Deep Models and Artificial Intelligence for Military Applications:
AAAI Technical Report FS-17-03

Simple Object Classification Using Binary Data

Deanna Needell, Rayan Saab, Tina Woolf

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 and propose a framework with low computation and resource costs. We illustrate the utility of the proposed approach through stylized and realistic numerical experiments, including military classification problems like facial and object recognition. We hope that our framework will serve as a foundation for studying similar types of approaches.

DN acknowledges support from the Alfred P. Sloan Foundation and NSF CAREER #1348721; RS from the NSF under DMS-1517204.

1 Introduction

Our focus is on data classification problems in which only a *binary* representation of the data is available or desired. 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. 2013; Laska et al. 2011) and efficient storage; such benefits have contributed to the success, for example, of 1-bit Sigma-Delta converters (Aziz, Sorensen, and Vn der Spiegel 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 (see, e.g., (Boufounos and Baraniuk 2008; Hunter et al. 2010; Gupta, Nowak, and Recht 2010; Hahn, Rosenkranz, and Zoubir 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 $b = (b_1, \; \cdots \; , b_p) \in \{1, \ldots, G\}^p$, 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 take A to have independent identically distributed standard Gaussian entries. 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 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, and illustrate its promise on synthetic and real data including imaging recognition.

1.2 Related 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 these research areas, any review of prior work here must necessarily be non-exhaustive. Thus, here we very briefly discuss related prior work, highlighting connections to our work but also stressing the distinctions.

Support vector machines (SVM) (see, e.g., Christianini and Shawe-Taylor [2000]; Hearst et al.[1998]; Andrew [2000]; Joachims [1998]; Steinwart and Christmann [2008]) have become popular in machine learning, and are often used for classification. Although related, 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. Since the original work of Johnson and Lindenstrauss (Johnson and Lindenstrauss 1982), much work on Johnson-Lindenstrauss embeddings has focused on randomized embeddings where the matrix associated with the linear embedding is drawn from an appropriate random distribution (see, e.g., Ailon and Chazelle [2006]; Achlioptas [2003]; Dasgupta and Gupta [2003]; Krahmer and Ward [2011]). 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 under the assumption of data sparsity (Candès, Romberg, and Tao 2006b), (Candès, Romberg, and Tao 2006a), (Donoho 2006).

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* (Plan and Vershynin 2013a; 2013b; Gopi et al. 2013; Jacques et al. 2013; Yan, Yang, and Osher 2012; Jacques, Degraux, and De Vleeschouwer 2013) and was introduced in (Boufounos and Baraniuk 2008). Since then, related work has been done on the construction of binary embeddings (embeddings into the binary cube, see e.g.,[Plan and Vershynin [2014]; Yu et al. [2014]; Gong et al. [2013]; Yi, Caravans, and Price [2015]; Choromanska et al. [2016]; Dirksen and Stollenwerk [2016]). Although the data we consider in this paper takes a similar one-bit form, the overall goal is different; rather than signal *reconstruction* and geometry preservation, our interest is data *classification*.

Deep Learning is an area of machine learning based on learning data representations using multiple levels of abstraction, or layers. Algorithms for such deep neural networks have recently obtained state of the art results for classification see e.g., (Krizhevsky, Sutskever, and Hinton 2012; Simonyan and Zisserman 2014; Szegedy et al. 2015; Russakovsky et al. 2015)). We consider deep learning and neural networks as motivation to our layered algorithm design. However, we are not tuning nor optimizing parameters as is typically done in deep learning, nor do our layers necessarily possess the structure typical in deep learning "architectures"; this makes our approach potentially simpler and easier to work with.

2 The Proposed Classification Algorithm

The training phase of our algorithm is detailed in Algorithm 1, where we suppose the training data $Q = \text{sign}(AX)$ and associated labels b are available. Indeed, the training algorithm proceeds in L "layers". In the ℓ-th layer, 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 layer, the rows of Q indexed by $\Lambda_{\ell,i}$ are used to form the $\ell \times p$ matrix $Q^{\Lambda_{\ell,i}} \in \{\pm 1\}^{\ell \times p}$, and the unique sign patterns $q \in \{\pm 1\}^{\ell}$ are extracted from the columns of $Q^{\Lambda_{\ell,i}}$. The number of unique sign patterns (i.e., distinct columns) in $Q^{\Lambda_{\ell,i}}$ is given by $T_{\ell,i} \in \mathbb{N}$.

