

---

# Rounding Methods for Discrete Linear Classification (Extended Version)

---

Yann Chevaleyre

LIPN, CNRS UMR 7030, Université Paris Nord, 99 Avenue Jean-Baptiste Clément, 93430 Villetaneuse, France

Frédéric Koriche

CRIL, CNRS UMR 8188, Université d'Artois, Rue Jean Souvraz SP 18, 62307 Lens, France

Jean-Daniel Zucker

INSERM U872, Université Pierre et Marie Curie, 15 Rue de l'Ecole de Médecine, 75005 Paris, France

CHEVALEYRE@LIPN.UNIV-PARIS13.FR

KORICHE@CRIL.FR

JEAN-DANIEL.ZUCKER@IRD.FR

## Abstract

Learning *discrete* linear classifiers is known as a difficult challenge. In this paper, this learning task is cast as combinatorial optimization problem: given a training sample formed by positive and negative feature vectors in the Euclidean space, the goal is to find a discrete linear function that minimizes the cumulative hinge loss of the sample. Since this problem is NP-hard, we examine two simple rounding algorithms that discretize the fractional solution of the problem. Generalization bounds are derived for several classes of binary-weighted linear functions, by analyzing the Rademacher complexity of these classes and by establishing approximation bounds for our rounding algorithms. Our methods are evaluated on both synthetic and real-world data.

## 1. Introduction

Linear classification is a well-studied learning problem in which one needs to extrapolate, from a set of positive and negative examples represented in Euclidean space by their feature vector, a linear hypothesis  $h(x) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle - b)$  that correctly classifies future, unseen, examples. In the past decades, a wide variety of theoretical results and efficient algorithms have been obtained for learning *real-weighted* linear functions (also known as “perceptrons”). Notably, it is well-known that the linear classification problem can

be cast as a convex optimization problem and solved in polynomial time by support vector machines if the performance of hypotheses is measured by convex loss functions such as the hinge loss (see e.g. [Shawe-Taylor and Cristianini \(2000\)](#)). Much less is known, however, about learning *discrete* linear classifier. Indeed, integer weights, and in particular  $\{0, 1\}$ -valued and  $\{-1, 0, 1\}$ -valued weights, can play a crucial role in many application domains in which the classifier has to be interpretable by humans.

One of the main motivating applications for this work comes from the field of *quantitative metagenomics*, which is the study of the collective genome of the micro-organisms inhabiting our body. It is now technically possible to measure the abundance of bacterial species by measuring the activity of specific tracer genes for that species. Moreover, it is known that the abundance of some bacterial species in our body is related to obesity or leanness. Instead of learning a standard linear classifier to predict obesity, biologists would like to find two small groups of bacterial species, such that if the abundance of bacteria in the first group is greater than that of the second group, then the individual is classified as being obese. Given a dataset in which features represent the abundance of specific bacterial species, this problem boils down to learning a linear classifier with  $\{-1, 0, 1\}$ -valued weights.

In other domains such as medical diagnosis, the interpretability of predictive models is also a key aspect. The most common diagnostic models are *M-of-N* rules ([Towell and Shavlik, 1993](#)) according to which patients are classified as ill if at least  $M$  criteria among  $N$  are satisfied. However, learning *M-of-N* rules is hard (a proof is provided in appendix). In binary classification, linear threshold functions with  $\{0, 1\}$ -valued

weights are equivalent to  $M$ -of- $N$  rules. Thus, the theory and the algorithms described in this paper can also be used to learn such rules, as shown in the experimental section.

Perhaps the major obstacle to the development of discrete linear functions lies in the fact that, in the standard distribution-free PAC learning model, the problem of finding an integer-weighted linear function that is consistent with a training set is equivalent to the (Zero-One) Integer Linear Programming problem (Pitt and Valiant, 1988), which is NP-complete. In order to alleviate this issue, several authors have investigated the learnability of discrete linear functions in distribution-specific models, such as the uniform distribution (Golea and Marchand, 1993a; Köhler et al., 1990; Oppen et al., 1990; Venkatesh, 1991), or the product distribution (Golea and Marchand, 1993b). Yet, beyond this pioneering work, many questions remain open, especially when the model is distribution-free but the loss functions are convex.

In this paper, we consider just such a scenario by examining the problem of learning binary-weighted linear functions with the *hinge loss*, a well-known surrogate of the zero-one loss. The key components of the classification problem are a set  $\mathcal{C} \subseteq \{0, 1\}^n$  of boolean vectors<sup>1</sup> from which the learner picks his hypotheses, and a fixed (yet hidden) probability distribution over the set  $\mathbb{R}^n \times \{\pm 1\}$  of examples. For a hinge parameter  $\gamma > 0$ , the hinge loss penalizes a hypothesis  $\mathbf{c} \in \mathcal{C}$  on an example  $(\mathbf{x}, y)$  if its margin  $y \langle \mathbf{c}, \mathbf{x} \rangle$  is less than  $\gamma$ .

The performance of a hypothesis  $\mathbf{c} \in \mathcal{C}$  is measured by its *risk*, denoted  $\text{risk}(\mathbf{c})$ , and defined as the expected loss of  $\mathbf{c}$  on an example  $(\mathbf{x}, y)$  drawn from the underlying distribution. Typically,  $\text{risk}(\mathbf{c})$  is upper-bounded by the sum of two terms: a sample estimate  $\text{risk}_m(\mathbf{c})$  of the performance of  $\mathbf{c}$  and a penalty term  $\mathcal{T}_m(\mathcal{C})$  that depends on the hypothesis class  $\mathcal{C}$  and, potentially, also on the training set. The sample estimate  $\text{risk}_m(\mathbf{c})$  is simply the averaged cumulative hinge loss of  $\mathbf{c}$  on a set  $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$  of examples drawn independently from the underlying distribution. The penalty term  $\mathcal{T}_m(\mathcal{C})$  can be given by the VC-dimension of  $\mathcal{C}$ , or its Rademacher complexity with respect to the size  $m$  of the training set. For binary-weighted linear classifiers, the penalty term induced by their Rademacher complexity can be substantially smaller than the penalty term induced by their VC dimension. So, by a simple adaptation of Bartlett and Mendelson’s framework

<sup>1</sup>As explained in Section 4.2,  $\{-1, 0, 1\}$ -weighted classification can be reduced to  $\{0, 1\}$ -weighted classification.

(2002), our risk bounds take the form of:

$$\text{risk}(\mathbf{c}) \leq \text{risk}_m(\mathbf{c}) + \frac{2}{\gamma} \mathcal{R}_m(\mathcal{C}) + \sqrt{\frac{8 \ln(2/\delta)}{m}} \quad (1)$$

where  $\mathcal{R}_m(\mathcal{C})$  is the Rademacher complexity of  $\mathcal{C}$  with respect to  $m$ , and  $\delta \in (0, 1)$  is a confidence parameter.

Ideally, we would like to have at our disposal an efficient algorithm for minimizing  $\text{risk}_m(\mathbf{c})$ . The resulting minimizer, say  $\mathbf{c}^*$ , would be guaranteed to provide an optimal hypothesis because the other terms in the risk bound (1) do not depend on the choice of the hypothesis. Unfortunately, because the class  $\mathcal{C}$  of discrete linear classifiers is not a convex set, the convexity of hinge loss does not help in finding  $\mathbf{c}^*$  and, as shown by Theorem 1 in the next section, the optimization problem remains NP-hard.

The key message to be gleaned from this paper is that the convexity of the loss function *does* help in approximating the combinatorial optimization problem, using simple rounding methods. Our first algorithm is a standard *randomized rounding* (RR) method that starts from a fractional solution  $\mathbf{w}^*$  in the convex hull of  $\mathcal{C}$ , and then builds  $\mathbf{c}$  by viewing the fractional value  $w_i^*$  as the probability that  $c_i$  should be set to 1. The second algorithm, called *greedy rounding* (GR), is essentially a derandomization of RR that iteratively rounds the coordinates of the fractional solution by maintaining a constraint on the sum of weights.

