Lecture 5:
Sequence Prediction & HMMs
Announcements

• Homework 1 Due Today @5pm
  – Via Moodle

• Homework 2 to be Released Soon.
  – Due Feb 3rd via Moodle (2 weeks)
  – Mostly Coding
    • Don’t start late!

• Use Moodle Forums for Q&A
Sequence Prediction
(POS Tagging)

• $x = “Fish Sleep”$
• $y = (N, V)$

• $x = “The Dog Ate My Homework”$
• $y = (D, N, V, D, N)$

• $x = “The Fox Jumped Over The Fence”$
• $y = (D, N, V, P, D, N)$
Challenges

• Multivariable Output
  – Makes multiple predictions simultaneously

• Variable Length Input/Output
  – Sentence lengths not fixed
Multivariate Outputs

• \( x = \text{“Fish Sleep”} \)
• \( y = (N, V) \)

• Multiclass prediction:

Replicate Weights:
\[
\begin{bmatrix}
  w_1 \\
  w_2 \\
  \vdots \\
  w_K
\end{bmatrix}
\]

Score All Classes:
\[
f(x \mid w, b) = \begin{bmatrix}
  w_1^T x - b_1 \\
  w_2^T x - b_2 \\
  \vdots \\
  w_K^T x - b_K
\end{bmatrix}
\]

Predict via Largest Score:
\[
\arg\max_k \begin{bmatrix}
  w_1^T x - b_1 \\
  w_2^T x - b_2 \\
  \vdots \\
  w_K^T x - b_K
\end{bmatrix}
\]

• How many classes?

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep
Multiclass Prediction

• $x = \text{“Fish Sleep”}$
• $y = (N, V)$

• Multiclass prediction:
  – All possible length-$M$ sequences as different class
    – $(D, D), (D, N), (D, V), (D, Adj), (D, Adv), (D, Pr)$
    – $(N, D), (N, N), (N, V), (N, Adj), (N, Adv), \ldots$

• $L^M$ classes!
  – Length 2: $6^2 = 36!$
Multiclass Prediction

• $x = \text{“Fish Sleep”}$
• $y = (N, V)$

Multiclass prediction:
– All possible length-$M$ sequences as different classes
  - $(D, D)$, $(D, N)$, $(D, V)$, $(D,$Adj$)$, $(D,$Adv$)$, $(D,$Pr$)$
  - $(N, D)$, $(N, N)$, $(N, V)$, $(N,$Adj$)$, $(N,$Adv$)$, ...

– $L^M$ classes!

Exponential Explosion in #Classes!
(Not Tractable for Sequence Prediction)

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

$L=6$

– Length 2: $6^2 = 36!$
Why is Naïve Multiclass Intractable?

x=“I fish often”

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

- (D, D, D), (D, D, N), (D, D, V), (D, D, Adj), (D, D, Adv), (D, D, Pr)
- (D, N, D), (D, N, N), (D, N, V), (D, N, Adj), (D, N, Adv), (D, N, Pr)
- (D, V, D), (D, V, N), (D, V, V), (D, V, Adj), (D, V, Adv), (D, V, Pr)
- ...
- (N, D, D), (N, D, N), (N, D, V), (N, D, Adj), (N, D, Adv), (N, D, Pr)
- (N, N, D), (N, N, N), (N, N, V), (N, N, Adj), (N, N, Adv), (N, N, Pr)
- ...

Assume pronouns are nouns for simplicity.
Why is Naïve Multiclass Intractable?

\[ x = "I \text{ fish often}" \]

- \((D, D, D), (D, D, N), (D, D, V), (D, D, \text{Adj}), (D, D, \text{Adv}), (D, D, \text{Pr})\)

Treats Every Combination As Different Class
(Learn \((w, b)\) for each combination)

Exponentially Large Representation!
(Exponential Time to Consider Every Class)
(Exponential Storage)

Assume pronouns are nouns for simplicity.
Independent Classification

x="I fish often"

POS Tags: 
Det, Noun, Verb, Adj, Adv, Prep

• Treat each word independently (assumption)
  – Independent multiclass prediction per word
  – Predict for x="I" independently
  – Predict for x="fish" independently
  – Predict for x="often" independently
  – Concatenate predictions.

Assume pronouns are nouns for simplicity.
Independent Classification

x=“I fish often”

• Treat each word independently (assumption)
  – Independent multiclass prediction per word

POS Tags:
Det, Noun, Verb, Adj, Adv, Prep

#Classes = #POS Tags
(6 in our example)

Solvable using standard multiclass prediction.

Assume pronouns are nouns for simplicity.
Independent Classification

x=“I fish often”

• Treat each word independently
  – Independent multiclass prediction per word

| P(y|x) | x=“I”  | x=“fish” | x=“often” |
|--------|--------|----------|-----------|
| y=“Det” | 0.0    | 0.0      | 0.0       |
| y=“Noun” | 1.0    | 0.75     | 0.0       |
| y=“Verb” | 0.0    | 0.25     | 0.0       |
| y=“Adj”  | 0.0    | 0.0      | 0.4       |
| y=“Adv”  | 0.0    | 0.0      | 0.6       |
| y=“Prep” | 0.0    | 0.0      | 0.0       |

Prediction: (N, N, Adv)
Correct: (N, V, Adv)

Why the mistake?

