ECE236A Project Report

Raayan Dhar	Kevin Xia	Kevin Sheng	Paul Serafimescu
506-010-953	105-940-996	406-196-414	005-495-722

1. Supervised Learning

1.1. Formulation

We learned in class that a linear classifier can be expressed with hinge loss:

$$\text{Minimize} \quad \sum_{i=1}^{N} \max(0, 1 - \mathcal{Y}_{\text{train}}{}^{(i)}(a_i^T \mathcal{X}_{\text{train}}{}^{(i)} + b_i))$$

where $\mathcal{Y}_{\text{train}}{}^{(i)} \in \{-1, 1\}$, and the variables are a_i and b_i.

We explore a different model that is not motivated by hinge loss, and show that it can produce well-performing linear decision boundaries as well. Our alternative model generalizes to $K \geq 2$. Instead of multiplying some affine transformation of $\mathcal{X}_{\text{train}}$ with $\mathcal{Y}_{\text{train}}$, we instead find the distance between them:

$$\text{Minimize} \quad \sum_{i=1}^{N} \left\| \mathbf{W}^T \mathcal{X}_{\text{train}}{}^{(i)} + \mathbf{b} - \mathcal{Y}_{\text{train, 1-hot}}{}^{(i)} \right\|_1$$

where $\mathcal{Y}_{\text{train, 1-hot}} \in \mathbb{R}^{N \times K}$ is defined as

$$\mathcal{Y}_{\text{train, 1-hot}}{}^{(i)} = \begin{bmatrix} \mathbb{1}(\mathcal{Y}_{\text{train}}{}^{(i)} = 1) \\ \mathbb{1}(\mathcal{Y}_{\text{train}}{}^{(i)} = 2) \\ \vdots \\ \mathbb{1}(\mathcal{Y}_{\text{train}}{}^{(i)} = K) \end{bmatrix}$$

and our variables are $\mathbf{W} \in \mathbb{R}^{M \times K}$ and $\mathbf{b} \in \mathbb{R}^K$. The motivation for this formulation is that any linear classifier necessarily performs some affine transformation from the input space to the output space. For each sample, we want its corresponding location in the output space to be as close as possible to the 1-hot vector for its label. So, we minimize the distance.

To express the above as a linear program, we introduce a slack variable for the L1 norm:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbf{S}_{ij} \\ \text{Subject to} \quad & \mathcal{X}_{\text{train}} \mathbf{W} + \mathbf{1}\mathbf{b}^T - \mathcal{Y}_{\text{train, 1-hot}} \leq \mathbf{S} \\ & \mathcal{X}_{\text{train}} \mathbf{W} + \mathbf{1}\mathbf{b}^T - \mathcal{Y}_{\text{train, 1-hot}} \geq -\mathbf{S} \\ & \mathbf{S} \geq 0 \end{aligned}$$

This auxiliary variable is $\mathbf{S} \in \mathbb{R}^{N \times K}$.

1.2. Implementation and Evaluation

For the synthetic dataset, the linear classifier gives the decision boundary in figure 1 (the plotted points are the test set). On the synthetic dataset, the linear classifier achieves an accuracy of 0.96, and on the MNIST dataset, the linear classifier achieves an accuracy of 0.754.

2. Unsupervised Clustering

2.1. Formulation

We took inspiration from [1] to formulate our ILP which we then relaxed to a standard LP.

$$\begin{aligned} \text{Minimize} \quad & \sum_{i,j} x_{ij} \rho(i,j) \\ \text{Subject to} \quad & \sum_j y_j \leq k \\ & \sum_j x_{ij} = 1 \\ & x_{ij} \leq y_j \\ & x_{ij}, y_j \in \{0, 1\} \end{aligned}$$

x_{ij} indicates whether point i belongs to the center that is point j, while y_j indicates if point j is a center in the first place. The constraints for this force there to be at most k centers, that every point has to be served by exactly one center, and that no point is assigned to another point that isn't a center. $\rho(i,j)$ can be any measure of similarity between the points i and j. Since it's constant over the problem, we can use any metric we like, such as the L_2 norm.