For the class  $\mathcal{C}$  of binary-weighted linear functions, we show that the greedy rounding algorithm is guaranteed to return a concept  $\mathbf{c} \in \mathcal{C}$  satisfying:

$$\text{risk}_m(\mathbf{c}) \leq \text{risk}_m(\mathbf{c}^*) + \frac{X_2}{2\gamma}$$

where  $X_p = \max_{i=1}^m \|\mathbf{x}_i\|_p$ , and  $\|\mathbf{x}\|_p$  is the  $L_p$ -norm of  $\mathbf{x}$ . We also show that the problem of improving this bound up to a constant factor is NP-hard. Combining greedy rounding’s performance with the Rademacher complexity of  $\mathcal{C}$  yields the risk bound:

$$\begin{aligned} \text{risk}(\mathbf{c}) &\leq \text{risk}_m(\mathbf{c}^*) \\ &+ \frac{X_2}{2\gamma} + \frac{2}{\gamma} X_1 \min \left\{ 1, \sqrt{\frac{n}{m}} \right\} + \sqrt{\frac{8 \ln(2/\delta)}{m}} \end{aligned}$$

For the subclass  $\mathcal{C}_k$  of *sparse* binary-weighted linear functions involving at most  $k$  ones among  $n$ , we show that greedy rounding is guaranteed to return a concept  $\mathbf{c} \in \mathcal{C}_k$  satisfying:

$$\text{risk}_m(\mathbf{c}) \leq \text{risk}_m(\mathbf{c}^*) + \frac{X_\infty \sqrt{k}}{\gamma}$$

Using the Rademacher complexity of  $\mathcal{C}_k$ , which is substantially smaller than that of  $\mathcal{C}$ , we have:

$$\begin{aligned} \text{risk}(\mathbf{c}) &\leq \text{risk}_m(\mathbf{c}^*) \\ &+ \frac{X_\infty \sqrt{k}}{\gamma} + \frac{2}{\gamma} X_\infty k \sqrt{\frac{2 \log \frac{n}{k}}{m}} + \sqrt{\frac{8 \ln(2/\delta)}{m}} \end{aligned}$$

Similar results are derived with the randomized rounding algorithm, with less sharp bounds due to the randomization process. We evaluate these rounding methods on a both synthetic and real-world datasets, showing good performance in comparison with standard linear optimization methods.

The proofs of preparatory lemmas 2 and 5 can be found in appendix.

## 2. Binary-Weighted Linear Classifiers

**Notation.** The set of positive integers  $\{1, \dots, n\}$  is denoted  $[n]$ . For a subset  $\mathcal{S} \subseteq \mathbb{R}^n$ , we denote by  $\text{conv}(\mathcal{S})$  the convex hull of  $\mathcal{S}$ . For two vectors  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$  and  $p \geq 1$  the  $L_p$ -norm of  $\mathbf{u}$  is denoted  $\|\mathbf{u}\|_p$  and the inner product between  $\mathbf{u}$  and  $\mathbf{v}$  is denoted  $\langle \mathbf{u}, \mathbf{v} \rangle$ . Given a vector  $\mathbf{u} \in \mathbb{R}^n$  and  $k \in [n]$ , we denote by  $\mathbf{u}_{1:k}$  the prefix  $(u_1, \dots, u_k)$  of  $\mathbf{u}$ . Finally, given a training set  $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ , we write  $X_p = \max_{i=1}^m \|\mathbf{x}_i\|_p$ .

In this study, we shall examine classification problems for which the set of instances is the Euclidean space  $\mathbb{R}^n$  and the hypothesis class is a subset of  $\{0, 1\}^n$ . Specifically, we shall focus on the class  $\mathcal{C} = \{0, 1\}^n$  of all binary-weighted linear functions, and the subclass  $\mathcal{C}_k$  of all binary-weighted linear functions with at most  $k$  ones among  $n$ . The parameterized loss function  $\ell_\gamma : \mathbb{R} \times \{\pm 1\} \rightarrow \mathbb{R}$  examined in this work is the *hinge loss* defined by:

$$\ell_\gamma(p, y) = \frac{1}{\gamma} \max(0, \gamma - py) \text{ where } \gamma > 0$$

### 2.1. Computational Complexity

For a training set  $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ , the empirical risk of a weight vector  $\mathbf{c} \in \mathcal{C}$ , denoted  $\text{risk}_m(\mathbf{c})$ , is defined by its averaged cumulative loss:

$$\text{risk}_m(\mathbf{c}) = \frac{1}{m} \sum_{i=1}^m \ell_\gamma(\langle \mathbf{c}, \mathbf{x}_i \rangle, y)$$

By  $\mathbf{c}^*$ , we denote any minimizer of the objective function  $\text{risk}_m$ . Recall that if  $\mathcal{C}$  is a convex subset of  $\mathbb{R}^n$  then  $\mathbf{c}^*$  can be found in polynomial time using convex optimization algorithms. However, for the discrete class  $\mathcal{C} = \{0, 1\}^n$ , the next result states that the optimization problem is much harder.

**Theorem 1.** *There exists a constant  $\alpha > 0$  such that, unless  $P=NP$ , there is no polynomial time algorithm capable of learning from any dataset of size  $m$  a hypothesis  $\mathbf{c} \in \mathcal{C}$  such that:*

$$\text{risk}_m(\mathbf{c}) \leq \min_{\mathbf{c}' \in \mathcal{C}} (\text{risk}_m(\mathbf{c}')) + \alpha \frac{X_2}{\gamma}$$

*Proof.* In what follows, we denote by  $\mathbf{c}^*$  any vector in  $\mathcal{C}$  for which  $\text{risk}_m(\mathbf{c}^*)$  is minimal. For an undirected graph  $G = (V, E)$ , the MAX-CUT problem is to find a subset  $S \subset V$  such that the number of edges with one end point in  $S$  and the other in  $V \setminus S$  is maximal. Unless  $P=NP$ , no polynomial-time algorithm can achieve the approximation ratio of 0.997 for MAXCUT in 3-regular graphs (Berman and Karpinski, 1999).

Based on this result, we first construct a dataset from a 3-regular graph  $G = (V, E)$  having an even number of vertices. Our dataset consist of  $n = |V| + 1$  features and  $m = 2|E|$  examples. The first  $|V|$  features are associated with the vertices of  $G$ . For each edge  $(j, j') \in E$ , we build two positively labeled examples  $\mathbf{x}$  and  $\mathbf{x}'$  in the following way. In the first example  $\mathbf{x}$ ,  $j$  and  $j'$  are set to  $\gamma$ , and all other features are set to 0. In the second example  $\mathbf{x}'$ ,  $j$  and  $j'$  are set to  $-\gamma$ , the feature  $|V| + 1$  is set to  $2\gamma$  and all others are set to 0. Consider any weight vector  $\mathbf{c}$  where  $c_{|V|+1}$  is equal to 0. Clearly, setting  $c_{|V|+1}$  to 1 will strictly decrease the loss of  $\mathbf{c}$  if at least one coordinate in  $\mathbf{c}$  is nonzero. Thus, *we will assume from now on and without loss of generality that  $c_{|V|+1}$  is always set to 1*. Observe that the loss on the two examples  $\mathbf{x}$  and  $\mathbf{x}'$  is

$$\ell(\langle \mathbf{c}, \mathbf{x} \rangle) + \ell(\langle \mathbf{c}, \mathbf{x}' \rangle) = \begin{cases} 0 & \text{if } c_j \neq c_{j'} \\ 1 & \text{otherwise} \end{cases}$$

Let us now define  $\text{cut}(\mathbf{c}) = |\{(i, j) \in E : c_i \neq c_j\}|$ . By viewing  $\mathbf{c}$  as the characteristic vector of a subset of vertices,  $\text{cut}(\mathbf{c})$  is the value of the cut in  $G$  induced by this subset. Note we have  $\text{cut}(\mathbf{c}) = |E| - 2|E| \text{risk}_m(\mathbf{c})$ . Thus, minimizing the loss on the dataset maximizes the cut on the graph. Consequently,  $\text{cut}(\mathbf{c}^*)$  is the optimal value of the Max-cut problem.

