

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **钟恩俊**

**Student ID 201530613818**

**E-mail 286843911@qq.com**

**Tutor**  **Prof.Mingkui Tan**

**Date submitted** **2017.12.04**

1. **Topic:**

**Linear Regression, Linear Classification and Gradient Descent**

**2. Time: 2017.12.04**

**3. Reporter: 钟恩俊**

**4. Purposes:**

1. Further understand of linear regression and gradient descent.
2. Conduct some experiments under small scale dataset.
3. Realize the process of optimizatio n and adjusting parameters.

**5. Data sets and data analysis:**

Linear Regression uses Housing in LIBSVM Data, including 506 samples and each sample has 13 features.

Linear classification uses australian in LIBSVM Data, including 690 samples and each sample has 14 features.

**6. Experimental steps:**

Linear Regression and Gradient Descent

1. Load the experiment data. You can use load\_svmlight\_file function in sklearn library.
2. Devide dataset. You should divide dataset into training set and validation set using train\_test\_split function. Test set is not required in this experiment.
3. Initialize linear model parameters. You can choose to set all parameter into zero, initialize it randomly or with normal distribution.
4. Choose loss function and derivation: Find more detail in PPT.
5. Calculate gradient G toward loss function from all samples.
6. Denote the opposite direction of gradient G as D .
7. Update model: .η is learning rate, a hyper-parameter that we can adjust.
8. Get the loss under the training set and by validating under validation set.
9. Repeat step 5 to 8 for several times, and drawing graph of as well as with the number of iterations.

Linear Classification and Gradient Descent

1. Load the experiment data.
2. Divide dataset into training set and validation set.
3. Initialize SVM model parameters. You can choose to set all parameter into zero, initialize it randomly or with normal distribution.
4. Choose loss function and derivation: Find more detail in PPT.
5. Calculate gradient G toward loss function from all samples.
6. Denote the opposite direction of gradient G as D.
7. Update model: .η is learning rate, a hyper-parameter that we can adjust.
8. Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative. Get the loss under the trainin set and by validating under validation set.
9. Repeat step 5 to 8 for several times, and drawing graph of as well as with the number of iterations.

**7. Code:**

Linear Regression Codesee in the RegressionExperiment.ipynb

Linear Classification Code see in the ClassificationExperiment.ipynb

**8. Selection of validation (hold-out, cross-validation, k-folds cross-validation, etc.): hold-out**

**9. The initialization method of model parameters:**

Linear Regression:

W (the weight of X) is initialized as 0. Iteration of training is 20 times.

The data set is randomly split into training set and validation set(3:1)

Linear Classification:

W (the weight of X) is initialized as 1.

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

Linear Regression

Loss function:

Derivatives:

Linear Classification

Loss function: hingeloss

Derivatives:

**11. Experimental results and curve:**

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

Linear Regression: LearningRate=0.0001 regularization parameter=0

Linear Classification:LearningRate=0.001

## Assessment Results (based on selected validation):

Linear Regression:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration times | Learning rate | lambda | Training set loss | Validation set loss |
| 20 | 0.001 | 10 | 35.178861022946322 | 48.053783883578973 |
| 20 | 0.001 | 0 | 26.835766814645478 | 27.356720680177357 |
| 20 | 0.0001 | 0 | 64.1792060836015 | 52.366920424296133 |
| 40 | 0.001 | 0 | 23.639819923173835 | 24.262870573056755 |
| 100 | 0.0005 | 0 | 23.267028580865851 | 23.753491557987342 |

Linear Classification:

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | Learning rate | Training set error rate | Validation set error rate |
| 20 | 0.001 | 0.14906832298136646 | 0.11594202898550725 |
| 20 | 0.0001 | 0.3084886128364389 | 0.30917874396135264 |
| 100 | 0.001 | 0.14492753623188406 | 0.13043478260869565 |

## Predicted Results (Best Results):

Linear Regression:

Best loss in training set is 23.267028580865851

Best loss in validation set is 23.753491557987342

Linear Classification:

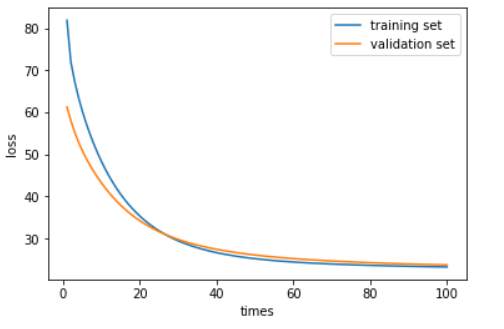
Best loss in training set is 0.14906832298136646

Best loss in validation set is 0.11594202898550725

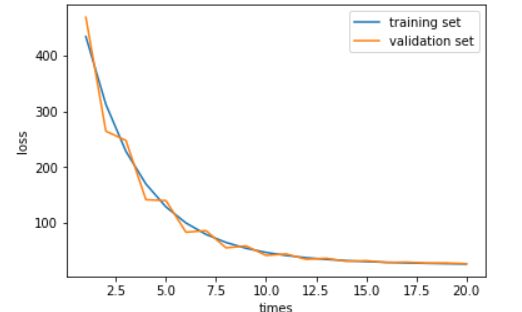
## Loss curve:

**12. Results analysis:**

Linear Regression:

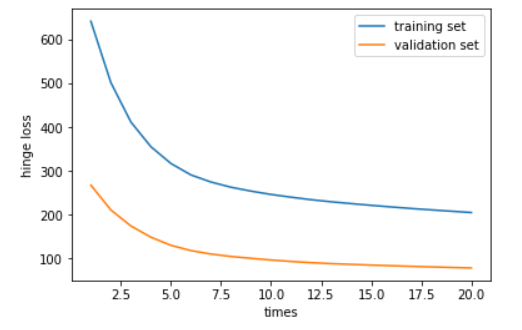


The loss reduce with the iteration times increase indicate that the algorithm is correct. I try different learning rate and find that the smaller learning rate can make the curve smoothly. The large learning perform like this:



But I can’t see the difference with the regularization parameter lambda change, maybe the model is too simple.

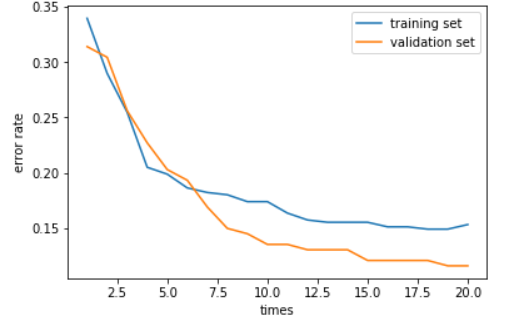
Linear Classification:



At first, I use the hingeloss to perform the result.The tendency of two lines are both decreasing. But two lines are far away from each other. Because the number of two sets are different.

To see the perform on classification, I use error rate as loss

This curve use the error rate as loss:



**13. Similarities and differences between linear regression and linear classification:**

Simailarities: both linear regression and linear classification are linear model.

Differences: linear regression solve the regression problem while linear classification solve the classification. The label in linear regression is continuous but discrete in linear classification.

**14. Summary:**

Linear Regression is used to slove the regression problem with the continuous label. Linear Classification is used to slove the binary classification problem with the discrete label.

The learning rate represent the step size of descent. The small learning rate slow down the learning speed. The large learning rate may jump out of the minimum, so the learning curve will look like a wave.