Classification Scheme for Binary Data with Extensions

Denali Molitor, Deanna Needell, Aaron Nelson, Rayan Saab and Palina Salanevich

Abstract In this chapter, we present a simple classification scheme that utilizes only 1-bit measurements of the training and testing data. Our method is intended to be efficient in terms of computation and storage while also allowing for a rigorous mathematical analysis. After providing some motivation, we present our method and analyze its performance for a simple data model. We also discuss extensions of the method to the hierarchical data setting, and include some further implementation considerations. Experimental evidence provided in this chapter demonstrates that our methods yield accurate classification on a variety of synthetic and real data.

1 Introduction

In this work, we discuss the problem of classification. More precisely, a supervised learning problem where one is given labeled *training data*, and from that data one wishes to determine a rule with which to accurately assign labels to unlabeled future *test data* points is considered. We focus on the setting where either by design or by

The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

D. Molitor · D. Needell · P. Salanevich
University of California, Los Angeles, CA, USA
e-mail: dmolitor@math.ucla.edu

D. Needell
e-mail: deanna@math.ucla.edu

P. Salanevich
e-mail: psalanevich@math.ucla.edu

A. Nelson (✉) · R. Saab
University of California, San Diego, CA, USA
e-mail: aan031@ucsd.edu

R. Saab
e-mail: rsaab@ucsd.edu

© Springer Nature Switzerland AG 2019

H. Boche et al. (eds.), *Compressed Sensing and Its Applications*,
Applied and Numerical Harmonic Analysis,
https://doi.org/10.1007/978-3-319-73074-5_4

application, the data is only available in a *binary* representation. Such representations may be obtained through compressive sampling, as in applications that have restricted bandwidth or energy constraints [19], or may be utilized to take advantage of simpler, faster, and cheaper hardware implementations [31, 37]. In general, compression and coarse quantization can be appealing due to efficient storage and computation. Additionally, such representations can still be utilized when performing inference tasks, e.g., see preliminary work of [4, 22, 26, 27]. This chapter presents a framework for learning inferences from highly quantized (single bit) data representations, with the key example being classification. Let us begin with some mathematical tools and notation.

1.1 Notation and Setup

Let $\{x_i\}_{i=1}^p \subset \mathbb{R}^n$ be a data set represented in matrix form

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

Let $A : \mathbb{R}^n \to \mathbb{R}^m$ be a linear map, and denote by $\text{sign} : \mathbb{R} \to \mathbb{R}$ the sign operator defined by

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

We generalize this operator for matrices elementwise, where for an m by p matrix M, and $(i, j) \in [m] \times [p]$, we define $\text{sign}(M)$ as the $m \times p$ matrix with entries

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

We now consider the setting where our method has access to training data of the form $Q = \text{sign}(AX)$, along with the labels $b = (b_1, \ \dots \ , b_p) \in \{1, \dots, G\}^p$, that identify each point x_i as belonging to one of the G possible classes. The rows of the $m \times n$ matrix A correspond to m *hyperplanes* in $\mathbb{R}^n$ and the sign information in Q captures on which side of the hyperplane each data point lies.

Throughout this chapter, A is assumed to have independent identically distributed standard Gaussian entries. We will present an approach from [41] that, given Q and b, allows for classification of a new unlabeled data point $x \in \mathbb{R}^n$ from its binary measurements $\text{sign}(Ax)$, and we will discuss various extensions and open problems associated with it.

1.2 Related Work and Background

We briefly mention here a few related topics that motivated the method proposed in [41]. We encourage the reader to see included references and others therein for more thorough background and details of these large areas of work.

Support vector machines (SVM) [3, 14, 24, 32, 47] are a popular method for classification. From labeled training data, SVMs seek an optimal hyperplane (or multiple hyperplanes) that separates the data or maximizes the geometric margin between the classes in the case where the data is not linearly separable. The approach described in this chapter is similar in flavor, but instead of optimizing hyperplane parameters to fit the data, it uses many random hyperplanes to identify separation of the data, and aggregates that information to decide upon a label.

The use of dimension reduction, that is the transformation of high dimensional data into geometrically similar low dimensional representations, appears in numerous contexts. The Johnson–Lindenstrauss Lemma guarantees the existence of a map that embeds p points into $O(\epsilon^{-2}\log(p))$ dimensions while approximately (up to ϵ) preserving the geometry [30]. In fact, a Johnson–Lindenstrauss map can be linear and can be obtained (with high probability) via a random draw from an appropriate distribution. Such random linear maps include those associated with Gaussian or subgaussian matrices and those resulting from selecting random rows of the discrete Fourier transform [1, 2, 5, 15, 35, 46]. Constructions of such maps play a crucial role in the field of *compressed sensing*, where they are used to sample high dimensional signals, yielding effective sampling rates that break the traditional Nyquist bounds [12, 13, 16]. Mathematically, for a signal $x \in \mathbb{R}^n$, one uses a measurement matrix $A \in \mathbb{R}^{m\times n}$ to acquire (possibly noisy) measurements of the form $y = Ax + z$, and the goal is to recover the signal x. For Johnson–Lindenstrauss type matrices A, the assumption that turns this ill-posed highly underdetermined problem into a well-posed problem is that the signal x is s-sparse, meaning that $\|x\|_0 := |\operatorname{supp}(x)| = s \ll n$.

For any digital compression scheme to be practical, one must consider the need to *quantize*, that is, to restrict data to a discrete set of values. Pushing quantization to the extreme in compressed sensing, the so-called *1-bit compressed sensing* problem captures only a single bit per measurement and asks to recover the measured signal x [4]. Formulated mathematically, one acquires measurements of the form $y = \operatorname{sign}(Ax)$, possibly with pre- or post-quantization noise. Clearly the normalization of x is lost under such scalar-invariant measurements, but under a norm assumption, efficient methods have been developed that accurately recover x [21, 29, 31, 42, 43, 55]. This normalization assumption can be overcome by introducing *dithers* to the measurements and modifying the methods [6, 34]. These branches of work have sparked recent interest in *binary embeddings*, those that map vectors to the binary cube while preserving angular information among the mapped vectors [7, 17, 20, 44, 53, 54]. The 1-bit compressed sensing problem and these binary embeddings motivate the work presented in this chapter, although the end goal of our consideration is classification rather than reconstruction.

Lastly, no modern chapter on classification would be complete without mentioning the burgeoning work on *deep learning*, which learns data representations using multiple levels of abstraction, usually referred to as layers or levels. Each layer can be viewed as a function that learns its parameters from the training data, so that its input data is transformed into a slightly more abstract and composite representation. From the composition of these functions, a (neural) network is constructed that solves the desired learning task (e.g., classification) by extracting relevant features of data. With the abundance of large data sets, these neural networks have become state of the art, yielding often astoundingly good results and techniques that continue to improve [33, 45, 50, 51]. On the other hand, although there is theoretical analysis (see e.g., [38, 40]), their success is often difficult to quantitatively analyze and interpret [56]. While the work presented in this chapter may be similar in flavor at a high scale, the aim is quite different—we intend to develop a simple approach to classification that allows for quantitative success bounds and simple geometric interpretability.

1.3 Organization

The remainder of the chapter is organized as follows. Section 2 motivates and describes the classification method. Section 2.1 provides an analysis for a simple data model, bounding the probability that a new test point is correctly labeled. Section 2.2 presents experimental results for the method on several synthetic and real examples. Section 3 extends the method to the setting of hierarchical classification, where the class labels have additional structure. Section 4 proposes several implementation variants that help guide parameter selection. We conclude with some final remarks in Sect. 5.

