Satisfying Real-world Goals with Dataset Constraints

Gabriel Goh
Dept. of Mathematics
UC Davis
Davis, CA 95616
ggoh@math.ucdavis.edu

Andrew Cotter, Maya Gupta
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
acotter@google.com
mayagupta@google.com

Michael Friedlander
Dept. of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1Z4
mpf@cs.ubc.ca

Abstract

The goal of minimizing misclassification error on a training set is often just one of several real-world goals that might be defined on different datasets. For example, one may require a classifier to also make positive predictions at some specified rate for some subpopulation (fairness), or to achieve a specified empirical recall. Other real-world goals include reducing churn with respect to a previously deployed model, or stabilizing online training. In this paper we propose handling multiple goals on multiple datasets by training with dataset constraints, using the ramp penalty to accurately quantify costs, and present an efficient algorithm to approximately optimize the resulting non-convex constrained optimization problem. Experiments on both benchmark and real-world industry datasets demonstrate the effectiveness of our approach.

1 Real-world goals

We consider a broad set of design goals important for making classifiers work well in real-world applications, and discuss how metrics quantifying many of these goals can be represented in a particular optimization framework. The key theme is that these metrics, which range from the standard precision and recall, to less well-known examples such as coverage and fairness [17, 27, 15], and including some new proposals, can be expressed in terms of the positive and negative classification rates on multiple datasets.

Coverage: One may wish to control how often a classifier predicts the positive (or negative) class. For example, one may want to ensure that only 10% of customers are selected to receive a printed catalog due to budget constraints, or perhaps to compensate for a biased training set. In practice, constraining the "coverage rate" (the expected proportion of positive predictions) is often easier than measuring e.g. accuracy or precision because coverage can be computed on unlabeled data—labeling data can be expensive, but acquiring a large number of unlabeled examples is often very easy.

Coverage was also considered by Mann and McCallum [17], who proposed what they call "label regularization", in which one adds a regularizer penalizing the relative entropy between the mean score for each class and the desired distribution, with an additional correction to avoid degeneracies.

Churn: Work does not stop once a machine learning model has been adopted. There will be new training data, improved features, and potentially new model structures. Hence, in practice, one will deploy a *series* of models, each improving slightly upon the last. In this setting, determining whether each candidate should be deployed is surprisingly challenging: if we evaluate on the *same* held-out testing set every time a new candidate is proposed, and deploy it if it outperforms its predecessor, then every compare-and-deploy decision will increase the statistical dependence between the deployed model and the testing dataset, causing the model sequence to fit the originally-independent testing data. This problem is magnified if, as is typical, the candidate models tend to disagree only on a relatively small number of examples near the true decision boundary.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

A simple and safe solution is to draw a *fresh* testing sample every time one wishes to compare two models in the sequence, only considering examples on which the two models disagree. Because labeling data is expensive, one would like these freshly sampled testing datasets to be as small as possible. It is here that the problem of "churn" arises. Imagine that model A, our deployed model, is 70% accurate, and that model B, our candidate, is 75% accurate. In the best case, only 5% of test samples would be labeled differently, and all differences would be "wins" for classifier B. Then only a dozen or so examples would need to be labeled in order to establish that B is the statistically significantly better classifier with 95% confidence. In the worst case, model A would be correct and model B incorrect 25% of the time, model B correct and model A incorrect 30% of the time, and both models correct the remaining 45% of the time. Then 55% of testing examples will be labeled differently, and closer to 1000 examples would need to be labeled to determine that model B is better.

We define the "churn rate" as the expected proportion of examples on which the prediction of the model being considered (model B above) differs from that of the currently-deployed model (model A). During training, we propose constraining the empirical churn rate with respect to a given deployed model on a large unlabeled dataset (see also Fard et al. [12] for an alternative approach).

Stability: A special case of minimizing churn is to ensure stability of an online classifier as it evolves, by constraining it to not deviate too far from a trusted classifier on a large held-out unlabeled dataset.

Fairness: A practitioner may be required to guarantee *fairness* of a learned classifier, in the sense that it makes positive predictions on different subgroups at certain rates. For example, one might require that housing loans be given equally to people of different genders. Hardt et al. [15] identify three types of fairness: (i) demographic parity, in which positive predictions are made at the same rate on each subgroup, (ii) equal opportunity, in which only the true positive rates must match, and (iii) equalized odds, in which both the true positive rates and false positive rates must match. Fairness can also be specified by a proportion, such as the 80% rule in US law that certain decisions must be in favor of group B individuals at least 80% as often as group A individuals [e.g. 3, 26, 27, 15].

Zafar et al. [27] propose learning fair classifiers by imposing linear constraints on the covariance between the predicted labels and the values of certain features, while Hardt et al. [15] propose first learning an "unfair" classifier, and then choosing population-dependent thresholds to satisfy the desired fairness criterion. In our framework, rate constraints such as those mentioned above can be imposed directly, at training time.

Recall and Precision: Requirements of real-world classifiers are often expressed in terms of precision and recall, especially when examples are highly imbalanced between positives and negatives. In our framework, we can handle this problem via Neyman-Pearson classification [e.g. 23, 9], in which one seeks to minimize the false negative rate subject to a constraint on the false positive rate. Indeed, our ramp-loss formulation is equivalent to that of Gasso et al. [13] in this setting.

Egregious Examples: For certain classification applications, examples may be discovered that are particularly embarrassing if classified incorrectly. One standard approach to handling such examples is to increase their weights during training, but this is difficult to get right: too large a weight may distort the classifier too much in the surrounding feature space, whereas too small a weight may not fix the problem. Worse, over time the dataset will often be augmented with new training examples and new features, causing the ideal weights to drift. We propose instead simply adding a constraint ensuring that some proportion of a set of such egregious examples is correctly classified. Such constraints should be used with extreme care, since they can cause the problem to become infeasible.

2 Optimization problem

A key aspect of many of the goals of Section 1 is that they are defined on different datasets. For example, we might seek to maximize the accuracy on a set of labeled examples drawn in some biased manner, require that its recall be at least 90% on 50 small datasets sampled in an unbiased manner from 50 different countries, desire low churn relative to a deployed classifier on a large unbiased unlabeled dataset, and require that 100 given egregious examples be classified correctly.

Another characteristic common to the metrics of Section 1 is that they can be expressed in terms of the positive and negative classification rates on various datasets. We consider only *unlabeled* datasets, as described in Table 1—a dataset with binary labels, for example, would be handled by partitioning it into the two unlabeled datasets D^+ and D^- containing the positive and negative examples,

Table 1: Dataset notation.

Notation	Dataset
D	Any dataset
D^+, D^-	Sets of examples labeled positive/negative, respectively
$D^{++}, D^{+-}, D^{-+}, D^{--}$	Sets of examples with ground-truth positive/negative labels, and for which a baseline classifier makes positive/negative predictions
D^A, D^B	Sets of examples belonging to subpopulation A and B, respectively

Table 2: The quantities discussed in Section 1, expressed in the notation used in Problem 1, with the dependence on w and b dropped for notational simplicity, and using the dataset notation of Table 1.

Metric	Expression
Coverage rate	$s_p\left(D\right)$
#TP, #TN, #FP, #FN	$\lvert D^+\rvert\, s_p\left(D^+\right)$, $\lvert D^-\rvert\, s_n\left(D^-\right)$, $\lvert D^-\rvert\, s_p\left(D^-\right)$, $\lvert D^+\rvert\, s_n\left(D^+\right)$
#Errors	$\#\text{FP} + \#\text{FN}$
Error rate	$\#\text{Errors}/\left(\lvert D^+\rvert + \lvert D^-\rvert\right)$
Recall	$\#\text{TP}/\left(\#\text{TP} + \#\text{FN}\right) = \#\text{TP}/\lvert D^+\rvert$
#Changes	$\lvert D^{+-}\rvert\, s_p\left(D^{+-}\right) + \lvert D^{-+}\rvert\, s_n\left(D^{-+}\right) + \lvert D^{+-}\rvert\, s_p\left(D^{+-}\right) + \lvert D^{-+}\rvert\, s_n\left(D^{-+}\right)$
Churn rate	$\#\text{Changes}/\left(\lvert D^{++}\rvert + \lvert D^{+-}\rvert + \lvert D^{-+}\rvert + \lvert D^{--}\rvert\right)$
Fairness constraint	$s_p\left(D^A\right) \geq \kappa s_p\left(D^B\right)$, where $\kappa > 0$
Equal opportunity constraint	$s_p\left(D^A \cap D^+\right) \geq \kappa s_p\left(D^B \cap D^+\right)$, where $\kappa > 0$
Egregious example constraint	$s_p\left(D^+\right) \geq \kappa$ and/or $s_n\left(D^-\right) \leq \kappa$ for a dataset D of egregious examples, where $\kappa \in [0, 1]$

respectively. We wish to learn a linear classification function $f(x) = \langle w, x\rangle - b$ parameterized by a weight vector $w \in \mathbb{R}^d$ and bias $b \in \mathbb{R}$, for which the positive and negative classification rates are:

$$s_p\left(D; w, b\right) = \tfrac{1}{|D|}\textstyle\sum_{x\in D}\mathbf{1}\left(\langle w, x\rangle - b\right), \qquad s_n\left(D; w, b\right) = s_p\left(D; -w, -b\right), \tag{1}$$

where $\mathbf{1}$ is an indicator function that is 1 if its argument is positive, 0 otherwise. In words, $s_p(D; w, b)$ and $s_n(D; w, b)$ denote the proportion of positive or negative predictions, respectively, that f makes on D. Table 2 specifies how the metrics of Section 1 can be expressed in terms of the s_ps and s_ns.

We propose handling these goals by minimizing an ℓ^2-regularized positive linear combination of prediction rates on different datasets, subject to upper-bound constraints on other positive linear combinations of such prediction rates:

Problem 1. *Starting point: discontinuous constrained problem*

$$\begin{aligned} \underset{w\in\mathbb{R}^d, b\in\mathbb{R}}{\text{minimize}} \quad & \textstyle\sum_{i=1}^{k}\left(\alpha_i^{(0)} s_p(D_i; w, b) + \beta_i^{(0)} s_n(D_i; w, b)\right) + \frac{\lambda}{2}\left\lVert w\right\rVert_2^2 \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{k}\left(\alpha_i^{(j)} s_p(D_i; w, b) + \beta_i^{(j)} s_n(D_i; w, b)\right) \leq \gamma^{(j)} \quad j \in \{1, \dots, m\}. \end{aligned}$$

Here, λ is the parameter on the ℓ^2 regularizer, there are k unlabeled datasets $D_1, \dots, D_k$ and m constraints. The metrics minimized by the objective and bounded by the constraints are specified via the choices of the nonnegative coefficients $\alpha_i^{(0)}$, $\beta_i^{(0)}$, $\alpha_i^{(j)}$, $\beta_i^{(j)}$ and upper bounds $\gamma^{(j)}$ for the ith dataset and, where applicable, the jth constraint—a user should base these choices on Table 2. Note that because $s_p + s_n = 1$, it is possible to transform *any* linear combination of rates into an equivalent positive linear combination, plus a constant (see Appendix B[1] for an example).

