

# 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
- Diskretisierung / Binarisierung
- Transformation
- Dimensionsreduktion
- Feature Subset Selection (Konzentration auf wichtige Features)
- Feature Creation

## 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

**Extended Jaccard Coefficient (Tanimoto):**

- $EJ: \frac{\langle x, y \rangle}{||x||^2 + ||y||^2 - \langle x, y \rangle}$
- Jaccard für alle Daten

**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:**

- $corr(x, y)$  über Taschenrechner
- zeigt linearen Zusammenhang

### 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
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 unabhängigen 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 Entscheidungsbäume

- Datensatz durchläuft 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 Trainingsdatensätze, die Knoten  $t$  erreichen
- Wenn  $D_t$  nur Datensätze enthält, die zur selben Klasse  $y$  gehören, dann ist  $t$  ein Blatt des Baumes und wird mit  $y$  gekennzeichnet.
- Falls  $D_t$  Datensätze enthält, die zu mehr als einer Klasse gehören, verwende eine Attribut-Testbedingung, um die Daten in kleinere Untermengen aufzuteilen

### 2.2.2 Split bei Attributen

- Binärer Split
- Mehrfach Split

Möglichkeiten der Diskretisierung

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

**Greedy Ansatz** Algorithmus der schrittweise den besten nächsten Schritt mit dem höchsten Gewinn wählt.

## 2.3 Maß für Knotenunreinheit

- $p_i(t)$  Häufigkeit 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)
- **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 Datensätze auf die Klassen gleich verteilt sind
- Minimum: 0 (Best Case, wenn alle Datensätze zu einer Klasse gehören)

## 2.4 Nachfolgende Berechnungen

- Können mit allen 3 Maßen 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 für einen guten Split bzw.  $M$  minimieren!!!
- **Problem: Splits mit vielen Kindsknoten mit wenigen aber einen Datensätze 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 Bäume, 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 groß)
- Overfitting: Modell ist zu komplex oder zu wenig Daten (Testfehler groß)
- Typischer Ellenbogen - Im "Knick" ist das Optimum

### 2.5.2 Fehlerabschätzung

- Ockhams Razor/Sparsamkeitsprinzip

- pess. Fehlerabschätzung:  $err_{gen}(T) = err(T) + \Omega * \frac{k}{N_{train}}$
- $err(T)$  = Gesamtfehlermenge Training
- $k$  = Anzahl Blätter im Baum
- $N_{train}$  Anzahl Trainingsdatensätze

### 2.5.3 Pruning

- Pre-Pruning
  - Stoppe im Prozess, wenn bspw.
  - Datensätze zur selben Klasse gehören
  - Alle Datensätze bei allen Attributen die selben Werte haben
  - Anzahl der Datensätze Schwellenwert unterschritten
  - Klassenverteilung nach  $\chi^2$  unabhängig 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 über)

### 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 ursprünglich positiven)
- False Positive Rate (% der falsch positiv klassifizierten innerhalb der ursprünglich 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!**

### 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

- $\epsilon$ 
  - Plotte 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

## 4 Übungsaufgaben und Musterlösungen