2 Simple Classification Approach

We next turn to a description of the classification method put forth in [41]. Let us first build some intuition for the approach. Consider the two-dimensional data X shown in the left plot of Fig. 1, consisting of three labeled classes (green, blue, red), and suppose we only have access to the binary data $Q = \text{sign}(AX)$. Note that Q contains information giving the side of each hyperplane (corresponding to the rows of A) on which each data point lies. Consider the four hyperplanes shown in the same plot, and suppose that we are given the new test point x (which visually appears to belong to the blue class) and its binary data $q = \text{sign}(Ax)$. Then, at first glance, a reasonable algorithm is to simply cycle through the hyperplanes and decide which class x matches most often. For example, for the hyperplane colored purple in the plot, x has the same sign (i.e., lies on the same side) as the blue and green classes.

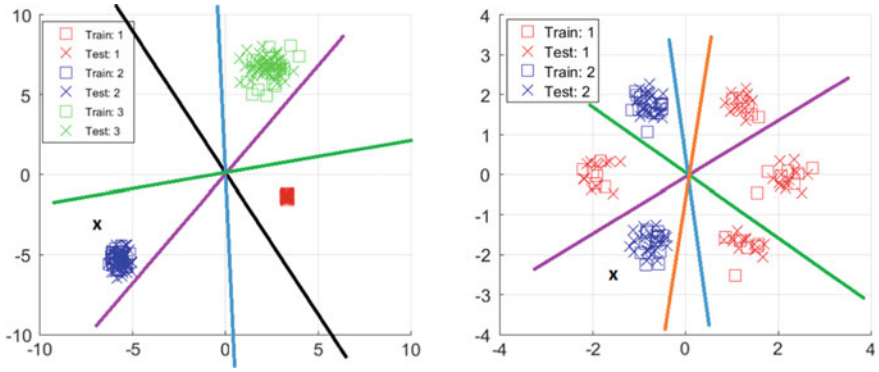


Fig. 1 Two motivating examples for the classification method

For the black hyperplane, x only matches the blue class, and so on. For this example, x will clearly match the blue class most often, and we could correctly assign it that label.

However, this may not work well for more complicated data. As an example, consider data as shown in the right plot of Fig. 1. Assuming that each point cloud has (approximately) the same cardinality, we have that for the blue and red hyperplanes, $\frac{2}{3}$ of the points lying in the same half-space as x are from the blue class. At the same time, for the purple and green hyperplanes $\frac{2}{3}$ of the points lying in the same half-space as x are from the red class. Thus, it is not possible to correctly classify x using just the information provided by individual hyperplanes. If we now consider hyperplane *pairs*, and find the class label that x most often agrees with (that is, we we find the class with points that most often share cones bounded by the pairs of hyperplanes with x), we are able to correctly classify x. Indeed, we have the following:

Pair	Blue class in the cone	Red class in the cone
Blue and red	$\frac{2}{3}$	$\frac{1}{3}$
Blue and purple	1	0
Blue and green	$\frac{1}{2}$	$\frac{1}{2}$
Red and purple	1	0
Red and green	$\frac{1}{2}$	$\frac{1}{2}$
Purple and green	$\frac{1}{2}$	$\frac{1}{2}$
Overall	$\frac{25}{6}$	$\frac{11}{6}$

We now describe the approach more formally. Again, denote by $X \in \mathbb{R}^{n\times p}$ the matrix whose columns contain the data points. Let $A \in \mathbb{R}^{m\times n}$ have rows corresponding to the normal vectors of m randomly oriented hyperplanes that pass through the origin (e.g., A could have i.i.d. Gaussian entries), and $Q = \text{sign}(AX)$ denote the binary sign information. Then, the training algorithm proceeds in L "levels". In the

ℓth level, m index sets $\Lambda_{\ell,i} \subset [m]$, $|\Lambda_{\ell,i}| = \ell$, $i = 1, \ldots, m$, are randomly selected, and each index set corresponds to an ℓ-tuple of hyperplanes. Let $Q^{\Lambda_{\ell,i}} \in \mathbb{R}^{\ell \times p}$ be the submatrix consisting of the rows of Q whose indices belong to $\Lambda_{\ell,i}$. Each column of $Q^{\Lambda_{\ell,i}}$ then gives a sign pattern of length ℓ corresponding to a training data point and the ℓ hyperplanes contained in $\Lambda_{\ell,i}$. Let us denote the number of different sign patterns of training points corresponding to $\Lambda_{\ell,i}$ (that is, number of different columns of $Q^{\Lambda_{\ell,i}}$) by $T_{\ell i}$.

At a given level ℓ, for the tth sign pattern and gth class, a *membership index* parameter $r(\ell, i, t, g)$ that uses knowledge of the number of training points in class g having the tth sign pattern, is calculated for every ℓ-tuple $\Lambda_{\ell,i}$. Below, $P_{g|t} = P_{g|t}(\Lambda_{\ell,i})$ denotes the number of training points from the gth class with the tth sign pattern at the ith set selection in the ℓth level:

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

Note that the first fraction in (1) indicates the proportion of training points in class g out of all points with sign pattern t (at the ℓth level and ith set selection). 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. Intuitively, larger values of $r(\ell, i, t, g)$ suggest that the tth sign pattern is more heavily dominated by class g; thus, if a signal with unknown label corresponds to the tth sign pattern, we will be more likely to classify it into the gth class. With this intuition, we can then assign a label to a new test point x using its binary data $q = \text{sign}(Ax)$. For each class g, we simply sum the membership index function values over all ℓ, i, and t, for those sign patterns t that match the sign pattern of the new test point x (which is known via the data q). Thus, we obtain a value for each class g and the label for x is then decided by simply taking the class g corresponding to the largest sum. The training and classification portions of this method are summarized in Algorithms 1 and 2.

Algorithm 1: Training

Input: binary training data Q, training labels b, number of classes G, number of layers L
for *ℓ from 1 to L, i from 1 to m* **do**
  select: Randomly select $\Lambda_{\ell,i} \subset [m]$, $|\Lambda_{\ell,i}| = \ell$
  determine: Determine the $T_{\ell,i} \in \mathbb{N}$ unique column sign patterns in $Q^{\Lambda_{\ell,i}}$
  for *t from 1 to $T_{\ell,i}$, g from 1 to G* **do**
    compute: Compute $r(\ell, i, t, g)$ by (1)
  end
end

Algorithm 2: Classification

Input: binary data q, number of classes G, number of layers L, learned parameters $r(\ell, i, t, g)$, $T_{\ell,i}$, and $\Lambda_{\ell,i}$ from Algorithm 1
Initialize: $\tilde{r}(g) = 0$ for $g = 1, \ldots, G$.
for *ℓ from 1 to L, i from 1 to m* **do**
  identify: Identify the pattern $t^\star \in [T_{\ell,i}]$ to which $q^{\Lambda_{\ell,i}}$ corresponds
  for *g from 1 to G* **do**
    update: $\tilde{r}(g) = \tilde{r}(g) + r(\ell, i, t^\star, g)$
  end
end
scale: Set $\tilde{r}(g) = \frac{\tilde{r}(g)}{Lm}$ for $g = 1, \ldots, G$
classify: $\widehat{b}_x = \text{argmax}_{g \in \{1,\ldots,G\}} \{\tilde{r}(g)\}$

2.1 Analytical Justification

One of the benefits of this simple approach to classification is that it can be mathematically analyzed and understood. Indeed, we present here a result from [41] that bounds the probability of accurate classification for a simple data model, showcasing the potential of this method to be rigorously supported mathematically. Here, we focus on the setting where the signals are two-dimensional, belonging to one of two classes, and consider a single level (i.e., $L = 1$, $n = 2$, and $G = 2$). For simplicity of analysis, we consider the continuous setting and assume the true classes G_1 and G_2 are two disjoint *cones* in $\mathbb{R}^2$ in which the training data lies in a uniform density. See Fig. 2 for a visualization of the setup; we will describe the relevant parameters next.

