# NEAREST NEIGHBOR CLASSIFIERS

# WITH IMPROVED ACCURACY AND EFFICIENCY

by

SINCHAN BHATTACHARYA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2017

Acknowledgements

I would like to express my sincere gratitude to my advising professor Dr. Michael T Manry for his continuous support throughout my master study and related research, for his patience, guidance and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

I would also like to thank Dr. Venkat Devarajan and Dr. R Stephen Gibbs for their time, valuable comments and being a member of my thesis defense committee.

Finally, I must express my sincere gratitude to my family for their love and support. I dedicate this thesis to my parents, Mr. Sanjay Bhattacharyya and Mrs. Sumita Bhattacharya, my brother, Mr. Sameek Bhattacharya and my fiancée Ms. Rituparna Sinha Roy.

December 6, 2017

Abstract

NEAREST NEIGHBOR CLASSIFIERS

WITH IMPROVED ACCURACY AND EFFICIENY

SINCHAN BHATTACHARYA, MS

The University of Texas at Arlington, 2017

Supervising Professor: Dr. Michael T. Manry

Nearest Neighbor algorithms are non-parametric algorithms that use distance measure techniques for classification and regressions. This thesis uses the method of pruning to improve accuracy and efficiency of a nearest neighbor classifier and also states the different stages the pruning algorithm can be applied and shows the best stage for pruning which gives the maximum accuracy. The performance of the classifier is shown to be better than other improved nearest neighbor classifiers. A fast method of finding the optimal *k* in a k-nearest neighbor classifier is proposed in the thesis. A method of optimizing the distance measure using a second order training algorithm in a k-nearest neighbor algorithm is also proposed in this thesis which results to better accuracy than the traditional k-nearest neighbor classifier.

# Chapter 1

# INTRODUCTION

This chapter gives an overview of the basic concepts of feature extraction, classifiers and its application. It also explains nearest neighbor classifier and its properties. In section 1.4 the objective of the thesis is defined.

## 1.1 Feature Extraction

Feature extraction is the process of deriving values (features) from measured data , which is intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction.[49]

When the input data to an algorithm is too large to be processed and it is suspected to be redundant, then it can be transformed into a reduced set of features (also named a feature vector). Determining a subset of the initial features is called feature selection. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data. The features extracted are often beneficial to mitigate the computational complexity and improve the accuracy classifiers.[51]

The data sets used in the experiments of the thesis are features extracted from real measured data.

## 1.2 Classifiers

In machine learning and statistics, classifications is the problem of identifying to which a set of categories a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. Classification is an example of pattern recognition [1]. Good examples of classification are assigning emails into spam and non-spam category, classification of different blood type, classifying number set from hand-written digit images. Algorithms that implements classification are called classifiers.

The kind of classifier needs to be implemented depends upon the dataset needed to be classified. For non-complex datasets linear and generalized linear classifiers like the perceptron [2], support vector machines [3] etc. are used. For highly complex datasets non-linear classifiers like multi-layer perceptron [4], random forest [5], decision tree [6] etc. are used. All these classifiers are called parametric classifiers as because the classifiers have parameters (also known as weights) which can be adjusted according to the training dataset to achieve higher efficiency of classification.

## 1.3 Nearest Neighbor Classifiers

The nearest neighbor classifier is used effectively for pattern recognition [7, 8, 48], computer vision and image recognition [9, 10, 49], text classification [11, 13], event recognition [14, 71], ranking models [15, 72, 73, 74], face recognition [56, 75, 76], intrusion detection [57, 77, 78, 79] and object recognition [16, 80, 81, 82]. This kind of classifiers achieves consistent high performance without priori assumptions about the distributions from which the training data samples are drawn.

Nearest neighbor classifiers are a type of instance based learning or lazy learning classifiers. In this kind of classifiers generalization beyond the training data is delayed until a new test data is encountered. Hence, the Nearest Neighbor classifiers are consistent non parametric estimators. The term non-parametric means that there is no prior knowledge of the statistical distribution of the data to be classified. During testing, the distance between the input vectors (pattern) and each cluster's center vector is calculated. The estimated class of the input vector is that of the nearest center vector. As the number of training patterns tends to infinity, classifiers based on nearest neighbor rule converge to the corresponding Bayes estimate.

## 1.4 Objective of this thesis

The thesis proposes an improved nearest neighbor algorithm which does pruning on the input data set to increase the accuracy and efficiency of the nearest neighbor classifier. The pruning algorithm finds the input vectors which outliers for the given data set or which does not affect the accuracy of the classifier and removes them from the input data set. The reduction of the input data set increases the accuracy of the classifier and also decreases the time taken by the classifier to predict correct class labels in application, making the classifier more efficient. The thesis also proposes a new method of finding the best value of *k* for a k-nearest neighbor classifier by making one single pass through the training data set, thus, helping in finding the unknown important parameter *k* efficiently. It also states an improved k-nearest neighbor classifier which uses a clustering algorithm to reduce the input data set and then perform assigning weights to input features and then optimizing them using a second order training algorithm to give important input features higher weights and less important and noisy features lower weights.

Chapter 2, reviews the structure and operations of the traditional nearest neighbor classifiers and its disadvantages. Chapter 3, explains previous works to improve the nearest neighbor classifier by implementing clustering on the input dataset to find center vectors for every class label and then applying distance measure optimization technique to differentiate between more important input features from less important and noisy input features and finally weigh them differently during classification and finally optimizing the center vector locations in input dimension space which directly minimizes the input-output mapping error. Chapter 4 introduces the method of pruning and how it can be implemented to the nearest neighbor classifier to improve both accuracy of prediction of correct class and efficiency of the classifier. Chapter 5 proposes a new efficient method of finding the best *k* for k-nearest neighbor classifier and also applies the method of clustering and distance measure optimization to the k-nearest

neighbor classifier. Chapter 6 presents the results of the various algorithms applied on several datasets.

# Chapter 2

# NEAREST NEIGHBOR AND k-NEAREST NEIGHBOR CLASSIFIERS

This chapter reviews the structure and operation of the traditional nearest neighbor classifier as well as the traditional k- nearest neighbor classifier. It also defines several distance measure techniques. A basic algorithm for k-means clustering is also explained along with implementing it with the Nearest Neighbor Classifier.

## 2.1 Data notation

The nearest neighbor classifier requires a dataset to compare to for predicting the correct class of an input pattern. This dataset is called the training data. The training vector is denoted by $\boldsymbol{x}_{tp}$ where $\boldsymbol{x}_{tp}$ is the $p^{th}$ $N$-dimensional training vector. The total number of training patterns in the training data is represented by $N_v$. The input test vector is represented by by $\boldsymbol{x}_p$, where $\boldsymbol{x}_p$ is the $p^{th}N$-dimensional test vector. The total number of classes is represented by $N_c$. The class labels are represented by $i$.

The nearest neighbor classifier compares the input test vector with the training data, which can be also referred to as example vectors, to predict the correct class of the classifier. The example vectors are represented by $\boldsymbol{m}_{ik}$, where $\boldsymbol{m}_{ik}$ is the $k$th example vector of the $i$th class. $\boldsymbol{m}_{ik}$ can be the entire training data, which is the case of the regular nearest neighbor,

$$\boldsymbol{m}_{ik} = \boldsymbol{x}_p$$

Or the example vectors, $\mathrm{m_{ik}}$, can be center vectors of the $i^{th}$ class, found by the method of clustering (discussed in details in section 2.5) $N_v(i)$ patterns, where $N_v(i)$ is the total number of training patterns in the $i$th class,

$$\mathrm{N_v} = \sum_{\mathrm{i}} \mathrm{N_v(i)}$$

The minimum distance between test vector $\boldsymbol{x}_p$and the $k$th example vector of the $i^{th}$ class, $\boldsymbol{m}_{ik}$, is denoted by $d_i$

$$\mathrm{d_i} = \min_{\mathrm{k}} \mathrm{d}(\mathbf{x_p}, \mathbf{m_{ik}})$$

where $\mathrm{d}(\mathbf{x_p}, \mathbf{m_{ik}})$ is the distance operator discussed in the following section, section 2.6

## 2.2 Structure of the Nearest Neighbor Classifier

The training data of '$K$' $N$-dimensional cluster center vectors, $\boldsymbol{m}_{ik}$, where $\boldsymbol{m}_{ik}$ is the center vector of the$k^{th}$ cluster of the $i^{th}$ class. The test vector is denoted by $\boldsymbol{x}_p$, where $\boldsymbol{x}_p$ is the $p^{th}N$-dimensional test vector. The total number of classes is dented by $N_c$. The distance between $\boldsymbol{x}_p$and the closest center vector of the $i^{th}$ class is denoted by $d_i$.

Nearest neighbor classifier is an instance based classifier [42], unlike many other artificial learners, so, they do not extract any information from the training data during the learning phase. During classification, an unlabeled test pattern is classified by assigning the class of the nearest trained pattern. The distance metric function can be empirically chosen among the Euclidean [18], Minkowski [20] and Mahalanobis [21] methods among others, based on the training data and application [23]. The most commonly used distance metric for continuous variables is the Euclidean distance ($L_2$).

## 2.3 Basic Nearest Neighbor Classifier operation

In the $N_v$,$N$ - dimensional training patterns, each pattern is associated with the class label to which it belongs [96]. There can be two stages of operations involved during classification using nearest neighbor classifiers.

1. Clustering: The training data of $N_v$ patters and $N$-dimensional input space is divided into into $K$ clusters. This division of input patterns into $K$ cluster's center vectorscan be done by implementing clustering algorithms such as K – Means [91], SOM [92], DBSCAN [93] or EM [18]. The value of $K$ is selected in such a way such that each cluster has more similar patterns compared to patterns in another cluster.

2. Classification: Clustering of the training pattern gives the center vectors of each cluster which is saved to the memory. Each time a new test pattern needs to be classified, the distance between the test vector and the center vectors is calculated. The closest center vector from each class is determined as

$$\mathrm{d_i} = \min_{\mathrm{k}} \mathrm{d}(\mathbf{x_p}, \mathbf{m_{ik}})$$

where, $\mathrm{d}(\mathbf{x_p}, \mathbf{m_{ik}})$, is the distance of the $p^{th}$test pattern, $\boldsymbol{x}_p$, from the center vector, $\boldsymbol{m}_{ik.}d_i$ is the distance of the new test pattern from the closest center vector of $i^{th}$ class. The class membership of this new test pattern is then estimated as

$$\mathrm{i_c'} = \underset{\mathrm{i}}{\mathrm{argmin}}\, \mathrm{d_i}$$

where, $i_c'$ is the estimated class of the $p^{th}$ test pattern, $\boldsymbol{x}_p$.

## 2.4 Basic k – Nearest Neighbor Classifier Operation

The k – Nearest Neighbor differs from the Nearest Neighbor Classifier in the aspect of predicting the correct class for an input test label. In the k- Nearest Neighbor classifier, each of the $N_v$,$N$ - dimensional training patterns has got a class label associated with it.

Every time a input test vector $\mathbf{x_p}$ is provided to the Classifier the distance $\mathbf{d_p}$ between the input test vector $\mathbf{x_p}$ and all the training center vectors $\mathbf{m_{ik}}$ is computed. The *k* least distances are found which gives the *k* closest training center vectors, $\mathbf{m_{ik}}$. The class seen most in these *k* closest training vector is the predicted class for the input test vector, $\mathbf{x_p}$**.** The following theorem will describe the convergence of k- Nearest neighbor Classifier probability of error, $P_{e(k\text{-}NNC)}$.

**Theorem 2.1** [27, 28]: As *k* and ($N_v$/k) approach infinity, the k-Nearest neighbor Classifier can be viewed as an attempt to estimate the *a-posteriori* probabilities from the training sample and

$$\lim_{k,\frac{Nv}{k}->\infty} P_{e(k-NNC)} = P_{eB}$$

Among the *k* Nearest Neighbors $\mathbf{m}_{\mathrm{jm}}$, let *k(i)* be the number of neighbors $\mathbf{m}_{\mathrm{im}}$ from class *i*. Then,

$$\mathrm{i}_{\mathrm{c}}' = \operatorname{argmin}_{\mathrm{i}}\{\mathrm{k(i)}\} = \operatorname{argmin}_{\mathrm{i}}\{\mathrm{k(i)/k}\}$$

But,

$$\lim_{k\to\infty}\left\{\frac{k(i)}{k}\right\} = P(i|\mathbf{x})$$

from the *Strong Law of Large Numbers* [29] P(i|**x**) is the $B_3$ Bayes discriminant.

## 2.5 Generating center vectors

Clustering is the task of grouping a set of objects such that objects in the same group are more similar to each other in some sense than to those in other groups [18]. It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics [36].Center vectors can be generated using various algorithms like K – Means [91], SOM [92], DBSCAN [93] and EM [18]. The appropriate center vector generating algorithm is dependent on the individual dataset and the intended use of the results. The training dataset can be used in different ways for classification using nearest neighbor classifiers. For example,

1) Entire training dataset or patterns can be used as center vectors. This avoids the need for using any clustering algorithms. The number of training patterns can be large, even in the order of millions. Thus this method can cause huge memory

strains as the entire dataset needs to be cached in memory. Along with this, computing distance of a test pattern from all these training patterns can be computationally very expensive. However, the above method can prove to be useful if the number of training patterns is limited and testing time is not a matter of concern.

2) We can randomly choose few training patterns and use them as center vectors. This method also avoids the need for using any clustering algorithms and does not cause much memory strains as only center vectors need to be cached in memory as compared to the entire training dataset. Number of center vectors is much less than the number of training patterns. Since these randomly picked center vectors are not optimal,they may group dissimilar objects to each other and might even prove to be use less center vectors. This can lead to decreased performance.

3) K-Means clustering [91] is a popular vector quantization method that partitions $N_v$patterns into $K$ clusters where each pattern belongs to the cluster with the nearest mean. This results in partitioning of the data space into Voronoi cells [94].

Given a set of $N_v$ training patterns, $\boldsymbol{x}_p$, where each pattern has $N$-dimensional inputs. K- Means clustering aims to partition $N_v$patterns into $K$ clusters, so as to minimize the within-cluster sum of squares distances.

Algorithm Summary [95]

1. $i_t$ = 0, where $i_t$ = iteration number and $N_{it}$ = total number of iterations.
2. $i_t = i_t + 1$
3. Calculate center vector, $\boldsymbol{m}_{k,}$as
4. $\mathbf{m}_{\mathrm{k}} = \frac{1}{\mathrm{N_v(k)}} \sum_{\mathrm{p:m(p)=k}} \mathbf{x}_{\mathrm{p}}$,

where $m(p)$ equals the cluster number of the $p^{th}$ pattern and

$N_v(k)$equals thenumber of patterns in the $k^{th}$

Reclassify $\boldsymbol{x}_p$s, in one data pass. If $\boldsymbol{x}_p$belongs to the $k^{th}$cluster, $m(p)$equals $k.m(p)$ therefore specifies the cluster membership of the $p^{th}$pattern. If any clusters change and iterationnumber <$N_{it}$, go to step 2.

The K – Means clustering error, $E_{k-means}$, is

$$\mathrm{E_{k-means}} = \frac{1}{\mathrm{N_v}}\sum_{\mathrm{p=1}}^{\mathrm{N_v}} \mathrm{d}\left(\mathbf{x}_\mathrm{p}, \mathbf{m}_{\mathrm{m(p)}}\right) = \frac{1}{\mathrm{N_v}}\sum_{\mathrm{k=1}}^{K} \mathrm{E_k}$$

$$\mathrm{E_k} = \sum_{\mathrm{p:m(p)=k}} \mathrm{d}\left(\mathbf{x}_\mathrm{p}, \mathbf{m}_\mathrm{k}\right)$$

## 2.6 Example Distance Measures

Nearest neighbor algorithms calculates the distance between the new test pattern and the training vectors to estimate the class of this new test pattern. There are a variety of distance measures available.

### *2.6.1 Euclidean Distance*

Euclidean distance measure is by far the most common distance measure technique used. The associated norm is called the Euclidean norm [24]. Euclidean distance between the $p^{th}$ test pattern, $\boldsymbol{x}_p$ and $\boldsymbol{m}_{ik}$center vector is given by

$$\mathrm{d}\left(\mathbf{x}_\mathrm{p}, \mathbf{m}_\mathrm{ik}\right) = \| \mathbf{x}_\mathrm{p} - \mathbf{m}_\mathrm{ik} \| = \left(\sum_{\mathrm{n=1}}^{\mathrm{N}} \left(\mathrm{x_p(n) - m_{ik}(n)}\right)^2\right)^{½} \tag{2.1}$$

The square root is often not computed in practice, because the closest center vector will still be the closest, regardless of whether or not the square root is taken [25] and hence, by not computing the square root computational efficiency of the classifier is improved.

### 2.6.2 *Mahalanobis Distance Measure [23]*

Mahalanobis distance measure computes the distance between p[th] test pattern, $\boldsymbol{x}_p$ and the distribution, $D$. It is a multi-dimensional generalization of the idea of measuring how many standard deviations away $\boldsymbol{x}_p$is from the mean of *D*. If the point is at the mean of distribution, $D$, the distance is zero [41]. Mahalanobis distance measure can also be defined as the measure of dissimilarity between the test pattern, $\boldsymbol{x}_p$, and the training center vector, $\boldsymbol{m}_{ik}$, of the same distribution [26]. Mahalanobis Distance is unitless and scale-invariant, and takes into account the correlations of the data set.

$$\mathrm{d}(\mathbf{x}_\mathrm{p}, \mathbf{m}_\mathrm{ik}) = \sum_{\mathrm{n}=1}^{\mathrm{N}} \sum_{m=1}^{N} a_k(n,m)[x_p(n) - m_{ik}(n)][x_p(m) - m_{ik}(m)] \tag{2.2}$$

where,

$$\mathrm{a_k}(n,m) \in \boldsymbol{C}_k^{-1}, \quad \boldsymbol{C}_k = E[(\boldsymbol{x}_p - \boldsymbol{m}_{ik})(\boldsymbol{x}_p - \boldsymbol{m}_{ik})^T]$$

### 2.6.3 *Minkowski Distance Measure [24]*

It is a generalization of Euclidean distance and the Manhattan distance. The distance of order $v$ between the $p^{th}$ test pattern, $\boldsymbol{x}_p$ and $\boldsymbol{m}_{ik}$ center vector is given by

$$\mathrm{d}(\mathbf{x}_\mathrm{p}, \mathbf{m}_\mathrm{ik}) = \left( \sum_{\mathrm{n}=1}^{\mathrm{N}} (\mathrm{x_p(n)} - \mathrm{m_{ik}(n)})^{\mathrm{v}} \right)^{\frac{1}{\mathrm{v}}} \tag{2.3}$$

If $v = 2$, the Minkowski distance is equivalent to Euclidean distance($\mathrm{L}_2$).

### 2.6.4 *Weighted Euclidean Distance*

Weighted Euclidean distance is a modified version of Euclidean Distance that incorporates weights in the distance calculation in such a way that distance measure for each input element $x_p(n)$ is multiplied with the corresponding weights element $w(n)$.

The weighted Euclidean distance between the $p^{th}$ test pattern, $\boldsymbol{x_p}$ and center vector, $\boldsymbol{m_{ik}}$, is given by

$$\mathrm{d}\left(\mathbf{x}_{\mathrm{p}}, \mathbf{m}_{\mathbf{ik}}\right) = \sum_{\mathrm{n}=1}^{\mathrm{N}} \left(\mathrm{w}(\mathrm{n}) \cdot \left(\mathrm{x}_{\mathrm{p}}(\mathrm{n}) - \mathrm{m}_{\mathrm{ik}}(\mathrm{n})\right)\right)^2 \tag{2.4}$$

In this thesis weighted Euclidean distance measure is used to compute the distances in the classifier, where the weights are a measure of importance for the corresponding input element. Optimal tuning of the weights are done in such a way so that the less important and noisy features are assigned smaller weights and important features are assigned larger weights. This causes the features with less importance and noisier to participate less in computing distance compared to the highly discriminative features. Thus this helps in solving the problem of misclassification due to noisy features.

***Weight Initialization***

Weights are one of the most important parameters that determine the performance of any classifier. The training error convergence, performance and training hyper-parameters like learning rates etc. depend heavily on initial weights. If the weights are too small the gradients would in turn be very small in magnitude and thus the classifier will take more time to converge or might not converge to a desired error value. On the other hand if weights are very large then their gradients would be very large in magnitude too. So a small change in weight update can lead to large change in the output, thus a very small learning rate would be required to compensate for this problem. Later, a small learning rate will need more iterations to converge weights with small magnitudes, since their gradients are also small. If some inputs have much larger variance than others, they can dominate the training. So to avoid dominance of some high variance inputs over others, inputs are normalized by initializing weights as $w(n) = \frac{1}{Var(x(n)) + \epsilon}$, where$\epsilon$is a small positive constant of order $10^{-3}$ used to avoid division by zero.

