CS481: Bioinformatics Algorithms

Can Alkan
EA224
calkan@cs.bilkent.edu.tr

http://www.cs.bilkent.edu.tr/~calkan/teaching/cs481/

HMM Parameter Estimation

- So far, we have assumed that the transition and emission probabilities are known.
- However, in most HMM applications, the probabilities are not known. It's very hard to estimate the probabilities.

HMM Parameter Estimation Problem

- Given
  - HMM with states and alphabet (emission characters)
  - Independent training sequences $x^1, \ldots x^m$
- Find HMM parameters Θ (that is, a_{kl}, $e_k(b)$) that maximize

$$P(x^1, \ldots, x^m \mid \Theta)$$

the joint probability of the training sequences.

Maximize the likelihood

$P(x^1, \ldots, x^m \mid \Theta)$ as a function of Θ is called the likelihood of the model.

The training sequences are assumed independent, therefore

$$P(x^1, \ldots, x^m \mid \Theta) = \Pi_i P(x^i \mid \Theta)$$

The parameter estimation problem seeks Θ that realizes

$$\max_{\Theta} \prod_{i} P(x^i \mid \Theta)$$

In practice the log likelihood is computed to avoid underflow errors

Two situations

Known paths for training sequences

- CpG islands marked on training sequences
- One evening the casino dealer allows us to see when he changes dice

Unknown paths

- CpG islands are not marked
- Do not see when the casino dealer changes dice

Known paths

A_{kl} = # of times each $k \rightarrow l$ is taken in the training sequences

$E_k(b)$ = # of times b is emitted from state k in the training sequences

Compute a_{kl} and $e_k(b)$ as maximum likelihood estimators:

$$a_{kl} = A_{kl} / \sum_{l'} A_{kl'}$$

$$e_k(b) = E_k(b) / \sum_{b'} E_k(b')$$

Pseudocounts

- Some state k may not appear in any of the training sequences. This means $A_{kl} = 0$ for every state l and a_{kl} cannot be computed with the given equation.
- To avoid this overfitting use predetermined pseudocounts r_{kl} and $r_k(b)$.

$$A_{kl} = \text{\# of transitions } k \rightarrow l + r_{kl}$$

$$E_k(b) = \text{\# of emissions of } b \text{ from } k + r_k(b)$$

The pseudocounts reflect our prior biases about the probability values.

Unknown paths: Viterbi training

Idea: use Viterbi decoding to compute the most probable path for training sequence x

- Start with some guess for initial parameters and compute π^* the most probable path for x using initial parameters.
- Iterate until no change in π^* :

1. Determine A_{kl} and $E_k(b)$ as before
2. Compute new parameters a_{kl} and $e_k(b)$ using the same formulas as before
3. Compute new π^* for x and the current parameters

Viterbi training analysis

- The algorithm converges precisely

  There are finitely many possible paths.

  New parameters are uniquely determined by the current π^*.

  There may be several paths for x with the same probability, hence must compare the new π^* with all previous paths having highest probability.
- Does not maximize the likelihood $\Pi_x P(x \mid \Theta)$ but the contribution to the likelihood of the most probable path $\Pi_x P(x \mid \Theta, \pi^*)$
- In general performs less well than Baum-Welch

Unknown paths: Baum-Welch

Idea:

1. Guess initial values for parameters.

   art and experience, not science

2. Estimate new (better) values for parameters.

   how ?

3. Repeat until stopping criteria is met.

   what criteria ?

Better values for parameters

Would need the A_{kl} and $E_k(b)$ values but cannot count (the path is unknown) and do not want to use a most probable path.

For all states *k,l,* symbol *b* and training sequence *x*

Compute A_{kl} and $E_k(b)$ as expected values, given the current parameters

Notation

For any sequence of characters x emitted along some <u>unknown path</u> π, denote by $\pi_i = k$ the assumption that the state at position i (in which x_i is emitted) is k.

Probabilistic setting for $A_{k,l}$

Given $x^1, \ldots, x^m$ consider a discrete probability space with elementary events

$\varepsilon_{k,l,}$ = "$k \rightarrow l$ is taken in $x^1, \ldots, x^m$"

For each x in $\{x^1, \ldots, x^m\}$ and each position i in x let $Y_{x,i}$ be a random variable defined by

$$Y_{x,i}(\varepsilon_{k,l}) = \begin{cases} 1, \; if \;\; \pi_i = k \;\; and \;\; \pi_{i-1} = l \\ 0, \; otherwise \end{cases}$$

Define $Y = \Sigma_x \Sigma_i Y_{x,i}$ random var that counts # of times the event $\varepsilon_{k,l}$ happens in $x^1, \ldots, x^m$.

