CS 1675: Intro to Machine Learning
Probabilistic Graphical Models

Prof. Adriana Kovashka
University of Pittsburgh
November 27, 2018
Plan for This Lecture

• Motivation for probabilistic graphical models
• Directed models: Bayesian networks
• Undirected models: Markov random fields (briefly)
• Directed models for sequence classification: Hidden Markov models
Probabilities: Example Use

Apples and Oranges
Marginal, Joint, Conditional

Marginal Probability

\[ p(X = x_i) = \frac{c_i}{N}. \]

Joint Probability

\[ p(X = x_i, Y = y_j) = \frac{n_{ij}}{N} \]

Conditional Probability

\[ p(Y = y_j | X = x_i) = \frac{n_{ij}}{c_i} \]
Sum and Product Rules

Sum Rule

\[ p(X = x_i) = \frac{c_i}{N} = \frac{1}{N} \sum_{j=1}^{L} n_{ij} \]

= \sum_{j=1}^{L} p(X = x_i, Y = y_j)

Product Rule

\[ p(X = x_i, Y = y_j) = \frac{n_{ij}}{N} = \frac{n_{ij}}{c_i} \cdot \frac{c_i}{N} \]

= \frac{p(Y = y_j | X = x_i)p(X = x_i)}{p(X = x_i)}
Independence

Marginal: $P$ satisfies $(X \perp Y)$ if and only if
\[
P(X=x, Y=y) = P(X=x) \cdot P(Y=y),
\]
\[
\forall x \in \text{Val}(X), \ y \in \text{Val}(Y)
\]

Conditional: $P$ satisfies $(X \perp Y \mid Z)$ if and only if
\[
P(X, Y \mid Z) = P(X \mid Z) \cdot P(Y \mid Z),
\]
\[
\forall x \in \text{Val}(X), \ y \in \text{Val}(Y), \ z \in \text{Val}(Z)
\]
Bayes’ Theorem

\[ p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)} \]

\[ p(X) = \sum_Y p(X|Y)p(Y) \]
Probabilistic Graphical Models

- It is sometimes desirable to have not only a prediction $y$ given features $x$, but a measure of confidence $P(y|x)$
- Let $x$ be a $d$-dim vector, each dim can take 2 values
- For each such vector $x$, $y=1$ or $y=0$
- We need statistics about how frequently $y=1$ occurs with each of $2^d$ possible feature vectors $\rightarrow 2^d$ parameters to estimate
- Thus we require an unrealistic amount of data to estimate parameters (probabilities); graphical models allow simplifying assumptions
A simple alternative: Naïve Bayes

• Assume all features are independent given the class i.e. $P(x \mid y) = \prod_d P(x_d \mid y)$

• Model $P(x_d=1 \mid y=1)$ and $P(x_d=1 \mid y=0)$, for all $d \rightarrow 2d$ parameters (as opposed to $2^d$)

• Then use Bayes rule to compute $P(y \mid x) = P(x \mid y) \ P(y) / Z = \prod_d P(x_d \mid y) \ P(y) / Z$

• Where $Z$ is a normalizing constant
Naïve Bayes example

If $J=1$

<table>
<thead>
<tr>
<th></th>
<th>Prob $W=1$</th>
<th>Prob $B=1$</th>
<th>Prob $C=1$</th>
<th>Prob $R=1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>0.8</td>
<td>0.2</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>False</td>
<td>0.3</td>
<td>0.5</td>
<td>0.3</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Waking Life  Borat  Cinema Paradiso  Requiem for a Dream
Advantages of Graphical Models

• If no assumption of independence is made, must estimate an exponential number of parameters
• If we assume all variables independent, efficient training and inference possible, but assumption too strong
• **Graphical models** use **graphs** over random variables to specify variable dependencies (relationships)
  • Allows for less restrictive independence assumptions while limiting the number of parameters that must be estimated
  • Allows some interpretability
  • **Bayesian networks**: Directed acyclic graphs indicate causal structure
  • **Markov networks**: Undirected graphs capture general dependencies

Adapted from Ray Mooney
Bayesian Networks

- Directed Acyclic Graph (DAG)
- Nodes are random variables
- Edges indicate causal influences

Diagram:
- Burglary
- Earthquake
- Alarm
- JohnCalls
- MaryCalls
Each node has a conditional probability table (CPT) that gives the probability of each of its values given every possible combination of values for its parents.

