Supervised Classification

• Classifiers so far:
  – Decision tree, nearest neighbor, neural nets, naïve bayes
  – Classification is done for instances separately

• What if there is a dependency between instances?
  – e.g., Label each word in a sentence with its part-of-speech tag
A Markov System

Has $N$ states, called $s_1, s_2, \ldots, s_N$

There are discrete timesteps, $t=0, t=1, \ldots$

$N=3$
$t=0$
$q_t=q_0=s_3$

On the $t$'th timestep the system is in exactly one of the available states. Call it $q_t$

Note: $q_t \in \{s_1, s_2, \ldots, s_N\}$
A Markov System

Has $N$ states, called $s_1, s_2, \ldots, s_N$

There are discrete timesteps, $t=0, t=1, \ldots$

On the $t$'th timestep the system is in exactly one of the available states. Call it $q_t$

Note: $q_t \in \{s_1, s_2, \ldots, s_N\}$

Between each timestep, the next state is chosen randomly.

$N = 3$
$t=1$
$q_t = q_1 = s_2$

The current state determines the probability distribution for the next state.
A Markov System

Has $N$ states, called $s_1, s_2, \ldots, s_N$
There are discrete timesteps, $t=0, t=1, \ldots$
On the $t$'th timestep the system is in exactly one of the available states. Call it $q_t$
Note: $q_t \in \{s_1, s_2, \ldots, s_N\}$
Between each timestep, the next state is chosen randomly.
The current state determines the probability distribution for the next state.

Formally: First-order Markov Model

A set of states $Q = q_1, q_2, \ldots q_N$, the state at time $t$ is $q_t$
Current state only depends on previous state

$$P(q_t | q_1 \ldots q_{t-1}) = P(q_t | q_{t-1})$$

Transition probability matrix $A$

$$a_{ij} = P(q_t = j | q_{t-1} = i) \quad 1 \leq i, j \leq N$$

Special initial probability vector $\pi$

$$\pi_i = P(q_1 = i) \quad 1 \leq i \leq N$$

Constraints:

$$\sum_{j=1}^{N} a_{ij} = 1; \quad 1 \leq i \leq N \quad \sum_{j=1}^{N} \pi_j = 1$$
Probability of a Sequence of States

\[ P(s_1s_2s_3\ldots s_T) = P(s_1) \times P(s_2 \mid s_1) \times \ldots \times P(s_T \mid s_1s_2s_3\ldots s_{T-1}) \]

\[ = P(s_1) \times P(s_2 \mid s_1) \times \ldots \times P(s_T \mid s_{T-1}) \]

\[ = \pi \prod_{t=1}^{T-1} a_{s_ts_{t+1}} \]

Note: I’m being sloppy about the starting time 0 or 1.

What is \( P(q_t = s) \)? Slow, Stupid Answer

Step 1: Work out how to compute \( P(Q) \) for any path \( Q \) of length \( t \) (i.e., \( Q = q_1 q_2 q_3 \ldots q_t \))

\[ P(q_1 q_2 \ldots q_t) = P(q_1)P(q_2|q_1)P(q_3|q_2)\ldots P(q_t|q_{t-1}) \]

Step 2: Use this knowledge to get \( P(q_t = s) \)

\[ P(q_t = s) = \sum_{Q=\text{Paths of length } t \text{ that end in } s} P(Q) \]

Computation is exponential in \( t \)
What is $P(q_t = s)$? Clever Answer

• For each state $s_i$,
  define $p_t(i) = \text{Prob. state is } s_i \text{ at time } t$
  $= P(q_t = s_i)$

• Easy to do inductive definition

$\forall i \quad p_0(i) =$

$\forall j \quad p_{t+1}(j) = P(q_{t+1} = s_j) =$

Or get probs from $\pi$
What is $P(q_t = s)$? Clever Answer

• For each state $s_i$
  
  define $p_t(i) = \text{Prob. state is } s_i \text{ at time } t = P(q_t = s_i)$

• Easy to do inductive definition

$$
\forall i \quad p_0(i) = \begin{cases} 
1 & \text{if } s_i \text{ is the start state} \\
0 & \text{otherwise}
\end{cases}
$$

$$
\forall j \quad p_{t+1}(j) = P(q_{t+1} = s_j) = 
\sum_{i=1}^{N} P(q_{t+1} = s_j \wedge q_t = s_i) = 
\sum_{i=1}^{N} P(q_{t+1} = s_j | q_t = s_i) P(q_t = s_i) = 
\sum_{i=1}^{N} a_{ij} p_t(i)
$$

