Linear Discriminant Function - Passerini

Mattia Carolo - @Carolino96

Linear Discriminant Function

While for generative algorithms we infer knowledge from the modelled data distibution we have, with **dicriminative learning** we focus directly on modelling the discriminant function. For example in classification woth this method we directly model the decision boundary.

Formalization

Now we want to learn an $f(x)$ which maps x in y ($f : x \to y$). So we formalize the linear function like

$$f(x) = w^T x + w_0$$

The discriminant function is a linear combination of example features in which w_0 is the so called *bias* or *threshold* and it is the simplest possibile discriminant function. The problem since it's linear we can't use it for complex problems but it's the most easy one to apply and once applied we can always expect the results so it's less prone to *overfitting*.

Linear Binary Classifier

Now a linear **binary** classifier it's a linear predictor which need to distinguish between two classes. So in this case we can get the normal linear discriminant function and get his sign so if the function will be positive the label will be 1 otherwise -1

$$f(x) = sign(w^T x + w_0)$$

if $f(x)$ is equal to 0 than we are on the **hyperplane** which will be the decision boundary. Moreover the weight vector w is orthogonal ($\perp$) to the decision hyperplane. We can formalize it with a bit of algebra where we take two points on the hyperplane so their function is equal to zero as we do below where

$$\forall x, x' : f(x) = f(x') = 0$$

now if we replace them with their formalization we get that

$$w^T x + \cancel{w_0} - w^T x' - \cancel{w_0} = 0 \Rightarrow w^T (x - x') = 0$$

which means that w is orthogonal respect to $(x - x')$

Functional Margin

The functional margin can be seen as the confidence we have on a particular prediction. It's value is the value of $f(x)$ before applying the sign. The larger the value the more confident we are on that particular prediction.

Geometric margin

We call the geometric margin the distance between x and the hyperplane. We obtain it from the functional margin by doing

$$r^x = \frac{f(x)}{||w||}$$

where r^x will be the margin

> we can see that the geometric margin is just the functional margin normalized

check demonstration on ot 27:17

Biological Motivation

shish

Perceptron

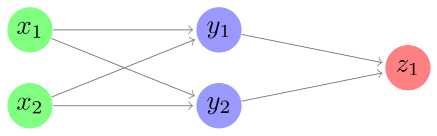
If we want to formalize a neuron we do it like a normal linear function made as

$$f(x) = sign(w^T x + w_0)$$

As we can see from the image below we have a set of inputs to the neuron from x_1 to x_m where every input will be multiplied for his weight(which can be positive or negative). Later all this weighted inputs will be summed up together with w_0 and if the sum will be higher than the threshold we will predict 1 otherwise -1 (u need to surpass the threshold in order to activate it so that's why is called activation function)

add perceptron img

now if we take the function written before we can represent a series of primitive logical operations like AND, NAND or NOT since the results can be separated by a single line. In cases like the XOR in which we cannot learn the linear function itself since it doesn't exist, we can represent those cases by a network of two levels of perceptron.



For example in the case of the XOR we can decompose it in

$$x_1 \oplus x_2 = (x_1 \wedge \overline{x_2}) \vee (\overline{x_1} \wedge x_2)$$

where the node y_1 will take care of learning $(x_1 \wedge \overline{x_2})$, y_2 will learn $(\overline{x_1} \wedge x_2)$ and the node z_1 will learn the conjuction of the two.

Learning linear models

We already know the formula for the perceptron which is

$$f(x) = sign(w^T x)$$

but we need first a way to incorporate bias in our formula. To do this we use **augmented vectors** in which we add for both the feature vectors $\hat{x}$ and the weight vector $\hat{w}$ so they are made like

$$\hat{w} = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix} \hat{x} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$$