EIN DYNAMIC-PROGRAMMING-ALGORITHMUS ZUR ANWENDUNG IN DER AUTOMATISCHEN SPRACHVERARBEITUNG

FRANK W. A CAMPO

Ruhr-Universität Bochum
Sprachwissenschaftliches Institut
4630 Bochum, P.O.B. 10 21 48, BRD

ABSTRACT

Eine Variante des Dynamic-Programming-Algorithmus' von Sakoe und Chiba wird experimentell untersucht. Es zeigt sich, daß durch sie die Rechenzeit bei der Spracherkennung erheblich gesenkt werden kann.

EINLEITUNG

Gegeben seien die Sprachsignale A und B. Zu gewissen Zeitpunkten sollen von ihnen Meßdaten erhoben werden, die nach Aufbereitung als Vektoren vorliegen. Der Abstand zwischen dem i-ten Meßdatenvektor von A und dem j-ten Meßdatenvektor von B sei durch eine Metrik d gegeben und werde mit d(i,j) bezeichnet. Die Menge der Paare (i,j) von auf A bzw. B definierten Meßdatenvektoren sei mit Ω bezeichnet. Fassen wir Ω als Menge von Punkten in der Ebene auf, bildet Ω ein Rechteck, und jeder Weg über Punkte aus diesem Rechteck läßt sich als Transformation der Zeitachsen von A und B auffassen. Für einen endlichen Weg in Ω läßt sich der mittlere gewichtete Abstand der Punkte (i,j), über die er führt, als Abstand zwischen A und B unter dieser Zeitachsentransformation interpretieren. Ein Durchbruch in der automatischen Spracherkennung bestand darin, daß für gewisse Mengen von Wegen ein rekursiver Optimierungsalgorithmus ("Dynamic Programming", "DP") gefunden wurde, der den minimalen Abstand zwischen den Sprachsignalen bezüglich dieser Mengen ermittelte. In diesem Beitrag möchte ich die von mir in [1] entwikkelte zweite Variante des schon als klassisch zu bezeichnenden DP-Algorithmus' von Sakoe und Chiba [2] untersuchen. Die gemeinsame Theorie der Algorithmen habe ich in [1] dargelegt; dort findet sich auch die Behandlung vieler Fragen, die ich hier nur berühren kann.

ALGORITHMEN

Wir definieren die Mengen von Wegen in Ω mittels Eigenschaften der in ihnen enthaltenen Wege. Diese Eigenschaften müssen einer theoretischen und einer praktischen Anforderung genügen: der praktischen, daß das Minimum bezüglich der durch sie bestimmten Wegmenge sich in annehmbarer Zeit ermitteln läßt, und der theoretischen, daß sich die Eigenschaften der Wege als physikalisch sinnvolle Eigenschaften von Zeitachsentransformationen deuten lassen. Folgende gängige Wegeigenschaften (i. f. We) sollen hier betrachtet werden:

We1) Jeder Weg beginnt mit (1,1) und endet mit (I,J), wobei I bzw. J die Anzahl der auf A bzw. B definierten Meßpunkte sei.

We2) Führt ein Weg hintereinander über die Punkte (i_1,j_1), (i_2,j_2), muß gelten:
- i) $(i_1,j_1) \neq (i_2,j_2)$;
- ii) $0 \leq i_2 - i_1 \leq 1$;
- iii) $0 \leq j_2 - j_1 \leq 1$.

We3) Für ein fest gewähltes $n \in \mathbb{N}$ darf ein Weg höchstens zwischen n+1 aufeinanderfolgenden Punkten parallel zu einer Rechteckseite verlaufen.

We4) Kein Weg darf rechtwinklig abknicken.

