《机器学习》课程实验报告

学　　院　　软件学院

专　　业　　软件工程

组　　员　　潘文杰

学　　号　　201530612590

邮　　箱　　3213359017@qq.com

指导教师　　吴庆耀

提交日期　　2017 年 12 月 1 日

1. 实验题目: 逻辑回归、线性分类与随机梯度下降

2. 实验时间：2017 年 12 月 1 日

3. 报告人:潘文杰

4. 实验目的:

1. 对比理解梯度下降和随机梯度下降的区别与联系。
2. 对比理解逻辑回归和线性分类的区别与联系。
3. 进一步理解 SVM 的原理并在较大数据上实践。

5. 数据集以及数据分析：

实验使用的是LIBSVM Data的中的a9a数据，包含32561 / 16281(testing)个样本，每个样本有 123/123 (testing)个属性。

6. 实验步骤:

逻辑回归与随机梯度下降

1)读取实验训练集和验证集。
2)逻辑回归模型参数初始化，可以考虑全零初始化，随机初始化或者正态分布初始化。
3)选择 Loss 函数及对其求导，过程详见课件 ppt。
4)求得部分样本对 Loss 函数的梯度 G。
5)使用不同的优化方法更新模型参数（NAG，RMSProp，AdaDelta 和 Adam）。
6)选择合适的阈值，将验证集中计算结果大于阈值的标记为正类，反之为负类。
7)在验证集上测试并得到不同优化方法的 Loss 函数值
L_{NAG}，$L_{RMSProp}$，$L_{AdaDelta}$和L_{Adam}。
8)重复步骤 4-6 若干次，画出L_{NAG}，$L_{RMSProp}$，$L_{AdaDelta}$和L_{Adam}随迭代次数的变化图。

线性分类与随机梯度下降

1)读取实验训练集和验证集。
2)支持向量机模型参数初始化，可以考虑全零初始化，随机初始化或者正态分布初始化。
3)选择 Loss 函数及对其求导，过程详见课件 ppt。
4)求得部分样本对 Loss 函数的梯度。
5)使用不同的优化方法更新模型参数（NAG，RMSProp，AdaDelta 和 Adam）。
6)选择合适的阈值，将验证集中计算结果大于阈值的标记为正类，反之为负类。
7)在验证集上测试并得到不同优化方法的 Loss 函数值

L_{NAG}，$L_{RMSProp}$，$L_{AdaDelta}$和L_{Adam}。

。

8)重复步骤 4-6 若干次，画出L_{NAG}，$L_{RMSProp}$，$L_{AdaDelta}$和L_{Adam}随迭代次数的变化图。

7. 代码内容:

