1. 课程介绍
   * + 1. 线性模型的含义.
       2. 线性回归的思想及其分类.
       3. Ridge回归与Lasso回归(掌握).
       4. 理解线性分类模型的含义.
2. 线性模型
   1. 线性回归

有许多种不同的线性回归模型. 其中,Ordinary Least Squares (OLS)是最经典和最简单的线性回归方法.

基本思想：找出使得预测值与真实的回归目标的平均平方误差(Mean Squared Error)最小的w和b的值.

线性回归模型的MSE损失函数为:

例1． 简单情形下的线性回归.

1. ＃linear\_regression.py
2. **import** mglearn
3. **import** matplotlib.pyplot as plt
4. **from** sklearn.model\_selection **import** train\_test\_split
5. **from** sklearn.linear\_model **import** LinearRegression
7. X, y = mglearn.datasets.make\_wave(n\_samples=60)
8. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=42)
9. lr = LinearRegression().fit(X\_train, y\_train)
10. # w are stored in the coef\_ attribute,
11. # b is stored in the intercept\_ attribute.
12. **print**("lr.coef\_:{}".format(lr.coef\_))
13. **print**("lr.intercept\_:{}".format(lr.intercept\_))
15. **print**("Traing set score: {:.2f}".format(lr.score(X\_train,y\_train)))
16. **print**("Test set score:{}".format(lr.score(X\_test,y\_test)))

结果：

lr.coef\_:[0.39390555]

lr.intercept\_:-0.0318043430268

Traing set score: 0.67

Test set score:0.659336859686

注意：

1. coef\_是Numpy数组；　intercept\_是一个浮点型数.

2. 训练集和测试集的准确度都不高.这说明我们的方法是欠拟合的，而不是过拟合.

3. 但是对于具有更多特征的数据集，线性模型是很有用的！这种情况下，更多的情况是过拟合出现，而非欠拟合.(例)

4. 本例中我们用的真实的数据集.

例2. 对于具有更复杂的特征的数据集,用OSL方法做线性回归.

1. #linear\_regression2.py
2. **import** mglearn
3. **import** matplotlib.pyplot as plt
5. **from** sklearn.model\_selection **import** train\_test\_split
6. **from** sklearn.linear\_model **import** LinearRegression
8. X, y = mglearn.datasets.load\_extended\_boston()
10. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)
11. lr = LinearRegression().fit(X\_train, y\_train)
13. # w are stored in the coef\_ attribute,
14. # b is stored in the intercept\_ attribute.
15. **print**("lr.coef\_:{}".format(lr.coef\_))
16. **print**("lr.intercept\_:{}".format(lr.intercept\_))
18. **print**("Traing set score: {:.2f}".format(lr.score(X\_train,y\_train)))
19. **print**("Test set score:{:.2f}".format(lr.score(X\_test,y\_test)))

$ python linear\_regression2.py

lr.coef\_:[-4.02752236e+02 -5.00710011e+01 -1.33316908e+02 -1.20021023e+01

-1.27106873e+01 2.83052585e+01 5.44920273e+01 -5.17339345e+01

2.52602789e+01 3.64990704e+01 -1.01038605e+01 -1.96288897e+01

-2.13677312e+01 1.46473758e+01 2.89505391e+03 1.51026852e+03

1.17995400e+02 -2.65658068e+01 3.12488451e+01 -3.14463610e+01

4.52535825e+01 1.28349557e+03 -2.24600306e+03 2.22198614e+02

-4.66264354e-01 4.07661992e+01 -1.34357879e+01 -1.90960479e+01

-2.77605103e+00 -8.09710327e+01 9.73141885e+00 5.13324280e+00

-7.87928926e-01 -7.60269229e+00 3.36717627e+01 -1.15051345e+01

6.62672250e+01 -1.75632324e+01 4.29826712e+01 1.27662043e+00

6.09633677e-01 5.71868313e+01 1.40823103e+01 5.53404218e+01

-3.03481907e+01 1.88121756e+01 -1.37772336e+01 6.09790464e+01

-1.25792000e+01 -1.20021023e+01 -1.76980605e+01 -3.40279371e+01

7.15036708e+00 -8.41017086e+00 1.69857259e+01 -1.29412471e+01

-1.18060747e+01 5.71334581e+01 -1.75814680e+01 1.69557926e+00

2.72177388e+01 -1.67445867e+01 7.50299805e+01 -3.02717569e+01

4.77801366e+01 -4.05414199e+01 5.50445125e+00 2.15309336e+01

2.53662918e+01 -4.94853071e+01 2.81089236e+01 1.04685421e+01

-7.15589798e+01 -2.37402020e+01 9.57368270e+00 -3.78757608e+00

1.21433995e+00 -4.71953075e+00 4.12379498e+01 -3.77021533e+01

-2.15605195e+00 -2.62959500e+01 -3.32015181e+01 4.59315483e+01

-2.30143466e+01 -1.75145726e+01 -1.40847792e+01 -2.04896476e+01

3.65253034e+01 -9.48974963e+01 1.43233527e+02 -1.56740667e+01

-1.49732134e+01 -2.86130369e+01 -3.12520263e+01 2.45648366e+01

-1.78047541e+01 4.03508210e+00 1.71067619e+00 3.44735201e+01

1.12185643e+01 1.14301989e+00 3.73717124e+00 3.13846206e+01]

