

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**

**Student ID 201722800144**

**E-mail syed.walid@qq.com**

**Tutor**   **Mingkui Tan**

**Date submitted** **2017. 12 . 08**

**1. Topic:** Linear Regression, Linear Classification and Gradient Descent.

**2. Time: 2017/12/07**

**3. Reporter: Karim Syed Mohammed Walid (王力).**

**4. Purposes:** The main purpose is to realize the process of optimization and adjusting parameters and further understand of liner regression and gradient descent.

**5. Data sets and data analysis:** In Linear Regression I used Housing (scaled to [-1,1]) in LIBSVM Data. LIBSVM allows for sparse training data. That is, the non-zero values are the only ones that are included in the dataset. Hence, the index specifies the column of the instance data (feature index). To convert from a conventional dataset just iterate over the data, and if the value of X(i,j) is non-zero, print j+1 : X(i,j).

-1 5:1 7:1 14:1 19:1 39:1 40:1 51:1 63:1 67:1 73:1 74:1 76:1 78:1 83:1

Here -1 is the class or label

The values of paramaters with indexes 1,2,3,4,6, and others unmentioned are 0's

In 5:1, 5 is index, and 1 is the value of parameter.

If you run it on weka , then it needs data file in arff format , other wise it is simple

feature row , group

or

feature row , label

or

feature row , class name

last column shoud be class ,group label

For most sets, we linearly scale each attribute to [-1,1] or [0,1]. The testing data is adjusted accordingly. Some training data are further separated to "training" (tr) and "validation" (val) sets.

And for Linear classification I used australian (scaled to [-1,1]) in LIBSVM Data.

1. **Experimental steps:**

**Linear Regression and Gradient Descent:**

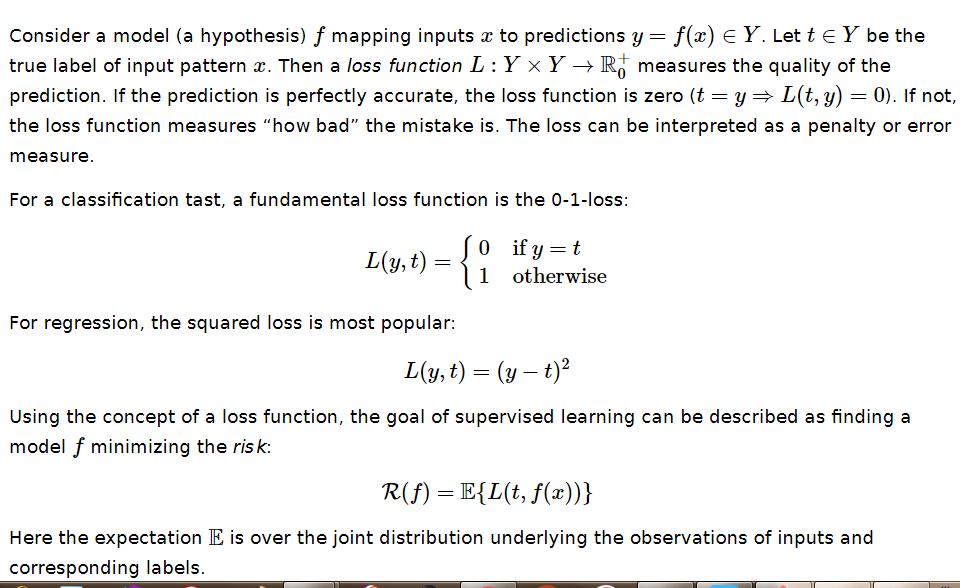
01: At the first step I loaded the experiment date set, I used load\_svmlight\_file function in sklearn library.

02: Then I divided dataset into training set and validation set using train\_test\_split function.

03: Here I initialize liner model parameters.

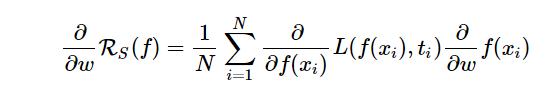
04: I picked up the loss function and derivation.

Loss function:



Derivatives:

When both the loss function and the model are differentiable, it is possible to calculate the derivative of the empirical risk with respect to the model parameters ω:



05: From all samples I calculated gradient G toward loss function.

