Editing the Nearest Feature Line Classifier

Kamran Kamaei

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
February 2013
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Muhammed Salamah
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Hakan Altınçay
Supervisor

Examining Committee

1. Prof. Dr. Hakan Altınçay

2. Assoc. Prof. Dr. Ekrem Varoğlu

3. Asst. Prof. Dr. Adnan Acan

ABSTRACT

The main drawbacks in Nearest Feature Line classifier are the extrapolation and interpolation inaccuracies. The former can easily be counteracted by considering segment rather than lines. However, the solution of the latter problem is more challenging. Recently developed techniques tackle with this drawback by selecting a subset of the feature line segments either during training or testing. In this study, a novel framework is developed that involves a discriminative component. The proposed approach is based on editing the feature line segments. It involves three major steps namely, error-based deletion, intersection-based deletion and pruning. The first step compares the benefit and cost of deleting each feature line segment and deletes those that contribute more to the classification error. For the implementation of the second step, a novel measure of intersection is defined and used for line segments in high dimensions to delete the longest of two intersecting segments. The pruning step re-evaluates the retained segments by considering their distances from the samples belonging to the other classes. The proposed approach is evaluated on fifteen real datasets from different domains. Experimental results have shown that the proposed scheme achieves better accuracies on majority of these datasets compared to two recently developed extensions of the nearest feature line approach, namely the rectified nearest feature line segment and shortest feature line segment on majority of these datasets.

Keywords: Pattern classification; nearest feature line; line segment editing; interpolation inaccuracy; extrapolation inaccuracy.

ÖZ

Enyakın öznitelik çizgisi sınıflandırıcısının en önemli zayıflıkları ekstrapolasyon ve interpolasyon hatalarıdır. İlki çizgiler yerine çizgi parçaları kullanılarak kolaylıkla telafi edilebilir. Ancak, sonraki problemin çözümü daha zorludur. Son dönemde önerilen yöntemler bu sorunla eğitme veya sınama aşamalarında öznitelik çizgi parçalarının altkümelerini seçerek başa çıkmaya çalışmaktadırlar. Bu çalışmada, ayırt edici bileşen de içeren yeni bir çerçeve geliştirilmiştir. Önerilen yöntem öznitelik çizgi parçalarını azaltmaya dayanmaktadır. Bu yaklaşım hataya-dayalı silme, kesmeye-dayalı silme ve budama olmak üzere toplam üç basamak içermektedir. Birinci aşama, her öznitelik çizgi parçasını silmenin kazanım ve bedelini karşılaştırır ve sınıflandırma hatasına katkı yapanları siler. İkinci basamağın uygulanması için yeni bir kesişme tanımı yapılmış ve yüksek boyutlu öznitelik uzayında kesişen öznitelik parçalarının uzun olanını silmek için kullanılmıştır. Budama aşamasında, geriye kalan öznitelik çizgi parçaları diğer sınıflara ait eğitme verisine olan uzaklıkları dikkate alınarak yeniden değerlendirilmiştir. Önerilen yöntem, farklı alanlardaki onbeş gerçek veri kümesi üzerinde denenmiştir. Deneysel sonuçlar, önerilen yöntemin son yıllarda enyakın öznitelik çizgisi yaklaşımının uzantısı olarak geliştirilen düzeltilmiş en yakın öznitelik çizgi parçası ve en kısa öznitelik çizgi parçası isimli yaklaşımlara göre, veri kümelerinin çoğunda daha iyi başarım elde ettiğini göstermiştir.

Anahtar Kelimeler: Örüntü sınıflandırma; en yakın öznitelik çizgisi; çizgi parçası seçme; interpolasyon hatası; ekstrapolasyon hatası.

ACKNOWLEDGMENT

I would never have been able to complete this dissertation without the help of the people who have supported me with their best wishes.

I would like to express my deepest gratitude and thanks to my supervisor, Prof. Dr. Hakan Altınçay, for his advice, support, guidance and sponsorship throughout my study at Eastern Mediterranean University. I sincerely thank to the committee members of my thesis defense jury for their helpful comments on this thesis.

Last but not least I would also like to thank my dear parents, my brothers, and younger sisters for their continuous supports in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Pattern Classification

Pattern classification is the science of labeling an unseen data as one of the known groups or categories [1, 2]. Some examples of data are speech signal, facial image, iris, handwritten word and e-mail message. Mostly, the classification algorithms match the input to the a priori defined categories by considering their statistical characteristics.

In pattern classification problem, a *class* denotes a group of objects that have common properties. For example, in the face recognition problem, the group of different facial images belonging to a different person forms a class. As another example, if we need to design an automated system for fish packing to detect different types of fish, then any type of fish forms a different class.

The first step in designing an automated classification system is defining the method of representing different objects. This step is problem dependent. Consider the fish packing problem. Raw data measurements such as length and weight, derived measurements or features (e.g. ratio of length to weight), a structural description such as length to weight ratios of different parts of the fish and spatial relationship of the various parts can be considered. Feature based representation approach is the most common. A *feature* is any distinctive aspect, quality or characteristic related with the

objects to be classified. A *feature vector* of an object represents a combination of features as an N-dimensional column vector where each entry corresponds to a different feature or measurement.

Each object employed in the classification is known as a *sample* and a collection of samples is named as a *dataset*. For example, in face recognition problems, each facial image that is available in the dataset is a different sample.

A pattern classification system is typically made up of two phases, *training phase* and *test phase*, as it is shown in Figure 1 [3]. The data acquisition step corresponds to getting the input from the physical environment by measuring physical variables such as recording the speech signal using a microphone or capturing the image of a person. Pre-processing methods tries to remove noises and redundant inputs. Feature extraction involves definition of measures for accurate description of raw input data. Small number of features may not be discriminative while larger number of features may lead to more complex classification models. Model estimation is used to compute a decision boundary or decision regions in the feature space. At the classification step, the classifier uses the trained model to map the input feature vectors onto one of the classes and this leads to the final decision for each sample.

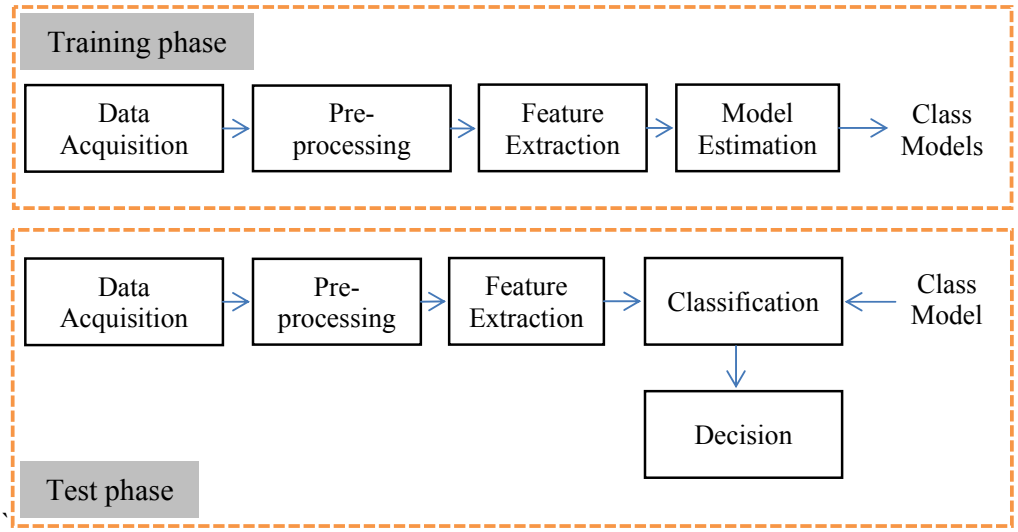

Figure 1: The main blocks of a pattern classification system.

Classifiers are roughly categorized into two groups: Parametric and non-parametric methods. In the parametric approach, the main aim is to fit a parametric model to the training data and interpolate to classify test data. For instance, the parametric methods may assume a specific functional form for the probability density function and optimize the function parameters to fit the training data. Some of these methods are Linear Discriminant Classifiers (LDC) and Quadratic Discriminant Classifier (QDC) [4]. In the non-parametric methods, no assumptions are made about the probability density function for each class, because an assumed function may not fit the training data. Therefore, the non-parametric methods determine the form of the probability density function from the data. Some widely used non-parametric methods are nearest neighbor classifier, neural networks and support vector machines [1].

The Nearest neighbor classifier (NN) is a simple yet effective non-parametric scheme that chooses the label of the nearest training sample as the final decision [5]. An extended version is *k*-NN [6] which makes the decisions by voting on the labels of

the *k* nearest neighbors of the test sample. The training phase is not intense. All data samples and their labels are stored. In case of real valued feature vectors, the most common function for the calculation of distances is the Euclidean metric [7].

Although, it is easy to implement and debug, *k*-NN approach has some disadvantages which are namely high computational cost and sensitivity to the outliers [6]. Moreover, there is a need for large number of samples for reasonable performance. In particular, as a geometrical neighborhood approach, the performance increases as the number of training samples increases [1]. It is known that the error of *k*-NN approaches to Bayes error rate as the number of samples goes to infinity [1]. However, in practice there will be limited number of samples due to practical restrictions in their collection. In cases where the training data is limited, the training data will not be able to represent the characteristics of the pattern classes and hence the performance of the *k*-NN will be below acceptable limits. To counteract the data insufficiency problem, nearest feature line (NFL) method is proposed as an extension of nearest neighbor approach [5].

NFL aims to generalize the representational capacity of the training data by considering lines passing through each pair of samples from the same class that are named as feature lines (FL) [5]. With the use of lines, NFL is generally argued to add information to the given data. NFL is originally proposed and successfully used for the face recognition problem [5]. However, it has been proved to achieve consistently better performance than the NN in terms of the error rate in many real and artificial data [8]. Classification by NFL is done by computing the distances from the test sample to all feature lines where the class to which nearest feature line belongs is selected as the final decision.

NFL has two major drawbacks, namely the interpolation and extrapolation inaccuracy [9]. Interpolation inaccuracy occurs when a feature line is defined using samples that are far away from each other. Such lines may pass through the regions where other classes exist. Consequently, such a line may be computed as the nearest for the samples belonging to a different class. The extrapolation inaccuracy occurs when a feature line passes through samples that are away from the test point [10]. In the NN and *k*-NN methods, for N training samples in a given class, N distances are computed. However, NFL suffers from increased computational complexity as well since N(N-1)/2 feature lines are defined using N samples [5].

It should be noted that NFL based approaches are employed for the classification problems involving real valued features. The main reason is that the concept of generalization using feature lines is not sensible in the case of binary features.

Following this technique, several editions are developed to reduce the error and/or the computational cost. Center-based nearest neighbor (CNN) [11] was proposed to reduce the computational cost of the NFL method by using center-based feature lines. The center-based feature lines are defined as the lines passing through each training sample and the center of all samples belonging to the class [9, 11]. During classification, the decision is made by finding the nearest center-based feature line to the query point. Experiments have shown that CNN achieves enhanced performance compared to NN and comparable performance with NFL [11]. Another approach for reducing the computational cost is the nearest neighbor line (NNL) [12]. It uses the line through the nearest pair of samples from each class during the classification phase. In other words, a single line for each class is considered. Experiments on face

recognition have shown that NNL has much lower computation time and achieves competitive performance compared to the NFL method [13].

More advanced methods are also proposed mainly to suppress the interpolation and extrapolation inaccuracies. The rectified nearest feature line segment technique (RNFLS) [14] uses FL segments so as to avoid extrapolation inaccuracy where a feature line segment (FLS) is defined as the region of a FL that is in between the corresponding samples. In order to suppress the interpolation inaccuracy, it removes all the FLSs trespassing the territory of other classes where, the territory of each class is defined as the union of the territories of all samples belonging to the same class and the sample territory is defined as a hyper-sphere centered at the point under concern with radius equal to distance to the nearest neighbor from a different class. During classification, if the projection point is on the extrapolation segment, it is replaced by nearest point of the FLS.

Shortest feature line segment (SFLS) [9] avoids extrapolation inaccuracy by using FLSs as in RNFLS. It also avoids interpolation inaccuracy in some cases by choosing the shortest FLS which satisfies a specific geometric relation. The decision is made by finding the smallest hyper-sphere that contains the test sample. There is not a FLS deletion step during training.

In summary, efforts for improving the accuracy of NFL mainly focus on using a subset of FLSs either by permanently deleting or by disregarding those that do not satisfy pre-specified constraints. However, selection of subsets of FLSs is not done in a discriminative way. In other words, FLS subsets are not determined by directly taking into account the classification error.