For example, at the first layer the possible unique sign patterns are 1 and -1, describing which side of the selected hyperplane the training data points lie on; at the second layer the possible unique 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 layers. 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, $P_{g|t}$ 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 layer (i.e., the t-th sign pattern determined from the set selection $\Lambda_{\ell,i}$):

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

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. 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 $j = 1, \ldots, G$, 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 layer. 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.

Algorithm 1 Training

```
input: binary training data $Q$, training labels $b$, number of classes $G$, number of layers $L$

for $\ell$ from 1 to $L$, $i$ from 1 to $m$ do
    select:      Random $\Lambda_{\ell,i} \subset [m]$, $|\Lambda_{\ell,i}| = \ell$
    determine:   $T_{\ell,i} \in \mathbb{N}$ unique col. 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 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 layers L, the learned membership index values $r(\ell, i, t, g)$, the number of unique sign patterns $T_{\ell,i}$, and the set selections $\Lambda_{\ell,i}$ at each iteration of each layer 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 layer ℓ and set selection i, the sign pattern $q^{\Lambda_{\ell,i}}$ is determined and the index $t^\star \in [T_{\ell,i}]$ is identified corresponding to the sign patterns that were determined during training. For each class g, $\tilde{r}(g)$ is updated via $\tilde{r}(g) \leftarrow \tilde{r}(g) + r(\ell, i, t^\star, g)$. If it happens that the sign pattern for x does not match any sign pattern determined during training, no update to $\tilde{r}$ is performed. Finally, after scaling $\tilde{r}$ with respect to the number of layers and measurements, the largest entry of $\tilde{r}$ identifies how the estimated label $\widehat{b}_x$ of x is set. Note that this scaling does not actually affect the outcome of classification, we use it simply to ensure the quantity does not become unbounded.

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 $\ell$ 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 for
end for
scale: Set $\tilde{r}(g) = \frac{\tilde{r}(g)}{Lm}$ for $g = 1, \ldots, G$
classify: $\widehat{b}_x = \text{argmax}_{g \in \{1,\ldots,G\}} \tilde{r}(g)$
```

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.

In all of the experiments, 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

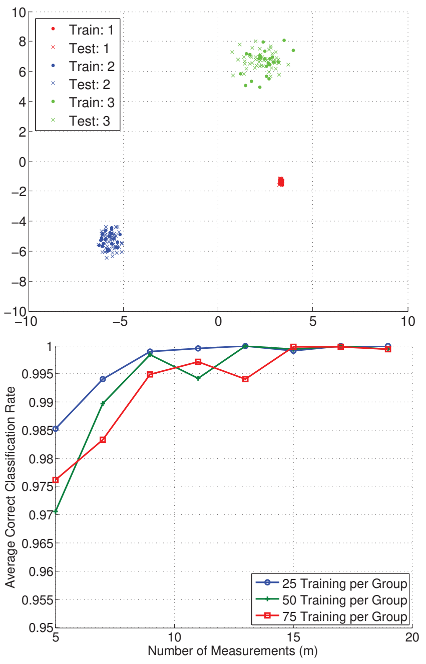


Figure 1: Synthetic classification experiment with three Gaussian clouds ($G = 3$), $L = 1$, $n = 2$, 50 test points per group, and 30 trials of randomly generating A. (Top) Example training and testing data setup. (Bottom) Average correct classification rate versus m and for the indicated number of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

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 by allowing for pre-processing of the data.

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 layer and 30 trials of generating A. We perform classification of 50 test points per group, and report the average correct classification rate over all trials. The right plot of Figure 1 shows that $m \geq 15$ results in nearly perfect classification.

Next, we present experiments where we again construct the classes as Gaussian clouds in $\mathbb{R}^2$, but utilize a non-uniformly alternating pattern around the origin with respect to the classes. In each case, we set the number of training data points for each class to be 25, 50, and 75. In Fig-

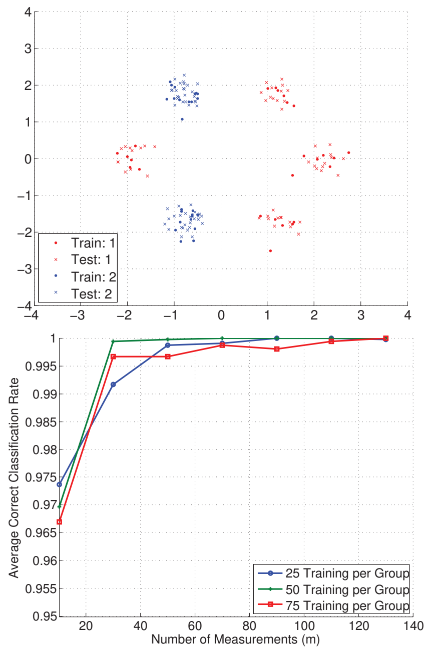


Figure 2: Synthetic classification experiment with six Gaussian clouds and two classes ($G = 2$), $L = 4$, $n = 2$, 50 test points per group, and 30 trials of randomly generating A. (Top) Example training and testing data setup. (Bottom) Average correct classification rate versus m and for the indicated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

