Training Conditional Random Fields via Gradient Boosting

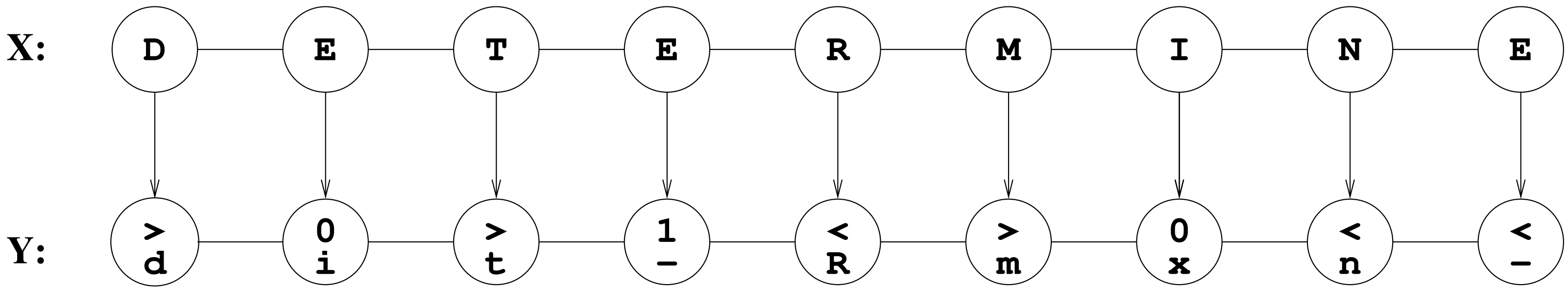
Thomas Dietterich and Adam Ashenfelter
Department of Computer Science
Oregon State University
Corvallis, Oregon 97330
{tgd|ashenfad}@cs.orst.edu

Sequential Supervised Learning

- **Given:** A set of training examples $\{\langle X_i, Y_i \rangle\}_{i=1}^{N}$ where
  Each X_i is a sequence $(\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,T_i})$ of feature vectors
  Each Y_i is a sequence $(y_{i,1}, \ldots, y_{i,T_i})$ of corresponding class labels
- **Find:** A classifier F that given a new sequence X can predict the sequence of class labels $\hat{Y}$

- **Many machine learning applications:** Part-of-speech tagging, information extraction from web pages, computer intrusion detection, text-to-speech mapping, fraud detection in transaction streams, etc.
- **No good off-the-shelf methods exist**

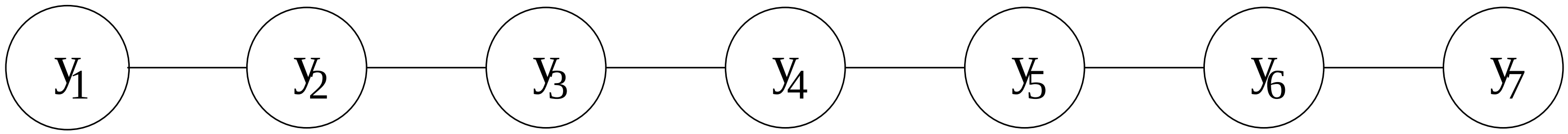
Example: Text-to-Speech Mapping (NETTalk)



- **Developed and first studied by Sejnowski and Rosenberg (1987)**
- **Standard data set:** Train on 1000 words; Test on 1000 words, 126 classes
- **Best known method:** 15-letter recurrent sliding window classifier based on decision trees and error-correcting output coding (Dietterich & Bakiri, 2001).

  This method only captures sequential relationships in one direction.

Promising New Method: Conditional Random Fields (Lafferty, McCallum, Pereira, 2001)



- The Y sequence is modeled as a Markov Random Field whose potentials $\Psi(y_{t-1}, y_t, X)$ are a function of X as well as the nodes y_{t-1} and y_t. Hence, they are *conditioned* on X.

