Prototype Selection for Nearest Neighbor

Rohan Shingre
rshingre@ucsd.edu

Abstract

This study explores efficient prototype selection strategies to accelerate nearest neighbor classification, focusing on K-means clustering, Condensed Nearest Neighbor (CNN), Interval-Diverse Distance Sampling (IDDS), and an ensemble approach combining the first three methods. The objective is to identify a representative subset of prototypes from the training set, optimizing classification accuracy on test data while mitigating computational costs. Experimental evaluation on the MNIST dataset demonstrates the effectiveness of these approaches in improving classification efficiency without compromising accuracy.

1 Introduction

Nearest neighbor classification is a powerful and intuitive method for pattern recognition and classification tasks. However, its computational complexity grows linearly with the size of the training dataset, making it impractical for large-scale applications. One effective strategy to mitigate this challenge is to replace the entire training set with a carefully selected subset of prototypes. These prototypes serve as representatives of the dataset, allowing for faster nearest neighbor search during classification while preserving or even enhancing accuracy.

In this study, we explore various prototype selection methods aimed at improving the efficiency of nearest neighbor classification, with a specific focus on the MNIST dataset—a classic benchmark in machine learning. We investigate four distinct approaches: K-means clustering, Condensed Nearest Neighbor (CNN), Interval-Diverse Distance Sampling (IDDS), and an ensemble method combining the strengths of the first three techniques. Each method is designed to select a subset of prototypes from the training data based on different criteria, such as cluster centroids, nearest neighbor editing, distance-based diverse interval sampling, or a combination thereof.

The ultimate goal of this research is to identify an effective prototype selection strategy that balances computational efficiency with classification performance on test data. By evaluating these methods on the MNIST dataset, we aim to provide insights into the relative merits of each approach and contribute to the development of more efficient and accurate nearest neighbor classifiers.

2 Prototype Selection Strategies

2.1 Baseline

Our prototype selection strategy involves comparing against a baseline approach of random subset selection. In this baseline, for a given subset size, instances are randomly chosen from the training data without any specific criteria. Our implemented methods, on the other hand, utilize a systematic approach such as K-means clustering or interval-based sampling to select prototypes from the training data. By comparing the performance of our approaches with the baseline, we aim to assess the effectiveness of our methods in improving classification accuracy and efficiency.

2.2 K-Means Clustering

In this approach, K-means clustering is used to partition the training dataset into a set number of clusters. Then, for each cluster, a specified subset size determines how many instances will be chosen. From each cluster, an equal number of the closest instances to the centroid are selected and added to the prototype subset. This method ensures that the prototype subset maintains representation from each cluster while adhering to the specified subset size, facilitating efficient nearest neighbor classification.

The objective of K-means clustering is to minimize the within-cluster sum of squares (WCSS), which measures the squared distances between each data point and the centroid of its assigned cluster. Mathematically, WCSS is defined as:

$$WCSS = \sum_{i=1}^{K} \sum_{x \epsilon C_i} \|x - \mu_i\|^2$$

where K is the number of clusters, C_i is the set of data points assigned to cluster i, μ_i is the centroid (mean) of cluster i.

2.3 Condensed Nearest Neighbor (CNN)

Our prototype selection method, Condensed Nearest Neighbor (CNN), aims to streamline the nearest neighbor classification process by reducing the size of the training dataset while maintaining its representativeness. CNN operates by iteratively adding instances to the prototype subset based on their classification performance. Initially, a random instance from each class is selected as a prototype. Subsequently, each remaining instance in the training set is compared to the prototypes. If an instance is misclassified, it is added to the prototype subset. This process continues until no more instances need to be added or until a maximum number of iterations is reached. The resulting prototype subset effectively captures the diversity and discriminatory information of the original dataset, enabling efficient and accurate nearest neighbor classification.

CNN computes distances between instances to determine their similarity or dissimilarity. The most commonly used distance metric is the Euclidean distance, given by:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

where x and y are two instances, each represented as a vector of n features.

