

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **李沐苡**

**Student ID 201530612019**

**E-mail** [**1051072143@qq.com**](mailto:1051072143@qq.com)

**Tutor**  谭明奎

**Date submitted** **2017.12.08**

**Part One：Basic Information**

1. Topic: Linear Regression，

Linear Classification and Gradient Descent

2. Time: 2017.12.02

3. Reporter: MuyiLi（李沐苡）

4. Purposes:

4.1 Further understand of linear regression and gradient descent.

4.2 Conduct some experiments under small scale dataset.

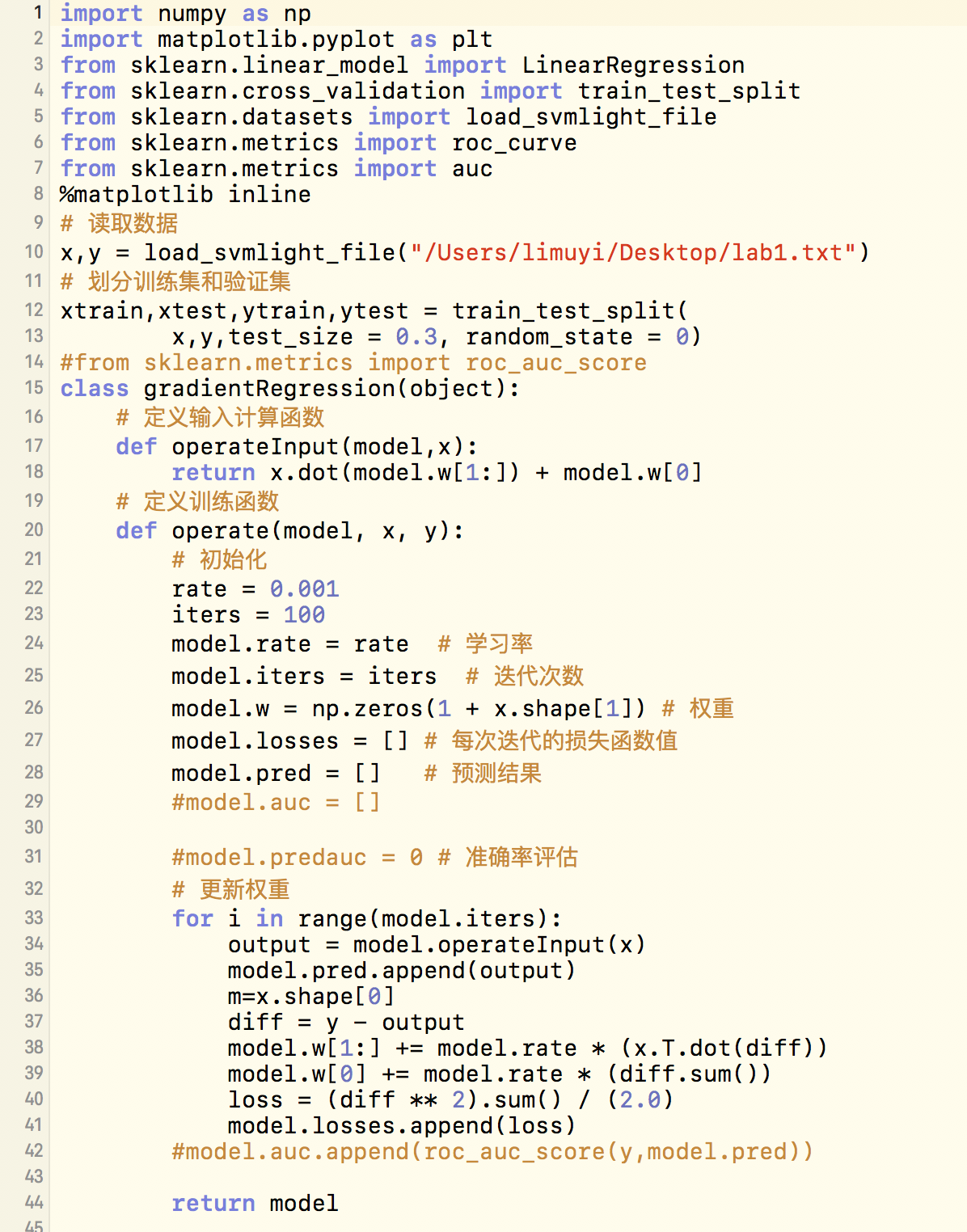
4.3 Realize the process of optimization and adjusting parameters.

