3 Vertiefung: Drei wichtige Algorithmen – Teil 3

Soweit vorhanden ist der jeweils englische Fachbegriff, so wie er in der Fachliteratur verwendet wird, in Klammern angegeben.

Beispiele von Computerkommandos und Skizzenanweisungen sind in `Schreibmaschinenschrift` wiedergegeben.

3.3 Künstliche neuronale Netze (ANN)

Im Prinzip können ANN für alle nur denkbaren Aufgaben der Mustererkennung und damit auch der Spracherkennung eingesetzt werden. Es ergeben sich die gleichen Probleme wie bei der statistischen Mustererkennung, nämlich die variierende zeitliche Dynamik der Sprache. Daher werden ANN oftmals nur als Bestandteil von 'klassischen' Verfahren wie HMM verwendet, z.B. zur Berechnung der 'Emissionswahrscheinlichkeiten' (WDF) in den Zuständen der HMM.
Man nennt solche Systeme dann auch 'hybride' Systeme.

`Skizze:`

`Beispiel eines hybriden HMM-ANN Systems`

3.3.1 Netztopologien, Synapsen und Neurone

Es werden eine Vielzahl verschiedener ANN-Strukturen (Topologien) in der ASR eingesetzt. Man unterscheidet u.a. zwischen 'feed-forward' (Synapsen verknüpfen alle in einer Richtung) und 'recursive' (Synapsen können wieder zurück auf Eingangsneurone verknüpfen), zwischen statischen (keine zeitliche Dynamik innerhalb des ANN) und 'time-delay neuronal networks' (TDNN) (die Weitergabe von Werten über die Synapsen und die Verarbeitung innerhalb der Neurone braucht

Zeit). Der 'Klassiker' in der ASR ist das einfache 'perceptron', d.h. ein vielschichtiges (multi-layered) 'feed-forward' Netzwerk mit einer Eingangsschicht, einer Ausgangsschicht und 1 bis 2 hidden layers.

`Skizze:`

`Beispiel: einfaches Perzeptron mit einer verborgenen Schicht`

Synapsen, d.h. Verbindungen von einem Neuron zum nächsten sind uni-directional (Werte werden nur in einer Richtung weitergegeben) und tragen jeweils einen Gewichtsfaktor (weight), der auf den Ausgangswert der sendenden Neurons multipliziert wird, und das Ergebnis wird dann an den Eingang des Zielneurons weitergegeben.
Das Neuron selber ist nur ein einfacher Summierer aller Eingänge mit einer festgelegten nachgeschalteten Kennlinie (wird also beim Trainieren eines ANN überhaupt nicht verändert).

`Skizze:`

`Synapse mit Gewicht und Neuron mit Addierer und Kennlinie`

Die nicht-lineare Kennlinie im Neuron dient dazu, den Wertebereich innerhalb des ANN auf einen definierten Bereich zu begrenzen. Sonst kann es leicht zum Aufschaukeln der weitergegebenen Werte kommen (vor allem bei mehrschichtigen ANN) und das Netz kollabiert (Analogie: Epilepsie beim Menschen).

3.3.2 Anwendung in der ASR

In der ASR dienen ANN meistens (nicht immer) dazu, Merkmalsvektoren zu bewerten. D.h. die Aufgabe des ANN ist es, einen gegebenen Merkmalsvektor einer endlichen Menge von Kategorien zuzuordnen.

Beispiel:
Ein einfaches Perzeptron mit einer verborgenen Schicht habe 39 Eingangsneurone (für jede Komponente des Merkmalsvektors ein Eingang), 10 Neurone in der verborgenen Schicht und 44 Neurone in der Ausgangsschicht (entsprechend der Anzahl der Phoneme-Klassen). Wird nun am Eingang ein Merkmalsvektor aus einem /a:/ angelegt, dann sollte der Ausgang des ANN mit dem Label /a:/ den höchsten Ausgangswert aufweisen ('am stärksten feuern').

```
Skizze:

Einfaches Perzeptron zur Bewertung von Merkmalsvektoren
```

Mit diesem ANN kann man aber noch keine Sprache erkennen, weil es die Dynamik der Sprache nicht modelliert; es ist eher vergleichbar dem einfachen GMM aus Abschnitt 'Statistische Modelle / Einfache single-state Modelle'.

