1 Basics

1.1 Motivation

Unterschiede der heutigen Situation zu früherer Zeit:

- große Datenmengen
- ungezielte Datenerhebung führt zu anderer (niedrigerer) Qualität der Daten

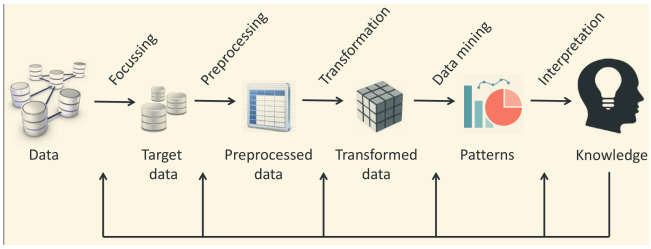
Information Mining Durch die Verwendung von Computern und entsprechenden Methoden hat ein Forscher deutlich mehr Möglichkeiten, mit den gegeben Daten zu arbeiten und ggf. Muster zu fördern, die nicht offensichtlich sind.

1.2 KDD

Knowledge Discovery in Databases (KDD) bezeichnet das (halb-)automatische Finden von Wissen ausgehend von Datenbanken, Wissen ist hierbei auch:

- Korrekt (valide)
- Bisher unbekannt
- Potentiell nützlich

Im Überblick:



KDD Process Model. Siehe auch Slide "01/Terminology"

KDD Process Model *5 Punkte*

1. *Focussing*: Einschränkung auf für Fragestellung relevante Daten.
2. *Preprocessing* Umgang mit unvollständigen, fehlerhaften oder ungeeigneten Daten. Methoden:
   - Data Reduction, z.B. sampling *(cf. 1.4.5)*
   - Data Cleaning *(cf. 1.4.1)*
   - Normalisierung *(cf. 1.4.2)*
   - Segmentierung

   Aufgabe: Wahl geeigneter Methoden, keine Verfälschung von Daten.
3. *Transformation*: ... in Form, auf die Algorithmen angewandt werden können. *z.B. Aufteilung pro Jahr, pro Region; oder auch Reduktion von Dimensionen.*
4. *Data Mining*: Anwendung statistischer Methoden, Algorithmen auf den Datensatz, um Muster oder Trends zu erkennen. *Aufgabe*: Wahl von geeign. Algorithmus und Parametern.
5. *Interpretation*: Wahrnehmen und Verstehen der Muster um Fakten oder neues Wissen ableiten zu können. *Aufgabe*: Ausgabe verstehen, ist Ergebnis sinnvoll?

Ist insgesamt iterativer Prozess. *u.a. Blatt 02*

1.3 Data Foundation

1.3.1 Daten-Typen

- *Nominal* Unterscheidbare Klassen. Keine quantitative Beziehung zwischen Kategorien, keine Ordnung. *zB Haarfarbe*
- *Ordinal* Ordnung aber keine sinnvolle Distanzangabe möglich *zB Skala Strongly Agree, ..., Strongly Disagree*
- *Numerisch* Ordnung gegeben, Abstände sinnvoll bestimmbar, mathematische Operationen möglich. *zB Körpergröße*

1.3.2 Typical Data Classes

- Scalar - Einzelner numerischer Wert
- Multivariate und multidimensionale Daten *zB Objekt ist eine Person, Dimensionen/Variablen sind Alter, Größe, ...*
- Vektoren *zB Einkaufswagen, Farbwert*
- Netzwerke / Graphen *zB Freundesnetzwerke*
- Hierarchische Daten *zB Computernetzwerke*
- Time-Series Data *zB Aktienkurse*

Multivariat vs. Multidimensional *Lediglich Anhaltspunkte, je nach Anwendung evtl. anders aufzufassen.*

- ***Dimensionen*** Unabhängige Variablen, fester, gegebener Raum *zB Koordinaten in 3D-Raum*.
- ***Multivariat*** Abhängige Variablen *zB Temperatur, gemessen an einem bestimmten Ort*.

Zu fragen ist: "Was messe ich, was ist gegeben?"

1.4 Data Preprocessing

1.4.1 Data Cleaning

Umgang mit **fehlenden Daten**:

- **Ignorieren**
  - ⊕ Einfach
  - ⊕ Kein Rechenaufwand
  - ⊖ Verliere Information
- **Manuell ausfüllen**
  - ⊖ Nur für kleine Datensätze machbar
  - ⊖ Wert muss in Erfahrung gebracht werden
- **Globale Konstante**
  - ⊕ Einfach, schnell
  - ⊖ Wert darf nicht für weitere Berechnungen hinzugezogen werden
- **Durchschnitt** des Attributs verwenden
  - ⊕ Einfach
  - ⊖ Womöglich unrealistisch (zB in Clustern)
- **Wahrscheinlichsten Wert** verwenden
  - ⊕ Genau
  - ⊖ Rechenaufwendig

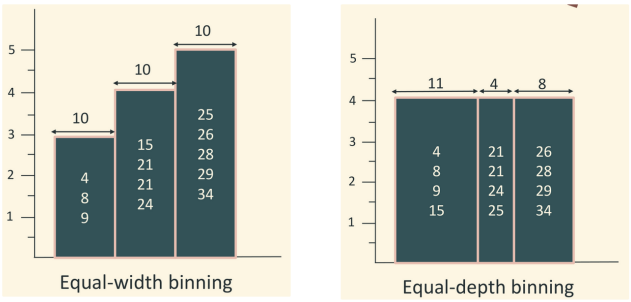
Umgang mit **"noisy" Daten**: Binning, Regression, Clustering, ...

Noise random error or variance in a measured variable.
:

Random error vs. Systematic error zufällige, *gleichverteilte* Abweichung; ggü. *systematic error* wie zB Verschiebung von Durchschnitt

Binning Entfernt unerwünschte Varianz bzw. Rauschen. Lege Intervalle in Dimension fest, ersetze Datenpunkte im selben Intervall durch genau einen Repräsentanten.

- **equal-width b.**: Intervalle sind von gleicher Größe
  - ⊖ **Vorsicht:** Daten mit Schwerpunkt, Outliern werden nicht korrekt abgebildet *da einzelne Outlier dann ein ganzes Bin erzeugen*
- **equal-depth b.**: Intervalle sind von gleicher Kardinalität
  - ⊕ Daten mit Schwerpunkt, Outliern (*skewed*) werden besser verarbeitet.
- ***smooth by bin means*** Jeder Wert in einem Bin wird durch den Durchschnittswert des Bins ersetzt.
- ***smooth by bin boundaries*** Jeder Wert in einem Bin wird durch den nächsten Boundary-Wert ersetzt.



equal-width vs equal-depth binning

Regression *Lineare Regression* findet Linie, die den *square error* minimiert. Linie: $y = a + bx$, $\bar{x}, \bar{y}$ arith. Mittel.

1. Steigung bestimmen: $b = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$
2. Bestimme damit a, verwende $a = \bar{y} - b\bar{x}$

Nur sinnvoll, wenn Daten sich auch einigermaßen durch eine Linie beschreiben lassen.
Desweiteren anwendbar für

- Prediction (Vorsicht!)
- Missing Values
- Save Disk Storage (insofern quasi-perfekte Korrelation)

Vorsicht bei *extra*polation, d.h. Ableiten von Werten außerhalb des gemessenen Bereichs.

Clustering *Fasse bzgl. Clustern "ähnliche"Datenpunkte zusammen.* cf 3

1.4.2 Normalisierung

Einsatzzwecke

- Abbildung verschiedener Attribute auf einen gemeinsamen Wertebereich stellt **Vergleichbarkeit** her, setzt Attribute zueinander in Beziehung. *z.B. Alter* $\in \{0, ..., 120\}$, *Gehalt* $\in \{0, ..., 10.000\}$, *bilde beides auf* $[0, 1]$ *ab.*
- **Mapping** eines Attributs auf einen anderen Wertebereich bezüglich vorliegendem Minimum und Maximum (zB $[0, 1]$ ermöglich einfacheres Mapping auf anderen Wertebereich) wie zB eine Farbskala für die Visualisierung.
- Um besser und effizienter damit arbeiten zu können.
  - *Temperaturaufzeichnungen zB zwischen ca. 0 und 30 Grad, besonders interessant sind aber kleine Veränderungen am oberen Ende von Wertebereich → logarithmische Normalisierung*
  - *Als Eingabe für Algorithmus*

Beachte Nur sinnvoll auf numerischen Daten mit Abstandsmaß. *zB nicht für Schulnoten.*

Formeln

- *Linear*: $f_{lin}(v) = \frac{v - min}{max - min}$
  Vorsicht: Outlier verzerren Wertebereich stark.
- *Logarithmisch*. $f_{log}(v) = \frac{\log(v) - \log(min)}{\log(max) - \log(min)}$
  "Verzerrt"den Wertebereich logarithmisch, d.h. in den unteren Bereichen werden die Abstände gestreckt, in den oberen gestaucht; macht somit manche Features besser sichtbar. **Vorsicht**: Daten dürfen nach dieser Transformation nicht falsch interpretiert werden.

Probleme bei Normalisierung von Data Streams Bisheriges Minimum, Maximum ist nicht zuverlässig. Abhilfe:

- Lege semantische Grenzen fest insofern möglich *zB Alter, Temperatur*
- Lasse neue Datenpunkte einfach weg, falls sie außerhalb liegen.
- Normalisierung neu durchführen.

1.4.3 Segmentation

Ziel: Daten in Teilmengen aufteilen, sodass Daten aussagekräftiger und einfacher zu analysieren.

- **Manuell**
- **Automatisch**, zB via. Clustering

1.4.4 Data Reduction

Motivation Der Umfang des ursprünglichen Datensatzes ist oft nicht ohne weiteres zu bewältigen.

Ziele

- Umfang der Daten reduzieren und gleichzeitig
- möglichst deren Aussagekraft bewahren

Probleme Ausgewählte Teilmenge muss entsprechend Fragestellung aussagekräftig ggü. der Gesamtpopulation sein.

Möglichkeiten

- Reduktion von Datenpunkten → Sampling
- Reduktion der Dimension → zB PCA

1.4.5 Sampling

Probabilistic vs. Nonprobabilistic Sampling *Probabilistic Sampling* bedeutet, dass irgendeine Art von zufälligem Prozess beim Sampling involviert ist; beim *nonprobabilistic sampling* wird deterministisch bestimmt.

Sampling-Techniken

- *Random Sampling*

  ⊕ Einfach
  ⊕ Praktisch kein Bias.
  ⊖ Nicht geeignet, wenn kleine Teilmengen von Interesse, da diese nach fast gar nicht mehr repräsentiert sind → *Stratified Random Sampling*
  ⊖ Charakteristika können von sampling möglicherweise verfehlt werden *zB Netzwerkstrukturen*
  → Anzuwenden, wenn vorgehen einfach sein muss oder Zufälligkeit wichtig ist.

- *Systematic Random Sampling* Wähle Sortierungsattribut A und $k \in \mathbb{N}$. Datensatz wird nach A sortiert, dann wird beginnend von einem *zufälligen* Index aus $0, ..., k$ jedes k-te Element als Sample gewählt.

  ⊕ Gleichmäßige Auswahl sichergestellt
  ⊖ Potentieller Moire-Effekt wenn Sampling-Muster mit Muster in den Daten übereinstimmt.
  ⊖ Löse durch Sortierung ggf. Beziehung zwischen Attributen auf.
  ⊖ Wahl des Sortierungsattributs bring Bias.
  → Anzuwenden, wenn vorgehen einfach sein muss und gleichmäßiges Sampling wichtig ist.

