Algorithmen der Bioinformatik

und verwandte Themen

Dipl.-Inform. Tobias Marschall
Prof. Dr. Sven Rahmann

LS 11, Fakultät für Informatik, TU Dortmund

2008–2009
Entwurf vom 25. Mai 2009

Inhaltsverzeichnis

Vorbemerkungen

Dieses Dokument enthält Skripte zu meinen Bioinformatik-Lehrveranstaltungen an der TU Dortmund, insbesondere

- Algorithmen auf Sequenzen (zuletzt gelesen im SoSe 2008)
- Algorithmische Bioinformatik (zuletzt gelesen im WiSe 2008/09)
- Einführung in die Systembiologie bzw. Rekonstruktion regulatorischer Netzwerke (zuletzt gelesen im SoSe 2009)

Zur Zeit befindet sich dieses Skript im Aufbau; mit Fehlern und Unvollständigkeiten ist daher leider noch zu rechnen. Für Hinweise dazu bin ich jederzeit dankbar.

– Prof. Dr. Sven Rahmann, TU Dortmund

KAPITEL 1

Motivsuche mit dem EM-Algorithmus

1.1 Ein likelihood-basierter Ansatz zur Motivsuche mit PFMs

Wir spezifizieren das allgemeine Motivsuche-Problem wie folgt.

Problem 1.1 (likelihood-basierte Motivsuche mit PFMs)**.** Gegeben sind N Sequenzen $y = (y_1, \ldots, y_N)$ endlicher Länge über einem Alphabet $A = \{a_1, \ldots, a_K\}$, sowie eine Motivlänge L.

Gesucht sind *auffällige Motive* in y. Bekanntlich müssen wir präzisieren, was Motive sind und was wir unter "auffällig" verstehen.

In diesem Abschnitt verstehen wir unter einem Motiv eine $K \times L$-Positions-Häufigkeitsmatrix (PFM): In jeder der L Spalten steht eine Wahrscheinlichkeitsverteilung über dem Alphabet.

Zum Bewerten eines Motivs benutzen wir daher ein probabilistisches Modell einen likelihood-Ansatz; d.h. wir suchen das Motiv, das die likelihood des Gesamtmodells maximiert. •

Wir beschreiben jetzt das Modell.

Zur Vereinfachung stellen wir uns vor, dass wir nicht N Sequenzen gegeben haben, sondern direkt deren Teilstrings der Länge L, die wir ab jetzt als unabhängig betrachten. Die Eingabe besteht also aus $x = (x_1, \ldots, x_n)$ mit $x_i = (x_{i1}, \ldots, x_{iL}) \in A^L$ für alle $1 \leq i \leq n$.

Wir stellen uns vor, dass jedes L-mer x_i entweder vom Motiv-Modell (der PFM) oder von einem Hintergrundmodell (einer nicht positionsspezifischen Verteilung) wie folgt erzeugt wird.

Zunächst wird für jedes x_i die Modell-Komponente (Motiv oder Hintergrund) ausgewählt. Mit Wahrscheinlichkeit λ_1 wird das Motiv gewählt; mit Wahrscheinlichkeit $\lambda_2 = 1 - \lambda_1$ der

Hintergrund. Diese Entscheidung fassen wir in Indikatorvariablen Z_{i1}, Z_{i_2} zusammen; es sei $Z_{ij} := 1$, wenn für x_i Komponente j gewählt wurde, und $Z_{ij} := 0$ sonst.

Die Parameter der beiden Komponenten bezeichnen wir mit θ_1 bzw. θ_2. Wurde das Motiv gewählt, so werden L Symbole nach der PFM $\theta_1 = f = (f_{ak})$ gezogen, das k-te Symbol nach der k-ten Spalte. Die Wahrscheinlichkeit, ein L-mer x_i zu erzeugen, ist damit

$$\mathbb{P}_{\theta_1}(x_i) = \prod_{k=1}^{L} f_{x_{ik},k}. \tag{1.1}$$

Wurde der Hintergrund gewählt, so werden L Symbole nach der Hintergrundverteilung $\theta_2 = f_0 = (f_{a0})$ gezogen. Die Wahrscheinlichkeit, ein L-mer x_i zu erzeugen, ist damit

$$\mathbb{P}_{\theta_2}(x_i) = \prod_{k=1}^{L} f_{x_{ik},0}. \tag{1.2}$$

Damit ergibt sich bei bekannten Parametern $\theta = (\theta_1, \theta_2)$ und $\lambda = (\lambda_1, \lambda_2)$ die Gesamtwahrscheinlichkeit für die Eingabe x wie folgt:

$$\begin{aligned}
\mathbb{P}_{\theta,\lambda}(x) &= \prod_{i=1}^{n} \mathbb{P}_{\theta,\lambda}(x_i) \\
&= \prod_{i=1}^{n} \sum_{j=1}^{2} \mathbb{P}_{\theta,\lambda}(x_i,\, Z_{ij} = 1) \\
&= \prod_{i=1}^{n} \sum_{j=1}^{2} \mathbb{P}_{\theta,\lambda}(x_i \mid Z_{ij} = 1) \cdot \mathbb{P}_{\theta,\lambda}(Z_{ij} = 1) \\
&= \prod_{i=1}^{n} \sum_{j=1}^{2} \mathbb{P}_{\theta_j}(x_i) \cdot \lambda_j.
\end{aligned}$$

Die likelihood-Funktion ist genau diese Wahrscheinlichkeit als Funktion der Parameter (θ, λ); wir schreiben

$$L_x(\theta, \lambda) := \mathbb{P}_{\theta,\lambda}(x).$$

Ziel der Motivsuche ist, die Parameter so zu bestimmen (bei gegebenem x), dass die likelihood maximal wird. Statt die likelihood zu maximieren, maximieren wir ihren Logarithmus (log-likelihood):

$$\mathcal{L}_x(\theta, \lambda) := \log L_x(\theta, \lambda) = \sum_{i=1}^{n} \log \left[\sum_{j=1}^{2} \lambda_j \, \mathbb{P}_{\theta_j}(x_i) \right]. \tag{1.3}$$

Wie man erahnen kann, ist eine direkte Maximierung dieses Ausdrucks in allen Parametern schwierig. Hinzu kommt, dass die log-likelihood-Funktion aufgrund der Problemstellung höchstwahrscheinlich mehrere lokale Maxima aufweist: Es ist ja plausibel, dass mehrere Motive in der Eingabe versteckt sind.