Finally, suppose by contradiction that for all  $\alpha > 0$ , there is a polynomial-time algorithm capable of learning from any dataset of size  $m$  a vector  $\mathbf{c}$  satisfying

$$\text{risk}_m(\mathbf{c}) \leq \text{risk}_m(\mathbf{c}^*) + \alpha \frac{X_2}{\gamma}$$

Notably, in the dataset constructed above, the value of  $X_2$  is  $\gamma\sqrt{6}$ . Thus, for this dataset, we get that  $\text{risk}_m(\mathbf{c}) \leq \text{risk}_m(\mathbf{c}^*) + \alpha\sqrt{6}$ , and hence,

$$\text{cut}(\mathbf{c}) \geq \text{cut}(\mathbf{c}^*) - 2\alpha|E|\sqrt{6} \quad (2)$$

To this point, Feige et al. (2001) have shown that on 3-regular graphs, the optimal cut has a value of at least  $|E|/2$ . By reporting this value into (2), we obtain  $\text{cut}(\mathbf{c}) \geq \text{cut}(\mathbf{c}^*) - 4\alpha\sqrt{6}\text{cut}(\mathbf{c}^*) = \text{cut}(\mathbf{c}^*)(1 - 4\alpha\sqrt{6})$ . Because  $\alpha$  can be arbitrarily close to 0, this implies that Max-Cut is approximable within any constant factor, which contradicts Berman and Karpinski's (1999) inapproximability result.  $\square$

## 2.2. Rademacher Complexity

Suppose that our training set  $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$  consists of examples generated by independent draws from some fixed probability distribution on  $\mathbb{R}^n \times \{\pm 1\}$ . For a class  $\mathcal{F}$  of real valued functions  $f: \mathbb{R}^n \rightarrow \mathbb{R}$ , define its Rademacher complexity on  $S$  to be:

$$\mathcal{R}_S(\mathcal{F}) = \mathbb{E} \left[ \sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \sigma_i f(\mathbf{x}_i) \right]$$

Here, the expectation is over the Rademacher random variables  $\sigma_1 \dots \sigma_m$ , which are drawn from  $\{\pm 1\}$  with equal probability. Since  $S$  is random, we can also take expectation over the choice of  $S$  and define  $\mathcal{R}_m(\mathcal{F}) = \mathbb{E}[\mathcal{R}_S(\mathcal{F})]$ , which gives us a quantity that depends on both the function class and the sample size. As indicated by inequality (1), bounds on the Rademacher complexity of a class immediately yield risk bounds for classifiers picked from that class. For continuous linear functions, sharp Rademacher complexity bounds have been provided by Kakade et al. (2008). We provide here similar bounds for two important classes of discrete linear functions.

**Lemma 2.** *Let  $\sigma_1, \dots, \sigma_m$  be Rademacher variables. Then  $\mathbb{E}[\|\sum_{i=1}^k \sigma_i\|] \geq \sqrt{k/8}$  for any even  $k \geq 2$ .*

**Theorem 3.** *Let  $\mathcal{C} = \{0, 1\}^n$  be the class of all binary-weighted linear functions. Then,*

$$\mathcal{R}_m(\mathcal{C}) \leq X_1 \min \left\{ 1, \sqrt{\frac{n}{m}} \right\}$$

*This bound is tight up to a constant factor.*

*Proof.* Consider the hypothesis class:

$$\mathcal{F}_{p,v} = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle : \mathbf{w} \in \mathbb{R}^n, \|\mathbf{w}\|_p \leq v\}$$

By Theorem 1 in (Kakade et al., 2008), we have  $\mathcal{R}_m(\mathcal{F}_{p,v}) \leq vX_2/\sqrt{m}$ . Moreover, for any  $\mathbf{c} \in \{0, 1\}^n$ , we have  $\|\mathbf{c}\|_2 \leq \sqrt{n}$ . It follows that  $\mathcal{C} \subseteq \mathcal{F}_{2,\sqrt{n}}$ , and since  $\|\cdot\|_2 \leq \|\cdot\|_1$ , we get that:

$$\mathcal{R}_m(\mathcal{C}) \leq \mathcal{R}(\mathcal{F}_{2,\sqrt{n}}) \leq X_2\sqrt{\frac{n}{m}} \leq X_1\sqrt{\frac{n}{m}}$$

Now, let us prove that this bound is tight. First, let us rewrite the rademacher complexity over samples in a more convenient form:

$$\begin{aligned} \mathcal{R}_S(\mathcal{C}) &= \frac{1}{m} \sum_{j=1}^n \mathbb{E} \left[ \sup_{c_j \in \{0,1\}} c_j \sum_{i=1}^m \sigma_i x_{i,j} \right] \\ &= \frac{1}{2m} \sum_{j=1}^n \mathbb{E} \left[ \sup_{w_j \in \{-1,1\}} w_j \sum_{i=1}^m \sigma_i x_{i,j} \right] \\ &= \frac{1}{2m} \sum_{j=1}^n \mathbb{E} \left[ \left| \sum_{i=1}^m \sigma_i \cdot x_{i,j} \right| \right] \end{aligned} \quad (3)$$

For the case  $n \geq m$ , consider a training set  $S$  such that  $x_{i,i} = X_1$  for all  $i \in [m]$ , and zero everywhere else. Clearly, equation 3 implies  $\mathcal{R}_S(\mathcal{C}) = \frac{X_1}{2}$ . For the case  $n < m$ , assume  $m$  is a multiple of  $2n$  and consider a dataset  $S$  in which each example contains only one non-zero value equal to  $X_1$ , and such that the number of nonzero values per column is  $\frac{m}{n}$ . Then, by applying Lemma 2 to equation 3, we obtain:

$$\mathcal{R}_S(\mathcal{C}) \geq \frac{nX_1}{m} \sqrt{\frac{m}{32}} = X_1 \sqrt{\frac{n}{32m}}$$

$\square$

**Theorem 4.** *For a constant  $k > 0$ , let  $\mathcal{C}_k$  be the class of binary-weighted linear functions with at most  $k$  ones among  $n$ . Then,*

$$\mathcal{R}_m(\mathcal{C}_k) \leq X_\infty k \sqrt{\frac{2 \log \frac{n}{k}}{m}}$$

*Proof.* For a closed convex set  $\mathcal{S} \subset \mathbb{R}_+^n$ , consider the hypothesis class:

$$\mathcal{F}_S = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle : \mathbf{w} \in \mathcal{S}\}$$

Using the convex function  $F(\mathbf{w}) = \sum_{j=1}^n \frac{w_j}{W_1} \ln \frac{w_j}{W_1} + \ln n$ , where  $W_1 = \max_{\mathbf{w} \in \mathcal{S}} \|\mathbf{w}\|_1$ , we get from Theorem 1 in (Kakade et al., 2008):

$$\mathcal{R}_m(\mathcal{F}_S) \leq X_\infty W_1 \sqrt{\frac{2 \sup\{F(\mathbf{w}) : \mathbf{w} \in \mathcal{S}\}}{m}} \quad (4)$$

For any  $k$ , let  $\mathcal{S}_k = \text{conv}(\{\mathbf{w} \in \{0, 1\}^n : \|\mathbf{w}\|_1 \leq k\})$ , where  $\text{conv}(\cdot)$  is the convex hull. Because  $F$  is convex and  $\mathcal{S}_k$  is a convex polytope, the supremum of  $F$  is one of the vertices of the polytope. Thus,

$$\sup_{\mathbf{w} \in \mathcal{S}_k} F(\mathbf{w}) = \sup\{F(\mathbf{w}) : \mathbf{w} \in \{0, 1\}^n, \|\mathbf{w}\|_1 \leq k\}$$

$$\leq \ln n + \sum_{l=1}^k \frac{1}{k} \ln \frac{1}{k} = \ln \frac{n}{k}$$

The result follows by reporting this value into (4).  $\square$

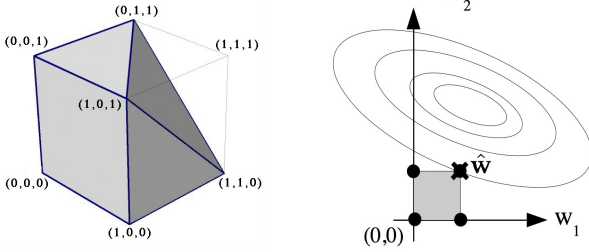


Figure 1. (left) Intersection of the  $l_1$  ball of radius 2, of the  $l_\infty$  ball of radius 1 for non negative coordinates. (right) The solution to the convex relaxation coincides with the solution to the original problem.

