Einführung in die KI

– Teil 2 –

Technische Hochschule Deggendorf

Wintersemester 2020/2021

Prof. Dr. Robert Hable

Inhaltsverzeichnis

1 Einführung in Teil 2: Maschinelles Lernen **3**
1.1 Data Mining Projekte in Unternehmen 3
1.2 Was ist Maschinelles Lernen? 3
1.3 Datenanalyse-Software R 3
1.3.1 Basis-Infos zu R 3
1.3.2 Installation von R 4
1.4 Einlesen von Daten in R 5

2 Supervised Learning (I): parameterbasierte Verfahren **10**
2.1 Einführung 10
2.2 Regression 14
2.2.1 Einfache lineare Regression 14
2.2.2 Multiple lineare Regression 15
2.3 (Binäre) Klassifikation: Logistische Regression 18

3 Supervised Learning (II): Nichtparametrische Verfahren **20**
3.1 Einführung 20
3.2 Trainings- und Testdatensatz 32
3.3 Entscheidungsbäume 34
3.4 Neuronale Netze 44

4 Bayes Netze **52**
4.1 Bedingte Wahrscheinlichkeiten 52
4.2 Repräsentation von Unsicherheit in Bayes Netzen 56
4.3 Multiplikationssatz und Inferenz im Bayes Netz 59

1 Einführung in Teil 2: Maschinelles Lernen

1.1 Data Mining Projekte in Unternehmen

Vgl. Präsentationsfolien

Data-Mining_Projekte_Unternehmen.pdf

im iLearn.

1.2 Was ist Maschinelles Lernen?

Vgl. Präsentationsfolien

Vortrag_Was-ist-maschinelles-Lernen.pdf

im iLearn.

1.3 Datenanalyse-Software R

1.3.1 Basis-Infos zu R

Datenanalyse ist kein Zuschauersport und die TH Deggendorf ist eine Hochschule für angewandte Wissenschaften. Ziel dieser Vorlesung ist daher, dass Sie hinterher erste maschinelle Lernverfahren selber anwenden können - und das geht nur mit dem Computer und der entsprechenden Software.

Als Software verwenden wir R. Dies ist

- eine freie Open Source Software
- bzw. eigentlich eine Programmiersprache mit einem Interpreter als Kommandozeilenumgebung, d.h. R ist kein Programm, in dem man rumklicken kann, sondern man muss Programmier-Code schreiben
- der Standard in Wirtschaft und Wissenschaft (zusammen mit Python)
- zusammen mit Python die mit Abstand leistungsfähigste Datenanalyse-Software, weil Experten an Universitäten in der ganzen Welt ständig an Erweiterungen und Verbesserungen arbeiten

Mit der Installation von R bekommt man das bereits extrem umfangreiche Basis-Programm. Jeder kann Erweiterungen (sogenannte Pakete) schreiben, und diese Erweiterungen kann man jederzeit (kostenlos) dazuinstallieren.

1.3.2 Installation von R

Um die Software R auf Ihrem Computer zu installieren, gehen Sie auf folgende Internet-Seite:

`http://cran.at.r-project.org`

Dort sehen Sie Folgendes:

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Für Windows klicken Sie auf “Download R for Windows”.

Dann sehen Sie Folgendes:

R for Windows

Subdirectories:

base	Binaries for base distribution (managed by Duncan Murdoch). This is what you want to **install R for the first time**.
contrib	Binaries of contributed packages (managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
Rtools	Tools to build R and R packages (managed by Duncan Murdoch). This is what you want to build your own packages on Windows, or to build R itself.

Klicken Sie auf “base”.

Nun sehen Sie Folgendes:

R-3.1.2 for Windows (32/64 bit)

Download R 3.1.2 for Windows (54 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

Klicken Sie auf “Download R 3.1.2 for Windows”, dann wird eine exe-Datei heruntergeladen. Speichern Sie diese exe-Datei auf Ihrem Computer; durch Doppelklick auf die Datei wird dann das Programm wie gewöhnlich auf Ihrem Computer installiert.

Außerdem benötigen Sie zusätzlich noch einen geeigneten Texteditor. Im Prinzip kann man jeden beliebigen Texteditor verwenden, aber es ist viel besser spezielle, auf R abgestimmte Texteditoren, mit R-Syntax-Highlighting zu verwenden – genaueres hierzu später. Installieren Sie sich daher zum Beispiel den Text-Editor “Tinn-R” auf Ihrem Computer. Tinn-R ist ein freies GNU Programm, das Sie kostenlos im Internet herunterladen können, zum Beispiel hier

`https://sourceforge.net/projects/tinn-r/`

Alternativ können Sie auch das Programm “RStudio” verwenden, das Sie ebenfalls kostenlos aus dem Internet herunterladen können.

1.4 Einlesen von Daten in R

Wenn Sie erfolgreich mit R arbeiten wollen, beachten Sie bitte unbedingt folgende Hinweise:

- Tippen Sie die Kommandobefehle keinesfalls direkt in R ein. Schreiben Sie stattdessen Ihren Programmiercode (mit dem Programm Tinn-R) in eine .r-Datei und lassen Sie den Programmiercode anschließend in R ausführen.

- Verwenden Sie beim Schreiben Ihres Programmiercodes die Vorlagedateien (.r-Dateien) und arbeiten Sie mit Copy-und-Paste! Tippen Sie den Programmiercode nicht direkt in R ein, sondern verwenden Sie Tinn-R!
- Beim Einlesen der Daten müssen Sie sich **exakt** an untenstehende Schritt-für-Schritt-Anleitung halten. Sonst funktioniert es nicht richtig. Beachten Sie hierbei:
  - Der Computer verzeiht beim Programmieren keinen noch so kleinen Fehler! Es reicht also nicht, die Schritte nur so ungefähr und unvollständig auszuführen.
  - Beachten Sie: Ein Punkt ist etwas anderes als ein Komma!
  - Tippen Sie die Kommandobefehle keinesfalls direkt in R ein. Schreiben Sie stattdessen Ihren Programmiercode (mit dem Programm Tinn-R) in eine .r-Datei und lassen Sie den Programmiercode anschließend in R ausführen.

Führen Sie zum Einlesen der Daten exakt folgende Schritte vollständig durch:

0. **Speichern der Daten in einer csv-Datei**

   Falls die Daten noch nicht in einer csv-Datei vorliegen, dann müssen Sie Ihre Daten zunächst als csv-Datei abspeichern. Wenn Sie zum Beispiel Ihre Daten in einer Excel-Datei haben, dann öffnen Sie die Datei in Excel, gehen auf "Speichern unter" und wählen beim Abspeichern der Datei das csv-Format (unter "Dateityp").
   Achten Sie darauf, dass die Spalten Ihres Datensatzes Überschriften haben.

1. **Kontrolle der csv-Datei**

   Dieser zentrale Punkt wird oft vergessen und ist fast immer dafür verantwortlich, wenn das Einlesen der Daten misslingt. Öffnen Sie die csv-Datei in einem einfachen Text-Editor (z.B. das Windows-Programm "Editor" unter "Zubehör") und überprüfen Sie in der geöffneten Datei:

   (a) Mit welchem Symbol werden in der csv-Datei Spalten getrennt? (Typischerweiße ist das ein Strichpunkt/Semikolon oder ein Komma oder ein Leerzeichen)

   (b) Wie werden in der csv-Datei Dezimalzahlen geschrieben? (Englische Notation mit Punkt oder deutsche Notation mit Komma)

2. **Einlesen der Daten in R**

   Zum Einlesen der Daten müssen Sie zwei Zeilen Programmiercode schreiben. Tippen Sie diese Programmierzeilen unter keinen Umständen direkt in R ein, sondern schreiben Sie in Tinn-R eine .r-Datei mit Ihrem Programmiercode. Verwenden Sie die Vorlagedatei

   R-Skript_1.r

   und speichern Sie Ihre .r-Datei für später ab.

   Nun zu den zwei Zeilen Programmiercode:

   (a) Zunächst müssen Sie angeben, in welchem Ordner Ihre csv-Datei mit den Daten liegt. Hierzu setzen Sie mit dem Befehl `setwd` den Pfad für den Ordner. Unter Windows müssen Sie dabei umbedingt folgendes beachten: Windows verwendet bei Pfadangaben den Backslash "\", in R müssen Sie den Backslash aber jeweils durch einen Forward Slash "/" ersetzen. Im Pfadnamen dürfen auch Leerzeichen enthalten sein, aber Umlaute und Sonderzeichen können Probleme machen - benennen Sie also die Ordner im Zweifel lieber um!

   Bei mir lautet der Pfad in Windows z.B.:

   C:\Users\rHable\Documents\RHable\Daten

   Entsprechend muss ich den Pfad in R folgendermaßen setzen:

   ```
   setwd("C:/Users/rhable/Documents/RHable/Daten")
   ```

   Vergessen Sie dabei nicht die Anführungszeichen!

   (b) Für die zweite Programmiercode-Zeile kommt es nun darauf an: Stehen in der csv-Datei die Zahlen in der englischen Dezimalschreibweise (mit Punkt) oder in der deutschen Dezimalschreibweise (mit Komma)?

   - englische Dezimalschreibweise: Befehl `read.csv`
   - deutsche Dezimalschreibweise: Befehl `read.csv2`

   Um die Datei Koerpergewicht.csv einzulesen, lautet die Programmierzeile dann z.B.

   ```
   Daten <- read.csv("Koerpergewicht.csv",header=TRUE,sep=";",fill=TRUE)
   ```

Dadurch wird bei der Ausführung in R ein Objekt mit dem Namen `Daten` angelegt, das die Daten aus dem Datensatz enthält. Den Namen `Daten` kann ich frei wählen; statt `Daten` könnte ich das Objekt zum Beispiel auch `Osterhase` nennen.

