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

$x$ is an input (e.g., sentence), $y$ is an output (syntactic tree)
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
    \[
    x = \text{John} \quad \text{carried} \quad \text{a} \quad \text{tin} \quad \text{can} .
    \]
    \[
    y = \text{NP} \quad \text{VBD} \quad \text{DT} \quad \text{NN} \quad \text{NN} .
    \]
  - Named-entity recognition, shallow parsing (“chunking”), gesture recognition from video-streams, ...
Part-of-speech tagging

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

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

Consider

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

In fact, even knowing that the previous word is a noun is not enough to 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)

$x = \text{Bill Clinton embarrassed Chelsea at her wedding at Astor Courts}$

$y = \text{B-PERS I-PERS O B-PERS O O O O B-LOC I-LOC}$

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:** $y = \arg\max_{y'} P(x, y'|\hat{\theta})$
Hidden Markov Models

A simplistic state diagram for noun phrases: \( N \) – 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

Example:

\begin{align*}
\text{Det} & \quad \text{Adj} & \quad \text{Noun} \\
0.8 & \quad 0.5 & \quad 0.1 \\
0.1 & \quad 0.2 & \quad 0.8 \\
1.0 & \quad 0.5 & \quad 0.5 \\
\text{[0.01 : dog} & \quad \text{[0.01 : herring, ...]} & \quad \text{[0.5 : a} \\
\text{0.01 : hungry, ...]} & \quad \text{0.5 : the]} & \quad \text{0.01 : the]}
\end{align*}

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 \( t \)

- \( y^{(1)} = \text{Det} \)
- \( y^{(2)} = \text{Adj} \)
- \( y^{(3)} = \text{Noun} \)
- \( y^{(4)} \)
- \( x^{(1)} = a \)
- \( x^{(2)} = \text{hungry} \)
- \( x^{(3)} = \text{dog} \)
- \( x^{(4)} \)
Hidden Markov Models: Estimation

- $N$ – the number of 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 estimation should be?
Hidden Markov Models: Estimation

- 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: \((x^{(l)}, y^{(l)})\), \( l = 1, \ldots, L \)

- Write down the probability of the corpus according to the HMM:
  \[
P(\{x^{(l)}, y^{(l)}\}_{l=1}^{L}) = \prod_{l=1}^{L} P(x^{(l)}, y^{(l)}) = \prod_{l=1}^{L} a_{y^{(l)}, y^{(l-1)}} \left( \prod_{t=1}^{L} b_{y^{(l)}, x^{(l)}, a_{y^{(l)}, y^{(l+1)}}} \right) b_{y^{(l)}, x^{(l)}, a_{y^{(l)}, y^{(l+1)}}} \]

  \[
  = \prod_{l=1}^{L} a_{y^{(l)}, y^{(l-1)}} \left( \prod_{t=1}^{L} b_{y^{(l)}, x^{(l)}, a_{y^{(l)}, y^{(l+1)}}} \right) b_{y^{(l)}, x^{(l)}, a_{y^{(l)}, y^{(l+1)}}} \]

\( C_T(i,j) \) is \#times tag \( i \) is followed by tag \( j \).

Here we assume that \$ is a special tag which precedes and succeeds every sentence.

\( C_E(i,k) \) is \#times word \( k \) is emitted by tag \( i \).
Hidden Markov Models: Estimation

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

- 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**
  \( i = 1, \ldots, N: \)
  \[ \max_{a_{i,1}, \ldots, a_{i,N}} \sum_{j=1}^N C_T(i,j) \log a_{i,j} \]
  \[ \text{s.t.} \quad \sum_{j=1}^N a_{i,j} = 1 \]

  **For emissions**
  \( i = 1, \ldots, N: \)
  \[ \max_{b_{i,1}, \ldots, b_{i,M}} C_E(i,k) \log b_{i,k} \]
  \[ \text{s.t.} \quad \sum_{i=1}^N b_{i,k} = 1 \]
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:
  \[
  \frac{\partial L}{\partial \lambda} = 1 - \sum_{j=1}^{N} a_{i,j} = 0 \\
  \frac{\partial L}{\partial a_{ij}} = \frac{C_T(i,j)}{a_{ij}} - \lambda = 0 \implies a_{ij} = \frac{C_T(i,j)}{\lambda}
  \]

  \[
  P(y^t = j \mid y^{t-1} = i) = a_{i,j} = \frac{C_T(i,j)}{\sum_j C_T(i,j')}
  \]

- Similarly, for emissions:
  \[
  P(x^t = k \mid y^t = i) = b_{i,k} = \frac{C_E(i,k)}{\sum_{k'} C_E(i,k')}
  \]

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 score of the seminar
HMMs as linear models

John carried a tin can.

