Bioinformatik - Fallstudie 3

Hochschule für Technik und Wirtschaft des Saarlandes

Sommersemester 2025

Leon Weyand
Matrikelnummer: 3849163

13. August 2025

Inhaltsverzeichnis

Aufgabe 1 Viterbi

1.1 Normale vs. Logarithmische Skala

Um zu testen, ob es einen Unterschied zwischen der Verwendung der normalen Wahrscheinlichkeitsskala und der logarithmischen Skala gibt, habe ich beide Ansätze in meinem Programm anhand des Würfelproblems gegenübergestellt. Das Ziel bestand darin, festzustellen, ob sich in der Praxis Unterschiede in den Ergebnissen oder im Verhalten des Algorithmus zeigen.

Da die logarithmische Skala nur eine mathematische Umformung ist, liefern beide Ansätze mathematisch dieselbe optimale Zustandsfolge. Der Unterschied liegt jedoch in der internen Berechnung:

- **Normale Skala:** Hier werden die Wahrscheinlichkeiten bei jedem Schritt miteinander multipliziert. Bei längeren Sequenzen werden die Werte durch die vielen Multiplikationen extrem klein. Irgendwann sind sie so klein, dass sie in Python nicht mehr durch einen Float-Wert dargestellt werden können und stattdessen als 0.0 erscheinen. Ab diesem Punkt sind keine korrekten Berechnungen mehr möglich und das Programm kann keinen gültigen Pfad finden.
- **Logarithmische Skala:** Hier werden nicht die Wahrscheinlichkeiten selbst, sondern deren Logarithmen gespeichert und addiert. Dadurch bleiben die Werte im darstellbaren Zahlenbereich, selbst bei sehr langen Sequenzen. Underflow-Fehler, wie sie bei der normalen Skala auftreten, werden so vermieden.

In meinem Experiment konnte ich die Datei „wuerfel2025.txt" mit der normalen Skala nicht vollständig verarbeiten, da die Werte zu früh auf 0 gerundet wurden. Mit der logarithmischen Skala lief der Algorithmus hingegen problemlos durch und der korrekte Pfad wurde mit einer Genauigkeit von 87% berechnet.

Zusammengefasst lässt sich festhalten, dass die logarithmische Skala in der Praxis bei längeren Sequenzen deutlich robuster ist und numerische Probleme verhindert, auch wenn beide Skalen theoretisch zum gleichen Ergebnis führen sollten.

1.2 Umgedrehte Sequenz

Um zu testen, ob es einen Unterschied macht, ob ich die originale oder die umgedrehte Sequenz verwende beziehungsweise ob am Ende derselbe (nur

umgedrehte) Viterbi-Pfad herauskommt, habe ich die gegebene Beobachtungssequenz zunächst mit meinem HMM normal decodiert. Anschließend habe ich die Sequenz vollständig umgedreht und erneut mit demselben HMM decodiert.

Meine Vermutung vorab war, dass sich die Pfade beim in der Vorlesung verwendeten Würfelproblem nicht verändern. Das heißt, der Viterbi-Pfad der umgedrehten Sequenz ist die exakte Umkehr des ursprünglichen Pfads. Meine Begründung ist, dass das Würfelproblem symmetrisch ist, es gibt keine Präferenz für den Anfang oder das Ende der Sequenz, P(F) und P(L) sind beide 50% und die Übergangswahrscheinlichkeiten sind in beide Richtungen gleich.

Diese Vermutung hat sich bei meinen Beobachtungen bestätigt. Beim Standard-Würfelproblem ist der Pfad der umgedrehten Sequenz tatsächlich genau die Umkehr des Originals. Dies gilt sowohl für die normale Wahrscheinlichkeitsskala als auch für die logarithmische Skala.

