Brief Study of Classification Algorithms in Machine Learning

A Dissertation Submitted to the Department of Electrical & Electronic Engineering, Uttara University, Dhaka, in Partial Fulfillment of Requirement for the Degree of B.Sc.

SUBMITTED BY

SL.No.	Name	ID	Batch
1	Md.Asraful Islam Sujon	2213171020	38th A
2	Md.Tarek Al Mamun	2213171021	38th A
3	Shahadat Hossain	2213171026	38th A
4	Md.SohelRana	2213171032	38th A

CANDIDATE'S DECLARATION

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree

Authors

SL.No.	Name	ID	Batch
1	Md.Asraful Islam Sujon	2213171020	38th A
2	Md.Tarek Al Mamun	2213171021	38th A
3	Shahadat Hossain	2213171026	38th A
4	Md.Sohel Rana	2213171032	38th A

DECLARATION

I hereby declare that this thesis is based on the results found by the mentioned students under my direct supervision. Materials of work found by other researchers are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Signature of Supervisor

……

Md. Rezaul Karim

Assistant Professor (EEE)

Department of Electrical & Electronic Engineering
Uttara University, Dhaka-1230, Bangladesh

DEDICATED TO
OUR
BELOVED PARENTS

ACKNOWLEDGEMENT

First of all we express all our admiration and devotion to the almighty Allah, the most beneficent who has enabled us to perform this research work and to submit this thesis. We express our profound gratitude to my honorable supervisor Md. Rezaul Karim, Assistant Professor (EEE), Department of Electrical & Electronic Engineering, Uttara University (UU), for his constant direction, constructive criticism and inspiration in pursuing the whole investigation of the present research. Words are always insufficient to express his working capacities and unending enthusiasm for scientific rigorousness for innovative investigations. This always becomes the everlasting source of inspiration for his students. It was really difficult for us to perform our research work at UU without amiable assistance, sincere support, and cordial collaboration from many friendly people of our surroundings.

ABSTRACT

Support Vector Machine (SVM) is a powerful supervised learning algorithm widely used for classification and regression tasks. It is particularly effective in high-dimensional spaces and in cases where the number of dimensions exceeds the number of samples. SVM works by finding an optimal hyperplane that maximizes the margin between different classes, ensuring better generalization to unseen data.

This thesis provides a brief study of SVM, covering its fundamental principles, mathematical formulation, and different kernel functions used to handle non-linearly separable data. The study explores the advantages of SVM, such as its robustness against overfitting and its effectiveness in handling small datasets. Additionally, the limitations of SVM, including computational complexity and sensitivity to parameter selection, are discussed.

Experimental analysis is conducted using real-world datasets to evaluate the performance of SVM compared to other classification techniques. The results demonstrate the impact of different kernel functions and hyperparameter tuning on classification accuracy. The findings highlight the significance of SVM in machine learning applications, making it a valuable tool in various domains, including image recognition, bioinformatics, and finance.

This research aims to provide a clear understanding of SVM and its practical applications, serving as a foundation for further studies and improvements in machine learning algorithms.

TABLE OF CONTENTS

Introduction

This thesis explores and compares popular classification algorithms in machine learning Support Vector Machines (SVM) performance, scalability, and suitability for various real-world tasks.

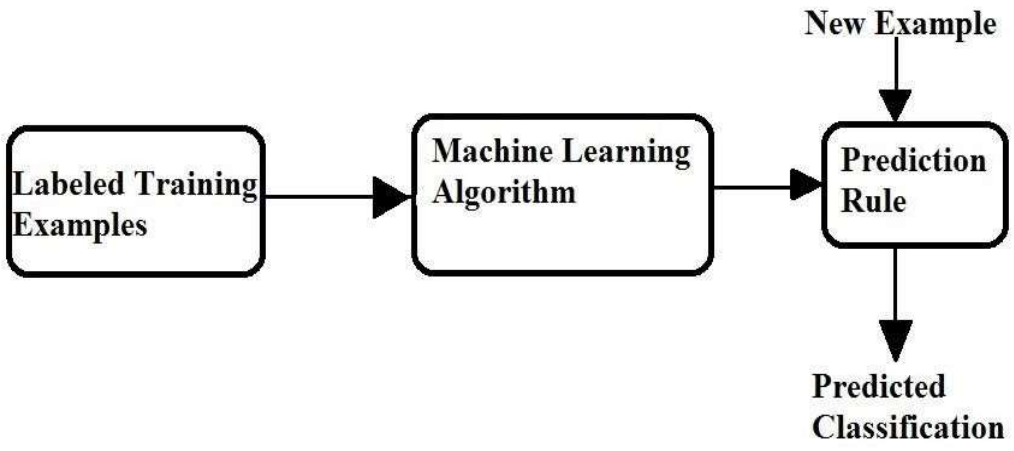


Figure :01: *Diagram of a general Machine Learning Process*

1. Background and Motivation

Machine learning has become a cornerstone of modern technology, driving advancements across various fields including finance, healthcare, and autonomous systems. At the heart of many machine learning applications is the task of classification—a process where the goal is to categorize data into predefined classes or categories. This task is pivotal in numerous real-world applications, such as spam detection in emails, disease diagnosis from medical images, and customer segmentation in marketing.

2. The Role of Classification Algorithms

Classification algorithms are fundamental tools in machine learning, enabling systems to make predictions or decisions based on input data. These algorithms leverage various statistical and computational techniques to analyze and interpret data, allowing for the automatic categorization of information. The effectiveness of these algorithms can significantly impact the performance and accuracy of machine learning models.

3. Importance of Comparative Analysis

Given the diverse range of classification algorithms available, choosing the right one for a specific task can be challenging. Each algorithm comes with its own set of methodologies, strengths, and limitations. Understanding these differences is crucial for selecting the most suitable algorithm based on the characteristics of the data and the problem at hand. This comparative analysis aims to provide clarity on the performance and applicability of several key classification algorithms, thereby assisting practitioners in making informed decisions.

4. Objectives of the Study

This study aims to:

- **Provide an Overview:**Present a detailed description of prominent classification algorithms.
- **Compare Algorithms:** Analyze and compare the methodologies, strengths, and weaknesses of these algorithms to understand their performance across different scenarios.
- **Highlight Practical Applications:** Explore real-world use cases and applications where these algorithms have been successfully implemented.
- **Identify Challenges and Future Directions:** Discuss the limitations of current classification techniques and explore emerging trends and technologies that could influence future developments in the field.

5. Structure of the Thesis

The thesis will be organized as follows: The next section will provide an in-depth overview of various classification algorithms, explaining their operational principles and use cases. Following this, a comparative analysis will be conducted to evaluate the performance of these algorithms based on established criteria. Practical applications and real-world examples will be discussed to illustrate the relevance and impact of these algorithms. Finally, the study willconclude with a summary of findings, recommendations, and a look into future trends in classification algorithms.

Types of Machine Learning Algorithms

1.Supervised Learning

In supervised learning, models are trained on labeled datasets. Each training example is paired with an output label, and the model learns to map inputs to outputs.

(i) Classification Algorithms:

- **Decision Trees:**
  - **Mechanism:** Constructs a tree where each node represents a feature, each branch represents a decision rule, and each leaf represents a class label. The tree splits data based on feature values to classify inputs.
  - **Applications**: Used for tasks like credit scoring, medical diagnosis, and image recognition.
  - **Example:** Predicting whether an email is spam or not based on features like the presence of certain keywords.
- **Support Vector Machines (SVM):**
  - **Mechanism:** Finds the hyperplane that best separates different classes in the feature space. The optimal hyperplane maximizes the margin between different classes.
  - **Applications:** Image classification, text categorization, and bioinformatics.
  - **Example:** Classifying handwritten digits or categorizing news articles into different topics.
- **k-Nearest Neighbors (k-NN):**
  - **Mechanism:** Classifies a new data point based on the majority class among its k-nearestneighbors in the feature space. The choice of k affects the model's performance.
  - **Applications:** Pattern recognition, recommendation systems, and anomaly detection.
  - **Example:** Recommending products based on user similarity or detecting fraudulent transactions.
- **Naive Bayes:**
  - **Mechanism:** Based on Bayes' theorem, it assumes independence between features. It calculates the probability of each class given the features and selects the class with the highest probability.
  - **Applications:** Text classification, sentiment analysis, and spam filtering.
  - **Example:** Classifying movie reviews as positive or negative based on wordmfrequencies.
- **Logistic Regression:**
  - **Mechanism:** Estimates probabilities using a logistic function, which outputs a value between 0 and 1. It is used for binary classification problems.
  - **Applications:** Medical diagnosis, financial forecasting, and marketing.
  - **Example:** Predicting whether a customer will buy a product based ondemographic features.

(ii) Regression Algorithms:

- **Linear Regression:**
  - **Mechanism:** Models the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data.
  - **Applications:** Predicting sales, housing prices, and stock market trends.
  - **Example:** Estimating house prices based on features like size, location, and number of rooms.

- **Polynomial Regression:**
  - **Mechanism:** Extends linear regression by fitting a polynomial equation to capture non-linear relationships in the data.
  - **Applications:** Modeling complex trends in financial data, engineering, and scientific research.
  - **Example**: Predicting temperature changes over time with a polynomial trend.
- **Support Vector Regression (SVR):**
  - **Mechanism:** A variant of SVM used for regression tasks. It tries to fit the best line within a specified margin of tolerance while minimizing errors.
  - **Applications:** Forecasting stock prices, weather prediction, and trend analysis.
  - **Example:** Predicting future sales based on historical data with nonlinear patterns.
- **Ridge and Lasso Regression:**
  - **Mechanism:** Variants of linear regression that include regularization terms to penalize large coefficients and reduce overfitting. Ridge regression uses L2 regularization, while Lasso uses L1 regularization.
  - **Applications:** Handling multicollinearity, feature selection, and improving model generalization.
  - **Example:**Predicting economic indicators while minimizing the risk of over-fitting.

2. Unsupervised Learning

Unsupervised learning algorithms find hidden patterns or intrinsic structures in data without labeled responses.

(i) Clustering Algorithms:

- **k-Means Clustering:**
  - **Mechanism:** Partitions data into k clusters by minimizing the variance within each cluster. The algorithm iterates to find cluster centroids that minimize the distance between data points and centroids.
  - **Applications:** Market segmentation, image compression, and social network analysis.
  - **Example:** Grouping customers based on purchasing behavior to tailor marketing strategies.
- **Hierarchical Clustering:**
  - **Mechanism:** Builds a hierarchy of clusters using either agglomerative (bottom-up) or divisive (top-down) methods. Results in a dendrogram that represents the nested groupings.
  - **Applications:** Gene expression analysis, document clustering, and taxonomy.
  - **Example:** Categorizing plant species based on their genetic similarity.
- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):**
  - **Mechanism:** Groups data points based on density. It finds clusters of varying shapes and sizes by connecting dense regions and identifying outliers as noise.
  - **Applications:** Spatial data analysis, anomaly detection, and image segmentation.
  - **Example:** Identifying clusters of crime incidents in a city or grouping areas of interest in satellite images.

(ii) Dimensionality Reduction Algorithms:

- **Principal Component Analysis (PCA):**
  - **Mechanism:** Reduces data dimensionality by transforming it into a set of orthogonal components (principal components) that capture the most variance.
  - **Applications:** Data visualization, noise reduction, and feature extraction.
  - **Example:** Reducing the number of features in a dataset for visualization while retaining key information.
- **t-Distributed Stochastic Neighbor Embedding (t-SNE):**
  - **Mechanism:** Reduces dimensionality while preserving the local structure of the data. It is particularly useful for visualizing high dimensional data in two or three dimensions.
  - **Applications**: Data exploration, pattern recognition, and feature analysis.
  - **Example:** Visualizing clusters in high-dimensional gene expression data.
- **Linear Discriminant Analysis (LDA):**
  - **Mechanism:** Projects data onto a lower-dimensional space to maximize class separability. Unlike PCA, LDA is supervised and uses class labels to find the best projection.
  - **Applications:** Face recognition, dimensionality reduction in classification tasks, andm pattern recognition.
  - **Example:** Enhancing the performance of a classifier by reducing the feature space while preserving class differences.

(iii) Association Rule Learning:

- **Apriori Algorithm:**
  - **Mechanism:** Identifies frequent item sets in transactional data and generates association rules based on these frequent item sets. It uses a breadth-first search approach.
  - **Applications**: Market basket analysis, recommendation systems, and fraud detection.
  - **Example:** Discovering that customers who buy bread are likely to buy butter as well.
- **Eclat Algorithm:**
  - **Mechanism**: Uses a depth-first search approach to find frequent item sets. It is often more efficient than A priori by using a vertical data format and intersection operations.
  - **Applications:** Market analysis, pattern mining, and recommendation systems.
  - **Example:** Identifying commonly purchased product combinations in retail data.

3. Semi-Supervised Learning

Semi-supervised learning combines a small amount of labeled data with a large amount of unlabeled data to improve model performance.

- **Self-Training:**
  - **Mechanism:** Starts with a model trained on labeled data, then iteratively labels the unlabeled data and retrains the model using the newly labeled data.
  - **Applications**: Text classification, image recognition, and web content classification.
  - **Example:** Enhancing a classifier for sentiment analysis by leveraging additional unlabeled text data.

- **Co-Training:**
  - **Mechanism:** Uses multiple models trained on different feature sets. Each model labels unlabeled data, and these labels are used to train the other models.
  - **Applications:** Document classification, speech recognition, and medical diagnosis.
  - **Example:** Improving a spam filter by combining models that analyze different aspects ofemail content.
- **Multi-View Learning:**
  - **Mechanism:** Utilizes multiple views or representations of the same data to improve learning. Each view provides complementary information that enhances the overall model.
  - **Applications:** Multi-modal data analysis, object recognition, and video classification.
  - **Example:** Combining image and text data for improved image captioning.

4. Reinforcement Learning

Reinforcement learning focuses on training agents to make decisions through interaction with anenvironment. The agent learns to maximize cumulative rewards through trial and error.

- **Q-Learning:**
  - **Mechanism:** A model-free algorithm that learns the value of actions in different states. It updates the Q-values based on the received rewards and the estimated future rewards.
  - **Applications:** Robotics, game playing, and autonomous systems.
  - **Example:** Teaching a robot to navigate a maze by learning from rewards received for reaching the goal.
- **Deep Q-Networks (DQN):**
  - **Mechanism**: Combines Q-learning with deep neural networks to handle high-dimensional state spaces.The neural network approximates the Qvalues for various actions.
  - **Applications:** Complex decision-making tasks, video game AI, and financial trading.
  - **Example:** Playing and mastering Atari games by learning from raw pixel inputs.
- **Policy Gradient Methods:**
  - **Mechanism:** Directly optimizes the policy (strategy) by adjusting it based on the gradient of the expected reward. It allows for more complex policy representations.
  - **Applications:** Continuous control tasks, robotics, and optimization problems.
  - **Example:** Training a robotic arm to perform precise movements by optimizing its control policy.

Support Vector Machine (SVM)

What follows is not an introduction about SVM. It is not my intention to go deep in theory or cover all the details, I will only cover in a quick way the key points about SVM that I may use later. Part of this thesis is about learning SVM (around three weeks). Usually thesis reports offer results about experiments, I will offer what I have learned as a result here. For a good introduction about SVM . What follows is learned from those texts.

If we have a two-class dataset what an SVM will do is finding anhyperplane that will divide the space where the data is. All the points at one side of the hyperplane will belong to one class and all the points at the other side will belong to the other class (multi class problems are usually done one vs all way, it can also solve multi-label problems. That is how SVM classifies.

The equation of anhyperplane is defined as$\langle w.x\rangle + b = 0$

Given this situation we can define two margins: functional and geometric. The functional margin is the value we obtain using in the hyperplane equation a training point where x ismultiplied by his label, the functional margin of a (x,y) point is$y.(\langle w.x\rangle + b)$where the value of y is -1 or +1. The geometric margin is the euclidean distance from any point to the hyperplane.

The critical points are a boundary of the class set. We will focus on them and we will fix the functional margin to 1 or -1 (depending on the class) when applied to a critical point. Critical points are called Support Vectors. Support Vectors are all the information we will need to classify new points. All the other points will not be considered and that means that given a train set with thousands of entries we will only work with a very little set.

The critical points of each class are defining an hyperplane too, so we have two more hyperplanes with functional margin 1 and -1 (the equations of the new hyperplanes will be$\langle w.x^{+}\rangle + b - 1 = 0$ and $\langle w.x^{-}\rangle + b + 1 = 0$ the sign over the x marks the class). There are different generalization bounds, however the most common is to reduce the problem to minimizing the norm of the weight vector (w), by minimizing it what we are achieving is maximizing the geometric margin (Euclidean distance) the most we can given the functional margins of 1 and -1. So now we have an equation (the hyperplane) and a condition (minimizing w) and that defines the problem.

The basic formulation of the problem is what follows:

Given a linearly separable training sample$S = \big((x_1, y_1),,(x_1, y_1)\big)$where "l" is the size of the training set,(x_1, y_1) represents an entry in the training set, where x is the vector that represents that entry and$y$ is its class (as suggested before when working with SVM x can be anything, a vector of reals, an image,...).

The formulation of the problem is:

$$\boldsymbol{minimise}_{w,b}\langle \boldsymbol{w}, \boldsymbol{w}\rangle$$

$$\boldsymbol{subject\ to}\ _{y_i}(\langle \boldsymbol{w}.\boldsymbol{x_i}\rangle + \boldsymbol{b}) \geq \boldsymbol{1},$$

$$\boldsymbol{i} = \boldsymbol{1}, \dots, \boldsymbol{l}$$

What follows now is a sequence of mathematic tricks to make that formulation as easiest to calculate as possible, the objective is to transform the problem to a dual formulation where the solution can be found as a linear combination of the training points.

We mix the objective and the condition in one equation using Lagrangian multipliers, in this way we obtain the primal form:

$$\boldsymbol{L(w, b\ \alpha) = \frac{1}{2}\langle w.w\rangle - \sum_{i=1}^{1} \alpha_1[y_1(\langle w.x_1\rangle + b) - 1]}$$

α are the Lagrange multipliers. If we not differentiate with respect to $w\ and\ b$:

$$\boldsymbol{\frac{\partial L(w.b.\propto)}{\partial w} \rightarrow (w = \sum_{i=1}^{1} y_1 \propto_1 x_1)}$$

$$\boldsymbol{\frac{\partial L(w, b, \propto)}{\partial b} \rightarrow \sum_{i=1}^{1} y_1 \propto_1 = 0}$$

And resubstituting the relations obtained into the primal form,we obtain the dual form

$$\boldsymbol{minimise\ W(\propto) = \sum_{i=1}^{1} \propto_1 - \frac{1}{2}\sum_{i,j=1}^{1} y_i y_j \propto_i \propto_j \langle x_i, x_j\rangle}$$

$$\boldsymbol{subject\ to \sum_{i=1}^{1} y_i \propto_i = 0}$$

$$\boldsymbol{\propto_1 \geq 0, i = 0, \dots, l}$$

The most important thing about this formulation is that now the solution can be described as a linear combination of the training points. A training point is represented as a vector and we are performing an inner product between them; the biggest is the result of the inner product, the most both vectors are pointing to a similar direction, the most similar that two vectors (training points) are.

Classifying documents:

The objective in the introduction was about understanding how SVM works, the theory beyond them. The objective now is to make them work. We will use a library called Shogun, that library has several data mining tools; we will focus on classifiers, specially SVM. That library, among other feature types, wants to receive lists of real numbers with their labels as input, “anything” converted to real numbers would be classified by Shogun. Since in software development one has to explain why some tools were chosen, we will also explain why we chose Shogun:

– Shogun is an alive project (compared to most of the others Data Mining projects), it is continually being updated1 .
– Shogun covers everything about SVM, it is in C++, a quick look at Shogun's code is enough to see that it is high quality done by expert people.
– Shogun's authors are top people in SVM nowadays, they are part of the MKL development.
– The community is alive, the authors are easy to contact and they provide quick help.

– Shogun is free software, they use a GPL v3 license. The code is high quality and this also means that is very easy to add new features.
– The documentation is very good with dozens of examples, each class is very well explained, all the code is well documented (more than 80K lines). Develop to Shogun is easy.
– It is in the Debian repositories. That means an instant installation.

We will work with documents; the documents will be in a raw “machine readable” state, by that we mean that documents will be in what is known as “txt”, each byte will contain a character in (probably) ASCII format. That means that we will not cover the part where documents are converted (or parsed) from PDF, HTML, DOC,... although there are easy ways to do it and the Natural Language Processing (NLP) tool we will use support many of them.

To process the documents we will use a library called NLTK, there are lots of NLP libraries, why we chose NLTK?

– It is implemented in Python, Python is probably with Perl the best programming language to work with documents.
– Shogun works with Python, so if the NLP also works with Python everything will be easier.
– We found that the NLTK tutorial is very good and straightforward and they also have a book to cover it.
– After a quick review, we found that NLTK did everything we need and also provide us with tools we didn't know before.
– The handicap is that Python is slower than, for example, C++, but the code is so easy that it saved by far in coding time all the time c++ in running time could win. Since Python is an interpret language the production cycle is faster.

Can we use any similarity with SVM?

Not all, but most of them are good.
SVM mathematical formulation has two big constraints related one with the other: We are working in a space with inner product (the feature space) and we have to satisfy Mercer's Conditions . So we can say that the similarity function must be a "masked" inner product and that constraints it quite a bit. It has to be an inner product in the usual formulation because it is using an Euclidean space (the feature space; an inner product space).

What we are calling "similarity" is commonly called "proximity index" or "proximity function".

There are two kinds of proximity indexes: similarities$(s_{i,j})$or dissimilarities$(\delta_{i,j})$

As said in a proximity index$(P_{i,j})$has to fulfill the following properties:

1. Non-negativity. The$(P_{i,j})$has to fulfill the following properties :
$$s_{i,j} \geq 0$$
$$\delta_{i,j} \geq 0 \forall \overrightarrow{x_\iota}, \overrightarrow{x_j} \in X$$
2. Symmetry. The$(P_{i,j})$do not depend on the order ofi, j.
$$s_{i,j} = s_{j,i}$$
$$\delta_{i,j} = \delta_{j,i} \forall \overrightarrow{x_\iota}, \overrightarrow{x_j} \in X$$
3. Boundedness. There is a maximum similarity and a minimum dissimilarity.
$$S_{i,j} \leq S_{max}$$
$$\delta_{i,j} \geq \delta_{min} \forall \overrightarrow{x_\iota}, \overrightarrow{x_j} \in X$$
4. Minimality. The extreme values are attained for equal objects, and only for them.
$$S_{i,j} = S_{max} \Leftrightarrow \overrightarrow{x_\iota} = \overrightarrow{x_j}$$
$$\delta_{i.j} = \delta_{min} \Leftrightarrow \overrightarrow{x_\iota} = \overrightarrow{x_j} \forall \overrightarrow{x_\iota}, \overrightarrow{x_j} \in X$$
5. Semantics. The semantic of$S_{i,j} > S_{i,k}$ is that object i is more similar to object$j$ than is to object k
The semantic of$\delta_{i,j} > \delta_{i,k}$ is that object i is more dissimilar to object j than is to object k .

A good thing about similarities is that it is not required for them to be "Positive"$definite$[10](as Kernels are required to), this means that we have a bigger semantic freedom to define what our similarity will do.

About Mercer's condition,says that even for kernels that do not satisfy Mercer's condition (and we don't know if it satisfies or not), one might still find that a given training set results in a positive semidefiniteHessian.

Can SVM work with a similarity function that would return descriptions instead of numbers? It should.

That descriptions should define features and our space has n dimensions (one dimension per feature); so given the description from a similarity function we should be able to allocate it in our space. Once everything is allocated we should be able to draw an hyperplane that would separate it in classes. It is easy to say, it is hard to implement. The core idea is the same that is using SVM: find an hyperplante. The formulation would probably not be the same.

The Advantages of SVM:

- **Based on a strong and nice Theory:**
  -In contrast to previous "black box" learning approaches, SVMs allow for some intuition and human understanding.

- **Training is relatively easy:**
  -No local optimal, unlike in neural network
  -Training time does not depend on dimensionality of feature space, only on fixed input space thanks to the kernel trick.

- **Generally avoids over-fitting:**
  - Tradeoff between classifier complexity and error can be controlled explicitly

  -eneralize well even in high dimensional spaces under small training set conditions. Also it is robust to noise.

The Drawbacks of SVM:

- It is not clear how to select a kernel function in a principled manner.
- What is the right value for the "Trade-off" parameter "C":
  - We have to search manually for this value, since we don't have a principled way for that.
- Tends to be expensive in both memory and computational time, especially for multiclass problems :
  - This is why some applications use SVMs for verification rather than classification. This strategy is computationally cheaper once SVMs are called just to solve difficult cases.

MATLAB IMPLEMENTATION FOR SVM

Background on Machine Learning: Machine learning has become a cornerstone of modern data analysis, enabling systems to identify patterns and make predictions based on empirical data. This rapidly evolving field encompasses various algorithms and methodologies, each tailored to address specific types of problems. Among these, classification algorithms are essential for tasks that require categorizing data into discrete classes, such as spam detection, image recognition, and medical diagnosis. As data continues to grow in volume and complexity, effective classification methods are increasingly critical.

Support Vector Machine Overview: Support Vector Machine (SVM) is a powerful supervised learning algorithm widely used for classification and regression tasks. Introduced in the early 1990's, SVM has gained popularity due to its effectiveness in high-dimensional spaces and its ability to model complex relationships through the use of kernel functions. The primary goal of SVM is to find the optimal hyperplane that maximizes the margin between different classes, making it robust against overfitting, particularly in scenarios with limited training data. SVM's flexibility, through various kernel options, allows it to tackle both linear and non-linear classification problems effectively.

Importance of MATLAB: MATLAB is a preferred tool for researchers and practitioners in the field of machine learning due to its user-friendly environment and powerful computational capabilities. The Statistics and Machine Learning Toolbox in MATLAB offers extensive functions that simplify the implementation of SVM, making advanced machine learning techniques accessible to a broader audience. By leveraging MATLAB's capabilities, this thesis aims to provide a clear and practical approach to implementing SVM, facilitating its application in real-world problems.

SVM Implementation in MATLAB:

Step 1: Load or Generate Data

You can either load an existing dataset or generate a simple synthetic dataset. Here's an example using the built-in iris dataset:

```
% Convert species to numeric labels
Y_num = grp2idx(Y); % Converts 'setosa', 'versicolor', 'virginica' to 1,2,3
```

Step 2: Convert Labels to Numeric Values

SVM requires numeric labels, so convert the categorical labels into numeric form:

```
% Load the iris dataset

Load fisheriris;

%Use the first two features for simplicity

X= meas(:,1:2); % Features (sepal length and sepal width)

Y= species;      % Labels (species of iris)
```

Step 3: Split the Data into Training and Testing Sets

It's common to split the data into training and testing sets:

```
%Split data (70% train, 30% test)

Cv = cvpartition(Y_num, 'Holdout', 0.3);

% Training data

X_train = X(~idx, :);

Y_train = Y_num(~idx);

% Testing data

X_trst = X(idx, :);

Y_test = Y_num(idx);
```

Step 4: Train the SVM Model

Now, train the SVM model using the training data:

```
% Train the SVM model using a linear kernel

SVNModel = fitcsvm(X_train, Y_train, 'KernelFunction', 'linear');
```

Step 5: Make Predictions

Use the trained model to make predictions on the test data:

```
% Make predictions on the test set

Prediction = predict(SVMModel, X_test);
```

Step 6: Evaluate the Model

You can evaluate the model's performance using confusion matrix and accuracy:

```
% Calculate accuracy

Accuracy – sum(prediction == Y_test) / length(Y_test);

Fprintf('Accuracy: %.2f%%\n', accuracy * 100);


% Visualize the results (optional)

gscatter(X_test(:,1), X_test(:,2), Y_test);

hold on;

gscatter(X_test(:,1), X_test(:,2) predictions, 'rbg', 'o', 8);

title('SVM Classification Results');

xlabel ('Sepal Length');

ylabel('Sepal Width');

legend('True Class', 'Predicted Class');

hold off;
```

Explanation of the Code:

1. **Data Preparation:** The code first loads the iris dataset and selects the first two features for simplicity. It converts the categorical labels into numeric values.
2. **Data Splitting:** It splits the dataset into training (70%) and testing (30%) sets.
3. **Model Training:** The SVM model is trained using a linear kernel.
4. **Prediction and Evaluation:** Predictions are made on the test set, and accuracy is calculated. A confusion matrix is also displayed for a better understanding of the model's performance.

SVM and supporting techniques

1990's SVM theory had been developed by Vapnik and his group. SVM concept was inherited from neural network or may be saying that SVM is mathematical extension of Neural Network. SVM can classify both linear and nonlinear data. SVM performs classification by transforming the original training data into multidimensional space and constructing hyper-plane in higher dimensional (Fig. 2). SVM is an optimal hyper-plane based mathematical learning scheme .

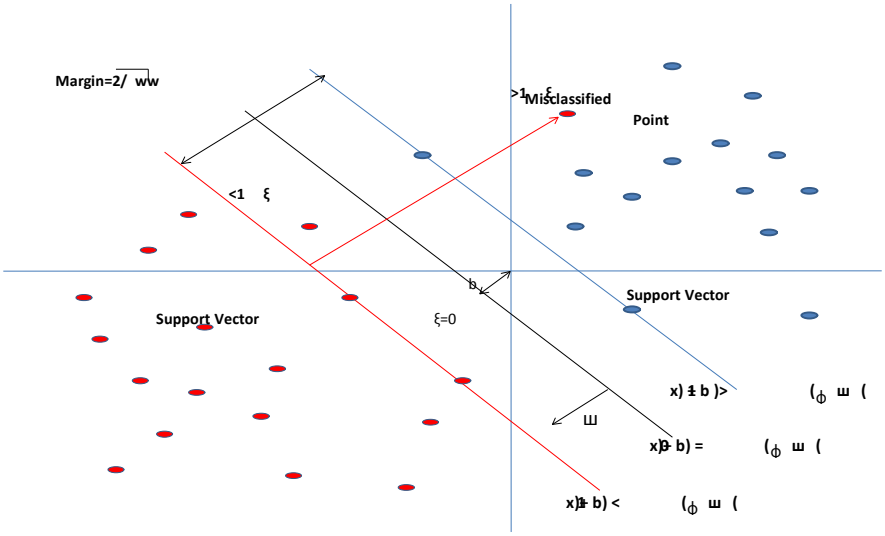


Figure :02: SVM separable problem in 2-dimensional space

An SVM performs classification by constructing a hyper-plane in higher dimensions. This hyper-plane refer as decision planes shown in Fig. 3. A decision plane is one that distinguishes a group of data of one type from another type. A suitable nonlinear mapping for higher dimensional space always classifies a data by constructing a hyper-plane. SVM search those vector points, refer as support vector, which define the decision boundary and give the large marginal separation between the classes. SVM separates the classes with maximum marginal distance in the decision plane.

Middle line represents the maximum-margin hyperplane. In other words, one would select boundary line that separates the two classes at a maximal distance to the closest data point. For two class problem, decision boundary or hyper-plane for classification is defined by Vapnik's theory. If input set is x_i is m ndimension and corresponding class is $y_1 \in \{-1, +1\}$.

$w^t \emptyset(x_i) + b$ .....(1)

The hyperplane is built to such that it fulfills the following inequality function for both the classes.

$w^t \emptyset(x_i) + b \geq +1 \qquad y_i = +1$ .....(2)

$w^t \emptyset(x_i) + b < +1 \qquad y_i = -1$ .....(3)

By using Eq 1 and Eq2 we get Eq4 .

$y_i(w^t \emptyset(x_i) + b) \geq +1 \qquad i = 1 \ldots \ldots N$ .....(4)

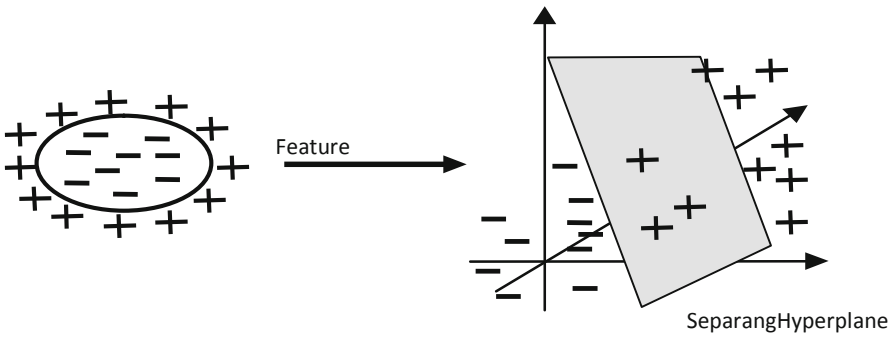


Figure :03 Decision space

Margin is the smallest perpendicular distance to the data point from the hyper-plane. The Maximum marginal hyperplane is the decision plane where the margin is largest. SVM selects the maximum margin separating hyper-plane. Maximum marginal hyper-plane selection done by SVM, increase the ability of accurate classification and reduces the chances of misclassification . Two hyper-planes D1 and D2 are shown in the Fig. 03. Figure shows that hyperplane D1 is better than hyper-plane D2 because it maximizes the margin. For analyzing the Fig. 3 the accepting hyper-plane is D1, if any, data points that lying on the wrong side or within the margin of correct side of the hyper-plane. To resolve such type of cases, the SVM algorithm has introduced the new term known as 'soft margin'. Data has noise and outliers this problem is solved in different ways by different flavors of SVM; here slack variable concept is used: Without ''slack variable'':

$y_i(w^t x_i + b) \geq 1$ ……(5)

With "slack variable":$S_i \geq 0 \; And \; Allow$

$y_i(w^t x_i + b) \geq 1 - S_i$ ……(6)

Now considering the convex optimization problem

$min_{w,b,s} f(w,b,f) = \frac{1}{2} w^t w + c \sum_i S_i$. ……..(7)

Lagrange gives the new terms known as saddle point which gives the solution of optimization problem represented in Eq. (8)

$y(x) = \text{sign}[\sum_{i=1}^{n} \alpha_i y_i k(x_i x) + b]$ ……(8)

Where $K(x_i x) = \emptyset(x_i)^T \emptyset(x)$is a kernel function that must satisfy the Mercer Theorem.

One of the best concepts used in SVM is kernel function. The kernel function is a solid mathematical method used for nonlinear mapping for higher dimensional data. Through this SVM can solve the higher dimensional

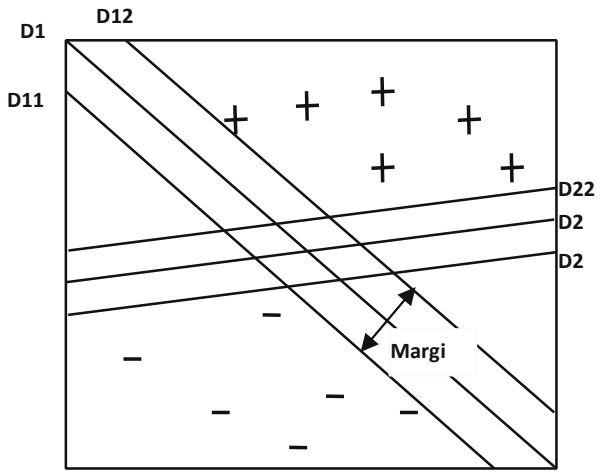


Figure :04: Maximum margin hyper-plane

Classification of set of originally input data space. In general, it calculates the value of the dot product mapped data point into feature space. The benefit of kernel function is that the complexity of the problem only dependent on the dimensionality of input space rather than feature space. Some of popular kernel functions are.

1. Linear Kernel Function: Simple kernel function known as Linear kernel function.

$K(x.x') = x.x^T$ ……(9)

Where x^T is a transpose of the input matrix x 'Lin' keyword used in MATLAB for implementing the linear function. If we have too many feature, then over fitting can happen, this is the drawback of this Linear Kernel Function.

2. Polynomial Kernel Function: It is a non-linear kernel function. It is directional that means output depends on the direction of input in a low dimensional space. This is due to the dot product in the kernel

$K(x, x_i) = (1 + x.x_i^T)^P$ ……(10)

Where, 'P' is the degree of kernel function. 'Poly' keyword used in MATLAB for implementing the Polynomial Function.

3. Radial Basis Kernel Function: Radial basis kernel function is widely used with SVM. Its select the solution those are smooth.

$K(x, x_i) = e^{-\gamma x - x_i^2} \gamma > 0$ ……(11)

Where, γ is a parameter that sets the spread of the kernel.'rbf' keyword used in MATLAB for implementing the Radial Basis Kernel Function.

4. Sigmoid Function: The hyperbolic tangent function known as a Sigmoid Function.

$K(x, y) = \tanh(ax^T y + c), a, c > 0$ ……(12)

Wavelet

Wavelet is a powerful mathematical function used to analyze data and extract information from different data set. The term wavelet comes from the fact that they integrate to zero; they wave up and down across the axis. In 1909 Wavelet first referenced by Alfred Haar on his Ph D Thesis titled ‘‘On the theory of the orthogonal function systems’’.In the early 1980’s, David Marr use wavelet concept in artificial vision for robots.M A Chandra uses wavelet for hiding data in video. Victor Wicker Hauser uses the wavelet for compression of sound. Wavelet theory has been applied to several subjects. Wavelet transform are now being adopted by a number of areas of signal processing, DNA analysis, protein analysis, image processing, data hiding, and so on.

In recent years, wavelet theory developed rapidly in every field of computer science. Mother Wavelet can be formed as eq. 13

$$\omega_{a,b}(x) = |a|^{1/2}\omega(\frac{x-b}{a}) \quad \text{.....(13)}$$

Where, ‘a’ is the dilatation factor and b is the translation factor. Simplest wavelet is haar wavelet, which is widely used in signal, image analysis. It’s dilation and translations of the function is

$$w_{a,b}(x) = 2^{a/2}\omega(2^a x - b) \quad \text{.....(14)}$$

The haar function is pairwise orthogonal in $L^2(R)$. The haar2×2 matrix is given by

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{.....(15)}$$

Due to the orthonormal property, the wavelet coefficients of a signal f(x) can be easily computed via Eq. 16

$$C_{a,b} = \int_{-\infty}^{+\infty} f(x)\omega a, b(x)dx \quad \text{.....(16)}$$

And the synthesis formula

$$f(x) = \sum_{a,b} C_{a,b}\omega a, b(x) \quad \text{.....(17)}$$

Equation no (17) can be used to recover f(x) from its wavelet coefficients. To construct the mother wavelet $\omega(x)$ we may first determine a scaling function(x).

$$\varphi(x)\sqrt{2}\sum_k h(k)\varphi(2x-k) \quad \text{.....(18)}$$

Then, the wavelet kernel wðxÞ is related to scaling function via

$$\omega(x) = \sqrt{2}\sum_k g(k)\varphi(2x-k) \quad \text{.....(19)}$$

Where

$$g(k) = (-1)^k h(1-k) \quad \text{.....(20)}$$

Wavelet transform generate the new possibility to extract image features at different scales. Discrete Wavelet Transform (DWT) is another strategy which is utilized to extricate the wavelet coefficients from various mother wavelet functions. DWT decomposed images into four sub bands as shown in Fig. 5 a, b. These sub bands labeled LH1, HL1 and HH1 represent the finest scale wavelet coefficients i.e., detail

images while the sub-band LL1 corresponds to coarse level coefficients i.e., approximation image. Figure 5b represents the Three Scale DWT.

Wavelet is broadly used in analysis of image components, feature extraction, image segmentation, image compression, removing the noise, etc. Wavelet transformation is not well explored in image classification but extract image features at different scales which have more discriminative capability than the original spectral feature Wavelet has a powerful mathematical model for extracting discriminative features for high dimensional data set. A discriminative feature of the data is expressed by few wavelet coefficients WT has been applied to a number of problems regarding data analysis and feature extraction.

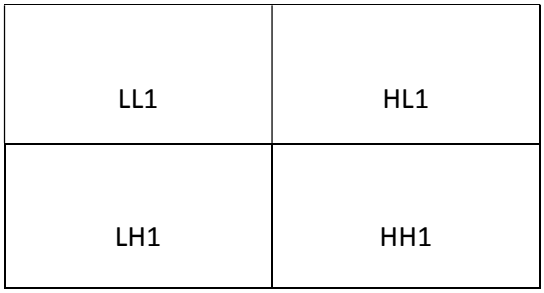


Figure 5 (a)

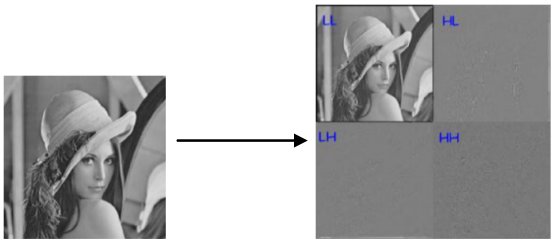


Figure 5 (b)

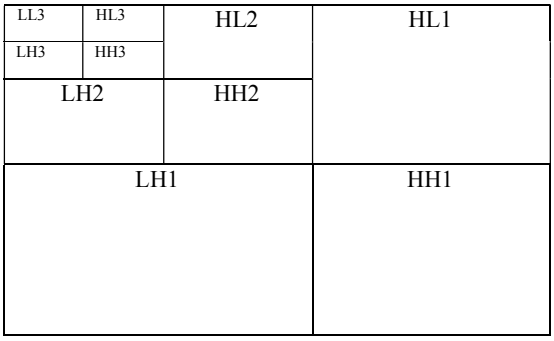


Figure 5 (c)

Fig. 5 a Image Decomposition. b Original lena image, lena image after wavelet decomposition. c Image decomposition different scale DWT.

Equation of a hyperplane

In coordinate space R^n equation

$$(\omega, x) + b = 0$$

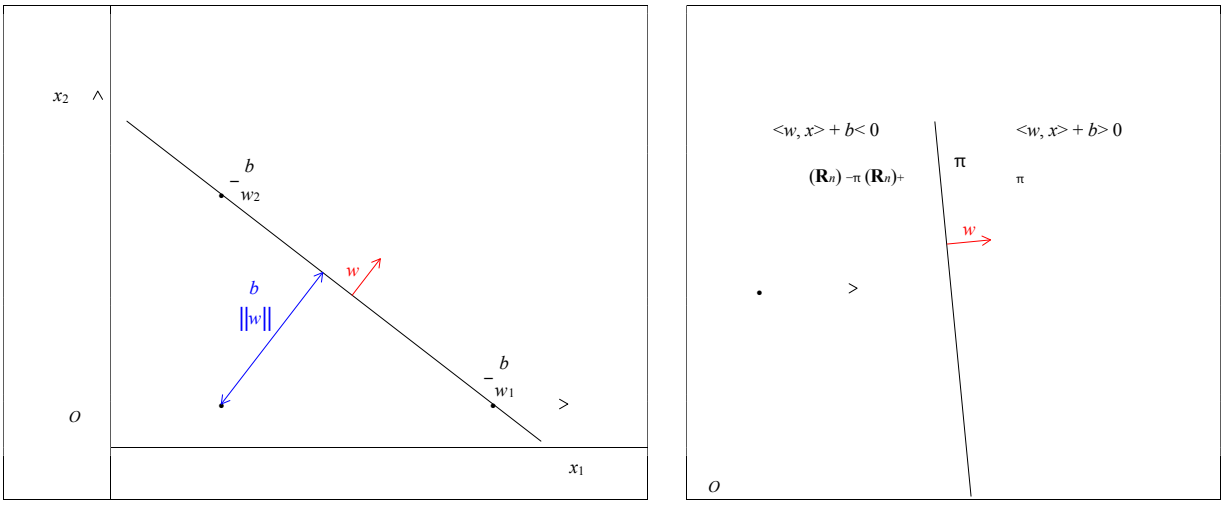
$$\sum_{k-1}^{n} w_k x_k + b = 0$$

Defines a $(n-1)$-dimensional set of vectors called hyperplane. That is, for a given nonzero vector $\omega = (\omega_1, \omega_2, \dots, \omega_n) \in R^n$ and a scalar$b \in R$, the set of all vectors $x = (x_1, x_2, \dots, x_n) \in R^n$ satisfying equation forms a hyperplane (Figure 6a). We will denote this hyperplane by letter π or by$\pi(\omega, b)$.

The term "hyperplane" means that the dimensionality of the plane is by one less than the dimensionality of the entire spaceR^n. For example, a point is a hyperplane in R; a line is a hyperplane inR^2; a plane is a hyperplane in R^3; a three-dimensional space is a hyperplane in R^4, and so on.

Vector $\omega$is called the *normal vector* of the hyperplane, and number $b$is called the *intercept* of the hyperplane. The normal vector defines the orientation of the hyperplane in space, while the ratio between $\|\omega\|$and b(not the intercept alone) defines the distance between the hyperplane and the origin of space[1]. The normal vector is perpendicular to all vectors parallel to the hyperplane. That is, if $z = x_1 - x_2$such that $\langle \omega, x_1 \rangle + b' = 0$ and $\langle \omega, x_2 \rangle + b' = 0$for some b'then $\langle \omega, z \rangle = 0$.

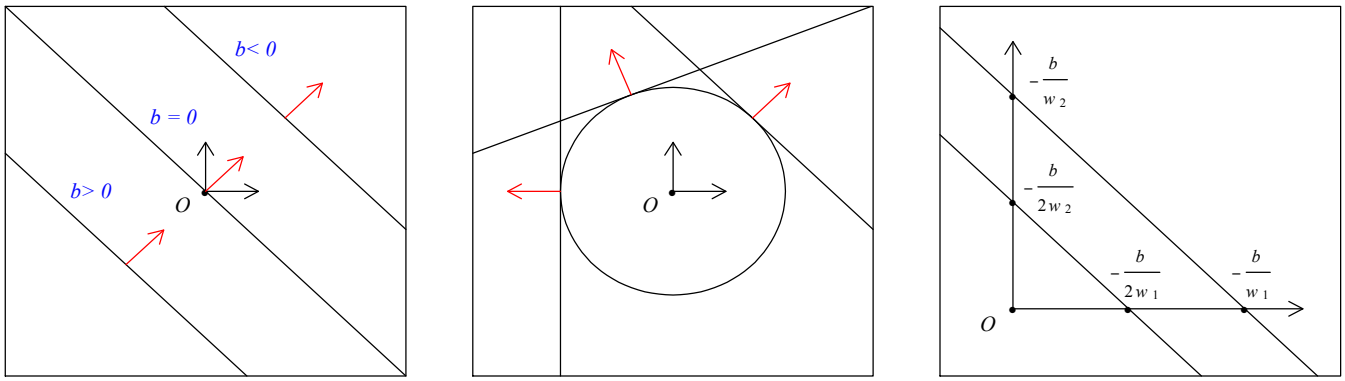
Hyperplane π divides coordinate space R^ninto two parts located sidewise of the hyperplane, called positive and negative *half-spaces*: $(R^n)^+_\pi$and $(R^n)^-_\pi$. The positive half-space is pointed by the normal vector of the hyperplane (Figure 6b). For any vector $x \in (R^n)^+_\pi$we have $\langle \omega, b \rangle + b > 0$, while for any $x \in (R^n)^-_\pi$ we have $\langle \omega, b \rangle + b > 0$.



(a) (b)

[1] The origin of space is zero vector **0** = (0,0,…,0); in figures, we denote it by capital letter *O*.

Figure: 6 (a) Hyperplane in two-dimensional space R^2 is a line.For the depicted line we can infer that $b < 0, \omega_1 > 0, \omega_2 > 0$. (b) Negative and positive half-spaces defined by hyperplane $\pi(\omega, b)$.



(a) (b) (c)

Figure: 7 Understanding the meaning of hyperplane parameters ω and b(see text).

A single hyperplane is defined by infinite number ofparameters ω, b. Indeed, multiplying equation by arbitrary constant $c \neq 0$ we see that parameters $c\omega, cb$ define the same $hyperplane^2$. In other words, if two hyperplanesπ_1 and π_2 are defined by parameters $\omega_1, b_1$and ω_2, b_2, and$\omega_2 = c\omega_1, b_2 = cb_1$ then π_1 and π_2 are the same hyperplanes. Since we can arbitrary scale parameters ω, b defining fixed hyperplane π, we can choose ω, bsuch that $\|\omega\| = 1$. Note that this pair of parameters is unique for any $hyperplane^3$.

The distance $\rho(x, \pi)$ between a vector x and a hyperplane $\pi(\omega, b)$ can be calculated according to the following equation:

$$\rho(x, \pi) = \frac{\langle \omega, x \rangle + b}{\|\omega\|}.$$

Note that this is a signed distance: $\rho(x, \pi) > 0$when $x \in (R^n)^+_\pi$, and $\rho(x, \pi)$when$x \in (R^n)^-_\pi$, . Obviously, $\rho(x, \pi) = 0$ when$x \in \pi$ .If$\|\omega\| = 1$,then the equation for the distance is simply $\rho(x, \pi) = \langle \omega, x \rangle + b$.

Hyperplane separating two classes Margin

We say that a hyperplane$\pi(\omega, b)$separates two classes (sets) of vectors C_1 and C_2 if either

$$\langle \omega, x \rangle + b > 0, \forall x \in C_1$$

$$\langle \omega, x \rangle + b < 0, \forall x \in C_2$$

Or …….(21)

$$\langle \omega, x \rangle + b < 0, \forall x \in C_1$$

$$\langle \omega, x \rangle + b > 0, \forall x \in C_2$$

Two classes are called linearly separable if there exists at least one hyperplane that separates them. If hyperplane$\pi(\omega, b)$ separates classes C_1 and C_2 according to Eq.21thendecision function

$$f(x) = sgn\{\langle \omega, x \rangle + b\} = \begin{cases} 1, & if\ \langle \omega . x \rangle + b \geq 0 \\ -1, & if \langle \omega, x \rangle + b < 0 \end{cases} \quad \text{…….(22)}$$

Gives us a classifier that correctly classifies all vectors from$C_1$and C_2.

It is clear that for two linearly separable classes that are finite there always exist an infinite number of hyperplanes (with differently oriented ω and different b) that separate them. Which one should be used to define a classifier Support vector machine chooses the one with the maximum margin. For a hyperplaneπ separating classes $C_1$and C_2, its margin $m(\pi, C_1, C_2)$is defined as the distance between π and class C_1, plus the distance between π and class C_2 (Figure 8a):

$$m(\pi, C_1, C_2) = \rho(\pi, C_1) + \rho(\pi, C_2)$$

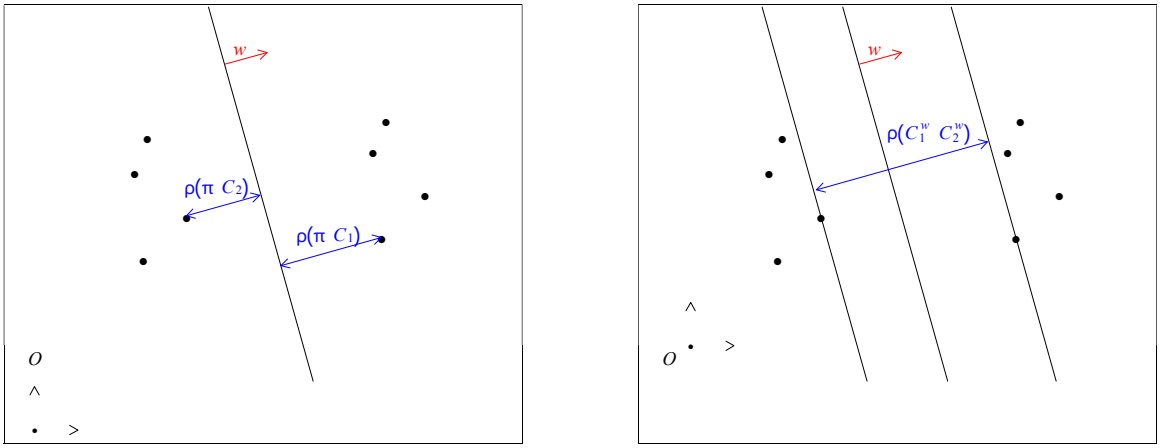
Here, the distance between hyperplaneπ and a set of vectors C is defined as the minimal distance between π and vectors from C:

$$m(\pi, C_1, C_2) = \rho(C_1^{\omega}, C_2^{\omega}) .$$

Note that in this definition we are using the absolute value of the signed distance $\rho(\pi, x)$ defined by equation eq.22, in order for this definition to make sense when all, or some, vectors from C lie in the negative half-space of π.

Equivalently, the margin can be defined as the distance between classes C_1 and C_1measured along the normal vector ω (Figure 8b). If C_1^{ω} is the set containing projections of all vectors from C_1 onto the line parallel to vector ω, andC_2^{ω} is the set containing similar projections of all vectors from C_2, then

$$m(\pi, C_1, C_2) = \rho(C_1^{\omega}, C_2^{\omega}) .$$



(a) (b)

Figure: 8 (a) Margin of hyperplane π is the distance from π to the first class (white points) plus the distance from π to the second class (black points). (b) Equivalently, it can be defined as the distance between two classes measured along the normal vector ω of the hyperplane.

Where,

$$\rho(C_1^{\omega}, C_2^{\omega}) = \min \rho\,(x_1, x_2)$$

$$x_1 \in C_1^{\omega}$$

$$x_2 \in C_2^{\omega}$$

Note the following two properties of margin. First, as long as the hyperplane lies between classes $C_1$and C_2, its margin only depends on the normal vector ω, and does not depend on the intercept b. Second, for any hyperplane separating two classes, its margin can not be larger than the distance between the classes:

$$m(\pi, C_1, C_2) \leq \rho(C_1, C_2)$$

Where,

$$\rho(C_1, C_2) = \min \rho\,(x_1, x_2)$$

Clearly, there exist infinitely many hyperplanes with the maximum margin equal to $\rho(C_1, C_2)$, all of which are perpendicular to the shortest line segment connecting C_1 andC_2 (Figure 9).

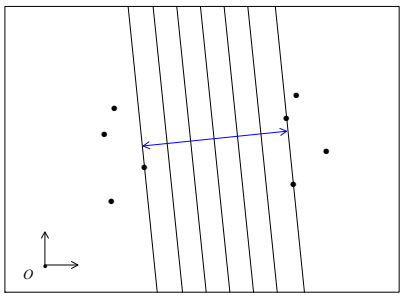


Figure: 9 All five hyperplanes shown here have the same margin equal to $\rho(C_1, C_2)$.

Maximum margin hyperplane for linearly separable classes

Suppose we have two linearly separable classes of training vectors. Support vector machine defined on such a training set is a classifier with decision function

$$f(x) = sgn\{[\omega, x] + b\}$$

Where $\langle \omega, x \rangle + b$ is an equation of hyperplane that separates the two classes , has maximum margin, and is equidistant from both classes (Figure 10). In this section we consider an optimization problem which is being solved in order to obtain parameters ω, b of this hyperplane, and explain where this problem comes from. We also consider important properties of the maximum margin hyperplane. Some concepts from calculus and optimization theory that are used in this section are briefly reviewed in Appendix.

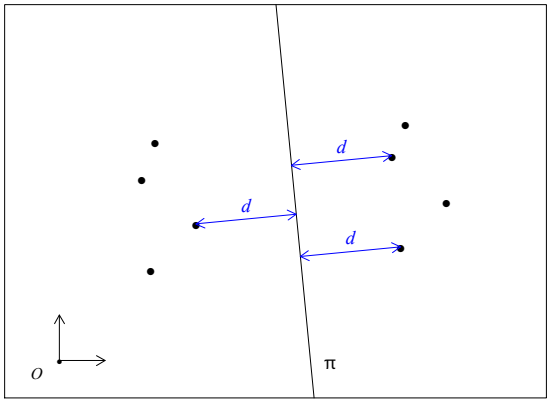


Figure: 10 Maximum margin hyperplane for two linearly separable classes: $d = \rho(\pi, C_1) = \rho(\pi, C_2)$ is maximized.

(i) Primary optimization problem

Parameters ω, b of the SVM hyperplane can be found as a solution to the following optimization problem:

$$\frac{1}{2}\|\omega\|^2 \to \min_{\omega, b} \quad \text{.....(23)}$$

Subject to

$$\langle \omega, x \rangle + b \geq 1, \forall_x \in C_1 \quad \text{.....(24)}$$

$$\langle \omega, x \rangle + b < -1, \forall_x \in C_2 \quad \text{.....(25)}$$

Where C_1 *and* C_2 are two classes of training examples.

In coordinate form, this problem is written as

$$\frac{1}{2}\sum_{k=1}^{n} \omega_k^2 \to \min_{\omega, b} \quad \text{.....(26)}$$

s.t.

$$\sum_{k=1}^{n} \omega_k x_k + b \geq 1, \forall_x \in C_1 \quad \text{.....(27)}$$

$\sum_{k=1}^{n} \omega_k x_k + b \leq -1, \forall_x \in C_2$ .....(28)

Note that parameter b is one of the optimization variables, although it is not present in the objective. Two sets of constraints (26), (27) can be written in a unified way as

$y_i(\sum_{k=1}^{n} \omega_k x_{ik} + b) \geq 1, \ i = 1,2, \dots, l$ .....(29)

where $y_i = 1 \ if \ x_i \in C_1$, and $y_i = -1 \ if \ x_i \in C_2$, l is the total number of training vectors x_i , and by x_i we denote the $k - th$ component of vector x_i .

Optimization problem (25)-(27) has quadratic objective function (25) and linear constraints (26), (27). Such problems are called quadratic programming problems. Their properties are well known and there are quite efficient algorithms for solving these problems.

Note that objective function (25) is strictly convex (since the matrix of its second order derivatives – the Hessian – is positive definite), and the feasible region defined by linear inequalities (26)-(27) is also convex. Therefore, this problem will have a unique solution (global minimum) ω^*, b^{*4} . In case when two classes are not linearly separable, the feasible region defined by constraints (26)-(27) will be empty, and the problem will have no solution. It will also have no solution when the training set contains only one class.

Why parameters of the maximum margin hyperplane can be found by solving problem (22)-(24)? To answer this question, let us transform this problem into an equivalent one that has more apparent geometrical interpretation. First, minimizing $\frac{1}{2}\|\omega\|^2$is equivalent to minimizing $\|\omega\|$, which in turn is equivalent to maximizing$1/\|\omega\|$, so we can rewrite problem (22)-(24) as follows:

$$\frac{1}{\|\omega\|} \rightarrow \underset{\omega, b}{max}$$

s.t.

$\langle \omega, x \rangle + b \geq 1, \forall_x \in C_1$ .....(30)

$\langle \omega, x \rangle + b \leq -1, \forall_x \in C_2$ .....(31)

*Note that the maximum margin hyperplane can be defined by infinite number of parameters ω, b; it is the solution of problem (25)-(27) which is unique.

Second, we can divide constraints by a positive number $\|\omega\|$:

$$\frac{1}{\|\omega\|} \to \underset{\omega, b}{max}$$

s.t.

$$\frac{\langle \omega, x \rangle}{\|\omega\|} \geq \frac{1}{\|\omega\|}, \forall_x \in C_1$$

$$\frac{\langle \omega, x \rangle}{\|\omega\|} \leq \frac{1}{\|\omega\|}, \forall_x \in C_2 d \to \underset{\omega, b}{max}$$

s.t.

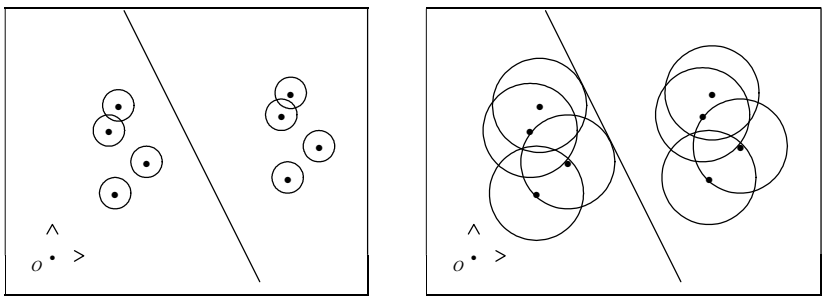
$\rho(\pi, x) \geq d, \forall_x \in C_1$ ……..(32)

$\rho(\pi, x) \leq d, \forall_x \in C_2$ ……..(33)

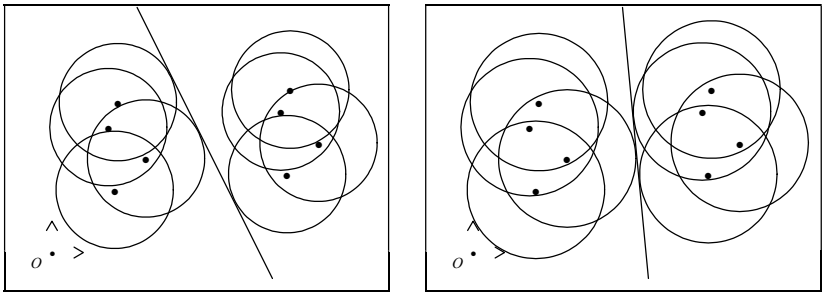
Recalling that $\frac{\langle \omega, x \rangle + b}{\|\omega\|}$ is the distance $\rho(\pi, x)$ between hyperplane $\pi(\omega, b)$ and point , and introducing new variable $d = 1/\|\omega\|$, we get the following problem which is equivalent to (22)-(24):

That is, find parameters ω, b that maximize margin $m = 2d$between , C_1 and C_2 .

Geometrically, connection between parameters ω, b and d can be illustrated in the following way. Let us draw a spherical hull of radius $d = 1/\|\omega\|$ around each training point x. Consider some feasible hyperplane $\pi(\omega, b)$). According to constraints (32), (33), this hyperplane must separate our points together with their hulls (Figure 11a). Now suppose we want to increase d twofold. Since d is a function of $\|\omega\|$, we have to decrease $\|\omega\|$ twofold. If we divide vector w by two, we move our hyperplane parallel to itself further from the origin. However, if we divide by two vector $\omega$and intercept b, we do not move the hyperplane. This way, downscaling w and b, we increase the radius of the hulls while keeping hyperplaneπ in the same position and orientation, until at least one hull touches it (Figure 11b). If we have space to move π parallel to itself away from the hull that touches it, we can do it by changing b only, and let the hulls grow further. At some point, our hulls will reach maximum size d achievable for hyperplanes with normal vectors collinear to ω (Figure 11c). If we have space for the hulls to grow further, we can change orientation of π by changing components of vector ω, while keeping $\|\omega\|$ equal to the current value of $1/d$. Doing so and adjusting b, we keep π feasible and increase d until we arrive to the optimal configuration (Figure 11d).



(a) (b)



(c) (d)

Figure :11 Finding maximum margin hyperplane: geometrical insight.

In case of maximum margin hyperplane, the dual formulation has two major benefits: its constraints are easier to handle than constraints of the primal problem, and it is better suited to deal with kernel functions . This is why it is the dual problem which is actually solved by most SVM $packages$[5] . Besides computational considerations, the dual formulation allows us to establish the concept of support vectors – the training points that define orientation and intercept of the maximum margin hyperplane.

So let us recall the primary optimization problem for finding maximum margin hyperplane$\pi(\omega, b)$:

$$\frac{1}{2}\sum_{k=1}^{n}\omega_k^2 \rightarrow \min_{\omega, b} \quad \text{.....(34)}$$

s.t.

$$y_1(\sum_{k=1}^{n}\omega_k x_{ik} + b) \geq 1, i = 1,2, \dots, l \quad \text{.....(35)}$$

The dual problem for (34)-(35) in its general form is written as

$$L_d(\alpha) \rightarrow \max_{\alpha} \quad \text{.....(36)}$$

s.t.

$$\alpha_i \geq 0, \quad i = 1,2, \dots, l, \quad \text{.....(37)}$$

where $\alpha = (\alpha_1\alpha_2, \dots, \alpha_l)$, $L_d(\alpha)$ is the dual function defined as

$$L_d(\alpha) = \min_{\omega, b} L(\omega, b, \alpha), \quad \text{.....(38)}$$

And $L(\omega, b, \alpha)$ is the Lagrangian defined as

$$L(\omega, b, \alpha) = \frac{1}{2}\sum_{k=1}^{n}\omega_k^2 - \sum_{i=1}^{m}\alpha_i\left(y_i\left(\sum_{k=1}^{n}\omega_k x_{ik} + b\right) - 1\right). \quad \text{.....(39)}$$

The dual can be written in a more explicit form. Since the Lagrangian is a convex function in our $case$[6] , for any α pair (ω^*, b^*) is the global minimum of $L(\omega, b, \alpha)$ if and only if

$$\nabla_{\omega,b} L(\omega^*, b^*, \alpha) = 0, \quad \text{.....(40)}$$

Where$\nabla_{\omega,b}\, L$ denotes the gradient of function L(vector of its first derivatives with respect to $\omega_i$and b), and 0 denotes the null vector from R^n. Thus, $L_d(\alpha) = L(\omega^*, b^*, \alpha)$given that (40) is satisfied, and therefore the dual problem (36)-(34) can be stated as

$L(\omega, b, \alpha) \rightarrow \underset{\omega, b, \propto}{max}$ ……(41)

s.t.

$\nabla_{\omega,b}\, L(\omega, b, \alpha) = 0$ ……(42)

$\alpha_i \geq 0, \;\; i = 1,2, \dots, l.$ ……(43)

The constraint (42) states that

$$\frac{\partial L}{\partial \omega_k} = 0, k = 1,2, \dots, n$$

$$\frac{\partial L}{\partial b} = 0,$$

Which is equivalent to

$\omega_k = \sum_{i=1}^{l} \alpha_i y_i x_{ik}\,, \quad k = 1,2, \dots, n$ ……(44)

$$\sum_{i=1}^{l} \alpha_i y_i = 0.$$

Finally, substituting (44) into (39), we rewrite the dual problem (41)-(43) as

$\sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j\, y_i y_j \langle x_i, x_j \rangle \rightarrow \underset{\alpha}{max}$ ……(45)

s.t.

$\sum_{i=1}^{l} \alpha_i y_i = 0$ ……(46)

$\alpha_i \geq 0, \quad i = 1,2, \dots, l.$ ……(47)

Like the primary problem, the dual is also a quadratic programming problem, but constraints (46)-(47) are easier to handle than constraints (35). This is one of the reasons why many SVM packages solve the dual problem instead of the primal.

The Hessian of objective function (45) (matrix of its second derivatives) with respect to variables α_i has the form

$-1 \times \left[y_i y_j \langle x_i, x_j \rangle\right] l \times l$ ……(48)

Matrix $\left[y_i y_j \langle x_i, x_j \rangle\right] l \times l$ is $congruent$[8]to $\left[\langle x_i, x_j \rangle\right] l \times l$, which is the Gramian matrix for vectors $x_1, x_2, \dots, x_l$. The Gramian matrix is always positive $semidefinite$[9] , therefore matrix (48) is negative semidefinite, which means that problem (45)-(47) is concave (but not strictly concave). A concave problem

may have a single global maximum or many global maxima. Therefore, generally speaking, the solution of problem (45)-(47) is not unique, while the solution of the primal problem (34)-(35) is. We will give an example of problem with several optimal dual solutions in devoted to support vectors.

What is the connection between the dual problem (45)-(47) and the primal problem (35)-(36)? If the primal has solution, so does the dual, and vice versa. If the primal is unbounded (which happens if all training vectors belong to a single class), then the dual is infeasible; if the primal is infeasible (which happens if two classes are not linearly separable), then the dual is either infeasible or unbounded.

What is the connection between the dual solution α^* and the primal solution (ω^*, b^*)? Rewriting equation (44) in vector form, we get:

$\omega^* = \sum_{i=1}^{l} \alpha_i^* y_i x_i$. ……(49)

Intercept b^* can be found from any of the constraints (35) that holds as an $equality$[10] . For example, if

$$\sum_{k=1}^{n} \omega_k^* x_{1k} + b^* = 1$$

Then,

$$b^* = 1 - \sum_{k=1}^{n} \omega_k^* x_{1k}$$

Let us recall that (ω^*, b^*) is the unique global solution to the primal problem, while α^*may be any of numerous global solutions to the dual one. It is also worth to note that in our case there is no duality gap between the primary and the dual objective functions, which means that

$$\frac{1}{2}\sum_{k=1}^{n}(\omega_k^*)^2 = \sum_{i=1}^{l}\alpha_i^* - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i^*\alpha_j^* y_i y_j \langle x_i, x_j\rangle.$$

Example with MATLAB code

Example :01

```
>>freqz(meas(1,1),'ctf');
ylabel("meas(1,1)");
title("meas(1,1)");
legend("show")
```

Result:

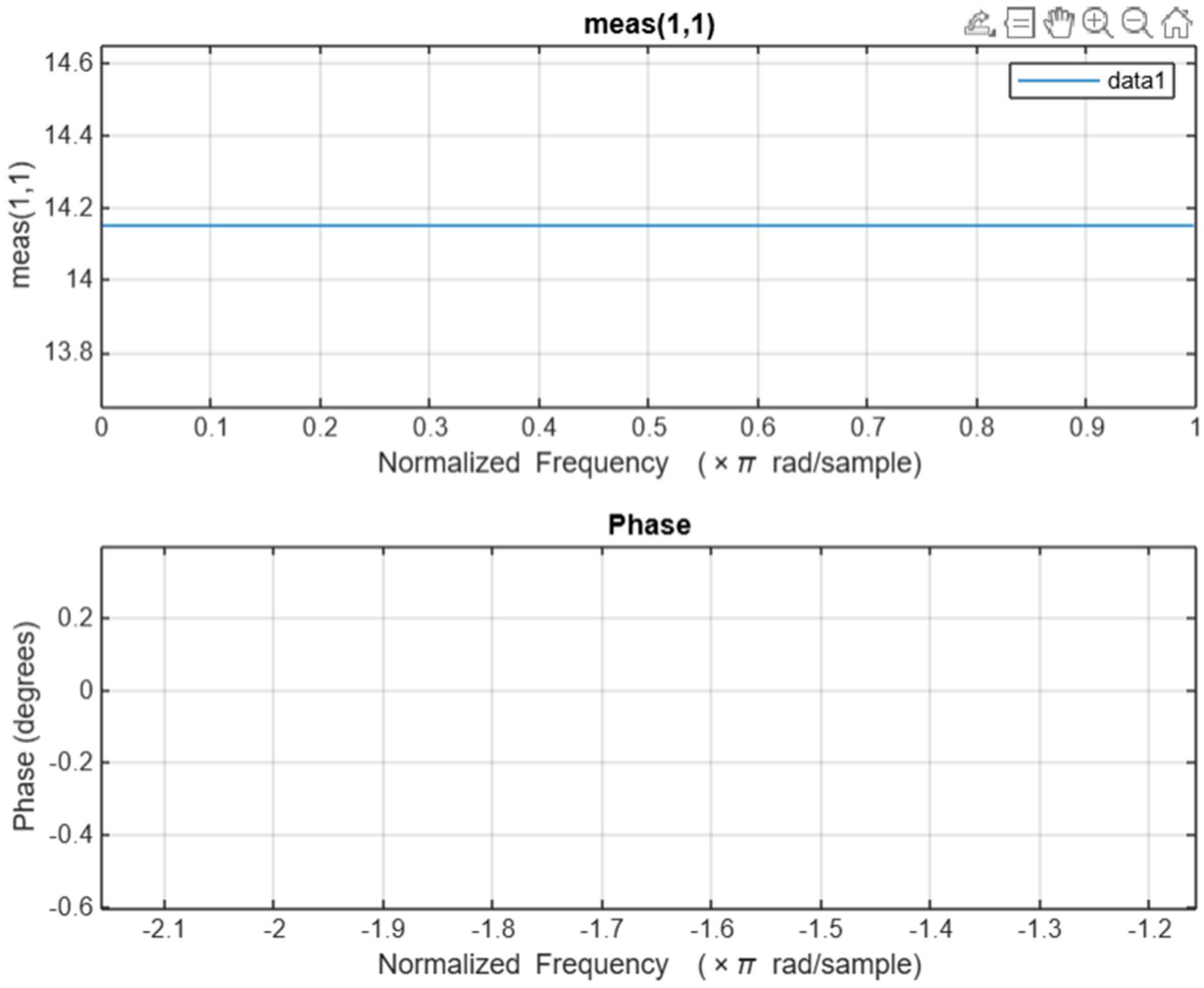


Figure: 12

Example:02

```
% Clear workspace
clc; clear; close all;

% Generate synthetic dataset
rng(1); % For reproducibility
X = [randn(10,2) + 2; randn(10,2) - 2]; % 20 samples, 2 features
Y = [ones(10,1); -ones(10,1)]; % Class labels (+1 and -1)

% Train SVM model
SVMModel = fitcsvm(X,Y,'KernelFunction','linear','Standardize',true);

% Plot data
figure; hold on;
gscatter(X(:,1), X(:,2), Y, 'rb', 'ox', 8);
legend('Class +1','Class -1');

% Get the decision boundary
d = 0.02;
[x1Grid, x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)), min(X(:,2)):d:max(X(:,2)));
XGrid = [x1Grid(:), x2Grid(:)];
[~, scores] = predict(SVMModel, XGrid);

% Plot decision boundary
contour(x1Grid, x2Grid, reshape(scores(:,2), size(x1Grid)), [0 0], 'k', 'LineWidth', 2);
title('SVM Classification with Linear Kernel');
xlabel('Feature 1'); ylabel('Feature 2');
hold off;
```

Example:03

```
% Clear workspace
clc; clear; close all;

% Generate synthetic dataset
rng(2); % For reproducibility
X = [randn(20,2) + 2; randn(20,2) - 2]; % 40 samples, 2 features
Y = [ones(20,1); -ones(20,1)]; % Class labels (+1 and -1)

% Train SVM model with RBF kernel
SVMModel = fitcsvm(X,Y,'KernelFunction','RBF','Standardize',true);

% Plot data
figure; hold on;
gscatter(X(:,1), X(:,2), Y, 'rb', 'ox', 8);
legend('Class +1','Class -1');

% Decision boundary
d = 0.02;
[x1Grid, x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)), min(X(:,2)):d:max(X(:,2)));
XGrid = [x1Grid(:), x2Grid(:)];
[~, scores] = predict(SVMModel, XGrid);

% Plot decision boundary
contour(x1Grid, x2Grid, reshape(scores(:,2), size(x1Grid)), [0 0], 'k', 'LineWidth', 2);
title('SVM Classification with RBF Kernel');
xlabel('Feature 1'); ylabel('Feature 2');
hold off;
```

Example:04

```
% Clear workspace
clc; clear; close all;

% Load Fisher's iris dataset
load fisheriris;
X = meas(:,1:2); % First two features
Y = species; % Three classes: 'setosa', 'versicolor', 'virginica'

% Train multi-class SVM model
SVMModel = fitcecoc(X,Y);

% Plot data
figure; hold on;
gscatter(X(:,1), X(:,2), Y, 'rgb', 'osd', 8);
legend('Setosa','Versicolor','Virginica');

% Decision boundary
d = 0.02;
[x1Grid, x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)), min(X(:,2)):d:max(X(:,2)));
XGrid = [x1Grid(:), x2Grid(:)];
predictedLabels = predict(SVMModel, XGrid);

% Plot decision boundary
gscatter(x1Grid(:), x2Grid(:), predictedLabels, 'rgb', 'o', 1, 'off');
title('Multi-Class SVM on Iris Dataset');
xlabel('Sepal Length'); ylabel('Sepal Width');
hold off;
```

Example:05

```
% Load MNIST Handwritten Digits Dataset
load('mnist.mat'); % Ensure you have an MNIST dataset

% Prepare data
X_train = double(train_images(:,:,:,1:5000)); % Use first 5000 images
X_train = reshape(X_train, [5000, 28*28]); % Flatten images
Y_train = categorical(train_labels(1:5000));

X_test = double(test_images(:,:,:,1:1000));
X_test = reshape(X_test, [1000, 28*28]);
Y_test = categorical(test_labels(1:1000));

% Train SVM
SVMModel = fitcecoc(X_train, Y_train);

% Predict on test data
YPred = predict(SVMModel, X_test);

% Accuracy Calculation
accuracy = sum(YPred == Y_test) / numel(Y_test) * 100;
fprintf('Test Accuracy: %.2f%%\n', accuracy);
```

Example:06

```
% Clear workspace
clc; clear; close all;

% Generate synthetic dataset
x = (1:0.5:10)'; % Feature
y = sin(x) + 0.1*randn(size(x)); % Target with noise

% Train SVM regression model
SVMModel = fitrsvm(x, y, 'KernelFunction', 'RBF', 'Standardize', true);

% Predict on test data
xTest = (1:0.1:10)'; % Test inputs
yPred = predict(SVMModel, xTest);

% Plot results
figure;
plot(x, y, 'ro', 'MarkerSize', 8, 'LineWidth', 2); hold on;
plot(xTest, yPred, 'b-', 'LineWidth', 2);
legend('Original Data', 'SVM Regression');
xlabel('X'); ylabel('Y');
title('SVM Regression using RBF Kernel');
grid on;
hold off;
```

Example:07

```
% Load dataset
load fisheriris;
X = meas;
Y = species;

% Train SVM model
SVMModel = fitcsvm(X, Y, 'KernelFunction', 'linear');

% Get feature importance using SVM weights
w = SVMModel.Beta;
featureImportance = abs(w) / sum(abs(w));

% Plot feature importance
figure;
bar(featureImportance);
xlabel('Feature Index');
ylabel('Importance Score');
title('Feature Importance using SVM');
grid on;
```

Example:08

```
% Generate synthetic normal data
rng(3);
X = [randn(50,2); randn(50,2) + 4]; % Normal samples

% Train One-Class SVM
SVMModel = fitcsvm(X, 'KernelFunction', 'RBF', 'Nu', 0.05, 'OutlierFraction', 0.1);

% Generate test data
Xtest = [randn(10,2) * 2; randn(5,2) + 6]; % Some anomalies included
[label, score] = predict(SVMModel, Xtest);

% Plot results
figure; hold on;
gscatter(X(:,1), X(:,2), ones(size(X,1),1), 'b', 'o');
gscatter(Xtest(:,1), Xtest(:,2), label, 'gr', 'x');
legend('Normal Data', 'Normal Test', 'Anomalies');
title('One-Class SVM Anomaly Detection');
xlabel('Feature 1'); ylabel('Feature 2');
hold off;
```

Example:09

```
% Load breast cancer dataset

load breastcancer;


% Prepare data

X = breastcancerInputs';

Y = breastcancerTargets(1,:)'; % Binary classification (benign/malignant)


% Train SVM model

SVMModel = fitcsvm(X, Y, 'KernelFunction', 'linear', 'Standardize', true);


% Predict on the dataset

YPred = predict(SVMModel, X);

accuracy = sum(YPred == Y) / numel(Y) * 100;


% Display result

fprintf('Breast Cancer SVM Accuracy: %.2f%%\n', accuracy);
```

Conclusion

This study is not a comparison between three classification algorithms because each algorithm has been given different machine learning problems to tackle which fits their strength accordingly.Image classification has made countless growth over past decade. This paper summarizes the comprehensive overview with generalized the typical functionality and highlighting their application in remote sensing and medical field. The main contribution of this paper is briefly reviewing the concept of SVM and SVM classification accuracy in the different data set. It is well known that classification methodologies may vary with different type image data. This paper mainly focuses on SVM method during 2015–2017.

This study shows that there are many ways of modifying the method to get better classification accuracy. Accuracy assessment is an integral part in image classification technique. Features are contained the informative data for image classification. Extraction and Selection of informative, feature play an important role in achieving the accuracy. So this paper also discusses the various feature selection method and kernel parameter selection strategies.

The success of an image classification depends on many factors like high quality image data set, ancillary data, extracted feature, feature selection and designing an appropriate classification method. A relative study of different classification method is helpful for enhancement of classification accuracy. It is essential for the forthcoming investigation to develop guidelines on the applicability and capability of major classification methodology. This survey of different technique provides a platform for the designing a new novel scheme in this area as a future work.

References

1. S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd ed., Prentice Hall, 1999.
2. V. Vapnik, The Nature of Statistical Learning Theory, Springer, 1995.
3. C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
4. T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed., Springer, 2009.
5. L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
6. J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd ed., Morgan Kaufmann, 2011.
7. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.
8. A. Ng, "Machine Learning," Stanford University Lecture Notes, 2011.
9. P. Domingos, "A Few Useful Things to Know About Machine Learning," Communications of the ACM, vol. 55, no. 10, pp. 78–87, 2012.
10. Cortes C, and Vapnik V (1995) Support-vector network. Mach Learn 20(3):273–297.
11. Tomar D, Agarwal S (2015) A comparison on multi-class classification methods based on least squares twin support vector machine. Knowl-Based Syst 81:131–147.
12. Mangasarian OL, Musicant DR (2001) Lagrangian support vector machines. J Mach Learn Res 1:161–177.
13. Xiang Z, XueqiangLv, Zhang K (2014) An Image Classification Method Based On Multi-feature Fusion and Multi-kernel SVM. In: Seventh International Symposium on Computational Intelligence and Design, Hangzhou, p 49–52
14. Hsu C-W, Lin C-J (2012) A comparison of methods for multiclass support vector machines. IEEE Trans Neural Networks 13(2):415–425.
15. Milgram J, Cheriet M, Sabourin R (2006) One Against One or One Against All: Which One is Better for Handwriting Recognition with SVMs. In: Tenth International Workshop on Frontiers in Handwriting Recognition, Oct 2006, La Baule (France), Suvisoft.