Manuel Ortiz Cachá (k20114995)

7CCSMPNN Pattern Recognition, Neural Networks and Deep Learning

Coursework Week 8: Support Vector Machines (SVMs)

Due: Thursday 8th April 2021 (23:59 UK time)

This coursework is assessed. A type-written report needs to be submitted online through KEATS by the deadline specified on the module's KEATS webpage. In this coursework, we consider a classification problem of 3 classes. A multi-class SVM-based classifier formed by multiple SVMs is designed to deal with the classification problem.

INTRODUCTION: Support Vector Machines (SVMs)

A Support Vector Machine (SVM) can help you solve a regression or a classification task. In this specific coursework, we will focus in a multi-label classification task. As a general idea, they are class separators that maximise the margin (space between the data classes), reducing the misclassification rate. They allow errors to occur in a controlled way, through the parameter ξ, the slack variable. If you wanted to solve a multiclass problem, an ensemble of SVMs could be used (later in the coursework).

The main benefit of these kind of Machine Learning models is that they are capable of creating a non-linear boundary between data classes, they do not have to be strictly a straight line. This allows us to capture much more complex relationships in the data. As a flaw, the time of training is higher, and they are computationally more expensive.

They are somewhat comparable to the Neural Networks: they usually are multi-input single-output models, and they allow non-linear mappings in a higher dimensional space. While with the Neural Networks this is achieved through the activation function, with SVMs it is obtained due to the Kernel function. These functions satisfy Mercer's Theorem and allow mapping to high-dimensional feature space:

$$K(x_i, \boldsymbol{x}) = \boldsymbol{\phi}(x_i)^{\boldsymbol{T}}\, \boldsymbol{\phi}(x)$$

Instead of finding the mapping function $\boldsymbol{\phi}(\cdot)$, and then $\boldsymbol{\phi}(\cdot)^{\boldsymbol{T}}\, \boldsymbol{\phi}(\cdot)$, it is easier to find the kernel function $\boldsymbol{K}(\cdot)$. This saves time of computation as the higher dimensional mapping can be found in one step instead of in two. The Kernel function allows SVMs to classify non-separable classes. There are several Kernel functions, some of the most commonly used are: linear kernel, polynomial kernel of degree q, Radial Basis Function (RBF) kernel, logarithmic kernel… In addition, a kernel can be constructed from other kernels, by a linear combination of kernels, or by a product of them.

We will explore this concept more thoroughly by using examples in this Coursework.

Q1. Write down the first 7 digits of your student ID as $s_1s_2s_3s_4s_5s_6s_7$. (5 marks)

$$s_1s_2s_3s_4s_5s_6s_7 = 2011499$$

Q2. Find R_1 which is the remainder of $\frac{s_1+s_2+s_3+s_4+s_5+s_6+s_7}{4}$. Table 1 shows the multi-class methods to be used corresponding to the value of R_1 obtained. (5 marks)

R_1	Method
0	One-against-one
1	One-against-all
2	**Binary decision tree**
3	Binary coded

Table 1: R_1 and its corresponding multi-class method

$$R_1 = \text{remainder of } \frac{2+0+1+1+4+9+9}{4} = \text{remainder of } \frac{26}{4} = 2$$

As we can see, we have to use the Binary Decision Tree as the approach for the multi-class problem. To solve a multi-class problem with SVMs, a combination of two-class SVMs (linear or nonlinear) is necessary. Some of the well-known approaches are the following:

- **One-against-one**: $\frac{R(R-1)}{2}$ binary SVMs are needed for a R-class problem. A two-class SVM is trained for each pair of classes, and the majority voting is used to choose the class. As a flaw, some regions cannot be classified. For a 3-class problem, we would have AvsB, AvsC and BvsC.

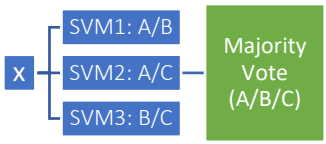

Figure 1: One-against-one approach for 3 classes

- **One-against-all**: R two-class SVMs are necessary for a R-class problem. Each SVM corresponds to a class, so it is trained for C_R versus all $C_j, j \neq R$. Then, like before, majority voting chooses the final class. Some regions cannot be classified using hard SVM classifiers, to deal with the undefined region, in some literature they just take the average. In this case, all regions can be classified using soft SVM classifiers. However, misclassification may occur near the boundary of a SVM. For a 3-class problem, we would have AvsNotA, BvsNotB and CvsNotC; or AvsB&C, BvsA&C and CvsA&B.

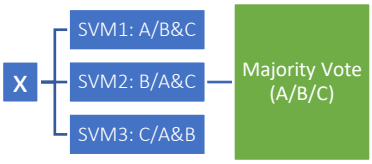

Figure 2: One-against-all approach for 3 classes

- **Binary Decision Tree**: $R-1$ two-class SVMs are necessary for a R-class problem. It consults at most $\lceil \log_2 R \rceil$ SVMs to make a final decision for sample $\boldsymbol{x}$, where $\lceil \cdot \rceil$ is the ceiling operator (rounding up). The classification accuracy depends heavily on the SVMs in the upper levels. If chosen badly, errors propagate and there is no turning back. We will investigate this type of multi-class SVM approach further on in the Coursework.

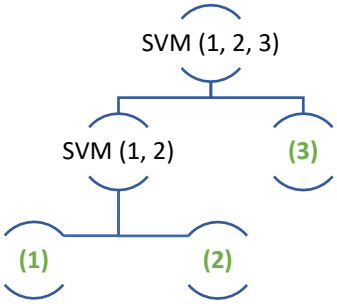

Figure 3: Binary decision tree approach for 3 classes

- **Binary coded approach**: it requires $\lceil \log_2 R \rceil$ SVMs for a problem with R classes. Each SVM cannot tell you which class it belongs to, but its combination can. One possible flaw is that **x** can lead to an undefined class. For a 3-class problem, we would have $\lceil \log_2 R \rceil = 2$ SVMs. One for the class A&B vs C, and another one for the class A&C vs B. In this example if both SVMs return -1, the class is undefined:

CLASS	$\boldsymbol{SVM_1}$ (A&B vs C)	$\boldsymbol{SVM_2}$ (A&C vs B)
A	1	1
B	1	-1
C	-1	-1

Table 2: class assignments for a binary code approach

Figure 4: Binary coded approach for 3 classes

Q3. Create a linearly separable two-dimensional dataset of your own, which consists of 3 classes. List the dataset in the format as shown in Table 2. Each class should contain at least 10 samples and all three classes have the same number of samples. *Note: This is your own created dataset. The chance of having the same dataset in other submissions is slim. Do not share your dataset with others to avoid any plagiarism/collusion issues*. (10 Marks)

This artificial dataset can be created through *scikit-learn*. This library has a package called *datasets*, this package creates customised datasets to experiment and test with models. Instead, to ease the calculations for the following parts we will create a customised dataset with integer values.

When represented in a 2D plot, the dataset looks like the following:

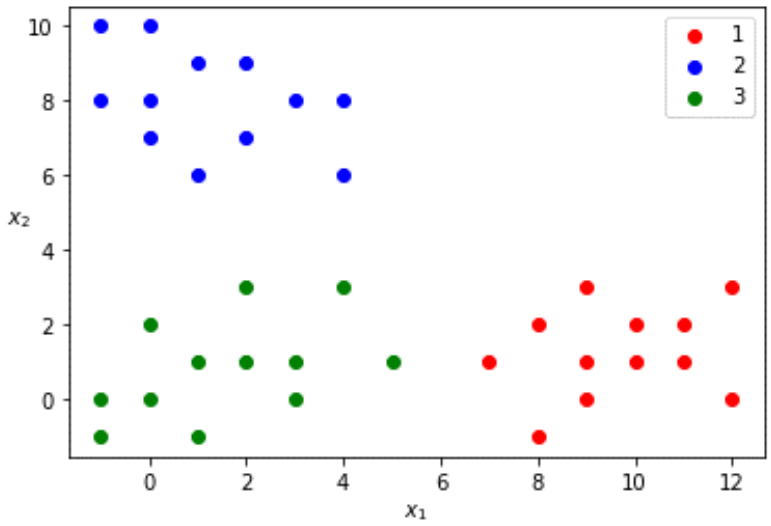

Figure 5: linearly separable dataset with 15 instances per class

Lastly, we will create a table like the one in the question 3 description to list the dataset:

INSTANCE	SAMPLE OF CLASS 1 (red)	SAMPLE OF CLASS 2 (blue)	SAMPLE OF CLASS 3 (green)
1	(7, 1)	(-1, 8)	(-1, -1)
2	(8, -1)	(-1, 10)	(-1, 0)
3	(8, 2)	(0, 7)	(0, 0)
4	(9, 0)	(0, 8)	(0, 2)
5	(9, 1)	(0, 10)	(1, -1)
6	(9, 3)	(1, 6)	(1, 1)
7	(10, 1)	(1, 9)	(2, 1)
8	(10, 2)	(2, 7)	(2, 3)
9	(11, 1)	(2, 9)	(3, 0)
10	(11, 2)	(3, 8)	(3, 1)
11	(12, 0)	(4, 6)	(4, 3)
12	(12, 3)	(4, 8)	(5, 1)

Table 3: Sample of three classes

Q4. Plot the dataset in Q3 to show that the samples are linearly separable. Explain why your dataset is linearly separable. *Hint: the Matlab built-in function plot can be used and show some example hyperplanes which can linearly separate the datasets. Identify which hyperplane is for which classes*. (20 Marks)

We plotted the dataset in the last question. Now, imagine a dataset of two classes represented in 2D, it will be linearly separable if there exists at least one line in the plane that separates the two groups of data instances. Extending it to a multi-class dataset, it will be linearly separable if there is a number of straight lines that separates the classes in our dataset. In our case, if every point can be grouped by its colour with 2 lines, the dataset is linearly separable.