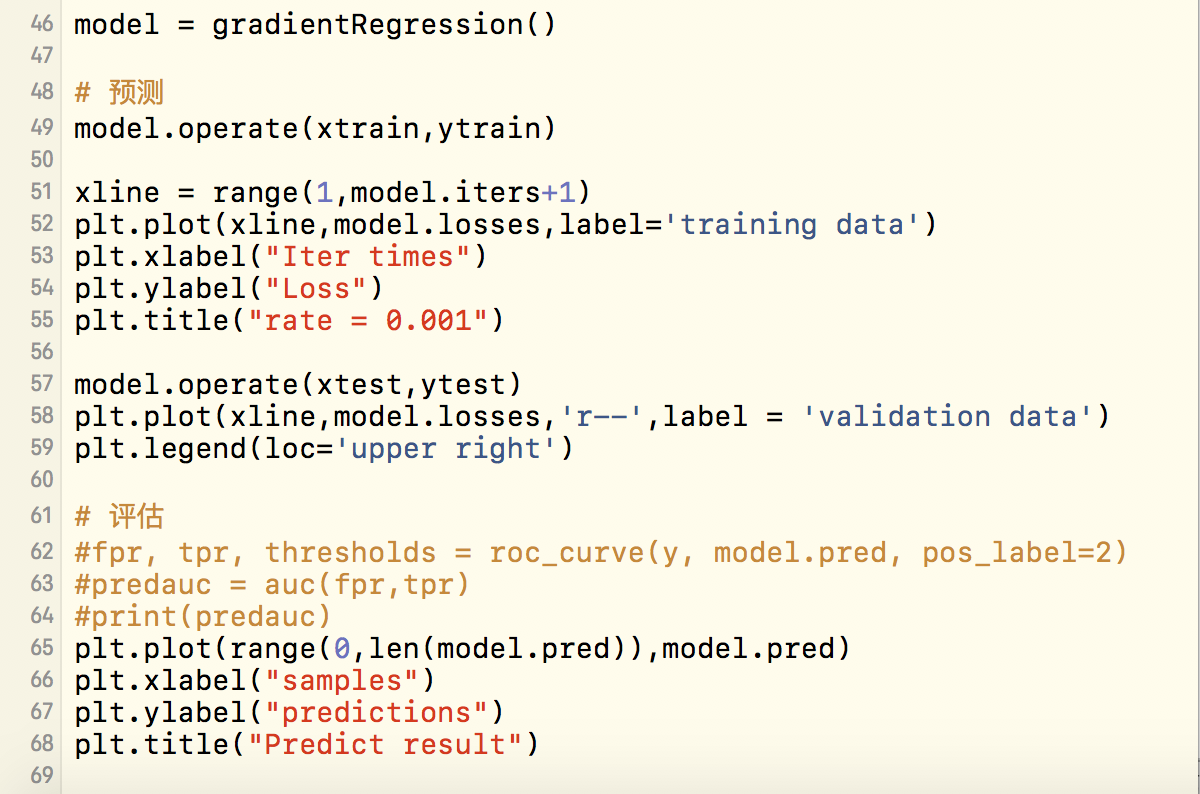
5. Data sets and data analysis:

|  |  |  |  |
| --- | --- | --- | --- |
| Experiment | Lisbsvm | Samples | Features |
| Linear Regression | Housing | 506 | 13 |
| Linear Classification | Australian | 690 | 14 |

**Part Two：Regression Experiment**

1. Experimental steps:
   1. Load the experiment data with load\_svmlight\_file function.
   2. Divide dataset only into training set and validation set with train\_test\_split function.
   3. Initialize linear model parameters.
   4. Choose loss function and derivation.
   5. Calculate gradient grad toward loss function from all samples.
   6. Denote the opposite direction of gradient grad as dire
   7. Update model: W += rate \* dire, in which rate means learning rate, a hyper-parameter that we can adjust.
   8. Predict and separately get the loss under the train set and validation set.
   9. Loop step 5 to 8 for several times, and draw graph of Losstrain and Lossvalidation with the number of iterations.
2. Code:





