

机器学习实验报告

学院: 软件学院

专业: 软件工程

|  |  |
| --- | --- |
| *作者:*  周治平 | *指导教师:*  吴庆耀 |
|  |  |
| *学号:*  202030483178 | *班级:*  20级软件工程1班 |

2020-10-24

# 逻辑回归和支持向量机

***摘要—简短的摘要意在给予读者一个对实验的整体介绍。它应该简要而切题。***

## **介绍**

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

## **方法和理论**

1实验模型

本实验采用的是线性模型，其定义为



其中为权重矩阵，为自变量，Y为数据标签，b为偏置量，在本实验中，在X中增加一个特征，该特征数据初始化为1，作为偏置量的训练数据。因此模型可以简化为

2定义损失函数



3求线性回归的闭式以及梯度求解



4随机梯度下降

输入：



输出：

随机挑选一个样本为依据的损失函数对的导数。



5全梯度下降

输入：



输出：

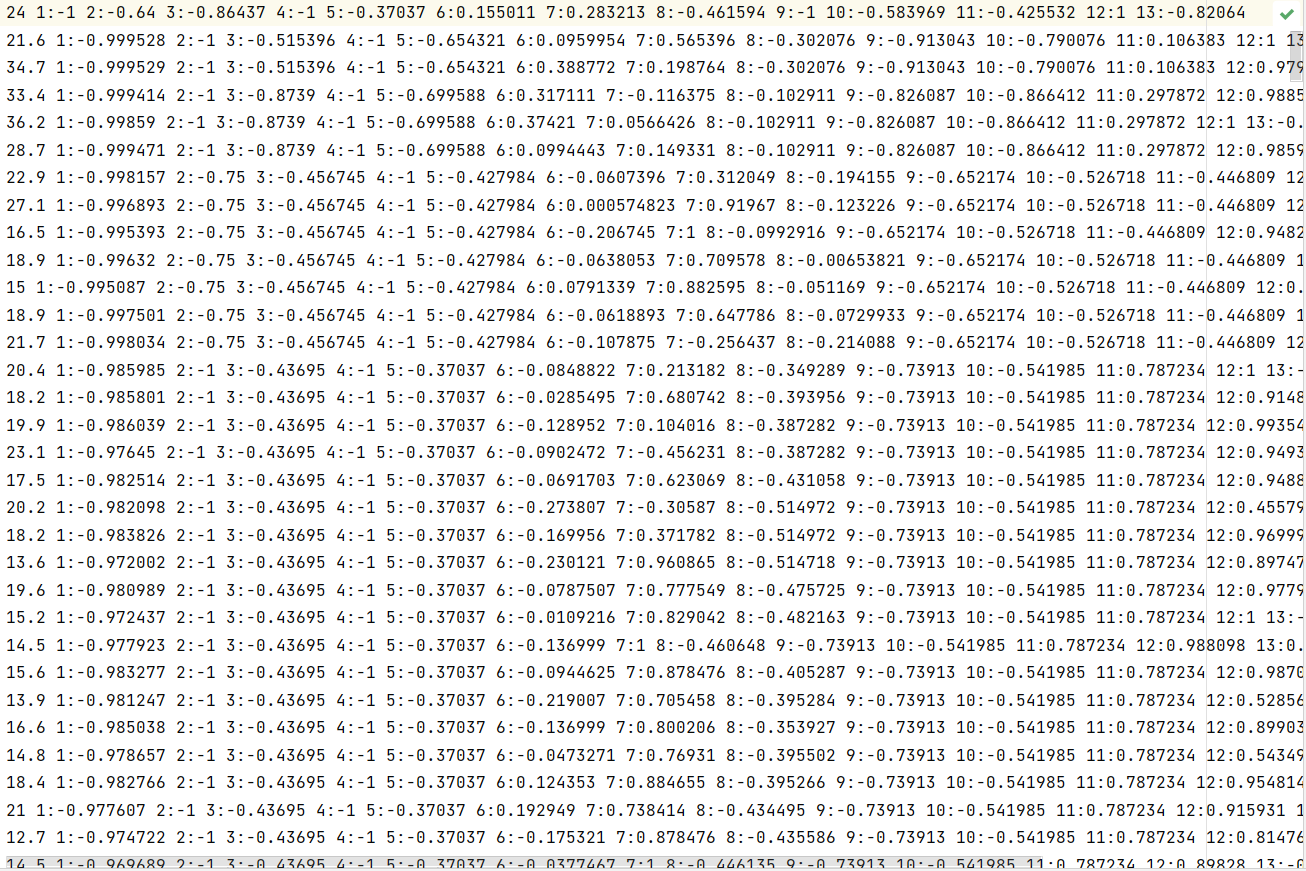


在本部分中，要求对实验进行完整的介绍。例如，选择的方法、相关理论、相关方程（如损失函数），推导过程（求解梯度）等等。

## **实验**

### 数据集

1数据内容



（Figure. 1.房产规模信息）

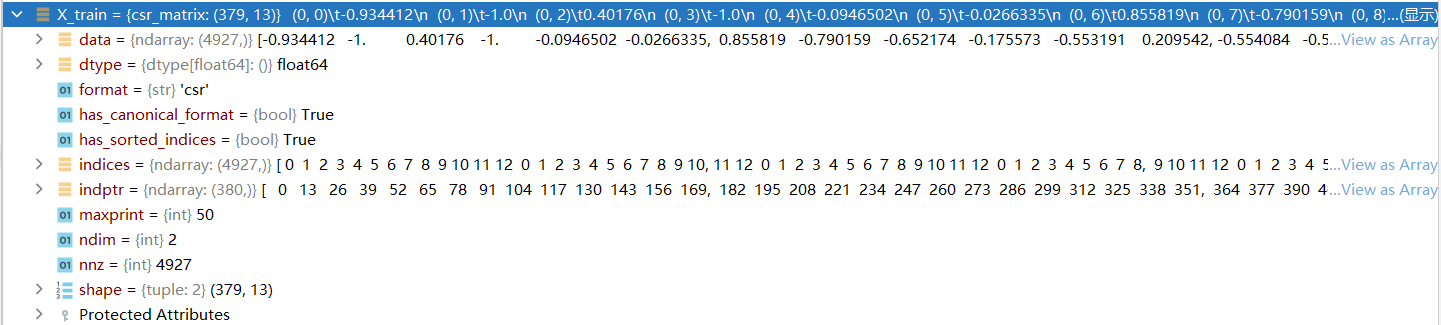
2数据数量

数据中包含着506组数据，其中包括1个数据标签和13个数据特征，因此数据量一共为7084个数据。

3训练集

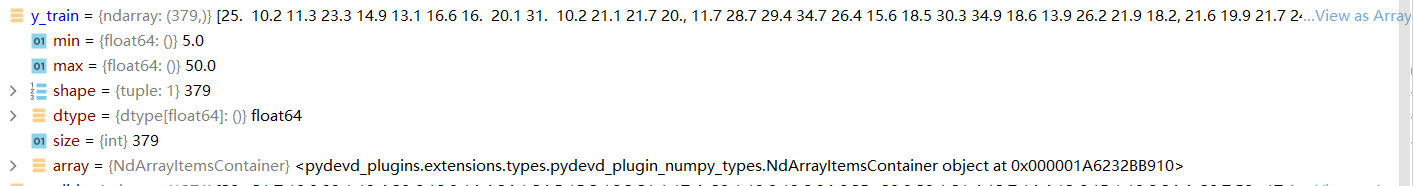
本实验中，程序将随机挑选75%组数据作为训练集，25%的数据作为验证集，最终可得训练数据如下：

