Machine Learning

题目：**中山大学**

学习小组 05 机器学习作业

姓　　名	方桂安，刘玥，周敏
学　　号	20354027,20354229,20354187
院　　系	智能工程学院
专　　业	智能科学与技术
指导教师	彭卫文 (副教授)

2021 年 10 月 27 日

一、数据分析

1.1 数据缺失检查

首先，为了我们能正常进行数据分析，我们进行了数据缺失分布情况检查。代码及结果如下：

```
#checking the null values in each column
traindata.isnull()
```

✓ 0.4s

	TV	radio	newspaper	sales
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False
...	...	...	...	...
175	False	False	False	False
176	False	False	False	False
177	False	False	False	False
178	False	False	False	False
179	False	False	False	False

180 rows × 5 columns

缺失值总数：

```
#checking the null values in each column
traindata.isnull().sum()
```

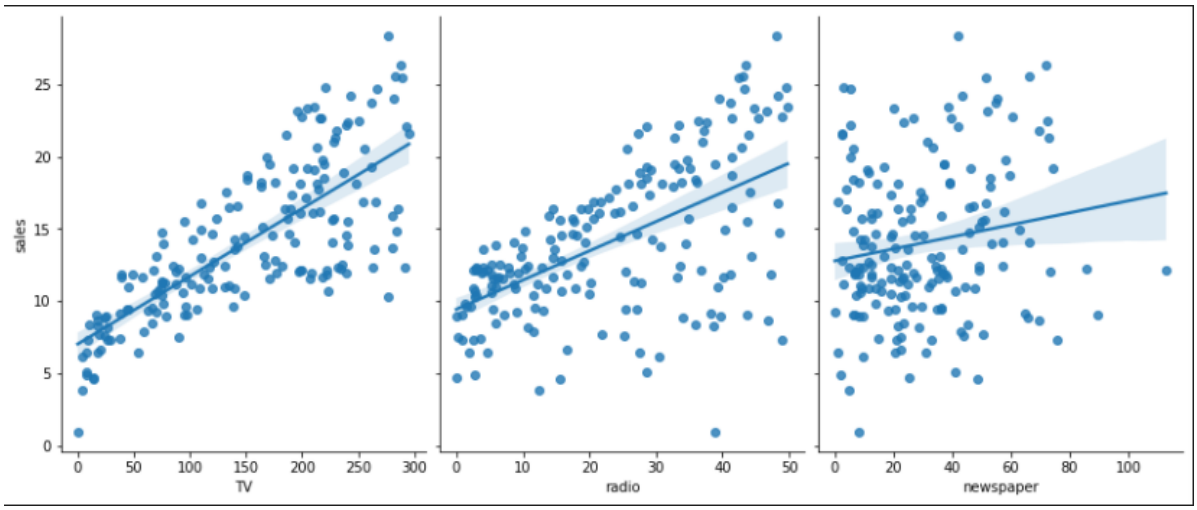
✓ 0.3s

```
             0
TV           0
radio        0
newspaper    0
sales        0
dtype: int64
```

由上可知，我们的数据中没有缺失值，不需要进行插值处理。

1.2 销售量与各媒体投入关系分析

1.2.1 散点图



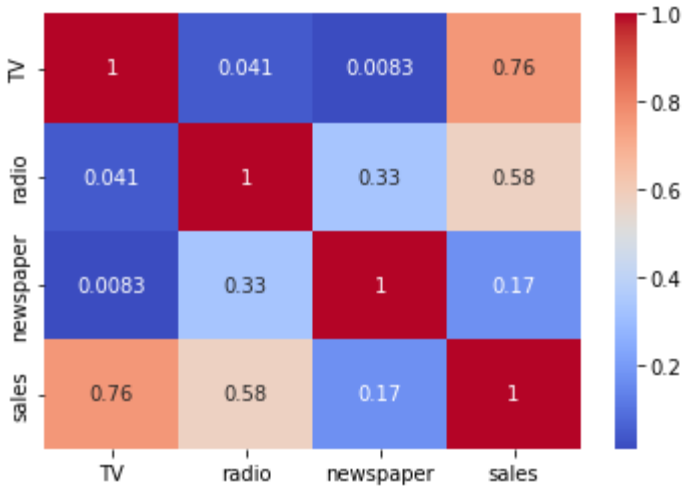
以上是销售量与各项媒体投入量的散点图。从上图我们可以看出，sales和TV投入量有明显的正相关关系，随着TV投入增多，sales大体上呈上升趋势。sales和radio投入量也有较弱的正相关趋势，但sales分布在以radio投入量为指标时，分布较零散，相关关系弱于sales与TV投入量。而sales和newspaper的相关性最弱，sales集中分布在newspaper低投入区域内。

1.2.2 各项数据分析

	TV	radio	newspaper	sales
count	180.000000	180.000000	180.000000	180.000000
mean	149.370813	22.729342	29.959906	14.028374
std	84.738353	14.881243	21.286009	5.226817
min	0.133171	0.000000	0.000000	0.998363
25%	75.406788	9.018985	12.280162	10.661550
50%	157.251734	21.555876	25.317784	12.627705
75%	219.515698	35.937680	43.465301	17.611063
max	295.198299	49.765164	112.927996	28.321386

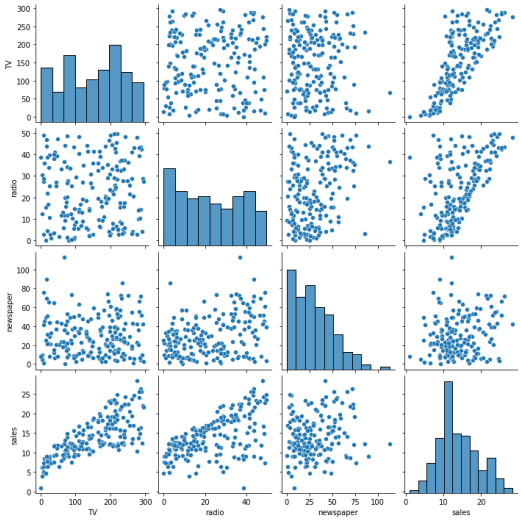
由上图可知，TV类广告的平均投入量最大，其投入量的最小值，二分位数，中位数和四分位数，最大值均大于其他类型的广告，说明企业偏向于在TV类广告投入更多资金。

1.2.3 相关系数



上图为四个变量的相关系数热力图，由此可以看出，销售量和TV，radio，newspaper的相关性依次减弱。

1.2.4 散点图矩阵，多变量之间的关系可视化



1.3 得出结论

由上面的分析可知，销售量和TV投入量相关性最大，其次是radio，newspaper，这也符合我们目前的社会情况，人们更多的是在电视等电子产品上获取信息。所以，加大上述三种广告方式的投入会对销售量有依次递减的增幅影响。