2.2. Implementation and Evaluation

We relax this ILP so that $x_{ij}, y_j \in [0, 1]$. To accommodate for this relaxation, we have to do some additional processing:

1. Let $\mathcal{D}$ be the training data. Select k points at random in $\mathcal{D}$ as cluster centers.
2. For each point in $\mathcal{D}$, we compute the Euclidean distance to each cluster center and store it in a matrix ρ such that ρ_{ij} is the distance between the ith point in $\mathcal{D}$ and cluster center j.
3. Solve the aforementioned relaxed standard LP to assign each point to a cluster label.
4. Compute new cluster centers such that for old cluster center i, the new cluster center i' is the median of all points assigned to i by the LP.
5. If $i = i' \; \forall i = 1 \dots k$, terminate iteration. Otherwise, goto step 2.

Table 1. Clustering NMI and Classification Accuracy for Synthetic and MNIST Datasets

Dataset	K	NMI	Classification Accuracy
Synthetic	3	0.8451	0.898
	5	0.7906	0.916
	10	0.8587	0.918
MNIST	3	0.1913	0.568
	10	0.4415	0.744
	32	0.5505	0.848

As we can see in figures 9 and 10, clustering performance tends to increase with higher K on the MNIST training data, but not necessarily with the synthetic training data. Our unsupervised classification outperforms the supervised classification on the synthetic dataset, likely due to extremely clear clusters in the data, while it only begins to outperform supervised on MNIST past a certain threshold $K = 10$.

3. Semi-supervised Learning

3.1. Formulation

For this task, we assume a budget to obtain correct labels for L samples in a given dataset. The objective is to select the most informative L samples to label, thereby maximizing the performance of a trained classifier. One heuristic to achieve this objective is to select a subset $P \subseteq X$ with $|P| = L$ that maximizes the coverage of the dataset within a specified neighborhood radius ϵ. This selection process is analogous to the maximum coverage problem, a variant of the minimum set cover problem, where each selected sample $p_i \in P$ covers its neighborhood $N_i = \{x_j \in X \mid d(x_i, x_j) \leq \epsilon\}$. By optimizing P to maximize $\sum_{x_j \in X} y_j$, we ensure that as many samples as possible are within ϵ of at least one labeled sample, thereby enhancing the classifier's ability to generalize effectively.

We then formulate the following linear program:

$$\begin{aligned} \text{Maximize} \quad & \sum_{j=1}^{n} y_j \\ \text{Subject to} \quad & \sum_{i=1}^{n} z_i = L, \\ & y_j \leq \sum_{i \in N_j} z_i \quad \forall j = 1, \dots, n, \\ & z_i \in \{0, 1\} \quad \forall i = 1, \dots, n, \\ & y_j \in \{0, 1\} \quad \forall j = 1, \dots, n. \end{aligned} \tag{1}$$

For our variables, we chose the following integer indicator variables:

- $z_i \in \{0, 1\}$ for $i = 1, \dots, n$: Indicator variable that equals 1 if sample x_i is selected for labeling, and 0 otherwise.
- $y_j \in \{0, 1\}$ for $j = 1, \dots, n$: Indicator variable that equals 1 if sample x_j is "covered" by at least one labeled sample within its neighborhood N_j, and 0 otherwise.

Our linear program also makes use of the following parameters:

- D: An $n \times n$ distance matrix where D_{ij} represents the distance between samples x_i and x_j.
- $N_j = \{i \mid D_{ij} \leq \epsilon\}$: The neighborhood of sample x_j, consisting of all samples within a radius ϵ of x_j.

Our objective aims to maximize the total coverage of the dataset, ensuring that as many samples as possible are within ϵ distance of at least one labeled sample. The objective value can be interpreted as the number of points covered by our selected samples respective ϵ-balls. This leaves the remaining constraints to be:

$$\begin{aligned} \text{Labeling Budget:} \quad & \sum_{i=1}^{n} z_i = L \\ \text{Coverage Constraints:} \quad & y_j \leq \sum_{i \in N_j} z_i \quad \forall j = 1, \dots, n \end{aligned}$$

- The labeling budget constraint ensures that exactly L samples are selected for labeling.

- The coverage constraints ensure that each sample x_j is marked as covered ($y_j = 1$) only if at least one sample in its neighborhood N_j is selected for labeling.

3.1.1 LP Relaxation

To make solving this ILP feasible, we allow z_i and y_j to be in $[0, 1]$ instead of $\{0, 1\}$.

After solving the LP, we apply randomized rounding to convert the continuous variables back to binary values. Specifically, each z_i is set to 1 with a probability equal to its value in the LP solution. We then adjust the number of selected samples to exactly L by selecting the top L samples based on the z_i values.