Zum Befehl `read.csv` bzw. `read.csv2`:

- Innerhalb des Befehls `read.csv` bzw. `read.csv2` steht am Anfang der Dateiname in Anführungszeichen.
- Mit der Angabe `header=TRUE` sage ich, dass die erste Zeile die Spaltenüberschriften sind. Falls die Daten noch keine Spaltenüberschriften haben, müsste man stattdessen `header=FALSE` schreiben.
- In einer csv-Datei werden Spalten durch ein Zeichen getrennt, das ist üblicherweise ein Komma, ein Strichpunkt (Semikolon) oder ein Leerzeichen. Welches Zeichen das bei Ihnen ist, sehen Sie ganz einfach, indem Sie die csv-Datei mit einem einfachen Texteditor (nicht mit Excel!) öffnen. Mit der Angabe `sep=";"` sage ich, dass das in meiner Datei ein Strichpunkt (Semikolon) ist. Alternative Angaben wären also z.B. `sep=","` oder `sep=""`

3. **Kontrolle der Daten in R**

   Zum Schluss sollten Sie noch kontrollieren, ob die Daten tatsächlich richtig eingelesen wurden. Lassen Sie sich dazu zum Beispiel die ersten 15 Zeilen des Datensatzes in R ausgeben. Ist der Datensatz unter den Namen `Daten` gespeichert geht das durch Eingabe von

   ```
   Daten[1:15,]
   ```

   Lassen Sie sich auch eine Zusammenfassung des Datensatzes ausgeben:

   ```
   summary(Daten).
   ```

Weitere Einzelheiten befinden sich in der R-Datei

R-Skript_1.r

Übung: Schreiben Sie (in Tinn-R) eine .r-Datei mit dem Programmiercode für folgende Aufgabe: Die Daten aus der Datei

FreierFall.xlsx

werden in R eingelesen, die ersten 12 Datenpunkte und die summary werden ausgegeben. Die Daten werden außerdem als Punkte in ein Diagramm eingezeichnet und es wird der Durchschnittswert sowie die Standardabweichung für die Variable "Position" berechnet.

2 Supervised Learning (I): parameterbasierte Verfahren

2.1 Einführung

Ziel: Schätzen eines Zusammenhangs zwischen Einflussvariablen und einer Zielvariable

$$\text{Einflussvariablen} \quad \longrightarrow \quad \text{Zielvariable}$$

Einflussvariablen:

- eine oder mehrere Einflussvariablen möglich
- Einflussvariablen können sowohl nominal als auch metrisch sein

Zielvariable:

- immer nur eine Zielvariable möglich
- Zielvariable kann nominal oder metrisch sein:
  - nominal: Klassifikation
  - metrisch: Regression

Wichtige Datentypen:

- metrisch: (gemessene) Zahlenwerte (meist mit einem physikalischen Hintergrund), der Wert der Abstände zwischen den Zahlen hat eine inhaltliche Bedeutung

  Beispiele: Größe, Gewicht, Zeit, Volumen, Anzahl, ...
- nominal: (ungeordnete) Kategorien ausgedrückt durch Buchstaben, Wörter oder Codierungszahlen

  Beispiele: Geschlecht, Blutgruppe, Geburtsland, Postleitzahl, ...

Übung: Sehen Sie sich die Daten aus der Datei `Koerpergewicht_2.xslx` an. Beschreiben Sie mögliche sinnvolle Regressionen/Klassifikationen.

Übung: Sehen Sie sich die Daten aus der Datei `Patientendaten.xslx` an. Beschreiben Sie mögliche sinnvolle Regressionen/Klassifikationen.

Beispiel für Klassifikation: Geschlecht in Abhängigkeit von Größe und Gewicht

Beispiel für Regression: Bremsweg in Abhängigkeit von der Geschwindigkeit
Einflussvariable:
Zielvariable:

Streudiagramm:

Der tendenzielle Zusammenhang zwischen Geschwindigkeit und Bremsweg wird durch die Regressionsgerade beschrieben (die rote Linie im Bild):

Die Geschwindigkeit ist hier in Meilen pro Stunde (mph) gemessen und der Bremsweg ist in Fuß (ft) gemessen. Die Steigung der Regressionsgerade (rote

Linie) in dem Bild beträgt 3,9 ft/mph. Das heißt: Wenn man die Geschwindigkeit um eine Meile pro Stunde erhöht, dann verlängert sich im Durchschnitt der Bremsweg um 3,9 Fuß. Umgerechnet auf das internationale Einheitensystem bedeutet das: Wenn man die Geschwindigkeit um einen Kilometer pro Stunde erhöht, dann verlängert sich im Durchschnitt der Bremsweg um 74,5 cm.
Hinweis: Hierbei handelt es sich um historische Daten aus den 1920er Jahren. Die Geschwindigkeiten sind hier entsprechend niedrig. Bei höheren Geschwindigkeiten ist der Bremsweg wesentlich länger. Bei einer Geschwindigkeit von um die 100 km/h verlängert sich der Bremsweg um etwa 2 Meter, wenn man die Geschwindigkeit um 1 km/h erhöht.

2.2 Regression

Wir behandeln zunächst die Regression und erst danach die Klassifikation. Die Regression ist leichter verständlich und anschaulicher als die Klassifikation.

2.2.1 Einfache lineare Regression

Der anschaulichste Spezialfall einer Regression ist die sogenannte einfache lineare Regression. Hierbei ist die Zielvariable metrisch und es gibt nur eine Einflussvariable, die ebenfalls metrisch ist.

Beispiel: Bremswegdaten
Wir haben hier 50 Datenpunkte $(x_1, y_1),\ (x_2, y_2),\ \ldots,\ (x_{50}, y_{50})$, wobei
x_i = Geschwindigkeit (in mph) bei Fahrt Nr. i
y_i = Bremsweg (in ft) bei Fahrt Nr. i

Das Regressionsmodell lautet hier

$$\text{Bremsweg} = \beta_0 \ + \ \beta_1 \cdot \text{Geschwindigkeit} \ + \ \text{Zufallsabweichung}$$

bzw. in etwas genauerer Formelschreibweise

$$y_i \ = \ \beta_0 \ + \ \beta_1 \cdot x_i \ + \ \varepsilon_i,$$

wobei ε_i die Zufallsabweichung von der Regressionsgerade bei der i-ten Fahrt bezeichnet. Wäre die Zufallsabweichung stets gleich 0 (also $\varepsilon_i = 0$), dann

würden alle alle Punkte genau auf einer Linie liegen (nämlich auf der Regressionsgeraden).

Die beiden Parameter β_0 und β_1 beschreiben die Regressionsgerade; dabei ist β_0 der Intercept (Achsenabschnitt) und β_1 die Steigung der Regressionsgeraden. Die Parameter β_0 und β_1 werden aus den Daten mit der Methode der kleinsten Quadrate berechnet:

$$\text{Finde } \beta_0,\ \beta_1,\ \text{so dass } \sum_{i=1}^{n} \big(y_i - (\beta_0 + \beta_1 x_i)\big)^2 \text{ minimal ist.}$$

Die Regressionsgerade ist also die Gerade, die am besen zu den Daten passt. Dabei heißt "am besten", dass sie die Summe der quadrierten Abstände zwischen Model $\beta_0 + \beta_1 x_i$ und Realität y_i minimiert.

2.2.2 Multiple lineare Regression

Nun der allgemeinere Fall: Wir haben eine metrische Zielvariable und mehrere (metrische oder nominale) Einflussvariablen.

Das Ergebnis einer linearen Regression sind Schätzungen für Koeffizienten. Diese Koeffizienten beschreiben die Größe des Einflusses[1] der Einflussvariablen.

Welche Koeffizienten werden in der Regression geschätzt und was bedeuten die jeweiligen Koeffizienten:

- metrische Einflussvariable: 1 Koeffizient

  Für jede metrische Einflussvariable wird ein Regressionskoeffizient geschätzt. Dieser Koeffizient beschreibt den Zusammenhang zwischen der jeweiligen Einflussvariable und der Zielvariable. Der Koeffizient besagt, um wieviele Einheiten sich die Zielvariable (im Mittel) erhöht, wenn sich die Einflussvariable um eine Einheit erhöht.

- kategorielle Einflussvariable mit k Kategorien: $k-1$ Koeffizienten

  Für eine kategorielle Einflussvariable mit k Kategorien werden $k-1$ Koeffizienten geschätzt. Bei einer kategoriellen Einflussvariablen ist eine

[1] Eigentlich ist das Wort "Einfluss" hier nicht richtig, denn es beinhaltet einen kausalen Zusammenhang, der allein durch die Berechnung einer Regression nicht nachgewiesen werden kann. Statt "Einfluss" muss man eigentlich "Zusammenhang" sagen.

Kategorie immer die "Basiskategorie". Für jede der anderen Kategorien wird jeweils ein Koeffizient geschätzt; dieser Koeffizient gibt an, um wieviel sich die Zielvariable (im Mittel) bei dieser Kategorie von der Basiskategorie unterscheidet.

- Intercept: 1 Koeffizient

  Bei jeder Regression wird (einmal) ein Koeffizient für den Intercept geschätzt (vgl. den Abschnitt "Einfache lineare Regression"). Dieser Koeffizient hat (meist) keine besondere interpretatorische Bedeutung.

Frage: Bei einer Regression gebe es für die Einflussvariablen zum Beispiel 2 metrische Variablen und 3 kategorielle Variablen mit 2, 4 bzw. 5 Kategorien. Wieviele Koeffizienten werden in diesem Beispiel geschätzt?

Zur Berechnung der linearen Regression in R siehe die Datei

R-Skript_2-Regression.r

Beispiel: der Datensatz `Koerpergewicht_2.xslx`
Einflussvariablen:

- Geschlecht (kategoriell)
- Alter (metrisch)
- Größe (metrisch)

Zielvariable:

- Gewicht (metrisch)

Ergebnis für die geschätzten Koeffizienten aus linearen Regression berechnet in R:

- Intercept: $\beta_0 = -28,8347$

- Geschlecht: $k = 2$ Kategorien, also 2-1=1 Koeffizient

  $$\beta_1 = -2.6776 \text{ (für “weiblich”)}$$

  Basiskategorie ist “männlich”

  Somit: Frauen sind – bei gleicher Körpergröße und gleichem Alter – im Schnitt um 2,6776 kg leichter als Männer

- Alter: $\beta_2 = 0,1173$

  Pro Lebensjahr nimmt das Gewicht also im Schnitt um 0,1173 kg zu.

- Größe: $\beta_3 = 58,0053$

  Mit jedem Meter Körpergröße nimmt das Gewicht also im Schnitt um 58,0053 kg zu.

  $$58{,}0053\,\frac{\text{kg}}{\text{m}} \;=\; 58{,}0053\,\frac{\text{kg}}{100\,\text{cm}} \;=\; 0{,}580053\,\frac{\text{kg}}{\text{cm}}\,.$$

  Also anders ausgedrückt: Mit jedem Zentimeter Körpergröße nimmt das Gewicht im Schnitt um 0,580053 kg zu.