Es ist natürlich möglich, beliebige nichtlineare Optimierungsroutinen für $\mathcal{L}_x(\theta, \lambda)$ zu verwenden; dabei sollte jedoch bedacht werden, dass bei der DNA-Motivsuche $3(L + 1) + 1$ freie

Parameter zu bestmmen sind ($3L$ aus θ_1, 3 aus θ_2, einer aus λ), also ein hochdimensionales Problem vorliegt. Statt die Optimierung in eine "black box" auszulagern, beschreiten wir einen anderen Weg.

Das Problem besteht darin, dass die Zugehörigkeiten Z_{ij} unbekannt sind. deswegen müssen wir, um $\mathbb{P}_{\theta,\lambda}(x)$ zu erhalten, über alle möglichen Werte von Z_{ij} summieren; dadurch kommt in Gleichung (1.3) die Summe in den Logarithmus, und deswegen ist die Optimierung schwierig. Wenn wir aber so tun, als seien die Z_{ij} bekannt, wird alles wesentlich einfacher. Das Problem dabei ist, dass wir die Kenntnis von Z_{ij} irgendwo hernehmen müssen.

Wir behandeln im nächsten Abschnitt ganz allgemein den EM-Algorithmus, mit dem man die likelihood in probabilistischen Mehrkomponenten-Modellen (engl. *mixture models*) iterativ maximiert, indem man in einem Schritt die Zugehörigkeit Z_{ij} der x_i zu den einzelnen Komponenten schätzt und in einem anderen Schritt die Modellparameter (θ, λ) für diese Schätzung so bestimmt, dass die likelihood dabei maximiert wird.

1.2 Der EM-Algorithmus für Mehrkomponentenmodelle

Wir betrachten folgende allgemeine Situation: Es gibt n Eingaben x_i der Dimension d über einem endlichen oder unendlichen Alphabet A, also $x = (x_1, \ldots, x_n)$ mit $x_i = (x_{i1}, \ldots, x_{id}) \in A^d$. Dabei wird jedes x_i unabhängig von einem C-Komponenten-Modell erzeugt.

Zunächst wird Komponente j mit Wahrscheinlichkeit λ_j ausgewählt; dabei ist $\lambda_j \geq 0$ für alle j und $\sum_{j=1}^{C} \lambda_j = 1$. Die gewählte Zugehörigkeit wird durch einen C-dimensionalen Vektor $Z_i = (Z_{i1}, \ldots, Z_{iC})$ ausgedrückt, so dass $Z_{ij} = 1$ genau dann wenn x_i zu Komponente j gehört, und $Z_{ij} = 0$ sonst.

Dann wird x_i nach einer für Komponente j spezifischen Verteilung gezogen, die wir mit P_{θ_j} bezeichnen; θ_j beinhaltet dabei alle Parameter für Komponente j. Wir gehen davon aus, dass alle diese Modelle bekannt sind und damit $P_{\theta_j}(x_i)$ berechnet werden kann, wenn Parameter θ_j bekannt sind.

Wir berechnen die Wahrscheinlichkeit, die Komponente j auszuwählen *und* ein Datum x zu erzeugen:

$$\mathbb{P}_{\theta,\lambda}(x_i,\, Z_{ij} = 1) = \lambda_j \cdot P_{\theta_j}(x_i).$$

Für einen beliebigen C-dimensionalen Einheitsvektor z schreiben wir dies als Produkt, bei dem wir über den Exponenten (0 oder 1) jeweils eine Komponente auswählen:

$$\begin{aligned}\mathbb{P}_{\theta,\lambda}(x_i,\, Z_i) &= \prod_{j=1}^{C} \lambda_j^{Z_{ij}} \cdot \prod_{j=1}^{C} \mathbb{P}_{\theta_j}(x_i)^{Z_{ij}} \\ &= \prod_{j=1}^{C} \left[\lambda_j \cdot \mathbb{P}_{\theta_j}(x_i)\right]^{Z_{ij}}.\end{aligned}$$

Damit ergibt sich die log-likelihood-Funktion zu allen Daten (x, Z) als

$$\mathcal{L}_{x,Z}(\theta, \lambda) = \sum_{i=1}^{n} \sum_{j=1}^{C} Z_{ij} \cdot \left[\log \lambda_j + \log \mathbb{P}_{\theta_j}(x_i)\right]. \tag{1.4}$$

Gegenüber der Gesamt-log-likelihood für die Daten x in (1.3) fällt die einfachere Struktur auf, aber eben auch, dass die unbekannten Z_{ij} auftreten.

