Learning for Structured Prediction

Linear Methods For Sequence Labeling: Hidden Markov Models vs Structured Perceptron

Ivan Titov
Last Time: Structured Prediction

1. **Selecting feature representation** $\varphi(x, y)$
   - It should be sufficient to discriminate correct structure from incorrect ones
   - It should be possible to decode with it (see (3))

2. **Learning**
   - Which error function to optimize on the training set, for example
     \[
     w \cdot \varphi(x, y^*) - \max_{y' \in \mathcal{Y}(x), y \neq y^*} w \cdot \varphi(x, y') > \gamma
     \]
   - How to make it efficient (see (3))

3. **Decoding**: $y = \arg\max_{y' \in \mathcal{Y}(x)} w \cdot \varphi(x, y')$
   - Dynamic programming for simpler representations $\varphi$?
   - Approximate search for more powerful ones?

We illustrated all these challenges on the example of dependency parsing.
Outline

- Sequence labeling / segmentation problems: settings and example problems:
  - Part-of-speech tagging, named entity recognition, gesture recognition
- Hidden Markov Model
  - Standard definition + maximum likelihood estimation
  - General views: as a representative of linear models
- Perceptron and Structured Perceptron
  - algorithms / motivations
- Decoding with the Linear Model
- Discussion: Discriminative vs. Generative
Sequence Labeling Problems

- **Definition:**
  - **Input:** sequences of variable length \( x = (x_1, x_2, \ldots, x_{|x|}), x_i \in X \)
  - **Output:** every position is labeled \( y = (y_1, y_2, \ldots, y_{|x|}), y_i \in Y \)

- **Examples:**
  - **Part-of-speech tagging**
    \[
    x = \text{John} \quad \text{carried} \quad \text{a} \quad \text{tin} \quad \text{can} \quad .
    \]
    \[
    y = \text{NP} \quad \text{VBD} \quad \text{DT} \quad \text{NN} \quad \text{NN} \quad .
    \]
  - **Named-entity recognition, shallow parsing (“chunking”), gesture recognition from video-streams, …**
Part-of-speech tagging

\[ x = \text{John carried a tin can}. \]
\[ y = \text{NNP VBD DT NN NN or MD?}. \]

- Labels:
  - NNP – proper singular noun
  - VBD – verb, past tense
  - DT – determiner
  - NN – singular noun
  - MD – modal

Consider

\[ x = \text{Tin can cause poisoning …} \]
\[ y = \text{NN MD VB NN …} \]

In fact, even knowing that the previous word is a noun is not enough. If you just predict the most frequent tag for each word you will make a mistake here.

One need to model interactions between labels to successfully resolve ambiguities, so this should be tackled as a structured prediction problem.
Named Entity Recognition

[ORG Chelsea], despite their name, are not based in [LOC Chelsea], but in neighbouring [LOC Fulham].

- Not as trivial as it may seem, consider:
  - [PERS Bill Clinton] will not embarrass [PERS Chelsea] at her wedding
  - Tiger failed to make a birdie in the South Course …

- Encoding example (BIO-encoding)

\[
\begin{align*}
  \mathbf{x} & = \text{Bill Clinton embarrassed Chelsea at her wedding at Astor Courts} \\
  \mathbf{y} & = \text{B-PERS I-PERS O B-PERS O O O O B-LOC I-LOC}
\end{align*}
\]

Chelsea can be a person too!

Is it an animal or a person?
Vision: Gesture Recognition

- Given a sequence of frames in a video annotate each frame with a gesture type:

- Types of gestures:

- It is hard to predict gestures from each frame in isolation, you need to exploit relations between frames and gesture types.