二、描述10折交叉验证对数据集的处理

2.1 引入10折交叉验证的原因

泛化能力是指模型在训练集上训练后,对新数据进行准确预测的能力。在机器学习的模型选择中，我们要对候选模型的泛化误差进行评估，然后选择泛化误差最小的那个模型。而实际应用中，我们无法直接获得泛化误差，而训练误差又由于过拟合现象的存在而不适合作为标准，所以我们随机将数据集切为三部分：

- 训练集：用来训练模型，对应训练误差。
- 验证集：用来选择模型，对应测试误差。
- 测试集：用来最终对学习方法进行评估，对应泛化误差的近似。

但是在实际应用中数据往往是不充足的，为了选择泛化能力更好的模型，我们可以对数据集D进行适当的处理，从中产生出训练集S和测试集T。几种常见的做法有：简单交叉验证(holdout cross-validation)、留一交叉验证(leave-one-out cross-validation,LOOCV)、*k*折交叉验证(*k*-fold cross-validation)、多重*k*折交叉验证、分层法(stratification-split cross-validation)、自助法(bootstraps)等。而综合考虑几种方法的特点后，本次我们选择的处理方法是10折交叉验证法。

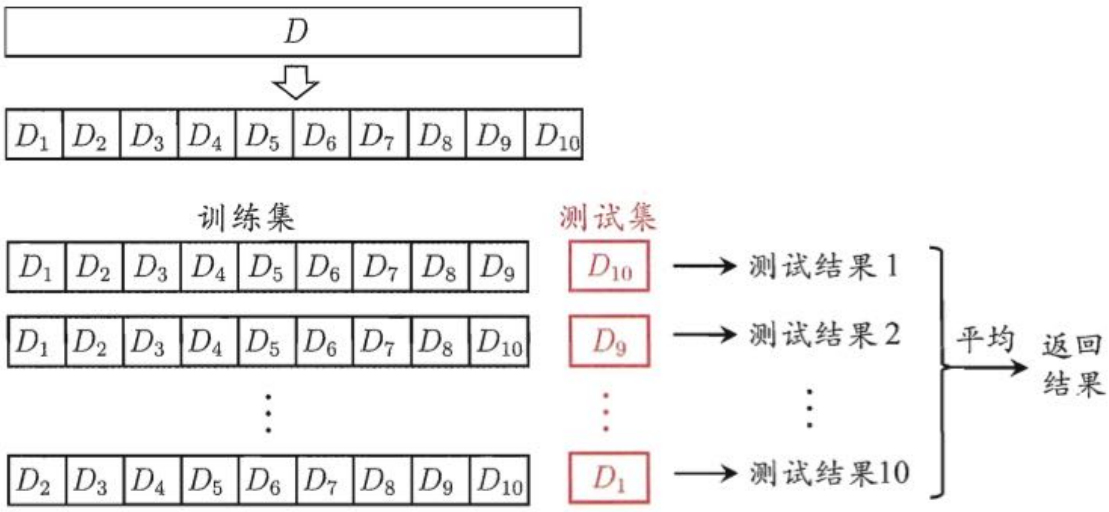
2.2 10折交叉验证的基本原理

10折交叉验证是指将原始数据集随机划分为样本数量近乎相等的10个子集，轮流将其中的9个合并作为训练集，其余1个作为测试集。在每次试验中计算正确率等评价指标，最终通过k次试验后取评价指标的平均值来评估该模型的泛化能力。

10折交叉验证的基本步骤如下:

1. 原始数据集划分为10个样本量尽可能均衡的子集;
2. 使用第1个子集作为测试集，第2～9个子集合并作为训练集;
3. 使用训练集对模型进行训练,计算多种评价指标在测试集下的结果;
4. 重复2-3步骤,轮流将第2-10个子集作为测试集;
5. 计算各评价指标的平均值作为最终结果，最终选出10次测评中平均测试误差最小的模型。

10折交叉验证的原理示意见下图。



由于将数据集D划分为k个子集同样存在多种划分方式，为了减小因样本划分不同而引入的差别，k折交叉验证通常要随机使用不同的划分重复p次。故我们可以采用“10次10折交叉验证”。

2.3 10折交叉验证函数python代码

```
from sklearn.model_selection import KFold  # 从sklearn导入KFold包
def Ten_Flod_spilt(fold,data,label):
    '''
    param fold: 要取第几折的数据。
    param data: 需要分块的数据
    param label: 对应的需要分块标签
    return: 对应折的训练集、测试集和对应的标签
    '''
    split_list = []
    kf = KFold(n_splits=10)
    for train, test in kf.split(data):
        split_list.append(train.tolist())
        split_list.append(test.tolist())
```

```
    train,test=split_list[2 * fold],split_list[2 * fold + 1]
    return  data[train], data[test], label[train], label[test]  #已经分好块的数据集
```

在后续使用中只需循环调用该函数即可达到10折交叉验证的目的：

```
for i in range(10):
        x_train, x_test, y_train, y_test = Ten_Flod_spilt(i,X_s,y_s)
```

三、描述所使用的线性模型

3.1 基本形式

给定由d个属性描述的示例**x**=(x_1;x_2;...；x_d)，其中x_i是**x**在第i个属性上的取值，线性回归(linear regression)试图学得一个通过属性的线性组合来进行预测的函数，即

$$f(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + b$$

一般用向量形式写成

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b$$

其中**w**=(w_1;w_2;...；w_d)。**w**和b学得之后，模型就得以确定。

故本题中的模型应该为

$$\widehat{Sales} = w_1 \cdot TV + w_2 \cdot radio + w_3 \cdot newspaper + b，使得 \widehat{Sales} \cong Sales$$

此处有三个属性描述样本，故又称为多元线性回归(multivariate linear regression)。

其基本形式为

$$\hat{f}\left(\hat{x}_{N+1}\right) = \hat{x}_{N+1}^{T} \widehat{\omega}^{*}$$

$$其中 \hat{x}_{N+1} = \left(x_{N+1}; 1\right) \in \mathbb{R}^{n+1}, \widehat{\omega}^{*} = \left(\omega^{*}; b^{*}\right) \in \mathbb{R}^{n+1}$$

3.2 岭回归

吉洪诺夫正则化以安德烈·尼古拉耶维奇·吉洪诺夫命名，为非适定性问题的正则化中最常见的方法。在统计学中，本方法被称为脊回归或岭回归（ridge regression）；在机器学习领域则称为权重衰减或权值衰减（weight decay）。因为有不同的数学家独立发现此方法，此方法又称做吉洪诺夫 - 米勒法（Tikhonov–Miller method）、菲利浦斯 - 图米法（Phillips–Twomey method）、受限线性反演（constrained linear inversion method），或线性正规化（linear regularization）。

