# Methoden zur Feature Subset Selection für Unsupervised Learning - Überblick und neuer Ansatz

Diplomarbeit

von

Frank Steinmann

# Inhalt

- Feature Subset Selection - wozu ?
- Probleme bei Unsupervised Learning
- Allgemeine Strategien
- Beispiele verschiedener Ansätze
- Neuer Ansatz
- Testergebnisse
- Schlussfolgerungen

# Feature Subset Selection - wozu?

- Wir wollen Informationen aus den Daten herausholen
- Warum nicht jede Information nutzen, die wir bekommen können?
- Datensätze oft groß, je mehr Attribute, desto langsamer die Verarbeitung

# Feature Subset Selection - wozu?

- Attribute können irrelevant sein und das Finden interessanter Informationen erschweren
- Deshalb: Möglichst viele Attribute herausnehmen und dabei möglichst wenig Information verlieren
- Das Erkennen irrelevanter Attribute ist selbst schon ein Informationsgewinn

# Probleme bei Unsupervised Learning

- Beim Supervised Learning wissen wir, nach welchen Informationen wir suchen
- Attribute sollen die Klasse vorhersagen
- Beim Unsupervised Learning gibt es kein Klassenattribut
- Wir versuchen Clusterings zu endecken, interessante Gruppierungen innerhalb der Daten

# Probleme bei Unsupervised Learning

- Wir müssen Attribute identifizieren, die Informationen enthalten über ein Clustering, welches wir noch gar nicht kennen
- Es gilt also, eine Menge von Attributen zu finden, mit denen sich gute Clusterings erstellen lassen

# Allgemeine Strategien

- Zwei Dinge sind für eine FSS entscheidend:
  - Ein Suchalgorithmus, der den Lösungsraum der möglichen Attributmengen durchsucht
  - Eine Bewertungsfunktion, die Attributmengen bewertet
- Man unterscheidet zwei grundsätzliche Ansätze: Wrapper und Filter

# Allgemeine Strategien - Wrapper

- Erzeugung von Clusterings auf Basis der ausgewählten Attribute
- Bewertung der Attributauswahl durch Bewertung des Clusterings
- Bewertungsfunktion misst, wie gut die Cluster separiert sind (Berechnung der Varianz innerhalb der Cluster und der Abstände der Cluster zueinander)

# Allgemeine Strategien - Filter

- Attributmengen werden direkt bewertet, keine Erzeugung von Clusterings
- Mögliche Bewertungskriterien:
  - Entropie: gut separierbare Cluster bei niedriger Entropie
  - Abhängigkeit: relevante Attribute sind abhängig vom unbekannten Clusterattribut => relevante Attribute sind voneinander abhängig

# Allgemeine Strategien - Suche

- FSS ist ein Optimierungsproblem
- Vielzahl von Algorithmen ist möglich, einsetzbar sowohl für Wrapper als auch für Filter
- Mögliche Suchstrategien z.B. genetische Suche oder einfache sequenzelle Vorwärts- oder Rückwärtssuche

# Beispiele verschiedener Ansätze - Wrapper: Dy & Brodley

- Scatter Separability als Bewertungskriterium für die Qualität von Clusterings
- Berechnet werden die Streuung der Instanzen innerhalb der Cluster sowie die Streuung der Clusterzentren
- sequenzielle Vorwärtssuche zum Finden der besten Attributauswahl

# Beispiele verschiedener Ansätze - Wrapper: Kim et al.

- Gesucht wird nicht eine einizge Lösung, sondern eine Menge von pareto-optimalen Lösungen bzgl. mehrerer Zielfunktionen (= Bewertungsfunktionen)
- Verwendung eines evolutionären Algorithmus
- Genom der Individuen enthält Info über ausgewählte Attribute und Clusteranzahl

# Beispiele verschiedener Ansätze - Wrapper: Kim et al.

- Individuen erhalten Energie von verschiedenen Energiequellen
- Zu jeder Bewertungsfunktion gibt es mehrere Energiequellen
- Bewertungen eines Individuums legen fest, zu welchen Energiequellen es Zugang hat
- Energie wird jeweils aufgeteilt zwischen den Indidivuen, die Zugang haben

