ECE 236A Project

Naoya Kumagai, Akira Suzuki, Pradyumna Chari,
Ryan Nemiroff, Deniz Bozkurt, Hayato Kato

December 9, 2021

1 Part I

1.1 Design of Classifier and Selection Algorithm

1.1.1 Classifier Linear Program

We began with the following linear program to train on our sample points y_i with labels s_i:

$$\begin{aligned}&\text{minimize } \textstyle\sum_{i=1}^{N} t_i, \text{ subject to}\\ &t_i \geq 0 \;\forall\; i = 1, ..., N\\ &1 - s_i(W^T y_i + w) <= t_i \;\forall\; i = 1, ..., N\end{aligned} \tag{1}$$

We intentionally did not include a margin maximization term because that necessitates tuning a regularization parameter, and we wanted our solution to be as general as possible without tuning. However, it is noteworthy that when the data is not linearly separable, a larger margin reduces the total penalty (the objective function) up to a certain point. More importantly, however, when the data *is* linearly separable, the objective function goes to 0 and the LP has infinite solutions. Therefore, it would be ideal to maximize the margin (minimize $||W||_2$) to create a best guess at how to classify future data points that may be close to the division hyperplane. Because of this, we changed our approach solve to solve the LP:

$$\begin{aligned}&\text{minimize } (1 + \tfrac{1}{\sqrt{M}})||W||_1 + (1 + \sqrt{M})||W||_\infty, \text{ subject to}\\ &1 - s_i(W^T y_i + w) <= 0 \;\forall\; i = 1, ..., N\end{aligned} \tag{2}$$

And if this is infeasible, the data is not linearly separable and we revert to solving (1). Here, the objective function is an approximation of $||W||_2$, which we can write as a linear function by introducing the proper slack variables and constraints. After trying a few options, it achieved the lowest $||W||_2$ in the most cases, and follows from the inequality: $||x||_\infty \leq ||x||_2 \leq ||x||_1 \leq \sqrt{M}||x||_2 \leq M||x||_\infty$, where $x \in \mathbb{R}^M$ (In (2), $W \in \mathbb{R}^M$).

Notably, the synthetic Gaussian dataset becomes non-linearly-separable after selecting just a few data points, while the entire MNIST dataset of 1's and 7's is linearly separable.

1.1.2 Sample Selection Algorithm

We began by employing a strategy of selecting data points that were classified incorrectly by the current model, which would ideally nudge the model in the right direction when it got something wrong. However, we came across a problem where the model would often be very inaccurate for the synthetic Gaussian dataset (as evidenced in Figure 1b). It is clear that this is due to the large number of data points that are necessarily classified incorrectly, even by the ideal linear classifier—a result of the large amount of noise. We alleviate the problem by modifying our algorithm to also include some points that are classified correctly, but within the margin by some amount. This leads to the following condition for selecting a point: $1 - s_i(W^T y_i + w) >= k$, where $0 \leq k \leq 1$ is a parameter chosen by us ($k = 1$ selects incorrect classifications, $k = 0$ selects any point in the margin). Of the values we tested, $k = 0.3125$ was the largest which appeared to select enough "good" points to stabilize the model for a 1000-sample synthetic Gaussian-distribution dataset (as described in the Project Description). As a point of clarification, we used exactly the same selection criteria for all tests presented in this report.

For the purposes of sample selection, we assume that the model is re-trained after every new sample is selected.

1.2 Results and Analysis

We tested with three different training datasets: A 1000-sample synthetic dataset as described in the Project Description, the full 13007-sample MNIST dataset (1's and 7's only), and a 1000-sample subset of the MNIST dataset. For each training dataset, there is a corresponding test dataset: another 1000 generated samples in the case of the synthetic dataset, and the 2163-sample MNIST testing dataset (1's and 7's only) for both MNIST training datasets.

For any classifier for any dataset, we define its *accuracy* as the proportion of correct classifications out of the total number of samples in the testing dataset (not the training dataset).

