Big Data & Predictive Analytics
Lecture notes on Classification

School of Informatics
University of Leicester

1 Introduction

Recall that regression aims to predict an output variable that is always a continuous variable. But, what if the output variable is a discrete number or a set of categorical variables? In these cases, we need to classify given records and we talk about *classification*. One of the classification algorithms in machine learning is *logistic regression*. *Logistic regression* is a supervised learning algorithm aiming at predicting a label (output) given a set of predictors. It is basically a variation of linear regression where the output variable is a binary or categorical variable. The two regressions are similar in the sense that they both assume a linear relationship between the predictor and output variables. However, as we will see later on, logistic regression requires the output to undergo some transformation.

Formally, given a training set (x_i, y_i) for $i = 1 \dots n$, we want to create a classification model f that can predict label y for a new x. f is found by using machine learning algorithms. Consider for example the function:

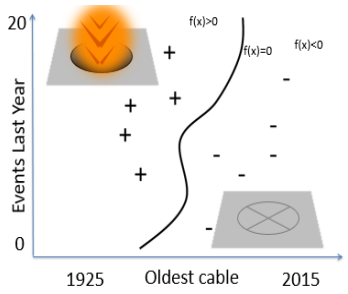
$$y = f(x) = \text{function}(\text{Events Last Year}, \text{Oldest Cable})$$

which takes two features `Events Last Year, Oldest Cable` and produces a number y such that he sign of y gives us the predicted class. If we have a new example we have never seen before, we take its features vector, and we plug those values into f, then we get a number. If it is a positive number we predict an explosion, if it is a negative number we predict no explosion. The function depicts a boundary that splits the set of examples in the the two classes: positive and negative.

However, the classifier may predict the wrong class for some examples. It is therefore important to find the percentage error of the classifier. We define the *loss function* as follows:

$$L(f) = \frac{1}{n} \sum_{i=1}^{n} (y_i \neq \text{sign}(f(x_i)))$$

The function $L(f)$ gives the fraction of the number of times y_i is not the same as $\text{sign}(f(x_i))$. Naturally we would like to minimise the Loss function.



In this example, prediction errors may occur in the following cases:

- the predicted $f(x_i)$ has been found by our model to be positive but the observed value y_i is negative.
- the predicted $f(x_i)$ has been found by our model to be negative but the observed y_i is positive.

We will need to be able to count the number of examples that are classified correctly as well those classified incorrectly, which should give some kind of classification error percentage. Indeed, the error is related to the quality of the classifier model f while f comes from a machine learning algorithm. A basic outline of a machine learning algorithm can be described as follows:

1. Split data randomly into a training and test sets
2. Estimate the coefficients of the model giving us a training model
3. Calculate the scores of the model for the training set and more importantly for the test set.
4. Evaluate the model

Finding the best model from a given dataset is an iterative process but we can rely on Ockham's razor, which is a principle used in statistical learning theory. The principle states that 'the best models are simple models that fit the data well'.

Here are few extra scenarios wherein logistic regression can be applied:

- To predict whether a random bank customer will default a particular loan or not, given their details such as income, gender, credit history, etc.
- To predict whether a team will win or lose a match, given the match and team details such as weather, form of players, stadium, etc.

2 Logistic Regression

Logistic regression is supervised learning algorithm used to predict categorical-value outcomes. It is simple, fast and often competes with the best algorithms in predictive modeling. It can also deals with question such as what is the probability of a customer buying a product given their gender.

Imagine a situation where we have a dataset from a supermarket store about the gender of the customer and whether that person bought a particular product or not. We are interested in finding the chances of a customer buying that particular product, given their gender. To answer this question, we can start by generating the *contingency table* from the given dataset.

