深 圳 大 学 实 验 报 告

课程名称 机器学习

项目名称 实验三：线性回归算法

学　　院 计算机与软件学院

专　　业 软件工程

指导教师 赖志辉

报 告 人 郑杨 学号 2020151002

实验时间 2022 年 4 月 4 日至 2022 年 4 月 7 日

实验报告提交时间 2022 年 4 月 7 日

教务处制

一、实验目的与要求

1. 实现基本的线性回归算法，对一个简单的数据进行预测
2. 参考相关论文与文献，实现 3-4 个现有论文中的回归算法，并比较其在人脸识别中的性能
3. 自行设计一个全新的线性回归算法（不是别人论文里的！而是自己创造的！），包括建模与优化，收敛性证明等（如果有），要求：你开发的新算法能在人脸识别实验中的识别率能比过基本的或现有的线性回归算法（至少在 2~3 个数据库中比较好，另 1~2 个中基本差不多）。 全方位比较你的方法与你复现的方法在不同维数的识别率。

二、实验内容与方法

1.实现基本的线性回归算法，简述线性回归算法的思想和优化证明。使用 matlab 实现程序并对一个简单的数据进行预测。
2.参考了 2 篇论文，复现了论文中的回归算法，与 PCA、LDA 比较了在人脸识别中的性能。

三、实验步骤与过程

1.实现基本的线性回归算法

（1）算法思想

线性回归模型是机器学习中最简单、基础的一类有监督学习模型。它要处理的一类问题是：给定一组输入样本和每个样本对应的目标值，在某一损失准则下，找到目标值和输入值的函数关系，当有一个新的样本到达时，可以预测其对应的目标值时多少。

其优点是，结果具有较好的可解释性；缺点是对于非线性数据的拟合效果不好。

（2）算法原理与模型推导

由一个具体的例子说起，我们使用工资和年龄预测银行贷款的额度，那么有两个特征：工资和年龄，目标是银行贷款的额度，如下表：

工资	年龄	额度
4000	25	20000
8000	30	70000
5000	28	35000
7500	33	50000
12000	40	85000

我们用x_1表示工资，x_2表示年龄，y表示额度，那么线性回归模型的目标就是通过$x_1 x_2$的线性组合去预测y。假设θ_1为x_1的参数，θ_2为x_2的参数，那么对y的预测函数（通常称为假设函数$hypothesis\ function$）为：

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

其中：θ_0 为偏置项，我们可以让 $x_0=1$，则：

$$h_\theta(x)=\theta_0x_0+\theta_1x_1+\theta_2x_2$$

整合一下就是：$h_\theta(x)=\sum_{i=0}^{2}\theta_ix_i=\theta^Tx$

推广到有 n 个特征的情况下，就是：

$$h_\theta(x)=\sum_{i=0}^{n}\theta_ix_i=\theta^Tx$$

接下来，我们来分析一下这个假设函数的误差，就是对于每一个样本而言，预测值和真实值都会有或多或少的差异，记 $y^{(i)}$ 为第 i 个样本的目标值，$x^{(i)}$ 为第 i 个样本所有特征组成的列向量，ε 为真实值与预测值之间的误差，那么：

$$\varepsilon=y^{(i)}-h_\theta(x^{(i)})=y^{(i)}-\theta^Tx^{(i)}$$

想要模型预测性能最好，就是让所有样本的预测误差尽可能小，我们定义损失函数为：

$$J(\theta)=\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2$$

可以用极大似然估计求出这个损失函数，由于篇幅问题这里就不作详细证明了。
接下来最小化损失函数的方法有**梯度下降和求导法**，这里讲一下求导法。
首先先把损失函数写成矩阵形式：

$$J(\theta)=\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2=\frac{1}{2}(X\theta-y)^T(X\theta-y)$$