### *2.6.5 Theoretical Properties of Nearest Neighbor and k-Nearest Neighbor Classifier*

The Bayes error is the minimum achievable error rate by any classifier. Incase if the classes overlap then the error rate will be nonzero. For example, suppose that the training input pattern, with the correct class label of that pattern, follows a Gaussian distribution with mean $\mu_i$ and fixed variance. The two Gaussians overlap so no classifier can predict the class label correctly for all training patterns, and the Bayes error rate is nonzero.

The Bayes error rate is the average over the space of all examples of the minimum error probability for each example. The optimal prediction for any test pattern$\boldsymbol{x}$is the label that has highest probability given $\boldsymbol{x}$. The error probability for this example is then one minus the probability of this label. Formally, the Bayes probability of error rate is

$$\mathrm{P}_{\mathrm{e-Bayes}} = \int_{\boldsymbol{x}} \mathrm{p}(\mathbf{x})[1 - \max_{\mathrm{i}} \mathrm{p}(\mathrm{i}|\mathbf{x})] \tag{1.1}$$

where $p(i|\boldsymbol{x})$ is the probability that $\boldsymbol{x}$ has label $i$ and $1 - p(i|\boldsymbol{x})$ is the probability that $\boldsymbol{x}$has a different label**.** The maximum is taken over the $N_c$ possible labels $i = 1$ to $i = N_c$ [30].

**Theorem:** When the number of training examples tends to infinity, the probability of error rate of NNC is at worst twice the Bayes error rate as proven in [1].

**Proof**: Let $\boldsymbol{x}$ be a test pattern and $\boldsymbol{m}_{ik}$ be its closest neighbor. If the number of training examples $N_v$ is large, then the probability distribution for any test pattern and its nearest neighbor will be essentially the same. In this case, for the $p^{th}$ test pattern, $\boldsymbol{x}$, the expected probability of error rate of NNC is

$$P_{e-NNC} = \sum_{\mathrm{i}=1}^{\mathrm{N_c}} \mathrm{p}(\mathrm{i}|\mathbf{x})[1 - \mathrm{p}(\mathrm{i}|\mathbf{x})] \tag{1.2}$$

To prove the theorem, we need to show that

$$\sum_{i=1}^{N_c} p(i|\mathbf{x})[1 - p(i|\mathbf{x})] \leq 2[1 - \max_i p(i|\mathbf{x})] \tag{1.3}$$

$$\text{i.e} \qquad P_{e-NNC} \leq 2 \cdot P_{e-Bayes}$$

Let $\max_i p(i|\boldsymbol{x}) = r$ and let the maximum be attained with $i = j$. Then the left hand side is

$$\sum_{i=1}^{N_c} p(i|\mathbf{x})[1 - p(i|\mathbf{x})] = r(1-r) + \sum_{i \neq j} p(i|\mathbf{x})[1 - p(i|\mathbf{x})] \tag{1.4}$$

and the right hand side is $2(1-r)$. The summation above is maximized when all the values $p(i|\boldsymbol{x})$ are equal for $i \neq j$. The value of left hand side is then

$$A = r(1-r) + (N_c - 1)\frac{1-r}{N_c - 1}\frac{(N_c - 1) - (1-r)}{N_c - 1} \tag{1.5}$$

$$\therefore \; A = r(1-r) + (1-r)\frac{N_c + r - 2}{N_c - 1}$$

Now $r \leq 1$ and $N_c - 2 + r < N_c - 1$ so $A < 2(1-r)$.This proves that, with large enough training set, no classifier can do better than half the probability of error rate of a 1-nearest neighbor classifier[30]

## 2.7 Problems with Nearest Neighbor and k-Nearest Neighbor Classifier

Though nearest neighbor methods are very easy to implement, they have many drawbacks.

1 Computationally expensive –Nearest neighbor classifiers compute distance of the input vector to all the input training vectors. This distance measurement is computationally expensive and requires that all the center vectors to be stored in memory. This increases the computational complexity and memory requirements. Due to these computational complexities they cannot be used for real time applications.

2 Curse of dimensionality - The accuracy of the nearest neighbor classifiers tends to decrease as the number of features or inputs increases [46]. The reason is that in a high-dimensional space all points tend to be far away from each other, so nearest neighbors are not meaningfully similar. Practically, if vectors (patterns) are represented using many features, then every pair of examples will likely disagree on many features, so it will be rather arbitrary which vectors are closest to each other [9].

3 Contaminated input features – noise and less discriminative input features can cause problems such as convergence difficulties, poor classification accuracy and contamination of the distance measure which leads to false classification.

4 Rigid Voronoi cells - Clustering algorithms often get stuck in local minima and the result is largely dependent on the choice of initial cluster centers [3] [4]. Generated clusters, $\boldsymbol{m}_{ik}$ are not changed after initialization, and are not chosen to minimize, $P_{e-NNC}$,the probability of error of the nearest neighbor classifierso they are not optimal. Clustering methods other than the Learning Vector Quantization (LVQ) method [2] do not adapt the center vectors in a way that minimizes the probability of error. [10].

# Chapter 3

## REVIEW OF OAWNNC

This chapter introduces a method to optimize the distance measure used in the Nearest Neighbor Classifier which gives a solution to the problem 3 (problem of contaminated input features) mentioned in section 2.6. The method is termed as Optimal Attribute Weighting in a Nearest Neighbor Classifier (OAWNNC) algorithm and it compares all the features and differentiates between noisy and unimportant features with important features.

## 3.1 Motivations behind the structure of OAWNNC

OAWNNC uses weighted Euclidean distance measure instead of regular Euclidean distance measure. The distance measure is initialized with weights as $w(n) = \frac{1}{Var(x(n))+\in}$ as mentioned in section 2.4.4 (weight initialization). This diminishes the dominance of inputs with high variance. However, these weights are not optimal since they barely contribute in improving the performance of the classifier. Traditional nearest neighbor classifier, has a probability of error, $P_{e-NNC}$, as a measure of how well the classifier performs. To calculate the optimal weights, there needs to be a way to minimize $P_{e-NNC}$ with respect to the weights, $\boldsymbol{w}$**.** Since, $P_{e-NNC}$ is a scalar value and its gradient with respect to weights is zero i.e. $\frac{\partial P_{e-NNC}}{\partial w(n)} = 0$, there is no direct way to minimize $P_e$, with respect to the weights to find optimal weights.

To solve this problem, OAWNNC maps traditional nearest neighbor classifier to a neural net. To calculate the optimal weights that improve the classification performance of the neural network optimization of the objective function is done. A mapping function is derived that provides one to one mapping between the NNC discriminant $d_i$ and the neural network discriminant $y(i)$, such that improving the classification performance of the neural network, improves the probability of error, $P_{e-NNC}$, of the Nearest Neighbor Classifier. A modified softmax discriminant function $y(i)$ is being used in the Optimal Attribute Weighting in a Nearest Neighbor Classifier (OAWNNC). This softmax function is defined as

$$\mathrm{y(i)} = \frac{(\mathrm{d_i})^{-1}}{\sum_{\mathrm{j=1}}^{\mathrm{N_c}}(\mathrm{d_j})^{-1}}$$

The distances of test vector from the closest center vectors of each class is inversed and divided by the sum of all the inversed distances which gives a score for each class represented by the softmax discriminant $y(i)$. It provides a one to one mapping from $d_i$ when the inverse of distances of the test pattern to the closest center vector of each class adds up to one, i.e

$\sum_{j=1}^{N_c}(d_i)^{-1} = 1$. Since this function is continuous at all points, gradients for optimization can be easily computed. The class of the test pattern is estimated from the output score vector, $\boldsymbol{y}$, as

$$i_c' = \underset{i}{\operatorname{argmin}}\, y(i)$$

Softmax function outputs a score in the range of 0 to 1. With the score of the correct class being close to 1 and that of the incorrect class close to 0. It makes it easier to comprehend the performance of the classifier if its outputs as the scores are interpretable as posterior probabilities of categorical target output. For this reason, OAWNNC chooses the target output of the $p^{th}$, $x^t{}_p$, of the correct class to be 1 and those of incorrect class to be 0 is chosen. This is called one-hot encoding technique.

$$t_p(i) = \delta(i - i_c(p))$$

where $i_c(p)$is the correct class of the $p^{th}$, pattern.

OAWNNC converts the classification problem into regression by using the mean square error function (MSE). Squared error loss is one of the most widely used loss function in statistics. In statistical modeling the MSE, representing the difference between the actual target output and the output values predicted by the neural network, is used to determine the extent to which the network fits the data and whether the removal of some explanatory variables, simplifying the model, is possible without significantly harming the model's predictive ability [36]. The objective function used in training $y(i)$ is

$$E = \frac{1}{N_v}\sum_{p=1}^{N_v}\sum_{j=1}^{N_c}\left(t_p(j) - y_p(j)\right)^2 \tag{3.1}$$

## 3.2 Modified Weighted Euclidean Distance Measure

During the training of the classifier optimal weights, $\boldsymbol{w}$, are calculated using regular weighted Euclidean distance measure as mentioned in section 2.4.4. While doing so few weight elements $w(n)$ become negative which causes the distance measure value, $d$, to become

negative and thus led to decreased performance and increased misclassification error of the classifier. To solve this issue, two different modified weighted Euclidean distance measure techniques are proposed:

1. Using absolute value of weights, $| w(n) |$, for distance calculation.

$$\mathrm{d}(\mathbf{x}_\mathrm{p}, \mathbf{m}_\mathrm{ik}) = \sum_{\mathrm{n}=1}^{\mathrm{N}} \left( | \mathrm{w(n)} | \cdot \left( \mathrm{x_p(n) - m_{ik}(n)} \right) \right)^2 + \in \tag{3.2}$$

2. Using squared values of weights, $\mathrm{w(n)}^2$, for distance calculation.

$$\mathrm{d}(\mathbf{x}_\mathbf{p}, \mathbf{m}_\mathbf{ik}) = \sum_{\mathrm{n}=1}^{\mathrm{N}} \left( \mathrm{w(n)}^2 \cdot \left( \mathrm{x_p(n) - m_{ik}(n)} \right) \right)^2 + \in \tag{3.3}$$

where $\in$ is a small positive constant of the order $10^{-3}$ that prevents the distances from being zero so that division by zero can be avoided while mapping the NNC to the neural network. Throughout this thesis squared weights, $w(n)^2$, are used, since during weight optimization training with different datasets, it was found that squared weights, $w(n)^2$, perform better as compared to absolute weights, $| w(n) |$.

## 3.3 Training OAWNNC for distance measure improvement

In order to solve the problems of the curse of dimensionality, and misclassifications caused by noise and less discriminative features, it is necessary to optimize the distance measure algorithm. To make sure that the distance measure emphasizes more on highly discriminative features than the less discriminative features, we optimize weights corresponding to each input feature.

The training of a classifier is done by changing of the weights in order to make the computed output as close as possible to the desired output, thus reducing the mean square error (MSE). It mainly involves the following two independent steps. First a search direction has to be determined. i.e., in what direction do we want to search in weight space for a new current

point. Once the search direction has been found we have to decide how far to go in the specified search direction, i.e., a step size has to be determined. Most of the optimization methods used to minimize error functions are based on the same strategy. The minimization is a local iterative process in which an approximation to the error function in a neighborhood of the current point in weight space is minimized.

### *3.3.1 First Order Training Algorithms for Weight Optimization*

In this section, steepest descent which is the first order optimization algorithm is used. The negative gradient of the MSE with respect to input weights are calculated as follows,

$$\mathbf{g} = -\frac{\partial \mathrm{E}}{\partial \mathbf{w}} \tag{3.4}$$

Each element of the negative gradient vector $\boldsymbol{g}$ is calculated from the above equation (3.4) as

$$\mathrm{g(n)} = -\frac{\partial \mathrm{E}}{\partial \mathrm{w(n)}} = \frac{2}{\mathrm{N_v}} \cdot \sum_{\mathrm{p=1}}^{\mathrm{N_v}} \sum_{\mathrm{j=1}}^{\mathrm{Nc}} \left(\mathrm{t_p(j)} - \mathrm{y_p(j)}\right) \cdot \frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{w(n)}} \tag{3.5}$$

where, taking the partial derivative of $E$ in equation (3.1) yields

$$\frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{w(n)}} = \frac{-\sum_{\mathrm{u=1}}^{\mathrm{N_c}} \mathrm{d_u^{-1}} \cdot \left(\mathrm{d_j}\right)^{-2} \cdot 2 \cdot w(n) \cdot \left[\mathrm{x_p(n)} - \mathrm{m_{jk}(n)}\right]^2 + \left(\mathrm{d_j}\right)^{-1} \cdot \sum_{\mathrm{u=1}}^{\mathrm{Nc}} (\mathrm{d_u})^{-2} \cdot 2 \cdot \mathrm{w(n)} \cdot \left[\mathrm{x_p(n)} - \mathrm{m_{uk}(n)}\right]^2}{\left(\sum_{\mathrm{u=1}}^{\mathrm{N_c}} (\mathrm{d_u})^{-1}\right)^2} \tag{3.6}$$

Input weight changes are calculated using the negative gradients from equations (3.4), (3.5) and (3.6) and learning factor $z$,where $z$, is a heuristically chosen scalar value. The process of updating the weights is as follow,

$$\mathbf{w} \leftarrow \mathbf{w} + \mathrm{z} \cdot \mathbf{g} \tag{3.7}$$

*Optimal Learning Factor*

The learning factor,$z$, decides the rate of convergence rate of the training. Usually a very small positive value for $z$ will work, but convergence is likely to be slow. If $z$ is too large the error,$E$, can increase [36]. In order to avoid this uncertainty, a lot of heuristic scaling approaches have been introduced to modify the learning factors between iterations and thus speed up the rate of convergence. In this thesis we are using Taylor's series for the error $E$, a non-heuristic Optimal Learning Factor (OLF) can be calculated as**,**

$$\mathbf{z} = \frac{-\frac{\partial \mathbf{E}}{\partial \mathbf{z}}}{\frac{\partial^2 \mathbf{E}}{\partial \mathbf{z}^2}} \tag{3.8}$$

where the numerator and denominator derivatives are evaluated at $z = 0$. Assume that the learning factor,$z$, is used to update only the input weights $\boldsymbol{w}$, as given in equation (3.7).

Using the negative gradient $\boldsymbol{g}$, the optimal learning factor is derived in the following steps,

$$-\frac{\partial \mathrm{E}}{\partial \mathrm{z}} = \frac{2}{\mathrm{N_v}} \cdot \sum_{\mathrm{p=1}}^{\mathrm{N}} \sum_{\mathrm{j=1}}^{\mathrm{Nc}} \left(\mathrm{t_p(j)} - \mathrm{y_p(j)}\right) \cdot \frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{z}} \tag{3.9}$$

where,

$$\frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{w(n)}} = \frac{\mathrm{A} - \mathrm{B}}{\left(\sum_{\mathrm{u=1}}^{\mathrm{N_c}} (\mathrm{d_u})^{-1}\right)^2} \tag{3.10}$$

where,

$$\mathrm{A} = -\sum_{\mathrm{u=1}}^{\mathrm{N_c}} \mathrm{d_u^{-1}} \cdot \left(\mathrm{d_j}\right)^{-2} \cdot \sum_{\mathrm{p=1}}^{\mathrm{N_v}} (2 \cdot \mathrm{g(n)} \cdot \mathrm{w(n)} \cdot \left[\mathrm{x_p(n)} - \mathrm{m_{jk}(n)}\right]^2)$$

and

$$\mathrm{B} = -\left(\mathrm{d_j}\right)^{-1} \cdot \sum_{\mathrm{u=1}}^{\mathrm{Nc}} \mathrm{d_u^{-2}} \cdot \sum_{\mathrm{p=1}}^{\mathrm{N_v}} (2 \cdot \mathrm{g(n)} \cdot \mathrm{w(n)} \cdot \left[\mathrm{x_p(n)} - \mathrm{m_{uk}(n)}\right]^2)$$

Also, Gauss-Newton approximation for second partial is given by,

$$\frac{\partial^2 \mathrm{E}}{\partial \mathrm{z}^2} = \frac{2}{\mathrm{N_v}} \cdot \sum_{\mathrm{p=1}}^{\mathrm{N}} \sum_{\mathrm{j=1}}^{\mathrm{Nc}} \left[ \frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{z}} \right]^2 \tag{3.11}$$

Thus the optimal learning factor is calculated using equations (3.8), (3.9), (3.10) and (3.11).

After finding the optimal learning factor (OLF) the input weights are updated as given in equation (3.7)

*First Order Training Algorithm Summary for Weight Optimization*

1) $K$ clusters are made out of $N_v$ training patterns using K-Means++ clustering algorithm [45], where $K = \sum_{\mathrm{i=1}}^{\mathrm{Nc}} k_i$. $k_i$ is the number of clusters of the $i^{th}$ class

   After clustering we get $K$ center vectors $\boldsymbol{m}_{ik}$, where $\boldsymbol{m}_{ik}$is the $k^{th}$ center vector of the $i^{th}$ class

2) Initialize $w(n)$
3) For iteration, $i_t$ = 1 to $N_{it}$, where $N_{it}$ is the total number of iterations,
4) During first data pass calculate $d_i, \boldsymbol{y}, E, \boldsymbol{g}$
5) During second data pass calculate– $\frac{\partial E}{\partial z}, \frac{\partial^2 E}{\partial z^2}, z$
6) Update $\boldsymbol{w} \leftarrow \boldsymbol{w} + z \cdot \boldsymbol{g}$
7) End iterations

* Refer to Appendix A for the pseudo-code

### *3.3.2 Second Order Training algorithm for Weight Optimization*

The second order training of a multi layer perceptron involves quadratic modeling of the error function. The advantage of using the second order training methods is that it has got fast convergence. However, they can lead to problems like memory limitation, since the hessian and gradient matrices should be computed and stored and they also are computationally very expensive.

*Newton's Method*

Newton's method is the basis of number of popular second order optimization algorithms. Newton's algorithm is iterative, where in each iteration, [28]

1) Calculate Newton weight change vector $\boldsymbol{e}$.
2) Update weights with this weight change vector $\boldsymbol{e}$.

The weight change vector $\boldsymbol{e}$ is calculated by solving the linear equations using OLS [27]

$$\mathbf{H} \cdot \mathbf{e} = \mathbf{g} \tag{3.12}$$

where, the negative gradient of MSE with respect to weights is represented by $\boldsymbol{g}$, and it is calculated using equation (3.4), (3.5) and (3.6) and $\boldsymbol{H}$ is Hessian of the objective function calculated with respect to all the weights in the network and has elements defined as,

$$\mathrm{h(i,j)} = \frac{\partial^2 \mathrm{E}}{\partial \mathrm{w(i)}\, \partial \mathrm{w(j)}} \tag{3.13}$$

Equation (3.12) is solved for $\boldsymbol{e}$ using Orthogonal Least Squares (OLS) [27] and $\boldsymbol{w}$ is updated as

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{e} \tag{3.14}$$

We keep on updating the weight $\boldsymbol{w}$ till the change in the training error from the previous iteration is less than $10^{-6}$.

*Second Order Training Algorithm Summary for Weight Optimization*

1) Clustering of the $N_v$ training patterns into $K$ clusters is done using K-Means++ clustering algorithm [38].

   After clustering we get $K$ center vectors represented as $\boldsymbol{m}_{ik}$, where $\boldsymbol{m}_{ik}$is the $k^{th}$ center vector of $i^{th}$ class

2) Initialization of $w(n)$
3) For iteration, $i_t$ = 1 to $N_{it}$, where $N_{it}$ is the total number of iterations,
4) Calculate $d_i, \boldsymbol{y}, E, \boldsymbol{g}, \boldsymbol{H}$
5) Calculate $\boldsymbol{e}$
6) Update weight using the formula $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{e}$
7) End iterations

* Refer to Appendix A for the pseudo code

## 3.4 OAWNNC Results

Table – 1 compares randomized 10-fold testing results from traditional nearest neighbor classifier using center vectors $\boldsymbol{m}_{ik}$ as input training vectors and OAWNNC with distance measure optimization (DMO).

| *Data Set* | *Number of inputs* | *Number of classes* | *NNC accuracy %* | *DMO accuracy %* |
|---|---|---|---|---|
| *F17C.dat* | *17* | *39* | *25.5005* | *68.7039* |
| *SKIN.dat* | *2* | *2* | *93.8531* | *94.4196* |
| *GONGTST.tst* | *16* | *10* | *66.8333* | *77.8000* |
| *COMF18.TRA* | *18* | *4* | *54.2776* | *73.6481* |
| *PHONEME.dat* | *5* | *2* | *61.4701* | *75.3252* |
| *OBJECT RECOG* | *576* | *2* | *32.24* | *47.64* |

Table 1- Classification Performance of NNC v/s OAWNNC with Distance Measure Optimization (DMO).

