Data Mining Zusammenfassung

Maximilian Ortwein

10. Februar 2013

Inhaltsverzeichnis

1 Data Understanding

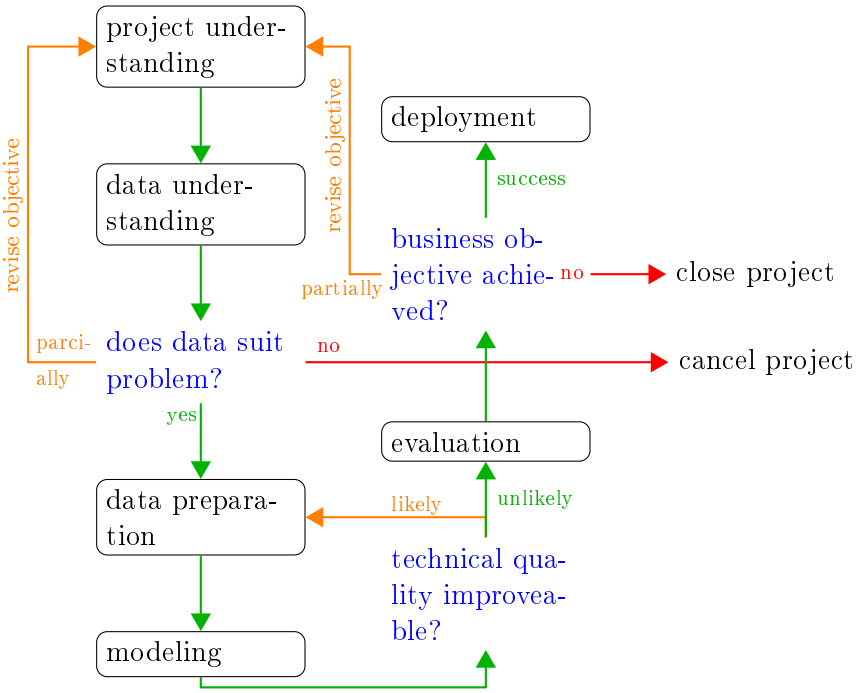
Project Understanding

- Problemformulierung → Mapping auf Datamining Task →
  Verstehen der Situation
- 80-20 Rule: 20% Wird in Data und Project Understanding verwendet, ist aber zu 80% ausschlaggebend für den Erfolg.

Data Understanding

- Questions:
  1. Welche Arten von Attributen haben wir?
  2. Wie ist die Qualität der Daten?
  3. Hilft eine Visualisierung?
  4. Korrelieren die Attribute?
  5. Gibt es Ausreißer?
  6. Wie sollen *missing values* behandelt werden?
- Reihen sind Instanzen, Objekte oder Records, Spalten sind Attribute, Eigenschaften oder Werte.
- Datentypen:
  1. Nominal (Klassen oder Kategorien; meist String)
  2. Ordinal(Lineare Ordnung; Schulnoten oder Temperaturen)
  3. Numeric (Zahlen)
  4. discrete (Numeric oder Ordinal als Teilmenge von Integern)
  5. continous (Reelle Zahlen)
- Data Quality:
  1. Garbage in, garbage out
  2. Accuracy (Genauigkeit) = Nähe des Wertes aus den Daten am realen Wert.
  3. Geringe Genauigkeit durch: Noise, fehlerhafte Eingaben, Tippfehler, ...
  4. **Syntaktische Genauigkeit:** Eintrag ist nicht in der Domain, z.B. Text in numerischen Daten. Einfach überprüfbar
  5. **Semantische Genauigkeit:** Eintrag ist in der Domain aber fehlerhaft z.B. John Smith Female. Überprüfung aufwändiger
  6. **Completeness:** Ist verletzt wenn die Daten nicht vollständig sind und dadurch verzerrt (biased)
  7. **Unbalanced Data:** Wenn eine Art von Einträgen extrem verrauscht ist.
  8. **Timeliness:** Sind die Daten aktuell?

1.1 Der Datamining Cycle



1.2 Data Vizualization

So bekommt man einen schnellen Überblick über die Daten und erkennt zum Beispiel Verzerrungen oder fehlende Werte.

