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:
Haoyuan Yao

Supervisor:
Mingkui Tan

Student ID：
201530613436

Grade:
Undergraduate

December 13, 2017

Logistic Regression, Linear Classification and Stochastic Gradient Descent

Abstract—

I. INTRODUCTION

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.

II. METHODS AND THEORY

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 G 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 L_{NAG} , $L_{RMSProp}$, $L_{AdaDelta}$ and L_{Adam}.
7. Repeate step 4 to 6 for several times, and **drawing graph of** L_{NAG} **,** $L_{RMSProp}$ **,** $L_{AdaDelta}$ **and** L_{Adam} **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 G 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 L_{NAG} , $L_{RMSProp}$, $L_{AdaDelta}$ and L_{Adam}.
7. Repeate step 4 to 6 for several times, and **drawing graph of L_{NAG} , $L_{RMSProp}$, $L_{AdaDelta}$ and L_{Adam} with the number of iterations**.

III. EXPERIMENT

A.Dataset and Environment

Dataset

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.

Environment for Experiment

python3, at least including following python package: sklearn , numpy , jupyter , matplotlib
It is recommended to install anaconda3 directly, which has built-in python package above.

B. Implementation

```
# RMSProp :
# 每次使用100组数组随机梯度下降
# 利用Gt计算学习率
for l in range(500):
    i = random.randint(0, m - 1)
    for j in range(n):
        if y_train[i, 0] * y_train_predict[i, 0] < 1:
            grad[0, j] += -1 * y_train[i, 0] * x_train[i, j]
# 计算C*sum(-x*y)
grad[0] *= 0.8
# 计算w-C*sum(x*y)
for j in range(n):
    grad[0, j] += abs(param[0, j])
#计算Gt=0.9 * Gt + 0.1 * grad * grad
Gt[0] = 0.9 * Gt[0] + 0.1 * grad[0] * grad[0]
# 更新梯度 param=param- n/sqrt(Gt+e) * grad
param[0] -= (0.001 / (np.sqrt(Gt[0]+0.00000001))) * grad[0]
```

```python
# NAG(Nesterov accelerated gradient):
# 第0次迭代 无先前梯度 直接随机梯度下降
if k == 0:
    for l in range(500):
        i = random.randint(0, m - 1)
        for j in range(n):
            if y_train[i, 0] * y_train_predict[i, 0] < 1:
                grad[0, j] += -1 * y_train[i, 0] * x_train[i, j]
    # 计算C*sum(-x*y)
    grad[0] *= 0.8
    # 计算w-C*sum(x*y)
    for j in range(n):
        grad[0, j] += abs(param[0, j])
    # 更新梯度 param=param-learning_rate * grad
    param[0] -= learning_rate * grad[0]
    # 记录第0次梯度
    pre_grad=grad.copy()

# 除了第0次迭代后 均为先根据之前的梯度更新param后再计算梯度 之后再根据目前梯度*学习率+先前梯度*Gamma更新参数
if k != 0:
    # 先利用先前梯度*Gamma更新出新的参数再计算出新的梯度
    paramNew = param.copy()
    for i in range(n):
        paramNew[0, i] = paramNew[0, i] - (pre_grad[0, i] * Gamma)
    y_train_predict = x_train.dot(paramNew.T)
    # 利用新的参数算出新的梯度
    for l in range(500):
        i = random.randint(0, m - 1)
        for j in range(n):
            if y_train[i, 0] * y_train_predict[i, 0] < 1:
                grad[0, j] += -1 * y_train[i, 0] * x_train[i, j]
    grad[0] *= 0.8
    for j in range(n):
        grad[0, j] += abs(param[0, j])
    # 根据新的梯度*学习率+先前梯度*Gamma更新参数
    param[0] = param[0]-learning_rate * grad[0]-Gamma*pre_grad[0]
    # 记录本次梯度 用于下次迭代更新参数
    pre_grad = grad.copy()
```

```python
# AdaDelta:
# 每次使用100组数组随机梯度下降
# 利用Gt和deltaT计算学习率
for l in range(500):
    i = random.randint(0, m - 1)
    for j in range(n):
        if y_train[i, 0] * y_train_predict[i, 0] < 1:
            grad[0, j] += -1 * y_train[i, 0] * x_train[i, j]
# 计算C*sum(-x*y)
grad[0] *= 0.8
# 计算w-C*sum(x*y)
for j in range(n):
    grad[0, j] += abs(param[0, j])

#计算Gt=0.9 * Gt + 0.1 * grad * grad
Gt[0] = 0.9 * Gt[0] + 0.1 * grad[0] * grad[0]
# 计算 deltaParam=-1*sqrt(deltaT+0.00000001)/sqrt(Gt+0.00000001) * grad
deltaParam[0] = -1 * (np.sqrt(deltaT[0]+0.00000001) / np.sqrt(Gt[0]+0.00000001)) * grad[0]
# 更新梯度 param=param+deltaParam
param[0] += deltaParam[0]
# 计算deltaT
deltaT[0]=0.95 * deltaT[0] + 0.05 * deltaParam * deltaParam
```

```
#Adam:
for l in range(500):
    i = random.randint(0, m - 1)
    for j in range(n):
        if y_train[i, 0] * y_train_predict[i, 0] < 1:
            grad[0, j] += -1 * y_train[i, 0] * x_train[i, j]
# 计算C*sum(-x*y)
grad[0] *= 0.8
# 计算w-C*sum(x*y)
for j in range(n):
    grad[0, j] += abs(param[0, j])

#计算Mt=0.9 * Mt + 0.1 * grad
Mt[0] = 0.9 * Mt[0] + 0.1 * grad[0]
#计算Gt=0.999 * Gt + 0.001 * grad * grad
Gt[0] = 0.999 * Gt[0] + 0.001 * grad[0] * grad[0]
# 计算 alpha= n * sqrt(1-0.999^t) / 1-0.9^t
alpha = 0.001 * math.sqrt(1-pow(0.999,k+1)) / (1-pow(0.9,k+1))
# 更新参数 param=param - alpha * Mt / sqrt(Gt+e)
param[0] -= alpha * Mt[0] / np.sqrt(Gt[0]+0.00000001)
```

