Journal of Machine Learning Research 19 (2018) 1-30 Submitted 7/17; Revised 2/18; Published 10/18

Simple Classification Using Binary Data

Deanna Needell DEANNA@MATH.UCLA.EDU
Department of Mathematics
520 Portola Plaza, University of California, Los Angeles, CA 90095

Rayan Saab RSAAB@UCSD.EDU
Department of Mathematics
9500 Gilman Drive, University of California, La Jolla, CA 92093

Tina Woolf TINA.WOOLF@CGU.EDU
Institute of Mathematical Sciences
150 E. 10th Street, Claremont Graduate University, Claremont CA 91711

Editor: David Wipf

Abstract

Binary, or one-bit, representations of data arise naturally in many applications, and are appealing in both hardware implementations and algorithm design. In this work, we study the problem of data classification from binary data obtained from the sign pattern of low-dimensional projections and propose a framework with low computation and resource costs. We illustrate the utility of the proposed approach through stylized and realistic numerical experiments, and provide a theoretical analysis for a simple case. We hope that our framework and analysis will serve as a foundation for studying similar types of approaches.

Keywords: binary measurements, one-bit representations, classification

1. Introduction

Our focus is on data classification problems in which only a *binary* representation of the data is available. Such binary representations may arise under a variety of circumstances. In some cases, they may arise naturally due to compressive acquisition. For example, distributed systems may have bandwidth and energy constraints that necessitate extremely coarse quantization of the measurements (Fang et al., 2014). A binary data representation can also be particularly appealing in hardware implementations because it is inexpensive to compute and promotes a fast hardware device (Jacques et al., 2013b; Laska et al., 2011); such benefits have contributed to the success, for example, of 1-bit Sigma-Delta converters (Aziz et al., 1996; Candy and Temes, 1962). Alternatively, binary, heavily quantized, or compressed representations may be part of the classification algorithm design in the interest of data compression and speed (Boufounos and Baraniuk, 2008; Hunter et al., 2010; Calderbank et al., 2009; Davenport et al., 2010; Gupta et al., 2010; Hahn et al., 2014). The goal of this paper is to present a framework for performing learning inferences, such as classification, from highly quantized data representations—we focus on the extreme case of 1-bit (binary) representations. Let us begin with the mathematical formulation of this problem.

Problem Formulation. Let $\{x_i\}_{i=1}^p \subset \mathbb{R}^n$ be a point cloud represented via a matrix

$$X = [x_1 \ x_2 \ \cdots \ x_p] \in \mathbb{R}^{n\times p}.$$

Moreover, let $A : \mathbb{R}^n \to \mathbb{R}^m$ be a linear map, and denote by $\mathrm{sign} : \mathbb{R} \to \mathbb{R}$ the sign operator given by

$$\mathrm{sign}(a) = \begin{cases} 1 & a \geq 0 \\ -1 & a < 0. \end{cases}$$

Without risk of confusion, we overload the above notation so the sign operator can apply to matrices (entrywise). In particular, for an m by p matrix M, and $(i, j) \in [m] \times [p]$, we define $\mathrm{sign}(M)$ as the $m \times p$ matrix with entries

$$(\mathrm{sign}(M))_{i,j} := \mathrm{sign}(M_{i,j}).$$

We consider the setting where a classification algorithm has access to training data of the form $Q = \mathrm{sign}(AX)$, along with a vector of associated labels $b = (b_1, \ \cdots \ , b_p) \in \{1, \dots , G\}^p$, indicating the membership of each x_i to exactly one of G classes. Here, A is an m by n matrix. The rows of A define *hyperplanes* in $\mathbb{R}^n$ and the binary sign information tells us which side of the hyperplane each data point lies on. Throughout, we will primarily take A to have independent identically distributed standard Gaussian entries (though experimental results are also included for structured matrices). Given Q and b, we wish to train an algorithm that can be used to classify new signals, available only in a similar binary form via the matrix A, for which the label is unknown.

1.1. Contribution

Our contribution is a *framework* for classifying data into a given number of classes using only a binary representation (obtained as the sign pattern from low-dimensional projections, as described above) of the data. This framework serves several purposes: (i) it provides mathematical tools that can be used for classification in applications where data is already captured in a simple binary representation, (ii) demonstrates that for general problems, classification can be done effectively using low-dimensional measurements, (iii) suggests an approach to use these measurements for classification using low computation, (iv) provides a simple technique for classification that can be mathematically analyzed. We believe this framework can be extended and utilized to build novel algorithmic approaches for many types of learning problems. In this work, we present one method for classification using training data, illustrate its promise on synthetic and real data, and provide a theoretical analysis of the proposed approach in the simple setting of two-dimensional signals and two possible classes. Under mild assumptions, we derive an explicit lower bound on the probability that a new data point gets classified correctly. This analysis serves as a foundation for analyzing the method in more complicated settings, and a framework for studying similar types of approaches.

1.2. Organization

We proceed next in Section 1.3 with a brief overview of related work. Then, in Section 2 we propose a two-stage method for classifying data into a given number of classes using only a binary representation of the data. The first stage of the method performs training on data with known class membership, and the second stage is used for classifying new data points with a priori unknown class membership. Next, in Section 3 we demonstrate the potential of the proposed approach on both synthetically generated data as well as real datasets with application to handwritten digit recognition and facial recognition. Finally, in Section 4 we provide a theoretical analysis of the proposed approach in the simple setting of two-dimensional signals and two classes. We conclude in Section 5 with some discussion and future directions.

1.3. Prior Work

There is a large body of work on several areas related to the subject of this paper, ranging from classification to compressed sensing, hashing, quantization, and deep learning. Due to the popularity and impact of each of these research areas, any review of prior work that we provide here must necessarily be non-exhaustive. Thus, in what follows, we briefly discuss related prior work, highlighting connections to our work but also stressing the distinctions.

Support vector machines (SVM) (Christianini and Shawe-Taylor, 2000; Hearst et al., 1998; Joachims, 1998; Steinwart and Christmann, 2008) have become popular in machine learning, and are often used for classification. Provided a training set of data points and known labels, the SVM problem is to construct the optimal hyperplane (or hyperplanes) separating the data (if the data is linearly separable) or maximizing the geometric margin between the classes (if the data is not linearly separable). Although loosely related (in the sense that at a high level we utilize hyperplanes to separate the data), the approach taken in this paper is fundamentally different than in SVM. Instead of searching for the *optimal* separating hyperplane, our proposed algorithm uses many, randomly selected hyperplanes (via the rows of the matrix A), and uses the relationship between these hyperplanes and the training data to construct a classification procedure that operates on information between the same hyperplanes and the data to be classified.

The process of transforming high-dimensional data points into low-dimensional spaces has been studied extensively in related contexts. For example, the pioneering Johnson-Lindenstrauss Lemma states that any set of p points in high dimensional Euclidean space can be (linearly) embedded into $O(\epsilon^{-2}\log(p))$ dimensions, without distorting the distance between any two points by more than a small factor, namely ϵ (Johnson and Lindenstrauss, 1982). Since the original work of Johnson and Lindenstrauss, much work on Johnson-Lindenstrauss embeddings (often motivated by signal processing and data analysis applications) has focused on randomized embeddings where the matrix associated with the linear embedding is drawn from an appropriate random distribution. Such random embeddings include those based on Gaussian and other subgaussian random variables as well as those that admit fast implementations, usually based on the fast Fourier transform (Ailon and Chazelle, 2006; Achlioptas, 2003; Dasgupta and Gupta, 2003).

Another important line of related work is *compressed sensing*, in which it has been demonstrated that far fewer linear measurements than dictated by traditional Nyquist sampling can be used to represent high-dimensional data (Candès et al., 2006b,a; Donoho, 2006). For a signal $x \in \mathbb{R}^n$, one obtains $m < n$ measurements of the form $y = Ax$ (or noisy measurements $y = Ax + z$ for $z \in \mathbb{R}^m$), where $A \in \mathbb{R}^{m \times n}$, and the goal is to recover the signal x. By assuming the signal x is s-sparse, meaning that $\|x\|_0 = |\mathrm{supp}(x)| = s \ll n$, the recovery problem becomes well-posed under certain conditions on A. Indeed, there is now a vast literature describing recovery results and algorithms when A, say, is a random matrix drawn from appropriate distributions (including those where the entries of A are independent Gaussian random variables). The relationship between Johnson-Lindenstrauss embeddings and compressed sensing is deep and bi-directional; matrices that yield Johnson-Lindenstrauss embeddings make excellent compressed sensing matrices (Baraniuk et al., 2006) and conversely, compressed sensing matrices (with minor modifications) yield Johnson-Lindenstrauss embeddings (Krahmer and Ward, 2011). Some initial work on performing inference tasks like classification from compressed sensing data shows promising results (Boufounos and Baraniuk, 2008; Hunter et al., 2010; Calderbank et al., 2009; Davenport et al., 2010; Gupta et al., 2010; Hahn et al., 2014).

To allow processing on digital computers, compressive measurements must often be *quantized*, or mapped to discrete values from some finite set. The extreme quantization setting where only the sign bit is acquired is known as *one-bit compressed sensing* and was introduced recently (Boufounos and Baraniuk, 2008). In this framework, the measurements now take the form $y = \mathrm{sign}(Ax)$, and the objective is still to recover the signal x. Several methods have since been developed to recover the signal x (up to normalization) from such simple one-bit measurements (Plan and Vershynin, 2013a,b; Gopi et al., 2013; Jacques et al., 2013b; Yan et al., 2012; Jacques et al., 2013a). Although the data we consider in this paper takes a similar form, the overall goal is different; rather than signal *reconstruction*, our interest is data *classification*.

More recently, there has been growing interest in binary embeddings (embeddings into the binary cube (Plan and Vershynin, 2014; Yu et al., 2014; Gong et al., 2013; Yi et al., 2015; Choromanska et al., 2016; Dirksen and Stollenwerk, 2016), where it has been observed that using certain linear projections and then applying the sign operator as a nonlinear map largely preserves information about the angular distance between vectors provided one takes sufficiently many measurements. Indeed, the measurement operators used for binary embeddings are Johnson-Lindenstrauss embeddings and thus also similar to those used in compressed sensing, so they again range from random Gaussian and subgaussian matrices to those admitting fast linear transformations, such as random circulant matrices (Dirksen and Stollenwerk, 2016), although there are limitations to such embeddings for subgaussian but non-Gaussian matrices (Plan and Vershynin, 2014, 2013a). Although we consider a similar binary measurement process, we are not necessarily concerned with geometry preservation in the low-dimensional space, but rather the ability to still perform data classification.

Deep Learning is an area of machine learning based on learning data representations using multiple levels of abstraction, or layers. Each of these layers is essentially a function whose parameters are learned, and the full network is thus a composition of such functions. Algorithms for such deep neural networks have recently obtained state of the art results for classification. Their success has been due to the availability of large training data sets coupled with advancements in computing power and the development of new techniques (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2015; Russakovsky et al., 2015). Randomization in neural networks has again been shown to give computational advantages and even so-called "shallow" networks with randomization and random initializations of deep neural networks have been shown to obtain results close to deep networks requiring heavy optimization (Rahimi and Recht, 2009; Giryes et al., 2016). Deep neural networks have also been extended to binary data, where the net represents a set of Boolean functions that maps all binary inputs to the outputs (Kim and Smaragdis, 2016; Courbariaux et al., 2015, 2016). Other types of quantizations have been proposed to reduce multiplications in both the input and hidden layers (Lin et al., 2015; Marchesi et al., 1993; Simard and Graf, 1994; Burge et al., 1999; Rastegari et al., 2016; Hubara et al., 2016). We will use randomized non-linear measurements but consider deep learning and neural networks as motivational to our multi-level algorithm design. Indeed, we are not tuning parameters nor doing any optimization as is typically done in deep learning, nor do our levels necessarily possess the structure typical in deep learning "architectures"; this makes our approach potentially simpler and easier to work with.

