Hierarchical Classification Using Binary Data

Denali Molitor, Deanna Needell

■ *In classification problems, especially those that categorize data into a large number of classes, the classes often naturally follow a hierarchical structure. That is, some classes are likely to share similar structures and features. Those characteristics can be captured by considering a hierarchical relationship among the class labels. Motivated by a recent simple classification approach on binary data, we propose a variant that is tailored to efficient classification of hierarchical data. In certain settings, specifically, when some classes are significantly easier to identify than others, we showcase computational and accuracy advantages.*

We consider the problem of classification, in which one is given a set of labeled data used for training and from that data wishes to accurately assign labels to new unlabeled data. In the general problem, the class labels themselves have no relation to one another; however, data can often be organized in a hierarchical way. 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, say, or humans and animals. The data could then be further subdivided into classes of various animal types, and so on. This structure can be visualized as a tree, where the children of each node correspond 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. One option, of course, would be to simply use generic classification schemes to classify the data using the leaf labels only. Hierarchical classification, however, makes use of information and structure between groups in classifying the data (Gordon 1987; Silla and Freitas 2011). Extensions of popular classification methods such as the support vector machine to the hierarchical setting are not straightforward, and such approaches often decompose the problem into many subproblems, leading to higher computational complexities (Cheong, Oh, and Lee 2004; Weston and Watkins 1998).

Recently, Needell, Saab, and Woolf (2017) proposed a simple classification scheme (SCB) that uses only binary representations of data to perform classification; such representations arise naturally or are particularly efficient in many applications (Fang et al. 2014; Jacques et al. 2011; Aziz, Sorensen, and Van der Spiegel 1996; Bottou and Bousquet 2011; Gupta, Nowak, and Recht 2010). Here, we show that this method lends itself well to performing hierarchical classification and, in particular, using the hierarchical structure to improve computational efficiency. The classification method uses position of data relative to random hyperplanes to predict in which class a point is most likely to belong. Needell, Saab, and Woolf (2017) demonstrated that for more complex data, using combinations of hyperplanes enables one to make more accurate predictions. However, the computation required to make a prediction scales exponentially in the number of hyperplane combinations used. Fortunately, the method is highly adjustable, and for data that are likely to be more or less difficult to classify, one can adjust the number of these hyperplane combinations. Such a 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.

Underlying Classification Algorithm

In this section, we briefly review the SCB classification algorithm that motivates our approach; Needell, Saab, and Woolf (2017) provide mathematical details. Suppose X is our data matrix and that each data point (column) of X is associated with one of G class labels. We are given binary measurements of the form $Q = \text{sign}(AX)$, where A is a wide random matrix (for example, Gaussian). The rows of A correspond to (random) hyperplanes, and thus $Q_{i,j}$ simply captures on which side of the ith hyperplane the jth data point lies.

Let us build some intuition for the approach. Consider the two-dimensional data X shown in the top plot of figure 1, consisting of three labeled classes (green, blue, red). Consider the four hyperplanes shown in the same plot, and suppose we had access only to the binary data $Q = \text{sign}(AX)$, where A contains the normals to each hyperplane as its rows. For the new test point x (which by visual inspection should be labeled blue) and its binary data $q = \text{sign}(Ax)$, one could 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 (that is, lies on the same side) as the blue and green classes. For the black hyperplane, x matches only the blue class, and so on. In this example, x will clearly match the blue class most often, and we could assign it that label correctly. However, next consider the more complex geometry given in the bottom plot, where the data consist of only two classes (red and blue), but these are now no longer linearly separable. This same strategy will no longer be accurate for the test point x. However, now instead of single hyperplanes, consider hyperplane pairs, and ask which class label x most often matches (note that in this context, by matches we now mean that points lie in the same cone into which the hyperplanes divide the space). For example, for the pair of hyperplanes colored orange and green, x matches both red and blue points, whereas for the pair of hyperplanes colored orange and purple, x matches only the blue class. One could now cycle through all pairs and again ask which class x matches most often. For complex data, we could aggregate such information across various levels, where at level ℓ we consider ℓ-tuples of hyperplanes in this way.

Concretely, we can aggregate this information across levels by using a membership index parameter that computes the fraction of points with a given sign pattern that correspond to a particular label [we also use a balancing term to handle large deviations in class sizes; see Needell, Saab, and Woolf (2017) for details]. Intuitively, the membership values indicate how likely a point with a particular sign pattern will lie in a particular class, given information from an ℓ-tuple of hyperplanes. Then, to classify a test point x with binary measurements $q = \text{sign}(Ax)$, the membership values are simply summed over all measurements m and levels ℓ. This process gives a vector $\tilde{r} \in \mathbb{R}^G$ that indicates the likelihood that the point belongs to each class g. The label assigned to x is simply the class g corresponding to the largest value of $\tilde{r}$. The SCB classification method is described in more detail by Needell, Saab, and Woolf (2017).