Or get probs from $\pi$

Remember,

$$a_{ij} = P(q_{t+1} = s_j | q_t = s_i)$$
What is $P(q_t = s)$? Clever Answer

• For each state $s_i$, define $p_t(i) = \text{Prob. state is } s_i \text{ at time } t = P(q_t = s_i)$

• Easy to do inductive definition

  $\forall i \quad p_0(i) = \begin{cases} 1 & \text{if } s_i \text{ is the start state} \\ 0 & \text{otherwise} \end{cases}$

  $\forall j \quad p_{t+1}(j) = P(q_{t+1} = s_j) = \sum_{i=1}^{N} P(q_{t+1} = s_j \land q_t = s_i) = \sum_{i=1}^{N} P(q_{t+1} = s_j \mid q_t = s_i) P(q_t = s_i) = \sum_{i=1}^{N} a_j p_i(i)$

Cost of computing $P_t(i)$ for all states $S_i$ is now $O(t N^2)$

The stupid way was $O(N^t)$

This was a simple example

It uses a trick, called **Dynamic Programming**.
Example

Initialize $p_t(j)

\[ p_t(j) = \sum_{i=1}^{N} \alpha_{ij} p_{t-1}(i) \]

<table>
<thead>
<tr>
<th>$p_1(3)$</th>
<th>$p_2(3)$</th>
<th>$p_3(3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1(2)$</td>
<td>$p_2(2)$</td>
<td>$p_3(2)$</td>
</tr>
<tr>
<td>$p_1(1)$</td>
<td>$p_2(1)$</td>
<td>$p_3(1)$</td>
</tr>
</tbody>
</table>

Assume that the state is not directly observable but that the observation is a probabilistic function of the state.

For example, assume that there are $N$ urns each containing balls of different colors with a specific distribution for each urn. An initial urn is chosen at random and a ball is drawn, its color announced and replaced. The next urn is chosen randomly based on the current urn and the process repeats.
HMM Formalism

\{S, O, \Pi, A, B\}

\Pi = \{\pi_i\} are the initial state probabilities

\(A = \{a_{ij}\}\) are the state transition probabilities

\(B = \{b_{ik}\}\) are the observation probabilities

HMM Formal Definition

An HMM, \(\lambda\), is a 5-tuple consisting of

- \(N\) the number of states
- \(M\) the number of possible observations

- \(\{\pi_1, \pi_2, \ldots, \pi_N\}\) The starting state probabilities:
  \[P(q_0 = S_i) = \pi_i\]

- \(\{a_{ij}\}\) The state transition probabilities:
  \[P(q_t = S_j | q_{t-1} = S_i) = a_{ij}\]

- \(\{b_{ik}\}\) The observation probabilities:
  \[P(O_t = k | q_{t} = S_i) = b_{ik}\]
**HMM Notation**

The states are labeled $S_1 \ S_2 \ldots \ S_N$.

For a particular trial:
- Let $T$ be the number of observations.
- $T$ is also the number of states passed through.

$O = O_1 \ O_2 \ldots O_T$ is the sequence of observations.
$Q = q_1 \ q_2 \ldots q_T$ is the notation for a path of states.