We1 und We2 bewirken, daß die Reihenfolge der Meßpunkte unter Zeitachsentransformation erhalten bleibt und keiner der Meßdatenvektoren vernachlässigt wird.
We3 beinhaltet, daß ein einem Meßdatenvektor zugeordneter Signalabschnitt im zeitachsentransformierten Signal um höchstens das (n+1)-fache verlangsamt wird. Durch diese Wegbedingung wird die Menge der Punkte aus Ω, über die ein Weg führen kann, eingeschränkt: Alle Punkte liegen in einem Parallelogramm bzw. auf dessen Rand, wobei die Seiten des Parallelogrammes die Steigungen n+1 bzw. 1/(n+1) haben und seine spitzen Ecken auf (0,0) und (I+1,J+1) liegen. Die Menge dieser Punkte sei mit P_n bezeichnet und als nicht leer angenommen. Dann gibt es für jedes i mit $1 \leq i \leq I$ ein minimales j und ein maximales j' so, daß (i,j) und (i,j') Elemente von P_n sind. Sie seien mit jmin(i) und jmax(i) bezeichnet.
Für We4 eine physikalische Begründung zu finden, war mir nicht möglich und ist meines Wissens bisher auch nicht versucht worden. We4 wurde schon von Sakoe und Chiba einzig unter praktischen Gesichtspunkten eingeführt: "This new constraint reduces the number of paths to be searched" [2]. Die Menge der Punkte aus Ω, über die ein Weg führen kann, ist bis auf höchstens zwei Punkte mit P_n identisch (s. [1], Kapitel 4); die häufig zu lesende Behauptung, die Ecken des Parallelogrammes lägen auf den Punkten (1,1) und (I,J), ist falsch.

Der DP-Algorithmus zur Ermittlung des Minimums bezüglich der durch We1 und We2 definierten Wegmenge ist gut bekannt. Er sei im folgenden V-Algorithmus genannt. Der Algorithmus für die durch We1, We2 und We3 definierte Wegmenge ist der zweite der von mir in [1] entwickelten. Er ist es, dem hier unser besonderes Interesse gilt. Ich bezeichne ihn als F_n-Algorithmus. Den Algorithmus für die durch We1 bis We4 bestimmte Wegmenge nenne ich S_n-Algorithmus. Mit einigen hier belanglosen Abweichungen handelt es sich um den DP-Algorithmus von Sakoe und Chiba.
Im folgenden mache ich von einer bewährten Gewichtsfunktion Gebrauch: d(i,j) wird mit dem Faktor 2 gewichtet, wenn (i,j) in einem Weg diagonal erreicht wird, sonst mit dem Faktor 1. Die Minima lassen sich auf folgende in Fortran-naher Notation dargestellte Arten berechnen:

V-Algorithmus (We1 und We2):

```
    ko(i,j) = ∞ f.a. (i,j) ∉ Ω
    ko(0,0) = 0
    DO 20 i=1,I
    DO 10 j=1,J
    x = d(i,j)
10  ko(i,j) = x + min { ko(i-1,j), ko(i-1,j-1) + x,
                        ko(i,j-1) }
20  CONTINUE
    Minimum = ko(I,J) / (I + J)
```

F_n-Algorithmus (We1, We2 und We3):

```
    ko(i,j) = ∞ f.a. (i,j) ∉ P_n
    ko(0,0) = 0
    g_μ(j) = ∞ f.a. 0 ≤ μ ≤ n und 1 ≤ j ≤ J
    DO 20 i=1,I
    f_μ = ∞ f.a. 0 ≤ μ ≤ n
    DO 10 j=jmin(i),jmax(i)
    x = d(i,j)
    g_μ(j) = g_μ-1(j) + x mit μ = n, n-1, ... , 1
    f_μ = f_μ-1 + x mit μ = n, n-1, ... , 1
    y = ko(i-1,j-1) + x + x
    g_0(j) = min { y, min { f_μ } }
                  1≤μ≤n
    f_0 = min { y, min { g_μ(j) } }
                  1≤μ≤n
10  ko(i,j) = min { g_0(j), f_0 }
20  CONTINUE
    Minimum = ko(I,J) / (I + J)
```

