Conditional Random Fields for Sequential Supervised Learning

Thomas G. Dietterich

Adam Ashenfelter

Department of Computer Science

Oregon State University

Corvallis, Oregon 97331

http://www.eecs.oregonstate.edu/~tgd

Supported by NSF IIS-0083292

Outline

- Goal: Off-the-Shelf Sequential Supervised Learning
- Candidate Methods
- Training CRFs by Gradient Boosting
- Concluding Remarks

Many Application Problems Require Sequential Learning

- Part-of-speech Tagging
- Information Extraction from the Web
- Text-to-Speech Mapping

Part-of-Speech Tagging

- Given an English sentence, can we assign a part of speech to each word?

- “Do you want fries with that?”
- <verb pron verb noun prep pron>

Information Extraction from the Web

<dl><dt><b>Srinivasan Seshan</b> (Carnegie Mellon University) <dt><a href=…><i>Making Virtual Worlds Real</i></a><dt>Tuesday, June 4, 2002<dd>2:00 PM , 322 Sieg<dd>Research Seminar

* * * name name * * affiliation affiliation affiliation * * * * title title title title * * * date date date date * time time * location location * event-type event-type

Text-to-Speech Mapping

- “photograph” => `/f-Ot@graf-/`

Sequential Supervised Learning (SSL)

- Given: A set of training examples of the form ($\mathbf{X}_i$,$\mathbf{Y}_i$), where

  $\mathbf{X}_i = \langle x_{i,1}, \ldots , x_{i,T_i} \rangle$ and

  $\mathbf{Y}_i = \langle y_{i,1}, \ldots , y_{i,T_i} \rangle$ are sequences of length T_i
- Find: A function f for predicting new sequences: $\mathbf{Y} = f(\mathbf{X})$.

Examples as Sequential Supervised Learning

Domain	Input $\mathbf{X}_i$	Output $\mathbf{Y}_i$
Part-of-speech Tagging	sequence of words	sequence of parts of speech
Information Extraction	sequence of tokens	sequence of field labels {name, …}
Test-to-speech Mapping	sequence of letters	sequence phonemes

Goal: Off-the-Shelf Learning Methods for SSL

- No existing machine learning, data mining, and statistical packages supports SSL
- No existing method meets all of the requirements needed for an “off-the-shelf” method

Requirements for Off-the-Shelf Methods

- Accuracy
  - Model both x ′ y and y_t ′ y_{t+1} relationships
  - Support rich X ′ y_t features
  - Avoid label bias problem
- Computational efficiency
- Easy to use
  - No parameter tuning required

Two Kinds of Relationships

$$\begin{tikzcd} y_1 \arrow[r, leftrightarrow] \arrow[d, leftrightarrow] & y_2 \arrow[r, leftrightarrow] \arrow[d, leftrightarrow] & y_3 \arrow[d, leftrightarrow] \\ x_1 & x_2 & x_3 \end{tikzcd}$$

- “Vertical” relationship between the x_t’s and y_t’s
  - Example: “Friday” is usually a “date”
- “Horizontal” relationships among the y_t’s
  - Example: “name” is usually followed by “affiliation”
- SSL can (and should) exploit both kinds of information

Example of y ′ y relationships

- Consider the text-to-speech problem:
  - "photograph" => /f-Ot@graf-/
  - "photography" =>/f-@tAgr@f-i/
- The letter "y" changes the pronunciation of all vowels in the word!
- x ′ y relationships are not sufficient:
  - "o" is pronounced as /O/, /@/, and /A/
  - need context to tell which is correct