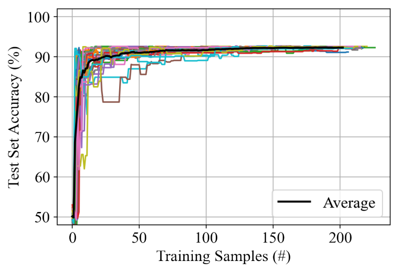
To judge the results, we first look at Table 1. The selection algorithm appears to be successful in choosing useful points from the MNIST dataset. This is evidenced by (1) the selection algorithm's 99.06% accuracy after selecting only 2.54% of the full MNIST dataset, versus the 99.31% accuracy of the classifier trained on all points, (2) the selection algorithm's 98.57% accuracy after selecting 8.06% of a 1000-point MNIST subset, versus the 98.80% accuracy of the classifier trained on all 1000 points, (3) the selection algorithm's 0.5% to 1% improvement over random selection, and (4) the faster convergence to 95% accuracy compared to random selection. The 50%, 65%, and 80% accuracy thresholds are trivial to achieve because they are reached extremely quickly in all cases.

For the synthetic dataset, the selection algorithm does not appear to be useful. In fact, the accuracy of full-dataset training, strategic selection, and random selection are all within 0.12% of each other. This implies that the selection algorithm is not designed well for this case and is choosing too many points. A graphical representation of the selection process for the synthetic set is shown in Figure 6. We argue that designing a selection algorithm that is effective on this dataset is more difficult than designing one that is effective for MNIST.

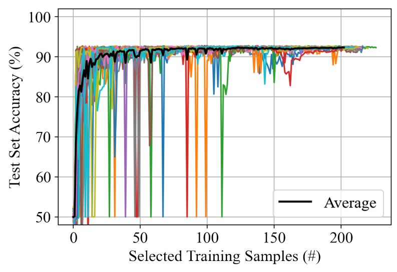
In addition to Table 1, Figures 1 and 2 show that MNIST training converges to a higher accuracy than synthetic dataset training. This is explained by the fact that MNIST is less noisy and also linearly separable. Additionally, we see signs of "instability" in training on strategically selected points from the synthetic dataset, as discussed in section 1.1.2. The general relationship between seen and selected training sample points for both data sets are also visualized in Figure 7.

dataset	**Synthetic (1000 pts)**		**MNIST (1000 pts)**		**MNIST (13007 pts)**	
all-points classifier accuracy	**91.20%**		**98.80%**		**99.31%**	
# points selected by algorithm	**218.9 (21.89%)**		**80.6 (8.06%)**		**330.4 (2.54%)**	
selection method	**algorithm**	**random**	**algorithm**	**random**	**algorithm**	**random**
# points required for 50% accuracy	2	0	0	2	1	2
# points required for 65% accuracy	2	3	2	2	3	3
# points required for 80% accuracy	5	6	4	4	5	5
# points required for 95% accuracy	N/A	N/A	10	18	11	18
final accuracy	91.08%	91.14%	98.57%	97.58%	99.06%	98.45%

Table 1: Performance metrics for the classifiers for various datasets. All metrics represent the average performance over 30 trials, where points are randomly selected/reordered each trial. N/A indicates that the particular data set did not achieve that accuracy even after using the entire subset of data.

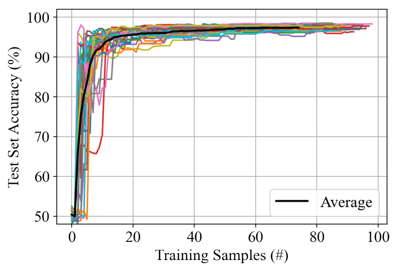


(a) Randomly Selected Training Data Set

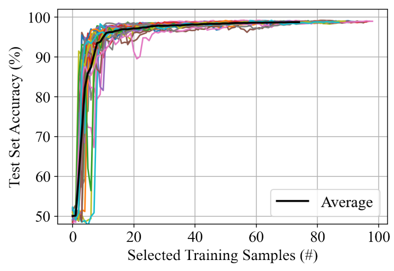


(b) Selection Algorithm Generated Data Set

Figure 1: 30 Trials of Binary Linear Classifier Training On a Synthetic Data Set (1000 Samples)



(a) Randomly Selected Training Data Set



(b) Selection Algorithm Generated Data Set

Figure 2: 30 Trials of Binary Linear Classifier Training On MNIST Data Set (1000 Samples)

2 Part II

2.1 Design of the Integer Linear Program

The goal is to design an Integer Linear Program (ILP) for near-optimal sample selection. We wish for the sample selection method to have certain desirable properties. First, we want control over chosen number of points, to enable analysis and comparison. Second, we wish to choose informative samples, i.e. samples closer to the ideal decision hyperplane, to improve hyperplane estimation accuracy from a few points. Third, we wish to choose representative samples, i.e. samples that represent the properties and distributional support (the region of the feature space where samples are likely to occur) of the overall training set. Finally, we want the linear classifier trained on the chosen samples to give a high classification accuracy on a test set.

