

Analyse & Visualisierung

KDD - Pipeline (Knowledge Discovery in Databases)

Database



Focussing:

- Verständnis für Anwendung
- Definieren eines KDD-Ziels
- Erwerb von Daten
- Datenmanagement
- Auswählen relevanter Daten (Filtern)

Preprocessing:

- Zusammenführen / Integration von Daten unterschiedlicher Quellen
- Überprüfen von Konsistenz (auch lassen von Inkonsistenzen)
- Vervollständigen der Daten
- Data Foundations:
 - viele Quellen, Daten können roh/unbehandelt sein (Vorverarbeitung notwendig)
 - Datentypen:
 - nominal: ausschließende Kategorien, keine Ordnung, nur Prüfung auf $=/!=$
 - ordinal: haben Ordnung, Prüfung auch auf $>/<$, Abstände ohne Bedeutung
 - nummerisch: Zahlenwerte, Abstände von Bedeutung, mathematische Operationen möglich
Intervallskala: $1, -$ (z.B. Datum)
Verhältnisskala: $+, -, \cdot, :$ (z.B. Größe)
- Fehlerarten:
 - Random Error (Noise): kein Einfluss auf Mittelwert aber auf Varianz um den Durchschnitt
 - Systematic Error (Bias): Einfluss auf Mittelwert
- Missing values:
 - Daten, die noch nicht in den Datensatz eingefügt wurden, aber dessen Werte schon existieren
 - Behandlung:
 - ignorieren der Tupel
 - manuelles Nachragen der Daten
 - globale Konstante einfügen
 - Mittelwert einfügen
 - wahrscheinlichsten Wert einfügen
 - eingegebene Werte sollen Informationen nach Möglichkeit nicht beeinflussen

- Data Cleaning:

- noisy data (random error/variance in einem gemessenen Attribut)

- Gründe:

- fehlerhafte Erhebungsinstrumente
- Dateneingabeprobleme
- Datenübertragungsprobleme
- Techniklimitation
- Inkonsistente Namenskonventionen

- Umgang:

- Binning: glättet Daten oder stuft sie bis zu einer gewissen Genauigkeit ab, zur Verringerung der Varianz

- equi-depth: n Intervalle mit etwa gleicher Anzahl an Elementen

- equi-width: n Intervalle mit gleicher Größe ($\frac{\max - \min}{n}$)

- Smoothing nach:

- Mittelwert

- Median

- Randzuweisung

- Interpolation:

- Polynomische Funktionen

- teilweise polynomische

- orthogonale polynomische

- trigonometrische Funktionen

- $|f(x) - f_i(x_i)| = 0$

- Approximation:

- Regressionen

- $|f(x) - f_i(x_i)| \leq \epsilon$

- Regression: glätten durch Anpassen einer Regressionsfunktion

$$b = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$a = \bar{y} - b\bar{x}, \quad y = a + bx$$

- Clustering: überprüfen von auffälligen Werten durch Menschen

- Normalisation: versucht Datenpunkte auf Intervall $[0, 1]$ zu mappen

- linear: $f_{lin}(v) = \frac{v - \min}{\max - \min}$

- square root: $f_{sr}(v) = \sqrt{v} / \sqrt{\max^2 - \min^2}$

- logarithmic: $f_{ln}(v) = \frac{\ln(v) - \ln(\min)}{\ln(\max) - \ln(\min)}$

- Quantil: es wird Wert darauf gelegt, dass sehr große & sehr kleine Werte keinen starken Einfluss haben

- Segmentation: für d -dimensionalen Datensaum soll natürl. Partitionierung in Cluster & Rauschen gefunden werden, manuelle oder automatische (Clusteralgs)