Using randomized non-linearities and simpler optimizations appears in several other works (Rahimi and Recht, 2009; Ozuysal et al., 2010). The latter work most closely resembles our approach in that the authors propose a "score function" using binary tests in the training phase, and then classifies new data based on the maximization of a class probability function. The perspective of this prior approach however is Bayesian rather than geometric, the score functions do not include any balancing terms as ours will below, the measurements are taken as "binary tests" using components of the data vectors (rather than our compressed sensing style projections), and the approach does not utilize a multi-level approach as ours does. We believe our geometric framework not only lends itself to easily obtained binary data but also a simpler method and analysis.

2. The Proposed Classification Algorithm

The training phase of our algorithm is detailed in Algorithm 1. Here, the method may take the binary data Q as input directly, or the training data $Q = \text{sign}(AX)$ may be computed as a one-time pre-processing step. For arbitrary matrices A, this step of course may incur a computational cost on the order of mnp. In Section 3, we also include experiments using structured matrices that have a fast multiply, reducing this cost to a logarithmic dependence on the dimension n. Then, the training algorithm proceeds in L "levels". In the ℓ-th level, m index sets $\Lambda_{\ell,i} \subset [m]$, $|\Lambda_{\ell,i}| = \ell$, $i = 1, ..., m$, are randomly selected, so that all elements of $\Lambda_{\ell,i}$ are unique, and $\Lambda_{\ell,i} \neq \Lambda_{\ell,j}$ for $i \neq j$. This is achieved by selecting the multi-set of $\Lambda_{\ell,i}$'s uniformly at random from a set of cardinality $\binom{\binom{m}{\ell}}{m}$. During the i-th "iteration" of the ℓ-th level, the rows of Q indexed by $\Lambda_{\ell,i}$ are used to form the $\ell \times p$ submatrix of Q, the columns of which define the sign patterns $\{\pm 1\}^\ell$ observed by the training data. For example, at the first level the possible sign patterns are 1 and -1, describing which side of the selected hyperplane the training data points lie on; at the second level the possible sign patters are $\begin{bmatrix}1\\1\end{bmatrix}$, $\begin{bmatrix}1\\-1\end{bmatrix}$, $\begin{bmatrix}-1\\1\end{bmatrix}$, $\begin{bmatrix}-1\\-1\end{bmatrix}$, describing which side of the two selected hyperplanes the training data points lie on, and so on for the subsequent levels. At each level, there are at most 2^ℓ possible sign patterns. Let $t = t(\ell) \in \{0, 1, 2, \dots\}$ denote the sign pattern *index* at level ℓ, where $0 \leq t \leq 2^\ell - 1$. Then, the binary (i.e., base 2) representation of each

$t = (t_\ell \dots t_2 t_1)_{\text{bin}} := \sum_{k=1}^{\ell} t_k 2^{k-1}$ is in one-to-one correspondence with the binary sign pattern it represents, up to the identification of $\{0,1\}$ with the images $\{-1,1\}$ of the sign operator. For example, at level $\ell = 2$ the sign pattern index $t = 2 = (10)_{\text{bin}}$ corresponds to the sign pattern $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$.

For the t-th sign pattern and g-th class, a *membership index* parameter $r(\ell, i, t, g)$ that uses knowledge of the number of training points in class g having the t-th sign pattern, is calculated for every $\Lambda_{\ell,i}$. Larger values of $r(\ell, i, t, g)$ suggest that the t-th sign pattern is more heavily dominated by class g; thus, if a signal with unknown label corresponds to the t-th sign pattern, we will be more likely to classify it into the g-th class. In this paper, we use the following choice for the membership index parameter $r(\ell, i, t, g)$, which we found to work well experimentally. Below, $P_{g|t} = P_{g|t}(\Lambda_{\ell,i})$ denotes the number of training points from the g-th class with the t-th sign pattern at the i-th set selection in the ℓ-th level:

$$r(\ell, i, t, g) = \frac{P_{g|t}}{\sum_{j=1}^{G} P_{j|t}} \frac{\sum_{j=1}^{G} |P_{g|t} - P_{j|t}|}{\sum_{j=1}^{G} P_{j|t}}. \tag{1}$$

Let us briefly explain the intuition for this formula. The first fraction in (1) indicates the proportion of training points in class g out of all points with sign pattern t (at the ℓ-th level and i-th iteration). The second fraction in (1) is a balancing term that gives more weight to group g when that group is much different in size than the others with the same sign pattern. If $P_{j|t}$ is the same for all classes $j = 1, \dots, G$, then $r(\ell, i, t, g) = 0$ for all g, and thus no class is given extra weight for the given sign pattern, set selection, and level. If $P_{g|t}$ is nonzero and $P_{j|t} = 0$ for all other classes, then $r(\ell, i, t, g) = G - 1$ and $r(\ell, i, t, j) = 0$ for all $j \neq g$, so that class g receives the largest weight. It is certainly possible that a large number of the sign pattern indices t will have $P_{g|t} = 0$ for all groups (i.e., not all binary sign patterns are observed from the training data), in which case $r(\ell, i, t, g) = 0$.

Remark 1 *Note that in practice the membership index value need not be stored for all 2^ℓ possible sign pattern indices, but rather only for the unique sign patterns that are actually observed by the training data. In this case, the unique sign patterns at each level ℓ and iteration i must be input to the classification phase of the algorithm (Algorithm 2).*

Algorithm 1 Training

```
input: training labels $b$, number of classes $G$, number of levels $L$, binary training data $Q$ (or raw training data $X$ and fixed matrix $A$)
if raw data: Compute $Q = \text{sign}(AX)$
for $\ell$ from 1 to $L$, $i$ from 1 to $m$ do
    select: Randomly select $\Lambda_{\ell,i} \subset [m]$, $|\Lambda_{\ell,i}| = \ell$
  for $t$ from 0 to $2^\ell - 1$, $g$ from 1 to $G$ do
      compute: Compute $r(\ell, i, t, g)$ by (1)
  end for
end for
```

Once the algorithm has been trained, we can use it to classify new signals. Suppose $x \in \mathbb{R}^n$ is a new signal for which the class is unknown, and we have available the quantized measurements $q = \text{sign}(Ax)$. Then Algorithm 2 is used for the classification of x into one of the G classes. Notice that the number of levels L, the learned membership index values $r(\ell, i, t, g)$, and the set selections $\Lambda_{\ell,i}$ at each iteration of each level are all available from Algorithm 1. First, the decision vector $\tilde{r}$ is initialized to the zero vector in $\mathbb{R}^G$. Then for each level ℓ and set selection i, the sign pattern, and hence the binary base 2 representation, can be determined using q and $\Lambda_{\ell,i}$. Thus, the corresponding sign pattern index $t^\star = t^\star(\ell, i) \in \{0, 1, 2, \dots\}$ such that $0 \le t^\star \le 2^\ell - 1$ is identified. For each class g, $\tilde{r}(g)$ is updated via $\tilde{r}(g) \leftarrow \tilde{r}(g) + r(\ell, i, t^\star, g)$. Finally, after scaling $\tilde{r}$ with respect to the number of levels and measurements, the largest entry of $\tilde{r}$ identifies how the estimated label $\widehat{b}_x$ of x is set. This scaling of course does not actually affect the outcome of classification, we use it simply to ensure the quantity does not become unbounded for large problem sizes. We note here that especially for large m, the bulk of the classification will come from the higher levels (in fact the last level) due to the geometry of the algorithm. However, we choose to write the testing phase using all levels since the lower levels are cheap to compute with, may still contribute to classification accuracy especially for small m, and can be used naturally in other settings such as hierarchical classification and detection (see remarks in Section 5).

Algorithm 2 Classification

```
input: binary data $q$, number of classes $G$, number of levels $L$, learned parameters
$r(\ell, i, t, g)$ and $\Lambda_{\ell,i}$ from Algorithm 1

initialize: $\tilde{r}(g) = 0$ for $g = 1, \dots, G$.
for $\ell$ from 1 to $L$, $i$ from 1 to $m$ do
    identify: Identify the sign pattern index $t^\star$ using $q$ and $\Lambda_{\ell,i}$
  for $g$ from 1 to $G$ do
      update: $\tilde{r}(g) = \tilde{r}(g) + r(\ell, i, t^\star, g)$
  end for
end for
scale: Set $\tilde{r}(g) = \frac{\tilde{r}(g)}{Lm}$ for $g = 1, \dots, G$
classify: $\widehat{b}_x = \text{argmax}_{g \in \{1, \dots, G\}} \tilde{r}(g)$
```

3. Experimental Results

In this section, we provide experimental results of Algorithms 1 and 2 for synthetically generated datasets, handwritten digit recognition using the MNIST dataset, and facial recognition using the extended YaleB database. We note that for the synthetic data, we typically use Gaussian clouds, but note that since our algorthms use hyperplanes to classify data, the results on these type of datasets would be identical to any with the same radial distribution around the origin. We use Gaussian clouds simply because they are easy to visualize and allow for various geometries. Of course, our methods require no particular structure other than being centered around the origin, which can be done as a pre-processing step (and the framework could clearly be extended to remove this property in future work). The real data like the hand-written digits and faces clearly have more complicated geometries and are harder to visualize. We include both types of data to fully characterize our method's performance.

We also remark here that we purposefully choose not to compare to other related methods like SVM for several reasons. First, if the data happens to be linearly separable it is clear that SVM will outperform or match our approach since it is designed precisely for such data. In the interesting case when the data is not linearly separable, our method will clearly outperform SVM since SVM will fail. To use SVM in this case, one needs an appropriate kernel, and identifying such a kernel is highly non-trivial without understanding the data's geometry, and precisely what our method avoids having to do.

Unless otherwise specified, the matrix A is taken to have i.i.d. standard Gaussian entries. Also, we assume the data is centered. To ensure this, a pre-processing step on the raw data is performed to account for the fact that the data may not be centered around the origin. That is, given the original training data matrix X, we calculate $\mu = \frac{1}{p}\sum_{i=1}^{p} x_i$. Then for each column x_i of X, we set $x_i \leftarrow x_i - \mu$. The testing data is adjusted similarly by μ. Note that this assumption can be overcome in future work by using *dithers*—that is, hyperplane dither values may be learned so that $Q = \text{sign}(AX + \tau)$, where $\tau \in \mathbb{R}^m$—or even with random dithers, as motivated by quantizer results (Baraniuk et al., 2017; Cambareri et al., 2017).

3.1. Classification of Synthetic Datasets

In our first stylized experiment, we consider three classes of Gaussian clouds in $\mathbb{R}^2$ (i.e., $n = 2$); see Figure 1 for an example training and testing data setup. For each choice of $m \in \{5, 7, 9, 11, 13, 15, 17, 19\}$ and $p \in \{75, 150, 225\}$ with equally sized training data sets for each class (that is, each class is tested with either 25, 50, or 75 training points), we execute Algorithms 1 and 2 with a single level and 30 trials of generating A. We perform classification of 50 test points per group, and report the average correct classification rate (ACCR) over all trials. Note that the ACCR is simply defined as the number of correctly classified testing points divided by the total number of testing points (where the correct class is known either from the generated distribution or the real label for real world data), and then averaged over the trials of generating A. We choose this metric since it captures both false negatives and positives, and since in all experiments we have access to the correct labels. The right plot of Figure 1 shows that $m \geq 15$ results in nearly perfect classification.

Next, we present a suite of experiments where we again construct the classes as Gaussian clouds in $\mathbb{R}^2$, but utilize various types of data geometries. In each case, we set the number of training data points for each class to be 25, 50, and 75. In Figure 2, we have two classes forming a total of six Gaussian clouds, and execute Algorithms 1 and 2 using four levels and $m \in \{10, 30, 50, 70, 90, 110, 130\}$. The classification accuracy increases for larger m, with nearly perfect classification for the largest values of m selected. A similar experiment is shown in Figure 3, where we have two classes forming a total of eight Gaussian clouds, and execute the proposed algorithm using five levels.

