BOĞAZİÇİ UNIVERSITY

CMPE 462

Project 2

İNCI MELIHA BAYTAŞ

BETAZERO

BARAN DENIZ KORKMAZ - 2015400183

DOĞUKAN KALKAN - 2015400132

SPRING 2020

Contents

1 Introduction

Support Vector Machines are supervised learning models that analyze data used for classification and regression analysis. Support Vector Machines have been recognized with their robustness for noise and strength in classification over decades. In modern era of advanced deep learning techniques, support vector machine still preserve its strength and benefits in machine learning applications.

One common task in machine learning is classification of data. SVMs provide the characteristics of robustness for noise by maximizing the margin for decision boundary that coins the term 'support vector'. More formally, a support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Today, there exists many hybrid applications that combines SVMs and Neural Networks from Deep Learning.

The second curricular project of Spring 20' term aims at encouraging assignees to combine their theoretical knowledge to practical skills about SVM by using **libsvm**. **Libsvm** is a popular open source machine learning library, developed at the National Taiwan University which provides a practical interface for building support vector machine applications. Below you may refer to the sections divided by subsequent tasks in the scope of project. Finally, you can read our conclusions about the project.

2 Task 1: Hard-Margin SVM

Linear SVMs are of two types: Hard-Margin and Soft-Margin. In the case of linearly separable data, we can select we can use two parallel hyperplanes that separate the two classes of data, so that the distance between these two parallel hyperplanes are maximized. The distance bounded by these two hyperplanes is called the "margin". Geometrically, to maximize the distance between the two hyperplanes, we want to minimize $||w||$.

Linear Support Vector Machines assume a linearly separable data that is correcly classified by an error margin of epsilon which implies toleration for error. We provide this assumption by adding a restriction into our formula for SVM that grants the correct classification of every data point by an error margin of epsilon. In Hard-Margin SVM, we enlarge our restriction in a way that we assume the correct classification of every data point without a tolerance for error leading zero-valued epsilon. Below, the primal formula for Hard-Margin SVM is given.

$$\min_{b,w} \quad w^T w \tag{1a}$$

$$\text{subject to} \quad y_i(w^T x_i + b) \geq 1, \forall i = 1, .., N \tag{1b}$$

Practically, using epsilon as zero requires a huge computational cost, since it requires extremely many number of iterations. We may predetermine epsilon as a sufficiently small number i.e. 10^{-5} . Therefore we must also choose a very large **C**, i.e.10^9, value that enforces a passive epsilon. These remarks lead us into applying the formula given below for SVM according to **libsvm** by setting the hyperparameters predetermined for the training of Hard-Margin SVM.

$$\min_{b,w} \quad w^T w + C \cdot \sum_{i=1}^{N} \epsilon_i \tag{2a}$$

$$\text{subject to} \quad y_i(w^T x_i + b) \geq 1 - \epsilon_i, \forall i = 1, .., N \tag{2b}$$

$$\epsilon_i \geq 0 \tag{2c}$$

Libsvm requires declaration of two class instances for the training of a support vector machine. In this section, you will find the information about the initialization required for Hard-Margin SVM:

1. **svm_parameter**: The declaration of conceptual parameters which determines the characteristics of Support Vector Machine. The parameters required for training of a hard-margin SVM are as follows:
   - **svm_type**: SVM Type, initialized to 0 which stands for Linear SVM.
   - **C**: Coefficient of ϵ_i
   - **eps**: Maximum threshold for epsilon. More formally, the error tolerance of SVM.

2. **svm_problem**: The declaration of training data and class labels.

After the initialization of class instances, we are ready to train our model and make predictions based on the returned model weights accordingly. Below, you can see the accuracy plots for Hard-Margin SVM that is trained in terms of predetermined hyperparameters. Please beware that in this section we have not introduced kernel tricks yet.

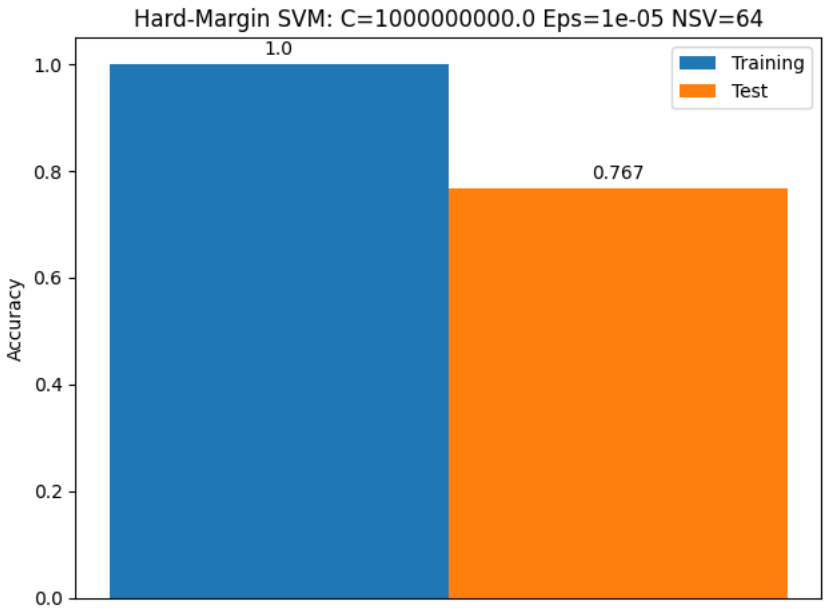

Figure 1: Hard-Margin SVM

3 Task 2: Soft-Margin SVM & Kernel Tricks

Libsvm requires declaration of two class instances for the training of a support vector machine. In this section, you will find the information about the initialization required for Soft-Margin SVM:

1. **svm_parameter**: The declaration of conceptual parameters which determines the characteristics of Support Vector Machine. The parameters required for training of a soft-margin SVM are as follows:
   - **svm_type**: SVM Type, initialized to 0 which stands for Linear SVM.
   - **C**: Coefficient of ϵ_i
   - **kernel_type**: Kernel Type, initialized to 2 which stands for radial basis function.
   - **gamma**: A parameter for kernel functions that have gamma term in them, set to 1 by default.
   - **coef0**: A parameter for kernel functions that have a constant term in them, set to 0 by default.
   - **eps**: Maximum threshold for epsilon. More formally, the error tolerance of SVM.
2. **svm_problem**: The declaration of training data and class labels.

