Programming Assignment 4

Part 1: Group member and contribution

Bin Zhang (5660329599): implantation of linear regression and logistic regression and report of them.

Yihang Chen (6338254416): scikit-learn implementation of all algorithms and report of software familiarization

Hanzhi Zhang (4395561906): implantation of Perceptron Learning algorithm and pocket algorithm and report of them, report of application

Part 2: Implementation

Perceptron Learning algorithm:

For this algorithm, original data is read by python, and stored into a nested list. Then this nested list would be converted into NumPy array. The initial weight is a 1×4 array which is filled with zeros. The final output is a tuple. The first element is weight, a 1×4 array. The second element is accuracy. After the final iteration, the weight is

[4.45986950e-03,
9.26753429e+00,
-7.44751279e+00,
-5.55154010e+00].

The accuracy is 100%.

In terms of code-level optimization, program uses the first error datasets instead of a random error dataset to update weight. In this case, we don't need to finish the whole iteration, which means we don't need calculate every lines of data. Therefore, the running time is reduced.

Pocket algorithm:

In Pocket algorithm, the data structure is similar with Perceptron Learning. Original data is stored into a nested list and converted into NumPy array. The initial weight is a 1×4 array which is filled with zeros. The final output

is a tuple. The first element is weight, a 1×4 array. The second element is accuracy. After the final iteration, the weight is

[-0.1 ,
-0.03483973,
0.13854714,
0.04461563]

The accuracy is 0.5335.

Both Pocket and perceptron learning algorithm have two changes. If you want to use recursion. Python has a default limitation of recursion, 1000 times. Therefore, I have to use while loop instead of recursion. Moreover, how to set a suitable learning rate is also difficult. Too large learning rate can lead to oscillation of final result. Therefore, I choose to set the learning rate as 1/ (iteration times). In this case, learning rate would reduce with the increasement of iteration.

Linear Regression and Logistic Regression:

For both of these two algorithms, original data is read and processed by pandas, and stored into a NumPy array, here we split the data into training set and test set in order to test the performance of the models. We set the X (n*m) format and Y (1*m) format, which n represents the feature's number and m represents the sample size. Instead of using theta as parameter, we use w and b as parameters because that's convenient when we user matrix operation and we have used this from before. During each iteration's optimization, we use a dictionary params = {"w": w, "b": b} and grads = {"dw": dw, db": db} to update our parameters. We also process the y label to make to the -1 turn to 0 which could also be convenient for later usage(use rint function to compare propobality), One thing that we encountered is the selection of learning rate, we tried for several times and found the 0.05 was the perfect choice for the performance of test set.
And the weight of the linear regression is

W = [[1.08261586]
[3.98916749]]
b = 0.014739833991257945

which means,

Z = 1.08261586*X + 3.98916749*Y + 0.014739833991257945

For the logistic regression, after 7000 iteration, the weight is

W = [[-0.15936798]
[0.24169404]
[0.46357442]]

b = -0.23592208999376346

The accuracy is 53.3 %

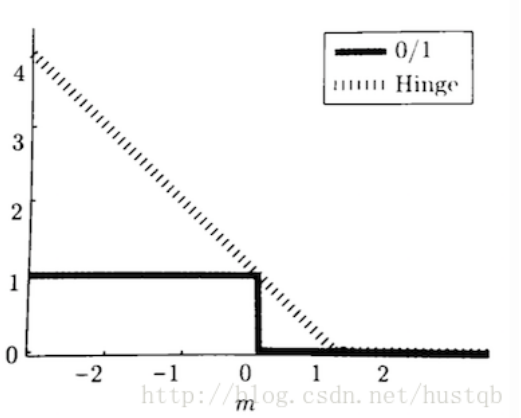
Part 3: Software Familiarization

Linear classification

For the implementation of linear classification algorithm, we used the SGDClassifier in the sklearn library. The difference between this SGD (stochastic gradient descent) classifier and the linear classification algorithm taught in the class is that it do the update with stochastic gradient descent, while the algorithm we learned in the class simply add the loss. It is worthy mentioned is that the default loss function in the SGDClassifier is 'hinge', which is specific for the binary classification problem. The loss function is shown below:

$$\mathrm{l(y)} = \max(0, 1 - y * x * \omega)$$

considering one side situation, the loss function is like below:



It grows linearly with the difference between target label and train output.

The default learning rate in the library is 'optimal', that is to say, the learning rate will change with the running of iteration.

$$\mathrm{eta} = 1.0 / (\mathrm{alpha} * (\mathrm{t} + \mathrm{t0}))$$

Linear regression

For the linear regression, we used the linearRegression method in the sklearn. It provided the ordinary least squares linear regression method. In this method, we can set parameter: "n_jobs" to set the number of jobs to use for the computation. In the algorithm, we can compute loss of every points simultaneously, So by setting this n_jobs to a number more than 1 can reduce the running time.

Also the library offers warm_start option to reuse the solution of the previous call. So when new data added to the trainset, we can set the parameter "warm_start" to True, so we can reduce the number of iterations by using previous coefficient.

logistic regression

We used Logistic regression method in the sklearn. This method fits a linear model with coefficients ω=(ω_1, ⟦… ω⟧ _d) to minimize the residual sum of the squares between the true label and the predicted label.

Like linear regression method, the logistic regression offers similar advantages like n_jobs and warm start to accelerate the process of training.

Part 4: Applications

Perceptron Learning algorithm and Pocket algorithm are the base of Neural Network. Therefore, we can use these linear classifications to make a judgement. For example, we can use linear classification to judge whether there is a cat on the picture.

Linear regression can help us to find the relationship between different variables. In business, we can use the linear regression assess risk in finance services or insurance domain. Insurance company can predict the risk by using the attributes of cars and driver information.

For logistic regression, this algorithm can also be applied to make prediction. However, it can only return binary result. For example, it can predict whether a person is depressed based on the words he is saying.