Assignment-2 Report

PART 1

Least Square Model

Accuracy

91.8918918918919 %

Process

- According to equation inference from PPT，get

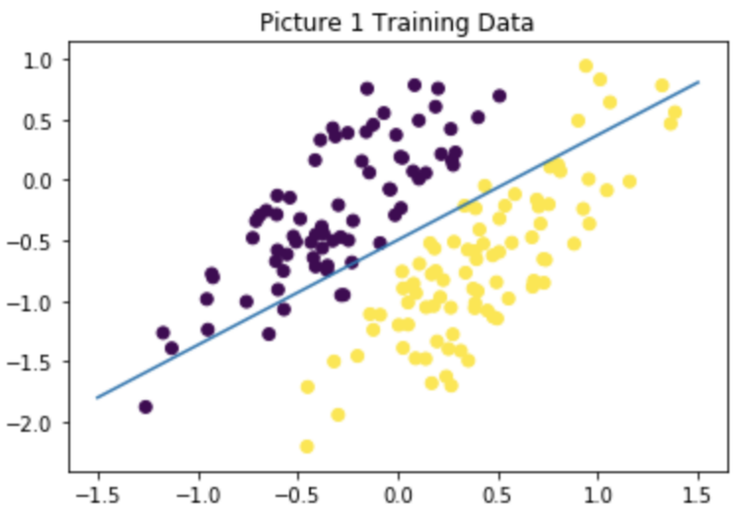
$$S_w^{-1}S_b w = \lambda w$$

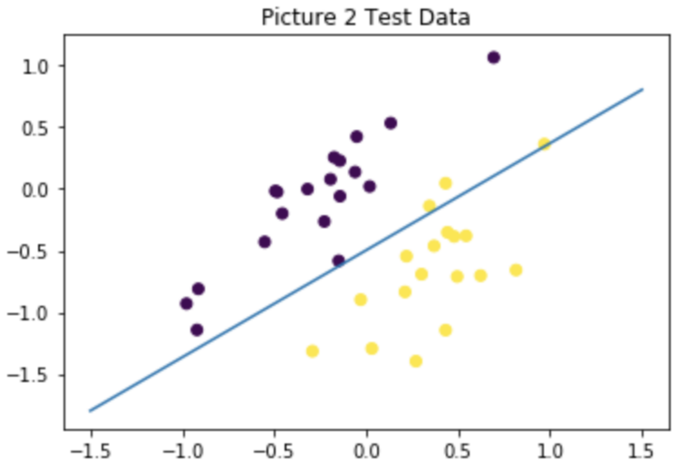
eigenvalues and eigenvectors， choose eigenvector whose eigenvalue is largest.

- to know what I think to see comment in codes

Decision line

- decision line is the red lines in "Picture 1 Train Data" and "Picture 2 Test Data"

