Chapter 1

PARAMETER ESTIMATION FOR STATISTICAL PARSING MODELS: THEORY AND PRACTICE OF DISTRIBUTION-FREE METHODS

Michael Collins
MIT Computer Science and Artificial Intelligence Laboratory
200 Technology Square, Cambridge, MA 02193, USA
mcollins@ai.mit.edu

Abstract A fundamental problem in statistical parsing is the choice of criteria and algorithms used to estimate the parameters in a model. The predominant approach in computational linguistics has been to use a parametric model with some variant of maximum-likelihood estimation. The assumptions under which maximum-likelihood estimation is justified are arguably quite strong. This paper discusses the statistical theory underlying various parameter-estimation methods, and gives algorithms which depend on alternatives to (smoothed) maximum-likelihood estimation. We first give an overview of results from statistical learning theory. We then show how important concepts from the classification literature – specifically, generalization results based on margins on training data – can be derived for parsing models. Finally, we describe parameter estimation algorithms which are motivated by these generalization bounds.

1. Introduction

A fundamental problem in statistical parsing is the choice of criteria and algorithms used to estimate the parameters in a model. The predominant approach in computational linguistics has been to use a parametric model with maximum-likelihood estimation, usually with some method for "smoothing" parameter estimates to deal with sparse data problems. Methods falling into this category include Probabilistic Context-Free Grammars and Hidden Markov Models, Maximum Entropy models for tagging and parsing, and recent work on Markov Random Fields.

This paper discusses the statistical theory underlying various parameter-estimation methods, and gives algorithms which depend on alternatives to (smoothed) maximum-likelihood estimation. The assumptions under which maximum-likelihood estimation is justified are arguably quite strong – in particular, an assumption is made that the structure of the statistical process generating the data is known (for example, maximum–likelihood estimation for PCFGs is justified providing that the data was actually generated by a PCFG). In contrast, work in computational learning theory has concentrated on models with the weaker assumption that training and test examples are generated from the same distribution, but that the form of the distribution is unknown: in this sense the results hold across all distributions and are called "distribution-free". The result of this work – which goes back to results in statistical learning theory by Vapnik (1998) and colleagues, and to work within Valiant's PAC model of learning (Valiant, 1984) – has been the development of algorithms and theory which provide radical alternatives to parametric maximum-likelihood methods. These algorithms are appealing in both theoretical terms, and in their impressive results in many experimental studies.

In the first part of this paper (sections 2 and 3) we describe linear models for parsing, and give an example of how the usual maximum-likelihood estimates for PCFGs can be sub-optimal. Sections 4, 5 and 6 describe the basic framework under which we will analyse parameter estimation methods. This is essentially the framework advocated by several books on learning theory (see Devroye et al., 1996; Vapnik, 1998; Cristianini and Shawe-Taylor, 2000). As a warm-up section 5 describes statistical theory for the simple case of finite hypothesis classes. Section 6 then goes on to the important case of hyperplane classifiers. Section 7 describes how concepts from the classification literature – specifically, generalization results based on margins on training data – can be derived for linear models for parsing. Section 8 describes parameter estimation algorithms motivated by these results. Section 9 gives pointers to results in the literature using the algorithms, and also discusses relationships to Markov Random Fields or maximum-entropy models (Ratnaparkhi et al., 1994; Johnson et al., 1999; Lafferty et al., 2001).

2. Linear Models

In this section we introduce the framework for the learning problem that is studied in this paper. The task is to learn a function $F : \mathcal{X} \rightarrow \mathcal{Y}$ where $\mathcal{X}$ is some set of possible inputs (for example a set of possible sentences), and $\mathcal{Y}$ is a domain of possible outputs (for example a set of parse trees). We assume:

- Training examples (x_i, y_i) for $i = 1, \ldots, m$, where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$.

- A function $\mathbf{GEN}$ 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 $\Phi(x, y) \in \Re^n$.

- A **parameter vector** $\Theta \in \Re^n$.

The components $\mathbf{GEN}, \Phi$ and Θ define a mapping from an input x to an output $F(x)$ through

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

where $\Phi(x, y) \cdot \Theta$ is the inner product $\sum_s \Theta_s \Phi_s(x, y)$. The learning task is to set the parameter values Θ using the training examples as evidence. (Note that the $\arg\max$ may not be well defined in cases where two elements of $\mathbf{GEN}(x)$ get the same score $\Phi(x, y) \cdot \Theta$. In general we will assume that there is some fixed, deterministic way of choosing between elements with the same score – this can be achieved by fixing some arbitrary ordering on the set $\mathcal{Y}$.)

Several natural language problems can be seen to be special cases of this framework, through different definitions of $\mathbf{GEN}$ and Φ. In the next section we show how weighted context-free grammars are one special case. Tagging problems can also be framed in this way (e.g., Collins, 2002b): in this case $\mathbf{GEN}(x)$ is all possible tag sequences for an input sentence x. In (Johnson et al., 1999), $\mathbf{GEN}(x)$ is the set of parses for a sentence x under an LFG grammar, and the representation Φ can track arbitrary features of these parses. In (Ratnaparkhi et al., 1994; Collins, 2000; Collins and Duffy, 2002) $\mathbf{GEN}(x)$ is the top N parses from a first pass statistical model, and the representation Φ tracks the log-probability assigned by the first pass model together with arbitrary additional features of the parse trees. Walker et al. (2001) show how the approach can be applied to NLP generation: in this case x is a semantic representation, y is a surface string, and $\mathbf{GEN}$ is a deterministic system that maps x to a number of candidate surface realizations. The framework can also be considered to be a generalization of multi-class classification problems, where for all inputs x, $\mathbf{GEN}(x)$ is a fixed set of k labels $\{1, 2, \ldots, k\}$ (e.g., see Crammer and Singer, 2001; Elisseeff et al., 1999).

2.1 Weighted Context-Free Grammars

Say we have a context-free grammar (see (Hopcroft and Ullman, 1979) for a formal definition) $G = (N, \Sigma, R, S)$ where N is a set of non-terminal symbols, Σ is an alphabet, R is a set of rules of the form $X \to Y_1 Y_2 \cdots Y_n$ for $n \geq 0, X \in N, Y_i \in (N \cup \Sigma)$, and S is a distinguished start symbol in N. The grammar defines a set of possible strings, and possible string/tree pairs, in a language. We use $\mathbf{GEN}(x)$ for all $x \in \Sigma^*$ to denote the set of possible trees (parses) for the string x under the grammar (this set will be empty for strings not generated by the grammar).

For convenience we will take the rules in R to be placed in some arbitrary ordering $r_1, \ldots, r_n$. A weighted grammar $G = (N, \Sigma, R, S, \Theta)$ also includes a parameter vector $\Theta \in \Re^n$ which assigns a weight to each rule in R: the i-th component of Θ is the weight of rule r_i. Given a sentence x and a tree y spanning the sentence, we assume a function $\Phi(x, y)$ which tracks the counts of the rules in (x, y). Specifically, the i-th component of $\Phi(x, y)$ is the number of times rule r_i is seen in (x, y). Under these definitions, the weighted context-free grammar defines a function h_Θ from sentences to trees:

$$h_\Theta(x) = \arg \max_{y \in \mathbf{GEN}(x)} \Phi(x, y) \cdot \Theta \tag{1.1}$$

Finding $h_\Theta(x)$, the parse with the largest weight, can be achieved in polynomial time using the CKY parsing algorithm (in spite of a possibly exponential number of members of $\mathbf{GEN}(x)$), assuming that the weighted CFG can be converted to an equivalent weighted CFG in Chomsky Normal Form.

In this paper we consider the structure of the grammar to be fixed, the learning problem being reduced to setting the values of the parameters Θ. A basic question is as follows: given a "training sample" of sentence/tree pairs $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, what criterion should be used to set the weights in the grammar? A very common method – that of Probabilistic Context-Free Grammars (PCFGs) – uses the parameters to define a distribution $P(x, y|\Theta)$ over possible sentence/tree pairs in the grammar. Maximum likelihood estimation is used to set the weights. We will consider the assumptions under which this method is justified, and argue that these assumptions are likely to be too strong. We will also give an example to show how PCFGs can be badly mislead when the assumptions are violated. As an alternative we will propose distribution-free methods for estimating the weights, which are justified under much weaker assumptions, and can give quite different estimates of the parameter values in some situations.

We would like to generalize weighted context-free grammars by allowing the representation $\Phi(x, y)$ to be essentially any feature-vector representation of the tree. There is still a grammar G, defining a set of candidates $\mathbf{GEN}(x)$ for each sentence. The parameters of the parser are a vector Θ. The parser's output is defined in the same way as equation (1.1). The important thing in this generalization is that the representation Φ is now not necessarily directly tied to the productions in the grammar. This is essentially the approach advocated by (Ratnaparkhi et al., 1994; Abney, 1997; Johnson et al., 1999), although the criteria that we will propose for setting the parameters Θ are quite different.

While superficially this might appear to be a minor change, it introduces two major challenges. The first problem is how to set the parameter values under these general representations. The PCFG method described in the next section, which results in simple relative frequency estimators of rule weights, is not applicable to more general representations. A generalization of PCFGs, Markov Random Fields (MRFs), has been proposed by several authors (Ratnaparkhi et al., 1994; Abney, 1997; Johnson et al., 1999; Della Pietra et al., 1997). In this paper we give several alternatives to MRFs, and we describe the theory and assumptions which underly various models.

The second challenge is that now that the parameters are not tied to rules in the grammar the CKY algorithm is not applicable – in the worst case we may have to enumerate all members of $\mathbf{GEN}(x)$ explicitly to find the highest-scoring tree. One practical solution is to define the "grammar" G as a first pass statistical parser which allows dynamic programming to enumerate its top N candidates. A second pass uses the more complex representation Φ to choose the best of these parses. This is the approach used in several papers (e.g., Ratnaparkhi et al., 1994; Collins, 2000; Collins and Duffy, 2002).

3. Probabilistic Context-Free Grammars

This section reviews the basic theory underlying Probabilistic Context-Free Grammars (PCFGs). Say we have a context-free grammar $G = (N, \Sigma, R, S)$ as defined in section 2.1. We will use $\mathcal{T}$ to denote the set of all trees generated by G. Now say we assign a weight $p(r)$ in the range 0 to 1 to each rule r in R. Assuming some arbitrary ordering $r_1, \ldots, r_n$ of the n rules in R, we use Θ to denote a vector of parameters, $\Theta = \langle \log p(r_1), \log p(r_2), \ldots, \log p(r_n) \rangle$. If $c(T, r)$ is the number of times rule r is seen in a tree T, then the "probability" of a tree T can be written as

$$P(T|\Theta) = \prod_{r \in R} p(r)^{c(T,r)}$$

or equivalently

$$\log P(T|\Theta) = \sum_{r \in R} c(T, r) \log p(r) = \Phi(T) \cdot \Theta$$

where we define $\Phi(T)$ to be an n-dimensional vector whose i-th component is $c(T, r_i)$.

Booth and Thompson (1973) give conditions on the weights which ensure that $P(T|\Theta)$ is a valid probability distribution over the set $\mathcal{T}$, in other words that $\sum_{T \in \mathcal{T}} P(T|\Theta) = 1$, and $\forall T \in \mathcal{T}$, $P(T|\Theta) \geq 0$. The main condition is that the parameters define conditional distributions over the alternative ways of rewriting each non-terminal symbol in the grammar. Formally, if we use $R(\alpha)$ to denote the set of rules whose left hand side is some non-terminal α, then $\forall \alpha \in N, \quad \sum_{r \in R(\alpha)} p(r) = 1$ and $\forall r \in R(\alpha), \quad p(r) \geq 0$. Thus the weight associated with a rule $\alpha \to \beta$ can be interpreted as a conditional probability $P(\beta|\alpha)$ of α rewriting as β (rather than any of the other alternatives in $R(\alpha)$).[1]

We can now study how to train the grammar from a training sample of trees. Say there is a training set of trees $\{T_1, T_2, \ldots, T_m\}$. The *log-likelihood* of the training set given parameters Θ is $L(\Theta) = \sum_j \log P(T_j|\Theta)$. The maximum-likelihood estimates are to take $\hat{\Theta} = \arg\max_{\Theta \in \Omega} L(\Theta)$, where Ω is the set of allowable parameter settings (i.e., the parameter settings which obey the constraints in Booth and Thompson, 1973). It can be proved using constrained optimization techniques (i.e., using Lagrange multipliers) that the maximum-likelihood estimate for the weight of a rule $r = \alpha \rightarrow \beta$ is $p(\alpha \rightarrow \beta) = \sum_j c(T_j, \alpha \rightarrow \beta) / \sum_j c(T_j, \alpha)$ (here we overload the notation c so that $c(T, \alpha)$ is the number of times non-terminal α is seen in T). So "learning" in this case involves taking a simple ratio of frequencies to calculate the weights on rules in the grammar.

