Chap. 3 Markov chains and hidden Markov models (2)

Biointelligence Laboratory
School of Computer Sci. & Eng.
Seoul National University
Seoul 151-742, Korea

This slide file is available online at
http://bi.snu.ac.kr/

Specifying the Model (HMM)

- The design of the structure
  - What states there are and how they are connected.
- The assignment of parameter values
  - The transition and emission probabilities, a_{kl} and $e_k(b)$.

The Framework for Parameter Estimation

- We have a set of *training sequences* that we want the model to fit well.
- Given the independent training sequences,
  - $x^1, \ldots, x^n$.
  - The log probability of these sequences given the model and its parameters θ (the log likelihood of the model).

$$l(x^1,...,x^n) = \log P(x^1,...,x^n \mid \theta) = \sum_{j=1}^{n} \log P(x^j \mid \theta)$$

  - The parameter value which maximizes the above probability (the log likelihood) is chosen.

Estimation when the State Sequence is Known

- It is easier to estimate the probability parameters when the state paths are known for all the training examples.
  - Given a set of genomic sequences in which the CpG islands were already labeled, based on the experimental data.
  - An HMM for the prediction of secondary structure of proteins, with training sequences obtained from the set of proteins with known structures.
  - An HMM predicting genes from genomic sequences, where the transcript structure has been determined by cDNA sequencing

The Maximum Likelihood Estimator

- The maximum likelihood estimator
  - Calculate the number of times in the training sequences for each transition (A_{kl}) and each emission ($E_k(b)$).

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \qquad \text{and} \qquad e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}$$

- In the case of insufficient data (overfitting)
  - A_{kl} = number of transitions k to l in training data + r_{kl}
  - $E_k(b)$ = number of emissions of b from k in training data + $r_k(b)$

Prior knowledge from users (pseudocounts)

(Cf. Dirichlet priors in Bayesian statistics)

Estimation when Paths are Unknown

- All the standard algorithms for optimization of continuous functions can be used.
- The Baum-Welch algorithm (informal description)
  1. Estimate the A_{kl} and $E_k(b)$ by considering probable paths for the training sequences using the current values of a_{kl} and $e_k(b)$.
  2. Estimated A_{kl}'s and $E_k(b)$'s are used for the update of a_{kl}'s and $e_k(b)$'s.
  3. Above process is iterated until some stopping criterion is reached.
- The overall log likelihood of the model is increased by the iteration → local maxima
  - There exist many local maxima. (strongly depends on the starting point of the algorithm.)

Estimation of the A_{kl} Value

- The probability that a_{kl} is used at position i in sequence x:

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

$$= \frac{P(x_1,...,x_i, \pi_i = k \mid \theta) P(x_{i+1},...,x_L, \pi_{i+1} = l \mid x_1,...,x_i, \pi_i = k, \theta)}{P(x \mid \theta)}$$

$$= \frac{P(x_1,...,x_i, \pi_i = k \mid \theta) P(x_{i+1},...,x_L, \pi_{i+1} = l \mid \pi_i = k, \theta)}{P(x \mid \theta)}$$

$$= \frac{P(x_1,...,x_i, \pi_i = k \mid \theta) P(x_{i+1}, \pi_{i+1} = l \mid \pi_i = k, \theta) P(x_{i+2},...,x_L \mid x_{i+1}, \pi_{i+1} = l, \pi_i = k, \theta)}{P(x \mid \theta)}$$

$$= \frac{P(x_1,...,x_i, \pi_i = k \mid \theta) P(x_{i+1}, \pi_{i+1} = l \mid \pi_i = k, \theta) P(x_{i+2},...,x_L \mid \pi_{i+1} = l, \theta)}{P(x \mid \theta)}$$

$$= \frac{f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)}{P(x)}.$$

Estimation of the $E_k(b)$ Value

- The probability that $e_k(b)$ is used at position i in sequence x:

$$P(\pi_i = k, x_i = b \mid x, \theta) = \frac{P(\pi_i = k, x_i = b, x \mid \theta)}{P(x \mid \theta)}$$

$$= \frac{P(x_1,..., x_{i-1}, x_i = b, \pi_i = k \mid \theta) P(x_{i+1},..., x_L \mid x_1,..., x_{i-1}, x_i = b, \pi_i = k, \theta)}{P(x \mid \theta)}$$

$$= \frac{P(x_1,..., x_{i-1}, x_i = b, \pi_i = k \mid \theta) P(x_{i+1},..., x_L \mid \pi_i = k, \theta)}{P(x \mid \theta)}$$

$$= \frac{f_k(i) b_k(i)}{P(x)}.$$

Only the case where $x_i = b$ is considered.

