Prof. Dr. Heiner Klocke
Fachhochschule Köln
Informatik

03.12.09

Praktikum: Dynamische Programmierung

Einführung: Sequence Alignment

In Anwendungen der Bioinformatik werden zwei oder mehrere DNA-Sequenzen verschiedener Organismen miteinander verglichen. Ein DNA-Strang besteht aus einer Kette von Molekülen, genannt Basen oder Nukleoside. Die DNA-Basen sind

- Adenin (A)
- Guanin (G)
- Cytosin (C)
- Thymin (T)

Ein DNA-Strang kann als Sequenz über der Basenmenge [A, C, G, T] dargestellt werden.

Durch den Vergleich von DNA-Strängen können Aussagen über die funktionelle oder die evolutionäre Verwandtschaft von Lebewesen gemacht werden.

Das Ziel des DNA-Sequenzvergleichs ist es somit zu entscheiden, wie ähnlich sich zwei DNA-Sequenzen sind. Die ***Ähnlichkeit*** kann auf verschiedene Art und Weisen definiert werden:

- Zwei Sequenzen **a** und **b** sind ähnlich, wenn die eine Sequenz Teilsequenz der anderen ist.
- Zwei Sequenzen sind ähnlich, wenn die Anzahl der Änderungen, die man benötigt, um eine Sequenz in die andere umzuwandeln, möglichst gering ist.
- Finde eine dritte Sequenz **c**, dessen Basen in **a** und in **b** vorkommen; die Basen müssen in derselben Reihenfolge auftreten, aber nicht unbedingt aufeinander folgen.

Das **Ähnlichkeitsmaß beim Alignment**, das hier verwendet werden soll, lautet:

- Schreibe die Sequenzen so untereinander, dass möglichst viele gleiche Buchstaben in einer Spalte stehen und in die Sequenzen möglichst wenig Gaps, also Lücken oder Platzhalter, eingefügt werden müssen.

Durch eine ***Bewertungsfunktion*** (alignment score) wird für jede Übereinstimmung zwischen den Buchstabenpaaren der Sequenzen eine positive Punktzahl vergeben. Damit ein sinnvolles Alignment überhaupt möglich ist und da die Sequenzen oft unterschiedlich lang sind, dürfen Leerstellen (**Gaps**) in die Sequenzen eingefügt werden. Allerdings wird das Einfügen von Gaps durch Punktabzug (negative Punktzahl, gap penalty) bestraft. Das Alignment mit der höchsten Punktzahl ist dann ein optimales Alignment.

Beispiel 1:

-AAACGG
AAAACCG

An der ersten Stelle wird ein Gap eingefügt, um den Längenunterschied auszugleichen.

Beispiel 2:

Eine Sequenzausrichtung zwischen zwei menschlichen Zinkfingerproteinen aus der GenBank, (http://www.ncbi.nlm.nih.gov/Genbank/) ist folgende:

```
AAB24882        TYHMCQFHCRYVNNHSGEKLYECNERSKAFSCPSHLQCHKRRQIGEKTHEHNQCGKAFPT 60
AAB24881        --------------------YECNQCGKAFAQHSSLKCHYRTHIGEKPYECNQCGKAFSK 40
                                    ****: .***:  * *:** * :****.:* ********..

AAB24882        PSHLQYHERTHTGEKPYECHQCGQAFKKCSLLQRHKRTHTGEKPYE-CNQCGKAFAQ- 116
AAB24881        HSHLQCHKRTHTGEKPYECNQCGKAFSQHGLLQRHKRTHTGEKPYMNVINMVKPLHNS 98
                 **** *:***********:***:**.: .***************     :  *.: :
```

(Quelle: mit dem System ClustalW erstellt, http://de.wikipedia.org/wiki/ClustalW)

Beim paarweisen Alignment – Vergleiche zweier DNA-Sequenzen – wird unterschieden zwischen

- globalem Alignment, bei dem alle Symbole einer Sequenz berücksichtigt werden, und
- lokalem Alignment, wenn keine Übereinstimmung der Sequenzen auf der gesamten Länge zu erwarten ist.

