CSE/ECE-471: Statistical Methods in AI **Spring 2020**

Lecture 11: Support Vector Machines (SVM)

Prepared by: Mrityunjay (2018801001) Aman Dev (2019201095) Srikanth (2019900090)

1 Introduction

Lecture 11: Support Vector Machines (SVM)

- SVM is a *discriminative classifier* that does the task of predicting whether the class in a given input belongs to any of the two classes that it was trained to classify.
  - A *discriminative classifier* doesn't know what is a cat or a dog. In other words, it doesn't know the features of a class.
- SVM task comes under *supervised learning*.
  - A *supervised learning* task of a Machine Learning is a predictive task where given an input, an output is estimated. Further, the task can be run using both discrete (classification problem) and continuos random variable (regression problem).
- The prediction is done by identifying a *hyperplane* that divides the partition.
  - A hyperplane is a sub-set of R^n which depends on **n-1** dimensions.
  - Since we are talking of 2 classes, we can assume that the elements belonging to theses 2 classes could be laid on Euclidean space across 3-dimensional R^n
  - A hyperplane for R^3 is a straight line of the generic form ax+by+c
- It is very much possible that multiple values of a,b and c will exist and hence, multiple lines.
- The task of SVM is to find the line by solving for the optimum values of a, b and c.

2 Linear Classification

It is important to understand linear classifiers if we are to understand SVMs. In simple words, a linear classification is a strategy by which we seggregate the data into 2 classes. In a 2-d visualization, it is equivalent to saying we draw a line that divides the plane into 2 parts - data on the left of this line, and data on the right of this line. In case of 3-d, it will be a plane that divides the space into 2 parts. In general, a N dimensional hyperspace can be divided into 2 parts by a N-1 dimensional hyperplane. Idea remains the same. See figure 1.

Such a classifier is called discriminative classifier, or algorithm - it can only separate the data into 2 (or more) groups, it can't say something about the groups thus formed. This is in contrast to generative models which tells something about the group, and can be used to generate more instances of the data in that group.

Typically, we say that point is less than 0, then it lies on the left of the line, otherwise it is on the right of the line. Here is a way to understand this using 2-d plan example. A linear classifier for a plane is 1-d, a line. Let's say $ax + by + c = 0$ So if the equation is $x + y = 4$, then $a = 1, b = 1, c = -4$

If we take point $(10, 10)$, clearly it is on the right side of the line $x + y - 4 = 0$. If we substitute the point in the line, we get $10 + 10 - 4 > 0$. Similarly, if we take origin $(0, 0)$, it is on the left side of the

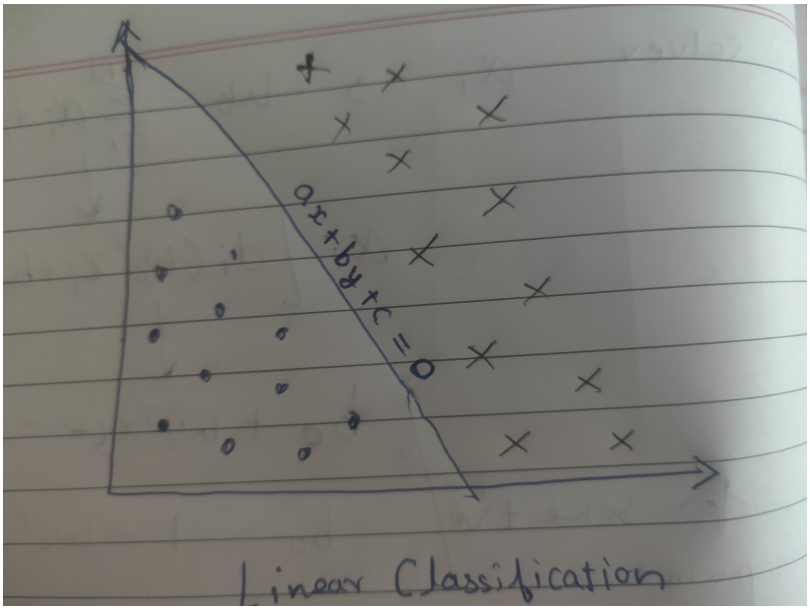

Figure 1: Linear classifier

line. When we substitute, we get $0 + 0 - 4 < 0$. So the points on the left of the line will yield < 0, and those on the right will yield > 0.

