Neural Networks in Computer Vision

Assignment 2

In this assignment, you will implement logistic regression algorithm and apply it to the given dataset. Before starting on the programming exercise, I would strongly recommend go through the lecture slides and make a clear outline on paper for all the steps involved.

Dataset

Use the following code to create a dummy dataset. It generates 10000 samples, 5000 for each class (-1 and +1). Each sample has two features x1 and x2. Randomly divide this dataset into training and testing data. Keep training and testing dataset as 80% and 20% of the complete dataset, respectively.

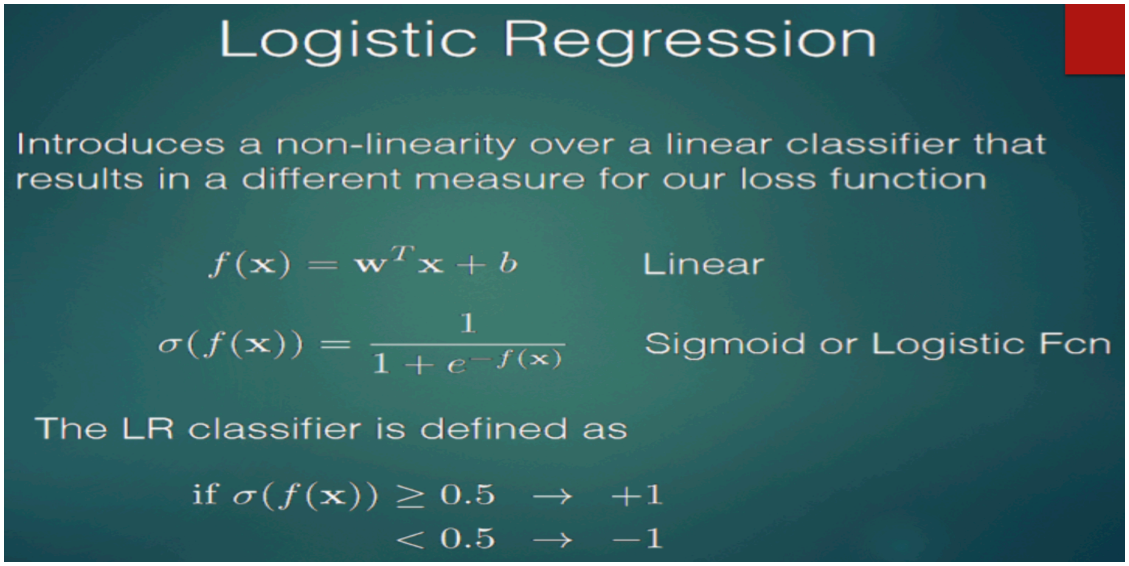
```
import numpy as np

np.random.seed(12)
num_observations = 5000

x1 = np.random.multivariate_normal([0, 0], [[1, .75],[.75, 1]], num_observations)
x2 = np.random.multivariate_normal([1, 4], [[1, .75],[.75, 1]], num_observations)

features = np.vstack((x1, x2)).astype(np.float32)
labels = np.hstack((np.full(num_observations,-1),
                    np.ones(num_observations)))
```

Visualize the dataset

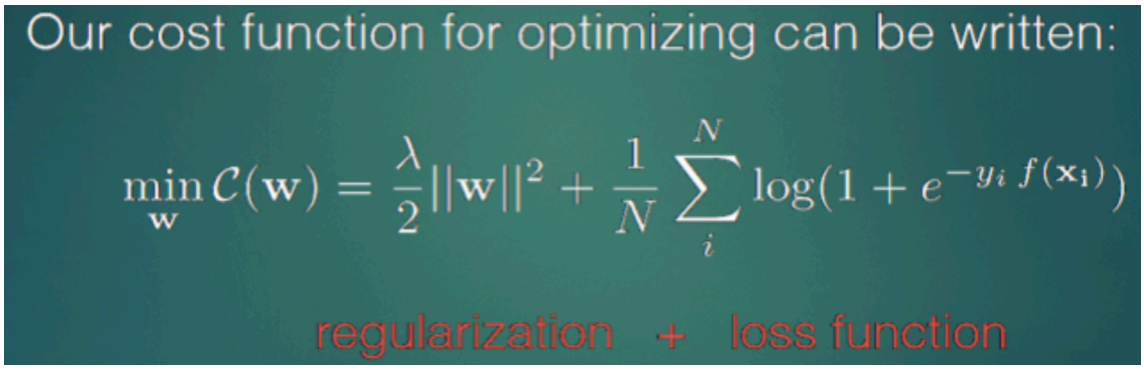
It is always a good practice to visualize the data before implementing any algorithm whenever possible. You have to visualize a scatter plot for all the 10000 samples (undivided dataset). Each sample will have a color and shape depending on which class (+1 or -1) it belongs. Also, mention the legend for the plot. Matplotlib and Seaborn are good python libraries for visualization.

