BINF6201/8201

Hidden Markov Models for Sequence Analysis 1

11-12-2013
Pattern recognition by machine learning algorithms

- Machine learning algorithms are a class of statistics-based algorithms that recognize patterns in data by first leaning the patterns from known examples using a statistical model.

- The process of leaning patterns in data is called the model training.

- To train and evaluate the model and algorithm, we typically divide the known examples into two sets: training dataset and test dataset.

- We first train the algorithm on the training dataset, and then test the algorithm on the test dataset.

- The development of a machine learning method involves the following steps:

  1. Understanding the problem
  2. Build the model
  3. Train the model
  4. Test the model
  5. Apply the model
A simple machine learning algorithm

- We build the model based on our understanding of the problem to be solved.

- Suppose that we want to predict transmembrane helices (TM) in proteins using their sequences.

- What do we know about TM helices?
  1. A TM typically consists of a stretch of 18-30 amino acids made predominately of hydrophobic residues;
  2. TM helices are linked by loops made largely of hydrophilic residues;
  3. Some amino acids in a TM helix are preferably followed by certain other amino acids.
A simple machine learning algorithm

- Our first pattern recognition algorithm will only use the first two facts about TM helices for the prediction.
- Let the frequency for amino acid \( a \) in a TM helix be \( p_a \) and that in a loop \( q_a \).
- We assume that the distributions of \( p_a \) and \( q_a \) are different, so they can be used for TM and loop predictions.
- We say that the distributions of \( p_a \) and \( q_a \) describe TM and loops, and they are the models for TM and loops, respectively. Let us denote them by \( M_1 \) and \( M_0 \), respectively.
- Given a segment of amino acid sequence of length \( L \), \( x = x_1x_2 \ldots x_L \), the likelihoods that this sequence is a TM or a loop are,

\[
L(x | M_1) = \prod_{i=1}^{L} p_{x_i} \quad \text{or} \quad L(x | M_0) = \prod_{i=1}^{L} q_{x_i}, \text{ respectively.}
\]
A simple machine learning algorithm

- We can use the logarithm of the odds ratio of these two likelihoods to predict whether the sequence is a TM helix or a loop:

\[
S(x) = \ln \frac{L(x \mid M_1)}{L(x \mid M_0)} = \ln \prod_{i=1}^{L} \frac{p_{x_i}}{q_{x_i}} = \sum_{i=1}^{L} \ln \frac{p_{x_i}}{q_{x_i}}.
\]

- The higher the \( S \) value, the more likely the sequence forms a helix; the lower the \( S \) value, the more likely the sequence forms a loop.

- Now, we need to train the models by finding the values of the parameters \( p_a \) and \( q_a \) using two training datasets.

- For this purpose, we collect two training datasets as follows:

1. \( D_1 \) : Known TM in proteins in the protein data bank (PDB) database;

2. \( D_0 \) : Known loops between TMs in proteins in the PDB database.
A simple machine learning algorithm

- If there are \( n_a \) number of amino acid \( a \) in the training dataset \( D \), and the total number of amino acids in the dataset \( D \) is \( n_{\text{tot}} \), then the likelihood for this to happen according to the model \( M \) is,

\[
L(D \mid M) = \prod_{a=1}^{20} p_a^{n_a}.
\]

- We train the model \( M \) by maximizing this likelihood function, this is called the principle of maximum likelihood.

- It can be shown that when \( p_a = \frac{n_a}{n_{\text{not}}} \), \( L(D \mid M) = \prod_{a=1}^{20} p_q^{n_a} \) is the maximum. That is,

\[
\hat{p}_a = \arg \max p_a \quad L(D \mid M) = \arg \max p_a \prod_{a=1}^{20} p_a^{n_a} \Rightarrow \hat{p}_a = \frac{n_a}{n_{\text{not}}}.
\]

- After we train both \( M_1 \) and \( M_0 \), we can use the log likelihood odds ratio function \( S(x) \) defined earlier for sequence alignment to predict TM or loops.

- However, to apply this algorithm to a sequence, we have to use a sliding window of arbitrary width \( L \).
Bayesian inference

Sometimes, we are more interested in question as “what is the probability that an alternative model $M_j$ can be described by the observed dataset $D$, i.e., the posterior probability $P(M_j | D)$.

