

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**   **陈秋伶**

**学 号 201530611210**

**邮 箱 1900363950@qq.com**

**指导教师**  **吴庆耀**

**提交日期** **2017年 12月 8 日**

## 实验题目:

线性回归、线性分类与梯度下降

## 2. 实验时间：2017年12 月 2 日

## 3. 报告人:陈秋伶

## 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](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/)中的[Housing](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html#housing)数据，包含506个样本，每个样本有13个属性。线性分类使用的是[LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/)中的[australian](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#australian)数据，包含690个样本.

## 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" \t "_blank)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G。

6)取梯度G的负方向，记为D。

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" \t "_blank)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：

**def** loss\_fn(x, y, theta):  
 m = len(y)  
 **return** np.sum((x.dot(theta) - y) \*\* 2) / (2 \* m)

**def** GradientDescent(x, y, theta, alpha, m, maxIterations):  
 xTrains = x.transpose()*#取x的转置矩阵* loss\_iteration = [0] \* maxIterations*#迭代loss* **for** i **in** range(0, maxIterations):  
 hypothesis = x.dot(theta)  
 loss =hypothesis - y  
 gradient = xTrains.dot(loss)/m  
 loss = loss\_fn(x, y, theta)  
 loss\_iteration[i] = loss  
 theta = theta - alpha \* gradient  
 **return** theta,loss\_iteration

线性分类：

def loss\_fn(w, x, y, reg):#定义loss函数

num\_train = x.shape[0]

num\_classes = w.shape[1]

scores = x.dot(w)

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(w \* w)

coeff\_mat = np.zeros((num\_train, num\_classes))

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 \* w

return loss, dw

class LinearClassifier:#定义线性分类模型

def \_\_init\_\_(self):

self.w = None

self.le = LabelEncoder()

def train(self, X, y, learning\_rate=0.005, reg=1e-5, num\_iters=2000, batch\_size=200):

num\_train, dim = X.shape

y = self.le.fit\_transform(y)

num\_classes = int(np.max(y)) + 1

if self.w is None:

self.w = 0.001 \* np.random.randn(dim, num\_classes)

loss\_history = []

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)

loss\_history.append(loss)

self.w += -1 \* learning\_rate \* grad

return loss\_history

class LinearSVM(LinearClassifier):#支持向量机

def loss(self, X\_batch, y\_batch, reg):

return loss\_fn(self.w, X\_batch, y\_batch, reg)

## 8. 选择的评估方法（留出法，交叉验证，k折交叉验证等）:

## 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:

线性回归：

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

gradient = xTrains.dot(loss)/m（xTrains为X的转置矩阵）  
 线性分类：

loss = np.sum(margins) / num\_train + 0.5 \* reg \* np.sum(w \* w)

dw = dw / num\_train + reg \* w

## 11.实验结果和曲线图:

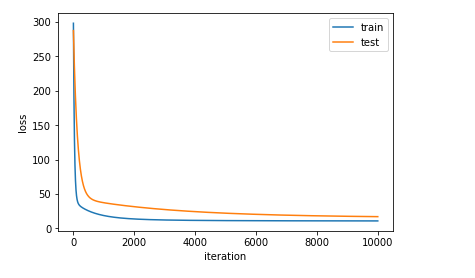
## 超参数选择（η,epoch等）：

线性回归：η=0.005

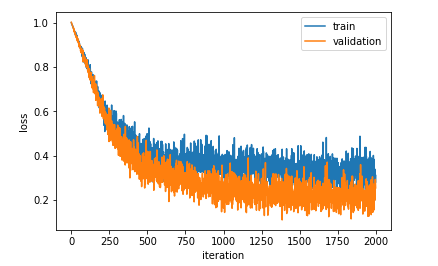
线性分类：η=0.005 epoch=0.00001

## 评估结果（根据选择的评估方法）：

线性回归

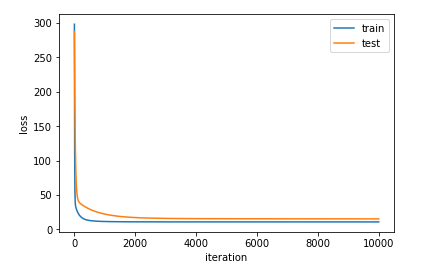


线性分类：

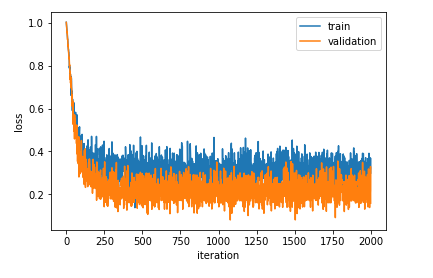


## 预测结果（最佳结果）：

线性回归：

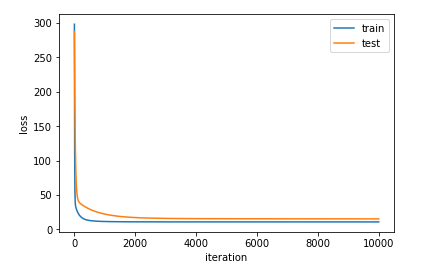


线性分类：

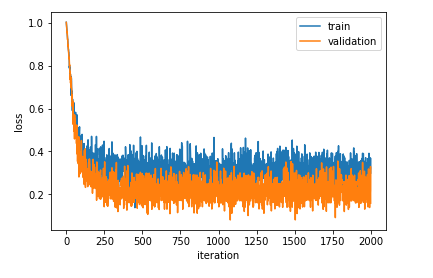


## loss曲线图：

线性回归



线性分类



## 12.实验结果分析:

## 13.对比线性回归和线性分类的异同点：

线性回归模型更倾向于 (很小区域内的x，或者一般是一个x)  对应着  (一个y)，它的loss函数比较平滑。

线性分类的y值(也称为label), 更离散化一些. 而且， 同一个y值可能对应着一大批的x,  这些x是具有一定范围的，所以线性分类问题更多的是 (一定区域的一些x) 对应着 (一个y)，它的loss函数比较在一定范围内会有所波动。

## 14.实验总结：

从这次实现中，我初步掌握了线性分类和线性回归模型的实现原理以及方法，掌握了python和相关工具的使用方法，同时复习到了大一所学的线性代数和数学分析感觉收获很多。同时意识到即使是同一个模型，通过适当调参也会获得

截然不同的结果，达到最优化的结果需要多次反复对模型进行调整。