lr.intercept\_:31.6451741008

Traing set score: 0.95

Test set score:0.61

分析：

1. 对于训练集，我们预测得非常准确，然而对于测试集上的,却非常差！

2. 上述差别说明：过拟合了！　怎么解决呢？在例4中介绍.

* 1. 岭回归(Ridge回归)

Ridge回归也是一种线性回归. 这个方法在OLS基础上，加入了额外的约束条件.

另外，我们也想让系数的绝对值尽可能的小，即w的所有元素都应该接近０．

要求每个特征尽可能少地影响输出结果，也就是说，斜率要尽可能地小,但是预测要尽可能准确！

加这个约束条件是正则化（regularization）的一个例子. 正则化的含义就是：约束一个模型以避免过拟合.

例3.linear\_model.Ridge实现Ridge回归.

1. **import** mglearn
2. **import** matplotlib.pyplot as plt
4. **from** sklearn.model\_selection **import** train\_test\_split
5. **from** sklearn.linear\_model **import** Ridge
7. X, y = mglearn.datasets.load\_extended\_boston()
9. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)
10. ridge = Ridge().fit(X\_train, y\_train)
12. # w are stored in the coef\_ attribute,
13. # b is stored in the intercept\_ attribute.
14. **print**("ridge.coef\_:{}".format(ridge.coef\_))
15. **print**("ridge.intercept\_:{}".format(ridge.intercept\_))
17. **print**("Traing set score: {:.2f}".format(ridge.score(X\_train,y\_train)))
18. **print**("Test set score:{:.2f}".format(ridge.score(X\_test,y\_test)))

结果如下：

$ python l2\_regression.py

ridge.coef\_:[-1.45195306e+00 -1.55625501e+00 -1.45850339e+00 -1.28253037e-01

-8.52751992e-02 8.32260468e+00 2.54137713e-01 -4.94126282e+00

3.90318054e+00 -1.05389293e+00 -1.58274105e+00 1.02803317e+00

-4.01360137e+00 4.36959418e-01 3.61745975e-03 -8.74003709e-01

7.45133472e-01 -1.48860902e+00 -1.67522332e+00 -1.44622038e+00

-5.55323714e-02 -1.78550090e+00 -1.50277930e+00 -1.36385715e+00

-1.59141256e+00 -5.34745744e-01 2.62135084e+00 -2.09149627e+00

1.96201591e-01 -2.77151091e-01 5.11228006e+00 -1.67212612e+00

-9.62022116e-02 6.33380690e-01 -6.07839844e-01 3.96903030e-02

-1.27556962e+00 -2.91333873e+00 3.39477673e+00 7.91219273e-01

1.35663683e+00 -4.03862042e+00 2.33027993e+00 -3.37422177e+00

1.82152566e+00 3.01788004e+00 -1.89823583e+00 -2.59921330e-01

-2.89472573e+00 -1.28253037e-01 -4.99749494e+00 -2.43474155e+00

2.84630673e+00 -8.55131396e-01 2.99363968e+00 2.34736077e+00

1.31329410e+00 1.71670743e+00 -2.59529805e+00 -1.33463017e+00

-2.82980121e+00 -2.09910281e+00 -1.08965503e+00 -2.76728239e+00

-1.63849858e+00 -2.82865852e+00 9.93164278e-01 -1.67575965e+00

1.66517202e+01 -1.12359092e+00 2.14249973e+00 -8.05329870e+00

-8.60057162e+00 -7.54566873e+00 1.03245956e+01 -7.99733592e+00

7.65959189e-01 -1.85682977e+00 2.52146688e+00 -3.38686719e-01

-1.79628860e+00 -2.83181398e-01 -4.78125653e+00 8.70270775e-01

4.06784998e-01 -1.42740707e+00 -2.14509207e-01 -5.08527518e+00

-5.55394560e-01 1.54949256e+00 1.82298709e+00 1.97806765e+00

1.82834358e+00 -7.13688488e+00 1.10265579e+00 1.42155827e+00

-1.31291756e+00 -6.76976345e+00 1.82608210e+00 -2.35756673e+00

3.45818300e-02 1.19002351e+00 -6.29851736e+00 1.03651240e+01]

ridge.intercept\_:21.4172241839

Traing set score: 0.89

Test set score:0.75

分析：

1. 与我们期望的相同，训练集的`$R^2$`变小了，但测试集的变大了！

2. Ridge回归是一个更加restrict的模型.

3. 更简单的模型意味着对训练集的预测力更差了，但是是更好的归纳模型！ 模型的简单性和性能之间存在一个平衡关系需要调节. 可以用Ridge()函数中的alpha参数来调节这个平衡.默认情况下，alpha=1.0.