We cannot optimize Problem 1 directly because the rate functions s_p and s_n are discontinuous. We can, however, work around this difficulty by training a classifier that makes *randomized* predictions based on the ramp function [7]:

$$\sigma(z) = \max\{0, \min\{1, 1/2 + z\}\}, \tag{2}$$

[1] Appendices may be found in the supplementary material

Algorithm 1 Proposed majorization-minimization procedure for (approximately) optimizing Problem 2. Starting from an initial feasible solution $w^{(0)}$, b_0, we repeatedly find a convex upper bound problem that is tight at the current candidate solution, and optimize it to yield the next candidate. See Section 2.1 for details, and Section 2.2 for how one can perform the inner optimizations on line 3.

```
   MajorizationMinimization $\left(w^{(0)}, b_0, T\right)$
1    For $t \in \{1, 2, \dots, T\}$
2        Construct an instance of Problem 3 with $w' = w^{(t-1)}$ and $b' = b_{t-1}$
3        Optimize this convex optimization problem to yield $w^{(t)}$ and $b_t$
4    Return $w^{(t)}$, $b_t$
```

where the randomized classifier parameterized by w and b will make a positive prediction on x with probability $\sigma\left(\langle w, x\rangle - b\right)$, and a negative prediction otherwise (see Appendix A for more on this randomized classification rule). For this randomized classifier, the *expected* positive and negative rates will be:

$$r_p\left(D; w, b\right) = \tfrac{1}{|D|}\textstyle\sum_{x \in D} \sigma\left(\langle w, x\rangle - b\right), \qquad r_n\left(D; w, b\right) = r_p\left(D; -w, -b\right). \tag{3}$$

Using these expected rates yields a continuous (but non-convex) analogue of Problem 1:

Problem 2. *Ramp version of Problem 1*

$$\begin{aligned} \underset{w \in \mathbb{R}^d, b \in \mathbb{R}}{\text{minimize}} \quad & \textstyle\sum_{i=1}^{k} \left(\alpha_i^{(0)} r_p(D_i; w, b) + \beta_i^{(0)} r_n(D_i; w, b)\right) + \frac{\lambda}{2}\left\|w\right\|_2^2 \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{k} \left(\alpha_i^{(j)} r_p(D_i; w, b) + \beta_i^{(j)} r_n(D_i; w, b)\right) \le \gamma^{(j)} \quad j \in \{1, \dots, m\}. \end{aligned}$$

Efficient optimization of this problem is the ultimate goal of this section. In Section 2.1, we will propose a majorization-minimization approach that sequentially minimizes convex upper bounds on Problem 2, and, in Section 2.2, will discuss how these convex upper bounds may themselves be efficiently optimized.

2.1 Optimizing the ramp problem

To address the non-convexity of Problem 2, we will iteratively optimize approximations, by, starting from an feasible initial candidate solution, constructing a convex optimization problem upper-bounding Problem 2 that is *tight* at the current candidate, optimizing this convex problem to yield the next candidate, and repeating.

Our choice of a ramp for σ makes finding such tight convex upper bounds easy: both the hinge function $\max\{0, 1/2 + z\}$ and constant-1 function are upper bounds on σ, with the former being tight for all $z \le 1/2$, and the latter for all $z \ge 1/2$ (see Figure 1). We'll therefore define the following upper bounds on σ and $1 - \sigma$, with the additional parameter z' determining which of the two bounds (hinge or constant) will be used, such that the bounds will always be tight for $z = z'$:

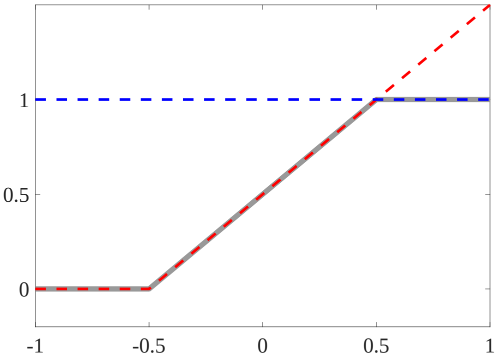

Figure 1: Convex upper bounds on the ramp function $\sigma(z) = \max\{0, \min\{1, 1/2 + z\}\}$. Notice that the hinge bound (red) is tight for all $z \le 1/2$, and the constant bound (blue) is tight for all $z \ge 1/2$.

$$\check{\sigma}_p\left(z; z'\right) = \begin{cases} \max\{0, 1/2 + z\} & \text{if } z' \le 1/2 \\ 1 & \text{otherwise} \end{cases}, \qquad \check{\sigma}_n(z; z') = \check{\sigma}_p\left(-z; -z'\right). \tag{4}$$

Based upon these we define the following upper bounds on the expected rates:

$$\begin{aligned} \check{r}_p\left(D; w, b; w', b'\right) &= \tfrac{1}{|D|}\textstyle\sum_{x \in D} \check{\sigma}_p\left(\langle w, x\rangle - b; \langle w', x\rangle - b'\right) \\ \check{r}_n\left(D; w, b; w', b'\right) &= \tfrac{1}{|D|}\textstyle\sum_{x \in D} \check{\sigma}_n\left(\langle w, x\rangle - b; \langle w', x\rangle - b'\right), \end{aligned} \tag{5}$$

which have the properties that both $\check{r}_p$ and $\check{r}_n$ are convex in w and b, are upper bounds on the original ramp-based rates:

$$\check{r}_p\left(D; w, b; w', b'\right) \ge r_p\left(D; w, b\right) \qquad \text{and} \qquad \check{r}_n\left(D; w, b; w', b'\right) \ge r_n\left(D; w, b\right),$$

Algorithm 2 Skeleton of a cutting-plane algorithm that optimizes Equation 6 to within ϵ for $v \in \mathcal{V}$, where $\mathcal{V} \subseteq \mathbb{R}^m$ is compact and convex. Here, $l_0, u_0 \in \mathbb{R}$ are finite with $l_0 \leq \max_{v \in \mathcal{V}} \mathcal{F}(v) \leq u_0$. There are several options for the CutChooser function on line 8—please see Appendix E for details. The SVMOptimizer function returns $w^{(t)}$ and b_t approximately minimizing $\Psi(w, b, v^{(t)}; w', b')$, and a lower bound $l_t \leq \mathcal{F}(v)$ for which $u_t - l_t \leq \epsilon_t$ for u_t as defined on line 10.

```
    CuttingPlane $(l_0, u_0, \mathcal{V}, \epsilon)$
1       Initialize $g^{(0)} \in \mathbb{R}^m$ to the all-zero vector
2       For $t \in \{1, 2, \dots\}$
3           Let $h_t(v) = \min_{s \in \{0,1,\dots,t-1\}} \left( u_s + \left\langle g^{(s)}, v - v^{(s)} \right\rangle \right)$
4           Let $L_t = \max_{s \in \{0,1,\dots,t-1\}} l_s$ and $U_t = \max_{v \in \mathcal{V}} h_t(v)$
5           If $U_t - L_t \leq \epsilon$ then
6               Let $s \in \{1, \dots, t-1\}$ be an index maximizing $l_s$
7               Return $w^{(s)}$, $b_s$, $v^{(s)}$
8           Let $v^{(t)}, \epsilon_t$ = CutChooser $(h_t, L_t)$
9           Let $w^{(t)}, b_t, l_t$ = SVMOptimizer $\left(v^{(t)}, h_t\left(v^{(t)}\right), \epsilon_t\right)$
10          Let $u_t = \Psi(w^{(t)}, b_t, v^{(t)}; w', b')$ and $g^{(t)} = \nabla_v \Psi(w^{(t)}, b_t, v^{(t)}; w', b')$
```

and are tight at w', b':

$$\check{r}_p(D; w', b'; w', b') = r_p(D; w', b') \qquad \text{and} \qquad \check{r}_n(D; w', b'; w', b') = r_n(D; w', b').$$

Substituting these bounds into Problem 2 yields:

Problem 3. *Convex upper bound on Problem 2*

$$\begin{aligned} \underset{w \in \mathbb{R}^d, b \in \mathbb{R}}{\text{minimize}} \quad & \textstyle\sum_{i=1}^{k} \left(\alpha_i^{(0)} \check{r}_p(D_i; w, b; w', b') + \beta_i^{(0)} \check{r}_n(D_i; w, b; w', b') \right) + \frac{\lambda}{2} \|w\|_2^2 \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{k} \left(\alpha_i^{(j)} \check{r}_p(D_i; w, b; w', b') + \beta_i^{(j)} \check{r}_n(D_i; w, b; w', b') \right) \leq \gamma^{(j)} \quad j \in \{1, \dots, m\}. \end{aligned}$$

As desired, this problem upper bounds Problem 2, is tight at w', b', and is convex (because any positive linear combination of convex functions is convex).

Algorithm 1 contains our proposed procedure for approximately solving Problem 2. Given an initial feasible solution, it's straightforward to verify inductively, using the fact that we construct tight convex upper bounds at every step, that every convex subproblem will have a feasible solution, every $(w^{(t)}, b_t)$ pair will be feasible w.r.t. Problem 2, and every $(w^{(t+1)}, b_{t+1})$ will have an objective function value that is no larger that that of $(w^{(t)}, b_t)$. In other words, no iteration can make negative progress. The non-convexity of Problem 2, however, will cause Algorithm 1 to arrive at a suboptimal solution that depends on the initial $(w^{(0)}, b_0)$.

2.2 Optimizing the convex subproblems

The first step in optimizing Problem 3 is to add Lagrange multipliers v over the constraints, yielding the equivalent unconstrained problem:

$$\underset{v \succeq 0}{\text{maximize}} \;\; \mathcal{F}(v) = \min_{w,b} \Psi(w, b, v; w', b'), \tag{6}$$

where the function:

$$\begin{aligned} \Psi(w, b, v; w', b') = & \textstyle\sum_{i=1}^{k} \left(\left(\alpha_i^{(0)} + \sum_{j=1}^{m} v_j \alpha_i^{(j)} \right) \check{r}_p(D_i; w, b; w', b') \right. \\ & \left. + \left(\beta_i^{(0)} + \sum_{j=1}^{m} v_j \beta_i^{(j)} \right) \check{r}_n(D_i; w, b; w', b') \right) + \frac{\lambda}{2} \|w\|_2^2 - \textstyle\sum_{j=1}^{m} v_j \gamma^{(j)} \end{aligned} \tag{7}$$

is convex in w and b, and concave in the multipliers v. For the purposes of this section, w' and b', which were found in the previous iteration of Algorithm 1, are fixed constants.

Because this is a convex-concave saddle point problem, there are a large number of optimization techniques that could be successfully applied. For example, in settings similar to our own, Eban et al. [10] simply perform SGD jointly over all parameters (including v), while Gasso et al. [13] use the Uzawa algorithm, which would alternate between (i) optimizing exactly over w and b, and (ii) taking gradient steps on v.

We instead propose an approach for which, in our setting, it is particularly easy to create an efficient implementation. The key insight is that evaluating $\digamma(v)$ is, thanks to our use of hinge and constant upper-bounds on our ramp σ, equivalent to optimization of a support vector machine (SVM) with per-example weights—see Appendix F for details. This observation enables us to solve the saddle system in an inside-out manner. On the "inside", we optimize over (w, b) for fixed v using an off-the-shelf SVM solver [e.g. 6]. On the "outside", the resulting (w, b)-optimizer is used as a component in a cutting-plane optimization over v. Notice that this outer optimization is very low-dimensional, since $v \in \mathbb{R}^m$, where m is the number of constraints.

Algorithm 2 contains a skeleton of the cutting-plane algorithm that we use for this outer optimization over v. Because this algorithm is intended to be used as an outer loop in a nested optimization routine, it does not expect that $\digamma(v)$ can be evaluated or differentiated exactly. Rather, it's based upon the idea of possibly making "shallow" cuts [4] by choosing a desired accuracy ϵ_t at each iteration, and expecting the SVMOptimizer to return a solution with suboptimality ϵ_t. More precisely, the SVMOptimizer function approximately evaluates $\digamma(v^{(t)})$ for a given fixed $v^{(t)}$ by constructing the corresponding SVM problem and finding a $(w^{(t)}, b_t)$ for which the primal and dual objective function values differ by at most ϵ_t.

After finding $(w^{(t)}, b_t)$, the SVMOptimizer then evaluates the dual objective function value of the SVM to determine l_t. The primal objective function value u_t and its gradient $g^{(t)}$ w.r.t. v (calculated on line 10 of Algorithm 2) define the cut $u_t + \left\langle g^{(t)}, v - v^{(t)} \right\rangle$. Notice that since $\Psi(w^{(t)}, b_t, v; w', b')$ is a linear function of v, it is equal to this cut function, which therefore upper-bounds $\min_{w,b} \Psi(w, b, v; w', b')$.

One advantage of this cutting-plane formulation is that typical CutChooser implementations will choose ϵ_t to be large in the early iterations, and will only shrink it to be ϵ or smaller once we're close to convergence. We leave the details of the analysis to Appendices E and F—a summary can be found in Appendix G.

