Wissensbasierte Systeme
Probabilistische Netze

Programmentwurf

Studiengang Angewandte Informatik

Duale Hochschule Baden-Würtemberg Stuttgart

von

Nico-Alexei Hein

und

Andreas Rau

bei

Prof. Dr. Dirk Reichardt

Januar 2015

Nico-Alexei Hein

Matrikelnummer, Kurs	7146604, TINF12A
Firma	Hewlett Packard, Bad Homburg

Andreas Rau

Matrikelnummer, Kurs	4186494, TINF12A
Firma	IBM, Böblingen

Contents

1 Aufgabenstellung

> „Entwickeln Sie eine Software, welche bei Eingabe (Datei, vgl. Beispielformat) von Testdaten die entsprechenden Klassifikationen mit Hilfe der Bayes Netze geeignet bestimmt und ausgibt.“

Wortlaut der Aufgabenstellung

2 Herangehensweise und Ansatz

Mit Probabilistichen Netzen (Bayesian Networks) werden insbesondere Aussagen über unbeobachtete statistische Variablen gemacht, welche zuvor aus Beobachteten stochastisch abhängigen Variablen gefolgert wurden.

Ziel dieser Aufgabe ist eine Prognose der Hochschulqualifikation mittels probabilistischem Netz zu realisieren, bei der eine Wahrscheinlichkeitsvariable (die Hochschulqualifikation) vorhergesagt wird. Die Klassifikation der Datensätze folgt daher immer dem selben Schema. (Es eigneten sich von daher auch andere weniger flexible Algorithmen des maschinellen Lernens wie One Rule oder Entscheidungsbäume.)

Um in einem probabilistischen Netz schlussfolgern zu können, muss die Struktur des Netzes festgelegt sein (Knoten = Zufallsvariablen und gerichtete Kanten = „direkter Einfluss“) und die Parameter eines jeden Knotens (Conditional Propability Table - CPT) müssen bekannt sein.

Das Erstellen von probabilistischen Netzten kann entweder manuell oder automatisch erfolgen. Wird ein probabilistisches Netz automatisch generiert, so kann man weiter Unterscheiden zwischen Parameterlernen und Strukturlernen. Bei erstem ist die Struktur schon vorgegeben - also oftmals manuell erstellt - und die CPT werden aus vorliegenden Datensätzen generiert. Beim Strukturlernen wird das gesamte Netz aus den Trainigsdaten generiert. Die manuelle Arbeit wird somit minimiert. Das manuelle Erstellen der Netze ist insbesondere dann sinnvoll, wenn die logischen Abhängigkeiten eines Netzes wie auch die Wahrscheinlichkeiten bekannt sind. Sind die Abhängigkeiten bekannt, nicht jedoch die Wahrscheinlichkeiten so eignet sich das Parameterlernen auf Testdaten. Wird es zudem kompliziert die Abhängigkeiten zwischen den Variablen zu erkennen kann das Strukturlernen abhilfe schaffen.

Analyse der gegebenen Informationen

Die 17 in dem Dokument `P_A02.csv` gegebenen beobachteten Variablen sind:

1. Qualifikation
2. Schnitt
3. Bundesland
4. Mathe
5. Physik
6. Deutsch
7. Schultyp
8. OLT-Mathe
9. OLT-Deutsch
10. Studierfähigkeitstest
11. Alter
12. Geschlecht
13. Jahreseinkommen der Eltern
14. Staatsbürgerschaft
15. Studiengang
16. Zwischenkalk
17. Abschluss

Bevor die Abhängigkeiten der Variablen und die Variablen selbst diskutiert werden, präzisieren wir an dieser Stelle die Aufgabenstellung unter Beachtung der bis hierhin gewonnenen Erkenntnisse:

Das zu erstellende probabilistische Netz soll eine Empfehlung für den geeignetsten Studiengang ausgeben oder vom Studium abraten. Da es sich bei dem Tool um ein Online-Beratungssystem handelt welches vor Studienantritt ausgeführt wird, kann davon ausgegangen werden, dass die Variablen 15 - 17 unbekannt sind. Des weiteren sind ausschließlich die Variablen „Studiengang“ und „Abschluss“ für die Aussage relevant, wodurch die Variable Zwischenkalk überflüssig wird und nicht weiter betrachtet wird.

Die Hochschulqualifikation kann durch zwei Abfragen im probabilistischen Netz erfolgen. Erstere mit $P(Abschluss|...)$ wobei aus der Prognose „nicht geschafft“ das Abraten vom Studiengang folgt. Und zweite mit $P(Studiengang|...)$ welche den zu Empfehlenden Studiengang prognostiziert.

Bei Betrachtung der Variablen fallen einige Abhängigkeiten sofort ins Auge wie „Mathe“ und „OLT-Mathe“ oder die allgemein Bekannte Abhängigkeit zwischen„Jahreseinkommen der Eltern“ und „Qualifikation“. Bei vielen anderen Variablen ist es jedoch nur möglich stochastische Abhängigkeiten zu vermuten wie zwischen „Qualifikation“ und „Studierfähigkeitstest“. Da diese Abhängigkeiten auch nur statistisch belegt werden können ist eine direkte Definition der Struktur des Netzes so nicht möglich. Eine Ausführliche Beschreibung zur Herangehensweise bei der Erstellung der Struktur folgt aus diesem Grund unter Abschnitt 4.

Da für die Anwendung der Bayes'schen Regel der Satz der vollständigen Ereignisdisjunktion gilt, sollten alle Wahrscheinlichkeitsvariablen diskret sein. Aus diesem Grund

werden im folgenden Abschnitt alle Variablen auf ihre Eigenschaften untersucht und ggf. diskretisiert (wobei kontinuierliche Variablen auf einen qualitativen diskreten Raum abgebildet werden). Dabei wird beachtet, das der Informationsverlust der durch die Einteilung in Klassen entsteht möglichst gering bleibt.

