同濟大學
TONGJI UNIVERSITY

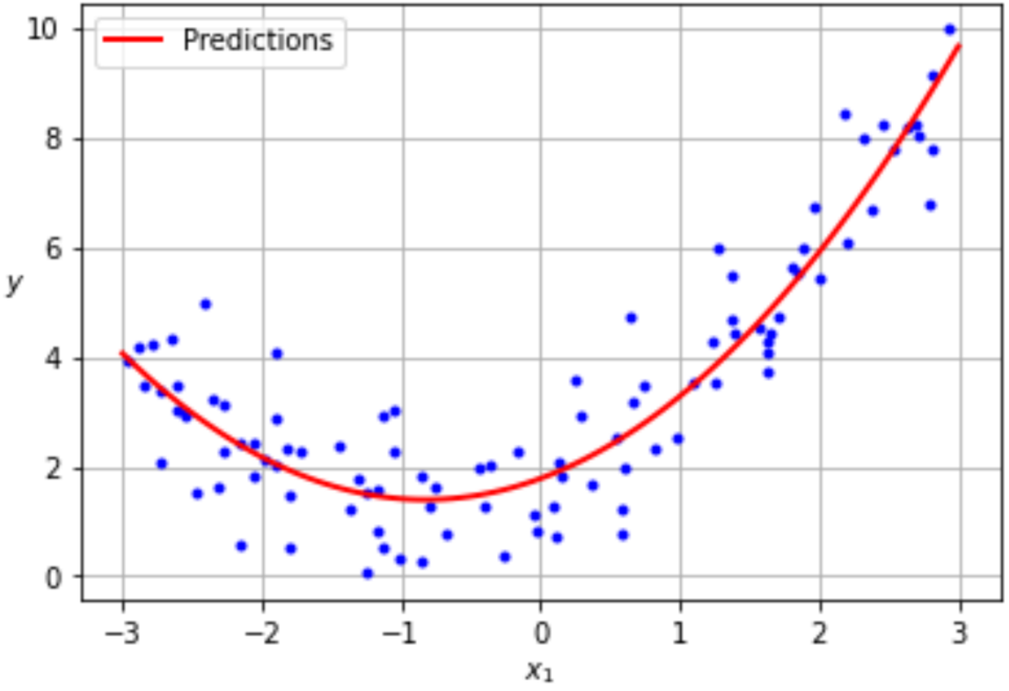
王睿智
ruizhiwang@tongji.edu.cn

人工智能技术与应用

第三章 监督学习

回归与分类

- **回归估计一个连续值**
- **分类预测离散的类别**



MNIST: 手写字符识别（10类）

3.3 Logistic 回归 | 从回归到分类

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

线性回归模型输出 z

寻找一种变换，将线性回归模型输出 z 与分类标签 y 联系起来

$y \in \{1, 0\}$

二分类任务 C_1 , C_2

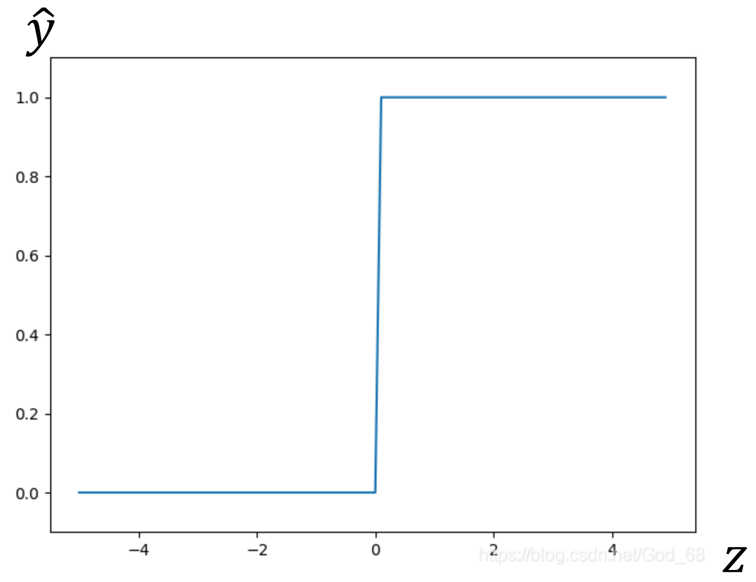
$y = 1$ 对应 C_1 （正类）

$y = 0$ 对应 C_2 （负类）

3.3 Logistic 回归 | 从回归到分类

- **最理想的函数——单位阶跃函数**

$$\hat{y} = \begin{cases} 0, & 若 z < 0 \\ 0.5, & 若 z = 0 \\ 1, & 若 z > 0 \end{cases}$$