- *Stratified Random Sampling* Datenmenge wird in Segmente unterteilt, aus jedem dieser Segmente werden Samples gewählt. Entweder jwls. gleich viele oder unterschiedlich viele.

  ⊕ Für einzelne Strata können untersch. Sampling-Techniken verwendet werden.
  ⊕ Habe jede Gruppe repräsentiert.
  ⊖ Nicht unbedingt klar, wie Strata definiert werden sollen
  ⊖ Größenverhältnisse zwischen Gruppen gehen verloren. *Bei manchen Gruppen geschieht over-, bei manchen undersampling*
  → Datenerhebung kann einfacher sein, wenn Strata bekannt.

  zB Migration von Bevölkerungsminderheiten, Aussagen bzgl. Bildungsabschluss. Wenn Strata unterschiedlich dicht sind (zB Bevölkerung in Distrikten) manchmal absichtlich over- bzw. undersampling damit ich von jd. Distrikt etwa gleich viele habe.

- *Cluster Random Sampling* Datenmenge wird geclustert, es werden *zufällig* k Cluster gewählt, aus denen dann alle Datenpunkte als Samples genommen werden.

  ⊕ Im Erfolgsfall sehr gute Repräsentation von versch. mögl. Ausprägungen der Datenpunkte (Cluster)
  ⊕ Weitere samples sind einfach hinzuzunehmen *(Nehme mehr Cluster)*.
  ⊕ Evtl. sinnvoll und besonders einfach im geographischen Kontext.
  ⊖ Ergebnis hängt von der Güte der Cluster ab
  ⊖ Weniger repräsentativ für Verteilung der mögl. Ausprägungen (Cluster) über Gesamtpopulation.
  ⊖ Wahl der Cluster bringt Bias
  ⊖ Da Cluster potentiell von unterschiedlicher Größe ist auch die resultierende Menge der Samples unterschiedlich Groß je nach Wahl der Cluster.
  → Wenn Datenmenge sehr groß ist und sich gut clustern lässt.

Wenn die Teilmenge von Interesse sehr klein ist, sind manche Sampling-Techniken ggfs. nicht sinnvoll.

1.4.6 Principal Component Analysis (PCA)

1. Daten standardisieren, normalisieren
2. Covarianz-Matrix berechnen
3. Bereche Eigenvektoren und Eigenwerte der Cov-Matrix (bis zu d viele, char. Polynom d-ten Grades, wobei letzter allerdings nur noch einen Freiheitsgrad mehr hat, also nicht informativ)
4. Eigenvektoren, geordnet nach Eigenwert, bilden *Feature Vectors*, neue Basis sodass die Basisvektoren die Varianzen minimieren.
5. Neuen Datensatz ableiten.

Einsatz Die gefundenen Eigenvektoren bilden eine neue Basis des Datenraums und die entsprechenden Eigenwerte repräsentieren die Varianz in der jeweiligen Dimension. *Es kann also jeweils abgelesen werden, welche Dimension die Daten zu welchem Ausmaß erklärt": Ist die Varianz klein, so erklärt diese Dimension wenig*.
Anschaulich: Versuche, Ellipsoid an Daten anzupassen, Achsen dieses E. liefern dann PCs.

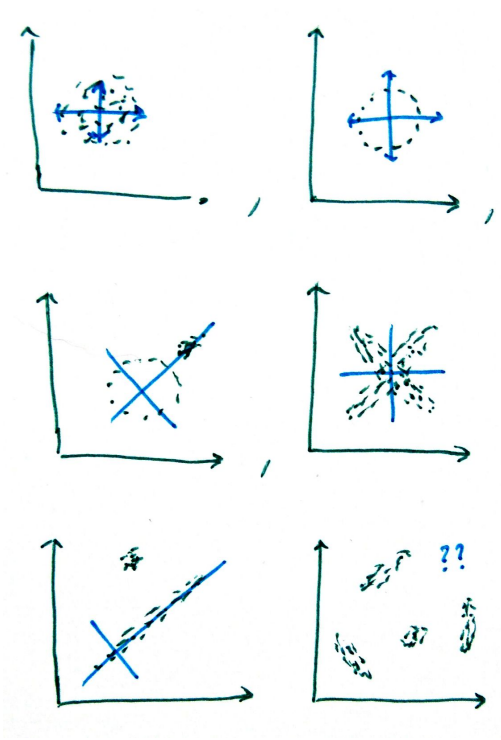
Zu beachten

- Principle Components (Eigenvektoren) werden nach Eigenwert sortiert. Eigenwert gibt Varianz an.
- PCs/EVs sind immer orthogonal.

- Empfindlich ggü. Outliern.
- Empfindlich ggü. Noise, denn ein kleiner Häufungspunkt kann ggf. eine Rotation verursachen.
- Robust ggü. Rotationen.

- Nur effektiv, wenn eine dominierende Korrelation im Datensatz vorhanden ist.
- Nur sinnvoll für numerische Daten (Berechnung von Covarianz, etc)
- Nur sinnvoll für globale Aussagen *(Temperatur/Luftfeuchtigkeit)*. Nicht sinnvoll, wenn ledigl. kleine Teilmengen interessant.
- Nicht sinnvoll wenn zB viele kleinere Cluster (cf. **??**)

- Ergebnis hängt von relativer Skalierung der Ausgangs-Dimensionen ab. Nicht klar zu sagen, welche da am Besten ist.
- Wie ist Ergebnis von PCA zu interpretieren? Achsen besagen ledglich PC1", PC2", ...
- Durch Weglassen von Dimension verliere ich Information.
- PCA selbst entfernt keine Dimension. Identifiert lediglich Principal Components.



Unterste Zeile: Links nicht korrekt, Achse ist an falscher Stelle. Rechts kein sinnvolles Ergebnis.

1.5 Begriffe

1.5.1 Clustering vs. Classification vs. Association Rules

Classification Die einzelnen Klassen sind im Voraus zur Analyse bereits definiert. Versuche, Daten-Objekte den vordefinierten Klassen *zuzuordnen*. *zB Studenten, Angestellte, Professoren, versuche dann, Personen einzuordnen.*

Clustering Die *Definition* der einzelnen Klassen ist im Vorfeld nicht bekannt. Versuche, Gruppierungen in Daten-Menge zu finden. *zB finde hochgradig aktive Internet-Nutzer: Clustere nach Datenvolumen und Zeit online; identifiziere Zielgruppen für Handytarife*

Association Rules Quasi Clustering von Bitvektoren (jd. kodiert einen Ëinkaufswagen") bzw. von Potenzmengen, suche ähnliche Teilmengen. *Habe idR gegeben eine Form von Warenkorb, d.h. Dinge, die miteinander in Beziehung stehen. Versuche, Regeln zu finden wie "A ist gegeben, dann ist auch B sehr wahrscheinlich"; zB Beziehungen zwischen Produkten in Supermarkt*.

Outlier Analysis Genaugenommen Sub-Problem von Clustering. *Versuche, Outlier zu identifieren, zB Verbrecher*

1.5.2 False positives vs. false negatives

False positive Algorithmus klassifiziert Objekt als zugehörig, ist es aber in Wirklichkeit nicht.

False negative Algorithmus klassifiziert Objekt als *nicht* zugehörig, dabei ist es tatsächlich zugehörig.

1.5.3 Supervised vs. unsupervised learning

Supervised learning Habe "Trainings"-Daten gegeben von denen die Klassen bekannt sind. Nutze diese als Grundlage für Algorithmus. *Beispiel: Neuronale Netze, Naive Bayes Classifier, Decision Trees*

Unsupervised learning Es wird nicht auf Trainingsdaten zugegriffen. *Beispiel: Clustering-Algorithmen*

1.5.4 Classification vs. Prediction

- ***Classification***: Will Klassen-Attribut für neue Werte vorhersagen
- ***Prediction***: Sage beliebigen numerischen Wert vorher. *zB Regression.*

1.5.5 Eager vs Lazy Learners

Lazy Learning Berechnung/Generalisierung wird erst dann angestellt, wenn Ergebnis (zB Klasse für Objekt) abgefragt wird (*zB k-nearest neighbour*)

Eager Learning Generalisierung wird bereits im Vorhinein durchgeführt (*zB Neuronale Netze, SVM*)

2 Classification

2.1 Evaluation of Classifiers

- **Speed**: *DecTrees, NNs Aufbau schwierig, Anwendung einfach. DecTrees aufbauen idR auch kein Problem heute.*
- **Robustness**: Algorithmus soll auf unbekannten Daten ja auch funktionieren.
  - Was geschieht bei sich widersprechenden Datensätzen? (*contradictory examples*)
  - Wie ändert sich Ergebnis, wenn Datensätze hinzugefügt oder weggelassen werden?
- **Scalability**: Komplexität; Dauer, Modell aufzubauen. *NNs idR nicht gut skalierbar da Trainingsaufwand hoch – Ausführung geht allerdings schnell und Genauigkeit recht hoch.*
- **Interpretierbarkeit**: Nachvollziehen, wie Entscheidung zustande kam; Vertrauen in Ergebnis/Algorithmus.

2.1.1 Validation

todo

2.1.2 Accuracy & Error

todo

2.1.3 Precision & Recall

todo

2.2 Beispiele für Classifier

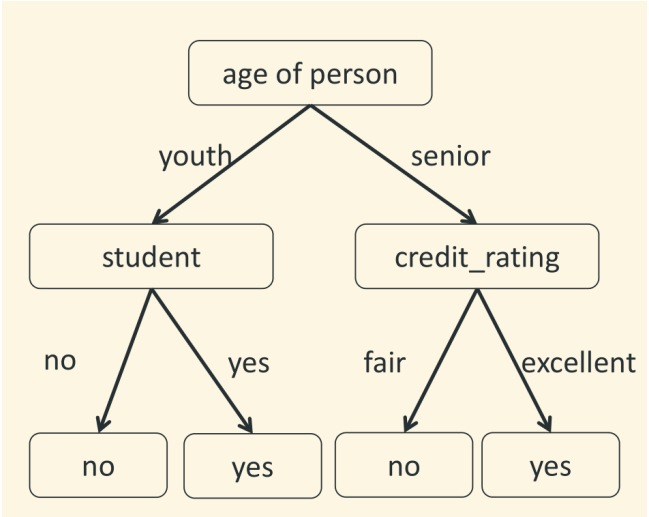
- Kreditwürdig oder nicht?
- Spam – Ja/Nein?
- Klassifizierung von Text – Sport/Politik/...?
- Wahl-o-mat

2.3 Decision Trees

2.3.1 Konstruktion

1. Finde ein Attribut, nach dem der Datensatz "am Besten" aufgeteilt werden kann
2. Nehme diesen Knoten als Wurzel, für jede Ausprägung des Attributs erzeuge ein Kind
3. Wiederhole iterativ bis Abbruchkriterium erreicht (zB gewünschte Reinheit eines Blattknotens).

Konstruktion ist also typischer Divide & Conquer-Algorithmus



Zu treffende Entscheidung: Soll Kredit gewährt werden?

2.3.2 Wahl des Split-Attributs

Möchte möglichst bald erreichen, dass Attribute unterhalb eines Knotens nur dieselbe Klasse enthalten. Suche also ein Maß der Purityünd stelle dann für jeden möglichen Split den Ìnformation Gainfest. Hierfür gibt es zwei Maße: *Information Gain* und *Gini Impurity*.

