Hidden Markov Models

Andrea Passerini
passerini@disi.unitn.it

Statistical relational learning
The aim

Modeling temporal sequences

- Model signals which vary over time (e.g. speech)
- Two alternatives:
  - **deterministic models** directly model the signal with parametric functions (e.g. sums of sine waves)
  - **statistical models** model the statistical properties of the signal with a **parametric random process** (e.g. Markov processes)
- Statistical models make less assumptions on the characteristics of the signal
First-order Markov models

Components

- A set of \( N \) possible states \( \{\omega_1, \ldots, \omega_N\} \)
- A set of transition probabilities between states:
  
  \[
a_{ij} = P(\omega_j(t) | \omega_i(t - 1)), \quad 1 \leq i, j \leq N
  \]

- A set of initial probabilities for states:
  
  \[
  \pi_i = P(\omega_i(1)), \quad 1 \leq i \leq N
  \]

- satisfying probability properties:
  
  \[
  a_{ij}, \pi_i \geq 0 \quad \sum_{j=1}^{N} a_{ij} = 1 \quad \sum_{i=1}^{N} \pi_i = 1
  \]
Let $\omega^5 = \{w_1, w_1, w_3, w_2, w_1\}$ be a sequence of 5 states.

Its probability according to a first order Markov model $\theta$ is:

$$P(\omega^5 | \theta) = \pi_1 a_{11} a_{13} a_{32} a_{21}$$
First-order Markov models

**Characteristics**

<table>
<thead>
<tr>
<th>Order</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>First order</strong></td>
<td>The probability of a state depends only on the previous one</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discrete time</strong></td>
<td>Time can be represented as a sequence of instants</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Static process</strong></td>
<td>Transition probabilities do not depend on the time instant</td>
</tr>
</tbody>
</table>

**Problem**

- The model assumes everything is observed (each state corresponds to an observable event)
- E.g. in speech modeling, states could be associated with phonemes, but only some properties of the emitted sound are observed.
First-order hidden Markov Models (HMM)

Components

- A set of \( N \) possible (hidden) states \( \{\omega_1, \ldots, \omega_N\} \)
- A set of \( M \) possible (visible) symbols \( \{v_1, \ldots, v_M\} \)
- A set of state transition probabilities:
  \[ a_{ij} = P(\omega_j(t)|\omega_i(t-1)), \ 1 \leq i, j \leq N \]
- A set of initial probabilities for states:
  \[ \pi_i = P(\omega_i(1)), \ 1 \leq i \leq N \]
- A set of emission probabilities for states:
  \[ b_{jk} = P(v_k(t)|\omega_j(t)), \ 1 \leq j \leq N, \ 1 \leq k \leq M \]

satisfying probability properties:

\[
\begin{align*}
    a_{ij}, \pi_i, b_{jk} &\geq 0 \\
    \sum_{j=1}^{N} a_{ij} &= 1 \\
    \sum_{i=1}^{N} \pi_i &= 1 \\
    \sum_{k=1}^{M} b_{jk} &= 1
\end{align*}
\]
### HMM Computation

#### Computation problems

<table>
<thead>
<tr>
<th><strong>evaluation</strong></th>
<th>Given an HMM with both emission and transition probabilities, compute the probability that a certain sequence of observations $v^T = {v_1, \ldots, v_T}$ was generated by the model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>decoding</strong></td>
<td>Given an HMM with both emission and transition probabilities, and a sequence of observations $v^T$, compute the most likely sequence of hidden states $\omega^T$ that produced those observations</td>
</tr>
<tr>
<td><strong>learning</strong></td>
<td>Given the structure of an HMM and a training set of observations, learn the HMM parameters (transition and emission probabilities).</td>
</tr>
</tbody>
</table>
Evaluation

**Probability of observation given model**

\[
P(v^T | \theta) = \sum_{r=1}^{r_{\text{max}}} P(v^T, \omega_r^T | \theta)
\]