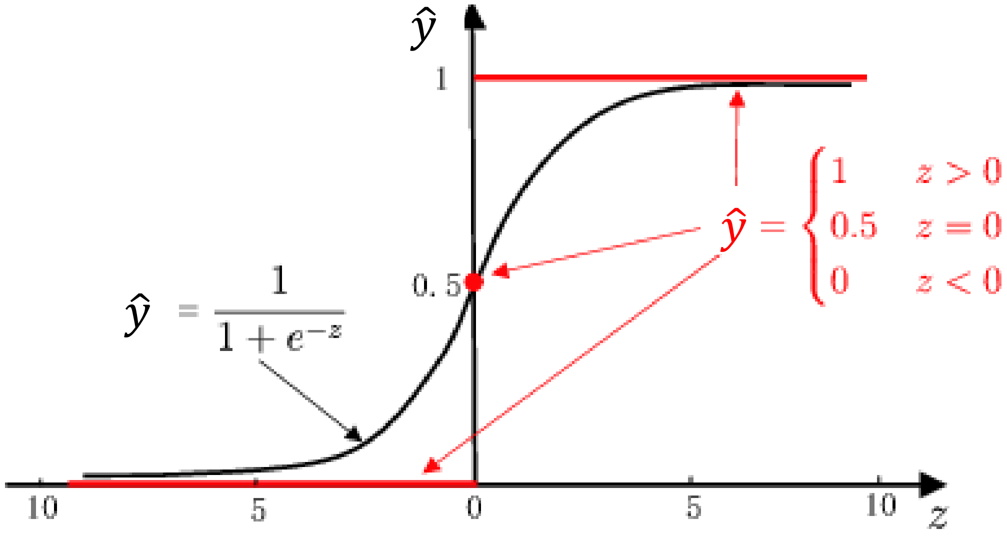
– **预测**：z 大于零判x属于C_1类($\hat{y}$=1)，小于零判为C_2类($\hat{y}$ =0)，若为临界值零则可任意判别。

– **单位阶跃函数缺点**：不连续，不处处可微，没法用梯度下降

3.3.1 Logistic 函数

- **替代函数 —— logistic函数**

$$\hat{y} = \sigma(z) = \frac{1}{1+e^{-z}}$$



单位阶跃函数与logistic函数的比较

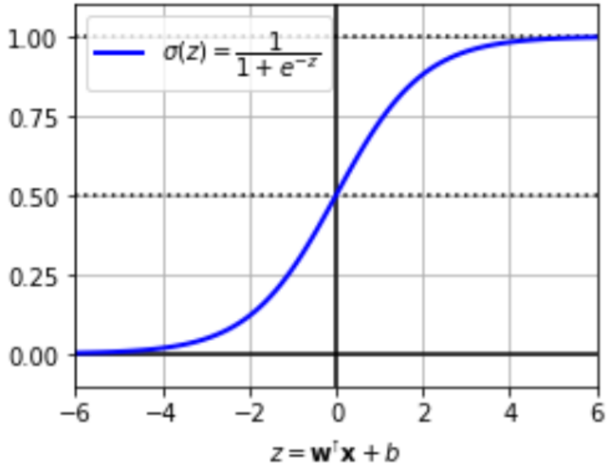
  - 单调可微、任意阶可导
  - logistic 函数是一种sigmoid函数(S型函数)，常记为：$\sigma(\cdot)$
  - Logistic函数把输入值（线性函数的输出）以一种平稳的方式转换为0~1的输出值，该输出值可解释为概率。

3.3.2 Logistic 回归 | 分类规则

如何利用logistic 函数进行类别预测?