- Data reduction:
 - Subsetting: Datensätze werden aufgabespezifisch betrachtet
 - Sampling: Datensätze zu groß
→ berücksichtigen einer repräsentativen Gruppe (muss Daten gut repräsent.)
 - non-probabilistic Sampling:
Auswählen auf nicht-zufallsbasierten Basis
 - probabilistic Sampling:
 - Systematic Random Sampling
(jeder k-te Datenpunkt gewählt)
 - Simple Random Sampling
(einfache Zufallszahl d. Punkte)
 - Stratified Random Sampling
(Gruppen bilden, in diesen zufälliges Sampling)
 - Cluster Random Sampling
(cluster des Datensatzes, in diesen zufälliges Sampling)
 - Biased Sampling
 - Dimensionssenkung: un relevante/redundante Attribute/Dimensionen werden entfernt
 - Daten sind schwer darzustellen
 - Lösung: Projektion (in viele weniger dimensionale Datensätze), Clustering (weniger hochdim. Daten)
 - Projektion: identifizieren der wichtigsten Eigenschaften
→ vereinfachen des Prozesses ohne Verlust der Qualität
 - PCA: Finden von versteckten Faktoren, die die Daten erklären, verringern der Dimensionen
 - weglassen der Daten mit geringstem Eigenvektor
 - kann nur auf numerische Daten angewendet werden
 - benutzt, wenn man gering korrelierende Daten darstellen möchte (Maximierung d. Varianz)
 - nicht sinnvoll für gleichmäßige Relationen
 - nicht robust gegen Outlier
 - robust gegen Noise (mitger. Dichte)
 - robust gegen kleine Rotationen im ges. Daterraum (bewirkt keine Änderung der Korrelation)



Transformation:

- Diskretisierung von numerischen Werten (abhängig von DU-Task)
- Generierung von abgeleiteten Attributen (Aggregation über Menge von Datensätzen, Kombination mehrerer Attribute)

- Auswahl von Attributen:
 - manuell (falls Fachwissen über Atribut-Semantik und DM-Task vorhanden)
 - automatisch (bottom-up (Beginn mit leerer Menge), top-down (Beginn mit Menge aller Attribute))
- zu viele Attribute führen zu ineffizientem/ ineffektivem DM
- manche Transformationen können mit OLAP-Systemen realisiert werden

Data Mining: - Definition: DM ist die Anwendung von effizienten Algorithmen, die Patterns (Muster), die in einer Datenbank vorhanden sind, bestimmen.

- Hauptaufgaben:

• Clustering:

- Distanzfunktionen: $\sqrt[p]{\sum_{i=1}^d |x_i - y_i|^p}$
- L_p -Metrik (Minkowski)
- Euklidische ($p=2$)
- Manhattan ($p=1$)
- maximum Distanz $\max \{ |x_i - y_i| \mid 1 \leq i \leq d \}$

- Eigenschaften:

- unterschiedliche Größe, Form, Dichte
- Hierarchie
- überlappend oder disjunkt
- identifiziert endliche Menge an Klassen, Kategorien, bei denen Objekte ähnlich sein sollen, die außerhalb möglichst unterschiedlich

- Ziele:

- Data Understanding (finden natürl. Cluster)
- Data class Identification
- Data reduction
- Outlier / Noise detection

- Partitionierungsmethoden

• k-means (ℓ -median)

- 1) k und (random) k erste Cluster-mittelpunkte bestimmen
- 2) jedem Datenpunkt nächsten Mittelpunkt zuweisen
- 3) Mittelpunkte neu berechnen
- 4) 2 & 3 wiederholen bis keine Veränderung (oder max. t erreicht)

- leicht durchführbar

- keine Noise Erkennung

- #Cluster muss bekannt sein

- keine arbitrary shapes erkannt

• PAM (Partitioning around Medoids) berechnet von k Repräsentanten (alle Kombinationen) Kompaktheit (Kosten), Auswählen Kombination mit geringsten Kosten, alle Objekte wählst ein Medoid zu weisen

