

# 1 Grundlagen

- Qualitative Attribute:
  - Variieren nach Beschaffenheit
- Quantitative Attribute:
  - Variieren nach Wert/Zahlen
- Diskrete Attribute:
  - abgestufte Werte
- Stetige Attribute:
  - können im Intervall jeden reellen Wert annehmen

## 1.1 Skalenniveaus

- Nominal
  - nur Gleichheit oder Andersartigkeit feststellbar (keine Bewertung)
  - stets qualitativ
- Ordinal
  - natürliche oder festzulegende Rangfolge
- Kardinal/Metrisch
  - numerischer Art
  - Ausprägung und Unterschied sind messbar
  - verhältnisskaliert (Absoluter Nullpunkt vorhanden; (Doppelt so viel.))
  - intervallskaliert (Kein Nullpunkt, nur Differenzen)

## 1.2 Sym. vs asym. Attribute

- Das symmetrische binäre Attribut ist ein Attribut, bei dem jeder Wert gleichwertig ist (w/m)
- Asymmetrisch ist ein Attribut, bei dem die beiden Ausprägungen nicht gleichwertig sind (Testergebnisse oder Vergleich von Umfragen)

## 1.3 Rauschen Artefakte, Ausreißer

- Rauschen (Random Verzerrung der Messung durch Einflussfaktoren)
- Artefakte (Unvollständige Messwerte)
- Ausreißer (Messwerte, die nicht im Normalbereich liegen)

## 1.4 Datenvorverarbeitung

- Aggregation (Zusammenfassung mehrerer Messwerte, Details gehen verloren)
- Sampling (Random (Behält gleiche Verteilung), Stratified (Einordnung in ähnliche Subgruppen nach Attribut und dann random daraus ziehen))
- Diskretisierung / Binarisierung (Gleiche Breite, gleiche Anzahl, Cluster)
- Transformation (Skalierung etc., PCA?, Kernel?)
- Dimensionsreduktion (Reduzierung und trotzdem Inhalte behalten)
- Feature Subset Selection (Konzentration auf wichtige Features)
- Feature Creation (W & H: Algo zusammenhang nicht verstehen, Größe (H\*W) geht besser)

## 1.5 Ähnlichkeits- und Distanzmaße

### 1.5.1 Ähnlichkeit

Eigenschaften:

- $s(x, y) 0 \leq s \leq 1$
- $s(x, y) = 0$ , wenn  $x \neq y$
- Symmetry:  $s(x, y) = s(y, x)$

**Simple Matching Coefficient (SMC):**

- $SMC = \frac{f_{00} + f_{11}}{f_{01} + f_{10} + f_{00} + f_{11}}$
- Binäre Daten
- gut für **sym. Attribute**, da Vorhandensein und Abwesenheit gleich gewertet wird

**Jaccard Coefficient:**

- $J = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$
- Binäre Daten
- gut für **asym. Attribute**, da Vorhandensein gewertet wird
- $0 \leq EJ \leq 1$  (1 sehr ähnlich)

**Extended Jaccard Coefficient (Tanimoto):**

- $EJ: \frac{\langle x, y \rangle}{||x||^2 + ||y||^2 - \langle x, y \rangle}$
- Jaccard für alle Daten
- $0 \leq EJ \leq 1$  (1 sehr ähnlich)

**Cosine Similarity:**

- $\cos(x, y) = \frac{\langle x, y \rangle}{||x|| * ||y||}$
- $-1 \leq \cos(x, y) \leq 1$
- 1 = sehr ähnlich, 0 = Vektor im 90° Winkel, -1 = Vektor im 180° Winkel

- Umrechnung von Zahl zu Winkel im Taschenrechner mit  $\cos^{-1}$
- auch für asym. Attribute da 0-0 Paare rausfallen

**Correlation:**

- $\text{corr}(x, y)$  über Taschenrechner
- zeigt linearen Zusammenhang ( $-1 \leq \text{corr} \leq 1$ )
- Taschenrechner:
  - Menü 6 -> 2 Statistik ->  $\leq a + bx$
  - Dateneingabe und AC drücken
  - $OPTN$  -> Regression
  - $r = \text{correlation}$