若 w 和 b 已确定，样本 x 属于C_1 的概率为

$$P(C_1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b) = \frac{1}{1 + e^{-(\boldsymbol{w}^T\boldsymbol{x}+b)}}$$



如果 $\boldsymbol{w}^T\boldsymbol{x} + b \geq 0 \Rightarrow P(C_1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b) \geq 0.5$，

则预测$\boldsymbol{x}$的类别是1（正类），即 $\hat{y} = 1$；

如果 $\boldsymbol{w}^T\boldsymbol{x} + b < 0 \Rightarrow P(C_1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b) < 0.5$，

则预测$\boldsymbol{x}$的类别为0（负类），即 $\hat{y} = 0$。

3.3.2 Logistic 回归 | 工作原理

机器学习三步曲：

Step1 **函数集**： $P_{\boldsymbol{w},b}(C_1|\boldsymbol{x}) = \sigma\left(\boldsymbol{w}^T\boldsymbol{x} + b\right)$

寻找最优函数（对应最优参数$\boldsymbol{w}$和b），使得模型对真实类别是正类的样本（C_1）输出的$P_{\boldsymbol{w},b}\left(C_1\middle|\boldsymbol{x}^{(i)}\right)$高，而对真实类别是负类的样本（$C_2$）输出的$P_{\boldsymbol{w},b}\left(C_1\middle|\boldsymbol{x}^{(i)}\right)$低的估算。

Step2 **损失函数**：目标真实值 y与输出值 $\hat{y}$ 的**交叉熵**

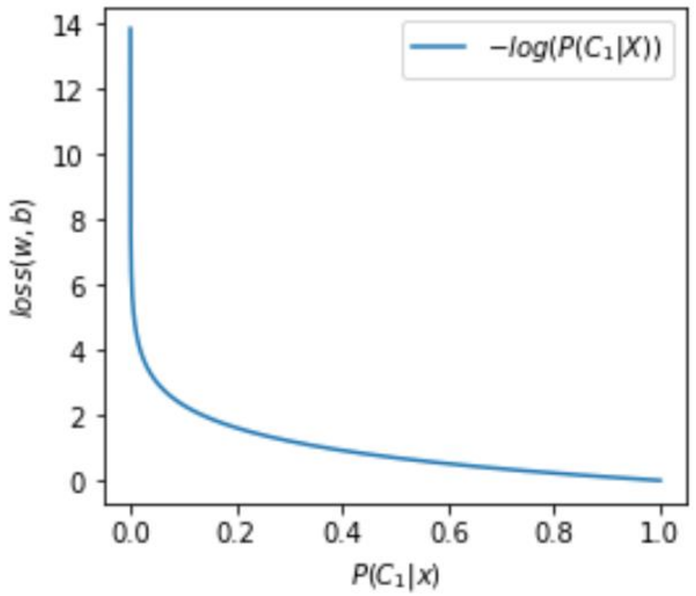
Step3 **优化算法**：梯度下降法，寻找最优参数 $\boldsymbol{w}$ 和 b 。

3.3.2 Logistic 回归 | 逻辑回归的损失函数

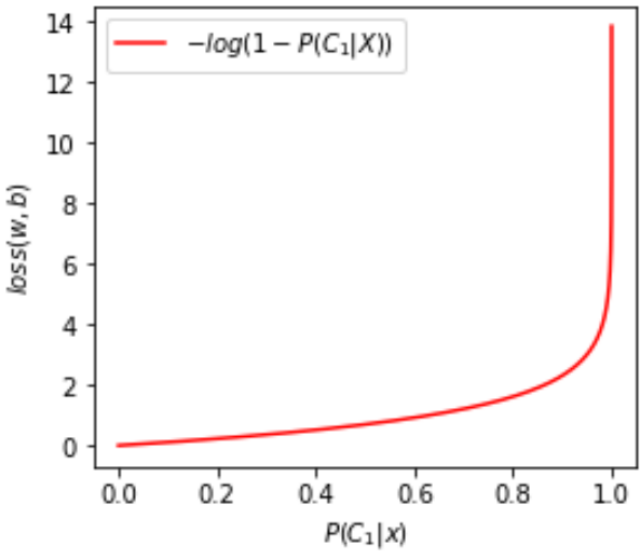
单个训练样本 $x^{(i)}$，其损失可计算如下

$$
\begin{cases}
-\log\left(P_{\boldsymbol{w},b}\left(C_1|\boldsymbol{x}^{(i)}\right)\right), & 若其真实标签y^{(i)}=1 \\
-\log\left(\boxed{1-P_{\boldsymbol{w},b}\left(C_1|\boldsymbol{x}^{(i)}\right)}\right), & 若其真实标签y^{(i)}=0
\end{cases}
$$

