# 2 Grundlagen des Data Mining

*If you file it, you'll know where it is but you'll never need it. If you don't file it, you'll need it but never know where it is.*
Tillis's Organizational Principle

## 2.1 Grundbegriffe

Zunächst definieren wir einige Grundbegriffe, die für ein Verständnis des Gebiets *Data Mining* unerlässlich sind. Zentral ist ein Verständnis der Begriffe *Daten*, *Information* und *Wissen*.

In unserem Buch werden wir von einer Begriffshierarchie ausgehen, wie sie sich in der Fachliteratur durchgesetzt hat. Die Grundlage der IT-basierten Verarbeitung sind *Daten*. Hat ein Datum, beispielsweise die Zahl $-4$, eine Bedeutung, dann wird sie zu einer *Information*. Handelt es sich beispielsweise um die Veränderung eines Aktienkurses (in Prozent), so wird die $-4$ zu einer Information. Haben wir nun zum Beispiel eine Regel, die einen Verkauf dieser Aktie auslöst, so handelt es sich um *Wissen*.

In anderen Quellen wird der Begriff *Information* an die Spitze dieser Hierarchie gesetzt. Die Begriffshierarchie kann auch um die Begriffe *Signal* oder *Zeichen* erweitert werden, die dann unterhalb des Begriffs *Daten* einzuordnen sind.

**Definition 2.1:** *Daten*

Mit **Daten** bezeichnet man eine Ansammlung von Zeichen mit der dazugehörigen Syntax.

*Daten* ist der Plural des Wortes *Datum*, welches im deutschen Sprachgebrauch eine Kalender- oder Zeitangabe ist. Wir werden hier die weniger gebräuchliche, nichtsdestoweniger aber gültige Interpretation eines Datums als eine *Informationseinheit* verwenden.

Man unterscheidet

- Unstrukturierte Daten
- Semistrukturierte Daten
- Strukturierte Daten

Typische Beispiele für *unstrukturierte* Daten sind Bilder oder Texte. Data Mining auf diesen Daten erweist sich als schwierig, da man im Allgemeinen zunächst aus den unstrukturierten Daten strukturierte herstellen muss.

Web-Seiten bestehen zwar überwiegend aus Text, was sie in die Kategorie der unstrukturierten Daten platzieren würde. Allerdings weisen Web-Seiten eine Struktur auf, so dass man sie als *semistrukturierte* Daten bezeichnet.

Schwerpunkt dieses Buchs sind *strukturierte* Daten. Unter strukturierten Daten versteht man meistens relationale Datenbank-Tabellen oder Daten in ähnlich strukturierten Datei-Formaten. Dazu zählen auch Datei-Formate einer Tabellenkalkulation oder insbesondere auch das Austauschformat CSV (comma-separated values), welches als ASCII-Text direkt mit einem Editor bearbeitet werden kann. Diese Formate weisen eine feste Struktur auf: Die in jedem Datensatz enthaltenen Daten haben eine feste Reihenfolge, die Attribute sind definiert, die Datentypen sind festgelegt.

Aus Daten entstehen *Informationen* dadurch, dass diese Daten eine Bedeutung bekommen.

**Definition 2.2:** *Information*

Eine **Information** ist ein Datum, welches mit einer Bedeutung gekoppelt ist.

Eine Information ist also die zweckbestimmte Interpretation von Daten. Daten bestehen zunächst nur aus Fakten und werden erst dann zur Information, wenn sie im Kontext betrachtet werden und eine Bedeutung erhalten.

**Definition 2.3:** *Wissen*

Eine Information in Verbindung mit der Fähigkeit, diese zu benutzen, wird als **Wissen** bezeichnet.

Eine Information wird folglich erst dann zu *Wissen*, wenn man mit ihr etwas anzufangen weiß. „Ein System S hat Wissen W, wenn S immer dann – wenn erforderlich – W anwendet.“ ([MN73], zitiert nach [Lau85]).

Nun können wir definieren, was wir unter Data Mining verstehen.

**Definition 2.4:** *Data Mining*

**Data Mining** (Datenschürfen) ist die Extraktion von Wissen aus Daten.

Wir sollten diese Definition um einige Forderungen erweitern. Wir möchten Wissen erhalten, welches bisher unbekannt war. Es geht also um die Extraktion von implizitem Wissen. Dieses Wissen sollte nicht trivial, sondern nützlich sein, wir müssen es also sinnvoll anwenden können. Data Mining sollte auch weitgehend automatisch ablaufen. Wer einige Erfahrung im Data-Mining-Bereich hat, weiß, dass die letzte Forderung bewusst durch das Wort *weitgehend* abgeschwächt wurde. Die eigentliche Datenanalyse

wird meistens automatisch ablaufen, aber die Vorbereitung bedarf einer intensiven Unterstützung durch den Analysten.

Nach der Definition des Begriffs *Data Mining* betrachten wir die Teilgebiete des Data Minings. Gemäß der Unterteilung in unstrukturierte, semistrukturierte und strukturierte Daten kategorisiert man Teilgebiete des Data Minings wie folgt:

- Text Mining (unstrukturierte Daten)
- Web Mining (semistrukturierte Daten)
- Data Mining im engeren Sinn (strukturierte Daten)

*Text Mining* befasst sich mit der Analyse von Texten, also von unstrukturierten Daten. Web Mining arbeitet im Allgemeinen auf den bereits diskutierten semistrukturierten Daten. *Data Mining im engeren Sinn* befasst sich dagegen mit strukturierten Daten. Häufig wird der Begriff Data Mining als Synonym für das Data Mining im engeren Sinn verwendet. Wir werden uns auf das Data Mining im engeren Sinne beschränken und nur vereinzelt auf die anderen beiden Teilgebiete eingehen.

Wir werden die Begriffe *Datensatz*, *Instanz*, *Objekt* oder *Muster* weitgehend synonym verwenden. Damit ist stets *ein* Datensatz in einer Datenbanktabelle gemeint, der ein Objekt oder eine Instanz anhand einer Menge von Merkmalswerten charakterisiert: Den Datensatz

```
sunny, hot, high, false, no
```

kann man mathematisch als Quintupel

```
(sunny, hot, high, false, no)
```

und damit auch als *ein* Objekt auffassen. Dieses Objekt ist zudem eine Instanz der Klasse aller Wetter-Situationen.