Estimation of A_{kl} and $E_k(b)$ Values

- The Baum-Welch algorithm calculates A_{kl} and $E_k(b)$ as the expected number of times each transition or emission is used, given the training sequences.

$$A_{kl} = \sum_j \frac{1}{P(x^j)} \sum_i f_k^j(i) a_{kl} e_l(x_{i+1}^j) b_l^j(i+1)$$

$$E_k(b) = \sum_j \frac{1}{P(x^j)} \sum_{\{i|x_i^j = b\}} f_k^j(i) b_k^j(i)$$

  - Having calculated the above expectations, the new model parameter values are calculated.
  - We are converging in a continuous-valued space.
    - Stopping criterion: the (average) change in the log likelihood.

The Baum-Welch Algorithm

- Initialization: Pick arbitrary model parameters
- Recurrence:
  - Set all the A and E variables to their pseudocount values r (or to zero)
  - For each sequence $j = 1, \ldots, n$
    - Calculate $f_k(i)$ for sequence j using the forward algorithm
    - Calculate $b_k(i)$ for sequence j using the backward algorithm
    - Add the contribution of sequence j to A and E.
  - Calculate the new model parameters.
  - Calculate the new log likelihood of the model.
- Termination:
  - Stop if the change in log likelihood is less than some predefined threshold or the maximum number of iterations is exceeded.

The Baum-Welch Algorithm (Cont'd)

- The pseudocount values r could not be interpreted in terms of Dirichlet priors rigorously.
- The Baum-Welch algorithm is a special case of a very powerful general approach to probabilistic parameter estimation called the EM (expectation-maximization) algorithm.

The Viterbi Training

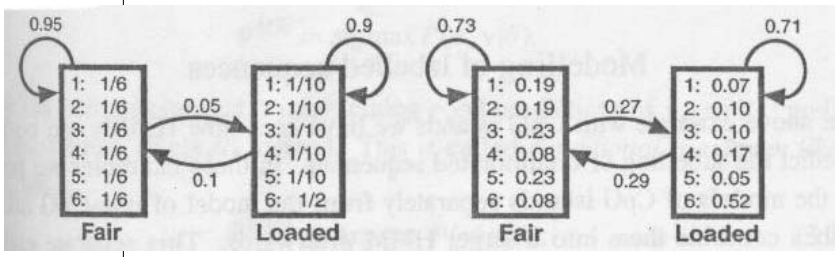
- The most probable path for each training sequence is derived by the Viterbi algorithm.
  - This information is used in the re-estimation process.
  - Once the information is given, it is the same as the training when the state path is known.
  - The algorithm converges precisely because the assignment of paths is a discrete process.
- The Baum-Welch maximizes the true likelihood.
  - $P(x^1, \ldots, x^n|\theta)$
- The Viterbi training finds the value of θ that maximizes the contribution to the likelihood, $P(x^1, \ldots, x^n|\theta, \pi^*(x^1), \ldots, \pi^*(x^n))$ from the most probable paths for all the sequences.
  - In general, the Baum-Welch performs better than the Viterbi training.
  - When the primary use of the HMM is Viterbi decoding.