Beispiel für die Verwendung der Regressionskoeffizienten: Prognoseberechnung

Wie schwer ist im Schnitt ein 1,80 m großer, 37 Jahre alter Mann:

Bewertung von Prognosen

Prognosen sind eines der Haupteinsatzgebiete von KI-Systemen. Um die Leistungsfähigkeit so eines Systems zu bewerten, muss schon bei der Entwicklung des Systems die Genaugkeit der Prognosefunktion f systematisch berechnet

werden. Dies geschieht durch Kennzahlen. Eine der am besten geeigneten Kennzahlen ist der sogenannte *Mean Absolute Error* (MAE):

$$\mathrm{MAE}(f) \;=\; \frac{1}{n}\sum_{i=1}^{n}\left|y_i - f(x_i)\right|.$$

Der MAE hat eine klare und einfache Bedeutung: $\mathrm{MAE}(f) = 6{,}2$ bedeutet, dass der Algorithmus den Wert der Zielvariablen durchschnittlich bis auf $\pm 6{,}2$ genau schätzt.

Übung:
Laden Sie die Daten aus der Datei

`Maschinendaten.xlsx`

in R und machen Sie folgendes:

1. Führen sie eine lineare Regression durch, wobei "Qualitaetsparameter" die Zielvariable ist. Wählen Sie als Einflussvariablen die Spalten Werkzeug, Maschine und Vorschub. Berechnen Sie auch den MAE.

2. Prognostizieren Sie die Qualität eines produzierten Teiles bei Verwendung von Werkzeug 3, Maschine A, Vorschub von 2.5.

2.3 (Binäre) Klassifikation: Logistische Regression

Binäre Klassifikation:

- wie Regression, nur dass die Zielvariable nominal mit 2 Kategorien (0 und 1) ist

- Schätzung der Koeffizienten wie bei Regression, aber die Koeffizienten β_j sind nicht mehr so leicht interpretierbar

- positive Werte des Koeffizienten β_j bedeuten: je höher der Wert der entsprechenden Einflussvariablen, desto wahrscheinlicher hat die Zielvariable den Wert "1";
  negative Werte des Koeffizienten β_j bedeuten: je höher der Wert der entsprechenden Einflussvariablen, desto wahrscheinlicher hat die Zielvariable den Wert "0";

Seien β_0 (Intercept) und $\beta_1,\ \beta_2,\ \ldots,\ \beta_m$ die geschätzten Koeffizienten und $x_1,\ x_2,\ \ldots,\ x_m$ konkrete Werte (Ausprägungen) der entsprechenden Einflussvariablen. Dann können hieraus Prognosen dafür berechnet werden, dass die Zielvariable den Wert "1" hat. Berechne

$$z \ = \ \beta_0 \ + \ \beta_1 \cdot x_1 \ + \ \beta_2 \cdot x_2 \ + \ \ldots \ \beta_m \cdot x_m$$

Dann ist die Wahrscheinlichkeit, dass die Zielvariable den Wert "1" hat, gleich

$$\frac{\exp(z)}{1 + \exp(z)} \ .$$

Bewertung von Prognosen

Auch im Fall einer Klassifikation, muss schon bei der Entwicklung des Systems die Genaugkeit der Prognosen systematisch berechnet werden. Eine übliche Kennzahl hierfür ist die **Misclassification Error Rate**. Bei einer (binären) Klassifikation gibt es nur zwei Möglichkeiten, nämlich $y = 0$ oder $y = 1$. Die Prognose $\hat{y} = 0$ oder $\hat{y} = 1$ ist also entweder richtig oder falsch. (Nur ein bisschen falsch wie in der Regression geht hier nicht.) Die Misclassification Error Rate zählt den prozentualen Anteil an falschen Prognosen.

Berechnung einer Klassifikation (mittels logistischer Regression) in R: siehe die Datei

R-Skript_2-Klassifikation.r

Übung:
Übung: Laden Sie die Daten aus der Datei
`Maschinendaten.xlsx`
in R und machen Sie folgendes:

1. Führen sie eine Klassifikation durch, wobei "Ausschuss" die Zielvariable ist. Wählen Sie als Einflussvariablen die Spalten Werkzeug, Maschine und Vorschub. Berechnen Sie auch die Misclassification Error Rate.

2. Berechnen Sie die (geschätzte) Wahrscheinlichkeit für Ausschuss, falls das Teil mit Werkzeug 3, Maschine A und einem Vorschub von 2.5 produziert wird. Können Sie durch eine geeignete Werkzeug- und Maschinenauswahl die Ausschusswahrscheinlichkeit reduzieren?

3 Supervised Learning (II): Nichtparametrische Verfahren

3.1 Einführung

Maschinelle Lernverfahren funktionieren im Wesentlichen wie klassische statistische Verfahren und sind nichts grundsätzlich neues. Viele Verfahren, die heute unter dem Begriff "Maschinelle Lernverfahren" oder "selbstlernende Algorithmen", eingeordnet werden, sind statistische Verfahren, die von Informatikern entwickelt wurden. In der Informatik werden für statistische Verfahren jedoch andere Begriffe verwendet als in der klassischen Statistik:

Bezeichnung in Statistik	**Bezeichnung in Informatik/KI**
Datensatz	Trainingsdaten
statistisches Verfahren z.B. logistische Regression, Support Vector Machines	Lern-Algorithmus z.B. logistische Regression, Support Vector Machines
Berechnung der Schätzung	Trainieren des Algorithmus / Lernen
Untersuchung von Zusammenhängen zwischen x und y	Mustererkennung

Betrachten wir wieder unser Beispiel aus dem Regressionskapitel: Bremsweg in Abhängigkeit von der Geschwindigkeit.

Es handelt sich also um ein überwachtes Lernproblem, wobei "Geschwindigkeit" die Einflussvariable ist und "Bremsweg" die Zielvariable ist.

Die Daten haben die Form:

	speed	**dist**
1	4	2
2	4	10
3	7	4
4	7	22
5	8	16
6	9	10
7	10	18
8	10	26
⋮	⋮	⋮

Streudiagramm:

Den tendenziellen Zusammenhang zwischen Geschwindigkeit und Bremsweg kann man durch eine Regressionsgerade beschreiben:

Wenn man so möchte, dann kann man dieses lineare Regressionsverfahren als ein maschinelles Lernverfahren bezeichnen:
Der Datensatz sind unsere Trainingsdaten, das lineare Regressionsverfahren ist der Lernalgorithmus, bei der Berechnung der Regression (wie in der Datei R-Skript_2-Regression.r beschrieben) wird das Verfahren trainiert. Die Regressionsgerade bzw. die in der Regression berechneten Regressionskoeffizienten beschreiben das in den Daten gefundene Muster.

Soweit ist das also nichts Neues. Was ist denn nun das Neue an den maschinellen Lernverfahren: Viele maschinelle Lernverfahren sind sogenannte nichtparametrische oder nichtlineare Verfahren.
Die lineare Regression ist ein parametrisches Verfahren, weil hierbei dem Verfahren bereits ein Modell vorgegeben ist, dass nur von wenigen Parametern abhängt:
Das Regressionsmodell lautet in unserem Bremsweg-Beispiel:

$$\text{Bremsweg} = \beta_0 \ + \ \beta_1 \cdot \text{Geschwindigkeit} \ + \ \text{Zufallsabweichung}$$

bzw. in etwas genauerer Formelschreibweise

$$y_i \ = \ \beta_0 \ + \ \beta_1 \cdot x_i \ + \ \varepsilon_i,$$

wobei ε_i die Zufallsabweichung von der Regressionsgerade bei der i-ten Fahrt bezeichnet.

Hierbei ist also schon ein genaues Modell vorgegeben, das besagt, wie der Zusammenhang zwischen Geschwindigkeit und Bremsweg aussieht: ein linearer Zusammenhang. Das Modell muss dabei vorher der Datenanalyst vorgeben und wenn das Modell "falsch" ist, dann sind auch die Ergebnisse aus der linearen Regression nicht gut. Im Bremsweg-Beispiel entspricht unser Modell ja auch tatsächlich nicht der Realität. Der Zusammenhang zwischen Geschwindigkeit und Bremsweg ist in Wirklichkeit nicht linear, sondern quadratisch. Besser wäre also ein quadratisches Modell:

$$y_i \;=\; \beta_0 \;+\; \beta_1 \cdot x_i \;+\; \beta_2 \cdot x_i^2 \;+\; \varepsilon_i.$$

Die Parameter β_0, β_1 und β_2 lassen sich hier auch mit einer linearen Regression berechnen: einfach im Datensatz eine Spalte mit der quadrierten Geschwindigkeit ergänzen und dann die lineare Regression (wie gehabt) in R berechnen. Hier ist das Ergebnis (blaue Linie):

Im obigen Text steht bzgl. dem Modell das Wort "falsch" in Anführungszeichen. Es ist nämlich nicht die Aufgabe eines Modells "richtig" zu sein in dem Sinn, dass das Modell die Realität exakt wiedergibt. Jedes Modell ist nur eine Vereinfachung der Realität. In unserem Bremsweg-Beispiel ist die lineare Kurve zwar "falsch" aber - zumindest bei den niedrigen Geschwindigkeiten - bereits eine sehr gute (für viele Zwecke ausreichende) Beschreibung des Zusammenhangs. Der Statistiker George Box schrieb hierzu: "All models are wrong; some models are useful".

Wie findet man aber nun ein gutes ("nützliches") Modell, das die Realität genügend gut beschreibt? Hierzu gibt es insbesondere drei Möglichkeiten:

1. durch Anschauen der Daten (schwierig bei mehreren Einflussvariablen),
2. durch physikalische Überlegungen oder
3. durch die Wahl eines nichtparametrischen Verfahrens.

