Sequential Supervised Learning: General Methods for Sequence Labeling and Segmentation

Thomas G. Dietterich
Intelligent Systems and Usability Group
School of EECS
Oregon State University
Corvallis, Oregon 97331
http://www.cs.orst.edu/~tgd

Acknowledgements

- Adam Ashenfelter
- Saket Joshi
- NSF grants IIS-0083292, IIS-0307592, and EIA-0224012

Many Data Mining Problems Involve Sequential Data

- Cellular Telephone Fraud
- Part-of-speech Tagging
- Information Extraction from the Web
- Protein Secondary Structure Prediction

Cellular Telephone Fraud

- Given the sequence of recent telephone calls, can we determine which calls (if any) are fraudulent?

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>Srinivasan Seshan (Carnegie Mellon University) <dt><i>Making Virtual Worlds Real</i><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

Protein Secondary Structure Prediction

```
K S V M G H N W V L T K E A D K E
h h h h _ _ _ _ e e e e _ _ _ h h
```

- Given input sequence of amino acid residues
- Predict protein secondary structure classification:
  - h: helix
  - e: beta sheet/turn
  - _: coil

Sequential Supervised Learning (SSL)

- Given: A set of training examples of the form ($\mathbf{X}_i$,$\mathbf{Y}_i$), where
  $\mathbf{X}_i = \langle x_{i,1}, \ldots, x_{i,Ti} \rangle$ and
  $\mathbf{Y}_i = \langle y_{i,1}, \ldots, y_{i,Ti} \rangle$ 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 $\mathbf{Y}_i$
Telephone Fraud	sequence of calls	sequence of labels {ok, fraud}
Part-of-speech Tagging	sequence of words	sequence of parts of speech
Information Extraction	sequence of tokens	sequence of field labels {name, …}
Protein Secondary	sequence of amino acids	sequence of {e,h,_}

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
  - Accurate
  - Easy-to-use
  - Efficient

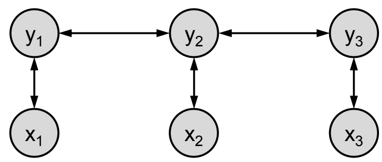
Outline

- Sequential Supervised Learning
- Off-The-Shelf Methods: Criteria
- Review of Existing and Proposed Approaches
- Two New Results
- Conclusions

Objectives

- Accurate
  - Must capture sequential relationships
  - Must allow rich input features
- Easy-to-use
  - Should not require careful modeling or assumptions about probability distributions
  - Should be robust to parameter settings
- Fast
  - Should train and run fast and scale well

Two Kinds of Relationships



- "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 should exploit both kinds of information

Rich X $\leftrightarrow$ 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: disambiguate "bank" based on surrounding words: "account", "river", "shot"
- Can't include global features
  - Example: "Sentence begins with question word"

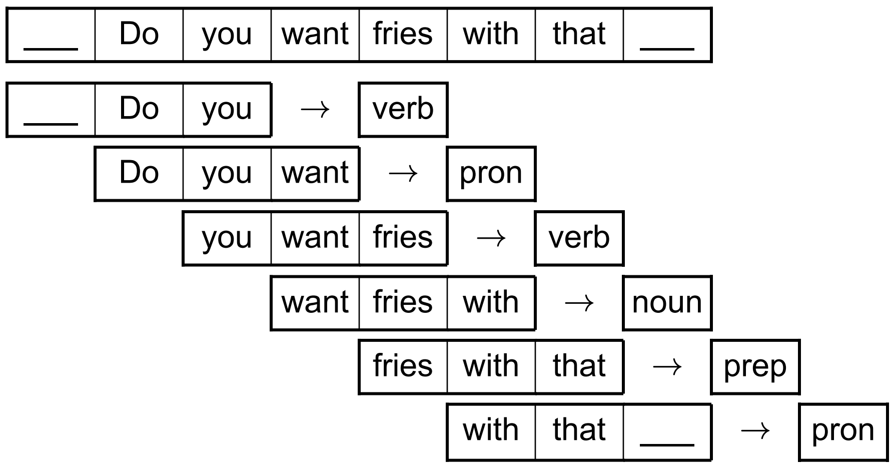
Outline

- Sequential Supervised Learning
- Off-The-Shelf Methods: Criteria
- **Review of Existing and Proposed Approaches**
- Two New Results
- Conclusions

