BIN3002 automne 2012 — Devoir 1b

Miklós Csűrös

3 décembre 2012

À remettre avant 23 :59 le 13 décembre par courriel (à csuros@iro. . .). Travaillez en groupes de 2.

1b. Segmentation par taux GC (10 points)

Le but de ce travail pratique est de développer un outil pour diviser une séquence d'ADN en segments selon la fréquence de nucléotides.

Une segmentation de la séquence $x[0..ell-1]$ se définit par la classification de positions $z[0..\ell-1]$. Dans nos modèles probabilistes, x est l'obsérvation de la vecteur de variables aléatoires $X[0..\ell-1]$, et z est la valeur inconnue de la vecteur de v.a. binaires $Z[0..\ell-1]$, où $\forall i\colon Z[i] \in \{\texttt{AT}\uparrow, \texttt{GC}\uparrow\}$. La v.a. $Z[i]$ détermine complètement la distribution de $X[i]$:

$$\mathbb{P}\Big\{X[i]=x \;\Big|\; Z[i]=z\Big\} = p_z(x).$$

On assume les règles de Chargaff :

$$\mathbb{P}\Big\{X[i]=\texttt{T} \;\Big|\; Z[i]=z\Big\} = p_z(\texttt{A})$$
$$\mathbb{P}\Big\{X[i]=\texttt{C} \;\Big|\; Z[i]=z\Big\} = \mathbb{P}\Big\{X[i]=\texttt{G} \;\Big|\; Z[i]=z\Big\} = \frac{1}{2} - p_z(\texttt{A}).$$

Ainsi, le modèle de segmentation a deux paramètres indépendants : $p_{\texttt{AT}\uparrow}(\texttt{A})$ et $p_{\texttt{GC}\uparrow}(\texttt{A})$. La classe $\texttt{AT}\uparrow$ correspond aux segments riches en $\texttt{A}+\texttt{T}$ et la classe $\texttt{GC}\uparrow$ engèndre des segments riches en $\texttt{G}+\texttt{C}$. En conséquence, les probabilités d'émission de A et T sont plus grandes en $\texttt{AT}\uparrow$ qu'en $\texttt{GC}\uparrow$. (Ce TP est basé sur une idée de Sean Eddy.)

Implantation

Implantez la segmentation Viterbi par pénalisation de complexité comme décrite en Note 03 [http://www.iro.umontreal.ca/~csuros/BIN3002/A12/materiel/

`notes03-segmentation.pdf`], en Java. (Si vous préférez un autre langage de programmation, consultez avec moi.) Les arguments de la segmentation comprend les paramètres du modèle (probabilités d'émission $p_{\texttt{AT}\uparrow}(\texttt{A})$ et $p_{\texttt{GC}\uparrow}(\texttt{A})$), et la pénalisation de complexité α (voir Théorème 3.1).

Algorithme de Viterbi

Implantez l'algorithme de Viterbi. La méthode en question a la signature

```
public int[] cheminViterbi(String sequence, double penalite);
```

L'argument `sequence` est la séquence génomique, et `penalite` et la pénalisation d'un segment α. La vecteur z retournée contient des 0s et des 1s : $z[i] = 0$ si le chemin passe par état `AT` $\uparrow$ pour générer le i-ème caractère ($i = 0, 1, \ldots,$ `sequence.length()` $-$ 1). Travaillez avec les logarithmes des probabilités pour éviter l'underflow. (La longueur de la séquence peut être dans les millions.) La séquence contient les caractères `A`, `C`, `G`, `T` (tous majuscules) ; autres caractères sont à ignorer. (Plus précisement, employez la probabilité d'émission $p_z(a) = 1$ quand a est un caractère ambigu comme `N` ou `S`.).

Implantez un algorithme d'apprentissage pour calculer $p_z(\texttt{A})$ dans les deux classes. La méthode `apprentissage` en question prend deux arguments, `int[] chemin` et `String seq`. Elle compte les fréquences de différentes émissions et recalcule les probabilités d'émission :

$$\hat{p}_j(\texttt{A}) = \frac{\sum_{i=0}^{\ell-1}\{x[i] \in \{\texttt{A},\texttt{T}\}; z[i] = j\}}{\sum_{i=0}^{\ell-1}\{z[i] = j\}}$$

Lecture de fichier Fasta

Implantez une classe dédiée à la lecture d'un fichier Fasta et stockage de la séquence génomique. Une solution simple est de servir par la classe `StringBuffer` pendant le lecture. (À la fin de fichier, on retrouve la séquence complète pour passer comme argument à `cheminViterbi` et `apprentissage` par `toString`.)