Der EM-Algorithmus besteht nun darin, iterativ bessere Parameter für (θ, λ) zu berechnen, indem

- sinnvolle oder zufällige Startwerte (θ^0, λ^0) für die Parameter (θ, λ) gewählt werden,
- im **E-Schritt** die zu aktuell optimierende Funktion $f_{\theta^0,\lambda^0,x}(\theta, \lambda)$ als *Erwartungswert* der log-likelihood $\mathcal{L}_{x,Z}(\theta, \lambda)$ gewählt wird; der Erwartungswert wird über die Z_{ij} gebildet; dabei wird die bedingte Verteilung der Z_{ij} gegeben x_i und die aktuellen Parameterwerte (θ^0, λ^0) zugrunde gelegt; es ist also

$$f_{\theta^0,\lambda^0,x}(\theta, \lambda) := \mathbb{E}_{Z \mid (x;\, \theta^0,\lambda^0)} \left[\mathcal{L}_{x,Z}(\theta, \lambda)\right]. \tag{1.5}$$

- im **M-Schritt** (*Maximierung*) neue Parameterwerte $(\theta^*, \lambda^*) = \operatorname{argmax}_{(\theta,\lambda)} f_{\theta^0,\lambda^0,x}(\theta, \lambda)$ berechnet werden und diese als neue Parameterwerte für die nächste Iteration verwendet werden.

Wir betrachten nun den E-Schritt genauer. Die erwartete log-likelihood über alle Wahlen von Z zu berechnen bietet sich an, da Z_{ij} in der Zielfunktion nur linear auftritt und der Erwartungswert ein linearer Operator ist. Es ergibt sich daher

$$\begin{aligned}
f_{\theta^0,\lambda^0,x}(\theta, \lambda) &= \mathbb{E}_{Z \mid (x;\, \theta^0,\lambda^0)} \left[\mathcal{L}_{x,Z}(\theta, \lambda)\right] \\
&= \mathbb{E}_{Z \mid (x;\, \theta^0,\lambda^0)} \left[\sum_{i=1}^{n} \sum_{j=1}^{C} Z_{ij} \cdot \left(\log \lambda_j + \log \mathbb{P}_{\theta_j}(x_i)\right)\right] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{C} \mathbb{E}_{\theta^0,\lambda^0}[Z_{ij} \mid x_i] \cdot \left(\log \lambda_j + \log \mathbb{P}_{\theta_j}(x_i)\right) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{C} Z^0_{ij} \cdot \left(\log \lambda_j + \log \mathbb{P}_{\theta_j}(x_i)\right),
\end{aligned}$$

wobei wir zur Abkürzung $Z^0_{ij} := \mathbb{E}_{\theta^0,\lambda^0}[Z_{ij} \mid x_i]$ gesetzt haben; diesen Wert rechnen wir jetzt aus. Da Z_{ij} eine Indikatorvariable ist, gilt

$$\begin{aligned}
Z^0_{ij} &= \mathbb{E}_{\theta^0,\lambda^0}[Z_{ij} \mid x_i] \\
&= \mathbb{P}_{\theta^0,\lambda^0}(Z_{ij} = 1 \mid x_i) \\
&= \frac{\mathbb{P}_{\theta^0,\lambda^0}(x_i \mid Z_{ij} = 1) \cdot \mathbb{P}_{\theta^0,\lambda^0}(Z_{ij} = 1)}{\mathbb{P}_{\theta^0,\lambda^0}(x_i)} \\
&= \frac{\lambda^0_j \cdot \mathbb{P}_{\theta^0_j}(x_i)}{\sum_{k=1}^{C} \lambda^0_k \cdot \mathbb{P}_{\theta^0_k}(x_i)},
\end{aligned} \tag{1.6}$$

wobei wir den Satz von Bayes für bedingte Wahrscheinlichkeiten angewendet haben; in der letzten Zeile wird wiederum die Wahrscheinlichkeit für x über alle Komponenten gewichtet summiert; es ist ja

$$L_{x_i}(\theta^0, \lambda^0) = \mathbb{P}_{\theta^0,\lambda^0}(x_i) = \sum_{k=1}^{C} \lambda^0_k \cdot \mathbb{P}_{\theta^0_k}(x_i)$$

die Gesamt-likelihood von (θ^0, λ^0) für das Datum x_i.

Die „erwartete Zugehörigkeit“ Z_{ij}^0 von x_i zu Komponente j ergibt sich also als Anteil der gewichteten Wahrscheinlichkeit von x_i aus Komponente j in Bezug auf die Gesamtwahrscheinlichkeit von x_i unter den Parametern (θ^0, λ^0) und kann explizit ausgerechnet werden. Wir können damit auch die aktuelle erwartete log-likelihood $f^0 := f_{\theta^0,\lambda^0,x}(\theta^0, \lambda^0)$ bestimmen.

Im M-Schritt werden die Parameter (θ^*, λ^*) gesucht, die $f_{\theta^0,\lambda^0,x}(\theta, \lambda)$ maximieren. Die Lösung für λ^* lässt sich allgemein angeben. Durch Einführen eines Lagrange'schen Multiplikators zur Wahrung der Bedingung $\sum_{j=1}^{C} \lambda_j = 1$, sowie Bilden und Nullsetzen der Ableitung $\partial f/\partial \lambda_j$ erhält man als Maximierer

$$\lambda_j^* = \frac{1}{n} \sum_{i=1}^{n} Z_{ij}^0,$$

also die (intuitiv sinnvolle) durchschnittliche Zugehörigkeit aller Datenpunkte zu Komponente j.

Die Bestimmung von θ^* ist naturgemäß modellabhängig; allgemein lässt sich immerhin die Ableitung nach dem ℓ-ten Parameter $\theta_{j\ell}$ der j-ten Komponente an der Stelle (θ, λ) berechnen als

$$\frac{\partial f_{\theta^0,\lambda^0,x}}{\partial \theta_{j\ell}}(\theta, \lambda) = \sum_{i=1}^{n} Z_{ij}^0 \cdot \frac{\partial[\log \mathbb{P}_{\theta_j}(x_i)]}{\partial \theta_{j\ell}};$$

diese hängt nicht mehr von λ ab. Zur Berechnung von θ^* müssen wir für das konkrete Modell die Ableitungen nach $\theta_{j\ell}$ berechnen und Null setzen, gegebenenfalls unter Hinzunahme von Lagrange'schen Multiplikatoren bei Nebenbedingungen. Wir bringen an dieser Stelle ein einfaches Beispiel.