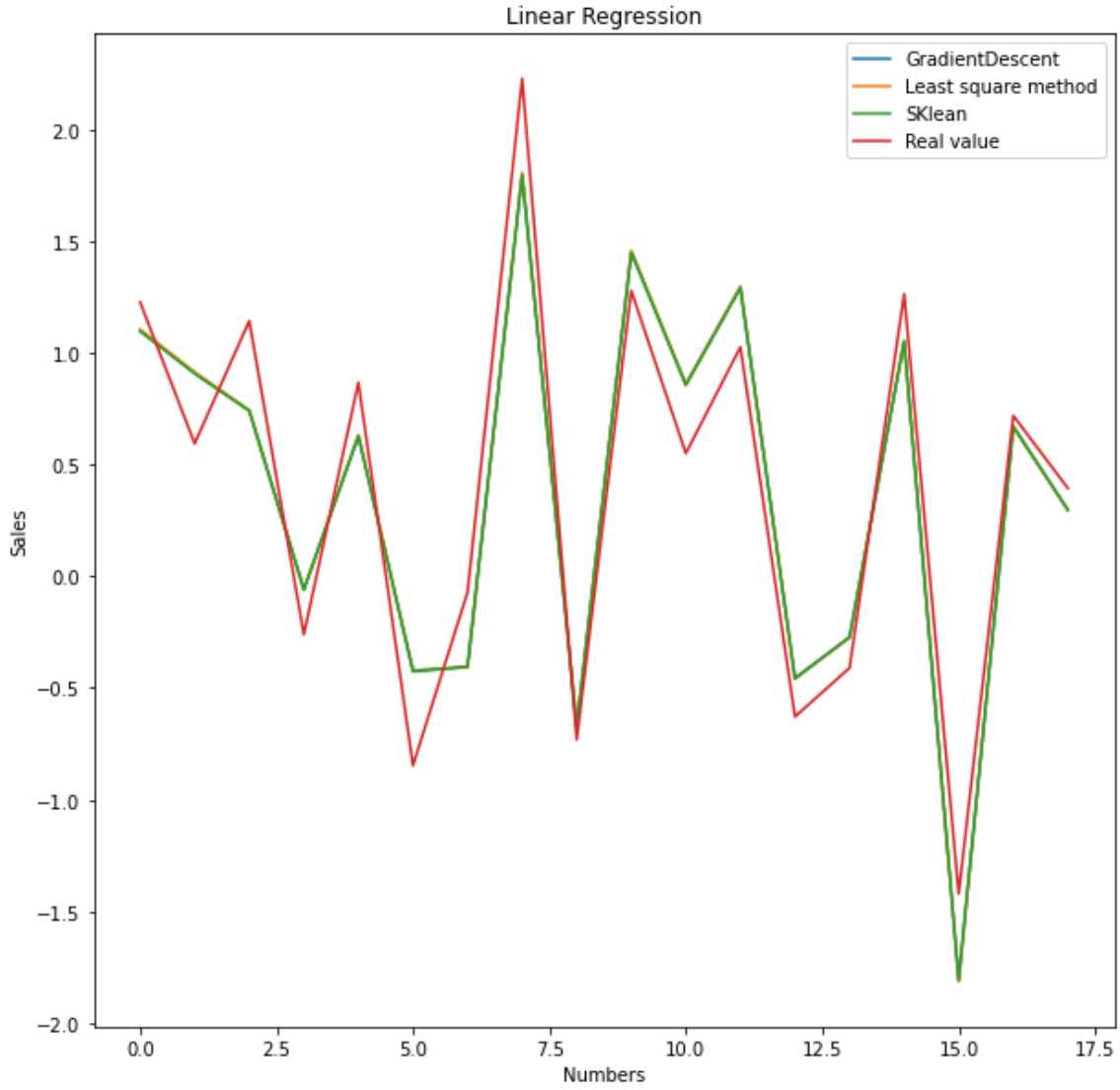
$$min\ L(W) = \frac{1}{2}(XW - y)^T(XW - y) + \frac{1}{2}\alpha||W||_2^2$$

$$W = (X^T X + \alpha I)^{-1} X^T y$$

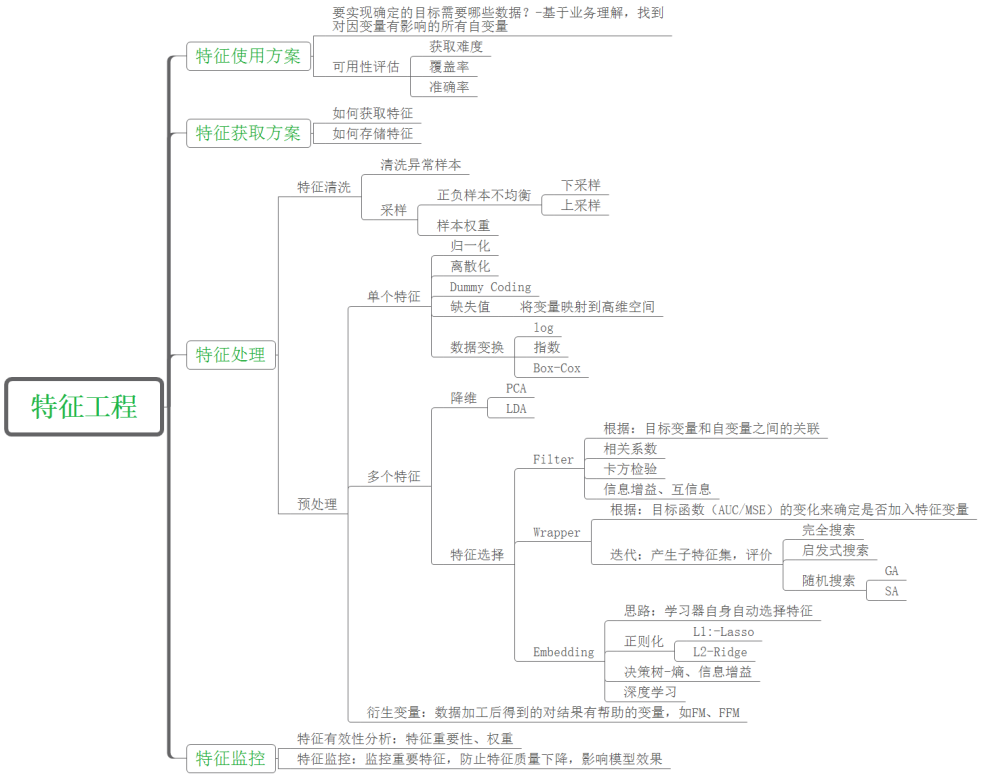
根据4.2、4.3的分析，我们最终决定在最小二乘法的基础上采取L2正则化，即岭回归。相应地，为了使用岭回归和缩减技术，首先需要对特征做标准化处理。因为，我们需要使每个维度特征具有相同的重要性，故采用了z-score标准化。随着模型复杂度的提升，在训练集上的效果就越好，即模型的偏差就越小；但是同时模型的方差就越大。对于岭回归的α而言，随着α的增大，$|X^T X + \alpha I|$就越大，$(X^T X + \alpha I)^{-1}$ 就越小，模型的方差就越小；而α越大使得**W**的估计值更加偏离真实值，模型的偏差就越大。所以岭回归的关键是找到一个合理的α值来平衡模型的方差和偏差。