Assume pronouns are nouns for simplicity.
Context Between Words

\[ x = \text{“I fish often”} \]

• Independent Predictions Ignore Word Pairs
  – In Isolation:
    • “Fish” is more likely to be a Noun
  – But Conditioned on Following a (pro)Noun...
    • “Fish” is more likely to be a Verb!

– “1st Order” Dependence (Model All Pairs)
  • 2\textsuperscript{nd} Order Considers All Triplets
  • Arbitrary Order = Exponential Size (Naïve Multiclass)

Assume pronouns are nouns for simplicity.
1st Order Hidden Markov Model

- $x = (x^1, x^2, x^4, x^4, \ldots, x^M)$ (sequence of words)
- $y = (y^1, y^2, y^3, y^4, \ldots, y^M)$ (sequence of POS tags)

- $P(x^i | y^i)$ Probability of state $y^i$ generating $x^i$
- $P(y^{i+1} | y^i)$ Probability of state $y^i$ transitioning to $y^{i+1}$
- $P(y^1 | y^0)$ $y^0$ is defined to be the Start state
- $P(\text{End} | y^M)$ Prior probability of $y^M$ being the final state
  - Not always used
1st Order Hidden Markov Model

Models All State-State Pairs (all POS Tag-Tag pairs)
Models All State-Observation Pairs (all Tag-Word pairs)

Additional Complexity of (#POS Tags)^2

Same Complexity as Independent Multiclass

- $P(x^i | y^i)$ Probability of state $y^i$ generating $x^i$
- $P(y^{i+1} | y^i)$ Probability of state $y^i$ transitioning to $y^{i+1}$
- $P(y^1 | y^0)$ $y^0$ is defined to be the Start state
- $P(\text{End} | y^M)$ Prior probability of $y^M$ being the final state
  - Not always used
1st Order Hidden Markov Model

\[ P(x, y) = P(\text{End} | y^M) \prod_{i=1}^{M} P(y^i | y^{i-1}) \prod_{i=1}^{M} P(x^i | y^i) \]

“Joint Distribution”

- \( P(x^i | y^i) \) Probability of state \( y^i \) generating \( x^i \)
- \( P(y^{i+1} | y^i) \) Probability of state \( y^i \) transitioning to \( y^{i+1} \)
- \( P(y^1 | y^0) \) \( y^0 \) is defined to be the Start state
- \( P(\text{End} | y^M) \) Prior probability of \( y^M \) being the final state
  - Not always used

Optional
1st Order Hidden Markov Model

\[ P(x \mid y) = \prod_{i=1}^{M} P(x^i \mid y^i) \]

Given a POS Tag Sequence \( y \):

**Can compute each \( P(x^i \mid y) \) independently!**

(\( x^i \) conditionally independent given \( y^i \))

“Conditional Distribution on \( x \) given \( y \)”

- \( P(x^i \mid y^i) \) Probability of state \( y^i \) generating \( x^i \)
- \( P(y^{i+1} \mid y^i) \) Probability of state \( y^i \) transitioning to \( y^{i+1} \)
- \( P(y^1 \mid y^0) \) \( y^0 \) is defined to be the Start state
- \( P(\text{End} \mid y^M) \) Prior probability of \( y^M \) being the final state
  - Not always used
P ( word | state/tag )

- Two-word language: “fish” and “sleep”
- Two-tag language: “Noun” and “Verb”

| P(x|y)   | y=“Noun” | y=“Verb” |
|---------|----------|----------|
| x=“fish”| 0.8      | 0.5      |
| x=“sleep”| 0.2     | 0.5     |

**Given Tag Sequence y:**

- P(“fish sleep” | (N,V) ) = 0.8*0.5
- P(“fish fish”  | (N,V) ) = 0.8*0.5
- P(“sleep fish” | (V,V) ) = 0.8*0.5
- P(“sleep sleep” | (N,N) ) = 0.2*0.5
**Sampling**

- HMMs are “generative” models
  - Models joint distribution $P(x,y)$
  - Can generate samples from this distribution
  - First consider conditional distribution $P(x|y)$

| $P(x|y)$     | y=“Noun” | y=“Verb” |
|-------------|----------|----------|
| x=“fish”    | 0.8      | 0.5      |
| x=“sleep”   | 0.2      | 0.5      |

**Given Tag Sequence $y = (N,V)$:**

**Sample each word independently:**
Sample $P(x^1|N)$ (0.8 Fish, 0.2 Sleep)
Sample $P(x^2|V)$ (0.5 Fish, 0.5 Sleep)

- What about sampling from $P(x,y)$?
Forward Sampling of $P(y,x)$

$$P(x, y) = P(\text{End} \mid y^M) \prod_{i=1}^{M} P(y^i \mid y^{i-1}) \prod_{i=1}^{M} P(x^i \mid y^i)$$

