

# Constraint Classification: A New Approach to Multiclass Classification

Sariel Har-Peled, Dan Roth, and Dav Zimak

Department of Computer Science,  
University of Illinois,  
Urbana, IL 61801,  
{sariel,danr,davzimak}@uiuc.edu

**Abstract.** In this paper, we present a new view of multiclass classification and introduce the constraint classification problem, a generalization that captures many flavors of multiclass classification. We provide the first optimal, distribution independent bounds for many multiclass learning algorithms, including winner-take-all (WTA). Based on our view, we present a learning algorithm that learns via a single linear classifier in high dimension. In addition to the distribution independent bounds, we provide a simple margin-based analysis improving generalization bounds for linear multiclass support vector machines.

## 1 Introduction

Multiclass classification is a central problem in machine learning, as applications that require a discrimination among many classes are ubiquitous. Examples studied in machine learning include handwritten character recognition [LS97,LBD<sup>+</sup>89], part-of-speech tagging [Bri94,RZ98], speech recognition [Jel98] and text page classification [ADW94,DKR97].

Learning approaches for these tasks often use a winner-take-all (WTA) strategy at some point in the decision process. As input, the WTA uses several real valued functions and returns the index of the function with the highest value. Classifiers making use of WTA exist in many sub-fields of machine learning, such as neural networks [AB99], self-organizing maps [Koh01] and, more recently, networks of linear threshold functions such as SNoW [CCR99, Rot98] and multiclass support vector machines [CS01a, ASS00, Sch97, WW99].

Currently there is only a limited understanding of multiclass classification. It is known, for example, that WTA is a powerful classifier in terms of expressivity [Maa00]: neural network using a single WTA unit as its only nonlinear computational element can, with enough inputs, approximate arbitrary continuous functions. However, little is known about generalization properties and convergent algorithmic approaches. A PAC-style analysis of multiclass functions that uses an extended notion of VC-dimension for multiclass case [BCHL95]

provides poor bounds on generalization for WTA, and the current best bounds rely on a generalized notion of margin [ASS00]. However, the more fundamental question about deriving such bounds directly has not been investigated.

Algorithmically, most approaches to multiclass classification make use of “standard” binary classifiers to encode and train the output labels. The most commonly used scheme makes a “one versus all” (OvA) assumption, dictating that for each class there exists a single separator between that class and all other classes. “All versus all” (AvA, or pairwise) classification [HT98] is a more expressive alternative which assumes the existence of a separator between any two classes (but requires training a quadratic number of Boolean classifiers).

This work is motivated by several successful practical approaches, such as multiclass SVM and SNoW that rely on WTA-style strategies over linear functions (that produce a decision surface consisting of linear boundaries). In this paper, we introduce a new constraint classification problem that generalizes many interesting variants of multiclass classification and derive both VC-dimension and margin-based generalization bounds. The VC-dimension bounds are optimal for the case of WTA classifiers over linear functions.

Underlying our results is a new view of multiclass classification based on a simple transformation. In Section 3.2, we present a new algorithm that transforms an example set into higher dimensional space and learns a simple separating hyperplane in this space. Our algorithm suggests a way to fix the popular “one-vs-all” training scheme so that it will guarantee consistent classification. This algorithm turns out to be similar to the ultraconservative online algorithm for multiclass problems introduced in [CS01b].

In Section 4, we use the high dimensional view of multiclass classification and standard bounds for learning separating hyperplanes to develop generalization bounds. Distribution independent bounds are developed using a multiclass version of the standard growth function, and margin-based bounds are given that improve over previous bounds.

## 2 Learning Problems

Learning problems often assume that examples,  $(x, y) \in \mathcal{X} \times \mathcal{Y}$ , are drawn *i.i.d.* from a fixed probability distribution,  $\mathcal{D}_{\mathcal{X} \times \mathcal{Y}}$ , over  $\mathcal{X} \times \mathcal{Y}$ .  $\mathcal{X}$  is referred to as the instance space and  $\mathcal{Y}$  is referred to as the output space (*label set*). The learning algorithm is presented with  $m$  training examples,  $P$ , drawn in such a fashion from  $\mathcal{D}_{\mathcal{X} \times \mathcal{Y}}$  and attempts to output a function (*hypothesis*)  $h : \mathcal{X} \rightarrow \mathcal{Y}$  from the set of all functions (hypotheses)  $\mathcal{H}$  that minimizes the *empirical error* on  $P$ .

**Definition 1 (Permutations)** Denote the set of full orders over  $\{1, \dots, k\}$  as  $S^k$ , consisting of all permutations of  $\{1, \dots, k\}$ . Similarly,  $\bar{S}^k$  denotes the set of all partial orders over  $\{1, \dots, k\}$ . A partial order,  $c \in \bar{S}^k$ , defines a binary relation,  $\prec_c$  and can be represented by set of pairs on which  $\prec_c$  holds,  $c = \{(i, j) | i \prec_c j\}$ . In addition, for any set of pairs  $c = \{(i_1, j_1), \dots, (i_n, j_n)\}$ , we refer to  $c$  both as a set of pairs and as the partial order produced by the transitive

closure of  $c$  with respect to  $\prec_c$ . Given two partial orders  $a, b \in \bar{S}^k$ ,  $a$  is *consistent* with  $b$  (denoted  $a \sqsubseteq b$ ) if for every  $(i, j) \in \{1, \dots, k\}^2$ ,  $i \prec_b j$  holds whenever  $i \prec_a j$ . If  $c \in S^k$  is a full order, then it can be represented by a list of  $k$  integers where  $i \prec_c j$  if  $i$  precedes  $j$  in the list, and is referred to as a *ranking*. The size of a partial order,  $|c|$  is the number of pairs specified in  $c$ .

**Definition 2 (Learning)** Given  $m$  examples,  $P = ((x_1, y_1), \dots, (x_m, y_m))$ , drawn *i.i.d.* from  $\mathcal{D}_{\mathcal{X} \times \mathcal{Y}}$ , a hypothesis class  $\mathcal{H}$  and an error function  $\mathcal{E} : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \rightarrow \{0, 1\}$ , a learning algorithm  $\mathcal{L}(P, \mathcal{H})$  outputs a function  $h \in \mathcal{H}$ , where  $h : \mathcal{X} \rightarrow \mathcal{Y}$ .

