CS 217: Artificial Intelligence and Machine Learning **Jan-Apr 2024**

Lecture 15: Support Vector Machine

Lecturer: Swaprava Nath *Scribe(s): SG29 & SG30*

Disclaimer: *These notes aggregate content from several texts and have not been subjected to the usual scrutiny deserved by formal publications. If you find errors, please bring to the notice of the Instructor.*

15.1 SVM: Support Vector Machines

In this lecture, we will talk about another linear classifier : SVMs (Support Vector Machines). Our dataset will be of the form:

$$D = \{(x_i, y_i)\}_{i=1}^{N} \text{ where } x_i \in \mathbb{R}^d$$
$$y \in \{-1, 1\}$$

Technically, the primary objective of the SVM algorithm is to identify a hyperplane that distinctively segregates the data points of different classes.

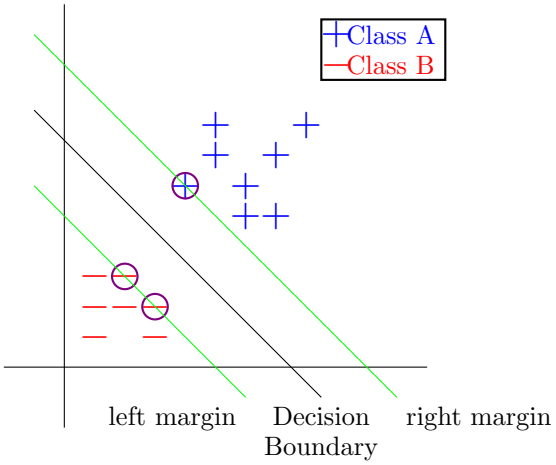


Figure 1

Now we define a few terms related to SVMs :

- **Positive Hyperplane:** The hyperplane that touches the points of the positive class.
- **Negative Hyperplane:** The hyperplane that touches the points of the negative class.
- **Margin:** The distance between the hyperplane and the observations closest to the hyperplane.
- **Support Vectors:** Those data points that touch the positive and negative hyperplanes.

In SVM large margin is considered a good margin, therefore, the SVM algorithm tries to maximize the margin between the support vectors.

Let us consider a dataset with two feature : BMI and BP.

Suppose we have the following classification and the three hyperplanes : L1(green), L2(blue) and L3 (red). Although all hyperplanes can separate data well, L1 and L3 have less tolerance for the data point distribution, the best fit hyperplane would be L2 as its margin is maximum.

The intuition behind keeping the margin maximum is that model with maximum margin would be more generalizable on unseen data. A classifier with a large margin makes no low certainty classification decisions. This gives you a classification safety margin: a slight error in measurement will not cause a misclassification.

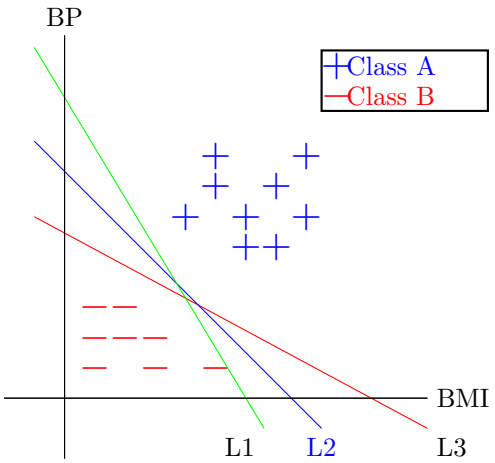


Figure 2

We choose two parameters $\mathbf{w}$ and b, such that

- For all points on the hyperplane: $\mathbf{w}^\intercal x_i + b = 0$
- For all points above the hyperplane: $\mathbf{w}^\intercal x_i + b > 0$
- For all points below the hyperplane: $\mathbf{w}^\intercal x_i + b < 0$

15.2 SVM Variants on the basis of Constraints Rigidity

15.2.1 Hard Margin SVM

The Hard Margin SVM is a type of SVM that does not allow any misclassifications. The data set is linearly separable if there exists a hyperplane that separates the positive and negative classes. Based on the training dataset there should be no point that lies between the positive and negative hyperplanes. All data points are linearly separable in the feature space.