2.3.3 Information Gain & GainRatio

1. Berechne "Reinheit" von aktuellem Knoten als

$$info(D) := -\sum_{i=1}^{m} p_i \log_2(p_i)$$

   für m Klassen wobei p_i Anteil der Tupel mit Klasse i. *Klein ist rein*.

2. Für jedes Attribut A in D, berechne

$$gain(A) := info(D) - info_A(D)$$

   wobei

$$info_A(D) := \sum_{j=1}^{v} \frac{|D_j|}{|D|} \cdot info(D_{A,j})$$

   mit D_j als Menge der Tupel mit Ausprägung j von Attribut A.

3. Wähle Attribut mit höchstem *gain* für Split. Jede Ausprägung des gewählten Attributs erzeugt also ein neues Kind.

4. Rekursiv weiter bis verbleibende Tupel alle in selber Klasse oder anderes Abbruchkriterium erreicht.

Bei numerischen Attributen prüfe alle möglichen Aufteilungen zwischen zwei diskreten Werten auf ihren *information gain* (möglicherweise sehr aufwendig).

Zu beachten Bei im Extremfall für jedes Tupel individuelle Attribute (zB *uniqueID*) sollten diese entweder weggelassen werden, oder *GainRatio* angewandt werden (dieser wäre dann sehr klein).

GainRatio Teile den ursprünglichen information gain durch den Informationsgehalt des Attributs.

$$gainRatio(A) = \frac{gain(A)}{splitInfo(A)}$$

$$splitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \cdot \log_2\left(\frac{|D_j|}{|D|}\right)$$

Wähle höchsten.

2.3.4 Gini Impurity

Idee Mache lediglich binäre Splits. Identifiziere Teilmengen von Ausprägungen, sodass binärer Split auf zugehörigkeit zu Teilmenge (*subset-split*) möglichst rein.

1. Berechne $\text{gini}(D) := \sum_{c \in \text{Classes}} {p_c}^2$
2. Für jedes Attribut A
   (a) Berechne bestmöglichen GiniGain(A) erreichbar durch einen subset-Split a.

   $$\text{giniGain(A)} := \text{gini}(D) - \min_{A \in \text{Attribute}} \min_{a \in \mathcal{P}(A)} \{\text{gini}_a\}$$

   Überprüfe dafür alle möglichen subset-splits bzgl. A. Sei $a \in \mathcal{P}(A)$. Dann

   $$\text{gini}_a := \frac{|D_a|}{|D|}\text{gini}(D_a) + \frac{|\overline{D_a}|}{|\overline{D}|}\text{gini}(\overline{D_a})$$

   , wobei D_a Menge der Datentupel mit Ausprägung a von Attribut A und $\overline{D_a} := D \backslash D_a$. *gewichtete Summe über resultierende Blätter bei diesem (binären) Split.*

Zu beachten Nur binäre Splits

2.3.5 Pruning

Overfitting vermeiden, Baum soll nicht zu groß werden. Standard-Algorithmus wie gegeben würde immer overfitten.

Pre-Pruning lege Abbruchkriterien fest:

- ***Minimal Support***: Minimale Anzahl von Datentupeln in einem Blatt
- ***Minimum Confidence***: Minimaler Anteil von größter Klasse in Knoten *(zB Ende, falls 95 % Ja)*

Post-Pruning modifiziere fertigen Baum:

- Fasse Knoten zusammen wenn Fehler bei Crossvalidation zu groß wird
- Maß für Cost Complexityfür Entfernen von Teilbäumen.

2.3.6 Diskussion

⊕ Leicht nachzuvollziehen, gute Interpretierbarkeit

⊕ Kann numerische und kategorische Daten handhaben

⊕ Robust

⊕ Schnell aufzubauen

⊖ Wiederholung von Splits an gleichem Attribut in unterschiedlichen Teilbäumen

⊖ Große Bäume sind schwer nachzuvollziehen

⊖ Overfitting

2.4 Naive Bayes Classifier

Gibt nicht lediglich eine Klasse an sondern Wahrscheinlichkeiten, mit welcher das Tupel zu jeder Klasse gehört.

Herleitung Sei $\boldsymbol{x} = (x_1, x_2, ..., x_k)$ ein Datentupel bzw. das Ereignis, dass das momentan betrachtete Tupel genau so aussieht. Sei C_i das Ereignis, dass ein besagtes Tupel in Klasse i liegt. Sei $P(C_i|\boldsymbol{x})$ die Wahrscheinlichkeit, dass das Tupel in C_i liegt, gegeben, dass es wie $\boldsymbol{x}$ aussieht. (Dann ist $P(\boldsymbol{x}|C_i)$ die Wsl., dass ein Tupel wie $\boldsymbol{x}$ aussieht, gegeben, dass es in Klasse C_i liegt.)
Bayes' Theorem besagt:

$$P(C_i|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_i) \cdot P(C_i)}{P(\boldsymbol{x})}$$

Folgt aus Definition von Conditional Probability.
Für die Klassifizierung ist der Zähler irrelevant, da die Klasse i gewählt wird, für die $P(C_i|\boldsymbol{x})$ am größten ist und der Zähler nicht von C_i abhängt.
Weiterhin ist $P(C_i)$ als für alle Klassen gleich angenommen (sofern nicht bekannt). Es dreht sich also nur noch um $P(\boldsymbol{x}|C_i)$.
Unter Annahme, dass alle $P(x_i|C)$ voneinander unabhängig sind, können wir schreiben:

$$P(\boldsymbol{x}|C_i) = P(x_1|C_i) \cdot P(x_2|C_i) \cdot ...$$

Dieser Wert ist bekannt: Anzahl der Tupel mit Klasse C_i, die den Wert x_k für das entsprechende Attribut haben, geteilt durch die Gesamtzahl der Tupel von Klasse C_i.

Vorgehen Für ein gegebenes Tupel $\boldsymbol{x}$

1. Für alle Klassen i:
   (a) Berechne $P(C_i)$
   (b) Berechne $P(\boldsymbol{x}|C_i)$ für jedes Attribut von $\boldsymbol{x}$ unter Verwendung von $P(\boldsymbol{x}|C_i) = P(x_1|C_i) \cdot P(x_2|C_i) \cdot ....$ Dabei ist

   $$P(x_j|C_i) := \frac{P(x_j \cap C_i)}{P(C_i)}$$

(c) Berechne $P(\boldsymbol{x}|C_i) \cdot P(C_i)$

2. Weise Wahrscheinlichkeiten je Cluster zu, bzw. wähle Klasse mit höchstem Wert für $P(\boldsymbol{x}|C_i) \cdot P(C_i)$

Zu beachten

- Wenn eines der $P(x_j|C_i)$ gleich 0 ist (d.h. wenn es kein einziges entsprechendes Beispiel in den Trainingsdaten gibt) ist das ganze Produkt gleich 0. Abhilfe: weglassen.
- **Annahmen**
  - Jede Ausprägung eines Attributs ist unabhängig von Ausprägungen der anderen Attribute. *Jedes der Attribute trägt unabhängig von den anderen zur Zugehörigkeit der Klasse bei, unterschlägt mögliche Korrelationen zwischen Attributen.*. Falls es Korrelationen/Abhängigkeiten zwischen Attributen gibt *(wie zB Raucher/Lungenkrebs, Alter/Einkommen)*, so werden diese hier nicht genutzt. *zB Raucher, Lungenkrebs, Herzprobleme, ...*
  - $P(C_i)$ (class prior probabilities) werden, insofern nicht bekannt, als pw. gleich angenommen.

2.4.1 Bayesian Belief Networks

2.5 Support Vector Machines

Idee: Finde Hyperplane in Raum, die Klassen bestmöglichst trennt.

Support Vector Trainings-Datenpunkte, die genau auf den Rändern der bestmöglichst trennenden Hyperplane liegen. Hierbei bedeutet "bestmöglichst trennen": Mit dem größten kürzesten Abstand zu einem Trainings-Datenpunkt.

⊕ Findet immer globale Lösung
⊕ wenig anfällig für Overfitting, da Ergebnis nur basierend auf den Support Vectors
⊕ Skaliert gut mit hochdimensionalen Daten da Komplexität von Anzahl der Support Vectors abhängt und nicht von Anzahl Dimensionen.
⊕ kann auch für Prediction genutzt werden
⊕ Gute Accuracy.
⊖ langsames Training
⊖ Modell ist schwer interpretierbar.
⊖ Gute Ergebnisse oft erst mit *Kernel Trick*.

2.5.1 Non-linear SVMs

Überführe Eingabedaten in höherdimensionalen Raum (mittels *Kernel function*) und finde dort eine trennende Hyperplane. Deren Oberfläche trennt dann die Datenpunkte im ursprünglichen Raum.

⊖ welches Mapping / Kernel ist am Besten geeignet?

2.6 Neural Networks

Multilayer Feed-Forward Neural Network besteht aus

- *Input Layer* (normalisiert auf $[0, 1]$)
- Einem oder mehreren *Hidden Layers*
- Gewichteten Kanten zwischen Knoten (falls *fully-connected NN* sind alle Knoten aus einem Layer mit allen aus dem nächsten Verbunden)
- Aktivierungsfunktionen für jeden Knoten.

Backpropagation

1. Initialisiere Gewichte (zB zufällig)
2. Propagiere Input nach vorn
3. Stelle Fehler fest, propagiere Fehler zurück, passe Gewichte an um Fehler zu minimieren
4. Wiederhole bis keine/kaum Verbessung.

Learning Rate Faktor aus $[0, 1]$ für Änderung in Gewicht bei Backpropagation. Vermeide feststecken in localem Maximum. Zu niedrig eingestellt: Training konvergiert sehr langsam; zu hoch eingestellt: Oszilliert zwischen nicht ausreichenden Lösungen.

- ⊕ Tolerant ggü. Noise
- ⊕ Gut geeignet für numerische/reellwertige Ein- und *Aus*-gaben
- ⊕ Gut parallelisierbar
- ⊖ Lange Trainingsdauer
- ⊖ Brauche viele Trainingsdaten
- ⊖ Muss Parameter wie zB Netzwerktopologie festlegen
- ⊖ Sehr schwer nachzuvollziehen

2.7 k-Nearest Neighbour

Klasse ergibt sich aus Mehrheitsentscheidung der k nächstliegenden Punkte.

- ⊖ Noisy oder irrelevante Features/Dimensionen können Ergebnis stark negativ beeinflussen (dann liegen womöglich eigentlich ähnliche Punkte in dieser Dimension weit auseinander)
- ⊖ Wenn Klassen unterschiedlich stark vertreten sind, ist für Klassifizierung eines neuen Punktes die besser vertretene Klasse wahrscheinlicher (Abhilfe: Gewichtung nach Inversem von Distanz bei Majority Vote).
- ⊖ Naive Implementation ist rechenaufwendig.

Beispiel für *Lazy Learner*.

3 Clustering

Güte/Kompaktheit eines Clusterings Innerhalb von Cluster Datenpunkte möglichst ähnlich, zwischen Clustern möglichst verschieden.

