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

Skript zur Vorlesung

Knowledge Discovery in Databases II

im Wintersemester 2012/2013

Lecture 2: Feature Selection und Feature Reduktion

Lectures: Dr Matthias Schubert, Dr. Eirini Ntoutsi
Tutorials: Erich Schubert

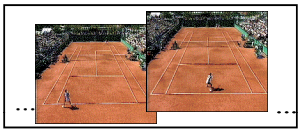
http://www.dbs.ifi.lmu.de/cms/Knowledge_Discovery_in_Databases_II_(KDD_II)

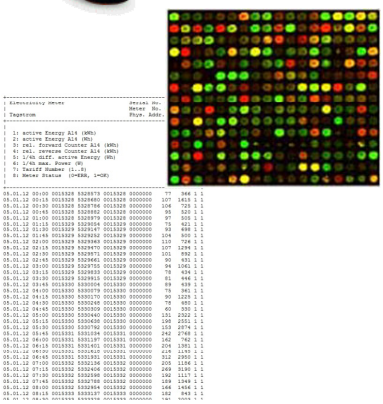
2. Featurereduktion und Featureselektion

Inhalt dieses Kapitels

Beispiele für hochdimensionale Objektdarstellungen

- Bilddaten
  - Bilddeskriptoren (Farbverteilungen, Pyramid of Gradients, Texturen,..)
  - Aufteilung der Bilder in Bereiche
  - Je nach Typ ca. 16 - 1000 Features
- Metabolomdaten
  - Feature entspricht Konzentration von Stoffwechselprodukt im Blut
  - Je nach Aufbau 50-2000 Features
- Mikro-Arrays
  - Features entsprechen z.B. Genen
  - Je nach Versuchsaufbau bis zu 20000 Features
- Texte
  - Features entsprechen Worten oder Termen
  - Je nach Anwendung 10000-20000 Features



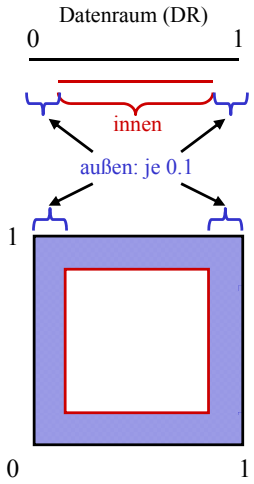


„Fluch der Dimensionalität“ (*Curse of Dimensionality*)

- Distanz zum nächsten Nachbarn unterscheidet sich im Hochdimensionalen kaum von der Distanz zu einem beliebigen Nachbarn

$$\frac{\text{nächsteNachbarnDist}}{\text{entfernteste NachbarnDist}} \approx 1$$

- Die Wahrscheinlichkeit, dass Datenobjekte am Rand des Datenraumes liegen, steigt mit der Anzahl der Dimensionen exponentiell



1D: $\mathcal{P}_{DR} = 1^1 = 1$

$\mathcal{P}_{innen} = 0.8^1 = 0.8$

$\mathcal{P}_{außen} = 1 - 0.8 = 0.2$

2D: $\mathcal{P}_{DR} = 1^2 = 1$

$\mathcal{P}_{innen} = 0.8^2 = 0.64$

$\mathcal{P}_{außen} = 1 - 0.64 = 0.36$

3D: $\mathcal{P}_{DR} = 1^3 = 1$

$\mathcal{P}_{innen} = 0.8^3 = 0.512$

$\mathcal{P}_{außen} = 1 - 0.512 = 0.488$

10D:

$\mathcal{P}_{außen} = 0.893$

Grundprobleme

andere Effekte des *Curse of Dimensionality*:

- Die Werte in jeder Dimension sind verrauscht.

  D.h. die Werte schwanken durch Störeinflüsse, die mit dem Objekt an sich nichts zu tun haben.

- Bei zunehmender Dimensionalität wird die Summe der Störeinflüsse so groß, dass die beobachteten Unterschiede zwischen 2 Objekten von den Störeinflüssen dominiert werden.

  => Summe der Unterschiede hängt von der Ausprägung der Störeinflüsse ab und nicht von den Objekteigenschaften.