- Formally:

$$\Psi(y_{t-1}, y_t, X) = \sum_{\alpha} \lambda_\alpha f_\alpha(y_{t-1}, y_t, X) + \sum_{\beta} \mu_\beta g_\beta(y_t, X)$$

$$P(Y|X) = \frac{\exp \sum_t \Psi(y_{t-1}, y_t, X)}{\sum_{Y'} \exp \sum_t \Psi(y'_{t-1}, y'_t, X)}$$

Advantages of Conditional Random Fields

- **Captures relationships among the y_t's**

  Unlike sliding windows and recurrent sliding windows

- **Does not impose a generative model on the x_t's. This permits greater freedom in designing features to describe X**

  Unlike Hidden Markov Models

- **Avoids "Label Bias" problem**

  Unlike Maximum Entropy Markov Models (MEMMs) and some HMM/Neural Network hybrids

Training Conditional Random Fields

- **Initial paper: Improved Iterative Scaling**

  Very slow; required careful initialization of the weights: λ_α and μ_β

- **Recent unpublished work: Conjugate Gradient**

  With careful preconditioning (Pereira's group)

  With scaled conjugate gradient (Wallach, 2002)

- **Our proposal: Apply Friedman's Gradient Boosting**

Gradient Boosting

- **Goal:** Fit a function F to a set of data points $(\mathbf{x}_i, y_i)$ to minimize a differentiable loss function $L(y_i, F(\mathbf{x}_i))$.
- **Method:** Construct a sequence of regression trees: $h(\mathbf{x}, \mathbf{a}_1), \ldots, h(\mathbf{x}, \mathbf{a}_M)$ and construct F as

$$F(\mathbf{x}) = F_0 + \phi_1 h(\mathbf{x}, \mathbf{a}_1) + \ldots + \phi_M h(\mathbf{x}, \mathbf{a}_M),$$

  where F_0 is an initial constant.
- **Functional Gradient:** Each tree $h(\mathbf{x}, \mathbf{a})$ is a least squares fit to a set of *pseudo targets*, $\tilde{y}_i$ computed as the "functional gradient" of the loss function:

$$\tilde{y}_i = - \left. \frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)} \right|_{F=F_{m-1}}$$

  In other words, $\tilde{y}_i$ captures how $F(\mathbf{x}_i)$ should change in order to reduce the loss. F_{m-1} is the fitted function after $m-1$ steps.

Gradient Boosting Algorithm

$$F_0 = \operatorname*{argmin}_{\phi} \sum_{i=1}^{N} L(y_i, \phi)$$ *compute constant*

for $m = 1$ **to** M **do**:

$$\tilde{y}_i = -\left.\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right|_{F=F_{m-1}}, \quad i = 1, N$$ *compute pseudo-targets*

$$\mathbf{a}_m = \operatorname*{argmin}_{\mathbf{a}} \sum_{i=1}^{N} [\tilde{y}_i - h(\mathbf{x}_i; \mathbf{a})]^2$$ *fit regression tree*

$$\phi_m = \operatorname*{argmin}_{\phi} \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \phi h(\mathbf{x}_i; \mathbf{a}_m))$$ *compute "step size"*

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \phi_m h(\mathbf{x}; \mathbf{a}_m)$$ *update F*

endfor

Applying Gradient Boosting to Train CRFs

Loss function: Log Conditional Pseudo Likelihood:

- Error is $\log \hat{P}(y_t|y_1, \ldots, y_{t-1}, y_{t+1}, \ldots, y_T, X) = \log \hat{P}(y_t|y_{t-1}, y_{t+1}, X)$
- Besag (1986) shows that pseudo-likelihood is a consistent estimator that avoids the need to compute $\hat{P}(y_t|X)$.
- A separate F is defined for each class label k:

$$F^k(y_{t-1}, X) = \Psi(y_{t-1}, y_t = k, X)$$

- The pseudo-target for example i, class k, time t is

$$\tilde{y}_{ikt} = y_{ikt} - \hat{P}(k|y_{t-1}, y_{t+1}, X)$$

where $\hat{P}$ is computed as

$$\hat{P}(k|y_{t-1}, y_{t+1}, X) = \frac{\exp[\Psi(y_{t-1}, k, X) + \Psi(k, y_{t+1}, X)]}{\sum_l \exp[\Psi(y_{t-1}, l, X) + \Psi(l, y_{t+1}, X)]}$$