TD Sei m_c der repräsentative Medoid für das Cluster c, d eine gegebene Metrik. Definiere für die Güte eines einzelenen Clusters

$$TD(C) = \sum_{p \in C} d(p, m_c)$$

Summe über Distanzen aller anderen Punkte aus Cluster zu Medoid und für die Güte eines gesamten Clusterings

$$TD = \sum_{c \in C} TD(c)$$

Summe über alle Cluster-Kosten

- ⊖ Hängt von Anzahl der Cluster, also gegebenem k ab! Beispiel

Silhouette Coefficient Sei

- $a(o)$ – Distanz von o zu Cluster-Repräsentant
- $b(o)$ – kleinste Distanz von o zu Repräsentant von einem anderen Cluster

. Dann ist die Silhouette von o definiert als:

$$\frac{b(o) - a(o)}{\max\{a(o), b(o)\}}$$

Es gilt $-1 \leq s(o) \leq +1$, höherer Wert ist besser.

Einsatzbereiche

- Identifikation/Bereinigung von Rauschen in Daten
- Reduktion von Daten
- Outlier Detection
- Verstehen der Daten foo bar anhand der inneliegenden Klassen
- Identifizieren von zusammengehörigen Teilmengen/Klassen

Distanzfunktionen Maß der Ähnlichkeit für zwei Datenpunkte. Entspricht einer Metrik. Beispielsweise:

- lp-Metrik
  - $p = 1$: ***Manhattan Distance*** *Denke: Nur Schritte entlang Gitter in x/y-Richtung, keine Diagonalen.*
  - $p = 2$: ***Euklidean Distance*** *Denke: Luftlinie.*
- *term frequency* und *inverted document frequency*.

3.1 Partitioning Methods

Vorgehen ist eher *top-down*.

3.1.1 k-Means Clustering

Parameter

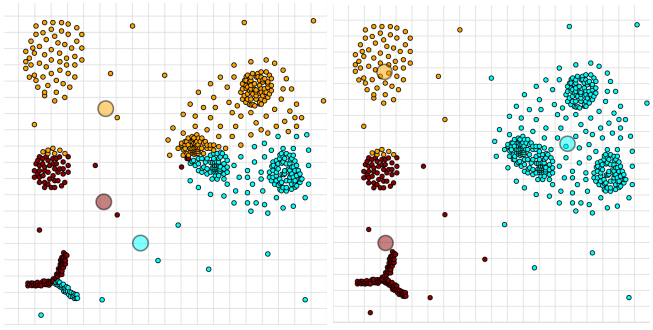
- k - Anzahl von Centroiden *d.h. Anzahl resultierender Cluster*
- (Metrik)

Ausgabe

- Zuweisung für jeden Punkt zu einem Cluster/Centroid (*Voronoi-Partitionierung der Datenmenge*)

Algorithm 1: k-means

1 Wähle k Punkte (zufällig) als initiale Centroide ;
2 **repeat**
3 Weise jeden Datenpunkt dem Cluster zu, wo der entsprechende Centroid am nähesten liegt ;
4 Aktualisiere Position der Centroide als Durchschnitt aller momentan zugewiesenen Punkte ;
5 **until** *Keine Verbessung*;



Schritt 1 Fertig

Beurteilung

⊕ **Speed**: Schnell ($\mathcal{O}(t \cdot k \cdot n)$) wobei t Anzahl Iterationen.
⊕ Einfach zu implementieren
⊖ Benötigt Parameter k, d.h. Anzahl Cluster muss vorher bekannt sein.
⊖ **Noise, Outlier**:
  – verzerren sehr stark die Berechnung der Durchschnitts-Centroide.
  – Noise-Punkte werden auch klassifiziert, keine Unterscheidung zwischen Daten- und Noise-Punkten
⊖ **Cluster Shape**: Kann nur Cluster identifizieren, die in Voronoi-Partitionierung passen, d.h. kann keine nicht-konvexen Formen erkennen.
⊖ Ergebnis abhängig von initaler Wahl der k Centroide, nicht deterministisch
⊖ Terminiert lediglich bei lokalem Maximum.

3.1.2 ISODATA

Erweiterung von k-Means. Cluster können aufgeteilt oder zusammengeführt werden, falls Constraints überschritten werden.

Parameter

- k: initiale Anzahl Centroide / Cluster
- `maxIter` Beschränkung für Anzahl Iterationen
- `MinPts` Mindestanzahl Punkte für ein valides Cluster
- `MinDist` Minimale Distanz zwischen zwei Clustern vor merge.
- `StdDev` Maximale Standardabweichung für Werte in Cluster bevor split.

Algorithmus Zunächst ähnlich wie k-Means. In Schleife zusätzlich:

- Cluster werden *zusammengeführt*, falls
  - Größe des Clusters kleiner als `MinPts`
  - Zwei Cluster sind einander näher als `MinDist`
- Cluster werden *getrennt*, falls
  - Standardabweichung des Clusters übersteigt `StdDev` *und* Größe des Clusters ist größer als $2 \cdot$ `MinPts`.

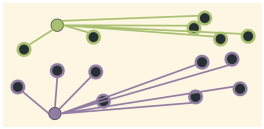
Beurteilung

⊕ Kann eine variable Anzahl Cluster erkennen.
⊖ Noch mehr Parameter nötig
- cf. k-Means.

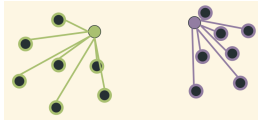
3.1.3 k-Medoids (PAM, Clarans)

Um Problem mit Outliern abzuhelfen, nutze statt Durchschnitts-Centroiden etwas anderes.
Im Unterschied zu k-Means (i-welche Werte) nutzt k-Medoids tatsächliche Datenpunkte als Cluster-Zentren.

Medoid Repräsentant eines Clusters, der am zentralsten gelegen ist in diesem Cluster. Frage, wie Medoid gewählt wird führt zu PAM.



Nicht so gut.



Besser

Parameter

- k – Anzahl Cluster

Vergleich

⊖ Muss k angeben

⊖ Kann nur konvexe Cluster finden.

Nun stellt sich die Frage, wie genau wir Medoide feststellen und Cluster zuweisen. Dafür gibt es verschiedene Herangehensweisen:

PAM *Paritioning around Medoids*

1 Initalisiere k Punkte als Medoide ;
2 Weise jeden Punkt dem Medoid zu, der anhand gegebener Metrik am Nähesten liegt. ;
3 **while** *Kosten des momentanen Zustands verringen sich* **do**
4 **foreach** *Medoid* m, *Datenpunkt* o **do**
5 Tausche m und o ;
6 Weise Punkte neu dem nächsten Medoid zu ;
7 Berechne neue Kosten ;
8 **if** *Kosten größer* **then**
9 Mache Tausch rückgängig.
10 **end**
11 **end**
12 **end**

$\mathcal{O}(k(n-k)^2)$ in jeder Iteration.

Clarans *Clustering Large Applications*
Im Prinzip gleich wie PAM, aber:

- Es werden nicht alle anderen Punkte als potentielle neue Medoide überprüft sondern es werden

Zusätzliche Parameter:

- `maxIts` – maximale Anzahl Iterationen *Komplette Neuversuche"des Algorithmus*
- `maxNeighs` – maximale Anzahl von betrachteten Nachbarn *potentiellen Alternativen für einen Medoid*

Algorithm 2: Clarans

// Komplette Neuversuche
for r *from* 1 *to* $maxIts$ **do**
  choose k objects as Medoids randomly ;
  $i \leftarrow 0$;
  while $i < maxNeigh$ **do**
  choose randomly a tuple of medoid and non-medoid (M, N) ;
  if $TD_{NM} < TD$ **then**
  /* better score after swap */
  $M \leftarrow N$ /* assign new medoid */
  $TD \leftarrow TD_{NM}$ /* assign new score */
  $i \leftarrow 0$ /* have to start over with neighbours */
  end
  else
  $i \leftarrow i + 1$ /* proceed */
  end
  end
  If better, update TD_{best} with TD ;
  Record the current medoids;
end
return current medoids

⊕ Schneller als PAM

⊕ `maxIts`, `maxNeighs` schränken Laufzeit ein

⊖ Mehr Parameter zu wählen

⊖ Ergebnis hängt von der Sampling-Methode ab

Beurteilung *k-medoids und Varianten* Im Wesentlichen ähnlich zu *k-means*.

- **Noise, Outlier**: Durch Verwendung von Medoiden (echten Datenpunkten) wird Empfindlichkeit ggü Outliern bzw. ungleich verteiltem Noise abgeschwächt im Vergleich zu *k-means* (*cf 3.1.1*).
- **Cluster Shape**: Nur konvexe Cluster
- **Dichte**: Unterschiedl. dichte Cluster verzerren ebenfalls Zuweisung der Centroide (*nach Def. von TD, wenn ich es richtig verstehe*).

3.1.4 Expectation Maximisation

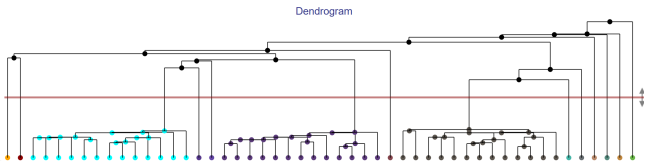
Algorithm 3: Expectation Maximisation

1 Randomly position k Gaussian distributions ;
2 Assign data points to the cluster distributions (functions) ;
3 Re-estimate mean and variance of the distributions ;
4 Repeat until no improvement ;

- **Noise, Outlier**: können Bild verzerren
- **Cluster Shape**: Lediglich konvexe Cluster
- **Dichte**: Kann gut mit unterschiedl. dichten Clustern umgehen.
- **Hierarchie**: Keine.

⊕ Liefert Wahrscheinlichkeit für Klasse, Mehrdeutigkeiten sind somit sichtbar.

3.2 Hierarchical & Linkage-based methods



liefert Dendrogramm. (Hierarchisches Clustering)

⊕ **Parameter**: Keine

⊕ **Cluster Shape**: beliebig, solange gut dichte-separiert.

⊕ **Hierarchie**: Kann hierarchische Cluster-Strukturen abbilden

⊖ **Noise**:

- Noise-Punkte werden nicht als solche erkannt
- Noise-Punkte können Dendrogramm unübersichtlich machen

⊖ Muss immernoch Clustering aus Dendrogramm herauslesen.

⊖ **Speed**: Ineffizient, in $\mathcal{O}(n^2)$. *Centroid Linkage* noch am schnellsten.

Algorithm 4: Linkage-based Clustering

1 Jeder Datenpunkt ist zunächst einzelnes Cluster. Berechne Distanzen zwischen jedem Paar von Clustern. $\mathcal{O}(n^2)$;
2 In jedem Schritt werden die beiden Cluster mit der minimalen Distanz vereint. $\mathcal{O}(n)$;
3 Berechne Distanzen neu. ;
4 Ende, falls nur noch ein Cluster vorhanden.

BIRCH Nutze Zusammenfassungen (*clustering features*, *CF*) von statistischen Werten um Mikro-Cluster zu beschreiben. Bei Merge von zwei Mikro-Clustern kann ein neues *CF*-Tupel einfach aus den direkten Kindern berechnet werden.

CF-Tree

- (Ähnlich wie B-Baum) Einschränkung vom Anzahl Schlüsseln in Knoten
- Füge nacheinander *CF*-Tupel ein, baue so Baum auf. Dabei darf Durchmesser (bezüglich eines Distanzmaßes) aller Punkte in einem Blattknoten nicht einen Schwellwert überschreiten.

Findet auf diese Weise konvexe Cluster.

Linkage-based Clustering Fasse iterativ zwei Punkte mit minimaler Distanz zusammen, baue so ein Dendrogramm auf.

⊕ keine Parameter

⊕ findet auch nicht-konvexe Cluster

⊖ muss passendes Distanzmaß finden

⊖ Ergebnis unklar falls nicht klar dichte-separiert

Method of Wishart Define a density around data points. All other points are regarded as noise and not considered for linkage-based clustering later on.

