Structured Perceptron

Ye Qiu, Xinghui Lu, Yue Lu,
Ruofei Shen

Outline

Brief review of perceptron

Perceptron Learning Algorithm

```
• $w^1 = 0, b^1 = 0$
• For t = 1 ….
    – Receive example (x,y)
    – If $h(x|w^t) = y$
        • $[w^{t+1}, b^{t+1}] = [w^t, b^t]$
    – Else
        • $w^{t+1} = w^t + yx$
        • $b^{t+1} = b^t + y$
```

$$h(x \mid w) = sign(w^T x - b)$$

Training Set:

$$S = \{(x_i, y_i)\}_{i=1}^{N}$$

$$y \in \{+1, -1\}$$

Go through training set in arbitrary order (e.g., randomly)

Error-Driven Updating: The perceptron algorithm

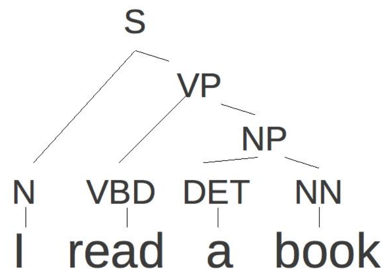
The perceptron is a classic learning algorithm for the neural model of learning. It is one of the most fundamental algorithm. It has two main characteristics:

It is online. Instead of considering the entire data set at the same time, it only ever looks at one example. It processes that example and then goes on to the next one.

It is error driven. If there is no error, it doesn’t bother updating its parameter.

Structured Perceptron

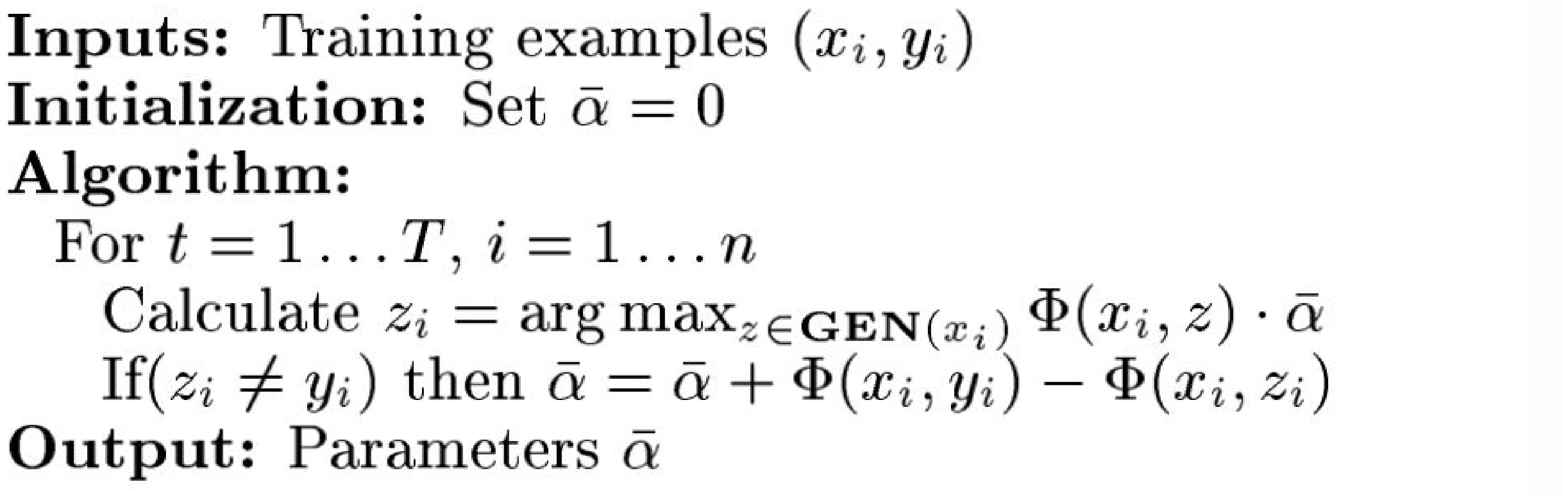
Prediction Problems

Input x	Predict y	Prediction Type
A book review Oh, man I love this book! This book is so boring...	Is it positive? Yes No	Binary prediction (2 choices)
A tweet On the way to the park! 正在去公园	Its language English Chinese	Multi-class prediction (several choices)
A sentence I read a book.	Its syntactic tags  S VP NP N VBD DET NN I read a book	Structured prediction (millions of choices)

Applications of Structured Perceptron

1. POS Tagging with HMMs
   - Collins “Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms” ACL02
2. Parsing
   - Huang+ “Forest Reranking: Discriminative Parsing with Non-Local Features” ACL08
3. Machine Translation
   - Liang+ “An End-to-End Discriminative Approach to Machine Translation” ACL06
4. Vocabulary speech recognition
   - Roark+ “Discriminative Language Modeling with Conditional Random Fields and the Perceptron Algorithm” ACL04

Structured perceptron algorithm



```
Inputs: Training examples $(x_i, y_i)$
Initialization: Set $\bar{\alpha} = 0$
Algorithm:
  For $t = 1 \ldots T$, $i = 1 \ldots n$
    Calculate $z_i = \arg\max_{z \in \mathbf{GEN}(x_i)} \Phi(x_i, z) \cdot \bar{\alpha}$
    If$(z_i \neq y_i)$ then $\bar{\alpha} = \bar{\alpha} + \Phi(x_i, y_i) - \Phi(x_i, z_i)$
Output: Parameters $\bar{\alpha}$
```

Feature representation

What is feature representation?

A feature vector is just a vector that contains information describing an object's important characteristics.

Some example of feature representation in different areas:

In character recognition, features may include histograms counting the number of black pixels along horizontal and vertical directions, number of internal holes, stroke detection and many others.

In spam detection algorithms, features may include the presence or absence of certain email headers, the email structure, the language, the frequency of specific terms, the grammatical correctness of the text.

Feature representation

In POS Tagging problem, general ways to create a feature:

Are capital letters nouns?

Are words that end with -ed verbs？

Can easily try many different ideas!

Some notations

$w_{[1:n]}$	A sequence of words, short-hand for $[w_1, w_2 \ldots w_n]$
$t_{[1:n]}$	A sequence of tags, short-hand for $[t_1, t_2 \ldots t_n]$
h	A history, the context in which a tagging decision is made.
	Usually, a 4-tuple $\langle t_{-1}, t_{-2}, w_{[1:n]}, i \rangle$
ϕ	A feature mapping function. Size is $\mathcal{H} \times \mathcal{T} \rightarrow \mathbb{R}^d$
	Maps a history-tag pair to a d-dimensional feature vector.
	Each component $\phi_s(h, t)$ could be an arbitrary function of h and t.

Some notations

A feature-vector representation

$$\phi : \mathcal{H} \times \mathcal{T} \rightarrow \mathbb{R}^d$$

For example, one such feature might be

$$\phi_{1001}(h, t) = 1 \;\; if \; \langle t_{-2}, t_{-1}, t \rangle = \langle D,\; N,\; V \rangle$$
$$0 \;\; otherwise.$$

Some notations

Φ A function from $(w_{[1:n]}, t_{[1:n]})$ pairs to d-dimensional feature vectors.

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

Refer as a “global” representation, in contrast to ϕ as a “local” representation.

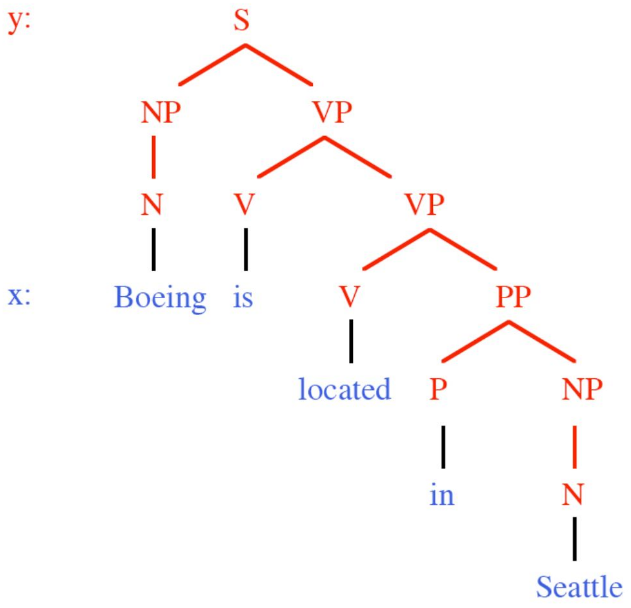
An example of feature vector - POS Tagging

y : <S>	D	N	V	D	N	<E>
x:	The	dog	ate	my	homework.	

$$\Phi(x,y) = \begin{array}{cl} \left(\begin{array}{c} 1 \\ 0 \\ 2 \\ 1 \\ 1 \\ 1 \\ 1 \\ \dots \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{array}\right) & \begin{array}{l} \mathrm{S} \rightarrow \mathrm{D} \\ \mathrm{S} \rightarrow \mathrm{N} \\ \mathrm{D} \rightarrow \mathrm{N} \\ \mathrm{N} \rightarrow \mathrm{V} \\ \mathrm{V} \rightarrow \mathrm{D} \\ \mathrm{N} \rightarrow \text{<E>} \\ \\ \dots \\ \mathrm{N} \rightarrow \text{Dog} \\ \mathrm{N} \rightarrow \text{Cat} \\ \mathrm{V} \rightarrow \text{ate} \\ \\ \\ \\ \dots \end{array} \end{array}$$

Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, Support Vector Machine Learning for Interdependent and Structured Output Spaces, ICML 2004

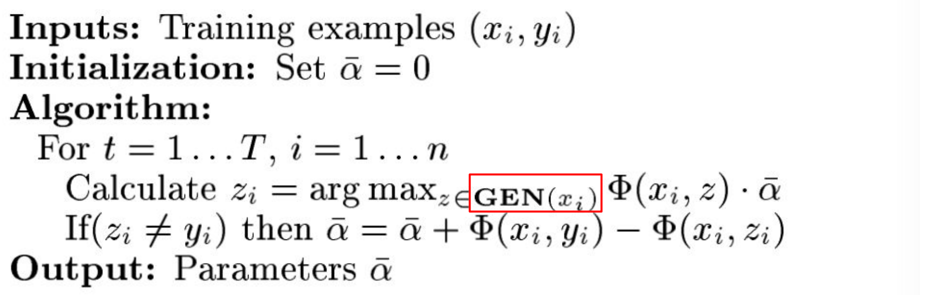
An example of feature vector - Parsing



$$\Phi(x,y) = \begin{array}{c} \left(\begin{array}{c} 1 \\ 0 \\ 2 \\ 1 \\ 1 \\ 1 \\ 1 \\ \dots \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{array}\right) \end{array} \begin{array}{r} \mathrm{S} \rightarrow \mathrm{NP\ VP} \\ \mathrm{S} \rightarrow \mathrm{VP} \\ \mathrm{NP} \rightarrow \mathrm{N} \\ \mathrm{VP} \rightarrow \mathrm{V\ VP} \\ \mathrm{VP} \rightarrow \mathrm{V\ PP} \\ \mathrm{PP} \rightarrow \mathrm{P\ NP} \\ \dots \\ \mathrm{N} \rightarrow \mathrm{Boeing} \\ \mathrm{N} \rightarrow \mathrm{Airbus} \\ \mathrm{V} \rightarrow \mathrm{is} \\ \mathrm{V} \rightarrow \mathrm{located} \\ \mathrm{P} \rightarrow \mathrm{in} \\ \mathrm{N} \rightarrow \mathrm{Seattle} \end{array}$$

Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, Support Vector Machine Learning for Interdependent and Structured Output Spaces, ICML 2004

Some notations



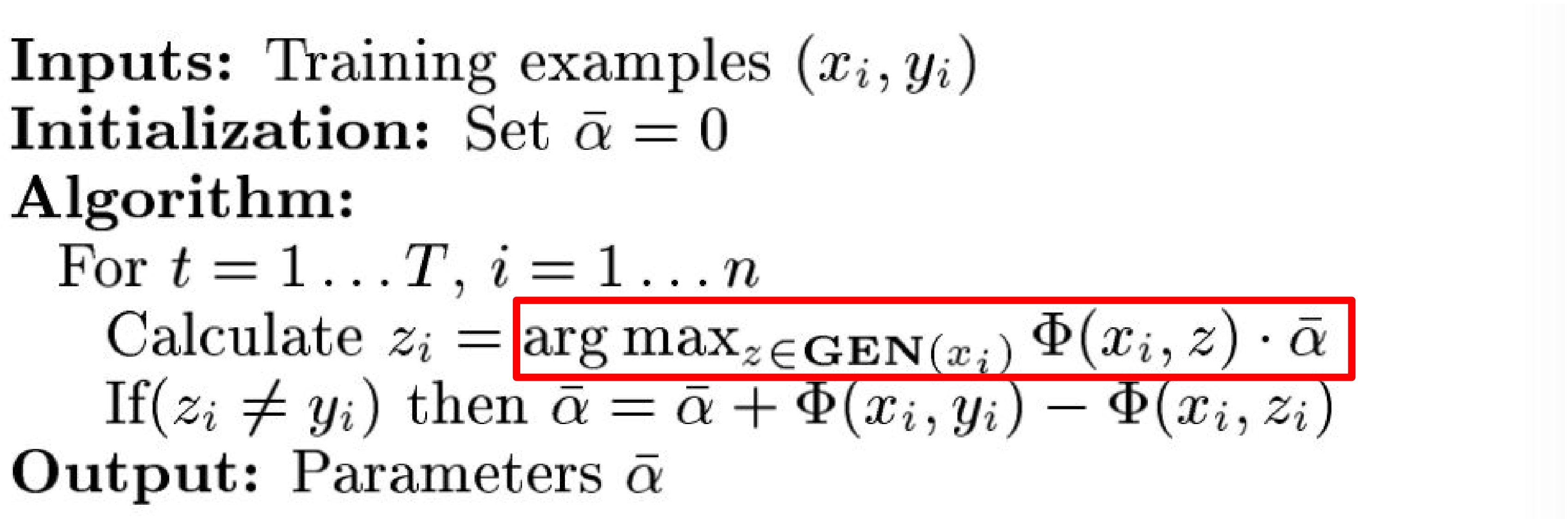
```
**Inputs:** Training examples $(x_i, y_i)$
**Initialization:** Set $\bar{\alpha} = 0$
**Algorithm:**
  For $t = 1 \ldots T$, $i = 1 \ldots n$
    Calculate $z_i = \arg\max_{z \in \boxed{\mathbf{GEN}(x_i)}} \Phi(x_i, z) \cdot \bar{\alpha}$
    If$(z_i \neq y_i)$ then $\bar{\alpha} = \bar{\alpha} + \Phi(x_i, y_i) - \Phi(x_i, z_i)$
**Output:** Parameters $\bar{\alpha}$
```

$\mathbf{GEN}(x)$ A function which enumerates a set of candidates for an input x. Given a set of possible tags T , we define

$$\mathbf{GEN}(w_{[1:n]}) = \mathcal{T}^n$$

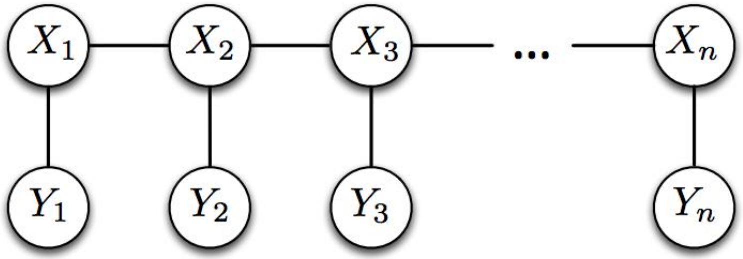
$\bar{\alpha}$ A d-dimensional parameter vector

Recap: Structured perceptron algorithm



```
**Inputs:** Training examples $(x_i, y_i)$
**Initialization:** Set $\bar{\alpha} = 0$
**Algorithm:**
  For $t = 1 \dots T$, $i = 1 \dots n$
    Calculate $z_i = \boxed{\arg\max_{z \in \mathbf{GEN}(x_i)} \Phi(x_i, z) \cdot \bar{\alpha}}$
    If$(z_i \neq y_i)$ then $\bar{\alpha} = \bar{\alpha} + \Phi(x_i, y_i) - \Phi(x_i, z_i)$
**Output:** Parameters $\bar{\alpha}$
```

Recap: Decoding - Viterbi Algorithm in HMM



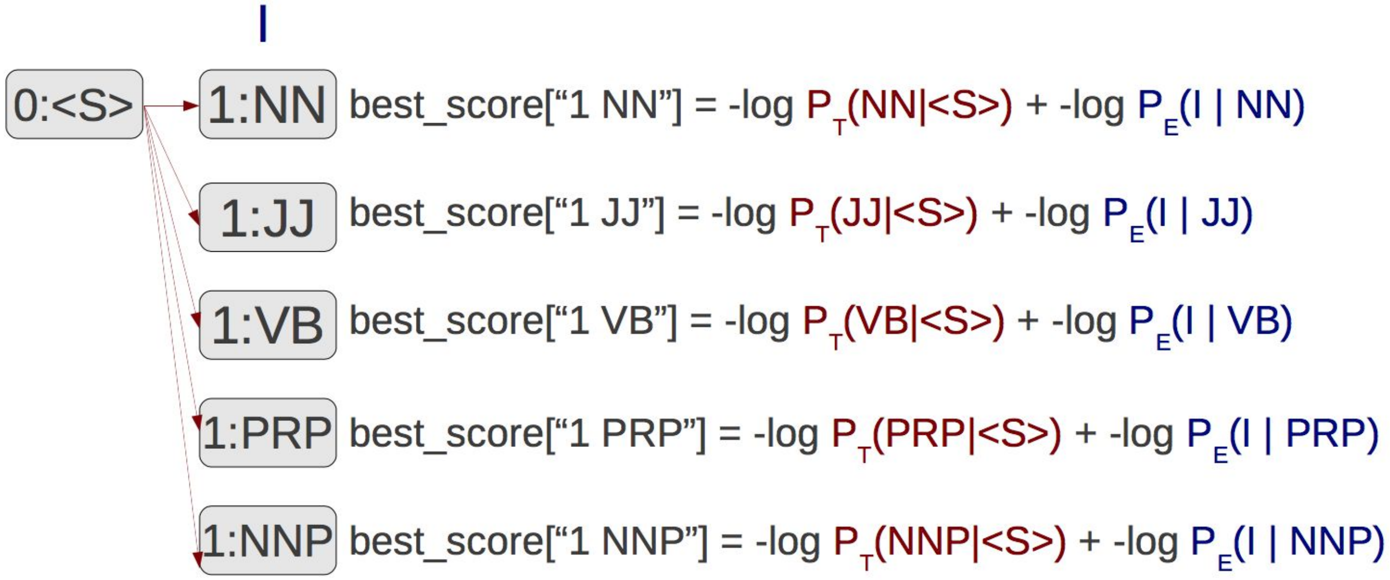
What is decoding problem?

Given a sequence of symbols (your observations) and a model, what is the most likely sequence of states that produced the sequence.

$$\underset{y}{\operatorname{argmax}} P(y \mid x)$$

An example of Viterbi Algorithm

First, calculate transition from <S> and emission of the first word for every POS.



best_score["1 NN"] = -log P_T(NN|<S>) + -log P_E(I | NN)

best_score["1 JJ"] = -log P_T(JJ|<S>) + -log P_E(I | JJ)

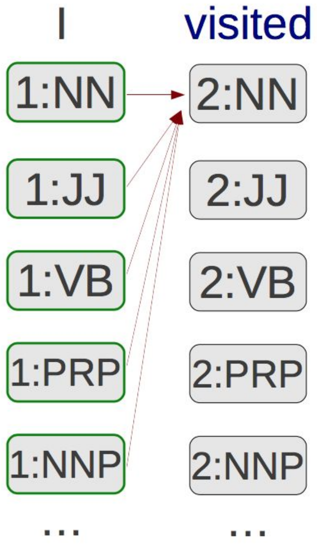
best_score["1 VB"] = -log P_T(VB|<S>) + -log P_E(I | VB)

best_score["1 PRP"] = -log P_T(PRP|<S>) + -log P_E(I | PRP)

best_score["1 NNP"] = -log P_T(NNP|<S>) + -log P_E(I | NNP)

An example of Viterbi Algorithm

For middle words, calculate the maximum score for all possible previous POS tags.

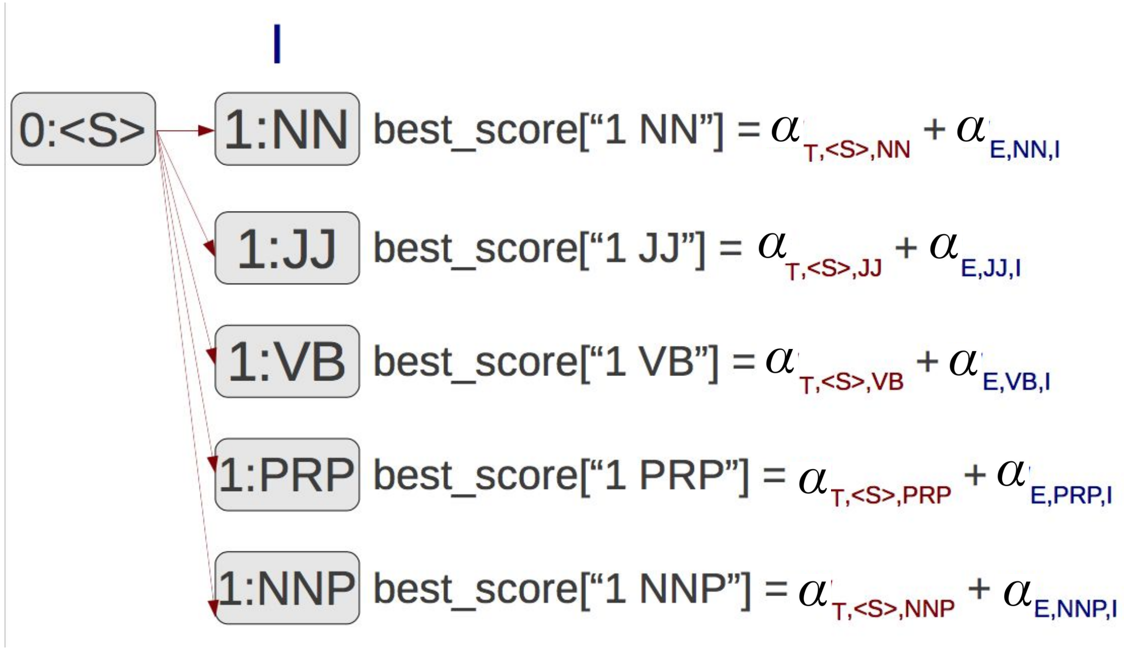


best_score[“2 NN”] = max(
best_score[“1 NN”] + -log P_T(NN|NN) + -log P_E(visited | NN),
best_score[“1 JJ”] + -log P_T(NN|JJ) + -log P_E(language | NN),
best_score[“1 VB”] + -log P_T(NN|VB) + -log P_E(language | NN),
best_score[“1 PRP”] + -log P_T(NN|PRP) + -log P_E(language | NN),
best_score[“1 NNP”] + -log P_T(NN|NNP) + -log P_E(language | NN),
...
)

best_score[“2 JJ”] = max(
best_score[“1 NN”] + -log P_T(JJ|NN) + -log P_E(language | JJ),
best_score[“1 JJ”] + -log P_T(JJ|JJ) + -log P_E(language | JJ),
best_score[“1 VB”] + -log P_T(JJ|VB) + -log P_E(language | JJ),
...

