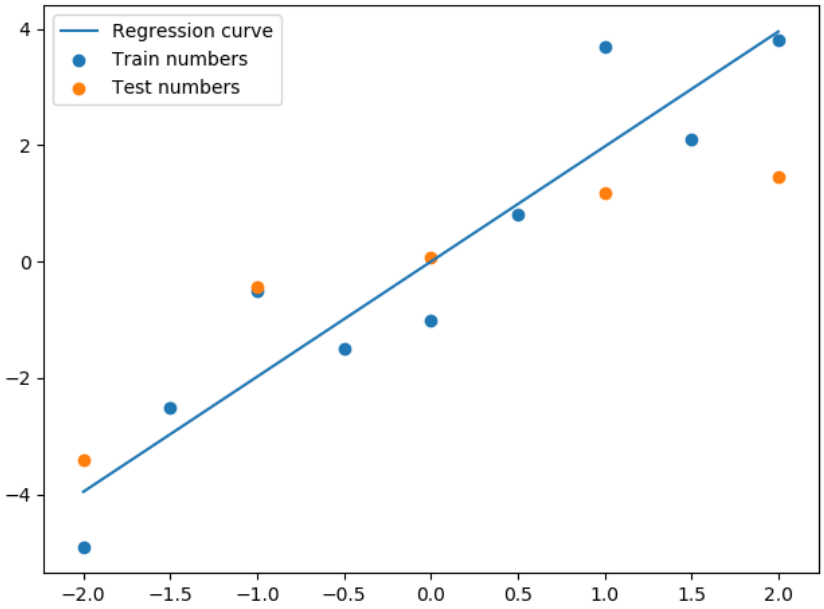
实验二: 回归模型 Regression Models

回归模型介绍

- 回归是监督学习的一个重要问题，回归用于预测输入变量和输出 变量之间的关系，特别是当输入变量的值发生变化时，输出变量 的值也随之发生变化。
- 回归模型是一种表示从输入变量到输出变量之间映射的函数
- 对连续值的预测
- 可以用合适的曲线揭示样本点随着自变量的变化关系

基本要求

a)根据数据集dataset_regression.csv，求最小二乘解，用得到的回归方程生成五个测试样本,画出回归曲线，给出训练误差和测试误差。

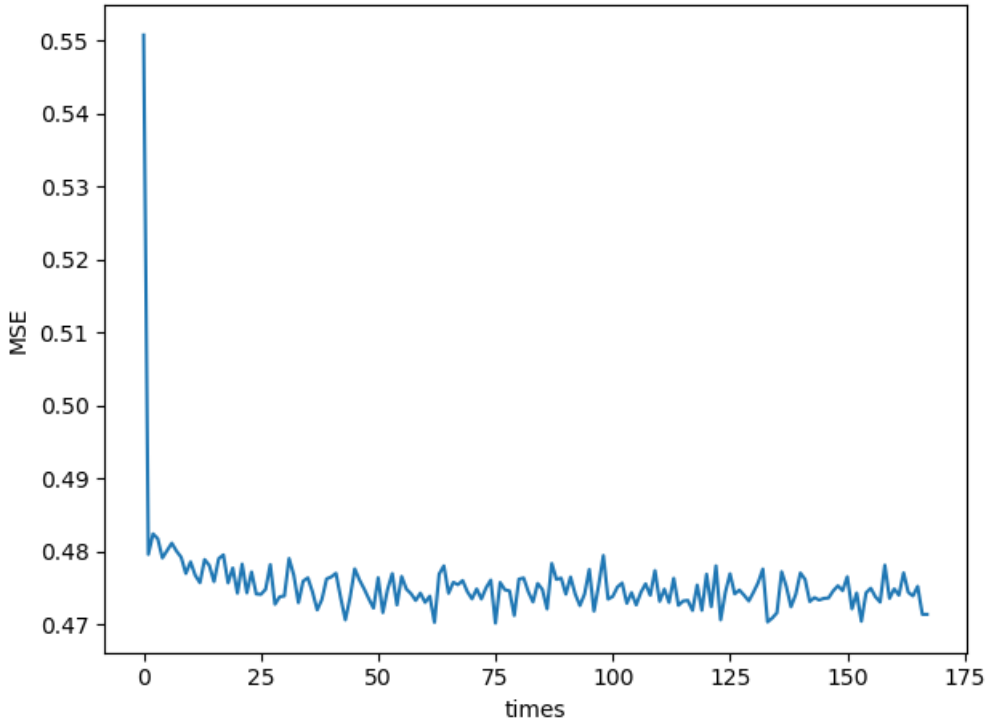


- 训练误差和测试误差:
  - 平均测试误差抖动较大，因为每次随机生成的数据都不一样。

```
最小二乘回归函数：y=-0.0000+1.9767*x
最小二乘回归的平均训练误差为0.3207，平均测试误差为0.6180
```