Figures from (Wang et al., CVPR 06)
Outline

- Sequence labeling / segmentation problems: settings and example problems:
  - Part-of-speech tagging, named entity recognition, gesture recognition
- Hidden Markov Model
  - Standard definition + maximum likelihood estimation
  - General views: as a representative of linear models
- Perceptron and Structured Perceptron
  - algorithms / motivations
- Decoding with the Linear Model
- Discussion: Discriminative vs. Generative
Hidden Markov Models

- We will consider the part-of-speech (POS) tagging example

  John carried a tin can .
  NP VBD DT NN NN .

- A “generative” model, i.e.:
  - Model: Introduce a parameterized model of how both words and tags are generated $P(x, y|\theta)$
  - Learning: use a labeled training set to estimate the most likely parameters of the model $\hat{\theta}$
  - Decoding: $y = \text{argmax}_{y'} P(x, y'|\hat{\theta})$
Hidden Markov Models

A simplistic state diagram for noun phrases: $N \rightarrow \text{tags, } M \rightarrow \text{vocabulary size}$

- States correspond to POS tags,
- Words are emitted independently from each POS tag
- Parameters (to be estimated from the training set):
  - Transition probabilities $P(y^{(t)}|y^{(t-1)})$ : $[N \times N]$ matrix
  - Emission probabilities $P(x^{(t)}|y^{(t)})$ : $[N \times M]$ matrix

Example:

```
a   hungry   dog
```

Stationarity assumption: this probability does not depend on the position in the sequence t
Hidden Markov Models

Representation as an instantiation of a graphical model: \( N \) – tags, \( M \) – vocabulary size

- States correspond to POS tags,
- Words are emitted independently from each POS tag
- Parameters (to be estimated from the training set):
  - Transition probabilities \( P(y^{(t)}|y^{(t-1)}) \) : \( [N \times N] \) matrix
  - Emission probabilities \( P(x^{(t)}|y^{(t)}) \) : \( [N \times M] \) matrix

A arrow means that in the generative story \( x^{(4)} \) is generated from some \( P(x^{(4)} | y^{(4)}) \)

Stationarity assumption: this probability does not depend on the position in the sequence \( t \)
Hidden Markov Models: Estimation

- $N$ – the number tags, $M$ – vocabulary size

- Parameters (to be estimated from the training set):
  - Transition probabilities $a_{ji} = P(y(t) = i | y(t-1) = j)$, A - $[N \times N]$ matrix
  - Emission probabilities $b_{ik} = P(x(t) = k | y(t) = i)$, B - $[N \times M]$ matrix

- Training corpus:
  - $x^{(1)} = (\text{In, an, Oct., 19, review, of, …}), y^{(1)} = (\text{IN, DT, NNP, CD, NN, IN, …})$
  - $x^{(2)} = (\text{Ms., Haag, plays, Elianti,}.), y^{(2)} = (\text{NNP, NNP, VBZ, NNP,}.)$
  - …
  - $x^{(L)} = (\text{The, company, said,}…), y^{(L)} = (\text{DT, NN, VBD, NNP,}.)$

- How to estimate the parameters using maximum likelihood estimation?
  - You probably can guess what these estimates should be?
Hidden Markov Models: Estimation

- Parameters (to be estimated from the training set):
  - Transition probabilities \( a_{ji} = P(y^{(t)} = i \mid y^{(t-1)} = j) \), \( A - [N \times N] \) matrix
  - Emission probabilities \( b_{ik} = P(x^{(t)} = k \mid y^{(t)} = i) \), \( B - [N \times M] \) matrix
- Training corpus: \((x^{(l)}, y^{(l)})\), \( l = 1, \ldots, L \)
- Write down the probability of the corpus according to the HMM:

\[
P(\{x^{(l)}, y^{(l)}\}_{l=1}^{L}) = \prod_{l=1}^{L} P(x^{(l)}, y^{(l)}) = \\
= \prod_{l=1}^{L} a_{y^{(l)}_{1}, y^{(l)}} \left( \prod_{t=1}^{L} b_{y^{(l)}_{t}, x^{(l)}_{t}} a_{y^{(l)}_{t}, y^{(l)}_{t+1}} \right) b_{y^{(l)}_{L}, x^{(l)}_{L}} a_{y^{(l)}_{L+1}, y^{(l)}} = \\
= \prod_{i,j=1}^{N} a_{i,j}^{CT(i,j)} \prod_{i=1}^{N} \prod_{k=1}^{M} b_{i,k}^{CE(i,k)}
\]

