Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with the Perceptron Algorithm

Michael Collins
AT&T Labs-Research

THE BASIC PROBLEM

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere .

- There are many possible tags in the position ??

- Need to learn a function from (context, tag) pairs to a real value indicating the “plausibility” of the tag in this context

OVERVIEW

- Representation of the problem
- Log-linear ("Maximum-entropy") taggers
- The perceptron algorithm for parameter estimation
  - The algorithm, experiments, generalization bounds
- Goals
  - Develop alternatives to Max-ent taggers, CRFs
  - Test parameter estimation methods which may be applicable to many models in NLP (e.g., Context-free grammars)

REPRESENTATION: HISTORIES

- A **history** is a 4-tuple $\langle t_{-1}, t_{-2}, w_{[1:n]}, i \rangle$
- t_{-1}, t_{-2} are the previous two tags.
- $w_{[1:n]}$ are the n words in the input sentence.
- i is the index of the word being tagged

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere .

- $t_{-1}, t_{-2} =$ DT, JJ
- $w_{[1:n]} = \langle Hispaniola, quickly, became, \ldots, Hemisphere, . \rangle$
- $i = 6$

FEATURE–VECTOR REPRESENTATIONS

- Take a history/tag pair (h, t).
- $\phi_s(h, t)$ for $s = 1 \ldots d$ are **features** representing tagging decision t in context h.

Example: POS Tagging [Ratnaparkhi 96]

- **Word/tag features**

$$\phi_{100}(h,t) = \begin{cases} 1 & \text{if current word } w_i \text{ is } \texttt{base} \text{ and } t = \texttt{VB} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_{101}(h,t) = \begin{cases} 1 & \text{if current word } w_i \text{ ends in } \texttt{ing} \text{ and } t = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

- **Contextual Features**

$$\phi_{103}(h,t) = \begin{cases} 1 & \text{if } \langle t_{-2}, t_{-1}, t \rangle = \langle \text{DT, JJ, VB} \rangle \\ 0 & \text{otherwise} \end{cases}$$

GLOBAL FEATURES FROM LOCAL FEATURES

- **Global features** $\Phi_s(w_{[1:n]}, t_{[1:n]})$ for $s = 1 \ldots d$ map an entire $(w_{[1:n]}, t_{[1:n]})$ pair to a feature vector

- Global features can be derived from local features

$$\Phi_s(w_{[1:n]}, t_{[1:n]}) = \sum_i \phi_s(h_i, t_i)$$

where $h_i = \langle t_{i-2}, t_{i-1}, w_{[1:n]}, i \rangle$

- Local features are indicator functions $\Rightarrow$ Global features are **counts**

$$\phi_{103}(h, t) = \begin{cases} 1 & \text{if } \langle t_{-1}, t \rangle = \langle \text{DT, NN} \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$\Phi_{103}(w_{[1:n]}, t_{[1:n]}) = \text{Number of times } \langle \text{DT, NN} \rangle \text{ is seen in } t_{[1:n]}$$

e.g., Φ_{103}(the/DT man/NN bit/VBD the/DT dog/NN) = 2

MAXIMUM ENTROPY MODELS

- $\phi_s(h, t)$ for $s = 1 \ldots d$ are **features**
  $\mathbf{W}_s$ for $s = 1 \ldots d$ are **parameters**

- Conditional distribution:

$$P(t|h) = \frac{e^{\sum_s \mathbf{W}_s \phi_s(h,t)}}{Z(h, \mathbf{W})}$$

  where $Z(h, \mathbf{W}) = \sum_{t' \in \mathcal{T}} e^{\sum_s \mathbf{W}_s \phi_s(h,t')}$

- Parameters estimated using maximum-likelihood
  e.g., iterative scaling, gradient descent

MAXIMUM ENTROPY MODELS

- Word sequence $w_{[1:n]} = [w_1, w_2 \ldots w_n]$
- Tag sequence $t_{[1:n]} = [t_1, t_2 \ldots t_n]$
- Histories $h_i = \langle t_{i-1}, t_{i-2}, w_{[1:n]}, i \rangle$

$$\log P(t_{[1:n]} \mid w_{[1:n]})$$

$$= \sum_{i=1}^{n} \log P(t_i \mid h_i)$$

$$= \underbrace{\sum_{i=1}^{n} \sum_{s} \mathbf{W}_s \phi_s(h_i, t_i)}_{\text{Linear Score}} - \underbrace{\sum_{i=1}^{n} \log Z(h_i, \mathbf{W})}_{\text{Local Normalization Terms}}$$

PROBLEMS WITH LOCALLY NORMALIZED MODELS

- "Label bias" problem [Lafferty, McCallum and Pereira 2001]
  See also [Klein and Manning 2002] (the next presentation)

