Hidden Markov Models

Hand In 3 - Gene finding
Even more biology

There can be genes in both directions

Gene finding

- Select initial model structure
- Select model parameters by training. Either “by counting” from examples of \((X,Z)\)'s, i.e. genes with known structure, or by EM- or Viterbi-training from examples of \(X\), i.e. sequences which are known to contain a gene.
- Given a new sequence \(X\), predict its gene structure using the Viterbi algorithm for finding the most likely sequence of underlying latent states, i.e. its gene structure

\[ \pi_N = 1 \]
\[ \pi_C = 0 \]
Selecting an initial model
There can be genes in both directions.
We observe

3 typical start-codons

3 stop-codons

We may ignore rare start and stop codons, i.e. a gene that starts (or ends) with a ignored start (or stop) codon does not correspond to a path in your model.
A typical model
In theory, we are given \((X, Z)\) pairs, but we are given \((X, "Z")\) pairs, where the NCR-annotations have to be translated to \(Z\)s.

Also, we may ignore rare start and stop codons, i.e. a gene that starts (or ends) with a ignored start (or stop) codon does not correspond to a path in your model.
Training by counting – Typical solution

To set the transition probabilities:

N → N  N → CCC, where CCC is ATG  N → RRR, where RRR is TTA
N → CCC, where CCC is GTG  N → RRR, where RRR is CTA
N → CCC, where CCC is TTG  N → RRR, where RRR is TCA

We count:

#(N→N) = no. of occurrences of “NN” our annotations.

#(N → CCC, were CCC is XYZ) = no. of occurrence “NCCC” in our annotations, where
CCC is an annotation of XYZ (in our training data).

#(N → RRR, were RRR is XYZ) = no. of occurrence “NRRR” in our annotations where,
RRR is an annotation of XYZ (in our training data).

We compute:

Total = #(N → N) + #(N → CCC where CCC is XYZ) + #(N → RRR, where RRR is XYZ)

We set:

P(N → X) = #(N → X) / Total for each of the 7 transitions
Evaluating performance

**Nucleotide Level**

<table>
<thead>
<tr>
<th></th>
<th>TN</th>
<th>FN</th>
<th>TP</th>
<th>FP</th>
<th>TN</th>
<th>FN</th>
<th>TP</th>
<th>FN</th>
<th>TN</th>
</tr>
</thead>
</table>

**REALITY**

**PREDICTION**

\[
Sn = \frac{TP}{TP + FN} \\
Sp = \frac{TP}{TP + FP}
\]

**Sensitivity**

**Specificity**

\[
CC = \frac{(TP \times TN) - (FN \times FP)}{\sqrt{(TP + FN) \times (TN + FP) \times (TP + FP) \times (TN + FN)}}
\]

**Correlation Coefficient**

\[
ACP = \frac{1}{4} \left[ \frac{TP}{TP + FN} + \frac{TP}{TP + FP} + \frac{TN}{TN + FP} + \frac{TN}{TN + FN} \right]
\]

**Approximate Correlation**

\[
AC = (ACP - 0.5) \times 2
\]
$ python compare_anms.py true-ann6.fa pred-ann6.fa

Cs   (tp=757332, fp=164766, tn=305197, fn=57217): Sn = 0.9298, Sp = 0.8213, AC = 0.6213
Rs   (tp=715865, fp=127462, tn=304830, fn=57584): Sn = 0.9255, Sp = 0.8489, AC = 0.6603
Both (tp=1473197, fp=292228, tn=247613, fn=114801): Sn = 0.9277, Sp = 0.8345, AC = 0.4520
**k-fold cross validation**

For each genome 1 to 5, you train your model by training-by-counting on the remaining 4 genomes, and predict the gene structure on the genome you picked. You compute and report the approximate correlation coefficient (AC) between your prediction and the true annotation using the small python program `compare_anns.py`. Include a table with the computed ACs in your report.