- \(\omega_r^T = \{\omega_r(1), \ldots, \omega_r(T)\}\) is a possible sequence of \(T\) hidden states.
- \(P(v^T, \omega_r^T | \theta)\) is the joint probability of a sequence of observations and states.
- The probability of observations is obtained summing over all possible sequences of hidden states.
Joint probability of observations and states

The joint probability of a sequence of observations and states given the Markov model can be represented as a Bayesian Network.

\[ P(v^T, \omega_r^T | \theta) = P(v^T | \omega_r^T, \theta)P(\omega_r^T | \theta) \]

- The joint probability of a sequence of observations and states given the Markov model can be represented as a Bayesian Network.
Joint probability of observations and states

\[
P(v^T, \omega^T | \theta) = P(v^T | \omega^T, \theta)P(\omega^T | \theta)
\]

- Emissions are independent of each other given the states, and depend only on the current state:

\[
P(v^T | \omega^T, \theta) = \prod_{t=1}^{T} P(v(t) | \omega_r(t), \theta)
\]
Joint probability of observations and states

\[ P(v^T, \omega_r^T | \theta) = P(v^T | \omega_r^T, \theta)P(\omega_r^T | \theta) \]

- The probability of a certain state only depends on the previous one:

\[ P(\omega_r^T | \theta) = \prod_{t=1}^{T} P(\omega_r(t) | \omega_r(t - 1), \theta) \]

- Note: we assume that \( P(\omega_r(1) | \omega_r(0)) = P(\omega_r(1)) \) is the initial probability of state \( \omega_r(1) \).
Joint probability of observations and states

from HMM to BN

Combining emission and transition probabilities we obtain:

\[
P(\mathbf{v}^T, \mathbf{\omega}^T | \theta) = P(\mathbf{v}^T | \mathbf{\omega}^T_r, \theta)P(\mathbf{\omega}^T_r | \theta) \]

\[
= \prod_{t=1}^{T} P(\mathbf{v}(t) | \mathbf{\omega}_r(t), \theta)P(\mathbf{\omega}_r(t) | \mathbf{\omega}_r(t - 1), \theta)
\]
Joint probability of observations and states

Example

\[ P(v_1, v_1, v_3, v_2, \omega_2, \omega_1, \omega_1, \omega_4 | \theta) = P(v_1 | \omega_2)P(v_1 | \omega_1)P(v_3 | \omega_1)P(v_2 | \omega_4)P(\omega_2, \omega_1, \omega_1, \omega_4) \]

\[ = b_{21}b_{11}b_{13}b_{42}P(\omega_2)P(\omega_1 | \omega_2)P(\omega_1 | \omega_1)P(\omega_4 | \omega_1) \]

\[ = b_{21}b_{11}b_{13}b_{42} \pi_2 a_{21}a_{11}a_{14} \]
Probability of observation given model

\[ P(v^T | \theta) = \sum_{r=1}^{\text{r}_{\text{max}}} P(v^T | \omega_r^T, \theta) P(\omega_r^T | \theta) \]

\[ = \sum_{r=1}^{\text{r}_{\text{max}}} \prod_{t=1}^{T} P(v(t) | \omega_r(t), \theta) P(\omega_r(t) | \omega_r(t - 1), \theta) \]

Problem

- Given \( N \) possible states, the number of distinct sequences of \( T \) states is \( r_{\text{max}} = N^T \).
- Naively summing over all possible sequences is thus intractable.
Forward procedure

Intuition

- Relies on a **forward variable** $\alpha_i(t)$ containing the probability of being in state $\omega_i(t)$ at time $t$ having observed the sequence $\{v(1), \ldots, v(t)\}$:

$$
\alpha_i(t) = P(v(1), \ldots, v(t), \omega_i(t)|\theta)
$$

- A **dynamic programming** procedure allows to compute the forward variable at time $t$ based on the one at time $t - 1$.

