Northeastern University
College of Professional Studies

Module 2 Assignment

Shivam Chauhan

January 23, 2020

Class: ALY 6020 – Predictive Analytics

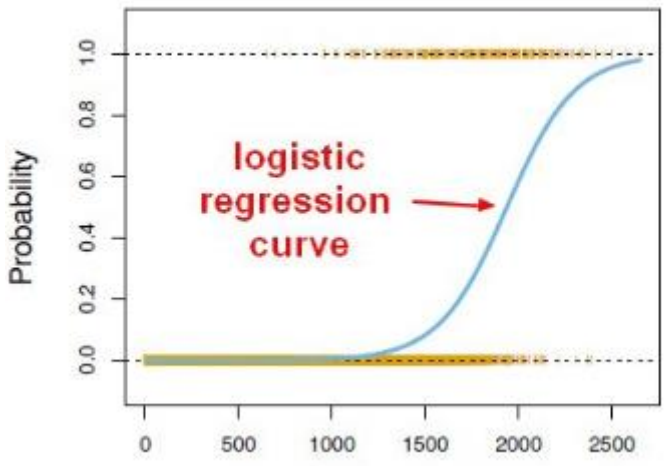
Professor: Dr. Marco Montes de Oca

Topic: Fashion items recognition with Logistic Regression.

Introduction

Logistic regression is a statistical machine learning algorithm that classifies the data into different categories and tries makes a logarithmic line that distinguishes between them (Medium, 2020). Logistic regression produces a logistic curve, that means it gives values from 0 to 1. Logistic regression is like linear regression, but the graph created is from using the logarithm of the odds of the target variable, rather than the probability. The main goal of creating a logistic regression is to predict the target variable based on its feature. Logistic regression models the data using the sigmoid function.