数据特征：



（Figure. 2.训练数据集中的数据特征）

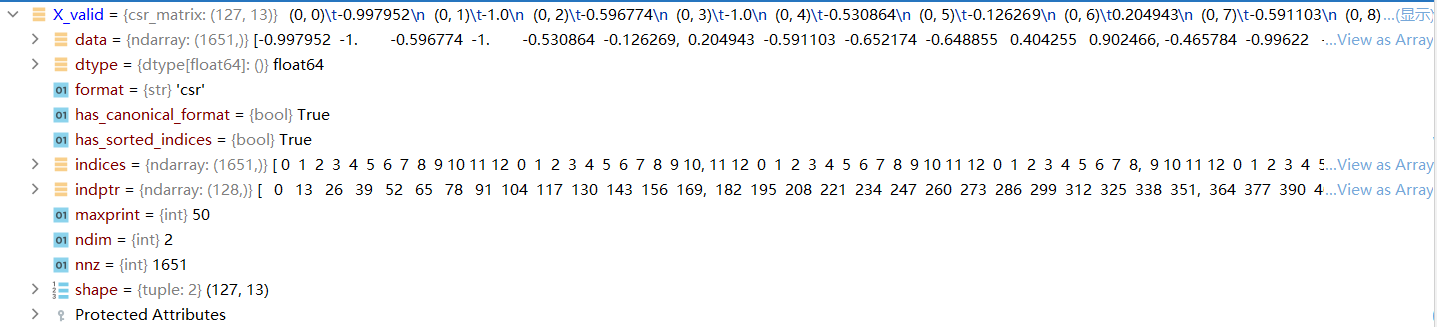
数据标签：



（Figure. 3.训练数据集中的数据标签）

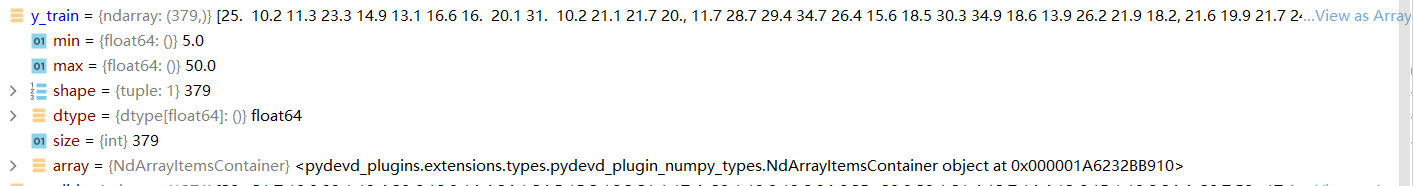
4验证集

数据特征：



（Figure. 4. 验证数据集中的数据特征）

数据标签：



（Figure. 5. 验证数据集中的数据标签）

此部分展示数据集的相关信息，例如内容、数据数量、训练集、验证集等。

### 实现

实验中的所有详细实现：初始化，处理过程，结果，各种参数。一言以蔽之，清楚地描述做了什么工作和工作方式。图和表应进行标记和编号，如表I和图1所示。

1初始化过程

1.1导入数据集

X, y = sd.load\_svmlight\_file(**'housing\_scale.txt'**,n\_features = 13)

1.2划分训练集与验证集

*# 将稀疏矩阵转为ndarray类型*X\_train = X\_train.toarray()  
X\_valid = X\_valid.toarray()  
y\_train = y\_train.reshape(len(y\_train),1)  
y\_valid = y\_valid.reshape(len(y\_valid),1)*#转化为1列*

1.3定义偏置量特征

*#让自变量添加一列，作为偏置量，即b，然后统一处理*X\_train = np.concatenate((np.ones((X\_train.shape[0],1)), X\_train), axis = 1)  
X\_valid = np.concatenate((np.ones((X\_valid.shape[0],1)), X\_valid), axis = 1)

1.4定义损失函数

*# 选取一个Loss函数，计算训练集的Loss函数值，记为loss*def compute\_loss(X, y, theta):  
 hx = X.dot(theta)*#w点乘X(矩阵乘法)* error = np.power((hx - y), 2).mean()  
 return error

1.5定义线性回归闭式解求解函数

def normal\_equation(X, y):  
 return (np.linalg.inv(X.T.dot(X))).dot(X.T).dot(y)

1.6初始化模型参数

theta = np.zeros((14, 1))

1.7定义梯度求解函数

def gradient(X, y, theta):  
 *#np.dot(X,theta)* return X.T.dot(X.dot(theta) - y)

1.8初始化学习率与迭代次数

alpha = 0.001  
iters = 30

2处理过程

2.1闭式解求解过程

theta = normal\_equation(X\_train, y\_train)  
loss\_train = compute\_loss(X\_train, y\_train, theta)  
loss\_valid = compute\_loss(X\_valid, y\_valid, theta)

2.2随机梯度下降求解过程

*#随机梯度下降  
# @X 训练集自变量  
# @Y 训练集因变量  
# @theta 模型权重系数  
# @alpha 学习率  
# @iters 迭代次数  
# @X\_valid 验证集自变量  
# @y\_valid 验证集因变量*def random\_descent(X, y, theta, alpha, iters, X\_valid, y\_valid):  
 *# n个特征* n=X.shape*#行为样本，列为特征* loss\_train = np.zeros((iters,1))*#初始化训练集损失数组* loss\_valid = np.zeros((iters,1))*#初始化验证集损失数组* for i in range(iters):  
 *#随机选择一个样本* num=np.random.randint(n,size=1)*#选择一个样本，包含所有特征* x\_select=X[num,:]*#获取该样本的所有特征值* y\_select=y[num,0]*#获取该样本的标签* grad = gradient(x\_select, y\_select, theta)*#进行梯度计算* theta = theta - alpha \* grad*#模型参数优化* loss\_train[i] = compute\_loss(X, y, theta)*#记录训练集中每一次迭代的损失* loss\_valid[i] = compute\_loss(X\_valid, y\_valid, theta)*#记录验证集中每一次迭代的损失* return theta, loss\_train, loss\_valid

opt\_theta, loss\_train, loss\_valid = random\_descent(X\_train, y\_train, theta, alpha, iters, X\_valid, y\_valid)  
*#选取矩阵中最小的值*print(loss\_train.min())  
print(loss\_valid.min())