Typically learning algorithms try to output the hypothesis  $h \in \mathcal{H}$  that minimizes the expected error on a randomly drawn example. The particular algorithm used may vary. We propose a framework that allows for any binary learning algorithm to be used, including support vector machines.

Many interesting learning problems can be expressed in this general learning framework simply by changing the type of output space,  $\mathcal{Y}$ . Binary classification, for example, sets  $\mathcal{Y} = \{-1, 1\}$ , whereas multiclass classification sets  $\mathcal{Y} = \{1, \dots, k\}$ .

In this paper, algorithms and generalization bounds are provided for the following learning problems (of course, the actual function classes used for classification must be restricted to produce a generalization bound).

**Binary Classification** Set  $\mathcal{Y} = \{-1, 1\}$ .

**Multiclass Classification** Set  $\mathcal{Y} = \{1, \dots, k\}$ .

The most commonly used multiclass classification scheme is the *winner-take-all (WTA)* strategy, where each class  $i \in \mathcal{Y}$  is represented with a real-valued function  $f_i : \mathcal{X} \rightarrow \mathbb{R}$ . WTA outputs  $y^* = \operatorname{argmax}_{\{1, \dots, k\}} f_i(x)$  as the final class. Notice that binary classification is a special case of multiclass classification where  $k = 2$ .

**$l$ -Multilabel Classification** Set  $\mathcal{Y} = \{1, \dots, k\}^l$ .

In practice, it is common for examples to be labeled with more than one class from  $\{1, \dots, k\}$ . When classifying web pages, for example, a single page might be a “homepage”, a “faculty page”, and a “machine learning page,” in which case a web classification system should be able to produce all three labels as output. Clearly multiclass classification and binary classification are special cases of  $l$ -multilabel classification.

Just as the WTA function is used for multiclass classification, the  $l$ -WTA function can be used for  $l$ -multiclass classification.  $l$ -WTA outputs  $l$  labels from  $\operatorname{argmax}_{\{1, \dots, k\}}^l f_i(x)$ , where  $\operatorname{argmax}^l$  outputs a set of consisting of the  $l$  highest values of  $f_i(x)$ .

**Ranking Classification** Set  $\mathcal{Y} = S^k$ .

A ranking classifier returns a full order, or permutation, of  $\{1, \dots, k\}$  from the set  $S^k$ . In some settings, class  $i$  might be “preferred” over class  $j$ . Often  $i$  precedes  $j$  when  $f_i(x) > f_j(x)$ .

**Constraint Classification** Set  $\mathcal{Y} = \bar{S}^k$ .

Constraint classification is a direct generalization of ranking classification where the classifier must output a partial order from  $\bar{S}^k$ , the set of all partial orders of  $\{1, \dots, k\}$ .

**$c$ -Constraint Classification** Set  $\mathcal{Y} = \bar{S}_c^k$ .

The set  $\bar{S}_c^k$  is simply a subset of  $\bar{S}^k$  where for all  $y \in \bar{S}_c^k$ ,  $|y| \leq c$ . Clearly constraint classification is a special case of  $c$ -constraint classification where  $c = \binom{k}{2}$ .

Constraint classification is very general and powerful since every example can have its own specific set of constraints. The distribution  $\mathcal{D}_{\mathcal{X} \times \bar{S}^k}$  may be constrained based on a strong relationship among the classes  $\{1, \dots, k\}$ , or even on the example,  $x$ , at hand. For example, suppose every constraint  $(i, j) \in y$  is such that  $i$  and  $j$  are either both even or both odd. Unfortunately, this type of information is unknown to the learning algorithm a-priori and might be difficult to learn. Therefore, a hypothesis is acceptable if it is consistent with the examples. This notion is captured in the error function.

**Definition 3 (Error Indicator Function)** For any  $(x, y) \in \mathcal{X} \times \bar{S}^k$ , and hypothesis  $h : \mathcal{X} \rightarrow \bar{S}^k$ , the *indicator function*  $\mathcal{E}(x, y, h)$  indicates an error on example  $x$ ,  $\mathcal{E}(x, y, h) = 1$  if  $y \not\sqsubseteq h(x)$ , and 0 otherwise.

Given a set of  $m$  labeled examples  $P = ((x_1, y_1), \dots, (x_m, y_m)) \in \{\mathcal{X} \times \mathcal{Y}\}^m$ , the *indicator vector* for a function  $h \in \mathcal{H}$ ,

$$\mathcal{E}(P, h) = (\mathcal{E}(x_1, y_1, h), \mathcal{E}(x_2, y_2, h), \dots, \mathcal{E}(x_m, y_m, h))$$

indicates which examples were classified incorrectly. Finally, we define

$$\mathcal{E}(P, \mathcal{H}) = \left\{ \mathcal{E}(P, h) \mid h \in \mathcal{H} \right\}$$

to be the set of all possible indicator vectors for  $\mathcal{H}$ .

For example, if  $k = 4$  and the example  $(x, \{(2, 3), (2, 4)\})$ ,  $h_1(x) = (2, 3, 1, 4)$ , and  $h_2(x) = (4, 2, 3, 1)$ , then  $h_1$  is correct since 2 precedes 3 and 2 precedes 4 in the full order  $(2, 3, 1, 4)$  whereas  $h_2$  is incorrect since 4 precedes 2 in  $(4, 2, 3, 1)$ .

**Definition 4 (Error)** Given an example  $(x, y)$  drawn *i.i.d.* from  $\mathcal{D}_{\mathcal{X} \times \mathcal{Y}}$ , the *true error* of  $h \in \mathcal{H}$ , where  $h : \mathcal{X} \rightarrow \mathcal{Y}$  is defined to be  $err(h) = \Pr_{\mathcal{D}}[\mathcal{E}(x, y, h) = 1]$ . Given  $P = ((x_1, y_1), \dots, (x_m, y_m))$  drawn from  $EX(\mathcal{D}_{\{\mathcal{X} \times \mathcal{Y}\}})$ , the *empirical error* of  $h \in \mathcal{H}$  with respect to  $P$  is defined to be  $err(P, h) = \frac{1}{|P|} \sum_{i=1 \dots m} \mathcal{E}(x_i, y_i, h)$ .