  => Abstand zwischen Objekt gleicht sich immer weiter an, da Störeinflüsse gleichverteilt über alle Objekte.

Grundprobleme

- Patterns und Modelle auf hochdimensionalen Daten sind oft schwer interpretierbar.

  $\Rightarrow$ Lange Entscheidungsregeln

- Effizienz bei hochdimensionalen Daten häufig problematisch.

  $\Rightarrow$ Indexstrukturen degenerieren auf hochdimensionalen Daten

  $\Rightarrow$ Distanzberechnungen werden i.d.R. teurer

- Muster treten nur in Teilräumen auf, aber nicht im gesamten Featureraum

- Cliquen von korrelierten Features dominieren das Objektverhalten.

2.2. Feature-Selektion

Idee: Bei sehr vielen Features sind nicht alle notwendig, um die Daten zu beschreiben:

- Features sind nicht aussagekräftig für ein Problem
- Informationsgehalt stark korrelierter Features ist fast identisch

Die Einschränkung auf einen Teilraum des gesamten Datenraums kann Verfahren effizienter und effektiver machen.

Lösung:
Streiche alle „überflüssigen“ Dimensionen aus dem Feature-Raum.

Feature-Selektion

Gegeben: Vektorraum $F = D_1 \times .. \times D_n$ mit $D = \{D_1,..,D_n\}$.

Gesucht: Minimaler Unterraum M über $D' \subseteq D$, der für ein gegebenes Data Mining Problem eine optimale Lösung erlaubt.

=> Minimalität erhöht Effizienz und verringert den Curse-of-Dimensionality.

Probleme:

- Optimalität hängt unmittelbar mit dem anschließend verwendeten Data Mining Algorithmus zusammen.
- Problem ist sehr komplex, da es 2^d mögliche Unterräume gibt.
- Die meisten Qualitätmaße haben keine Monotonie-Eigenschaften.
  (Features können erst in Kombinatione mit anderen Features nützlich werden)
  => Suche läuft meist über Heuristiken, die für ein gegebenes Qualitätsmaß nicht den besten sondern nur einen guten Raum findet

Überblick über Feature Selektions Methoden

1. Forward Selektion und Feature Ranking
   Information Gain , χ^2-Statistik, Mutual Information
2. Backward Elimination und Zufällige Unterräume
   Nächste Nachbarn Kriterium, Modellbasierte Suche
3. Branch and Bound Suche mit Inkonsistenzmaß
4. Genetische Algorithmen zur Unterraum-Suche
5. Feature Clusterings für Unsupervized Probleme

Forward Selektion und Feature Ranking

Gegeben: ein Supervised Learning Task

- eine Zielvariable *C* soll vorhergesagt werden
- es gibt Trainingsobjekte für die Relation zwischen Feature-Raum und Klassenvariable

Vorgehen

- Berechne Güte $G(D_i, C)$ für jede Dimension $D' \in \{D_1,..,D_n\}$ und die Zielvariable *C*
- Sortiere Dimensionen $\{D_1,..,D_n\}$ nach Güte $G(D_i, C)$
- Wähle die k besten Dimensionen

Annahme:

Feature sind außer über die Zielvariabel nicht direkt korreliert
=> es reicht aus die Korrelation zwischen Feature und Zielvariable zu untersuchen

Statistische Gütekriterien für Features

Wie gut kann die Zielvariable bei gegebenen Feature-Wert vorhergesagt werden?

Bewertung der Features über statistische Maße:

- Basieren auf Verteilungen bzgl. der Klassen und der Feature-Werte
  Achtung bei reellen Features ist Schätzung einer Zufallsvariable nicht direkt durchführbar. Daher:
  - Split zur Umwandlung von reellen Features in diskrete Features
  - Annahme über Verteilungsfunktion (z.B. Normalverteilung,..)
- Wie stark korrelieren die Verteilungen von Klassen und Features?
- Wie stark unterscheidet sich Aufteilung bzgl. dieses Features von einer zufälligen Aufteilung ?

