Prototype selection for Nearest Neighbors

Anay Kulkarni
Department of Computer Science
UC San Diego
ankulkarni@ucsd.edu

Abstract

Nearest Neighbor (1-NN) classification is a simple yet effective approach for many machine learning tasks, but its computational cost becomes prohibitive for large datasets such as MNIST. To mitigate this issue, prototype selection methods aim to reduce the dataset size while preserving classification performance. This report evaluates multiple prototype selection strategies, including random sampling, similarity-based selection using mean and median, farthest-point clustering, and optimal facility location. We compare these methods in terms of classification accuracy and execution time, highlighting the trade-offs between efficiency and performance. Among all methods, optimal facility location consistently achieves the best classification accuracy, even with a reduced number of prototypes, demonstrating its effectiveness in preserving the structure of the dataset. We also discuss several directions for future improvements, including exploring alternative distance metrics, optimizing facility location through parallelization and approximation techniques, and integrating dimensionality reduction methods to enhance efficiency.

1 Introduction

In many classification tasks, particularly those involving large datasets such as MNIST (which contains 60,000 training images of handwritten digits), the 1-Nearest Neighbor (1-NN) classifier can be prohibitively expensive in both computation and memory. A major drawback arises from the need to compare each test example to every training example when making a prediction. One approach to mitigating this challenge is to select a smaller set of *prototypes* that collectively preserve most of the discriminative power of the original training set. By reducing the dataset size in a controlled manner, we aim to maintain high accuracy while achieving faster classification and lower storage requirements.

Sample Size	**Entire set**	
	Time (sec)	**Accuracy**
60000	1287.28	96.91

Table 1: Results from using entire MNIST trainset

1.1 Baseline: Random Sampling

A straightforward baseline is to sample a subset of M points uniformly at random from the training set. This reduces storage and computation proportionally to the sampling ratio and is trivially easy to implement. However, random subsets can fail to capture important modes or boundary cases of each class, leading to suboptimal classification accuracy—especially at smaller sample sizes.

1.2 Similarity-based sampling using Classwise Mean or Median

This approach identifies a small set of M training examples—referred to as prototypes—from the dataset. We first divide the training data by their labels (0 through 9 for MNIST). For each digit $c \in \{0, \ldots, 9\}$, we compute a representative vector $\mathbf{r}_c$. This representative can be, for instance, the mean of all training vectors $\{\mathbf{x}_i\}$ labeled c:

$$\mathbf{r}_c = \frac{1}{N_c} \sum_{i=1}^{N_c} \mathbf{x}_i$$

where N_c = no. of training examples labeled c

Alternatively, we can define $\mathbf{r}_c$ as the median by taking the component-wise median over all vectors labeled c. Next, we use cosine similarity to measure how similar each training vector is to its class representative. Specifically, for two vectors $\mathbf{x}$ and $\mathbf{y}$, the cosine similarity is given by:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \, \|\mathbf{y}\|}$$

Algorithm 1 Most-similar prototypes by classwise Mean or Median

```
Require: $D$: Training dataset of $(\mathbf{x}_i, y_i)$ pairs with $y_i \in \{0, \ldots, 9\}$
         $M$: Desired total number of prototypes
Ensure: $P$: Set of $M$ selected prototypes
 1: Initialize $P \leftarrow \varnothing$ ▷ Set of prototypes to be selected
 2: for each class $c$ in $\{0, 1, \ldots, 9\}$ do
 3:     $X_c \leftarrow \{\mathbf{x}_i \in D \mid y_i = c\}$ ▷ Extract all vectors of class $c$
 4:     ▷ Compute representative vector $\mathbf{r}_c$
 5:     Option 1 (Mean):
            $\mathbf{r}_c = \frac{1}{|X_c|} \sum_{\mathbf{x} \in X_c} \mathbf{x}$
 6:     Option 2 (Median):
            $\mathbf{r}_c = \text{median}\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|X_c|}\}$
 7:     for each $\mathbf{x} \in X_c$ do
 8:         Compute cosine similarity:
                $\text{sim}(\mathbf{x}, \mathbf{r}_c) = \frac{\mathbf{x} \cdot \mathbf{r}_c}{\|\mathbf{x}\|\|\mathbf{r}_c\|}$
 9:     end for
10:     Sort $X_c$ in descending order by $\text{sim}(\mathbf{x}, \mathbf{r}_c)$
11:     $K \leftarrow \frac{M}{10}$ ▷ Number of prototypes per class
12:     $P_c \leftarrow$ {top $K$ vectors from $X_c$} ▷ Most similar examples
13:     $P \leftarrow P \cup P_c$
14: end for
15: return $P$ ▷ Set of $M$ prototypes
```