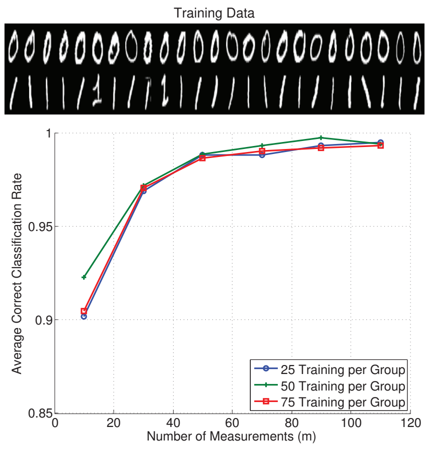


Figure 3: Classification experiment using the handwritten "0" and "1" digit images from the MNIST dataset, $L = 1$, $n = 28 \times 28 = 784$, 50 test points per group, and 30 trials of randomly generating A. (Top) Training data images when $p = 50$. (Bottom) Average correct classification rate versus m and for the indicated number of training points per class.

ure 2, we have two classes forming a total of six Gaussian clouds, and execute Algorithms 1 and 2 using four layers and $m \in \{10, 30, 50, 70, 90, 110, 130\}$. The classification accuracy increases for larger m, with nearly perfect classification

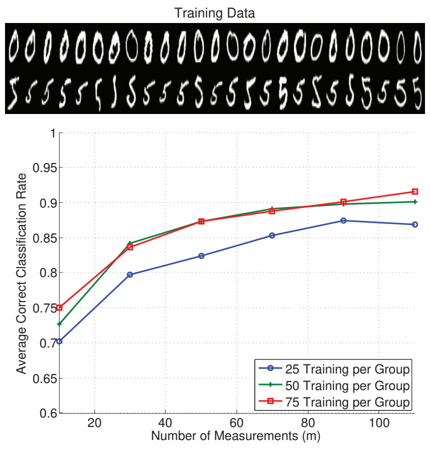


Figure 4: Classification experiment using the handwritten "0" and "5" digit images from the MNIST dataset, $L = 4$, $n = 28 \times 28 = 784$, 50 test points per group, and 30 trials of randomly generating A. (Top) Training data images when $p = 50$. (Bottom) Average correct classification rate versus m and for the indicated number of training points per class.

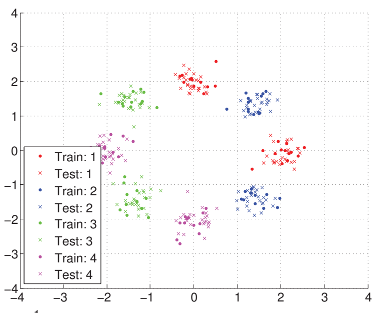
for the largest values of m selected.

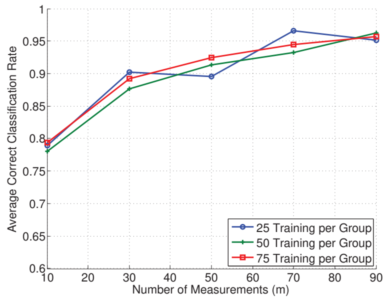
In the next experiment, we display the classification results of Algorithms 1 and 2 when using $m \in \{10, 30, 50, 70, 90\}$ and one through four layers, and see that adding layers can be beneficial for more complicated data geometries. In Figure 5, we have four classes forming a total of eight Gaussian clouds. 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.

3.2 Handwritten Digit Classification

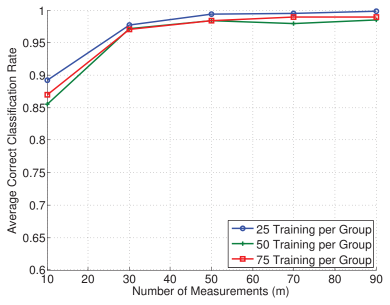
In this section, we apply Algorithms 1 and 2 to the MNIST (LeCun) dataset, which is a benchmark dataset of images of handwritten digits, each with 28×28 pixels. In total, the dataset has $60,000$ training examples and $10,000$ testing examples.