Looking at the previous *Figure 5*, some straight lines that separate the dataset come to mind visually. For example, the straight line in $x_1 = 6$ could separate the class 1 from the other two classes (2&3), while a straight line in $x_2 = 4$ could separate classes 3 and 1&2. These two lines parallel to the axes demonstrate that our dataset is linearly separable by creating three groups of samples separated by straight lines. This example can be interpreted better with a 2D plot:

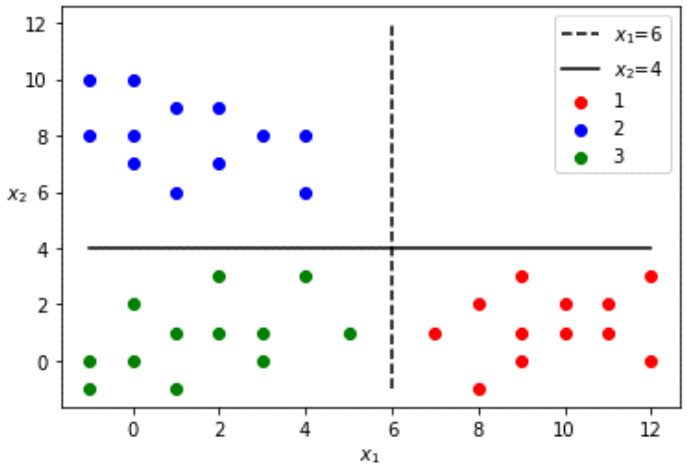

Figure 6: linearly separable dataset with 3 classes

As every class can be separated perfectly by a line (hyperplane), $x_2 = a \times x_1 + b$, this dataset is linearly separable. Here are more possibilities (there are infinite) that separate the datasets:

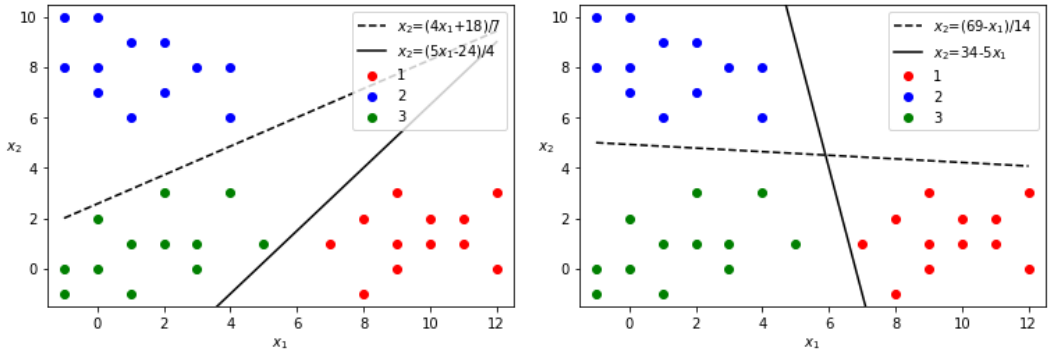

Figure 7: more possible hyperplanes that separate the data.

As we can see, the hyperplanes $x_2' = \frac{4x_1+18}{7}$ and $x_2' = \frac{5x_1-24}{4}$ at the left side; and the hyperplanes $x_2'' = \frac{69-x_1}{14}$ and $x_2'' = 34 - 5x_1$, at the right side, also separate, the three classes.

Q5. According to the method obtained in Q2, draw a block diagram at SVM level to show the structure of the multi-class classifier constructed by linear SVMs. Explain the design (e.g., number of inputs, number of outputs, number of SVMs used, class label assignment, etc.) and describe how this multi-class classifier works.

Remark: A blocking diagram is a diagram which is used to, say, show a concept or a structure, etc. Here in this question, a diagram is used to show the structure of the multi-class SVM classifier, i.e., how to put binary SVM classifiers together to work as a multi-class SVM classifier. For example, Q5 of tutorial 8 is an example of a block diagram at SVM level. Neural network diagram is a kind of diagram to show its structure at neuron level. The block diagrams in lecture 9 are to show the architecture of ensemble classifier, etc. (20 Marks)

We already introduced a little the four approaches in Q2: one-against-one, one-against-all, binary decision tree, and binary coded approach. Now, we will explain a bit further the one we are interested in for this task: binary decision tree.

The positive aspects of this kind of multi-class approach are many, the key one that differentiates this method from the others is the number of SVMs that have to be consulted. This is the only one that does not need to check the output of all SVMs, it consults at most $\lceil \log_2 R \rceil$ SVMs to make a final decision for sample $\boldsymbol{x}$, where R is the number of classes. This also shortens the time of training, as not all SVMs use the whole training dataset. Another advantage is that when adding classes, only SVMs for new classes are needed to be trained, the existing SVMs can be re-used again.

As disadvantages it has the opposite effect when removing classes. In this case, all SVMs are needed to be re-trained. As we said in Q2, the overall performance of this method depends heavily on the performance of the SVMs in the upper level. If it does not work very well, the error will propagate to the SVMs below. This is why it is important to choose wisely this upper SVM, by previsualising the data one can get an approximated idea of what suits better.

Referring to the block diagram, the structure of our binary decision tree is the following:

- **Number of inputs**: a data point. In our case, as we are dealing with 2-dimensional space data instances, it will be a feature vector with 2 dimensions.

- **Number of outputs**: we end up with one final output that is a class label. However, we will need to consult the binary outputs of at most $\lceil \log_2 3 \rceil = 2$ SVMs.

- **Number of SVMs used**: this approach will use R-1 SVMs, where R is the number of classes in our multi-class problem. The main advantage is that it will consult at most 2 SVMs, as we said before. In our 3-class problem, I would use an upper SVM ($\boldsymbol{SVM_1}$) that differentiates class 1 from classes 2&3, as they can be separated easy. Then I would use another SVM ($\boldsymbol{SVM_2}$) that would separate class 2 from class 3 if the upper SVM classified the data point as 2&3; if not, I would already know that the data point is from class 1 and only 1 SVM would be consulted. This major advantage could be compared with a binary decision tree classifier, or the binary tree data structure, where searches are much faster because a small number of nodes has to be consulted.

- **Class label assignment**: as we said before, a data point would be introduced in the first SVM ($\boldsymbol{SVM_1}$) which separates new instances in class 1 or in class 2&3. If this SVM returns $+1$, we would know it belongs to class 1 (red class). If not, if $\boldsymbol{SVM_1}$ returns -1, we would know it belongs to class 2 (blue class) or to class 3 (green class). To differentiate these two classes, we would make the data instance go through another SVM, $\boldsymbol{SVM_2}$, which would return $+1$ if the data instance belongs to the class 2, and -1 otherwise, if it belonged to class 3.

A block diagram to explain this multi-class SVM approach would look like something like this:

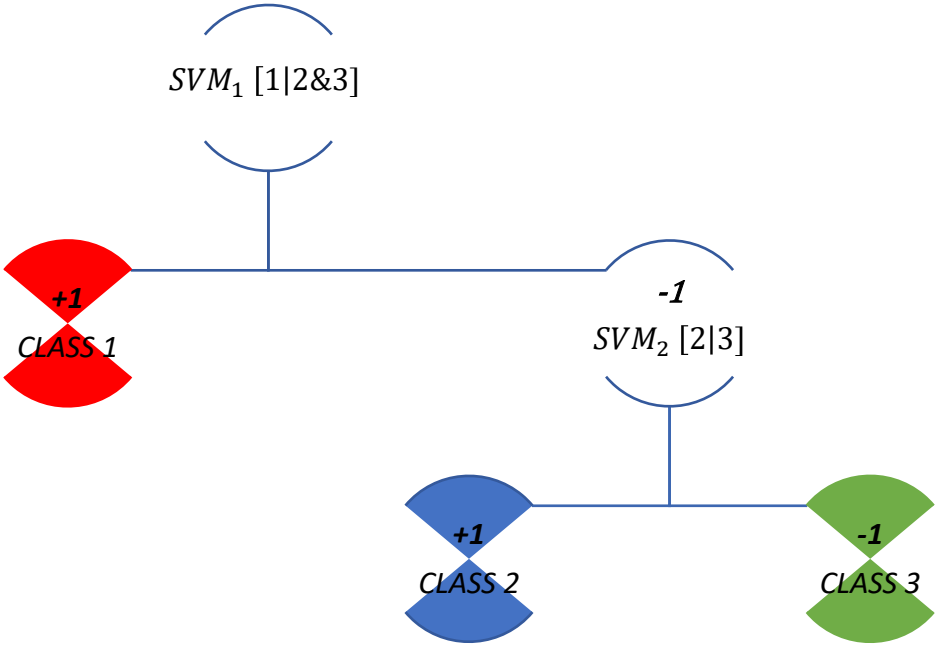

Figure 8: Block diagram for a binary decision tree approach at the SVM level

So, the path to each class would be the following:

- CLASS 1: the first SVM, $\boldsymbol{SVM_1}$, would have to output $+\mathbf{1}$. In this case, the second SVM would not have to be consulted.
- CLASS 2: the first SVM, $\boldsymbol{SVM_1}$, would have to output $-\mathbf{1}$. Later, the second SVM, $\boldsymbol{SVM_2}$, would have to output $+\mathbf{1}$.
- CLASS 3: the first SVM, $\boldsymbol{SVM_1}$, would have to output $-\mathbf{1}$. Later, the second SVM, $\boldsymbol{SVM_2}$, would have to output $-\mathbf{1}$.

Another way to show this example is using *Figure 6*, the one where we show the data points and two hyperplanes in the axes that separate the data. In this example, the straight line $x_1 = 6$ would perform like the $\boldsymbol{SVM_1}$, separating class 1 from classes 2&3. The other straight line, in the horizontal axis, $x_2 = 4$ would perform like $\boldsymbol{SVM_2}$, separating class 2 from class 3.