Computational Complexity

We aim to reduce the computational cost of classification via SCB by making use of a hierarchical structure of data. We do not consider the cost of calculating $q = \text{sign}(Ax)$ in our analysis, as we assume that the algorithm is provided these binary measurements. Additionally, one may not have access to the underlying vector x and knows only the binary measurements q.

Identifying the sign pattern of a test point with respect to each hyperplane tuple and finding the corresponding membership value is one of the most costly components of testing. The number of flops required for this step depends on the way in which the unique sign patterns from the training data are stored. We consider an implementation in which one uses a lookup table for all possible sign patterns and thus this step incurs a constant lookup time. The suggested hierarchical strategy leads to computational savings irrespective of this implementation choice. See Molitor and Needell (2018) for more detailed flop counts.

Adjustment for Hierarchical Classification

We now describe our proposed adjustment for handling hierarchical classification, where the labels possess some sort of tree structure. The classification scheme described above and by Needell, Saab, and Woolf (2017) has the property that more levels (higher

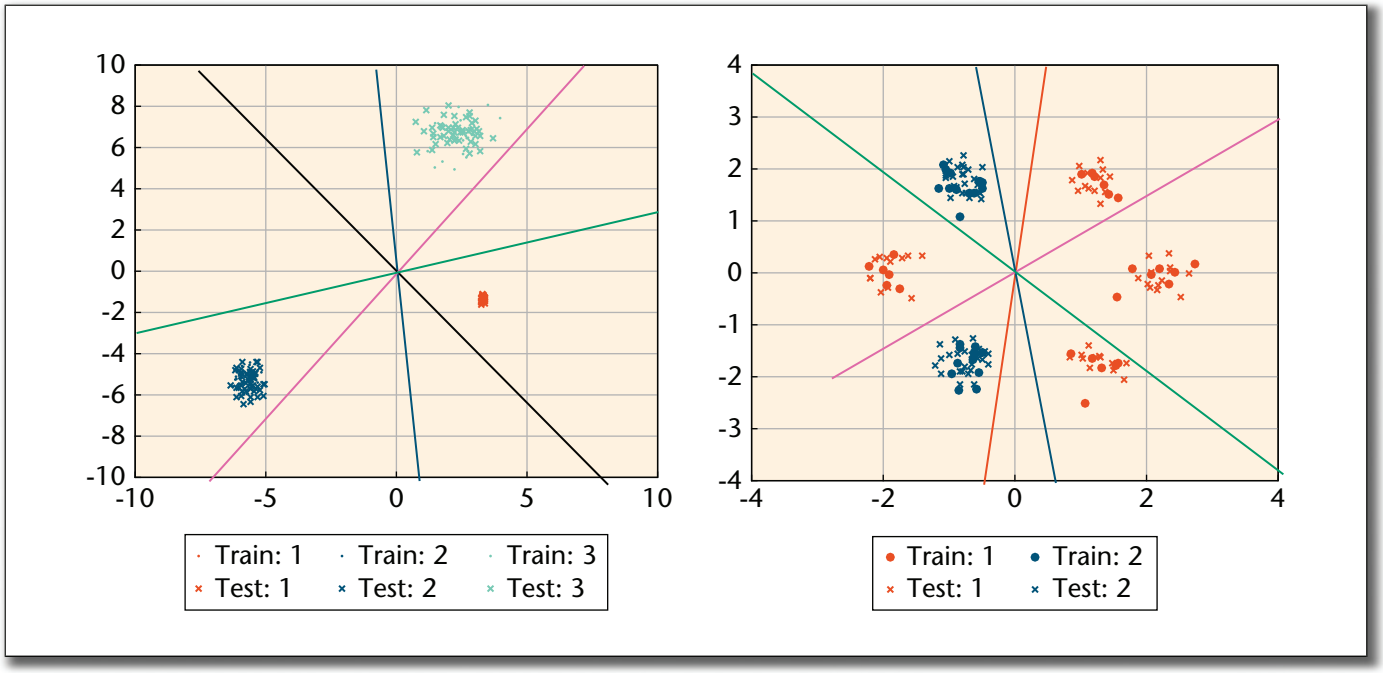

Figure 1. Two Motivating Examples for the Classification Method.