- Roots of the DAG that have no parents are given prior probabilities.
Aside: Naïve Bayes version
CPT Comments

• Probability of false not given since rows must add to 1
• Example requires 10 parameters rather than $2^5 - 1 = 31$ for specifying the full joint distribution
• Number of parameters in the CPT for a node is exponential in the number of parents
Bayes Net Inference

• Given known values for some **evidence variables**, determine the posterior probability of some **query variables**

• Example: Given that John calls, what is the probability that there is a Burglary?

John calls 90% of the time there is an Alarm and the Alarm detects 94% of Burglaries so people generally think it should be fairly high.

However, this ignores the prior probability of John calling.
Bayes Net Inference

• Example: Given that John calls, what is the probability that there is a Burglary?

John also calls 5% of the time when there is no Alarm. So over 1,000 days we expect 1 Burglary and John will probably call. However, he will also call with a false report 50 times on average. So the call is about 50 times more likely a false report:

\[
P(\text{Burglary} | \text{JohnCalls}) \approx 0.02
\]
Bayes Nets (not yet useful)

- No independence encoded

\[
p(x_1, \ldots, x_K) = p(x_K| x_1, \ldots, x_{K-1}) \ldots p(x_2|x_1)p(x_1)
\]

\[
p(a, b, c) = p(c|a, b)p(a,b) = p(c|a, b)p(b|a)p(a)
\]
Bayes Nets (formulation)

- More interesting: Some independences encoded

\[ p(x_1, \ldots, x_7) \]

General Factorization

\[ p(x) = \prod_{k=1}^{K} p(x_{k} | p_{a_{k}}) \]
Conditional Independence

\(a\) is independent of \(b\) given \(c\)

\[ p(a|b, c) = p(a|c) \]

Equivalently

\[ p(a, b|c) = p(a|b, c)p(b|c) = p(a|c)p(b|c) \]

Notation

\(a \perp b \mid c\)
Conditional Independence: Example 1

\[
p(a, b, c) = p(a|c)p(b|c)p(c)
\]

\[
p(a, b) = \sum_c p(a|c)p(b|c)p(c)
\]

\[
a \notin b \mid \emptyset
\]

Node \(c\) is “tail to tail” for path from \(a\) to \(b\):
**No independence** of \(a\) and \(b\) follows from this path
Conditional Independence: Example 1

Node $c$ is “tail to tail” for path from $a$ to $b$: Observing $c$ blocks the path thus making $a$ and $b$ conditionally independent.
Conditional Independence: Example 2

Node $c$ is “head to tail” for path from $a$ to $b$:

No independence of $a$ and $b$ follows from this path

$$p(a, b, c) = p(a)p(c|a)p(b|c)$$

$$p(a, b) = p(a) \sum_c p(c|a)p(b|c) = p(a)p(b|a)$$

$$a \not\perp b \mid \emptyset$$

$$\Sigma_c p(c|a) p(b|c) =$$

$$\Sigma_c p(c|a) p(b|c, a) = \text{ // next slide}$$

$$\Sigma_c p(b, c|a) =$$

$$p(b|a)$$
Node $c$ is “head to tail” for path from $a$ to $b$:
Observing $c$ blocks the path thus making $a$ and $b$ **conditionally independent**
Conditional Independence: Example 3

Node $c$ is “head to head” for path from $a$ to $b$: Unobserved $c$ blocks the path thus making $a$ and $b$ independent

Note: this is the opposite of Example 1, with $c$ unobserved.
Conditional Independence: Example 3

Node \( c \) is “head to head” for path from \( a \) to \( b \):
Observing \( c \) unblocks the path thus making \( a \) and \( b \) conditionally dependent

