The Experiment Report of Machine Learning



December 10, 2017

Grade:

Undergraduate

Student ID：201530613061

Supervisor:

Qingyao Wu

Author:Weimin Wu

**SUBJECT:**SOFTWARE ENGINEERING

**SCHOOL:** SCHOOL OF SOFTWARE ENGINEERING

[[1]](#footnote-0)

Logistic regression, linear classification and stochastic gradient descent

Abstract—

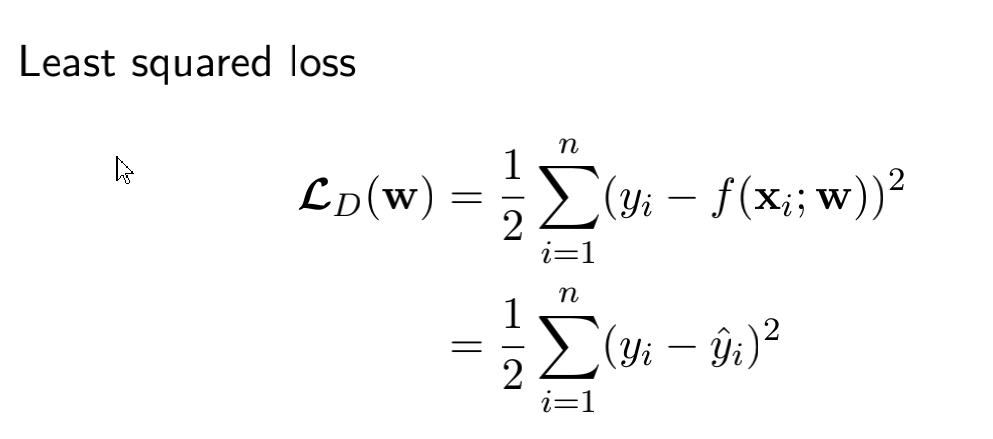
# INTRODUCTION

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，进一步理解支持向量机的原理并在较大数据上实践。

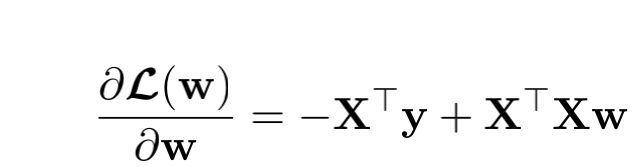
# METHODS AND THEORY



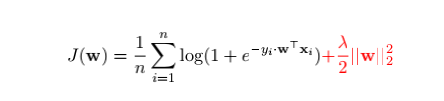
线性分类损失函数



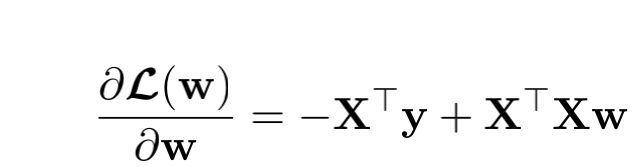
线性分类损失导数



逻辑回归损失函数



逻辑回归损失导数



# Experiment

1. Dataset

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。

B. Implementation

|  |  |
| --- | --- |
| 学习率 | 0.001 |
| 阈值 | 0.0 |
| e (防止分母为0) | 10^(-8) |
| C | 0 |
| P1(衰减1) | 0.9 |
| P2(衰减2) | 0.999 |
| y | 0.9 |

线性分类

class LinearClassification(object):

def \_\_init\_\_(self,Learning\_rate=0.001,threshold=0.0,epoch=8):

self.Learning\_rate=Learning\_rate

self.epoch=epoch

def fit\_nag(self,x,Y,threshold,C):

self.w=np.zeros((1,1+x.shape[1]))

self.v=np.zeros((1,x.shape[1]))

self.cost\_list=[]

self.threshold=threshold

self.C=C

self.y=0.9

for i in range(self.epoch):

cost=0

for j in range(x.shape[0]):

output=self.Classification\_input(x[j,:],self.threshold)

if(output!=Y[j]):

#计算动量项

t=Y[j]\*(x[j,:])

#print('t.shape:',t.shape)

#print('self.w.shape:',self.w.shape)

#print('self.v.shape:',self.v.shape)

#print('self.y.shape:',self.y.shape)

dwv=-self.C\*t+self.w[0,1:]-self.y\*self.v

self.v=self.y\*self.v+self.Learning\_rate\*dwv

#计算cost值

cost+=(1-Y[j]\*output)

#print('cost:',cost)

#更新w

self.w[0,1:]=self.w[0,1:]-self.v

self.w[0,0]=self.w[0,0]-self.v.sum()

self.w1=self.w.T

cost=cost+(self.w\*self.w1).sum()/2

self.cost\_list.append(cost)

return self

def fit\_adadelta(self,x,Y,threshold,C):

self.e=10\*\*(-8)

self.cost\_list=[]

self.w=np.zeros((1,1+x.shape[1]))

self.v=np.zeros((1,1+x.shape[1]))

self.threshold=threshold

self.C=C

self.y=0.9

for i in range(self.epoch):

cost=0

for j in range(x.shape[0]):

output=self.Classification\_input(x[j,:],self.threshold)

if(output!=Y[j]):