Viterbi algorithm with features

Same as probabilities, use feature weights



best_score["1 NN"] = $\alpha_{T,<S>,NN} + \alpha_{E,NN,I}$

best_score["1 JJ"] = $\alpha_{T,<S>,JJ} + \alpha_{E,JJ,I}$

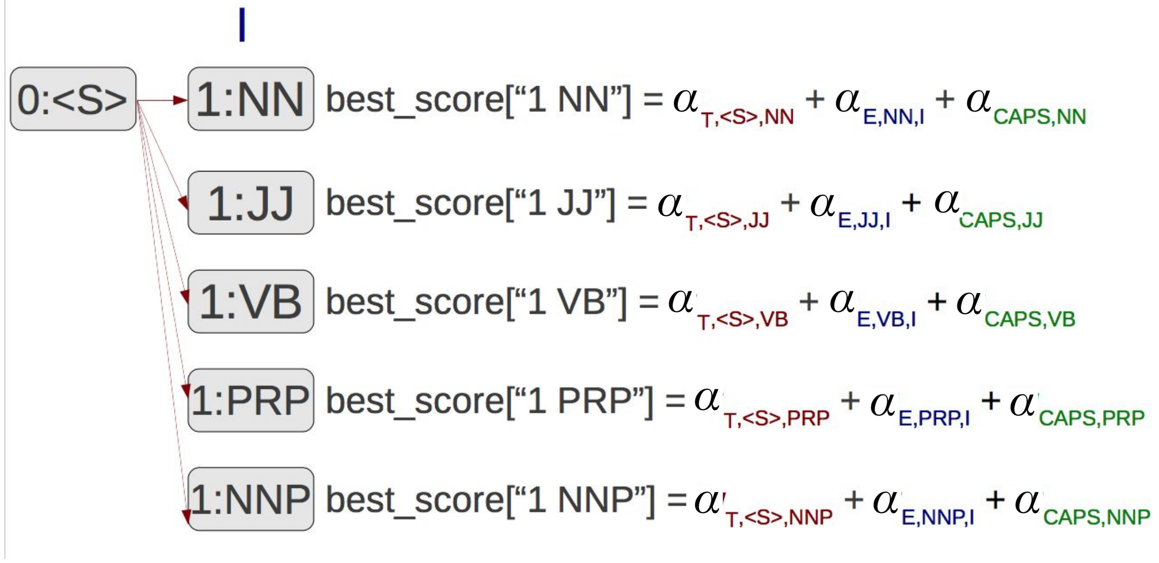
best_score["1 VB"] = $\alpha_{T,<S>,VB} + \alpha_{E,VB,I}$

best_score["1 PRP"] = $\alpha_{T,<S>,PRP} + \alpha_{E,PRP,I}$

best_score["1 NNP"] = $\alpha_{T,<S>,NNP} + \alpha_{E,NNP,I}$

Viterbi algorithm with features

Can add additional features.



best_score["1 NN"] = $\alpha_{T,<S>,NN} + \alpha_{E,NN,I} + \alpha_{CAPS,NN}$

best_score["1 JJ"] = $\alpha_{T,<S>,JJ} + \alpha_{E,JJ,I} + \alpha_{CAPS,JJ}$

best_score["1 VB"] = $\alpha_{T,<S>,VB} + \alpha_{E,VB,I} + \alpha_{CAPS,VB}$

best_score["1 PRP"] = $\alpha_{T,<S>,PRP} + \alpha_{E,PRP,I} + \alpha_{CAPS,PRP}$

best_score["1 NNP"] = $\alpha_{T,<S>,NNP} + \alpha_{E,NNP,I} + \alpha_{CAPS,NNP}$

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

Michael Collins

Outline

1. A motivating example
2. Algorithm for tagging
3. General Algorithm
    a. Proof for Convergences
4. Experiment result

Part of Speech (POS) Tagging

1. Maximum-entropy (ME) models
   a. Advantages: flexible in the features that can be incorporated in the model.
   b. Disadvantages: inference problem for such models is intractable
2. Conditional Random Fields (CRFs)
   a. Better than ME
3. Structured Perceptron
   a. 11.9% relative reduction in error for POS tagging compared with ME

A Motivating Example

- Trigram HMM taggers (Second order HMMs)
- Parameters $\alpha_{x,y,z}$ - trigram <x, y, z>, $\alpha_{t,w}$ - tag/word pair <t, w>
- Common approach(maximum-likelihood):

$$\alpha_{x,y,z} = \log P(z \mid x, y),\ \alpha_{t,w} = \log P(w \mid t).$$

The score for a tag sequence t[1:n] with a word sequence w[1:n]

$$\textstyle\sum_{i=1}^{n} \alpha_{t_{i-2},t_{i-1},t_i} + \sum_{i=1}^{n} \alpha_{t_i,w_i}$$

Using Viterbi algorithm to find the highest scoring.

A Motivating Example

- An alternative to maximum-likelihood parameter estimates
    - Choose a T defining the number of iterations over the training set.
    - Initially set all parameters $\alpha_{x,y,z}$ and $\alpha_{t,w}$ to be zero.
    - In each iteration, use Viterbi to find best tagged sequence for w[1:n] to get z[1:n]
    - Update parameters:
        - <x, y, z> c1 times in t[1:n], c2 times in z[1:n]
        - $\alpha_{x,y,z} = \alpha_{x,y,z} + c_1 - c_2$
        - <t, w> c1 times in t[1:n], c2 times in z[1:n]
        - $\alpha_{t,w} = \alpha_{t,w} + c_1 - c_2$

A Motivating Example

- An example
  - Training data :
    - the/D man/N saw/V the/D dog/N
  - The highest scoring tag sequence z[1:n]:
    - the/D man/N saw/N the/D dog/N
  - Update parameters
    - $\alpha_{D,N,V}$, $\alpha_{N,V,D}$, $\alpha_{V,D,N}$, $\alpha_{V,saw}$ + 1
    - $\alpha_{D,N,N}$, $\alpha_{N,N,D}$, $\alpha_{N,D,N}$, $\alpha_{N,saw}$ - 1
    - If tag sequence is correct -- no change.

Generalize the algorithm to tagged sequences

- Recap: feature vector representations
  - maps a history-tag pairs to a d-dimensional feature vector

$$\phi : \mathcal{H} \times \mathcal{T} \to \mathbb{R}^d$$

Each components φ is an arbitrary function of (h, t), eg:

h: <t-1, t-2, w[1:n], i>

$$\phi_{1000}(h,t) = \begin{cases} 1 & \text{if current word } w_i \text{ is } \texttt{the} \\ & \text{and } t = \texttt{DT} \\ 0 & \text{otherwise} \end{cases}$$

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

General Representation

- Maximum-Entropy Taggers
  - Conditional probability:

$$P(t \mid h, \bar{\alpha}) = \frac{e^{\sum_s \alpha_s \phi_s(h,t)}}{Z(h, \bar{\alpha})}$$

Log probability:

$$\sum_i \sum_{s=1}^{d} \alpha_s \phi_s(h_i, t_i) - \sum_i \log Z(h_i, \bar{\alpha})$$

New estimate method

The "score" of a tagged sequence

$$\sum_{i=1}^{n} \sum_{s=1}^{d} \alpha_s \phi_s(h_i, t_i) = \sum_{s=1}^{d} \alpha_s \Phi_s(w_{[1:n]}, t_{[1:n]})$$

The training algorithm for tagging

- Inputs: A training set of tagged sentences $(w^i_{[1:n_i]}, t^i_{[1:n_i]})$
- Initialization: Set parameter vector $\bar{\alpha} = 0.$
- For t = 1 … T, i = 1 ... n:
  - $z_{[1:n_i]} = \arg\max_{u_{[1:n_i]} \in \mathcal{T}^{n_i}} \sum_s \alpha_s \Phi_s(w^i_{[1:n_i]}, u_{[1:n_i]})$
  - If $z_{[1:n_i]} \neq t^i_{[1:n_i]}$:

    $$\alpha_s = \alpha_s + \Phi_s(w^i_{[1:n_i]}, t^i_{[1:n_i]}) - \Phi_s(w^i_{[1:n_i]}, z_{[1:n_i]})$$
- Output: Parameter vector
- Refinement: “averaged parameters” method

General Algorithm

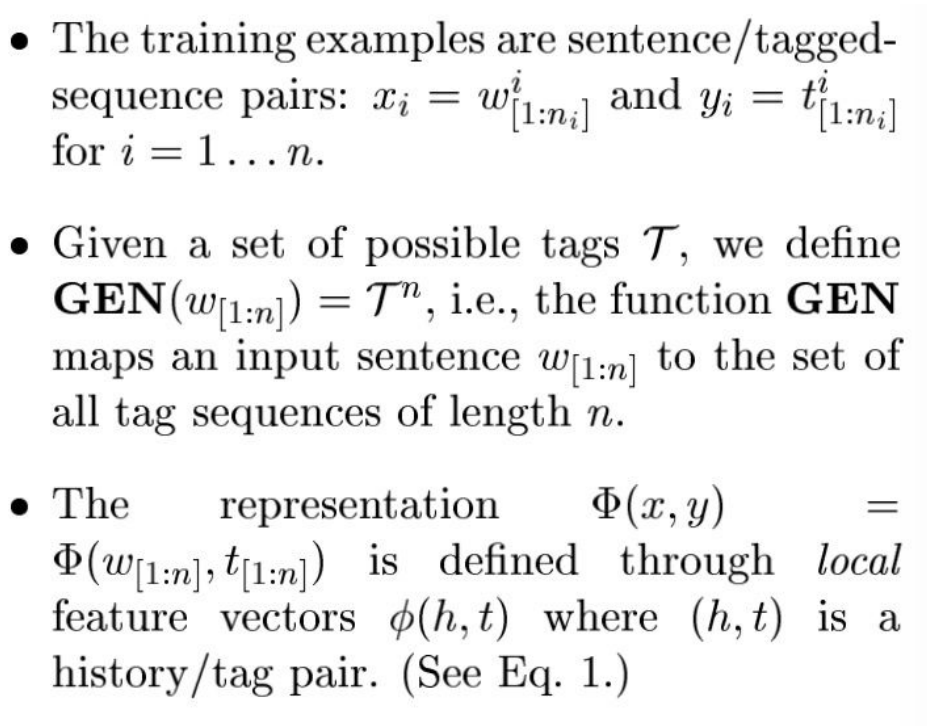
- Training examples (x_i, y_i) for $i = 1 \dots n$.
- A function **GEN** which enumerates a set of candidates $\mathbf{GEN}(x)$ for an input x.
- A **representation** Φ mapping each $(x, y) \in \mathcal{X} \times \mathcal{Y}$ to a feature vector $\Phi(x, y) \in \mathbb{R}^d$.
- A **parameter vector** $\bar{\alpha} \in \mathbb{R}^d$.