Let A_1 denote the angular measure of G_1 and define A_2 similarly for G_2. Also, define A_{12} as the angle between classes G_1 and G_2. Assume all angles are such that no hyperplane will intersect both classes at once, i.e., $A_{12} + A_1 + A_2 \leq \pi$. Suppose

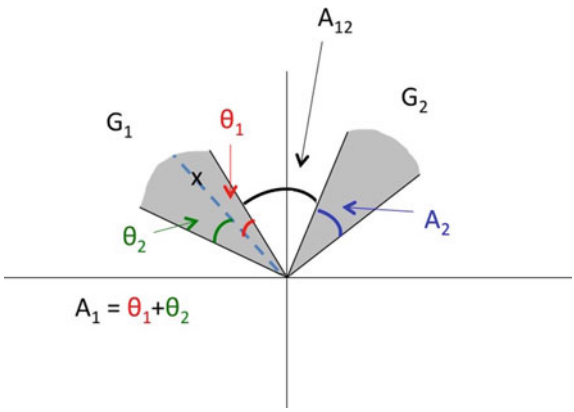


Fig. 2 Visualization of the analysis setup for two classes in two dimensions

that the test point $x \in G_1$, partitioning A_1 into two disjoint pieces, yielding angles θ_1 and θ_2, where $A_1 = \theta_1 + \theta_2$ (see Fig. 2).

The membership index parameter (1) is still used; however, in this continuous setting, we use the analogous formula with angles instead of numbers of training points:

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

where $A_{g|t}$ denotes the angle of class g with the tth sign pattern for the ith ℓ-tuple of hyperplanes in the ℓth layer. Denote by $t_i^\star$ the sign pattern of the test point x with the ith hyperplane at the first level (i.e., $\ell = 1$). Let $\widehat{b}_x$ denote the classification label assigned to x by Algorithm 2. Then Theorem 1 below describes the probability that x is classified correctly with $\widehat{b}_x = 1$. For simplicity, Theorem 1 is stated under the assumption that $A_1 = A_2$ and the test point x lies in the middle of class G_1 (i.e., $\theta_1 = \theta_2$). The analysis follows similarly for the general case, with more tedious computations and messier results, see [41] for the proof details.

Theorem 1 (From [41]) *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,*

$$\begin{aligned}\mathbb{P}[\widehat{b}_x = 1] \geq 1 - \sum_{\substack{j=0 \\ j+k_{1,\theta_1}+k_{1,\theta_2}+k_2+k=m,\ k_{1,\theta_2} \geq 9(j+k_{1,\theta_1})}}^{m} \sum_{k_{1,\theta_1}=0}^{m} \sum_{k_{1,\theta_2}=0}^{m} \sum_{k_2=0}^{m} \sum_{k=0}^{m} \binom{m}{j, k_{1,\theta_1}, k_{1,\theta_2}, k_2, k} \\ \times \left(\frac{A_{12}}{\pi}\right)^{j} \left(\frac{A_1}{2\pi}\right)^{k_{1,\theta_1}+k_{1,\theta_2}} \left(\frac{A_1}{\pi}\right)^{k_2} \left(\frac{\pi - 2A_1 - A_{12}}{\pi}\right)^{k}.\end{aligned} \tag{3}$$

Although the bound on the probability given in this theorem is quite cumbersome, some useful properties are immediate. For example, this probability bound tends to 1 as m grows large. Indeed, the following two corollaries show precisely this behavior.

Corollary 1 *Consider the setup of Theorem 1. Suppose $A_{12} \geq A_1$ and $A_{12} \geq \pi - 2A_1 - A_{12}$. Then $\mathbb{P}[\widehat{b}_x = 1] \to 1$ as $m \to \infty$.*

Corollary 2 *Consider the setup of Theorem 1. 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 $m \to \infty$.*

These asymptotic results are noteworthy, but of course one more importantly would like to know at what rate this probability increases to 1 as a function of the

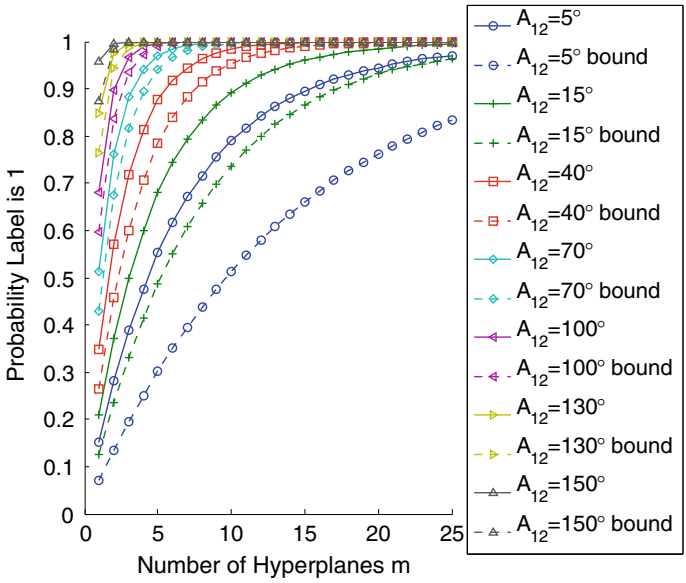


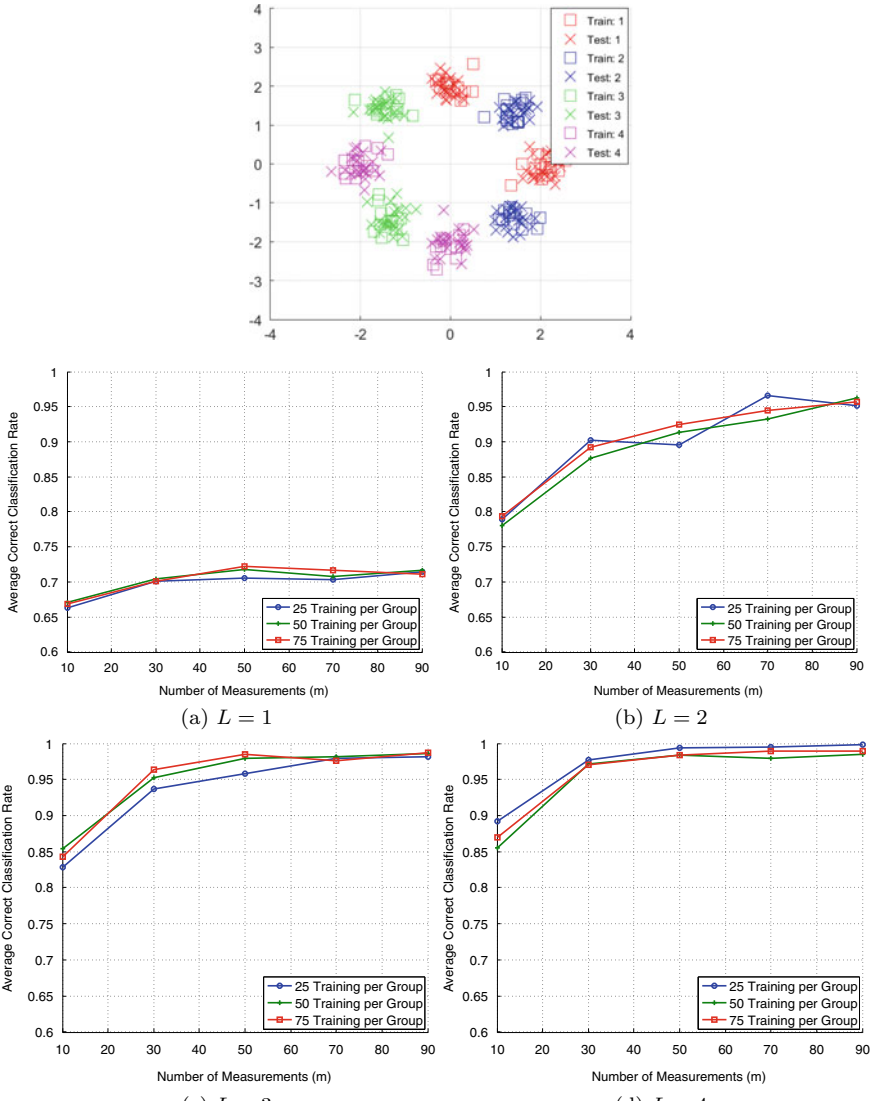
Fig. 3 $\mathbb{P}[\widehat{b}_x = 1]$ versus the number of hyperplanes m when A_{12} is varied (see legend), $A_1 = A_2 = 15°$, and $\theta_1 = \theta_2 = 7.5°$. The solid lines indicate the true (simulated) probability and the dashed lines indicate the bound (3) provided in Theorem 1