Eine einfache Scoring-Alignment-Funktion wäre z.B. folgende:

- **match**: score +1 (die beiden untereinander stehenden Buchstaben stimmen überein)
- **mismatch**: score -1 (keine Übereinstimmung, Mutation)
- **gap**: score -2 (gap penalty, "Insert or Deletion", d.h. Basen fehlen oder wurden eingefügt)

Globales Sequence Alignment – der Needleman-Wunsch-Algorithmus

Das Verfahren benötigt zwei Sequenzen a und b, eine Matrix *D* der Größe *D(n+1, m+1)* mit *n = len(a)*, *m=len(b)*. Weiterhin wird eine Bewertungsfunktion $w(a_i, b_j)$ benötigt, die zwei Nukleoside vergleicht und bewertet. Die Matrix wird mit einer Rekursionsgleichung erstellt, welche die Teillösungen (Teilalignments) und das globale Alignment berechnet.

Beispiel 3:

Sequenz a : A C G T C E
Sequenz b : A G T C D E

Die Scorematrix D zum Speichern der Teillösungen wird angelegt, wobei eine Zeile und eine Spalte hinzugefügt wird, um Gaps zu ermöglichen. Da die Sequenzen **a** und **b** die Größe 6 haben, erhalten wir eine 7 × 7 Matrix *D(7, 7)*. Die Indizierung der Zeilen i=0, 1, ... ,n und der Spalten j = 0, 1, ... , m beginnt wegen der Gap-Zeile bzw. –Spalte bei 0!

$$D_{init} = \begin{bmatrix} & - & A & G & T & C & D & E \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ G & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ T & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ E & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Eine mögliche **Bewertungsfunktion** lautet:

$$w(a_i, b_j) = \begin{pmatrix} 1 & falls\ a_i = b_j \\ -1 & falls\ a_i \neq b_j \\ -1 & falls\ a_i\ oder\ b_j\ ein\ Gap\ (= "-") \end{pmatrix}$$

Die in der **Scorematrix** D gespeicherten optimalen Substrukturen sind wie folgt definiert:

D(0,0) = 0

D(i, 0) = f(i), 1 ≤ i ≤ n

D(0, j) = f(j), 1 ≤ j ≤ m

$$D(i,j) = \max \begin{Bmatrix} D(i-1, j-1) + w(a_i, b_j) & Match\ bzw.\ Mismatch \\ D(i-1, j) + w(a_i, -) & Deletion \\ D(i, j-1) + w(-, b_j) & Insertion \end{Bmatrix}, 1 \le i \le m, 1 \le j \le n$$

Die erste Zeile und die erste Spalte der Matrix D stehen für einen für einen Match einer Sequenz gegenüber einer anderen leeren Sequenz, also einem Gap. Mit der obigen Bewertungsfunktion w wird hier ein Gap Penalty von -1 angenommen. Aber auch jede andere Penalty-Funktion, abhängig von der Anwendung, ist möglich.

In der *D(i, j)*-Definition wird durch das Zeichen „–“ ein Gap bezeichnet.

Erster Schritt: Initialisiere die Bewertungsmatrix D. Berechne zuerst die Einträge der Matrix *D*(*i*, *j*) für die erste Zeile und die erste Spalte: Die Bewertung für den Eintrag *D(1, 0)* wird

berechnet aus der darüber liegenden Bewertung *D(i-1, j*) = D(0,0) = 0 und dem Score an der Stelle $w(a_i, b_i) = w(a_1, -) = w(A, -) = -1$. Also *D(1, 0)* = 0 + (-1) = -1 die anderen Werte werden nun analog berechnet.

$$D_{init} = \begin{bmatrix} & - & A & G & T & C & D & E \\ - & 0 & -1 & -2 & -3 & -4 & -5 & -6 \\ A & -1 & 1 & -1 & -1 & -1 & -1 & -1 \\ C & -2 & -1 & -1 & -1 & 1 & -1 & -1 \\ G & -3 & -1 & 1 & -1 & -1 & -1 & -1 \\ T & -4 & -1 & -1 & 1 & -1 & -1 & -1 \\ C & -5 & -1 & -1 & -1 & 1 & -1 & -1 \\ E & -6 & -1 & -1 & -1 & -1 & -1 & 1 \end{bmatrix}$$

Die erste Zeile und die erste Spalte sind schon komplett durchgerechnet; eigentlich würden diese Werte erst ganz am Schluss dort stehen.