3 Related work

The problem of finding optimal trade-offs in the presence of multiple objectives has been studied generically in the field of multi-objective optimization [18]. Two common approaches are (i) linear scalarization [18, Section 3.1], and (ii) the method of ϵ-constraints [18, Section 3.2]. Linear scalarization reduces to the common heuristic of reweighting groups of examples. The method of ϵ-constraints puts hard bounds on the magnitudes of secondary objectives, like our dataset constraints. Notice that, in our formulation, the Lagrange multipliers v play the role of the weights in the linear scalarization approach, with the difference being that, rather than being provided directly by the user, they are dynamically chosen to satisfy constraints. The user controls the problem through these constraint choices, which have concrete real-world meanings.

While the hinge loss is one of the most commonly-used convex upper bounds on the $0/1$ loss [22], we use the ramp loss, trading off convexity for tightness. For our purposes, the main disadvantage of the hinge loss is that it is unbounded, and therefore cannot distinguish a single very bad example from say, 10 slightly bad ones, making it ill-suited for constraints on rates. In contrast, for the ramp loss the contribution of any single datum is bounded, no matter how far it is from the decision boundary.

The ramp loss has also been investigated in Collobert et al. [7] (without constraints). Gasso et al. [13] use the ramp loss both in the objective and constraints, but their algorithm only tackles the Neyman-Pearson problem. They compared their classifier to that of Davenport et al. [9], which differs in that it uses a hinge relaxation instead of the ramp loss, and found with the ramp loss they achieved similar or slightly better results with up to $10\times$ less computation (our approach does not enjoy this computational speedup).

Narasimhan et al. [19] considered optimizing the F-measure and other quantities that can be written as concave functions of the TP and TN rates. Their proposed stochastic dual solver adaptively linearizes concave functions of the rate functions (Equation 1). Joachims [16] indirectly optimizes upper-bounds on functions of $s_p(D^+)$, $s_p(D^-)$, $s_n(D^+)$, $s_n(D^-)$ using a hinge loss approximation.

Finally, for some simple problems (particularly when there is only one constraint), the goals in Section 1 can be coarsely handled by simple bias-shifting, i.e. first training an unconstrained classifier, and then attempting to adjust the decision threshold to satisfy the constraints as a second step.

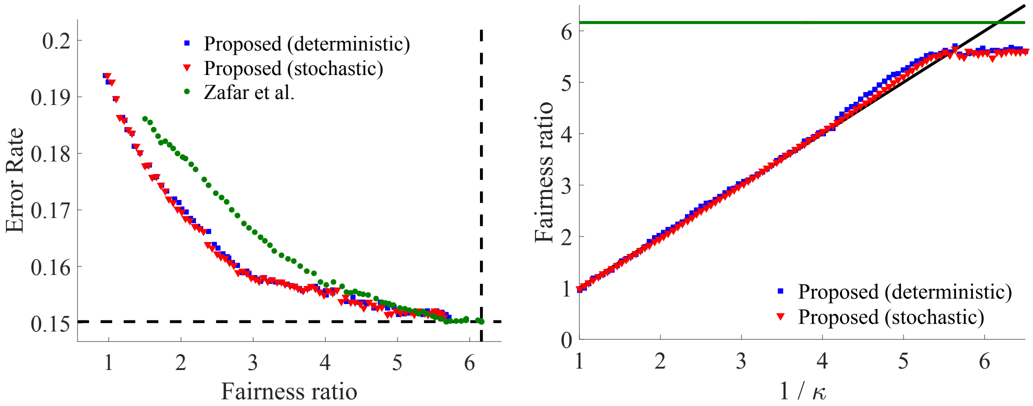

Figure 2: Blue dots: our proposal, with the classification functions' predictions being deterministically thresholded at zero. Red dots: same, but using the randomized classification rule described in Section 2. Green dots: Zafar et al. [27]. Green line: unconstrained SVM. **(Left)** Test set error plotted vs. observed test set fairness ratio $s_p\left(D^M\right)/s_p\left(D^F\right)$. **(Right)** The $1/\kappa$ hyper-parameter used to specify the desired fairness in the proposed method, and the observed fairness ratios of our classifiers on the test data. All points are averaged over 100 runs.

4 Experiments

We evaluate the performance of the proposed approach in two experiments, the first using a benchmark dataset for fairness, and the second on a real-world problem with churn and recall constraints.

4.1 Fairness

We compare training for fairness on the Adult dataset [2], the same dataset used by Zafar et al. [27]. The 32 561 training and 16 281 testing examples, derived from the 1994 Census, are 123-dimensional and sparse. Each feature contains categorical attributes such as race, gender, education levels and relationship status. A positive class label means that individual's income exceeds 50k. Let D^M and D^F denote the sets of male and female examples. The number of positive labels in D^M is roughly six times that of D^F. The goal is to train a classifier that respects the fairness constraint $s_p\left(D^M\right) \leq s_p\left(D^F\right)/\kappa$ for a parameter $\kappa \in (0,1]$ (where $\kappa = 0.8$ corresponds to the 80% rule mentioned in Section 1).

Our publicly-available `Julia` implementation[3] for these experiments uses `LIBLINEAR` [11] with the default parameters (most notably $\lambda = 1/n \approx 3 \times 10^{-5}$) to implement the SVMOptimizer function, and does not include an unregularized bias b. The outer optimization over v does not use the m-dimensional cutting plane algorithm of Algorithm 2, instead using a simpler one-dimensional variant (observe that these experiments involve only one constraint). The majorization-minimization procedure starts from the all-zeros vector ($w^{(0)}$ in Algorithm 1).

We compare to the method of Zafar et al. [27], which proposed handling fairness with the constraint:

$$\langle w, \bar{x} \rangle \leq c, \qquad \bar{x} = \left|D^M\right|^{-1} \textstyle\sum_{x \in D^M} x \;-\; \left|D^F\right|^{-1} \textstyle\sum_{x \in D^F} x. \tag{8}$$

An SVM subject to this constraint (see Appendix D for details), for a range of c values, is our baseline.

Results in Figure 2 show the proposed method is much more accurate for any desired fairness, and achieves fairness ratios not reachable with the approach of Zafar et al. [27] for any choice of c. It is also easier to control: the values of c in Zafar et al. [27] do not have a clear interpretation, whereas κ is an effective proxy for the fairness ratio.

4.2 Churn

Our second set of experiments demonstrates meeting real-world requirements on a proprietary problem from Google: predicting whether a user interface element should be shown to a user, based

[2] "a9a" from `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`

[3] `https://github.com/gabgoh/svmc.jl`

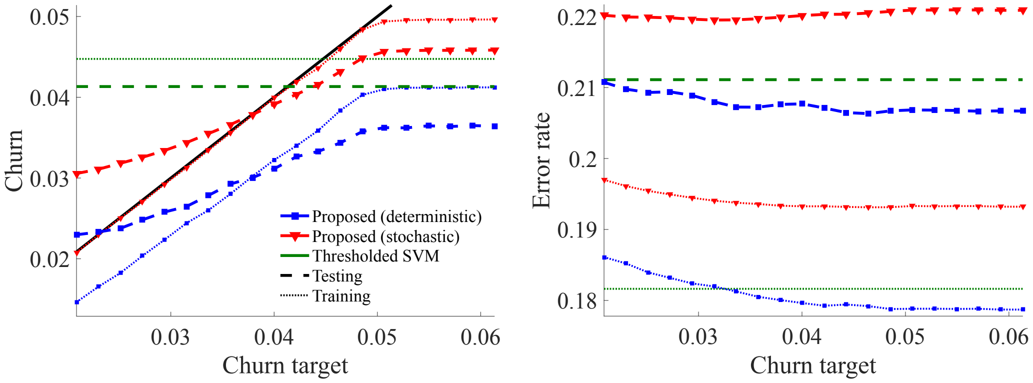

Figure 3: Blue: our proposal, with the classification functions' predictions being deterministically thresholded at zero. Red: same, but using the randomized classification rule described in Section 2. Green: unconstrained SVM trained on $D_1 \cup D_2$, then thresholded (by shifting the bias b) to satisfy the recall constraint on D_2. Dashed and dotted curves denote results on the testing and training datasets, respectively. **(Left)** Observed churn (vertical axis) vs. the churn target used during training (horizontal axis), on the unlabeled dataset D_3. **(Right)** Empirical error rates (vertical axis) vs. the churn target, on the union $D_1 \cup D_2$ of the two labeled datasets. All curves are averaged over 10 runs.

on a 31-dimensional vector of informative features, which is mapped to a roughly $30\,000$-dimensional feature vector via a fixed kernel function Φ. We train classifiers that are linear with respect to $\Phi(x)$. We are given the currently-deployed model, and seek to train a classifier that (i) has high accuracy, (ii) has no worse recall than the deployed model, and (iii) has low churn w.r.t. the deployed model.

We are given three datasets, D_1, D_2 and D_3, consisting of $131\,840$, $53\,877$ and $68\,892$ examples, respectively. The datasets D_1 and D_2 are hand-labeled, while D_3 is unlabeled. In addition, D_1 was chosen via active sampling, while D_2 and D_3 are sampled *i.i.d.* from the underlying data distribution. For all three datasets, we split out 80% for training and reserved 20% for testing. We address the three goals in the proposed framework by simultaneously training the classifier to minimize the number of errors on D_1 plus the number of false positives on D_2, subject to the constraints that the recall on D_2 be at least as high as the deployed model's recall (we're essentially performing Neyman-Pearson classification on D_2), and that the churn w.r.t. the deployed model on D_3 be no larger than a given target parameter.

These experiments use a proprietary `C++` implementation of Algorithm 2, using the combined SDCA and cutting plane approach of Appendix F to implement the inner optimizations over w and b, with the CutChooser helper functions being as described in Appendices E.1 and F.2.1. We performed 5 iterations of the majorization-minimization procedure of Algorithm 1.

Our baseline is an unconstrained SVM that is thresholded after training to achieve the desired recall, but makes no effort to minimize churn. We chose the regularization parameter λ using a power-of-10 grid search, found that 10^{-7} was best for this baseline, and then used $\lambda = 10^{-7}$ for all experiments.

The plots in Figure 3 show the achieved churn and error rates on the training and testing sets for a range of churn constraint values (red and blue curves), compared to the baseline thresholded SVM (green lines). When using deterministic thresholding of the learned classifier (the blue curves, which significantly outperformed randomized classification–the red curves), the proposed method achieves lower churn and better accuracy for all targeted churn rates, while also meeting the recall constraint.

As expected, the empirical churn is extremely close to the targeted churn on the training set when using randomized classification (red dotted curve, left plot), but less so on the 20% held-out test set (red dashed curve). We hypothesize this disparity is due to overfitting, as the classifier has $30\,000$ parameters, and D_3 is rather small (please see Appendix C for a discussion of the generalization performance of our approach). However, except for the lowest targeted churn, the actual classifier churn (blue dashed curves) is substantially lower than the targeted churn. Compared to the thresholded SVM baseline, our approach significantly reduces churn without paying an accuracy cost.

References

[1] K. Ball. An elementary introduction to modern convex geometry. *Flavors of Geometry*, 31: 1–58, 1997.

[2] P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *JMLR*, 3:463–482, 2002.

[3] D. Biddle. *Adverse Impact and Test Validation: A Practitioner's Guide to Valid and Defensible Employment Testing*. Gower, 2005.

[4] R. G. Bland, D. Goldfarb, and M. J. Todd. Feature article—the ellipsoid method: A survey. *Operations Research*, 29(6):1039–1091, November 1981.

[5] S. Boyd and L. Vandenberghe. Localization and cutting-plane methods, April 2011. Stanford EE 364b lecture notes.

[6] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[7] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *ICML*, 2006.

[8] A. Cotter, S. Shalev-Shwartz, and N. Srebro. Learning optimally sparse support vector machines. In *ICML*, pages 266–274, 2013.