Within each class c, we select the top $\frac{M}{10}$ training examples whose vectors have the highest cosine similarity with $\mathbf{r}_c$. By combining these top examples across all ten digit classes, we gather a total of MM prototypes. These prototypes are then used as the "effective training set" for 1-NN classification, substantially reducing both the computational and storage burdens compared to using the entire original training set.

1.3 Farthest-Point (k-Center) Clustering

Another strategy is to select prototypes by maximizing coverage of each class distribution via farthest-point clustering (also known as the k-center approach). Instead of focusing on similarity to a single mean or median, we iteratively pick points in a class that are farthest from all already-selected prototypes. Concretely:

1. Choose an initial prototype at random in the class.
2. For each subsequent prototype, select the point that is farthest (in Euclidean or another metric) from all existing prototypes.
3. Repeat until reaching $M/10$ prototypes per class.
4. By spreading out the selection, we ensure that each class is represented by a diverse set of prototypes covering its primary modes and boundary regions, often improving classification accuracy on varied test samples.

Algorithm 2 Farthest Point Clustering for Prototype Selection

```
Require: Dataset $X$, number of prototypes $M$
Ensure: Selected prototype set $S$
 1: Initialize $S \leftarrow x_0$, where $x_0$ is a randomly chosen point from $X$
 2: Compute initial distances $d_i \leftarrow d(x_i, x_0)$ for all $x_i \in X$
 3: for $j = 2$ to $M$ do
 4:     Find $s_j = \arg\max_{x_i \in X \setminus S} d(x_i, S)$
 5:     Add $s_j$ to $S$
 6:     for each $x_i \in X$ do
 7:         Update $d_i \leftarrow \min(d_i, d(x_i, s_j))$
 8:     end for
 9: end for
10: return $S$
```

1.4 Facility Location (Core-Set) Approach

The facility location problem is a classical combinatorial optimization problem that can be adapted for prototype selection in the context of 1-Nearest Neighbor (1-NN) classification. The fundamental idea is to choose a subset of training points (facilities) such that the total cost of serving all other points (clients) is minimized. This is mathematically framed as:

$$\min_{S \subseteq X, |S| = M} \sum_{x \in X} \min_{s \in S} d(x, s) \qquad (1)$$

where X is the full dataset, S is the set of selected prototypes, and $d(x, s)$ is a chosen distance function (e.g., Euclidean distance, cosine similarity). The objective is to ensure that every sample x

is as close as possible to at least one prototype in S, thereby minimizing the overall coverage cost.

Computational Complexity The naive implementation requires $O(Mn^2d)$ operations due to the repeated recomputation of distances for every candidate prototype. This becomes intractable for large datasets such as MNIST. Optimizations may need to be applied. We optimize the execution using dimensionality reduction (PCA) during the process of prototype selection process. Then the selected prototypes are projected back to their original dimensions.