## 2.2 Datentypen

Nach der Diskussion, welche Daten vorliegen können (strukturiert, semistrukturiert oder unstrukturiert), betrachten wir die unterschiedlichen Datentypen, mit denen Data Mining konfrontiert sein kann.

Daten können sehr unterschiedlich sein. Wir haben täglich mit Zahlen zu tun. Mit Zahlen können wir rechnen. Wir können Mittelwerte berechnen und Zahlen miteinander vergleichen und sie bezüglich ihrer Größe ordnen. Ebenso sind aber auch Daten der Form *klein, mittelgroß, groß, sehr groß* möglich. Diese haben – wie die Zahlen – eine Ordnung, wir können sie bezüglich ihrer Größe vergleichen und folglich sortieren. Rechnen können wir natürlich mit Daten von diesem Typ nicht. Und dann haben wir Datentypen wie den Datentyp *Farbe*. Hier haben wir leider keine Ordnung. Wir können bei den Farben

*braun* und *schwarz* nicht sagen, welche besser, größer oder bunter ist. Und ein Rechnen ist hier natürlich nicht möglich.

Mit diesen zwei Kriterien – Ordnung und Rechnen – können wir nun eine erste Kategorisierung von Datentypen vornehmen:

**Nominale** Daten unterliegen keinerlei Rangfolge. Sie können lediglich nach dem Kriterium *gleich* beziehungsweise *nicht gleich* sortiert werden.

**Ordinale** Daten haben zumindest eine Ordnungsrelation (wie $<$). Rechnen kann man mit ihnen aber nicht.

**Metrische** Daten besitzen alle Ordnungsmerkmale der reellen Zahlen. Man kann mit ihnen „rechnen“.

Metrische Datentypen können weiter unterteilt werden:

**Diskrete** Datentypen sind solche, die nur eine endliche Anzahl von Werten annehmen können. Die Schulnoten – wenn wir von den Noten 1, 2, 3, 4, 5 ausgehen – sind ein solcher Datentyp.

**Kontinuierliche** Datentypen sind numerische Daten, die jeden beliebigen Zahlenwert innerhalb des Definitionsbereichs annehmen können.

Auch kontinuierliche Daten sind genau genommen im Computer diskret, da sie durch die begrenzte Darstellung auf Computern nicht exakt wiedergegeben werden können. Diese Nuance werden wir aber ignorieren und davon ausgehen, dass solche Daten beliebig genau darstellbar seien.

Kontinuierliche, numerische Datentypen können in diskrete Daten umgewandelt werden, beispielsweise durch Intervallbildung. Man spricht dann von *diskretisierten* Datentypen.

Selbst wenn die Werte an sich numerisch sind, muss es sich nicht zwingend um einen metrischen Datentyp handeln. Kundennummern sind zwar Zahlen, vom Typ her sind sie aber nominale Datentypen. Falls die Kundennummern in einer zeitlichen Reihenfolge vergeben wurden, dann sind die Kundennummern (aus der Sicht der Zeit) zumindest ordinale Daten, da man sie bezüglich der Zeit ordnen kann.

Numerische Datentypen können auch nach dem Kriterium unterteilt werden, welche Skala ihnen zugrunde liegt.

**Intervallbasierte** Datentypen sind unsere Jahreszahlen. Charakteristisch ist hier, dass der Nullpunkt *willkürlich* festgelegt wurde. Der gregorianische Kalender definiert das Jahr 0 anders als die Unix-Welt, in der die 0 im Jahr 1970 liegt. Ebenso sind unsere Temperaturangaben in Celsius intervallbasiert. Man kann mit diesen Werten aber nur eingeschränkt rechnen. Beispielsweise ist 20°C – physikalisch – eben nicht viermal wärmer als 5°C.

**Verhältnisbasierte** Datentypen haben einen natürlichen Nullpunkt. Entfernungsangaben erlauben nun neben Addition und Subtraktion auch den relativen Vergleich, 60 km ist doppelt so weit wie 30 km. Verhältnisskalen sind aber abhängig von der jeweilig verwendeten Maßeinheit. 60 bedeutet bei einer Maßeinheit km etwas anderes als bei der Maßeinheit cm.

**Absolutskalenbasierte** Datentypen unterscheiden sich von den verhältnisbasierten Datentypen dadurch, dass sie von keiner Maßeinheit abhängen. Beispielsweise gehören alle Datentypen, die etwas zählen, dazu.

**Beispiel 2.1:** *Datentypen*

Beispiele für Datentypen sind:

**nominal** :

- Geschlecht = {weiblich, männlich}
- Haarfarbe = {blond, dunkelblond, braun, rot, schwarz, grau}
- Familienstand = {ledig, verheiratet, verwitwet}

**ordinal** :

- Schulnoten {1, 2, 3, 4, 5}
- Meinung {stimme zu, stimme teilweise zu, stimme nicht zu}
- Verbale Kategorien {sehr reich, reich, durchschnittlich, arm}
- Gesundheitszustand {gesund, krank}
  Dies ist sicherlich ein Grenzfall, da wir hier berücksichtigen, dass „gesund“ besser als „krank“ ist. Soll dies nicht berücksichtigt werden, dann ist es ein nominaler Datentyp.

**metrisch, Intervallskala** :

- Jahreszahlen
- Temperatur in Celsius, Fahrenheit

**metrisch, Verhältnisskala** :

- Messwerte wie Strom, Windgeschwindigkeit
- Entfernung
- Körpergröße

**metrisch, Absolutskala** :

- Lebensjahre
- Zahl der Kinder

Die Grenze zwischen ordinal und metrisch kann durchaus fließend sein. Ordinale Daten sind natürlich nicht metrisch, aber wir können beispielsweise die Schulnoten oder die Erdbebenstärke auch als metrische Attribute interpretieren, indem wir uns von den diskreten Werten lösen und auch Noten wie 1,87 zulassen.