<table>
<thead>
<tr>
<th>P($x \mid y$)</th>
<th>y=&quot;Noun&quot;</th>
<th>y=&quot;Verb&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=&quot;fish&quot;</td>
<td>0.8</td>
<td>0.5</td>
</tr>
<tr>
<td>x=&quot;sleep&quot;</td>
<td>0.2</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Initialize $y^0 = \text{Start}$
Initialize $i = 0$

1. $i = i + 1$
2. Sample $y^i$ from $P(y^i \mid y^{i-1})$
3. If $y^i == \text{End}$: Quit
4. Sample $x^i$ from $P(x^i \mid y^i)$
5. Goto Step 1

Exploits Conditional Ind.
Requires $P(\text{End} \mid y^i)$
Forward Sampling of $P(y,x|L)$

$$P(x, y | M) = P(\text{End} | y^M) \prod_{i=1}^{M} P(y^i | y^{i-1}) \prod_{i=1}^{M} P(x^i | y^i)$$

**Experiments**

| $P(x | y)$ | y="Noun" | y="Verb" |
|-----------|-----------|-----------|
| x="fish"  | 0.8       | 0.5       |
| x="sleep" | 0.2       | 0.5       |

**Graphical Model**

- Initialize $y^0 = \text{Start}$
- Initialize $i = 0$

1. $i = i + 1$
2. If($i == M$): Quit
3. Sample $y^i$ from $P(y^i | y^{i-1})$
4. Sample $x^i$ from $P(x^i | y^i)$
5. Goto Step 1

Exploits Conditional Ind.
Assumes no $P(\text{End} | y^i)$

Slides borrowed from Ralph Grishman
1st Order Hidden Markov Model

\[ P(x_{k+1:M}, y_{k+1:M} \mid x_{1:k}, y_{1:k}) = P(x_{k+1:M}, y_{k+1:M} \mid y^k) \]

“Memory-less Model” – only needs \( y^k \) to model rest of sequence

- \( P(x^i \mid y^i) \) Probability of state \( y^i \) generating \( x^i \)
- \( P(y^{i+1} \mid y^i) \) Probability of state \( y^i \) transitioning to \( y^{i+1} \)
- \( P(y^1 \mid y^0) \) \( y^0 \) is defined to be the Start state
- \( P(\text{End} \mid y^M) \) Prior probability of \( y^M \) being the final state
  - Not always used
Viterbi Algorithm
Most Common Prediction Problem

• Given input sentence, predict POS Tag seq.
  \[ \arg\max_y P(y \mid x) \]

• Naïve approach:
  – Try all possible \( y \)'s
  – Choose one with highest probability
  – **Exponential time:** \( L^M \) possible \( y \)'s
Bayes’s Rule

\[
\text{argmax } P(y \mid x) = \text{argmax } y \frac{P(y, x)}{P(x)} \\
= \text{argmax } P(y, x) \\
= \text{argmax } y P(x \mid y) P(y)
\]

\[
P(x \mid y) = \prod_{i=1}^{L} P(x^i \mid y^i)
\]

\[
P(y) = P(End \mid y^L) \prod_{i=1}^{L} P(y^i \mid y^{i-1})
\]
$$ \arg\max P(y, x) = \arg\max_y \prod_{i=1}^{M} P(y^i | y^{i-1}) \prod_{i=1}^{M} P(x^i | y^i) $$

$$ = \arg\max_{y^M} \arg\max_{y^{1:M-1}} \prod_{i=1}^{M} P(y^i | y^{i-1}) \prod_{i=1}^{M} P(x^i | y^i) $$

$$ = \arg\max_{y^M} \arg\max_{y^{1:M-1}} P(y^M | y^{M-1}) P(x^M | y^M) P(y^{1:M-1} | x^{1:M-1}) $$

$$ P\left(y^{1:k} \mid x^{1:k}\right) = P(x^{1:k} \mid y^{1:k}) P(y^{1:k}) \quad P\left(x^{1:k} \mid y^{1:k}\right) = \prod_{i=1}^{k} P(x^i \mid y^i) $$

$$ P\left(y^{1:k}\right) = \prod_{i=1}^{k} P(y^{i+1} \mid y^i) $$

**Exploit Memory-less Property:**

The choice of $y^1$ only depends on $y^{1:M-1}$ via $P(y^M \mid y^{M-1})$!
Dynamic Programming

- **Input:** \( x = (x^1, x^2, x^3, \ldots, x^M) \)

- **Computed:** best length-\( k \) prefix ending in each Tag:
  - Examples:
    
    $\hat{Y}^k(V) = \left( \arg\max_{y^{l:k-1}} P(y^{l:k-1} \oplus V, x^{l:k}) \right) \oplus V$
    
    $\hat{Y}^k(N) = \left( \arg\max_{y^{l:k-1}} P(y^{l:k-1} \oplus N, x^{l:k}) \right) \oplus N$

- **Claim:**
  
  $\hat{Y}^{k+1}(V) = \left( \arg\max_{y^{l:k} \in \{\hat{Y}^k(T)\}_T} P(y^{l:k}, x^{l:k+1}) \right) \oplus V$
  