- **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:

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

The number of times tag \( i \) is followed by tag \( j \) in the candidate \( y' \)

The number of times tag \( i \) corresponds to word \( k \) in \((x, y')\)

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

\[(x, y') = \begin{pmatrix} \text{John} \\ \text{NP} \end{pmatrix} \text{ carried } \begin{pmatrix} \text{a} \\ \text{DT} \end{pmatrix} \text{ tin } \begin{pmatrix} \text{NN} \\ \text{NN} \end{pmatrix} \text{ can } . \]

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

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

\[w_{ML} = \begin{pmatrix} \log b_{NP, John} \\ \log b_{NP, Mary} \\ \ldots \\ \log a_{NN, VBD} \\ \log a_{NN, -} \\ \log a_{MD, -} \\ \ldots \end{pmatrix} \]

- 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:

- \( y^{(1)} \)
- \( y^{(2)} \)
- \( y^{(3)} \)
- \( y^{(4)} \)
- \( x^{(1)} \)
- \( x^{(2)} \)
- \( x^{(3)} \)
- \( x^{(4)} \)

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 \( \{x^{(l)}, y^{(l)}\}_{l=1}^{L} \):

\[
\mathbf{w} = \mathbf{0} \quad // \text{initialize} \\
do \\
\text{err} = 0 \\
\text{for } l = 1 \ldots L \\
\quad \text{if } ( y^{(l)}(\mathbf{w} \cdot \varphi(x^{(l)})) < 0) \quad // \text{if mistake} \\
\quad \mathbf{w} += \eta y^{(l)} \varphi(x^{(l)}) \quad // \text{update}, \eta > 0 \\
\quad \text{err} ++ \quad // \# \text{ errors} \\
endfor \\
while (\text{err} > 0) \quad // \text{repeat until no errors} \\
\text{return } \mathbf{w}
\]
Linear classification

- Linear separable case, “a perfect” classifier:

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

- Linear functions are often written as: \[y = \text{sign} (\mathbf{w} \cdot \varphi(x) + b), \text{ but we can assume that } \varphi(x)_0 = 1 \text{ for any } x\]
if $(y^{(l)}(w \cdot \varphi(x^{(l)})) < 0)$  
  $w += \eta y^{(l)} \varphi(x^{(l)})$  
endif
Perceptron: geometric interpretation

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

if \( y^{(l)}(w \cdot \varphi(x^{(l)})) < 0 \) // if mistake
\[ w \leftarrow w + \eta y^{(l)} \varphi(x^{(l)}) \] // 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

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

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

\[
(y^{(l)})^2 = 1 \quad \text{ squared norm } > 0
\]

- 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

\[
\begin{align*}
    w &= 0, \quad w^\Sigma = 0 \quad \text{// initialize} \\
    \text{for } k &= 1 \ldots K \quad \text{// for a number of iterations} \\
    \text{for } l &= 1 \ldots L \quad \text{// over the training examples} \\
    \text{if } (y^{(l)}(w \cdot \varphi(x^{(l)})) < 0) \quad \text{// if mistake} \\
    w &= +\eta y^{(l)} \varphi(x^{(l)}) \quad \text{// update, } \eta > 0 \\
    w^\Sigma &= +w \quad \text{// sum of } w \text{ over the course of training} \\
\end{align*}
\]

Do not run until convergence

Note: it is after endif

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

Let us start with the structured problem: \( y = \arg \max_{y' \in \mathcal{Y}(x)} w \cdot \varphi(x, y') \)

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

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

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

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

- We want after the update to increase \( \mathbf{w} \cdot (\varphi(\mathbf{x}^{(l)}, y^{(l)}) - \varphi(\mathbf{x}^{(l)}, \hat{y})) \)
- 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

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

There might be other \( y' \in \mathcal{Y}(\mathbf{x}^{(l)}) \) 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 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: $y = \arg\max_{y' \in \mathcal{Y}(x)} w \cdot \varphi(x, y')$

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

- 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: \( y = \text{argmax}_{y' \in Y(x)} \sum_{t=1}^{|x|} f_t(y'_{t-1}, y'_t, x) \)
  - Decoding: a dynamic programming algorithm - Viterbi algorithm

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

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

- **Loop invariant:** \((t = 1, \ldots, |x|)\)
  - \(\text{score}_t[y]\) - score of the highest scoring sequence up to position \(t\) with
  - \(\text{prev}_t[y]\) - 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**
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: Lea about Models of Parsing, PCFGs vs general WCFGs (Michael Collins’ book chapter)