Algorithm 3 Facility Location Greedy Algorithm for Prototype Selection

```
Require: Dataset $X$, number of prototypes $M$
Ensure: Selected prototype set $S$
 1: Initialize $S \leftarrow \emptyset$ ▷ Set of prototypes to be selected
 2: Compute initial distances $d_i \leftarrow \infty$ for all $x_i \in X$
 3: for $j = 1$ to $M$ do
 4:     Find
            $s_j = \arg\max_{s \in X \setminus S} \sum_{x_i \in X} \max(d_i - d(x_i, s), 0)$
        ▷ select facility that provides best improvement on sum of distances
 5:     Add $s_j$ to $S$
 6:     for each $x_i \in X$ do
 7:         Update $d_i \leftarrow \min(d_i, d(x_i, s_j))$ ▷ Update distances of clients based on newly added facilities
 8:     end for
 9: end for
10: return $S$
```

Unlike mean- or median-based prototype selection, which may over-represent dense regions, the facility location approach explicitly balances prototype selection across the entire feature space. It is particularly useful when class distributions are multi-modal, ensuring that peripheral and diverse examples are included. Compared to random sampling, it provides a more structured and representative subset, leading to better generalization in 1-NN classification.

2 Results and discussion

The performance of different prototype selection strategies was evaluated in terms of classification accuracy and execution time. The results, shown in Tables 2 and 3, illustrate the trade-offs between computational efficiency and classification performance.

Accuracy Analysis

From Table 2, we observe that random sampling provides a relatively strong baseline, particularly for larger sample sizes, achieving an accuracy of 94.93% at 10,000 prototypes, which is very close to the $\sim$97% accuracy that can be achieved on the entire set. Similarity-based selection with mean and median underperforms random selection at all sample sizes, indicating that selecting points closest to the class mean or median does not necessarily lead to a more representative subset. Farthest-point clustering exhibits significant improvement at larger sample sizes, reaching 94.24% at 10,000 prototypes, demonstrating the advantage of selecting diverse and well-separated prototypes. The best performance is achieved by Optimal Facility Selection, which reaches an accuracy of 95.00% at 5,000 prototypes, outperforming other methods consistently.

Execution Time Analysis

Table 3 highlights the computational costs associated with each method. Random sampling is the most efficient, requiring only 169.18 seconds for 10,000 prototypes. Similarity-based sampling (mean and median) is notably more expensive, particularly at higher sample sizes. A possible explanation could be that similarity-based selection (Mean/Median) tends to pick prototypes that are highly clustered near the mean or median of each class. This can lead to redundant or overly similar prototypes that do not sufficiently cover the class distribution. Farthest-point clustering and optimal facility selection exhibit similar execution times, both requiring $\sim$140 seconds at 10,000 prototypes. While these methods are computationally heavier than random selection, they provide significantly improved accuracy, making them viable options for prototype selection.

One of the key findings from the results is that the Optimal Facility Selection algorithm scales down exceptionally well compared to other prototype selection strategies. As observed in Table 2, even when the number of prototypes is significantly reduced, the accuracy of the model using Optimal Facility Selection remains consistently high. For instance, with just 100 prototypes, this method already achieves 84% accuracy, while random sampling, similarity-based selection (mean and

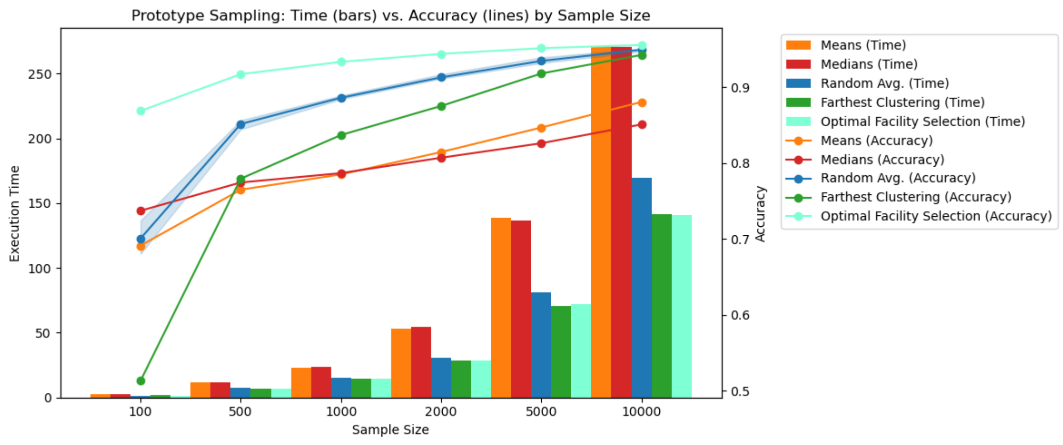

Figure 1: A comparison of the performance of various prototype sampling strategies measured across different sample sizes. The results shown for random sampling are averages calculated over 5 iterations. The random sampling accuracy is charted in a *min-max* range. The execution times shown are *prediction times*, and not sampling times, since sampling times can vastly vary for different strategies. We see that Optimal Facility Location strategy performs exceedingly well in comparison to the other sampling strategies showing minimal deterioration in performance even as the sample size is scaled down significantly.