- Example of a conditional distribution that locally normalized models can't capture (under bigram tag representation):

$$\text{a b c} \Rightarrow \begin{array}{ccccc} \text{A} & \text{—} & \text{B} & \text{—} & \text{C} \\ | & & | & & | \\ \text{a} & & \text{b} & & \text{c} \end{array} \quad \text{with } P(\text{A B C} \mid \text{a b c}) = 1$$

$$\text{a b e} \Rightarrow \begin{array}{ccccc} \text{A} & \text{—} & \text{D} & \text{—} & \text{E} \\ | & & | & & | \\ \text{a} & & \text{b} & & \text{e} \end{array} \quad \text{with } P(\text{A D E} \mid \text{a b e}) = 1$$

- Impossible to find parameters that satisfy

$$P(A \mid a) \times P(B \mid b, A) \times P(C \mid c, B) = 1$$

$$P(A \mid a) \times P(D \mid b, A) \times P(E \mid e, D) = 1$$

CONDITIONAL MARKOV RANDOM FIELDS

[Lafferty, McCallum and Pereira 2001],[Johnson et. al 1999]

- "Global" normalization

$$\log P(t_{[1:n]} \mid w_{[1:n]}) = \underbrace{\sum_i \sum_s \mathbf{W}_s \phi_s(h_i, t_i)}_{\text{Linear Score}} - \underbrace{\log Z(\mathbf{W}, w_{[1:n]})}_{\text{Global Normalization}}$$

where

$$Z(\mathbf{W}, w_{[1:n]}) = \sum_{t_{[1:n]} \in \mathcal{T}^n} e^{\sum_i \sum_s \mathbf{W}_s \phi_s(h_i, t_i)}$$

- Computational issues
  - Viterbi algorithm finds most likely tag sequence
  - Gradients w.r.t parameters calculated using Baum-Welch!

A PERCEPTRON ALGORITHM

- Score for a $(w_{[1:n]}, t_{[1:n]})$ pair is

$$
\begin{aligned}
F(w_{[1:n]}, t_{[1:n]}) &= \sum_i \sum_s \mathbf{W}_s \phi_s(h_i, t_i) \\
&= \sum_s \mathbf{W}_s \mathbf{\Phi}_s(t_{[1:n]}, w_{[1:n]})
\end{aligned}
$$

- Note: no normalization terms
- Note: $F(w_{[1:n]}, t_{[1:n]})$ is not a log probability
- Viterbi algorithm for

$$
\arg \max_{t_{[1:n]} \in \mathcal{T}^n} F(w_{[1:n]}, t_{[1:n]})
$$

TRAINING THE PARAMETERS

Inputs: Training set $(w^i_{[1:n_i]}, t^i_{[1:n_i]})$ for $i = 1 \ldots n$.

Initialization: $\mathbf{W} = 0$

Algorithm: For $t = 1 \ldots T, i = 1 \ldots n$

$$z_{[1:n_i]} = \arg \max_{u_{[1:n_i]} \in \mathcal{T}^{n_i}} \sum_s \mathbf{W}_s \mathbf{\Phi}_s(w^i_{[1:n_i]}, u_{[1:n_i]})$$

$z_{[1:n_i]}$ is output on i'th sentence with current parameters

If $z_{[1:n_i]} \neq t^i_{[1:n_i]}$ then

$$\mathbf{W}_s = \mathbf{W}_s \quad + \quad \underbrace{\mathbf{\Phi}_s(w^i_{[1:n_i]}, t^i_{[1:n_i]})}_{\text{Correct tags' feature value}} - \underbrace{\mathbf{\Phi}_s(w^i_{[1:n_i]}, z_{[1:n_i]})}_{\text{Incorrect tags' feature value}}$$

Output: Parameter vector $\mathbf{W}$.

AN EXAMPLE

Say the correct tags for i'th sentence are

the/DT man/NN bit/VBD the/DT dog/NN

Under current parameters, output is

the/DT man/NN bit/NN the/DT dog/NN

Assume also that features track: (1) all bigrams; (2) word/tag pairs

Parameters incremented:

$$\langle \text{NN, VBD} \rangle, \langle \text{VBD, DT} \rangle, \langle \text{VBD} \rightarrow \text{bit} \rangle$$

Parameters decremented:

$$\langle \text{NN, NN} \rangle, \langle \text{NN, DT} \rangle, \langle \text{NN} \rightarrow \text{bit} \rangle$$

A REFINEMENT: AVERAGED PARAMETERS

$\mathbf{W}_{t,i}$ = weight vector at i'th example on t'th pass

$$\mathbf{W}_{AVG} = \sum_{t=1 \ldots T, i=1 \ldots n} \frac{\mathbf{W}_{t,i}}{Tn}$$

