A Cutting-Plane Algorithm for Learning from Ambiguous Examples

Stuart Andrews and Thomas Hofmann

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-06-06
April 2006

A Cutting-Plane Algorithm for Learning from Ambiguous Examples

Stuart Andrews STU@CS.BROWN.EDU
Department of Computer Science, Brown University, Providence, RI, 02912

Thomas Hofmann HOFMANN@INT.TU-DARMSTADT.DE
Computer Science Department TU Darmstadt, Dolivostrasse 15, 64293 Darmstadt, Germany

Abstract

We present a novel algorithm developed within the risk minimization framework for learning from ambiguous and unlabeled examples as encountered in multi-instance, transductive and semi-supervised learning. For these problems, risk minimization involves optimization over a non-convex feasible region defined by the ambiguous examples. Our method generalizes 1-norm SVM. We adapts lift-and-project cutting-planes from integer-programming to approximate the convex hull of the feasible region to arbitrary precision. The resulting classifier is competitive on several benchmarks, and in particular, outperforms existing algorithms on text categorization tasks due to its feature selectivity. This algorithm demonstrates that risk minimization can be used to effectively disambiguate and learn from ambiguous examples.

1. Introduction

A basic requirement of traditional supervised learning algorithms is that the inputs consist of instance-label pairs that are *examples* of the concept to be learned. However, for certain problems (i.e. multiple-instance, transductive, and semi-supervised learning) the inputs that are readily available are *ambiguous*, having several interpretations as instance-label pairs.

Multiple-instance learning addresses the problem of learning from inputs that are ambiguous due to polymorphism. The inputs, denoted $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{K(\mathbf{X})}\} \in 2^{\mathbb{X}}$ with $\mathbf{x} \in \mathbb{X}$, are called *multi-instances*; others call them *bags* on occasion. The instance-level labels $y(\mathbf{x}_1), \ldots, y(\mathbf{x}_{K(\mathbf{X})}) \in \mathbb{Y}$ are not observed directly but must be inferred through the multi-instance label $Y \in \mathbb{Y}$ which models a "contains-a" relationship. For T/F labels, the multi-instance label is a logical disjunction of the instance-level labels. The pairs $(\mathbf{X}, Y)$ are *ambiguous example*s. Viewed another way, the ambiguous example $(\mathbf{X}, Y)$ has several interpretations $(\mathbf{x}_1, Y), \ldots, (\mathbf{x}_{K(\mathbf{X})}, Y)$, and at least one must be valid.

Transductive and semi-supervised inductive inference deal with the problem of learning from examples, a subset of which are ambiguous because they are not labeled. For T/F labels, there are two interpretations, $(\mathbf{x}, T)$ and $(\mathbf{x}, F)$, for unlabeled examples. Under certain weak assumptions, we can map unlabeled inputs to ambiguous examples $(\mathbf{X}, Y) \equiv (\{\mathbf{x}, -\mathbf{x}\}, T)$, reducing these problems to multiple-instance learning.

Given a training set consisting of ambiguous examples $\mathcal{D} = \{(\mathbf{X}_i, Y_i) \,|\, i = 1, \ldots, m\}$, the goal is to learn a classifier for individual instances $\mathbf{x}$ that predicts labels $\hat{y}(\mathbf{x})$, and hence $\hat{Y}(\mathbf{X}) = \bigvee_{\mathbf{x} \in \mathbf{X}} \hat{y}(\mathbf{x})$, that are consistent with the labels Y in the training set. Notice the asymmetry in learning objective due to the disjunction: if the training label $Y = T$ then *at least one* interpretation must satisfy $\hat{y}(\mathbf{x}) = Y$ while if the training label $Y = F$ then *all* interpretations must satisfy the same. The process of selecting of an interpretation $(\mathbf{x}, Y), \mathbf{x} \in \mathbf{X}$ is called *disambiguation*. The key intuition that we use is that the true interpretation of ambiguous inputs is self-reinforcing.

1.1. Related Work

Allowing for ambiguity within a learning problem has several advantages related to the collection of training data. For this reason, researchers have formulated solutions to many problems using multiple-instance learning (MIL). For example, in text classification and content-based image categorization (Andrews et al., 2003; Chen & Wang, 2004; Ray & Craven, 2005), it is often easier for a human to provide binary labels for whole documents or images, than it is to annotate the topics of individual paragraphs or to provide

geometric annotations within the image. In certain drug-design applications, the labels are ambiguous because the reactive conformations of a molecule cannot be distinguished from those that are unreactive (Dietterich et al., 1997). We refer the reader to (Ray & Craven, 2005) for a survey of applications and empirical comparison of MIL algorithms. Alternatively, several researchers have considered transductive learning (Bennett & Demiriz, 1999; Bie & Cristianini, 2003; Joachims, 1999) for text categorization and optical character recognition, among other applications.

To develop our approach, we considered the principle of *structural risk minimization* (Vapnik, 1998). As demonstrated in (Andrews et al., 2003; Bennett & Demiriz, 1999; Joachims, 1999), the structural risk minimization (SRM) principle can be used in order to derive max-margin algorithms for transductive and multiple-instance learning problems. These approaches perform *inference*, whereby a set of consistent instance-label pairs is derived from ambiguous inputs, by optimizing over 0-1 integer variables. Two methods (Andrews et al., 2003; Joachims, 1999) are based on heuristic local search methods, and are susceptible to getting stuck in local minima. On the other hand, branch-and-cut, which is a global optimization technique for integer-programming, is not practical for problems with large numbers of variables, as observed in (Bennett & Demiriz, 1999).