number m of hyperplanes. Indeed, it can be seen from the proofs in [41] that the probability converges to 1 exponentially in m. To illustrate this, the rates of the bound provided by Theorem 1 are displayed in Fig. 3 along with the (simulated) true value of $\mathbb{P}[\widehat{b}_x = 1]$. Although the bound is clearly not sharp, it exhibits the same overall behavior as the true probability of accurate classification.

2.2 *Experimental Results*

We present here a small collection of experimental results for the classification method that show its performance on synthetic and real data. The first experiment considers synthetic data consisting of eight Gaussian clouds, belonging to four classes. A new test point is drawn according to one of these distributions and is then classified by the method. The average correct classification rate (where the "correct" label is deemed to be the label matching the point cloud from which the test point x was drawn) is calculated over 50 trials and displayed. Figure 4 showcases the classification accuracy for various numbers of levels L, showing that as one expects, more levels are needed for accurate classification for complicated data geometries.

Next, we test the method on several real data sets. First, Fig. 5 shows average accuracy results for classifying the "0" versus "1" handwritten digits from the MNIST



(a) $L = 1$ (b) $L = 2$

(c) $L = 3$ (d) $L = 4$

Fig. 4 Data (top) is 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. 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) $L = 4$

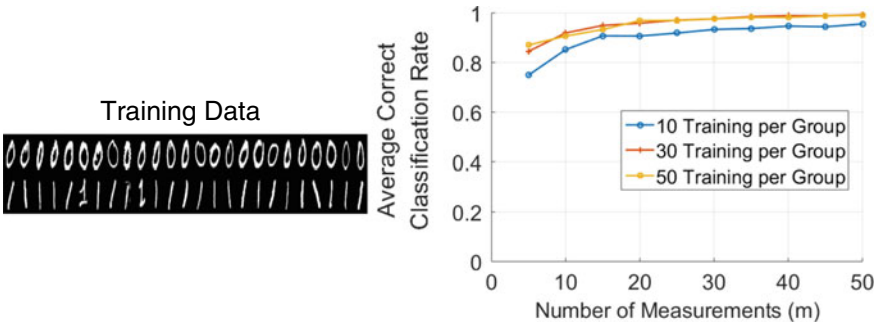


Fig. 5 Classification experiment using the handwritten "0" and "1" digit images from the MNIST data set, with 50 test points per group, $L = 1$, $n = 28 \times 28 = 784$, and 30 trials of randomly generating A. Left: example data. Right: average correct classification rate versus m and for the indicated number of training points per class

data set [36]. For this data, we only needed one level to get accurate results, perhaps because the images of the "0" and "1" digits are well separated in space. Not surprisingly, when classifying all ten digits, more levels are needed in order to obtain decent accuracy; see Fig. 6.

We also tested the method on the problem of facial classification, using the YaleB data set [8–10, 28]. Figure 7 shows classification results using six layers. Note that the results appear noisier due to the smaller size of the data set.

Lastly, we tested the method on recently acquired survey data from patients with Lyme disease, from the MyLymeData project hosted by lymedisease.org that now has

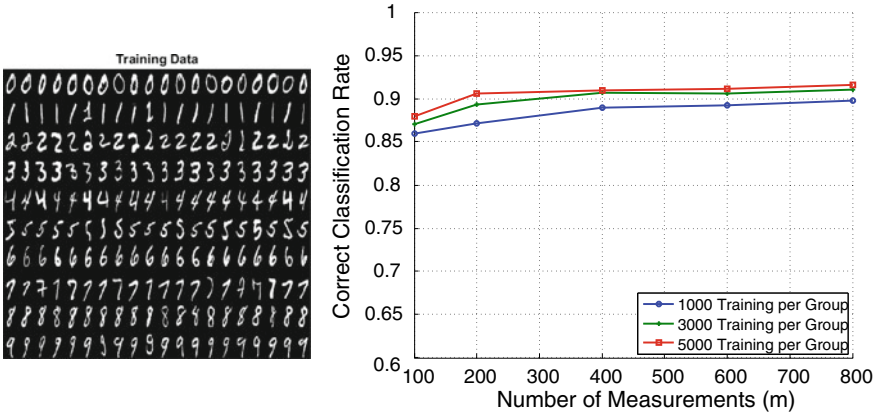


Fig. 6 Correct classification rate (right) versus m when using all ten (0–9) handwritten digits from the MNIST data set (left) with 1,000, 3,000, and 5,000 training points per group, $L = 18$, $n = 28 \times 28 = 784$, 800 test points per group (8,000 total), and a single instance of randomly generating A

over 10,000 patients enrolled. Figure 8 shows classification results using the survey responses for the symptom-related questions as our data matrix. This matrix consists of 3686 "unwell" patients and 362 "well" patients (4048 patients in total), that each answered 12 symptom-related questions (the "well" patients were asked about their

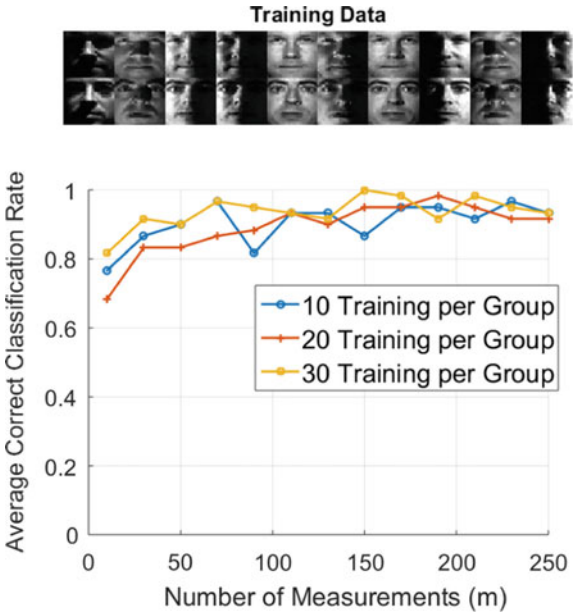


Fig. 7 Classification experiment using two individuals from the extended YaleB data set (top), $L = 6$, $n = 32 \times 32 = 1024$, 30 test points per group, and 30 trials of randomly generating A. Bottom: average correct classification rate versus m and for the indicated number of training points per class

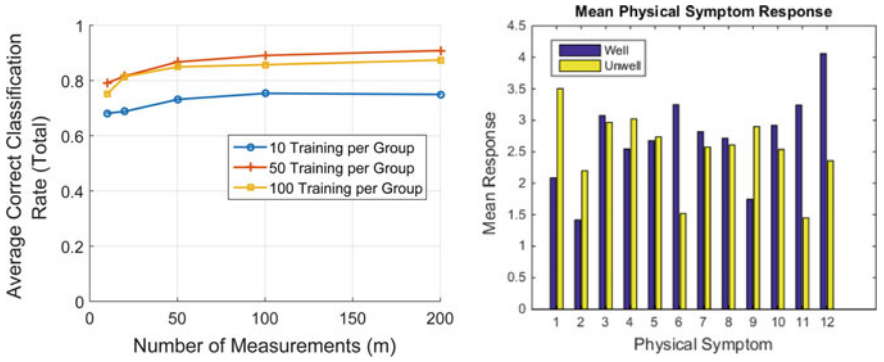


Fig. 8 Left: Results from classification approach on symptom data using 5 layers for various numbers of randomly selected training points (patients). Right: Means on the survey questions for these groups

worst symptoms while being sick). We randomly select a number of those patients from each group to serve as our training data, and the remaining as our test data. The left plot of Fig. 8 demonstrates the ability to accurately identify well versus unwell patients from the symptoms (current or past) that they report. Since the "well" patients were asked about their worst *prior* symptoms, one might ask whether it is simply the case that "well" patients showcase higher (or lower) symptom levels in general, making classification easy. However, the right plot of the figure demonstrates this is not the case, and that perhaps more intricate and complex symptom patterns are at work.