1.2.1 Types

- Bar Charts/Histograms: Anzahl der Bins: Sturges rule $k = \lceil log_2(n) + 1 \rceil$
- Boxplots: Boxgröße = Interquartilsabstand, Median einzeichnen als Linie, Antennen: jeweils 1.5 facher Interquartilsabstand
- Scatterplots

1.2.2 Correlation Analysis

- Pearsons R: $r = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum(x_i - \overline{x})^2 \sum(y_i - \overline{y})^2}}$
  - Pearsons erkennt lineare Korrelation
  - Auch für monotone nicht lineare Korrelationen ist er nicht -1 oder 1
  - Er kann auch fast null sein trotz einer monotonen Korrelation
  - Lösung: Rang Koeffizient
- Spearmans Rho: $\rho = 1 - 6 \cdot \frac{\sum_{i=1}^{n}(r(x_i) - r(y_i))^2}{n(n^2-1)}$

1.2.3 Outlier Detection

Ausreißer sind Datenpunkte die weit entfernt vom Rest liegen. Verursacht werden sie Durch schlechte Qualität oder ungewöhnliche Extremwerte.
Handling: Meist ist es Sinnvoll diese Daten aus der Analyse auszuschließen. Wenn sie durch Fehler verursacht wurden auf jeden Fall.
Manchmal können Ausreißer auch das Ziel einer Analyse sein.
Für die Erkennung kann man Grubbs Test oder Boxplots verwenden.

1.2.4 Missing Values

Gründe für Missing Values sind fehlerhafte Sensoren, Verweigerung einer Antwort, oder unsinnige Antworten unter bestimmten Bedingungen.
Behandeln kann man sie in dem man *null* Werte oder Standartwerte (Median, wahrscheinlichster Wert usw.) einsetzt.

1.2.5 Checklist MUST DO

- Die Verteilung eines Attributes überprüfen
- Korrelationen und Zusammenhänge aufdecken

1.3 Visualizing multidimensional data

1.3.1 Types

- 3D-Scatterplot
- Parallel Coordinates
- Radarplot
- Star Plot

1.3.2 Outlier Detection

- Scatterplots
- Visualize PCA or MDS
- Clustering: Punkte die keinem Cluster angehören sind Outlier

1.3.3 MDS Multi Dimensional Scaling

Positioniert die Datenpunkte im 2D-Raum und nutzt dabei die Abstände im n-D Raum um die Struktur zu erhalten.
Braucht extrem viele Parameter für das Iris Set z.B. 300 da es für jeden Datenpunkt die Position x und y bestimmen muss. Zum Minimieren der Funktion wird das *gradient descent* Verfahren verwendet.

2 Version Space

- Syntaktisch unterschiedlicher Hypothesenraum wird mit (k+2) multipliziert.
- Semantisch unterschiedlicher Hypothesenraum wird durch $(k_1+1)\cdot(k_2+2)\cdots(k_n+n)$ berechnet

2.1 Candidate Elimination

Anfang:
$S = \{< \emptyset, \emptyset, \emptyset >\}$
$G = \{<?, ?, ? >\}$
Dann werden für jedes Positive oder Negative Beispiel S und G so angepasst das es akzeptiert wird oder nicht. S wird verallgemeinert, G wird spezialisiert. Die Reihenfolge der Beispiele hat dabei keinen Einfluss auf das Ergebnis.
Hierbei kann es auch sein, dass G zum Schluss mehr als nur ein Tupel enthält.
HIER EVENTUELL ERGÄNZEN!!!!!!!!!!

3 Data Preparation

3.1 Feature Selection

Irrelevante und redundante Features entfernen

- Wähle die Features mit der besten Bewertung wenn einzelne Features evaluiert werden.
- Wähle das bestes Subset aus. Dies ist sehr kostenintensiv und nur für sehr kleine Mengen möglich.
- Forward Selection: Starte mit einer leeren Menge und füge immer das Feature hinzu das die Genauigkeit am meisten erhöht.
- Backward Elimination: Starte mit allen Features und entferne alle ungeeigneten.