[9] M. Davenport, R. G. Baraniuk, and C. D. Scott. Tuning support vector machines for minimax and Neyman-Pearson classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[10] E. E. Eban, M. Schain, A. Gordon, R. A. Saurous, and G. Elidan. Large-scale learning with global non-decomposable objectives, 2016. URL `https://arxiv.org/abs/1608.04802`.

[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[12] M. M. Fard, Q. Cormier, K. Canini, and M. Gupta. Launch and iterate: Reducing prediction churn. In *NIPS*, 2016.

[13] G. Gasso, A. Pappaionannou, M. Spivak, and L. Bottou. Batch and online learning algorithms for nonconvex Neyman-Pearson classification. *ACM Transactions on Intelligent Systems and Technology*, 2011.

[14] B. Grünbaum. Partitions of mass-distributions and convex bodies by hyperplanes. *Pacific Journal of Mathematics*, 10(4):1257–1261, December 1960.

[15] M. Hardt, E. Price, and N. Srebro. Equality of opportunity in supervised learning. In *NIPS*, 2016.

[16] T. Joachims. A support vector method for multivariate performance measures. In *ICML*, 2005.

[17] G. S. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning with expectation regularization. In *ICML*, 2007.

[18] K. Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

[19] H. Narasimhan, P. Kar, and P. Jain. Optimizing non-decomposable performance measures: a tale of two classes. In *ICML*, 2015.

[20] A. Nemirovski. Lecture notes: Efficient methods in convex programming. 1994. URL `http://www2.isye.gatech.edu/~nemirovs/Lect_EMCO.pdf`.

[21] L. Rademacher. Approximating the centroid is hard. In *SoCG*, pages 302–305, 2007.

[22] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2: 21–42, 2000.

[23] C. D. Scott and R. D. Nowak. A Neyman-Pearson approach to statistical learning. *IEEE Transactions on Information Theory*, 2005.

[24] S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss. *JMLR*, 14(1):567–599, Feb. 2013.

[25] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. *Mathematical Programming*, 127(1):3–30, March 2011.

[26] M. S. Vuolo and N. B. Levy. Disparate impact doctrine in fair housing. *New York Law Journal*, 2013.

[27] M. B. Zafar, I. Valera, M. G. Rodriguez, and K. P. Gummadi. Fairness constraints: A mechanism for fair classification. In *ICML Workshop on Fairness, Accountability, and Transparency in Machine Learning*, 2015.

Table 3: Key notation, listed in the order in which it was introduced.

Symbol	Introduced	Description
k	Section 2	Number of datasets
m	Section 2	Number of dataset constraints
D_i	Section 2	ith dataset
s_p, s_n	Section 2, Equation 1	Positive and negative indicator-based rates
λ	Section 2, Problem 1	Regularization parameter
$\alpha_i^{(0)}$, $\beta_i^{(0)}$	Section 2, Problem 1	Coefficients defining the objective function
$\alpha_i^{(j)}$, $\beta_i^{(j)}$	Section 2, Problem 1	Coefficients defining the jth dataset constraint
$\gamma^{(j)}$	Section 2, Problem 1	Given upper bound of the jth dataset constraint
σ	Section 2, Equation 2	Ramp function: $\sigma(z) = \max\{0, \min\{1, {}^1\!/\!{}_2 + z\}\}$
r_p, r_n	Section 2, Equation 3	Positive and negative ramp-based rates
$\check{\sigma}_p$, $\check{\sigma}_n$	Section 2.1, Equation 4	Convex upper bounds on ramp functions
$\check{r}_p$, $\check{r}_n$	Section 2.1, Equation 5	Convex upper bounds on ramp-based rates
Ψ	Section 2.2, Equation 7	SVM objective (for minimizing over w and b)
$\digamma$	Section 2.2, Equation 6	Optimum of Ψ (for maximizing over v)
v	Section 2.2	Lagrange multipliers associated with dataset constraints
$\mathcal{V}$	Section 2.2, Algorithm 2	Set of allowed vs
$v^{(s)}$	Section 2.2, Algorithm 2	Candidate solution at the tth iteration
l_t, u_t	Section 2.2, Algorithm 2	Lower and upper bounds on $\digamma(v^{(t)})$
$g^{(t)}$	Section 2.2, Algorithm 2	Gradient of the cutting plane inserted at the tth iteration
h_t	Section 2.2, Algorithm 2	Concave function upper-bounding $\digamma(v)$
L_t, U_t	Section 2.2, Algorithm 2	Lower and upper bounds on $\max_{v \in \mathcal{V}} \digamma(v)$
V	Appendix C	Maximum allowed v_j: $\mathcal{V} \subseteq [0, V]^m$
$\bar{s}_p$, $\bar{s}_n$	Appendix C, Equation 10	Expected positive and negative indicator-based rates
μ	Appendix E	Lebesgue measure
S_ℓ	Appendix E.2, Equation 11	Superlevel set
S_h	Appendix E.2, Equation 12	Superlevel hypograph
n	Appendix F.1	Total size of datasets: $n = \sum_{i=1}^{k} \lvert D_i \rvert$
$\check{\alpha}_i^{(0)}$, $\check{\beta}_i^{(0)}$	Appendix F.1, Equation 13	Coefficients defining the convex objective function
$\check{\alpha}_i^{(j)}$, $\check{\beta}_i^{(j)}$	Appendix F.1, Equation 14	Coefficients defining the jth convex dataset constraint
$\check{\gamma}^{(j)}$	Appendix F.1, Equation 15	Given upper bound of the jth convex dataset constraint
$\ell_{i,x}$	Appendix F.1, Equation 16	Loss of example x in dataset D_i, in the SVM objective
$\check{\alpha}_i$, $\check{\beta}_i$	Appendix F.1, Equation 17	Coefficients defining the SVM objective function
L	Appendix F.1, Equation 18	Lipschitz constant of the $\ell_{i,x}$s
ξ	Appendix F.1, Equation 19	SVM dual variables
Ψ^*	Appendix F.1, Equation 19	SVM dual objective (for maximizing over ξ)
b_s	Appendix F.2, Algorithm 3	Candidate solution at the tth iteration
l'_t, u'_t	Appendix F.2, Algorithm 3	Lower and upper bounds on $\min_{w \in \mathbb{R}^d} \Psi(w, b_t, v; w', b')$
g'_t	Appendix F.2, Algorithm 3	Derivative of the cutting plane inserted at the tth iteration
h'_t	Appendix F.2, Algorithm 3	Convex function lower-bounding $\min_{w \in \mathbb{R}^d} \Psi(w, b, v; w', b')$
L'_t, U'_t	Appendix F.2, Algorithm 3	Lower and upper bounds on $\min_{b \in \mathcal{B}, w \in \mathbb{R}^d} \Psi(w, b, v; w', b')$

A Randomized classification

The use of the ramp loss in Problem 2 can be interpreted in two ways, which are exactly equivalent at training time, but lead to the use of different classification rules at evaluation time.

Deterministic: This is the obvious interpretation: we would like to optimize Problem 1, but cannot do so because the indicator-based rates s_p and s_n are discontinuous, so we approximate them with the ramp-based rates r_n and r_p, and and hope that this approximation doesn't cost us too much, in terms of performance. The result is Problem 2. At evaluation time, on an example x, we make a positive prediction if $\langle w, x \rangle - b$ is nonnegative, and a negative prediction otherwise.

Randomized: In this interpretation (also used by Cotter et al. [8]), we reinterpret the ramp loss as the expected 0/1 loss suffered by a randomized classifier, with the result that the rates aren't being approximated *at all*—instead, we're using the indicator-based rates throughout, but randomizing the classifier and taking expectations to smooth out the discontinuities in the objective function. To be precise, at evaluation time, on an example x, we make a positive prediction with probability $\sigma(\langle w, x \rangle - b)$, and a negative prediction otherwise (with σ being the ramp function of Equation 2).

Table 4: Some ratio metrics (Appendix B), which are metrics that can be written as ratios of linear combinations of rates. #Wins and #Losses are actually linear combination metrics, but are needed for the other definitions (as are Recall and #Changes from Table 2).

Metric	**Expression**
Precision	$\#\text{TP}/\left(\#\text{TP}+\#\text{FP}\right)$
F_1-score	$2\text{Precision}\cdot\text{Recall}/\left(\text{Precision}+\text{Recall}\right)=2\#\text{TP}/\left(2\#\text{TP}+\#\text{FN}+\#\text{FP}\right)$
#Wins	$\lvert D^{+-}\rvert\, s_p\left(D^{+-}\right)+\lvert D^{-+}\rvert\, s_n\left(D^{-+}\right)$
#Losses	$\lvert D^{++}\rvert\, s_n\left(D^{+-}\right)+\lvert D^{--}\rvert\, s_p\left(D^{-+}\right)$
Win/loss Ratio	#Wins/#Losses
Win/change Ratio	#Wins/#Changes

Taking expectations of the indicator-based rates s_p and s_n over the randomness of this classification rule yields the ramp-based rates r_n and r_p, resulting, once again, in Problem 2.

This use of a randomized prediction isn't as unfamiliar as it may at first seem: in logistic regression, the classifier provides probability estimates at evaluation time (with σ being a sigmoid instead of a ramp). Furthermore, at training time, the learned classifier is assumed to be randomized, so that the optimization problem can be interpreted as maximizing the data log-likelihood.

In the setting of this paper, the main advantages of the use of a randomized classification rule are that (i) we can say something about generalization performance (Appendix C), and (ii) because the rates are never being approximated, the dataset constraints will be satisfied *tightly* on the training dataset, in expectation (this is easily seen in the dotted red curve in the left plot of Figure 3). Despite these apparent advantages, deterministic classifiers seem to work better in practice.

B Ratio metrics

Problem 1 minimizes an objective function and imposes upper-bound constraints, all of which are written as linear combinations of positive and negative rates—we refer to such as "linear combination metrics". Some metrics of interest, however, cannot be written in this form. One important subclass are the so-called "ratio metrics", which are *ratios* of linear combinations of rates. Examples of ratio metrics are precision, F_1-score, win/loss ratio and win/change ratio (recall is a linear combination metric, since its denominator is a constant).

Ratio metrics may not be used directly in the objective function, but can be included in constraints by multiplying through by the denominator, then shifting the constraint coefficients to be non-negative. For example, the constraint that precision must be greater than 90% can be expressed as follows:

$$\begin{aligned}
\left|D^{+}\right| s_p\left(D^{+}\right) \geq & 0.9\left(\left|D^{+}\right| s_p\left(D^{+}\right)+\left|D^{-}\right| s_p\left(D^{-}\right)\right)\\
0.1\left|D^{+}\right| s_p\left(D^{+}\right)-0.9\left|D^{-}\right| s_p\left(D^{-}\right) \geq & 0\\
-0.1\left|D^{+}\right| s_p\left(D^{+}\right)+0.9\left|D^{-}\right| s_p\left(D^{-}\right) \leq & 0\\
0.1\left|D^{+}\right| s_n\left(D^{+}\right)+0.9\left|D^{-}\right| s_p\left(D^{-}\right) \leq & 0.1\left|D^{+}\right|,
\end{aligned}$$

where we used the fact that $s_p\left(D^{+}\right)+s_n\left(D^{+}\right)=1$ on the last line—this is an example of a fact that we noted in Section 2: since positive and negative rates must sum to one, it is possible to write any linear combination of rates as a positive linear combination, plus a constant.

Multiplying through by the denominator is fine for Problem 1, but a natural question is whether, by using a randomized classifier and optimizing Problem 2, we're doing the "right thing" in expectation. The answer is: not quite. Since the expectation of a ratio is not the ratio of expectations, e.g. a precision constraint in our original problem (Problem 1) becomes only a constraint on a precision-like quantity (the ratio of the expectations of the precision's numerator and denominator) in our relaxed problem.