Note: this is the opposite of Example 1, with \( c \) observed.
Example: “Am I out of fuel?”

\[
\begin{align*}
\Pr(G = 1 | B = 1, F = 1) &= 0.8 \\
\Pr(G = 1 | B = 1, F = 0) &= 0.2 \\
\Pr(G = 1 | B = 0, F = 1) &= 0.2 \\
\Pr(G = 1 | B = 0, F = 0) &= 0.1
\end{align*}
\]

- \(B = \) Battery (0=flat, 1=fully charged)
- \(F = \) Fuel Tank (0=empty, 1=full)
- \(G = \) Fuel Gauge Reading
  (0=empty, 1=full)

\[
\begin{align*}
\Pr(B = 1) &= 0.9 \\
\Pr(F = 1) &= 0.9 \\
\text{and hence} \\
\Pr(F = 0) &= 0.1
\end{align*}
\]
Example: “Am I out of fuel?”

\[
p(G = 0|F = 0) = \sum_{B \in \{0,1\}} p(G = 0|B, F = 0)p(B) = 0.81
\]

\[
p(F = 0) = 0.1
\]

\[
p(G = 0) = \sum_{B \in \{0,1\}} \sum_{F \in \{0,1\}} p(G = 0|B, F)p(B)p(F) = 0.315
\]

\[
p(F = 0|G = 0) = \frac{p(G = 0|F = 0)p(F = 0)}{p(G = 0)} \approx 0.257
\]

Probability of an empty tank increased by observing \( G = 0 \).
Example: “Am I out of fuel?”

\[
p(F = 0|G = 0, B = 0) = \frac{p(G = 0|B = 0, F = 0)p(F = 0)}{\sum_{F \in \{0,1\}} p(G = 0|B = 0, F)p(F)} \approx 0.111
\]

Probability of an empty tank reduced by observing \( B = 0 \).
This is referred to as “explaining away”.

Chris Bishop
D-separation

- $A$, $B$, and $C$ are non-intersecting subsets of nodes in a directed graph.
- A path from $A$ to $B$ is blocked if it contains a node such that either
  a) the arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set $C$, or
  b) the arrows meet head-to-head at the node, and neither the node, nor any of its descendants, are in the set $C$.
- If all paths from $A$ to $B$ are blocked, $A$ is said to be d-separated from $B$ by $C$.
- If $A$ is d-separated from $B$ by $C$, the joint distribution over all variables in the graph satisfies $A \perp B \mid C$. 
D-separation: Example

\[ a \n\not\perp\!
\!
\not\mid\ b \mid c \]

\[ a \perp\!
\!
\perp\ b \mid f \]
Naïve Bayes

Conditioned on the class $z$, the distributions of the input variables $x_1, ..., x_D$ are independent.

Are the $x_1, ..., x_D$ marginally independent?
Bayes Nets vs. Markov Nets

• Bayes nets represent a subclass of joint distributions that capture non-cyclic causal dependencies between variables.
• A Markov net can represent any joint distribution.
Cliques and Maximal Cliques

\[ \text{Clique} \]

\[ \text{Maximal Clique} \]
Joint Distribution for a Markov Net

• The distribution of a Markov net is described in terms of a set of potential functions, $\psi_c$, for each clique $C$ in the graph.

• For each joint assignment of values to the variables in clique $C$, $\psi_c$ assigns a non-negative real value that represents the compatibility of these values.

Adapted from Ray Mooney
Joint Distribution for a Markov Net

\[ p(x) = \frac{1}{Z} \prod_C \psi_C(x_C) \]

where \( \psi_C(x_C) \) is the potential over clique \( C \) and

\[ Z = \sum_x \prod_C \psi_C(x_C) \]

is the normalization coefficient; note: \( M K \)-state variables \( \rightarrow K^M \) terms in \( Z \).