2.4 Interval-Diverse Distance Sampling (IDDS)

IDDS (Interval-Diverse Distance Sampling) is a novel prototype selection approach aimed at improving nearest neighbor classification efficiency. It systematically selects prototypes by dividing the training dataset into classes and sorting instances within each class based on their distances to a randomly chosen prototype. Then, given a subset size, a specified number of instances are sampled at regular intervals from the sorted distances within each class. This method ensures a diverse representation across classes while maintaining a balanced subset, contributing to both computational efficiency and classification accuracy on test data.

The intuition to use this algorithm is that we select equal number of elements from each class, thereby, reducing bias. We also ensure sparsity of the data by selecting elements at regular intervals of distance. This helps to consider all kinds of data for the training of our KNN algorithm.

IDDS computes distances between instances within each class to determine their similarity or dissimilarity. As with CNN, the Euclidean distance is commonly used as the distance metric.

2.5 Ensemble Modeling

Our ensemble prototype selection method combines the strengths of K-means clustering, Condensed Nearest Neighbor (CNN), and Interval-Diverse Distance Sampling (IDDS). In this approach, given a subset size, each individual method is employed to generate a smaller prototype subset from the training data. Each individual method is given a weight based on how well they perform. This weight decides the size of the subset to be generated by that method. The resulting subsets are then combined to form an ensemble prototype subset.

In our ensemble techniques, base models are assigned weights to reflect their relative importance or performance. Mathematically, the weighted combination of predictions can be represented as:

$$(x, y)_{ensemble} = \sum_{i=1}^{n} w_i \cdot (x, y)_i$$

where w_i is the weight assigned to the i-th model, n is the number of models and $(x,y)_i$ is the subset produced by ith method.

3 Implementation

3.1 K-Means Clustering

K-means clustering is an unsupervised machine learning algorithm used for partitioning data into K clusters based on their similarities. The algorithm iteratively assigns data points to the nearest cluster centroid and then updates the centroids based on the mean of the assigned points.

This process continues until convergence, where the centroids no longer change significantly or a specified number of iterations is reached. K-means clustering aims to minimize the within-cluster sum of squares (WCSS), ensuring that data points within the same cluster are close to each other.

Finally, after training K-means, we calculate the distance from each centroid to each point in the training set. For a given value of M, we return M/n closest points for each centroid. This ensures that we give equal weight for all clusters, thereby, reducing bias towards a specific class.

Algorithm 1 K-Means Clustering

```
procedure K-MEANS
    Train K-means for n clusters
    for each centroid c do
        for each x in X_train do
            Calculate distance from c to x
    for Different subset size M do
        for each centroid c do
            Get M/n closest points
    return subset
```

3.2 Condensed Nearest Neighbor (CNN)

This CNN algorithm aims to construct a subset S of the training set TR in such a way that each instance in TR is closer to a member of S from the same class than to a member of S from a different class. Initially, it randomly selects one instance from each output class of TR and adds them to S. Then, each instance in TR is classified using only the instances in S. If an instance is misclassified, it is included in S, ensuring correct classification. This process iterates until all instances in TR are correctly classified. While this algorithm guarantees correct classification for all instances in TR, it may not produce a minimal subset.

3.3 Interval-Diverse Distance Sampling (IDDS)

Implementing Interval-Diverse Distance Sampling (IDDS) involves several steps to select a representative subset of instances from the training dataset for classification tasks. Initially, random instances from each class in the training dataset are chosen and added to the subset. Then, for each class, the distances between the selected instance and all other instances from the same class are computed and sorted in ascending order. Next, the subset

Algorithm 2 Condensed Nearest Neighbor

```
procedure CNN
    n = Number of classes
    Initialize n randoms points
    Add the random points to the subset
    for Different subset size M do
        while subset length is less than M do
            Train KNN on the subset
            Classify remaining data in X_train
            Add misclassified data to subset
            if No misclassified data then
                break
    return subset
```

size is divided by the number of classes to determine the number of instances to be selected from each class. The sorted distances are then divided into intervals, and instances are chosen at regular intervals from each interval until the desired subset size is reached. This process ensures diversity within the selected subset while maintaining a representative sample from each class, contributing to improved classification performance.

