



华南理工大学

South China University of Technology

---

## The Experiment Report of *Machine Learning*

---

**SCHOOL:** SCHOOL OF SOFTWARE ENGINEERING

**SUBJECT:** SOFTWARE ENGINEERING

*Author:*  
Lizhao Liu

*Supervisor:*  
Mingkui Tan

*Student ID:*  
201730683109

*Grade:*  
Undergraduate

November 12, 2019

# Logistic Regression and SVM

**Abstract**—In this report, we solve the binary classification problem in a9a dataset by Logistic Regression (LR) and Support Vector Machine (SVM).

We perform experiments on four aspects:

1. Hyper-parameter tuning in SVM.
2. Loss function comparison in SVM.
3. Initializer comparison in both LR and SVM.
4. Optimizer comparison in both LR and SVM.

## I. INTRODUCTION

**L**OGISTIC Regression(LR) is an extension of linear regression, which use  $\sigma$  function as activation to map the output of linear regression into a interval (0, 1). So that LR can output the probability in binary classification. SVM is also an extension of linear regression. Based on the idea "large margin classification", SVM can perform very well in many classification task. Even now, LR and SVM are widely used in academic and industry community. These two algorithms are classic, so we fully explore them by solving a9a binary classification task. Beyond that, we will explore the initializer and optimizer selection in both LR and SVM.

## II. METHODS AND THEORY

In this part, we first define logistic regression and LR's loss function Binary Cross Entropy (BCE). Then we define SVM and SVM's loss function Hinge Loss (HL).

### A. Logistic Regression

Given dataset  $D = \{x_i, y_i\}_{i=1}^m$ , where  $x_i \in \mathbf{R}^n$  and  $y_i \in \{0, 1\}$ . Mathematically, for a single training datapoint  $(x, y)$ , LR assumes:

$$P(Y = 1|X = x) = \sigma(z) \quad (1)$$

$$P(Y = 0|X = x) = 1 - \sigma(z) \quad (2)$$

$$z = F(x; W) \quad (3)$$

where  $F(x; W)$  is linear regression parameterized by  $W$  and  $\sigma(\cdot)$  is sigmoid function, where  $\sigma(z) = \frac{1}{1+e^{-z}} \in (0, 1)$ .

### B. Binary Cross Entropy

Logistic regression assumes that each datapoint is independent, so the likelihood of all data is:

$$\begin{aligned} L(W) &= \prod_{i=1}^m P(Y = y_i|X = x_i) \\ &= \prod_{i=1}^m \sigma(z)^{y_i} (1 - \sigma(z))^{1-y_i} \end{aligned} \quad (4)$$

So our goal is to maximize the likelihood of data, in practice we maximize the log likelihood.

$$\begin{aligned} LL(W) &= \log \prod_{i=1}^m \sigma(z)^{y_i} (1 - \sigma(z))^{1-y_i} \\ &= \sum_{i=1}^m (y_i \sigma(z) + (1 - y_i)(1 - \sigma(z))) \end{aligned} \quad (5)$$

When using mini batch gradient descent (MBGD) algorithm, we want to minimize something, so we minimize the negative log likelihood.

$$NLL(W) = -\sum_{i=1}^m (y_i \sigma(z) + (1 - y_i)(1 - \sigma(z))) \quad (6)$$

Equation 6 is also called binary cross entropy.

### C. SVM and Hinge Loss

Given dataset  $D = \{x_i, y_i\}_{i=1}^m$ , where  $x_i \in \mathbf{R}^n$  and  $y_i \in \{-1, 1\}$ . SVM is based on linear regression. When given a data point, it wants:

$$\begin{cases} w^T x + b \geq 1, & \text{if } y = 1 \\ w^T x + b \leq -1, & \text{if } y = -1 \end{cases} \quad (7)$$

Equation 7 can also be written to:

$$y(w^T x + b) \geq 1 \quad (8)$$

Other than make the classification correct, SVM also wants to make the classification boundary as safe as possible. To achieve that, it wants to make the margin between the positive class and negative class as large as possible.