Consider the following figure:

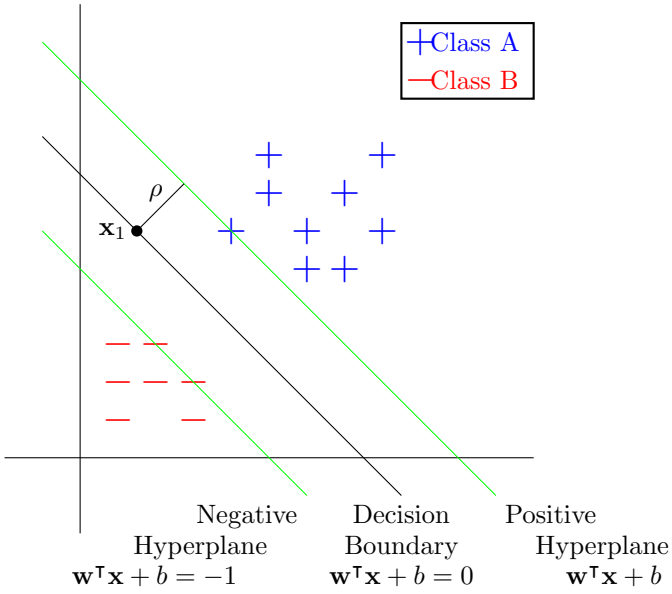


Figure 3

Now we introduce the key variants whose choice depends on the constraints imposed on the model followed by the Data Set Points:

- **Weights Vector (w)**: The weights vector defines the orientation of the decision boundary (hyperplane) in the feature space (Analogous to the slope of a line in a linear equation).
- **Bias Term (b)**: The scalar value that adjusts the position of the decision boundary along the direction of the weights vector (Analogous to the line's intercept in a linear equation).
- **Data Set Points**: These are the instances of data that the SVM uses to learn the decision boundary. Each data point consists of a set of features and a class label. In a binary classification problem, the labels are usually +1 and -1.

The decision function is given by $f(\mathbf{x}) = \mathbf{w}^\intercal\mathbf{x} + b$, where $\mathbf{x}$ is a data point. The sign of $f(\mathbf{x})$ determines the predicted class of $\mathbf{x}$.

The equation of Positive Hyperplane is given by $\mathbf{w}^\intercal\mathbf{x} + b = 1$. The reason for chosing 1 and not any other constant is that, the distance the distance between the positive and negative hyperplanes or the margin is independent of this constant. The equation of Negative Hyperplane is given by $\mathbf{w}^\intercal\mathbf{x} + b = -1$.

The larger the margin, the more robust the model is to errors in the data. Lets denote the margin as ρ. We now go on to find the value of ρ in terms of $\mathbf{w}$ and b.

Lets pick some point on the decision boundary, say $\mathbf{x}_1$. This means that $\mathbf{w}^\intercal\mathbf{x}_1 + b = 0$.

Consider taking a step of length ρ in the direction of the weights vector. This will take us to a point $(\mathbf{x}_1 + \rho(\frac{\mathbf{w}}{||\mathbf{w}||}))$ which lies on the positive hyperplane. So the equation of the positive hyperplane at this point is given by $\mathbf{w}^\intercal(\mathbf{x}_1 + \rho(\frac{\mathbf{w}}{||\mathbf{w}||})) + b = 1$. Expanding this equation, we get $\mathbf{w}^\intercal\mathbf{x}_1 + \rho||\mathbf{w}|| + b = 1$. But we know that $\mathbf{w}^\intercal\mathbf{x}_1 + b = 0$. So the equation becomes $\rho||\mathbf{w}|| = 1 \Rightarrow \rho = \frac{1}{||\mathbf{w}||}$.

Therefore the margin is given by $\frac{1}{||\mathbf{w}||}$.

The constraints for the Hard Margin SVM are as follows:

- For all positive data points, $\mathbf{w}^\intercal \mathbf{x}_i + b \geq 1$, with $\mathbf{y}_i = +1$
- For all negative data points, $\mathbf{w}^\intercal \mathbf{x}_i + b \leq -1$, with $\mathbf{y}_i = -1$

The above constraints can be combined into a single constraint as follows:

$$\mathbf{y}_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geq 1, \forall i \in [1, n] \tag{15.1}$$

Maximizing the margin is equivalent to minimizing $\frac{1}{2}||\mathbf{w}||^2$.

The objective function for the Hard Margin SVM is therefore given by:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 \text{ subject to } \mathbf{y}_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geq 1, \forall i \in [1, n] \tag{15.2}$$

15.2.2 Soft Margin SVM

There are a few limitations of Hard Margin Classifers, beacuse of the rigidity of the constraints imposed. Some of them are :-

- **Sensitive to Outliers** :- Hard margin SVM is sensitive to outliers, as even a single misclassified point can lead to an infeasible solution. Outliers can disrupt the hyperplane and prevent proper separation of classes. For example in **Figure 4**, points such as **1** is most possibly an outlier, but the two classes cannot be seperated by a single straight line, so Hard Margin SVM cannot classify this type of dataset.
- **Margin Dependant** :- Even if the data is linearly separable, if the margin is too small, between the 2 adjacent classes, then the Hard Margin SVM may end up fitting the training data too closely, including outliers, leading to poor generalization to new data.So it's prone to overfitting. Like in **Figure 4**, points like **2** and **3** lie too close to the hyperplane, leading to the given placing of the decision boundary, which is actually over-fitted, and is prone to errors while classifying test dataset.