Algorithm 3 Interval-Diverse Distance Sampling

```
procedure IDDS
    n = Number of classes
    Initialize n randoms points
    Add the random points to the subset
    for each class i do
        r = point from subset of class i
        for each x in X_train do
            if class of x is i then
                Calculate distance from x to r
    Sort the distances
    for Different subset size M do
        for each class c do
            K = ((size of c)*n)/M
            Choose M/n data at intervals of K
            Add data to the subset
    return subset
```

3.4 Ensemble Modeling

Implementing an ensemble model that combines K-means clustering, Condensed Nearest Neighbors (CNN), and Interval-Diverse Distance Sampling (IDDS) involves integrating the outputs of each individual method to make collective predictions. First, K-means clustering is applied to partition the dataset into clusters, and instances from each cluster are selected to form a representative

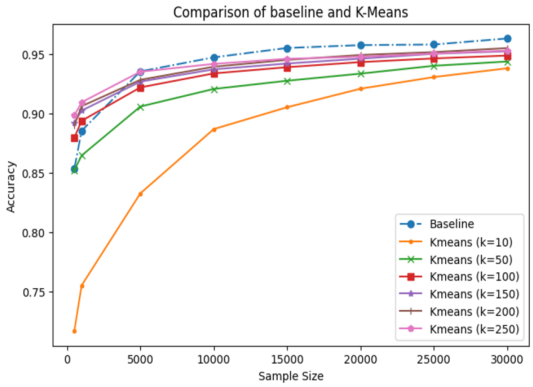

Figure 1: K-Means Performance Against Baseline

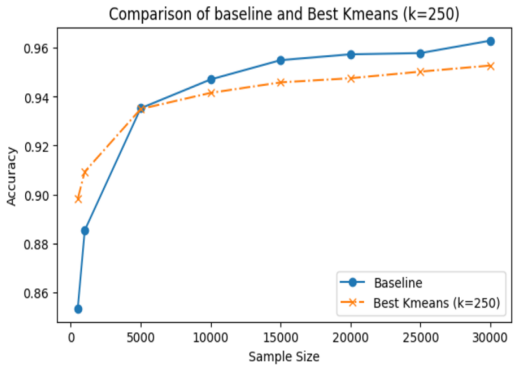

Figure 2: Best K-Means Performance Against Baseline

subset. Simultaneously, CNN is employed to extract features from the dataset, and instances are chosen based on the learned representations. Additionally, IDDS selects diverse instances based on distance intervals. These selected subsets from each method are then combined to form the ensemble dataset. Finally, a classification model is trained on the ensemble dataset to make predictions. The combination of diverse selection strategies ensures comprehensive coverage of the dataset, potentially leading to improved classification performance compared to individual methods alone.

Algorithm 4 Ensemble Modelling

```
procedure ENSEMBLE
    Assign weight w_i to each model
    for Different subset size M do
        Get w_1*M data from K-means
        Get w_2*M data from CNN
        Get w_3*M data from IDDS
    Combine all the subsets
    return subset
```

4 Experimental Results

4.1 K-Means Clustering

Our experiments reveal an intriguing trend where the K-means clustering approach outperforms the baseline method for smaller subset sizes, while the baseline method exhibits superior performance for larger subset sizes. This observation suggests that the effectiveness of each approach is contingent upon the size of the subset being considered.

For smaller subsets, K-means clustering excels in selecting representative instances, possibly due to its ability to identify distinct clusters within the dataset. Conversely, as the subset size increases, the baseline method, which randomly selects instances from the entire training dataset, appears to capture a more diverse and comprehensive representation of the data.

