ECE 236A Project

Zhaomeng Chen

November 28th, 2023

Task 1: Supervised Classification

In this task, we assume that all of our data has been correctly labeled. This gives us the unique ability to simply "combine" multiple linear classifiers, one for each pair of classes, and use all of them to predict the labels of new data. This means that for K classes, we have $\frac{K(K-1)}{2}$ binary classifiers. These classifiers effectively divide the $\mathbb{R}^m$ space into K polyhedras, one for each class. This allows us to easily label new datapoints by simply looking at which polyhedra they are in.

The binary linear classifiers are formulated using SVMs (Support Vector Machines) similar to the ones seen in class. SVM's were chosen because they allow for us to find a better, more "generalized" hyperplane. However, one weakness of SVMs seen in class is that they can only classify into 2 classes. The design of using many binary classifiers according to each pair of classes is essentially a workaround this weakness. However, we do give up a lot of speed in our algorithm since we are training $\frac{K(K-1)}{2}$ binary classifiers instead of one big classifier.

For an individual binary classifier, we have the following LP:

$$\text{minimize } \sum_{i=1}^{N} \max\{0, 1 - s_i(\vec{a_i}^T \vec{x_i} + b) + \lambda ||\vec{a}||_2^2\}$$

where N is the number of datapoints, $\vec{x_i}, b$ are variables, $\vec{a_i}$ our training data, s_i the labels of the training data, and λ the regularization parameter. Since there are K classes while this is a binary classifier between classes $C_1, C_2 \in [K], C_1 \neq C_2$ and there are K classes, we let

$$s_i = \begin{cases} 1 & \vec{a_i} \in \text{class } C_1 \\ -1 & \vec{a_i} \in \text{class } C_2 \\ 0 & \text{else} \end{cases}$$

The cases for 1 and -1 is standard for SVMs. The nonstandard case is when $s_i = 0$, which occurs only when the datapoint is not in the classes we currently care about. Another thing to note is that the ℓ_2 norm is not linear, but it can be estimated using the ℓ_1 norm thanks to equivalence of norms, hence, we will allow it in our LP formulation (if you really want to be percise, we replace $||\cdot||_2$ with $\sqrt{m}||\cdot||_1$).

Task 2: Unsupervised Classification

In this task, we no longer have data labels. We will work around this by exploiting the fact that elements of the same class are "similar" to each other, and hence must be "similar" to the mean of their class.

We begin by choosing K random datapoints that are not too similar to each other (cosine similarity $> C$ between any two datapoints, where $C \in [-1, 1]$ is a parameter to be tuned). We will use these K points to be our initial means $\vec{M_1}, \ldots, \vec{M_K}$. Next, we will use the following ILP to label our datapoints:

$$\text{minimize } \sum_{i=1}^{N} \sum_{j=1}^{K} x_{ij} ||\vec{a_i} - \vec{M_j}||_2$$

$$\text{such that } x_{ij} \in \{0, 1\}, \sum_{j=1}^{K} x_{ij} = 1$$

Where N is the size of our dataset, x_{ij} are variables, $\vec{a_i}$ are our datapoints. This ILP essentially tries to associate each datapoint with the M_j that is closest to it according to the ℓ_2 norm. Of course, the constraints essentially force each datapoint to be associated with exactly one M_j at a time. Note that the objective function can be turned into a ILP for the same reason as in Task 1.

After we have labeled all the datapoints, we can find new means $M'_1, \ldots, M'_j$ and find the distances $D_i = ||M'_i - M_i||_2$. By continually repeating the ILP with our new means, eventually the means will converge to the "true means" (we know when we are close enough to the true means once $D_i < \varepsilon$ for all $i \in [K]$, where ε is a prechosen, small number). Finally, we use the following ILP to label new datapoints:

$$\text{minimize } \sum_{i=1}^{N}\sum_{j=1}^{K} d(\vec{b_i}, \vec{M_j})x_{ij}$$

$$\text{such that } x_{ij} \in \{0,1\}, \sum_{j=1}^{K} x_{ij} = 1$$

where b_i are our new datapoints, M_j our true means, d the cosine similarity function, and x_{ij} our variables. Again, note that due to equivalence of all norms in $\mathbb{R}^m$, we can estimate cosine similarity using the ℓ_1 norm. But for the sake of clarity and simplicity, we will leave it as is in our ILP formulation.

Task 3: Label Selection

In this task, we are trying to determine which datapoints are the most valuable to label. Needless to say, it is valuable to obtain labels for the datapoints that are most uncertain. Considering our classifier from Task 1, $\frac{K(K+1)}{2}$ binary classifiers that break up $\mathbb{R}^m$ into K polyhedra, the datapoints of interest would be the ones closest to the boundary of our polyhedra.

