Focused Subset Selection in Nearest Neighbor Classification

Joshua Chen
UC San Diego
CSE 251A
jpc005@ucsd.edu

Abstract

The Nearest Neighbor (NN) classification model, known for its simplicity and effectiveness, faces significant challenges in terms of training time and memory usage due to its nature of memorizing the entire training dataset. This paper addresses these challenges by proposing an innovative approach for selecting a focused subset of the training data. My method aims to drastically reduce the training dataset size while maintaining high classification accuracy. I demonstrate the effectiveness of our approach through extensive experiments on the MNIST dataset

1 Introduction

The main goal is to find a way to down sample to size n from the entire training set so that the training time for the nearest neighbor could be reduced while still able to remain the accuracy of the original training set. To achieve the goal, I've implemented multiple ways:

- K-Means Clustering 1: Split the dataset into 10 clusters (as there are 10 digits) and take an equal amount of datapoints from each cluster as the subsample
- K-Means Clustering 2: Split the dataset into n clusters and take one datapoint from each cluster as the subsample
- Modified Active Learning: Randomly select datapoints and run a classification model and select the most uncertain datapoints.
- Modified K-Means: Subsample proportionately to training, identify digits that are likely to be misclassified and apply K-means to clustered digits

I found that the best method was the Modified K-means method, as it maintained high accuracy.

2 Methods

My study commenced with the implementation of two rudimentary subsampling methods before progressing to more advanced technique tailored to the MNIST dataset. The first approach involved a simplistic, random selection from the entire dataset. While straightforward, this method displayed significant limitations, particularly as the subsample size (M) was reduced, leading to an unrepresentative subset of the entire database. Recognizing this flaw, I explored an alternative approach, utilizing K-means, an unsupervised clustering algorithm used to partition a dataset into K clusters based on their similarity. Below is the process of K-Means:

Algorithm 1 K-Means

```
Randomly initialize K cluster centroids, denoted as $c_1, c_2, ..., c_k$
**Assignment**:
**for** each data point $x_i$ in training **do**
    Assign it to the nearest cluster $c_j$
    j is determined by $j = argmin_k||x_i - c_k||^2$
**end for**
**Update**:
**for** each cluster k **do**
    calculate the new centroid $c_k$ with the formula
    $c_k = \frac{1}{S_K} \sum_{x_i in S_k} x_i$
**end for**
Repeat Assignment and Update until convergence criteria:
    • Minimal change in cluster assignments
    • Fixed number of iterations
```

The second method applied K-means with M clusters, returning the data point closest to the centroid (calculated with equation 1) of each cluster. Although this resulted in high accuracies across all tested M, it took a significantly long time to train.

$$distance = \sqrt{\sum_{j=1}^{784} (x_{ij} - c_{ij})^2} \tag{1}$$

Subsequently, I employed another K-Means clustering approach with K = 10, aligning with the 10 unique digit labels in the MNIST dataset. The initial hypothesis was that each cluster would predominantly correspond to a different digit, allowing for a balanced and representative subsample by choosing an equal number of points nearest to each cluster's centroid (described in algorithm 2, with the nearest calculation as equation 1). However, this approach faced a significant challenge when dealing with the MNIST dataset. The unique pixel representations of MNIST digits, while sometimes distinct to the human eye, could exhibit surprising similarities when viewed from a pixel-based perspective. This similarity posed a substantial obstacle for clustering algorithms.

In practice, visually distinct digits might cluster together due to subtle similarities in their pixel representations. Conversely, some digits with apparent visual similarities might end up in different clusters. This phenomenon highlighted the complexity of the pixel-based relationships between different digits, making it difficult to achieve a clear one-to-one mapping of clusters to distinct digits when clustering into 10 clusters.

To illustrate this challenge, consider Figures 1 and 2, representing digits 7 and 9, respectively. Despite their distinct appearances to the human eye, these digits can share subtle pixel-level similarities that can lead to unexpected clustering outcomes. Consequently, clustering to 10 clusters did not guarantee that each cluster predominantly contained a single distinct digit, thus affecting the accuracy of our subsampling approach.

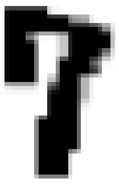
Figure 1: 7

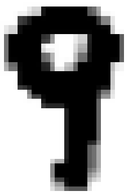
Figure 2: 9

Algorithm 2 K-Means Clustering 1

```
Perform K-Means Clustering with $K = 10$
Initialize an empty subsample list
**for** each cluster **do**
    • Select M/10 closest data points to the center
    • Add these points to the subsample list
**end for**
Return subsample list
```

To refine my methodology, I analyzed the clusters formed from the K-means clustering with k = 10. This analysis revealed patterns in digit grouping; for example, digits 0, 1, 2, and 6 tended to form individual clusters, while digits 4, 7, 9; 3, 5, and 8 were more likely to cluster together. Furthermore, upon analyzing the labels, I've realized that each labels are slightly unevenly sampled. Based on this insight, I came up with an enhanced K - Means algorithm as shown in Algorithm 3.

