Parameter Estimation for Statistical Parsing Models: Theory and Practice of Distribution-Free Methods

Michael Collins
AT&T Labs-Research.
mcollins@research.att.com

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. As an alternative, we propose algorithms based on distribution-free analysis. We describe two algorithms based on these methods. The first uses boosting algorithms to rerank the output of an existing statistical parser. The second method uses the Perceptron or Support Vector Machine algorithms.

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 generative models such as Probabilistic Context-Free Grammars and Hidden Markov Models, Maximum Entropy models for tagging and parsing, and recent work on Markov Random Fields.

The first part of this paper discusses the statistical theory underlying various parameter-estimation methods. The assumptions under which maximum-likelihood estimation is justified are arguably quite strong – namely, that the structure of the process that generated 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 and colleagues, and to work within Valiant’s PAC model of learning – has been an explosion 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.

The second part of the paper discusses two parsing methods based on distribution-free training methods. The first uses boosting algorithms to rerank the output of an existing statistical parser. The second method uses the Perceptron or Support Vector Machine algorithms; a key insight is that these algorithms allow representation of parse trees through "kernels" – the paper discusses how the kernel trick can be used to give polynomial time algorithms for models with an exponential number of parameters, such as a representation tracking all subtrees of a tree (as in the DOP1 model for parsing [Bod 1998]).

2 Linear Models for Parsing

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 $X \to Y_1 Y_2 \ldots Y_n$ for $X, Y_i \in (N \cup \Sigma)$, 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 $G(x)$ 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).

A weighted grammar $G = (N, \Sigma, R, S, \Theta)$ also includes a parameter vector Θ which assigns a weight to each rule in R. If there are n rules in R, then $\Theta \in \Re^n$ (we assume that there is some arbitrary ordering $r_1 \ldots r_n$ of the rules in R, and that the i'th component of Θ is the weight on 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 from sentences to trees:

$$h_\Theta(x) = \arg\max_{y \in G(x)} \phi(x, y) \cdot \Theta \tag{1}$$

Finding $h_\Theta(x)$, the parse with the largest weight, can be achieved in polynomial time using a variant of the CKY parsing algorithm (in spite of a possibly exponential number of members of $G(x)$).

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. The basic question is: 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 quite 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 $G(x)$ for each sentence. The parameters of the parser are a vector Θ. The parser's output is defined in the same way as Eq. 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 [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 is: how should the parameter values be set 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 [Abney 1997; Johnson et al. 1999; Della Pietra et al. 1997]. This paper gives several alternatives to MRFs, and describes the theory and assumptions which underly various models.

A 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 $G(x)$ 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 [Collins 2000; Collins and Duffy 2001].

3 Probabilistic Context-Free Grammars

This section gives a review of the basic theory behind Probabilistic Context-Free Grammars (PCFGs). Say we have a context-free grammar $G = (N, \Sigma, R, S)$ as defined in section 2. 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,

$\Theta = \{\log p(r_1), \log p(r_2) \ldots \log p(r_n)\}$. 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

$$P(T|\Theta) = \prod_{r \in R} p(r)^{c(T,r)} \quad \text{or equivalently} \quad \log P(T|\Theta) = \sum_r c(T,r) \log p(r) = \phi(T) \cdot \Theta$$

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 $r = \alpha \to \beta$ is $p(\alpha \to \beta) = \sum_j c(T_j, \alpha \to \beta) / \sum_j c(T_j, \alpha)$ (here we overload the notation c so that $c(\alpha)$ 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 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(a). 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.

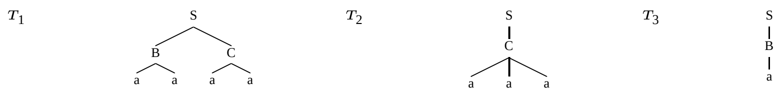
As the size of the training sample goes to infinity, the relative frequencies of trees $\{T_1, T_2, T_3\}$ in the training sample will converge to $\{p_1, p_2, p_3\}$. This makes it easy to calculate the rule weights that maximum-likelihood estimation converges to – see figure 1(b). We will call the PCFG with these asymptotic weights the *asymptotic PCFG*. Notice that the grammar generates trees never seen in training data, shown in figure 1(c). 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 misparsed. Figure 1(d) 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 misparsed by the asymptotic PCFG, resulting in an expected error rate of $(p_1 + p_2) = 30\%$ on newly drawn test examples.