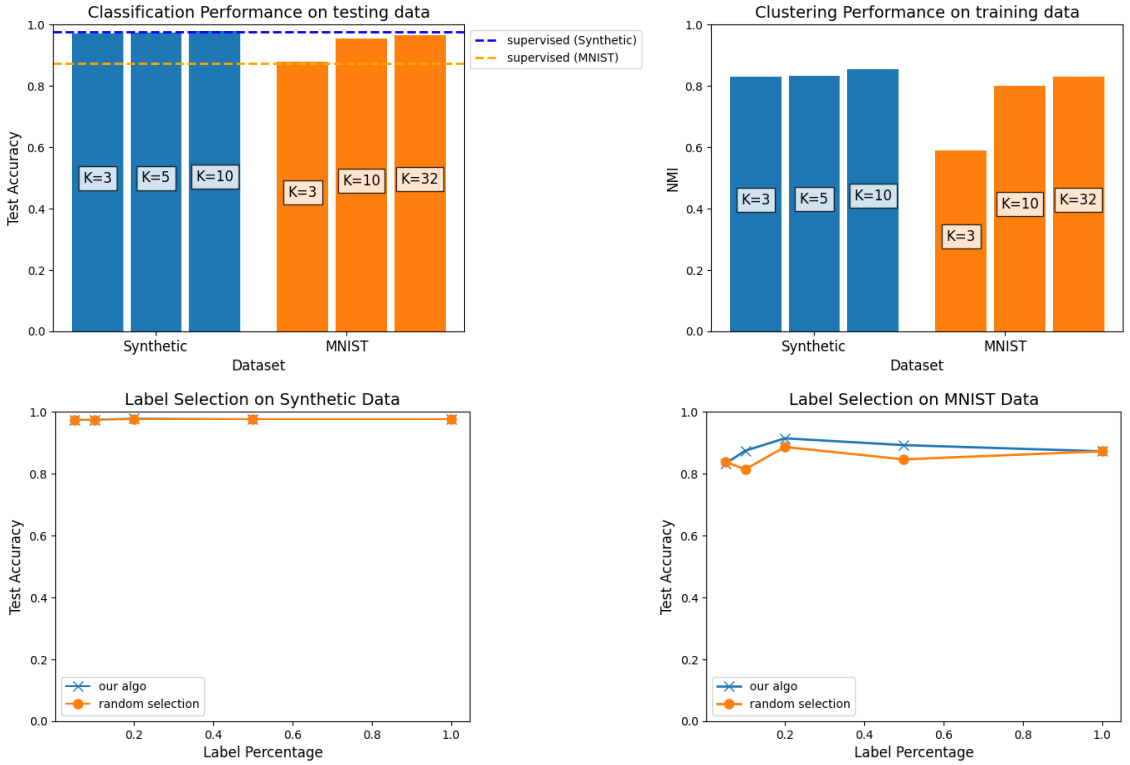
Of course, we do not know where this boundary lies until the LP's are solved, and doing so requires us to already have labels for our data. Therefore, what we will do is to use our clustering ILP from Task 2 determine the "true means" $M_1, \ldots, M_K$, and then we get the labels for the datapoints that have large cosine similarities to all M_i. However, in order for this algorithm to be optimal, we need to know the correct number of classes K in order to get good "true means". This problem will be discussed more in the next section. For now, we assume knowledge of K and formulate our ILP as follows:

$$\text{maximize } \sum_{i=1}^{N}\sum_{j=1}^{K} d(\vec{M_j}, \vec{a_i})x_i$$

$$\text{such that } x_i \in \{0,1\}, \sum_{i=1}^{N} x_i = L$$

Where N is the size of our dataset, L is the number of labels we can have, d is the cosine similarity function, and x_i are our variables. For the same reason as before, we will allow cosine similarity in our ILP.

Results and Discussion



The test results indicate that the supervised classifier in Task 1 performed quite well for the synthetic data, but significantly worse for the MNIST data. This is likely due to the large number of dimensions (N = 784) for the MNIST dataset as opposed to the small number of dimensions (N = 2) for the synthetic data. It was also noted during testing that the algorithm was quite slow when training on MNIST data. A possible solution is to adjust our SVM's to compare one class against all other classes, instead of in pairs. This would reduce the number of classifiers from $\frac{K(K-1)}{2}$ to K and hence increase the speed at which the algorithm runs. However, there is the possibility of a decrease in classification accuracy.

Our test results indicate that our clustering algorithm performed quite well across both datasets for classification, but curiously performed very poorly when clustering the MNIST data for $K = 3$. We hypothesize that this is because ℓ_2 norm was used during the clustering process and cosine similarity for classification. However, when the clustering algorithm was adjusted to use cosine similarity, the sequence of means failed to converge.

Our label selection algorithm performed slightly better than random selection for the MNIST data. Curiously, the classifier performed best with 20% labeled data for both randomly selected labeling and our labeling algorithm. One possible explaination is that the polyhedra partitioning $\mathbb{R}^m$, which our $\frac{K(K-1)}{2}$ binary classifiers, are too "complexly shaped" and will likely benefit greatly by adjusting to the "one vs all" approach suggested earlier in the discussion.

Finally, recall that our label selection algorithm assumes knowledge of the number of classes K. While implementations to solve this problem have not been made, 2 solutions seem plausible. The first solution is to modify our clustering algorithm to become X-means clustering[1] by "repetitively attempting subdivision and keeping the best resulting splits". The second is to test our clustering program on different values of K then computing the silhouette score[2]. We then choose the K with the highest silhouette score. Details of these algorithms can be found in references.

References

[1] Determining the number of clusters in a data set. (2023, November 28). In Wikipedia. https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set

[2] Silhouette (clustering). (2023, November 28). In Wikipedia. https://en.wikipedia.org/wiki/Silhouette_(clustering)