Insbesondere kann man ordinale in metrische Daten umwandeln. Betrachten wir ein Attribut mit den Ausprägungen *klein*, *mittelgroß*, *groß*, *sehr groß*, so können wir dies leicht in ein metrisches Attribut umwandeln, indem wir die Werte durch Zahlen ersetzen:

- *klein* $\rightarrow$ 0
- *mittelgroß* $\rightarrow$ 0,3
- *groß* $\rightarrow$ 0,7
- *sehr groß* $\rightarrow$ 1

Bei der Wahl der Zahlenwerte haben wir Spielräume. Natürlich sollte *klein* auf 0, *sehr groß* auf 1 abgebildet werden. Wie wir die anderen Werte abbilden, ist uns überlassen. Durch die (willkürliche) Wahl der Zahlen wird jedoch der Abstand zwischen den Werten festgelegt und somit eventuell das Resultat beeinflusst.

Auch nominale Attribute können wir in metrische Attribute umwandeln. Binäre Attribute wie Geschlecht transformiert man in ein metrisches Attribut, indem man die Geschlechter ganz einfach durch 0 und 1 codiert. Nominale Attribute lassen sich aber durchaus immer in metrische Attribute überführen. Betrachten wir dazu das Attribut *Farbe*. Für jede Haarfarbe, die in unserem Datenbestand vorkommt, führen wir ein neues Attribut ein. Dort wird dann eine 1 eingetragen, falls die Haarfarbe exakt dem Attributnamen (also beispielsweise schwarz) entspricht, sonst 0. Dies funktioniert immer, hat aber den Nachteil, dass wir *ein* Attribut durch *mehrere* ersetzen, was die Größe unseres Datenbestands vergrößert. Diese Codierung wird Binärcodierung gennant. Sie wird häufig bei den neuronalen Netzen genutzt.

Können wir metrische Attribute in ordinale überführen? Dies ist möglich, indem man beispielsweise Intervalle bildet und diesen dann Namen gibt: *klein, groß, sehr groß*.

Die Umwandlungsmöglichkeiten sind im Data Mining relevant, da es Verfahren gibt, die entweder nur oder vorzugsweise mit einem bestimmten Datentyp arbeiten. So ist für das Verfahren *k-Nearest Neighbour* (vergleiche Abschnitt 5.1) ein kontinuierliches, metrisches Attribut geeignet. Ordinale und nominale Daten sind zwar nicht komplett ausgeschlossen, aber doch eher ungeeignet.

Welche Vergleichs- und welche Rechenoperatoren sind bei den verschiedenen Datentypen verfügbar?

Bei *nominalen* Attributen ist allein ein Vergleich auf *gleich* oder *ungleich* möglich. *Ordinale* Attribute erlauben darüber hinaus auch Vergleiche *kleiner* und *größer*. Bei *metrischen* Attributen kann man nun zusätzlich rechnen. Metrische Attribute erlauben zumindest Addition und Subtraktion, meist aber sogar alle vier Grundrechenarten. Bei den intervallbasierten Datentypen sind Multiplikation und Division meist nicht sinnvoll.

**Aufgabe 2.1:** *Datentypen*

Ordnen Sie den Attributen den jeweiligen Datentyp zu:

- Postleitzahlen
- ISB-Nummer
- Kraftstoffverbrauch
- Fahrzeugleistung
- Hausnummer

**Aufgabe 2.2:** *Datentypen*

Finden Sie für alle Datentypen mindestens zwei weitere Beispiele.

## 2.3 Abstands- und Ähnlichkeitsmaße

Viele Anwendungen im Data Mining beruhen darauf, dass Datensätze miteinander verglichen werden. Bei der Bildung von Clustern (siehe Abschnitte 3.1 und 6) ist das Ziel, ähnliche Objekte zu einer Gruppe zusammenzufassen. Das setzt aber voraus, dass wir die Ähnlichkeit von 2 Datensätzen quantifizieren können. Dies realisiert man meistens über sogenannte Abstandsmaße, also Maße, welche die „Unähnlichkeit“ der Datensätze quantifizieren. Hat man ein solches *Abstandsmaß*

$$\mathbf{dist}(v, w),$$

welches den Abstand zweier Datensätze $v$ und $w$ misst, so kann man die Ähnlichkeit zweier Datensätze

$$\mathbf{simil}(v, w)$$

in Abhängigkeit von einem Abstandsmaß dist definieren. Je größer die Distanz zwischen den Datensätzen, desto geringer ist die Ähnlichkeit.

$$\text{simil}(v, w) = f(\text{dist}(v, w))$$

Eine solche Abstandsfunktion (auch *Distanzfunktion* genannt) muss folgende Eigenschaften erfüllen:

1. $\text{dist}(x, y) \geq 0$
2. $\text{dist}(x, x) = 0$
3. $\text{dist}(x, y) = \text{dist}(y, x)$
4. $\text{dist}(x, y) \leq \text{dist}(x, z) + \text{dist}(z, y)$

Der Abstand eines Datensatzes $x$ zu sich selbst sollte natürlich 0 sein. Man kann diese Forderung verschärfen zu

$$\text{dist}(x, y) = 0 \text{ genau dann, wenn } x = y.$$

Ein Abstandsmaß muss auch *kommutativ* sein: Der Abstand zwischen $x$ und $y$ ist derselbe wie zwischen $y$ und $x$. Die dritte Bedingung (*Transitivität*) fordert, dass der direkte Abstand zwischen $x$ und $y$ kleiner (oder gleich) als der Abstand ist, der sich ergibt, wenn man über einen Zwischenpunkt $z$ geht.

Wir betrachten einige typische Distanzfunktionen.