First, we apply Algorithms 1 and 2 when considering only two digit classes. Figure 3 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 layer. Figure 3 shows the results of a similar setup for the digits "0" and "5". In this experiment, we increased to four layers 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





(a) $L = 2$



(b) $L = 4$

Figure 5: Synthetic classification experiment with eight Gaussian clouds and four classes ($G = 4$), $n = 2$, 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) $L = 2$, (bottom) $L = 4$.

up than "0" and "1", which is understandable due to the more similar digit shape between "0" and "5".

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 layers and $m \in \{100, 200, 400, 600, 800\}$ are shown in Figure 6, 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 next experiment considers facial recognition using the extended YaleB dataset (Cai et al. 2007; Cai, He, and Han 2007; Cai et al. 2006; He et al. 2005). This dataset includes

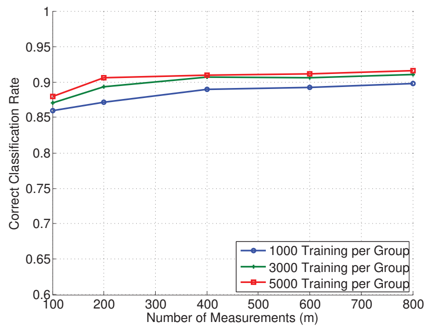


Figure 6: Correct classification rate versus m when using all ten (0-9) handwritten digits from the MNIST dataset, $L = 18$, $n = 28 \times 28 = 784$, 1,000, 3,000, and 5,000 training points per group, 800 test points per group (8,000 total), and a single instance of randomly generating A.

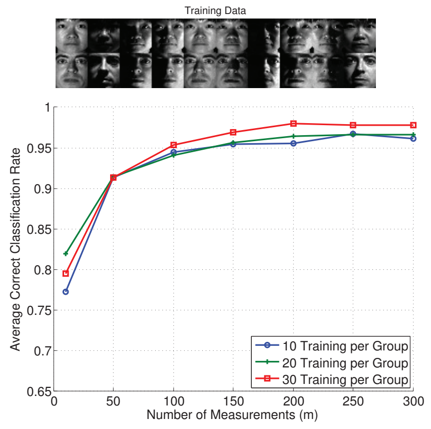
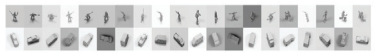


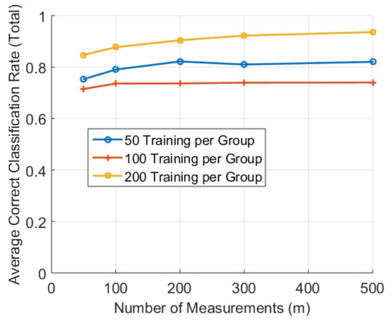
Figure 7: Classification experiment using two individuals from the extended YaleB dataset, $L = 4$, $n = 32 \times 32 = 1024$, 30 test points per group, and 30 trials of randomly generating A. (Top) Training data images when $p = 20$. (Bottom) Average correct classification rate versus m and for the indicated number of training points per class.

32×32 images of 38 individuals with roughly 64 near-frontal images under different illuminations per individual. We select two 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 layers 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 in Figure 7. Above 95% correct classification is achieved for $m \geq 150$ for each training set size included.

3.4 Object Recognition

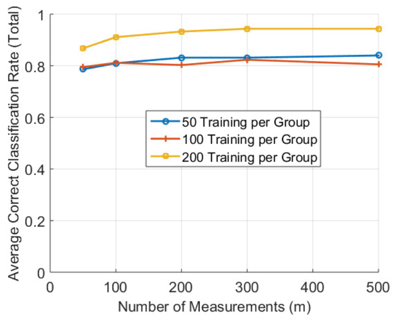
Our last experiment considers object recognition using the Norb dataset (LeCun, Huang, and Bottou 2004). It contains images ($n = 96 \times 96 = 9216$)) of 50 toys categorized as





Average Correct Classification Rate (Total)
Number of Measurements (m)
50 Training per Group
100 Training per Group
200 Training per Group

(a) $L = 1$



(b) $L = 5$

Figure 8: Classification experiment using two objects from the Norb dataset, $n = 96 \times 96 = 9216$, and 30 trials of randomly generating A. (Top) Example training data images. Average correct classification rate versus m and for (middle) $L = 1$ and (bottom) $L = 5$ and the indicated number of training points per class.

four-legged animals, human figures, airplanes, trucks, and cars. The objects were taken by two cameras under 6 lighting conditions, 9 elevations, and 18 azimuths. We select two categories from the dataset (human figure and truck), and randomly select images from that category to be included in the training and testing sets. We execute Algorithms 1 and 2 and display the results in Figure 8. Above 95% correct classification is again achieved for $m \geq 150$ for each training set size included.