3 Hierarchical Classification

Next, we extend the classification approach described in the previous sections to the problem of hierarchical or multi-scale classification. In this setting, the class labels have additional structure, often taking the form of a tree. For example, in image classification problems, the data may contain images of inanimate and living objects. Then, within each of those classes the data may be further identified as images of vehicles and toys, or humans and animals. The data could then be further subdivided into classes of various animal types, and so on. Visualized as a tree, we view the children of each node as corresponding to its subclasses. Each data point in this case would have a label corresponding to a leaf of the tree, but also possesses the characteristics of all the labels of its ancestors. *Hierarchical classification* makes use of this information and structure between groups in classifying the data [23, 49]. Extensions of popular classification methods such as the support vector machine (SVM) to the hierarchical setting are not straightforward, and such approaches often decompose the problem into many subproblems leading to higher computational complexities [11, 52]. Here, we apply the simple classification method discussed in Sect. 2 to this hierarchical setting, and show that computational advantages are often possible. In particular, the method is likely to be particularly useful for hierarchical data in which certain subclasses of data are more or less difficult to classify than others.

We now describe the proposed adjustment for handling hierarchical classification, based on [39], where the labels possess some sort of tree structure. We use the same notation as for the methods described in previous sections. The key observation for the modification is that, if we know in advance that certain classes may require fewer levels for classification with sufficient accuracy, we may isolate these classes in an initial classification that uses fewer levels and then further classify among the remaining classes using more levels, as needed. This strategy leads to computational savings without sacrificing accuracy when some classes are more easily discerned from the others. Fortunately, this type of structure occurs naturally in many applications. For example, in medical brain imaging, it is typically much easier to classify patients with tumors than patients with various types of dementia [18, 25]. In cases like this, the method may utilize fewer levels for the easier classification steps. This approach is described formally in Algorithms 3 and 4.

Algorithm 3: Proposed adjustment for hierarchical classification (training).

Input: binary training data Q, training labels b, set of class groupings S_c for each node H_c in the tree of classifications H, number of levels $L = (L_1, \dots L_C)$ to be used in each classification.

for $H_c \in H$ **do**

  identify: Q_c, the submatrix of rows of Q corresponding to training labels of b contained in some set in S_c.

  define: $\tilde{b}$ as labels indicating to which set of S_c a given row of Q_c corresponds.

  train: a classifier as in Algorithm 1 with training data Q_c, labels $\tilde{b}$, number of groups $|S_c|$ and number of levels L_c as input.

end

Algorithm 4: Proposed adjustment for hierarchical classification (testing).

Input: binary testing data q, set of class groupings S_c, learned parameters $r(\ell, i, t, g)$, $T_{\ell,i}$ and $\Lambda_{l,i}$ for the classification associated to each node H_c in the tree of classifications H, number of levels $L = (L_1, \dots L_C)$ to be used in each classification.

set: $H_c = H_1$, the root classification.

while H_c *is not null,* **do**

  classify: q into one of the sets contained in S_c, as in Algorithm 2, with learned parameters $r(\ell, i, t, g)$, $T_{l,i}$, $\Lambda_{l,i}$ from H_c.

  if q *is predicted to belong to a single class* **then**

    set: H_c to be null.

  else

    set: H_c to the node corresponding to the predicted set of classes within S_c.

  end

end

3.1 Experimental Results

In this section, we showcase experiments from [39] that demonstrate the computational gains achieved by Algorithms 3 and 4 compared with direct classification into each individual group via "flat multiclass classification" as in Algorithms 1 and 2 (see Fig. 9). We first consider a simple two-dimensional example to aid in visualization; the data is shown in Fig. 10, where each color represents a different class from six classes in total. Since we expect classifying points from the red and yellow classes to be easier, we may use fewer levels than in classifying points as green, black, blue or cyan. Therefore, we first predict whether a test point is red or yellow versus green, black, blue or cyan using only one level. If the test point was predicted to be red or yellow, we then discern between these two classes again using only a single level. If the test point was predicted to be green, black, blue or cyan, we then predict among these classes by using varying numbers of levels. Accuracy and computational results are shown in Fig. 10 for varying numbers of measurements m. We see a significant reduction in computational cost using the hierarchical strategy without sacrificing accuracy. Note that the computational savings are realized for the test points predicted to belong to the red or yellow class, since classifying into these

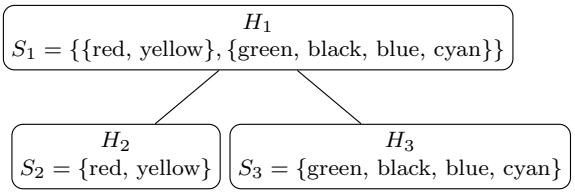


Fig. 9 Hierarchical classification tree used to classify two-dimensional synthetic data as shown in Fig. 10

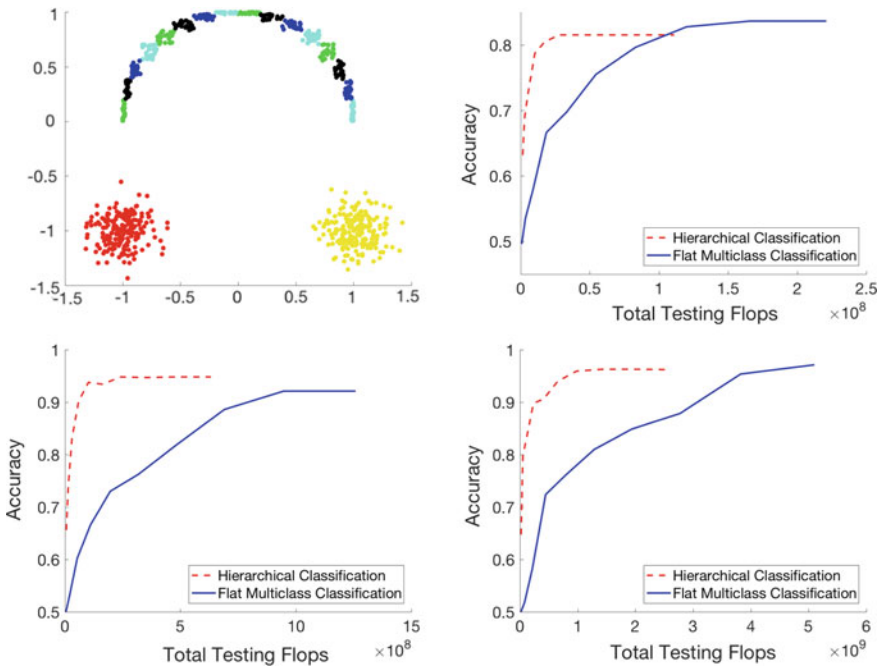


Fig. 10 For the data distributed as given in the upper left plot, where each color represents a different class, we classify test data either by flat multiclass classification or our proposed hierarchical classification strategy where the first classification discerns between red or yellow versus green, black, blue or cyan. Accuracy and testing flops required are given in the subsequent plots using varying numbers of levels and $m = 20, 50$ and 100 respectively. Results are averaged over 10 trials

groups requires fewer levels and thus fewer calculations. The computational savings of the hierarchical strategy are thus highly dependent on the distribution of the test data. In this experiment, we classify 200 test points from each of the red and yellow classes and 100 test points from each of the green, black, blue, and cyan classes, so that there are an equal number of test points from the "arc" and from the Gaussian clusters.