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.

[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 B $\to$ a, it is possible to set the weight B $\to$ a sufficiently low for T_1 and T_2 to get higher scores than T_4 and T_5.

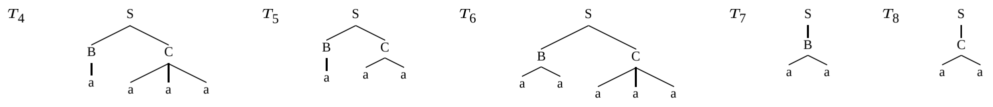
(a)



(b)

Rule No.	1	2	3	4	5	6	7
Rule	S → B C	S → C	S → B	B → a a	B → a	C → a a	C → a a a
Asymptotic ML Estimate	p_1	p_2	p_3	$p_1/(p_1+p_3)$	$p_3/(p_1+p_3)$	$p_1/(p_1+p_2)$	$p_2/(p_1+p_2)$

(c)



(d)

Tree	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
Rules Used	1,4,6	2,7	3,5	1,5,7	1,5,6	1,4,7	3,4	2,6
Asymptotic Prob. Estimate with $p_1 = 0.2$, $p_2 = 0.1$, $p_3 = 0.7$	0.0296	0.0333	0.544	0.0519	0.104	0.0148	0.156	0.0667

Figure 1: (a) Training and test data consists of trees T_1, T_2 and T_3 drawn with probabilities p_1, p_2 and p_3. (b) The ML estimates of rule probabilities converge to simple functions of p_1, p_2, p_3 as the training size goes to infinity. (c) The grammar also generates $T_4 \ldots T_8$, which are never seen in training or test data. (d) The probabilities assigned to the trees as the training size goes to infinity, for $p_1 = 0.2, p_2 = 0.1, p_3 = 0.7$.

4 Theory

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 $\mathcal{X}$ to $\mathcal{Y}$. 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.
- 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 $\hat{Er}(h) = \frac{1}{m} \sum_i L(y_i, h(x_i))$.

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.

4.1 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, $\hat{\Theta} = \arg\max \sum_i \log P(x_i, y_i|\Theta)$. 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.

4.2 An Overview of Distribution-Free Methods

From the arguments in the previous section, parametric methods are optimal *providing that the distribution generating the data is in the class of distributions being considered*. But what happens if this assumption is violated? In this case there are no guarantees on the expected error rate of the maximum-likelihood method. The example in section 3 shows how maximum-likelihood estimation can be badly mislead when the distribution generating the data is not in the class being considered.

This paper proposes alternatives to maximum-likelihood methods which give theoretical guarantees without making the assumption that the distribution generating the data comes from some predefined class. The only assumption is that the same distribution generates both training and test examples. These methods also provide bounds on how many training samples are required for learning, dealing with the case where there is only a finite amount of training data. Thus the methods address a second weakness of the parametric approach: *the guarantees of ML estimation are asymptotic, holding only in the limit as the training data size goes to infinity*.

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 $\mathcal{H} = \{h_\Theta(x) = \arg\max_{y \in G(x)} \phi(x, y) \cdot \Theta : \Theta \in \Re^n\}$. 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, $h^* = \arg\min_{h \in \mathcal{H}} Er(h) = \arg\min_{h \in \mathcal{H}} \sum_{x,y} D(x, y)L(y, h(x))$. A first learning method to study is as follows. Given a training sample (x_i, y_i) for $i = 1 \ldots m$, the method 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" by Vapnik [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 weighted grammar example above, the method is also consistent. However, some infinite hypothesis spaces can lead to the method being inconsistent – specifically, if a measure called the VC dimension [Vapnik 1998] 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.

4.3 Convergence Results 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. Hyperplane classifiers go back to one of the earliest learning algorithms, the Perceptron algorithm [Rosenblatt 1958]. There has been a large amount of effort devoted to the theory of hyperplane classifiers. 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 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. 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, $\mathcal{H} = \{h_{\Theta,b}(x) : \Theta \in \Re^n, b \in \Re\}$.