$P(C_2|\boldsymbol{x})$



正类样本的情况



负类样本的情况

逻辑回归的损失函数：

整个训练集的损失函数是所有训练样本的**平均损失。** m是样本数目。

设一样本$\boldsymbol{x}^{(i)}$，其真实标签$y^{(i)}$；模型输出值 $\hat{p}^{(i)} = P_{\boldsymbol{w},b}\left(C_1 \middle| \boldsymbol{x}^{(i)}\right)$，$i = 1,2,\cdots m$

$$L(\boldsymbol{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(\hat{p}^{(i)}\right) + \left(1 - y^{(i)}\right)\log(1 - \hat{p}^{(i)})\right]$$

交叉熵

这个函数没有闭式解，

但该函数是凸函数，可以用梯度下降等迭代优化算法，

找到全局最小值。

3.3.2 Logistic回归 | 梯度计算

梯度计算

$$L(\boldsymbol{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(\hat{p}^{(i)}\right) + \left(1 - y^{(i)}\right)\log\left(1 - \hat{p}^{(i)}\right)\right]$$

令 $\boldsymbol{w} \leftarrow \begin{bmatrix} b \\ \boldsymbol{w} \end{bmatrix}$ 是损失函数的参数， $\hat{p}^{(i)} = P_{\boldsymbol{w}}\left(C_1 \middle| \boldsymbol{x}^{(i)}\right) = \sigma\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}\right)$

$$\frac{\partial \boldsymbol{L}(\boldsymbol{w})}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}\left(\underline{\sigma\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}\right) - \boldsymbol{y}^{(i)}}\right)x_j^{(i)}$$

预测值与真实类别的误差

梯度下降法参数更新：

$$\boxed{\boldsymbol{w}_j \leftarrow \boldsymbol{w}_j - \frac{\eta}{m}\sum_{i=1}^{m}\left(\sigma\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}\right) - \boldsymbol{y}^{(i)}\right)x_j^{(i)}}$$

3.3.2 Logistic回归 | 逻辑回归 vs. 线性回归

	Logistic Regression	Linear Regression
Step1 函数集	$f_w(x) = \sigma(\boldsymbol{w}^T\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$ 输出：[0,1]　**分类任务**	$f_w(x) = \boldsymbol{w}^T\boldsymbol{x}$ 输出：任意实数　**回归任务**
Step2 损失函数	训练集 $(\boldsymbol{X}, \boldsymbol{y})$ $y^{(i)}$：1 对于 C_1类，0 对于 C_2 类 $L(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m} C\left(f_w\left(x^{(i)}\right), y^{(i)}\right)$	训练集 $(\boldsymbol{X}, \boldsymbol{y})$ $y^{(i)}$：实数 $L(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m}\left(f_w\left(x^{(i)}\right) - y^{(i)}\right)^2$
Step3 优化算法	$\boldsymbol{w}_j \leftarrow \boldsymbol{w}_j - \frac{\eta}{m}\sum_{i=1}^{m}\left(\sigma\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)}\right) - \boldsymbol{y}^{(i)}\right)x_j^{(i)}$	$\boldsymbol{w}_j \leftarrow \boldsymbol{w}_j - \eta\frac{2}{m}\sum_{i=1}^{m}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)}\right)x_j^{(i)}$

交叉熵：$C\left(f_w\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right) = -\left[y^{(i)}\log\left(f_w\left(\boldsymbol{x}^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - f_w\left(\boldsymbol{x}^{(i)}\right)\right)\right]$

补充：交叉熵（cross-entropy）

定义　交叉熵（cross-entropy）　　信息熵：**H(X) = -∑P(x)log$_2$P(x)**

对于同一个随机变量x有两个单独的概率分布$P(x)$和$Q(x)$，分布P和Q的交叉熵为，

$$H(P,Q) = -E_{x\sim P}\log Q(x) = -\sum_x P(x)\log Q(x)$$

x的真实 (伯努利) 分布P：

$$P(x=1) = y^{(i)}$$

$$P(x=0) = 1 - y^{(i)}$$

交叉熵

$$H(P,Q) = -\sum_x P(x)\log(Q(x))$$

x的假设(伯努利)分布Q:

$$Q(x=1) = f(x^{(i)})$$

$$Q(x=0) = 1 - f(x^{(i)})$$

3.3.2 Logistic 回归 | 优点

logistic回归的优点：

- 直接对类别可能性建模，无需事先假设数据分布，这样就避免了假设分布不准确所带来的问题；
- 它不仅预测出“类别”，而且可得到近似概率预测，这对许多需要利用概率辅助决策的任务很有用；
- logistic函数是任意阶可导的凸函数，有很好的数学性质，现有的许多数值优化算法都可直接用于求取最优解;
- 原用于二分类问题，易扩展到多分类问题。使用Softmax函数
- 原是线性分类模型（分类边界是线性的），易扩展到分类边界为非线性边界的情况。利用核技巧