As in the case of Hard-Margin SVM, our interest in the Soft-Margin SVM is to obtain two parallel hyperplanes that separates the two classes of data while maximizing the distance between these two hyperplanes. Maximizing the distance between the hyperplanes also means minimizing $||w||$, as stated in Task 1.

However, unlike Hard-Margin SVM, Soft-Margin SVM can be used to separate two data classes by tolerating some error. In other words, in Hard-Margin SVM, a strict assumption is made and the assumption is that all the data points in our data set are correctly separable. In the field of Machine Learning, this assumption is actually not realistic. As the name suggests, Soft-Margin SVM tolerates classification errors stem from the nature of the data set which does not allow utilization of Hard-Margin SVM.

Below, the minimization problem for Soft-Margin SVM is given.

$$\min_{b,w} \quad w^T w + C \cdot \sum_{i=1}^{N} \epsilon_i \tag{3a}$$

$$\text{subject to} \quad y_i(w^T x_i + b) \geq 1 - \epsilon_i, \forall i = 1, .., N \tag{3b}$$

$$\epsilon_i \geq 0. \tag{3c}$$

The term ϵ_i in (3b) captures by how much (x_i, y_i) fails to be separated. In the case of Hard-Margin SVM, this terms is zero since no failure in classification of the data is tolerated in Hard-Margin SVM. Also we can infer that as ϵ_i increases, the tolerance for incorrectly classified data points also increases.

It is the advantage of Soft-Margin SVM to tolerate some of the incorrectly classified data points. However, if these violations are too large, then the model will not be desired. In order to discourage large violations, we add a penalty term in the objective function:

$$\text{Penalty term} = C \cdot \textstyle\sum_{i=1}^{N} \epsilon_i.$$

The coefficient, C in the penalty term determines the SVM type. When C is large, any violation(i.e. incorrect classification) causes a large penalty value added to the objective function which we want to minimize. As a result, a large C gets us closer to Hard-Margin SVM.

Another useful property that Soft-Margin SVM provides for us is that we are able to utilize Kernel Trick. Both primal and dual SVM formulations for a certain task give the same optimal hyperplane. However, in certain situations, it is more efficient to solve the dual formulation, especially when the dimensionality is high. This dual view view of SVM enables us to use Kernel Trick. Kernel is basically an efficient way to transform the space in which our data points exist into a higher dimensional space. This is useful in many cases, because the data set that we work with might not be linearly separable and it is possible to find a higher dimensional space in which our data is linearly separable by using Kernel Trick.

Finally, we are ready to interpret the results. Below, you can see several Soft-Margin SVMs trained with different parameters and observations about them.

The observations and inferences on Soft-Margin SVMs with different Kernel functions will be discussed in two parts. We will explain the performances for different C values for a fixed Kernel function and the performances of different kernels for different Kernel functions.

Task 2 - Soft Margin SVM: Report for C = 0.1

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV
Linear	0	0	0	0.1	86.0	83.333	74
Polynomial	2	1	0	0.1	93.333	78.333	74
Polynomial	3	1	0	0.1	99.333	75.0	78
Polynomial	4	1	0	0.1	100.0	72.5	88
RBF	0	0.001	0	0.1	53.333	58.333	140
RBF	0	0.01	0	0.1	53.333	58.333	140
RBF	0	0.1	0	0.1	83.333	84.167	130
RBF	0	1	0	0.1	53.333	58.333	140
RBF	0	10	0	0.1	53.333	58.333	142
Sigmoid	0	0.001	0	0.1	53.333	58.333	140
Sigmoid	0	0.01	0	0.1	53.333	58.333	140
Sigmoid	0	0.1	0	0.1	82.0	84.167	118
Sigmoid	0	1	0	0.1	75.333	80.0	102
Sigmoid	0	10	0	0.1	70.667	75.0	100

Figure 2: Soft-Margin SVM with Different Kernels (C=0.1)

Task 2 - Soft Margin SVM: Report for C = 1

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV
Linear	0	0	0	0.1	86.667	84.167	60
Polynomial	2	1	0	0.1	97.333	77.5	67
Polynomial	3	1	0	0.1	100.0	75.833	76
Polynomial	4	1	0	0.1	100.0	72.5	88
RBF	0	0.001	0	0.1	53.333	58.333	140
RBF	0	0.01	0	0.1	84.0	84.167	100
RBF	0	0.1	0	0.1	86.667	84.167	82
RBF	0	1	0	0.1	98.0	79.167	138
RBF	0	10	0	0.1	100.0	60.833	150
Sigmoid	0	0.001	0	0.1	53.333	58.333	140
Sigmoid	0	0.01	0	0.1	82.667	84.167	116
Sigmoid	0	0.1	0	0.1	82.667	83.333	73
Sigmoid	0	1	0	0.1	65.333	65.833	60
Sigmoid	0	10	0	0.1	62.0	62.5	60

Figure 3: Soft-Margin SVM with Different Kernels (C=1)

Task 2 - Soft Margin SVM: Report for C = 10

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV
Linear	0	0	0	0.1	88.667	81.667	54
Polynomial	2	1	0	0.1	100.0	73.333	60
Polynomial	3	1	0	0.1	100.0	75.833	76
Polynomial	4	1	0	0.1	100.0	72.5	88
RBF	0	0.001	0	0.1	84.0	84.167	99
RBF	0	0.01	0	0.1	88.0	84.167	71
RBF	0	0.1	0	0.1	96.0	78.333	77
RBF	0	1	0	0.1	100.0	73.333	135
RBF	0	10	0	0.1	100.0	61.667	150
Sigmoid	0	0.001	0	0.1	82.667	83.333	116
Sigmoid	0	0.01	0	0.1	86.0	83.333	74
Sigmoid	0	0.1	0	0.1	77.333	74.167	47
Sigmoid	0	1	0	0.1	64.667	64.167	54
Sigmoid	0	10	0	0.1	61.333	61.667	58