本次使用10折交叉验证法来确定α值，每一种训练集和测试集下都会有对应的一个模型及模型评分（如均方误差），进而可以得到一个平均评分。对于α值则选择平均评分最优的α值。

3.3 特征工程



如图所示为梯度下降法，最小二乘法和sklearn调用所得结果与真实值的对比折线图。从中可以看出，三种折线都已经接近重合，但又与真实值存在差异。查阅资料后，我们了解了特征工程的相关知识。

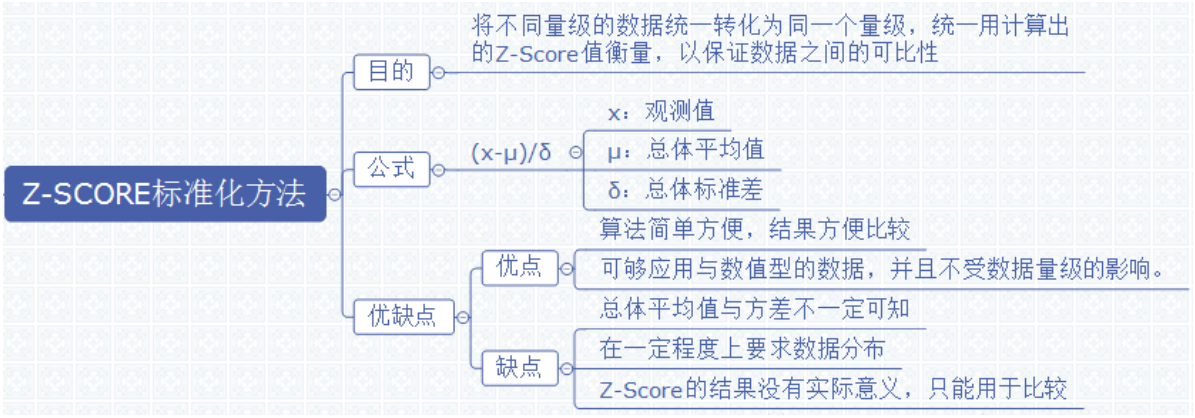


“数据决定了机器学习的上限，而算法只是尽可能逼近这个上限”，这里的数据指的就是经过特征工程得到的数据。特征工程指的是把原始数据转变为模型的训练数据的过程，它的目的就是获取更好的训练数据特征，使得机器学习模型逼近这个上限。特征工程能使得模型的性能得到提升，有时甚至在简单的模型上也能取得不错的效果。特征工程在机器学习中占有非常重要的作用，一般认为括特征构建、特征提取、特征选择三个部分。特征构建比较麻烦，需要一定的经验。 特征提取与特征选择都是为了从原始特征中找出最有效的特征。它们之间的区别是特征提取强调通过特征转换的方式得到一组具有明显物理或统计意义的特征；而特征选择是从特征集合中挑选一组具有明显物理或统计意义的特征子集。两者都能帮助减少特征的维度、数据冗余，特征提取有时能发现更有意义的特征属性，特征选择的过程经常能表示出每个特征的重要性对于模型构建的重要性。

本次作业中主要使用了特征构建、特征选择、特征缩放，具体结果将在第五部分讨论。

四、描述训练模型所使用的算法

4.1 数据预处理



本次数据处理使用的是z-score标准化，转换公式为：

$$z = \frac{x - \mu}{\sigma}$$

使用python具体实现如下：

```
def fit_transform(x):
    x = np.asarray(x)
    std_ = np.std(x, axis=0)  # 标准差
    mean_ = np.mean(x, axis=0)  # 均值
    return (x - mean_) / std_
```

4.2 策略

4.2.1 经验风险最小化

均方误差是回归任务中最常用的性能度量，因此我们可试图让均方误差最小化，即

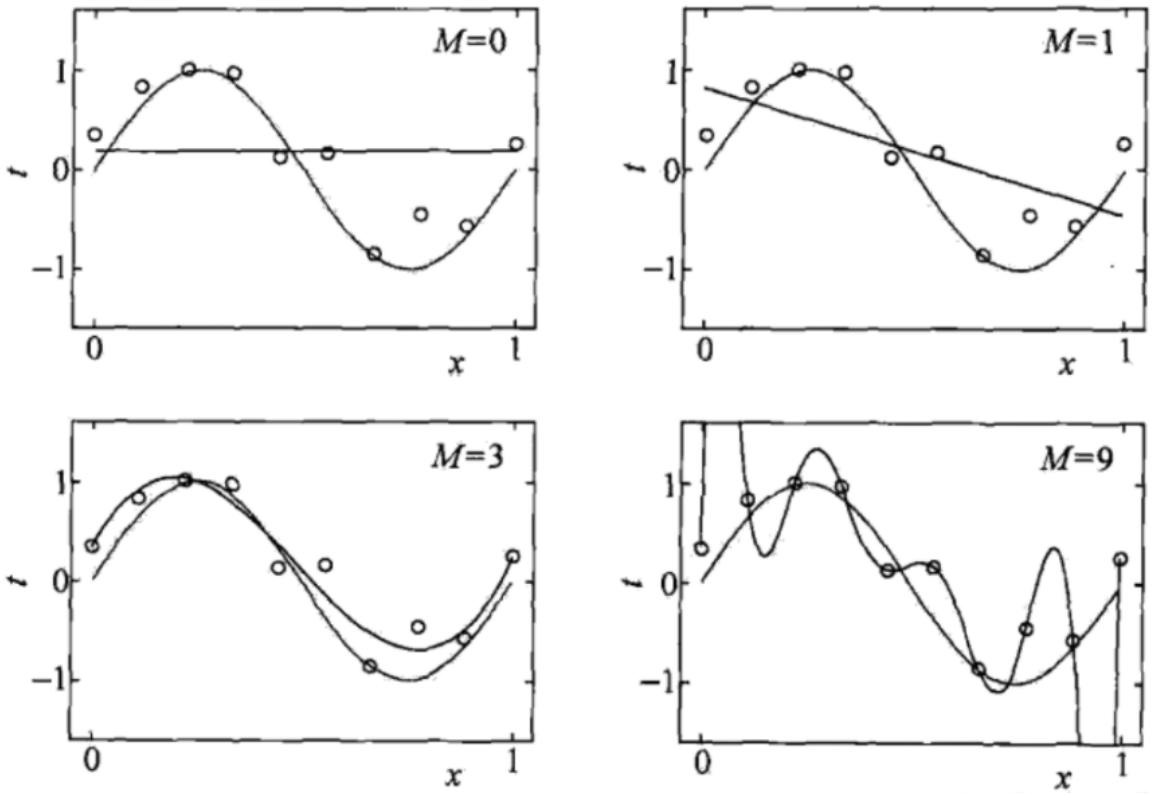
$$\begin{aligned}(w^*, b^*) &= \arg\min_{(w,b)} \sum_{i=1}^{m} (f(x_i) - y_i)^2 \\ &= \arg\min_{(w,b)} \sum_{i=1}^{m} (y_i - wx_i - b)^2\end{aligned}$$