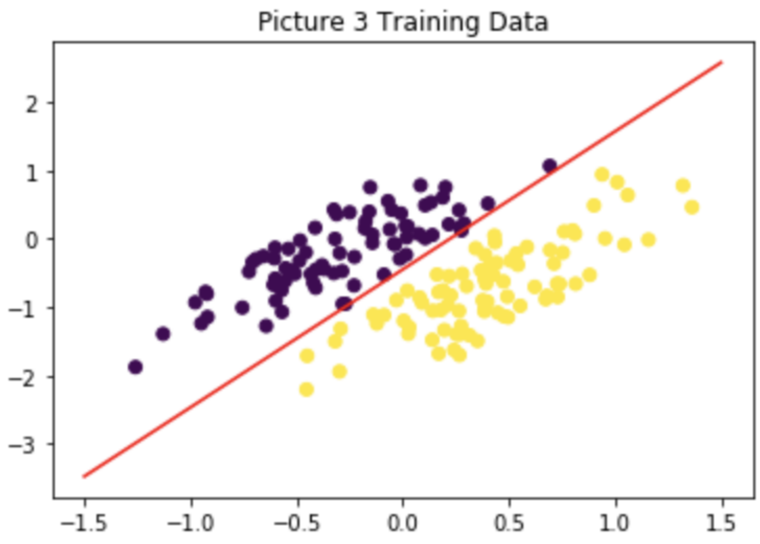


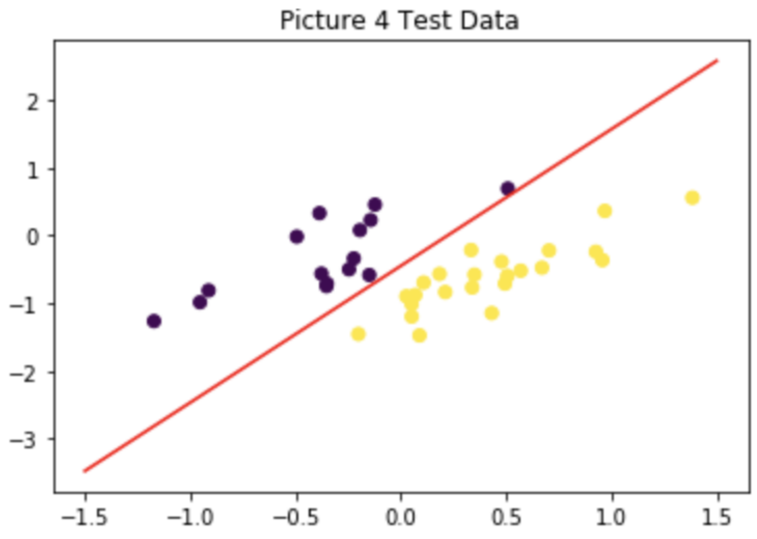


Perception Algorithm

Decision line

- Decision Line is red lines in "Picture 3 Training Data" and "Picture 4 Test Data"



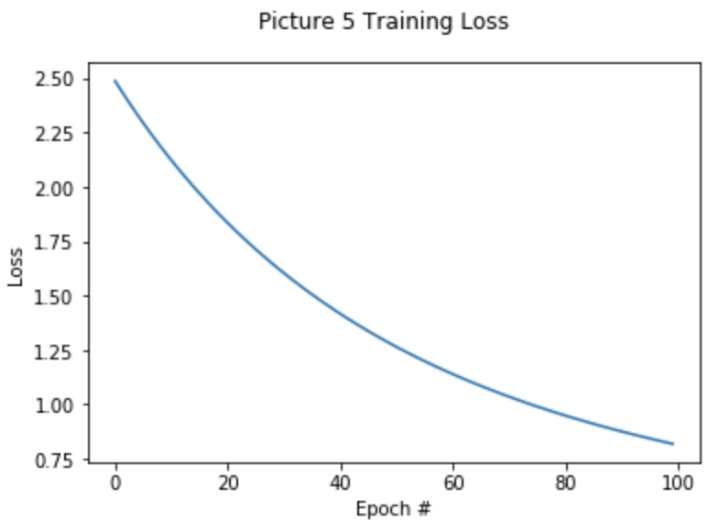


Accuracy

- given epoch 100， learning_rate 0.01, accuracy can achieve 100%

Code comments

- use perceptron algorithm and SGD to compute W
- frome Picture 5 Training Loss, loss decreases as the iteration increases.



PART 2

Requirement 1

Document to Multi-hot Vector

Traverse dataset.data whose element is a string.