Gradient Boosting for CRFs (2)

$F_0^k = 0 \qquad \text{for } k = 1, \ldots, K$

for $m = 1$ **to** M **do**

for $k = 1$ **to** K **do**

$$\tilde{y}_{ikt} = y_{ikt} - \hat{P}(y_{it} = k | y_{t-1}, y_{t+1}, X)$$

$$\mathbf{a}_m^k = \underset{\mathbf{a}}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_t [\tilde{y}_{ikt} - h(y_{i,t-1}, \mathbf{x}_i; \mathbf{a})]^2$$

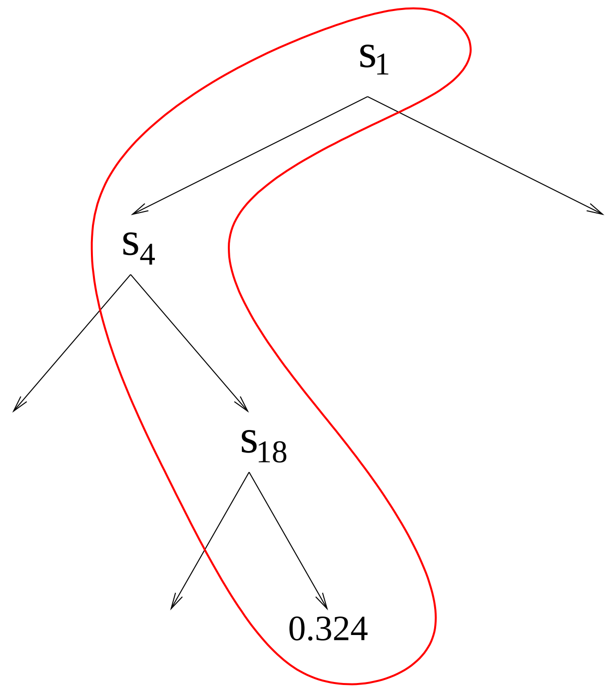
$$F_m^k(\mathbf{x}) = F_{m-1}^k(\mathbf{x}) + h(\mathbf{x}; \mathbf{a}_m^k)$$

endfor

endfor

Note: No step size is computed

Sum of Regression Trees is Equivalent to CRF



The circled path is equivalent to an expression of the form $\lambda_\alpha f_\alpha$, where