initalizing zeros:

```
param = np.ones((1, n))
learning_rate = 0.001
num_iter = 1000
#NAG 参数
Gamma=0.001
pre_grad=np.zeros((1, n))
#RMSProp 参数
Gt=np.zeros((1,n))
#AdaDelta 参数
deltaParam=np.zeros((1,n))
deltaT=np.zeros((1,n))
#Adam 参数
Mt=np.zeros((1,n))
```

Loss function:

逻辑回归：

loss=-1/m*sum[y*log(sigmoid(wx+b)) + (1-y)*log(1-sigmoid(wx+b))]

loss'=sum(sigmoid(xw+b)-y)x

线性分类：

loss=0.5*||w||^2+C*sum(max(0,1-y*(wx+b)))

loss'=||w||+C*sum(max(0,-y*x))

IV. CONCLUSION

A. 预测结果（最佳结果）：

NAG 参数：
[[-3.23733839e-01 2.05191979e+00 2.98983707e+00 3.40568181e+00
3.55446301e+00 2.94999976e+00 1.73348993e+00 1.90208869e+00
1.81163379e+00 1.71126697e+00 1.27837478e+00 1.47853014e-04
2.44353238e-03 2.41187027e+00 2.61634913e+00 2.75934346e+00
2.55513410e+00 2.78684333e+00 1.09896713e+00 2.15636006e+00
4.31481163e-01 1.94828172e+00 1.14041542e+00 1.24181809e+00
1.29054739e+00 2.13850430e-01 2.35536637e-01 3.15395611e-01
1.26721322e+00 7.94314268e-02 4.27080403e-01 1.14465621e+00
1.63197646e-01 2.20189504e-03 1.76996674e+00 1.94828172e+00
2.15636006e+00 2.52991392e+00 4.64389732e+00 6.27037367e+00
2.53080964e+00 2.28257629e+00 7.01844217e-01 9.09354038e-01
3.27524611e-01 1.10773213e-01 7.92246059e-01 2.66441083e-01
-1.18747895e+00 6.49474020e-01 1.49005916e+00 1.58981944e+00
-3.54303726e+00 -7.88763916e-01 8.83506033e-01 -3.95894240e+00
-1.29058721e-01 -4.98647456e-01 6.73488846e-01 1.16443735e-02
3.23570299e+00 6.70630369e-01 2.88466541e+00 3.82414248e+00
3.35196615e-01 2.18097816e+00 6.21006059e+00 2.93757444e+00
4.34971242e-01 2.81956283e-01 3.26473119e+00 6.11228378e+00
6.99848385e+00 6.53107377e+00 6.54534627e+00 7.48453474e+00
5.61463787e+00 1.58178320e+00 1.97344689e+00 3.26520962e+00
2.74521977e+00 3.51687974e+00 5.93300389e+00 9.57315600e-02
3.32985652e-01 1.22871246e-01 4.25961058e-01 4.48339023e-01
1.56867648e-03 4.16920041e-01 2.69088284e-01 5.71098563e-02
1.66913840e-01 2.09768565e-01 2.69055247e-01 1.82131396e-01
1.20325893e-02 6.94063844e-01 3.05506126e-01 1.37647247e-01
1.24986162e-01 -1.32807552e-01 3.86016131e-01 5.42341438e-02
5.53368830e-02 1.29665728e-01 -6.62242949e-02 2.21889422e-02
4.38602896e-02 2.28194441e-01 5.70957320e-02 -1.44517321e+00
4.01754840e-02 3.61372406e-02 3.52906674e-02 3.02355850e-02
3.31888505e-02 6.11124047e-02 1.07349186e-01 2.79258251e-02
2.07127609e-02 7.42086283e-02 2.43057132e-03 -5.84543327e+01]]