### 1.5.2 Distanz (Minkowski)

Eigenschaften:

- Positivity ( $d(x, y) \geq 0$ ,  $d(x, y) = 0$ , wenn  $x = y$ )
- Symmetry ( $d(x, y) = d(y, x)$ )
- Triangle Inequality ( $d(x, z) \leq d(x, y) + d(y, z)$ )

$$d(x, y) = \sqrt[r]{\sum_{k=1}^n |x_k - y_k|^r}$$

Name	r	Anwendung
Hamming	1	Bin. Vekt.
CityBlock	1	nur gerade
Euclid	2	schräg (gleiche Skalierung)
Supremum	$\infty$	nur größte Dist.

### 1.5.3 Weiteres

**Verhalten für Multiplikation und Addition:**

Property	Cosine	Correlation	Minkowski
Invariant to multiplication	Yes	Yes	No
Invariant to addition	No	Yes	No

**Mutual Information:**

- Ähnlich wie Correlation, aber für nicht linearen Zusammenhang
- 0 = kein Zusammenhang, 1 = starker Zusammenhang
- **HIER fehlt**

**Umrechnung Ähnlichkeit < - > Distanz:**

Bspw:

- $s = \frac{1}{d} - 1$
- $s = \ln(x) * -1$
- $d = 1 - s$
- $d = \sqrt[2]{1 - s}$

## 2 Klassifikation

- Zuordnung einer abhängigen Variable (y) anhand von unabhngigen Variablen
- Model hat beim Training (Induction) gelernt zuzuordnen
- Model wendet das Gelernte bei der Klassifikation an (Deduction)

### 2.1 Beispiele von Klassifikationsverfahren

- Elementare Verfahren (Decision Trees, KNN, Naive Bays, SVM, NN)
- Ensemble Verfahren (Random Forests, bagging, Boosting, ...)

### 2.2 Entscheidungsbume

- Datensatz durchluft von der Wurzel bis zum Blatt die Knoten und wird anhand der Entscheidungen am Knoten klassifiziert
- Hunts Algo entscheidet, wie Splits gesetzt werden (gibt noch mehr)

#### 2.2.1 Hunts Algo

- Sei  $D_t$  die Menge der Trainingsdatenstze, die Knoten t erreichen
- Wenn  $D_t$  nur Datenstze enthlt, die zur selben Klasse ytgehren, dann ist t ein Blatt des Baumes und wird mit ytgekennzeichnet.
- Falls  $D_t$  Datenstze enthlt, die zu mehr als einer Klasse gehren, verwende eine Attribut-Testbedingung, um die Daten in kleinere Untermengen aufzuteilen

#### 2.2.2 Split bei Attributen

- Binrer Split
- Mehrfach Split

Mglichkeiten der Diskretisierung

- Einteilung in gleichbelegte Bereiche (Percentile)
- Einteilung in gleiche Bereiche (Clustering)
- Binre Entscheidung: ( $A < v$ ) und ( $A \geq v$ )

**Greedy Ansatz** Algorithmus der schrittweise den besten nchsten Schritt mit dem hchsten Gewinn whlt.

### 2.3 Ma fr Knotenunreinheit

- $p_i(t)$  Hufigkeit von Klasse i beim Knoten t
- $c$  Gesamtzahl der Klassen
- **Gini Index**
  - $GI = 1 - \sum_{i=0}^{c-1} p_i(t)^2$
  - Maximum:  $1 - \frac{1}{c}$

- Minimum: 0 (Best Case)
- Ablesbare Werte ohne Berechnungen:
  - \* **Nur bei 2 Klassen:**
  - \* Klassen mit 0 und 0 = 0
  - \* Klassen mit 0 und X = 0
  - \* Klassen mit X und X = 0.5

#### • Entropy

- $E = -\sum_{i=0}^{c-1} p_i(t) * \log_2 p_i(t)$
- Maximum:  $\log_2 c$
- Minimum: 0 (Best Case)