3.3.3 Training eines ANN

Trainingsalgorithmen für ANN sind stark von der gewählten Topologie anhängig. Für 'feed-forward' ANN ist der 'backpropagation' Algorithmus weit verbreitet:
Zunächst werden alle Gewichte (Synapsen) im ANN mit zufälligen Werten (nicht 0!) vorbesetzt.
Dann werden alle gelabelten Trainingsdaten (Merkmalsvektoren) nacheinander an den Eingang des ANN gelegt und die Ausgangswerte berechnet. Wie nicht anders zu erwarten, sind diese völlig chaotisch.
Die Ausgangswerte werden nun mit einer Zielfunktion (target function) verglichen, welche alle Ausgänge mit 0 belegt außer dem Ausgang, der das Label des angelegten Merkmalsvektors trägt, welche zu 1 gesetzt wird. Aus der Differenz von Zielfunktion (ideales Ergebnis) und den tatsächlichen Ausgangswerten wird eine Fehlerfunktion berechnet, und diese längs der Synapsen in das ANN 'zurückgeschickt' (daher 'backpropagation'). Trifft so ein Fehlerwert auf eine Synapse, so wird das Gewicht dieser Synapse um einen winzigen Betrag (learn rate) so verändert, dass die Fehlerfunktion am Ausgang des ANN ein wenig geringer wird (Fehlerminimierung). Das geschieht zunächst in den Synapsen zwischen vorletzter Schicht und Ausgangsschicht, und dann, basierend

auf den so korrigierten Werten an der verborgenen Schicht über alle vorhandenen Schichten bis hin zur Eingangsschicht.
Mit anderen Worten: Das ANN bekommt für jeden Merkmalsvektor am Eingang eine Vorgabe, wie es eigentlich (idealerweise) reagieren sollte. Da es das nicht kann, werden die Gewichte innerhalb des ANN so verändert, dass zumindest der Fehler am Ausgang etwas geringer wird.
Dann wiederholt man den ganzen Vorgang (epoch=Lernschritt), bis das ANN konvergiert, also keine Verbesserung in der Fehlerfunktion mehr zeigt.

```
Skizze:

Training mit Backpropagation
```

3.3.4 Test eines Merkmalsvektors mit ANN

Alle Merkmalsvektoren der unbekannten Äusserung an den Eingangsneuronen 'vorbeiziehen'. Die Werte der Ausgangsneurone in eine Entscheidungsfunktion geben (z.B. Mehrheitsentscheidung)

```
Skizze:

Test eines Merkmalsvektors mit ANN
```

Es lässt sich zeigen (Bourlard&Morgan, 1994), dass unter Einhaltung bestimmter Randbedingungen die Ausgangswerte eines ANN (bis auf einen Normierungsfaktor $p(\vec{x})$) der bedingten Wahrscheinlichkeit $p(L|\vec{x})$ entsprechen. Diese entspricht zwar genau dem Term, der in der statistischen

Spracherkennung maximiert werden soll (vgl. Kapitel 'Bayes' in Abschnitt 2), jedoch bezieht sich hier das modellierte Ereignis L genaugenommen nicht auf ein Wort (oder einen Laut) sondern nur auf die Zugehörigkeit eines einzigen Merkmalvektors (frames) zu einem Laut L ohne Berücksichtigung der Dynamik. Um also das ANN wie oben geschildert in einem hybriden Modell, also eingebettet in einem HMM zu verwenden, brauchen wir den umgekehrten Term $p(\vec{x}|L)$, welcher dann genau wie das Ergebnis eines statistischen Modells (der WDF) mit den gleichen Algorithmen der Hidden Markov Modelle weiterverarbeitet werden kann. Aus diesem Grunde wird der Output des ANN $p(L|\vec{x})$ mit $P(L)$ normiert, denn nach Bayes ist

$$max_L p(\vec{x}|L) = max_L \frac{p(L|\vec{x})p(\vec{x})}{P(L)}$$

wobei $p(\vec{x})$ wie üblich vernachlässigt werden kann, weil $\vec{x}$ nicht von L abhängig ist. $P(L)$ lässt sich ganz einfach durch Auszählen der Phonem-Häufigkeit im Trainingskorpus ermitteln:

$$P(L) = \frac{N(L)}{N_{ges}}$$

mit:
$N(L)$: Anzahl der Phoneme der Klasse L
N_{ges} : Anzahl aller Phoneme im Korpus

3.3.5 Eigenschaften der ANN in der ASR

Der wichtigste Unterschied zu anderen Verfahren ist die Tatsache, dass ein ANN *diskriminativ* vorgeht, d.h. es gibt nicht zu jeder Klasse ein Modell (HMM) oder Muster (DTW), welche unabhängig voneinander mit dem unbekannten Muster verglichen werden, sondern ein ANN 'sieht' beim Training immer Daten aller Klassen gleichzeitig, lernt also, diese Klassen zu unterscheiden.

- sprecherunabhängig
- diskriminativ
- viel Trainingsmaterial notwending
- aufwändiges Training; Training kann in einem 'lokalen Optimum' enden
- Problem der Überadaption an das Trainingsmaterial

Bemerkung:
Diese Eigenschaften beziehen sich auf das hier skizzierte einfache Perzeptron. Außer diesem gibt es noch viele andere Varianten von ANN; diese haben dann geg.falls auch andere Eigenschaften.