So under what circumstances is maximum-likelihood estimation justified? Say there is a true set of weights Θ^*, which define an underlying distribution $P(T|\Theta^*)$, and that the training set is a sample of size m from this distribution. Then it can be shown that as m increases to infinity, then with probability 1 the parameter estimates $\hat{\Theta}$ converge to values which give the same distribution over trees as the "true" parameter values Θ^*.

To illustrate the deficiencies of PCFGs, we give a simple example. Say we have a random process which generates just 3 trees, with probabilities $\{p_1, p_2, p_3\}$, as shown in figure 1.1a. The training sample will consist of a set of trees drawn from this distribution. A test sample will be generated from the same distribution, but in this case the trees will be hidden, and only the surface strings will be seen (i.e., $\langle aaaa \rangle$, $\langle aaa \rangle$ and $\langle a \rangle$ with probabilities p_1, p_2, p_3 respectively). We would like to learn a weighted CFG with as small error as possible on a randomly drawn test sample.

As the size of the training sample goes to infinity, the relative frequencies of trees $\{T_1, T_2, T_3\}$ in the training sample will converge to $\{p_1, p_2, p_3\}$. This makes it easy to calculate the rule weights that maximum-likelihood estimation converges to – see figure 1.1b. We will call the PCFG with these asymptotic weights the *asymptotic PCFG*. Notice that the grammar generates trees never seen in training data, shown in figure 1.1c. The grammar is ambiguous for strings $\langle aaaa \rangle$ (both T_1 and T_4 are possible) and $\langle aaa \rangle$ (T_2 and T_5 are possible). In fact, under certain conditions T_4 and T_5 will get higher probabilities under the asymptotic PCFG than T_1 and T_2, and both strings $\langle aaaa \rangle$ and $\langle aaa \rangle$ will be mis-parsed. Figure 1.1d shows the distribution of the asymptotic PCFG over the 8 trees when $p_1 = 0.2, p_2 = 0.1$ and $p_3 = 0.7$. In this case both ambiguous strings are mis-parsed by the asymptotic PCFG, resulting in an expected error rate of $(p_1 + p_2) = 30\%$ on newly drawn test examples.

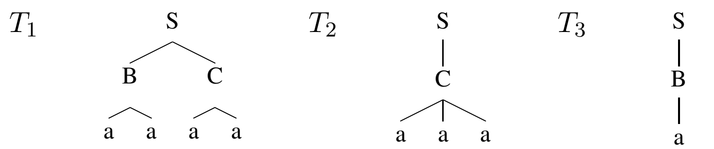

Figure 1.1a. Training and test data consists of trees T_1, T_2 and T_3 drawn with probabilities p_1, p_2 and p_3.

Rule Number	Rule	Asymptotic ML Estimate
1	`S` $\rightarrow$ `B C`	p_1
2	`S` $\rightarrow$ `C`	p_2
3	`S` $\rightarrow$ `B`	p_3
4	`B` $\rightarrow$ `a a`	$p_1/(p_1+p_3)$
5	`B` $\rightarrow$ `a`	$p_3/(p_1+p_3)$
6	`C` $\rightarrow$ `a a`	$p_1/(p_1+p_2)$
7	`C` $\rightarrow$ `a a a`	$p_2/(p_1+p_2)$

Figure 1.1b. The ML estimates of rule probabilities converge to simple functions of p_1, p_2, p_3 as the training size goes to infinity.

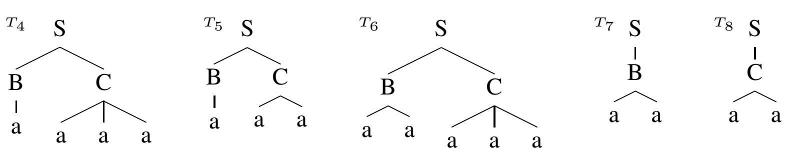

Figure 1.1c. The CFG also generates $T_4, \ldots, T_8$, which are unseen in training or test data.

Tree	Rules used	Asymptotic Estimate
T_1	1,4,6	0.0296
T_2	2,7	0.0333
T_3	3,5	0.544
T_4	1,5,7	0.0519
T_5	1,5,6	0.104
T_6	1,4,7	0.0148
T_7	3,4	0.156
T_8	2,6	0.0667

Figure 1.1d. The probabilities assigned to the trees as the training size goes to infinity, for $p_1 = 0.2, p_2 = 0.1, p_3 = 0.7$. Notice that $P(T_4) > P(T_1)$, and $P(T_5) > P(T_2)$, so the induced PCFG will incorrectly map $\langle \texttt{aaaa} \rangle$ to T_4 and $\langle \texttt{aaa} \rangle$ to T_2.

This is a striking failure of the PCFG when we consider that it is easy to derive weights on the grammar rules which parse both training and test examples with no errors.[2] On this example there exist weighted grammars which make no errors, but the maximum likelihood estimation method will fail to find these weights, even with unlimited amounts of training data.

4. Statistical Learning Theory

The next 4 sections of this chapter describe theoretical results underlying the parameter estimation algorithms in section 8. In sections 4.1 to 4.3 we describe the basic framework under which we will analyse the various learning approaches. In section 5 we describe analysis for a simple case, finite hypothesis classes, which will be useful for illustrating ideas and intuition underlying the methods. In section 6 we describe analysis of hyperplane classifiers. In section 7 we describe how the results for hyperplane classifiers can be generalized to apply to the linear models introduced in section 2.

4.1 A General Framework for Supervised Learning

This section introduces a general framework for supervised learning problems. There are several books (Devroye et al., 1996; Vapnik, 1998; Cristianini and Shawe-Taylor, 2000) which cover the material in detail. We will use this framework to analyze both parametric methods (PCFGs, for example), and the distribution–free methods proposed in this paper. We assume the following:

- An input domain $\mathcal{X}$ and an output domain $\mathcal{Y}$. The task will be to learn a function mapping each element of $\mathcal{X}$ to an element of $\mathcal{Y}$. In parsing, $\mathcal{X}$ is a set of possible sentences and $\mathcal{Y}$ is a set of possible trees.

- There is some underlying probability distribution $D(x, y)$ over $\mathcal{X} \times \mathcal{Y}$. The distribution is used to generate both training and test examples. It is an unknown distribution, but it is constant across training and test examples – both training and test examples are drawn independently, identically distributed from $D(x, y)$.

- There is a loss function $L(y, \hat{y})$ which measures the cost of proposing an output $\hat{y}$ when the "true" output is y. A commonly used cost is the 0-1 loss $L(y, \hat{y}) = 0$ if $y = \hat{y}$, and $L(y, \hat{y}) = 1$ otherwise. We will concentrate on this loss function in this paper.

- Given a function h from $\mathcal{X}$ to $\mathcal{Y}$, its *expected loss* is

$$Er(h) = \sum_{x,y} D(x, y) L(y, h(x))$$

Under 0-1 loss this is the expected proportion of errors that the hypothesis makes on examples drawn from the distribution D. We would like to learn a function whose expected loss is as low as possible: $Er(h)$ is a measure of how successful a function h is. Unfortunately, because we do not have direct access to the distribution D, we cannot explicitly calculate the expected loss of a hypothesis.

- The training set is a sample of m pairs $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ drawn from the distribution D. This is the only information we have about D. The *empirical loss* of a function h on the training sample is

$$\hat{Er}(h) = \frac{1}{m} \sum_i L(y_i, h(x_i))$$

Finally, a useful concept is the *Bayes Optimal* hypothesis, which we will denote as h_B. It is defined as $h_B(x) = \arg\max_{y \in \mathcal{Y}} D(x, y)$. The Bayes optimal hypothesis simply outputs the most likely y under the distribution D for each input x. It is easy to prove that this function minimizes the expected loss $Er(h)$ over the space of all possible functions – the Bayes optimal hypothesis cannot be improved upon. Unfortunately, in general we do not know $D(x, y)$, so the Bayes optimal hypothesis, while useful as a theoretical construct, cannot be obtained directly in practice. Given that the only access to the distribution $D(x, y)$ is indirect, through a training sample of finite size m, the learning problem is to find a hypothesis whose expected risk is low, using only the training sample as evidence.

4.2 Parametric Models

Parametric models attempt to solve the supervised learning problem by explicitly modeling either the joint distribution $D(x, y)$ or the conditional distributions $D(y|x)$ for all x.

In the joint distribution case, there is a parameterized probability distribution $P(x, y|\Theta)$. As the parameter values Θ are varied the distribution will also vary. The parameter space Ω is a set of possible parameter values for which $P(x, y|\Theta)$ is a well-defined distribution (i.e., for which $\sum_{x,y} P(x, y|\Theta) = 1$).

A crucial assumption in parametric approaches is that there is some $\Theta^* \in \Omega$ such that $D(x, y) = P(x, y|\Theta^*)$. In other words, we assume that D is a member of the set of distributions under consideration. Now say we have a training sample $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ drawn from $D(x, y)$. A common estimation method is to set the parameters to the maximum-likelihood estimates, $\hat{\Theta} = \arg\max_{\Theta \in \Omega} \sum_i \log P(x_i, y_i|\Theta)$. Under the assumption that $D(x, y) = P(x, y|\Theta^*)$ for some $\Theta^* \in \Omega$, for a wide class of distributions it can be shown that $P(x, y|\hat{\Theta})$ converges to $D(x, y)$ in the limit as the training size m goes to infinity. Because of this, if we consider the function $\hat{h}(x) = \arg\max_{y\in\mathcal{Y}} P(x, y|\hat{\Theta})$, then in the limit $\hat{h}(x)$ will converge to the Bayes optimal function $h_B(x)$. So under the assumption that $D(x, y) = P(x, y|\Theta^*)$ for some $\Theta^* \in \Omega$, and with infinite amounts of training data, the maximum-likelihood method is provably optimal.

Methods which model the conditional distribution $D(y|x)$ are similar. The parameters now define a conditional distribution $P(y|x, \Theta)$. The assumption is that there is some Θ^* such that $\forall x,\ D(y|x) = P(y|x, \Theta^*)$. Maximum-likelihood estimates can be defined in a similar way, and in this case the function $\hat{h}(x) = \arg\max_{y\in\mathcal{Y}} P(y|x, \hat{\Theta})$ will converge to the Bayes optimal function $h_B(x)$ as the sample size goes to infinity.

4.3 An Overview of Distribution-Free Methods

From the arguments in the previous section, parametric methods are optimal provided that two assumptions hold:

1 The distribution generating the data is in the class of distributions being considered.

2 The training set is large enough for the distribution defined by the maximum-likelihood estimates to converge to the "true" distribution $D(x, y)$ (in general the guarantees of ML estimation are asymptotic, holding only in the limit as the training data size goes to infinity).

This paper proposes alternatives to maximum-likelihood methods which give theoretical guarantees without making either of these assumptions. There is no assumption that the distribution generating the data comes from some predefined class – the only assumption is that the same, unknown distribution generates both training and test examples. The methods also provide bounds suggesting how many training samples are required for learning, dealing with the case where there is only a finite amount of training data.

A crucial idea in distribution-free learning is that of a *hypothesis space*. This is a set of functions under consideration, each member of the set being a function $h : \mathcal{X} \rightarrow \mathcal{Y}$. For example, in weighted context-free grammars the hypothesis space is

$$\mathcal{H} = \{h_\Theta : \Theta \in \Re^n\}$$

where

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

So each possible parameter setting defines a different function from sentences to trees, and $\mathcal{H}$ is the infinite set of all such functions as Θ ranges over the parameter space $\Re^n$.