The components **GEN**, Φ and $\bar{\alpha}$ define a mapping from an input x to an output $F(x)$ through

$$F(x) = \arg \max_{y \in \mathbf{GEN}(x)} \Phi(x, y) \cdot \bar{\alpha} \longrightarrow \textstyle\sum_s \alpha_s \Phi_s(x, y)$$

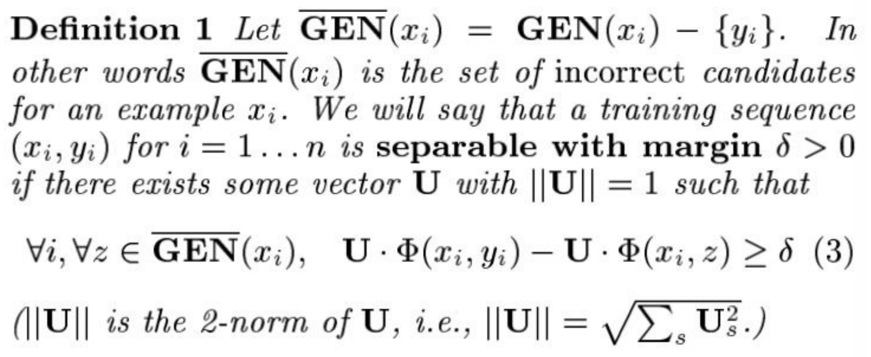
```
Inputs: Training examples $(x_i, y_i)$
Initialization: Set $\bar{\alpha} = 0$
Algorithm:
  For $t = 1 \dots T$, $i = 1 \dots n$
    Calculate $z_i = \arg\max_{z \in \mathbf{GEN}(x_i)} \Phi(x_i, z) \cdot \bar{\alpha}$
    If$(z_i \neq y_i)$ then $\bar{\alpha} = \bar{\alpha} + \Phi(x_i, y_i) - \Phi(x_i, z_i)$
Output: Parameters $\bar{\alpha}$
```

Assumption for Algorithm for taggers



- The training examples are sentence/tagged-sequence pairs: $x_i = w^i_{[1:n_i]}$ and $y_i = t^i_{[1:n_i]}$ for $i = 1 \dots n$.
- Given a set of possible tags $\mathcal{T}$, we define $\mathbf{GEN}(w_{[1:n]}) = \mathcal{T}^n$, i.e., the function **GEN** maps an input sentence $w_{[1:n]}$ to the set of all tag sequences of length n.
- The representation $\Phi(x, y) = \Phi(w_{[1:n]}, t_{[1:n]})$ is defined through *local* feature vectors $\phi(h, t)$ where (h, t) is a history/tag pair. (See Eq. 1.)

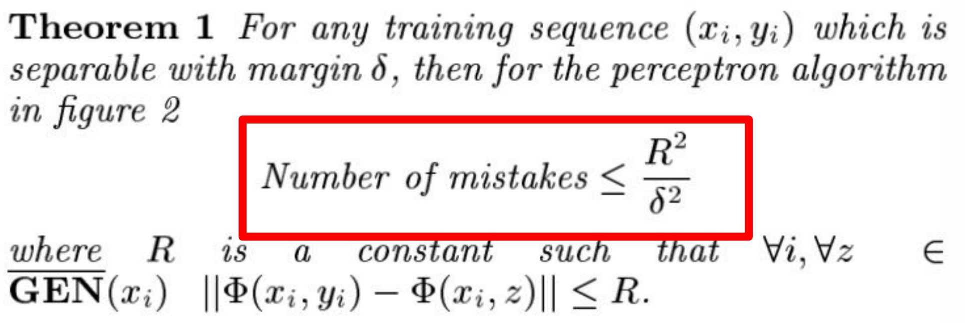
Convergence for separable data



Definition 1 *Let* $\overline{\mathbf{GEN}}(x_i) = \mathbf{GEN}(x_i) - \{y_i\}$. *In other words* $\overline{\mathbf{GEN}}(x_i)$ *is the set of* incorrect *candidates for an example* x_i. *We will say that a training sequence* (x_i, y_i) *for* $i = 1 \ldots n$ *is* **separable with margin** $\delta > 0$ *if there exists some vector* $\mathbf{U}$ *with* $||\mathbf{U}|| = 1$ *such that*

$$\forall i, \forall z \in \overline{\mathbf{GEN}}(x_i), \quad \mathbf{U} \cdot \Phi(x_i, y_i) - \mathbf{U} \cdot \Phi(x_i, z) \geq \delta \quad (3)$$

($||\mathbf{U}||$ *is the 2-norm of* $\mathbf{U}$, *i.e.,* $||\mathbf{U}|| = \sqrt{\sum_s \mathbf{U}_s^2}$.*)*



Theorem 1 *For any training sequence* (x_i, y_i) *which is separable with margin* δ, *then for the perceptron algorithm in figure 2*

$$\boxed{\textit{Number of mistakes} \leq \frac{R^2}{\delta^2}}$$

where R *is a constant such that* $\forall i, \forall z \in \overline{\mathbf{GEN}}(x_i)$ $||\Phi(x_i, y_i) - \Phi(x_i, z)|| \leq R$.

Independent of the number of candidates for each example,
Depending only on the separation of the training data

Proof of Theorem 1

$\bar{\alpha}^k$: the weights before k'th mistake is made.

$$\bar{\alpha}^1 = 0$$

$$z = \arg\max_{y \in \mathbf{GEN}(x_i)} \Phi(x_i, y) \cdot \bar{\alpha}^k$$

$$\bar{\alpha}^{k+1} = \bar{\alpha}^k + \Phi(x_i, y_i) - \Phi(x_i, z)$$

$$\begin{aligned} \mathbf{U} \cdot \bar{\alpha}^{k+1} &= \mathbf{U} \cdot \bar{\alpha}^k + \mathbf{U} \cdot \Phi(x_i, y_i) - \mathbf{U} \cdot \Phi(x_i, z) \\ &\geq \mathbf{U} \cdot \bar{\alpha}^k + \delta \end{aligned}$$

$\mathbf{U} \cdot \bar{\alpha}^{k+1} \geq k\delta.$ (induction)

$$\mathbf{U} \cdot \bar{\alpha}^{k+1} \leq ||\mathbf{U}|| \; ||\bar{\alpha}^{k+1}|| \quad \Rightarrow \quad ||\bar{\alpha}^{k+1}|| \geq k\delta$$

$$\mathbf{U} \cdot \bar{\alpha}^{k+1} \leq ||\mathbf{U}|| \; ||\bar{\alpha}^{k+1}||$$

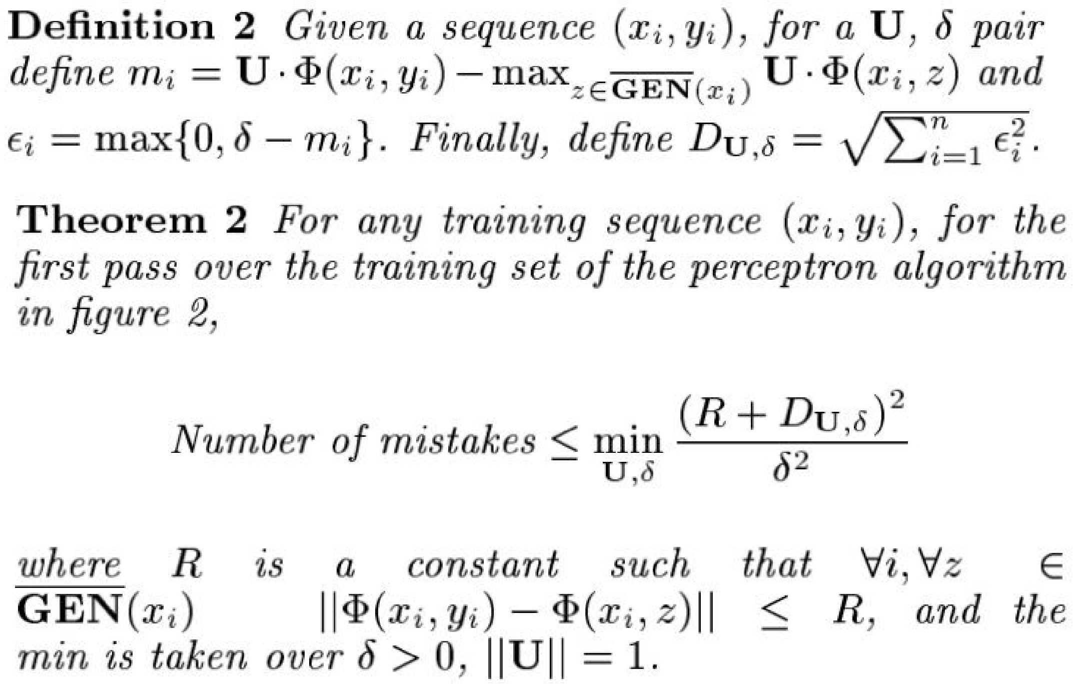
$$||\Phi(x_i, y_i) - \Phi(x_i, z)||^2 \leq R^2$$

$$\bar{\alpha}^k \cdot (\Phi(x_i, y_i) - \Phi(x_i, z)) \leq 0$$

$$\begin{aligned} ||\bar{\alpha}^{k+1}||^2 &= ||\bar{\alpha}^k||^2 + ||\Phi(x_i, y_i) - \Phi(x_i, z)||^2 \\ &\quad + 2\bar{\alpha}^k \cdot (\Phi(x_i, y_i) - \Phi(x_i, z)) \\ &\leq ||\bar{\alpha}^k||^2 + R^2 \end{aligned}$$

$$k^2\delta^2 \leq ||\bar{\alpha}^{k+1}||^2 \leq kR^2 \quad \Rightarrow \quad k \leq R^2/\delta^2$$

Convergence for inseparable data



Definition 2 *Given a sequence* (x_i, y_i), *for a* $\mathbf{U}$, δ *pair define* $m_i = \mathbf{U} \cdot \Phi(x_i, y_i) - \max_{z \in \overline{\mathbf{GEN}}(x_i)} \mathbf{U} \cdot \Phi(x_i, z)$ *and* $\epsilon_i = \max\{0, \delta - m_i\}$. *Finally, define* $D_{\mathbf{U},\delta} = \sqrt{\sum_{i=1}^{n} \epsilon_i^2}$.

Theorem 2 *For any training sequence* (x_i, y_i), *for the first pass over the training set of the perceptron algorithm in figure 2,*

$$\textit{Number of mistakes} \leq \min_{\mathbf{U},\delta} \frac{(R + D_{\mathbf{U},\delta})^2}{\delta^2}$$

where R *is a constant such that* $\forall i, \forall z \in \mathbf{GEN}(x_i)$ $||\Phi(x_i, y_i) - \Phi(x_i, z)|| \leq R$, *and the min is taken over* $\delta > 0$, $||\mathbf{U}|| = 1$.