Beispiel 1.1 (Gezinkter Würfel)**.** Wir vermuten, dass ein betrügerischer Spieler mit zwei Würfeln arbeitet, einem fairen und einem gezinkten, aber wir wissen nicht, auf welche Weise der gezinkte Würfel gezinkt ist. In einer Runde wird jeweils $d = 20$-mal mit demselben Würfel gewürfelt. Wir beobachten $n = 1000$ Runden und wissen nicht, in welcher Runde welcher Würfel verwendet wurde. Wir wollen die Parameter (Augenwahrscheinlichkeiten) des gezinkten Würfels schätzen.

Offenbar ist $C = 2$ (fairer Würfel und gezinkter Würfel); die Parameter für die erste Komponente stehen sogar mit $\theta_1 = (1/6, \ldots, 1/6)$ schon fest und müssen nicht geschätzt werden. Unbekannt sind die Wahrscheinlichkeiten λ_1 und $\lambda_2 = 1 - \lambda_1$, mit denen jeweils der faire bzw. gezinkte Würfel gewählt wird und natürlich die Parameter $\theta_2 = (\theta_{2,1}, \ldots, \theta_{2,6})$ des gezinkten Würfels. Beobachtet werden die Augenzahlen der Würfe (x_{ik}); nicht beobachtet werden kann die Würfelwahl Z_i für die i-te Runde (hier steht $Z_i = (1, 0)$ für den fairen und $Z_i = (0, 1)$ für den gezinkten Würfel).

Um den EM-Algorithmus zur Parameterschätzung anwenden zu können, müssen wir die Modelle für die beiden Komponenten angeben. Offensichtlich ist für den fairen Würfel

$$\mathbb{P}_{\theta_1}(x_i) = (1/6)^d = (1/6)^{20} \quad \text{für alle } 1 \le i \le n$$

eine Konstante. Für den gezinkten Würfel gilt

$$\mathbb{P}_{\theta_2}(x_i) = \prod_{k=1}^{d} \theta_{2,x_{ik}} = \prod_{q=1}^{6} \theta_{2,q}^{y_{i,q}}, \quad \text{und}$$

$$\log \mathbb{P}_{\theta_2}(x_i) = \sum_{q=1}^{6} y_{i,q} \cdot \log \theta_{2,q},$$

wobei $y_{i,q} := \#\{k : x_{ik} = q\}$ gesetzt wurde (Häufigkeit von Augenzahl q in Versuch i). Jetzt berechnen wir

$$\frac{\partial[\log \mathbb{P}_{\theta_2}(x_i)]}{\partial \theta_{2,q}} = \frac{y_{i,q}}{\theta_{2,q}}.$$

Für den M-Schritt ist die Nebenbedingung $\sum_q \theta_{2,q} = 1$ zu beachten. Die Lagrange-Funktion

$$g(\theta, \lambda, \mu) := f_{\theta^0, \lambda^0, x}(\theta, \lambda) + \mu \cdot \left(1 - \sum_{q=1}^{6} \theta_{2,q}\right)$$

hat die Ableitungen

$$\frac{\partial g}{\partial \theta_{2,q}}(\theta, \lambda, \mu) = -\mu + \sum_{i=1}^{n} Z_{i2}^0 \cdot \frac{y_{i,q}}{\theta_{2,q}} \stackrel{!}{=} 0$$

$$\frac{\partial g}{\partial \mu}(\theta, \lambda, \mu) = 1 - \sum_{q=1}^{6} \theta_{2,q} \stackrel{!}{=} 0$$

Dies ergibt die Lösung $\theta_{2,q}^* = \frac{\sum_{i=1}^n Z_{i,2}^0 y_{i,q}}{\mu}$, wobei μ so gewählt werden muss, dass sich die $\theta_{2,q}$ zu 1 summieren; also $\mu = \sum_{q=1}^6 \sum_{i=1}^n Z_{i,2}^0 \, y_{i,q} = \sum_{i=1}^n Z_{i,2}^0 \sum_{q=1}^6 y_{i,q} = d \cdot \sum_{i=1}^n Z_{i,2}^0$. Das führt auf das Endergebnis

$$\theta_{2,q}^* = \frac{\sum_{i=1}^n Z_{i,2}^0 \cdot y_{i,q}/d}{\sum_{i=1}^n Z_{i,2}^0}, \tag{1.7}$$

das intuitiv Sinn macht: Die Schätzung für $\theta_{2,q}^*$, also für die Wahrscheinlichkeit, mit dem gezinkten Würfel die Augenzahl q zu würfeln, ist der relative Anteil der Beobachtungen von q Augen, die zu Komponente 2 zugeordnet werden, in Bezug auf alle Beobachtungen, die zu Komponente 2 zugeordnet werden. Damit lässt sich jetzt ein EM-Schritt implementieren. ♡