3.2 Record Selection

Gründe für Subsamples:

- Schnellere Berechnung
- *Crossvalidation* mit einen Trainings- und einem Testset
- *Timeliness*: veraltete Daten können entfernt werden
- *Represenetativeness*: finde ein repräsentatives Subsample
- *Rare Events* müssen insbesondere mit einbezogen werden

3.3 Data Cleansing

Finde und korrigiere oder entferne ungenaue, inkorrekte oder unvollständige Einträge aus dem Datensatz.

3.4 improve data quality

1. Alle Buchstaben zu Großbuchstaben ändern.
2. Spaces und unsichtbare Zeichen entfernen.
3. Format von Zahlen anpassen.
4. Aufteilen von Spalten mit mehreren Informationen.
5. Abkürzungen entfernen und Rechtschreibung korrigieren.
6. Schreibweise von Adressen vereinheitlichen.
7. Zahlen in Standardformat bringen.
8. Überprüfe durch Wörterbücher ob die Werte in die Domäne passen.

3.5 Missing Values

- Entferne den Eintrag
- Ersetze den Wert (Median, Mittelwert...)
- Ersetze den Wert durch einen speziellen Wert

3.6 Transformation of Data

3.6.1 kategorisch zu numerisch

- Binary: 1 und 0
- Ordinal: In ihrer Ordnung nummerieren z.B. 1 - k
- Categorical: Sollten nicht in eine einzelne Zahl konvertiert werden.

3.6.2 numerisch zu kategorisch

Einen numerischen Bereich in Bins aufteilen.

- *Equi-Width discretization*: Teile die Intervalle in Bins mit gleicher Breite auf.
- *Equi-frequency discretization*: Teile die Daten in Bins mit der gleichen Anzahl an Elementen auf.
- *V-optimal discretization*
- *Minimal entropy discretization*: Minimieren der Entropy. (Nur anwendbar in Fall eines Klassifizierungsproblems.)

3.7 Normalisierung

Für manche Datenanalysen (PCA, MDS Clustering) ist die Skala der Daten entscheidend. Deshalb ist es notwendig die Daten auf eine Standardskala zu mappen. Normalerweise werden die Daten auf einen Bereich zwischen 0 und 1 gemappt.

3.8 Principal Component Analysis PCA

Erzeugt eine Projektion aus einem hochdimensionalen Raum in einen niederdimensionalen Raum. Es nutzt die Varianz der Daten zur Strukturerhaltung. Es versucht möglichst viel der Originalvarianz zu erhalten.

4 Modeling

1. Auswählen der *model class*
2. Auswählen der *score function*
3. Anwenden des Algorithmus
4. Validieren des Ergebnisses

Wähle das Einfachste Model was die Daten noch erklärt!
Globale Modelle (Regression) zeigen das komplette Dataset, lokale Modelle (Association Rules) nur ein Subset.

4.1 Errorfunctions

Falschklassifiziertenrate: $\frac{\#\text{falsch klassifiziert}}{\#\text{alle Daten}}$
Sagt nichts über die Qualität des Classifiers aus.
Klassen können nicht balanciert sein.

4.2 Cost Matrix

Ein generellerer Ansatz als die Errorfunktion.
Die Kosten einer falschen Klassifikation können für jede Klasse anders sein.
Deshalb werden sie für jede Klasse in eine Matrix geschrieben.

4.3 Cross Validation

Teile die Daten in k Teile. Dann nutze immer einen Teil als Testdatensatz den Rest als Trainingsdatensatz.
Die durchschnittliche Fehlerrate ist die Fehlerrate des Modells.

4.4 MDL (Minimum Description Length Principle)

Ein zu komplexes Modell führt oft zu Overfitting (Das Modell ist zu stark auf die Trainingsdaten angepasst).
Die MDL besagt, wähle das Modell, das bei gleicher Vorhersagequalität, das Modell mit den wenigsten Trainingsdaten ist.