2.1 Contingency table

A contingency table is basically a representation of the frequency of observations falling under various categories of two or more variables. The following code snippet

```
contingency_table=pd.crosstab(df['Gender'],df['Purchase'])
contingency_table
```

will give us the contingency table for the variables `Purchase` and `Gender` as follows:

Gender/Purchase	No	Yes
Female	106	159
Male	125	121

We can then use compute conditional probabilities by simply using the following formula (no need to use Bayes theorem in this case).

$$P(Purchase|Male) = \frac{\text{Total number of purchases by males}}{\text{Total number of males in the group}}$$

Using this kind of formula and the contingency table, we get the following probabilities

$$\begin{array}{ll} P(Purchase|Male) & = \frac{121}{246} = 0.49 \\ P(NotPurchase|Male) & = \frac{125}{246} = 1 - 0.49 = 0.51 \\ P(Purchase|Female) & = \frac{159}{265} = 0.60 \\ P(NotPurchase|Female) & = \frac{106}{265} = 1 - 0.60 = 0.40 \end{array}$$

Let us now introduce the concept of *odds ratios* important to understand the logistic regression algorithm.

2.2 Odds ratios

Odds of success for a group are defined as the ratio of probability of successes (purchases) to the probability of failures (non-purchases). If p is the probability of success, then the odd ratio for success is defined as

$$p/(1-p)$$

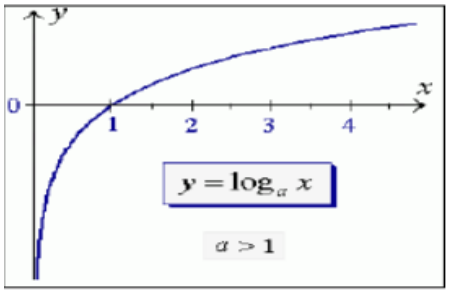
If this ratio is greater than 1 then the event is more likely to happen within the group; otherwise it is not. For example the odds of purchase by males is calculated as

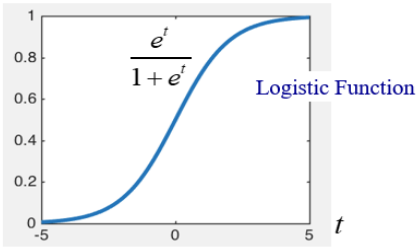
$$(121/246)/(125/246) = 121/125$$

and the odds of purchase by female customers is

$$(159/265)/(106/265) = 159/106$$

These odds ratios expressed the likelihood of a customer buying a product given they male or female. Similarly, we can also calculate the odds of success for males as the ratio between the number of successes within the males group over the number of failures within the males group.





2.3 From Linear to Logistic Regression

We know that in the case where the model f is linear with a single feature x, we can write

$$y_i = f(x_i) = b_0 + b_1 x_i$$

As the outcome is binary 0 or 1, we will not aim at predicting exactly 0 or 1 but the probability associated with 0 or 1, therefore calculating numbers between 0 and 1 and then find a threshold enabling to map the obtained probability to either the outcome 0 or 1.

The problem we are facing here is to do with the range of the probability p between 0 and 1 and the quantity $b_0 + b_1 x_i$, which covers the whole set of real numbers. However, if we look at the plot of the logarithmic function; for a base greater than 1, the plot is as follows: We can make the following observations:

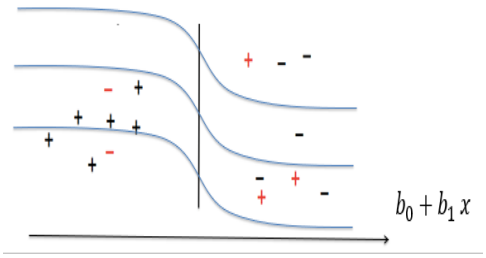
- the odds ratios $p/(1-p)$ are between 0 and $+\infty$
- $\log(\frac{p}{1-p})$ is well defined and covers the set of real numbers as does the quantity $b_0 + b_1 x$.

