Ways to Find the Optimal Parameters of Kernel Functions for Support Vector Machine Classification

Abstract:

This report reviews and summaries the main ideas of the kernel method and support vector machine for classification. By transforming the original space into a high dimensional space, kernel functions make a nonlinear classification problem linearly separable. Support vector machine (SVM) can be used to find hyperplanes or linear separators, which can divide different labelled data classes in the kernel feature space. As a result, researchers try to find optimal parameters for a better separation of data. An essay written by scholars in National Taichung University gives an automatic way to select the parameters in RBF kernels. The authors aim to find the parameters in the RBF kernel by a trade-off between maximizing the within-class cosine values and minimizing the between-class cosine values (Cheng-Hsuan, Chin-Teng, Bor-Chen, &Hui-Shan, 2010). Considering the differences in sizes between different classes, this report proposes two alternative methods to find the optimal parameters and generalizes it to some other kernels of one parameter. The first method is a modification of the existing method in the previous essay. Different from the existing method, it finds kernel parameters by first taking the average sum of kernel function values between points within each class or in every two classes. The second method finds parameters by minimizing the average expected distances from centers of mass and maximizing the average distances between centers of mass. The optimal parameter found by these two methods is supposed to contribute to a better SVM classification.

Key words: kernel method, optimal parameters, classification, support vector machine

Acknowledge:

I would like to acknowledge the Chinese University of Hongkong, Shenzhen and the Undergraduate Research Award Support, which give me a research opportunity and financial support. I would like to express my very great gratitude to professor Wang Dong, for his guidance and suggestions in learning and support for the project. I would like to appreciate my groupmates Qiao Gaojie, LinZhitong, and WanSicheng, for their assistance in learning. I

would also like to thank the book *Kernel Method for Pattern Analysis*, which contains a lot of useful knowledge and contributions to the project. I would like to give special thanks for some free online courses: Andrew Ng *Machine learning* course in Coursera and LiZhengxuan *Kernel Method* (a series of courses) in YouTube and other online materials, for their help in learning and the research.

1. Introduction

1.1 The Importance and Advantages of Kernel Method

In the era of data explosion, data analysis is increasingly prevalent nowadays. Based on the existing data or the previous data, people hope to find a certain and regular pattern and use it to predict the unknowns in order to make wise choices. This process is called machine learning. We hope to develop some algorithms to build models and find patterns from the training set and evaluate the model in the test set. How to analyze and predict the data precisely and efficiently is always a central topic.

In the real world, pure linear models are almost non-existence, because there are usually external disturbance. In machine learning problems, nonlinear models are often required. Linear model is well-understood and it possesses a lot of useful properties. Therefore, we desire a way to transform our non-linear problem into a linear one. In order to make the data linearly separable, there is a strategy named kernel method to transform the original low dimensional feature space into a high dimensional feature space. In addition to this transformation, kernel method has some other advantages, such as computational efficiency and applicability to non-vectorial inputs, which contribute to its practical implementation in real-world nonlinear problem.

1.2 Kernel Function and Kernel Matrix

The followings are summary and reinterpretation of the book *Kernel Methods for Pattern Analysis* (John & Nello, 2020, p.34-p.53).

Definition (Kernel Function)

A kernel is a function κ that for all $\mathbf{x}, \mathbf{z} \in \mathrm{X}$ satisfies

$$\kappa(\boldsymbol{x}, \boldsymbol{z}) = < \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) > \quad (1.1)$$

Where ϕ is a mapping from **X** to an inner product feature space *F*

$$\phi : \boldsymbol{x} \to \phi(\boldsymbol{x}) \in F$$

Definition (Kernel Matrix)

Given a set of vectors $\mathrm{S} = \{x_1, \ldots, x_l\}$ the kernel function κ and the feature map ϕ, the kernel matrix (Gram matrix) an $l \times l$ matrix **K** whose entries are

$$\boldsymbol{K}_{ij} = < \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) > = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad (1.2)$$

Although the kernel matrix loses information about the orientation of the original data set with respect to the origin due to the rotational invariance of inner product, the kernels matrix contains all the information needed to compute the pairwise distances within the data set. To compute the distances, the equation is

$$\left\| \phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j) \right\| = \sqrt{\kappa(\boldsymbol{x}_i, \boldsymbol{x}_i)^2 - 2\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) + \kappa(\boldsymbol{x}_j, \boldsymbol{x}_j)^2} \quad (1.3)$$

To compute the angle between feature space, the equation is

$$cos\theta = \frac{< \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) >}{\left\| \phi(\boldsymbol{x}_i) \right\| \left\| \phi(\boldsymbol{x}_j) \right\|} = \frac{\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sqrt{\kappa(\boldsymbol{x}_i, \boldsymbol{x}_i)} \sqrt{\kappa(\boldsymbol{x}_j, \boldsymbol{x}_j)}} \quad (1.4)$$

It is usually easier to compute the inner product between the projections of two points into the feature space than the explicit coordinates of the embedding points. This lays a theoretical foundation for solving the complex classification or regression problems in the high-dimensional feature space since only pairs of inner product are needed in these algorithms.

1.3 Various Kinds of Kernels and Kernel Construction

The followings are mainly summary and reinterpretation of the book *Kernel Methods for Pattern Analysis* (John & Nello, 2020, p.49-p.79).

There are several kinds of widely-used kernels and their corresponding properties.

1)Linear kernel:

$$\kappa(\boldsymbol{x}, z) = < \boldsymbol{x}, z > \quad (1.5)$$

The feature mapping of this kernel is an identity matrix.

2)Polynomial kernel:

$$\kappa(\boldsymbol{x}, z) = (< \boldsymbol{x}, z > + R)^d \quad d \in Z^+ \ (1.6)$$

Polynomial kernel allows learning nonlinear model. However, sometimes it is computationally inefficient and faces the problem of overfitting.

3)Radial basis function (RBF/Gaussian) kernel:

$$\kappa(\boldsymbol{x}, z) = exp\left(\frac{- \| \boldsymbol{x} - z \|^2}{2\sigma^2}\right) \ \sigma \in R\backslash\{0\} \ (1.7)$$

Gaussian kernel can transform a low dimensional feature space into an infinite dimensional feature space.

4)Sigmoid kernel:

$$\kappa(\boldsymbol{x}, z) = tanh(\gamma \boldsymbol{x}^{\boldsymbol{T}} z + r) \ (1.8)$$

There are some ways to determine whether a kernel is a linear kernel. **K** is a valid kernel provided its kernel matrices are positive semi-definite for all training sets S or the kernel function satisfies the positive semi-definite property. There are more details explanation about verifying a valid kernel in the book *Kernel Function for Pattern Analysis.* Given a valid kernel, there are ways to find its corresponding kernel feature space, or *Reproducing Kernel Hilbert Space(RKHS)*.

There are some rules to manipulating and combining simple kernels to construct more complicated and useful ones. Let κ_1 and κ_2 be kernels over $\mathbf{X} \times \mathbf{X}, \mathbf{X} \subseteq R^n$, a $\in R^+, f(\bullet)$ a real-valued function on $\mathbf{X}$. The following kernel functions can be proved to be valid kernels. (Closure Properties)

i) $\kappa(\boldsymbol{x}, z) = \kappa_1(\boldsymbol{x}, z) + \kappa_2(\boldsymbol{x}, z)$

ii) $\kappa(\boldsymbol{x}, z) = a\kappa_1(\boldsymbol{x}, z)$

iii) $\kappa(\boldsymbol{x}, z) = \kappa_1(\boldsymbol{x}, z)\kappa_2(\boldsymbol{x}, z)$

iv) $\kappa(\boldsymbol{x}, z) = f(\boldsymbol{x})f(z)$

The polynomial is a valid kernel since it can be constructed by the above operations. Gaussian kernel is a polynomial of infinite degree and it is a limit of polynomial kernel.

$$\exp(\mathbf{x}) = \sum_{i=0}^{\infty} \frac{1}{i!}\boldsymbol{x}^{\boldsymbol{i}} \quad (1.9)$$

The Gaussian kernel is a valid kernel since the finitely positive semi-definiteness property is closed under taking pointwise limits.

1.4 Fitting Problem and Kernel Functions' Parameters for Regression

A linear model is usually not enough to describe a non-linear pattern. A regression problem produces a continued valued output. For this kind of problem, the linear model may have high bias and suffer from underfitting. Linear model is often used when the model has large number of features but relatively small number of data. For model built on polynomial kernel and Gaussian kernel, the σ in the RBF kernel cannot be too small and the degree of polynomial kernel d cannot be too large. Otherwise, the model may suffer from high variance and overfitting, which means it does not perform well in the test data despite the model is well-fitted for the training data. Therefore, we need to choose suitable parameters and find techniques to ameliorate or reduce the overfitting problem. Except from reducing the number of features, there is a strategy called regularization, which defines a relative trade-off between the norm and the loss. The coefficient of the norm is a parameter that can control the degree of regularization. (Andrew, 2019).

1.5 Kernel Support Vector Machine for Classification

1.5.1 SVM Introduction

Different from the aim of a regression problem, the goal of a classification problem is to produce a discrete value output. However, similar to regression problem, the selection of kernel parameters are of vital importance.

Support vector machine is a statistical technique used in supervised learning, where labelled dataset and certain data structure is given. It is used to find an optimal boundary to divide different classes. In this articles, only batch algorithms are considered, which means all

of the training examples are processed at once. Although logistic regression is a method for classification, if the number of training examples are several times larger than the number of features, SVM with Gaussian kernel is a better choice. (Andrew, 2019). There are two common types of support vector machine algorithms for classification. One is hard margin classifier (maximum margin classifier), the other is soft margin classifier. Section 1.6.2 reviews the main ideas behind the hard margin and soft margin SVM.

2. Binary SVM Classification

The following equations are modifications of the equations in LiZhengxuan's videos. (Zhengxuan, 2018).

1) Kernel Hard Margin Support Vector Machine Classification

The aim is to find a hyperplane $\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) + \boldsymbol{b} = 0$ that separate data into two classes, where the label $y_i \in \{-1, +1\}$. A training set is linearly separable with a certain margin if there exist $\boldsymbol{w}$ and $\boldsymbol{b}$ such that for all i from 1 to l, $\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x_i}) + \boldsymbol{b} \geq 1$ if $y_i = +1$ and $\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x_i}) + \boldsymbol{b} \leq -1$ if $y_i = -1$. The margin is the distance between two hyperplanes $\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x_i}) + \boldsymbol{b} = 1$ and $\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x_i}) + \boldsymbol{b} = -1$, which is $\frac{2}{\|\boldsymbol{w}\|}$. The vectors on these two hyperplane are called support vectors and maximizing the margin gives a good separation.

Primal Optimization Problem:

$$Minimize \;\; \Phi(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w^T w}$$

$$Subject\ to \;\; y_i\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x_i}) + \boldsymbol{b}\right) \geq 1 \quad (1.10)$$

Dual Problem:

$$Maximize \;\; W(\boldsymbol{\alpha}) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j \kappa(\boldsymbol{x_i}, \boldsymbol{x_j})$$

$$Subject\ to \;\; \sum_{i=1}^{l}\alpha_i y_i = 0$$

$$\alpha_i \geq 0 \quad \forall i = 1,\ldots,l \quad (1.11)$$

2) Kernel Soft Margin Support Vector Machine Classification

The training set may not be linearly separable. The algorithm add slack variables to allow some errors.

Primal Optimization Problem:

$$Minimize \;\; \Phi(\boldsymbol{w}, \boldsymbol{\varepsilon}) = \frac{1}{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w} + C\sum_{i=1}^{l}\varepsilon_i$$

$$Subject\ to \;\; y_i\left(\boldsymbol{w}^T\phi\left(\boldsymbol{x_i}\right) + \boldsymbol{b}\right) \geq 1 - \varepsilon_i \quad (1.12)$$

The parameters C can control the value of $\sum_{i=1}^{l}\varepsilon_i$ and consequently it controls the margin.

Dual Problem in Vector Form:

$$Maximize \;\; W(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\boldsymbol{T}}1_{l\times 1} - \frac{1}{2}\boldsymbol{\alpha}^{\boldsymbol{T}}\boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha}$$

$$Subject\ to \;\; \boldsymbol{\alpha}^{\boldsymbol{T}}\boldsymbol{y} = 0 \quad (1.13)$$

Where $\boldsymbol{y} = [y_1 \ldots y_l]^T$, $\boldsymbol{Y} = diag(y_1 \ldots y_l)$.

3. Multiclass SVM Classification

For multiclass SVM classification, there are basically two approaches: one-to-one approach and one-to-rest approach. The aim of one-to-one approach is to find hyperplanes to separate between every two classes. The classifier should use $\frac{L(L-1)}{2}$ SVMs. The goal of one-to-rest approach is to find hyperplanes to separate between each class and all corresponding other classes. The classifier should use L SVMs. (Chirag, 2021).

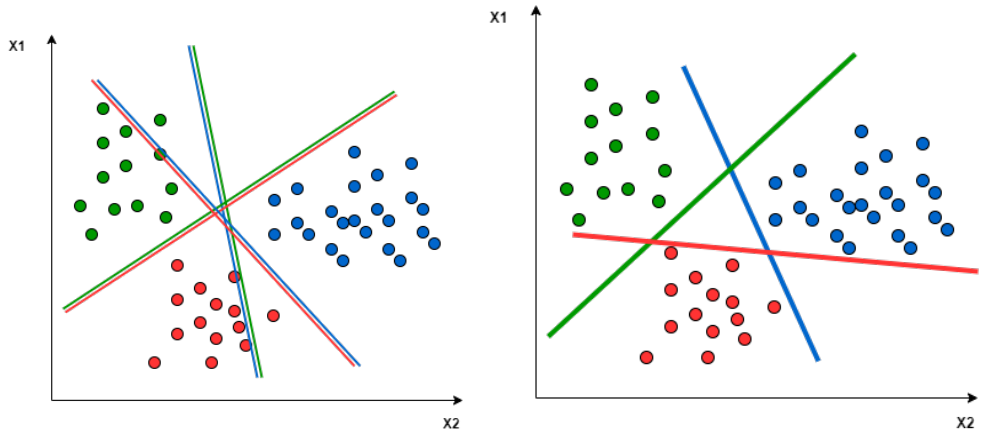

Figure 3(a): One-to-One Approach Figure 3(b): One-to-Rest Approach

For one-to-rest method, we use find these hyperplanes $w_i^T \phi(\boldsymbol{x}) + b_i = 0 \, for \, i = 1 \ldots L$ through support vector machines. The equation that determine the class of data point $\boldsymbol{x}$ is

$$k* = arg_k \max(\boldsymbol{w}_k^T \boldsymbol{\phi}(\boldsymbol{x}) + \boldsymbol{b}_k) \quad (1.14)$$

(Chirag, 2021).

1.6 Feature Scaling

Support vector machine classification algorithm use distances to determine similarity. Therefore, feature scaling is important to prevent kernel functions and this pattern analysis algorithm from being governed by certain features. “Normalization (Min-Max scaling) is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1.”(Aniruddha, 2020).

$$\boldsymbol{x'} = \frac{\boldsymbol{x} - \boldsymbol{x_{min}}}{\boldsymbol{x_{max}} - \boldsymbol{x_{min}}} \quad (1.15)$$

“Standardization is another scaling technique where the values are centered around the mean with a unit distribution has a unit standard deviation.” (Aniruddha, 2020).

$$\boldsymbol{x'} = \frac{\boldsymbol{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \quad (1.16)$$

Where μ is the mean of the feature values and σ is the standard deviation of the feature values.

1.7 Model Evaluation Technique

After developing a model by kernel SVM classification, we hope to evaluate the performance of the model. The accuracy and the error are not a best way to evaluate the model, because some classes may have much smaller size of data than other classes. These kind of classes are called skewed class. For simplicity, we consider model containing two classes. The percentage of the positive data is 99%, while the percentage of the negative data is 1%. Predicting positive all the time only has 1% error and it achieves 99% accuracy. However, it is not a good prediction model since we fail to detect the negative data.

There is an effective way to evaluate the model by calculating the F_1 scores. Among all the predicted positives data, the percentage that we have predicted correctly is

$$\mathrm{P} = \mathrm{Precision} = \frac{True\ Positives}{Predicted\ Positives} \quad (1.17)$$

Among all the actual positives data, the percentage that we predicted them correctly is

$$\mathrm{R} = \mathrm{Recall} = \frac{True\ Positives}{Actual\ Positives} \quad (1.18)$$

It is important to build a single real evaluate error metric to evaluate the models. Therefore, we use F_1 scores to compare precision and recall numbers, which is a trade-off between the precision and recall.

$$F_1 = 2\frac{PR}{P+R} \quad (1.19)$$

(Andrew, 2019)

2. Problem Statement and Formulation

2.1 The General Aim

We see in section 1 that the selection of parameters is of vital importance to kernel SVM classification problem. Kernel function with a good parameter can map data points into a space where data classes are well-separated. Therefore, this report aims to propose a method to find the optimal kernel parameters for kernel SVM classification by improving an existing method.

2.2 Review of a Parameters Selection Method

An essay written by scholars in National Taichung University gives an automatic way to select the parameters in the RBF kernel. “In the feature space determined by the RBF kernel, the norm of every sample is one and positive. Hence, the samples will be mapped onto the surface of a ball. The cosine values indicates the similarities between samples.” (Cheng-Hsuan, Chin-Teng, Bor-Chen, &Hui-Shan, 2010).

$$\left\|\phi(\boldsymbol{x})\right\|^2 = \kappa(\mathbf{x}, \mathbf{x}, \sigma) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}\|^2}{2\sigma^2}\right) = 1 \quad (2.1)$$

$$\cos\theta = \kappa(\mathbf{x}, \mathbf{z}, \sigma) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{z}\|^2}{2\sigma^2}\right) \quad (2.2)$$

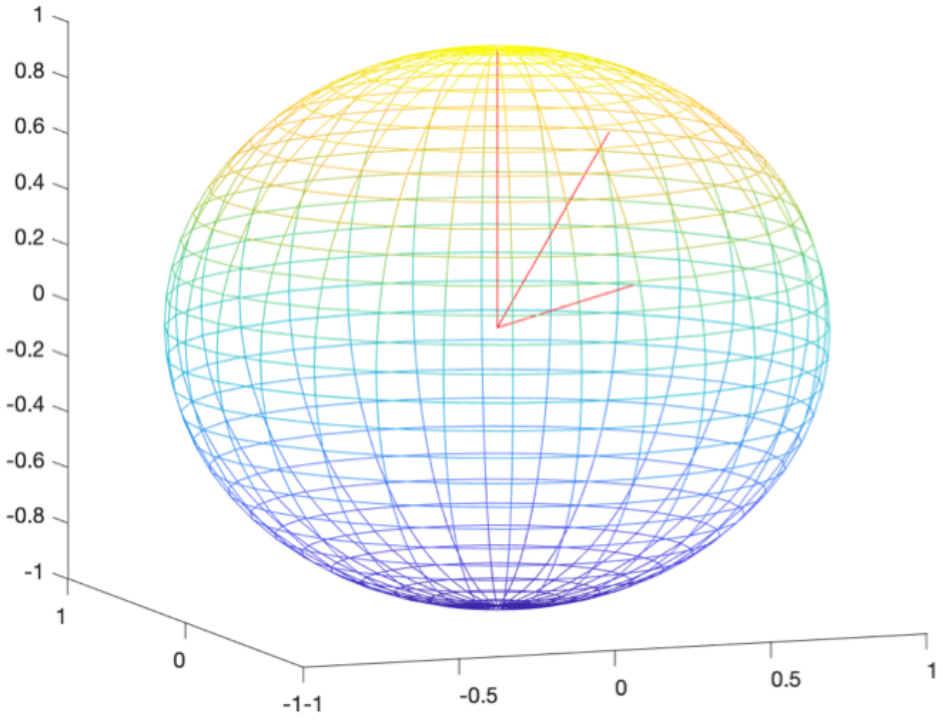

Figure 1: Visualization of three data vectors in the feature space determined by a RBF kernel

In order to obtain a better separation, the authors of the essay state that the distances between samples in the same clusters should be as small as possible while the distances between samples in the different class should be as large as possible. To adjust σ, the RBF

kernel function should be close to 1 if samples are in the same class and should be close to 0 if samples are in different classes. From a geometry point of view, in the feature space, the vectors in the same class should be nearly parallel while the vectors in the different classes should be approximately orthogonal to each other.

We assume that there are L clusters and the size of i^{th} cluster is N_i. Let $\boldsymbol{x}_i^{(j)}$ represents the i^{th} sample data in class j. The ideal distribution is as follows.

The cosines between vectors in the same class:

$$\kappa\left(\boldsymbol{x}_l^{(i)}, \boldsymbol{x}_k^{(i)}, \sigma\right) \approx 1$$

The cosines between vectors in the different classes:

$$\kappa\left(\boldsymbol{x}_l^{(i)}, \boldsymbol{x}_k^{(j)}, \sigma\right) \approx 0$$

The average within-class cosine values:

$$w(\sigma) = \frac{1}{\sum_{i=1}^{L} N_i^2} \sum_{i=1}^{L} \sum_{l=1}^{N_i} \sum_{k=1}^{N_i} \exp\left(-\frac{\left\|\boldsymbol{x}_l^{(i)} - \boldsymbol{x}_k^{(i)}\right\|^2}{2\sigma^2}\right) \approx 1 \quad (2.3a)$$

The average between-class cosine values:

$$b(\sigma) = \frac{1}{\sum_{i=1}^{L} \sum_{\substack{j=1 \\ j \neq i}}^{L} N_i N_j} \sum_{i=1}^{L} \sum_{\substack{j=1 \\ j \neq i}}^{L} \sum_{l=1}^{N_i} \sum_{k=1}^{N_i} \exp\left(-\frac{\left\|\boldsymbol{x}_l^{(i)} - \boldsymbol{x}_k^{(j)}\right\|^2}{2\sigma^2}\right) \approx 0 \quad (2.4a)$$

We represent the kernel matrix as a block matrix.

$$K = \begin{bmatrix} B_{11} & \cdots & B_{1L} \\ \vdots & \ddots & \vdots \\ B_{L1} & \cdots & B_{LL} \end{bmatrix}$$

$$\text{where } B_{ij} = \begin{bmatrix} \kappa\left(\boldsymbol{x}_1^{(i)}, \boldsymbol{x}_1^{(j)}, \sigma\right) & \cdots & \kappa\left(\boldsymbol{x}_1^{(i)}, \boldsymbol{x}_{N_j}^{(j)}, \sigma\right) \\ \vdots & \ddots & \vdots \\ \kappa\left(\boldsymbol{x}_{N_i}^{(i)}, \boldsymbol{x}_1^{(j)}, \sigma\right) & \cdots & \kappa\left(\boldsymbol{x}_{N_i}^{(i)}, \boldsymbol{x}_{N_j}^{(j)}, \sigma\right) \end{bmatrix}$$

The average sum of entries in the diagonal block ($w(\sigma)$) should be close to 1.The average sum of entries in the off-diagonal block ($b(\sigma)$) should be closed to 0. $b(\sigma)$ can be computed by subtracting the average sum of all entries in **K** from $w(\sigma)$.

$$w(\sigma) = \frac{1}{\sum_{i=1}^{L} N_i^2} \sum_{i=1}^{L} sum\ of\ entries(B_{ii}) \quad (2.3b)$$

$$b(\sigma) = \frac{1}{\sum_{i=1}^{L} \sum_{j=1}^{L} N_i N_j} sum\ of\ entries(B_{ij}) - w(\sigma) \quad (2.4b)$$

The proposed equation is defined as

$$\mathrm{J}(\sigma) = \left(1 - w(\sigma)\right) + b(\sigma) \quad (2.5)$$

and the strategy is minimizing $\mathrm{J}(\sigma)$, which indicates a trade-off between maximizing the within-class cosines and minimizing the between-class cosines.

Therefore, $\mathrm{J}(\sigma)$ indeed minimizes the average within-class squared distances and maximizes the average between-class squared distances.

The experiment result in the essay shows that $\mathrm{J}(\sigma)$ is a convex function which only has one global minima.

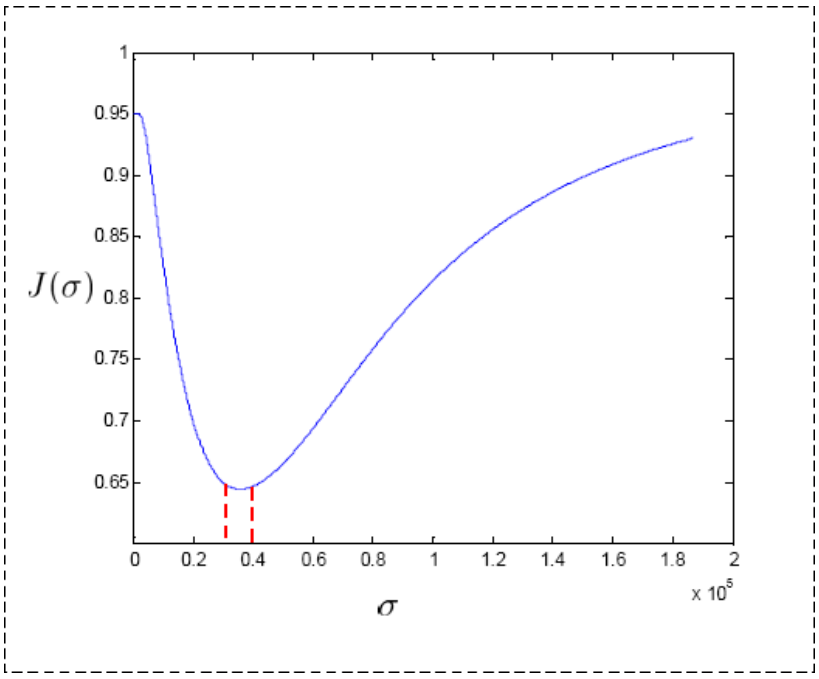

Figure 3: $\mathrm{J}(\sigma)$ *vs*. σ. The experiment result.

The author proposed gradient descent method to find the optimizer.

$$\sigma_{n+1} = \sigma_n - \gamma_n \nabla J(\sigma_n) \quad (2.6)$$

$$\nabla J(\sigma_n) = \frac{\partial}{\partial \sigma} b(\sigma_n) - \frac{\partial}{\partial \sigma} w(\sigma_n) \quad (2.7)$$

However, the computational time of gradient descent method is almost as much as that of k-fold cross-validation. Therefore, the author use another way to find the optimizer and it comes out that the algorithm runs more faster than k-fold cross-validation. As for the accuracy, the difference between k-fold cross-validation method and the proposed method are similar.

2.3 The Weakness of the Parameters Selection Method

$w(\sigma)$ and $b(\sigma)$ is the average sum of cosine values. $w(\sigma)$ is the average sum of the within-class cosine values of all classes. $b(\sigma)$ is the average sum of distances of all between-class cosine values. Therefore, this method does not consider the size of different clusters. If there is a large difference in size between different classes, the sum of cosine values related to the classes with smaller sizes has little weight than the sum of cosine values related to those with larger sizes. As a result, the clusters with small size may not have a well-centered data and a good separation from other classes in the kernel-defined feature space. In addition, the article does not generalize the method to other kernels. Without some modifications, this method is restricted to be used in RBF kernel.

2.4 The Specific Aim

This report aim to propose an alternative way to find the optimal parameters σ that tackles the problems in section 2.3. We aim to find parameter selection methods that solve problem in which data classes have large differences in size. In addition, we aim to find methods that could be generalized to various kinds of kernels.

3. Main Results

3.1 Method I

We assume that there are L clusters and the size of i^{th} cluster is N_i. Let $\boldsymbol{x}_i^{(j)}$ represents the i^{th} sample data in class j.

According the Law of Cosines, the cosine value between two vectors is negatively proportional to the squared distances between them. In RBF kernel, the equation is

$$\left\|\phi(\boldsymbol{x}_l^{(i)}) - \phi(\boldsymbol{x}_k^{(j)})\right\|^2 = 2 - 2cos\theta_{\boldsymbol{x}_l^{(i)}\boldsymbol{x}_k^{(j)}} = 2 - 2exp(-\frac{\left\|\boldsymbol{x}_l^{(i)} - \boldsymbol{x}_k^{(j)}\right\|^2}{2\sigma^2}) \quad (3.1)$$

Therefore, for the parameters selection method in section 2, although the authors use the average sum of cosine values, it is indeed the same as using the average sum of squared distances. To be specific, maximizing the within-class cosine values and minimizing the between-class cosine values are the same as minimizing the within-class squared distances and maximizing the between-class squared distances. As a result, kernel function with the chosen optimal parameter does give a good separation of different classes in the feature space.

In the normalized feature space, the inner product, or the value of kernel function between two vectors are also negatively proportional to the squared distances between them. Therefore, we can generalize the method in section 2.2 by replacing the RBF kernel to other kinds of kernel with normalized feature space. In this way, we generalize the method to some other kernels of one parameter *. As long as the $J(*)$ is convex, we can use gradient descent method to find the parameter *.

To avoid the "Size Effect" in Section 2.3, we do some further modifications to the method. $w(*)$ are calculated by first taking average kernel function values within each classes, denoted as $w_i(*)$, and then taking the average of all $w_i(*)$. $b(*)$ are calculated by first taking average kernel function values every two classes, denoted as $b_{ij}(*)$, and then taking the average of all $b_{ij}(*)$. In this way, we give every classes same emphasis.

$$w(*) = \frac{\sum_{i=1}^{L} w_i(*)}{L} = \frac{\sum_{i=1}^{L} \frac{1}{N_i} sum\ of\ entries(B_{ii})}{L}$$

$$b(*) = \frac{\sum_{i=1}^{L} b_{ij}(*)}{L} = \frac{\sum_{i=1}^{L}\sum_{i=1}^{L} \frac{1}{N_i N_j} sum\ of\ entries(B_{ij})}{L}$$

Instead of summing the $1 - w(*)$ and $b(*)$, we could also define $J(*)$ as the ratio of $b(*)$ to $w(*)$ and minimize it.

$$J(*) = \frac{b(*)}{w(*)} \quad (3.2)$$

3.2 Method II

The followings give a different method using distances and centers of mass. Some insights of this method are gained from an unsupervised learning approach, the k-means method. This method finds kernel parameters by minimizing the ratio of the average expected distances from centers of mass to the average distances between centers of mass.

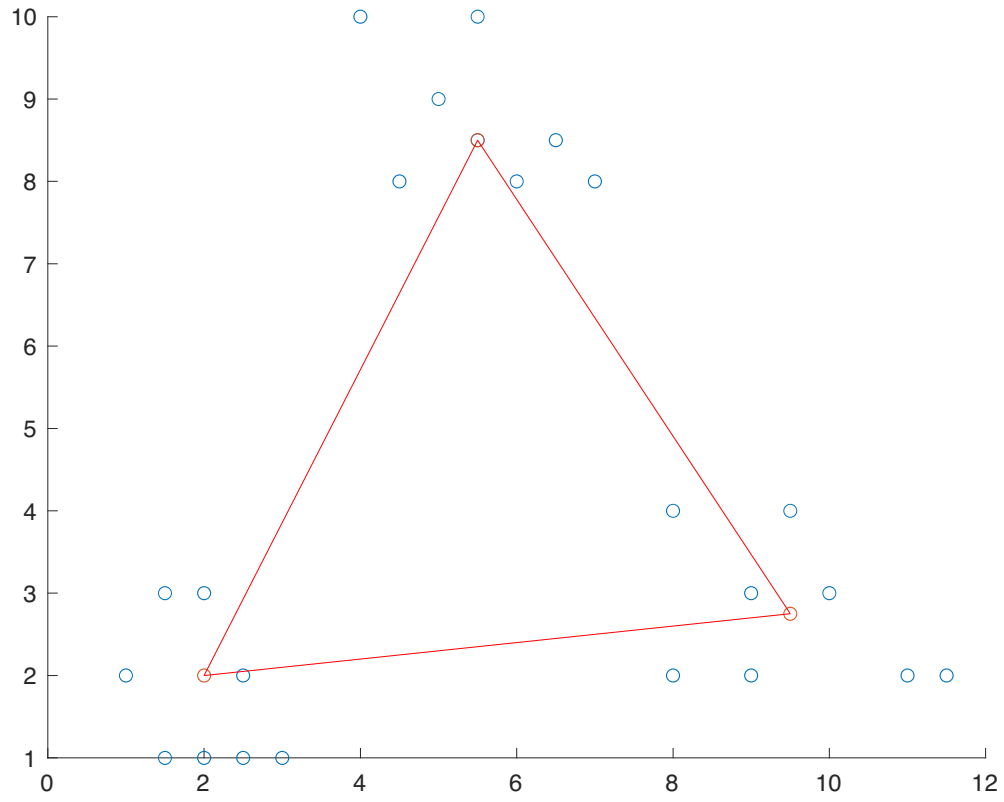

Figure 2: The distances between centers of mass of different clusters. (The red dots are the centers of mass. The red line segments are the distances between centers of mass.)

The equation 3.2 and equation 3.3 are slight modifications the equations in the book *kernel method of pattern analysis* (John & Nello, 2020, p.112-p.115).

The center of mass of class i:

$$\phi_S^{(i)} = \frac{1}{N_i} \sum_{k=1}^{N_i} \phi(x_k^{(i)}) \quad (3.3)$$

Although $\phi_S^{(i)}$ may not be computed explicitly, the norm if it can be evaluated through kernel matrix. The norm is (3.4)

$$\left\|\phi_S^{(i)}\right\|^2 = \langle \frac{1}{N_i}\sum_{k=1}^{N_i}\phi(x_k^{(i)}), \frac{1}{N_i}\sum_{l=1}^{N_i}\phi(x_l^{(i)})\rangle$$

$$= \frac{1}{N_i^2}\sum_{k,l=1}^{N_i}\kappa(x_k^{(i)}, x_l^{(i)})$$

$$= \frac{1}{N_i^2} sum\ of\ entries(B_{ii})$$

Expected distance from the center of mass of class i: (3.5)

$$\frac{1}{N_i}\sum_{s=1}^{N_i}\left\|\phi(x_s^{(i)}) - \phi_s^{(i)}\right\|^2 = \frac{1}{N_i}\sum_{s=1}^{N_i}\kappa(x_s^{(i)}, x_s^{(i)})^2 + \frac{1}{N_i^2}\sum_{k,l=1}^{N_i}\kappa(x_k^{(i)}, x_l^{(i)}) - \frac{2}{N_i^2}\sum_{k,s=1}^{N_i}\kappa(x_s^{(i)}, x_k^{(i)}) = \frac{1}{N_i}\sum_{s=1}^{N_i}\kappa(x_s^{(i)}, x_s^{(i)})^2 - \frac{1}{N_i^2}\sum_{k,l=1}^{N_i}\kappa(x_k^{(i)}, x_l^{(i)}) = \frac{1}{N_i} sum\ of\ diagonal\ entries(B_{ii}) - \frac{1}{N_i^2} sum\ of\ entries(B_{ii})$$

For Gaussian kernel,

Distances between the centers of mass of class *i* and class *j:* (3.6)

$$\left\|\phi_S^{(i)} - \phi_S^{(j)}\right\|^2 = \left\|\frac{1}{N_i}\sum_{k=1}^{N_i}\phi\left(x_k^{(i)}\right) - \frac{1}{N_j}\sum_{l=1}^{N_j}\phi\left(x_l^{(j)}\right)\right\|^2$$

$$= \frac{1}{N_i^2}\sum_{k,l=1}^{N_i}\kappa(x_k^{(i)}, x_l^{(i)}) + \frac{1}{N_j^2}\sum_{k,l=1}^{N_j}\kappa(x_k^{(j)}, x_l^{(j)}) - \frac{2}{N_i N_j}\sum_{k=1}^{N_i}\sum_{l=1}^{N_j}\kappa(x_k^{(i)}, x_l^{(j)}) = \frac{1}{N_i^2} sum\ of\ entries\left(B_{ii}\right) + \frac{1}{N_j^2} sum\ of\ entries\left(B_{jj}\right) - \frac{2}{N_i N_j} sum\ of\ entries\left(B_{ij}\right)$$

Define $\mathrm{R}(*)$ as the ratio of average expected distances from the center of mass to the average distances between the centers of mass. (3.7)

$$\mathrm{R}(*) = \frac{\frac{1}{L}\sum_{i=1}^{L}\left(\frac{1}{N_i}\sum_{s=1}^{N_i}\left\|\phi(x_s^{(i)}) - \phi_s^{(i)}\right\|^2\right)}{\frac{1}{L^2}\sum_{i=1}^{L}\sum_{j=1}^{L}\left\|\phi_S^{(i)} - \phi_S^{(j)}\right\|^2} = \frac{\frac{1}{L}\sum_{i=1}^{L}\left(\frac{1}{N_i}\sum_{s=1}^{N_i}\kappa(x_s^{(i)}, x_s^{(i)})^2 - \frac{1}{N_i^2}\sum_{k,l=1}^{N_i}\kappa(x_k^{(i)}, x_l^{(i)})\right)}{\frac{1}{L^2}\sum_{i=1}^{L}\sum_{j=1}^{L}\left(\frac{1}{N_i^2}\sum_{k,l=1}^{N_i}\kappa(x_k^{(i)}, x_l^{(i)}) + \frac{1}{N_j^2}\sum_{k,l=1}^{N_j}\kappa(x_k^{(j)}, x_l^{(j)}) - \frac{2}{N_i N_j}\sum_{k=1}^{N_i}\sum_{l=1}^{N_j}\kappa(x_k^{(i)}, x_l^{(j)})\right)} = \frac{\frac{1}{L}\sum_{i=1}^{L}\left(\frac{1}{N_i} sum\ of\ diagonal\ entries(B_{ii}) - \frac{1}{N_i^2} sum\ of\ entries(B_{ii})\right)}{\frac{1}{L^2}\sum_{i=1}^{L}\sum_{j=1}^{L}\left(\frac{1}{N_i^2} sum\ of\ entries\left(B_{ii}\right) + \frac{1}{N_j^2} sum\ of\ entries\left(B_{jj}\right) - \frac{2}{N_i N_j} sum\ of\ entries\left(B_{ij}\right)\right)} = \frac{\frac{1}{L}\sum_{i=1}^{L}\left(\frac{1}{N_i} sum\ of\ diagonal\ entries(B_{ii}) - \frac{1}{N_i^2} sum\ of\ entries(B_{ii})\right)}{\frac{1}{L}\sum_{i=1}^{L}\left(\frac{2}{N_i^2} sum\ of\ entries\left(B_{ii}\right)\right) - \frac{1}{L^2}\sum_{i=1}^{L}\sum_{j=1}^{L}\left(\frac{2}{N_i N_j} sum\ of\ entries\left(B_{ij}\right)\right)}$$

We aim to minimize $\mathrm{R}(*)$.

3.2 Implementation Steps to Solve Classification Problems

1) We may preprocess the data by feature scaling (Section 1.6).

2) We choose a kernel or construct a valid kernel based on the data pattern or other standards. (Section 1.3)

3) We may divide the data set into three parts: the searching set, the training set, and the test set.

4) In the searching set, we find the optimal parameters in the chosen kernel using Method I or Method II and gradient descent to achieve better separation.

5) In the training set, we use multi-class support vector machine classification algorithms to find hyperplanes that separates different clusters (Section 1.5).

6) In the test set, we will test the performance and calculate F_1 score (Section 1.7) of our model.

Alternatively, we may divide the data set into two parts: the training set and the test set, where the training set are used to find both the parameters and the hyperplanes.

3.3 Experiment

3.4 Strengths and Weaknesses of the Methods

1)Strengths:

These methods consider the differences in size among clusters. The prediction accuracy of the data in classes with smaller size are supposed to be high since these classes are also well-separated from other classes and each class is well- centered. These methods could be used in some other kernels of one parameter.

2)Weaknesses:

If the classes have similar size, it would be better to use the existing method in Section 2 since in these cases, our proposed methods cost more computational time and have little improvement in separation. In addition, there should be ways, either statistically or mathematically, to determine the convexity of $\mathrm{J}(*)$ and $\mathrm{R}(*)$. Moreover, abandoning some data to reduce the size of large classes and use the existing method in Section 2.2 may outperform these methods.

3.5 Further Study

The article does not prove the function $\mathrm{J}(*)$ and $\mathrm{R}(*)$ is convex. If the equation is convex for some kernels, we can use gradient descent method to find the optimal parameters. In addition,

intuitively, Method I and Method II contain some similarities but this report do not contain the comparisons between them. This article does not do experiments and evaluate the performance of these methods using data. Further study may compute the CPU time and F1 scores of our proposed methods and compare our methods with the existing method. In summary, further study could explore more about the convexity of $J(\sigma)$ and $R(\sigma)$, the relationship between Method I and Method II, and the performance of these two methods in experiment.

4. References

Bhandari, A. (2020). Feature scaling for machine learning: understanding the difference between normalization vs. standardization. *Analytics Vidhya*. Retrieved from: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/

Goyal, C. (2021). Multiclass classification using SVM. *Analytics Vidhya*. Retrieved from: https://www.analyticsvidhya.com/blog/2021/05/multiclass-classification-using-svm/

Li, C.H., Lin, C.T., Kuo, B.L.&Chu, H.S. (2010). An Automatic Method for Selecting the Parameter of the RBF Kernel Function to Support Vector Machines. *2010 IEEE International Geoscience and Remote Sensing Symposium*. Retrieved from: https://ieeexplore.ieee.org/document/5649251/authors#authors

Li, Z.X. (2018). Kernel Method. *YouTube*. Retrieved from: https://www.youtube.com/watch?v=p4t6O9uRX-U&list=PLt0SBi1p7xrRKE2us8doqryRou6eDYEOy

Ng, A. (2019). Machine Learning. *Coursera*. Retrieved from: https://www.coursera.org/learn/machine-learning

Shawe-Taylor, J. & Cristianini, N. (2020). Kernel Methods for Pattern Analysis. *World Book Publishing Company*.