Statistical Techniques in Robotics (16-831, S22) Lecture #10 (Monday, February 21)

Support Vector Maxines

Lecturer: Kris Kitani *Scribes: Rishabh Pahuja, Sarthak Shetty*

1 Review

In the previous lectures, we reviewed the mechanics of perceptrons and winnow algorithms. We then looked at online convex optimization with the application of follow the leader, follow the regularized leader and the relation with online mirror descent.

1.1 Gradient Descent

We briefly discussed gradient descent in the previous lecture. Gradient descent is a technique for arriving at the minimum of some differentiable convex function. We essentially move in the direction opposite to the gradient at the given point in the function, to eventually arrive at the minimum of the function.

Gradient descent is an algorithm that is used to find the minimum value of a differentiable function. Gradient descent follows Algorithm 1:

Algorithm 1 Gradient Descent

1: $w^{(0)} = 0$
2: **for** $t = 1, \cdots, T$ **do**
3: **Compute**$(\nabla f(w^{t-1}))$
4: $w^t = w^{t-1} - \eta \nabla f(w^{t-1})$
5: **end for**

The '-' sign is important here. It indicates that we change the value of w in the direction opposite to the gradient of the function. Therefore, to find the minimum of a given function we keep moving in the direction decreasing gradient, until we converge to some function mimima.

1.2 Online Gradient Descent

Online Mirror Descent can be used to solve optimization problems that comprise of quadratic regularization and a linear loss function. In the case of Online Gradient Descent, we can write the regularization function as:

$$\psi(\mathbf{w}) = \frac{1}{2\eta}||w||_2^2 \tag{1}$$

And a linear loss function defined as:

$$f(\mathbf{w}) = \langle w, \theta \rangle \tag{2}$$

Here, θ is the parameter for the dual space. We then solved for the optimal, θ, and find the following expression:

$$w_n = \eta\theta \tag{3}$$

From Eq. 3, we see that the mirror function for Online Gradient Descent, represented as $g(\theta) = \eta\theta$.

1.3 Online Normalized Exponentiated Gradient Descent

We also briefly discussed Online Normalized Exponentiated Gradient Descent. The regularization function for ONEGD is represented as:

$$\psi(\mathbf{w}) = \sum_{k=1}^{K} w_k \; log \; w_k \; w \in \mathbb{S}^k \tag{4}$$

And the loss function remains the same as:

$$f(\mathbf{w}) = \langle w, \theta \rangle \tag{5}$$

We then find the mirror function for ONEGD as:

$$g(\theta) = \frac{\eta\theta}{\sum_{n'} exp(\eta\theta_{n'})} \tag{6}$$

Therefore, we saw that for ONEGD we have a linear loss, as seen in Eq. 5, with entropy regularization as seen in Eq. 6.

1.4 Online Learning vs. Supervised Learning

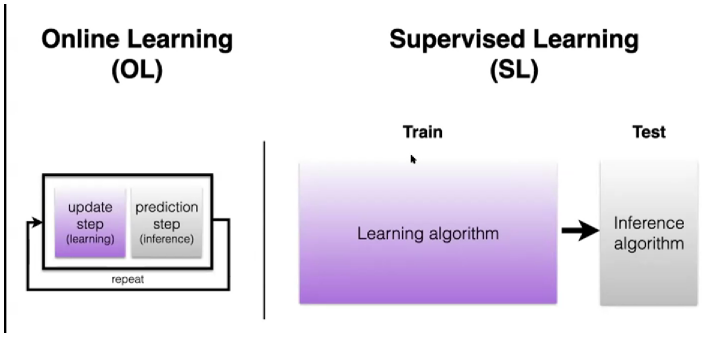

Figure 1: Comparison of Online Learning (OL) with Supervised Learning (SL)

In online learning, we have an update step followed by the prediction step. This pair of actions are repeated during the operation of the model. This relationship can be visualized in Fig. 1.