Candidate Methods

1. Sliding windows
2. Recurrent sliding windows
3. Hidden Markov models
4. Maximum entropy Markov models
5. Input/Output Markov models
6. Conditional Random Fields
7. Maximum Margin Markov models

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

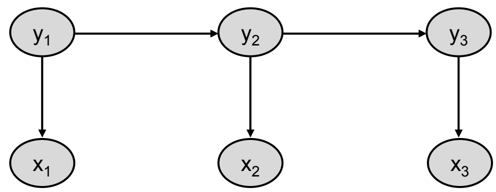
- Captures relationship among the y's, but only in one direction!
- Results on text-to-speech:

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%

WEKA RSW Package

- WEKA is a java-based machine learning and data mining package available from the University of Waikato, NZ
- Saket Joshi has implemented a general recurrent sliding window package for WEKA. Can apply any WEKA classifier with recurrent sliding windows

Hidden Markov Models

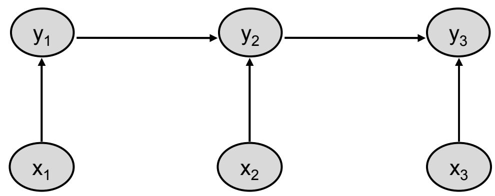


- y_t's are generated as a Markov chain
- x_t's are generated independently (as in naïve Bayes or Gaussian classifiers).

Hidden Markov Models (2)

- Models both the $x_t \leftrightarrow y_t$ relationships and the $y_t \leftrightarrow y_{t+1}$ relationships.
- Does not permit rich $X \leftrightarrow y_t$ relationships
  - Unlike the sliding window, we can't use several x_t's to predict y_t.

HMM Alternatives: Maximum Entropy Markov Models



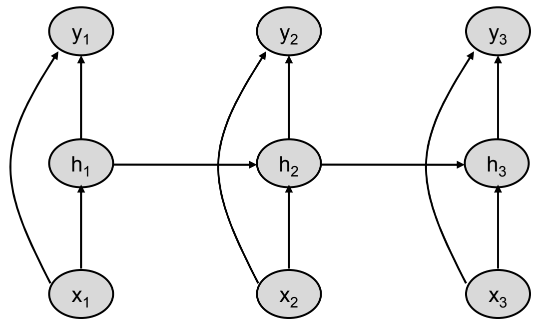
MEMM Properties

- Permits complex X $\leftrightarrow$ y_t relationships by employing a sparse maximum entropy model of $P(y_{t+1}|X, y_t)$:

  $P(y_{t+1}|X,y_t) \propto \exp(\Sigma_b\ \alpha_b\ f_b(X,y_t,y_{t+1}))$

  where f_b is a boolean feature.
- Training can be expensive (gradient descent or iterative scaling)

HMM Alternatives (2): Input/Output HMM



(Bengio & Frasconi, 1996)

IOHMM Properties

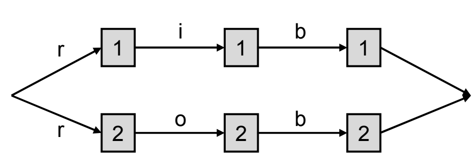
- Hidden states permit "memory" of long distance effects (beyond what is captured by the class labels)
- As with MEMM, arbitrary features of the input X can be used to predict y_t.

Label Bias Problem

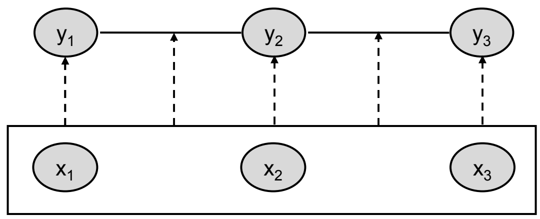
- Forward models that are normalized at each step exhibit a problem.
- Consider a domain with only two sequences: “rib” $\rightarrow$ “111” and “rob” $\rightarrow$ “222”.
- Consider what happens when an MEMM sees the sequence “rib”.

Label Bias Problem (2)

- After “r”, both labels 1 and 2 have same probability. After “i”, label 2 must *still* send all of its probability forward, even though it was expecting “o”. Result: both output strings “111” and “222” are assigned the same probability.