4. 我们应该选择用Ridge回归模型，而不是OLS模型.

例4.如果在上例中，我们取alpha=0.01.则结果为：

Traing set score: 0.94

Test set score:0.70

注意：

1. 我们尽量将alpha的值取得小些，可以使得模型的归纳性变得更好。

* 1. Lasso回归

Lasso回归(Least Absolute Shrinkage and Selection Operator regression)是正规化线性回归的另一种. 与岭回归一样,该方法也是添加一个正则项到损失函数(cost function); 与岭回归模型不同的是，在Lasso回归模型中，我们用的是L1正规化.

例5. Lasso回归.

1. ＃lasso\_regression.py
2. **import** mglearn
3. **import** matplotlib.pyplot as plt
5. **from** sklearn.model\_selection **import** train\_test\_split
6. **from** sklearn.linear\_model **import** Lasso
8. X, y = mglearn.datasets.load\_extended\_boston()
10. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)
11. lasso = Lasso().fit(X\_train, y\_train)
13. # w are stored in the coef\_ attribute,
14. # b is stored in the intercept\_ attribute.
15. **print**("lasso.coef\_:{}".format(lasso.coef\_))
16. **print**("lasso.intercept\_:{}".format(lasso.intercept\_))
18. **print**("Traing set score: {:.2f}".format(lasso.score(X\_train,y\_train)))
19. **print**("Test set score:{:.2f}".format(lasso.score(X\_test,y\_test)))

结果为：

$ python lasso\_regression.py

lasso.coef\_:[-0. 0. -0. 0. -0. 0.

-0. 0. -0. -0. -0. 0.

-5.3529079 -0. 0. -0. 0. -0.

-0. -0. -0. -0. -0. -0.

-0. -0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. -0. 0. -0. -0.

-0. -0. -0. -0. -0. -0.

-0. 0. 0. 0. 0. 0.

0. 0. 0. 0. -0. -0.

-0. -0. -0. -0. -0. -0.

-0. -0. 0. 0. 0. -0.

-0. -0. 0. -0. -0. 0.

-0. -1.05063037 -3.3104274 -0. -0. 0.

-0. -0. -0. 0. -0. -0.41386744

-0. -0. -0. -0. -0. -0.

-0. -0. -0. -0. -0. 0.

-0. -0. ]

lasso.intercept\_:26.124530145

Traing set score: 0.29

Test set score:0.21

从结果分析:

1.　共有105个特征，我们只用到４个特征。

2.　无论对训练集还是对测试集，R^2值都极其低！

3. 意味着欠拟合！

4. Lasso()有正规化参数alpha. 其默认值alpha=1.0. alpha的值描述将系数推向０的力度.

5. 为了减少欠拟合的影响，我们增加另一个参数max\_iter之值. mat\_iter的值为运行时迭代的次数之最大值.

例6．Lasso回归：控制参数。

1. ＃lasso001\_regression.py
2. **import** mglearn
3. **import** matplotlib.pyplot as plt
5. **from** sklearn.model\_selection **import** train\_test\_split
6. **from** sklearn.linear\_model **import** Lasso
8. X, y = mglearn.datasets.load\_extended\_boston()
10. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)
11. lasso = Lasso(alpha=0.01,max\_iter=10000).fit(X\_train, y\_train)
13. **print**("lasso.coef\_:{}".format(lasso.coef\_))
14. **print**("lasso.intercept\_:{}".format(lasso.intercept\_))
16. **print**("Traing set score: {:.2f}".format(lasso.score(X\_train,y\_train)))
17. **print**("Test set score:{:.2f}".format(lasso.score(X\_test,y\_test)))

运行并显示结果:

$ python lasso001\_regression.py

lasso.coef\_:[ -0. -0. -0. 0. -0.

0. -0. -1.30683918 10.94664016 0.

0. 0. -0.31532732 -0. -0.

-0. 0. -0. -0. -0.

-0. -8.91227147 -0. -0. -0.

-0. 2.09533733 -0. 0. -0.

0. -0. 0. 0. -0.

0. -0. -0. 0. 0.

0. -0. 0. -4.08249984 0.

6.66787706 -0. -0. -0. 0.

-4.40104807 -2.10548411 3.77595291 -0. 4.38381849

0. 0. 0.18662246 -0. -1.17594281

-4.29006257 -0. -0. -2.2160665 -0.

