

Der Smith-Waterman-Algorithmus

Praktikum Algorithmen in der Bioinformatik

Rolf Haynberg - 65109948

1 Motivation und Hintergrund

Der hier vorgestellte Algorithmus baut auf dem Needleman-Wunsch-Algorithmus auf und erweitert dessen Funktionalität. Deswegen werden zum Verständnis der folgenden Ausführungen Vorkenntnisse zur Funktionsweise des Needleman-Wunsch-Algorithmus empfohlen. Informationen dazu sind z.B. unter [Hay09, Heu06, Gus97] zu finden.

Sequenz-Alignments haben ein breites Spektrum von biologischen Anwendungen (vgl. [Gus97, S. 215]). Häufig ist es nicht nur von Interesse die Ähnlichkeit zwischen zwei Sequenzen zu bestimmen, sondern *teilweise* Übereinstimmungen zu finden. D.h. relativ kurze Teilabschnitte mit hoher Übereinstimmung. Ein wichtiger Spezialfall dabei ist, eine kurze Sequenz in einer längeren Sequenz zu finden (genau genommen wird der Abschnitt mit größtmöglicher Ähnlichkeit zu der kurzen Sequenz gesucht).

Das Ziel des Smith-Waterman-Algorithmus ist es, die Teilsequenzen mit der größten Übereinstimmung zwischen zwei gegebenen Sequenzen zu finden. Ein solches Paar aus Teilsequenzen nennt man *lokales Alignment*.

Algorithmen wie der Needleman-Wunsch-Algorithmus bieten diese Funktionalität nicht. Sie liefern eine Editier-Vorschrift zwischen zwei Strings zurück. Diese nennt man entsprechend *globales Alignment*.

Wie wir sehen werden, lässt sich der Needleman-Wunsch-Algorithmus jedoch auf einfache Weise so erweitern, dass er *lokale Alignments* findet.

Das Problem *lokale Alignments* zu finden könnte im ersten Moment viel schwieriger erscheinen als das für *globale Alignments*, da es viel mehr Kombinationsmöglichkeiten gibt. Ein naiver Algorithmus müsste alle Teilsequenzen aus der ersten Sequenz mit allen Teilsequenzen der zweiten Sequenz vergleichen. Glücklicherweise lässt sich mit dem Smith-Waterman-Algorithmus eine deutlich geringere Laufzeitkomplexität erzielen (für die Laufzeitanalyse siehe Abschnitt 4).

2 Der Algorithmus

Als Eingabe erhält der Algorithmus zwei Zeichenketten S_1 und S_2 mit Länge n bzw. m . Dabei sei $S[i]$ das i -te Zeichen und $S[1..i]$ das Präfix der Länge i . Wir definieren außerdem Hilfsweise, $S[1..0]$ als leeres Wort und $S[0]$ als ein spezielles Zeichen \perp , welches nicht im verwendeten Alphabet enthalten ist.

Gesucht ist das *lokale Alignment* mit der größten Übereinstimmung.

Aus dem Needleman-Wunsch-Algorithmus sind uns die Kostenfunktionen *insert*, *delete* und *replacematch* bekannt. Ziel war es, die Kosten zu minimieren. Dieser Ansatz lässt sich nicht auf den Smith-Waterman-Algorithmus übertragen. Weil hier Teilsequenzen gesucht werden, würden Kostenfunktionen dazu führen, dass Teilsequenzen der Länge 0 immer die geringsten Kosten haben. Deswegen verwenden wir hier Scoring-Funktionen, welche Scores für Ähnlichkeit definieren. Ziel ist es nun eine möglichst hohe Score zu erreichen.

Die Scoring-Funktionen sind für jede mögliche Editier-Operation definiert und tragen den selben Namen wie die Kostenfunktionen (genauer wird weiter unten beschrieben).

Eine weitere Eigenschaft die wir von den Scoring-Funktionen fordern ist, dass sie einen negativen Wert für Missmatches definieren. Dies ist notwendig, um ein Verhalten nach dem Prinzip *je länger desto besser* zu vermeiden. Denn andernfalls hätte ein Alignment welches die gesamte Sequenz umfasst nie eine niedrigere Score als ein Alignment mit einer Teilsequenz.

