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

Skript zur Vorlesung

Knowledge Discovery in Databases II

im Wintersemester 2013/2014

Kapitel 2: Feature Selection und Feature Reduktion

Vorlesung: PD Dr. Matthias Schubert
Übungen: Erich Schubert

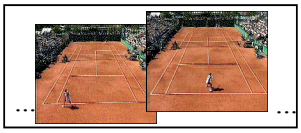
http://www.dbs.ifi.lmu.de/cms/Knowledge_Discovery_in_Databases_II_(KDD_II)

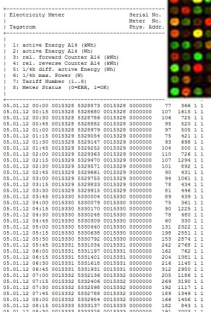
2. Featurereduktion und Featureselektion

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



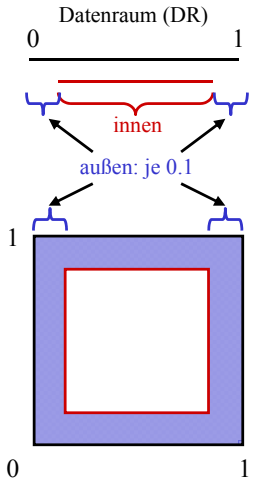


„Fluch der Dimensionalität“ (*Curse of Dimensionality*)

- 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 Qualitätsmaße haben keine Monotonie-Eigenschaften. (Features können erst in Kombination 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

$A = |\{o \mid x \le s \wedge Class(o) = C_j\}|$ 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.

$C = |\{o \mid x > s \wedge Class(o) = C_j\}|$ 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

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

rel. 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 *d* einzelne Features statt $\binom{d}{k}$ Unterräumen untersucht werden
- Abschätzung 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 die Qualität 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

