Prototype selection for nearest neighbor

Vivekanand Gyanchand Sahu
PID: A69027331
vsahu@ucsd.edu

Abstract

This paper presents a novel method for enhancing nearest neighbor classification speed by employing bisecting k-means clustering to select representative prototypes from the training set. I define prototype effectiveness through intra-cluster cohesion and inter-cluster separation, formalizing these properties for automatic selection. The implemented algorithm demonstrates improved test accuracy on the MNIST dataset using 1-NN classification.

1 Introduction

In the pursuit of enhancing nearest neighbor classification efficiency, this paper addresses the challenge of selecting an optimal subset of prototypes from the training set. By strategically choosing a representative subset instead of using the entire training set, the classification process can be expedited. However, the key lies in devising a robust strategy to ensure the selected prototypes contribute to superior test accuracy. I propose employing the Bisecting K-Means clustering method, followed by output class-specific data sampling. This process not only refines prototype selection but also incorporates class-specific information. My approach aims to automatically formalize properties that render a set of examples effective for nearest neighbor classification, with a subsequent implementation and evaluation on the MNIST dataset.

2 Implementation Details

2.1 Bisecting K-Means (BKM)

The Bisecting K-Means algorithm, an extension of traditional K-Means, exhibits distinctive logic for clustering data points. Its fundamental objective is to iteratively partition the dataset into K clusters by recursively bisecting the cluster with the highest intra-cluster variance. This process involves selecting the cluster with the maximum variance, splitting it into two sub-clusters, and repeating until the desired number of clusters is attained. The algorithm optimizes the cluster splits by minimizing the sum of squared distances between data points and their respective cluster centroids.

2.2 Bisecting K-Means Steps

One of my passions lies in the intricacies of data science, where *D* represents a dataset with *n* data points and *K* signifies the desired number of clusters. The initial stage entails the random selection of *K* centroids, denoted as C_1 and C_2.

Initialization: C_1 and C_2 $\leftarrow$ Randomly selected initial centroids.

Assignment: For each data point x_i, assign it to the cluster with the nearest centroid C_j based on Euclidean distance: $argmin_j \|\mathbf{x}_i - \mathbf{C}_j\|^2$

Update Centroids: Update the centroids by computing the mean of the data points within each cluster:
$\mathbf{C}_j \leftarrow \frac{1}{number of points in cluster_j} \sum_{cluster_j} \mathbf{x}_i$

Bisecting: Identify the cluster with the highest intra-cluster variance and bisect it into two sub-clusters.

Repeat: Iterate the update and bisecting until the desired number of clusters (K) (10 in case of MNIST) is achieved.

The iterative nature of the Bisecting K-Means algorithm ensures a refined partitioning of data into clusters, contributing to effective clustering outcomes.

2.3 Pseudocode:

The BisectingKMeans function accepts input data and the desired size for the output sample. It then applies the bisecting k-means clustering algorithm to partition the input data, providing the clustered

data and the distances of each sample from their respective centroids as outputs. This facilitates a comprehensive understanding of data distribution within clusters and serves as valuable information for further analysis and sampling considerations.

data = ImportMnistData

function BISECTINGKMEANS (data, M)
$nClusters \leftarrow 10$
$labels, centroids, distances \leftarrow BisectingKMeans(data, nClusters)$
$output \leftarrow Combine(data, labels), distances$
end function

In the context of the MNIST dataset, after applying the Bisecting K-Means clustering, the logic for selecting a specific number of samples from each cluster is driven by the aim to ensure a balanced representation of digits/classes in the final training sample. By calculating the minimum distance of each data point to its corresponding cluster centroid, I prioritized proximity and diversity. The selected samples are then proportionally chosen from each cluster to maintain an equitable distribution across digit classes. This strategy ensures that the K-nearest neighbors (KNN) classifier is trained on a subset that reflects the variability inherent in each digit category, enhancing classification robustness and accuracy.

Selecting M data points by picking M/10 data points from each cluster Training K-NN for each M

$fullOutput \leftarrow Combine(output, distances)$

for each M in $MValues$ **do**:
$croppedFinal \leftarrow EmptyDataFrame()$
$uniqueClusters \leftarrow GetUniqueClusters(fullOutput)$
for each $cluster$ in $uniqueClusters$ **do**:
$croppedFinal \leftarrow AppendClosestRows(fullOutput, M/10Rows)$
end for
$applyKNN(croppedFinal)$
show results
end for

3 Experiment Results

The presented table provides a comprehensive overview of accuracy values associated with diverse sample sizes (M), ranging from 500 to 15000. BKM is Bisecting K Means. The analysis contrasts random sampling against samples obtained post-application of the Bisecting K-Means algorithm (designated as BKM).