  $= \left( \arg\max_{y^{l:k} \in \{\hat{Y}^k(T)\}_T} P(y^{l:k}, x^{l:k}) P(y^{k+1} = V | y^k) P(x^{k+1} | y^{k+1} = V) \right) \oplus V$
Solve: \[ \hat{Y}^2(V) = \left( \arg\max_{y^1 \in \hat{Y}(T)} P(y^1, x^1) P(y^2 = V | y^1) P(x^2 | y^2 = V) \right) \oplus V \]

Store each \( \hat{Y}^1(Z) \) & \( P(\hat{Y}^1(Z), x^1) \)

\( \hat{Y}^1(Z) \) is just \( Z \)
Solve:

$$\hat{Y}^2(V) = \left\{ \arg\max_{y^1 \in \hat{Y}^1(T)} P(y^1, x^1) P(y^2 = V \mid y^1) P(x^2 \mid y^2 = V) \right\} \oplus V$$

Store each
\(\hat{Y}^1(Z) \& P(\hat{Y}^1(Z), x^1)\)

\(\hat{Y}^1(Z)\) is just \(Z\)

Ex: \(\hat{Y}^2(V) = (N, V)\)
Solve: $\hat{Y}^3(V) = \left( \arg\max \ P(y^{1:2}, x^{1:2}) P(y^3 = V \mid y^2) P(x^3 \mid y^3 = V) \right) \oplus V$

Store each $\hat{Y}^1(Z) \& P(\hat{Y}^1(Z), x^1)$

Store each $\hat{Y}^2(Z) \& P(\hat{Y}^2(Z), x^{1:2})$

Ex: $\hat{Y}^2(V) = (N, V)$
\[ \hat{Y}^3(V) = \left( \arg\max_{y^{1:2} \in \{\hat{Y}^2(T)\}_T} P(y^{1:2}, x^{1:2})P(y^3 = V \mid y^2)P(x^3 \mid y^3 = V) \right) \oplus V \]

Store each \(\hat{Y}^1(Z)\) & \(P(\hat{Y}^1(Z), x^1)\)

Store each \(\hat{Y}^2(Z)\) & \(P(\hat{Y}^2(Z), x^{1:2})\)

Claim: Only need to check solutions of \(\hat{Y}^2(Z)\), \(Z=V,D,N\)

Ex: \(\hat{Y}^2(V) = (N, V)\)
Solve:  \[ \hat{Y}^3(V) = \left( \arg\max_{y^{1:2} \in \{\hat{Y}^2(T)\}_T} P(y^{1:2}, x^{1:2}) P(y^3 = V | y^2) P(x^3 | y^3 = V) \right) \oplus V \]

Store each \( \hat{Y}^1(Z) \) & \( P(\hat{Y}^1(Z), x^1) \)
Store each \( \hat{Y}^2(Z) \) & \( P(\hat{Y}^2(Z), x^{1:2}) \)

Claim: Only need to check solutions of \( \hat{Y}^2(Z) \), \( Z=V,D,N \)

Suppose \( \hat{Y}^3(V) = (V,V,V) \)...
...prove that \( \hat{Y}^3(V) = (N,V,V) \) has higher prob.

Proof depends on 1\textsuperscript{st} order property
\begin{itemize}
  \item Prob. of \( (V,V,V) \) & \( (N,V,V) \) differ in 3 terms
  \item \( P(y^1 | y^0) \), \( P(x^1 | y^1) \), \( P(y^2 | y^1) \)
  \item None of these depend on \( y^3 \)!
\end{itemize}

Ex: \( \hat{Y}^3(V) = (N,\cdot,\cdot) \)
\[ \hat{Y}^M(V) = \left( \arg\max_{Y^{1:M-1} \in \hat{Y}^{M-1}(r)} P(Y^{1:M-1}, x^{1:M-1}) P(y^M = V | y^{M-1}) P(x^M | y^M = V) P(\text{End} | y^M = V) \right) \oplus V \]

Store each \( \hat{Y}^1(Z) & P(\hat{Y}^1(Z), x^1) \)

Store each \( \hat{Y}^2(Z) & P(\hat{Y}^2(Z), x^{1:2}) \)

Store each \( \hat{Y}^3(Z) & P(\hat{Y}^3(Z), x^{1:3}) \)

Ex: \( \hat{Y}^2(V) = (N, V) \)

Ex: \( \hat{Y}^3(V) = (D, N, V) \)
Viterbi Algorithm

• Solve: \[ \arg\max_y P(y \mid x) = \arg\max_y \frac{P(y, x)}{P(x)} = \arg\max_y P(y, x) = \arg\max_y P(x \mid y)P(y) \]

• For k=1..M
  – Iteratively solve for each \( \hat{Y}^k(Z) \)
    • Z looping over every POS tag.