3 Diskretisierung der Wahrscheinlichkeitsvariablen

Für die Definition der diskreten Klassen kann kein Gesetz angewandt werden. Da wir hier jedoch vor einem Problem, ähnlich dem eines Histogramms stehen, können wir uns an der in der Mathematik Vorlesung vorgestellten Faustregel (für $n < 1000$) zur Bestimmung der Anzahl der Klassen $number \sim \sqrt{n}$, wobei n die Größe der Domäne darstellt, orientieren (rer. nat. Bernd Klöss, 2014).

Wie im Einzelnen abgebildet wird, ist nun aufgelistet:

Die Qualifikation ist gegeben durch
$Rg(Qualifikation) = \{Abitur, FHReife, Meister, Techniker\}$

Der Schnitt ist zwar eine diskrete Wahrscheinlichkeitsvariable, doch sie hat mit ihren Verwirklichungen $\{1.0, 1.1, \ldots 4.0\}$ (ab 4.1 gilt als nicht bestanden) 31 Ausprägungen. Für eine bessere Vergleichbarkeit sind nicht mehr als 20 Klassen empfehlenswert rer. nat. Bernd Klöss (2014). Durch $\sqrt{31} = 5.6$ kommen wir auf Intervalle von $[0, 0.5]$ und 6 Klassen:

$$\begin{aligned} Rg(Schnitt) =& \{x_1 = [1.0, 1.4], x_{1.5} = [1.5, 1.9], x_2 = [2.0, 2.4], x_{2.5} = [2.5, 2.9]\} \\ & \cup \{x_3 = [3.0, 3.4], x_{3.5} = [3.5, 4.0]\} \end{aligned}$$

Bundesland Auch wenn in den Testdaten nur wenige Bundesländer Auftauchen, kann das Netz mit allen möglichen Ausprägungen der Variable „Bundesland“ erstellt werden:
$Rg(Bundesland) = \{Schleswig - Holstein, Hamburg, Bremen,$
$Mecklenburg - Vorpommern, Berlin, Brandenburg, Sachsen,$
$Sachsen - Anhalt, Niedersachsen, Th\ddot{u}ringen, Nordrhein - Westfahlen,$
$Rheinland - Pfalz, Saarland, Hessen, Baden - W\ddot{u}rttemberg, Bayern\}$
Die Wahrscheinlichkeiten für in den Testdaten nicht vorkommende Bundesländer müssten mit minimalen werten definiert werden.

Mathe Der Schnitt wird um eine Klasse erweitert $Rg(Mathe) = Rg(Schnitt) \cup \{'keine'\}$. Der Wert „keine“ kann beobachtet werden. Es bedeutet in diesem fall nicht, das keine Daten vorliegen, sondern, das es keine Note gibt. (Vergl. Schultyp und Studierfähigkeitstest)

Physik Es gilt selbiges wie bei Mathe: $Rg(Physik) = Rg(Mathe)$

Deutsch Es gilt selbiges wie bei Mathe: $Rg(Deutsch) = Rg(Mathe)$

Schultyp Aus den Testdaten lassen sich vier Ausprägungen ableiten:

$$\begin{aligned} Rg(Schultyp) =\{&AllgemeinesGymnasium, Technisches \\ &Gymnasium, Wirtschaftsgymnasium, n.a.\} \end{aligned}$$

Es ist jedoch anzunehmen, dass bei Angabe von „n.a.“ diese Variable unbeobachtet ist. Damit entfernen wir diese Ausprägung für die Variable Schultyp.

OLT-Mathe Aus den Testdaten lässt sich schließen, dass sich beim OLT 100 Punkte erzielen lassen. Durch $number \sim \sqrt{100}$ ergeben sich 10 Klassen. Außerdem scheint dieser Test pflicht zu sein.

$$\begin{aligned} Rg(OLTM) =\{&x_1 = [0, 10], x_2 = [11, 20], x_3 = [21, 30], \\ &x_4 = [31, 40], x_5 = [41, 50], x_6 = [51, 60], x_7 = [61, 70], \\ &x_8 = [71, 80], x_9 = [81, 90], x_{10} = [91, 100]\} \end{aligned}$$

OLT-Deutsch Für OLT-Deutsch gilt selbiges wie für OLT-Mathe
$Rg(OLTD) = Rg(OLTM)$

Studierfähigkeitstest Anhand der Testdaten für den Test lässt sich ablesen, das dieser Test keine Pflicht ist, und die Ergebnisse im Interval $[0, 1000]$ liegen. Da hier keine höhere Präzision als bei den OLTs erwartet werden kann, ist eine genauere Differenzierung auch nicht notwendig. Der Wert „n.a.“ bedeutet, dass hier keine Daten vorliegen. Damit wird diese Variable unbeobachtet, was uns dazu veranlasst diesen Wert zu entfernen. Wir legen somit für die Klassen fest:

$$\begin{aligned} Rg(Studienfähigkeitstest) = \{&x_1 = [0, 100], x_2 = [101, 200], x_3 = [201, 300], \\ &x_4 = [301, 400], x_5 = [401, 500], x_6 = [501, 600], \\ &x_7 = [601, 700], x_8 = [701, 800], x_9 = [801, 900], \\ &x_{10} = [901, 1000]\} \end{aligned}$$

Alter Um hier eine Abgeschlossene Menge zu bekommen, beschränken wir uns auf die in den Testdaten vorkommenden 17 - 27 Akzeptiert. Diese 11 Ausprägungen werden vorerst beibehalten.
$Rg(Alter) = \{[17, 27]\}$

Geschlecht Für das Geschlecht werden zwei Klassen angenommen. Ethische Fragen hierzu werden der Einfachheit halber ignoriert. $Rg(Geschlecht) = \{m, w\}$

Jahreseinkommen der Eltern Bei den Jahreseinkommen lässt sich kein abschließender Rahmen finden. Aus diesem Grund nähern wir uns über des Intervall angegeben in den Testdaten ($[36k, 209k]$).Über $\sqrt{174}$ erhalten wir 13 Klassen. Da jedoch 13 Klassen - für die relative Wahrscheinlichkeiten auf Basis von nur 100 Datensätzen ausgerechnet werden - sehr viel sind verwenden wir stattdessen nur 10 Klassen:

$$\begin{aligned} Rg(Jahreseinkommen) =& \{x_1 = [0, 50k], x_2 = [51k, 60k], x_3 = [61k, 70k], x_4 = [71k, 80k], \\ & x_5 = [81k, 90k], x_6 = [91k, 100k], x_7 = [101k, 110k], \\ & x_8 = [111k, 120k], x_9 = [121, 130], x_{10} = [131, \infty]\} \end{aligned}$$

