Introduction to Supervised Learning
Linear Regression

AcadView

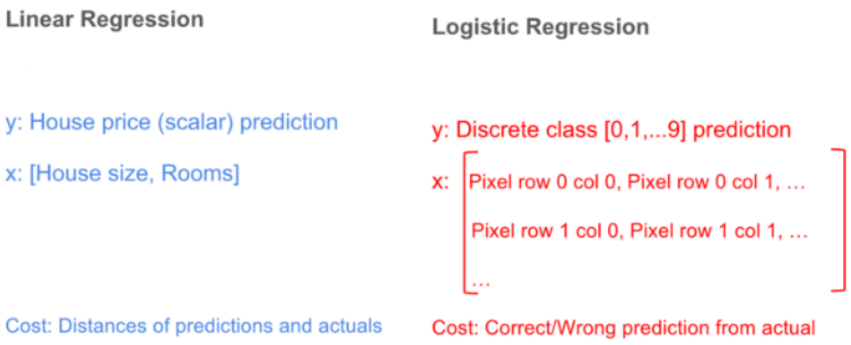
June 4, 2018

---

1 Overview

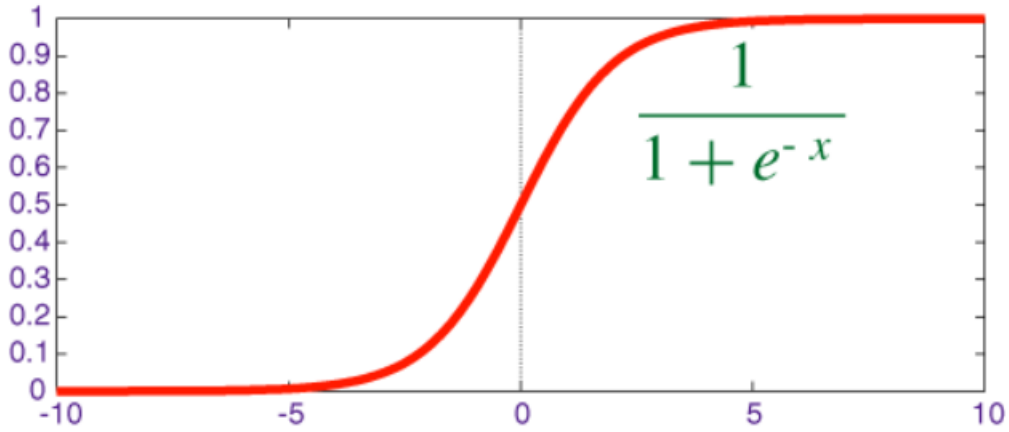
We have learnt how to perform linear regression to predict an outcome of scalar value. However, there are times when we want to classify things rather than predict a value, e.g., given an image of a digit we can to classify it as either $0, 1, 2, , 9$ or given a song we want to classify it as pop, rock, rap, etc. Each of the classification in the set $[0, 1, 2, , 9]$ or [pop, rock, rap, etc.], is known as a class, which in the computer world we represent using a number, e.g., pop = 0, rock = 1, etc. To perform classification, we can employ logistic regression using sklearn.

In this article, we will use logistic regression to classify the image of a digit, as belonging to classes $0, 1, 2, , or, 9$.
The good news is a lot of concepts in linear regression still applies in logistic regression. We can reuse the formula, but with some tweaks. Let's look at both side-by-side for linear and logistic regression:



2 Logistic or Sigmoid curve

There is an awesome function called Sigmoid or Logistic function , we use to get the values between 0 and 1.



This function squashes the value (any value) and gives the value between 0 and 1.
e here is 'exponential function' the value is 2.71828 this is how the value is always between 0 and 1:

2.71828^{+x} = Positive value

$2.71828^{-x} = \frac{1}{\text{Positive value}}$ = Value between (0 and 1)

$\frac{1}{1 + \text{Positive value}}$ or $\frac{1}{1 + \frac{1}{\text{Positive value}}}$ $= \frac{1}{1 + e^{-x}}$