Recently, an alternative convex programming solution was proposed (Andrews & Hofmann, 2004). Here, the authors use techniques from disjunctive programming (Balas, 1985) to formulate a convex approximation to the problem. Disjunctive programming was developed to solve 0-1 programs in the area combinatorial optimization, and excellent surveys are available (Balas, 2001; Balas, 1985).

Also, we mention the relaxed online max-margin algorithm (ROMMA) due to (Li & Long, 2002). This work motivated our pursuit of a sequential cutting-plane technique. ROMMA is a supervised learning algorithm.

1.2. Overview of Our Approach

Following (Andrews et al., 2003; Andrews & Hofmann, 2004), we use the structural risk principle to derive a max-margin formulation of the multiple-instance learning problem. This results in a non-convex formulation of multiple-instance learning that is a natural extension of the 1-norm SVM (Zhu et al., 2004). In this paper, we avoid the introduction of integer variables to our representation, and instead show how to construct a linear programming relaxation to this problem. In theory, this step simply replaces the non-convex feasible region with its convex hull. In practice, we use a cutting-plane algorithm to construct a compact sequential approximation to the convex hull using linear inequalities in the original feature space. These inequalities are generated using a technique based on *lift-and-project* cuts due to (Balas, 2001). This differs significantly from the approach of (Andrews & Hofmann, 2004) which uses the technique of lifting to represent an enclosing set called the *hull-relaxation*. Moreover, the authors of (Andrews & Hofmann, 2004) were not able to scale their algorithm to large data sets, as we do here. For the sake of comparison, ROMMA uses a 2-norm SVM formulation and approximates the convex hull using a single cutting-plane. We embed our cutting-plane algorithm within a boosting framework for feature selection. In this way, our learning algorithm is able to consider high-dimensional feature mappings required to generate expressive linear classifiers. Since the lift-and-project method is the central component of the cutting-plane algorithm, we call the entire algorithm LNPBoost.

2. Risk Minimization for MIL

For convenience of notation in the sequel we will assume that labels belong to $\mathbb{Y} = \{+1, -1\}$, with $+1$ corresponding to a (T) logical truth value. We define a classifier for individual instances $\hat{y} = \operatorname{sgn} f(\mathbf{x})$ by selecting a real-valued discriminant function $f : \mathbf{x} \in \mathbb{X} \to \mathbb{R}$. In our case, the admissible discriminant functions $f \in \mathcal{F}$ will be linear $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$ parameterized by a *weight vector* $\mathbf{w}$ with $\|\mathbf{w}\| = 1$ (i.e. a hyperplane through the origin). We define the *functional margin of an* **unambiguous** *training example* by $\gamma(\mathbf{x}, y) = yf(\mathbf{x})$. Recalling that the multi-instance labels (T/F) are computed as the disjunction of the individual instance labels, the classifier for individual instances naturally extends to one for multi-instances as follows $\hat{Y} = \operatorname{sgn} \max_{\mathbf{x} \in \mathbf{X}} f(\mathbf{x})$. Note that the "most positive" instance has a significant role, since alone, this member of the multi-instance is sufficient to determine the multi-instance label. For this reason, we define the *functional margin of an* **ambiguous** *example* by $\gamma(\mathbf{X}, Y) = Y \max_{\mathbf{x} \in \mathbf{X}} f(\mathbf{x})$. Disambiguation of the ambiguous example $(\mathbf{X}, Y)$ is encoded implicitly in the definition through the selection of the "most positive" instance.

Our goal is to find discriminant function $f(\mathbf{x})$ that, when used to classify the individuals member instances, yields a disambiguation that is consistent with the observed multi-instance labels Y. Since many discriminant functions may succeed in doing so, the SRM

principle adapted for multiple-instance learning selects the one that is most confident as follows:

Proposition 1 (SRM for MIL) *The optimal discriminant function $f \in \mathcal{F}$ is the one that maximizes the minimum functional margin of ambiguous examples over the training set $\gamma(\mathcal{D}) = \min_{1 \le i \le m} \gamma(\mathbf{X}_i, Y_i)$.*

To justify this use of the SRM principle to the multiple-instance learning problem, observe that the chosen discriminant function $f(\mathbf{x})$ obtains the lowest possible expected error bound after disambiguation of the training data set. Therefore, SRM for MIL selects the multi-instance classifier with the lowest possible expected error.

The *geometric margin* is defined as the distance between the point $\mathbf{x}$ and the closest point $\mathbf{p}$ on the hyperplane $\langle \mathbf{w}, \mathbf{p} \rangle = 0$, measured with respect to the norm $\|\cdot\|'$ that is dual to $\|\cdot\|$ (Mangasarian, 1997). Because $\|\mathbf{w}\| = 1$ for $f \in \mathcal{F}$, SRM for MIL is maximizing the minimal *geometric margin*. As shown in (Mangasarian, 1997), there is an equivalent formulation in which the norm of the weight vector is minimized, while constraining all *functional margins* to be greater than or equal to 1. In this case, the constraints are called *margin constraints*.

