**S21CSE471: Statistical Methods in AI** **Spring 2021**

# Support Vector Machines

*Prepared by:* *2019121010, 2020201060, 2020201072*

In this note, we start with introduction to Support Vector Machines(SVM) and then formulate the problem mathematically. We define the optimization problem and solve it using Lagrangian Duality. We also see how to modify it to allow errors/outliers and the SVM associated with it is commonly called Soft Margin SVMs.

## 1 Introduction

**Definition 1.1.** *Support Vector Machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.*

SVMs can be used for two things: **classification and regression**.

- SVMs are used for classification
- Support Vector Regression(SVRs) are used for regression

We will only focus on SVMs used for classification problems. SVMs are trained to learn a linear model i.e. it learns a hyperplane that separates data into two classes. If our data has only two dimensions, the hyperlane learned would be a line which separates the data points into two classes. Note that an important assumption that we make is that the data is linearly separable.

Let's assume we have a training set as shown in Figure 1 where there are heights and weights of some people and there is a way to distinguish between men and women based on those two features.

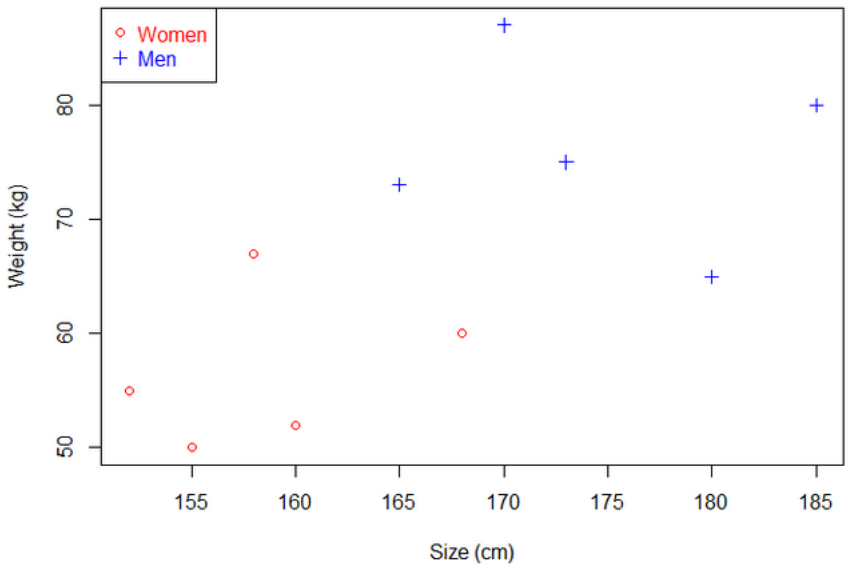



Figure 1: An example of training set

## 2 Hyperplane separating data points of two classes

By looking at Figure 1, we can think of many lines which would keep all data points represnting men are above the line and all data points representing women are below the line. One of such lines is drawn in Figure 2. Although, we see that there could be infinite lines which would classify the training data perfectly. Some of the possible hyperplanes are shown in Fig 3. The goal is to select a hyperplane which works well for test data and not just the training data.

