South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
谢家骏

Supervisor:
吴庆耀

Student ID：
201530613191

Grade:
2015 级

December 15, 2017

Linear Regression, Linear Classification and Gradient Descent

Abstract—

I. INTRODUCTION

实验目的

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

数据集

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。请自行下载训练集和验证集。

实验步骤

本次实验代码及画图均在 jupyter 上完成。

逻辑回归与随机梯度下降

读取实验训练集和验证集。
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。
选择 Loss 函数及对其求导，过程详见课件 ppt。
求得部分样本对 Loss 函数的梯度。
使用不同的优化方法更新模型参数（NAG，RMSProp，AdaDelta 和 Adam）。
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。在验证集上测试并得到不同优化方法的 Loss 函数值，，和。
重复步骤 4-6 若干次，画出，，和随迭代次数的变化图。

线性分类与随机梯度下降

读取实验训练集和验证集。
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。
选择 Loss 函数及对其求导，过程详见课件 ppt。
求得部分样本对 Loss 函数的梯度。
使用不同的优化方法更新模型参数（NAG，RMSProp，AdaDelta 和 Adam）。
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。在验证集上测试并得到不同优化方法的 Loss 函数值，，和。
重复步骤 4-6 若干次，画出，，和随迭代次数的变化图。

II. METHODS AND THEORY

Loss()函数

随机梯度下降函数

NAG 优化算法

$$\mathbf{g}_t \leftarrow \nabla J(\boldsymbol{\theta}_{t-1} - \gamma \mathbf{v}_{t-1})$$
$$\mathbf{v}_t \leftarrow \gamma \mathbf{v}_{t-1} + \eta \mathbf{g}_t$$
$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \mathbf{v}_t$$

RMSProp 优化算法

$$\mathbf{g}_t \leftarrow \nabla J(\boldsymbol{\theta}_{t-1})$$
$$G_t \leftarrow \gamma G_t + (1-\gamma)\mathbf{g}_t \odot \mathbf{g}_t$$
$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot \mathbf{g}_t$$

AdaDelta 优化算法

$$\mathbf{g}_t \leftarrow \nabla J(\boldsymbol{\theta}_{t-1})$$
$$G_t \leftarrow \gamma G_t + (1-\gamma)\mathbf{g}_t \odot \mathbf{g}_t$$
$$\Delta\boldsymbol{\theta}_t \leftarrow -\frac{\sqrt{\Delta_{t-1}+\epsilon}}{\sqrt{G_t+\epsilon}} \odot \mathbf{g}_t$$
$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} + \Delta\boldsymbol{\theta}_t$$
$$\Delta_t \leftarrow \gamma\Delta_{t-1} + (1-\gamma)\Delta\boldsymbol{\theta}_t \odot \Delta\boldsymbol{\theta}_t$$

Adam 优化算法

$$\mathbf{g}_t \leftarrow \nabla J(\boldsymbol{\theta}_{t-1})$$
$$\mathbf{m}_t \leftarrow \beta_1\mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t$$
$$G_t \leftarrow \gamma G_t + (1-\gamma)\mathbf{g}_t \odot \mathbf{g}_t$$
$$\alpha \leftarrow \eta\frac{\sqrt{1-\gamma^t}}{1-\beta^t}$$
$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha\frac{\mathbf{m}_t}{\sqrt{G_t+\epsilon}}$$