Denoting a halfspace by $H(\mathbf{x}, \gamma) = \left\{\mathbf{w} \in \mathbb{R}^d | \langle \mathbf{w}, \mathbf{x} \rangle \ge \gamma\right\}$, the margin constraint for an **unambiguous** example $(\mathbf{x}, y)$ can be expressed using set notation as follows

$$\mathbf{w} \in H(y\mathbf{x}, 1).$$

Similarly, the following identity allows us to write the margin constraint for an **ambiguous** example $(\mathbf{X}, Y)$ in set notation

$$\mathbf{w} \in \left\{\mathbf{w} \in \mathbb{R}^d | Y \max_{\mathbf{x} \in \mathbf{X}} \langle \mathbf{w}, \mathbf{x} \rangle \ge 1\right\} = \bigcup_{\mathbf{x} \in \mathbf{X}} H(Y\mathbf{x}, 1).$$

Following (Andrews & Hofmann, 2004), we formulate a disjunctive program (DP) using the ∞-norm margin and its dual 1-norm on the weight vector. Without loss of generality, we will assume that the optimal solution lies in the positive orthant (see (Demiriz et al., 2002)), and furthermore, has 1-norm bounded by some a priori known constant M , i.e. $\mathbf{w} \in Q = \left\{\mathbf{w} \in \mathbb{R}^d | \mathbf{w} \ge 0, \|\mathbf{w}\|_1 \le M\right\}$. The optimization problem is

$$\text{DP} \quad \min_{\mathbf{w}} \|\mathbf{w}\|_1 \ \text{ s.t. } \mathbf{w} \in \bigcap_{i=1}^{m} \bigcup_{\mathbf{x} \in \mathbf{X}_i} H(Y_i\mathbf{x}, 1) \cap Q. \quad (1)$$

The non-convexity of the feasible region is the primary challenge of solving this optimization problem. To alleviate problems local-minima, we formulate an LP relaxation to the problem by replacing the feasible region with its convex hull $\mathbf{F} = \text{conv}\left[\bigcap_{i=1}^{m} \bigcup_{\mathbf{x} \in \mathbf{X}_i} H(Y_i\mathbf{x}, 1) \cap Q\right]$. This results in the following *relaxed* DP

$$\text{RDP} \quad \min_{\mathbf{w}} \|\mathbf{w}\|_1 \ \text{ s.t. } \mathbf{w} \in \mathbf{F}. \quad (2)$$

For linear discriminant functions, it is also possible to extend these definitions to allow for a soft-margin using a simple mapping: $\mathbf{x} \leftarrow (\mathbf{x}, Y_i\mathbf{e}_i/C), \forall \mathbf{x} \in \mathbf{X}_i, \Sigma_i \leftarrow C\xi_i, \mathbf{w} \leftarrow (\mathbf{w}, \Sigma)$, where $\mathbf{e}_i \in \mathbb{R}^m$ is zero apart from a 1 in the i-th dimension, and $\xi_i \ge 0$ is a slack variable. Each margin constraint (ambiguous or unambiguous) can be weakened by increasing the corresponding slack variable. Positive slack variables, scaled by a constant C, are penalized linearly in the objective.

3. The Cutting-Plane Algorithm

The key idea employed by cutting-plane algorithms is that we are not so much interested in representing the complete feasible region $\mathbf{F}$, but instead finding a good approximation to it near the optimal solution (Kelley, 1960; Nemhauser & Wolsey, 1988). Each iteration generates a hyperplane that "cuts-away" a slice of the approximation. We show that it is possible to construct an approximation to the convex $\mathbf{F}$ efficiently in this manner. We can also prove that, in the limit, the solution will be inside $\mathbf{F}$ (proof omitted due to lack of space). The proof is based on the work of (Owen & Mehrotra, 2001) on a sequential convexification algorithm for mixed-integer programming.

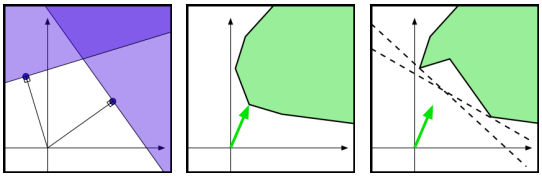
3.1. Sequential Convexification

The cutting-plane algorithm constructs a nested sequence of convex sets denoted $\mathbf{F}_t$ such that $\mathbf{F} \subseteq \mathbf{F}_t$. Each approximation $\mathbf{F}_t$ is defined by a finite intersection of halfspaces

$$\mathbf{F}_t = \bigcap_{i=1}^{N_t} H(\alpha_i, \beta_i). \quad (3)$$

Each defining hyperplane (α, β) is called a *cut*, and the set $\mathcal{C}_t = \{(\alpha_i, \beta_i) | i = 1, \ldots, N_t\}$ is called the *cut set*. The weight vector $\mathbf{w}_t$ is defined implicitly as the feasible vector with minimal norm $\mathbf{w}_t = \text{argmin}_{\mathbf{w} \in \mathbf{F}_t} \|\mathbf{w}\|_1$. Although $\mathbf{F}_t$ and $\|\mathbf{w}\|_1$ are convex, the minimum may not be unique since the 1-norm is not a strictly convex function.[1] This does not affect the convergence proper-

[1] Multiple minima exist iff the plane of points p satisfying $\langle \mathbf{1}, p \rangle = \|\mathbf{w}_t\|_1$ intersects $\mathbf{F}_t$ on a d-dimensional face of $\mathbf{F}_t$ for some $d \ge 1$.