See figure 2.

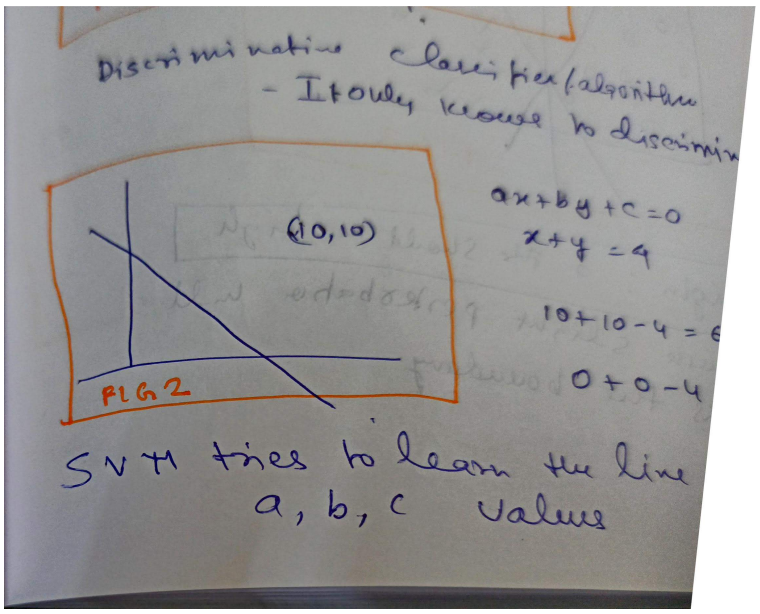

Figure 2: Linear classifier

SVM tries to figure out what this line will be. Essentially, that means getting to the value of a, b and c in the equation above, if it is 2-d. It uses the given data to come up with these values. For now, let's assume that the data is linearly separable, which means such a line does exist that separates the data into to clean classes. Later we will see how to handle data that is not linearly separable (next class).

Another thing to note is that it is not necessary that only a unique line exists that separates the data, there can be more than one such line. In that case, how do we pick such a line? Which is the best line to pick?

See figure 3.

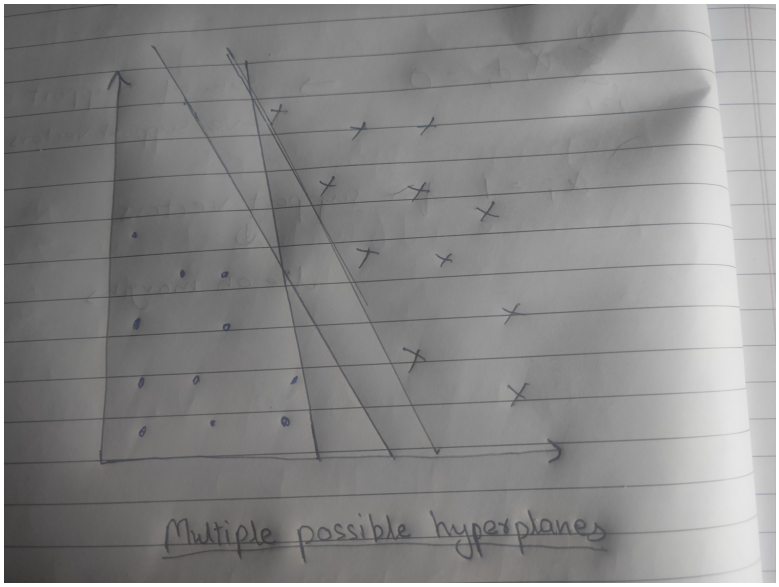

Figure 3: Linear classifier

The 'best line' is the one which maximizes the 'margin', where margin is the distance between the line and the nearest data point. The reason for this is simple: if there is a data point that is very close to the line, even a slight perturbation can change the classification of the data point (the point will 'tip' to the other side of the line). To keep the classifier robust, we want to reduce such scenarios, and hence we want a line with the highest possible margin.