Then, the output of each SVM will determine the final class of the multi-class SVM. It could be interpreted in a more visual manner with the figure below:

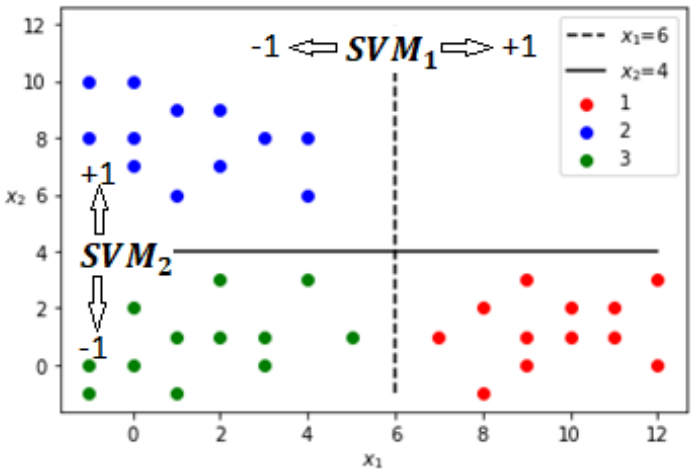

Figure 9: example of potential SVMs that would separate the data.

Note that these may not be the final SVMs because they do not maximise the margin. They are simply an example of how the class assignment would be.

Q6. According to your dataset in Q3 and the design of your multi-class classifier in Q5, identify the support vectors of the linear SVMs BY "INSPECTION" and design their hyperplanes by hand. Show the calculations and explain the details of your design. (20 Marks)

As this is a multi-class problem, we have to define two different SVMs. We will start by $\boldsymbol{SVM_1}$, the one that separates class 1 (red) from the other ones, classes 2 and 3 (blue and green, respectively).

1. $\boldsymbol{SVM_1}$: [1|2&3]

We have to follow 4 different steps: define labels for the classes, design the form of classifier (hyperplane), obtain the conditions from the SVM principle, and determine the parameter values of the SVM classifier.

1.1. Define labels for the classes: as we stated with the block diagram, the class 1 will have label $+1$ for SVM_1, and the other two classes will have label -1 for this first SVM. Therefore, our label $+1$ will have 15 data instances, and our label -1 will have 30 data instances. Hence, from the perspective of $\boldsymbol{SVM_1}$, the data will look like this:

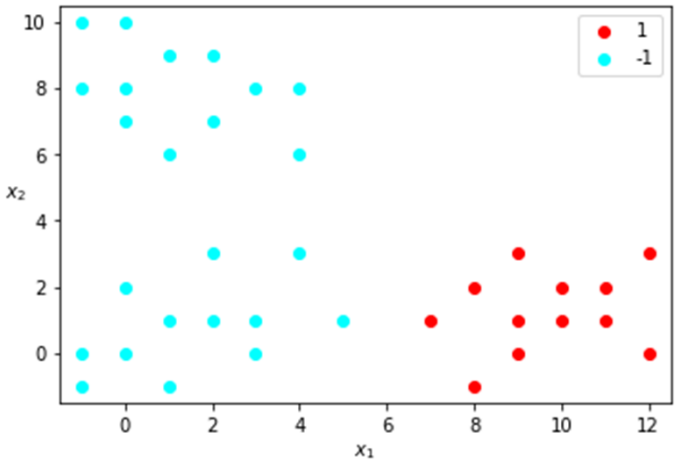

Figure 10: data instances for the first SVM, $\boldsymbol{SVM_1}$

I have simply represented the data instances from classes 2 and 3 with the colour cyan. As we can see in the upper right corner, its label is -1. The data instances from class 1 remain with the same colour.

The number of instances of each class does not matter very much, as the SVM only uses what is known as support vectors, data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. These are the points from which the SVM is defined [1]. These support vectors represent the samples that are the hardest to classify. In case any support vector is removed from or added to the dataset, it will affect the position and orientation of the dividing line or hyperplane.

[1] https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47#:~:text=Support%20vectors%20are%20data%20points,help%20us%20build%20our%20SVM.

1.2. Design the form of classifier (hyperplane): as our data is in a 2-dimensional space, our hyperplanes will have the following structure:

$$\boldsymbol{w^T x} + w_0 = w_1\boldsymbol{x_1} + w_2\boldsymbol{x_2} + w_0 = 0$$

1.3. Obtain the conditions from the SVM principle: the main goal of our problem is to design the optimal hyperplane so it can classify correctly (all) training instances. When we say optimal, we mean that it does not make errors (no misclassification) and that the distance between the nearest support vectors (from different classes) is as high as possible. In other words, we want to maximise the margin, which is the same as minimising the error rate.

In these kind of problems, the distance (z) of a point($\boldsymbol{x}$) from a hyperplane ($\boldsymbol{w}$) is:

$$z = \frac{|f(\boldsymbol{x})|}{\|\boldsymbol{w}\|} = \frac{|f(\boldsymbol{x})|}{\sqrt{w_1^2 + w_2^2}}$$

Where $f(\boldsymbol{x})$ is a data point location; $\boldsymbol{w}$ is the hyperplane, or the line in our 2-dimensional case; and $\|\cdot\|$ is the l^2 *norm operator*, also known as the Euclidean norm. Therefore, we want to maximise the margin, find the largest distance, z, between the hyperplane and the support vectors, which is given by:

$$z = \min_{\boldsymbol{x_i}:y_i=-\mathbf{1}} \frac{|f(\boldsymbol{x_i})|}{\|\boldsymbol{w}\|} + \min_{\boldsymbol{x_i}:y_i=+\mathbf{1}} \frac{|f(\boldsymbol{x_i})|}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|}\left(\min_{\boldsymbol{x_i}:y_i=-\mathbf{1}} |f(\boldsymbol{x_i})| + \min_{\boldsymbol{x_i}:y_i=+\mathbf{1}} |f(\boldsymbol{x_i})|\right) = \frac{2}{\|\boldsymbol{w}\|}$$

Where $\left(\min_{\boldsymbol{x_i}:y_i=-\mathbf{1}} |f(\boldsymbol{x_i})| + \min_{\boldsymbol{x_i}:y_i=+\mathbf{1}} |f(\boldsymbol{x_i})|\right) = 1 + 1$ because the minimal $f(\boldsymbol{x_i}) = -1$ for the samples with label -1, and $f(\boldsymbol{x_i}) = +1$ for instances with label $+1$. These distances correspond to the support vectors, the ones that are the closest to the hyperplane, or line.

Now, we have a **constrained optimisation problem**. Once we have the margin defined, we need to maximise it, which is the same as minimising $\|\boldsymbol{w}\|$. Hence, our optimisation problem can be defined like the following:

$$\min_{\boldsymbol{w}} J(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|$$
$$subject\ to\ y_i(\boldsymbol{w^T x} + w_0) \geq 1 \quad i = 1, 2, \ldots, N$$

The constraint "$y_i(\boldsymbol{w^T x} + w_0) \geq 1$" is to ensure that all samples $\boldsymbol{x_i}$ are correctly classified. N is the number of samples in our training set.

To solve this problem we will help ourselves with the **method of the Lagrange multipliers**. These multipliers are used when we want to find extrema of a given function $f(x, y, z, \ldots)$, subject to the condition $g(x, y, z, \ldots) = k$. The idea used in the Lagrange multiplier is that the gradient of the objective function *f*, lines up either in parallel or anti-parallel direction to the gradient of the constraint *g*, at an optimal point. In such case, one of the gradients should be some multiple of another [2].

[2] https://towardsdatascience.com/understanding-support-vector-machine-part-1-lagrange-multipliers-5c24a52ffc5e

Hence, our problem is now defined like the following *Primal Problem:*

$$\mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \lambda_i \left(y_i\left(\boldsymbol{w^T x_i} + w_0\right) - 1\right)$$

$$where\ \lambda_i = [\lambda_1 \ \ \lambda_2 \ \ \ldots \ \ \lambda_N], \lambda_i \geq 0\ for\ all\ i = 1, 2, \ldots, N\ (number\ of\ samples)$$

The above primal problem can be transformed to the following *Dual Problem*:

$$\min_{\boldsymbol{w}, \boldsymbol{w_0}} \max_{\boldsymbol{\lambda} \geq 0} \left(\frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \lambda_i \left(y_i\left(\boldsymbol{w^T x_i} + w_0\right) - 1\right) \right)$$

Consequently, we have to minimise the cost and maximise the Lagrange multipliers. We will start by minimising the cost, finding where the derivative is null:

(1) $$\frac{\partial \mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda})}{\partial \boldsymbol{w}} = 0 \rightarrow \boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x_i}$$

(2) $$\frac{\partial \mathcal{L}(\boldsymbol{w}, w_0, \boldsymbol{\lambda})}{\partial w_0} = 0 \rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

Now, putting these two equations (1) and (2) in the *Primal Problem*, $\mathcal{L}(\boldsymbol{w}, \boldsymbol{w_0}, \boldsymbol{\lambda})$, we obtain an expression that will depend on $\boldsymbol{\lambda}$. We start with the following equation:

$$\mathcal{L}(\boldsymbol{\lambda}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \lambda_i \left(y_i\left(\boldsymbol{w^T x} + w_0\right) - 1\right) =$$

Now we substitute and square (1) in the first element and we rearrange the terms in the second element of the *Primal Problem*:

$$= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \boldsymbol{x_i^T x_j} - \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{w^T x_i} - \sum_{i=1}^{N} \lambda_i y_i w_0 + \sum_{i=1}^{N} \lambda_i =$$

Once again, we substitute (1) in the second element of the previous equation. We end up with the same value as before.

$$= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \boldsymbol{x_i^T x_j} - \sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \boldsymbol{x_i^T x_j} - \sum_{i=1}^{N} \lambda_i y_i w_0 + \sum_{i=1}^{N} \lambda_i =$$

Now, as w_0 is a constant parameter, we know from (2) that $\sum_{i=1}^{N} \boldsymbol{\lambda_i} y_i = 0$. Therefore, the third element of the previous equation is cancelled. The two first elements are the same, so they can be combined. Finally, after all this algebra, we end up with the following expression:

$$\boxed{= \sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \boldsymbol{x_i^T x_j}}$$

On that account, the *Dual Problem* is reduced to:

(3) $$\boxed{\max_{\lambda_i \geq 0} \sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \boldsymbol{x_i^T x_j} \quad subject\ to\ \sum_{i=1}^{N} \lambda_i y_i = 0, \lambda_i \geq 0, i = 1, 2, \ldots, N}$$

The solution to this previous equation can be found by applying quadratic programming solver, this solution to $\boldsymbol{\lambda_i}$ may not be unique. However, the hyperplane characterised by

$\boldsymbol{w}$ and w_0 is unique. λ_i is a scalar function, which does not depend explicitly on the dimension of the input space.