**Hamming-Distanz** $\text{dist}_{\text{H}}(v, w) = \text{count}_i(v_i \neq w_i)$

**Euklidische Distanz** $\text{dist}_{\text{E}}(v, w) = \sqrt{\sum_i (v_i - w_i)^2}$

**Manhattan-Distanz** $\text{dist}_{\text{Man}}(v, w) = \sum_i |v_i - w_i|$

**Maximum-Distanz** $\text{dist}_{\text{Max}}(v, w) = \max_i(|v_i - w_i|)$

Die Hamming-Distanz zählt, an wie vielen Positionen sich die Datensätze unterscheiden. Der Abstand zwischen den Datensätzen *(schulze, 1978, bmw)* und *(mueller, 1979, bmw)* beträgt 2, da sich die Datensätze an 2 Stellen unterscheiden. Die Hamming-Distanz erkennt nicht, dass sich die Jahreszahl 1978 von 1979 nur minimal unterscheidet. Dies ist ein Nachteil der Hamming-Distanz. Von Vorteil ist aber, dass die Hamming-Distanz auf beliebige Datentypen anwendbar ist, sowohl auf nominale, ordinale als auch metrische Daten.

Die euklidische Distanz nutzen wir tagtäglich. Sie ist die räumliche Distanz zweier Orte, im verallgemeinerten oder mathematischen Sinne der Abstand zweier Punkte im n-dimensionalen Raum. Sie ist nur auf metrische Daten anwendbar.

Dies gilt auch für die Manhattan-Distanz. Sie hat ihren Namen von der Schachbrettartigen Anordnung der Straßen in Manhattan. In Abbildung 2.1 auf der nächsten Seite ist veranschaulicht, dass es egal ist, welchen Weg man wählt. Die Strecke ist immer die Summe der Schritte, die man jeweils in beiden Dimensionen gehen muss.

Die Maximum-Distanz betrachtet ebenso die unterschiedlichen Dimensionen separat und wählt den größten Abstand, der sich über alle Attribute ergibt. Sie wird auch als Tschebyscheff-Distanz bezeichnet.

Abbildung 2.2 auf der nächsten Seite zeigt an einem Beispiel die vier unterschiedlichen Distanzen, die man zwischen zwei Objekten bestimmen kann.

Es gibt eine Erweiterung der euklidischen Distanz, die *Minkowski-Distanz*:

$$\text{dist}^n_{\text{Minkowski}}(v, w) = \sqrt[n]{\sum_i |v_i - w_i|^n}$$

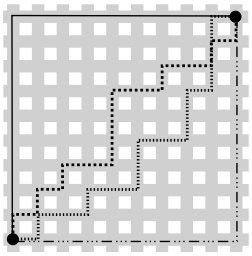
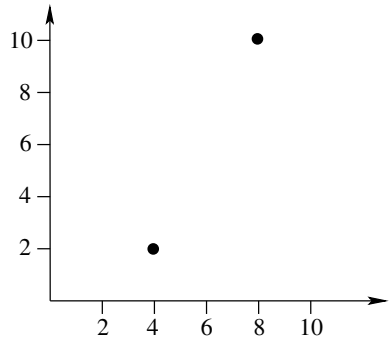


***Abb. 2.1:*** *Beispiel Manhattan-Distanz*




$\text{dist}_{\text{H}} = 2$, $\text{dist}_{\text{E}} \approx 8.9$, $\text{dist}_{\text{Man}} = 12$, $\text{dist}_{\text{Max}} = 8$

***Abb. 2.2:*** *Beispiel Distanzen*

Diese ist nur auf numerische Attribute anwendbar.

Für den mathematisch interessierten Leser ist sicher die Tatsache interessant, dass für $n \to \infty$ die Minkowski-Distanz zur Maximum-Distanz wird.

$$\lim_{n\to\infty} \text{dist}^n_{\text{Minkowski}} \;=\; \text{dist}_{\text{Max}}$$

Für $n = 1$ ergibt sich die Manhattan-Distanz:

$$\text{dist}^1_{\text{Minkowski}}(v, w) = \sqrt[1]{\sum_i |v_i - w_i|^1} = \sum_i |v_i - w_i|$$

Eine interessante Modifikation der euklidischen Distanz ist die *gewichtete euklidische Distanz*:

$$\text{dist}_{\text{EW}}(v, w) = \sqrt{\sum_i w_i \cdot (v_i - w_i)^2}$$

Auch dieses Abstandsmaß ist nur auf numerische Attribute anwendbar. Dieses Abstandsmaß ist deshalb interessant, da man die Attribute durch die Faktoren $w_i$ unterschiedlich gewichten kann. Je höher das Gewicht, desto größer ist der Einfluss auf den

Abstand. Man kann durch die gewichtete euklidische Distanz Attribute bevorzugen und diesen somit eine höhere Priorität geben.

In den Beispielen nutzen wir für numerische Attribute meistens die euklidische Distanz. Sie können aber auch mit anderen Abstandsmaßen experimentieren. Man muss bei numerischen Attributen nicht zwingend immer die euklidische Distanz nehmen.

Bisher haben wir die Ähnlichkeit auf die Distanz, also die Unähnlichkeit zurückgeführt. Je kleiner die Distanz, desto größer die Ähnlichkeit. In [Run10] werden einige Ähnlichkeitsmaße, die nicht auf Distanzmaßen beruhen, behandelt, zum Beispiel die Ähnlichkeitsmaße *Cosinus*, *Überlapp*, *Dice* und *Jaccard*.

Welche Eigenschaften sollte ein Ähnlichkeitsmaß **simil** erfüllen?

1. $\text{simil}(x, y) \geq 0$
2. $\text{simil}(x, y) = \text{simil}(y, x)$
3. $\text{simil}(x, y) \leq \text{simil}(x, x)$

Ein Ähnlichkeitsmaß heißt normiert, wenn gilt:

$$\text{simil}(x, x) = 1$$

Das Cosinus-Ähnlichkeitsmaß misst die Ähnlichkeit von Vektoren, es setzt somit numerische Attribute voraus. Sind zwei Vektoren identisch (beziehungsweise zeigen sie in die gleiche Richtung), so bilden sie einen Winkel von 0°. Der Cosinus von 0° ist 1. Stehen zwei Vektoren senkrecht aufeinander, ergibt sich 0. Zeigen die Vektoren in die entgegengesetzte Richtung, dann ergibt sich $-1$.

$$\cos(x, y) = \frac{x \cdot y}{\sqrt{\sum_i {x_i}^2 \cdot \sum_i {y_i}^2}} = \frac{\sum_i x_i \cdot y_i}{\sqrt{\sum_i {x_i}^2 \cdot \sum_i {y_i}^2}}$$

Ein Wert nahe 1 drückt die große Ähnlichkeit der Vektoren aus, ein Wert nahe $-1$ steht für eine große Unähnlichkeit.