L) are needed to accurately classify more complex data. Thus, 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 these groups of classes using only the required number of levels for sufficient accuracy. This strategy leads to computational savings without sacrificing accuracy when some classes are more easily discerned from the others.

For illustration, consider a simple example in which we have three classes, g_1, g_2, g_3. Suppose that L_1 levels are necessary to classify data belonging to g_1, but L_2 levels are required to differentiate between classes g_2 and g_3 where $L_2 > L_1$. We can perform binary classification between g_1 and $\{g_2, g_3\}$ using L_1 levels, followed by classification between g_2 and g_3 using L_2 levels. These classifications can be organized as a tree with nodes H_1 and H_2 as shown in figure 2.

To further discern between points predicted to belong to g_2 or g_3, we can use the same measurements (or random hyperplanes) as used in the first classification. The overhead cost to carrying out two classifications instead of one is quite limited overall. For classifications in which some classes require fewer levels to predict, this hierarchical structure can lead to significant computational savings, as shown in the experimental results that follow. The magnitude of the computational savings is highly dependent on the distribution of the testing data, however, as we reduce computational costs only for those points predicted to be in one of the classes that is easier to discern, that is, requires fewer levels. See Molitor and Needell (2018) for details.

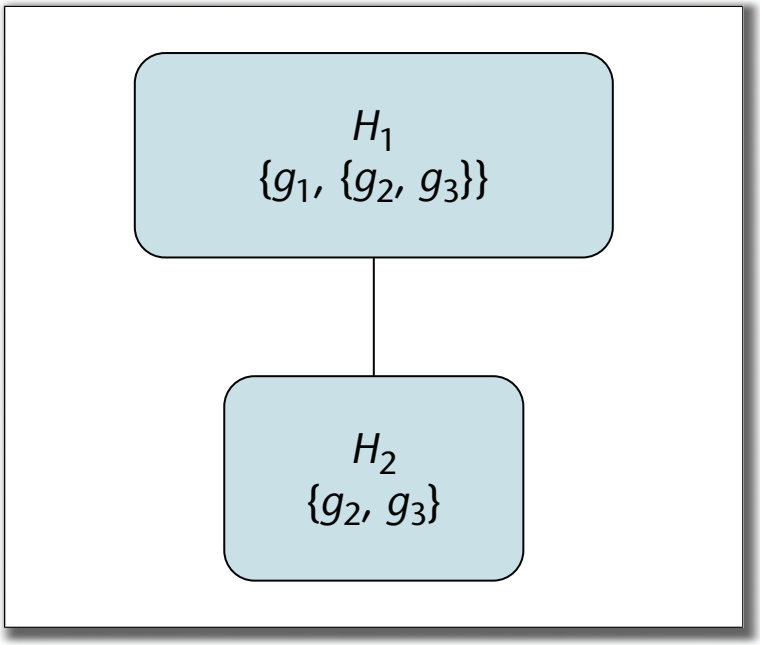

Figure 2. Hierarchical Classification Tree for a Simple Three-Class Example.

In this example, differentiating g_1 is significantly easier than differentiating g_2 and g_3.

This hierarchical classification strategy naturally generalizes to incorporate more complicated and deeper hierarchical structures in which the classifications can be structured as a tree. See figure 3 for an example. To maximize computational gains, however,

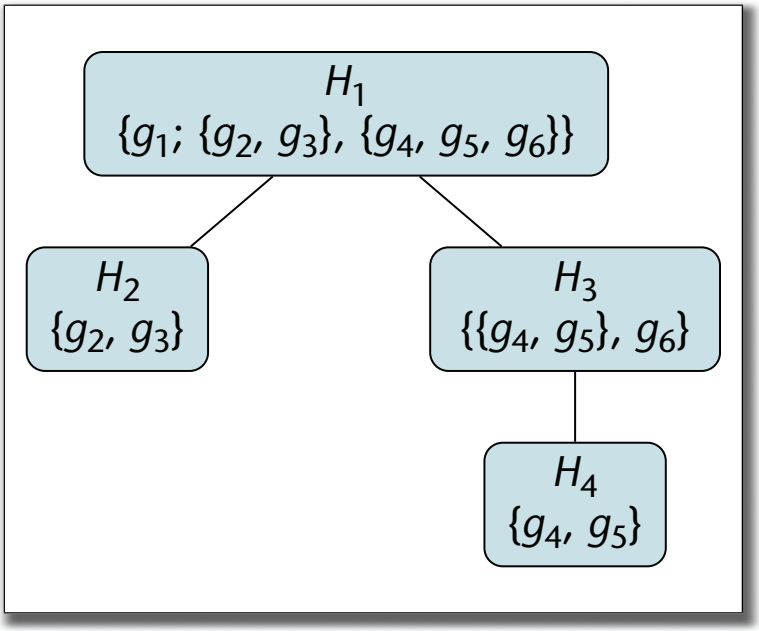

Figure 3. Example Hierarchical Classification Tree.