逻辑回归和随机梯度下降

```
# 实验: 逻辑回归和随机梯度下降
## 1. 读取实验数据，划分训练集合验证集
import sklearn
from sklearn.externals.joblib import Memory
from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import train_test_split
from numpy import *

mem = Memory("./mycache")

@mem.cache
def getData():
    train_X, train_y = load_svmlight_file('data/a9a', n_features=123)
    test_X, test_y = load_svmlight_file('data/a9a.t', n_features=123)

    train_y =  train_y.reshape(train_y.shape[0],1)
    test_y =  test_y.reshape(test_y.shape[0],1)

    train_y[train_y == -1] = 0
    test_y[test_y == -1] = 0
    return train_X, test_X, train_y, test_y

train_X, test_X, train_y, test_y = getData()

## 2.初始化模型参数
import numpy as np
m, n = np.shape(train_X)
theta = np.ones((n, 1))
alpha = 0.03
maxIteration = 500

## 3.选择 Loss 函数
```

```
def getLoss(x,y,theta):
    return  -( y * log(sigmoid(x * theta)) + (1 - y) * log(1 - sigmoi
d(x * theta)) ).sum() / x.shape[0]

def sigmoid(a):
    return 1 / (1 + np.exp(-a))

## 4.随机梯度下降函数训练
def getGradientSGD(w):
    random_num = np.random.randint(0,m)
    return (train_X[random_num].T * (sigmoid(train_X[random_num] *
w) - train_y[random_num]))

train_loss = []
evaluation_loss = []

def SGD(theta):
    for i in range(0, maxIteration):
        gradient = getGradientSGD(theta)
        theta = theta - alpha * gradient

        train_loss.append(getLoss(train_X,train_y,theta))
        evaluation_loss.append(getLoss(test_X,test_y,theta))

SGD(theta)

## 5.绘制 Loss train 和 Loss validation 随迭代次数的变化图
import matplotlib.pyplot as plt
%matplotlib inline
plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.plot(train_loss, label="train")
plt.plot(evaluation_loss,label="evaluation" )
plt.legend(loc ='upper right')

## 6.使用不同的优化方法更新模型参数
### NAG
train_loss_nag,evaluation_loss_nag,train_accr_nag,evaluation_accr
_nag = [],[],[],[]
theta = np.ones((n, 1))

def NAG(theta):
    gama = 0.9
    vt = 0
```

```
    for i in range(0, maxIteration):
        gradient = getGradientSGD(theta - gama*vt)
        vt = gama*vt + alpha * gradient
        theta = theta - vt

        train_loss_nag.append(getLoss(train_X,train_y,theta))
        evaluation_loss_nag.append(getLoss(test_X,test_y,theta))

NAG(theta)

plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.plot(train_loss_nag, label="train")
plt.plot(evaluation_loss_nag,label="evaluation" )
plt.legend(loc ='upper right')

### RMSProp

train_loss_RMSProp,evaluation_loss_RMSProp,train_accr_RMSProp,eva
luation_accr_RMSProp = [],[],[],[]
theta = np.ones((n, 1))

def RMSProp(theta):
    gama = 0.9
    vt = 0
    Egt = 0
    e=0.00000001

    learning_rate = 0.3

    for i in range(0, maxIteration):
        gradient = getGradientSGD(theta - gama*vt)
        Egt = gama * Egt + ((1-gama)*(gradient**2)).sum()
        theta = theta - learning_rate*gradient/math.sqrt(Egt + e)

        train_loss_RMSProp.append(getLoss(train_X,train_y,theta))
        evaluation_loss_RMSProp.append(getLoss(test_X,test_y,thet
a))

RMSProp(theta)

plt.xlabel("Iterations")
plt.ylabel("Loss")
```

```
plt.plot(train_loss_RMSProp, label="train")
plt.plot(evaluation_loss_RMSProp,label="evaluation" )
plt.legend(loc ='upper right')

### AdaDelta
train_loss_adaDelta,evaluation_loss_adaDelta,train_accr_adaDelta,
evaluation_accr_adaDelta = [],[],[],[]
theta = np.ones((n, 1))

def adaDelta(theta):

    rho = 0.9
    Egt=0
    Edt = 0
    e=0.00000001
    delta = 0
    learning_rate = 2000

    for i in range(0, maxIteration):

        gradient = getGradientSGD(theta)
        Egt = rho * Egt + ((1-rho)*(gradient**2) ).sum()
        delta = - math.sqrt(Edt + e)*gradient/math.sqrt(Egt + e)
        Edt =rho*Edt+( (1-rho)*(delta**2) ).sum()
        theta = theta + learning_rate*delta

        train_loss_adaDelta.append(getLoss(train_X,train_y,theta))
        evaluation_loss_adaDelta.append(getLoss(test_X,test_y,thet
a))

adaDelta(theta)

plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.plot(train_loss_adaDelta, label="train")
plt.plot(evaluation_loss_adaDelta,label="evaluation" )
plt.legend(loc ='upper right')

### Adam
train_loss_adam,evaluation_loss_adam,train_accr_adam,evaluation_a
ccr_adam = [],[],[],[]
theta = np.ones((n, 1))

def adam(theta):
```

```
    t = 0
    m = 0
    v = 0
    b1 = 0.9
    b2 = 0.995
    learning_rate = 0.05

    for i in range(0, maxIteration):

        gradient = getGradientSGD(theta)
        t +=1
        m = b1 * m + ((1 - b1) * gradient).sum()
        v = b2 * v + ((1 - b2) * (gradient ** 2)).sum()
        mt = m / (1 - (b1 ** t))
        vt = v / (1 - (b2 ** t))
        theta = theta- learning_rate * mt / (math.sqrt(vt) + e)

        train_loss_adam.append(getLoss(train_X,train_y,theta))
        evaluation_loss_adam.append(getLoss(test_X,test_y,theta))

adam(theta)

plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.plot(train_loss_adam, label="train")
plt.plot(evaluation_loss_adam,label="evaluation" )
plt.legend(loc ='upper right')

## 五种不同随机梯度下降方法对比

plt.plot(train_loss, label="train_loss")
plt.plot(train_loss_nag, label="train_loss_nag")
plt.plot(train_loss_adaDelta, label="train_loss_adaDelta")
plt.plot(train_loss_RMSProp, label ="train_loss_RMSProp")
plt.plot(train_loss_adam, label="train_loss_adam")
plt.legend(loc="upper right")
```