均方误差有非常好的几何意义，它对应了常用的欧几里得距离或简称“欧氏距离”(Euclidean distance)。基于均方误差最小化来进行模型求解的方法称为“最小二乘法”(least square method)。具体求解过程在4.3中进行介绍。

4.2.2 结构风险最小化

4.2.2.1 正则化

当模型的复杂度增大时，训练误差会逐渐减小并趋于0；而测试误差会先减小，达到最大值后又增大。当选择的模型复杂度过大时，就会发生过拟合，如下图所示。



为了避免因为过拟合问题而导致拟合效果不佳，我们在经验风险上加一个正则化项或罚项，使结构风险最小，这种方法叫做正则化，一般具有如下形式：

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{N} L\left(y_i, f\left(x_i\right)\right) + \lambda J(f)$$

4.2.2.2 L1与L2正则化

使用L1范数（也称曼哈顿距离或Taxicab范数，只允许在与空间轴平行行径的距离）又叫**lasso**回归，损失函数变为：

$$J(\mathbf{W}) = \frac{1}{2n}(\mathbf{XW} - \mathbf{Y})^T(\mathbf{XW} - \mathbf{Y}) + \alpha\|W\|_1$$

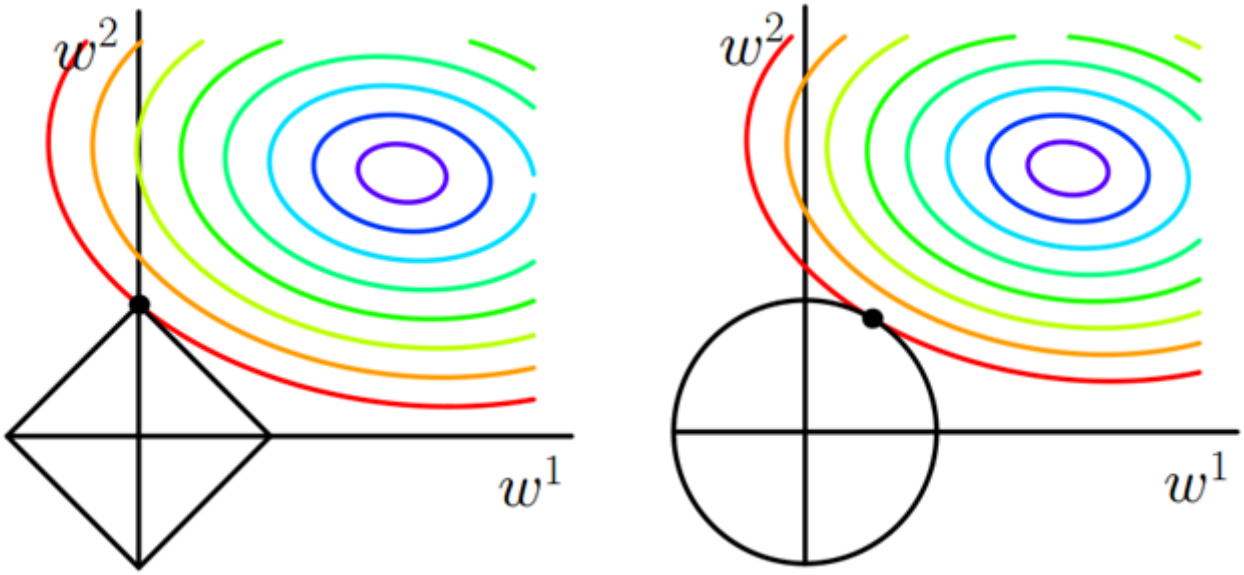
使用L2范数（也称欧几里德距离，是向量到原点的最短距离）又叫**ridge**回归，损失函数变为：

$$J(\mathbf{W}) = \frac{1}{2}(\mathbf{XW} - \mathbf{Y})^T(\mathbf{XW} - \mathbf{Y}) + \frac{1}{2}\alpha\|W\|_2^2$$

L1能使得一些特征的系数变小，甚至还使一些绝对值较小的系数直接变为0，产生稀疏解，起到特征选择的作用，增强模型的泛化能力。

L2的优点是可以限制|w|的大小，从而使模型更简单，更稳定，即使加入一些干扰样本也不会对模型产生较大的影响，而且还能解决非正定的问题，强制使XTX可逆有解。

$$\theta = \left(XX^T + \alpha I\right)^{-1}XY$$



在上图中，两个坐标分别是要学习到的两个参数ω1和ω2；彩色线是损失函数的等高线即损失值相等线；方形和圆形就分别是L1和L2所产生的额外误差（约束空间）；最后的目标要是两者最小，即要得到能使两者相加最小的点，也就是图中的黑色交点。在画等差图时，L1的效果就很容易与坐标轴相交了，这就是会产生很多0，即造成参数稀疏的原因。而且同时如果给一个微小的偏移，L2移动不会很大，而L1可能会移动到方形边上产生很多的交点，所以L1比较不稳定。

L2倾向于使ω的分量取值更均衡，即非零分量个数更稠密，而L1倾向ω的分量取值更稀疏，即非零分量个数更少。所以从图可以看出L1的边缘比较尖锐，与目标函数的等高线相交时，交点会常在那些尖锐的地方，所以很多的参数就是0，即L1能产生稀疏解。所以在调参时如果我们主要的目的只是为了解决过拟合，一般选择L2正则化就够了。但是如果选择L2正则化发现还是过拟合，即预测效果差的时候，就可以考虑L1正则化。另外，如果模型的特征非常多，我们希望一些不重要的特征系数归零，从而让模型系数稀疏化的话，也可以使用L1正则化。

综合考虑，我们在本次的损失函数中引入的是L2正则化。

4.3 算法

4.3.1 最小二乘法

4.3.1.1 问题分析

我们的策略是

$$min\ L(W)=\frac{1}{2}(XW-y)^T(XW-y)+\frac{1}{2}\alpha||W||_2^2$$

$$其中，X=\begin{pmatrix}x_{11} & x_{12} & \dots & x_{1n} & 1\\ x_{21} & x_{22} & \dots & x_{2n} & 1\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ x_{m1} & x_{m2} & \dots & x_{mn} & 1\end{pmatrix},W=\begin{pmatrix}\omega_1\\ \omega_2\\ \vdots\\ \omega_n\\ \omega_0\end{pmatrix}$$

我们进行展开

$$\begin{aligned}L(W)&=\frac{1}{2}(XW-y)^T(XW-y)+\frac{1}{2}\alpha||W||_2^2\\&=\frac{1}{2}(W^TX^T-y^T)(XW-y)+\frac{1}{2}\alpha W^TW\\&=\frac{1}{2}(W^TX^TXW-W^TX^Ty-y^TXW+y^Ty)+\frac{1}{2}\alpha W^TW\end{aligned}$$

