PRML Assignment2 Report

Part 1

1. Least Square Model

To build a least square model, we need to find an optimal parameter matrix $\tilde{W}$. We can learn the derivation of $\tilde{W}$ from the book(not repeat here), and get the following formula:

$$\tilde{W} = (\tilde{X}^{\mathrm{T}}\tilde{X})^{-1}\tilde{X}^{\mathrm{T}}T \tag{1}$$

Then we can get the $\tilde{W}$ and the classification line is like that:

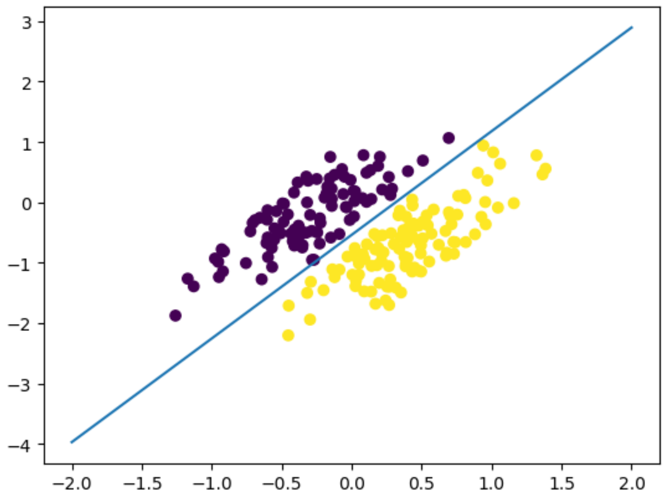


Figure 1.1

It seems that the line classify the data successfully. In fact it dose, and the accuracy calculated is 1.0.

2. Perceptron Algorithm

For this algorithm, we also want to find a optimal parameter matrix W. According to the teaching material, we also use a non-linear function $\phi()$, but here I just directly use x to take place of $\phi(x)$ and I also get a nice result. The difference between perceptron algorithm and above is that we can not directly calculate the W in this algorithm. Instead, we should use gradient descent algorithm to find a relatively good parameter matrix.

To do the gradient descent algorithm, we should first get the loss function which is(no more explanation):

$$E_p(\omega) = -\sum_{n\in\mathcal{M}} \omega^{\mathrm{T}}\phi_n t_n \tag{2}$$

And we can take its partial derivative which is $\sum_{n\in\mathcal{M}} \phi_n t_n$. Then we need a learning rate, η which represent the speed of the gradient descenting. Every time we just need to subtract this value dot η from W. To get a better training effect, we can use stochastic gradient descent algorithm which means each time we don't need to calculate the weight change from all of the wrong points, instead we only calculate it from a random one.

What's more, for the learning rate, at first I set it equal to 0.1 to make the training more quickly, then I will decrease it after a certain training to get an more accurate model. At last, I also get a model which can get the accuracy equal to 1.0 nearly everytime, and its figure is following:

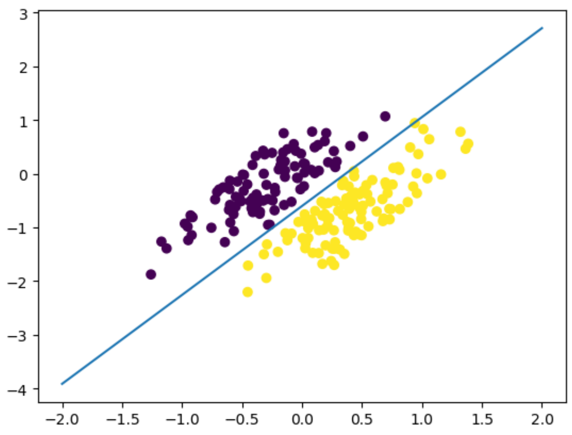
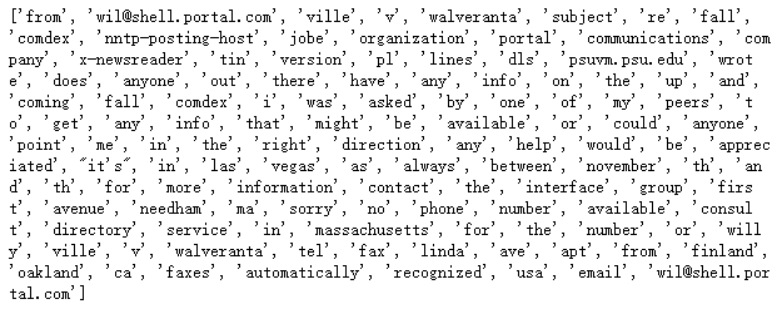