Dieses Ähnlichkeitsmaß kann man bei den selbstorganisierenden Karten in Abschnitt 6.7 zur Bestimmung des Gewinner-Neurons benutzen.

**Aufgabe 2.3:** *Abstandsmaß und Schachbrett*

Wenn man die Schritte des Königs auf einem Schachbrett als Distanz wählt, welchem Distanzbegriff entspricht das? Und welchem Begriff entspricht die Anzahl der Felder, die der Turm passieren müsste?

**Aufgabe 2.4:** *Abstandsmaß*

Berechnen Sie die Distanz zwischen den Punkten $(0, 1, 2)$, $(1, 5, 3)$ und $(4, -2, 3)$. Verwenden Sie alle 4 aufgeführten Distanzfunktionen.

**Aufgabe 2.5:** *Hamming-Distanz*

Begründen Sie, wieso die Hamming-Distanz unempfindlich gegen Ausreißer ist.

**Aufgabe 2.6:** *Weitere Abstandsmaße*

Suchen Sie weitere Abstandsmaße. Für welche Datentypen sind diese geeignet? Was unterscheidet sie von den hier behandelten Abstandsmaßen?

**Aufgabe 2.7:** *Kombiniertes Abstandsmaß*

Wie würden Sie ein Abstandsmaß für Datensätze entwickeln, die heterogen sind, die also gemischte Datentypen – nominal, ordinal und metrisch – enthalten?

## 2.4 Grundlagen Künstlicher Neuronaler Netze

Ein *künstliches neuronales Netz* ist der Versuch, einen Wissensspeicher zu schaffen, der ähnlich dem leistungsfähigen menschlichen Gehirn funktioniert. Will man diesem biologischen Vorbild folgen, sind dessen Elemente und Vorgehensweisen in die Welt des Computers abzubilden.

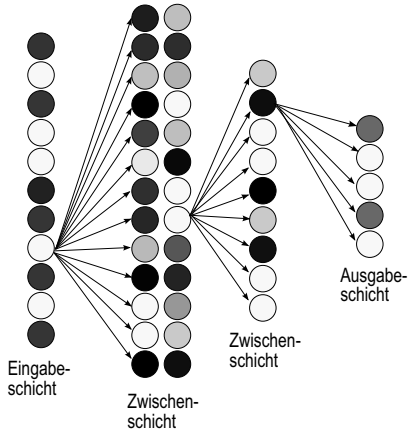



***Abb. 2.3:*** *Ein künstliches neuronales Netz*

**Definition 2.5:** *Künstliches Neuronales Netz*

Ein **künstliches neuronales Netz** besteht aus einer Menge von Neuronen, die durch gerichtete und gewichtete Verbindungen untereinander verknüpft sind (vergleiche [LC12]).

Die Abbildung 2.3 auf der vorherigen Seite zeigt ein vorwärtsgerichtetes neuronales Netz, bestehend aus einer Eingabe-Schicht, zwei verdeckten Schichten (auch Zwischenschichten genannt) sowie einer Ausgabe-Schicht. Eine Schicht besteht aus einer Menge einzelner künstlicher Neuronen.

**Definition 2.6:** *Künstliches Neuron*

Ein **künstliches Neuron** ist eine Einheit, die

- aus den, über gewichtete Verbindungen, eingehenden Werten
- unter Berücksichtigung eines Schwellwertes
- mittels einfacher mathematischer Funktionen
- einen Erregungswert berechnet und
- diesen an nachfolgende Neuronen weiterleitet.

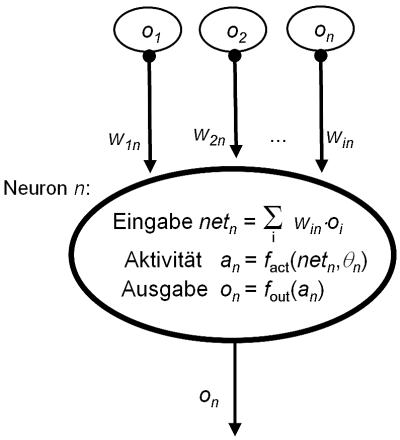



***Abb. 2.4:*** *Ein künstliches Neuron*

Die *Netzeingabe net* eines Neurons $n$ wird berechnet, indem die Ausgaben der Vorgänger-Neuronen $o_i$ mit den entsprechenden Werten $w_{in}$ an den Verbindungen gewichtet addiert werden. Eine *Aktivierungsfunktion act* benutzt diesen Wert $net_n$ und berechnet unter Berücksichtigung eines *Schwellwertes* $\theta_n$ (englisch threshold oder bias) die Aktivierung $act_n$ des Neurons. Eine einstellige Ausgabefunktion mit dem Wert der Aktivierung als Parameter bestimmt dann die Ausgabe $o_n$ des Neurons. Abbildung 2.4 illustriert sowohl den Aufbau eines künstlichen Neurons als auch die in einem künstlichen Neuron stattfindenden Berechnungen.

Betrachten wir die im Neuron eingesetzten Funktionen genauer.

## Propagierungsfunktion

Die *Propagierungsfunktion* ermittelt die Netzeingabe eines Neurons: Dazu werden die Ausgaben der Vorgängerneuronen mit den entsprechenden Gewichten an den Verbindungen multipliziert und aufsummiert:

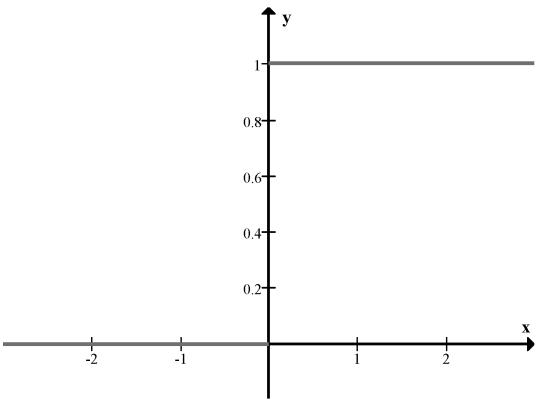
$$net_n = \sum_i w_{in} \cdot o_i$$

## Aktivierungsfunktion

Als *Aktivierungsfunktion* können unterschiedliche Funktionen mit einem sigmoiden Verhalten eingesetzt werden. Von einem sigmoiden Verhalten wird gesprochen, wenn der Funktionsgraph in der Umgebung eines Punktes stark ansteigt und davor sowie danach keinen oder nur einen sehr schwachen Anstieg aufweist. Sigmoide Funktionen sind somit eine Art Schalter. In einem Neuron schalten sie die Aktivierung des Neurons an oder aus. Sigmoide Funktionen, die in Neuronen zur Berechnung der Aktivierung eingesetzt werden, sind:

- *Schwellwertfunktion*,
- *logistische Funktion*,
- *Tangens Hyperbolicus*.