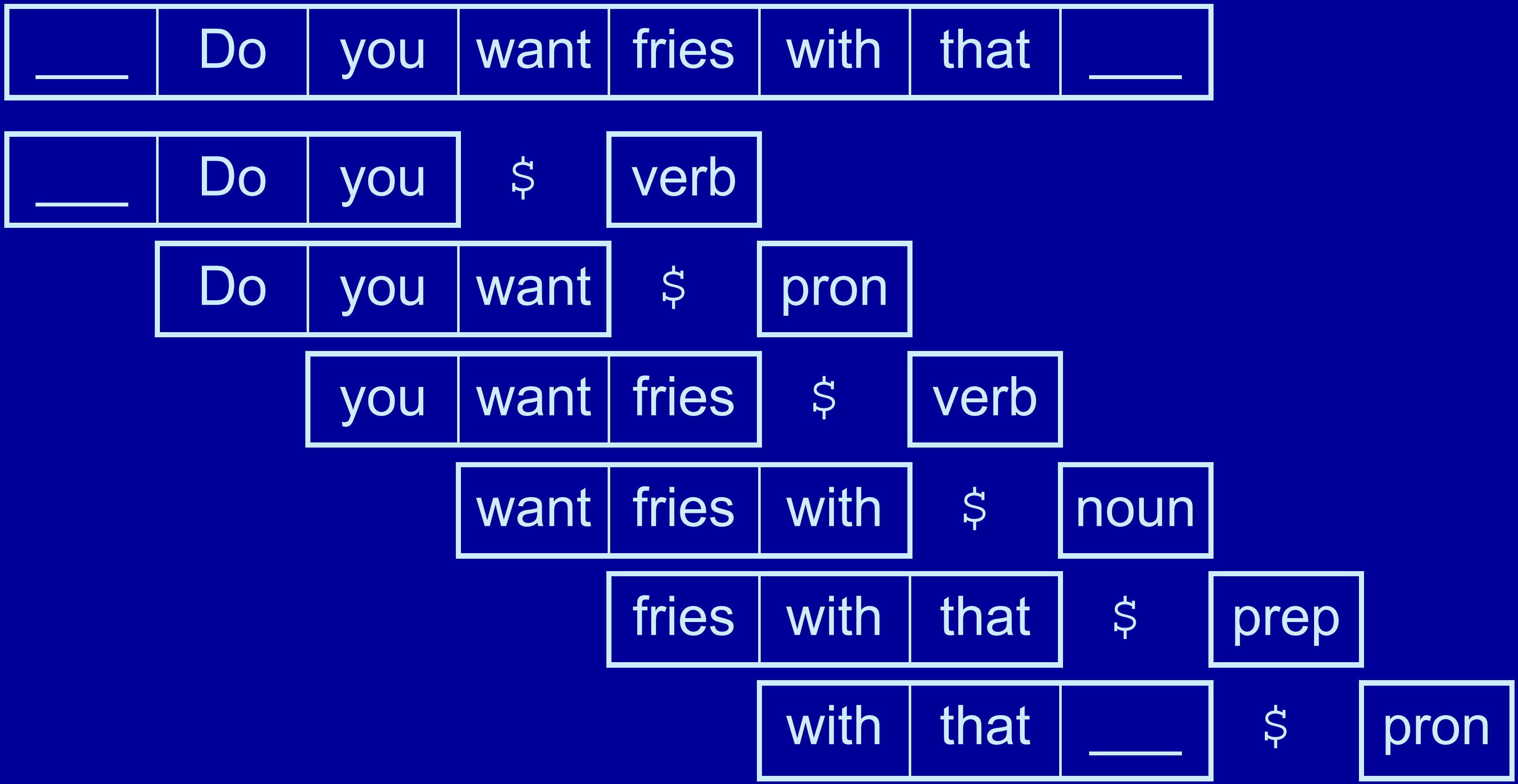
Rich X ′ y Relationships

- Generative models such as HMMs model each x_t as being generated by a single y_t
- Can’t incorporate the context around x_t
  - Example: Decide how to pronounce “h” based on surrounding letters “th”, “ph”, “sh”, “ch”.
- Can’t include global features
  - Example: “Sentence begins with question word”

Existing Methods

- Sliding windows
- Recurrent sliding windows
- Hidden Markov models
- Maximum entropy Markov models
- Input/Output Markov models
- Conditional Random Fields
- Maximum Margin Markov Networks

