedrige Anzahl von Nachfolgern an jedem cluster (childcount)

**Was ist schön?**

Pareto-optimale Lösungen für die Optimierung nach

- Anzahl Kinder vs.
- Vollständigkeit

Alle Lösungen werden für ein multi-kriterielles Problem in einem Lauf gewonnen.



**Multi-kriterielle Optimierung**

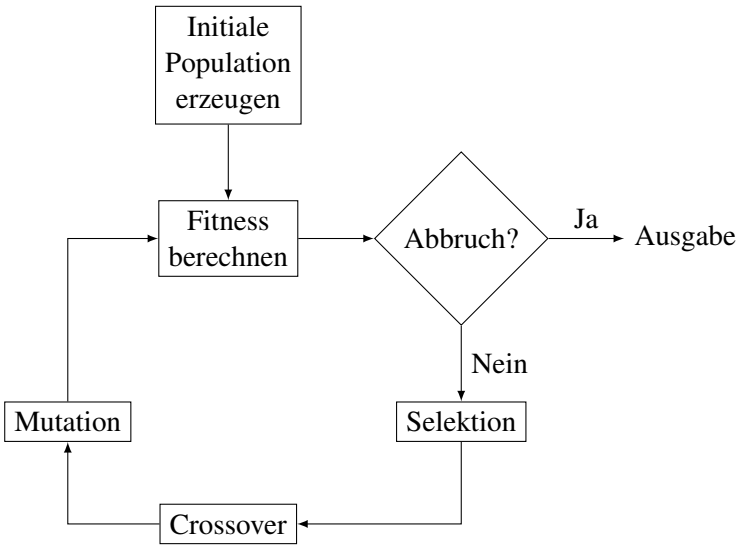
Gegensätzliche Gütekriterien

- Nur empirisch feststellbar
- Hier:
  - Vollständigkeit vs. Anzahl Kinder und
  - Verteiltheit der Ressourcen vs. Überschneidungsfreiheit

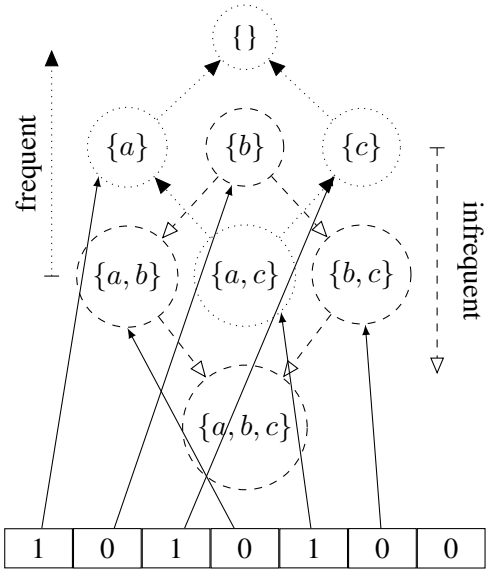
**Problem** Auswahl aus der Menge aller möglichen Teilmengen des FTS clusterings gemäß zweier gegensätzlicher Kriterien