4 Discussion and Conclusion

In this work, we have presented a supervised classification algorithm that operates on binary, or one-bit, data. We believe our framework is relevant to analyzing similar, layered-type algorithms. Future directions of this work include the use of dithers for more complicated data geometries, as well as a generalized theory for high dimensional data belonging to many classes and utilizing multiple layers within the algorithm.

References

Ailon, N., and Chazelle, B. 2006. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, 557–563. ACM.

Aziz, P. M.; Sorensen, H. V.; and Vn der Spiegel, J. 1996. An overview of sigma-delta converters. *IEEE signal processing magazine* 13(1):61–84.

Boufounos, P., and Baraniuk, R. 2008. 1-bit compressive sensing. In *Proc. IEEE Conf. Inform. Science and Systems (CISS)*.

Cai, D.; He, X.; Han, J.; and Zhang, H.-J. 2006. Orthogonal laplacianfaces for face recognition. *IEEE Transactions on Image Processing* 15(11):3608–3614.

Cai, D.; He, X.; Hu, Y.; Han, J.; and Huang, T. 2007. Learning a spatially smooth subspace for face recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition Machine Learning (CVPR'07)*.

Cai, D.; He, X.; and Han, J. 2007. Spectral regression for efficient regularized subspace learning. In *Proc. Int. Conf. Computer Vision (ICCV'07)*.

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

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

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

Choromanska, A.; Choromanski, K.; Bojarski, M.; Jebara, T.; Kumar, S.; and LeCun, Y. 2016. Binary embeddings with structured hashed projections. In *Proceedings of The 33rd International Conference on Machine Learning*, 344–353.

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

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

Fang, J.; Shen, Y.; Li, H.; and Ren, Z. 2014. Sparse signal recovery from one-bit quantized data: An iterative reweighted algorithm. *Signal Processing* 102:201–206.

Gong, Y.; Lazebnik, S.; Gordo, A.; and Perronnin, F. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(12):2916–2929.

Gopi, S.; Netrapalli, P.; Jain, P.; and Nori, A. V. 2013. One-bit compressed sensing: Provable support and vector recovery. In *ICML (3)*, 154–162.

Gupta, A.; Nowak, R.; and Recht, B. 2010. Sample complexity for 1-bit compressed sensing and sparse classification. In *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, 1553–1557. IEEE.

Hahn, J.; Rosenkranz, S.; and Zoubir, A. M. 2014. Adaptive compressed classification for hyperspectral imagery. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, 1020–1024. IEEE.

He, X.; Yan, S.; Hu, Y.; Niyogi, P.; and Zhang, H.-J. 2005. Face recognition using laplacianfaces. *IEEE Trans. Pattern Anal. Mach. Intelligence* 27(3):328–340.

Hunter, B.; Strohmer, T.; Simos, T. E.; Psihoyios, G.; and Tsitouras, C. 2010. Compressive spectral clustering. In *AIP Conference Proceedings*, volume 1281, 1720–1722. AIP.

Jacques, L.; Laska, J.; Boufounos, P.; and Baraniuk, R. 2013. Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *IEEE Trans. Inform. Theory* 59(4):2082–2102.

Jacques, L.; Degraux, K.; and De Vleeschouwer, C. 2013. Quantized iterative hard thresholding: Bridging 1-bit and high-resolution quantized compressed sensing. *arXiv preprint arXiv:1305.1786*.

Johnson, W., and Lindenstrauss, J. 1982. Extensions of Lipschitz mappings into a Hilbert space. In *Proc. Conf. Modern Anal. and Prob.*

Krahmer, F., and Ward, R. 2011. New and improved johnson–lindenstrauss embeddings via the restricted isometry property. *SIAM Journal on Mathematical Analysis* 43(3):1269–1281.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

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

LeCun, Y.; Huang, F. J.; and Bottou, L. 2004. Learning methods for generic object recognition with invariance to pose and lighting. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, II–104. IEEE.

LeCun, Y. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

Plan, Y., and Vershynin, R. 2013a. One-bit compressed sensing by linear programming. *Communications on Pure and Applied Mathematics* 66(8):1275–1297.

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

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

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115(3):211–252.

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

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9.

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

Yi, X.; Caravans, C.; and Price, E. 2015. Binary embedding: Fundamental limits and fast algorithm.

Yu, F. X.; Kumar, S.; Gong, Y.; and Chang, S.-F. 2014. Circulant binary embedding. In *International conference on machine learning*, volume 6, 7.