Learning is then usually framed as the task of choosing a "good" function in $\mathcal{H}$ on the basis of a training sample as evidence. Recall the definition of the expected error of a hypothesis $Er(h) = \sum_{x,y} D(x,y)L(y,h(x))$. We will use h^* to denote the "best" function in $\mathcal{H}$ by this measure,

$$h^* = \arg\min_{h \in \mathcal{H}} Er(h) = \arg\min_{h \in \mathcal{H}} \sum_{x,y} D(x,y)L(y,h(x))$$

As a starting point, consider the following approach. Given a training sample (x_i, y_i) for $i = 1, \ldots, m$, consider a method which simply chooses the hypothesis with minimum empirical error, that is

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{Er}(h) = \arg\min_{h \in \mathcal{H}} \frac{1}{m} \sum_i L(y_i, h(x_i))$$

This strategy is called "Empirical Risk Minimization" (ERM) by Vapnik (1998). Two questions which arise are:

- In the limit, as the training size goes to infinity, does the error of the ERM method $Er(\hat{h})$ approach the error of the best function in the set, $Er(h^*)$, regardless of the underlying distribution $D(x,y)$? In other words, is this method of choosing a hypothesis always consistent?

  The answer to this depends on the nature of the hypothesis space $\mathcal{H}$. For finite hypothesis spaces the ERM method is always consistent. For many infinite hypothesis spaces, such as the hyperplane classifiers described in section 6 of this paper, the method is also consistent. However, some infinite hypothesis spaces can lead to the method being inconsistent – specifically, if a measure called the Vapnik-Chervonenkis (VC) dimension (Vapnik and Chervonenkis, 1971) of $\mathcal{H}$ is infinite, the ERM method may be inconsistent. Intuitively, the VC dimension can be thought of as a measure of the complexity of an infinite set of hypotheses.

- If the method is consistent, how quickly does $Er(\hat{h})$ converge to $Er(h^*)$? In other words, how much training data is needed to have a good chance of getting close to the best function in $\mathcal{H}$? We will see in the next section that the convergence rate depends on various measures of the "size" of the hypothesis space. For finite sets, the rate of convergence depends directly upon the size of $\mathcal{H}$. For infinite sets, several measures have been proposed – we will concentrate on rates of convergence based on a concept called the *margin* of a hypothesis on training examples.

5. Convergence Bounds for Finite Sets of Hypotheses

This section gives results and analysis for situations where the hypothesis space $\mathcal{H}$ is a finite set. This is in some ways an unrealistically simple situation – many hypothesis spaces used in practice are infinite sets – but we give the results and proofs because they can be useful in developing intuition for the nature of convergence bounds. In the following sections we consider infinite hypothesis spaces such as weighted context-free grammars.

A couple of basic results from probability theory will be very useful. The first results are the *Chernoff bounds*. Consider a binary random variable X (such as the result of a coin toss) which has probability p of being 1, and $(1-p)$ of being 0. Now consider a sample of size m, $\{x_1, x_2, \ldots, x_m\}$ drawn from this process. Define the relative frequency of $x_i = 1$ (the coin coming up heads) in this sample to be $\hat{p} = \sum_i x_i/m$. The relative frequency $\hat{p}$ is a very natural estimate of the underlying probability p, and by the law of large numbers $\hat{p}$ will converge to p as the sample size m goes to infinity. Chernoff bounds give results concerning how quickly $\hat{p}$ converges to p. Thus Chernoff bounds go a step further than the law of large numbers, which is an asymptotic result (a result concerning what happens as the sample size goes to infinity). The bounds are:

Theorem 1 (Chernoff Bounds). *For all* $p \in [0,1], \epsilon > 0$*, with the probability* P *being taken over the distribution of training samples of size* m *generated with underlying parameter* p*,*

$$P[p - \hat{p} > \epsilon] \leq e^{-2m\epsilon^2} \tag{1.2}$$

$$P[\hat{p} - p > \epsilon] \leq e^{-2m\epsilon^2} \tag{1.3}$$

$$P[\,|\hat{p} - p| > \epsilon] \leq 2e^{-2m\epsilon^2} \tag{1.4}$$

The first bound states that for all values of p, and for all values of ϵ, if we repeatedly draw training samples of size m of a binary variable with underlying probability p, the relative proportion of training samples for which the value $(p - \hat{p})$ exceeds ϵ is at most[3] $e^{-2m\epsilon^2}$. The second and third bounds make similar statements. As an example, take $m = 1000$, and $\epsilon = 0.05$. Then $e^{-2m\epsilon^2} = e^{-5} \approx 1/148$. The first bound implies that if we repeatedly take samples of size 1000, and take the estimate $\hat{p}$ to be the relative number of heads in that sample, then for (roughly) 147 out of every 148 samples the value of $(p - \hat{p})$ will be less than 0.05. The second bound says that for roughly 147 out of every 148 samples the value of $(\hat{p} - p)$ will be less than 0.05, and the last bound says that for 146 out of every 148 samples the absolute value $|p - \hat{p}|$ will be less than 0.05. Roughly speaking, if we draw a sample of size 1000, we would be quite unlucky for the relative frequency estimate to diverge from the true probability

p by more than 0.05. It is always *possible* for $\hat{p}$ to diverge substantially from p – it is possible to draw an extremely unrepresentative training sample, such as a sample of all heads when $p = 0.7$, for example – but as the sample size is increased the chances of us being this unlucky become increasingly unlikely.

A second useful result is the *Union Bound*:

Theorem 2 *(**Union Bound**). For any n events $\{A_1, A_2, \ldots, A_n\}$, and for any distribution P whose sample space includes all A_i,*

$$P[A_1 \cup A_2 \cup \cdots \cup A_{n-1} \cup A_n] \leq \sum_i P[A_i] \tag{1.5}$$

Here we use the notation $P[A \cup B]$ to mean the probability of A or B occurring. The Union Bound follows directly from the axioms of probability theory. For example, if $n = 2$, then $P[A_1 \cup A_2] = P[A_1] + P[A_2] - P[A_1 A_2] \leq P[A_1] + P[A_2]$, where $P[A_1 A_2]$ means the probability of both A_1 and A_2 occurring. The more general result for all n follows by induction on n.

We are now in a position to apply these results to learning problems. First, consider just a single member of $\mathcal{H}$, a function h. Say we draw a training sample $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ from some unknown distribution $D(x, y)$. We can calculate the relative frequency of errors of h on this sample,

$$\hat{Er}(h) = \frac{1}{m} \sum_i [[h(x_i) \neq y_i]]$$

where $[[\pi]]$ is 1 if π is true, 0 otherwise. We are interested in how this quantity is related to the true error-rate of h on the distribution D, that is $Er(h) = \sum_{x,y} D(x, y)[[h(x) \neq y]]$. We can apply the first Chernoff bound directly to this problem to give for all $\epsilon > 0$

$$P[Er(h) > \hat{Er}(h) + \epsilon] \leq e^{-2m\epsilon^2} \tag{1.6}$$

So for any single member of $\mathcal{H}$, the Chernoff bound describes how its observed error on the training set is related to its true probability of error. Now consider the entire set of hypotheses $\mathcal{H}$. Say we assign an arbitrary ordering to the $n = |\mathcal{H}|$ hypotheses, so that $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$. Consider the probability of *any one* of the hypotheses h_i having its estimated loss $\hat{Er}(h_i)$ diverge by more than ϵ from its expected loss $Er(h_i)$. This probability is

$$P\left[\left(Er(h_1) > \hat{Er}(h_1) + \epsilon\right) \cup \left(Er(h_2) > \hat{Er}(h_2) + \epsilon\right) \cup \cdots \cup \left(Er(h_n) > \hat{Er}(h_n) + \epsilon\right)\right]$$

By application of the union bound, and the result in equation (1.6), we get the following bound on this probability

$$P\left[\left(Er(h_1) > \hat{Er}(h_1) + \epsilon\right) \cup \left(Er(h_2) > \hat{Er}(h_2) + \epsilon\right) \cdots\right]$$

$$\begin{aligned} &\leq \sum_{i=1}^{|\mathcal{H}|} P\left[Er(h_i) > \hat{E}r(h_i) + \epsilon\right] \\ &\leq |\mathcal{H}|e^{-2m\epsilon^2} \end{aligned}$$

It is useful to rephrase this result by introducing a variable $\delta = |\mathcal{H}|e^{-2m\epsilon^2}$, and solving in terms of ϵ, which gives $\epsilon = \sqrt{(\log|\mathcal{H}| + \log(1/\delta))\,/2m}$. We then have the following theorem:

Theorem 3 *For any distribution $D(x, y)$ generating training and test instances, with probability at least $1 - \delta$ over the choice of training set of size m drawn from D, for all $h \in \mathcal{H}$,*

$$Er(h) \leq \hat{E}r(h) + \sqrt{\frac{\log|\mathcal{H}| + \log\frac{1}{\delta}}{2m}}$$

Thus for all hypotheses h in the set $\mathcal{H}$, $\hat{E}r(h)$ converges to $Er(h)$ as the sample size m goes to infinity. This result is known as a *Uniform Convergence Result*, in that it describes how a whole set of empirical error rates converge to their respective expected errors. Note that this result holds for the hypothesis with minimum error on the training sample. It can be shown that this implies that the ERM method for finite hypothesis spaces – choosing the hypothesis $\hat{h}$ which has minimum error on the training sample – is consistent, in that in the limit as $m \to \infty$, the error of $\hat{h}$ converges to the error of the minimum error hypothesis.

Another important result is how the rate of convergence depends on the size of the hypothesis space. Qualitatively, the bound implies that to avoid overtraining the number of training samples should scale with $\log|\mathcal{H}|$.

5.1 Structural Risk Minimization over Finite Hypothesis Spaces

Ideally, we would like a learning method to have expected error that is close to the loss of the bayes-optimal hypothesis h_B. Now consider the ERM method. It is useful to write the difference from h_B in the form

$$Er(\hat{h}) - Er(h_B) = \left(Er(\hat{h}) - \min_{h\in\mathcal{H}} Er(h)\right) + \left(\min_{h\in\mathcal{H}} Er(h) - Er(h_B)\right)$$

Breaking the error down in this way suggests that there are two components to the difference from the optimal loss $Er(h_B)$. The first term captures the errors due to a finite sample size – if the hypothesis space is too large, then theorem 3 states that there is a good chance that the ERM method will pick a hypothesis that is far from the best in the hypothesis space, and the first term will be large. Thus the first term indicates a pressure to keep $\mathcal{H}$ small, so that there is a good chance of finding the best hypothesis in the set. In contrast, the second term reflects a pressure to make $\mathcal{H}$ large, so that there is a good chance that at least one of the hypotheses is close to the Bayes optimal hypothesis. The two terms can be thought of as being analogues to the familiar "bias–variance" trade-off, the first term being a variance term, the second being the bias.

In this section we describe a method which explicitly attempts to model the trade-off between these two types of errors. Rather than picking a single hypothesis class, Structural Risk Minimization (Vapnik, 1998) advocates picking a set of hypothesis classes $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_s$ of increasing size (i.e., such that $|\mathcal{H}_1| < |\mathcal{H}_2| < \cdots < |\mathcal{H}_s|$). The following theorem then applies (it is an extension of theorem 3, and is derived in a similar way through application of the Chernoff and Union bounds):

Theorem 4 *Assume a set of finite hypothesis classes $\{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_s\}$, and some distribution $D(x, y)$. For all $i = 1, \ldots, s$, for all hypotheses $h \in \mathcal{H}_i$, with probability at least $1 - \delta$ over the choice of training set of size m drawn from D,*

$$Er(h) \leq \hat{Er}(h) + \sqrt{\frac{\log|\mathcal{H}_i| + \log\frac{1}{\delta} + \log s}{2m}}$$

This theorem is very similar to theorem 3, except that the second term in the bound now varies depending on which $\mathcal{H}_i$ a function h is drawn from. Note also that we pay an extra price of $\log(s)$ for our hedging over which of the hypothesis spaces the function is drawn from. The SRM principle is then as follows:

1 Pick a set of hypothesis classes, $\mathcal{H}_i$ for $i = 1, \ldots, s$, of increasing size. This must be done independently of the training data for the above bound to apply.

2 Choose the hypothesis h which minimizes the bound in theorem 4.

Thus rather than simply choosing the hypothesis with the lowest error on the training sample, there is now a trade-off between training error and the size of the hypothesis space of which h is a member. The SRM method advocates picking a compromise between keeping the number of training errors small versus keeping the size of the hypothesis class small.