As we said before, the straight line will be defined by the support vectors. This data points will have $\lambda_i \neq 0$, thus, their value will be $\lambda_i > 0$. For the rest of the data points, those that will not have any effect on the hyperplane, their value is $\lambda_i = 0$. Thus, from (1) we will have:

$$\boldsymbol{w} = \sum_{\boldsymbol{i=1}}^{N_S} \lambda_i y_i \boldsymbol{x_i}\,, N_S \leq N$$

Where N_S denotes the number of support vectors. As we can see, the only points that characterise the linear separator are the support vectors (N_S). They lie on the two hyperparameters satisfying $\boldsymbol{w^T x_i} + w_0 = \pm 1$, depending on the class they belong to. Now, rearranging the terms in $\mathcal{L}(\lambda)$**:**

$$\mathcal{L}(\boldsymbol{\lambda}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \lambda_i \big(y_i\big(\boldsymbol{w^T x_i} + w_0\big) - 1\big)$$
$$= \frac{1}{2}\boldsymbol{w^T} \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x_i} + \frac{1}{2}\sum_{i=1}^{N} \lambda_i y_i w_0 - \sum_{i=1}^{N} \lambda_i \big(y_i\big(\boldsymbol{w^T x_i} + w_0\big) - 1\big)$$
$$= \frac{1}{2}\sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N} \lambda_i \big(y_i\big(\boldsymbol{w^T x_i} + w_0\big) - 1\big)$$

Mainly, in the second line we expand the first term, obtaining half of the second term. So, when we subtract it to the last term in the second line, we obtain the final result shown in the third line.

Now, we want to maximise $\mathcal{L}$ with respect to $\boldsymbol{\lambda}$. Since the definition of the Lagrange multipliers states that $\lambda_i \geq 0$, and from the definition of the constrained optimisation problem, $y_i\big(\boldsymbol{w^T x_i} + w_0\big) - 1 \geq 0$ to ensure that all samples $\boldsymbol{x_i}$ are correctly classified.

One possibility is to have $\lambda_i\big(y_i\big(\boldsymbol{w^T x_i} + w_0\big) - 1\big) = 0$ for all $y_i\big(\boldsymbol{w^T x_i} + w_0\big) - 1 \geq 0$, $i = 1, 2, \dots, N$. This option is supported by the Karush-Kuhn-Tucker (KKT) conditions [3].

Then, as $y_i\big(\boldsymbol{w^T x_i} + w_0\big) - 1 \geq 0$, it suggests that some $\lambda_i = 0$ for those $\boldsymbol{x_i}$ not being a support vector. Finally, we end up that only the support vectors matter and characterise the hyperparameter or the straight line that separates the two classes:

$$Support\ vector\ \boldsymbol{x_i}\colon y_i\big(\boldsymbol{w^T x_i} + w_0\big) - 1 = 0 \rightarrow \lambda_i \neq 0$$
$$Non - support\ vector\ \boldsymbol{x_i}\colon y_i\big(\boldsymbol{w^T x_i} + w_0\big) - 1 > 0 \rightarrow \lambda_i = 0$$

As a summary, the linear SVM classifier (linearly separable case) can be found by solving the solution ($\boldsymbol{w}, w_0$ and $\boldsymbol{\lambda_i}$) to the following conditions:

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, w_0, \lambda)}{\partial \boldsymbol{w}} = 0 \rightarrow \boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x_i}$$

[3] https://www.cs.cmu.edu/~ggordon/10725-F12/slides/16-kkt.pdf

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, w_0, \lambda)}{\partial \boldsymbol{w_0}} = 0 \rightarrow \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \lambda_i y_i = \mathbf{0}$$

$$\lambda_i \geq 0, i = 1, 2, \ldots, N$$

$$\lambda_i \left(y_i \left(\boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x_i} + w_0\right) - 1\right) = 0, i = 1, 2, \ldots, N$$

For this exercise we will use the hard classifier once we have the straight line defined:

Hard classifier: $f(x) = sgn\left(\boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x} + w_0\right)$, where $sgn(z) = \begin{cases} -1 \; if \; z < 0 \\ +1 \; if \; z \geq 0 \end{cases}$

1.4. Determine the parameter values of the SVM classifier: once we have the problem correctly defined, we are in position of determining the values of our straight line, as we are in a 2D space problem. We will start by choosing the support vectors for the first SVM, $\boldsymbol{SVM_1}$, of our problem:

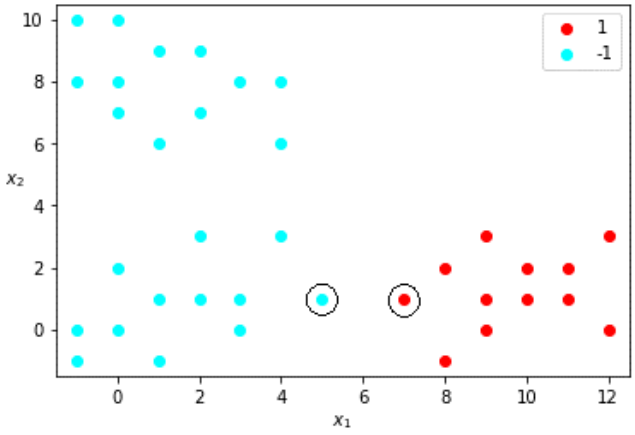

Figure 11: Support vectors chosen for $\boldsymbol{SVM_1}$

These are the support vectors we have chosen to define our $\boldsymbol{SVM_1}$. We have chosen these ones because we believe they lie the closest to the straight line we want to define. These support vectors have the following coordinates (from *Table 3*):

SUPPORT VECTORS FOR $\boldsymbol{SVM_1}$			
Nº INSTANCE	**SAMPLE OF CLASS +1 (red)**	**Nº INSTANCE**	**SAMPLE OF CLASS -1 (cyan)**
1 [1]	(7, 1)	**12** (green) **[2]**	(5, 1)

Table 4: Support vectors for $\boldsymbol{SVM_1}$

Once we have our support vectors located by inspection, the two classes can be separated by constructing a hyperplane in the region in between. This hyperplane will be characterised by:

$$\boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x} + w_0 = w_1 x_1 + w_2 x_2 + w_0 = 0$$

$$\boldsymbol{w} = \lambda_1 y_1 \boldsymbol{x_1} + \lambda_2 y_2 \boldsymbol{x_2} = \lambda_1 \begin{bmatrix} 7 \\ 1 \end{bmatrix} - \lambda_2 \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

We have only put in the expression the values λ_i of the support vectors, as the rest are $\lambda_i = 0$. We have named them from 1 to 2 because with the naming we used in *Table 3* it is hard to distinguish them. The support vector 1 belongs to class $+1$, which explains

why it adds to the equation ($y_1 = +1$). On the other hand, the support vector 2 belongs to class -1, which explains why the minus in the equation ($y_2 = -1$).

We also stated before that for the support vectors:

$$y_i(\boldsymbol{w^T x_i} + w_0) = 1$$

Ergo, for each support vector:

$$\boldsymbol{x_1}(y_1 = 1):$$

$$1 \times \left(\left(\lambda_1 \begin{bmatrix}7\\1\end{bmatrix} - \lambda_2 \begin{bmatrix}5\\1\end{bmatrix}\right)^T \begin{bmatrix}7\\1\end{bmatrix} + w_0\right) = 1$$

$$50\lambda_1 - 36\lambda_2 + w_0 = 1$$

$$\boldsymbol{x_2}(y_2 = -1):$$

$$-1 \times \left(\left(\lambda_1 \begin{bmatrix}7\\1\end{bmatrix} - \lambda_2 \begin{bmatrix}5\\1\end{bmatrix}\right)^T \begin{bmatrix}5\\1\end{bmatrix} + w_0\right) = 1$$

$$36\lambda_1 - 26\lambda_2 + w_0 = -1$$

After rearranging the two equalities, we end up with the following equations:

$$50\lambda_1 - 36\lambda_2 + w_0 = 1$$

$$36\lambda_1 - 26\lambda_2 + w_0 = -1$$

Lastly, when we minimised the cost, we arrived at the following equality:

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, w_0, \lambda)}{\partial w_0} = 0 \rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

Therefore:

$$\lambda_1 - \lambda_2 = 0 \rightarrow \lambda_1 = \lambda_2$$

Finally, we have 3 equations and 3 unknown factors. We are in position of solving the unknowns. We can reconsider this problem with matrixes:

$$\begin{bmatrix}50 & -36 & 1\\ 36 & -26 & 1\\ 1 & -1 & 0\end{bmatrix}\begin{bmatrix}\lambda_1\\ \lambda_2\\ w_0\end{bmatrix} = \begin{bmatrix}1\\ -1\\ 0\end{bmatrix}$$

Once we have this system of linear equations, we can solve it with a computer. We could make use of the library *NumPy* and its package called *linalg*, perfect for algebra. By applying the function *linalg.solve()* we obtain the solution to these unknowns. For the function to work, we have to pass as parameters the 3×3 matrix of the left, and the array of length 3 at the right.