Algorithm 3 Enhanced K-Means Clustering

```
 1: Perform initial K-Means Clustering on MNIST dataset with $K = 10$.
 2: Manually analyzes clusters to identify which labels are correctly clustered together and which are not.
 3: Calculate the ratio of each label in the entire dataset:
    $ratio_{label} = \frac{occurrenceoflabel}{totalnumberofdatapoints}$.
 4: Determine the number of data points for each label needed in the sub sample based on the calculated ratios: $n_{label} = ratio_{label} \times M$.
 5: Adjust the $n_{label}$ for labels that tend to cluster together by reducing the number for labels that mostly cluster alone, and increasing it for those that cluster with others.
 6: For labels that cluster mostly by themselves:
 7: **for** each label **do**
        • Randomly select 90% of $n_{label}$.
        • apply a K-means with k = 10 on all data that correspond to this label
 8:     **for** each cluster **do**
        • Select $1\%$ of $n_{label}$ from each cluster
        • Add these points to the subsample list
 9:     **end for**
10: **end for**
11: For labels that tend to cluster together,
12: **for** labels that cluster together **do**
        • Add the data points together
        • Apply K-Means with k = $\sum n_{label}$
13:     **for** each cluster **do**
        • Select the closest point from each cluster
14:     **end for**
15: **end for**
16: Combine the selected data points from steps 6 to 14 to form the final subsample.
```

This approach allowed me to effectively down sample digits that already tend to cluster together and are less likely to be confused with other digits, allocating more space to digits prone to mis-classification. For labels that predominantly cluster by themselves, a strategic decision was made to randomly select 90% of the required data points directly. For the remaining 10%, a further K-means clustering with k = 10 was applied to account for the risk of selecting only the 'typical' representations of a digit. For labels that tended to cluster together, the algorithm employed K-means with k = number of data in these clusters. The final subsample offers a more comprehensive and representative sample of the MNIST dataset. However, this approach still takes quite a while to sample as it creates a K-means with a high amount of clusters and has to search through each cluster, although the time it took was noticeably less than the original approach that simply applied K-means with M clusters.

In addition to applying K-means, I've also applied a modified active learning approach, which employs a logistic regression model, initially trained on a subset of the data to ensure representativeness. The algorithm iteratively selects new samples for labeling based on an uncertain criterion. The uncertainty is calculated with the entropy formula in Equation 2 and a p close to 0.5 is considered uncertain.

$$P(y=1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_n x_n)}} \quad (2)$$

In each iteration, the model is updated with an augmented training set that includes newly added uncertain samples. This refinement continues until a predefined number of samples have been selected or the amount of samples required has been reached. This approach is a slight modification of studies by Settles (2010) [1]. The algorithm is shown in Algorithm 4.

Algorithm 4 Active Learning

```
Set M = desired size of condensed set
Randomly select a small subset from training set
Use the subset to train an initial model
While the size of the set is less than M:
    • use the model to predict on training set
    • identify the samples where the model's predictions are most uncertain
    • add the uncertain samples to the condensed set
    • retrain the model on the updated condensed set.
    • repeat until the condensed set reaches the size M
```

3 Results

In the base case, employing 1-Nearest Neighbor (1NN) with the entire dataset resulted in an accuracy of 0.9691, with a corresponding CPU time of 46.4 seconds.

For all methods, the experiments were repeated 10 times to account for potential randomness. The following are the mean averages of accuracy and runtime for each method:

M	Avg Acc (%)	Std dev Acc (%)	Avg time (s)
100	0.71283	0.02516	0.189
1000	0.887	0.00344	2.07
5000	0.93545	0.001733	8.1
10000	0.9494	0.001723	7.9

Table 1: Random

M	Avg Acc (%)	Std dev Acc (%)	Avg time (s)
100	0.64463	0.01563	2.8
1000	0.7515	0.00658	3.76
5000	0.82537	0.0084	6.8
10000	0.86909	0.0089	10.1

Table 2: K - Means (K=10)

M	Avg Acc (%)	Std dev Acc (%)	Avg time (s)
100	0.70295	0.01198	0.0875
1000	0.8842	0.00295	0.711
5000	0.93596	0.00204	4.3
10000	0.94796	0.0018	10.3

Table 3: Modified Active Learning

M	Avg Acc (%)	Std dev Acc (%)	Avg time (s)
100	0.84719	0.005584	45.9
1000	0.92582	0.001717	157.7
5000	0.95037	0.001269	421.1
10000	0.95754	0.001065	898

Table 4: K-Means K = M

M	Avg Acc (%)	Std dev Acc (%)	Avg time (s)
100	0.74251	0.01588	9
1000	0.90231	0.002869	34.8
5000	0.94259	0.001798	77.8
10000	0.95071	0.001266	148

Table 5: Enhanced K-Means

4 Critical Evaluation

My methods, notably enhanced K-means and K-means=M, outperformed random selection. However, the active learning loop's performance was unexpectedly similar to random selection, which may be due to potential modifications and uncertainties in my implementation. The most accurate method was K-means=M, but its extended processing time is a significant drawback. The enhanced K-means, while slightly less accurate, drastically reduced training time, making it the most effective method in my study. For future improvements, particularly in the active learning loop, I plan to explore alternative classifiers and enhance the selection process using density-weighted methods, which consider both model uncertainty and regional density of samples, and query by committee, involving multiple models to identify data points with the highest disagreement for labeling. Further optimization of enhanced K-means, especially for handling distinct digits that tend to cluster together, could also improve the model while reducing training time. Some potential possibilities involve applying the modified active learning on the combined data of digits that get clustered together. Additionally, using Edited Nearest Neighbor (ENN) and Condensed Nearest Neighbor (CNN) for initial downsampling is another avenue. My research indicates that both CNN and ENN downsample to a set amount, allowing further downsampling to size M using the methods I've described, potentially improving accuracy.

Note: in my experimentations, when I tried implementing CNN, I encountered memory issues or excessively long processing times without achieving a result.

References

[1] Settles, B. (2010). "Active Learning Literature Survey." University of Wisconsin–Madison.