Note that this approach has a somewhat similar flavour to Bayesian approaches. The Maximum A-Posteriori (MAP) estimates in a Bayesian approach involve choosing the parameters which maximize a combination of the data likelihood and a prior over the parameter values,

$$\Theta_{MAP} = \arg\max_{\Theta} (\log P(\text{data} \mid \Theta) + \log P(\Theta))$$

The first term is a measure of how well the parameters Θ fit the data. The second term is a prior which can be interpreted as a term which penalizes more complex parameter settings. The SRM approach in our example implies choosing the hypothesis that minimizes the bound in theorem 4, i.e.,

$$h_{SRM} = \arg\min_h \left(\hat{Er}(h) + \sqrt{\frac{\log|\mathcal{H}_i| + \log\frac{1}{\delta} + \log s}{2m}} \right)$$

where $|\mathcal{H}_i|$ is the size of the hypothesis class containing h. The function indicating the "goodness" of a hypothesis h again has two terms, one measuring how well the hypothesis fits the data, the second penalizing hypotheses which are too "complex". Here complexity has a very specific meaning: it is a direct measure of how quickly the training data error $\hat{Er}(h)$ converges to its true value $Er(h)$.

6. Convergence Bounds for Hyperplane Classifiers

This section describes analysis applied for binary classifiers, where the set $\mathcal{Y} = \{-1, +1\}$. We consider hyperplane classifiers, where a linear separator in some feature space is used to separate examples into the two classes. This section describes uniform convergence bounds for hyperplane classifiers. Algorithms which explicitly minimize these bounds – namely the Support Vector Machine and Boosting algorithms – are described in section 8.

There has been a large amount of research devoted to the analysis of hyperplane classifiers. They go back to one of the earliest learning algorithms, the Perceptron algorithm (Rosenblatt, 1958). They are similar to the linear models for parsing we proposed in section 2 (in fact the framework of section 2 can be viewed as a generalization of hyperplane classifiers). We will initially review some results applying to linear classifiers, and then discuss how various results may be applied to linear models for parsing.

We will discuss a hypothesis space of n-dimensional hyperplane classifiers, defined as follows:

- Each instance x is represented as a vector $\Phi(x)$ in $\Re^n$.
- For given parameter values $\Theta \in \Re^n$ and a bias parameter $b \in \Re$, the output of the classifier is

  $$h_{\Theta,b}(x) = \text{sign}\left(\Phi(x) \cdot \Theta + b\right)$$

  where sign(z) is $+1$ if $z \geq 0$, -1 otherwise. There is a clear geometric interpretation of this classifier. The points $\Phi(x)$ are in n-dimensional Euclidean space. The parameters Θ, b define a hyperplane through the space, the hyperplane being the set of points z such that $(z \cdot \Theta + b) = 0$. This is a hyperplane with normal Θ, at distance $b/||\Theta||$ from the origin, where $||\Theta||$ is the Euclidean norm, $\sqrt{\sum_j \Theta_j^2}$. This hyperplane is used to classify points: all points falling on one side of the hyperplane are classified as $+1$, points on the other side are classified as -1.

- The hypothesis space is the set of all hyperplanes,

$$\mathcal{H} = \{h_{\Theta,b} : \Theta \in \Re^n, b \in \Re\}$$

It can be shown that the ERM method is consistent for hyperplanes, through a method called VC analysis (Vapnik and Chervonenkis, 1971). We will not go into details here, but roughly speaking, the VC-dimension of a hypothesis space is a measure of its size or complexity. A set of hyperplanes in $\Re^n$ has VC dimension of $(n + 1)$. For any hypothesis space with finite VC dimension the ERM method is consistent.

An alternative to VC-analysis is to analyse hyperplanes through properties of "margins" on training examples. For any hyperplane defined by parameters (Θ, b), for a training sample $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, the *margin* on the i-th training example is defined as

$$\gamma^i_{\Theta,b} = \frac{y_i\,(\Phi(x_i) \cdot \Theta + b)}{||\Theta||} \tag{1.7}$$

where $||\Theta||$ is again the Euclidean norm. Note that if $\gamma^i_{\Theta,b}$ is positive, then the i-th training example is classified correctly by $h_{\Theta,b}$ (i.e., y_i and $(\Phi(x_i) \cdot \Theta + b)$ agree in sign). It can be verified that the absolute value of $\gamma^i_{\Theta,b}$ has a simple geometric interpretation: it is the distance of the point $\Phi(x_i)$ from the hyperplane defined by (Θ, b). If $\gamma^i_{\Theta,b}$ is much greater than 0, then intuitively the i-th training example has been classified correctly and with *high confidence*.

Now consider the special case where the data is *separable* – there is at least one hyperplane which achieves 0 training errors. We define the margin of a hyperplane (Θ, b) on the training sample as

$$\gamma_{\Theta,b} = \min_i \gamma^i_{\Theta,b} \tag{1.8}$$

The margin $\gamma_{\Theta,b}$ has a simple geometric interpretation: it is the minimum distance of any training point to the hyperplane defined by Θ, b. The following theorem then holds:

Theorem 5 *(Shawe-Taylor et al. 1998). Assume the hypothesis class $\mathcal{H}$ is a set of hyperplanes, and that there is some distribution $D(x, y)$ generating examples. Define R to be a constant such that $\forall x, ||\Phi(x)|| \leq R$. For all*

$h_{\Theta,b} \in \mathcal{H}$ with zero error on the training sample, with probability at least $1-\delta$ over the choice of training set of size m drawn from D,

$$Er(h_{\Theta,b}) \leq \frac{c}{m}\left(\frac{R^2}{\gamma_{\Theta,b}^2}\log^2 m + \log\frac{1}{\delta}\right)$$

where c is a constant.

The bound is minimized for the hyperplane with maximum margin (i.e., maximum value for $\gamma_{\Theta,b}$) on the training sample. This bound suggests that if the training data is separable, the hyperplane with maximum margin should be chosen as the hypothesis with the best bound on its expected error. It can be shown that the maximum margin hyperplane is unique, and can be found efficiently using algorithms described in section 8.1. Search for the maximum-margin hyperplane is the basis of "Support Vector Machines" (hard-margin version; Vapnik, 1998).

The previous theorem does not apply when the training data cannot be classified with 0 errors by a hyperplane. There is, however, a similar theorem that can be applied in the non-separable case. First, define $\hat{L}(\Theta, b, \gamma)$ to be the proportion of examples on training data with margin less than γ for the hyperplane $h_{\Theta,b}$:

$$\hat{L}(\Theta, b, \gamma) = \frac{1}{m}\sum_i \left[\left[\gamma_{\Theta,b}^i < \gamma\right]\right] \tag{1.9}$$

The following theorem can now be stated:

Theorem 6 *Cristianini and Shawe-Taylor, 2000, Theorem 4.19. Assume the hypothesis class $\mathcal{H}$ is a set of hyperplanes, and that there is some distribution $D(x, y)$ generating examples. Let R be a constant such that $\forall x, ||\Phi(x)|| \leq R$. For all $h_{\Theta,b} \in \mathcal{H}$, for all $\gamma > 0$, with probability at least $1-\delta$ over the choice of training set of size m drawn from D,*

$$Er(h_{\Theta,b}) \leq \hat{L}(\Theta, b, \gamma) + \sqrt{\frac{c}{m}\left(\frac{R^2}{\gamma^2}\log^2 m + \log\frac{1}{\delta}\right)}$$

where c is a constant.

(The first result of the form of Theorem 6 was given in (Bartlett 1998). This was a general result for large margin classifiers; the immediate corollary that implies the above theorem was given in (Anthony and Bartlett 1999). Note that Zhang (2002) proves a related theorem where the $\log^2 m$ factor is replaced by $\log m$. Note also that the square-root in the second term of theorem 6 means that this bound is in general a looser bound than the bound in theorem 5. This is one cost of moving to the case where some training samples are misclassified, or where some training samples are classified with a small margin.)

This result is important in cases where a large proportion of training samples can be classified with relatively large margin, but a relatively small number of outliers make the problem inseparable, or force a small margin. The result suggests that in some cases a few examples are worth "giving up on", resulting in the first term in the bound being larger than 0, but the second term being much smaller due to a larger value for γ. The *soft margin* version of Support Vector Machines (Cortes and Vapnik, 1995), described in section 8.1, attempts to explicitly manage the trade-off between the two terms in the bound.

A similar bound, due to (Schapire et al., 1998), involves a margin definition which depends on the 1-norm rather than the 2-norm of the parameters Θ ($||\Theta||_1$ is the 1-norm, $\sum_j |\Theta_j|$):

$$\hat{L}_1(\Theta, b, \gamma) = \frac{1}{m} \sum_i \left[\left[\frac{y_i \left(\Phi(x_i) \cdot \Theta + b\right)}{||\Theta||_1} < \gamma \right]\right] \tag{1.10}$$

Theorem 7 *(Schapire et al., 1998). Assume the hypothesis class $\mathcal{H}$ is a set of hyperplanes in $\Re^n$, and that there is some distribution $D(x, y)$ generating examples. For all $h_{\Theta,b} \in \mathcal{H}$, for all $\gamma > 0$, with probability at least $1 - \delta$ over the choice of training set of size m drawn from D,*

$$Er(h_{\Theta,b}) \leq \hat{L}_1(\Theta, b, \gamma) + O\left(\sqrt{\frac{1}{m}\left(\frac{R_\infty^2 \log m \log n}{\gamma^2} + \log\frac{1}{\delta}\right)}\right)$$

where R_∞ is a constant such that $\forall x, ||\Phi(x)||_\infty \leq R_\infty$. ($||\Phi(x)||_\infty$ is the infinity norm, $||\Phi(x)||_\infty = \max_i |\Phi(x)_i|$.)

This bound suggests a strategy that keeps the 1-norm of the parameters low, while trying to classify as many of the training examples as possible with large margin. It can be shown that the AdaBoost algorithm (Freund and Schapire, 1997) is an effective way of achieving this goal; its application to parsing is described in section 8.2.

7. Application of Margin Analysis to Parsing

We now consider how the theory for hyperplane classifiers might apply to the linear models for parsing described in section 2. Recall that given parameters $\Theta \in \Re^n$, the hypothesis h_Θ is defined as

$$h_\Theta(x) = \arg \max_{y \in \mathbf{GEN}(x)} \Phi(x, y) \cdot \Theta \tag{1.11}$$

and the hypothesis class $\mathcal{H}$ is the set of all such functions,

$$\mathcal{H} = \{h_\Theta \ : \ \Theta \in \Re^n\} \tag{1.12}$$

The method for converting parsing to a margin-based problem is similar to the method for ranking problems described in (Freund et al., 1998), and to the approach to multi-class classification problems in (Schapire et al., 1998; Crammer and Singer, 2001; Elisseeff et al., 1999). As a first step, we give a definition of the margins on training examples. Assume we have a training sample $\{(x_1, y_1), \ldots, (x_m, y_m)\}$. We define the margin on the i-th training example with parameter values Θ as

$$\gamma_{\Theta}^{i} = \frac{1}{||\Theta||}\left(\Phi(x_i, y_i) \cdot \Theta - \max_{y \in \mathbf{GEN}(x_i), y \neq y_i} \Phi(x_i, y) \cdot \Theta\right) \tag{1.13}$$

The margin on the i-th example is now the difference in scores between the correct tree for the i-th sentence and the highest scoring incorrect tree for that sentence. Notice that this has very similar properties to the value $\gamma_{\Theta,b}^{i}$ defined for hyperplanes in equation (1.7). If $\gamma_{\Theta}^{i} > 0$, then h_{Θ} gives the correct output on the i-th example. The larger the value of γ_{Θ}^{i}, the more "confident" we can take this prediction to be.

We can now make a very similar definition to that in equation (1.9):

$$\hat{L}(\Theta, \gamma) = \frac{1}{m} \sum_{i} \left[\left[\gamma_{\Theta}^{i} < \gamma\right]\right] \tag{1.14}$$

So $\hat{L}(\Theta, \gamma)$ tracks the proportion of training examples with margin less than γ. A similar theorem to theorem 6 can be stated:

Theorem 8 *Assume the hypothesis class $\mathcal{H}$ is a set of linear models as defined in equation (1.11) and equation (1.12), and that there is some distribution $D(x, y)$ generating examples. For all $h_{\Theta} \in \mathcal{H}$, for all $\gamma > 0$, with probability at least $1 - \delta$ over the choice of training set of size m drawn from D,*

$$Er(h_{\Theta}) \leq \hat{L}(\Theta, \gamma) + O\left(\sqrt{\frac{1}{m}\left(\frac{R^2}{\gamma^2}(\log m + \log N) + \log \frac{1}{\delta}\right)}\right)$$

where R is a constant such that $\forall x \in \mathcal{X}, \forall y \in \mathbf{GEN}(x), \forall z \in \mathbf{GEN}(x)$, $||\Phi(x, y) - \Phi(x, z)|| \leq R$. The variable N is the smallest positive integer such that $\forall x \in \mathcal{X}, |\mathbf{GEN}(x)| - 1 \leq N$.