06: Denote the opposite direction of gradient G -> D.

07: I updated the parameters here.

08:Get the loss Ltrain under the training set and Lvalidation by validating under validation set and draw graph of Ltrain and Lvalidation with the number of iterations.

**Linear Classification and Gradient Descent:**

1: First download the dataset using requests.get() function. Then load the experiment data.

2: Divide dataset into training set and validation set using train\_test\_split function.

3: Initialize svm model parameters with normal distribution.

4:Define loss function and derivation.

5: Calculate gradient descent toward loss function from all samples.

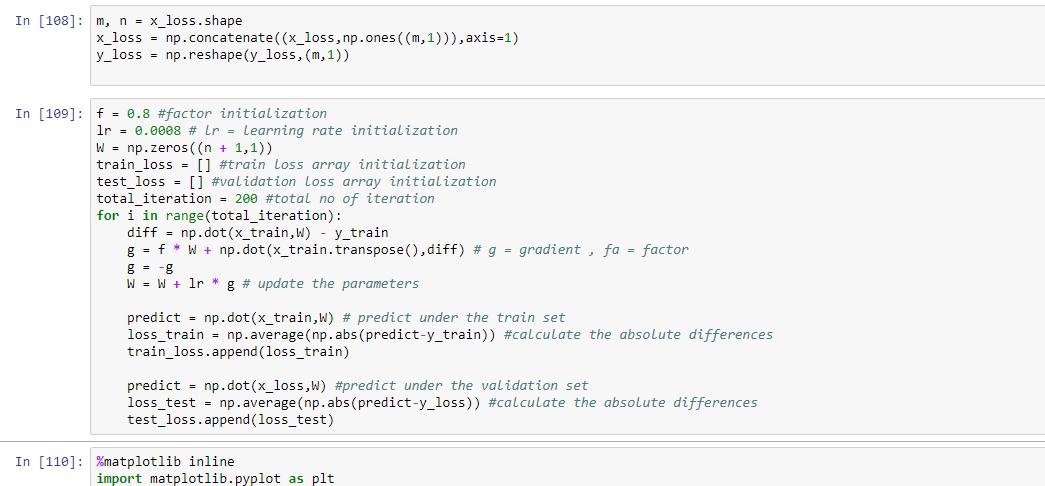
6: Denote the opposite direction of gradient G as D.

7: Update the parameter until the iteration complete.