We take inspiration from literature in online and active learning. We note that the margin of a sample from the ideal hyperplane is a measure of the informativeness of the sample. We view representativeness as ensuring sample selection parallel to and along the decision hyperplane. This ensures angular finetuning of the hyperplane.

Let N be the total number of points we want to choose. Let $\mathbf{n}_{+1}$ be a vector indicator variable for all the samples in the $+1$ class and let $\mathbf{n}_{-1}$ be a vector indicator variable for all the samples in the -1 class. Mathematically, our ILP is described follows:

$$\begin{aligned}
&\text{minimize } v_1 - v_2 \text{ //\textit{Maximize } } v_2 \text{ \textit{and minimize} } v_1.\\
&\text{subject to}\\
&\mathbf{1}^T\mathbf{n}_{+1} = 0.5N \text{ //\textit{No. of samples chosen from class label +1 set to 50\% of sample budget.}}\\
&\mathbf{1}^T\mathbf{n}_{-1} = 0.5N \text{ //\textit{No. of samples chosen from class label -1 set to 50\% of sample budget.}}\\
&\mathbf{d}_{\text{same}}{}^{+1}\mathbf{n}_{+1} \geq v_2 \text{ //\textit{Aggregate distance estimate of chosen +1 samples from all other +1 samples larger than} } v_2.\\
&\mathbf{d}_{\text{same}}{}^{-1}\mathbf{n}_{-1} \geq v_2 \text{ //\textit{Aggregate distance estimate of chosen -1 samples from all other -1 samples larger than} } v_2.\\
&\mathbf{d}_{\text{diff}}{}^{+1}\mathbf{n}_{+1} \leq v_1 \text{ //\textit{Aggregate distance estimate of chosen +1 samples from all other -1 samples less than} } v_1.\\
&\mathbf{d}_{\text{diff}}{}^{-1}\mathbf{n}_{-1} \leq v_1 \text{ //\textit{Aggregate distance estimate of chosen -1 samples from all other +1 samples less than} } v_1.\\
&\mathbf{M}_{+1}\mathbf{n}_{+1} \geq \mathbf{0} \text{ //\textit{Margin estimate for chosen +1 samples greater than 0.}}\\
&\mathbf{M}_{-1}\mathbf{n}_{-1} \geq \mathbf{0} \text{ //\textit{Margin estimate for chosen -1 samples greater than 0.}}\\
&\mathbf{n}^i_{+1}, \mathbf{n}^i_{-1} \in \{0,1\} \text{ for all valid elements } i \text{ //\textit{The indicator variables take integer values.}}\\
&v, v_2 \in \mathbb{Z}. \text{ //\textit{The constraint variables are enforced to be integers (since this is an ILP).}}
\end{aligned} \tag{3}$$

As can be seen, the objective function minimizes the difference between the constants associated with the distance constraints. We now describe the various constraints used. Please refer to our provided code for explicit implementation details.

- We used distance constraints to ensure that the distance between points from different classes is low. Specifically, the mean distance of a particular point from all other points in the opposite class in the original training corpus is used as the estimate. However, because we do not want to fix these constraints (which may require hyperparameter tuning), we made them optimization variables.
- We used distance constraints to keep the points within the same class that are not too close to each other since that introduces some redundancy. Specifically, the minimum distance for a particular point from all other points in the same class in the original training corpus is used as the estimate.
- We also enforced constraints to choose the number of selected samples. We also enforce balanced sample selection from the two classes.
- We used an approximate margin estimate and enforced that the margin is non-negative for the chosen samples. We first estimate the cluster centers (means) for each of the two classes. The vector from the negative to the positive cluster center is an estimate for the normal direction, and the mid point of the two clusters centers is the estimated location from which we calculate margin (as the sign corrected distance along the normal direction). Note that we do not train a classifier for the margin estimates; our estimates are approximate. The goal is to identify a correlated variable able to provide similar useful information as the margin.

Relaxation of the ILP to an LP: For the linear program, we used the same objective function and constraints apart from relaxing indicator variables $\mathbf{n}_{+1}$ and $\mathbf{n}_{-1}$ so that they can take any value between 0 and 1, and removed the integer constraints on v_1 and v_2. We then converted the real solution of the associated relaxed LP to an integral solution by masking with all values of n above 0.5. This is a simple approach, but worked well for our purposes.