3.2. Implementation and Evaluation

Our implementation solves the relaxed linear program and applies the randomized rounding as described. For our choice of ϵ, we used a chosen percentile (hyper-parameter) of the pairwise distances in the training set, ensuring that each selected sample (prototype) captures a compact neighborhood of nearby points.

For the synthetic dataset, we found the 5th percentile to perform well across each ratio, and for the MNIST dataset, we found the 10th percentile to perform well across each ratio. We suspect that choosing ϵ specifically for each ratio could improve performance. Additionally, for the synthetic dataset, we chose to add a *low-density filter* to remove points with low coverage counts ("outliers") as options for our linear program. This is a pre-processing step that can be done in the linear program. Mathematically, this means we have $z = z \odot e_{\text{filter}}$, where we assign zero's in e_{filter} if that sample does not meet the coverage count, and one's otherwise. Without this, our linear program at low label ratios will have a propensity to choose outliers over points already covered by an ϵ-ball, as illustrated in figure 3 (compared to filtered, in figure 2).

The filtered selected samples vs. random samples for each ratio can be visually seen in figures 2, 4, 5, 6, and 7. On synthetic data, our label selection performs better than random selection, as seen in figures 8 and 11. The performance does not drastically improve, in part due to the classifier's strong performance on the synthetic dataset: when we sample 3 random points with different labels, the classifier achieves a test accuracy of **96%**, as seen in figure 14.

On MNIST, we do not use a low-density filter, as the data is high-dimensional enough. Additionally, our LP produces a significant improvement over random selection along all ratios, as seen in figures 12 13. Further improvement can be made by tuning hyperparameters.

4. Discussion and Comparison

4.1 We don't think unsupervised learning requires more data for the same performance as supervised learning. In figure 15 we see that for the synthetic dataset, the clustering and linear classifier perform similarly across varying ratios. At the same time, in figure 16, clustering accuracy does not vary as we give it more training samples, staying level around 0.63, while the linear classifier achieves above 0.70 even with much fewer training samples. In either case, the clustering NMI does not increase with more samples. and it is clear that, clustering performance does not *require* more data to match linear classifier performance.

In Task 3, we saw in figures 11 and 13 that the semi-supervised learning also does not seem to improve with more training samples.

4.2 For the synthetic dataset, we see in figure 15 the performance ramps up very quickly. Past around 10-15% of the data, the accuracy plateaus. It takes longer for the NMI to plateau (around 40-45%).

Limiting the MNIST data doesn't really seem to affect the performance at all, as shown in figure 16. While the NMI decreases slightly, the accuracy stays basically unchanged, even outperforming the linear classifier at some points.

4.3 With soft decisions, the clustering algorithm now provides a measure of confidence for each point. We can use this to filter out outliers. If a point is halfway between two representative points, then it is likely an outlier and does not correspond to a cluster, so the classifier should not assign weight to it.

This would likely improve performance with fewer samples, since it makes the classifier more robust to outliers, thus requiring less input to achieve reasonable performance.

4.4 We can reuse the *representative* samples in task 3. If we let the samples chosen by our relaxed LP be the true labels for this task, then the ϵ-balls could reinterpreted as a nearest-neighbor rule. Namely, any samples within the ϵ-neighborhood of our chosen labels would be labeled with this true label. If there are any overlaps (i.e., a sample belongs to multiple ϵ-balls), we could use a majority rule among the neighborhoods it belongs to, and then break any further ties with distance to the representative samples. In order to cover the whole dataset (and label every sample), we can either increase ϵ or use more true labels. Since our clustering LP is reasonably robust, we can run it on our new pseudo-labeled dataset.