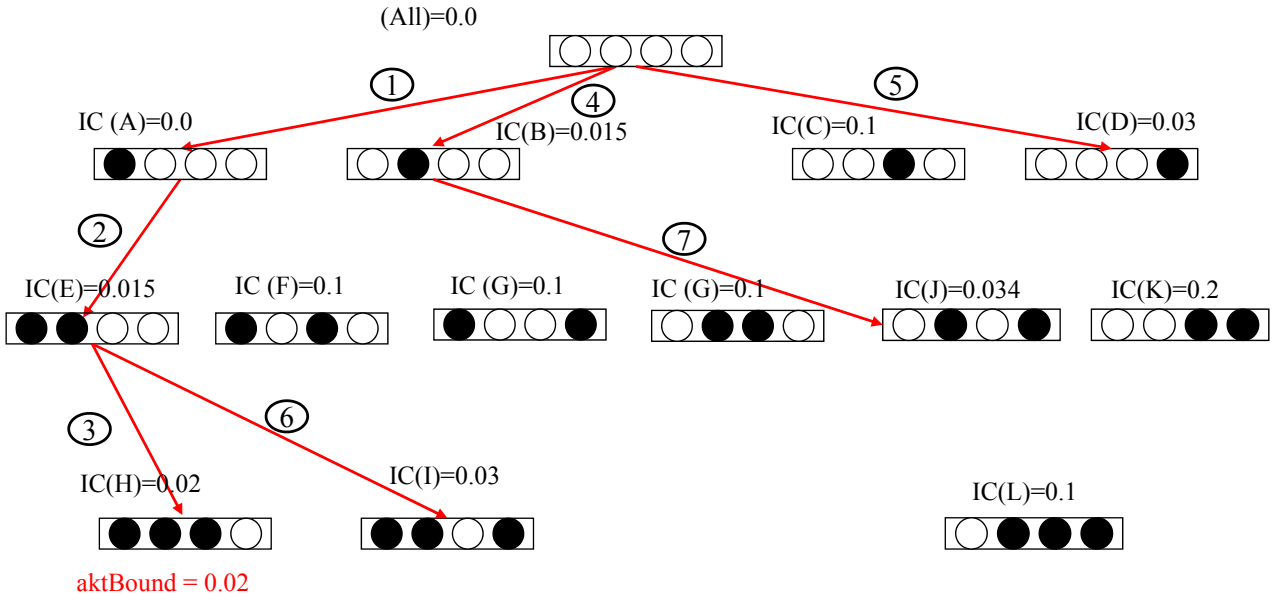
Backward-Elimination mit Branch and Bound:

```
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 Features im Raum der Objekte und selektiere einen repräsentativen Feature 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.

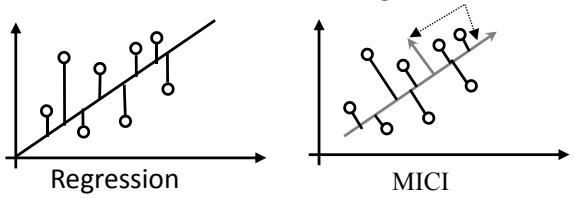
$$\det(\Sigma - \lambda E) = \det\begin{pmatrix} VAR(X)-\lambda & COV(X,Y) \\ COV(X,Y) & VAR(Y)-\lambda \end{pmatrix}$$

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

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

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

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



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

DATABASE SYSTEMS GROUP

Feature-Clustering mit Korrelation

LMU

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

2.3 Featurereduktion

Idee: Anstatt Features einfach wegzulassen, generiere einen neuen niedrigdimensionalen Feature-Raum aus allen Features:

- Redundante Features können zusammengefasst werden
- Irrelevantere Features haben einen entsprechend kleineres Gewicht in den neuen Features

Lösungsansätze:

- Referenzpunktansatz
- Hauptkomponentenanalyse (PCA)
- Singulärwert-Zerlegung (SVD)
- Fischer-Faces (FF) und Relevant Component Analysis(RCA)
- Large Margin Nearest Neighbor (LMNN)

1. Referenzpunkt Transformation

Idee:

Position eines Objekts kann häufig recht gut über den Abstand zu anderen Objekten beschrieben werden.

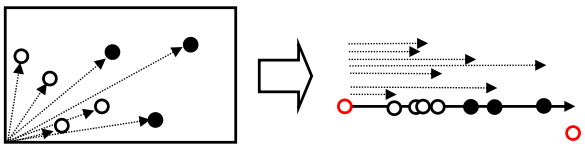
Wähle k Referenzpunkte und beschreibe Objekte über den k-dimensionalen Vektor der Abstände zu den Referenzpunkten.

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

Gesucht: *k*-dimensionaler Raum *R*, der für ein gegebenes Data Mining Problem eine optimale Lösung erlaubt.

Methode: Für die Menge der Referenzpunkte $R = \{r_1,..,r_k\}$ und Distanzmaß $d()$:.

Transformation von Vektor $x \in F$:



$$r = \begin{pmatrix} d(r_1, x) \\ \vdots \\ d(r_k, x) \end{pmatrix}$$

1. Referenzpunkt Transformation

- Abstandsmaß ist meist durch Applikation gegeben.
- Auswahl der Referenzpunkte:
  - Wähle Centroide der Klassen oder Cluster-Centroide als Referenzpunkte
  - Häufig Wahl der Referenzpunkte am Rand und möglichst weit weg von allen Datenobjekten.

Vorteile:

- leicht umzusetzender Ansatz

Nachteil:

- selbst bei gleicher Feature Anzahl ist die Abbildung nicht eindeutig
- Performanz stark von der Wahl der Referenzpunkte abhängig.

Latente Räume und Matrix Faktorisierung

Basics über Vektoren und Matritzen:

- Inneres Produkt von zwei Vektoren x,y:

$$x \cdot y^T = (x_1 \quad \cdots \quad x_d) \cdot \begin{pmatrix} y_1 \\ \vdots \\ y_d \end{pmatrix} = \langle x, y \rangle = \sum_{i=1}^{d} x_i \cdot y_i$$

- Äußeres Produkt vonzwei Vektoren x,y:

$$x^T \cdot y = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix} \cdot (y_1 \quad \cdots \quad y_d) = \begin{pmatrix} x_1 y_1 & \cdots & x_1 y_d \\ \vdots & \ddots & \vdots \\ x_d y_1 & \cdots & x_d y_d \end{pmatrix}$$

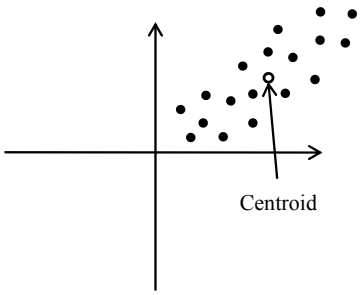
- Matrix-Multiplikation:

$$\begin{pmatrix} \vec{x}_1 \\ \vdots \\ \vec{x}_n \end{pmatrix} \cdot (\vec{y}_1 \quad \cdots \quad \vec{y}_m) = \begin{pmatrix} \langle \vec{x}_1, \vec{y}_1 \rangle & \cdots & \langle \vec{x}_1, \vec{y}_m \rangle \\ \vdots & \ddots & \vdots \\ \langle \vec{x}_n, \vec{y}_1 \rangle & \cdots & \langle \vec{x}_n, \vec{y}_m \rangle \end{pmatrix}$$

Datenmatrix

- Gegeben eine Menge von *n* Vektoren $v_i \in IR^d$, die *n*×*d* Matrix

$$D = \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} = \begin{pmatrix} v_{1,1} & \cdots & v_{1,d} \\ \vdots & \ddots & \vdots \\ v_{n,1} & \cdots & v_{n,d} \end{pmatrix}$$ heißt Datenmatrix

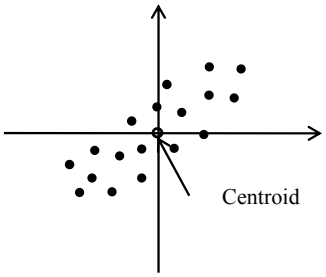


- Centroid/Erwartungsvektor einer Datenmenge:

$$\vec{\mu} = \frac{1}{n} \cdot \sum_{i=1}^{n} v_i$$

- Zentrierte Datenmatrix:

$$D_{cent} = \begin{pmatrix} v_1 - \vec{\mu} \\ \vdots \\ v_d - \vec{\mu} \end{pmatrix}$$



Mahalanobis Distanz

- Quadratische Form oder Mahalanobis Distanz:

$$d_A(x,y) = \left((x-y)A(x-y)^T\right)^{\frac{1}{2}} = \sqrt{(x-y)\begin{pmatrix} A_{1,1} & \cdots & A_{1,d} \\ \vdots & \ddots & \vdots \\ A_{d,1} & \cdots & A_{d,d} \end{pmatrix}(x-y)^T} = \sqrt{\sum_{i=1}^{d}\sum_{j=1}^{d}(x_i-y_i)A_{i,j}(x_j-y_j)}$$

Bermerkung: Ist A symmetrisch und positiv definit dann ist d_M eine Metrik.

- Gewichtete euklydische Distanz: A ist eine Diagonalmatrix mit $A_i > 0$:

$$d_A(x,y) = \sqrt{(x-y)\begin{pmatrix} A_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_d \end{pmatrix}(x-y)^T} = \sqrt{\sum_{i=1}^{d} A_i(x_i-y_i)^2}$$

- **Zusammenhang Basistransformation**:

  Wenn es ein symmetrische Zerlegung von $A = B \cdot B^T$ gibt, dann entspricht die Mahalanobis Distanz der euklydischen Distanz im mit B transformierten Raum:

$$d_M(x,y) = \left((x-y)B \cdot B^T(x-y)^T\right)^{\frac{1}{2}} = \left((xB-yB)\cdot(xB-yB)^T\right)^{\frac{1}{2}} = d_{eucl}(xB,yB)$$

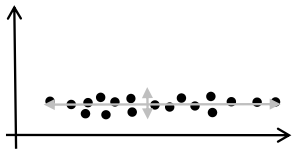
Varianzanalyse

- Welche Attribute sind die wichtigsten für die Bestimmung der Distanz?

  => Attribute mit stark unterschiedlichen Werten $|x_i - y_i|$

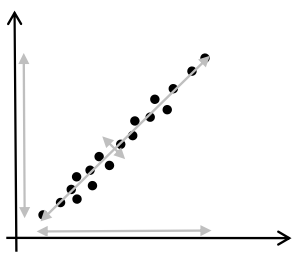
  => Abstand zum Erwartungswert ist groß $|x_i - \mu_i|$

  => Varianz ist groß: $\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_i)^2$



Idee: Varianzanalyse (eigtl. unsupervised Feature Selection)

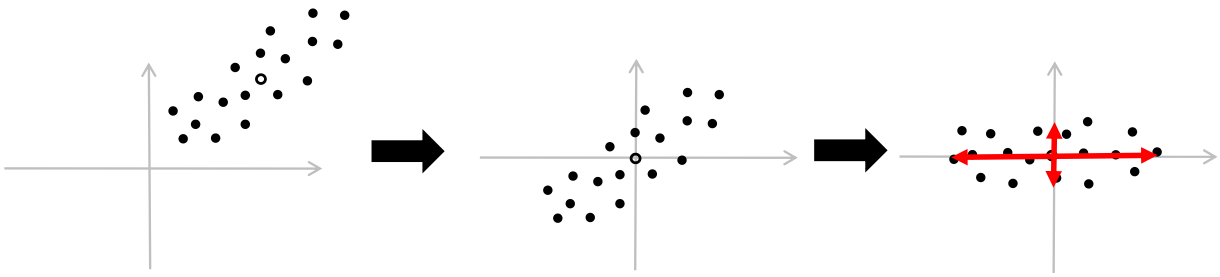
- Attribute mit großer Varianz ermöglichen Unterscheidung von Objekten
- Attribute mit kleiner Varianz: alle Objekte sind ähnlich unterschiedlich
- Vorgehen
  - Bestimme Varianz der Werteverteilung in allen Dimensionen
  - Sortiere alle Features absteigend nach der Varianz
  - Wähle die *k* mit der höchsten Varianz



ACHTUNG: Selbst lineare Korrelation kann eindimensionale starke Varianzen auf beliebig viele Dimensionen aufteilen !

Principal Component Analysis (PCA)

Idee: Rotiere die Datenmenge so, dass die Hauptausdehnungsrichtungen auf den Hauptachsen angetragen werden und erlaube so ein Varianzanalyse auf den Hauptkomponenten (Principal Components).



- Drehe so, dass die Richtung mit der höchsten Varianz auf eine Hauptachse gelegt wird.