• Predict best \( \hat{Y}^M(Z) \)
  • Also known as Mean A Posteriori (MAP) inference
Numerical Example

\[ x = (\text{Fish Sleep}) \]
A Simple POS HMM

| P(x|y) | y="Noun" | y="Verb" |
|-------|----------|----------|
| x="fish" | 0.8      | 0.5      |
| x="sleep" | 0.2      | 0.5      |

Slides borrowed from Ralph Grishman
A Simple POS HMM

Token 1: fish

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.2 * .5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.8 * .8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

P(x|y) | y=“Noun” | y=“Verb”
--- | --- | ---

x=“fish” | 0.8 | 0.5
x=“sleep” | 0.2 | 0.5

Slides borrowed from Ralph Grishman
Token 1: fish

<table>
<thead>
<tr>
<th>State</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

P(x|y) | y=“Noun” | y=“Verb” |
--- | --- | --- |
x=“fish” | 0.8 | 0.5 |
x=“sleep” | 0.2 | 0.5 |

Slides borrowed from Ralph Grishman
Token 2: sleep
(if ‘fish’ is verb)

P(x|y) | y="Noun" | y="Verb"
---|---|---
x="fish" | 0.8 | 0.5
x="sleep" | 0.2 | 0.5

Slides borrowed from Ralph Grishman
Token 2: sleep
(if ‘fish’ is verb)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td>.005</td>
<td></td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td>.004</td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

P(x|y) | y=“Noun” | y=“Verb”
---|---|---
x=“fish” | 0.8 | 0.5 |
x=“sleep” | 0.2 | 0.5 |

Slides borrowed from Ralph Grishman
Token 2: sleep
(if ‘fish’ is a noun)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td>.005</td>
<td>.64*.8*.5</td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td>.004</td>
<td>.64*.1*.2</td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

$P(x|y)$

<table>
<thead>
<tr>
<th>y=&quot;Noun&quot;</th>
<th>y=&quot;Verb&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=&quot;fish&quot;</td>
<td>0.8</td>
</tr>
<tr>
<td>x=&quot;sleep&quot;</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Slides borrowed from Ralph Grishman
Token 2: sleep
(if ‘fish’ is a noun)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td>.005</td>
<td>.256</td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td>.004</td>
<td>.0128</td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

P(x|y) y=“Noun” y=“Verb”

- x=“fish” 0.8 0.5
- x=“sleep” 0.2 0.5

Slides borrowed from Ralph Grishman
Token 2: sleep
take maximum, set back pointers

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td>.005</td>
<td>.256</td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

$P(x|y)$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$=&quot;Noun&quot;</th>
<th>$y$=&quot;Verb&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>fish</td>
<td>0.8</td>
<td>0.5</td>
</tr>
<tr>
<td>sleep</td>
<td>0.2</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Slides borrowed from Ralph Grishman
Token 2: sleep
take maximum,
set back pointers

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td>.256</td>
<td></td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td>.0128</td>
<td></td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>
Token 3: end

\[
P(x|y) \quad y=\text{“Noun”} \quad y=\text{“Verb”}
\]
\[
| x=\text{“fish”} | 0.8 | 0.5 |
| x=\text{“sleep”} | 0.2 | 0.5 |
\]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
\text{start} & 1 & 0 & 0 & 0 \\
\text{verb} & 0 & .1 & .256 & - \\
\text{noun} & 0 & .64 & .0128 & - \\
\text{end} & 0 & 0 & - & .256*.7 \\
& & & & .0128*.1
\end{array}
\]

Slides borrowed from Ralph Grishman
Token 3: end
take maximum,
set back pointers

<table>
<thead>
<tr>
<th>State</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>verb</td>
<td>0</td>
<td>.1</td>
<td>.256</td>
<td>-</td>
</tr>
<tr>
<td>noun</td>
<td>0</td>
<td>.64</td>
<td>.0128</td>
<td>-</td>
</tr>
<tr>
<td>end</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>.256*.7</td>
</tr>
</tbody>
</table>

\[
P(x|y) \quad \begin{array}{c|cc}
\text{y=“Noun”} & \text{y=“Verb”} \\
\hline
x=“fish” & 0.8 & 0.5 \\
x=“sleep” & 0.2 & 0.5 \\
\end{array}
\]

Slides borrowed from Ralph Grishman
**Decode:**

- *fish* = noun
- *sleep* = verb

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>start</strong></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>verb</strong></td>
<td>0</td>
<td>.1</td>
<td>.256</td>
<td>-</td>
</tr>
<tr>
<td><strong>noun</strong></td>
<td>0</td>
<td>.64</td>
<td>.0128</td>
<td>-</td>
</tr>
<tr>
<td><strong>end</strong></td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>.256* .7</td>
</tr>
</tbody>
</table>

**P(x|y)**

<table>
<thead>
<tr>
<th></th>
<th>y=&quot;Noun&quot;</th>
<th>y=&quot;Verb&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=&quot;fish&quot;</td>
<td>0.8</td>
<td>0.5</td>
</tr>
<tr>
<td>x=&quot;sleep&quot;</td>
<td>0.2</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Slides borrowed from Ralph Grishman
Recap: Viterbi