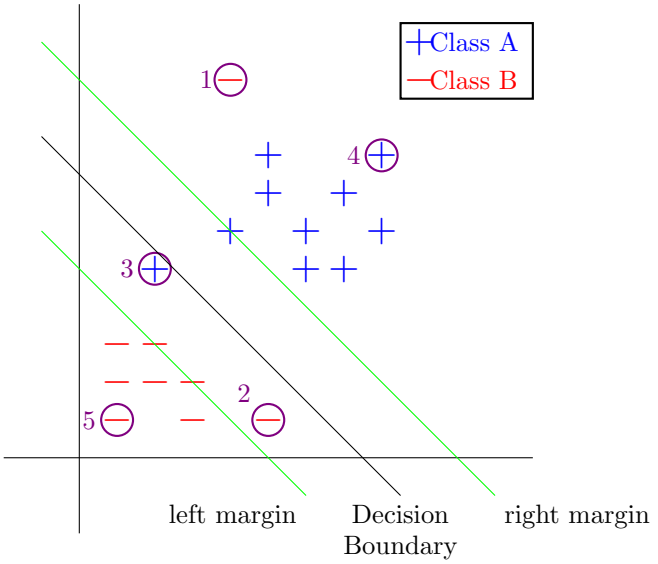


Figure 4

Thus to deal with the above problems, Soft Margin SVM comes to the picture. Like its name, its main property is to allow some misclassifcations to happen, though as low as possible. That means, we will have to deal with one more constraint that will check the degree of misclassifcations, or in simple terms, Loss/Error of the classifier. So let's now design the loss function.

We know, from the example in Hard Margin SVM, that for a correct classification, $(\mathbf{w}^T x_i + b)$ has to have the same sign as y_i for any $i \in [1, n]$. We can also write this as follows :-

$$y_i * (\mathbf{w}^T x_i + b) = \begin{cases} 1 & \text{for } ProperClassification \\ -1 & \text{for } Misclassification \end{cases} \tag{15.3}$$

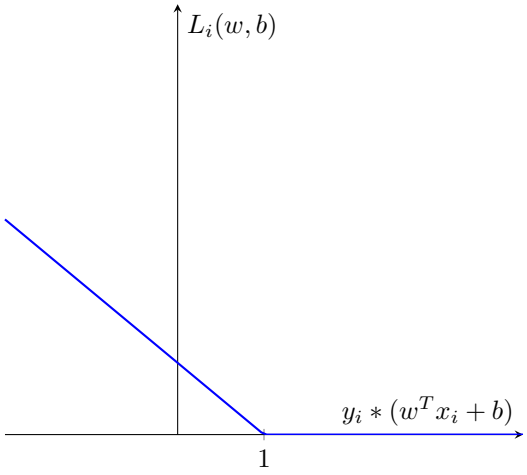
or,

$$1 - y_i * (\mathbf{w}^T x_i + b) \text{ will be } \begin{cases} \leq 0 & \text{for } ProperClassification \\ > 0 & \text{for } Misclassification \end{cases} \tag{15.4}$$

We can assume that if a proper prediction is made by the model, it should not contribute to loss/error that will be required for training it. So from equation (15.4), we can modify the loss as follows :-

$$\text{Loss function} = \max_{\mathbf{w},b} \big(0, (1 - y_i * (\mathbf{w}^T x_i + b))\big) = L_i(\mathbf{w}, b) \tag{15.5}$$

Note that, the given Loss function for this model, $L_i(\mathbf{w}, b)$ is actually a replica of hinge loss only. For a given i in $[1, n]$, its graph is like the one given below :-



Generally, the loss function is considered as the sum of the loss function of Hard Margin SVM multiplied by a hyperparameter , and the loss function of Soft Margin SVM, in order to check the norm of w, and the extent of misclassifications at the same time. So the objective function turns out to be :-

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} L_i(w,b) + \lambda \|w\|^2$$

Again this is a convex optimization problem, for the loss function of Soft Margin SVM. Here λ determines the tradeoff between misclassification and margin maximization. If λ is small, it prioritises increasing the margin, just like Hard Margin SVM, and vice versa.

Another mathematical approach to the derivation of loss function for Soft Margin SVMs can be found here in this video.