### 3. Rounding Methods

This section exploits the convexity of the hinge loss to derive simple approximation algorithms for minimizing empirical risk. The overall idea is to first relax the optimization problem by deriving a fractional solution  $\mathbf{w}^*$ , and then to round the solution  $\mathbf{w}^*$  using a deterministic or a randomized method. The convex optimization setting we consider is defined by:

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w} \in [0,1]^n \cap \mathcal{S}} \operatorname{risk}_m(\mathbf{w}) \quad (5)$$

where  $\mathcal{S} = \mathbb{R}_+^n$  for the hypothesis class  $\mathcal{C} = \{0,1\}^n$ , and  $\mathcal{S} = \{\mathbf{w} \in \mathbb{R}_+^n : \|\mathbf{w}\|_1 \leq k\}$  for the subclass  $\mathcal{C}_k$  of binary-weighted linear functions with at most  $k$  ones among  $n$ . Note that the empirical risk minimization problem for  $\mathcal{C}$  can be viewed as an optimization problem over  $\mathbb{R}_+^n$  under the  $L_\infty$ -norm constraint. The problem of minimizing empirical risk in the convex hull of  $\mathcal{C}_k$  is illustrated in the left part of Figure 1.

The accuracy of rounding methods depend on the number of non-fractional values in the relaxed solution  $\mathbf{w}^*$ . Indeed, if most weights of  $\mathbf{w}^*$  are already in  $\{0,1\}$ , then these values will remain unchanged by the rounding phase, and the final approximation  $\mathbf{c}$  will close to  $\mathbf{w}^*$ . Figure 1 illustrates this phenomenon by representing a case where  $\mathbf{w}^*$  and  $\mathbf{c}$  coincide. The objective function is represented by ellipses, and the four dots at the corner of the square are the vectors  $\{0,1\}^2$ .

The hinge loss also takes an important part in the quality of the rounding process. Increasing the parameter  $\gamma$  increases the likelihood that weights become binary. Taking this to the extreme, if  $\gamma \geq X_1$ , then the hinge loss is linear inside the  $[0,1]^n$  hypercube and all convex optimization tasks described above will yield solutions with binary weights. We note in passing that a similar phenomenon arises in the Lasso feature selection procedure, where the weight vectors are more likely to fall on a vertex of the  $L_1$ -ball as the margin increases.

---

#### Algorithm 1 Randomized Rounding (RR)

---

Parameters: A set of  $m$  examples, a convex set  $\mathcal{S}$

1. Solve  $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w} \in [0,1]^n \cap \mathcal{S}} \operatorname{risk}_m(\mathbf{w})$
  2. For each  $i \in [n]$ , set  $c_i$  to 1 with probability  $w_i$
  3. Return  $\mathbf{c}$
- 

#### 3.1. Randomized Rounding

The randomized rounding (RR) algorithm is one of the most popular approximation schemes for combinatorial optimization (Raghavan and Thompson, 1987; Williamson and Shmoys, 2011). In the setting of our framework, the algorithm starts from the fractional solution of the problem and draws a random concept  $\mathbf{c} \in \mathcal{C}_k$  by choosing each value  $c_i$  independently to 1 with probability  $w_i^*$  and to 0 with probability  $1 - w_i^*$ . The following lemma (derived from Bernstein's inequality) states that using  $\mathbf{c}$  instead of  $\mathbf{w}^*$  to compute a dot product yields a bounded deviation.

**Lemma 5.** *Let  $\mathbf{x} \in \mathbb{R}^n$ ,  $\mathbf{w}^* \in [0,1]^n$  and  $\mathbf{c} \in \{0,1\}^n$  be a random vector such that  $\mathbb{P}[c_i = 1] = w_i^*$  for all  $i \in 1 \dots n$ . Then, with probability at least  $1 - \delta$ , the following inequalities hold:*

$$\langle \mathbf{c}, \mathbf{x} \rangle \in \left[ \langle \mathbf{w}^*, \mathbf{x} \rangle \pm 1.52 \|\mathbf{x}\|_2 \ln \frac{2}{\delta} \right] \quad \text{and}$$

$$\langle \mathbf{c}, \mathbf{x} \rangle \in \left[ \langle \mathbf{w}^*, \mathbf{x} \rangle \pm \|\mathbf{x}\|_\infty \left( \frac{2}{3} + 1.7 \sqrt{\|\mathbf{w}^*\|_1} \right) \ln \frac{2}{\delta} \right]$$

**Theorem 6.** *Let  $\mathbf{c}$  be the vector returned by the randomized rounding algorithm. Then, with probability  $1 - \delta$ , the following hold:*

- For the class  $\mathcal{C}$ :  
 $\operatorname{risk}_m(\mathbf{c}) \leq \operatorname{risk}_m(\mathbf{c}^*) + \frac{1.52}{\gamma} X_2 \ln \frac{2m}{\delta}$
- For the class  $\mathcal{C}_k$ :  
 $\operatorname{risk}_m(\mathbf{c}) \leq \operatorname{risk}_m(\mathbf{c}^*) + \frac{\frac{2}{3} + 1.7 \sqrt{\|\mathbf{w}^*\|_1}}{\gamma} X_\infty \ln \frac{2m}{\delta}$

*Proof.* Since the  $\gamma$ -hinge loss is  $\frac{1}{\gamma}$ -Lipschitz, we have:

$$\begin{aligned} \operatorname{risk}(\mathbf{c}) - \operatorname{risk}(\mathbf{c}^*) &\leq \operatorname{risk}(\mathbf{c}) - \operatorname{risk}(\mathbf{w}^*) \\ &\leq \frac{1}{\gamma m} \sum_{i=1}^m |\langle \mathbf{c}, \mathbf{x}_i \rangle - \langle \mathbf{w}^*, \mathbf{x}_i \rangle| \quad (6) \end{aligned}$$

Taking expectations and applying the union bound on Lemma 5, we get with probability  $1 - \delta'$  that:

$$\begin{aligned} \mathbb{P}[\exists i \in [m], |\langle \mathbf{c}, \mathbf{x}_i \rangle - \langle \mathbf{w}^*, \mathbf{x}_i \rangle| \geq t] \\ \leq \sum_{i=1}^m \mathbb{P}[|\langle \mathbf{c}, \mathbf{x}_i \rangle - \langle \mathbf{w}^*, \mathbf{x}_i \rangle| \geq t] \leq m\delta' \end{aligned}$$

**Algorithm 2** Greedy Rounding (GR)

 Parameters: A set of  $m$  examples, an integer  $k \leq n$ 

1. Solve  $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w} \in [0,1]^n \cap \mathcal{S}} \operatorname{risk}_m(\mathbf{w})$
2. For  $k = 1$  to  $n$ , set

$$A_k \leftarrow \{a \in \{0, 1\} : \forall i = 1 \dots m$$

$$\begin{aligned} & (\theta_{i,k-1} + x_{i,k}(a - w_k^*))^2 \\ & \leq \theta_{i,k-1}^2 + x_{i,k}^2 w_k^* (1 - w_k^*) \} \end{aligned}$$

$$c_k \leftarrow \operatorname{argmin}_{a \in A_k} \operatorname{risk}_m(c_1, \dots, c_{k-1}, a, w_{k+1}^*, \dots, w_n^*)$$

