Overview of Supervised Learning

Introduction

As seen in introduction the process of predicting the value of output variables from a set of input variables after learning from a training data set is known as supervised learning. The term input can also be replaced by features, predictors or independent variables; whereas the term output can be replaced by responses or dependent variables.

Variable Types and Terminology

The output variable can be either quantitative or qualitative (also known as categorical or discrete or factors); in case of qualitative output the classes are usually denoted via. some class label instead of a number, however during the learning process numbers (referred to as targets) are used instead of the labels. Another important category is ordered categorical (e.g. small, medium, large); these categories apply to input as well.

These two types lead to a distinction in the task of supervised learning, when predicting qualitative output we term the task as regression and for quantitative output we term it as classification; however in general terms both of these are function approximation tasks. Distinction in task name involved can also be attributed to different input types (more on this later).

If the output type is categorical with more than one categories the best option is to use dummy variables; e.g. the output has K different categories, then in learning task a vector of K binary variables is used to denote the class, where only one of the K different values is on at a given point of time, although more compact notations are possible, dummy variables are chosen because they are symmetric in levels of factor. (levels of a factor are the number of variations of the factor that were used in the experiment; symmetric in levels of factor means that no matter the number of variations used, the decoding scheme remains the same, useful for computers).

Book notations

Generic level (means the entire set of input or output)

X – Input variable, if X is a vector then its components can be accesses with X_j.

Y – Quantitative output

G – Qualitative output, G stands for group here

Observed level (referring to specific input or output)

x_i – i^{th} observed value in the input, if X is a vector, then so is x.

X – Input matrix, N x p in dimension if input is comprised of N p-vectors x_i, i = 1 to N (dimension of x is p). All vectors are assumed as column vectors and as such the ith row of **X** is denoted by x_i^T.

A learning task can thus be defined as given input X we need to predict the output value Y (predicted value denoted by Y') if output takes real values or G (predicted value G'), where G' takes the values from the same set as that of G. For a two class G, output can be taken as binary variable and then treated as quantiative output Y; this approach can be generalized to K-level outputs as well. The training data is thus a set of (x_i, y_i) or (x_i, g_i) depending on the problem involved.

Linear Models and Least Squares

Given a vector of inputs $X^T = (X_1, X_2,...,X_p)$, the output variable Y in terms of a linear model is given as:

(1.0)

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j.$$

(Note Y cap is written as Y' by me, all cap notations will be written in this format henceforth)

The term β_0’ is the intercept also known as the bias term. For the sake of convenience 1 is added as the first term to the vector of inputs and β_0’ is included in the vector of coefficients β_j’. In that case equation 1.0 can be re-written as:

(1.1) $Y' = X^T\beta'$

The dimensions of vector β depend on the dimensions of the input and output vectors, in general terms if X is a p-dimensional vector and Y a K-dimensional one, then the dimension of β will be p x K.

It’s easy to see that the output variable here is modelled as a linear function in the input space. Now the question arises how do we fit this model in terms of the training data. The most popular method is known as the method of least squares. In this method the co-efficients β are chosen such that the residual sum of square is minimized, mathematically we want to minimize the value of the following function:

(1.2)

$$\mathrm{RSS}(\beta) = \sum_{i=1}^{N}(y_i - x_i^T\beta)^2$$

In matrix notation the function can be re-written as:

(1.3)

$$\mathrm{RSS}(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$$

Here **X** is the N x p dimensional input matrix and **y** is the N dimensional output vector.

Differentiating equation 1.3 w.r.t β and equating it to 0 for the minima we get the equation:

(1.4)

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) = 0$$

If $\mathbf{X}^T\mathbf{X}$ is non-singular then the unique solution for β is given by:

(1.5)

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Now the predictions can be made easily using this value of β.

Nearest Neighbour Methods

This model can be described in mathematical terms as:

(2.0)

$$\hat{Y}(x) = \frac{1}{k}\sum_{x_i \in N_k(x)} y_i$$

Here $N_k(x)$ are the k closes points to x. Closeness is defined by a metric, this one being Euclidean distance. So in general terms we take the average output value of k closest points to the given input and consider that as the predicted value.

Emperically it can be seen that the error using this method will be zero when we take k=1 and the error is increasing function w.r.t the value of k. However this is true only for the training data and any true conclusions can only be drawn after application of the testing data.

Seemingly this model has only parameter for training and that is k, however if construct the decision boundaries beforehand it is easy to see that the total number of neighbourhoods is going to be N/k which is certainly more than p parameters of the least square model and this value decreases with increasing value of k.