RMS 参数：
[[-1.18082859e+00 -4.02587707e-01 1.78677478e-02 1.85445243e-01
1.58675229e-01 1.41577840e-01 -1.95387966e-01 2.91109462e-01
3.07065782e-01 -3.23213569e-02 -1.22519962e-01 -1.48126325e+00
-1.74194085e+00 -1.60851044e-01 -2.27094278e-02 -1.03047612e-02
2.41929969e-03 1.11639166e-02 2.32603764e-02 -1.71726249e-01
-5.92572083e-01 -3.60884274e-01 4.04553567e-01 -5.38310124e-02
-1.00143549e-01 -1.24887464e+00 -1.31099682e+00 -7.18385793e-01
1.51220028e-01 -1.56121245e+00 -7.93886585e-01 4.25910479e-01
-1.39950792e+00 -1.55757262e+00 -3.83040591e-01 -3.60884274e-01
-1.71726249e-01 -1.04941472e-01 3.18364270e-01 2.97451474e-01
-5.34751688e-01 -6.78720807e-01 -6.23667327e-01 -3.40211970e-01
-1.00850450e+00 7.71793470e-02 3.27966195e-01 -3.60762846e-01
-8.11576322e-01 8.99547388e-02 4.18150232e-01 2.61227239e-01
-8.16474229e-01 -5.52765291e-01 -8.16698713e-02 -9.31277686e-01
-4.55213072e-01 -1.60197016e+00 1.50889937e-01 -1.46831006e+00
6.16430640e-01 -1.07410797e+00 6.04005924e-02 -4.51442324e-01
-1.19908778e+00 -7.02367259e-01 2.54922274e-02 1.42229296e-02
-9.44071042e-01 -1.03001179e+00 7.66390566e-03 -2.01579701e-01
9.81922418e-02 -4.24207877e-01 8.00468739e-01 -1.63808175e-01
4.89093003e-01 -9.15868522e-01 -1.29956659e-01 -5.74512920e-02

```
 1.50382905e-01  2.38867727e-01  9.81540426e-02  2.84746866e-02
 5.40970773e-02 -8.75863228e-01  1.08391465e-01  6.47238091e-02
-1.57848333e+00 -6.86618575e-02  3.71988019e-02 -9.75768036e-01
-6.32629978e-01 -3.33233971e-01  3.35982141e-02  2.45510186e-02
-1.62966886e+00  1.53915980e-01  9.94447498e-02 -1.40768095e-01
-3.49285808e-01 -1.24256349e+00 -8.92030807e-01 -1.00381353e+00
 1.78107494e-02  4.43843163e-02 -1.38650386e+00 -1.29753712e+00
-9.18791689e-01 -2.52665969e-02 -8.67795301e-01 -1.42549135e+00
-1.01834084e+00 -8.48648765e-01 -1.36897967e+00 -3.17155928e-02
-1.08634964e+00 -7.25888950e-04 -7.77966302e-01 -1.25827937e+00
-1.10535282e+00 -5.17744933e-02 -1.83380698e+00  4.92529593e-02]]
```

paramAdaDelta 参数： [[-4.30608065e-01 -2.93931942e-01 -1.34998556e-02 1.03372311e-01