# of Samples	BKM	Random
500	92.11%	78.5%
1000	97.33%	80.0%
5000	96.80%	91.2%
10000	97.73%	93.07%
15000	97.60%	95.73%

Table 1: Accuracy of Sampling Strategies for Different Data Samples

Accuracy, a crucial performance metric, was computed using the formula:

$$Accuracy = \frac{CorrectlyClassifiedSamples_{test}}{TotalSamples_{test}}$$

This ratio serves as a pivotal metric, meticulously capturing the efficacy of the classification process. The experimentation process involves a systematic iteration, dynamically adjusting the number of data points in each run within a loop. This methodical exploration is designed to offer a nuanced and comprehensive understanding of accuracy trends, particularly concerning varying subset sizes. The outcomes shed light on the discernible impact of sample size on classification accuracy, providing detailed insights into the potential advantages conferred by the Bisecting K-Means approach when compared to the conventional method of random sampling.

In my experimental analysis, I meticulously assessed the computational efficiency of the k-nearest neighbors (KNN) algorithm across varying sample sizes. I measured the time duration required for KNN execution on randomly sampled data with values ranging from 500 to 15000. Additionally, I conducted the same analysis on data sampled after the application of the bisecting k-means algorithm. This comprehensive evaluation aimed to scrutinize the temporal performance of KNN for different dataset sizes, comparing the outcomes of random sampling against the more strategic approach of sampling after employing bisecting means.

4 Evaluation and Inferences

The experimental results reveal a substantial disparity in accuracy between the models employing ran-

# of Samples	Time (sec)	Accuracy
BKM 500	0.06	92.11%
BKM 1000	0.09	97.33%
BKM 5000	0.39	96.80%
BKM 10000	0.82	97.73%
BKM 15000	1.45	97.60%
UnSampled 70000	13.8	96.93%

Table 2: Comparison of Execution Time and Accuracy Between the Models Under Consideration

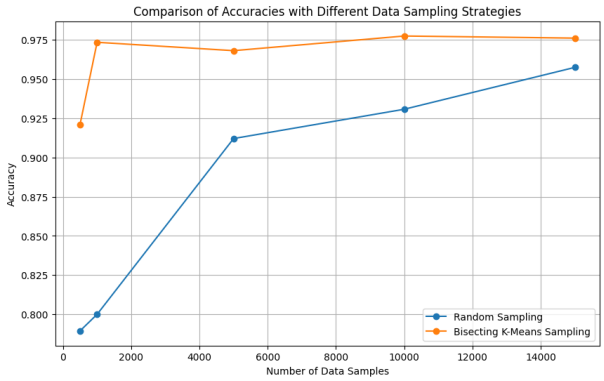

Figure 1: Comparison of Accuracies with Different Data Sampling Strategies

dom sampling and sampling after the application of bisecting k-means, particularly in the initial stages where the dataset is limited. Notably, as the volume of data increases from 500 to 15000 samples, both models exhibit a progressive enhancement in accuracy. Crucially, the observed trend indicates a convergence of accuracies between the random sampled model and the model employing sampling after bisecting k-means clustering, suggesting a diminishing difference with the expansion of the dataset. This trend underscores the efficacy of sampling after applying bisecting k-means, especially when confronted with limited data, demonstrating its ability to outperform random sampling. The diminishing discrepancy in accuracies as data size increases implies that the strategic selection of prototypes using bisecting k-means contributes significantly to model performance. These findings underscore the suitability of the proposed technique, offering a compelling solution for improved accuracy in scenarios characterized by limited data availability. The robust performance of the sampling after bisecting k-means approach substantiates its merit as a valuable tool for enhancing model accuracy, particularly in resource-constrained environments.

A notable distinction in time consumption is evident between the random sampling model and the model employing sampling after bisecting k-means. Surprisingly, the time required for random sampling closely parallels that of the bisecting k-means model, showcasing comparable efficiency. As the dataset size escalates from 500 to 15000 samples, both models exhibit an increase in processing time, albeit accompanied by a corresponding rise in accuracies. The optimal choice between the two approaches hinges on the desired accuracy threshold and the urgency of results. The comparable time efficiency highlights the flexibility of both methodologies, allowing the selection of the most suitable strategy based on specific requirements.

5 Limitations and future scope

While Bisecting K-Means proves effective in clustering, a limitation arises when selecting the number of clusters (k). The algorithm inherently divides the dataset into two subsets at each iteration. Consequently, if the desired number of clusters is not a power of 2, uneven splitting occurs. In the context of MNIST, where k is 10, two clusters may have disproportionally fewer samples. This uneven distribution poses a challenge for ensuring representativity across all classes. A potential future enhancement lies in optimizing the algorithm for scenarios where the number of clusters aligns with powers of 2, such as binary digit encoding, thereby ensuring more balanced and efficient clustering outcomes.

6 Reference

1. Scikit-Learn BisectingKMeans Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.BisectingKMeans.html

2. K-Nearest Neighbor Classification Lesson Sample: https://customers.pyimagesearch.com/lesson-sample-k-nearest-neighbor-\classification/