把损失函数 $J(\theta)$ 对 θ 求偏导可以得到：

$$\frac{\partial J(\theta)}{\partial\theta}=\frac{\partial(\frac{1}{2}(X\theta-y)^T(X\theta-y))}{\partial\theta}=\frac{\partial(\frac{1}{2}(\theta^TX^T-y^T)(X\theta-y))}{\partial\theta}$$

$$=\frac{\partial(\frac{1}{2}(\theta^TX^TX\theta-\theta^TX^Ty-y^TX\theta+y^Ty))}{\partial\theta}=\frac{1}{2}(2X^TX\theta-X^Ty-(y^TX)^T)$$

$$=X^TX\theta-X^Ty$$

令偏导为 0，得：

$$\theta=(X^TX)^{-1}X^Ty$$

故，对于某一个测试样本 t 的预测值为：

$$\hat{t}=\theta^Tt$$

（3）代码实现并预测一个简单的数据

代码实现上比较简单，根据$\theta=(X^TX)^{-1}X^Ty$直接返回θ的结果即可。

```
function theta = regression(X, y)
    % Inputs:
    %       -X: training set, [data_num x dimension]
    %       -y: real value of training set, [data_num x 1]
    % Outputs:
    %       -theta: the parameter vector, [dimension x 1]
    % Author: Yang Zheng
    % Date: 2022. 4. 4

    theta = (X' * X)^-1 * X' * y;
end
```

该简单数据有 47 行，每一行有三个字段，我们把前两个字段作为特征，最后一个字段作为结果，数据如下所示：

```
2104,3,399900
1600,3,329900
2400,3,369000
1416,2,232000
3000,4,539900
1985,4,299900
1534,3,314900
1427,3,198999
1380,3,212000
1494,3,242500
```

把前 35 行作为训练集，后 12 行作为测试集进行预测，测试预测效果：

预测值	真实值	误差
261100	249900	4.48%
231160	229900	0.55%
365820	345000	6.03%
680740	549000	24.00%
370350	287000	29.04%
295770	368500	-19.74%
382200	329900	15.85%
431590	314000	37.45%
225260	299000	-24.66%
173000	179900	-3.84%
323490	299900	7.87%
225720	239500	-5.75%
平均误差		6.48%

可以看到误差在可控制范围内。

2.参考论文与资料并复现算法，比较识别率

（1）Linear Regression for Face Recognition

论文算法简述：

论文首先提出了 LRC 算法，利用线性回归进行人脸识别。论文提出，特征空间的选择可能不再那么关键，可以使用下采样方法把原始图像的维度缩小。然后基于线性子空间的概念提出每一类的训练样本都可以生成一个线性子空间，待预测样本属于这一类当且仅当可以被这一类的训练样本的线性组合表示。

假设当前有 N 个类，每一类有 p_i 个样本，每个样本为 $a\times b$ 的图像，首先下采样成一个 $c\times d$ 的图像，再把每一个图像矩阵转化成 $q\times 1$ 的列向量，$q=cd<ab$，再把每一个

列向量归一化（把向量的模长变为 1）。记下采样后第i类的第 j 个样本为w_{ij}，第i类的样本矩阵为$X_i=[w_{i1},w_{i2},...,w_{ipi}](q\times p_i)$，对于某一个测试样本$y$，用线性回归的思想，用第$i$类的测试样本去拟合$y$可以得到一个参数列向量$\beta_i=(X_i^TX_i)^{-1}X_i^Ty$，拟合之后的结果为：

$$y=X_i\beta_i=X_i(X_i^TX_i)^{-1}X_i^Ty=H_iy$$

其中，$H_i=X_i(X_i^TX_i)^{-1}X_i^T$

那么对于某个一测试样本y，可以得到N个预测结果，令：

$$d_i(y)=\|y-y_i\|_2,i=1,2,...,N$$

我们的目标是：

$$\min_i d_i(y),i=1,2,...,N$$

另外，论文针对连续遮挡问题提出了图像的模块化线性回归算法 MLRC。该算法就是把所有图像划分为M个模块，对于每一个模块使用 LRC 算法找到测试样本该模块的最佳拟合类：

$$d_i(y^{(j)})=\|y^{(j)}-y_i^{(j)}\|_2,i=1,2,...,N$$

其中$d_i(y^{(j)})$表示用第i类拟合y的第 j 个模块的误差，现在对于第 j 个模块，中间决策$c^{(j)}$表示拟合第 j 个模块最优的类，对应的误差为