# Beispiele verschiedener Ansätze - Wrapper: Kim et al.

- Konstanter Energiewert wird in jeder Runde abgezogen
- Individuen mit Energie < 0 werden entfernt
- Bestehende Individuen werden kopiert, Energie wird aufgeteilt, Mutation sort für Veränderungen

# Beispiele verschiedener Ansätze - Wrapper: Kim et al.

- 4 Bewertungsfunktionen
  1. Basierend auf Abständen innerhalb der Cluster
  2. Basierend auf Abständen der Cluster zueinander
  3. Basierend auf der Clusteranzahl (weniger werden bevorzugt)
  4. Basierend auf der Dimension (kleinere Dimension wird bevorzugt)

# Beispiele verschiedener Ansätze - Filter: Søndberg-Madsen et al.

- Idee: Clusterzugehörigkeit wird durch unbekannte Wahrscheinlichkeitsverteilung definiert
- Existenz einer unbekannten Zufallsvariablen C
- bekannte Zufallsvariablen $Y = (Y_1, \ldots, Y_n)$, die Attribute des Datensatzes

# Beispiele verschiedener Ansätze - Filter: Søndberg-Madsen et al.

- Erkenntnis: relevante Attribute sind solche, die von C abhängig sind
- Folgerung: relevante Attribute müssen auch voneinander abhängig sein
- Konsequenz: unabhängige Attribute sind irrelevant
- Überprüfung der Relevanz eines Attributs durch Berechnung der Abhängigkeiten

# Beispiele verschiedener Ansätze - Filter: Dash et al.

- Finden der besten Attributauswahl mit Hilfe eines Entropiemaßes
- Bei Gleichverteilung maximale Entropie, weil maximale Unrodnung
- Bei Daten mit gut separierbaren Clustern sollte Entropie niedrig sein
- Berechnet wird die Entropie der Distanzen zwischen den Instanzen

# Beispiele verschiedener Ansätze - Filter: Dash et al.

- Als Suche wird eine sequenzielle Vorwärtssuche verwendet

# Neuer Ansatz

- Verwendung eines genetischen Algorithmus
- Problem: Lange Laufzeit
- Lösung: Ein paar einfache Modifikationen verkürzen die Laufzeit erheblich
- Voraussetzung ist Einsatz eines Clustering-Algorithmus mit iterativer Optimierung, in diesem Fall k-Means

# Neuer Ansatz - Suchalgorithmus

- Grundidee
  - Im Genom jedes Individuums sind die ausgewählten Attribute und die Clusteranzahl kodiert
  - In jeder Runde Erzeugung und Bewertung eines Clusterings für jedes Individuum
  - Schlechteste Individuen werden entfernt, beste Individuen erzeugen Nachkommen

# Neuer Ansatz - Suchalgorithmus

- **Modifikation**
  - Clustering-Algorithmus muss nicht immer komplett durchlaufen
  - In den ersten Runden „fertige“ Clusterings nicht unbedingt notwendig
  - Daher k-Means pro Runde nur 3 Iterationen berechnen lassen
  - Ergebnis merken und als Startwert für die nächste Runde einsetzen

# Neuer Ansatz - Suchalgorithmus

- Konsequenzen
  - wesentlich kürzere Laufzeit
  - nach einigen Runden sind fertige Clusterings vorhanden
  - am Anfang Bewertung unfertiger Clusterings, für Aussortierung schlechter Lösungen aber ausreichend

# Neuer Ansatz - Bewertungsfunktion

- Bewertungsfunktion basiert auf Streuung innerhalb der Cluster
- Ansatz:

$$score = \frac{1}{n} \sum_{i=1}^{k} n_i \cdot s_i$$

- Funktion soll aber unabhängig von Dimension und Clusteranzahl einsetzbar sein, Anpassungen notwendig

# Neuer Ansatz - Bewertungsfunktion

- Unabhängigkeit von der Clusteranzahl:

$$score = \frac{1}{n} \sqrt[dim]{n} \sum_{i=1}^{k} \left( \frac{n_i}{\sqrt[dim]{n_i}} \cdot s_i \right)$$

- bei Gleichverteilung nun gleiche Bewertung unabhängig von der Clustergröße und damit von der Clusteranzahl