Example: Casino, Part 5

- Training Sequences:

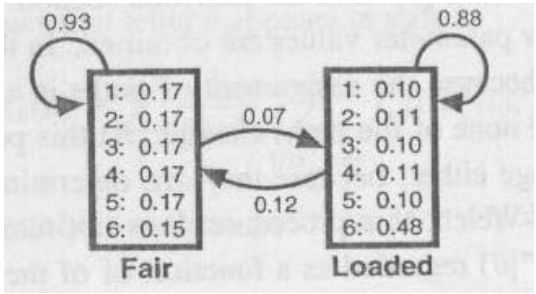
```
Rolls   315116246446644245311321631164152133625144543631656626566666

Rolls   651166453132651245636664631636663162326455236266666625151631

Rolls   222555441666566563564324364131513465146353411126414626253356

Rolls   366163666466232534413661661163252562462255265252266435353336

Rolls   233121625364414432335163243633665562466662632666612355245242
```


The Model Estimated by the Baum-Welch Algorithm



- Estimated probabilities are somewhat different.
  - Due to the problem of local maxima.
  - The limited amount of data does not permit to estimate the accurate parameter values.

The Model Learned from 30,000 Rolls



- The log-odds per roll (assuming a fair die)

$$\frac{1}{300}\log\frac{P(x\,|\,\text{a loaded die model})}{P(x\,|\,\text{a fair die model})}$$

The correct model: 0.101 bits

Model from 300 rolls: 0.097 bits

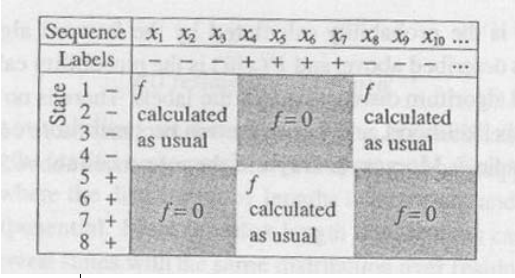
Model from 30,000 rolls: 0.097 bits

Estimation of the HMM Parameters for the Classification Problem

- We have to train the models of one class separately from the model of the other class and then combine them into a larger HMM.
  - This separate estimation can be quite tedious. → more than two classes
  - The estimation of the transitions is not a simple counting problem. ← when the transitions between the submodels are ambiguous.

Modeling of Labeled Sequences

- The combined model of all the classes
  - Each state is assigned a class label.
  - The label on the observation, $x_1, \ldots, x_L$.
    - $y_1, \ldots, y_L$.
  - In the Baum-Welch algorithm, only the valid paths are allowed.



Discriminative Estimation

- Unless there are ambiguous transitions between submodels, the above estimation procedure gives the same result as if the submodels were estimated separately by the Baum-Welch algorithm and then combined with appropriate transitions afterwards.

$$\theta^{ML} = \arg\max_{\theta} P(x, y \mid \theta)$$

- Our primary interest is in obtaining good predictions of y:
  - The conditional maximum likelihood

$$\theta^{CML} = \arg\max_{\theta} P(y \mid x, \theta)$$

The Maximum Mutual Information

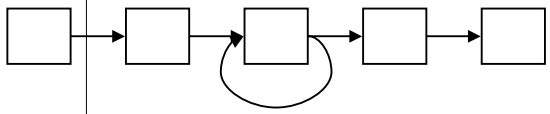
- $P(y|x, \theta) = P(x, y|\theta) / P(x|\theta)$
  - $P(x, y|\theta)$: the probability calculated by the forward algorithm for labeled sequences.
  - $P(x|\theta)$: the probability calculated by the standard forward algorithm disregarding all the labels.
  - There is no EM algorithm for optimizing this likelihood.

Choice of Model Topology

- Fully-connected models
  - The severe problem of local maxima → rarely used in practice
    - The less constrained the model is, the more severe the local maximum problem becomes.
  - There exist methods which attempt to adapt the morel topology based on the training data.
- Successful HMMs are constructed based on the knowledge about the problem under investigation.
  - Ex) the model of CpG islands
  - To disable the transition from state k to l → just set a_{kl} to be zero.

Duration Modeling

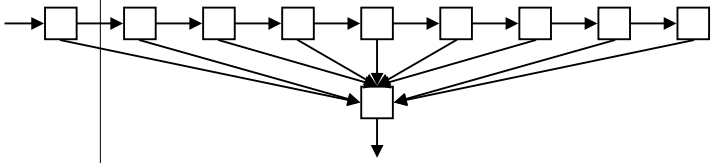
- The nucleotide distribution does not change for a certain length of DNA
  - The CpG islands model or the dishonest casino
- The probability of staying in the state for l residues:
  - $P(l \text{ residues}) = (1 - p)p^{l-1}$ → geometric distribution (exponential decay) → could be inappropriate in some applications
- A minimum length of 5 residues



  - An exponentially decaying distribution over longer sequences.