Proof of Theorem 2

$\Phi(x,y) \in \mathbb{R}^d$ $\qquad \bar{\Phi}(x,y) \in \mathbb{R}^{d+n}$ $\qquad \bar{\mathbf{U}} \in \mathbb{R}^{d+n}$

$i = 1 \ldots d \quad \bar{\Phi}_i(x,y) = \Phi_i(x,y)$ $\qquad \bar{\mathbf{U}}_i = \mathbf{U}_i$

$j = 1 \ldots n \quad \bar{\Phi}_{d+j}(x,y) = \Delta$ if $(x,y) = (x_j, y_j)$ $\qquad \bar{\mathbf{U}}_{d+j} = \epsilon_j/\Delta$

0 otherwise $\qquad$ 0 otherwise

$$\forall i, \forall z \in \overline{\mathbf{GEN}}(x_i),\ \bar{\mathbf{U}} \cdot \bar{\Phi}(x_i, y_i) - \bar{\mathbf{U}} \cdot \bar{\Phi}(x_i, z) \geq \delta$$
$$\forall i, \forall z \in \overline{\mathbf{GEN}}(x_i),\ ||\bar{\Phi}(x_i, y_i) - \bar{\Phi}(x_i, z)||^2 \leq R^2 + \Delta^2$$
$$||\bar{\mathbf{U}}||^2 = ||\mathbf{U}||^2 + \sum_i \epsilon_i^2/\Delta^2 = 1 + D_{\mathbf{U},\delta}^2/\Delta^2$$

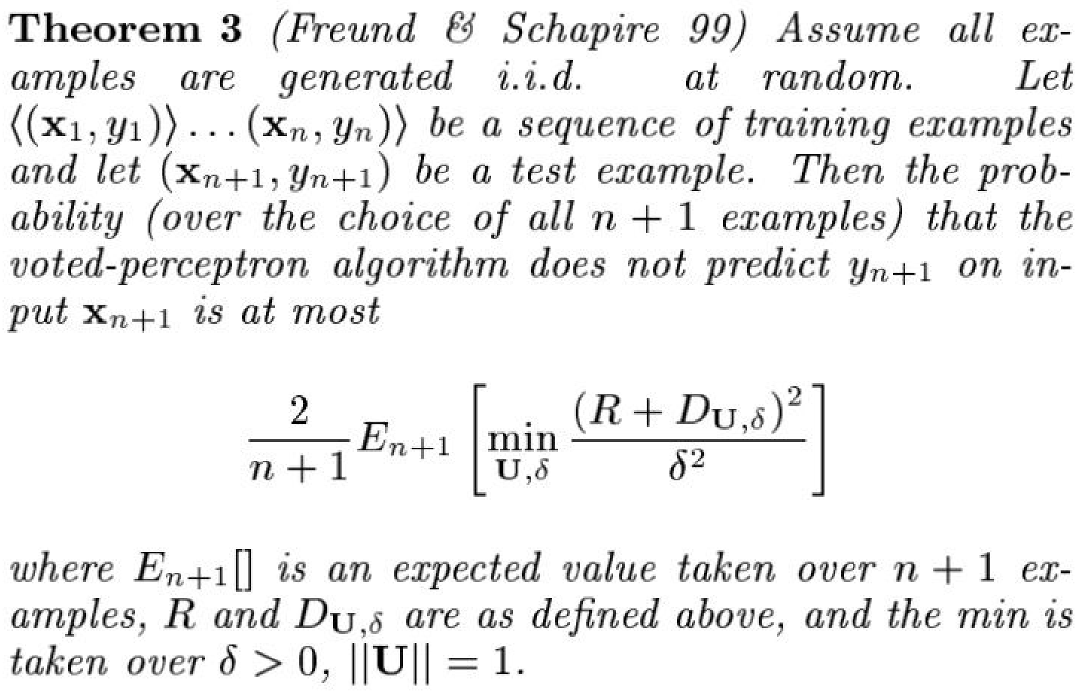
$\bar{\mathbf{U}}/||\bar{\mathbf{U}}||$ ($\bar{\Phi}(x_i, y_i) - \bar{\Phi}(x_i, z)$) >= $\delta/\sqrt{1 + D_{\mathbf{U},\delta}^2/\Delta^2}$

$$\Delta = \sqrt{R D_{\mathbf{U},\delta}}$$

Using Theorem 1:

$$k_{max}(\Delta) = \frac{1}{\delta^2}(R^2 + \Delta^2)\left(1 + \frac{D_{\mathbf{U},\delta}^2}{\Delta^2}\right) \quad \Rightarrow \quad k_{max}(\sqrt{R D_{\mathbf{U},\delta}}) = (R^2 + D_{\mathbf{U},\delta}^2)/\delta^2$$

Generalization to New Test Examples

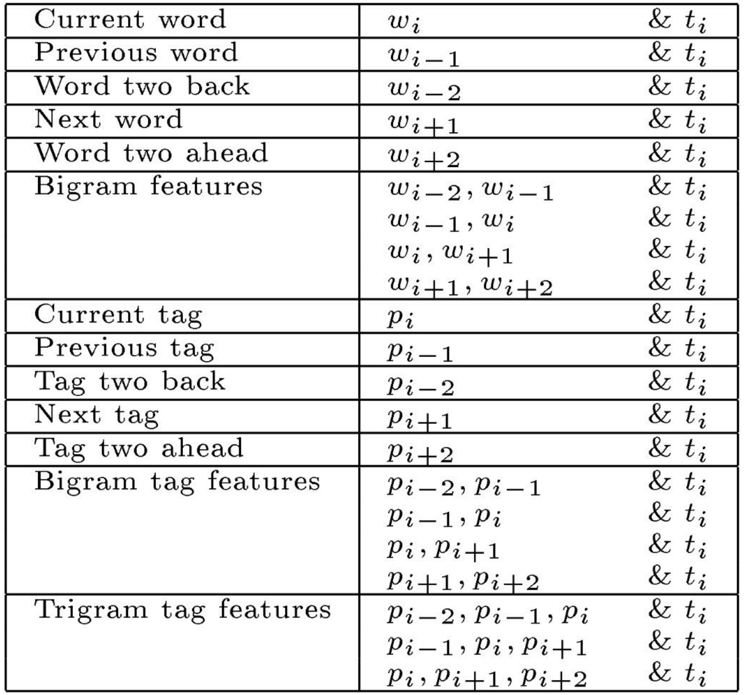


Theorem 3 *(Freund & Schapire 99) Assume all examples are generated i.i.d. at random. Let* $\langle(\mathbf{x}_1, y_1)\rangle \ldots (\mathbf{x}_n, y_n)\rangle$ *be a sequence of training examples and let* $(\mathbf{x}_{n+1}, y_{n+1})$ *be a test example. Then the probability (over the choice of all* $n+1$ *examples) that the voted-perceptron algorithm does not predict* y_{n+1} *on input* $\mathbf{x}_{n+1}$ *is at most*

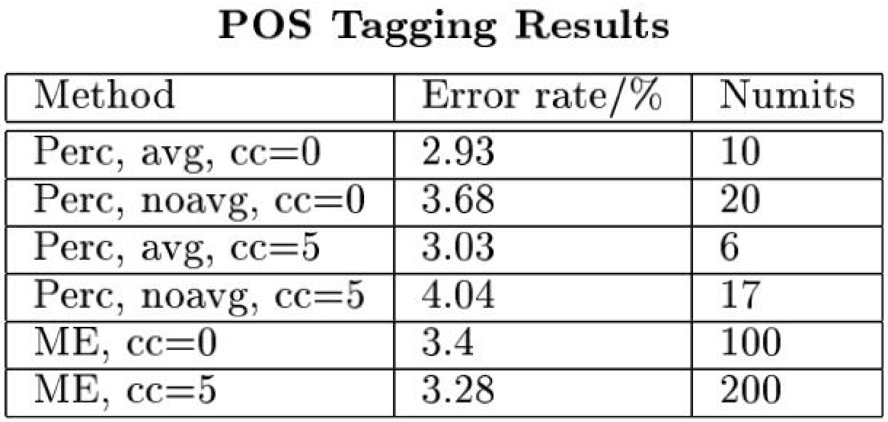
$$\frac{2}{n+1} E_{n+1}\left[\min_{\mathbf{U},\delta} \frac{(R + D_{\mathbf{U},\delta})^2}{\delta^2}\right]$$

where $E_{n+1}[]$ *is an expected value taken over* $n+1$ *examples,* R *and* $D_{\mathbf{U},\delta}$ *are as defined above, and the min is taken over* $\delta > 0$, $||\mathbf{U}|| = 1$.

Experiment Results



Current word	w_i	& t_i
Previous word	w_{i-1}	& t_i
Word two back	w_{i-2}	& t_i
Next word	w_{i+1}	& t_i
Word two ahead	w_{i+2}	& t_i
Bigram features	w_{i-2}, w_{i-1}	& t_i
	w_{i-1}, w_i	& t_i
	w_i, w_{i+1}	& t_i
	w_{i+1}, w_{i+2}	& t_i
Current tag	p_i	& t_i
Previous tag	p_{i-1}	& t_i
Tag two back	p_{i-2}	& t_i
Next tag	p_{i+1}	& t_i
Tag two ahead	p_{i+2}	& t_i
Bigram tag features	p_{i-2}, p_{i-1}	& t_i
	p_{i-1}, p_i	& t_i
	p_i, p_{i+1}	& t_i
	p_{i+1}, p_{i+2}	& t_i
Trigram tag features	p_{i-2}, p_{i-1}, p_i	& t_i
	p_{i-1}, p_i, p_{i+1}	& t_i
	p_i, p_{i+1}, p_{i+2}	& t_i



POS Tagging Results

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 = feature count cut-offs set

Conclusion

- Described new structured perceptron algorithm for tagging.
- The generic algorithm and the theorems describing its convergence could apply to several other models in the NLP.