The meaning of A_{kl}

Let A_{kl} be the expectation of Y

$$E(Y) = \Sigma_x \Sigma_i E(Y_{x,i}) = \Sigma_x \Sigma_i P(Y_{x,i} = 1) =$$
$$\Sigma_x \Sigma_i P(\{\varepsilon_{k,l} \mid \pi_i = k \text{ and } \pi_{i+1} = l\}) =$$
$$\Sigma_x \Sigma_i P(\pi_i = k, \pi_{i+1} = l \mid x)$$

Need to compute $P(\pi_i = k, \pi_{i+1} = l \mid x)$

Probabilistic setting for $E_k(b)$

Given $x^1, \ldots, x^m$ consider a discrete probability space with elementary events

$$\varepsilon_{k,b} = \text{“}b \text{ is emitted in state } k \text{ in } x^1, \ldots, x^m\text{”}$$

For each x in $\{x^1,\ldots,x^m\}$ and each position i in x let $Y_{x,i}$ be a random variable defined by

$$Y_{x,i}(\varepsilon_{k,b}) = \begin{cases} 1, \ if \ \ x_i = b \ and \ \pi_i = k \\ 0, \ otherwise \end{cases}$$

Define $Y = \Sigma_x \Sigma_i Y_{x,i}$ random var that counts # of times the event $\varepsilon_{k,b}$ happens in $x^1,\ldots,x^m$.

The meaning of $E_k(b)$

Let $E_k(b)$ be the expectation of Y

$$E(Y) = \Sigma_x \Sigma_i E(Y_{x,i}) = \Sigma_x \Sigma_i P(Y_{x,i} = 1) =$$
$$\Sigma_x \Sigma_i P(\{\varepsilon_{k,b} \mid x_i = b \text{ and } \pi_i = k\})$$

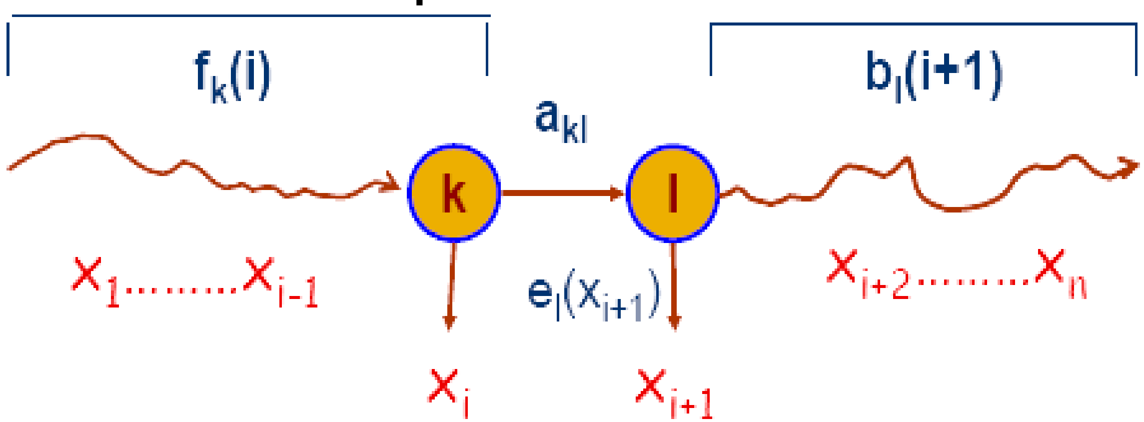
$$\sum_x \sum_{\{i|x_i=b\}} P(\{\varepsilon_{..,b} \mid x_i = b, \pi_{.} = k\}) = \sum_x \sum_{\{i|x_i=b\}} P(\pi_{.} = k \mid x)$$

Need to compute $P(\pi_i = k \mid x)$

Computing new parameters

Consider $x = x_1 \ldots x_n$ training sequence

Concentrate on positions i and $i+1$



Use the forward-backward values:

$$f_{ki} = P(x_1 \ldots x_i\, ,\, \pi_i = k)$$

$$b_{ki} = P(x_{i+1} \ldots x_n \mid \pi_i = k)$$

Compute A_{kl} (1)