We can also calculate it knowing that $\lambda_1 = \lambda_2$:

$$50\lambda_1 - 36\lambda_1 + w_0 = 1 = 14\lambda_1 + w_0$$

$$36\lambda_1 - 26\lambda_1 + w_0 = -1 = 10\lambda_1 + w_0$$

We subtract both equations to eliminate w_0:

$$2 = 4\lambda_1 \rightarrow \lambda_1 = 0.5 = \lambda_2$$

$$w_0 = 1 - 14\lambda_1 = 1 - 7 = -6$$

The results obtained are the following:

$$\lambda_1 = 0.5$$
$$\lambda_2 = 0.5$$
$$w_0 = -6$$

Now, recalling the equation we showed at the beginning to define our straight line:

$$\boldsymbol{w} = \lambda_1 y_1 \boldsymbol{x_1} + \lambda_2 y_2 \boldsymbol{x_2}$$

Therefore,

$$\boldsymbol{w} = \lambda_1 \begin{bmatrix} 7 \\ 1 \end{bmatrix} - \lambda_2 \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

$$\boldsymbol{w} = 0.5 \begin{bmatrix} 7 \\ 1 \end{bmatrix} - 0.5 \begin{bmatrix} 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

After all the calculations, we end up with a pretty simple straight line, where:

$$w_1 = 1, w_2 = 0, \text{ and from before, } w_0 = -6$$

Hence,

$$\boldsymbol{w^T x} + w_0 = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 6 = x_1 - 6 = 0$$

This line, when represented in a 2-dimensional space, looks like the one we drew at the beginning of this coursework. It is a line parallel to the vertical axis that cuts the horizontal axis in 6:

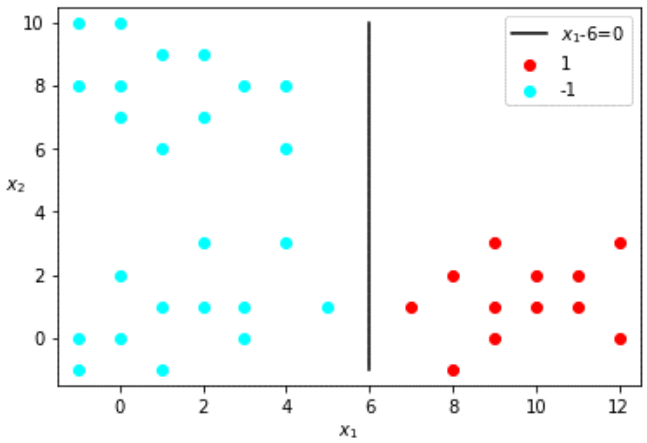

Figure 12: straight line of the first SVM, $\boldsymbol{SVM_1}$

Now we can test if the classifier works as we wish. As we can see, this SVM only cares about the x_1 value, as x_2 does not appear in the line $\boldsymbol{w}$. Therefore, when applying the

hard classifier (see equation below), every point at the left of $x_1 = 6$ will return a negative value, classifying that point as class -1, as we designed it. On the other hand, every point at the right of $x_1 = 6$ will return a positive value, classifying that point with the label $+1$.

Hard classifier: $f(x) = sgn(\boldsymbol{w^T x} + w_0)$, where $sgn(z) = \begin{cases} -1 \; if \; z < 0 \\ +1 \; if \; z \geq 0 \end{cases}$

Also, as we stated before, the support vectors accomplish:

$$y_i(\boldsymbol{w^T x_i} + w_0) = 1$$

$$\boldsymbol{x_1} = \begin{bmatrix} 7 \\ 1 \end{bmatrix}, (y_1 = 1) \rightarrow \mathbf{1} \times \left(\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 7 \\ 1 \end{bmatrix} + (-6) \right) = \mathbf{1} \times (\mathbf{1} \times \mathbf{7} + \mathbf{0} \times \mathbf{1} - 6) = \mathbf{1}$$

$$\boldsymbol{x_2} = \begin{bmatrix} 5 \\ 1 \end{bmatrix}, (y_1 = -1) \rightarrow (-\mathbf{1}) \times \left(\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 5 \\ 1 \end{bmatrix} + (-6) \right) = (-\mathbf{1}) \times (\mathbf{1} \times \mathbf{5} + \mathbf{0} \times \mathbf{1} - 6) = \mathbf{1}$$

2. $\boldsymbol{SVM_2}$: [2|3]

For this SVM, we only have to focus on the data points from class 2 and 3. The data samples that belong to class 1 are, presumably, already classified. As we said before, we have to follow four steps to design our second SVM, $\boldsymbol{SVM_2}$.

2.1. Define labels for the classes: as we stated with the block diagram, the class 2 will have label $+1$ for $\boldsymbol{SVM_2}$, and the other class, class 3, will have label -1 for this second SVM. Therefore, in this case, both our classes will have 15 data instances.

As we can see in the following figure, this SVM only cares about the instances of class 2 and 3. As we stated in the design of our multi-class SVM in Q5, the data instances from class 2 (blue) will return $+1$, and the data samples from class 3 (green) will return -1. This can be verified taking a look at the upper right corner of the following figure:

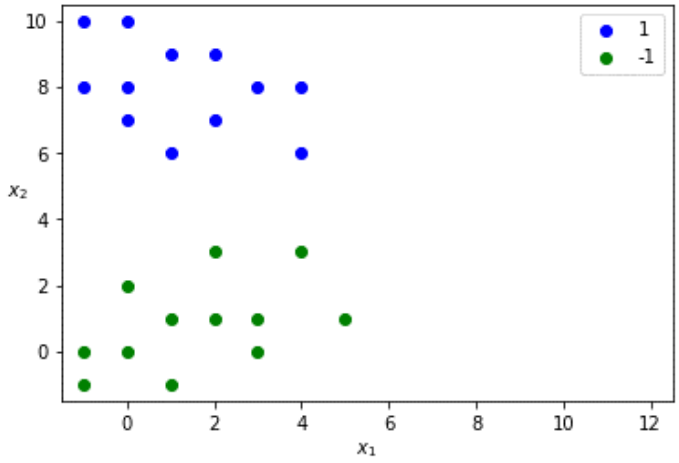

Figure 13: data instances for the first SVM, $\boldsymbol{SVM_2}$

2.2. Design the form of classifier (hyperplane): as our data is in a 2-dimensional space, our straight line will have the following structure, just like in the previous SVM:

$$\boldsymbol{w^T x} + w_0 = w_1 x_1 + w_2 x_2 + w_0 = 0$$

We have to obtain the values of w_0, w_1 and w_2 to define our straight line. w_0 will be found after applying the method of the Lagrange multipliers, alongside with λ_i, knowing that $\lambda_i \geq 0$, where $i = 1, 2, \ldots, N_S$, and N_S is the number of support vectors defined in our SVM.

After obtaining the Lagrange multipliers (λ_i), we can obtain the other two terms that will define our straight line (w_1 and w_2). In the next point we can see the equations that define these parameters.

2.3. Obtain the conditions from the SVM principle: the theory behind these conditions is the same as we explained before. As a summary, we will remind which are the main equations that will help us design this classifier:

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, w_0, \lambda)}{\partial \boldsymbol{w}} = 0 \rightarrow \boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x_i}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, w_0, \lambda)}{\partial w_0} = 0 \rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\lambda_i \geq 0, i = 1, 2, \ldots, N$$

$$\lambda_i \big(y_i \big(\boldsymbol{w^T x_i} + w_0 \big) - 1 \big) = 0, where\ i = 1, 2, \ldots, N$$

Then, once we have the straight line defined, we will use the hard classifier to label the data instances:

Hard classifier: $f(\boldsymbol{x}) = sgn\big(\boldsymbol{w^T x} + w_0\big)$, where $sgn(z) = \begin{cases} -1\ if\ z < 0 \\ +1\ if\ z \geq 0 \end{cases}$

2.4. Determine the parameter values of the SVM classifier: once we have the problem correctly defined, we are in position of determining the values of our straight line, as we are in a 2D space problem. If there were more dimensions, we would be talking about a hyperplane. We will start by finding the support vectors for the second and last SVM, $\boldsymbol{SVM_2}$, of our problem.

As we said before, this time we only care about the data instances from classes 2 and 3 (labels $+1$ and -1, respectively). In a binary decision tree, once we have divided the data instances in the upper SVM, we do not care more about them. So, in our case, once we have classified it as class 2&3 ($\boldsymbol{SVM_1}$ outputs -1), the lower SVM only focuses on those other classes.

Meaning that, this SVM will only have support vectors from class 2 and 3. The upper SVM only had support vectors from classes 1 and 3, but because those were the ones

that defined better the straight line. It could have been that a sample from class 2 could have defined the straight line of $\boldsymbol{SVM_1}$.

Therefore, $\boldsymbol{SVM_2}$ will have the following support vectors:

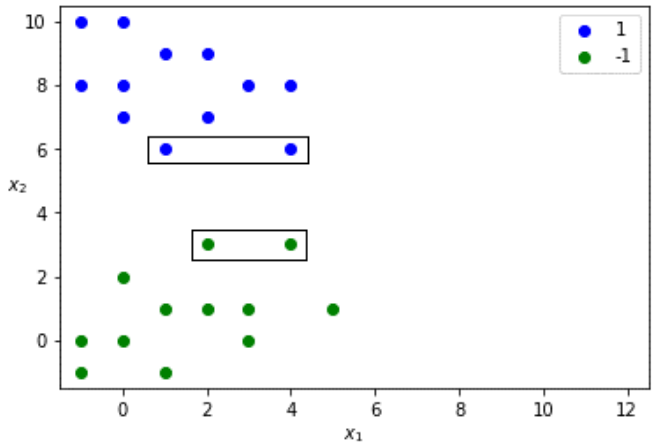

Figure 14: Support vectors chosen for $\boldsymbol{SVM_2}$.

These are the support vectors we have chosen to define our $\boldsymbol{SVM_2}$. As it can be observed in *Figure 14*, we have removed the data instances from class 1 as they do not matter for this part.

We have chosen these support vectors because we believe they lie the closest to the straight line we want to define. These support vectors have the following coordinates (from *Table 3*):

SUPPORT VECTORS FOR SVM_2			
Nº INSTANCE	**SAMPLE OF CLASS +1 (blue)**	**Nº INSTANCE**	**SAMPLE OF CLASS -1 (green)**
6 [1]	(1, 6)	**8 [3]**	(2, 3)
11 [2]	(4, 6)	**11 [4]**	(4, 3)

Table 5: Support vectors for $\boldsymbol{SVM_2}$

Once we have our support vectors located by inspection, the two classes can be separated by constructing a hyperplane in the region in between. This hyperplane will be characterised by:

$$\boldsymbol{w^T x} + w_0 = w_1 x_1 + w_2 x_2 + w_0 = 0$$

Where $\boldsymbol{w}$ is defined like:

$$\boldsymbol{w} = \lambda_1 y_1 \boldsymbol{x_1} + \lambda_2 y_2 \boldsymbol{x_2} + \lambda_3 y_3 \boldsymbol{x_3} + \lambda_4 y_4 \boldsymbol{x_4} = \lambda_1 \begin{bmatrix} 1 \\ 6 \end{bmatrix} + \lambda_2 \begin{bmatrix} 4 \\ 6 \end{bmatrix} - \lambda_3 \begin{bmatrix} 2 \\ 3 \end{bmatrix} - \lambda_4 \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

Just as before, we have only put in the expression the values λ_i for the support vectors, as the rest are $\lambda_i = 0$. The naming we used is just as the one used for $\mathbf{SVM_1}$, it is shown in *Table 5* between brackets, [·].

We also stated before that for the support vectors:

$$y_i(\boldsymbol{w^T x_i} + w_0) = 1$$

Ergo, for each support vector:

$$\boldsymbol{x_1}(y_1 = 1):$$

$$1 \times \left(\left(\lambda_1 \begin{bmatrix} 1 \\ 6 \end{bmatrix} + \lambda_2 \begin{bmatrix} 4 \\ 6 \end{bmatrix} - \lambda_3 \begin{bmatrix} 2 \\ 3 \end{bmatrix} - \lambda_4 \begin{bmatrix} 4 \\ 3 \end{bmatrix} \right)^T \begin{bmatrix} 1 \\ 6 \end{bmatrix} + w_0 \right) = 1$$

$$\boldsymbol{37}\lambda_1 + 40\lambda_2 - 20\lambda_3 - 22\lambda_4 + w_0 = 1$$

$$\boldsymbol{x_2}(y_2 = 1):$$

$$1 \times \left(\left(\lambda_1 \begin{bmatrix} 1 \\ 6 \end{bmatrix} + \lambda_2 \begin{bmatrix} 4 \\ 6 \end{bmatrix} - \lambda_3 \begin{bmatrix} 2 \\ 3 \end{bmatrix} - \lambda_4 \begin{bmatrix} 4 \\ 3 \end{bmatrix} \right)^T \begin{bmatrix} 4 \\ 6 \end{bmatrix} + w_0 \right) = 1$$

$$\boldsymbol{40}\lambda_1 + 52\lambda_2 - 26\lambda_3 - 34\lambda_4 + w_0 = 1$$

$$\boldsymbol{x_3}(y_3 = -1):$$

$$-1 \times \left(\left(\lambda_1 \begin{bmatrix} 1 \\ 6 \end{bmatrix} + \lambda_2 \begin{bmatrix} 4 \\ 6 \end{bmatrix} - \lambda_3 \begin{bmatrix} 2 \\ 3 \end{bmatrix} - \lambda_4 \begin{bmatrix} 4 \\ 3 \end{bmatrix} \right)^T \begin{bmatrix} 2 \\ 3 \end{bmatrix} + w_0 \right) = 1$$

$$\boldsymbol{20}\lambda_1 + 26\lambda_2 - 13\lambda_3 - 17\lambda_4 + w_0 = -1$$

$$\boldsymbol{x_2}(y_2 = -1):$$

$$-1 \times \left(\left(\lambda_1 \begin{bmatrix} 1 \\ 6 \end{bmatrix} + \lambda_2 \begin{bmatrix} 4 \\ 6 \end{bmatrix} - \lambda_3 \begin{bmatrix} 2 \\ 3 \end{bmatrix} - \lambda_4 \begin{bmatrix} 4 \\ 3 \end{bmatrix} \right)^T \begin{bmatrix} 4 \\ 3 \end{bmatrix} + w_0 \right) = 1$$

$$22\lambda_1 + 34\lambda_2 - 17\lambda_3 - 25\lambda_4 + w_0 = -1$$

After rearranging the four equalities, we end up with the following equations:

$$\begin{aligned} \boldsymbol{37}\lambda_1 + 40\lambda_2 - 20\lambda_3 - 22\lambda_4 + w_0 &= \quad 1 \\ 4\boldsymbol{0}\lambda_1 + 52\lambda_2 - 26\lambda_3 - 34\lambda_4 + w_0 &= \quad 1 \\ \boldsymbol{20}\lambda_1 + 26\lambda_2 - 13\lambda_3 - 17\lambda_4 + w_0 &= -1 \\ 22\lambda_1 + 34\lambda_2 - 17\lambda_3 - 25\lambda_4 + w_0 &= -1 \end{aligned}$$

Lastly, when we minimised the cost, we arrived at the following equation:

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, w_0, \lambda)}{\partial w_0} = 0 \rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

Therefore:

$$\lambda_1 + \lambda_2 - \lambda_3 - \lambda_4 = 0$$

Finally, we have 5 equations and 5 unknown factors. We are in position of solving the unknowns. We can reconsider this problem with matrixes, so we are able to solve it with *NumPy* and its *linalg* package, as explained before. After all, it is a system of 5 linear equations.

$$\begin{bmatrix} 37 & 40 & -20 & -22 & 1 \\ 40 & 52 & -26 & -34 & 1 \\ 20 & 26 & -13 & -17 & 1 \\ 22 & 34 & -17 & -25 & 1 \\ 1 & 1 & -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ w_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ 0 \end{bmatrix}$$

After solving this with algebra, we arrive to the following values:

$$\lambda_1 = 0.02814$$
$$\lambda_2 = 0.19407$$
$$\lambda_3 = 0.04222$$
$$\lambda_4 = 0.18$$
$$w_0 = -3$$

Now, recalling the equation we showed before to define our straight line:

$$\boldsymbol{w} = \lambda_1 y_1 \boldsymbol{x_1} + \lambda_2 y_2 \boldsymbol{x_2} + \lambda_3 y_3 \boldsymbol{x_3} + \lambda_4 y_4 \boldsymbol{x_4}$$

Therefore,

$$\boldsymbol{w} = \lambda_1 \begin{bmatrix} 1 \\ 6 \end{bmatrix} + \lambda_2 \begin{bmatrix} 4 \\ 6 \end{bmatrix} - \lambda_3 \begin{bmatrix} 2 \\ 3 \end{bmatrix} - \lambda_4 \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

$$\boldsymbol{w} = 0.02814 \begin{bmatrix} 1 \\ 6 \end{bmatrix} + 0.19407 \begin{bmatrix} 4 \\ 6 \end{bmatrix} - 0.04222 \begin{bmatrix} 2 \\ 3 \end{bmatrix} - 0.18 \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.667 \end{bmatrix}$$

After all the calculations, we end up with a pretty simple straight line, where:

$w_1 = 0, w_2 = {}^{2}/_{3}$, and from before, $w_0 = -3$

Hence,

$$\boldsymbol{w^T x} + w_0 = \begin{bmatrix} 0 & {}^{2}/_{3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 3 = \frac{2x_2}{3} - 3 = 0$$

This line represented in a 2-dimensional plot would look like this:

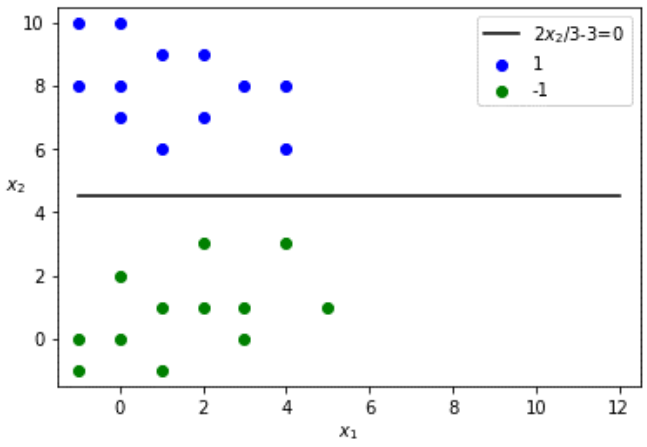

Figure 15: straight line of the second SVM, $\boldsymbol{SVM_2}$

As we can see, this line divides the two data groups perfectly, and they also maximise the margin between them. It could be expressed also as:

$$\frac{2x_2}{3} - 3 = 0 \rightarrow x_2 - \frac{3 \times 3}{2} = 0 \rightarrow x_2 - 4.5 = 0$$

Now we can test if the classifier works as we wish. As we can see, this SVM only cares about the x_2 value, as x_1 does not appear in the line $\boldsymbol{w}$. It is a straight line parallel to the horizontal axis that cuts the vertical axis in $x_2 = 4.5$.

Therefore, when applying the hard classifier (see equation below), every point above $x_2 = 4.5$ will return a positive value, classifying that point as class $+1$, as we designed it. On the other hand, every point below $x_2 = 4.5$ will return a negative value, classifying that point with the label -1.

Hard classifier: $f(\boldsymbol{x}) = sgn(\boldsymbol{w^T x} + w_0)$, where $sgn(z) = \begin{cases} -1 \; if \; z < 0 \\ +1 \; if \; z \geq 0 \end{cases}$

Also, as we stated before, the support vectors accomplish:

$$y_i(\boldsymbol{w^T x_i} + w_0) = 1$$

$$\boldsymbol{x_1} = \begin{bmatrix} 1 \\ 6 \end{bmatrix}, (y_1 = 1) \rightarrow \mathbf{1} \times \left(\begin{bmatrix} 0 & {}^2/_3 \end{bmatrix} \begin{bmatrix} 1 \\ 6 \end{bmatrix} + (-3) \right) = \mathbf{1} \times \left(\mathbf{0} \times \mathbf{1} + {}^2/_3 \times \mathbf{6} - 3 \right) = \mathbf{1}$$

$$\boldsymbol{x_2} = \begin{bmatrix} 4 \\ 6 \end{bmatrix}, (y_2 = 1) \rightarrow \mathbf{1} \times \left(\begin{bmatrix} 0 & {}^2/_3 \end{bmatrix} \begin{bmatrix} 4 \\ 6 \end{bmatrix} + (-3) \right) = \mathbf{1} \times \left(\mathbf{0} \times \mathbf{4} + {}^2/_3 \times \mathbf{6} - 3 \right) = \mathbf{1}$$

$$\boldsymbol{x_3} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, (y_3 = -1) \rightarrow (-\mathbf{1}) \times \left(\begin{bmatrix} 0 & {}^2/_3 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} + (-3) \right) = (-\mathbf{1}) \times \left(\mathbf{0} \times \mathbf{2} + {}^2/_3 \times \mathbf{3} - 3 \right) = \mathbf{1}$$

$$\boldsymbol{x_4} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}, (y_4 = -1) \rightarrow (-\mathbf{1}) \times \left(\begin{bmatrix} 0 & {}^2/_3 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} + (-3) \right) = (-\mathbf{1}) \times \left(\mathbf{0} \times \mathbf{4} + {}^2/_3 \times \mathbf{3} - 3 \right) = \mathbf{1}$$

We have already designed our multi-class SVM. It looks quite similar to the one we defined in *Figure 9* of Q5, but this one does maximise the margin at the second SVM, $\mathbf{SVM_2}$. We are going to plot a similar figure. However this time we will put the actual straight lines that divide the data:

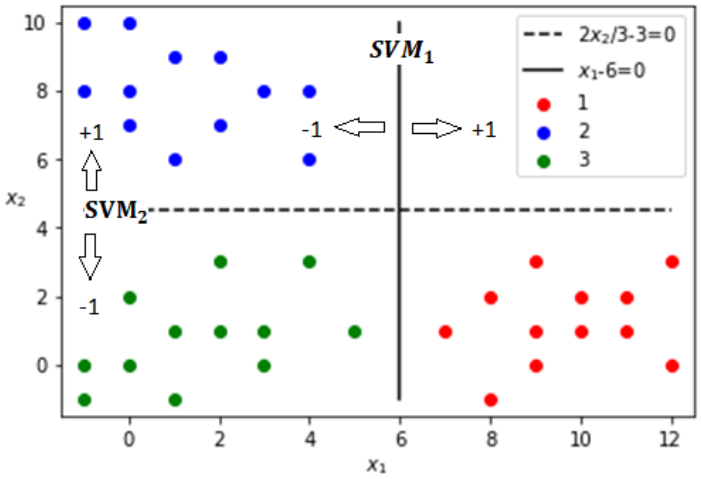

Figure 16: multi-class SVM for the data defined in Q3.

Hence, as we defined before, if the first SVM, $\mathbf{SVM_1}$, outputs $+1$, the multi-class classifier has already done its job labelling the data instance as class 1 (red). On the other hand, if this SVM outputs -1, the data instance should pass through a second SVM, $\mathbf{SVM_2}$, to determine its class.

If this SVM returns $+1$, the class label is 2 (blue); if not, the output would be -1, meaning it belongs to class 3 (green).

Q7. Produce a test dataset by averaging the samples for each row in *Table 2*, i.e., (sample of class 1 + sample of class 2 + sample of class 3)/3. Summarise the results in the form of *Table 3*, where N is the number of SVMs in your design and "Classification" is the class determined by your multi-class classifier. Explain how to get the "Classification" column using one test sample. Show the calculations for one or two samples to demonstrate how to get the contents in the table. (20 Marks)

This last question demands us to test the classifier. To accomplish this, we have to create a new test set by averaging every sample in each class for each row in *Table 3*, so we obtain 12 new samples. We will reuse *Table 3* to obtain the new samples:

INSTANCE	SAMPLE OF CLASS 1	SAMPLE OF CLASS 2	SAMPLE OF CLASS 3	TEST SAMPLE
1	(7, 1)	(-1, 8)	(-1, -1)	(1.667, 2.667)
2	(8, -1)	(-1, 10)	(-1, 0)	(2, 3)
3	(8, 2)	(0, 7)	(0, 0)	(2.667, 3)
4	(9, 0)	(0, 8)	(0, 2)	(3, 3.333)
5	(9, 1)	(0, 10)	(1, -1)	(3.333, 3.333)
6	(9, 3)	(1, 6)	(1, 1)	(3.667, 3.333)
7	(10, 1)	(1, 9)	(2, 1)	(4.333, 3.667)
8	(10, 2)	(2, 7)	(2, 3)	(4.667, 4)
9	(11, 1)	(2, 9)	(3, 0)	(5.333, 3.333)
10	(11, 2)	(3, 8)	(3, 1)	(5.667, 3.667)
11	(12, 0)	(4, 6)	(4, 3)	(6.667, 3)
12	(12, 3)	(4, 8)	(5, 1)	(7, 4)

Table 6: samples of each class and samples of new test set (average of the 3 classes per row)

We are going to represent these data instances in a 2-dimensional plot to have a better sense of how they are distributed in space.

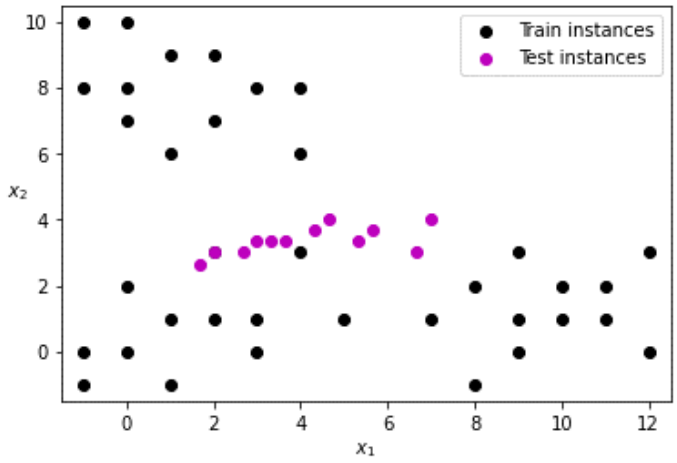

Figure 17: test dataset represented in space.

We could classify these instances directly by applying the two lines that define each SVM. They would belong to the class of the quadrant where they fall in to. This would look like the 2D plot shown below:

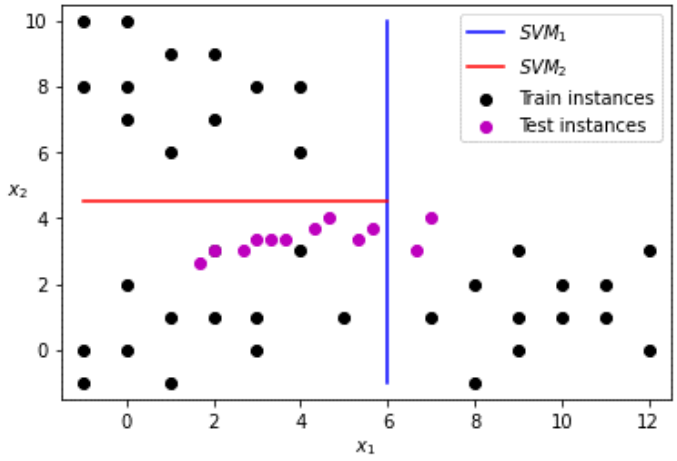

Figure 18: test dataset represented in 2D space.

As we can see, there are a lot of data instances that are very close to the line separator. They would fall in the margin. So we can see it in a more visual manner, I will plot the lines that limit the margin too:

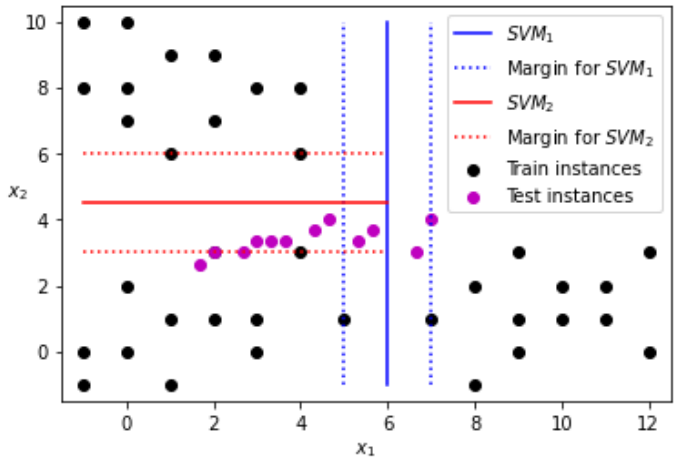

Figure 19: test dataset represented in space.

As we can see, from the 12 test instances, 2 of them belong to class 1. Specifically, the two test data samples with the x_1 coordinate above 6, where the first SVM lies. The remaining test data samples belong to class 3 because they have an x_2 coordinate below 4.5, and obviously their coordinate x_1 is below 6 too. The only difference would be that these points that rely within the margin will have an output between $(-1, 1)$, while the points that actually are lying between the margin boundaries, will have an output that will be ± 1.

If these points would have been included in the initial training set, obviously the line separator would have other coordinates. Basically because the support vectors that would define the hyperplane, or straight line in our case, would have been different.

Once we have analysed these test data instances, we are in position of filling the table shown in the question description. We will fill it completely and then explain the three possible cases: when the sample is outside the margin, when it is on the margin boundary, and when it is between the margin boundary and the straight line.

The table from the question 7 description would look like the following:

#	TEST SAMPLE	OUTPUT SVM_1	OUTPUT SVM_2	CLASSIFICATION
1	(1.667, 2.667)	-1	-1	Class **3**
2	(2, 3)	-1	-1	Class **3**
3	(2.667, 3)	-1	-1	Class **3**
4	(3, 3.333)	-1	-1	Class **3**
5	(3.333, 3.333)	-1	-1	Class **3**
6	(3.667, 3.333)	-1	-1	Class **3**
7	(4.333, 3.667)	-1	-1	Class **3**
8	(4.667, 4)	-1	-1	Class **3**
9	(5.333, 3.333)	-1	-1	Class **3**
10	(5.667, 3.667)	-1	-1	Class **3**
11	(6.667, 3)	1	N/A	Class **1**
12	(7, 4)	1	N/A	Class **1**

Table 7: representation of the classification of the test dataset.

As we can said before, most of the samples are classified as class 3. This is understandable because these samples are in between the other two samples, so it is normal that they are around that area.

Now, we will explain the three possible cases stated before independently. Before we must remember how our multi-class SVM works:

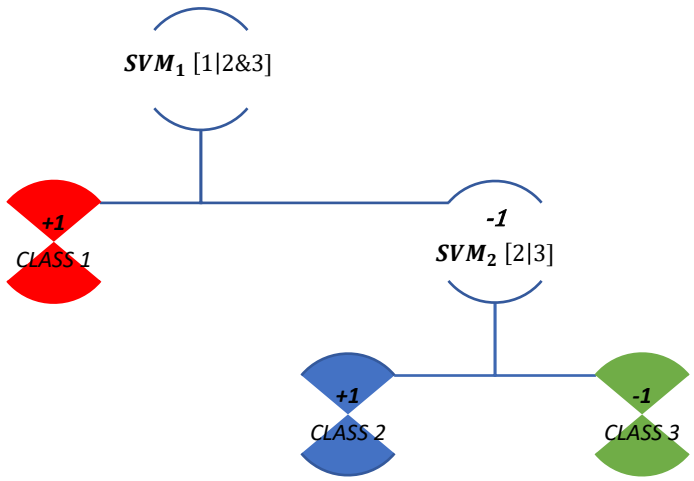

Figure 20: block diagram for a binary decision tree approach at the SVM level

As we can see, every data instance has to pass through $\boldsymbol{SVM_1}$, this will determine if it belongs to class 1. In case it returns -1, meaning it is not from that class, it should pass through the second SVM, $\boldsymbol{SVM_2}$. The output of this SVM will determine whether it belongs to class 2 or 3.

1. <u>The data sample is outside of the margin</u>: this example could work for the first test sample (1.667, 2.667). First we will pass it through the first SVM using the hard classifier:

Hard classifier: $f(\boldsymbol{x}) = sgn(\boldsymbol{w^T x} + w_0)$, where $sgn(z) = \begin{cases} -1 \; if \; z < 0 \\ +1 \; if \; z \geq 0 \end{cases}$

In our case, $\boldsymbol{x_1} = \begin{bmatrix} \mathbf{1.667} \\ \mathbf{2.667} \end{bmatrix}$, and for the first SVM: $\boldsymbol{w} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \end{bmatrix}$ and $w_0 = -6$. Therefore:

$$f(\boldsymbol{x_1}) = sgn\left(\boldsymbol{w^T x_1} + w_0\right) = \boldsymbol{sgn}\left([\mathbf{1} \quad \mathbf{0}]\begin{bmatrix}\mathbf{1.667}\\ \mathbf{2.667}\end{bmatrix} + (-6)\right) = \boldsymbol{sgn}(\mathbf{-4.333}) = -1$$

Just as a remark, the absolute value in the $\boldsymbol{sgn}(\cdot)$ function is the distance to the straight-line separator. This makes total sense as $6 - 1.667 = 4.333$.

As we can see from the last output, this data instance does not belong to class 1. To determine to which class it truly belongs, we have to pass it through the second SVM, $\boldsymbol{SVM_2}$. We repeat the same process as below, now passing the parameters of the second and last SVM:

For the second SVM: $\boldsymbol{w} = \begin{bmatrix}\mathbf{1}\\ 2/3\end{bmatrix}$ and $w_0 = -3$. Therefore:

$$f(\boldsymbol{x_1}) = sgn\left(\boldsymbol{w^T x_1} + w_0\right) = \boldsymbol{sgn}\left([\mathbf{0} \quad 2/3]\begin{bmatrix}\mathbf{1.667}\\ \mathbf{2.667}\end{bmatrix} + (-3)\right) = \boldsymbol{sgn}(\mathbf{-1.222}) = -1$$

From the output of $\boldsymbol{SVM_2}$ we finally know that this data sample belongs to the class 3, the green class. The value of the $\boldsymbol{sgn}(\cdot)$ function is the distance to the straight line. However, in this case it is scaled different because of the margin boundaries. This SVM has a larger margin, therefore it does not correspond to the actual coordinates. Anyway, we can see that this distance is much lower than the distance from the first SVM, something that can be checked in *Figure 19*.

2. <u>The data sample lies on the margin</u>: this example could work for the last test sample (7, 4). As it lies on the margin, it would have been a support vector if it would have been part of the training set. Hence, the output of the $\boldsymbol{sgn}(\cdot)$ function of the hard classifier should be exactly $+1$, because it belongs to this class. First we will pass it through the first SVM using the hard classifier:

$$f(\boldsymbol{x_{12}}) = sgn\left(\boldsymbol{w^T x_{12}} + w_0\right) = \boldsymbol{sgn}\left([\mathbf{1} \quad \mathbf{0}]\begin{bmatrix}\mathbf{7}\\ \mathbf{4}\end{bmatrix} + (-6)\right) = \boldsymbol{sgn}(\mathbf{1}) = +1$$

As we imagined, the output of the $\boldsymbol{sgn}(\cdot)$ function is $+1$, so our test sample belongs to the class 1, the red class.

3. <u>The data sample lies within the margin</u>: this example could work for the test sample number 8, (4.667, 4). This data instance should return a negative value for the first SVM, and for the second SVM too. However, the output of the second SVM should be between $(-1, 0)$, as it is within the line separator and the margin boundary. First, lets pass this data point through $\boldsymbol{SVM_1}$:

$$f(\boldsymbol{x_8}) = sgn\left(\boldsymbol{w^T x_8} + w_0\right) = \boldsymbol{sgn}\left([\mathbf{1} \quad \mathbf{0}]\begin{bmatrix}\mathbf{4.667}\\ \mathbf{4}\end{bmatrix} + (-6)\right) = \boldsymbol{sgn}(\mathbf{-1.333}) = -1$$

As we predicted, this SVM returns a negative value. However, its absolute value is slightly higher than 1, meaning that it is out of the margin boundaries but very close. This can be verified in *Figure 19*.

Once we know it does not belong to class 1, we have to see if it belongs to class 2 or 3. This will be done by passing this data instance through the second SVM, $\boldsymbol{SVM_2}$:

$$f(\boldsymbol{x_8}) = sgn(\boldsymbol{w^T x_8} + w_0) = \boldsymbol{sgn}\left(\begin{bmatrix}\mathbf{0} & 2/3\end{bmatrix}\begin{bmatrix}\mathbf{4.667}\\ \mathbf{4}\end{bmatrix} + (-3)\right) = \boldsymbol{sgn}(-\mathbf{0.333}) = -1$$

We can see that the hard classifier outputs a -1, meaning it belongs to class 3, the green class. As we supposed, the output of the $\boldsymbol{sgn}(\cdot)$ function is between $(-1, 0)$, and actually very close to 0. The closer to 0, the harder it is to classify. The larger the distance to the line separator, the easier it is to classify.

In the next plot we will be able to see these three points separated from the rest, and with the training data samples in their original colour:

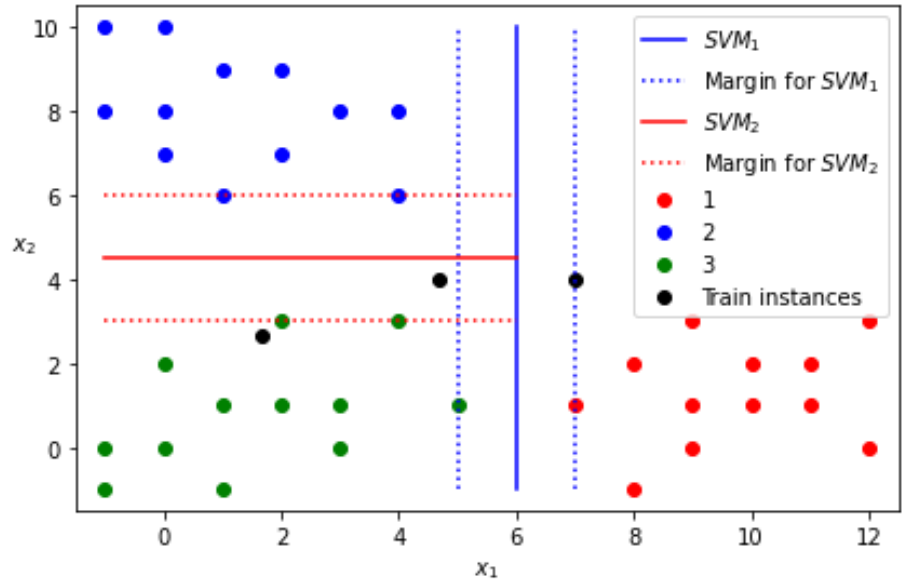

Figure 21: test samples examples and their location in the 2-dimensional space