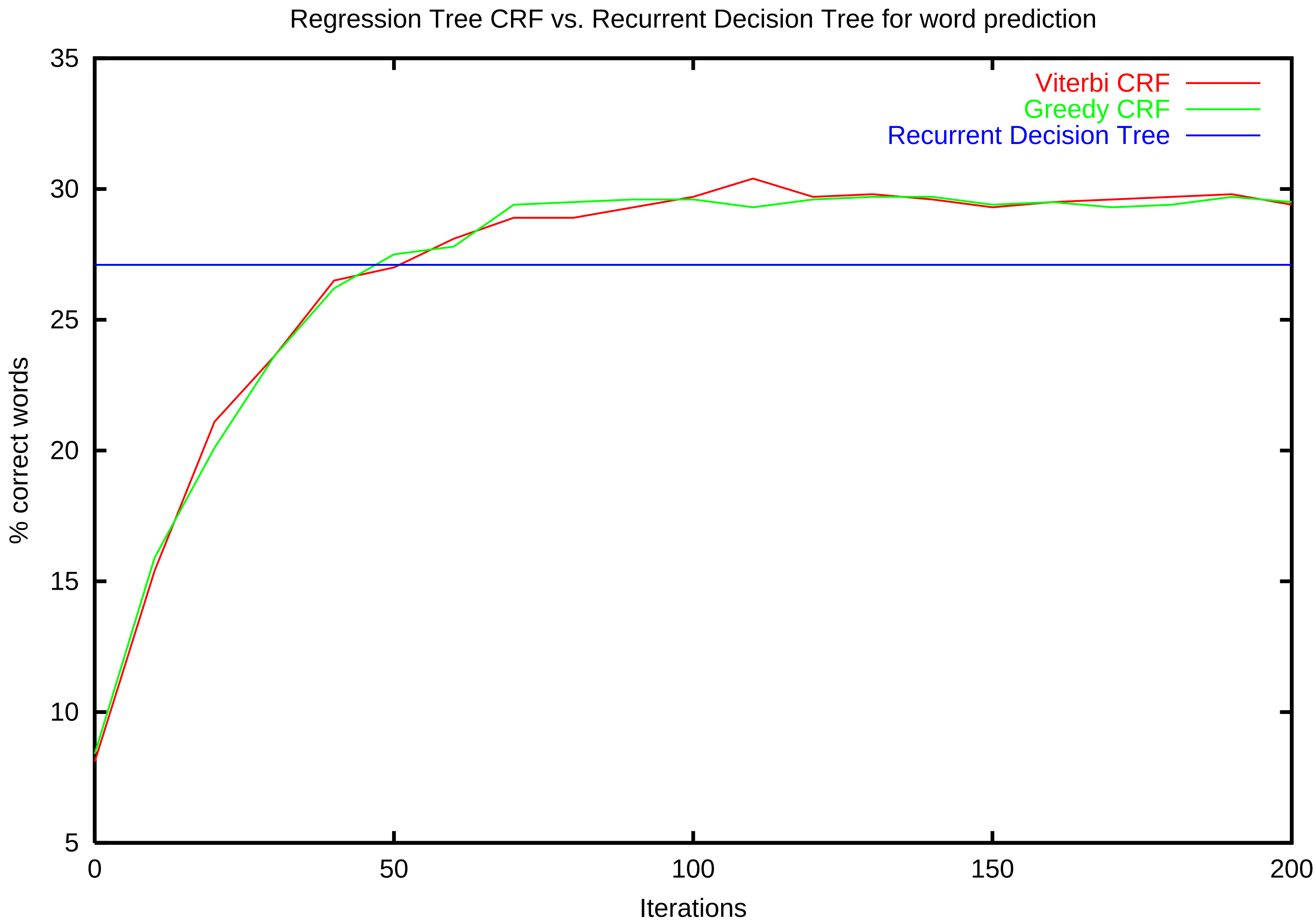
- $\lambda_\alpha = 0.324$
- $f_\alpha = s_1 \cdot s_4 \cdot s_{18}$, which is a product (conjunction) of the primitive features

Hence, the regression tree CRFs implement a form of feature selection and feature combination.