Conditional Random Fields



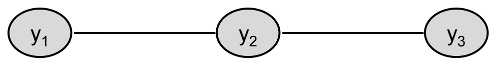
- The y_t's form a Markov Random Field conditioned on X: P(Y|X)

Lafferty, McCallum, & Pereira (2001)

Markov Random Fields

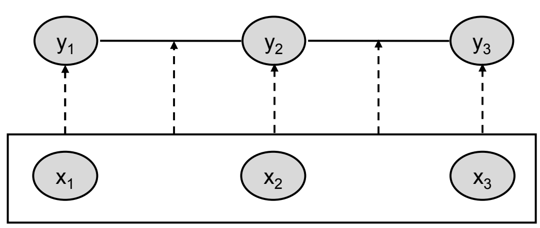
- Graph G = (V,E)
  - Each vertex $v \in V$ represents a random variable y_v.
  - Each edge represents a direct probabilistic dependency.
- $P(Y) = 1/Z \exp [\sum_c \Psi_c(c(Y))]$
  - c indexes the cliques in the graph
  - Ψ_c is the potential function for clique c
  - c(Y) selects the random variables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

A Simple MRF



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

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 = "bank" and y_t = "noun"]

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)
- Method: Gradient descent

CRFs on Part-of-speech tagging

	HMM	MEMM	CRF
baseline	5.69	6.37	5.55
spelling features	5.69	4.87	4.27
spelling features (OOV)	45.99	26.99	23.76

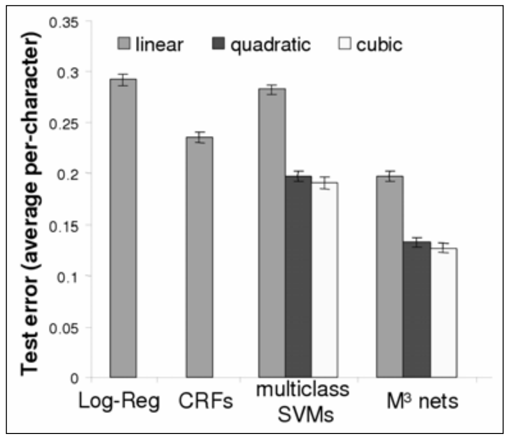
Lafferty, McCallum & Pereira (2001)
(error rates in percent)

Maximum Margin Markov networks

(Taskar, Guestrin, Koller, NIPS 2003)

- MMM = CRF but with a different objective function during training
  - HMMs: Train to maximize $P(X_i,Y_i)$ on the training data
  - CRF: Train to maximize $P(Y_i|X_i)$ on the training data
  - MMM: Train to maximize the margin
    $P(Y_i|X_i) - \max_{Y' \neq Y} P(Y'|X_i)$
    Can incorporate kernels (a la SVMs)

MMM Results on OCR Task



Summary of Methods

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

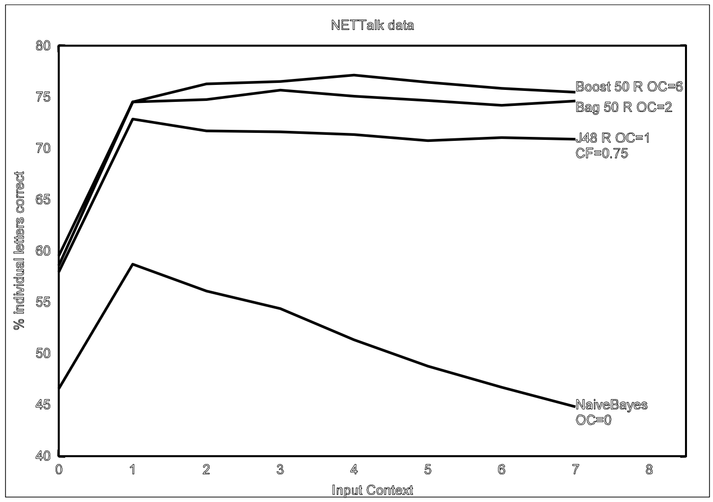
Outline

- Sequential Supervised Learning
- Off-The-Shelf Methods: Criteria
- Review of Existing and Proposed Approaches
- **Two New Results**
  - **Choosing Input and Output Window Sizes**
  - **A Faster Method for Training CRFs**
- Conclusions

Result 1: Choosing Input and Output Window Sizes

- Design Decision for most SSL Methods:
  - Size of input window
  - Amount of output context (degree of Markov model)
- How can these decisions be made?
  - Essentially a kind of feature selection
  - Maybe fit a simple model (mutual information? Naïve Bayes) and use it?