It can be shown that the ERM method is consistent for hyperplanes, through a method called VC analysis [Vapnik 1998]. 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. First consider the case where a training sample $\{(x_1, y_1) \ldots (x_m, y_m)\}$ is "linearly separable" – there is a hyperplane which achieves 0 errors on the training data. Then for each hyperplane with 0 error (there will in general be more than one), the *margin* on the training set for hyperplane $h_{\Theta,b}$ is defined as[3]

$$\gamma_{\Theta,b} = \min_i \frac{y_i \left(\phi(x_i) \cdot \Theta + b\right)}{||\Theta||} \qquad (2)$$

[3] $||\Theta||$ is the Euclidean norm, $\sqrt{\sum_j \Theta_j^2}$

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 1 *Special case of [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 $h_{\Theta,b} \in \mathcal{H}$ 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,*

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

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 5.2. 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 $\hat{L}(h_{\Theta,b}, \gamma)$ to be the proportion of examples on training data with margin less than γ for the hyperplane $h_{\Theta,b}$:

$$\hat{L}(h_{\Theta,b}, \gamma) = \frac{1}{m}\sum_i \left[\left[\frac{y_i\left(\phi(x_i)\cdot\Theta + b\right)}{||\Theta||} < \gamma\right]\right] \tag{3}$$

The following theorem can now be stated:

Theorem 2 *[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,*

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

where c is a constant.

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 5.2, 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, $\sum_j |\Theta_j|$):

$$\hat{L}_1(h_{\Theta,b}, \gamma) = \frac{1}{m}\sum_i \left[\left[\frac{y_i\left(\phi(x_i)\cdot\Theta + b\right)}{||\Theta||_1} < \gamma\right]\right] \tag{4}$$

Theorem 3 *[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,*

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

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 section 5.3.

```
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 $\Theta$ to be 0
            For $t$ = 1 to $T$, For $i$ = 1 to $m$,
              Calculate $y = h_\Theta(x_i) = \arg\max_{z \in G(x_i)} \phi(x_i, z) \cdot \Theta$
              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 $\Theta$
```

Figure 2: The perceptron algorithm for parsing. It takes T passes over the training set.

4.4 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. The method for converting parsing to a margin-based problem is very similar to the method for ranking problems described in [Freund et al. 1998]. As a first step, we can define the concept of margin on the training set, which is analogous to the definition in Eq. 2 of the margin for hyperplane classifiers:

$$\gamma_\Theta = \min_{i, y \in G(x_i), y \neq y_i} \left(\phi(x_i, y_i) \cdot \Theta - \phi(x_i, y) \cdot \Theta\right) / ||\Theta|| \tag{5}$$

The margin on the training set is now the minimum difference between the correct tree for a sentence and the next highest scoring tree for that sentence. The first SVM algorithm described in section 5.2 searches for the parameter values which give the maximum value for γ_Θ.

The bounds in theorems 2 and 3 suggested a tradeoff between keeping the values for $\hat{L}(h_{\Theta,b}, \gamma)$ and $\hat{L}_1(h_{\Theta,b}, \gamma)$ low and keeping the value of γ high. For parsing, we suggest the following analogous terms to $\hat{L}$ and $\hat{L}_1$:

$$\hat{RL}(h_\Theta, \gamma) = \frac{1}{m} \sum_i \frac{1}{|G(x_i)| - 1} \sum_{y \in G(x_i), y \neq y_i} \left[\left[\frac{\phi(x_i, y_i) \cdot \Theta - \phi(x_i, y) \cdot \Theta}{||\Theta||} < \gamma \right]\right] \tag{6}$$

$$\hat{RL}_1(h_\Theta, \gamma) = \frac{1}{m} \sum_i \frac{1}{|G(x_i)| - 1} \sum_{y \in G(x_i), y \neq y_i} \left[\left[\frac{\phi(x_i, y_i) \cdot \Theta - \phi(x_i, y) \cdot \Theta}{||\Theta||_1} < \gamma \right]\right] \tag{7}$$

The algorithms described in section 5 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. The bounds in theorems 2 and 3 do not apply to the parsing case, but it is likely that similar theorems apply – we leave this to future work. Theorem 6 of [Schapire et al. 1998] treats a similar case to the parsing example, and it is likely that this this proof holds for the parsing set-up.

5 Algorithms

5.1 A Variant of the Perceptron Algorithm for Parsing

The first algorithm for setting the parameter values Θ is the perceptron algorithm, as introduced by [Rosenblatt 1958]. Figure 2 shows the algorithm. Note that the main computational difficulty 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. Thus for the weighted CFG representation, the perceptron algorithm is relatively efficient. Other representations may have to rely on explicitly calculating $\phi(x_i, z) \cdot \Theta$ for all $z \in G(x_i)$, and hence depend computationally on the number of candidates $|G(x_i)|$ for $i = 1 \ldots m$.