# Neuer Ansatz - Bewertungsfunktion

- Unabhängigkeit von der Dimension:

$$score = \frac{1}{n} \sqrt[dim]{n} \frac{1}{\sqrt{dim}} \sum_{i=1}^{k} \left( \frac{n_i}{\sqrt[dim]{n_i}} \cdot s_i \right)$$

- Die größeren Distanzen bei höherer Dimensionalität werden ausgeglichen

# Testergebnisse

- Vergleich zwischen
  - SimpleGeneticFSS (der neue Ansatz, ohne die Modifikationen für kürzere Laufzeit)
  - FastGeneticFSS (der neue Ansatz)
  - KimFSS (der Wrapper von Kim et al.)
  - DashFSS (der Filter von Dash et al.)
- getestet wird mit synthetischen und mit realen Daten

# Testergebnisse - synthetische Daten

- $D_{klein}$:
  - 2 Attribute, die Cluster enthalten
  - 6 irrelevante Attribute
- $D_{groß}$:
  - 4 Attribute, die Cluster enthalten
  - 30 irrelevante Attribute
- irrelevante Attribute gleich- bzw. normalverteilt

# Testergebnisse - synthetische Daten

| $D_{klein}$ | $A_i$ | $A_r$ | $PA_{k\_gef}$ | $PA_{k\_korr}$ |
|---|---|---|---|---|
| Alle Attribute | | | | 0.6386 |
| SimpleGeneticFSS | 0.7 | 0.4667 | 0.8323 | 0.8561 |
| FastGeneticFSS | 0.6333 | 0.2667 | 0.8729 | 0.9053 |
| KimFSS | 0.7333 | 1.7333 | 0.5800 | 0.5706 |
| DashFSS | 1.5667 | 1.6667 | - | 0.5760 |

# Testergebnisse - synthetische Daten

| $D_{groß}$ | $A_i$ | $A_r$ | $PA_{k\_gef}$ | $PA_{k\_korr}$ |
|---|---|---|---|---|
| Alle Attribute | | | | 0.7313 |
| SimpleGeneticFSS | 1.2 | 0.8667 | 0.8155 | 0.9013 |
| FastGeneticFSS | 1.7 | 1.1333 | 0.8271 | 0.8782 |
| KimFSS | 2.4667 | 3.3667 | 0.5709 | 0.6000 |
| DashFSS | 1.3667 | 3.5 | - | 0.5983 |

# Testergebnisse - reale Daten

- iris:
  - 4 Attribute
  - 3 Klassen
- wine:
  - 13 Attribute
  - 3 Klassen
- Klassenattribut beim Lernen ignoriert

# Testergebnisse - reale Daten

| iris | ausgew. | $PA_{k\_gef}$ | $PA_{k\_korr}$ |
|---|---|---|---|
| Alle Attribute | 4 | | 0.823 |
| SimpleGeneticFSS | 2.3 | 0.719 | 0.925 |
| FastGeneticFSS | 2.3 | 0.710 | 0.925 |
| KimFSS | 1 | 0.844 | 0.943 |
| DashFSS | 1 | - | 0.578 |

# Testergebnisse - reale Daten

| wine | ausgew. | $PA_{k\_gef}$ | $PA_{k\_korr}$ |
|---|---|---|---|
| Alle Attribute | 13 | | 0.920 |
| SimpleGeneticFSS | 5.2 | 0.653 | 0.891 |
| FastGeneticFSS | 5.2 | 0.642 | 0.871 |
| KimFSS | 1 | 0.609 | 0.657 |
| DashFSS | 4 | - | 0.776 |

# Testergebnisse - k-Means-Iterationen

- $D_{klein}$
  - SimpleGeneticFSS: 18109 Iterationen
  - FastGeneticFSS: 3120 Iterationen
- $D_{groß}$
  - SimpleGeneticFSS: 34415 Iterationen
  - FastGeneticFSS: 6114 Iterationen
- bei jeweils gleicher Rundenanzahl des genetischen Algorithmus

# Schlussfolgerungen

- Neuer Algorithmus schneidet besser ab als Vergleichsalgorithmen
- Modifikationen brachten deutlichen Laufzeitgewinn ohne Einbußen bei der Qualität der Lösungen
- Bewertungsfunktionen allgemein noch verbesserungsbedürftig