This is a plot of mean square error (MSE) versus iteration number, $N_{it}$, for the nearest neighbor classifiers mapped neural network with first order and second training for weight optimization on COMF18.tra dataset. From the plot it concludes, that MSE converges much faster using second order training as compared to the first order training.

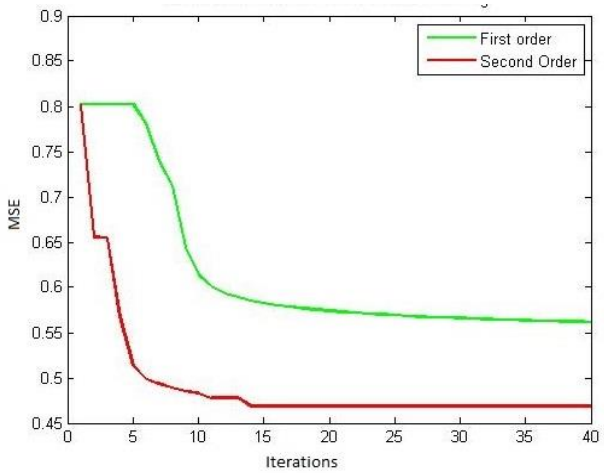


Figure 1–Weight Optimization Training Iteration Difference for COMF18.tra dataset.

From the results shown in Table – 1, it can be concluded that OAWNNC with distance measure optimization outperforms the traditional NNC on all the datasets. From Figure – 1,

second order training error converges much faster than first order training. The trained weights get assigned such that the magnitude of the weights corresponding to the noisy or unimportant features is less compared to high impacting features. Thus OAWNNC with distance measure optimization (DMO)solves problem 3 mentioned in
section 2. 6.

## 3.5 CENTER VECTOR OPTIMIZATION (CVO)

The center vectors $\boldsymbol{m}_{ik}$, can be optimized to reduce the probability of error and improve classification of error. This section introduces to a method of moving the center vectors $\boldsymbol{m}_{ik}$ in an optimized method using second order training to reduce the probability of error and improve the classification accuracy.

Cluster center vectors formed by using clustering algorithms are rigid so they do not change after initialization. Thus our OAWNNC algorithm is dependent on these initial clusters. Since these clusters do not contribute to minimizing the probability of error, they are not optimal. In addition to this, clustering is sensitive to the choice of initial clusters, clustering parameters etc. Therefore, few of the center vectors generated may not be unique. Hence we can say the classifier is highly dependent of the initialization of the center vectors and the rigid nature of the clusters. This is why the center vectors needs to be moved and adjusted so that it reduces the probability of errors. The Learning Vector Quantization method is a technique somewhat does the same thing of optimizing the codebook of input vector so that it decreases the input-output mapping [39,40].

## 3.6 Structure of Center Vector Optimization (CVO)

Here, Center Vector Optimization (CVO) minimizes the MSE error from equation (3.1) with respect to the $i^{th}$class,$k^{th}$center vector $\boldsymbol{m}_{ik}$**.** In the first step it calculates the negative gradient with respect to $\boldsymbol{m}_{ik}$ and then move the center vector in the direction of the negative gradient

using a second order algorithm. The negative gradient, $\boldsymbol{g}_{ik}$, of the MSE with respect to $\boldsymbol{m}_{ik}$ is calculated as follows,

$$\boldsymbol{g}_{ik} = -\frac{\partial \mathrm{E}}{\mathbf{m}_{\mathrm{ik}}} \tag{3.15}$$

$$\mathbf{g}_{\mathrm{ik}} = -\frac{\partial \mathrm{E}}{\partial \mathrm{m}_{\mathrm{ik}}(\mathrm{n})} = \frac{2}{\mathrm{N_v}} \cdot \sum_{\mathrm{p=1}}^{\mathrm{N_v}} \sum_{\mathrm{j=1}}^{\mathrm{Nc}} \left(\mathrm{t_p(j)} - \mathrm{y_p(j)}\right) \cdot \frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{m_{ik}(n)}} \tag{3.16}$$

where, $y_p(j)$ is calculated as,

$$\mathrm{y_p(j)} = \frac{\left(\mathrm{d_j}\right)^{-1}}{\sum_{\mathrm{u=1}}^{\mathrm{N_c}} (\mathrm{d_u})^{-1}} \tag{3.17}$$

And $\frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{m_{ik}(n)}}$ is computed as,

$$\frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{m_{ik}(n)}} = 0$$

if $\boldsymbol{m}_{ik}$ is not participating in computing $y_p(j)$

or else,

$$\frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{m_{ik}(n)}} = 2 \cdot \mathrm{d_j^{-2}} \cdot \mathrm{w(n)^2} \cdot \left(\mathrm{x_p(n)} - \mathrm{m_{ik}(n)}\right) . \left(\sum_{\mathrm{u=1}}^{\mathrm{Nc}} \mathrm{d_u^{-1}} - \mathrm{d_j^{-1}}\right) / \sum_{\mathrm{u=1}}^{\mathrm{Nc}} \mathrm{d_u^{-1}}) \tag{3.18}$$

when i = j, and

$$\frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{m_{ik}(n)}} = -2 \cdot \mathrm{d_j^{-1}} \cdot \mathrm{d_i^{-2}} \cdot \mathrm{w(n)^2} \cdot \left(\mathrm{x_p(n)} - \mathrm{m_{ik}(n)}\right) / \sum_{\mathrm{u=1}}^{\mathrm{Nc}} \mathrm{d_u^{-1}}$$

when i ≠ j

Once the calculation of the negative gradient $\boldsymbol{g}_{ik,}$ of the MSE with respect to $\boldsymbol{m}_{ik}$ is done the Center Vectors are moved in the direction of the negative gradients in order to minimize the error.

The center vector is updated as follows,

$$\boldsymbol{m}_{ik} \leftarrow \boldsymbol{m}_{ik} + z_{ik} \cdot \boldsymbol{g}_{ik} \tag{3.19}$$

## 3.7 Multiple Optimal Learning Factor (MOLF)

The learning factor, $z$, decides the rate of convergence of classifier training. So, different learning factors can be used for different center vectors. This technique of using multiple learning factors $z$,proves to e efficient. Using a Taylor's series for the error $E$, and Newton's algorithm [36] a non-heuristic multiple optimal learning factor (MOLF) can be calculated as,

$$\boldsymbol{H} \cdot \boldsymbol{z} = \boldsymbol{g_z} \tag{3.20}$$

where,$\boldsymbol{g}_z$is the negative gradient of error $E$with respect to the learning factors$\boldsymbol{z}$, after replacing center vector$\boldsymbol{m}_{ik}$ as given in equation (3.19) and evaluated at $z_{ik} = 0$. Matrix $\boldsymbol{H}$, is the Hessian matrix of the objective function.Assume that the learning factor,$z_{ik}$, is used to update only the center vector,$\boldsymbol{m}_{ik}$, as given in equation (3.19).Using $\boldsymbol{g}_{ik}$from equation (3.15), negative gradient $\boldsymbol{g}_{zik}$ is calculated as follows,

$$\mathbf{g}_{\mathrm{zik}} = -\frac{\partial \mathrm{E}}{\partial \mathrm{z}_{ik}} = \frac{2}{\mathrm{N_v}} \cdot \sum_{\mathrm{p=1}}^{\mathrm{N_v}} \sum_{\mathrm{j=1}}^{\mathrm{Nc}} \left(\mathrm{t_p(j)} - \mathrm{y_p(j)}\right) \cdot \frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{z}_{ik}} \tag{3.21}$$

as mentioned in section 3.6, if center vector, $\boldsymbol{m}_{ik,}$ is not participating in computing $y_p(j)$ using equation (3.17) then,

$$\frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{z_{ik}}} = 0$$

else,

$$-\frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{z_{ik}}} = 2 \cdot \mathrm{d_j^{-2}} \cdot \left(\sum_{\mathrm{n=1}}^{\mathrm{N}} \mathrm{g_{ik}(n)} \cdot \mathrm{w(n)^2} \cdot \left(\mathrm{x_p(n)} - \mathrm{m_{ik}(n)}\right)\right) \cdot \left(\sum_{\mathrm{u=1}}^{\mathrm{Nc}} \mathrm{d_u^{-1}} - \mathrm{d_j^{-1}}\right) / \sum_{\mathrm{u=1}}^{\mathrm{Nc}} \mathrm{d_u^{-1}} \tag{3.22}$$

when i = j, and

$$-\frac{\partial \mathrm{y_p(j)}}{\partial \mathrm{z_{ik}}} = -2 \cdot \mathrm{d_j^{-1}} \cdot \mathrm{d_i^{-2}} \cdot \left(\sum_{\mathrm{n=1}}^{\mathrm{N}} \mathrm{g_{ik}(n)} \cdot \mathrm{w(n)^2} \cdot \left(\mathrm{x_p(n)} - \mathrm{m_{ik}(n)}\right)\right) / \sum_{\mathrm{u=1}}^{\mathrm{Nc}} \mathrm{d_u^{-1}}$$

when $\mathrm{i} \neq \mathrm{j}$ .The Hessian matrix, $\boldsymbol{H}$, of the objective function calculated with respect to all the center vectors is computed as,

$$h(i,j) = \frac{\partial^2 E}{\partial z_{ik} \partial z_{jk}} \tag{3.23}$$

Equation (3.20) is solved for $\boldsymbol{z}$, using OLS [41] and $\boldsymbol{m}_{ik}$ is updated according to equation (3.19).

*CVO Training Algorithm Summary*

1) Iterating, $i_t$ = 1 to $N_{it}$, where $N_{it}$ is the total number of iterations.
2) On first data pass calculate $d_i, \boldsymbol{y}, E, \boldsymbol{g}_{ik}$
3) On second data pass calculate $-\frac{\partial E}{\partial z_{ik}}, \frac{\partial^2 E}{\partial z_{ik}^2}, \boldsymbol{z}$
4) Update $\boldsymbol{m}_{ik} \leftarrow \boldsymbol{m}_{ik} + z_{ik} \cdot \boldsymbol{g}_{zik}$
5) End iterations

## 3.8 Results of using Center Vector Optimization

Table – 2 compares randomized 10-fold testing results from traditional nearest neighbor classifier, OAWNNC and OAWNNC with center vector optimization.

| *Data Set* | *Number of inputs* | *Number of classes* | *NNC accuracy %* | *OAWNNC with DMO accuracy %* | *OAWNNC with DMO and CVO accuracy %* |
|---|---|---|---|---|---|
| *F17C.dat* | *17* | *39* | *25.50* | *68.70* | *69.02* |
| *SKIN.dat* | *2* | *2* | *93.85* | *94.42* | *95.42* |
| *GONGTST.tst* | *16* | *10* | *66.83* | *77.80* | *80.47* |
| *COMF18.TRA* | *18* | *4* | *54.28* | *73.65* | *77.28* |
| *PHONEME.dat* | *5* | *2* | *61.47* | *75.33* | *76.32* |
| *OBJECT RECOG* | *576* | *2* | *32.24* | *47.64* | *50.22* |

Table 2 - Classification Performance of NNC v/s OAWNNC with DMO v/s OAWNNC with DMO and CVO

From the results shown in the Table – 2, it can be concluded that center vector optimization makes OAWNNC insensitive to initial cluster center vectors. It optimally moves the cluster center vectors that further improve the performance of OAWNNC as compared to traditional nearest neighbor classifier. Thus CVO solves problem 4 mentioned in section 2.5.

### 3.9 Problem of misplaced center vectors

As we see, using the OAWWNC and CVO method on the classifier reduces the classification error and increases the prediction Accuracy of the classifier. It also reduces the computational time and storage used during the classification process. But training data generally contains a lot of outliers which is recorded because of human or computing errors during the observation of the experiment. These outliers results into incorrect the initialization of the center vectors during clustering. The Center Vector Optimization rule causes the increase in accuracy of prediction of the Classifier by moving the Center Vectors. But if there is a concentration of training vector which are outliers then the Center Vector found during Clustering is situated at the region of the outliers. Center Vector optimization causes the outlying Center Vector move to give an optimized Classifier, but still the Center Vector will be present which can cause incorrect predictions. There are even Center Vectors which are very near to one another and removing one might increase the accuracy of prediction and decrease computational speed and storage. For all such Center Vectors, which reduces the Accuracy of the Classifier or plays no part in the prediction of testing data, the efficiency of the Classifier decreases? This problem of unwanted Center Vectors is handled in the next section of Pruning.

## Chapter 4

## PRUNING AND IMPROVED NEAREST NEIGHBOR CLASSIFIER

This section introduces a way of pruning center vectors $\boldsymbol{m}_{ik}$ in an optimized method to increase the efficiency of prediction of the Nearest Neighbor Classifier.

The distance between the Center Vectors and a new testing vector is measured to find the correct class. So, the efficiency of the classifier is dependent on the position of the center

vectors and the position of the center vectors is dependent on the training data. Data contains outliers. In statistics, an outlier is an observation point that is far away from other observations. An outlier may be because of variability in the measurement or it may signify experimental error [17]. Because of outliers the clustering algorithm initializes some clusters which are distant from the actual observation points. This data points are not important during classification and this excess unimportant center vectors decreases the efficiency of the classification by increasing the processing time. Some center vectors causes incorrect classification, removing which increases the accuracy of prediction of the Nearest Neighbor Classifier [42, 45]. Pruning is the methodical way of removing unimportant and bad center vectors which causes the increase of accuracy of the classifier [46].

## 4.1 STRUCTURE OF PRUNING

The method of pruning removes center vectors $\boldsymbol{m}_{ik}$ one at a time and calculates the accuracy of the classifier for each case when a center vector is removed. Then it compares all *k* accuracy and finds out the $k^{th}$ center vector $\boldsymbol{m}_{ik}$, removing which causes the accuracy of prediction A, to increase.

This process is continued until pruning of center vectors does not increase the accuracy of the classifier. The resultant center vectors are then used as input training vectors for the purpose of classification. This process of pruning results into an even smaller training dataset compared to the complete training vectors or even the center vectors found after Clustering and Center Vector Optimization. This eventually decreases the computational time for classification, helping in increasing the efficiency of the Nearest Neighbor classifier. Thus, it also allows the Classifier to be used in a real-time application.

The accuracy *A*, for a classifier is calculated as [47],

$$A = 1 - P_e \quad (4.1)$$

where, $\hat{y}_i$ is the predicted value of the $i^{th}$ sample and $y_i$ is the corresponding true value.

$f(\hat{y}_i = y_i)$ is an indicator function, it returns 1 when $\hat{y}_i$ is equal to $y_i$ and returns 0 when $\hat{y}_i$ is not equal to $y_i$.

Accuracy $A_k$ is calculated every time a center vector is removed, where $A_k$ is the accuracy of the Classifier when the $k$th cluster, $\boldsymbol{m}_{ik}$, is pruned.

The cluster to be pruned is done using the following logic,

$$k = \underset{k}{\operatorname{argmax}} A_k$$

## 4.2 PRUNING ALGORITHM SUMMARY

1) the $\boldsymbol{m}_{ik}$ center vector is removed
2) The accuracy $A_k$ is found
3) The pruned center vector which results to the maximum Accuracy is removed from the training dataset and the maximum Accuracy is saved
4) Pruning stops when the maximum Accuracy of the classifier starts decreasing due to removing input center vectors.

## 4.3 CHOOSING THE BEST STAGE FOR PRUNING

The pruning method removes the training centre vectors $\boldsymbol{m}_{ik}$ which decreases or does not affects the accuracy of the classifier, but the question remains when to perform the pruning operation. The pruning method can be applied to the method of preparing the training vectors in several stages. After clustering of the training vectors $x_p$ and distance measure optimization, it needs to be decided whether to perform pruning of centre vectors $\boldsymbol{m}_{ik}$ and then move the centre vectors optimally or to perform pruning after the centre vector positions are optimized. The several stages where pruning can be applied are:

i. Distance measure optimization, then pruning. (OAWNNC + Pruning)
ii. Distance measure optimization, then pruning and then moving centre vectors (OAWNNC + Pruning + CVO)

iii. Distance measure optimization followed by moving centre vectors and then pruning. (OAWNNC + CVO + Pruning)

iv. Distance measure optimization followed by pruning which followed by moving centre vectors and then again performing weight optimization (WO). (OAWNNC + Pruning + CVO + WO)

All this four different methods of finding the optimized training vectors is tested and the corresponding results recorded.

## 4.4 RESULT OF PRUNING AT DIFFERENT STAGES

Pruning is done at different stages of the improved nearest neighbor classification algorithm and the results of accuracy of the classifier is recorded. The results are compared and displayed below,

| *Data Set* | *Number of inputs* | *Number of classes* | *OAWNNC + Pruning + CVO accuracy %* | *OAWNNC + CVO + Pruning accuracy %* | *OAWNNC + Pruning + CVO + WO accuracy %* | *OAWNNC + Pruning accuracy %* |
|---|---|---|---|---|---|---|
| *F17C.dat* | *17* | *39* | *86.61* | *84.83* | *86.61* | *88.41* |
| *SKIN.dat* | *2* | *2* | *99.21* | *99.26* | *99.31* | *99.35* |
| *GONGTST.tst* | *16* | *10* | *82.53* | *72.96* | *83.03* | *85.20* |
| *COMF18.TRA* | *18* | *4* | *78.77* | *77.92* | *78.77* | *75.78* |
| *PHONEME.dat* | *5* | *2* | *85.02* | *84.69* | *85.02* | *81.72* |
| *OBJECT RECOG* | *576* | *2* | *76.56* | *82.58* | *84.94* | *85.79* |

Table 3- Accuracy of the classifier at different stages of pruning

## 4.5 RESULT OF PRUNING

In section 4.4, the results of different stages of pruning shows the best accuracy of the classifier model is achieved by optimizing distance measure and then performing pruning of the center vectors. The results of accuracy of the new improved classifier with weight optimization followed by pruning ($NNC_{WO-Pr}$) is compared with the regular nearest neighbor classifier which uses th entire training pattern for testing ($NNC_R$), the nearest neighbor classifier which uses training center vectors as training patterns ($NNC_{CV}$) and the improved nearest neighbor classifier from chapter 3, which uses the training center vectors for testing, improves the distance measure and also optimizes the position of the center vectors (OAWNNC+CVO)

| *Data Set* | *Number of inputs* | *Number of classes* | *$NNC_{CV}$ accuracy %* | *OAWNNC + CVO accuracy %* | *$NNC_R$ accuracy %* | *$NNC_{WO-Pr}$ accuracy %* |
|---|---|---|---|---|---|---|
| *F17C.dat* | *17* | *39* | *25.50* | *69.02* | *46.10* | *88.41* |
| *SKIN.dat* | *2* | *2* | *93.85* | *95.42* | *99.95* | *99.35* |
| *GONGTST.tst* | *16* | *10* | *66.83* | *80.47* | *87.20* | *85.20* |
| *COMF18.TRA* | *18* | *4* | *54.28* | *77.28* | *73.64* | *78.77* |
| *PHONEME.dat* | *5* | *2* | *61.47* | *76.32* | *88.46* | *81.72* |
| *OBJECT RECOG* | *576* | *2* | *31.44* | *50.22* | *95.09* | *85.79* |

Table 4- Comparison of accuracy of $NNC_{CV}$ vs. OAWNNC + CVO vs. $NNC_R$ vs. $NNC_{WO-Pr}$

From the above table it is evident that the proposed method of applying pruning to the nearest neighbor significantly increases the accuracy of the classifier from previous nearest neighbor classifiers which uses center vectors as input training vectors (OAWNNC+CVO and $NNC_{CV}$). But the regular nearest neighbor classifier ($NNC_R$) results to better accuracy than the proposed classifier, $NNC_{WO-Pr}$. This is because the $NNC_R$ takes the entire training dataset into

consideration for classification and hence, it has a lot more number of patterns to compare with to find the predicted class of the input testing vector. But considering all the patterns for classification makes the classifier computationally complex and the classifier takes more time to predict, hence, making it impractical for real time classification purposes. The proposed classifier in this thesis applies clustering which reduces the training patterns and pruning further decreases the size of the training pattern set. Hence, decreasing the time taken for classification and thus, increasing the efficiency of the classifier, making it applicable for real time purposes. The efficiency of the classifier is calculated by computing the time taken, in seconds, to predict the correct class of the testing data. The time taken for classification for testing data for OAWNNC + CVO, $NNC_R$ and $NNC_{WO\text{-}Pr}$ is compared in the following table.

| *Data Set* | *Number of inputs* | *Number of classes* | *OAWNNC + CVO Time taken in sec* | *$NNC_R$ Time taken in sec* | *$NNC_{WO\text{-}Pr}$ Time taken in sec* |
|---|---|---|---|---|---|
| *F17C.dat* | *17* | *39* | *0.07256* | *0.770755* | *0.0621* |
| *SKIN.dat* | *2* | *2* | *11.58744* | *1854.2547* | *4.98541* |
| *GONGTST.tst* | *16* | *10* | *0.21585* | *1.396677* | *0.193966* |
| *COMF18.TRA* | *18* | *4* | *0.9867558* | *4.62871* | *0.39433* |
| *PHONEME.dat* | *5* | *2* | *0.546507* | *1.199788* | *0.40522* |
| *OBJECT RECOG* | *576* | *2* | *0.832548* | *192.99* | *0.758221* |

Table 5- Efficiency of classifiers OAWNNC + CVO vs. $NNC_R$ vs. $NNC_{WO\text{-}Pr}$

From the above table we can conclude that nearest neighbor classifier with distance measure optimization and pruning performs the best in terms of efficiency.