L'outil

En combinant la classes pour la segmentation et pour la lecture d'un fichier Fasta, implantez une classe executable. La méthode `main` prend cinq arguments : fichier, $p_{\texttt{AT}\uparrow}(\texttt{A})$, $p_{\texttt{GC}\uparrow}(\texttt{A})$ et la pénalité α [dans cet ordre]. Après avoir lu le fichier, et initialisé le modèle, le chemin Viterbi est calculé, et les paramètres du modèle sont optimisés.

```
...
Segmentation M=new Segmentation(p0A,p1A);
int[] chemin=M.cheminViterbi(sequence, penalite);
for (int rep=0; rep<5; rep=++){ // répéter 5 à 10 fois
    M.aprentissage(chemin, sequence);
    chemin=M.cheminViterbi(sequence, penalite);
}
... // (afficher chemin)
```

Le logiciel doit afficher (sur `System.stdout`) la liste des segments où le chemin reste dans la classe `GC` $\uparrow$ pour au moins 50 positions consécutives. Le format de la liste est illustré ici (avec l'exemple d'un autre génome).

```
358764..358940  177     61.0
359972..360044  73      65.7
402967..403055  89      62.9
```

La première colonne donne le début et la fin (séparés par `..`) de chaque segment, la deuxième colonne donne sa longueur, et la troisième colonne donne le pourcentage de $(\mathtt{G}+\mathtt{C})$ dans le segment (arrondi jusqu'à 0.1).

Recherche de gènes ARN

L'organisme *Nanoarchaeum equitans* possède le plus petit génome jamais séquencé (en excluant les viruses). Il est une archée hyperthérmophylique.

► Téléchargez la séquence de son génome (nombre d'accession `NC_005213` en format Fasta (`ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria/Nanoarchaeum_equitans_Kin4_M_uid58009/NC_005213.fna`).

► En utilisant les paramètres initiaux $p_{\mathtt{AT}\uparrow}(\mathtt{A}) = 0.3$, $p_{\mathtt{GC}\uparrow}(\mathtt{A}) = 0.2$, compilez la liste de segments riche en $(\mathtt{G}+\mathtt{C})$ à des pénalités différentes. En particulier, examinez le résultat de pénalisation par AIC et BIC. Essayez d'autres pénalités de magnitude similaire pour voir comment la segmentation change. Inspectez aussi si l'apprentissage montre la convergence (des probabilités d'émission).

► Comparez les segments à la liste de gènes non-codant dans le génome.

Vous avez besoin de la liste des gènes ARNt dans le génome. Le logiciel `tRNAscan-SE` les trouve pour vous (`http://lowelab.ucsc.edu/tRNAscan-SE/`) : installez une copie et annotez le génome. Identifiez les ARNs de transfert trouvés par segmentation. Compilez un tableau qui montre la correspondance entre les segments de votre logiciel et les gènes ARN selon l'annotation. Pour chaque segment qui se chevauche avec un gène ARN, donnez le nom du gène et le début et la longueur du gène. Essayez d'identifier les segments riches en $\mathtt{G}+\mathtt{C}$ qui ne correspondent pas à des ARNt en se servant de l'annotation dans le fichier Genbank `NC_005213.gbk` et/ou de la recherche de similarités par BLASTN (n'oubliez pas de séléctionner «Archaea» parmi les organismes pour limiter la recherche).

Remise de travail

Il faut remettre votre code, la liste de segments trouvés par le logiciel dans le génome de *N. equitans*, ainsi que le tableau de comparaisons (dans un fichier) entre les gènes et les segments, et vos théories concernant la reste des segments.

Références

[1] Klein, R.J., Misulovin, Z., Eddy, S.R. : Noncoding RNA genes identified in AT-rich hyperthermophiles. Proc. Natl. Acad. Sci. USA **99** (2002) 7542–7547

[2] Lowe, T.M., Eddy, S.R. : tRNAscan-SE : a program for improved detection of transfer RNA genes in genomic sequence. Nucleic Acids Res. **25** (1997) 955–964

[3] Schattner, P. : Searching for RNA genes using base composition statistics. Nucleic Acids Res. **30** (2002) 2076–2082

[4] Waters, E., et al. : The genome of Nanoarchaeum equitans : insights into early archaeal evolution and derived parasitism. Proc. Natl. Acad. Sci. USA **100** (2003)