Figure 1.2

Part 2

Task 1

For this part, first we should tokenize the document into words and turn it into multi-hot vectors. To do this, I use a regular expression to replace every digit, continuous whitespace and some special symbol that won't be in a word by a only whitespace. Then I can split it in to a list of strings, and use the *strip()* to remove the punctuaion occuring in the head or tail of those strings. and we get the result like that: According to the task prompt, I konw that we also need a **min_count** to filter the words that only occurs for



```
['from', 'wil@shell.portal.com', 'ville', 'v', 'walveranta', 'subject', 're', 'fall',
'comdex', 'nntp-posting-host', 'jobe', 'organization', 'portal', 'communications', 'com
pany', 'x-newsreader', 'tin', 'version', 'pl', 'lines', 'dls', 'psuvm.psu.edu', 'wrot
e', 'does', 'anyone', 'out', 'there', 'have', 'any', 'info', 'on', 'the', 'up', 'and',
'coming', 'fall', 'comdex', 'i', 'was', 'asked', 'by', 'one', 'of', 'my', 'peers', 't
o', 'get', 'any', 'info', 'that', 'might', 'be', 'available', 'or', 'could', 'anyone',
'point', 'me', 'in', 'the', 'right', 'direction', 'any', 'help', 'would', 'be', 'apprec
iated', "it's", 'in', 'las', 'vegas', 'as', 'always', 'between', 'november', 'th', 'an
d', 'th', 'for', 'more', 'information', 'contact', 'the', 'interface', 'group', 'firs
t', 'avenue', 'needham', 'ma', 'sorry', 'no', 'phone', 'number', 'available', 'consul
t', 'directory', 'service', 'in', 'massachusetts', 'for', 'the', 'number', 'or', 'will
y', 'ville', 'v', 'walveranta', 'tel', 'fax', 'linda', 'ave', 'apt', 'from', 'finland',
'oakland', 'ca', 'faxes', 'automatically', 'recognized', 'usa', 'email', 'wil@shell.por
tal.com']
```

a little times. After that, just do a count and we get the dictionary having about 6000 keys. Since the process of using the dic to turn the list into multi-hot vectors is easy, here I just pass it.

Task 2

The loss function for the logistic regression is following:

$$\mathcal{L} = -\frac{1}{N}\sum_{n=1}^{N} y_n log\hat{y}_n + \lambda\|W\|^2 \tag{3}$$

To get the differential coefficient easily, I use $\tilde{W}$ which equals to (W,b) and $\tilde{x}$ which equals to $(1, x^T)^T$(1 means a vector consist of 1), and what we need to calculate is only $\frac{\partial\mathcal{L}}{\partial\tilde{W}}$.For each $\mathcal{L}_i$ we have(ignore the regular term and $\frac{1}{N}$ first):

$$\mathcal{L}_i = -{y_i}^T log\hat{y}_i = -{y_i}^T log\frac{\exp(\tilde{W}x_i)}{sum(\exp(\tilde{W}x_i))} = -{y_i}^T log\frac{\exp(\tilde{W}x_i)}{1^T\exp(\tilde{W}x_i)} \tag{4}$$

Let $\tilde{W}x_i = z_i$, and we can calculate $\frac{\partial\mathcal{L}}{\partial z_i}$ first(assume that $y_{nt} = 1$ here, so the true class is t):

for the z_{ik}(k!=t) we have:

$$\frac{\partial\mathcal{L}}{\partial z_{ik}} = \frac{\partial(-z_{ik} * 0 + log\sum exp(z_{ij}))}{\partial z_{ik}} = \frac{exp(z_{ik})}{\sum exp(z_{ij})} = \hat{y}_{ik} - y_{ik} \tag{5}$$

for the z_{it} we have:

$$\frac{\partial\mathcal{L}}{\partial z_{it}} = ... = \hat{y}_{it} - 1 = \hat{y}_{it} - y_{it} \tag{6}$$

So in conclusion, the value of $\frac{\partial\mathcal{L}}{\partial z_i}$ is $\tilde{y}_i - y_i$, Then, dot the $\frac{\partial z_i}{\partial\tilde{W}}$, the final result will be $(\hat{y}_i - y_i)x_i$. Then we reconsider the $\frac{1}{N}$ and regular term, and we wil get(here, the regular term use W rather than $\tilde{W}$):

$$\frac{\partial\mathcal{L}}{\partial\tilde{W}} = \frac{1}{N}(\hat{y}_i - y_i)x_i + 2 * \lambda * \|W\| \tag{7}$$

Question 1 I won't regularize the bias term. For one reason, the bias stand for the numerical deviation, I think it won't cause the overfitting. For the other, the number of parameter in the bias is far less than W, so it won't bring a big influence.