- \( CT(i,j) \) is \#times tag \( i \) is followed by tag \( j \).
- Here we assume that \$ is a special tag which precedes and succeeds every sentence.
- \( CE(i,k) \) is \#times word \( k \) is emitted by tag \( i \).
Hidden Markov Models: Estimation

Maximize: \[ P(\{ x^{(l)}, y^{(l)} \}_{l=1}^L) = \prod_{i,j=1}^N a_{i,j}^{C_T(i,j)} \prod_{i=1}^N \prod_{k=1}^M b_{i,k}^{C_E(i,k)} \]

Equivalently maximize the logarithm of this:

\[ \log( P(\{ x^{(l)}, y^{(l)} \}_{l=1}^L)) = \sum_{i=1}^N \left( \sum_{j=1}^N C_T(i,j) \log a_{i,j} + \sum_{k=1}^M C_E(i,k) \log b_{i,k} \right) \]

subject to probabilistic constraints:

\[ \sum_{j=1}^N a_{i,j} = 1, \quad \sum_{i=1}^N b_{i,k} = 1, \quad i = 1, \ldots, N \]

Or, we can decompose it into 2N optimization tasks:

For transitions

\[ i = 1, \ldots, N: \]

\[ \max_{a_{i,1}, \ldots, a_{i,N}} \sum_{j=1}^N C_T(i,j) \log a_{i,j} \]

s.t. \[ \sum_{j=1}^N a_{i,j} = 1 \]

For emissions

\[ i = 1, \ldots, N: \]

\[ \max_{b_{i,1}, \ldots, b_{i,N}} C_E(i,k) \log b_{i,k} \]

s.t. \[ \sum_{i=1}^N b_{i,k} = 1 \]
Hidden Markov Models: Estimation

- For transitions (some $i$)
  \[
  \max_{a_i,1,\ldots,a_i,N} \sum_{j=1}^{N} C_T(i, j) \log a_{i,j} \]
  \[
  \text{s.t. } 1 - \sum_{j=1}^{N} a_{i,j} = 0
  \]

- Constrained optimization task, Lagrangian:
  \[
  L(a_i,1, \ldots, a_i,N, \lambda) = \sum_{j=1}^{N} C_T(i, j) \log a_{i,j} + \lambda \times (1 - \sum_{j=1}^{N} a_{i,j})
  \]

- Find critical points of Lagrangian by solving the system of equations:
  \[
  \frac{\partial L}{\partial \lambda} = 1 - \sum_{j=1}^{N} a_{i,j} = 0
  \]
  \[
  \frac{\partial L}{\partial a_{i,j}} = \frac{C_T(i,j)}{a_{i,j}} - \lambda = 0 \implies a_{i,j} = \frac{C_T(i,j)}{\lambda}
  \]

  \[
  P(y^t = j | y^{t-1} = i) = a_{i,j} = \frac{C_T(i,j)}{\sum_j C_T(i,j')}
  \]
  \[
  P(x^t = k | y^t = i) = b_{i,k} = \frac{C_E(i,k)}{\sum_{k'} C_E(i,k')}
  \]

The maximum likelihood solution is just normalized counts of events. Always like this for generative models if all the labels are visible in training.