Method	**Accuracy by Sample Size**					
	100	**500**	**1000**	**2000**	**5000**	**10000**
Random Sampling	70.04	85.13	88.59	91.28	93.44	94.93
Similarity (Mean)	69.11	76.48	78.49	81.44	84.67	88.03
Similarity (Median)	73.71	77.42	78.66	80.69	82.60	85.08
Farthest Point Clustering	51.30	77.92	83.67	87.51	91.79	94.24
Optimal Facility Selection	**86.86**	**91.71**	**93.33**	**94.38**	**95.13**	**95.56**

Table 2: Accuracy comparison of prototype selection methods

median), and farthest-point clustering suffer from more significant accuracy degradation at smaller sample sizes. This suggests that Optimal Facility Selection is highly effective at retaining the most representative points in the dataset, ensuring that classification performance remains stable even under aggressive data reduction. In contrast, methods like random sampling and similarity-based selection exhibit more erratic scaling behavior, with accuracy dropping sharply at smaller sample sizes. This makes Optimal Facility Selection a particularly robust choice when memory and computational efficiency are critical, as it minimizes the trade-off between dataset reduction and classification performance.

3 Critical evaluation and Future Work

The results of this study highlight the trade-offs between prototype selection efficiency and classification accuracy in the 1-NN setting. While random sampling provides a strong baseline in terms of execution speed, farthest-point clustering and optimal facility selection emerge as superior choices for balancing efficiency with classification performance. However, one limitation of the current approach is the reliance on Euclidean distance as the primary similarity measure. Different distance metrics, such as Manhattan distance, Mahalanobis distance, or learned distance functions, could significantly impact the selection of prototypes and the effectiveness of classification. For example, Manhattan distance may work better in high-dimensional sparse data, whereas Mahalanobis distance, which accounts for feature correlations, might provide improved separation between classes. Future work could systematically evaluate how alternative distance metrics influence prototype selection and classification accuracy.

Another important consideration is the computational cost of the facility location algorithm. While the greedy algorithm used in this study provides a near-optimal selection of prototypes, its quadratic time complexity makes it impractical for very large datasets. Several approaches can be explored to optimize facility location methods, including vectorization using efficient numerical libraries (e.g., NumPy, JAX, or TensorFlow), parallelizing distance computations across multiple cores or GPUs, and using approximations such as locality-sensitive hashing (LSH) or clustering-based pre-selection. Additionally, employing better search structures like k-d trees, ball trees, or vantage-point trees could significantly speed up the nearest-neighbor search during classification, reducing overall runtime.

Lastly, dimensionality reduction techniques play a crucial role in optimizing both the selection and classification processes. While PCA was briefly discussed as a potential preprocessing step, alternative approaches such as t-SNE, UMAP, or autoencoders may offer more meaningful low-dimensional representations that improve prototype diversity while preserving class separability. Future research could investigate hybrid methods, where prototype selection is performed in a reduced-dimensional space but classification occurs in the original feature space. Furthermore, integrating meta-learning techniques to dynamically adjust the number and distribution of prototypes based on dataset complexity could further enhance the adaptability of prototype selection methods. By combining these strategies, we can move toward more scalable, efficient, and robust solutions for nearest-neighbor classification in large-scale datasets.

Acknowledgments & References

- Most of the LaTex formatting for this paper was prepared using ChatGPT with minor changes
- Content was proof-read and revised using Grammarly with minor changes

- Wikipedia: Optimal Facility Location
- Medium.com: Farthest Point Sampling for K-Means Clustering