This nuanced understanding underscores the importance of considering the interplay between subset size and selection strategy when devising effective prototype selection algorithms.

4.2 Condensed Nearest Neighbor (CNN)

The performance evaluation of the Condensed Nearest Neighbors (CNN) algorithm against the baseline method across various subset sizes reveals a consistent trend: CNN consistently underperforms compared to the baseline approach for all subset sizes evaluated. This unexpected finding suggests that the CNN algorithm may struggle to effectively distill the essential information from the dataset, resulting in suboptimal prototype selection.

Further analysis is warranted to identify potential limitations or shortcomings of the CNN algorithm, and exploration of alternative instance-based learning techniques may be necessary to improve its performance in prototype selection tasks. Additionally, fine-tuning the parameters or refining the algorithmic implementation of CNN could be explored as potential avenues for enhancing its effectiveness in prototype selection.

4.3 Interval-Diverse Distance Sampling (IDDS)

The evaluation of Interval-Diverse Distance Sampling (IDDS) against the baseline method across various subset sizes reveals a promising trend: IDDS consistently outperforms or equals the base-

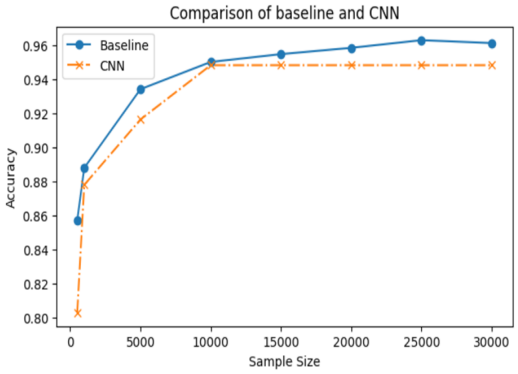

Figure 3: CNN Performance Against Baseline

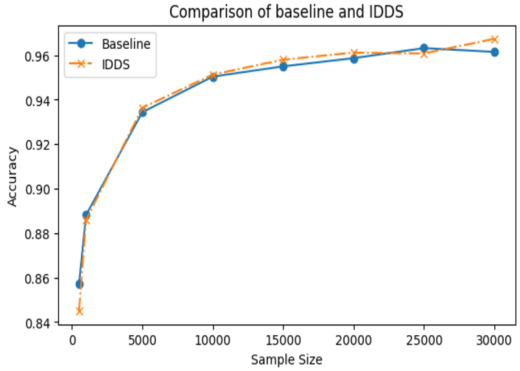

Figure 4: IDDS Performance Against Baseline

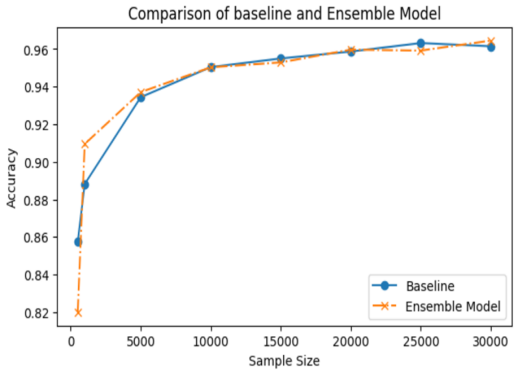

Figure 5: Ensemble Performance Against Baseline

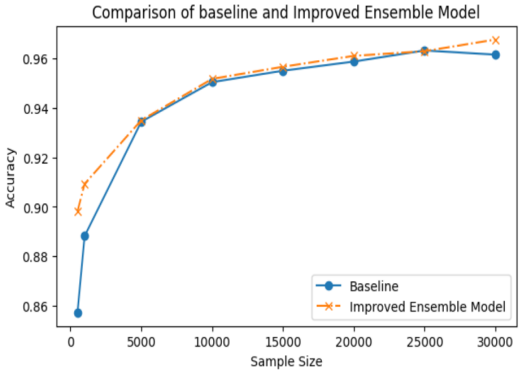

Figure 6: Improved Ensemble Performance Against Baseline

line approach for the majority of subset sizes evaluated. This positive finding suggests that the IDDS algorithm effectively selects representative instances from the dataset, resulting in improved classification performance compared to random selection.