It is useful to define the maximum-achievable margin γ on a separable training set as $\gamma = \max_{\Theta \in \Re^n} \gamma_\Theta = \max_{\Theta \in \Re^n} \min_{i, y \in G(x_i), y \neq y_i} \frac{\phi(x_i, y_i) \cdot \Theta - \phi(x_i, y) \cdot \Theta}{||\Theta||}$. The following theorem can then be stated:

Theorem 4 *(Simple modification of theorem from [Block 1962; Novikoff 1962], see also [Freund and Schapire 1999]). Let $\{(x_1, y_1) \ldots (x_n, y_n)\}$ be a sequence of examples such that $\forall i$, $\forall y \in G(x_i)$, $||\phi(x_i, y_i) - \phi(x_i, y)|| \leq R$. 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: Simple modification of the proof by [Block 1962; Novikoff 1962], see also [Freund and Schapire 1999].

This theorem implies that if the training sample in figure 2 is separable, and we iterate the algorithm repeatedly over the training sample (i.e., $T \to \infty$), then the algorithm converges to a parameter setting that classifies the training set with zero errors. 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.

5.2 Support Vector Machines

Now consider search for the maximum margin hyperplane, the hypothesis Θ with maximum value for γ_Θ (Eq. 5). It can be shown [Vapnik 1998] that the parameter values which give the maximum-margin solution can be found by minimizing $||\Theta||^2$ subject to the constraints $\forall i, \forall y \in G(x_i)$ s.t. $y \neq y_i, \quad \phi(x_i, y_i) \cdot \Theta - \phi(x_i, y) \cdot \Theta \geq 1$. Thus there are $\sum_i (|G(x_i)| - 1) = \left(\sum_i |G(x_i)| - m\right)$ constraints.

Next, consider search for a hypothesis Θ which has a low value of $\hat{RL}(h_\Theta, \gamma)$ (Eq. 6) for some relatively large value of γ. [Cortes and Vapnik 1995] suggest the following constrained optimization problem: minimize $||\Theta||^2 + C \sum_{i, y \in G(x_i), y \neq y_i} \epsilon(i, y)$ subject to the constraints $\forall i, \forall y \in G(x_i)$ s.t. $y \neq y_i, \quad \phi(x_i, y_i) \cdot \Theta - \phi(x_i, y) \cdot \Theta \geq 1 - \epsilon(i, y)$. Here $\epsilon(i, y)$ are a set of "slack variables". Any examples (i, y) with $\epsilon(i, y) = 0$ are classified with at least a margin of $1/||\Theta||$; any examples with a positive–valued slack variable are classified with a margin less than $1/||\Theta||$. The variable C is a constant which manages the balance between keeping $||\Theta||^2$ small and the slack variables small. 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 $\epsilon(i, y) > 0$) 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 2. 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 have been studied extensively (e.g., see [Joachims 1998; Platt 1998]) and can be solved relatively efficiently. (A publicly available package for Support Vector Machines, written by Thorsten Joachims, is available from `http://ais.gmd.de/˜ thorsten/svm_light/`.)

5.3 Boosting

The AdaBoost algorithm [Freund and Schapire 1997] is one method for optimizing the bound in Theorem 3 [Schapire et al. 1998]. Figure 3 shows the AdaBoost algorithm, altered slightly so that it applies to the parsing problem. The algorithm converts the training set into a set of triples: $\mathcal{T} = \{(x_i, y_i, y) : i = 1 \ldots m, y \in G(x_i)$ 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 $G(x)$. AdaBoost maintains a distribution D^t over the training examples such that $D^t(x, y_1, y_2)$ is proportional to $\exp\{-\Theta \cdot \left(\phi(x, y_1) - \phi(x, y_2)\right)\}$. 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 s'th component of Θ has r_s as a measure of how well correlated it is with the current distribution, $r_s = \sum_{(x, y_1, y_2) \in \mathcal{T}} D^t(x, y_1, y_2) \left(\phi_s(x, y_1) - \phi_s(x, y_2)\right)$. The magnitude of r_s can be taken as a measure of how correlated $\left(\phi_s(x, y_1) - \phi_s(x, y_2)\right)$ 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 fact, there is a strong relation between the values of $|r_s|$, and the margin-based bound in Theorem 3. If we define $\epsilon_t = (1 - |r_{s_t}|)/2$ then the following theorem holds:

