Bioinformatics Chapter 7.

Hiden Markov Models: The Theory

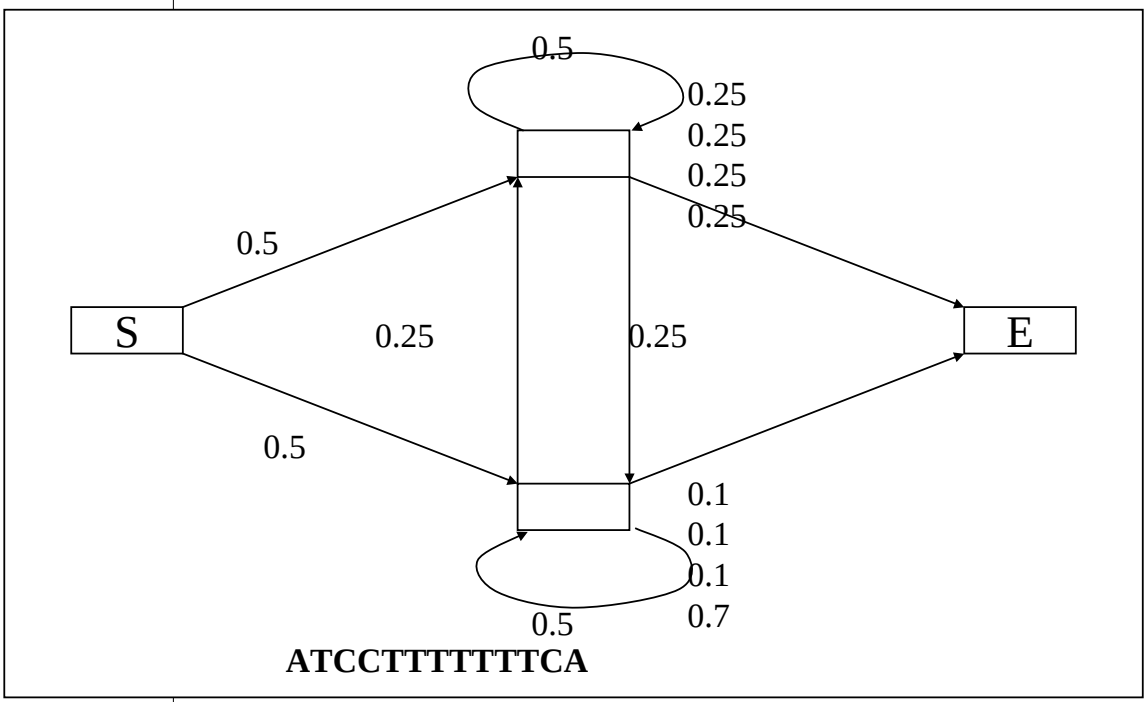
Outline

- Introduction
- Prior Information and Initialization
- Likelihood and Basic Algorithms
- Learning Algorithms
- Application of HMMs: General Aspects

Hidden Markov Model (HMM): Definition

- First order discrete HMM: stochastic generative model for time series
  - S : set of states
  - A : discrete alphabet of symbols
  - $T = (t_{ji})$: probability transition matrix, $t_{ji} = P(S^{t+1} = j \mid S^t = i)$
  - $E = (e_{ix})$: probability emission matrix, $e_{ix} = P(X \mid S^t = i)$
  - First order assumption: The emission and transition depend on the current state only, not on the entire previous states.
- Meaning of "Hidden"
  - Only emitted symbols are observable.
  - Random walk between states are hidden.

HMM: Example



HMMs for Biological Sequences (1)

- Example of symbols in biological sequence application
  - 20-letter amino acid alphabet for protein sequences
  - 4-letter nucleotide alphabet for DNA/RNA sequences
- Standard HMM architecture

  S = {start, m_1, …, m_N, i_1,…,i_{N+1}, d_1,…, d_N, end}
  - start, end
  - main states
  - insert states
  - delete states
  - N : length of model, typically average length of the sequences in the family

HMMs for Biological Sequences (2)