Distributed Training Strategies for the Structured Perceptron

Ryan McDonald Keith Hall Gideon Mann

Outline

1. Introduction of distributed training strategies for the structured perceptron
2. Background (Related Work and Structured perceptron recap)
3. Iterative Parameter mixing distributed structured perceptron
4. Experiments and results
    a. Serial (All Data)
    b. Serial (Sub Sampling)
    c. Parallel (Parameter Mix)
    d. Parallel (Iterative Parameter Mix)

Limitation of Structured Perceptron

- Increasing size of available training sets

- Training complexity is proportional to inference, which is frequently non-linear in sequence length, even with strong structural independence

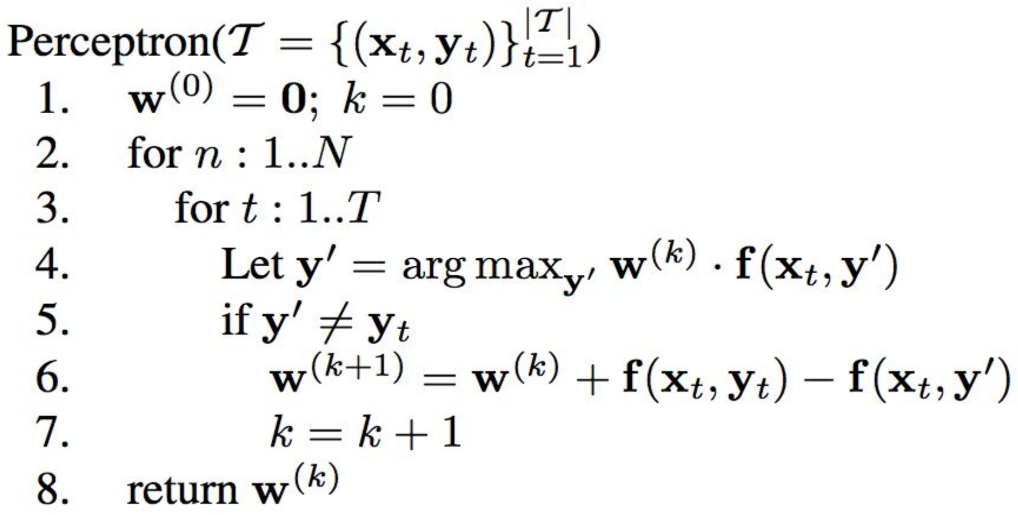
Distributed Training Strategies

- Iterative parameter mixing
  - have similar convergence properties to the standard perceptron algorithm
  - find a separating hyperplane if the training set is separable
  - reduce training times significantly
  - produce models with comparable (or superior) accuracies to those trained serially on all the data

Related Work

- Sub-gradient distributed training: asynchronous optimization via gradient descent
  - Require shared memory
  - Less suitable to the more common cluster computing environment
- Other structured prediction classifiers: like CRF
  - Conditional random fields (CRFs)
  - Distributed through the gradient
  - Require computation of a partition, expensive and sometimes intractable

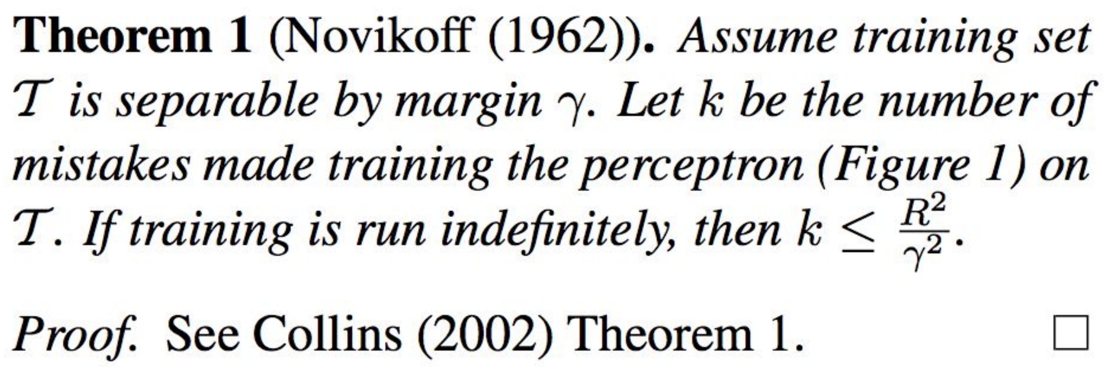
Structured Perceptron Recap



```
Perceptron($\mathcal{T} = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{|\mathcal{T}|}$)
1.    $\mathbf{w}^{(0)} = \mathbf{0};\ k = 0$
2.    for $n$ : 1..$N$
3.        for $t$ : 1..$T$
4.            Let $\mathbf{y}' = \arg\max_{\mathbf{y}'} \mathbf{w}^{(k)} \cdot \mathbf{f}(\mathbf{x}_t, \mathbf{y}')$
5.            if $\mathbf{y}' \neq \mathbf{y}_t$
6.                $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{f}(\mathbf{x}_t, \mathbf{y}')$
7.                $k = k + 1$
8.    return $\mathbf{w}^{(k)}$
```

Figure 1: The perceptron algorithm.

Supporting Theorem 1 Recap



Theorem 1 (Novikoff (1962)). *Assume training set $\mathcal{T}$ is separable by margin γ. Let k be the number of mistakes made training the perceptron (Figure 1) on $\mathcal{T}$. If training is run indefinitely, then $k \le \frac{R^2}{\gamma^2}$.*

Proof. See Collins (2002) Theorem 1. $\square$

Theorem 1 implies that if T is separable then

- The perceptron will converge in a finite amount of time
- It will produce a w that separates T.

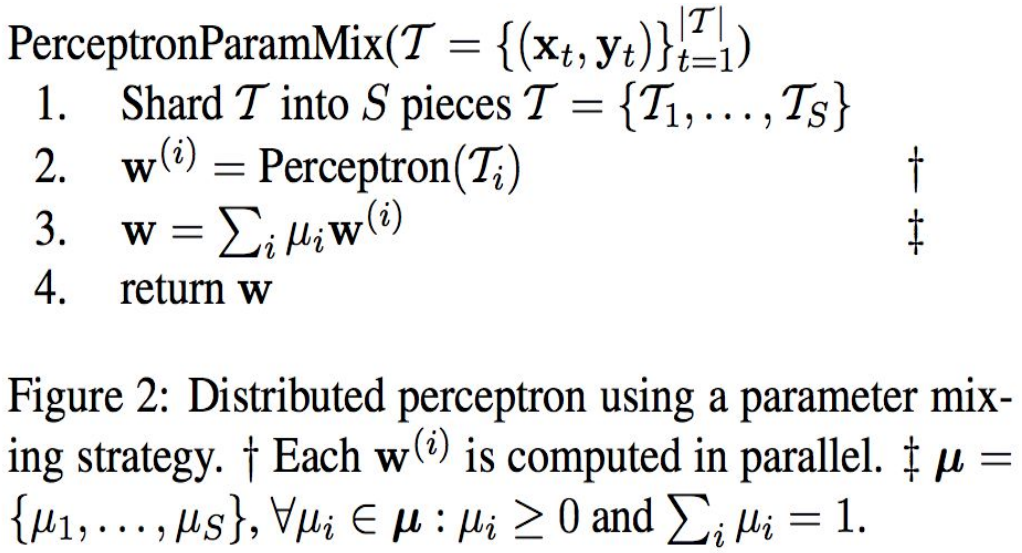
Distributed Training Strategies for the Perceptron

- Parameter Mixing

- Iterative Parameter Mixing

Parameter Mixing

- Algorithm



```
PerceptronParamMix($\mathcal{T} = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{|\mathcal{T}|}$)
  1.  Shard $\mathcal{T}$ into $S$ pieces $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_S\}$
  2.  $\mathbf{w}^{(i)} = \text{Perceptron}(\mathcal{T}_i)$                    †
  3.  $\mathbf{w} = \sum_i \mu_i \mathbf{w}^{(i)}$                           ‡
  4.  return $\mathbf{w}$
```

Figure 2: Distributed perceptron using a parameter mixing strategy. † Each $\mathbf{w}^{(i)}$ is computed in parallel. ‡ $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_S\}, \forall \mu_i \in \boldsymbol{\mu} : \mu_i \geq 0$ and $\sum_i \mu_i = 1$.

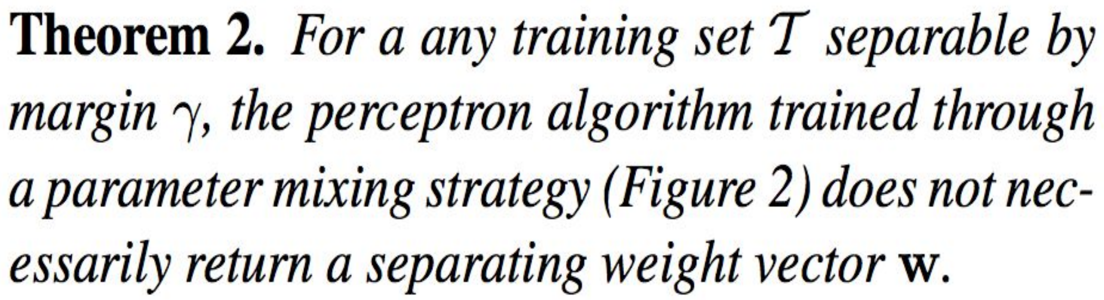
Parameter Mixing

- Parameter Mixing Map-Reduce framework
  - Map stage - Trains the individual models in parallel
  - Reduce stage - Mixes their parameters

Parameter Mixing

- Limitations
    - their analysis requires a stability bound on the parameters of a regularized maximum entropy model, which is not known to hold for the perceptron.

Supporting Theorem 2



Theorem 2. *For a any training set $\mathcal{T}$ separable by margin γ, the perceptron algorithm trained through a parameter mixing strategy (Figure 2) does not necessarily return a separating weight vector* $\mathbf{w}$.

Theorem 2 Proof

Consider a binary classification setting where Y = {0, 1} and T has 4 instance.

$T_1 = \{(x_{1,1}, y_{1,1}),(x_{1,2}, y_{1,2})\}$ $T_2 = \{(x_{2,1}, y_{2,1}),(x_{2,2}, y_{2,2})\}$

$y_{1,1} = y_{1,2} = 0$ $y_{2,1} = y_{2,2} = 1$

$f(x_{1,1}, 0) = [1\ 1\ 0\ 0\ 0\ 0]$ $f(x_{1,1}, 1) = [0\ 0\ 0\ 1\ 1\ 0]$

$f(x_{1,2}, 0) = [0\ 0\ 1\ 0\ 0\ 0]$ $f(x_{1,2}, 1) = [0\ 0\ 0\ 0\ 0\ 1]$

$f(x_{2,1}, 0) = [0\ 1\ 1\ 0\ 0\ 0]$ $f(x_{2,1}, 1) = [0\ 0\ 0\ 0\ 1\ 1]$

$f(x_{2,2}, 0) = [1\ 0\ 0\ 0\ 0\ 0]$ $f(x_{2,2}, 1) = [0\ 0\ 0\ 1\ 0\ 0]$

Theorem 2 Proof cont.