I ignore “smoothing” to process rare or unseen word tag combinations… Outside of the scope of the seminar.
HMMs as linear models

Decoding: \( y = \arg\max_y P(x, y'|A, B) = \arg\max_y \log P(x, y'|A, B) \)

We will talk about the decoding algorithm slightly later, let us generalize Hidden Markov Model:

\[
\log P(x, y'|A, B) = \sum_{l=1}^{|x|+1} \log b_{y'_i, x_i} + \log a_{y'_i, y'_{i+1}} \\
= \sum_{i=1}^N \sum_{j=1}^N C_T(y', i, j) \times \log a_{i,j} + \sum_{i=1}^N \sum_{k=1}^M C_E(x, y', i, k) \times \log b_{i,k}
\]

- But this is just a linear model!!
Scoring: example

\[(x, y') = \begin{align*}
&\text{John} \quad \text{carried} \quad a \quad \text{tin} \quad \text{can} \\
&\text{NP} \quad \text{VBD} \quad \text{DT} \quad \text{NN} \quad \text{NN}
\end{align*}\]

\[\varphi(x, y') = \begin{pmatrix} 1 & 0 & \ldots & 1 & 1 & 0 & \ldots \end{pmatrix}\]

Unary features \(C_E(x, y', i, k)\) \[
\begin{align*}
w_{ML} &= \begin{pmatrix} \log b_{NP, John} & \log b_{NP, Mary} & \ldots & \log a_{NN, VBD} & \log a_{NN, -} & \log a_{MD, -} & \ldots \end{pmatrix}
\end{align*}
\]

Their inner product is exactly \(\log P(x, y'| A, B)\)

\[w_{ML} \cdot \varphi(x, y') = \sum_{i=1}^{N} \sum_{j=1}^{N} C_T(y', i, j) \times \log a_{i, j} + \sum_{i=1}^{N} \sum_{k=1}^{M} C_E(x, y', i, k) \times \log b_{i, k}\]

But may be there other (and better?) ways to estimate \(w\), especially when we know that HMM is not a faithful model of reality?

It is not only a theoretical question! (we’ll talk about that in a moment)
Feature view

Basically, we define features which correspond to edges in the graph:

\[ x^{(1)} \rightarrow y^{(1)} \rightarrow x^{(2)} \rightarrow y^{(2)} \rightarrow x^{(3)} \rightarrow y^{(3)} \rightarrow x^{(4)} \rightarrow y^{(4)} \rightarrow \ldots \]

Shaded because they are visible (both in training and testing)
Generative modeling

- For a very large dataset (asymptotic analysis):
  - If data is generated from some “true” HMM, then (if the training set is sufficiently large), we are guaranteed to have an optimal tagger.
  - Otherwise, (generally) HMM will not correspond to an optimal linear classifier.
  - Discriminative methods which minimize the error more directly are guaranteed (under some fairly general conditions) to converge to an optimal linear classifier.

- For smaller training sets:
  - Generative classifiers converge faster to their optimal error. [Ng & Jordan, NIPS 01]

Errors on a regression dataset (predict housing prices in Boston area):
Outline

- Sequence labeling / segmentation problems: settings and example problems:
  - Part-of-speech tagging, named entity recognition, gesture recognition
- Hidden Markov Model
  - Standard definition + maximum likelihood estimation
  - General views: as a representative of linear models
- Perceptron and Structured Perceptron
  - algorithms / motivations
- Decoding with the Linear Model
- Discussion: Discriminative vs. Generative
Let us start with a binary classification problem $y \in \{+1, -1\}$.

For binary classification, the prediction rule is: $y = \text{sign} (\mathbf{w} \cdot \varphi(x))$.