From table we can conclude that the best accuracy can be achieved in most cases with a regular nearest neighbor classifier ($NNC_R$) but the efficiency of this classifier is too low to be used in a real time application. On the other hand the nearest neighbor classifier using center

vectors as training samples and implementing distance measure optimization and center vectors optimization (OAWNNC + CVO) has better efficiency than $NNC_R$ much lower accuracy than $NNC_R$. The nearest neighbor classifier with distance measure optimization and pruning ($NNC_{WO\text{-}Pr}$) has better accuracy than OAWNNC + CVO and nearly same or better accuracy compared to $NNC_R$. $NNC_{WO\text{-}Pr}$ has the best efficiency compared to OAWNNC + CVO and $NNC_R$. Hence, the classifier proposed in the thesis is better in performance compared to the regular nearest neighbor classifier and other improved versions of it and can be applicable for real time applications.

# Chapter 5

# IMPROVEMENT OF k-NEAREST NEIGHBOR CLASSIFIER

The k-Nearest neighbor classifier is a non-parametric method for classification [52]. The k-nearest neighbor classifier differs from the nearest neighbor classifier in selecting the number of nearest neighbor center vectors $\mathbf{m_{ik}}$ from the test data $\mathbf{x^t_p}$. In the nearest neighbor classifier only the nearest neighbor is considered for inferring the class of the input data. But in k-nearest neighbor classifier *k* nearest neighbor center vectors are taken in account for inferring the correct class for the input vector. The challenge faced in the k-nearest neighbor classifier is deciding on the optimum *k* value which gives the best accuracy of the classifier [53]. The optimal value of *k* is dependent on the data to be classification. This chapter reviews previous works to find the optimal *k* and then defines a method to find the optimal *k* by doing just one pass through the training data which is an improvement for the k-nearest neighbor classifier.

## 5.1 REVIEW WORK ON FINDING OPTIMAL *k*

The performance of the k-nearest neighbor classifier depends on choosing the optimal number of neighbors (k), which is different from one data sample to another. Performance of the classifier varies significantly when K is changed [83, 84].

Guo *et al.* converted the training data set into multiple smaller datasets called the “KNN Model”. The model groups each similar pattern together and consider the number of patterns in

a group as optimal *k* for that “KNN Model”. By doing so the need to choose the best k is eliminated. However, there is still a need to define other thresholds such as “error tolerant degree” and the minimum number of points allowed in each group. [85]

Hamamoto *et al.* used a bootstrap method for nearest neighbor classifier. But this classifiers performed well when the tested examples are in high dimensions. [86]

The “inverted indexes of neighbors classifier” (IINC) [28], [29] and [30] is one of the best attempts found in the literature to solve the problem. The aim of their work was not intentionally to solve the problem of the k parameters; rather it was designed to increase the accuracy of the classifier. The main idea of the IINC is to use all the neighbors in the training set, rewarding the nearest neighbors, and penalizing the furthest one. But this classifiers had few problems like, all the training data was needed to calculate all the inverted indices; non-uniform number of patterns in each class caused the accuracy of prediction to decrease; distances need to be sorted, which took long time and hence decreased the efficiency of the classifier.[90]

In the next section a method is proposed to find the optimal *k* by passing through the input train dataset just once.

## 5.2 EFFICIENT METHOD OF FINDING OPTIMAL *k*

A new method to find the optimal *k* parameter is proposed in which single pass through the data is done. The training data contains $N_v$-$N$-dimensional pattern. The training data is splitted into two parts, namely training data and validation data. The split percent is selected to be 70% by 30% which is considered to be fair split [58]. The larger chunk of split data is considered to be the training data, represented by ($\mathbf{x_p}$, $i_c(p)$)., where the $p^{th}$ dimensional training vector x is represented by $\mathbf{x_p}$ and $i_c(p)$ is it’s corresponding class label. The data is divided into two parts for the purpose of cross-validation.

*5.2.1 Cross Validation* – It is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. Cross-validation is a way to

predict the fit of a model to a hypothetical validation set when an explicit validation set is not available [59, 60].

Suppose we have a model where one or more parameters are unknown and we have a data set which can be fitted to the model. The process of fitting the data to the model causes optimization of the fitting parameters of the model and the end results gives a model which fits the data set as well as possible.[59,61] If we have an independent sample of validation data from the same population as the training data, it will generally turn out that the model does not fit the validation data as well as it fits the training data. Cross-validation is a way to predict the fit of a model to a hypothetical validation set when an explicit validation set is not available. [59, 61]

Two types of cross-validation can be distinguished, exhaustive and non-exhaustive cross-validation [59]. In the process of finding the optimized *k* for the k-nearest neighbor classifier, we use the k-fold cross-validation technique which is a non-exhaustive type of cross-validation. The K-fold cross-validation technique is more tractable than the exhaustive methods of cross-validation and it also uses all the observations for training and validation and each observation is used exactly once for validation [63]. Thus in the method of finding optimal *k* for the k nearest neighbor algorithm we use the K-fold validation technique.

5.2.2 Idea of K-fold validation - Given a single data set, randomly divide it into K disjoint subsets $D_k$ of equal size, for $1 \leq k \leq K$. Form K separate training/validation set pairs as

$$T_k = \{\bigcup_{j \neq k} D_j\}, V_k = D_k$$

for $1 \leq k \leq K$ [64]. The training vector during the $k^{th}$ fold of the validation is represented as $\mathbf{T_k}$ and the validation vector $k^{th}$ fold of the validation is denoted as $\mathbf{V_k}$. For every value of k, train the network on dataset $\mathbf{T_k}$ and test it on $\mathbf{V_k}$ and obtain $A_{cv}(K)$. $A_{cv}(K)$ is the cross-validation accuracy, *A*, of the classifier during the $K^{th}$ fold of the validation.

## 5.3 Choosing maximum *k*

The training data is split into two parts (namely training data, $\mathbf{T_k}$ and validation data, $\mathbf{V_k}$) if validation data is not available. It is done for the purpose of K-fold cross validation. Then a maximum value of *k* is chosen. Choosing the maximum value needs to be done in such a way that it should not be too large which might cause into reduced accuracy as because a large *k* means a large portion of the space is considered for calculation which would cause the prediction to be less dependent on the actual location of the space [52]. If the *k* is chosen too small then it will lead to noisy decision boundary [66]. The value of *k* depends on various parameters like the total number of patterns in the training data, $N_v$, the number of features in the data, $N$ ,the number the class labels, $N_c$ and also the number of patterns in each class label. The value of *k* is directly proportional to the value $N_v$, $N$ and $N_c$.

Past studies suggest selecting the value of maximum *k* to be the square root of the total number of patterns, $N_v$,[65,67,68,69]

i.e. $$\max(k) = \sqrt{N_v}$$

Intuitively the value of *k* should be less than twice the number of patterns of that class which has the least number of data.

i.e. $$i_{c\text{-}min} = \text{argmin}(N_v(i_c))$$

where $i_c$ is the $c^{th}$ class label and $i_{c\text{-}min}$ is the class which has the least number of patterns. Then the maximum value of *k* is given as,

$$\max(k) < 2N_v(i_{c\text{-}min}) < 2N_v(\text{argmin}(N_v(i_c)))$$

The reason for choosing the maximum value of *k* by this method is because, consider the distribution of the number of patterns of each class is uneven. Considering the worst case condition, if the k is selected to be more than twice the number of patterns of that class which has the least number of data, $2*N_v(\text{argmin}(N_v(i_c)))$,

$$\max(k) > 2N_v(i_{c\text{-}min})$$

then the classifier would not predict the class $i_{c\text{-}min}$, even though if it is the correct class, as the rest of the nearest neighbors vectors might belong to another incorrect class. Since, the number of rest of the nearest neighbors will be greater than the nearest neighbors of the class with the least number of patterns,

i.e. $2N_v(i_{c\text{-}min}) > N_v(i_{c\text{-}min})$,

Considering $\max(k) = 2N_v(i_{c\text{-}min}) + \beta$

where β is any positive integer value greater than 1.

$$\max(k) - N_v(i_{c\text{-}min}) = 2N_v(i_{c\text{-}min}) + \beta - N_v(i_{c\text{-}min}) = N_v(i_{c\text{-}min}) + \beta$$

Therefore, $\max(k) - N_v(i_{c\text{-}min}) > N_v(i_{c\text{-}min})$

As, $N_v(i_{c\text{-}min}) + \beta > N_v(i_{c\text{-}min})$

Now, if the all $N_v(i_{c\text{-}min}) + \beta$ patterns belongs from one different class then the classifier would predict the other class which is actually incorrect.

From previous works and above equations the maximum value of *k* is selected as whichever is maximum between the square root of the total number of patterns, $\sqrt{N_v}$, and one less than twice the number of patterns of that class which has the least number of data,

$$k_{max} = \max(\sqrt{N_v},\ 2N_v(i_{c\text{-}min}) - 1) \qquad (5.1)$$

## 5.4 Finding the optimal *k*

After splitting the training data into training and validation data, $\mathbf{T_k}$ and $\mathbf{V_k}$ respectively and deciding on the maximum value for *k*, the method of finding the best *k* for the k-nearest neighbor classifier is performed.

The distance between the first *k* training vectors $T_{Kk}$ and the *p*th validation vector $V_{Kp}$ is found and stored. The equation used to find the Euclidean distance is as follow,

$$d(T_{Kk},\ V_{Kp}) = \sum_{n=1}^{N} \left(T_{Kk}(n) - V_{Kp}(n)\right)^2$$

The maximum value of $k$, $k_{max}$, is found using the equation Eq 5.1. Then the distance of validation vector from the first $k_{max}$ training vectors are found and stored in the distance vector denoted as, $\mathbf{d_k}$. The remaining training vectors, from $k_{max}$+ 1 to $N_v$, is considered. Distance of each of this training vector from the validation vector is found one at a time and the symbol which denotes it is $d_{new}$.

The training vector, among the first $k_{max}$ training vectors, farthest from the test validation vector is found by finding the maximum distance value in the distance vector $\mathbf{d_k}$. The equations are as follow,

$$k_x = \underset{k}{\operatorname{argmax}} (\mathbf{d_k})$$

where $k_x$ is the position of the maximum distance in the $\mathbf{d_k}$ vector.

Now, the distance of the new training vector from the validation vector is compared to the maximum distance among $\mathbf{d_k}$ and updated as follow,

$$\text{IF } d_{new} < d_{kx}, \text{ THEN } d_{kx} \leftarrow d_{new}$$

where, $d_{kx}$, is the distance in the $k_x$ position of the $\mathbf{d_k}$ vector, hence $d_{kx}$ is the maximum value of distance in the $\mathbf{d_k}$ vector.

After of one pass through the data, the distance vector $\mathbf{d_k}$ will contain the $k_{max}$ nearest training vectors from the validation vector. Then the $\mathbf{d_k}$ vector is sorted in ascending order. Now the value of $k$ is considered from 1 to $k_{max}$ and the predicted class for each value of $k$ is calculated as follow,

$$k(i_c(k)) = k(i_c(k)) + 1$$

where, $i_c$ is the class label of the $c^{th}$ class and $i_c(k)$ is the predicted class of the $k^{th}$ vector. Thus, $k(i_c(k))$ is the count of the class labels of $c^{th}$ class calculated from the $k^{th}$ vector when the number of nearest neighbor considered for the calculation is $k$.

Then the predicted class is calculated as the class of the maximum occurring training pattern. Then the error vector, $\mathbf{e}$, is updated after comparing if the predicted class, $i_c(k)$, is same as the actual class label $i_c(p)$. The error vector is denoted as e(k) which is the error of the

classifier when *k* nearest neighbors are chosen for classification. The error vector e(k) is assigned zero if the predicted class is matches the actual class label, else the e(k) is assigned one.

$$\text{IF } (i_c(k) = i_c(p)), \text{ THEN } e(k) \leftarrow 0, \text{ ELSE } e(k) \leftarrow 1$$

Then the probability of error of the classifier is calculated using the error vector,

$$P_e(k) = \frac{P_e(k) + e(k)}{N_{vtest}}$$

where $N_{vtest}$ is the number of patterns in the validation vectors and $P_e(k)$ is the probability of error of the classifier when *k* nearest neighbors are considered for classification.

Then the accuracy, A, is calculated by using the equation,

$$A(k) = 1 - P_e(k)$$

where the accuracy of the classifier when *k* nearest neighbor is selected for classification is A(k).

The optimal *k*, $k_{optimal}$, is found by finding the *k* which results to the maximum accuracy, A(k).

$$k_{optimal} = \underset{k}{\operatorname{argmax}} (A(k))$$

5.4 Algorithm to find the best *k*

***Initialization***

The value of $k_{max}$ is initialized. The elements of error vector, **e**, initialized as 0.

Find $d_k$ where, $1 \leq k \leq k_{max}$

***Algorithm***

1) Start K-Fold cross-validation.
2) FOR $p = k_{max} + 1$ TO $N_v$
3) Calculate $k_x$, $d_{kx}$ and $d_{new}$
4) IF $d_{new} < d_{kx}$, THEN $d_{kx} \leftarrow d_{new}$
5) END FOR

6) SORT $d_k$

7) FIND $i_c(k)$

8) FOR k = 1 TO $k_{max}$

9) FIND $k(i_c(k))$

10) IF ($i_c(k) = i_c(p)$), THEN e(k) = 0

11) $P_e(k) = P_e(k) + e(k)$

12) $P_e(k) = P_e(k) / N_{vtest}$

13) $A(k) = 1 - P_e(k)$

14) END FOR k

15) Calculate $k_{optimal}$

## 5.5 Results of finding the optimal *k*

The above mentioned algorithm to find the optimal k is applied on the following six datasets and the results are displayed:

| **Dataset** | GONGTST.tra |
|---|---|
| $k_{optimal}$ | 4 |
| **Testing Accuracy** | 86.4% |

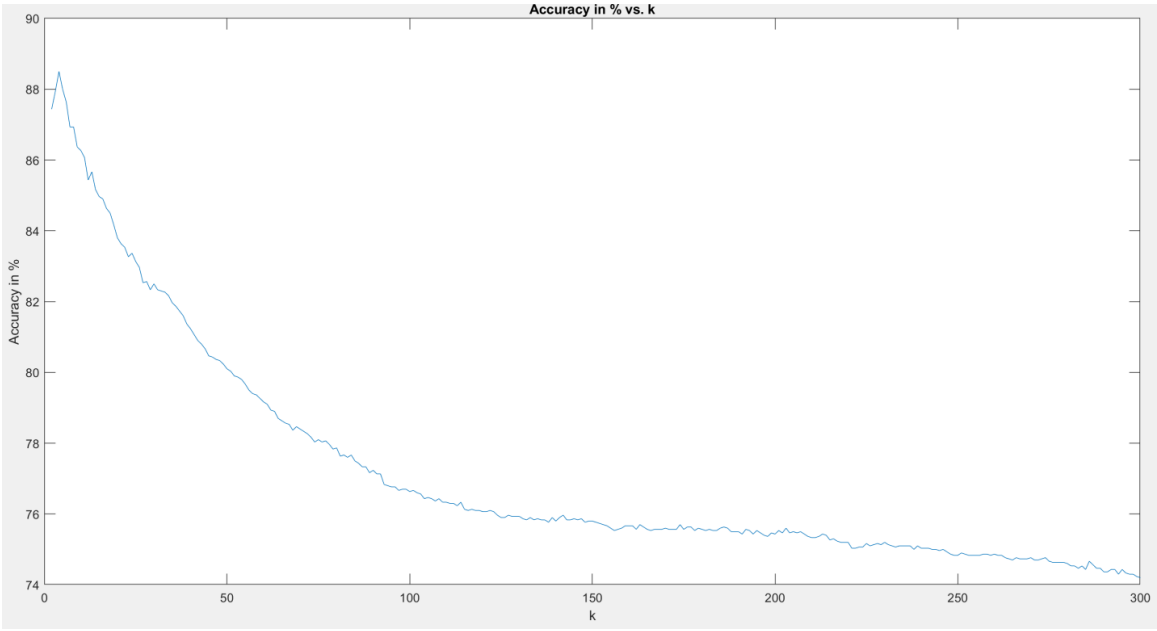

Fig-2 – Accuracy in % vs. k for GONGTST.tra data for range of k from 2 to 300

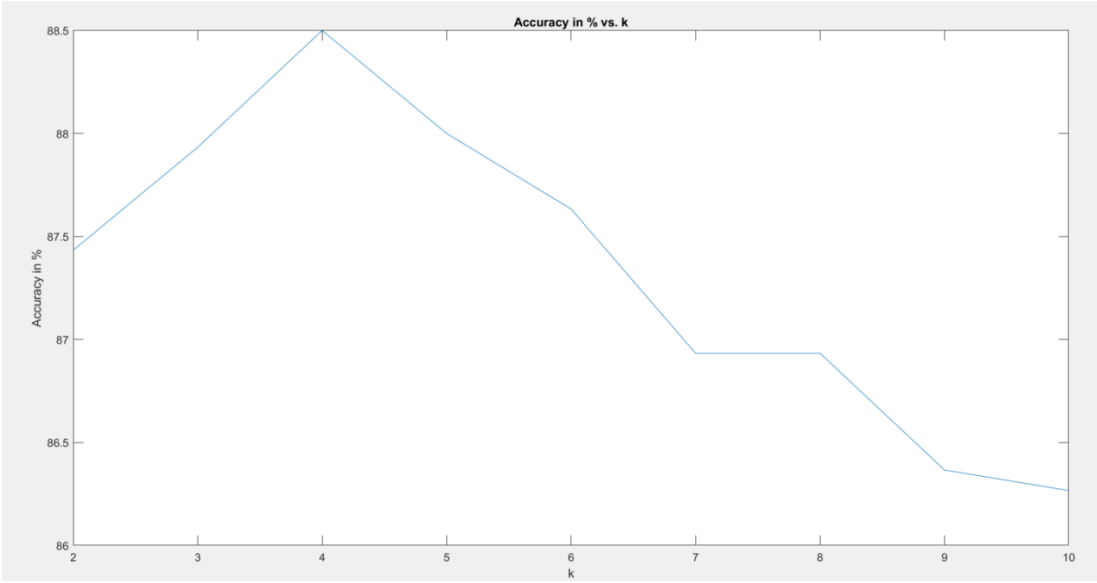

Fig-3 – Accuracy in % vs. k for GONGTST.tra data for range of k from 2 to 10

| | |
|---|---|
| **Dataset** | Comf18.tra |
| $k_{optimal}$ | 14 |
| **Testing Accuracy** | 78.73% |