Throughout the rest of this paper, the each hypothesis from  $\mathcal{H}$  consists of a collection of linear functions over  $\mathcal{X} = \mathbb{R}^d$ . Specifically, hypothesis  $h \in \mathcal{H}$  can be represented as  $k$  weight vectors  $w_1, \dots, w_k \in \mathbb{R}^d$ , where  $w_i$  is associated with class  $i$ . Given an example  $x$ ,  $(w_i \cdot x)$  represents the “strength” or “confidence” of class  $i$ . With this assumption each of the above learning problems takes on a specific form, and furthermore, can be represented within the constraint classification setting. Table 1 shows their representation space and hypothesis classes.

Problem	Internal Representation	Output Space ( $\mathcal{Y}$ )	Hypothesis
binary	$w \in \mathbb{R}^d$	$\{-1, 1\}$	$\text{sign } w \cdot x$
multiclass	$(w_1, \dots, w_k) \in \mathbb{R}^{kd}$	$\{1, \dots, k\}$	$\text{argmax}_{\{1, \dots, k\}} w_i \cdot x$
$l$ -multiclass	$(w_1, \dots, w_k) \in \mathbb{R}^{kd}$	$\{1, \dots, k\}^l$	$\text{argmax}_{\{1, \dots, k\}}^l w_i \cdot x$
ranking	$(w_1, \dots, w_k) \in \mathbb{R}^{kd}$	$S^k$	$\text{argsort}_{\{1, \dots, k\}} w_i \cdot x$
constraint*	$(w_1, \dots, w_k) \in \mathbb{R}^{kd}$	$\bar{S}^k$	$\text{argsort}_{\{1, \dots, k\}} w_i \cdot x$
$c$ -constraint*	$(w_1, \dots, w_k) \in \mathbb{R}^{kd}$	$\bar{S}_c^k$	$\text{argsort}_{\{1, \dots, k\}} w_i \cdot x$

**Table 1.** Learning problems differ based on their output space and on their internal representation. Definitions for each of the problems considered in this paper (\*notice that the output space for the constraint classification problems is different than the range of the hypothesis space).  $\text{argmax}^l$  is a variant of  $\text{argmax}$  that returns the indices of the weight vectors that give the  $l$  largest values of  $w_i \cdot x$  instead of only the largest.  $\text{argsort}$  is a linear sorting function (see Definition 5).

The relationships among all of these problems can be made explicit. For example, 2-class multiclass classification is equivalent to binary classification since  $\text{argmax}_{\{1,2\}} w_i \cdot x = 1$  exactly when  $(w_1 \cdot x - w_2 \cdot x) = (w_1 - w_2) \cdot x > 0$ ; both problems compute separating hyperplanes in  $\mathbb{R}^d$ . Furthermore, constraint classification can capture all of the problems in Table 1.

**Definition 5 (Linear Sorting Function)** Let  $w = (w_1, \dots, w_k)$  be a set of  $k$  vectors, where  $(w_1, \dots, w_k) \in \mathbb{R}^d$ . Given  $x \in \mathbb{R}^d$ , a *linear sorting classifier* is a function  $h : \mathbb{R}^d \rightarrow S^k$  computed in the following way:

$$h(x) = \underset{i=1 \dots k}{\text{argsort}} w_i \cdot x,$$

where  $\text{argsort}$  returns a permutation of  $\{1, \dots, k\}$  where  $i$  precedes  $j$  if  $w_i \cdot x > w_j \cdot x$ . In the case that  $w_i \cdot x = w_j \cdot x$ ,  $i$  precedes  $j$  if  $i < j$ .

**Lemma 1 (Problem mappings).** *All of the learning problems in Table 1 can be expressed as constraint classification problems.*

*Proof.* Because binary and multilabel classification are special cases of  $l$ -multilabel classification, and ranking and constraint classification are special cases of  $c$ -constraint classification, it suffices to show that  $l$ -multilabel classification can be expressed as a  $c$ -constraint classification problem.

Given an example  $(x, y) \in \mathbb{R}^d \times \{1, \dots, k\}^l$ , define a new example  $(x, y_c) \in \mathbb{R}^d \times \bar{S}_{l(k-l)}^k$  (notice that  $y$  is a set of integers and  $y_c$  is a set of integer pairs), where

$$y_c = \left\{ (i, j) \mid i \in y, j \in \{1, \dots, k\} \setminus y \right\}.$$

Any  $l$ -multilabel classifier  $h_l(x) = \text{argmax}_{\{1, \dots, k\}}^l w_i \cdot x$  will err on example  $(x, y)$  exactly when the  $c$ -constraint classifier  $h_c(x) = \text{argsort}_{\{1, \dots, k\}} w_i \cdot x$  errs, and the indicator functions for the two classifiers are equal,  $\mathcal{E}(x, h_l) = \mathcal{E}(x, h_c)$ .

The size of the constraint sets for each of the problem mappings appears up in the margin-based generalization bound in Section 4.2. Binary classification becomes a 2-constraint classification problem and multiclass classification is transformed into a  $(k - 1)$ -constraint classification problem. Ranking classification becomes a  $(k - 1)$ -constraint classification problem by noticing that any full order from  $S^k$  can be represented by only  $k - 1$  constraints, namely  $(i, j) \in y_c$  if  $i$  precedes  $j$  in  $y$ .

Consider a 4-class multiclass example,  $(x, 3)$ . It is transformed into the 3-constraint example,  $(x, \{(3, 1), (3, 2), (3, 4)\})$ . If we find a constraint classifier that correctly labels  $x$  according to the given constraints ( $w_3 \cdot x > w_1 \cdot x$ ,  $w_3 \cdot x > w_2 \cdot x$ , and  $w_3 \cdot x > w_4 \cdot x$ ), then  $3 = \operatorname{argmax}_{1,2,3,4} w_i \cdot x$ .

The goal of a learning algorithm for constraint classification is to return a linear sorting function that has small error on the training data.

### 3 Learning

In this section, the  $k$ -class constraint classification problem is transformed into a binary classification problem in higher dimension. Each example  $(x, y) \in \mathbb{R}^d \times \bar{S}^k$  is transformed into a set of examples in  $\mathbb{R}^{kd} \times \{-1, 1\}$  based on the information contained in  $y$ . Then, a single separating hyperplane in  $\mathbb{R}^{kd}$  can be interpreted as a collection of  $k$  weight vectors, in  $\mathbb{R}^d$ , of a linear sorting classifier.

It is important to point out the connection to optimization when using linear sorting functions. Weston and Watkins' [WW99] transformation of multiclass classification to an optimization problem is easily extended to constraint classification.