Staatsbürgerschaft Aus den Testdaten lassen sich drei Klassen ableiten:
$Rg(Staatsbürgerschaft) = \{deutsch, EUBuerger, NonEuropean\}$

Studiengang Aus den Testdaten lassen sich die folgenden fünf Studiengänge und damit Klassen ableiten:

$$\begin{aligned} Rg(Studiengang) =& \{Maschinenbau, SozialeArbeit, Elektrotechnik, \\ & Wirtschaftswissenschaften, Informatik\} \end{aligned}$$

Zwischenkalkulation Der Schnitt wird erweitert:

$$\begin{aligned} Rg(Zwischenkalkulation) =& Rg(Schnitt) \\ & \cup \{Abgebrochen, x_4 = [4.1, 4.4], \\ & x_{4.5} = [4.5, 4.9], x_5 = [5.0, 5.4], x_{5.5} = [5.5, 5.9]\} \end{aligned}$$

Die 4.0 wird noch auf die $x_{3.5}$ gemapped, da diese noch als bestanden gilt.

Abschluss Es gilt selbiges wie für die Zwischenkalkulation.

Umsetzung

Die oben aufgeführten Klassen werden auf die Testdaten durch ein kleines Java Tool angewandt. Zuvor haben wir jedoch die Datei `P_A02.csv` in UTF-8 encoded um einfacher alle Umlaute in Java erkennen zu können. (Beim Einlesen werden zudem alle Kommata gegen Punkte ersetzt um das Erkennen von Zahlen in Java leichter zu gestalten.)

```
1 public boolean transform(){
2    this.deleteUnknown();
3    this.schnitt = this.toNote(this.schnitt);
```

```
4     this.mathe = this.toNote(this.mathe);
5     this.physik = this.toNote(this.physik);
6     this.deutsch = this.toNote(this.deutsch);
7     this.oltMathe = this.toOltGroup(this.oltMathe);
8     this.oltDeutsch = this.toOltGroup(this.oltDeutsch);
9     this.studierfaehigkeitstest = this.toTestGroup(this.
          studierfaehigkeitstest);
10    //this.alter = this.toAlterGroup(this.alter);
11    this.jahreseinkommen = this.toEinkommenGroup(this.jahreseinkommen);
12    this.zwischenkalk = this.toNote(this.zwischenkalk);
13    this.abschluss = this.toNote(this.abschluss);
14    //this.abschluss = this.toAbgebrochen(this.abschluss);
15    return true;
16  }
17  public boolean deleteUnknown(){
18    if(this.schultyp.equals("n.a.")) this.schultyp = "*";
19    if(this.studierfaehigkeitstest.equals("n.a.")) this.
          studierfaehigkeitstest = "*";
20    return true;
21  }
22  public String toNote(String note){
23    try{
24      Double x = Double.parseDouble(note);
25      if(x==4) x=x-0.1;
26      //map in 0.5 intervals
27      x = Math.floor(x*2)/2;
28      note = x.toString();
29    }catch(Exception e){
30      return note; //do nothing if not a number
31    }
32    return note;
33  }
34  public String toOltGroup(String olt){
35    try{
36      Double x = Double.parseDouble(olt);
37      //map 0-100 into 10 Groups
38      if(x==100) x--;
39      x = Math.floor(x/10);
40      olt= x.toString();
41    }catch(Exception e){
42      return olt; //do nothing if not a number
43    }
44    return olt;
45  }
46  public String toTestGroup(String test){
47    try{
48      Double x = Double.parseDouble(test);
```

```
49      //map 0-1000 into 10 Groups
50      if(x==1000) x--;
51      x = Math.floor(x/100);
52      test= x.toString();
53    }catch(Exception e){
54      return test; //do nothing if not a number
55    }
56    return test;
57  }
58  public String toEinkommenGroup(String einkommen){
59    try{
60      Double x = Double.parseDouble(einkommen);
61      //map 0-inf k into 10 Groups
62      if(x<50000) return "0.0"; //in the first group
63      if(x>130000) return "9.0"; //in the 10th group
64      x = x/1000;
65      x = Math.floor((x-50)/10)+1; //Split everithing in between into 8
            Groups
66      einkommen= x.toString();
67    }catch(Exception e){
68      return einkommen; //do nothing if not a number
69    }
70    return einkommen;
71  }
```

In dem vorausgegangenen Quelltext sehen wir die Methode „transform“, die „n.a.“ aus den Testdaten entfernt, und durch ein Sternchen ersetzt, welches für „unbeobachtet“ steht. Im Anschluss werden in der Methode die Funktionen aufgerufen, welche die Diskretisierung vornehmen. Diese sind analog zu den vorausgegangenen Angaben zu verstehen.

4 Aufstellen des Probabilistischen Netzes

Nun kommen wir zurück zu der Diskussion der Abhängigkeiten. Hier betrachten wir die Variable „Zwischenkalk“ nicht mehr näher, da sie weder zu den beobachteten noch zu den abgefragten Variablen gehört. Von den folgenden Abhängigkeiten ist sehr Wahrscheinlich auszugehen:
(Notation: X ~ Y steht für X abhängig von Y)

- OLT-Mathe ~ Mathe
- OLT-Deutsch ~ Deutsch
- Schnitt ~ Mathe $\wedge$ Deutsch
- Qualifikation ~ Schultyp

Bei der Aufstellung dieser Liste haben wir uns auf die gegebenen Sachzusammenhänge und unseren Verstand verlassen.

Bei anderen Beziehungen können begründete Vermutungen aufgestellt werden. In der folgenden Liste haben wir einige Vermutungen formuliert.