III. EXPERIMENT

逻辑回归与随机梯度下降：

实验代码

```
from sklearn import datasets as ds
from sklearn.model_selection
import train_test_split
import numpy as np
import math
from numpy import random
import matplotlib.pyplot as plt

def Loss(x_train,y_train,w,lmd):
    temp1=0
    n=0
    for x_sample, y_sample in zip(x_train,y_train):
        s=-y_sample*np.dot(x_sample,w)
        temp1+=np.log(1+np.exp(0))
        n+=1
    loss=temp1/n+lmd*np.dot(w.T,w)/2
    return loss

def Grad(x_train,y_train,w,lmd):
    temp2=0
    n=0
    for x_sample, y_sample in zip(x_train,y_train):
        temp2+=y_sample*x_sample/(1 +
np.exp(np.dot(y_sample, np.dot(x_sample, w))))
        n+=1
    grad=-temp2/n+lmd*w
    return grad

def NAG(x_train,y_train,w,lmd,v,h):
    g=Grad(x_train,y_train,w,lmd)
    v=lmd*v+h*g
    w=w-v
    return w

def RMSProp(x_train,y_train,w,lmd,e,h,G):
    g=Grad(x_train,y_train,w,lmd)
    G=lmd*G+np.dpt(1-lmd,np.dot(g,g))
    w=w-np.dot(h/np.sqrt(G+e),g)
    return w

def AdaDelta(x_train,y_train,w,lmd,e,dt,G):
    g=Grad(x_t,y_t,w,lmd)
    G=lmd*G+np.dpt(1-lmd,np.dot(g,g))
    dw=-np.dot(np.sqrt(dt+e)/np.sqrt(G+e),g)
    w=w+dw
    dt=lmd*dt+np.dot(1-lmd,np.dot(dw,dw))
    return w

def Adam(x_train,y_train,w,lmd,b,m,e,h,G):
    g=Grad(x_t,y_t,w,lmd)
    m=b*m+(1-b)*g
    G=lmd*G+np.dpt(1-lmd,np.dot(g,g))
    a1=h*(np.sqrt(1-lmd)/(1-b))
    lmd*=lmd
    b*=b
    w=w-a1*(m/np.sqrt(G+e))
    return w

def
iterationNAG(x_train,x_validation,y_train,y_validat
```

```
ion,w,lmd,v,h):
    itera = 100
    lr = 0.0001
    train_loss=[]
    validation_loss=[]
    for i in range(itera):
        loss_t=Loss(x_train,y_train,w,lmd)
        tx,ty=x_train.shape
        lt=loss_t[0,0]/tx
        train_loss.append(lt)

loss_v=Loss(x_validation,y_validation,w,lmd)
        vx,vy=x_validation.shape
        lv=loss_v[0,0]/vx
        validation_loss.append(lv)
        w=NAG(x_train,y_train,w,lmd,v,h)
    return w,train_loss,validation_loss

  def
iterationRMSProp(x_train,x_validation,y_train,y_v
alidation,w,lmd,e,h,G):
    itera = 100
    lr = 0.0001
    train_loss=[]
    validation_loss=[]
    for i in range(itera):
        loss_t=Loss(x_train,y_train,w,lmd)
        tx,ty=x_train.shape
        lt=loss_t[0,0]/tx
        train_loss.append(lt)

loss_v=Loss(x_validation,y_validation,w,lmd)
        vx,vy=x_validation.shape
        lv=loss_v[0,0]/vx
        validation_loss.append(lv)
        w=RMSProp(x_train,y_train,w,lmd,e,h,G)
    return w,train_loss,validation_loss

  def
iterationAdaDelta(x_train,x_validation,y_train,y_va
lidation,w,lmd,e,dt,G):
    itera = 100
    lr = 0.0001
    train_loss=[]
    validation_loss=[]