- Once time $T$ is reached, the probability of observations is simply:

$$
P(v^T|\theta) = \sum_{i=1}^{M} P(v^T, \omega_i(T)|\theta) = \sum_{i=1}^{M} \alpha_i(T)
$$
Forward procedure

The algorithm

1 Initialization:

\[ \alpha_i(1) = P(\omega_i(1), v(1)|\theta) = \pi_i b_{iv(1)}, \quad 1 \leq i \leq N \]

2 Induction:

\[ \alpha_j(t) = \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} b_{jv(t)}, \quad 2 \leq t \leq T, 1 \leq j \leq N \]

3 Termination:

\[ P(v^T|\theta) = \sum_{i=1}^{N} \alpha_i(T) \]

Note

Complexity is \( N^2 T \) instead of \( N^T T \)
Forward procedure

\[ \alpha_1(2) \]
\[ \alpha_2(2) \]
\[ \alpha_3(2) \]
\[ \alpha_N(2) \]

\[ v_k \]
\[ b_{2k} \]

\[ t = 1 \ 2 \ 3 \ T-1 \ T \]
Intuition

- Relies on a **backward variable** $\beta_i(t)$ containing the probability of observing $\{v(t+1), \ldots, v(T)\}$, given the state $\omega_i(t)$ (and the model):

  $$\beta_i(t) = P(v(t+1), \ldots, v(T)|\omega_i(t), \theta)$$

- A **dynamic programming** procedure allows to compute the backward variable at time $t$ based on the one at time $t + 1$.

Aim

- Will be used for parameter learning
- Is the counterpart of the forward procedure, when going back in time
### Backward procedure

#### The algorithm

1. **Initialization:**
   \[
   \beta_i(T) = 1, \quad 1 \leq i \leq N
   \]

2. **Induction:**
   \[
   \beta_i(t) = \sum_{j=1}^{N} a_{ij} b_{j(t+1)} \beta_j(t + 1), \quad T - 1 \geq t \geq 1, \quad 1 \leq i \leq N
   \]

#### Note

Complexity is again \(N^2 T\) instead of \(N^T T\)
Decoding

Most-likely sequence of states given observations and model

\[ \omega^*_T = \arg\max_{\omega^T} P(\omega^T | v^T, \theta) \]
\[ = \arg\max_{\omega^T} P(\omega^T | v^T, \theta) P(v^T | \theta) \]
\[ = \arg\max_{\omega^T} P(\omega^T, v^T | \theta) \]

Problem

As for the evaluation case, the naive approach of computing the joint probability for all possible state sequences and taking the max is intractable \(O(N^T T)\)
Viterbi decoding

**Intuition**

- Relies on a **max variable** $\delta_i(t)$ containing the probability of the best sequence of states up to $t - 1$ plus state $\omega_i(t)$ and the observations up to time $t$, given the model.

$$
\delta_i(t) = \max_{\omega(1), \ldots, \omega(t-1)} P(\omega(1), \ldots, \omega(t-1), \omega_i(t), v(1), \ldots, v(t)|\theta)
$$

- A **dynamic programming** procedure allows to compute the max variable at time $t$ based on the one at time $t - 1$.
- An array $\psi$ allows to keep track of the states which maximized each step.
- Once time $T$ is reached, a backtracking procedure allows to recover the sequence of states which maximized overall probability.
## Viterbi decoding

### The algorithm

1. **Initialization:**
   \[
   \delta_i(1) = P(\omega_i(1), v(1)|\theta) = \pi_i b_1 v(1), \quad 1 \leq i \leq N
   \]

2. **Induction:**
   \[
   \delta_j(t) = \max_{1 \leq i \leq N} [\delta_i(t-1) a_{ij}] b_j v(t), \quad 1 \leq j \leq N, \quad 2 \leq t \leq T
   \]
   \[
   \psi_j(t) = \arg\max_{1 \leq i \leq N} [\delta_i(t-1) a_{ij}], \quad 1 \leq j \leq N, \quad 2 \leq t \leq T
   \]

3. **Termination:**
   \[
   P^* = \max_{1 \leq i \leq N} \delta_i(T)
   \]
   \[
   \omega^*(T) = \arg\max_{1 \leq i \leq N} \delta_i(T)
   \]