1. Selection of validation： cross-validation
2. The initialization method of model parameters:

all parameter into zero

1. The selected loss function and its derivatives:

h(x) = w0\*1 + w1\*x0 + w2\*x1 +...+ w13\*x12 = SUM (wj\*xj-1)+w0

loss = SUM [(h(xi)-yi)^2] / 2

gradj = SUM（h（xi）- yi）\* Xij

direj = - gradj

wj = wj-1 + rate \* direj

-= rate \* gradj

1. Experimental results and curve:

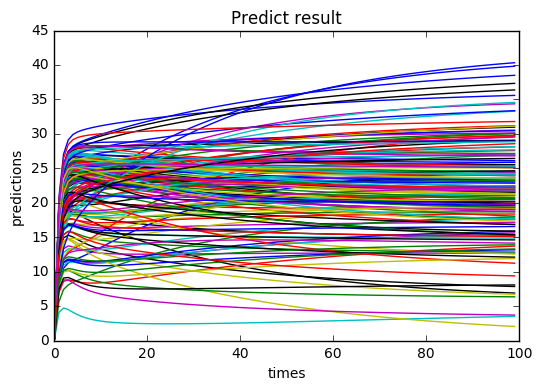
6.1 Hyper-parameter selection (η, epoch, etc.):

η = 0.001, epoch = 100

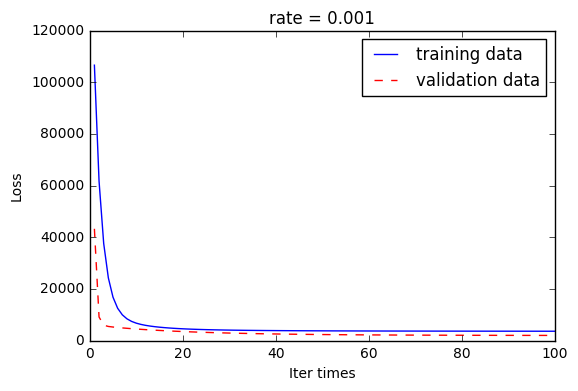
6.2 Assessment Results (based on selected validation):

as shown in the project file

6.3 Predicted Results (Best Results):



6.4 Loss curve:



1. Results analysis:

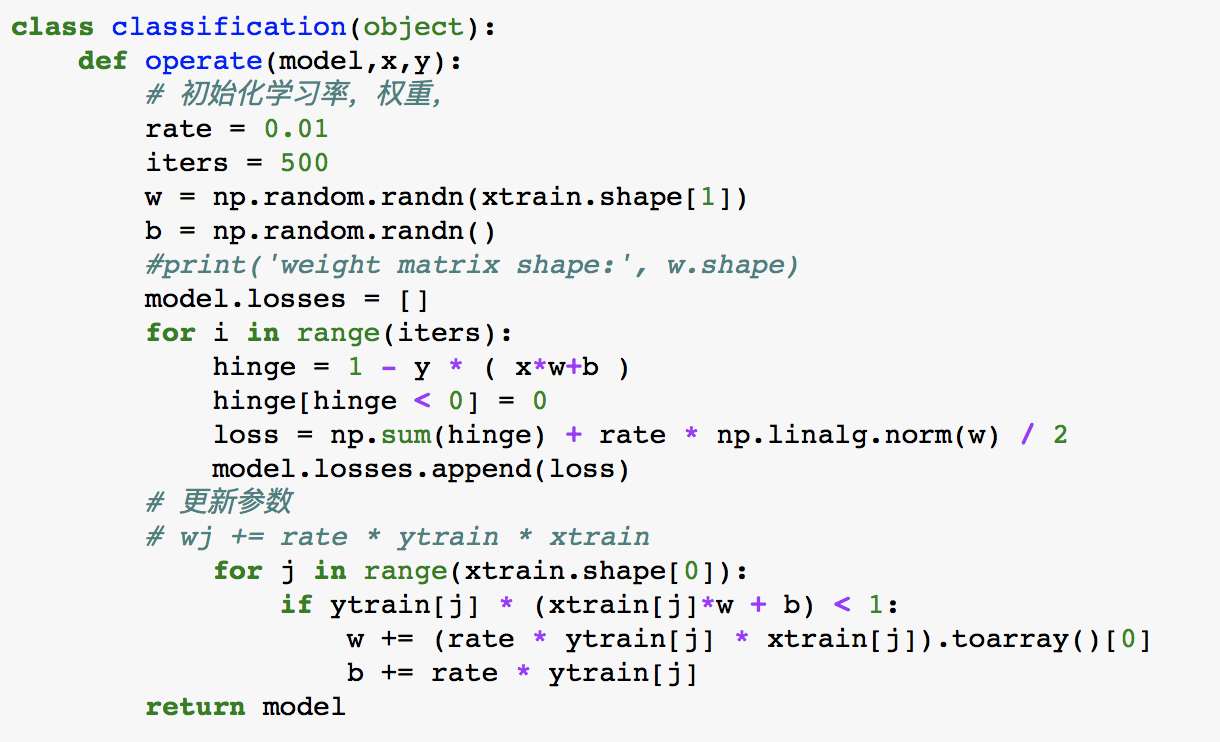
|  |  |  |
| --- | --- | --- |
| Rate | Validation loss | Predict results |
| 0.01 |  | RegressionExperiment/0_01_pred.png |
| 0.001 |  | RegressionExperiment/0_001_pred.png |
| 0.0001 |  | RegressionExperiment/0_0001_pred.png |