线性分类和随机梯度下降

```
# 实验：线性分类和随机梯度下降
## 1. 读取实验数据，划分训练集合验证集
import sklearn
```

```
from sklearn.externals.joblib import Memory
from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import train_test_split
from numpy import *

mem = Memory("./mycache")

@mem.cache
def getData():
    train_X, train_y = load_svmlight_file('data/a9a', n_features=12
3)
    test_X, test_y = load_svmlight_file('data/a9a.t', n_features=12
3)

    train_y =  train_y.reshape(train_y.shape[0],1)
    test_y =  test_y.reshape(test_y.shape[0],1)

    train_y[train_y == -1] = 0
    test_y[test_y == -1] = 0
    return train_X, test_X, train_y, test_y

train_X, test_X, train_y, test_y = getData()

## 2.初始化 SVM 模型参数
import numpy as np

m, n = np.shape(train_X)

theta = np.ones((n, 1))
maxIteration = 300
c = 0.5
learning_rate = 0.01

## 3.计算随机梯度下降函数和计算 Loss 函数
def getStochasticGradient(theta):
    index =  (1 - train_y * (train_X * theta) < 0)
    y = train_y.copy()
    y[index] = 0
    randomNum = np.random.randint(0,train_X.shape[0])
    epsilon_gradient = - ((train_X)[randomNum].T * y[randomNum]).re
shape(123,1)
    gradient = theta +  epsilon_gradient
    return gradient
```

```
def getHingeLoss(theta,x,y):
    epsilon_loss = 1 - y * x.dot(theta)
    epsilon_loss[epsilon_loss<0] = 0
    loss = 0.5 * np.dot(theta.transpose(), theta).sum() + epsilon_l
oss.sum()
    return loss/x.shape[0]

## 4.随机梯度下降训练
train_loss, evaluation_loss, train_accr, evaluation_accr = [],[],
[],[]

def gradientDescent(w):
    for i in range(maxIteration):
        gradient = getStochasticGradient(w)
        w -= learning_rate*gradient

        train_loss.append(getHingeLoss(w,train_X,train_y))
        evaluation_loss.append( getHingeLoss(w,test_X,test_y))

gradientDescent(theta)

## 5.绘制 Loss train 和 Loss validation 随迭代次数的变化图

import matplotlib.pyplot as plt
%matplotlib inline
plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.plot( train_loss, label="train")
plt.plot( evaluation_loss,label="evaluation" )
plt.legend(loc="upper right")

## 6.使用不同的优化方法更新模型参数
### NAG
train_loss_nag,evaluation_loss_nag,train_accr_nag,evaluation_accr
_nag = [],[],[],[]
theta = np.ones((n, 1))

def NAG(w):
    vt = 0
    gama = 0.9

    for i in range(maxIteration):
        gradient = getStochasticGradient(w -gama*vt)
        vt = gama*vt + learning_rate * gradient
```

```
        w = w - vt

        train_loss_nag.append(getHingeLoss(w,train_X,train_y))
        evaluation_loss_nag.append( getHingeLoss(w,test_X,test_y))


NAG(theta)

plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.plot(train_loss_nag, label="train")
plt.plot(evaluation_loss_nag,label="evaluation" )
plt.legend(loc ='upper right')

### RMSProp
train_loss_RMSProp,evaluation_loss_RMSProp,train_accr_RMSProp,eva
luation_accr_RMSProp = [],[],[],[]
theta = np.ones((n, 1))

def RMSProp(w):
    gama = 0.9
    vt = 0
    Egt = 0
    e=0.00000001

    learning_rate = 0.3

    for i in range(0, maxIteration):
        gradient = getStochasticGradient(w - gama*vt)
        Egt = gama * Egt + ((1-gama)*(gradient**2)).sum()
        w -= learning_rate*gradient/math.sqrt(Egt + e)

        train_loss_RMSProp.append(getHingeLoss(w,train_X,train_y))
        evaluation_loss_RMSProp.append( getHingeLoss(w,test_X,test_
y))

RMSProp(theta)

plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.plot(train_loss_RMSProp, label="train")
plt.plot(evaluation_loss_RMSProp,label="evaluation" )
plt.legend(loc ='upper right')
```

```
### AdaDelta
train_loss_adaDelta,evaluation_loss_adaDelta,train_accr_adaDelta,
evaluation_accr_adaDelta = [],[],[],[]
theta = np.ones((n, 1))

def adaDelta(w):

    rho = 0.9
    Egt=0
    Edt = 0
    e=0.00000001
    delta = 0
    learning_rate = 2000

    for i in range(0, maxIteration):

        gradient = getStochasticGradient(w)
        Egt = rho * Egt + ((1-rho)*(gradient**2) ).sum()
        delta = - math.sqrt(Edt + e)*gradient/math.sqrt(Egt + e)
        Edt =rho*Edt+( (1-rho)*(delta**2) ).sum()
        w = w + learning_rate*delta

        train_loss_adaDelta.append(getHingeLoss(w,train_X,train_y))
        evaluation_loss_adaDelta.append( getHingeLoss(w,test_X,test
_y))

adaDelta(theta)

plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.plot(train_loss_adaDelta, label="train")
plt.plot(evaluation_loss_adaDelta,label="evaluation" )
plt.legend(loc ='upper right')

### Adam
train_loss_adam,evaluation_loss_adam,train_accr_adam,evaluation_a
ccr_adam = [],[],[],[]
theta = np.ones((n, 1))

def adam(w):

    t = 0
    m = 0
    v = 0
```

```
    b1 = 0.9
    b2 = 0.995
    learning_rate = 0.05

    for i in range(0, maxIteration):

        gradient = getStochasticGradient(w)
        t +=1
        m = b1*m + ((1-b1)*gradient).sum()
        v = b2*v + ((1-b2)*(gradient**2)).sum()
        mt = m/(1-(b1**t))
        vt = v/(1-(b2**t))
        w = w- learning_rate * mt/(math.sqrt(vt)+e)

        train_loss_adam.append(getHingeLoss(w,train_X,train_y))
        evaluation_loss_adam.append( getHingeLoss(w,test_X,test_y))

adam(theta)

plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.plot(train_loss_adam, label="train")
plt.plot(evaluation_loss_adam,label="evaluation" )
plt.legend(loc ='upper right')

## 五种不同随机梯度下降方法对比
plt.plot(train_loss, label="train_loss")
plt.plot(train_loss_nag, label="train_loss_nag")
plt.plot(train_loss_adaDelta, label="train_loss_adaDelta")
plt.plot(train_loss_RMSProp, label ="train_loss_RMSProp")
plt.plot(train_loss_adam, label="train_loss_adam")
plt.legend(loc="upper right")
```

8. 模型参数的初始化方法:

逻辑回归

```
m, n = np.shape(train_X)
theta = np.ones((n, 1))
alpha = 0.03
maxIteration = 500
```

线性分类

```
m, n = np.shape(train_X)
theta = np.ones((n, 1))
```

```
maxIteration = 300
c = 0.5
learning_rate = 0.01
```

9.选择的 loss 函数及其导数:

逻辑回归

定义逻辑函数

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

Label 为{0,1}

Loss 函数为

$$J(w) = -\frac{1}{m}[\sum_{i=1}^{m} y_i log h_w(x_i) + (1 - y_i) log(1 - h_w(x_i))]$$

其导数为

$$\frac{\partial J(w)}{\partial w} = -y \frac{1}{h_w(x)} \frac{\partial h_w(x)}{\partial w} + (1 - y) \frac{1}{1 - h_w(x)} \frac{\partial h_w(x)}{\partial w}$$

线性分类

定义

$$\ell(y) = \max(0, 1 - t \cdot y)$$

Hinge loss 函数

$$\frac{\partial \ell}{\partial w_i} = \begin{cases} -t \cdot x_i & \text{if } t \cdot y < 1 \\ 0 & \text{otherwise} \end{cases}$$

则 loss 函数为

$$J(w) = \frac{1}{2}\|w\|^2 + C \sum \max(0, 1 - y_i(w_i^x + b))$$

10.实验结果和曲线图:

逻辑回归

1) Vanilla SGD

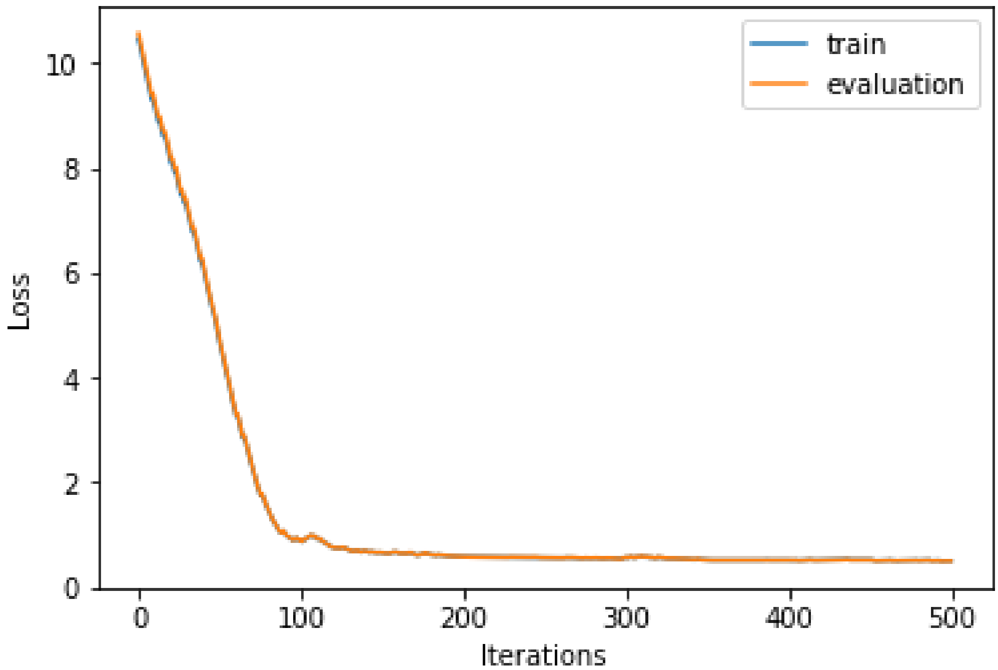
超参数选择:

```
alpha = 0.03
maxIteration = 500
```

预测结果（最佳结果）:

0.495440805639

loss 曲线图:



2) NAG

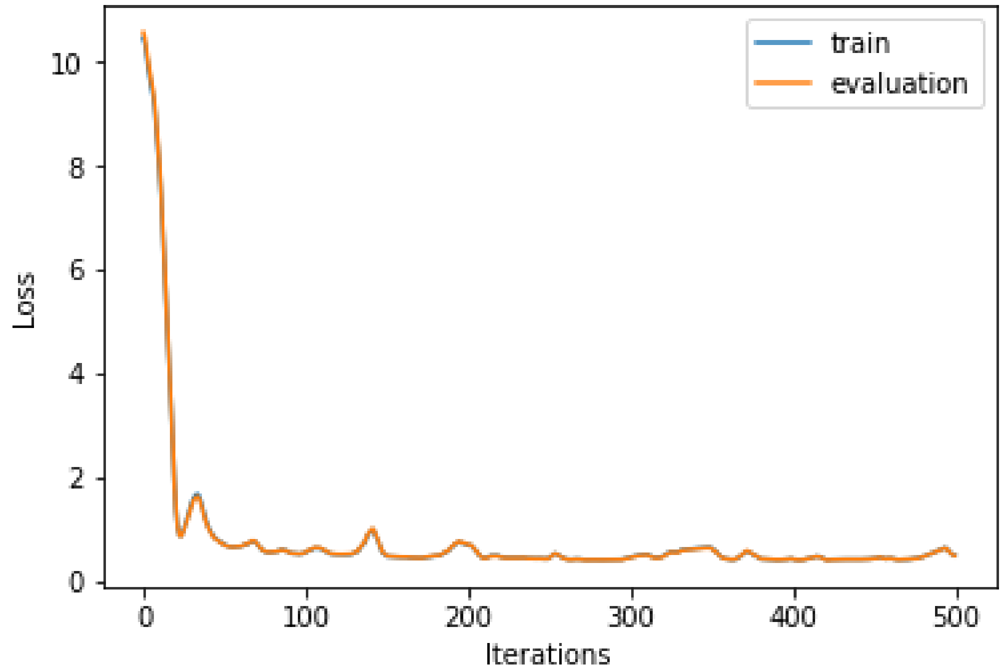
超参数选择:

```
alpha = 0.03
maxIteration = 500
```

预测结果（最佳结果）:

0.424465702885

loss 曲线图：



3) RMSProp

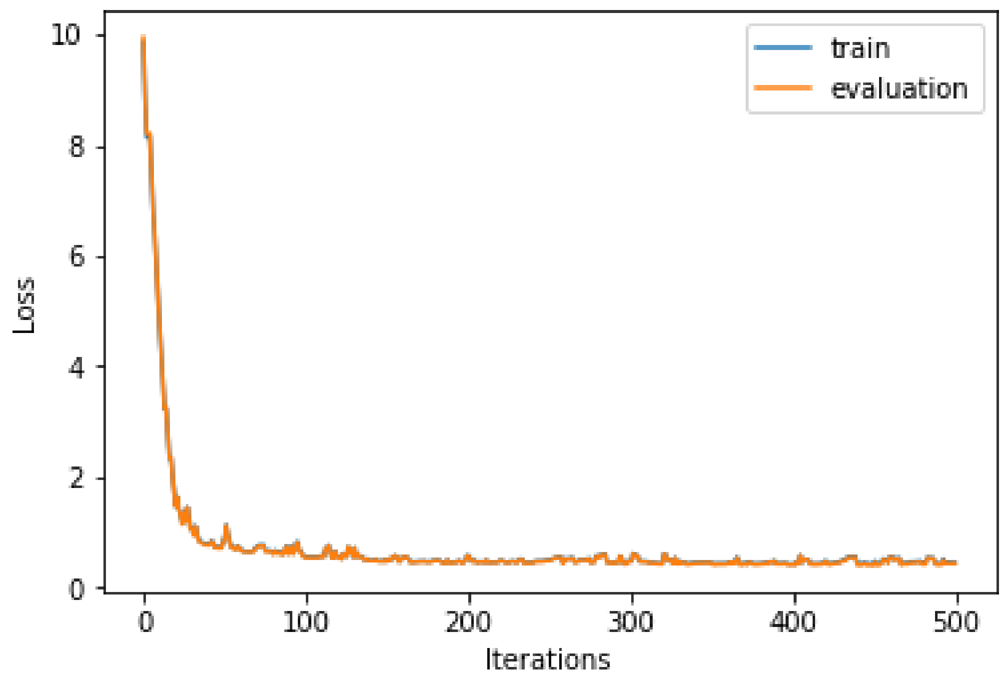
超参数选择：

```
learning_rate = 0.3
maxIteration = 500
```