Nichtparametrische Verfahren sind statistische Verfahrens, die keine Modellvorgabe benötigt und das Modell eigenständig aus den Daten berechnen. Es handelt sich hierbei also um eine Regression der Form

$$y_i \;=\; f(x_i) \;+\; \varepsilon_i,$$

wobei die Funktion f irgendeine unbekannte Funktion ist, ohne dass die Form der Funktion (linear, quadratisch etc.) vorgegeben ist. Die Funktion f wird von nichtparametrischen Verfahren rein aus den Daten berechnet.
Maschinelle Lernverfahren sind oft nichtparametrische Verfahren und bei der Verwendung solcher Verfahren muss man ein paar Dinge beachten, die bei parametrischen Verfahren (z.B. lineare Regression) noch keine Rolle gespielt haben.

Folgendes Bild zeigt den theoretischen Zusammenhang zwischen zwei Variablen x und y:

In einer idealen Welt ohne Messfehler und ohne Zufallsabweichungen könnte man Daten erheben, die genau dem theoretischen Zusammenhang folgen würden. Die erhobenen Daten würden dann etwa so aussehen:

Hieraus würde sich der genaue theoretische Zusammenhang leicht rekonstruieren lassen. In der realen Welt mit Messfehlern und Zufallsabweichungen sehen die Daten aber eher so aus:

Wie lässt sich der Zusammenhang zwischen den Variablen x und y nun aus solchen Daten (mit Messfehlern und Zufallsabweichungen) rekonstruieren – ohne Kenntnis irgendeines passenden Modells?

1. Idee: Gehe vor wie bei linearer Regression, nur dass nun jede beliebige Funktion f zugelassen ist:

- Lineare Regression: Modell z.B. wie beim Bremsweg

$$f_\beta(x) \;=\; \beta_0 \;+\; \beta_1 \cdot x \;+\; \beta_2 \cdot x^2 \,.$$

  Suche nun Werte für β_0, β_1 und β_2, die am besten zu den Daten passen, d.h. der (quadratische) Abstand zwischen Modell und Daten ist minimal:

$$\text{Minimiere} \qquad \Big(y_1 - f_\beta(x_1)\Big)^2 + \ldots + \Big(y_n - f_\beta(x_n)\Big)^2$$

- Nichtparametrische Regression: kein Modell vorhanden

$$f(x) \;=\; ???$$

Suche nun nach einer beliebigen Funktion f, die am besten zu den Daten passt, d.h. der (quadratische) Abstand zwischen der Funktion f und den Daten ist minimal:

$$\text{Minimiere} \qquad \Big(y_1 - f(x_1)\Big)^2 + \ldots + \Big(y_n - f(x_n)\Big)^2$$

Wenn man das macht, dann hat man aber das Problem des Overfittings: Die vom Algorithmus errechnete Funktion passt "zu gut" zu den Daten, beschreibt aber keinen allgemeinen Zusammenhang.
Dies sieht man gut an folgendem Datenbeispiel:

Hier folgen die Daten in etwa einem linearen Trend. Geht man aber wie in der 1. Idee beschrieben vor, erhält man keine lineare Funktion f, sondern folgendes unsinniges Ergebnis:

Die nach der 1. Idee berechnete Funktion ist unsinnig, weil sie für die Daten zu kompliziert ist.
Man muss daher einen Kompromiss finden zwischen

- Wie gut soll die Funktion f die Daten beschreiben?
- Wie kompliziert darf die Funktion f sein?

Je besser die Funktion die Daten beschreibt, desto komplizierter wird die Funktion. Bei vielen Verfahren / Algorithmen wird daher die Komplexität der Funktion "bestraft", d.h. einfache Funktionen werden bevorzugt:

2. Idee: Gehe vor wie bei linearer Regression, nur dass nun jede beliebige Funktion f zugelassen ist, aber einfache Funktionen bevorzugt werden:

- Lineare Regression: Modell z.B. wie beim Bremsweg

$$f_\beta(x) \;=\; \beta_0 \;+\; \beta_1 \cdot x \;+\; \beta_2 \cdot x^2 \,.$$

  Suche nun Werte für β_0, β_1 und β_2, die am besten zu den Daten passen, d.h. der (quadratische) Abstand zwischen Modell und Daten ist minimal:

$$\text{Minimiere} \qquad \Big(y_1 - f_\beta(x_1)\Big)^2 + \;\ldots\; + \Big(y_n - f_\beta(x_n)\Big)^2$$

- Nichtparametrische Regression: kein Modell vorhanden

$$f(x) \;=\; ???$$

Suche nun nach einer beliebigen Funktion f, die einen möglichst guten Kompromiss zwischen den Daten und der Komplexität darstellt, d.h. die Kombination aus dem (quadratische) Abstand zwischen der Funktion f und den Daten mit der Komplexität der Funktion f ist minimal:

$$\text{Minimiere} \quad \Big(y_1 - f(x_1)\Big)^2 + \ldots + \Big(y_n - f(x_n)\Big)^2 \;+\; \text{Komplexität}(f)$$

Dieses Vorgehen löst das Problem des Overfittings. Als Ergebnis erhält man für die Beispieldaten:

Sofern genügend Daten vorhanden sind, kann mit solchen Verfahren nun jeder beliebige Zusammenhang aus den Daten errechnet werden. Betrachten wir noch einmal folgende Beispieldaten mit Messfehlern und Zufallsabweichungen vom theoretischen Zusammenhang zwischen zwei Variablen x und y:

Mit dem Vorgehen aus der 2. Idee lässt sich der unbekannte theoretischen Zusammenhang (schwarze Linie) recht gut aus den Daten rekonstruieren (blaue Linie):

3.2 Trainings- und Testdatensatz

In den Kapiteln zu Regression und Klassifikation wurde bereits erwähnt, wie wichtig es ist die Genauigkeit von Prognosen zu bewerten. Jedes maschinelle Lernverfahren im Supervised Learning berechnet ein Modell für den Zusammenhang zwischen den Einflussvariablen und der Zielvariable. Dieser Zusammenhang wird dargestellt durch eine Funktion

$$f : \mathbb{R}^m \to \mathbb{R},$$

wobei m die Anzahl der Einflussvariablen ist. Diese Funktion $f : \mathbb{R}^m \to \mathbb{R}$ heißt auch Prognosefunktion, weil mit ihr zu jedem Datenpunkt x (Einflussvariablen) eine Prognose $\hat{y} = f(x)$ für die Zielvariable y berechnet wird.
Aus der Genauigkeit dieser Prognosen folgt, wie gut das Modell f die Realität abbildet: je genauer die Prognosen, desto besser das Modell. Um die Genauigkeit von Prognosen zu errechnen, haben wir schon den sogenannten

Mean Absolute Error (MAE) kennengelernt:

$$\mathrm{MAE}(f) \;=\; \frac{1}{n}\sum_{i=1}^{n}\left|y_i - f(x_i)\right|.$$

Der MAE ist die mittlere Abweichung der Prognosen $f(x_i)$ von der Realität y_i.

Wird aber nun die Prognosegenauigkeit anhand derselben Daten berechnet, mit denen die Prognosefunktion f berechnet wurde, dann ergeben sich Probleme. Im vorherigen Unterkapitel hatten wir das Beispiel folgender, sehr schlechter Modellfunktion f:

Obwohl das ein sehr schlechtes Modell ist, hat diese Modellfunktion f aber einen Prognosefehler von 0 auf dem Datensatz und hätte somit einen optimalen Fehlerwert $\mathrm{MAE}(f) = 0$. Um die Genauigkeit der Prognosen und damit das Modell f zu bewerten, darf man nicht den selben Datensatz verwenden, mit dem das Modell (also die Funktion f) schon berechnet wurde. Denn bei der Berechnung von f hat der Algorithmus die Daten ja bereits gesehen und berücksichtigt. Das wäre dann in etwa so, wie wenn man nicht die Zukunft, sondern die Vergangenheit vorhersagen müsste.

Für eine valide Bewertung der Prognosegenauigkeit braucht man daher zwei getrennte Datensätze:

1. Trainingsdatensatz: Der Trainingsdatensatz wird verwendet, um das Modell f zu berechnen, also das neuronale Netz, den Entscheidungsbaum, etc.

2. Testdatensatz: Auf dem Testdatensatz wird dann die Genauigkeit der Prognosen berechnet, die das Modell f erzeugt. Das heißt der MAE oder die Misclassification Error wird auf dem Testdatensatz berechnet.

In der Praxis wird das dadurch umgesetzt, dass die verfügbaren Daten zufällig in zwei Teile geteilt werden. Die Aufteilung ist meistens 70% zu 30%. Der größere Teildatensatz (70%) wird als Trainingsdatensatz verwendet, der kleinere Teildatensatz (30%) wird als Testdatensatz verwendet.

3.3 Entscheidungsbäume

Fallbeispiel: In einem Projekt der TH Deggendorf sollten in Zusammenarbeit mit einem Unternehmen Untersuchungen zur Entwicklung eines entscheidungsunterstützenden Systems bei der Herstellung von Glaswolle durchgeführt werden.

Glaswolle wird zur Wäremdämmung bei Gebäuden verwendet. Um Glaswolle zu erhalten werden dünne Glasfäden zerfasert. Dies geschieht ähnlich wie bei der Herstellung von Zuckerwatte.