- Rotation entspricht einer Basistransformation mit einer Orthonormalbasis
  - Abbildung ist winkeltreu und abstandserhaltend:

$$x \cdot B = x(b_{*,1},\ldots,b_{*,d}) = (\langle x, b_{*,1}\rangle,\ldots,\langle x, b_{*,d}\rangle) \quad mit \quad \forall_{i\neq j}\langle b_i, b_j\rangle = 0 \wedge \forall_{1\le i\le d}\|b_i\| = 1$$

- *B* entsteht aus der Richtung die jeweils die höchste Varianz aufweist und orthogonal zu allen bisher ausgewählten Basisvektoren ist.

Eigenwertzerlegung und Kovarianzmatrix

- Kovarianzmatrix:
  - Beschreibt Varianzen und Korrelationen(Kovarianzen)

$$VAR(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2 \qquad COV(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_x)(y_i-\mu_y)$$

  - Kovarianzmatrix Σ_D (*d*×*d*) der n×*d Datenmatrix D*:

$$\Sigma_D = \begin{pmatrix} VAR(X_1) & \cdots & COV(X_1,X_d) \\ \vdots & \ddots & \vdots \\ COV(X_d,X_1) & \cdots & VAR(X_d) \end{pmatrix} = \frac{1}{n} D_{cent}^T D_{cent}$$

- Eigenwert λ_i und Eigenvektor v_i einer Matrix *d*×*d D:* $D \cdot v_i = \lambda_i \cdot v_i$
- Eigenwertzerlegung: $M = V\Lambda V^T$

$$V = (v_1,\cdots,v_d) \quad mit \; \forall_{i\neq j}\langle v_i, v_j\rangle = 0 \quad und \; \forall_{i=1}^{d}\|v_i\| = 1$$

$$\Lambda = \begin{pmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d \end{pmatrix}$$

Eigenwertzerlegung der Kovarianzmatrix

- Wendet man die Eigenwertzerlegung auf die Kovarianzmatrix an:

$$\Sigma_D = V \Lambda V^T = (v_1, \cdots, v_d) \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_d \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix}$$

- v_i: Orthogonale Hauptkomponenten
- λ_i: Varianzen in den Dimensionen

Achtung: Bei nullwertigen sind bestimmte Dimensionen komplett als Linearkombination anderer Dimensionen darstellbar.

=> Abhilfe man addiert eine Diagonalmatrix mit einem kleinen Wert δ_i auf alle Diagonalelemente

Ablauf PCA

Dimensionsreduktion via PCA

1. Berechne Kovarianzmatrix Σ
2. Berechne Eigenwerte und Eigenvektoren von Σ
3. Wähle die *k* größten Eigenwerte und deren Eigenvektoren (v‘)
4. Die *k* resultierenden Eigenvektoren bilden eine Orthonormalbasis
5. Transformiere die $n \times d$ Datenmatrix D mit der neuen $d \times k$ Basis V‘:

$$D \cdot V' = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} (v'_1, \cdots, v'_k) = \begin{pmatrix} \langle x_1, v'_1 \rangle & \cdots & \langle x_1, v'_k \rangle \\ \vdots & \ddots & \vdots \\ \langle x_n, v'_1 \rangle & \cdots & \langle x_n, v'_k \rangle \end{pmatrix} = D$$

Singular Value Decomposition (SVD)

Verallgemeinerung der Eigenwertzerlegung

Zerlegung einer beliebigen $n \times d$ Matrix in 2 orthogonale Matritzen O,A und 1 Diagonalmatrix S aus Singulärwerten.

$$D = OSA^T$$

$$= \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,d} \end{bmatrix} = \begin{bmatrix} o_{1,1} & \cdots & o_{1,k} \\ \vdots & \ddots & \vdots \\ o_{n,1} & \cdots & t_{n,k} \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_k \end{bmatrix} \cdot \begin{bmatrix} a_{1,1} & \cdots & a_{1,d} \\ \vdots & \ddots & \vdots \\ a_{k,1} & \cdots & a_{k,d} \end{bmatrix}$$