Zweiter Schritt: Erzeuge die Bewertungsmatrix D von links oben nach rechts unten nach der Rekursionsgleichung für D.

Endergebnis (bitte nachprüfen, ob es stimmt): a : A C G T C E, b : A G T C D E

D(1,2)

$$D_{init} = \begin{bmatrix} & - & A & G & T & C & D & E \\ - & 0 & -1 & -2 & -3 & -4 & -5 & -6 \\ A & -1 & 1 & 0 & -1 & -2 & -3 & -4 \\ C & -2 & 0 & 0 & -1 & 0 & -1 & -2 \\ G & -3 & -1 & 1 & 0 & -1 & -1 & -2 \\ T & -4 & -2 & 0 & 2 & 1 & 0 & -1 \\ C & -5 & -3 & -1 & 1 & 3 & 2 & 1 \\ E & -6 & -4 & -2 & 0 & 2 & 2 & 3 \end{bmatrix}$$

$$D(1,1) = \max \begin{cases} D(0,0) + w(A,A) & \Rightarrow 0 + 1 \\ D(0,1) + w(A,-) & \Rightarrow -1 + (-1) \\ D(1,0) + w(-,A) & \Rightarrow -1 + (-1) \end{cases}$$

→ Das Maximum entsteht aus dem ersten Fall, d. h. A wird mit A ausgerichtet.

$$D(1,2) = \max \begin{cases} D(0,1) + w(A,C) & \Rightarrow -1 + (-1) \\ D(0,2) + w(A,-) & \Rightarrow -2 + (-1) \\ D(1,1) + w(-,C) & \Rightarrow 1 + (-1) \end{cases}$$

→ Das Maximum entsteht aus dem dritten Fall, da hier das Maximum der Berechnung, nämlich 0 entsteht, d. h. ein Gap (–) würde mit C ausgerichtet.

$$D(1,3) = \max\begin{cases} D(0,2) + w(A,G) & \Rightarrow -2 + (-1) \\ D(0,3) + w(A,-) & \Rightarrow -3 + (-1) \\ D(1,2) + w(-,C) & \Rightarrow 0 + (-1) \end{cases}$$

$$D(1,4) = \max\begin{cases} D(0,3) + w(A,T) & \Rightarrow -3 + (-1) = -4 \\ D(0,4) + w(A,-) & \Rightarrow -4 + (-1) = -5 \\ D(1,3) + w(-,C) & \Rightarrow -1 + (-1) = -2 \end{cases}$$

$$D(2,1) = \max\begin{cases} D(1,0) + w(G,A) & \Rightarrow -1 + (-1) = -2 \\ D(1,1) + w(G,-) & \Rightarrow 1 + (-1) = 0 \\ D(2,1) + w(-,A) & \Rightarrow 0 + (-1) = -1 \end{cases}$$

usw.

Ergebnis: Alignment a : A C G T C – E
Alignment b : A – G T C D E

Aufgabe 5 (Needleman-Wunsch-Algorithmus)

Schreiben Sie ein Java-Programm, mit dem das Sequence-Alignment-Problem nach dem Needleman-Wunsch-Algorithmus gelöst werden kann.

Folgende Parameter sollen interaktiv eingegeben werden:

1. Die Sequenzen a und b
2. Die Bewertungsfunktion w

Führen Sie das Programm schrittweise aus und stellen Sie nach jedem Schritt die Score-Matrix D dar.

Ermitteln Sie aus der Ergebnismatrix D die berechnete optimale Ausrichtung der beiden Sequenzen a und b und stellen Sie die Ausrichtung dar.

Erklären Sie genau das Prinzip der dynamischen Programmierung, das hier zum Einsatz kommt.

Fertigen Sie eine Dokumentation an.

Aufgabe 6 (Smith-Waterman-Algorithmus)

Implementieren Sie den Smith-Waterman-Algorithmus für lokales Sequence Alignment. Recherchieren Sie selbst. Dokumentation!