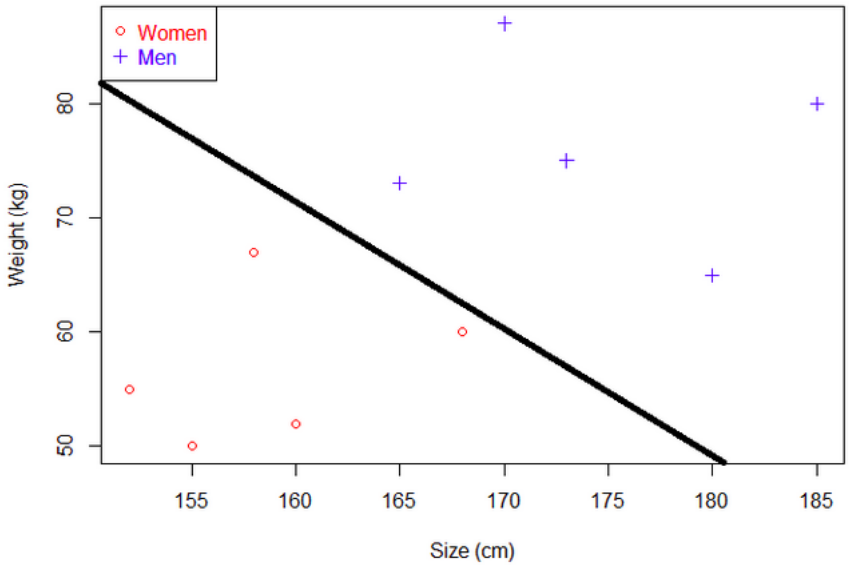



Figure 2: One of the hyperplanes that can separate data points of two classes

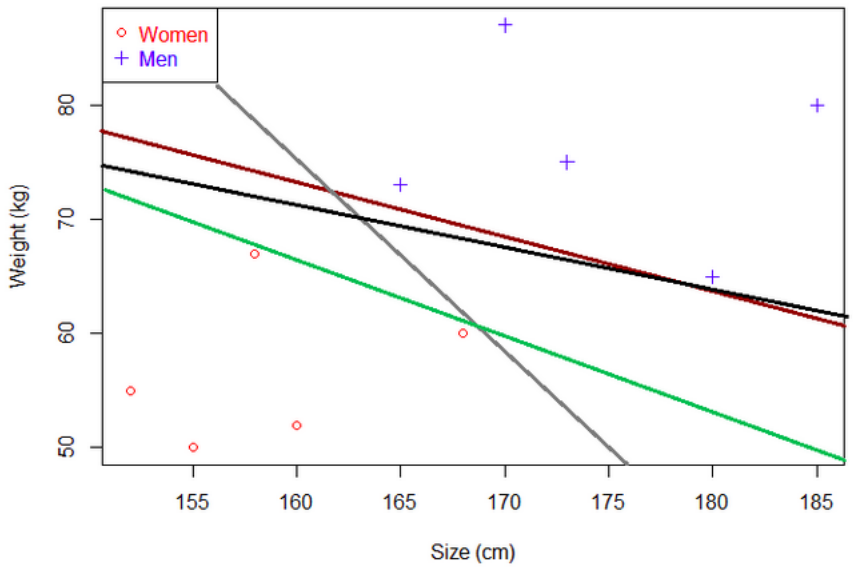



Figure 3: Multiple hyperplanes which classify training data perfectly

Test data would vary a bit from training data. Say, there is a test point that is near(similar height and weight) to some points that represent women, we would want the hyperplane to classify that point to represent a woman. In other words, we want a hyperplane which is as far as possible from training

data points of each category. For example, the green hyperplane in Figure 3 will classify 3 test points incorrectly as men as shown in 4 where we would have classified those points to represent women intuitively as they have nearly same weight and height as women in the training set. On the other hand, a hyperplane in Figure 5 would do a better job as it is as far away as possible from data points of both classes i.e. it has a maximum margin where margin is a distance of the hyperplane to the closest data points of either class.

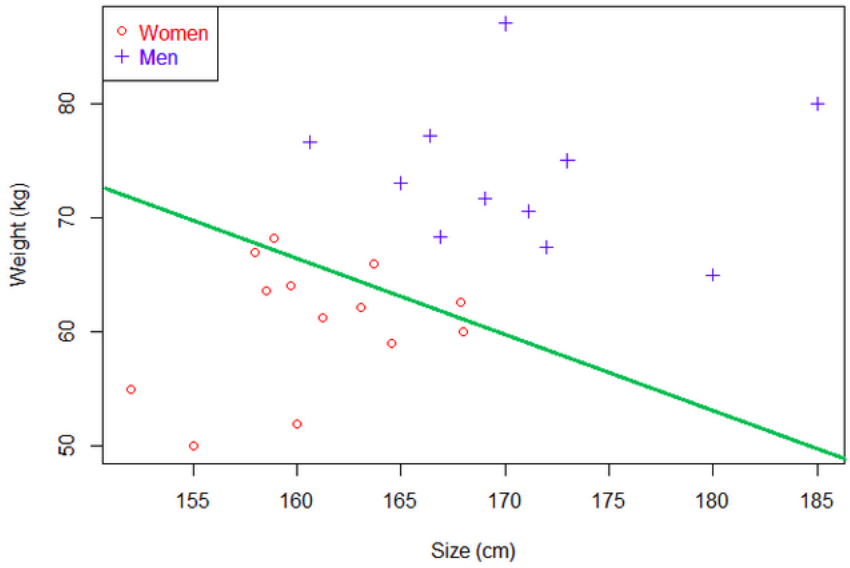



Figure 4: Incorrect classification with green hyperplane of Figure 3

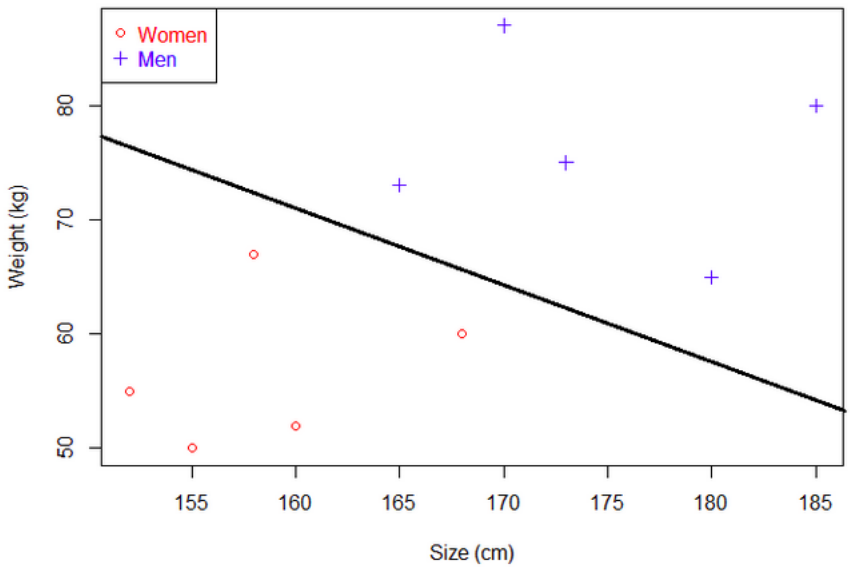



Figure 5: An optimal hyperplane

# 3 Margin of a hyperplane

Margin of a hyperplane is twice the distance between the hyperplane and the closest data point in the training set. There will be no points in the region inside the margin. Do note that this would cause problems when data is noisy(and it almost always is the case) and hence we will see soft margin SVM later to incorporate the noise in the model. Margin of a hyperplane is demonstrated in Figure 6 Two important observations to make are:

- Closer the hyperplane to a data point, smaller is the margin of the hyperplane
- Further the hyperplane from a data point, larger ther margin of the hyperplane

As we have already seen, we want the hyperplane to be as far as possible from the data points. Hence, the **optimal hyperplane** is the one with maximum margin and we have to find that hyperplane.