Mögliche Distanzmaße :

- ***Single Linkage***: *Minimum*

$$d(C_1, C_2) = \min_{p \in C_1, q \in C_2} (d(p, q))$$

⊕ Gut auch bei zB Snake Data wenn doch noch dichteseparierte.

⊖ Noise-Punkte können "Brücke"bilden und somit Dendrogramm schwer interpretierbar machen (*cf Educlust*)

- ***Complete Linkage***: *Maximum*

$$d(C_1, C_2) = \max_{p \in C_1, q \in C_2} (d(p, q))$$

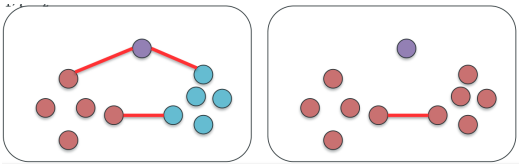
⊖ Nicht gut bei nicht-konvexen verschlungen Clustern *zB Snake Data*

- ***Centroid Linkage***: *Durchschnitt*

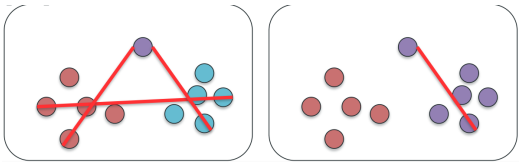
$$d(C_1, C_2) = d(\text{mean}(C_1), \text{mean}(C_2))$$

⊖ Nicht gut bei nicht-konvexen verschlungen Clustern *zB Snake Data* da eher runde Gruppen zusammengefasst werden.

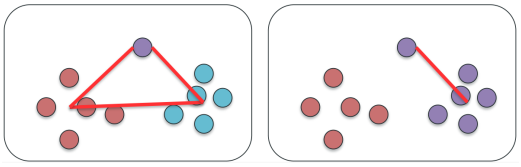
- ***Ward's Method*** *Error sum-of-squares*: $\sum D(x, \mu)^2$



Single Linkage



Complete Linkage



Centroid Linkage

3.3 Density-based Methods

3.3.1 Definitions

Core Object Objekt $o \in O$ ist *Kern-Objekt* in O gdw.

$$|N_\varepsilon(o)| \geq MinPts$$

, wobei $|N_\varepsilon(o)| = \{ o' \in O \mid d(o, o') \leq \varepsilon \}$ (Beachte: $o \in N_\varepsilon(o)$)

directly density reachable $p \in O$ ist *direkt dichte-erreichbar* von $q \in O$ (w.r.t ε und $MinPts$) gdw.

- $p \in N_\varepsilon(q)$
- q ist Kern-Objekt

density reachable p ist *dichte-erreichbar* von q, falls es eine Kette von direkt dichte-erreichbaren Objekten zwischen q und p gibt.

density-connected p und q sind *dichte-verbunden*, wenn beide von einem dritten Objekt o dichte-erreichbar sind.

border object $o \in O$ ist *Rand-Objekt* in O gdw.

- o ist kein Kern-Objekt
- o ist dichte-erreichbar von einem anderen Objekt p

Cluster w.r.t. ε $MinPts$ ist eine nichtleere Teilmenge von O mit

- **Maximalität**:

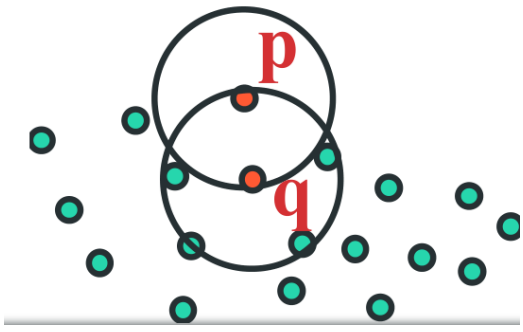
  $\forall p, q \in O : p \in C \wedge p$ density-reachable from $p \Rightarrow q \in C$

- **Konnektivität**:

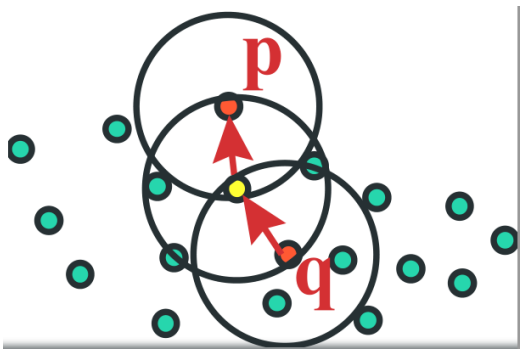
  $\forall p, q \in C : p$ ist dichte-verbunden mit q

Clustering besteht aus...

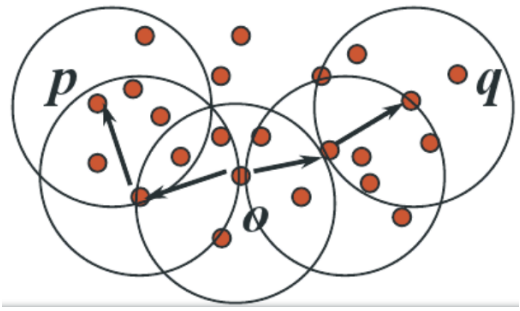
- Menge aller Cluster
- Menge von Noise-Punkten (die, die keinem Cluster zugewiesen wurden.)



directly density reachable



density reachable



density connected

3.3.2 DBSCAN

Parameter

- ε – Radius für Nachbarschaft
- *MinPoints* – Mindestanzahl in Nachbarschaft

Algorithm 5: DBSCAN

```
1  Wähle bisher noch nicht betrachteten Punkt P ;
2  Finde alle Punkte, die von P aus
     dichte-erreichbar sind ;
3  if P ist Kern-Objekt then
4  |   Erstelle neues Cluster // mit Punkten
5  else
6  |   Klassifiziere P als Noise ;
   |   /* Punkte können später noch einem Cluster zugewiesen
   |      werden                                          */
7  end
8  Finde alle von der Nachbarschaft von P aus
     dichte-erreichbaren Punkte und weise sie dem
     momentanen Cluster zu. ;
9  if Ein Punkt r ist Randobjekt then
   |   // Keine Punkte sind von r aus dichte-erreichbar
10 |   Gehe zu Schritt 1
11 end
```

⊕ Kann gut nicht-konvexe Cluster abbilden (zB Snake Data).

⊕ Muss Anzahl Cluster nicht vorgeben.

⊕ Kann Noise unterscheiden.

⊖ Kein gutes Ergebnis falls unterschiedliche Cluster unterschiedlich dicht (*bräuchte dann unterschiedl. Parameter*)

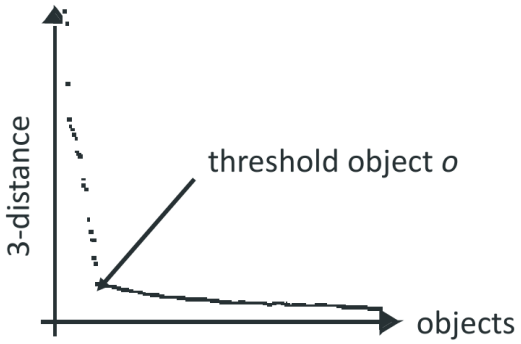
⊖ liefert keine Hierarchie

⊖ Sehr empfindlich ggü. Parametern ε und `minPts`. Beispiel: *todo*

k-distance Diagram hilft beim Finden von ε, *MinPts*:

- *k-Distanz*: Abstand eines Objektes zu seinem k-nächsten Nachbarn.
- Faustregel: $MinPts \leftarrow k + 1$
- Falls Cluster und Noise gut dichteseparierter, so ergibt sich in Plot zwischen k-Distanz und Objekten ein Knick, wähle diese Distanz als ε.

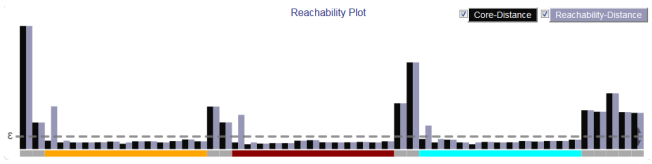
⊖ Falls nicht gut dichteseparierter /hierarchisch ergibt sich auch kein gut erkennbarer Knick in Plot (→ *OPTICS*)



k-distance diag.

3.3.3 OPTICS

variable epsilons liefert reachability-plot



Reachability Plot

Core Distance *wird in EduClust, falls undefiniert, auf den gleichen Wert als die CoreDist gesetzt.*

$$CoreDistance_{\varepsilon,minPoints}(o) = \begin{cases} MinPtsDistance(o) & \text{iff } o \text{ is core object} \\ undefined & \text{else} \end{cases}$$

wobei *MinPtsDistance* der Radius, sodass *MinPts* viele Punkte inneliegend. Ist *undefined* wenn kein Kern-Objekt.

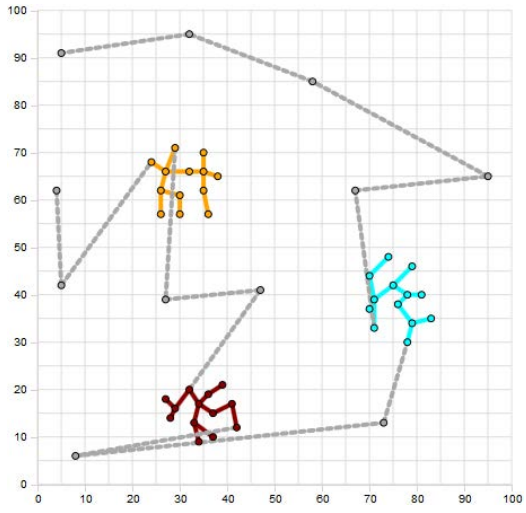
Reachability Distance *In OPTICS ist o das "vorherige" Objekt und p der neu hinzukommende Punkt. In EduClust wird bei "Schritt zu Noise-Punkt", d.h. neuer Iteration der `while`-Schleife trotzdem der zuletzt betrachtete Punkt für ReachabilityDist herangezogen (obwohl kein Kernobjekt), $CoreDist$ wird dann dem gleichgesetzt da eigentlich undefiniert.*

$$ReachabilityDist_{\varepsilon,minPoints}(p,o) = \begin{cases} \max\{\ CoreDist(o),\ d(o,p)\ \} & \text{iff } o \text{ is core object} \\ undefined & \text{else} \end{cases}$$

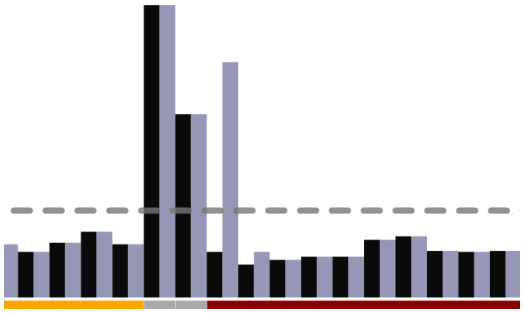
Algorithm 6: OPTICS

```
1  Set all points as unprocessed. ;
2  Insert random unprocessed point into ControlList ;
3  while ControlList not empty do
4      Select first element o from ControlList ;
5      determine N_ε(o) and CoreDist(o) ;
6      set o as processed ;
7      write (o, rdist, cdist) to file;
       // these are technically undefined for very first elem
8      if o is core-object then
           // Falls Kernobjekt: schaue Nachbarn an
9          foreach p ∈ N_ε(o) not yet processed do
10             rdistO ← ReachabilityDist(p, o) ;
11             insert (p, rdistO, cdist) into ControlList
                 or update if rdistO is smaller.
12         end
13     end
14 end
```

Erzeugt praktisch lediglich Plot aus Reachability Distance und Core Distance



Möglicher Verlauf



Bei erstem Punkt in neuem Cluster ist *Core Distance* niedrig (da Kernobjekt), *Reachability Distance* aber hoch, da dies hier die Distanz zum vorher betrachteten Punkt ist. – Bei einem Schritt zwischen zwei Nicht-Kernobjekten (neue `while`-Iterationen) wird *R.d* und *C.d* willkürlich auf deren Distanz gesetzt.