(a) (b) (c)

Figure 1. The separation problem. (a) Disjunctive feasible region for an ambiguous example with two instances. (b) Current approximation $\mathbf{F}_t$ to feasible region $\mathbf{F}$. (c) Intersection of these regions and two examples of cuts (α, β) separating the intersection from the current solution. Note that the cuts approximate the convex hull of the intersection, which we denote by $\mathbf{D}_i(\mathbf{F}_t) = \text{conv}\left[\bigcup_{\mathbf{x}\in\mathbf{X}_i} H(Y_i\mathbf{x}, 1) \cap \mathbf{F}_t\right]$.

ties of the algorithm.

The initial cut set $\mathcal{C}_1$ is defined so that $\mathbf{F}_1 = Q$, and by the above definition, the initial solution is $\mathbf{w}_1 = \mathbf{0}$. On the t-th iteration, a single example (unambiguous or ambiguous) denoted $(\mathbf{X}_t, Y_t)$ is presented to the algorithm. An update to the feasible region $\mathbf{F}_t$ is performed when *the example can be used to derive a new cut*, parameterized by (α, β), that properly separates the current solution from $\mathbf{F}$. This is certainly possible if $\mathbf{w}_t \notin \mathbf{D}_i(\mathbf{F}_t) = \text{conv}\left[\bigcup_{\mathbf{x}\in\mathbf{X}_i} H(Y_i\mathbf{x}, 1) \cap \mathbf{F}_t\right]$ since we know that $\mathbf{F} \subseteq \mathbf{D}_i(\mathbf{F}_t)$ (see Figure 1). The problem of deriving such a cut, which is a *separation problem*, is discussed below.

A hyperplane with this separating property is called a *valid cut*. If a valid cut is generated, the cut set is augmented $\mathcal{C}_{t+1} \leftarrow \mathcal{C}_t \cup \{(\alpha_t, \beta_t)\}$, the updated region of feasibility obeys $\mathbf{F}_{t+1} \subset \mathbf{F}_t$. In particular, the new feasible region $\mathbf{F}_{t+1}$ excludes the current solution $\mathbf{w}_t \in \mathbf{F}_t \backslash \mathbf{F}_{t+1}$. The algorithm continues cycling through examples $(\mathbf{X}_t, Y_t)$ from the training data set as long as valid cuts can be generated. Unless the current solution $\mathbf{w}_t$ lies within $\mathbf{F}$, the algorithm will eventually solve a separation problem that generates a valid cut, thereby updating $\mathbf{w}_t$. Due to the role of convexity in each step of the sequence, the process is called *sequential convexification*.

To summarize, since $\mathbf{F}_t \supseteq \mathbf{F}$, each iteration solves a *relaxation* of the RDP problem in Eqn. 2, to obtain $\mathbf{w}_t$. Each cut *strengthens* the current relaxation of the RDP problem, and the objective increases monotonically. At convergence, $\mathbf{w}_t \in \mathbf{F}$ and we have a solution to RDP. In order to use CPLEX effectively, we solve the dual of the relaxed RDP. From this perspective, the separation sub-problem can be viewed as a *column generation* (Amor et al., 2004).

In order to generate the cuts for a given solution $\mathbf{w}_t$ and an **ambiguous** training example $(\mathbf{X}_t, Y_t)$ with the current feasible region $\mathbf{F}_t$, we consider the family of lift-and-project cuts introduced by (Balas, 2001).

3.2. Separation by Lift-and-Project Cuts

The following lemma gives a characterization of valid cuts for $\mathbf{D}_t(\mathbf{F}_t)$ that is a consequence of (a generalized version) of Farkas lemma (Balas, 2001; Rockafellar, 1970).

Lemma 1 *Let* $(\alpha_0^{\mathbf{x}}, \beta_0^{\mathbf{x}}) = (Y_t\mathbf{x}, 1)$ *for all* $\mathbf{x} \in \mathbf{X}_t$. *The cut defined by* $(\alpha, \beta) \in \mathbb{R}^{d+1}$ *is a valid cut if and only if, for all* $\mathbf{x} \in \mathbf{X}_t$, *there exists positive multipliers* $(u^{\mathbf{x}}, u_0^{\mathbf{x}}) \in \mathbb{R}^N \times \mathbb{R}$ *such that the following linear constraints are satisfied*

$$\alpha \geq \sum_{i=1}^{N} u_i^{\mathbf{x}}\alpha_i + u_0^{\mathbf{x}}\alpha_0^{\mathbf{x}} \quad \textit{and} \quad \beta \leq \sum_{i=1}^{N} u_i^{\mathbf{x}}\beta_i + u_0^{\mathbf{x}}\beta_0^{\mathbf{x}} \quad (4)$$