A classifier would be trained at each node, H_c, to classify data among the sets.

we would like the tree to be imbalanced in terms of the maximum number of levels required for sufficient classification accuracy along different paths of the tree. Such an imbalance arises naturally in many applications. For example, consider brain imaging and the problem of detecting brain abnormalities including tumors and dementia; tumor detection is a fairly easy learning problem, whereas classifying the various types of dementia remains very challenging (Duncan and Strohmer 2016; Higdon et al. 2004).

Determining Class Hierarchies

When the number of classes is large, reorganizing a flat multiclass classification problem into hierarchical (binary) classifications can be used as a general strategy to reduce the computation required for testing (Griffin and Perona 2008). Our proposed strategy need not be applied only in settings where the data follows or are presented within the context of a clear hierarchical structure. A variety of previous works studied ways to detect structure among classes and to use this information to construct an informed hierarchy of the classes (Griffin and Perona 2008; Godbole, Sarawagi, and Chakrabarti 2002; Silva-Palacios, Ferri, and Ramírez-Quintana 2017; Li, Zhu, and Ogihara 2007; Zupan et al. 1999). These strategies generally aim to group classes that are deemed similar by some measure, to reduce the number of misclassifications that occur high in the tree. For example, some work suggests constructing a hierarchy based on the confusion matrix of the flat multiclass classification problem (Griffin and Perona 2008; Godbole, Sarawagi, and Chakrabarti 2002; Silva-Palacios, Ferri, and Ramírez-Quintana 2017). Preferentially constructing class hierarchies that are imbalanced in terms of ease of classification along different paths will also largely affect the computational savings achieved by our proposed hierarchical classification method. We save details on how one might achieve this for future work.

Experimental Results

In the following experiments, we test the computational gains achieved by the proposed hierarchical classification strategy compared with direct classification into each individual group via SCB from Needell, Saab, and Woolf (2017).

Two-Dimensional Synthetic Data

We first test the computational gains achieved by the proposed hierarchical classification strategy on the two-dimensional data shown in figure 4. Each color represents a different class, and there are six classes in total. The red and yellow clusters each contain 200 training and testing points, and the remaining four classes, green, black, blue and cyan, contain 100 training and testing points each. The distribution of testing points among the classes will have a significant effect on the computation needed for testing in the hierarchical case. We expect classifying points from the red and yellow classes to be easier and to require fewer levels than correctly classifying points as green, black, blue, or cyan.

To take advantage of this structure in the data, we first predict whether a testing point is red or yellow versus green, black, blue, or cyan by using only one level. If the test point is predicted to be red or yellow, we then discern between these two classes again by using only a single level. If the test point is predicted to be green, black, blue, or cyan, we then predict among these classes by using varying numbers of levels. Accuracies and testing flops for the hierarchical classification strategy versus SCB are shown in figure 5 using $m = 50$. We see a significant reduction in computational cost using the hierarchical strategy without sacrificing accuracy.

Three-Dimensional Synthetic Data

We test the hierarchical classification strategy and SCB on three-dimensional synthetic data as given in figure 6. Each color represents a different class. Again, we expect the four Gaussian clusters to require fewer levels for sufficiently accurate classification than do the arcs. The training data are distributed so that there is an equal number of training and testing points in the Gaussian clusters and arcs. Specifically we have 100 training and testing points in each arc and 200 training and testing points in each Gaussian cluster.

Using a strategy similar to that used in the two-dimensional experiment, we first build a classifier to predict whether a point belongs to one of the arcs or