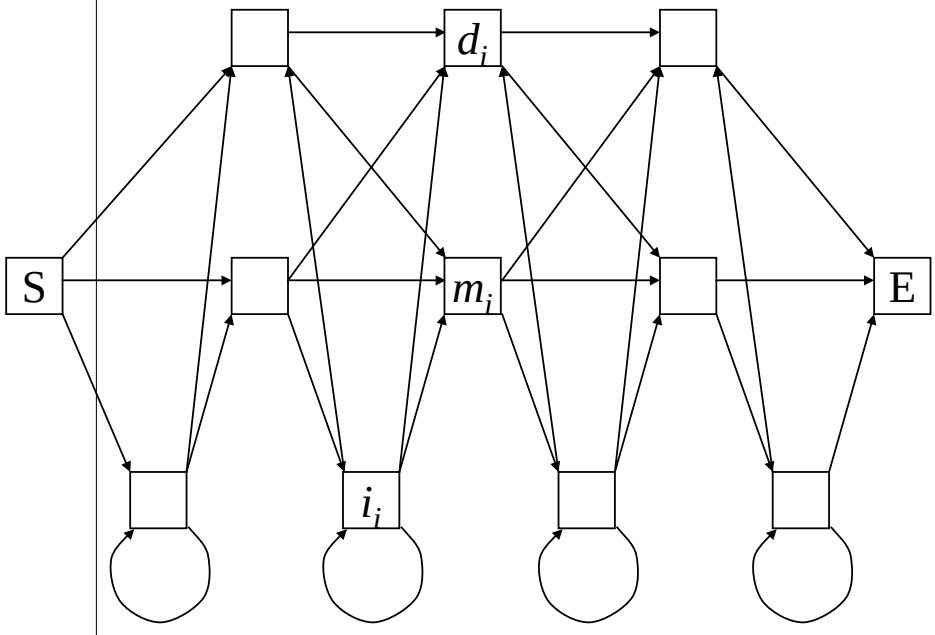


Figure 7.2 : The Standard HMM Architecture. S is the start state, E is the end state, and d_i, m_i and i_i denote delete, main, and insert state, respectively.

Three Questions in HMM

- Likelihood question
  - How likely is this sequence for this HMM?

Given a sequence $O_1, O_2, \ldots, O_r$
find $P(O_1, O_2 \ldots, O_r)$

- Decoding question
  - What is the most probable sequence of transitions and emissions through the HMM underlying the production of this particular sequence?

Given a sequence $O_1, O_2, \ldots, O_r$
find $\arg\max_{S_1, \ldots, S_r} P(S_1, \ldots, S_r \mid O_1, \ldots, O_r)$

- Learning question
  - How should their values be revised in light of the observed sequence?

Given sequences $\{O^1, O^2, \ldots, O^n\}$
find $w^* = \arg\max_w P(O^1, \ldots, O^n \mid w)$

Prior Information and Initialization

- Once the architecture is selected, one can further restrain the feedom of the parameters if the corresponding information is available.
- In the Bayesian approach, this background information can be incorporated by priors.
- Because of the multinomial models associated with HMM emissions and transitions, the natural priors on HMM parameters are Dirichlet distributions (Chap. 2).
  - Dirichlet priors on transitions: $D_{\alpha_m Q_m}(t_{ji})$, $D_{\alpha_i Q_i}(t_{ji})$, $D_{\alpha_d Q_d}(t_{ji})$
  - Dirichlet priors on emissions: $D_{\alpha_m Q_m}(e_{iX})$, $D_{\alpha_i Q_i}(t_{ji})$

Initialization of Parameters

- Different methods
  - Unform
  - Random
  - Average composition
- Uniform initialization without a prior that favors transitions toward the main states is not, in general, a good idea.
- Figure 7.3: Main states have a lower fan-out (3) than insert or delete states (4).
- Multiple alignment can be used to initialize the parameters prior to learning.
  - E.g.: assigning a main state to any column of the alignment that contains less than 50% gaps.

Likelihood and Basic Algorithm

- Likelihood of a sequence O according to HMM $M=M(w)$
  - O = $X^1 \ldots X^t \ldots X^T$: sequence of emission
  - $M(w)$: HMM with parameter w
  - Path π in M: sequence of consecutive states in which emitting states emit the corresponding letter.

$$P(O,\pi \mid w) = \prod_{start}^{end} t_{ji} \prod_{t=1}^{T} e_{iX^t} \qquad (7.1)$$