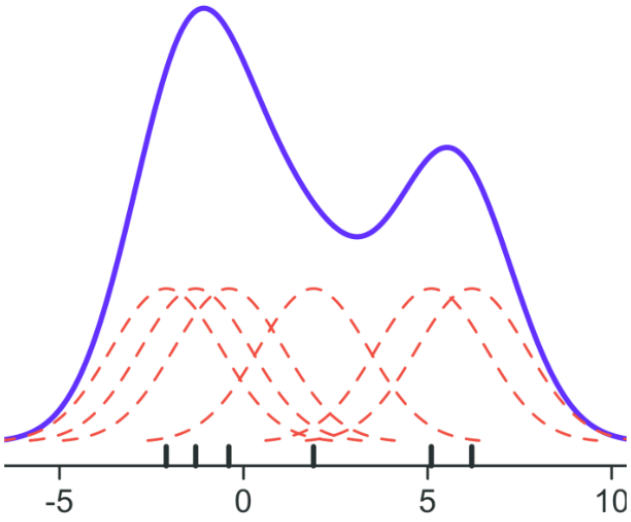
- ⊕ Kann ε interaktiv anpasen
- ⊕ Reachability Plot liefert Informationen über Cluster-Hierarchien
- ⊖ Findet selbst keine Cluster, liefert lediglich Reachability Plot

---

3.3.4 DENCLUE

Jeder Punkt erhält eine *influence function*, die den Einfluss umliegender Punkte auf ihn angibt (wsl. in Abhängigkeit der Distanz).
Die Summe über die *influence functions* spiegelt die Dichte im Datenraum wieder (*density function*).
Identifiziere nun *density attractors* (*"Spitzen"* der Berge).
y ist zu $x*$ *density-attracted*, falls eine Folge $y, x_1, ..., x*$ existiert sodass die Steigung von x_{i-1} in Richtung x_i geht. Punkte um einen *density attractor* bilden ein Cluster (siehe unten).
Mit gegebenem Threshold ζ:

1. ***Center-defined Cluster*** um $x*$: Punkte, die *density-attracted* zu $x*$ sind und wo die *density function* nicht unterhalb von ζ kommt.
2. ***Arbitrary-Shape Cluster***: Menge von Center-defined Clustern wobei zwischen jedem *density attractor* ein Pfad von Punkten mit Dichte-Wert oberhalb von ζ existiert.



Addition von *influence functions*.

3.4 Beispiele

Unklar abgegrenzte Cluster, Gaußsche Cluster

k-means Gutes Ergebnis, Noise-Punkte dazwischen werden aber auch klassifiziert.

EM Womöglich wegen Noise kaum erkennbares Ergebnis, da die Gauß-Kurven sehr ähnlich zueinander werden. Falls $k = 3$ gewählt könnten sich die Cluster mit kleinem Unterschied herausbilden.

Linkage Cluster sind in Dendrogramm erkennbar, eventuell aber keine gute Abgrenzung zu finden.

DBSCAN Keine guten Ergebnisse, da Punkte nicht gut dichteseparierbar. Hängt hier sehr von der perfekten Wahl der Parameter ab (wsl. insbesondere ε).

OPTICS Reachability Plot wird nur unklare Abgrenzugen liefern, manuelles Herantasten an ε könnte für gegebenen Datensatz gut helfen, könnte aber auch Overfitting darstellen.

DENCLUE Hier auch nicht besser als EM.

Cluster verschiedener Dichte

k-means Sehr dichtes Cluster würde Zuweisung der Centroide verzerren, d.h. diese würden alle Stark in Richtung dieses Clusters tendieren. Im Extremfall würden sich alle Centroide dort sammeln und ein Punkt die restlichen Cluster abdecken. Mit höherem k wird die Chance erhöht, dass Centroide in undichtere Cluster fallen, dann eher noch eine Chance mit Nachverarbeitung.

EM Gute Ergebnisse insofern k korrekt gewählt da sich Gauß-Kurven eindeutig in den Clustern konzentrieren und aufsummieren. Noise-Punkte sind klar unterscheidbar, da dort niedrige Werte.

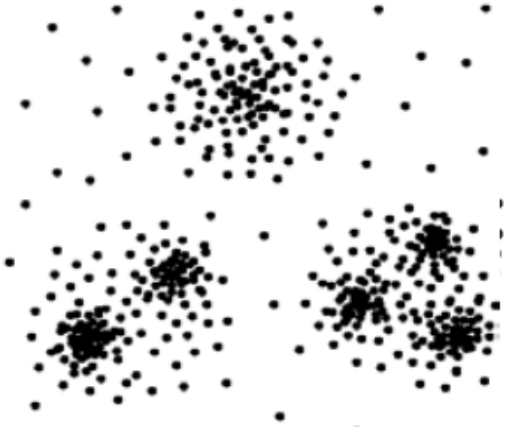
Linkage Für dichte Cluster sehr gute Ergebnisse. Noise allerdings womöglich nicht gut von oberem Ende von Dendrogramm abgrenzbar.

DBSCAN Sehr gute Ergebnisse für dichte Cluster, mit passender Wahl von ε auch das dünne Cluster da noch Unterschied besteht. Treffender Parameter könnte mit *k-distance diagram* wsl. gefunden werden.

OPTICS Sehr gutes Ergebnis, Abgrenzung zwischen dünnem Cluster und Noise auffindbar.

DENCLUE Gleich wie EM?

Snakey Data



Hierarchische Cluster

k-means Kann kein sinnvolles Clustering liefern da k-means lediglich Vornoi-Partitionierung von Raum liefert, hier sind Cluster aber nicht-konvex.

EM Kein sinnvolles Ergebnis, da sich die k Gaußkurven irgendwie an ihr Umfeld anpassen würden. EM kann nur konvexe Cluster erkennen.

Linkage Gute Ergebnisse, einzelne Cluster am oberen Ende des Dendrogramms abzulesen.

DBSCAN Gute Ergebnisse mit richtiger Wahl von ε, *MinPts*

OPTICS Gute Ergebnisse, Unterscheidung zwischen Clustern erkennbar in *Reachability Plot*.

DENCLUE Keine Guten Ergebnisse, kann nur konvexe Cluster.

k-means Wenn Cluster jeweils ähnlich in Dichte könnte mit passendem k ein gutes Ergebnis erzielt werden. Bekomme aber keine Informationen über Hierarchie.

EM Erkennbar mit passender Wahl von k.

Linkage Gute Ergebnisse, insbesondere hierarchische Struktur erkennbar.

DBSCAN Je nach Wahl von Parametern werden second-level-Cluster entweder als Noise klassifiziert oder alle Punkte beider Ebenen bilden ein Cluster.

OPTICS In Reachability-Plot womöglich Hierarchie erkennbar.

DENCLUE Mit Variante *multi-center-defined cluster* und richtiger Wahl von ξ könnten dünnere Cluster zusammen mit Spitzen als eines klassifiziert werden. Mit *single-center-defined cluster* und höherem ξ hätte man nur die “Spitzen”.

4 Association Rule Mining

4.1 Definitionen

Transactions Sei $T = \{t_1, t_2, ..., t_n\}$ die *Transaction Database*. Jede Transaktion $t \in T$ stellt ein itemset dar.

Support Anteil von Transaktionen, die ein itemset beinhalten.

$$s(A \to B) := P(A \cup B)$$

Frequent Itemset Ein itemset M ist *frequent/häufig* gdw. $s(M) \geq minSup$ wobei $minSup$ manuell festgelegt.

Confidence

$$c(A \to B) := P(B|A) = \frac{s(A \cup B)}{s(A)}$$

Angenommen A wurde gekauft gibt c die Wahrscheinlichkeit an, dass auch B gekauft wurde.

Beispiel computer → software [1%, 50%]
Wahrscheinlichkeit, dass beiderseits Computer und Software gekauft wird ist 1%. Wsl., dass, wenn Computer gekauft wurde, dann auch Software gekauft wurde ist 50%.

Problem: Anzahl möglicher itemsets (alle möglichen Teilmengen) ist zu hoch für einfache Überprüfung.

Apriori-Prinzip Ein itemset kann nicht häufig sein, wenn eine Teilmenge nicht häufig ist. D.h. falls ein itemset nicht häufig ist, muss keine Obermenge davon geprüft werden.

4.2 Algorithmus ohne FP-Tree

1. Candidate generation
2. Support determination

Algorithm 7: Apriori-Algorithm

```
/* Baue sukzessiv größere itemsets aus kleineren, eliminiere
   jeweils die, die nicht häufig sind.                    */
// C_k: Candidate k-itemsets
// L_k: frequent k-itemsets
L_1 ← frequent 1-itemsets ;
for k = 1, L_k ≠ ∅, k++ do
    C_{k+1} ← candidates generated from L_k ;
    // noch offen, wie diese candidate itemsets effizient
       generiert werden.
    L_{k+1} ← candidates in C_{k+1} that are frequent ;
    // Noch offen, wie der support eines itemsets effizient
       festgestellt werden kann.
end
return ⋃ L_k
```

An zwei Stellen *pruning* (candidate generation, support determination). je nach aufgabenstellung am ende die rules noch nach minimum confidence filtern.

4.2.1 Candidate Generation

finden von C_k mittels L_{k-1}.
Annahme: Alle itemsets sind sortiert.

1. **join** – Führe $p, q \in L_{k-1}$ zusammen zu $(s_1, s_2, ..., s_{k-2}, p_{k-1}, q_{k-1})$, falls sie in den ersten $k-2$ Positionen übereinstimmen.
2. **prune** – Entferne alle Elemente, die eine $k-1$-Teilmenge haben, die nicht in L_{k-1} liegt. *Dann ist diese Teilmenge nach Apriori-Prinzip nicht frequent*

L_3 = {(1 2 3), (1 2 4), (1 3 4), (1 3 5), (2 3 4)}
After join step: C_4 = {(1 2 3 4), (1 3 4 5)}
In pruning step: remove (1 3 4 5)

Wird entfernt, da $(3, 4, 5) \notin L_3$

4.2.2 Support Determination without FP-trees

Wie kann der *support* eines Candidate Itemsets $c \in C_k$ bestimmt werden? Dies ist entsprechend obiger Definition die Anzahl aller Transaktionen, in denen c als Teilmenge vorkommt.

Start bei Wurzel

1. Speichere C_k als Hash-Tree.
2. Bestimmte Hashwert $h(\tau_i)$ für alle $\tau \in t$ (einzelne Items)
3. Suche in entsprechendem Kind weiter.