- kann bei lokalem Optimum terminieren

- CLARA (Clustering Large Applications)
wie PAM, nur bedenken werden auf Teilmenge (Sample) berechnet, für schlechtere Samples wiederholt, bestes wird benutzt
- CLARANS (CLARA based upon Randomized Search)
wie CLARA, nur Anzahl an Iterationen und Anzahl geprüfter Nachbarn pro Iteration sind begrenzt
+ Laufzeit meist $O(n^2)$
- EM (Expectation Maximization)
Generalisierung von k-Means:
Zuweisung der Punkte basiert auf Wahrscheinlichkeiten
Idee:
 - Schätzen der Parameter der K Normalverteilungen
 - Optimieren der Wahrscheinlichkeiten, dass Mix der parametrisierten Normalverteilungen zu den Daten passt
 - liegt in $O(n \cdot k \cdot \# it)$
 - kann bei lokalem Minima terminieren
- Linkage-Based Methods
 - top-down: Alle Objekte initial in einem Cluster, dann Splits (Dividing)
 - bottom-up: Jedes Objekt initial als eigenes Cluster, dann merge (Agglomerating)
 - Distanzfunktionen:
 - Single: $\text{dist}(C_1, C_2) = \min \text{dist}(p, q)$
 - Complete: $\text{dist}(C_1, C_2) = \max \text{dist}(p, q)$
 - Average: $\text{dist}(C_1, C_2) = \frac{1}{\#C_1 \cdot \#C_2} \sum_{p \in C_1} \sum_{q \in C_2} \text{dist}(p, q)$
(durchschnittlicher Abstand)
 - Centroid: $\text{dist}(C_1, C_2) = \text{dist}(\text{mean}(C_1), \text{mean}(C_2))$
 - Finden von bel. Clustern, hierarchische Cluster, kein k benötigt
 - nicht robust gegen Noise, ineffizient ($O(n^2)$)
 - BIRCH: es werden Zusammenfassungen von „Mikro-Clustern“ bestimmt, Auf unten liegende Cluster in CF-Tree kann beliebiger Clusteralgorithmus angewandt werden
 - CF-Tree (Clustering Feature):
 - Menge C aus Punkten x_i : $CF = (N, LS, SS)$
 - $N = |C| = \# \text{Punkte in } C$
 - $LS = \text{lineare Summe der } N \text{ Punkte}$
 - $SS = \text{quadratische Summe der } N \text{ Punkte}$
 - ist höchstbalancierter Baum zum Speichern von CF's
 - Effizient ($O(n) - O(n \log n), \dots$)
 - nur für numerische Daten
 - Dichtebasierter Clustering
 - für jedes Cluster muss die lokale Dichte eine gewisse Grenze überschreiten, Objekte eines Clusters müssen räumlich verbaudelt sein
 - Cluster können beliebige Form haben, robust gegen Noise, effizient

• Basics

- Core Object: Objekt o hat mindestens minPis Nachbarn in seiner Nähe (ϵ -Umgebung)
- Border Object: Alle, die kein Core sind
- directly density-reachable Object: Nachbar eines Core Objektes
- density-reachable Object: transiv über mehrere Core Objekte erreichbar
- density-connected Object:
 p und q sind d-c $\Leftrightarrow p$ & q sind zu drittem Punkt density-reachable
- Cluster: müssen Maximalität und Komplettivität haben
- Noise: alles, das zu keinem Cluster zugeordnet werden konnte
- Parameterwahl (ϵ , minPis) muss passend gewählt werden, Problem bei unterschiedlichen Dichten

• DB Scan

- Für jeden Punkt wird ϵ -Umgebung bestimmt und geprüft ob genug Punkte enthalten sind.
 - 1) Bestimmen von minPis, ϵ & ϵ -Umgebung (für jeden Punkt)
 - 2) Falls Punkt CoreObject, weise ClusterID auf, mache mit allen directly density-reachablen Objekten weiter und weise ihnen gleiche ClusterID zu, falls # direkter Nachbarn < minPis setze Punkt Noise
- Hierarchical Density-based Clustering (Optics)
- modifizierter DB-Scan
- hat variables ϵ
- gibt "Diagramm" aus, auf dem Täler und Berge zu erkennen sind (Täler = Cluster)
- Kernel Density Estimation (wie EM)
- DENCLUE:
 - kann je nach Parameterwahl verschiedene Clustering-Verfahren generalisieren (dichtebasierte, partionierende, hierarchische)
 - Verwendung einer Dichtefunktion, die den Einfluss eines Objektes auf seinen Nachbarn bestimmt
 - gesuchte Dichte des Datenraumes zeigt sich aus den Dichtefunktionen aller Objekte, lokale Maxima dieser Dichtefunktion ergeben die Cluster
 - robust gegen Noise, unregelmäßig geformte Cluster werden gefunden, gute Verarbeitung von hochdimensionalen Daten
 - Parameterwahl ist problematisch

