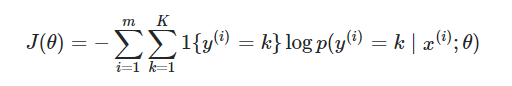
**Midterm Project Report**

20170240 박기범

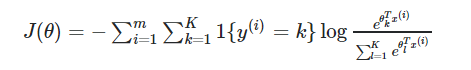
1. **Task 1 – Iris Problem**

* **Loss Function and Gradient Descent Method**

This is the loss function of softmax regression.



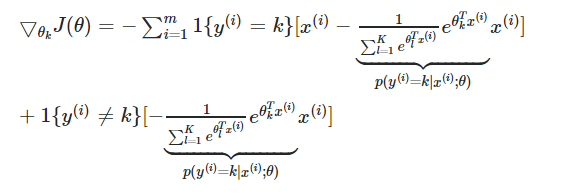
Since the probability inside of the log term means the softmax function we can interchange this term.



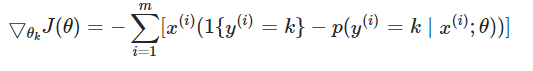
By the property of log function, we can rewrite like this.



Now, we are going to get the gradient of loss function at certain theta k. Since we get derivative regarding to the theta k, the inside sigma part except k index goes to zero when we get derivative. The second part is when the constant term(1) is not the k.

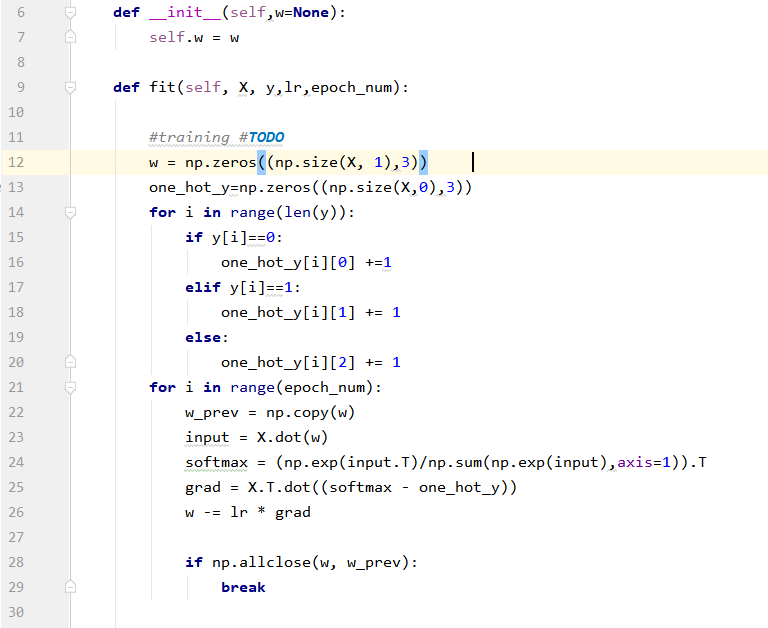


After merging two parts and interchange the softmax function to the probability notation, we can get this equation.



* **Code Explanation and Accuracy**

\*\*\* #n : line n-th explanation



#6-7 : I made global variable self.w in this class.

#12 : First, set the initial w which is weight array for the softmax regression. The size of w array is (the column size of input data) X 3. Since we should use different weights combination for each class there is 3 columns and each column is for the each class weights.

#13 : Also we should make target ‘y’ be one-hot encoding. Therefore, I made array for y, and its size is (input data size) X 3. There are total n target data where n is the data size of input and 3 classes therefore I initialize one\_hot\_y as above.

#14-20 : This part is to encode the one\_hot\_y. For each class we assign number ‘1’ to appropriate class position in one\_hot\_y. For example, the first ‘if statement’ is to assign class 0 targets. Do this assignment process for whole targets in ‘for statement’.

#21-29 : This part is for the gradient descent method. Since we have all things to do the gradient descent method now it is time for it.

#21 : Open the ‘for statement’ for epoch number of iterations.

#22 : Store the current ‘w’ to the w\_prev. This is for #28.

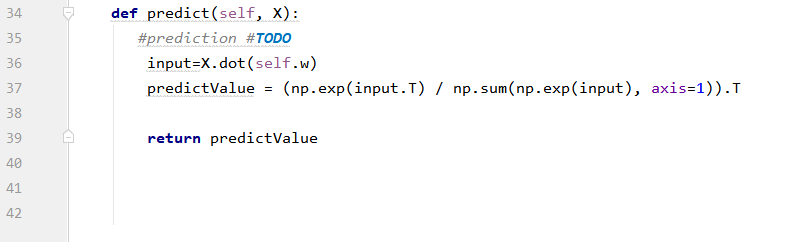
#23 : Multiplies input data ‘X’ and weights ‘w’

#24 : This line is to calculate the probability of classification. This is array of 1 by 3 size.

#25 : Calculate gradient as we wrote formula above. As you can see ‘softmax’ is the probability part and ‘one\_hot\_y’ is the target y part in the formula.

#26 : Update the ‘w’ array.

#28-29 : If the ‘w\_prev’ and updated ‘w’ has no much difference break to stop.



#36 : As we did above method calculate the input.