3. return  $(c_1 \dots c_n)$

The result follows by setting  $\delta = m\delta'$  and reporting into (6) the values  $t = 1.52X_2 \ln \frac{2}{\delta'}$  and  $t = \left(\frac{2}{3} + 1.7\sqrt{\|\mathbf{w}^*\|_1}\right) X_\infty \ln \frac{2}{\delta'}$ .  $\square$

### 3.2. Greedy Rounding

Despite its relative weak guarantees, the randomized rounding procedure can be used as a building block for constructing more efficient algorithms. Specifically, the new approximation scheme we propose, called Greedy Rounding (GR), is essentially a derandomization of RR with some improvements. As described in Algorithm 2, the procedure starts again by computing the fractional solution  $\mathbf{w}^*$  of the optimization task (Line 1). Then, the coordinates of  $\mathbf{w}^*$  are rounded in a sequential manner by simply maintaining a constraint on the admissible values (Line 2). The algorithm uses a matrix  $\Theta = [\theta_{i,k}]$  of parameters defined as follows. For any  $k \in [n]$ , let  $\mathbf{c}_{1:k} = (c_1, \dots, c_k)$  be the prefix of the vector  $\mathbf{c}$  build at the end of step  $k$ . Then,  $\theta_{i,k} = \sum_{j=1}^k x_{i,j}(c_j - w_j^*)$  for each  $i \in [m]$ .

The next result, which we call the *derandomization lemma*, shows that at each step  $k$  of the rounding procedure, there is a value  $a \in \{0, 1\}$  which does not increase the loss too much.

**Lemma 7.** *For any  $k \leq n$  and any  $(c_1 \dots c_{k-1}) \in \{0, 1\}^{k-1}$ , there exist  $a \in \{0, 1\}$  such that for all  $i \in [m]$ ,*

$$(\theta_{i,k-1} + x_{i,k}(a - w_k^*))^2 \leq \theta_{i,k-1}^2 + x_{i,k}^2 w_k^* (1 - w_k^*)$$

*Proof.* Let  $\mathbf{z}$  be a random vector taking values in  $\{0, 1\}^n$  such that  $\mathbb{P}[z_j = 1] = w_j^*$  for all  $j \in [n]$ . Clearly, we have  $\mathbb{E}[\mathbf{z} - \mathbf{w}^*] = \mathbf{0}$ . For any  $i \in [m]$ , let  $f_i(\mathbf{z}, \mathbf{w}^*) = \langle \mathbf{z} - \mathbf{w}^*, \mathbf{x}_i \rangle^2$ . We can observe that  $\mathbb{E}[f_i(\mathbf{z}, \mathbf{w}^*)] = \sum_{j=1}^n x_{i,j}^2 w_j^* (1 - w_j^*)$ . Taking condi-

tional expectations, we have:

$$\mathbb{E}[f_i(\mathbf{z}, \mathbf{w}^*) \mid \mathbf{z}_{1:k-1} = \mathbf{c}_{1:k-1}] =$$

$$\mathbb{E}\left[\langle \mathbf{c}_{1:k-1} - \mathbf{w}_{1:k-1}^*, \mathbf{x}_{i,1:k-1} \rangle^2 + \langle \mathbf{z}_{k:n} - \mathbf{w}_{k:n}^*, \mathbf{x}_{i,k:n} \rangle^2\right]$$

In the right hand side of this equation, the squared sum is equal to the sum of squares because the term  $\langle \mathbf{c}_{1:k-1} - \mathbf{w}_{1:k-1}^*, \mathbf{x}_{i,1:k-1} \rangle \cdot \langle \mathbf{z}_{k:n} - \mathbf{w}_{k:n}^*, \mathbf{x}_{i,k:n} \rangle$  is null in expectation. We get that:

$$\mathbb{E}[f_i(\mathbf{z}, \mathbf{w}^*) \mid \mathbf{z}_{1:k-1} = \mathbf{c}_{1:k-1}] =$$

$$= \left( \sum_{j=1}^{k-1} x_{i,j}(c_j - w_j^*) \right)^2 + \sum_{j=k}^n x_{i,j}^2 w_j^* (1 - w_j^*)$$

$$= \theta_{i,k-1}^2 + \sum_{j=k}^n x_{i,j}^2 w_j^* (1 - w_j^*)$$

Now, let  $U_1, \dots, U_n$  denote random variables taking values in some domain  $\mathcal{D} \subseteq \mathbb{R}$ , and  $g$  be a function from  $\mathcal{D}^n$  into  $\mathbb{R}$ . Using the definition of conditional expectation, we know that for any  $j \in [n]$  there exists a value  $u \in \mathcal{D}$  such that  $\mathbb{E}[g(U_1, \dots, U_n) \mid U_j = u] \leq \mathbb{E}[g(U_1, \dots, U_n)]$ . By application, there exists a value  $c_k \in \{0, 1\}$  such that  $\mathbb{E}[f_i(\mathbf{z}, \mathbf{w}^*) \mid \mathbf{z}_{1:k} = \mathbf{c}_{1:k}] \leq \mathbb{E}[f_i(\mathbf{z}, \mathbf{w}^*) \mid \mathbf{z}_{1:k-1} = \mathbf{c}_{1:k-1}]$ . The result follows using  $a = c_k$ .  $\square$

Based on this lemma, the approximation guarantees of the greedy rounding algorithm are summarized in the next theorem. Interestingly, a comparison with the lower bound for the class  $\mathcal{C}$  obtained in Theorem 1 reveals that the approximation bound of GR for this class is tight up to a constant factor. In other words, GR is an optimal approximation algorithm for the class of binary-weighted linear functions.

**Theorem 8.** *Let  $\mathbf{c}$  be the vector returned by the GR algorithm. Then,*

- For the class  $\mathcal{C}$ ,  $\operatorname{risk}_m(\mathbf{c}) \leq \operatorname{risk}_m(\mathbf{c}^*) + X_2/2\gamma$
- For the class  $\mathcal{C}_k$ ,  $\operatorname{risk}_m(\mathbf{c}) \leq \operatorname{risk}_m(\mathbf{c}^*) + X_\infty \sqrt{k}/\gamma$

*Proof.* Since the  $\gamma$ -hinge loss is  $1/\gamma$ -Lipschitz, we can use inequality (6) to derive that:

$$\operatorname{risk}_m(\mathbf{c}) - \operatorname{risk}_m(\mathbf{c}^*) \leq \frac{1}{\gamma m} \sum_{i=1}^m |\langle \mathbf{c}, \mathbf{x}_i \rangle - \langle \mathbf{w}^*, \mathbf{x}_i \rangle|$$

$$\leq \frac{1}{\gamma m} \sum_{i=1}^m \langle \mathbf{c} - \mathbf{w}^*, \mathbf{x}_i \rangle$$

$$\leq \frac{1}{\gamma m} \sum_{i=1}^m |\theta_{i,n}| \quad (7)$$

Now, by application of lemma 7, we know that for each step  $k$  of GR, any value  $a \in A_k$  is such that  $(\theta_{i,k-1} + x_{i,k}(a - w_k^*))^2 \leq \theta_{i,k-1}^2 + x_{i,k}^2 w_k^*(1 - w_k^*)$  for all  $i \in [m]$ . Since  $c_k \in A_k$ , we must have  $\theta_{i,k}^2 \leq \sum_{j \in [k]} x_{i,j}^2 w_j^*(1 - w_j^*)$  for all  $i \in [m]$  and  $k \in [n]$ . Reporting this inequality into (7),

$$\text{risk}_m(\mathbf{c}) - \text{risk}_m(\mathbf{c}^*) \leq \frac{1}{\gamma m} \sum_{i=1}^m \sqrt{\sum_{j=1}^n x_{i,j}^2 w_j^*(1 - w_j^*)}$$