5. Figures

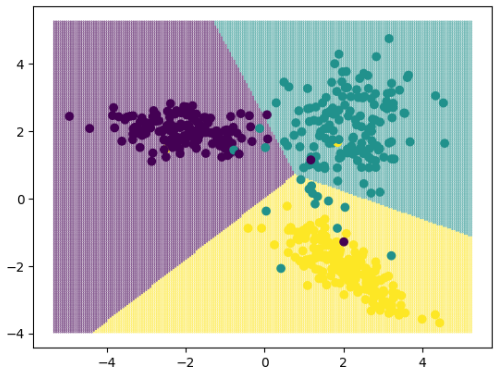

Figure 1. Linear classifier decision boundary for synthetic dataset

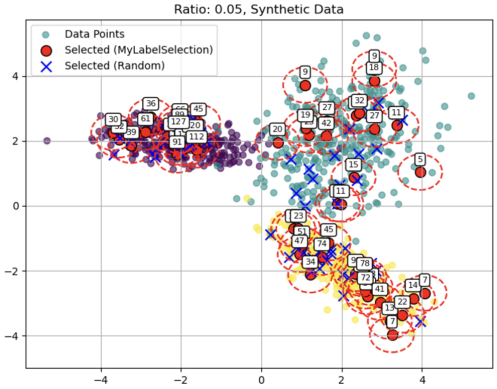

Figure 2. Selected (with ϵ-balls) vs. random points; 0.05 ratio. Annotations are coverage counts for selected samples.

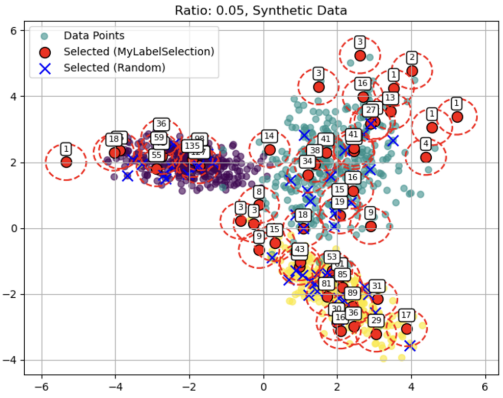

Figure 3. Selected (with ϵ-balls, **without filter**) vs. random points; 0.05 ratio. Annotations are coverage counts for selected samples.

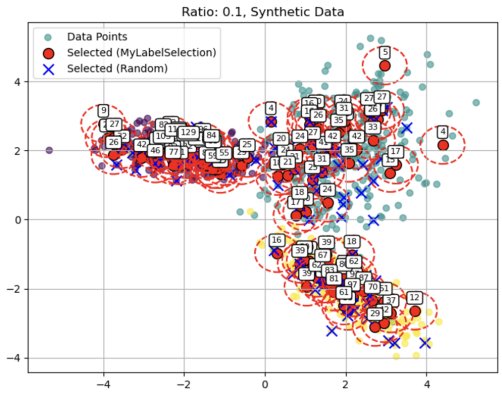

Figure 4. Selected (with ϵ-balls) vs. random points; 0.1 ratio. Annotations are coverage counts for selected samples.