  - Likelihood equation

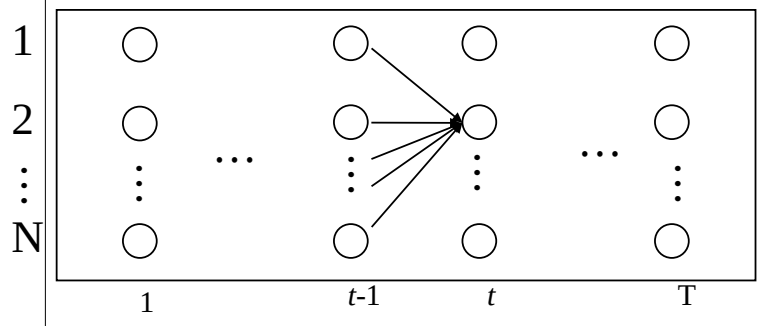
$$P(O \mid w) = \sum_{\pi} P(O,\pi \mid w)$$

- Difficulty of direct manipulation of likelihood equation
  - Number of paths is typically exponential!
  - Forward-algorithm: avoiding looking at all possible hidden paths

Forward-Backward Algorithm: Basic Concept

- Forward pass:
  - ♦ Define $\alpha_i(t) = \left[\sum_{j=1}^{N} \alpha_i(t-1) P(S_i \mid S_j)\right] P(O_t \mid S_i)$
  - ♦ Probability of subsequence $O_1, O_2, \ldots, O_t$ when in S_i at t

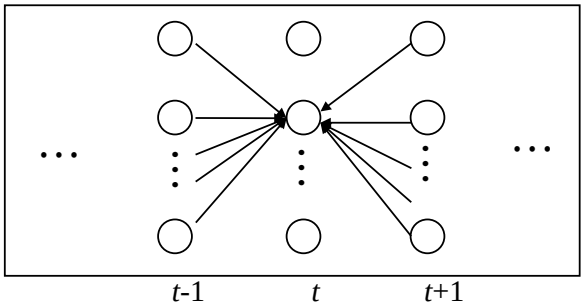
    Note: any path must be in one of N states at t



Forward-Backward Algorithm

- Notice $P(O_1, O_2, \mathrm{K}, O_T) = \sum_{i=1}^{N} \alpha_i(T)$
- Define an analogues backward pass so that:

$$\beta_t(i) = \sum_{j=1}^{N} \beta_j(t+1) P(S_j \mid S_i) P(O_{t+1} \mid S_j)$$



and

$$P(O_t \text{ came from } S_i) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}$$

$$= \gamma_i(t)$$

Forward: j i

Backward: i j

The Forward Algorithm (1)

- Objective: $P(O \mid w) = \alpha_{end}(T)$
- Algorithm: $\alpha_i(t+1) = \sum_{j \in S} \alpha_j(t) t_{ij} e_{iX^{t+1}} = \sum_{j \in N^-(i)} \alpha_j(t) t_{ij} e_{iX^{t+1}}$
  - The probability of being in state *i* at time *t*, having observed the letters $X^1 \ldots X^t$ in the model $M(w)$

    $$\alpha_i(t) = P(S^t = i, X^1 \text{ K } X^T \mid w), \qquad \alpha_{start}(0) = 1$$

  - For delete states,

    $$\alpha_i(t+1) = \sum_{j \in N^-(i)} \alpha_j(t+1) t_{ij}$$

  - Convergence in the case of delete states (Appendix D)

The Forward Algorithm (2)

- *Silent* path from *j* to *i*
  - If the only internal nodes in the path are delete (silent) states.
  - t_{ij}^D : the probability of moving from *j* to *i* silently.
- Forward variables for delete states

  $$\alpha_i(t+1) = \sum_{j \in E} \alpha_j(t+1) t_{ij}^D$$

  - E : all emitting states
- Relationship with other algorithms
  - Forward propagation as in a linear neural network: with *T* layers (each time step) and *M* units in each layer (one for each HMM states)
  - An HMM can be viewed as a dynamic mixture model
    - Probability of emitting the letter X^t : $\sum_i \alpha_i(t) e_{iX^t}$

The Backward Algorithm (1)

- Objective: the reverse of the forward algorithm
  - Backward variable: the probability of being in state *i* at time *t*, with partial observation of the sequence from X^{t+1} to end.

$$\beta_i(t) = P(X^{t+1} \text{ K } X^T \mid S^T = i, w), \qquad \beta_{end}(T) = 1$$

  - Propagation equation for emitting states

$$\beta_i(t) = \sum_{j \in N^+(i)} \beta_j(t+1) t_{ji} e_{jX^{t+1}}$$

  - Propagation equation for delete states

$$\beta_i(t) = \sum_{j \in E} \beta_j(t) t_{ij}^D$$

The Backward Algorithm (2)