In the next two experiments, we display the classification results of Algorithms 1 and 2 when using $m \in \{10, 30, 50, 70, 90\}$ and one through four levels, and see that adding levels can be beneficial for more complicated data geometries. In Figure 4, we have three classes forming a total of eight Gaussian clouds. We see that from both $L = 1$ to $L = 2$ and $L = 2$

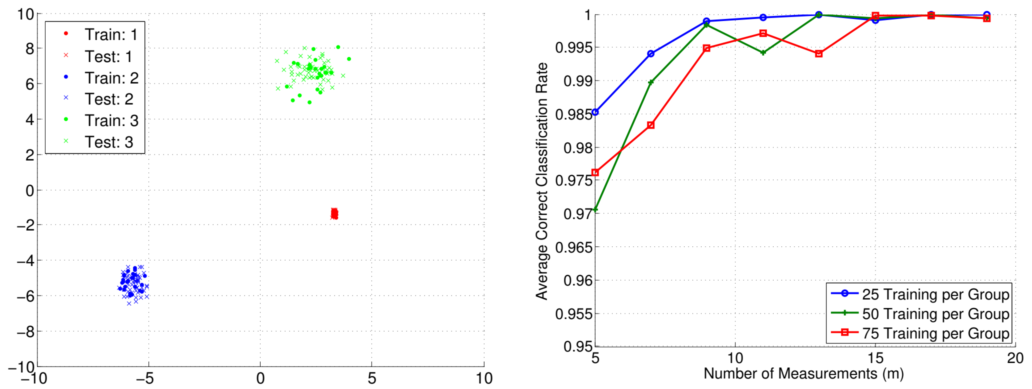


Figure 1: Synthetic classification experiment with three Gaussian clouds ($G = 3$), $L = 1$, $n = 2$, 50 test points per group, and 30 trials of randomly generating A. (Left) Example training and testing data setup. (Right) Average correct classification rate versus m and for the indicated number of training points per class.

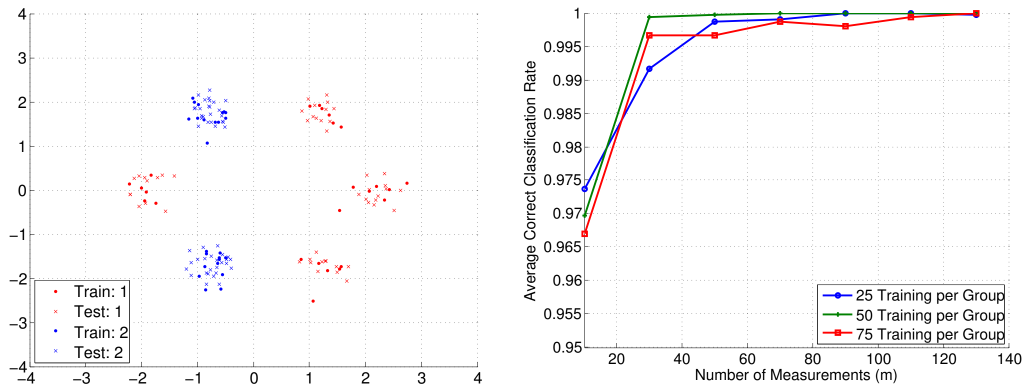


Figure 2: Synthetic classification experiment with six Gaussian clouds and two classes ($G = 2$), $L = 4$, $n = 2$, 50 test points per group, and 30 trials of randomly generating A. (Left) Example training and testing data setup. (Right) Average correct classification rate versus m and for the indicated number of training points per class.

to $L = 3$, there are huge gains in classification accuracy. In Figure 5, we have four classes forming a total of eight Gaussian clouds. Again, from both $L = 1$ to $L = 2$ and $L = 2$ to $L = 3$ we see large improvements in classification accuracy, yet still better classification with $L = 4$. We note here that in this case it also appears that more training data does not improve the performance (and perhaps even slightly decreases accuracy); this is of course unexpected in practice, but we believe this happens here only because of the construction of the Gaussian clouds—more training data leads to more outliers in each cloud, making the sets harder to separate.

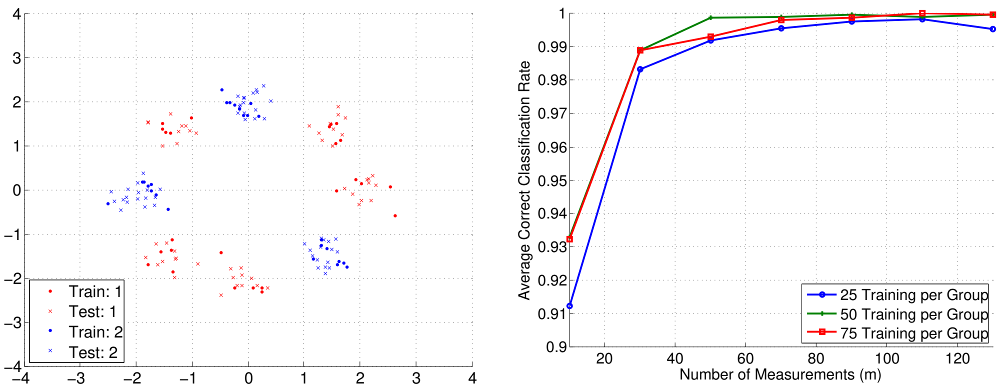
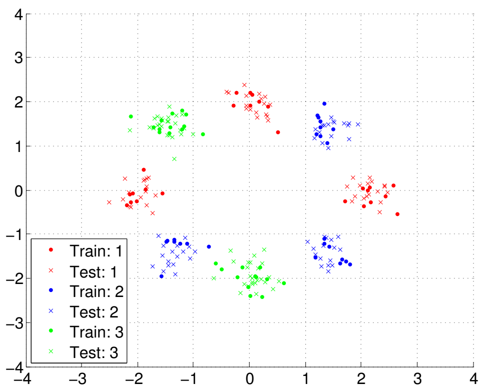


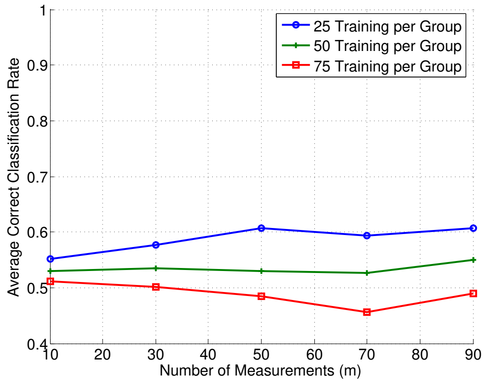
Figure 3: Synthetic classification experiment with eight Gaussian clouds and two classes ($G = 2$), $L = 5$, $n = 2$, 50 test points per group, and 30 trials of randomly generating A. (Left) Example training and testing data setup. (Right) Average correct classification rate versus m and for the indicated number of training points per class.

3.2. Handwritten Digit Classification

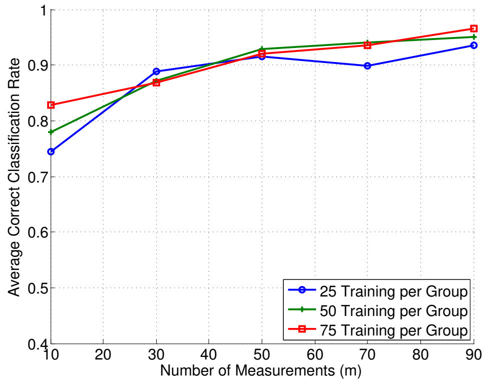
In this section, we apply Algorithms 1 and 2 to the MNIST (LeCun, 2018) dataset, which is a benchmark dataset of images of handwritten digits, each with 28×28 pixels. In total, the dataset has $60,000$ training examples and $10,000$ testing examples.

First, we apply Algorithms 1 and 2 when considering only two digit classes. Figure 6 shows the correct classification rate for the digits "0" versus "1". We set $m \in \{10, 30, 50, 70, 90, 110\}$, $p \in \{50, 100, 150\}$ with equally sized training data sets for each class, and classify 50 images per digit class. Notice that the algorithm is performing very well for small m in comparison to $n = 28 \times 28 = 784$ and only a single level. Figure 7 shows the results of a similar setup for the digits "0" and "5". In this experiment, we increased to four levels and achieve classification accuracy around 90% at the high end of m values tested. This indicates that the digits "0" and "5" are more likely to be mixed up than "0" and "1", which is understandable due to the more similar digit shape between "0" and "5". In Figure 7, we include the classification performance when the matrix A is constructed using the two-dimensional Discrete Cosine Transform (DCT) in addition to our typical Gaussian matrix A (note one could similarly use the Discrete Fourier Transform instead of the DCT but that requires re-defining the sign function on complex values). Specifically, to construct A from the $n \times n$ two-dimensional DCT, we select m rows uniformly at random

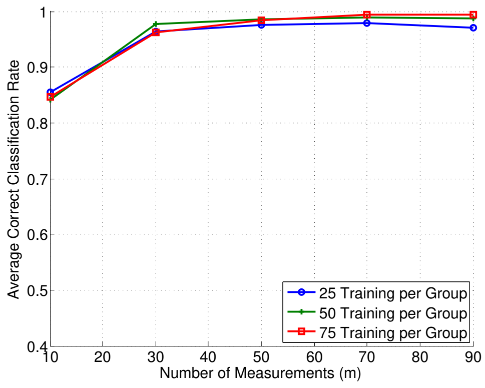




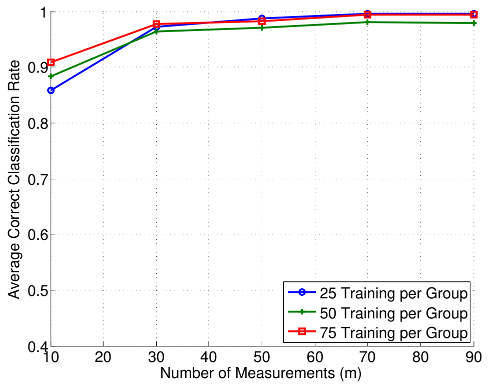
(a) $L = 1$



(b) $L = 2$

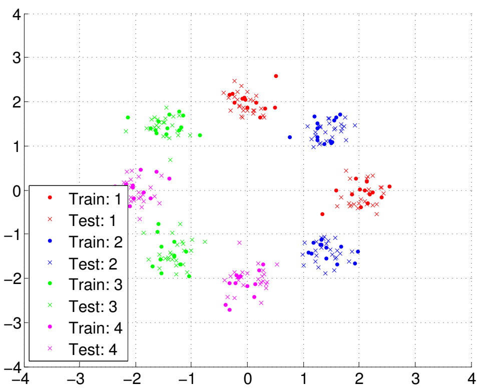


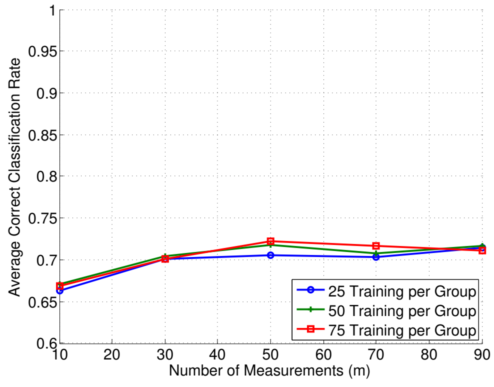
(c) $L = 3$



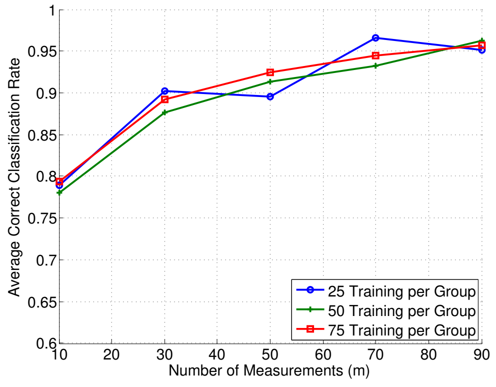
(d) $L = 4$

Figure 4: Synthetic classification experiment with eight Gaussian clouds and three classes ($G = 3$), $L = 1, \dots, 4$, $n = 2$, 50 test points per group, and 30 trials of randomly generating A. (Top) Example training and testing data setup. Average correct classification rate versus m and for the indicated number of training points per class for: (middle left) $L = 1$, (middle right) $L = 2$, (bottom left) $L = 3$, (bottom right) $L = 4$.