The goal of the (hard) optimization problem is to minimize

$$\Phi(w_1, \dots, w_k),$$

subject to constraints derived from examples in  $P$ ,

$$w_i \cdot x > w_j \cdot x,$$

$$\forall (x, y) \in P, \forall (i, j) \in y.$$

The solution  $w^* = (w_1^*, \dots, w_k^*)$  that minimizes  $\Phi(\cdot)$  is a point in  $\mathbb{R}^{kd}$  that can be interpreted as the coefficients of a linear sorting classifier consistent with the constraints given in a constraint classification problem. While  $w^*$  is certainly one feasible solution, there are many others. By viewing the constraint classification problem as a feasibility problem rather than an optimization problem, any standard learning algorithm for binary classification can be used – including perceptron, winnow, and kernel-based SVM's.

#### 3.1 Transformation

Here, a dual version of the above feasibility problem is described in detail to highlight the fact that the constraint classification problem is being transformed into an equivalent binary classification problem.

**Definition 6 (Chunk)** A vector  $\mathbf{v} = (v_1, \dots, v_{kd}) \in \mathbb{R}^{kd} = \mathbb{R}^d \times \dots \times \mathbb{R}^d$ , is broken into  $k$  chunks  $(\mathbf{v}_1, \dots, \mathbf{v}_k)$  where the  $i$ -th chunk,  $\mathbf{v}_i = (v_{(i-1)*d+1}, \dots, v_{i*d})$ .

**Definition 7 (Expansion)** Let  $\text{Vec}(x, i)$  be a vector  $x \in \mathbb{R}^d$  embedded in  $kd$  dimensions, by writing the coordinates of  $x$  in the  $i$ -th chunk of a vector in  $\mathbb{R}^{k(d+1)}$ . Denote by  $\mathbf{0}^l$  the zero vector of length  $l$ . Then  $\text{Vec}(x, i)$  can be written formally as the concatenation of three vectors,  $\text{Vec}(x, i) = (\mathbf{0}^{(i-1)*d}, x, \mathbf{0}^{(k-i)*d}) \in \mathbb{R}^{kd}$ . Finally,  $\text{Vec}(x, i, j) = \text{Vec}(x, i) - \text{Vec}(x, j)$ , as the embedding of  $x$  in the  $i$ -th chunk and  $-x$  in the  $j$ -th chunk of a vector in  $\mathbb{R}^{kd}$ .

**Definition 8 (Expanded Example Sets)** Given an example  $(x, y)$ , where  $x \in \mathbb{R}^d$  and  $y \in \bar{S}^k$ , we define the *expansion* of  $(x, y)$  into a set of examples as follows,

$$\mathbf{P}_+(x, y) = \left\{ (\text{Vec}(x, i, j), 1) \mid (i, j) \in y \right\} \subseteq \mathbb{R}^{kd} \times \{1\},$$

A set of negative examples is defined as the reflection of each expanded example through the origin, specifically

$$\mathbf{P}_-(x, y) = \left\{ (-\mathbf{x}, -1) \mid (\mathbf{x}, 1) \in \mathbf{P}_+(x, y) \right\} \subseteq \mathbb{R}^{kd} \times \{-1\},$$

and the set of both positive and negative examples is denoted by  $\mathbf{P}(x, y) = \mathbf{P}_+(x, y) \cup \mathbf{P}_-(x, y)$ . The expansion of a set of examples,  $P$ , is defined as the union of all of the expanded examples in the set,

$$\mathbf{P}(P) = \bigcup_{(x, y) \in P} \mathbf{P}(x, y) \subseteq \mathbb{R}^{kd} \times \{-1, 1\}.$$

For example, consider the example  $((x_1, x_2), \{(2, 3), (2, 4)\})$ . Set

$$\mathbf{P}_+((x_1, x_2), \{(2, 3), (2, 4)\}) = \left\{ \begin{aligned} &((0, 0, x_1, x_2, -x_1, -x_2, 0, 0), 1), \\ &((0, 0, x_1, x_2, 0, 0, -x_1, -x_2), 1) \end{aligned} \right\},$$

and

$$\mathbf{P}_-((x_1, x_2), \{(2, 3), (2, 4)\}) = \left\{ \begin{aligned} &((0, 0, -x_1, -x_2, x_1, x_2, 0, 0), -1), \\ &((0, 0, -x_1, -x_2, 0, 0, x_1, x_2), -1) \end{aligned} \right\}.$$

### 3.2 Algorithm

Given a set of examples  $P$ , compute the expanded examples  $\mathbf{P}(P)$  and learn a separating hyperplane,  $\mathbf{v} \in \mathbb{R}^{kd}$ , using any linear classification algorithm. If the learning algorithm is successful in finding a vector  $\mathbf{v}$  that correctly classifies all points in  $\mathbf{P}(P)$ , then  $h(x) = \text{argsort}_{1, \dots, k} \mathbf{v}_i \cdot x$  is also consistent with all constraints from all examples in  $P$ .

The significance of framework in multiclass classification is the observation that the hypothesis learned above is more expressive than the one learned by using the OvA assumption.

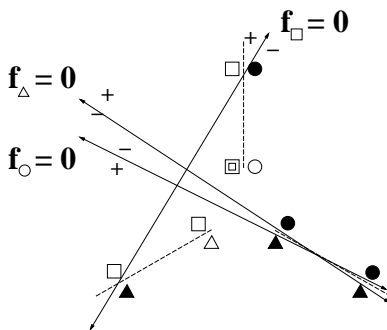
### 3.3 Comparison to “One vs. All”

A common approach to multiclass classification ( $\mathcal{Y} = \{1, \dots, k\}$ ) is to make the *one-versus-all* (OvA) assumption, that each class can be separated from the rest using a binary classification algorithm. Learning proceeds by learning  $k$  binary classifiers, each trained separately, where for example  $(x, y)$ ,  $x$  is considered a positive example for classifier  $i$  if  $y = i$  and as a negative example if  $y \neq i$ . Although a consistent classifier can be found for any set of training data when arbitrarily complex hypotheses are used, it will suffer poor generalization error.