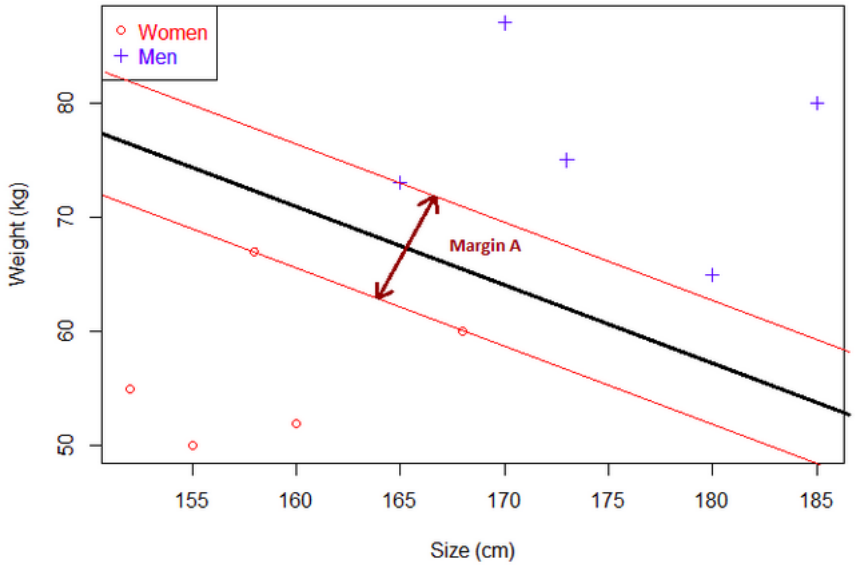



Figure 6: Margin of a hyperplane

## 3.1 Orthogonal projection of a vector

Given two vectors $\mathbf{x}$ and $\mathbf{y}$, we want to find orthogonal projection $\mathbf{z}$ of $\mathbf{x}$ onto $\mathbf{y}$.

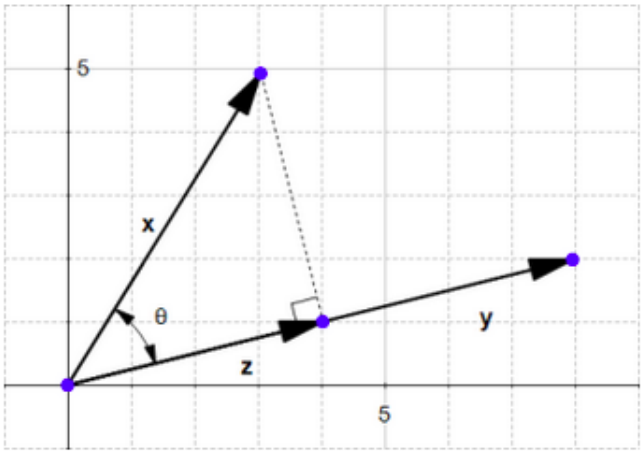



Figure 7: Orthogonal projection

$$cos\theta = \frac{||z||}{||x||} \tag{1}$$

$$||z|| = ||x|| cos\theta \tag{2}$$

But, we know $cos\theta$ from dot product to be as follows:

$$cos\theta = \frac{\mathbf{x}.\mathbf{y}}{||x||\ ||y||} \tag{3}$$

So, from equations (2) and (3), we get:

$$||z|| = ||x|| \frac{\mathbf{x}.\mathbf{y}}{||x||\ ||y||} \tag{4}$$

If we define a vector $\mathbf{u}$ as a unit vector in direction of $\mathbf{y}$, we get $||z||$ to be as follows:

$$\mathbf{u} = \frac{\mathbf{y}}{||y||} \tag{5}$$

$$||z|| = \mathbf{u}.\mathbf{x} \tag{6}$$

Since the direction of vector $\mathbf{z}$ is same as of vector $\mathbf{y}$, we can write

$$\mathbf{z} = ||z||u \tag{7}$$

## 3.2 Equation of Hyperplane

We know that equation of a line is $y = ax + b$ which is same as $y - ax - b = 0$. We can write the same in form of $\mathbf{w}^T\mathbf{x} = 0$ where

$$\mathbf{w} = \begin{pmatrix} -b \\ -a \\ 1 \end{pmatrix} \text{ and } \mathbf{x} = \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$$

Similarly, we can write any hyperplane in the form of $\mathbf{w}^T\mathbf{x} = 0$. Note that the vector $\mathbf{w}$ is always normal to the hyperplane. We are using this vector to define the hyperplane and hence it will be normal to the hyperplane by definition. When we define a hyperplane, we start by assuming that we have a vector that is normal to the hyperplane and here that vector is $\mathbf{w}$.