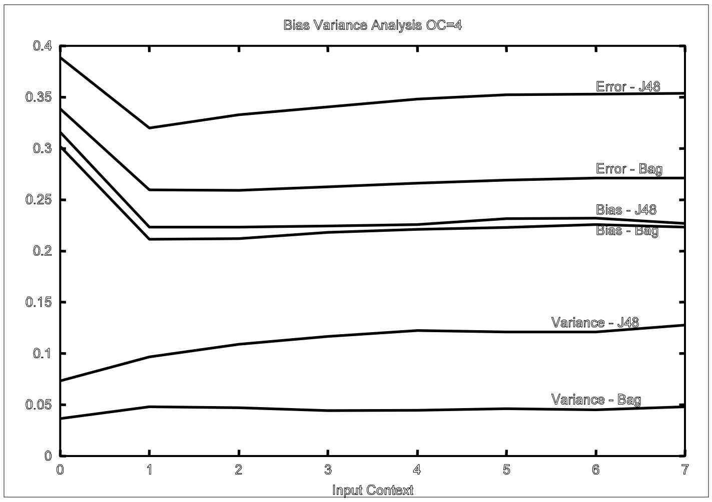
Systematic Study using WEKA



What's Going On?

- Increasing window size…
  - increases variance (extra features)
  - reduces bias (more accurate model)
- Bagging and Boosting reduce variance
  - permits them to use a larger window

Bias/Variance Study
Nettalk J48(C4.5) Bagging



Conclusion

- To choose window sizes, we must perform cross-validation
  - The best window size depends on the algorithm
  - Basing the decision on a simple algorithm will give the wrong results

Result 2: Faster Training for CRFs

- Can we make CRFs fast enough to be off-the-shelf?
  - Iterative Scaling (very very slow)
  - Gradient Descent (very slow)
  - Functional Gradient Descent (fast enough?)
    - Gradient "tree boosting"

Gradient Descent Search

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

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

- Method:

$$\theta := \theta - \eta \nabla_\theta\, L(Y,F_\theta(Y,X))$$

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)	$\nabla_\theta L(Y_1,F_\theta(Y_1,X_1))$
(X_2,Y_2)	$\nabla_\theta L(Y_2,F_\theta(Y_2,X_2))$
(X_3,Y_3)	$\nabla_\theta L(Y_3,F_\theta(Y_3,X_3))$
(X_4,Y_4)	$\nabla_\theta L(Y_4,F_\theta(Y_4,X_4))$

$$\theta := \theta - \eta \sum_i \nabla_\theta L(Y_i, F_\theta(Y_i,X_i))$$

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.
  - 100 line searches = 4000 hours (64000 CPU hours)

Functional Gradient Descent

(Breiman; Friedman; et al.)

- Standard gradient descent:

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

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

- 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 $\nabla_F \sum_i L(Y_i,F_{m-1}(Y_i,X_i))$

- Idea: Use regression trees for h_m's

Functional Gradient Descent (2)

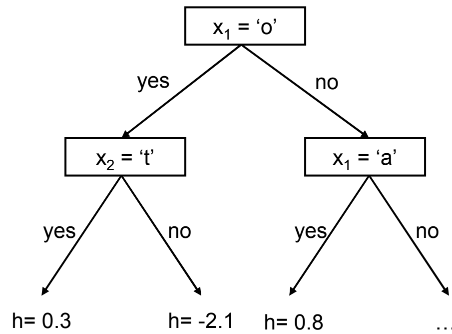
example	functional gradient	functional gradient example
(X_1,Y_1)	$\nabla_F L(Y_1,F_{m-1}(Y_1,X_1)) = g_1$	(X_1,g_1)
(X_2,Y_2)	$\nabla_F L(Y_2,F_{m-1}(Y_2,X_2)) = g_2$	(X_2,g_2)
(X_3,Y_3)	$\nabla_F L(Y_3,F_{m-1}(Y_3,X_3)) = g_3$	(X_3,g_3)
(X_4,Y_4)	$\nabla_F L(Y_4,F_{m-1}(Y_4,X_4)) = g_4$	(X_4,g_4)