- Klassifikation

• Evaluation of Classifiers

- Train and Test: Trainingsdaten

trainieren den Klassifikator,
Testdaten werden ihm aus

- m-fold cross validation: Daten werden
in m gleichgroße Teilmengen geteilt,
m Klassifikatoren werden trainiert
mit m-1 Teilmengen, zum Testen
wird die übrige Teilmenge verwendet,
Auswertung aller m Klassifikatoren
wird kombiniert

• Auswahlkriterien:

- Genauigkeit der Klassifikation
- Nachvollziehbarkeit
- Effizienz der Modellkonstruktion
und Anwendung
- Eignung für große Datensätze
- Robustheit

• Klassifikationsgenauigkeit:

$$G_{\text{Test}}(K) = \frac{\# \text{ richtig klassifizierte}}{\# \text{ aller Testobjekte}}$$

• Klassifikationsfehler:

$$F_{\text{Test}}(K) = \frac{\# \text{ falsch klassifizierte}}{\# \text{ aller Testobjekte}}$$

• Confusion matrix:

		PP	P'N	Precision = $\frac{TP}{TP+FN}$
AP	TP	FN	$\frac{TP}{TP+FP}$	
	AN	FP		

$$\text{Recall} = \frac{TP}{TP+FN}$$

- erwünscht: high P & high R

$$F\text{-Measure} = \frac{2 \cdot P \cdot R}{P+R}$$

• Decision trees

- Konstruktionsalgorithmus:

Alle Daten gehören zur Menge. Nun
wählt das wichtigste Attribut \rightarrow sinnvoller
Split nach diesem Attribut, weiter
von vorne bis keine Attribut mehr
zum splitten da sind

$$-\text{Entropy: } E(T) = -\sum_i p_i \cdot \log_2(p_i)$$

$$-\text{Information Gain: } \text{Gain}(T, A) = E(T) - \sum_i \frac{|T_i|}{|T|} \cdot E(T_i)$$

$$-\text{Gini - Index: } G(T) = 1 - \sum_j p_j^2$$

je kleiner, desto besser

- Overfitting: zu hohe Klassifikations-
genauigkeit führt zu schlechterem
Ergebnis auf Testdaten

• Verhindern durch:

- minSup, minConf (meist vorkommende
Klasse soll geringen Anteil an Blatt-
knoten haben), Cleaning / Pruning,
Cross-Validation, # behandelte
Attribute verringern

- ID3-Algorithmus

- 1) Berechnen der Entropy für Klassenattribut
- 2) Berechnen der Entropy für alle Attribute:
 - a) falls $E(T) = 0 \Rightarrow$ bilde Blatt
 - b) nimm Attribut mit geringster Entropy und tue 2^g erneut für Teilräume

- C4.5 Vorteile gegenüber ID3

- Zahlenwerte möglich
- Gain Ratio (modifizierte Splitkriterien)
- Regeln extrahieren \rightarrow stoppt verarbeiten von Nodern, die keinen Gewinn bringen

$$\text{- Splitinformation } (S, A) = - \sum_{i=1}^{|S|} \frac{|S_i|}{|S|} \cdot \log_2 \left(\frac{|S_i|}{|S|} \right)$$

$$\text{- GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\text{Splitinfo}(S, A)}$$

• Bayesian Classification

Für jedes zu klassifizierende Objekt wird die Wahrscheinlichkeit berechnet

- Optimal Bayes Classifier

$$\arg\max_{C_j \in \text{Klasse}} \sum P(C_j | h_j) \cdot P(h_j | o)$$

- Naïve Bayes

$$\arg\max_{C_j \in \text{Klasse}} P(C_j) \cdot \prod_{i=1}^d P(O_i | C_j)$$

• Nearest-Neighbor-Classification

- Mean-Vektoren für alle Objekte einer Klasse berechnen

- zu klassifizierendes Objekt der Klasse zuweisen, dessen Mean-Vektor dem Objektvektor am nächsten ist (bei 1NN)