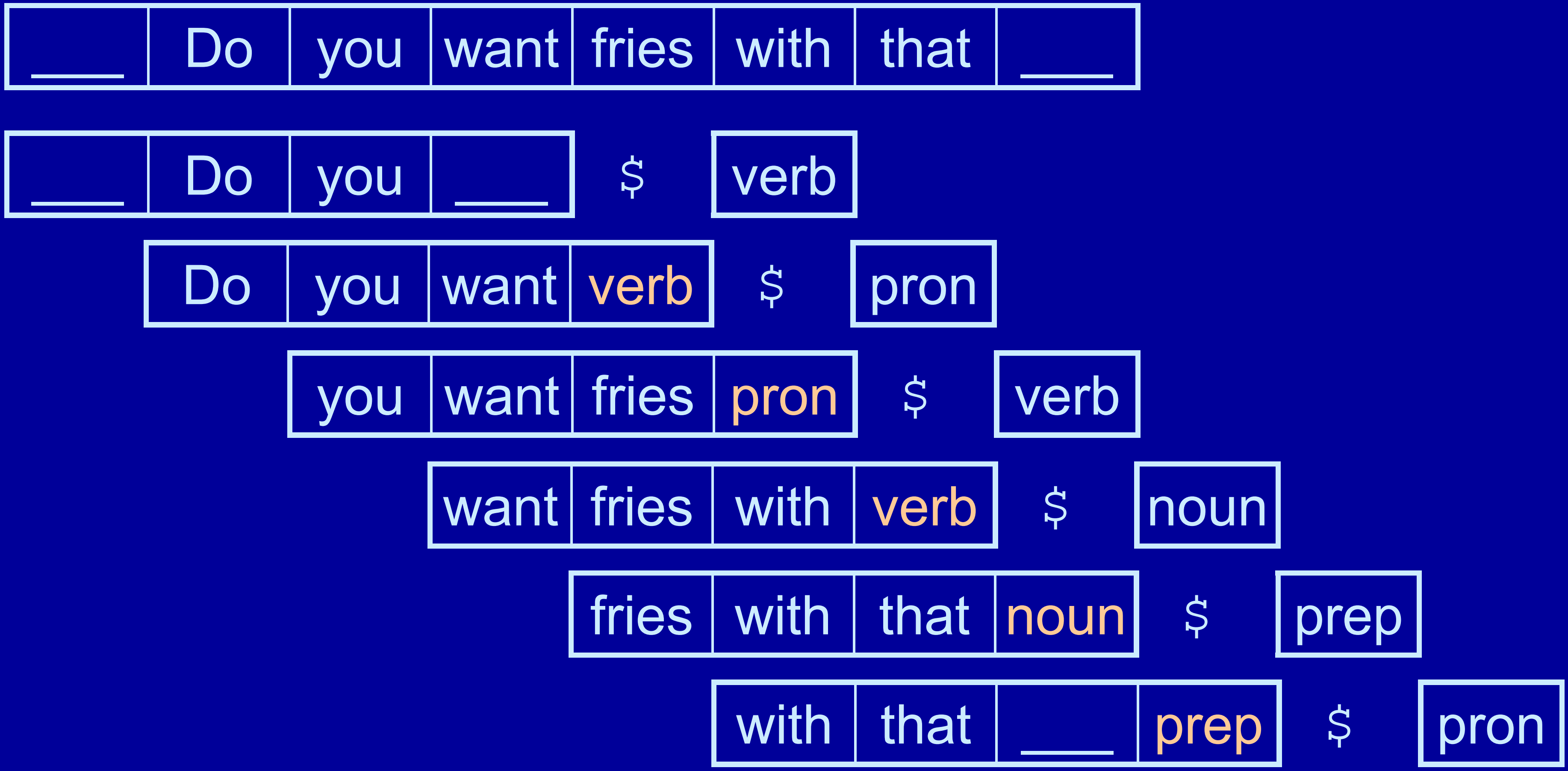
Sliding Windows



Properties of Sliding Windows

- Converts SSL to ordinary supervised learning
- Only captures the relationship between (part of) X and y_t. Does not explicitly model relations among the y_t's
- Assumes each window is independent

Recurrent Sliding Windows


___ Do you want fries with that ___
___ Do you ___ $ verb
Do you want verb $ pron
you want fries pron $ verb
want fries with verb $ noun
fries with that noun $ prep
with that ___ prep $ pron

Recurrent Sliding Windows

- Key Idea: Include y_t as input feature when computing y_{t+1}.
- During training:
  - Use the correct value of y_t
  - Or train iteratively (especially recurrent neural networks)
- During evaluation:
  - Use the predicted value of y_t

Properties of Recurrent Sliding Windows

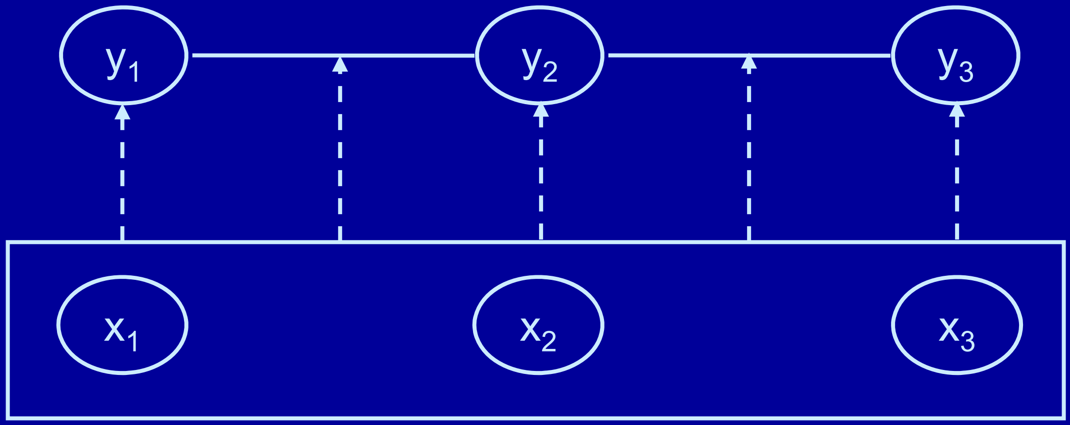
- Captures relationship among the y's, but only in one direction!