Energies and the Boltzmann distribution

\[ \psi_C(x_C) = \exp \left\{ -E(x_C) \right\} \]
Illustration: Image De-Noising

Original Image

Noisy Image
Illustration: Image De-Noising

$y_i$ in \{+1, -1\}: labels in noisy image (which we have),

$x_i$ in \{+1, -1\}: labels in noise-free image

(which we want to recover),

$i$ is the index over pixels

$$p(x, y) = \frac{1}{Z} \exp\{-E(x, y)\}$$

$$E(x, y) = \sum_i h x_i - \beta \sum_{\{i,j\}} x_i x_j - \eta \sum_i x_i y_i$$

Prior

Pixels are like their neighbors

Pixels of noisy and noise-free images are related

Adapted from Chris Bishop
Illustration: Image De-Noising

Noisy Image

Restored Image (ICM)
Aside: Graphical vs other models

- Some graphical models are generative i.e. model $p(x)$ not just $p(y|x)$
- Consider relationships of the features
- Somewhat interpretable
- We’ll also discuss one model appropriate for sequence classification (e.g. weather)
Classifying Connected Samples (Sequences)

- Standard classification problem assumes individual cases are disconnected and independent (i.i.d.: independently and identically distributed).
- Many problems do not satisfy this assumption and involve making many connected decisions which are mutually dependent.
Markov Chains

- A finite state machine with probabilistic state transitions.
- Makes Markov assumption that next state only depends on the current state and independent of previous history.
Markov Chains

• General joint probability distribution:

\[ p(x_1, \ldots, x_N) = \prod_{n=1}^{N} p(x_n|x_1, \ldots, x_{n-1}) \]

• First-order Markov chain:

\[ p(x_1, \ldots, x_N) = p(x_1) \prod_{n=2}^{N} p(x_n|x_{n-1}) \]
Markov Chains

- Second-order Markov chain:

\[ p(x_1, \ldots, x_N) = p(x_1)p(x_2|x_1) \prod_{n=3}^{N} p(x_n|x_{n-1}, x_{n-2}) \]
Hidden Markov Models

- Latent variables ($z$) satisfy Markov property
- Observed variables/predictions ($x$) do not
- Example: $x =$ words, $z =$ parts of speech

\[
p(x_1, \ldots, x_N, z_1, \ldots, z_N) = p(z_1) \left[ \prod_{n=2}^{N} p(z_n | z_{n-1}) \right] \left[ \prod_{n=1}^{N} p(x_n | z_n) \right]
\]
Example: Part Of Speech Tagging

- Annotate each word in a sentence with a part-of-speech marker.

John saw the saw and decided to take it to the table.
NNP VBD DT NN CC VBD TO VB PRP IN DT NN

Adapted from Ray Mooney
English Parts of Speech

• Noun (person, place or thing)
  – Singular (NN): dog, fork
  – Plural (NNS): dogs, forks
  – Proper (NNP, NNPS): John, Springfields
  – Personal pronoun (PRP): I, you, he, she, it
  – Wh-pronoun (WP): who, what

• Verb (actions and processes)
  – Base, infinitive (VB): eat
  – Past tense (VBD): ate
  – Gerund (VBG): eating
  – Past participle (VBN): eaten
  – Non 3rd person singular present tense (VBP): eat
  – 3rd person singular present tense: (VBZ): eats
  – Modal (MD): should, can
  – To (TO): to (to eat)
English Parts of Speech (cont.)

- Adjective (modify nouns)
  - Basic (JJ): red, tall
  - Comparative (JJR): redder, taller
  - Superlative (JJS): reddest, tallest

- Adverb (modify verbs)
  - Basic (RB): quickly
  - Comparative (RBR): quicker
  - Superlative (RBS): quickest

- Preposition (IN): on, in, by, to, with

- Determiner:
  - Basic (DT) a, an, the
  - WH-determiner (WDT): which, that

- Coordinating Conjunction (CC): and, but, or,

- Particle (RP): off (took off), up (put up)
Ambiguity in POS Tagging

• “Like” can be a verb or a preposition
  – I like/VBP candy.
  – Time flies like/IN an arrow.
• “Around” can be a preposition, particle, or adverb
  – I bought it at the shop around/IN the corner.
  – I never got around/RP to getting a car.
  – A new Prius costs around/RB $25K.
• Context from other words can help classify