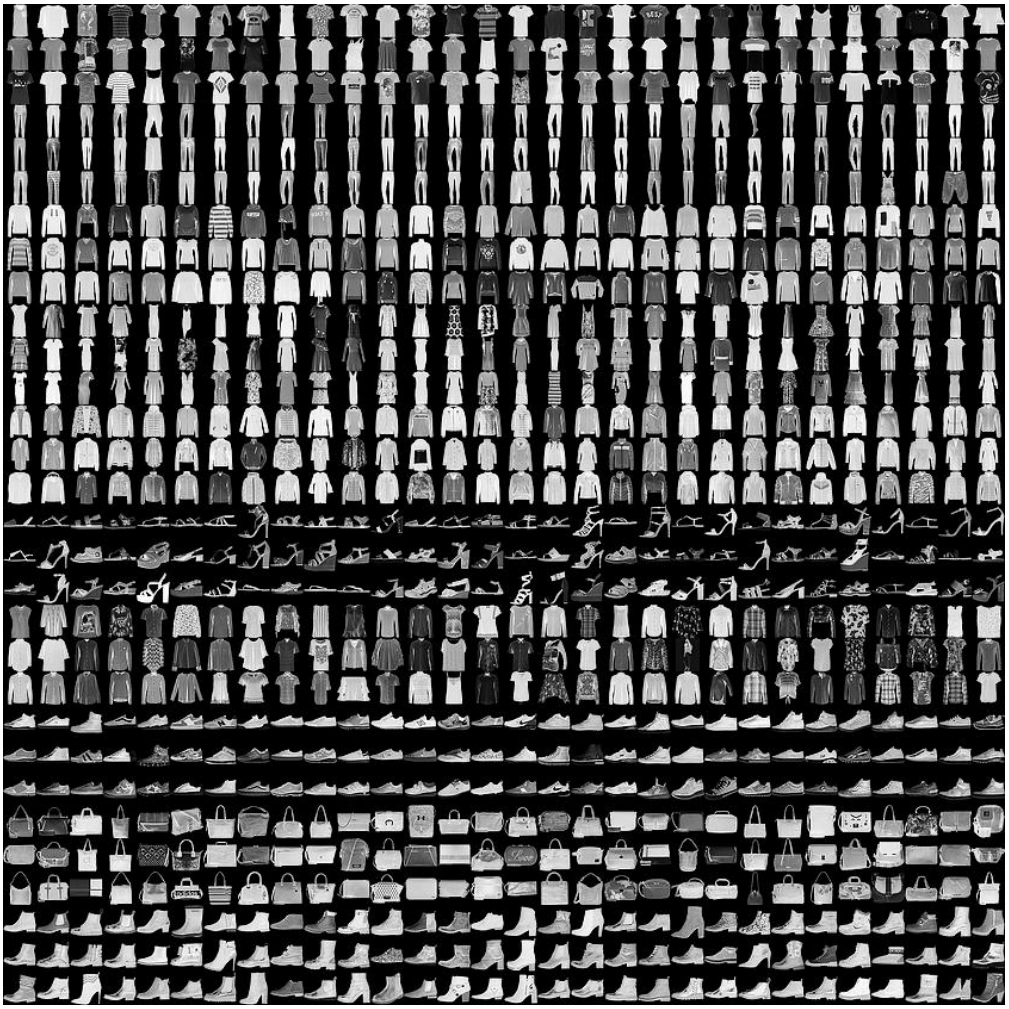
$$g(z) = \frac{1}{1+e^{-z}}$$



The main objective of this assignment is to make a predictive model that recognize the fashion items available to us as the data of images which is encoded as a row of 784 integer values between 0 and 255 indicating the brightness of each pixel. Data files contains thousands of 28x28 grayscale images. The label associated with each image is encoded as an integer value between 0 and 9.

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Sample Data:



Data Preparation

```
#library used
import pandas as pd
import os
import numpy as np


# defining the column names
name=[]
for i in range(785):
    name.append(i)

#Loading the data and looking into the data type
df = pd.read_csv("fashion_train.csv" , names = name)
df_test = pd.read_csv("fashion_test.csv", names = name)

#df columns
```

Column names are added because data was available without the column names. Columns names are integer 0 to 784.

```
#creating traing data set for 10 different models for each fashion item category
dfs = {}
for i in range(10):
    dfs[i] = df.copy(deep=True)

#creating test data set for 10 different models for each fashion item category
dfs_test = {}
for i in range(10):
    dfs_test[i] = df_test.copy(deep=True)
```

```
#preparing the train data for logistic regression
for i in range(10):
    if i == 0:
        dfs[0][0] = dfs[0][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [1,0,0,0,0,0,0,0,0,0])
    if i == 1:
        dfs[1][0] = dfs[1][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,1,0,0,0,0,0,0,0,0])
    if i == 2:
        dfs[2][0] = dfs[2][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,1,0,0,0,0,0,0,0])
    if i == 3:
        dfs[3][0] = dfs[3][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,1,0,0,0,0,0,0])
    if i == 4:
        dfs[4][0] = dfs[4][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,0,1,0,0,0,0,0])
    if i == 5:
        dfs[5][0] = dfs[5][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,0,0,1,0,0,0,0])
    if i == 6:
        dfs[6][0] = dfs[6][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,0,0,0,1,0,0,0])
    if i == 7:
        dfs[7][0] = dfs[7][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,0,0,0,0,1,0,0])
    if i == 8:
        dfs[8][0] = dfs[8][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,0,0,0,0,0,1,0])
    if i == 9:
        dfs[9][0] = dfs[9][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,0,0,0,0,0,0,1])
```

Relabelling each row with a 1 if it corresponds to the item for training data, and 0 for the rest.

```
#preparing the test data for logistic regression
for i in range(10):
    if i == 0:
        dfs_test[0][0] = dfs_test[0][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [1,0,0,0,0,0,0,0,0,0])
    if i == 1:
        dfs_test[1][0] = dfs_test[1][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,1,0,0,0,0,0,0,0,0])
    if i == 2:
        dfs_test[2][0] = dfs_test[2][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,1,0,0,0,0,0,0,0])
    if i == 3:
        dfs_test[3][0] = dfs_test[3][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,1,0,0,0,0,0,0])
    if i == 4:
        dfs_test[4][0] = dfs_test[4][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,0,1,0,0,0,0,0])
    if i == 5:
        dfs_test[5][0] = dfs_test[5][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,0,0,1,0,0,0,0])
    if i == 6:
        dfs_test[6][0] = dfs_test[6][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,0,0,0,1,0,0,0])
    if i == 7:
        dfs_test[7][0] = dfs_test[7][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,0,0,0,0,1,0,0])
    if i == 8:
        dfs_test[8][0] = dfs_test[8][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,0,0,0,0,0,1,0])
    if i == 9:
        dfs_test[9][0] = dfs_test[9][0].replace([0, 1, 2, 3,4,5,6,7,8,9], [0,0,0,0,0,0,0,0,0,1])
```