5 Clustering

- Objekte eines Clusters sollten möglichst gleich sein, und Objekte unterschiedlicher Cluster möglichst verschieden.
- Cluster können unterschiedliche Formen, Größen und Dichten haben.
- Können eine Hirarchie bilden
- Können sich überschneiden
- Data Understanding, Class Identification, Reduction (Repräsentanten finden), Outlier Detection, Noise Detection
- Vor dem Clustern sollten die Daten normalisiert werden.
- Cluster sollten nicht von Maßeinheiten abhängen!

Typen:

- Linkage Based
- by Partitions
- Desity based
- grid based

5.1 Hierarchical Clustering

Cluster werden Schritt für Schritt aufgebaut normalerweise *Bottom up*. Das heißt jeder Datenpunkt ist ein Cluster und wird dann in jedem Schritt verschmolzen (*agglomerative*).
Das Gegenteil ist die *Top Down* Strategie, alle Daten sind in einem Cluster und werden zerlegt (*divisive*).
Für die Entscheidung welcher Datenpunkt zu welchem Cluster gehört, muss man die *Similarity* (Ähnlichkeit) messen.

5.1.1 Dissimilarity (Abstände)

- Centroid: Der Abstand zwischen den Mittelpunkten der Cluster.
- Average: Der durchschnittliche Abstand aller Punkte der Cluster
- Single Linkage: der Abstand der zwei am engsten beieinander liegenden Punkte der Cluster. (Kann Ketten verursachen!)
- Complete Linkage: Abstand der am weitesten entfernten Punkte der Cluster

5.2 Dendograms

Sind binäre Bäume. Unten oder links kommen die Datenpunkte hin. Die Cluster werden miteinander verbunden und es wird bei der Verbindung der Abstand zwischen den Clustern eingezeichnet.

5.3 k-Means

1. bestimme die Anzahl von Clustern k (Ist eine Benutzereingabe)
2. wähle zufällig k Datenpunkte als Anfangsmittelpunkte
3. weise jedem Datenpunkt sein Zentrum zu (das mit dem kleinsten Abstand)
4. Berechne die Zentren neu als Durchschnitt aller Punkte eines Clusters
5. Wiederhole ab Schritt 3 bis keine Änderungen mehr auftreten

5.4 DBSCAN

Dichtebasierte Clusteringalgorithmen haben oft die besten Ergebnisse auf numerische Daten. Sie weisen die Regionen mit hohen Dichten einem Cluster zu.

- setze ϵ als größter Abstand von einem Punkt (Epsilonumgebung)
- setze $minPts$ als die minimaler Anzahl von Datenpunkten in der Epsilonumgebung (Knoten selbst zählt auch dazu!)
- Finde einen Datenpunkt mit einer hohen Dichte um sich
- Alle e Nachbarn gehören zu dem Cluster
- So lange die Dichte hoch ist, erweitere das Cluster
- entferne das Cluster aus dem Datensatz und fange wieder von vorne an

6 Association Rule

6.1 frequent itemset mining

- Support: $\frac{\#\text{Transaktionen in denen das Itemset vor kommt}}{\#\text{alle Transaktionen}}$
- Confidence: $\frac{support(X \cup Y)}{support(X)}$

6.2 Searching for frequent itemsets

6.2.1 Apriori

Nimmt den MinSupport entgegen.
Es werden nach und nach alle Itemsets ausgeschlossen, die den MinSupport nicht erfüllen. Man fängt bei einzelnen Items an und schließt nach und nach noch verbleibende Kombinationen aus. Alle übrigbleibenden Sets sind dann Regeln.
Immer die ersten k-2 (k = Anzahl der Elemente des nun zu bildenden Itemsets) Elemente eines Itemsets müssen gleich sein zum verschmelzen.
Pruning: (Beispiel)
$\{A, B, C\}, \{A, B, D\} \Rightarrow \{A, B, C, D\}$
Da aber $\{B, C, D\}$ nicht vorhanden ist, fällt auch $\{A, B, C, D\}$ wieder raus.

6.2.2 Hassediagramm/Baum

Wenn man jedem Knoten in einem Hassediagramm nur einen Elternknoten zuweist, zum Beispiel das erste Item, erhält man den Baum, dieser hat den Vorteil das bei einer Traversierung jeder Knoten genau einmal besucht werden muss.