Adapted from Ray Mooney
Aside: Why talking about HMMs

• A probabilistic graphical model
• Introduce sequence classification, nice way to model dynamical processes
• Introduce dealing with latent variables (not observed during training)
First Attempt: Markov Model

- No hidden variables
- Assume all POS are annotated by a human
- We can then reason about transitions between POS
- Goal: tag (classify) all words in a sentence with their POS
Sample Markov Model for POS
Sample Markov Model for POS

\[ P(\text{PropNoun Verb Det Noun}) = 0.4 \times 0.8 \times 0.25 \times 0.95 \times 0.1 = 0.0076 \]
Hidden Markov Models

- Probabilistic generative model for sequences.
- Assume an underlying set of hidden (unobserved) states in which the model can be (e.g. parts of speech, abbreviated POS).
- Assume probabilistic transitions between states over time (e.g. transition from POS to another POS as sequence is generated).
- Assume probabilistic generation of tokens from states (e.g. words generated per POS).
- Advantages of using hidden (un-annotated) variables?
Sample HMM for POS
Sample HMM Generation
Sample HMM Generation
Sample HMM Generation

```
<table>
<thead>
<tr>
<th>Sample HMM Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
</tr>
<tr>
<td>Tom 0.5</td>
</tr>
<tr>
<td>John 0.4</td>
</tr>
<tr>
<td>Mary</td>
</tr>
<tr>
<td>Jerry</td>
</tr>
<tr>
<td>PropNoun</td>
</tr>
<tr>
<td>start 0.1</td>
</tr>
<tr>
<td>a 0.95</td>
</tr>
<tr>
<td>a 0.95</td>
</tr>
<tr>
<td>the 0.95</td>
</tr>
<tr>
<td>the 0.95</td>
</tr>
<tr>
<td>that 0.95</td>
</tr>
<tr>
<td>Det</td>
</tr>
<tr>
<td>cat 0.05</td>
</tr>
<tr>
<td>dog 0.05</td>
</tr>
<tr>
<td>car 0.95</td>
</tr>
<tr>
<td>bed 0.85</td>
</tr>
<tr>
<td>apple 0.8</td>
</tr>
<tr>
<td>Noun</td>
</tr>
<tr>
<td>bit 0.1</td>
</tr>
<tr>
<td>ate 0.25</td>
</tr>
<tr>
<td>saw 0.25</td>
</tr>
<tr>
<td>played 0.1</td>
</tr>
<tr>
<td>hit 0.1</td>
</tr>
<tr>
<td>gave 0.1</td>
</tr>
<tr>
<td>Verb</td>
</tr>
<tr>
<td>stop 0.5</td>
</tr>
</tbody>
</table>
```

Ray Mooney
Sample HMM Generation

The diagram illustrates a hidden Markov model (HMM) for generating sentences. The model includes states for determiners (Det), propositions (PropNoun), nouns (Noun), and verbs (Verb). Transition probabilities between states are indicated by arrows, with numbers representing the probability of moving from one state to another. For example, the probability of transitioning from the determiner state to the noun state is 0.95. Similarly, the probability of transitioning from the noun state to the verb state is 0.85. The model also includes a start state and a stop state to define the beginning and end of a sentence.
Sample HMM Generation

John bit

Ray Mooney
Sample HMM Generation
Sample HMM Generation

John bit the
Sample HMM Generation

John bit the cat.
John bit the apple

Sample HMM Generation

The diagram represents a Hidden Markov Model (HMM) for generating sentences. The model includes states for determiners (Det), nouns (Noun), and verbs (Verb). The transitions between these states are labeled with probabilities. For example, the transition from "the" to "a" has a probability of 0.95, and the transition from "a" to "the" has a probability of 0.05. The sentence "John bit the apple" is generated through these transitions.
Sample HMM Generation