$$d^{c^{(j)}}=\min_i d_i(y^{(j)}),i=1,2,...,N$$

在这之后，我们得到了M决策$c^{(j)}$与M个对应的误差$d^{c^{(j)}}$，论文中的做法是，取出M个误差中最小的那一个所对应的类作为预测结果，即：

$$Decision=\arg\min_c d^{c(j)},j=1,2,...,M$$

算法识别率比较：

测试了 **LRC 算法**，我使用了 ORL 数据集和 FERET 数据集进行比较测试：

i. ORL 数据集

根据论文里的方法，把原始图像下采样成 10×5 的图像。（LDA 降维到 39 维、PCA 降维到 50 维，这两个维度就是自己定的，论文里没说明）

然后进行两种测试，第一种是采用每一类人物的前五张图片作为训练集，后五张作为测试集；第二种是使用留一法（leave-one-out）进行测试，测试结果如下：

TABLE 1

Results for EP1 and EP2 Using the ORL Database

Evaluation Protocol	Approach	Recognition Rate
EP1	PCA	80.50%
	LDA	72.50%
	LRC	**84.50%**
EP2	PCA	89.25%
	LDA	95.00%
	LRC	**95.50%**

ii. FERET 数据集

使用 100 类人物图像，将每类图像的前五张图像作为训练集，后两张作为测试集进行测试，分别测试维度为 40，60，80，100，120 时三个算法的识别率（PCA、LCA 和 LRC），测试结果如下：

TABLE 2

Results for EP Using the FERET Database

Dataset	Dimension	Recognition Rate
PCA	40	79.50%
	60	80.50%
	80	81.50%
	100	81.00%
	120	81.50%
LDA	40	47.00%
	60	48.00%
	80	47.50%
	100	49.00%
	120	47.00%
LRC	40	**83.00%**
	60	**85.50%**
	80	**88.00%**
	100	**86.50%**
	120	**90.00%**

MLRC 算法实现与识别率测试：

首先需要对所有图像进行模块划分，我在实现的时候把划分部分放在数据读入函数里，读入数据之后顺便进行划分，然后返回 M 个模块的样本矩阵。然后对 M 个模块的样本矩阵做统一处理。

在 AR 数据集上与 LRC 算法进行比较，分别取前 10~20 张人脸做训练集，比较识别率：

	Training set size								
Algorithm	10	11	12	13	14	15	16	17	18
LRC	0.3919	0.5007	0.4657	0.44	0.6233	0.6255	0.641	0.6367	0.6062
MLRC	0.3881	0.4567	0.43	0.39	0.5667	0.5882	0.589	0.5778	0.5713

（2）Ridge Regression: Biased Estimation for Nonorthogonal Problems

算法思想简述：

论文提出了解决病态矩阵的正则化线性回归方法：Ridge Regression，通过给代价函数添加一个正则化项（L2 正则化）：

$$J(\theta)=\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{i=1}^{n}\theta_i^2$$

把它写成矩阵形式为：

$$J(\theta)=\frac{1}{2}(X\theta-y)^T(X\theta-y)+\lambda\theta^T\theta$$
$$=\frac{1}{2}(\theta^T X^T X\theta-\theta^T X^T y-y^T X\theta+y^T y)+\lambda\theta^T\theta$$

对θ求偏导并令倒数为 0 得：

$$\frac{\partial J(\theta)}{\partial\theta}=X^T X\theta-X^T y+\lambda\theta=0$$

故可得：$\theta=(X^T X+\lambda I)^{-1}X^T y$

对于λ的选择，一般有一下两个指标：
各回归系数的岭估计基本稳定；残差平方和增大不太多。

算法实现：

代码实现上还是比较简单的，就是加上一个正则项而已。

```
n = size(X, 2);
I = eye(n);
I(1, 1) = 0;
theta = (X' * X + lambda * I)^-1 * X' * y;
```

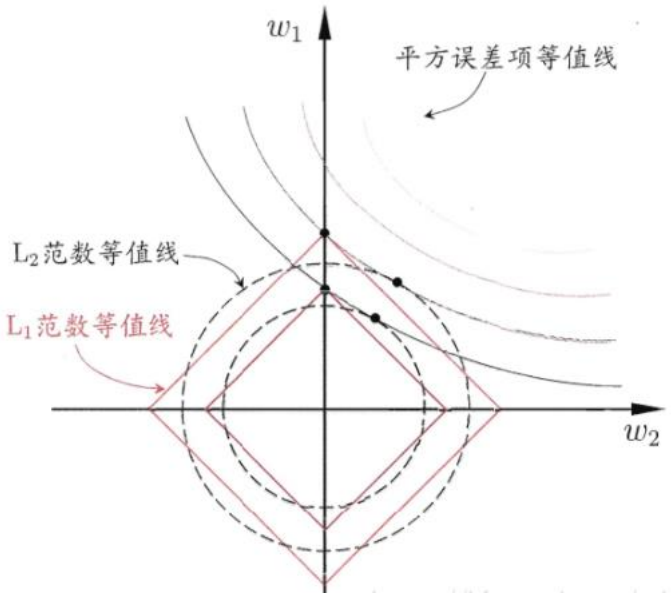
（3）Lasso Regression

算法思想简述：

与岭回归不同的是，Lasso 回归加上的正则项为参数θ的L_1范数，即代价函数为：

$$J(\theta)=\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\sum_{i=1}^{n}\|\theta_i\|$$