Information Gain

Idee: Bewerte wie gut jede Dimension die Klassen „unterscheidet“.
(vgl. Entscheidungsbäume)

Zerlege Trainingsmenge anhand Feature/Unterraum in Teilmengen
(Unterteilung: nach Werten oder Splitkriterien).

Die *Entropie* für eine Menge *T* von Trainingsobjekten ist definiert als

$$entropie(T) = -\sum_{i=1}^{k} p_i \cdot \log p_i$$ (p_i steht für Häufigkeit der Klasse *i* in *T*)

entropie(*T*) = 0, falls $p_i = 1$ für ein *i*
entropie(*T*) = 1 für *k* = 2 Klassen mit $p_i = 1/2$

$$\text{Informationsgewinn}(T, a) = entropie(T) - \sum_{j=1}^{m} \frac{|T(a)_j|}{|T|} \cdot entropie(T(a)_j)$$

mit $T(a)_j$ Teilmenge von *T*, für die Attribut a den Wert *j*-ten Wert annimmt.

Bei reell wertigen Attributen muss ein Split gefunden werden.

χ2-Statistik

Idee: Bewertet Unabhängigkeit einer Dimension von einer Klasse.
Aufteilung der Daten anhand der Splitwerte s oder anhand diskreter Attributwerte

$A = \left|\{o \mid x \le s \wedge Class(o) = C_j\}\right|$ Objekte in C_j mit Wert $x \le$ Splitwert

$B = \left|\bigcup_{l \ne j} \{o \mid x \le s \wedge Class(o) = C_l\}\right|$ Objekte anderer Klassen mit $x \le$ Splitwert.

$C = \left|\{o \mid x > s \wedge Class(o) = C_j\}\right|$ Objekte in C_j mit $x >$ Splitwert.

$D = \left|\bigcup_{l \ne j} \{o \mid x > s \wedge Class(o) = C_l\}\right|$ Objekte anderer Klassen, mit x > Splitwert

χ2-Statistik ist definiert durch: $\chi^2(t, C_j) = \frac{|DB|(AD - CB)^2}{(A+C)(B+D)(A+B)(C+D)}$

Je höher Maximum oder Durchschnitt über alle Klassen desto besser das Feature a:

$\chi^2_{\max}(a) = \max_{i=1}^{m}\{\chi^2(a, C_i)\}$ oder $\chi^2_{avg}(a) = \sum_{i=1}^{m} \Pr(C_i)\chi^2(a, C_i)$

Mutual Information (MI)

Maß für gegenseitige Abhängigkeit zweier Zufallsvariablen.

Hier: Vergleich der Abhängigkeit von der allgemeinen Klassenverteilung mit Verteilung der Feature Vektoren.

1. diskreter Fall.

$$I(X,Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

2. Kontinuierlicher Fall

$$I(X,Y) = \int_Y \int_X p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy$$

Häufigkeit der Wertekombination x,y

Term der die Abhängigkeit darstellt:
bei Unabhängigkeit p(x,y)= p(x)p(y)
=> log(1) = 0

Forward Selektion und Feature Ranking

Vorteile:

- Effizient da nur einzelne *d* Features statt $\binom{d}{k}$ Unterräumen untersucht werden
- Anschätzung der Verteilungen benötigt moderate Anzahl an Beispielen

Nachteil:

- Dimensionen werden einzeln betrachtet: Klasse und Dimensionswert müssen direkt korreliert sein. Erkennt keine nicht achsenparallelen Muster.

- Korrelierte Dimensionen: Auswahl von bedeutungsgleichen Dimensionen falls diese am stärksten mit der Klasse korreliert sind

Backward Elimination

Idee: Starte mit dem gesamten Feature-Raum und lasse unnütze Feature weg.

Vorgehen: Greedy Backward Elimination

1. Generiere alle Unterräume *R* des Feature-Raums *F*
2. Bewerte alle Unterräume *R* mit Gütekriterium für Räume *G(R)*
3. Wähle den besten Unterraum R*
4. Fall R* die Zieldimensionalität hat, wird Suche abgebrochen.
   ansonsten wird Algorithmus mit R* aufgerufen.