Mathematically, the linear equation above can be written as $w_1x_1 + w_2x_2 + b = 0$, or in a vector form:

$$w^T X + b = 0 \tag{1}$$

w is a column vector. For ex, for 4-dimensional vector, above equation looks like this:

$$\begin{bmatrix} w_1 w_2 w_3 w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + b = 0$$

So now the question is this: how do we find the line that is a linear classfier, AND has the maximum margin. There are 2 ways of intuitively thinking about it:

1. We grow a bubble around each point, and then see how big a bubble is required to touch the line.
2. We select the line which has the biggest margin - the distance between the line and the nearest points.

See figure 4.

As you may notice, only a few points will make a difference and determine which line has the best margin. These are the points that are closest to the line, and are called **support vector**. Hence the name of this algorithm.

Vapnik and Chervonenkis, through their 3 papers, provided the breakthrough insight that there is an upper bound on the test error of a classification model. This is through what is called VC dimension.

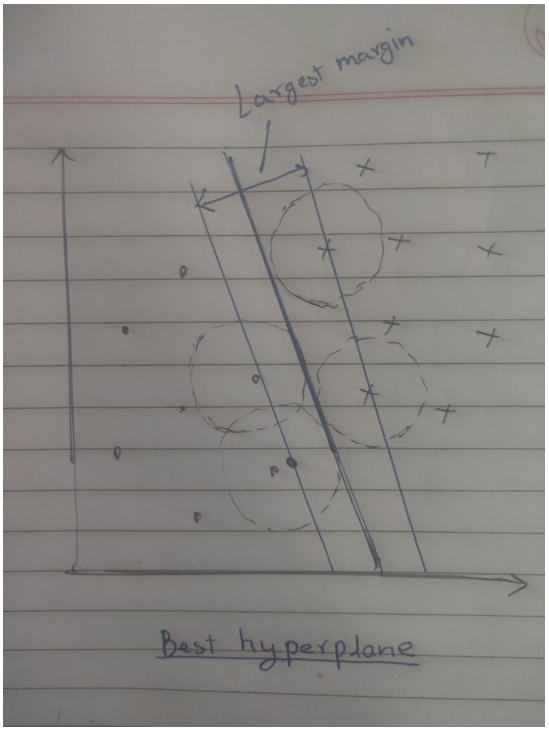

Figure 4: Linear classifier

Bound on expected loss is

$$R(\alpha) \leq R_{train}(\alpha) + \sqrt{\frac{f(h)}{N}} \tag{2}$$

Second term is the VC dimension.

$$h \leq min\left\{d, \left\lceil \frac{D^2}{\rho^2} \right\rceil\right\} + 1 \tag{3}$$

where

D: data diameter

ρ: margin

d: dimensionality

$\frac{\rho}{D}$: Relative margin

See figure 5.

So the best way of minimizing the error in test is to minimize the 2 terms in the formula - keep training error low, and minimize h. In fact, we can ensure that second term becomes independent of the dimension if relative margin is maximized ($\frac{\rho}{D}$ maximized means $\frac{D}{\rho}$ is minimized, and hence will be less than d, the dimension).