4. **Path (state sequence) backtracking:**
   \[
   \omega^*(t) = \psi_{\omega^*(t+1)}(t + 1), \quad t = T - 1, T - 2, \ldots, 1
   \]
Learning HMM parameters

Settings

- Single training sequence $v(T)$ (generalization to multiple sequences obtained simply summing over each of them)
- Fully observed training data (both observations $v(T)$ and states $\omega(T)$):
  - Simply compute parameters from counts (e.g. $a_{ij}$ as fraction of transitions from $\omega_i$ to $\omega_j$ over occurrences of state $\omega_i$).
- Training data with observations only (hidden states):
  - Maximum likelihood computation by gradient descent
  - Maximum likelihood computation by generalized EM
Generalized EM

Intuition

- Sufficient statistics are counts of transitions, emissions and initial states.

Procedure

1. Initialize parameters.
2. Iterate:
   1. Estimate counts according to current version of the parameters.
   2. Update parameters in order to better match counts.
### Forward-backward (or Baum-Welch) algorithm

#### Components

- **Training sequence** $\mathbf{v}(T)$.
- **Probability of transition from a state** $\omega_i(t)$ to $\omega_j(t + 1)$ in any possible path with which the model can generate $\mathbf{v}(T)$:

$$
\gamma_{ij}(t) = P(\omega_i(t), \omega_j(t + 1) | \mathbf{v}(T), \theta)
= \frac{P(\mathbf{v}(T), \omega_i(t), \omega_j(t + 1) | \theta)}{P(\mathbf{v}(T) | \theta)}
= \frac{\alpha_i(t) a_{ij} b_{j\mathbf{v}(t+1)} \beta_j(t + 1)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i(t) a_{ij} b_{j\mathbf{v}(t+1)} \beta_j(t + 1)}
$$

- **Probability of transition from state** $\omega_i(t)$ to any state:

$$
\gamma_i(t) = \sum_{j=1}^{N} \gamma_{ij}(t)
$$
Forward-backward algorithm

E-step: computing counts

\[ \gamma_i(1) \quad \text{Expected frequency of times in initial state } \omega_i \]

\[ \sum_{t=1}^{T-1} \gamma_{ij}(t) \quad \text{Expected number of transitions from state } \omega_i \text{ to state } \omega_j \]

\[ \sum_{t=1}^{T-1} \gamma_i(t) \quad \text{Expected number of transitions from state } \omega_i \]

\[ \sum_{t=1}^{T} \gamma_i(t) \quad \text{Expected number of times in state } \omega_i \]

\[ \sum_{t=1}^{T} \gamma_i(t) \quad \text{Expected number of times in state } \omega_i \text{ and observing symbol } v_k \]
Forward-backward algorithm

M-step: updating parameters

- **Initial state probabilities:**
  \[ \pi_i = \gamma_i(1) \]

- **State transition probabilities:**
  \[ a_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \]

- **Emission probabilities:**
  \[ b_{ik} = \frac{\sum_{t=1}^{T} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)} \]
The simplest approach is a fully connected model, where transitions are possible between any pair of states. This corresponds to making no independency assumption between state variables. Fully connected architectures almost never work in practice, as they typically badly overfit training data.
Choosing a structure by domain knowledge

- The best approach usually consists of choosing a reasonable structure according to the available knowledge of the domain at hand.
- This corresponds to setting some of the transition probabilities $a_{ij} = 0$, while all algorithms remain valid.
- Different states can be introduced which share the same emission probabilities.
- A common option is the so called left-right models, where connections are only allowed between a state and the following ones.
HMM model structure

Duration modelling

- It is possible to add constraints on the minimum length of paths within the “same” state.
- This is obtained combining:
  - a self transition
  - a number of explicit transitions between states having the same emission probabilities (green in the figure)
  - a final transition to a “different” state (red in the figure)
It is possible to add constraints on the maximum length of paths within the “same” state. This is obtained combining:

- a number of explicit transitions between states having the same emission probabilities (green in the figure)
- shortcut transitions between each green state and a different (red) state
HMM model structure

Silent states

- Sometimes shortcut connections are needed all along the model.
- This happens when arbitrary length deletions should be allowed in sequences (e.g. biological sequences).
- Connecting each state to all possible successors produces a large number of parameters.
- A possible alternative consists of adding silent states, i.e. states that do not emit symbols.
A sequence of *silent* states is added (squares), parallel to the one of normal (emitting) states.

