

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**  Tengyun Wang

**Student ID 201710106574**

**E-mail wty9391@gmail.com**

**Tutor**   **Mingkui Tang**

**Date submitted** **2017.12 .12**

1. **Topic:**

Logistic Regression, Linear Classification and Stochastic Gradient Descent

**2. Time:**

2017.12.12

**3. Reporter:**

Tengyun Wang

**4. Purposes:**

* 1. Compare and understand the difference between gradient descent and stochastic gradient descent.
  2. Compare and understand the differences and relationships between Logistic regression and linear classification.
  3. Further understand the principles of SVM and practice on larger data.

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

Experiment uses a9a of LIBSVM Data, including 32561/16281(testing) samples and each sample has 123/123 (testing) features. Please download the training set and validation set.

**6. Experimental steps:**

The experimental code and drawing are completed on jupyter.

* **Logistic Regression and Stochastic Gradient Descent**

1. Load the training set and validation set.
2. Initalize logistic regression model parameters, you can consider initalizing zeros, random numbers or normal distribution.
3. Select the loss function and calculate its derivation, find more detail in PPT.
4. Calculate gradient toward loss function from **partial samples**.
5. Update model parameters using different optimized methods(**NAG，RMSProp，AdaDelta and Adam**).
6. Select the appropriate threshold, mark the sample whose predict scores **greater than the threshold as positive, on the contrary as negative**. Predict under validation set and get the different optimized method loss ，， and .
7. Repeate step 4 to 6 for several times, and **drawing graph** of ，， and **with the number of iterations**.

* **Linear Classification and Stochastic Gradient Descent**

1. Load the training set and validation set.
2. Initalize SVM model parameters, you can consider initalizing zeros, random numbers or normal distribution.
3. Select the loss function and calculate its derivation, find more detail in PPT.
4. Calculate gradient toward loss function from **partial samples**.
5. Update model parameters using different optimized methods(**NAG，RMSProp，AdaDelta and Adam**).
6. Select the appropriate threshold, mark the sample whose predict scores **greater than the threshold as positive, on the contrary as negative.** Predict under validation set and get the different optimized method loss ，， and .
7. Repeate step 4 to 6 for several times, and drawing graph of ，， and **with the number of iterations**.

**7. Code:**

(Fill in the contents of 8-11 respectively for logistic regression and linear classification)

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

* **Logistic Regression and Stochastic Gradient Descent**

w = np.random.normal(size=(num\_features,1))

* **Linear Classification and Stochastic Gradient Descent**

w = np.random.normal(size=(num\_features,1))

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

* **Logistic Regression and Stochastic Gradient Descent**

hypothesis

loss function

loss function’s derivatives

* **Linear Classification and Stochastic Gradient Descent**

hypothesis

loss function

loss function’s derivatives

**10. Experimental results and curve:**(Fill in this content for various methods of gradient descent respectively)

* **Logistic Regression and Stochastic Gradient Descent**

## Hyper-parameter selection:

I fix some hyper-parameters and I use exhaustive grid search to tuning the hyper parameter.

Fixed hyper-parameters are:

max\_iterate=50, means that I use the whole training set 50 times to perform the gradient decent.

batch\_size = 8000

The hyper-parameters’ candidate are as follows

param\_grid = {

'NAG': {'lamda': [0.01, 0.1],

'eta': [0.01, 0.05],

'gamma': [0.8, 0.9, 0.95],

'threshold': [0.5,0.6]},

'Adadelta' : {'lamda': [0.01, 0.1],

'gamma': [0.8, 0.9, 0.95],

'threshold': [0.5,0.6]},

'RMSprop' : {'lamda': [0.01, 0.1],

'eta': [0.01, 0.05],

'gamma': [0.8, 0.9, 0.95],

'threshold': [0.5,0.6]},

'Adam' : {'lamda': [0.01, 0.1],

'eta': [0.01, 0.05],

'Adam\_beta1': [0.9, 0.95],

'Adam\_beta2' : [0.99, 0.999],

'threshold': [0.5,0.6]},

'GD' : {'lamda': [0.01, 0.1, 0.5],

'eta': [0.1, 0.2, 0.3, 0.4, 0.5],

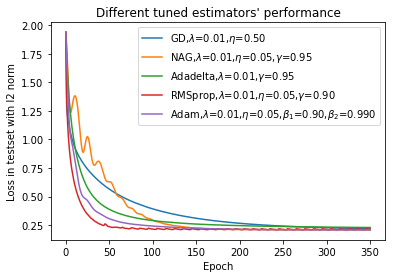
'threshold': [0.4,0.5,0.6]}}

## Predicted Results (Best Results):

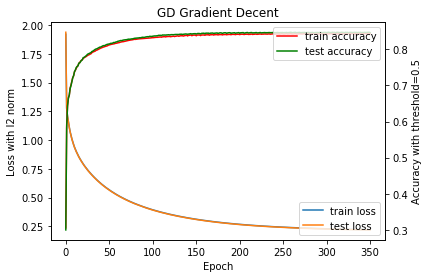
The performance of best estimators with five different optimize algorithm are as follows:

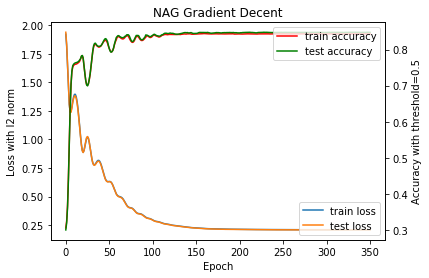
|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Test accuracy** | **Train accuracy** | **Hyper-parameters** |
| GD | 0.841927 | 0.842265 | {'eta': 0.5, 'lamda': 0.01, 'threshold': 0.5} |
| Adadelta | 0.842357 | 0.842189 | {'gamma': 0.95, 'lamda': 0.01, 'threshold': 0.5} |
| Adam | 0.844139 | **0.844477** | {'Adam\_beta1': 0.9, 'Adam\_beta2': 0.99, 'eta': 0.05, 'lamda': 0.01, 'threshold': 0.5} |
| NAG | 0.843954 | 0.844369 | {'eta': 0.05, 'gamma': 0.95, 'lamda': 0.01, 'threshold': 0.5} |
| RMSprop | **0.844354** | 0.844139 | {'eta': 0.05, 'gamma': 0.9, 'lamda': 0.01, 'threshold': 0.6} |

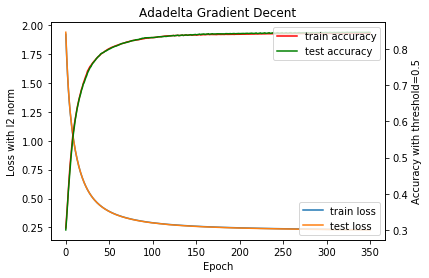
## Loss curve:

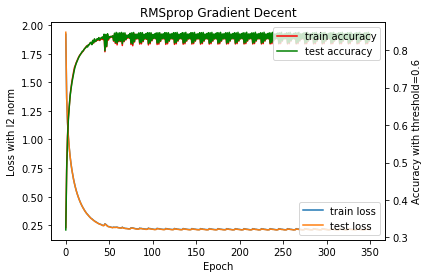


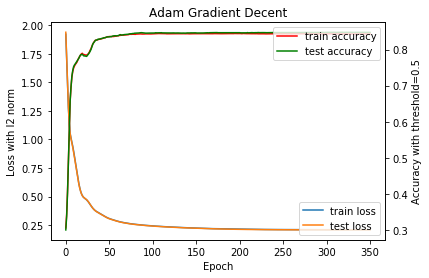
Metrics evolution process details:



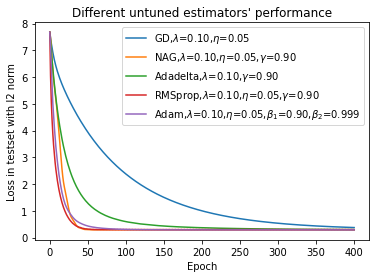








I append a experiment to show the untuned estimators performance



* **Linear Classification and Stochastic Gradient Descent**

## Hyper-parameter selection:

I fix some hyper-parameters and I use exhaustive grid search to tuning the hyper parameter.

Fixed hyper-parameters are:

max\_iterate=50, means that I use the whole training set 50 times to perform the gradient decent.

batch\_size = 8000

The hyper-parameters’ candidate are as follows

param\_grid = {

'NAG': {'lamda': [0.01, 0.1],

'eta': [0.01, 0.05],

'gamma': [0.8, 0.9, 0.95],

'threshold': [0.5,0.6]},

'Adadelta' : {'lamda': [0.01, 0.1],

'gamma': [0.8, 0.9, 0.95],

'threshold': [0.5,0.6]},

'RMSprop' : {'lamda': [0.01, 0.1],

'eta': [0.01, 0.05],

'gamma': [0.8, 0.9, 0.95],

'threshold': [0.5,0.6]},

'Adam' : {'lamda': [0.01, 0.1],

'eta': [0.01, 0.05],

'Adam\_beta1': [0.9, 0.95],

'Adam\_beta2' : [0.99, 0.999],

'threshold': [0.5,0.6]},

'GD' : {'lamda': [0.01, 0.1, 0.5],

'eta': [0.1, 0.2, 0.3, 0.4, 0.5],

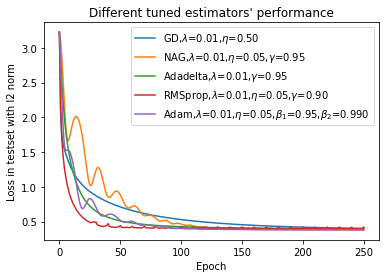
'threshold': [0.4,0.5,0.6]}}

## Predicted Results (Best Results):

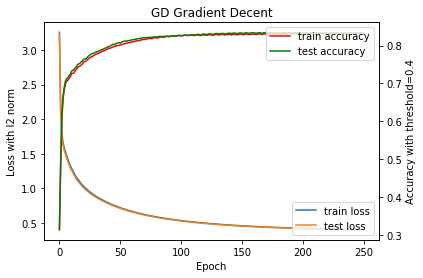
The performance of best estimators with five different optimize algorithm are as follows:

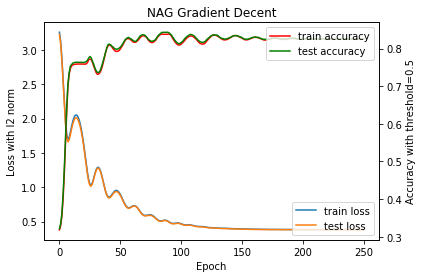
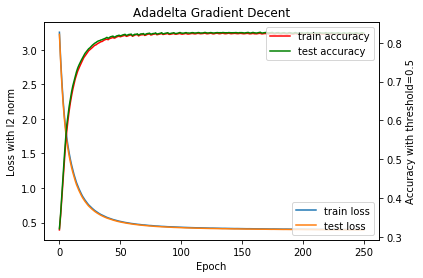
|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Test accuracy** | **Train accuracy** | **Hyper-parameters** |
| GD | **0.8313319615** | **0.8317465648** | {'eta': 0.5, 'lamda': 0.01, 'threshold': 0.4} |
| Adadelta | 0.8255888946 | 0.8259728123 | {'gamma': 0.95, 'lamda': 0.01, 'threshold': 0.5} |
| Adam | 0.8307177298 | 0.8315162541 | {'Adam\_beta1': 0.95, 'Adam\_beta2': 0.99, 'eta': 0.05, 'lamda': 0.01, 'threshold': 0.5} |
| NAG | 0.8268480698 | 0.827800148 | {'eta': 0.05, 'gamma': 0.95, 'lamda': 0.01, 'threshold': 0.5} |
| RMSprop | 0.8290285925 | 0.8297654828 | {'eta': 0.05, 'gamma': 0.9, 'lamda': 0.01, 'threshold': 0.5} |

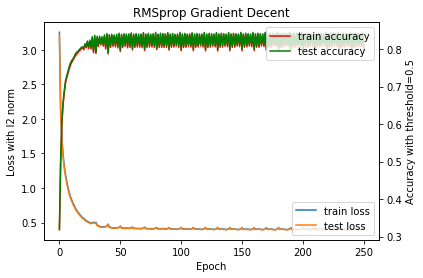
## Loss curve:

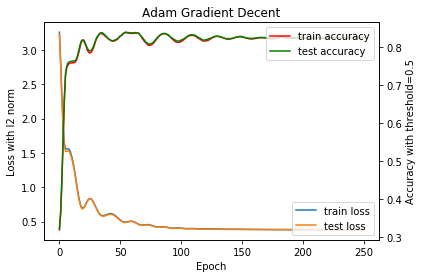


Metrics evolution process details:

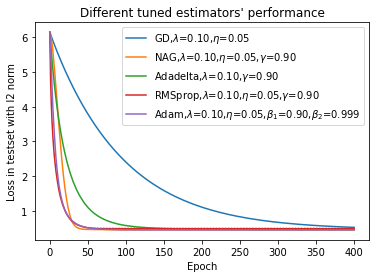






I append a experiment to show the untuned estimators performance



**11. Results analysis:**

Logistic Regression which accuracy is 0.8444 is slightly better than Linear Classification which accuracy is 0.8313.

The performance of different optimization algorithms are ranked as **RMSprop, Adam, Adadelta, GD, NAG** with exhaustive grid search tuning hyper-parameters. And they are ranked as **RMSprop, Adam, NAG, Adadelta, GD**.

With mini-bath gradient decent and no tuning, **RMSprop, Adam, NAG** reach convergence within 50 epochs. Adadelta needs 100+ epochs to converge and GD needs 400+ epochs.

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

They both solve the classification problems. Thus, for binary classification, they both need threshold to give a final prediction. The raw prediction of classifier is a float between 0 and 1. The greater raw prediction the higher probability of positive.

The loss of linear classification is hinge loss, but the logistic regression is negative log likelihood.

**13. Summary:**

In practice, we prefer logistic regression rather than linear classification if we face a classification problem. And if we can’t tuning the hyper-parameters due to the resource limits, we can propose RMSprop or Adam. Both of the algorithm are not so sensitive to hyper-parameters especially learning rate.