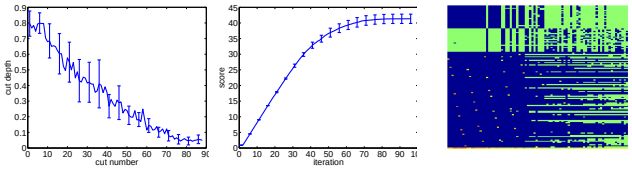
A lift-and-project cut is generated by maximizing the separation achieved by a valid cut. We define the signed *depth* of a cut (α_t, β_t) to be the signed functional distance from the solution $\mathbf{w}_t$ to the halfspace $H(\alpha_t, \beta_t)$, which is $\beta_t - \langle \alpha_t, \mathbf{w}_t \rangle$ as derived in (Mangasarian, 1997). Maximizing the depth over valid cuts can be formulated as an LP over cut parameters (α, β) and multipliers $(u^{\mathbf{x}}, u_0^{\mathbf{x}})$. In order to ensure that the LP has a finite optimum we normalize $\|\alpha\|_1 \leq C_{\text{LNP}}$, where C_{LNP} controls the complexity of the cut, as suggested in (Balas, 2001). Thus we arrive at lift-and-project cut generating LP:

$$\text{LNP} \quad \max_{\beta,\alpha,u^{\mathbf{x}}} \beta - \langle \alpha, \mathbf{w}_t \rangle \ \text{ s.t. } \ \|\alpha\|_1 \leq C_{\text{LNP}} \quad (5)$$

$$(\alpha, \beta) \ \text{ is valid.} \quad (6)$$

When the depth is positive, a valid cut exists. The converse is also true: if the point $\mathbf{w}_t \notin \mathbf{D}_t(\mathbf{F}_t)$, the convexity of $\mathbf{D}_t(\mathbf{F}_t)$ guarantees that a valid cut exists (Corollary 11.4.2 pg. 99, (Rockafellar, 1970)). On the other hand, if the depth is negative, we can be assured that the $\mathbf{w}_t \in \mathbf{D}_t(\mathbf{F}_t)$. The solution (α, β) can be thought of as the cut that would have been generated by the margin constraint of an unambiguous training example $\left(\left(\frac{Y}{\beta}\alpha\right), Y\right)$.

We note that the cut defined by $(\alpha_t, \beta_t) = (y_t\mathbf{x}_t, 1)$ for the **unambiguous** example $(\mathbf{x}_t, y_t)$ is valid and has depth equal to the standard hinge loss, as long as $\mathbf{w} \notin H(y_t\mathbf{x}_t, 1)$. These are the cuts used by the ROMMA algorithm (Li & Long, 2002).



(a) (b) (c)

Figure 2. Analysis of the cutting-plane algorithm on MUSK1. Error bars show 95% confidence intervals over 10 folds. (a) Cut depth. (b) Value of objective. (c) Generated cuts (columns, L to R). Upper bands correspond to features, while lower bands correspond to slack variables. Initial cuts are typically derived from unambiguous examples, including negatively labeled multi-instances. Ambiguous examples play a greater role in the later stages. A subset of slack variables become active in the later cuts, indicating which examples have tight margin constraints.

3.3. Convergence and Cut Set Regularization

For 0-1 mixed-integer programming problems, there exists a proof showing that the sequential convexification procedure converges in a finite number of steps (Balas, 2001). Unfortunately, this does not hold for general disjunctive programs, and we instead demonstrate asymptotic convergence empirically.

There is a trade-off between the size of the cut-set, the accuracy of the resulting classifier, and the running time, that may be controlled using various conditions. The size of the cut-set has an significant impact on running time, since the LNP sub-problem constitutes the majority of computation in the entire sequential convexification procedure. However, if our conditions are too strict, then the approximation $\mathbf{F}_t$ will be poor and classifier accuracy will drop. We implemented the following two techniques used by (Balas, 2001) for regulating the size of the cut-set.

The first technique is to avoid shallow cuts in favor of subsequent deeper ones. This idea is implemented by generating LNP cuts in rounds of 10 from which the single deepest cut is applied; the remaining cuts are discarded. Also, a cut selected for the round is discarded if its depth measured relative to the current objective is below a specified tolerance. The second technique is to avoid redundancy due to identical or near parallel cuts. The generation of near-parallel cuts especially near convergence, is a commonly observed behavior and the main cause of slow convergence of most cutting-plane algorithms (Barnhart et al., 1998; Amor et al., 2004). To alleviate this problem, we only add a cut to the cut-set when the angle made between it and any of the previous cutting-planes of the cut-set is sufficiently large. Empirically, the minimum-angle tolerance has a direct and noticeable effect on both the running time of the overall procedure and the accuracy of the resulting classifier.

Algorithm 1 Sequential convexification algorithm [SC]

input: $\mathcal{D}, \delta_A, \delta_D, \delta_S, N_{\text{LNP}}$
initialize: $\mathcal{C}_1 \leftarrow \{(\mathbf{e}_k, 0) : k = 1, \ldots, d\}$,
  $\mathbf{F}_1 \leftarrow Q$, $\mathbf{w}_1 \leftarrow \text{argmin}_{\mathbf{w} \in \mathbf{F}_1} \|\mathbf{w}\|_1$
repeat
  $(\alpha_t, \beta_t) \leftarrow (0, 0)$, $D_t \leftarrow 0$
  for $i = 1, \ldots, N_{\text{LNP}}$ **do**
    Get training example $(\mathbf{X}_t, Y_t)$
    $(\alpha_i, \beta_i) \leftarrow \text{LNP}(\mathbf{w}_t, \mathbf{F}_t, \mathbf{X}_t, Y_t)$
    if $\min_{(\alpha,\beta) \in \mathcal{C}_t} \measuredangle((\alpha_i, \beta_i), (\alpha, \beta)) \geq \delta_A$ **then**
      if $\text{depth}((\alpha_i, \beta_i); \mathbf{w}_t) \geq D_t$ **then**
        $(\alpha_t, \beta_t) \leftarrow (\alpha_i, \beta_i)$
        $D_t \leftarrow \text{depth}((\alpha_t, \beta_t); \mathbf{w}_t)$
      end if
    end if
  end for
  if $D_t \geq \delta_D * \|\mathbf{w}_t\|_1$ **then**
    $\mathcal{C}_{t+1} \leftarrow \mathcal{C}_t \cup \{(\alpha_t, \beta_t)\}$, $\mathbf{F}_{t+1} \leftarrow \mathbf{F}_t \cap H(\alpha_t, \beta_t)$
    $\mathbf{w}_{t+1} \leftarrow \text{argmin}_{\mathbf{w} \in \mathbf{F}_{t+1}} \|\mathbf{w}\|_1$
  end if
  $t \leftarrow t + 1$
until $\|\mathbf{w}_{t+1}\|_1 - \|\mathbf{w}_t\|_1 < \delta_S * \|\mathbf{w}_t\|_1$