1. transfer all letter into lowcase
2. replace whitespace and punctuation with space
3. split that processed string by space and get a list
4. use set operation to remove repeated words in that list and get a tokenized list of word, namely sentence in source code
5. append all sentence in a list and get a list of list, namely tokenized_sentences in source code
6. put elements of all sentence into a big list, namely list_of_string in source code and remove repeated element by set()
7. sort list_of_string and we can use it to get a vocabulary
8. use that vocabulary map tokenized_sentences into multi-hot vectors, namely multi_hot_v in source code

Categories to One-hot Vector

As follows:

1. each category is represented by a one-hot vector whose length is the same as the number of categories
2. set k-th element to 1 if the category ranks number k

3. set other element to 0

Requirement 2

Computation

$$\frac{\partial L}{\partial w_{i,j}} = -\frac{1}{N}\sum_{n=1}^{N} y_n \frac{\partial log^{\overset{\wedge}{y_n}}}{\partial w_{i,j}}$$

make

$$a^n = log\hat{y} = [a_1^n, \quad a_2^n, \quad ..., \quad a_K^n]^T$$

get

$$\frac{\partial L}{\partial w_{i,j}} = -\frac{1}{N}\sum_{n=1}^{N} y_n \frac{\partial log^{\overset{\wedge}{y_n}}}{\partial w_{i,j}} = -\frac{1}{N}\sum_{n=1}^{N} y_n \frac{\partial a^n}{\partial w_{i,j}}$$

out of softmax function, we can get

$$a_f^n = log\frac{e^{z_f}}{\sum_{m=1}^{K} e^{z_m}}$$

assume

$$\begin{gathered} W^T = [w_{11}, w_{12}, ..., w_{1d} \\ w_{21}, w_{22}, ..., w_{2d} \\ ... \\ w_{K1}, w_{K2}, ..., w_{Kd}] \end{gathered}$$

and assume

$$x^n = [x_1^n, x_2^n, ..., x_d^n]^T$$

K is the total number of class, and the d is the dimension of the feature vector

thus

$$\frac{\partial a_f^n}{\partial w_{i,j}} = \frac{x_j^n(\sum_{m=1,m\neq i}^{K} e^{z_m})}{\sum_{m=1}^{K} e^{z_m}} \qquad (f = i)$$
$$\frac{\partial a_f^n}{\partial w_{i,j}} = \frac{x_j^n(e^{z_i})}{\sum_{m=1}^{K} e^{z_m}} \qquad (f \neq i)$$
$$where$$
$$z_m = [w_{m1}, w_{m2}, ..., w_{md}]\overrightarrow{x} + b_m$$

finally, we get

$$\frac{\partial L}{\partial w_{i,j}} = -\frac{1}{N}\sum_{n=1}^{N}(y_n[V_1^n, V_2^n, ..., V_K^n]^T)$$
$$and$$
$$V_f^n = \frac{\partial a_f^n}{\partial w_{i,j}}$$

Compute

$$\frac{\partial L}{\partial b_i} = -\frac{1}{N}\sum_{n=1}^{N} y_n \frac{\partial log^{\overset{\wedge}{y_n}}}{\partial b_i}$$

make

$$x_j^n = 1$$

and by former inference, we can easily get

$$\frac{\partial L}{\partial w_{i,j}} = -\frac{1}{N}\sum_{n=1}^{N}(y_n[B_1^n, B_2^n, ..., B_K^n]^T)$$
$$and$$
$$B_f^n = \frac{\partial a_f^n}{\partial b_i}$$
$$where$$
$$\frac{\partial a_f^n}{\partial b_i} = \frac{(\sum_{m=1,m\neq i}^{K} e^{z_m})}{\sum_{m=1}^{K} e^{z_m}} \qquad (f = i)$$
$$\frac{\partial a_f^n}{\partial b_i} = \frac{(e^{z_i})}{\sum_{m=1}^{K} e^{z_m}} \qquad (f \neq i)$$

Question 1:

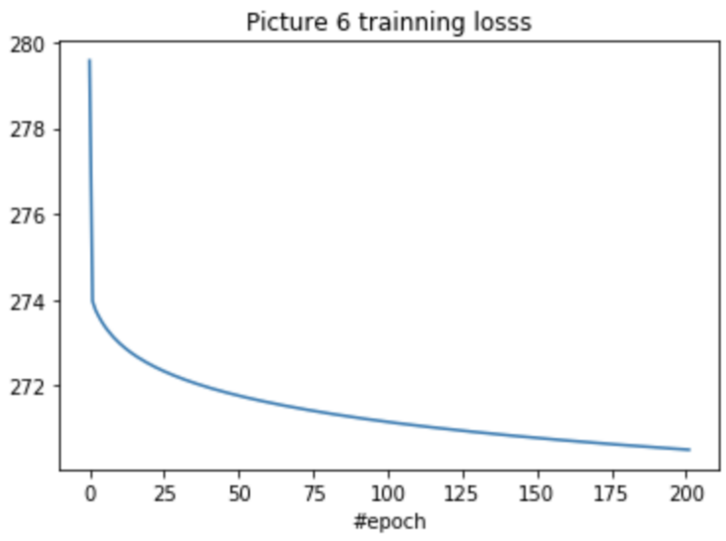
I will not regularize the bias term. Regularization is used to overcome overfitting while bias term only influence curve position or average value rather than direction of curve or distribution. So no need to add.

Question 2:

1. after computing, automatic differentiation can be use to check whether the computation is right by some CAD
2. we can also use finite difference method to verify whether the computation result is right. 4 or 6 digits precision is fine. And I found on the website that 4 digits precision is enough for SGD.

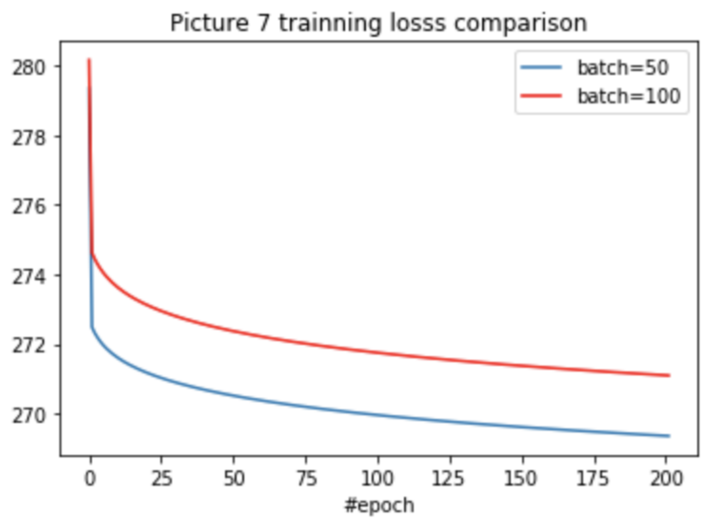
Requirement 3

1. learning rate determination
   I will randomly choose a learning rate and draw the trainning loss. If the loss converge right before the end of trainning, the learning rate is proper. If the loss dose not converge, a lager learning rate will be proper. If the loss converge too early, a smaller learning rate will be proper. Repeat the process until a proper learning rate is found.
2. how do you determine when to terminate the training procedure?
   We can first determine a boundary. Then save last trainning loss and compute the absolute difference between last trainning loss and current trainning loss. If absolute difference is less than the boundary, it is fitting and proper that we can terminate the trainning.
3. trainning loss



Requirement 4

1. what do you observe by doing the other 2 different ways of gradient descent
   stochastic gradient descent often have a between trainning result but slower
   batched gradient descent have average performance both on trainning result and speed compared to other 2 GD.



2. pros and cons of 3 GD

pros/cons	**full batch gradient descent**	**stochastic gradient descent**	**batched gradient descent**
pros	always update to optimal	very quick	quick and often to optimal
cons	slow	noisy data, not always optimal	not very quick or always to optimal

Requirement 5

1. accuracy on test data
   93.38%

Tips

When you test part 1 of my code, please comment code belonging to part 2.

When you test part 2 of my code, please comment code belonging to part 1.

References

Differentiate cross entropy function
Differences between GDs