Anwendung:

- kann für Supervized oder unsupervized Verfahren verwendet werden (G(R) kann sich auch an strukturellen Eigenschaften orientieren und braucht keine Zielvariable)
- Heuristische Suche, die nur Sinn ergibt wenn G(R) nicht monoton ist
  => bei Monotonie kann exakt mit Branch and Bound gesucht werden.

Distanzbasierte Kriterien

Idee: Unterraum ist gut, wenn Abstand von Objekten innerhalb einer Klasse durchschnittlich kleiner ist, als zwischen Objekten unterschiedlicher Klassen.

Qualitätsmaß:

Für alle Objekte o ∈ DB berechne den nächsten Nachbarn in Klasse *C=Class(o)* $NN_C(o)$ und den kleinsten nächsten Nachbarn $NN_{K \neq C}(o)$ in einer der anderen Klassen .

Güte des Unterraums U: $$Q(U) = \frac{1}{|DB|} \cdot \sum_{o \in DB} \frac{NN^U_{K \neq C}(o)}{NN^U_C(o)}$$

Bemerkung: Kriterium ist nicht monoton.

=> Durch Streichen einer Dimension kann Qualtität besser oder auch schlechter werden.

Modellbasierter Ansatz

Idee: Teste den Unterraum direkt damit, wie gut er für das gegebene Data Mining Problem funktioniert.

Beispiel: Bewerte jeden Teilraum mit einem Naive Bayes Klassifikator und 10-facher Überkreuzvalidierung auf einer Stichprobe von 200 Instanzen.

Anwendung:

- Erfolg des Data Mining Algorithmus muss messbar sein.
  (z.B. Klassifikationsgenauigkeit)
- Laufzeit sollte nicht zu hoch sein
- Parameter Auswahl sollte keinen zu hohen Einfluss haben
- Testmenge sollte überschaubar sein

Backward Elemination

Vorteile:

- Betrachtet ganze Unterräume anstatt einzelner unabhängiger Dimensionen (mehrfache Abhängigkeiten der Zielvariable bleiben erhalten
- Redundante Features können also solche erkannt und entfernt werden.

Nachteile:

- Tests bzgl. der Qualität eines Unterraums sind meist wesentlich aufwendiger.
- Ansatz ist ebenfalls nur eine Heuristik da es keine Monotonie gibt.

Branch and Bound Suche

Gegeben: Klassifikationsproblem über *F*.

Ziel: Selektiere *k* Dimensionen

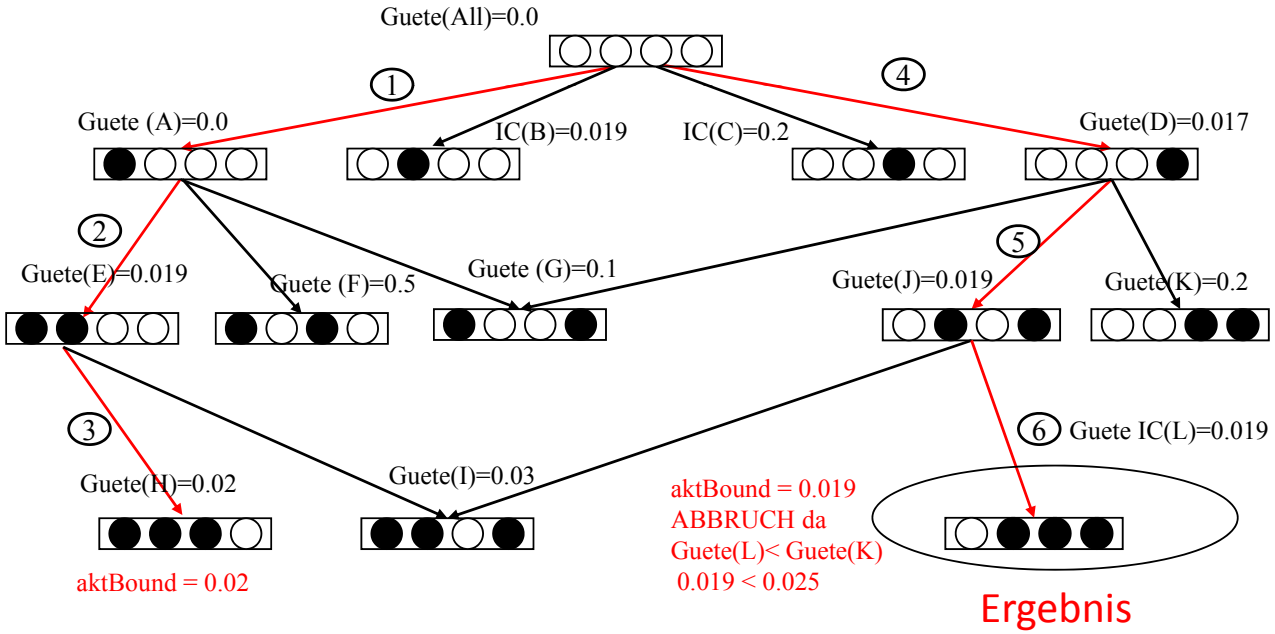
Backward-Elimination mit Branch and Bound:

```
FUNCTION BranchAndBound(Featurespace F, int k)
        queue.init(ASCENDING);
        queue.add(F,Guete(F))
        aktBound:= INFINITY;
        WHILE queue.NotEmpty() and aktBound > queue.top() DO
                aktURaum := queue.top();
                FOR ALL Unterräume U von aktURaum DO
                        IF U.dimensionality() = k THEN
                             IF Guete(U)< aktBound THEN
                                  aktBound  := Guete(U);
                                  BestURaum := U;
                        ELSE
                             queue.add( U, Guete(U));
        RETURN BestURaum
```