The procedure is terminated if the relative increase of the objective falls below a specified tolerance. The steps of the sequential convexification algorithm are depicted in algorithm box 1. Figure 2 provides a diagnostic view of the running of the cutting-plane algorithm.

4. Projection and Feature Selection

We use a simple projection operation in order to compute an upper bound on DP in Eqn. 1. Projection maps the current solution, which lies in the convex region $\mathbf{F}$, onto the original region of feasibility $\bigcap_{i=1}^{m} \bigcup_{\mathbf{x} \in \mathbf{X}_i} H(Y_i \mathbf{x}, 1) \cap Q$. Since the solution value of the projected model and the solution value of the RDP give upper and lower bounds on the true DP minimum, the difference in these values indicates how close the projected solution is to being a global minima.

The second reason for projection, is that it is useful for performing feature selection. In this way, computation occurs over a significantly reduced feature set.

One way to project the solution is to consider a restricted version of DP. We first select the most pos-

Algorithm 2 Feature selection algorithm. Steps 4 and 6 are computed using only features from A_T.

input: $\mathcal{D}, \delta_A, \delta_D, \delta_S, N_{\text{LNP}}$
initialize: $A_1 \leftarrow random$, $\mathbf{w}_0 \leftarrow \inf$
repeat
  $\mathbf{w}_T \leftarrow \text{SC}\left(\mathcal{D}, \delta_A, \delta_D, \delta_S, N_{\text{LNP}}; A_T\right)$
  $\mathcal{D}^* \leftarrow \left\{\left(\text{argmax}_{\mathbf{x}\in\mathbf{X}_i} \left\langle \mathbf{w}_T, h_{A_T}(\mathbf{x})\right\rangle, Y_i\right) | \forall i\right\}$
  $u \leftarrow \text{PRDP}\left(\mathcal{D}^*; A_T\right)$
  $k \leftarrow \text{argmax}_{k\in\bar{A}_T} \text{score}\left(k; u, \mathcal{D}\right)$
  $A_{T+1} \leftarrow A_T \cup \{k\}$
  $T \leftarrow T+1$
until $\|\mathbf{w}_{T-1}\|_1 - \|\mathbf{w}_T\|_1 < \delta_c * \|\mathbf{w}_{T-1}\|_1$

itive instance $\mathbf{x}_i = \text{argmax}_{\mathbf{x}\in\mathbf{X}_i} f(\mathbf{x})$ of each multi-instance using the solution $f(\mathbf{x})$ of RDP. Discarding the remaining instances, we create a data set of *unambiguous* examples $\mathcal{D}^* = \{(\mathbf{x}_i, Y_i) | i = 1, \ldots, m\}$. The projection from the RDP solution is then obtained by solving

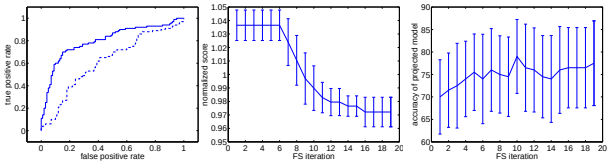
$$\text{PRDP} \quad \min_{\mathbf{w}} \|\mathbf{w}\|_1 \quad \text{s.t. } \mathbf{w} \in \bigcap_{i=1}^{m} H\left(Y_i \mathbf{x}_i, 1\right) \cap Q. \quad (7)$$

This is equivalent to the 1-norm SVM formulation using $\mathcal{D}^*$. We observed that the performance of the PRDP solution exceeds that of the RDP solution (see Figure 3(a)).

In order to select new features, we follow the approach of LP-Boost (Demiriz et al., 2002). Features are added when they help to minimize the PRDP upper bound. An important characteristic of the solution to PRDP in Eqn. 7 is the fact that the dual variables u_i are now associated with *instances* in the training data set, instead of *cutting-planes*. This allows us to compute a *feature score*

$$\text{score}\left(k; u, \mathcal{D}\right) = \sum_{i=1}^{m} u_i Y_i h_k(\mathbf{x}) - 1 \quad (8)$$

where the notation $h_*(\mathbf{x})$ is used to denote the projection of $\mathbf{x}$ onto the indicated feature. To initialize the procedure, we begin with a small randomly selected subset of the features (typically 5), denoted by A_T. After solving the PRDP and retrieving its dual variables, we evaluate the score equation and select new features (typically 2) with greatest score $\left(k; u, \mathcal{D}\right) \geq 0$. We refer the reader to (Demiriz et al., 2002) for details. Box 2 shows the augmented algorithm. Figure 3 provides a detailed view of the inner workings of the algorithm.