- Results on text-to-speech:

Method	Direction	Words	Letters
sliding window	none	12.5%	69.6%
recurrent s. w.	left-right	17.0%	67.9%
recurrent s. w.	right-left	24.4%	74.2%

Evaluation of Methods

Issue	SW	RSW	HMM	MEMM	IOHMM	CRF
x_t ′ y_t y_t ′ y_{t+1}	NO	Partly	YES	YES	YES	YES
X ′ y_t rich?	YES	YES	NO	YES	YES	YES
label bias ok?	YES	YES	YES	NO	NO	YES
efficient?	YES	YES	YES	YES?	NO	NO

Conditional Random Fields



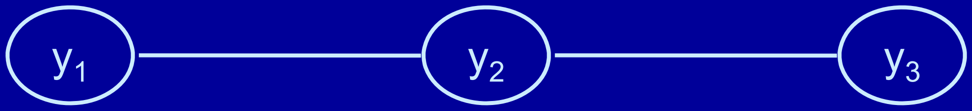
- The y_t's form a Markov Random Field conditioned on X: P(Y|X)

Lafferty, McCallum, & Pereira (2001)

Markov Random Fields

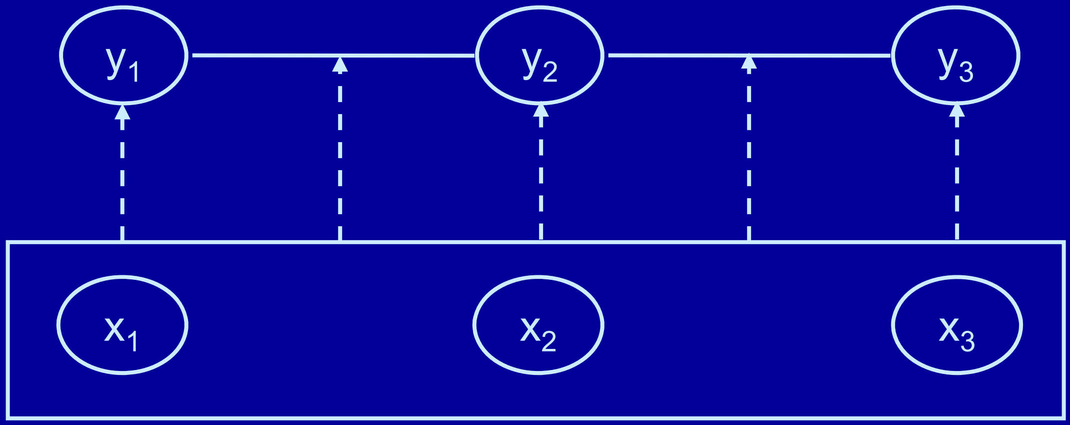
- Graph G = (V,E)
  - Each vertex v 5 V represents a random variable y_v.
  - Each edge represents a direct probabilistic dependency.
- $P(Y) = 1/Z \exp [\sum_c \Psi_c(c(Y))]$
  - c indexes the cliques in the graph
  - Ψ_c is a potential function
  - c(Y) selects the random variables participating in clique c.