2.2 Results

Synthetic dataset performance: Our results show comparable performance to the Part 1 implementation. First, in Figure 3, we see that the chosen points accurately describe the boundary between the two classes. For both LP and ILP solutions, to achieve 50% accuracy, it needed 1 sample, while only 2 samples to achieve 65% and 80%. However, since the best case accuracy in this case is about 92%, both the LP and ILP do not reach 95% accuracy (similar to Part I).

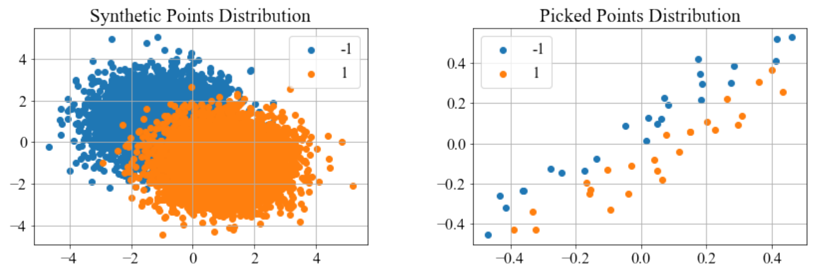


Figure 3: Part 2 Synthetic Data Sampling for n=50.

MNIST dataset performance: Next, we look at our performance on MNIST data. Results are in Figure 4. Trend lines are found by exponential regression with scatter point inputs. We need on average 2 point for 50% accuracy, 3 for 65%, 6 for 80%, and 18 for 95%. The LP sees slightly worse performance, where on average 1 point is needed for 50% accuracy, 3 for 65%, 8 for 80%, and 25 for 95%. There is also a dip in the accuracy past 50 samples taken. We postulate this to arise from intermediate values that were being unconfidently rounded selected.

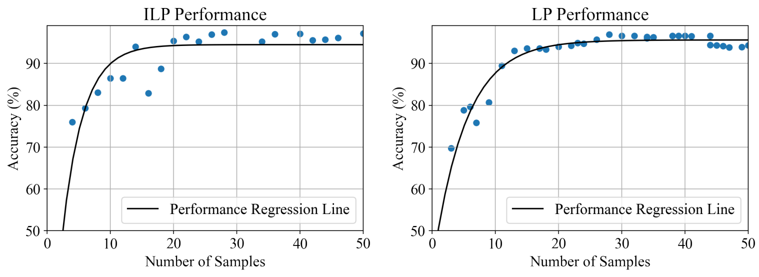


Figure 4: Accuracy trends for 2 approaches in Part 2.

Runtime analysis: We measured the runtime for both the LP and ILP, with sample budgets (optimization parameter N) of 50, 60...100 (6 runs). The analysis was carried out on Google Colaboratory using the assigned CPU with two 2250 MHz cores. The LP and ILP have runtimes of 95.38 ± 0.22 seconds and 145.38 ± 1.01 seconds respectively. Both sets of runtimes do not show correlation with the sample budget in the tested range. The LP has a smaller runtime; however, neither of the runtimes is prohibitively large.

Comparison to Part I Performance: Plots are shown in Figure 5. For the synthetic data, the part II implementation converges to its maximum performance much quicker than the part I implementation ($N = 2$ vs $N \approx 18$). For the MNIST data, although Part II ILP produces better accuracies for small N, the Part I performance with $N \geq 8$ produce better results. We postulate that the performance topoff for the part II problem solutions comes from the absence of an accurate decision hyperplane estimate and the introduction to noise from the constraints we use. also results in non-optimal sampling.