Wird jedoch ein Hidden-Markov-Modell gewählt, das anders als beim Würfelproblem nicht symmetrisch ist, ändert sich der Pfad der umgekehrten Sequenz. Um dies zu testen, habe ich das Experiment erneut durchgeführt, diesmal habe ich jedoch die Startwahrscheinlichkeiten vollständig auf einen Zustand gesetzt, beispielsweise P(F) = 100 %. In diesem Fall ist der Viterbi-Pfad der umgekehrten Sequenz nicht einfach die Umkehr des ursprünglichen Pfads, da die Symmetrie dadurch gebrochen wurde.

Aufgabe 2 Viterbi versus Posteriori-Decodierung

Die Viterbi-Dekodierung und die Posteriori-Dekodierung sind zwei unterschiedliche Verfahren, um die wahrscheinlichste Zustandsinformation aus einem Hidden-Markov-Modell zu gewinnen. Bei der Viterbi-Dekodierung wird der gesamte wahrscheinlichste Pfad durch alle Zustände gesucht. Das bedeutet, dass die eine einzige Zustandsfolge bestimmt wird, die die gegebene Beobachtungssequenz mit der höchsten Wahrscheinlichkeit erzeugt. Das Ergebnis ist also eine zusammenhängende Abfolge von Zuständen, die als am wahrscheinlichsten angenommen wird.

Die Posteriori-Dekodierung funktioniert anders. Hier wird nicht ein einzelner Gesamtpfad bestimmt, sondern für jede Position der Beobachtungssequenz wird berechnet, welcher Zustand dort am wahrscheinlichsten ist. Das Ergebnis ist eine Abfolge der jeweils wahrscheinlichsten Zustände an den einzelnen Positionen. Dadurch liefert diese Methode ein genaueres Bild darüber, wie sicher oder unsicher die Zustandszuordnung an den einzelnen Positionen ist. [2]

Für die Posteriori-Dekodierung werden zwei Teilschritte kombiniert, die als Forward- und Backward-Berechnung bezeichnet werden. Die Forward-Berechnung ermittelt, wie wahrscheinlich es ist, die bisher gesehene Sequenz bis zu einer bestimmten Position zu erzeugen, wenn man sich in einem bestimmten Zustand befindet. Die Backward-Berechnung macht das Gleiche in die andere Richtung und berechnet, wie wahrscheinlich es ist, den Rest der Beobachtungssequenz ab dieser Position zu erzeugen, wenn man in einem bestimmten Zustand startet. Durch die Kombination beider Werte lässt sich für jede Position genau bestimmen, wie wahrscheinlich jeder mögliche Zustand dort ist, wenn man die gesamte Sequenz betrachtet. [1]

Der Hauptunterschied zwischen den beiden Ansätzen besteht also darin, dass Viterbi einen einzigen konsistenten Gesamtpfad liefert, während die Posteriori-Dekodierung eine Position-für-Position-Analyse ist. Der Viterbi-Algorithmus eignet sich daher vor allem, wenn ein einzelner Pfad deutlich dominiert. Die Posteriori-Dekodierung berücksichtigt dagegen konkurrierende Pfade und liefert bessere Ergebnisse, wenn mehrere Pfade fast gleich wahrscheinlich sind. [3]

Literatur

[1] LibreTexts, Manolis Kellis et al.: *Posterior Decoding and Learning*. https://bio.libretexts.org/Bookshelves/Computational_Biology/Book%3A_Computational_Biology_-_Genomes_Networks_and_Evolution_%28Kellis_et_al.%29/08%3A_Hidden_Markov_Models_II-Posterior_Decoding_and_Learning/8.02%3A_Posterior_Decoding, Zugriff am 16. Mai 2025.

[2] Krogh, A.: *Hidden Markov models for gene finding in DNA*. BMC Bioinformatics, 11(Suppl 1):S28, 2010. https://doi.org/10.1186/1471-2105-11-S1-S28

[3] He, X., Cai, D., and Niyogi, P.: *Posterior Decoding for Gene Finding Based on Hidden Markov Models*. BMC Bioinformatics, 6(Suppl 4):S12, 2005. https://doi.org/10.1186/1471-2105-6-S4-S12.