Proof: The proof follows from results in (Zhang, 2002). See the appendix of this chapter for the proof.

Note that this is similar to the bound in theorem 6. A difference, however, is the dependence on N, a bound on the number of candidates for any example. Even though this term is logarithmic, the dependence is problematic because the number of candidate parses for a sentence will usually have an exponential dependence on the length of the sentence, leading to $\log N$ having linear dependence on the maximum sentence length. (For example, the number of labeled binary-branching trees for a sentence of length n, with G non-terminals, is $\frac{G^n(2n)!}{(n+1)!n!}$, the log of this number is $O(n \log G + n \log n)$.) It is an open problem whether tighter bounds – in particular, bounds which do not depend on N – can be proved. Curiously, we show in section 8.3 that the perceptron algorithm leads to a margin-based learning bound that is independent of the value for N. This suggests that it may be possible to prove tighter bounds than those in theorem 8.

Not surprisingly, a theorem based on 1-norm margins, which is similar to theorem 7, also holds. We first give a definition based on 1-norm margins:

$$\hat{L}_1(\Theta, \gamma) = \frac{1}{m} \sum_i \left[\left[\gamma_\Theta^i < \gamma\right]\right] \tag{1.15}$$

where γ_Θ^i now depends on $||\Theta||_1$:

$$\gamma_\Theta^i = \frac{1}{||\Theta||_1} \left(\Phi(x_i, y_i) \cdot \Theta - \max_{y \in \mathbf{GEN}(x_i), y \neq y_i} \Phi(x_i, y) \cdot \Theta \right) \tag{1.16}$$

The following theorem then holds:

Theorem 9 *Assume the hypothesis class $\mathcal{H}$ is a set of linear models as defined in equation (1.11) and equation (1.12), and that there is some distribution $D(x, y)$ generating examples. For all $h_\Theta \in \mathcal{H}$, for all $\gamma > 0$, with probability at least $1 - \delta$ over the choice of training set of size m drawn from D,*

$$Er(h_\Theta) \leq \hat{L}_1(\Theta, \gamma) + O\left(\sqrt{\frac{1}{m} \left(\frac{R_\infty^2 (\log m + \log N) \log n}{\gamma^2} + \log \frac{1}{\delta} \right)} \right)$$

where R_∞ is a constant such that $\forall x \in \mathcal{X}, \forall y \in \mathbf{GEN}(x), \forall z \in \mathbf{GEN}(x)$, $||\Phi(x, y) - \Phi(x, z)||_\infty \leq R_\infty$. The variable N is the smallest positive integer such that $\forall x \in \mathcal{X}, |\mathbf{GEN}(x)| - 1 \leq N$.

Proof: The proof for the multi-class case, given in (Schapire et al., 1998), essentially implies this theorem. A different proof also follows from results in (Zhang, 2002) – see the appendix of this chapter for the proof.

The bounds in theorems 8 and 9 suggested a trade-off between keeping the values for $\hat{L}(\Theta, \gamma)$ and $\hat{L}_1(\Theta, \gamma)$ low and keeping the value of γ high. The algorithms described in section 8 attempt to find a hypothesis Θ which can achieve low values for these quantities with a high value for γ. The algorithms are direct modifications of algorithms for learning hyperplane classifiers for binary classification: these classification algorithms are motivated by the bounds in theorems 6 and 7.

8. Algorithms

In this section we describe parameter estimation algorithms which are motivated by the generalization bounds for linear models in section 7 of this paper. The first set of algorithms, support vector machines, use constrained optimization problems that are related to the bounds in theorems 8 and 9. The second algorithm we describe is a modification of AdaBoost (Freund and Schapire, 1997), which is motivated by the bound in theorem 9. Finally, we describe a variant of the perceptron algorithm applied to parsing. The perceptron algorithm does not explicitly attempt to optimize the generalization bounds in section 7, but its convergence and generalization properties can be shown to be dependent on the existence of parameter values which separate the training data with large margin under the 2-norm. In this sense they are a close relative to support vector machines.

8.1 Support Vector Machines

We now describe an algorithm which is motivated by the bound in theorem 8. First, recall the definition of the margin for the parameter values Θ on the i-th training example,

$$\gamma_\Theta^i = \frac{1}{||\Theta||}\left(\Phi(x_i, y_i)\cdot\Theta - \max_{y\in\mathbf{GEN}(x_i), y\neq y_i}\Phi(x_i, y)\cdot\Theta\right) \tag{1.17}$$

We will also define the margin for parameter values Θ on the entire training sample as

$$\gamma_\Theta = \min_i \;\; \gamma_\Theta^i \tag{1.18}$$

If the data is separable (i.e., there exists some Θ such that $\gamma_\Theta > 0$), then of all hyperplanes which have zero training errors, the "best" hyperplane by the bound of theorem 8 is the hyperplane Θ^* with maximum margin on the training sample,

$$\Theta^* = \arg\max_{\Theta\in\Re^n}\gamma_\Theta \tag{1.19}$$

This hyperplane minimizes the bound in theorem 8 subject to the constraint that $\hat{L}(\Theta, \gamma)$ is 0.

Vapnik (1998) shows that the hyperplane Θ^* is unique[4], and gives a method for finding Θ^*. The method involves solving the following constrained optimization problem:

Minimize

$$||\Theta||^2$$

subject to the constraints

$$\forall i, \forall y\in\mathbf{GEN}(x_i), y\neq y_i, \quad \Theta\cdot\Phi(x_i, y_i) - \Theta\cdot\Phi(x_i, y) \geq 1$$

Any hyperplane Θ satisfying these constraints separates the data with margin $\gamma_\Theta = 1/||\Theta||$. By minimizing $||\Theta||^2$ (or equivalently $||\Theta||$) subject to the constraints, the method finds the parameters Θ with maximal value for γ_Θ.

Simply finding the maximum-margin hyperplane may not be optimal or even possible: the data may not be separable, or the data may be noisy. The bound in theorem 8 suggests giving up on some training examples which may be difficult or impossible to separate. (Cortes and Vapnik, 1995) suggest a refined optimization task for the classification case which addresses this problem; we suggest the following modified optimization problem as a natural analogue of this approach (our approach is similar to the method for multi-class classification problems in Crammer and Singer, 2001):

Minimize

$$||\Theta||^2 + C \sum \epsilon_i$$

with respect to Θ, ϵ_i for $i = 1, \ldots, m$,
subject to the constraints

$$\forall i, \forall y \in \mathbf{GEN}(x_i), y \neq y_i, \quad \Theta \cdot \Phi(x_i, y_i) - \Theta \cdot \Phi(x_i, y) \geq 1 - \epsilon_i$$

$$\forall i, \quad \epsilon_i \geq 0$$

Here we have introduced a "slack variable" ϵ_i for each training example. At the solution of the optimization problem, the margin on the i-th training example is at least $(1-\epsilon_i)/||\Theta||$. On many examples the slack variable ϵ_i will be zero, and the margin γ_Θ^i will be at least $1/||\Theta||$. On some examples the slack variable ϵ_i will be positive, implying that the algorithm has "given up" on separating the example with margin $1/||\Theta||$. The constant C controls the cost for having non-zero values of ϵ_i. As $C \to \infty$, the problem becomes the same as the hard-margin SVM problem, and the method attempts to find a hyperplane which correctly separates all examples with margin at least $1/||\Theta||$ (i.e., all slack variables are 0). For smaller C, the training algorithm may "give up" on some examples (i.e., set $\epsilon_i > 0$) in order to keep $||\Theta||^2$ low. Thus by varying C, the method effectively modifies the trade-off between the two terms in the bound in theorem 8. In practice, a common approach is to train the model for several values of C, and then to pick the classifier which has best performance on some held-out set of development data.

Both kinds of SVM optimization problem outlined above have been studied extensively (e.g., see Joachims, 1998; Platt, 1998) and can be solved relatively efficiently. (A package for SVMs, written by Thorsten Joachims, is available from `http://ais.gmd.de/~thorsten/svm_light/`.)

A closely related approach which is based on 1-norm margins – the bound in theorem 9 – is as follows:

Input: Examples $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, Grammar G, representation $\Phi : \mathcal{X} \times \mathcal{Y} \rightarrow \Re^n$ such that $\forall (x, y_1, y_2) \in \mathcal{T}$, where $\mathcal{T}$ is defined below, for $s = 1, \ldots, n$, $-1 \leq (\Phi_s(x, y_1) - \Phi_s(x, y_2)) \leq 1$

Algorithm:

- Define the set of triples $\mathcal{T} = \{(x_i, y_i, y) : i = 1, \ldots, m, y \in \mathbf{GEN}(x_i) \text{ s.t. } y \neq y_i\}$
- Set initial parameter values $\Theta = 0$
- For $t = 1$ to T
  - Define a distribution over the training sample $\mathcal{T}$ as

$$\forall (x, y_1, y_2) \in \mathcal{T}, \quad D^t(x, y_1, y_2) = \frac{1}{Z^t} \frac{e^{-\Theta \cdot (\Phi(x, y_1) - \Phi(x, y_2))}}{|\mathbf{GEN}(x)| - 1}$$

  where $Z^t = \sum_{(x, y_1, y_2) \in \mathcal{T}} e^{-\Theta \cdot (\Phi(x, y_1) - \Phi(x, y_2))} / (|\mathbf{GEN}(x)| - 1)$.
  - For $s = 1$ to n calculate $r_s = \sum_{(x, y_1, y_2) \in \mathcal{T}} D^t(x, y_1, y_2) \left(\Phi_s(x, y_1) - \Phi_s(x, y_2)\right)$
  - Choose $s_t = \arg\max_s |r_s|$
  - Update single parameter $\Theta_{s_t} = \Theta_{s_t} + \frac{1}{2} \log \left(\frac{1 + r_{s_t}}{1 - r_{s_t}}\right)$

Figure 1.2. The AdaBoost algorithm applied to parsing.

Minimize

$$||\Theta||_1 + C \sum \epsilon_i$$

with respect to Θ, ϵ_i for $i = 1, \ldots, m$,
subject to the constraints

$$\forall i, \forall y \in \mathbf{GEN}(x_i), y \neq y_i, \quad \Theta \cdot \Phi(x_i, y_i) - \Theta \cdot \Phi(x_i, y) \geq 1 - \epsilon_i$$

$$\forall i, \quad \epsilon_i \geq 0$$

This can be framed as a linear programming problem. See (Demiriz et al., 2001) for details, and the relationships between linear programming approaches and the boosting algorithms described in the next section.

8.2 Boosting

The AdaBoost algorithm (Freund and Schapire, 1997) is one method for optimizing the bound for hyperplane classifiers in theorem 7 (Schapire et al., 1998). This section describes a modified version of AdaBoost, applied to the parsing problem. Figure 1.2 shows the modified algorithm. The algorithm converts the training set into a set of triples:

$$\mathcal{T} = \{(x_i, y_i, y) : i = 1, \ldots, m, y \in \mathbf{GEN}(x_i) \text{ s.t. } y \neq y_i\}$$

Each member (x, y_1, y_2) of $\mathcal{T}$ is a triple such that x is a sentence, y_1 is the correct tree for that sentence, and y_2 is an incorrect tree also proposed by $\mathbf{GEN}(x)$. AdaBoost maintains a distribution D^t over the training examples such that $D^t(x, y_1, y_2)$ is proportional to $\exp\{-\Theta \cdot (\Phi(x, y_1) - \Phi(x, y_2))\}$. Members of $\mathcal{T}$ which are well discriminated by the current parameter values Θ are given low weight by the distribution, whereas examples which are poorly discriminated are weighted more highly.

The value $r_s = \sum_{(x,y_1,y_2)\in\mathcal{T}} D^t(x, y_1, y_2)\,(\Phi_s(x, y_1) - \Phi_s(x, y_2))$ is a measure of how well correlated Φ_s is with the distribution D^t. The magnitude of r_s can be taken as a measure of how correlated $(\Phi_s(x, y_1) - \Phi_s(x, y_2))$ is with the distribution D^t. If it is highly correlated, $|r_s|$ will be large, and the s-th parameter will be useful in driving down the margins on the more highly weighted members of $\mathcal{T}$.