Beispiel Branch and Bound Suche

Beispiel: k = 1.

○ Feature selektiert ● Feature ausgeschlossen



Konsistenzbasierte Kriterien

Idee: Existieren im Unterraum *U* identische Vektoren *u* ,*v* mit $v_i = u_i$ $1 \le i \le d$ aber unterschiedlichen Klassen labels $C(u) \neq C(v)$.

=> Unterraum ist inkonsistent

Maß für die Konsistenz des Unterraums *U*:

- $X_U(A)$: Anzahl aller zu *A* identischen Vektoren in *U*
- $X^c_U(A)$: Anzahl aller zu *A* identischen Vektoren in *U die zur Klasse c gehören*
- $IC_U(A)$: Inkonsistenz bzgl. *A* in *U*

$$IC_U(A) = X_u(A) - \max_{c \in C} X_u^c(A)$$

Für ganz U: $$IC(U) = \frac{\sum_{A \in DB} IC_U(A)}{|DB|}$$

Monotonie: $$U_1 \subset U_2 \Rightarrow IC(U_1) \geq IC(U_2)$$

Branch and Bound und Inkonsistenzkriterium

Vorteile:

- Monotonie ermöglicht die effiziente Suche nach optimalen Unterräumen mit Branch and Bound
- Gut geeignet für Binäre Attribute, da hier die Wahrscheinlichkeit von identischen Vektoren hoch ist.

Nachteil:

- Schlecht geeignet für numerische Attribute (ohne identische Vektoren nutzlos)
- Worst-Case Laufzeit immer noch exponentiell in *d*

K-dimensionale Projektionen und Modellabhängige Qualitätsmaße

Idee: Wähle *n* zufällige Teilräume der Zieldimensionalität *k* aus $\binom{d}{k}$ mögliche Teilräumen, teste diese und nimm den besten.

Anwendung:

- Benötigt Qualitätsmaß für ganze Teilräume.
- Aufwand und Qualtiät hängen stark von der Anzahl der Sample-Teilräume *n ab.*

Nachteil:

- Sucht nicht gezielt nach guter Kombination aussagekräftiger und unabhängiger Attribute
- Aufwand stark von der Wahl des Parameter n und angewendeten dem Qualitätsmaß abhängig.

Genetischer Algorithmus

Idee: Versuche gezielt hochqualitative Unterräume zu generieren.