A Simple MRF



- Cliques:
  - singletons: $\{y_1\}$, $\{y_2\}$, $\{y_3\}$
  - pairs (edges); $\{y_1,y_2\}$, $\{y_2,y_3\}$
- $P(\{y_1,y_2,y_3\}) = 1/Z \exp[\Psi_1(y_1) + \Psi_2(y_2) + \Psi_3(y_3) + \Psi_{12}(y_1,y_2) + \Psi_{23}(y_2,y_3)]$

CRF Potential Functions are Conditioned on X



- $\Psi_t(y_t, X)$
- $\Psi_{t,t+1}(y_t, y_{t+1}, X)$

CRF Potentials are Log Linear Models

- $\Psi_t(y_t,X) = \sum_b \beta_b\, g_b(y_t,X)$
- $\Psi_{t,t+1}(y_t,y_{t+1},X) = \sum_a \lambda_a\, f_a(y_t,y_{t+1},X)$

- where g_b and f_a are user-defined boolean functions (“features”)
  - Example: g_{23} = [x_t = “o” and y_t = /@/]

Training CRFs

- Let $\theta = \{\beta_1, \beta_2, \ldots, \lambda_1, \lambda_2, \ldots\}$ be all of our parameters
- Let F_θ be our CRF, so $F_\theta(Y,X) = P(Y|X)$
- Define the "loss" function $L(Y,F_\theta(Y,X))$ to be the Negative Log Likelihood

  $L(Y,F_\theta(Y,X)) = -\log F_\theta(Y,X)$
- Goal: Find θ to minimize loss (maximize likelihood)

Algorithms

- Iterative Scaling
- Gradient Descent
- Functional Gradient Descent
  - Gradient “tree boosting”

Gradient Descent Search

- From calculus we know that the minimum loss will be where

$$\frac{d\, L(Y,F_{\theta}(Y,X))}{d\,\theta} = u_{\theta}\, L(Y,F_{\theta}(Y,X)) = 0$$

- Method:

$$\theta := \theta - \eta\, u_{\theta}\, L(Y,F_{\theta}(Y,X))$$

Gradient Descent with Set of Training Examples

- We have N training examples (X_i, Y_i)
- Negative log likelihood of all N examples is the sum of the neg log likelihoods of each example
- The gradient of the negative log likelihood is the sum of the gradients of the neg log likelihoods of each example.

Gradients from Each Example

example	gradient
(X_1,Y_1)	$u_{\theta}\, L(Y_1,F_{\theta}(Y_1,X_1))$
(X_2,Y_2)	$u_{\theta}\, L(Y_2,F_{\theta}(Y_2,X_2))$
(X_3,Y_3)	$u_{\theta}\, L(Y_3,F_{\theta}(Y_3,X_3))$
(X_4,Y_4)	$u_{\theta}\, L(Y_4,F_{\theta}(Y_4,X_4))$

$$\theta := \theta - \eta \sum_i u_{\theta}\, L(Y_i, F_{\theta}(Y_i,X_i))$$

Problem: Gradient Descent is Very Slow

- Lafferty et al. employed modified iterative scaling but reported that it was very slow.
- We (and others) implemented conjugate gradient search, which is faster, but not fast enough
- For text-to-speech: 16 parallel processors, 40 hours per line search.

Functional Gradient Descent (Breiman, et al.)

- Standard gradient descent:

$\theta_{final} = \theta_0 + \delta_1 + \delta_2 + \ldots + \delta_M$

where $\delta_m = -\eta \; u_{\;\theta m\text{-}1} \sum_i L(Y_i, F_{\theta m\text{-}1}(Y_i, X_i))$

- Functional Gradient Descent:

$F_{final} = F_0 + \Delta_1 + \Delta_2 + \ldots + \Delta_M$

where $\Delta_m = -\eta \; h_m$, and h_m is a function that approximates $u_{\;F}\sum_i L(Y_i, F_{m\text{-}1}(Y_i, X_i))$

Functional Gradient Descent (2)

