# Enhancing One Nearest Neighbor Classification Efficiency through Prototype Selection: A Study on the MNIST Dataset

Allen Zou

February 1, 2024

## Abstract

Nearest neighbor classification often poses computational challenges, especially with large datasets. This paper explores the application of established prototype selection methodologies to enhance the efficiency of nearest neighbor classification. Leveraging insights from previous work, including DBSCAN clustering [1] and Active Learning [4], the approach focuses on selecting a representative subset of prototypes from the training set. The parameter $M$ represents the number of prototypes selected from the training set. A concise pseudocode for the prototype selection algorithm is provided, and extensive experiments are conducted on the MNIST dataset using varying values of $M$, including $M = 10000, 5000, 1000, 500, 100$ [2]. Evaluation results are presented to assess the performance of the proposed method.

## 1 Introduction

Nearest neighbor classification is a fundamental technique in machine learning, but its computational complexity grows with the size of the training set. Prototype selection offers a solution by identifying a representative subset of the training data for classification. This paper explores a few prototype selection strategies and evaluates their impact on classification accuracy compared to the baseline of uniform random sampling using the MNIST dataset [2]. It presents pseudocode for the proposed algorithm, describes the experimental setup, and analyzes the obtained results.

## 2 Prototype Selection Strategies

Prototype selection methods aim to reduce the computational complexity of nearest neighbor classification by identifying a subset of the training data that captures its essential characteristics. In this section, we describe three established prototype selection strategies: Uniform Random Sampling, DBSCAN-based Selection, and Active Learning. The parameter $M$ represents the number of prototypes selected from the training set.

### 2.1 Random Sampling

Uniform Random sampling is a straightforward prototype selection strategy that involves selecting a subset of prototypes randomly from the training set. This method serves as our baseline approach for comparison with more sophisticated strategies. By randomly choosing prototypes, we aim to establish a performance benchmark against which other strategies will be evaluated.

**Algorithm 1** Random Sampling

$N \leftarrow$ Length of $X_{\text{train}}$
$indices \leftarrow$ Randomly choose $M$ from $[0, N)$
$prototypes \leftarrow$ Sample at $indices$ from $X_{\text{train}}$
$labels \leftarrow$ Select labels at $indices$ from $y_{\text{train}}$
**return** $prototypes, labels$

### 2.2 DBSCAN-based Selection

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm [1] implemented in the scikit-learn library [3]. In prototype selection, DBSCAN identifies dense regions in the feature space and selects prototypes from core points within these regions, excluding outliers. This strategy focuses on capturing meaningful structures in the data by emphasizing densely populated regions.

For our implementation, $eps$ and $min\_samples$ is to 20 and 10, respectively. The $eps$ parameter defines the maximum distance between two samples for them to be considered as

**Algorithm 2** DBSCAN Prototype Selection

$core\_points \leftarrow$ Apply DBSCAN($X_{\text{train}}$)
$indices \leftarrow$ Randomly Select M $core\_points$
$prototypes \leftarrow$ Sample at $indices$ from $X_{\text{train}}$
$labels \leftarrow$ Select labels at $indices$ from $y_{\text{train}}$
**return** $prototypes, labels$

in the same neighborhood, while $min\_samples$ specifies the minimum number of samples in a neighborhood for a point to be considered as a core point. These parameter values were chosen empirically to ensure that the algorithm captures clusters effectively while excluding noisy points.

### 2.3 Active Learning

Active Learning is a dynamic prototype selection strategy that leverages uncertainty in the classifier's predictions. In our implementation, we utilize logistic regression as the base classifier, which is available in the scikit-learn library [3]. The algorithm selects prototypes based on the uncertainty associated with each sample, emphasizing instances where the classifier exhibits higher uncertainty.

**Algorithm 3** Active Learning Prototype Selection

Train LR classifier $C$ on $X_{\text{train}}, y_{\text{train}}$
$prob \leftarrow$ predicted probabilities from $C$
$uncertainty \leftarrow 1 - \max(P, axis = 1)$
$indices \leftarrow \text{argsort}(uncertainty)[-M:]$
$prototypes \leftarrow$ Sample at $indices$ from $X_{\text{train}}$
$labels \leftarrow$ Select labels at $indices$ from $y_{\text{train}}$
**return** $prototypes, labels$

## 3 Experimental Setup

We conducted experiments to evaluate the performance of prototype selection strategies in one nearest neighbor classification using the MNIST dataset. These experiments aim to compare the classification accuracy and computational efficiency of three prototype selection methods: Random Sampling, DBSCAN-based Selection, and Active Learning.

### 3.1 Dataset

We used the MNIST dataset, a benchmark dataset in machine learning, consisting of 28x28 pixel grayscale images of handwritten digits ranging from 0 to 9. The dataset comprises 70,000 samples, divided into 60,000 training images and 10,000 test images. Each image is associated with a corresponding label indicating the digit it represents.

### 3.2 Evaluation Metrics

We evaluated the prototype selection methods based on the following metrics:

- **Accuracy**: classification accuracy over multiple experiments.
- **Training Time**: time taken to select prototypes (including training time if relevant).
- **Test Time**: time taken to classify test samples.

### 3.3 Experimental Procedure

We conducted experiments for each prototype selection method using different values of $M$, representing the number of prototypes selected from the training set. For each combination of method and $M$, we performed 10 experiments to account for randomness and obtain estimates of the evaluation metrics.

It's important to note that while increasing the number of experiments generally provides more robust results, the computational demands grow proportionally. While 10 experiments provide insights into the performance of the prototype selection methods, conducting additional experiments could further enhance the analysis. Extensive computational resources required for additional experiments should be considered in the future.