**Definition 9 (OvA Assumption)** Let  $\mathcal{L}^*(P, \mathcal{H})$  be an “optimal” learning algorithm in the sense that  $h^* = \mathcal{L}^*(P, \mathcal{H})$  is a hypothesis,  $h^* \in \mathcal{H}$ , that has minimum error on  $P$ ,  $err(P, h^*) = \min_{h \in \mathcal{H}} err(P, h)$ .

Given a set of examples  $P = ((x_1, y_1), \dots, (x_m, y_m)) \in \mathbb{R}^d \times \{1, \dots, k\}$ , define  $P_i = ((x_1, eq(y_1, i)), \dots, (x_m, eq(y_m, i))) \in \mathbb{R}^d \times \{-1, 1\}$ , where  $eq(a, b) = 1$  if  $a = b$ , and  $eq(a, b) = -1$  otherwise. Let  $\mathcal{H}$  be the class of linear threshold functions where for all  $h \in \mathcal{H}$ ,  $h(x) = \text{sign}(w \cdot x)$ ,  $w, x \in \mathbb{R}^d$ , where  $\text{sign}(a) = 1$  if  $a > 0$  and  $\text{sign} = -1$  otherwise. Set  $h_i^* = \mathcal{L}^*(P_i, \mathcal{H})$ . The final WTA hypothesis trained on  $P$  is  $h^{WTA}(x) = \text{argmax}_{\{1, \dots, k\}} w_i^* \cdot x$ .

The above assumption is commonly used in practice [Rot98, AB99] where convergence is guaranteed only if each class is linearly separable from all other classes. However, this is often not the case and convergence may not occur even though a consistent multiclass classifier exists (see Figure 1 and Theorem 10).



**Fig. 1.** An example in  $\mathbb{R}^2$  showing that one-versus-all (OvA) will not converge to a consistent hypothesis. A three class classification problem (squares, triangles, and circles), where the solid points have weight 10 times that of the hollow points. The weighted points restrict the minimum error OvA separating hyperplanes. The square-classifier will predict the point outlined with a double square as negative while the circle-classifier will predict it as positive. Therefore, the final prediction for this point will be incorrect. One can verify the existence of a consistent WTA classifier for this pointset.

**Theorem 10.** *Winner-take-all classifiers over linear functions trained according to the one-versus-all assumptions in Definition 9 can not learn all data sets consistent with some winner-take-all hypothesis.*

*Proof.* see Figure 1.

### 3.4 Comparison to Networks of Linear Threshold Gates (Perceptron)

It is possible to re-write the constraint classification algorithm in Section 3.2 in a way that allows a network of linear classifiers (e.g., SNoW, or a multiclass SVM) to run it, with only a slight modification to the update rule. Such a network has  $x \in \mathbb{R}^d$  as input and  $k$  outputs where the  $i$ -th output computes  $\mathbf{v}_i \cdot x$ .

Specifically, in an on-line case such as Perceptron or Winnow, given  $(x, y)$ , the update rule cycles through the outputs from 1 to  $k$  and when  $w_i \cdot x > w_y \cdot x$ , it “demotes” weights connected to output  $i$  and “promotes” weights connected to output  $y$ , while updating  $w_y \cdot x$  along the way.

Using a network in this way results in an algorithm very similar to the ultraconservative online algorithms presented in [CS01b]. Instead of viewing an example as positive for one class and negative for all others and training each output independently, as in OvA, only those outputs that have a higher value than the correct index are changed.

## 4 Generalization Bounds

When considering  $\mathcal{H}$  to be the class of linear sorting functions, a consistent classifier for constraint classification can be found via the use of a hyperplane to separate randomly generated points in  $\mathbb{R}^{kd}$ . Both VC-dimension-based (based on growth function), and margin-based bounds for the class of hyperplanes in  $\mathbb{R}^{kd}$  have been heavily studied[Vap98,AB99]. Unfortunately, these results cannot directly be applied here since the transformation produces points in  $\mathbb{R}^{kd}$  that are random, but not *independently* drawn. In this section, generalization bounds for linear threshold functions are used to indirectly bound generalization error of the original, low dimension, constraint classification problem.

### 4.1 Growth Function-Based Bounds

Although VC-dimension cannot be used explicitly for constraint classification problems, bounds based on the growth function can be used.

#### Preliminaries

**Definition 11 (Prediction Vector)** Let  $\mathcal{H}$  be a class of functions  $h : \mathcal{X} \rightarrow \mathcal{Y}$ . Given a set of  $m$  unlabeled examples  $P = (x_1, \dots, x_m) \in \mathcal{X}^m$  we denote the *prediction vector* for a function  $h$  as  $h(P) = (h(x_1), \dots, h(x_m))$ . The set of all prediction vectors for  $\mathcal{H}$  is  $\mathcal{H}(P) = \{h(P) \mid h \in \mathcal{H}\}$ .

The growth function is usually defined when labels come from  $\{-1, 1\}$ , however, the definition holds in more general settings as well.

**Definition 12 (Growth Function)** Let  $\mathcal{H}$  be a hypothesis class where  $h \in \mathcal{H}$  is a function  $h : \mathcal{X} \rightarrow \mathcal{Y}$ . The number of assignments to a set of *unlabeled* examples,  $P = (x_1, \dots, x_m) \in \mathcal{X}^m$ , by hypothesis class  $\mathcal{H}$  can be measured by the size of  $\mathcal{H}(P)$  as it gives the set of all possible labeling for a particular sample. We define this number as  $\mathcal{N}_{\mathcal{H}}(P) = |\mathcal{H}(P)|$ . The *growth function* for  $\mathcal{H}$  is  $\mathcal{G}_{\mathcal{H}}(m) = \sup_{P \in \mathcal{X}^m} \log \mathcal{N}_{\mathcal{H}}(P)$ .

Similarly, the number of (binary) assignments to a set of *labeled* examples  $P = ((x_1, y_1), \dots, (x_m, y_m)) \in \{\mathcal{X} \times \mathcal{Y}\}^m$  by  $\mathcal{H}$ , an indicator function for  $\mathcal{H}$ , can be measured by the size of  $\mathcal{E}(P, \mathcal{H})$ . We define this number as  $\mathcal{N}_{\mathcal{E}}(P) = |\mathcal{E}(P, \mathcal{H})|$ . The *growth function* for  $\mathcal{E}(\cdot)$  is  $\mathcal{G}_{\mathcal{E}}(m) = \sup_{P \in \mathcal{X}^m} \log \mathcal{N}_{\mathcal{E}}(P)$ .