#### • Klassifikationsfehler

- $CE = 1 - \max[p_i(t)]$
- Maximum: Wenn alle Datenstze auf die Klassen gleich verteilt sind
- Minimum: 0 (Best Case, wenn alle Datenstze zu einer Klasse gehren)

### 2.4 Nachfolgende Berechnungen

- Knnen mit allen 3 Maen berechnet werden.
- Split

- $split = \sum_{i=1}^k \frac{n_i}{n} * \text{Knotenunreinheit}$
- $n_i$  = Anzahl der Daten im Kindknoten i
- $n$  = Anzahl der Daten im Elternknoten

#### • Gain

- $gain = P - M$
- $P$  = Knotenunreinheit des Elternknoten
- $M$  = Split der Kindknoten
- Gain maximieren fr einen guten Split bzw. M minimieren!!!
- **InformationGain:** Gain berechnet mit der Entropie!

- **Problem: Splits mit vielen Kindsknoten mit wenigen aber einen Datenstze werden bevorzugt!**

#### • SplitInfo

- $splitInfo = -\sum_{i=1}^k \frac{n_i}{n} \log_2 \frac{n_i}{n}$
- $splitInfo$  = Entropie der Partitionierung

#### • GainRatio

- $gainRatio = \frac{gain_{split}}{splitInfo}$
- Korrigierter Gain um Entropie -> Bestrafung hoher Anzahl kleiner Partitionen
- Maximum (Best Case)

### 2.5 Bewertung Bume, Overfitting etc.

- Trainingsfehler: Klassifikationsfehler von Daten aus Training

- Testfehler: Klassifikationsfehler von Daten aus Test
- Generalisierungsfehler: Erwarteter K-Fehler bei random Daten

#### 2.5.1 Under-/Overfitting

- Underfitting: Modell ist zu simpel (Training- / Testfehler gro)
- Overfitting: Modell ist zu komplex oder zu wenig Daten (Testfehler gro)
- Typischer Ellenbogen - Im "Knick" ist das Optimum

#### 2.5.2 Fehlerabschtzung

- Ockhams Razor/Sparsamkeitsprinzip
- pess. Fehlerabschtzung:  $err_{gen}(T) = err(T) + \Omega * \frac{k}{N_{train}}$
- $err(T)$  = Gesamtfehlermenge Training
- $k$  = Anzahl Bltter im Baum
- $N_{train}$  Anzahl Trainingsdatenstze

#### 2.5.3 Pruning

- Pre-Pruning
  - Stoppe im Prozess, wenn bspw.
  - Datenstze zur selben Klasse gehren
  - Alle Datenstze bei allen Attributen die selben Werte haben
  - Anzahl der Datenstze Schwellenwert unterschreitet
  - Klassenverteilung nach  $\chi^2$  unabhngig ist
  - Gain nicht hoch genug ist
- Post-Pruning
  - Nach Fertigstellung des Baumes
  - bottom-up Ansatz

## 2.6 Modell Evaluation

#### 2.6.1 Validierung

- Holdout (Split zwischen Training- und Testdaten)
- Kreuzvalidierung (Mehrfach Holdout mit disjunkten Mengen und Durchschnitt ber)

#### 2.6.2 Konfusionsmatrix

Siehe Bild im Repo

- Precision (% der richtig klassifizierten innerhalb der positiven Vorhersagen)
- Recall/True Positive Rate (% der richtig klassifizierten von den ursprnglich positiven)
- False Positive Rate (% der falsch positiv klassifizierten innerhalb der ursprnglich negativen)

- Accuracy (% der richtig klassifizierten Daten über allen)
- F1-Score (Gewichtetes Maß zwischen Precision und Recall)