下面，我们进行梯度推导

$$\begin{aligned}\frac{\partial L}{\partial W}&=\frac{1}{2}(2X^TXW-X^Ty-X^Ty)+\frac{1}{2}*2\alpha IW\\&=X^TXW-X^Ty+\alpha IW\quad(I是n\times n的单位矩阵)\end{aligned}$$

由于$L(W)$是关于W的凸函数，所以我们在梯度为零的点，即是我们要求的最优解。

$$令\frac{\partial L}{\partial W}=0$$

$$得(X^TX+\alpha I)W=X^Ty$$

我们要通过此方法求得W，需要的条件是$X^TX+\alpha I$可逆，若其可逆，则W的解是

$$W=(X^TX+\alpha I)^{-1}X^Ty$$

因为最小二乘法要求$X^TX+\alpha I$必须存在可逆矩阵，在实际问题中可能不满足，于是我们下面采用梯度下降法进行迭代求解。

4.3.1.2 代码实现

```
def lms(x_train, x_test, y_train, y_test):
    x_train_=np.c_[np.ones([x_train.shape[0],1]),x_train]
    x_test_=np.c_[np.ones([x_test.shape[0],1]),x_test]
    theta_n = np.dot(np.dot(inv(np.dot(x_train_.T,
x_train_)+0.1*np.eye(x_train_.shape[1])), x_train_.T), y_train)  # theta =
(X`X)^(-1)X`Y，其中X`表示X的转置，使用L2范数正则化
    y_pre = np.dot(x_test_, theta_n)
    mse = np.average((y_test - y_pre) ** 2)
    return theta_n, y_pre, mse
```

4.3.2 梯度下降法

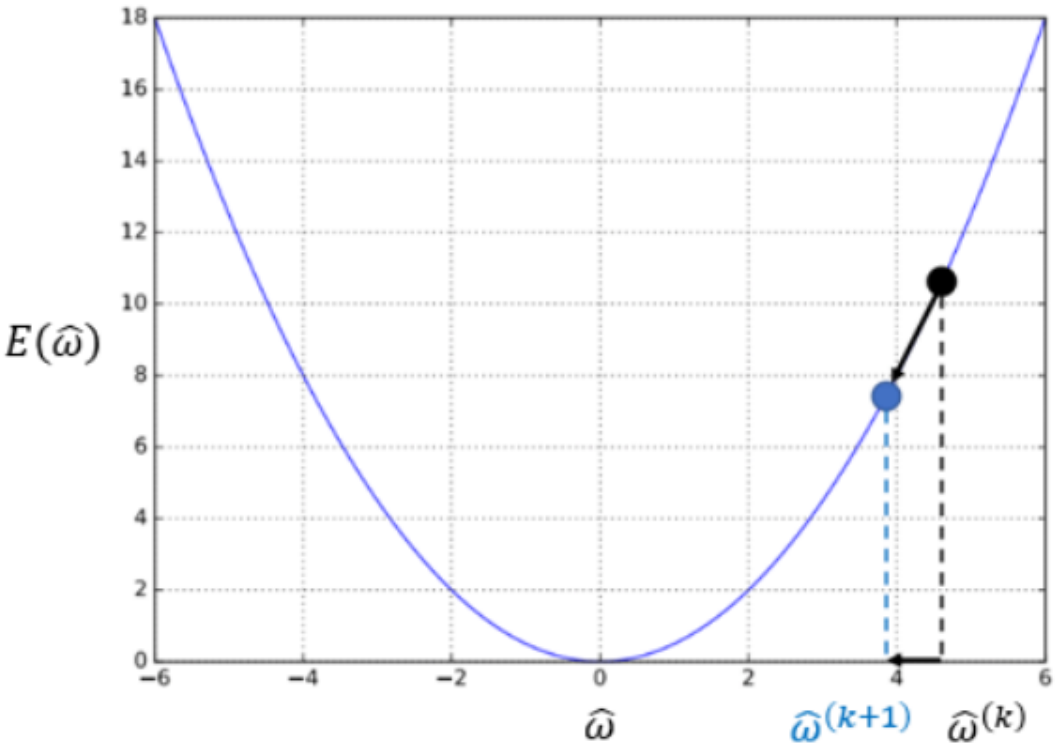
4.3.2.1 问题分析

首先，我们的目标是下式

$$令\hat{\omega} = W$$

$$E(\hat{\omega}) = \frac{1}{2}(X\hat{\omega} - y)^T(X\hat{\omega} - y) + \frac{1}{2}\alpha||\hat{\omega}||_2^2 \quad ,\hat{\omega} = \underset{\hat{\omega}}{argmin}\ E(\hat{\omega})$$

梯度下降法是一种迭代算法：我们选取适当的初始值$\hat{\omega}^{(0)}$，不断迭代，更新$\hat{\omega}$的值，进行目标函数的极小化，直到收敛。由于负梯度方向是使得函数值下降最快的方向，在迭代的每一步，以负梯度方向更新$\hat{\omega}$的值，从而达到减小函数值的目的。如下图形象化表示：



$$\hat{\omega}^{(k+1)} \leftarrow \hat{\omega}^{(k)} - \eta \times \left.\frac{\partial E(\hat{\omega})}{\partial \hat{\omega}}\right|_{\hat{\omega}=\hat{\omega}^{(k)}}$$

4.3.2.2 核心思想

1. $E(\hat{\omega})$是具有一阶连续偏导数的凸函数，其极值点在一阶导数为零的地方取得
2. 一阶泰勒展开：$E(\hat{\omega}) \approx E(\hat{\omega}^{(k)}) + \nabla E(\hat{\omega}^{(k)})(\hat{\omega} - \hat{\omega}^{(k)})$，其中，$\nabla E(\hat{\omega}^{(k)})$是$E(\hat{\omega})$在$\hat{\omega}^{(k)}$的梯度：

$$\nabla E(\hat{\omega}^{(k)}) = \frac{\partial E(\hat{\omega})}{\partial \hat{\omega}}|_{\hat{\omega}=\hat{\omega}^{(k)}}$$

3. 求取第k+1次迭代值：$\hat{\omega}^{k+1} = \hat{\omega}^{(k)} + \eta_k * (-\nabla E(\hat{\omega}^{(k)}))$，其中$\eta_k$是步长，有我们最初指定。梯度如下（推导在上面部分）：

$$\frac{\partial E}{\partial \hat{\omega}} = X^TX\hat{\omega} - X^Ty + \alpha I\hat{\omega} \quad (I是n \times n的单位矩阵)$$

4.3.2.3 求解步骤

输入：目标函数$E(\hat{\omega})$，梯度函数$\nabla E(\hat{\omega})$，计算精度ε，步长η_k；

输出： $E(\hat{\omega})$的极小点$\hat{\omega}^*$。

(1) 取初始值$\hat{\omega}^{(0)} \in \mathbb{R}^{d+1}$，置k=0；

(2) 计算$E(\hat{\omega}^{(k)})$；

(3) 计算梯度$\nabla E(\hat{\omega}^{(k)})$，当$||\nabla E(\hat{\omega}^{(k)})|| < \varepsilon$时，令$\hat{\omega}^* = \hat{\omega}^{(k)}$，