The positive boundary of SVM is  $w^T x + b = 1$  and negative boundary of SVM is  $w^T x + b = -1$ . So the margin between this two boundary is:

$$\text{Margin} = \frac{2}{\|w\|} \quad (9)$$

In practice, instead of maximize Equation 9, we want to minimize:

$$\frac{w^T w}{2} \quad (10)$$

So, the optimization problem for SVM is:

$$\begin{aligned} \min_{w,b} \quad & \frac{w^T w}{2} \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1, i = 1, 2, \dots, m \end{aligned} \quad (11)$$

Moreover, data point may not be linearly separable. Even the data point is linear separable, the outliers in dataset will hurt the performance of SVM. So, SVM introduce slack variable  $\xi_i$  for each data point, which represents how much data point  $i$  is on the wrong side of margin boundary.

1. if  $\xi_i = 0$ , it is ok.

2. if  $0 < \xi_i < 1$ , the data point is correctly classified, but with a smaller margin than  $\frac{1}{\|w\|}$ .
3. if  $\xi_i > 1$ , the data point is incorrectly classified.

Then the optimization problem becomes:

$$\begin{aligned} \min_{w,b} \quad & \frac{w^T w}{2} + \frac{C}{m} \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i, i = 1, 2, \dots, m \end{aligned} \quad (12)$$

As  $\xi_i \geq 1 - y_i(w^T x_i + b)$ ,  $i = 1, 2, \dots, m$ , so

$$\xi_i = \begin{cases} 1 - y_i(w^T x_i + b), & \text{if } 1 - y_i(w^T x_i + b) \geq 0 \\ 0, & \text{if } 1 - y_i(w^T x_i + b) < 0 \end{cases} \quad (13)$$

So

$$\xi_i = \max(0, 1 - y_i(w^T x_i + b)) \quad (14)$$

Equation 14 is the Hinge Loss (HL). Finally the optimization problem for SVM becomes

$$\min_{w,b} \quad \frac{w^T w}{2} + \frac{C}{m} \sum_{i=1}^m \max(0, 1 - y_i(w^T x_i + b)) \quad (15)$$

We do a small modification for Equation 15, so that we can see the role of  $C$  in SVM.

$$\min_{w,b} \quad \lambda w^T w + \frac{1}{m} \sum_{i=1}^m \max(0, 1 - y_i(w^T x_i + b)) \quad (16)$$

where  $\lambda = \frac{1}{2C}$

Here, we can see that  $\lambda$  e.g.  $\frac{1}{2C}$  is the regularizer, like Ridge Regression (RR). So, the magnitude of  $W$  is in direct proportion to  $C$ .

### III. EXPERIMENTS

#### A. Dataset

We conduct all the experiments on a9a dataset, which has total 48842 examples and each example's form is  $(x, y)$ , where  $x \in \mathbf{R}^{123}$  and  $y \in \{0, 1\}$  (in LR) and  $\{-1, +1\}$  (in SVM). We split the dataset into three part: 26049 training examples, 6512 validation examples and 16281 test examples. We normalized the  $X_{train}, X_{val}, X_{test}$  by using the mean and variance of  $X_{train}$ .

#### B. Implementation

We implement the LR, SVM, optimizers, initilizers and loss functions using python and mainly rely on the numpy package.

#### C. Hyper-parameter $C$ in SVM

In this section, we explore the influence of  $C$  in SVM. We set  $\log_{10} C$  in a coarse-grained range e.g. -2, -1, 0, 1, 2, 3, 4, and compare  $loss_{val}$ ,  $accuracy_{val}$  and  $l_2norm$  of  $w$ , where  $loss = \frac{w^T w}{2} + \frac{C}{m} \sum_{i=1}^m \max(0, 1 - y_i(w^T x_i + b))$  and  $l_2norm(w) = \sqrt{\sum_{i=1}^n w_i^2}$ . We zero initialize  $w$  and  $b$ , use SGD as optimizer with learning rate 0.001, set epoch to 200 and batch size to 64.

From Fig 1, we can see that, when  $C$  is very small, e.g. smaller than 1, the gradient descent process is very unstable and the loss is higher than other. As  $C$  goes up, the loss

decreases, e.g. from 1 to 10. But when  $C$  gets larger than 10, the loss become slightly bigger, but still better than small  $C$ , like 0.01 and 0.1. So, we can draw the conclusion that, bigger  $C$  usually better than small  $C$ , and the best  $C$  is in the middle, which is not very big but also not very small. As we can see from Fig 1,  $C = 10$  gets the lowest loss.