In the classification case, Schapire et al. (1998) show that the AdaBoost algorithm has direct properties in terms of optimizing the value of $\hat{L}_1(\Theta, b, \gamma)$ defined in equation (1.10). Unfortunately it is not possible to show that the algorithm in figure 1.2 has a similar effect on the parsing quantity $\hat{L}_1(\Theta, \gamma)$ in equation (1.15). Instead, we show its effect on a similar quantity[5] $\hat{RL}_1$:

$$\hat{RL}_1(\Theta, \gamma) \quad = \quad \frac{1}{m} \sum_i \frac{1}{|\mathbf{GEN}(x_i)| - 1} \sum_{y \in \mathbf{GEN}(x_i), y \neq y_i} \left[\left[\gamma_\Theta^{i,y} < \gamma\right]\right] \tag{1.20}$$

where

$$\gamma_\Theta^{i,y} = \frac{1}{||\Theta||_1} \left(\Phi(x_i, y_i) \cdot \Theta - \Phi(x_i, y) \cdot \Theta\right)$$

In these definitions $\gamma_\Theta^{i,y}$ for $i = 1, \ldots, m, y \in \mathbf{GEN}(x_i), y \neq y_i$ is the degree to which the correct parse on the i-th sentence is separated from an incorrect parse y. The quantity $\hat{RL}_1(\Theta, \gamma)$ measures the proportion of margins $\gamma_\Theta^{i,y}$ that are at least γ, where the proportions are normalized for the number of candidates on each sentence (ensuring that sentences with very large numbers of candidates do not dominate). It is clear that $\hat{RL}_1$ is related to $\hat{L}_1$ – for example $\hat{RL}_1(\Theta, \gamma)$ is 0 if and only if $\hat{L}_1(\Theta, \gamma)$ is 0 – although they are somewhat different quantities.

There is a strong relation between the values of $|r_s|$, and the effect on the values of $\hat{RL}_1(\Theta, \gamma)$. If we define $\epsilon_t = (1 - |r_{s_t}|)/2$ then the following theorem holds:

Theorem 10 *(Slight modification of theorem 5 of Schapire et al., 1998). If we define $\hat{RL}_1(\Theta, \gamma)$ as in equation (1.20), and the Adaboost algorithm in*

Input: Examples $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, Grammar G, representation $\Phi : \mathcal{X} \times \mathcal{Y} \rightarrow \Re^n$
Algorithm: Initialise parameters Θ to be 0
For t = 1 to T, For i = 1 to m,
Calculate $y = h_\Theta(x_i) = \arg\max_{z \in \mathbf{GEN}(x_i)} \Phi(x_i, z) \cdot \Theta$
If($y = y_i$) then do nothing; else if($y \neq y_i$) then $\Theta = \Theta + \Phi(x_i, y_i) - \Phi(x_i, y)$
Output: Parameter values Θ

Figure 1.3. The perceptron algorithm for parsing. It takes T passes over the training set.

figure 1.2 generates values $\epsilon_1, \epsilon_2, \ldots, \epsilon_T$, then for all $\gamma > 0$,

$$\hat{RL}_1(\Theta, \gamma) \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t^{1-\gamma}(1-\epsilon_t)^{1+\gamma}}$$

Schapire et al. (1998) point out that if for all $t = 1, \ldots, T$, $\epsilon_t \leq 1/2 - \delta$ (i.e., $|r_{s_t}| \geq 2\delta$) for some $\delta > 0$, then the theorem implies that

$$\hat{RL}_1(\Theta, \gamma) \leq \left(\sqrt{(1-2\delta)^{1-\gamma}(1+2\delta)^{1+\gamma}}\right)^T = f(\delta, \gamma)^T$$

It can be shown that $f(\delta, \gamma)$ is less than one providing that $\gamma < \delta$: the implication is that for all $\gamma < \delta$, $\hat{RL}_1(\Theta, \gamma)$ decreases exponentially in the number of iterations, T. So if the AdaBoost algorithm can successfully maintain high values of $|r_{s_t}|$ for several iterations, it will be successful at minimizing $\hat{RL}_1(\Theta, \gamma)$ for a relatively large range of γ. Given that $\hat{RL}_1$ is related to $\hat{L}_1$, we can view this as an approximate method for optimizing the bound in theorem 9. In practice, a set of held-out data is usually used to optimize T, the number of rounds of boosting.

The algorithm states a restriction on the representation Φ. For all members (x, y_1, y_2) of $\mathcal{T}$, for $s = 1, \ldots, n$, $(\Phi_s(x, y_1) - \Phi_s(x, y_2))$ must be in the range -1 to $+1$. This is not as restrictive as it might seem. If Φ is always strictly positive, it can be rescaled so that its components are always between 0 and $+1$. If some components may be negative, it suffices to rescale the components so that they are always between -0.5 and $+0.5$. A common use of the algorithm, as applied in (Collins, 2000), is to have the n components of Φ to be the values of n indicator functions, in which case all values of Φ are either 0 or 1, and the condition is satisfied.

8.3 A Variant of the Perceptron Algorithm

The final parameter estimation algorithm which we will describe is a variant of the perceptron algorithm, as introduced by (Rosenblatt, 1958). Figure 1.3 shows the algorithm. Note that the main computational expense is in calculating $y = h_\Theta(x_i)$ for each example in turn. For weighted context-free grammars this step can be achieved in polynomial time using the CKY parsing algorithm. Other representations may have to rely on explicitly calculating $\Phi(x_i, z) \cdot \Theta$ for all $z \in \mathbf{GEN}(x_i)$, and hence depend computationally on the number of candidates $|\mathbf{GEN}(x_i)|$ for $i = 1, \ldots, m$.

It is useful to define the maximum-achievable margin γ on a separable training set as follows. Recall the definition of the maximum margin hyperplane in equation (1.19),

$$\Theta^* = \arg \max_{\Theta \in \Re^n} \gamma_\Theta$$

Then we define the maximum achievable margin as

$$\gamma = \gamma_{\Theta^*} = \max_{\Theta} \gamma_\Theta$$

It can then be shown that the number of mistakes made by the perceptron algorithm in figure 1.3 depends directly on the value of γ:

Theorem 11 *Let $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ be a sequence of examples such that $\forall i,\ \forall y \in \mathbf{GEN}(x_i),\ ||\Phi(x_i, y_i) - \Phi(x_i, y)|| \leq R$. Assume the sequence is separable, and take γ to be the maximum achievable margin on the sequence. Then the number of mistakes made by the perceptron algorithm on this sequence is at most $(R/\gamma)^2$.*

Proof: See (Collins, 2002b) for a proof. The proof is a simple modification of the proof for hyperplane classifiers (Block, 1962; Novikoff, 1962, see also Freund and Schapire, 1999).

This theorem implies that if the training sample in figure 1.3 is separable, and we iterate the algorithm repeatedly over the training sample, then the algorithm converges to a parameter setting that classifies the training set with zero errors. (In particular, we need at most $(R/\gamma)^2$ passes over the training sample before convergence.) Thus we now have an algorithm for training weighted context-free grammars which will find a zero error hypothesis if it exists. For example, the algorithm would find a weighted grammar with zero expected error on the example problem in section 3.

Of course convergence to a zero-error hypothesis on training data says little about how well the method generalizes to new test examples. Fortunately a second theorem gives a bound on the generalization error of the perceptron method:

Theorem 12 *(Direct consequence of the sample compression bound in (Littlestone and Warmuth, 1986); see also theorem 4.25, page 70, Cristianini and Shawe-Taylor, 2000). Say the perceptron algorithm makes d mistakes when run to convergence over a training set of size m. Then for all distributions $D(x, y)$, with probability at least $1 - \delta$ over the choice of training set of size*

m drawn from D, if h_Θ is the hypothesis at convergence,

$$Er(h_\Theta) \leq \frac{1}{m-d}\left(d \log \frac{em}{d} + \log m + \log \frac{1}{\delta}\right)$$

Given that $d \leq (R/\gamma)^2$, this bound states that if the problem is separable with large margin – i.e., the ratio R/γ is relatively small – then the perceptron will converge to a hypothesis with good expected error with a reasonable number of training examples.

The perceptron algorithm is remarkable in a few respects. First, the algorithm in figure 1.3 can be efficient even in cases where $\mathbf{GEN}(x)$ is of exponential size in terms of the input x, providing that the highest scoring structure can be found efficiently for each training example. For example, finding the $\arg\max$ can be achieved in polynomial time for context-free grammars, so they can be trained efficiently using the algorithm. This is in contrast to the support vector machine and boosting algorithms, where we are not aware of algorithms whose computational complexity does not depend on the size of $\mathbf{GEN}(x_i)$ for $i = 1, \ldots, n$. Second, the convergence properties (number of updates) of the algorithm are also independent of the size of $\mathbf{GEN}(x_i)$ for $i = 1, \ldots, n$, depending on the maximum achievable margin γ on the training set. Third, the generalization theorem (theorem 12) shows that the generalization properties are again independent of the size of each $\mathbf{GEN}(x_i)$, depending only on γ. This is in contrast to the bounds in theorems 8 and 9, which depended on N, a bound on the number of candidates for any input.

The theorems quoted here do not treat the case where the data is not separable, but results for the perceptron algorithm can also be derived in this case. See (Freund and Schapire, 1999) for analysis of the classification case, and see (Collins, 2002b) for how these results can be carried over to problems such as parsing. Collins (2002b) shows how the perceptron algorithm can be applied to tagging problems, with improvements in accuracy over a maximum-entropy tagger on part-of-speech tagging and NP chunking; see this paper for more analysis of the perceptron algorithm, and some modifications to the basic algorithm.

9. Discussion

In this section we give further discussion of the algorithms in this chapter. Section 9.1 describes experimental results using some of the algorithms. Section 9.2 describes relationships to Markov Random Field approaches.

9.1 Experimental Results

There are several papers describing experiments on NLP tasks using the algorithms described in this paper. Collins (2000) describes a boosting method which is related to the algorithm in figure 1.2. In this case $\mathbf{GEN}(x)$ is the top N most likely parses from the parser of (Collins, 1999). The representation $\Phi(x, y)$ combines the log probability under the initial model, together with a large number of additional indicator functions which are various features of trees. The paper describes a boosting algorithm which is particularly efficient when the features are indicator (binary-valued) functions, and the features are relatively sparse. The method gives a 13% relative reduction in error over the original parser of (Collins, 1999). (See (Ratnaparkhi et al., 1994) for an approach which also uses a N-best output from a baseline model combined with "global" features, but a different algorithm for training the parameters of the model.)

Collins (2002a) describes a similar approach applied to named entity extraction. $\mathbf{GEN}(x)$ is the top 20 most likely hypotheses from a maximum-entropy tagger. The representation again includes the log probability under the original model, together with a large number of indicator functions. The boosting and perceptron algorithms give relative error reductions of 15.6% and 17.7% respectively.

Collins and Duffy (2002) and Collins and Duffy (2001) describe the perceptron algorithm applied to parsing and tagging problems. $\mathbf{GEN}(x)$ is again the top N most likely parses from a baseline model. The particular twist in these papers is that the representation $\Phi(x, y)$ for both the tagging and parsing problems is an extremely high-dimensional representation, which tracks all subtrees in the parsing case (in the same way as the DOP approach to parsing, see Bod, 1998), or all sub-fragments of a tagged sequence. The key to making the method computationally efficient (in spite of the high dimensionality of Φ) is that for any pair of structures (x_1, y_1) and (x_2, y_2) it can be shown that the inner product $\Phi(x_1, y_1) \cdot \Phi(x_2, y_2)$ can be calculated efficiently using dynamic programming. The perceptron algorithm has an efficient "dual" implementation which makes use of inner products between examples – see (Cristianini and Shawe-Taylor, 2000; Collins and Duffy, 2002). Collins and Duffy (2002) show a 5% relative error improvement for parsing, and a more significant 15% relative error improvement on the tagging task.

Collins (2002b) describes perceptron algorithms applied to the tagging task. $\mathbf{GEN}(x)$ for a sentence x of length n is the set of all possible tag sequences of length n (there are T^n such sequences if T is the number of tags). The representation used is similar to the feature-vector representations used in maximum-entropy taggers, as in (Ratnaparkhi, 1996). The highest scoring tagged sequence under this representation can be found efficiently using the perceptron algorithm, so the weights can be trained using the algorithm in figure 1.3 without having to exhaustively enumerate all tagged sequences. The method gives improvements over the maximum-entropy approach: a 12% relative error reduction for part-of-speech tagging, a 5% relative error reduction for noun-phrase chunking.