The success of IDDS underscores the importance of diversity in prototype selection and the significance of considering distance-based intervals for capturing a comprehensive representation of the dataset. Further exploration of IDDS and its potential applications in prototype selection tasks may provide valuable insights into optimizing subset selection strategies for enhanced classification accuracy.

4.4 Ensemble Modelling

Through rigorous experimentation, we devised an ensemble modeling strategy that dynamically adjusts the weights assigned to K-means, CNN, and IDDS based on the subset size, optimizing classification performance. Initially, we employed a weighted combination of 0.3 for K-means, 0.05 for CNN, and 0.65 for IDDS when the subset size was less than or equal to 5000, and adjusted the weights to 0.2 for K-means, 0.05 for CNN, and 0.75 for IDDS for larger subset sizes. We noticed that K-means performs better for smaller subset size and IDDS performs better for larger subset size.

Subsequently, we trained an additional ensemble model by assigning a weight of 1 to K-means when the subset size was less than or equal to 5000 and 1 to IDDS otherwise. Surprisingly, our second ensemble model demonstrated superior performance, indicating the efficacy of prioritizing either K-means or IDDS exclusively based on subset size. This iterative refinement underscores the importance of adaptability and optimization in ensemble modeling to achieve optimal classification accuracy. The second ensemble model consistently outperforms the baseline.

All the models have been tested for different values of M (subset size = [500, 1000, 5000, 10000, 15000, 20000, 25000, 30000]). All the accuracies have been listed at the end.

Method	Accuracy (M=500)	Accuracy (M=30000)
Baseline	0.8574	0.9615
K-Means (k=250)	0.8982	0.9528
CNN	0.8032	0.9485
IDDS	0.8451	0.9674
Ensemble	0.8201	0.9645
Improved Ensemble	0.8982	0.9676

Table 1: Accuracy comparison of KMeans

5 Conclusion

In conclusion, our comprehensive exploration of prototype selection methods, including K-means clustering, Condensed Nearest Neighbors (CNN), Interval-Diverse Distance Sampling (IDDS), and ensemble modeling, provides valuable insights into their effectiveness for classification tasks. We found that K-means clustering demonstrates strong performance for smaller subset sizes, effectively identifying representative instances.

However, Condensed Nearest Neighbors (CNN) consistently underperformed across all subset sizes, suggesting limitations in its ability to distill essential dataset information. In contrast, IDDS consistently outperformed the baseline method for a wide range of subset sizes, highlighting its ability to select diverse and representative instances based on distance intervals. Moreover, ensemble modeling approaches, where weights were adaptively adjusted based on subset size, exhibited significant improvements in classification accuracy.

Notably, the second ensemble model, which prioritized either K-means or IDDS exclusively based on subset size, demonstrated superior performance, underscoring the importance of dynamic weighting strategies. These findings underscore the significance of considering both the selection method and subset size in prototype selection tasks, paving the way for more effective and adaptable machine learning algorithms in diverse real-world applications.

Another interesting observation is that the CNN algorithm produces output with smaller subset sizes than required. This happens due to the iterative nature of CNN. This is an advantage of CNN as it tried to minimize the size required for training.

Continued research and refinement of these techniques hold the potential to further enhance classification accuracy and robustness in various domains.