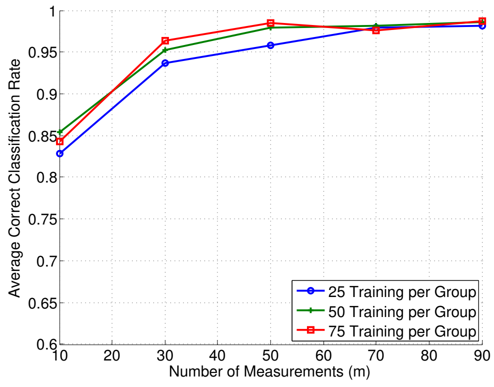




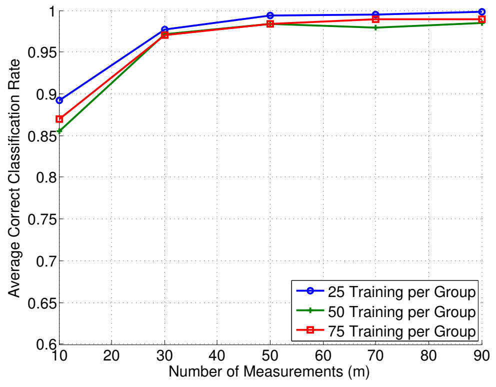
(a) $L = 1$



(b) $L = 2$



(c) $L = 3$



(d) $L = 4$

Figure 5: Synthetic classification experiment with eight Gaussian clouds and four classes ($G = 4$), $L = 1, \ldots, 4$, $n = 2$, 50 test points per group, and 30 trials of randomly generating A. (Top) Example training and testing data setup. Average correct classification rate versus m and for the indicated number of training points per class for: (middle left) $L = 1$, (middle right) $L = 2$, (bottom left) $L = 3$, (bottom right) $L = 4$.

and then apply a random sign (i.e., multiply by +1 or -1) to the columns. We include these two results to illustrate that there is not much difference when using the DCT and Gaussian constructions of A, though we expect analyzing the DCT case to be more challenging and limit the theoretical analysis in this paper to the Gaussian setting. The advantage of using a structured matrix like the DCT is of course the reduction in computation cost in acquiring the measurements.

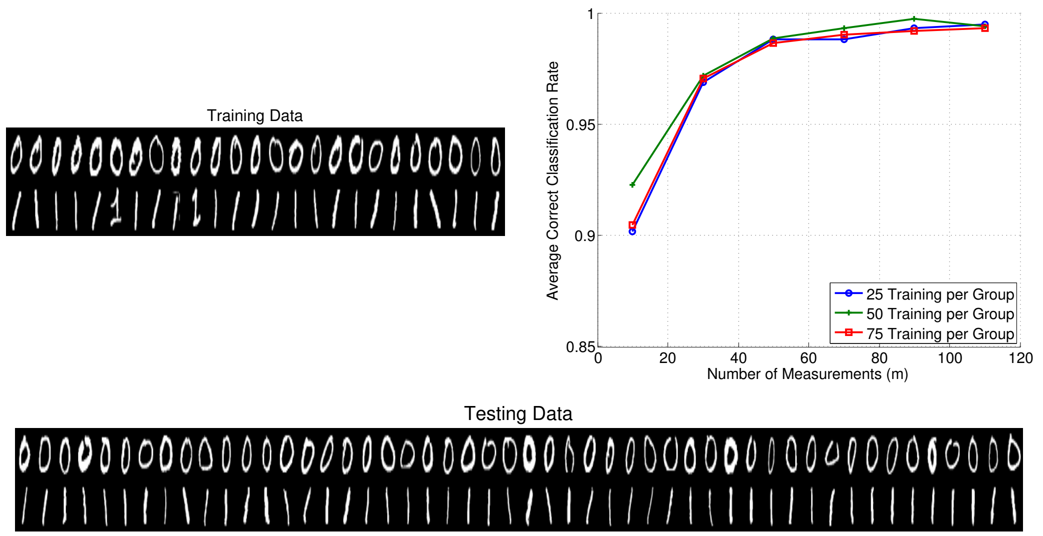


Figure 6: Classification experiment using the handwritten “0” and “1” digit images from the MNIST dataset, $L = 1$, $n = 28 \times 28 = 784$, 50 test points per group, and 30 trials of randomly generating A. (Top left) Training data images when $p = 50$. (Top right) Average correct classification rate versus m and for the indicated number of training points per class. (Bottom) Testing data images.

Next, we apply Algorithms 1 and 2 to the MNIST dataset with all ten digits. We utilize $1,000$, $3,000$, and $5,000$ training points per digit class, and perform classification with 800 test images per class. The classification results using 18 levels and $m \in \{100, 200, 400, 600, 800\}$ are shown in Figure 8, where it can be seen that with $5,000$ training points per class, above 90% classification accuracy is achieved for $m \geq 200$. We also see that larger training sets result in slightly improved classification.

3.3. Facial Recognition

Our last experiment considers facial recognition using the extended YaleB dataset (Cai et al., 2007b,a, 2006; He et al., 2005). This dataset includes 32×32 images of 38 individuals with roughly 64 near-frontal images under different illuminations per individual. We select four individuals from the dataset, and randomly select images with different illuminations to be included in the training and testing sets (note that the same illumination was included for *each* individual in the training and testing data). We execute Algorithms 1 and 2 using four levels with $m \in \{10, 50, 100, 150, 200, 250, 300\}$, $p \in \{20, 40, 60\}$ with equally sized training data sets for each class, and classify 30 images per class. The results are displayed

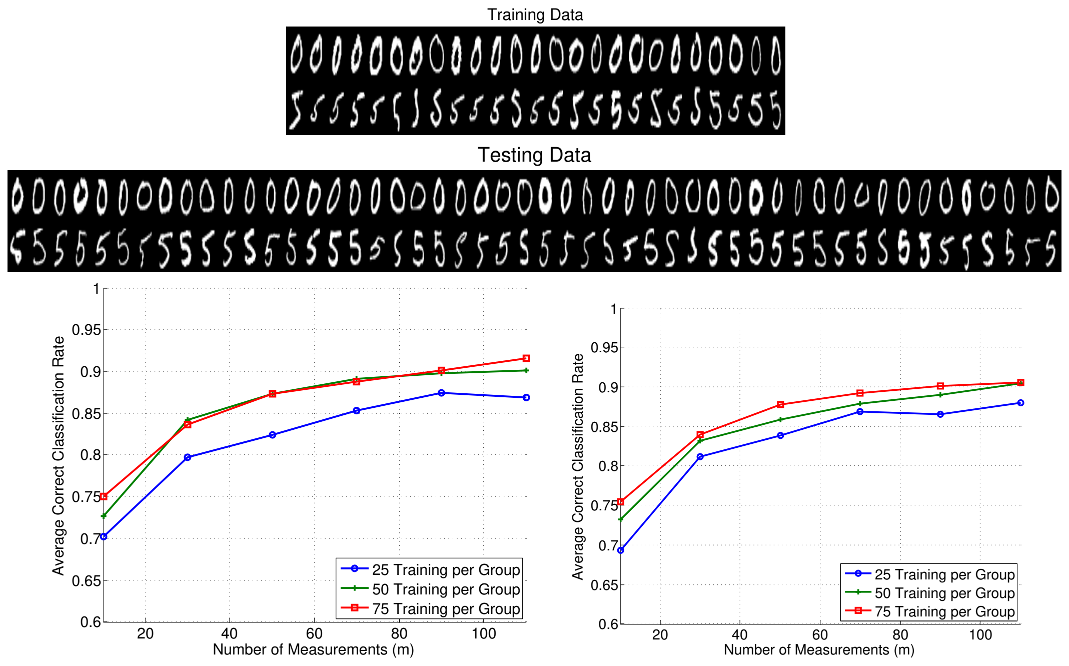


Figure 7: Classification experiment using the handwritten “0” and “5” digit images from the MNIST dataset, $L = 4$, $n = 28 \times 28 = 784$, 50 test points per group, and 30 trials of randomly generating A. (Top) Training data images when $p = 50$. (Middle) Testing data images. Average correct classification rate versus m and for the indicated number of training points per class (bottom left) when using a Gaussian matrix A and (bottom right) when using a DCT matrix A.

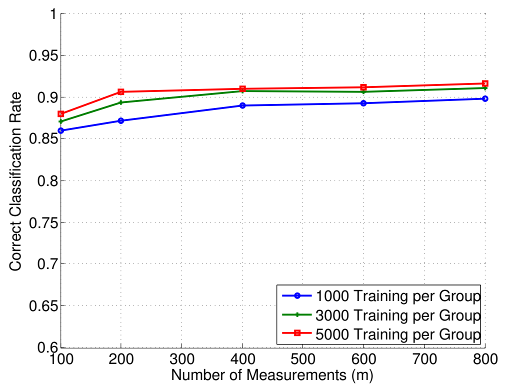


Figure 8: Correct classification rate versus m when using all ten (0-9) handwritten digits from the MNIST dataset, $L = 18$, $n = 28 \times 28 = 784$, 1,000, 3,000, and 5,000 training points per group, 800 test points per group (8,000 total), and a single instance of randomly generating A.

in Figure 9. Above 90% correct classification is achieved for $m \geq 150$ when using the largest training set.

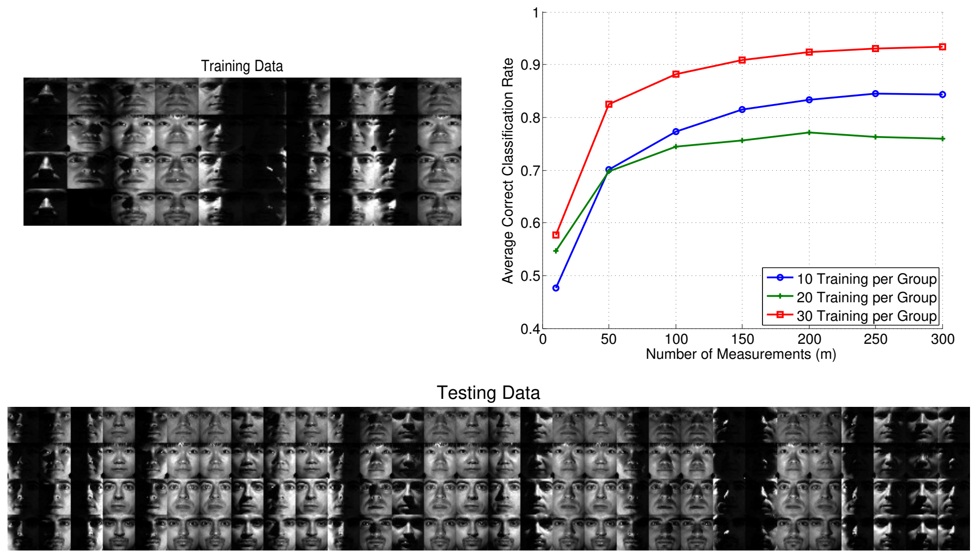


Figure 9: Classification experiment using four individuals from the extended YaleB dataset, $L = 4$, $n = 32 \times 32 = 1024$, 30 test points per group, and 30 trials of randomly generating A. (Top left) Training data images when $p = 20$. (Top right) Average correct classification rate versus m and for the indicated number of training points per class. (Bottom) Testing data images.