Beispiel 1.2 (Mischung von Gaussverteilungen)**.** Die Daten sind d-dimensionale Punkte $x_i = (x_{i1}, \ldots, x_{id})$, die von C verschiedenen d-dimensionalen Normalverteilungen (Gaussverteilungen) erzeugt werden. Die j-te Normalverteilung wird mit Wahrscheinlichkeit λ_j ausgewählt, einen Punkt x_i zu generieren. Ihr Mittelpunkt sei $\mu_j \in \mathbb{R}^2$; ihre Kovarianzmatrix sei $\sigma_j^2 \mathbb{I}$ (ein skalares Vielfaches der Einheitsmatrix) mit $\sigma > 0$, so dass die Niveaulinien konzentrische Kreise um μ_j bilden. Damit beinhaltet $\theta_j = (\mu_j, \sigma_j)$ die Parameter der j-ten Komponente, und die Dichte ergibt sich als

$$p_{\theta_j}(x_i) = \frac{1}{(2\pi\sigma_j^2)^{d/2}} \cdot \exp\left(-\frac{\|x_i - \mu_j\|^2}{2\sigma_j^2}\right).$$

Obwohl wir jetzt nicht über Wahrscheinlichkeiten, sondern über Dichten sprechen, bleiben die gemachten Aussagen gültig, wenn wir $\mathbb{P}$ (für Wahrscheinlichkeiten) wo nötig durch p (für Dichten) ersetzen.

Zum Ausführen des M-Schritts berechnen wir $\frac{\partial[\log p_{\theta_j}(x_i)]}{\partial \mu_j}$ und $\frac{\partial[\log p_{\theta_j}(x_i)]}{\partial \sigma_j^2}$, setzen die Ableitungen gleich Null und erhalten die intuitiv einleuchtenden Ergebnisse

$$\mu_j^* = \frac{\sum_{i=1}^n Z_{ij}^0 \, x_i}{\sum_{i=1}^n Z_{ij}^0},$$
$$(\sigma_j^2)^* = \frac{\sum_{i=1}^n Z_{ij}^0 \, \|x_i - \mu_j^*\|^2/d}{\sum_{i=1}^n Z_{ij}^0}.$$

Zur Übung wird empfohlen, dies nachzurechnen. ♡

Wir sehen, dass (nach längerem Rechnen) eine Iteration des EM-Algorithmus relativ einfach durchzuführen ist. Dabei werden aus den "alten" Paramteren (θ^0, λ^0) "neue" Parameter (θ^*, λ^*) berechnet, indem eine mit Hilfe der alten Parametern berechnete erwartete log-likelihood (1.5) maximiert wird. Die Frage, die wir bisher nicht beantwortet haben, ist: Hilft dies auch, unser eigentliches Ziel zu erreichen, nämlich die Gesamt-likelihood

$$\mathcal{L}_x(\theta, \lambda) := \log L_x(\theta, \lambda) = \sum_{i=1}^n \log \left[\sum_{j=1}^C \lambda_j \, \mathbb{P}_{\theta_j}(x_i) \right], \tag{1.8}$$

vgl. (1.3), zu maximieren? Die positive Antwort gibt der folgende Satz.

Satz 1.1. *Nach einem EM-Schritt mit $(\theta^0, \lambda^0) \mapsto (\theta^*, \lambda^*)$ gilt*

$$\mathcal{L}_x(\theta^*, \lambda^*) \geq \mathcal{L}_x(\theta^0, \lambda^0);$$

die Gesamt-likelihood verbessert sich also in jedem Schritt.

Beweis. Zu Abkürzung schreiben wir $\phi := (\theta, \lambda)$ und definieren ferner für feste Daten x und gegebene Parameter ϕ^0 die Funktion

$$Q(\phi) := f_{\theta^0, \lambda^0, x}(\theta, \lambda) = \mathbb{E}_{Z \mid (x, \phi^0)} \left[\mathcal{L}_{x,Z}(\phi) \right];$$

diese Funktion wird im M-Schritt des EM-Algorithmus in ϕ maximiert und liefert ϕ^*. Deswegen gilt $Q(\phi^*) \geq Q(\phi^0)$. Genauer ist

$$\begin{aligned} 0 \leq Q(\phi^*) - Q(\phi^0) &= \sum_{i=1}^n \sum_{j=1}^C Z_{ij}^0 \log \frac{\lambda_j^* \, \mathbb{P}_{\theta_j^*}(x_i)}{\lambda_j^0 \, \mathbb{P}_{\theta_j^0}(x_i)} \\ &= \sum_{i=1}^n \sum_{j=1}^C Z_{ij}^0 \log \frac{\lambda_j^* \, \mathbb{P}_{\theta_j^*}(x_i)}{Z_{ij}^0 \, \mathbb{P}_{\phi^0}(x_i)}, \end{aligned}$$

da nach (1.6) genau $Z_{ij}^0 \, \mathbb{P}_{\phi^0}(x_i) = \lambda_j^0 \, \mathbb{P}_{\theta_j^0}(x_i)$. Wir zeigen jetzt, dass die log-likelihood in jedem EM-Schritt um mindestens diese nichtnegative Differenz zunimmt.

Wir betrachten jetzt die log-likelihood-Differenz

$$\begin{aligned}
\mathcal{L}_x(\phi^*) - \mathcal{L}_x(\phi^0) &= \sum_{i=1}^{n} \log \frac{\mathbb{P}_{\phi^*}(x_i)}{\mathbb{P}_{\phi^0}(x_i)} \\
&= \sum_{i=1}^{n} \log \frac{\sum_{j=1}^{C} \lambda_j^* \, \mathbb{P}_{\theta_j^*}(x_i) \cdot \frac{Z_{ij}^0}{Z_{ij}^0}}{\mathbb{P}_{\phi^0}(x_i)} \\
&= \sum_{i=1}^{n} \log \sum_{j=1}^{C} Z_{ij}^0 \, \frac{\lambda_j^* \, \mathbb{P}_{\theta_j^*}(x_i)}{Z_{ij}^0 \, \mathbb{P}_{\phi^0}(x_i)} \\
&\geq \sum_{i=1}^{n} \sum_{j=1}^{C} Z_{ij}^0 \log \frac{\lambda_j^* \, \mathbb{P}_{\theta_j^*}(x_i)}{Z_{ij}^0 \, \mathbb{P}_{\phi^0}(x_i)} \\
&= Q(\phi^*, \phi^0) - Q(\phi^0, \phi^0).
\end{aligned}$$

