Automatische Spracherkennung

3 Vertiefung: Drei wichtige Algorithmen – Teil 1

Der folgende Abschnitt skizziert die grundlegenden Prinzipien dreier Methoden zur Mustererkennung.

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.

3.1 Klassische Mustererkennung mit DTW

Die einfachste Methode der Spracherkennung besteht aus einem direkten Vergleich von abgespeicherten Sprachmustern von Trainingssprechern mit dem unbekannten Signal.

Generelle Probleme:

- Zeitsignale können direkt nur sehr schwer miteinander verglichen werden
  Lösung: Merkmalsextraktion, Merkmalsvektorfolgen.
- Unterschiedliche Dynamik/Längen
  Lösung: Durch Verwendung von Dynamic Time Warping (DTW) werden Muster gestaucht/gedehnt und somit optimal aufeinander angepasst
- Abgespeicherte Sprachmuster generalisieren nicht
  Lösung: möglichst viele Sprachmuster von verschiedenen Sprechern
  Nachteil: Aufwand steigt linear mit Anzahl der Muster

DTW Training

Von jeder zu erkennenden/zu bewertenden Sprachkategorie (z.B. Wort) werden eine oder mehrere Varianten aufgezeichnet, die Merkmale extrahiert und die Merkmalsfolgen als Muster abgespeichert und mit dem Label der Kategorie versehen.

`Skizze: Abspeichern von Mustern für DTW`

DTW Test (Erkennung)

Die zu erkennende Merkmalsvektorfolge wird mit DTW nacheinander mit *allen* gespeicherten Muster verglichen. Dabei werden sog. Abstandsmaße für den quantitativen Vergleich zweier Merkmals*vektoren* verwendet.

Die Wahl des Abstandmaßes kann das Ergebnis der DTW entscheidend beeinflussen. Die wichtigsten Abstandsmaße (in steigender Komplexität) sind:

- City-Block : Summe über Differenzen der einzelnen Merkmale
- Euklid : 'Echter' Abstand zweier Vektoren im Raum
- Mahalanobis : für den Vergleich mit dem statistischen Mittelwertsvektor einer ganzen Stichprobe von Vektoren

`Skizze: Abstandsmaße Cityblock, Euklid, Mahalanobis`

Der Gesamtabstand zweier *Merkmalsvektorfolgen* wird definiert als die Summe aller paarweisen Abstände zweier Vektoren; die Paarung wird aus einer optimalen Zuordnung der Vektoren des unbekannten Sprachmusters auf das gespeicherte Muster ermittelt. 'Optimal' heißt hier, dass die Zuordnung der Vektoren zu einem minimalen Gesamtabstand (= Summe aller Einzelabstände) führen soll.

Das Muster mit dem minimalen Gesamtabstand wird als das Erkannte ausgewählt (Entscheidungsregel: Min-Entscheidung).

Skizze: Blockschaltbild Erkennung mit Dynamic Time Warping

DTW Eigenschaften

- Sprecherabhängig (es sei denn, man nimmt sehr viele Sprecher auf)
- Muster generalisieren nicht
- Rechenaufwand (bei der Erkennung) steigt linear mit Trainingsmaterial an
- einfacher Algorithmus
- relativ robust
- nur für Ganzworterkennung geeignet[1]
- falls für einen statistischen Ansatz notwendig, können Wks $p(s|W)$ aus dem Abstandsmaßen berechnet werden

Dynamic Time Warping (DTW), Dynamische Programmierung (dynamic programming, DP)

Die DTW findet die nicht-lineare Zuordnung von einer Reihe von Merkmalsvektoren eines unbekannten Musters auf die Reihe von Merkmalsvektoren eines bekannten Musters. Dadurch können zeitliche Verzerrungen, z.B. bei länger oder kürzer ausgesprochenen Silben, ausgeglichen werden und somit Muster mit unterschiedlicher Dynamik dennoch objektiv mit einem abgespeicherten Muster verglichen werden. Der Gesamtabstand zweier Muster ist definiert als die Summe aller Abstände aufeinander abgebildeter Merkmalsvektoren.

Skizze: Einfaches Beispiel mit zwei kurzen Mustern; mögliche Pfade

Wie man unschwer erkennen kann, ergeben sich sehr viele mögliche Abbildungen der einzelnen Merkmalsvektoren. Auch wenn der mögliche 'Pfad' insofern eingeschränkt wird, daß keine 'Rücksprünge stattfinden dürfen, ist doch die Kombinatorik so groß, dass es nicht möglich ist, alle möglichen Pfade separat zu berechnen und zu vergleichen.