Theorem 5 *(Slight Modification of Theorem 5 of [Schapire et al. 1998]). If we define $\hat{RL}_1(h_\Theta, \gamma)$ as in Eq. 7, and the Adaboost algorithm in figure 3 generates values $\epsilon_1, \epsilon_2, \ldots \epsilon_T$, then for all γ,*

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

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

Algorithm:

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

$$\forall (x, y_1, y_2) \in \mathcal{T}, \quad D^t(x, y_1, y_2) = \frac{1}{Z^t} \frac{e^{-\Theta \cdot \left(\phi(x,y_1) - \phi(x,y_2)\right)}}{|G(x)| - 1}$$

    where Z^t is a normalization term, i.e., $Z^t = \sum_{(x,y_1,y_2) \in \mathcal{T}} e^{-\Theta \cdot \left(\phi(x,y_1) - \phi(x,y_2)\right)} / (|G(x)| - 1)$.
  - For $s = 1 \dots n$ 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 3: The AdaBoost algorithm applied to parsing.

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

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

It can be shown that $f(\delta, \gamma)$ is less than one providing that $\gamma < \delta$: the implication is that for all $\gamma < \delta$, $\hat{RL}_1(h_\Theta, \gamma)$ 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 $\hat{RL}_1(h_\Theta, \gamma)$ for a relatively large range of γ, and by implication it will be successful in optimizing the bound in Theorem 3. 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 $s = 1 \dots n$, $\left(\phi_s(x, y_1) - \phi_s(x, y_2)\right)$ 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.

5.4 Dual forms of the Perceptron and SVM Algorithms

In the boosting algorithms, the training set was effectively converted into a set of triples, $\mathcal{T}$ (see figure 3). This set is of size $M = \sum_i |G(x_i)| - m$. For convenience, in this section we assume that the M elements of $\mathcal{T}$ are indexed, such that (x'_j, y'_j, z'_j) is the j'th element of $\mathcal{T}$. We also assume a function $I(i, y)$ which maps a triple (x_i, y_i, y) (for $i = 1 \dots m, y \in G(x_i), y \neq y_i$) to its index $j \in 1 \dots M$.

Now consider an alternative form for the perceptron algorithm, shown in figure 4. The algorithm does not explicitly represent the parameter vector Θ, but instead maintains weights α_j over the M examples in the training set. These "dual" variables α_j do, however, implicitly define the parameter vector Θ, through the identity $\Theta = \sum_{j=1}^{M} \alpha_j \left(\phi(x'_j, y'_j) - \phi(x'_j, z'j)\right)$. For example, the ranking score for a new example (x, y) can be calculated as

$$\phi(x, y) \cdot \Theta = \sum_{j=1}^{M} \alpha_j \left(\phi(x'_j, y'_j) \cdot \phi(x, y) - \phi(x'_j, z'j) \cdot \phi(x, y)\right) \tag{8}$$

```
**Input:** Examples $\{(x_1, y_1) \ldots (x_m, y_m)\}$, Grammar $G$, representation $\phi : \mathcal{X} \times \mathcal{Y} \to \Re^n$
**Algorithm:**
Initialise $\alpha_j = 0$ for $j = 1 \ldots M$
For $t$ = 1 to $T$,
  For $i$ = 1 to $m$,
    Calculate $y = \arg\max_{z \in G(x_i)} \sum_{j=1 \ldots M} \alpha_j \left( \phi(x'_j, y'_j) \cdot \phi(x_i, z) - \phi(x'_j, z'_j) \cdot \phi(x_i, z) \right)$
    If($y = y_i$) then do nothing; else if($y \neq y_i$) then $\alpha_{I(i,y)} = \alpha_{I(i,y)} + 1$

**Output:** The dual parameters $\alpha_j$. Output on a new sentence $x$ is
    $\arg\max_{y \in G(x)} \sum_j \alpha_j \left( \phi(x'_j, y'_j) \cdot \phi(x, y) - \phi(x'_j, z'_j) \cdot \phi(x, y) \right)$
```

Figure 4: The perceptron algorithm for parsing in dual form.

It can be verified that the algorithm in 4 is completely equivalent to the perceptron algorithm in 2: we will refer to the algorithm in 4 as the perceptron algorithm in "dual form".

