Science of Information, Statistics, and Learning
Lecture 15

Parmanand Khajuriya 12D070041, Bankuru Dharma Teja 130050049,
Jay Mardia13D070011, Sudipto Mitra 130100014,
Sachin Garg 13D070061

Mar 17,2017

Abstract

In the previous lecture, we studied linear regression and spoke about the need to choose a hypothesis class, and given a hypothesis class, to choose an underlying model for generating our data which could motivate the error measure we use to pick the best hypothesis from out hypothesis class. Specifically, we saw that Gaussian measurement noise coupled with wanting to maximize likelihood lead to a square-error. In this lecture, we study the binary classification problem and relate the quantities that arise to Physics and Information Theory. In doing so, we end up studying the single neuron as a classifier.

Most of today's lecture is based on Ch-39 from "Information Theory, Inference, and Learning Algorithms" by David J.C. MacKay

Introduction

Suppose we are given a bunch of points in an n-dimensional feature space ($\mathbb{R}^n$) with each point labelled by a Yes, No or 0,1.

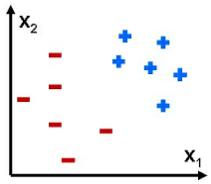

Figure 1: A bunch of data points with labels of Yes,No represented by [Plus,Minus]

1 Problem Formulation

Input:
$x_1 \in \mathbb{R}^n, t_1 \in \{Yes, No\}$
$x_2 \in \mathbb{R}^n, t_2 \in \{Yes, No\}$
$x_3 \in \mathbb{R}^n, t_3 \in \{Yes, No\}$
.
.
$x_N \in \mathbb{R}^n, t_N \in \{Yes, No\}$

Desired Output:
A procedure $l : \mathbb{R}^n \to [0, 1]$ such that l does a *good job* classifying data points. What we meant mean by good job will depend on the error measure we use which will depend on what our model for generating the data is.

The things we want to keep in mind are:

We want to reduce some measure or error. Some proposed measures were squar-distance, number-of-misclassifications.

We want to fix a hypothesis class that is big but not too big. Suppose out hypothesis class allowed many many functions. We could choose a function that is 'Yes' on the 'Yes' samples and no everywhere else. This would give zero error, but would not be a good predictor. Hence we want a restricted hypothesis class that reflects our assumptions on how the data must have been generated.

A good idea would be to use a linear separator. Specifically, a hyperplane in $\mathbf{R}^n$ that can separate the two types of points. This is demonstrated in the image.

Some issues with this could be that the data points may inherently be such that they cannot be separated by a line (hyperplane).

Our procedure should be robust to a few misclassifications of our given data. It should be able to discount outliers.

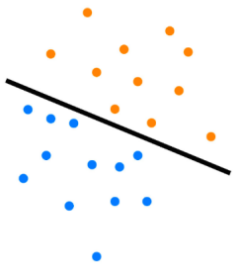

Figure 2: A separating line that can help classify our data

2 Logistic Regression (Single Neuron Classifier)

We restrict our hypothesis class to linear separators (hyperplanes) and, given an input $x \in \mathbb{R}^n$, compute $a = \sum_{i=0}^{n} w_i x^i$. Here the vector $w \in \mathbb{R}^n$ is our classifier and x^i is the i-th component of the vector x.

The quantity a is then sent through a function whose output we use to classify the input x.
Some candidate functions are:

1. $y(a) = a$
2. $y(a) = \frac{1}{1+e^{-a}}$
3. $y(a) = sign(a)$

Pictorially, this process can be represented as a neuron in the following manner.

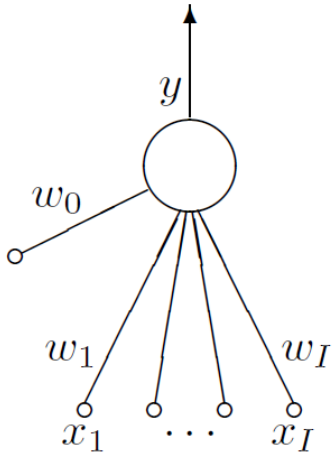

Figure 3: A single neuron

Why do people choose $y(a) = \frac{1}{1+e^{-a}}$? This function, called the logistic function, has the disadvantage that it is hard to compute digitally. It's advantage is that it is analytically nice and smooth.
But the key reason is that using it allows us a nice interpretation of the neuron via physics.

Physics Interpretation to the Logistic function: Suppose there is a 2-state neuron. The state "Yes" has energy E and the state "No" has energy 0. If $E(Yes) = -\sum_{i=0}^{n} w_i x^i$ and E(No) = 0.

Then from thermodynamics we would expect $P(Yes|E) \propto e^a$ and $P(No|E) \propto e^0$.

This would give us $Pr(Yes) = \frac{e^a}{1+e^a} = \frac{1}{1+e^{-a}}$. Hence we can interpret this as the energy barrier for neurons firing. The probability of the neuron firing changes with change in a.

Of course, the actual neuron in the brain is much more complicated and this is merely a very simplified toy model that is nevertheless useful in providing intuition when we consider many neurons together.

3 Error Model

We want to choose the weights w that minimize the error according to some error model. We now present an error model that can be interpreted both information-theoretically as well as statistically.

$$G(w) = -\sum_{i=1}^{N}[t_i log(y(x_i; w)) + (1 - t_i) log(1 - y(x_i; w))]$$

This can be interpreted as the total expected surprise from the data given our prior belief w. Hence minimizing this quantity seems reasonable. This is the information theoretic interpretation.

This quantity can also be seen as the negative log-likelihood of the data. Hence minimizing it is the same as maximizing the likelihood.

As a function of w, $G(w)$ need not be convex. However, the easiest thing we can do is gradient-descent and obtain a local optima. We shall see that gradient descent for this particular error model has a nice interpretation.

Gradient Descent:

$$\frac{\partial G}{\partial w_j} = -\sum_{i=1}^{N}[(t_i - y_i)x_i^j]$$

The error decreases in the direction of the negative of the gradient. Hence we pick a learning rate and move in the direction opposite to the gradient in w-space.

This is called Hebbian Learning. It has the nice interpretation that if some input coordinate (say x^j) was right and we did not listen to it, it's contribution in the weights is increased (w_j increases). Similarly if some input is wrong but we gave it large weight, we reduce its contribution.

Remarks on Hebbian Learning:

1. If the data points are linearly separable, after some time, Hebbian learning does not converge but diverges and all the weights go to ∞. This means

the sigmoids become sharper and sharper but this may not lead to any actual decrease in prediction error. Hence this is false refining.

2. People counter this in different ways. Some just stop the algorithm after a while, even though it has not converged. This is called early stopping.

3. Some add a regularizing term ($\lambda||w||^2$) corresponding to the weight of w to the error function being optimized. This means that weights of w won't increase too much unless there is a significant gain associated with it.