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
  - $g_i := \nabla_F L(Y_i,F_{m-1}(Y_i,X_i))$, i = 1, …, 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

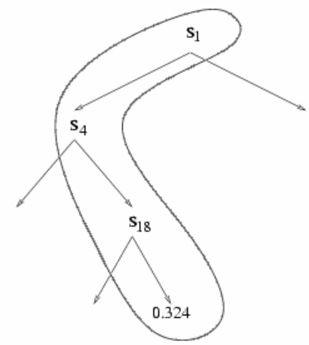
- Recall CRF model:
  $\Psi(y_{t-1},y_t,X) = \Sigma_a \lambda_a f_a(y_{t-1},y_t,X)$
  $\Psi(y_t,X) = \Sigma_b \beta_b g_b(y_t,X)]$
- Represent $\Psi(y_{t-1},y_t,X) + \Psi(y_t,X)$ by a set of K functions (one per class label):
  - $\Psi(\ell,k,X) + \Psi(k,X) = F^k(\ell,X)$, $k = 1, \ldots, K$
    - where $F^k(\ell,X) = \Sigma_m \eta_m h_{k,m}(\ell,X)$
    - 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 $\lambda_a f_a$

$\lambda_a = 0.324$

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



Advantages of Gradient Tree Boosting

- Each potential function is represented as a weighted sum of regression trees
- Trees can be learned very quickly
- Requires no assumptions about probability distributions
- Can introduce combinations of features, which is difficult to do in gradient descent (although see McCallum, UAI 2003)

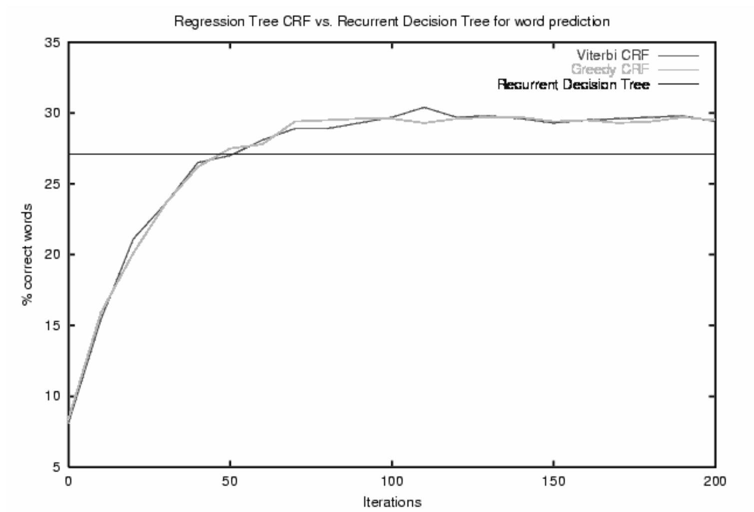
Training CRFs by Gradient Tree Boosting

- Generate training examples
  - Apply forward-backward algorithm to compute $P(y_{it}|X_i)$
  - Construct regression tree training example (X_i,g_{it})
- Fit regression tree for each output class y
- Repeat until convergence

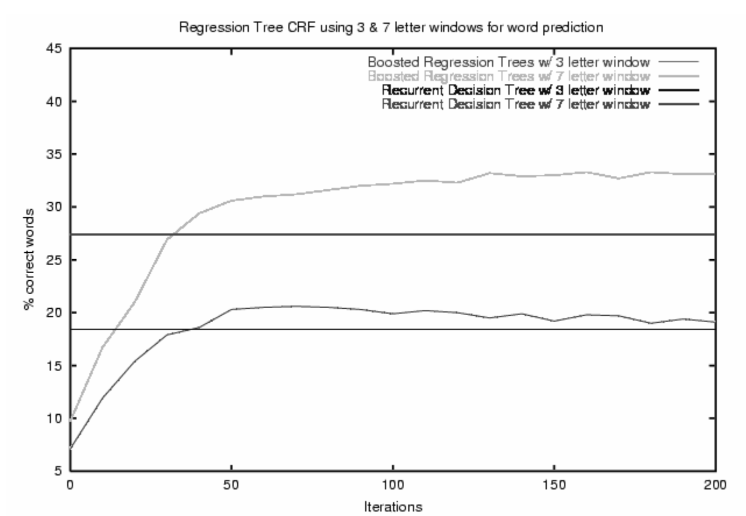
Initial Results: Training Times

- Gradient Boosting
  - 1 processor: 100 iterations requires 6 hours (compared to 16*40*100 = 64,000 hours for conjugate gradient)
  - However: only forward part of gradient boosting algorithm was implemented