2.3全梯度下降求解过程

*#全梯度下降  
#随机梯度下降  
# @X 训练集自变量  
# @Y 训练集因变量  
# @theta 模型权重系数  
# @alpha 学习率  
# @iters 迭代次数  
# @X\_valid 验证集自变量  
# @y\_valid 验证集因变量*def descent(X, y, theta, alpha, iters, X\_valid, y\_valid):  
 loss\_train = np.zeros((iters,1))*#初始化训练集损失数组* loss\_valid = np.zeros((iters,1))*#初始化验证集损失数组* for i in range(iters):*#全部样本都进行一次迭代* grad = gradient(X, y, theta)*#对每一个样本都进行梯度计算* theta = theta - alpha \* grad*#参数优化* loss\_train[i] = compute\_loss(X, y, theta)*#记录训练集中每一次迭代的损失* loss\_valid[i] = compute\_loss(X\_valid, y\_valid, theta)*#记录验证集中每一次迭代的损失* return theta, loss\_train, loss\_valid

opt\_theta, loss\_train, loss\_valid = descent(X\_train, y\_train, theta, alpha, iters, X\_valid, y\_valid)  
*#选取矩阵中最小的值*print(loss\_train.min())  
print(loss\_valid.min())

3实验结果绘图

3.1随机梯度下降

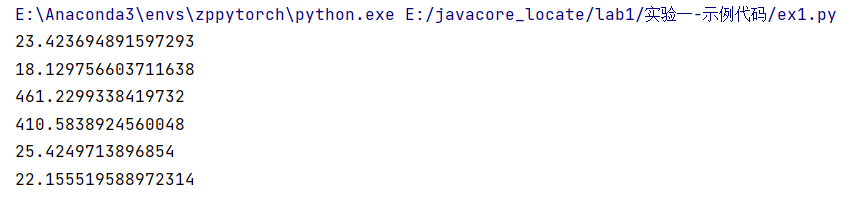
iteration = np.arange(0, iters, step = 1)  
fig, ax = plt.subplots(figsize = (12,8))  
ax.set\_title(**'Train'**)  
ax.set\_xlabel(**'iteration'**)  
ax.set\_ylabel(**'loss'**)  
plt.plot(iteration, loss\_train, **'b'**, label=**'Train'**)  
*# plt.plot(iteration, loss\_valid, 'r', label='Valid')*plt.legend()  
plt.show()

3.2全批量梯度下降

iteration = np.arange(0, iters, step = 1)  
fig, ax = plt.subplots(figsize = (12,8))  
ax.set\_title(**'Train'**)  
ax.set\_xlabel(**'iteration'**)  
ax.set\_ylabel(**'loss'**)  
plt.plot(iteration, loss\_train, **'b'**, label=**'Train'**)  
*# plt.plot(iteration, loss\_valid, 'r', label='Valid')*plt.legend()  
plt.show()