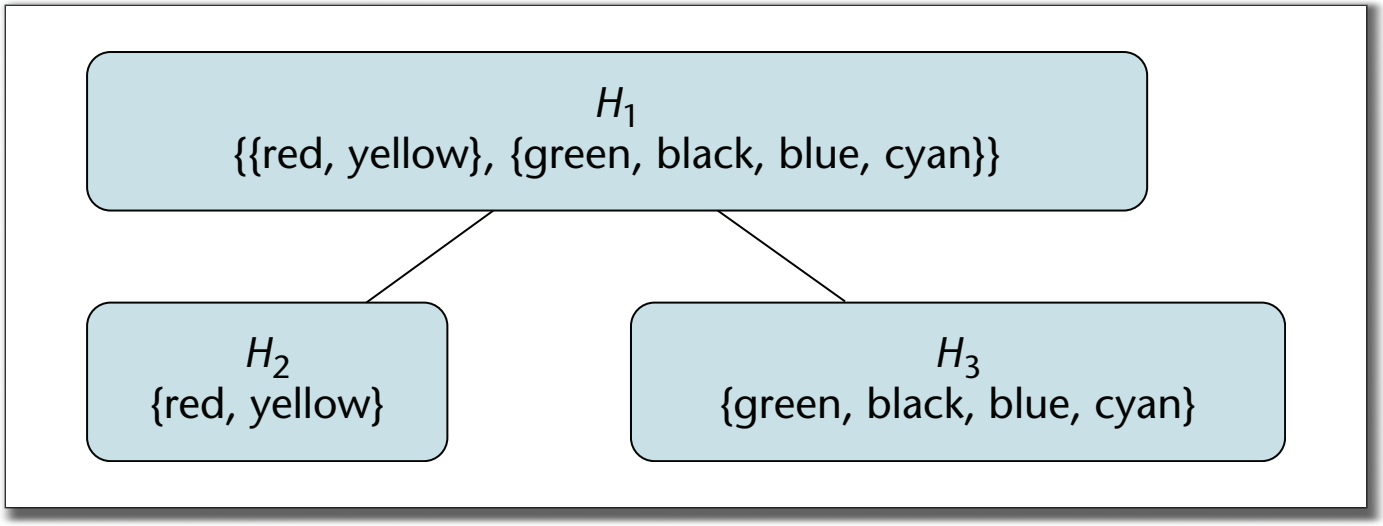

Figure 4. Hierarchical Classification Tree for Two-Dimensional Synthetic Data in Figure 5.

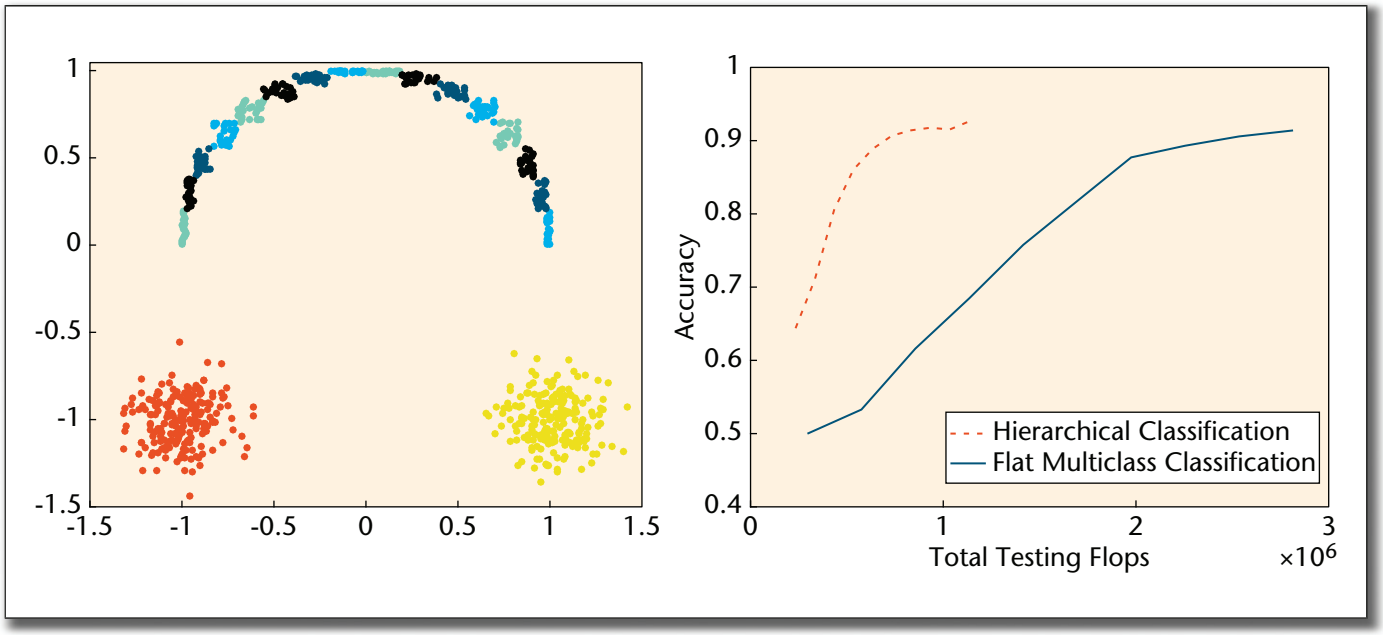

Figure 5. Accuracies and Total Testing Flops.