- Using the forward and backward variables, we can compute the probability $\gamma_i(t)$ of being in state *i* at time *t*, given the observation sequence *O* and the model *w*.

$$\gamma_i(t) = P(S^t = i \mid O, w) = \frac{\alpha_i(t)\beta_i(t)}{P(O \mid w)} = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j \in S} \alpha_j(t)\beta_j(t)}$$

- Probability $\gamma_{ji}(t)$ of using $i \rightarrow j$ transition at time *t*.

$$P(S^{t+1} = j, S^t = i \mid O, w) = \begin{cases} \alpha_i(t) t_{ji} e_{jX^{t+1}} \beta_j(t+1) / P(O \mid w) & \text{if } j \in E \\ \alpha_i(t) t_{ji} \beta_j(t) / P(O \mid w) & \text{if } j \in D \end{cases}$$

- Also $\gamma_i(t) = P(S^t = i \mid O, w) = \sum_{j \in S} \gamma_{ji}(t)$

The Viterbi Algorithm

- Objective: Find the most probable path accounting for the first t symbols of O and terminating in state i.

$$\delta_i(t) = \max_{\pi_i(t)} P(\pi_i(t) \mid w)$$

  - $\pi_i(t)$: a prefix path, with emissions $X^1,\ldots,X^t$ ending in state i.
- Propagation
  - For emitting states

$$\delta_i(t+1) = [\max_j \delta_j(t) t_{ij}] e_{iX^{t+1}}$$

  - For deleting states

$$\delta_i(t+1) = [\max_j \delta_j(t+1) t_{ij}]$$

Computing Expectations

- Posterior distribution $Q(\pi) = P(\pi \mid O, w)$
- For learning, some expectations needs to be calculated.
  - $n(i, \pi, O)$: the number of times i is visited, given π and O

$$n_i = \sum_{\pi} n(i,\pi,O) P(\pi \mid O, w) = \sum_{t=0}^{T} \gamma_i(t)$$

  - $n(i, X, \pi, O)$: the number of times the letter X is emitted from i, given π and O

$$n_{iX} = \sum_{\pi} n(i, X, \pi, O) P(\pi \mid O, w) = \sum_{t=0, X^t = X}^{T} \gamma_i(t)$$

  - $n(j, i, X, \pi, O)$: the number of times the $i \rightarrow j$ transition is used, given π and O

$$n_{ji} = \sum_{\pi} n(i, j, \pi, O) P(\pi \mid O, w) = \sum_{t=0}^{T} \gamma_{ji}(t)$$

Learning Algorithms (1)

- HMM training algorithms
  - Baum-Welch or EM algorithm
  - Gradient-descent algorithms
  - Generalized EM algorithms
  - Simulated annealing
  - Markov chain Monte Carlo
  - …
- We concentrate on the first level of Bayesian inference, i.e. MAP estimation, proceeding as follows:
  - ML with on-line learning (given one sequence)
  - ML with batch learning (with multiple sequences)
  - MAP (considering priors)

\

Learning Algorithms (2)

- Consider the likelihoo $P(O \mid w) = \sum_{\pi} P(O, \pi \mid w)$
- Lagrangian optimization function of ML

$$L = -\log P(O \mid w) - \sum_{i \in E} \lambda_i \left(1 - \sum_X e_{iX}\right) - \sum_{i \in S} \mu_i \left(1 - \sum_j t_{ji}\right) \qquad (7.22)$$

  - λ_i, μ_i : Lagrange multipliers from normalization constraints

$$\sum_X e_{iX} = 1, \quad i \in E; \qquad \sum_j t_{ji} = 1, \quad i \in S$$

- From (7.1) $P(O, \pi \mid w) = \prod_{start}^{end} t_{ji} \prod_{t=1}^{T} e_{iX^t}$, we have

$$\frac{\partial P(O, \pi \mid w)}{\partial e_{iX}} = \frac{n(i, X, \pi, O)}{e_{iX}} P(O, \pi \mid w) \qquad (7.23)$$

- By setting the partial derivative of L to 0, at the optimum we must have $\lambda_i e_{iX} = \sum_{\pi} n(i, X, \pi, O) Q(\pi) = n_{iX}$ $\qquad Q(\pi) = P(\pi \mid O, w)$

Learning Algorithms (3)

- By summing over all alphabet letters,

$$\lambda_i = \sum_{\pi}\sum_{X} n(i, X, \pi, O)Q(\pi) = \sum_{\pi} n(i, \pi, O)Q(\pi) = n_i$$