- Each normal state has a transition to the following normal state, and a transition to the following silent state.
- Each silent state has a transition to the following silent state, and a transition to the following normal state.
Standard forward, backward and Viterbi algorithms can be easily generalized to include silent states. The idea is that whenever a path passes from a silent state, it does not emit an observation, thus the time instant $t$ is not incremented (but the path will have skipped a certain emitting state). However passing from a silent state does change the probability of observation/state sequence pair and has to be counted.
HMM model structure

learning the structure

1. When no domain information is available to decide a reasonable structure for the model, structure has to be learned from data.
2. The approaches add or remove states and transitions trying to adapt the topology to the data.
3. A possible approach consists of (Stolcke and Omohundro, 1994):
   1. creating a model with a single path for each training sequence
   2. repeatedly merging states trying to maximize the posterior probability of the model given the data
4. where model prior should favour simpler topologies
Biological macro-molecules such as proteins, DNA, RNA are sequences of small molecules, which eventually fold into a 3D structure.

Their primary (1D) structure can be represented as a sequence of letters (indicating small molecules) from a certain alphabet (20 amino-acids for proteins, 4 nucleotides for DNA or RNA)

Evolutionary related biological molecules are obtained from a common ancestor by repeated mutations.

There are three main types of mutations:
- substitution of the letter at a certain position with a different one
- insertion of a certain sequence between two successive positions
- deletion of a certain subsequence
Discovering evolutionary relationships between related sequences is a main task in computational molecular biology. For a pair of sequences, the task can be formalized as finding an optimal alignment between them. Certain letter substitutions are more likely than others, if maintaining part of the characteristics of the replaced letter. For instance, replacing an aspartic acid (D) with a glutamic acid (E) retains the negative charge. An alignment can also include gaps, to account for insertions or deletions in one of the sequences wrt the common ancestor.
Alignment example

- Given two sequences **HEAGAWGHEE** and **PAWHEAE**

- A possible alignment could be the following (where “-” indicates a gap in the alignment):

  HEAGAWGHE-E
  --P-AW-HEAE

- The probability of a certain alignment depends on the probabilities of the matches between aligned positions, and the probabilities of the gaps
Alignment scores

- Let $p_{ab}$ be the probability that $a$ and $b$ have been independently derived from a common ancestor $c$ (related to their compatibility)
- Let $q_a$ be the probability of observing a certain letter $a$ (independently on any other letter)
- Let $g$ be the probability of starting an insertion (gap in the alignment)
- Let $e$ be the probability of extending an insertion (an insertion of a sequence of $n$ letters is more likely than $n$ separate insertions of a single letter)
Probabilistic model

- We want to build a model for the joint probability of two sequences $x$ and $y$ being derived from a common ancestor.
- We assume that mutations at different sites are independent of each other.
- We can build a HMM with three states:
  - $M$ a match state representing aligned positions (possibly implying a substitution)
  - $X$ an insertion state for sequence $x$
  - $Y$ an insertion state for sequence $y$
## Emission pairs

- Let $i, j$ indicate the current positions in sequence $x$ and $y$ respectively, while scanning the alignment.
- In state $X$, only sequence $x$ should emit a symbol $x_i$.
- In state $Y$, only sequence $y$ should emit a symbol $y_j$.
- In state $M$, both sequences emit symbols $x_i$ and $y_j$ respectively.

## Pair-HMM

- A pair-HMM is a special HMM in which states emit *pairs* of symbols.
Pair-HMM

Hidden Markov Models
Pair-HMM

Hidden Markov Models
For semplicity, we will assume that $M$ is also the initial state for position (0,0).