9.2 Relationship to Markov Random Fields

Another method for training the parameters Θ can be derived from log-linear models, or Markov Random Fields (otherwise known as maximum-entropy models). Several approaches (Ratnaparkhi et al., 1994; Johnson et al., 1999; Lafferty et al., 2001) use the parameters Θ to define a conditional probability distribution over the candidates $y \in \mathbf{GEN}(x)$:

$$P(y \mid x, \Theta) = \frac{e^{\Phi(x,y)\cdot\Theta}}{\sum_{z \in \mathbf{GEN}(x)} e^{\Phi(x,z)\cdot\Theta}} = \frac{1}{1 + \sum_{z \in \mathbf{GEN}(x), z \neq y} e^{\Phi(x,z)\cdot\Theta - \Phi(x,y)\cdot\Theta}} \tag{1.21}$$

Once the model is trained, the output on a new sentence x is the highest probability parse, $\arg\max_{y \in \mathbf{GEN}(x)} P(y \mid x, \Theta) = \arg\max_{y \in \mathbf{GEN}(x)} \Phi(x, y) \cdot \Theta$. So the output under parameters Θ is identical to the method used throughout this paper.

The differences between this method and the approaches advocated in this paper are twofold. First, the statistical justification differs: the log-linear approach is a parametric approach (see section 4.2), explicitly attempting to model the conditional distribution $D(y \mid x)$, and potentially suffering from the problems described in section 4.3.

The second difference concerns the algorithms for training the parameters. In training log-linear models, a first crucial concept is the log-likelihood of the training data,

$$\begin{aligned} \text{L-Loss}(\Theta) &= \sum_i \log p(y_i \mid x_i, \Theta) \\ &= -\sum_i \log \left(1 + \sum_{z \in \mathbf{GEN}(x_i), z \neq y_i} e^{(\Phi(x_i,z)\cdot\Theta - \Phi(x_i,y_i)\cdot\Theta)} \right) \end{aligned}$$

Parameter estimation methods in the MRF framework generally involve maximizing the log-likelihood while controlling for overfitting the training data. A first method for controlling the degree of overfitting, as used in (Ratnaparkhi et al., 1994), is to use feature selection. In this case a greedy method is used to minimize the log likelihood using only a small number of features. It can be shown that the boosting algorithms can be considered to be a feature selection method for minimizing the exponential loss

$$\text{E-Loss}(\Theta) = \sum_i \sum_{z \in \mathbf{GEN}(x_i), z \neq y_i} e^{(\Phi(x_i,z)\cdot\Theta - \Phi(x_i,y_i)\cdot\Theta)} \tag{1.22}$$

The two functions L-Loss and E-Loss look similar, and a number of papers (Friedman et al., 1998; Lafferty, 1999; Collins, Schapire and Singer, 2002;

Lebanon and Lafferty, 2001) have drawn connections between the two objective functions, and algorithms for optimizing them. One result from (Collins, Schapire and Singer, 2002) shows that there is a trivial change to the algorithm in figure 1.2 which results in the method provably optimizing the objective function L-Loss. The change is to redefine D^t as $D^t(x, y_1, y_2) = p(y_2 \mid x, \Theta)/Z^t$ where Z^t is a normalization term, and $p(y_2 \mid x, \Theta)$ takes the form in equation (1.21).

A second method for controlling overfitting, used in (Johnson et al., 1999; Lafferty et al., 2001), is to use a gaussian prior over the parameters. The method then selects the MAP parameters – the parameters which maximize the objective function

$$\text{L-Loss}(\Theta) - C||\Theta||^2$$

for some constant C which is determined by the variance term in the gaussian prior. This method has at least a superficial similarity to the SVM algorithm in section 8.1 (2-norm case), which also attempts to balance the norm of the parameters versus a function measuring how well the parameters fit the data (i.e., the sum of the slack variable values).

We should stress again, however, that in spite of some similarities between the algorithms for MRFs and the boosting and SVM methods, the statistical justification for the methods differs considerably.

10. Conclusions

This paper has described a number of methods for learning statistical grammars. All of these methods have several components in common: the choice of a grammar which defines the set of candidates for a given sentence, and the choice of representation of parse trees. A score indicating the plausibility of competing parse trees is taken to be a linear model, the result of the inner product between a tree's feature vector and the vector of model parameters. The only respect in which the methods differ is in how the parameter values (the "weights" on different features) are calculated using a training sample as evidence.

Section 4 introduced a framework under which various parameter estimation methods could be studied. This framework included two main components. First, we assume some fixed but unknown distribution over sentence/parse-tree pairs. Both training and test examples are drawn from this distribution. Second, we assume some loss function, which dictates the penalty on test examples for proposing a parse which is incorrect. We focused on a simple loss function, where the loss is 0 if the proposed parse is identical to the correct parse, 1 otherwise. Under these assumptions, the "quality" of a parser is its expected loss (expected error rate) on newly drawn test examples. The goal of learning is to use the training data as evidence for choosing a function which has small expected loss.

A central idea in the analysis of learning algorithms is that of the margins on examples in training data. We described theoretical bounds which motivate approaches which attempt classify a large proportion of examples in training with a large margin. Finally, we described several algorithms which can be used to achieve this goal on the parsing problem.

There are several open problems highlighted in this paper:

- The margin bounds for parsing (theorems 8 and 9) both depend on N, a bound on the number of candidates for any input sentence. It is an open question whether bounds which are independent of N can be proved. The perceptron algorithm in section 8.3 has generalization bounds which are independent of N, suggesting that this might also be possible for the margin bounds.

- The Boosting and Support Vector Machine methods both require enumerating all members of $\mathbf{GEN}(x_i)$ for each training example x_i. The perceptron algorithm avoided this in the case where the highest scoring hypothesis could be calculated efficiently, for example using the CKY algorithm. It would be very useful to derive SVM and boosting algorithms whose computational complexity can be shown to depend on the separation γ rather than the size of $\mathbf{GEN}(x_i)$ for each training example x_i.

- The boosting algorithm in section 8.2 optimized the quantity $\hat{RL}_1$, rather than the desired quantity $\hat{L}_1$. It would be useful to derive a boosting algorithm which provably optimized $\hat{L}_1$.

Acknowledgments

I would like to thank Sanjoy Dasgupta, Yoav Freund, John Langford, David McAllester, Rob Schapire and Yoram Singer for answering many of the questions I have had about the learning theory and algorithms in this paper. Fernando Pereira pointed out several issues concerning analysis of the perceptron algorithm. Thanks also to Nigel Duffy, for many useful discussions while we were collaborating on the use of kernels for parsing problems. I would like to thank Tong Zhang for several useful insights concerning margin-based generalization bounds for multi-class problems. Thanks to Brian Roark for helpful comments on an initial draft of this paper, and to Patrick Haffner for many useful suggestions. Thanks also to Peter Bartlett, for feedback on the paper, and some useful pointers to references.

Appendix: Proof of theorems 8 and 9

The proofs in this section closely follow the framework and results of (Zhang, 2002). The basic idea is to show that the covering number results of (Zhang, 2002) apply to the parsing problem, with the modification that any dependence on m (the sample size) is replaced by a dependence on mN (where N is the smallest integer such that $\forall x, \ \ |\mathbf{GEN}(x)| \leq (N+1)$).

Zhang (2002) takes each sample (x, y) where $x \in \Re^n$, $y \in \{-1, +1\}$ and "folds" the label into the example to create a new sample point $z = xy$. The new point z is therefore also in $\Re^n$. He then gives covering numbers for linear function classes

$$\mathcal{L}(\Theta, z) = \Theta \cdot z$$

under various restrictions, for example restrictions on the norms of the vectors Θ and z.

In the problems in this paper we again assume that sample points are (x, y) pairs, where $x \in \mathcal{X}$ is an input, and $y \in \mathcal{Y}$ is the correct structure for that input. There is some function $\mathbf{GEN}(x)$ which maps any $x \in \mathcal{X}$ to a set of candidates. There is also a function $\Phi : \mathcal{X} \times \mathcal{Y} \rightarrow \Re^n$ that maps each (x, y) pair to a feature vector. We will transform any sample point (x, y) to a matrix $Z \in \Re^{N \times n}$ in the following way. Take N to be a positive integer such that $\forall x \in \mathcal{X}, \ \ |\mathbf{GEN}(x)| - 1 \leq N$. First, for simplicity assume that $\forall x, \ \ |\mathbf{GEN}(x)| = (N+1)$. Assume that there is some fixed, arbitrary ordering on the members of $\mathcal{Y}$, implying an ordering $y_1, y_2, \ldots, y_N$ on the members of $\mathbf{GEN}(x)$ which are not equal to the correct output y. Then we will take the j-th row of the matrix Z to be

$$Z_j = \Phi(x, y) - \Phi(x, y_j)$$

In the case that $|\mathbf{GEN}(x)| = N'$ is strictly less than $N - 1$, we will simply define $Z_j = Z_{N'}$ for $j > N'$, thereby "padding" the final rows of Z with $\Phi(x, y) - \Phi(x, y_{N'})$. Under this transformation, the distribution $D(x, y)$ over $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is mapped to a distribution $D(Z)$ which generates training and test examples that are in $\Re^{N \times n}$.

The next step is to replace $\mathcal{L}$ with a new function, $\mathcal{M}(\Theta, Z)$ where $Z \in \Re^{N \times n}$. We define

$$\mathcal{M}(\Theta, Z) = \min_{j=1,\ldots,N} \Theta \cdot Z_j$$

It can be seen that if Z is created from a pair (x, y) then

$$\mathcal{M}(\Theta, Z) = \Theta \cdot \Phi(x, y) - \max_{z \in \mathbf{GEN}(x), z \neq y} \Theta \cdot \Phi(x, z)$$

Because of this there are some other useful relationships:

$$Er(h_\Theta) = \sum_{x,y} D(x, y)[[h_\Theta(x) \neq y]] = \sum_{Z} D(Z)[[\mathcal{M}(\Theta, Z) \leq 0]]$$

and for a sample $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ creating a transformed sample $\{Z^1, \ldots, Z^m\}$

$$\frac{1}{m}\sum_{i=1}^{m}[[\Theta \cdot \Phi(x, y) - \max_{z \in \mathbf{GEN}(x), z \neq y} \Theta \cdot \Phi(x, z) < \gamma]] = \frac{1}{m}\sum_{i=1}^{m}[[\mathcal{M}(\Theta, Z^i) < \gamma]]$$

Zhang (2002) shows how bounds on the covering numbers of $\mathcal{L}$ lead to the theorems 6 and 8 of (Zhang, 2002), which are similar but tighter bounds than the bounds given in theorems 6 and 7 in section 6 of the current paper. Theorem A1 below states a relationship between the covering numbers for $\mathcal{L}$ and $\mathcal{M}$. Under this result, theorems 8 and 9 in the current paper follow from the covering bounds on $\mathcal{M}$ in exactly the same way that theorems 6 and 8 of (Zhang, 2002) are derived from the covering numbers of $\mathcal{L}$, and theorem 2 of (Zhang, 2002). So theorem A1 leads almost directly to theorems 8 and 9 in the current paper.