Let  $R = \text{risk}_m(\mathbf{c}) - \text{risk}_m(\mathbf{c}^*)$ . For the class  $\mathcal{C}$ , using the fact that  $w_j^*(1 - w_j^*) \leq \frac{1}{4}$  we have  $R \leq X_2/(2\gamma)$ . For the class  $\mathcal{C}_k$ , using Hölder's inequality and the fact that  $\|\mathbf{w}^*\|_1 \leq k$ , we obtain  $R \leq X_\infty \sqrt{k}/\gamma$ .  $\square$

## 4. Experiments

We tested the empirical performance of our algorithms by conducting experiments on a synthetic problem and several real-world domains. Besides the Randomized Rounding (RR) algorithm and the Greedy Rounding (GR) algorithm, we evaluated the behavior of two fractional optimization techniques: the Convex (Cvx) optimization method that returns the fractional solution of the problem specified by (5), and the Support Vector Machine (L1-SVM) that solves the  $\ell_1$ -constrained version of the problem. For small datasets, we also evaluated MIP (mixed integer programming) which is the exact solution to the combinatorial problem.

In our implementation of the algorithms, we used the linear programming software CPLEX that returns the fractional solution of convex optimization tasks.

### 4.1. Synthetic Data

In order to validate different aspects of our algorithms, we designed a simple artificial dataset generator. Called with parameters  $k, n, m, \eta$ , the generator builds a dataset composed of  $m$  examples, each with  $n$  features. Examples are drawn from a uniform distribution over  $[-10, 10]^n$ . Also, the generator draws randomly a target function with exactly  $k$  ones, and each example is labeled with respect to this target. Finally, the coordinates of each example are perturbed with a normal law of standard deviation  $\eta$ .

We first evaluated the generalization performance of the optimization algorithms. Setting  $k = 10$ ,  $n = 100$ ,  $\eta = 0.1$ , we generated datasets with an increasing number  $m$  of examples, and plotted the generalization zero-one loss measured on test data (upper part of figure 2). Next, we evaluated the robustness of our algorithms with respect to irrelevant attributes. Setting  $k = 10$ ,  $m = 50$ ,  $\eta = 0.1$ , we generated datasets with

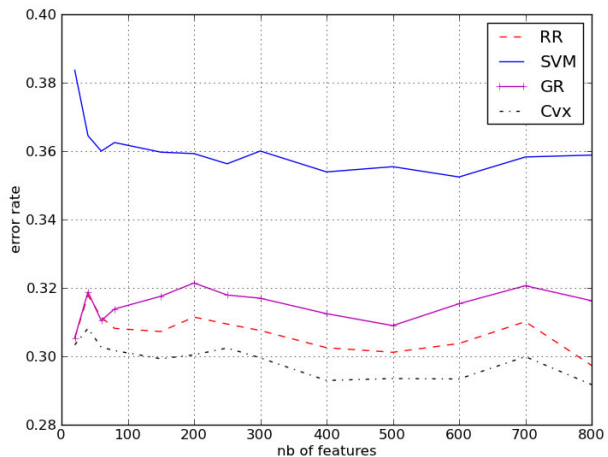
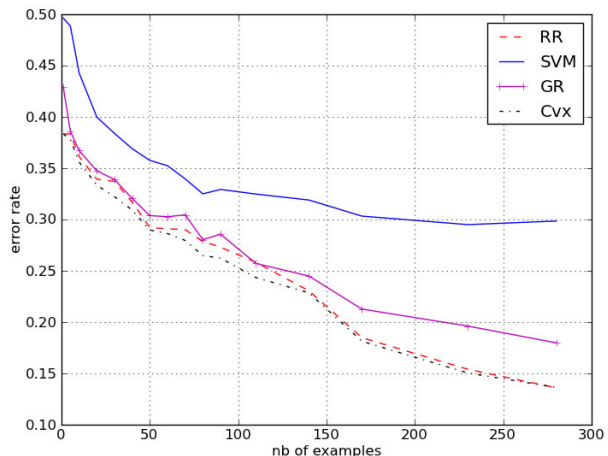


Figure 2. Test error rates on synthetic data, comparing the number of examples (upper part) and the number of irrelevant features (lower part)

$n$  varying from 20 to 800, and plotted again the generalization zero-one loss, measured on test data (bottom part of figure 2). It is apparent that GR and RR perform significantly better than L1-SVM, which is not surprising, because the target concepts have  $\{0, 1\}$ -weights. On synthetic data, GR is slightly less accurate than RR, whose performance is close to Cvx.

### 4.2. Metagenomic Data

In metagenomic classification, discrete linear functions have a natural interpretation in term of bacterial abundance. We used a real-world dataset containing 38 individuals and 69 features. The dataset is divided into two well-balanced classes: obese people and non obese. Each feature represents the abundance of a bacterial species. As mentioned in the introduction, the weight

$k$	L1-SVM	Cvx	RR	GR	MIP
10	0.46	0.48	0.41	0.43	0.35
20	0.46	0.44	0.41	0.44	
$\infty$	0.44	0.40	0.39	0.43	
run time	0.02s	0.02s	0.04s	0.98s	13s

Table 1. Test error rates and average running time in seconds on metagenomic data

of each feature captures a qualitative effect encoded by a value in  $\{-1, 0, +1\}$  (negative, null effect, positive). Let  $POS$  (respectively  $NEG$ ) denote the group of bacterial species whose feature has a weight of 1 (respectively  $-1$ ). If the abundance of all bacteria in  $POS$  is greater than the abundance of the bacteria in  $NEG$ , then the individual will be classified as obese.

In order to learn ternary-weighted linear functions with our algorithms, we used a simple trick that reduces the classification task to a binary-weighted learning problem. The idea is to duplicate attributes in the following way: to each instance  $\mathbf{x} \in \mathbb{R}^n$  we associate an instance  $\mathbf{x}' \in \mathbb{R}^d$  where  $d = 2n$  and  $\mathbf{x}' = (x_1, -x_1, x_2, -x_2, \dots, x_n, -x_n)$ . Given a binary-weighted concept  $\mathbf{c}' \in \{0, 1\}^d$ , the corresponding ternary-weighted concept  $\mathbf{c} \in \{-1, 0, +1\}^n$  is recovered by setting  $c_i = c'_{2i-1} - c'_{2i}$ . Based on this transformation, it is easy to see that  $\ell_\gamma(\mathbf{c}'; \mathbf{x}', y) = \ell_\gamma(\mathbf{c}; \mathbf{x}, y)$ . So, if  $\mathbf{c}'$  minimizes empirical risk on the set  $\{(\mathbf{x}'_t, y_t)\}$ , then  $\mathbf{c}$  minimizes empirical risk on  $\{(\mathbf{x}_t, y_t)\}$ . If, in addition,  $\mathbf{c}'$  is  $k$ -sparse, then  $\mathbf{c}$  is  $k$ -sparse.

The test error rates of algorithms are reported in Table 4.2. Test errors was measured by conducting 10 fold cross validation, averaged over 10 experiments. In light of these results, it is apparent that RR slightly outperforms both SVM and Cvx, which clearly overfit the data even in presence of the  $L_1$ -ball constraint (for the first two rows). Unsurprisingly, the MIP solver generated a model superior to the others. For  $k \geq 20$ , the mixed integer program did not finish in reasonable time, so we left the corresponding entries of the table blank. In a nutshell, we can conclude that the accuracy does not suffer from switching to ternary weights, but this learning task looks challenging.