4. Theoretical Analysis for a Simple Case

4.1. Main Results

We now provide a theoretical analysis of Algorithms 1 and 2 in which we make a series of simplifying assumptions to make the development more tractable. We focus on the setting where the signals are two-dimensional, belonging to one of two classes, and consider a single level (i.e., $\ell = 1$, $n = 2$, and $G = 2$). Moreover, we assume the true classes G_1 and G_2 to be two disjoint *cones* in $\mathbb{R}^2$ and assume that regions of the same angular measure have the same number (or density) of training points. Of course, the problem of non-uniform densities relates to complicated geometries that may dictate the number of training points required for accurate classification (especially when many levels are needed) and is a great direction for future work. However, we believe analyzing this simpler setup will provide a foundation for a more generalized analysis in future work.

Let A_1 denote the angular measure of G_1, defined by

$$A_1 = \max_{x_1, x_2 \in G_1} \angle(x_1, x_2),$$

where $\angle(x_1, x_2)$ denotes the angle between the vectors x_1 and x_2; define A_2 similarly for G_2. Also, define

$$A_{12} = \min_{x_1 \in G_1, x_2 \in G_2} \angle(x_1, x_2)$$

as the angle between classes G_1 and G_2. Suppose that the test point $x \in G_1$, and that we classify x using m random hyperplanes. For simplicity, we assume that the hyperplanes can intersect the cones, but only intersect *one* cone at a time. This means we are imposing the condition $A_{12} + A_1 + A_2 \leq \pi$. See Figure 10 for a visualization of the setup for the analysis. Notice that A_1 is partitioned into two disjoint pieces, θ_1 and θ_2, where $A_1 = \theta_1 + \theta_2$. The angles θ_1 and θ_2 are determined by the location of x within G_1.

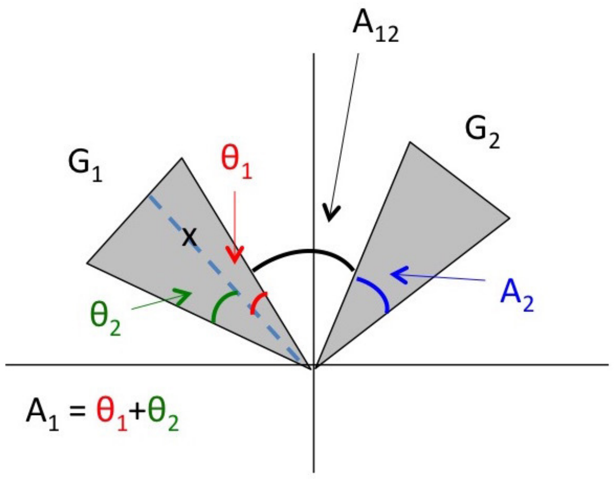


Figure 10: Visualization of the analysis setup for two classes of two dimensions. If a hyperplane intersects the θ_1 region of G_1, then x is not on the same side of the hyperplane as G_2. If a hyperplane intersects the θ_2 region of G_1, then x is on the same side of the hyperplane as G_2. That is, θ_1 and θ_2 are determined by the position of x within G_1, and $\theta_1 + \theta_2 = A_1$.

The membership index parameter (1) is still used; however, now we have angles instead of numbers of training points. That is,

$$r(\ell, i, t, g) = \frac{A_{g|t}}{\sum_{j=1}^{G} A_{j|t}} \frac{\sum_{j=1}^{G} |A_{g|t} - A_{j|t}|}{\sum_{j=1}^{G} A_{j|t}}, \tag{2}$$

where $A_{g|t} = A_{g|t}(\Lambda_{\ell,i})$ denotes the angle of the part of class g with the t-th sign pattern index at the i-th set selection in the ℓ-th level. Throughout, let $t_i^\star$ denote the sign pattern index of the test point x with the i-th hyperplane at the first level, $\ell = 1$; i.e. $t_i^\star = t^\star_{\Lambda_{\ell,i}}$ with the identification $\Lambda_{\ell,i} = \{i\}$ (since $\ell = 1$ implies a single hyperplane is used). Letting $\widehat{b}_x$ denote the classification label for x after running the proposed algorithm, Theorem 2 describes the probability that x gets classified correctly with $\widehat{b}_x = 1$. Note that for simplicity, in Theorem 2 we assume the classes G_1 and G_2 are of the same size (i.e., $A_1 = A_2$) and the test point x lies in the middle of class G_1 (i.e., $\theta_1 = \theta_2$). These assumptions are for convenience and clarity of presentation only (note that (3) is already quite cumbersome), but the proof follows analogously (albeit without easy simplifications) for the general case; for convenience we leave the computations in Table 1 in general form and do not utilize the assumption $\theta_1 = \theta_2$ until the end of the proof. We first state a technical result in Theorem 2, and include two corollaries below that illustrate its usefulness.

Theorem 2 *Let the classes G_1 and G_2 be two cones in $\mathbb{R}^2$ defined by angular measures A_1 and A_2, respectively, and suppose regions of the same angular measure have the same density of training points. Suppose $A_1 = A_2$, $\theta_1 = \theta_2$, and $A_{12} + A_1 + A_2 \leq \pi$. Then, the probability that a data point $x \in G_1$ gets classified in class G_1 by Algorithms 1 and 2 using a single level and a measurement matrix $A \in \mathbb{R}^{m\times 2}$ with independent standard Gaussian entries is bounded as follows,*

$$\mathbb{P}[\widehat{b}_x = 1] \geq 1 - \sum_{\substack{j=0 \\ j+k_{1,1}+k_{1,2}+k_2+k=m,\ k_{1,2}\geq 9(j+k_{1,1})}}^{m} \sum_{k_{1,1}=0}^{m} \sum_{k_{1,2}=0}^{m} \sum_{k_2=0}^{m} \sum_{k=0}^{m} \binom{m}{j, k_{1,1}, k_{1,2}, k_2, k} \left(\frac{A_{12}}{\pi}\right)^{j} \left(\frac{A_1}{2\pi}\right)^{k_{1,1}+k_{1,2}} \times \left(\frac{A_1}{\pi}\right)^{k_2} \left(\frac{\pi - 2A_1 - A_{12}}{\pi}\right)^{k}. \tag{3}$$

Figure 11 displays the classification probability bound of Theorem 2 compared to the (simulated) true value of $\mathbb{P}[\widehat{b}_x = 1]$. Here, $A_1 = A_2 = 15°$, $\theta_1 = \theta_2 = 7.5°$, and A_{12} and m are varied. Most importantly, notice that in all cases, the classification probability is approaching 1 with increasing m. Also, the result from Theorem 2 behaves similarly as the simulated true probability, especially as m and A_{12} increase.

The following two corollaries provide asymptotic results for situations where $\mathbb{P}[\widehat{b}_x = 1]$ tends to 1 when $m \to \infty$. Corollary 3 provides this result whenever A_{12} is at least as large as both A_1 and $\pi - 2A_1 - A_{12}$, and Corollary 4 provides this result for certain combinations of A_1 and A_{12}. These results of course should match intuition, since as m grows large, our hyperplanes essentially chop up the space into finer and finer wedges. Below, the dependence on the constants on A_1, A_{12} is explicit in the proofs.

Corollary 3 *Consider the setup of Theorem 2. Suppose $A_{12} \geq A_1$ and $2A_{12} \geq \pi - 2A_1$. Then $\mathbb{P}[\widehat{b}_x = 1] \to 1$ as $m \to \infty$. In fact, the probability converges to 1 exponentially, i.e. $\mathbb{P}[\hat{b}_x = 1] \geq 1 - Ce^{-cm}$ for positive constants c and C that may depend on A_1, A_{12}.*

Corollary 4 *Consider the setup of Theorem 2. Suppose $A_1 + A_{12} > 0.58\pi$ and $A_{12} + \frac{3}{4}A_1 \leq \frac{\pi}{2}$. Then $\mathbb{P}[\widehat{b}_x = 1] \to 1$ as $m \to \infty$. In fact, the probability converges to 1 exponentially, i.e. $\mathbb{P}[\hat{b}_x = 1] \geq 1 - Ce^{-cm}$ for positive constants c and C that may depend on A_1, A_{12}.*

4.2. Proof of Main Results

4.2.1. Proof of Theorem 2

Proof Using our setup, we have five possibilities for any given hyperplane: (i) the hyperplane completely separates the two classes, i.e., the cones associated with the two classes

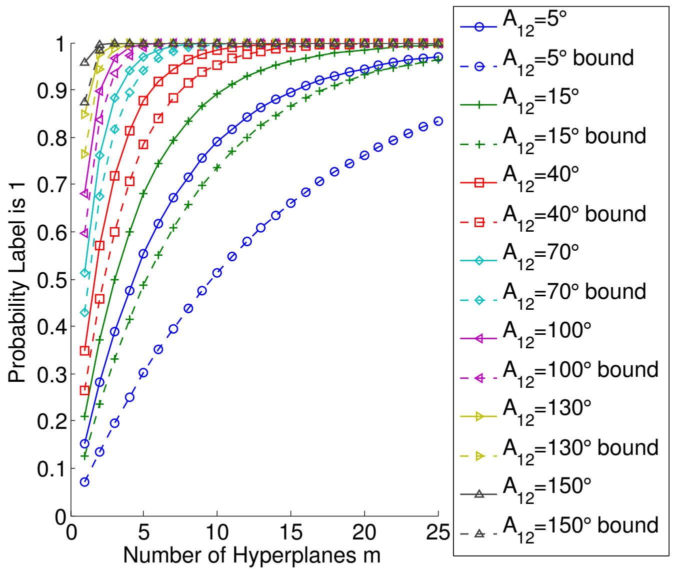


Figure 11: $\mathbb{P}[\widehat{b}_x = 1]$ versus the number of hyperplanes m when A_{12} is varied (see legend), $A_1 = A_2 = 15°$, and $\theta_1 = \theta_2 = 7.5°$. The solid lines indicate the probability (5) with the multinomial probability given by (6) and the conditional probability (9) simulated over 1000 trials of the uniform random variables. The dashed lines indicate the result (3) provided in Theorem 2.

fall on either side of the hyperplane, (ii) the hyperplane completely does not separate the two classes, i.e., the cones fall on the same side of the hyperplane, (iii) the hyperplane cuts through G_2, (iv) the hyperplane cuts through G_1 via θ_1, or (v) the hyperplane cuts through G_1 via θ_2. Using this observation, we can now define the event

$$E(j, k_{1,1}, k_{1,2}, k_2) \tag{4}$$

whereby from among the m total hyperplanes, j hyperplanes separate the cones, $k_{1,1}$ hyperplanes cut G_1 in θ_1, $k_{1,2}$ hyperplanes cut G_1 in θ_2, and k_2 hyperplanes cut G_2. See Table 1 for an easy reference of these quantities. Note that we must distinguish between hyperplanes that cut through θ_1 and those that cut through θ_2; $k_{1,1}$ hyperplanes cut G_1 and land within θ_1 so that x is *not* on the same side of the hyperplane as G_2 whereas $k_{1,2}$ hyperplanes cut G_1 and land within θ_2 so that x *is* on the same side of the hyperplane as G_2. These orientations will affect the computation of the membership index. Using the above definition of (4), we use the law of total probability to get a handle on $\mathbb{P}[\widehat{b}_x = 1]$, the probability that the test point x gets classified correctly, as follows,

$$\begin{aligned}\mathbb{P}[\widehat{b}_x = 1] &= \mathbb{P}\left[\sum_{i=1}^{m} r(\ell, i, t_i^\star, 1) > \sum_{i=1}^{m} r(\ell, i, t_i^\star, 2)\right] \\ &= \sum_{\substack{j,k_{1,1},k_{1,2},k_2 \\ j+k_{1,1}+k_{1,2}+k_2 \le m}} \mathbb{P}\left[\sum_{i=1}^{m} r(\ell, i, t_i^\star, 1) > \sum_{i=1}^{m} r(\ell, i, t_i^\star, 2)\ |E(j, k_{1,1}, k_{1,2}, k_2)\right]\end{aligned}$$

$$\times \mathbb{P}\left[E(j, k_{1,1}, k_{1,2}, k_2)\right]. \tag{5}$$

The latter probability in (5) is similar to the probability density of a multinomial random variable:

$$\begin{aligned}
&\mathbb{P}\left[E(j, k_{1,1}, k_{1,2}, k_2)\right] \\
&= \binom{m}{j, k_{1,1}, k_{1,2}, k_2, m-j-k_{1,1}-k_{1,2}-k_2} \left(\frac{A_{12}}{\pi}\right)^{j} \left(\frac{\theta_1}{\pi}\right)^{k_{1,1}} \left(\frac{\theta_2}{\pi}\right)^{k_{1,2}} \\
&\quad \times \left(\frac{A_2}{\pi}\right)^{k_2} \left(\frac{\pi - A_1 - A_2 - A_{12}}{\pi}\right)^{m-j-k_{1,1}-k_{1,2}-k_2}, \qquad (6)
\end{aligned}$$

where $\binom{n}{k_1, k_2, \ldots, k_m} = \frac{n!}{k_1! k_2! \cdots k_m!}$.