From Fig 2, we can see that, when  $C$  is very small, e.g. 0.01, the accracy in both training set and validation set is much lower than others. Small  $C$  like 0.01 and 0.1 get worse performance comparing with others. In training set, the accracy is very similar when  $C$  bigger than 1. In validation set,  $C = 10$  gets slightly better performance comparing with others. Best validation performance under each  $C$  setting can be found in Table I.

From Fig 3, we can see that, as  $C$  goes up,  $l_2norm(w)$  goes up except  $C = 100$ . As said above,  $\frac{1}{2C}$  can be viewed as regurilizer, so when  $C$  is bigger, the smaller of its impact on regurization, so the weight maginitude goes up.

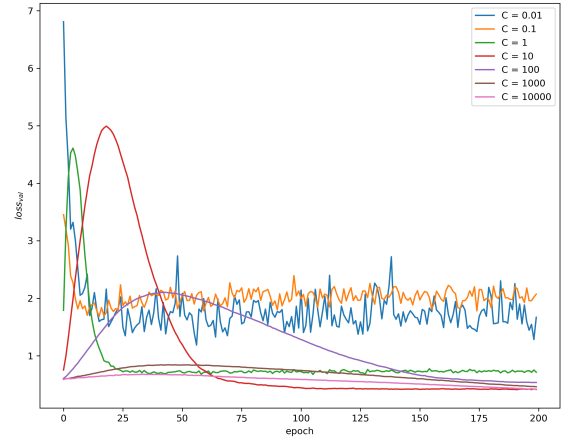


Fig. 1.  $Loss_{val}$  under different  $C$ .

TABLE I  
BEST RESULT IN VALIDATION SET UNDER DIFFERENT  $C$ 'S SETTINGS.  
BEST RESULT ARE IN BOLD.

$\log_{10} C$	-2	-1	0	1	2	3	3
$Accuracy_{val}$	84.39	84.42	84.72	<b>84.75</b>	84.73	84.56	84.49
Epoch	<b>2</b>	86	62	138	173	147	145

#### D. Loss function comparision in SVM

In this section, we conduct experiment on loss function comparision in SVM. Especially, we compare two loss function, hinge loss which is illustrated above And Exponential Loss (EL), where  $EL = \exp\{-y_i(w^T x_i + b)\}$ . More formally,  $HL(z) = \max(0, 1 - z)$  and  $EL(z) = \exp^{-z}$ . Fig 4 shows that HL and EL both have one advantage and one disadvantage. HL is stable when  $z$  is very mall, however EL become very large, which will lead to numeric unstable. EL is derivable every but HL is not derivable when  $z = 1$ . We

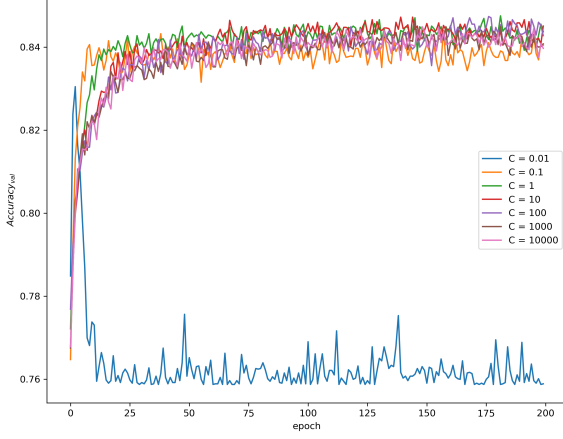


Fig. 2.  $Accuracy_{val}$  under different  $C$ .

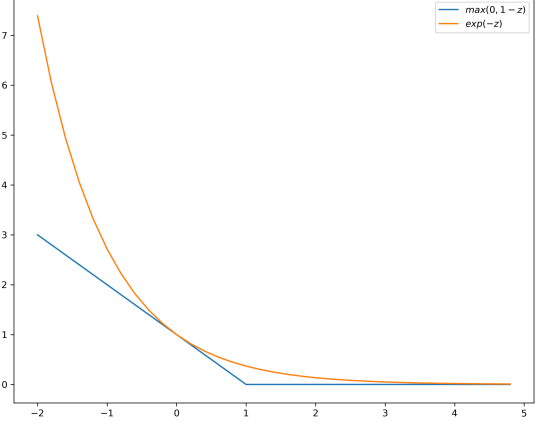


Fig. 4. The plot of HL and EL.