预测结果（最佳结果）：

0.392042296275

loss 曲线图：



4) AdaDelta

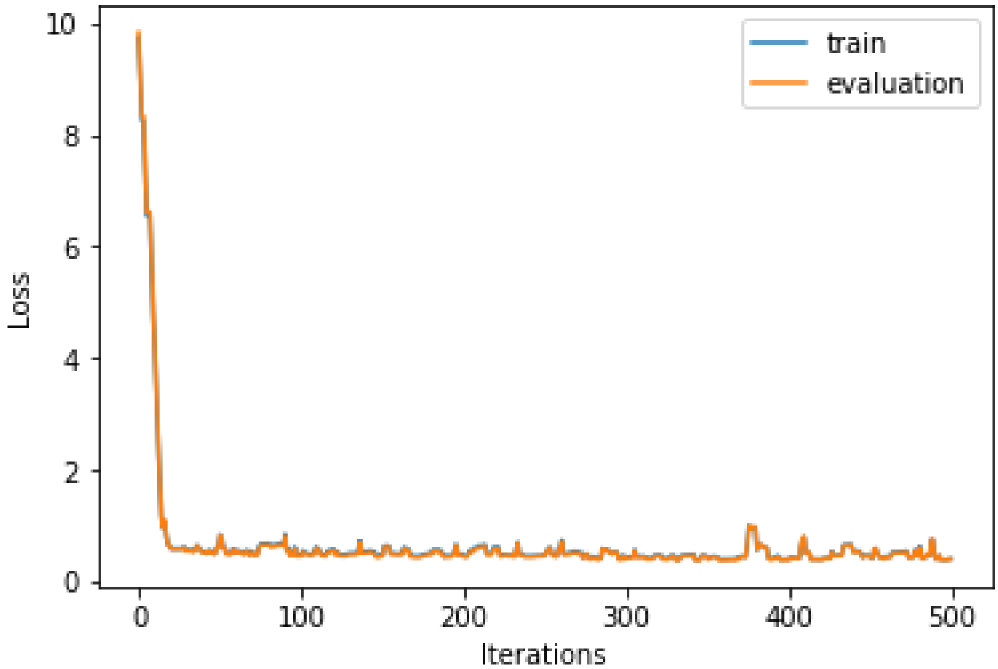
超参数选择：

```
learning_rate = 2000
maxIteration = 500
```

预测结果（最佳结果）：

0.402268825518

loss 曲线图：



5) Adam

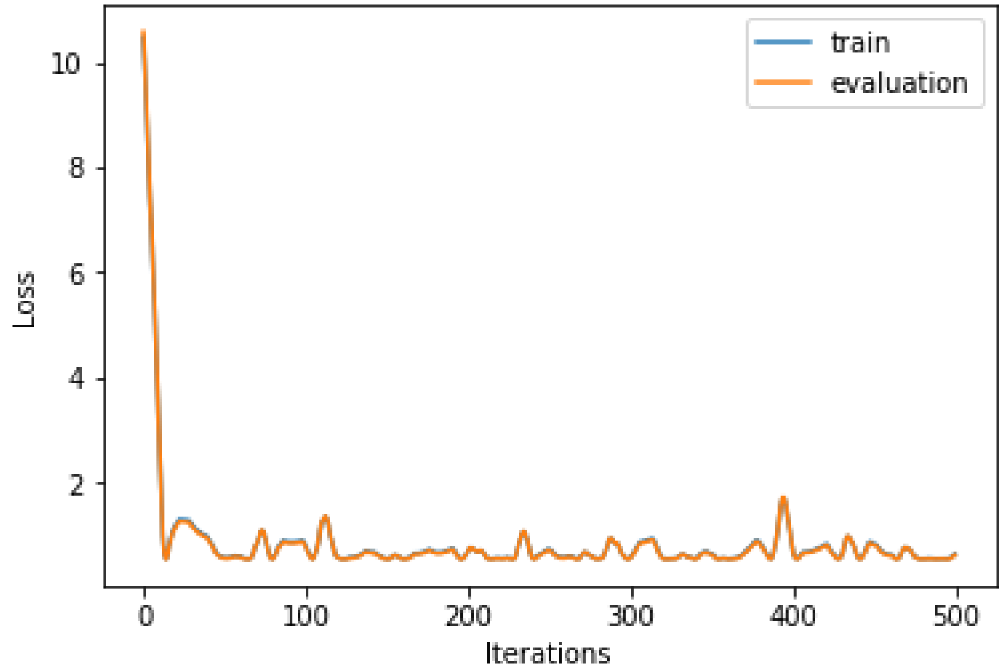
超参数选择：

```
learning_rate = 0.05
maxIteration = 500
```