Perceptron algorithm, given a training set $\{x^{(l)}, y^{(l)}\}_{l=1}^{L}$:

\[
\mathbf{w} = \mathbf{0} \quad \text{// initialize}
\]

\[
\begin{align*}
\text{do} \\
\quad \text{err} = 0 \\
\quad \text{for } l = 1 \ldots L \quad \text{// over the training examples} \\
\quad \quad \text{if } ( \ y^{(l)}(\mathbf{w} \cdot \varphi(x^{(l)})) < 0 ) \quad \text{// if mistake} \\
\quad \quad \quad \mathbf{w} =+\quad \eta y^{(l)} \varphi(x^{(l)}) \quad \text{// update, } \eta > 0 \\
\quad \quad \quad \text{err} =+\quad \text{// # errors} \\
\quad \quad \text{endif} \\
\quad \text{endfor} \\
\text{while ( err > 0 ) } \quad \text{// repeat until no errors} \\
\text{return } \mathbf{w}
\end{align*}
\]

break ties (0) in some deterministic way
Linear classification

- Linear separable case, “a perfect” classifier:

\[ (w \cdot \varphi(x) + b) = 0 \]

- Linear functions are often written as: \( y = \text{sign} (w \cdot \varphi(x) + b) \), but we can assume that \( \varphi(x)_0 = 1 \) for any \( x \)

Figure adapted from Dan Roth’s class at UIUC
Perceptron: geometric interpretation

\[
\text{if } ( y^{(l)} (w \cdot \varphi(x^{(l)})) < 0) \quad \text{// if mistake} \\
\quad w += \eta y^{(l)} \varphi(x^{(l)}) \quad \text{// update} \\
\text{endif}
\]
Perceptron: geometric interpretation

\[
\text{if } ( y^{(l)}(w \cdot \varphi(x^{(l)})) < 0) \quad \text{// if mistake}
\]
\[
w += \eta y^{(l)} \varphi(x^{(l)}) \quad \text{// update}
\]
\text{endif}
Perceptron: geometric interpretation

\[
\text{if } (y^{(l)}(w \cdot \varphi(x^{(l)})) < 0) \quad // \text{if mistake}
\]
\[
w += \eta y^{(l)} \varphi(x^{(l)}) \quad // \text{update}
\]
\text{endif}
Perceptron: geometric interpretation

\[
\text{if } (y^{(l)}(\mathbf{w} \cdot \varphi(x^{(l)})) < 0) \quad \text{// if mistake}
\]
\[
\mathbf{w} += \eta y^{(l)} \varphi(x^{(l)}) \quad \text{// update}
\]
\text{endif}
Perceptron: algebraic interpretation

if \( (y^{(l)}(w \cdot \varphi(x^{(l)})) < 0) \)  // if mistake
    \( w += \eta y^{(l)} \varphi(x^{(l)}) \)  // update
endif

- We want after the update to increase \( y^{(l)}(w \cdot \varphi(x^{(l)})) \)
- If the increase is large enough than there will be no misclassification
- Let’s see that’s what happens after the update

\[
y^{(l)}((w + \eta y^{(l)} \varphi(x^{(l)})) \cdot \varphi(x^{(l)}))
\]
\[
= y^{(l)}(w \cdot \varphi(x^{(l)})) + \eta(y^{(l)})^2 (\varphi(x^{(l)}) \cdot \varphi(x^{(l)}))
\]

\( (y^{(l)})^2 = 1 \)

- squared norm > 0

- So, the perceptron update moves the decision hyperplane towards
- misclassified \( \varphi(x^{(l)}) \)
The perceptron algorithm, obviously, can only converge if the training set is linearly separable.

It is guaranteed to converge in a finite number of iterations, dependent on how well two classes are separated (Novikoff, 1962).
Averaged Perceptron

- A small modification

\[
\mathbf{w} = \mathbf{0}, \quad \mathbf{w}^\sum = \mathbf{0} \quad // \text{initialize}
\]

for \( k = 1 .. K \)  // for a number of iterations

for \( l = 1 .. L \)  // over the training examples

if \( (\mathbf{y}^{(l)}(\mathbf{w} \cdot \varphi(x^{(l)})) < 0) \)  // if mistake

\( \mathbf{w} += \eta \mathbf{y}^{(l)} \varphi(x^{(l)}) \)  // update, \( \eta > 0 \)

endif

\( \mathbf{w}^\sum += \mathbf{w} \)  // sum of \( \mathbf{w} \) over the course of training

endfor

endfor

return \( \frac{1}{KL} \mathbf{w}^\sum \)

