Logistic Regression

Ishita Jaju
16BCE1059

1 The algorithm

Logistic Regression is based on the Gradient Descent algorithm. Gradient descent is an algortihm to minimize the cost function of a particular dataset and find the required weights of the hypothesis.
Gradient descent works on simulataneously updating the value for every weight like so:

$$\text{Repeat until convergence } \{$$

$$\theta_j \leftarrow \theta_j - \alpha \tfrac{\partial}{\partial \theta_j} J(\theta)$$

$$\}$$

where θ are the weights of the hypothesis, J is the cost function to minimize and α is the learning rate.

For logistic regression, the hypothesis function is the sigmoid function, with the input being g($\theta^t x$) (where g is the sigmoid function). So the hypothesis would be:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

where θ is the weight vector and x is the attribute matrix.

1.1 Logistic Regression for multiclass classification

The above equation of the hypothesis gives an output between 0 and 1. So classes are defined based on the nearest integer between 0 and 1, which is binary classification. For multiclass classification, the logistic regression classifier is trained as $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$. So, on a new input x, we need to make a prediction and pick the class i that maximises the hypothesis: $\max\limits_i h_\theta^{(i)}(x)$

2 Using the seeds dataset

2.1 The dataset

```
headers = ["area","perim","compactness","kernel_length","kernel_width","ass_coeff","kernel_grove_length","class"]
seeds = pd.read_csv("/home/ijaju/Documents/datasets/seeds_dataset.csv",header = None,names=headers)
seeds.tail()
```

	area	perim	compactness	kernel_length	kernel_width	ass_coeff	kernel_grove_length	class
205	12.19	13.20	0.8783	5.137	2.981	3.631	4.870	3
206	11.23	12.88	0.8511	5.140	2.795	4.325	5.003	3
207	13.20	13.66	0.8883	5.236	3.232	8.315	5.056	3
208	11.84	13.21	0.8521	5.175	2.836	3.598	5.044	3
209	12.30	13.34	0.8684	5.243	2.974	5.637	5.063	3

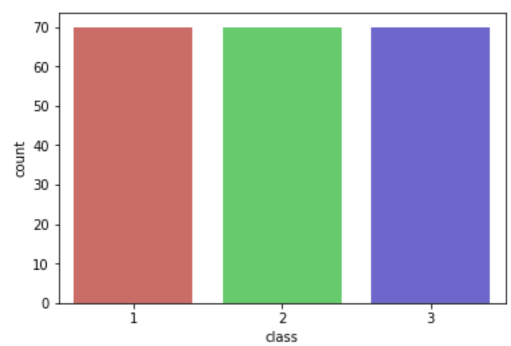
The dataset has attributes like area,kernel grove length etc and the classification is done for 3 species of wheat. The species have 70 entries each and there are no null values in any of the columns.

```
seeds.isnull().sum()
```

```
area                   0
perim                  0
compactness            0
kernel_length          0
kernel_width           0
ass_coeff              0
kernel_grove_length    0
class                  0
dtype: int64
```

```
sb.countplot(x='class',data=seeds, palette='hls')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fefaf9f97b8>
```



2.2 Running the algorithm

On running the algorithm, the accuracy comes out to be 91% for this multiclass dataset.

```
LogReg = LogisticRegression()
LogReg.fit(X_train, y_train)
```

```
y_pred = LogReg.predict(X_test)
```

```
confusion_matrix = confusion_matrix(y_test, y_pred)
confusion_matrix
```

```
array([[17,  3,  2],
       [ 1, 25,  0],
       [ 0,  0, 15]])
```

```
print(classification_report(y_test, y_pred))
```

```
             precision    recall  f1-score   support

          1       0.94      0.77      0.85        22
          2       0.89      0.96      0.93        26
          3       0.88      1.00      0.94        15

avg / total       0.91      0.90      0.90        63
```

3 My dataset - Chronic Kidney Disease

This is a binary classification dataset with the classes being - 'ckd' and 'notckd'.

```
ckd.dtypes
```

```
age               float64
bp                float64
sg                float64
al                float64
su                float64
rbc                object
pc                 object
pcc                object
ba                 object
bgr               float64
bu                float64
sc                float64
sod               float64
pot               float64
hemo              float64
pcv                 int64
wc                  int64
rc                float64
htn                object
dm                 object
cad                object
appet              object
pe                 object
ane                object
classification     object
dtype: object
```

With there being so many 'object' datatypes where the inputs are classifiers, like 'normal' and 'abnormal'; and 'present' and 'notpresent', we need to convert these into numerical inputs like '0' and '1' or '-1' and '1' for the algorithm to work on them. The function below takes care of the same:

```
class_types = ['rbc','pc','pcc','ba','htn','dm','cad','appet','pe','ane']
for i in ckd.columns:
    for j in class_types:
        if i==j:
            e = ckd[j].unique()
            ckd[j] = np.where(ckd[j]==e[0],-1,1)
```

Then after running the algorithm, the accuracy is found to be 97%.

```python
X = ckd.iloc[:,0:23].values
y = ckd.iloc[:,24].values
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3, random_state=25)
LogReg = LogisticRegression()
LogReg.fit(X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```python
y_pred = LogReg.predict(X_test)
```

```python
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

```
0.9791666666666666
```