---

**HMMs: An Example**

Start randomly in state 1 or 2.
Choose one of the output symbols in each state at random.

\[
\begin{array}{c}
N = 3 \\
M = 3 \\
\pi_1 = 1/2 \\
\pi_2 = 1/2 \\
\pi_3 = 0 \\
a_{11} = 0 \\
a_{12} = 1/3 \\
a_{13} = 2/3 \\
a_{21} = 1/3 \\
a_{22} = 0 \\
a_{23} = 2/3 \\
a_{31} = 1/3 \\
a_{32} = 1/3 \\
a_{33} = 1/3 \\
b_1(X) = 1/2 \\
b_1(Y) = 1/2 \\
b_1(Z) = 0 \\
b_2(X) = 0 \\
b_2(Y) = 1/2 \\
b_2(Z) = 1/2 \\
b_3(X) = 1/2 \\
b_3(Y) = 0 \\
b_3(Z) = 1/2 \\
\end{array}
\]
HMMs: An Example

Start randomly in state 1 or 2

Choose one of the output
symbols in each state at
random.

Let's generate a sequence of
observations:

\[
\begin{align*}
q_0 & = \_ & o_0 & = \_ \\
q_1 & = \_ & o_1 & = \_ \\
q_2 & = \_ & o_2 & = \_
\end{align*}
\]

\[
\begin{align*}
q_0 & = \_ & o_0 & = \_ \\
q_1 & = \_ & o_1 & = \_ \\
q_2 & = \_ & o_2 & = \_
\end{align*}
\]

\[
\begin{align*}
q_0 & = \_ & o_0 & = \_ \\
q_1 & = \_ & o_1 & = \_ \\
q_2 & = \_ & o_2 & = \_
\end{align*}
\]
HMMs: An Example

Let's generate a sequence of observations:

Start randomly in state 1 or 2
Choose one of the output symbols in each state at random.

Goto $S_3$ with probability 2/3 or $S_2$ with prob. 1/3

HMMs: An Example

Let's generate a sequence of observations:

Start randomly in state 1 or 2
Choose one of the output symbols in each state at random.

50-50 choice between Z and X
HMMs: An Example

N = 3
M = 3
π₁ = 1/2
π₂ = 1/2
π₃ = 0

a₁₁ = 0
a₁₂ = 1/3
a₁₃ = 2/3
a₂₁ = 1/3
a₂₂ = 0
a₂₃ = 2/3
a₃₁ = 1/3
a₃₂ = 1/3
a₃₃ = 1/3

b₁(X) = 1/2
b₁(Y) = 1/2
b₁(Z) = 0
b₂(X) = 0
b₂(Y) = 1/2
b₂(Z) = 1/2
b₃(X) = 1/2
b₃(Y) = 0
b₃(Z) = 1/2

Start randomly in state 1 or 2
Choose one of the output symbols in each state at random.

Let's generate a sequence of observations:

Each of the three next states is equally likely

Start randomly in state 1 or 2
Choose one of the output symbols in each state at random.

Let's generate a sequence of observations:

50-50 choice between Z and X
HMMs: An Example

Start randomly in state 1 or 2
Choose one of the output symbols in each state at random.

Let’s generate a sequence of observations:

This is what the observer has to work with …

N = 3
M = 3
π₁ = 1/2     π₂ = 1/2     π₃ = 0

a₁₁ = 0     a₁₂ = 1/3     a₁₃ = 2/3
a₂₁ = 1/3     a₂₂ = 0     a₂₃ = 2/3
a₃₁ = 1/3     a₃₂ = 1/3     a₃₃ = 1/3

b₁(X) = 1/2     b₁(Y) = 1/2     b₁(Z) = 0
b₂(X) = 0     b₂(Y) = 1/2     b₂(Z) = 1/2
b₃(X) = 1/2     b₃(Y) = 0     b₃(Z) = 1/2

q₀ = S₁     O₀ = X
q₁ = S₃     O₁ = X
q₂ = S₃     O₂ = Z
Three Problems in Hidden Markov Models

• **Question 1: Evaluation**
  What is the probability of the observation sequence $O_1 O_2 \ldots O_T$, $P(O_1 O_2 \ldots O_T | \lambda)$?

• **Question 2: Most Probable Path**
  Given $O_1, O_2, \ldots, O_T$, what is the most probable path that I took?

• **Question 3: Learning HMMs:**
  Given $O_1, O_2, \ldots, O_T$, what is the maximum likelihood HMM that could have produced this string of observations?