More stable in training: a vector \( \mathbf{w} \) which survived more iterations without updates is more similar to the resulting vector \( \frac{1}{KL} \mathbf{w}^\sum \), as it was added a larger number of times.
Structured Perceptron

- Let us start with structured problem:
  \[ y = \arg\max_{y'} \in \mathcal{Y}(x) w \cdot \varphi(x, y') \]
- Perceptron algorithm, given a training set \( \{x^{(l)}, y^{(l)}\}_{l=1}^L \)

\[
\begin{align*}
  w &= 0 \quad \text{// initialize} \\
  &\text{do} \\
  &\text{err } = 0 \\
  &\text{for } l = 1 \ldots L \quad \text{// over the training examples} \\
  &\quad \hat{y} = \arg\max_{y'} \in \mathcal{Y}(x^{(l)}) w \cdot \varphi(x^{(l)}, y') \quad \text{// model prediction} \\
  &\quad \text{if } ( w \cdot \varphi(x^{(l)}, \hat{y}) > w \cdot \varphi(x^{(l)}, y^{(l)})) \quad \text{// if mistake} \\
  &\quad\quad w += \eta \left( \varphi(x^{(l)}, y^{(l)}) - \varphi(x^{(l)}, \hat{y}) \right) \quad \text{// update} \\
  &\quad\quad \text{err ++} \quad \text{// # errors} \\
  &\text{endif} \\
  &\text{endfor} \\
  &\text{while } ( \text{err} > 0 ) \quad \text{// repeat until no errors} \\
  &\text{return } w
\end{align*}
\]

*Pushes the correct sequence up and the incorrectly predicted one down*
**Str. perceptron: algebraic interpretation**

\[
\text{if } (w \cdot \varphi(x^{(l)}, \hat{y}) > w \cdot \varphi(x^{(l)}, y^{(l)})) \quad \text{// if mistake}
\]
\[w += \eta \left( \varphi(x^{(l)}, y^{(l)}) - \varphi(x^{(l)}, \hat{y}) \right) \quad \text{// update}
\]

- We want after the update to **increase** \(w \cdot (\varphi(x^{(l)}, y^{(l)}) - \varphi(x^{(l)}, \hat{y}))\)
- If the increase is large enough then \(y^{(l)}\) will be scored above \(\hat{y}\)
- Clearly, that this is achieved as this product will be increased by

\[
\eta \| \varphi(x^{(l)}, y^{(l)}) - \varphi(x^{(l)}, \hat{y}) \|^2
\]

There might be other \(y' \in \mathcal{Y}(x^{(l)})\) but we will deal with them on the next iterations.
Structured Perceptron

Positive:
- Very easy to implement
- Often, achieves respectable results
- As other discriminative techniques, does not make assumptions about the generative process
- Additional features can be easily integrated, as long as decoding is tractable

Drawbacks
- "Good" discriminative algorithms should optimize some measure which is closely related to the expected testing results: what perceptron is doing on non-linearly separable data seems not clear
- However, for the averaged (voted) version a generalization bound which describes generalization properties of Perceptron (Freund & Shapire 98)

Later, we will consider more advance learning algorithms
Outline

- Sequence labeling / segmentation problems: settings and example problems:
  - Part-of-speech tagging, named entity recognition, gesture recognition
- Hidden Markov Model
  - Standard definition + maximum likelihood estimation
  - General views: as a representative of linear models
- Perceptron and Structured Perceptron
  - algorithms / motivations
- Decoding with the Linear Model
- Discussion: Discriminative vs. Generative
Decoding with the Linear model

- Decoding: \( y = \arg\max_{y' \in Y(x)} w \cdot \varphi(x, y') \)

- Again a linear model with the following edge features (a generalization of a HMM)

- In fact, the algorithm does not depend on the feature of input (they do not need to be local)
Decoding with the Linear model