In Innerem Knoten Innerer Knoten wurde erreicht durch $h(\tau)$

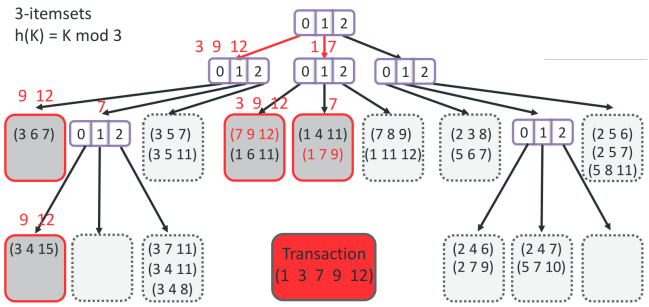
- Suche weiter für alle Hashwerte $h(\tau_k)$ mit $k > i$

Vorsicht: Es geht hier um alle $\tau_k \in t$, nicht lediglich die τ_i mit Hashwert gleich dem momentanen Knoten.

Bei Blattknoten

- Für jedes Itemset X in diesem Knoten, teste $X \subseteq T$. Falls, so inkrementiere den Support Count.

Beispiel .



1. Hashe separat $1, 3, 7, 9, 12$
2. In bspweise erstem Knoten auf zweiter Ebene erreicht durch $h(\tau_2) = 3$ mache weiter mit Hashen von $7, 9, 12 \in t > 3 = h(\tau_2)$

Diskussion

⊖ Anzahl der Kandidaten immernoch exponentiell.

⊖ Viele Scans der Datenbank notwendig wieso?

4.3 FP-Trees

Algorithm 8: FP-Tree Mining

```
/* Bestimme frequent 1-itemsets durch Abzählen.          */
L ← frequent 1-itemsets in descending order ;
foreach t ∈ L (start from back) do
    construct conditional pattern base ;
    draw conditional FP-tree for cpb // branches
      with insufficient support can be discarded
    if tree contains a single path p then
        generate patterns t ∪ α where α is a
          combination of nodes in p , with support
          = minimum support count of nodes in
          combination.
    else
        recurse on tree, concat resulting patterns
          with t.
    end
end
```

Optimierung Kann Teilbäume getrennt bearbeiten. todo Projection etc.

Diskussion Only when the support threshold is set very low does the FP-tree's ability to compress the dataset degrade. Under these conditions, the tree becomes bushy, with little node sharing.

5 Diverses

Beispiele für Machine Learning

- Warenkorb
- Amazon-Crosselllng
- Erkennung von Virus durch Analyse von Datenverkehr

6 Visualisierung

6.1 Grundlagen

Motivation Weshalb Visualisierung? Visualisierung ergänzt die statistische Analyse komplementär. Statistische Methoden nicht ausreichend, um alle Muster oder Informationen in Daten zu erkennen. (*Beispielsweise können sehr unterschiedliche Datenmengen gleiche statistische Kennwerte besitzen, cf Statosaurus*).

Visualisation Datenmengen sind durch einfache Inspektion nicht handzuhaben. Es geht bei Visualisierung darum, die menschliche Wahrnehmung zu unterstützen, bzw. neue Blickwinkel zu ermöglichen. Visualisierung ergänzt die statistische/mathematische Analyse komplementär. *Beachte auch: die Verfahren an sich generieren keine Einsicht, erst die Interpretation.*

Visualisation The use of computer-generated, interactive, visual representations of abstract data to amplify cognition.

Visualisation vs. Visual Analytics

- **Visual Analytics**: Visualisierung ist Schnittstelle zwischen Analyseprogramm und menschl. Experte, Programm nimmt Expertenwissen auf.
- dahingegen **Visualisierung**: statisches In-/Output, keine direkte Interaktion. *Ziele* (nur in eine Richtung, einzelner Schritt):
  - **Präsentation** von Daten. Die darzustellenden Fakten sind bereits bekannt und es gilt, eine geeignete Visualisierung zu finden um diese Fakten zu kommunizieren.
  - **Confirmatory Analysis** Gegeben ist eine Hypothese. Führe eine zielorientierte Überprüfung durch.
  - **Exploratory Analysis** Keine Hypothese gegeben. Versuche durch (oft interaktive, indirekte) Exploration der Daten Muster oder Trends zu gewinnen.

  Vis. kann hier Mittel von Visual Analytics sein (*wie ich es verstehe*).

Pre-attentive Perception Information wird ohne bewusste Interpretation wahrgenommen. zB Scatterplot von gleichförmigen Glyphen, eine hat dabei eine auffällig andere Farbe.

Gestalt Laws

1. Nähe
2. Ähnlichkeit
3. Verbundenheit
4. Geschlossenheit
5. Fortsetzung
6. Symmetrie
7. Figur/Grund

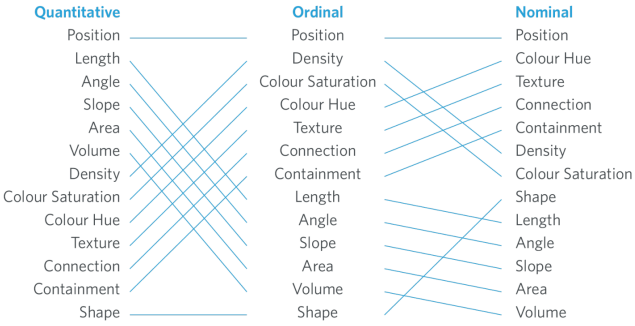
Sollten genutzt werden, um Objekte zu gruppieren, Zugehörigkeiten zu Verdeutlichen, verschiedene Gruppen voneinander abzugrenzen, ...

Visual Variables

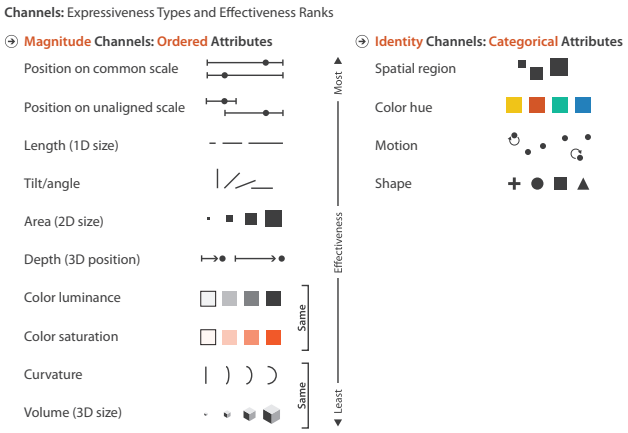
- Position
- Shape / Mark
- Size (Length, Area, Volume)
- Brightness
- Hue (Color)
- Angle / Orientation
- Texture
- Motion

Interpretation Accuracy of Vis. Var.s of numerical values In dieser Reihenfolge können abgebildete Werte (*mapping* von Daten auf visuelle Variablen) mit absteigender Genauigkeit wieder ausgelesen werden:

1. Position
2. Length
3. Angle, Shape
4. Area
5. Volume (3D)
6. Color, Density



... oder in anderer Ausführung



cf. Munzner: Data Analysis and Visualisation

Levels of information Information über...

1. *... einzelne Objekte*: zB Welches Auto hat den höheren Preis?
2. *... Gruppen von Objekten*: zB "Was ist besonders an der Marke Audi?", "Was passierte in dieser Zeitspanne?"
3. *... Mengen von Objekten*, d.h. abstrakte, globale Information, wie zB "Zu welcher Jahreszeit und welchen Angeboten kommen wie viele Besucher?", wird genutzt für Entscheidungen.

Embellishments können einen sehr negativen Effekt auf *interpretation accuracy* haben.

⊕ Memorability
⊖ Misleading
⊖ Distracting

Grundliegende Visualisierungsmethoden

- Linechart
- Map
- Barchart
- Scatterplots

Manchmal sinnvoll, diese zu kombinieren.

Information Visualisation Reference Model

1. Raw Data
2. Data Tables
   nach Einschränkung, Cleaning, Transformationen, ...
3. Visual Structures
   Mapping von Attributen auf visuelle Elemente/Parameter
4. Visualisation

Glyph abgeschlossene (kleine) grafische Einheit, die einen Datenpunkt repräsentiert. Attribute des Datenpunkts sind auf visuelle Variablen der Glyphe abgebildet. *zum Beispiel Star Glyph, Gesichter, kleine Schiffchen, Pfeile, ...* Zu beachten: Bekannte Metaphern berücksichtigen.

Irreführende Fehler

- Unterschiedliche Skalierungen (zB der Achsen) bei Nebeneinandergestellten Graphen.
- Skalierung nicht geeignet für Task.
- Skala wird mittendrin verändert.
- Ungeeignetes Mapping (keine präzise Interpretation möglich, zB numerischen Wert auf Volume oder kategorischen auf Position).
- Wichtige Informationen werden nicht dargestellt (ungeeignete Skalierung, Mapping, Gegenüberstellung)
- Inkonsistente Metaphern
- Baseline nicht bei Null (manchmal aber sinnvoll)
- Stacking auf Werte-Dimension (zB. vertikales Stacking von Bar-Chars)
- 3D-Effekt, Perspektive

Visualisierung beurteilen – Tipps

- Ranking von "Accuracy of Interpretation beachten"
- Gut geeignete Mappings beachten (zB numerisch & wichtig auf Länge).
- Schauen, ob Gestalt Laws angewendet werden oder diese womöglich der Intention widersprechen.
- Pre-attentive Processing?
- Werden Metaphern genutzt?
- Wird Information redundant kodiert?
- Eine "gute" Visualisierung hat zwei Merkmale:
  - **Expressiveness**: V. zeigt *alle* und *nur* die gegebene Informationsmenge
  - **Effectiveness**: Schnelle und genaue Interpretation ohne hohe Kosten

6.2 Visualising Multivariate Data

6.3 Point-based techniques

Ein Datenpunkt wird von genau einem Glpyh/Punkt/Mark repräsentiert.

6.3.1 Dimension Embedding

Eine weitere Dimension einbetten, indem man dafür eine weitere, bisher noch nicht verwendete visuelle Variable benutzt. *Zum Beispiel Marks in Scatterplot*.
Probleme beim Darstellen eines 3-dimensionalen Raumes in zwei Dimensionen:

- *occlusion* – Ein Objekt verdeck das andere
- *deptch perception* – uU schwierig auszumachen, in welcher Tiefe genau ein Objekt liegt
- *projection* – die gewählte Projektion könnte gewisse Strukturen nicht darstellen.

Beispiele von Eignung von zusätzlicher visueller Variable in einem Scatterplot:
Geeignet:

- Form für kategorische Daten
- Fläche für ordinale oder numerische Daten
- Helligkeit für ordinale oder numerische Daten
- Farbton für kategorische Daten

Schwieriger:

- Ausrichtung – unklar, wie zu interpretieren da unklar, wo Nullpunkt liegt.
- Textur – womöglich schwer unterscheidbar bei kleiner Fläche

6.3.2 Multiple Displays

Angeführtes Beispiel: *Scatterplot-Matrix*

⊖ nicht geeignet für mehr als eine handvoll Dimensionen

⊖ Symmetrie verschwendet Platz

⊖ sinnvolle Anordnung unklar.

6.3.3 Dimension Reduction

Ansätze:

- PCA
- MDS (*Multidimensional Scaling*)
- RadVis

6.3.4 Dimension Subsetting

Wähle (automatisch) eine Teilmenge von Dimensionen, die visualisiert werden basierend auf einem Maß von Ïnteressantheit". Dafür gibt es Taxonomien. Hängt natürlich immer wieder von Aufgabenstellung ab.