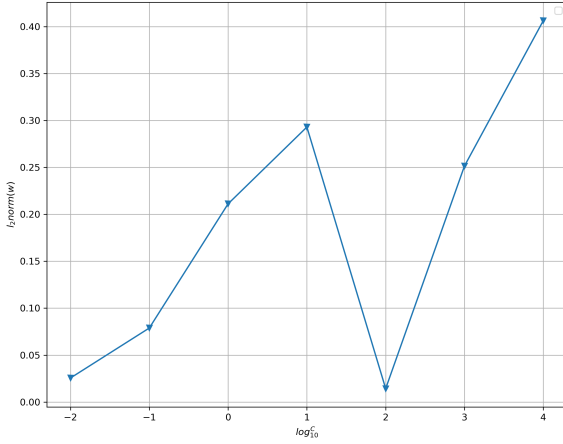


Fig. 3. The magnitude of  $W$  under different  $C$ .

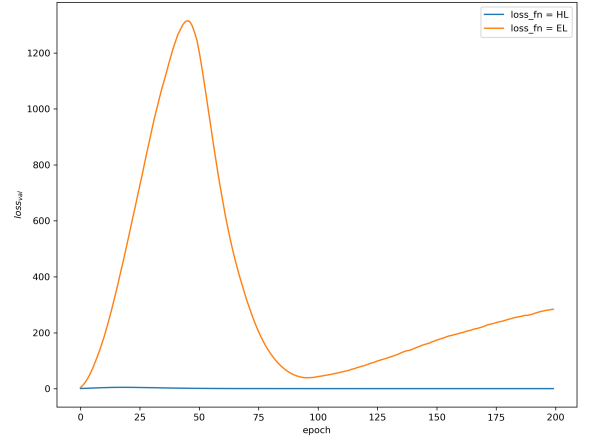


Fig. 5.  $loss_{val}$  under different loss function.

use SGD as optimizer with learning rate 0.001, set epoch to 200, batch size to 64 and  $C$  to 10.

From Fig 5 and Fig 6, we can see that the performance of HL is superior. EL has numeric unstablity problem during gradient descent process.

#### E. Initilizer comparision in both LR and SVM

In this section, we compare the performance of LR and SVM under different initilizers. We explore over ones, zeros, xavier uniform and random initializer. Ones initilizers makes the initial  $w$  to all one and Zeros initilizers makes the initial  $w$  to all zero. Xavier uniform initializer uniformly initial  $w$ , but bounded to interval  $[-n, n]$ , where  $n$  is  $W$ 's dimension. Random initializer initial  $w$  from standard normal distribution. We use SGD as optimizer with learning rate 0.001, set epoch to 200, batch size to 64 and  $C$  to 10 in SVM. We use HL as SVM's loss function and BCE as LR's loss function.

From Fig 7 and Fig 8, we can see that LR's performance is affected by the initilizer. Typically Ones initilizer brings the worst performance and Xavier uniform initilizer bring the best performance.

From Fig 9 and Fig 10, we can see that SVM is very robust to the initilizer, different initilizer didn't disturb the performance.

#### F. Optimizer comparision in both LR and SVM

In this section, we compare the performance of LR and SVM with different optimizers. We explore over Stochastic Gradient Descent (SGD), Momentum, Adam and RMSProp. We use xavier uniform initilizer, set learning rate to 0.001 epoch to 200, batch size to 64 and  $C$  to 10 in SVM. We use HL as SVM's loss function and BCE as LR's loss function.

From Fig 11 and Fig 12, we can see that LR's performance is affected by the optimizer. Typically SGD converges very

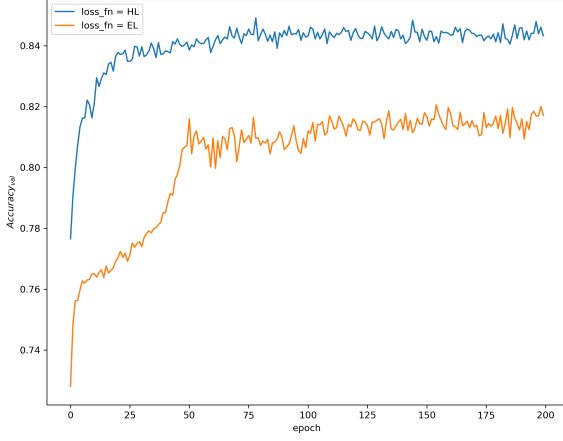


Fig. 6.  $Accuracy_{val}$  under different loss function.