Konkret ging es darum, Muster zwischen den Einstellungen der Produktionsanlagen und der Qualität der produzierten Glaswolle zu finden. Eine Vorgabe war, dass die von den Algorithmen identifizierten Muster für den Anwender nachvollziehbar sein sollten. Bei der Problemstellung handelt es sich um ein überwachtes Lernproblem:

$$\text{Einstellungen der Produktionsanlagen} \quad \longrightarrow \quad \text{Qualität der Glaswolle}$$

Die meisten maschinellen Lernverfahren wie Support Vector Regression oder Neuronale Netze eignen sich hier nicht, weil sie Black-Box-Lösungen liefern: die von den Algorithmen identifizierten Muster sind für den Anwender nicht nachvollziehbar. Dies liegt daran, dass die Algorithmen eine beliebig komplexe Prognosefunktion

$$f \;:\; \mathbb{R}^m \;\to\; \mathbb{R}\,, \quad (x_1, x_2, \ldots, x_m) \;\mapsto\; f(x_1, x_2, \ldots, x_m)$$

berechnen. Diese Funktion f ist bei diesen Verfahren nicht zugänglich für Interpretationen und kann – aufgrund der hohen Dimensionalität – nicht grafisch dargestellt werden. Hat man z.B. $m = 10$ Einflussvariablen, bräuchte man ein Bild in einem 11-dimensionalen Raum. Es gibt aber auch maschinelle Lernverfahren, die nachvollziehbare Muster erzeugen. Dies ist möglich, indem man sich auf bestimmte Prognosefunktionen f beschränkt. Ein solches Verfahren sind Entscheidungsbäume. Bei Entscheidungsbäumen beschränkt man sich auf Funktionen der Gestalt

$$f(x) \;=\; c_1 \mathbb{1}_{R_1}(x) + c_2 \mathbb{1}_{R_2}(x) + \ldots c_T \mathbb{1}_{R_T}(x)$$

Hierbei ist jeweils R_ℓ ein Rechteck im $\mathbb{R}^m$ und

$$\mathbb{1}_{R_\ell}(x) \;=\; 1 \;\text{ falls }\; x \in R_\ell \qquad \text{und} \qquad \mathbb{1}_{R_\ell}(x) \;=\; 0 \;\text{ falls }\; x \notin R_\ell\,.$$

Wenn sich die Rechtecke nicht überlappen, dann kann man die Funktion auch so darstellen:

$$f(x) \;=\; \begin{cases} c_1 & \text{falls} \quad x \in R_1 \\ \vdots & \vdots \qquad\quad \vdots \\ c_\ell & \text{falls} \quad x \in R_\ell \\ \vdots & \vdots \qquad\quad \vdots \\ c_T & \text{falls} \quad x \in R_T \end{cases}$$

Diese Funktionen lassen sich sehr schön grafisch durch Bäume darstellen. Folgendes Bild zeigt einen möglichen Entscheidungsbaum aus dem Projekt der TH Deggendorf.

Entscheidungsbaum zur Einstellung der Anlage bei der Produktion von Glaswolle

Hierbei sind "Gas.Pressure.in", "Epo_leff", "Gas.Temp" etc. die Einstellparameter der Produktionsanlage (Einflussvariablen). Die Dezimalzahlen in den Kästchen sind genau die Werte c_ℓ der Prognosefunktion f. Dabei handelt es sich also um die jeweilige Prognose für die produzierte Qualität (Zielvariable). Die Qualität wird mit einer Kennzahl zwischen 1 (niedrigste Qualität) und 10 (höchste Qualität) gemessen. Die Prozentzahlen geben noch ergänzend an, wieviele Datenpunkte in den jeweiligen Bereich (also das entsprechende Rechteck R_ℓ) fallen. Das heißt: Wenn der Gasdruck ("Gas.Pressure.in") den Schwellenwert 1,4 erreicht oder überschreitet, dann ist die Prognose für die Qualität gleich 3,7 (ziemlich niedrige Qualität). Im Trainingsdatensatz wurde der Schwellenwert 1,4 für den Gasdruck in 9% der Fälle überschritten.

Frage: Wie sollte die Anlage eingestellt werden, um eine höchstmögliche Qua-

lität zu erreichen?

Wenn $x_1, x_2, \ldots, x_m$ die verschiedenen Einflussvariablen sind, dann werden die Rechtecke durch Schnitte an einem Schwellenwert t_ℓ gebildet. Dabei wird der Raum $\mathbb{R}^m$ bei jedem Schnitt am Schwellenwert t_ℓ in zwei Hälften geteilt:

$$x_j \leq t_\ell \qquad \text{und} \qquad x_j > t_\ell \,.$$

Das nachfolgende Bild zeigt für $m = 2$ Einflussvariablen x_1 und x_2, wie durch solche Schnitte an Schwellenwerten t_ℓ der Raum $\mathbb{R}^2$ in rechteckige Regionen R_ℓ aufgeteilt wird[2]:

Eine hieraus resultierende Prognosefunktion f kann in diesem einfachen Fall in einem 3-dimensionalen Bild gemalt werden (weil wir ja nur $m = 2$ Einflussvariablen haben). Die Höhen der "Treppenstufen" stellen die Werte c_ℓ der Prognosefunktion f dar.

[2]Die nachfolgenden drei Bilder sind aus "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) mit Erlaubnis der Autoren: G. James, D. Witten, T. Hastie and R. Tibshirani

Aufgrund der besonderen Gestalt lässt sich diese Prognosefunktion f auch als Baumdiagramm darstellen. Das nachfolgende Bild zeigt das Baumdiagramm für dieses Bild:

Jeder Schnitt im Raum $\mathbb{R}^m$ entspricht im Entscheidungsbaum einem Knotenpunkt.
Das Baumdiagramm hat zwei Vorteile:

- Das Baumdiagramm kann auch für beliebig viele Einflussvariablen m gezeichnet werden.
- Das Baumdiagramm kann auch von Data-Mining-Laien problemlos interpretiert werden.

Den zweiten der beiden Vorteile sehen wir im folgenden Beispiel.

Beispiel: Spielergehälter im Baseball

Der Datensatz **Hitters** enthält Daten zu Baseballspielern. Neben dem Jahresgehalt (`Salary`), ist unter anderem auch aufgeführt, wie lange der Spieler schon in den höchsten Ligen spielt (`Years`) und wieviele Hits[3] er im letzten Jahr hatte (`Hits`).
Mit Hilfe eines Entscheidungsbaums kann nun dargestellt werden, wie die Zielvariable `Salary` von den Einflussvariablen `Year` und `Hits` abhängt. Vor der Berechnung eines Entscheidungsbaum wurde das Gehalt (`Salary`) logarithmiert, um eine etwas symmetrischere Verteilung zu erhalten:

Das nachfolgende Bild[4] zeigt einen entsprechenden Entscheidungsbaum:

[3] "Als Hit gilt im Baseball, wenn der Batter (Schlagmann) nach einem erfolgreichen Treffer des Balls eine Base erreicht. In der Baseballstatistik erhält der Batter einen Hit, wenn er nach einem gültigen Schlag des Balls ohne den Vorteil eines Errors oder Fielder's Choice die erste Base erreicht." (Wikipedia)

[4] Aus "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) mit Erlaubnis der Autoren: G. James, D. Witten, T. Hastie and R. Tibshirani

Aufgaben:

- Interpretieren Sie die Ergebnisse des Entscheidungsbaumes.
- Zeichnen Sie die entsprechenden Rechtecke R_ℓ in ein Diagramm und geben Sie die resultierende Prognosefunktion f formelmäßig an.
- Wie lautet die Gehaltsprognose für einen Spieler, der bereits seit 6 Jahren in der höchsten Liga spielt und im letzten Jahr 112 Hits hatte? Wie lautet die Gehaltsprognose für einen Spieler, der im letzten Jahr 125 Hits hatte, aber erst seit 2 Jahren in der höchsten Liga spielt?

Berechnung eines Entscheidungsbaumes

Um einen Entscheidungsbaum zu berechnen, sind zunächst zwei wesentliche Punkte von Bedeutung:

1. Der von den Einflussvariablen $x_1, x_2, \ldots, x_m$ aufgespannte Raum $\mathbb{R}^m$ muss in Rechtecke $R_1, R_2, \ldots, R_T$ aufgeteilt werden.

2. Für jedes Rechteck R_ℓ muss ein Prognosewert c_ℓ bestimmt werden.

Wir betrachten diese beiden Punkte zunächst am Beispiel der Regression:

Im Fall der Regression mit Entscheidungsbäumen wird typischerweise als Gütekriterium die Summe der quadrierten Abweichungen der Prognosen von den realen Daten (Kleinste-Quadrate-Verlustfunktion) verwendet. Hieraus ergibt sich, dass als Prognosewert c_ℓ das arithmetische Mittel optimal ist. Das heißt, zur Berechnung von c_ℓ nimmt man alle Trainingsdaten, die in das Rechteck R_ℓ fallen, und wählt als c_ℓ das arithmetische Mittel der Werte der Zielvariablen y_i von diesen Trainingsdaten.

Die Aufteilung von $\mathbb{R}^m$ in Rechtecke R_1, R_2, ..., R_T geschieht rekursiv. Dabei wird in jedem Schritt eine Einflussvariable x_j und ein Schwellenwerts t ausgewählt, so dass sich die Summe der quadrierten Abweichungen der Prognosen von den realen Daten in jedem Schritt bestmöglich verringert.

Im Fall der (binären) Klassifikation (mit einer 0-1-Zielvariablen) ist das Vorgehen änlich. Das arithmetische Mittel über die Werte der Zielvariablen y_i ist dann genau der Anteil $\hat{p}_\ell$ der Trainingsdaten im Rechteck R_ℓ mit Wert $y_i = 1$. Der einzige Unterschied zur Regression ist, dass als Gütekriterium nicht die Summe der quadrierten Abweichungen der Prognosen verwendet wird. Stattdessen ist z.B. der sogenannte Gini-Index

$$\hat{p}_\ell \cdot (1 - \hat{p}_\ell)$$

gebräuchlich. Der Gini-Index ist umso kleiner (gut), je homogener die Trainingsdaten im Rechteck R_ℓ sind, also wenn entweder (fast) alle Werte $y_i = 0$ oder (fast) alle Werte $y_i = 1$ sind. Der Gini-Index ist umso größer (schlecht), je inhomogener die Trainingsdaten im Rechteck R_ℓ sind.

Vermeiden von Overfitting: Regularisierung

Wie in der Einleitung zum maschinellen Lernen besprochen, muss jedes maschinelle Lernverfahren einen Kompromiss finden: Einerseits soll die Prognosefuntion gut zu den Daten passen, andererseits soll die Prognosefunktion nicht überkomplex sein. Dies wird häufig durch eine Regularisierung gelöst, die die Komplexität bestraft. Dies ist auch bei Entscheidungsbäumen der Fall. Ein gutes Maß für die Komplexität eines Entscheidungsbaumes ist die Zahl T der Endknoten des Baumes. Die Endknoten des Entscheidungsbaumes entsprechen genau den Rechtecken R_ℓ. Je mehr Endknoten ein Entscheidungsbaum hat, desto komplexer ist der Entscheidungsbaum. Das folgende Bild[5] zeigt den Entscheidungsbaum für den Datensatz **Hitters**, wenn die Zahl der Endknoten nicht beschränkt wird. Dabei sieht man übrigens auch, dass eine Einflussvariable (z.B. `Years`) problemlos auch mehrfach ausgewählt werden kann.