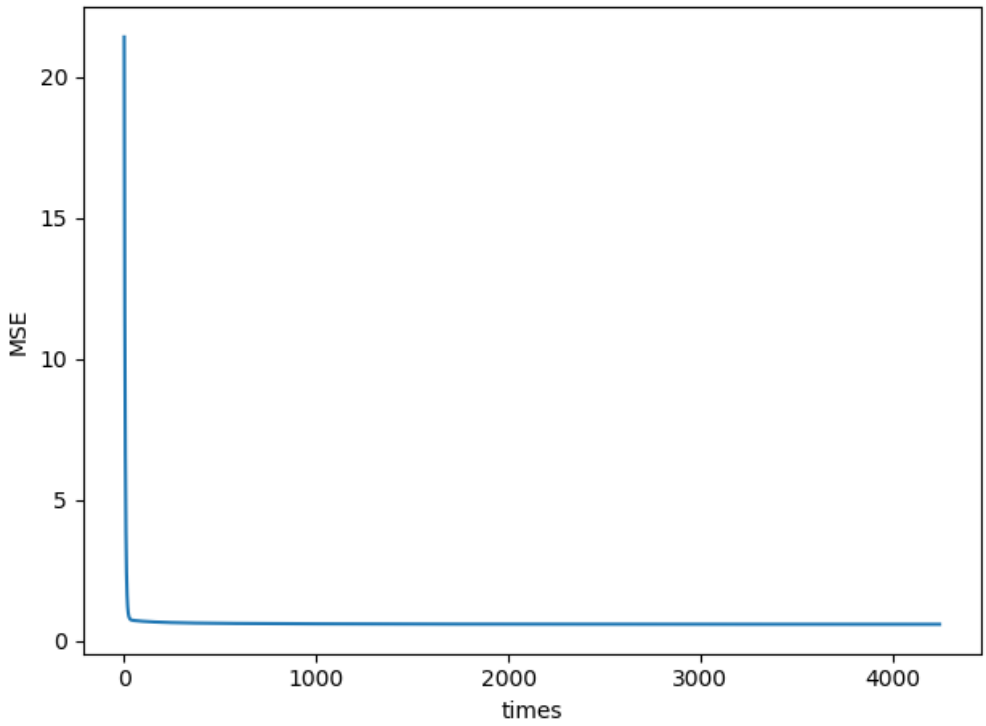
b)将数据集winequality-white.csv按照4:1划分为训练集和测试集，构造线性回归模型， 采用批量梯度下降或者随机梯度下降均可；输出训练集和测试集的均方误差（MSE），画出MSE收敛曲线。

- 随机梯度下降结果:



可以看出随机梯度下降由于其随机选择某一个样本的梯度近似总体的梯度，所以**loss**的下降是有波动的。但是优点是其迭代的次数比较少，这可能是它较快的落入局部最优的原因

- 批量梯度下降结果：



在批量梯度下降中，一直考虑的是全局的梯度。因此**MSE**下降的趋势比较平稳且迅速，但是为了到达全局最优，迭代次数变多。优点是收敛厚**MSE**最后比梯度下降的方式要小。

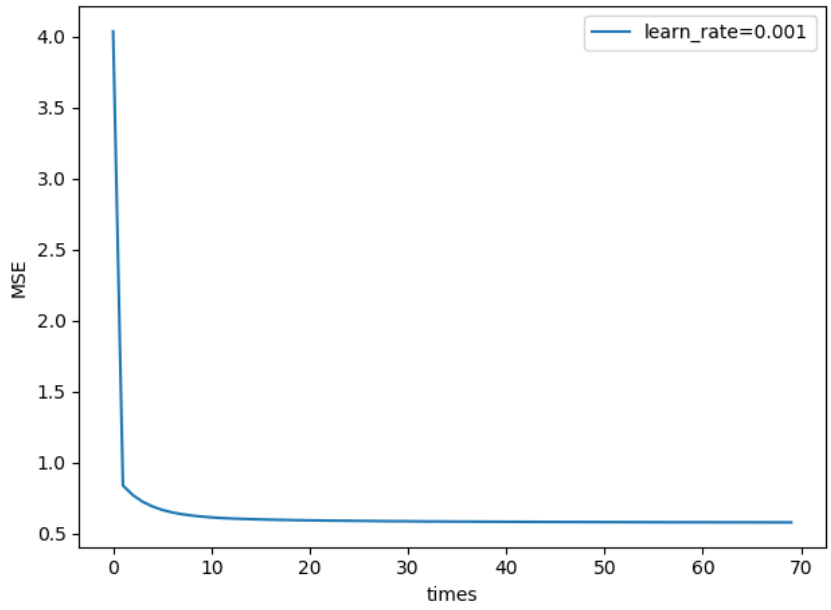
- 训练集测试集的均方误差：

```
Wine quality regression:
learn rate= 0.1
train error: 0.6549896347404952
test error: 0.6330651781455737
```