For the data distributed as given in the upper plot, where each color represents a different class, we classify testing data by either SCB or our proposed hierarchical classification strategy, in which the first classification discerns between red or yellow versus green, black, blue, or cyan. Accuracy and testing flops required are given in the lower plot using $m = 50$. Results are averaged over 50 trials.

one of the Gaussian clusters using only a single level. If a data point is predicted to be in one of the Gaussian clusters, we then use a single level again to predict to which of the clusters it belongs. If a data point is predicted to be in one of the arcs, we use more levels to perform the subsequent classification to discern between the arcs. We test the accuracy and computation required for using a variety of levels in this second classification. As in the two-dimensional experiment, we again see a reduction in the computational cost of testing without sacrificing accuracy.

MNIST

We demonstrate that our hierarchical strategy can lead to computational savings on the MNIST data set of handwritten digits,[1] although MNIST is not

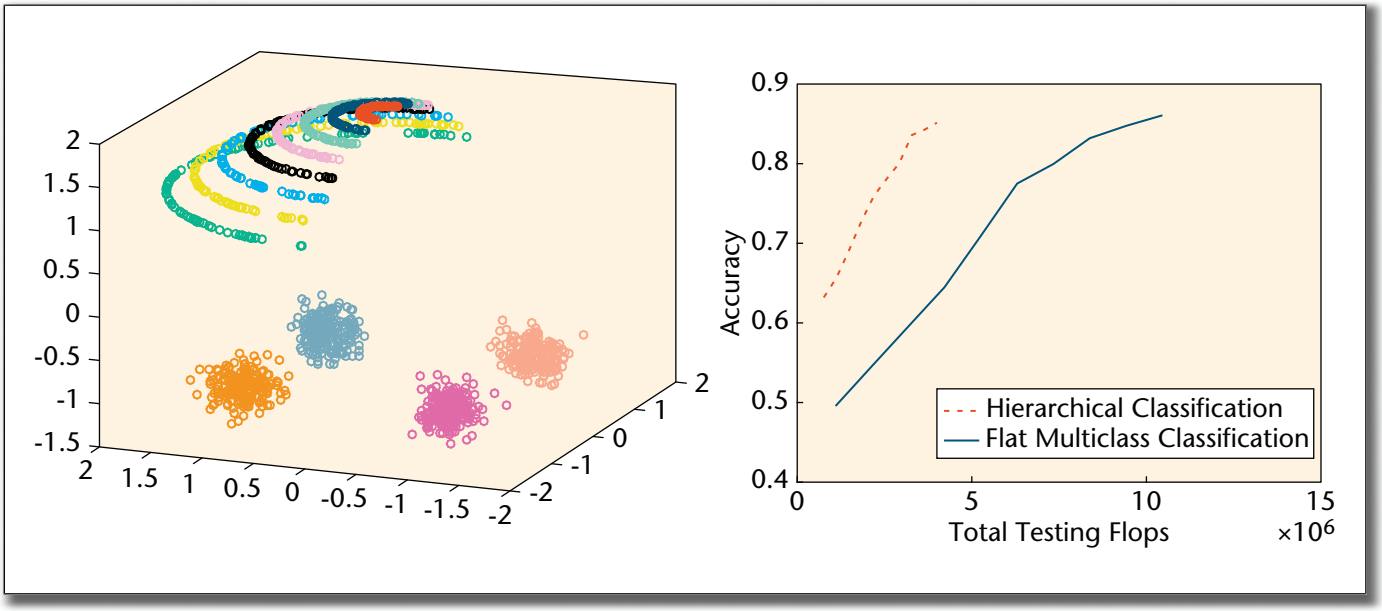

Figure 6. Three-Dimensional Data.

For the data distributed as given in the upper plot, where each color represents a different class, we classify testing data by either SCB or our proposed hierarchical classification strategy, in which the first classification discerns between red or yellow versus green, black, blue, or cyan. Accuracy and testing flops required are given in the lower plot using $m = 50$. Results are averaged over 50 trials.