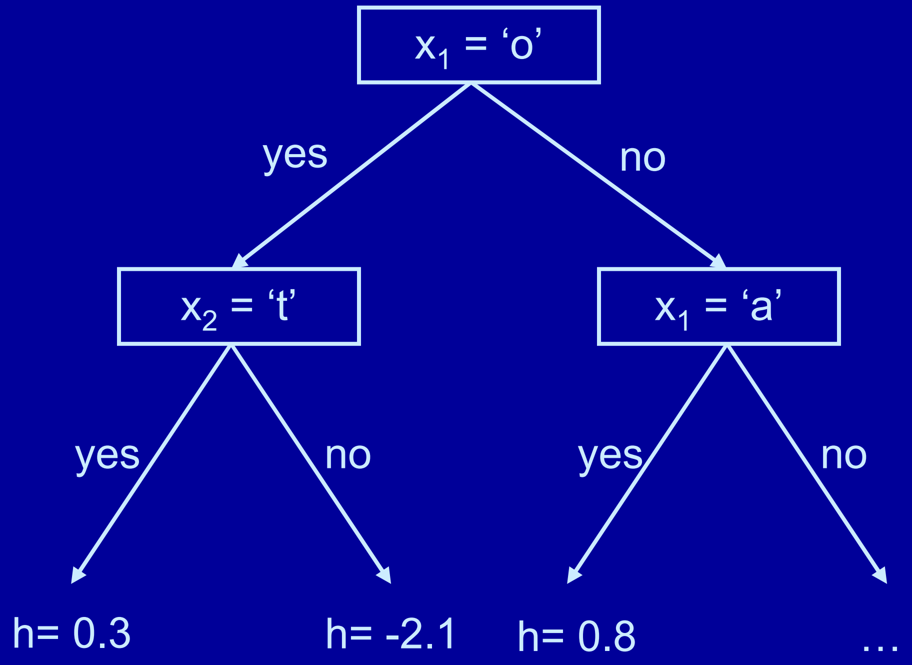
example	functional gradient	functional gradient example
(X_1,Y_1)	$u\ _F L(Y_1,F_{m-1}(Y_1,X_1)) = g_1$	(X_1,g_1)
(X_2,Y_2)	$u\ _F L(Y_2,F_{m-1}(Y_2,X_2)) = g_2$	(X_2,g_2)
(X_3,Y_3)	$u\ _F L(Y_3,F_{m-1}(Y_3,X_3)) = g_3$	(X_3,g_3)
(X_4,Y_4)	$u\ _F L(Y_4,F_{m-1}(Y_4,X_4)) = g_4$	(X_4,g_4)

Fit h to minimize $\sum_i [h(X_i) - g_i]^2$

Friedman’s Gradient Boosting Algorithm

- $F_0 = \text{argmin}_\phi \sum_i L(Y_i,\phi)$
- For m = 1, …, M do
    - $g_i := u_F\ L(Y_i,F_{m-1}(Y_i,X_i))$, $i = 1, \dots, N$
    - fit regression tree $h := \text{argmin}_f \sum_i [f(X_i) – g_i]^2$
    - $\eta_m = \text{argmin}_\phi \sum_i L(Y_i,\ F_{m-1}(Y_i,X_i) + \phi\ h(X_i))$
    - $F_m = F_{m-1} + \eta_m\ h_m$

Regression Trees



Very fast and effective algorithms

Application to CRF Training

- Recall CRF model:

  $\Psi(y_{t-1},y_t,X) = \Sigma_a \lambda_a f_a(y_{t-1},y_t,X)$

  $\Psi(y_t,X) = \Sigma_b \beta_b g_b(y_t,X)]$

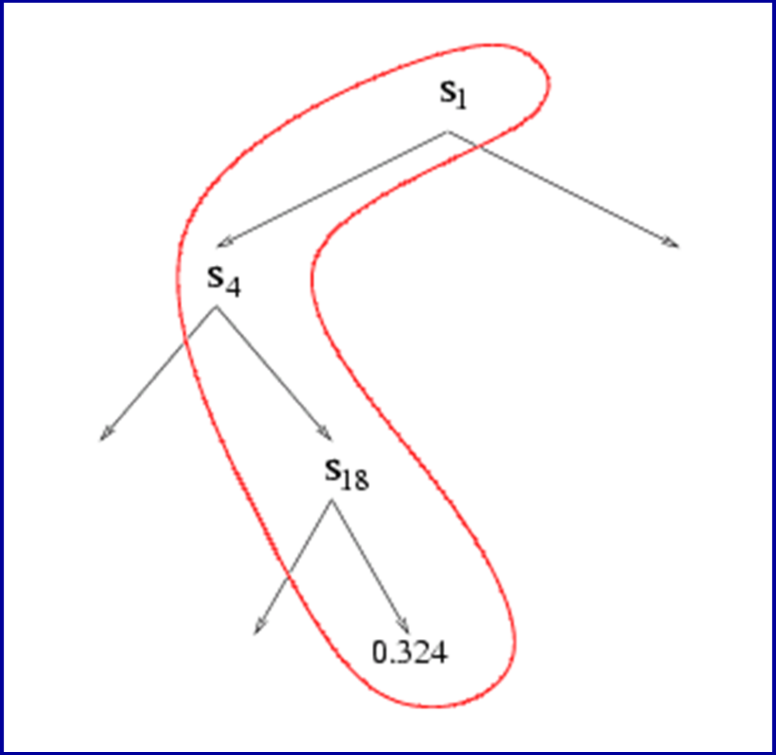
- Represent $\Psi(y_{t-1},y_t,X) + \Psi(y_t,X)$ by a set of K functions (one per class label):
  - $\Psi(\ell,k,X) + \Psi(k,X) = F^k(\ell,X)$, $k = 1, \ldots, K$
    - where $F^k(\ell,X) = \Sigma_m \eta_m h_{k,m} (\ell,X)$
    - Each $h_{k,m}$ is a regression tree that tests the features $\{f_a, g_b\}$ of the CRF
    - The values in the leaves of the tree become the weights λ_a and β_b