- Klassen können gewichtet werden

- benötigt: Distanzfunktion, k

- falls k sehr klein \rightarrow auffällig für Outlier

- hohe Klassifikationsgenauigkeit, inkrementell

- teure Anwendung, kein genaues Wissen über die Daten generiert

• SVM

- sucht Hyperebene, die den Datenspace auf besten scheidt

- Hypes plane mit maximaler Distanz zur nächsten Trainingsobjekten wird gewählt

- Klassifikation vs. Clustering

• Klassifikation: Hänge an Klassenverfahren bekannt

• Clustering: Klassen werden gesucht

- Association Rules

• Frequent Pattern Analysis: Suche nach wiederkehrenden Mustern

$$\bullet \text{Support } \sigma = \frac{\# \text{HemA}}{\# \text{aller Hemsets}}$$

$$\bullet \text{Confidence } c = \frac{\sigma(\Sigma A \cap B)}{\sigma(\Sigma A)}$$

• Frequent Itemset Mining

- Apriori -Algorithmus

- Wenn ein Itemset frequent ist, müssen alle Teilmengen frequent sein, wenn Teilmenge nicht frequent \rightarrow Obermenge auch nicht \rightarrow wird nicht getestet

- 1) Scannen der Database nach frequent-1-Itemsets
- 2) generieren von k -candidate-Itemsets mit den $k-1$ frequent-Itemsets
- 3) Join: alle Itemsets, deren Elemente sich nur im letzten unterscheiden werden kombiniert

Prune: alle $k-1$ großen Teilmengen werden auf frequent getestet, falls nicht \rightarrow wieder loslau

- FP-Tree

- 1) alle Frequents aus der Datenbank bestimmen (für 1-el., in Bezug auf minSup)
- 2) sortieren der Frequents nach Häufigkeit
- 3) Datenbank wieder scannen und FP-Tree konstruieren: (minSup bestimmt) anhand der sortierten frequent itemsets werden Triebe konstruiert

• Vollständig, kompakt

- FP-Growth

- FP-Tree aufbauen und rekursiv alle tieferen conditional FP-Trees aufbauen, stoppen falls nur noch ein Pfad vorhanden oder resultierender FP-Tree leer ist

Patterns

Evaluation:

- Präsentation der erkannten Muster durch geeignete Visualisierung unterstützt
- Auswertung der Muster durch Benutzer
- falls Auswertung unbefriedigend: wiederholen des DU mit anderen Parametern / Metriken / Daten
- falls ok: Integration der erkannten Wissens in Unternehmenswissenbasis, benutzen des neuen Wissens für zukünftige KDD-Prozesse

Knowledge

Human Perception

- Warum auf Wahrnehmung achten?
 - Hörgerät, Film/TV, Cockpit (Wo ist was?), ...
- Gestalt Laws:
 - 1) Law of Proximity (Nähe)
 - 2) Law of Closure (Geschlossenheit)
(Verbinden zu Ganzen)
 - 3) Law of Symmetry ([] [])
 - 4) Figure-Ground Segregation (Figur-Grund Abgrenzung)
(mehrere Bilder in einem)
 - 5) Law of Good Continuation (gute Fortführung)
 - 6) Law of Similarity (Ähnlichkeit)
(ähnliche Form/Farbe, etc.)
- Visualisierungstechniken
 - Expressiveness: Visualisierung teilt alle Informationen - und nur diese - mit
 - Effectiveness: Informationen lassen sich leicht interpretieren
- Visuelle Variablen
 - 1) Position
 - 2) Farbe
 - 3) Helligkeit
 - 4) Form
 - 5) Ausrichtung
 - 6) Größe, Länge, Volumen
 - 7) Textur
- Effeekte visueller Variablen:
 - Selektiv (zuordnung zu Gruppen) - 6, 3, 7, 2, 5
 - Assoziativ (Ähnliche Objekte in Beziehung) - 7, 1, 2, 3, 5, 4
 - Separation - 7, 1, 2, 5, 4
 - Ordinal (Ordnung (Rangordnungen)) - 7, 1, 2, 3
 - Proportional (relative Größe) - 6, 5, 3
- Visual Analytics Mantra:

Analyse first, show the important, zoom, filter
and analyse, details on demand.