**Observation 13** For a given set of  $m$  labeled examples,  $P = ((x_1, y_1), \dots, (x_m, y_m)) \in \{\mathcal{X} \times \mathcal{Y}\}^m$  and a hypothesis class  $\mathcal{H}$ , the size of the set of all indicator vectors is smaller than the size of the set of all prediction vectors,  $\mathcal{N}_{\mathcal{E}}(P) = |\mathcal{E}(P, \mathcal{H})| \leq |\mathcal{H}(P_{\mathcal{X}})| = \mathcal{N}_{\mathcal{H}}(P_{\mathcal{X}})$ , where  $P_{\mathcal{X}} = (x_1, \dots, x_m)$  are the examples from  $P$  without their labels. Therefore, the maximum number of assignments by the indicator functions is bounded by the maximum number of assignments by  $\mathcal{H}$ . Namely,

$$\sup_{P \in \{\mathcal{X} \times \mathcal{Y}\}^m} \mathcal{N}_{\mathcal{E}}(P) = \sup_{P \in \{\mathcal{X} \times \mathcal{Y}\}^m} |\mathcal{E}(P, \mathcal{H})| \leq \sup_{P \in \mathcal{X}^m} |\mathcal{H}(P)| = \sup_{P \in \mathcal{X}^m} \mathcal{N}_{\mathcal{H}}(P).$$

Thus,  $\mathcal{G}_{\mathcal{E}}(m) \leq \mathcal{G}_{\mathcal{H}}(m)$ . Furthermore, notice when  $\mathcal{H}$  is the class of linear sorting functions the indicator function,  $\mathcal{E}(P, \mathcal{H})$ , can be interpreted in the constraint classification setting. In this case, the above inequalities still hold.

As a consequence of Sauer's lemma, the following lemma holds, as stated in [AB99].

**Lemma 2 (Corollary 3.8 [AB99]).** For a function class  $\mathcal{H}$  with VC-dimension  $d$ , if  $m > d$ , then

$$d < G_{\mathcal{H}}(m) < d \log_2(em/d)$$

**Growth Function for Linear Sorting Functions** When considering binary classification tasks, the fundamental characteristic of a hypothesis class  $\mathcal{H}$  used to bound generalization error is that for any set of examples there exists only a finite number of *effective* hypotheses. That is, there are only limited number of ways that a set of  $m$  examples can be labeled according to  $\mathcal{H}$ . This property is what facilitates the usage of  $\mathcal{G}_{\mathcal{H}}(m)$  (recall Definition 12) to bound generalization error. When assigning labels from  $\{-1, 1\}$ , the trivial bound for the number of possible assignments is  $2^m$ , which can be extended easily to  $(k!)^m$  when assigning labels from  $S^k$ . In this section, we show that there exists a bound of  $(emk/d)^{kd}$  when  $\mathcal{H}$  is the class of linear sorting functions.

It is also important to notice that all generalization bounds presented in this subsection are fundamentally based on  $\mathcal{G}_{\mathcal{E}}(m)$ , and only by Observation 13, on  $\mathcal{G}_{\mathcal{H}}(m)$ .

**Lemma 3.** *Let  $\mathcal{H}$  be the class of linear sorting functions over  $k$  weight vectors, each in  $\mathbb{R}^d$ , then  $\mathcal{G}_{\mathcal{H}}(m) \leq kd \log(emk/d)$ .*

*Proof.* Let  $P = (x_1, \dots, x_m)$  be a set of  $m$  unlabeled examples where  $x_i \in \mathbb{R}^d$ . Similar to the definition of  $\mathbf{P}(x, y)$  in Definition 8,  $\mathbf{P}^U(x)$  is defined for unlabeled examples as

$$\mathbf{P}^U(x) = \left\{ (\text{Vec}(x, i, j)) \mid (i, j) \in \{1, \dots, k\} \times \{1, \dots, k\} \right\} \setminus \mathbf{0}^{kd},$$

and  $\mathbf{P}^U(P) = \bigcup_{x \in P} \mathbf{P}^U(x)$ .

Let  $\mathcal{H}'$  be the class of separating hyperplanes in  $\mathbb{R}^{kd}$ . For every linear sorting function  $h \in \mathcal{H}$ , there is a linear threshold function  $h' \in \mathcal{H}'$  which labels  $\text{Vec}(x, i, j)$  as positive if and only if  $i$  precedes  $j$  in  $h(x)$ . If  $h(x) = \text{argsort}_{\{1, \dots, k\}} w_i \cdot x$ , then  $h'(x') = (w_1, \dots, w_k) \cdot x'$ , where  $x' \in \mathbb{R}^{kd}$ , satisfies this property. Notice that  $h'(\text{Vec}(x, i, j)) = w_i \cdot x - w_j \cdot x > 0$  when  $w_i \cdot x > w_j \cdot x$ , precisely when  $i$  precedes  $j$  according to  $h(x)$ .

Therefore, we can bound  $\mathcal{G}_{\mathcal{H}}(m)$  by  $\mathcal{G}_{\mathcal{H}'}((k-1)^2 m)$  since the number of examples in  $\mathbf{P}^U(P)$  is  $(k-1)^2 m$ . Of course  $\mathcal{G}_{\mathcal{H}'}((k-1)^2 m)$  is an over estimate since not all points in  $\mathbb{R}^{kd}$  can be examples in  $\mathbf{P}^U(P)$  (every example in  $\mathbf{P}^U(P)$  has  $(k-1) * d$  zeros, for example). Since the VC-dimension of  $\mathcal{H}'$  is  $kd - 1$ , by Lemma 2 and simple algebra,

$$kd < \mathcal{G}_{\mathcal{H}'}((k-1)^2 m) < kd \log(emk/d)$$

**Theorem 14.** *For any  $0 < \delta < 1$ , any  $h \in \mathcal{H}$ , the class of linear sorting functions over  $k$  linear functions in  $\mathbb{R}^d$ , where  $h : \mathbb{R}^d \rightarrow S^k$ , given  $P = ((x_1, y_1), \dots, (x_m, y_m))$ , a sample of size  $m$  drawn i.i.d. from  $\mathcal{D}_{\mathbb{R}^d \times S^k}$ , with probability at least  $1 - \delta$ ,*

$$\text{err}(h) \leq \text{err}(P, h) + \sqrt{4 \frac{kd \log(2emk/d) - \log \delta/4}{m}} + \frac{1}{m}$$