Key points to keep in mind:

1. Maximizing margin improves generalization

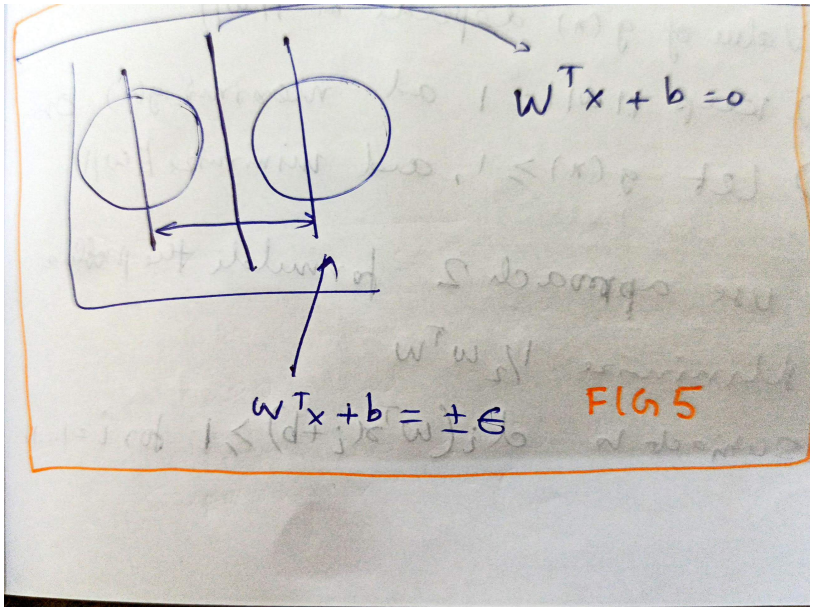

Figure 5: Linear classifier

2. h can be made independent of d

Let's consider 2 parellel lines to the classifier line. The classifier line is $w^T X + b = 0$, and since these other 2 lines are parellel, they only differ in the constant term, they can be written as $w^T X + b = \pm\epsilon$

or, we can also say that we have the line $g(x) = 0, where g(x) = w^T X + b$, and the margin lines are $g(x) = k$ and $g(x) = -k$.

See figure 6.

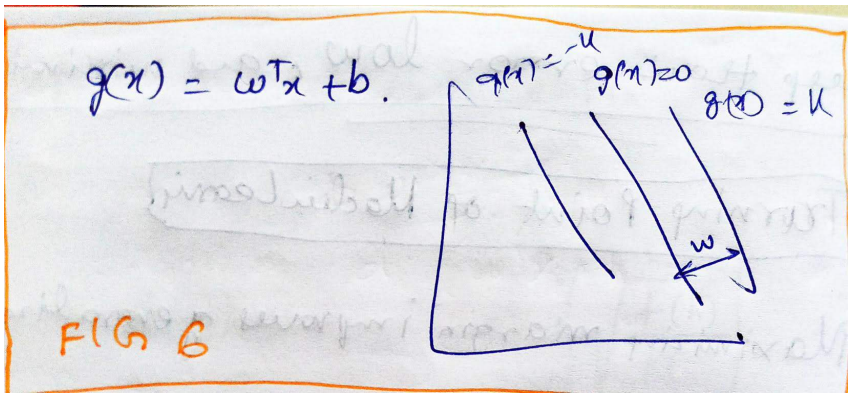

Figure 6: Linear classifier

Our target is to maximize k such that

$$w^T x_i + b \geq k for d_i = 1 w^T x_i + b \leq k for d_i = -1 \tag{4}$$

where

$d_i = 1$ is the class on the right of the line

$d_i = -1$ is the class on the left of the line

Value of $g(x)$ depends on $\|w\|$. We have 2 ways of solving this equation.

1. Keep $\|w\| = 1$ and maximize $g(x)$, or,
2. Let $g(x) \geq 1$ and minimize $\|w\|$

We use approach 2 since it is more tractable mathematically. Using this, we can formulate the problem like this:

Minimize $\frac{1}{2}w^T w$

subject to $d_i(w^T x_i + b) \geq 1$ for $i = 1..N$

$w^T w$ is norm of the vector. In case of scalar, this becomes square, ensuring this doesn't become negative The condition above is a clever way fo combining the 2 equations we have for the points on the right side of the line ($d_i = 1$) and for the points on the left side of the line ($d_i = -1$):

$$w^T x_i + b \geq 1 for d_i = 1 w^T x_i + b \leq 1 for d_i = -1 \tag{5}$$

See figure 7.