Sum of Regression Trees is Equivalent to CRF

Circled Path is equivalent to expression of the form $\lambda_a f_a$

$\lambda_a = 0.324$

$f_a = s_1 \& \neg s_4 \& \neg s_{18}$



Resulting CRF Model

$$P(Y|X) = 1/Z * \exp[\textstyle\sum_t F^{y_t}(y_{t-1},X)]$$

Forward-Backward Algorithm: Recursive Computation of Z

- Let

  $\alpha(k,1) = \exp F^{k}(\mathrm{B},X)$

  $\alpha(k,t) = \sum_{k'} [\exp F^{k}(k',X)]\, \alpha(k',t-1)$

  $\beta(k,T) = 1$

  $\beta(k,t) = \sum_{k'} [\exp F^{k'}(k,X)]\, \beta(k',t+1)$

- $Z = \sum_{k} \alpha(k,T) = \beta(\mathrm{B},0)$

Functional Gradient Computation

- Let $w_t(X_i)$ be the "window" of X_i used by the features at time t.
- We get one training example for each k, ℓ, i, and t:

$$g_{k,\ell,i,t} = \frac{\partial \log L(Y_i, P(Y_i \mid X_i))}{\partial F^k(\ell, w_t(X_i))}$$

- Training example for F^k:

$$(\langle \ell, w_t(X_i)\rangle, g_{k,\ell,i,t})$$

Functional Gradient Computation (2)

$$g_{k,\ell,i,t} = \frac{\partial \log L(Y_i, P(Y_i \mid X_i))}{\partial F^k(\ell, w_t(X_i))}$$

$$= \frac{\partial}{\partial F^k(\ell, w_t(X_i))} \sum_t F^{y_t}(y_{t-1}, w_t(X_i)) - \log Z$$

$$= I[y_{t-1}=\ell, y_t=k] - \frac{1}{Z} \frac{\partial}{\partial F^k(\ell, w_t(X_i))} Z$$

Functional Gradient Computation (3)

$$\frac{1}{Z}\frac{\partial}{\partial F^k(\ell, w_t(X_i))} Z =$$

$$\frac{1}{Z}\frac{\partial}{\partial F^k(\ell, w_t(X_i))} \Sigma_u \left(\Sigma_v \left[\exp F^u(v, w_t(X_i))\right] \alpha(v, t-1)\right) \beta(u,t) =$$

$$\frac{F^k(\ell, w_t(X_i)\ \alpha(\ell, t-1)\ \beta(k,t)}{Z} =$$

$$P(y_{i,t}=k,\ y_{i,t-1}=\ell \mid X_i)$$

Functional Gradient Computation (4)

$$g_{k,\ell,i,t} = \frac{\partial \log L(Y_i, P(Y_i \mid X_i))}{\partial F^k(\ell, w_t(X_i))}$$

$$= I[y_{i,t-1}=\ell,\ y_{i,t}=k] - P(y_{i,t}=k,\ y_{i,t-1}=\ell \mid X_i)$$

This is our residual on the probability scale

Training Procedure

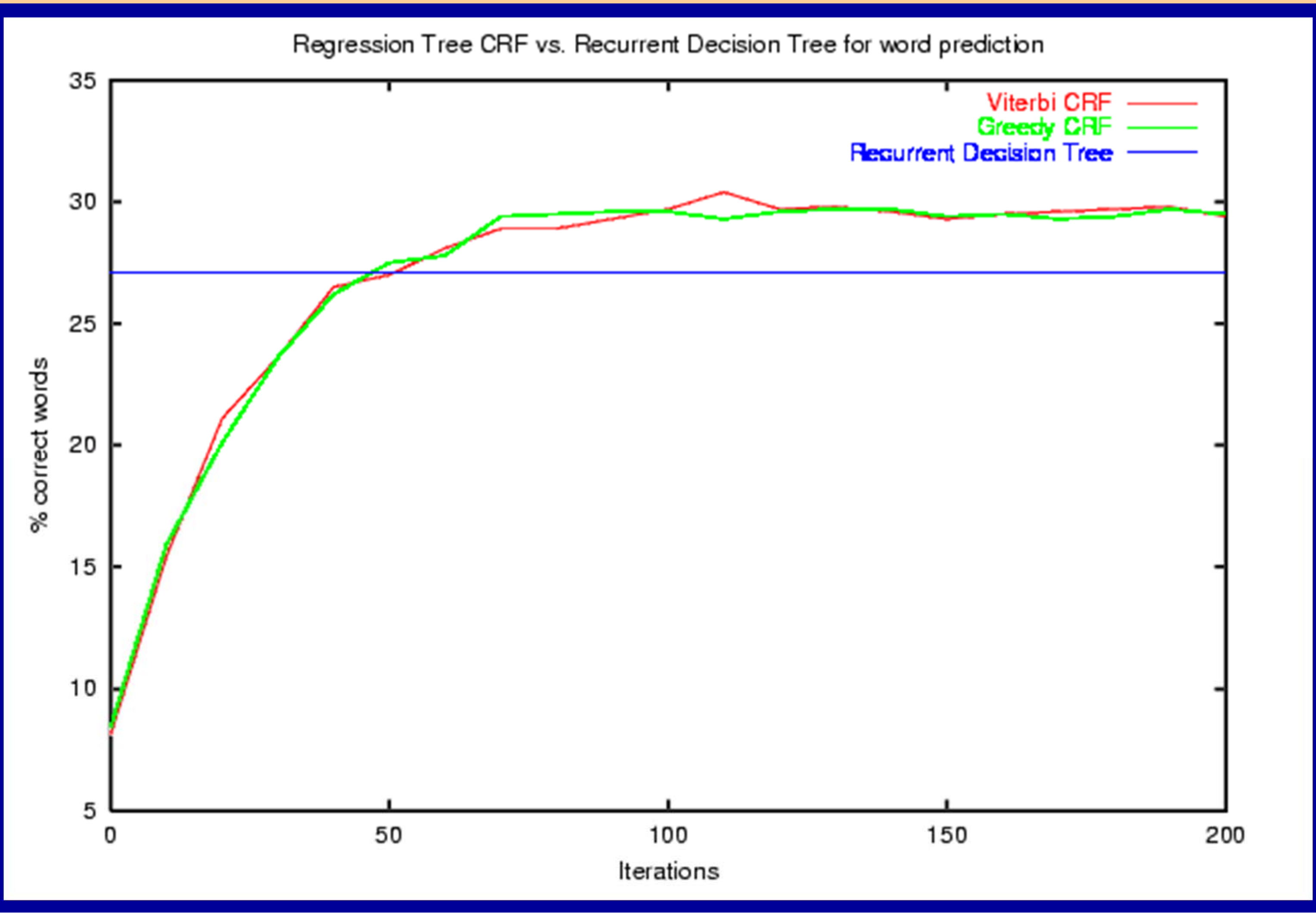
- Initialize $F^k = 0$; $k=1,\ldots,K$
- For $m = 1, \ldots, M$
  - For $i = 1, \ldots, N$
    - Compute $\alpha(k,t)$, $\beta(k,t)$, Z via forward/backward for (X_i,Y_i)
    - Compute gradients for F^k according to