    for i in range(itera):
        loss_t=Loss(x_train,y_train,w,lmd)
        tx,ty=x_train.shape
        lt=loss_t[0,0]/tx
        train_loss.append(lt)

loss_v=Loss(x_validation,y_validation,w,lmd)
        vx,vy=x_validation.shape
        lv=loss_v[0,0]/vx
        validation_loss.append(lv)
        w=AdaDelta(x_train,y_train,w,lmd,e,dt,G)
    return w,train_loss,validation_loss

  def
iterationAdam(x_train,x_validation,y_train,y_valida
tion,w,lmd,b,m,e,h,G):
    itera = 100
    lr = 0.0001
    train_loss=[]
    validation_loss=[]
    for i in range(itera):
        loss_t=Loss(x_train,y_train,w,lmd)
        tx,ty=x_train.shape
        lt=loss_t[0,0]/tx
        train_loss.append(lt)

loss_v=Loss(x_validation,y_validation,w,lmd)
        vx,vy=x_validation.shape
        lv=loss_v[0,0]/vx
        validation_loss.append(lv)
        w=Adam(x_train,y_train,w,lmd,b,m,e,h,G)
    return w,train_loss,validation_loss

  if __name__ == '__main__':
    x_train,          y_train          =
ds.load_svmlight_file('E:/test/train.txt')
    x_validation,       y_validation       =
ds.load_svmlight_file('E:/test/val.txt')
    x_train = x_train.toarray()
    x_t=[]
    y_t=[]
    yl=len(y_train)
    y_train=y_train.reshape(yl,1)
    n_sample,n_feature=x_train.shape
    for sample in np.random.randint(1,100,[1,25]):
        x_t.append(x_train[sample])
        y_t.append(y_train[sample])
    w0=np.zeros(shape=(n_feature,1))
    lmd=0.9
```

```
    v=0.01
    h=0.01

wn,train_lossn,validation_lossn=iterationNAG(x_train,x_validation,y_train,y_validation,w0,lmd,v,h)
    plt.plot(validation_lossn, label='validation loss')
    plt.xlabel('Iteration')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    w1=np.zeros(shape=(n_feature,1))
    h1=0.001
    e=0
    G=0.001

wr,train_lossr,validation_lossr=iterationRMSProp(x_train,x_validation,y_train,y_validation,w1,lmd,e,h1,G)
    plt.plot(validation_lossr, label='validation loss')
    plt.xlabel('Iteration')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    w2=np.zeros(shape=(n_feature,1))
    lmd_1=0.95
    dt=0

wa,train_lossa,validation_lossa=iterationAdaDelta(x_train,x_validation,y_train,y_validation,w2,lmd_1,e,dt,G)
    plt.plot(validation_lossa, label='validation loss')
    plt.xlabel('Iteration')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    w3=np.zeros(shape=(n_feature,1))
    lmd_2=0.999
    b=0.9
    m=0

wam,train_lossam,validation_lossam=iterationAdam(x_train,x_validation,y_train,y_validation,w3,lmd_2,b,m,e,h1,G)
    plt.plot(validation_lossam, label='validation loss')
    plt.xlabel('Iteration')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
```