In this case the initial state does not emit anything, but simply serves to transit to the following state.
Pair-HMM

Joint probability example

Let’s compute joint probability of the alignment:

\[
x \quad \text{HEAGAWGHE-E}
\]
\[
y \quad \text{--P-AW-HEAE}
\]

The sequence of states (ignoring the initial one) is:

\[
\text{HEAGAWGHE-E}
\]
\[
\text{--P-AW-HEAE}
\]
\[
\text{XXMXMMXMMYM}
\]

The joint probability is computed as:

\[
g \cdot q_H \cdot e \cdot q_E \cdot (1 - e) \cdot p_{AP} \cdot g \cdot q_G \cdot (1 - e) \cdot p_{AA} \cdot (1 - 2g) \cdot p_{WW} \cdot g \cdot q_G \cdot (1 - e) \cdot p_{HH} \cdot (1 - 2g) \cdot p_{EE} \cdot g \cdot q_A \cdot (1 - e) \cdot p_{EE}
\]
Viterbi decoding

1. Initialization \((i = 1, \ldots, n, j = 1, \ldots, m, n = |x|, m = |y|)\):
   \[
   \delta_M(0, 0) = 1 \\
   \delta_X(i, 0) = \delta_Y(i, 0) = 0 \quad \forall \quad i, j
   \]

2. Induction:
   \[
   \delta_M(i, j) = \max \left\{ \begin{array}{l}
   (1 - 2g)\delta_M(i - 1, j - 1) \\
   (1 - e)\delta_X(i - 1, j - 1) \\
   (1 - e)\delta_Y(i - 1, j - 1)
   \end{array} \right. \\
   \delta_X(i, j) = q_{x_i} \max \left\{ \begin{array}{l}
   g \cdot \delta_M(i - 1, j) \\
   e \cdot \delta_X(i - 1, j)
   \end{array} \right. \\
   \delta_Y(i, j) = q_{y_j} \max \left\{ \begin{array}{l}
   g \cdot \delta_M(i, j - 1) \\
   e \cdot \delta_Y(i, j - 1)
   \end{array} \right.
   \]

3. Termination:
   \[
   P^* = \max(\delta_M(n, m), \delta_X(n, m), \delta_Y(n, m))
   \]
Optimal path

As for decoding in standard HMM, the \textit{max} previous state at each position is saved in an auxiliary variable for recovering the optimal path:

\begin{enumerate}
  \item Iteration:
    \begin{align*}
      \psi_M(i, j) &= \operatorname{argmax}_{M,X,Y} \left\{ (1 - 2g) \delta_M(i - 1, j - 1), (1 - e) \delta_X(i - 1, j - 1), (1 - e) \delta_Y(i - 1, j - 1) \right\} \\
      \psi_X(i, j) &= \operatorname{argmax}_{M,X} \left\{ g \cdot \delta_M(i - 1, j), e \cdot \delta_X(i - 1, j) \right\} \\
      \psi_Y(i, j) &= \operatorname{argmax}_{M,Y} \left\{ g \cdot \delta_M(i, j - 1), e \cdot \delta_Y(i, j - 1) \right\}
    \end{align*}
  \item Terminate:
    \begin{align*}
      \omega^*(n, m) &= \operatorname{argmax}_{M,X,Y} (\delta_M(n, m), \delta_X(n, m), \delta_Y(n, m))
    \end{align*}
\end{enumerate}
Pair-HMM

Forward procedure

- Simply replace max with sum in decoding.