3.3.3 Scikit-learn中的Logistic回归类

sklearn.linear_model.LogisticRegression类：用于**二分类或多分类**问题

```
LogisticRegression(penalty='l2', C=1.0, max_iter=100, …)
```

主要参数	• *penalty*: 指定惩罚中使用的范数，如*l1,l2,Elastic-Net*。默认值为*l2*。 • *C*：正则化强度的平衡参数，是正则化强度的倒数。Float，默认1.0。C越小，正则强度越大。 • *max_iter* ：最大迭代次数，默认100。 • *multi_class* ：{'auto', 'ovr', 'multinomial'}, 默认'auto'。多分类用multinomial。**若选auto，则自动根据数据类别调整，对二类用ovr，多类用multinomial（此时切换成softmax回归）**。
方法	• fit(X,y) 训练模型，X为特征矩阵；y为目标向量。 • score(X,y) 计算预测的平均精度。 • predict(newX) 预测新数据的类别。返回一维数组(样本数)。 • predict_proba(newX) 预测新数据属于各类别概率。返回二维数组,形状(样本数, 类别数)。
属性	• coef_ 决策函数中的特征系数。**二维数组**。 • intercept_ 截距，float型数。

例3.9 iris数据分类

新建：例3.9 iris数据分类_v1.ipynb

功能：

1. 对iris数据做如下处理：

   取其中“petal length”和“petal width”两个特征列，并根据target的取值是否为“virginica”，将数据集变为目标类别数为2的数据集。

2. 训练一个Logistic回归分类器，并画出决策边界。

代码实现

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import numpy as np
```

1.数据处理

```
iris = load_iris()
X = iris.data[:,2:] # “petal length”和“petal width”
y = (iris.target_names[iris.target] == 'virginica').astype(int)# 两类
```

```
    print(iris.target)  #?
    iris.target_names[0] #?
```

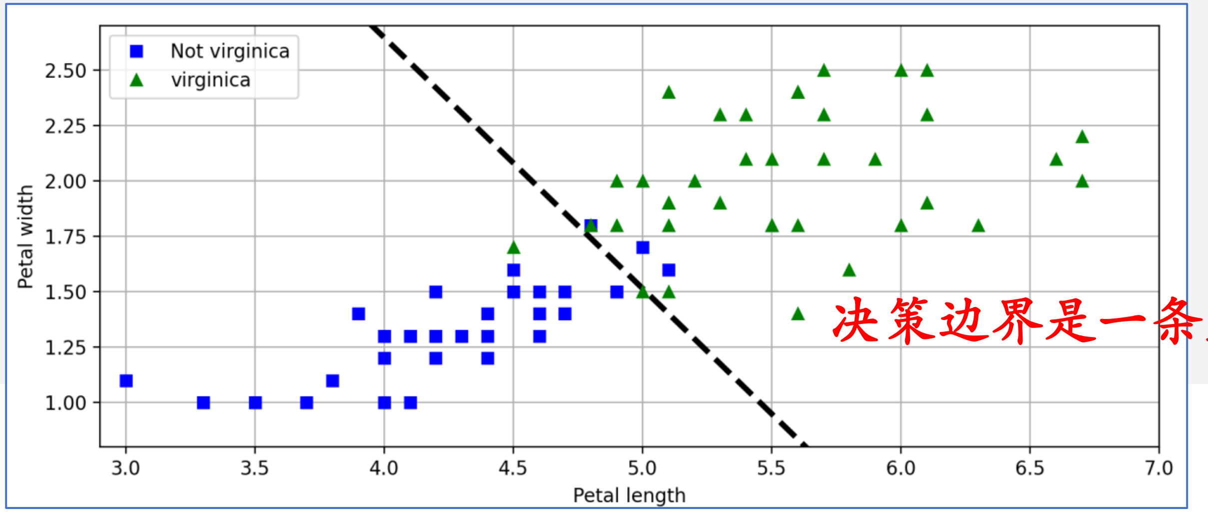
2. 训练Logistic回归分类器，令C=2

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                            random_state=42)
log_reg = LogisticRegression(C=2, random_state=42)
log_reg.fit(X_train, y_train)
```