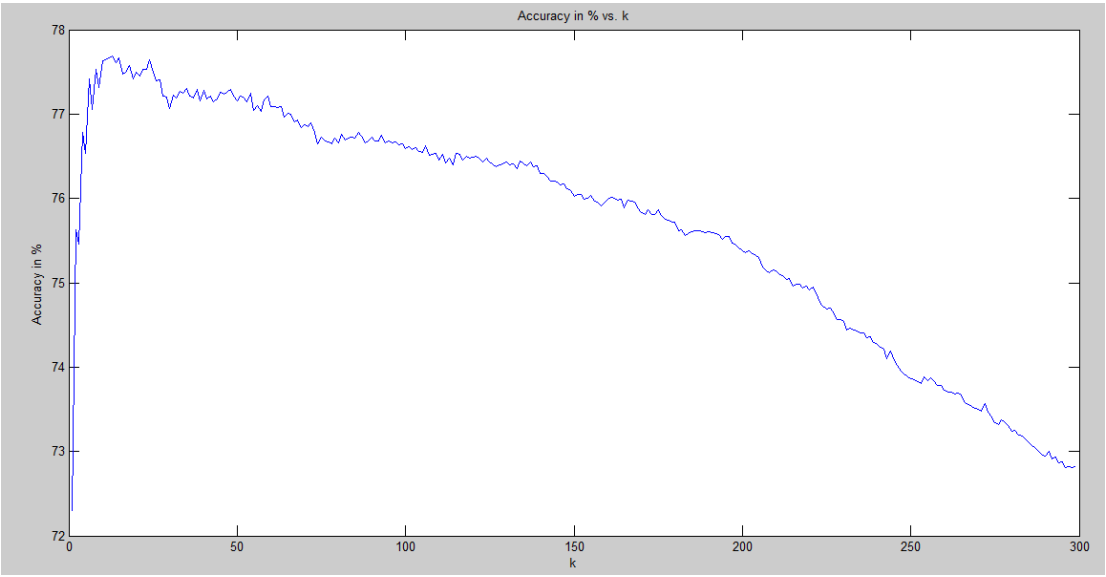


Fig-4 – Accuracy in % vs. k for Comf18.tra data for range of k from 2 to 300

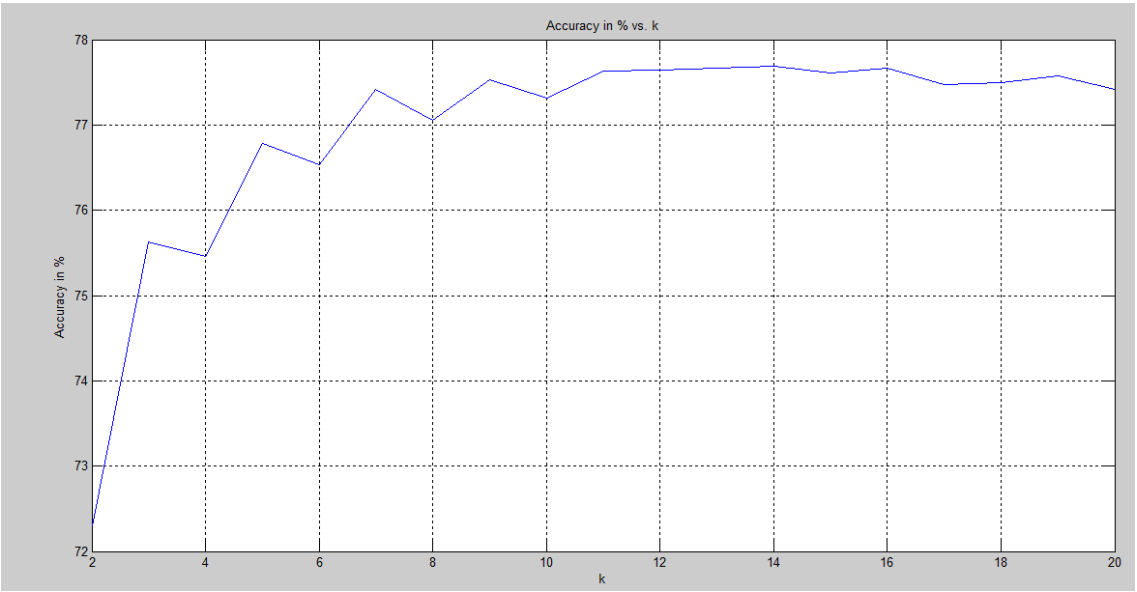


Figure 5– Accuracy in % vs. k for Comf18.tra data for range of k from 2 to 300

| | |
|---|---|
| **Dataset** | F17C.dat |
| $k_{optimal}$ | 5 |
| **Testing Accuracy** | 46.29% |

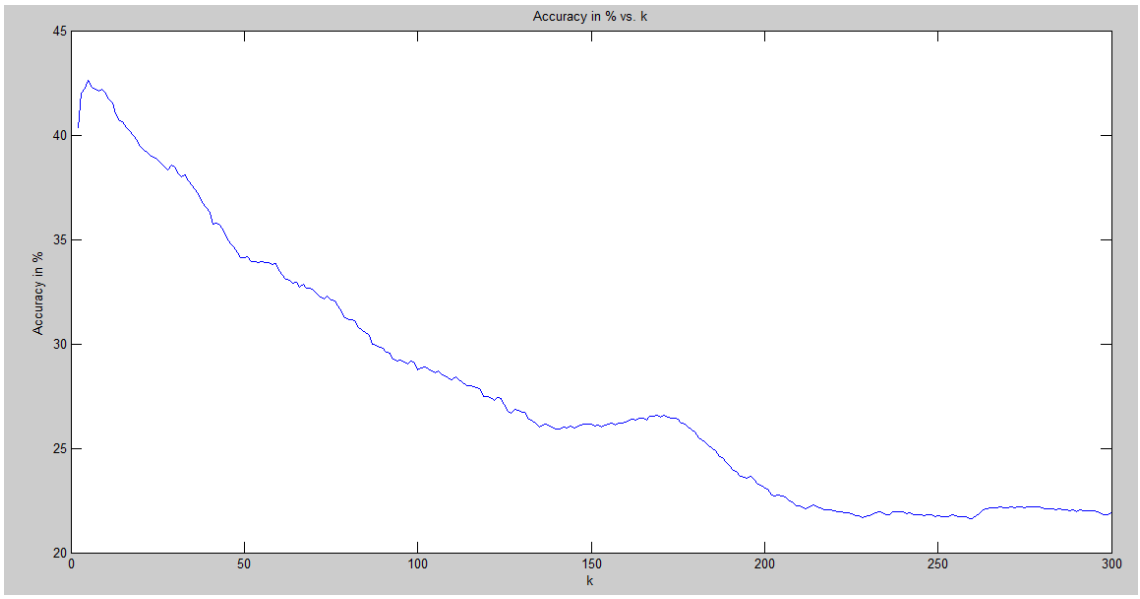


Fig-6 – Accuracy in % vs. k for F17C.dat data for range of k from 2 to 300

.

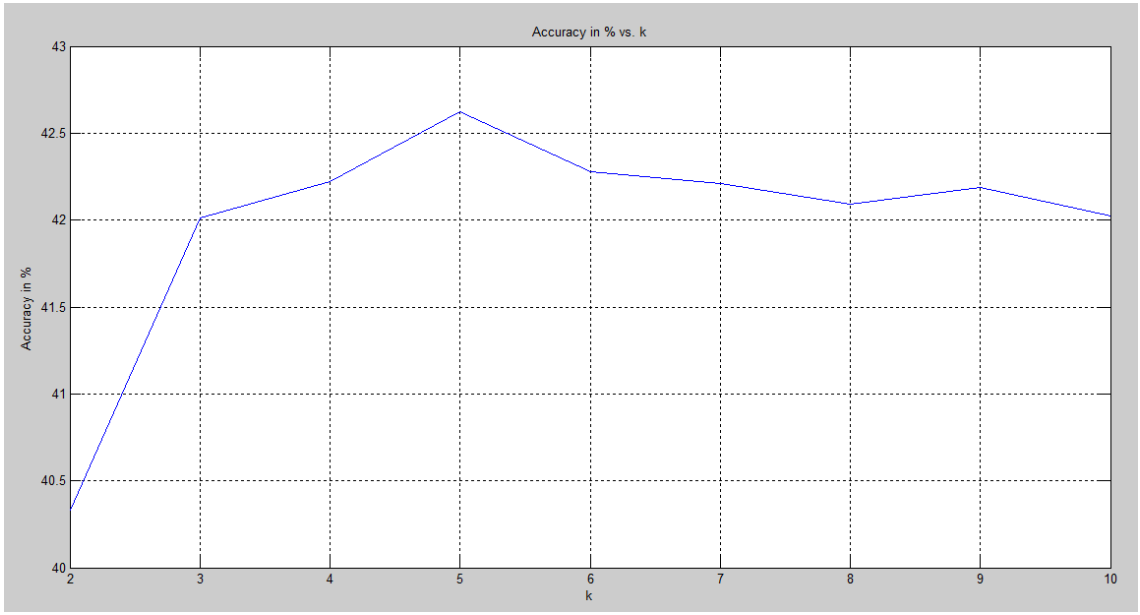


Fig-7 – Accuracy in % vs. k for F17C.dat data for range of k from 2 to 10

| | |
|---|---|
| **Dataset** | Object Recognition |
| $k_{optimal}$ | 2 |
| **Testing Accuracy** | 45.20% |

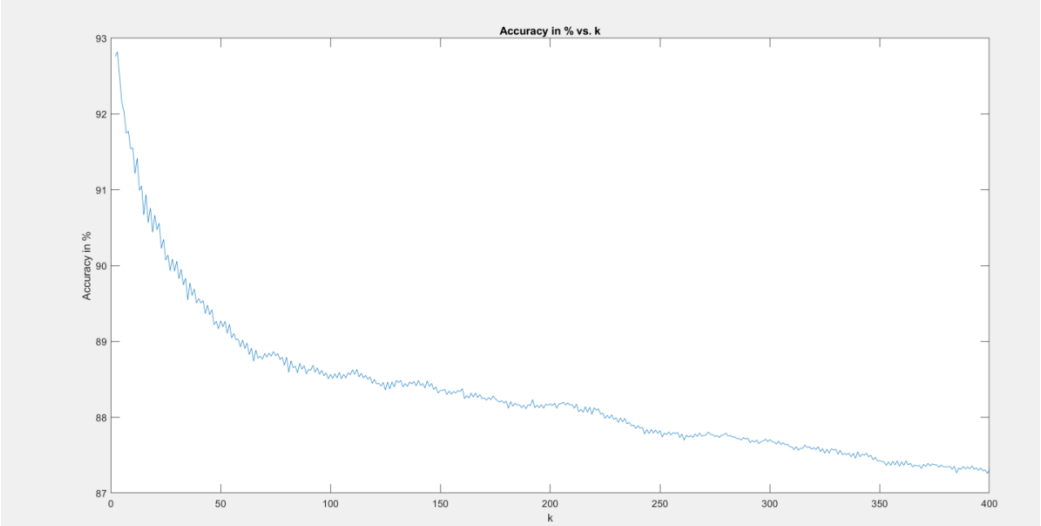


Fig-8–Accuracy in % vs. k for Object Recognition data for range of k from 2 to 400

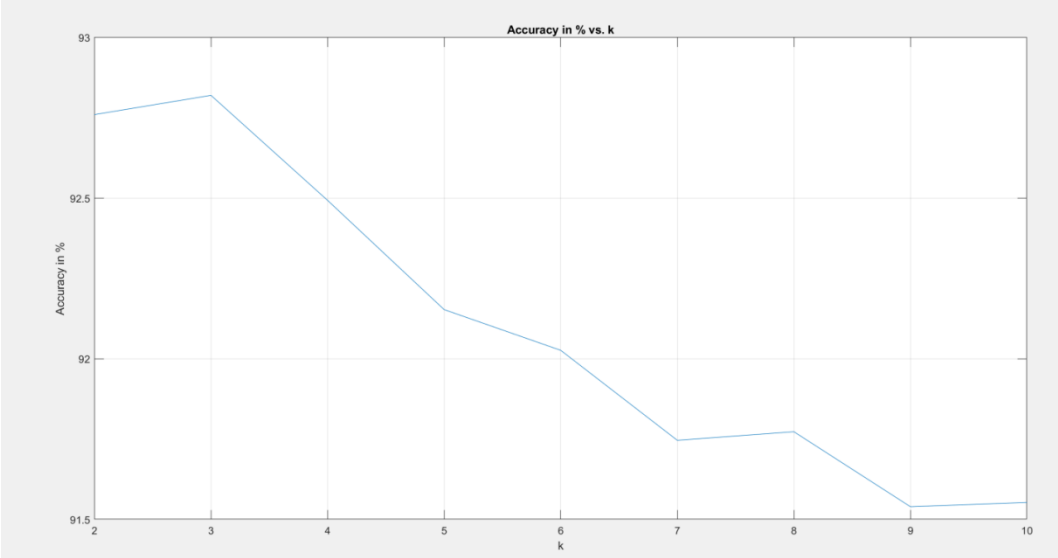


Fig-9 – Accuracy in % vs. k for Object Recognition data for range of k from 2 to 10

| | |
|---|---|
| **Dataset** | Phoneme |
| $k_{optimal}$ | 5 |
| **Testing Accuracy** | 86.34% |

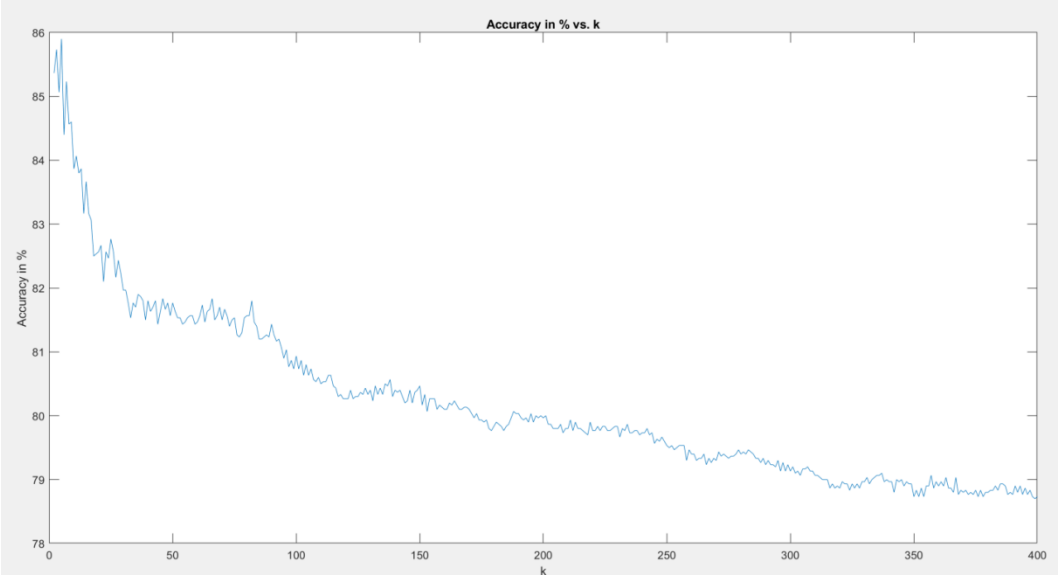


Fig-10 – Accuracy in % vs. k for Phoneme data for range of k from 2 to 400

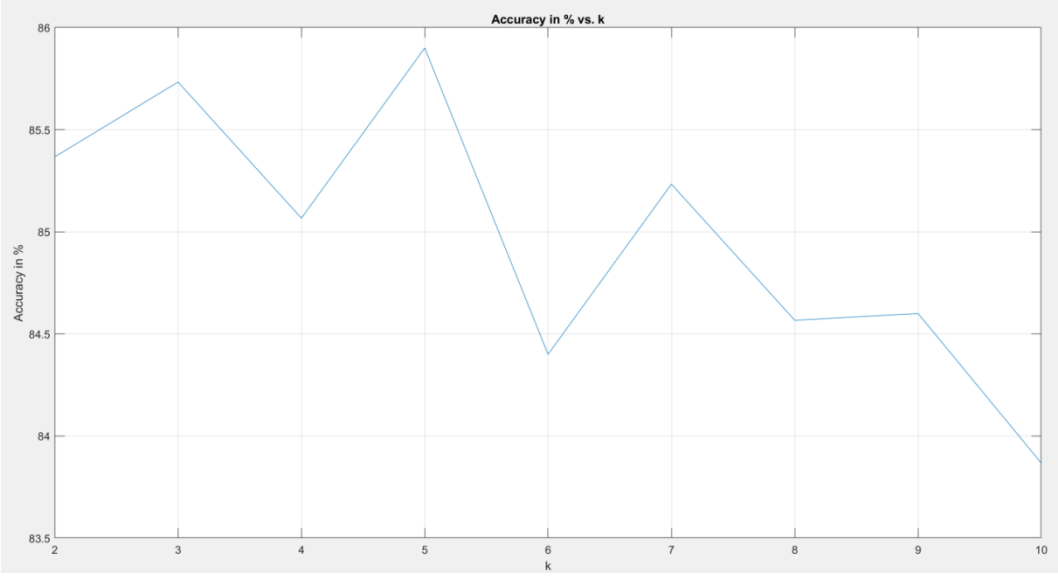


Fig-11 – Accuracy in % vs. k for Phoneme data for range of k from 2 to 10

| | |
|---|---|
| **Dataset** | Skin Segmentation |
| $k_{optimal}$ | 3 |
| **Testing Accuracy** | 99.94% |

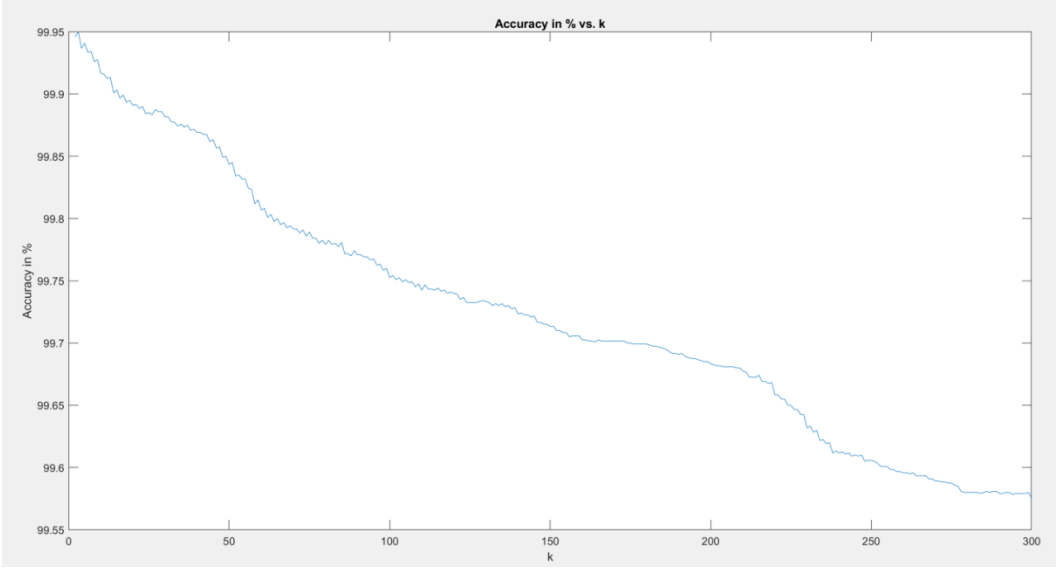


Fig-12 – Accuracy in % vs. k for Skin Segmentation data for range of k from 2 to 300

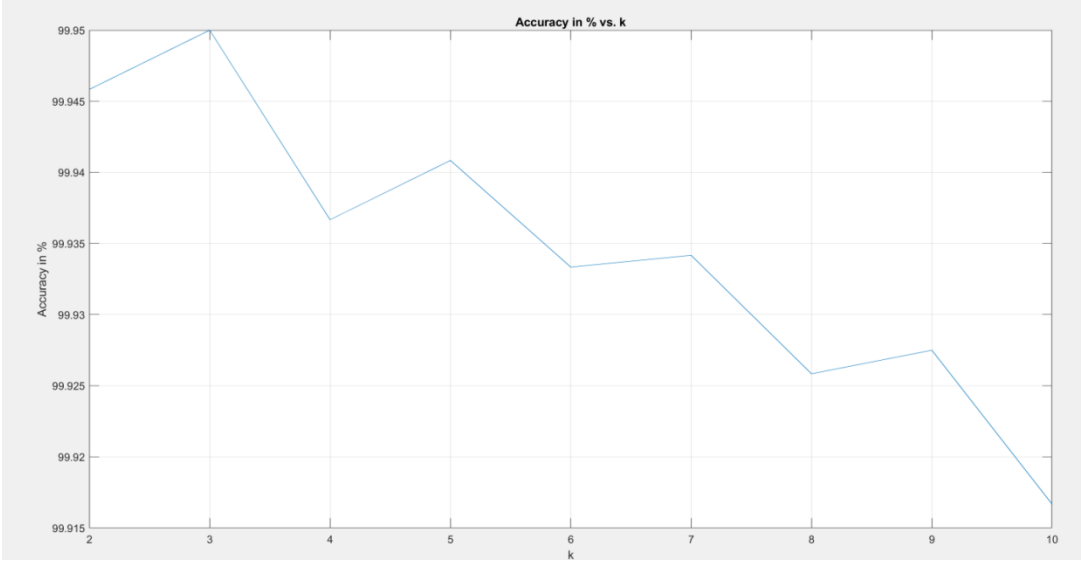


Fig-13 – Accuracy in % vs. k for Skin Segmentation data for range of k from 2 to 10

## 5.7 Improving the k-nearest neighbor classifier

The k-nearest neighbor classifier is arguably the simplest and most intuitively appealing nonparametric classification procedure [70]. But it is computationally expensive and it also faces the curse of dimensionality. It is computationally expensive because the classifier computes distance of the input test vector to all the input training vectors. This distance measurement is stored in the memory which causes a lot of memory to be used and large numbers of computations are done by the classifier for prediction which makes the classifier inefficient for real time purposes. Moreover, the accuracy of the nearest neighbor classifiers tends to decrease as the number of features or inputs increases [46]. The reason is that in a high-dimensional space all points tend to be far away from each other, so nearest neighbors are not meaningfully similar. The input training data can have noisy and less discriminative input features which can cause problems such as convergence difficulties, poor classification accuracy and contamination of the distance measure which leads to false classification. To solve these problems an improved k-nearest neighbor algorithm is proposed.