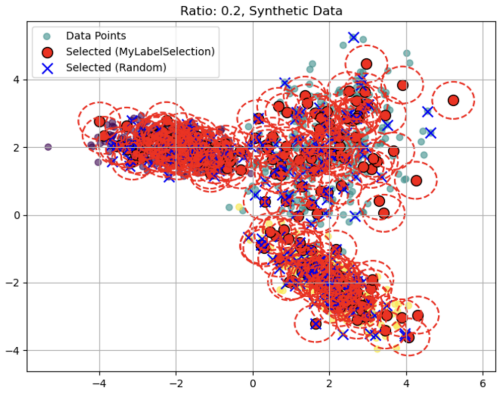

Figure 5. Selected (with ϵ-balls) vs. random points; 0.2 ratio

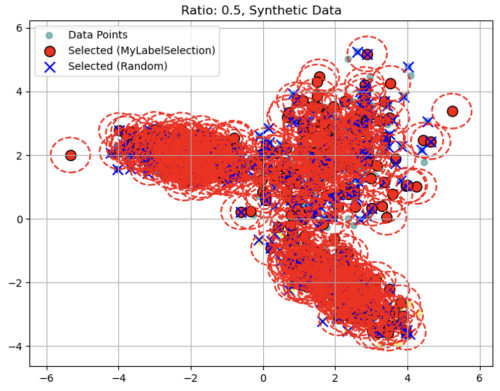

Figure 6. Selected (with ϵ-balls) vs. random points; 0.5 ratio

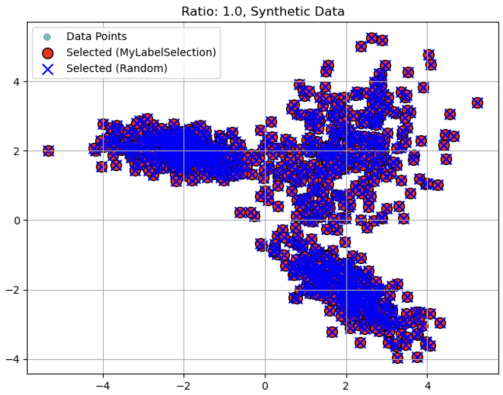

Figure 7. Selected (with ϵ-balls) vs. random points; 1.0 ratio

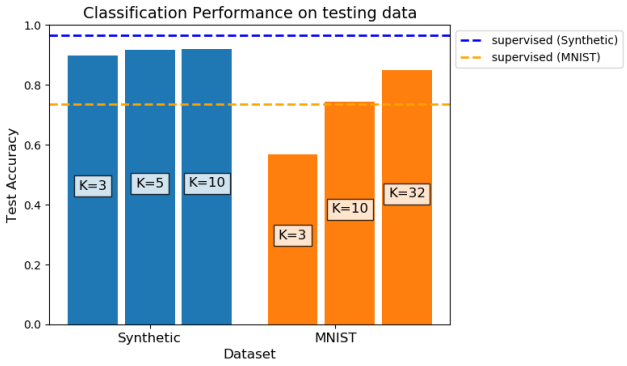

Figure 10. Unsupervised classification performance on training data

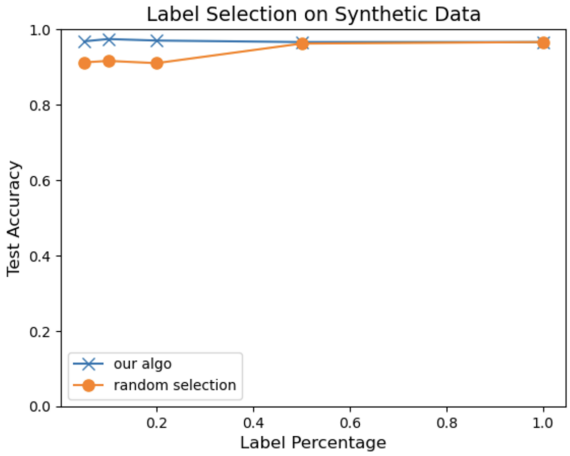

Figure 8. Label selection performance on synthetic data

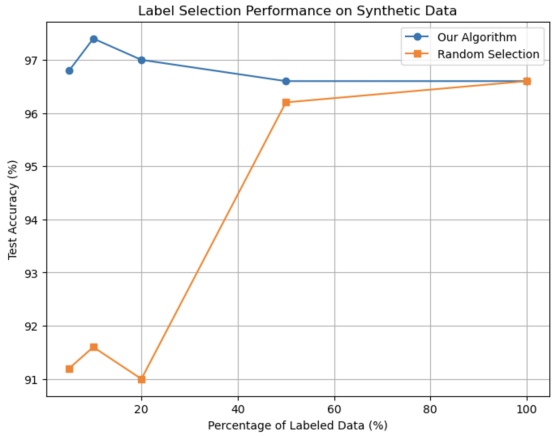

Figure 11. Label selection performance on synthetic data, close-up

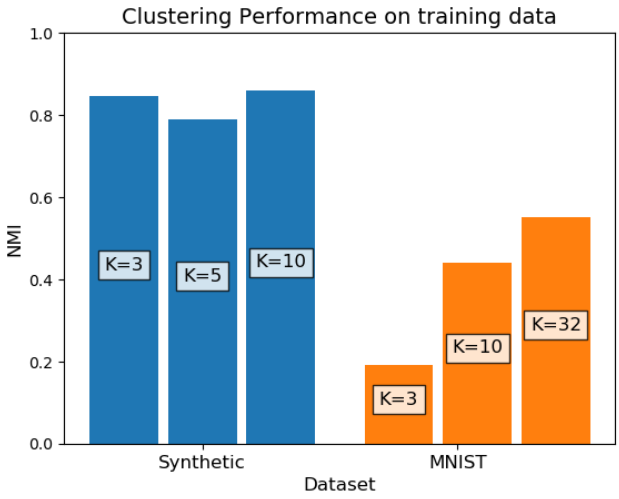