Theorem A1 *Let $\mathcal{N}_\infty(\mathcal{L}, \epsilon, m)$ be the covering number, as defined in definition 1 of (Zhang, 2002), for $\mathcal{L}(\Theta, z)$ under restrictions R_1 on Θ and R_2 on each sample point $z \in \Re^n$. Let $\mathcal{N}_\infty(\mathcal{M}, \epsilon, m)$ be the covering number for the function class $\mathcal{M}(\Theta, Z)$ where Θ also satisfies restriction R_1, and any row Z_j of a sample matrix $Z \in \Re^{N\times n}$ satisfies restriction R_2. Then*

$$\mathcal{N}_\infty(\mathcal{M}, \epsilon, m) \leq \mathcal{N}_\infty(\mathcal{L}, \epsilon, mN)$$

Proof: The proof rests on the following result. Take any sample $\mathcal{S}^m = \{Z^1, Z^2, \ldots, Z^m\}$ where $\forall i, \quad Z^i \in \Re^{N\times n}$. Construct another sample $\bar{\mathcal{S}}^{mN}$ of length $m \times N$, of elements Z^i_j for $i = 1, \ldots, m, j = 1, \ldots, N$, where $Z^i_j \in \Re^n$:

$$\bar{\mathcal{S}}^{mN} = \{Z^1_1, Z^1_2, \ldots, Z^1_N, Z^2_1, Z^2_2, \ldots, Z^2_N, \ldots, Z^m_1, Z^m_2, \ldots, Z^m_N\}$$

We will show that

$$\mathcal{N}_\infty(\mathcal{M}, \epsilon, \mathcal{S}^m) \leq \mathcal{N}_\infty(\mathcal{L}, \epsilon, \bar{\mathcal{S}}^{mN}) \tag{1.A.1}$$

This implies the result in the theorem, because $\mathcal{N}_\infty(\mathcal{L}, \epsilon, \bar{\mathcal{S}}^{mN}) \leq \mathcal{N}_\infty(\mathcal{L}, \epsilon, mN)$ by definition, and therefore for all samples $\mathcal{S}^m$, $\mathcal{N}_\infty(\mathcal{M}, \epsilon, \mathcal{S}^m) \leq \mathcal{N}_\infty(\mathcal{L}, \epsilon, mN)$, which implies that $\mathcal{N}_\infty(\mathcal{M}, \epsilon, m) = \max_{\mathcal{S}^m} \mathcal{N}_\infty(\mathcal{M}, \epsilon, \mathcal{S}^m) \leq \mathcal{N}_\infty(\mathcal{L}, \epsilon, mN)$.

To prove equation (1.A.1), we introduce the definitions

$$\begin{array}{rcl}
v_{\mathcal{L}}(\Theta) & = & \langle \mathcal{L}(\Theta, Z^1_1), \mathcal{L}(\Theta, Z^1_2), \ldots, \mathcal{L}(\Theta, Z^m_1), \ldots, \mathcal{L}(\Theta, Z^m_N)\rangle \\
\mathcal{V}_{\mathcal{L}} & = & \{\ v_{\mathcal{L}}(\Theta)\ :\ \Theta \in \Re^n\} \\
v_{\mathcal{M}}(\Theta) & = & \langle \mathcal{M}(\Theta, Z^1), \ldots, \mathcal{M}(\Theta, Z^m)\rangle \\
\mathcal{V}_{\mathcal{M}} & = & \{\ v_{\mathcal{M}}(\Theta)\ :\ \Theta \in \Re^n\}
\end{array}$$

So $v_{\mathcal{L}}$ and $v_{\mathcal{M}}$ are functions which map Θ to vectors in $\Re^{mN}$ and $\Re^m$ respectively. Let $\mathcal{A}$ be a set of vectors which form an ϵ–cover of the set $\mathcal{V}_{\mathcal{L}}$, and for which $|\mathcal{A}| \leq \mathcal{N}_\infty(\mathcal{L}, \epsilon, \bar{\mathcal{S}}^{mN})$. Each member of $\mathcal{A}$ is a vector $\langle a^1_1, a^1_2, \ldots, a^m_N\rangle \in \Re^{mN}$. Because $\mathcal{A}$ forms an ϵ–cover of $\mathcal{V}_{\mathcal{L}}$,

$$\forall v \in \mathcal{V}_{\mathcal{L}},\ \exists \bar{a} \in \mathcal{A} \ \text{ s.t. } \ \forall i, j \ \ |v^i_j - a^i_j| \leq \epsilon$$

We define a new set $\mathcal{B}$ of vectors in $\Re^m$ as

$$\mathcal{B} = \{\langle b^1, \ldots, b^m\rangle\ :\ \langle a^1_1, a^1_2, \ldots, a^m_N\rangle \in \mathcal{A}, \forall i = 1, \ldots, n, b^i = \min_{j=1,\ldots,N} a^i_j\}$$

Thus $|\mathcal{B}| \leq |\mathcal{A}|$, because each member of $\mathcal{B}$ is formed by a deterministic mapping from an element of $\mathcal{A}$. We will show that $\mathcal{B}$ forms an ϵ–cover of $\mathcal{V}_{\mathcal{M}}$. Consider any vector $v_{\mathcal{M}}(\Theta)$ in $\mathcal{V}_{\mathcal{M}}$. Consider a "parallel" vector $v_{\mathcal{L}}(\Theta)$. There is some $\bar{a} \in \mathcal{A}$ such that $\forall i, j \quad |a^i_j - (v_{\mathcal{L}}(\Theta))^i_j| \leq \epsilon$. Next consider the vector $\bar{b} \in \Re^m$ such that $b^i = \min_j a^i_j$. Then clearly $\bar{b} \in \mathcal{B}$. It can also be shown that $\forall i, \quad |b^i - (v_{\mathcal{M}}(\Theta))^i| \leq \epsilon$. This is because

$$\begin{array}{rl}
& \forall i, j \quad |a^i_j - \Theta \cdot Z^i_j| \leq \epsilon \\
\Rightarrow & \forall i \quad |\min\limits_j a^i_j - \min\limits_j \Theta \cdot Z^i_j| \leq \epsilon \\
\Rightarrow & \forall i \quad |b^i - \mathcal{M}(\Theta, Z^i)| \leq \epsilon
\end{array}$$

Thus we have constructed a set $\mathcal{B}$ which forms an ϵ–cover of $\mathcal{V}_{\mathcal{M}}$, and also has $|\mathcal{B}| \leq |\mathcal{A}|$. Because $|\mathcal{A}| \leq \mathcal{N}_\infty(\mathcal{L}, \epsilon, \bar{\mathcal{S}}^{mN})$ we have $|\mathcal{B}| \leq \mathcal{N}_\infty(\mathcal{L}, \epsilon, \bar{\mathcal{S}}^{mN})$, and the theorem follows.

Notes

1. Booth and Thompson (1973) also give a second, technical condition on the probabilities $p(r)$, which ensures that the probability of a derivation halting in a finite number of steps is 1.

2. Given any finite weights on the rules other than $\texttt{B} \rightarrow \texttt{a}$, it is possible to set the weight $\texttt{B} \rightarrow \texttt{a}$ sufficiently low for T_1 and T_2 to get higher scores than T_4 and T_5.

3. By "at most" we mean in the worst case under the choice of p. For some values of p convergence may be substantially quicker.

4. In our formulation this is not quite accurate: the values γ^i_Θ and γ_Θ remain constant when Θ is scaled by a constant $\beta > 0$ (i.e., $\gamma^i_\Theta = \gamma^i_{\beta\Theta}$ and $\gamma_\Theta = \gamma_{\beta\Theta}$ for any $\beta > 0$). To be more precise, the optimal hyperplane is unique up to arbitrary scalings by some value $\beta > 0$.

5. We are implicitly assuming that there are at least two candidates for each training sentence – sentences with only one candidate can be discarded from the training set.

References

Abney, S. (1997). Stochastic attribute-value grammars. *Computational Linguistics, 23*, 597-618.

M. Anthony and P. L. Bartlett. (1999). *Neural Network Learning: Theoretical Foundations.* Cambridge University Press.

Bartlett, P. L. (1998). The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2): 525-536, 1998.

Block, H. D. (1962). The perceptron: A model for brain functioning. *Reviews of Modern Physics*, 34, 123–135.

Bod, R. (1998). *Beyond Grammar: An Experience-Based Theory of Language*. CSLI Publications/Cambridge University Press.

Booth, T. L., and Thompson, R. A. (1973). Applying probability measures to abstract languages. *IEEE Transactions on Computers*, C-22(5), 442–450.

Collins, M. (1999). Head-Driven Statistical Models for Natural Language Parsing. PhD Dissertation, University of Pennsylvania.

Collins, M. (2000). Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 175–182. San Francisco: Morgan Kaufmann.

Collins, M., and Duffy, N. (2001). Convolution kernels for natural language. In Dietterich, T. G., Becker, S., and Ghahramani, Z., (eds.) *Advances in Neural Information Processing Systems 14 (NIPS 14)*. MIT Press, Cambridge, MA.

Collins, M., Schapire, R. E., and Singer, Y. (2002). Logistic regression, AdaBoost and Bregman distances. In *Machine Learning*, 48(1–3):253–285.

Collins, M., and Duffy, N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 263–270. San Francisco: Morgan Kaufmann.

Collins, M. (2002a). Ranking algorithms for named–entity extraction: Boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting of* *the Association for Computational Linguistics (ACL 2002)*, pages 489–496. San Francisco: Morgan Kaufmann.

Collins, M. (2002b). Discriminative training methods for hidden markov models: Theory and experiments with the perceptron algorithm. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8.

Cortes, C. and Vapnik, V. (1995). Support–vector networks. In *Machine Learning*, 20(3):273-297.

Crammer, K., and Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. In *Journal of Machine Learning Research*, 2(Dec):265-292.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines (and other Kernel-Based Learning Methods)*. Cambridge University Press.

Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*, 380–393.

Devroye, L., Gyorfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. Springer.

Demiriz, A., Bennett, K. P., and Shawe-Taylor, J. (2001). Linear programming boosting via column generation. In *Machine Learning*, 46(1):225–254.

Elisseeff, A., Guermeur, Y., and Paugam-Moisy, H. (1999). Margin error and generalization capabilities of multiclass discriminant systems. *Technical Report NeuroCOLT2*, 1999-051.

Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Freund, Y. and Schapire, R. (1999). Large margin classification using the perceptron algorithm. In *Machine Learning*, 37(3):277–296.

Freund, Y., Iyer, R.,Schapire, R.E., & Singer, Y. (1998). An efficient boosting algorithm for combining preferences. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, pages 170–178. Morgan Kaufmann.

Friedman, J. H., Hastie, T. and Tibshirani, R. (1998). Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 38(2), 337-374.

Hopcroft, J. E., and Ullman, J. D. (1979). *Introduction to automata theory, languages, and computation*. Reading, MA: Addison–Wesley.

Joachims, T. (1998). Making large-scale SVM learning practical. In (Scholkopf et al., 1998), pages 169–184.

Johnson, M., Geman, S., Canon, S., Chi, S., & Riezler, S. (1999). Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th An-*

nual Meeting of the Association for Computational Linguistics (ACL 99), pages 535–541. San Francisco: Morgan Kaufmann.

Lafferty, J. (1999). Additive models, boosting, and inference for generalized divergences. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory (COLT'99)*, pages 125–133.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 282–289. Morgan Kaufmann.

Lebanon, G., and Lafferty, J. (2001). Boosting and maximum likelihood for exponential models. In Dietterich, T. G., Becker, S., and Ghahramani, Z., (eds.) *Advances in Neural Information Processing Systems 14 (NIPS 14)*. MIT Press, Cambridge, MA.

Littlestone, N., and Warmuth, M. (1986). Relating data compression and learnability. *Technical report, University of California, Santa Cruz.*

Novikoff, A. B. J. (1962). On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, Vol XII, 615–622.

Platt, J. (1998). Fast training of support vector machines using sequential minimal optimization. In (Scholkopf et al., 1998), pages 185–208.

Ratnaparkhi, A., Roukos, S., and Ward, R. T. (1994). A maximum entropy model for parsing. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP 1994)*, pages 803-806. Yokohama, Japan.

Ratnaparkhi, A. (1996). A maximum entropy part-of-speech tagger. In *Proceedings of the 1996 Conference on Empirical Methods in Natural Language Processing (EMNLP 1996)*, pages 133–142.

Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65, 386–408.

Schapire R., Freund Y., Bartlett P. and Lee W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651-1686.

Scholkopf, B., Burges, C., and Smola, A. (eds.). (1998). *Advances in Kernel Methods – Support Vector Learning*, MIT Press.

Shawe-Taylor, J., Bartlett, P. L., Williamson, R. C., and Anthony, M. (1998). Structural Risk Minimization over Data-Dependent Hierarchies. *IEEE Transactions on Information Theory*, 44(5): 1926-1940, 1998.

Valiant, L. G. (1984). A theory of the learnable. In *Communications of the ACM*, 27(11):1134–1142.

Vapnik, V. N., and Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of probability and its applications*, 16(2):264–280.

Vapnik, V. N. (1998). *Statistical Learning Theory*. New York: Wiley.

Walker, M., Rambow, O., and Rogati, M. (2001). SPoT: A trainable sentence planner. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2001)*, pages 17–24.

Zhang, T. (2002). Covering Number Bounds of Certain Regularized Linear Function Classes. In *Journal of Machine Learning Research*, 2(Mar):527-550, 2002.