S_n-Algorithmus (We1, We2, We3 und We4):

```
    ko(i,j) = ∞ f.a. (i,j) ∉ P_n
    ko(0,0) = 0
    g_μ(j) = ∞ f.a. 0 ≤ μ ≤ n und f.a. 1 ≤ j ≤ J
    DO 20 i=1,I
    f_μ = ∞ f.a. 0 ≤ μ ≤ n
    DO 10 j=jmin(i),jmax(i)
    x = d(i,j)
    g_μ(j) = g_μ-1(j) + x mit μ = n, n-1, ... , 1
    f_μ = f_μ-1 + x  mit μ = n, n-1, ... , 1
    g_0(j) = ko(i,j) + x + x
    f_0 = g_0(j)
10  ko(i,j) = min { f_0, min { f_μ, g_μ(j) } }
                  1≤μ≤n

20  CONTINUE
    Minimum = ko(I,J) / (I + J)
```

Es sei noch angemerkt, daß bei den hier wiedergegebenen Verfahren zur Berechnung der Minima zur Verwaltung der ko(i,j) eigentlich nur J+2 Speicherplätze erforderlich sind (s. [1], Kapitel 10). Die Darstellung mit (I+1)(J+1) Speicherplätzen ist aber anschaulicher.
Zur Abschätzung der vom S_n- und F_n-Algorithmus benötigten Rechenzeiten machen wir zwei Annahmen: Eine Addition soll ebensoviel Rechenzeit erfordern wie eine Minimierung über eine zweielementige Menge, und eine Minimierung über eine m-elementige Menge ebensoviel wie m-1 Minimierungen über eine zweielementige Menge, $m \in \mathbb{N}$. Dann benötigt der S_n-Algorithmus pro $(i,j) \in P_n$ 4n+2 Operationen mit Dauer einer Addition und der F_n-Algorithmus 4n+3. (Zur Berechnung von Indizes nötige Operationen seien vernachlässigt.) Der prozentuale Mehrbedarf an Rechenzeit pro $(i,j) \in P_n$ bei der Berechnung des F_n-Minimums gegenüber der des S_n-Minimums ist in Tabelle 1 für einige n wiedergegeben.

allgemein	n=1	n=2	n=3	n=4
100/(4n+2)%	17%	10%	7%	6%

Tabelle 1: Prozentualer Mehrbedarf an Rechenzeit pro $(i,j) \in P_n$ bei der Berechnung des F_n-Minimums gegenüber der des S_n-Minimums.

Aus praktischen Gründen ist die Einführung von We4 also nicht zwingend. Folgendes ist aber noch zu beachten: Die geringe Anzahl der pro $(i,j) \in P_n$ benötigten Operationen ist mit der rekursiven Berechnung der Koeffizienten f_μ und $g_\mu(j)$ verbunden. Diese hat aber zur Folge, daß bei der Berechnung des F_n- und S_n-Minimums P_n ausgeschöpft wird. Bei Berechnungsverfahren, bei denen dies vermieden wird [3], steigt die Anzahl der pro errechnetem ko(i,j) benötigten Operationen stark an, und zwar bei Berechnung des F_n-Minimums etwas stärker als bei der des S_n-Minimums. (Dabei muß man an Stelle des originalen Algorithmus' von Sakoe und Chiba die erste in [1] entwickelte Variante verwenden.) Welche Berechnungsstrategie günstiger ist, die hier vorgestellte exhaustive oder die aus [3], muß die Praxis entscheiden.
Im folgenden sollen die Leistungen des V-, F_n- und S_n-Algorithmus' bei der Spracherkennung in einem Experiment untersucht und miteinander verglichen werden.