If we have $K$ alternative models to explain the data, then using the Bayer’s theorem, we have,

$$P(M_j | D) = \frac{P(D | M_j)P(M_j)}{P(D)} = \frac{P(D | M_j)P(M_j)}{\sum_{i=1}^{K} P(D | M_i)P(M_i)} = \frac{L(D | M_j)P(M_j)}{\sum_{i=1}^{K} L(D | M_i)P(M_i)}$$

If there are only two alternative models, $M_1$ and $M_0$, then, we have,

$$P(M_1 | D) = \frac{P(D | M_1)P(M_1)}{L(D | M_0)P(M_0) + L(D | M_1)P(M_1)}$$

Similarly, we can define posterior odds ratio function as,

$$S' = \ln \frac{L(M_1 | D)}{L(M_0 | D)} = \ln \frac{L(D | M_1)P(M_1)}{L(D | M_0)P(M_0)} = \ln \frac{L(D | M_1)}{L(D | M_0)} + \ln \frac{P(M_1)}{P(M_0)} = S + \ln \frac{P(M_1)}{P(M_0)}.$$
Markov chain models for correlations in sequences

- To predict TM helices more accurately, we might want to consider the dependence between the adjacent amino acids, e.g. the correlations between two, three or even more adjacent residues.

- As a simple example, to predict CpG islands in a genome sequence, we need to consider the dependency of two adjacent nucleotides.

- We can model such dependency using a **first order Markov chain**, similar to our nucleotide evolutionary substitution models.

- To predict CpG islands, we can construct the following four-state Markov model \( M \), which is described by a 4 x 4 **state transition probability matrix**.

  \[
  \begin{pmatrix}
  a_{AA} & a_{AG} & a_{AC} & a_{AT} \\
  a_{GA} & a_{GG} & a_{GC} & a_{GT} \\
  a_{CA} & a_{CG} & a_{CC} & a_{CT} \\
  a_{TA} & a_{TG} & a_{TC} & a_{TT}
  \end{pmatrix}
  \]

  Where, \( i \) is the position of a nucleotide in the sequence.

  Sequence \( x=x_1x_2x\ldots x_{i-1}x_i\ldots x_L \)

  Transition probability \( a_{st} = P(x_i=t|x_{i-1}=s) \), where, \( i \) is the position of a nucleotide in the sequence.
Markov chain models for correlations in sequences

When the Markov chain model is given, then the probability (likelihood) that a given sequence \(x\) is generated by the model is,

\[
P(x) = P(x_1, x_2, \ldots, x_{L-1}, x_L) = P(x_L, x_{L-1}, \ldots, x_2, x_1)
= P(x_L | x_{L-1}, \ldots, x_2, x_1) P(x_{L-1}, \ldots, x_2, x_1)
= P(x_L | x_{L-1}, \ldots, x_2, x_1) P(x_{L-1} | x_{L-2}, \ldots, x_2, x_1) \ldots P(x_2 / x_1) P(x_1)
= P(x_L | x_{L-1}) P(x_{L-1} | x_{L-2}) \ldots P(x_2 / x_1) P(x_1) = P(x_1) \prod_{i=2}^{L} a_{x_{i-1}x_i}.
\]

The first nucleotide can be any type, so we introduce a start state \(B\), and transition probabilities, \(a_{Bs}\). To model the end of the sequence, we introduce the end state \(E\), and transitional probabilities \(a_{tE}\). i.e.,

\[
P(x_1 = s | B) = a_{Bs} \quad \text{and} \quad P(E | x_L = t) = a_{tE}.
\]
Markov chain models for correlations in sequences

- To predict CpG island more effectively, and also to predict non-CpG island sequences at the same time, we construct a Markov chain model for non-CpG island sequences, let’s call this model $M_0$, and rename the model for CpG island as $M_1$.

- Then we can compute the log likelihood ratio as,

$$S(x) = \log \frac{P(x \mid M_1)}{P(x \mid M_0)}$$

$$= \log \frac{P(x_1) \prod_{i=2}^{L} a_{x_i \mid x_{i-1}}^{(1)}}{P(x_1) \prod_{i=2}^{L} a_{x_i \mid x_{i-1}}^{(0)}} = \sum_{i=2}^{L} \log \frac{a_{x_i \mid x_{i-1}}^{(1)}}{a_{x_i \mid x_{i-1}}^{(0)}}$$

$$= \sum_{i=2}^{L} \beta_{x_{i-1} x_i}.$$

- If nucleotides in different sites are independent, we have,

$$a_{x_i \mid x_{i-1}}^{(1)} = p_{x_i} \quad \text{and} \quad a_{x_i \mid x_{i-1}}^{(0)} = q_{x_i},$$
then the Markov chain models reduce to our previous simple models.
Markov chain models for correlations in sequences

➢ To train the models, we collect two sets of sequences, one contains known CpG islands and the other contains sequences that are not CpG islands.