It follows that

$$\begin{aligned} \log(\tfrac{p}{1-p}) &= b_0 + b_1 x \\ \tfrac{p}{1-p} &= e^{b_0+b_1 x} \\ p &= \tfrac{e^{b_0+b_1 x}}{1+e^{b_0+b_1 x}} \end{aligned}$$

It turns out that the function $f(t) = \frac{e^t}{1+e^t}$ for t taking its value in the set of real numbers is between 0 and 1 as shown in the following plot: The machine learning algorithm will provide us with the coefficients b_0, b_1 and then we can estimate the probabilities $p = \frac{e^{b_0+b_1 x}}{1+e^{b_0+b_1 x}}$ for a given value of the predictor x.

How do we map the obtained probabilities to the outcomes 0/negative/No or 1/positive/yes? We have to define a threshold $t = 0.4$ for example such as a probability less than t



classify the example as negative and positive otherwise. Recall the classifier will usually have errors due to positive examples being predicted negative and vice versa.

3 Model Evaluation and Validation

To evaluate and validate the model, we need to get back to the notion of loss function and study the types of errors that can occur while running the model.

3.1 Type I and Type II errors

Denoting by $\hat{y}$ the predicted value of the outcome and by y the observed value of the outcome, we will need to count the number of times $y \neq \hat{y}$. We identify the following types of errors:

- A type I error is the error that occurs when we predict an example as being positive while it is actually negative. This is a *False Positive (FP)*.
- A type II error is the error that occurs when we predict an example as being negative while it is actually positive. This is a *False Negative (FN)*.

We also have the following cases:

- *True Positive (TP)* corresponds to a correct positive prediction of $\hat{y}$ while the actual value y is also positive.
- *True Negative (TN)* corresponds to a correct negative prediction of $\hat{y}$ while the actual value y is also negative.

So, True Positives/Negatives are the ones that are actually positive/negative, and the model has also predicted a positive/negative outcome for them. False Positives are false successes. These are actually failures, but the model is predicting them as successes. False Negatives are actually successes, but the model predicts them as failures.

3.2 Confusion Matrix

The result of how many correct and incorrect predictions were made can be summarized using what is called a *confusion matrix*. A confusion matrix is just a tabular representation to state the number of TPs, TNs, FPs, and FNs. It can be produced in a similar way as the contingency table discussed earlier. In our example of finding if the outcome is positive or negative, we may get the following confusion matrix for example:

$\hat{y}$ / y	Positive	Negative
Positive	723 (TP)	15 (FP)
Negative	72 (FN)	409 (TN)

Let us state some totals in terms of these categories:

- The total number of actual positive = TP+FN
- The total number of actual negative = TN+FP
- The total number of correct predictions = TP+TN
- The total number of incorrect predictions = FP+FN

Performance measures include the following:

Accuracy: The misclassification error for n data points is defined as:

$$E = \frac{FP + FN}{n}$$

and the accuracy of the classifier is then:

$$A = 1 - E$$

TPR (True Positive Rate) : TPR or sensitivity or Recall is the proportion of the positive outcomes that are identified as such (as positives) by the model:

$$TPR = TP/(TP + FN)$$

This may indicate how many relevant items are selected depending on the context.

TNR (True Negative Rate) : TNR or specificity is the proportion of the negative outcomes that are identified as such (as negatives) by the model:

$$TNR = TN/(TN + FP)$$

TPR wards off against False Positive, while the TNR does the same against False Negative. A perfect model will be 100% TPR and also 100% TNR.

Similarly, we define the followings:

FPR (False Positive Rate) : FPR is the proportion of the positive outcomes that are identified as negative by the model:

$$FPR = FP/(FP + TN)$$

FNR (False Negative Rate) : TNR is the proportion of the negative outcomes that are identified as positive by the model:

$$FNR = FN/(FN + TP)$$

Precision : Precision is the proportion of the true positive outcomes that are identified as positive by the model:

$$Precision = TP/(FP + TP)$$

Precision indicates how many selected items are relevant depending on the context.