Backward procedure

1. Initialization:

\[ \beta_M(n, m) = \beta_X(n, m) = \beta_Y(n, m) = 1 \]
\[ \beta_*(i, m + 1) = \beta_*(n + 1, j) = 0 \quad \forall \ i, j \]

2. Induction:

\[ \beta_M(i, j) = (1 - 2g) p_{x_{i+1}y_{j+1}} \beta_M(i + 1, j + 1) + 
\]
\[ + g (q_{x_{i+1}} \beta_X(i + 1, j) + q_{y_{j+1}} \beta_Y(i, j + 1)) \]
\[ \beta_X(i, j) = (1 - e) p_{x_{i+1}y_{j+1}} \beta_M(i + 1, j + 1) + e q_{x_{i+1}} \beta_X(i + 1, j) \]
\[ \beta_Y(i, j) = (1 - e) p_{x_{i+1}y_{j+1}} \beta_M(i + 1, j + 1) + e q_{y_{j+1}} \beta_Y(i, j + 1) \]
Motivation

- Biological sequences are typically grouped into families with a certain functionality
- A relevant task is that of detecting whether a target sequence belongs to a certain family
- This could be done aligning the sequence to each of the sequences from the family
- However, pairwise alignments alone can miss cases of distantly related sequences
- A better way to detect such relationship would be:
  1. building a model of the family
  2. testing whether the target sequence is compatible with the model
A multiple alignment consists of the simultaneous alignment of a set of sequences.

All sequences from a certain family could be aligned to form a multiple alignment representing the family.

The family model should be a compact probabilistic representation of such multiple alignment.
Multiple alignment: example for the globin family
Profile HMM

Note

- For simplicity of exposition we will focus on protein sequences
- The small molecules composing proteins are called *amino-acids* (each corresponds to a letter)
- When appearing in a protein sequence, amino-acids are called *residues* (because part of them is eliminated when the attach to form the sequence)
Dealing with ungapped regions

- Large portions of multiple alignments for a protein family consist of ungapped sequences of residues.
- Each position in such regions has a certain amino-acid profile, representing the frequencies with which each amino-acid occurs in the column of the alignment.
- By normalizing such profiles, it is possible to derive a probability of observing a certain residue *in that position*. 
Probabilistic model for ungapped regions

- Each position in the region can be modelled with a match state with position specific emission probabilities.
- The whole region can be modelled as a sequence of match states, with transitions only between successive states.
- Beginning and end of the region can be modelled with special non-emitting begin and end states.
Dealing with gaps

- Gaps in the alignment tend to occur at certain positions (i.e. gaps align columnwise)
- Gaps can be dealt with by modelling the two type of corresponding modifications:
  - insertions of a sequence of residues
  - deletions of a sequence of residues
An insertion should be modelled with a specific insertion state $I_j$ (represented as a diamond). As insertions in different positions have different probabilities, transition probabilities should be position specific. An insertion state should also have a self transition to account for insertions of sequences of residues. Emission probabilities could instead be set for all insertion states equal to the background probability $q_a$ of observing a certain amino-acid $a$ in an arbitrary sequence.
Probabilistic model with deletions

- Deletions should be modelled as special *silent* states $D$ which do not emit symbols (represented as a square).
- Allowing self transitions as in insertions would complicate inference algorithms.
- Sequences of deletions are instead modelled as sequences of deletion states.
- This also allows to specify different transition probabilities between deletion states.
Note

- We allow direct transitions between insertion and deletion states

- These situations are quite rare, but leaving such transitions out would give zero probability to these cases
Parameter estimation

- We assume a multiple alignment profile for the family of interest is available (created with multiple alignment algorithms, possibly relying on 3D information).

- We need to estimate transition probabilities between states, and emission probabilities for match states (those for insertion states are set to background probabilities for arbitrary sequences).

- We first decide which positions in the alignment correspond to match states, and which to insertions or deletions:
  - A reasonable approach is that if half of the column elements in a position are gaps, the position is not a match state.