SPRACHMATERIAL UND DATENAUFBEREITUNG

Als Sprachmaterial wurden die Zahlen von Null bis Sechs

sowie Acht und Neun verwendet. Sie wurden von drei ca. 25-jährigen männlichen Sprechern des Hochdeutschen in einer schallisolierten Aufnahmekabine je zweimal auf Tonband gesprochen. Die Aufnahmen wurden bandpaßgefiltert mit den Abschneidefrequenzen 75 Hz und 5 kHz, mit einem 10Bit-AD-Wandler bei einer Abtastrate von 10kHz digitalisiert und auf Magnetplatte gespeichert. Daraufhin wurde für jedes Wort eine lückenlose Folge von Leistungsspektren mittels einer 256-Punkte-FFT errechnet, wobei die Fensterausschnitte mit einem Blackman-Harris-Fenster [4] gewichtet wurden. Anfang und Ende eines jeden Zahlwortes bestimmte ich manuell mit Hilfe des SONATA-Programmsystems [5]. Die Anzahl der für ein Zahlwort berechneten Spektren schwankte zwischen 15 und 27; im Mittel betrug sie 21,1. Der Gleichanteil der Spektren wurde im folgenden vernachlässigt. Jedes der Spektren normierte ich bezüglich seiner Gesamtenergie. Die normierten Spektren verwendete ich als Meßdatenvektoren und erstellte zu jedem Paar (A,B) von Zahlwörtern die Distanzmatrix $D_{A,B} = (d_{A,B}(i,j))$, wobei ich als Metrik d den mittleren euklidischen Abstand zwischen zwei normierten Spektren wählte.

EXPERIMENTE UND AUSWERTUNG

Ich führte drei Experimente durch, die ich auf zwei verschiedene Arten bezüglich des V-, F_n- und S_n-Minimums mit $1 \le n \le 4$ auswertete. Im ersten Experiment (E1) berechnete ich die Minima für die Distanzmatrizen, im zweiten (E2) verkleinerte ich die Distanzmatrizen, indem ich jede zweite Zeile und Spalte aus ihnen strich und die Minima bezüglich der verkleinerten Matrizen berechnete. Im dritten Experiment (E3) erstellte ich für jedes Spektrum i eines jeden Zahlwortes A eine Pseudo-Zufallszahl $r_A(i)$, die mit der Wahrscheinlichkeit 1/6 den Wert 4/5, mit gleicher Wahrscheinlichkeit den Wert 5/4 und mit der Wahrscheinlichkeit 2/3 den Wert 1 annahm. Für alle Paare (A,B) von Testworten berechnete ich nun die gestörte Distanzmatrix $\Delta_{A,B} = (\delta_{A,B}(i,j))$ mit $\delta_{A,B}(i,j) = d_{A,B}(i,j)\, r_A(i)\, r_B(j)$. Man überzeugt sich leicht, daß in jedem $\Delta_{A,B}$ die Hälfte der Elemente gegenüber $D_{A,B}$ verändert ist.
Bei jedem Experiment erhielt ich pro Testwort 53 Abstände zu von ihm verschiedenen Testwörtern. Um sprecherabhängige Effekte zu vermeiden, schied ich die vom jeweils selben Sprecher stammenden Testworte aus; es verblieben 36 Referenzworte pro Testwort. Die Worterkennung führte ich auf die beiden folgenden Arten durch: Beim MIN-Verfahren wies ich das Testwort dem Zahlwort zu, unter das das Referenzwort mit dem kleinsten Abstand zum Testwort fiel; beim MINQUER-Verfahren diente der minimale mittlere Abstand zu den unter ein Zahlwort fallenden Referenzworten als Kriterium.

INTERPRETATION DER VERSUCHSERGEBNISSE

An den in Abbildung 1 wiedergegebenen Erkennungsraten fällt zunächst auf, daß sie vergleichsweise niedrig sind. Das ist eine Folge der nur rudimentären Datenaufbereitung. Dennoch glaube ich, daß sich aus den Erkennungsraten einige Aussagen über die von mir verwendeten Algorithmen ableiten lassen.

- Bei Verwendung gestörter Distanzmatrizen werden beim MINQUER-Verfahren die Erkennungsraten der S_n- und F_n-Algorithmen nivelliert.
- Die schlechtesten Erkennungsraten liefert der V-Algorithmus. Die vom S_n-Algorithmus her bekannte Tendenz, mit steigendem n schlechtere Ergebnisse zu erbringen, findet sich auch beim F_n-Algorithmus.
- Für $1 \le n \le 2$ gilt: Für die S_n -Minima sinken die Erkennungsraten beim Übergang von vollständigen Distanzmatrizen (E1) zu verkürzten (E2) stark ab, während sich die F_n-Minima nicht wesentlich verändern.