Ansatz:

- **Population von Lösungen** := Menge *k*-dimensionaler Unterräume
- **Fitnesskriterium**: Qualitätsmaß für Unterräume
- **Mutationregel und Mutationwahrscheinlichkeit**:
  mit Wahrscheinlichkeit *x%* wird Dimension D_i in *U* durch D_j ersetzt
- **Fortpflanzung**: Kombinationsregel für 2 Unterräume *U1* und *U2*:
  Wähle *50%* der Dimensionen aus *U1* und *50%* aus *U2*
- **Selektionsregel**: Alle Kandidationräume die *z%* schlechtere Fitness haben als der beste der bisherigen Generation sind nicht lebensfähig.
- **Freilos**: Zusätzlich zur Selektion kann jeder Unterraum mit Wahrscheinlichkeit *u%* in die nächste Generation übernommen.

Genetischer Algorithmus

Ablauf:

```
Initialisiere Population
WHILE Max_Fitness > Old_Fitness  DO
        Mutiere Population gemäß Mutationsrate
        WHILE nextGeneration < PopulationSize DO
            Generiere neuen Kandidaten K durch Fortpflanzung
            IF K hat Freilos oder K ist fit enough THEN
                    K darf in die nächste Generation
 RETURN fittester Unterraum
```

Genetischer Algorithmus

Bemerkung:

- hier nur Skizze des Grundalgorithmus (viele Erweiterungen)
- Konvergenz meist nur unter „Simulated Annealing“ (Freiloswahrscheinlichkeit sinkt mit Anzahl der Generationen)

Vorteil:

- Vermeidung von lokalen Maxima
- häufig gute Approximation des optimalen Unterraums

Nachteile:

- kann lange Laufzeiten aufweisen
- viele Parameter müssen richtig gewählt werden, um einen guten Trade-Off zwischen Qualität und Laufzeit zu erzielen

Feature-Clustering mit Korrelation

Gegeben: Clusteringproblem über *F*.
Ziel: Reduziere Featurespace auf *k* Dimensionen.

Vorgehen: Da man irrelevante Attribute nicht erkennen kann, beschränkt man sich auf die Elimination von redundanter Information.

Idee: Clustere die Feature im Raum der Objekte und selektiere 1 Repräsentanten pro Cluster.

Maß für die Abhängigkeit Ähnlichkeit der Features:

Korrelation zwischen 2 Features: $COR(X,Y) = \frac{COV(X,Y)}{\sqrt{VAR(Y)}\sqrt{VAR(Y)}}$

Feature-Clustering mit Korrelation

Maximaler Information Compression Index (MICI):

Idee: Messe den kleinsten Eigenwert der Kovarianzmatrix Σ zwischen beiden verglichenen Unterräumen.

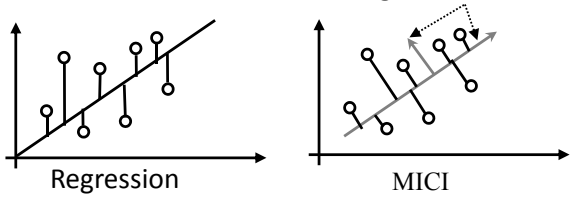
$$\det(\Sigma - \lambda E) = \det\begin{pmatrix} VAR(X)-\lambda & COV(X,Y) \\ COV(X,Y) & VAR(Y)-\lambda \end{pmatrix}$$

$$= (VAR(X)-\lambda)\cdot(VAR(Y)-\lambda) - COV(X,Y)^2$$

$$\Rightarrow 0 = \lambda^2 - \lambda\cdot VAR(Y) - \lambda\cdot VAR(X) + VAR(X)\cdot VAR(Y) - COV(X,Y)^2$$

$$\Rightarrow \lambda = \frac{(VAR(Y)+VAR(X)) \pm \sqrt{(VAR(Y)+VAR(X))^2 + 4\cdot 1\cdot VAR(X)\cdot VAR(Y) - COV(X,Y)^2}}{2\cdot 1}$$

$$MICI(X,Y) = VAR(Y)+VAR(X) - \sqrt{(VAR(Y)+VAR(X))^2 + 4\cdot VAR(X)\cdot VAR(Y) - COV(X,Y)^2}$$



MICI:
- Symmetrisch
- Kann aus beiden Dimensionen 1 gebildet werden, die die gesamte Information beider widerspiegelt.