John bit the apple
Formal Definition of an HMM

- A set of $N + 2$ states $S = \{s_0, s_1, s_2, \ldots s_N, s_F\}$
  - Distinguished start state: $s_0$
  - Distinguished final state: $s_F$
- A set of $M$ possible observations $V = \{v_1, v_2 \ldots v_M\}$
- A state transition probability distribution $A = \{a_{ij}\}$
  \[ a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i) \quad 1 \leq i, j \leq N \text{ and } i = 0, j = F \]
  \[ \sum_{j=1}^{N} a_{ij} + a_{iF} = 1 \quad 0 \leq i \leq N \]
- Observation probability distribution for each state $j$
  $B = \{b_j(k)\}$
  \[ b_j(k) = P(v_k \text{ at } t \mid q_t = s_j) \quad 1 \leq j \leq N \quad 1 \leq k \leq M \]
- Total parameter set $\lambda = \{A, B\}$
Example

- States = weather (hot/cold)
- Observations = number of ice-creams eaten

Figure 9.3 A hidden Markov model for relating numbers of ice creams eaten by Jason (the observations) to the weather (H or C, the hidden variables).
Three Useful HMM Tasks

- **Compute observation likelihood**: How likely is a given sequence of words, regardless of how they might be POS-tagged?
- **Estimate most likely state sequence**: What is the most likely underlying sequence of tags for the observed sequence of words?
- **Maximum likelihood training**: Estimate transition/emission probabilities given training data (not discussed in this class)

Adapted from Ray Mooney
HMM: Observation Likelihood

- Given a sequence of observations, $O$, and a model with a set of parameters, $\lambda$, what is the probability that this observation was generated by this model: $P(O|\lambda)$?
- Allows HMM to be used as a language model: Assigns a probability to each string saying how likely that string is to be generated by the language.
- Example uses:
  - Sequence Classification
  - Most Likely Sequence

Adapted from Ray Mooney
Sequence Classification

• Assume an HMM is available for each category (i.e. language or word).
• What is the most likely category for a given observation sequence, i.e. which category’s HMM is most likely to have generated it?
• Used in speech recognition to find most likely word model to have generated a given sound or phoneme sequence.

\[ P(O \mid \text{Austin}) > P(O \mid \text{Boston})? \]
Most Likely Sequence

• Of two or more possible sequences, which one was most likely generated by a given model?

• Used to score alternative word sequence interpretations in speech recognition.

\[
P(O_2 \mid \text{OrdEnglish}) > P(O_1 \mid \text{OrdEnglish})?\]
HMM: Observation Likelihood
Naïve Solution

• Consider all possible state sequences, $Q$, of length $T$ that the model could have traversed in generating the given observation sequence.

• Compute
  – the probability of a given state sequence from $A$, and
  – multiply it by the probabilities (from $B$) of generating each of the given observations in each of the corresponding states in this sequence,
  – to get $P(O,Q|\lambda) = P(O|Q,\lambda)P(Q|\lambda)$.

• Sum this over all possible state sequences to get $P(O|\lambda)$.

• Computationally complex: $O(TN^T)$. 

Adapted from Ray Mooney
Example

- States = weather (hot/cold), observations = number of ice-creams eaten
- What is the probability of observing \{3, 1, 3\}?
Example

• What is the probability of observing \(\{3, 1, 3\}\) and the state sequence being \(\{\text{hot, hot, cold}\}\)?

\[
P(O, Q) = P(O|Q) \times P(Q) = \prod_{i=1}^{T} P(o_i|q_i) \times \prod_{i=1}^{T} P(q_i|q_{i-1})
\]

\[
P(3 \ 1 \ 3, \text{hot hot cold}) = P(\text{hot}|\text{start}) \times P(\text{hot}|\text{hot}) \times P(\text{cold}|\text{hot})
\times P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold})
\]

• What is the probability of observing \(\{3, 1, 3\}\)?

\[
P(O) = \sum_Q P(O, Q) = \sum_Q P(O|Q)P(Q)
\]

\[
P(3 \ 1 \ 3) = P(3 \ 1 \ 3, \text{cold cold cold}) + P(3 \ 1 \ 3, \text{cold cold hot}) + \ldots
\]
HMM: Observation Likelihood
Efficient Solution

• Due to the Markov assumption, the probability of being in any state at any given time $t$ only relies on the probability of being in each of the possible states at time $t-1$.