Assuming label is tie-breaking, parameter mixing might returns:
$w^1=[1\ 1\ 0\ -1\ -1\ 0]$ $w^2=[0\ 1\ 1\ 0\ -1\ -1]$

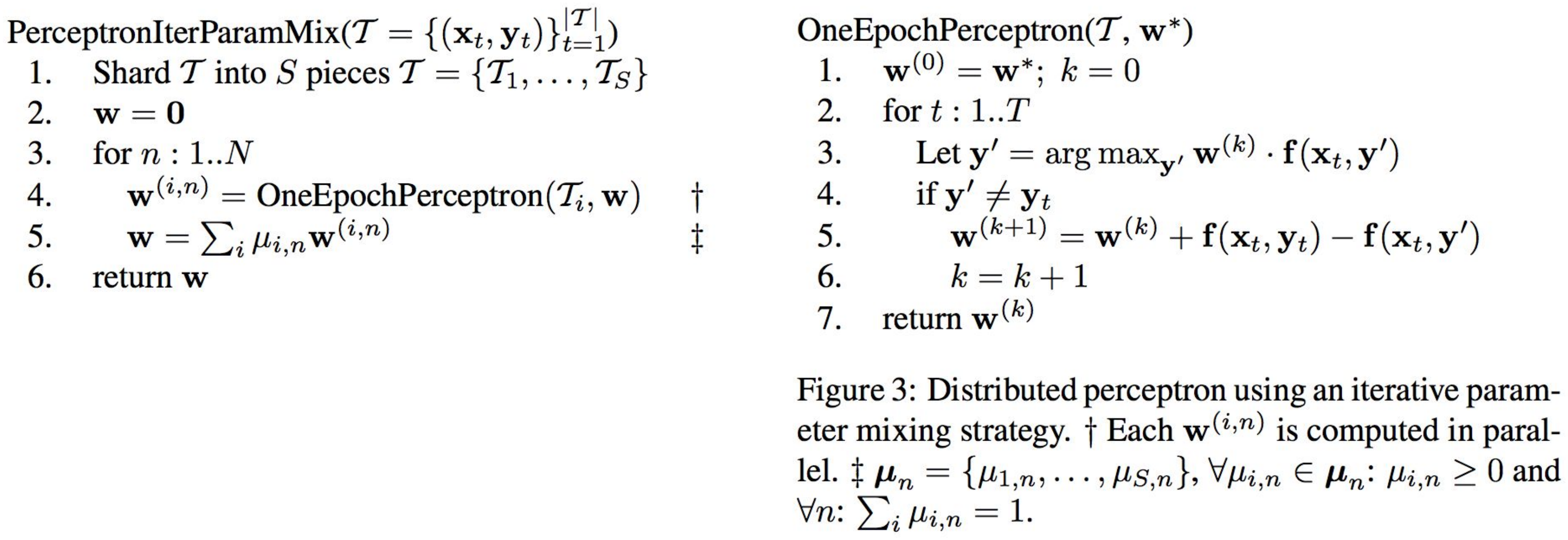
If both μ_1, μ_2 are non-zero: all examples will be classified as 0

If $\mu_1=1$ and $\mu_2=0$: $(x_{2,2}, y_{2,2})$ will be incorrectly classified as 0

If $\mu_1=0$ and $\mu_2=1$: $(x_{1,2}, y_{1,2})$ will be incorrectly classified as 0

There is a separating vector $w = [-1\ 2\ -1\ 1\ -2\ 1]$!!!

Iterative Parameter Mixing



```
PerceptronIterParamMix($\mathcal{T} = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{|\mathcal{T}|}$)
1.  Shard $\mathcal{T}$ into $S$ pieces $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_S\}$
2.  $\mathbf{w} = \mathbf{0}$
3.  for $n : 1..N$
4.      $\mathbf{w}^{(i,n)} = \text{OneEpochPerceptron}(\mathcal{T}_i, \mathbf{w})$      †
5.      $\mathbf{w} = \sum_i \mu_{i,n} \mathbf{w}^{(i,n)}$      ‡
6.  return $\mathbf{w}$
```

```
OneEpochPerceptron($\mathcal{T}$, $\mathbf{w}^*$)
1.  $\mathbf{w}^{(0)} = \mathbf{w}^*$; $k = 0$
2.  for $t : 1..T$
3.      Let $\mathbf{y}' = \arg\max_{\mathbf{y}'} \mathbf{w}^{(k)} \cdot \mathbf{f}(\mathbf{x}_t, \mathbf{y}')$
4.      if $\mathbf{y}' \neq \mathbf{y}_t$
5.          $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{f}(\mathbf{x}_t, \mathbf{y}')$
6.          $k = k + 1$
7.  return $\mathbf{w}^{(k)}$
```

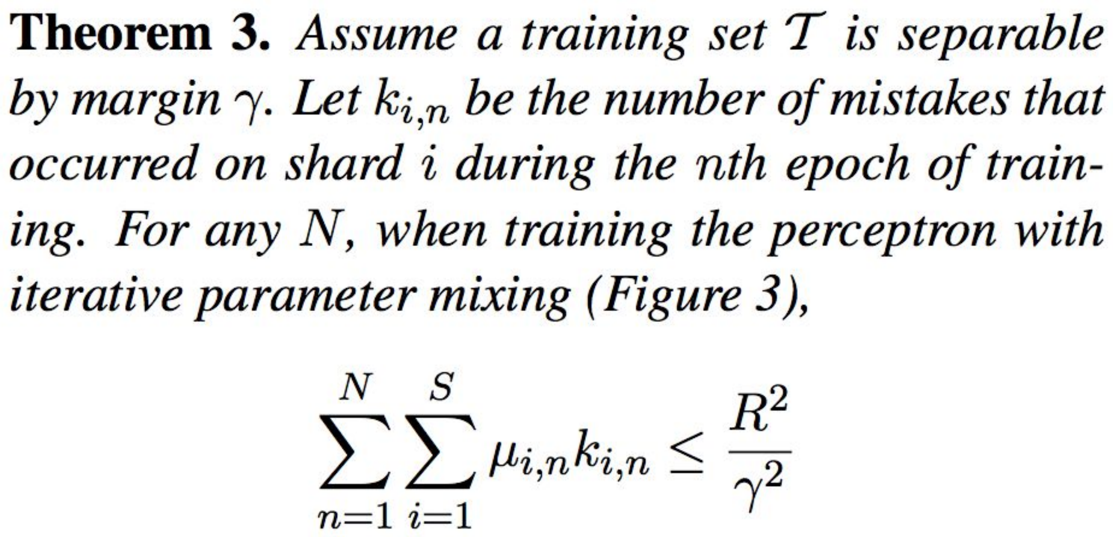
Figure 3: Distributed perceptron using an iterative parameter mixing strategy. † Each $\mathbf{w}^{(i,n)}$ is computed in parallel. ‡ $\boldsymbol{\mu}_n = \{\mu_{1,n}, \ldots, \mu_{S,n}\}$, $\forall \mu_{i,n} \in \boldsymbol{\mu}_n$: $\mu_{i,n} \geq 0$ and $\forall n$: $\sum_i \mu_{i,n} = 1$.

Iterative Parameter Mixing

- Iterative Parameter Mixing Map-reduce
  - map computes the parameters for each shard for one epoch and
  - reduce mixes and re-sends them

- The disadvantage of iterative parameter mixing, relative to simple parameter mixing, is that the amount of information sent across the network will increase.

Supporting Theorem 3



Theorem 3. *Assume a training set $\mathcal{T}$ is separable by margin γ. Let $k_{i,n}$ be the number of mistakes that occurred on shard i during the nth epoch of training. For any N, when training the perceptron with iterative parameter mixing (Figure 3),*

$$\sum_{n=1}^{N}\sum_{i=1}^{S}\mu_{i,n}k_{i,n} \leq \frac{R^2}{\gamma^2}$$

Theorem 3 Proof

$w^{(i,n)}$ the weight vector for the ith shard after the nth epoch of the main loop

$w^{([i,n]-k)}$ the weight vector that existed on shard i in the nth epoch k errors before $w^{(i,n)}$

$w^{(avg,n)}$ the mixed vector from the weight vectors returned after the nth epoch

$$\mathbf{w}^{(\text{avg},n)} = \sum_{i=1}^{S} \mu_{i,n} \mathbf{w}^{(i,n)}$$

Theorem 3 Proof cont.

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{f}(\mathbf{x}_t, \mathbf{y}')$$

$$\begin{aligned}
\mathbf{u} \cdot \mathbf{w}^{(i,n)} &= \mathbf{u} \cdot \mathbf{w}^{([i,n]-1)} \\
&\quad + \mathbf{u} \cdot (\mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{f}(\mathbf{x}_t, \mathbf{y}')) \\
&\geq \mathbf{u} \cdot \mathbf{w}^{([i,n]-1)} + \gamma \\
&\geq \mathbf{u} \cdot \mathbf{w}^{([i,n]-2)} + 2\gamma \\
\ldots &\geq \mathbf{u} \cdot \mathbf{w}^{(\text{avg},n-1)} + k_{i,n}\gamma \quad \text{(A1)}
\end{aligned}$$

Theorem 3 Proof cont.

$$\begin{aligned}
\|\mathbf{w}^{(i,n)}\|^2 &= \|\mathbf{w}^{([i,n]-1)}\|^2 \\
&\quad + \|\mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{f}(\mathbf{x}_t, \mathbf{y}')\|^2 \\
&\quad + 2\mathbf{w}^{([i,n]-1)}(\mathbf{f}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{f}(\mathbf{x}_t, \mathbf{y}')) \\
&\leq \|\mathbf{w}^{([i,n]-1)}\|^2 + R^2 \\
&\leq \|\mathbf{w}^{([i,n]-2)}\|^2 + 2R^2 \\
\ldots &\leq \|\mathbf{w}^{(\mathrm{avg},n-1)}\|^2 + k_{i,n}R^2 \quad \text{(A2)}
\end{aligned}$$

Theorem 3 Proof cont.

Base Case

$$\mathbf{u} \cdot \mathbf{w}^{\text{avg},1} = \sum_{i=1}^{S} \mu_{i,1} \mathbf{u} \cdot \mathbf{w}^{(i,1)} \geq \sum_{i=1}^{S} \mu_{i,1} k_{i,1} \gamma$$

$$\begin{aligned} \|\mathbf{w}^{(\text{avg},1)}\|^2 &= \left\| \sum_{i=1}^{S} \mu_{i,1} \mathbf{w}^{(i,1)} \right\|^2 \\ &\leq \sum_{i=1}^{S} \mu_{i,1} \|\mathbf{w}^{(i,1)}\|^2 \leq \sum_{i=1}^{S} \mu_{i,1} k_{i,1} R^2 \end{aligned}$$

Inductive Hypotheses

$$\mathbf{u} \cdot \mathbf{w}^{(\text{avg},N)} \geq \sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} \gamma \qquad \text{(IH1)}$$

$$\|\mathbf{w}^{(\text{avg},N)}\|^2 \leq \sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} R^2 \qquad \text{(IH2)}$$

Theorem 3 Proof cont.

$$\begin{aligned}
\mathbf{u} \cdot \mathbf{w}^{(\text{avg},N)} &= \sum_{i=1}^{S} \mu_{i,N} (\mathbf{u} \cdot \mathbf{w}^{(i,N)}) \\
&\geq \sum_{i=1}^{S} \mu_{i,N} (\mathbf{u} \cdot \mathbf{w}^{(\text{avg},N-1)} + k_{i,N}\gamma) \\
&= \mathbf{u} \cdot \mathbf{w}^{(\text{avg},N-1)} + \sum_{i=1}^{S} \mu_{i,N} k_{i,N} \gamma \\
&\geq \left[\sum_{n=1}^{N-1} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} \gamma \right] + \sum_{i=1}^{S} \mu_{i,N} k_{i,N} \\
&= \sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} \gamma
\end{aligned}$$

$$\begin{aligned}
\|\mathbf{w}^{(\text{avg},N)}\|^2 &\leq \sum_{i=1}^{S} \mu_{i,N} \|\mathbf{w}^{(i,N)}\|^2 \\
&\leq \sum_{i=1}^{S} \mu_{i,N} (\|\mathbf{w}^{(\text{avg},N-1)}\|^2 + k_{i,N} R^2) \\
&= \|\mathbf{w}^{(\text{avg},N-1)}\|^2 + \sum_{i=1}^{S} \mu_{i,N} k_{i,N} R^2 \\
&\leq \left[\sum_{n=1}^{N-1} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} R^2 \right] + \sum_{i=1}^{S} \mu_{i,N} k_{i,N} R^2 \\
&= \sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n} R^2
\end{aligned}$$

Theorem 3 Proof cont.

$$\left[\sum_{n=1}^{N}\sum_{i=1}^{S}\mu_{i,n}k_{i,n}\right]^2 \gamma^2 \;\leq\; \left[\sum_{n=1}^{N}\sum_{i=1}^{S}\mu_{i,n}k_{i,n}\right] R^2$$

Analysis

If we set each $\mu_{i,n}$ to be the uniform mixture, $\mu_{i,\,n}= 1/S$, theorem 3 guarantees convergence to a separating hyperplane.