Die *Schwellwertfunktion* ist 0 für alle Werte kleiner oder gleich dem Schwellwert $x \leq \theta$, und für alle Werte $x > \theta$ wird das Neuron mit dem Wert 1 aktiviert (Abbildung 2.5). Die Schwellwertfunktion lässt sich für das Delta-Lernverfahren für vorwärtsgerichtete neuronale Netze ohne eine innere Schicht einsetzen. Vorwärtsgerichtete neuronale Netze ohne Zwischenschicht werden als Perzeptron bezeichnet. Eine Schwellwertfunktion ist

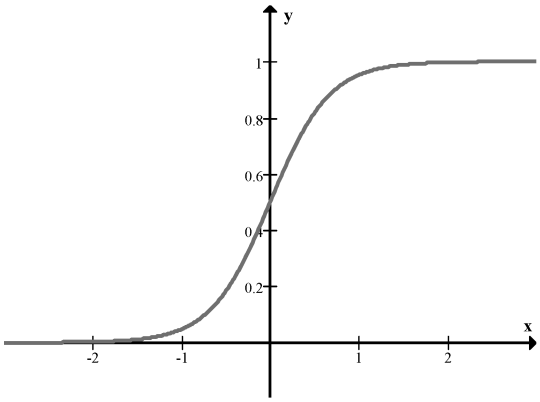



$$f(x) = \begin{cases} 1, & x > \theta \\ 0, & x \leq \theta \end{cases}$$

***Abb. 2.5:*** *Schwellwertfunktion*

nicht stetig differenzierbar und kann deshalb nicht für Lernverfahren eingesetzt werden, die auf einem Gradientenabstiegsverfahren basieren.

Die *Logistische Funktion* (Abbildung 2.6) ist stetig differenzierbar und die am häufigsten eingesetzte Aktivierungsfunktion in vorwärtsgerichteten neuronalen Netzen.

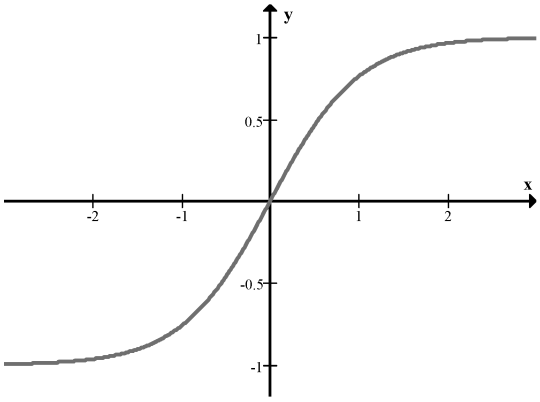



$$f(x) = \frac{1}{1 + e^{-px}}$$

***Abb. 2.6:*** *Logistische Funktion*

Die logistische Funktion ist einfach differenzierbar und kann in Lernverfahren verwendet werden, die auf einem Gradientenabstiegsverfahren basieren. Mit dem Parameter $p$ kann die Stärke des Anstiegs der Kurve im Bereich des Schaltpunktes, in einem Neuron ist dies der Schwellwert, gesteuert werden.

Der *Tangens Hyperbolicus* wird verwendet, wenn für die Aktivierung des Neurons der Bereich $[-1, +1]$ in Frage kommt. Der *Tangens Hyperbolicus* ist ebenso wie die logistische Funktion einfach zu differenzieren und wird in den Lernverfahren mehrschichtiger vorwärtsgerichteter neuronaler Netze insbesondere dann eingesetzt, wenn Werte aus dem Intervall $[-1, +1]$ zu verarbeiten sind, siehe Abbildung 2.7.




$$f(x) = \tanh(x)$$

***Abb. 2.7:*** *Die Funktion Tangens Hyperbolicus*

## Ausgabefunktion

Als *Ausgabefunktion* wird in vielen Fällen die Identität $f(x) = x$ eingesetzt. Damit ist die Aktivität eines Neurons gleichzeitig sein Ausgabewert $o_n = act_n$. In Data-Mining-Anwendungen kann eine Schwellwertfunktion in der Ausgabeschicht zur Anwendung kommen, die dann eine binäre Ausgabe erzeugt, 0 oder 1. Anstatt die übliche Rundungsfunktion hierfür einzusetzen, kann ein selbst festgelegter Schwellwert (oft deutlich kleiner als 0,5) das Ergebnis entscheidend verbessern.

## Beispiel-Neuron

Betrachten wir ein simples „Netz“, welches nur aus einem arbeitenden Neuron besteht, welches die logische UND-Funktion simulieren kann (siehe Abbildung 2.8): Die Eingabe-Schicht besteht aus zwei Neuronen, die die Netz-Eingabe darstellen. Dabei werden die Aktivierungen von außen gesetzt. Diese Aktivierungen entsprechen den Werten der Eingabe-Muster oder Eingabe-Daten. Sind beide Eingaben 1 (also wahr), so soll die Netzausgabe ebenfalls 1 sein, sonst 0. Dieses Verhalten muss durch das einzelne Neuron der zweiten Schicht erreicht werden. Das Neuron ist somit ein Ausgabe-Neuron.