But even after allowing misclassifications to some extent, the linear SVMs still cannot properly classify the real-world classification problems, which don't have any linear hyperplane to separate 2 classes. 2 such examples of classifications are shown below :-

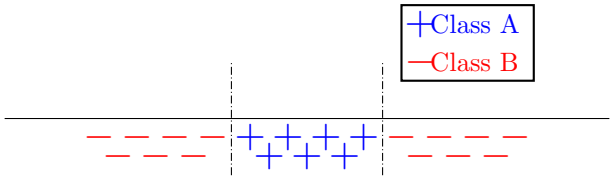


Figure 5: One Dimensional Dataset

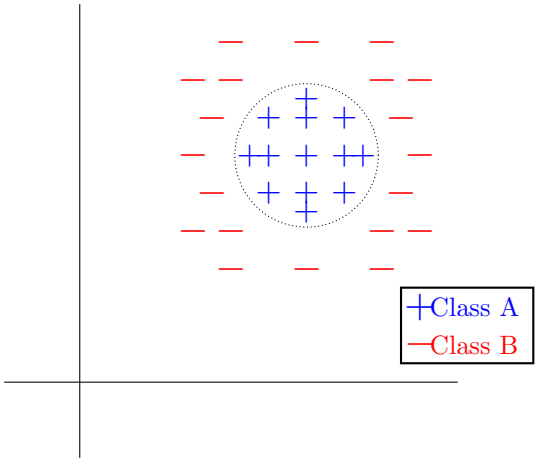


Figure 6: Two Dimensional Dataset

But again, these datasets can be solved by using SVM, but only after applying transformations to the data, which map the data from the original space into a higher dimensional feature space. The goal is that after the transformation to the higher dimensional space and adding that to the existing feature space, the classes are now linearly separable in this higher dimensional feature space. Like the **Figure 5** can be mapped to 2d plane with the help of this parabolic function $f(x) = (x-3)^2 + 1$, and thus can be linearly separated as shown below:-

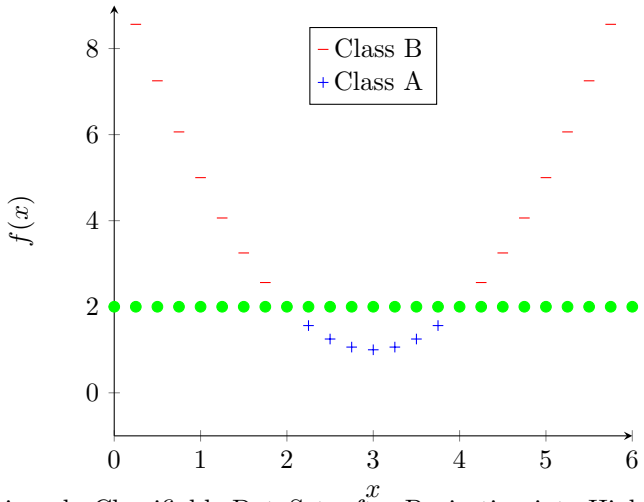


Figure 7:Linearly Classifiable DataSet, after Projection into Higher Dimension

Same mapping can also be done on the **Figure 6: Two Dimensional Dataset**, with the new Z axis being a function of x and y as $f(x,y) = 15 * ((x-3)^2 + (y-3)^2)$. Here the hyperplane $z = 15$, can separate the two classes, and thus can be the separator can be mapped on the 2d XY plane, as its equation is the intersection of plot $z = 15 * ((x-3)^2 + (y-3)^2)$ and the plane $z = 15$.

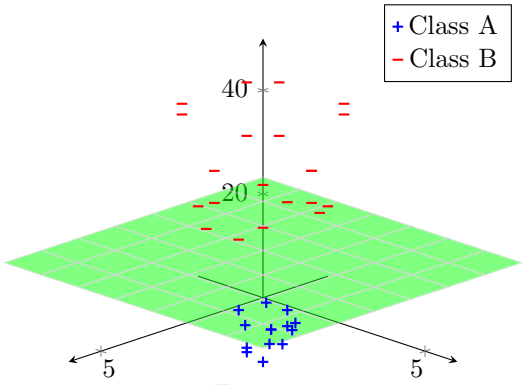


Figure 8

15.3 Background of Kernelization

Definition 15.1 (Kernelization)**.** A method to calculate higher dimension transformation in a computationally efficient manner

15.3.1 Lagrangian

In a general optimization problem (also called as Primal problem), the objective function is :

$$\begin{aligned} \min_{x} \quad & f_o(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned}$$