### 3.3 Computing distance of a point from a hyperplane

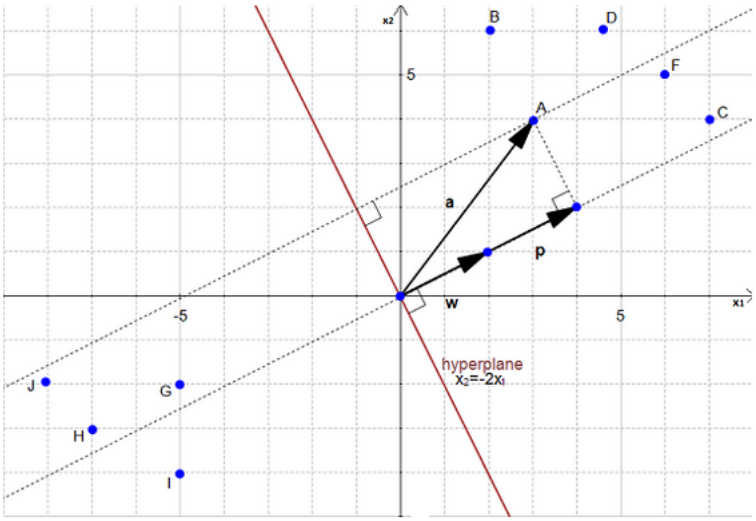



Figure 8: Distance of a point from hyperplane

Suppose, we have a hyperplane as shown in Figure 8 and we want to find the distance of point A from the hyperplane. Equation of the hyperplane is $x_2 = -2x_1$. Hence,

$$\mathbf{w} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \text{and} \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Let vector **u** be the unit vector in direction of vector **w**. So, $\mathbf{u} = (\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}})$. **p** is orthogonal projection of **a** onto **w**. We can find **p** by using $\mathbf{p} = (a.u)\mathbf{u}$. We can find $||p|| = \mathbf{a}.\mathbf{u} = \frac{10}{\sqrt{5}} = 2 * \sqrt{5}$.
The **margin** of the hyperplane is simply $2 * ||p|| = 4\sqrt{5}$

## 4 Finding the optimal hyperplane

We know that the optimal hyperplane has the maximum margin. We can find the biggest margin by finding two hyperplanes such that there are no data points between them and the distance between the two hyperplanes is maximum. The distance between the two hyperplanes itself is the margin we are looking for. The optimal hyperplane is at the middle point between the two selected hyperplanes.

**One notation to write the equation of a hyperplane is $\mathbf{w}.\mathbf{x} + b = 0$** . We can write $y - ax + b = 0$ as $\mathbf{w}.\mathbf{x} = 0$ follows:

$$\mathbf{w} = \begin{bmatrix} b \\ -a \\ 1 \end{bmatrix} \text{ and } \mathbf{x} = \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \tag{8}$$

But, we can also write it as $\mathbf{w}.\mathbf{x} + b = 0$ where:

$$\mathbf{w} = \begin{bmatrix} -a \\ 1 \end{bmatrix} \text{ and } \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{9}$$

Given a hyperplane $H_o$ which separates data and satisfies $\mathbf{w}.\mathbf{x} + b = 0$, we can select two hyperplanes $H_1$ and $H_2$ which have the following equations:

$$\mathbf{w}.\mathbf{x} + b = \delta \tag{10}$$
$$\mathbf{w}.\mathbf{x} + b = -\delta \tag{11}$$

such that $H_o$ is equidistant from both $H_1$ and $H_2$.

Here $\delta$ is not necessary as we can always normalize it by $\delta$. So, we rather keep it as the following two equations:

$$\mathbf{w}.\mathbf{x} + b = 1 \tag{12}$$
$$\mathbf{w}.\mathbf{x} + b = -1 \tag{13}$$

In addition to this, we also want these hyperplanes to satisfy the following conditions:

$$\mathbf{w}.\mathbf{x} + b >= 1 \text{ for } x_i \text{ with class } y_i = 1 \tag{14}$$
$$\mathbf{w}.\mathbf{x} + b <= -1 \text{ for } x_i \text{ with class } y_i = -1 \tag{15}$$