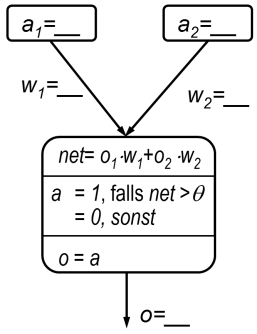



***Abb. 2.8:*** *Beispiel: Ein Neuron fungiert als UND-Schalter*

Im üblichen Einsatzfall neuronaler Netze werden die Gewichte an den Verbindungen sowie die zugehörigen Gewichte durch ein Training gelernt. In diesem Beispiel werden wir diese manuell bestimmen. Die Frage lautet somit:

Welche Werte können wir für $w_1$, $w_2$ sowie $\theta$ einsetzen, damit das Neuron das gewünschte Verhalten zeigt? Wir gehen davon aus, dass die Ausgabefunktion die identische Funktion, somit $o_3 = act_3$, ist.

Richtig, es gibt unendlich viele Möglichkeiten. Eine ist: $w_1 = w_2 = 1$ und $\theta = 1,2$.

Die in diesem Abschnitt vorgestellten Abläufe und Berechnungen in einem Neuron werden in allen künstlichen neuronalen Netzen zumindest in ähnlicher Form eingesetzt.

## 2.5 Logik

Wir können an dieser Stelle keine Einführung in die Logik geben. Da aber sehr viele formale Formen der Wissensdarstellung auf *Logik* aufbauen und Data Mining den Anspruch erhebt, Wissen aus Massendaten zu extrahieren, muss ein Grundverständnis der Logik vorhanden sein. Dazu definieren wir hier einige Grundbegriffe der mathematischen Logik.

Die Logik – oder konkreter *logische Formeln* – werden für die Darstellung von Wissen eingesetzt. Was aber ist eine logische Formel?

Wir beantworten diese Frage im Rahmen des sogenannten Prädikatenkalküls 1. Stufe (PK1), welcher ausgangs des 19. Jahrhunderts vom deutschen Mathematiker und Philosophen Gottlob Frege in seiner Begriffsschrift vorgestellt wurde [Fre79]. Eine ausführliche Einführung in diese formale Logik findet man unter anderem in [LC12].

Formal ist eine *logische Formel* nach diesen Regeln aufgebaut:

**Definition 2.7:** *Logische Formel*

1. Ist $p$ ein $n$-stelliges Prädikatensymbol und sind $t_1, \ldots, t_n$ Terme, dann ist auch $p(t_1, \ldots, t_n)$ eine (*atomare*) logische Formel.
2. Sind $A$ und $B$ logische Formeln, dann sind auch folgende Verknüpfungen logische Formeln:
   - *Konjunktion*: $A$ und $B$ $(A \wedge B)$,
   - *Disjunktion*: $A$ oder $B$ $(A \vee B)$,
   - *Negation*: nicht $A$ $(\neg A)$,
   - *Implikation*: Wenn $A$ Dann $B$ $(A \rightarrow B)$ sowie
   - *Äquivalenz*: $A$ genau dann, wenn $B$ $(A \leftrightarrow B)$.
3. Ist $x$ eine Variable und $F$ eine logische Formel, so sind auch folgende sogenannte *Quantifizierungen* logische Formeln:
   - *Allquantifizierung*: Für alle $x$ gilt $F$ $(\forall x\ F)$,
   - *Existenzquantifizierung*: Es existiert ein $x$, für das $F$ gilt $(\exists x\ F)$.
4. Es gibt keine anderen logischen Formeln.

In der Bildungsvorschrift für logische Formeln wird der Begriff eines *Terms* verwendet, der nun definiert wird:

**Definition 2.8:** *Term*

1. Jede *Konstante* ist ein Term.
2. Jede *Variable* ist ein Term.
3. Ist $f$ ein $n$-stelliges Funktionssymbol und sind $t_1, \ldots, t_n$ Terme, dann ist auch $f(t_1, \ldots, t_n)$ ein Term.
4. Es gibt keine anderen Terme.

Die Wirkung der logischen Verknüpfungen wird in Form einer Wahrheitswerte-Tabelle gezeigt:

| A | B | A ∨ B | A∧B | ¬A | A→B | A↔B |
|---|---|---|---|---|---|---|
| W | W | W | W | F | W | W |
| W | F | W | F | F | F | F |
| F | W | W | F | W | W | F |
| F | F | F | F | W | W | W |

***Tabelle 2.1:*** *Tabelle der Wahrheitswerte der logischen Verknüpfungen*

Überlegen Sie sich anhand der Wahrheitswerte, dass die Implikation $A \rightarrow B$ zu folgender Disjunktion äquivalent ist: $\neg A \vee B$.

Wenn $A$ falsch ist, ist die Implikation $A \rightarrow B$ (WENN $A$ DANN $B$) stets richtig. Oder es gilt $B$, das heißt, wenn $A$ nicht falsch ist, also gilt, dann *muss* auch $B$ gelten.

Mittels der Prädikatensymbole können Eigenschaften von Objekten oder auch Beziehungen zwischen Objekten formal durch logische Formeln ausgedrückt werden. Hierzu ein paar Beispiele:

```
weiblich(anna).
sterblich(sokrates).
gruen(haus1).
guterKunde(meier).
kind(anna,paul,beate).
klasse(meier,guterKunde).
entfernung(a,b,100).
```

Nicht alle logischen Formeln können wir wirklich als Wissen betrachten. Eine logische Formel wie guterKunde($x$) sagt wenig aus:
Gibt es ein $x$, welches ein guter Kunde ist?
Gibt es kein $x$?
Sind alle $x$ gute Kunden?
Wird nun das $x$ quantifiziert und sind generell alle Variablen in einer logischen Formel quantifiziert, so können wir dieser Formel einen Wahrheitswert zuordnen:

$$\forall x \text{ guterKunde}(x) \rightarrow \text{rabatt}(x)$$

Für die Darstellung von Wissen sind diese speziellen logischen Formeln von Interesse: die *Aussagen*. Aussagen sind logische Formeln, denen ein Wahrheitswert, *wahr* oder *falsch*, zugeordnet werden kann. Gemäß unserer Definition von Wissen als Information verbunden mit der Fähigkeit diese anzuwenden, ist Wissen immer richtig. Wird Wissen mittels logischer Formeln dargestellt, so sind dies damit offensichtlich nur solche Formeln, denen wir den Wahrheitswert *wahr* zuordnen. Die Aussage „Wenn x ein guter Kunde ist, dann wird dem Kunden x Rabatt gewährt." ist eine solche logische Formel, die als Handlungsanleitung zu verstehen ist und somit als wahr anzusehen ist.