实验结果

尚未运行出

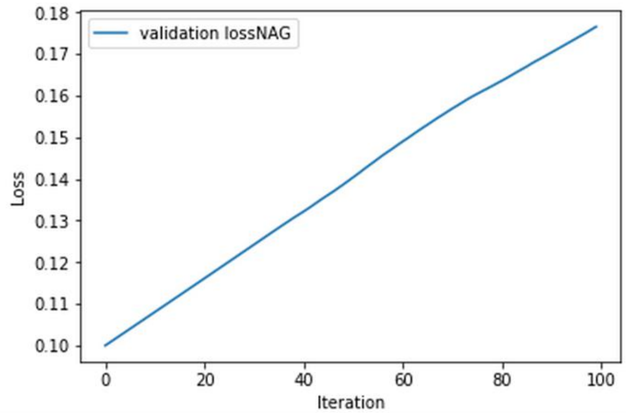
线性分类与随机梯度下降

实验代码

```
from sklearn import datasets as ds
from sklearn.cross_validation import train_test_split
import numpy as np
import matplotlib.pyplot as plt
import os
import matplotlib.pyplot as plt


def Loss(x_train,y_train,w,C,b):
    ls=0
    i=0
    for x_sample, y_sample in zip(x_train,y_train):
        temp = np.ones((n_sample,1)) - y_sample * np.dot(x_sample,w)
        if(0<temp[i,0]):
            ls+=temp[i,0]
        else:
            ls+=0
        i+=1
    loss=np.dot(w.T,w)/2+C*ls
    return loss

def Gradw(x_train,y_train,w,C,b):
    g=0
    i=0
    for x_sample, y_sample in zip(x_train,y_train):
        temp = np.ones((n_sample,1)) - y_sample * np.dot(x_sample,w)
        if(temp[i,0]>0):
            gw=-y_sample*x_sample
        else:
            gw=0
        g+=gw
        i+=1
    gradw=w+C*g
    return gradw

def Gradb(x_train,y_train,w,C,b):
```

```
    g1=0
    i=0
    for x_sample, y_sample in zip(x_train,y_train):
      temp = np.ones((n_sample,1)) - y_sample * 
np.dot(x_sample,w)
      if(temp[i,0]>0):
        gb=-y_sample
      else:
        gb=0
      g1+=gb
      i+=1
    gradb=C*g1
    return gradb

  def NAG(x_train,y_train,w,lmd,v,h):
    g=Grad(x_train,y_train,w,lmd)
    v=lmd*v+h*g
    w=w-v
    return w

  def RMSProp(x_train,y_train,w,lmd,e,h,G):
    g=Grad(x_train,y_train,w,lmd)
    G=lmd*G+np.dpt(1-lmd,np.dot(g,g))
    w=w-np.dot(h/np.sqrt(G+e),g)
    return w

  def AdaDelta(x_train,y_train,w,lmd,e,dt,G):
    g=Grad(x_t,y_t,w,lmd)
    G=lmd*G+np.dpt(1-lmd,np.dot(g,g))
    dw=-np.dot(np.sqrt(dt+e)/np.sqrt(G+e),g)
    w=w+dw
    dt=lmd*dt+np.dot(1-lmd,np.dot(dw,dw))
    return w

  def Adam(x_train,y_train,w,lmd,b,m,e,h,G):
    g=Grad(x_t,y_t,w,lmd)
    m=b*m+(1-b)*g
    G=lmd*G+np.dpt(1-lmd,np.dot(g,g))
    a1=h*(np.sqrt(1-lmd)/(1-b))
    lmd*=lmd
    b*=b
    w=w-a1*(m/np.sqrt(G+e))
    return w

  def 
iteration(x_train,x_validation,y_train,y_validation,
w,C,b):
    itera = 100
    lr = 0.001
    train_loss=[]
    validation_loss=[]
    for i in range(itera):
      loss_t=Loss(x_train,y_train,w,C,b)
      tx,ty=x_train.shape
      lt=loss_t[0,0]/tx
      train_loss.append(lt)

loss_v=Loss(x_validation,y_validation,w,C,b)
      vx,vy=x_validation.shape
      lv=loss_v[0,0]/vx
      validation_loss.append(lv)
      w=w-lr*Gradw(x_train,y_train,w,C,b)
      b=b-lr*Gradb(x_train,y_train,w,C,b)
    return w,b,train_loss,validation_loss

  if __name__ == '__main__':
    C=0.1
    x_train,          y_train          = 
ds.load_svmlight_file('E:/test/2.txt')
    x_train = x_train.toarray()
    yl=len(y_train)
    y_train=y_train.reshape(yl,1)
    n_sample,n_feature=x_train.shape
    x_train, x_validation, y_train, y_validation = 
train_test_split(x_train,    y_train,    test_size=0.2, 
random_state=1)
    n_sample,n_feature=x_train.shape
    w0=np.zeros(shape=(n_feature,1))
    b=0

w,b,train_lossNAG,validation_lossNAG=iteration(x
_train, x_validation, y_train, y_validation,w0,C,b)
    plt.plot(validation_lossNAG,  label='validation 
lossNAG')
    plt.xlabel('Iteration')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
```

实验结果（部分）



IV. CONCLUSION

总体来说，自己对逻辑回归、线性分类和随机梯度下降的知识掌握得并不好，导致在逻辑回归与随机梯度下降的实验中代码写得不好，运行过慢，得不出结果，而在线性分类与随机梯度下降的实验中 loss 曲线则随着迭代次数呈上升趋势。在之后的学习中我会继续学习这一部分内容，尽快掌握。但通过这次的实验我也获益良多，了解了四种梯度下降优化算法和随机梯度下降，进一步加深了对逻辑回归、线性分类的理解。