*Proof.* The proof follows from proof of [Vap98, Theorem 4.1]. This bound is a consequence of bounding the probability that any hypothesis taken from  $\mathcal{H}$  will differ by more than  $\varepsilon$  on two finite half-samples. Since we are bounding the probability that the two errors will differ, it suffices to consider only how many different possible error vectors might be generated when allowed to choose any  $h \in \mathcal{H}$ . Since the number of ways that  $\mathcal{H}$  can err on any given example set is less than the total possible assignments of labels from  $\mathcal{H}$  on that set, it is possible to phrase the generalization bounds based on the multiclass growth function as well as the binary growth function as is done by Vapnik. In particular, interpreting the bound of Lemma 3 as a bound on the number of indicator vectors generated by  $\mathcal{H}$  on a sample of size  $m$ , as suggested in Observation 13, and plugging it into Vapnik's proof, results in the required bound. Indeed,

$$\begin{aligned} \text{err}(h) &\leq \text{err}(P, h) + \sqrt{4 \frac{\mathcal{G}_{\mathcal{H}}(2m) - \log \delta/4}{m}} + \frac{1}{m} \\ &\leq \text{err}(P, h) + \sqrt{4 \frac{kd \log(2emk/d) - \log \delta/4}{m}} + \frac{1}{m} \end{aligned}$$

The bound of Theorem 14 can be extended also for the case where there is no error on the sample (i.e.  $err(P, h) = 0$ ).

**Theorem 15.** *For any  $0 < \delta < 1$ , any  $h \in \mathcal{H}$ , the class of linear sorting functions over  $k$  linear functions in  $\mathbb{R}^d$ , where  $h : \mathbb{R}^d \rightarrow S^k$ , given  $P = ((x_1, y_1), \dots, (x_m, y_{size}))$ , a sample of size  $m$  drawn i.i.d. from  $\mathcal{D}_{\mathbb{R}^d \times \bar{S}^k}$ , with probability at least  $1 - \delta$ , the inequality*

$$err(h) \leq err(P, h) + 2 \frac{\mathcal{G}_{\mathcal{H}}(2m) - \log \delta/4}{m} \left( 1 + \sqrt{\left( 1 + \frac{m \cdot err(P, h)}{\mathcal{G}_{\mathcal{H}}(2m) - \log \delta/4} \right)} \right)$$

holds true, where  $\mathcal{G}_{\mathcal{H}}(2m) = kd \log(2emk/d)$  (see Definition 12).

*Proof.* Omitted, as it follows by the same argumentation used in Theorem 14 and modification to the proof of [Vap98, Theorem 4.2] in a straightforward fashion.

**Corollary 1.** *To guarantee that the sampled error differ from the true error by less than  $\varepsilon$  with probability at least  $1 - \delta$ , it is sufficient to draw  $m > m(\varepsilon, \delta)$  examples drawn i.i.d. from  $\mathcal{D}_{\mathbb{R}^d \times \bar{S}^k}$  where,  $m(\varepsilon, \delta) = O\left(\frac{1}{\varepsilon^2} \max(kd \log \frac{kd}{\varepsilon^2}, \log \frac{1}{\delta})\right)$ .*

*If we are able to find a classifier  $h$  which is consistent on all the examples (i.e.  $err(P, h) = 0$ ), then to achieve true error less than  $\varepsilon$ , we need to pick  $m > m_1(\varepsilon, \delta)$ , where  $m_1(\varepsilon, \delta) = O\left(\frac{1}{\varepsilon} \max(kd \log \frac{kd}{\varepsilon^2}, \log \frac{1}{\delta})\right)$ .*

*Proof.* Follows by simple algebra from Theorem 14 and Theorem 15.

## 4.2 Margin-Based Generalization Bounds

A multiclass support vector machine can be implemented by learning a maximal margin hyperplane in  $\mathbb{R}^{kd}$  for the expanded point set defined in Definition 8. This hyperplane will maximize a notion of margin defined below that is also used in previous multiclass support vector machine work[CS00, WW99]. This subsection develops generalization bounds for our simple multiclass SVM implementation.

**Definition 16 (Binary Margin)** The margin of an example  $(x_i, y_i)$ , where  $(x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}$ , with respect to a separating hyperplane,  $h(x) = w \cdot x$ , where  $w, x \in \mathbb{R}^d$ , is defined to be

$$\gamma_i = y_i h(x_i) = y_i (w \cdot x_i).$$

The minimal margin over a set of examples,  $P = ((x_1, y_1), \dots, (x_m, y_m))$ , is defined to be

$$\text{mar}_P(h) = \min_{(x_i, y_i) \in P} \gamma_i.$$

In light of the transformation described in Section 3.1, a natural extension of binary margin to the constraint case can be derived. The constraint margin of a set of a set of examples,  $P$ , is simply the binary margin of a hyperplane in high dimension of  $\mathbf{P}(P)$ .

**Definition 17 (Constraint Margin)** The margin of an example  $(x_i, y_i)$ , where  $(x_i, y_i) \in \mathbb{R}^d \times \bar{S}^k$ , with respect to a linear sorting function,  $h(x) = \text{argsort}_{\{1, \dots, k\}} w_i \cdot x$ , where  $w_i, x \in \mathbb{R}^d$ , is defined to be

$$\gamma_i = \min_{(j, j') \in y_i} (w_j \cdot x_i - w_{j'} \cdot x_i).$$

The minimal margin over a set of examples,  $P = ((x_1, y_1), \dots, (x_m, y_m))$ , is defined to be

$$\text{mar}_P(h) = \min_{(x_i, y_i) \in P} \gamma_i.$$

Given some linear threshold function achieving large margin ( $\geq \gamma$ ) on  $m$  randomly drawn input data, standard generalization bounds are known [CST00]. With probability  $1 - \delta$ , generalization error is less than

$$\text{err}_{\mathcal{D}}(h) = \text{epsilon}(\gamma, m, \delta, R) \leq \frac{2}{m} \left( \frac{64R^2}{\gamma^2} \log \frac{em\gamma}{8R^2} \log \frac{32m}{\gamma^2} + \log \frac{4}{\delta} \right), \quad (1)$$

where the size of any example is less than  $R$ .

