Fakultät für Informatik
Übung zu Kognitive Systeme
Sommersemester 2017

M. Sperber (matthias.sperber@kit.edu)
S. Nguyen (thai.nguyen@kit.edu)

Übungsblatt 3

Maschinelles Lernen und Spracherkennung

Abgabe online vor 11:30 Uhr, 21. Juni 2017
Abgabe der schriftlichen Ausarbeitung vor der Übung am 21. Juni 2017

Bonuspunkte für die Klausur werden nur vergeben,
wenn eine schriftliche Lösung aller Aufgaben (incl. Onlinefragen) abgegeben wurde.

Die Antworten auf die Onlinefragen müssen unter
http://his.anthropomatik.kit.edu/Teaching/VorlesungKognitiveSysteme/websubmit/
eingegeben werden.

Aufgabe 1: *Neuronales Netz*

Ein Autoencoder ist ein künstliches neuronales Netzwerk, welches eine Repräsentation (ein Encoding) eines Datensatzes lernt, typischerweise mit dem Ziel der Dimensionen-Reduktion. Meist besteht ein Autoencoder aus einem Multilayer-Perzeptron (MLP) mit gleicher Zahl von Eingabe- und Ausgabe-Knoten, und einer versteckten Schicht von geringerer Dimension. Dieses Netz wird dann so trainiert, dass es die Eingabe wieder möglichst genau rekonstruiert. Mit anderen Worten, im Training ist der erwünschte Ausgabe-Vektor gleich dem Eingabevektor.

Onlinefrage Nr. 1: Welche Eigenschaften gelten für einen Autoencoder?

i) unüberwacht

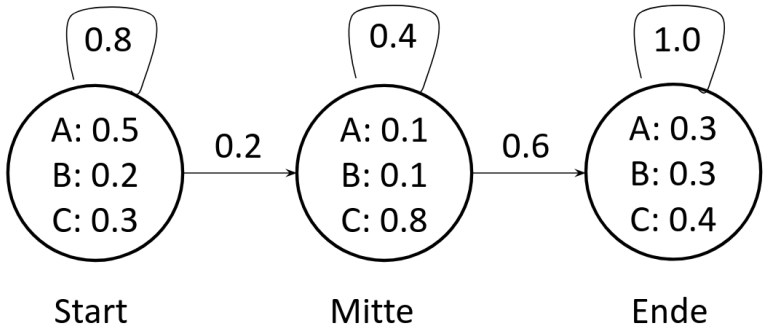
ii) überwacht

iii) nicht-parametrisch

iv) parametrisch

Aufgabe 2: *HMM - Forward- / Viterbi-Algorithmus*

Gegeben sei das Modell λ wie folgt:



Hinweis: Für diese Aufgabe nehmen wir an dass im Start-Zustand begonnen, im Ende-Zustand terminiert und im Start-Zustand q0 noch kein Symbol emittiert wird (d.h., es kommt immer zuerst ein Zustandsübergang, und erst danach eine Beobachtung).

a) Berechnen Sie die Wahrscheinlichkeit der Zeichenketten $O_1 = ABC$ und $O_2 = CBB$ und $O_3 = CCC$ mit dem Forward-Algorithmus.

b) Welche Zustandskette $Q = (q_0 q_1 q_2 q_3)$ liefert die maximale Wahrscheinlichkeit $P(O, Q|\lambda)$ für die Ausgabe O_1 mit $q_0 = Start$ wobei im Startzustand q_0 noch kein Symbol emittiert wird?

   Hinweis: Wenden Sie hier den Viterbi-Algorithmus an.

c) **Onlinefrage Nr. 2:** Welche Zeichenkette aus Teilaufgabe a) ist wahrscheinlicher?

   i) O_1
   ii) O_2
   iii) O_3
   iv) O_1 und O_2
   v) O_2 und O_3

Aufgabe 3: *Sprachmodelle*

Ein 3-gram Sprachmodell einer formalen Sprache mit dem Vokabular
$V = \{wait, and, Tea, drink\}$ enthalte die folgenden Wahrscheinlichkeiten.