If $\sum_{i=1}^{S} \mu_{i,n} k_{i,n} = 0$ then the previous weight vector already separated the data.

Otherwise $\sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n} k_{i,n}$ is bounded and cannot increase indefinitely.

Analysis

Non-distributed: $N_{\text{non\_dist}} \leq R^2/\gamma^2$

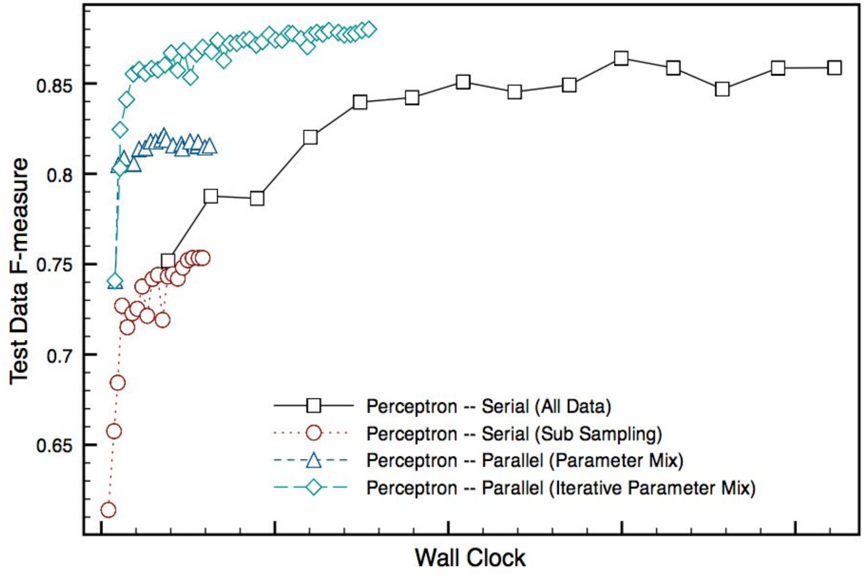
Distributed: Setting $\mu_{i,n} = k_{i,n}/k_n$ where $k_n = \sum_i k_{i,n}$

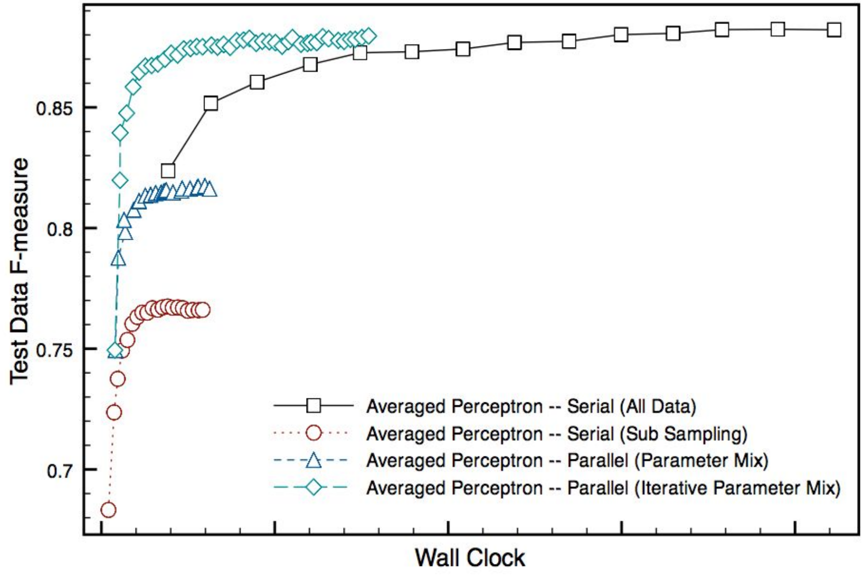
$$N_{\text{dist}} \leq \sum_{n=1}^{N_{\text{dist}}} \prod_{i=1}^{S} [k_{i,n}]^{\frac{k_{i,n}}{k_n}} \leq \sum_{n=1}^{N_{\text{dist}}} \sum_{i=1}^{S} \frac{k_{i,n}}{k_n} k_{i,n} \leq \frac{R^2}{\gamma^2}$$

Experiments

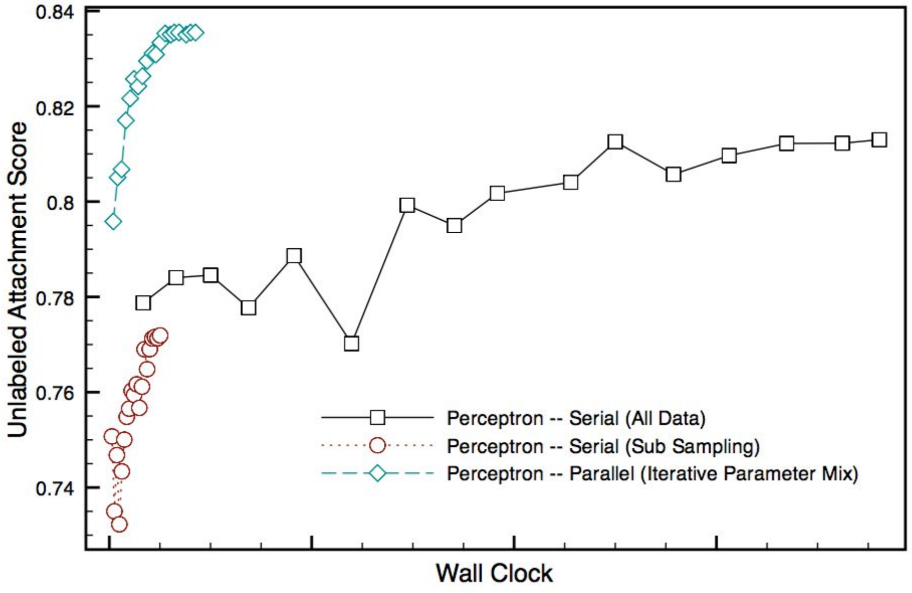
- Serial (All Data)
  - Trained serially on all the available data
- Serial (Sub Sampling)
  - Shard the data, select randomly and train serially
- Parallel (Parameter Mix)
  - Parameter Mixing distributed strategy
- Parallel (Iterative Parameter Mix)
  - Iterative Parameter Mixing distributed strategy

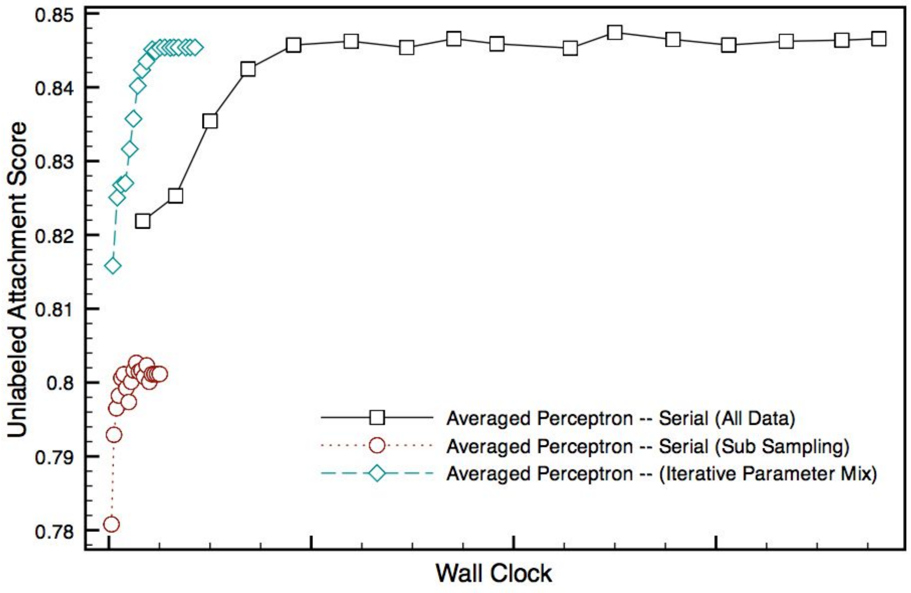
Experiment 1





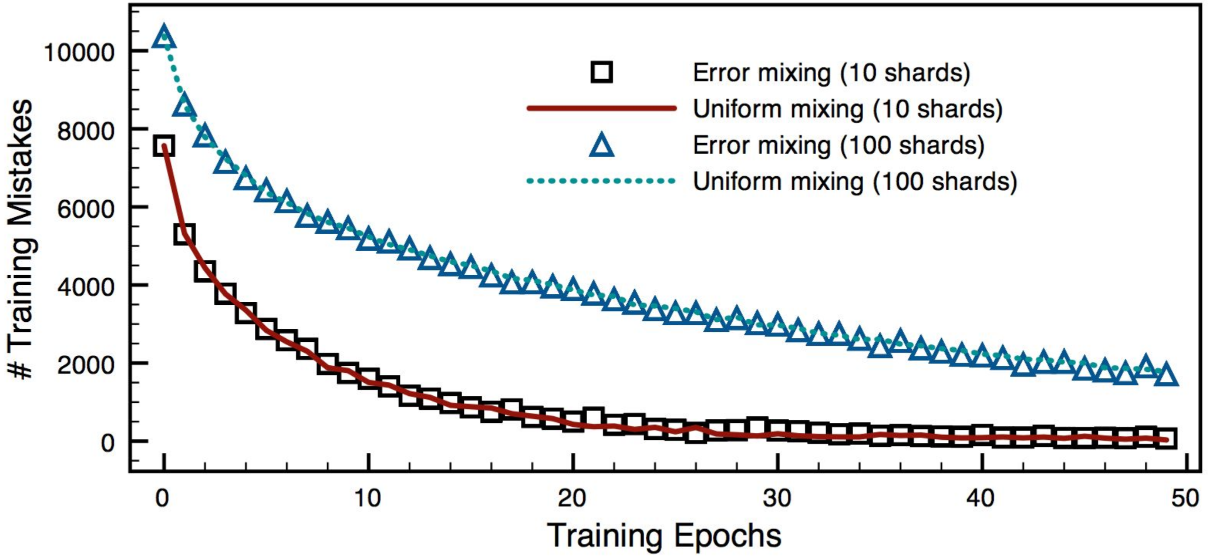
Experiment 2





Convergence Properties

$$\sum_{n=1}^{N}\sum_{i=1}^{S}\frac{k_{i,n}}{S} \leq \frac{R^2}{\gamma^2} \quad \Longrightarrow \quad \sum_{n=1}^{N}\sum_{i=1}^{S}k_{i,n} \leq S \times \frac{R^2}{\gamma^2}$$



Conclusions

- The analysis shows that an iterative parameter mixing strategy is both guaranteed to separate the data (if possible) and significantly reduces the time required to train high accuracy classifiers.

- There is a trade-off between increasing training times through distributed computation and slower convergence relative to the number of shards.