### 2.6.3 ROC Kurve

Siehe Bild im Repo

- Achsen:
  - X: False Positive Rate
  - Y: True Positive Rate
  - (0,0): alle Prognosen negativ
  - (1,1): alle Prognosen positiv
  - (1,0): Idealzustand, alle Prognosen korrekt
- Diagonale (Ergebnis zufälligen Ratens)
- Area under the Curve (AUC)
  - Idealwert 1, Zufallsmodell 0.5
- **Bester Split beim Punkt, der am nächsten an (1,0) liegt!**
- ROC visuell lösen:
  - Oben rechts in Vis anfangen, links in der Tabelle
  - Tabelle vorstellen, dass Grenze Schritt für Schritt nach rechts geschoben wird. (Links der Grenze Klasse A, rechts Klasse B)
  - Wenn hinzukommender Eintrag richtig klassifiziert wird: Curve geht 1 Schritt nach links
  - Wenn hinzukommender Eintrag falsch klassifiziert wird: Curve geht 1 Schritt nach unten
  - Schrittgröße Links =  $\frac{1}{Anzahl_{neg. Einträge}}$
  - Schrittgröße runter =  $\frac{1}{Anzahl_{pos. Einträge}}$
- Schwellenwerte ablesen:
  - Oben rechts anfangen und abzählen
  - Punkt finden und dann  $\geq S_{Punkt}$
- Gewichtung ändern:
  - FP x-mal schwerer als FN (FPR Achse mal x skalieren)
  - FN x-mal schwerer als FN (TPR Achse mal x skalieren)

## 3 Clustering

- Ordne Datenobjekt einem Cluster zu
- Objekt innerhalb des Clusters möglichst ähnlich
- Objekte aus unterschiedlichen Clustern möglichst unterschiedlich
- Exklusives Clustering (Daten dürfen nur einem C angehören) - nicht exklusives

- Probabilistisches Clustering (Datenpunkte gehören mit Wahrscheinlichkeit X zu Cluster. Alle Wahrscheinlichkeiten aufaddiert = 1)
- Fuzzy Cluster (daten gehören anteilig zu Clustern)
- Vollständiges Clustering (Alle Datenpunkte sind in einem C)
- Partielles Clustering (Es gibt Daten ohne Cluster)

### 3.1 Arten von Clustern

- Wahl-separierte Cluster
  - Jeder Punkt eines Clusters liegt näher an jedem anderen Punkt des Clusters als an irgendeinem Punkt eines anderen Clusters
- prototyp-basiert
  - Jeder Punkt liegt näher am Prototypen des Clusters, als an einem anderen Prototypen
  - Zentroid: Durchschnitt aller Punkte (Schwerpunkt)
  - Mediod: mittlerer Punkt (Median)
- kontiguitäts-basiert
  - Jeder Punkt eines Clusters liegt näher an (ist ähnlicher zu) einem oder mehreren Punkten des Clusters als zu irgendeinem Punkt eines anderen Clusters
- dichte-basiert
  - Ein Cluster ist eine Region dichter Punkte, die durch Regionen mit geringer Punktdichte von anderen Regionen mit hoher Punktdichte separiert ist
  - Einsatz bei **unregelmäßig geformten Clustern & bei Datenrauschen oder Ausreißern**

### 3.2 Partitionierendes Clustering (K-Means)

#### 3.2.1 Eigenschaften:

- Vollständig partitionierend
- prototyp-basiert (Zentroid)
- Anzahl der Cluster muss vorgegeben werden
- Speicherplatzkomplexität:  $O((n + K) * d)$
- Laufzeitkomplexität:  $O(n * K * I * d)$
- $n$ = Anzahl der Punkte,  $K$ = Anzahl der Cluster,  $I$ = Anzahl der Iterationen,  $d$ =Anzahl der Attribute
- Algo konvergiert recht schnell
- Durch zufällig gewählte Zentroids unterschiedliche Ergebnisse
- Häufig wird nur lokale und nicht globale Minimum gefunden!