Algorithm: For $t = 1 \ldots T, i = 1 \ldots n$

- $z_{[1:n_i]} = \arg\max_{u_{[1:n_i]} \in \mathcal{T}^{n_i}} \sum_s \mathbf{W}_s \mathbf{\Phi}_s(w^i_{[1:n_i]}, u_{[1:n_i]})$
- If $z_{[1:n_i]} \neq t^i_{[1:n_i]}$ then
  $\mathbf{W}_s = \mathbf{W}_s + \mathbf{\Phi}_s(w^i_{[1:n_i]}, t^i_{[1:n_i]}) - \mathbf{\Phi}_s(w^i_{[1:n_i]}, z_{[1:n_i]})$
- $\mathbf{W}_{AVG} = \mathbf{W}_{AVG} + \mathbf{W}/Tn$

EXPERIMENTS

- Wall Street Journal part-of-speech tagging data

  $\approx$ One million words of training data
  $300,000$ words of development data
  $300,000$ words of test data

- [Ramshaw and Marcus 95] NP chunking data

  $\approx$ 150,000 words of training data
  $50,000$ words of development data
  $50,000$ words of test data

RESULTS ON THE POS TAGGING TASK

Development Data:

Method	Error rate/%	Numits
Perc, avg, cc=0	**2.93**	10
Perc, noavg, cc=0	3.68	20
Perc, avg, cc=5	3.03	6
Perc, noavg, cc=5	4.04	17
ME, cc=0	3.4	100
ME, cc=5	**3.28**	200

$cc = 0$ All features included

$cc = 5$ Only features occurring 5 times or more in training

avg Averaged perceptron, **noavg** Regular perceptron

Test Data: Perceptron = 2.89%, Max-ent = 3.28%
(11.9% relative error reduction)

RESULTS ON THE CHUNKING TASK

Development Data:

Method	F-Measure	Numits
Perceptron, avg, cc=0	**93.53**	13
Perceptron, noavg, cc=0	93.04	35
Perceptron, avg, cc=5	93.33	9
Perceptron, noavg, cc=5	91.88	39
Max-ent, cc=0	92.34	900
Max-ent, cc=5	**92.65**	200

$cc = 0$ All features included

$cc = 5$ Only features occurring 5 times or more in training

avg Averaged perceptron, **noavg** Regular perceptron

Test Data: Perceptron = 93.63%, Max-ent = 93.29%
(5.1% relative error reduction)

THEORY BEHIND THE ALGORITHM

- The method as a special case of a more general algorithm
- Convergence of the general algorithm on training data
- Generalization properties

LINEAR MODELS

- $\mathcal{X}$ is set of sentences, $\mathcal{Y}$ is set of possible outputs (e.g. trees)
- Need to learn a function $F : \mathcal{X} \to \mathcal{Y}$
- Three components:

  GEN is a function from a string to a set of candidates

  $\mathbf{\Phi}$ maps a candidate to a feature vector

  W is a parameter vector

- **GEN**, $\mathbf{\Phi}$, **W** define

$$F(x) = \arg\max_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(x, y) \cdot \mathbf{W}$$

Choose the highest scoring candidate

- Given examples (x_i, y_i), how to set **W**?

THE PERCEPTRON ALGORITHM FOR LINEAR MODELS

Inputs: Training set (x_i, y_i) for $i = 1 \ldots n$

Initialization: $\mathbf{W} = 0$

Define: $F(x) = \mathrm{argmax}_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(x, y) \cdot \mathbf{W}$

Algorithm: For $t = 1 \ldots T$, $i = 1 \ldots n$

$z_i = F(x_i)$

If $(z_i \neq y_i)$ $\quad \mathbf{W} = \mathbf{W} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$

Output: Parameters $\mathbf{W}$

THE TAGGING ALGORITHM AS A SPECIAL CASE

- Input is a sentence $w_{[1:n]}$,
  output is a tagged sequence $t_{[1:n]}$

- $\mathbf{GEN}(w_{[1:n]}) = \mathcal{T}^n$
  Generate all tag sequences of length n as candidates

- $\mathbf{\Phi}(w_{[1:n]}, t_{[1:n]}) = \sum_{i=1}^{n} \phi(h_i, t_i)$
  Global representation is defined through local features

- One twist: **GEN** is of size exponential in n
  But can use Viterbi algorithm to calculate

$$F(w_{[1:n]}) = \arg \max_{t_{[1:n]} \in \mathcal{T}^n} \mathbf{\Phi}(w_{[1:n]}, t_{[1:n]}) \cdot \mathbf{W}$$