• **Forward Algorithm**: Uses dynamic programming to exploit this fact to efficiently compute observation likelihood in $O(TN^2)$ time.
  
  – Compute a *forward trellis* that compactly and implicitly encodes information about all possible state paths.
Forward Probabilities

- Let $\alpha_t(j)$ be the probability of being in state $j$ after seeing the first $t$ observations (by summing over all initial paths leading to $j$).

$$\alpha_t(j) = P(o_1, o_2, \ldots o_t, q_t = s_j \mid \lambda)$$
Forward Step

- Consider all possible ways of getting to \( s_j \) at time \( t \) by coming from all possible states \( s_i \) and determine probability of each.
- Sum these to get the total probability of being in state \( s_j \) at time \( t \) while accounting for the first \( t - 1 \) observations.
- Then multiply by the probability of actually observing \( o_t \) in \( s_j \).
• Continue forward in time until reaching final time point, and sum probability of ending in final state.
Computing the Forward Probabilities

• Initialization

\[ \alpha_1(j) = a_{0j} b_j(o_1) \quad 1 \leq j \leq N \]

• Recursion

\[
\alpha_t(j) = \left[ \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t) \quad 1 \leq j \leq N, \quad 1 < t \leq T
\]

• Termination

\[ P(O \mid \lambda) = \alpha_{T+1}(s_F) = \sum_{i=1}^{N} \alpha_T(i)a_{iF} \]
Example  \( \alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t) \)
Forward Computational Complexity

• Requires only $O(TN^2)$ time to compute the probability of an observed sequence given a model.

• Exploits the fact that all state sequences must merge into one of the $N$ possible states at any point in time and the Markov assumption that only the last state effects the next one.
Three Useful HMM Tasks

- **Compute observation likelihood**: How likely is a given sequence of words, regardless of how they might be POS-tagged?

- **Estimate most likely state sequence**: What is the most likely underlying sequence of tags for the observed sequence of words?

- **Maximum likelihood training**: Estimate transition/emission probabilities given training data (not discussed in this class)
Most Likely State Sequence (Decoding)

• Given an observation sequence, $O$, and a model, $\lambda$, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?

John gave the dog an apple.
Most Likely State Sequence

• Given an observation sequence, \( O \), and a model, \( \lambda \), what is the most likely state sequence, \( Q=q_1,q_2,…q_T \), that generated this sequence from this model?
Most Likely State Sequence

- Given an observation sequence, $O$, and a model, $\lambda$, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?

John gave the dog an apple.
Most Likely State Sequence

Given an observation sequence, $O$, and a model, $\lambda$, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?

John gave the dog an apple.

Ray Mooney
Most Likely State Sequence

• Given an observation sequence, $O$, and a model, $\lambda$, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?

John gave the dog an apple.

Det Noun PropNoun Verb
Most Likely State Sequence

• Given an observation sequence, $O$, and a model, $\lambda$, what is the most likely state sequence, $Q=q_1,q_2,\ldots q_T$, that generated this sequence from this model?

John gave the dog an apple.

Det Noun PropNoun Verb
Most Likely State Sequence

• Given an observation sequence, \( O \), and a model, \( \lambda \), what is the most likely state sequence, \( Q=q_1,q_2,\ldots q_T \), that generated this sequence from this model?