6.4 Line-based techniques

Braided graph may be very expressive, depending on task (highlight changes of maxixmum)

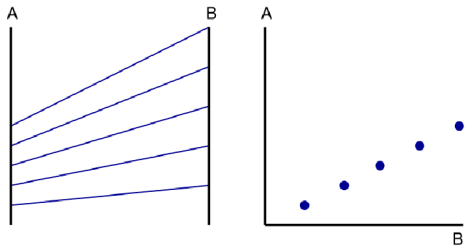
Horizon graph Meaningful colour steps should be chosen

⊕ saves space!

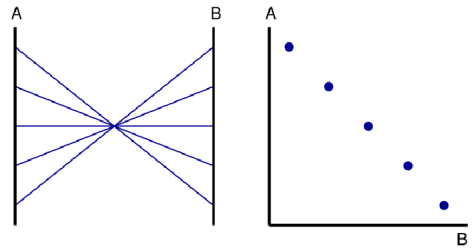
⊖ detail lookups are harder (*high-level information is more apparent, low-level less*)

Issues with line-based techniques

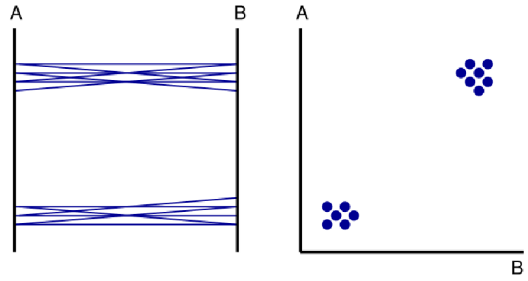
- superimposing linegraphs with different scales, variances. Potential solutions:
  - Juxtaposition, each with own scale.
    - ⊖ (disadv.: not easily comparable for absolute values)
  - Normalise (percentage of change)
    - ⊖ cant get exact values.



(a) Linear relation



(b) Inverse relation



(c) Multiple clusters

Achsen (möglicherweise auch mehr als zwei) werden parallel angeordnet, eine Linie je Datenpunkt.

Parallel Coordinates

⊖ Wie sollen Achsen angeordnet werden?

⊖ evtl. nicht ganz intuitiv für Anfänger

Types of hierarchical information

- Connectedness
- Social Networks
- derived-from relations
- Any kind of binary relation
- Shared property – Different to clustering! Clustering considers similarity across all attributes, not just one property.

6.5 Region-based techniques

- Bar and Pie Charts
- Tabular Displays
- Dimensional Stacking
- ...

6.6 Space-filling Methods

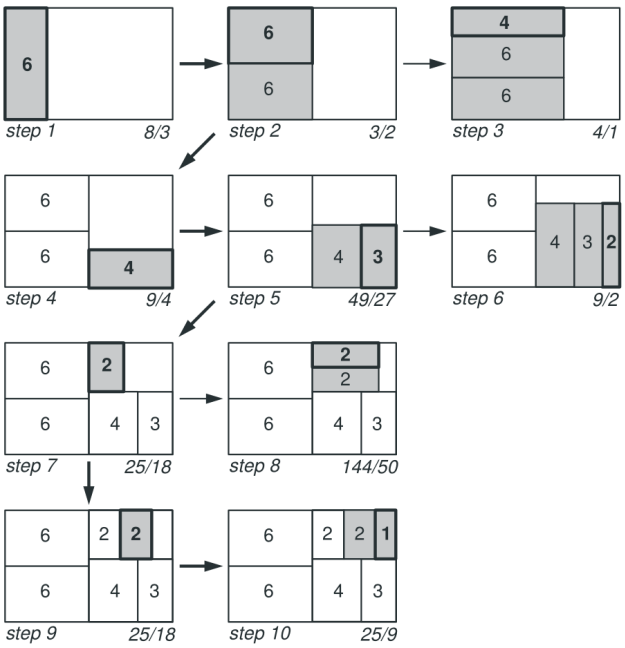
Tree Map Hierarchical Partition of screen space.

Construction (simple Algorithm):

- Divide alternating (vertically, horizontally, vertically, ...)
- Width of split is determined by number of leaf nodes in each subtree.
- Begin with most important attribute

⊕ Space efficient
⊕ Maintains hierarchy
⊕ Maintains proportionality
⊖ Tiling algorithm might create long, skinny boxes.

Squarified Treemap *tries to keep aspect ratios small*



Gehe zu nächster Spalte/Zeile über, wenn Aspect Ratio über Schwellwert kommt.

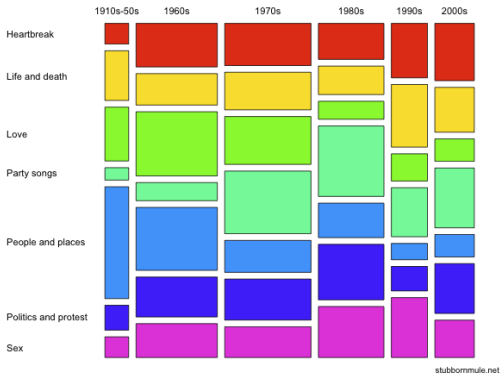
Construction

Comparison

⊕ areas easier to compare
⊖ hierarchical structure is less visible
⊖ does not preserve ordering
⊖ changes in data set may drastically change Vis.

Mosaic Plot *similar to Treemap*
Unterteile horizontale / vertikale Achsen jeweils abwechselnd nach vorher geordneten Attributen.

1. Sortiere Variablen nach einer Ordnung
2. Weise abwechselnd jeder Variable/Dimension die horizontale oder vertikale Achse zu und unterteile diese in Höhe/Breite entsprechend den Anteilen in den Daten.
3. Zusätzlich können die resultierenden Blöcke auf eingefärbt werden (Redundanz)



Der Flächenanteil einer einzelnen Box entspricht genau dem Anteil von Datenpunkten, die die entsprechende Kombination von Ausprägungen besitzen.

- Nicht für numerische Attribute möglich
- Mindestens zwei Variablen, bei zu vielen wird unübersichtlich

6.7 Dense Pixel Displays

Pixel bar chart Divide data among an attribute - each division makes up a bar. Within bars, each pixel represents a data point. Arrange points by putting down axes according to other attributes. Improvement: use equal-height-differing-width bars for more efficient use of space.

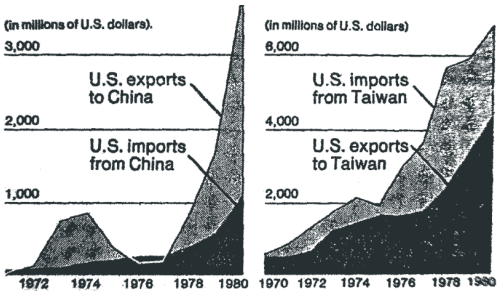
⊖ Choosing dimensions has big impact on appereance.

6.8 Übungen

Ab hier sämtlich Eigenversuche – keine Gewähr!

6.8.1 Beurteilung

US trade with China and Taiwan .

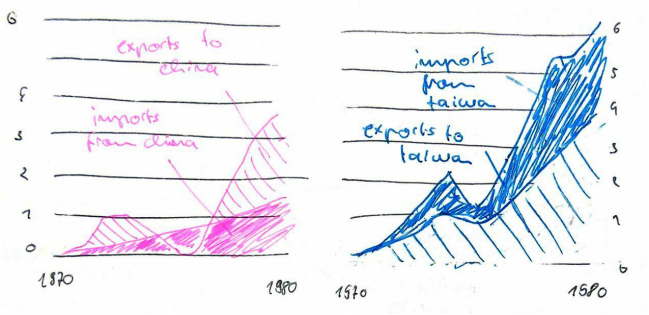


What's wrong?

- Durch das Nebeneinanderstellen der beiden Diagramme wirkt das *Gesetz der Fortsetzung* und die Messlinien der Skalen werden als fortlaufend wahrgenommen. ⇒ wird leicht übersehen, dass die Messlinien in beiden Diagrammen in Wirklichkeit verschiedene Werte markieren. ⇒ Das Niveau von Importen/Exporten bzgl Taiwan wird ggü. China völlig

falsch eingeschätzt, nämlich als etwa gleich wahrgenommen.

- Durch Verwendung der Farbkodierung einerseits für Exporte, andererseits für Importe wirkt das *Gesetz der Ähnlichkeit* und das Eine kann für das Andere wahrgenommen werden. ⇒ Insgesamt entsteht der Eindruck, dass Export- und Import-Verhältnisse für China und Taiwan in etwa gleich sind.
- In Wirklichkeit sind aber Import/Export-Balance sowie absolute Werte für China und Taiwan sehr unterschiedlich.

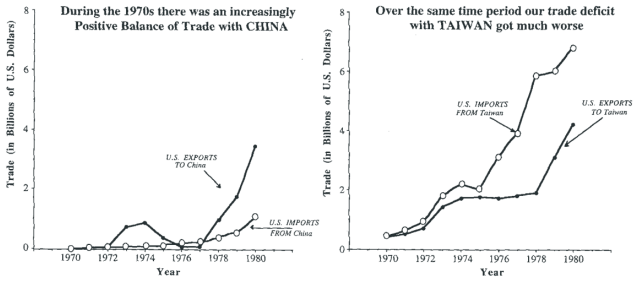


Jenachdem, ob Werte oder Zeitpunkte wichtiger sind könnte man die Diagramme auch vertikal übereinanderstellen.

Mappings:

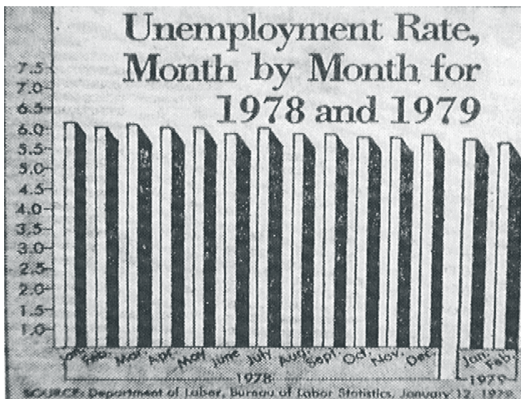
- Farbe auf Land
- Textur, Helligkeit (nicht abgebildet) auf Import/Export
- Position auf Wert

Eventuell noch Problem: Überdeckung von Flächen nimmt vom sichtbaren Flächeninhalt einer Fläche. Eventuell Transparenz-Effekte? Oder nur Linien?



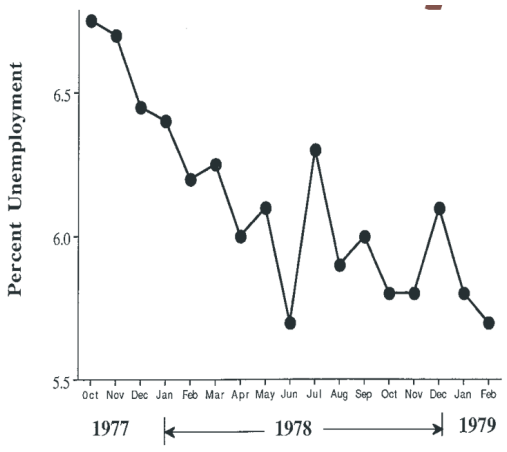
Hier nur Linien: besser aber langweilig? Und Mapping von Import/-Export auf Punkt-Glype

Unemployment Rate .



What's wrong?

- Skala ist nicht geeignet für die Anwendung: Bei einer Arbeitslosenquote sind Unterschiede von 0.5% bereits erheblich. Diese sind hier kaum sichtbar.



Aussagekräftige Skala, Mapping straightforward.