Figure 4: Soft-Margin SVM with Different Kernels (C=10)

Task 2 - Soft Margin SVM: Report for C = 100

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV
Linear	0	0	0	0.1	89.333	81.667	50
Polynomial	2	1	0	0.1	100.0	69.167	61
Polynomial	3	1	0	0.1	100.0	75.833	76
Polynomial	4	1	0	0.1	100.0	72.5	88
RBF	0	0.001	0	0.1	86.0	84.167	69
RBF	0	0.01	0	0.1	91.333	83.333	61
RBF	0	0.1	0	0.1	99.333	78.333	71
RBF	0	1	0	0.1	100.0	73.333	135
RBF	0	10	0	0.1	100.0	61.667	150
Sigmoid	0	0.001	0	0.1	86.0	83.333	74
Sigmoid	0	0.01	0	0.1	85.333	84.167	60
Sigmoid	0	0.1	0	0.1	74.667	71.667	45
Sigmoid	0	1	0	0.1	64.0	64.167	54
Sigmoid	0	10	0	0.1	61.333	61.667	58

Figure 5: Soft-Margin SVM with Different Kernels (C=100)

Task 2 - Soft Margin SVM: Report for C = 1000

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV
Linear	0	0	0	0.1	90.0	81.667	50
Polynomial	2	1	0	0.1	100.0	69.167	61
Polynomial	3	1	0	0.1	100.0	75.833	76
Polynomial	4	1	0	0.1	100.0	72.5	88
RBF	0	0.001	0	0.1	89.333	85.0	61
RBF	0	0.01	0	0.1	96.667	76.667	59
RBF	0	0.1	0	0.1	100.0	75.833	66
RBF	0	1	0	0.1	100.0	73.333	135
RBF	0	10	0	0.1	100.0	61.667	150
Sigmoid	0	0.001	0	0.1	86.667	85.0	61
Sigmoid	0	0.01	0	0.1	88.0	81.667	50
Sigmoid	0	0.1	0	0.1	74.667	75.0	46
Sigmoid	0	1	0	0.1	64.0	64.167	54
Sigmoid	0	10	0	0.1	61.333	61.667	58

Figure 6: Soft-Margin SVM with Different Kernels (C=1000)

3.1 Different C values for a fixed Kernel

1. **Linear:** The linear kernel is the simplest kernel function. It is given by the following equation:

$$k(a, b) = a^T b + c$$

Models that use the linear kernel are often equivalent to models that do not use a kernel function. The reason is intuitive: The transformation function k transforms the data points into the same space with different values.

And if we look at the tables shown above, we can make a conclusion that as C increases, training accuracy increases without exception. The reason is that as C increases, we shift from Soft-Margin SVM to Hard-Margin SVM. We know from our previous observations that in Hard-Margin SVM, the training accuracy is always 100% given that the data points are correctly classifiable. And we also observe that as C increases, test accuracy decreases and the reason for the drop in test accuracy is that large values of C enforce the model to overfit the training data to ensure that the training accuracy is 100%.

2. **Polynomial:** The polynomial kernel function is a kernel function that transforms the data points into a higher dimensional space depending on the degree and is given by the following equation:

$$k(a, b) = (\gamma a^T b + c)^d$$

where γ is the slope, c is constant, and d is the degree.

We can also see that the linear kernel is actually a special case of the polynomial kernel with $\gamma = 1$ and $d = 1$.

Looking at the tables shown above, we can immediately say that the polynomial kernel functions work better than the linear kernel function in terms of training accuracy. Even when C=0.1, the model with the polynomial with d=4 has a training accuracy of 100% as can bee seen in Figure 2. And as C increases the training accuracy for the models with the polynomial kernel functions with lower degrees is getting closer to 100% as well. However, the same decline in test accuracy still exists for the case with d=2.

3. **Radial Basis Function:** The radial basis function kernel is given by the following equation:

$$k(a, b) = \exp(-\gamma ||a - b||^2)$$

where γ is the parameter that sets the spread of the kernel.

The RBF kernel transforms the data points into an infinite dimensional space. The parameter γ determines the importance of the distance between two data points. If a small γ value is used in the process, then the faraway points from the decision boundary(i.e. the hyperplane) influence it as much as the near points do. For the large values of γ, the decision boundary is affected heavily by the points near it. γ itself plays a role of regularization parameter and should be carefully selected. If overestimated, the exponential will behave almost linearly. And if underestimated, the function will lack regularization and the decision boundary will highly sensitive to the noise(i.e.faraway points) in the data set.

Now, we know that γ is a very sensitive parameter and its value has a great influence on the results. As we can interpret from the tables shown above, value of the parameter C affects the training and test results the same way it affects the other kernel functions discussed above meaning that as C increases, the training accuracy increases and the test accuracy decreases. For the parameter γ, it is harder to interpret and make a precise conclusion. However, for example, for C=0.1, the best γ value is 0.1 and the corresponding model gives a training accuracy of 83.333% and test accuracy of 84.617%, which is actually very good compared to the other models with different γ values. But as the parameter C increases, we can clearly state that higher γ values give better training results but also test results decline since the model starts to overfit the training data.

4. **Sigmoid(Hyperbolic Tangent):** The Sigmoid Kernel comes from the Neural Networks field, where the bipolar Sigmoid function is often used as an activation function for artificial neurons. It is also similar to the Sigmoid function in logistic regression. The equation of the Sigmoid Kernel is as follows:

$$k(a, b) = \tanh(\gamma a^T b + c)$$

where γ is the slope and c is constant.