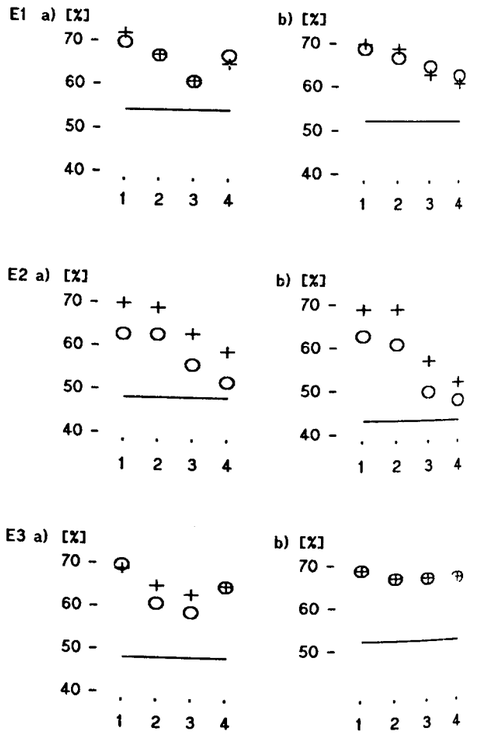

Abbildung 1: Experimentell ermittelte Erkennungsraten.
Abszisse: n, Ordinate: Erkennungsrate in Prozent;
a) MIN-Verfahren, b) MINQUER-Verfahren;
—: V-Algorithmus, +: F_n-Algorithmus,
O: S_n-Algorithmus.

Da bei E2 infolge der Verkürzung der Distanzmatrizen die Anzahl der in ihnen enthaltenen Elemente auf ca. ein Viertel sinkt, erniedrigt sich die Mächtigkeit der jeweiligen P_n auf ungefähr ein Viertel bis die Hälfte, je nach dem Verhältnis von I zu J.
Mit aller wegen der geringen Datenmenge gebotenen Vorsicht sei also folgende Hypothese aufgestellt: Für $1 \le n \le 2$ können wir beim F_n-Algorithmus die Rechenzeit senken, ohne die Erkennungsrate nennenswert zu beeinflussen, indem wir verkürzte Distanzmatrizen an Stelle der vollständigen verwenden. Beim S_n - Algorithmus haben wir diese Möglichkeit nicht. Trotz der etwas größeren Rechenzeit für eine Einzelminimierung (s. Tabelle 1) können wir also für $1 \le n \le 2$ mit dem F_n - Algorithmus bei der für die Worterkennung nötigen Gesamtminimierung Rechenzeit gegenüber dem S_n -Algorithmus sparen. Für die angegebenen Verkleinerungen von P_n ergibt sich aus den oben angegebenen für ein ko(i,j) jeweils benötigten Operationen eine Ersparnis von ca. 42% bis 71% für den F_1-Algorithmus und zwischen ca. 45% bis 73% für den F_2-Algorithmus.

LITERATURNACHWEISE

[1] F. W. a Campo (1987): Varianten des Dynamic-Programming-Algorithmus' von Sakoe und Chiba, IPKöln-Berichte 14, in Vorbereitung.
[2] H. Sakoe, S. Chiba (1978): Dynamic Programming Algorithm Optimization for Spoken Word Recognition, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol ASSP-26, 43-49.
[3] M. K. Brown, L. R. Rabiner (1982): An Adaptive, Ordered Graph Search Technique for Dynamic Time Warping for Isolated Word Recognition, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-30, 535-544.
[4] F. J. Harris (1978): On the use of windows for harmonic analysis with the discrete Fourier Transformation, Proc. IEEE 66, 51-83.
[5] J. E. Philipp, F. W. a Campo (1982): SONATA-Handbuch: Bedienungsanleitung zum **SONA**gram and Time Signal **A**nalyser, IPKöln-Berichte Nr. 12, 7-42.