

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  丁可

**学 号 201530611449**

**邮 箱 952214630@qq.com**

**指导教师**

**提交日期** **2017年12月8日**

## 1. 实验题目: 线性回归、线性分类与梯度下降

## 2. 实验时间：2017年 月 日

## 3. 报告人:丁可

## 4. 实验目的:

1. 进一步理解线性回归和梯度下降的原理。
2. 在小规模数据集上实践。
3. 体会优化和调参的过程。

## 5. 数据集以及数据分析：

线性回归使用的是LIBSVM Data中的Housing数据，包含506个样本，每个样本有13个属性。请自行下载scaled版本，并将其切分为训练集，验证集。

线性分类使用的是LIBSVM Data中的australian数据，包含690个样本，每个样本有14 个属性。请自行下载scaled版本，并将其切分为训练集，验证集。

## 6. 实验步骤:

*线性回归和梯度下降*

1. 读取实验数据，使用sklearn库的[load\_svmlight\_file](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html" \t "_blank)函数读取数据。
2. 将数据集切分为训练集和验证集，本次实验不切分测试集。使用[train\_test\_split](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)函数切分数据集。
3. 线性模型参数初始化，可以考虑全零初始化，随机初始化或者正态分布初始化。
4. 选择Loss函数及对其求导，过程详见课件ppt。
5. 求得**所有样本**对Loss函数的梯度。
6. 取梯度的负方向，记为。
7. 更新模型参数，。为学习率，是人为调整的超参数。
8. 在训练集上测试并得到Loss函数值，在验证集上测试并得到Loss函数值。
9. 重复步骤5-8若干次，**画出和随迭代次数的变化图**。

*线性分类和梯度下降*

1. 读取实验数据，使用sklearn库的[load\_svmlight\_file](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html" \t "_blank)函数读取数据。
2. 将数据集切分为训练集和验证集，本次实验不切分测试集。使用[train\_test\_split](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)函数切分数据集。
3. 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。
4. 选择Loss函数及对其求导，过程详见课件ppt。
5. 求得**所有样本**对Loss函数的梯度。
6. 取梯度的负方向，记为。
7. 更新模型参数，。为学习率，是人为调整的超参数。
8. 选择合适的阈值，将计算结果**大于阈值的标记为正类，反之为负类**。在训练集上测试并得到Loss函数值，在验证集上测试并得到Loss函数值。
9. 重复步骤5-8若干次，**画出和随迭代次数的变化图**。

## 7. 代码内容:

**线性回归：**

# coding=gbk

from sklearn.datasets import load\_svmlight\_file

import numpy as np

import matplotlib.pyplot as plt

from scipy.sparse import hstack

x,y=load\_svmlight\_file("C:/Users/Bean/Desktop/机器学习实验课/housing\_scale.txt")

from sklearn.model\_selection import train\_test\_split

train\_x,cv\_x,train\_y,cv\_y=train\_test\_split(x,y,test\_size=0.2,random\_state=0)

def GradientDescent(x, y, theta=None, alpha=0.001, num\_iter=12000):

x0 = np.ones((len(y), 1))

x = hstack([x0, x])

if theta == None:

theta= np.zeros(x.shape[1])

train\_loss=[0]\*num\_iter

m=len(y)

for i in range(num\_iter):

h = x.dot(theta)

loss = h - y

gradient = x.T.dot(loss)/m

theta = theta - alpha \* gradient

loss=np.sum((x.dot(theta) - y) \*\* 2) / (2 \* m)

train\_loss[i]=loss

return theta,train\_loss

update\_theta,train\_loss=GradientDescent(train\_x, train\_y)

print(update\_theta)

a,cv\_loss=GradientDescent(cv\_x,cv\_y)

import matplotlib.pyplot as plt

plt.figure('loss', figsize=(12, 10))

plt.plot(train\_loss, label='train loss')

plt.plot(cv\_loss, label='validation loss')

plt.xlabel('iterations')

plt.ylabel('loss')

plt.legend()

plt.show()

**线性分类：**

# coding=gbk  
from sklearn.datasets import load\_svmlight\_file  
from sklearn.model\_selection import train\_test\_split  
import numpy as np  
from sklearn.preprocessing import LabelEncoder  
import matplotlib.pyplot as plt  
x,y=load\_svmlight\_file("C:/Users/Bean/Desktop/机器学习实验课/australian\_scale.txt")  
train\_x,cv\_x,train\_y,cv\_y=train\_test\_split(x,y,test\_size=0.2,random\_state=0)  
  
def lossFunction(theta,x,y,reg):  
 num\_train=x.shape[0]  
 num\_class=theta.shape[1]  
 scores=x.dot(theta)  
 correct\_class\_scores = scores[range(num\_train), list(y)].reshape(-1, 1)  
 margins = np.maximum(0, scores - correct\_class\_scores + 1)  
 margins[range(num\_train), list(y)] = 0  
 loss = np.sum(margins) / num\_train + 0.5 \* reg \* np.sum(theta \* theta)  
 coeff\_mat = np.zeros((num\_train, num\_class))  
 coeff\_mat[margins > 0] = 1  
 coeff\_mat[range(num\_train), list(y)] = 0  
 coeff\_mat[range(num\_train), list(y)] = - np.sum(coeff\_mat, axis=1)  
  
 dw = x.T.dot(coeff\_mat)  
 dw = dw / num\_train + reg \* theta  
 return loss, dw  
  
  
class LinearClassifier:  
 def \_\_init\_\_(self):  
 self.theta = None  
 self.le = LabelEncoder()  
  