• Predict most likely $y$ given $x$:
  – E.g., predict POS Tags given sentences

\[
\text{argmax } P( y \mid x ) = \text{argmax } \frac{ P( y, x ) }{ P( x ) } \\
= \text{argmax } P( y, x ) \\
= \text{argmax } P( x \mid y ) P( y )
\]

• Solve using Dynamic Programming
Recap: Independent Classification

x="I fish often"

- Treat each word independently
  - Independent multiclass prediction per word

| P(y|x)  | x="I" | x="fish" | x="often" |
|---------|-------|----------|-----------|
| y="Det"| 0.0   | 0.0      | 0.0       |
| y="Noun"| 1.0   | 0.75     | 0.0       |
| y="Verb"| 0.0   | 0.25     | 0.0       |
| y="Adj"| 0.0   | 0.0      | 0.4       |
| y="Adv"| 0.0   | 0.0      | 0.6       |
| y="Prep"| 0.0   | 0.0      | 0.0       |

POS Tags: Det, Noun, Verb, Adj, Adv, Prep

Prediction: (N, N, Adv)

Correct: (N, V, Adv)

Mistake due to not modeling multiple words.

Assume pronouns are nouns for simplicity.
Recap: Viterbi

• Models pairwise transitions between states
  – Pairwise transitions between POS Tags
  – “1st order” model

\[
P(x, y) = P(\text{End} \mid y^M) \prod_{i=1}^{M} P(y^i \mid y^{i-1}) \prod_{i=1}^{M} P(x^i \mid y^i)
\]

\[x=\text{“I fish often”}\]

Independent: (N, N, Adv)

HMM Viterbi: (N, V, Adv)

*Assuming we defined \(P(x,y)\) properly
Training HMMs
Supervised Training

• **Given:**

\[ S = \left\{ (x_i, y_i) \right\}_{i=1}^{N} \]

Word Sequence (Sentence)

POS Tag Sequence

• **Goal:** Estimate \( P(x, y) \) using \( S \)

\[
P(x, y) = P(End | y^M) \prod_{i=1}^{M} P(y^i | y^{i-1}) \prod_{i=1}^{M} P(x^i | y^i)
\]

• **Maximum Likelihood!**
Aside: Matrix Formulation

• Define Transition Matrix: \( A \)
  
  \[
  A_{ab} = P(y^{i+1}=a \mid y^i=b) \text{ or } -\log(P(y^{i+1}=a \mid y^i=b))
  \]

<table>
<thead>
<tr>
<th>( y^{\text{next}} )</th>
<th>( y=\text{“Noun”} )</th>
<th>( y=\text{“Verb”} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y^{\text{next}}=\text{“Noun”} )</td>
<td>0.09</td>
<td>0.6667</td>
</tr>
<tr>
<td>( y^{\text{next}}=\text{“Verb”} )</td>
<td>0.91</td>
<td>0.333</td>
</tr>
</tbody>
</table>

• Observation Matrix: \( O \)
  
  \[
  O_{wz} = P(x^i=w \mid y^i=z) \text{ or } -\log(P(x^i=w \mid y^i=z))
  \]

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y=\text{“Noun”} )</th>
<th>( y=\text{“Verb”} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>fish</td>
<td>0.8</td>
<td>0.5</td>
</tr>
<tr>
<td>sleep</td>
<td>0.2</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Aside: Matrix Formulation

\[
P(x, y) = P(End \mid y^M) \prod_{i=1}^{M} P(y^i \mid y^{i-1}) \prod_{i=1}^{M} P(x^i \mid y^i)
\]

\[
P(x, y) = P(End \mid y^M) \prod_{i=1}^{M} P(y^i \mid y^{i-1}) \prod_{i=1}^{M} P(x^i \mid y^i)
\]

\[
= A_{End, y^M} \prod_{i=1}^{M} A_{y^i, y^{i-1}} \prod_{i=1}^{M} O_{x^i, y^i}
\]

\[
-\log(P(x, y)) = \tilde{A}_{End, y^M} + \sum_{i=1}^{M} \tilde{A}_{y^i, y^{i-1}} + \sum_{i=1}^{M} \tilde{O}_{x^i, y^i}
\]

Log prob. formulation
Each entry of \(\tilde{A}\) is define as \(-\log(A)\)
Maximum Likelihood

\[
\arg\max_{A,O} \prod_{(x,y)\in S} P(x,y) = \arg\max_{A,O} \prod_{(x,y)\in S} P(End \mid y^M) \prod_{i=1}^{M} P(y^{i} \mid y^{i-1}) \prod_{i=1}^{M} P(x^{i} \mid y^{i})
\]

- Estimate each component separately:

\[
A_{ab} = \frac{\sum_{j=1}^{N} \sum_{i=0}^{M_j} 1[(y_{j+1}=a) \land (y_j=b)]}{\sum_{j=1}^{N} \sum_{i=0}^{M_j} 1[y_j=b]}
\]

\[
O_{wz} = \frac{\sum_{j=1}^{N} \sum_{i=1}^{M_j} 1[(x_j=w) \land (y_j=z)]}{\sum_{j=1}^{N} \sum_{i=1}^{M_j} 1[y_j=z]}
\]

- Can also minimize neg. log likelihood
Recap: Supervised Training

\[
\arg\max_{A,O} \prod_{(x,y)\in S} P(x, y) = \arg\max_{A,O} \prod_{(x,y)\in S} P(\text{End} \mid y^M) \prod_{i=1}^{M} P(y^i \mid y^{i-1}) \prod_{i=1}^{M} P(x^i \mid y^i)
\]

• Maximum Likelihood Training
  – Counting statistics
  – Super easy!
  – Why?