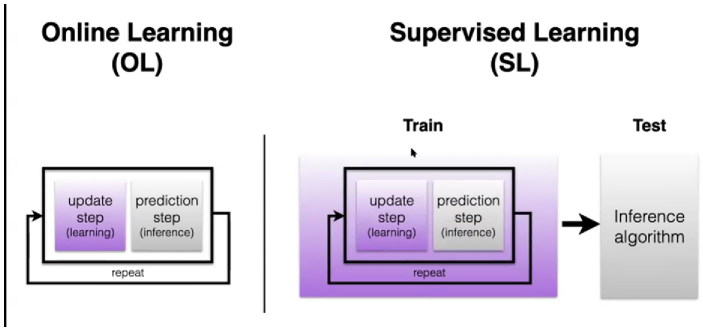

Figure 2: Online Learning can be represented within a Supervised Learning model

Whereas, in supervised learning problems, we train the model with the available data and then utilize the trained algorithm for inference. Often, we can include online learning algorithms within these supervised learning models, as shown in Fig. 2.

Therefore, **most supervised learning algorithms can be solved used online learning models**.

2 Online Support Vector Maxines

2.1 Geometric Intuition

2.1.1 Hyperplanes in 2D

We can represent hyperplanes in 2D using different parameterizations. One of the parameterizations that we can use is:

$$w_1 x_1 + w_2 x_2 + b = 0 \tag{7}$$

Therefore, to fully represent the line in 2D we need a total of 3 parameters, w_1, w_2 and b. Here, $w \in R^2$

A shorthand for Eq. 7, can also be:

$$w \cdot x = 0 \tag{8}$$

Here, in Eq. 8, $w \in R^3$.

An important property of such hyperplanes, is that any normalization can be chosen for w, such that the following equation holds:

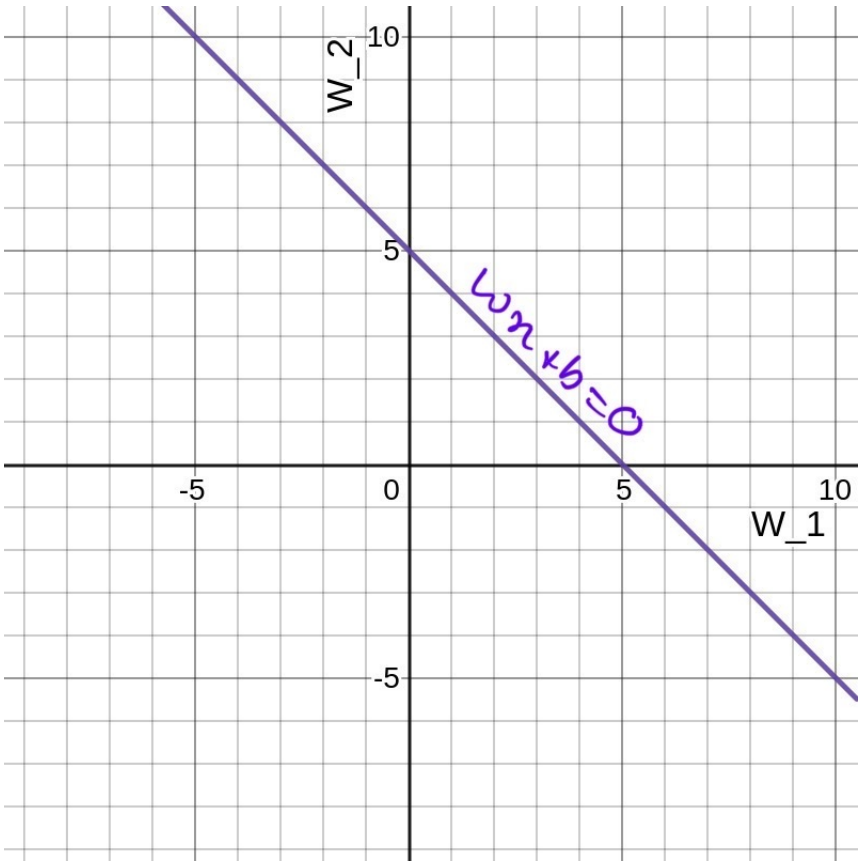

Figure 3: Line parameterization in 2D

$$\lambda(w_1 x_1 + w_2 x_2 + b) = 0 \tag{9}$$

This implies that w can be normalized by any scalar value, λ. Therefore, the line represented in Fig. 3, remains the same, even though w is being normalized by different values.