-1.8820987 -0. -0. 29.74050877 -2.08168251

0. -12.00134952 -11.14786769 -11.66008855 13.11691565

-11.15372311 -0. -0. 3.47562768 0.

-0. -0. -8.56630681 0. 0.

-0. 0. -7.38806253 -0. 0.

1.00758109 0. 0. -7.590696 1.61168692

0. 0. -17.43126251 0. 0.

-0. 0.28683048 -8.1576745 17.49647215]

lasso.intercept\_:20.481841374

Traing set score: 0.90

Test set score:0.77

分析：

1. alpha=0.01给出了更好的结果：　无论对于训练集还是测集，分数都更高了。

2. 用到了更多的特征！

3.注意：不要把alpha的值设得过低！

* 1. 小结

1. 通常Ridge是你做线性回归时的首选.

2. 如果你数据集中有大量的特征，而你只希望其中少数几个更重要，那么你就需要尝试Lasso回归模型.

3. Lasso回归模型更容易让我们理解数据.

4. 实际工作中，往往把Ridge回归和Lasso回归结合起来使用.

* 1. 线性分类模型

线性模型也可用于分类问题. 预测值为如下公式：

y = w[0]\*x[0] + w[1]\*x[1] + ... + w[p]\*x[p] +b

线性分类模型的规则如下：

如果 `$y < 0$`, 预测其类为　-1;

如果 `$y > 0$`, 预测其类为　1.

二进制分类器就是用直线，平面或超平面将两类分开的分类器.

例8. 决策边界的确定.

1. #coding:utf-8
2. **import** mglearn
3. **import** matplotlib.pyplot as plt
4. **from** sklearn.linear\_model **import** LogisticRegression
5. **from** sklearn.svm **import** LinearSVC
7. X,y = mglearn.datasets.make\_forge()
8. fig, axes = plt.subplots(1, 2, figsize=(10,3))
9. **for** model, ax **in** zip([LinearSVC(), LogisticRegression()],axes):
10. clf = model.fit(X,y)
11. mglearn.plots.plot\_2d\_separator(clf, X, fill=False, eps=0.5, ax=ax, alpha=.7)
12. mglearn.discrete\_scatter(X[:, 0], X[:,1], y, ax=ax)
13. ax.set\_title("{}".format(clf.\_\_class\_\_.\_\_name\_\_))
14. ax.set\_xlabel("F0")
15. ax.set\_ylabel("F1")
16. axes[0].legend()
17. plt.show()

分析：

1. F0: 第一个特征； F1:第二个特征.

2. 图形显示了分别用两种算法确定的直线边界

3. 任意新数据点落在上方，将被归为第一类，反之则被归为第二类. 两种方法所得的边界相似.

1. 课程总结
   1. 重点
      * 1. 线性模型的数学原理.
        2. 线性回归的两种基本方法(Ridge和Lasso)
   2. 难点

1.选择最合适的alpha,max\_iter值,以平衡线性模型的复杂度与准确度.

* 1. 课后练习

1.假设我们已经有一个在训练集上表现出过拟合的模型,那么它在测试集上的表现将会如何?

1. 在测试集上它将至少表现得和训练集一样好
2. 在测试集上它将表现得和训练集一样好
3. 在测试集上它将至少表现得更差

2.我们为什么要将一个数据集分成训练集和测试集?

1. 通过分离数据,我们可以训练出两个模型
2. 通过分离数据,使得模型的拟合更加快速
3. 通过估算我们的模型在新数据上的表现,我们可以看到模型是否具有普遍性

3.下面是将几率转换为赔率的函数:

**def** prob\_to\_odds(p):

**if** p <= 0 **or** p >= 1:

**print**("Probabilities must be between 0 and 1.")

**return** p / (1-p)

假设所有的数据都属于两个类别中的一个. 已知一个数据属于第一类别的几率是0.4. 该点属于第二个类别的赔率是多少?

A. 0.67

B. 0.6

C. 1.50

D. 2.50

4.关于分类器对象clf(),有几个非常重要的方法.

如果你想训练模型,该用哪一个?

1. clf.fit()
2. clf.estimate()
3. clf.score()

如果你想计算模型的准确度,该用哪一个方法?

1. clf.fit()
2. clf.estimate()
3. clf.score()

如果你想计算一个数据点位于每一类别的概率,该用哪一个方法?

1. clf.predict\_proba()
2. clf.fit()
3. clf.predict()

如果想知道一个新数据点属于哪一类别,该用哪一个方法?

1. clf.predict\_proba()
2. clf.fit()
3. clf.estimate()
4. clf.predict()
5. 扩展
   * + 1. 了解如何在实际工作中将Ridge回归和Lasso回归结合使用.