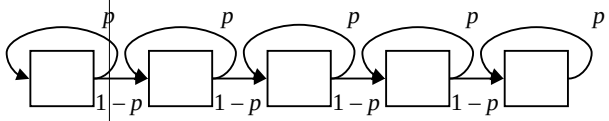
Duration Modeling (Cont'd)

- Any distribution of lengths between 2 and 10



A More Subtle Modeling for Non-Geometric Length Distribution

- An array of n states:

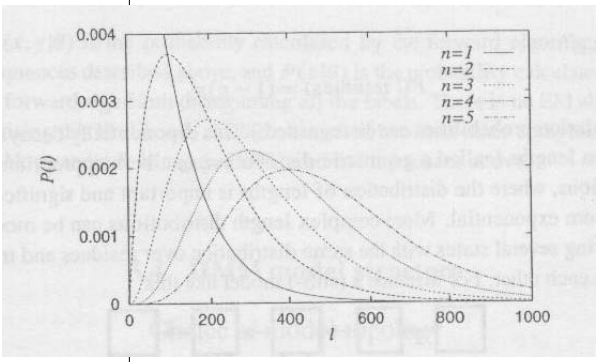


  - For any path of length n, the transition probability is $p^{l-n}(1 - p)^n$
- The probability of all the possible sequences of length l.

$$P(l) = \binom{l-1}{n-1} p^{l-n}(1-p)^n$$

The Negative Binomial Distribution

- ($p = 0.99$, $n \leq 5$)



For a continuous Markov process: Erlang distribution and phase-type distribution

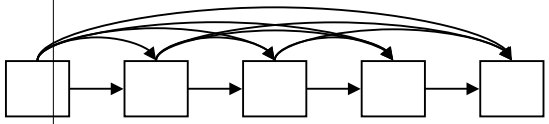
It is possible to model the length distribution explicitly.

Silent States

- The states that do not emit any symbol
  - Silent or null states (begin and end states)
- Example in Chapter 5
  - The Markov model where all states in a chain need to be connected to all states later in the chain.
  - If the length of the chain is 200 → about 20,000 transitions
    - Too large to be reliably estimated from realistic data sets.

The Forward Connected Model

- In order to allow for arbitrary deletions:

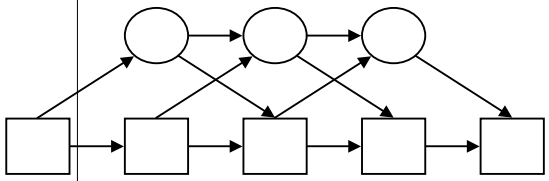


  - The number of parameters required for the above model consisting of 200 states is:

$$\sum_{i=1}^{199} i = 199 \times 198 / 2 \cong 20{,}000.$$

A Parallel Chain of Silent States

- A model with silent states



  - The number of parameters of the model correspondent to the above forward connected model is
    - 199 + 198 + 198 + 197 ≅ 800
  - The reduction of the parameters also reduces the representation power.
    - 1 → 5 and 2 → 4 with high probabilities while 1 → 4 and 2 → 5 with low probabilities.

Extensions to the Forward Algorithm

- So long as there are no loops consisting entirely of silent states, it is easy to extend all the HMM algorithms to incorporate them.
- For the forward algorithm
  - $f_k(i)$: $P(x_1, \ldots, x_i, \pi_i = k)$
  1. For all 'real' states l, calculate $f_l(i+1)$ as before from $f_k(i)$ for states k.
  2. For any silent state l, set $f_l(i+1)$ to $\Sigma_k f_k(i+1)a_{kl}$ for 'real' states k.
  3. Starting from the lowest numbered silent state l add $\Sigma_k f_k(i+1)a_{kl}$ to $f_l(i+1)$ for all silent states $k < l$.
- If there are loops entirely of silent states
  - Eliminate the silent states from the calculations → the fully connected model

