Big Data & Predictive Analytics
Lecture notes on Regression

School of Informatics
University of Leicester

1 Introduction

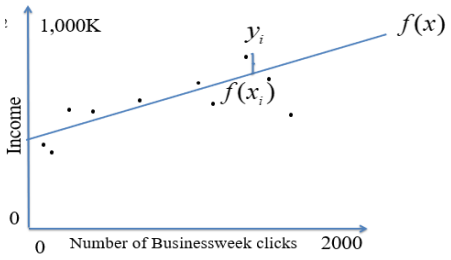
Building up a predictive model comes after carrying out the descriptive and exploratory analyses of the data and gaining a deeper understanding of the data. At the stage, we should have a clear idea of what are the outcome (output) variables if there are and what are the input variables or predictors. This should enable to think of the kind of algorithms we need:

- **supervised:** algorithms from supervised learning if the output variables are well defined.
- **unsupervised:** algorithms from unsupervised learning if no specific output variables are identified but instead one would like to cluster the data in a certain way to identify interesting patterns.
- **semi-supervised:** algorithms when the outcomes are found after an intermediare process that involves clustering.

Examples of supervised algorithms include linear regression (fitting a line into the data) or logistic regression concerns mainly a classification problem (if the output variable is a categorical variable). In a regression problem, we want to predict the output while in a classification problem we want to know if the outputs falls into a specific category. Examples of unsupervised algorithms include k-means or hierarchical clustering. This task involves an understanding of the mathematics behind the chosen model: How do we create the equation from data and what are the assumptions made for the model.

To list the most important steps of a predictive model construction, note the followings:

- specify the modeling objective (what behaviour do we need to capture).
- specify the relevant predictor variables (what predictors to include or exclude)
- decide on the type of model to employ (linear using minimum least squares, decision tree, or an ensemble classifier)



- run the obtained model and inspect the statistics and standard measures of performance to decide if the model needs further tweaking or not.

Naturally, model-building is a interactive process and we may even needs to change the objective or the type of model in order to get a useful model. Note that setting up a *good question* (answerable with the given data is essential in model building).

2 Model building - regression

Regression is supervised learning algorithm used to predict real-valued outcomes. For example if we want to know how many customers will arrive at our website next week; how many TVs will we sell next year; or what is likely to be someone's income from their click through information.

To answer such a question, we need to identify the *outcome* and the *predictors*.

- The *outcome* y is the value we would like to predict, for example the income of a person.
- The *predictors* are a set of features X that may impact the value of the outcome.

In a given dataset, the outcome and predictors are columns of that dataset and a row represents an instance of the data. Given a dataset, how do we build up a model $y = f(x)$ enabling us to predict the value y_i given an instance x_i? Ideally, we would like the model to be as accurate as possible by minimising the difference between the predicted value and the observed value. This is carried out by minimising the SSE (error sum squared) as defined

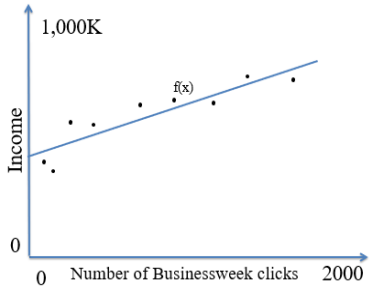
$$\sum_{i=1}^{n}(y_i - f(x_i))^2$$

for n data points. Note that the SSE is differentiable; this makes easier the optimisation as we can rely upon standard convex optimisation algorithms such as the OLS (ordinary least squared) method if the case wherein the model f is linear.

2.1 Simple Linear Regression

In the case where the model f is linear with a single feature x, we can write

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



The problem is then to find the coefficients b_0, b_1 the error sum squared (SSE)

$$\text{SSE}(f) = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

For example, assuming the linear model

$$\begin{aligned} f(x) &= \text{function(Number of Businessweek clicks)} \\ &= 5 * \text{Number of Businessweek clicks} + 100 \end{aligned}$$

was built up from a given dataset on people's income from the number of Businessweek clicks, we get a kind of graph as follows:

This model gives everyone a baseline of 100K just for existing and then estimate that for each click they make on the Businessweek website, they are 5K richer. Obviously this an unrealistic model since it predicts that anyone who spends all of their time on Businessweek makes 100K but this is just an example.