Feature-Clustering mit Korrelation

Ablauf:
- Cluster Feature mit *k*-medoid Clustering für *k* Cluster und Abstandsmaß MICI.
- Selektiere die Clusterrepräsentanten als Feature-Dimensionen

Bemerkung:
- Versucht für jede Gruppe abhängiger Dimensionen eine representative Dimension
- Anwendung anderer Clustering Algorithmen denkbar.
  (K-Means: Wähle Dimension die am nächsten am Cluster-Centroid liegt)
- Häufig werden Cluster-Algorithmen für Streams verwendet, wegen Ihrer Laufzeit O(*d*)

Feature-Clustering mit Korrelation

Vorteile:

- Verhältnismäßig schnelle Selektionsmethode
- Kommt ohne Klasseneinteilung aus

Nachteile:

- Meist kein eindeutiges Ergebnis, da Clustering von Parametern und Reihenfolge abhängen kann.
- Repräsentative Dimensionen wechseln bei unterschiedlichen Cluster Algorithmen
- basiert auf paarweiser Korrelation

  =>höherwertige Dimensionen werden nicht untersucht.

Zusammenfassung der Ansätze

- *Forward-Selection*: Untersuche einzelne Dimensionen D‘ $\in\{D_1,..,D_d\}$. und bilde neue Unterräume durch Kombination.
  - Greedy Selection mit Information Gain, $\chi 2$ Statistik oder Mutual Information
- *Backward-Elimination*: Beginne mit dem gesamten Featurespace *F* und entferne überflüssige Dimensionen.
  - Greedy Elimination mit dem Modelbasierten oder Nächsten Nachbar Ansätzen
  - Branch and Bound Suche auf Basis des Inkonsistenzkriteriums
- *k-dimensionale Projektionen*: Suche direkt in der Menge der k-dimensionalen Unteräume nach einem geeigneten
  - Genetische Algorithmen zu Suche geeigneter Teilräume (Qualtität wie beim Backward Elimination)
  - Feature Clustering mit dem MICI

Feature-Selektion

Diskussion:

- Viele Algorithmen basierend auf unterschiedlichen Heuristiken
- Feature können aus 2 Gründen eliminiert werden:
  - es existieren andere bedeutungsgleiche Feature (Redundanz)
  - Features sind nicht mit der Aufgabe korreliert
- häufig können auch schon nicht optimale Ergebnisse sowohl Effizienz als auch Effektivität verbessern
- **Vorsicht**: Selektierte Features müssen keinen direkten Einfluß auf Zielvariable haben, sondern können auch nur von den gleichen versteckten Einflüssen abhängen.

Lernziele

- Gründe für Feature Selektion
- Curse-of-Dimensionality
- Forward Selection und Feature Ranking
  - Information Gain
  - $\chi 2$ Statistik
  - Mutual Information
- Backward Elimination
  - Modelbasierte Güte
  - Nächste Nachber Methode
  - Inkonsistenz und Branch & Bound
- k-dimensionale Projektionen
  - Genetische Algorithmen
  - Feature Clustering

Literatur

- A.Blum and P. Langley: *Selection of Relevant Features and Examples in Machine Learning*, Artificial Intelligence (97),1997
- H. Liu and L. Yu: *Feature Selection for Data Mining* (WWW), 2002
- L.C. Molina, L. Belanche, Â. Nebot: Feature Selection Algorithms: *A Survey and Experimental Evaluations*, ICDM 2002, Maebashi City, Japan
- P. Mitra, C.A. Murthy and S.K. Pal: *Unsupervised Feature Selection using Feature Similarity*, IEEE Transacitons on pattern analysis and Machicne intelligence, Vol. 24. No. 3, 2004
- J. Dy, C. Brodley: *Feature Selection for Unsupervised Learning*, Journal of Machine Learning Research 5, 2004
- I. Guyon, A. Elisseeff: An Introduction to Variable and Feature Selection, Journal of Machine Learning Research 3, 2003
- M. Dash, H. Liu, H. Motoda*: Consistency Based Feature Selection*, 4th Pacific-Asia Conference, PADKK 2000, Kyoto, Japan, 2000