预测结果（最佳结果）：

0.548395087657

loss 曲线图：



线性分类

1) Vanilla SGD

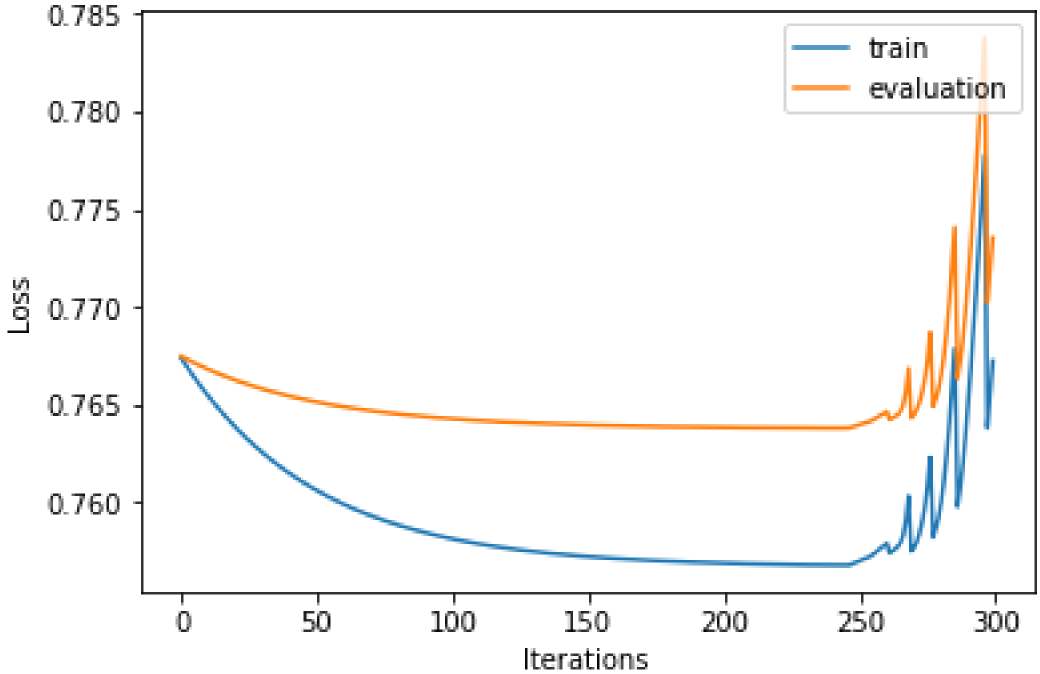
超参数选择：

```
learning_rate = 0.01
maxIteration = 300
```

预测结果（最佳结果）：

0.763805192026

loss 曲线图：



2) NAG

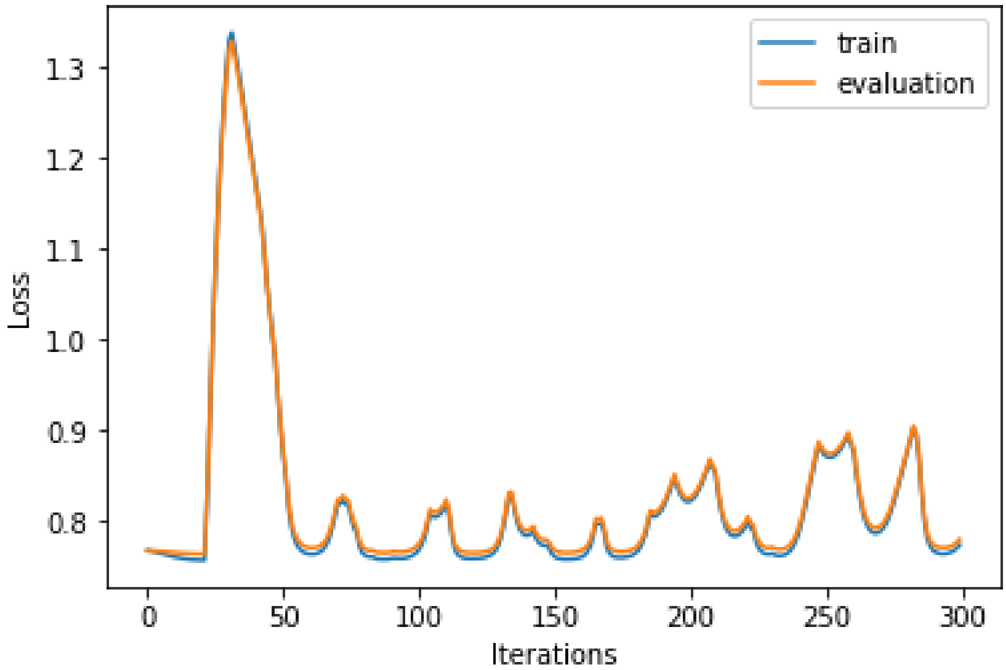
超参数选择：

```
learning_rate = 0.01
maxIteration = 300
```