画出决策边界

$w_0x_0 + w_1x_1 + b = 0 \quad \Rightarrow \quad x_1 = -(w_0x_0 + b)/w_1$

```
%matplotlib notebook
x0_left_right = np.array([2.9, 7]) #取"petal length:2.9和7两个值"作为x轴最小和最大值
x1_top_down = -((log_reg.coef_[0, 0] * x0_left_right + log_reg.intercept_[0])
                 / log_reg.coef_[0, 1]) # "计算petal width对应值"
plt.figure(figsize=(10, 4))
plt.plot(x0_left_right, x1_top_down, "k--", linewidth=3)
plt.plot(X_train[y_train==0, 0],X_train[y_train==0,1],"bs",label="Not virginica")
plt.plot(X_train[y_train==1, 0],X_train[y_train==1,1],"g^",label="virginica")
plt.xlabel("Petal length")
plt.ylabel("Petal width")
plt.legend(loc="best")
plt.axis([2.9, 7, 0.8, 2.7])
plt.grid()
plt.show()
```



决策边界是一条直线

例3.10 乳腺癌预测(1)

新建：例3.10 乳腺癌预测_v1.ipynb

功能：

① 用sklearn的LogisticRegression，采用默认参数的“L2”正则化，强度系数C=1，迭代次数max_iter=10000，构建一个logistic回归分类器，应用到乳腺癌数据集上。训练模型，然后输出模型在训练集和测试集上的准确率得分。

② 尝试调整参数C=100、0.01，观察训练得分和测试得分。

③ 比较正则化参数取三个不同的值(1.0,100,0.01)时模型学得的系数，绘制对应的散点图。

数据集

sklearn.datasets包下的cancer数据集

- 威斯康星州乳腺癌数据集(cancer)
  - 每个数据都被标记为“良性”或“恶性”,其任务是基于人体组织的测量数据来学习预测肿瘤是否为恶性。
  - 共569个样本(良性357个，恶性212个)，30个特征。
  - 利用scikit-learn模块的load_breast_cancer函数来加载数据。

```
from sklearn.datasets import load_breast_cancer
cancer=load_breast_cancer() #返回bunch对象
cancer.data.shape # cancer.data是数据矩阵
```

```
(569, 30)
```

① 正则化强度参数C取默认值1, 迭代10000次

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

X=cancer.data
y=cancer.target
X_train,X_test,y_train,y_test=train_test_split(X,y,stratify=y,
                                    random_state=42)

logreg=LogisticRegression(max_iter=10000).fit(X_train,y_train)
print("训练得分: {:.3f}".format(logreg.score(X_train,y_train)))
print("测试得分: {:.3f}".format(logreg.score(X_test,y_test)))
```

```
训练得分: 0.958
测试得分: 0.958
```

stratify=y,划分出来的测试集或训练集中，其类标签的比例同输入的数组中类标签的比例相同。可用于处理不均衡的数据集。

分析：训练准确率得分和测试得分一样，说明欠拟合了。

② 增大C，令其等于100，训练一个更复杂、灵活的模型

C是正则化强度的倒数

```
logreg100=LogisticRegression(C=100,max_iter=10000).fit(X_train,y_train)
print("训练得分: {:.3f}".format(logreg100.score(X_train,y_train)))
print("测试得分: {:.3f}".format(logreg100.score(X_test,y_test)))
```

分析：训练集准确率更高，测试集准确率也略有提高，说明复杂的模型性能更好。