- This allows us to turn our alignment into a fully observed set of training examples: probabilities can be estimated from counts.
Parameter estimation: examples

- Non-zero emission probabilities for match state $M_3$:
  \[ e_{M_3}(V) = \frac{5}{7} \quad e_{M_3}(F) = \frac{1}{7} \quad e_{M_3}(I) = \frac{1}{7} \]

- Non-zero transition probabilities from match state $M_3$:
  \[ a_{M_3M_4} = \frac{6}{7} \quad a_{M_3D_4} = \frac{1}{7} \]

- Non-zero transition probabilities from match state $M_5$:
  \[ a_{M_5M_6} = \frac{6}{7} \quad a_{M_5I_5} = \frac{1}{7} \]
Parameter estimation: adding pseudocounts

- All transitions and emissions never observed in the multiple alignment will be set to zero using only counts.
- This can be a problem if an unsufficient number of examples is available (i.e. always)
- A simple solution consists of adding a non-zero prior probability for any transition or emission, to be combined to the counts observed on data
- Such prior probability can be thought of coming from pseudocounts of hypothetical observations of emissions/translations
- The simplest pseudocount (Laplace smoother) consists of adding a single hypothetical observation of any possible emission/transition
Profile HMM

Viterbi decoding

\[
\delta_{M_j}(t) = e_{M_j}(x_t) \max \left\{ \begin{array}{l}
a_{M_{j-1}M_j}\delta_{M_{j-1}}(t-1) \\
a_{I_{j-1}M_j}\delta_{I_{j-1}}(t-1) \\
a_{D_{j-1}M_j}\delta_{D_{j-1}}(t-1) \\
\end{array} \right.
\]

\[
\delta_{I_j}(t) = e_{I_j}(x_t) \max \left\{ \begin{array}{l}
a_{M_jI_j}\delta_{M_j}(t-1) \\
a_{I_jI_j}\delta_{I_j}(t-1) \\
a_{D_jI_j}\delta_{D_j}(t-1) \\
\end{array} \right.
\]

\[
\delta_{D_j}(t) = \max \left\{ \begin{array}{l}
a_{M_{j-1}D_j}\delta_{M_{j-1}}(t) \\
a_{I_{j-1}D_j}\delta_{I_{j-1}}(t) \\
a_{D_{j-1}D_j}\delta_{D_{j-1}}(t) \\
\end{array} \right.
\]

Hidden Markov Models
Profile HMM

**Note**

- We decrease the time instant $t$ in the right hand side whenever a symbol is emitted by the current state (i.e. not in silent states $D$).
- We decrease the state identifier $j$ in the right hand side whenever the state corresponds to a position in the model sequence (i.e. not in insertion states $I$).
Viterbi decoding: match state

$$\delta_{M_j}(t) = e_{M_j}(x_t) \max \left\{ a_{M_{j-1}M_j} \delta_{M_{j-1}}(t - 1), a_{I_{j-1}M_j} \delta_{I_{j-1}}(t - 1), a_{D_{j-1}M_j} \delta_{D_{j-1}}(t - 1) \right\}$$
Viterbi decoding: insert state

\[ \delta_{I_j}(t) = e_{I_j}(x_t) \max \left\{ a_{M_jI_j} \delta_{M_j}(t - 1), \right. \]
\[ \quad a_{I_jI_j} \delta_{I_j}(t - 1), \]
\[ \quad a_{D_jI_j} \delta_{D_j}(t - 1) \right\} \]
Viterbi decoding: delete state

\[ \delta_{D_j}(t) = \max \left\{ a_{M_{j-1}D_j} \delta_{M_{j-1}}(t), a_{I_{j-1}D_j} \delta_{I_{j-1}}(t), a_{D_{j-1}D_j} \delta_{D_{j-1}}(t) \right\} \]
Other algorithms

- Forward procedure is obtained from Viterbi replacing max with sum.
- Backward procedure can be derived in similar fashion (we usually don’t need it for parameter estimation as training is done using counts).

Profile HMM
Other HMM applications

- Speech recognition
- Phylogenetic tree modeling
- Handwriting recognition
- Intrusion detection systems
- Natural language processing
Resources

References


Software

- Hidden Markov Model (HMM) Toolbox for Matlab
- HMMER: profile HMM
  - http://hmm.janelia.org/
- SAM: Sequence Alignment and Modeling System (includes profile HMM)
  - http://compbio.soe.ucsc.edu/sam.html