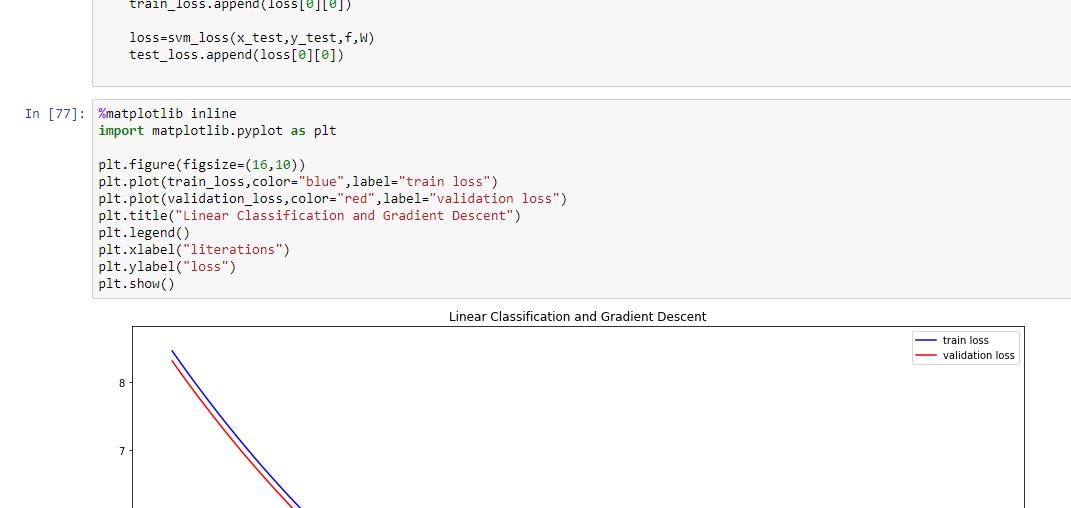
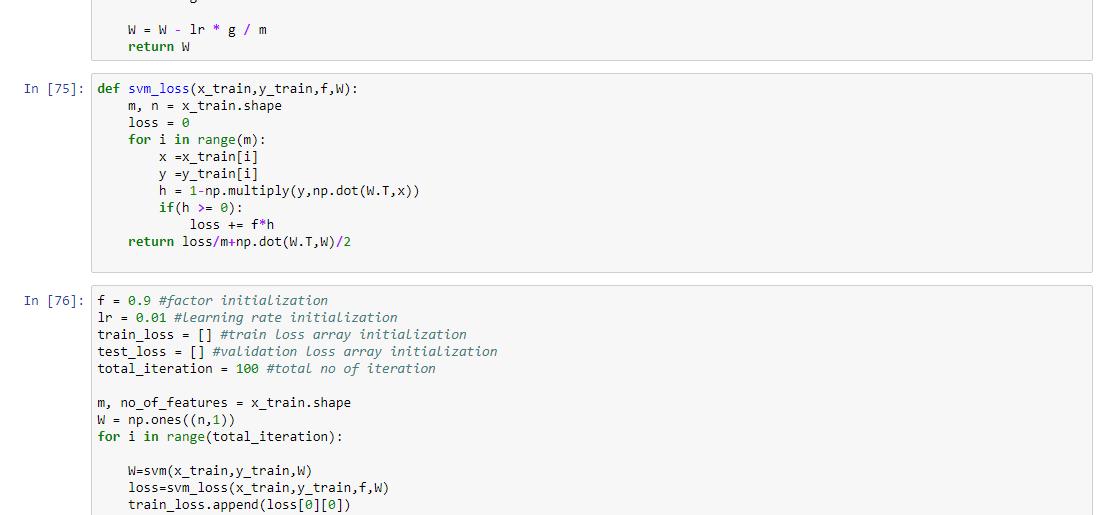
8: Get the loss Ltrain under the training set and Lvalidation by validating under validation set and draw graph of Ltrain and Lvalidation with the number of iterations.