1.2 Objectives

As described above some FLSs can cause interpolation inaccuracy. As an alternative approach to improve the performance to the NFL, editing can be applied to remove the feature lines leading to misclassification. In other words, the deletion of the FLSs can be one in a discriminative way. In fact, editing is extensively studied for improving the performance of *k*-NN classifier, especially in the case of outliers and noisy training data. Editing can be considered as selection of a subset of the training data which provides the highest classification accuracy on the training set. The idea of editing is proposed by Wilson where the edited nearest neighbor approach deletes the training samples whose label do not agree with its neighbors [15]. The idea is then extended into the multiedit algorithm by Devijver and Kittler which applies edited nearest neighbor algorithm in a repeated way [16]. The use of Genetic algorithms for this purpose is also widely considered [4, 17].

The major aim of this study is to propose an editing based selection of feature line segments to reduce the interpolation inaccuracy in NFL. The proposed method is based on the iterative evaluation of deleting FLSs in three steps namely error-based deletion, intersection-based deletion and pruning.

The error-based deletion step takes into account the classification accuracy on the training set in deciding to keep or delete a FLS. Score computation is firstly performed. For each segment, we calculate and record the number of correct and incorrect classification that it makes (negative and positive scores, respectively). Then, the sum of positive and negative scores is computed for each segment. The resultant scores are sorted in ascending order. The deletion of the top-rank segment is

investigated. If, by removing the corresponding segment, a better accuracy is achieved, it is permanently deleted. After deletion of a FLS, the scores are re-computed. This step is repeated until there is no more segment that needs to be deleted.

In the second step, the intersection of segments is investigated. If two segments from different classes intersect, the longer segment is removed. For multiple dimensional feature spaces, intersections of segments rarely occur. However they may be close to each other, still leading to interpolation inaccuracy. In multiple dimensional case, if the minimum distance between two FLSs is below a threshold, they are considered as intersecting segments and the longer is deleted.

As a last step, pruning is being applied. The aim of this step is to delete the FLSs that are very close to samples from a different class. More specifically, for a given training sample, if the nearest FLS belongs to a different class and it is closer than the nearest sample from the same class, the FLS is considered as a candidate for deletion. Although they are not making any misclassification in training phase, such FLSs have the risk to harm the model in the testing phase. Experiments on artificial data have shown that this improves the margin of the resultant decision boundary.

During testing, NFL is applied on the remaining FLSs. The proposed approach is evaluated on fifteen datasets, majority of which are from the UCI machine learning repository [18]. Experimental results have shown that the proposed approach provides better accuracies compared to NFL, RNFLS and SFLS on 14, 11 and 12 datasets, respectively.

1.3 Layout of the Thesis

The rest of the thesis is organized as follows. Chapter 2 presents a brief literature review. The proposed method is presented in Chapter 3. Chapter 4 presents the experimental results on three artificial and fifteen real datasets. Chapter 5 lists the conclusion drawn from this study.

Chapter 2

LITERATURE REVIEW

2.1 The Nearest Neighbor Approach (NN)

The Nearest Neighbor approach which was proposed in 1967 labels an unseen query sample as the same label of the nearest training sample [19]. As a non-parametric rule it is the simplest yet effective and popular method. Despite its simplicity, it has several advantages. For example, it can learn from a small set of samples, there is no pre-processing task, new information can be added at runtime and may give competitive performance with many other advanced classification techniques [20].

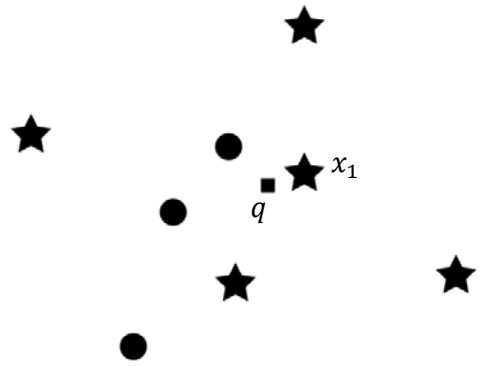

Figure 2: An illustration for the operation of the NN rule.

Consider the query point q given in Figure 2 where there are two different classes. For the given query point, the nearest training sample belongs to class '★'. Hence, it is similarly labeled as '★'. Since the NN rule utilizes only the label of the nearest

neighbor, the remaining training samples are ignored. In case of noisy training data, this method may lead to large number misclassifications.

An extension to the NN rule is the *k*-NN approach. In this method, larger numbers of neighbors (k) are considered where voting over the labels of the k nearest samples is performed to compute the most likely class. The most common distance measure used to find the nearest samples is the Euclidean distance [7]. A major disadvantage of the NN and *k*-NN methods is the time complexity of making predictions when compared to many other methods.

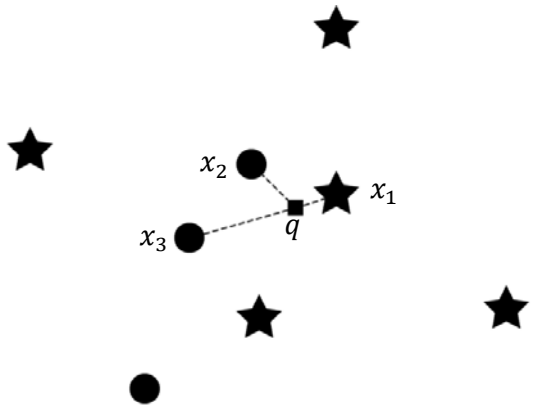

Figure 3: The *k*-NN approach considers a wider neighborhood.

In Figure 3 let $k = 3$. The nearest three samples for the query point q are $x_1, x_2,$ and x_3. By applying voting, q is labeled as the class represented by "●".

Similar to NN, the classification performance of *k*-NN increases as the number of training samples increases.

2.2 Nearest Feature Line (NFL) Method

The objective of the nearest feature line method which was originally proposed for face recognition is to generalize the representational capacity of data samples using

lines passing through each pair of samples belonging to the same class [5]. This technique is expected to be superior to NN especially in cases where the training data is limited.

The NFL approach is a two-step scheme. The first step corresponds to the construction of feature lines (FL). In the second step, the query point is projected to all FLs and the distances from the projection points to the query point are computed. During classification; the class to which the nearest line belongs is selected as the label of the query point.

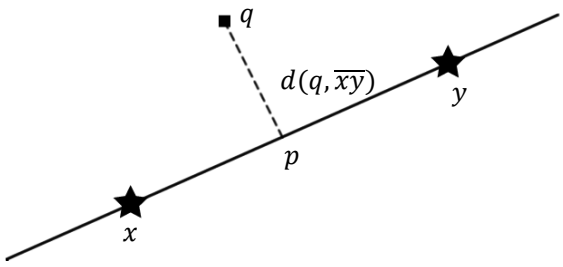

Figure 4: Classification using the NFL method in a subspace represented by FLs passing through each pair of samples within the same class.

Let $\overline{xy}$ denote the FL passing through x and y as shown in Figure 4. Let p denote the projection point of q on $\overline{xy}$ which can be computed as

$$p = x + \mu(y - x)$$

where μ is the position parameter that is defined as

$$\mu = \frac{(q - x).(y - x)}{\|y - x\|^2}$$

The symbol '.' represents the dot product. The parameter μ describes the position of p relative to x and y. When $\mu < 0$, p is on backward extrapolation part of $\overline{xy}$. When

$\mu > 1$, p is on forward extrapolation part of $\overline{xy}$ and p is on interpolation part if $0 < \mu < 1$. When $\mu = 0$, p is on x and $\mu = 1$ means that p is on y as illustrated in Figure 5.

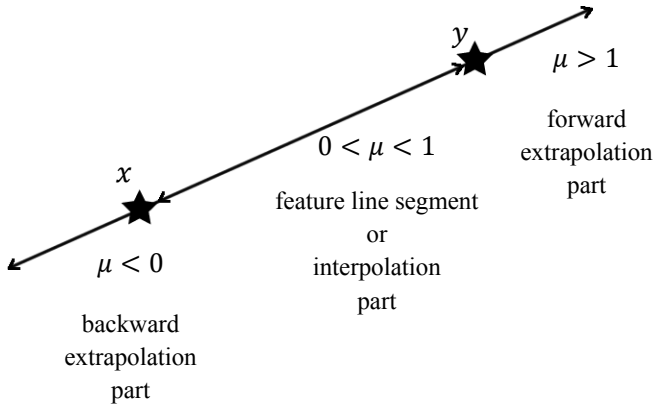

Figure 5: The position parameter values.

The distance from the query point to the FL is defined as

$$d(q, \overline{xy}) = \|q - p\|$$

where $\|.\|$ denotes the Euclidean distance. Assuming that p_i and q_i represent the *i*th entries in the corresponding vectors and D is the vector dimensionality, d is computed as

$$d = \|q - p\| = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots} = \sqrt{\sum_{i=1}^{D} |q_i - p_i|^2}$$

Let N_c denote the number of samples that belong to class c where there are C classes. In this case, the total number of FLs can be calculated as

$$N_L = \sum_{c=1}^{C} \frac{N_c(N_c-1)}{2}.$$

It is obvious that the number of FLs grows fast as the number of training samples increases. Hence, NFL is computationally more demanding than NN.

Although the NFL method is successful in improving the classification ability of the NN approach, there is room for further improvements [12]. It has two main sources of errors, namely the interpolation and extrapolation inaccuracies. The extrapolation inaccuracy mainly occurs in a low dimensional feature space when a sample pair is far away from the query point [14]. An example is presented in Figure 6. The query point q belongs to the class "★", but is classified to class "●" although x_1 and x_2 are far away. This error in caused by the backward extrapolation part of the FL $\overline{x_1 x_2}$ that belongs to the class denoted by "●".

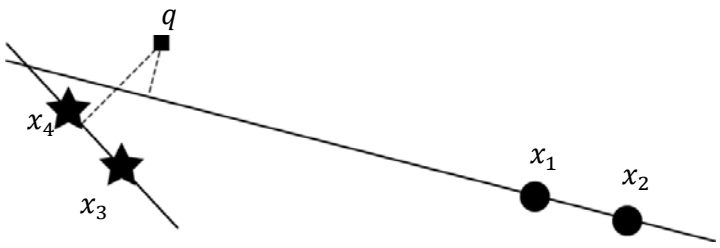

Figure 6: Extrapolation inaccuracy in NFL.

The interpolation inaccuracy occurs when a FL passes through samples that are away from each other and trespasses a cluster of a different class. Interpolation inaccuracy creates inconsistency in classification decision. Consider the example presented in Figure 7. q is misclassified as class "●" although it belongs to the class represented by "★".

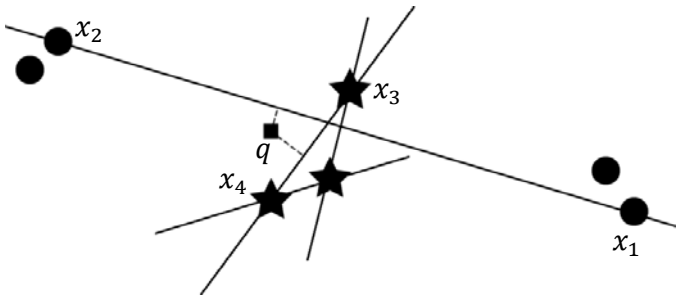

Figure 7: Interpolation inaccuracy in NFL.

In order to avoid the above-mentioned weaknesses, some extensions of NFL are proposed. Two most widely known schemes are the rectified nearest feature line segment and the shortest feature line segment.

2.3 Rectified Nearest Feature Line Segment (RNFLS)

In RNFLS, both extrapolation and interpolation inaccuracies are suppressed [14]. The first step of RNFLS is to define a subspace named as nearest feature line segment subspace (NFLS-subspace). This subspace is defined as the union of FL segments (FLS) where the forward and backward extrapolation parts are discarded. During testing, in order to implement this, RNFLS firstly finds the projection point on all FLs. If, for a particular FL, the projection point is on either of the extrapolation parts, the nearest endpoint is chosen to be the projection point for calculating the FL distance. When the projection point is on the interpolation part, that point is considered in the distance computation as in the NFL method. Consider the example presented in Figure 8. The projection of q_2 is in the forward extrapolation part of $\overline{x_1 x_2}$. Hence, the nearest sample, i.e. x_2 is considered instead of p_2. Consequently, since no extrapolation segments are used, there will be no extrapolation inaccuracy.

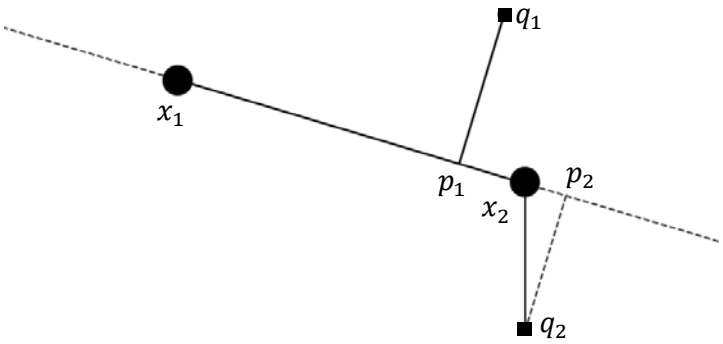

Figure 8: NFLS subspace used by RNFLS for avoiding extrapolation inaccuracy.