- Decoding: \( \mathbf{y} = \arg\max_{\mathbf{y}' \in \mathcal{Y}(\mathbf{x})} \mathbf{w} \cdot \varphi(\mathbf{x}, \mathbf{y}') \)

- Again a linear model with the following edge features (a generalization of a HMM)

- In fact, the algorithm does not depend on the feature of input (they do not need to be local)
Decoding with the Linear model

- Decoding: \( y = \text{argmax}_{y' \in \mathcal{Y}(x)} w \cdot \varphi(x, y') \)

- Let’s change notation:
  - Edge scores \( f_t(y_{t-1}, y_t, x) \) : roughly corresponds to \( \log a_{y_{t-1}, y_t} + \log b_{y_t, x_t} \)
  - Defined for \( t = 0 \) too (“start” feature: \( y_0 = \$ \))
  - Decode: \( y = \text{argmax}_{y' \in \mathcal{Y}(x)} \sum_{t=1}^{\|x\|} f_t(y'_{t-1}, y'_t, x) \)
  - Decoding: a dynamic programming algorithm - Viterbi algorithm

Start/Stop symbol information (\$) can be encoded with them too.
**Viterbi algorithm**

- **Decoding:** \( y = \arg\max_{y' \in \mathcal{Y}(x)} \sum_{t=1}^{\text{|x|}} f_t(y'_{t-1}, y'_t, x) \)

- **Loop invariant:** \((t = 1, \ldots, |x|)\)
  - \(\text{score}_t[y]\) - score of the highest scoring sequence up to position \(t\) ending with \(y\)
  - \(\text{prev}_t[y]\) - previous tag on this sequence

- **Init:** \(\text{score}_0[\$] = 0, \text{score}_0[y] = -\infty\) for other \(y\)

- **Recomputation** \((t = 1, \ldots, |x|)\)
  \[
  \text{prev}_t[y] = \arg\max_{y'} \text{score}_t[y'] + f_t(y', y, x)
  \]
  \[
  \text{score}_t[y] = \text{score}_{t-1}[\text{prev}_t[y]] + f_t(\text{prev}_t[y], y, x)
  \]

- **Return:** retrace prev pointers starting from \(\arg\max_y \text{score}_x[y]\)

Time complexity? \(O(N^2|x|)\)
Outline

- Sequence labeling / segmentation problems: settings and example problems:
  - Part-of-speech tagging, named entity recognition, gesture recognition
- Hidden Markov Model
  - Standard definition + maximum likelihood estimation
  - General views: as a representative of linear models
- Perceptron and Structured Perceptron
  - Algorithms / motivations
- Decoding with the Linear Model
- Discussion: Discriminative vs. Generative
Recap: Sequence Labeling

- Hidden Markov Models:
  - How to estimate

- Discriminative models
  - How to learn with structured perceptron

- Both learning algorithms result in a linear model
  - How to label with the linear models
Discriminative vs Generative

- **Generative models:**
  - **Cheap to estimate:** simply normalized counts
  - Hard to integrate **complex features:** need to come up with a generative story and this story may be wrong
  - Does not result in an optimal classifier when model assumptions are wrong (i.e., always)

- **Discriminative models**
  - **More expensive to learn:** need to run decoding (here, Viterbi) during training and usually multiple times per an example
  - **Easy to integrate features:** though some feature may make decoding intractable
  - Usually **less accurate on small datasets**

*Not necessary the case for generative models with latent variables*
Reminders

- **Speakers**: slides about a week before the talk, meetings with me before/after this point will normally be needed
- **Reviewers**: reviews are accepted only before the day we consider the topic
- **Everyone**: References to the papers to read at GoogleDocs,

- These slides (and previous ones) will be online today
  - speakers: send me the last version of your slides too

- Next time: Jelke about Models of Parsing, PCFGs vs general WCFGs (Michael Collins’ book chapter)