Although not inherently hierarchical in nature, we demonstrate that our hierarchical strategy can lead to computational savings on the MNIST data set of handwritten digits [36]. Consider the digits 1–5. Intuitively and in practice, the digit 1 tends to be easier to classify correctly than the other digits. For example, if we apply the multiclass classification from [41] to classify the digits 1–5 using 1000 training points for each class, 10 levels and testing on 200 training points from each class, we find that 98.5% of the 1s are classified correctly, whereas the overall accuracy of classifying the digits 1–5 was 89.2% (the accuracy for classifying digits 2–5 was 86.88%). Thus, it is reasonable to expect that fewer levels are required for sufficiently accurate classification of the 1s than are required to classify the remaining digits.

Considering the digits 1–5, we can thus induce hierarchical structure by first classifying into 1s (which tend to be easier) versus the other digits, followed by classification into the digits 2, 3, 4, and 5. Five levels are used for the first classification into 1s versus 2–5s and a varying number of levels (5–10) are used for the subsequent classification. We again see a reduction in the total testing flops required to achieve a given accuracy. Since this tree is fairly "shallow," as expected the improvements are mild. We would expect a more significant reduction in computation via a hierarchical strategy for real data that has a larger and more imbalanced tree structure. Additionally, the test data includes an equal number of points corresponding to each digit, so we see computational savings for approximately 1/5 of the test points. If we expected the frequency of the digit 1 to be higher, we would expect the computational savings to be more significant as well.

4 Implementation Considerations

Here, we consider some implementation details and remarks for future work in this direction.

4.1 Parameter Selection

The key parameters the user must select in this simple classification approach are the number of measurements m and the number of levels L. The relationship between the number of measurements m and the performance of the method (see Corollaries 1 and 2) conforms with their analogous relationship in other settings like 1-bit compressed sensing and binary hashing (see e.g., [4, 22, 48]). Namely, increasing m exponentially improves the success probability of the method at hand. Henceforth, we focus here on the choice of levels L. We propose a simple scheme that uses the membership index function values on the training data to decide how many levels L are sufficient for accurate classification. This scheme can be viewed as a simple analog of cross-validation (Fig. 11).

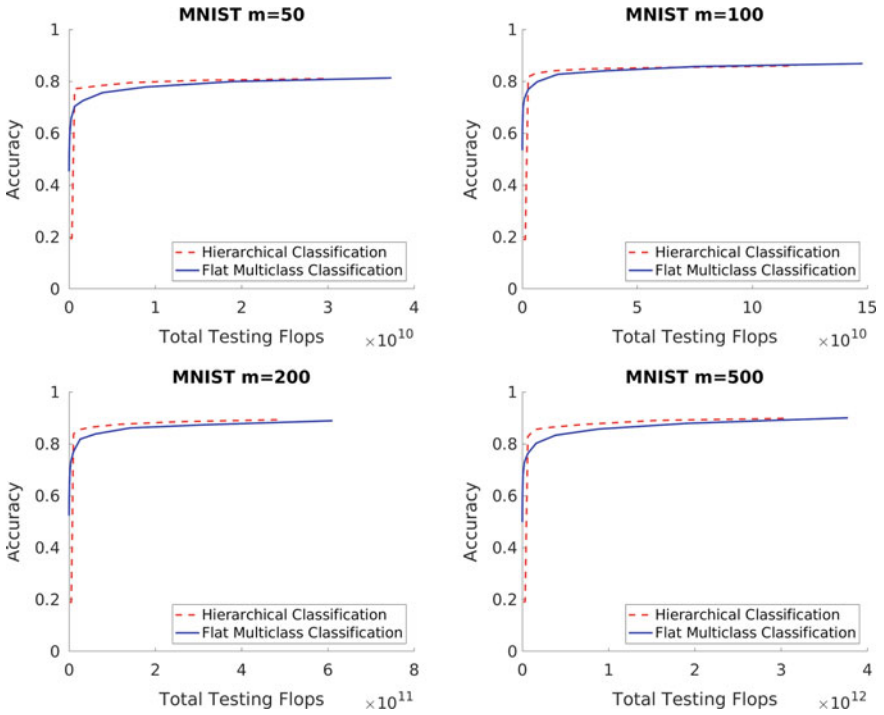


Fig. 11 Accuracy and testing flops required for flat multiclass classification versus the proposed hierarchical classification strategy in classifying digits 1–5 in the MNIST data set are given using $m = 50$, 100, 200, and 500, respectively. Results are averaged over 10 trials

Intuitively, the membership index values correspond to the level of confidence that a point belongs to a certain class. Thus ideally, for a fixed data point, we hope to see a single large membership value for one class and small values for the other classes. This motivates a scheme where examining the largest membership function value across classes, averaged over all the data, dictates an appropriate choice of L. More precisely, for a given level ℓ, one could consider running the testing method Algorithm 2 over all (or part of) the training data and computing the functions $\tilde{r}(g)$ for all represented classes g. Doing this at level ℓ for a data point with sign pattern t yields a value $\tilde{r}(g)$ for each class g, which we will now write as $\tilde{r}_{\ell,t}(g)$. We may then consider the average over all sign patterns at the level ℓ of the largest membership indices that is given by

$$\mu_\ell := \frac{1}{T} \sum_{t \in T} \max_g \tilde{r}_{\ell,t}(g),$$

where T is a set containing all represented sign patterns in the training data. We view large values of μ_ℓ as informing us that level ℓ is providing strong classification accuracy. Thus, if we view these values over various ℓ, we could stop using more

levels once these values plateau or start decreasing. To verify this approach, we test this scheme on MNIST data for classifying 0–1 digits or 0–5 digits (with $p = 500$ and $m = 50$), see Figs. 12 and 13. Here, we notice that there appears to be a correlation between the level ℓ that yields maximal value μ_ℓ (right) and the point where using more levels does not lead to significant more accuracy (left).

Investigating this correlation quantitatively and theoretically will be an interesting direction for future work. For example, if we assume that the angle between any two classes is at least θ, we conjecture that after $L \leq O\left(\frac{1}{\theta^{n-1}}\right)$ levels (where n is the ambient dimension of the problem), the plateau begins, as adding more hyperplanes will create empty cones with high probability. Of course, a better bound should also depend on the total number m of hyperplanes out of which we select, and the number of test points p. Drawing such connections would be fruitful future directions of work.