The NFLS-subspace denoted by $\tilde{S}$ is the set of line segments which passes through each pair of samples of the same class. The NFLS-subspace for class c can be represented as

$$\tilde{S}_c = \left\{ \left(\widetilde{x_i\, x_j} \right) \middle| 1 \le i, j \le N_c, x_i \in c, x_j \in c, i \ne j \right\}$$

where x_i and x_j are samples belonging to class c, $\widetilde{x_i\, x_j}$ is the line segment connecting x_i and x_j, and N_c is the number of samples that belong to class *c*.

During testing, the distance from a query point q to the NFLS-subspace is calculated as

$$d\left(q, \tilde{S}_c\right) = \min_{y \in \tilde{S}_c} \|q - y\|,$$

where *y* depends on the position parameter, μ. For a particular FLS $\widetilde{x_i\, x_j}$, if $0 < \mu < 1$, since the projection point is between x_i and x_j, $d\left(q, \tilde{S}_c\right) = \|q - p\|$. On the other hand, $d\left(q, \tilde{S}_c\right) = \|q - x_i\|$ when $\mu < 0$ (backward extrapolation part) and $d\left(q, \tilde{S}_c\right) = \left\|q - x_j\right\|$ when $\mu > 1$ (forward extrapolation part).

In order to avoid the interpolation inaccuracy, RNFLS deletes the FLSs trespassing the other classes. The resultant subspace is named as the rectified nearest feature line segment subspace (RNFLS-subspace). In order to compute the trespassing segments, sample and class territories are firstly defined. The sample territory is defined as the hyper-sphere whose radius is equal to the distance from the sample and its nearest neighbor from a different class. Assume that x_i belongs to class c_i and x_k belongs to a different class, c_k. The radius, r_{x_i} of the sample territory T_{x_i} is defined as

$$r_{x_i} = \min_{\forall x_k, c_k \neq c_i} \|x_i - x_k\|.$$

Hence,

$$T_{x_i} = \{x \in \mathbb{R}^D | \ \|x - x_i\| < r_{x_i}\}.$$

The territory of class c_i is defined as the union of all sample territories belonging to the same class as

$$T_{c_i} = \bigcup_{x_i \in c_i} T_{x_i}$$

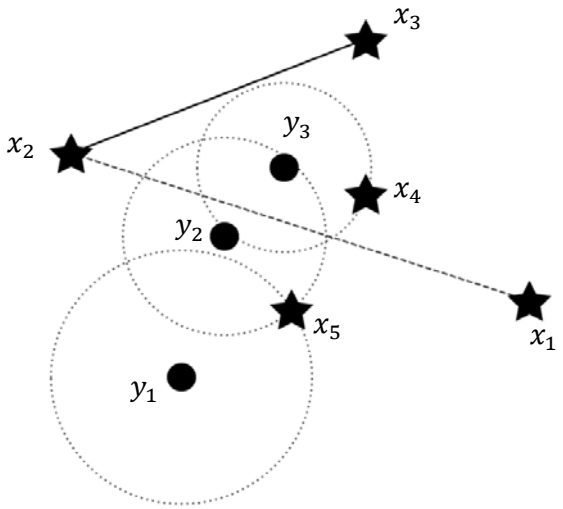

Figure 9: Territories of the samples are shown by dotted lines whose union constitutes the class territory. The segment $\widetilde{x_1x_2}$ is removed because it trepasses the territory of the other class.

Computation of the rectified space is illustrated in Figure 9. The sample territories of the class represented by "●" are shown by circles. The territory of the class represented by "●", $T_{\bullet}$ is obtained as the union of all three circles. The FLS $\widetilde{x_1x_2}$ is trespassing $T_{\bullet}$ and hence it is deleted.

Let $\widetilde{U}_{c_i}$ denote the set of FLSs that belong to the class c_i which trespass other class(es). The RNFLS-subspace of c_i is defined as

$$\tilde{S}^*_{c_i} = \tilde{S}_{c_i} \setminus \widetilde{U}_{c_i}$$

where

$$\widetilde{U}_{c_i} = \left\{ (\widetilde{x_m x_n}) \middle| \exists c_y, c_i \neq c_y \wedge \widetilde{x_m x_n} \in \tilde{S}_{c_i} \wedge \widetilde{x_m x_n} \cap T_{c_y} \neq \emptyset \right\}$$

and '\' is set difference operator.

It should be noted that, as seen in Figure 9, $\widetilde{x_1x_2} \cap T_{\bullet} \neq \emptyset$. Hence, $\widetilde{x_1x_2} \notin \tilde{S}_{\star}^{*}$. On the other hand, $\widetilde{x_2x_3} \in \tilde{S}_{\star}^{*}$ since $\widetilde{x_2x_3} \cap T_{\bullet} = \emptyset$.

Classification in RNFLS-subspace is similar to the NFLS-subspace. However, in this step, $\tilde{S}_{c_i}^{*}$ that is the set of remaining segments are employed during the classification.

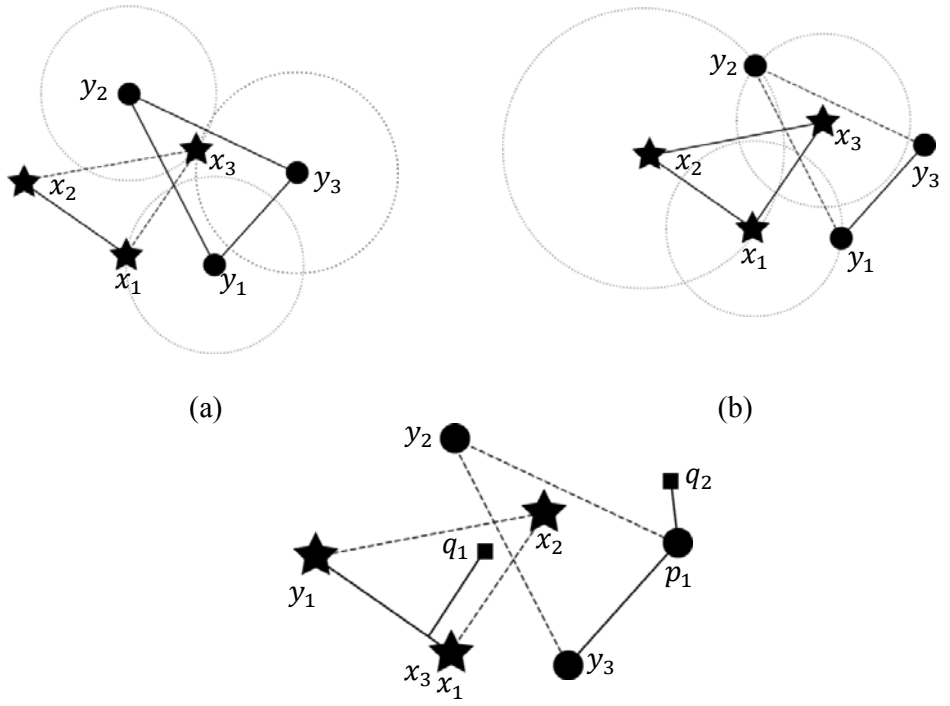

Figure 10: Classification using the RNFLS-subspace.

Figure 10 illustrates classification using the RNFLS approach. Part (a) shows the sample territories of y_1, y_2, and y_3 using dotted circles. Segments $\widetilde{x_1x_3}$ and $\widetilde{x_2x_3}$ are deleted. Part (b) shows the sample territories of x_1, x_2, and x_3. Segments $\widetilde{y_1y_2}$ and $\widetilde{y_2y_3}$ are deleted. In part (c), the projection of the query point q_1 is on the interpolation part of the segment $\widetilde{x_1x_2}$. Thus, $d(q, \widetilde{x_1x_2}) = d(q, p_1)$. For the query

point q_2, the projection point is on the forward extrapolation part of the line segment $\widehat{y_1y_3}$. Therefore, $d(q, \widehat{y_1y_3}) = d(q, y_3)$.

2.4 Shortest Feature Line Segment (SFLS)

As an alternative approach to overcome the inaccuracies of NFL, Han et al. [9] recently proposed the shortest feature line segment technique. SFLS aims to find the shortest FLS considering the geometric relation constraints between the query point and FLSs instead of calculating the FL distances. This approach does not have any pre-processing step.

During classification, hyper-spheres that are centered at the midpoints of all FLSs are considered where the length of a given segment is equal to the diameter of the corresponding hyper-sphere. SFLS finds the smallest hyper-sphere which contains the query point (inside or on that hyper-sphere). For a given test sample, all FLSs for which the query point is inside or on the coresponding hyperspheres are firstly tagged. Then, the shortest tagged FLS is found. The class that the corresponding segment belongs is computed as most likely. It should be noted that, as in RNFLS, there will be no extrapolation inaccuracy problem since segments are used.

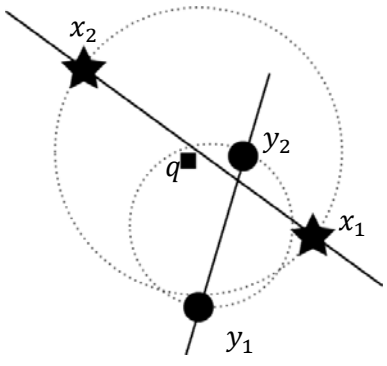

Figure 11: Classification of q in SFLS.

In the examplar case presented in Figure 11, the query point q is labeled as the class represented by "●" because the smallest hypersphere that contains this point is formed by a FLS from that class.

In order to determine whether a given test sample q is contained by a hypersphere formed by x_i and x_j, the angle α between $\widehat{q\tilde{x}_i}$ and $\widehat{q\tilde{x}_j}$ which is defined as

$$\alpha = \frac{180}{\pi}.\frac{\arccos\left((q-x_i)^T(q-x_j)\right)}{\|\widehat{q\tilde{x}_i}\|.\|\widehat{q\tilde{x}_j}\|}$$

is firstly computed. If $0 \leq \alpha < 90$, the feature line is not tagged because the query point is not inside or on the hypersphere. On the other hand, if $90 \leq \alpha \leq 180$, the FLS is tagged as a candidate because the geometric contraint is satisfied. Figure 12 illustrates three possible cases. In part (a), $\alpha < 90$ and hence $\widehat{x_i\tilde{x}_j}$ is not tagged. In parts (b) and (c), $\widehat{x_i\tilde{x}_j}$s are tagged.

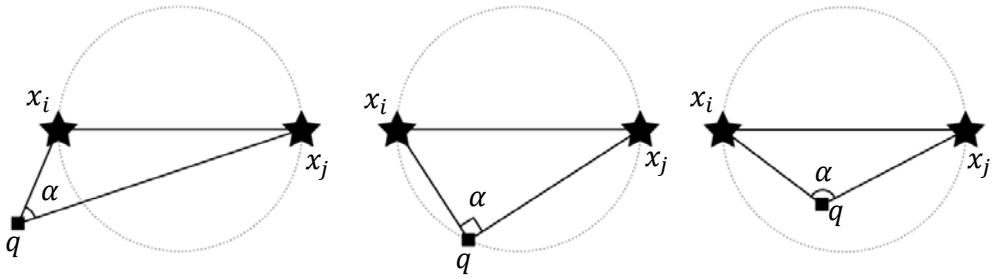

Figure 12: Geometric relation between the query point and FL segment.

In some cases, there may not be any tagged segment for the query sample. The corresponding query point is either rejected or the nearest neighbor method is applied to make the final decision.

2.5 Comparing NFL, RFLS, and SFLS

NFL is originally proposed to counteract the major weakness in the NN method which is its high error rate in cases where small number of training samples exist. It has two drawbacks, namely the interpolation and extrapolation inaccuracies. RNFLS can counteract the two inaccuracies existing in the NFL method leading to better classification performance. The compuaional complexity is also reduced due to deleting some segments. However, the order of reduction is problem dependent. The computational complexity of SFLS is also less than NFL [9]. SFLS supresses the extrapolation inaccuracy. However, it is able to counteract the interpolation inaccuracy only in some cases.

Chapter 3

EDITED NEAREST FEATURE LINE APPROACH

As mentioned in Chapter 1, editing corresponds to removing some prototypes from the training data. The main idea in the edited NFL (eNFL) is to delete the feature line segments that lead to interpolation inaccuracies. The approach consists of three major steps, namely error-based deletion, intersection-based deletion and pruning. In each step, some segments are iteratively removed from the training data by considering several criteria. At the end of the iterations, a subset of the feature line segments are preserved which form a reduced subspace for each class.

