Machine Learning
hw2 report

Han Hao Chen, r05922021

October 21, 2016

1 Logistic Regression

Logistic Regression The hw2 is a binary classification problem. We need to use pre-processed features to predict the real labels. Therefore, I use logistic regression with sigmoid function and use binary cross entropy as loss function. In inference part, use the testing feature as input, if the output probability is larger than 0.5, then the predict label is 1, otherwise 0.

$$sigmoid(x) = \frac{1}{1 + exp^{(-x)}} \tag{1}$$

$$loss = -\sum_{x}^{|X|} \{\hat{y} \cdot log[sigmoid(W^T x + b)] + (1 - \hat{y}) \cdot [1 - sigmoid(W^T x + b)]\} \tag{2}$$

2 Another method

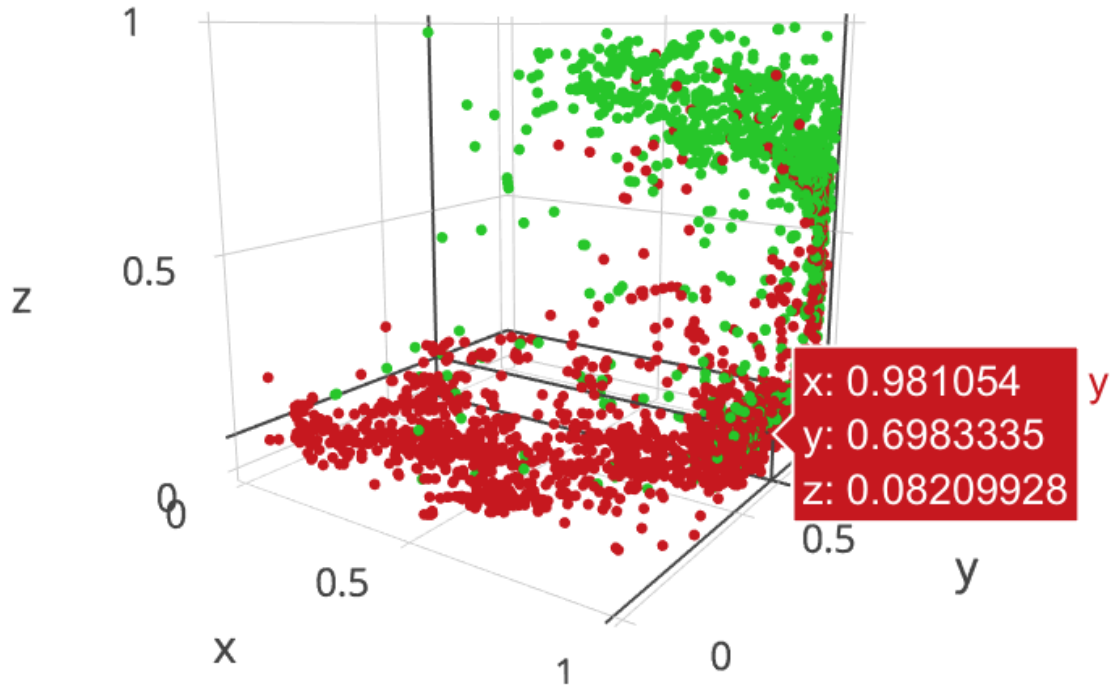
Neuron Network I also use Deep Neuron Network to train on this problem. There are two hidden layers and the output layer has only one node (feature size, 10, 10, 1). Using sigmoid as activation function and using the same loss function in the logistic regression method. The inference part is also similar to logistic regression method, the predict label is 1 if the output is greater than 0.5, otherwise 0.

3 Comparison

Comparison The both methods can achieve 92% accuracy on training data within 500 iterations. The DNN method can even achieve 99% accuracy. On testing data, DNN method can achieve 95% accuracy on the leaderboard. The logistic regression method can get 93.667% on testing with the training accuracy 93%.

4 Training Tips

- Using mini-batch with size 100 in both methods to speed up the training.
- Feature scaling to resist the variance of the features
- The learning rate is set to 1e-4 in logistic regression and 1e-1 in DNN, using adagrad in both methods.



5 Visualization

Auto-encoder I also use the DNN method to train an auto-encoder on the training feature. Projecting 57 dimensions features into 3 dimensions, and using tool plot to visualize it. The green point is the feature with label 0, and the red points is the feature with label 1.

- logistic regression

```
for iters in range(args.iteration):
    cost = 0.
    for i, dat in enumerate(batch_x):

        dot_v = np.dot(w, dat.T) + b
        diff = -(batch_y[i] - sigmoid(dot_v))

        v = sigmoid(dot_v)
        index = np.where(v <= 0.)
        v[index] = 1e-100

        _v = 1 - sigmoid(dot_v)
        _index = np.where(_v <= 0.)
        _v[_index] = 1e-100

        cost += (-(batch_y[i]*np.log(v) + (1 - batch_y[i])*np.log(_v))).sum()

        w_g = np.sum(eta * diff[None,:].T * dat, axis=0)
        b_g = (eta * diff).sum()

        w -= w_g / np.sqrt(grad_w)
        b -= b_g / np.sqrt(grad_b)

        grad_w += w_g**2
        grad_b += b_g**2
```