1. **Code:**

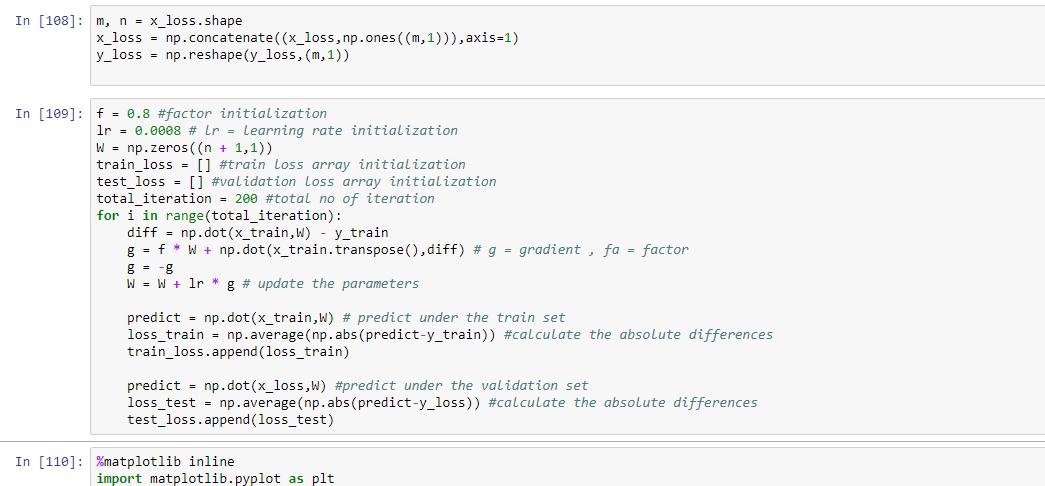
**The first one:**



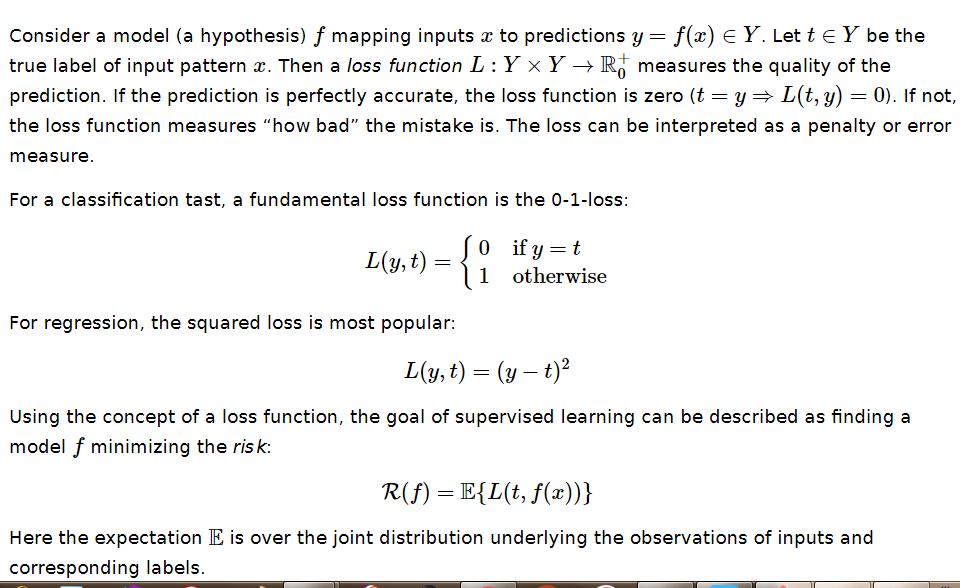
**Code for the 2nd one:**



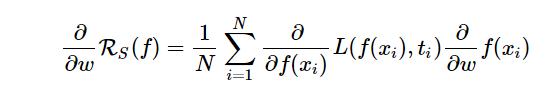
1. **Selection of validation (hold-out, cross-validation, k-folds cross-validation, etc.):** K affects accuracy and generalization, and this may depend on the learning algorithm, but it definitely affects the computational complexity almost linearly (asymptotically, linearly) for training algorithms with algorithmic complexity linear in the number of training instances. The computational time for training increases K-1 times if the training time is linear in the number of training instances. So for small training sets I'd consider the accuracy and generalization aspects, especially given that we need to get the most out of a limited number of training instances.
2. **The initialization method of model parameters:**