So the training and decoding algorithms are efficient

THEORY: ERRORS ON TRAINING SAMPLE

- **Training Set:** $(w^i_{[1:n_i]}, t^i_{[1:n_i]})$ for $i = 1 \ldots n$
- **Definition:** The training set is **separable with margin** γ, if there is a vector $\mathbf{U} \in \mathbb{R}^d$ with $||\mathbf{U}|| = 1$ such that

$$\forall i, \forall z_{[1:n]} \in \mathcal{T}^n, z_{[1:n]} \neq t^i_{[1:n_i]} \quad \mathbf{U}\cdot\mathbf{\Phi}(w^i_{[1:n_i]}, t^i_{[1:n_i]}) - \mathbf{U}\cdot\mathbf{\Phi}(w^i_{[1:n_i]}, z_{[1:n]}) \geq \gamma$$

THERE EXISTS $\mathbf{U}$ WHERE $\forall i$ CORRECT TAG SEQUENCE BEATS NEXT BEST SEQUENCE BY AT LEAST γ

Theorem: For any training sequence which is separable with margin γ, then for the perceptron algorithm

$$\text{Number of mistakes} \leq \frac{R^2}{\gamma^2}$$

$$\forall i, \forall z_{[1:n]} \in \mathcal{T}^n, z_{[1:n]} \neq t_{[1:n]}, \ ||\mathbf{\Phi}(w^i_{[1:n_i]}, t^i_{[1:n_i]}) - \mathbf{\Phi}(w^i_{[1:n_i]}, z_{[1:n]})|| \leq R$$

THEORY: ERRORS ON TRAINING SAMPLE

- **Training Set:** (x_i, y_i) for $i = 1 \ldots n$
- **Definition:** The training set is **separable with margin** γ, if there is a vector $\mathbf{U} \in \mathbb{R}^d$ with $||\mathbf{U}|| = 1$ such that

$$\forall i, \forall z \in \mathbf{GEN}(x_i), z \neq y_i \quad \mathbf{U} \cdot \mathbf{\Phi}(x_i, y_i) - \mathbf{U} \cdot \mathbf{\Phi}(x_i, z) \geq \gamma$$

THERE EXISTS $\mathbf{U}$ WHERE $\forall i$ CORRECT TAG SEQUENCE BEATS NEXT BEST SEQUENCE BY AT LEAST γ

Theorem: For any training sequence which is separable with margin γ, then for the perceptron algorithm

$$\text{Number of mistakes} \leq \frac{R^2}{\gamma^2}$$

$\forall i, \forall z \in \mathbf{GEN}(x_i), z \neq y_i, \ ||\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z)|| \leq R$

THEORY: GENERALIZATION BOUNDS

- Assume some (unknown) distribution $D(x,y)$ generating word-sequence/tag sequence pairs (x,y)

- For any tagger h, define

$$E(h) = \sum_{x,y} D(x,y)[[h(x) \neq y]]$$

- Consequence of sample Compression bounds [Littlestone and Warmuth 1986]:

For any distribution $D(x,y)$ on word/tag sequences, with probability at least $1-\delta$ over a sample of size n from D, if the perceptron algorithm makes k mistakes when run to convergence on the training set, the resulting h has

$$E(h) \leq \frac{1}{n-k}\left(k \log \frac{en}{k} + \log \frac{n}{\delta}\right)$$

MORE THEORY FOR THE PERCEPTRON ALGORITHM

- Question 1: what if the data is not separable?

[Freund and Schapire 99] give a modified theorem for this case

- Question 2: performance on training data is all very well, but what about performance on new test examples?

Assume some distribution $P(x, y)$ underlying examples

Theorem [Helmbold and Warmuth 95]: For any distribution $P(x, y)$ generating examples, if e = expected number of mistakes of an online algorithm on a sequence of $m + 1$ examples, then a randomized algorithm trained on m samples will have probability $\frac{e}{m+1}$ of making an error on a newly drawn example from P.

[Freund and Schapire 99] use this to define the **Voted Perceptron**

RELEVANCE TO OTHER WORK IN NLP

- **Basic point**: Any linear model where you can (approximately) find the best scoring hypotheses under current parameters can be discriminatively trained using the perceptron algorithm
- **Many** probabilistic models in NLP are (log) linear models
  - Probabilistic context-free grammars
  - Lexicalized PCFGs
  - Unification grammars [Johnson, Geman, Canon, Chi, and Riezler 1999]
- Generalization bounds, and number of iterations needed for training depend only on **separation**, not on number of candidate tag sequences for each example

SUMMARY

- Local representations:
  Feature vector representation of (history, tag) pairs

- Ranking score for each tagged sequence is a linear combination of representation, parameter vector

- Parameter estimation:
  Perceptron algorithm = Viterbi decoding + additive updates

- Experimental results showed gains over max-ent on POS tagging, NP chunking

- The tagging algorithm is a special case of a general algorithm:
  Analysis is based on separability of training examples