A Appendix A

GitHub link for code : https://github.com/Ulorewien/Prototype-Selection-for-Nearest-Neighbors

Method	Clusters	Samples	Accuracy	Max Accuracy	Min Accuracy
K-Means Clustering	10	500	0.7167	-	-
	10	1000	0.7552	-	-
	10	5000	0.8326	-	-
	10	10000	0.8867	-	-
	10	15000	0.9051	-	-
	10	20000	0.9207	-	-
	10	25000	0.9305	-	-
	10	30000	0.9379	-	-
	50	500	0.8521	-	-
	50	1000	0.8645	-	-
	50	5000	0.9057	-	-
	50	10000	0.9205	-	-
	50	15000	0.9274	-	-
	50	20000	0.9334	-	-
	50	25000	0.94	-	-
	50	30000	0.9436	-	-
	100	500	0.8795	-	-
	100	1000	0.8937	-	-
	100	5000	0.9218	-	-
	100	10000	0.9336	-	-
	100	15000	0.9388	-	-
	100	20000	0.9431	-	-
	100	25000	0.9462	-	-
	100	30000	0.9486	-	-
	150	500	0.8921	-	-
	150	1000	0.9023	-	-
	150	5000	0.9267	-	-
	150	10000	0.937	-	-
	150	15000	0.9419	-	-
	150	20000	0.946	-	-
	150	25000	0.9501	-	-
	150	30000	0.9522	-	-
	200	500	0.89	-	-
	200	1000	0.906	-	-
	200	5000	0.9282	-	-
	200	10000	0.9392	-	-
	200	15000	0.945	-	-
	200	20000	0.9491	-	-
	200	25000	0.9515	-	-
	200	30000	0.9547	-	-
	250	500	0.8982	-	-
	250	1000	0.9093	-	-
	250	5000	0.935	-	-
	250	10000	0.9416	-	-
	250	15000	0.9459	-	-
	250	20000	0.9476	-	-
	250	25000	0.9503	-	-
	250	30000	0.9528	-	-
	-	-	-	0.9547	0.7167

Table 2: K-Means Accuracy

Method	Samples	Accuracy	Max Accuracy	Min Accuracy
	500	0.8574	-	-
	1000	0.8882	-	-
	5000	0.9344	-	-
Baseline	10000	0.9504	-	-
	15000	0.955	-	-
	20000	0.9587	-	-
	25000	0.9632	-	-
	30000	0.9615	-	-
	-	-	0.9632	0.8574

Table 3: Baseline Accuracy

Method	Samples	Accuracy	Max Accuracy	Min Accuracy
	500	0.8032	-	-
	1000	0.8782	-	-
	5000	0.9166	-	-
CNN	10000	0.9485	-	-
	15000	0.9485	-	-
	20000	0.9485	-	-
	25000	0.9485	-	-
	30000	0.9485	-	-
	-	-	0.9485	0.8032

Table 4: CNN Accuracy

Method	Samples	Accuracy	Max Accuracy	Min Accuracy
	500	0.8451	-	-
	1000	0.8859	-	-
	5000	0.9364	-	-
IDDS	10000	0.9513	-	-
	15000	0.958	-	-
	20000	0.9612	-	-
	25000	0.9608	-	-
	30000	0.9674	-	-
	-	-	0.9674	0.8451

Table 5: IDDS Accuracy

Method	Samples	Accuracy	Max Accuracy	Min Accuracy
	500	0.8201	-	-
	1000	0.9095	-	-
	5000	0.9371	-	-
Ensemble	10000	0.9503	-	-
	15000	0.9529	-	-
	20000	0.9597	-	-
	25000	0.9592	-	-
	30000	0.9645	-	-
	-	-	0.9645	0.8201

Table 6: Ensemble Accuracy

Method	Samples	Accuracy	Max Accuracy	Min Accuracy
Improved Ensemble	500	0.8982	-	-
	1000	0.9093	-	-
	5000	0.935	-	-
	10000	0.9517	-	-
	15000	0.9566	-	-
	20000	0.961	-	-
	25000	0.9629	-	-
	30000	0.9676	-	-
	-	-	0.9676	0.8982

Table 7: Improved Ensemble Accuracy