[5] Aus "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) mit Erlaubnis der Autoren: G. James, D. Witten, T. Hastie and R. Tibshirani

Wird die Zahl der Endknoten nicht begrenzt, dann entstehen zu komplexe Entscheidungsbäume, die zum Overfitting führen. Während der anfangs gezeigte (optimale) Entscheidungsbaum nur $T = 3$ Endknoten hatte, besitzt dieser Entscheidungsbaum $T = 12$ Endknoten.
Eine solches Overfitting wird beim Entscheidungsbaum mit Hilfe einer Regularisierung ähnlich wie bei Support Vector Machines verhindert. Anstelle der Summe der quadrierten Abweichungen der Prognosen wird

$$\text{Kennzahl}(f) \;=\; \text{Risiko}(f) \;+\; \underbrace{\lambda \cdot \text{Komplexität}(f)}_{\text{Regularisierung}}$$

minimiert. Dabei ist in der Regression das Risiko gleich

$$\begin{aligned}\text{Risiko}(f) \;&=\; \frac{\big(f(x_{1,1}, x_{1,2}, \ldots, x_{1,k}) - y_1\big)^2 \;+\; \ldots \;+\; \big(f(x_{n,1}, x_{n,2}, \ldots, x_{n,k}) - y_n\big)^2}{n} \\ &=\; \text{mittlere quadratische Abweichung der Prognosen von} \\ &\quad\;\; \text{den tatsächlichen Daten}\end{aligned}$$

und das Maß für die Komplexität ist

$$\text{Komplexität}(f) \;=\; \text{Anzahl T der Endknoten des Entscheidungsbaumes } f\,.$$

Mit dem Parameter λ wird gesteuert, wie stark regularisiert werden soll, d.h. wie stark die Komplexität der Funktion f bestraft werden soll. (Einfache Entscheidungsbäume mit wenigen Endknoten werden bevorzugt!)

Praxisteil

Zur Berechnung von Entscheidungsbäumen in R siehe die Datei
R-Skript_3-Entscheidungsbaeume.r

Übung: Laden Sie die Daten aus der Datei
`Maschinendaten.xlsx`
in R und machen Sie folgendes:

1. Führen sie eine Regression durch, wobei "Qualitaetsparameter" die Zielvariable ist. Wählen Sie als Einflussvariablen die Spalten Werkzeug, Maschine und Vorschub. Verwenden Sie für die Regression einen Entscheidungsbaum als maschinelles Lernverfahren. Teilen Sie dabei den Datensatz im Verhältnis (70% zu 30%) in Trainings- und Testdaten auf. Trainieren Sie das maschinelle Lernverfahren auf dem Trainingsdatensatz und berechnen Sie anschließend den mittleren Prognosefehler (also den MAE: Mean Absolute Error) auf dem Testdatensatz.

2. Prognostizieren Sie die Qualität eines produzierten Teiles bei Verwendung von Werkzeug 3, Maschine A, Vorschub von 2.5.

3.4 Neuronale Netze

Betrachten wir wieder ein überwachtes Lernproblem, also eine Regression oder eine Klassifikation. Dabei haben wir wieder

- Einflussvariablen: $x_1,\ x_2,\ \ldots,\ x_m$
- Zielvariable: y

Neuronale Netze kann man als Verallgemeinerung der linearen Regression (bzw. auch der logistischen Regression) betrachten:

Lineare Regression

Modell:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_m x_m + \varepsilon$$

Neuronale Netze

Für ein neuronales Netz werden zunächst sogenannte "abgeleitete Einflussvariablen" gebildet:

$$\begin{aligned}
z_1 &= \sigma\big(w_{0,1} + w_{1,1}x_1 + w_{2,1}x_2 + \ldots + w_{m,1}x_m\big) \\
z_2 &= \sigma\big(w_{0,2} + w_{1,2}x_1 + w_{2,2}x_2 + \ldots + w_{m,2}x_m\big) \\
\vdots \quad & \vdots \quad \vdots \\
z_{m_1} &= \sigma\big(w_{0,m_1} + w_{1,m_1}x_1 + w_{2,m_1}x_2 + \ldots + w_{m,m_1}x_m\big)
\end{aligned}$$

Das Modell lautet dann:

$$y = g\big(\beta_0 + \beta_1 z_1 + \beta_2 z_2 + \ldots + \beta_{m_1} z_{m_1}\big) + \varepsilon$$

Folgendes Bild zeigt nochmal das Modell mit den üblichen Bezeichnungen:

Würde man als Aktivierungsfunktion σ die Identitätsfunktion $\sigma(x) = x$ und als Ausgabefunktion g ebenfalls die Identitätsfunktion $g(t) = t$ wählen, so würde man genau wieder die lineare Regression erhalten. (Die logistische Regression erhält man als Spezialfall, wenn man als Aktivierungsfunktion σ die Identitätsfunktion $\sigma(x) = x$ und als Ausgabefunktion g die sogenannte Sigmoid-Funktion wählt.)

Das Modell lässt sich auch in einer **Kompakteren Schreibweise** aufschreiben:

$$X := \big(1, x_1, x_2, \ldots, x_m\big)^\mathsf{T}, \quad \beta := \big(\beta_0, \beta_1, \beta_2, \ldots, \beta_{m_1}\big)$$

$$Z^{(1)} := \big(1, z_1, z_2, \ldots, z_{m_1}\big)^{\mathsf{T}}, \quad W^{(1)} := \begin{pmatrix} w_{0,1} & \ldots & w_{m,1} \\ \vdots & \ddots & \vdots \\ w_{0,m_1} & \ldots & w_{m,m_1} \end{pmatrix}$$

Hierbei handelt es sich um ein Modell mit einem Hidden Layer, weil wir nur einmal abgeleitete Einflussvariablen gebildet haben. Man kann den Prozess der Bildung abgeleiteter Einflussvariablen aber beliebig oft wiederholen. Dann hat man ein Modell mit mehreren Hidden Layer. Hierzu nimmt man die abgeleiteten Einflussvariablen und bildet aus diesen nochmal neue abgeleitete Einflussvariablen, in dem man sie mit neuen Gewichten versieht und in die Aktivierungsfunktion steckt.

Modell mit 1 Hidden Layer:

$$y \;=\; g\big(\beta Z^{(1)}\big) + \varepsilon \;=\; g\Big(\beta\sigma\big(W^{(1)}X\big)\Big) + \varepsilon$$

Modell mit 2 Hidden Layer:

$$Z^{(1)} := \sigma\big(W^{(1)}X\big), \quad Z^{(2)} := \sigma\big(W^{(2)}Z^{(1)}\big)$$

$$y \;=\; g\big(\beta Z^{(2)}\big) + \varepsilon \;=\; g\Big(\beta\sigma\big(W^{(2)}\sigma(W^{(1)}X)\big)\Big) + \varepsilon$$

Modell mit k Hidden Layer:

$$Z^{(1)} := \sigma\big(W^{(1)}X\big), \quad Z^{(2)} := \sigma\big(W^{(2)}Z^{(1)}\big), \; \ldots, \quad Z^{(k)} := \sigma\big(W^{(k)}Z^{(k-1)}\big)$$

$$\begin{aligned} y \;&=\; g\big(\beta Z^{(k)}\big) + \varepsilon \;= \\ &=\; g\Big(\beta\sigma\big(W^{(k)}\sigma(\ldots\sigma\big(W^{(2)}\sigma(W^{(1)}X)\big)\big)\ldots\big)\Big) + \varepsilon \end{aligned}$$

Neuronale Netze lassen sich auch grafisch anschaulich darstellen.
Zunächst grafisch das Modell im Fall der linearen Regression:

Neuronales Netz mit 1 Hidden Layer

Neuronales Netz mit 2 Hidden Layer

Typische Aktivierungsfunktionen σ:

Sigmoid-Funktion mit Skalenparameter s: $\sigma(t) = \frac{1}{1+\exp(-st)}$

Sigmoid-Funktion mit $s = 1$ (rot), $s = 0,5$ (blau) und $s = 10$ (schwarz)

Typische Ausgabefunktionen g:

- Regression: identische Funktion $g(t) = t$
- Klassifikation: softmax Funktion

Berechnung des Neuronalen Netzes mit Backpropagation:

Um ein Neuronales Netz zu trainieren, das dann gute Prognosen liefern soll, müssen die Gewichte $w_{j,l}$ und die Parameter β_j gut gewählt werden. Hierzu kann zum Beispiel der Backpropagation-Algorithmus verwendet werden. Diesen Algorithmus sehen wir uns im Folgenden am Beispiel der Regression mit $g(t) = t$ bei nur einem Hidden Layer an. Der allgemeine Fall funktioniert im Wesentlichen identisch, ist aber formelmäig aufwändiger.
Wir verwenden wieder unsere kompakte Schreibweise:

$$X \;:=\; \big(1, x_1, x_2, \ldots, x_m\big)^\mathsf{T}, \quad \beta \;:=\; \big(\beta_0, \beta_1, \beta_2, \ldots, \beta_{m_1}\big)$$
$$Z^{(1)} := \big(1, z_1, z_2, \ldots, z_{m_1}\big)^\mathsf{T}, \quad W^{(1)} := \begin{pmatrix} w_{0,1} & \ldots & w_{m,1} \\ \vdots & \ddots & \vdots \\ w_{0,m_1} & \ldots & w_{m,m_1} \end{pmatrix}$$

Im Fall der linearen Regression haben wir also

$$Y \;=\; \beta X \;+\; \varepsilon$$

Die Regressionskoeffizienten β werden in der linearen Regression durch die Lösung des Optimierungsproblems

$$\min_\beta \sum_{i=1}^n \big(y_i - \beta X_i\big)^2$$

berechnet. Hierbei handelt es sich um ein relativ einfaches konvexes Optimierungsproblem, dass analytisch mit Hilfe von Matrizenrechnungen gelöst werden kann.
Im Fall des Neuronalen Netzes mit 1 Hidden Layer haben wir

$$Y \;=\; g\Big(\beta\sigma\big(W^{(1)}X\big)\Big) \;+\; \varepsilon$$

und das entsprechende Optimierungsproblem lautet

$$\min_{\beta, W^{(1)}} \sum_{i=1}^{n} \Big(y_i - g\big(\beta \sigma\big(W^{(1)} X\big)\big)\Big)^2$$

Hierbei handelt es sich um ein schwieriges und nicht-konvexes Optimierungsproblem, das näherungsweise mit Hilfe eines Optimierungsalgorithmus gelöst werden muss.
Hierzu wird häufig **Backpropagation** verwendet. Dabei handelt es sich um ein Gradientenverfahren. Die Grundidee eines jeden Gradientenverfahrens wird durch folgendes Bild gut dargestellt:

Der Gradient ist die Ableitung einer Funktion mit mehreren Variablen. Dabei ist der Gradient ein Wegweiser, der einem den Weg "nach unten" zeigt. Das Verfahren beginnt an einem irgendwie gewählten Startpunkt und wandert dann schrittweise über die zu optimierende Funktion. Bei jedem Schritt wird der Gradient berechnet und als Wegweiser für den nächsten Schritt verwendet. Das Verfahren endet, wenn es in einer Talsohle angelangt ist – und es somit in keiner Richtung mehr weiter nach unten geht.