C Generalization

In this appendix, we'll provide generalization bounds for an algorithm that is *nearly* identical to Algorithm 1. The two differences are that (i) we assume that the optimizer used on line 3 will prefer smaller biases b to larger ones, i.e. that if Problem 3 has multiple equivalent minima, then the optimizer will return one for which $|b|$ is minimized, and (ii) that the Lagrange multipliers are upper-bounded by a parameter $V \geq v_j$, i.e. that instead of optimizing Equation 6, line 3 of Algorithm 1 will optimize:

$$\max_{0 \preceq v \preceq V} \min_{w,b} \Psi\left(w, b, v; w', b'\right), \tag{9}$$

the difference being the upper bound on v. If V is large enough that no v_js are bound to a constraint, then this will have no effect on the solution. If, however, V is too small, then the solution might not satisfy the dataset constraints. Notice that Algorithm 2 assumes that $v \in \mathcal{V}$, with $\mathcal{V}$ being compact—hence, for our proposed optimization procedure, the assumption is that $\mathcal{V} \subseteq [0, V]^m$.

With these assumptions in place, we're ready to move on to defining a function class that contains any solution that could be found by our algorithm, and bounding its Rademacher complexity.

Lemma 1. *Define $\mathcal{F}$ to be the set of all linear functions $f(x) = \langle w, x\rangle - b$ with $\|w\|_2 \leq XB/\lambda$ and $|b| \leq 1/2 + X^2B/\lambda$, where $X \geq \|x\|_2$ is a uniform upper bound on the magnitudes of all training examples, and:*

$$B = \sum_{i=1}^{k}\left(\alpha_i^{(0)} + \beta_i^{(0)} + V\sum_{j=1}^{m}\left(\alpha_i^{(j)} + \beta_i^{(j)}\right)\right).$$

Then $\mathcal{F}$ will contain all $|b|$-minimizing optimal solutions of Equation 9 for any (w', b') and any training dataset.

Proof. Let $f(w,b) + (\lambda/2)\|w\|_2^2$ be the the objective function of Problem 3, and $g_j(w,b) \leq \gamma^{(j)}$ the jth constraint. Then it follows that:

$$\begin{aligned}\left\|\nabla_w f\left(w,b\right)\right\|_2 \leq & X\sum_{i=1}^{k}\left(\alpha_i^{(0)} + \beta_i^{(0)}\right)\\ \left\|\nabla_w g_j\left(w,b\right)\right\|_2 \leq & XV\sum_{i=1}^{k}\left(\alpha_i^{(j)} + \beta_i^{(j)}\right).\end{aligned}$$

Differentiating the definition of Ψ (Equation 7) and setting the result equal to zero shows that any optimal w must satisfy (this is the stationarity KKT condition):

$$\lambda w = -\nabla_w f\left(w,b\right) - \sum_{j=1}^{m} v_j \nabla_w g_j\left(w,b\right),$$

implying by the triangle inequality that $\|w\|_2 \leq XB/\lambda$, where B is as defined in the theorem statement.

Now let's turn our attention to b. The above bound implies that, if w is optimal, then $|\langle w,x\rangle| \leq X^2B/\lambda$, from which it follows that the hinge functions $\max\{0, 1/2 + (\langle w,x\rangle - b)\}$ and $\max\{0, 1/2 - (\langle w,x\rangle - b)\}$ will be nondecreasing in $|b|$ as long as $|b| > 1/2 + X^2B/\lambda$. Problem 3 seeks to minimize a positive linear combination of such hinge functions subject to upper-bound constraints on positive linear combinations of such hinge functions, so our assumption that the optimizer used on line 3 of Algorithm 1 will always choose the smallest optimal b gives that $|b| \leq 1/2 + X^2B/\lambda$. □

Lemma 2. *The function class $\mathcal{F}$ of Lemma 1 has Rademacher complexity [2]:*

$$\mathcal{R}_n\left(\mathcal{F}\right) \leq \frac{1}{2\sqrt{n}} + \frac{2X^2}{\lambda\sqrt{n}}\sum_{i=1}^{k}\left(\alpha_i^{(0)} + \beta_i^{(0)} + V\sum_{j=1}^{m}\left(\alpha_i^{(j)} + \beta_i^{(j)}\right)\right),$$

where $X \geq \|x\|_2$, as in Lemma 1, is a uniform upper bound on the magnitudes of all training examples.

Proof. The Rademacher complexity of $\mathcal{F}$ is:

$$\begin{aligned}\mathcal{R}_n\left(\mathcal{F}\right) =& \mathbb{E}\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}\epsilon_i f\left(x_i\right)\right]\\ =& \mathbb{E}\left[\sup_{w:\|w\|_2 \leq \frac{XB}{\lambda}} \frac{1}{n}\sum_{i=1}^{n}\epsilon_i\left\langle w,x\right\rangle\right] + \mathbb{E}\left[\sup_{b:|b|\leq\frac{1}{2}+\frac{X^2B}{\lambda}} \frac{1}{n}\sum_{i=1}^{n}\epsilon_i b\right],\end{aligned}$$

where the expectations are taken over the *i.i.d.* Rademacher random variables $\epsilon_1, \dots, \epsilon_n$ and the *i.i.d.* training sample $x_1, \dots, x_n$, and B is as in Lemma 1. Applying the Khintchine inequality and substituting the definition of B yields the claimed bound. □

We can now apply the results of Bartlett and Mendelson [2] to prove bounds on the generalization error. To this end, we assume that each of our training datasets D_i is drawn *i.i.d.* from some underlying unknown distribution $\mathcal{D}_i$. We will bound the expected positive and negative prediction rates w.r.t. these distributions:

$$\bar{s}_p\left(\mathcal{D}; f\right) = \mathbb{E}_{x \sim \mathcal{D}}\left[f\left(x\right)\right] \qquad \bar{s}_n\left(\mathcal{D}; f\right) = \bar{s}_p\left(\mathcal{D}; 1 - f\right), \tag{10}$$

where $f : \mathbb{R}^d \rightarrow \{0, 1\}$ is a binary classification function.

Theorem 1. *For a given (w, b) pair, define $f_{w,b}(x)$ such that it predicts 1 with probability $\sigma(\langle w, x\rangle - b)$, and 0 otherwise (σ is as in Equation 2, so this is the randomized classifier of Appendix A).*

Suppose that the k training datasets D_i have sizes $n_i = |D_i|$, and that D_i is drawn i.i.d. *from $\mathcal{D}_i$ for all $i \in \{1, \dots, k\}$. Then, with probability $1 - \delta$ over the training samples, uniformly over all (w, b) pairs that are optimal solutions of Equation 9 for some (w', b') under the assumptions listed at the start of this appendix, the expected rates will satisfy:*

$$\begin{aligned} \bar{s}_p\left(\mathcal{D}_i; f_{w,b}\right) \leq & r_p\left(D_i; w, b\right) + E/\sqrt{n_i} \\ \bar{s}_n\left(\mathcal{D}_i; f_{w,b}\right) \leq & r_n\left(D_i; w, b\right) + E/\sqrt{n_i}, \end{aligned}$$

the above holding for all $i \in \{1, \dots, k\}$, where:

$$E = 1 + \frac{4X^2}{\lambda} \sum_{i=1}^{k} \left(\alpha_i^{(0)} + \beta_i^{(0)} + V \sum_{j=1}^{m} \left(\alpha_i^{(j)} + \beta_i^{(j)} \right) \right) + \sqrt{8 \ln\left(\frac{4k}{\delta}\right)},$$

with $X \geq \left\|x\right\|_2$, as in Lemmas 1 and 2, being a uniform upper bound on the magnitudes of all training examples $x \sim \mathcal{D}_i$ for all $i \in \{1, \dots, k\}$.

Proof. Observe that the ramp rates r_p and r_n are 1-Lipschitz. Applying Theorems 8 and 12 (part 4) of Bartlett and Mendelson [2] gives that each of the following inequalities hold with probability $1 - \delta/2k$, for all $i \in \{1, \dots, k\}$:

$$\begin{aligned} \mathbb{E}_{x \sim \mathcal{D}_i}\left[r_p\left(\{x\}; w, b\right)\right] \leq & r_p\left(D_i; w, b\right) + 2\mathcal{R}_{n_i}\left(\mathcal{F}\right) + \sqrt{\frac{8}{n_i} \ln\left(\frac{4k}{\delta}\right)} \\ \mathbb{E}_{x \sim \mathcal{D}_i}\left[r_n\left(\{x\}; w, b\right)\right] \leq & r_n\left(D_i; w, b\right) + 2\mathcal{R}_{n_i}\left(\mathcal{F}\right) + \sqrt{\frac{8}{n_i} \ln\left(\frac{4k}{\delta}\right)}, \end{aligned}$$

where $\mathcal{R}_n(\mathcal{F})$ is as in Lemma 2. The union bound implies that all $2k$ inequalities hold simultaneously with probability $1 - \delta$. The LHSs above are the expected ramp-based rates of a deterministic classifier, but as was explained in Appendix A, these are identical to the expected indicator-based rates of a randomized classifier, which is what is claimed. □

An immediate consequence of this result is that (with probability $1 - \delta$) if (w, b) suffers the training loss:

$$\hat{\mathcal{L}} = \sum_{i=1}^{k} \left(\alpha_i^{(0)} r_p(D_i; w, b) + \beta_i^{(0)} r_n(D_i; w, b) \right),$$

then the expected loss on previously-unseen data (drawn *i.i.d.* from the same distributions) will be upper-bounded by:

$$\hat{\mathcal{L}} + E \sum_{i=1}^{k} \frac{\alpha_i^{(0)} + \beta_i^{(0)}}{\sqrt{n_i}}.$$

Likewise, if (w, b) satisfies the constraint:

$$\sum_{i=1}^{k} \left(\alpha_i^{(j)} r_p(D_i; w, b) + \beta_i^{(j)} r_n(D_i; w, b) \right) \leq \gamma^{(j)},$$

then the corresponding rate constraint on previously-unseen data will be violated by no more than:

$$E \sum_{i=1}^{k} \frac{\alpha_i^{(j)} + \beta_i^{(j)}}{\sqrt{n_i}}$$

in expectation, where, here and above, E is as in Theorem 1.

D Fairness constraints of Zafar et al. [27]

The constraints of Zafar et al. [27] can be interpreted as a relaxation of the constraint $-c \leq s_p(D^A; w) - s_p(D^B; w) \leq c$ under the linear approximation

$$s_p(D; w, b) \approx \frac{1}{|D|} \sum_{x \in D} (\langle w, x \rangle - b),$$

giving:

$$s_p(D^A; w, b) - s_p(D^B; w, b) \approx \frac{1}{|D^A|} \sum_{x \in D^A} (\langle w, x \rangle - b) - \frac{1}{|D^B|} \sum_{x \in D^B} (\langle w, x \rangle - b) = \langle w, \bar{x} \rangle \, ,$$

where $\bar{x}$ is defined as in Equation 8. We can therefore implement the approach of Zafar et al. [27] within our framework by adding the constraints:

$$\begin{aligned} \langle w, \bar{x} \rangle \leq c &\iff \max\{0, 1 - \langle w, \bar{x} \rangle\} \leq c + 1 \\ c \leq \langle w, \bar{x} \rangle &\iff \max\{0, 1 + \langle w, \bar{x} \rangle\} \leq c + 1, \end{aligned}$$

and solving the hinge constrained optimization problem described in Problem 3. Going further, we could implement these constraints as egregious examples using the constraint:

$$\begin{aligned} \langle w, \bar{x} \rangle \leq c \iff \left\langle w, \frac{1}{4c} \bar{x} \right\rangle \leq \frac{1}{4} \iff \frac{1}{2} + \left\langle w, \frac{1}{4c} \bar{x} \right\rangle \leq \frac{3}{4} \\ \iff \min\left\{ \max\left\{ \frac{1}{2} + \left\langle w, \frac{1}{4c} \bar{x} \right\rangle, 0 \right\}, 1 \right\} \leq \frac{3}{4} \iff r_p(\bar{x}) \leq \frac{3}{4}, \end{aligned}$$

permitting us to perform an analogue of their approximations in ramp form.