6.2.3 kanonische Form

Eine kanonische Form ist dann gegeben wenn die Items lexikografisch geordnet sind: aus einer Menge $\{A, B, C, D\}$ wären ABC kanonisch ACB aber nicht.

6.2.4 Kanonische Präfix Regel

Jeder Präfix einer kanonischen Form ist selbst kanonisch!
Mit dieser Regel können Itemsets einfach rekursiv durchsucht werden. Es können einfach durch Hinzufügen eines Items an das Ende eines kanonischen Wortes alle daraus resultierenden kanonischen gefunden werden. Dazu muss nur noch überprüft werden ob das neue Wort kanonisch ist.

6.2.5 Frequent, Closed, Maximal

Frequent: Itemsets, die den Minimum Support erfüllen.
Closed: Itemsets, die keine Supersets mit gleichem oder größerem Support haben.
Maximal: Itemsets die keine Supersets mehr haben die Frequent sind.

7 Bayes + Regression

7.1 Bayes Theorem

$P(h|E) = \frac{P(h|E) \cdot P(h)}{P(E)}$ Es wird immer die höchste Wahrscheinlichkeit genommen.
Naive Bayes heißt, es wird davon ausgegangen, dass die Ereignisse/Attribute voneinander unabhängig sind.
LaPlace Correction kann Divisionen durch 0 entfernen.
Pros:

- Gold-Standard für den Vergleich mit anderen Klassifikatoren
- hohe Klassifizierungsgenauigkeit in vielen Anwendungen
- Klassifikator kann leicht an neue Trainings-Objekte angepasst werden
- Integration von *domain knowledge*

Cons:

- die bedingten Wahrscheinlichkeiten müssen nicht immer verfügbar sein
- unabhängige Annahmen gelten nicht unbedingt für den Datensatz

7.2 Regression

7.2.1 Regression Line

Eine Lineare Funktion die sich den Daten möglichst gut annähert.
$y = a \cdot x + b$
$b = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n} (x_i - \overline{x})^2}$
$a = \overline{y} - b\overline{x}$

7.2.2 Overfitting

Komplexe Funktionen neigen zum Overfitting. Das heißt die Funktion ist zu stark an die Trainingsdaten angepasst und daher dann schlechtere Vorhersagen liefert.
Pros:

- starke mathematische Grundlagen
- einfach zu berechnen und zu verstehen (für mäßige Anzahl an Dimensionen)
- hohe Vorhersagegenauigkeit

Cons:

- viele Abhängigkeiten sind nicht linear
- globale Modelle passen sich nicht den anderen lokalen Verteilungen an
- $\Rightarrow$ lokal gewichtete Regression

8 Decision Tree

8.1 ID3 Algorythmus

- Berechne die Entropy aller Attribute
- Das Attribut mit der höchsten Entropy wird die Wurzel
- Führe das für alle Teilbäume aus
- den Baum dadurch rekursiv aufbauen

8.2 entropy

Entropy: $H = -\sum_{i=1}^{n} p_i \log p_i$
Information Gain = H(Class) - H(Class|Atrribut)
$H(C|A) = -\sum_{j=1}^{An} p_j - (\sum_{i=1}^{Cn} p_{i|j} \log p_{i|j})$
Gini Index: $1 - \sum_{j=1}^{An} p_j$

8.3 prunning

Schlechte Zweige abschneiden und durch ein Blatt austauschen.
Schützt vor *Overfitting*.

8.4 Regression Tree

Damit werden numerische Werte vorhergesagt statt Klassen.
Ähnlich aufgebaut wie Entscheidungsbäume.

9 Rule Learning

9.1 Propositional Rules

Nimmt nur atomistische Fakten.
Zur Kombination werden logische Operationen verwendet.
Es gibt keine Variablen.
IF $x_1 < 100 \wedge x_2 = 4$ THEN Class D

9.1.1 Finding Proposional Rules

Extracting Rules from Trees
Mann kann die Regeln direkt aus dem Baum ableiten.
Die Regeln sind *Mutual Exclusive* und konfliktfrei, ungeordnet und vollständig.
Allerdings sind die Regeln instabil und redundant.