- Studierfähigkeitstest ~ Mathe ∧ Deutsch ∧ Physik ∧ Schnitt - *Es könnte sich jedoch auch nur um eine Teilmenge dieser handeln, oder weitere Variablen umfassen.*
- Abschluss ~ Jahreseinkommen - *Über derartige zusammenhänge wird häufig in der Politik diskutiert.*
- Qualifikation ~ Jahreseinkommen - *Diese Aussage nimmt auch die politischen Diskussionen zum Thema als Grundlage*
- Studierfähigkeitstest ~ Qualifikation ∧ Schultyp ∧ Schnitt ∧ Bundesland - *Diese Vermutung basiert der Annahme, das der Test nicht ganz unabhängig von anderen Leistungen und Umständen sein kann. Die gewählten Variablen könnten einem Betrachter einen groben Überblick über den bisherigen Bildungsweg verraten.*

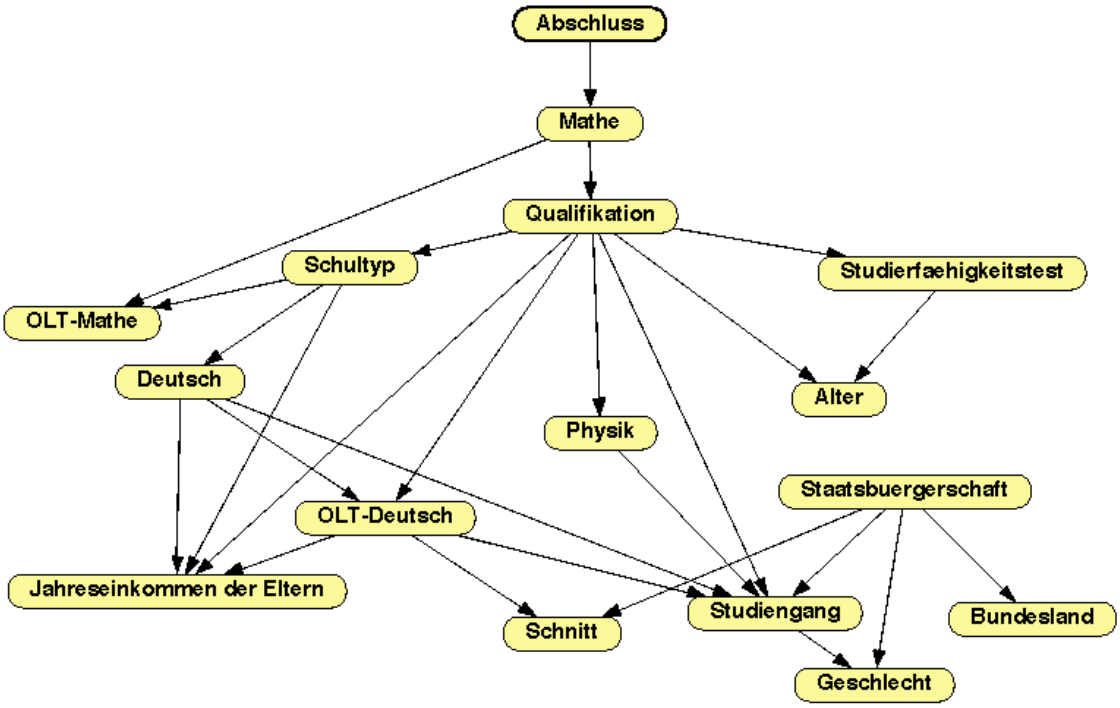
Über weitere Abhängigkeiten lassen sich Vermutungen aufstellen, die ohne Beweis einer Abhängigkeit zu Missverständnissen oder falschen Annahmen führen können. So immer die Abhängigkeit einer Variable von der Staatsbürgerschaft oder des Geschlechts. Ebenfalls kritisch ist das Alter.

Wegen möglichen ethischen Konflikte haben wir uns dazu entschieden die Daten für sich sprechen zu lassen und automatisiertes Strukturlernen anzuwenden. So können wir auf einzelne Tests in denen wir Abhängigkeiten prüfen verzichten.

4.1 Strukturlernen

Nach vorangegangener Recherche haben wir das Tool `org.openmarkov.jar` ausfindig gemacht, welches die Möglichkeit des strukturellen Lernens bietet. Ein Screenshot des Tools findet sich im Anhang unter A. Das Strukturlernen ist über das heuristische Optimierungsverfahren des Bergsteigeralgorithmuses implementiert.

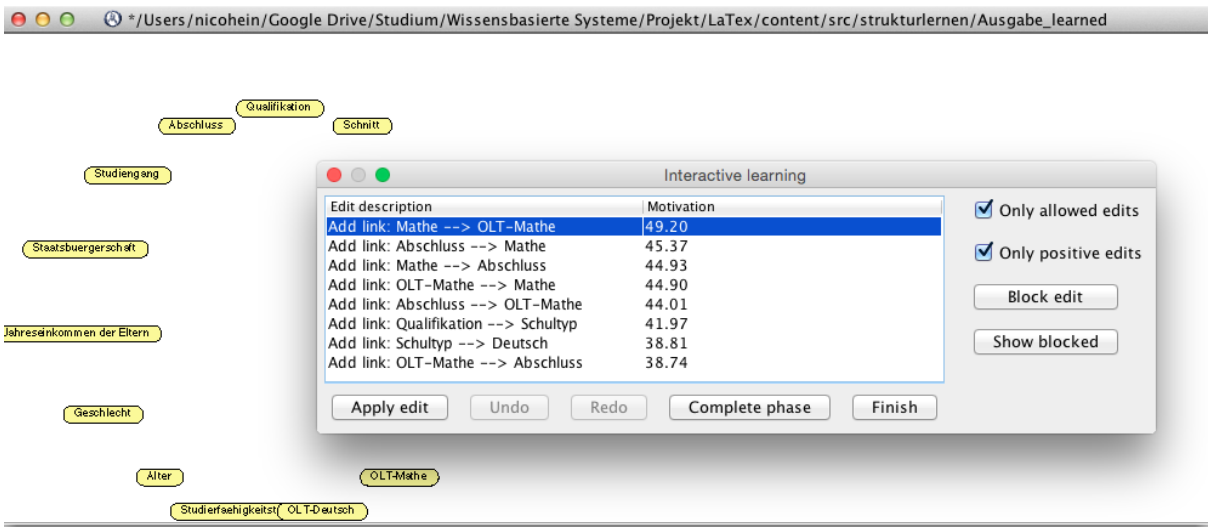
Ein erster Durchlauf vollautomatisierten Strukturlernen resultiert in folgendem Netz:



Es kann beobachtet werden, dass bereits fast alle als sehr Wahrscheinlich angenommenen Abhängigkeiten vertreten sind. Wieder erwarten scheint der Schnitt jedoch Stärker von OLT-Deutsch und der Staatsbürgerschaft abzuhängen als von Mathe und Deutsch. Unter den weiteren vermuteten Abhängigkeiten finden sich nur sehr wenige wieder, was jedoch durchaus Plausibel ist, da es sich nur um wage Vermutungen handelte. Schauen wir jedoch genauer auf die Abhängigkeiten, so finden wir Abhängigkeiten die uns an der kompletten Korrektheit zweifeln lassen. So hängt Mathe z.B. direkt vom Abschluss ab, Deutsch jedoch vom Schultyp, der wiederum von der Qualifikation und diese von Mathe anhängt. Ein zweites Beispiel ist der Studierfähigkeitstest welcher ausschließlich von der Qualifikation abzuhängen scheint. Wegen dieser Zweifel starten wir einen Zweiten Anlauf, bei dem wir statt auf vollautomatisiertes lernen auf interaktives lernen setzen.

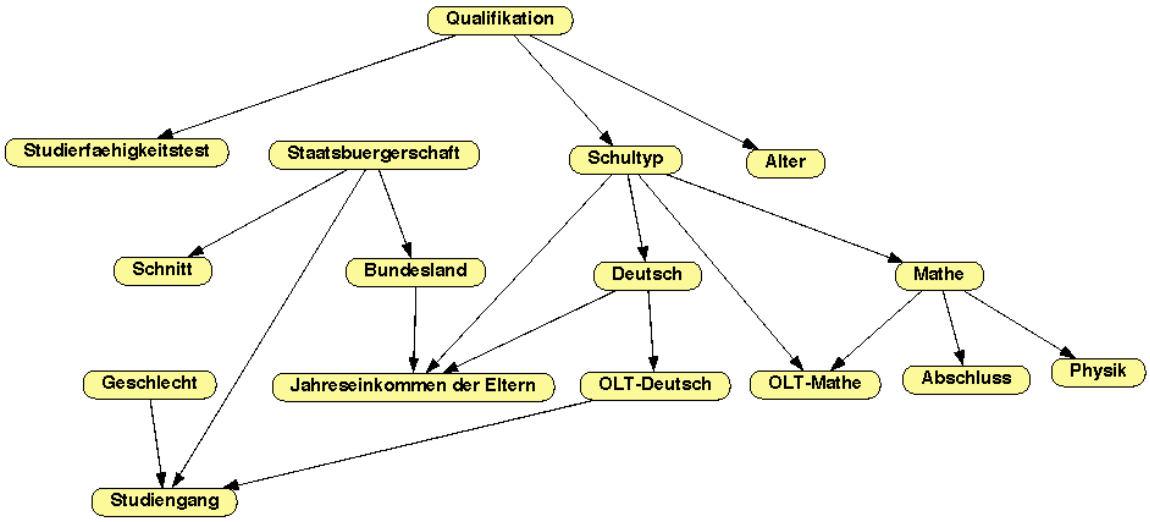
Strukturlernen mit Einflussnahme

Beim interaktiven Lernen können wir direkten Einfluss auf die zu setzenden Links nehmen. Dabei bekommen wir als User eine Liste angezeigt die uns in Frage kommende Links mit einer zugehörigen Motivation anzeigt. Es bleibt entsprechend die Möglichkeit Abhängigkeiten zu ignorieren und/oder das Netzwerk durch die Reihenfolge des Hinzufügens der Links zu beeinflussen.



Eine Liste in welcher Reihenfolge wir die Links hinzugefügt haben findet sich im Anhang unter B.

Das Resultat ist folgendes:



Das abgebildete Netz ist deutlich strukturierter als das welches komplett automatisch generiert wurde. Hier finden sich nun auch die Symmetrien von OLT Mathe ~ Mathe und OLT Deutsch ~ Deutsch deutlicher wieder. Überraschend ist z.B. dass der Abschluss nur direkt von Mathe abhängt. Letztendlich finden wir auch in diesem Netz - auch wenn es plausibler erscheint - noch Zusammenhänge die nicht ganz unseren Erwartungen entsprechen.

Die weitere Entwicklung wird jetzt am Beispiel des zweiten Netzes vorgestellt. Ein vergleich der Performance folgt dann unter 5.

Die mithilfe von `org.openmarkov.jar` erstellten Netze setzten wir nun in Netica um. Netica wurde beriets in der Vorlesung ausführlich besprochen und wird aus diesem Grund hier nicht weiter Diskutiert.

Bemerkung: *Bei der Umsetzung übernehmen wir für die Variablen alle Klassen die auch in den Testdaten vorkommen, nicht alle Möglichen.*

Der nächste Schritt zu einem voll funktionstüchtigen Netz ist das aufstellen der CPT. Diese erstellen wir durch Parameterlernen in Netica. Wir wählen diesen Weg, da die Wahrschinlichkeiten nicht gegeben sind sondern lediglich absloute Häufigkeiten, aus denen wir die relativen errechnen können. Dies kann ein Algorithmus zuverlässiger und schneller als wir es per Hand könnten.