E Cutting plane algorithm

We'll now discuss some variants of Algorithm 2. We assume that $\mathcal{F}(v)$ is the function that we wish to maximize for $v \in \mathcal{V}$, where:

1. $\mathcal{V} \subseteq \mathbb{R}^m$ is compact and convex.
2. $\mathcal{F} : \mathcal{V} \to \mathbb{R}$ is concave.
3. $\mathcal{F}$ has a (not necessarily unique) maximizer $v^* = \operatorname{argmax}_{v \in \mathcal{V}} \mathcal{F}(v)$.

For the purposes of Algorithm 2, we would take $\mathcal{F}$ to be as in Equation 6, but the same approach can be applied more generally.

E.1 Maximization-based

We're primarily interested in proving convergence rates, and will do so in Appendix E.2. With that said, there is one easy-to-implement variant of Algorithm 2 for which we have not proved a convergence rate, but that we use in some of our experiments due to its simplicity:

Definition 1. ***(Maximization-based Algorithm 2)*** *CutChooser chooses* $v^{(t)} = \operatorname{argmax}_{v \in \mathcal{V}} h_t(v)$ *and* $\epsilon_t = (U_t - L_t)/2$.

Observe that this $v^{(t)}$ can be found at the same time as U_t is computed, since both result from optimization of the same linear program. However, despite the ease of implementing this variant, we have not proved any convergence rates about it.

E.2 Center of mass-based

We'll now discuss a variant of Algorithm 2 that chooses $v^{(t)}$ and ϵ_t based on the center of mass of the "superlevel hypograph" determined by h_t and L_t, which we define as the intersection of the hypograph of h_t (the set of $m+1$-dimensional points (v, z) for which $z \leq h_t(z)$) and the half-space containing all points (v, z) for which $z \geq L_t$. Notice that, in the context of Algorithm 2, the superlevel hypograph defined by h_t and L_t corresponds to the set of pairs of candidate maximizers and their possible function values at the tth iteration. Because this variant is based on finding a cut center in the $m+1$-dimensional hypograph, rather than an m-dimensional level set (which is arguably more typical), this is an instance of what Boyd and Vandenberghe [5] call an "epigraph cutting plane method".

Throughout this section, we will take μ to be the Lebesgue measure (either 1-dimensional, m-dimensional, or $m+1$-dimensional, depending on context). We also must define some notation for dealing with superlevel sets and hypographs. For a concave $f : \mathcal{V} \to \mathbb{R}$ and $y \in \mathbb{R}$, define:

$$S_\ell\left(f, y\right) = \left\{v \in \mathcal{V} \mid f\left(v\right) \geq y\right\} \tag{11}$$

as the superlevel set of f at y. Further define:

$$S_h\left(f, y\right) = \left\{(v, z) \in \mathcal{V} \times \mathbb{R} \mid f\left(v\right) \geq z \geq y\right\} \tag{12}$$

as the superlevel hypograph of f above y. With these definitions in place, we're ready to explicitly state the center of mass-based rule for the CutChooser function on line 8 of Algorithm 2:

Definition 2. ***(Center of mass-based Algorithm 2)*** *CutChooser takes $(v^{(t)}, z_t)$ to be the center of mass of $S_h(h_t, L_t)$, and chooses $\epsilon_t = (z_t - L_t)/2$.*

Finding the center of mass of a polytope is a difficult problem in general [20, 21], so our convergence results for this version of CutChooser are mostly of theoretical interest. With that said, for one dimensional problems (the setting of Appendix F.2) it may be implemented efficiently.

Our final bit of "setup" before getting to our results is to state two classic theorems, plus a corollary, which will be needed for our proofs. The first enables us to interpolate the areas of superlevel sets:

Theorem 2. *Suppose that the superlevel sets of a concave $f : \mathcal{V} \to \mathbb{R}$ at y_1 and y_2 are nonempty, and take $\gamma \in [0, 1]$. Then:*

$$\left(\mu\left(S_\ell\left(f, \gamma y_1 + \left(1-\gamma\right) y_2\right)\right)\right)^{1/m} \geq \gamma \left(\mu\left(S_\ell\left(f, y_1\right)\right)\right)^{1/m} + \left(1-\gamma\right)\left(\mu\left(S_\ell\left(f, y_2\right)\right)\right)^{1/m}.$$

Proof. This is the Brunn-Minkowski inequality [e.g. 1]. □

This theorem has the immediate useful corollary:

Corollary 1. *Suppose that $f : \mathcal{V} \to \mathbb{R}$ is concave with a maximizer $v^* \in \mathcal{V}$, and that $\delta \geq 0$. Then:*

$$\left(\frac{\delta}{m+1}\right) \mu\left(S_\ell\left(f, f\left(v^*\right) - \delta\right)\right) \leq \mu\left(S_h\left(f, f\left(v^*\right) + \delta\right)\right) \leq \delta \mu\left(S_\ell\left(f, f\left(v^*\right) - \delta\right)\right).$$

Proof. By Theorem 2 (lower-bounding the second term on the RHS by zero), for $0 \leq z \leq \delta$:

$$\mu\left(S_\ell\left(f, f\left(v^*\right) - z\right)\right) \geq \left(\frac{z}{\delta}\right)^m \mu\left(S_\ell\left(f, f\left(v^*\right) - \delta\right)\right),$$

from which integrating $\mu\left(S_h\left(f, f\left(v^*\right) - \delta\right)\right) = \int_0^\delta \mu\left(S_\ell\left(f, f\left(v^*\right) - z\right)\right) m\mu(z)$ yields the claimed lower bound. The upper bound follows immediately from the fact that the superlevel sets shrink as z increases (i.e. $\mu\left(S_\ell\left(f, z'\right)\right) \leq \mu\left(S_\ell\left(f, z\right)\right)$ for $z' \geq z$). □

The second classic result enables us to bound how much "progress" is made by a cut based on the center of mass of a superlevel hypograph:

Theorem 3. *Suppose that $S \subseteq \mathbb{R}^m$ is a convex set. If we let $z \in S$ be the center of mass of S, then for any half-space $H \ni z$:*

$$\frac{\mu\left(S \cap H\right)}{\mu\left(S\right)} \geq \left(\frac{m}{m+1}\right)^m \geq \frac{1}{e}.$$

Proof. This is Theorem 2 of Grünbaum [14]. □

With the preliminaries out of the way, we're ready to move on to our first result: bounding the volumes of the superlevel hypographs of our h_ts, assuming that we base our cuts on the centers of mass of the superlevel hypographs:

Lemma 3. *In the context of Algorithm 2, suppose that we choose $v^{(t)}$ and ϵ_t as in Definition 2. Then:*

$$\mu\left(S_h\left(h_{t+1}, L_{t+1}\right)\right) \leq \left(1 - \frac{1}{2e}\right) \mu\left(S_h\left(h_t, L_t\right)\right),$$

from which it follows that:

$$\mu\left(S_h\left(h_t, L_t\right)\right) \leq \left(1 - \frac{1}{2e}\right)^{t-1} \left(u_0 - l_0\right) \mu\left(\mathcal{V}\right),$$

for all t.

Proof. We'll consider two cases: $u_t \leq z_t$ and $u_t > z_t$, corresponding to making a "deep" or "shallow" cut, respectively.

Deep cut case: If $u_t \leq z_t$, then the hyperplane $u_t + \left\langle g^{(t)}, v - v^{(t)} \right\rangle$ passes below the center of mass of $S_h(h_t, L_t)$, implying by Theorem 3 that:

$$\mu\left(S_h\left(h_{t+1}, L_{t+1}\right)\right) \leq \mu\left(S_h\left(h_{t+1}, L_t\right)\right) \leq \left(1 - \frac{1}{e}\right) \mu\left(S_h\left(h_t, L_t\right)\right).$$

Shallow cut case: Now suppose that $u_t > z_t$. Applying Theorem 3 to the level cut $\{(v, z) \mid z \leq z_t\}$ at z_t:

$$\begin{aligned} \frac{1}{e} \mu\left(S_h\left(h_t, L_t\right)\right) \leq & \int_{L_t}^{z_t} \mu\left(\left\{v \in \mathcal{V} \mid h_t\left(v\right) \geq z\right\}\right) d\mu(z) \\ \leq & \int_{L_t}^{(z_t + L_t)/2} \mu\left(\left\{v \in \mathcal{V} \mid h_t\left(v\right) \geq z\right\}\right) d\mu(z) \\ & + \int_{(z_t + L_t)/2}^{z_t} \mu\left(\left\{v \in \mathcal{V} \mid h_t\left(v\right) \geq z\right\}\right) d\mu(z). \end{aligned}$$

Since h_t is concave, its superlevel sets shrink for larger z, so the first integral on the RHS above is larger than the second, implying that:

$$\frac{1}{2e} \mu\left(S_h\left(h_t, L_t\right)\right) \leq \int_{L_t}^{(z_t + L_t)/2} \mu\left(\left\{v \in \mathcal{V} \mid h_t\left(v\right) \geq z\right\}\right) d\mu(z).$$

The fact that $\epsilon_t = (z_t - L_t)/2$ implies that $l_t > (z_t + L_t)/2$, so $L_{t+1} > (z_t + L_t)/2$, and:

$$\frac{1}{2e} \mu\left(S_h\left(h_t, L_t\right)\right) \leq \int_{L_t}^{L_{t+1}} \mu\left(\left\{v \in \mathcal{V} \mid h_t\left(v\right) \geq z\right\}\right) d\mu(z),$$

showing that we will cut off at least a $1/2e$-proportion of the total volume, completing the proof of the first claim.

The second claim follows immediately by iterating the first, and observing that $\mu\left(S_h\left(h_1, L_1\right)\right) = \left(u_0 - l_0\right) \mu\left(\mathcal{V}\right)$. □

The above result shows that the volumes of the superlevel hypographs of the h_ts shrink at an exponential rate. However, our actual stopping condition (line 5 of Algorithm 2) depends not on the volume, but rather the "height" $U_t - L_t$, so we would prefer a bound on this height, rather than the volume. We find such a bound in the (proof of the) following lemma, which establishes how many iterations must elapse before the stopping condition is satisfied:

Lemma 4. *In the context of Algorithm 2, suppose that we choose $v^{(t)}$ and ϵ_t as in Definition 2. Then there is a iteration count T_ϵ satisfying:*

$$T_\epsilon = O\left(m \ln\left(\frac{u_0 - l_0}{\epsilon}\right) + \ln\left(\frac{\mu\left(\mathcal{V}\right)}{\mu\left(S_\ell\left(F, l_0\right)\right)}\right)\right),$$

such that, if $t \geq T_\epsilon$, then $U_t - L_t \leq \epsilon$. Hence, Algorithm 2 will terminate after T_ϵ iterations.