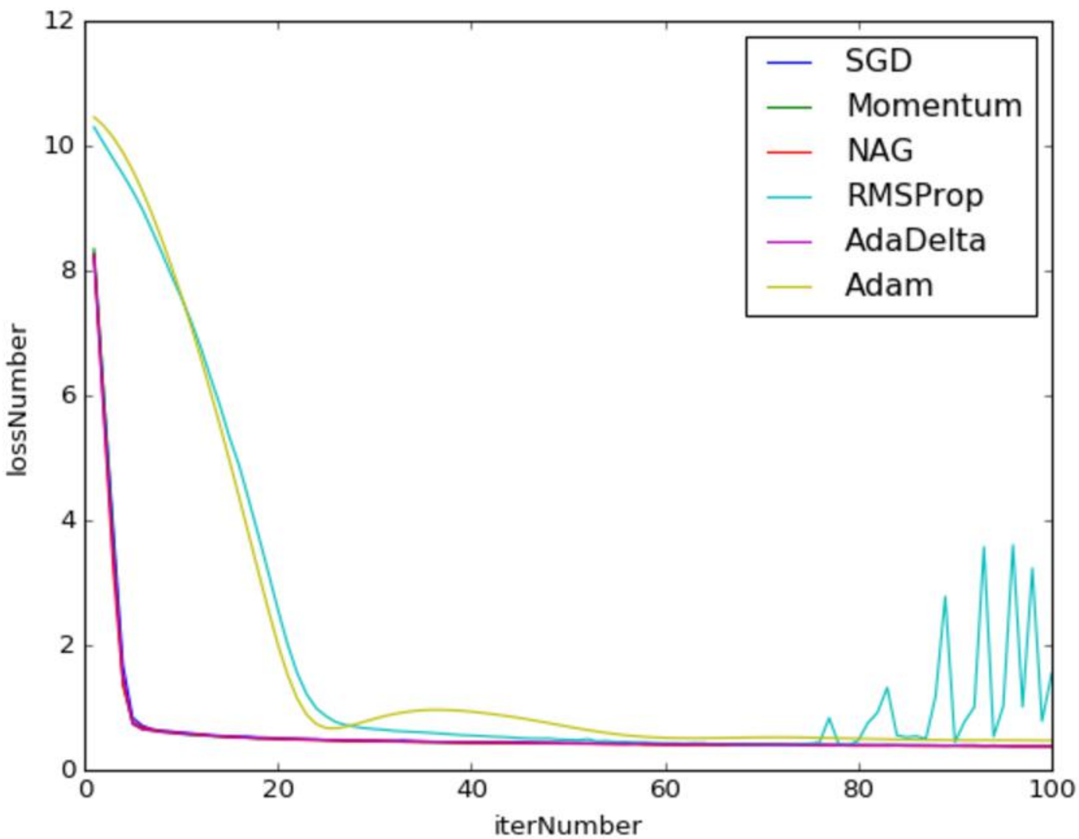
```
 3.89082252e-02  9.09703059e-03 -2.23063710e-01  3.90702215e-01
 2.40610129e-01 -4.40720569e-02 -9.16939887e-03  1.63648131e-01
-2.89903948e-90 -1.65650682e-01 -8.10510761e-02 -7.94441447e-02
-7.81952765e-02 -6.06145052e-02  1.99075571e-01 -9.80232076e-02
-6.84997744e-02 -2.19050143e-01  6.34389515e-01  7.99861093e-02
 1.84480523e-02 -6.34944728e-02 -1.14598977e-01  7.33575923e-02
 3.94769298e-01  9.24794708e-02 -8.22690255e-02  6.40546529e-01
 1.52624107e-02  1.97321978e-01 -5.55550633e-01 -2.19050143e-01
-9.80232076e-02 -7.35742469e-02  3.43531672e-01  2.38645889e-01
-2.70388492e-01 -4.39349182e-01 -1.26221381e-01 -4.22902469e-02
-3.27434213e-03  2.24364475e-01  2.05013437e-01 -1.27746442e-01
-3.08401972e-01  1.06104308e-01  4.09453223e-01  2.35835870e-01
-1.80155912e-01 -1.80525975e-01 -4.21157266e-02 -2.51297745e-01
-1.40744331e-01  1.50314516e-01  1.61782039e-01  1.05365034e-01
 4.06720207e-01 -3.77446315e-01  1.96065516e-01 -3.78360164e-01
-1.59667078e-01 -3.49339759e-01 -3.01455231e-02 -1.07871897e-01
 7.01591407e-02  5.92057957e-02 -1.02538541e-01 -3.05410634e-01
-2.32822888e-02 -3.12680392e-01  6.13124506e-01 -2.16232728e-01
 4.06351218e-01 -4.23890494e-01 -1.08351436e-01 -1.24447179e-01
 9.79888699e-02  1.61584022e-01  6.20170309e-02  2.06270461e-01
 2.87485290e-01  1.73041830e-01  2.69924726e-01  2.62726198e-01
 1.58430014e-01  2.51101786e-01  2.85084806e-01  2.06730102e-01
 2.26153951e-01  2.51238961e-01  2.42226015e-01  2.59635362e-01
 4.63461572e-02  2.38702065e-01  2.82636922e-01  2.34319035e-01
 2.33284635e-01  1.95953929e-01 -1.20224466e-01  2.18446948e-01
 1.98563545e-01  2.40199624e-01  1.89421986e-01  1.68136445e-01
 2.22196326e-01  2.34130989e-01  2.18113870e-01  1.76208298e-01
 1.76852519e-01  2.16297871e-01  2.07595290e-01  1.52821674e-01
 1.88720741e-01  1.75901573e-01  2.15015370e-01  1.78215529e-01
 2.02948686e-01  1.91592540e-01 -3.42884860e-01 -1.56736144e-01]]
```

