K-Means Clustering based Prototype Selection for Nearest Neighbor Classifier

Ambareesh S N Jayakumari
Department of Electrical and Computer Engineering
University of California, San Diego
San Diego, CA 92093
asreekum@eng.ucsd.edu

Abstract

Speeding up nearest neighbor classification can be done by replacing the training set by a carefully chosen subset of "prototypes". For choosing prototypes from the training set, a K-means clustering based algorithm was experimented with. The prototype chosen was used for 1-NN based classification. The algorithm was implemented and tested on the MNIST data set.

1 Description

To create a prototype of the data set with the best representation of the original training set - the ratio of the labels have to be kept intact. Also the data points of the prototype have to be as close to the different type of data points that belong to the same label in the original, so as to give an accurate representation. Clustering is a method for achieving such a simplified representation. K-means clustering algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible. After figuring out how many prototype points are needed for each label (in accordance to the ratio in the original data set), those many clusters are formed and hence their centroids combined will form the needed prototype.

2 Algorithm Model

Figure 1 is a pictorial representation of the algorithm for the case where M=4.

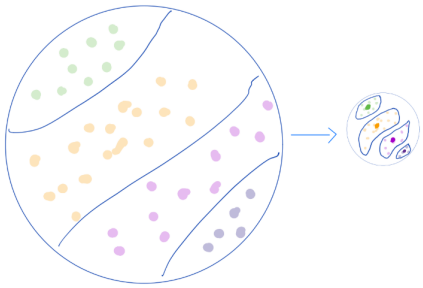

Figure 1: Prototype generation from labelled original data using K-means clustering

3 Pseudocode

Input: Labelled Training Set, M - Prototype Size
Returns: Prototype set of size M.

1. Retrieve ratio of each label in the entire original data-set:
   - For each label:
     - ratio_of_label[label] = Occurrence of label / Total number of data points
2. Calculate number of data points for each label needed in M, according to the ratio from step (1):
   - For each label:
     - number_of_datapoints_for_each_label[label] = ratio_of_label[label] * M
3. Perform K_means clustering on original data to retrieve required number of centroids as the number required for each label :
   - For each label:
     - Perform K-Means clustering on data points of label and return "number_of_datapoints_for_each_label[label]" number of centroids
4. Combine centroids obtained from step 3, for all labels
5. The consolidated centroids from Step 4 are the representations of the different type of original data points which can be best represented by a prototype of size M

4 Experimental Results

1-NN classification was performed on the prototypes generated from the algorithm. As control, random uniform-random sampling was used to sample the same number of prototype elements, to compare the difference.

To account for the randomness for each case of M, 100 iterations were run, and the average results and the deviation in accuracy across iterations were recorded. Cases considered for M = 100, 1000, 5000, 10000.

The results are collected in Table 2 and plotted in Figure 3.

M Values	Uniform Random_Prototyping		K Means Clustering based Prototyping	
	Mean Accuracy	Std_Deviation	Mean Accuracy	Std_Deviation
100	70.026	0.941713332	92.448	0.04621688
1000	88.616	0.319662322	95.824	0.018547237
5000	93.734	0.173158887	96.676	0.107443008
10000	94.988	0.108885261	96.896	0.10892199

Figure 2: Results : Proposed algorithm vs Random Sampling, for MNIST

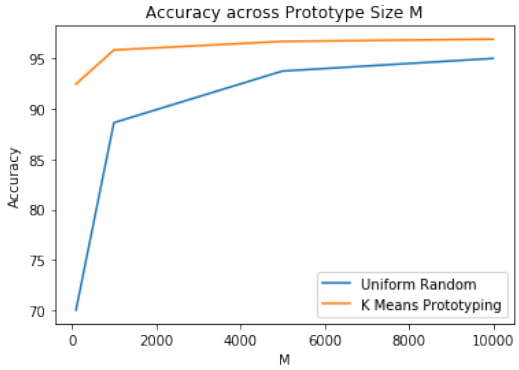

Figure 3: Results : Proposed algorithm vs Random Sampling, for MNIST

5 Critical evaluation

5.1 Analysis

Clear improvement over random sampling has been observed in the results. K-means clustering makes clusters of similar data points and the number of such clusters are proportional to their representation in the original training dataset.

- Accurate : Accuracy is higher compared to random sampling, and especially pronounced in smaller numbers of M (prototype size). Random sampling is not able to capture the properties of the data set accurately.
- Stable : The standard deviations of the results from the proposed approach is much smaller compared to those of the random sampling, because the solution is more stable than random sampling.
- Dealing with data set imbalance : The method would prove even better results in cases where the data set has imbalanced data. Say medical data (For e.g, if the data is about incidence of cancer, there will be way more data points of healthy patients than diseased patients, and the data set will be skewed). Random sampling will not be accurate in terms of drawing a prototype, whereas the proposed approach keeps the inherent skewness of the data set, as well as giving a proper representation of the data

5.2 Further scope for improvement

Variances of different dimensions of the data was not considered by the given approach. K-d trees based methods would be an interesting route to try, as the split of data will be based on variances over features of the data. As another measure to incorporate variance/spread of the data PCA could also be experimented with, for prototyping.