Proof. By Corollary 1:

$$\mu\left(S_h\left(h_t, L_t\right)\right) \geq \left(\frac{U_t - L_t}{m+1}\right) \mu\left(S_\ell\left(h_t, L_t\right)\right).$$

If $L_t \leq \digamma\left(v^*\right) - \epsilon$, then $\mu\left(S_\ell\left(h_t, L_t\right)\right) \geq \mu\left(S_\ell\left(h_t, \digamma\left(v^*\right) - \epsilon\right)\right)$ because h_t is concave. If $L_t > \digamma\left(v^*\right) - \epsilon$, then by Theorem 2:

$$\mu\left(S_\ell\left(h_t, L_t\right)\right) \geq \left(\frac{U_t - L_t}{U_t - \digamma\left(v^*\right) + \epsilon}\right)^m \mu\left(S_\ell\left(h_t, \digamma\left(v^*\right) - \epsilon\right)\right).$$

In either case, $L_t \leq \digamma\left(v^*\right)$ by definition, and we'll assume that $U_t - L_t > \epsilon$ (this will lead to a contradiction), so:

$$\mu\left(S_h\left(h_t, L_t\right)\right) \geq 2^{-m} \left(\frac{U_t - L_t}{m+1}\right) \mu\left(S_\ell\left(h_t, \digamma\left(v^*\right) - \epsilon\right)\right).$$

Applying Lemma 3 yields that:

$$\left(1 - \frac{1}{2e}\right)^{t-1} \left(u_0 - l_0\right) \mu\left(\mathcal{V}\right) \geq 2^{-m} \left(\frac{U_t - L_t}{m+1}\right) \mu\left(S_\ell\left(h_t, \digamma\left(v^*\right) - \epsilon\right)\right).$$

Next observe that, by Theorem 2:

$$\mu\left(S_\ell\left(h_t, \digamma\left(v^*\right) - \epsilon\right)\right) \geq \left(\frac{U_t - \digamma\left(v^*\right) + \epsilon}{U_t - l_0}\right)^m \mu\left(S_\ell\left(h_t, l_0\right)\right) \geq \left(\frac{\epsilon}{u_0 - l_0}\right)^m \mu\left(S_\ell\left(\digamma, l_0\right)\right).$$

Combining the previous two equations gives:

$$U_t - L_t \leq \left(1 - \frac{1}{2e}\right)^{t-1} (m+1) \left(\frac{2}{\epsilon}\right)^m \left(u_0 - l_0\right)^{m+1} \left(\frac{\mu\left(\mathcal{V}\right)}{\mu\left(S_\ell\left(\digamma, l_0\right)\right)}\right).$$

Simplifying this inequality yields that, if we have performed the claimed number of iterations, then $U_t - L_t \leq \epsilon$ (this contradicts our earlier assumption that $U_t - L_t > \epsilon$, so this is technically a proof by contradiction). □

The second term in the bound on T_ϵ measures how closely $\mathcal{V}$ matches with the set of all points z on which $\digamma\left(z\right)$ exceeds our initial lower bound l_0. Observe that if $l_0 \leq \digamma\left(v\right)$ for all $v \in \mathcal{V}$, then $\mu\left(S_\ell\left(\digamma, l_0\right)\right) = \mu\left(\mathcal{V}\right)$, and this term will vanish.

Bounding the number of cutting-plane iterations that will be performed is not enough to establish how quickly our procedure will converge, since we rely on performing an inner SVM optimizations with target suboptimality ϵ_t, and the runtime of these inner optimizations naturally will depend on the magnitudes of the ϵ_ts, which are bounded in our final lemma:

Lemma 5. *In the context of Algorithm 2, suppose that we choose $v^{(t)}$ and ϵ_t as in Definition 2. Then:*

$$\epsilon_t \geq \frac{U_t - L_t}{2e\left(m+1\right)},$$

and in particular, for all t (before termination):

$$\epsilon_t \geq \frac{\epsilon}{2e\left(m+1\right)},$$

since we terminate as soon as $U_t - L_t \leq \epsilon$.

Proof. Because h_t is concave:

$$\mu\left(S_h\left(h_t, L_t\right)\right) - \mu\left(S_h\left(h_t, z_t\right)\right) \leq \left(z_t - L_t\right) \mu\left(S_\ell\left(h_t, L_t\right)\right),$$

where z_t is as in Lemma 3. By Corollary 1, $\mu\left(S_\ell\left(h_t, L_t\right)\right) \leq \frac{m+1}{U_t - L_t} \mu\left(S_h\left(h_t, L_t\right)\right)$, which combined with the above inequality gives that:

$$\frac{\mu\left(S_h\left(h_t, L_t\right)\right) - \mu\left(S_h\left(h_t, z_t\right)\right)}{\mu\left(S_h\left(h_t, L_t\right)\right)} \leq \frac{z_t - L_t}{U_t - L_t} \left(m+1\right).$$

By Theorem 3, the LHS is at least $^1/_e$, and $z_t - L_t = 2\epsilon_t$, giving the claimed result. □

F SVM optimization

We'll now move onto a discussion of how we propose implementing the SVMOptimizer of Algorithm 2. The easier-to-analyze approach, based on an inner SDCA optimization over w [24] and an outer cutting plane optimization over b (Algorithm 3), will be described in Appendices F.1 and F.2. The easier-to-implement version, which simply calls an off-the-shelf SVM solver, will be described in Appendix F.3.

F.1 SDCA w-optimization

To simplify the presentation, we're going to begin by reformulating Equation 7 in such a way that all of the datasets are "mashed together", with the coefficients being defined on a per-example basis, rather than per-dataset. To this end, for fixed w' and b', we define, for every $i \in \{1, \dots, k\}$ and every $x \in D_i$:

$$\check{\alpha}_{i,x}^{(0)} = \begin{cases} \alpha_i^{(0)} & \text{if } \langle w', x\rangle - b' \le 1/2 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

$$\check{\beta}_{i,x}^{(0)} = \begin{cases} \beta_i^{(0)} & \text{if } \langle w', x\rangle - b' \ge -1/2 \\ 0 & \text{otherwise} \end{cases} .$$

This takes care of the loss coefficients. For the constraint coefficients, define:

$$\check{\alpha}_{i,x}^{(j)} = \begin{cases} \alpha_i^{(j)} & \text{if } \langle w', x\rangle - b' \le 1/2 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

$$\check{\beta}_{i,x}^{(j)} = \begin{cases} \beta_i^{(j)} & \text{if } \langle w', x\rangle - b' \ge -1/2 \\ 0 & \text{otherwise} \end{cases} .$$

and finally, we need to handle the constraint upper bounds:

$$\begin{aligned} \check{\gamma}^{(j)} =& \gamma^{(j)} - \sum_{i=1}^{k} \frac{1}{|D_i|} \Big(\alpha_i^{(j)} \left|\{x \in D_i \mid \langle w', x\rangle - b' > 1/2\}\right| \\ &+ \beta_i^{(j)} \left|\{x \in D_i \mid \langle w', x\rangle - b' < -1/2\}\right| \Big) . \end{aligned} \tag{15}$$

Observe that the $\check{\alpha}_{i,x}^{(0)}$s, $\check{\beta}_{i,x}^{(0)}$s, $\check{\alpha}_{i,x}^{(j)}$s, $\check{\beta}_{i,x}^{(j)}$s, and $\check{\gamma}^{(j)}$s all have implicit dependencies on w' and b'. In terms of these definitions, the Ψ defined in Equation 7 can be written as:

$$\begin{aligned} \Psi\left(w, b, v; w', b'\right) =& \sum_{i=1}^{k} \frac{1}{|D_i|} \sum_{x \in D_i} \left(\left(\check{\alpha}_{i,x}^{(0)} + \sum_{j=1}^{m} v_j \check{\alpha}_{i,x}^{(j)} \right) \max\left\{0, \frac{1}{2} + \left(\langle w, x\rangle - b\right)\right\} \right. \\ &+ \left. \left(\check{\beta}_{i,x}^{(0)} + \sum_{j=1}^{m} v_j \check{\beta}_{i,x}^{(j)} \right) \max\left\{0, \frac{1}{2} - \left(\langle w, x\rangle - b\right)\right\} \right) \\ &+ \frac{\lambda}{2} \left\|w\right\|_2^2 - \sum_{j=1}^{m} v_j \check{\gamma}^{(j)} . \end{aligned}$$

This formulation makes it clear that minimizing Ψ as a function of w and b is equivalent to optimizing an SVM, since Ψ is just a positive linear combination of hinge losses, plus a ℓ^2 regularizer, plus a term that does not depend on w or b. Since Ψ can have both "positive" and "negative" hinge losses associated with the same example, however, it's slightly simpler to combine both hinge losses together into a single piecewise linear per-example loss, rather than decomposing it into two separate hinges:

$$\ell_{i,x}\left(z\right) = \check{\alpha}_{i,x} \max\left\{0, \frac{1}{2} + z\right\} + \check{\beta}_{i,x} \max\left\{0, \frac{1}{2} - z\right\} , \tag{16}$$

where:

$$\check{\alpha}_{i,x} = \frac{n}{|D_i|} \left(\check{\alpha}_{i,x}^{(0)} + \sum_{j=1}^{m} v_j \check{\alpha}_{i,x}^{(j)} \right) \quad \text{and} \quad \check{\beta}_{i,x} = \frac{n}{|D_i|} \left(\check{\beta}_{i,x}^{(0)} + \sum_{j=1}^{m} v_j \check{\beta}_{i,x}^{(j)} \right) . \tag{17}$$

Here, $n = \sum_{i=1}^{k} \left|D_i\right|$ is the total number of examples across all of the datasets—we introduced the n factor here so that Ψ will be written in terms of the *average* loss (rather than the *total* loss). Although it is not represented explicitly in our notation, it should be emphasized that $\ell_{i,x}$ implicitly depends on v, w' and b'.

As the sum of two hinges, the $\ell_{i,x}$s are Lipschitz continuous in z, with the Lipschitz constant being:

$$L = \max_{i \in \{1,\dots,k\}} \frac{n}{\left|D_i\right|} \left(\left(\alpha_i^{(0)} + \beta_i^{(0)} \right) + \sum_{j=1}^{m} v_j \left(\alpha_i^{(j)} + \beta_i^{(j)} \right) \right). \tag{18}$$

Notice that, if the datasets are comparable in size, then $n / \left|D_i\right|$ will be on the order of k, so L will typically not be as large as the n-dependence of its definition would appear to imply.

We may now write Ψ in terms of the loss functions $\ell_{i,x}$:

$$\Psi\left(w, b, v; w', b'\right) = \frac{1}{n} \sum_{i=1}^{k} \sum_{x \in D_i} \ell_{i,x}\left(\langle w, x \rangle - b\right) + \frac{\lambda}{2} \left\|w\right\|_2^2 - \sum_{j=1}^{m} v_j \tilde{\gamma}^{(j)}.$$

This is the form considered by Shalev-Shwartz and Zhang [24], so we may apply SDCA:

Theorem 4. *If we use SDCA [24] to optimize Equation 19 for fixed b and v, then we will find a suboptimal solution with duality gap ϵ'' after performing at most:*

$$T_{\epsilon''} = O\left(\max\left\{ 0, n \ln\left(\frac{\lambda n}{L^2 X^2} \right) \right\} + n + \frac{L^2 X^2}{\lambda \epsilon''} \right)$$

iterations, where $X = \max_{i \in \{1,\dots,k\}} \max_{x \in D_i} \left\|x\right\|_2$ is a uniform upper bound on the norms of the training examples.

Proof. This is Theorem 2 of Shalev-Shwartz and Zhang [24]. □

SDCA works by, rather than directly minimizing Ψ over w, instead maximizing the following over the dual variables ξ:

$$\Psi^*\left(\xi, b, v; w', b'\right) = \tag{19}$$
$$-\frac{1}{n} \sum_{i=1}^{k} \sum_{x \in D_i} \ell_{i,x}^*\left(\xi_{i,x}\right) - \frac{1}{2\lambda} \left\| \frac{1}{n} \sum_{i=1}^{k} \sum_{x \in D_i} \xi_{i,x} x \right\|_2^2 - \frac{1}{n} \sum_{i=1}^{k} \sum_{x \in D_i} \xi_{i,x} b - \sum_{j=1}^{m} v_j \tilde{\gamma}^{(j)},$$

using stochastic coordinate ascent, where:

$$w = -\frac{1}{\lambda n} \sum_{i=1}^{k} \sum_{x \in D_i} \xi_{i,x} x$$

is the primal solution w corresponding to a given set of dual variables ξ, and:

$$\ell_{i,x}^*\left(\xi_{i,x}\right) = \frac{1}{2} \left| \xi_{i,x} - \check{\alpha}_{i,x} + \check{\beta}_{i,x} \right| - \frac{1}{2} \left(\check{\alpha}_{i,x} + \check{\beta}_{i,x} \right)$$

is the Fenchel conjugate of $\ell_{i,x}$, and is defined for $-\check{\beta}_{i,x} \leq \xi_{i,x} \leq \check{\alpha}_{i,x}$ (these bounds become box constraints on the ξs of Equation 19).

F.2 Cutting plane b-optimization

Having described in the previous section how we may optimize over w for fixed b and v using SDCA, we now move on to the problem of creating the SVMOptimizer needed by Algorithm 2, which must optimize over both w and b.