也称为L_1正则化，这个正则化项比L_2正则化更容易得到稀疏解，具体的数学证明我看不太懂，只能利用几何形式直观地理解。假设x只有两个属性，w只有两个参数w_1和w_2，绘制不带正则项的目标函数，平方误差等值线，再绘制L_1，L_2范数等值线，如图，正则化后优化目标的解要在平方误差项和正则项之间折中，即出现在图中等值线相交处。采用L_1范数时，交点更容易出现于坐标轴上，此时其中一个参数为 0，就是更容易获得稀疏解。



对于Lasso回归优化目标的推导，由于该函数存在不可导点，不能通过求偏导进行优化，常见的优化方法有：坐标轴下降法和最小角回归法，都有点高深莫测目前都没看懂。

四、实验结论或体会

这次实验参考了比较多的论文，复现了两篇论文的算法，对于回归问题有了基本的了解，对于正则化方法有了初步的了解。但由于时间问题，只是初步了解了正则化的用法，知道了正则化是用来处理过拟合问题，没有过于深入地了解其证明以及几何含义。由于看论文与复现实验使用了大量的时间，对于股票预测没有深入的进行建模与分析。

五、思考题

机器学习之股票价格预测大 PK----- 论从即日起到 6 月 30 日上午收盘涨幅最大的锂电池或煤炭相关行业股票

由于时间不足，就只是简单学习了使用 sklearn 机器学习库对 GOOGLE 的一支股票做出预测，没有对 6 月 30 日上午收盘涨幅最大的锂电池或煤炭相关行业股票进行预测。

首先使用 quandl 获取互联网上的数据：

```
df = quandl.get("WIKI/GOOGL")
```

使用股票的开盘价，最高价，最低价和交易量对收盘价进行预测：

```
df = df[['Open', 'High', 'Low', 'Close', 'Volume']]
```

设置预测数量并预处理数据：

```
predict_count = 30
df['label'] = df['Close'].shift(-predict_count)
X = df.drop(['label'], axis = 1)
y = df['label'][:-predict_count]
```

将数据标准化：

```
scale = StandardScaler()
scale.fit(X)
X = scale.transform(X)
```

划分训练集和测试集：

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1)
```

使用线性回归库对数据进行拟合，并对测试数据进行预测：

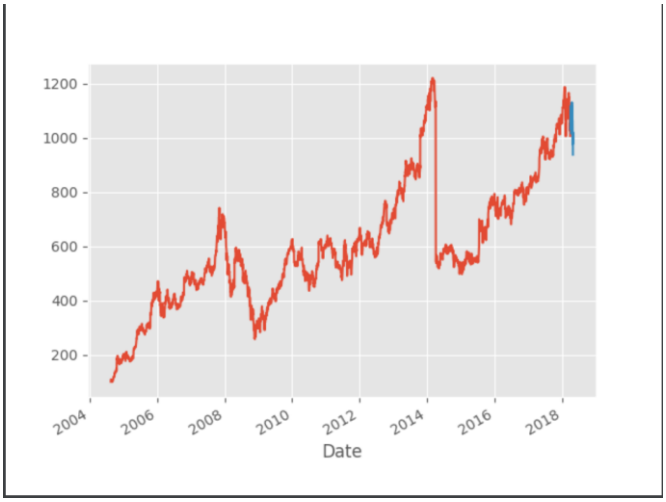
```
model = LinearRegression()
model.fit(X_train, y_train)
predict = model.predict(X_lately)
```

画出原始图像与预测部分图像：

```
df['predict'] = np.nan
last_date_st = df.index[-1].timestamp()
next_date_st = last_date_st + 86400
for i in predict:
    next_date = datetime.datetime.fromtimestamp(next_date_st)
    df.loc[next_date] = [np.nan for _ in range(len(df.columns) - 1)] + [i]
    next_date_st += 86400

style.use('ggplot')
df['Close'].plot()
df['predict'].plot()
plt.show()
```

预测图像如下：



指导教师批阅意见： 成绩评定： 指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。