#计算梯度gt

t=Y[j]\*(x[j,:])

g=self.w[0,1:]-self.C\*t

#计算E[g^2]t

g1=g.T

E=(g\*g1).sum()

#计算RMS[g]t

RMSgt=np.sqrt(self.e+E)

#计算dθt,保留学习率

dw=(-self.Learning\_rate/RMSgt)\*g

#计算E[dw^2]t-1

dw1=dw.T

Edwt=(dw\*dw1).sum()

#计算RMS[dw]t-1

RMSwt1=np.sqrt(Edwt+self.e)

#计算dθt,去除学习率

dwt=-(RMSwt1/RMSgt)\*g

#计算E[dw^2]t,dw1是dw转置

dw1=dw.T

Edwt=self.y\*Edwt+(1-self.y)\*((dw\*dw1).sum())

#计算cost

cost+=(1-Y[j]\*output)

#更新w

self.w[0,1:]=self.w[0,1:]+dwt

self.w[0,0]=self.w[0,0]+dwt.sum()

self.w1=self.w.T

cost=cost+(self.w\*self.w1).sum()/2

self.cost\_list.append(cost)

return self

def fit\_rms(self,x,Y,threshold,C):

self.e=10\*\*(-8)

self.cost\_list=[]

self.w=np.zeros((1,1+x.shape[1]))

self.v=np.zeros((1,1+x.shape[1]))

self.threshold=threshold

self.C=C

self.y=0.9

for i in range(self.epoch):

cost=0

for j in range(x.shape[0]):

output=self.Classification\_input(x[j,:],self.threshold)

if(output!=Y[j]):

#计算梯度gt

t=Y[j]\*(x[j,:])

g=self.w[0,1:]-self.C\*t

#计算E[g^2]t

g1=g.T

E=(g\*g1).sum()

#计算RMS[g]t

RMSgt=np.sqrt(self.e+E)

#计算dw

dw=(-self.Learning\_rate/RMSgt)\*g

#计算cost

cost+=(1-Y[j]\*output)

#更新w

self.w[0,1:]=self.w[0,1:]+dw

self.w[0,0]=self.w[0,0]+dw.sum()

self.w1=self.w.T

cost=cost+(self.w\*self.w1).sum()/2

self.cost\_list.append(cost)

return self

def fit\_Adam(self,x,Y,threshold,C):

self.e=10\*\*(-8)

self.cost\_list=[]

self.w=np.zeros((1,1+x.shape[1]))

self.m=np.zeros((1,x.shape[1]))

self.v=0

self.C=C

self.threshold=threshold

self.p1=0.9

self.p2=0.999

for i in range(self.epoch):

cost=0

for j in range(x.shape[1]):

output=self.Classification\_input(x[j,:],self.threshold)

if(output!=Y[i]):

#计算梯度gt

t=Y[j]\*(x[j,:])

g=self.w[0,1:]-self.C\*t

#计算m

self.m=self.m\*self.p1+(1-self.p1)\*g

#计算m的修正误差M

M=self.m/(1-(self.p1\*\*(j+1)))

#计算v

g1=g.T

G=(g1\*g).sum()

self.v=self.v\*self.p2+(1-self.p2)\*G

#计算V

V=self.v/(1-(self.p2\*\*(j+1)))

#计算cost

cost+=(1-Y[j]\*output)

#计算W

self.w[0,1:]=self.w[0,1:]-(self.Learning\_rate/(np.sqrt(V)+self.e))\*M

self.w[0,0]=self.w[0,0]-((self.Learning\_rate/(np.sqrt(V)+self.e))\*M).sum()

self.w1=self.w.T

cost=cost+(self.w\*self.w1).sum()/2

self.cost\_list.append(cost)

return self

def Classification\_input(self,x,threshold):

#print('x.shape:',x.shape)

#print('w[0,1:].shape:',self.w[0,1:].shape)

f=x\*self.w[0,1:]+self.w[0,0]

if(f>=threshold):

return 1

else:

return -1

逻辑回归

class LogisticRegression(object):

def \_\_init\_\_(self,Learning\_rate=0.001,epoch=6):

self.Learning\_rate=Learning\_rate

self.epoch=epoch

self.threshold=0.5

self.y=0.9

def fit\_NAG(self,x,Y):

self.w=np.zeros((1,x.shape[1]))

self.cost\_list=[]

self.v=np.zeros((1,x.shape[1]))

for i in range(self.epoch):

cost=0

for j in range(x.shape[0]):

output=self.Logistic\_input(x[j,:],self.threshold)

if(output!=Y[j]):

t=-Y[j]\*(x[j,:])

dwv=self.w-self.y\*self.v

t=t.T.dot(dwv)

t=t.sum()

dv=self.w-(Y[j]\*(x[j,:]))/(1+np.exp(t))

self.v\*=self.y

self.v+=self.Learning\_rate\*dv

self.w=self.w-self.v

cost+=np.log(1+np.exp((-Y[j]\*x[j,:].T.dot(self.w)).sum()))

cost=cost/x.shape[0]+(self.w\*\*2).sum()/2

self.cost\_list.append(cost)