The observable trend is that for small values of the parameter C, models with larger γ values give better results in training accuracy and apparently the Sigmoid Kernel does not allow any overfitting as it can be seen in Figure 2. On the other hand, for large values of the parameter C, models with small γ values give better results also. When we look at all the tables, we can observe that there is no huge differences between training accuracy and test accuracy meaning that the model does not overfit the data.

3.2 Different Kernel Functions for a Fixed C

After making inferences from models with different C values for a fixed Kernel Function, now we can draw our attention to making inferences from models with different Kernel Functions for a fixed C. Before starting, it is important to note that, choosing a C that is too large or too small does not make any sense and yields no interpretable results since large C values force the model to overfit the training data and small C values tolerate a high amount of errors that we cannot tolerate as Machine Learning students. Thus, the optimal C value for making observations is 1 since as it can be seen in Figure 3, the results are various and very interpretable.

The very first thing that draws our attention when we look at Figure 3 is that almost all models except ones with the Sigmoid Kernel overfit the training data. Evidently, C=1 is a large enough number for our models to overfit the training data.

The Linear Kernel is a simple kernel functions that calculates the inner product of two points and the training accuracy of 86.667 is a good enough result and so is the test accuracy of 84.167. There is a slight overfitting but it is negligible.

The Polynomial Kernel is a great indicator of how a single square operation can change the results. The Linear Kernel is a special case of the Polynomial Kernel with parameter d=1. Now, look at the model with the Polynomial Kernel with d=1. We observe a considerable amount of increase in the training accuracy and almost the same amount of decrease in the test accuracy. So, taking square of the function transforms the data points to a higher dimensional space, resulting in a better classification for the training data but relatively poor classification for the test data. And if the degree is incremented, as it can be seen in Figure 3, the training accuracy hits 100% and the test accuracy decreases as well.

In the case RBF Kernel, we can observe that relatively small values of γ does not yield a good result, as we can see that the training accuracy is 53.333% and the test accuracy is 58.333 for γ=0.001. As γ increases, we observe that each model overfits the training data more than the previous one, resulting in poor test accuracy results.

The Sigmoid Kernel is different from the other Kernels. It never overfits the training data and gives poor test accuracy results. And two γ values, 0.01 and 0.1 give fairly good accuracy results without overfitting.

4 Task 3: Support Vector Analysis

Task 3 - Kernel vs C Analysis: Report

Kernel Description	C = 0.1	C = 1	C = 10	C = 100	C = 1000
Linear	74	60	54	50	50
Polynomial: Degree=2 Coef0=0	74	67	60	61	61
Polynomial: Degree=3 Coef0=0	78	76	76	76	76
Polynomial: Degree=4 Coef0=0	88	88	88	88	88
RBF: Gamma=0.001	140	140	99	69	61
RBF: Gamma=0.01	140	100	71	61	59
RBF: Gamma=0.1	130	82	77	71	66
RBF: Gamma=1	140	138	135	135	135
RBF: Gamma=10	142	150	150	150	150
Sigmoid: Gamma=0.001 Coef0=0	140	140	116	74	61
Sigmoid: Gamma=0.01 Coef0=0	140	116	74	60	50
Sigmoid: Gamma=0.1 Coef0=0	118	73	47	45	46
Sigmoid: Gamma=1 Coef0=0	102	60	54	54	54
Sigmoid: Gamma=10 Coef0=0	100	60	58	58	58

Figure 7: Support Vector Analysis with different C Values

We begin this section by explaining the mechanism of Support Vector Machines. A Support Vector Machine classifies the data points in a data set by finding a decision boundary(i.e. a hyperplane) that separates the two data classes while maximizing the margin between two data classes. If our data set is an ideal data set which means the data points in it are completely separable, then the SVM must produce a hyperplane that separates the the data points into two non-overlapping data classes.

However, when we do not have such a data set meaning that the perfect separation is not possible, SVM finds the hyperplane that maximizes the margin between the two classes while minimizing the number of misclassifications and penalizing them.

The area between two hyperlanes that we reach by moving outwards from the decision boundary in two directions is called safe region. When performing a Hard-Margin SVM, we, intuitively, expect the area to be small. The reason is that the algorithm does not allow any data point from either of the two classes into the safe region. On the other hand, we expect the are to be larger in Soft-Margin SVM since now some data points from the two data classes are allowed to fall of the margin.

In construction of the decision boundary, support vectors play the key role

because we might think of support vectors as a definition for the decision boundary. Support vectors are the data points which are on the two hyperplanes between which the safe region stays.

As stated many times above, the parameter C plays an important role in the nature of the resulting SVM. When we start to increase the value of C, we also start to care more about violations(i.e. misclassifications), which gets us closer to Hard-Margin SVM.

Another parameter that characterizes the SVM is ϵ. ϵ simply captures by how much x_i, y_i fails to be separated. When we pick a small ϵ, the distance by which a point can deviate from the margin is also small which suggest that the corresponding model is close to Hard-Margin SVM. And if ϵ is zero, then the SVM is effectively a Hard-Margin SVM since now it does not allow any data point to fall off the margin.

If we look at the table in Figure 7, we can observe that the common behavior for almost all the models is that as C increases, the number of support vectors decreases. With all the facts stated above in mind, first of all, we can state that it is quite normal for the number of support vectors to decrease as we increase C since a large C results in an SVM that is closer to a Hard-Margin SVM which has a smaller safe region compared to Soft-Margin SVM. As we increase C, our model gets closer to Hard-Margin SVM and the data points that were misclassified previously and were also possibly support vectors on the wrong side of the decision boundary are now correctly classified and at the same time the safe region gets smaller, leaving all the data points on the correct side of the two hyperplanes.

5 Task 4: Decision Boundary Analysis

This section provides the observations about how decision boundary changes as we have removed a data point that is a support vector or not. The observations have been conducted based on subsequent training of Hard-Margin SVM by the predetermined C values and kernel configurations. After the analysis of previously trained models for different C values and kernel configurations has been completed, we have subsequently removed just a data point that is either a support vector or not in each iterations.