Wenn wir statt Kostenfunktionen die oben genannten Scoring-Funktionen im Needleman-Wunsch-Algorithmus verwenden, finden wir bereits die Similarity-Score aller Präfix-Paare zweier Sequenzen. Denn der Wert einer Zelle (i, j) in der erzeugten Matrix ist dann so definiert, dass er die Similarity-Score zwischen $S_1[1..i]$ und $S_2[1..j]$ darstellt.

Wir suchen allerdings die *Teilsequenzen* mit der größten Übereinstimmung. Dazu nutzen wir die Beobachtung, dass jede Teilsequenz Suffix eines Präfixes ist. Der hier vorgestellte Algorithmus wird es uns erlauben, für jeden Präfix-Vergleich die Stelle in den verglichenen Präfixen zu finden, an der das Suffix mit der größten Übereinstimmung beginnt.

Die Berechnungsvorschrift ist ähnlich zu der des Needleman-Wunsch-Algorithmus:

$$D_{0,0} = 0 \tag{1}$$

$$D_{i,j} = \max \begin{cases} 0 & \tag{2a} \\ D_{i,j-1} + \text{insert}(S_2[j]) & j \geq 1 & \tag{2b} \\ D_{i-1,j} + \text{delete}(S_1[i]) & i \geq 1 & \tag{2c} \\ D_{i-1,j-1} + \text{replacematch}(S_1[i], S_2[j]) & i, j \geq 1 & \tag{2d} \end{cases}$$

Da die Kostenfunktion durch eine Scoring-Funktion ersetzt wurde, muss bei der Berechnung das Maximum statt dem Minimum gebildet werden. Außerdem ist der Fall 0 (Gleichung 2a) hinzugekommen. Dieser verhindert, dass negative Werte in der Matrix auftauchen. Und genau dies ist die Idee hinter dem Smith-Waterman-Algorithmus: Würde eine Score negativ werden, wäre das entsprechende Alignment kein Kandidat mehr für ein lokales Alignment denn bereits die leere Teilsequenz hat Score 0. Deshalb wird hier der Wert auf 0 gesetzt und die Score ab dieser Stelle neu berechnet.

Um das lokale Alignment mit der größten Übereinstimmung zu finden, wird der größte Wert in der Matrix gesucht. Da es sich um den größten Wert handelt gibt es keine Erweiterung dieser Teilsequenz welche zu einer höheren Score führt. Die Zelle markiert also die Enden der gesuchten Teilsequenz in S_1 bzw. S_2 . Der Wert in dieser Zelle ist außerdem die Similarity-Score der Teilsequenzen.

Das Traceback

Um die Anfangsstellen der gesuchten Teilsequenzen in S_1 bzw. S_2 zu finden, wird ein Traceback ähnlich dem des Needleman-Wunsch-Algorithmus ausgeführt (der Traceback-Algorithmus für den Needleman-Wunsch-Algorithmus ist in [Hay09, Abschnitt 2.1] beschrieben). Hier wird jedoch abgebrochen sobald mit dem nächsten Schritt eine Zelle mit Wert 0 erreicht werden kann, denn dies bedeutet, dass an der aktuellen Stelle mit dem Berechnen der Score begonnen werden kann um den größten Wert der Matrix zu erhalten. Deshalb markiert diese Stelle den Beginn der gesuchten Teilsequenzen. Das ist auch der Grund dafür, dass die Zelle mit dem größten Wert die Similarity-Score der Teilsequenzen angibt.

3 Beispiel

Betrachten wir nun ein konkretes Beispiel mit

$$S_1 = \text{A A L}$$

$$S_2 = \text{A L T}$$

Die Similarity-Score Funktionen seien:

$$\text{insert}(x) \equiv \text{delete}(x) \equiv -1$$

und

$$\text{replacematch}(x, y) = \begin{cases} 1 & \text{falls } x = y \\ -1 & \text{sonst} \end{cases}$$

Aus der Definition (1) können wir bereits $D_{0,0} = 0$ ablesen.