<table>
<thead>
<tr>
<th></th>
<th>Round 1</th>
<th>Round 2</th>
<th>Round 3</th>
<th>Round 4</th>
<th>Round 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genome 1</td>
<td>Validate</td>
<td>Train</td>
<td>Train</td>
<td>Train</td>
<td>Train</td>
</tr>
<tr>
<td>Genome 2</td>
<td>Train</td>
<td>Validate</td>
<td>Train</td>
<td>Train</td>
<td>Train</td>
</tr>
<tr>
<td>Genome 3</td>
<td>Train</td>
<td>Train</td>
<td>Validate</td>
<td>Train</td>
<td>Train</td>
</tr>
<tr>
<td>Genome 4</td>
<td>Train</td>
<td>Train</td>
<td>Train</td>
<td>Validate</td>
<td>Train</td>
</tr>
<tr>
<td>Genome 5</td>
<td>Train</td>
<td>Train</td>
<td>Train</td>
<td>Train</td>
<td>Validate</td>
</tr>
</tbody>
</table>
For each genome 1 to 5, you train your model by training-by-counting on the remaining 4 genomes, and predict the gene structure on the genome you picked. You compute and report the approximate correlation coefficient (AC) between your prediction and the true annotation using the small python program `compare_anns.py`. Include a table with the computed ACs in your report.

Table 1: ACs between the predictions on the genomes 1 to 5 and their true annotation. The gene predictor was trained by the remaining four genomes respectively.

<table>
<thead>
<tr>
<th>Genome</th>
<th>Round 1</th>
<th>Round 2</th>
<th>Round 3</th>
<th>Round 4</th>
<th>Round 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Validate</td>
<td>Train</td>
<td>Train</td>
<td>Train</td>
<td>Train</td>
</tr>
<tr>
<td>2</td>
<td>Train</td>
<td>Validate</td>
<td>Train</td>
<td>Train</td>
<td>Train</td>
</tr>
<tr>
<td>3</td>
<td>Train</td>
<td>Train</td>
<td>Validate</td>
<td>Train</td>
<td>Train</td>
</tr>
<tr>
<td>4</td>
<td>Train</td>
<td>Train</td>
<td>Train</td>
<td>Validate</td>
<td>Train</td>
</tr>
<tr>
<td>5</td>
<td>Train</td>
<td>Train</td>
<td>Train</td>
<td>Train</td>
<td>Validate</td>
</tr>
</tbody>
</table>
Performance on genome 6-10

https://services.birc.au.dk/genefinder-verifier/
Performance on genome 6-10

Input is a fasta-file containing your predictions. It is about 10 Mb big!

$ cat pred-ann6.fa pred-ann7.fa pred-ann8.fa pred-ann9.fa pred-ann10.fa > pred-ann6-10.fa

Beware of the newlines!

pred-annX.fa should end with a newline in order for the above to work
Last year
### ACs between predictions and true annotations sorted by average