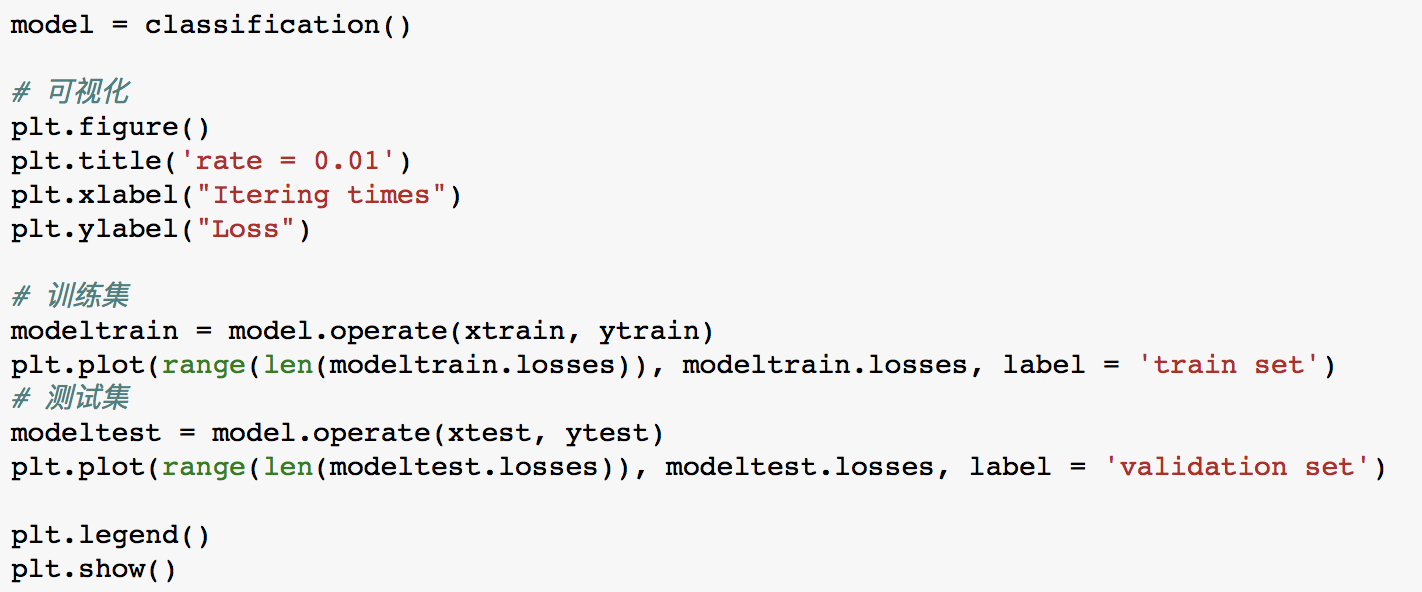
During the experiments we found if η is too small, the loss function has a slow convergence, and if η is too large, the loss function may not decrease on every iteration, may not converge.