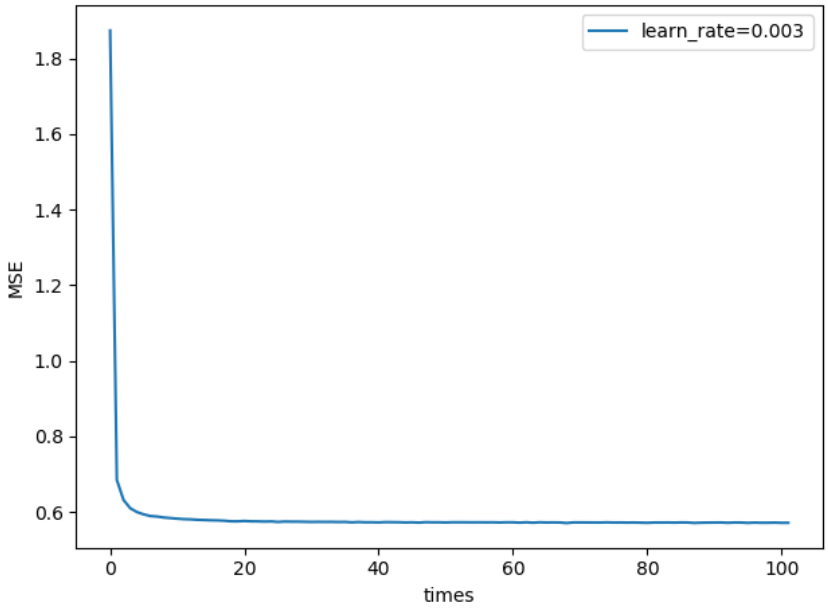
中级要求

中级要求: 尝试不同的学习率并进行MSE曲线展示，分析选择最佳的学习率。

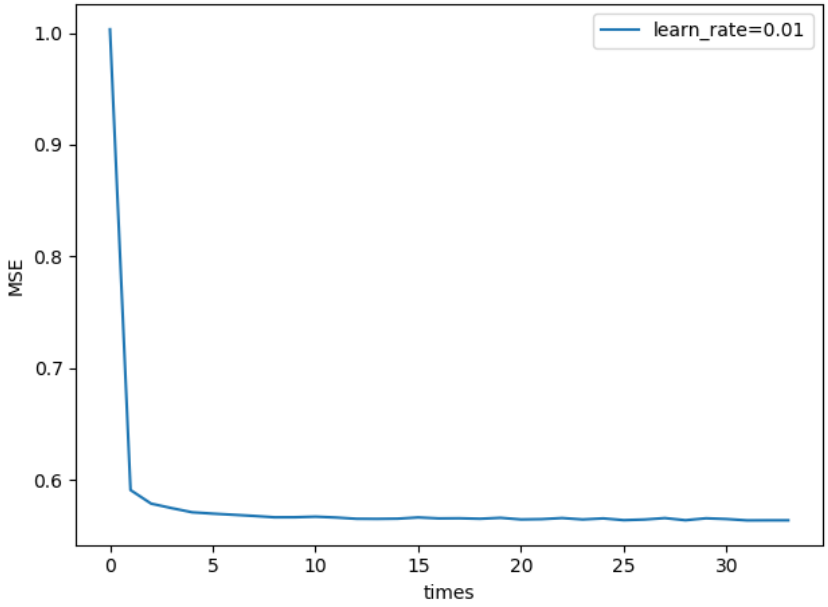
- 我尝试了alpha = [0.001, 0.003, 0.01, 0.03, 0.1, 0.3]这些学习率，使用的是随即梯度下降的方式，下面是结果：
    - alpha=0.001：
        - train error: 0.57510218844449482
          test error: 0.5506867341913374