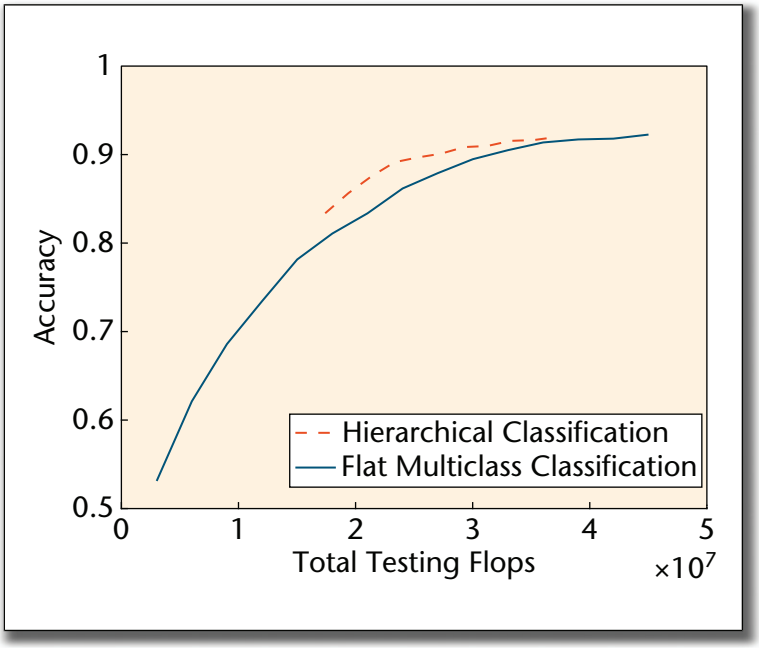

Figure 7. MNIST Data.

Accuracy and testing flops required for SCB versus our proposed hierarchical classification strategy in classifying digits 1 through 5 in the MNIST data set are given for $m = 500$. Results are averaged over 10 trials.

inherently hierarchical. Consider the digits 1 through 5. Intuitively and in practice, digit 1 tends to be easier to classify correctly than the other digits. For example, if we apply SCB to classify digits 1 through 5 using 1000 training points for each class and 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 digits 1 through 5 was 89.2% (the accuracy for classifying digits 2 through 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.

We induce a hierarchical structure by first classifying 1s versus not 1s, followed by classification into digits 2, 3, 4, and 5 for those test points that were predicted to not be 1s in the first classification. When training the first classifier, we downsample the training data for digits 2 through 5 so that we have an equal number of training data points for 1s and not 1s. We found that this adjustment improved the accuracy of the first classification. Five levels are used for the first classification into 1s versus not 1s, and a varying number of levels (5 to 10) are used for the subsequent classification. We again see a reduction in the total testing flops required to achieve a given accuracy. Here, we use an equal number of test points for each digit and thus get computational savings for approximately one-fifth of the test points, specifically, for all test points that are predicted to be 1s. If we had a much higher proportion of 1s than the other digits, then we would expect the computational savings to be even more significant. These results are depicted in figure 7. Additionally, because this tree is fairly shallow, as expected the improvements are mild, and we would expect more significant

improvement for real data that has a larger and more imbalanced tree structure, as in the other experiments.

Conclusion

We have demonstrated that the classification algorithm proposed by Needell, Saab, and Woolf (2017) can be readily adapted to classify data in a hierarchical way that improves computational efficiency. We achieve this by using fewer levels to classify data points predicted to be from classes that are more readily identifiable. We could potentially further reduce computational costs for easier-to-classify data by reducing the number of measurements m in those cases as well. Theoretical guarantees as well as modifications that alleviate error propagation down the tree are important directions for future work.

Acknowledgments

The authors were partially supported by NSF CAREER Grant 1348721 and NSF BIGDATA Grant 1740325.

Note

1. yann.lecun.com/exdb/mnist

References

Aziz, P. M.; Sorensen, H. V.; and Van der Spiegel, J. 1996. An Overview of Sigma-Delta Converters. *IEEE Signal Processing Magazine* 13(1): 61–84. doi.org/10.1109/79.482138

Bottou, L., and Bousquet, O. 2011. The Tradeoffs of Large-Scale Learning. In *Optimization for Machine Learning*, edited by S. Sra, S. Nowozin, and S. J. Wright, 351–68. Cambridge, MA: MIT Press.

Cheong, S.; Oh, S. H.; and Lee, S.-Y. 2004. Support Vector Machines with Binary Tree Architecture for Multi-class Classification. *Neural Information Processing-Letters and Reviews* 2(3): 47–51.