#37 : Same as above. This time, we will call the calculated probability as ‘predictValue’ not ‘softmax’ as we did before.

#39 : Finally, return the predictValue.

Accuracy : I got 100 percent of accuracy



* **Binary Logistic Regression Models to Multi-class Problems**

We can split the multi-class classification into binary classification. For example, let’s assume there are n classes to classify. We are going to split this multi-class classification into n times of binary classification. First, just do the binary classification for 1st class and the other classes then we can get the weights for classification of 1st class. Then do the same thing for 2nd class, 3rd class, 4th class, and go on. After the n times of binary classification learning, we have n number of weights array and we can calculate each probability to be kth class for certain data point. To classify what class is for each data, we just choose which has the maximum calculated probability among n number of probabilities.

1. **Task 2 – Titanic Problem**

* **Data Preprocessing Part**

It was very ambiguous and difficult to deal with titanic data. First of all, the data set have so many null values and ambiguous values. For example, ‘Embarked’ or ‘Cabin’ is hard to tell the meaning of it.

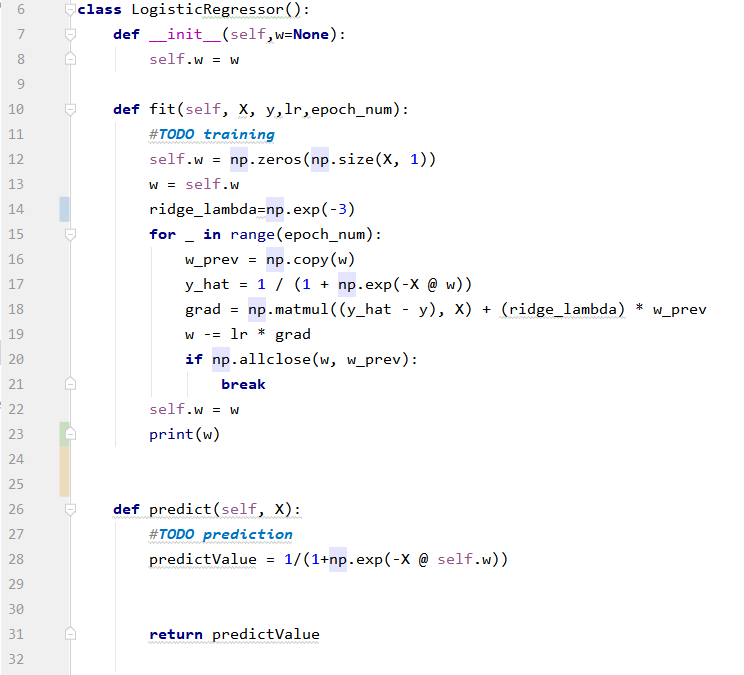
However, after finding meanings of each data, I just dropped columns which I think are useless for classification (‘Embarked’, ‘Cabin’, ‘Name’, ‘Ticket’, ‘Fare’)

I tried lot of ways for data preprocessing. For example, I put new columns which stand for ‘Family Size’ or ‘Travel Alone’. However, I couldn’t find the efficacy of using new columns. Actually a lot of people say that it is better to use it, but it wasn’t for my model.

Also, the key point of data preprocessing was to fill the null values. There are several null values in ‘Age’. At the first time, I just filled the average age of other people which have non null value. It was not bad to use this way, but I thought it is better to fill the null ages more sophisticatedly. Therefore, I filled different ages for each person. The hint comes from person’s name. There are some initial part in the name such as ‘Dr’, ‘Mrs’, ‘Mr’, etc. So after extracting this initials, I calculated average of age for each initials. Therefore I could fill these values for each person.

Actually, there were so many options to do pre-processing. So I should investigate which method has the best outcome. I made some tables and investigated which is the best model to use. The best data pre-processing method was to drop the columns that I stated above and fill the null values in ‘Age’ column by sophisticatedly.

* **Logistic Model Code Explanation**



#7-8 : I made global variable ‘self.w’ in this class.

#12 : Initialize self.w with 0s array with size of input’s features

#13 : Assign self.w to the local variable ‘w’.

#14 : I use ridge regularization for this model to prevent overfitting so give ridge term weight as like this.

#15 : Epoch number of iterations loop.

#16 : Store current ‘w’ to ‘w\_prev’.

#17 : calculate the probability as y\_hat.

#18 : Calculate gradient as we did in ‘Assignment 2’. However, there is one more term which is for the ridge regularization.   
#19 : Update the ‘w’

#20-21 : If the ‘w\_prev’ and updated ‘w’ has no much difference break to stop.

#28 : Calculate the probability of logistic regression.

1. **Questions**

* **The efficacy of K-cross validation**

I tried K-cross validation for titanic problem. If you go to the K-cross validation python file, you can see how I tried. But I couldn’t find the efficacy. Is it really helpful to use this method when we do learning step?

* **Ridge Regularization**

I did parameter tuning for ridge weight term using log scale number such as exp(-3), exp(-2) ,…

Are there special technique when we do hyper parameter tuning? Or just Brute Force method?

* **Data Preprocessing**

Why doing a understandable and sophisticated way of data processing do give bad result? I thought more sophisticated it would be better. But it wasn’t in titanic problem. How can we find which is the best method for pre-processing for each data?