## 4 Experimental Results

We present the experimental results of evaluating prototype selection methods using the MNIST dataset. Table 1 provides a summary of the mean accuracy and mean time metrics across various values of $M$. Figures 1, 2, and 3 illustrate the relationship between these metrics and the number of prototypes selected. In these figures, each dot represents the mean value of the corresponding metric, and the shadings around the dots indicate one standard deviation from the mean.

Table 1: Experimental results for prototype selection methods

| | Mean Accuracy | | | Mean Test Time (s) | | | Mean Training Time (s) | | |
|---|---|---|---|---|---|---|---|---|---|
| **Prototypes** | **Random** | **DBSCAN** | **AL** | **Random** | **DBSCAN** | **AL** | **Random** | **DBSCAN** | **AL** |
| 100 | 0.639 | 0.647 | 0.298 | 0.171 | 0.231 | 0.189 | 0.001 | 43.599 | 351.509 |
| 500 | 0.792 | 0.796 | 0.433 | 0.253 | 0.270 | 0.243 | 0.002 | 43.491 | 329.956 |
| 1000 | 0.833 | 0.836 | 0.512 | 0.325 | 0.307 | 0.320 | 0.003 | 36.377 | 315.747 |
| 5000 | 0.892 | 0.893 | 0.723 | 1.108 | 0.985 | 0.961 | 0.011 | 37.132 | 304.187 |
| 10000 | 0.913 | 0.910 | 0.844 | 2.057 | 1.790 | 1.741 | 0.030 | 36.170 | 295.461 |

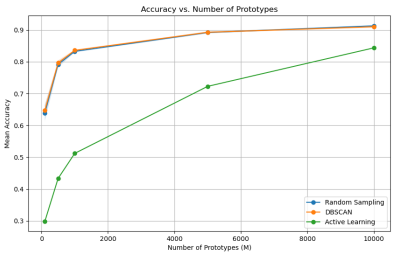


Figure 1: Mean accuracy vs. number of prototypes

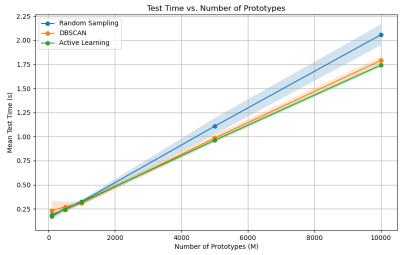


Figure 2: Mean test time vs. number of prototypes

# 5 Discussion

The experimental results, as outlined in Table 1, offer insights into the behavior of prototype selection methods and their impact on nearest neighbor classification using the MNIST dataset.

## 5.1 Accuracy Analysis

The mean accuracy results reveal unexpected patterns among the various prototype selection methods. Particularly, the accuracy of Active Learning appears to be lower compared to Random Sampling and DBSCAN. A possible explanation for this discrepancy is the intricate nature of handwritten digits in MNIST. However, it's important to note that these results could be influenced by specific parameters or aspects of the Logistic Regression implementation, warranting further investigation.

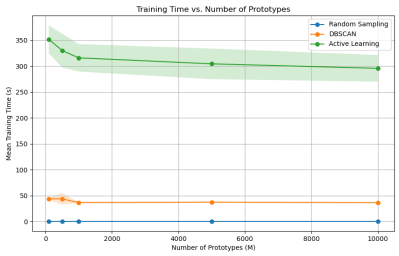


Figure 3: Mean training time vs. number of prototypes

## 5.2 Test Time Comparison

Comparing the mean test times, DBSCAN and Active Learning exhibit no significant difference between their test times. On the contrary, Random Sampling, despite being a simpler strategy, shows slightly higher mean test times. This result could be attributed to the random nature of the sampling process.

## 5.3 Training Time Comparison

The mean training times for the prototype selection methods follow the expected trend of increasing from Random Sampling to DBSCAN and then to Active Learning. However, the training time for Random Sampling is significantly higher than that of the other two prototype selection methods. Particularly noteworthy is the disparity between Active Learning and the other methods. Given that Active Learning also underperformed noticeably compared to DBSCAN and Random Sampling in terms of test time, this may necessitate a closer examination of the codebase.

## 5.4 Number of Prototypes

As the number of prototypes $M$ increases, we observe an improvement in test accuracy, with the

curve approaching an asymptote for larger M values. Meanwhile, the test time exhibits a linear increase, directly proportional to the size of the selected prototype set. Notably, the training time remains relatively stable across different $M$ values, indicating that the training process for the nearest neighbor classifier does not significantly vary with the number of prototypes.

## 6 Conclusion

Our exploration of prototype selection methodologies reveals insights into enhancing one nearest neighbor classification efficiency. Through experimentation on the MNIST dataset, we compared the performance of prototype selection methods against uniform random sampling.

The accuracy analysis revealed unexpected trends among prototype selection methods, with Active Learning exhibiting lower accuracy compared to Random Sampling and DBSCAN. Additionally, the comparison of test times showed no significant difference between DBSCAN and Active Learning, while Random Sampling exhibited slightly higher mean test times. Furthermore, the underperformance of Logistic Regression in the Active Learning prototype sampling method prompts further investigation. Lastly, as the number of prototypes $M$ increased, test accuracy improved, with test time increasing linearly, while training time remained relatively stable.

Future work could focus on refining parameter settings, exploring alternate selection strategies, and evaluating the approach across diverse datasets to establish its broader applicability.

## References


[1] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 1996.

[2] Hojjatk. Mnist dataset. https://www.kaggle.com/datasets/hojjatk/mnist-dataset, n.d.

[3] Scikit-learn Contributors. Scikit-learn documentation, n.d.

[4] B. Settles. Active learning literature survey. Technical Report 1648, University of Wisconsin-Madison, Department of Computer Sciences, 2010.