O : n×*k* links-singuläre Vektoren, orthogonale Spaltenmatrix
S : k×*k* Diagonalmatrix aus Singulär-Werten
A : k×*d* rechts-singuläre Vektoren, orthogonale Spaltenmatrix
k : Rang der Datenmatrix D (max. Anzahl an unabhängigen Zeilen/Spalten)
Zerlegung mittels numerischer Algorithmen zur Matrix-Faktorisierung.

Zusammenhang SVD und PCA

- Anwendung der SVD auf die Kovarianzmatrix

$$D_{cent} = OSA^T$$

$$\Sigma_D = \frac{1}{n} D_{cent}^T D_{cent} = \left(OSA^T\right)^T OSA^T = AS^T\left(O^T O\right)SA^T = A\left(S^T S\right)A^T = A\begin{pmatrix} \lambda_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_k^2 \end{pmatrix} A^T$$

- In diesem Fall stellt A die Matrix der Eigenvektoren dar.
- Die Eigenwerte der Kovarianzmatrix sind die quadrierten Singulärwerte von D
- Da O eine n×k Matrix auf orthonormalen Spaltenvektoren ergibt O^TO die Einheitsmatrix E

Fazit: Sowohl die Eigenwerte als auch die Eigenvektoren der Kovarianzmatrix können auch über die SVD der Datenmatrix bestimmt werden.

⇒ Je nach Dimensionalität und eingesetzten Berechnungsalgorithmen ist SVD eine bessere alternative
⇒ SVD ist auch bei nullwertigen Eigenwerten einsetzbar (k<d ist bei SVD erlaubt)

Kernel PCA

Zusammenhang zwischen den Orthnormalbasen O und A: $D = OSA^T$

- A enthält eine k-dimensionale Basis aus Eigenvektoren von $D^T \cdot D$ *(vorherige Folie)*
- *Analog: O enthält eine* k-dimensionale Basis aus Eigenvektoren $D \cdot D^T$
  - $D \cdot D^T$ ist die Kernelmatrix zum linearen Kernel <x,y> (vgl. SVM aus KDD I)
  - Die Vektoren von A und O hängen dabei wie folgt zusammen

$$D_{cent} = OSA^T \Rightarrow O^T D_{cent} = O^T OSA^T = SA^T \Rightarrow S^{-1} O^T D_{cent} = A^T$$

$$\Rightarrow a_j = \sum_{i=1}^{n} o_{i,j} x_i$$

Der *j*-te *d*-dimensionale Eigenvektor kann als Linearkombination der Datenmenge mit dem Gewichtungsvektor des k-dimensionalen j-ten Eigenvektors dargestellt werden.

=> Hat man die Basis im Kernelraum kann man die Basis im Featureraum bestimmen

=> Diese Beobachtung erlaubt die Berechnung der PCA in beliebigen Kernelräumen (Kernel PCA)

Kernel PCA

Sei $K(x,y) = \langle \Phi(x), \Phi(y) \rangle$ eine Kernelfunktion zur nicht-linearen Transformation Φ(x).
Annahme: K(x,y) ist bekannt aber Φ(x) ist nicht explizit gegeben.

- Sei K die Kernelmatrix über D: $K = \begin{pmatrix} K(x_1,x_1) & \cdots & K(x_i,x_n) \\ \vdots & \ddots & \vdots \\ K(x_n,x_1) & \cdots & K(x_n,x_n) \end{pmatrix}$
- Die Eigenwertzerlegung von K ergibt dann: $K = VSV^T$
  wobei *V* eine n-dimensionale Basis aus Eigenvektoren von k ist
- Um *D* bzgl. *V* auf die Hauptkomponenten des Zielraums abzubilden müssen die Vektoren x_i in *D* mittels der Kernelfunktion *K* transformiert werden.

$$y' = \begin{pmatrix} \left\langle \Phi(y), \sum_{i=1}^{n} v_{i,1} \Phi(x_i) \right\rangle \\ \vdots \\ \left\langle \Phi(y), \sum_{i=1}^{n} v_{i,k} \Phi(x_i) \right\rangle \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} v_{i,1} \langle \Phi(y), \Phi(x_i) \rangle \\ \vdots \\ \sum_{i=1}^{n} v_{i,k} \langle \Phi(y), \Phi(x_i) \rangle \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} v_{i,1} K(y,x_i) \\ \vdots \\ \sum_{i=1}^{n} v_{i,k} K(y,x_i) \end{pmatrix}$$