• What about unsupervised case?
Recap: Supervised Training

\[
\arg\max_{A, O} \prod_{(x, y) \in S} P(x, y) = \arg\max_{A, O} \prod_{(x, y) \in S} P(End \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)
\]

• Maximum Likelihood Training
  – Counting statistics
  – Super easy!
  – Why?

• What about unsupervised case?
Conditional Independence Assumptions

\[
\arg\max_{A,O} \prod_{(x,y) \in S} P(x,y) = \arg\max_{A,O} \prod_{(x,y) \in S} P(\text{End} \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)
\]

- Everything decomposes to products of pairs
  - I.e., \( P(y^{i+1}=a \mid y^i=b) \) doesn’t depend on anything else

- Can just estimate frequencies:
  - How often \( y^{i+1}=a \) when \( y^i=b \) over training set
  - Note that \( P(y^{i+1}=a \mid y^i=b) \) is a common model across all locations of all sequences.
Conditional Independence Assumptions

\[
\arg\max_{A,O} \prod_{(x,y) \in S} P(x,y) = \arg\max_{A,O} \prod_{(x,y) \in S} P(\text{End} \mid y^M) \prod_{i=1}^M P(y^i \mid y^{i-1}) \prod_{i=1}^M P(x^i \mid y^i)
\]

# Parameters:
Transitions A: \#Tags^2
Observations O: \#Words \times \#Tags
Avoids directly model word/word pairings
\#Tags = 10s
\#Words = 10000s
Unsupervised Training

• What about if no y’s?
  – Just a training set of sentences

• Still want to estimate $P(x,y)$
  – How?
  – Why?

$S = \{x_i\}_{i=1}^N$
Why Unsupervised Training?

• Supervised Data hard to acquire
  – Require annotating POS tags
• Unsupervised Data plentiful
  – Just grab some text!
• Might just work for POS Tagging!
  – Learn y’s that correspond to POS Tags
• Can be used for other tasks
  – Detect outlier sentences (sentences with low prob.)
  – Sampling new sentences.
EM Algorithm (Baum-Welch)

- If we had y’s → max likelihood.
- If we had (A,O) → predict y’s

1. Initialize A and O arbitrarily
2. Predict prob. of y’s for each training x
3. Use y’s to estimate new (A,O)
4. Repeat back to Step 1 until convergence

http://en.wikipedia.org/wiki/Baum%E2%80%93Welch_algorithm
Expectation Step

• Given (A,O)

• For training x=(x₁,...,xₜ)
  – Predict P(yᵢ) for each y=(y₁,...yₜ)

<table>
<thead>
<tr>
<th></th>
<th>x₁</th>
<th>x₂</th>
<th>...</th>
<th>xₜ</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(yᵢ=Noun)</td>
<td>0.5</td>
<td>0.4</td>
<td>...</td>
<td>0.05</td>
</tr>
<tr>
<td>P(yᵢ=Det)</td>
<td>0.4</td>
<td>0.6</td>
<td>...</td>
<td>0.25</td>
</tr>
<tr>
<td>P(yᵢ=Verb)</td>
<td>0.1</td>
<td>0.0</td>
<td>...</td>
<td>0.7</td>
</tr>
</tbody>
</table>

– Encodes current model’s beliefs about y
– “Marginal Distribution” of each yᵢ
Recall: Matrix Formulation

• Define Transition Matrix: $A$
  
  $A_{ab} = P(y_{i+1} = a | y_i = b) \text{ or } -\log(P(y_{i+1} = a | y_i = b))$

| $P(y_{\text{next}} | y)$ | $y=\text{“Noun”}$ | $y=\text{“Verb”}$ |
|--------------------------|-----------------|-----------------|
| $y_{\text{next}}=\text{“Noun”}$ | 0.09            | 0.667           |
| $y_{\text{next}}=\text{“Verb”}$ | 0.91            | 0.333           |

• Observation Matrix: $O$

  $O_{wz} = P(x_i = w | y_i = z) \text{ or } -\log(P(x_i = w | y_i = z))$

| $P(x_i | y)$ | $y=\text{“Noun”}$ | $y=\text{“Verb”}$ |
|-------------|-----------------|-----------------|
| $x=\text{“fish”}$ | 0.8             | 0.5             |
| $x=\text{“sleep”}$ | 0.2             | 0.5             |
Maximization Step

- Max. Likelihood over Marginal Distribution

**Supervised:**

\[
A_{ab} = \frac{\sum \sum_j P(y_j^i = b) P(y_j^{i+1} = a)}{\sum \sum_j P(y_j = b)}
\]