#### 3.2.2 Algorithmus:

- Select k points as initial centroids
- repeat
  - Form k clusters by assigning all points to the closest centroid
  - recompute the centroid of each cluster
- until centroids don't change

#### 3.2.3 Zielfunktion

- Sum of Squared Error (SSE)
- Fehler: Abstand eines Datenpunkts zum nächstgelegenen Zentroid
- $SSE = \sum_{i=1}^k \sum_{x \in c_i} dist(m_i, x)^2$
- Summe sinkt mit jeder Iteration bis lokales/globales Min erreicht ist

#### 3.2.4 Optimierung KMeans

- Mehrfach ausführen (Wahrscheinlichkeit schlecht)
- Zentroid Wahl über Heuristik (Wahl nacheinander mit möglichst großem Abstand)
- Bisecting (Erst zwei Cluster, dann weiter spalten)

#### 3.2.5 Grenzen und Probleme KMeans

Schlecht bei:

- Cluster unterschiedlicher Größe
- Cluster unterschiedlicher Dichte
- Cluster nicht kugelförmig sind
- Cluster mit Ausreißer oder Rauschen haben

### 3.3 DBSCAN - dichte basiertes Clustering

#### 3.3.1 Eigenschaften

- Keine Probleme mit unterschiedlichen Clustergrößen und -formen
- Unempfindlich gegenüber Rauschen
- Kernpunkt
  - Punkt, in dem  $\epsilon$  Umgebung sich mindestens  $MinPts$  befinden
  - Kernpunkt wird mitgezählt
- Randpunkt
  - Nicht Kernpunkt
  - Aber in  $\epsilon$  Distanz von kernpunkt entfernt ist
- Rauschpunkt
  - Weder Kern- noch Randpunkt

### 3.3.2 Algorithmus

- Kennzeichne alle Punkte als, kern, Rand und Rauschpunkt
- Lösche Randpunkte
- Verbinde alle Kernpunkte, die in  $\epsilon$  Distanz liegen
- Jede Gruppe von Kernpunkten wird ein Cluster
- Jeder Randpunkt wird einem Cluster zugeordnet

### 3.3.3 Parameter auswählen

- $\epsilon$ 
  - Plote Abstände zu allen k Nachbarn
  - Randpunkte haben kurze Distanz, Rauschpunkte große
  - Sobald Distanz sign. ansteigt,  $\epsilon$  ablesen
- *MinPts*
  - Zu kleiner Wert führt zu Miniclustern
  - Wert erhöhen bis Anzahl der Cluster nicht mehr stark sinkt

### 3.3.4 Grenzen und Probleme DBSCAN

Schlecht bei:

- Cluster unterschiedlicher Dichte
- hochdimensionalen Daten

## 3.4 Hierarchisches Clustering

### 3.4.1 Eigenschaften

- Erzeugt Menge von verschachtelten Clustern
- Baumdiagramm der Cluster als Dendrogramm darstellbar
- Keine Angabe von Clusteranzahl nötig! (Anzahl kann sich nach Erstellung beliebig ausgesucht werden)
- Agglomeratives Clustern (verschmelzent)
- Divisives Clustern (teilend)
- Verwenden beide Ähnlichkeits- oder Distanzmaße (Greedy Ansatz)
- Speicherplatzkomplexität:  $O(n^2)$
- Zeitkomplexität:  $O(n^3)$

### 3.4.2 Algo agglomeratives Clustering

- Abstandsmatrix berechnen
- repeat
  - Merge 2 dichteste Cluster
  - Update Abstandsmatrix
- until, 1 Cluster übrig bleibt

### 3.4.3 Inter-Cluster Abstand

- Minimaler Abstand (Single Link)

- Wähle immer das Minimum der kleinsten Abstände aus
- Kein Problem mit nicht elliptischen Clustern!
- Probleme mit verrauschten Daten und Ausreißern
- Maximaler Abstand (Complete Link)
  - Wähle immer das Minimum der maximalsten Abstände aus
  - Kaum/Keine Probleme mit verrauschten Daten und Ausreißern
  - Bevorzugt kugelförmige Cluster
  - Trennt große Cluster häufiger auf
- Durchschnittlicher Abstand (Average Link)
  - Der Abstand zweier Cluster entspricht dem Durchschnitt der Abstände aller Punktpaare aus den verschiedenen Clustern
  - Wähle immer das Minimum der durchschnittlichen Abstände
  - Kompromiss zwischen Min und Max Distanz
  - Stärke (Weniger anfällig für Rauschen, Trennt große Cluster weniger häufiger)
  - Schwäche (Bevorzugt kugelförmige Cluster, Berechnung aufwändig)