Matrixfaktorisierung als Optimierungsproblem

SVD und Eigenwertzerlegung sind Standardverfahren aus der Mathematik.

- Matrixzerlegungen können aber auch als Optimierungsproblem formuliert werden.
- dies ermöglicht häufig eine Berechnung über numerischen Optimierung
- Bei der Formulierung als Optimierungsproblem wird die Diagonalmatrix häufig auf die resultierenden Matritzen aufgeteilt.

$$D = ASB^T = \left(A \begin{pmatrix} \sqrt{\lambda_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{\lambda_k} \end{pmatrix} \right) \left(\begin{pmatrix} \sqrt{\lambda_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{\lambda_k} \end{pmatrix} B^T \right) = UV^T$$

- Als Optimierungsproblem: $L(U,V) = \left\| D - UV^T \right\|_f^2$
  mit $\|M\|_f^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} |m_{i,j}|^2$ (quadriete Frobeniusnorm einer Matrix)
  Nebenbedingungen: $\forall_{i \neq j} : \langle v_i, v_j \rangle = 0 \wedge \langle u_i, u_j \rangle = 0$

Fischer Faces

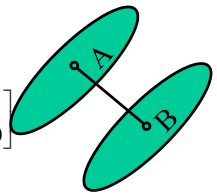
Idee: Nutze Klasseninformationen um relevanten Teil des Raumes zu erhalten.

Ziel:

- Minimiere die Ähnlichkeit zwischen Objekten unterschiedlicher Klassen
  (Between Class Scatter Matrix: Σ_b)
  Σ_b: Kovarianzmatrix der Klassencentroide

$$\Sigma_b = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\Sigma_w = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$



$$\bar{\mu} = \frac{1}{|C|} \sum_{c \in C} \mu_c \qquad \Sigma_b = \frac{1}{|C|} \begin{bmatrix} \mu_1 - \bar{\mu} \\ .. \\ \mu_m - \bar{\mu} \end{bmatrix}^T \cdot \begin{bmatrix} \mu_1 - \bar{\mu} \\ .. \\ \mu_m - \bar{\mu} \end{bmatrix}$$

- Maximiere die Ähnlichkeit zwischen Objekten derselben Klasse
  (Within Class Scatter Matrix Σ_w)
  Σ: Durchschnittliche Kovarianzmatrix innerhalb der Klassen

$$\Sigma_w = \frac{\sum_{C_i \in C} \Sigma_{C_i}}{|C|}$$

Fischer Faces

Suche Basisvektoren x_i so dass $S = \frac{x_i^T \cdot \Sigma_b \cdot x_i}{x_i^T \cdot \Sigma_w \cdot x_i}$ maximal wird unter der Bedingung $i \neq j : \langle x_i, x_j \rangle = 0$

Berechnung: Gesucht orthonormale Basis der Dimension d‘ < d.
Rückführung des Problems auf Eigenwertproblem.

$$\lambda_i \cdot x_i = \lambda_i \cdot \Sigma_w^{-1} \cdot \Sigma_b$$

Bemerkung: Der Vektor mit der dem größten Eigenwert entspricht dem Normalenvektor der trennenden Hyperebene einer LDA (Fisher‘s Diskriminanzanalyse)

Relevant Component Analysis (RCA)

Fischer Faces haben Nachteile bzgl. Σ_b und Σ_w :

- Annahme mono-modaler Klassen:
  jede Klasse lässt sich durch 1 Normalverteilung darstellen
  => Verteilung der Centroide modeliert Σ_b
  => Verteilung der Klasse basiert auf Kovarianz Σ_w für Klasse C
- Bei nicht-normalverteilten Klassen schlechte Darstellung der Daten

Relevant Component Analysis:

- Entfernen vollständig abhängiger Dimensionen (z.B. PCA mit Hilfe der SVD)
- Es existieren Chunks von denen man, weiß das Sie ähnlich sind.