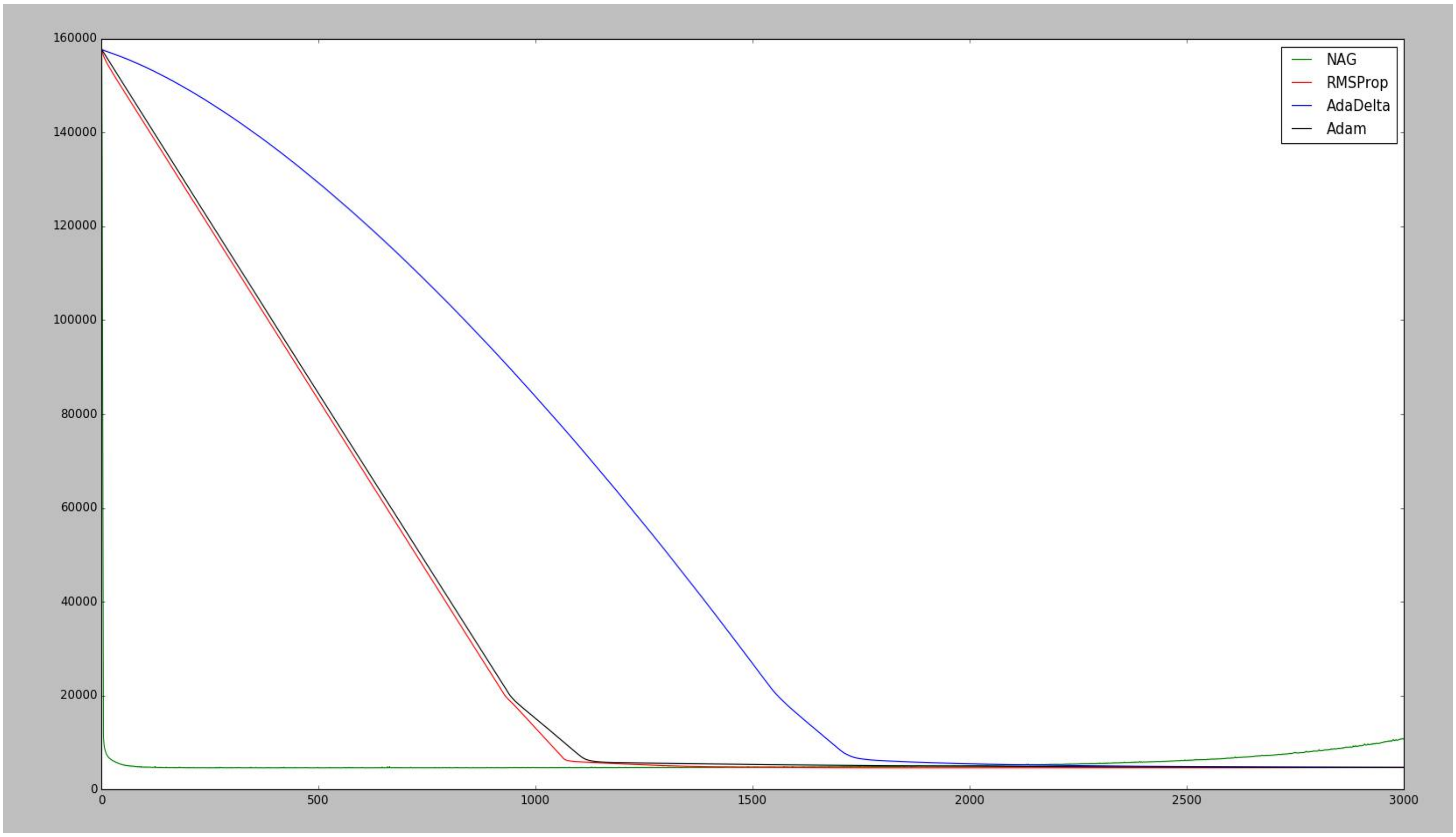
paramAdam 参数： [[-2.94449194e-01 -2.90434195e-01 -1.00749874e-02 1.12254103e-01

```
 4.63749179e-02 -5.58931474e-02 -2.82947077e-01  2.97321023e-01
 2.33989165e-01 -1.54796578e-01 -1.76481606e-01 -8.29275736e-01
 3.70107560e-04 -1.80435826e-01 -1.04488560e-01 -1.06519352e-01
-9.25434889e-02 -7.46885867e-02  1.10860461e-01 -1.28878576e-01
-2.34373493e-01 -2.13540612e-01  6.93689685e-01 -3.57002454e-02
-5.66711724e-02 -3.68283426e-01 -4.22095581e-01 -1.97235139e-01
 3.49377142e-01 -4.53936371e-01 -2.87560724e-01  6.97484242e-01
-3.95024372e-01 -5.51298741e-01 -3.33518536e-01 -2.13540612e-01
-1.28878576e-01 -6.71924386e-02  3.26687656e-01  2.79439942e-01
-2.68035971e-01 -3.23944598e-01 -2.45206262e-01 -2.13804491e-01
-3.19470025e-01  8.34509518e-02  1.73525385e-01 -1.73314948e-01
-2.59640009e-01  4.13969914e-02  4.71160597e-01  1.96091246e-01
-2.78789849e-01 -2.46139489e-01 -8.93181859e-02 -4.39380723e-01
-2.41133330e-01 -2.31209469e-01  6.86756117e-02 -1.50298286e-01
 5.15602724e-01 -2.67865291e-01  2.46560405e-01 -3.24825992e-01