4.2 Parameterlernen

Bemerkung: *Da Netica in der uns zur verfügung gestellten Version nur 15 Nodes unterstützt, wir ohne die Zwischenkalkulation noch immer 16 haben mussten wir uns für eine weitere Variable die wegfällt entscheiden. Da bei der interaktiven Erstellung des Netzes eine sehr hohe Abhängigkeit von Mathe und Physik festgestellt wurde (Motivation 81.03), und Physik in beiden Netzen die Erreichbarkeit von den anderen Nodes nicht beeinflusst (Qualifikation - Studiengang existiert neben Qualifikation - Physik - Studiengang (im ersten Netz) oder Mathe - Physik ist der einzige Link zu Physik (im zweiten Netz)) haben wir uns für diese Variable entschieden.*

Für das Parameterlernen erscheint es sinnvoll die Daten in Test und Trainingsdaten einzuteilen. Da wir jedoch nur einen recht kleinen Datensatz mit 100 Elementen haben ist der zu erwartende Qualitätsverlust des Gelernten bei einer Aufteilung sehr hoch. Die in der Data Mining Vorlesung vorgestellte Holdout Methode geht mit diesem Dilemma so um, das etwa 10% der Daten zum Testen reserviert werden, während die restlichen 90% zum Training verwandt werden. (Schmeier, 2014)). Da uns dann jedoch nur noch 10 Testdatensätze bleiben und auch mit stratisfaction oder der repeated-holdout Methode keine wesentliche Besserung zu erwarten ist, wählen wir den selben Datensatz zum Testen wie zum Lernen. *Annahme: Dies ist bei probabilistischen Netzten nicht so kritisch wie bei 1 Rule oder Regelbäumen, da wir aus der absoluten Häufigkeit eine Relative berechnen, und keine starren Regeln definieren. Was zu dieser Überlegung veranlasst hat war der Versuch eine representative Teilmenge zu entnehmen. Dies hat sich als quasi unmöglich entpuppt, da wir es mit weit mehr als 100 ($(Mathe)7 * (Deutsch)7 * (OLTMathe)10 * ...$) Kombinationsmöglichkeiten zu tun haben. Was bedeutet, dass jeder Datensatz schon jetzt sehr ausschlaggebend sein kann, und sich letztendlich das Netz mit jedem fehlenden Datensatz signifikant verschlechtert.*

Für das Parameterlernen bietet Netica u.a. die Funktion `Cases -> Learn -> Learn using EM` an, die wir genutzt haben.

Um diese Funktion zu Nutzen müssen die Daten als "netica case file" vorliegen. Die Bedingungen für dieses Dateivormat finden sich unter dieser URL: http://www.norsys.com/WebHelp/NETICA/X_Case_File_Format.htm. Da wir die Daten bereits eingelesen

haben für die Diskretisierung muss lediglich die Ausgabe für Netica folgendermaßen angepasst angepasst werden:

```
public boolean writeNetica(ArrayList<Student> datenliste){
  String csv = "Ausgabe.cas";
  BufferedWriter bw = null;
  int counter = 0;

  try{
    bw = new BufferedWriter(new FileWriter(csv));

    bw.write("// ~->[CASE-1]->~");
    bw.write("\n");
    bw.write("\n");

    for(Student student : datenliste){
      if(counter == 0){
        bw.write("IDnum\t");
      }else{
           bw.write(counter+"\t");
      }
      counter++;

      for(String a: student.getAll()){
        bw.write(a + "\t");
      }
      bw.write("\n");
    }
    bw.flush();
    bw.close();

  }catch(IOException e){
    e.printStackTrace();
  }finally {
    if (bw != null) {
      try {
        bw.close();
        System.out.println("Write for Netica done");
        return true;
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
  }
  return false;
}
```

5 Test und Anpassungen

Das Testen der Netze haben wir über die Netica Funktion `Cases -> Test with Cases` durchgeführt. Dabei haben wir insbesondere die „Errorrate“ im Blick. Aus den unter Abschnitt 2 genannten Gründen testen wir auf die beiden Variablen Abschluss und Studiengang.

Erster Test: Beim Test des interaktiv erstellten Netzes mit allen Testdaten erhalten wir eine Errorrate von 41% für den Studiengang und eine von 47% für den Abschluss. Beim test des automatisch generierten Netzes eine von 28% für den studiengang und eine von 47% für den Abschluss.

Interpretation des ersten Tests: Beide Errorraten sind erstaunlich hoch, dafür das wir mit den selben Daten testen, mit denen wir auch das Netzt trainiert haben. Bei dem interaktiv erstellten Netz jedoch für den Studiengang nochmals deutlich schlechter. Für den Abschluss schenken sich die beiden Netze nichts. Betrachten wir den Abschluss genauer so fällt auf, das wir derzeit versuchen vorherzusagen ob das Studium abgebrochen wird, und wenn nicht wie es abgeschlossen wird. Zweites erschwert allerdings die Vorhersage unnötig, da es nicht von maßgeblichem Interesse ist. Deswegen korrigieren wir die Ausprägungen der Variable Abschluss auf $Rg(Abschluss) = \{Abgebrochen, NichtAbgebrochen\}$.

Zweiter Test: Mit den nun korrigierten Netzen, bei denen der Abschluss nur noch zwei Werte annehmen kann, erhalten wir jeweils die Errorrate 8% für den Abschluss. Für den Studiengang ändert sich nichts.

Interpretation des zweiten Tests: Die Errorrate zur vorhersage des Abschlusses konnten wir so wesentlich verbessern. Da die Vorhersage für den Studiengang gleichbleibend schlecht ist versuchen wir über eine Korrektur der Klassen für das Alter weiter zu kommen. Wir haben uns dazu entschieden drei Altersgruppen zu erstellen wobei wir uns an der Menge der Datensätze die in jede Klasse fallen würden orientierten: $Rg(Alter) = \{x_1 = [-\inf, 17], x_1 = [18, 19], x_1 = [20, +\inf]\}$ (Mit diesen Klassen fallen 16% der Datensätze in die erste Klasse 55% in die Zweite und 29% in die Dritte.

Dritter Test: Die beiden neuen Netze haben die selben Errorraten für Studiengang und Abschluss wie bei dem vorherigen Test.

Interpretation des dritten Tests: Dadurch das sich keine Änderungen mehr feststellen ließen, weder Verbesserung noch Verschlechterung, belassen wir es nun damit an den Klassen Änderungen vorzunehmen. Die neue Alterseinteilung behalten wir bei, um mit späterem Testeingaben flexibler umzugehen.