The practical analysis derived from the project output will serve presenting the illustration of theoretical aspects of support vectors. The removed data samples are randomly picked from the sets of support vector and non-support vector points belonging to the corresponding model. Randomness increases the likelihood of generalization, therefore utilized in most machine learning applications. We aim at presenting more scenarios by randomly removing data samples, providing a much generalized perspective into the SVM.

First, let us introduce the notion of support vectors in detail. As stated previously, SVMs aim at maximizing the margin that is the distance between the two parallel hyperplanes in both sides of decision boundary. This procedure can be interpreted as building a safe region between the two sides of decision boundary. Linear classification mechanism, then will be provided by the two hyperplanes determines the boundaries of positive and negative classes.

The key role of support vectors derive from the construction of safe region. The maximization of margin can also be visualized as the procedure we iteratively move outwards from the decision boundary until a threshold distance obtained determined by the hyperparameter ϵ, more explicitly, the smaller the epsilon the smaller the safe region. The safe region aims providing a tolerance for error. As denoted previously, hard-margin svm is the case where, theoretically, no error tolerance is accepted that is zero epsilon value. This way, hard-margin svm ensures that every data point is classified correctly. Let us present the restriction that guarantees this assumption in hard-margin svm.

$$y_i(w^T x_i + b) \geq 1 \tag{4}$$

In soft-margin svm, the parameter epsilon is not zero which yields a flexibility in classification of points. The larger the epsilon, the larger tolerance we construct. By the inequality below, we observe that, by increasing the epsilon we decrease the number of data samples that are guaranteed to classify correctly, thus a larger safe region.

$$y_i(w^T x_i + b) \geq 1 - \epsilon_i \tag{5}$$

The motivation behind the construction of safe region is to generalization of problem and avoidance of overfitting. The new data points may exhibit different characteristics by a small threshold compared to the training data,

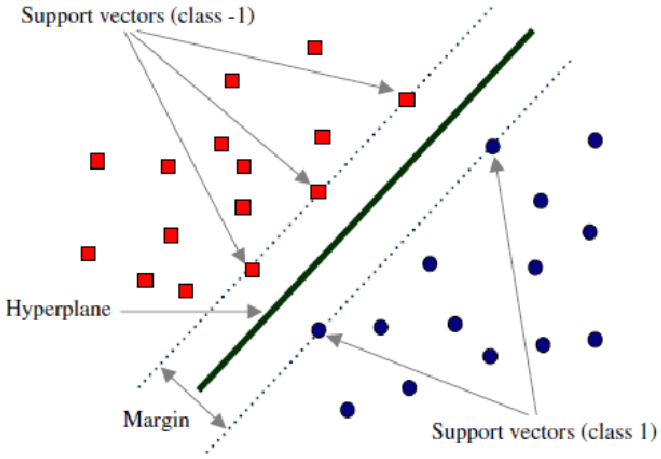

Figure 8: Support Vectors

therefore the construction of safe region will ensure that our model adapts well, and generalizes the problem.

Support vectors are data samples that are right on the parallel hyperplanes that construct the safe region. More formally, support vectors are the data samples that satisfies the inequalities given above in equality form. Implicitly, the expression tells that support vectors are the points on the final boundary that we guarantee the correct classification. As we move more inward, there is no guarantee that the points in the current region are correctly classified. We can interpret support vectors as the guardians of safe region. As stated previously, we begin to move outward starting from the hypothetical (imaginary) decision boundary, and stop when we hit into the support vector(s). In **Figure 8**, there is an illustration of the support vectors.

We can conclude that the support vectors play the key role in the construction of decision boundary. As explained above, support vectors are data samples that determines the decision boundary. The removal of the data samples that are not support vectors has no effect on the construction of decision boundary. In the project, we are asked to observe practical outputs we have obtained from our models by changing the data samples that are either a support vector or not, and to compare our practical observations into the theoretical expectations.

The further analysis will be given after the figures. First, let us share the output we obtain from the project. Below you can find the tables for different C values and the results for the cases when a support vector and non-support vector is removed for different kernel configurations.

Task 4 - Removal of Support Vectors: Report for C = 0.1

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV	Removed Sample
Linear	0	0	0	0.1	86.577	84.167	60	65
Polynomial	2	1	0	0.1	97.315	78.333	64	140
Polynomial	3	1	0	0.1	100.0	75.833	76	36
Polynomial	4	1	0	0.1	100.0	73.333	89	53
RBF	0	0.001	0	0.1	53.02	58.333	140	44
RBF	0	0.01	0	0.1	83.893	84.167	100	46
RBF	0	0.1	0	0.1	86.577	84.167	80	125
RBF	0	1	0	0.1	97.987	78.333	136	85
RBF	0	10	0	0.1	100.0	60.833	149	96
Sigmoid	0	0.001	0	0.1	53.02	58.333	140	112
Sigmoid	0	0.01	0	0.1	81.879	83.333	116	40
Sigmoid	0	0.1	0	0.1	83.221	83.333	76	75
Sigmoid	0	1	0	0.1	63.087	69.167	62	29
Sigmoid	0	10	0	0.1	62.416	61.667	58	38

(a) Removal of Support Vectors with Different Kernel Configurations

Task 4 - Removal of Non-Support Vectors: Report for C = 0.1

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV	Removed Sample
Linear	0	0	0	0.1	86.577	84.167	60	10
Polynomial	2	1	0	0.1	97.315	77.5	68	142
Polynomial	3	1	0	0.1	100.0	75.833	76	66
Polynomial	4	1	0	0.1	100.0	72.5	88	95
RBF	0	0.001	0	0.1	53.02	58.333	140	139
RBF	0	0.01	0	0.1	83.893	84.167	100	33
RBF	0	0.1	0	0.1	86.577	84.167	82	79
RBF	0	1	0	0.1	97.987	79.167	137	126
RBF	0	10	0	0.1	100.0	60.0	149	115
Sigmoid	0	0.001	0	0.1	53.02	58.333	140	119
Sigmoid	0	0.01	0	0.1	82.55	84.167	116	78
Sigmoid	0	0.1	0	0.1	82.55	83.333	73	123
Sigmoid	0	1	0	0.1	65.101	65.833	60	58
Sigmoid	0	10	0	0.1	61.745	62.5	60	146

(b) Removal of Non-Support Vectors with Different Kernel Configurations

Figure 9: C = 0.1

Task 4 - Removal of Support Vectors: Report for C = 1

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV	Removed Sample
Linear	0	0	0	0.1	87.919	82.5	61	98
Polynomial	2	1	0	0.1	97.315	77.5	65	2
Polynomial	3	1	0	0.1	100.0	75.0	75	107
Polynomial	4	1	0	0.1	100.0	73.333	87	19
RBF	0	0.001	0	0.1	53.02	58.333	140	144
RBF	0	0.01	0	0.1	83.893	84.167	100	43
RBF	0	0.1	0	0.1	87.919	81.667	82	6
RBF	0	1	0	0.1	97.987	79.167	137	56
RBF	0	10	0	0.1	100.0	60.0	149	100
Sigmoid	0	0.001	0	0.1	53.02	58.333	140	136
Sigmoid	0	0.01	0	0.1	83.221	83.333	114	92
Sigmoid	0	0.1	0	0.1	83.893	82.5	74	65
Sigmoid	0	1	0	0.1	65.101	65.833	60	141
Sigmoid	0	10	0	0.1	63.087	61.667	58	44

(a) Removal of Support Vectors with Different Kernel Configurations

Task 4 - Removal of Non-Support Vectors: Report for C = 1

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV	Removed Sample
Linear	0	0	0	0.1	86.577	84.167	60	104
Polynomial	2	1	0	0.1	97.315	77.5	67	103
Polynomial	3	1	0	0.1	100.0	75.833	76	76
Polynomial	4	1	0	0.1	100.0	72.5	88	52
RBF	0	0.001	0	0.1	53.02	58.333	140	139
RBF	0	0.01	0	0.1	83.893	84.167	100	36
RBF	0	0.1	0	0.1	86.577	84.167	82	26
RBF	0	1	0	0.1	97.987	79.167	138	115
RBF	0	10	0	0.1	100.0	60.833	150	
Sigmoid	0	0.001	0	0.1	53.02	58.333	140	100
Sigmoid	0	0.01	0	0.1	82.55	84.167	116	146
Sigmoid	0	0.1	0	0.1	82.55	83.333	73	33
Sigmoid	0	1	0	0.1	65.101	65.833	60	127
Sigmoid	0	10	0	0.1	61.745	62.5	60	101

(b) Removal of Non-Support Vectors with Different Kernel Configurations

Figure 10: C = 1

Task 4 - Removal of Support Vectors: Report for C = 10

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV	Removed Sample
Linear	0	0	0	0.1	87.919	84.167	60	113
Polynomial	2	1	0	0.1	97.315	76.667	65	44
Polynomial	3	1	0	0.1	100.0	73.333	75	97
Polynomial	4	1	0	0.1	100.0	73.333	87	19
RBF	0	0.001	0	0.1	53.691	58.333	138	57
RBF	0	0.01	0	0.1	84.564	83.333	98	48
RBF	0	0.1	0	0.1	87.248	81.667	83	99
RBF	0	1	0	0.1	97.987	79.167	137	7
RBF	0	10	0	0.1	100.0	60.0	149	142
Sigmoid	0	0.001	0	0.1	53.02	58.333	140	112
Sigmoid	0	0.01	0	0.1	82.55	83.333	116	6
Sigmoid	0	0.1	0	0.1	83.221	83.333	73	135
Sigmoid	0	1	0	0.1	63.087	67.5	62	65
Sigmoid	0	10	0	0.1	63.087	62.5	60	39

(a) Removal of Support Vectors with Different Kernel Configurations

Task 4 - Removal of Non-Support Vectors: Report for C = 10

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV	Removed Sample
Linear	0	0	0	0.1	86.577	84.167	60	127
Polynomial	2	1	0	0.1	97.315	77.5	67	117
Polynomial	3	1	0	0.1	100.0	75.833	75	138
Polynomial	4	1	0	0.1	100.0	72.5	88	120
RBF	0	0.001	0	0.1	53.691	58.333	138	9
RBF	0	0.01	0	0.1	83.893	84.167	100	148
RBF	0	0.1	0	0.1	86.577	84.167	82	84
RBF	0	1	0	0.1	97.987	79.167	138	79
RBF	0	10	0	0.1	100.0	60.833	150	
Sigmoid	0	0.001	0	0.1	53.691	58.333	138	127
Sigmoid	0	0.01	0	0.1	82.55	84.167	116	33
Sigmoid	0	0.1	0	0.1	82.55	83.333	73	35
Sigmoid	0	1	0	0.1	65.101	65.833	60	19
Sigmoid	0	10	0	0.1	61.745	62.5	60	71

(b) Removal of Non-Support Vectors with Different Kernel Configurations

Figure 11: C = 10

Task 4 - Removal of Support Vectors: Report for C = 100

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV	Removed Sample
Linear	0	0	0	0.1	87.919	83.333	59	24
Polynomial	2	1	0	0.1	97.315	77.5	64	82
Polynomial	3	1	0	0.1	100.0	75.833	76	106
Polynomial	4	1	0	0.1	100.0	74.167	88	62
RBF	0	0.001	0	0.1	53.02	58.333	140	77
RBF	0	0.01	0	0.1	83.893	84.167	100	5
RBF	0	0.1	0	0.1	86.577	84.167	81	7
RBF	0	1	0	0.1	97.987	80.0	136	72
RBF	0	10	0	0.1	100.0	60.0	149	140
Sigmoid	0	0.001	0	0.1	53.691	58.333	138	102
Sigmoid	0	0.01	0	0.1	81.879	84.167	114	138
Sigmoid	0	0.1	0	0.1	83.893	82.5	74	65
Sigmoid	0	1	0	0.1	65.101	68.333	62	149
Sigmoid	0	10	0	0.1	63.087	62.5	60	63

(a) Removal of Support Vectors with Different Kernel Configurations

Task 4 - Removal of Non-Support Vectors: Report for C = 100

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV	Removed Sample
Linear	0	0	0	0.1	86.577	84.167	60	101
Polynomial	2	1	0	0.1	97.315	77.5	67	88
Polynomial	3	1	0	0.1	100.0	75.833	76	18
Polynomial	4	1	0	0.1	100.0	72.5	88	116
RBF	0	0.001	0	0.1	53.02	58.333	140	13
RBF	0	0.01	0	0.1	83.893	84.167	100	7
RBF	0	0.1	0	0.1	86.577	84.167	82	128
RBF	0	1	0	0.1	97.987	79.167	138	93
RBF	0	10	0	0.1	100.0	60.833	150	
Sigmoid	0	0.001	0	0.1	53.691	58.333	138	34
Sigmoid	0	0.01	0	0.1	81.879	83.333	116	49
Sigmoid	0	0.1	0	0.1	82.55	83.333	73	130
Sigmoid	0	1	0	0.1	65.101	65.833	60	19
Sigmoid	0	10	0	0.1	61.745	62.5	60	7

(b) Removal of Non-Support Vectors with Different Kernel Configurations

Figure 12: C = 100

Task 4 - Removal of Support Vectors: Report for C = 1000

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV	Removed Sample
Linear	0	0	0	0.1	86.577	82.5	61	131
Polynomial	2	1	0	0.1	97.315	78.333	66	10
Polynomial	3	1	0	0.1	100.0	75.0	75	107
Polynomial	4	1	0	0.1	100.0	75.0	89	15
RBF	0	0.001	0	0.1	53.691	58.333	138	106
RBF	0	0.01	0	0.1	83.893	84.167	100	106
RBF	0	0.1	0	0.1	86.577	84.167	82	59
RBF	0	1	0	0.1	97.987	79.167	137	18
RBF	0	10	0	0.1	100.0	60.0	149	2
Sigmoid	0	0.001	0	0.1	53.02	58.333	140	24
Sigmoid	0	0.01	0	0.1	82.55	84.167	114	29
Sigmoid	0	0.1	0	0.1	83.221	84.167	75	107
Sigmoid	0	1	0	0.1	65.101	65.833	60	15
Sigmoid	0	10	0	0.1	63.758	64.167	58	68

(a) Removal of Support Vectors with Different Kernel Configurations

Task 4 - Removal of Non-Support Vectors: Report for C = 1000

Kernel Type	Degree	Gamma	Coef0	Eps	Tr Acc	Test Acc	nSV	Removed Sample
Linear	0	0	0	0.1	87.919	85.833	59	99
Polynomial	2	1	0	0.1	97.315	77.5	67	76
Polynomial	3	1	0	0.1	100.0	75.833	76	75
Polynomial	4	1	0	0.1	100.0	72.5	88	79
RBF	0	0.001	0	0.1	53.691	58.333	138	95
RBF	0	0.01	0	0.1	83.893	84.167	100	105
RBF	0	0.1	0	0.1	86.577	84.167	82	146
RBF	0	1	0	0.1	97.987	79.167	138	89
RBF	0	10	0	0.1	100.0	60.833	150	
Sigmoid	0	0.001	0	0.1	53.691	58.333	138	66
Sigmoid	0	0.01	0	0.1	82.55	84.167	116	58
Sigmoid	0	0.1	0	0.1	82.55	82.5	73	132
Sigmoid	0	1	0	0.1	66.443	66.667	58	84
Sigmoid	0	10	0	0.1	61.745	62.5	60	100

(b) Removal of Non-Support Vectors with Different Kernel Configurations

Figure 13: C = 1000

Let us start our observations by the effect of the parameter C on the changes in the number of support vectors. Given that smalller C values imply a much more flexible margin, we must perceive that the flexibility in soft-margin boundary will give rise into much flexible set of support vectors. For this reason, we must conduct our analysis in C values higher than 0.1 in order to make a valid comparison of theoretical expectations and practical outcomes.

As stated previously, removal of support vectors might yield changes in decision boundary. However, removal of non-support vectors should have no effect on the determination of decision boundary. Considering that our data set is smaller, we might easily obtain noisy outcomes.

Please refer to **Figure 3: Soft-Margin SVM with Different Kernels (C=1)**, for Polynomial kernel with degree 4, we see the number of support vectors are 88 with test accuracy of 72.5. Now let us turn our attention into **Figure 10: C = 1**. In (a), the removal of support vector in 19th row has resulted into a new model with the number of support vectors 87 and a test accuracy of 73.333. The removal of support vector has changed the decision boundary. For the same kernel configuration, in (b), after the removal of non-support vector sample in 52nd row, the number of support vectors remain constant and we can clearly observe that the test accuracy remains constant which is due to stationary decision boundary. Same conclusions may be derived for polynomial kernel with degree 3 in the same tables.

Let's look at another example. Please refer to **Figure 4: Soft-Margin SVM with Different Kernels (C=10)**, for Polynomial Kernel with degree 3, we see the number of support vectors are 76 with test accuracy of 75.833. Now let us turn our attention into **Figure 11: C = 10**. In (a), the number of support vectors decreases by one and test accuracy decreases after the removal of support vector in 97th row. The table leads us into the fact that the removal of support vectors caused a change in decision boundary, since we work on a small and probably noisy data. Therefore, the removal of support vector has caused into a new arrangement on SVM model, which has changed the decision boundary in a way that it generalizes the problem worse. In (b), after the removal of non-support vector in 138th row, the number of support vectors have decreased by one and test accuracy remains constant. We can interpret that the noise in data might cause unexpected behavior in the outcomes compared to the theoretical expectations.

Finally, looking at the tables for other kernel types, you will see varying number of support vectors and accuracy levels in a large scale. We can interpret that fact as the noise and lack of density in the data we work with.

However, it is possible to observe the experimental results that are compatible with the expected behaviors as stated above in detail.

6 Bonus Task: Hard-Margin SVM by QP Solver

The optimization problem constructed by the formulation of SVM is actually a Quadratic Program. For hard-margin svm, the primal formulation is given by:

$$\min_{b,w} \quad w^T w \tag{6a}$$

$$\text{subject to} \quad y_i(w^T x_i + b) \geq 1, \forall i = 1, .., N \tag{6b}$$

Please notice that the problem above is in the form given below which is a Quadratic Program.

$$\min_{u \in R^q} \quad \frac{1}{2} u^T Q u + p^T u \tag{7a}$$

$$\text{subject to} \quad Au \geq c, \tag{7b}$$

CVXOPT is a software package for convex optimization based on the Python programming language which provides an interface for solving Quadratic Programs for a given set of parameters. Please observe from the link given below that the problem formulation provided for the QP is the following:

$$\min \quad \frac{1}{2} x^T P x + q^T x \tag{8a}$$

$$\text{subject to} \quad Gx \leq h, \tag{8b}$$

$$Ax = b \tag{8c}$$

http://cvxopt.org/userguide/coneprog.html#quadratic-programming

Please observe that the **Optimization Problem 8** is exactly the same as that we have defined above in **Optimization Problem 7** except the direction of inequality constraint and the existence of equality constraint. We can easily adapt our problem into **CVXOPT** by ignoring A and b, and switching the direction of inequality constraint by multiplying both sides by '-1'.

Since we have given our problem definition, now it is time to explicitly present our parameters for QP solver to implement Hard-Margin SVM. The return value and parameters are defined below:

1. Unknown Parameter (Return Value)

   (a) **x**
   The vector x is the returned output from the QP solver that is defined as below:

$$\mathrm{x} = \begin{bmatrix} b \\ w \end{bmatrix}, \mathrm{x} \in \mathrm{R}^{d+1}$$

   The vector x is composed of the optimal values of b and model weights.

2. Parameters

   (a) **P**
   The matrix P $\in \mathrm{R}^{(d+1)*(d+1)}$ is in the following form in order to bring the objective function into the desired form:

$$\mathrm{P} = \begin{bmatrix} 0 & 0_d^T \\ 0_d & I_d \end{bmatrix}, \mathrm{P} \in \mathrm{R}^{(d+1)*(d+1)}$$

   (b) **q**
   The linear coefficient of the unknown x must be set into $\vec{0}$. Therefore the vector q is in the given form:

$$\mathrm{q} = 0_{d+1}, \mathrm{q} \in \mathrm{R}^{(d+1)}$$

   (c) **G**
   The matrix G is formed by our training data in the following form:

$$\mathrm{G} = \begin{bmatrix} -y_1 & -y_1 x_1^T \\ . & . \\ . & . \\ . & . \\ -y_N & -y_N x_N^T \end{bmatrix}, \mathrm{G} \in \mathrm{R}^{N*(d+1)}$$

   Recall that the reason why we carry out element-wise multiplication by -1 is to switch the direction of inequality constraint as defined in **CVXOPT**.

   (d) **h**
   The vector h is formed by the right hand side of the inequality constraint denoted in the **Optimization Problem 7** which is given above.

$$\mathrm{h} = \begin{bmatrix} -1 \\ . \\ . \\ . \\ -1 \end{bmatrix}, \mathrm{h} \in \mathrm{R}^{N}$$

Likewise we did for parameter **G**, we carry out element-wise multiplication by -1 to switch the direction of inequality constraint as defined in **CVXOPT**.

(e) The remaining parameters **A** and **b** are not defined explicitly which means NULL. This way, **CVXOPT** automatically ignores the equality constraint.

After the parameters are provided into the function **cvxopt.solvers.qp**, the QP solver returns the optimal solution to the problem. In the project, we are expected to obtain the optimal solution into the example given in the lecture slides. Let us move forward into our example and the solution provided by the **CVXOPT** QP Solver.

6.1 Toy Example

The problem data set is given by:

$$\mathrm{X} = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 2 & 0 \\ 3 & 0 \end{bmatrix}, \mathrm{y} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

Let us show the corresponding parameters **P,q,G,h** we form considering the detailed explanations above.

$$\mathrm{P} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathrm{q} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \mathrm{G} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 2 \\ -1 & -2 & 0 \\ -1 & -3 & 0 \end{bmatrix}, \mathrm{h} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

The QP solver prints the following output:

```
     pcost       dcost       gap    pres   dres
 0:  3.2653e-01  1.9592e+00  6e+00  2e+00  4e+00
 1:  1.5796e+00  8.5663e-01  7e-01  2e-16  1e-15
 2:  1.0195e+00  9.9227e-01  3e-02  4e-16  9e-16
 3:  1.0002e+00  9.9992e-01  3e-04  2e-16  2e-15
 4:  1.0000e+00  1.0000e+00  3e-06  2e-16  1e-15
 5:  1.0000e+00  1.0000e+00  3e-08  1e-16  8e-16
Optimal solution found.
[-1.00e+00]
[ 1.00e+00]
[-1.00e+00]
```

Figure 14: Toy Example

which returns the following vector as the optimal solution:

$$x^* = \begin{bmatrix} b^* \\ w_1^* \\ w_2^* \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

7 Conclusion

Support vector machines are powerful machine learning algorithms providing secure accuracy results. What makes them preferable is their robustness against the noise of data and ability to provide high accuracy in classification.

Team BetaZero put on their best efforts on presenting the Support Vector Machines extensively in theoretical and practical aspects. Our initial observations consists of the power of support vector machines in accuracy, and its complexity what makes it harder to understand compared to other classical core algorithms. Our ambition is to provide a comprehensive work that might aid individuals those are interested in machine learning applications. Instead of focusing on the sole tasks, we adapted the approach to extend our analysis for general use as possible. We would be very glad if we are on the right page.

8 References

The references utilized in the implementation and documentation of the project are listed below:

- https://en.wikipedia.org/wiki/Support_vector_machine
- http://cvxopt.org/userguide/coneprog.html#quadratic-programming