停止迭代；

(4) 置$\hat{\omega}^{(k+1)} = \hat{\omega}^{(k)} + \eta_k(-\nabla E(\hat{\omega}^{(k)}))$，计算$E(\hat{\omega}^{(k+1)})$，

当$||E(\hat{\omega}^{(k+1)}) - E(\hat{\omega}^{(k)})|| < \varepsilon$或$||\hat{\omega}^{(k+1)} - \hat{\omega}^{(k)}|| < \varepsilon$时，

令$\hat{\omega}^* = \hat{\omega}^{(k)}$，停止迭代；

(5) 否则，置k=k+1，转步骤（3）。

4.3.2.4 代码实现

```
class GradientDescent_MultiLine:
    def __init__(self, lr, epochs):
        self.lr = lr  # 学习率，用来控制步长（权重调整幅度）
        self.epochs = epochs  # 循环迭代的次数
        self.lose = []  # 损失值计算（损失函数）：均方误差

    '''根据提供的训练数据对模型进行训练'''

    def fit(self, x, y):
        x = np.asarray(x)
        y = np.asarray(y)
        y = np.squeeze(y)  # 去掉冗余的维度

        self.w = np.zeros(1 + x.shape[1])  # 初始权重，权重向量初始值为0（或任何其他值），长度比X的特征数量多1（多出来的为截距）

        # 开始训练
        for i in range(self.epochs):
            y_hat = np.dot(x, self.w[1:]) + self.w[0]  # 计算预测值
            error = y - y_hat  # 计算真实值与预测值之间的差距
            self.lose.append(np.sum(error ** 2) / 2 + 0.1* np.dot(self.w.T, self.w))  # 将损失加入到损失列表中，使用L2范数正则化
            #print("迭代次数:{0},进度：{1}%".format(i + 1, 100.0 * (i + 1) / self.epochs), "  loss:", np.sum(error ** 2) / 2)
            # j <- j + α * sum((y - y_hat) * x(j))
            x_=np.c_[np.ones([x.shape[0],1]),x]
            #self.w[0] += self.lr * np.sum(error)
            #self.w[1:] += self.lr * np.dot(x.T, error)
            I=np.identity(x_.shape[1])
            self.w=self.w-self.lr*(np.dot((np.dot(x_.T, x_)+0.2*I), self.w)-np.dot(x_.T, y))
```

五、分析模型训练结果，包括训练误差和测试误差

5.1 评估指标计算公式

训练误差是模型关于训练数据集的平均损失；测试误差是模型关于测试数据集的平均损失。计算公式如下：

$$R_{emp}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L\left(y, \hat{f}\left(x_i\right)\right)$$

$$e_{test} = \frac{1}{N'}\sum_{i=1}^{N'} L\left(y_i, \hat{f}\left(x_i\right)\right)$$

其中N为训练样本容量，*N'*为测试样本容量。由于我们在线性回归中使用的是平方损失函数，故上述计算结果又叫均方误差 MSE（Mean Squared Error）：

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left(y_{\text{test}}^{(i)} - \hat{y}_{\text{test}}^{(i)}\right)^2$$

但是，MSE 公式有一个问题是会改变量纲。因为公式平方了，我们可以对这个MSE开方，得到第二个评价指标：均方根误差 RMSE（Root Mean Squared Error）：

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y_{\text{test}}^{(i)} - \hat{y}_{\text{test}}^{(i)}\right)^2}$$

但是MSE不甚全面，某些情况下决定系数 R2（coefficient of determination）显得尤为有用，它可以看作是MSE的标准化版本，用于更好地解释模型的性能。R2值的定义如下：

$$R^2 = 1 - \frac{\left(\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2\right)/m}{\left(\sum_{i=1}^{m}\left(y^{(i)} - \bar{y}\right)^2\right)/m} = 1 - \frac{MSE(\hat{y}, y)}{\text{Var}(y)}$$

5.2 误差分析思路

结合前文的推导分析，我们最终采用的是线性最小二乘法与L2正则化，即alpha取值为1.0的**Ridge回归**，并结合特征工程中特征构建（将**TV*radio**，**radio*newspaper**作为新的特征），特征选择（加入新的特征，舍弃相关系数较小的newspaper），特征缩放（将TV，radio，newspaper进行开方、平方、三次方等）的思路进行了14种情况的实验，并得出了每一种情况的MSE，RMSE，R^2。

所使用的代码为：

```
X = np.asarray(data_s.get(['TV','radio','newspaper']))
y = np.asarray(data_s.get('sales'))
clf.fit(X,y)
print('MSE=%f'%(-0.1*cross_val_score(clf, X, y, cv=10,
scoring='neg_mean_squared_error').sum()))
print('RMSE=%f'%(-0.1*cross_val_score(clf, X, y, cv=10,
scoring='neg_root_mean_squared_error').sum()))
print('R2=%f'%(10*cross_val_score(clf, X, y, cv=10, scoring='r2').sum())+'%')
print(clf.coef_)
print(clf.intercept_)
```

5.3 训练结果

14种情况的训练误差及测试误差记录在jupyter notebook的ipynb文件中，此处展示其中三种。

```
X1 = np.asarray(data_s.get(['TV','radio','newspaper']))
y = np.asarray(data_s.get('sales'))
clf.fit(X1,y)
print('MSE=%f'%(-0.1*cross_val_score(clf, X1, y, cv=10, scoring='neg_mean_squared_error').sum()))
print('RMSE=%f'%(-0.1*cross_val_score(clf, X1, y, cv=10, scoring='neg_root_mean_squared_error').sum()))
print('R2=%f'%(10*cross_val_score(clf, X1, y, cv=10, scoring='r2').sum())+'%')
print(clf.coef_)
print(clf.intercept_)
```

```
MSE=0.005061
RMSE=0.069802
R2=83.421193%
[ 0.46045157  0.33068973 -0.00784677]
0.09504622538264329
```

```
X2 = np.asarray(data_s.get(['TV','radio']))
clf.fit(X2,y)
print('MSE=%f'%(-0.1*cross_val_score(clf, X2, y, cv=10, scoring='neg_mean_squared_error').sum()))
print('RMSE=%f'%(-0.1*cross_val_score(clf, X2, y, cv=10, scoring='neg_root_mean_squared_error').sum()))
print('R2=%f'%(10*cross_val_score(clf, X2, y, cv=10, scoring='r2').sum())+'%')
print(clf.coef_)
print(clf.intercept_)
```

```
MSE=0.005031
RMSE=0.069601
R2=83.515904%
[0.46047397 0.32913892]
0.09366144373035651
```

```
data_s['TV_min'] = data_s['TV'].apply(lambda x:x**0.2)
data_s['TV_radio']=data_s['TV']*data_s['radio']
X13 = np.asarray(data_s.get(['TV_radio','TV_min','radio','newspaper']))
clf.fit(X13,y)
print('MSE=%f'%(-0.1*cross_val_score(clf, X13, y, cv=10, scoring='neg_mean_squared_error').sum()))
print('RMSE=%f'%(-0.1*cross_val_score(clf, X13, y, cv=10, scoring='neg_root_mean_squared_error').sum()))
print('R2=%f'%(10*cross_val_score(clf, X13, y, cv=10, scoring='r2').sum())+'%')
print(clf.coef_)
print(clf.intercept_)
```

```
MSE=0.001193
RMSE=0.033860
R2=95.937956%
[ 0.4974528   0.4217457   0.10899676 -0.01095106]
-0.03811764712515803
```