• Used for sequence labeling, assuming each state corresponds to a tag, it determines the globally best assignment of tags to all tokens in a sequence using a principled approach grounded in probability theory.
HMM: Most Likely State Sequence Efficient Solution

• Could use naïve algorithm, examining every possible state sequence of length $T$.
• Dynamic Programming can also be used to exploit the Markov assumption and efficiently determine the most likely state sequence for a given observation and model.
• Standard procedure is called the Viterbi algorithm (Viterbi, 1967) and also has $O(TN^2)$ time complexity.
Viterbi Scores

• Recursively compute the probability of the *most likely* subsequence of states that accounts for the first $t$ observations and ends in state $s_j$.

$$v_t(j) = \max_{q_0, q_1, \ldots, q_{t-1}} P(q_0, q_1, \ldots, q_{t-1}, o_1, \ldots, o_t, q_t = s_j \mid \lambda)$$

• Also record “backpointers” that subsequently allow backtracing the most probable state sequence.
  • $b_{t}(j)$ stores the state at time $t-1$ that maximizes the probability that system was in state $s_j$ at time $t$ (given the observed sequence).
Computing the Viterbi Scores

- **Initialization**
  \[ v_1(j) = a_{0j} b_j(o_1) \quad 1 \leq j \leq N \]

- **Recursion**
  \[ v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \leq j \leq N, \quad 1 < t \leq T \]

- **Termination**
  \[ P^* = v_{T+1}(s_F) = \max_{i=1}^{N} v_T(i) a_{iF} \]

Analogous to Forward algorithm except take \( \text{max} \) instead of sum.
Computing the Viterbi Backpointers

- **Initialization**
  \[ bt_1(j) = s_0 \quad 1 \leq j \leq N \]

- **Recursion**
  \[ bt_t(j) = \arg\max_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \leq j \leq N, \quad 1 \leq t \leq T \]

- **Termination**
  \[ q_T^* = bt_{T+1}(s_F) = \arg\max_{i=1}^N v_T(i) a_{iF} \]

Final state in the most probable state sequence. Follow backpointers to initial state to construct full sequence.
Viterbi Backpointers
Viterbi Backtrace

Most likely Sequence: $s_0 s_N s_1 s_2 \ldots s_2 s_F$
HMM Learning

- **Supervised Learning**: All training sequences are completely labeled (tagged).
- **Unsupervised Learning**: All training sequences are unlabelled (but generally know the number of tags, i.e. states).

Adapted from Ray Mooney
Supervised Parameter Estimation

- Estimate state transition probabilities based on tag bigram and unigram statistics in the labeled data.

\[ a_{ij} = \frac{C(q_t = s_i, q_{t+1} = s_j)}{C(q_t = s_i)} \]

- Estimate the observation probabilities based on tag/word co-occurrence statistics in the labeled data.

\[ b_{jk}(k) = \frac{C(q_i = s_j, o_i = v_k)}{C(q_i = s_j)} \]

- Use appropriate smoothing if training data is sparse.
Maximum Likelihood Training

• Given an observation sequence, \( O \), what set of parameters, \( \lambda \), for a given model maximizes the probability that this data was generated from this model (\( P(O|\lambda) \))?

• Used to train an HMM model and properly induce its parameters from a set of training data.

• Only need to have an unannotated observation sequence (or set of sequences) generated from the model. Does not need to know the correct state sequence(s) for the observation sequence(s). In this sense, it is unsupervised.
HMM: Maximum Likelihood Training
Efficient Solution

- There is no known efficient algorithm for finding the parameters, $\lambda$, that truly maximizes $P(O|\lambda)$.
- However, using iterative re-estimation, the Baum-Welch algorithm (a.k.a. forward-backward), a version of a standard statistical procedure called Expectation Maximization (EM), is able to locally maximize $P(O|\lambda)$.
- Recall: K-means (also an example of EM)
Sketch of Baum-Welch (EM) Algorithm for Training HMMs

Assume an HMM with $N$ states. Randomly set its parameters $\lambda=(A,B)$ (making sure they represent legal distributions).

Until convergence (i.e. $\lambda$ no longer changes) do:

E Step: Use the forward/backward procedure to determine the probability of various possible state sequences for generating the training data

M Step: Use these probability estimates to re-estimate values for all of the parameters $\lambda$