**Theorem 18 (Constraint Margin Bound).** *Consider real-valued linear sorting functions  $\mathcal{H}$  with  $\sum_{i=1, \dots, k} \|w_i\| = 1$ , where  $h : \mathbb{R}^d \rightarrow S^k$ , and fix  $\gamma \in \mathbb{R}^+$ . For any probability distribution  $\mathcal{D}$  on  $\mathbb{R}^d \times \bar{S}_C^k$  with support in a ball of radius  $R$  around the origin, with probability  $1 - \delta$  over  $m$  random examples  $P$ , any hypothesis  $h \in \mathcal{H}$  that has constraint margin  $\text{mar}_P(h) \geq \gamma$  on  $P$  has error no more than*

$$\text{err}_{\mathcal{D}}(h) = \epsilon(\gamma, m, \delta, R, C) \leq \frac{2C}{m} \left( \frac{256R^2}{\gamma^2} \log \frac{em\gamma}{32R^2} \log \frac{32m}{\gamma^2} + \log \frac{4}{\delta} \right)$$

provided  $m > \frac{2}{\epsilon}$  and  $\frac{256R^2}{\gamma^2} < m$ .

*Proof.* Consider  $\mathbf{P}(P)$  in  $\mathbb{R}^{kd} \times \{-1, 1\}$  and notice that  $|\mathbf{P}(P)| \leq Cm$ . It is not possible to apply Equation 1 directly because examples in  $\mathbf{P}(P)$  are not independently generated. Therefore, a new distribution  $\mathcal{D}'$  over examples in  $\mathbb{R}^{kd}$  is used to generate a set of  $m$  examples,  $\mathbf{P}'(P)$ , in  $\mathbb{R}^{kd} \times \{-1, 1\}$  based on the original set  $P$ .

For each example  $(x, y) \in P$ , define  $\mathbf{P}'(x, y)$  as a single point in  $\mathbb{R}^{kd} \times \{-1, 1\}$  chosen uniformly at random from  $\mathbf{P}(x, y)$ . Then define

$$\mathbf{P}'(P) = \bigcup_{(x, y) \in P} \mathbf{P}'(x, y).$$

For each example generated randomly according to  $\mathcal{D}'$ , Equation 1 can be applied to bound the chance of making an error. Furthermore, since an error is made on example  $(x, y)$  if any example from  $\mathbf{P}(x, y)$  is classified incorrectly, it is necessary to guarantee that no error is made. If  $C$  is the maximum number of constraints per example, then by the union bound and by noticing that the size of any example in  $\mathbf{P}(x, y)$  is less than  $2R$ , the theorem follows.

**Observation 19** *Because all learning problems described in Table 1 can be mapped to constraint classification with a fixed number of constraints, the bound in Theorem 18 applies. For multiclass classification, there are  $k - 1$  constraints and the above bound is similar to those given in [CS00, WW99]. The number of constraints may also be limited by the problem at hand, as is the case with context-sensitive spelling correction where each example may have at most three or four constraints, and the bounds will be vastly improved.*

## 5 Conclusions

The view of multiclass classification presented here simplifies the implementation, analysis, and understanding of many preexisting approaches. Multiclass support vector machines, ultraconservative online algorithms, and traditional one-versus-all approach can be cast in this framework. Furthermore, this view allows for a very natural extension of multiclass classification to constraint classification – capturing within it complex learning tasks such as multilabel classification and ranking problems. Because constraint classification is a very intuitive approach and its implementation can be carried out by any discriminant technique and not only optimization techniques, we think it will have useful real-world applications.

## References

- AB99. M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge, England, 1999.
- ADW94. Chidanand Apte, Fred Damerau, and Sholom M. Weiss. Automated learning of decision rules for text categorization. *Information Systems*, 12(3):233–251, 1994.
- ASS00. E. Allwein, R.E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proc. 17th International Conf. on Machine Learning*, pages 9–16. Morgan Kaufmann, San Francisco, CA, 2000.
- BCHL95. S. Ben-David, N. Cesa-Bianchi, D. Haussler, and P. Long. Characterizations of learnability for classes of  $0, \dots, n$ -valued functions. *J. Comput. Sys. Sci.*, 50(1):74–86, 1995.
- Bri94. E. Brill. Some advances in transformation-based part of speech tagging. In *AAAI, Vol. 1*, pages 722–727, 1994.
- CCRR99. A. Carlson, C. Cumby, J. Rosen, and D. Roth. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department, May 1999.
- CS00. K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learning Theory*, pages 35–46, 2000.
- CS01a. K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Machine Learning Research*, 2(December):265–292, 2001.

- CS01b. K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. In *COLT/EuroCOLT*, pages 99–115, 2001.
- CST00. Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.
- DKR97. I. Dagan, Y. Karov, and D. Roth. Mistake-driven learning in text categorization. In *EMNLP-97, The Second Conference on Empirical Methods in Natural Language Processing*, pages 55–63, 1997.
- HT98. T. Hastie and R. Tibshirani. Classification by pairwise coupling. In *NIPS-10, The 1997 Conference on Advances in Neural Information Processing Systems*, pages 507–513. MIT Press, 1998.
- Jel98. F. Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts, 1998.
- Koh01. T. Kohonen. *Self-Organizing Maps*. Springer Verlag, New York, 3rd edition, 2001.
- LBD<sup>+</sup>89. Y. Le Cun, B. Boser, J. Denker, D. Hendersen, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:pp 541, 1989.
- LS97. D. Lee and H. Seung. Unsupervised learning by convex and conic coding. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 515. The MIT Press, 1997.
- Maa00. W. Maass. On the computational power of winner-take-all. *Neural Computation*, 12(11):2519–2536, 2000.
- Rot98. D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *Proc. of AAAI*, pages 806–813, 1998.
- RZ98. D. Roth and D. Zelenko. Part of speech tagging using a network of linear separators. In *COLING-ACL 98, The 17th International Conference on Computational Linguistics*, pages 1136–1142, 1998.
- Sch97. R.E. Schapire. Using output codes to boost multiclass learning problems. In *Proc. 14th Internat. Conf. on Machine Learning*, pages 313–321. Morgan Kaufmann, 1997.
- Vap98. V. Vapnik. *Statistical Learning Theory*. Wiley, 605 Third Avenue, New York, New York, 10158-0012, 1998.
- WW99. J. Weston and C. Watkins. Support vector machines for multiclass pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, 4 1999.