预测结果（最佳结果）：

0.763822922757

loss 曲线图：



3) RMSProp

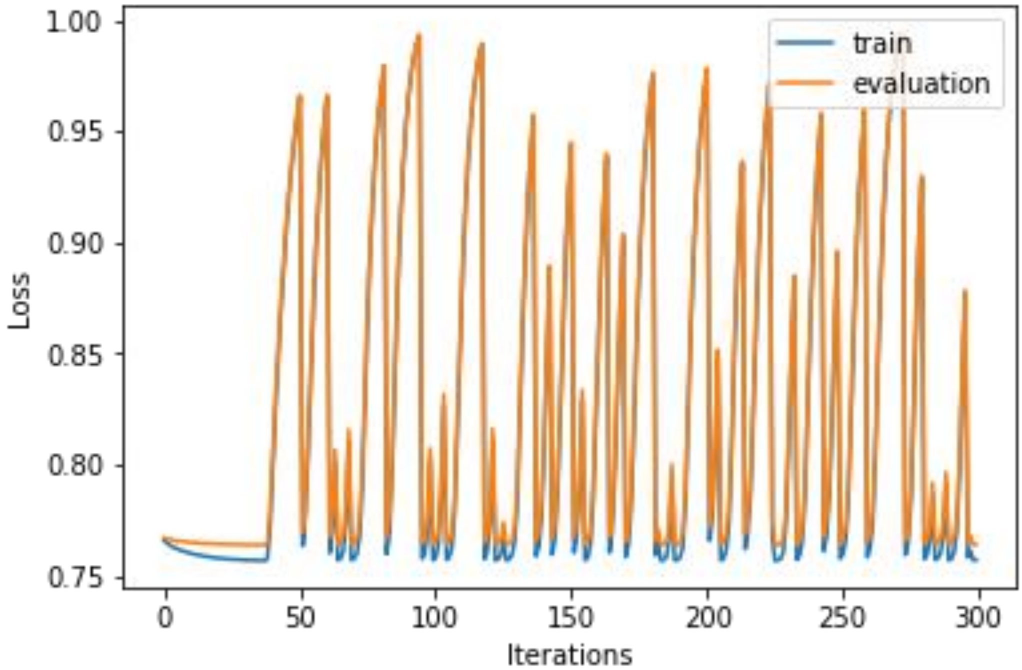
超参数选择：

```
learning_rate = 0.3
maxIteration = 300
```

预测结果（最佳结果）：

0.763797625979

loss 曲线图：



4) AdaDelta

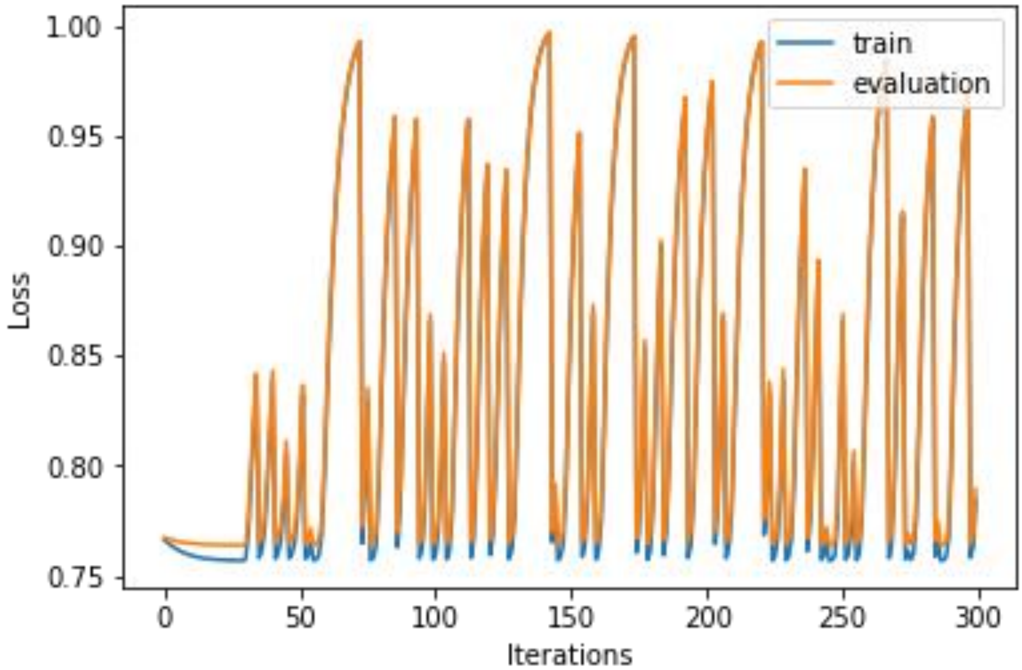
超参数选择：

```
learning_rate = 2000
maxIteration = 300
```

预测结果（最佳结果）：

0.763807229843

loss 曲线图：



5) Adam

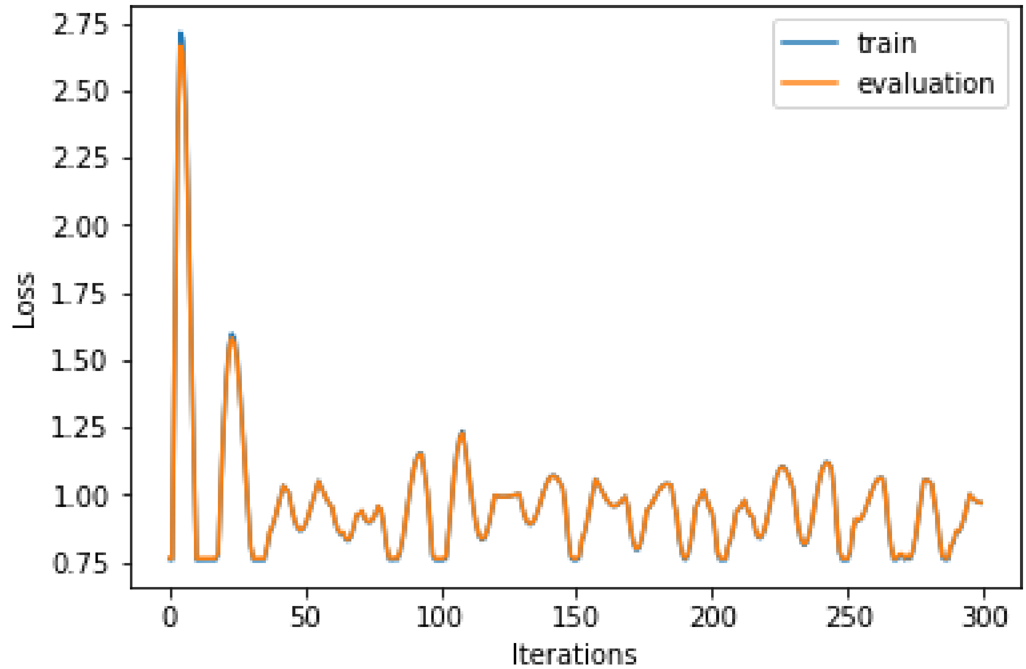
超参数选择：

```
learning_rate = 0.05
maxIteration = 300
```