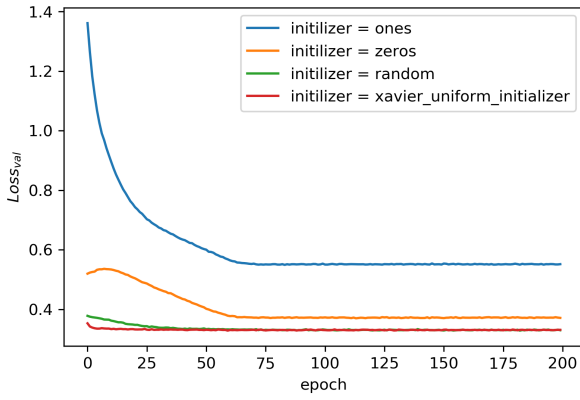


Fig. 7. LR's  $BCE_{val}$  under different initializer.

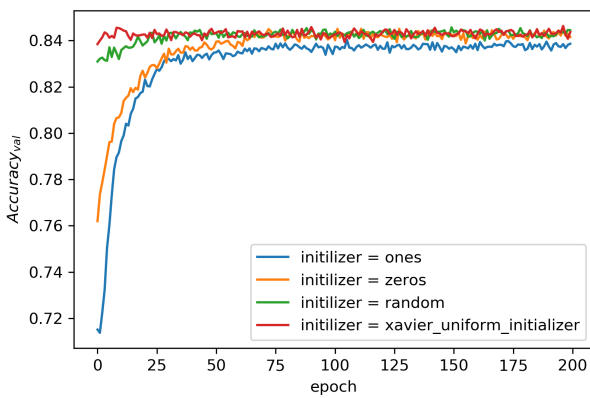


Fig. 8. LR's  $Accuracy_{val}$  under different initializer.

slowly; Momentum converges faster than SGD, but not as good as SGD's performance; both Adam and RMSProp converge very fast and get to the best performance.

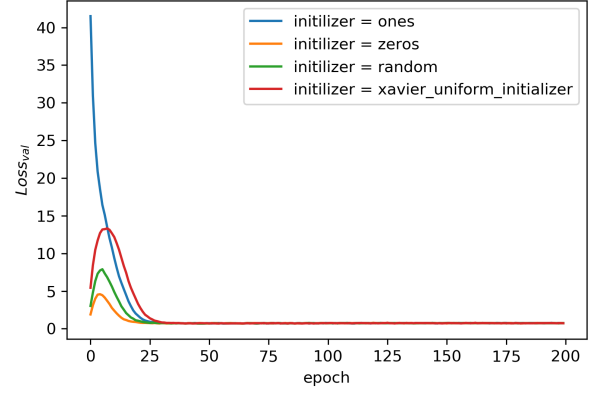


Fig. 9. SVM's  $HL_{val}$  under different initializer.

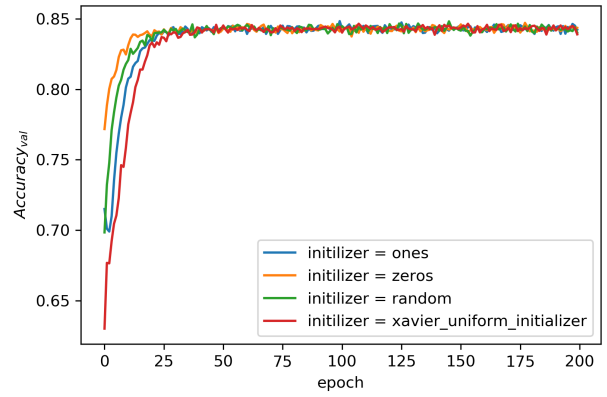


Fig. 10. SVM's  $Accuracy_{val}$  under different initializer.

From Fig 13 and Fig 14, we can see that SVM is very robust to the optimizer too, different optimizer didn't disturb the performance except Momentum which converges slower and get worse performance comparing with other optimizers.