Die $\geq$-Abschätzung ist die Jensen'sche Ungleichung $\log \sum_j \mu_j v_j \geq \sum_j \mu_j \log v_j$, die für beliebige positive v_j und Wahrscheinlichkeitsverteilungen $\mu = (\mu_j)$ wie z.B. (Z_{ij}^0) direkt aus der Konkavität der log-Funktion folgt ("der Logarithmus einer Konvexkombination ist größer gleich der Konvexkombination der Logarithmen"). □

1.3 Motivsuche mit dem EM-Algorithmus

Nach der allgemeinen Diskussion des EM-Algorithmus für Mehrkomponentenmodelle (und Beispielen, die nichts mit Motivsuche zu tun haben), kehren wir zum Thema Motivsuche zurück. Wir wissen nun, dass sich die Parameter $\theta_1 = (f_{a,k})$ der PFM für $a \in A$ und $1 \leq k \leq L$ und die Parameter $\theta_2 = (f_{a,0})$ des Hintergrundmodells ebenso wie die relativen Modellhäufigkeiten λ_1, λ_2 mit dem EM-Algorithmus iterativ schätzen lassen. Ungeklärt dabei sind noch folgende Fragen, die wir jetzt angehen:

- Worin besteht konkret der M-Schritt?
- Wie werden sinnvolle Startparameter $(\theta_1, \theta_2, \lambda)$ gewählt?
- Wie gehen wir mit der Tatsache um, dass im ursprünglichen Problem Sequenzen und keine unabhängigen L-Tupel gegeben sind; welche Auswirkungen hat die Umformulierung in L-Tupel auf die Ergebnisse?
- Wie kann man mehrere verschiedene Motive finden, die nicht mit bereits gefundenen überlappen?

Ausführung des M-Schritts. Wir müssen für PFM- und Hintergrundmodell noch die ML-Schätzer der neuen Parameter berechnen. Dazu bemerken wir, dass die gleiche Situation vorliegt wie im Beispiel des gezinkten Würfels (Beispiel 1.1). Sei $y_{aik} := 1$, wenn $x_{ik} = a \in A$; sei $y_{aik} := 0$ sonst. Sei ferner $y_{ai0} := \sum_{k=1}^{L} y_{aik}$ die Anzahl der a-Symbole in x_i. Damit lässt

sich leicht ausrechnen, dass

$$f^*_{a,k} = \frac{\sum_{i=1}^n Z^0_{i1}\, y_{aik}}{\sum_{i=1}^n Z^0_{i1}} \qquad \text{in der PFM, und}$$

$$f^*_{a,0} = \frac{\sum_{i=1}^n Z^0_{i2}\, y_{ai0}/L}{\sum_{i=1}^n Z^0_{i2}} \qquad \text{im Hintergrund.}$$

Wahl der Startparameter. Die Parameter $\theta_2 = (f_{a,0})$ des Hintergrundmodells bereiten keine Probleme; hier nimmt man die relativen Symbolhäufigkeiten gemittelt über allen L-Tupel x_i. Ebenso fällt die ad-hoc Wahl von beispielsweise $\lambda = (0.01, 0.99)$ nicht schwer: Der Anteil der Motive an der Gesamtsequenz sollte relativ gering sein. Es bleibt, entweder geeignete $\theta_1 = (f_{a,k})$ für Positionen $1 \leq k \leq L$ und Symbole $a \in A$ auszuwählen, oder aber Indizes i zu wählen, für die $Z_{i1} = 1$ sein soll (also unter den x_i Motivinstanzen zu wählen).

Korrektur für überlappende Motive.

Mehrere verschiedene Motive.

KAPITEL 2

Hinweise zur richtigen Benutzung von LaTeX

2.1 Einleitung

In diesem Kapitel stellen wir einige Hinweise zur richtigen Benutzung von LaTeX bereit. Insbesondere werden Methoden zur Formatierung von Code (Abschnitt 2.2) und häufig gemachte Fehler (Abschnitt 2.3) vorgestellt. Zum Beispiel sollte ein Abschnitt (`\section{}`) normalerweise nicht so kurz sein wie dieser hier.

Zum Nachschlagen von LaTeX-Befehlen empfiehlt sich das Internet oder aber das Buch "LaTeX kurz und gut" von Dalheimer (2005).

2.2 Darstellung von Code

Wir verwenden das `listings`-Paket zur Darstellung von Python-Code. Die Grundeinstellungen werden in der Präambel des Dokuments vorgenommen. Verbesserungsvorschläge sollten mit dem Seminarleiter abgesprochen werden.

Eine *binäre DeBruijn-Sequenz* ist ein String über dem Alphabet $\{0, 1\}$, der alle 2^q möglichen Strings der Länge q als Teilstrings enthält. Er muss daher mindestens die Länge $2^q + q - 1$ haben. In der Tat existiert ein solcher String dieser Länge für jedes $q \geq 0$. Listing 2.1 zeigt eine Funktion, die einen solchen String für gegebenes q berechnet. Listing 2.2 zeigt, wie diese Funktion mit einem über die Kommandozeile übergebenen Parameter aufgerufen wird.

Sehen Sie sich auch den Quellcode dieses Abschnitts an, um zu lernen, wie die Einbindung von Listings funktioniert.