To solve the problem of high computational cost and curse of dimensionality, k-means++ clustering method is applied to the training data as in section 2.6.1, which reduces the input training vector by finding input center vectors. Then the distance measure optimization (DMO) algorithm, discussed in section 3.3, is applied which weights the input features in such a way so that the more important features are given higher weights and less significant features are assigned lower weights, hence noisy and less discriminative features separated and treated differently than the more important and significant features. The results of the experiment are provided in the next section.

## 5.8 Results of improving *k*-Nearest Neighbor Classifier

The accuracy of a k-Nearest neighbor classifier with distance measure optimization (k-$NNC_{DMO}$), regular k-nearest neighbor classifier (k-$NNC_R$), compared in the following table

| *Data Set* | *Number of inputs* | *Number of classes* | *$k_{optimal}$ for k-$NNC_R$* | *k-$NNC_R$ accuracy %* | *$k_{optimal}$ for k-$NNC_{DMO}$* | *k-$NNC_{DMO}$ accuracy %* |
|---|---|---|---|---|---|---|
| *F17C.dat* | *17* | *39* | *5* | *46.29* | *4* | *45.40* |
| *SKIN.dat* | *2* | *2* | *3* | *99.94* | *2* | *99.95* |
| *GONGTST.tst* | *16* | *10* | *4* | *86.4* | *3* | *87.16* |
| *COMF18.TRA* | *18* | *4* | *14* | *78.73* | *19* | *78.85* |
| *PHONEME.dat* | *5* | *2* | *5* | *86.34* | *3* | *87.62* |
| *OBJECT RECOG* | *576* | *2* | *3* | *93.62* | *2* | *93.62* |

Table 6- Efficiency k-$NNC_R$ vs. k-$NNC_{DMO}$

From Table 6 we can say k-$NNC_{DMO}$ gives better accuracy than k-$NNC_R$.

# Chapter 6

# RESULTS AND CONCLUSION

This chapter presents results on several data sets to show that the new methods are successful in improving traditional NNC. These datasets are described in details in Appendix B. Center vectors were generated from $N_v$ training patterns using K – Means clustering algorithm [45].

## 6.1 Results of different stages of pruning

Pruning can be implemented at different stages of the improved nearest neighbor classification algorithm. The different stages of pruning compared here are:-

1. Weight optimization, then running and then center vector optimization (OAWNNC + Pruning + CVO)
2. Weight Optimization, followed by center vector optimization and then applying pruning (OAWNNC + CVO + Pruning)
3. Weight optimization followed by pruning and then implementing center vector optimization method followed by weight optimization again.( OAWNNC + Pruning + CVO + WO)
4. Weight optimization followed by pruning (OAWNNC + Pruning)

The results are compared and displayed below for all four stages are displayed below,

| *Data Set* | *Number of inputs* | *Number of classes* | *OAWNNC + Pruning + CVO accuracy %* | *OAWNNC + CVO + Pruning accuracy %* | *OAWNNC + Pruning + CVO + WO accuracy %* | *OAWNNC + Pruning accuracy %* |
|---|---|---|---|---|---|---|
| *F17C.dat* | *17* | *39* | *86.61* | *84.83* | *86.61* | *88.41* |
| *SKIN.dat* | *2* | *2* | *99.21* | *99.26* | *99.31* | *99.35* |
| *GONGTST.tst* | *16* | *10* | *82.53* | *72.96* | *83.03* | *85.20* |
| *COMF18.TRA* | *18* | *4* | *78.77* | *77.92* | *78.77* | *75.78* |
| *PHONEME.dat* | *5* | *2* | *85.02* | *84.69* | *85.02* | *81.72* |
| *OBJECT RECOG* | *576* | *2* | *76.56* | *82.95* | *84.94* | *85.79* |

Table 7– Different stages of pruning

From the above table we can conclude that the best classification accuracy is achieved when pruning is applied after weight optimization.

### 6.2 Results of Pruning with respect to accuracy

This sub-section shows the improvements in the accuracy of the classification by applying the method of pruning of training center vectors after performing distance measure optimization. The results of pruning (OAWNNC + Pruning) is compared with regular NNC which uses all the training patterns ($NNC_R$), NNC which uses center vectors for training (NNCCV) and improved NNC by performing distance measure optimization and center vector optimization (OAWNNC + CVO).Table – 4 displays the compared results.

| *Data Set* | *Number of inputs* | *Number of classes* | *$NNC_{CV}$ accuracy %* | *OAWNNC + CVO accuracy %* | *OAWNNC + Pruning accuracy %* | *$NNC_R$ accuracy %* |
|---|---|---|---|---|---|---|
| *F17C.dat* | *17* | *39* | *25.50* | *69.02* | *88.41* | *46.10* |
| *SKIN.dat* | *2* | *2* | *93.85* | *95.42* | *99.35* | *99.95* |
| *GONGTST.tst* | *16* | *10* | *66.83* | *80.47* | *85.20* | *87.20* |
| *COMF18.TRA* | *18* | *4* | *54.28* | *77.28* | *78.77* | *73.64* |
| *PHONEME.dat* | *5* | *2* | *61.47* | *76.32* | *81.72* | *88.46* |
| *OBJECT RECOG* | *576* | *2* | *31.44* | *50.22* | *85.79* | *95.09* |

Table 8–Results of pruning compared with other Nearest Neighbor Classifiers

From Table 4 it is evident that implementation of pruning with distance measure optimization improves the accuracy of the classifier and provides the best result compared to any method nearest neighbor classification which uses center vectors for classification. But the nearest neighbor classifier which uses all the training patterns for classification has better accuracy than the improved nearest neighbor classifier with pruning. The reason being that more correct training data helps the classifier to find closest patterns from the testing pattern. But larger the training data set, higher is the complexity of classification and thus it takes more time for classification and results to lower efficiency of the classifier. The efficiency of different NNC is compared in the next section.

### 6.3 Results of pruning with respect to efficiency

Efficiency of the classifier is found by calculating the time taken by the classifier to find the classify the testing data. The lesser the time taken to classify, higher will be the efficiency of the

classifier. The experiments are done using the same hardware with the same background processes running on it, so that any external factors, like machine speed, number of threads available etc. does not affect the results of the experiment. The efficiency of the regular NNC, improved NNC with distance measure optimization and center vector optimization and improved NNC with distance measure optimization and pruning.

| *Data Set* | *Number of inputs* | *Number of classes* | *OAWNNC + CVO Time taken in sec* | *$NNC_R$ Time taken in sec* | *$NNC_{WO\text{-}Pr}$ Time taken in sec* |
|---|---|---|---|---|---|
| *F17C.dat* | *17* | *39* | *0.07256* | *0.770755* | *0.0621* |
| *SKIN.dat* | *2* | *2* | *11.58744* | *1854.2547* | *4.98541* |
| *GONGTST.tst* | *16* | *10* | *0.21585* | *1.396677* | *0.193966* |
| *COMF18.TRA* | *18* | *4* | *0.9867558* | *4.62871* | *0.39433* |
| *PHONEME.dat* | *5* | *2* | *0.546507* | *1.199788* | *0.40522* |
| *OBJECT RECOG* | *576* | *2* | *0.832548* | *192.99* | *0.758221* |

Table 9–Time taken by OAWNNC + CVO vs. $NNC_R$ vs. $NNC_{WO\text{-}Pr}$

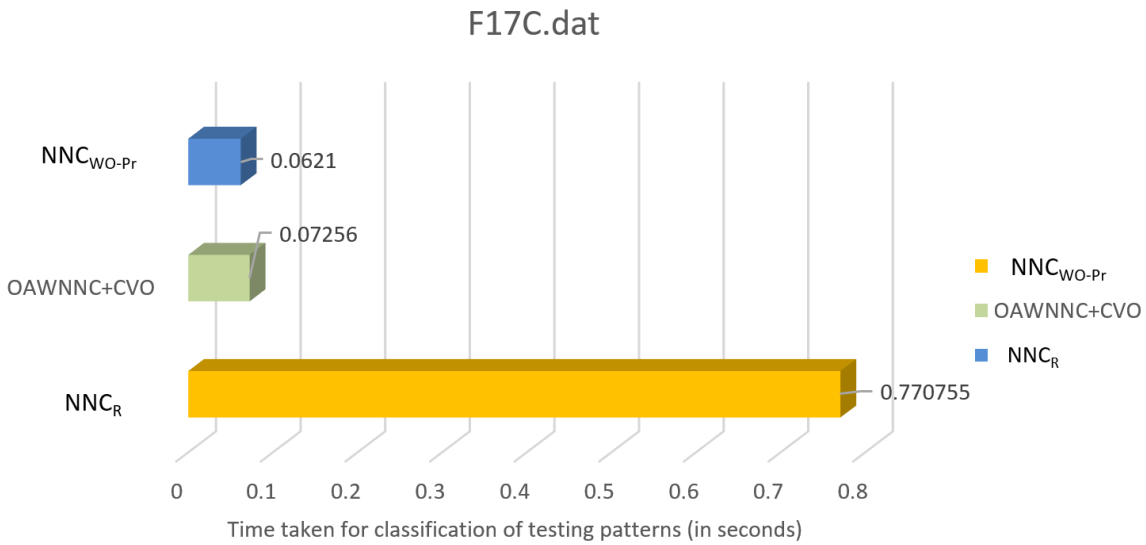


Fig-14 – Efficiency of $NNC_R$ vs. OAWNNC + CVO vs. $NNC_{WO\text{-}Pr}$ for F17C.dat file

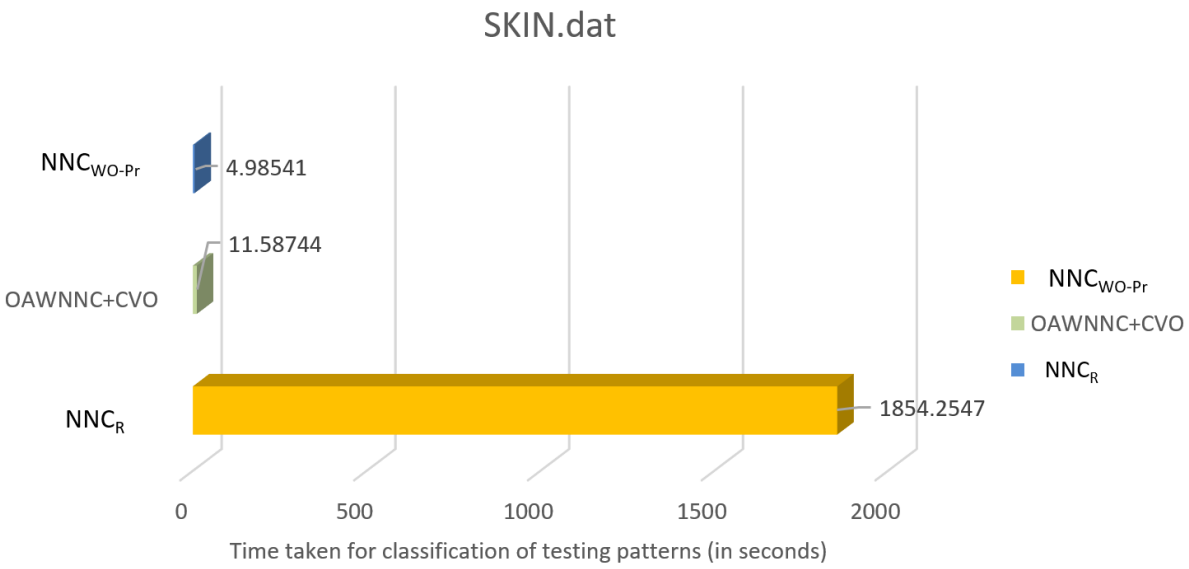


Fig-15 – Efficiency of $NNC_R$ vs. OAWNNC + CVO vs. $NNC_{WO\text{-}Pr}$ for SKIN.dat file

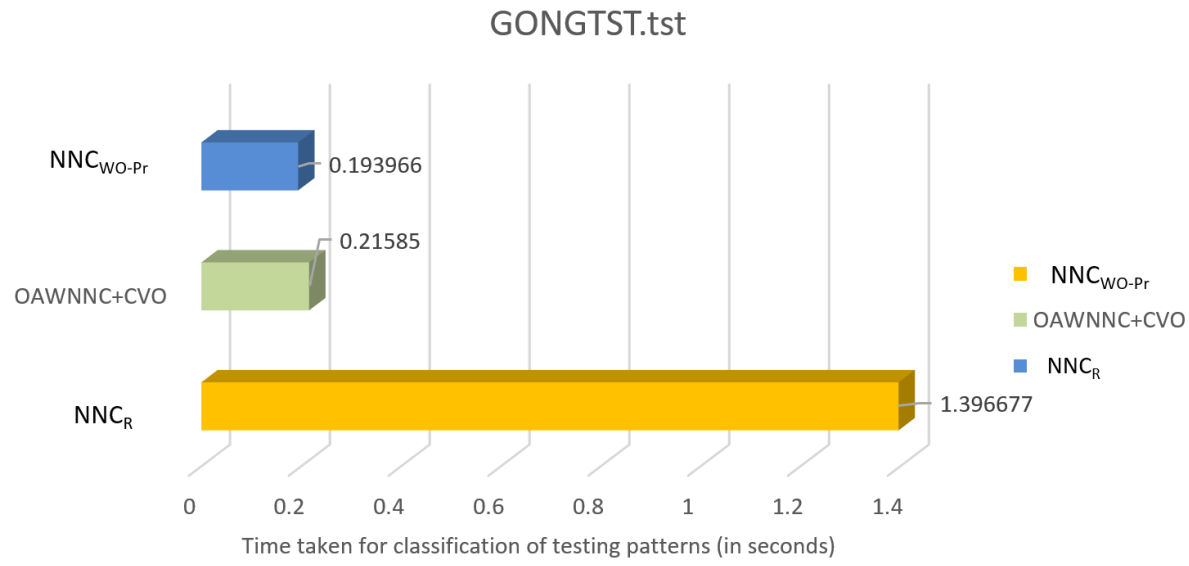


Fig-16 – Efficiency of $NNC_R$ vs. OAWNNC + CVO vs. $NNC_{WO\text{-}Pr}$ for GONGTST.tst file

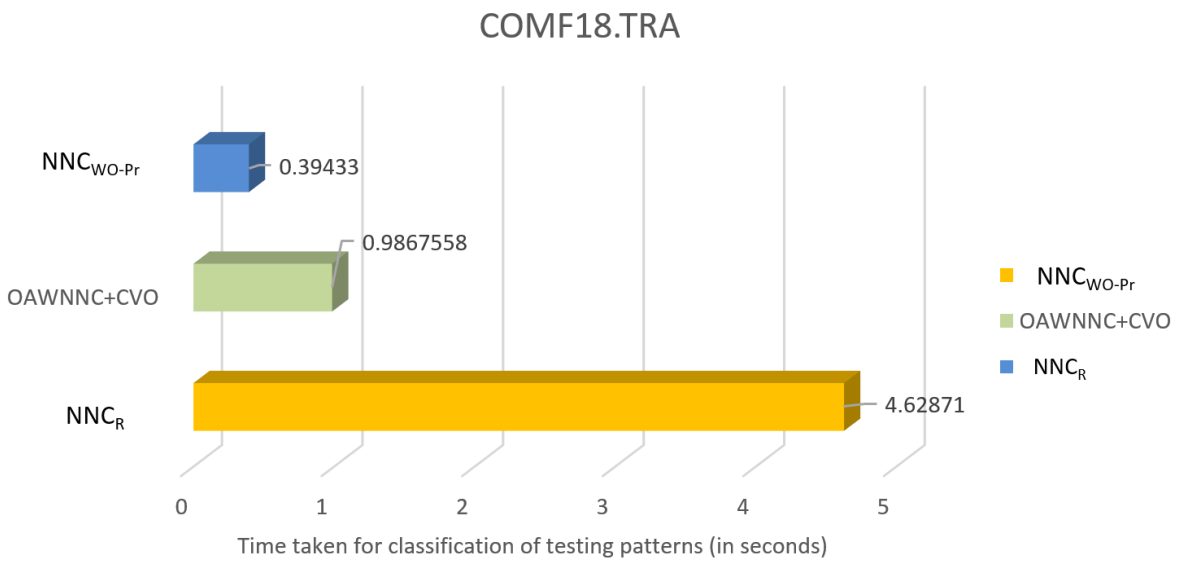


Fig-17 – Efficiency of $NNC_R$ vs. OAWNNC + CVO vs. $NNC_{WO-Pr}$ for COMF18.tra file

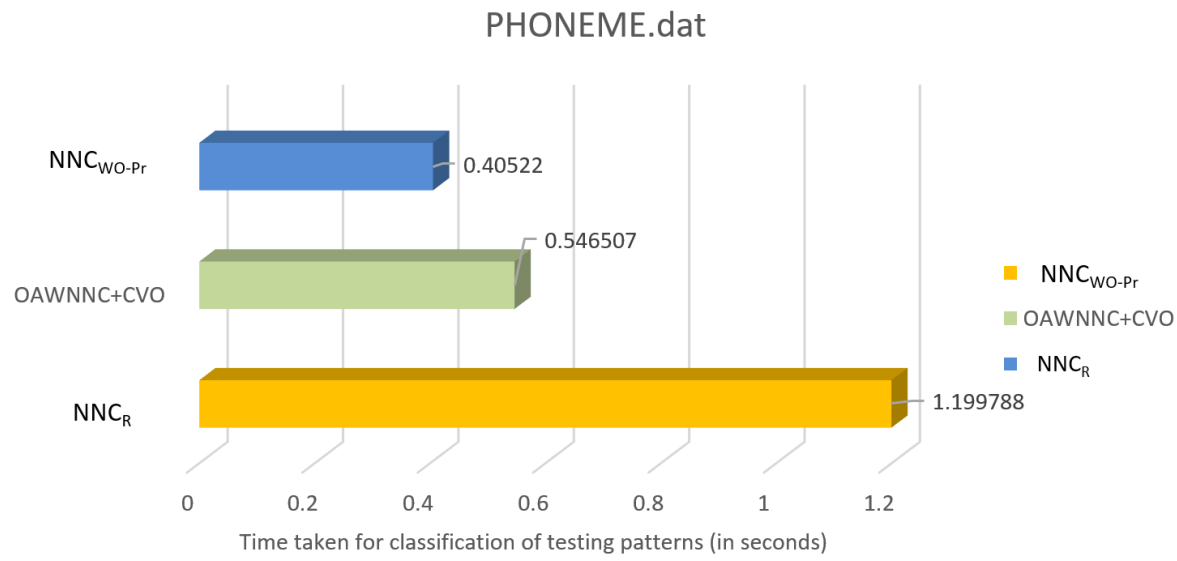


Fig-18 – Efficiency of $NNC_R$ vs. OAWNNC + CVO vs. $NNC_{WO-Pr}$ for PHONEME.dat file

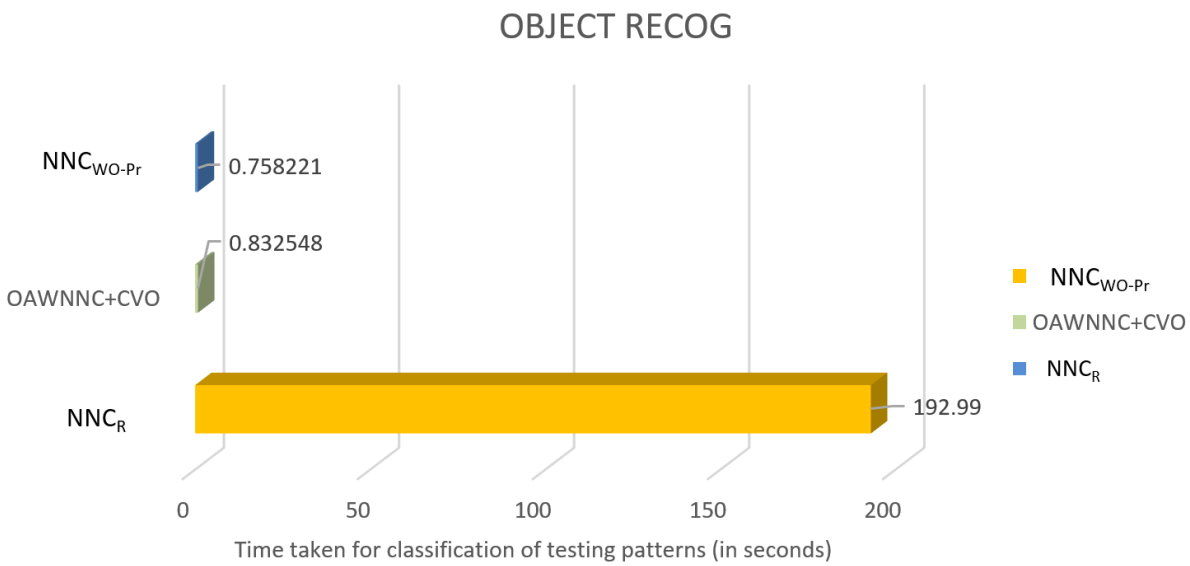


Fig-19 – Efficiency of $NNC_R$ vs. OAWNNC + CVO vs. $NNC_{WO\text{-}Pr}$ for PHONEME.dat file

### 6.4 Results of finding optimal k in k-NNC

A new algorithm to find the optimal *k* for a k-Nearest neighbor classifier is proposed in the Chapter 5, where passing through the training data only once, can determine optimal *k*. The method is tested on six different datasets, where the optimal *k* is found for each dataset and the training accuracy of the classifier for a range of *k* is plotted. Then the testing accuracy of the classifier for optimal *k* is recorded.

| *Data Set* | *Number of inputs* | *Number of classes* | $k_{optimal}$ | *Accuracy of k-NNC in %* |
|---|---|---|---|---|
| *F17C.dat* | *17* | *39* | *5* | *46.29* |
| *SKIN.dat* | *2* | *2* | *3* | *99.94* |
| *GONGTST.tst* | *16* | *10* | *4* | *86.4* |
| *COMF18.TRA* | *18* | *4* | *14* | *78.73* |
| *PHONEME.dat* | *5* | *2* | *5* | *86.34* |
| *OBJECT RECOG* | *576* | *2* | *3* | *93.62* |

Table 10 – Optimal *k* in k-NNC and its corresponding testing accuracy

## 6.5 Result of applying distance measure optimization on k-NNC

Contaminated input features in the dataset decreases the accuracy of a classifier. The noisy and less discriminative input features should be affect the classification results less than highly discriminative input features. To solve this problem of distance measure optimization, discussed in section 3.3, is applied to the k-Nearest neighbor classifier (k-$NNC_{DMO}$) and the accuracy of the k-$NNC_{DMO}$ is compared with the regular k-nearest neighbor classifier.