### 4.3. Colon cancer

To demonstrate the performance of discrete linear classifiers for gene selection, we applied our algorithms to microarray data on colon cancer, which is publicly available. The dataset consists of 62 samples, 22 of which are normal and 40 of which are from colon cancer tissues. The genes are already pre-filtered, consisting of the 2,000 genes. We launched our algorithms

L1-SVM	Cvx	RR	RR $\times$ 5
0.15	0.15	0.2	0.164
0.04s	0.04s	0.04s	2.22s

Table 2. Test error rates and average running time in seconds on colon cancer data

If at least 3 of the following conditions are met, then the mushroom is poisonous
<i>bruises = yes</i> <i>odor</i> $\in \{almond, foul, musty, none, pungent\}$ <i>gill_attachment = attached</i> <i>gill_spacing = crowded</i> <i>stalk_root = rooted</i> <i>stalk_color_above_ring = pink</i> <i>stalk_color_below_ring = pink</i> <i>ring_number = one</i> <i>ring_type</i> $\in \{large, pendant\}$ <i>spore_print_color = brown</i>

Table 3.  $M$ -of- $N$  rule for the mushrooms dataset

with  $k = 15$  to select 15 genes only. We did not plot the result of GR because each run of GR took a huge amount of time. Instead, RR  $\times$  5 is a variant of randomized rounding that selects the best out of 5 random roundings at each time step. It turns out that RR  $\times$  5 achieves a much better error rate than RR in this case (but not on the datasets of the previous subsections). Thus, we obtain a concept much simpler than the linear hypothesis generated by the SVM, with comparable accuracy.

### 4.4. Mushrooms

Finally, we ran experiments on the “mushrooms” dataset to evaluate how  $M$ -of- $N$  rules are learnt using rounding algorithms. This dataset contains 22 features which are all nominal. We transformed these features into binary features, and ran our discrete linear learning algorithms on this dataset, without imposing any cardinality constraint.

With an accuracy of 98%, the  $M$ -of- $N$  rule shown in Table 3 was produced. We ran 10 times 10 fold cross validation with our algorithms (see table 4). Algorithm Cvx achieves a perfect classification. Here, GR outperforms RR, but running RR several times (and choosing the best solution) considerably improves the accuracy results.



Cvx	RR	RR×20	GR
0	0.6	0.014	0.023
0.24s	0.26s	0.74s	11s

Table 4. Test error rates and average running time in seconds on colon cancer data

## 5. supplemental material

Results and proofs shown in this section only appear in the extended version of this work.

**Theorem 9.** For a training set  $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ , checking if there exists a  $m$ -of- $n$  rule achieving zero error is NP-hard.

*Proof.* By reduction from the exact 3-cover problem. Let  $U = \{1 \dots m\}$  be a set of elements and  $\mathcal{C} = \{C_1 \dots C_n\}$  a collection of 3-subsets of  $U$ . There exists an exact 3-cover of  $U$  iff there exists  $\mathcal{C}' \subseteq \mathcal{C}$  such that each element of  $U$  is covered exactly once by some subset of  $\mathcal{C}'$ . Let us build our dataset, consisting of  $2m + 3$  examples and  $n + 2$  features. As usual, the  $i^{\text{th}}$  example is  $\mathbf{x}_i = (x_{i,1} \dots x_{i,n+2})$  and its label is  $y_i$ . To begin, let us describe the  $2m$  first examples. For each  $i \in U$ , we have a positive example  $\mathbf{x}_i$  and a negative example  $\mathbf{x}_{m+i}$ . We have  $x_{i,n+2} = 1$  and  $x_{i,n+1} = x_{m+i,n+1} = x_{m+i,n+2} = 0$ . Also, for all  $i \in \{1 \dots m\}$  and  $j \in \{1 \dots n\}$ , we have  $x_{i,j} = x_{m+i,j} = 1$  if  $i \in C_j$ , otherwise  $x_{i,j} = x_{m+i,j} = 0$ . The last three examples are build as follows. We have  $y_{2m+1} = +1$  and  $y_{2m+2} = y_{2m+3} = -1$ . Finally, we have  $x_{2m+1,n+1} = x_{2m+1,n+2} = x_{2m+2,n+1} = x_{2m+3,n+1} = 1$ . All other attributes of last three examples are set to zero. This construction is summarized on figure ??.

Let us now prove that  $\mathcal{C}'$  is a solution to the exact 3-cover problem if and only if there is a rule classifying the data correctly. Let us start with the *only if* part. It is straightforward to check that if  $\mathcal{C}'$  is a solution to the X3C problem, then the rule “if *at least 2* features among the subset  $\mathcal{C}' \cup \{n + 1, n + 2\}$  are set to one, then the example is positive” correctly classifies the data. Now the *if* part. Assume there exists some learnt rule correctly classifying the data. To correctly classify the last three examples, such a learnt rule must necessarily be of the form “if *at least 2* features among the subset  $\mathcal{C}' \cup \{n + 1, n + 2\}$  are set to one, then the example is positive”. So each positive examples must be covered by at least two features. Let us show that each positive example is covered by *exactly* two features. First note that each example  $\mathbf{x}_i$  for  $i \in \{1 \dots m\}$  is covered once by the feature  $n + 2$ . Assume by contradiction that example  $\mathbf{x}_i$  is also covered more than once by the first  $n$  features. Then, the

$a_1 \dots a_n$	$a_{n+1}$	$a_{n+2}$	lab
	0	1	+
$C_1 \dots C_n$	$\vdots$	$\vdots$	$\vdots$
	0	1	+
$C_1 \dots C_n$	$\vdots$	0	−
	0	0	−
0	1	1	+
0	0	1	−
0	1	0	−

Table 5. Summary of the construction of theorem ??

negative example  $\mathbf{x}_{i+m}$  would be incorrectly classified. Thus, the set  $\mathcal{C}'$  consists in an exact 3-set cover.  $\square$

*Proof. of lemma 2*

Let  $\sigma_1 \dots \sigma_n$  denote rademacher random variables. Let us show that for any  $k \geq 2$ , we have  $\mathbb{E} \left[ \left| \sum_{i=1}^k \sigma_i \right| \right] > \sqrt{\frac{k}{8}}$  is  $k$  is even, and  $\mathbb{E} \left[ \left| \sum_{i=1}^k \sigma_i \right| \right] > \sqrt{\frac{k-1}{8}}$  is  $k$  is odd. Let  $z$  be a binomial random variable with parameters  $p = \frac{1}{2}$  and  $k$ . Then the mean deviation (see e.g. (?) for the definition) is  $MD = \mathbb{E} [|z - \mathbb{E}[z]|] = \frac{1}{2^k} \left( \binom{k}{\frac{k}{2}} + 1 \right) \binom{k}{\frac{k}{2} + 1}$ . If  $k$  is even, then  $\binom{k}{\frac{k}{2} + 1} = \frac{k! \frac{k}{2}}{\frac{k}{2}! \frac{k}{2}! (\frac{k}{2} + 1)} = \binom{k}{\frac{k}{2}} \frac{k}{k+2} \geq \frac{1}{2} \binom{k}{\frac{k}{2}}$  for  $k \geq 2$ . Then from Stirling’s formula, we get  $\binom{k}{\frac{k}{2}} > \frac{2^{k-1}}{\sqrt{\frac{k}{2}}} = \frac{2^k}{\sqrt{2k}}$  (see e.g.

corrolary 2.9 of (?)). Thus,  $MD > \frac{k}{4\sqrt{2k}} = \frac{1}{4} \sqrt{\frac{k}{2}}$ . Now note that  $\mathbb{E} \left[ \left| \sum_{i=1}^k \sigma_i \right| \right] = 2\mathbb{E} [|z - \mathbb{E}[z]|]$ . Thus, for even  $k$ , we get  $\mathbb{E} \left[ \left| \sum_{i=1}^k \sigma_i \right| \right] > \sqrt{\frac{k}{8}}$ . Also, if  $k$  is odd, we can write  $\mathbb{E} \left[ \left| \sum_{i=1}^k \sigma_i \right| \right] > \mathbb{E} \left[ \left| \sum_{i=1}^{k-1} \sigma_i \right| \right] > \sqrt{\frac{k-1}{8}}$ . Let us now derive the upper bound:  $\mathbb{E} \left[ \left| \sum_{i=1}^k \sigma_i \right| \right] \leq \sqrt{\mathbb{E} \left[ \left( \sum_{i=1}^k \sigma_i \right)^2 \right]} = \sqrt{\text{Var} \left( \sum_{i=1}^k \sigma_i \right)} = \sqrt{k}$ .  $\square$