Eine komplette Python-Datei kann so eingebunden werden:

```
\lstinputlisting[
  float=p,
  caption={Python-Funktion, die ...},
  label=python:debruijn]%
  {latexhinweise/debruijn.py}
```

Will man den Code direkt in die LaTeX-Datei schreiben, geht das so:

```
\begin{lstlisting}[%
  float=p,
  caption={Python-Code, um ...},
  label=python:debruijnmain]
...Code...
\end{lstlisting}
```

2.3 Häufig gemachte Fehler

Zu große Abstände nach Satzzeichen. LaTeX setzt hinter einem Punkt einen größeren Abstand als sonst zwischen Wörtern. Normalerweise ist das gewollt, damit zwei Sätze voneinander besser getrennt sind. Es führt aber zu Problemen bei Abkürzungen wie z. B. Prof. Rahmann (beachte die zu große Abstandslänge; LaTeX versucht schlau zu sein und interpretiert einen einzelnen Buchstaben wie `B.` nicht als Satz). Erstens kann LaTeX einen häufig nicht gewollten Zeilenumbruch einfügen; zweitens sollte der Abstand die normale Länge haben.

Will man einen Zeilenumbruch ausschließen, so kann man einen nichtumbrechenden Zwischenraum (non-breaking space) verweden, in LaTeX geht das mit dem Zeichen `~`. Will man den Umbruch zulassen, muss man dem Leerzeichen einen backslash voranstellen:

```
z.~B.\ Prof.~Rahmann
```

verhindert Umbrüche zwischen `z.` und `B.`, sowie swischen `Prof` und `Rahmann`.

Umlaute. Es ist durchaus möglich, im LaTeX-Quellcode Umlaute einzugeben. Es ist auch möglich, dass diese im Dokument korrekt dargstellt werden. Dies ist jedoch abhängig vom akutellen Encoding, und niemand garantiert, dass es bei einem Dokument, das von mehreren Autoren geschrieben wird, einheitlich ist. Daher müssen deutsche Umlaute wir ä, ö oder ß wie folgt geschrieben werden:

```
\"a, \"o oder {\ss}
```

Fehlende Abstände nach Befehlen. Der Befehl `\LaTeX` erzeugt LaTeX. wenn man nun schreibt (wie oben): `\LaTeX setzt`, erhält man: LaTeXsetzt; es fehlt der Abstand! Dasselbe Problem ergibt sich nach allen Befehlen, da das Leerzeichen das Befehlsende markiert.

Zur Lösung kann man entweder den Befehl einklammern, `{\LaTeX} setzt`, oder (einfacher), wieder backslash-space verwenden: `\LaTeX\ setzt`.

Listing 2.1: Python-Funktion, die eine binäre DeBruijn-Sequenz der Ordnung q berechnet

```
def BinaryDeBruijnSequence(q):
    """returns a bit sequence of length 2^q + q - 1
    such that each bit sequence of length q
    is contained exactly once in it.
    """
    zq = 1<<q             # 2^q
    zq1 = 1<<(q-1)        # 2^(q-1)
    L = zq + q - 1        # length of the sequence
    s = bytearray(L)      # the sequence to be constructed
    seen = bytearray(zq) # which values have we seen?
    # first q bits are zero
    val = 0
    i = q
    seen[val]=1
    while(i<L):
        newval = ((val % zq1) << 1) + 1
        if seen[newval]!=0:
            newval = newval - 1
        val = newval
        assert seen[val]==0, "Error"
        s[i] = val & 1
        seen[val] = 1
        i += 1
    return s
```

Listing 2.2: Python-Code, um die Berechnung einer binären DeBruijn-Sequenz aufzurufen

```

def printBinaryDeBruijnString(q):
    s = BinaryDeBruijnSequence(q)
    for c in s:
        print(chr(c+48), sep='', end='')

def main(argv):
    if len(argv)>=1:
        q=int(argv[0])
    else:
        q=6
    printBinaryDeBruijnString(q)

if __name__=="__main__":
    main(sys.argv[1:])
```

Elementare mathematische Funktionen. Variablennamen werden grundsätzlich kursiv geschrieben: i, j, m, n. Dazu wechselt man mit $ in den Mathe-Modus (und auch wieder zurück). Falsch wäre es, hier etwa *italics* text, *i, j, m, n*, zu benutzen (beachte das unterschiedliche Schriftbild).

Konstanten hingegen schreibt man nicht kursiv, insbesondere die imaginäre Einheit $\mathrm{i} = -1$ oder $\mathrm{e} \approx 2.71....$ Im Mathemodus bekommt man das z.B. mit

```
\mbox{\upshape i}
```

hin; eleganter aber ist es, wenn man sich einen eigenen Befehl dafür definiert, zum Beispiel `\imunit` für imaginary unit:

```
\newcommand{\imunit}{\mbox{\upshape i}}
```

Wichtig ist auch, dass Namen elementarer Funktionen wie sin, cos, log nicht kursiv geschrieben werden, ebenso wie Operatorennamen. Dafür stellt LaTeX bereits häufig vordefinierte Befehle zur Verfügung, etwa

`\sin`, `\cos`, `\log`.

Hingegen würde ein vorgebildeter Leser *log* als das Produkt $l \cdot o \cdot g$ interpretieren. Auch das Differential d wird nicht kursiv geschrieben: $\int x \,\mathrm{d}x = x^2/2$. Das wird leider häufig auch in ansonsten guten Büchern falsch gemacht ($\int xdx = x^2/2$ sieht furchtbar aus – beachten Sie die Unterschiede im source code!).