-2.60315039e-01 -3.04299518e-01 -9.82049494e-02 -2.03643497e-01
```

```
-3.69499048e-01  -3.57164858e-01  -1.51988971e-01  -2.51640380e-01
-2.46407192e-02  -1.76513437e-01   8.71460748e-01  -1.44294927e-01
 5.53964095e-01  -3.71909299e-01  -1.49025480e-01  -1.10724629e-01
 9.36606173e-02   1.62398647e-01  -7.58095405e-02   4.26330487e-02
 3.63210123e-02  -2.35536979e-01   4.01613937e-02   1.38512117e-02
-6.70445342e-01  -5.47281170e-02   7.35381729e-02  -3.50039097e-01
-2.55399767e-01  -1.41896523e-01  -1.13347026e-01   1.87156148e-03
 7.38805145e-04   1.07808727e-01   4.57778467e-02  -2.14777302e-01
-4.79153537e-02  -5.59045800e-01  -2.99323810e-01  -1.70731754e-01
-7.60979573e-02   4.76562151e-02  -4.78067539e-01  -3.45214730e-02
-7.01385851e-02  -1.63036391e-02  -1.63754643e-01  -5.36532641e-01
-2.86501557e-01  -1.11204526e-01  -5.02326825e-01  -2.07959125e-01
-2.04943165e-02   1.31299690e-03  -1.92063552e-01  -1.95989988e-01
-3.31666264e-01   5.72894312e-03   2.70800296e-04  -1.16818073e-01]]
```

B. Loss：




NAG
RMSProp
AdaDelta
Adam
160000
140000
120000
100000
80000
60000
40000
20000
0
0
500
1000
1500
2000
2500
3000