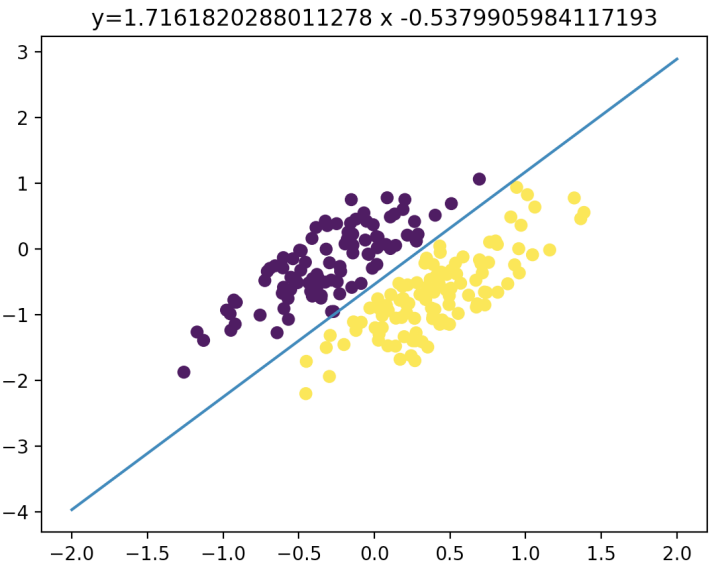
PRML Assignment 2 Report

April 2nd, 2019

Part 1

Least Square Model

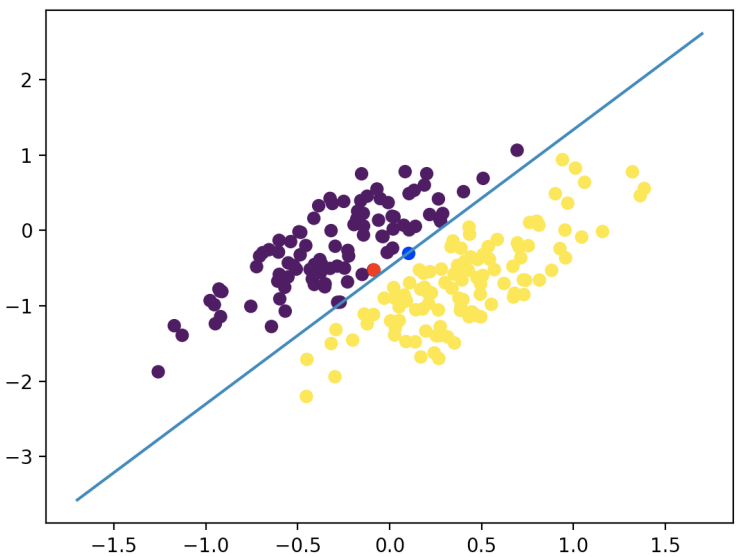
accuracy = 100%



-

Perceptron Algorithm

accuracy = 100%



-

Part 2

a) Indicate how you implemented preprocess pipeline to generate multi-hot and one-hot.

First, in `data_preprocess()` train set and test set are loaded.
I used `item.translate( str.maketrans('', '', string.punctuation) )` to remove all the punctuations, and used regular expression `r'[\s]+'` to split the item into words.
Then I count all the words from train set, and select those frequently appeared words to organize a vacabulary.
Next, in `get_multi_hot(data, vacab)` multi-hot `X` and one-hot `T` are generated.
For every item in data, a `D + 1` vector is generated, and `D` is the size of vacabulary.
For every item in target, a `K` vector is generated, and `K` is the number of classes, in this case `4`.

b) Write down how you compute derivative, and then implement the calculation in vectorized style in numpy.

According to the formula in our text book, we can first write a simple version of derivative calculation.
Here, we just use two layer of iteration. In the inner layer, the contribution of each item of `X` is sumed up. In the outer layer, we merge them together to build the whole derivative.

```
dEW = []
for j in range(K):
  dew = [0.0]*( D+1 )
  for n in range(N):
    dew += ( Y[n][j]-T[n][j] )*X[n]
  dEW.append(dew)
```

Then we can make some improvements.
It is noticeable that the inner loop can be eliminated by using `sum()`. Then we get:

```
dEW = []
for j in range(K):
  dew = sum( ( Y[n][j]-T[n][j] )*X[0:N] )
  dEW.append(dew)
```

Finally, I found that the inner loop can be replaced by matrix multiplication:

```
dEW = (Y - T).transpose() @ X
```

