

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **邓嘉乐**

**Student ID 201530611371**

**E-mail sewlwn@mail.scut.edu.cn**

**Tutor**   **谭明奎**

**Date submitted** **2017.12.08**

1. **Topic:**

Linear Regression, Linear Classification and Gradient Descent

**2. Time:** 2017.12.02

**3. Reporter:** 邓嘉乐

**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:**

Linear Regression uses [Housing](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html#housing) in [LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/), including 506 samples and each sample has 13 features.

Linear classification uses [australian](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html" \l "australian" \t "_blank) in [LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/), including 690 samples and each sample has 14 features.

Both are divided into training set, validation set by sklearn using  [train\_test\_split](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html) .

**6. Experimental steps:**

***Linear Regression and Gradient Descent***

1. Load the experiment data. You can use [load\_svmlight\_file](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html) function in sklearn library.
2. Devide dataset. You should divide dataset into training set and validation set using [train\_test\_split](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html) 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: *Wt*=*Wt*−1+*ηD* . *η* is learning rate, a hyper-parameter that we can adjust.
8. Get the loss *Ltrain* under the training set and *Lvalidation* by validating under validation set.
9. Repeate step 5 to 8 for several times, and **drawing graph of** *Ltrain* **as well as** *Lvalidation* **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: *Wt*=*Wt*−1+*ηD*. *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 *Ltrain* under the trainin set and *Lvalidation* by validating under validation set.
9. Repeate step 5 to 8 for several times, and **drawing graph of** *Ltrain* **as well as** *Lvalidation* **with the number of iterations**.

**7. Code:**

***Linear Regression and Gradient Descent***

#gradient descent

y\_train\_pred = x\_train\*W;

deltaW = learning\_rate\*(x\_train.T\*(y\_train - y\_train\_pred))

W = W + deltaW

loss\_train.append(1 / 2 \* (np.linalg.norm(y\_train - x\_train\*W))\*\*2)

***Linear Classification and Gradient Descent***

#gradient descent

y\_train\_pred = x\_train\*W;

y\_train\_pred = x\_train\*W;

g = 1-np.multiply(y\_train,y\_train\_pred)

idx\_1 = np.where(g>=0)[0]

idx\_2 = np.where(g<0)[0]

deltaW\_2 = -W.T

deltaW\_1 = C\*y\_train.T\*x\_train+deltaW\_2

deltaW = learning\_rate\*((deltaW\_1\*len(deltaW\_1)+deltaW\_2\*len(deltaW\_2))/n)

W = W+deltaW.T

loss\_train.append((np.linalg.norm(W))\*\*2/2+C\*sum(np.maximum(zeros((n,1)),g)))

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

**9. The initialization method of model parameters:** All parameters are set into zero in both experiment.

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

***Linear Regression and Gradient Descent***

*Loss function:*

*Derivatives****:***

***Linear Classification and Gradient Descent***

*Loss function:*

*Derivatives:*

*And*

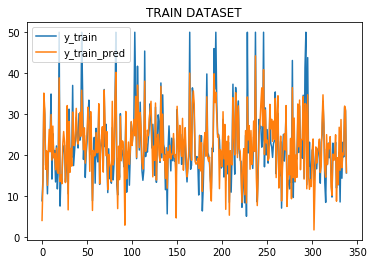
**11. Experimental results and curve:**

***Linear Regression and Gradient Descent***

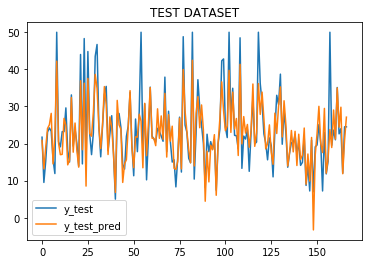
## Hyper-parameter selection (η, epoch, etc.):

η= 0.00005 , epoch = 1 , iteration = 10000

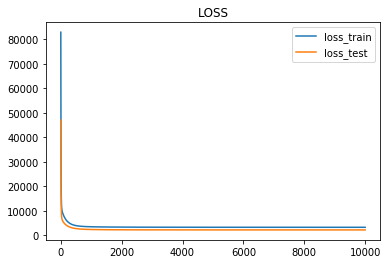
## Assessment Results (based on selected validation):



## Predicted Results (Best Results):



## Loss curve:



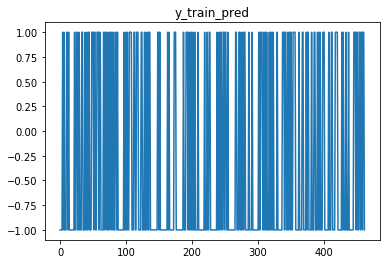
***Linear Classification and Gradient Descent***

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

η= 0.03 , epoch = 1 , iteration = 100000 , C = 0.001

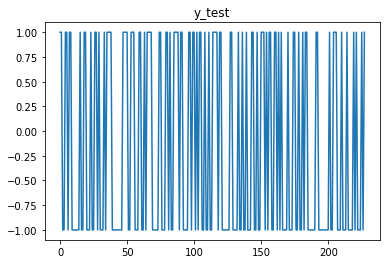
## Assessment Results (based on selected validation):

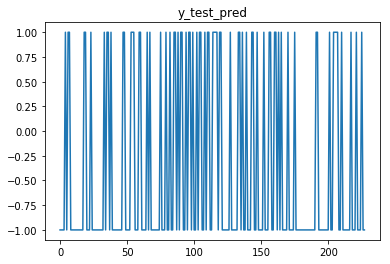


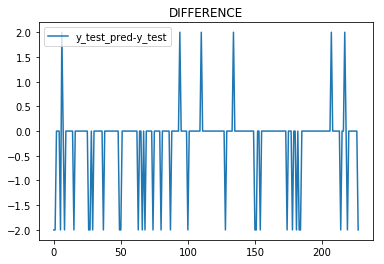


## Predicted Results (Best Results):

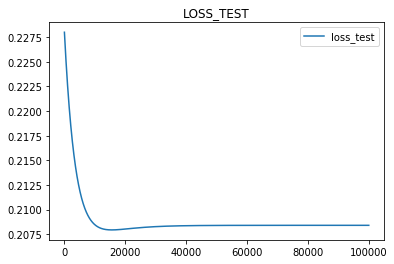
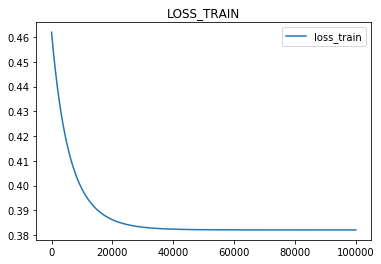
The accuracy: 0.8421052631578947







## Loss curve:



1. **Results analysis:**

With the increase of iterations, both two loss lines reduces rapidly initially and slope becomes less. Finally, the slope shock around zero value and the loss lines seem to be flat.

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

Similarities:

* 1. The methods to update the model parameters are the same.

Differences:

1. The selected loss function and its gradient.
2. Linear classification can use accuracy to measure the fitting model.
3. Linear classification should define a threshold to distinguish the positive or negative sample.

**13. Summary:**

The ideal loss curve may be a decreasing curve with a increasing slope (and the slope finally is around zero). And learning rate can not be too large (the loss can not convergent) or too small (gradient descent very slowly). At the same time, the loss is defined by the loss function and not always smaller is better: when in linear classification, the loss become smaller obviously with the decrease of C but the accuracy may be worse.