w_{i-2}	w_{i-1}	w_i = *wait*	w_i = *and*	w_i = *Tea*	w_i = *drink*	$w_i =<$ */S* $>$
n/a	<S>	0,5	0,25	0,1	0,1	0,05
<S>	wait	0,04	0,6	0,05	0,05	0,26
<S>	and	0,4	0,05	0,4	0,05	0,1
<S>	Tea	0,04	0,6	0,05	0,05	0,26
<S>	drink	0,04	0,5	0,25	0,05	0,16
wait	wait	0,2	0,2	0,25	0,2	0,15
wait	and	0,04	0,1	0,5	0,3	0,06
wait	Tea	0,1	0,5	0,2	0,1	0,1
wait	drink	0,04	0,35	0,4	0,1	0,11
and	wait	0,04	0,05	0,3	0,25	0,36
and	and	0,2	0,2	0,25	0,2	0,15
and	Tea	0,1	0,1	0,1	0,5	0,2
and	drink	0,04	0,05	0,6	0,1	0,21
Tea	wait	0,1	0,6	0,1	0,1	0,1
Tea	and	0,2	0,1	0,1	0,25	0,35
Tea	Tea	0,1	0,1	0,1	0,1	0,6
Tea	drink	0,2	0,2	0,2	0,25	0,15
drink	wait	0,1	0,3	0,1	0,1	0,4
drink	and	0,4	0,05	0,05	0,05	0,45
drink	Tea	0,1	0,3	0,1	0,1	0,4
drink	drink	0,2	0,2	0,25	0,2	0,15

Hierbei steht das Symbol <S> für den Anfang des Satzes und </S> für das Satzende. Berechnen Sie die Wahrscheinlichkeiten der folgenden Sätze und die dazugehörigen Perplexitäten des Sprachmodells.

Hinweis:
Als „Anzahl der Wörter“ bei der Berechnung der Perplexität in der Spracherkennung wird oft die Anzahl der Wortübergänge benutzt (bei n-Grammen mit $n > 1$), da diese letztendlich entscheidend für die Wahrscheinlichkeit der Wortsequenz sind. Benutzen Sie zur Berechnung der Perplexität die aus den Vorlesungsfolien bekannten Formeln für Perplexität und verwenden Sie dazu die Anzahl der Wortübergänge als n in der Berechnung. <S> und </S> werden dabei als Wörter angesehen. Zur Erinnerung, normalisierte Logprob: $H(W) = -\frac{1}{N}\sum_{i=1}^{N} \log_2 P(w_i|\Psi(w_1, \ldots, w_{i-1}))$, und Perplexität: $\text{PPL}(W) = 2^{H(W)}$.

a) „ Tea “

b) „ drink Tea “

c) „ Tea wait and drink “

d) „ Tea drink and wait “

e) **Onlinefrage Nr. 3:** Geben Sie die Sätze a)-d) nach aufsteigender Perplexität sortiert an.

i) $abcd$
ii) $bcda$
iii) $acbd$
iv) $bcad$
v) $cdab$

Aufgabe 4: ***Programmieraufgabe:*** *Dynamische Programmierung*

Gegeben sind die korrekte Sequenz (Referenz) „ if there is no rain in April you will have a great summer“ und fünf Hypothesen:

1) no rain in april then great summer come
2) there is rain in April you have summer
3) in April no rain you have summer great
4) there is no rain in apple a great summer comes
5) you have a great summer comes if there is no rain in April

a) Schreiben Sie ein Programm, um die gegebenen Hypothesen mit der korrekten Sequenz auf Wortebene (englisch *word edit distance*) zu vergleichen. Implementieren Sie dabei eine Zurückverfolgung des Pfades und vergeben Sie Strafpunkte, um dadurch die ähnlichste Hypothese auszuwählen. Dabei sollen übereinstimmende Wörter 0 Strafpunkte bekommen, und substitutions *(Nichübereinstimmungen)*, insertions *(Einfügungen)*, und deletions *(Auslassungen)* jeweils 1 Strafpunkt. Achten Sie darauf, dass ihr Algorithmus *case-sensitive* arbeitet, d.h. “Summer” und “summer” als 2 unterschiedliche Wörter betrachtet. Welche der 5 Hypothesen ist der Referenz am ähnlichsten?

b) Modifizieren Sie Ihr Programm, so dass die Ähnlichkeit auf Buchstabenebene bestimmt wird (*character edit distance*). Dazu soll jeder Buchstabe inklusive Leerzeichen als eigenes “Wort” behandelt werden. Welche der 5 Hypothesen ist nun der Referenz am ähnlichsten?

c) **Onlinefrage Nr. 4:** Nun sei die Anzahl der Strafpunkte für substitutions *(Ersetzungen)* 2 (statt 1), die anderen Strafpunkte bleiben wie zuvor. Welche Hypothese ist nun der Referenz am ähnlichsten (gemessen in Editierdistanz auf Wortebene), und was ist die entsprechende Editierdistanz?

   i) 4, 6
   ii) 2, 5
   iii) 3, 8
   iv) 1, 6
   v) 2, 6
   vi) 3, 5

   Hinweis:
   Die Onlinefrage Nr. 4 kann auch durch manuelles Berechnen der Punktzahlen beantwortet werden, jedoch dürfte eine Implementierung schneller und sicherer sein, da man sich bei diesem Algorithmus leicht verrechnen kann.