You can also choose a polynomial to fit the data as follows:

$$y_i = f(x_i) = b_0 + b_1 x_i + b_2 x_i^2 + b_3 x_i^3 + \ldots$$

The main issue with this is *over-fitting*, which we will explore later on.

2.2 Multiple Linear Regression

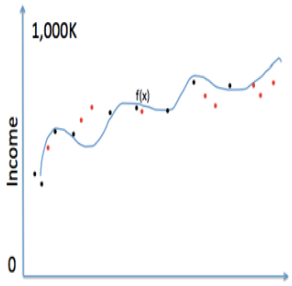
In the case of multiple linear regression, we do consider more than one predictor. For example, we may consider that a person's income depends not only on the number of Businessweek clicks but also the number of visits to airlines etc.

$$\begin{aligned} f(x) = \quad & 3 * \text{Number of visits to upscale furniture websites} \\ & +10 * \text{Number of Businessweek clicks} \\ & +100 * \text{Number of distinct people emailed per day} \\ & +2 * \text{Number of purchases of over 5K within the last month} \\ & +10 * \text{Number of visits to airlines} \end{aligned}$$

A more general form of a multiple linear model is

$$y_i = f(x_i) = b_0 + b_1 x_{i,1} + b_2 x_{i,2} + \ldots + b_m x_{i,m}$$

Note that if we have too many predictors, the optimisation problem gets harder and we may run into curse of dimensionality issues in the sense that a solution can take a long time



to be found. However, we really should put what we think are all of the potentially important predictors; this should be often informed by the exploratory data analysis.

Sometimes a straight line may not be the best fit for the data and we may consider curve lines by adding

- **polynomials**; for example by considering the variables $age, age^2, age^3, \ldots$
- **indicator variables**; for example a variable that has the value 1 if $age < 65$ and 0 otherwise.

If you did put polynomials in there it would allow the function to be kind of curvy and interesting but the problem may be *overfitting*. Overfitting is when the model corresponds too closely or exactly to a particular set of data but fail to fit additional data or predict future observations as good as it did on the initial data.

3 Evaluating Regression Models

To evaluate a linear regression model, say

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

we need to address the following questions:

1. Are the estimated parameters statistically significant?
2. How much variation in the output variable can be explained by the predictor variables?
3. How accurate is the model? Does the accuracy remain the same when we use similar types of data?

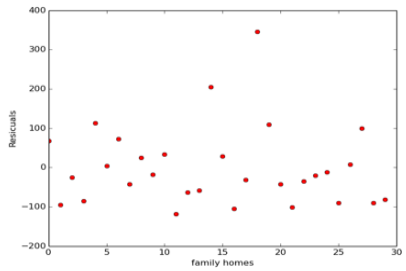
To answer Question 1), we need to use Hypothesis testing. In this case, we can calculate confidence intervals or p-values from within Python for example. In here, the null hypothesis is

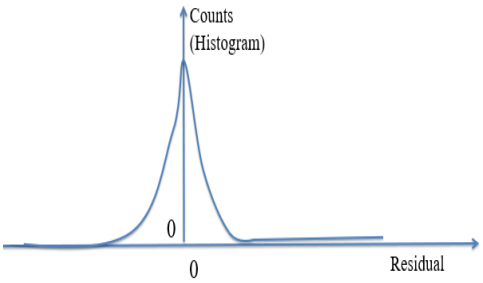
$$H_0 : b_0 = b_1 = 0.$$

Small p-values will indicate that b_0, b_1 are non zeros and statistically significant. We can therefore reject the null hypothesis.

To answer Question 2, we need to calculate the coefficient of determination r^2

$$r^2 = \frac{\text{regression sum of squares}}{\text{total sum of squares}} = \frac{SSR}{SST}$$





wherein the Regression Sum of Squares (SSR) is defined as

$$\text{SSR} = \text{Sum(Predicted Y value–Mean Y value)}^2$$

and the Total Sum of Squares (SST) is defined as

$$\text{SST} = \text{Sum(Observed Y value–Mean Y value)}^2$$

The coefficient of determination can be interpreted as follows: If $r^2 = 0.7498$, then this means that 75% of the variation in the output can be explained by the variability in the predictor. This shows a strong positive linear relationship between the two variables, because the use of a regression model has reduced the variability in predicting the output by 75%. Only 25% of the sample variability in the output can be explained by factors other than what is accounted for by the linear regression model that uses the predictors.