#print('self.cost\_list:',self.cost\_list)

return self

def fit\_Rms(self,x,Y):

self.e=10\*\*(-8)

self.cost\_list=[]

self.w=np.zeros((1,x.shape[1]))

for i in range(self.epoch):

cost=0

for j in range(x.shape[0]):

output=self.Logistic\_input(x[j,:],self.threshold)

if(output!=Y[j]):

#计算梯度gt

t=Y[j]\*(x[j,:])

h=self.Logistic\_input(t,self.threshold)

g=self.w-t\*h

#计算E[g^2]t

g1=g.T

E=g\*g1

E=E.sum()

#计算RMS[g]t

RMSgt=np.sqrt(self.e+E)

#计算dw

dw=(-self.Learning\_rate/RMSgt)\*g

#更新w

self.w=self.w+dw

#计算cost

cost+=np.log(1+np.exp((-Y[j]\*x[j,:].T.dot(self.w)).sum()))

self.w1=self.w.T

cost=cost/x.shape[0]+(self.w\*self.w1).sum()/2

self.cost\_list.append(cost)

return self

def fit\_Adadelta(self,x,Y):

self.e=10\*\*(-8)

self.cost\_list=[]

self.w=np.zeros((1,x.shape[1]))

self.v=np.zeros((1,x.shape[1]))

for i in range(self.epoch):

cost=0

for j in range(x.shape[0]):

output=self.Logistic\_input(x[j,:],self.threshold)

if(output!=Y[i]):

#计算梯度gt

t=Y[j]\*(x[j,:])

h=self.Logistic\_input(t,self.threshold)

g=self.w-t\*h

#计算E[g^2]t

g1=g.T

E=g\*g1

E=E.sum()

#print('E:',E)

#E=np.mean(E)

#计算RMS[g]t

RMSgt=np.sqrt(self.e+E)

#计算dθt,保留学习率

dw=(-self.Learning\_rate/RMSgt)\*g

#计算E[dw^2]t-1

dw1=dw.T

Edwt=dw\*dw1

Edwt=Edwt.sum()

#Edwt=np.mean(Edwt)

#计算RMS[dw]t-1

RMSwt1=np.sqrt(Edwt+self.e)

#计算dθt，去掉学习率

dwt=-(RMSwt1/RMSgt)\*g

#计算E[dw^2]t,dw1是dw转致

dw1=dw.T

Edwt=self.y\*Edwt+(1-self.y)\*((dw\*dw1).sum())

#Wt+1=Wt+dWt，更新w

#print(self.w.shape)

#print('dwt.shape:',dwt.shape)

self.w=self.w+dwt

#print('self.w.shape:',self.w.shape)

#print('self.x[j,:].shape:',x[j,:].shape)

#计算cost

cost+=np.log(1+np.exp((-Y[j]\*x[j,:].T.dot(self.w)).sum()))

#print(cost)

self.w1=self.w.T

cost=cost/x.shape[0]+(self.w\*self.w1).sum()/2

self.cost\_list.append(cost)

return self

def fit\_Adam(self,x,Y):

self.e=10\*\*(-8)

self.cost\_list=[]

self.w=np.zeros((1,x.shape[1]))

self.m=np.zeros((1,x.shape[1]))

self.v=0

self.p1=0.900

self.p2=0.999

for i in range(self.epoch):

cost=0

for j in range(x.shape[1]):

output=self.Logistic\_input(x[j,:],self.threshold)

if(output!=Y[i]):

#计算梯度gt

t=Y[j]\*(x[j,:])

h=self.Logistic\_input(t,self.threshold)

g=self.w-t\*h

#计算m

self.m=self.m\*self.p1+(1-self.p1)\*g

#计算m的修正误差M

M=self.m/(1-(self.p1\*\*(j+1)))

#计算v

g1=g.T

G=(g1\*g).sum()

self.v=self.v\*self.p2+(1-self.p2)\*G

#计算V

V=self.v/(1-(self.p2\*\*(j+1)))

#计算cost

cost+=np.log(1+np.exp((-Y[j]\*x[j,:].T.dot(self.w)).sum()))

#计算W

self.w=self.w-(self.Learning\_rate/(np.sqrt(V)+self.e))\*M

self.w1=self.w.T

cost=cost/x.shape[0]+(self.w\*self.w1).sum()/2

self.cost\_list.append(cost)

return self

def Logistic\_input(self,x,threshold):

t=x.T.dot(self.w)

t=t.sum()

h=1/(1+np.exp(t))

if(h<self.threshold):

return -1

else:

return 1

# conclusion

通过这次实验，我是真真实实的收获到很多东西，从课堂上讲的听起来好像比较简单，到真正实践才发现有很多困难。自己一个人独立完成，找到自己不懂的地方，不断搞

懂

1. [↑](#footnote-ref-0)