Für die Berechnung der Werte in Zeile 0 muss das Maximum nur aus Fall 1 (Gleichung 2b) oder 0 (Gleichung 2a) gefunden werden, da in den anderen Fällen die Bedingungen nicht erfüllt sind (denn in der ersten Zeile ist $i = 0$). Fall 1 hat für die Zellen der ersten Zeile den Wert -1 weshalb das Maximum 0 ist. Für die Berechnung der Werte in Spalte 0 muss das Maximum von 0 und Fall 2 (Gleichung 2c) gefunden werden (die anderen Fälle werden nicht betrachtet da in der ersten Spalte $j = 0$ ist). Fall 2 ist für die erste Spalte immer -1 und daher ist das Maximum auch hier immer 0. Damit ergibt sich die Matrix in Abbildung 1.

	⊥	A	L	T
⊥	0	0	0	0
A	0			
A	0			
L	0			

Abbildung 1: Die Scoring-Matrix D zu den Zeichenketten $S_1 = \text{AAL}$ und $S_2 = \text{ALT}$. Die erste Zeile sowie erste Spalte sind bereits ausgefüllt. Da die Kosten für Mismatches negativ sind, ergibt sich für die erste Spalte und die erste Zeile überall der Wert 0. Zum besseren Verständnis sind die Zeichenketten an den Rändern aufgetragen.

Jetzt wird die Matrix nach und nach ausgefüllt. Dabei werden Zellen (i, j) berechnet, zu denen $(i - 1, j - 1)$, $(i - 1, j)$ und $(i, j - 1)$ bereits bekannt sind. Solche Zellen lassen sich immer finden (dies ist z.B. der Fall wenn zeilenweise vorgegangen wird). Für $D_{1,1}$ berechnen wir

$$D_{1,1} = \max\{0, 0 + 1, 0 - 1, 0 - 1\} = 1$$

Für $D_{2,1}$ ergibt sich

$$D_{2,1} = \max\{0, 0 + 1, 1 - 1, 0 - 1\} = 1$$

Schließlich erhalten wir die Matrix aus Abbildung 2. In Zelle $(3, 2)$ steht der größte Wert der Matrix, nämlich 2. Dies ist gleichzeitig die Similarity-Score des gesuchten Alignments. Das Traceback führt zu Zelle $(2, 1)$. Die gesuchte Teilsequenz in S_1 ist also $S_1[2, 3] = \text{A L}$ und in S_2 ist es $S_2[1, 2] = \text{A L}$.

In diesem Beispiel sind die Teilsequenzen mit der größten Übereinstimmung gleich lang und enthalten die gleichen Zeichen. Beides ist aber im Allgemeinen nicht der Fall.

	⊥	A	L	T
⊥	0	0	0	0
A	0	1	0	0
A	0	1	0	0
L	0	0	2	1

Abbildung 2: Die vollständig ausgefüllte Scoring-Matrix D für S_1 und S_2 . Mit maximalem Wert 2.

4 Zusammenfassung

Der Smith-Waterman-Algorithmus stellt eine einfache Erweiterung des Needleman-Wunsch-Algorithmus dar um lokale Alignments zu finden. Es ergibt sich auch eine vergleichbare Laufzeit was durchaus überraschend erscheinen mag unter Anbetracht der Menge von möglichen Teilsequenzpaaren.

Laufzeitanalyse

Um den Wert einer Zelle $D_{i,j}$ zu berechnen müssen die drei Nachbarzellen $D_{i-1,j}$, $D_{i,j-1}$, $D_{i-1,j-1}$ untersucht werden sowie die Strings an den Positionen $i-1$ und $j-1$. Diese Kosten können als Konstant angenommen werden (z.B. im RAM-Modell). Die Matrix enthält $(n+1) \times (m+1)$ Zellen, sodass sich insgesamt eine Laufzeit in $O(nm)$ ergibt.

Literatur

- [Gus97] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences*, chapter 11, pages 215–253. Cambridge University Press, first edition edition, 1997.
- [Hay09] Rolf Haynberg. Der Needleman-Wunsch-Algorithmus. November 2009. Dokumentation des Algorithmus für das Praktikum Algorithmen der Bioinformatik.
- [Heu06] Volker Heun. Skriptum zur Vorlesung Algorithmische Bioinformatik I/II, 2005-2006. gehalten im SS 05 und WS 05/06.