Using Eq. 8, we can find the perpendicular distance of the line visualized in Fig. 3 from the origin through the following steps:

We divide Eq. 8 with $||w||$, such that:

$$\frac{w}{||w||} \cdot x + \frac{b}{||w||} = 0 \tag{10}$$

We can rearrange this equation and compare it to the normal form of a line:

$$\frac{w}{||w||} \cdot x + 0 = -\frac{b}{||w||} \tag{11}$$

$$x \cos\theta + y \sin\theta = \rho \tag{12}$$

Eq. 12. is equivalent to Eq. 11 holds only if θ is equal to $0°$. This implies that the distance to the origin ρ is equal to $\frac{b}{||w||}$.

Consider the now we have another line, offset from the first line that we saw in Fig. 3. We visualize the pair of lines as follows:

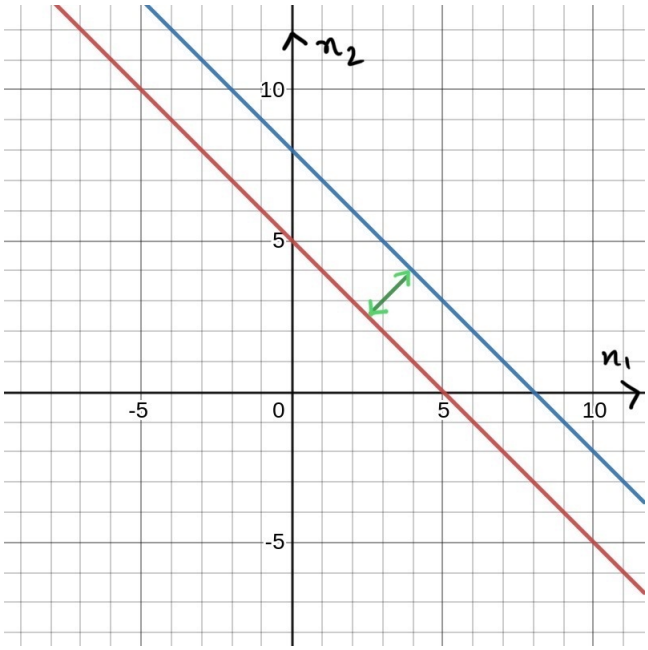

Figure 4: A second line parallel to the first line, offset by some distance

We can now find the distance between the pair of lines as follows:

The second line can be described as:

$$w \cdot + b = -1 \tag{13}$$

It's distance from the origin (following the same schema as Eq. 11 and Eq. 12) is given as:

$$\rho_b = \frac{b+1}{||w||} \tag{14}$$

Therefore, the distance between the two lines is:

$$\rho_d = \frac{1}{||w||} \tag{15}$$

Similarly, for a hyperplane in 3D the only change that we have to make to our equations is the dimensionality of w. Instead of being 2-dimensioned in the previous case, w will now be a 3-dimensioned vector.

Consider we have 2 planes in a 3-dimensioned space as shown in Fig. 5:

Since we know the equation of the planes, and their distances to the origin through our previously, derived equations in Eq. 12 and 11, we find the distance between the outermost planes as:

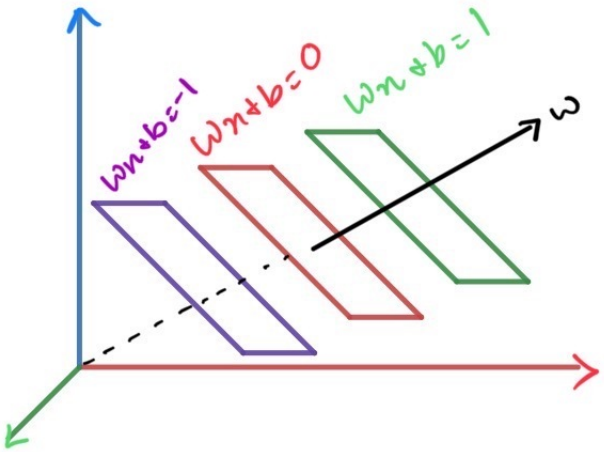

Figure 5: Planes in 3D, separated by some distance

$$\rho_{OD} = \frac{2}{||w||} \tag{16}$$

3 Support Vector Machines (SVM)

3.1 Introduction

Consider the binary classification problem, where we have two clusters of data as shown here:

Our objective is to come up with a hyperplane that best separates this data, such that we have perfect classification between these two classes. Even though we can have any number of hyperplanes, w that split the data reasonably, intuitively the w that's farthest away from the interior-most points of these two clusters will perfectly classify the data into these two clusters.

Graphically this hypothetical w can be represented as:

This hyperplane, that has the "maximum margin" to the innermost points, is also the plane, w, most likely to be the most stable to perturbations in the data. The innermost points of each cluster that form the definition of the margin to the given hyperplane w are also called "support vectors", and therefore, these planes are also called "Support Vector Machines" or SVMs.

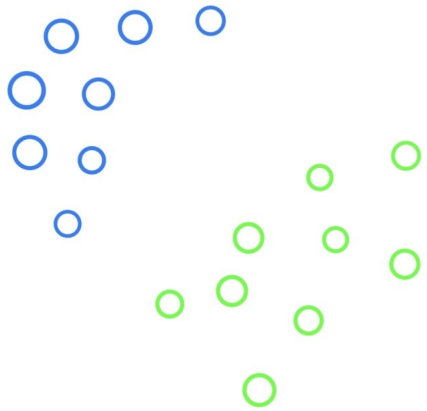

Figure 6: Two clusters of data that have to be separated with a suitable hyperplane w

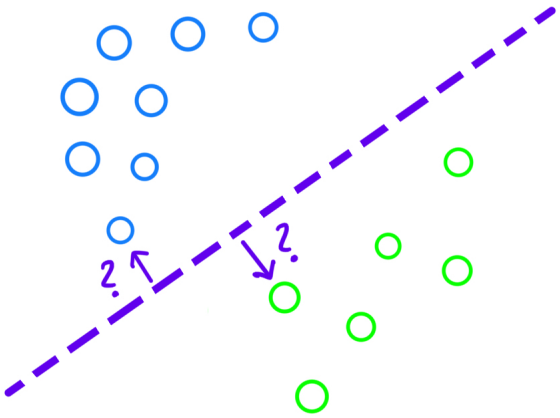

Figure 7: Hyperplane, w, that has the maximum distance from the innermost points of both clusters

3.2 Objective of Hyperplanes

The objective of the SVM is to find a hyperplane, w, such that the hyperplanes that define the margin of two clusters is maximised. This objective can be visualized in Fig. 8:

From Eq. 15, we see that in order to maximise the distance between the two hyperplanes, we have to maximise the term $\frac{2}{||w||}$, i.e,

$$\rho_{max} = \max_{w} \frac{2}{||w||} \tag{17}$$

Referencing Fig. 8, the objective of this SVM is to correctly classify any given data point that it encounters into the correct cluster. Let's look at what that would mean mathematically:

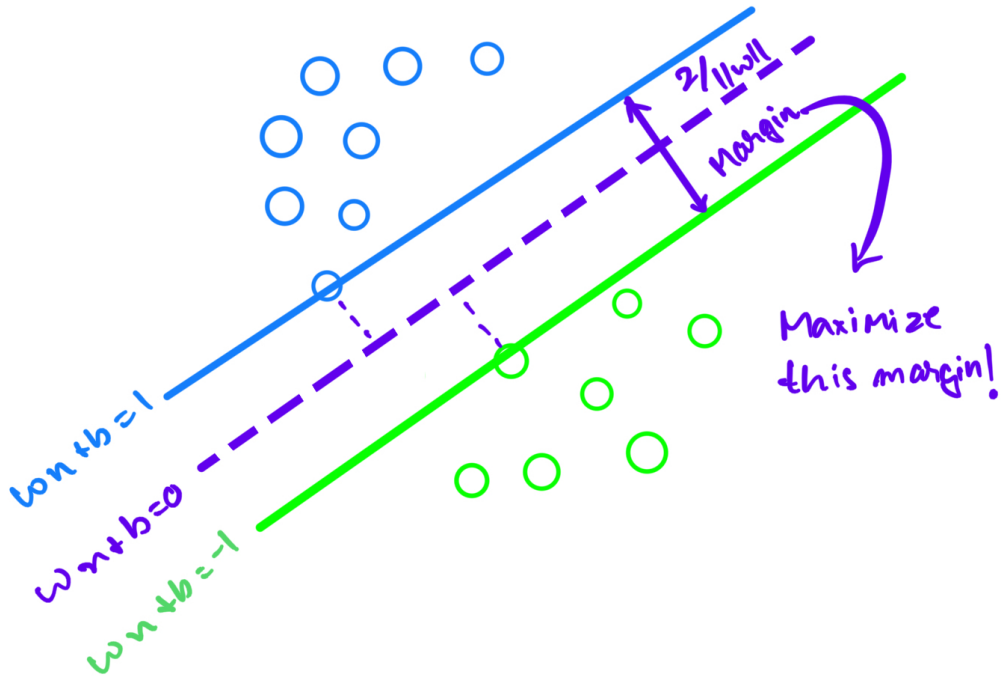

Figure 8: The objective of SVM is to find the hyperplane, w, such that the margin between the clusters is maximised

$$w \cdot x + b = \begin{cases} +1, & \text{if } y_i = +1 \\ -1, & \text{if } y_i = -1 \forall\ i \in [1, 2, 3 \dots N] \end{cases} \tag{18}$$

Eq. 18 would imply that the SVM would always correctly label the given x_i into one of the clusters. This is the expected performance of the given SVM.

In order to maximize Eq. 18, we can minimize the following equation instead:

$$\min_{w} ||w||^2 \tag{19}$$

Given the constraint,

$$y_i(w \cdot x_i + b) \geq 1, \forall i \in [1, 2, \dots N] \tag{20}$$

In Eq. 19, we square w in order to introduce a greater penalty on w. The minimum value of $||w||$ will be the same as the minimum value of $||w||^2$, which justifies this substitution. In Eq. 19, since we're performing binary classification (i.e, predicting one of two labels $\hat{y}$ in $[-1, 1]$), we can multiply the output of the classification of the SVM with the label, to arrive at the condition within the equation.

Therefore, we can formalize the working of an SVM as:

Objective Function:

$$\min_{w} ||w^2|| \tag{21}$$

Constraints

The objective function in Eq. 21 is subject to hard constraints:

$$y_i(w \cdot x_i + b) \geq 1 \text{For i} \in [1, 2 \dots N] \tag{22}$$

This problem is known as a **Convex Quadratic Problem** (QP) with linear constraints. Such problems might have unique solutions, but may not satisfy the constraints of the data (i.e, the data is not separable or is noisy).

Such SVMs are called **"Hard SVMs"**, since we make the assumption that all constraints can be satisfied. This may not be the case at all times, as we see below.

3.2.1 Soft SVM

The above discussion holds true when the data is linearly separable. However, when working with real data, it may not be linearly separable and this is where we use soft margins to separate the data.