Sigmoid Function

3 Logistic Regression

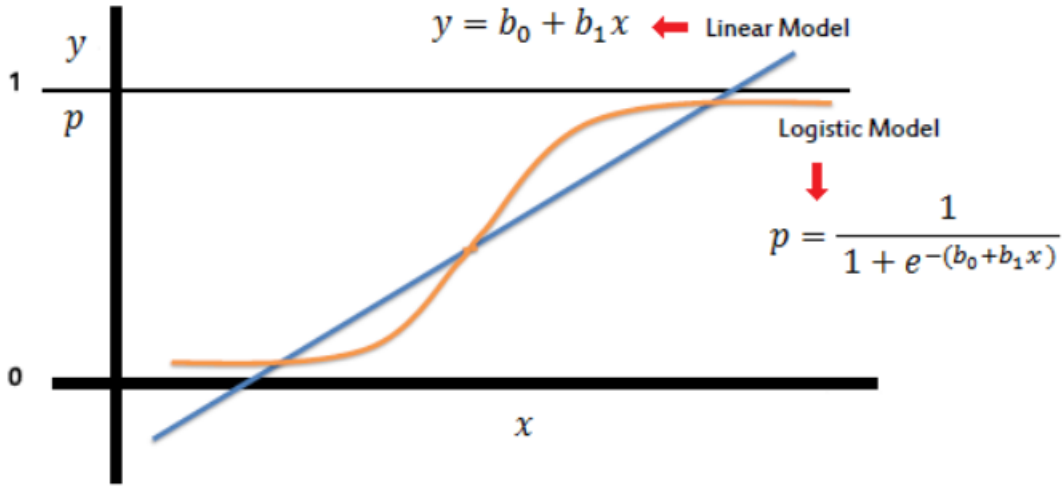
Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

- A linear regression will predict values outside the acceptable range (e.g. predicting

probabilities outside the range 0 to 1)

- Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1. Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the odds of the target variable, rather than the probability. Moreover, the predictors do not have to be normally distributed or have equal variance in each group.



In the logistic regression the constant (b0) moves the curve left and right and the slope (b1) defines the steepness of the curve. By simple transformation, the logistic regression equation can be written in terms of an odds ratio.

$$\frac{p}{1-p} = \exp(b_0 + b_1 x)$$

Finally, taking the natural log of both sides, we can write the equation in terms of log-odds (logit) which is a linear function of the predictors. The coefficient (b1) is the amount the logit (log-odds) changes with a one unit change in x.

$$ln\left(\frac{p}{1-p}\right) = b_0 + b_1 x$$

As mentioned before, logistic regression can handle any number of numerical and/or

categorical variables.

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p)}}$$

3.1 Logistic Regression Predicts Probabilities

Logistic regression models the probability of the default class (e.g. the first class).

For example, if we are modeling peoples sex as male or female from their height, then the first class could be male and the logistic regression model could be written as the probability of male given a persons height, or more formally:

$\mathbf{P(sex = male|height)}$

Written another way, we are modeling the probability that an input (X) belongs to the default class (Y=1), we can write this formally as:

$\mathbf{P(X) = P(Y = 1|X)}$

Were predicting probabilities? I thought logistic regression was a classification algorithm?

Note that the probability prediction must be transformed into a binary values (0 or 1) in order to actually make a probability prediction. More on this later when we talk about making predictions.

Logistic regression is a linear method, but the predictions are transformed using the logistic function. The impact of this is that we can no longer understand the predictions as a linear combination of the inputs as we can with linear regression, for example, continuing on from above, the model can be stated as:

$\mathbf{P(X)} = \frac{\mathbf{1}}{\mathbf{1+e^{-b_0+b_1 x}}}$

We can turn around the above equation as follows (remember we can remove the e from one side by adding a natural logarithm (ln) to the other):

$\mathbf{ln(P(X)/1 - P(X)) = b0 + b1 * X}$

This is useful because we can see that the calculation of the output on the right is linear

again (just like linear regression), and the input on the left is a log of the probability of the default class.

4 Advantages / Disadvantages

It is a widely used technique because it is very efficient, does not require too many computational resources, its highly interpretable, it doesnt require input features to be scaled, it doesnt require any tuning, its easy to regularize, and it outputs well-calibrated predicted probabilities.

Like linear regression, logistic regression does work better when you remove attributes that are unrelated to the output variable as well as attributes that are very similar (correlated) to each other. Therefore Feature Engineering plays an important role in regards to the performance of Logistic and also Linear Regression. Another advantage of Logistic Regression is that it is incredibly easy to implement and very efficient to train. In general ML engineers typically start with a Logistic Regression model as a benchmark and try using more complex algorithms from there on.

Because of its simplicity and the fact that it can be implemented relatively easy and quick, Logistic Regression is also a good baseline that you can use to measure the performance of other more complex Algorithms.

5 Binary classification and Multi-classification

The name itself signifies the key differences between binary and multi-classification. Below examples will give you the clear understanding about these two kinds of classification. Lets first look at the binary classification problem example. Later we will look at the multi-classification problems.

Binary Classification:

- Given the subject and the email text predicting, Email Spam or not.
- Sunny or rainy day prediction, using the weather information.
- Based on the bank customer history, Predicting whether to give the loan or not.

Multi-Classification:

- Given the dimensional information of the object, Identifying the shape of the object.
- Identifying the different kinds of vehicles.
- Based on the color intensities, Predicting the color type.

I hope the above examples given you the clear understanding about these two kinds of classification problems. In case you miss that, Below is the explanation about the two kinds of classification problems in detail.