To evaluate the conditional probability in (5), we must determine the value of $r(\ell, i, t_i^\star, g)$, for $g = 1, 2$, given the hyperplane cutting pattern event. Table 1 summarizes the possible cases. In the cases where the hyperplane cuts through either G_1 or G_2, we model the location of the hyperplane within the class by a random variable defined on the interval $[0, 1]$, with no assumed distribution. We let u, u', u_h, $u'_h \in [0, 1]$ (for an index h) denote independent copies of such random variables.

Hyperplane Case	Number in event (4)	Class g	Value of $r(\ell, i, t_i^\star, g)$ (see (2))
(i) separates	j	1	1
		2	0
(ii) does not separate	$m - j - k_2 - k_{1,1} - k_{1,2}$	1	$\frac{A_1\lvert A_1 - A_2\rvert}{(A_1+A_2)^2}$
		2	$\frac{A_2\lvert A_1 - A_2\rvert}{(A_1+A_2)^2}$
(iii) cuts G_2	k_2	1	$\frac{A_1\lvert A_1 - A_2u'\rvert}{(A_1+A_2u')^2}$
		2	$\frac{A_2u'\lvert A_1 - A_2u'\rvert}{(A_1+A_2u')^2}$
(iv) cuts G_1, θ_1	$k_{1,1}$	1	1
		2	0
(v) cuts G_1, θ_2	$k_{1,2}$	1	$\frac{(\theta_1+\theta_2u)\lvert\theta_1+\theta_2u - A_2\rvert}{(\theta_1+\theta_2u+A_2)^2}$
		2	$\frac{A_2\lvert\theta_1+\theta_2u - A_2\rvert}{(\theta_1+\theta_2u+A_2)^2}$

Table 1: Summary of (2) when up to one cone can be cut per hyperplane, where u, u' are independent random variables defined over the interval $[0, 1]$.

Using the computations given in Table 1 and assuming j hyperplanes separate (i.e. condition (i) described above), $k_{1,1}$ hyperplanes cut G_1 in θ_1 (condition (iv) above), $k_{1,2}$ hyperplanes cut G_1 in θ_2 (condition (v) above), k_2 hyperplanes cut G_2 (condition (iii) above), and $m - j - k_{1,1} - k_{1,2} - k_2$ hyperplanes do not separate (condition (ii) above), we compute the membership index parameters defined in (2) as:

$$\sum_{i=1}^{m} r(\ell, i, t_i^\star, 1) = j + (m - j - k_{1,1} - k_{1,2} - k_2)\frac{A_1\lvert A_1 - A_2\rvert}{(A_1 + A_2)^2} + k_{1,1}$$

$$
\begin{aligned}
&+\sum_{h=1}^{k_{1,2}}\frac{(\theta_1+\theta_2u_h)|\theta_1+\theta_2u_h-A_2|}{(\theta_1+\theta_2u_h+A_2)^2}+\sum_{h=1}^{k_2}\frac{A_1|A_1-A_2u_h'|}{(A_1+A_2u_h')^2}\\
&=j+k_{1,1}+\sum_{h=1}^{k_{1,2}}\frac{(\theta_1+\theta_2u_h)|\theta_1+\theta_2u_h-A_1|}{(\theta_1+\theta_2u_h+A_1)^2}+\sum_{h=1}^{k_2}\frac{A_1|A_1-A_1u_h'|}{(A_1+A_1u_h')^2} \qquad (7)
\end{aligned}
$$

and

$$
\begin{aligned}
\sum_{i=1}^{m}r(\ell,i,t_i^\star,2)&=(m-j-k_{1,1}-k_{1,2}-k_2)\frac{A_2|A_1-A_2|}{(A_1+A_2)^2}\\
&\quad+\sum_{h=1}^{k_{1,2}}\frac{A_2|\theta_1+\theta_2u_h-A_2|}{(\theta_1+\theta_2u_h+A_2)^2}+\sum_{h=1}^{k_2}\frac{A_2u_h'|A_1-A_2u_h'|}{(A_1+A_2u_h')^2}\\
&=\sum_{h=1}^{k_{1,2}}\frac{A_1|\theta_1+\theta_2u_h-A_1|}{(\theta_1+\theta_2u_h+A_1)^2}+\sum_{h=1}^{k_2}\frac{A_1u_h'|A_1-A_1u_h'|}{(A_1+A_1u_h')^2}, \qquad (8)
\end{aligned}
$$

where in both cases we have simplified using the assumption $A_1 = A_2$. Thus, the conditional probability in (5), can be expressed as:

$$
\mathbb{P}\left[j+k_{1,1}+\sum_{h=1}^{k_{1,2}}\frac{|\theta_1+\theta_2u_h-A_1|(\theta_1+\theta_2u_h-A_1)}{(\theta_1+\theta_2u_h+A_1)^2}+\sum_{h=1}^{k_2}\frac{|A_1-A_1u_h'|(A_1-A_1u_h')}{(A_1+A_1u_h')^2}>0\right], \qquad (9)
$$

where it is implied that this probably is conditioned on the hyperplane configuration as in (5). Once the probability (9) is known, we can calculate the full classification probability (5).

Since by assumption, $\theta_1+\theta_2 = A_1$, we have $\theta_1+\theta_2u-A_1 \le 0$ and $A_1-A_1u' \ge 0$. Thus, (9) simplifies to

$$
\mathbb{P}\left[j+k_{1,1}-\sum_{h=1}^{k_{1,2}}\frac{(\theta_1+\theta_2u_h-A_1)^2}{(\theta_1+\theta_2u_h+A_1)^2}+\sum_{h=1}^{k_2}\frac{(A_1-A_1u_h')^2}{(A_1+A_1u_h')^2}>0\right] \;\ge\; \mathbb{P}[\gamma>\beta] \qquad (10)
$$

where

$$
\beta = k_{1,2}\left(\frac{\theta_2}{A_1+\theta_1}\right)^2 \quad \text{and} \quad \gamma = j+k_{1,1}.
$$

To obtain the inequality in (10), we used the fact that

$$
j+k_{1,1}-\sum_{h=1}^{k_{1,2}}\frac{(\theta_1-A_1)^2}{(\theta_1+\theta_2u_h+A_1)^2}+\sum_{h=1}^{k_2}\frac{(A_1-A_1u_h')^2}{(A_1+A_1u_h')^2}\ge j+k_{1,1}-\sum_{h=1}^{k_{1,2}}\frac{(\theta_1-A_1)^2}{(\theta_1+A_1)^2}+0=\gamma-\beta.
$$

By conditioning on $\gamma > \beta$, the probability of interest (5) reduces to (note the bounds on the summation indices):

$$
\mathbb{P}[\widehat{b}_x=1] = \sum_{\substack{j,k_{1,1},k_{1,2},k_2\\ j+k_{1,1}+k_{1,2}+k_2\le m}} \mathbb{P}\left[\sum_{i=1}^{m}r(\ell,i,t_i^\star,1)>\sum_{i=1}^{m}r(\ell,i,t_i^\star,2)\;|E(j,k_{1,1},k_{1,2},k_2)\right]
$$

$$\times\, \mathbb{P}\left[E(j, k_{1,1}, k_{1,2}, k_2)\right] \tag{11}$$

$$\geq \sum_{\substack{j,k_{1,1},k_{1,2},k_2 \\ j+k_{1,1}+k_{1,2}+k_2 \leq m, \\ \beta-\gamma<0}} \binom{m}{j, k_{1,1}, k_{1,2}, k_2, m-j-k_{1,1}-k_{1,2}-k_2} \left(\frac{A_{12}}{\pi}\right)^j \left(\frac{\theta_1}{\pi}\right)^{k_{1,1}}$$

$$\times \left(\frac{\theta_2}{\pi}\right)^{k_{1,2}} \left(\frac{A_2}{\pi}\right)^{k_2} \left(\frac{\pi - A_1 - A_2 - A_{12}}{\pi}\right)^{m-j-k_{1,1}-k_{1,2}-k_2}. \tag{12}$$

The condition $\beta - \gamma < 0$ is equivalent to $k_{1,2}(\frac{\theta_2}{A_1+\theta_1})^2 - (j + k_{1,1}) < 0$, which implies $k_{1,2}(\frac{\theta_2}{A_1+\theta_1})^2 < j + k_{1,1}$. Assuming $\theta_1 = \theta_2$ simplifies this condition to depend *only* on the hyperplane configuration (and not A_1, θ_1, and θ_2) since $\frac{\theta_2}{A_1+\theta_1} = \frac{\theta_2}{3\theta_2} = \frac{1}{3}$. Thus, the condition $\beta - \gamma < 0$ reduces to the condition $k_{1,2} < 9(j + k_{1,1})$ and (12) then simplifies to

$$\sum_{\substack{j+k_{1,1}+k_{1,2}+k_2 \leq m, \\ k_{1,2}<9(j+k_{1,1})}} \binom{m}{j, k_{1,1}, k_{1,2}, k_2, m-j-k_{1,1}-k_{1,2}-k_2} \left(\frac{A_{12}}{\pi}\right)^j \left(\frac{\theta_1}{\pi}\right)^{k_{1,1}+k_{1,2}}$$

$$\times \left(\frac{A_2}{\pi}\right)^{k_2} \left(\frac{\pi - 2A_1 - A_{12}}{\pi}\right)^{m-j-k_{1,1}-k_{1,2}-k_2} \tag{13}$$

$$= \sum_{\substack{j+k_{1,1}+k_{1,2}+k_2+k=m, \\ k_{1,2}<9(j+k_{1,1})}} \binom{m}{j, k_{1,1}, k_{1,2}, k_2, k} \left(\frac{A_{12}}{\pi}\right)^j \left(\frac{\theta_1}{\pi}\right)^{k_{1,1}+k_{1,2}} \left(\frac{A_2}{\pi}\right)^{k_2} \left(\frac{\pi - 2A_1 - A_{12}}{\pi}\right)^k, \tag{14}$$

$$= \sum_{\substack{j+k_{1,1}+k_{1,2}+k_2+k=m, \\ k_{1,2}<9(j+k_{1,1})}} \binom{m}{j, k_{1,1}, k_{1,2}, k_2, k} \left(\frac{A_{12}}{\pi}\right)^j \left(\frac{A_1}{2\pi}\right)^{k_{1,1}+k_{1,2}} \left(\frac{A_1}{\pi}\right)^{k_2} \left(\frac{\pi - 2A_1 - A_{12}}{\pi}\right)^k, \tag{15}$$

where we have introduced k to denote the number of hyperplanes that do not separate nor cut through either of the groups, and simplified using the assumptions that $\theta_1 = \frac{A_1}{2}$ and $A_1 = A_2$.