As we have seen about correlation, the coefficient of correlation r is a measure of the strength of the linear relationship between two variables. The values of this coefficient vary from –1, which indicates perfect negative correlation, to +1, which indicates perfect positive correlation. The sign of the correlation coefficient r is the same as the sign of the slope in simple linear regression.

To answer Question 3, we need to start by analysing the residuals $y_i - f(x_i)$. We expect the residuals to be as close to zero as possible with no apparent pattern. Note that we can use the residual graphs to spot and remove outliers by removing any point away from zero using an appropriate threshold.

More importantly, if we plot the histogram of the residuals, we should see that the residuals approximately follow a normal distribution. If the distribution of the residuals is not normal (Gaussian), then the regression model is not valid. We need to explore another model. As for the consistency of the model or to mitigate the issues of over-fitting, we need to **split the data into a training set and a test set**. This way can build a model using the training

set, evaluate it and test it with the test set. The model should produce similar performances on both the training and test sets.

4 Using Regression Models with Python

The practical use of regression models within Python is better understood by doing the corresponding Lab and understanding the Python code used. In these notes, we will only mention the key methods Python offers to carry out such a task.

4.1 Main packages

We can use

1. the `statsmodel.formula.api` package

```
  import statsmodels.formula.api as smf
  model=smf.ols(formula='y~x',data=df).fit()
  model.params #gives model parameters
model.summary() #generates statistics
```

   In this case we should try to make sense of the generated statistics and carry out some analysis in order to improve the model's performance.

   - Select some variables and discard others for the model depending on their significance into the model.
   - Assess the relationship between the predictor and output variable and check whether a predictor variable is significant in the model or not by looking at the p-values for example.
   - Calculate the error in the values predicted by the selected model and see if the error is acceptable.

2. or the `scikit-learn` package

```
  from sklearn.linear_model import LinearRegression
lm = LinearRegression()
  model=lm.fit(x,y)
print lm.intercept_
  print lm.coef_
```

   An advantage of using `scikit-learn` is that we can use RFE (Recursive Features Elimination) to select the most relevant predictors to the outcome. We also have access to SVM (Support Vector Machines) enabling us to carry out another type of regression thanks to SVR (support vector machine regressor), see the corresponding Lab and the online documentation for more details.

```
  from sklearn.feature_selection import RFE
  from sklearn.svm import SVR
  feature_cols = ['TV', 'Radio', 'Newspaper']
  X = df[feature_cols]
  Y = df['Sales']
  estimator = SVR(kernel="linear")
  selector = RFE(estimator,2,step=1) #2 predictors desired
  selector = selector.fit(X, Y)
selector.ranking_ #produces desired variables with rank 1.
```

	sex_female	sex_male
0	1	0
1	0	1
2	1	0
3	0	1
4	1	0

We use the methods named RFE and SVR in-built in `scikit-learn`. We indicate that we want to estimate a linear model and the number of predictors in the model should be two. All the selected variables will have a ranking of 1 while the subsequent ones will be ranked in descending order of their significance. A variable with rank 2 will be more significant to the model than the one with a rank of 3 etc.

4.2 Training and Testing data split

To split the data into a training set and testing set, we can use for example the package *sklearn* and write the following code snippet:

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size=0.25)
```

This will split the data into a test set of of size 25% and 75% of the original data will go to the training set. We may build up a specific model $Y = f(X)$ using the training dataset and then use the test dataset in order to predict the outcome Y given X. Overall, model validation is crucial in reaping the benefits of predictive analytics in a data-driven world.

4.3 Handling Categorical Variables

So far, we have assumed that the predictor variables can only be numerical, but we know that often the dataset contains a categorical or qualitative variable that can have a significant impact on the outcome. The question is then how to process these variables in order to include them in the model? We should not assign values such as 0, 1, 2, etc., and then use them in the model, since it will give undue importance to the categories because of the numbers assigned to them. More importantly, it might give a wrong result and will change, as the number assigned to a particular category changes.

Assume a dataset contains the categorical variable `Gender` with two values `male` and `female`.

```
dummy_sex=pd.get_dummies(df['Gender'],prefix='sex')
column_name=df.columns.values.tolist()
column_name.remove('Gender')
df1=df[column_name].join(dummy_sex)
```

The `dummy_sex` variables created will have the values 0 or 1 as follows:

Once created, the categorical variable `Gender` is removed and the dummy variables are attached to the main data frame. This way, we can use them in the model building as we did for the numerical variables, build a model and evaluate its performance in the usual way.