5.1 Binary Classification Explanation:

In the binary classification task. The idea is to use the training data set and come up with any classification algorithm. In the later phase use the trained classifier to predict the target for the given features. The possible outcome for the target is one of the two different target classes.

If you see the above binary classification problem examples, In all the examples the predicting target is having only 2 possible outcomes. For email spam or not prediction, the possible 2 outcome for the target is email is spam or not spam.

On a final note, binary classification is the task of predicting the target class from two possible outcomes.

5.2 Multi-classification Explanation:

In the multi-classification problem, the idea is to use the training dataset to come up with any classification algorithm. Later use the trained classifier to predict the target out of more than 2 possible outcomes.

If you see the above multi-classification problem examples. In all the examples the predicting target is having more than 2 possible outcomes. For identifying the objects, the target object could be triangle, rectangle, square or any other shape. Likewise other examples too.

On a final note, multi-classification is the task of predicting the target class from more two possible outcomes.

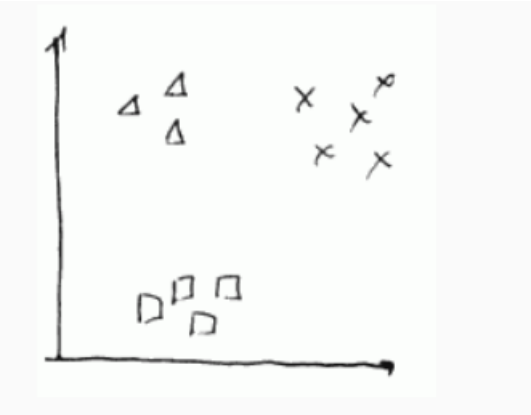
6 One-vs-All Classification

6.1 One vs All Classifier

Suppose we have a classifier for sorting out input data into 3 categories:

- class 1 (Δ)

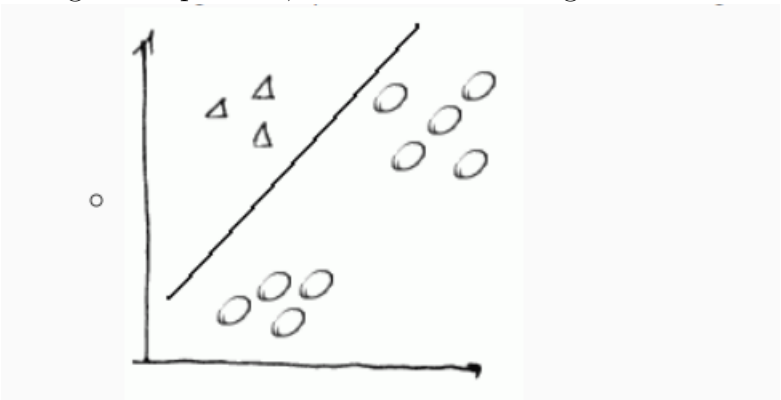
- class 2 ($\square$)
- class 3 ($\times$)



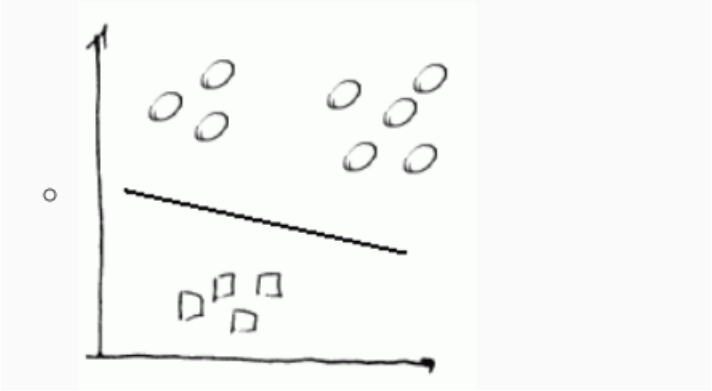
We may turn this problem into 3 binary classification problems (i.e. where we predict only $y \in 0, 1$) to be able to use classifiers such as Logistic Regression.

We take values of one class and turn them into positive examples, and the rest of classes - into negatives

- **Step 1:**
  triangles are positive, and the rest are negative - and we run a classifier on them.