WYSIWYG. Häufig wird der (oft nicht sichtbare) Fehler gemacht, bestimmte Textstellen nur optisch statt logisch auszuzeichnen. Nehmen wir an, dass wir sowohl zu definierende Begriffe als auch Spezies-Namen kursiv hervorheben wollen:

Eine *kontextfreie Grammatik* ist ein 4-Tupel ...
Das Bodenbakterium *C. glutamicum* lebt ...

Wenn wir dies jeweils mit `\textit{}` machen und uns dann entscheiden, dass wir alle zu definierenden Begriffe doch lieber unterstrichen fett haben wollen, bekommen wir ein Problem! Wir müssen jedes `\textit` durchgehen und prüfen, ob es für eine Definition oder anderweitig verwendet wird. Besser ist es, wenn wir eigene Befehle definieren, etwa:

```
\newcommand{\df}[1]{\emph{#1}}
\newcommand{\species}[1]{\textit{#1}}
```

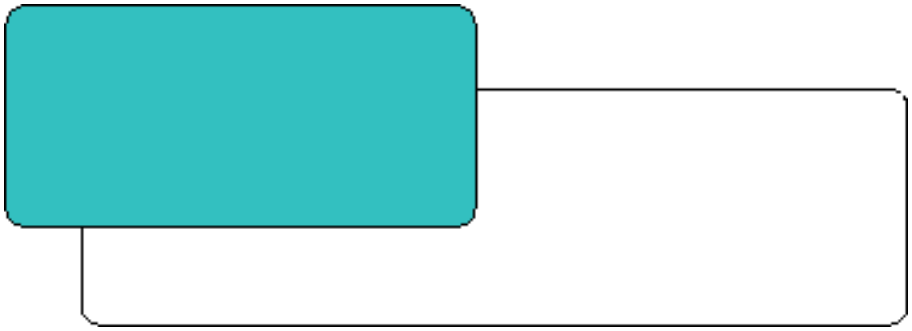

Abbildung 2.1: Ein Kasten

Einbindung von Abbildungen und Tafeln. Keinesfalls sollten Abbildungen und Tafeln direkt in den Text geschrieben werden, da dies im Zweifelsfall zu einem sehr schlechten Seitenumbruch führen kann. Sinnvoller ist es, wenn solche Objekte als frei verschiebbar definiert werden und man Hinweise zur gewünschten Positionierung gibt. Dies geschieht mit Hilfe der `figure` und `table`-Umgebungen.

Wir beschreiben den typischen Fall, das eine Grafik aus einer Datei eingebunden werden soll. Je nachdem, ob mit `latex` nach dvi und dann ps oder mit `pdflatex` direkt in ein pdf übersetzt wird, müssen die Abbildungen als encapsulated postscript (eps) oder als pdf/jpg/png vorliegen. Wenn man beim Dateinamen keine Endung angibt und beide Versionen zur Verfügung stellt, wird die richtige automatisch gewählt.

Der Code zur Einbindung von Abbildung 2.1 sieht so aus:

```
\begin{figure}[t!]
\includegraphics{latexhinweise/kasten}
\caption{\label{fig:kasten}Ein Kasten}
\end{figure}
```

Dabei nehmen wir an, dass Dateien `kasten.eps` und `kasten.png` im aktuellen Verzeichnis existieren.

Kleine Tabellen kann man direkt in den Text einbinden: $\begin{array}{|c|r|}\hline x & 1.20 \\ \hline y & 12.30 \\ \hline \end{array}$. Das ist aber nicht so schön. Man könnte sie zwischen zwei Absätze in eine `center`-Umgebung einfügen, allerdings ergibt sich bei größeren Tabellen wieder das Problem des Seitenumbruchs.

x	1.20
y	12.30

Besser setzt man auch Tabellen beweglich in eine `\table`-Umgebung, etwa mit folgendem Code für Tabelle 2.1:

```
\begin{table}[b!]\centering
\begin{tabular}{l|r}
$x$ &  1.20\\\hline
```

```
$y$ & 12.30\\
\end{tabular}
\caption{\label{tab:daten}Wichtige Daten}
\end{table}
```

Wichtig ist auch, dass jedes `figure` oder `table`-Objekt im Text referenziert werden muss, und wenn es durch ein einfaches (siehe Abbildung 2.1) ist. Der Leser will schließlich wissen, *wann* er auf die Abbildung schauen soll.

Jedes Objekt bekommt mit Hilfe von `\label{}` einen Namen, auf den man sich mit `\ref{}` beziehen kann. Auch die Seite, auf der ein Objekt steht, kann man mit `\pageref{}` ausgeben lassen: `Tabelle~\ref{tab:daten} auf Seite~\pageref{tab:daten}` erzeugt: Tabelle 2.1 auf Seite 16.

Literatur-Referenzen (Zitationen). Verweise auf Werke anderer erfolgen mit `\cite{}` oder `\citep{}`. Da wir das `natbib`-Paket verwenden, erscheinen Literaturverweise nicht als einfache Zahlen, sondern als Kombination von Autor(en) und Erscheinungsjahr. Es ist darauf zu achten, dass der Literaturbezug gut in den Text eingebaut wird. Richtig ist:

- Das Buch "LaTeX kurz und gut" von Dalheimer (2005) enthält eine Übersicht....
- Das Buch "LaTeX kurz und gut" (Dalheimer, 2005) enthält eine Übersicht....
- Dalheimer (2005) beschreibt, wie....

Dabei erzeugt `\cite{}` die Form "Autor (Jahr)" und `\citep{}` die Form "(Autor, Jahr)". Verwenden Sie die natürlich passendere.

x	1.20
y	12.30

Tabelle 2.1: Wichtige Daten

Literaturverzeichnis

M. K. Dalheimer. *LaTeX kurz und gut.* O'Reilly, 2. edition, 2005.