 def train(self, X, y, learning\_rate=1e-3, reg=1e-5, num\_iters=2000, batch\_size=200, verbose=False):  
 num\_train, dim = X.shape #维度  
 y = self.le.fit\_transform(y)  
  
  
 num\_classes = int(np.max(y)) + 1  
 if self.theta is None:  
 print(dim, num\_classes)  
 self.theta = 0.001 \* np.random.randn(dim, num\_classes)  
  
 train\_loss = []  
 for i in range(num\_iters):  
 batch\_idx = np.random.choice(num\_train, batch\_size, replace=True)  
 X\_batch = X[batch\_idx]  
 y\_batch = y[batch\_idx]  
  
 loss, grad = self.loss(X\_batch, y\_batch, reg)  
  
 train\_loss.append(loss)  
  
 self.theta += -1 \* learning\_rate \* grad  
  
 if verbose and i % 100 == 0:  
 print('iteration %d / %d: loss %f' % (i, num\_iters, loss))  
  
 return train\_loss  
  
 def predict(self, X):  
 y\_pred = np.zeros(X.shape[1])  
 scores = X.dot(self.w)  
 y\_pred = np.argmax(scores, axis=1)  
 y\_pred = self.le.inverse\_transform(y\_pred)  
 return y\_pred  
  
class LinearSVM(LinearClassifier):  
 def loss(self, X\_batch, y\_batch, reg):  
 return lossFunction(self.theta, X\_batch, y\_batch, reg)  
  
a = LinearSVM()  
train\_loss = a.train(train\_x, train\_y, verbose=True)  
cv\_loss = LinearSVM().train(cv\_x, cv\_y)  
  
plt.figure('loss', figsize=(12, 10))  
plt.plot(train\_loss, label='train loss')  
plt.plot(cv\_loss, label='validation loss')  
plt.xlabel('iterations')  
plt.ylabel('loss')  
plt.legend()  
plt.show()

## 8. 选择的评估方法（留出法，交叉验证，k折交叉验证等）：交叉验证

## 9. 模型参数的初始化方法:全零初始化

## 10.选择的loss函数及其导数:least square loss

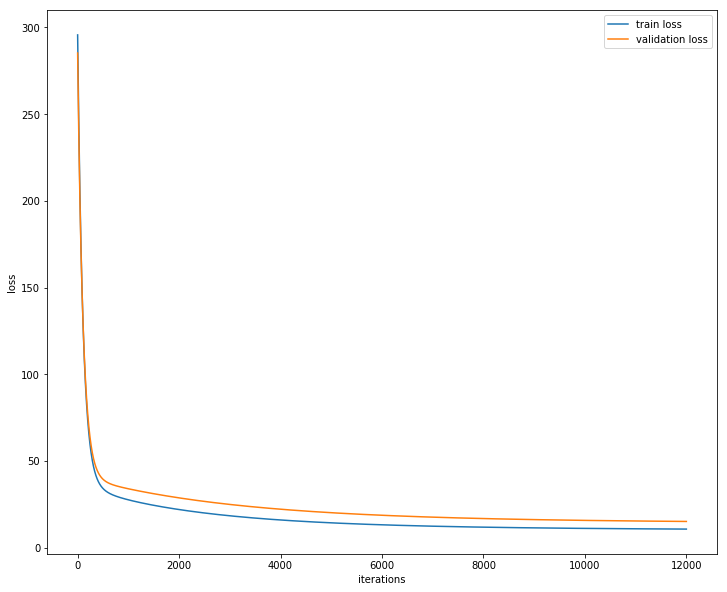
## 11.实验结果和曲线图:

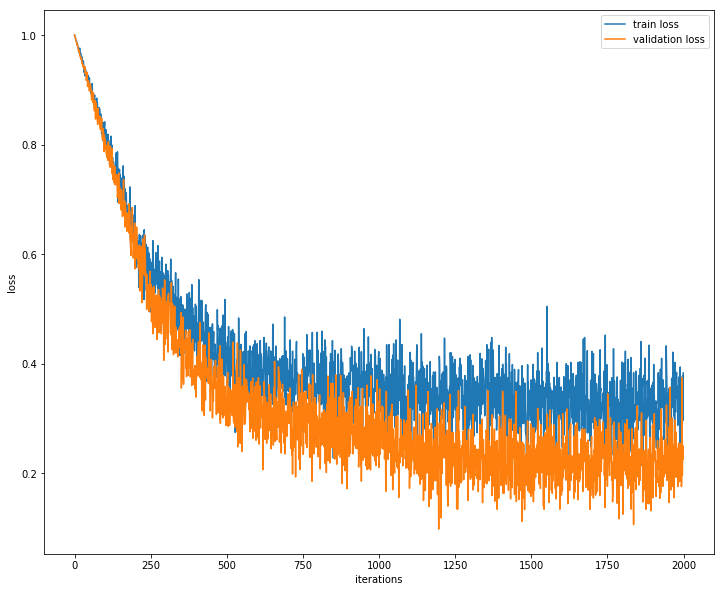
## 超参数选择（η,epoch等）：

## 评估结果（根据选择的评估方法）：

## 预测结果（最佳结果）：10.6473353545 0.279170

## loss曲线图：





## 12.实验结果分析:

## 13.对比线性回归和线性分类的异同点：分类问题是一定区域的一些x对应一个y，而回归问题是很小区域内的x， 对应着一个y

## 14.实验总结：

懂得原理但是不知道怎么用代码去展示出来是最大的问题，而且真的困扰了很久，现在也是一知半解，磨蹭了很久才弄出来。