The more parameters, the more complex the model, the more complicated the model is.

**Part Three：Regression Experiment**

1. Experimental steps:
   1. Load the experiment data.
   2. Divide the dataset into training set and validation set.
   3. Initialize SVM model parameters.
   4. Loss function and derivation.
   5. Calculate gradient(grad) toward loss function from all samples.
   6. Denote the opposite direction of grad as dire.
   7. Update model: Wj = Wj-1 + rate \* dire.
   8. Select appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative. Get the loss under the training set and validation set.
   9. Repeat step 5 to 8 for several times, and draw graph as before.
2. Key Code:





1. Selection of validation: Cross-validation.
2. The initialization method of model parameters: All zeros.
3. The selected loss function and its derivatives:

5.1 for i in range(iters):

hinge = 1 - y \* ( x\*w+b )

hinge[hinge < 0] = 0

loss = np.sum(hinge) + rate \* np.linalg.norm(w) / 2

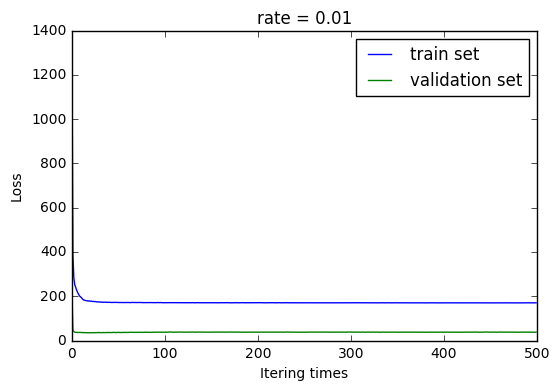
model.losses.append(loss)

5.2 wj += rate \* ytrain \* xtrain

1. Experimental results and curve:
   1. Hyper-parameter selection (η, epoch, etc.):

η= 0.01， epoch = 500

* 1. Assessment Results (based on selected validation): in the project file
  2. Loss curve:



1. Results analysis:

|  |  |  |
| --- | --- | --- |
| η | Testsize | Loss value |
| 0.1 | 0.3 | RegressionExperiment/c-r0.1-tt0.3-mix.png |
| 0.1 | 0.2 | RegressionExperiment/c-r0.1-tt-0.2-mix.png |
| 0.01 | 0.2 | RegressionExperiment/c-r0.01-tt0.2-mix.png |
| 0.001 | 0.2 | RegressionExperiment/c-r0.001-tt0.2-mix.png |

If I use regression to operate on the dataset of lab2, here come some problems. Because regression is not enough to deal with the dataset like this.

And the result is shown as below:

|  |  |  |
| --- | --- | --- |
| η | Loss value | Pred result |
| 0.01 | ../../../../../0.01%20test0.3%20lab2.png | ../../../../../0.01%200.3pred啦吧2.png |
| 0.001 | ../../../../../0.001reg2mix.png | ../../../../../0.001pred2.png |

**Part Three：**

1. Similarities and differences between linear regression and linear classification:

|  |  |  |
| --- | --- | --- |
|  | Linear regression | Linear classification |
| Label (y) | Often used to analyze the relationship between two variables, x and y. | Labels are more discrete.  More of it is (certain area of x) corresponds to (a y). |
| Steps | 1 How to build a rational model.  2 The loss function.  3 Specific techniques to solve the optimal parameters. | |
| Use | Both are fitting(matching) of the model. | |

1. Summary:

Regression analysis is often used to analyze the relationship between two variables, x and y. If the relationship is linear, we will choose a proper linear model to fit the data dots. Then we have to match the model with data using loss function, and iterate over the parameters to get the best values by using gradient regression. When the labels become more discrete, single linear model is not enough, the loss function to parameters are non-convex, therefore, it’s hard to get the global min of loss. So that we use loss function with log, and the value of loss can go down.