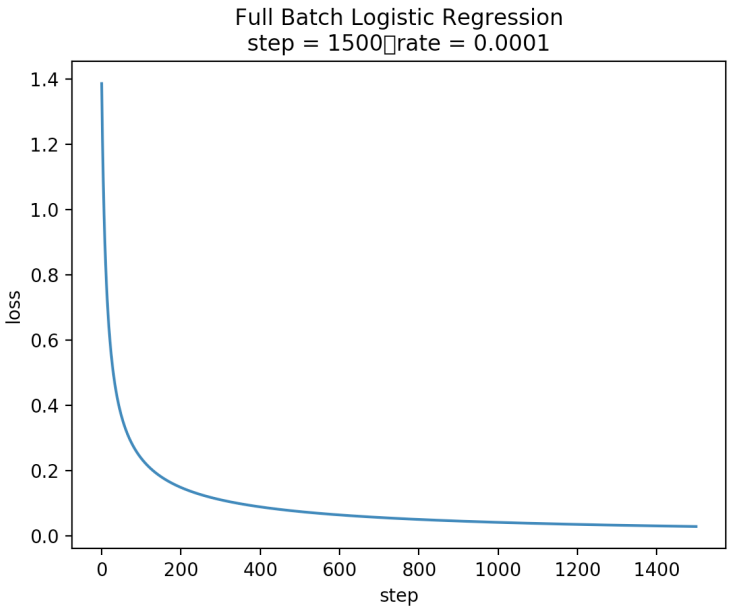
1) Sometimes, to overcome overfitting, people also use L2 regularization for logistic regularization, if you add the L2 regularization, should you regularize the bias term?

No, you don't have to.
The bias term won't contribute to the curvature of the model, so usually overfitting won't be influenced by it. Therefore, there is little meaning to regularize the bias term.

2) How do you check your gradient calculation is correct?

I simply use both the original version and the simplified version of deriative calculation, and make sure they output the same result.
Also, if the gradient calculation is correct, the loss will decrease through the training.

c) Finish training logistic regression model, show the plot for the loss curve.



1) how do you determine the learning rate?

I just pick the learning rate by observing the loss curve, neither should the curve be too steep nor should the loss decrease too slow.

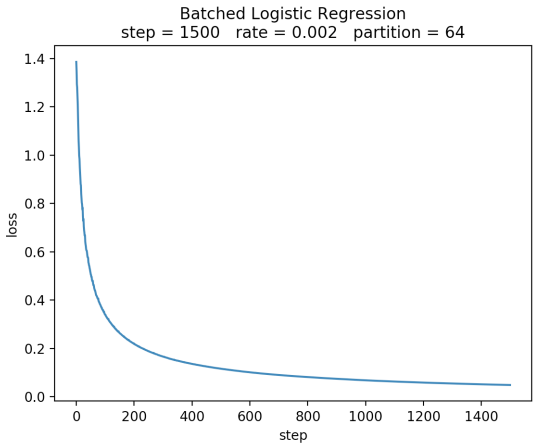
2) how do you determine when to terminate the training procedure?

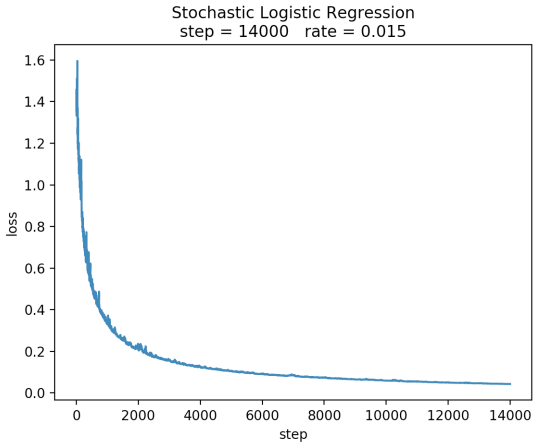
When the loss becomes stable and the curve becomes almost horizontal, the training is finished.

d) Sometimes people use stochastic gradient descent and batched gradient descent.

1) What do you observe by doing the other 2 different ways of gradient descent?

With batched gradient descent, the curve is a little bit spiky.
With stochastic gradient descent, the curve becomes very spiky and the training speed is fast.





2) Can you tell what are the pros and cons of each of the three different gradient update strategies?

- full batch

  > pros:
  > a) global optimum achievable
  > b) very stable
  > cons:
  > a) can be very slow
  > b) require large memory size

- batched

  > pros:
  > a) require less memory on the fly
  > b) relatively stable
  > cons:
  > a) need to adjust the batch size

- stochastic

  > pros:
  > a) fast training speed
  > b) require less memory on the fly
  > cons:
  > a) unstable
  > b) spiky learning curve

e) Report your result for the three differently trained model on the test dataset.

- full batch

  > correct: 1388 (92.78%)
  > wrong: 108
  > total: 1496

- batched

> correct: 1386 (92.65%)
> wrong: 110
> total: 1496

- stochastic

> correct: 1385 (92.58%)
> wrong: 111
> total: 1496