Note that if we did not have the condition $k_{1,2} < 9(j + k_{1,1})$ in the sum (15) (that is, if we summed over all terms), the quantity would sum to 1 (this can easily be seen by the Multinomial Theorem). Finally, this means (15) is equivalent to (3), thereby completing the proof. ■

4.2.2. Proof of Corollary 3

Proof We can bound (3) from below by bounding the excluded terms in the sum (i.e., those that satisfy $k_{1,2} \geq 9(j + k_{1,1})$) from above. One approach to this would be to count the number of terms satisfying $k_{1,2} \geq 9(j + k_{1,1})$ and bound them by their maximum. Using basic combinatorics (see the appendix, Section A.1), that the number of terms satisfying

$k_{1,2} \geq 9(j + k_{1,1})$ is given by

$$W_1 = \frac{1}{12}\left(\left\lfloor\frac{m}{10}\right\rfloor + 1\right)\left(\left\lfloor\frac{m}{10}\right\rfloor + 2\right)\left(150\left\lfloor\frac{m}{10}\right\rfloor^2 - 10(4m+1)\left\lfloor\frac{m}{10}\right\rfloor + 3(m^2+3m+2)\right) \sim m^4. \tag{16}$$

Then, the quantity (3) can be bounded below by

$$\begin{gathered}1 - W_1 \max\left(\binom{m}{j, k_{1,1}, k_{1,2}, k_2, k}\left(\frac{A_{12}}{\pi}\right)^j\left(\frac{A_1}{2\pi}\right)^{k_{1,1}+k_{1,2}}\left(\frac{A_1}{\pi}\right)^{k_2}\left(\frac{\pi - 2A_1 - A_{12}}{\pi}\right)^k\right) = \\ 1 - W_1 \max\left(\binom{m}{j, k_{1,1}, k_{1,2}, k_2, k}\left(\frac{1}{2}\right)^{k_{1,1}+k_{1,2}}\left(\frac{A_{12}}{\pi}\right)^j\left(\frac{A_1}{\pi}\right)^{k_{1,1}+k_{1,2}+k_2}\left(\frac{\pi - 2A_1 - A_{12}}{\pi}\right)^k\right),\end{gathered} \tag{17}$$

where the maximum is taken over all $j, k_{1,1}, k_{1,2}, k_2, k = 0, \ldots, m$ such that $k_{1,2} \geq 9(j+k_{1,1})$. Ignoring the constraint $k_{1,2} \geq 9(j + k_{1,1})$, we can upper bound the multinomial coefficient using the trivial upper bound of 5^m:

$$\binom{m}{j, k_{1,1}, k_{1,2}, k_2, k} \leq 5^m. \tag{18}$$

Since we are assuming A_{12} is larger than A_1 and $\pi - 2A_1 - A_{12}$ (from the assumption that $2A_{12} \geq \pi - 2A_1$), the strategy is to take j to be as large as possible while satisfying $k_{1,2} \geq 9j$ and $j + k_{1,2} = m$. Since $k_{1,2} \geq 9j$, we have $j + 9j \leq m$ which implies $j \leq \frac{m}{10}$. So, we take $j = \frac{m}{10}$, $k_{1,2} = \frac{9m}{10}$, and $k_{1,1} = k_2 = k = 0$. Then

$$\left(\frac{1}{2}\right)^{k_{1,1}+k_{1,2}}\left(\frac{A_{12}}{\pi}\right)^j\left(\frac{A_1}{\pi}\right)^{k_{1,1}+k_{1,2}+k_2}\left(\frac{\pi - 2A_1 - A_{12}}{\pi}\right)^k \tag{19}$$

$$\begin{aligned}&\leq \left(\frac{1}{2}\right)^{9m/10}\left(\frac{A_{12}}{\pi}\right)^{m/10}\left(\frac{A_1}{\pi}\right)^{9m/10} \\ &= \left(\frac{1}{2^9}\frac{A_{12}}{\pi}\left(\frac{A_1}{\pi}\right)^9\right)^{m/10}.\end{aligned} \tag{20}$$

Combining (17) with the bounds given in (18) and (20), we have

$$\begin{aligned}&\geq 1 - W_1 5^m\left(\frac{1}{2^9}\frac{A_{12}}{\pi}\left(\frac{A_1}{\pi}\right)^9\right)^{m/10} \\ &\sim 1 - m^4 5^m\left(\frac{1}{2^9}\frac{A_{12}}{\pi}\left(\frac{A_1}{\pi}\right)^9\right)^{m/10} \\ &= 1 - m^2\left(5^{10}\frac{1}{2^9}\frac{A_{12}}{\pi}\left(\frac{A_1}{\pi}\right)^9\right)^{m/10}.\end{aligned} \tag{21}$$

For the above to tend to 1 as $m \to \infty$, we need $\frac{5^{10}}{2^9}\frac{A_{12}}{\pi}\left(\frac{A_1}{\pi}\right)^9 < 1$. This is equivalent to $A_{12}\left(\frac{A_1}{2}\right)^9 < \frac{\pi^{10}}{5^{10}}$, which implies $A_{12}\theta_1^9 < \left(\frac{\pi}{5}\right)^{10} = \frac{\pi}{5}\left(\frac{\pi}{5}\right)^9$. Note that if $\theta_1 = \frac{\pi}{5}$, then $A_1 = A_2 = 2\theta_1 = \frac{2\pi}{5}$. Then A_{12} could be at most $\frac{\pi}{5}$. But, this can't be because we have assumed $A_{12} \geq A_1$. Thus, we must have $\theta_1 < \frac{\pi}{5}$. In fact, $\theta_1 = \frac{\pi}{6}$ is the largest possible, in which case $A_{12} = A_1 = A_2 = \frac{\pi}{3}$. If $\theta_1 = \frac{\pi}{6}$, then $A_{12}\theta_1^9 < \frac{\pi}{5}\left(\frac{\pi}{5}\right)^9$ becomes $A_{12} < \frac{\pi}{5}\left(\frac{6}{5}\right)^9 \approx 3.24$. Therefore, since we are already assuming $A_{12} + 2A_1 \leq \pi$, this is essentially no further restriction on A_{12}, and the same would be true for all $\theta_1 \leq \frac{\pi}{6}$. This completes the proof. ■

4.2.3. Proof of Corollary 4

Proof Consider (3) and set $j' = j + k_{1,1}$ and $r = k_2 + k$. Then we view (3) as a probability equivalent to

$$1 - \sum_{\substack{j'=0 \\ j'+k_{1,2}+r=m,\ k_{1,2}\geq 9j'}}^{2m} \sum_{k_{1,2}=0}^{m} \sum_{r=0}^{2m} \binom{m}{k_{1,2}, j', r} \left(\frac{A_{12}+\frac{A_1}{2}}{\pi}\right)^{j'} \left(\frac{A_1}{2\pi}\right)^{k_{1,2}} \left(\frac{\pi - A_1 - A_{12}}{\pi}\right)^{r}. \tag{22}$$

Note that multinomial coefficients are maximized when the parameters all attain the same value. Thus, the multinomial term above is maximized when $k_{1,2}$, j' and r are all as close to one another as possible. Thus, given the additional constraint that $k_{1,2} \geq 9j'$, the multinomial term is maximized when $k_{1,2} = \frac{9m}{19}$, $j' = \frac{m}{19}$, and $r = \frac{9m}{19}$ (possibly with ceilings/floors as necessary if m is not a multiple of 19), (see the appendix, Section A.2, for a quick explanation), which means

$$\begin{aligned}
\binom{m}{k_{1,2}, j', r} &\leq \frac{m!}{(\frac{9m}{19})!(\frac{m}{19})!(\frac{9m}{19})!} && (23)\\
&\sim \frac{\sqrt{2\pi m}(\frac{m}{e})^m}{2\pi\frac{9m}{19}(\frac{9m}{19e})^{18m/19}\sqrt{2\pi\frac{m}{19}}(\frac{m}{19e})^{m/19}} && (24)\\
&= \frac{19\sqrt{19}}{18\pi m}\left((\frac{19}{9})^{18/19}19^{1/19}\right)^m \\
&\approx \frac{19\sqrt{19}}{18\pi m}2.37^m, && (25)
\end{aligned}$$

where (24) follows from Stirling's approximation for the factorial (and we use the notation $\sim$ to denote asymptotic equivalence, i.e. that two quantities have a ratio that tends to 1 as the parameter size grows).

Now assume $A_{12} + \frac{3}{4}A_1 \leq \frac{\pi}{2}$, which implies $\pi - A_1 - A_{12} \geq A_{12} + \frac{A_1}{2}$. Note also that $\pi - A_1 - A_{12} \geq A_1$ since it is assumed that $\pi - 2A_1 - A_{12} \geq 0$. Therefore, we can lower bound (22) by

$$1 - W_2\frac{19\sqrt{19}}{18\pi m}2.37^m\left(\frac{\pi - A_1 - A_{12}}{\pi}\right)^m, \tag{26}$$

where W_2 is the number of terms in the summation in (22), and is given by

$$W_2 = \frac{1}{6}\left(\left\lfloor\frac{m}{10}\right\rfloor + 1\right)\left(100\left\lfloor\frac{m}{10}\right\rfloor^2 + (5-30m)\left\lfloor\frac{m}{10}\right\rfloor + 3(m^2+3m+2)\right) \sim m^3. \tag{27}$$

Thus, (26) goes to 1 as $m \to \infty$ when $2.37\left(\frac{\pi - A_1 - A_{12}}{\pi}\right) < 1$, which holds if $A_1 + A_{12} > 0.58\pi$. ■

5. Discussion and Conclusion

In this work, we have presented a supervised classification algorithm that operates on binary, or one-bit, data. Along with encouraging numerical experiments, we have also included a theoretical analysis for a simple case. We believe our framework and analysis approach is relevant to analyzing similar, multi-level-type algorithms. Future directions of this work include the use of dithers for more complicated data geometries, identifying settings where real-valued measurements may be worth the additional complexity, analyzing geometries with non-uniform densities of data, as well as a generalized theory for high dimensional data belonging to many classes and utilizing multiple levels within the algorithm. In addition, we believe the framework will extend nicely into other applications such as hierarchical clustering and classification as well as detection problems. In particular, the membership function scores themselves can provide information about the classes and/or data points that can then be utilized for detection, structured classification, false negative rates, and so on. We believe this framework will naturally extend to these types of settings and provide both simplistic algorithmic approaches as well as the ability for mathematical rigor.

Acknowledgments

DN acknowledges support from the Alfred P. Sloan Foundation, NSF CAREER DMS #1348721 and NSF BIGDATA DMS #1740325. RS acknowledges support from the NSF under DMS-1517204. The authors would like to thank the reviewers for their suggestions, questions, and comments which significantly improved the manuscript.

Appendix A. Elementary Computations

A.1. Derivation of (16)

Suppose we have M objects that must be divided into 5 boxes (for us, the boxes are the 5 different types of hyperplanes). Let n_i denote the number of objects put into box i. Recall that in general, M objects can be divided into k boxes $\binom{M+k-1}{k-1}$ ways.

How many arrangements satisfy $n_1 \geq 9(n_2 + n_3)$? To simplify, let n denote the total number of objects in boxes 2 and 3 (that is, $n = n_2 + n_3$). Then, we want to know how many arrangements satisfy $n_1 \geq 9n$?