(a) (b) (c)

Figure 3. Analysis of the effect of projection and boosting on the Elephant data set. (a) ROC curves for RDP (broken) and PRDP (solid) classifiers at convergence of cutting-plane algorithm. (b) RDP objective value vs. boosting round. (c) PRDP accuracy vs. boosting round. Error bars show 95% confidence intervals obtained over 10-folds.

5. Experiments

5.1. Methodology

We implemented LNPBoost in MATLAB[2] using the Java[3] interface in order to solve LP sub-problems using CPLEX[4]. We performed 10-fold crossvalidation experiments to evaluate two variants of LNPBoost: one with linear features, and one with a radial basis function (RBF) features. In both cases, features were preprocessed following (Weston et al., 2004). First, we centered and normalized each feature so that its variance on the training data was 1. For RBF features, the inputs were subsequently transformed as follows $\mathbf{x} \mapsto \left(h_1(\mathbf{x}), \ldots, h_M(\mathbf{x})\right)$, where $h_j(\mathbf{x}) = \exp\left(-\|\mathbf{x} - \mathbf{x}_j\|^2 / \sigma^2\right), \forall j$, and $\mathbf{x}_j$ is the j-th individual from the training data set. All experiments used the same default parameter settings for the cutting-plane algorithm: $C = 1$, $C_{\text{LNP}} = 1$, $N_{LNP} = 10$, $\delta_D = 10^{-5}$ and $\delta_S = 10^{-4}$. The minimum angle tolerance $\delta_A \in [0, \pi/4]$ was chosen large enough to ensure timely convergence. For boosting, we initialized with 5 randomly selected features, added 2 features in each boosting round. We used a single fixed setting for $\sigma \approx \sqrt{d}$.

5.2. Results

We tested the algorithm on MIL and transductive learning benchmarks. For MIL, we used two MUSK data sets[5], in addition to three Corel image data sets (Tiger, Elephant, Fox) and seven TREC9 text data sets (1,2,3,4,7,9,10)[6]. For transductive learning, we used the well-known database of USPS handwritten

[2] http://www.mathworks.com
[3] http://java.sun.com
[4] http://www.ilog.com
[5] http://www.ics.uci.edu/~mlearn/MLRepository.html
[6] http://www.cs.brown.edu/people/stu/mil/datasets.html

	MI examples		MI size		features	
	ambig	o.w.	max.	med.	dim.	%fill
MUSK1	47	45	40	4	166	100
MUSK2	39	63	1044	12	166	100
COREL	100	100	13	6.3	230	50.4
TREC9	200	200	19.5	8	$6.6e^4$	0.03
USPS	380	20	2	2	256	100

Table 1. Data set statistics. The "otherwise" column under "MI examples" reports the number of negatively labeled multi-instances. For transduction, this is the number of labeled examples. Values for the three Corel data sets (Tiger, Elephant and Fox) as well as the seven Trec9 data sets have been averaged since the data sets are so similar.

digits[7], following the experiment design of (Bie & Cristianini, 2003). A subsample containing 100 of each of the digits "0"-"3" was randomly drawn from the entire collection. Five samples from each digit category were labeled. "0" and "1" samples were labeled +1, while "2" and "3" samples were labeled −1. The remaining 380 digits were used to create multi-instance examples of size two as described in the introduction. We compare several key characteristics of these datasets in Table 1.

As a baseline for MIL , we use the results reported in (Andrews et al., 2003) for their mi-SVM and MI-SVM algorithms and for the EM-DD algorithm of (Zhang & Goldman, 2002). To date, these are the only algorithms that have been evaluated on all 12 data sets. Table 2 summarizes the results. Under the LNPBoost columns, we report the performance of the final PRDP (projected) solution.

On TREC9, LNPBoost with linear features obtains the highest accuracy in four cases, and is outperformed by a significant margin in only one case. On average, seven features are used to define the classifier. LNPBoost with linear features has the best accuracy on Elephant, and LNPBoost with RBF features obtains the highest accuracy on Fox and MUSK2. Note, however, that there are several algorithms that obtain accuracies in the range 85-93% for MUSK1 and MUSK2, as detailed in (Chen & Wang, 2004).

On the transductive learning task, LNPBoost with RBF features had an average AROC of 0.93 ± 0.03. As reported in (Bie & Cristianini, 2003), the inductive SVM (using labeled points only) obtained 0.75 ± 0.03 with an RBF kernel, and the relaxed transductive SVM proposed by the authors obtained 0.96 ± 0.03. Notice that our method learns a discriminant function suitable for classifying unseen test points.

[7]http://www.gaussianprocess.org/gpml/data/

	EM	mi	MI	LNPBoost	
	DD	SVM	SVM	lin	rbf
MUSK1	84.8	**87.4**R	77.9R	80.0	84.4
MUSK2	84.9	83.6R	84.3R	68.0	**85.0**
Tiger	72.1	78.9R	**84.0**L	79.5	75.0
Elephant	78.3	82.2L	81.4L	**85.0**	77.5
Fox	56.1	58.2L	59.4P	57.5	**59.5**
Tst 1	85.8	93.6L	93.9L	**94.8**	-
Tst 2	84.0	78.2L	84.5L	**86.0**	-
Tst 3	69.0	87.0L	85.1P	**90.3**	-
Tst 4	80.5	82.8L	**82.9**P	82.5	-
Tst 7	75.4	**81.3**L	78.7P	78.5	-
Tst 9	65.5	**67.5**L	63.7P	60.0	-
Tst 10	78.5	79.6L	81.0P	**82.3**	-

Table 2. Comparison of accuracies for the algorithm evaluated on Musk, Corel and TREC9 (tst 1, tst 2, ...) data sets using linear and radial basis features. Bold indicates the highest accuracies obtained per row. We report the best obtained accuracy for mi-SVM and MI-SVM, using (L/P) to indicate which kernel was used.