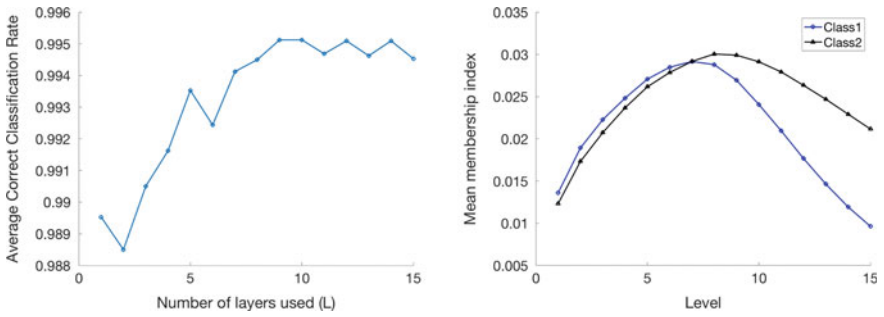


Fig. 12 MNIST 0–1 digits (single trial). Left: Average classification rate as function of levels. Right: Mean membership index function

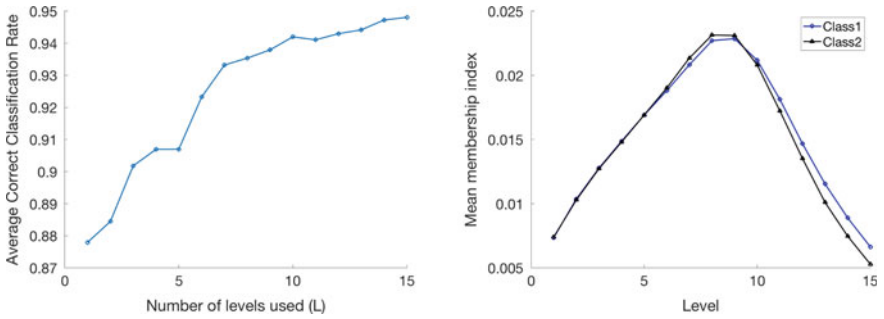


Fig. 13 MNIST 0 and 5 digits. Left: Average classification rate as function of levels. Right: Mean membership index function

4.2 Dynamic Hyperplane Selection

Another implementation concern and possible direction for future research is the optimal choice of hyperplanes on each layer. In the presented implementation of the classification algorithm, for each layer $\ell \in \{1, \ldots, L\}$, the collections $\Lambda_{\ell,i}$ of hyperplanes are selected uniformly at random out of all possible ℓ-tuples. While this approach allows one to derive nice theoretical bounds such as Theorem 1, it might be beneficial for reconstruction if one instead chooses sets $\Lambda_{\ell,i}$ in a data-dependent way, so that hyperplanes in $\Lambda_{\ell,i}$ together provide a good separation of the training data. One might achieve this by using cross-validation or an approach similar to that described in Sect. 4.1, so that for each layer ℓ we reuse information obtained from the previous levels to decide which ℓ-tuples of hyperplanes could potentially allow for good class separation.

For instance, in the example described in Fig. 1 (right), we can see that for the blue and red hyperplanes, we have that in one half-space $\frac{2}{3}$ of all point are blue and $\frac{1}{3}$ are red, and in the other half-space all the points are red. Similarly, for the purple and green hyperplanes, we have that in both half-spaces, $\frac{1}{3}$ of all points are blue and $\frac{2}{3}$ are red. We can deduce that these pairs of points are "similar" in the sense that they divide the training data in similar ways. Thus it may be more beneficial to consider pairs of hyperplanes from different groups for the next level, that is {red, purple}, {blue, green}, {red, green}, and {blue, purple}. One can see that these pairs are indeed enough to separate clusters of training points. Alternatively, one could simply ignore hyperplane tuples that produce empty cones, which could happen frequently especially in high dimensions. Such dynamic selection of hyperplane tuples could lead to improved performance but perhaps more challenging analysis.

4.3 Efficient Representations

Next, we consider settings where the data in raw form is either not available or is too large to measure. Often, such data is instead available only by its adjacency graph, capturing distance measures between points. Such graphs arise naturally in many applications such as wireless communications, sensor networks and astronomy. Alternatively, we may wish to use such a representation to improve the classification accuracy. In this section, we demonstrate empirically that our approach is also robust to this type of data representation. In our first experiment, we use the MNIST 0–1 handwritten digit data but rather than measuring this data directly, we select a subset of training data and compute its adjacency matrix X where X_{ij} is the (Euclidean) distance between the ith and jth image. We then measure $Q = \operatorname{sign}(AX)$ and proceed as usual. The results are shown in Fig. 14 (left), where actually we see an *improvement* in classification accuracy. We conjecture the improvement arises from the fact that

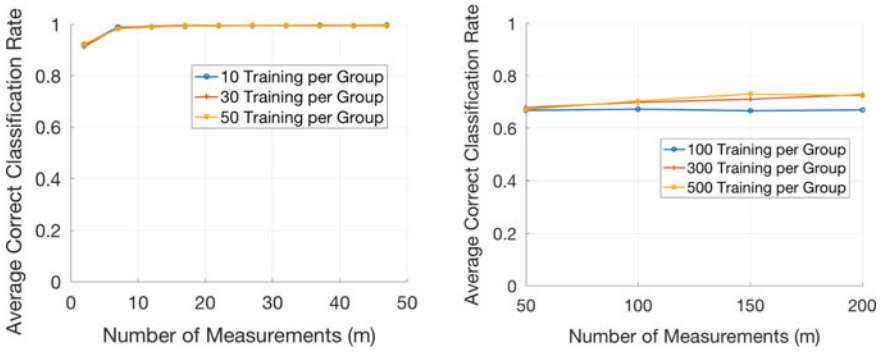


Fig. 14 Graph representations, average classification rate as function of levels. Left: MNIST 0–1 digits. Right: MNIST 0–9 digits

the adjacency columns are more linearly separable than the original raw data. Next, we use all ten digits of the MNIST data and create the adjacency matrix. Since the adjacency matrix scales with the number of training points, we are no longer free to use as many as we wish without bogging down computation. Thus, we are forced to use a smaller number of training points than in Fig. 6, and unsurprisingly, we see less accurate classification results, as shown in Fig. 14 (right). It would be interesting future work to study the geometry of such adjacency data and to develop an analogous analysis.

5 Conclusion

We have presented a simple classification method from [41] that can be applied to data represented in binary form. We have provided experimental results showcasing its classification accuracy on real and synthetic data as well as supporting theoretical analysis. In addition, we have demonstrated that the classification algorithm can be readily adapted to classify data in a hierarchical way that improves computational efficiency. In addition, we present some preliminary implementation modifications that can yield both computational and accuracy gains, and point out interesting directions for future work.

Acknowledgements Molitor and Needell were partially supported by NSF CAREER grant #1348721 and NSF BIGDATA #1740325. Saab was partially supported by the NSF under DMS-1517204.