If $n = 0$, then $n_1 \geq 9n$ is satisfied no matter how many objects are in box 1. So, this reduces to the number of ways to arrange M objects into 3 boxes, which is given by $\binom{M+2}{2}$.

Suppose $n = 1$. For $n_1 \geq 9n$ to be true, we must at least reserve 9 objects in box 1. Then $M-10$ objects remain to be placed in 3 boxes, which can be done in $\binom{(M-10)+2}{2}$ ways. But, there are 2 ways for $n = 1$, either $n_2 = 1$ or $n_3 = 1$, so we must multiply this by 2. Thus, $\binom{(M-10)+2}{2} \times 2$ arrangements satisfy $n_1 \geq 9n$.

Continuing in this way, in general for a given n, there are $\binom{M-10n+2}{2} \times (n+1)$ arrangements that satisfy $n_1 \geq 9n$. There are $n+1$ ways to arrange the objects in boxes 2 and 3, and $\binom{M-10n+2}{2}$ ways to arrange the remaining objects after $9n$ have been reserved in box 1.

Therefore, the total number of arrangements that satisfy $n_1 \geq 9n$ is given by

$$\sum_{n=0}^{\lfloor \frac{M}{10} \rfloor} \binom{M-10n+2}{2} \times (n+1). \tag{28}$$

To see the upper limit of the sum above, note that we must have $M - 10n + 2 \geq 2$, which means $n \leq \frac{M}{10}$. Since n must be an integer, we take $n \leq \lfloor \frac{M}{10} \rfloor$. After some heavy algebra (i.e. using software!), one can express this sum as:

$$W = \frac{1}{12}\left(\left\lfloor \frac{M}{10} \right\rfloor + 1\right)\left(\left\lfloor \frac{M}{10} \right\rfloor + 2\right)\left(150\left\lfloor \frac{M}{10} \right\rfloor^2 - 10(4M+1)\left\lfloor \frac{M}{10} \right\rfloor + 3(M^2+3M+2)\right) \tag{29}$$

$$\sim M^4. \tag{30}$$

A.2. Derivation of (23)

Suppose we want to maximize (over the choices of a, b, c) a trinomial $\frac{m!}{a!b!c!}$ subject to $a + b + c = m$ and $a > 9b$. Since m is fixed, this is equivalent to choosing a, b, c so as to minimize $a!b!c!$ subject to these constraints. First, fix c and consider optimizing a and b subject to $a + b = m - c =: k$ and $a > 9b$ in order to minimize $a!b!$. For convenience, suppose k is a multiple of 10. We claim the optimal choice is to set $a = 9b$ (i.e. $a = \frac{9}{10}k$ and $b = \frac{1}{10}k$). Write $a = 9b + x$ where x must be some non-negative integer in order to satisfy the constraint. We then wish to compare $(9b)!b!$ to $(9b+x)!(b-x)!$, since the sum of a and b must be fixed. One readily observes that:

$$(9b+x)!(b-x)! = \frac{(9b+x)(9b+x-1)\cdots(9b+1)}{b(b-1)\cdots(b-x+1)} \cdot (9b)!b! \geq \frac{9b \cdot 9b \cdots 9b}{b \cdot b \cdots b} \cdot (9b)!b! = 9^x \cdot (9b)!b!.$$

Thus, we only increase the product $a!b!$ when $a > 9b$, so the optimal choice is when $a = 9b$. This holds for any choice of c. A similar argument shows that optimizing b and c subject to $9b + b + c = m$ to minimize $(9b)!b!c!$ results in the choice that $c = 9b$. Therefore, one desires that $a = c = 9b$ and $a + b + c = m$, which means $a = c = \frac{9}{19}m$ and $b = \frac{1}{19}m$.

References

Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.

Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proc. 38th annual ACM Symposium on Theory of computing*, pages 557–563. ACM, 2006.

Pervez M Aziz, Henrik V Sorensen, and J Vn der Spiegel. An overview of sigma-delta converters. *IEEE Signal Proc. Mag.*, 13(1):61–84, 1996.

Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. The Johnson-Lindenstrauss lemma meets compressed sensing. *preprint*, 2006.

Richard Baraniuk, Simon Foucart, Deanna Needell, Yaniv Plan, and Mary Wootters. Exponential decay of reconstruction error from binary measurements of sparse signals. *IEEE Trans. Info. Theory*, 63(6):3368–3385, 2017.

Petros Boufounos and Richard Baraniuk. 1-bit compressive sensing. In *Proc. IEEE Conf. Inform. Science and Systems (CISS)*, Princeton, NJ, March 2008.

Peter S. Burge, Max R. van Daalen, Barry J. P. Rising, and John S. Shawe-Taylor. Pulsed neural networks. In *Stochastic Bit-stream Neural Networks*, pages 337–352. MIT Press, Cambridge, MA, 1999.

Deng Cai, Xiaofei He, Jiawei Han, and Hong-Jiang Zhang. Orthogonal Laplacianfaces for face recognition. *IEEE Trans. Image Proc.*, 15(11):3608–3614, 2006.

Deng Cai, Xiaofei He, and Jiawei Han. Spectral regression for efficient regularized subspace learning. In *Proc. Int. Conf. Computer Vision (ICCV'07)*, 2007a.

Deng Cai, Xiaofei He, Yuxiao Hu, Jiawei Han, and Thomas Huang. Learning a spatially smooth subspace for face recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition Machine Learning (CVPR'07)*, 2007b.

Robert Calderbank, Sina Jafarpour, and Robert Schapire. Compressed learning: Universal sparse dimensionality reduction and learning in the measurement domain. *preprint*, 2009.

Valerio Cambareri, Chunlei Xu, and Laurent Jacques. The rare eclipse problem on tiles: Quantised embeddings of disjoint convex sets. *arXiv preprint arXiv:1702.04664*, 2017.

Emmanuel Candès, Justin Romberg, and Terrence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theory*, 52(2):489–509, 2006a.

Emmanuel Candès, Justin Romberg, and Terrence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 59(8):1207–1223, 2006b.

James C Candy and Gabor C Temes. *Oversampling delta-sigma data converters: theory, design, and simulation.* University of Texas Press, 1962.

Anna Choromanska, Krzysztof Choromanski, Mariusz Bojarski, Tony Jebara, Sanjiv Kumar, and Yann LeCun. Binary embeddings with structured hashed projections. In *Proc. 33rd International Conference on Machine Learning*, pages 344–353, 2016.

Nello Christianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge University Press, Cambridge, England, 2000.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

Mark A Davenport, Petros T Boufounos, Michael B Wakin, and Richard G Baraniuk. Signal processing with compressive measurements. *IEEE J. Sel. Topics Signal Process.*, 4(2): 445–460, 2010.

Sjoerd Dirksen and Alexander Stollenwerk. Fast binary embeddings with gaussian circulant matrices: improved bounds. *arXiv preprint arXiv:1608.06498*, 2016.

D. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52(4):1289–1306, 2006.

Jun Fang, Yanning Shen, Hongbin Li, and Zhi Ren. Sparse signal recovery from one-bit quantized data: An iterative reweighted algorithm. *Signal Processing*, 102:201–206, 2014.

Raja Giryes, Guillermo Sapiro, and Alexander M Bronstein. Deep neural networks with random gaussian weights: a universal classification strategy? *IEEE Trans. Signal Process.*, 64(13):3444–3457, 2016.

Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. Pattern Anal. Machine Intell.*, 35(12):2916–2929, 2013.

Sivakant Gopi, Praneeth Netrapalli, Prateek Jain, and Aditya V Nori. One-bit compressed sensing: Provable support and vector recovery. In *ICML (3)*, pages 154–162, 2013.

Ankit Gupta, Robert Nowak, and Benjamin Recht. Sample complexity for 1-bit compressed sensing and sparse classification. In *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, pages 1553–1557. IEEE, 2010.

Jurgen Hahn, Simon Rosenkranz, and Abdelhak M Zoubir. Adaptive compressed classification for hyperspectral imagery. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, pages 1020–1024. IEEE, 2014.

Xiaofei He, Shuicheng Yan, Yuxiao Hu, Partha Niyogi, and Hong-Jiang Zhang. Face recognition using Laplacianfaces. *IEEE Trans. Pattern Anal. Machine Intell.*, 27(3):328–340, 2005.

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Syst. Appl.*, 13(4):18–28, 1998.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*, 2016.

Blake Hunter, Thomas Strohmer, Theodore E Simos, George Psihoyios, and Ch Tsitouras. Compressive spectral clustering. In *AIP Conference Proceedings*, volume 1281, pages 1720–1722. AIP, 2010.

Laurent Jacques, Kévin Degraux, and Christophe De Vleeschouwer. Quantized iterative hard thresholding: Bridging 1-bit and high-resolution quantized compressed sensing. *arXiv preprint arXiv:1305.1786*, 2013a.

Laurent Jacques, Jason Laska, Petros Boufounos, and Richard Baraniuk. Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *IEEE Trans. Inform. Theory*, 59(4):2082–2102, 2013b.

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. *Machine learning: ECML-98*, pages 137–142, 1998.

William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Proc. Conf. Modern Anal. and Prob.*, New Haven, CT, June 1982.

Minje Kim and Paris Smaragdis. Bitwise neural networks. *arXiv preprint arXiv:1601.06071*, 2016.

Felix Krahmer and Rachel Ward. New and improved Johnson–Lindenstrauss embeddings via the restricted isometry property. *SIAM J. Math. Anal.*, 43(3):1269–1281, 2011.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Adv. in Neural Processing Systems (NIPS)*, pages 1097–1105, 2012.

Jason N Laska, Zaiwen Wen, Wotao Yin, and Richard G Baraniuk. Trust, but verify: Fast and accurate signal recovery from 1-bit compressive measurements. *IEEE Trans. Signal Processing*, 59(11):5289–5301, 2011.

Yann LeCun. The MNIST database of handwritten digits, 2018. `http://yann.lecun.com/exdb/mnist/`.

Zhouhan Lin, Matthieu Courbariaux, Roland Memisevic, and Yoshua Bengio. Neural networks with few multiplications. *arXiv preprint arXiv:1510.03009*, 2015.

Michele Marchesi, Gianni Orlandi, Francesco Piazza, and Aurelio Uncini. Fast neural networks without multipliers. *IEEE Trans. Neural Networks*, 4(1):53–62, 1993.

Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *IEEE Trans. Pattern Anal. Machine Intell.*, 32(3):448–461, 2010.

Yaniv Plan and Roman Vershynin. One-bit compressed sensing by linear programming. *Commun. Pur. Appl. Math.*, 66(8):1275–1297, 2013a.

Yaniv Plan and Roman Vershynin. Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach. *IEEE Trans. Inform. Theory*, 59(1):482–494, 2013b.

Yaniv Plan and Roman Vershynin. Dimension reduction by random hyperplane tessellations. *Discrete & Computational Geometry*, 51(2):438–461, 2014.

Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Proc. Adv. in Neural Processing Systems (NIPS)*, pages 1313–1320, 2009.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Computer Vision*, 115(3):211–252, 2015.

Patrice Y Simard and Hans Peter Graf. Backpropagation without multiplication. In *Proc. Adv. in Neural Processing Systems (NIPS)*, pages 232–239, 1994.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1–9, 2015.

Ming Yan, Yi Yang, and Stanley Osher. Robust 1-bit compressive sensing using adaptive outlier pursuit. *IEEE Trans. Signal Processing*, 60(7):3868–3875, 2012.

Xinyang Yi, Constantine Caravans, and Eric Price. Binary embedding: Fundamental limits and fast algorithm. In *Int. Conf. on Machine Learning*, pages 2162–2170, 2015.

Felix X. Yu, Sanjiv Kumar, Yunchao Gong, and Shih-Fu Chang. Circulant binary embedding. In *Int. Conf. on machine learning*, volume 6, page 7, 2014.