      $$g_{k,\ell,i,t} = I[y_{i,t} = k,\ y_{i,t-1} = \ell] - \alpha(\ell,t-1)\,[\exp F^k(\ell,X_i)]\,\beta(k,t)/Z$$
  - Fit regression trees $h_{k,m}$ to $(\lfloor \ell, w_t(X_i) \rfloor, g_{k,\ell,i,t})$ pairs
  - Update: $F^k := F^k + h_{k,m}$

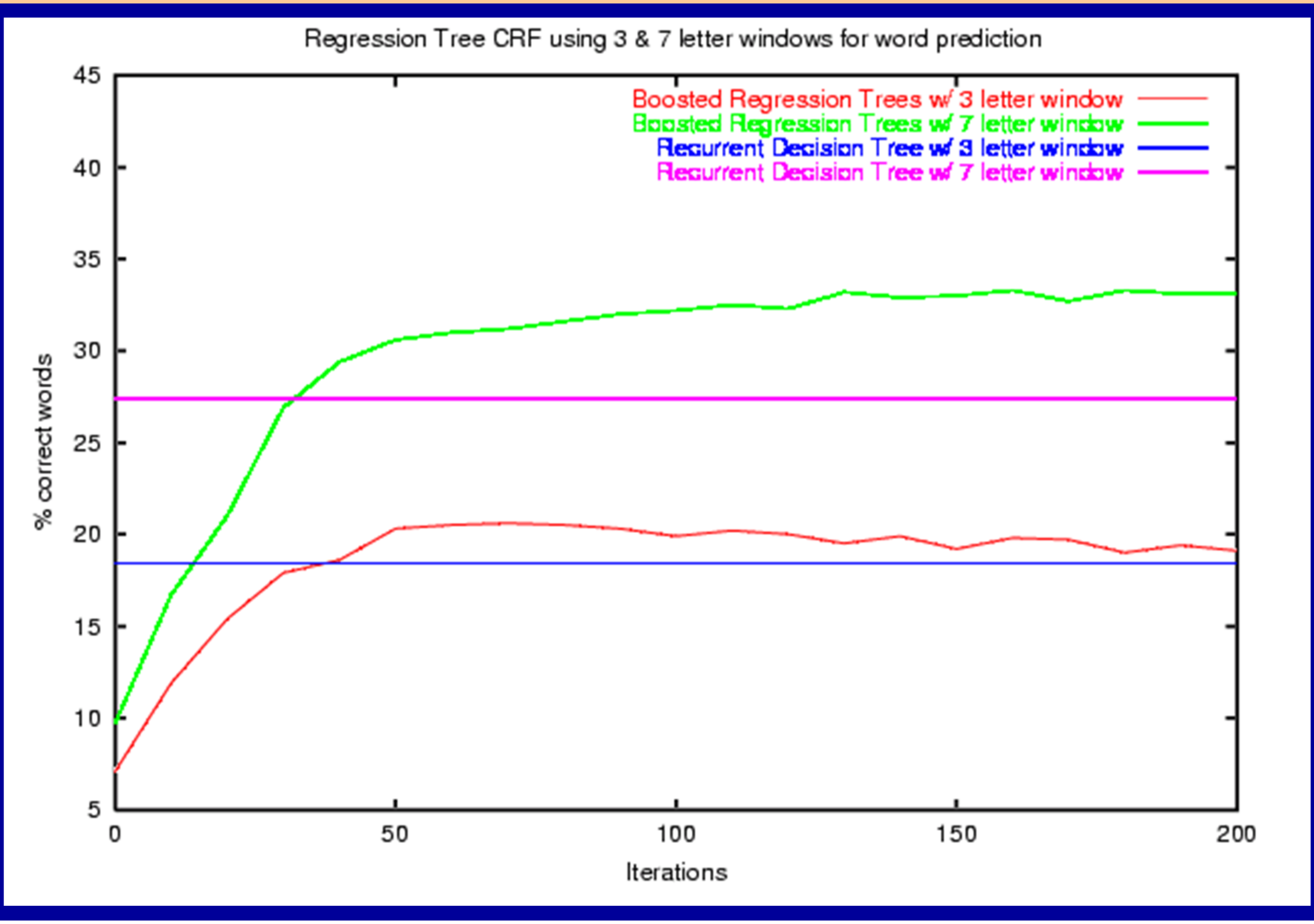
Initial Results: Training Times

- Gradient Boosting
  - 1 processor: 100 iterations requires 6 hours (compared to 16*40*100 = 64,000 hours for conjugate gradient)
  - However: Full Gradient Boosting algorithm was not implemented

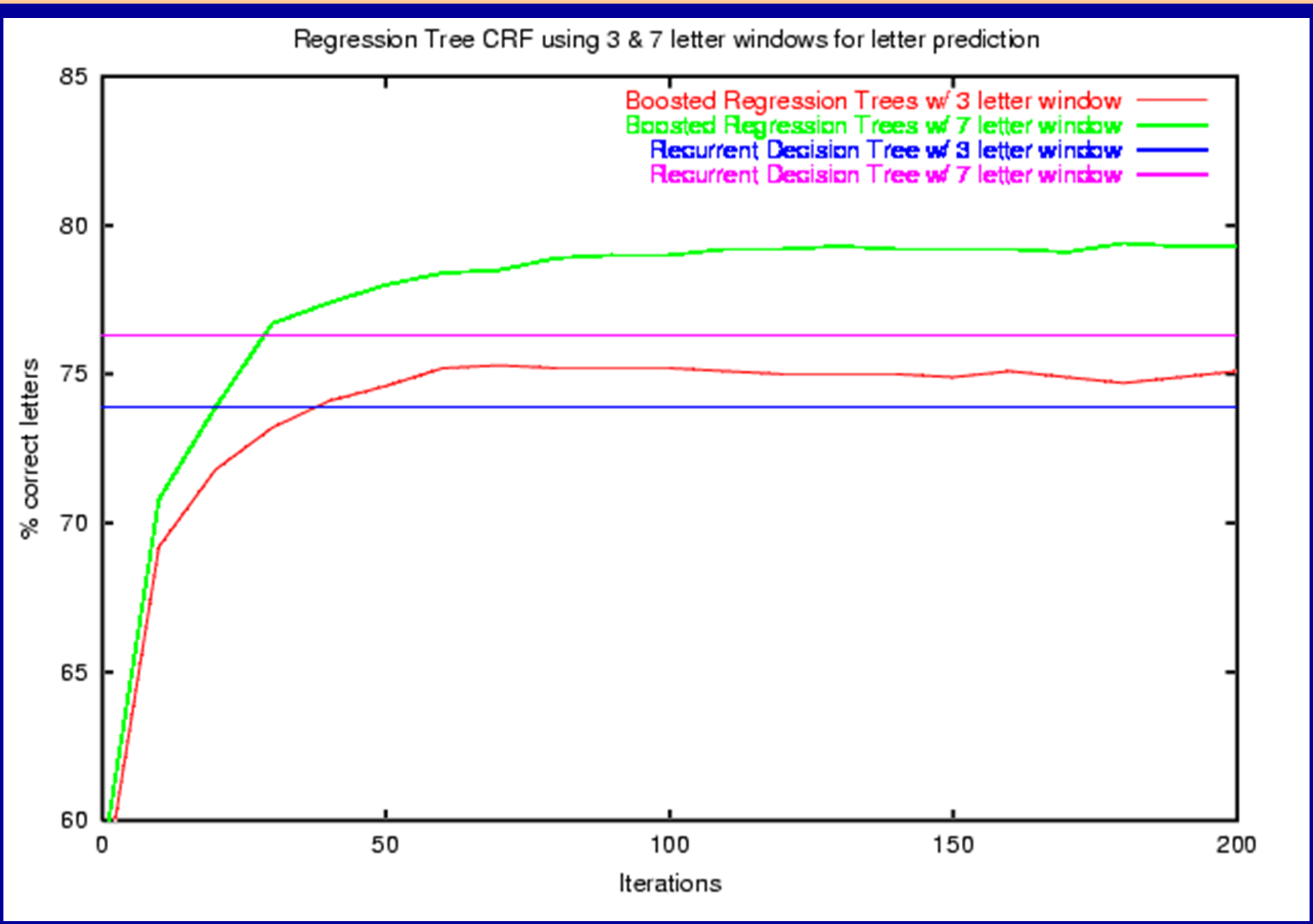
Results: Whole words correct 5-letter window Viterbi beam width 20.



Whole Words: Window Sizes of 3 and 7



Predicting Single Letters



Why Gradient Boosting is More Effective

- Each step is large: Each iteration adds one regression tree to the potential function for each class
- Parameters are introduced only as necessary
- Combinations of features are constructed

Concluding Remarks

- Many machine learning applications can be formalized as <u>Sequential Supervised Learning</u>
- Similar issues arise in other complex learning problems (e.g., spatial and relational data)
- Many methods have been developed specifically for SSL, but none is perfect
- Gradient boosting may provide a general, off-the-shelf way of fitting CRFs.