High Order Markov Chains

- An *n*th order Markov process

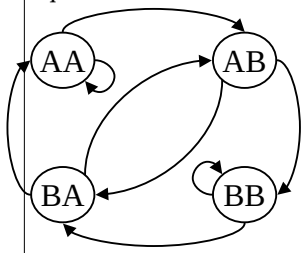
$$P(x_i \mid x_{i-1}, x_{i-2}, \ldots, x_1) = P(x_i \mid x_{i-1}, \ldots, x_{i-n})$$

  - An *n*th order Markov chain over some alphabet **A** is equivalent to a first order Markov chain over the alphabet $\mathbf{A}^n$ of *n*-tuples.
  - Because

$$P(x_k \mid x_{k-1}, \ldots, x_{k-n}) = P(x_k, x_{k-1}, \ldots, x_{k-n+1} \mid x_{k-1}, \ldots, x_{k-n})$$

A Second Order Markov Chains

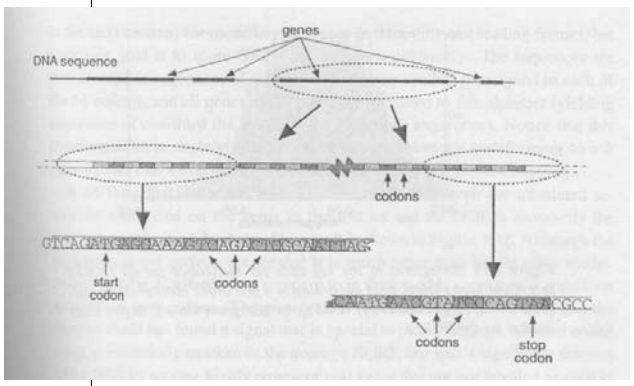
- A second order Markov chain for sequences of A and B
  - ABBAB → AB – BB – BA – AB
  - The equivalent first order Markov chain:



    - BB or BA following BA or AA is disallowed.
- Sometimes, the framework of high order model is convenient.

Finding Prokaryotic Genes

- Genes of prokaryotes (bacteria):
  - Very simple one-dimensional structure
  - Start codon : codons for amino acids : stop codon



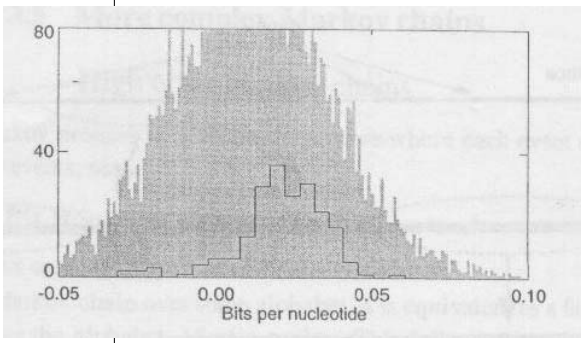
Open Reading Frames

- Finding good gene candidates from DNA stretches:
  - Start with one of the possible start codons.
  - Continuing with a number of non-stop codons.
  - Ending with one of the possible stop codons.
  - → Open reading frames (ORFs)
- Overlapping ORFs
  - The same stop codon with different start codons.
- There are many more ORFs than real genes.
  - → Discriminate between a non-coding ORF and a real gene.

Example: DNA from *E. Coli*

- The training data set
  - 1,100 genes more than 100 nucleotides long
  - 900 for training the model and 200 for testing the trained model.
- A first order model just as for the CpG islands was estimated from the training data
  - In the test set, 6,500 ORFs with a length of more than 100 bases were found.
  - ORFs that share the stop codon with a known real genes were not included.

Histograms of the Log-Odds per Nucleotide



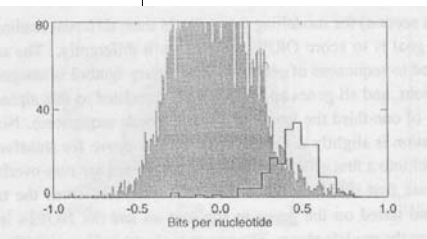
The null model for log-odds: the simples model with the probability for each nucleotide equal to the frequency by which it occurs in all the data.