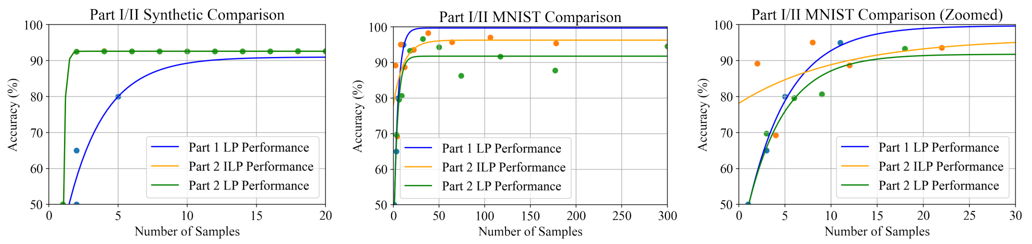
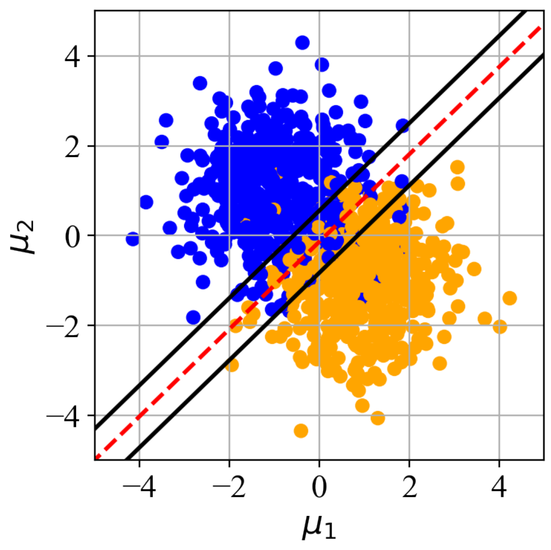
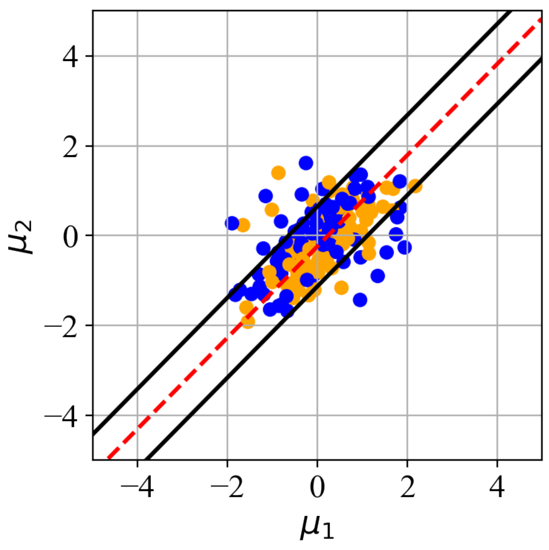


Figure 5: Accuracy trends for Part 1 and Part 2.

3 Appendix

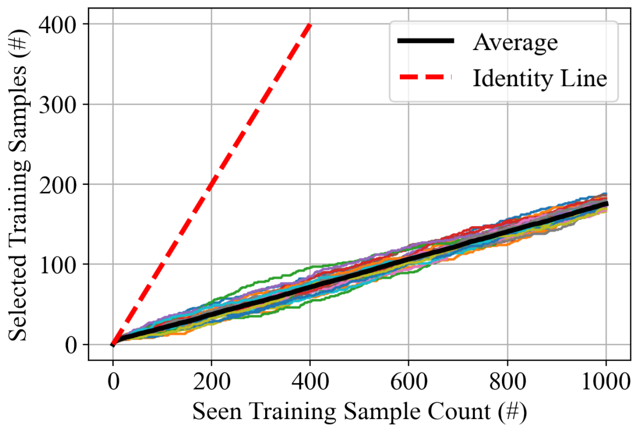


(a) Before Using Selection Algorithm

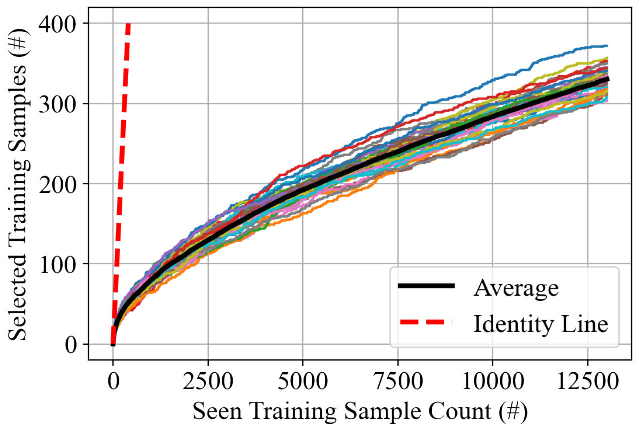


(b) After Using Selection Algorithm

Figure 6: Graphical Representation of Selective Sampling Algorithm in Effect from Part I



(a) Synthetic Data Set



(b) MNIST Data Set

Figure 7: Seen vs Selected Training Sample Point Ratio Comparison from Part I. The identity line shows the theoretical case where every single data point seen is selected to be part of the model. Both methods select less sample points than seen sample points, hence the smaller general slope. It is worth noting how the algorithm slows down its rate of adding new training samples to the model for the MNIST set due to increase in accuracy of the model.