4实验结果截图

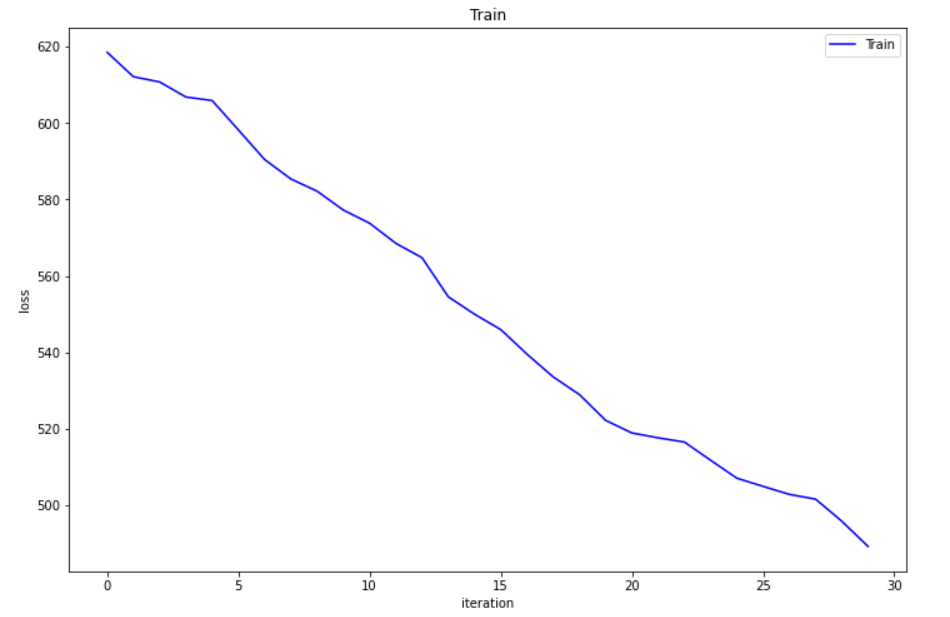
4.1最小损失结果



（Figure. 6. 数据集中的最小损失）

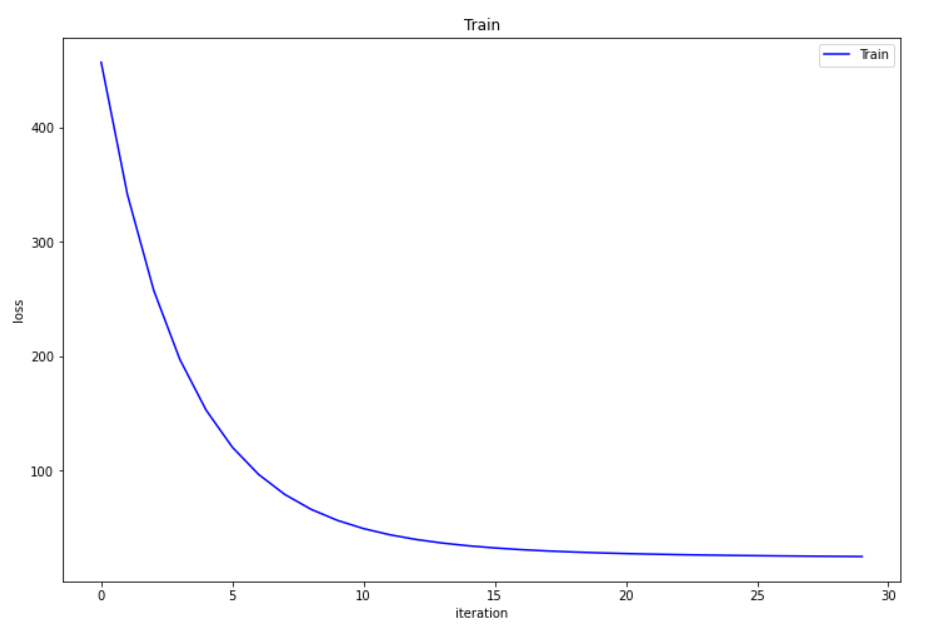
（tips：第一行为线性回归笔试解训练集的最小损失，第二行为线性回归笔试解验证集的最小损失，第三行为随机梯度下降训练集的最小损失，第四行为随机梯度下降训练验证集的损失，第五行为全批量梯度下降训练集的损失，第六行为全批量梯度下降验证集的损失）

4.2随机梯度下降训练结果



（Figure. 7. 随机梯度下降训练集损失变化）

4.3全批量梯度下降训练结果



（Figure. 8. 全批量梯度下降训练集损失变化）

## **结论**

本节总结了实验报告。也可以在此处写下在实验中的收获和灵感。

在本次实验中，主要实现线性回归模型，其主要步骤是先导入数据，划分训练集与验证集，然后定义线性模型，由于模型中有14个特征（13个特征+1个偏置量），因此定义权重系数有14个，然后定义损失函数，然后分别通过线性回归闭式解，随机梯度下降训练以及全批量梯度下降训练的方式进行训练，进行参数优化，线性回归闭式解则不需要进行参数优化，因为直接求解可以直接获取最优参数，而梯度下降法需要通过迭代进行参数，最后训练结果输出与绘图就完成了。

对于本次实验而言，主要有代码参考，加上线性模型结构比较简单，其闭式解与梯度函数的公式比较容易理解，但更多是了解如何去进行模型的训练。以及真正的代码实现，这与在课本上仅学习到的公式是相差很大的。此外还学习到了许多numpy的库，让自己对numpy库中的函数有了更进一步的了解与运用。总体而言本次实验的收获是挺大的。