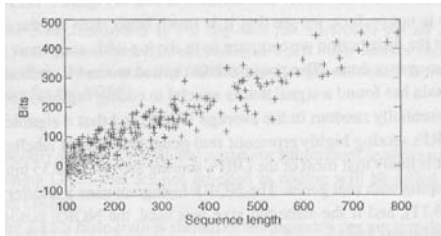
Average for genes: 0.018, average for NORFs: 0.009

Discrimination with this model is nearly impossible.

A Markov Chain Consisting of Codons

- All the sequences are transformed to sequences of codons.
- A 64-state first order Markov chain was estimated.
  - The null model: a uniform distribution over codons





Inhomogeneous Markov Chains

- Using the position information in the codon
  - Three models for the position 1, 2, and 3.

$$a^1_{x_1x_2} a^2_{x_2x_3} a^3_{x_3x_4} a^1_{x_4x_5} a^2_{x_5x_6} \ldots$$

- GENEMARK gene-finding program
  (http://opal.biology.gatech.edu/GeneMark/)
  - Inhomogeneous Markov chains are used.
- Extensions to the emission probabilities

$$e_k(b \mid b_1,\ldots,b_n) = P(x_i \mid \pi_i = k, x_{i-1} = b_1,\ldots, x_{i-n} = b_n)$$

Numerical Stability of HMM Algorithms

- Many calculations in Viterbi, forward, and backward algorithms
  - ♦ Multiplying many probabilities
  - ♦ A model of genomic sequences with 100,000 bases with a typical emission and transition probability of 0.1
    - ■ The resulting probability will be $10^{-100000}$.
- Most computers could not deal with such a small number.

The Log Transformation

- $\log_{10} 10^{-100000} = -100000$
- For the Viterbi algorithm

$$V_l(i+1) = \tilde{e}_l(x_{i+1}) + \max_k (V_k(i) + \tilde{a}_{kl})$$

- For the forward and backward algorithms
  - ♦ The logarithm of a sum of probabilities → exponentiation is required.

$$\tilde{r} = \log(p+q)$$
$$\tilde{p} = \log p \quad \text{and} \quad \tilde{q} = \log q$$
$$\tilde{r} = \log(\exp(\tilde{p}) + \exp(\tilde{q}))$$
$$= \tilde{p} + \log(1 + \exp(\tilde{q} - \tilde{p}))$$

Approximated by interpolation from a table

Scaling of Probabilities

- To rescale the *f* and *b* variables

$$\tilde{f}_l(i) = \frac{f_l(i)}{\prod_{j=1}^{i} s_j}$$

$$\tilde{f}_l(i+1) = s_{i+1} e_l(x_{i+1}) \sum_k \tilde{f}_k(i) a_{kl}$$

$$s_{i+1} = \sum_l e_l(x_{i+1}) \sum_k \tilde{f}_k(i) a_{kl} \quad (\textstyle\sum_l \tilde{f}_l(i) = 1)$$

$$\tilde{b}_k(i) = \frac{1}{s_i} \sum_l a_{kl} \tilde{b}_l(i+1) e_l(x_{i+1})$$

Further Reading

- Basic introduction to HMMs: [Rabiner and Juang, 1986; Krogh, 1998]
- Early applications of HMM-like models to sequence analysis: [Borodovsky et al., 1986a, 1986b, 1986c]
- GENEMARK genefinder program: [Borodovsky and McIninch, 1993]
- EM algorithm for modeling protein binding motifs: [Cardon and Stormo, 1992]
- Combining neural networks and HMMs: [Stormo and Haussler, 1996; Kulp et al., 1996; Reese et al., 1997; Burge and Karlin, 1997]
- HMMs for modeling compositional differences between DNA from mitochondria and from the human X chromosome and bacteriophage lambda: [Churchill, 1989]
- A three-state HMM for prediction of protein secondary structure: [Asai, Hayamizu, and Handa, 1993]
- A HMM with ten states in a ring for modeling an oscillatory pattern in nucleosomes [Baldi et al., 1996]