Basic Operations in HMMs

For an observation sequence $O = O_1, \ldots, O_T$, the three basic HMM operations are:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Algorithm</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calculating $P(O_1 O_2 \ldots O_T</td>
<td>\lambda)$</td>
<td>Forward</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Backward</td>
</tr>
<tr>
<td>Inference:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computing $Q^* = \arg\max P(Q</td>
<td>O)$</td>
<td>Viterbi decoding</td>
</tr>
<tr>
<td>Learning:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computing $\lambda^* = \arg\max P(O</td>
<td>\lambda)$</td>
<td>Baum-Welch (EM)</td>
</tr>
</tbody>
</table>

$T = \#$ timesteps, $N = \#$ states
Computing prob. of a series of observations

What is \( P(O) = P(O_1 \; O_2 \; O_3) = P(O_1 = X \; O_2 = X \; O_3 = Z)? \)

Slow, stupid way:

\[
P(O) = \sum_{Q \text{ paths of length 3}} P(O \land Q) = \sum_{Q \text{ paths of length 3}} P(O \mid Q)P(Q)
\]

How do we compute \( P(Q) \) for an arbitrary path \( Q? \)

How do we compute \( P(O \mid Q) \) for an arbitrary path \( Q? \)

P(Q) = P(q_1, q_2, q_3)
= P(q_1) P(q_2 \mid q_1) P(q_3 \mid q_2, q_1) (\text{chain rule})
= P(q_1) P(q_2 \mid q_1) P(q_3 \mid q_2) \text{ (why?)}

Example in the case \( Q = S_1 \; S_3 \; S_3 \):
= 1/2 \times 2/3 \times 1/3 = 1/9
Computing prob. of a series of observations

What is \( P(O) = P(O_1 O_2 O_3) = P(O_1 = X \land O_2 = X \land O_3 = Z) \)?

Slow, stupid way:

\[
P(O) = \sum_{Q: \text{Paths of length 3}} P(O \land Q) = \sum_{Q: \text{Paths of length 3}} P(O|Q)P(Q)
\]

How do we compute \( P(Q) \) for an arbitrary path \( Q \)?

How do we compute \( P(O|Q) \) for an arbitrary path \( Q \)?

Example in the case \( Q = S_1 S_3 S_3 \):

\[
P(O|Q) = P(O_1|q_1)P(O_2|q_2)P(O_3|q_3) = 1/2 \times 1/2 \times 1/2 = 1/8
\]

P(\( O \)) would need 27 P(\( Q \)) computations and 27 P(\( O|Q \)) computations

A sequence of 20 observations would need \( 3^{20} = 3.5 \times 10^6 \) computations and

3.5 billion P(\( O|Q \)) computations

So let's be smarter.
The Prob. of a given series of observations, non-exponential-cost-style

Given observations $O_1 O_2 \ldots O_T$
Define
\[ \alpha_t(i) = \Pr(O_1 O_2 \ldots O_t \land q_t = S_i \mid \lambda) \quad \text{where } 1 \leq t \leq T \]

\[ \alpha_t(i) = \text{Probability that, in a random trial,} \]
\[ \cdot \text{We'd have seen the first } t \text{ observations} \]
\[ \cdot \text{We'd have ended up in } S_i \text{ as the } t'\text{th state visited.} \]

Defining $\alpha_t(i)$ Recursively

\[ \alpha_t(i) = \Pr(O_1 O_2 \ldots O_t \land q_t = S_i \mid \lambda) \]

\begin{align*}
\alpha_t(i) &= \Pr(O_t \land q_t = S_i) \\
&= \Pr(g_t = S_i) \Pr(o_t \mid q_t = S_i) \\
&= \text{what?} \\
\alpha_{t+1}(j) &= \Pr(o_{t+1} \land q_{t+1} = S_j) \\
&= \sum_i \Pr(o_{t+1} \land q_{t+1} = S_j) \Pr(o_{t+1} \land q_{t+1} = S_j) \\
&= \sum_i \Pr(o_{t+1} = S_j) \Pr(o_{t+1} \land q_{t+1} = S_j) \\
&= \sum_i \Pr(o_{t+1} = S_j) \Pr(q_{t+1} = S_j) \alpha_t(i) \\
&= \sum_i \Pr(q_{t+1} = S_j) \Pr(o_{t+1} = S_j) \alpha_t(i) \\
&= \sum_i \alpha_t(i) \Pr(o_{t+1} \mid q_{t+1} = S_j) \\
&= \sum_i \alpha_t(i) b_j(o_{t+1}) \alpha_{t}(i)
\end{align*}
Forward Procedure