- At the optimum, we must have (in case of emission probability)

$$e_{iX} = \frac{\sum_{\pi} n(i, X, \pi, O)Q(\pi)}{\sum_{\pi} n(i, \pi, O)Q(\pi)} = \frac{\sum_{\pi} P(\pi \mid O, w)n(i, X, \pi, O)}{\sum_{\pi} P(\pi \mid O, w)n(i, \pi, O)} \qquad (7.26)$$

- ML equation cannot be solved directly since the posterior distribution Q depends on the values of e_{iX}
- EM algorithms can do it.
  - First, estimate $Q(\pi) = P(\pi|O,w)$
  - Second, update parameter using (7.26)

EM Algorithm (1): Baum-Welch

- We define the energy over hidden configurations (Chs. 3 and 4) as

$$f(\pi) = -\log P(O, \pi \mid w)$$

- EM is then defined as an iterative double minimization process of the function (i.e., free energy at temperature 1) w.r.t. Q and w

$$F(w, Q) = E_Q(f) - H(Q)$$

- First step, calculate Q

$$Q(\pi) = P(\pi \mid O, w) = P(\pi, O \mid w) / P(O \mid w)$$

- Second step, minimize F, with respect to w, with $Q(\pi)$ **fixed**. Since the entropy term H is independent of w, we minimize the Lagrangean:

$$L = -E_Q(f) - \sum_{i \in E} \lambda_i (1 - \sum_{X} e_{iX}) - \sum_{i \in S} \mu_i (1 - \sum_{j} t_{ji}) \qquad Q(\pi) = P(\pi \mid O, w)$$

EM Algorithm (2): Baum-Welch

- Following similar steps as before (Eqns. 7.23-7.25), we obtain the EM reestimation equations

$$e_{iX}^{+} = \frac{\sum_{\pi} n(i, X, \pi, O) Q(\pi)}{\sum_{\pi} n(i, \pi, O) Q(\pi)} = \frac{\sum_{t=0, X^t = X}^{T} \gamma_i(t)}{\sum_{t=0}^{T} \gamma_i(t)} = \frac{n_{iX}}{n_i} \qquad \text{(emmission)}$$

$$t_{iX}^{+} = \frac{\sum_{\pi} n(j, i, \pi, O) Q(\pi)}{\sum_{\pi} n(i, \pi, O) Q(\pi)} = \frac{\sum_{t=0}^{T} \gamma_{ji}(t)}{\sum_{t=0}^{T} \gamma_i(t)} = \frac{n_{ji}}{n_i} \qquad \text{(transition)}$$

- In words, the statistic $e_{iX}^{+} = n_{iX} / n_i$ is the expected number of times in state *i* observing symbol *X*, divided by the expected number of times in state *i*.
- Forward-backward procedures can calculate above statistics.

Batch EM Algorithm

- In the case of K sequences $O_1, \ldots, O_K$

$$e_{iX}^{+} = \frac{\sum_{j=1}^{K} \sum_{\pi} n(i, X, \pi, O_j) P(\pi \mid O_j, w)}{\sum_{j=1}^{K} \sum_{\pi} n(i, \pi, O_j) P(\pi \mid O_j, w)}$$

- Online use of EM can be problematic.
  - No learning rate available and the EM can take large steps in the descent direction, leading to poor local minima.
  - 'Carpet jumping' effect

Gradient Descent (1)

- Reparameterize HMM using normalized exponentials, with new variables w_{ix}, w_{ij}.

$$e_{iX} = \frac{e^{w_{iY}}}{\sum_Y e^{w_{iY}}} \qquad t_{ji} = \frac{e^{w_{ji}}}{\sum_k e^{w_{ki}}}$$

  - Advantage 1: Automatically preserving normalization constrains
  - Advantage 2: Never reaching zero probabilities.

- Partial derivatives

$$\frac{\partial e_{iX}}{\partial w_{iX}} = e_{iX}(1 - e_{iX}), \quad \frac{\partial e_{iX}}{\partial w_{iY}} = -e_{iX} e_{iY}$$

Gradient Descent (2)

- Chain rule:

$$\frac{\partial \log P(O \mid w)}{\partial w_{iX}} = \sum_Y \frac{\partial \log P(O \mid w)}{\partial e_{iY}} \frac{\partial e_{iY}}{\partial w_{iX}}$$

- On-line gradient descent:

$$\Delta w_{iX} = \eta(n_{iX} - n_i e_{iX}), \qquad \Delta w_{ji} = \eta(n_{ji} - n_i t_{ji})$$

- Remarks
  - For MAP estimation, add the derivative of the log prior.
  - O(KN^2) operations per training cycle.
  - Just like EM, one forward and backward propagation are needed per iteration.
  - Unlike EM, online gradient-descent is a smooth algorithm: unlearning (reversing the effect of gradient descent) is easy.