**Unsupervised:**

\[
A_{ab} = \frac{\sum \sum_j P(y_j^i = b) P(y_j^{i+1} = a)}{\sum \sum_j P(y_j = b)}
\]

**Unsupervised:**

\[
O_{wz} = \frac{\sum \sum_j 1[(x_j^i = w) \land (y_j^i = z)]}{\sum \sum_j P(y_j = z)}
\]

Marginals

65
Computing Marginals
(Forward-Backward Algorithm)

• Solving E-Step, requires compute marginals

<table>
<thead>
<tr>
<th></th>
<th>$x^1$</th>
<th>$x^2$</th>
<th>...</th>
<th>$x^L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(y^i=\text{Noun})$</td>
<td>0.5</td>
<td>0.4</td>
<td>...</td>
<td>0.05</td>
</tr>
<tr>
<td>$P(y^i=\text{Det})$</td>
<td>0.4</td>
<td>0.6</td>
<td>...</td>
<td>0.25</td>
</tr>
<tr>
<td>$P(y^i=\text{Verb})$</td>
<td>0.1</td>
<td>0.0</td>
<td>...</td>
<td>0.7</td>
</tr>
</tbody>
</table>

• Can solve using Dynamic Programming!
  – Similar to Viterbi
Notation

Probability of observing prefix $x^{1:i}$ and having the $i$-th state be $y^i = Z$

$$\alpha_z(i) = P(x^{1:i}, y^i = Z \mid A, O)$$

Probability of observing suffix $x^{i+1:M}$ given the $i$-th state being $y^i = Z$

$$\beta_z(i) = P(x^{i+1:M} \mid y^i = Z, A, O)$$

Computing Marginals = Combining the Two Terms

$$P(y^i = z \mid x) = \frac{a_z(i)\beta_z(i)}{\sum_{z'} a_{z'}(i)\beta_{z'}(i)}$$

http://en.wikipedia.org/wiki/Baum%E2%80%93Welch_algorithm
Forward (sub-)Algorithm

• Solve for every:  \[ \alpha_z(i) = P(x^{1:i}, y^i = Z | A, O) \]

• Naively:
  \[ \alpha_z(i) = P(x^{1:i}, y^i = Z | A, O) = \sum_{y^{1:i-1}} P(x^{1:i}, y^i = Z, y^{1:i-1} | A, O) \]

  Exponential Time!

• Can be computed recursively (like Viterbi)
  \[ \alpha_z(1) = P(y^1 = z | y^0)P(x^1 | y^1 = z) = O_{x^1, z} A_{z, \text{start}} \]
  \[ \alpha_z(i + 1) = O_{x^{i+1}, z} \sum_{j=1}^L \alpha_j(i) A_{z, j} \]

  Viterbi effectively replaces sum with max
Backward (sub-)Algorithm

• Solve for every: \[ \beta_z(i) = P(x^{i+1:M} | y^i = Z, A, O) \]

• Naively:
  \[ \beta_z(i) = P(x^{i+1:M} | y^i = Z, A, O) = \sum_{y^{i+1:L}} P(x^{i+1:M}, y^{i+1:M} | y^i = Z, A, O) \]
  
  Exponential Time!

• Can be computed recursively (like Viterbi)
  \[ \beta_z(L) = 1 \]
  \[ \beta_z(i) = \sum_{j=1}^{L} \beta_j(i+1) A_{j,z} O_{x^{i+1},j} \]
Forward-Backward Algorithm

• Runs Forward
  \[ \alpha_z(i) = P(x^{1:i}, y^i = Z \mid A, O) \]

• Runs Backward
  \[ \beta_z(i) = P(x^{i+1:M} \mid y^i = Z, A, O) \]

• For each training \( x = (x^1, \ldots, x^M) \)
  – Computes each \( P(y^i) \) for \( y = (y^1, \ldots, y^M) \)
  \[
  P(y^i = z \mid x) = \frac{a_z(i) \beta_z(i)}{\sum_{z'} a_{z'}(i) \beta_{z'}(i)}
  \]
Recap: Unsupervised Training

• Train using only word sequences: \[ S = \left\{ x_i \right\}_{i=1}^{N} \]

• y’s are “hidden states”
  – All pairwise transitions are through y’s
  – Hence hidden Markov Model

• Train using EM algorithm
  – Converge to local optimum
Initialization

• How to choose #hidden states?
  
  – By hand
  
  – Cross Validation
    • P(x) on validation data
    • Can compute P(x) via forward algorithm:

\[
P(x) = \sum_y P(x, y) = \sum_z \alpha_z(M)P(End | y^M = z)
\]
Recap: Sequence Prediction & HMMs

- Models pairwise dependences in sequences
  - $x=\text{“I fish often”}$

- Compact: only model pairwise between y’s

- **Main Limitation:** Lots of independence assumptions
  - Poor predictive accuracy
Next Lecture

• Conditional Random Fields
  – Removes many independence assumptions
  – More accurate in practice
  – Can only be trained in supervised setting

• Recitation tomorrow:
  – Recap of Viterbi and Forward/Backward