Question 2 Do gradient checking, set a Δx(a matrix) with small value and calculate $\frac{\mathcal{L}(\tilde{W}+\Delta x)-\mathcal{L}(\tilde{W}-\Delta x)}{2\Delta x}$, and compare the result with the result of $\frac{\partial \mathcal{L}}{\partial \tilde{W}}$.

Task 3

See the code for the specific training process. Here I plot the loss curve of the training in a full batch gradient descent:

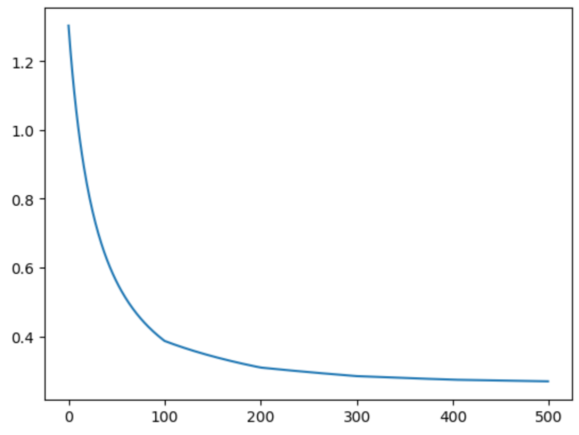


Figure 2.1

Question 1 In my opinion, determining the learning rate still depends on experience. I know that a big learning rate may lead to unstable convergence of the model, while a tiny rate usually cause a low training efficiency. For myself, I start to train with a relatively large learning rate, like 0.2 or 0.1, and after every certain iteration I will decrease it(it will be about 0.0001 in the end).

Question 2 In the beginning, I set a large number of iteration for the training because I think after such adequate training the algorithm is bound to converge. In fact, there is no need to train so many times. we just need to define a reasonable threshold, and finish the training when the difference between two iterations is less than the threshold.

Task 4

Question 1 For the stochastic gradient descent(Figure 2.2) and batched gradient descent(Figure 2.3), the loss curves is as follow: By looking at these curve and the loss

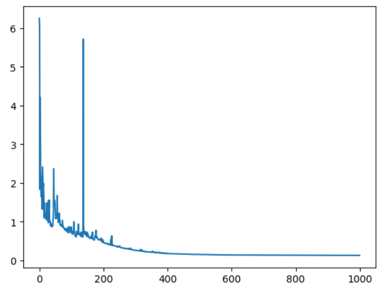


Figure 2.2

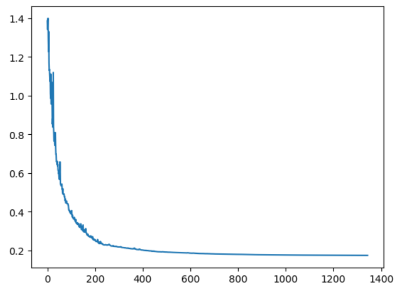


Figure 2.3

in the training, We can find that, when using stochastic gradient descent(pick a point every 20 iteration), the gradient descent is unstable in the beginning, and converge after a certain amount of training. When use this method to train, the model will converge faster the full batch method for it update at a faster rate.
For the batched gradient descent, it's more like a compromise between these two methods discussed above. It will also unstable at the beginning, but is in a less degree than stochastic one. Also, the model will have a faster converging speed by this method.

Question 2 The comparation between the three method is discussed in the Question 1. Here is the conclusion: The full batched gradient descent is easy to implement and stable, which means we can always find the globally optimal solution by this method. But when the number of training dataset is big, the speed of training by this way will be slow, which can be improved a lot by using stochastic gradient descent. And the shortcoming of stochastic method is that it's unstable which means it sometimes will miss the globally optimal solution. And the batched gradient descent may be a compromise between the two methods.

Task 5

The accuracy of three trained model is as follow(the result is going to fluctuate a little bit randomly and the parameter is step $= 0.2$, $\lambda = 0.001$, batch $= 10$):

Full batch gradient descent with 500 times iteration: 0.9258021390374331

Stochastic gradient descent with 20000 times iteration: 0.9271390374331551

Batched gradient descent with 6 epochs: 0.9318181818181818