  => ersetze Σ_w durch Within-Chunk-Matrix: $\Sigma_{wc} = \frac{1}{|C|} \sum_{C_i \in C} \frac{1}{|C_i|} C_i^T C_i$
- Betrachtet man die Kovarianz aller Daten ist diese überwiegend durch unähnliche Objekte bestimmt: $\Sigma = \frac{1}{|D|} D^T D$

Large Margin Nearest Neighbor (LMNN)

Beobachtung: Nicht alle Elemente einer Klasse müssen gleich sein.

Idee: Bilde ein Optimierungsproblem, dass nur die Distanz zu den *k*-nächsten Nachbarn der gleichen Klasse zu minimieren.

- $y_{i,j}=1$ falls x_i und x_j in der gleichen Klasse. $y_{i,j}=0$ *sonst*
- *Gesucht:* $L: IR^d \rightarrow IR^d$ lineare Transformation des Raums: $D(x,y) = \|L(x) - L(y)\|^2$
- Targetneighbors: T_x k-nächste Nachbarn innerhalb der gleichen Klasse
  $\eta_{i,j}=1$: x_j ist Targetneighbor von x_i, $\eta_{i,j}=0$ *sonst*
- Lernen der Transformation über Minimieren der folgenden Fehlerfunktion:

$$E(L) = \sum_{i=1}^{n}\sum_{j=1}^{n} \eta_{i,j} \|L(x_i) - L(x_j)\|^2 + c \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n} \eta_{i,j}(1 - y_{i,l}) \left[1 + \|L(x_i) - L(x_j)\|^2 - \|L(x_i) - L(x_l)\|^2\right]_+$$

  mit $[z]_+ = \max(z, o)$
- Optimierung über semidefinites Programm
  (Optimierungsproblem bei denen die zu optimierenden Paramter eine semidefinite Matrix bilden. hier L die Basis des Zielraumes)

Zusammenfassung

- Lineare Basistransformationen bilden ein reiches Framework zur Optimierung von Feature-Räumen
- Es gibt unsupervised Methoden, die niedrig-variante Faktoren eleminieren (PCA und SVD)
- Kernel PCA erlaubt die Berechnung der PCA in nicht-linearen Kernelräumen
- Es gibt Supervised Verfahren, die versuchen Distanzen zwischen Objekten der gleichen Klasse zu minimieren und Distanzen zwischen Objekten unterschiedlicher Klassen zu maximieren.
- Fischer Faces erweitern Lineare Diskriminanz Analyse basieren aber auf der Annahme normalverteilter Klassen
- Relevant Component Analysis(RCA) verallgemeinert diese Annahme und minimiert nur die Distanzen innerhalb von Chunks
- Large Margin Nearest Neighbor(LMNN) minimiert Distanzen zu den nächsten Nachbarn und bestraft kleine Distanzen zu nicht Nicht-Target-Neighbors, die zu anderen Klassen gehören.

Literatur

- S. Deerwester, S. Dumais, R. Harshman: *Indexing by Latent Semantic Analysis,* Journal of the American Society of Information Science,Vol. 41, 1990
- L. Yang and R. Jin. Distance metric learning: A comprehensive survey. Technical report, Department of Computer Science and Engineering, Michigan State University, 2006.
- K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classication. Journal of Machine Learning Research, 10:207,244, 2009.
- P. Comon. Independent component analysis, a new concept? Signal Processing, 36(3):287{314, 1994.
- J. Davis, B. Kulis, S. Sra, and I. Dhillon. Information theoretic metric learning. In in NIPS 2006 Workshop on Learning to Compare Examples, 2007.
- A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In Proceedings of the 20th International Conference on Machine Learning (ICML), Washington, DC, USA, pages 11{18, 2003.