F1-score: F1-score combines Precision and Recall (TPR) in a single number as follows:

$$F1 - score = 2\frac{Precision \times Recall}{Precision + Recall}$$

F1-score reaches its best value at 1 (perfect precision and recall) and worst at 0.

It is amazing how many performance metrics we can use to evaluate this type of model. But which one to choose? In machine learning we use accuracy just because it is one number that you can directly compare across algorithms. You need a single measure of quality to compare algorithms. Once you have 2 measures of quality you cannot directly make a comparison, because what if one algorithm is better according to one quality measure but not to another one? Then you cannot compare them. But this only works when errors for the positive class count equally to errors for the negative class. This does not work when the data are imbalanced, but anyway, that is what we do. Doctors want to know how many of the positives they got right and how many of the negatives they got right. That makes sense. They want to look at both of them. If you are in information retrieval, then precision and recall make sense. Let's say you are judging the quality of a search engine, say Bing for instance. You might care about the precision. Again precision is: Of the web pages the engine returned, how many really were relevant? That's precision. And recall is the fraction of relevant webpages that the search engine returned. And then they use F1-score so it's a single measure and they can compare the quality of different search engines in an easy way.

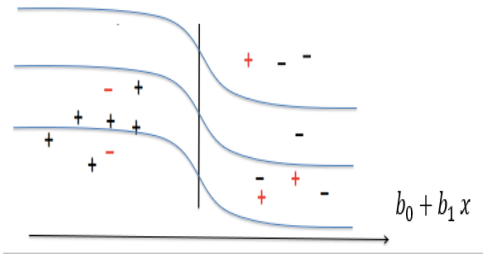
3.3 ROC Curves

An *ROC curve* is a plot of True Positive Rate vs False Positive Rate. As we saw earlier, the number of positive and negative outcomes change as we change the threshold of probability values to classify a probability value as a positive or negative outcome. Thus, the TPR and FPR will change as well. A ROC curve has the following important properties:

- Any increase in Sensitivity will decrease the Specificity
- The closer the curve is to the left and upper border of the quadrant, the better the model prediction
- The closer the curve is to the diagonal line, the worse the model prediction is
- The larger the area under the curve, the better the prediction

To plot a ROC curve, we proceed as follows:

1. Define several probability thresholds and calculate TPR and FPR for each threshold.
2. Plot TPR and FPR points obtained in this way.



The ROC curve lies above the diagonal line and, hence, the model is a better predictor than a random guess. However, there can be many ROC curves lying above the diagonal line. How to determine one ROC curve is better than the other? This is determined by calculating the area enclosed under the ROC curves. The more the area enclosed by the ROC curve, the better it is. The area under the curve can lie between 0 and 1. The closer it is to 1, the better it is. We will see in the Lab how to produce a ROC and calculate the area by using the built-in methods in `scikit-learn`.

3.4 Cross validation

Cross validation is required to deal with an issue common with the predictive models. The models are developed by finding the best fit based on a training dataset. This leads to a problem called overfitting, wherein the model fits the given (training) data very well, but doesn't reproduce the good fitting with some other (testing) dataset. This problem is more severe in the case of datasets with less observations. Splitting up a dataset in the training and testing dataset is the simplest way to do cross validation.

The most popular way to perform a cross validation is using something called as k-fold cross validation. It is carried out as follows:

1. Divide the data set into k partitions.
2. One partition is used as the test set, while the other $k-1$ partitions together are used as the training set.
3. The process in (2) is repeated k times with a different partition as the testing dataset and the rest of them as the training dataset in each iteration.
4. For each iteration, the model accuracy is calculated and averaged out over the iterations. The averaged value is the value of the model accuracy.
5. If the accuracy of the model does not vary much and the average accuracy remains closer to the accuracy numbers calculated for the model before, then it can be confirmed that the model generalizes well.

$K = 10$ is generally the norm, but can be changed according to the situation. We will use built-in methods from `scikit-learn` to perform cross validation in the Lab.