Da in der Aufgabenstellung nicht spezifiziert wird, ob die Klassifizierung beschränkt ist auf eine Klasse oder nicht sehen wir eine Möglichkeit die Vorhersage zu verbessern, indem wir statt einem konkreten Studiengang den wir empfehlen eine Liste ausgeben in dem die Studiengänge nach ihren Ergebnissen im Netz sortiert werden. Wir sehen nun eine Vorhersage als erfolgreich an, wenn sich die tatsächliche Klasse in den ersten beiden Feldern der Liste befindet. (Zur Sortierung siehe Abfragen unter 6)

Vierter Test: Um dieses neue Verständnis zu testen war eine eigene Testfunktion notwendig. In dem folgenden Code ist vor allem Zeile 17 entscheidend. (Testen wir auf einen direkten Treffer so erhalten wir die selben Resultate, wie Netica.)

```
1  //Test every cases and Compare if in first two elemensts of list
2  for(Student student : datenliste){
3  if(student.getQualifikation().equals("Qualifikation")) continue;
4
5  qualifikation.finding().clear();
6  schnitt.finding().clear();
7  [...]
8
9  qualifikation.finding().enterState(student.getQualifikation());
10 schnitt.finding().enterState(translate(student.getSchnitt()));
11 [...]
12 if(!student.getSchultyp().equals("*"))schultyp.finding().enterState(
       student.getSchultyp());
13 if(!student.getStudierfaehigkeitstest().equals("*"))
       studierfaehigkeitstest.finding().enterState(translate(student.
       getStudierfaehigkeitstest()));
14
15 //test for match
16 studiengaenge = studiengang(studiengang);
17 if(studiengaenge[0].equals(student.getStudiengang()) || studiengaenge
       [1].equals(student.getStudiengang()))counter ++;
18
19 }
20 errorrate = 100-((double) counter/ ((double) datenliste.size()-1))
       *100;
21
22 System.out.println("Errorrate: "+ errorrate);
```

In dem wir nun diesen Test anwenden, können wir feststellen, das die Errorrate für das automatisch erstellte Netz bei 11% liegt, also in 89% der fälle die Richtige Klassifikation an erster ODER zweiter Stelle der Liste ist. Für das interaktiv erstellte Netz verbessert sich Die Rate auf 17%.

Für die Abfragen in dem Tool verwenden wir dementsprechend das automatisch erstellte Netz und geben statt eines einzelnen Studiengangs die zwei wahrscheinlichsten Studiengänge aus.

6 Abfragen

Wie bereits unter Abschnitt 2 beschreiben kann die Hochschulqualifikation durch zwei Abfragen im probabilistischen Netz mit $P(Abschluss|...)$ und $P(Studiengang|...)$ erfolgen.

Im folgenden Codeabschintt ist die Abfrage von $P(Abschluss|...)$ gezeigt, mit der Annahme das alle beobachteten Variablen bereits auf das Netz angewendet wurden.

```
1 public static String abgebrochen(Node abschluss) throws NeticaException
      {
2   if(abschluss.getBelief("abgebrochen")>abschluss.getBelief("
        nichtAbgebrochen"){
3       return "abgebrochen";
4   }
5   return "nichtAbgebrochen";
6 }
```

Haben States die selbe Wahrscheinlichkeit, so wird „nichtAbgebrochen“ gewählt.

Bei der Ausgabe des Studiengangs wollen wir nicht behaupten, dass es sich um einen passenden Studiengang für die jeweilige Person handelt, sondern lediglich um die zutreffendsten Studiengänge aus der Liste die durch die Testdaten festgelegt wurde. In der Aufgabenstellung heißt es, dass eine Klassifikation geeignet bestimmt werden soll, in diesem Fall ist die geeignetste Bestimmung eben die Auswahl aus den Studiengängen mit der höchsten Übereinstimmung.

Für die Bestimmung des Studiengangs kommt es mehr auf die Reihenfolge an, als das genau ein Studiengang ausgegeben wird. In dem gezeigten Quellcode haben wir dementsprechend einen einfachen Bubblesort zum sortieren des Arrays nach dem Believe verwendet.

```
1   public static String[] studiengang(Node studiengang) throws
        NeticaException {
2       String[] states = {"Maschinenbau", "SozialeArbeit", "
            Elektrotechnik", "Wirtschaftswissenschaften", "Informatik"};
3     int belive = 100;
4     //compare all belives and sort the array (Bubblesort)
5     String temp;
6     int d=0, k, g=-1;
7     k= states.length-2;
8     while(g<0){
9       g=0;
```

```
10      for(int i=0; i<=k; i++){
11        if(studiengang.getBelief(states[i])<studiengang.getBelief(
              states[(i+1)])){
12          temp=states[i];
13          states[i]=states[(i+1)];
14          states[(i+1)]=temp;
15          g--;
16        }
17      }
18      k--;
19    }
20    return states;//Da gleiche Wahrscheinlichkeiten eine Rolle spielen,
          geben wir ein Sortiertes Array zurueck
21  }
```

Zuletzt sellen wir noch kurz auf das Interface für das Tool vor.

7 Interface und Kurzanleitung

Das Tool mit dem Anfragen an das Netz gestellt werden können ist als Commandline Programm umgesetzt. Es besteht aus einer ausführbaren .jar die als Eingabe eine Datei im Format der Beispieldatei aus der Aufgabenstellung akzeptiert. Die Datei mit den Eingabedaten muss dabei die Anzahl und Reihenfolge der Attribute aus der Beispieldatei übernehmen.

Beim Programmaufruf kann mit dem Parameter -test angegeben werden, dass die Daten zur Ermittlung der Errorrate für die Variable Studiengang mir unserer eigenen Testfunktion genutz werden. Ein Beispiel für einen Kommandozeilenaufruf wäre wie folgt:
`java -Djava.library.path=/YOUR/PATH/TO/NETICA/NeticaJ_504/bin -jar Hochschulqualifikation.jar net_file.dne source_file.csv` (Für der Parameter -test wird folgendermaßen mitgegeben:
`java -Djava.library.path=/YOUR/PATH/TO/NETICA/NeticaJ_504/bin -jar Hochschulqualifikation.jar -test net_file.dne source_file.csv`)

Achtung: Um eine fehlerfreie Bearbeitung zu garantieren sind nur Eingabedateien zu verwenden welche in **UTF-8** kodiert sind. Des weitere werden nur die Sonderzeichen ä Ä ö Ö ü Ü und ß unterstützt. Nicht angegebene Werte in der Eingabedatei müssen mit **„n.a.“** gekennzeichnet werden.