**Lösung** durch genetische Programmierung

**Genetische Programmierung**



**Individuen**

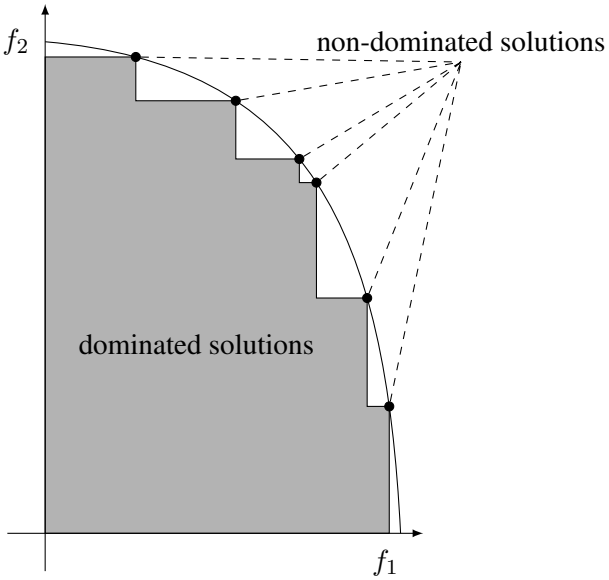



**Fitness Funktion**

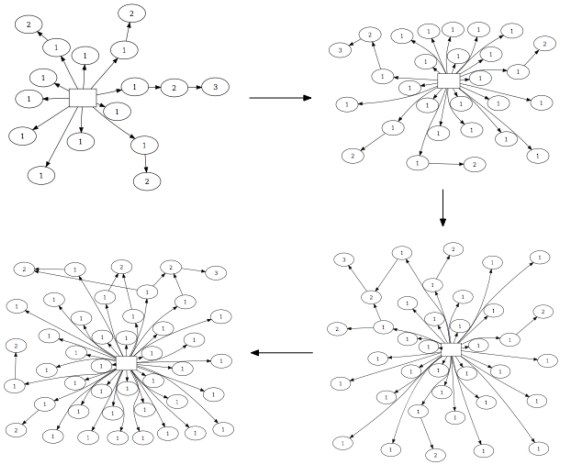
- Jedes Individuum $x$ wird bewertet $f : S \to \mathbb{R}^n$
- Vektor mit Gütemaßen für $n$ Kriterien $\mathbb{R}^n$ Hier: Zwei reelle Werte, für jedes Kriterium eines.
- Pareto-Dominanz: Ein Vektor $\mathbf{u}$ dominiert einen Vektor $\mathbf{v}$, wenn für alle $i$ gilt: $u_i \geq v_i$ und es gibt ein $i$, so dass $u_i > v_i$.
- Pareto-optimale Menge: für ein multikriterielles Optimierungsproblem besteht die Pareto-optimale Menge aus allen nicht-dominierten Vektoren:

$$Q = \{x \in S \mid \text{es gibt kein } y \in S \text{ mit } f(y) > f(x)\}$$

**Paretofront**




**Entlang der Paretofront**
Overlap vs. Coverage



**Was wissen Sie jetzt?**

- Man kann aus häufigen Mengen Clusterings gewinnen.
- Clusterings sind Teilmengen der Menge häufiger Mengen.
- Es gibt sehr viele Kriterien, nach denen man die Auswahl der Teilmengen steuern kann.
- Bei vielen Ansätzen sind die Kriterien fest in den Algorithmus eingebaut, z.B. durch Löschen schon einsortierter Ressourcen. Man erhält dann ein Clustering.
- Man kann aber auch gegensätzliche Kriterien mit GA optimieren und erhält verschiedene gleich gute Lösungen.

# Literaturverzeichnis


[1] Florian Beil, Martin Ester, and Xiaowei Xu. Frequent term-based text clustering. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 436–442. ACM, 2002.

[2] J. Bliss, R. Saljo, and P. Light, editors. *Learning Sites – Social and technological Resources for Learning*. Pergamon Press, 1999.

[3] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[4] P. Dillenbourg, editor. *Collaborative Learning – Cognitive and Computational Approaches*. Pergamon Press, 1998.

[5] Jason Eisner. An interactive spreadsheet for teaching the forward-backward algorithm. In Dragomir Radev and Chris Brew, editors, *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL*, pages 10–18, 2002.

[6] Gerald Farin and Dianne Hansford. *Lineare Algebra – ein geometrischer Zugang*. Springer, 2003.

[7] Benjamin CM Fung, Ke Wang, and Martin Ester. Hierarchical document clustering using frequent itemsets. In *Proceedings of SIAM international conference on data mining*, pages 59–70, 2003.

[8] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, New York, USA, 2001.

[9] Boris Mirkin. Reinterpreting the category utility function. *Machine Learning*, 45(2):219–228, 2001.

[10] Katharina Morik and Martin Mühlenbrock. In order to learn – how the sequence of topics influences learning. pages 119–136. Oxford University Press, 2007.

[11] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[12] Robert S. Siegler. *Childrens's Thinking*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 1991.

[13] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, 2003.

[14] Gerald Teschl and Susanne Teschl. *Mathematik für Informatiker*. Springer, 2006.

[15] Alexander Topchy, Anil K Jain, and William Punch. Combining multiple weak clusterings. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 331–338. IEEE, 2003.

# Index