Better performance for all experiments may be possible by tuning parameters on a held-out validation set, in particular the slack penalty C, the cut complexity C_{LNP}, and σ for the RBF features. Confidence intervals of 95% for the reported accuracies were $\pm s$ where $3 \leq s \leq 10\%$. We believe the high variance is due to the heuristic projection method used to obtain the PRDP classifier. We intend to explore alternative projections and ways to combine more than one (Amor et al., 2004). Running times were between 30 minutes - 3 hours per fold on 1.7 GHz Athlon MP processors for all reported experiments. Over 90% of the time is spent solving LNP sub-problem instances.

6. Conclusions

We have proposed LNPBoost, a new algorithm developed within a risk minimization framework for learning from ambiguous examples. LNPBoost solves a convexified disjunctive program that is an extension of 1-norm SVM. A sequential convexification cutting plane algorithm was derived using lift-and-project cuts adapted from (Balas, 2001), and a proof of convergence has been adapted (Owen & Mehrotra, 2001). The resulting cutting-plane algorithm is augmented by projection and feature selection for efficiency and improved accuracy. Our experiments show that LNPBoost outperforms previous algorithms on many MIL benchmarks, and is also competitive on a transductive learning benchmark. Moreover, our formulation and cutting-plane algorithm admit an intuitive geometric understanding. LNPBoost demonstrates that risk minimization can be used to effectively disambiguate and learn from ambiguous exam-

ples.

LNPBoost is founded upon the notion of a disjunction, in order to explain *at least one* interpretation of an ambiguous input. As such, there is no inductive bias toward models that provide *coverage* if and when there are multiple valid interpretations (Ray & Craven, 2005). For this reason, our approach may be best suited for problems where one *and only one* interpretation is valid, as is the case in transductive inference.

While experiments on moderate sized data sets currently take less than 3 hours, there is much room for improvement. Advances in combinatorial optimization indicate that LNP cuts can be computed much more efficiently (Balas & Perregaard, 2003).

We would like to develop an online version, and to apply LNPBoost to the multi-class case.

References

Amor, H. B., Desrosiers, J., & Frangioni, A. (2004). *Stabilization in column generation* (Technical Report). GERAD.

Andrews, S., & Hofmann, T. (2004). Disjunctive programming boosting. *Advances in Neural Information Processing Systems*.

Andrews, S., Tsochantaridis, I., & Hofmann, T. (2003). Support vector machines for multiple-instance learning. *Advances in Neural Information Processing Systems*.

Balas, E. (1985). Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *Industrial and Applied Mathematics*, *6*, 466–486.

Balas, E. (2001). Projection and lifting in combinatorial optimization. *Computational Combinatorial Optimization*.

Balas, E., & Perregaard, M. (2003). A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer gomory cuts for 0-1 programming. *Mathematical Programming*.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*.

Bennett, K., & Demiriz, A. (1999). Semisupervised support vector machines. *Advances in Neural Information Processing Systems (NIPS*12)*.

Bie, T. D., & Cristianini, N. (2003). Convex methods for transduction. *Advances in Neural Information Processing Systems*.

Chen, Y., & Wang, J. Z. (2004). Image categorization by learning and reasoning with regions. *Machine Learning Research*, *5*, 913–939.

Demiriz, A., Bennett, K. P., & Shawe-Taylor, J. (2002). Linear programming boosting via column generation. *Machine Learning*, *46*, 225–254.

Dietterich, T. G., Lathrop, R. H., & Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*, 31–71.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *International Conference on Machine Learning* (pp. 200–209). Morgan Kaufmann, San Francisco, CA.

Kelley, J. E. (1960). The cutting-plane method for solving convex programs. *Industrial and Applied Mathematics*.

Li, Y., & Long, P. M. (2002). The relaxed online maximum margin algorithm. *Machine Learning*.

Mangasarian, O. (1997). *Arbitrary-norm separating hyperplane* (Technical Report). University of San Diego.

Nemhauser, G. L., & Wolsey, L. A. (1988). *Integer and combinatorial optimization*. Wiley-Interscience, New York, NY.

Owen, J. H., & Mehrotra, S. (2001). A disjunctive cutting plane procedure for general mixed-integer linear programs. *Mathematical Programming*, 437–448.

Ray, S., & Craven, M. (2005). Supervised versus multiple instance learning: An empirical comparison. *International Conference on Machine Learning*.

Rockafellar, R. T. (1970). *Convex analysis*. Princeton University Press.

Vapnik, V. (1998). *Statistical learning theory*. Wiley.

Weston, J., Elisseeff, A., Bakir, G., & Sinz, F. (2004). The spider.

Zhang, Q., & Goldman, S. A. (2002). EM-DD: An improved multiple-instance learning technique. *Advances in Neural Information Processing Systems*.

Zhu, J., Rosset, S., Hastie, T., & Tibshirani, R. (2004). 1-norm support vector machines. *Neural Information Processing Systems*.