Gradientenverfahren gibt es in verschiedenen Varianten. Um die Grundidee des Gradientenverfahrens "Backpropagation" besser zu verstehen, beschränken wir uns im folgenden auf den Spezialfall

- Regression: Ausgabefunktion $g(t) = t$
- Neuronales Netz mit 1 Hidden Layer

Die Grundidee ist im allgemeinen Fall identisch, aber die Formeln sind unübersichtlicher. Für den gewählten Spezialfall lautet das Optimierungsproblem:

$$\min_{\beta, W} \sum_{i=1}^{n} \underbrace{\Big(y_i - \underbrace{\beta \sigma(W X_i)}_{= f(X_i)}\Big)^2}_{=: R_i(\beta, W)}$$

Hieraus ergeben sich die Gradienten ("Wegweiser zum Optimum") durch partielles Ableiten :

$$\frac{\partial R_i}{\partial \beta_j} = \underbrace{-2(y_i - f(X_i))}_{=: \delta_i} z_{j,i}$$

$$\frac{\partial R_i}{\partial w_{j,l}} = \underbrace{-2(y_i - f(X_i))\beta_j \sigma'(W_l^T X_i)}_{=: s_{j,i}} x_{l,i}$$

Setzt man die obere Formel in die untere Formel ein, dann erhält man die Backpropagation-Gleichung:

$$s_{j,i} = \sigma'(W_l^T X_i)\beta_j \, \delta_i$$

Der Name "Backpropagation" kommt daher, weil man sich bei der Berechnung dieser Gleichung rückwärts durch das neuronale Netz bewegt. Nach der Berechnung des Gradienten erfolgt der Update-Schritt:

$$\beta_j^{neu} = \beta_j^{alt} - \gamma \sum_{i=1}^{n} \frac{\partial R_i}{\partial \beta_j}(\beta^{alt}, W^{alt})$$

$$w_{j,l}^{neu} = w_{j,l}^{alt} - \gamma \sum_{i=1}^{n} \frac{\partial R_i}{\partial w_{j,l}}(\beta^{alt}, W^{alt})$$

Die neuen Werte für β und W ergeben sich, indem zu den alten Werten der (negative) Gradient addiert wird. Der Faktor $\gamma \in (0, \infty)$ ist dabei die sogenannte Schrittweite und gibt an, in welchen Abständen neue Gradienten ("Wegweiser") berechnet werden.

Praxisteil

Zur Berechnung von neuronalen Netzen in R siehe die Datei
R-Skript_3-neuronalesNetz.r

Übung: Laden Sie die Daten aus der Datei
`Maschinendaten.xlsx`
in R und machen Sie folgendes:

1. Führen sie eine Regression durch, wobei "Qualitaetsparameter" die Zielvariable ist. Wählen Sie als Einflussvariablen die Spalten Werkzeug, Maschine und Vorschub. Verwenden Sie für die Regression neuronale Netze als maschinelles Lernverfahren. Teilen Sie dabei den Datensatz im Verhältnis (70% zu 30%) in Trainings- und Testdaten auf. Trainieren Sie das maschinelle Lernverfahren auf dem Trainingsdatensatz und berechnen Sie anschließend den mittleren Prognosefehler (also den MAE: Mean Absolute Error) auf dem Testdatensatz. Führen Sie die Berechnung mehrfach durch und variieren Sie dabei die Anzahl der Hidden Layer und die Anzahl der Hidden Units. Entscheiden Sie sich basierend auf den Ergebnissen für eine geeignete Anzahl an Hidden Layer und Hidden Units.

2. Prognostizieren Sie die Qualität eines produzierten Teiles bei Verwendung von Werkzeug 3, Maschine A, Vorschub von 2.5.

4 Bayes Netze

4.1 Bedingte Wahrscheinlichkeiten

Ereignisse, die mit Unsicherheit behaftet sind, werden mathematisch durch Zufallsvariablen modelliert. Hierzu verwendet man Großbuchstaben, häufig z.B. X oder Y, aber es kann jeder andere Buchstabe verwendet werden. Ein unsicheres Ereignis tritt nur mit einer gewissen Wahscheinlichkeit ein, für die Wahrscheinlichkeit schreibt man den Großbuchstaben P.

Beispiel: Die Zufallsvariable X soll nun modellieren, ob es morgen regnet. Wenn wir nun für "Regen" die Zahl 1 und für "kein Regen" die Zahl 0 schreiben, dann ist

$$P(X = 1) \;=\; \text{Wahrscheinlichkeit, dass es morgen regnet}$$

und

$$P(X = 0) \;=\; \text{Wahrscheinlichkeit, dass es morgen nicht regnet}\,.$$

Die Wahrscheinlichkeit ist eine Zahl zwischen 0 und 1 und die Summe der einzelnen Wahrscheinlichkeiten für eine Zufallsvariable muss sich auf 1 aufsummieren. Daher gilt

$$P(X = 0) \;=\; 1 - P(X = 1)\,.$$

Häufig verfügt man auch über ein Vorwissen, das Einfluss auf die Wahrscheinlichkeiten hat. Zum Beispiel könnte die Wahrscheinlichkeit, dass es morgen regnet, "a priori" (d.h. ohne Vorwissen) gleich

$$P(X = 1) \;=\; 0{,}2 \;=\; 20\%$$

sein. Wenn es aber bereits heute schon regnet, dann erhöht das die Wahrscheinlichkeit, dass es morgen regnen wird. Für das Ereignis, dass es heute regnet, verwenden wir nun den Buchstaben Y. Dann bezeichnet

$$P(X = 1|Y = 1)$$

die Wahrscheinlichkeit, dass es morgen regnet ($X = 1$), unter der Bedingung/Voraussetzung, dass es heute regnet ($Y = 1$). Es könnte zum Beispiel sein, dass

$$P(X = 1) = 0,2\,, \quad P(X = 1|Y = 1) = 0{,}65\,, \quad P(X = 1|Y = 0) = 0{,}13\,.$$

Wahrscheinlichkeiten für X mit einem Vorwissen über ein anderes Ereignis Y bezeichnet man als bedingte Wahrscheinlichkeiten und werden in der Form

$$P(X = \cdot | Y = \cdot)$$

geschrieben. Dabei steht an erster Stelle (also vor dem Strich) das Ereignis, für das eine Wahrscheinlichkeit angegeben wird, und an zweiter Stelle (also hinter dem Strich) das Ereignis, unter dessen Bedingung die angegebene Wahrscheinlichkeit gilt. In obigem Zahlenbeispiel muss dann entsprechend

$$P(X = 0|Y = 1) \;=\qquad\qquad \text{und} \qquad P(X = 0|Y = 0) \;=$$

gelten.

Es ist auch möglich, *gemeinsame* Wahrscheinlichkeiten für beliebig viele Ereignisse anzugeben. Zum Beispiel modelliere U, ob es morgen eine Überschwemmung geben wird ($U = 1$) oder nicht ($U = 0$). Außerdem modelliere Z, ob der heutige Wetterbericht für morgen Regen ankündigt ($Z = 1$) oder nicht ($Z = 0$). Dann bezeichnet im obigem Beispiel

$$P(X = 1,\, U = 0)$$

die Wahrscheinlichkeit für das "Doppelereignis", dass es morgen regnet und es morgen keine Überschwemmung gibt. Und der Ausdruck

$$P(X = 1, U = 0 \,|\, Y = 0, Z = 1)$$

bedeutet:

Falls $P(Z = j) > 0$, so berechnet sich die bedingte Wahrscheinlichkeit aus der Bayes-Regel

$$P(X = i \,|\, Y = j) \;=\; \frac{P(X = i, Y = j)}{P(Y = j)}\,,$$

wobei i und j Platzhalter für die verschiedenen Werte sind, die X und Y annehmen können. Die Formel gilt entsprechend auch bei mehreren Ereignissen, also zum Beispiel

$$P(X = 1, U = 0 \,|\, Y = 0) \;=\; \frac{P(X = 1, U = 0, Y = 0)}{P(Y = 0)}$$

und

$$P(X = 1, U = 1 \,|\, Y = 1, Z = 0) \;=\; \frac{P(X = 1, U = 1, Y = 1, Z = 0)}{P(Y = 1, Z = 0)}\,.$$

Beispiele für bedingte Wahrscheinlichkeiten:

- Aus einem Kartenstapel mit 32 Spielkarten (Deutsches Blatt ohne 6er) werden 2 Karten (Ziehen ohne Zurücklegen) gezogen. Hierbei modelliere X_1 das Ergebnis der ersten gezogenen Karte und X_2 das Ergebnis der zweiten gezogenen Karte. Dann ist

  $$P(X_2 = \text{Herzkönig} \,|\, X_1 = \text{Grasunter}) \;=$$

  $$P(X_2 = \text{Herzkönig}) \;=$$

Aus der Bayes-Regel lassen sich eine Unmenge von Wahrscheinlichkeiten herleiten. Zum Beispiel gilt dann ja auch

$$P(X = i, Y = j) \;=\; P(X = i \,|\, Y = j) \cdot P(Y = j)$$

und somit

$$P(Y = j \,|\, X = i) \;=\; \frac{P(X = i, Y = j)}{P(X = i)} \;=\; \frac{P(X = i \,|\, Y = j) \cdot P(Y = j)}{P(X = i)}\,.$$

Diese Wahrscheinlichkeit wird auch als inverse Wahrscheinlichkeit bezeichnet, weil dabei die Rolle von X und Y vertauscht werden: In der Wahrscheinlichkeit $P(Y = j \,|\, X = i)$ auf der linken Seite der Gleichung ist Y das Ereignis, für das eine Wahrscheinlichkeit angeben wird, und X die Bedingung. In der Wahrscheinlichkeit $P(X = i \,|\, Y = j)$ auf der rechten Seite der Gleichung ist es umgekehrt: X ist das Ereignis, für das eine Wahrscheinlichkeit angeben wird, und Y die Bedingung.
Solche Rechnungen mit bedingten Wahrscheinlichkeiten unter Verwendung der Bayes-Regel sind für Bayes Netze von entscheidender Bedeutung. Um das ein wenig einzuüben, sehen wir uns zuvor das folgende sehr wichtige Beispiel an.