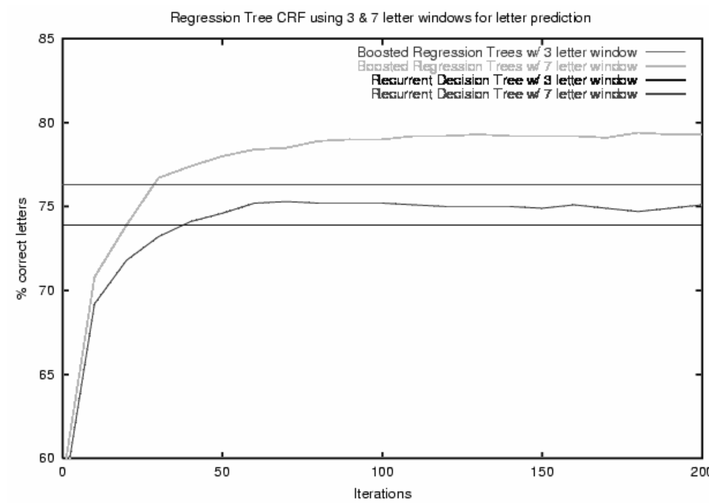
Results: Whole words correct
5-letter window
Viterbi beam width 20.



Whole Words:
Window Sizes of 3 and 7



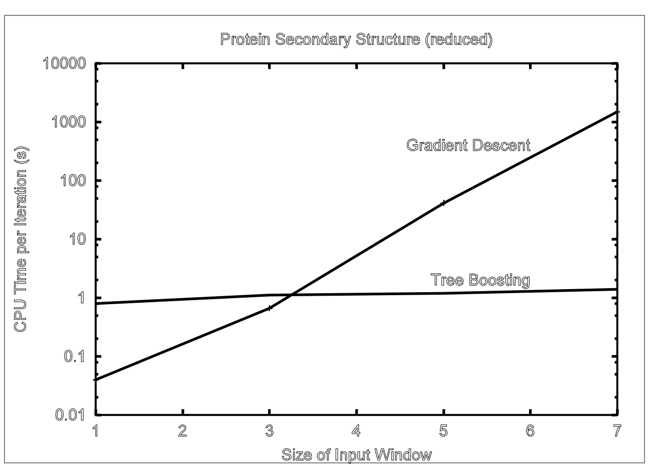
Predicting Single Letters



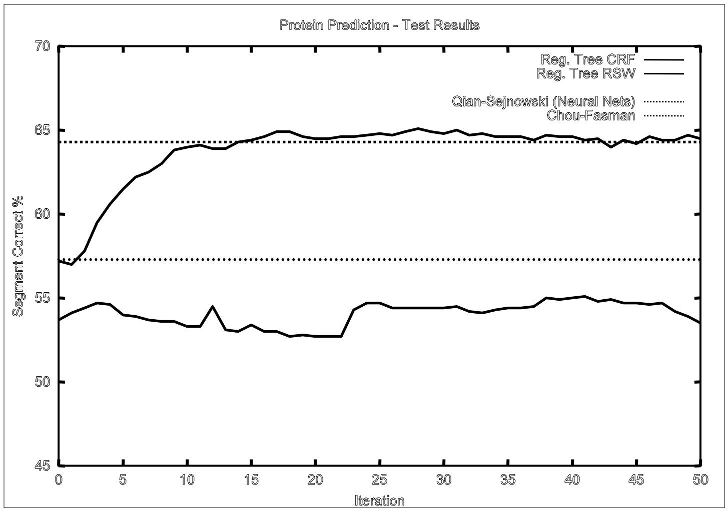
Protein Secondary Structure Prediction

(Qian & Sejnowski)

- Training time per iteration:



Protein Secondary Structure: Accuracy



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 (although see McCallum, UAI 2003)

Outline

- Sequential Supervised Learning
- Off-The-Shelf Methods: Criteria
- Review of Existing and Proposed Approaches
- Two New Results
- **Conclusions**

Discussion

- Sequential Supervised Learning problems arise in many domains
  - language processing
  - fraud detection, intrusion detection
  - bioinformatics
- Off-the-shelf methods are needed
  - Basic off-the-shelf method: recurrent sliding windows
  - Possible “high-tech” alternatives: CRFs, MMMs

Choosing Window Sizes

- Bias/Variance Tradeoff
  - Depends on particular learning algorithm
  - Requires cross-validation
- Can we find a computationally less expensive method?