预测结果（最佳结果）：

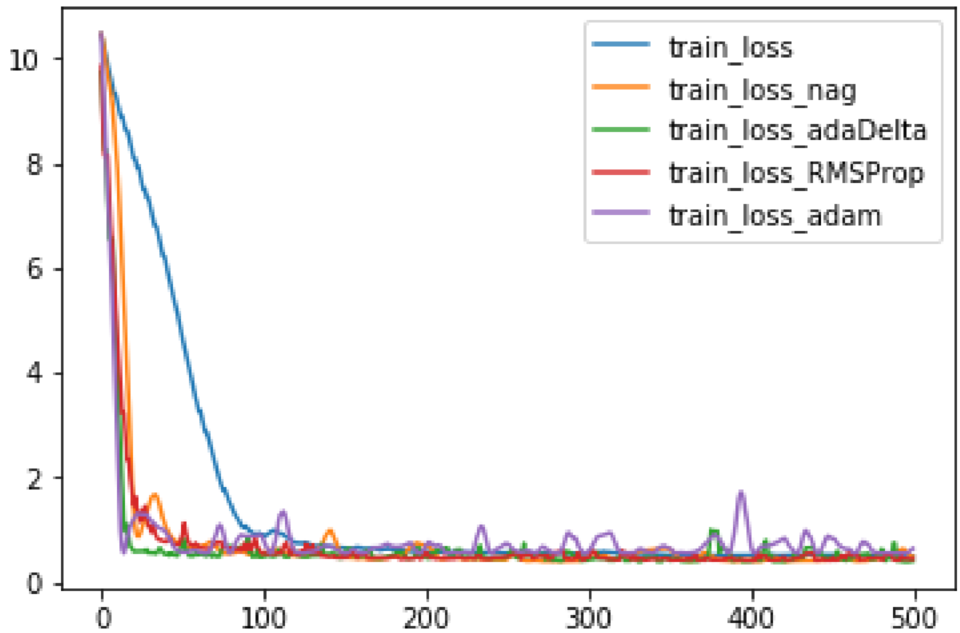
0.763805383034

loss 曲线图：



11.实验结果分析:

逻辑回归：

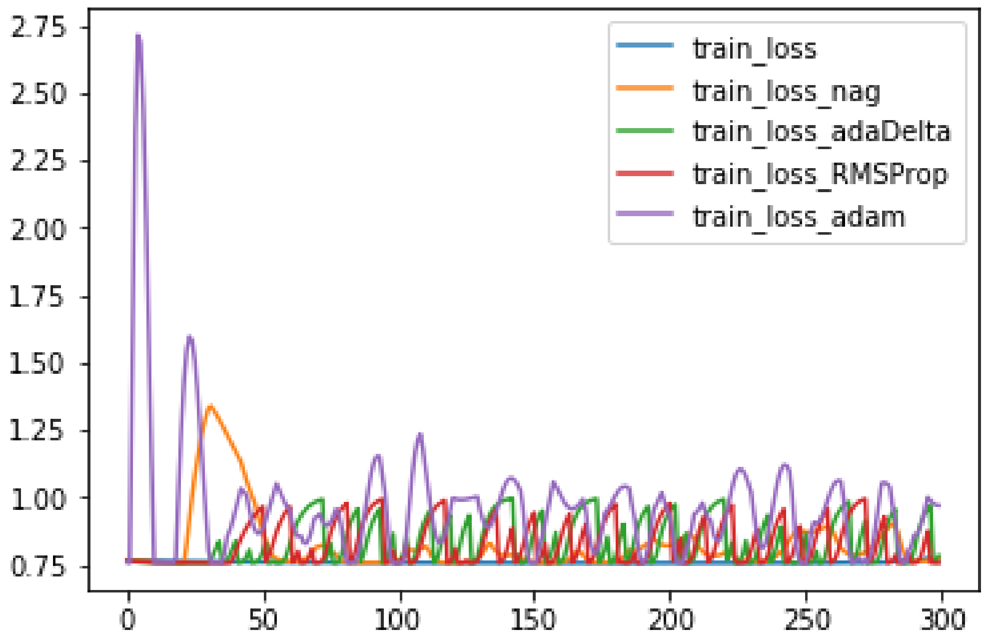


对比分析五种不同的梯度下降可以得出以下结论：

1) 就该问题，在当前数据集上，train_loss 下降较为缓慢，而其他几种方式都下降得比较快

2) RMSProp 方法具有最好的效果，而 Adam 方法结果较差。

线性分类：



对比分析五种不同的梯度下降可以得出以下结论：

1) 就该问题，在当前数据集上，train_loss_adam 的波动较大，较不稳定快
2) RMSProp 方法具有最好的效果，而 Adam 方法结果较差。

12.对比逻辑回归和线性分类的异同点：

相同：

逻辑回归和线性分类都可以总结成为一种流程

即

1.选择一个合适的模型
2.选合适的 loss 函数
3.通过各种方法（这里是梯度下降的方法）求出最好的模型参数

差异：

逻辑回归致力于解决连续性问题，而线性分类致力于解决离散性问题。他们的输出具有较大的差异。

13.实验总结：

在本次实验中，我使用 python 实现逻辑回归、线性分类和随机梯度下降算法，这加深了对逻辑回归、线性分类和随机梯度下降算法的理解，有了更深的体会。此外，还实现了多种不同的梯度下降方式，并且尝试对比他们的优势和劣势，对这几种方法有了更加深入的认识。