Duncan, D., and Strohmer T. 2016. Classification of Alzheimer's Disease Using Unsupervised Diffusion Component Analysis. *Mathematical Biosciences and Engineering* 13(6): 1119–30. doi.org/10.3934/mbe.2016033

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: 201–6. doi.org/10.1016/j.sigpro.2014.03.026

Godbole, S.; Sarawagi, S.; and Chakrabarti, S. 2002. Scaling Multi-Class Support Vector Machines Using Inter-Class Confusion. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 513–8. New York: Association for Computing Machinery. doi.org/10.1145/775047.775122

Gordon, A. D. 1987. A Review of Hierarchical Classification. *Journal of the Royal Statistical Society. Series A (General)* 150(2): 119–37. doi.org/10.2307/2981629

Griffin, G., and Perona P. 2008. Learning and Using Taxonomies for Fast Visual Categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. New York: IEEE. doi.org/10.1109/CVPR.2008.4587410

Gupta, A.; Nowak, R.; and Recht, B. 2010. Sample Complexity for 1-bit Compressed Sensing and Sparse Classification. In *International Symposium on Information Theory*. New York: IEEE. doi.org/10.1109/ISIT.2010.5513510

Higdon, R.; Foster, N. L.; Koeppe, R. A.; DeCarli, C. S.; Jagust, W. J.; Clark, C. M.; Barbas, N. R.; Arnold, S. E.; Turner, R. S.; and Heidebrink, J. L.. 2004. A Comparison of Classification Methods for Differentiating Fronto-Temporal Dementia from Alzheimer's Disease Using FDG-PET Imaging. *Statistics in Medicine* 23(2): 315–26. doi.org/10.1002/sim.1719

Jacques, L.; Laska, J. N.; Boufounos, P. T.; and Baraniuk R. G. 2013. Robust 1-Bit Compressive Sensing via Binary Stable Embeddings of Sparse Vectors. *IEEE Transactions on Information Theory* 59(4): 2082–102. doi.org/10.1109/TIT.2012.2234823

Li, T.; Zhu, S.; and Ogihara, M. 2007. Hierarchical Document Classification Using Automatically Generated Hierarchy. *Journal of Intelligent Information Systems* 29(2): 211–30. doi.org/10.1007/s10844-006-0019-7

Molitor, D., and Needell, D. 2018. A Simple Approach to Hierarchical Classification. In *Proceedings of the International Traveling Workshop on Interactions between Low-Capacity Data Models and Sensing Techniques (iTWIST)*. arXiv preprint. arxiv: 1812.00648 [cs.CL]. Ithaca, NY: Cornell University Library.

Needell, D.; Saab, R.; and Woolf, T. 2018. Simple Classification Using Binary Data. *Journal of Machine Learning Research* 19(61): 1–30.

Silla, C. N., and Freitas, A. A. 2011. A Survey of Hierarchical Classification across Different Application Domains. *Data Mining and Knowledge Discovery* 22(1-2): 31–72. doi.org/10.1007/s10618-010-0175-9

Silva-Palacios, D.; Ferri, C.; and Ramírez-Quintana, M. J. 2017. Improving Performance of Multiclass Classification by Inducing Class Hierarchies. *Procedia Computer Science* 108: 1692–701. doi.org/10.1016/j.procs.2017.05.218

Weston, J., and Watkins, C. 1998. Multi-Class Support Vector Machines. Technical Report CSD-TR-98-04. Egham, UK: Royal Holloway, University of London, Department of Computer Science.

Zupan, B.; Bohanec, M.; Demšar, J.; and Bratko, I. 1999. Learning by Discovering Concept Hierarchies. *Artificial Intelligence* 109(1-2): 211–42. doi.org/10.1016/S0004-3702(99)00008-9

Denali Molitor is a PhD candidate in the Department of Mathematics at the University of California, Los Angeles. She is broadly interested in developing and analyzing machine learning algorithms and has recently been studying sketch and project methods for solving large linear systems, data completion for structured data, and classification methods using binary data.

Deanna Needell is a full professor in the Department of Mathematics at the University of California, Los Angeles. Her research interests include compressed sending, randomized algorithms, functional analysis, computational mathematics, probability, and statistics. She earned her PhD in mathematics from the University of California, Davis, in 2009 before working as a postdoctoral fellow at Stanford University, then as an assistant and later associate professor in the Mathematics Department at Claremont McKenna College in Claremont, California. She has earned many awards, including the IEEE Best Young Author award, an Alfred P. Sloan fellowship, NSF CAREER and NSF BIGDATA awards, and the IMA Prize in Applied Mathematics. She is associate editor of *IEEE Signal Processing Letters, Linear Algebra and Its Applications, SIAM Journal on Imaging Sciences,* and *Transactions in Mathematics and Its Applications* and serves on the organizing committees for sessions of the Society for Industrial and Applied Mathematics and the Association for Women in Mathematics.