Sehr häufig werden in diesem Buch Wenn-Dann-Sätze verwendet, siehe Abschnitt 3.2 oder Abschnitt 4. Der folgende Satz entstammt Abbildung 3.2 auf Seite 60:

*WENN Alter = 31 ... 40 UND Einkommen = hoch DANN Kreditwürdigkeit = sehr gut*

Diese Sätze werden als Regeln bezeichnet und sind im Sinne der Prädikatenlogik spezielle logische Formeln:

**Definition 2.9:** *Regel*

Eine *Regel* ist eine logische Formel mit folgenden Eigenschaften:

1. Die Struktur ist:
   $\forall x_1, x_2 \ldots x_n \; (A_1 \wedge A_2 \wedge \ldots \wedge A_k) \rightarrow (B_1 \wedge B_2 \wedge \ldots \wedge B_l)$
2. Es treten keine weiteren Variablen als $x_1, x_2 \ldots x_n$ in den $A_i$ oder $B_j$ auf. Somit sind alle Variablen in der Formel allquantifiziert.
3. Die $A_i$ und $B_j$ sind atomare Formeln oder negierte atomare Formeln.

Die oben angeführte Wenn-Dann-Aussage entspricht den Forderungen an eine Regel und kann wie folgt als logische Formel notiert werden:

$$\forall x \; (\text{alter}(x) \geq 31 \wedge \text{alter}(x) \leq 40 \wedge \text{einkommen}(x, \text{hoch})) \rightarrow \text{kreditwürdigkeit}(x, \text{sehrGut})$$

Da eine Regel eine logische Formel ist, in der alle Variablen allquantifiziert sind, wird in der Praxis auf die Darstellung der Quantifizierung verzichtet. Zudem wird aus Gründen der Lesbarkeit statt des Implikationszeichens ($\rightarrow$) die WENN-DANN-Schreibweise verwendet:

$$\textit{WENN}\ \text{alter}(x) \geq 31\ \textit{UND}\ \text{alter}(x) \leq 40\ \textit{UND}\ \text{einkommen}(x, \text{hoch})\ \textit{DANN}\ \text{kreditwürdigkeit}(x, \text{sehrGut})$$

Eine Menge von Regeln, die eine Klassifikation oder eine Prognose beschreiben, ist ein Ziel des Data Mining. Gelingt die Erarbeitung einer solchen Regelmenge, dann ist das Ziel der Wissensextraktion aus Massendaten erreicht: Das Wissen liegt nun in Regelform vor. Man kann sagen, das Wissen ist als eine Menge logischer Formeln dargestellt. Die Logik ist somit die Grundlage dieser Wissensdarstellung.

Regeln können – wie hier beschrieben – als logische Formeln dargestellt werden, Regeln können aber auch in Form einer Tabelle angegeben werden (Abschnitt 4.1) oder sind

in einem Entscheidungsbaum enthalten. In einer Entscheidungstabelle entspricht eine Spalte einer Regel; in einem Entscheidungsbaum stellt jeder Pfad vom Wurzelknoten zu einem Blatt eine Regel dar.

## 2.6 Überwachtes und unüberwachtes Lernen

Data Mining beginnt mit einer gegebenen Menge von Beispielen. Diese nennt man *Instanzenmenge* oder *Beispielmenge* $E$. Auf einer Teilmenge von $E$ „lernt“ man. Man benutzt $E$, um beispielsweise Klassen zu bilden oder Assoziationen herzustellen. Dies ist die sogenannte *Trainingsmenge* $T \subset E$. Eine weitere Teilmenge von $E$ (meist $E \setminus T$) benutzt man, um das Gelernte zu prüfen, zu *validieren*. Diese Menge heißt folglich *Testmenge* (oder Validierungsmenge) $V \subset E$. Dieses Vorgehen ist ausführlich im Abschnitt 9.4 erläutert.

Für das Training können die zwei folgenden *Lernstrategien* unterschieden werden:

**Unüberwachtes Lernen**
: Die zu entdeckenden Muster sind gänzlich unbekannt. Auch Beispiele, die eine Gruppierung oder Klassifikation vorgeben, sind nicht gegeben. Ein Beispiel für nicht-überwachtes Lernen ist die Cluster-Analyse (Abschnitte 3.1 und 6). Die Aufgabe ist, Gruppen von ähnlichen Objekten zu finden und diese zu Untermengen, also Clustern zusammenzufassen. Hier ist keine Lösung vorgegeben, wir können also die durch einen Algorithmus entwickelte Lösung nicht mit einer gewünschten Clustermenge vergleichen.

**Überwachtes Lernen**
: Es werden Beispiele vorgegeben, in denen das Resultat gegeben ist. Haben wir zum Beispiel Datensätze für Nadel- oder Laubbäume gegeben, die zum einen Merkmale der Bäume enthalten, zum anderen aber auch die Klassifizierung in Nadel- oder Laubbaum, dann wissen wir also, welcher Baum in welche Kategorie gehört. Nun können wir prüfen, ob unsere Verfahren wirklich eine korrekte Klassenzuordnung liefern (vgl. Klassifikation, Abschnitte 3.2 und 5).

Gerade beim überwachten Lernen fällt auf, dass wir eine Grundannahme stillschweigend voraussetzen: Die gegebenen Beispieldaten sind repräsentativ, das bedeutet:

*Zukünftige Daten verhalten sich ähnlich wie die gegebenen Beispiele.*

Auf dieser Annahme basiert jedes Verfahren, welches aus der Vergangenheit auf zukünftige Ereignisse schließen möchte. Diese Annahme muss nicht immer gerechtfertigt sein, aber wenn wir diese Annahme nicht als gültig annehmen, sollten wir mit Data Mining erst gar nicht beginnen.