Statistical Machine Learning - Project 1

Ajay Kannan

The specific algorithmic tasks:

1) Extracting the features and then estimating the parameters for the 2-D normal distribution for each class, using the training data. Note: You will have two distributions, one for each class.

- Finding the row wise means and standard deviations for each in the training set and testing set.
  - Storing the values
  - The functions 'mean' and 'stdev' is used.
  - The output shape is train - (12000, 2) and test - (2000,2) (the two values represent mean and standard deviation)
- Then we find the training parameters.
  - The functions 'avgmean' and 'avgstdev' is used in both values of the mean set and stdev set to **estimate the parameters** for Naive Bayes classifier.
  - These functions are calculated column wise.

<u>train_para</u>

train_para_class0 :

```
[[[0.325607766439909, 0.11337491460878085],
  [0.3200360871033629, 0.08798281005982794]],
```

train_para_class1 :

```
[[0.22290531462585023, 0.05695100874843002],
  [0.333941712027219, 0.05703228654279648]]]
```

2) Use the estimated distributions for doing Naïve Bayes classification on the testing data. Report the classification accuracy for both the classes in the testing set.

- Using the class parameters, find the PDFs (Probability Density Function) of each class for a given testing data. Find which is higher and determine the predicted class.
- Then, match the prediction to the actual value, for getting the accuracy.

```
Accuracy  - 83.15%
```

	precision	recall	f1-score	support
class 0	0.87	0.78	0.82	1000
class 1	0.80	0.88	0.84	1000
accuracy			0.83	2000
macro avg	0.83	0.83	0.83	2000
weighted avg	0.83	0.83	0.83	2000

3) Use the training data to train a Logistic Regression model using gradient ascent.

- Use the row wise means and standard deviations found in the first step.
- For each testing points, calculate the dot product of the weights and data points along with the bias. (W and b are random numbers)
- Apply sigmoid function to predict the Y value. Run this for the entire training dataset.
- Once it is done, update the weights using this formula.
- The run the same thing until the efficiency is reached.

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$\}$

- Run the weights and bias in the testing set for getting the accuracy.

```
Accuracy - 92.15%
```

	precision	recall	f1-score	support
class 0	0.93	0.92	0.92	1000
class 1	0.92	0.93	0.92	1000
accuracy			0.92	2000
macro avg	0.92	0.92	0.92	2000
weighted avg	0.92	0.92	0.92	2000