The regular k-Nearest neighbor classifier (k-$NNC_R$) uses the entire training pattern to for the purpose of prediction. Hence, every time there is a new input, the classifier has to calculate the distance between the input test vector and the entire training pattern, hence resulting into high computational complexity and high computational time for classification, making it inappropriate for real time applications. To solve this problem, clustering is applied to the input training vectors to reduce the training vector set. The accuracy of the *k*-nearest neighbor classifier using center vectors as training pattern (k-$NNC_{CV}$) is also displayed in the table.

| *Data Set* | *Number of inputs* | *Number of classes* | $k_{optimal}$ *for k-NNC$_{CV}$* | *k-NNC$_{CV}$ accuracy %* | $k_{optimal}$ *for k-NNC$_{R}$* | *k-NNC$_{R}$ accuracy %* | $k_{optimal}$ *for k-NNC$_{DMO}$* | *k-NNC$_{DMO}$ accuracy %* |
|---|---|---|---|---|---|---|---|---|
| *F17C.dat* | *17* | *39* | *2* | *23.45* | *5* | *46.29* | *4* | *45.40* |
| *SKIN.dat* | *2* | *2* | *2* | *97.06* | *3* | *99.94* | *2* | *99.95* |
| *GONGTST.tst* | *16* | *10* | *6* | *74.67* | *4* | *86.4* | *3* | *87.16* |
| *COMF18.TRA* | *18* | *4* | *6* | *67.83* | *14* | *78.73* | *19* | *78.85* |
| *PHONEME.dat* | *5* | *2* | *2* | *74.99* | *5* | *86.34* | *3* | *87.62* |
| *OBJECT RECOG* | *576* | *2* | *4* | *90.20* | *3* | *93.62* | *2* | *93.62* |

Table 11 – Accuracy of different k-NNC is compared

The result of efficiency of the regular k-Nearest neighbor classifier (k-NNC$_R$), *k*-nearest neighbor classifier using center vectors as training pattern (k-NNC$_{CV}$) and k-Nearest neighbor classifier with distance measure optimization is compared in the following table (k-NNC$_{DMO}$)

| *Data Set* | *Number of inputs* | *Number of classes* | *k-NNC$_R$ efficiency seconds* | *k-NNC$_{DMO}$ efficiency seconds* | *k-NNC$_{CV}$ efficiency seconds* |
|---|---|---|---|---|---|
| *F17C.dat* | *17* | *39* | *1.11* | *0.62* | *0.07799* |
| *SKIN.dat* | *2* | *2* | *2045.06* | *2128.70075* | *15.5212* |
| *GONGTST.tst* | *16* | *10* | *1.415827* | *1.549507* | *0.25084* |
| *COMF18.TRA* | *18* | *4* | *5.717* | *5.71094* | *0.1606* |
| *PHONEME.dat* | *5* | *2* | *1.217627* | *1.098723* | *0.134554* |
| *OBJECT RECOG* | *576* | *2* | *391.21* | *261.201977* | *1.091677* |

Table 12– Efficiency of different k-NNC is compared

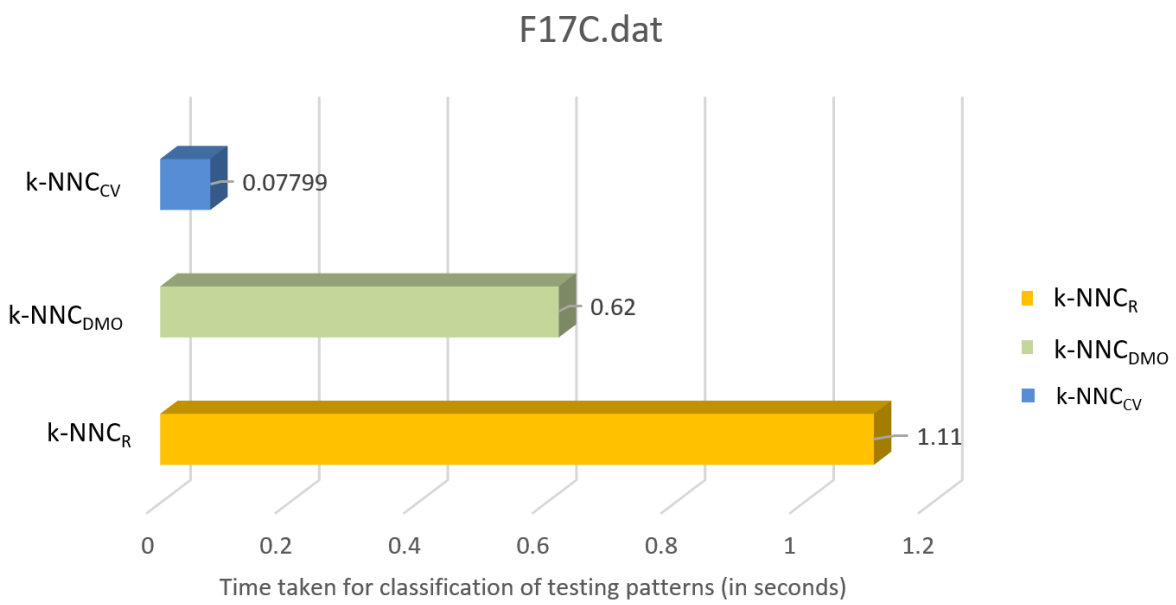


Fig-20 – Efficiency of k-NNC$_R$ vs. k-NNC$_{DMO}$ vs. k-NNC$_{CV}$ for F17C.dat

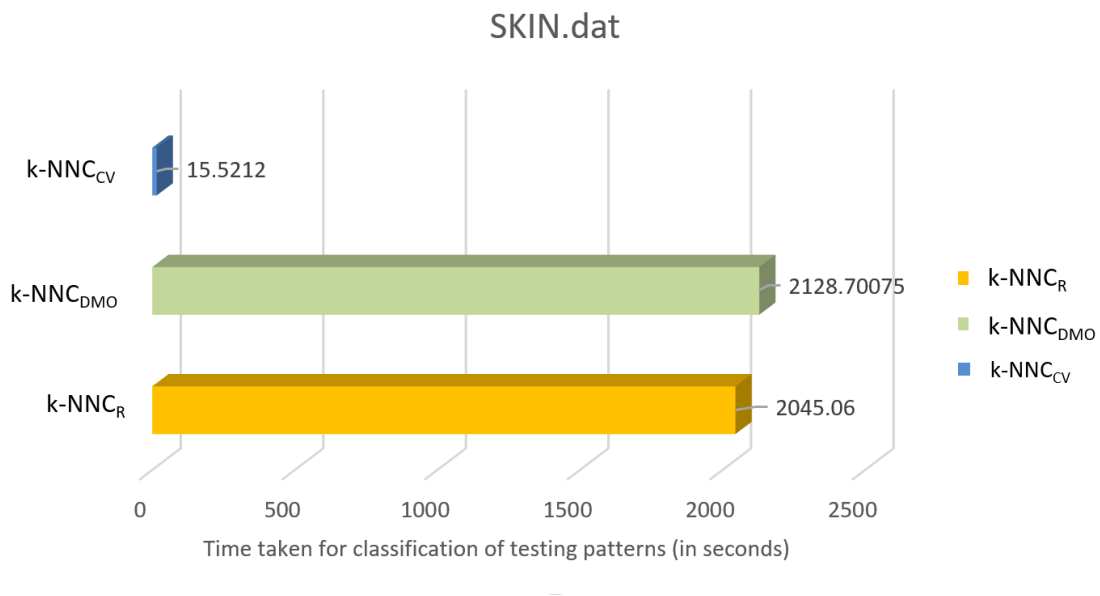


Fig-21 – Efficiency of k-NNC$_R$ vs. k-NNC$_{DMO}$ vs. k-NNC$_{CV}$ for SKIN.dat

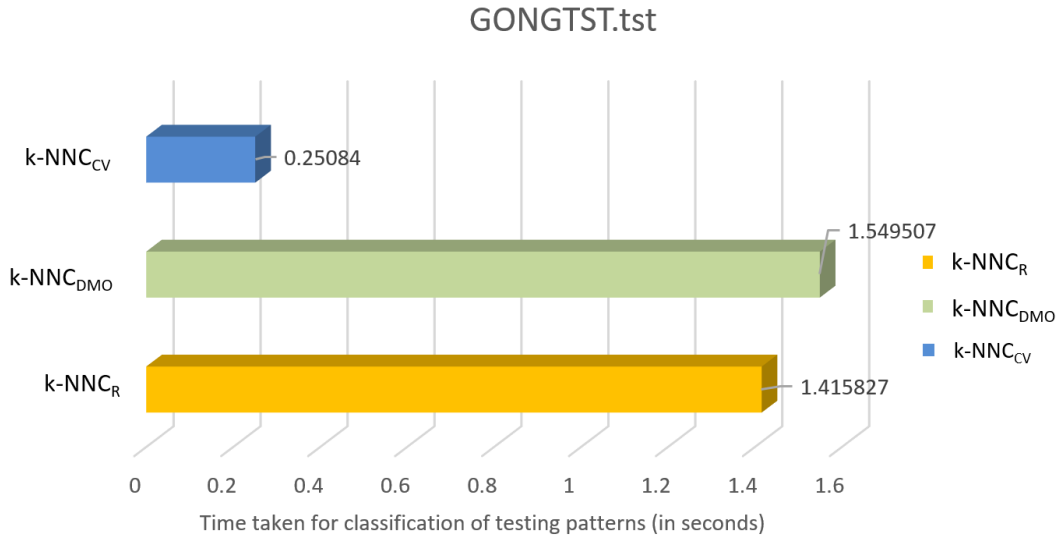


Fig-22 – Efficiency of k-NNC$_R$ vs. k-NNC$_{DMO}$ vs. k-NNC$_{CV}$ for GONGTST.tst

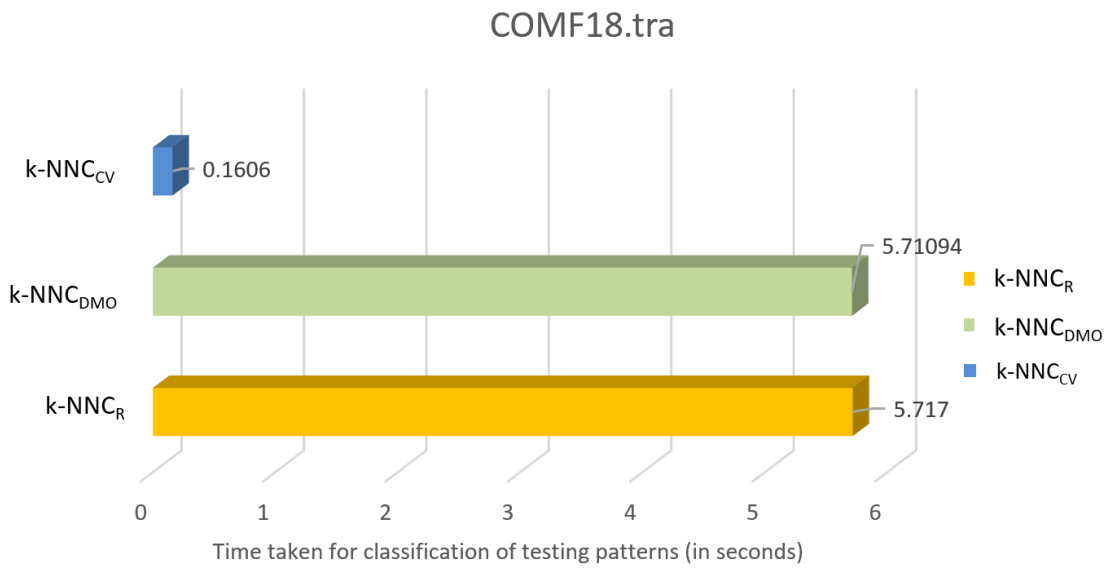


Fig-23 – Efficiency of k-$NNC_R$ vs. k-$NNC_{DMO}$ vs. k-$NNC_{CV}$ for COMF18.tra

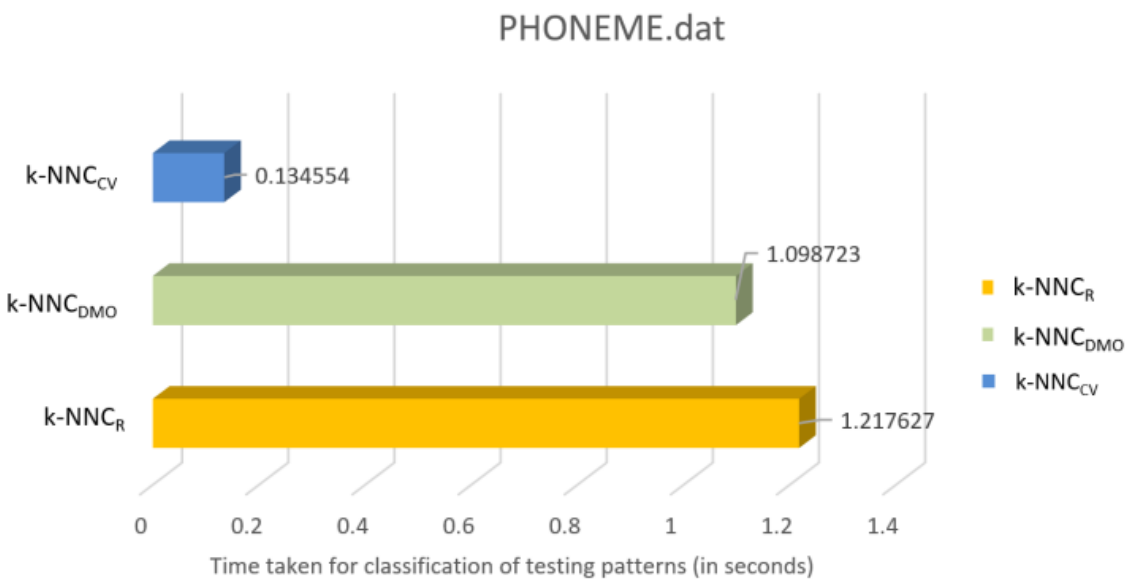


Fig-24 – Efficiency of k-$NNC_R$ vs. k-$NNC_{DMO}$ vs. k-$NNC_{CV}$ for PHONEME.dat

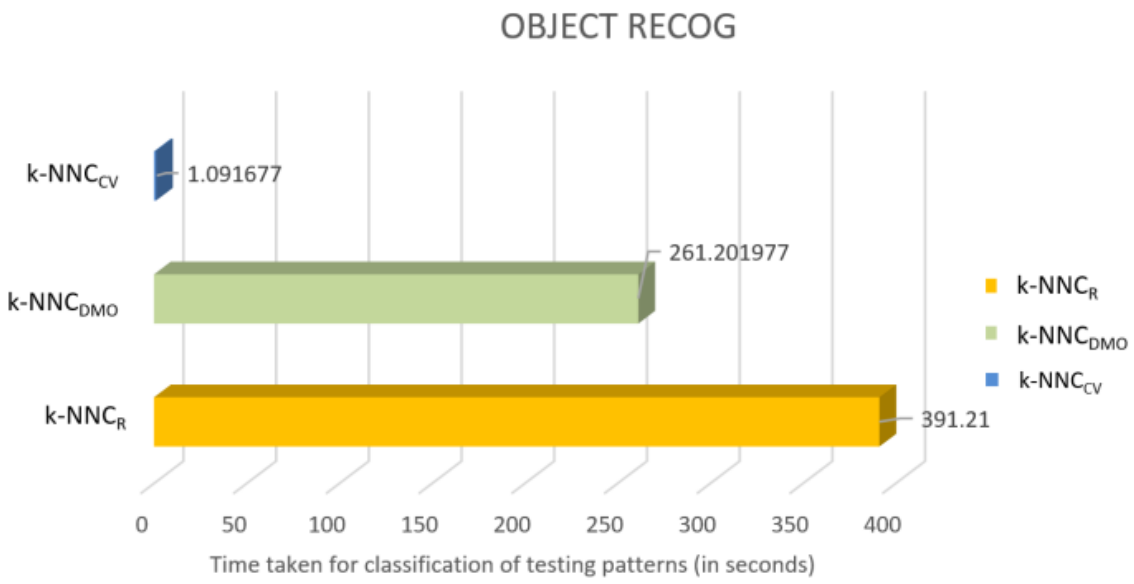


Fig-25 – Efficiency of k-$NNC_R$ vs. k-$NNC_{DMO}$ vs. k-$NNC_{CV}$ for OBJECT RECOG

From the above figures we can conclude that the k-$NNC_{CV}$ has the best efficiency. But as the accuracy of the k-$NNC_{CV}$ is much lower than regular k-nearest neighbor, it cannot be used unless further improvement is done on the algorithm.

## 6.6 Conclusion

From Table 8 we can conclude that applying pruning to the improved nearest neighbor classifier which already implements distance measure optimization, the accuracy is the best compared to any other forms of the improved nearest neighbor classifier. For datasets OBJECT RECOG and PHONEME.dat the regular nearest neighbor classifier out performs the proposed nearest neighbor classifier with respect to the accuracy and for SKIN.dat dataset both the classifier has almost the same classification accuracy. These three datasets have huge number of training patterns and hence the $NNC_R$ takes a large amount of time to perform classification and thus becomes inapplicable in real time applications. The nearest neighbor classifier implementing distance measure optimization and pruning that is proposed in this thesis has a significantly higher efficiency compared to the regular nearest neighbor classifier. Hence, pruning solves the problem of having outliers in the data and also noisy center vectors which affects the performance of the classifier. Pruning also improves the efficiency of the classifier.

In k-nearest neighbor classifier, the optimal *k* can be found by a single pass through the training data. The parameter *k* is dependent on the dataset, so finding the optimal *k* using an efficient method is highly useful in implementing the *k*-nearest neighbor classifier.

From Table 11 we can conclude that applying distance measure optimization to the k-nearest neighbor algorithm increases the accuracy of the classifier. Using center vectors as training vector for the k-nearest neighbor classifier increases the efficiency of the classifier but decreases the accuracy. The future endeavor in this field is to apply growing algorithm along with pruning to the nearest neighbor algorithm and applying pruning and center vector optimization along with the distance measure optimization to the k-nearest neighbor classifier.

Appendix A

**Optimizing distance measure weights using Newton's algorithm.**

1) Cluster $N_v$ patternsinto $K$ clusters, where $K = \sum_{i=1}^{Nc} k_i$.

