INDIAN INSTITUTE OF TECHNOLOGY GOA

CS331

[Machine Learning]

Lab Assignment-3

Submitted By Group :-

OmNiSiYaKrMa

I. Perceptron

1) **(Perceptron) Using the make blobs function in sklearn generates a dataset of 100 points with two classes. Write the perceptron algorithm by scratch and show the intermediate hyperplanes generated.**

   Using the make_blobs function a dataset was created with 100 points and 2 classes. The perceptron algorithm (with stochastic descent) was implemented on it and intermediate hyperplanes were also shown.

2) **(Perceptron) Repeat the above exercise with batch gradients instead. Step size might need to be adjusted for convergence.**

   Perceptron is again implemented on the above data and now, algorithm based on batch gradient is used and step size is adjusted for convergence and set to 0.03.

3) **(Perceptron) Using the make circles function in sklearn generates two classes so that they form different concentric circles. Generate 100 points. Create second order features and train the perceptron. Using the contour function shows the final decision boundary in the original two dimensional space.**

   Using the make circles function in sklearn 2 classes of concentric circles were generated with a total of 100 points. Now, second order features (x^2, xy and y^2) were added along with first order features (x and y). And the algorithm was again run on this new data and the final boundary is shown using contours. (As now classes are circles separating boundaries was expected to be more of circular form and that's what we got).

4) **(Perceptron) Repeat the two perceptron examples above (make blobs and make circles) generating 1000 points. We are not interested to see the plots now. We instead want to split the dataset into tests (50%) and train(50%). Report the test accuracy of perceptron in each case.**

SImilar to above 2, again dataset, now of 1000 points were generated for the circular and make_blob case and the both data were splitted into 50% training and 50% testing. Training of perceptrons was done on training and then test data was used for testing accuracy.
In both cases, **accuracy was found to be 100%.**

II. Regression

1) **Implement LMS algorithm for linear regression from scratch. Visualize the learnt house prices on the scatter plot of the input training dataset**

   For the question, the dataset was taken from here. First data was visualized based on two features: number of bedrooms and area of houses independently. Then features were scaled and moved in order to avoid overflow. Then the LMS algorithm(based on full batch descent) was run using house areas as only features and house prices were predicted and visualization was done.

2) **On the full batch gradient descent visualize the contours of J(W) for different values of the learning rate η.**

   Using various values of η, the algorithm was run again and again and corresponding contours as well as convergence of the w vector was shown.

3) **On the stochastic gradient descent visualize the contours of J(W) for different values of the learning rate and batch sizes η.**

   Similar thing as done in the previous part was done for stochastic gradient descent.

4) **Implement locally weighted linear regression as described in Stanford lecture notes.**

For locally weighted linear regression, point x = 1500000 is taken for prediction(point of interest). Using this point, weights were calculated and assigned to each term in summation of loss function and again using corresponding gradient, gradient descent algorithm was run and new solution is plotted along with previously learnt values(the small distinction is clearly visible).

III. Logistic Regression

1) **Using the make blobs function in sklearn generates a dataset of 100 points with two classes. Implement Logistic regression with cross entropy loss.**

   Using the make_blobs function in sklearn, a dataset of 100 points and 2 classes were generated. Logistic regression was implemented with cross entropy loss (or negative log likelihood).

2) **Using the make blobs function in sklearn generates a dataset of 100 points with two classes. Implement Logistic regression with least mean square loss.**

   On the same data, logistic regression is implemented with least mean square loss as loss and corresponding optimal values were printed.

   In both of the above, the step size in gradient descent was kept constant which was decided based on observing the pattern of loss function convergence.

3) **Compare the trajectory of gradient descent (batch) with both cross entropy loss and least mean square loss**

   (not to be done)

IV. Regularization

1) **Generate points with the model y = ax+b+ϵ where epsilon is standard gaussian. x is distributed as uniform rv between [0,10]. Train a linear regression model with following polynomials**
   - **2**
   - **5**
   - **10**

   Study the out of sample performance for each of the above. Compare this when training dataset size is changed.

   Using the above given information, a dataset was generated and polynomials of different orders were fitted as asked in questions.
   Out of sample performance was found to be poor for higher degree polynomials (indicates overfitting).
   And when the training dataset is increased, the error was found to be increased. (Maybe because as the number of data points increased they were expected to be more contributing towards error).

2) **On the above, fix a suitable training dataset size, train a 10 degree polynomial which exhibits overfitting. Implement the following regularization schemes**
   - **Lasso**
   - **Ridge**
   - **Elastic**

   Observe how the coefficients change via a plot for different values of regularization constant. Using a validation approach fixes a regularization constant. Implement a six-fold cross validation method for fixing regularization constant.

   After fixing a suitable training dataset size(size = 30) and fitting the 10 degree polynomial to the data, we could clearly see an overfitted curve.
   Now, the given regularization schemes were imposed. And we observe that

with an increase in regularization constant, the coefficients dropped in magnitudes. Using 6-fold validation approach, we got following regularization constants for above schemes:
Lasso: 10
Ridge: 0.1
Elastic: 0.1, 1
After using these regularization constant, we could see that overfitting has reduced.