Prob $k \rightarrow l$ is taken at position i of x

$$P(\pi_i = k, \pi_{i+1} = l \mid x_1 \ldots x_n) = P(x, \pi_i = k, \pi_{i+1} = l) / P(x)$$

Compute $P(x)$ using either forward or backward values

[next slide] $P(x, \pi_i = k, \pi_{i+1} = l) = b_{li+1} \cdot e_l(x_{i+1}) \cdot a_{kl} \cdot f_{ki}$

Expected # times $k \rightarrow l$ is used in training sequences

$$A_{kl} = \Sigma_x \Sigma_i (b_{li+1} \cdot e_l(x_{i+1}) \cdot a_{kl} \cdot f_{ki}) / P(x)$$

Compute A_{kl} (2)

$P(x, \pi_i = k, \pi_{i+1} = l) =$

$P(x_1 \ldots x_i, \pi_i = k, \pi_{i+1} = l, x_{i+1} \ldots x_n) =$

$P(\pi_{i+1} = l, x_{i+1} \ldots x_n \mid x_1 \ldots x_i, \pi_i = k) \cdot P(x_1 \ldots x_i, \pi_i = k) =$

$P(\pi_{i+1} = l, x_{i+1} \ldots x_n \mid \pi_i = k) \cdot f_{ki} =$

$P(x_{i+1} \ldots x_n \mid \pi_i = k, \pi_{i+1} = l) \cdot P(\pi_{i+1} = l \mid \pi_i = k) \cdot f_{ki} =$

$P(x_{i+1} \ldots x_n \mid \pi_{i+1} = l) \cdot a_{kl} \cdot f_{ki} =$

$P(x_{i+2} \ldots x_n \mid x_{i+1}, \pi_{i+1} = l) \cdot P(x_{i+1} \mid \pi_{i+1} = l) \cdot a_{kl} \cdot f_{ki} =$

$P(x_{i+2} \ldots x_n \mid \pi_{i+1} = l) \cdot e_l(x_{i+1}) \cdot a_{kl} \cdot f_{ki} =$

$b_{li+1} \cdot e_l(x_{i+1}) \cdot a_{kl} \cdot f_{ki}$

Compute $E_k(b)$

Prob x_i of x is emitted in state k

$P(\pi_i = k \mid x_1 \ldots x_n) = P(\pi_i = k,\ x_1 \ldots x_n)/P(x)$

$P(\pi_i = k,\ x_1 \ldots x_n) = P(x_1 \ldots x_i, \pi_i = k, x_{i+1} \ldots x_n) =$
$P(x_{i+1} \ldots x_n \mid x_1 \ldots x_i, \pi_i = k) \cdot P(x_1 \ldots x_i, \pi_i = k) =$
$P(x_{i+1} \ldots x_n \mid \pi_i = k) \cdot f_{ki} = b_{ki} \cdot f_{ki}$

Expected # times b is emitted in state k

$$E_k(b) = \sum_x \sum_{i:x_i=b} f_{ki} \cdot b_{ki} \Big/ P(x)$$

Finally, new parameters

$$a_{kl} = A_{kl} / \sum_{l'} A_{kl'}$$

$$e_k(b) = E_k(b) / \sum_{b'} E_k(b')$$

Can add pseudocounts as before.

Stopping criteria

Cannot actually reach maximum (optimization of continuous functions)

Therefore need stopping criteria

- Compute the log likelihood of the model for current Θ

$$\sum_x \log P(x \mid \Theta)$$

  Compare with previous log likelihood

  Stop if small difference

- Stop after a certain number of iterations

The Baum-Welch algorithm

Initialization:

Pick the best-guess for model parameters (or arbitrary)

Iteration:

1. Forward for each x
2. Backward for each x
3. Calculate A_{kl}, $E_k(b)$
4. Calculate new a_{kl}, $e_k(b)$
5. Calculate new log-likelihood

Until log-likelihood does not change much

Baum-Welch analysis

- Log-likelihood is increased by iterations

  Baum-Welch is a particular case of the EM (expectation maximization) algorithm
- Convergence to local maximum. Choice of initial parameters determines local maximum to which the algorithm converges