➢ Let \( n_{st} \) be the number of times nucleotide \( t \) follows nucleotide \( s \), and \( n_s \) the total number of nucleotide \( s \) in the segment of sequence. Using the principle of maximal likelihood, the transition probabilities are,

\[
\hat{a}_{st}^{(1)} = \arg \max_{a_{st}^{(1)}} \prod_{s,t} (a_{st})^{n_{st}^{(1)}} = \frac{n_{st}^{(1)}}{\sum_{t'} n_{st'}^{(1)}}, \text{ for the CpG island model } M_1, \text{ and}
\]

\[
\hat{a}_{st}^{(0)} = \arg \max_{a_{st}^{(0)}} \prod_{s,t} (a_{st})^{n_{st}^{(0)}} = \frac{n_{st}^{(0)}}{\sum_{t'} n_{st'}^{(0)}}, \text{ for the non-CpG island model } M_0.
\]

Training results based on 47 sequences in each dataset

\[
\begin{array}{cccccc}
M_1 & + & A & C & G & T \\
\hline
A & 0.180 & 0.274 & 0.426 & 0.120 \\
C & 0.171 & 0.368 & 0.274 & 0.188 \\
G & 0.161 & 0.339 & 0.375 & 0.125 \\
T & 0.079 & 0.355 & 0.384 & 0.182 \\
\end{array}
\]
Markov chain models for correlations in sequences

- The values of $\beta$ are as follows.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-0.740</td>
<td>0.419</td>
<td>0.580</td>
<td>-0.803</td>
</tr>
<tr>
<td>C</td>
<td>-0.913</td>
<td>0.302</td>
<td>1.812</td>
<td>-0.685</td>
</tr>
<tr>
<td>G</td>
<td>-0.624</td>
<td>0.461</td>
<td>0.331</td>
<td>-0.730</td>
</tr>
<tr>
<td>T</td>
<td>-1.169</td>
<td>0.573</td>
<td>0.393</td>
<td>-0.679</td>
</tr>
</tbody>
</table>

- The distributions of the log likelihood score $S(x)$ for the CpG islands and non-CpG sequences are well separated by the models on the test datasets.

- Dark: CpG
- Light: non-CpG
- The scores were normalized to the length of sequences.
Hidden Markov models

- There is still a serious limitation in Markov chain models: they cannot model the length of sequence in a natural way.

- If we want to predict CpG islands in a long sequence, we have to use a window of fixed length to scan the sequence, and predict that a window is a CpG island if the score is high enough, but CpG islands have varying length.

- Ideally, we want to model the sequence pattern and the length of the pattern simultaneously. A variant of Markov chain models called hidden Markov models (HMMs) can achieve such a purpose.

- In a HMM, when the process transits from one state $k$ to another state $l$, $l$ will emit a letter $b$ according to a probability, $e_l(b)$, which is called an emission probability.
Formal definition of a HMM

- A HMM contains the following four components:
  1. **A set of N states** $S=\{S_1, S_2, ..., S_N\}$ that specify what state a process can be in; a start and an end state can be added to the state set.
  2. **Transition probability matrix** $a_{kl}$ that specifies how likely the process transits from one state $k$ to another state $l$. The sequence of states that the process goes through, $\pi = \pi_1 \pi_2, ..., \pi_L$, is called a path. Thus a path is simply a Markov chain.
  3. **A set of letters or symbols** $V= \{b_1, b_2, ..., b_V\}$ that can be emitted in a state.
  4. **Emission probability matrix** $e_l(b)$ that specifies how likely a letter $b$ is emitted in a state $l$. The sequence of letters emitted in the process, $x=x_1 x_2, ..., x_L$, is called an **observation sequence**.

- Usually, the observation sequence is observable, but the path is not; it is for this reason that we call the model a **hidden** Markov model.

- Formally we define a HMM as a quadruple, $M=\{S, V, a_{kl}, e_l(b)\}$, where,

  \[ a_{kl} = P(\pi_i = l \mid \pi_{i-1} = k), \quad \text{and} \quad e_l(b) = P(x_i = b \mid \pi_i = l). \]
Example 1: the occasionally dishonest casino

A dishonest casino alternates between a fair and a loaded die, the process can be modeled by a four-state HMM.

In this case, we see the outcome of each roll of a die. If they play \( L \) times, we will see a sequence of \( L \) symbols (observation sequence), \( x=x_1x_2,...x_L \), but we do not know which die has been used in each roll, i.e., the sequence of dices being used (the path of states), \( \pi=\pi_1\pi_2,...\pi_L \), is hidden from us.
Example 2: prediction of TM helices and loops

The alternation between TM helices and loops in a protein sequence can be also modeled by a four-state HMM.

Here, we know the protein sequences (the output of model) \( x=x_1x_2,...x_L \), but we do not know which state (helix or loop) generates each of the residues. If we can decode the path \( \pi=\pi_1\pi_2,...\pi_L \) that generates the sequence, then we can predict the locations of TM helices and loops in the sequence.
A HMM for CpG islands

\[ M_1 \rightarrow a_{11} \rightarrow B \rightarrow a_{B1} \rightarrow M_2 \rightarrow a_{21} \rightarrow E \rightarrow a_{1E} \rightarrow M_1 \]

\[ M_2 \rightarrow a_{12} \rightarrow B \rightarrow a_{B2} \rightarrow M_2 \rightarrow a_{22} \rightarrow E \rightarrow a_{2E} \rightarrow M_2 \]