9.1.2 Learning Propositional Rules

Generalization
Verschmelze zwei Regeln zu einer neuen Regel oder erweitere eine Regel um ein weiteres Pattern
Specialisation
Fange mit einer generellen Regel an und spezialisier sie nach und nach.

9.2 Geometrical Rule Learners

9.2.1 RecBF

Der RecBF Algorithmus erstellt Regeln als Rechtecke. Der Learner erstellt Rechtecke in denen gerade so alle Datenpunkte die zu einer Klasse gehören liegen aber keine einer anderen Klasse. Der Predictor überprüft in welchem Rechteck ein gegebener Punkt liegt und kann ihn somit klassifizieren.
Operationen:

- *Rule Stability*: Wenn die Daten konfliktfrei sind terminiert der Algorithmus.
- *Covered*: Überprüft ob neue Regeln die Grenzen abdecken.
- *Commit*: Einfügen neuer Regeln.
- *Shrink*: Verkleinere die Rechtecke, damit keine Konflikte entstehen.

9.3 CN2

Bekanntes Beispiel einer der ersten Rule Learning Algorithmen. Er hat eine einfach Heuristik für das Finden der wichtigsten Rule.
Der Algorithmus nimmt einen Trainingsdatensatz entgegen und speichert in einer Liste alle Regeln. In einer Schleife über die verbleibenden Trainingsdaten (am Anfang über alle), sucht er eine gute Regel die einen Teil der Daten abdeckt und fügt sie der Liste hinzu. Dann werden alle Daten aus dem Training entfernt die durch die Regel abgedeckt werden. Das geht solange bis der *Performance Grenzwert* erreicht ist. Dann wird die Liste mit Regeln zurückgegeben. Es wird eine *Beam search* verwendet um eine gute Regel zu finden. Mit der relativen Häufigkeit könnte man eine gute Regel finden:

$$p(klasse|condition) = \frac{\#\text{richtig abgedekt}}{\#\text{aller Abgedeckten}}$$

LaPlace:

$$p(klasse|R) = \frac{\#\text{Richtig abgedekt} + 1}{\#\text{aller Abgedeckten} + \#\text{classes}}$$

m-Estimates:

$$p(klasse|R) = \frac{\#\text{richtig abgedekt} + m \cdot p(klasse)}{\#\text{aller Abgedeckten} + m}$$

Diese heuristischen Methoden schlagen auf Daten aus der echten Welt relativ oft fehl. Scharfe Grenzen sind für *Noisy* Daten schlecht.
Die Regeln sind oft nicht aussagekräftig.

9.4 First Order Rules

Verwenden im Gegensatz zu Propositional Rules Variablen.
Es gibt: Konstanten, Variablen, Prädikate, Funktionen.
Daraus können: Terme, Literale, Ground Literals, Klauseln und horn clauses gebaut werden.
Horn Clauses haben mindestens ein positives Literal. IF L_1 AND ... AND L_n THEN H.
Sie werden auch in Prolog verwendet.
Eine Regel ist erfüllt wenn es mindestens eine Bindung gibt die alle Literale erfüllt.

9.4.1 Learning First Order Rules: FOIL

Es gibt keine Funktionen!
FOIL kombiniert *outer Specific to General* und *inner General to Specific.*
FOIL kann auch rekursive Konzepte lernen.
Durch Erweiterungen kann man auch Noisy Data verarbeiten.

10 Neuronale Netzwerke

Neuronen:
Neuronen sind Schalter mit zwei Zuständen und haben einen festen Grenzwert ab dem sie Schalten. Ein Neuron bekommt Input von Synapsen (Verbindungen zu anderen Neuronen). Wenn der Grenzwert eines Neurons erreicht ist wird es aktiv und sendet ein Signal an die angeschlossenen Neuronen.

