AIML - CS 337

Lecture 10: Classification

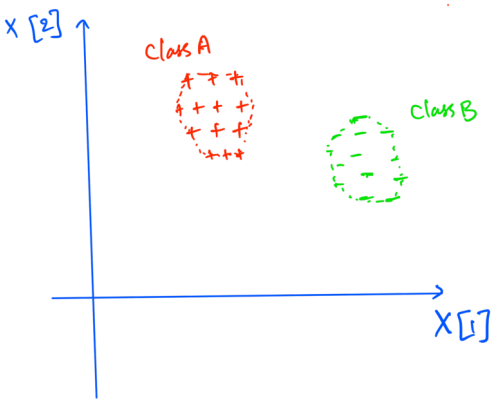
8th September

Lecturer: Abir De *Scribe: Hardik, Shrey, Teja, Tejal*

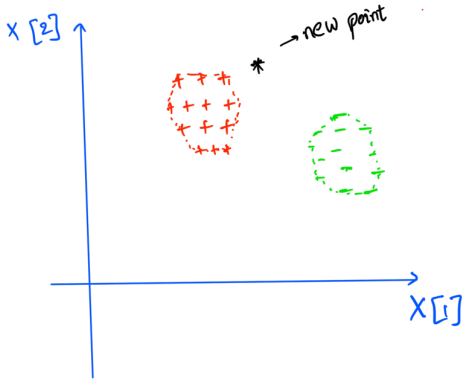
1 Classification: Primitive Approaches

Introduction

We are given a discrete data set D = $\{x_i, y_i\}$, where $y_i \in C$. The elements of C have no ordinal relation ie. their order is irrelevant. Ex. C = {"cat" , "dog", ...} or {+1, -1}
Let us assume $x_i \in \mathbb{R}^2$, i.e; x_i has two entries x[1] and x[2], and a simple set of classes C, where $|C| = 2$ (i.e; C = $\{+1, -1\}$). Plotting data in plane with X[1] on x-axis and X[2] on y-axis then, the plot may look something like this



We want to find an unambiguous mechanism to classify new points into one of the given classes. To this end, we discuss some ideas below:



Constant model

This model is not practical and gives non-zero error. It is equivalent to completely ignoring the existence of the minority class.

Unsupervised Classifier:

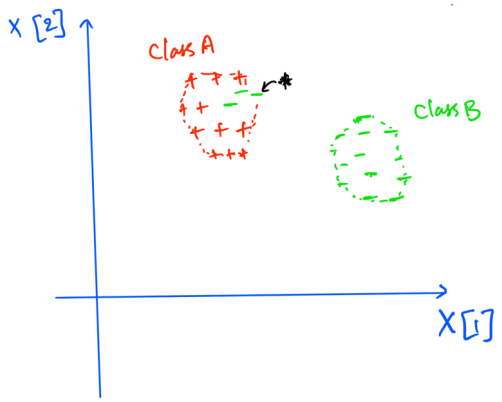
This involves methods like calculating the nearest point, or the centroid of a class.

This model gives uncertain output as it is sensitive to distance values.

Unsupervised Classifiers give uncertain results because, when we make a transformation(like scaling or normalizing) to the data then this new point may get closer to other classifications but in the Linear Regression change in x also makes adjusting w when we are training.

So arguing just by using distance in classifiers, we don't get good accuracy.

Outliers are another problem with some of the unsupervised methods. For example, if a point of class B is in a region that is mostly made up of points from class A, the new point may get classified as belonging to class B.



So, we move on to using a linear classifier for the problem:

2 Linear Classifier (SVM)

Let us consider a simple Linear classifier, $w^T X$, i.e.

$$y = \begin{cases} 1 & \text{if } w^T X \geq 1, \\ -1 & \text{if } w^T X \leq -1. \end{cases}$$

Geometrically, $w^T X = a$ represents a plane in the input space. In our case, the separation hyperplane would be $w^T X = 0$ which would be a plane passing through origin. This will not be able to generalise over the input data space. To overcome this, let's add a bias term, b in our classifier, i.e.

$$y = \begin{cases} 1 & \text{if } w^T X + b \geq 1, \\ -1 & \text{if } w^T X + b \leq -1. \end{cases}$$

This allows the separating hyperplane to move more freely in space and lets us classify more generic data points.

How to find w and b?

As we discussed, the classifier $w^T X + b$ works quite well for any general input space, but how should we find these w and b for our input data.

Proposal 1 One of the student suggested to find two points, one in each class such that the distance between them is minimum and draw a line perpendicular to the line joining these points. This will be the classifier. The problem with this approach is, first, the complexity. We need to iterate over all pairs of points, so if there are N points in each class, the complexity would be $\mathcal{O}(N^2)$. Secondly, it is not scalable, i.e. as the number of classes grows, this gets much more complicated.

Convex optimisation

A much faster and systematic method exists that we are already familiar with. This is the method of Convex optimisation. We define a objective function $C(w, b)$ and find w^* and b^* which minimize the objective function. An obvious question comes to mind, what objective function should we choose?

Proposal 1 For a given line $w^T X + b$, we find the points from each class with minimum perpendicular distance and maximize the product of these distances. More formally, let $d_i^+(w, b)$ denote the distance of a point i in the $+$ class and $d_j^-(w, b)$ denote the distance of a point j in $-$ class. Then we have the expression:

$$\max_{w,b}(\min_i(d_i^+(w, b)) * \min_j(d_j^-(w, b)))$$

This is a pretty good strategy but there is a small drawback. We know that the perpendicular distance would be given by $\frac{|w^T x_i + b|}{\|w\|}$ Since we are considering two points simultaneously, this becomes a bit hard to optimize.