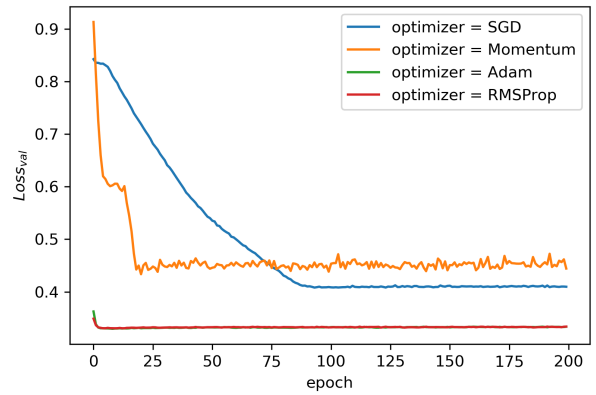


Fig. 11. LR's  $BCE_{val}$  under different optimizer.

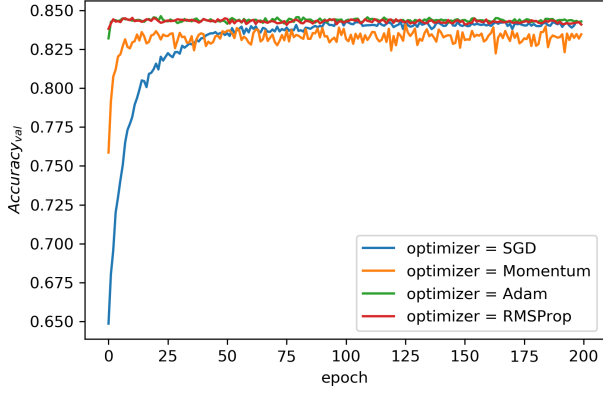


Fig. 12. LR's  $Accuracy_{val}$  under different optimizer.

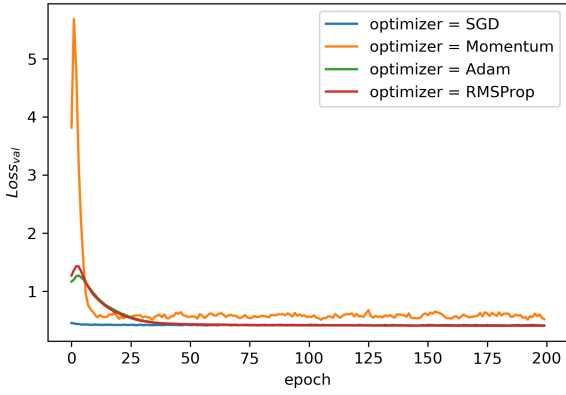


Fig. 13. SVM's  $HL_{val}$  under different optimizer.

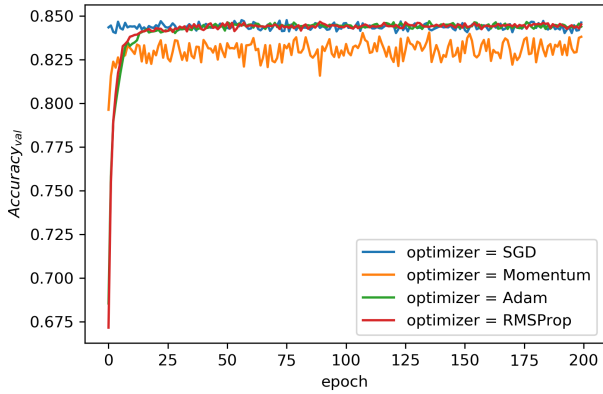


Fig. 14. SVM's  $Accuracy_{val}$  under different optimizer.

#### IV. CONCLUSION

In this report, we explore the CFS optimized LR, CFS optimized RR and MBGD optimized LR. We conduct the experiments on the CFS optimized LR and RR in terms of performance and weight magnitude. We also explore GD,

SGD, MBGD in terms of convergence and time efficiency. Moreover, we tuning the hyper-parameter learning rate in MBGD. We also compare the weight magnitude between CFS optimized LR and MBGD optimized LR. Finally, we report the performance in CFS optimized LR, RR and MBGD LR. After a series of experiments, we now get more insight of LR, RR and GD and its variants SGD and MBGD.