**10. The selected loss function and its derivatives:**

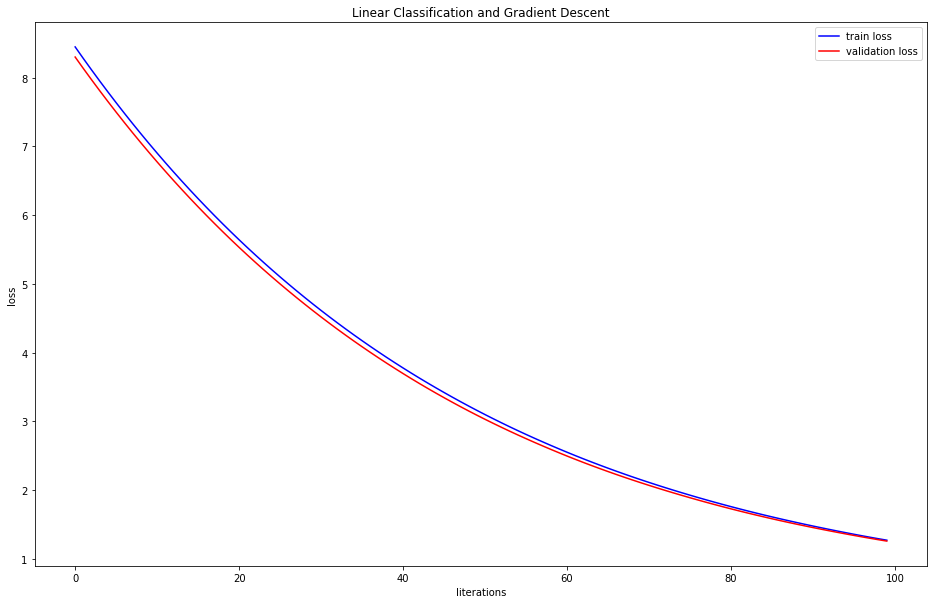
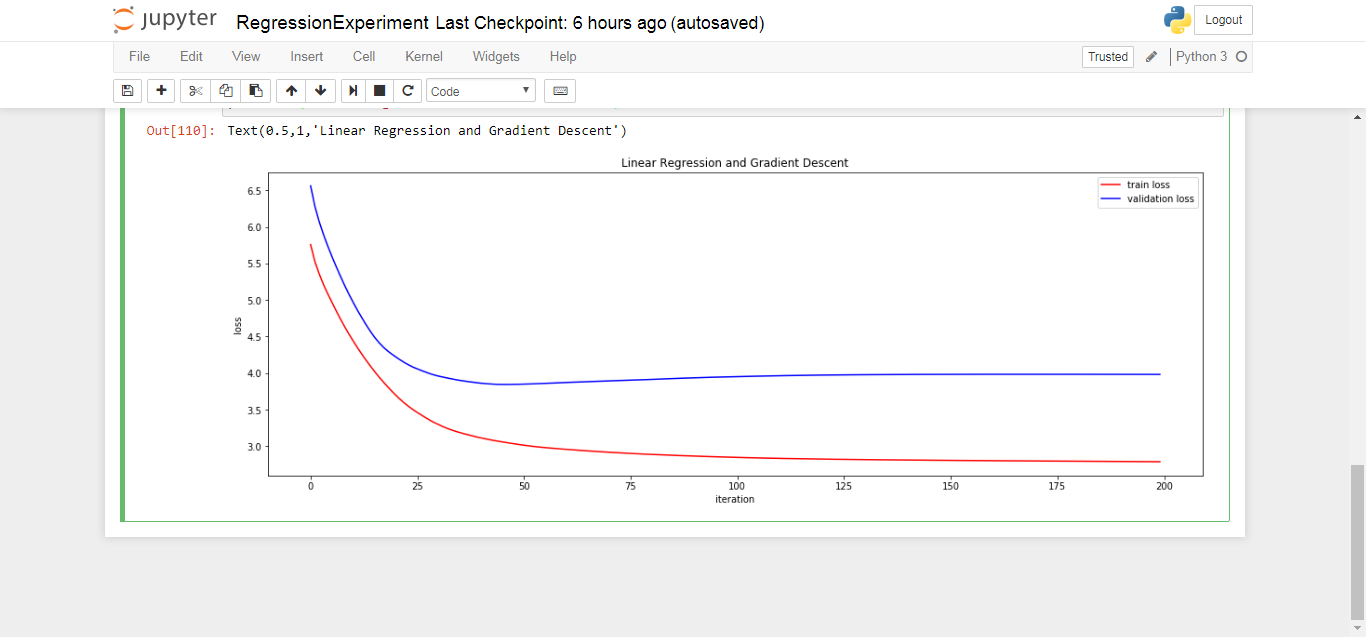


**Derivatives :**



1. **Experimental results and curve:**

Hyper-parameter selection (η, epoch, etc.): epoch = 100 and 200



1. **Results analysis:**

From the graph its easily understandable that the train curve and the test curve are almost same.

**13. Similarities and differences between linear regression and linear classification:** Linear regression and linear classification are both related to prediction, where regression predicts a value from a continuous set, whereas classification predicts the 'belonging' to the class.

For example, the price of a house depending on the 'size' (in some unit) and say 'location' of the house, can be some 'numerical value' (which can be continuous): this relates to regression.

Similarly, the prediction of price can be in words, viz., 'very costly', 'costly', 'affordable', 'cheap', and 'very cheap': this relates to classification.

Each class may correspond to some range of values.

Regression: the output variable takes continuous values.

Classification: the output variable takes class labels.

1. **Summary:**

In this experiment I discovered the simple linear regression model and how to train it using gradient descent.

I work through the experiment of the update rule for gradient descent. I also learned how to make predictions with a learned linear regression model.