Viterbi Algorithm (1)

- Idea: Focus on the most likely one path with each sequence
  - EM and gradient descent update equation: expectation over all possible hidden paths.
- Replace: $n(i, X, \pi, O) \rightarrow n(i, X, \pi^*(O))$
  - In standard architecture, $n(i, X, \pi^*(O)) = 0$ or 1, except for insert states.
  - On-line Viterbi EM makes little sense: mostly updated to 0 or 1
- On-line gradient descent, at each step along a Viterbi path, and for any state i on the path, update the parameters.

$$\Delta w_{iX} = \eta(E_{iX} - n_i e_{iX}), \qquad \Delta w_{ji} = \eta(T_{ji} - n_i t_{ji})$$

  - $E_{ix} = 1$ ($T_{ji} = 1$) if emission of X from i ($i \rightarrow j$ transition) is used.

Viterbi Algorithm (2)

- Remark 1
  - Quick approximation to the corresponding non-Viterbi version.
  - Speed up of order of factor 2: $\pi^*(O)$ with no backward propagation
- Remark 2
  - Crude approximation
  - Likelihoods in general are not sharply peaked around a single optimal path
- Remark 3
  - Minimizing a different objective function

$$E = \sum_{k=1}^{K} -\log P^V(O \mid w) \qquad P^V(O \mid w) = \frac{P(\pi^*(O) \mid w)}{\sum_O P(\pi^*(O) \mid w)}$$

Other Aspects (1)

- Scaling
  - Machine precision vs. $P(\pi|O,w)$
  - Some techniques to avoid underflow are needed.
- Learning the Architecture
  - [389]: merge states from complex model.
  - [142]: small starting point; deleting transitions with low probability; duplicating most connected states.
  - Good results in small test cases, but unlikely in practice
- Adaptable Model Length
  - Adaptation of N of average length of the sequences being modeled.
  - [251]: "*surgery*" algorithm
  - 50% used insert state: replace it with new main state (together with corresponding new delete and insert state.)
  - 50% used delete state: remove it together with corresponding main and insert states.

Other Aspects (2)

- Architectural Variations
  - The variations of standard architecture.
  - Multiple HMM architecture, loop and wheel are introduced in Chapter 8.
- Ambiguous Symbols
  - The reason of adoption of ambiguous symbols: imperfect sequencing techniques.

Applications of HMMs: General Aspects

- Successfully derived from a family of sequences.
- For any given sequence
  - The computation of its probability according to the model as well as its most likely associated path
  - Aanalysis of the model structure.
- Applications
  - Multiple alignments
  - Database mining and classification of sequence and fragments
  - Structural analysis and pattern discovery

Multiple Alignments (1)

- Aligning the Viterbi paths to each other.
  - Computing the Viterbi path of a sequence: "*aligning a sequence to the model*"
  - O(KN^2): when multiple alignment of K sequences.
  - O(N^K): in case of multi-dimensional dynamic programming
- Rich expression power against conventional methods
  - Gap: deletion in the second sequence of insertion in the first sequence → Two distinct sets of Viterbi paths in HMM
  - Conventional methods can not distinguish.
  - HMM becomes conventional techniques if N is fixed to length of longest sequence and all insert states are removed.

Multiple Alignments (2)

- Insert and delete states of HMM represent formal operations on sequences.
  - Whether and how they can be related to evolutionary events?
  - Phylogenetic trees?: require tree structure as well as a clear notion of substitution (Chapter 10)
- Full Bayesian treatment is nearly intractable in HMM.

Database Mining and Classification

- The likelihood score of any given sequence can be used for the purpose of discriminative test and database search.
- For classification
  - Training a model for each class.
  - Give class label to the largest likelihood score.

Structural Analysis and Pattern Discovery

- Information or new pattern is discovered by examining the structure of a trained HMM.
  - High emission or transition probabilities are usually associated with conserved regions or consensus patterns.
  - One technique: plot the entropy of the emission distributions along the backbone of the model.
- Initial weak detection can guide the design of more specialized architectures (Chapter 8).