Beispiel: Krankheitsdiagnose

In einer Bevölkerung von 100 Millionen Menschen haben 10000 Personen eine bestimmte Krankheit. Zur Diagnose der Krankheit gibt es einen Test, der bei einer erkrankten Testperson die Krankheit mit 99-prozentiger Wahrscheinlichkeit erkennt (Sensitivität); bei nicht-erkrankten Testpersonen erkennt der Test die Nicht-Erkrankung mit 98-prozentiger Wahrscheinlichkeit. Herr T. Stark lässt sich bei einer allgemeinen Routineuntersuchung testen und erhält das niederschmetternde Testergebnis, dass er erkrankt sei. Wie groß ist die Wahrscheinlichkeit, dass Herr Stark tatsächlich erkrankt ist?

Um die gesuchte Wahrscheinlichkeit zu berechnen, verwenden wir folgende Modellierung: Die Zufallsvariable K gebe an, ob die Person tatsächlich krank ist ($K = 1$) oder nicht ($K = 0$); die Zufallsvariable T gebe an, ob der Test auf Krankheit entscheidet ($T = 1$) oder nicht ($T = 0$).

.

4.2 Repräsentation von Unsicherheit in Bayes Netzen

Im letzten Beispiel über den Test zur Diagnose einer Krankheit besteht folgender kausale Zusammenhang:

Wie wir gesehen haben, ist dieser Zusammenhang aber mit einer Unsicherheit verbunden, die mit Wahrscheinlichkeiten angegeben wird. Wenn die Person erkrank ist, erkennt das der Test eben nur mit einer gewissen Wahrscheinlichkeit. Außerdem gibt es Fehlalarme. Über das Vorliegen der Krankheit besteht also Unsicherheit – trotz dem Vorliegen eines Testergebnisses! Die Ereignisse treten nur mit gewissen Wahrscheinlichkeiten ein. Dieses Beispiel stellt ein (extrem) einfaches Bayes Netz mit folgenden Wahrscheinlichkeiten dar:

$P(K=0)$	$P(K=1)$
0,9999	0,0001

	$P(T=0\|K=*)$	$P(T=1\|K=*)$
$K=0$	0,98	0,02
$K=1$	0,01	0,99

Wir erweitern dieses Bayes Netz nun auf

mit folgenden Wahrscheinlichkeiten:

$P(R=0)$	$P(R=1)$
0,80	0,20

$P(M=0)$	$P(M=1)$
0,40	0,60

und

	$P(K=0\|R=*,M=*)$	$P(K=1\|R=*,M=*)$
$R=0,M=0$	0,999946	0,000054
$R=0,M=1$	0,99988	0,00012
$R=1,M=0$	0,99991	0,00009
$R=1,M=1$	0,99985	0,00015

und

	$P(T=0\|K=*)$	$P(T=1\|K=*)$
$K=0$	0,98	0,02
$K=1$	0,01	0,99

	$P(S=0\|K=*)$	$P(S=1\|K=*)$
$K=0$	0,90	0,10
$K=1$	0,75	0,25

Bayes Netze stellen kausalen Beziehungen zwischen verschiedenen Variablen dar. Jede Variable wird in einem Bayes Netz durch einen Knoten dargestellt. Die Pfeile zwischen den Knoten stellen die kausalen Beziehungen zwischen den Variablen dar. Dadurch ergibt sich ein sogenannter "gerichteter Graph". Die Pfeilrichtungen stellen dar, in welche Richtungen die Kausalitäten wirken. So nimmt in unserem Beispiel das Vorhandensein der Krankheit (Variable K) Einfluss auf das Testergebnis (Variable T), aber das Testergebnis hat natürlich keinen kausalen (ursächlichen) Einfluss auf die Krankheit.
Die Knoten, die direkte Pfeile zu einem Knoten X haben, heißen Elternknoten von X. In unserem Beispiel hat K die beiden Elternknoten R und M. Der Knoten T hat nur den Elternknoten K. Der Knoten R besitzt keinen Elternknoten. Die Knoten, zu denen Pfeile ausgehend vom Knoten X führen, heißen Kindknoten von X. In unserem Beispiel hat K die beiden Kindknoten T und S. Der Knoten R hat nur K als Kindknoten.
Der grichtete Graph eines Bayes Netzes muss immer *azyklisch* sein. Das heißt, wenn man ausgehend von einem beliebigen Knoten entlang der Pfeile durch den Graphen wandert, dann darf es nicht möglich sein, dass man irgendwann wieder beim Ausgangsknoten ankommt. Der Graph muss also so sein, dass keine Rundreisen ("Zyklen") möglich sind.

Die kausalen Einflüsse sind in der Regel nicht deterministisch: selbst wenn die Krankheit vorliegt, erkennt dies in unserem Beispiel der Test nur mit einer gewissen Wahrscheinlichkeit. Zu jedem Knoten gibt es daher eine Wahrscheinlichkeitstabelle, die die Wahrscheinlichkeiten für die entsprechende Variable in Abhängigkeit von den Elternknoten angibt (bedingte Wahrscheinlichkeiten).
Die Wirkung der Mechanismen im Bayes Netz sind lokal gedacht. Das heißt, der Zustand einer Variablen hängt kausal gesehen nur vom Zustand seiner direkten Eltern ab. Als Folge davon ist jede Variable im Bayes Netz unabhängig von allen anderen Variablen, wenn der Zustand der direkten Eltern bekannt ist. In unserem Beispiel ist somit z.B.

$$P(T = 1|K = 0, M = 1) \;=\; P(T = 1|K = 0)$$

und

$$P(T = 1|K = 0, S = 1) \;=\; P(T = 1|K = 0)\,.$$

In Bayes Netzen gibt es drei grundlegende Kausalstrukturen: seriell, divergierend und konvergierend.

4.3 Multiplikationssatz und Inferenz im Bayes Netz

Aus der Bayes-Regel folgt allgemein für Zufallsvariablen $X_1, X_2, \ldots, X_n$ der allgemeine Multiplikationssatz

$$\begin{aligned} P(X_1 = i_1, X_2 = i_2, \ldots, X_n = i_n) \;=\; & P(X_n = i_n|X_1 = i_1, \ldots, X_{n-1} = i_{n-1}) \cdot \\ & P(X_{n-1} = i_{n-1}|X_1 = i_1, \ldots, X_{n-2} = i_{n-2}) \cdot \\ & \cdot \ldots \cdot \\ & P(X_2 = i_2|X_1 = i_1) \cdot P(X_1 = i_1) \end{aligned}$$

Wenn nun $X_1, X_2, \ldots, X_n$ die Zufallsvariablen zu allen Knoten in einem Bayes Netz sind, dann ergibt sich hieraus aufgrund der im vorherigen Unterkapitel erwähnten lokalen Kausalitäten und der damit verbunden Unabhängigkeiten:

$$\begin{aligned} &P(X_1 = i_1, X_2 = i_2, \ldots, X_n = i_n) \\ &\qquad = \prod_{\ell=1}^{n} P(X_\ell = i_\ell | \text{alle Elternknoten von } X_\ell) \end{aligned}$$

Für unser Beispiel heißt das

$$\begin{aligned} &P(T = t, S = s, K = k, R = r, M = m) \; = \\ &\quad = \; P(T = t | K = k) \cdot P(S = s | K = k) \cdot \\ &\qquad P(K = k | R = r, M = m) \cdot P(R = r) \cdot P(M = m)\,, \end{aligned}$$

wobei in unserem Beispiel t, s, k, r, m jeweils die Werte 0 und 1 annehmen können.
Es gilt also beispielsweise

$$\begin{aligned} &P(T = 1, S = 0, K = 0, R = 0, M = 1) \; = \\ &\quad = \; P(T = 1 | K = 0) \cdot P(S = 0 | K = 0) \cdot \\ &\qquad P(K = 0 | R = 0, M = 1) \cdot P(R = 0) \cdot P(M = 1) \; = \\ &\quad = \end{aligned}$$

Aufgrund der Additionsregel für Wahrscheinlichkeiten gilt aber beispielsweise auch

$$\begin{aligned} &P(T = 1, K = 0, M = 1) \; = \\ &\quad = \sum_{s \in \{0;1\}} \sum_{r \in \{0;1\}} P(T = 1, S = s, K = 0, R = r, M = 1) \; = \\ &\quad = \end{aligned}$$

.

.

.

Zusammen mit der Bayes-Regel lassen sich mit diesen Rechenregeln alle möglichen (bedingten) Wahrscheinlichkeiten im Bayes Netz errechnen.
Als Beispiele berechnen wir im folgenden

(a) die Wahrscheinlichkeit, dass der Test bei einer (zufällig ausgewählten) Person auf "krank" entscheidet.

(b) die Wahrscheinlichkeit, dass der Test bei einem Raucher auf "krank" entscheidet.

(c) die Wahrscheinlichkeit, dass eine (zufällig ausgewählten) Person tatsächlich krank ist, falls von der Person bereits bekannt ist, dass der Test auf "krank" entschieden hat.

Die Berechnung der bedingten Wahrscheinlichkeit aus (b) in unserem Bayes Netz nennt man "kausale Inferenz", da ein kausaler Zusammenhang besteht: Rauchen begünstigt die Krankheit und das Vorhandensein der Krankheit hat einen kausalen Einfluss auf das Testergebnis.
Im Gegensatz dazu ist die Berechnung der bedingten Wahrscheinlichkeit aus (c) in unserem Bayes Netz eine sogenannte "diagnostische Inferenz", da sie umgekehrt zu den Kausalitäten verläuft: das Testergebnis ist eine Folge der Krankheit, aber das Testergebnis verursacht nicht die Krankheit.

.

.

.