[1] Es gab auch Ansätze für sog. mehrstufige DTW-Erkennung, bei welcher auf der ersten Ebene Phoneme als Kategorien verwendet und auf der zweiten Ebene diese zu Wörtern kombiniert wurden. Aber diese Ansätze haben sich nicht durchgesetzt, weil die Performanz schlechter ist als bei HMM.

Daher geht man anders vor: Für *jeden* Punkt der von den beiden Mustern aufgespannten Abstandsmatrix wird zunächst der **lokale** Abstand der beiden zugeordneten Merkmalsvektoren berechnet. Sodann berechnet man rekursiv für jeden Punkt der Abstandsmatrix, der überhaupt von potentiellen Pfaden durchlaufen werden kann, das minimale Abstandsmaß, das Pfade bis zu diesem Punkt minimal haben müssen. Dies kann sehr einfach dadurch geschehen, indem man von dem betreffenden Punkt *zurück* auf alle Punkte schaut, von welchen ein Pfad den Punkt erreichen kann, deren minimale Abstandsmaße liest, den lokalen Abstand dazu addiert und schließlich nur das minimale Ergebniss aller möglichen auswählt.

Beispiel anhand obiger Skizze

Geht man nach dieser Methode vor, so ist immer garantiert, dass man für jeden Punkt den optimalen Pfad bis dorthin bereits weiß und abspeichern kann. Diese Methode wird - wie gesagt - für *alle* Punkte der Abstandsmatrix von links nach rechts und von unten nach oben durchgeführt. Ist man in der letzten Spalte angekommen, so ergibt sich automatisch, dass der Pfad, der für den Wert rechts oben berechnet wird, das optimale Gesamtabstandsmaß enthalten muss.

Ist man daran interessiert, wie die nicht-lineare Zuordnung der Merkmalsvektoren aussieht, so kann man diese durch Zurückverfolgen der erfolgten Min-Entscheidungen (sog. 'backtracking') ermitteln. Dazu muss aber vorher zusätzlich in jedem Punkt der Abstandsmatrix ein Zeiger eingetragen werden, der angibt, aus welcher Richtung der minimale Pfad für diesen Punkt gekommen ist.

Beispiel anhand obiger Skizze

Dieses Verfahren garantiert das Auffinden der Abbildung, bei welcher der Gesamtabstand der beiden verglichenen Muster minimal wird. Der Aufwand steigt quadratisch mit der Länge der beiden verglichenen Muster; im Gegensatz dazu würde der Aufwand bei der expliziten Berechnung aller möglichen Abbildungen exponentiell ansteigen.

DTW - Betrachtung des Suchproblems als Gebirge über der Lattice

Warum eine einfache lokale Suchstrategie nicht funktioniert, lässt sich mit einem Gedankenexperiment deutlich machen. Man denke sich über der aufgespannten Abstandsmatrix ein Gebirge. Die Höhe dieses Gebirges ist direkt proportional zu den lokalen Vektorabstand der darunter liegenden Zuordnungen. An der einen Ecke des Gebirges befindet sich unser Zugang, diametral gegenüber der Ausgang. Das Suchproblem lässt sich dann so formulieren:

> 'Finde den Weg über das Gebirge, bei dem die Summe aller Höhenpunkte, die du auf deinem Weg passierst, das absolute Minimum darstellt!'

Eine *lokale Suchstrategie* entspricht einem Wesen, das immer nur die unmittelbare Umgebung kennt und bei jedem neuen Schritt ein für alle mal entscheiden muss, in welche Richtung der beste Pfad weitergehen wird. Solche Verfahren heißen z.B. Gradientenanstiegsverfahren und folgen immer der gerade niedrigsten Steigung. Natürlich kann so eine Strategie auch den optimalen Pfad finden; das ist aber eher ein Glücksfall. Viel wahrscheinlicher ist, dass sie einem zunächst vielversprechenden 'Tal' im Gebirge folgt, welches dann in einer 'Sackgasse' endet.

Eine *globale Suchstrategie* sucht sich ihren optimalen Pfad über das Gebirge vom Flugzeug aus, d.h. unter Berücksichtigung aller Höhenpunkte gleichzeitig. Tatsächlich lässt sich zeigen, dass auch eine globale Strategie nicht alle möglichen Pfade berechnen muss, um den optimalen Pfad zu finden (Redundanzausnutzung gemeinsamer Pfade!). Sie muss aber zumindest jeden Höhenpunkt, der potentiell durchlaufen werden kann, einmal unter die Lupe nehmen und sein Verhältnis zu seiner unmittelbaren Umgebung registrieren. Vom Ausgangspunkt des Gebirges lässt sich dann durch sogenanntes 'backtracking' der optimale Pfad zurückverfolgen.