Relabelling each row with a 1 if it corresponds to the item for test data, and 0 for the rest.

```
#creating empty dictionary that can store test and train data.
x_train = {}
x_test =  {}
y_train = {}
y_test = {}

#Splitting the data into test and train for each item category and storing it in dictionary
for i in range(10):
    x_train[i] = np.array(dfs[i].iloc[:,1:])
    x_test[i] =  np.array(dfs_test[i].iloc[:,1:])
    y_train[i] = np.array(dfs[i][0])
    y_test[i] = np.array(dfs_test[i][0])
```

Test and train data are created for each 10 models which classify each image into the respective clothing item categories. We have our train data with 60000 rows/images and 785 columns including the label or the clothing categories. And we have 10000 rows/images with 785 columns including the label.

The data is split into:

1. X_train: 60000 images pixel data only without the label column.
2. Y_train: 60000 images label data which is the clothing categories.
3. X_test: 10000 images pixel data only without the label column.
4. Y_test: 10000 images label data which is the clothing categories.

Logistic Model

```
#Loading the Logistic Regression model from sklearn
from sklearn.linear_model import LogisticRegression

#defing the logistic model as log_m
log_m = LogisticRegression(solver = "lbfgs")

#Fitting the log_m model for every item categories and storing it in dictionary
log_models = {}
y_pred = {}
y_pred_prob = {}
for i in range(10):
    #fitting the model for respective train and test data
    log_models[i] = log_m.fit(x_train[i],y_train[i])
    y_pred[i] = log_m.predict(x_test[i])            #adding the predicted reults in dictionary y_pre
    y_pred_prob[i] = log_m.predict_proba(x_test[i]) #adding the probability of each row/item
```

The log_m logistic model was created with solver lbfgs parameter which is the default solver to optimize smooth functions of many variables.

```
"""
Creating a dataframe that contains the probabilty of
an image being a particular clothing item.
These probabilities are stored as unrelated_prob.
"""
prob = {}
for i in range(10):
    prob[i] = y_pred_prob[i][:,1]

unrelated_prob = pd.DataFrame(prob)
```

Index	0	1	2	3	4	5	6	7	8	9
0	1.12847e-05	2.40845e-09	0.00114563	3.44987e-07	5.55069e-06	0.0357725	0.00313049	0.0103037	0.018103	0.657888
1	0.000130764	1.06171e-10	0.855447	5.38222e-08	0.0555576	1.85423e-20	0.0335784	8.05437e-34	9.01025e-06	1.01887e-29
2	0.00196761	0.999989	0.000715872	0.000134609	0.00111848	2.36537e-21	4.88784e-05	2.48054e-18	1.68095e-05	1.50694e-10
3	0.000682767	0.999964	0.00251635	0.00593906	0.00575085	2.38099e-17	7.72179e-05	8.55932e-18	2.53506e-06	1.80505e-07
4	0.127516	5.93896e-07	0.0393995	0.00204747	0.0859768	8.80776e-13	0.260347	1.81174e-26	0.000135808	2.4252e-16

Image showing sample of data frame unrealted_prob. This shows 10 different probability generated by 10 different models that classify image being a particular clothing item.

```
""" defining the softmax function """
def softmax(x):
    """Compute softmax values for each sets of scores in x."""
    return np.exp(x) / np.sum(np.exp(x), axis=0)
```

SoftMax function is a function that takes as input a vector of K real numbers and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers (En.wikipedia.org,2020). Sum of all probability in the distribution is always 1.

The standard (unit) softmax function $\sigma : \mathbb{R}^K \to \mathbb{R}^K$ is defined by the formula

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

```
""" Transforming the unrealted_prob into a probability distribution
which is stored in a dictionary soft_m using softmax function
"""
soft_m = {}
for i in range(len(unrelated_prob)):
    soft_m[i] = softmax(unrelated_prob.iloc[i,:])


""" Creating a dataframe of probability distribution of each image/row """
prob_dist = pd.DataFrame.from_dict(soft_m, orient='index')
```

Sample of prob_dist data frame:

Index	0	1	2	3	4	5	6	7	8	9
0	0.0909097	0.0909086	0.0910128	0.0909087	0.0909091	0.0942195	0.0911937	0.0918502	0.0925693	0.175518
1	0.0873947	0.0873832	0.205563	0.0873832	0.0923754	0.0873832	0.0903672	0.0873832	0.087384	0.0873832
2	0.0854758	0.231888	0.0853689	0.0853193	0.0854033	0.0853078	0.085312	0.0853078	0.0853092	0.0853078
3	0.0852865	0.231666	0.0854431	0.085736	0.0857199	0.0852283	0.0852349	0.0852283	0.0852285	0.0852283
4	0.10752	0.0946474	0.0984509	0.0948413	0.103145	0.0946474	0.122794	0.0946474	0.0946602	0.0946474

Image showing the probability distribution of each image being a clothing item as per 10 different clothing items.

```
"""
This loop scan teh row of prob_dist dataframe and selects the
maximum probabity and append the column number into the dictionary
predicted_cat
"""
predicted_cat = {}
for i in range(len(prob_dist)):
    predicted_cat[i] = prob_dist.iloc[i,:].idxmax(axis=1)
```

This above code fetches the column number of the highest probability in the row which indicate that the odds of being the item for a category is highest by the predictive logistic model.

```
""" converting  predicted_cat dictionary into a data frame predicted_categories """
predicted_categories = pd.DataFrame.from_dict(predicted_cat, orient='index')
```

```
""" Selecting the label column from the original dataset which is actual image categories
 that can be compared with the predicted categories. """
actual_categories = df_test[0]
```

```
""" Renaming the columns after merging actual and predicted labels """
data_for_matrix = data_for_matrix.rename(columns = {"0_x":"Actual", "0_y":'Predicted'})
```

```
""" Creating a matrix with the counts of the correct and incorrect classifications. """
matrix = data_for_matrix.groupby(['Actual','Predicted']).size().unstack(fill_value=0)
print(matrix)
```

```
""" renaming the columns and row index as per the clothing items labels """
matrix = matrix.rename(columns = {0:"T-shirt/top",1 :'Trouser',2:"Pullover",3:"Dress",4:"Coat",5:"Sandal",6:"Shirt",7:"Sr
                    index = {0:"T-shirt/top",1 :'Trouser',2:"Pullover",3:"Dress",4:"Coat",5:"Sandal",6:"Shirt",7:"Snea
```

Here columns represent the actual item category and rows represent the predicted items.

Actual

Predicted

Index	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
T-shirt/top	812	7	14	48	11	0	93	0	14	1
Trouser	3	954	8	25	3	0	3	1	3	0
Pullover	25	4	724	12	152	0	72	0	11	0
Dress	24	21	14	859	38	0	37	0	7	0
Coat	0	1	111	40	778	1	62	0	7	0
Sandal	2	1	0	0	0	891	0	54	15	37
Shirt	143	5	127	43	121	0	528	0	33	0
Sneaker	0	0	0	0	0	28	0	945	0	27
Bag	3	2	5	9	5	6	15	4	951	0
Ankle boot	0	0	0	0	0	16	1	41	0	942

Matrix showing the counts of the correct and incorrect classifications.

We can observe that models were able to predict their clothing categories accurately using the 784 columns of pixels data. Almost all the categories were accurately predicted, however we can see category shirt was predicted as T-shirt, Pullover and coat due to which the error rate of predicting shirt is very high. This makes sense because the shape of the shirt is very similar to these categories and because we have built our model based on the brightness of each pixel, model might predict which has similar shape.

Trouser and bags items category have maximum accuracy rate.

Performance

```
""" listing all the values along the diagonals of the matrix """
dia_sum = []
for i in range(10):
    s = matrix.iloc[i][i]
    dia_sum.append(s)

""" Calculating the accuracy of the 10 models combine as percentage """
accuracy_model = (sum(dia_sum)/10000)*100
accuracy model
```

The accuracy of the model is **83.84** %. This means this model predicted 83.84% of the items correctly for this test data set.

Conclusion

In conclusion the logistic regression is a powerful machine learning algorithm which can be used as a classifier. From this assignment we have observed how multiple logistic regression models can be used to predict 10 different categories of the clothing items. This model has accuracy of 83.84 and this can be compared with the CNN model which is for image classification to know the difference in the accuracy. Moreover, this model works well for classifying the trousers and bags with a very high sensitivity rate.

Reference

All about Logistic regression. (2018). Medium. Retrieved 23 January 2020, from https://towardsdatascience.com/logistic-regression-b0af09cdb8ad

SoftMax function. (2020). En.wikipedia.org. Retrieved 23 January 2020, from https://en.wikipedia.org/wiki/Softmax_function

Attachment

Week 2 assignment _ Logestic regressior