Many linear SVM optimizers do not natively handle an unregularized bias parameter b, and this has long been recognized as a potential issue. For example, Shalev-Shwartz et al. [25] suggest using Pegasos to perform inner optimizations over w, and a bisection-based outer optimization over b. Our proposal is basically this, except that Algorithm 3, rather than using bisection, optimizes over b using essentially the same cutting plane algorithm as we used in Algorithm 2, except that optimizing over b is a minimization problem (over v is maximization), and we might increase u_0' on line 2 of Algorithm 3 for a technical reason (it will be needed by the proof of Lemma 6, but is probably not helpful in practice).

Algorithm 3 Skeleton of a cutting-plane algorithm that finds a $b \in \mathcal{B}$ minimizing (to within ϵ) $\min_{b \in \mathcal{B}, w \in \mathbb{R}^d} \Psi(w, b, v; w', b')$, where $\mathcal{B} \subseteq \mathbb{R}$ is a closed interval. It is assumed that $\tilde{u}'_0 \in \mathbb{R}$ is a finite upper bound on $\min_{b \in \mathcal{B}, w \in \mathbb{R}^d} \Psi(w, b, v; w', b')$, while by the definition of Ψ (Equation 7), the l'_0 chosen on line 1 will lower bound the same quantity. The u'_0 increase that is "maybe" performed on line 2, and the CutChooser function on line 9, are discussed in Appendix F.2. The SDCAOptimizer function is as described in Appendix F.1.

```
   SVMOptimizer $(v, \tilde{u}'_0, \epsilon')$
1    Initialize $g'_0 \in \mathbb{R}$ to zero and $l'_0 = -\sum_{j=1}^{m} v_j \check{\gamma}^{(j)}$
2    *Maybe* set $u'_0 = 2\tilde{u}'_0 - l'_0$, otherwise $u'_0 = \tilde{u}'_0$        *// needed for Lemma 6*
3    For $t \in \{1, 2, \dots\}$
4        Let $h'_t(b) = \max_{s \in \{0,1,\dots,t-1\}} (l'_s + g'_s(b - b_s))$
5        Let $L'_t = \min_{b \in \mathcal{B}} h'_t(b)$ and $U'_t = \min_{s \in \{0,1,\dots,t-1\}} u'_s$
6        If $U'_t - L'_t \leq \epsilon'$ then
7            Let $s \in \{1, \dots, t-1\}$ be an index minimizing $u'_s$
8            Return $w^{(s)}$, $b_s$, $L'_t$
9        Let $b_t, \epsilon'_t$ = CutChooser $(h'_t, U'_t)$
10       Let $\xi^{(t)}, w^{(t)}$ = SDCAOptimizer $(b_t, v, \epsilon'_t)$
11       Let $u'_t = \Psi(w^{(t)}, b_t, v; w', b')$
12       Let $l'_t = \Psi^*(\xi^{(t)}, b_t, v; w', b')$ and $g'_t = \frac{\partial}{\partial'_b} \Psi^*(\xi^{(t)}, b_t, v; w', b')$
```

F.2.1 Minimization-based

Perhaps the easiest-to-implement version of Algorithm 3 is based on the idea of simply solving for the minimizer of h'_t at every iteration.

Definition 3. ***(Minimization-based Algorithm 3)*** *Do* not *increase u'_0 on line 2, and have CutChooser choose $b_t = \operatorname{argmin}_{b \in \mathcal{B}} h'_t(b)$ and $\epsilon'_t = (U_t - L_t)/2$.*

As was the case in Appendix E.1, we have no convergence rates for this version. Furthermore, since this is a one-dimensional problem, the center of mass-based version of Algorithm 3 is implementable and efficient, so this minimization-based approach is not recommended.

F.2.2 Center of mass-based

Essentially the same center of mass-based approach as was described in Appendix E.2 can be used in this setting, except that we must find the center of mass of a 2-dimensional sublevel epigraph, rather than an $m + 1$-dimensional superlevel hypograph:

Definition 4. ***(Center of mass-based Algorithm 3)*** Do *increase u'_0 on line 2, have CutChooser take (b_t, z_t) to be the center of mass of $\{(b, z) \mid h'_t(b) \leq z \leq U'_t\}$, and choose $\epsilon'_t = (U'_t - z_t)/2$.*

Unlike in Appendix E.1, the fact that this problem is one-dimensional enables us to efficiently implement this CutChooser by explicitly representing each h'_t as a set of piecewise linear segments, over which computing an integral (and therefore the center of mass) is straightforward, with a runtime that is linear in the number of segments.

Due to the similarity between Algorithms 3 and 2, we can simply recycle the results of Appendix E.2, with the troublesome second term in the bound of Lemma 4 removed by combining the "maybe" portion of Algorithm 3 with the Lipschitz continuity of Ψ as a function of b:

Lemma 6. *In the context of Algorithm 3, suppose that we choose b_t and ϵ'_t as in Definition 4. Then there is a iteration count $T_{\epsilon'}$ satisfying:*

$$T_{\epsilon'} = O\left(\ln\left(\frac{LB\left(\tilde{u}'_0 - l'_0\right)}{\epsilon'}\right)\right),$$

such that, if $t \geq T_{\epsilon'}$, then $U'_t - L'_t \leq \epsilon'$, where B is the length of $\mathcal{B}$ and L is as in Equation 18. Hence, Algorithm 3 will terminate after $T_{\epsilon'}$ iterations.

Proof. Starting from (and adapting) the final equation in the proof of Lemma 4:

$$\begin{aligned} U_t' - L_t' \leq & 4\left(1-\frac{1}{2e}\right)^{t-1}\left(\frac{1}{\epsilon'}\right)(u_0' - l_0')^2 \\ & \cdot \left(\frac{B}{\mu\left(\{b \in \mathcal{B} \mid \min_{w\in\mathbb{R}^d} \Psi\left(w, b, v; w', b'\right) \leq u_0'\}\right)}\right). \end{aligned}$$

Observe that, as a function of b, $\Psi\left(w, b, v; w', b'\right)$ is L-Lipschitz. Hence, if we let $w^* \in \mathbb{R}^d, b^* \in \mathcal{B}$ be the optimal weight and bias, then:

$$\mu\left(\{b \in \mathcal{B} \mid \Psi\left(w^*, b, v; w', b'\right) \leq u_0'\}\right) \geq \min\left\{B, \frac{u_0' - \Psi\left(w^*, b^*, v; w', b'\right)}{L}\right\}.$$

Since $\min_{w\in\mathbb{R}^d} \Psi\left(w, b, v; w', b'\right) \leq \Psi\left(w^*, b, v; w', b'\right)$, it follows that:

$$U_t' - L_t' \leq 4\left(1-\frac{1}{2e}\right)^{t-1}\left(\frac{1}{\epsilon'}\right)(u_0' - l_0')^2 \max\left\{1, \frac{LB}{u_0' - \Psi\left(w^*, b^*, v; w', b'\right)}\right\}.$$

This is the reason that we increased u_0' on line 2 of Algorithm 3, since doing so has the result that $u_0' - \Psi\left(w^*, b^*, v; w', b'\right) \geq \tilde{u}_0' - l_0'$. Since we also have that $u_0' - l_0' = 2(\tilde{u}_0' - l_0')$:

$$U_t' - L_t' \leq 16\left(1-\frac{1}{2e}\right)^{t-1}\left(\frac{1}{\epsilon'}\right)(\tilde{u}_0' - l_0') \max\left\{\tilde{u}_0' - l_0', LB\right\}.$$

The same reasoning as was used in the proof of Lemma 4 then gives the claimed bound on $T_{\epsilon'}$. □

In addition to the above result, the obvious analogue of Lemma 5 holds as well:

Lemma 7. *In the context of Algorithm 3, suppose that we choose b_t and ϵ_t' as in Definition 4. Then:*

$$\epsilon_t' \geq \frac{U_t' - L_t'}{2e},$$

and in particular, for all t (before termination):

$$\epsilon_t' \geq \frac{\epsilon'}{2e},$$

since we terminate as soon as $U_t' - L_t' \leq \epsilon'$.

Proof. Same as Lemma 5. □

In Appendix G, we'll combine these results with those of Appendices F.1 and E to bound the overall convergence rate of Algorithm 2.

F.3 Kernelization

The foregoing discussion covers the case in which we wish to learn a linear classifier, and use an SVM optimizer (SDCA) that doesn't handle an unregularized bias. It's clear that we could freely substitute another linear SVM optimizer for SDCA, as long as it finds both a primal and dual solution so that we can calculate the lower and upper bounds required by Algorithm 2.

Our technique is easily kernelized—the resulting algorithm simply depends on inner kernel SVM optimizations, rather than linear SVM optimizations. SDCA can be used in the kernel setting, but the per-iteration cost increases from $O(d)$ arithmetic operations to $O(n)$ kernel evaluations, where n is the total size of all of the datasets. Kernel-specific optimizers, such as LIBSVM [6], will generally work better than SDCA in practice, since they typically have the same per-iteration cost, but each iteration is "smarter". More importantly, such optimizers usually jointly optimize over w and b, eliminating the need for Algorithm 3 entirely—in other words, these algorithms could be used to implement the higher-level SVMOptimizer, instead of the lower-level SDCAOptimizer. For this reason, an implementation based on such an optimizer is the simplest version of our proposed approach.

G Overall convergence rates

We may now combine the results in Appendices E and F into one bound on the overall convergence rate of Algorithm 2, assuming that we use Algorithm 3, rather than an off-the-shelf SVM solver, to implement the SVMOptimizer:

Theorem 5. *Suppose that we take $l_0 = -\sum_{j=1}^{m} v_j \gamma^{(j)}$ in Algorithm 2, that SVMOptimizer is implemented as in Algorithm 3, and that the CutChooser functions in Algorithms 2 and 3 are implemented using the center of mass (as in Definitions 2 and 4). Then Algorithm 2 will perform:*

$$O\left(m \ln\left(\frac{u_0 - l_0}{\epsilon} \right) + \ln\left(\frac{\mu\left(\mathcal{V}\right)}{\mu\left(S_\ell\left(F, l_0\right)\right)} \right) \right)$$

iterations, each of which contains a single call to Algorithm 3, with each such call requiring:

$$O\left(\ln\left(\frac{LBm\left(u_0 - l_0\right)}{\epsilon} \right) \right)$$

iterations, each of which contains a single call to SDCAOptimizer, with each such call requiring:

$$O\left(\max\left\{ 0, n \ln\left(\frac{\lambda n}{L^2 X^2} \right) \right\} + n + \frac{L^2 X^2 m}{\lambda \epsilon} \right)$$

iterations, each of which requires $O(d)$ arithmetic operations.

Proof. Notice that $u_0 \geq \tilde{u}_0' \geq l_0' \geq l_0$ (the first inequality because Algorithm 2 passes a quantity upper bounded by u_0 to SVMOptimizer, and the second by our choice of l_0). By Lemma 5, we also have that $\epsilon' \geq \epsilon / 2e(m+1)$. The claimed results follow immediately from these facts, combined with Lemmas 4, 5, 6 and 7 and Theorem 4. □

We can simplify (or perhaps *over*simplify) this result by considering only the total number of training examples n, number of constraints m, number of datasets k, dimension d and desired suboptimality ϵ, dropping all of the other factors, and assuming that the sizes of the k datasets differ only by a constant factor (so that, as explained in Appendix F.1, we can take the Lipschitz constant L to be of order k). Then the overall cost of finding an ϵ-suboptimal solution to Problem 3 will be $\tilde{O}\left(dnm + dm^2k^2/\epsilon\right)$ total arithmetic operations in the inner SDCA optimizers, plus $O\left(m \ln^2\left(k/\epsilon\right)\right)$ calls to the center of mass oracles in Algorithms 2 and 3, and another $O\left(m \ln^2\left(k/\epsilon\right)\right)$ calls to a linear programming oracle for finding U_t in Algorithm 2 and L_t in Algorithm 3.

We must reiterate that, as we mentioned in Appendix E.2, finding the center of mass is a computationally difficult problem. Hence, our reliance on a center of mass oracle for the optimization over v is unrealistic (there is no problem when optimizing over b, since the underlying problem is one-dimensional). With that said, we hope that these results can provide a basis for future work.