   $k_i$ is the number of clusters of $i^{th}$ class

   After clustering we get $K$ center vectors$\boldsymbol{m_{ik}}$, where $\boldsymbol{m}_{ik}$ is the $k^{th}$ center vector of $i_{th}$ class

2) Initialize $w(n) = \frac{1}{var(x(n))}$for $1 \leq n \leq N$

3) Initialize $\boldsymbol{w_{old}} = \boldsymbol{w}$, $z_{old} = 0$

4) For $iter$ = 1 to $MaxIter$

   a) Initialize$\boldsymbol{g} = 0$, $\boldsymbol{H} = 0$

   a) For $p = 1\ to\ N_v$

      i. For $i = 1\ to\ N_c$

      i. Compute $d_i = \min\left(d(\boldsymbol{x}_p, \boldsymbol{m}_{i,k})\right)$ using equation (3.3)

      ii. Compute $\boldsymbol{y}$

      **End** $i$

      iii. Compute $MSE$ using equation (3.1) with one-hot encoding target output

      iv. Compute negative gradient vector $\boldsymbol{g}$using equation (3.4), (3.5) and (3.6)

      v. Compute Hessian matrix $\boldsymbol{H}$ using equation (3.13)

      **End** $p$

      vi. Solve equation (3.14) to compute update vector $\boldsymbol{e}$ using OLS [27]

      vii. Update weights as $w(n) \leftarrow w(n) + e(n)$

   **End** $iter$

5) Save Weights to disk

# Appendix B

## Description of datasets

*1 GONGTST.TST*

The raw data consists of images from hand printed numerals collected by the Internal Revenue Service. Images are 32 by 24 binary matrices. An image scaling algorithm is used to remove size variation in characters. This dataset contains 16 input features. The 10 classes correspond to 10 Arabic numerals.

*2 COMF18.TRA*

This dataset comes from [47] and has 18 input features to classify patterns into 4 distinct classes. These features are extracted from images as per Level 1 of the US Geological Survey Land Use/Land Cover Classification System to categorize into four regions of land use: urban areas, fields or open grass lands, trees (forest land) and water (lakes or river).

3 *F17C*

This dataset is used for the application of prognostics or flight condition recognition. It consists of parameters that are available in the basic health usage monitoring systems (HUMS), plus some others. The data was collected from M430 flight load level survey conducted in Mirabel Canada in early 1995. It has 17 input features and 39 classes.

*4 Skin Segmentation Data Set*

The skin dataset is collected by randomly sampling B,G,R values from face images of various age groups (young, middle, and old), race groups (white, black, and asian), and genders obtained from FERET database and PAL database [48]. Total

number of training patterns is 245057; out of which 50859 is the skin samples and 194198 is non-skin samples. It has 3 input features and 2classes.

*5 Phoneme Data Set*

This dataset distinguishes between nasal and oral sounds. It has 3818 patterns with 5 input features and 2 classes [49].

*6 Object Recognition Data Set*

The features are extracted from a trained Convolutional Neural Network (CNN) after throwing away the fully-connected layer at the top. So the features are the output of the last convolutional layer. The CNN was trained on 128x128 grayscale images.

References

[1] En.wikipedia.org. (2017). Statistical classification. [online] Available at: https://en.wikipedia.org/wiki/Statistical_classification [Accessed 4 Dec. 2017].

[2] Bishop, Christopher M. "Pattern recognition." Machine Learning 128 (2006).

[3] Cortes, C. and Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), pp.273-297.

[4] Rumelhart, D., Hinton, G. and Williams, R. (1986). Learning representations by back-propagating errors. Nature, 323(6088), pp.533-536.S

[5] Anon, (2017). [online] Available at: http://Quinlan, J. Ross. "Simplifying decision trees." International journal of man-machine studies 27.3 (1987): 221-234 [Accessed 4 Dec. 2017].

[6] Broder, A. (1990). Strategies for efficient incremental nearest neighbor search. Pattern Recognition, 23(1-2), pp.171-178.

[7] Broder, A. J. Strategies for efficient incremental nearest neighbour search. *Pattern Recognition* 23(1/2): 171–178. 1990.

[8] Shizen, Y. Wu, “An Algorithm for Remote Sensing Image Classification based on Artificial Immune b-cell Network”, Springer Berlin, Vol 40.

[9] Efficient Nearest Neighbor Classification Using a Cascade of Approximate Similarity Measures.

Vassilis Athitsos, Jonathan Alon, and Stan Sclaroff. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 486-493, June 2005.

[10] Online and Offline Character Recognition Using Alignment to Prototypes. Jonathan Alon, Vassilis Athitsos, and Stan Sclaroff.

International Conference on Document Analysis and Recognition (ICDAR), August 2005.

[11] G. Toker, O. Kirmemis, “Text Categorization using k Nearest Neighbor Classification”, Survey Paper, Middle East Technical University

[12] Y. Liao, V. R. Vemuri, “Using Text Categorization Technique for Intrusion detection”, Survey Paper, University of California.

[13] E. M. Elnahrawy, “Log Based Chat Room Monitoring Using Text Categorization: A Comparative Study”, University of Maryland.

[14] Anon, (2017). [online] Available at: http://Y. Yang and T. Ault, “Improving Text Categorization Methods for event tracking”, Carnegie Mellon University [Accessed 4 Dec. 2017].

[15] Csie.ntu.edu.tw. (2017). Cite a Website - Cite This For Me. [online] Available at: http://www.csie.ntu.edu.tw/~htlin/other/mlrt2010/MLRT-TsungHsienChiang.pdf [Accessed 4 Dec. 2017].

[16] F. Bajramovie et. al “A Comparison of Nearest Neighbor Search Algorithms for Generic Object Recognition”, ACIVS 2006, LNCS 4179, pp 1186-1197.

[17] En.wikipedia.org. (2017). Outlier. [online] Available at: https://en.wikipedia.org/wiki/Outlier [Accessed 4 Dec. 2017].

[18] Anon, (2017). [online] Available at: http://Wikipedia contributors. "K-nearest neighbors algorithm." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 28 Nov. 2016. Web. 28 Nov. 2016 [Accessed 4 Dec. 2017].

[19] En.wikipedia.org. (2017). Euclidean distance. [online] Available at: https://en.wikipedia.org/wiki/Euclidean_distance [Accessed 4 Dec. 2017].

[20] Ichino, Manabu, and Hiroyuki Yaguchi. "Generalized Minkowski metrics for mixed feature-type data analysis." IEEE Transactions on Systems, Man, and Cybernetics 24.4 (1994): 698-708.

[21] De Maesschalck, Roy, Delphine Jouan-Rimbaud, and Désiré L, Massart. "The mahalanobis distance." Chemometrics and intelligent laboratory systems 50.1 (2000): 1-18.

[22] En.wikipedia.org. (2017). Hamming distance. [online] Available at: https://en.wikipedia.org/wiki/Hamming_distance [Accessed 4 Dec. 2017].

[23] Hu, L., Huang, M., Ke, S. and Tsai, C. (2017). The distance function effect on k-nearest neighbor classification for medical datasets.

[24] En.wikipedia.org. (2017). Norm (mathematics). [online] Available at: https://en.wikipedia.org/wiki/Norm_(mathematics)#Euclidean_norm [Accessed 4 Dec. 2017].

[25] Wilson, D. Randall, and Tony R. Martinez. "Improved heterogeneous distance functions." Journal of artificial intelligence research 6 (1997): 1-34.

[26] Manry, Michael T. "Unsupervised Learning and Neural Nets That Use It." Neural networks EE 5353. University of Texas at Arlington. Texas. 16 November, 2016. Lecture.

[27] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd. edition, 1990, Academic Press.

[28] R.O. Duda, P.E. Hart, and D.G. Stork, Pattern Classification, John Wiley & Sons, 2nd edition, 2001.

[29] A. Papoulis, Probability, Random Variables, and Stochastic Processes, McGraw-Hill, New York, 1965.

[30] En.wikipedia.org. (2017). Cluster analysis. [online] Available at: https://en.wikipedia.org/wiki/Cluster_analysis [Accessed 11 Dec. 2017].

[31] Vajda, S., & Santosh, K. C. (2017). A Fast k-Nearest Neighbor Classifier Using Unsupervised Clustering. Communications in Computer and Information Science Recent Trends in Image Processing and Pattern Recognition, 185-193. doi:10.1007/978-981-10-4859-3_17

[32] Selim SZ, Ismail MA (1984) K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. IEEE Transactions on Pattern Analysis and Machine Intelligence 6(1):81{87

[33] Kohonen, Teuvo, Self-organization and associative memory. Vol. 8. Springer Science & Business Media, 2012.

[34] Manry, Michael T. "Unsupervised Learning and Neural Nets That Use It." Neural networks EE 5353. University of Texas at Arlington. Texas. 16 November, 2016. Lecture.

[35] Du, Qiang, Vance Faber, and Max Gunzburger, "Centroidal Voronoi tessellations: applications and algorithms." SIAM review 41.4 (1999): 637-676.

[36] Wikipedia contributors. "Mean squared error." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 18 Nov. 2016. Web. 18 Nov. 2016.

[37] Arthur, David, and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding." Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2007.

[38] Arthur, David, and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding." Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2007.

[39] En.wikipedia.org. (2017). Learning vector quantization. [online] Available at: https://en.wikipedia.org/wiki/Learning_vector_quantization [Accessed 11 Dec. 2017].

[40] Rawat Rohit, ManryMichael T. “Second Order Training of a Smoothed Piecewise Linear Network”. December 2016.

[41] M. D. Robinson, and M. T. Manry, “Two-Stage Second Order Training in Feedforward Neural Networks,” Twenty-Sixth International Florida Artificial Intelligence Research Society Conference, 2013.

[42] Ieeexplore.ieee.org. (2017). A pruned fuzzy k-nearest neighbor classifier with application to electrocardiogram based cardiac arrhytmia recognition - IEEE Conference Publication. [online] Available at: http://ieeexplore.ieee.org/document/4777725/ [Accessed 5 Dec. 2017].

[43] Wilson, D. R., & Martinez, T. R. (2000). Machine Learning, 38(3), 257-286. doi:10.1023/a:1007626913721

[44] Sci2s.ugr.es. (2017). Cite a Website - Cite This For Me. [online] Available at: http://sci2s.ugr.es/keel/pdf/algorithm/congreso/Zhao03Cpruner.pdf [Accessed 5 Dec. 2017].

[45] Batchelor, B. (2017). Growing and pruning a pattern classifier.

[46] En.wikipedia.org. (2017). Pruning (decision trees). [online] Available at: https://en.wikipedia.org/wiki/Pruning_(decision_trees) [Accessed 5 Dec. 2017].

[47] Scikit-learn.org. (2017). 3.3. Model evaluation: quantifying the quality of predictions — scikit-learn 0.19.1 documentation. [online] Available at: http://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics [Accessed 5 Dec. 2017].

[48] Kowalski, B. and Bender, C. (2017). K-Nearest Neighbor Classification Rule (pattern recognition) applied to nuclear magnetic resonance spectral interpretation.

[49] En.wikipedia.org. (2017). Feature extraction. [online] Available at: https://en.wikipedia.org/wiki/Feature_extraction [Accessed 5 Dec. 2017].

[50] Parsons, S. (2010). Introduction to Machine Learning, Second Editon by EthemAlpaydin, MIT Press, 584 pp., $55.00. ISBN 978-0-262-01243-0. The Knowledge Engineering Review,25(03), 353. doi:10.1017/s0269888910000056

[51] Ieeexplore.ieee.org. (2017). Feature Extraction Algorithm Based on K Nearest Neighbor Local Margin - IEEE Conference Publication. [online] Available at: http://ieeexplore.ieee.org/document/5344145/ [Accessed 5 Dec. 2017].

[52] En.wikipedia.org. (2017). K-nearest neighbors algorithm. [online] Available at: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm [Accessed 5 Dec. 2017].

[53] Arxiv.org. (2017). Cite a Website - Cite This For Me. [online] Available at: https://arxiv.org/ftp/arxiv/papers/1409/1409.0919.pdf [Accessed 5 Dec. 2017].

[54] Pdfs.semanticscholar.org. (2017). Cite a Website - Cite This For Me. [online] Available at: https://pdfs.semanticscholar.org/604b/32f6aac14f23b786e4da561af9cea766c3d3.pdf [Accessed 5 Dec. 2017].

[55] Ieeexplore.ieee.org. (2017). A K-nearest neighbor classifier for ship route prediction - IEEE Conference Publication. [online] Available at: http://ieeexplore.ieee.org/document/8084635/ [Accessed 5 Dec. 2017].

[56] Ieeexplore.ieee.org. (2017). Unknown aware k nearest neighbor classifier - IEEE Conference Publication. [online] Available at: http://ieeexplore.ieee.org/document/7983027/ [Accessed 5 Dec. 2017].

[57] Ieeexplore.ieee.org. (2017). Genetic programming and K-nearest neighbour classifier based intrusion detection model - IEEE Conference Publication. [online] Available at: http://ieeexplore.ieee.org/document/7943121/ [Accessed 5 Dec. 2017].

[58] Citeseerx.ist.psu.edu. (2017). Cite a Website - Cite This For Me. [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.1337&rep=rep1&type=pdf [Accessed 5 Dec. 2017].

[59] En.wikipedia.org. (2017). Cross-validation (statistics). [online] Available at: https://en.wikipedia.org/wiki/Cross-validation_(statistics) [Accessed 5 Dec. 2017].

[60] Geisser, Seymour (1993). Predictive Inference. New York, NY: Chapman and Hall. ISBN 0-412-03471-9.

[61] Kohavi, Ron (1995). "A study of cross-validation and bootstrap for accuracy estimation and model selection". Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. San Mateo, CA: Morgan Kaufmann. 2 (12): 1137–1143.

[62] Devijver, Pierre A.; Kittler, Josef (1982). Pattern Recognition: A Statistical Approach. London, GB: Prentice-Hall.

[63] Cs.utoronto.ca. (2017). Cite a Website - Cite This For Me. [online] Available at: http://www.cs.utoronto.ca/~fidler/teaching/2015/slides/CSC411/tutorial3_CrossVal-DTs.pdf [Accessed 5 Dec. 2017].

[64] Manry, Michael T. "Unsupervised Learning and Neural Nets That Use It." Neural networks EE 5353. University of Texas at Arlington. Texas. 16 November, 2016. Lecture.

[65] Arxiv.org. (2017). Cite a Website - Cite This For Me. [online] Available at: https://arxiv.org/ftp/arxiv/papers/1409/1409.0919.pdf [Accessed 5 Dec. 2017].

[66] Cs.haifa.ac.il. (2017). Cite a Website - Cite This For Me. [online] Available at: http://www.cs.haifa.ac.il/~rita/ml_course/lectures/KNN.pdf [Accessed 5 Dec. 2017].

[67] M. Jirina and M. J. Jirina, "Classifier Based on Inverted Indexes of Neighbors," Institute of Computer Science, Technical Report No. V-1034, 2008.

[68] M. Jirina and M. J. Jirina, "Using Singularity Exponent in Distance Based Classifier," in Proceedings of the 10th International Conference on Intelligent Systems Design and Applications (ISDA2010), Cairo, 2010, pp. 220-224.

[69] M. Jirina and M. J. Jirina, "Classifiers Based on Inverted Distances," in New Fundamental Technologies in Data Mining, K. Funatsu, Ed. InTech, 2011, vol. 1, ch. 19, pp. 369-387.

[70] Hall, P., Park, B. and Samworth, R. (2017). Choice of neighbor order in nearest-neighbor classification.

[71] Ieeexplore.ieee.org. (2017). Multi-agent event detection system using k-nearest neighbor classifier - IEEE Conference Publication. [online] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6914382 [Accessed 11 Dec. 2017].

[72] Cs.cmu.edu. (2017). Cite a Website - Cite This For Me. [online] Available at: http://www.cs.cmu.edu/~aarnold/cald/fp025-geng.pdf [Accessed 11 Dec. 2017].

[73] Proceedings.mlr.press. (2017). Cite a Website - Cite This For Me. [online] Available at: http://proceedings.mlr.press/v25/chiang12/chiang12.pdf [Accessed 11 Dec. 2017].

[74] Jiang, Liangxiao, et al. “Learning k-Nearest Neighbor Naive Bayes for Ranking.” Advanced Data Mining and Applications Lecture Notes in Computer Science, 2005, pp. 175–185., doi:10.1007/11527503_21.

[75] Wang, S. and Liu, Z. (2010). Infrared Face Recognition Based on Histogram and K-Nearest Neighbor Classification. Advances in Neural Networks - ISNN 2010, pp.104-111.

[76] Ebrahimpour-ko..., H. “FACE RECOGNITION USING BAGGING KNN.”Users.cecs.anu.edu.au, www.academia.edu/1080778/FACE_RECOGNITION_USING_BAGGING_KNN.

[77] Govindarajan, M., & Chandrasekaran, R. (2009). Intrusion detection using k-Nearest Neighbor. 2009 First International Conference on Advanced Computing. doi:10.1109/icadvc.2009.5377998

[78] Li, Wenchao, et al. “A New Intrusion Detection System Based on KNN Classification Algorithm in Wireless Sensor Network.” Journal of Electrical and Computer Engineering, vol. 2014, 2014, pp. 1–8., doi:10.1155/2014/240217.

[79] Web.cs.ucdavis.edu. (2017). Cite a Website - Cite This For Me. [online] Available at: http://web.cs.ucdavis.edu/~vemuri/papers/knn-ss02.pdf [Accessed 11 Dec. 2017].

[80] Kang, Myunga, and Jongmin Kim. “Real Time Object Recognition Using K-Nearest Neighbor in Parametric Eigenspace.” Bio-Inspired Computational Intelligence and Applications Lecture Notes in Computer Science, pp. 403–411., doi:10.1007/978-3-540-74769-7_44.

[81] R.Muralidharan, Dr. “Object Recognition Using K-Nearest Neighbor Supported By Eigen Value Generated From the Features of an Image.” International Journal of Innovative Research in Computer and Communication Engineering, Research and Reviews, 1 Jan. 1970, www.rroij.com/open-access/object-recognition-using-knearest-neighborsupported-by-eigen-value-generated-fromthe-features-of-an-image.php?aid=46808.

[82] Boiman, O., Shechtman, E., & Irani, M. (2008). In defense of Nearest-Neighbor based image classification. 2008 IEEE Conference on Computer Vision and Pattern Recognition. doi:10.1109/cvpr.2008.4587598

[83] Y. Song, J. Huang, D. Zhou, H. Zha, and C. L. Giles, "Iknn: Informative k-nearest neighbor pattern classification," in Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, 2007, pp. 248-264.

[84] M. Latourrette, "Toward an explanatory similarity measure for nearest-neighbor classification," in Proceedings of the 11th European Conference on Machine Learning, London, 2000, pp. 238-245.

[85] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN ModelBased Approach in Classification," Lecture Notes in Computer Science, vol. 2888, pp. 986-996, 2003.

[86] Y. Hamamoto, S. Uchimura, and S. Tomita, "A Bootstrap Technique for Nearest Neighbor Classifier Design," IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, vol. 19, no. 1, pp. 73-79, 1997.

[87] M. Jirina and M. J. Jirina, "Classifier Based on Inverted Indexes of Neighbors," Institute of Computer Science, Technical Report No. V-1034, 2008.

[88] M. Jirina and M. J. Jirina, "Using Singularity Exponent in Distance Based Classifier," in Proceedings of the 10th International Conference on Intelligent Systems Design and Applications (ISDA2010), Cairo, 2010, pp. 220-224.

[89] ] M. Jirina and M. J. Jirina, "Classifiers Based on Inverted Distances," in New Fundamental Technologies in Data Mining, K. Funatsu, Ed. InTech, 2011, vol. 1, ch. 19, pp. 369-387.

[90] “Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach.” Table I from Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach - Semantic Scholar, www.semanticscholar.org/paper/Solving-the-Problem-of-the-K-Parameter-in-the-KNN-Hassanat-Abbadi/e0c716aee8e5b960515d3928e84e07baa0fcb7aa/figure/2.

[91] Selim SZ, Ismail MA (1984) K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. IEEE Transactions on Pattern Analysis and Machine Intelligence 6(1):81{87

[92] Kohonen, Teuvo, Self-organization and associative memory. Vol. 8. Springer Science & Business Media, 2012.

[93] En.wikipedia.org. (2017). DBSCAN. [online] Available at: https://en.wikipedia.org/wiki/DBSCAN [Accessed 11 Dec. 2017].

[94] Du, Qiang, Vance Faber, and Max Gunzburger, "Centroidal Voronoi tessellations: applications and algorithms." SIAM review 41.4 (1999): 637-676.

[95] Manry, Michael T. "Unsupervised Learning and Neural Nets That Use It." Neural networks EE 5353. University of Texas at Arlington. Texas. 16 November, 2016. Lecture.

[96] Sheth, Jugal R. “OPTIMAL ATTRIBUTE WEIGHTING IN A NEAREST NEIGHBOR CLASSIFIER .” University of Texas at Arlington, UTA, 2016, pp. 1–53.

## Biographical Information

Sinchan Bhattacharya was born in India in 1991. He received his Bachelor of Technology in Electrical Engineering from West Bengal University of Technology, Kolkata, India in May 2013 and Master of Science in Electrical Engineering from the University of Texas at Arlington in December 2017.

He interned at Unique Software Development in spring 2017 and fall 2016. He has been involved in research activities under the guidance of Dr. Michael T. Manry in Image Processing and Neural Networks Laboratory (IPNNL) since 2015. His main research interests include Neural Networks and Pattern Recognition.