Faster and More Robust Method for Training CRFs

- Boosted regression trees
  - CPU time scales linearly with window size
  - Introduces feature combinations

Open Questions

- Can we train MMMs by gradient tree boosting?
- Can we train SVMs by gradient tree boosting?
- Will standard techniques for handling missing values in trees (C4.5, CART) work for tree boosting?

Concluding Remarks

- SSL problems are instances of relational learning problems with a single relation: the sequence
- SSL requires "collective classification"
- Machine Learning is in the midst of a revolution:

  IID is dead; long live relational learning!

References

- Bakiri, G., Dietterich, T. G. (2001). Achieving high-accuracy text-to-speech with machine learning. In B. Damper (Ed.) *Data mining in speech synthesis*. Kluwer Academic Publishers, Boston, MA.
  - Describes our application of recurrent sliding windows to the text-to-speech problem.
- Bengio, Y., Frasconi, P. (1996) Input-Output HMM's for Sequence Processing, *IEEE Transactions on Neural Networks,* 7(5): 1231-1249.
- Breiman, L. (1997) Arcing the Edge. Tech. Report. Department of Statistics, University of California at Berkeley. ftp://ftp.stat.berkeley.edu/pub/users/breiman/arcing-the-edge.ps.Z
  - First description of functional gradient descent
- Dietterich, T. G. (2002). Machine Learning for Sequential Data: A Review. In T. Caelli (ed.) *Structural, Syntactic, and Statistical Pattern Recognition. Lecture Notes in Computer Science, Science,* Vol. 2396. New York: Springer Verlag, 15-30.

References

- Friedman, J. H. (2000). Greedy Function Approximation: A Gradient Boosting Machine, Stanford University, Department of Statistics, http://www-stat.stanford.edu/~jhf/ftp/trebst.ps
  - Describes gradient tree boosting method applied to supervised regression and classification.
- Friedman, J. H. (1999). Stochastic Gradient Boosting Stanford University, Department of Statistics, http://www-stat.stanford.edu/~jhf/ftp/stobst.ps.
  - Follow-on to the above paper: describes improvements to treeboosting using randomization.
- Joshi, S. (2003) Calibration of Recurrent Sliding Window Classifiers for Sequential Supervised Learning, OSU EECS Tech Report 2003-29. (http://eecs.oregonstate.edu/library/)
- Lafferty, J., McCallum, A., Pereira, F. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, In *Proceedings of the 18th International Conference on Machine Learning*, 282-289, Morgan Kaufmann, San Francisco, CA.
  - Key paper on CRFs

References

- McCallum, A., Freitag, D., and Pereira, F. (2000). Maximum Entropy Markov Models for Information Extraction and Segmentation. *Proc. 17th International Conf. on Machine Learning Morgan Kaufmann*, San Francisco, CA, 591--598.
  - Key paper on MEMMs
- McCallum, A. (2003). Efficiently Inducing Features of Conditional Random Fields. *Conference on Uncertainty in Articifical Intelligence (UAI)*. 403-410. Morgan Kaufmann.
  - Shows how to dynamically add features to a CRF during training.
- Qian, N. and Sejnowski, T. J. (1988). Predicting the secondary structure of globular proteins using neural network models, *Journal of Molecular Biology,* 202: 865-884.
  - Old protein structure prediction paper – and example SSL problem
- Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of the IEEE,* 77(2): 257-286.
  - Classic tutorial on hidden Markov models

References

- Taskar, B., Guestrin, C., and Koller, D. (2003). Max-Margin Markov Networks. To appear in Neural Information Processing Systems Conference (NIPS03), Vancouver, Canada.
  - New and very interesting work on training CRF-like models to maximize margins
- Witten, I. H, Frank, E. (1999). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann.
  - Textbook for WEKA
- WEKA: Machine Learning Software in Java. http://www.cs.waikato.ac.nz/ml/weka/
  - WEKA package