It should be noted that, since the proposed approach employs only segments as in RNFLS and SFLS techniques presented in Chapter 2, the extrapolation inaccuracy does not occur.

3.1 Error-based FLS Deletion

The main idea is that the FLSs obtained using samples that are far away from each other are mainly expected to contribute more to the misclassification rate than correct classification and hence they should be deleted. The first step of eNFL involves ranking all FLSs by taking into account the number of correct classifications and misclassifications they participate. In other words, the benefit of employing each individual FLS is investigated. This is done by taking each sample out of the training set one by one to be utilized as a query point and recording the nearest FLS. Then,

the numbers of times each FLS participates in correct classification and misclassification are computed. The decision about deletion is based on these scores.

As an example, assume that there are four training samples from class '★' and five from class '●' as shown in part (a) of Figure 13. Let us take x_7 out of the training set and assume that it is a query point. The nearest FLS to x_7 is $\widetilde{x_3x_4}$. Although x_7 belongs to class '●', because of $\widetilde{x_3x_4}$, it is labeled as class '★'. This means that $\widetilde{x_3x_4}$ leads to a misclassification. By removing $\widetilde{x_3x_4}$, the query point x_7 will be classified correctly as '●' since $\widetilde{x_5x_8}$ will be computed as the nearest FLS in this case. However, by removing a FLS, the benefits obtained by correcting some misclassification may be lost due to new misclassifications. For instance, although deleting $\widetilde{x_3x_4}$ leads to a correct decision for x_7, two new misclassifications occurs. In order to clarify this, consider the case presented in part (b) where x_9 is left out of the training data. In this case, due to deleting $\widetilde{x_3x_4}$ that is the nearest FLS for that sample, it is misclassified since $\widetilde{x_5x_8}$ is now the nearest FLS. Assume that we similarly take x_6 out of the training data as illustrated in part (c). In this case, due to deleting $\widetilde{x_3x_4}$, this sample is also misclassified since $\widetilde{x_5x_8}$ is again the nearest FLS. Consequently, before removing $\widetilde{x_3x_4}$, there was one misclassification and after removing it, two misclassifications occurred. Hence, removing this FLS may not be a good idea. The decision to delete a segment or not should be made after taking into account the new labels generated by the remaining FLSs for the training samples for which the corresponding FLS used to be the nearest before its deletion.

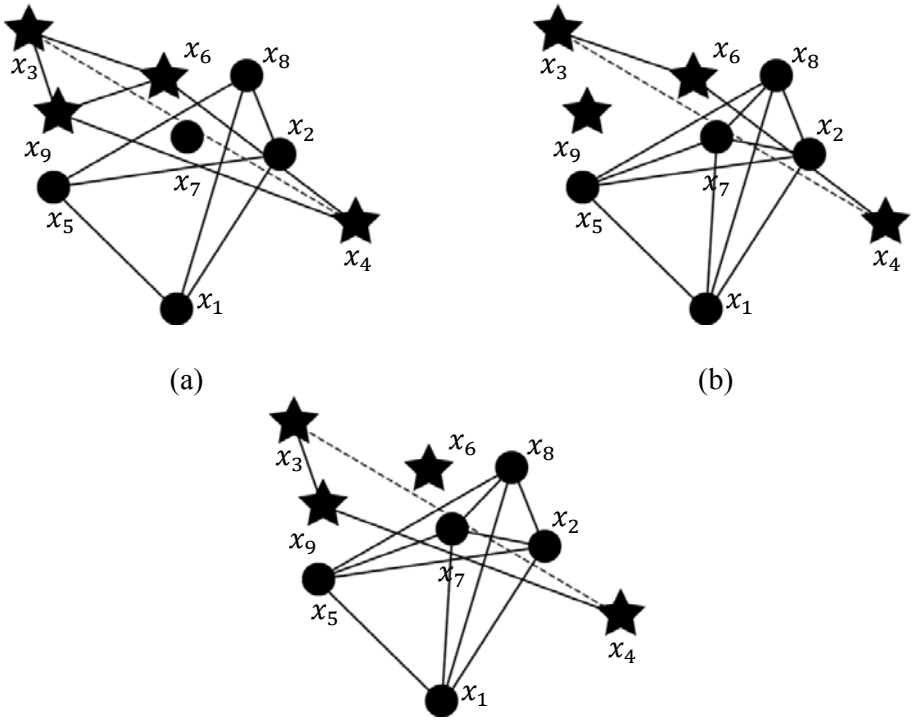

Figure 13: Choosing different samples for the evaluation of nearest FLSs. The samples x_7, x_9 and x_6 are taken out in parts (a), (b) and (c) respectively.

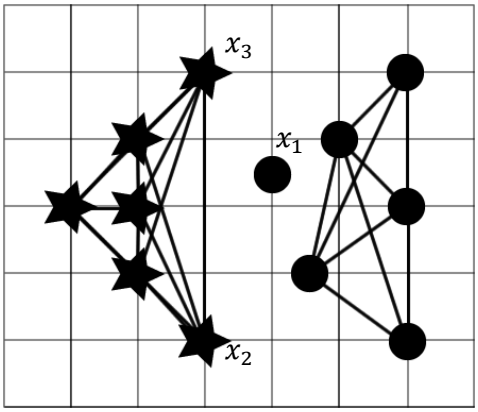

Figure 14: An example where a FLS can be deleted, leading to a decrease in the error rate.

As another example, consider the scatter plot presented Figure 14. Let us take x_1 out of the training set and assume that it is a query point. The nearest FLS to x_1 is $\widetilde{x_2x_3}$. which belongs to the other class. Deleting this FLS will lead to a correct

classification for x_1. It can be seen that, due to this deletion, new misclassifications are not generated. Hence, removing this FLS should be taken into consideration.

In order to determine the FLSs to be deleted, this step firstly records the number of samples which each FLS leads to correct or incorrect decision as *positive* or *negative scores*, respectively. Positive score of each segment $\rho_+(\widetilde{x_i\,x_j})$ denotes the number of correctly classified samples where $\widetilde{x_i\,x_j}$ is computed as the nearest FLS and the negative score, $\rho_-(\widetilde{x_i\,x_j})$ denotes the number of misclassified samples where $\widetilde{x_i\,x_j}$ is computed as the nearest FLS. For the example presented in Figure 13, $\rho_+(\widetilde{x_3x_4}) = 2$ and $\rho_-(\widetilde{x_3x_4}) = 1$. The *total score* of a segment is defined as

$$\rho(\widetilde{x_ix_j}) = \rho_+(\widetilde{x_ix_j}) - \rho_-(\widetilde{x_ix_j}),\ \ i \neq j,\ \ x_i, x_j \in c.$$

Hence, we obtain $\rho(\widetilde{x_3x_4})$ as

$$\rho(\widetilde{x_3x_4}) = 2 - 1 = 1.$$

When $\rho > 0$, the accuracy is expected to decrease if the segment is deleted. However, if $\rho < 0$, the segment should be considered as a candidate to be removed.

Let $\mathcal{H}_{\widetilde{x_ix_j}}$ denote the *relevant set* of the FLS $\widetilde{x_ix_j}$, which is defined as the set of training samples for which $\widetilde{x_ix_j}$ is the nearest. This set may be empty for some segments which means that they are not used for any of the samples.

The pseudo code of this step is as follow:

$S = \{x_i\}_{i=1}^{N}$: set of all samples
$F = \{\widetilde{x_i x_j}\}_{x_i, x_j \in C_k}$: set of all FLSs
for n=1 to N
$F' = F \setminus \{\widetilde{x_n x_j}\}_{j \neq n}$
$\widetilde{x_p x_q} = \arg\min_{\widetilde{x_i x_j}} d(x_n, \widetilde{x_i x_j})$
if label ($\widetilde{x_p x_q}$) = label(x_n)
$\rho^+(\widetilde{x_p x_q}) = \rho^+(\widetilde{x_p x_q}) + 1$
else
$\rho^-(\widetilde{x_p x_q}) = \rho^-(\widetilde{x_p x_q}) + 1$
end
end

After the ρ values are computed for all FLS, they are sorted in ascending order and the following procedure is applied to the FLSs starting from the top to determine the segments to be deleted. The main idea is to take into consideration the performance of the remaining FLSs on $\mathcal{H}_{\widetilde{x_i x_j}}$ for making the final decision. The *updated score* of a FLS, $\rho^*(\widetilde{x_i x_j})$ is firstly calculated as the number of samples in $\mathcal{H}_{\widetilde{x_i x_j}}$ that are correctly classified by the remaining FLSs after $\widetilde{x_i x_j}$ is deleted. Then, if $\rho^*(\widetilde{x_i x_j}) \geq \rho(\widetilde{x_i x_j})$, it means that deleting segment $\widetilde{x_i x_j}$ will contribute to correct classification and hence it is deleted. The deletion is also done in the case of equality since keeping the FLS does not contribute to the classification accuracy. After a FLS is deleted, the scores are re-computed for all remaining FLSs and ranking is updated. The procedure described above is repeated until $\rho^*(\widetilde{x_i x_j}) < \rho(\widetilde{x_i x_j})$ for the top ranked FLS.

It should be noted that this step is mainly useful for removing misleading FLSs that are located close to the nonlinear decision boundaries and are formed using samples that are away from each other. Figure 14 is an example for this case.

In the example presented in Figure 13, the feature line $\widetilde{x_3x_4}$ is found to be useful. However, it is clearly seen that it leads to interpolation inaccuracy. In other words, it is trespassing the region of another class. In fact, deletion of such lines should be reconsidered by employing an alternative criterion which is done in the intersection-based deletion step described below.

3.2 Intersection-based Deletion

In a 2-dimensional space, the interpolation inaccuracy can be easily detected by computing the intersecting feature line segments. In this study, the intersection-based deletion step is applied for this purpose. The main idea is to delete the longer segment in the case of an intersecting pair of segments. As an example, consider Figure 15, where the FLS $\widetilde{x_1x_2}$ has an intersection point o with $\widetilde{x_3x_4}$. Since $\widetilde{x_3x_4}$ is longer, i.e. $\|x_4 - x_3\| > \|x_2 - x_1\|$, $\widetilde{x_3x_4}$ is deleted. The main logic behind deleting longer segments is the fact that interpolation inaccuracy is generally caused due to the segments through the samples that are far away from each other. If the length of both segments is exactly same, the segment to be deleted is randomly selected.

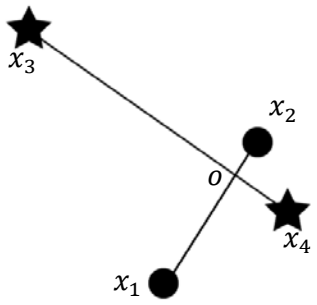

Figure 15: Two FLSs that intersect with each other.

In higher dimensional space, intersection of two FLSs is less likely to occur. However, in some regions of the feature space, it is possible that they may be very

close to each other, still leading to the interpolation inaccuracy. In multiple dimensional case, if the minimum distance between two FLSs is below a threshold, they are considered as intersecting segments and the longer is deleted.

In order to implement this rule, the threshold should be defined. Intuitively, when the segments are too short, the threshold should be too small. The threshold should be larger for longer FLSs. This is analogous to considering a hyper-sphere in shortest feature line segment approach. Remember that a FLS is tagged only if the query point is within the corresponding hyper-sphere that has the radius defined as the half of the segment length.

In this thesis, we studied two strategies for setting the threshold, τ. The first strategy is to assign a fixed value. The value of the threshold may be optimally estimated for each dataset.

As an alternative approach, for a given FLS denoted by $\widetilde{x_i x_j}$, we can consider hyper-cylinders having the base radius defined as

$$r_{\widetilde{x_i x_j}} = \frac{\|x_i - x_j\|}{\beta}$$

and β is a design parameter. The base radius is proportional with the length of the segment. Then, two segments are defined to be intersecting if the distance between the FLSs is less than the base radius of the hyper-cylinder defined for the shorter FLS. More specifically, the segments $\widetilde{x_1 x_2}$ and $\widetilde{x_3 x_4}$ are assumed to intersect if

$$d(\widehat{x_1x_2}, \widehat{x_3x_4}) < \min\left(r_{\widehat{x_1x_2}}, r_{\widehat{x_3x_4}}\right).$$

Hence, $\tau = \min\left(r_{\widehat{x_1x_2}}, r_{\widehat{x_3x_4}}\right)$. This means that the minimum distance between the FLSs should be smaller than the base radius of the thinner hyper-cylinder. In other words, the whole cross-section of the thinner hyper-cylinder should be completely within the thicker one and it should include the longer FLS in the region around the minimum distance. Figure 16 presents an illustration for the proposed scheme. Two exemplar segments are given with the corresponding hyper-cylinders as shown on the left. The segments, $\widehat{x_1x_2}$ and $\widehat{x_3x_4}$ are assumed to be intersecting if the hyper-cylinder corresponding to $\widehat{x_1x_2}$ is passing through the hyper-cylinder corresponding to $\widehat{x_3x_4}$ and the distance between $\widehat{x_1x_2}$ and $\widehat{x_3x_4}$ is less than the base radius of the hyper-cylinder corresponding to $\widehat{x_1x_2}$. On the right, three possible cross-section views are presented. The two segments are intersecting only in the case (a). The computation of the smallest distance is presented in the Appendix.

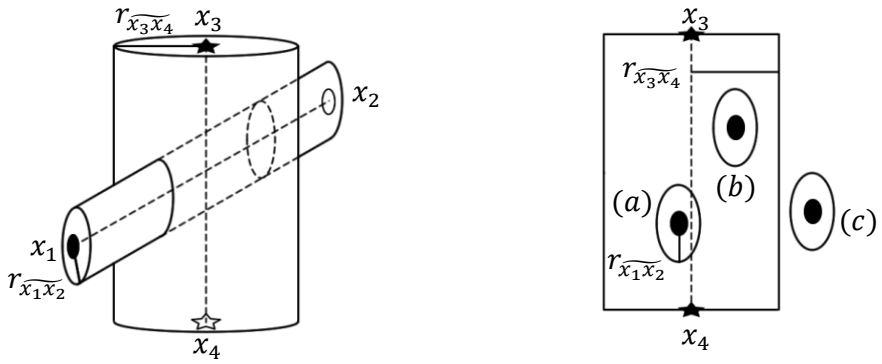

Figure 16: An illustration for the cylinder based distance model.

The design parameter, β controls the number of deleted segments. A larger β leads to smaller radiuses and hence smaller number of deletions. For a classification problem where distances between training samples is high, a larger value of β should

be used to avoid large number of deletions. When the training samples are very close, a small number should be chosen for β to enforce some deletions. Thus, the value of β is depending on the distribution of samples in the feature space. In this study, we studied different settings and also an exhaustive search method to select the best fitting $\beta \in \{2,3,4,5\}$ using 3-fold cross-validation.

The pseudo code of this step is as follow:

$F = \{\widetilde{x_i x_j}\}_{x_i, x_j \in C_k}$: set of all FLSs remaining after Error-based deletion
Let $|F| = K$
Let $\tilde{f}_k$ denote k^{th} FLS in F
for k = 1 to K
    for m=k+1 : K
        if $d(\tilde{f}_k, \tilde{f}_m) \leq \tau$
            if $\|\tilde{f}_k\| > \|\tilde{f}_m\|$
                delete $\tilde{f}_k$
            else
                delete $\tilde{f}_m$
            end
        end
    end
end

3.3 Pruning

Majority of the FLSs leading to interpolation inaccuracy are expected to be deleted in the first two steps described above. However, the FLSs that are located near the decision boundary where overlaps among different classes occur are generally retained. As it will be verified by the simulations presented in next chapter, a small percentage of the FLSs are deleted in the first step which means that the FLSs that are close to the boundary may contribute to the misclassification rate during testing. In the pruning step, the FLSs that are very close to samples from a different class are deleted. More specifically, for a given training sample, if the nearest FLS belongs to

a different class and it is closer than the nearest sample from the same class, the FLS is a candidate for deletion.

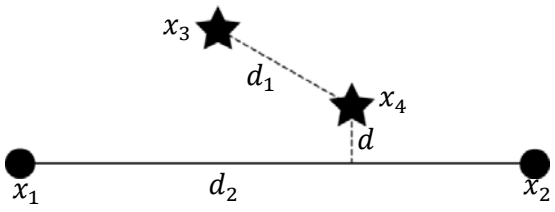

Figure 17: An exemplar case to describe pruning step.

Consider the exemplar case presented in Figure 17. Let $\widetilde{x_1x_2}$ be a FLS that is not deleted by any of the first two steps. Consider the sample x_4. Let d_1 denote the distance to the nearest sample from same class, d_2 denote the length of $\widetilde{x_1x_2}$ and d denote distance to nearest FLS from any of the other classes. Segment $\widetilde{x_1x_2}$ should be deleted if $d < d_1$ and $d_2 > d_1$. It means that a FLS is removed if it is closer to a training sample from another class than its nearest neighbor from the same class and its length is longer than this distance. The pseudo code of this step is as follow:

$F = \{\widetilde{x_ix_j}\}_{x_i,x_j \in C_k}$: set of all FLSs remaining after Intersection-based deletion
Let $|F| = K$
Let $\tilde{f}_k$ denote k^{th} FLS in F
Let $x_n \in C_n$

for n = 1 to N
    $\tilde{f}_k = \arg\min_{\tilde{f}_i} d(x_n, \tilde{f}_i), \tilde{f}_i \notin C_n$
    $x_k = \arg\min_{x_i} d(x_n, x_i)$ where $x_n, x_i \in C_n$
    if $d(x_n, \tilde{f}_k) < \|\widetilde{x_nx_k}\|$ && $\|\widetilde{x_nx_k}\| < \|\tilde{f}_k\|$
        delete $\tilde{f}_k$
    end
end

After the application of these steps, the FLSs retained are used during testing. The effect of each step is studied by considering three artificial datasets. The following chapter firstly presents the simulations on artificial data and then on fifteen real datasets.

Chapter 4

EXPERIMENTAL RESULTS

4.1 Experiments on Artificial Data

In order to evaluate the proposed scheme, three 2-D artificial datasets are employed. These datasets are two-spirals, rings, and cone-torus. Two-spirals dataset contains two spirals generated as follows:

$$spiral\ 1: \begin{cases} Feature\ 1 = \theta \cos(\theta) \\ Feature\ 2 = \theta \sin(\theta) \end{cases}$$

$$spiral\ 2: \begin{cases} Feature\ 1 = \theta \cos(\theta + \pi) \\ Feature\ 2 = \theta \sin(\theta + \pi) \end{cases}$$

Figure 18 shows two hundred samples generated by equal increments of θ from $\pi/2$ to 3π and then polluted by zero-mean Gaussian noise with standard deviation 0.5. The horizontal and vertical axes correspond to two different features.

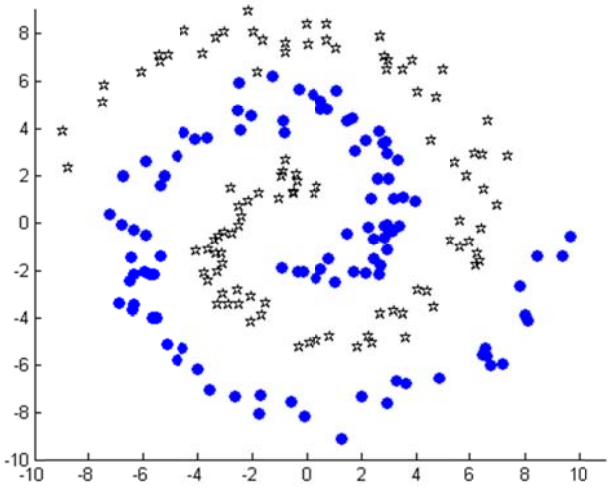

Figure 18: Scatter plot for the two-spirals dataset.

Rings data has two classes and contains two hundred samples that are generated as follows:

$$Inner\ ring: \begin{cases} Feature\ 1 = \cos(\theta) \\ Feature\ 2 = \sin(\theta) \end{cases}$$

$$Outer\ ring: \begin{cases} Feature\ 1 = 2cos(\theta) \\ Feature\ 2 = 2\sin(\theta) \end{cases}$$

The data are created by increasing the value of θ from 0 to 2π in equal steps. The data are then polluted by Gaussian noise whose mean is zero and standard deviation is 0.1. Rings dataset is plotted in Figure 19.

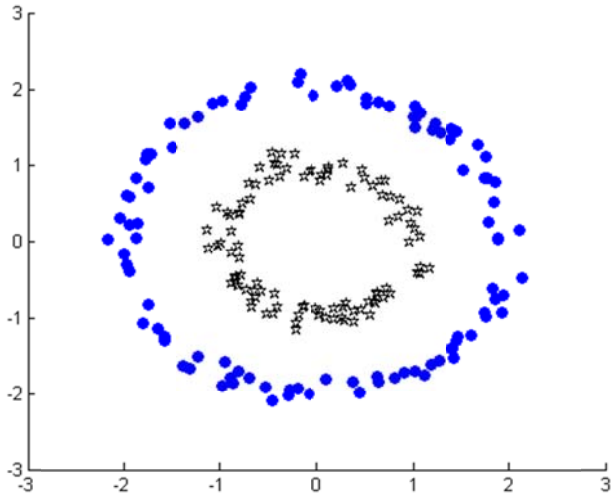

Figure 19: Scatter plot for the rings dataset.

Cone-torus data contains eight hundred samples in three classes. The scatter plot is presented in illustrated in Figure 20.

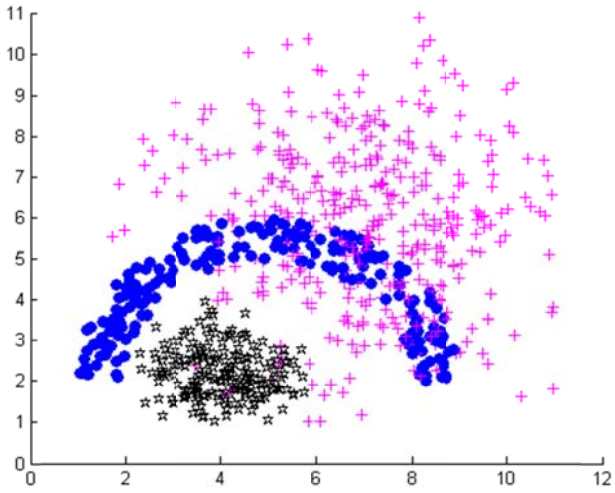

Figure 20: Scatter plot for the cone-torus dataset.

Each dataset is randomly divided into two parts. The first part is used for training and the other for testing.

Using the training data, the feature lines obtained for the class '★' are illustrated in Figure 21, Figure 22, and Figure 23 for the datasets considered. It is obvious that classification errors using NFL should be expected to be very high in all three datasets since the FLs are overlapping with the samples of the other class(es). Our simulation studies show that the classification error is 38.00% in rings, 43.00% in two-spirals, and 44.25% in cone-torus dataset when the test data are considered.

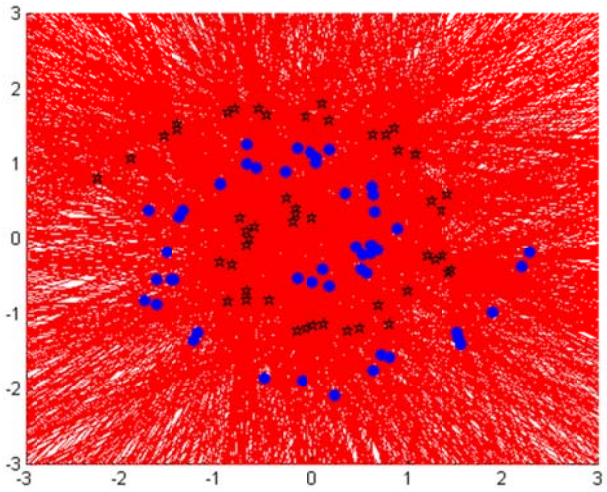

Figure 21: NFL feature space for class '★' in the two-spirals dataset.

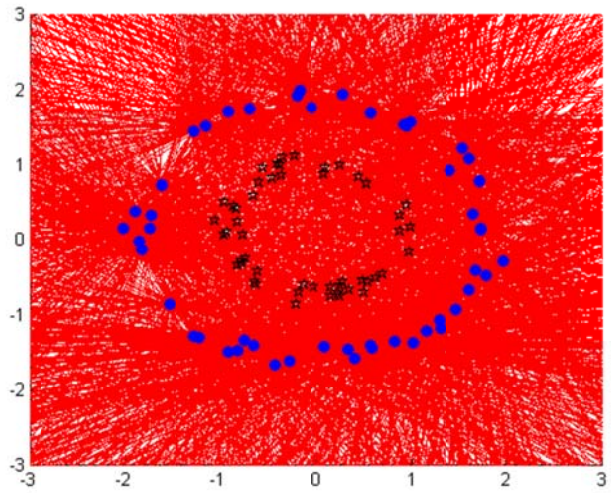

Figure 22: NFL feature space for class '★' in the rings dataset.

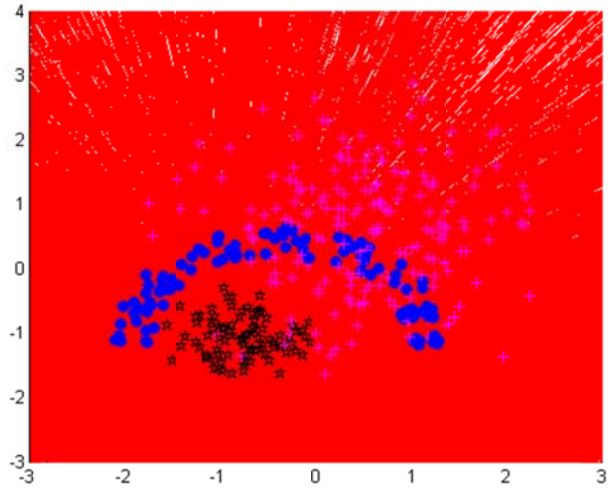

Figure 23: NFL feature space for class '★' in the cone-torus dataset.

Figure 24, Figure 25, and Figure 26 present the NFLS feature space respectively for two-spirals, rings and cone-torus datasets for different classes. It can be seen in the figures that the error rates should be due to the interpolation inaccuracy.

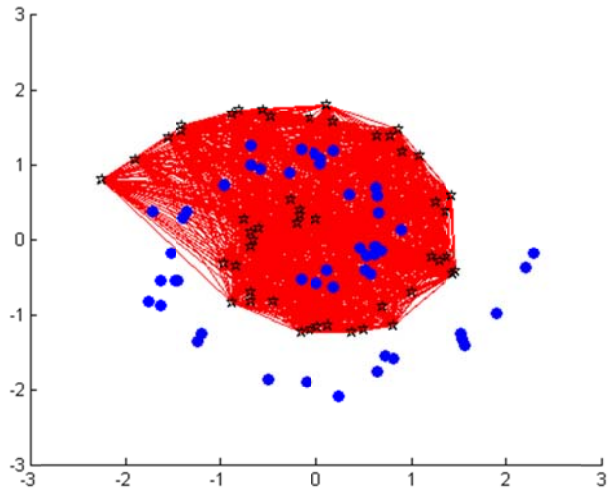

Figure 24: NFL segments for class '★' of the two-spirals dataset.

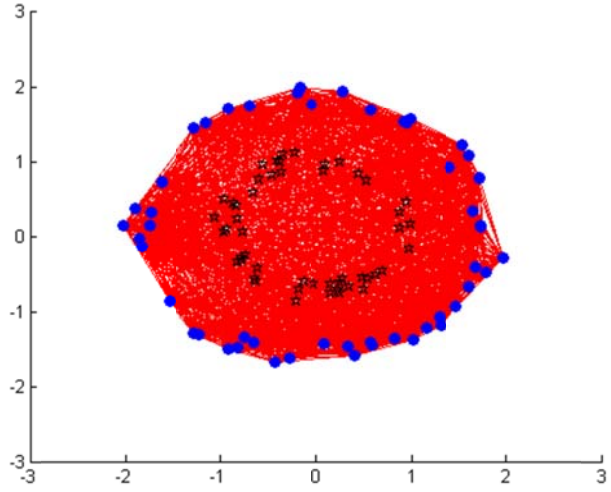

Figure 25: NFL segments for class '●' of the rings dataset.

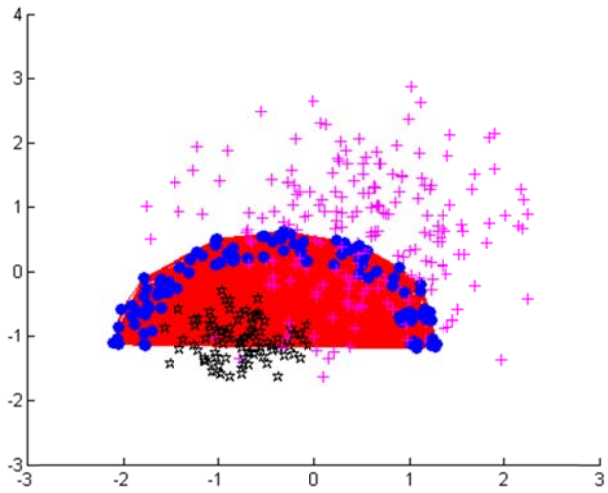

Figure 26: NFL segments for class '●' of the cone-torus dataset.

Our simulation studies show that the classification errors are reduced from 38.00% to 23.00% in rings, from 43.00% to 32.00% in two-spirals, and from 55.75% to 22.55% in cone-torus dataset when the test data are considered. It can be concluded that, avoiding the extrapolation inaccuracy by using FLSs instead of feature lines, the error rates can be significantly reduced.

By applying the error-based deletion step, the number of segments deleted in two-spirals, rings and cone-torus datasets are 25, 28, and 102 respectively. Compared to total number of segments (2450, 2450, and 30773) these numbers can be considered as small. However, the number of deletions are observed to be much larger in the case of real data as it is presented in section 4.2.

Figure 27, Figure 28, and Figure 29 show the deleted segments after applying the first step. It can be seen that the deletions are reasonable and help to counteract the interpolation inaccuracy.

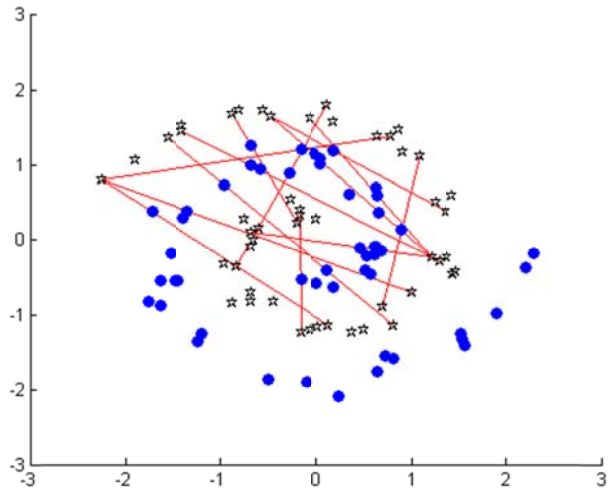

Figure 27: Deleted segments after applying error-based deletion step for class '★' of the two-spirals dataset.

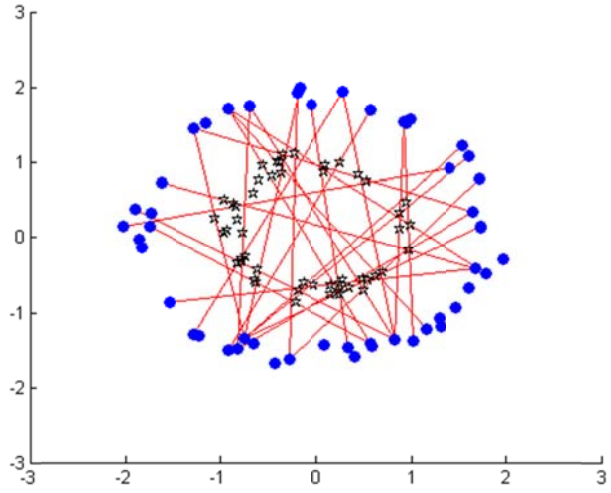

Figure 28: Deleted segments after applying error-based deletion step for class '●' of the rings dataset.

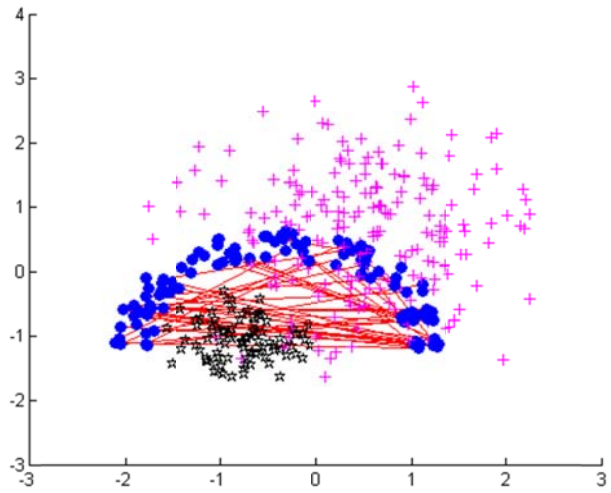

Figure 29: Deleted segments after applying error-based deletion step for class '●' of the cone-torus dataset.

Figure 30 to Figure 35 present the deleted and remaining FLSs after applying intersection-based deletion step. The number of deleted segments is 454 for rings, 1451 for the two-spirals and 15101 for the cone-torus dataset.

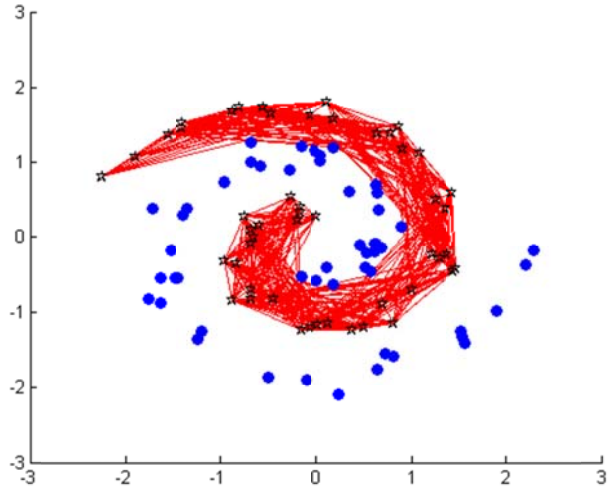

Figure 30: Remaining segments after applying intersection-based deletion step for class '★' of the two-spirals dataset.

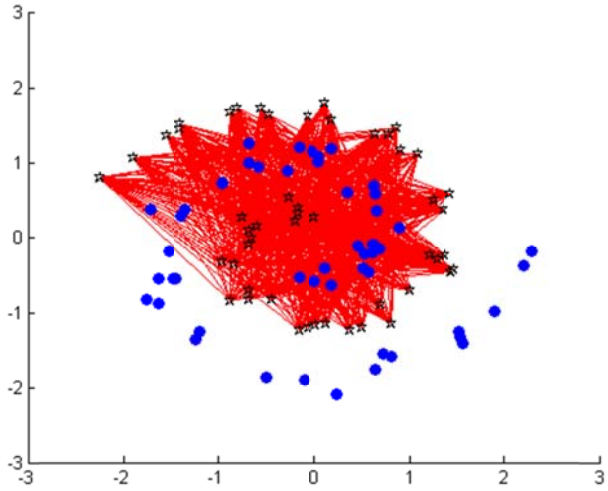

Figure 31: Deleted segments after applying intersection-based deletion step for class '★' of the two-spirals dataset.

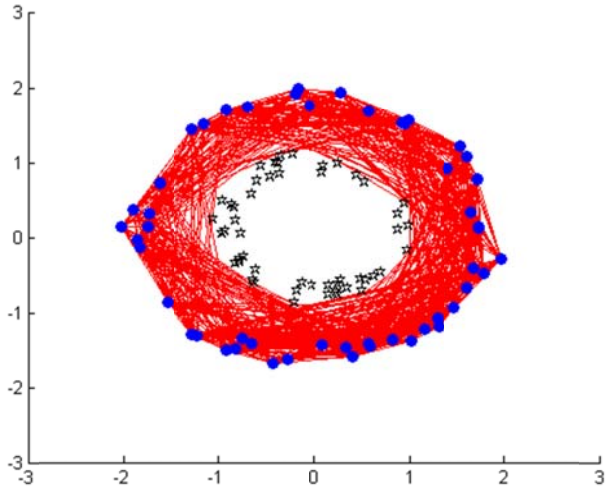

Figure 32: Remaining segments after applying intersection-based deletion step for class '●' of the rings dataset.

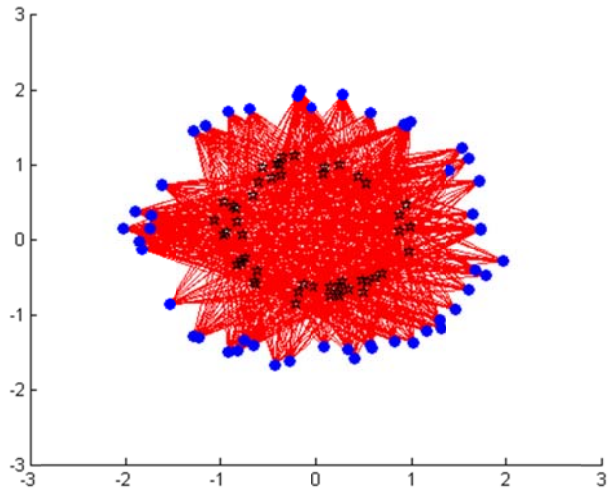

Figure 33: Deleted segments after applying intersection-based deletion step for class '●' of the rings dataset.

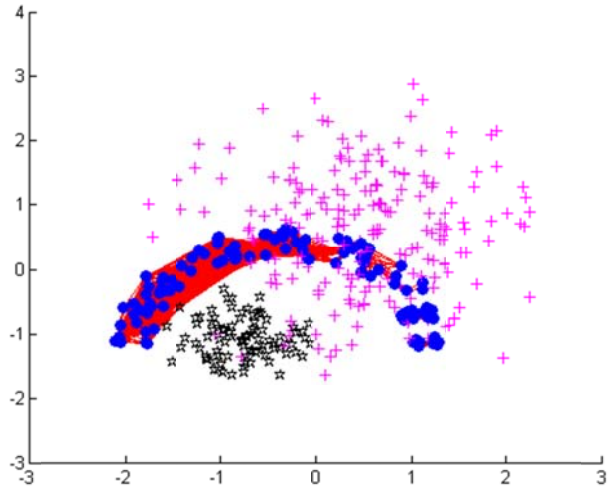

Figure 34: Remaining segments after applying intersection-based deletion step for class '●' of the cone-torus dataset.

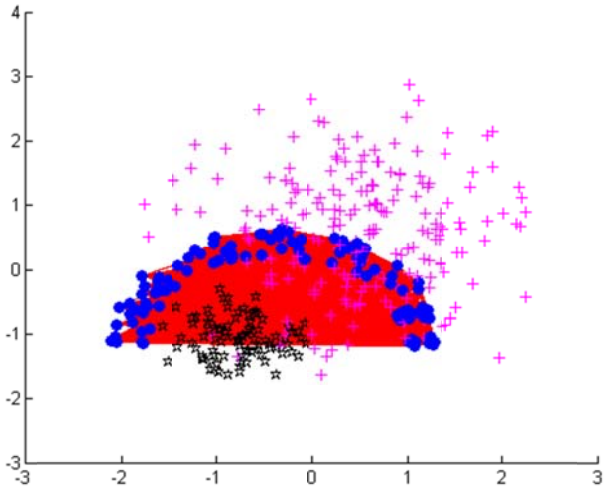

Figure 35: Deleted segments after applying intersection-based deletion step for class '●' of the cone-torus dataset.

The remaining and deleted FLSs after applying the pruning step are presented in Figure 36 to Figure 41 respectively for rings, two-spirals and cone-torus datasets.

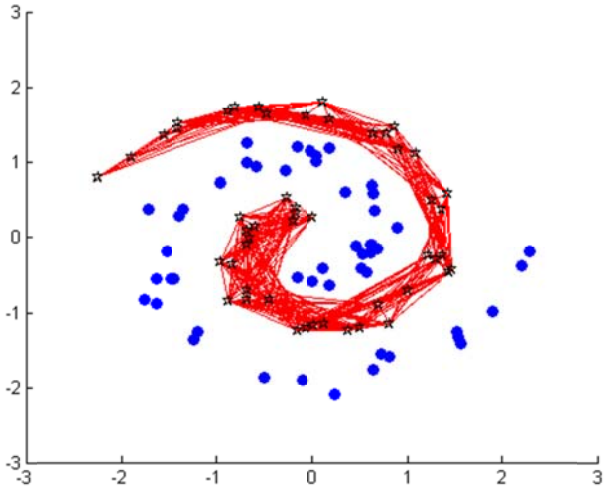

Figure 36: Remaining segments after applying the pruning step for class '★' of the two-spirals dataset.

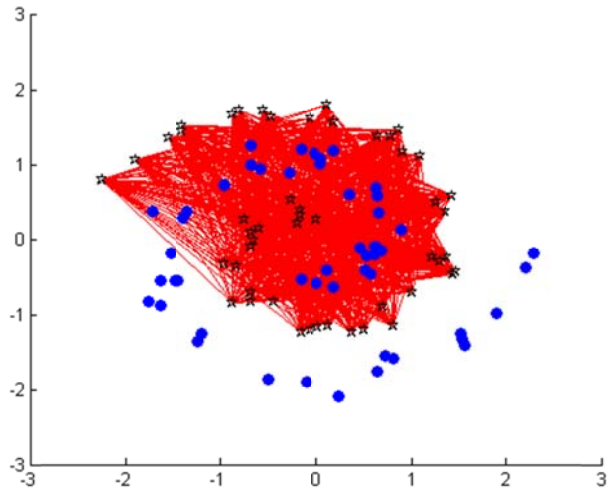

Figure 37: Deleted segments after applying the pruning step for class '★' of the spirals dataset.

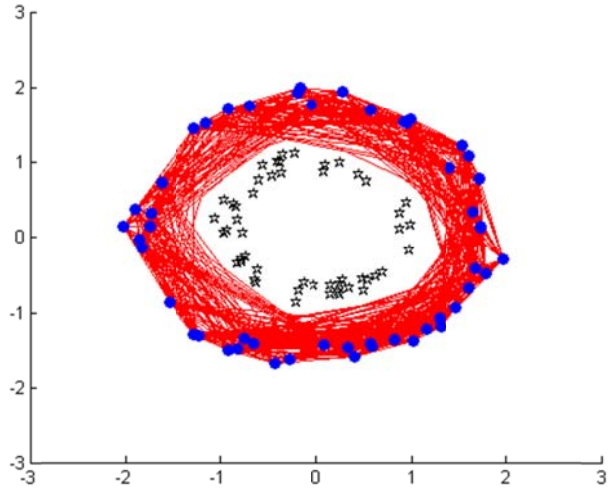

Figure 38: Remaining segments after applying the pruning step for class '●' of the rings dataset.

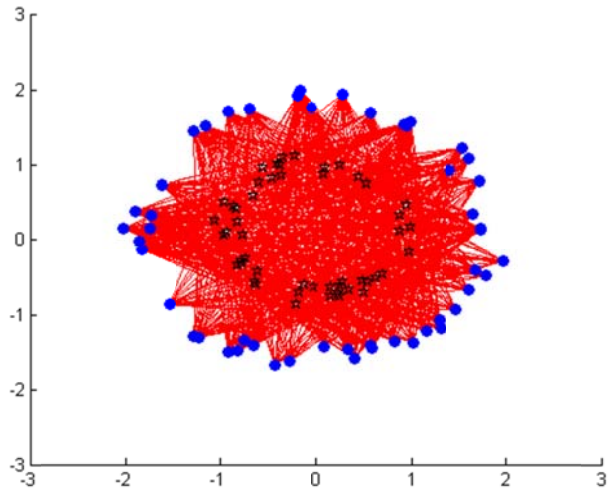

Figure 39: Deleted segments after applying the pruning step for class '●' of the rings dataset.

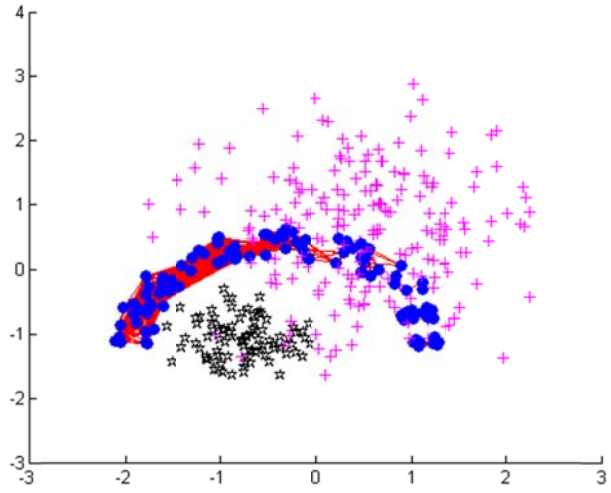

Figure 40: Remaining segments after applying the pruning step for class '●' of the cone-torus dataset.

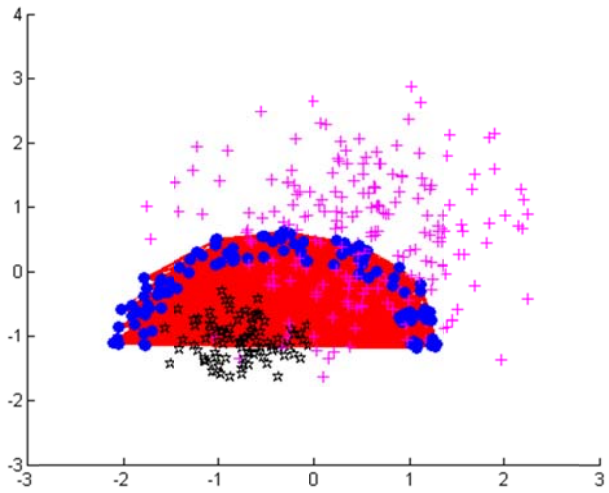

Figure 41: Deleted segments after applying the pruning step for class '●' of the cone-torus dataset.

The effect of pruning can be clearly seen by comparing Figure 30 and Figure 36 for two-spirals or by comparing Figure 32 and Figure 38 for the rings dataset. The deletions mainly modify the decision boundaries, removing the feature lines that are very close to the samples of a different class. At the end of pruning step the number of deleted segments are 552 for the rings, 1674 for the two-spirals and 17210 for the cone-torus dataset.

Table 1 presents the total number of FLSs deleted in each step of the algorithm and the numbers of deletions are also presented for the RNFLS algorithm. It can be seen that the numbers of deleted segments are comparable on cone-torus dataset whereas the proposed algorithm deletes smaller number of segments for the rings and two-spirals dataset. In fact, larger number of deleted segments corresponds to reduced computational complexity during testing. This can also be achieved by the proposed scheme by choosing smaller β value in higher dimensional space. However, the primary performance criterion of this study is the classification accuracy rather than

the number of deletions. In fact, if a given scheme deletes more segments at the expense of the accuracy, this is not desired. In this study, the proposed approach is compared with the reference systems in terms of both the number of segments employed and the accuracies achieved on fifteen real datasets.

Table 1: Number of deleted segments.

Datasets	Error-based deletion	Intersection-based deletion	Pruning for each class	eNFL Total	eNFL Percentage	RNFLS
rings	0+25	0+454	0+552	552	22.53	1113
two-spirals	13+15	692+849	803+871	1674	68.33	2090
cone-torus	3+37+62	363+3167+11571	1318+3729+12163	17210	55.93	17359

4.2 Experiments on Real Data

The experiments are conducted on twelve datasets from the UCI machine repository, "Clouds" and "Concentric" from ELENA and "Australian" from IAP TC 5 datasets. Table 1 presents the description of the datasets including the number of classes, number of features and the number of samples.

Table 1: Characteristics of the datasets.

dataset	Number of classes	Number of features	Number of samples
Australian	2	42	690
Cancer	2	9	683
Clouds	2	2	5000
Concentric	2	2	2500
Dermatology	6	34	366
Haberman	2	3	306
Heart	2	13	303
Ionosphere	2	34	351
Iris	3	4	150
Pima	2	8	768
Spect	2	22	267
Spectf	2	44	267
Wdbc	2	30	569
Wine	3	13	178
Wpbc	2	32	194

In order to compare different approaches, the hold-out method is employed to generate the training and test sets. The given data is randomly divided into two equal parts. The first part is used for training and the second part is used for testing. The data are normalized using zero-mean unit-variance normalization method where the normalization parameters are estimated using the training data. This procedure is repeated ten times to compute ten train/test splits. The simulations are done for each split and the average accuracies are reported. For "clouds" and "concentric", 10% of the data is used for training and 90% for testing. The average accuracies achieved using the reference systems are presented in Table 2. It can be easily seen in the table that both RNFLS and SFLS surpasses NFL on majority of the datasets. More specifically, RNFLS provides better accuracies than NFL on 12 datasets and SFLS provided better accuracies on 10 datasets. On the other hand, the performances of SFLS and RNFLS are comparable.

Table 2: The average accuracies achieved on ten independent simulations.

Dataset	**NFL**	**RNFLS**	**SFLS**
Australian	79.85	79.88	81.28
Cancer	95.04	96.86	96.77
Clouds	65.09	86.48	86.94
Concentric	63.58	97.23	96.87
Dermatology	96.15	95.27	95.55
Haberman	70.66	69.41	70.07
Heart	78.34	79.27	78.54
Ionosphere	84.11	90.74	89.09
Iris	87.73	94.00	94.40
Pima	68.23	73.02	71.77
Spect	80.38	81.50	80.23
Spectf	76.33	78.13	78.88
Wdbc	94.33	96.13	95.60
Wine	96.14	95.80	94.77
Wpbc	71.24	73.61	70.10
Average	80.48	85.82	85.39

The accuracies achieved by the proposed scheme are presented in Table 3 for fixed and minimum segment length based thresholding approaches. The second column provides the accuracies for $\tau = 1$. The following four columns present the accuracies achieved for five different β values. The last column presents the scores achieved when the best-fitting value of β is computed by applying 3-fold cross-validation on the training data. As it illustrated in Figure 42, each training set is randomly partitioned into three subsets for this purpose. Two subsets are used for training and the remaining for evaluation. This procedure is repeated three times and the β value providing the best average result over all three partitions is selected. The parameter tuning described above is done for each of the ten train/test splits separately. During testing, the best-fitting value is considered. It should be noted that, for $\beta < 2$ and $\beta > 5$ the average accuracies achieved are generally worse compared to β in the interval [2,5]. Because of this, we employed $2 \leq \beta \leq 5$ for computing the best-fitting β value.

Figure 42: Splitting the training data into three folds for the tuning of β.
White parts denote the evaluation data.

Table 3: The accuracies achieved by the proposed approach. The best scores achieved for each datasets are presented in boldface.

Dataset	Fixed threshold $\tau = 1$	Minimum segment length based threshold				optimum β
		$\beta = 2$	$\beta = 3$	$\beta = 4$	$\beta = 5$	
Australian	81.60	80.00	81.74	81.80	81.63	**81.92**
Cancer	96.69	96.60	96.77	96.86	**96.98**	96.80
Clouds	**87.13**	**87.13**	86.41	86.72	**87.13**	**87.13**
Concentric	**97.48**	**97.48**	**97.48**	**97.48**	**97.48**	**97.48**
Dermatology	95.71	95.88	95.71	95.71	95.71	**95.82**
Haberman	**71.51**	70.79	70.20	70.13	69.67	71.12
Heart	79.87	**80.00**	79.93	79.80	79.80	79.80
Ionosphere	**91.09**	86.57	90.46	90.74	90.74	90.63
Iris	93.07	94.00	**94.40**	94.13	94.00	**94.40**
Pima	**73.54**	71.72	72.76	73.07	73.10	72.79
Spect	80.98	80.45	80.60	**81.05**	80.98	80.45
Spectf	78.35	77.90	**78.73**	**78.73**	**78.73**	78.35
Wdbc	96.20	96.34	**96.37**	96.27	96.23	94.61
Wine	97.28	**97.39**	**97.39**	97.27	97.27	**97.39**
Wpbc	72.78	**74.85**	73.61	72.99	72.99	74.12
Average	86.22	85.81	80.41	86.15	80.35	86.19

The best scores achieved for each dataset are presented in boldface. It can be seen in the table that the best-fitting value of β is problem dependent. By employing the best-fitting value of β, the highest scores are achieved on six datasets. However, the simpler system corresponding to $\tau = 1$ achieved comparable performance, even providing a slightly better average accuracy. The results clearly show that the proposed hyper-cylinder based approach has the potential to provide improved accuracies compared to the fixed threshold scheme. However, employing a better scheme for tuning the threshold parameter β is essential.

In the following context, we will refer the proposed fixed threshold system for $\tau = 1$ as the eNFL scheme. Comparing the results in Tables 2 and 3, it can be seen that eNFL provides better accuracies compared to NFL, RNFLS and SFLS on 14, 11 and 12 datasets respectively.

eNFL is also compared with the references in terms of the ranking performances. More specifically, the performances achieved by the proposed and reference systems in terms of their ranks when sorted using average accuracies are computed. The results are presented in Table 4. For instance, in the case of "Australian" dataset, eNFL is the best and RNFLS is the third best system. As seen in the table, eNFL has remarkably better performance compared to the reference systems.

The numbers of segments deleted RNFLS and eNFL are presented in Table 5. The total number of segments for each dataset is presented in the second column. On the average, approximately half of the total numbers of segments are deleted by both RNFLS and eNFL where RNFLS is found to delete approximately 20% more compared to eNFL. On "Australian", "Dermatology", "Heart", "Ionosphere" and "Wine", the number of segments deleted by RNFLS is much above the average compared to eNFL. However, on all these datasets, eNFL performed better. It can be concluded that deleting more segments does not necessarily lead to a better scheme in terms of classification accuracy. On the contrary, useful segments may be lost. It should also be noted that, in SFLS, there is not segment deletion during training.

Table 4: The performances achieved by the proposed and reference systems in terms of their ranks when sorted using average accuracies.

Dataset	NFL	SFLS	RNFLS	eNFL
Australian	4	2	3	1
Cancer	4	2	1	3
Clouds	4	2	3	1
Concentric	4	3	2	1
Dermatology	1	3	4	2
Haberman	2	3	4	1
Heart	4	3	2	1
Ionosphere	4	3	1	1
Iris	4	1	1	2
Pima	4	3	1	1
Spect	3	4	1	1
Spectf	4	1	3	1
Wdbc	4	2	1	1
Wine	2	4	3	1
Wpbc	3	4	1	2
Average	3.40	2.67	2.07	1.33

Table 5: The total number of segments in each dataset and the number of deleted segments for four different schemes.

Dataset	Total number of segments	RNFLS	eNFL
Australian	30117	26893	11112
Cancer	31671	3034	3259
Clouds	62250	40586	45492
Concentric	16681	8524	7594
Dermatology	3305	1403	257
Haberman	7148	4946	5132
Heart	5736	5041	2946
Ionosphere	8281	3509	844
Iris	900	112	150
Pima	40036	27546	23497
Spect	5943	2115	2483
Spectf	5930	2605	2017
Wdbc	21496	3930	3845
Wine	1341	820	137
Wpbc	2954	2391	1635
Average	16252.6	8897	7360

As a final remark, it should be mentioned that, for the "Dermatology" dataset which has 6 classes, the performance of NFL is the best among all other classifiers and this is the only dataset that the proposed method provided inferior performance compared to NFL.

Chapter 5

CONCLUSION AND FUTURE WORK

The focus of this study was to edit segments employed by the NFL classifier and propose a new approach to suppress the interpolation inaccuracy of NFL. The proposed approach is composed of three steps namely, error-based deletion, intersection-based deletion and pruning. The characteristics of the steps applied are clarified by running the proposed system on three real datasets where the deleted and retained segments are presented

The proposed method is evaluated on fifteen different datasets from different domains and improved accuracies are achieved compared to NFL, RNFLS and SFLS on 14, 11 and 12 datasets respectively.

By ranking the accuracies achieved by the schemes considered, it is observed that the proposed method ranked best on 11 datasets and second on 3 datasets.

The proposed method is also evaluated in terms of the number of deleted segments. It is observed that, on the average over fifteen datasets, approximately half of the total number of segments are deleted by both RNFLS and eNFL where RNFLS is found to delete approximately 20% more compared to eNFL.

There are two major topics that should be further explored. The first is the optimal estimation of τ using the training data instead of using the constant one. The other is

to explore better schemes for the computation of β in hyper-cylinder based approach. Instead of 3-fold cross validation, the leave-one-out error estimation scheme can be considered.

REFERENCES

[1] Duda, R.O., P.E. Hart, and D.G. Stork, (2001), *Pattern classification*. John Wiley & Sons.

[2] Bishop, C.M., (1997), *Neural networks for pattern recognition*. Oxford.

[3] Jain, A.K., R.P.W. Duin, and J.C. Mao, (2000), *Statistical pattern recognition: A review.* IEEE Transactions on Pattern Analysis and Machine Intelligence. **22**(1): p. 4-37.

[4] Kuncheva, L.I., (1995), *Editing for the k-nearest neighbors rule by a genetic algorithm.* Pattern Recognition Letters. **16**(8): p. 809-814.

[5] Li, S.Z. and J. Lu, (1999), *Face recognition using the nearest feature line method.* IEEE Transactions on Neural Networks. **10**(2): p. 439-443.

[6] Cunningham, P. and S.J. Delany, (2007), *k-Nearest neighbour classifiers.* Multiple Classifier Systems: p. 1-17.

[7] Elkan, C., (2011), *Nearest Neighbor Classification,* University of California.

[8] Zhou, Z., S.Z. Li, and K.L. Chan. *A theoretical justification of nearest feature line method*. in *Proceedings. 15th International Conference on Pattern Recognition*. 2000. IEEE: p. 759-762.

[9] Han, D.Q., C.Z. Han, and Y. Yang, (2011), *A novel classifier based on shortest feature line segment.* Pattern Recognition Letters. **32**(3): p. 485-493.

[10] He, Y. *Face recognition using kernel nearest feature classifiers*. in *Computational Intelligence and Security, 2006 International Conference on*. 2006. IEEE: p. 678-683.

[11] Gao, Q.B. and Z.Z. Wang, (2007), *Center-based nearest neighbor classifier.* Pattern Recognition. **40**(1): p. 346-349.

[12] Zheng, W., L. Zhao, and C. Zou, (2004), *Locally nearest neighbor classifiers for pattern classification.* Pattern Recognition. **37**(6): p. 1307-1309.

[13] Zhou, Y.L., C.S. Zhang, and J.C. Wang, (2004), *Tunable nearest neighbor classifier.* Pattern Recognition, Lecture notes in computer science, **3175**: p. 204-211.

[14] Du, H. and Y.Q. Chen, (2007), *Rectified nearest feature line segment for pattern classification.* Pattern Recognition. **40**(5): p. 1486-1497.

[15] Wilson, D.L., (1972), *Asymptotic properties of nearest neighbor rules using edited data.* IEEE Transactions on Systems, Man and Cybernetics(3): p. 408-421.

[16] Devijver, P.A. and J. Kittler, (1982), *Pattern recognition: A statistical approach*. Prentice/Hall International.

[17] Nanni, L. and A. Lumini, (2011), *Prototype reduction techniques: A comparison among different approaches.* Expert Systems with Applications. **38**(9): p. 11820-11828.

[18] Frank, A. and A. Asuncion, (2010), *UCI Machine Learning Repository [http://archive. ics. uci. edu/ml]. Irvine, CA: University of California.* School of Information and Computer Science. **213**.

[19] Cover, T. and P. Hart, (1967), *Nearest neighbor pattern classification.* Information Theory, IEEE Transactions on. **13**(1): p. 21-27.

[20] Bay, S.D. *Combining nearest neighbor classifiers through multiple feature subsets*. in *Proceedings of the Fifteenth International Conference on Machine Learning*. 1998: p. 37-45.

APPENDICES

Minimum Distance between Two Lines in N-Dimensional Space

The minimum distance between two segments (line) is the length of the line while is perpendicular to both of the lines [21].

Let $x_1$and x_2 denote two points that belong to class '●' and x_3 and x_4 belongs to class '★'. Let p_1 and p_2 denote the unique points when the two lines are closet where d is the unique minimum as illustrated in Figure 43. It can be shown that these points are unique when the lines are not parallel [21]. When, $\widehat{x_1x_2}$ and $\widehat{x_2x_3}$ are not parallel and do not intersect each other, $\widehat{p_1p_2}$ joining these points in unique and perpendicular to both segments.

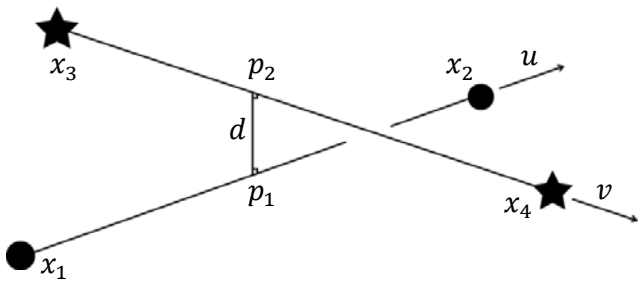

Figure 43: Minimum distance between two lines.

Let $u = \frac{x_2 - x_1}{\|x_2 - x_1\|}$ and $v = \frac{x_4 - x_3}{\|x_4 - x_3\|}$ denote the unit vectors in the directions presented in Figure 43. p_1 and p_2 can be written as

$$p_1 = x_1 + su$$

$$p_2 = x_3 + tv$$

where $s, t \in \mathbb{R}$. Let $w = (p_2 - p_1)$, where $d = \|w\|$. We can rewrite w as

$$w = (x_1 - x_3) + (su - tv) = w_0 + (su - tv).$$

Since, w is perpendicular to u and v,

$$u.w = 0$$

$$v.w = 0$$

Substituting w in the expressions above, we get

$$u.w_0 + su.u - tu.v = 0$$

$$v.w_0 + sv.v - tv.v = 0.$$

Rewriting, we obtain

$$(u.u)s - (v.v)t = -u.w_0$$

$$(v.u)s - (v.v)t = -u.w_0.$$

Letting $a = u.u$, $b = u.v$, $c = v.v$, $d = u.w_0$ and $e = v.w_0$, s and t can be obtained as

$$s = \frac{be - cd}{ac - b^2}$$

$$t = \frac{ae - bd}{ac - b^2}$$

hence, w can be computed as

$$w = p_2 - p_1 = x_3 + \frac{ae - bd}{ac - b^2}u - x_1 - \frac{be - cd}{ac - b^2}v$$

$$w = (x_3 - x_1) + \frac{(ae - bd)u - (be - cd)v}{(ac - b^2)}.$$

After computing w, d is obtained as $d = \|w\|$.