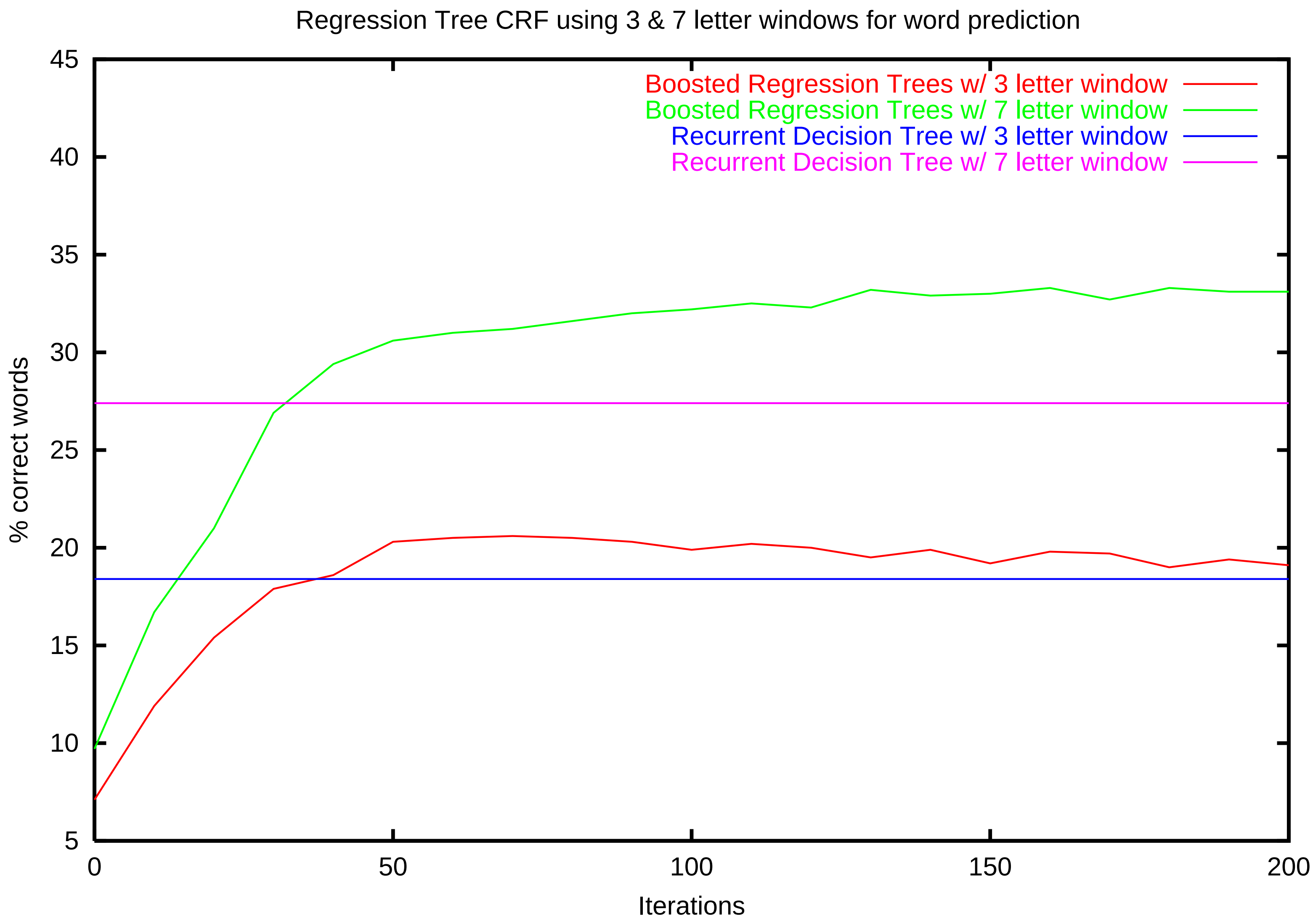
Training Times

- **Conjugate Gradient Search**
  Parallelized with 16 processors required 40 hours (i.e., 640 CPU hours) per line search
- **Gradient Boosting**
  Single processor. 100 iterations requires 6 CPU hours.