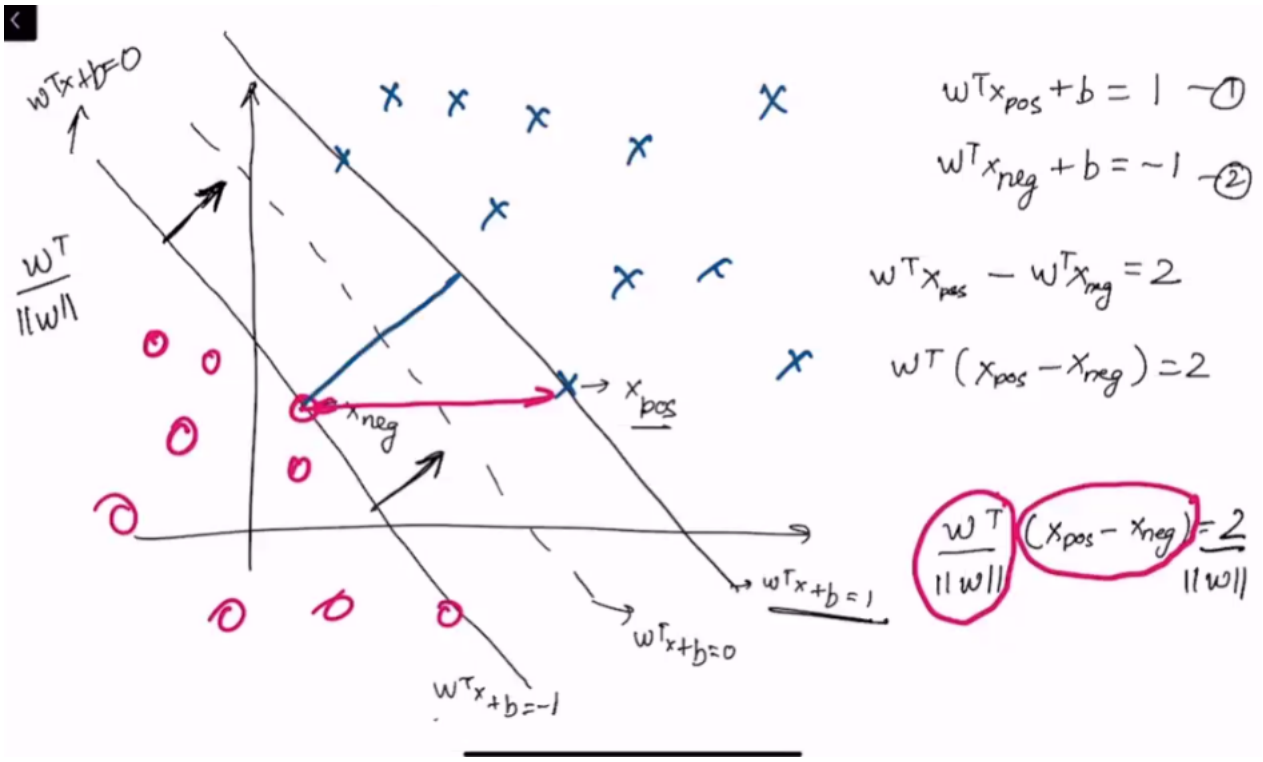



Figure 9: Distance of a point from hyperplane

We can combine these two conditions into a single equation as follows:

$$y_i(\mathbf{w}.\mathbf{x} + b) >= 1 \text{for all data points} \tag{16}$$

Say, we have a point $x_{pos}$ on $H_1$ and $x_{neg}$ on $H_2$. So, we have the following equations:

$$\mathbf{w}.\mathbf{x_{pos}} + b = 1 \tag{17}$$
$$\mathbf{w}.\mathbf{x_{neg}} + b = -1 \tag{18}$$
$$\mathbf{w}.(\mathbf{x_{pos}} - \mathbf{x_{neg}}) = 2 \tag{19}$$
$$\frac{\mathbf{w}}{||\mathbf{w}||}.(\mathbf{x_{pos}} - \mathbf{x_{neg}}) = \frac{2}{||\mathbf{w}||} \tag{20}$$

Vector $x_{pos} - x_{neg}$ is the vector shown in red in Figure 9. The of the hyperplane is the distance between two hyperplanes which is shown in blue. In other words, margin is the orthogonal projection of $\mathbf{x_{pos}} - \mathbf{x_{neg}}$ in the direction of $\mathbf{w}$. Remember that we saw $\mathbf{w}$ is orthogonal to the hyperplane itself and the margin is also the perpendicular distance between the two hyperplanes(margins). We know that the magnitute of the projection a vector $\mathbf{a}$ onto $\mathbf{w}$ is $\mathbf{a}.\mathbf{u}$ where $u = \frac{w}{||w||}$.

Left hand side of the equation (20) is the magnitude of the projection of $x_{pos} - x_{neg}$ in the direction of the margin(equivalently $w$). In other words, the left hand side of the equation is the magnitude of the margin itself. So, margin = $\frac{2}{||w||}$. To maximize the margin, we have to minimize $||w||$ given the constraint $y_i(\mathbf{w}.\mathbf{x} + b) >= 1$ for all data points

# 5 Lagrangian form and Duality

We can formulate the problem to solve the following problem in lagrangian form with the constraint that $\alpha_i >= 0 for all i$:

$$J(\mathbf{w}, b, \mathbf{ff}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i(\mathbf{w}^T\mathbf{x_i} + b) + \sum_{i=1}^{N} \alpha_i \tag{21}$$

We know that at the optimum, $\frac{\partial J}{\partial \mathbf{w}} = 0$ and $\frac{\partial J}{\partial b} = 0$. Solving for both of these, we get

$$\mathbf{w_0} = \sum_{i=1}^{N} \alpha_i * y_i * \mathbf{x_i} \tag{22}$$
$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{23}$$
$$\alpha_i(y_i(\mathbf{w_0}^T\mathbf{x_i} + b_0) - 1) = 0 \tag{24}$$

On converting to dual form, we get

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j} \tag{25}$$

This can be solved efficiently using linear solvers.

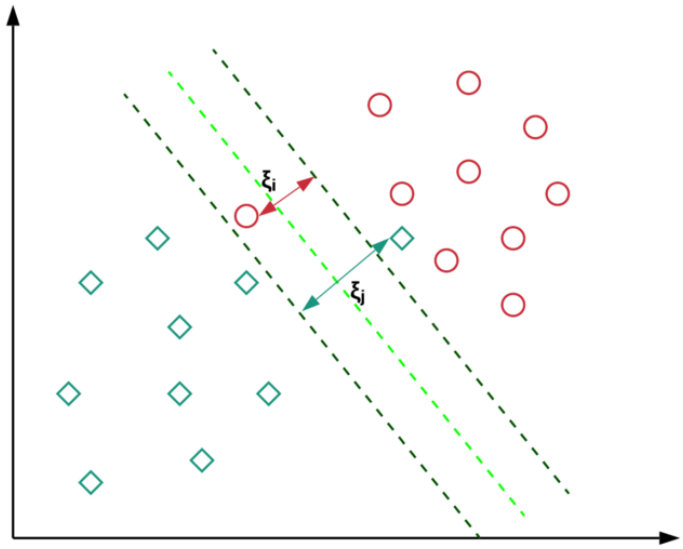
# 6 Small Margin SVM

**Introduction:** This idea is based on a simple premise: allow SVM to make a certain number of mistakes and keep margin as wide as possible so that other points can still be classified correctly. This can be done simply by modifying the objective of SVM.

**Motivation:**

1. As mentioned earlier, almost all real-world applications have data that is linearly inseparable.

2. In rare cases where the data is linearly separable, we might not want to choose a decision boundary that perfectly separates the data to avoid overfitting.
For example, consider the following diagram:




Here the red decision boundary perfectly separates all the training points. However, is it really a good idea of having a decision boundary with such less margin? Do you think such kind of decision boundary will generalize well on unseen data? The answer is: No. The green decision boundary has a wider margin that would allow it to generalize well on unseen data. In that sense, soft margin formulation would also help in avoiding the overfitting problem.

**Mathematics:**

Let us see how we can modify our objective to achieve the desired behavior. In this new setting, we would aim to minimize the following objective:

$$L = \frac{1}{2}\|w\|^2 + C(\#\ of\ mistakes)$$

This differs from the original objective in the second term. Here, C is a hyperparameter that decides the trade-off between maximizing the margin and minimizing the mistakes. When C is small, classification mistakes are given less importance and focus is more on maximizing the margin, whereas when C is large, the focus is more on avoiding misclassification at the expense of keeping the margin small.

The idea is: for every data point xi, we introduce a slack variable Ei. The value of Ei is the distance of xi from the corresponding class's margin if xi is on the wrong side of the margin, otherwise zero. Thus the points that are far away from the margin on the wrong side would get more penalty.

Given these constraints, our objective is to minimize the following function:
where we have used the concepts of Lagrange Multiplier for optimizing loss function under constraints. Let us compare this with SVM's objective that handles the linearly separable cases (as

$$L = \frac{1}{2}\|\vec{w}\|^2 + C\sum_i \xi_i + \sum_i \lambda_i(y_i(\vec{w}\ .\ \vec{x}_i + b) - 1 + \xi_i)$$

given below).

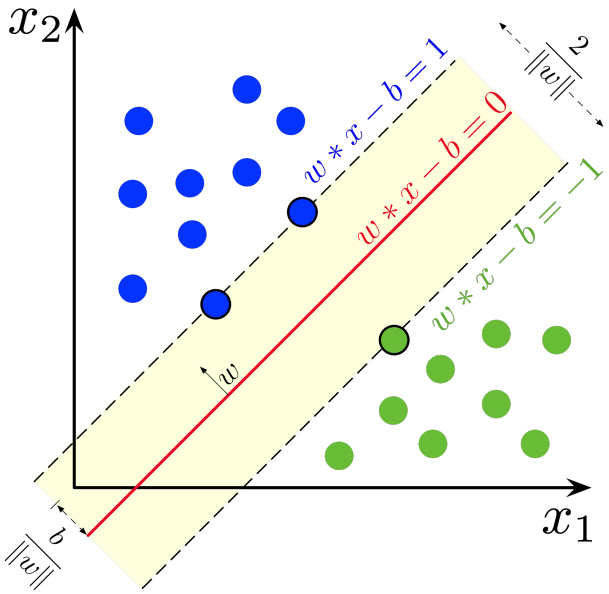
$$L = \frac{1}{2}\|\vec{w}\|^2 + \sum_i \lambda_i(y_i(\vec{w}.\vec{x}_i + b) - 1)$$

We see that only Ei terms are extra in the modified objective and everything else is the same.

In the final solution, Lambda of i corresponding to points that are closest to the margin and on the wrong side of the margin (i.e. having non-zero Ei) would be non-zero as they play a key role in positioning of the decision boundary, essentially making them the support vectors.

## 7 Multiclass classification using SVM

The objective is to find a hyperplane in an n-dimensional space that separates the data points to their potential classes. The hyperplane should be positioned with the maximum distance to the data points. The data points with the minimum distance to the hyperplane are called Support Vectors. Due to their close position, their influence on the exact position of the hyperplane is bigger than of other data points. In the graphic below the Support Vectors are the 3 points (2 blue, 1 green) laying on the lines.




The computations of data points separation depend on a kernel function. There are different kernel functions: Linear, Polynomial, Gaussian, Radial Basis Function (RBF), and Sigmoid. Simply put, these functions determine the smoothness and efficiency of class separation, and playing around with their hyperparameters may lead to overfitting or underfitting.

In its most simple type, SVM doesn't support multiclass classification natively. It supports binary classification and separating data points into two classes. For multiclass classification, the same principle is utilized after breaking down the multiclassification problem into multiple binary classification problems.

The idea is to map data points to high dimensional space to gain mutual linear separation between every two classes. This is called a One-to-One approach, which breaks down the multiclass problem into multiple binary classification problems. A binary classifier per each pair of

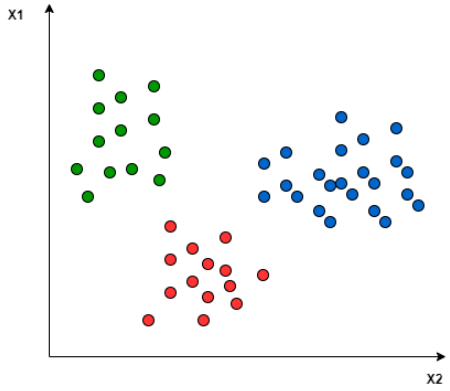
classes.

Another approach one can use is One-to-Rest. In that approach, the breakdown is set to a binary classifier per each class.

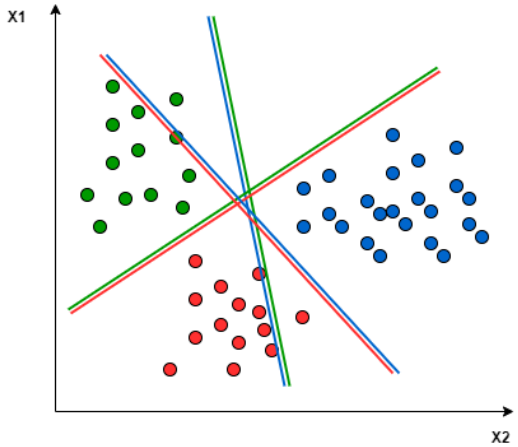
A single SVM does binary classification and can differentiate between two classes. So that, according to the two breakdown approaches, to classify data points from m classes data set:

In the One-to-Rest approach, the classifier can use m SVMs. Each SVM would predict membership in one of the m classes. In the One-to-One approach, the classifier can use $\frac{m(m-1)}{2}$ SVMs.

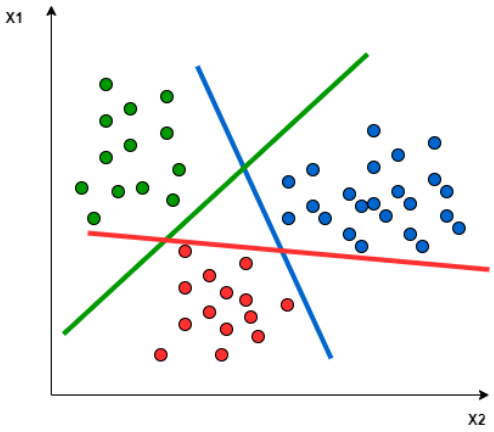
Let's take an example of 3 classes classification problem; green, red, and blue, as the following image:




In the One-to-One approach, we need a hyperplane to separate between every two classes, neglecting the points of the third class. This means the separation takes into account only the points of the two classes in the current split. For example, the red-blue line tries to maximize the separation only between blue and red points. It has nothing to do with green points:

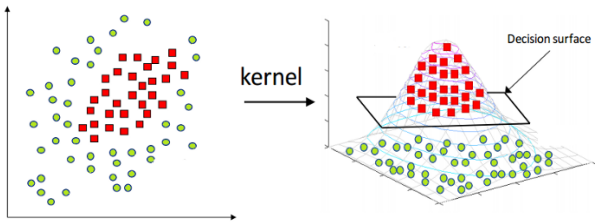



In the One-to-Rest approach, we need a hyperplane to separate between a class and all others at once. This means the separation takes all points into account, dividing them into two groups; a group for the class points and a group for all other points. For example, the green line tries to maximize the separation between green points and all other points at once.




## 8 Kernel Trick Intuition

Simply put kernel trick is used in SVM for bridging the gap between linearity and non-linearity. Consider a 2D pattern of two concentric circles.Inner circle is class 1 and outer circle from class 2. How to approach this problem?. The idea is that with an appropriate feature transformation the problem becomes simpler and the classifier/algorithm can do superior.




As you can see in the above picture, if we find a way to map the data from 2-dimensional space to 3-dimensional space, we will be able to find a decision surface that clearly divides between different classes. My first thought of this data transformation process is to map all the data point to a higher dimension (in this case, 3 dimension), find the boundary, and make the classification.

Consider a feature transformation (or feature map) as Let us start with a specific example: Let $P = [p1, p2]^T$ and $\theta(P) = [{p_1}^2, {p_2}^2, \sqrt{2}p_1p_2]$, You may wonder how this helps us or why we do this.We will see the utility as we move forward. Note the notations. Here the sample $P$ has two features/dimensions $p_1$ and $p_2$. A number of algorithms in machine learning use dot product as the basic operation. You already had seen $W^TX$ . The beauty of dot product is that the result is scalar independent of the dimensionality of the vector. i.e., the dot product of two vectors in 2D and 3D will be still a scalar, independent of the dimension.

Let us now compute the dot product of $\theta(P)$ and $\theta(Q)$, Here $Q$ is also a 2D vector similar to $P$. We know that $P^TQ = p_1q_1 + p_2q_2$. Let us compute the dot/scalar/inner product in the new feature space.

$$\theta(P)^T\theta(Q) = [{p_1}^2, {p_2}^2, \sqrt{2}p_1p_2]^T[{q_1}^2, {q_2}^2, \sqrt{2}q_1q_2] \tag{26}$$

$$\theta(P)^T\theta(Q) = {p_1}^2{q_1}^2 + {p_2}^2{q_2}^2 + 2p_1p_2q_1q_2 \tag{27}$$

$$\theta(P)^T\theta(Q) = (p_1q_1 + p_2q_2)^2 \tag{28}$$

$$\theta(P)^T\theta(Q) = (P^TQ)^2 \tag{29}$$

$$\theta(P)^T\theta(Q) = \kappa(P, Q) \tag{30}$$

What it says is something simple.If we want to compute the dot products of $\theta(P)^T$ and $\theta(Q)$,what we need to do is only computing dot product of $P$ and $Q$ and then square it. (indeed, that is true only for this specific $\theta$) Note that if we compute $\theta(P)^T\theta(Q)$ like this, we do not have to really compute $\theta$ explicitly. That could be a huge advantage in many situations. Later today, we also would like to map $P$ into infinite dimension with a $\theta$.The advantage of not requiring to compute $\theta$ will be a big advantage then. The function $\kappa()$ is a kernel function. We saw the kernel function of $(P^TQ)^2$. This need not be square. It could be $(P^TQ)^d$. This is a polynomial kernel. With some effort we can write the $\theta$ corresponding to this kernel. There are many other potential kernels. A kernel functions allows us to compute an inner/dot product in a new feature space.

# 9 Revised Formulation of SVM

The notion of Kernels is very nicely related to SVMs.One may wonder whether SVMs are designed for Kernels or Kernels are designed for SVMs!!. Let us now come back to our SVM problem (dual)

$$maximize J(\alpha) = \sum_{i=1}^{N}[\alpha_i] - 1/2\sum_{i=1}^{N}\sum_{j=1}^{N}[\alpha_i\alpha_j y_i y_j {X_i}^T X_j] \tag{31}$$

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \tag{32}$$

$$\alpha_i \geq 0 \forall i \tag{33}$$

Assume the input data $(x_i, y_i)$ was mapped by $\theta()$,leading to $\theta(x_i, y_i)$. We can find a linear boundary in the new feature space. And the corresponding decision boundary in the original space is going to be a nonlinear one. This makes the SVM problem (with appropriate constraints) as :-

$$max J(\alpha) = \sum_{i=1}^{N}\alpha_i - 1/2\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \theta(X_i)^T\theta(X_j) \tag{34}$$

$$max J(\alpha) = \sum_{i=1}^{N}\alpha_i - 1/2\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \theta(X_i^T X_j) \tag{35}$$

$$max J(\alpha) = \sum_{i=1}^{N}\alpha_i - 1/2\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \kappa(X_i, X_j) \tag{36}$$

Many practical solvers use the precomputed $N \times N$ kernel matrix. This avoids repeated computation of kernel functions. But this necessitates the need to store and manipulate a $N \times N$ matrix while solving. Not very nice for big data sets.

When you solve this nonlinear SVM problem, we get $\alpha_i$ corresponding to the new nonlinear SVM. Or what we get is the nonlinear SVM in the original feature space. Typically the output of the training is a set of $\alpha_i$ (that are non zero). At the test time, we only need to evaluate the sign of $W^T\theta(X) + b$

The kernel trick sounds like a "perfect" plan. However, one critical thing to keep in mind is that when we map data to a higher dimension, there are chances that we may overfit the model. Thus choosing the right kernel function (including the right parameters) and regularization are of great importance.

## 10 Popular Kernel Functions

- Linear Kernel : $\kappa(x, y) = x.y + c$
- Polynomial Kernel : $\kappa(x, y) = (x.y + c)^d$
- Sigmoid Kernel : $\kappa(x, y) = tanh(\alpha x.y + c)$
- Gaussian Kernel : $\kappa(x, y) = exp(-\gamma \times \|x - y\|^2)$

## References

[1] https://www.svm-tutorial.com/2017/02/svms-overview-support-vector-machines/

[2] https://medium.com/@sathvikchiramana/svm-dual-formulation-7535caa84f17

[3] https://towardsdatascience.com/multiclass-classification-with-support-vector-machines-svm-kernel-trick-kernel-functions-f9d5377d6f02

[4] https://towardsdatascience.com/support-vector-machines-soft-margin-formulation-and-kernel-trick-4c9729dc8efe

[5] https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d

[6] http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/