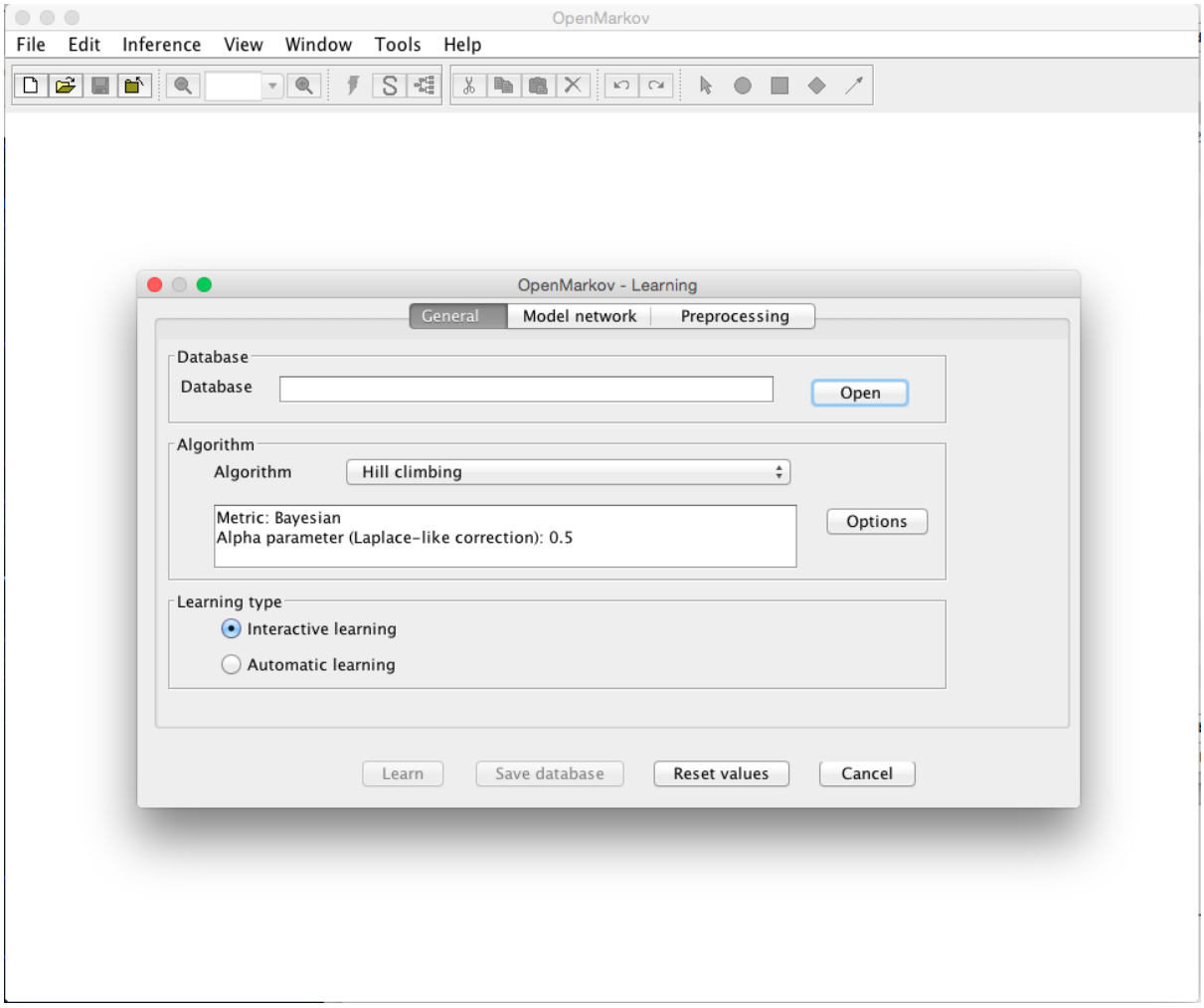
References

rer. nat. Bernd Klöss, . D. (2014). Probability theory and statistics for computer scientists. Script of the lecture held at the Cooperative State University Stuttgart, Baden-Wuerttemberg.

Schmeier, S. (2014). Data mining – evaluation. Script of the lecture held at the Cooperative State University Stuttgart, Baden-Wuerttemberg.

Appendix

A Openmarkov



B Interaktives Strukturlernen

Reihenfolge des Hinzufügens der Links.

Link	**Motivation**
Mathe –> Physik	81.03
Mathe –> OLT-Mathe	49.20
Mathe –> Abschluss	45.37
Schultyp –> Qualifikation	38.63
Schultyp –> Deutsch	38.81
Schultyp –> Mathe	38.36
Deutsch –> OLT-Deutsch	28.62
Qualifikation –> Studierfähigkeitstest	29.03
Qualifikation –> Alter	16.49
Geschlecht –> Studiengang	8.92
Schultyp –> Jahreseinkommen der Eltern	4.36
Staatsbürgerschaft –> Bundesland	3.43
Staatsbürgerschaft –> Schnitt	2.94
Bundesland –> Jahreseinkommen der Eltern	1.07
Deutsch –> Jahreseinkommen der Eltern	6.0
Invert Schultyp –> Qualifikation	3.3
Schultyp –> OLT-Mathe	3.07
Staatsbürgerschaft –> Studiengang	0.00
OLT-Deutsch –> Studiengang	0.81

Vom Tool noch angebotenen Links welche wir haben wegfallen lassen:

Link	**Motivation**
Studierfähigkeitstest –> Alter	1.1
Qualifikation –> OLT-Deutsch	0.6
OLT-Deutsch –> Schnitt	0.07
Invert Geschlecht –> Studiengang	1.87