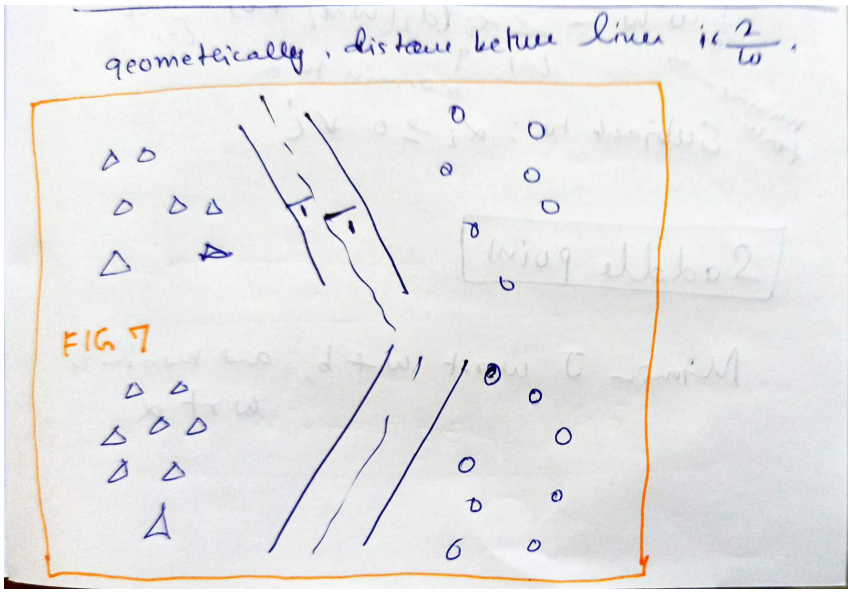

Figure 7: Linear classifier

If we look at it geometrically, the distance between the lines is $\frac{2}{w}$, and so when we minimize $\|w\|$, we get the best $g(x)$

3 Primal and Dual Forms

The primal form of the SVM optimization is:

$$\begin{gathered} Minimize : \phi(w) = \tfrac{1}{2} w^T w \\ Subject\,to : d_i(w^T x_i + b) - 1 \geq 0 \; \forall i \end{gathered} \tag{6}$$

The same optimization problem has a Lagrangian form as:

$$\begin{gathered} J(w, b, \alpha) = \tfrac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i d_i (w^T x_i + b) + \sum_{i=1}^{n} \alpha_i \\ Subject\,to : \alpha_i \geq 0 \end{gathered} \tag{7}$$

At optima, from eq 7.

$$\begin{gathered} \tfrac{\delta J}{\delta w} = 0 \\ \implies w = \sum_{i=1}^{n} \alpha_i d_i x_i \end{gathered} \tag{8}$$

$$\begin{gathered} \tfrac{\delta J}{\delta b} = 0 \\ \implies \sum_{i=1}^{n} \alpha_i d_i = 0 \end{gathered} \tag{9}$$

Also, from KKT conditions,

$$\alpha_i [d_i (w^T x_i + b) - 1] = 0 \tag{10}$$

Finally, the dual form of the optimization problem (6) can be formulated, using (8),(9),(10) in eq 7:

$$\begin{gathered} Maximize : Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j x_i^T x_j \\ Subject\,to : \alpha_i \geq 0 \qquad \sum_{i=1}^{n} \alpha_i d_i = 0 \end{gathered} \tag{11}$$

Each point, for which $\alpha_i > 0$, is known as a support vector.
Further, it can be seen that the vector w is just a linear combination of the support vectors (eq 8).

For more on VC dimension in SVM and SVM in general, here are some good references:

References

[1] SVM - Columbia Statistics, Link: http://www.stat.columbia.edu/ madigan/DM08/svm.ppt.pdf

[2] https://jeremykun.com/2017/06/05/formulating-the-support-vector-machine-optimization-problem/

[3] https://towardsdatascience.com/demystifying-maths-of-svm-13ccfe00091e

[4] https://www.svm-tutorial.com/