Einführung in die Funktionsweise von ARJA

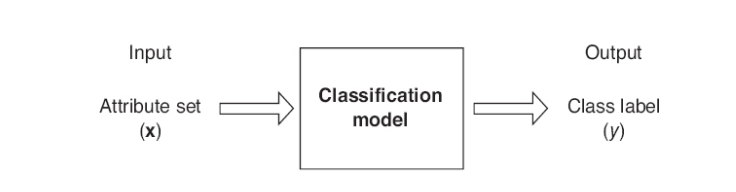
Machine Learning und genetische Algorithmen Einführung

Um die Funktionsweise von ARJA tiefgründig und umfassend zu verstehen ist es wichtig ein paar Grundkonzepte und Grundbegriffe in den komplexen Bereichen des Machine Learning sowie genetischen Algorithmen zu verstehen. Beide Teilgebiete sind wichtige Konzepte in dem Fachgebiet der künstlichen Intelligenz und beide werden dafür eingesetzt, komplexe Probleme zu lösen und um Muster in Daten zu erkennen.

Sowohl ML, als auch GA umfassen Systemkomponenten, die operatives Datenmaterial zur Informations- und letztlich Wissensgenerierung aufbereiten und speichern sowie Auswertungsund Präsentationsfunktionalität anbieten. Anders gesagt: Machine Learning umfasst den nichttrivialen Prozess der Identifikation gültiger, neuer potentiell nützlicher und letztlich verständlicher Muster in Datenbeständen^1“. Typische Verfahren im Machine Learning, bzw. Data Mining bilden die Clusteranalyse, Entscheidungsbaumverfahren, Assoziationsanalyse, Neuronale Netze oder auch die Regressionsanalyse, wobei in dieser Ausarbeitung für das bessere Verständnis von ARJA besonders auf das Entscheidungsbaumverfahren eingegangen werden soll. Vorher jedoch ein paar Grundbegriffe.

Maschinelles Lernen lässt sich in verschieden Arten unterteilen:

**Supervised learning** beschreibt ein Konzept des maschinellen Lernens, bei dem ein Modell anhand eines vorgegeben Datensatzes eine Zielfunktion f bildet, die dann ein Objekt O mit einer Menge Attributen x zu einer vordefinierten Klasse zuordnen kann.



Quelle: Tan; Steinbach; Kumar: Introduction to Data Mining, Addison-Wesley, 2005

Ein durchaus beliebtes Verfahren dieser Methode bilden sog. **Entscheidungsbäume**.

Sie stellen eine Baumstruktur dar, in der jeder Knoten eine Bedingung repräsentiert und jeder Ast einen möglichen Ausgang oder eine weitere Entscheidung darstellt. Am Ende der Äste werden die Vorhersagen oder Entscheidungen getroffen.

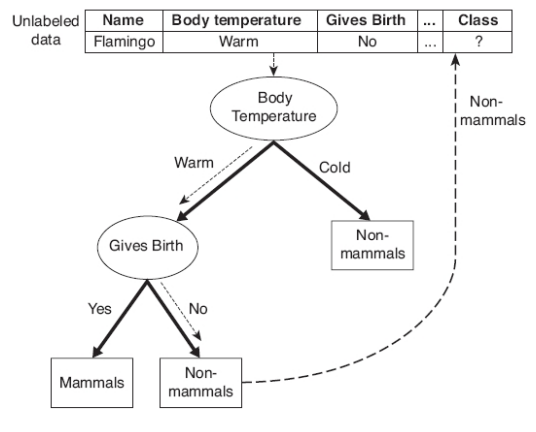
Ein Entscheidungsbaum besteht aus folgenden Elementen:

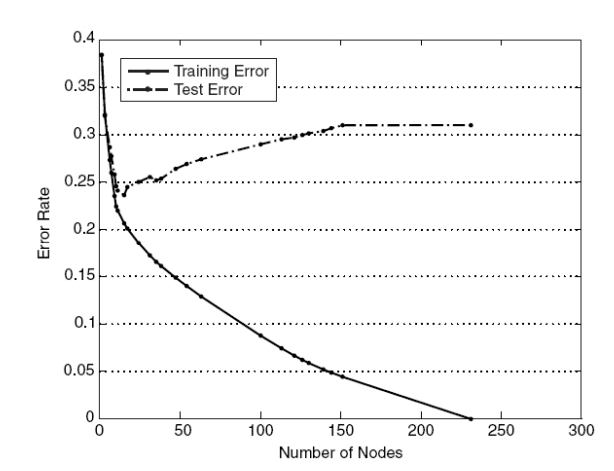
1. Wurzelknoten: Der oberste Knoten des Baums, von dem aus der Baum beginnt. Er stellt die erste Bedingung dar, anhand derer eine Entscheidung getroffen wird.
2. Innere Knoten: Die nachfolgenden Knoten im Baum, die weitere Bedingungen repräsentieren. Sie teilen den Datenfluss basierend auf bestimmten Merkmalen auf.
3. Blattknoten: Die Endknoten des Baums, die die Vorhersagen oder Entscheidungen repräsentieren. Sie enthalten die Ausgabe oder Klasse, die dem gegebenen Datenpunkt zugeordnet wird.

Die Konstruktion eines Entscheidungsbaums erfolgt in der Regel durch rekursive Teilung des Datensatzes basierend auf bestimmten Merkmalen. Der Algorithmus wählt die besten Bedingungen aus, um die Daten in den einzelnen Schritten aufzuteilen, wobei das Ziel darin besteht, die homogensten Teilmengen an Daten zu erhalten.

Entscheidungsbäume bieten einige Vorteile, wie z. B. ihre Fähigkeit, komplexe Entscheidungsregeln abzubilden, ihre Interpretierbarkeit und ihre Fähigkeit, mit numerischen und kategorischen Daten umzugehen. Sie können in verschiedenen Aufgaben eingesetzt werden, wie beispielsweise Klassifikation, Regression oder auch bei Entscheidungsproblemen.

Ein wichtiger Aspekt bei der Verwendung von Entscheidungsbäumen ist die Vermeidung von Überanpassung (**Overfitting**). Überanpassung tritt auf, wenn der Baum zu stark auf die Trainingsdaten angepasst ist und nicht in der Lage ist, auf neuen Daten gut zu generalisieren. Techniken wie **Pruning**, Regularisierung oder Ensemble-Methoden wie Random Forests können verwendet werden, um Überanpassung zu reduzieren und die Leistung des Entscheidungsbaums zu verbessern.

**Unsupervised learning** beschreibt wiederum einen anderen Ansatz. Ziel dieser Lernmethode ist es verborgene Muster, Strukturen oder Beziehungen in den Daten zu finden und diese zu gruppieren oder zu segmentieren, um wertvolle Einblicke und Erkenntnisse zu gewinnen.



Re-substitution errors: error on training ( e(t) )

• Generalization errors: error on testing ( e’(t))

• Methods for estimating generalization errors:

– Optimistic approach: e’(t) = e(t)

– Pessimistic approach:

• For each leaf node: e’(t) = (e(t)+0.5)

• Total errors: e’(T) = e(T) + N \* 0.5 (N: number of leaf nodes)

• For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):

Training error = 10/1000 = 1%

Generalization error = (10 + 30\*0.5)/1000 = 2.5%

– Reduced error pruning (REP):

• uses validation data set to estimate generalization error

Ocams Razor:

• Given two models of similar generalization errors, one should

prefer the simpler model over the more complex model

• For complex models, there is a greater chance that it was

fitted accidentally by errors in data

• Therefore, one should include model complexity when

evaluating a model