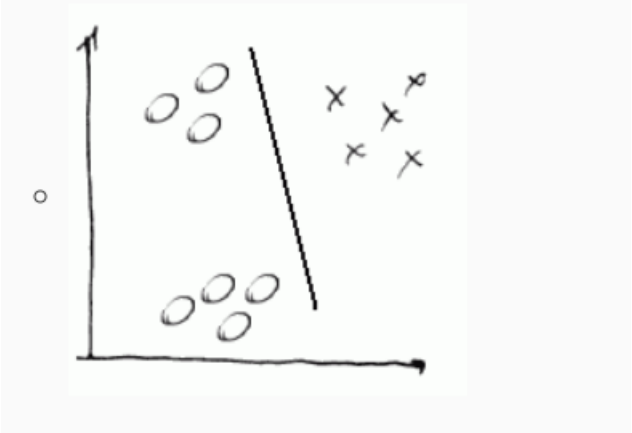
  

  and we calculate $p_1$for it
- **Step 2:** next we do same with squares: make them positive, and the rest - negative



and we calculate p_2:

- **Step 3**: finally, we make $\times s$ as positive and the rest as negative and calculate p_2



So we have fit 3 classifiers:

$p_i = P(y = i|x; parameters), i = 1, 2, 3$

Now, having calculated the vector $P = [p_1, p_2, p_3]$ we just pick up the maximal value i.e. we choose $\mathbf{max_i P_i}$

7 Logistic Regression for multi-class classification using scikit-learn

We shall discuss in details how we use multi-class classification on the digits dataset (popularly known as the MNIST dataset). All the implementation of logistic regression is done using scikit-learn library.

7.1 Loading the Data (Digits Dataset)

The digits dataset is one of datasets scikit-learn comes with that do not require the downloading of any file from some external website. The code below will load the digits dataset.

```
from sklearn.datasets import load_digits
digits = load_digits()
```

Now that you have the dataset loaded you can use the commands below

```
# Print to show there are 1797 images (8 by 8 images for a
dimensionality of 64)
print("Image Data Shape" , digits.data.shape)

# Print to show there are 1797 labels (integers from 0-9)
print("Label Data Shape", digits.target.shape)
```

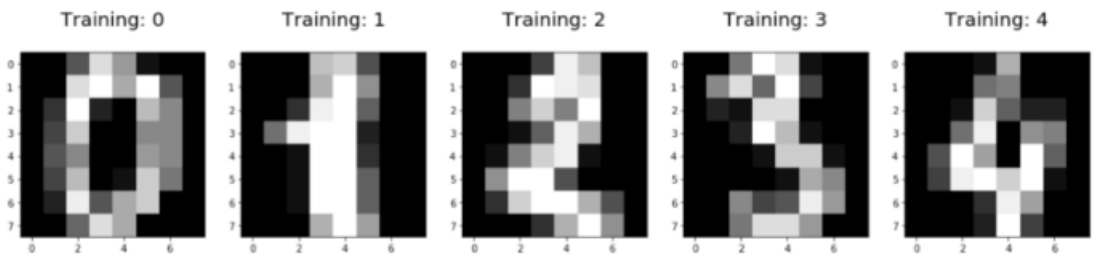
to see that there are 1797 images and 1797 labels in the dataset.

7.2 Showing the Images and the Labels (Digits Dataset)

This section is really just to show what the images and labels look like. It usually helps to visualize your data to see what you are working with.

```
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize=(20,4))
for index, (image, label) in enumerate(zip(digits.data[0:5],
digits.target[0:5])):
 plt.subplot(1, 5, index + 1)
 plt.imshow(np.reshape(image, (8,8)), cmap=plt.cm.gray)
 plt.title('Training: %i\n' % label, fontsize = 20)
```



Visualizing the Images and Labels in our Dataset

7.3 Splitting Data into Training and Test Sets (Digits Dataset)

We make training and test sets to make sure that after we train our classification algorithm, it is able to generalize well to new data.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.25, random_state=0)
```

7.4 Scikit-learn 4-Step Modeling Pattern (Digits Dataset)

- **Step 1:** Import the model you want to use
  In sklearn, all machine learning models are implemented as Python classes

```
from sklearn.linear_model import LogisticRegression
```

- **Step 2:** Make an instance of the Model

```
# all parameters not specified are set to their defaults
logisticRegr = LogisticRegression()
```

- **Step 3:** Training the model on the data, storing the information learned from the data

Model is learning the relationship between digits (x_train) and labels (y_train)

```
logisticRegr.fit(x_train, y_train)
```

- **Step 4:** Predict labels for new data (new images)

  Uses the information the model learned during the model training process

```
# Returns a NumPy Array
# Predict for One Observation (image)
logisticRegr.predict(x_test[0].reshape(1,-1))
```

Predict for Multiple Observations (images) at Once:

```
logisticRegr.predict(x_test[0:10])
```

Make predictions on entire test data:

```
predictions = logisticRegr.predict(x_test)
```

7.5 Measuring Model Performance (Digits Dataset)

While there are other ways of measuring model performance, we are going to keep this simple and use accuracy as our metric.
To do this are going to see how the model performs on the new data (test set)

accuracy is defined as:

(fraction of correct predictions): correct predictions / total number of data points

```
# Use score method to get accuracy of model
score = logisticRegr.score(x_test, y_test)
print(score)
```

Our accuracy was **95.3%**.

Note: We will study various ways of measuring performance of the models that we create in the next class section.