<table>
<thead>
<tr>
<th>Genome 6</th>
<th>Genome 7</th>
<th>Genome 8</th>
<th>Genome 9</th>
<th>Genome 10</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.6313</td>
<td>0.6378</td>
<td>0.6007</td>
<td>0.5310</td>
<td>0.4762</td>
</tr>
<tr>
<td>22</td>
<td>0.6313</td>
<td>0.6378</td>
<td>0.6007</td>
<td>0.5310</td>
<td>0.4762</td>
</tr>
<tr>
<td>9</td>
<td>0.6053</td>
<td>0.6147</td>
<td>0.5708</td>
<td>0.4919</td>
<td>0.4197</td>
</tr>
<tr>
<td>10</td>
<td>0.4628</td>
<td>0.4587</td>
<td>0.4245</td>
<td>0.3244</td>
<td>0.3611</td>
</tr>
<tr>
<td>21</td>
<td>0.4619</td>
<td>0.4582</td>
<td>0.4219</td>
<td>0.3239</td>
<td>0.3603</td>
</tr>
<tr>
<td>13</td>
<td>0.4539</td>
<td>0.4547</td>
<td>0.4159</td>
<td>0.3121</td>
<td>0.3650</td>
</tr>
<tr>
<td>11</td>
<td>0.4522</td>
<td>0.4504</td>
<td>0.4172</td>
<td>0.3139</td>
<td>0.3648</td>
</tr>
<tr>
<td>1</td>
<td>0.4520</td>
<td>0.4529</td>
<td>0.4165</td>
<td>0.3122</td>
<td>0.3646</td>
</tr>
<tr>
<td>25</td>
<td>0.4543</td>
<td>0.4515</td>
<td>0.4144</td>
<td>0.3133</td>
<td>0.3596</td>
</tr>
<tr>
<td>18</td>
<td>0.4521</td>
<td>0.4481</td>
<td>0.4112</td>
<td>0.3095</td>
<td>0.3619</td>
</tr>
<tr>
<td>15</td>
<td>0.4501</td>
<td>0.4449</td>
<td>0.4187</td>
<td>0.2974</td>
<td>0.3586</td>
</tr>
<tr>
<td>24</td>
<td>0.4237</td>
<td>0.4326</td>
<td>0.4063</td>
<td>0.2769</td>
<td>0.3816</td>
</tr>
<tr>
<td>14</td>
<td>0.4359</td>
<td>0.4383</td>
<td>0.3974</td>
<td>0.2642</td>
<td>0.3467</td>
</tr>
<tr>
<td>2</td>
<td>0.4286</td>
<td>0.4548</td>
<td>0.3725</td>
<td>0.2473</td>
<td>0.3738</td>
</tr>
<tr>
<td>19</td>
<td>0.3856</td>
<td>0.4075</td>
<td>0.3727</td>
<td>0.2926</td>
<td>0.3059</td>
</tr>
<tr>
<td>23</td>
<td>0.3854</td>
<td>0.4073</td>
<td>0.3551</td>
<td>0.2366</td>
<td>0.3710</td>
</tr>
<tr>
<td>5</td>
<td>0.3850</td>
<td>0.4008</td>
<td>0.3619</td>
<td>0.2559</td>
<td>0.3119</td>
</tr>
<tr>
<td>6</td>
<td>0.3821</td>
<td>0.3914</td>
<td>0.3504</td>
<td>0.2366</td>
<td>0.3263</td>
</tr>
<tr>
<td>16</td>
<td>0.2917</td>
<td>0.3462</td>
<td>0.2618</td>
<td>0.2240</td>
<td>0.3213</td>
</tr>
<tr>
<td>7</td>
<td>0.0247</td>
<td>0.0978</td>
<td>0.0311</td>
<td>0.0642</td>
<td>0.1441</td>
</tr>
<tr>
<td>4</td>
<td>0.0254</td>
<td>0.0939</td>
<td>0.0253</td>
<td>0.0631</td>
<td>0.1387</td>
</tr>
<tr>
<td>20</td>
<td>0.0775</td>
<td>0.0526</td>
<td>0.1344</td>
<td>-0.0246</td>
<td>0.0765</td>
</tr>
<tr>
<td>12</td>
<td>0.0344</td>
<td>0.0820</td>
<td>0.0089</td>
<td>0.0549</td>
<td>0.1144</td>
</tr>
<tr>
<td>17</td>
<td>-0.2748</td>
<td>-0.2571</td>
<td>-0.2635</td>
<td>-0.2657</td>
<td>-0.3075</td>
</tr>
</tbody>
</table>

**Codon-models**

Models inspired by the model presented in class maybe allowing more start/stop codons.

**Problems with training or prediction**
C: coding left-to-right

A: >0
C: >0
G: >0
T: >0

N: Non-coding

R: coding right-to-left
What makes it possible to distinguish between coding and non-coding regions is the difference in nucleotide frequencies.

The explicit model of start- and stop-codons makes sure that we only make “syntactically” correct transitions between coding and non-coding regions.
A “codon” model
More complex gene finding problems

Eukaryotic gene structure
Eukaryotic gene structure in both directions (GenScan)
Exam

**Overall:** Oral, 20 min all included, 10 min for your **prepared** presentation, 5 min for discussion/question, you may be interrupted along the way. See Blackboard for all the details.

**HMM Question:** Hidden Markov Models (Basic algorithms and applications, Building models and selecting model parameters)

What is a HMM, explain/show parameters and assumptions

How to compute the joint probability $P(X,Z)$

Basic problems: Decoding and selecting models parameter, e.g:

How to compute a Viterbi decoding, $Z^*$, where $P(Z^*) = \max_z P(X,Z)$

How to compute $P(X) = \sum_z P(X,Z)$ using the forward algorithm

How to compute a posterior decoding using the forward and backward algorithms

How to select parameters from $(X,Z)$ using training-by-counting, or from $X$ using EM

Remember to be **precise**, and remember **what, why, and how**, when you explain