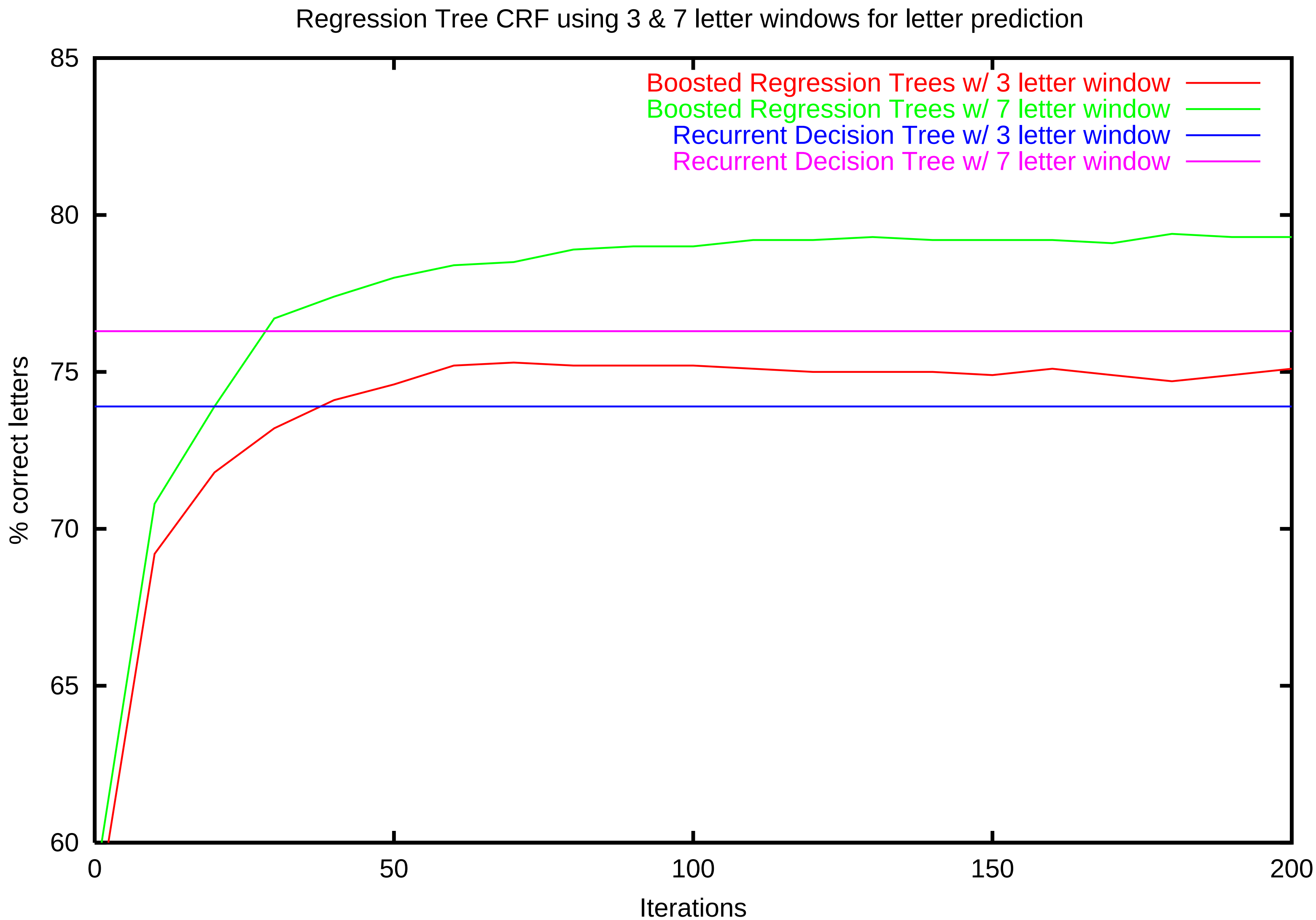
Results: Whole Words Correct
5-letter window, Viterbi beam width 20



Window sizes of 3 and 7



Accuracy of Predicting Individual Letters



Why Gradient Boosting is More Effective

- **Each "step" is large**. Each functional gradient step adds another regression tree to the potential function (for each class k).
- **Parameters are introduced only as necessary**. Previous training methods introduced all of the features from the start, which introduces a vast number of parameters to be tuned. These parameters interact, which makes optimization slow.
- **Combinations of features are constructed.** Previous methods could not consider feature combinations, because this would increase the number of parameters even further.

Conclusions

- **CRF's show promise of providing a versatile off-the-shelf method for sequential supervised learning**
  CRFs out-perform a comparable recurrent sliding window classifier.
  We plan to test CRF ensembles (or ECOC) to see if we can beat our previous best method.
- **Gradient boosting provides an efficient and effective algorithm for fitting CRFs to data**
- **Gradient boosting can be extended to fit general Markov random fields and relational probabilistic models**