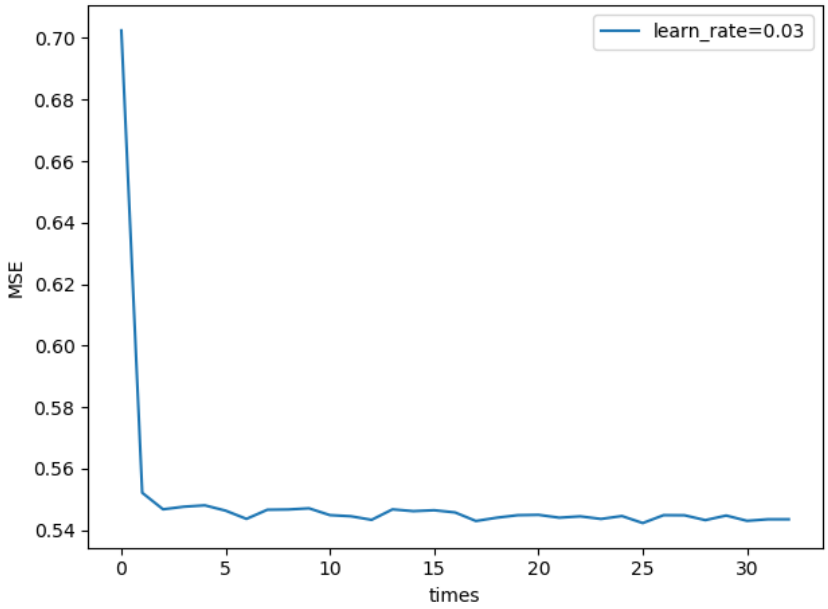
    - alpha=0.003：
        - train error: 0.5757445780180929
          test error: 0.5508716457030419



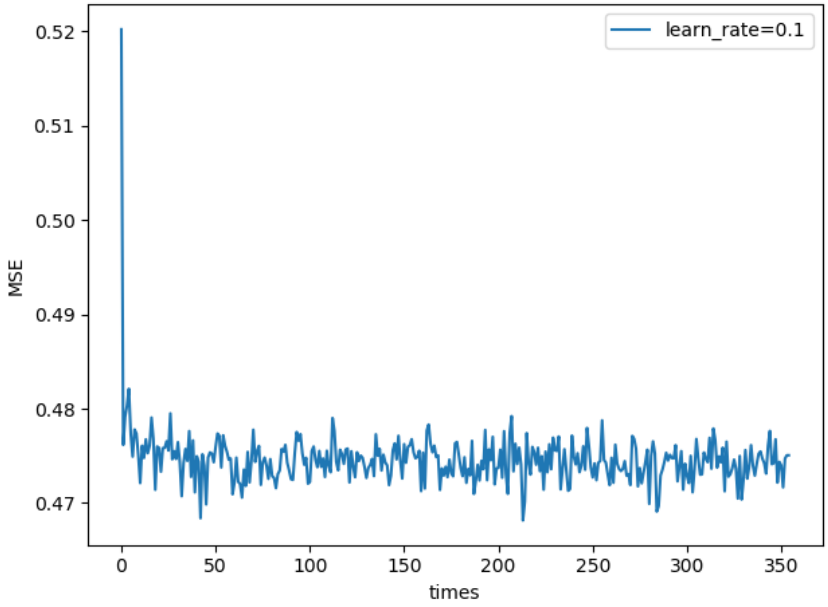
- alpha=0.01:
  - train error: 0.5732752949079186
    test error: 0.5505579855245638