Figure 9. Unsupervised clustering performance on training data

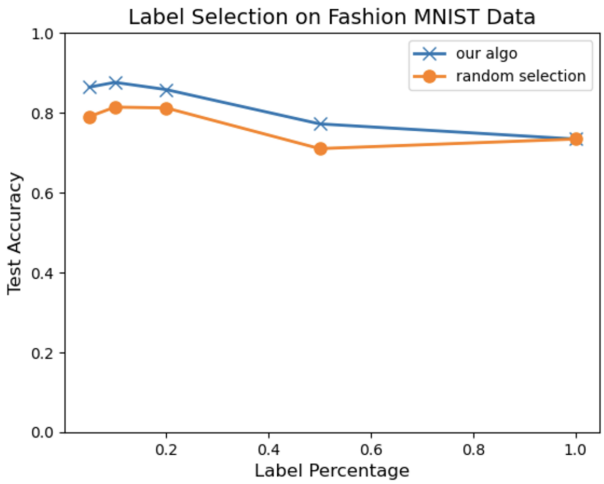

Figure 12. Label selection performance on MNIST

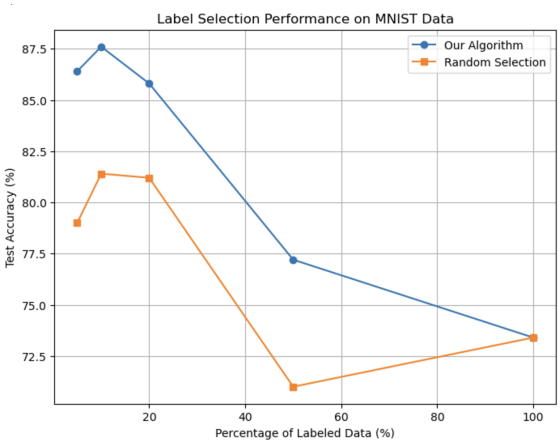

Figure 13. Label selection performance on MNIST, close-up

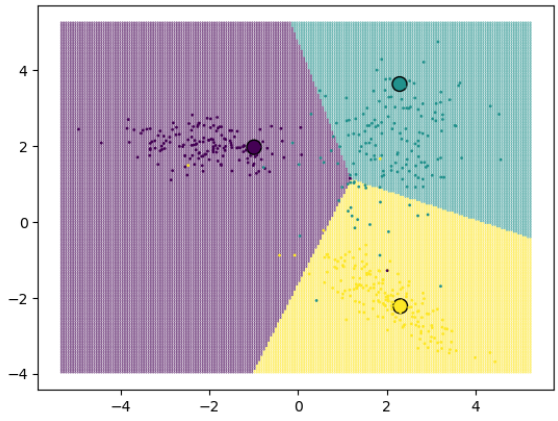

Figure 14. 3 randomly sampled points (enlarged), the resulting decision boundary, and the test set (small points)

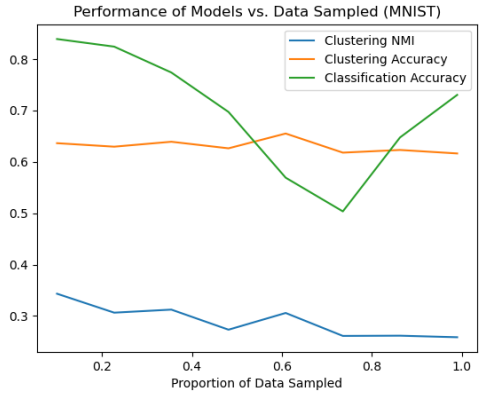

Figure 16. Accuracy/NMI on MNIST as more data is given

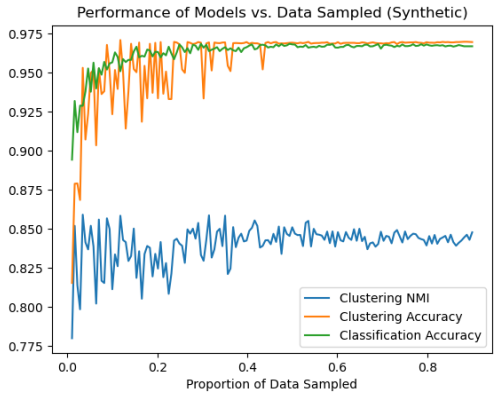

Figure 15. Accuracy/NMI on the synthetic data as more data is given

References

[1] Sanjoy Dasgupta. Lecture 2- the k-median clustering problem, April 2013. 1