The dual form is useful because for some representations the dual form algorithm is much more computationally efficient than the usual algorithm. This occurs when the inner product between two examples (x_1, y_2) and (x_2, y_2) (i.e., $\phi(x_1, y_1) \cdot \phi(x_2, y_2)$) can be computed efficiently, in spite of the representation ϕ being very high dimensional. See chapter 3 of [Cristianini and Shawe-Taylor 2000] for examples of many such representations.

To illustrate this, we will consider one particular representation as an example. The DOP1 model [Bod 1998] describes a representation which keeps track of all subtrees seen in training data. We will consider linear models with this representation: $\phi(x, y)$ has as many components as there are subtrees in training data, and it tracks the count of each of these subtrees in the example (x, y). This is a very high dimensional representation, because in general a tree has an exponential number of subtrees. This makes the perceptron algorithm in its original form (figure 2) prohibitively inefficient – directly computing $\Theta \cdot \phi(x, y)$ for some example will take time linear in the number of subtrees of $\phi(x, y)$, an exponential number.

In contrast, it turns out that the dual form algorithm can be applied to the problem efficiently. The key to this is that the inner product between any two trees, $\phi(x_1, y_1) \cdot \phi(x_2, y_2)$, can be calculated in polynomial time using dynamic programming, in spite of the size of ϕ. See [Collins and Duffy 2001] for details. Armed with a subroutine which calculates $\phi(x_1, y_1) \cdot \phi(x_2, y_2)$ for any two trees efficiently, the dual form algorithm can find a set of dual parameters which define a separating hyperplane in the DOP1 representation space.

The SVM algorithms have a similar dual form: the final hypothesis (maximum margin hyperplane) can be expressed through a linear combination of training examples (as in Eq. 8), and the optimization problem can be solved through calculations involving inner products between training examples.

6 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.

Acknowledgements. 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. Thanks also to Nigel Duffy, for many useful discussions while we were collaborating on the use of kernels for parsing problems.

References

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

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

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

[Booth and Thompson 1973] Booth, T. L., and Thompson, R. A. 1973. Applying Probability Measures to Abstract Languages. *IEEE Transactions on Computers*, C-22(5), 442–450.

[Collins 2000] Collins, M. (2000). Discriminative Reranking for Natural Language Parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*. San Francisco: Morgan Kaufmann.

[Collins and Duffy 2001] Collins, M. and Duffy, N. 2001. Parsing with a Single Neuron: Convolution Kernels for Natural Language Problems. Technical Report, University of California at Santa Cruz.

[Cortes and Vapnik 1995] Cortes, C. and Vapnik, V. 1995. Support–Vector Networks. In *Machine Learning*, 20(3):273-297.

[Cristianini and Shawe-Taylor 2000] Cristianini, N. and Shawe-Taylor, J. 2000. *An introduction to support vector machines (and other kernel-based learning methods)*. Cambridge University Press.

[Della Pietra et al. 1997] Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*, 380–1.593.

[Devroye et. al 1996] Devroye, L., Gyorfi, L., and Lugosi, G. 1996. *A Probabilistic Theory of Pattern Recognition*. Springer.

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

[Freund and Schapire 1999] Freund, Y. and Schapire, R. (1999). Large Margin Classification using the Perceptron Algorithm. In *Machine Learning*, 37(3):277–296.

[Freund et al. 1998] Freund, Y., Iyer, R.,Schapire, R.E., & Singer, Y. 1998. An efficient boosting algorithm for combining preferences. In *Machine Learning: Proceedings of the Fifteenth International Conference*. Morgan Kaufmann.

[Hopcroft and Ullman 1979] Hopcroft, J. E., and Ullman, J. D. 1979. *Introduction to automata theory, languages, and computation*. Reading, Mass.: Addison–Wesley.

[Joachims 1998] Joachims, T. 1998. Making large-Scale SVM Learning Practical. In [Scholkopf et al. 1998].

[Johnson et al. 1999] Johnson, M., Geman, S., Canon, S., Chi, S., & Riezler, S. (1999). Estimators for stochastic ‘unification-based” grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. San Francisco: Morgan Kaufmann.

[Novikoff 1962] 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 1998] Platt, J. 1998. Fast Training of Support Vector Machines using Sequential Minimal Optimization. In [Scholkopf et al. 1998].

[Rosenblatt 1958] Rosenblatt, F. 1958. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65, 386–408. (Reprinted in *Neurocomputing* (MIT Press, 1998).)

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

[Vapnik 1998] Vapnik, V. N. 1998. *Statistical Learning Theory*. New York: Wiley.