Computation of $\alpha_t(j)$ by summing all previous values $\alpha_{t-1}(i)$ for all $i$

In our example ...

$$\alpha_t(i) = p(O_1O_2..O_t \land q_t = S_j^z)$$
$$\alpha_t(i) = b_j(O_t)\pi_i$$
$$\alpha_{t+1}(j) = \sum_i a_{ij}b_j(O_{t+1})\alpha_t(i)$$

We saw $O_1O_2O_3 = X \times X$
Easy Questions

We can cheaply compute
\[ \alpha_t(i) = P(O_1 O_2 \ldots O_t \land q_t = S_i) \]

How can we cheaply compute
\[ P(O_1 O_2 \ldots O_t) ? \]

(How) can we cheaply compute
\[ P(q_t = S_i \mid O_1 O_2 \ldots O_t)? \]

The Forward Procedure

Forward variables are calculated as follows:
\[ \alpha_t(i) = P(O_i O_{i+1} \ldots O_T \mid q_i = S_i) \]

Initialization:
\[ \alpha_1(i) = b_i(O_1) \pi_i \]

Induction:
\[ \alpha_{t+1}(j) = \sum_i a_{ij} b_j(O_{t+1}) \alpha_t(i) \]

Total: \[ P(O_1 O_2 \ldots O_T \mid \lambda) = \Sigma_{i=1,N} \alpha_T(i) \]

This algorithm requires \( O(N^2 T) \) multiplications --- much less than the direct method which takes \( O(T N^T) \)

Note: we will discuss backward procedure later
Three Problems in Hidden Markov Models

- Question 1: Evaluation
  What is the probability of the observation sequence $O_1O_2 \ldots O_T$, $P(O_1O_2 \ldots O_T | \lambda)$?

- Question 2: Most Probable Path
  Given $O_1O_2 \ldots O_T$, what is the most probable path that I took?

- Question 3: Learning HMMs:
  Given $O_1O_2 \ldots O_T$, what is the maximum likelihood HMM that could have produced this string of observations?

Most Probable Path Given Observations

What is the most probable path given $O_1O_2 \ldots O_T$ i.e.,
What is $\arg\max_Q p(Q|O_1O_2 \ldots O_T)$?

Slow, stupid answer:

$$\arg\max_Q p(Q|O_1O_2 \ldots O_T)$$

$$= \arg\max_Q \frac{p(O_1O_2 \ldots O_T | Q)p(Q)}{p(O_1O_2 \ldots O_T)}$$

$$= \arg\max_Q p(O_1O_2 \ldots O_T | Q)p(Q)$$
Efficient MPP Computation

We’re going to compute the following variables:

\[ \delta_t(i) = \max_{q_1q_2...q_{t-1}} P(q_1, q_2, .. q_{t-1}, q_t = S_i, O_1..O_t) \]

= The probability of the path of length t-1 with the maximum chance of doing all these things:

...OCCURRING
and
...ENDING UP IN STATE S_i
and
...PRODUCING OUTPUT O_1...O_t

Define: \( \text{mpp}_t(i) = \) that path

So: \( \delta_t(i) = \text{Prob}(\text{mpp}_t(i)) \)

---

The Viterbi Algorithm

\[ \delta_t(i) = \max_{q_1q_2...q_{t-1}} P(q_1q_2...q_{t-1} \land q_t = S_i \land O_1...O_t) \]

\[ \text{mpp}_t(i) = \max_{q_1q_2...q_{t-1}} P(q_1q_2...q_{t-1} \land q_t = S_i \land O_1...O_t) \]

\[ \delta_t(i) = \text{one choice } P(q_t = S_i \land O_t) \]
\[ = P(q_t = S_i)P(O_t|q_t = S_i) \]
\[ = \pi_{S_i}b_i(O_t) \]

Now, suppose we have all the \( \delta_t(i) \)'s and \( \text{mpp}_t(i) \)'s for all i.