上图所示的第13种情况训练所得模型的各项评估指标最优，故将其model文件保存，助教老师可以使用test.ipynb自动载入模型，并计算出测试误差MSE。

```
import joblib
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error
model = joblib.load('model.pickle') #载入模型
data=pd.read_csv('5_test.csv') #读入数据
data_s = (data-data.min())/(data.max()-data.min()) #归一化
data_s['TV_min'] = data_s['TV'].apply(lambda x:x**0.2)
data_s['TV_radio']=data_s['TV']*data_s['radio']
X = np.asarray(data_s.get(['TV_radio','TV_min','radio','newspaper']))
y = np.asarray(data_s.get('sales'))
print('测试误差MSE=%f'%mean_squared_error(y,model.predict(X)))
```

六、总结模型训练过程中的收获

6.1 学习数据分析处理

在进行计算之前，我们首先对数据进行了预处理和分析。首先，我们检查了数据是否缺失。然后，我们画出了散点图，散点图矩阵，相关系数热力图等，分析了销售量和各项广告投入量之间的数据关系，以便于对数据的进一步处理。在对数据的处理中，我们首先进行了数据标准化，将不同量级的数据统一转化为同一量级，以保证数据之间的可比性。而后，我们查阅资料，为了获取更好的训练数据特征，了解了特征工程相关内容，再根据之前对数据的分析，我们对数据进行了特征构建、特征选择等，具体结果上面已经展示。

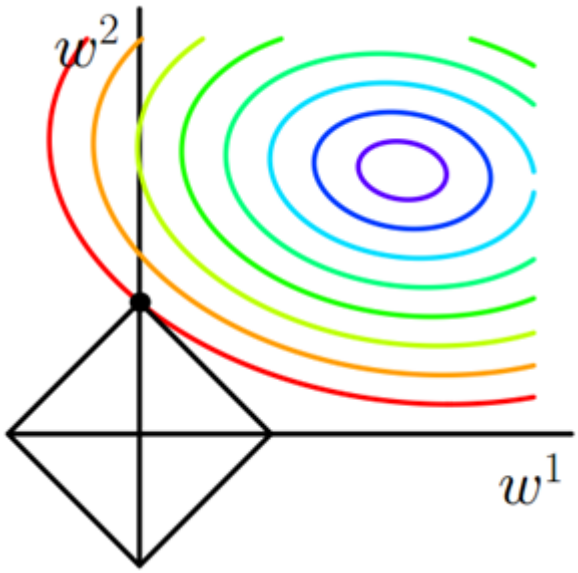
6.2 加深对十折交叉验证的理解

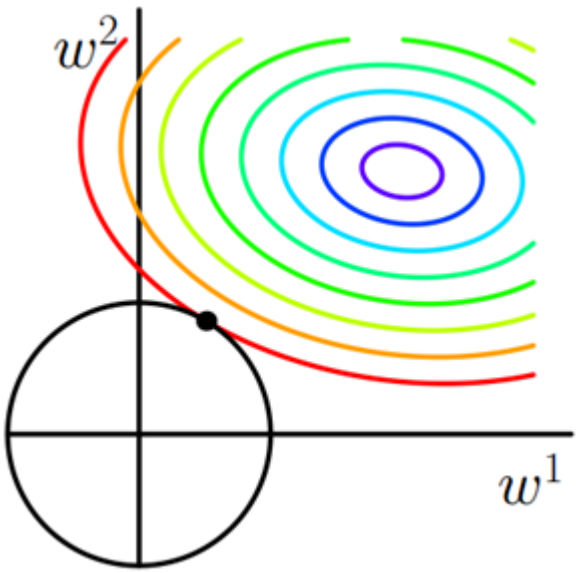
在机器学习的模型选择中，我们要对候选模型的泛化误差进行评估，然后选择泛化误差最小的那个模型。而实际应用中，我们无法直接获得泛化误差，而训练误差又由于过拟合现象的存在而不适合作为标准，所以我们随机将数据集切为三部分：训练集，验证集和测试集。在十折交叉验证中，我们通过某种特定的划分，将所有数据划分为十个，并依次选取作为测试集，剩下的作为训练集。在这个过程中，我们加深了对十折交叉验证的理解。

6.3 对于正则化的理解加深

正则化的目的：防止过拟合。过拟合指的是给定一堆数据，这堆数据带有噪声，利用模型去拟合这堆数据，可能会把噪声数据也给拟合了，这一方面会造成模型比较复杂，比如，原本一次函数能够拟合的数据，由于数据带有噪声，导致需要用五次函数来拟合；另一方面，同时会导致模型的泛化性能很差，在测试集上的结果准确率非常高，但测试新数据时，因为得到的是过拟合的模型，正确率会很低。

正则化的本质：约束（限制）要优化的参数。本来**解空间**是全部区域，但通过正则化添加了一些约束，使得解空间变小了，甚至在个别正则化方式下，解变得稀疏了。正如下图所示：





彩色线就是优化过程中遇到的等高线，一圈代表一个目标函数值，圆心就是样本观测值（假设一个样本），半径就是误差值，受限条件就是黑色边界（就是正则化的部分），二者相交处，才是最优参数。

可以看到，L1 与L2 的不同就在于L1在和每个坐标轴相交的地方都有"角"出现，而目标函数的测地线除非位置摆得非常好，大部分时候都会在角的地方相交。注意到在角的位置就会产生稀疏性，例如图中的相交点就有w1=0，而更高维的时候除了角点以外，还有很多边的轮廓也是既有很大的概率成为第一次相交的地方，又会产生稀疏性。相比之下，L2就没有这样的性质，因为没有角，所以第一次相交的地方出现在具有稀疏性的位置的概率就变得非常小了。这就从直观上来解释了为什么L1-regularization 能产生稀疏性，而L2-regularization 不行的原因了。

6.4 关于算法的择优

最开始我们分析结构风险最小化的策略，最小二乘法可能不可逆，同时为了增加模型的泛化能力，我们在损失函数中加入了惩罚项，由于对L1，L2正则化的分析，我们选择L2正则化，经过推导，发现最小二乘法可以直接得到解析解，解决了W系数矩阵非正定问题。由于梯度下降法是通过迭代逼近结果，所以只能得到近似解，所以我们选择最小二乘法来进行计算。