References

1. N. Ailon, B. Chazelle, Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform, in *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing* (ACM, 2006), pp. 557–563
2. D. Achlioptas, Database-friendly random projections: Johnson-Lindenstrauss with binary coins. J. Comput. Syst. Sci. **66**(4), 671–687 (2003)
3. A.M. Andrew, *An introduction to support vector machines and other kernel-based learning methods by Nello Christianini and John Shawe-Taylor* (Cambridge University Press, Cambridge, 2000), xiii+ pp. 189, ISBN 0-521-78019-5 (hbk, £ 27.50)
4. P. Boufounos, R. Baraniuk, 1-bit compressive sensing, in *Proceedings of IEEE Conference on Information, Science and Systems (CISS)* (Princeton, NJ, 2008)
5. R. Baraniuk, M. Davenport, R. DeVore, M. Wakin, The Johnson-Lindenstrauss lemma meets compressed sensing. Preprint **100**(1) (2006)
6. R. Baraniuk, S. Foucart, D. Needell, Y. Plan, M. Wootters, Exponential decay of reconstruction error from binary measurements of sparse signals. IEEE Trans. Inf. Theory **63**(6), 3368–3385 (2017)
7. A. Choromanska, K. Choromanski, M. Bojarski, T. Jebara, S. Kumar, Y. LeCun, Binary embeddings with structured hashed projections, in *Proceedings of The 33rd International Conference on Machine Learning* (2016), pp. 344–353
8. D. Cai, X. He, J. Han, Spectral regression for efficient regularized subspace learning, in *Proceedings of International Conference on Computer Vision (ICCV'07)* (2007)
9. D. Cai, X. He, Y. Hu, J. Han, T. Huang, Learning a spatially smooth subspace for face recognition, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Machine Learning (CVPR'07)* (2007)
10. D. Cai, X. He, J. Han, H.-J. Zhang, Orthogonal laplacianfaces for face recognition. IEEE Trans. Image Process. **15**(11), 3608–3614 (2006)
11. S. Cheong, S.H. Oh, S.-Y. Lee, Support vector machines with binary tree architecture for multi-class classification. Neural Inf. Process. Lett. Rev. **2**(3), 47–51 (2004)
12. E. Candès, J. Romberg, T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Trans. Inf. Theory **52**(2), 489–509 (2006)
13. E. Candès, J. Romberg, T. Tao, Stable signal recovery from incomplete and inaccurate measurements. Comm. Pure Appl. Math. **59**(8), 1207–1223 (2006)
14. N. Christianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods* (Cambridge University Press, Cambridge, England, 2000)
15. S. Dasgupta, A. Gupta, An elementary proof of a theorem of Johnson and Lindenstrauss. Random Struct. Algorithms **22**(1), 60–65 (2003)
16. D. Donoho, Compressed sensing. IEEE Trans. Inf. Theory **52**(4), 1289–1306 (2006)
17. S. Dirksen, A. Stollenwerk, Fast binary embeddings with gaussian circulant matrices: improved bounds. arXiv preprint arXiv:1608.06498 (2016)
18. D. Duncan, T. Strohmer, Classification of alzheimer's disease using unsupervised diffusion component analysis. Math. Biosci. Eng. **13**, 1119–1130 (2016)
19. J. Fang, Y. Shen, H. Li, Z. Ren, Sparse signal recovery from one-bit quantized data: an iterative reweighted algorithm. Signal Process. **102**, 201–206 (2014)
20. Y. Gong, S. Lazebnik, A. Gordo, F. Perronnin, Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. IEEE Trans. Pattern Anal. Mach. Intell. **35**(12), 2916–2929 (2013)
21. S. Gopi, P. Netrapalli, P. Jain, A.V. Nori, One-bit compressed sensing: provable support and vector recovery. ICML **3**, 154–162 (2013)
22. A. Gupta, R. Nowak, B. Recht, Sample complexity for 1-bit compressed sensing and sparse classification, in *2010 IEEE International Symposium on Information Theory Proceedings (ISIT)* (IEEE, 2010), pp. 1553–1557
23. A.D. Gordon, A review of hierarchical classification. J. R. Stat. Soc. Ser. A Gen. 119–137 (1987)

24. M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support vector machines. IEEE Intell. Syst. Appl. **13**(4), 18–28 (1998)
25. R. Higdon, N.L. Foster, R.A. Koeppe, C.S. DeCarli, W.J. Jagust, C.M. Clark, N.R. Barbas, S.E. Arnold, R.S. Turner, J.L. Heidebrink et al., A comparison of classification methods for differentiating fronto-temporal dementia from alzheimer's disease using FDG-PET imaging. Stat. Med. **23**(2), 315–326 (2004)
26. J. Hahn, S. Rosenkranz, A.M. Zoubir, Adaptive compressed classification for hyperspectral imagery, in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2014), pp. 1020–1024
27. B. Hunter, T. Strohmer, T.E. Simos, G. Psihoyios, C. Tsitouras, Compressive spectral clustering, in *AIP Conference Proceedings*, vol. 1281 (AIP, 2010), pp. 1720–1722
28. X. He, S. Yan, Y. Hu, P. Niyogi, H.-J. Zhang, Face recognition using laplacianfaces. IEEE Trans. Pattern Anal. Mach. Intell. **27**(3), 328–340 (2005)
29. L. Jacques, K. Degraux, C. De Vleeschouwer, Quantized iterative hard thresholding: bridging 1-bit and high-resolution quantized compressed sensing. arXiv preprint arXiv:1305.1786 (2013)
30. W. Johnson, J. Lindenstrauss, Extensions of Lipschitz mappings into a Hilbert space, in *Proceedings of Conference in Modern Analysis and Probability* (New Haven, CT, 1982)
31. L. Jacques, J. Laska, P. Boufounos, R. Baraniuk, Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. IEEE Trans. Inf. Theory **59**(4), 2082–2102 (2013)
32. T. Joachims, Text categorization with support vector machines: learning with many relevant features, in *Machine Learning: ECML-98* (1998), pp. 137–142
33. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems* (2012), pp. 1097–1105
34. K. Knudson, R. Saab, R. Ward, One-bit compressive sensing with norm estimation. IEEE Trans. Inf. Theory **62**(5), 2748–2758 (2016)
35. F. Krahmer, R. Ward, New and improved Johnson-Lindenstrauss embeddings via the restricted isometry property. SIAM J. Math. Anal. **43**(3), 1269–1281 (2011)
36. Y. LeCun, The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/
37. J.N. Laska, Z. Wen, W. Yin, R.G. Baraniuk, Trust, but verify: fast and accurate signal recovery from 1-bit compressive measurements. IEEE Trans. Signal Process. **59**(11), 5289–5301 (2011)
38. H. Li, Y. Yang, D. Chen, Z. Lin, Optimization algorithm inspired deep neural network structure design. arXiv preprint arXiv:1810.01638 (2018)
39. D. Molitor, D. Needell, Hierarchical classification using binary data (2018). Submitted
40. G.F. Montufar, R. Pascanu, K. Cho, Y. Bengio, On the number of linear regions of deep neural networks, in *Advances in Neural Information Processing Systems* (2014), pp. 2924–2932
41. D. Needell, R. Saab, T. Woolf, Simple classification using binary data. J. Mach. Learn. Res. (2017). Accepted
42. Y. Plan, R. Vershynin, One-bit compressed sensing by linear programming. Commun. Pure Appl. Math. **66**(8), 1275–1297 (2013)
43. Y. Plan, R. Vershynin, Robust 1-bit compressed sensing and sparse logistic regression: a convex programming approach. IEEE Trans. Inf. Theory **59**(1), 482–494 (2013)
44. Y. Plan, R. Vershynin, Dimension reduction by random hyperplane tessellations. Discret. Comput. Geom. **51**(2), 438–461 (2014)
45. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., Imagenet large scale visual recognition challenge. Int. J. Comput. Vision **115**(3), 211–252 (2015)
46. M. Rudelson, R. Vershynin, On sparse reconstruction from Fourier and Gaussian measurements. Comm. Pure Appl. Math. **61**(8), 1025–1171 (2008)
47. I. Steinwart, A. Christmann, *Support Vector Machines* (Springer Science & Business Media, 2008)
48. H.M. Shi, M. Case, X. Gu, S. Tu, D. Needell, Methods for quantized compressed sensing, in *Proceedings of Information Theory and Applications (ITA)* (2016)

49. C.N. Silla, A.A. Freitas, A survey of hierarchical classification across different application domains. Data Min. Knowl. Discov. **22**(1–2), 31–72 (2011)
50. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1–9
51. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
52. J. Weston, C. Watkins, Multi-class support vector machines. Technical Report (Citeseer, 1998)
53. X. Yi, C. Caravans, E. Price, Binary embedding: fundamental limits and fast algorithm (2015)
54. F.X. Yu, S. Kumar, Y. Gong, S.-F. Chang, Circulant binary embedding, in *International Conference on Machine Learning*, vol. 6 (2014), p. 7
55. M. Yan, Y. Yang, S. Osher, Robust 1-bit compressive sensing using adaptive outlier pursuit. IEEE Trans. Signal Process. **60**(7), 3868–3875 (2012)
56. Q.-S. Zhang, S.-C. Zhu, Visual interpretability for deep learning: a survey. Front. Inf. Technol. Electron. Eng. **19**(1), 27–39 (2018)