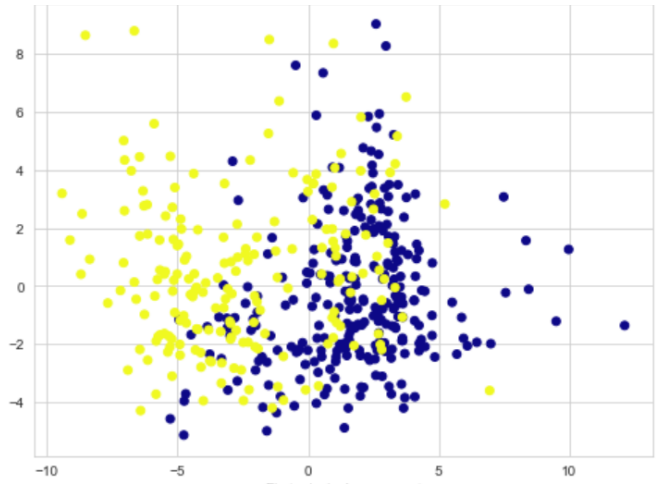
We make make a slight modification to make it work. Instead of taking the product of the distances, we taking the minimum across all points and try to maximize this distance. Formally,

$$\max_{w,b}(\min_{i,j}(d_i^+(w,b), d_j^-(w,b))$$

There we go, we have a objective function that we can plug-in in the convex optimisation problem and get w^* and b^* that give us our classifier.

3 SVM with Complications

The boundary between the two classes is often not so pronounced, Especially when dealing with a large number of samples, outliers and edge-case samples tend to creep in. This can also be attributed to the collected data not having sufficient information for the classification. The data is then said to be *linearly inseparable*. An example is given below:



While there are no values for w and b to get $y_i(w^T x_i + b) > 0 \quad \forall (x_i, y_i)$, one possible suggestion is to minimize the number of misclassified points:

$$w^*, b^* = \text{argmin}_{w,b}(\sum_{i=1}^{n} \mathbb{I}(y_i(w^T x_i + b) < 0))$$

However, the set and number of misclassified points may vary largely with w and b. One "algorithm" in line with this idea is to:

1. Get w,b = SVM(batch).

2. Remove points (x_i, y_i) from the batch that are misclassified by the SVM.

3. w,b = SVM(batch')

...until a large number of points are classified. But with each new pair (w,b), there might be new points which are misclassified. I quote,

> "whatta Circular logic **!!!**"

Soft Margin SVM

A different idea is to use the notion of *Hinge Loss* to measure the misclassification error. The expression is:

$$L_H = \sum_{i=1}^{n} max(0, 1 - y_i(w^T x_i + b))$$

One other method of dealing with this is *soft margin SVM*. Here the idea is to allow the SVM to make a certain number of mistakes and keep the margin as wide as possible so other points can still be classified correctly.
The loss function is chosen as:

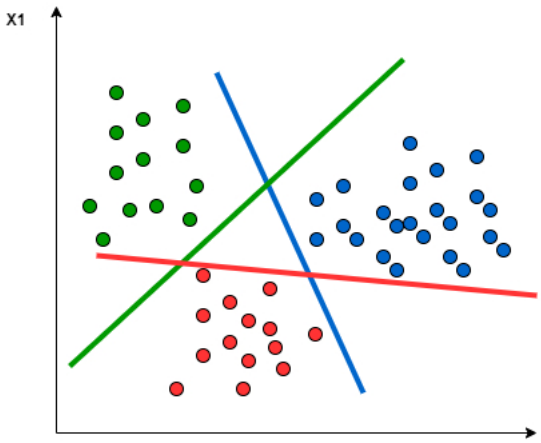
$$L = \frac{1}{2}||w||^2 + C(L_H)$$

C is a hyper-parameter that decides the tradeoff between maximizing the margin and minimizing the mistakes. Without the worry for linear inseparability, the problem is also called Hard Margin SVM.

SVM for k classes

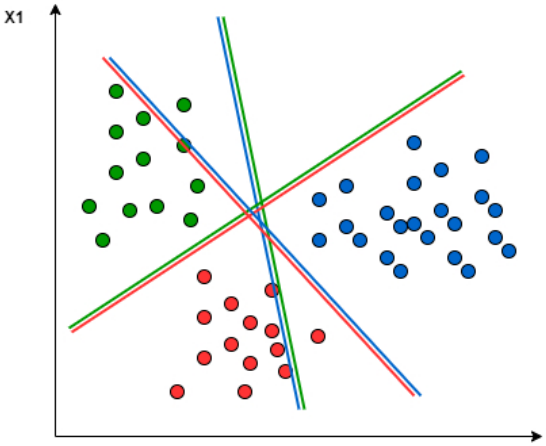
While we have looked studied how to use SVM for binary classification problems, we can also apply two separate approaches to use SVM for multiclass classifications. These are:

1. **One-to-Rest Approach**: We generate k boundaries, for each run of SVM with classes as $(C_k, \cup_{i \neq k} C_i)$. The class of a point is the one corresponding the margin across which it lies. Note that a single point may lie in the accepting region of multiple classes.



One-to-Rest Approach

2. **One-to-One Approach**: We generate ${}^{k}C_2$ boundaries, one for each run of SVM on applying to each pair of classes $(C_i, C_j)_{i \neq j}$. This approach is more useful for making comparative statements between classes.



One-to-One Approach