10.1 Einfache Perceptrons

Ein einfaches Perceptron besteht aus einer Eingabeschicht aus Eingabeneuronen und einem Ausgabeneuron. Es kann AND, OR und NOT darstellen. In den Eingabeneuronen wird die Eingabe vorsortiert und ein Gewicht berechnet (1 für positiv, 0 für neutral und -1 für negativ) diese Gewichte werden im Ausgabeneuron aufsummiert. Wenn diese Summe den Grenzwert erreicht oder überschreitet feuert das Neuron, wird also aktiv. Durch dieses Gebilde kann man Daten in zwei Klassen einteilen.

Zum Trainieren werden zufällige Gewichte verteilt, wenn falsch klassifiziert wird müssen die Gewichte und der Grenzwert angepasst werden.

10.1.1 The Delta Rule

Wenn das Ergebnis 0 ist aber 1 sein sollte verringere den Grenzwert und passe die Gewichte nach Vorzeichen an.
Wenn das Ergebnis 1 ist aber 0 sein sollte erhöhe den Grenzwert und passe die Gewichte an.

10.1.2 Perceptron Convergence

Wenn es für einen Datensatz mit zwei Klassen ein Perceptron gibt, dann passt es die Delta Regel so an das alle Daten richtig vorhergesagt werden.

10.1.3 Linear Seperability

Ein Perceptron mit n Eingaben kann alle Daten eines Datensatzes in zwei Klassen aufteilen wenn es eine Hyperebene gibt die die beiden Klassen teilt.

10.2 XOR

Ein XOR kann nur mit einer zweiten Schicht gebaut werden.
Diese zweite Schicht, macht aus dem XOR ein lineares Problem was dann durch das *output perceptron* gelöst wird.

10.3 Multilayer Perceptrons

Um die Gewichte der *Hidden Layer* zu berechnen verwendet man Multilayerperceptrons die die Gewichte mit dem Gradientenverfahren bestimmen. Die Grenzwertfunktion muss ableitbar sein.
Multilayer Perceptrons besitzen einen Input, einen Output und mehrere versteckte Schichten. Jede Schicht ist nur mit der nächsten Schicht verbunden.
Mit *Backpropagation* werden Minima gefunden, es ist eine Gradient Decent Methode.

11 SVM

Suport Vektor Machines, Klassifiezieren Daten indem sie zwischen den Daten eine Hyperebene einzieht die die Daten in Klassen aufteilt. Die Datenpunkte liegen alle als Vektoren vor. Die Hyperebene wird nur anhand der ihr am nächsten liegenden Datenpunkten berechnet den Sogenannten Support Vectoren. Der Abstand zwischen den Vectoren und der Hyperebene wird maximiert um Platz für vorherzusagende Datenpunkte zu schaffen. Die SVM funktioniert aber nur für linear lösbare Probleme. Da das in der Praxis meist nicht der Fall ist gibt es den Kernel Trick um nicht lineare Probleme zu lösen. Die Idee ist die Daten in eine höhere Dimension zu transformieren in der das Problem linear lösbar ist. Dazu wird die Kernel Funktion verwendet. Spätestens im unendlich hoch dimensionalen Raum kann man die SVM anwenden.

12 k-nearest-nighbor

KNN ist ein Lazy Algorithmus das heisst die Lernphase besteht nur darin die Trainingsdaten zu Speichern. Die ganze Arbeit wird bei der Vorhersage gemacht.

12.1 1-NN

Suche für den gegebenen Datenpunkt den Punkt der ihm aus den Trainingsdaten am nächsten liegt. Weise dem Punkt diese Klasse zu.

12.2 KNN

Suche für einen Datenpunkt die k am nächsten benachbarten Punkte.
- Weise dem Punkt die häufigste Klasse der k-Punkte zu.
- Man kann die Klassen auch gewichten nach der nähe zum Datenpunkt. Zum Beispiel in dem man die Distanz invertiert. Dann hat der nächste Trainingspunkt einen höheren Einfluss auf die Klasse als der am weitesten entfernte.

Zur Bestimmung eines optimalen k werden Methoden wie Cross-Field-Validation angewendet. Meist sind kleine k sehr anfällig für Noise und zu große k Klassifizieren die Daten falsch oder in nur noch eine Klasse.