Training the dataset

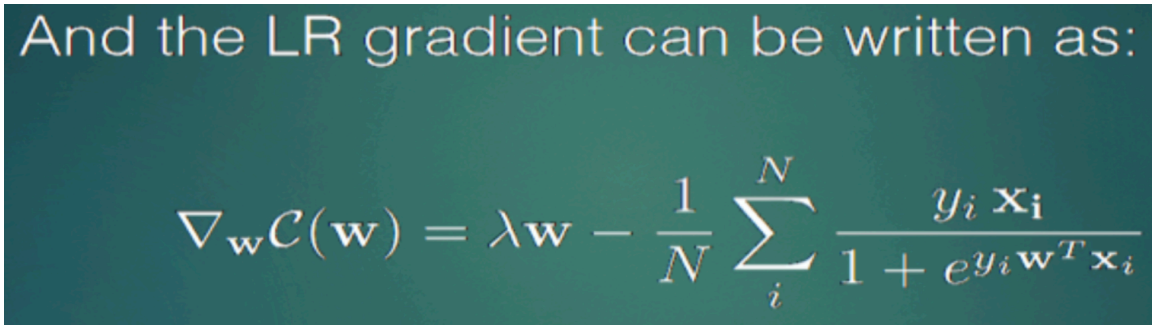
As you see in the following image, logistic regression introduces a non-linearity over a linear classifier. The goal of logistic regression training is to find the optimal W such that ideally for any new sample (test dataset) from class “+1” has sigmoid(w1x1+w2x2+b) value greater than 0.5 and for class “-1” less than 0.5.



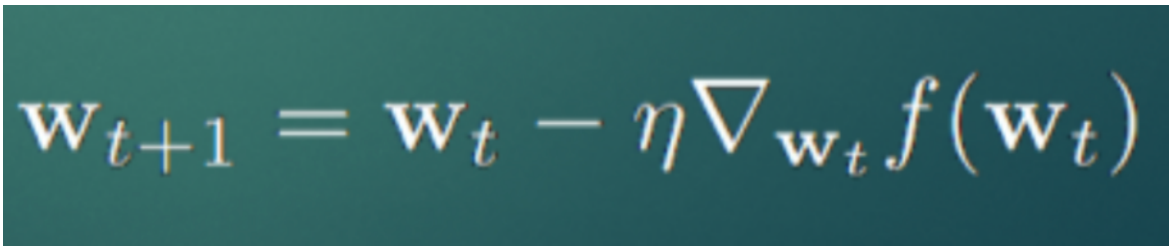
For logistic regression, your first step is to define the cost function which you will try to minimize while training. Write a cost function which calculates the cost for complete training dataset in every iteration. To make it easier avoid the regularization part for this assignment and only calculate the loss.



Once you define the cost function, your next step would be to minimize the cost function to find the optimal *W*. Optimal *W is* where cost function is minimum. To find the minimum value of cost function, you have to define gradient function which calculates the derivative of cost function. Define the gradient function as below. Again, avoid the λW part to make the implementation easier.



Use the gradient function to calculate gradients for complete training dataset in every iteration. Now that you have implemented the cost function and its gradient, it is time to update the *W* based on learning rate (η) and gradient using following equation. Don't update the *W* for each sample, instead update it only once for each iteration. Calculate average gradient for all the training samples for and update the *W* with average gradient.



Play with the learning rate (0.00001, 0.0001, 0.001, 0.01, 0.1 or any value you think would make algorithm converge faster) and the number of iterations (1000 to 30,000 in multiplication of 1000) to find the optimal *W*. See the example below for how to update *W* for each iteration.

```
def logistic_regression_PB(features, label, num_steps, learning_rate):
  weights = np.zeros(features.shape[1])
  total_gradient = 0
  for step in range(num_steps):
    gradient = calc_gradient(features, label, weights)

    avg_gradient = np.mean(gradient,axis=0)

    weights -= learning_rate * avg_gradient

    if step % 1000 == 0:
        print (cost_function(features, label, weights), weights)

  return weights
```

Stop the iteration when your cost function stops decreasing. Print your cost function and corresponding weights for every 1000th iteration.

Train you model with and without bias or intercept parameter and compare the accuracy on testing dataset for both models. Also, compare your results and weights against SKlearn LogisticRegression. While comparing with Sklearn make sure you do apple to apple comparison.

Testing and results

Test each model (with and without intercept) on the testing dataset and print the accuracy, confusion matrix, final weights and intercept.

Tip

Avoid looping whenever possible, instead use matrix dot product (numpy.dot). It is much faster.