*Proof. of lemma 5* Let  $\mathbf{z} \in \mathbb{R}^n$  be a random vector defined as follows  $z_i = x_i (c_i - w_i)$ . Then, we have  $\mathbb{E}[z_i] = 0$  and  $\mathbb{E}[z_i^2] = x_i^2 (w_i - w_i^2) = x_i^2 w_i (1 - w_i) \leq \frac{x_i^2}{4}$ . Because the variables  $z_i$  are independent zero mean random variables, we can apply the following Bernstein’s inequality:

$$\mathbb{P} \left[ \left| \sum z_i \right| > t \right] \leq 2 \exp \left\{ - \frac{t^2/2}{\sum_i \mathbb{E} z_i^2 + \|\mathbf{z}\|_\infty t/3} \right\}$$

To derive the first bound, we start by noting that  $\sum_i \mathbb{E} z_i^2 \leq \frac{1}{4} \|\mathbf{x}\|_2^2$ . Thus, rewriting Bernstein inequality and bounding it by  $\delta$  yields

$$\mathbb{P} \left[ \sum z_i > t \right] \leq 2 \exp \left\{ -\frac{t^2/2}{\frac{1}{4} \|\mathbf{x}\|_2^2 + \|\mathbf{x}\|_\infty \frac{t}{3}} \right\} \leq \delta$$

By taking the logarithm, this last inequation becomes

$\frac{t^2}{2} - \ln \frac{2}{\delta} \left( \frac{1}{4} \|\mathbf{x}\|_2^2 + \|\mathbf{x}\|_\infty \frac{t}{3} \right) \geq 0$ , which is a polynomial inequation of the form  $at^2 + bt + c \geq 0$  with  $a = \frac{1}{2}$ ,  $b = -\frac{1}{3} \|\mathbf{x}\|_\infty \ln \frac{2}{\delta}$  and  $c = -\frac{1}{4} \|\mathbf{x}\|_2^2 \ln \frac{2}{\delta}$ . Let  $t^* = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$ . First note that any  $t \geq t^*$  is a solution to this inequation. We will be interested in finding an  $t \geq t^*$  (thus satisfying Bernstein's inequality) which has a more readable form than  $t^*$ .

$$\begin{aligned} t^* &= \frac{-b + \sqrt{b^2 - 4ac}}{2a} \\ &\leq \frac{-2b + \sqrt{-4ac}}{2a} \\ &= \frac{2}{3} \ln \frac{2}{\delta} \|\mathbf{x}\|_\infty + \sqrt{\frac{1}{2} \ln \frac{2}{\delta} \|\mathbf{x}\|_2^2} \\ &\leq \ln \frac{2}{\delta} \left( \frac{2}{3} \|\mathbf{x}\|_\infty + \|\mathbf{x}\|_2 \frac{1}{\sqrt{2 \ln \frac{2}{\delta}}} \right) \\ &\leq \ln \frac{2}{\delta} \|\mathbf{x}\|_2 \left( \frac{2}{3} + \frac{1}{\sqrt{2 \ln \frac{2}{\delta}}} \right) \\ &\leq 1.52 \|\mathbf{x}\|_2 \ln \frac{2}{\delta} = t \end{aligned}$$

Note that the first inequality holds because  $\sqrt{u+v} \leq \sqrt{u} + \sqrt{v}$ . Also the last inequality holds because for any  $\delta \leq 1$ , we have  $\frac{2}{3} + \frac{1}{\sqrt{2 \ln \frac{2}{\delta}}} \leq 1.52$ .

Now let us prove the second bound with the exact same technique. Note that  $\sum_i \mathbb{E} z_i^2 \leq \sum_i x_i^2 w_i \leq \|\mathbf{x}\|_\infty^2 W_1$ . Plugging this formula into Bernstein's bound, we get:

$$\mathbb{P} \left[ \sum z_i > t \right] \leq 2 \exp \left\{ -\frac{t^2/2}{W_1 \|\mathbf{x}\|_\infty^2 + \|\mathbf{x}\|_\infty \frac{t}{3}} \right\} \leq \delta$$

Again, this can be written as a polynomial inequation  $at^2 + bt + c \geq 0$  where  $a = 1/2$ ,  $b = -\|\mathbf{x}\|_\infty \frac{1}{3} \ln \frac{2}{\delta}$ ,  $c = -W_1 \|\mathbf{x}\|_\infty^2 \ln \frac{2}{\delta}$ . A solution is  $t^\dagger = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$ . Looking for an upper bound on  $t^\dagger$ , we get

$$\begin{aligned} t^\dagger &= \frac{-b + \sqrt{b^2 - 4ac}}{2a} \\ &\leq \frac{-2b + \sqrt{-4ac}}{2a} \\ &= \frac{2}{3} \|\mathbf{x}\|_\infty \ln \frac{2}{\delta} + \sqrt{2W_1 \|\mathbf{x}\|_\infty^2 \ln \frac{2}{\delta}} \\ &\leq \|\mathbf{x}\|_\infty \ln \frac{2}{\delta} \left( \frac{2}{3} + \sqrt{\frac{2}{\ln \frac{2}{\delta}}} \cdot \sqrt{W_1} \right) \\ &\leq \|\mathbf{x}\|_\infty \ln \frac{2}{\delta} \left( \frac{2}{3} + 1.7\sqrt{W_1} \right) \end{aligned}$$

Last line holds because  $\sqrt{\frac{2}{\ln \frac{2}{\delta}}} \leq 1.7$  for any  $\delta \leq 1$   $\square$

## References

- P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3: 463–482, 2002.
- P. Berman and M. Karpinski. On some tighter inapproximability results (extended abstract). In *Automata, Languages and Programming, 26th International Colloquium (ICALP)*, pages 200–209. Springer, 1999.
- Y. Chevaleyre, F. Koriche, and J. D. Zucker. Rounding methods for discrete linear classification (extended version). Technical Report hal-00771012, hal, 2013.
- U. Feige, M. Karpinski, and M. Langberg. A note on approximating Max-Bisection on regular graphs. *Information Processing Letters*, 79(4):181–188, 2001.
- M. Golea and M. Marchand. Average case analysis of the clipped Hebb rule for nonoverlapping Perception networks. In *Proceedings of the 6th annual conference on computational learning theory (COLT'93)*. ACM, 1993a.
- M. Golea and M. Marchand. On learning perceptrons with binary weights. *Neural Computation*, 5(5):767–782, 1993b.
- S. M. Kakade, K. Sridharan, and A. Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems (NIPS)*, pages 793–800, 2008.
- H. Köhler, S. Diederich, W. Kinzel, and M. Opper. Learning algorithm for a neural network with binary synapses. *Zeitschrift fr Physik B Condensed Matter*, 78:333–342, 1990.

- M. Opper, W. Kinzel, J. Kleinz, and R. Nehl. On the ability of the optimal perceptron to generalise. *Journal of Physics A: Mathematical and General*, 23(11):L581–L586, 1990.
- L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *J. ACM*, 35(4):965–984, 1988.
- P. Raghavan and C. D. Thompson. Randomized rounding: A technique for probably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.
- J. Shawe-Taylor and N. Cristianini. *An Introduction to Support Vector Machines*. Cambridge, 2000.
- G. G. Towell and J. W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13:71–101, 1993.
- S. Venkatesh. On learning binary weights for majority functions. In *Proceedings of the 4th Annual Workshop on Computational Learning Theory (COLT)*, pages 257–266. Morgan Kaufmann, 1991.
- D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge, 2011.