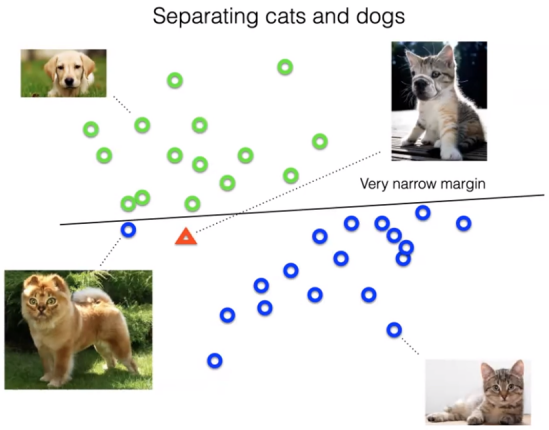

Figure 9: Performance of Hard SVM of linearly in-separable data

Considering Fig. 9, we can see that the data is not linearly separable. This is because, some of the dogs look slightly like cats and vice versa. As we can see from the data,there is a very small margin between the labels of cats and dogs and in attempts to rightly classify each image, the whole line will be shifted for very rare cases. Therefore, we should allow for some miss-classification if we can get more robust classification by coming up with a hyperplane that has a higher margin, at the cost of some miss-classification. This can be classified as classifying the data 'softly'.

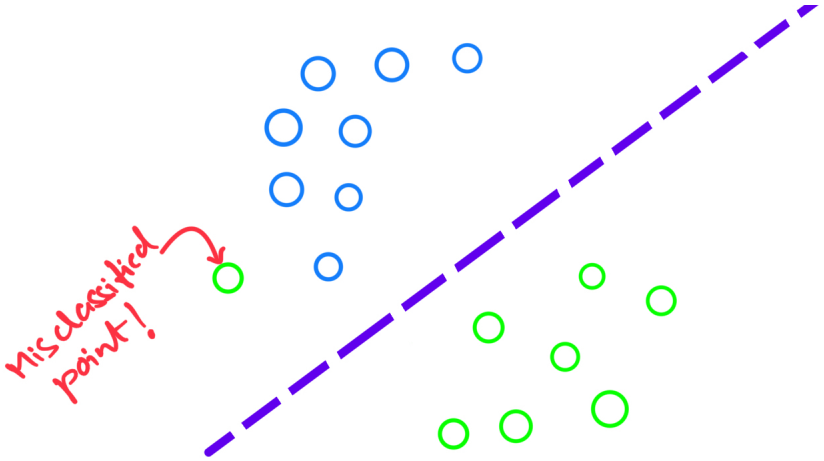

Figure 10: Performance of Soft SVM of linearly in-separable data

Fig. 10 shows, implementation of some soft SVM at the cost of some miss-classifications. It can be seen that the hyperplane has higher margin and thus, is a better classifier.

3.2.2 Mathematical interpretation of soft margin

The 'slack' in classification can be increased by adding a slack variable to the expression we saw earlier for hard SVM classification. Thus, the expression now becomes:

$$y_i(w^T \cdot x_i + b) \geq 1 - \xi_i \tag{23}$$

Here ξ introduced will decrease the threshold value and some of the values might be miss classified. The value of ξ should ideally be as small as possible because smaller value indicates that the points are on the right side of the hyperplane. The variable ξ is called as the slack variable and geometrically represents the distance of the miss-classified element from the margin.

The object function now becomes:

$$\min_{w,\xi} ||w||^2 + C \sum \xi_i \tag{24}$$

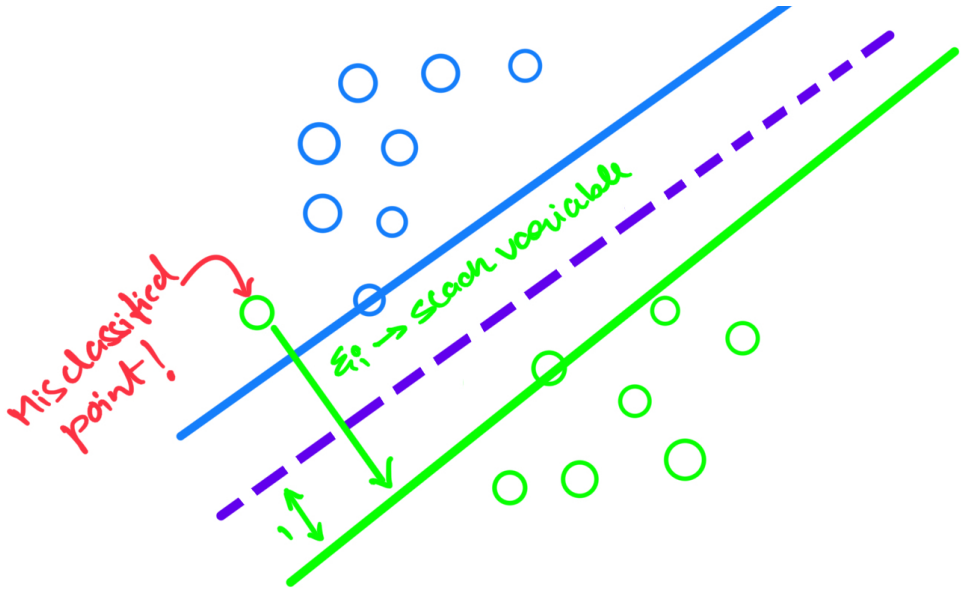

Figure 11: Geometric interpretation of slack variables in soft SVM

C here is a regularization parameter. When the value of C is very small, that means that the constraints will be ignored, there will be more miss-classifications and the margins will be higher. And if the value of C is large, constraints won't be ignored, the margin will be small and hence, there will be lesser classifications.

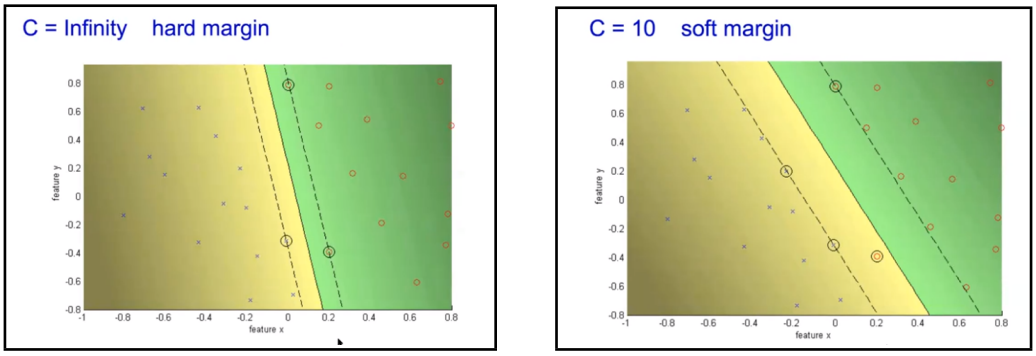

Figure 12: Comparison between Hard Margin and Soft Margin

3.2.3 Solution for Soft SVM

The new objective function $\min_{\mathbf{w},\xi} ||\mathbf{w}||^{\mathbf{2}} + \mathbf{C}\sum \xi_{\mathbf{i}}$ when subjected to $\mathbf{y_i}(\mathbf{w^T x_i} + \mathbf{b} \geq \mathbf{1} - \xi_{\mathbf{i}})$ can be solved using Lagrange's method:

Thus, the equation becomes:

$$\min_w \frac{\lambda}{2}||w||^2 + \frac{1}{N}\Sigma_{i=1}^{N}(1 - y_i \cdot w^T x_i) \tag{25}$$

Where, λ is Lagrange's multiplier and ξ becomes:

$$\xi \geq 1 - y_i(w \cdot x_i) \tag{26}$$

We can see that when the predictions are correct, the value of ξ can vary from ∞ to 1 and when the predictions are wrong, the values can range from 1 to ∞. However, we should not be concerned for all the values of ξ but only for those values such that the margins become less or the mistakes are too high. So to bound the value of ξ, we use hinge loss such that the new equation becomes:

$$\min_w \frac{\lambda}{2}||w||^2 + \frac{1}{N}\Sigma_{i=1}^{N} \max(0, 1 - y_i w^T \cdot x_i) \tag{27}$$

One thing to note here is that the function is convex here but is not differentiable throughout because of the hinge. Therefore, to solve this function, we introduce sub-gradients. The necessity to introduce sub-gradient arises when a function is not differentiable throughout its input values. In the second sub-figure of 13, the various lines lower bounding the convex function at the hinge are called the sub-gradients.

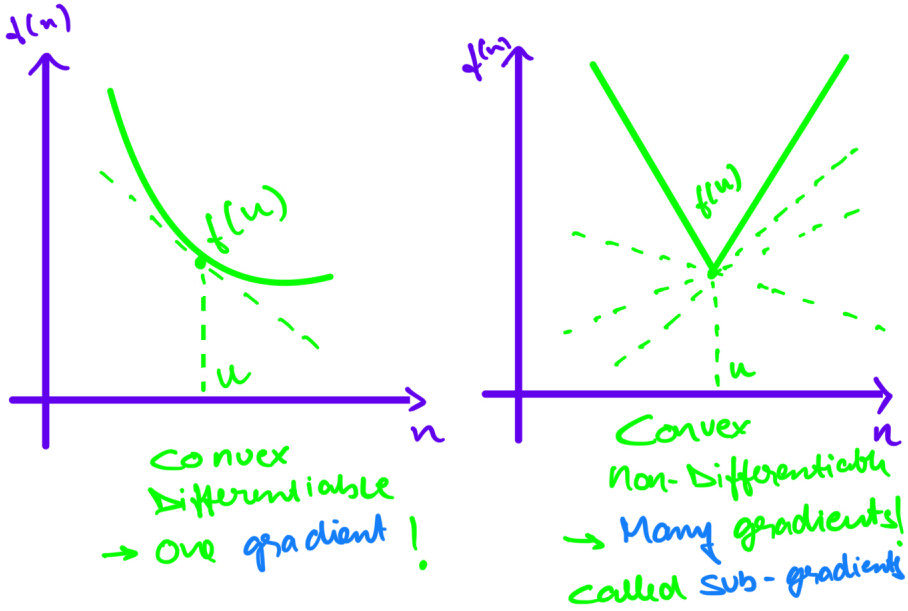

Figure 13: Gradient vs Sub-Gradient

From all the possible sub-gradients, we can define two gradients for the function such that it can be differentiated:

$$z_{[m]} = \begin{cases} 0 & \text{if, } \; y_m w^T x_m \geq 1 \\ -y_m x_m & \text{otherwise} \end{cases} \tag{28}$$

So the algorithm becomes:

Algorithm 2 Soft SVM

1: $\theta^{(0)} = 0$
2: **for** $t = 1, \cdots, T$ **do**
3: $\quad y_d, x_d \sim D$
4: $\quad \theta^t = \theta^{t-1} + y_d x_d \cdot 1[y_d(w^t x_d) < 1]$
5: $\quad \hat{y}^{(t)} = h(\mathbf{x}^{(t)})$
6: $\quad w^{t+1} \leftarrow \dfrac{1}{\lambda(t+1)} \cdot \theta^t$
7: **end for**

In line 5, $y_d x_d$ is the sub-gradient and the indicator function is the hinge condition, such that the θ^t will only be updated if there is a wrong prediction or if the label is on the wrong side of the hyperplane.

This algorithm is very close to online perceptron algorithm. However something to note here is that in online perceptron algorithm, indicator function is $1 \cdot [y_d(w^(t).x_d) < 0]$ implying that the perceptron algorithm considers if the prediction is right or wrong and does not consider correctness or in-correctness of the prediction. SVM on the other hand, considers both correctness and the in-correctness of predictions.

4 Conclusion

In this lecture, we looked at the underlying mechanics of how Support Vector Maxines work (SVMs). We also learnt the differences between Hard and Soft SVMs, and their performance when presented with linearly separable and in-separable data. We also looked at the need for sub-gradients when dealing with convex, but non-differentiable functions.

Note: We referred to the previous years' scribe notes while preparing this document [1].

5 Appendix

References

[1] M. Paritosh and S. Tanay. Lecture 10a. https://github.com/Xingyu-Lin/16831-spring-2021/tree/master/src/lecture_10_A, 2021.