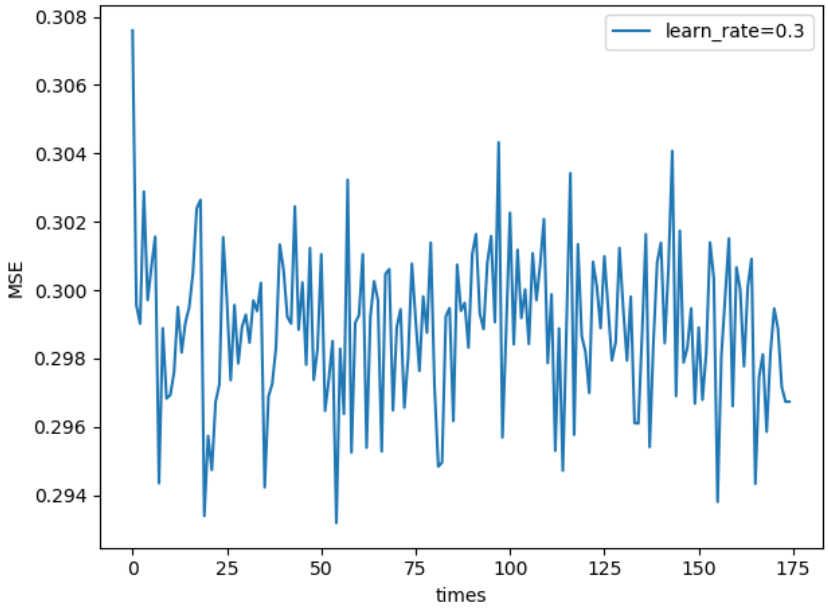
- alpha=0.03:
  - train error: 0.5732973637495883
    test error: 0.5488855229320233



- alpha=0.1:
  - train error: 0.5735563922538154
    test error: 0.5425729755236841



- alpha=0.3:
  - train error: 0.5980819102072039
    test error: 0.5722818681682563



①分析：

- 随着学习率的不断增大有如下的三个明显趋势：
  1. 随着学习率的不断增大，MSE抖动加剧。原因可能是大的学习率很可能会跳过极值点，甚至跳过某个波峰和波谷进入另一个波峰波谷区域。
  2. 随着学习率的不断增大，MSE值总体呈现下降趋势。原因可能是大的学习率能够避免过早的收敛到局部最优中，从而找到更好的收敛极值点。
  3. 随着学习率的不断增大，迭代次数不断增加。这也很容易理解，和第1点的原因相似，学习率太大从而导致其四处横跳，难以收敛。

②最佳学习率选择：

- 首先如果只看迭代次数的话，0.01 和0.03都收敛非常快，放入备选；
- 其次再只看MSE的话，0.1以下的学习率的MSE最后都在0.5以上，所以0.1和0.3的MSE均方误差表现最好。其中学习率等于0.3 的时候，甚至MSE小于了0.3；
- 最后再看erro的表现，在0.1及其以前都变化不大（其中0.1的时候最低），但是到0.3的时候，出现了训练集和测试集的erro都有上升的趋势，因此0.3移出备选。

  最终我选择0.03作为最佳学习率，因为其迭代次数很快（30次），MSE的收敛虽然比较一般（0.54左右收敛），但是erro相比于0.01来说泛化性能更好（0.5488左右）

高级要求

高级要求: 编程实现岭回归算法，求解训练样本的岭回归模型，平均训练误差和平均测试误差（解析法、批量梯度下降法和随机梯度下降法均可）。

- 岭回归的公式：
  -

$$J_N(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - f(\mathbf{x}^{(i)};\boldsymbol{\theta})\right)^2 + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$$
$$= \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$$

对$\boldsymbol{\theta}$求导得

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

解得

$$\boldsymbol{\theta} = (\mathbf{X}^T\mathbf{X} + \lambda\boldsymbol{I})^{-1}\mathbf{X}^T\mathbf{y}$$

- 代码实现：

```
def ridgeRegres(X,Y,lam=0.2):
    X=np.vstack((np.ones(X.shape[0]),X)).T
    xTx = np.dot(X.T,X)
    denom = xTx + np.eye(np.shape(xTx)[1])*lam
    theta = np.dot(np.linalg.inv(denom), np.dot(X.T,Y))
    print("岭回归的回归函数：y={:.4f}+
{:4f}*x".format(theta[0],theta[1]))
    return theta
```

- 和最小二乘进行对比（使用第一个数据集进行）
  - 最小二乘：

```
最小二乘回归函数：y=-0.0000+1.9767*x
最小二乘回归的平均训练误差为0.3207，平均测试误差为0.2259
```

  - 岭回归：

```
岭回归的回归函数：y=-0.0000+1.950658*x
岭回归的平均训练误差为0.3209，平均测试误差为0.3596
```

    二者训练误差相差不大，而且平均误差的波动受到随机生成的测试集的影响，因此在此无法评判谁更加优秀。

    但是岭回归主要解决的问题是两种：一是当预测变量的数量超过观测变量的数量的时候（预测变量相当于特征，观测变量相当于标签），二是数据集之间具有多重共线性，即预测变量之间具有相关性。所以如果有以上两个问题，最好使用岭回归而不是最小二乘的方式。