The domain D which contains the feasible x's is

$$D = \{x : f_i(x) \leq 0, \forall i \in [1, m], h_j(x) = 0, \forall j \in [1, p]\}$$

Its Lagrangian is defined as:

$$L(x, \lambda, \gamma) = f_o(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} \gamma_j h_j(x)$$

where λ_i is the Lagrange multiplier for $f_i(x) \leq 0$
and γ_j is the Lagrange multiplier for $h_j(x) = 0$
subject to $\lambda_i \geq 0 \ \forall i \in [1, m]$

Langrage dual function is given by:

$$g(\lambda, \gamma) = \min_{x \in D} L(x, \lambda, \gamma)$$

Lower bound property: if $\lambda \geq 0$, $g(\lambda, \gamma) \leq p*$, where p* is the optimal value of the primal.
Let's consider a feasible solution x' with $\lambda \geq 0$. According to the definition of $L(x, \lambda, \gamma)$, we have:

$$L(x, \lambda, \gamma) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} \gamma_j h_j(x)$$

Since x' is feasible, $f_i(x') \leq 0$ for $i = 1, \ldots, m$ and $h_j(x') = 0$ for $j = 1, \ldots, p$. Therefore:

$$L(x', \lambda, \gamma) = f_0(x') + \sum_{i=1}^{m} \lambda_i f_i(x') \leq f_0(x')$$

This implies that $L(x, \lambda, \gamma) \leq f_0(x)$ for all $x \in D$. Consequently, for any x in D, the value of $g(\lambda, \gamma)$ is a lower bound on $f_0(x)$.

Minimizing $f_0(x)$ over all feasible solutions x' yields p^* (primal optimal). Therefore, we have:

$$g(\lambda, \gamma) \leq p^*$$

So our problem is to maximize g(λ, γ)

$$\max_{\lambda, \gamma} g(\lambda \geq 0, \gamma)$$

$$\max_{\lambda, \gamma} [\min_{x \in D} L(x, \lambda, \gamma)]$$

15.3.2 SVM and Lagrangian

In hard margin SVMs our goal is:

$$\min \frac{1}{2} ||\mathbf{w}||^2$$

such that

$$y_i(\mathbf{w}^T x_i + b) \geq 1, \forall i = 1, 2, \ldots, n$$

We can also write it as

$$-(y_i(\mathbf{w}^T x_i + b) - 1) \leq 0, \forall i = 1, 2, \ldots, n$$

Now the Lagrangian of our problem can be written as

$$L(\mathbf{w}, b, \lambda) = \frac{1}{2} ||\mathbf{w}||^2 - \sum_i \lambda (y_i(\mathbf{w}^T x_i + b) - 1) \text{such that } \lambda \geq 0$$

Our goal now is to solve the Lagrangian dual, which in this case will be

$$g(\lambda) = \min_{\mathbf{w}, b} L(\mathbf{w}, b, \lambda)$$

Calculating derivatives,

$$\frac{dL}{d\mathbf{w}} = 0 \implies \mathbf{w} = \sum_{i=1}^{n} \lambda_i y_i x_i$$

$$\frac{dL}{db} = 0 \implies \sum_{i=1}^{n} \lambda_i y_i = 0$$

Now if we simplify both of these equations we get

$$\max_{\lambda_i \geq 0, \forall i} g(\lambda) = \sum \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j x_i^T x_j$$

If we project x_i and x_j to higher dimensions, we have to calculate the inner product of their basis functions, i.e., $\phi(x_i)^T \phi(x_j)$, instead of x_i and x_j. When solving the primal problem, we need to compute $\mathbf{w}^T \phi(x_i)$. Since $d \gg n$, where d is the dimensionality of the feature space and n is the number of samples, $nd \gg n^2$. Therefore, calculating $\phi(x_i)^T \phi(x_j)$ is a better option compared to computing $\mathbf{w}^T \phi(x_i)$.

15.3.3 Advantages of solving a dual problem over a primal problem

- It is computationally effective.Both primal and dual problem has O(n) constraints.However primal problem has O(d) variables $[\sum_i^d w_i^2]$ while dual has O(n) variables.So the time complexity of solving the primal problem is O(nd) while that of dual is O(n^2). According to our assumption $d >> n$, hence $nd >> n^2$, so solving is dual is much more computationally effective than that of primal
- It is kernel friendly. For solving the dual problem for the transformed space created by kernelization, we just have to replace $x_i^T x_j$ by $\phi(x_i)^T \phi(x_j)$ .The time complexity of solving just the dual part will remain O(n^2)

References

[1] Stephen Boyd , "Convex Optimization," *web.stanford.edu/class/ee364a/lectures/duality.pdf*