HOW TO GET \( \delta_{t+1}(j) \) and \( \text{mpp}_{t+1}(j) \)?

\[ \text{mpp}_1(1) \]
\[ \text{mpp}_1(2) \]
\[ \vdots \]
\[ \text{mpp}_1(N) \]

\[ \text{mpp}_t(1) \]
\[ \text{mpp}_t(2) \]
\[ \vdots \]
\[ \text{mpp}_t(N) \]
The Viterbi Algorithm

The most prob path with last two states $S_i, S_j$ is the most prob path to $S_i$, followed by transition $S_i \rightarrow S_j$.

What is the prob of that path?

$$\delta_t(i) \times P(S_i \rightarrow S_j \wedge O_{t+1} \mid \lambda) = \delta_t(i) a_{ij} b_j (O_{t+1})$$

So, the most probable path to $S_j$ has $S_i^*$ as its penultimate state where $i^* = \arg \max_i \delta_t(i) a_{ij} b_j (O_{t+1})$.
The Viterbi Algorithm

The most prob path with last two states $S_i, S_j$

is

the most prob path to $S_i$,
followed by transition $S_i \rightarrow S_j$

What is the prob of that path?

$\delta(t)(i) \times P(S_i \rightarrow S_j \land O_{t+1} \mid \lambda) = \delta(t)(i) a_{ij} b_j(O_{t+1})$

So, the most probable path to $S_j$

$S_i^*$ as its penultimate state

where $i^* = \operatorname{argmax}_i \delta(t)(i) a_{ij} b_j(O_{t+1})$

Summary:

$\delta_{t+1}(j) = \delta_t(i^*) a_{ij} b_j(O_{t+1})$

$mpp_{t+1}(j) = mpp_t(i^*) S_i^*$

with $i^*$ defined to the left

---

The Viterbi Algorithm

- The Idea: Just like Forward, fold exponential paths into a simple trellis, so that all possible paths will remerge into $N$ states at every time slice.

- We define the viterbi probability as follows:

  $v_t(i) = P(o_0 o_1 \cdots o_t, q_{t-1}, q_t = i \Phi)$

  $v_t(i)$ is the probability that the HMM $\Phi$ is in state $i$ at time $t$ having generated partial observation $o_t$ by passing through the most likely state sequence $q_{t-1}$.

- We again compute it by induction:

  - Initialization:

    $v_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N$

    $b_{t_1}(i) = 0$

  - Induction:

    $v_t(j) = \max_{1 \leq i \leq N} v_{t-1}(i) a_{ij} b_{j}(o_t)$,
The Viterbi Algorithm

\[ 2 \leq t \leq T, 1 \leq j \leq N \]
\[ b_t(j) = \left[ \arg \max_{1 \leq i \leq N} v_{t-1}(i) a_{ij} \right], \]
\[ 2 \leq t \leq T, 1 \leq j \leq N \]

- Termination: The best score is \( \max_{1 \leq i \leq N} v_T(i) \)

\[ q_T^* = \arg \max_{1 \leq i \leq N} b_T(i) \]

- Backtracking

\[ q_t^* = b_{t+1}(q_{t+1}^*) ; t = T - 1, T - 2, ..., 1 \]

\[ Q^* = (q_1^*, q_2^*, ..., q_T^*) \text{ is the best state sequence} \]
Example

We saw $O_1 \ O_2 \ O_3 = X \ Z \ Y$

What’s Viterbi Used for?

Classic example: Speech recognition

Goal: Signal $\rightarrow$ words

An HMM approach:
  $\rightarrow$ observable is signal
  $\rightarrow$ Hidden state is phone or part of word formation

What is the most probable word given this signal?

Part-of-speech tagging:
  Given a sentence (observations), find the most likely POS tag sequence.
What You Should Know

• What is an HMM?
• Computing (and defining) \( \alpha_t(i) \)
• The Viterbi algorithm
• To be happy with the kind of math and analysis needed for HMMs
• Further reading: