Evaluation of Support Vector Machine Kernels for Detecting Network Anomalies

by

Prerna Batta

B.Tech, Lovely Professional University, 2013

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Applied Science

in the
School of Engineering Science
Faculty of Applied Science (Engineering Science

SIMON FRASER UNIVERSITY
Summer 2019

Approval

Name:	**Prerna Batta**
Degree:	**Master of Applied Science**
Title:	**Evaluation of Support Vector Machine Kernels for Detecting Network Anomalies**
Examining Committee:	**Chair:** **Ivan V. Bajić** Professor
	Ljiljana Trajković Senior Supervisor Professor
	Parvaneh Saeedi Supervisor Associate Professor
	Mirza Faisal Beg Internal Examiner Professor
Date Defended:	**April 18, 2019**

Abstract

Border Gateway Protocol (BGP) is used to exchange routing information across the Internet. BGP anomalies severely affect network performance and, hence, algorithms for anomaly detection are important for improving BGP convergence. Efficient and effective anomaly detection mechanisms rely on employing machine learning techniques. Support Vector Machine (SVM) is a widely used machine learning algorithm. It employs a set of mathematical functions called kernels that transform the input data into a higher dimensional space before classifying the data points into distinct clusters. In this Thesis, we evaluate the performance of linear, polynomial, quadratic, cubic, Gaussian radial basis function, and sigmoid SVM kernels used for classifying power outage such as Moscow Power Blackout, BGP mis-configuration, and BGP anomalies such as Slammer, Nimda and Code Red I. The SVM kernels are compared based on accuracy and the F-Score when detecting anomalous events in the Internet traffic traces. Simulation results indicate that the performance heavily depends on the selected features and their combinations.

Keywords: Border gateway protocol; routing anomalies; machine learning; feature selection; support vector machine; kernels

Dedication

This thesis is dedicated to my parents, for their endless love, support, and encouragement.

Acknowledgements

I express my gratitude to my senior supervisor Prof. Ljiljana Trajković for her guidance and countless hours spent for reviewing my work. She has always been very encouraging and supportive. She always allowed me to explore my ideas and I am very thankful to her for the time and efforts she invested in me and my research work. This would not have been possible without her hard work and support.

I would also like to thank my committee chair Ivan V. Bajić and committee members Prof. Parvaneh Saeedi and Prof. Faisal Beg for providing valuable comments and suggestions.

I am very thankful to my colleagues in the Communication Networks Laboratory. Special thanks to Zhida Li for his friendship and support during my graduate studies. I also thank Dr. Maninder Singh, Qingye Ding, and Ana Gonzalez for their valuable comments and suggestions.

My deepest gratitude goes to my family for their support and unconditional love. I am very thankful to my mother Swati Batta and father Rajeev Batta for always advising me to work hard and pursue my studies. You both are my guiding light and my pillar of strength. Thank you for believing in me. Last but not least, I would also like to extend my gratitude to my friends Anmit Deol, Jaspreet Gill, and Gursimran Sahota who have provided support in countless ways and always motivated me to keep working hard to achieve my goals. I am very grateful to my brother Ankit Batta. You supported me through the toughest times of my life. Without you, I would not have made it this far.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine learning is a subset of artificial intelligence that deals with designing algorithms, building models and making decisions based on the input datasets. Machine learning models classify data using a feature matrix. The rows and columns of the matrix correspond to data points and feature values, respectively. By providing a sufficient number of relevant features, machine learning algorithms help build models to more accurately classify data. They have been recently used to optimally solve a variety of engineering and scientific problems. Among various machine learning approaches such as Logistic Regression, naïve Bayes, and Support Vector Machine (SVM), SVM is often used due to its robust nature [16]. It classifies data by defining a decision boundary to lie geometrically midway between the support vectors. Support vectors are the data points that lie closest to the decision surface. SVM deals with both linearly and nonlinearly separable features of the input dataset by using kernels.

Machine learning algorithms are classified as supervised and unsupervised learning. Kernel functions are used to map a nonlinearly separable into a higher-dimensional linearly separable data [30].

1.1 Supervised Learning

The term "supervised learning" refers to algorithms that learn from training dataset and are being trained to supervise the learning process. The algorithm learns the mapping function to derive an output from the given input. The learned mapping function is then used to predict the output for new input data. After multiple iterations, the algorithm achieves a desirable level of performance for predicting the outcomes. Classes used in supervised learning are known which implies that all input data is labelled. The goal of supervised learning is to identify clusters of data. Supervised learning is typically used when we wish to perform classification (map input labels to output labels) or regression (map input labels to a continuous output labels). Supervised learning is used to train a model to perform regression or classification. Hence, supervised learning algorithms are classified into classification and regression groups:

- **Classification**: The output variable falls into a category. Example: "This apple is green in colour" or "this apple is red in colour".
- **Regression**: The output variable has a numerical value. Example: "The height of this building is 100 feet" or "Frank only has 100 dollars in his account".

In both cases, the goal is to find specific connection or structure among the input variables that would help produce the correct output. The output is determined based on the training data. The supervised learning produces accurate and reliable results because the input variables are known and labeled, which implies that the algorithm will only attempt to identify the hidden patterns.

1.2 Unsupervised Learning

In "unsupervised machine" learning, the input data is uncategorized (unlabeled). Unlike supervised learning, there is no prior knowledge of the data and the number of classes are not known. Without knowing the labels, unsupervised learning algorithm models the distribution of the input data [14]. The algorithm has to define and label the input data before determining the hidden patterns and structure. It is expected to group the data into clusters using various categories. These algorithms are grouped into clustering and association groups:

- **Clustering**: Find inherent groupings within the input data. Example: grouping customers by their social status.
- **Association**: Find various associations between different groups of input data. Example: customers that buy item A tend to buy item B as well.

Clustering is the most common unsupervised technique used to find the hidden patterns or create clusters. Unsupervised learning is used to explore the data and train a model to spilt data up into clusters. The most common algorithm used for performing clustering is k-means.

1.3 Support Vector Machine

SVM is a supervised machine learning algorithm that takes a known set of data as an input. It builds a model for the known input. Based on the model, it predicts the outcome [19] for new data. There are two main stages in supervised learning: training and testing. In the training stage, the supervised algorithm builds the model while in the testing phase, the same model predicts the output. SVM is the most robust Machine Learning algorithm designed by Cortes and Vapnik [19].

SVM considers each feature as a point in n-dimensional space and classifies the data into two classes by using a hyperplane. For example, for a two-dimensional space, the hyperplane is a line. In higher dimensions (3 and higher), the separation is a hyperplane. The goal of SVM is to find maximum separation between two classes. Data points that lie on the edge of the plane are called support vectors. Two basic rules that govern the selection are:

- Step 1: Select the hyperplane that separates the data into two distinct classes.
- Step 2: Maximize the distance between the nearest data points of either class such that no data point is misclassified.

Support Vector Machine (SVM) was originally formulated to address binary classification problems. It was used as a method in extending logistic regression to generate maximum entropy classification models. The general formulation of SVM [52] includes:

- Linear learning function as the classification or the regression model;
- Constraints defining the theoretical bounds;
- Convex duality and formulations concerning the associated dual-kernel;
- Dual-kernel formulations sparseness.

The combination of these processes makes SVM robust and unique. SVM offers a clear computational advantage over the standard statistical methods.

1.4 Research Contributions

In this Thesis, we have revised and extended our previous research findings and results [10, 11, 13, 27, 28, 33, 34] by employing various SVM kernels for detecting BGP anomalies. We consider BGP update messages that are extracted from the data collected during the time periods when the Internet experiences known anomalies. The performance of anomaly classifiers is closely related to the selected features. Past approaches to select BGP feature employed minimum Redundancy Maximum Relevance (mRMR) [44] algorithms such as Mutual Information Deference (MID), Mutual Information Quotient (MIQ), and Mutual Information Base (MIBASE). In this Thesis, we employ the decision tree algorithm for feature selection and evaluate performance of SVM using linear, quadratic, cubic, polynomial, Gaussian RBF, and sigmoid kernels.

For experiments, we use a Windows 10 64-bit PC with 16 GB RAM and Intel Core i5 processor. We perform experiments in Matlab R2019a [4] with Statistics and Machine Learning Toolbox to evaluate the performance of SVM kernels. All kernel functions are in-built in the Classification Learner application of the Matlab tool. We select 37 features or the most relevant features using the decision tree algorithm. Some features are removed from the constructed decision tree because they are repeatedly used [33]. We train the SVM with linear, polynomial, quadratic, cubic, Gaussian RBF, and sigmoid kernels. We test the models using various datasets and evaluate the performance of the SVM kernels based on accuracy and F-Score. The performance of SVM using linear, quadratic, and cubic kernels has been reported in [13]. In this Thesis, we have extended our work by adding the performance of SVM using polynomial, Gaussian RBF, and sigmoid kernels.

1.5 Organization of the Thesis

The Thesis is organized as follows: We first briefly describe supervised learning, unsupervised learning, and SVM algorithms. We provide an overview of the SVM algorithm as well as the description of hyperplane and margins in Chapter 2. In Chapter 3, we describe SVM with kernels including linear and various types of nonlinear kernels. BGP datasets used for detecting network anomalies, BGP anomalies that have been considered in this Thesis, and feature selection is given in Chapter 4. We compare the performance of SVM using linear and nonlinear kernels in Chapter 5 and conclude with Chapter 6. The list of references is provided in the Bibliography section.

Chapter 2

Support Vector Machine Algorithm

Support Vector Machine (SVM) is one of the most effective classifiers. It handles linear as well as nonlinear functions using various kernels. SVM may be used with datasets having larger number of features without adding too much complexity to the system [51]. SVM tries to classify the given data points into two groups as shown in Fig. 2.1.

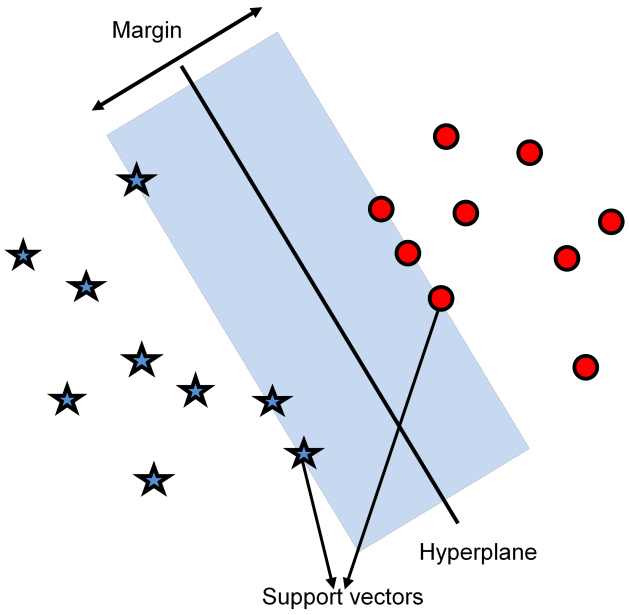

Figure 2.1: Support Vector Machine.

By selecting hyperplane B one data point is misclassified. Fig. 2.2 shows that hyperplane A is best suited for classifying the given data.

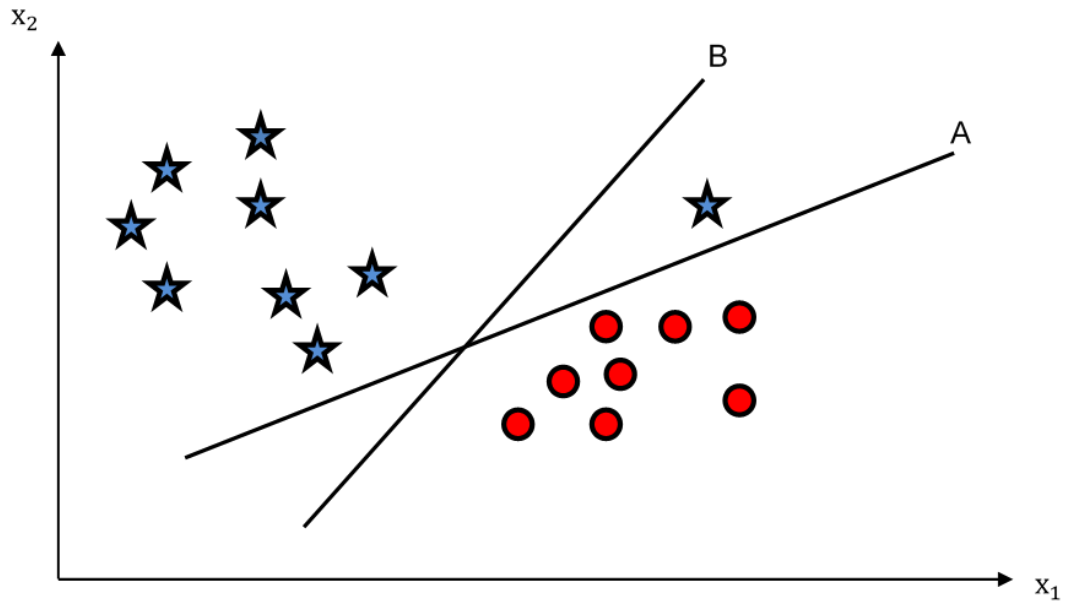

Figure 2.2: Selection of hyperplanes in a two-dimensional feature space.

Consider the case of logistic regression where we consider the output of the linear function and find the value within the range of 0 and 1 by using the logistic sigmoid function, which is an s-shaped curve that takes any real number and maps it into a value between 0 and 1. Typically, label 1 is used if the value is greater than a threshold value. Otherwise, label 0 is assigned. If x and y are the input and output variables, respectively, logistic regression tries to find the probability that $y = 1$ given the input x:

$$\text{P}(y{=}1 \mid x). \tag{2.1}$$

We use the logistic sigmoid function to map predicted values to probabilities:

$$\text{h(x)} = \frac{1}{1 + e^{-x}}, \tag{2.2}$$

where $h(x)$ is the output between 0 and 1, x is the input to the function, and e is the base of natural log. The model attempts to maximize the distance between the training instance and the decision surface. A surface in a multidimensional state space that divides the space into different regions is called decision surface. Data points lying on side of a decision surface belong to a different class from those lying on the other side of a decision surface. The distance between the closest point and the decision surface is called a margin, as shown in Fig. 2.3.

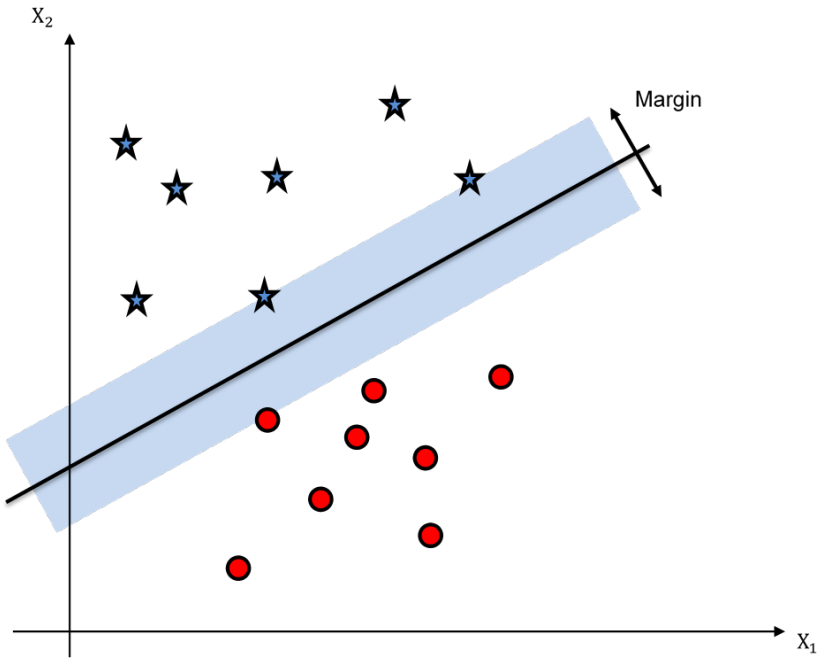

Figure 2.3: Decision surface and the margin.

Among all decision surfaces, the surface that gives maximum margin is selected. The points lying closest to the decision surface are called support vectors. For a linearly separable two-dimensional feature space, there are at least two support vectors. These support vectors determine the decision surface [45]. Various SVM kernels may be employed for classification. If a dataset is not linearly separable, SVM with linear kernel performs poorly [31]. In such a scenario, polynomial nonlinear SVM kernels are used. Nonlinear kernels enlarge the feature space of input datasets to generate nonlinear boundary between the classes. This is achieved by mapping the data to a higher dimensional space. Such a boundary created by nonlinear kernels appears as a linear hyperplane and is used to extend SVM to nonlinear surfaces.

2.1 Nonlinear Hyperplane

For a two-dimensional dataset, it is relatively easy to find a hyperplane. In such cases, kernels are used to generate the hyperplanes. Kernels are the mathematical functions used to transform two-dimensional points into n-dimensional space [29]. It is a type of data transformation function. Once the data points are transformed into n-dimensional space, a hyperplane is then built so that it separates the data into distinct sets. As shown in Fig. 2.4, the kernel function used to generate the hyperplane is of the form $x^2 + y^2 = z^2$. The corresponding hyperplane is a circle that separates the two data points without misclassifications.

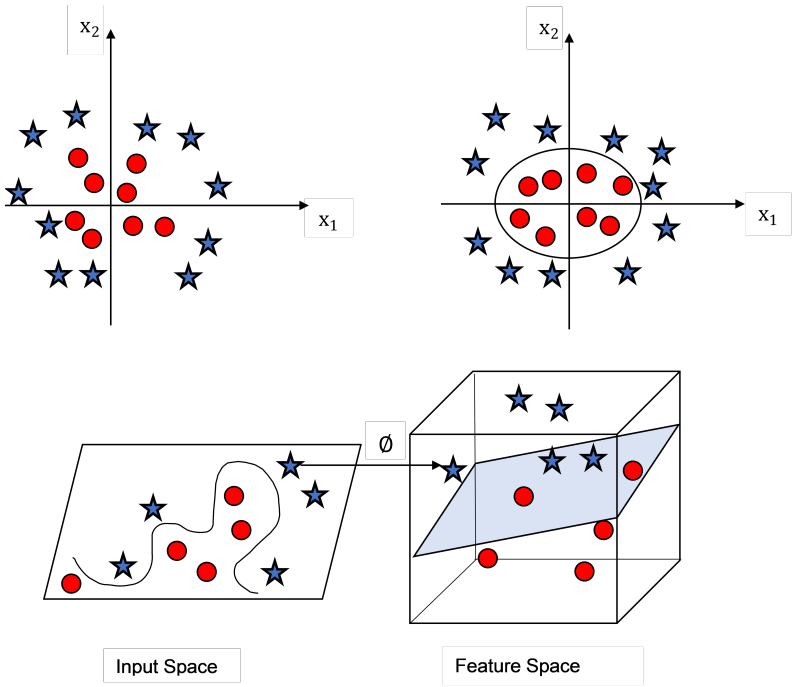

Figure 2.4: Selection of a hyperplane in n-dimensional feature space, where ϕ represents mapping

The objective function for SVM with nonlinear kernel has the form [16]:

$$\max \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j, \tag{2.3}$$

where λ_i and λ_j correspond to the points that are closest to the margin. The objective function (2.3) depends on the dot product of input vector pairs $\mathbf{x}_i \cdot \mathbf{x}_j$ that represents a kernel function. Standard machine learning methods may be formulated as the task of minimizing a convex function f that depends on a variable vector $\mathbf{w}$ with n entries. This can be formally written as the optimization problem:

$$min_{w \in R^n} f(\mathbf{w}), \tag{2.4}$$

where $f(\mathbf{w})$ is the objective function of the form:

$$f(\mathbf{w}) = \lambda R(\mathbf{w}) + \frac{1}{n} \sum_{n}^{i=1} L(\mathbf{w}; \mathbf{x}_i, y_i), \tag{2.5}$$

where $\mathbf{x}_i \in R^n$ are the training data for $1 \leq i \leq n$ and $y_i \in R$ are their corresponding labels that we wish to predict. In any classification problem, the objective function may be defined as a sum of regularization and loss functions. The objective function f has two parts: the regularization function that controls the complexity of the model and the loss function that measures the error of the model using the training data. The loss function $L(\mathbf{w}; x, y)$ is a convex function of $\mathbf{w}$. The

regularization parameter $\lambda \geq 0$ defines the trade-off between minimizing the loss (training error) and the model complexity (overfitting).

Let x_i be the input sample, y_i the output label, $\mathbf{w}$ the weight vector, and λ the regularization parameter. The objective function may be written as:

$$\min_{\mathbf{w}} \lambda ||\mathbf{w}||^2 + \sum_{i=1}^{n} (1 - y_i[x_i, \mathbf{w}]). \tag{2.6}$$

A loss function indicates the quality of the employed classifier. The smaller the loss function, the better is the classifier at classifying the input data. The amount of work required in order to increase the classification accuracy is proportional to the loss function. The input vector is x_i, the output label is y_i, and the weight vector $\mathbf{w}$ needs to be calculated. Hence, the objective function calculates weight so that the system trains and builds its model. Therefore, the loss function in (2.5) is the dot product of the output vector and input samples x_i. The sum:

$$\sum_{i=1}^{n} (1 - y_i[x_i, \mathbf{w}]). \tag{2.7}$$

gives the total loss over the entire set of sample points. Minimizing the objective function leads to the minimum loss.

2.2 Functional Margin

Consider an arbitrary point (x_i, y_i) lying on one side of the decision surface. Suppose that the equation of the decision surface is given by:

$$\mathbf{w}x + b = 0, \tag{2.8}$$

where $\mathbf{w}$ is a weight vector, x is the input vector, and b is the bias. Let x_1 and x_2 be the two features. The equation of the decision surface may be written as:

$$x_1\mathbf{w} + x_2\mathbf{w} + b = 0. \tag{2.9}$$

Consider the point (x_i, y_i) with respect to the line $(\mathbf{w}, b)$. The functional margin is a distance between (x_i, y_i) and the decision boundary $(\mathbf{w}, b)$. To find this distance, consider the straight line $\mathbf{w}x + b = 0$ shown in Fig. 2.5.

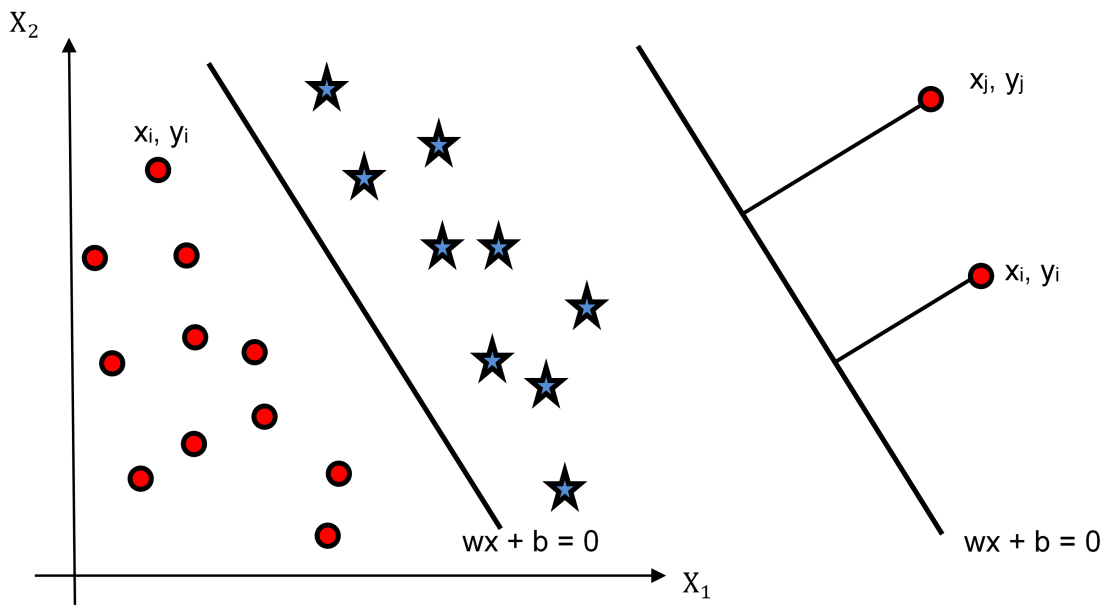

Figure 2.5: Illustration of arbitrary point (x_i, x_j) lying on one side of the decision surface $\mathbf{w}x + b = 0$ (left) and the functional margin (right).

Let a vector $(\mathbf{w}_1, \mathbf{w}_2)$ lie perpendicularly to this line. The distance of (x_i, y_i) from the decision surface γ^i is given by:

$$\gamma^i = y_i(\mathbf{w}^T x_i + b), \tag{2.10}$$

where $\mathbf{w}^T$ is the optimal weight vector, (x_i, y_i) are arbitrary points, and b is the bias. An arbitrary point (x_j, y_j) will have a different functional margin. As shown in Fig. 2.5, the functional margin for (x_j, y_j) is larger than the functional margin for (x_i, y_i), which implies that if the point is further away from the decision surface, there is higher confidence in its classification. Therefore, larger functional margin implies better confidence in predicting the class of an arbitrary point. Weight vector $\mathbf{w}^T$ and bias b may be arbitrarily scaled (2.10). Even though the slope remains the same, the functional distance increases or decreases depending on $\mathbf{w}^T$ and b. Hence, functional margin is seldom used as the most important SVM parameter [12]. For example, for multiple training data points $S = (x_1, y_1), (x_2, y_2), ..., (x_i, y_i)$, the functional margin γ is given as:

$$\gamma = \min_{i=1}^{n} \gamma^i. \tag{2.11}$$

A geometrical margin is invariant to the scaling as shown in Fig. 2.5. Any line to the decision surface $\mathbf{w}x + b = 0$ is $\mathbf{w}$. Hence, the unit vector perpendicular to the decision surface is given by $\frac{w}{||\mathbf{w}||}$. As an example, consider the line given by $2x + 3y + 1 = 0$. Hence, $\mathbf{w}$ = (2,3) and $\frac{w}{||\mathbf{w}||}$:

$$\frac{w}{||\mathbf{w}||} = \Big(\frac{2}{\sqrt{13}}, \frac{3}{\sqrt{13}}\Big). \tag{2.12}$$

Now consider a point P with coordinates (a_1, a_2), as shown in Fig. 2.6. Geometric margin of a hyperplane is defined as functional margin scaled by **w**.

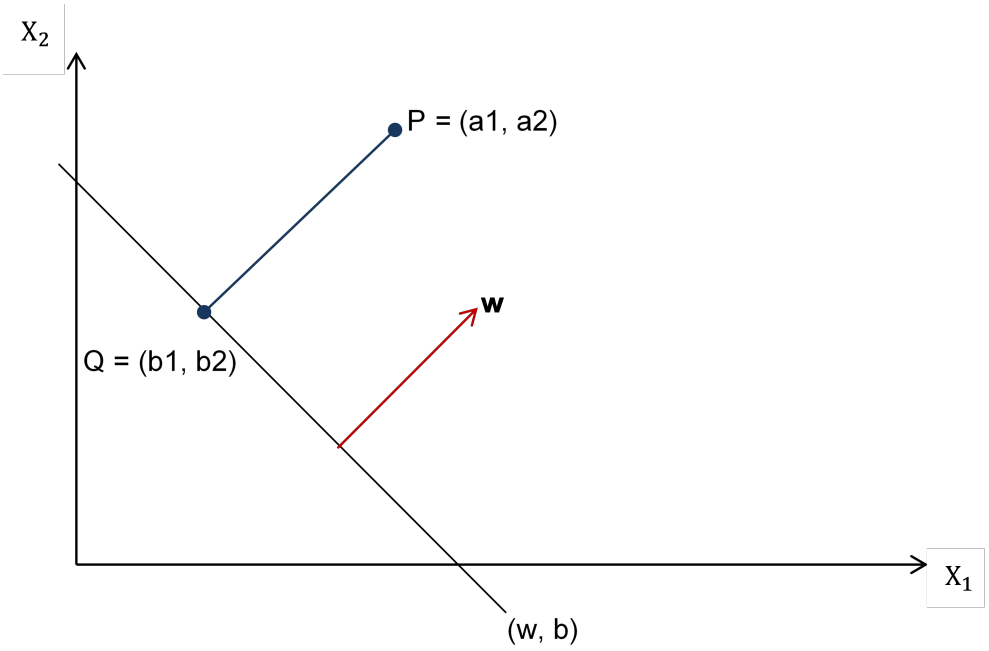

Figure 2.6: Geometric margin: SVM maximizes the geometric margin by learning a suitable decision surface.

A perpendicular line from a point P to the decision surface connects at a point Q with coordinates (b_1, b_2). The smallest distance between P and Q is in the direction of the perpendicular vector. Hence:

$$\begin{aligned} P &= \quad Q \quad + \gamma \frac{w}{||\mathbf{w}||} \\ (a_1, a_2) &= (b_1, b_2) + \gamma \frac{w}{||\mathbf{w}||}, \end{aligned} \tag{2.13}$$

where γ is the decision surface. Solving for γ gives:

$$\mathbf{w}^T \Big((a_1, a_2) - \gamma \frac{w}{||\mathbf{w}||} \Big) + b = 0. \tag{2.14}$$

Hence,

$$\begin{aligned} \gamma &= \frac{\mathbf{w}^T (a_1, a_2) + b}{||\mathbf{w}||} \\ &= \frac{\mathbf{w}^T}{||\mathbf{w}||} (a_1, a_2) + \frac{b}{||\mathbf{w}||} \\ &= y \Big(\frac{\mathbf{w}^T}{||\mathbf{w}||} (a_1, a_2) + \frac{b}{||\mathbf{w}||} \Big) \ \ \forall \ \ y = \pm 1, \end{aligned} \tag{2.15}$$

where w is scaled so that $||\mathbf{w}|| = 1$. The geometric margin after normalization is:

$$\gamma = y\Big(\mathbf{w}^T(a_1, a_2) + b\Big). \tag{2.16}$$

For a given set of points, the geometric margin is the smallest of all margins for the individual points. Hence,

$$\gamma = \min_{i=1}^{n} \gamma^i. \tag{2.17}$$

We have assumed that the data points are linearly separable. The selected classifier should have maximum margin width and be robust to the outliers. Such a classifier will have a strong generalization ability to separate all the data points into two distinct classes. The geometric margin $\gamma||\mathbf{w}||$ needs to be maximized subject to certain constraints. Hence, the optimization problem reduces to:

$$\begin{aligned} &\text{maximize} \quad \frac{\gamma}{||\mathbf{w}||} \\ &\text{subject to} \quad y_i(\mathbf{w}^T x_i + b) \geq \gamma, \quad i = 1, 2, ..., N, \end{aligned} \tag{2.18}$$

where N is the size of training data.

If (w, b) characterizes the decision surface, $\gamma/||\mathbf{w}||$ is the geometric margin. SVM learns the values of $(\mathbf{w}, b)$ so that the geometric mean is the largest subject to the following constraints:

$$\begin{aligned} &y_i(\mathbf{w}x_i + b) \geq \gamma, \quad \text{for positive data points} \\ &y_i(\mathbf{w}x_i + b) \leq \gamma, \quad \text{for negative data points.} \end{aligned} \tag{2.19}$$

In order to normalize the function, $\mathbf{w}$ is scaled so that the geometric margin is $1/||\mathbf{w}|| = 1$, as shown in Fig. 2.7. Transforming the problem into minimization, normalized SVM may be formulated as:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}||\mathbf{w}|| \, , \\ &\text{such that} \quad y_i(\mathbf{w}^T x_i + b) \geq 1. \end{aligned} \tag{2.20}$$

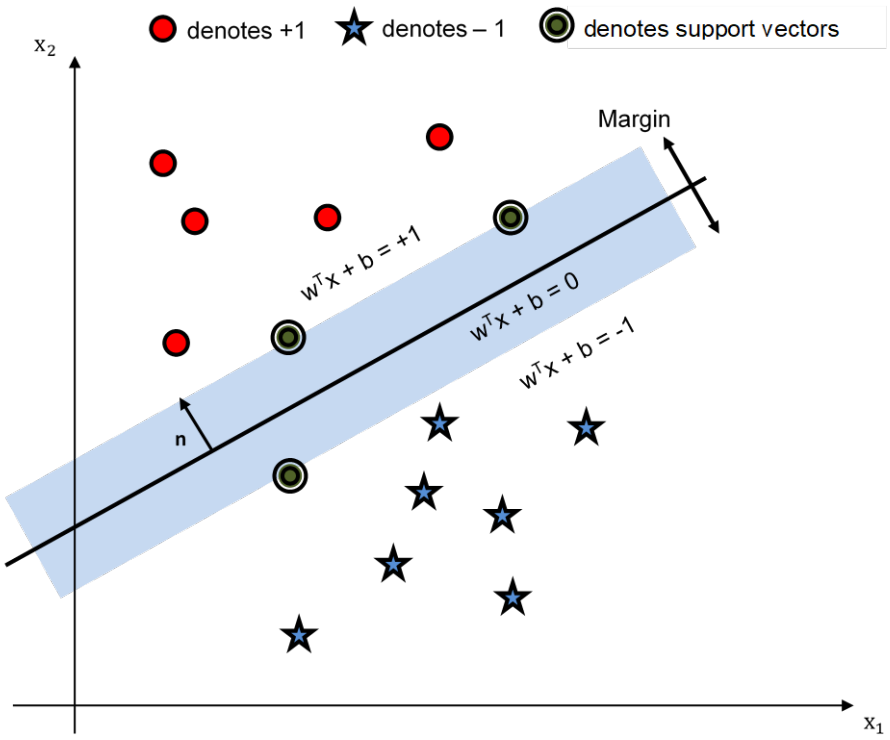

Figure 2.7: Illustration of normalized SVM formulation.

Hence, SVM is a quadratic optimization problem with convex quadratic objective and a set of linear inequality constraints. It may be solved using quadratic programming solvers.

2.3 Maximum Margin With Noise

When data is not linearly separable, linear SVM mathematical formulations fails to perform. To deal with the nonlinear separability of data points, changes are made to the objective function. Along with maximizing the margin, objective function should also incorporate a function to minimize the number of misclassifications, as shown in Fig. 2.8.

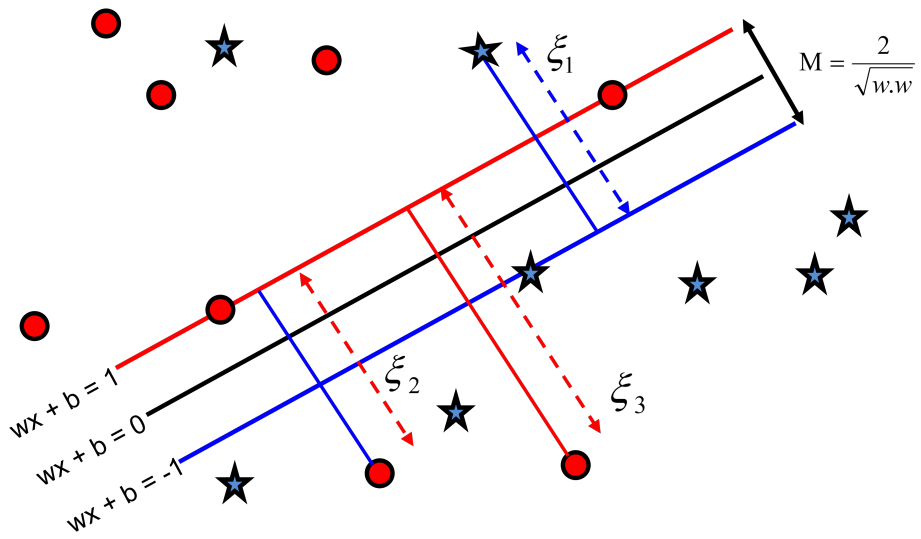

Figure 2.8: Illustration of SVM with noise margin.

The objective function may be written as:

$$\text{minimize:} \quad ||\mathbf{w}|| + \ C \cdot \text{training errors}, \tag{2.21}$$

where C is the control parameter. Let $f(\xi_n)$ be the penalty function associated with each misclassified training point. Minimizing the penalty function reduces the number of misclassified instances. Hence, the problem may be formulated as:

$$\begin{aligned} &\text{minimize} && ||\mathbf{w}|| + \ \mathrm{C} \cdot \sum_{n=1}^{N} f(\xi_n), \\ &\text{with constraints} && y^n(\mathbf{w} \cdot x^n + b) \geq 1 - \xi_n \\ & && \xi_n \geq 0, \end{aligned} \tag{2.22}$$

where N is the size of training data.

Slack variables ξ_n are used to calculate the training errors by measuring the distance of each training data point to its marginal hyperplane. By adding the slack variables $(\xi_s, \xi_2, ..., \xi_n)$ to the penalty function, the objective function reduces to a minimization problem. The constraint $(y_n(\mathbf{w}x_n + b) \geq 1)$ will no longer be true for misclassified points. Hence, penalty function is remodeled to $y_n(\mathbf{w} \cdot x_n + b) \geq (1 - \xi_n \ \forall \ n = 1, 2, ..., N)$. In order to convert a constrained optimization problem into an unconstrained optimization problem, Lagrangian multipliers are used. The Lagrangian

multipliers may be written as:

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2}\|\mathbf{w}\| + \mathbf{C} \cdot \sum_{n=1}^{N} \xi_n + \sum_{n=1}^{N} \alpha_i [y_n(x \cdot \mathbf{w} + b) - 1 + \xi_n] - \sum_{n=1}^{N} \beta_n \xi_n, \tag{2.23}$$

where C is a trade-off parameter to balance the violation of the margin while α_n and β_n are the Lagrange multipliers ≥ 0. Given these Lagrangian multipliers, the dual formulation is:

$$\max_{\alpha}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \alpha_n y_n (\mathbf{x}_n^T). \tag{2.24}$$

In case of SVM without noise:

$$\begin{aligned} &\alpha_i n \geq 0, \quad n = 1, ..., N \\ &\sum_{n=1}^{N} \alpha_n y_n = 0. \end{aligned} \tag{2.25}$$

In case of SVM with noise:

$$\begin{aligned} &0 \leq \alpha_n \leq C, \quad n = 1, ..., N \\ &\sum_{n=1}^{N} \alpha_n y_n = 0. \end{aligned} \tag{2.26}$$

In a dual formulation, the objective function remains the same while the constraints vary depending whether or not the noise factor is taken into account. If there is noise, the SVM value $\alpha_n \in [0, C]$. The SVM classifier with noise is commonly called as *soft SVM* because it lacks a hard decision boundary that separates the different classes of data points. The soft margin classification is defined as identifying support vectors. Data points x_n that have non-zero corresponding Lagrangian multipliers α_n serve as the support vectors. The solution to the dual problem is:

$$\begin{aligned} \mathbf{w} &= \sum_{i=n}^{N} \alpha_n y_n x_n \\ b &= y_n(1 - \xi_n) - \sum_{n=1}^{N} \alpha_n y_n x_n. \end{aligned} \tag{2.27}$$

Solution to a classification problem may be formulated as:

$$f(x) = \sum_{n=1}^{N} \alpha_n y_n x_n x + b, \tag{2.28}$$

where x is the test point, n corresponds to those points where $\alpha_n \neq 0$ are the support vectors, and b is the bias. Bias b (2.27) may be found using any support vector. Hence, there is no need to find the value of $\mathbf{w}$ explicitly in order to calculate $f(x)$.

Chapter 3

Using SVM with Kernels

The kernel function defines inner products $\langle \mathbf{x}_n,\ \mathbf{x}_m \rangle$. Let $\phi(\mathbf{x})$ be a mapping of feature vectors from an input space to a feature space:

$$\langle \mathbf{x}_n, \mathbf{x}_m \rangle \rightarrow \langle \phi(\mathbf{x}_n),\ \phi(\mathbf{x}_m) \rangle. \tag{3.1}$$

The input space represents the selected features from the original dataset while the feature space is generated by the mapping $\phi(\mathbf{x})$. An example of mapping $\phi(\mathbf{x}) : \mathbf{R}^2 \rightarrow \mathbf{R}^3$ is shown in Fig. 3.1, where:

$$(x_i, x_j) \rightarrow (x_i^2, \sqrt{2} x_i x_j, x_j^2). \tag{3.2}$$

If $\mathbf{x}(x_1, x_2)$ and $\mathbf{x}'(x_1', x_2')$ are two feature vectors (data points) in the input space, then:

$$\begin{aligned} &\langle \phi(x_1, x_2), \phi(x_1', x_2') \rangle = \\ &\langle ({x_1}^2, \sqrt{2} x_1 x_2, x_2^2), ({x_1'}^2, \sqrt{2} x_1' x_2', {x'}_2^2) \rangle = \\ &(x_1 x_1' + x_2 x_2')^2 = (\langle x, x' \rangle)^2 = k(x, x'), \end{aligned} \tag{3.3}$$

where k is a kernel function. Instead of calculating each $\phi(\mathbf{x})$, the “kernel trick" is introduced to directly calculate the inner product in the input space:

$$k(\mathbf{x}_n, \mathbf{x}_m) = \langle \phi(\mathbf{x}_n),\ \phi(\mathbf{x}_m) \rangle. \tag{3.4}$$

The mapping (3.4) defines the feature space and generates a decision boundary for input data points. Using the “kernel trick” reduces the complexity of the optimization problem that now only depends on the input space instead of the feature space [50]. If the dataset is nonlinearly separable, the original feature space or the input features x are transformed into a new feature space $\phi(x)$. The training data points are usually mapped to high dimensional feature space to make them linearly separable. The computational costs for such a transformation are very high [18]. For input feature space having D dimensions, the computational time required is $O(D^2)$ and the computational cost increases with dimension D. Using kernel functions achieves the transformation without increasing the computational cost.

As shown in Fig. 3.1, the graph on the left-hand side shows two attributes (x_1, x_2), the regular data points (circles) lie inside while the anomalous data points (stars) lie outside. No line can be drawn to separate them into two distinct classes. However, if the same set of data points are transformed into another feature space having two dimensions (x_1^2, x_2^2), the regular data points (circles) and anomalous data points (stars) become linearly separable, as shown on the right-hand side of Fig. 3.1.

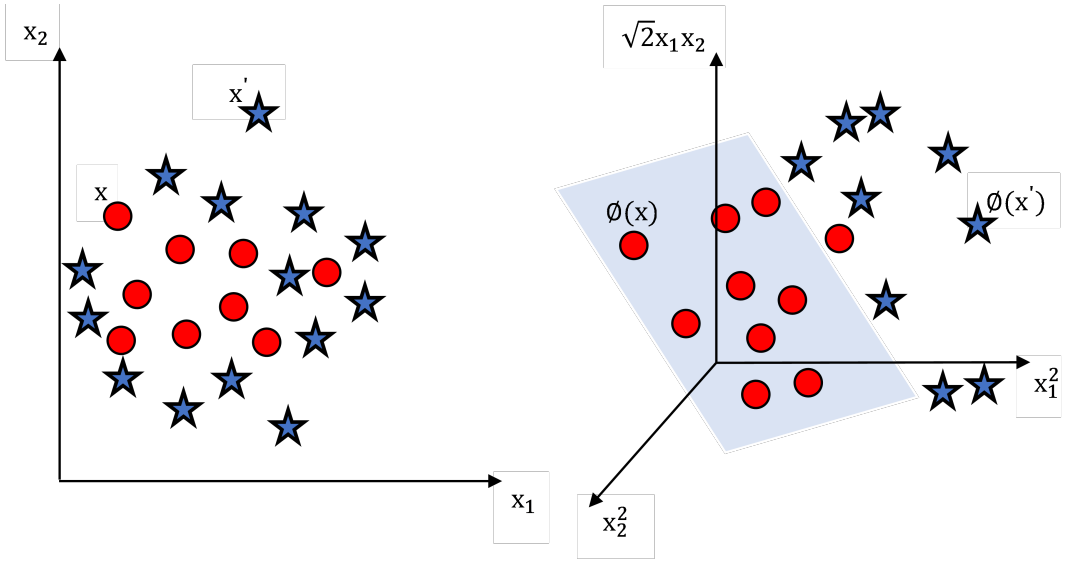

Figure 3.1: Illustration of SVM using the nonlinear kernel function $k(\mathbf{x}_1, \mathbf{x}_2) = \langle \mathbf{x}_1, \mathbf{x}_2 \rangle^2$. View in the three-dimensional space shows a hyperplane dividing regular (circles) and anomalous (stars) data points.

Suppose that x and x′ are original features that are transformed to $\phi(\mathrm{x})$ and $\phi(\mathrm{x}')$, respectively. To solve SVM in this new feature space, the scalar product $\phi(\mathrm{x}) \cdot \phi(\mathrm{x}')$ is required. For certain functions (ϕ), there is a simple operation on two vectors in the lower dimensional space that may be used to compute the scalar product of their two mappings in the higher dimensional space:

$$\phi(\mathrm{x}) \cdot \phi(\mathrm{x}') = k(\mathrm{x}, \mathrm{x}'). \tag{3.5}$$

There is no need to individually solve or expand the original ϕ (x) and ϕ (x′). Functions such as ϕ are called kernel functions. The following steps solve SVM using kernel functions:

- **Step 1**: Original input attributes are mapped to a new set of input features via feature mapping ϕ.
- **Step 2**: The algorithm may be now applied using the scalar product: replace x · x′ with ϕ (x) · ϕ (x′).
- **Step 3**: For certain ϕ, there is a simple operation on two vectors in the lower dimensional space that may be used to compute the scalar product of their two mappings in the higher

dimensional space:

$$k(\mathrm{x}, \mathrm{x}') = \phi(\mathrm{x}) \cdot \phi(\mathrm{x}'). \tag{3.6}$$

The scalar product in the high dimensional space ensures the lower computational costs. Using the feature mapping, the discriminant function may be written as:

$$g(\mathrm{x}) = \mathbf{w}^T \phi(\mathrm{x}) + b = \sum_{i \in SV} \alpha_i \phi(\mathrm{x}_i)^T \phi(\mathrm{x}) + b, \tag{3.7}$$

where i ranges over the support vectors (SV). Consider 2-dimensional vector **x** having two attributes (x_1, x_2). The kernel function may be written as:

$$\begin{aligned} k(\mathrm{x}_1, \mathrm{x}_2) &= (1 + \mathrm{x}_1, \mathrm{x}_2)^2 \\ &= 1 + \mathrm{x}_1^2\mathrm{x}_2^2 + 2\mathrm{x}_1\mathrm{x}_2\mathrm{x}_1'\mathrm{x}_2' + \mathrm{x}_1^{2\prime}\mathrm{x}_2^{2\prime} + 2\mathrm{x}_1\mathrm{x}_1 + 2\mathrm{x}_1'\mathrm{x}_2' \\ &= (\mathrm{x}_1^2\sqrt{2}\mathrm{x}_1\mathrm{x}_1'\mathrm{x}_1^{2\prime}\sqrt{2}\mathrm{x}_1\sqrt{2}\mathrm{x}_1') \cdot (\mathrm{x}_2^2\sqrt{2}\mathrm{x}_2\mathrm{x}_2'\mathrm{x}_2^{2\prime}\sqrt{2}\mathrm{x}_2\sqrt{2}\mathrm{x}_2') \\ &= \phi(\mathrm{x}_1)\phi(\mathrm{x}_2), \end{aligned} \tag{3.8}$$

where $\phi(\mathrm{x}) = \mathrm{x}_1^2\sqrt{2}\mathrm{x}_1\mathrm{x}_2\mathrm{x}_2^2\sqrt{2}\mathrm{x}_1\sqrt{2}\mathrm{x}_2$. Hence, by using the transformation ϕ, the kernel function may be easily computed without expanding the scalar product. Kernel function may be considered as a similarity measure between the input objects. However, not all similarity measures may be used as kernel functions. Only similarity functions that satisfy the Mercer's condition qualify as kernel functions [49]. Mercer's condition states: If similarities between the points are written as a matrix and this matrix is symmetric positive semi-definite, than a kernel function exists. For any positive semi-definite kernel $k(\mathrm{x}_1,\mathrm{x}_2)$, satisfying:

$$\sum_{1,2} k(\mathrm{x}_1, \mathrm{x}_2)\mathrm{c}_1\mathrm{c}_2 \geq 0. \tag{3.9}$$

$k(\mathrm{x}_1,\mathrm{x}_2)$ may be considered as kernel function for any real value of c_1 and c_2 because such functions may be expressed as a dot product in high dimensional space. Some commonly used kernel functions are linear, polynomial, quadratic, cubic, Gaussian radial-basis function, and sigmoid. The choice of the kernel depends on the type of input data. For example, Gaussian kernel defines a function space that is much larger than the function space defined by the linear or polynomial kernels. Linear and polynomial kernels have limited complexity and fixed size. Hence, the SVM model using linear or polynomial kernel converges much faster. Gaussian kernel belongs to the category of nonlinear kernels and its complexity increases with the size of input data. However, it is more flexible than the linear and polynomial kernels.

For a given dataset, there exists a hyperplane that correctly classifies the data points into distinct classes. Indeed, in most cases there is a number of such hyperplanes. With many choices, there is always a question of the best selection of the hyperplane. The necessary condition is to ensure correct classification of the training data. The classifier still has to be applied to the test data. Intuitively, the

classifier should be tested using all the kernels functions and, as per statistical learning theory [50], that kernel function should finally be used as a hyperplane classifier that provides the maximum margin between various attributes.

3.1 Linear Kernel

Linear SVM corresponds to linear kernel where $\phi(\mathrm{x}) = \mathrm{x}$ and $\phi(\mathrm{x}') = \mathrm{x}'$. Often used for text classification, the linear kernel optimization problem is much faster compared to others [49]. It is typically used when the number of features is large and nonlinear mapping fails to improve the performance due to large feature space. As shown in Fig. 3.2, decision boundary separates both correctly and incorrectly classified data points.

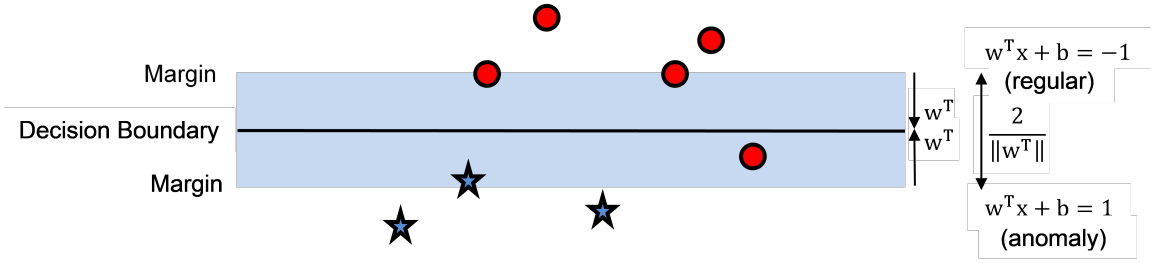

Figure 3.2: Illustration of SVM with linear kernel: Shown are correctly classified regular (circles) and anomalous (stars) data points as well as one incorrectly classified regular (circle) data point.

Finding a suitable class for thousands of training data is often very time consuming. To reduce the running time, two tuning parameters (regularization parameter and gamma) are introduced in SVM classifiers. By fine tuning the two tuning parameters, nonlinear classification may be achieved within reasonable time. Regularization parameter sets the limit on the number of misclassifications that may be tolerated for each training example. For higher values of the regularization parameter, the optimizer will choose a small margin hyperplane. The gamma parameter defines the influence a given training example exerts: smaller values imply higher influence while larger values imply smaller influence.

3.2 Nonlinear SVM Kernels

Soft SVM enabled accounting for the noise and finding the classifiers that do not precisely separate different sets of data points. However, the classification decision surface is still linear and cannot handle cases where the decision surface is nonlinear. To develop an input-output functional relationship, the easiest and straightforward approach is to consider the output variable as a linear combination of its input vector components [42]. The linear combination may use the least square methods for regression and classification problems. Although simple, linear models have limited classifica-

tion power. Nonlinear models give more enhanced predicted outputs. However, these models have two significant problems:

- It is rather difficult to analyze nonlinear models.
- They require large computational infrastructure to achieve a satisfactory solution.

The optimal solution for a linear-nonlinear dilemma is to utilize advantageous features of both models. Instead of using only nonlinear models, problems are modeled so that the linear character of the problem remains intact while and the features are modified to increase the classification power of the linear methods. Nonlinear function feature components are transformed using various kernels so that nonlinear functional relationship may be considered as a linear function.

3.2.1 Polynomial Kernel

Polynomial kernel employs polynomial function to the input variables to find the similarity in training vectors. These types of nonlinear kernels are extensively used for problems where training data is normalized. Its generalized form is defined as:

$$K(\mathrm{x}_i, \mathrm{x}_j) = (1 \ + \ \mathrm{x}_i, \mathrm{x}_j)^p, \tag{3.10}$$

where p is the degree of the polynomial. For example, $p = 2$ and $p = 3$ for a quadratic and cubic kernels, respectively. A polynomial kernel is often used for visual pattern recognition and natural language processing classifications.

A degree of the polynomial kernel governs the flexibility of the resulting classifier. For $p = 1$, the classifier has a linear kernel that is not sufficient to occupy the nonlinear feature space. The function space of the problem increases with the degree of polynomial. Polynomial kernels with degrees 1 and 2 are shown in Fig. 3.3.

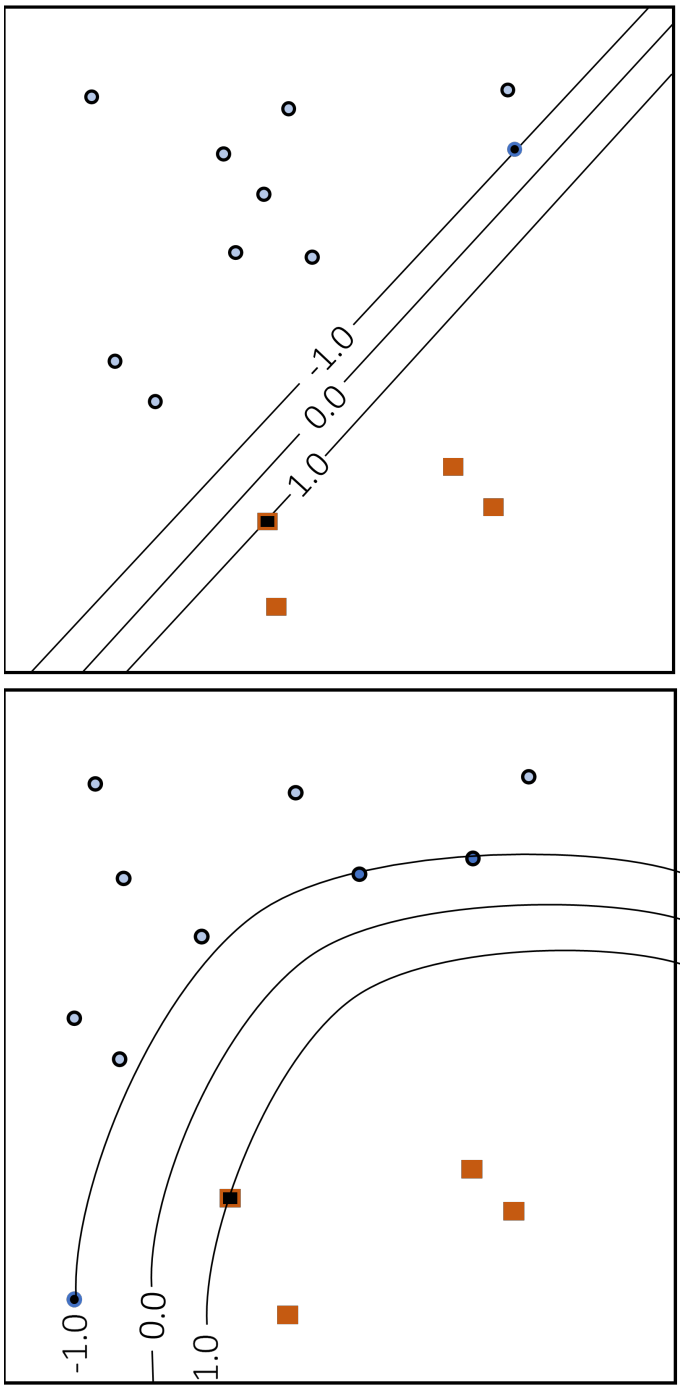

Figure 3.3: Polynomial kernels for $p = 1$ (top) and $p = 2$ (bottom).

For a p^{th} order polynomial kernel, all derivatives of order $p + 1$ are constant and, subsequently, all derivatives of order $p + 1$ and higher are zero.

3.2.1.1 Quadratic Kernel

Each type of classifier and the kernel generates a distinct decision boundary. For example, the decision boundary generated using quadratic kernel is quite different from the shape of a simple quadratic function in 2-dimensional space. Using the quadratic kernel, one may visualize the given data points being lifted to fit into the shape of a quadratic function and forming a shape of the ellipse if cut by the plane. Suppose there are only two features (x_1, x_2). Quadratic kernel expands the two features into five $(x_1, x_2, x_1^2, x_2^2, x_1x_2)$. The decision boundary in such a case may be given by:

$$\mathbf{w}^T\phi(x) + b = 0, \tag{3.11}$$

where $\mathbf{w}^T$ is a vector, b is bias, and $\phi(x) = (x_1, x_2, x_1^2, x_1, x_2, x_2^2)$.

Quadratic kernel leads to decision boundary containing a mixture of quadratic functions. The decision boundary is defined as $y \mid \sum_i \alpha_i k(x_i, y) = b$. For example, consider the decision boundary and the kernel function $k(x, y) = (x^T y + c)^2$:

$$\begin{aligned}\sum_i \alpha_i (x_i^T y + c)^2 &= \sum_i [\alpha_i (x_i^T y)^2 + 2\alpha_i x_i^T y + \alpha_i c^2] \\ &= \sum_i \alpha_i y^T x_i x_i^T y + \Big(\sum_i 2\alpha_i x_i\Big)^T y + c^2 \sum_i \alpha_i \\ &= y^T \Big(\sum_i \alpha_i x_i x_i^T y\Big) y + \Big(\sum_i 2\alpha_i x_i\Big)^T y + c^2 \sum_i \alpha_i,\end{aligned} \tag{3.12}$$

where c is a trade-off parameter to balance the violation of the margin.

3.2.1.2 Cubic Kernel

Cubic kernel is defined as a third-order polynomial function. Using the cubic kernel, one may visualize the given data points being lifted to fit into the shape of a cubic function and forming a shape of the cube if cut by a plane. Suppose there are only two features (x_1, x_2). The decision boundary in such a case may be given by:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + \beta_4 x_2^2 + \beta_5 x_2^3 = 0. \tag{3.13}$$

The equation for SVM model trained with cubic kernel is:

$$K(\mathrm{x}_1, \mathrm{x}_2) = (\mathrm{x}_1^T \mathrm{x}_2 + 1)^3, \tag{3.14}$$

where K is the kernel function.

3.2.2 Gaussian Radial Basis Function Kernel

The Gaussian radial basis function (RBF) kernel is the most widely used kernel function in SVM classification. It is used when there exists no prior knowledge about the input data [38]. It is defined as:

$$K(\mathrm{x}_i, \mathrm{x}_j) = \exp\left[-\frac{||\mathrm{x}_i - \mathrm{x}_j||^2}{2\sigma^2}\right]. \tag{3.15}$$

Gaussian RBF kernel is a stationary kernel, invariant to translations. For a single parameter, it exhibits isotropic property, which implies that scaling of one parameter leads to automatic scaling of all other parameters. The adjustable parameter σ is tuned according to the nature of the problem. If set to a higher value, the kernel behaves almost linearly, causes overestimation of the problem, and the nonlinear higher dimensional projection no longer holds. Similarly, if set to a very smaller value, regularization function is affected and this underestimation makes decision boundary sensitive to noise in the training data.

The parameter σ plays the same role in Gaussian RBF kernel as p in the polynomial kernel. It controls the flexibility of the classifier, as shown in Fig. 3.4.

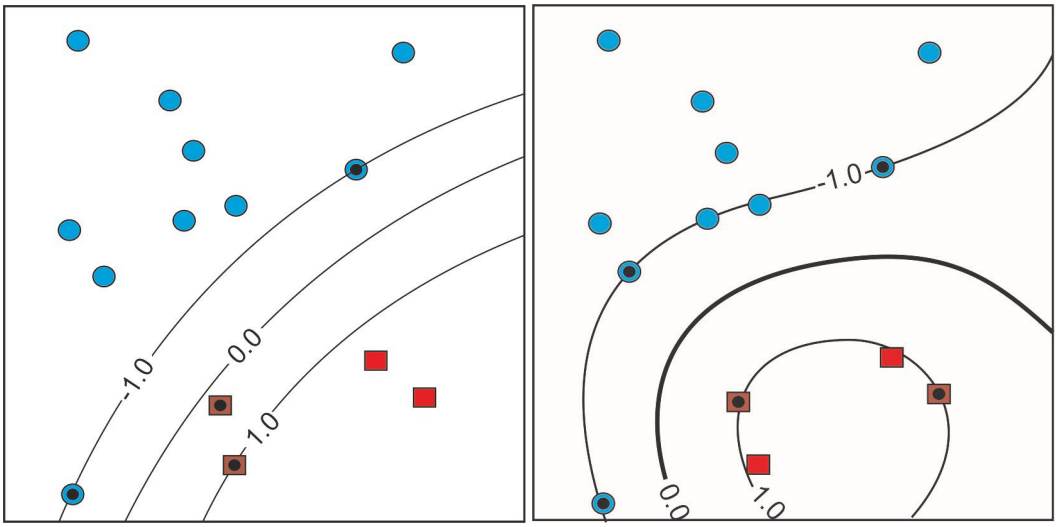

Figure 3.4: Gaussian kernels for $\sigma = 20$ (left) and $\sigma = 1$ (right).

The discriminator function distinguishes the whether or not the input is linearly separable. The output of Gaussian RBF kernel is zero if $||\mathrm{x}_i - \mathrm{x}_j||^2 \gg \sigma$. In other words, if the value of x_j is fixed, there exists a region around x_j having a large kernel value. In such cases, the resultant discriminator function is the algebraic sum of Gaussian areas centered around the support vectors. For large values of σ shown in Fig. 3.4, non-zero kernel value is assigned to a given data point and all the support vectors together affect the discriminator function thereby generating a smooth decision boundary. In case the value of σ is reduced, the decision surface becomes more curvy with kernel function more localized. For smaller values of σ, the discriminator function becomes non-zero only in the

areas lying in the vicinity of each support vector. Hence, in such cases, the discriminator function is constant outside of regions where the data points are concentrated.

3.2.3 Sigmoid Kernel

The sigmoid kernel originated from the neural networks principles [36]:

$$k(\mathrm{x}_i, \mathrm{x}_j) = \tanh(\beta_o \mathrm{x}_i \cdot \mathrm{x}_j - \beta_1), \tag{3.16}$$

where β_o and β_1 are control parameters calculated through primal-dual relationship. Parameter β_o is a scaling parameter of the input data and β_1 is a moving parameter that defines the threshold of mapping. When $\beta_o < 0$, the input data dot product is reversed. This implies that it is more appropriate to use the sigmoid kernel when $k > 0$ and $\beta_o < 0$. A positive semi-definite kernel represents the inner product in some higher-dimensional space. In practice, sigmoid kernel is not a positive semi-defined function. The sigmoid kernel matrix might not be positive semi-defined for certain values of β_o and β_1 [19]. The sigmoid kernel has been successfully used in many practical applications and, for certain parameters, it behaves similarly to Gaussian kernel [49].

Parameters β_o and β_1 control the operation of the sigmoid kernel, as shown in Table 3.1. For small values of β_1, the sigmoid kernel behaves as a conditionally positive defined function. Non-positive semi defined nature of the sigmoid kernel makes it unsuitable for use in many scenarios [49]. Even though sigmoid kernel may be conditionally positive defined and may provide a solution in such cases, local minimum solution for all parameters may never be guaranteed, which makes it is impossible to optimize all the input parameters at the same time.

Table 3.1: Behaviour of sigmoid kernel

	β_o	β_1	Result
Case 1	Positive	Negative	Kernel is conditionally positive defined for small values of β_o
Case 2	Positive	Positive	Inferior to Case 1
Case 3	Negative	Positive	Separability of kernel matrix not clear as objective value goes to $-\infty$ for higher values of β_1
Case 4	Negative	Negative	Objective value $-\infty$ for all the cases

Chapter 4

BGP Dataset Used for Detecting Network Anomalies

Datasets play a crucial role in the field of machine learning. The quality of the dataset governs the learning process and the classification results. Reliable machine learning results may only be achieved with the availability of highly reliable training datasets. High quality indexed training datasets used for semi-supervised and supervised algorithms are difficult to collect due to the nature of the data. The cost of a dataset includes the cost of obtaining the raw data, labeling and cleaning the data, and storing and transferring various fields into machine-readable form [6]. Although the labeling of datasets is not required for unsupervised learning, high-quality data still remains difficult and costly to produce.

A dataset is imbalanced when a class is represented by a smaller number of training samples compared to other classes. Most classification algorithms minimize the number of incorrectly predicted class labels while ignoring the difference between types of misclassification by assuming that all misclassification errors have equal costs. As a result, a classifier that is trained using an imbalanced dataset may successfully classify the majority class with a good accuracy while it may be unable to accurately classify the minority class. The assumption that all misclassification types are equally costly is not valid in many application domains. In the case of BGP anomaly detection, incorrectly classifying an anomalous sample is more costly than incorrect classification of a regular sample. Various approaches have been proposed to achieve accurate classification results when dealing with imbalanced datasets. Examples include assigning a weight to each class or learning from one class (recognition-based) rather than two classes (discrimination-based). The recognition-based approach is an alternative solution when the classifier is modeled using the examples of the target class in the absence of examples of the non-target class. However, recognition-based approaches cannot be applied to many machine learning algorithms such as decision tree, naïve Bayes, and associative classifiers. The weighted Support Vector Machines (SVMs) assign distinct weights to data samples so that the training algorithm learns the decision surface according to the relative importance of data points in the training dataset.

4.1 Border Gateway Protocol

Border Gateway Protocol (BGP) is a path vector protocol whose main function is to optimally route data between Autonomous Systems (ASes). An AS is a collection of BGP routers (peers) within a single administrative domain. BGP relies on Transmission Control Protocol (TCP), using port 179 for reliable router–to–router communication with only four message types (open, keepalive, update, and notification). Due to this simple design, BGP is prone to malicious attacks. In most cases, an attacked router advertises fraudulent information via BGP thus causing a large scale redirection of the Internet traffic [9], [15]. BGP anomalies include the worm attacks such as Slammer [7], Nimda [5], and Code Red I (version 2) [3] as well as routing misconfigurations, Internet Protocol (IP) prefix hijacks, and electrical failures. Statistical approaches have been extensively used for detection of BGP anomalies [53]. However, they are not suitable to detect anomalies having high number of features. Rule-based detection techniques are based on prior knowledge of network conditions. They are not adaptable learning mechanisms, are slow, have high degree of computational complexity, and require a prior knowledge. Routing decisions between the autonomous systems are made depending on the available paths and network policies as defined by the network administrator. BGP supports two kinds of protocols:

- Interior Gateway Protocol (IGP) when BGP is used for routing within an autonomous system.
- Exterior Gateway Protocol (EGP) when BGP is used for routing among autonomous systems such as Internet service providers.

BGP is extensively used for Internet services and is also known as *de-facto* Internet routing protocol. The performance of BGP is tuned using BGP attributes. Important BGP attributes are shown in Table 4.1. BGP uses Autonomous System (AS) number to determine the next hop when routing packets. Even though BGP is a slow routing protocol, it is very reliable and robust.

4.2 Border Gateway Protocol Attributes and Messages

The metrics used by BGP are called path attributes and may be divided into two broad categories: *well-known* and *optional*. Well-known mandatory BGP attributes should be recognized and understood by all BGP implementations. They are part of every BGP update messages sent by an autonomous machine. Well-known discretionary attributes are compulsorily understood by all BGP implementations but are not necessarily part of every BGP update message [48]. Optional transitive BGP attributes do not require to be understood by all BGP implementations. Nevertheless, they are labeled with a transitive flag and all neighboring BGP implementations should have the capability to forward the information to the intended destination [53].

Table 4.1: List of BGP attributes

Attribute Name	Category
origin	well-known mandatory
AS-PATH	well-known mandatory
next_hop	well-known mandatory
local_pref	well-known discretionary
atomic_aggregate	well-known discretionary
aggregator	optional transitive
community	optional transitive
multi_exit_disc (MED)	optional non-transitive
originator_ID	optional non-transitive
cluster_list	optional non-transitive
DPA	designation point attribute
advertiser	BGP/IDRP route server
cluster_ID	BGP/IDRP route server
multiprotocol_reachable_NLRI	optional non-transitive
multiprotocol_unreachable_NLRI	optional non-transitive

Inter-domain Routing Protocol (IDRP) specifies how routers communicate with routers in other domains. BGP messages are categorized as AS-*path* or *volume* features, as shown in Table 4.2 [28]. They are BGP update message attributes that enable the protocol to select the best path for routing packets. We filter the collected traffic for BGP update messages during the time period when the Internet experienced anomalies. We use Zebra tool [8] to parse the ASCII files to extract the features. If a feature is derived from an AS-*path* attribute, it is categorized as an AS-*path* feature. Otherwise, it is categorized as a *volume* feature.

Table 4.2: List of features extracted from BGP update messages

Feature	Definition	Type	Category
1	Number of announcements	continuous	volume
2	Number of withdrawals	continuous	volume
3	Number of announced NLRI prefixes	continuous	volume
4	Number of withdrawn NLRI prefixes	continuous	volume
5	Average AS-PATH length	categorical	AS-PATH
6	Maximum AS-PATH length	categorical	AS-PATH
7	Average unique AS-PATH length	continuous	volume
8	Number of duplicate announcements	continuous	volume
9	Number of duplicate withdrawals	continuous	volume
10	Number of implicit withdrawals	continuous	volume
11	Average edit distance	categorical	AS-PATH
12	Maximum edit distance	categorical	AS-PATH
13	Inter-arrival time	continuous	volume
14-24	Maximum edit distance = n, where n = (7;...; 17)	binary	AS-PATH
25-33	Maximum AS-PATH length = n, where n = (7;...;15)	binary	AS-PATH
34	Number of Interior Gateway Protocol packets	continuous	volume
35	Number of Exterior Gateway Protocol packets	continuous	volume
36	Number of incomplete packets	continuous	volume
37	Packet size (B)	continuous	volume

4.3 Description of Anomalous Events

In this Thesis, we consider anomalous events such as worms, power outages, and BGP router configuration error events. We use datasets collected during Slammer, Nimda, Code Red I (version 2), Moscow power blackout affecting the Moscow Internet Exchange (MSIX) [25] [26], and AS 9121 routing table leak [21], as listed in Table 4.3. Two types of rrc (remote route collectors), rrc04 and rrc05, are considered.

Table 4.3: Details of the anomalous events

Event	**Date**	**RRC**	**Peers that sent the messages**
Slammer	Jan. 2003	rrc04	AS 513, AS 559, AS 6893
Nimda	Sept. 2001	rrc04	AS 513, AS 559, AS 6893
Code Red I (version 2)	July 2001	rrc04	AS 513, AS 559, AS 6893
Moscow power blackout	May 2005	rrc05	AS 1853, AS 12793, AS 13237
AS 9121 routing table leak	Dec. 2004	rrc05	AS 1853, AS 12793, AS 13237

The Internet routing data used in this Thesis to detect BGP anomalies are acquired from the Réseaux IP Européens (RIPE) [6] Network Coordination Centre (NCC). Regular data from RIPE were collected in December 2001 and reflect higher traffic volume due to the historical growth of the Internet. The RIPE collects and stores chronological routing data that offer a unique view of the Internet topology. The BGP update messages are publicly available to the research community in the multi-threaded routing toolkit (MRT) binary format. The Internet Engineering Task Force (IETF) introduced MRT [39] to export routing protocol messages, state changes, and content of the routing information base (RIB). We filter the collected traffic for BGP update messages during the time period when the Internet experienced anomalies. Their traffic traces are shown in Fig. 4.1-4.4. We use 37 features extracted from BGP update messages originated from rrc04 and rrc05. Five-day periods are considered: the day of the attack as well as two days prior and two days after the attack [13]-[11].

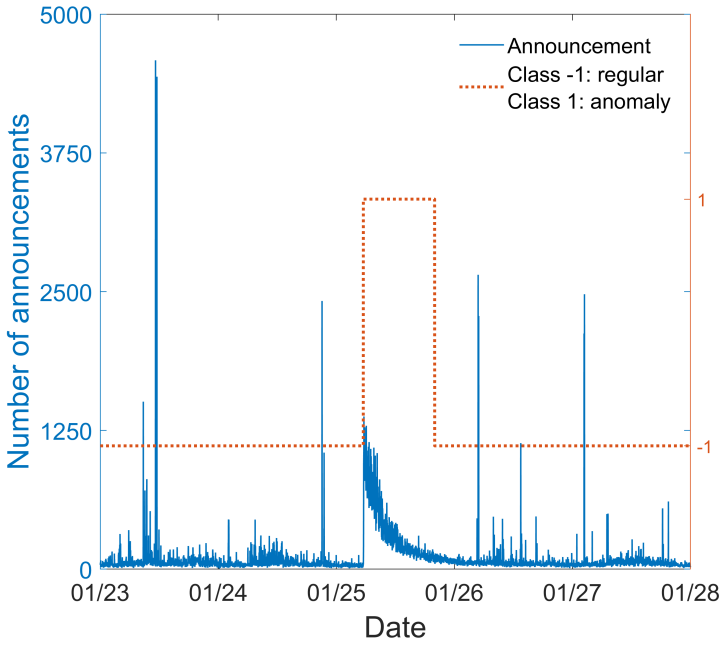

Figure 4.1: Number of BGP announcements between January 23, 2003 and January 28, 2003. The announcements occurred during the Slammer worm attack are labelled as the "anomaly" class while others are labelled as the "regular" class.

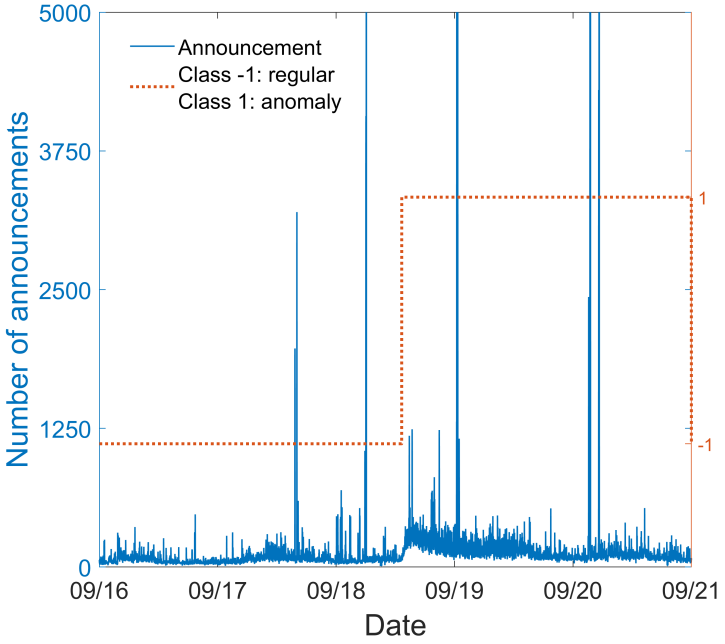

Figure 4.2: Number of BGP announcements between September 16, 2001 and September 21, 2001. The announcements occurred during the Nimda worm attack are labelled as the "anomaly" class while others are labelled as the "regular" class.

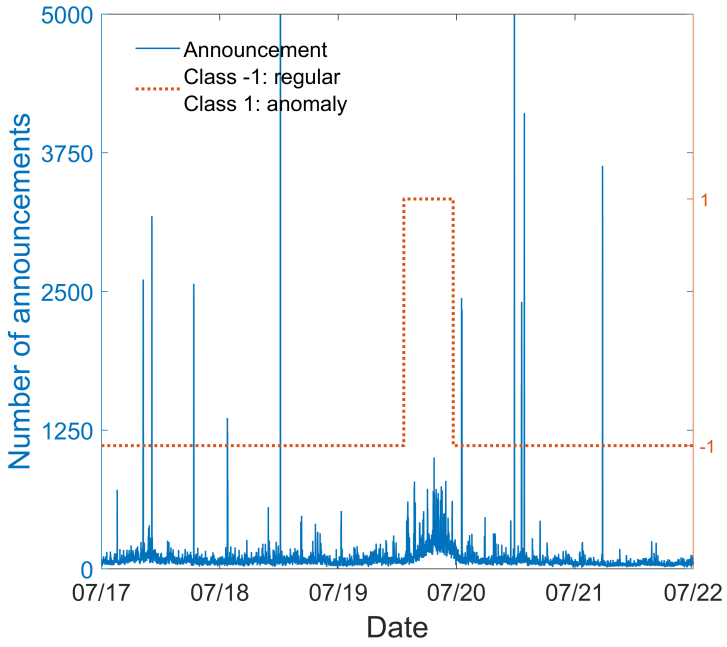

Figure 4.3: Number of BGP announcements between July 17, 2001 and July 22, 2001. The announcements occurred during the Code Red I (version 2) worm attack are labelled as the "anomaly" class while others are labelled as the "regular" class.

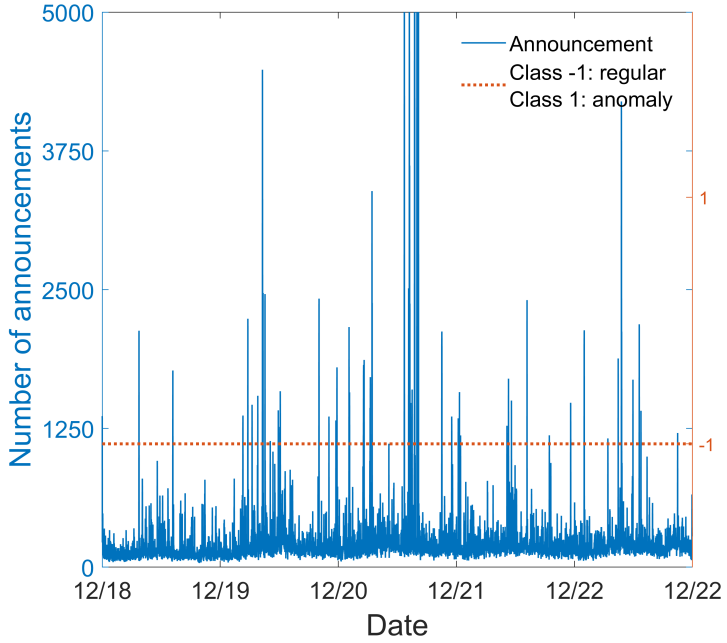

Figure 4.4: Number of BGP announcements between December 18, 2001 and December 22, 2001. The announcements occurred during regular RIPE traffic belong to the "regular" class.

The Structured Query Language (SQL) Slammer worm attacked Microsoft SQL servers on January 25, 2003. The Slammer worm is a code that generates random IP addresses and replicates itself by sending 376 bytes of code to those IP addresses. If the destination IP address is a Microsoft SQL server or a user's PC with the Microsoft SQL Server Data Engine (MSDE) installed, the server becomes infected and begins infecting other servers. The number of infected machines doubled approximately every 9 seconds. As a result, the update messages consume most of the routers' bandwidth, which in turn slows down the routers and in some cases causes the routers to crash.

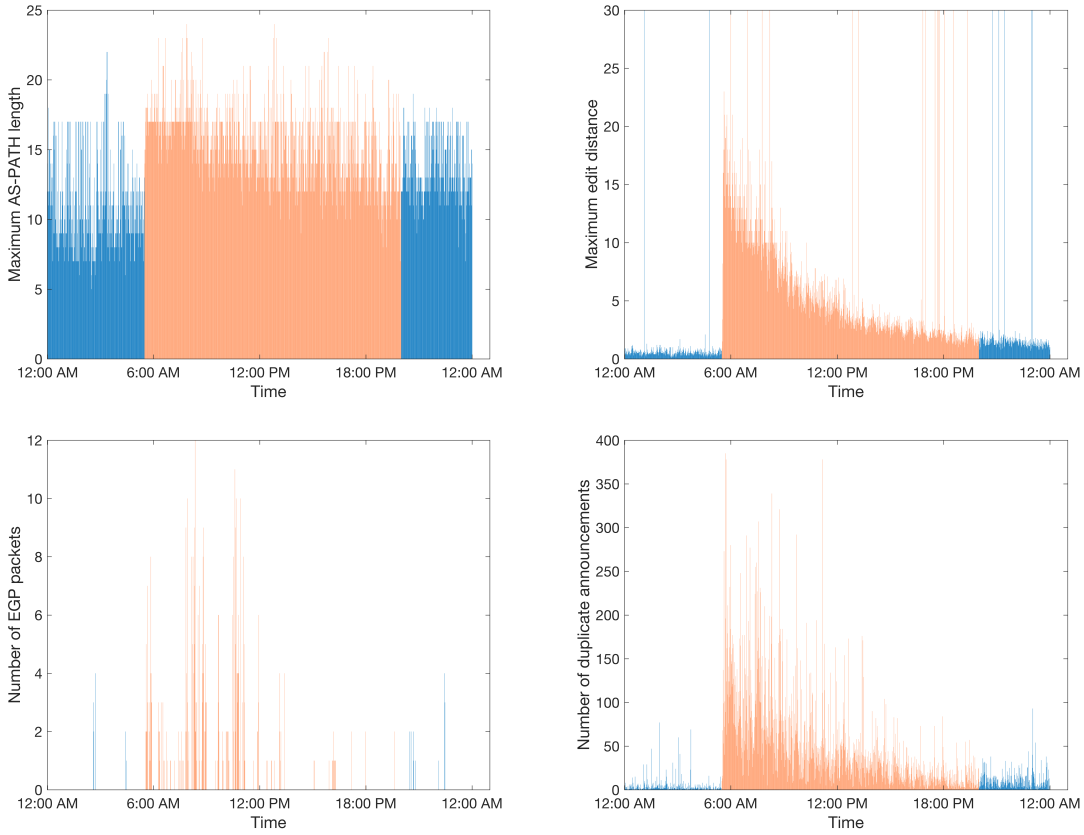

Figure 4.5: BGP announcements during Slammer: maximum AS-path length (top left), maximum AS-path edit distance (top right), number of EGP packets (bottom left), and number of duplicate announcements (bottom right) [28].

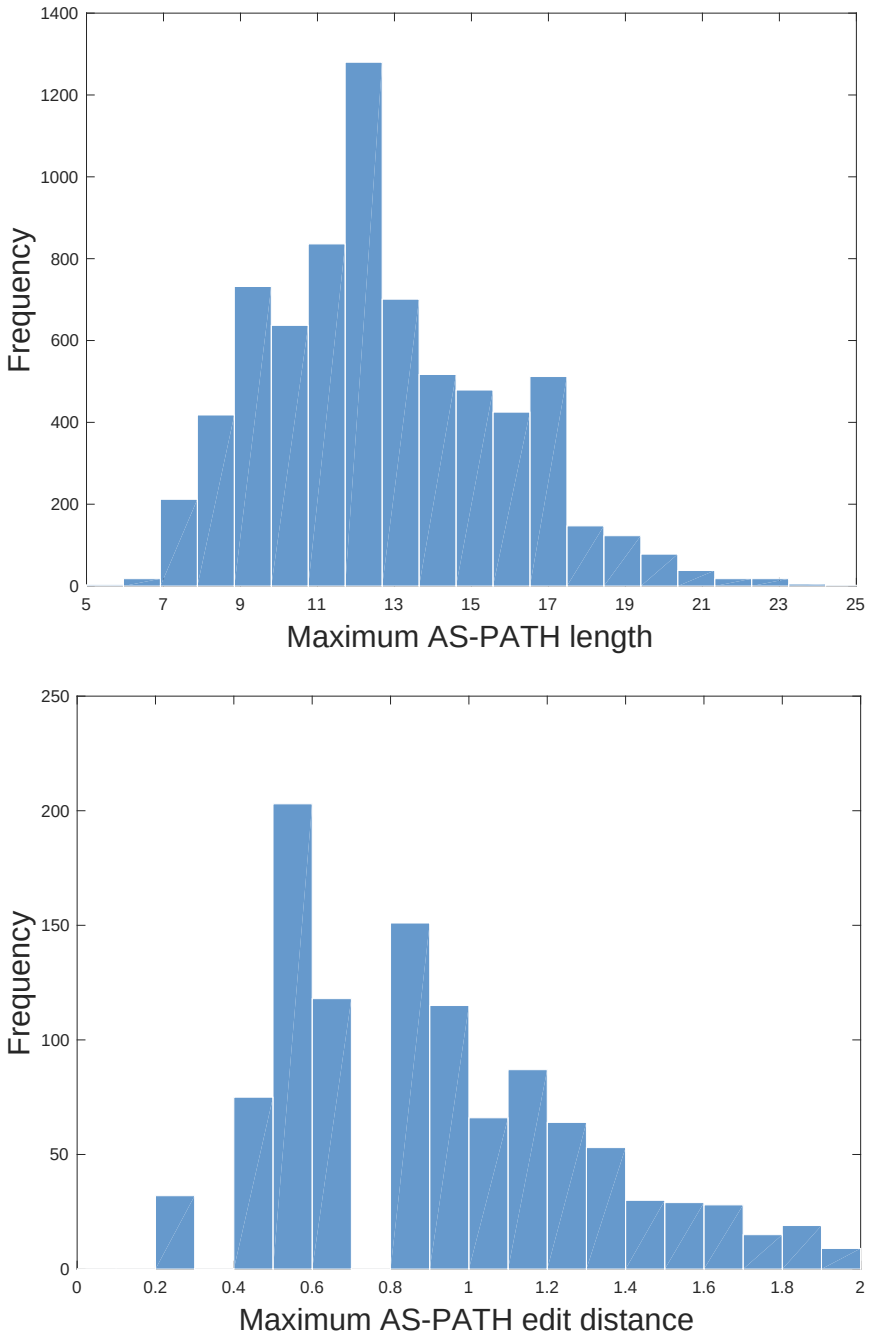

Figure 4.6: Distribution of maximum AS-path length (top) and maximum AS-path edit distance (bottom) collected during the Slammer worm attack [28].

The Nimda worm was released on September 18, 2001. The worm propagated by sending an infected attachment that was automatically downloaded once the email was viewed. It propagates fast through email messages, web browsers, and file systems. Viewing the email message triggers the worm payload. The worm modifies the content of the web document file in the infected hosts and

copies itself in all local host directories. Nimda exploited vulnerabilities in the Microsoft Internet Information Services (IIS) web servers for Internet Explorer 5.

The Code Red I (version 2) worm attacked Microsoft Internet Information Services (IIS) web servers on July 19, 2001. The worm affected approximately half a million IP addresses a day. It takes advantage of vulnerability in the IIS indexing software. It triggers a buffer overflow in the infected hosts by writing to the buffers without checking their limits. Unlike the Slammer worm, Code Red I (version 2) searched for vulnerable servers to infect. The rate of infection was doubling every 37 minutes.

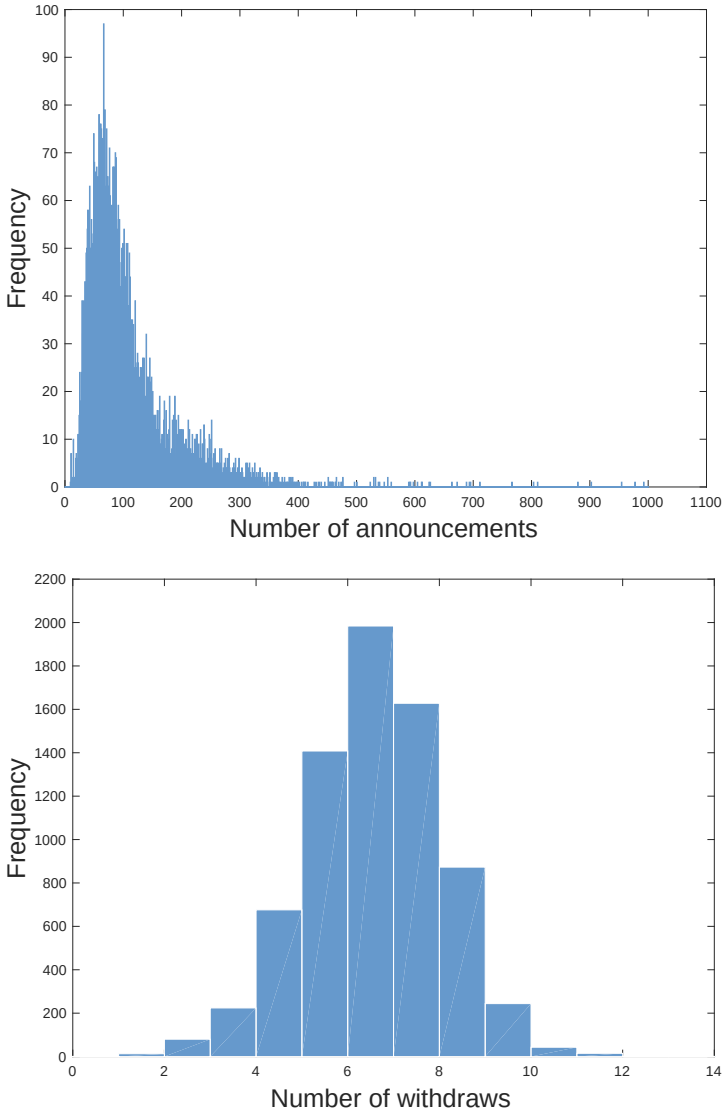

Figure 4.7: Distribution of number of BGP announcements (top) and withdrawals (bottom) for the Code Red I (version 2) worm attack [28].

Moscow power blackout occurred on May 25, 2005 and lasted several hours. The Moscow Internet exchange (MSIX) was shut down during the blackout. Routing instabilities were observed

due to the loss of connectivity of some ISPs peering at MSIX. The volume of announcements and withdrawal messages received at the RIPE rrc05 surged during the blackout.

AS 9121 routing table leak occurred on December 24, 2004 when the AS 9121 announced to the peers that it could be used to reach almost 70% of all the prefixes (over 106,000) [23]-[21]. Hence, numerous networks had either misdirected or had lost their traffic. The AS 9121 started announcing prefixes to the peers around 9:20 GMT and the event lasted just after 10:00 GMT. The AS 9121 continued to announce bad prefixes throughout the day and the announcement rate had reached the second peak at 19:47 GMT.

4.4 Feature Selection

Feature selection reduces redundancy among features and improves the classification accuracy. Feature selection is used to preprocess data prior to applying machine learning algorithm for classification. The combination of extracted features affects classification results. For example, the scatterings of regular and anomalous classes for Feature 9 (volume) vs. Feature 6 (*AS-path*) and vs. Feature 1 (*volume*) are shown in Fig. 4.8 (top) and (bottom), respectively [11]. The graphs indicate that using Feature 9 and Feature 6 would lead to a poor classification while using Feature 9 and Feature 1 may lead to a feasible classification.

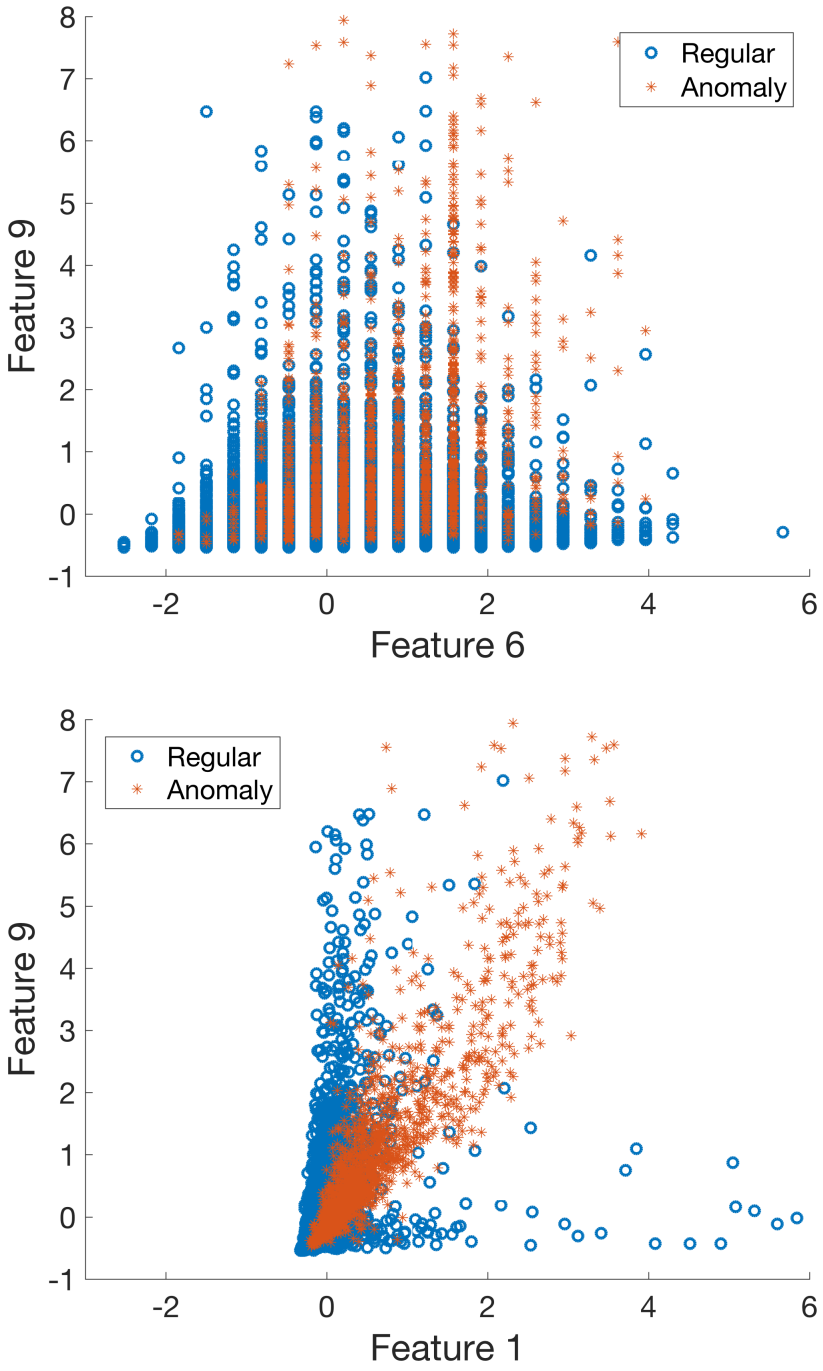

Figure 4.8: Scattered graph of Feature 9 vs. Feature 6 (top) and Feature 1 (bottom) extracted from the BCNET traffic [28].

Decision tree is one of the most successful techniques for supervised classification learning [47] because it may handle both numerical and categorical features. We use decision tree algorithm for feature selection implemented in the publicly available software tool C5.0 [2]. Some features are removed from the constructed tree because they are repeatedly used [33]. A decision or a classification tree is a directed tree where the root is the source sample set and each internal (non-leaf) node is labeled with an input feature to perform a test. Branches emanating from a node are labeled with all possible values of a feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes. A tree may be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated for each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node contains all values of the target variable or when the splitting no longer adds value to the predictions. After a decision tree is learned, each path from the root node (source set) to a leaf node may be transformed into a decision rule. Therefore, a set of rules may be obtained by a trained decision tree that may be used for classifying unseen samples.

Chapter 5

Comparison of SVM Kernels

We apply SVM as a binary classifier for a classification rather than regression tasks. Given a set of labeled training samples, the SVM algorithm learns a classification hyperplane (decision boundary) by maximizing the minimum distance between data points belonging to various classes. In classification problems, unbalanced datasets are very frequently used. They have disproportionately high number of examples from one class. Since classifiers that are evaluated using unbalanced dataset tend to be biased towards one class, we use only balanced training datasets to evaluate SVM models with linear, polynomial, quadratic, cubic, Gaussian RBF, and sigmoid kernels [13]. The datasets are trained using 10-fold cross validation to select parameters $(C, 1/2\sigma^2)$ that generate the best accuracy. The classification procedure consists of four steps:

- Step 1: Use 37 features or select the most relevant features using the decision tree algorithm.
- Step 2: Train the SVM with linear, polynomial, quadratic, cubic, Gaussian RBF, or sigmoid kernels.
- Step 3: Test the models using various datasets.
- Step 4: Evaluate performance of the SVM kernels based on accuracy and F-Score.

Confusion matrix 5.1 is used to evaluate performance of classification algorithms.

Table 5.1: Confusion matrix

	Predicted class	
Actual class	Anomaly (negative)	Regular (positive)
Anomaly (positive)	TP	FN
Regular (negative)	FP	TN

True positive (TP) and false negative (FN) are the number of anomalous data points that are classified as anomaly and regular, respectively. Accuracy reflects the true prediction over the entire dataset. However, it assumes equal cost for misclassifications and a relatively uniform distribution of classes. Hence, we also evaluate the F-Score that considers false predictions [28]. F-Score signifies

harmonic mean between precision and sensitivity that further measure the discriminating ability of the classifier to identify classified and misclassified anomalies [34]. The performance measures are:

$$\text{accuracy} = \frac{\text{TP + TN}}{\text{TP + TN + FP + FN}} \tag{5.1}$$

$$\text{F-Score} = 2 \times \frac{\text{precision} \times \text{sensitivity}}{\text{precision + sensitivity}}. \tag{5.2}$$

The precision and sensitivity are defined as:

$$\text{precision} = \frac{\text{TP}}{\text{TP + FP}} \text{ , sensitivity} = \frac{\text{TP}}{\text{TP + FN}}. \tag{5.3}$$

The receiver operating characteristics (ROC) is the plot of the true positive rate against the false positive rate for a predictive model using different probability threshold values between 0.0 and 1.0. The true positive rate is the number of true positives divided by the sum of the number of true positives and the number of false negatives. It describes how good the model can be at predicting the positive class when actual outcome is positive. True positive rate is also called sensitivity (5.3):

$$\text{True positive rate} = \frac{\text{TP}}{\text{TP + FN}}. \tag{5.4}$$

The false positive rate is the number of false positives divided by the sum of the number of false positives and the number of true negatives. False positive rate is also called specificity:

$$\text{False positive rate} = \frac{\text{FP}}{\text{FP + TN}}. \tag{5.5}$$

The value of the area under the curve (AUC) is the most frequently used performance measure extracted from the ROC curve. The ROC curve is a probability curve while AUC represents the degree or measure of separability. AUC is used to infer how capable a model is in distinguishing between the regular and anomalous data points. The higher the AUC, the better the model is at distinguishing between the regular and anomalous data points. The model has a good measure of separability if AUC $\in [0.5, 1]$ while if AUC ≤ 0.5 then the model has a poor measure of separability. We used Matlab 2019a to plot ROC curves and calculate the AUC for SVM with linear, polynomial, quadratic, cubic, Gaussian RBF and sigmoid kernels. The ROC curve for the kernels with highest AUC is shown in Fig. (5.1). AUC for the six evaluated SVM kernels are shown in Table 5.2.

Table 5.2: Area under the curve for the six evaluated SVM kernels

SVM kernel type	AUC
Linear	0.96
Polynomial	0.94
Quadratic	0.80
Cubic	0.75
Gaussian RBF	0.93
Sigmoid	0.70

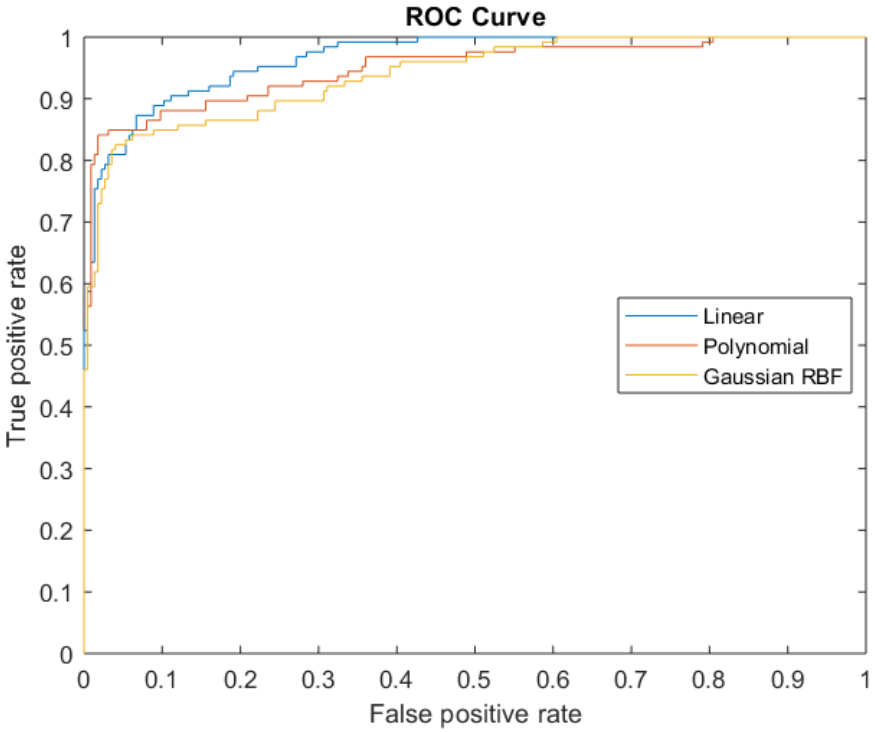

Figure 5.1: ROC curve for the kernels with the highest AUC: Linear (0.96), Polynomial (0.94), and Gaussian RBF (0.93).

Performance of SVM with various kernels is evaluated using combinations of datasets shown in Table 5.3. Experiments were performed using MATLAB 2019a with the Statistics and Machine Learning Toolbox. We measure the performance of SVM based on accuracy and F-Score.

Table 5.3: Training and test datasets

	Training dataset	**Test dataset**
Dataset 1	Slammer and Nimda	Code Red I (version 2)
Dataset 2	Nimda and Code Red I (version 2)	Slammer
Dataset 3	Slammer and Code Red I (version 2)	Nimda
Dataset 4	Slammer	Nimda and Code Red I (version 2)
Dataset 5	Nimda	Slammer and Code Red I (version 2)
Dataset 6	Code Red I (version 2)	Slammer and Nimda
Dataset 7	Slammer and Nimda	Moscow power blackout
Dataset 8	Slammer and Code Red I (version 2)	Moscow power blackout
Dataset 9	Nimda and Code Red I (version 2)	Moscow power blackout
Dataset 10	Slammer and Nimda	AS 9121 routing table leak
Dataset 11	Slammer and Code Red	AS 9121 routing table leak
Dataset 12	Nimda and Code Red I (version 2)	AS 9121 routing table leak
Dataset 13	Moscow power blackout	Slammer and Nimda
Dataset 14	Moscow power blackout	Slammer and Code Red I (version 2)
Dataset 15	Moscow power blackout	Nimda and Code Red I (version 2)
Dataset 16	AS 9121 routing table leak	Slammer and Nimda
Dataset 17	AS 9121 routing table leak	Slammer and Code Red I (version 2)
Dataset 18	AS 9121 routing table leak	Nimda and Code Red I (version 2)

5.1 SVM Algorithm with Linear Kernel

Performance of SVM with linear kernel for all the eighteen datasets is shown in Table 5.4. Dataset 13 based on Moscow power blackout as the training dataset with the combination of Slammer and Nimda as the test dataset outperforms the other datasets when all 37 features are selected. Dataset 9 based on the combination of Nimda and Code Red I (version 2) as the training dataset and Moscow power blackout as the test dataset outperforms other datasets when features are selected using the decision tree feature selection algorithm. For most of the datasets, SVM algorithm with linear kernel gives the best performance when features are selected using the decision tree feature selection algorithm. The best accuracy (85.35%) and F-Score (80.61%) are obtained when the model is trained using Dataset 13 with all 37 features. The second best accuracy (82.63%) and F-Score (79.73%) are obtained when the model is trained using Dataset 9 using the decision tree feature selection algorithm.

Table 5.4: Accuracy and F-Score using SVM with Linear kernel

		Accuracy (%)			F-Score (%)
Selected features	Training dataset	Test	RIPE	BCNET	Test
1-37	Dataset 1	72.76	61.34	54.21	73.60
	Dataset 2	70.81	52.89	45.36	73.19
	Dataset 3	73.36	64.27	56.18	74.62
	Dataset 4	68.91	46.83	42.49	70.85
	Dataset 5	61.03	40.97	38.90	67.40
	Dataset 6	61.28	42.55	39.71	68.07
	Dataset 7	72.77	36.40	45.21	75.62
	Dataset 8	58.57	71.09	40.31	61.57
	Dataset 9	50.64	48.83	34.54	46.39
	Dataset 10	45.47	36.62	38.70	47.51
	Dataset 11	55.09	50.86	55.20	41.44
	Dataset 12	60.08	41.17	40.35	46. 79
	Dataset 13	85.35	72.68	60.14	80.61
	Dataset 14	72.22	54.68	50.31	61.08
	Dataset 15	59.73	50.45	47.80	49.85
	Dataset 16	65.31	63.25	59.55	60.47
	Dataset 17	60.43	54.77	50.23	56.28
	Dataset 18	57.55	53.23	52.07	54.88
1-21, 23-29, 34-37	Dataset 1	74.71	63.26	55.39	76.29
	Dataset 2	73.27	54.12	49.38	74.48
	Dataset 3	70.63	53.89	49.01	72.05
	Dataset 4	69.25	50.13	42.44	68.33
	Dataset 5	66.31	50.78	41.49	65.03
	Dataset 6	69.66	53.41	46.87	69.56
	Dataset 7	48.03	45.18	41.97	50.73
	Dataset 8	52.20	49.38	40.07	43.97
	Dataset 9	82.63	76.35	71.77	79.73
	Dataset 10	64.66	58.35	51.23	55.98
	Dataset 11	59.64	55.49	43.69	40.83
	Dataset 12	45.88	40.25	44.15	41.59
	Dataset 13	69.20	64.80	64.01	61.23
	Dataset 14	43.99	40.14	49.43	43.84
	Dataset 15	60.90	55.97	59.38	52.67
	Dataset 16	45.37	44.84	37.15	42.85
	Dataset 17	53.70	52.10	47.22	50.63
	Dataset 18	54.83	50.34	49.45	51.39

5.2 SVM Algorithm with Nonlinear Kernels

Results obtained using SVM with nonlinear kernels are shown in Table 5.5 - 5.9. SVM with polynomial kernel shows the best accuracy (76.46%) and F-Score (77.12%) using the decision tree feature selection algorithm when tested using Dataset 9. SVM with polynomial kernel outperforms SVM with quadratic and sigmoid kernels using all 37 features when tested using Dataset 13 and using the decision tree feature selection algorithm when tested using Dataset 9. SVM with Gaussian RBF kernel gives the best performance when features are selected using the decision tree feature selection algorithm. Polynomial, quadratic, and cubic kernels have better accuracy and F-Score when compared to the sigmoid kernel. Among the five nonlinear SVM kernels, sigmoid kernel obtained

the lowest accuracy (66.45%) and F-Score (65.90%). Gaussian RBF kernel has the highest accuracy (75.84%)and F-Score (76.67%) using all 37 features when tested using Dataset 13. Polynomial kernel has the highest accuracy (76.46%) and F-Score(77.12%) using the decision tree feature selection algorithm when tested using Dataset 9. SVM with Gaussian RBF kernel outperformed other four nonlinear kernels using all 37 features while SVM with polynomial kernel outperformed all the other four nonlinear kernels using the decision tree feature selection algorithm. Using kernels resulted in better accuracy and F-Score using Dataset 13 and Dataset 9 when compared to the remaining 16 datasets.

Table 5.5: Accuracy and F-Score using SVM with Polynomial kernel

		Accuracy (%)			F-Score (%)
Selected features	Training dataset	Test	RIPE	BCNET	Test
1-37	Dataset 1	66.42	59.26	48.19	68.43
	Dataset 2	64.73	46.53	37.27	66.71
	Dataset 3	68.78	60.37	52.41	69.09
	Dataset 4	58.27	50.65	45.56	56.33
	Dataset 5	54.40	44.56	41.87	53.35
	Dataset 6	57.31	49.37	42.75	54.47
	Dataset 7	41.39	36.23	35.47	49.58
	Dataset 8	61.07	56.58	52.43	62.26
	Dataset 9	53.19	46.12	44.99	54.34
	Dataset 10	55.34	50.23	51.18	50.66
	Dataset 11	55.05	54.20	54.26	58.97
	Dataset 12	41.26	38.09	40.43	46.34
	Dataset 13	71.73	63.65	62.33	73.46
	Dataset 14	57.69	49.56	42.17	54.98
	Dataset 15	45.37	35.69	37.09	44.04
	Dataset 16	49.58	42.54	43.64	44.57
	Dataset 17	43.77	41.37	40.53	43.29
	Dataset 18	56.98	55.87	50.60	59.37
1-21, 23-29, 34-37	Dataset 1	70.26	59.43	49.86	74.39
	Dataset 2	67.51	46.69	40.73	69.84
	Dataset 3	66.80	45.38	37.41	67.05
	Dataset 4	63.02	42.95	36.03	65.73
	Dataset 5	60.29	41.24	33.92	64.24
	Dataset 6	65.37	44.50	38.10	68.59
	Dataset 7	51.07	46.88	45.63	49.32
	Dataset 8	40.93	39.04	32.44	35.47
	Dataset 9	76.46	73.54	70.33	77.12
	Dataset 10	48.75	36.91	40.12	44.68
	Dataset 11	58.31	53.33	54.45	50.35
	Dataset 12	39.26	30.81	34.45	42.16
	Dataset 13	50.10	42.84	43.52	52.09
	Dataset 14	62.57	61.97	53.57	62.65
	Dataset 15	34.19	30.44	25.08	33.39
	Dataset 16	44.31	40.65	33.15	45.80
	Dataset 17	44.46	39.99	32.81	44.09
	Dataset 18	51.70	50.73	51.12	59.80

Table 5.6: Accuracy and F-Score using SVM with Quadratic kernel

		Accuracy (%)			F-Score (%)
Selected features	Training dataset	Test	RIPE	BCNET	Test
1-37	Dataset 1	58.55	52.73	43.68	58.85
	Dataset 2	61.27	42.87	35.52	63.15
	Dataset 3	62.78	56.28	45.30	60.19
	Dataset 4	59.68	39.17	33.15	55.73
	Dataset 5	54.04	37.65	31.49	53.64
	Dataset 6	59.13	40.92	34.61	61.35
	Dataset 7	46.46	40.05	41.26	43.06
	Dataset 8	36.73	33.26	30.64	39.94
	Dataset 9	46.71	42.88	41.36	45.69
	Dataset 10	41.85	35.46	37.52	44.74
	Dataset 11	36.51	31.27	30.44	40.84
	Dataset 12	45.33	45.12	43.01	44.72
	Dataset 13	70.47	66.34	68.29	74.14
	Dataset 14	49.03	46.68	41.19	47.07
	Dataset 15	46.27	35.42	42.81	48.53
	Dataset 16	43.85	39.50	31.39	45.12
	Dataset 17	49.26	45.09	47.38	44.75
	Dataset 18	42.17	33.05	36.21	40.43
1-21, 23-29, 34-37	Dataset 1	63.84	58.51	46.39	67.24
	Dataset 2	63.36	46.55	38.73	64.68
	Dataset 3	62.53	43.30	37.12	63.09
	Dataset 4	57.40	40.59	34.78	60.33
	Dataset 5	55.58	37.13	30.53	58.29
	Dataset 6	60.21	41.85	35.17	62.48
	Dataset 7	47.57	38.16	36.34	46.24
	Dataset 8	46.80	42.54	41.26	47.23
	Dataset 9	74.50	68.12	70.34	77.43
	Dataset 10	40.73	38.13	33.32	45.65
	Dataset 11	57.28	52.47	54.83	55.12
	Dataset 12	33.30	29.55	25.42	30.34
	Dataset 13	60.45	56.71	55.30	61.97
	Dataset 14	57.13	55.34	50.18	52.42
	Dataset 15	50.52	47.67	46.96	48.49
	Dataset 16	46.70	38.91	34.16	36.20
	Dataset 17	49.52	40.38	38.42	48.52
	Dataset 18	48.64	43.15	47.71	53.84

Table 5.7: Accuracy and F-Score using SVM with Cubic kernel

		Accuracy (%)			F-Score (%)
Selected features	Training dataset	Test	RIPE	BCNET	Test
1-37	Dataset 1	65.33	54.31	45.53	68.55
	Dataset 2	63.15	46.23	40.17	65.49
	Dataset 3	68.83	57.57	46.47	70.03
	Dataset 4	59.50	41.44	35.88	62.15
	Dataset 5	50.37	35.47	30.17	55.28
	Dataset 6	59.28	38.04	33.20	58.33
	Dataset 7	52.28	42.84	41.83	45.48
	Dataset 8	49.94	45.57	44.46	46.50
	Dataset 9	42.89	41.94	36.62	40.82
	Dataset 10	54.35	50.13	53.84	55.29
	Dataset 11	50.88	44.72	49.80	54.74
	Dataset 12	53.05	51.80	52.45	49.88
	Dataset 13	74.12	70.07	73.41	76.68
	Dataset 14	60.46	58.51	55.13	63.29
	Dataset 15	52.23	48.14	50.56	55.42
	Dataset 16	51.98	49.26	47.07	53.20
	Dataset 17	63.52	60.94	61.24	65.23
	Dataset 18	50.19	45.34	47.89	51.58
1-21, 23-29, 34-37	Dataset 1	69.21	58.12	49.26	70.14
	Dataset 2	67.79	49.78	42.36	69.55
	Dataset 3	65.58	48.20	40.44	66.92
	Dataset 4	58.70	41.56	35.18	56.66
	Dataset 5	55.19	37.23	32.71	51.58
	Dataset 6	61.05	45.23	38.23	61.35
	Dataset 7	33.94	29.14	31.42	35.78
	Dataset 8	57.23	50.48	53.89	55.23
	Dataset 9	75.53	72.90	73.12	78.02
	Dataset 10	39.41	37.90	37.56	40.12
	Dataset 11	61.05	50.08	48.63	60.55
	Dataset 12	44.35	40.14	42.00	48.51
	Dataset 13	42.79	36.44	39.15	44.87
	Dataset 14	49.02	46.30	44.29	43.65
	Dataset 15	51.23	50.53	47.80	55.79
	Dataset 16	47.80	42.29	40.55	50.09
	Dataset 17	63.75	60.27	52.31	55.11
	Dataset 18	59.16	56.99	57.09	58.84

Table 5.8: Accuracy and F-Score using SVM with Gaussian kernel

		Accuracy (%)			F-Score (%)
Selected features	Training dataset	Test	RIPE	BCNET	Test
1-37	Dataset 1	70.11	60.36	51.76	70.42
	Dataset 2	68.28	49.23	40.85	69.19
	Dataset 3	72.82	63.39	54.12	71.48
	Dataset 4	64.49	46.12	37.49	64.29
	Dataset 5	58.30	37.31	35.11	60.42
	Dataset 6	61.25	40.28	36.78	63.04
	Dataset 7	55.13	50.56	52.98	56.50
	Dataset 8	53.17	49.55	47.80	57.13
	Dataset 9	43.12	40.27	39.07	43.59
	Dataset 10	39.41	33.17	38.91	40.74
	Dataset 11	53.09	51.89	52.62	57.10
	Dataset 12	36.39	32.40	35.17	37.89
	Dataset 13	75.84	71.57	73.82	76.67
	Dataset 14	43.40	41.47	42.98	46.21
	Dataset 15	48.88	45.54	47.03	50.53
	Dataset 16	42.55	36.76	40.96	44.24
	Dataset 17	35.12	31.47	34.31	37.60
	Dataset 18	51.75	46.96	50.63	56.17
1-21, 23-29, 34-37	Dataset 1	72.84	62.37	52.49	75.21
	Dataset 2	70.53	50.19	44.60	70.09
	Dataset 3	69.48	48.05	43.29	68.23
	Dataset 4	64.12	45.89	39.11	62.18
	Dataset 5	61.23	42.18	37.98	60.42
	Dataset 6	65.03	46.12	41.04	67.45
	Dataset 7	49.58	43.25	44.13	47.90
	Dataset 8	60.93	51.87	56.59	59.42
	Dataset 9	74.09	68.97	70.13	78.56
	Dataset 10	55.98	52.63	51.89	59.07
	Dataset 11	59.20	57.29	55.17	58.04
	Dataset 12	45.75	40.67	41.93	48.88
	Dataset 13	56.87	55.16	54.25	68.12
	Dataset 14	61.89	53.73	58.62	59.91
	Dataset 15	49.59	47.54	43.44	50.18
	Dataset 16	62.30	58.60	56.23	57.97
	Dataset 17	47.98	43.25	44.13	43.72
	Dataset 18	42.52	40.97	41.62	40.07

Table 5.9: Accuracy and F-Score using SVM with Sigmoid kernel

		Accuracy (%)			F-Score (%)
Selected features	Training dataset	Test	RIPE	BCNET	Test
1-37	Dataset 1	60.37	55.72	44.51	62.39
	Dataset 2	62.55	43.96	38.24	64.87
	Dataset 3	65.18	58.30	47.39	64.95
	Dataset 4	58.90	43.05	36.45	51.78
	Dataset 5	53.12	39.11	30.53	45.30
	Dataset 6	55.38	40.48	32.96	48.59
	Dataset 7	41.84	33.16	35.08	40.87
	Dataset 8	43.47	41.35	40.01	41.92
	Dataset 9	62.35	53.38	51.86	53.26
	Dataset 10	50.14	45.64	43.28	58.37
	Dataset 11	59.98	45.64	58.22	64.61
	Dataset 12	54.17	52.29	52.78	58.13
	Dataset 13	66.45	62.28	55.74	65.90
	Dataset 14	48.16	41.47	46.35	50.23
	Dataset 15	63.96	56.53	58.13	64.27
	Dataset 16	64.61	60.72	63.87	62.01
	Dataset 17	40.44	37.29	39.72	46.24
	Dataset 18	59.98	56.41	58.57	61.08
1-21, 23-29, 34-37	Dataset 1	66.12	59.24	48.43	68.34
	Dataset 2	65.49	47.93	41.88	67.19
	Dataset 3	63.53	46.89	38.19	66.30
	Dataset 4	58.42	40.71	32.37	60.25
	Dataset 5	55.71	36.34	30.42	55.37
	Dataset 6	60.11	41.35	35.90	64.48
	Dataset 7	60.98	55.51	56.43	62.15
	Dataset 8	48.40	45.19	46.38	50.45
	Dataset 9	69.48	65.23	60.19	71.06
	Dataset 10	58.80	50.52	52.85	51.06
	Dataset 11	49.14	47.41	44.46	52.89
	Dataset 12	43.09	40.02	41.17	47.85
	Dataset 13	58.60	57.12	55.71	64.18
	Dataset 14	38.22	33.24	36.13	43.60
	Dataset 15	42.46	35.02	40.25	45.39
	Dataset 16	51.93	49.14	41.74	52.89
	Dataset 17	48.81	47.82	46.80	49.85
	Dataset 18	64.62	60.53	59.49	62.11

5.3 Performance Comparisons

The best accuracy (85.35%) and F-Score (80.61%) are obtained using the linear SVM kernel when trained with Dataset 13. The results are obtained using the dataset with the most relevant features rather than considering all 37 features. For all the six datasets, the SVM with linear, polynomial, and Gaussian RBF kernels perform better than the SVM with quadratic, cubic, and sigmoid kernels. SVM with linear, polynomial, and Gaussian RBF kernels results in better accuracy and F-Score for all datasets. Quadratic and cubic kernels have better accuracy and F-Score when compared to the sigmoid kernel. Sigmoid kernel has the lowest accuracy and F-Score among the six kernels (one linear and five nonlinear). Linear kernel outperformed all the nonlinear kernels. Gaussian RBF

kernel had the second highest accuracy and F-Score when trained using with Dataset 13. It also outperformed the other four nonlinear kernels. Polynomial kernel had the second highest accuracy and F-Score when trained using Dataset 9 with the most relevant features. All kernels achieved better accuracy and F-Score with Dataset 9 and Dataset 13 when compared to other datasets. We also computed ROC curve for the kernels and reported their AUC. SVM with linear kernel achieved the highest AUC 0.96 while SVM with polynomial and Gaussian RBF kernels achieved 0.94 and 0.93, respectively. SVM with sigmoid kernel achieved the lowest AUC 0.70.

We provide here a short survey of related work where SVM was employed with a variety of kernels to classify images and detect gender, termites, diseases, and various network anomalies.

SVM with linear, quadratic, cubic, nearest neighbor, and RBF kernels were used for gender recognition by Moghaddam et al. [40]. 21-by-12 pixels low resolution thumbnail images of human faces were used to detect gender. SVM with RBF kernel achieved the minimum error rate of 4% as compared to other kernels used in the study. SVM using linear kernel had the maximum error rate of 35%.

Performance of several algorithm proposed by Deshpande et al. [25] has been evaluated using genuine Internet trace data. The study employed statistical patter recognition techniques for extracting meaningful features from the BGP update message data. The datasets considered were Nimda, Slammer, misconfiguration error, Code Red I, and Moscow Blackout affecting the Moscow Internet exchange (MSIX). The authors concluded that a very low false alarm rate is significant since the detection of an anomaly event may be used to trigger a mechanism that could alter the route exchange process of BGP routers.

The study by Deshpande et al. [26] presented a mechanism for capturing the statistical changes in features extracted from the BGP update messages data for online detection of routing instabilities on a single router. BGP features such as AS path length and AS path edit distance that reflect the actual topological changes rather than only volume anomalies were found to be useful in characterizing the behaviour of the topology under stress. The originating ASes for the root cause of the instabilities were also accurately identified based on the AS path changes.

An anomaly detection framework combined with a feature extraction mechanism was developed by Cazenave et al. [17]. The preprocessing modules extracts several high-level features from the BGP traffic. Various BGP events including worm attacks, power supply outages, submarine cable cuts, and misconfigurations such as the network prefix hijack and the routing table leaks were considered to test the framework for anomaly detection. The results demonstrated that trained with variety of abnormal BGP events, the system was able to detect a similar type of event that might occur in the future.

A comparison study of SVM with Gaussian RBF, polynomial, and linear kernels for disease detection was reported by Liu et al. [37] using Matlab to perform the experiments to detect diseases such as Parkinson, Ionosphere, Sonar, Wine, and Iris. Eight benchmark datasets from the University of California Irvine (UCI) repository were used. SVM with linear kernel achieved the best accuracy in all datasets. Gaussian RBF had the highest accuracy across all datasets. SVM with linear kernel

had the second best accuracy while SVM using polynomial kernel has the least accuracy among all eight datasets.

The performance of SVM were also evaluated with linear, cubic, anova, and sigmoid kernels using multi-spectral satellite images [32]. The dataset was collection of multi-spectral satellite images of agricultural lad in rural areas of India. Matlab was used to perform the experiments and evaluate SVM kernels based on accuracy and classification time. Linear kernel outperformed the other kernels by achieving accuracy of 92.36% with classification time of 141.68 seconds. Remaining kernels used in the study achieved an average of 67% accuracy with average classification time of 35 seconds.

Performance of BGP detection models based on naïve Bayes and decision tree J48 classifiers has been evaluated in the past by Cosovic et al. [24]. The Moscow Power Blackout, AS 9121 routing table leak, Panic Hijack, and AS path error datasets are the well-known anomalies that were tested when developing anomaly detection algorithms. The datasets were transformed employing feature discretization and feature selection using both filter and wrapper methods. The performance of classifiers was influenced by the employed datasets. Thirteen out of sixteen wrapper classifiers performed in the top half of all the other classifiers considered in the study.

A comparison study of SVM with linear, RBF, sigmoid, and polynomial kernels for termite detection in wood logs was reported by Nanda et al. [41]. 220 subterranean termites were inserted into a pine wood for feeding activity. The purpose of the study was to overcome the termite attacks and avoid a higher wood damage costs. SVM with polynomial kernel outperformed the other three kernels by achieving an accuracy of 91.99%.

Broad Learning System (BLS) and its extensions were evaluated by Liu et al. [35] based on accuracy and F-Score using the optimized Matlab function *mapminmax*(), which resulted in shorter training time. The models were trained and tested using Border Gateway Protocol (BGP) datasets that contain routing records collected from RIPE and BCNET as well as the NLS-KDD dataset containing network connection records. The used BGP datasets contain anomalous and regular data points. Multiple experiments were performed with various number of training and test data points that were selected from periods of anomalies. The best performance results were achieved using 60% of data for training and 40% for testing for each collected dataset.

Chapter 6

Conclusion and Future Work

The datasets used in this Thesis are collected from BCNET and RIPE. We analyzed variety of anomalous events such as Slammer, Nimda, Code Red I (version 2), Moscow power blackout affecting the Moscow Internet Exchange (MSIX), and AS 9121 routing table leak. In this Thesis, we have classified anomalies in the collected BGP traffic traces by conducting experiments using linear and nonlinear SVM kernels. Performance of employed classification algorithms and models depends on selection of features and combinations of training and test datasets. Linear, quadratic, cubic, polynomial, Gaussian RBF, and sigmoid kernels were used. We compared the accuracy and F-Score with and without feature selection using six different datasets. The SVM algorithm is one of the most efficient machine learning tools. SVM uses a set of mathematical functions called kernels that transform the input data into a high dimensional space before classifying the data points into distinct clusters. The performance of an SVM kernel depended on both the feature selection and the type of dataset. If the input dataset is linearly separable, then SVM with kernel performed better than other kernels.

The accuracy and F-score when using 37 features compared to features selected using decision tree for various combinations of training and test datasets differed since performance of models depends on the extracted features and the combination of selected features. Linear kernel outperformed all the nonlinear kernels. Linear kernel had the highest accuracy using both all the features and the most relevant features. Polynomial kernel had the second highest accuracy and F-Score using all 37 features when tested using Dataset 13. Gaussian RBF kernels had the second highest accuracy using the decision tree feature selection algorithm when tested on Dataset 9. Both polynomial and Gaussian RBF outperformed the other three nonlinear kernels. All kernels achieved better accuracy and F-Score with Dataset 9 and Dataset 13 when compared to the remaining sixteen datasets. Analyzed BGP anomaly datasets were linearly separable and, hence, the linear SVM kernel outperformed the nonlinear SVM kernels. Therefore, using SVM classifier with linear kernels may serve as a feasible approach for detecting BGP anomalies in communication networks.

In further studies, additional datasets such as data collected by the Center for Applied Internet Data Analysis (Caida), KDD Cup 1999 data, Routing Assets Database (RADb) as well as other types of blackouts and anomalies such as prefix hijack attacks may be used to evaluate the performance

of SVM with different kernels. In this Thesis, we combined two anomalies to generate training datasets and used the third anomaly as the test dataset. The models may also be evaluated if one anomaly was appropriately partitioned and used for both training and testing. A different approach to detect an anomaly may be to concatenate three anomalies to form a larger dataset and then partition it into the training and test datasets. For example, use 60% (80%) of data for training and 40% (20%) for testing and then compare the present results achieved by using the entire dataset. This method may help machine learning algorithms to identify additional characteristics of the datasets and relationships among data points.

Bibliography

[1] BCNET [Online]. Available: http://www.bc.net/. Accessed: Aug. 25, 2019.

[2] C5.0 [Online]. Available: http://www.rulequest.com/see5-info.html/. Accessed: Aug. 25, 2019.

[3] Code Red Worm [Online]. Available: https://www.caida.org/research/security/code-red/. Accessed: Aug. 25, 2019.

[4] Matlab 2019a [Online]. Available: https://www.mathworks.com/solutions/projects.html/. Accessed: Aug. 25, 2019.

[5] Nimda Worm [Online]. Available: http://www.cert.org/advisories/CA-2001-26.html/. Accessed: Aug. 25, 2019.

[6] RIPE NCC [Online]. Available: http://www.ripe.net/data-tools/. Accessed: Aug. 25, 2019.

[7] Slammer: Why security benefits from proof of concept code [Online]. Available: http://www.theregister.co.uk/2003/02/06/slammer why security benefits/. Accessed: Aug. 25, 2019.

[8] Zebra BGP parser [Online]. Available: http://www.linux.it/md/software/zebra-dump-parser.tgz/. Accessed: Aug. 25, 2019.

[9] T. Ahmed, B. Oreshkin, and M. Coates, "Machine learning approaches to network anomaly detection," in *Proc. USENIX Workshop Tackling Comput. Syst. Problems with Mach. Learn. Techn.*, Cambridge, MA, USA, Apr. 2007, pp. 1–6.

[10] N. Al-Rousan and Lj. Trajković, "Machine learning models for classification of BGP anomalies," in *Proc. IEEE Conf. on High Perform. Switching and Routing, HPSR 2012*, Belgrade, Serbia, June 2012, pp. 103–108.

[11] N. Al-Rousan, S. Haeri, and Lj. Trajković, "Feature selection for classification of BGP anomalies using Bayesian models," in *Proc. Int. Conf. Mach. Learn. Cybern., ICMLC 2012*, Xi'an, China, July 2012, pp. 140–147.

[12] R. Amami, D. B. Ayed, and N. Ellouze, "An empirical comparison of SVM and some supervised learning algorithms for vowel recognition," *J. Intell. Info. Processing*, vol. 3, no. 1, pp. 63–70, 2012.

[13] P. Batta, M. Singh, Z. Li, Q. Ding, and Lj. Trajković, "Evaluation of support vector machine kernels for detecting network anomalies," in *Proc. IEEE Int. Symp. Circuits and Syst., ISCAS 2018*, Florence, Italy, May 2018, pp. 1–4.

[14] B. L. Bhatnagar, S. Singh, C. Arora, and C. V. Jawahar, "Unsupervised learning of deep feature representation for clustering egocentric actions," in *Proc. Intl. Joint Conf. Artif. Intell., IJCAI 2017*, Melbourne, Australia, Aug. 2017, pp. 1447–1453.

[15] M. Bhuyan, D. Bhattacharyya, and J. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, Mar. 2014.

[16] C. M. Bishop, *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer-Verlag, 2006, pp. 325–358.

[17] I. O. Cazenave, E. Köşlũk, and M. C. Ganiz, "An anomaly detection framework for BGP," in *2011 Int. Symp. Innovations in Intell. Syst. and Appl.*, Istanbul, Turkey, June 2011, pp. 107–111.

[18] J. Chen and J. Ye, "Training SVM with indefinite kernels," in *Proc. Int. Conf. Mach. Learn., ICML 2008*, Helsinki, Finland, July 2008, pp. 136–143.

[19] C. Cortes and V. Vapnik, "Support-vector networks," *J. of Mach. Learn.*, vol. 20, no. 3, Sept. 1995, pp. 273–297.

[20] M. Cosović, S. Obradović, and Lj. Trajković, "Feature selection techniques for machine learning," in *Proc. Int. Scientific Conf., UNITECH 2013*, Gabrovo, Bulgaria, Nov. 2013, no. 1, pp. 85–89.

[21] M. Cosović, S. Obradović, and Lj. Trajković, "Algorithms for investigation of abnormal BGP events," in *Proc. Int. Scientific Conf., UNITECH 2013*, Gabrovo, Bulgaria, Nov. 2013, no. 2, pp. 253–257.

[22] M. Cosović, S. Obradović, and Lj. Trajković, "Using databases for BGP data analysis," in *Proc. UNITECH 2014*, Gabrovo, Bulgaria, Nov. 2014, vol. 2, pp. 367–370.

[23] M. Cosović, S. Obradović, and Lj. Trajković, "Performance evaluation of BGP anomaly classifiers," in *Proc. 3rd Int. Conf. Digital Info., Netw., and Wireless Comm. (DINWC 2015)*, Moscow, Russia, Feb. 2015, pp. 115–120.

[24] M. Cosović, S. Obradović, and Lj. Trajković, "Classifying anomalous events in BGP datasets," in *Proc. The 29th Annual IEEE Canadian Conf. Electrical and Comp. Eng. (CCECE 2016)*, Vancouver, Canada, May 2016, pp. 697–700.

[25] S. Deshpande, M. Thottan, T. K. Ho, and B. Sikdar, "A statistical approach to anomaly detection in interdomain routing," in *Proc. 3rd Int. Conf. Broadband Comm., Netw. and Syst.*, San Jose, USA, Oct. 2006, pp. 1–10.

[26] S. Deshpande, M. Thottan, T. K. Ho, and B. Sikdar, "An online mechanism for BGP instability detection and analysis," *IEEE Trans. Comput.*, vol. 58, no. 11, pp. 1470–1484, Nov. 2009.

[27] Q. Ding, Z. Li, P. Batta, and Lj. Trajković, "Detecting BGP anomalies using machine learning techniques," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Budapest, Hungary, Oct. 2016, pp. 3352–3355.

[28] Q. Ding, Z. Li, S. Haeri, and Lj. Trajković, "Application of machine learning techniques to detecting anomalies in communication networks: datasets and feature selection algorithms," in *Cyber Threat Intell.*, M. Conti, A. Dehghantanha, and T. Dargahi, Eds., Berlin: Springer, pp. 47-70, 2018.

[29] J. Grange, *Machine Learning for Absolute Beginners: A Simple, Concise & Complete Introduction to Supervised and Unsupervised Learning Algorithms*. CreateSpace Independent Publishing Platform, 2017, pp. 1–76.

[30] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, 2008.

[31] M. M. Hossain and M. S. Miah, "Evaluation of different SVM kernels for predicting customer churn," in *Proc. 18th Int. Conf. Comput. and Inform. Technol.*, Dhaka, Bangladesh, Dec. 2015, pp. 1–4.

[32] A. N. Khobragade, M. M. Raghuwanshi, and L. Malik, "Evaluating Kernel Effect on Performance of SVM Classification using Satellite Images," *J. Sci. Eng. Res. Intell. Res.*, vol. 7, no. 3, pp. 742–749, Mar. 2016.

[33] Y. Li, H. J. Xing, Q. Hua, X.-Z. Wang, P. Batta, S. Haeri, and Lj. Trajković, "Classification of BGP anomalies using decision trees and fuzzy rough sets," in *Proc. IEEE Trans. Syst., Man, Cybern.*, San Diego, CA, USA, Oct. 2014, pp. 1331–1336.

[34] Z. Li, Q. Ding, S. Haeri, and Lj. Trajković, "Application of machine learning techniques to detecting anomalies in communication networks: classification algorithms," in *Cyber Threat Intell.*, M. Conti, A. Dehghantanha, and T. Dargahi, Eds., Berlin: Springer, pp. 47-70, 2018.

[35] Z. Li, A. L. Gonzalez Rios, G. Xu, and Lj. Trajković, "Machine learning techniques for classifying network anomalies and intrusions," in *Proc. IEEE Int. Symp. Circuits and Syst.*, Sapporo, Japan, May 2019, pp. 1–5.

[36] H. T. Lin and C. J. Lin, "A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods," *Technical Report*, *J. of Neural Comput.*, vol. 3, no. 1, pp. 1–32, 2003.

[37] Z. Liu and H. Xu, "Kernel selection for support vector machine classification," *J. Algo. Compt. Tech.*, vol. 8, no. 2, pp. 163–177, June 2013.

[38] Y. Liu and K. K. Parhi, “Computing RBF kernel for SVM classification using stochastic logic,” in *Proc. IEEE Int. Workshop Signal Process. Syst., SiPS 2016*, Dallas, TX, USA, Oct. 2016, pp. 327–332.

[39] T. Manderson, “Multi-threaded routing toolkit (MRT) border gateway protocol (BGP) routing information export format with geo-location extensions,” [Online]. Available: http://www.ietf.org/rfc/rfc6397.txt/. Accessed: Aug. 25, 2019.

[40] B. Moghaddam and M. H. Yang, “Sex with support vector machines,” in *Neural Info. Processing Syst., NIPS* vol. 13. no. 1, pp. 1–7, 2000.

[41] M. A. Nanda, K. B. Seminar, and D. Nandika, “A comparison study of kernel functions in the support vector machine and its application for termite detection,” *J. Info. 2018*, vol. 9, no. 1, pp. 5–19, Jan. 2018.

[42] C. S. Ong, X. Mary, S. Canu, and A. J. Smola, “Learning with non-positive kernels,” in *Proc. Int. Conf. Mach. Learn., ICML 2004*, Banff, Alberta, Canada, July 2004, pp. 81–83.

[43] M. Papathomas, “On the correspondence from Bayesian log-linear modelling to logistic regression modelling with g-priors," in *Test*, J. Lopez-Fidalgo and M. D. Ugarte Eds., Berlin: Springer, vol. 27, no. 1, pp. 197–220, Apr. 2017.

[44] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, Aug. 2005.

[45] J. C. Platt, *Probabilistic Outputs for Support Vector Machines and Comparison to Regularized Likelihood Methods*, *MIT Press*, 1999, pp. 61–74.

[46] A. C. Popescu, B. J. Premore, and T. Underwood, “The anatomy of a leak: AS9121,” Renesys Corporation, Manchester, NH, USA, May 2005. [Online]. Available: http://50.31.151.73/meetings/nanog34/presentations/underwood.pdf/. Accessed: Aug. 25, 2019.

[47] F. J. Provost and G. M. Weiss, “Learning when training data are costly: the effect of class distribution on tree induction,” *J. Artif. Intell. Res.*, vol. 19, no. 1, pp. 315-354, July 2003.

[48] Y. Rekhter and T. Li, ”A Border Gateway Protocol 4 (BGP-4),” RFC 1771, *IETF*, Mar. 1995.

[49] V. D. Sánchez, “Advanced support vector machines and kernel methods,” *J. of Neurocomputing*, vol. 55, no. 1–2, pp. 5–20, Sept. 2003.

[50] B. Schölkopf, “The kernel trick for distances,” *Technical Report*, Microsoft Research, 2000.

[51] V. Vural and J. G. Dy, “A hierarchical method for multi-class support vector machines,” in *Proc. Int. Conf. Mach. Learn., ICML 2004*, Banff, Alberta, Canada, July 2004, pp. 831–838.

[52] T. Zhang, "On the dual formulation of regularized linear systems with convex risks," *J. of Mach. Learn.*, vol. 46, no. 1, Jan. 2002, pp. 91–129.

[53] J. Zhang, J. Rexford, and J. Feigenbaum, "Learning-based anomaly detection in BGP updates," in *Proc. Workshop on Mining Netw. Data*, Philadelphia, PA, USA, Aug. 2005, pp. 219–220.

[54] Y. Zhang, Z. M. Mao, and J. Wang, "A firewall for routers: protecting against routing misbehavior," in *Proc. 37th Annu. IEEE/IFIP Int. Conf. Dependable Syst. and Netw.*, Edinburgh, UK, June 2007, pp. 20–29.