```
训练得分: 0.984
测试得分: 0.965
```

降低C，令其为0.01，使用正则化程度更强的模型，看看怎样？

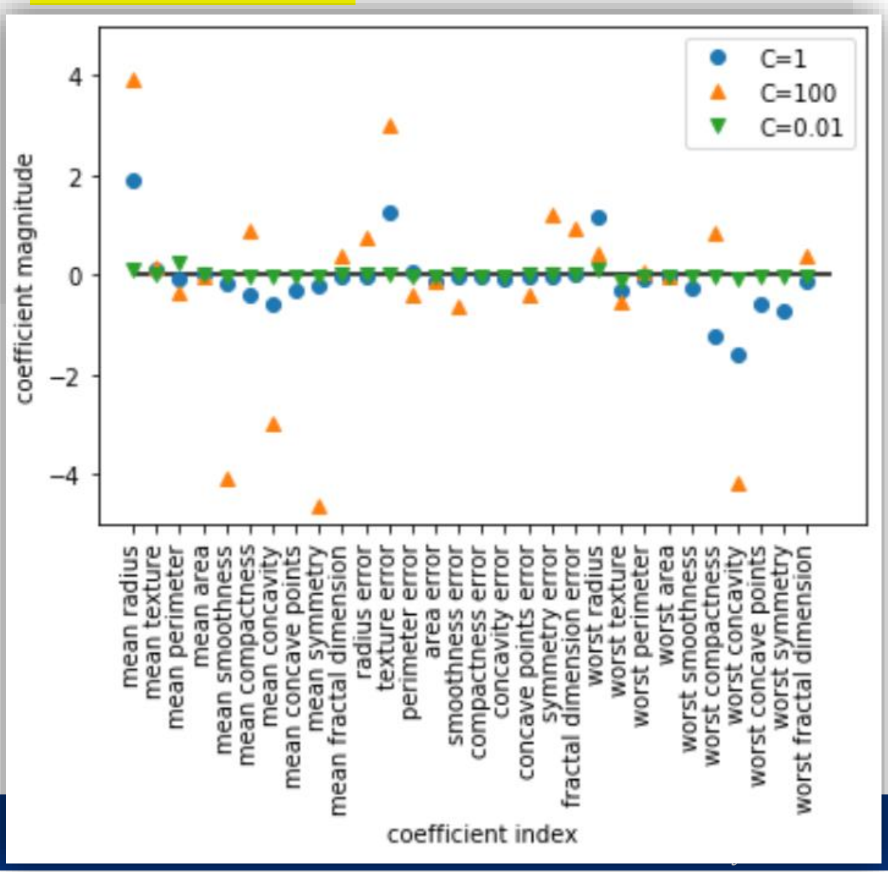
```
logreg001=LogisticRegression(C=0.01,max_iter=10000).fit(X_train,y_train)
print("训练得分: {:.3f}".format(logreg001.score(X_train,y_train)))
print("测试得分: {:.3f}".format(logreg001.score(X_test,y_test)))
```

分析：训练集、测试集的准确率都比默认值还小，说明简单模型的性能不如复杂模型。

```
训练得分: 0.953
测试得分: 0.951
```

③ 比较正则化参数取三个不同的值(1.0,100,0.01)时模型学得的系数

```
import matplotlib.pyplot as plt
plt.plot(*logreg.coef_,'o',label='C=1') #取出二维数组logreg.coef_的元素,一维数组
plt.plot(*logreg100.coef_,'^',label="C=100")
plt.plot(*logreg001.coef_,'v',label="C=0.01")
plt.xticks(range(cancer.data.shape[1]),cancer.feature_names,rotation=90)
plt.hlines(0,0,cancer.data.shape[1]) # 横轴0~30, 纵轴都是0
plt.ylim(-5,5)
plt.xlabel("coefficient index")
plt.ylabel("coefficient magnitude")
plt.legend()
```



函数: plt.hlines(y, xmin, xmax)　绘制水平线
说明: Plot horizontal lines at each *y* from *xmin* to *xmax*.

例3.10 乳腺癌预测(2)

新建： 例3.10 乳腺癌预测_v2.ipynb

功能：

① 观察原数据X，计算各特征的均值（最大与最小）、方差（最大与最小）、取值区间（各区间上下界差）。

② 数据规范化预处理，调用sklearn.preprocessing中的StandardScaler，使得处理后的数据集均值为零，方差为1。

③ 用规范化后的数据，令C分别等于0.1,1,10,100，找到最优C对应的模型。

④ 用最优C重新训练一个新模型。预测测试数据所属类别和概率，用DataFrame表示。

⑤ 输出混淆矩阵和分类性能报告。

1.观察数据

```python
import numpy as np
from sklearn.datasets import load_breast_cancer
import pandas as pd

cancer=load_breast_cancer()  # bunch对象

X=cancer.data # 形状(569, 30)
y=cancer.target # 形状(569,)
```

```python
features_mean = [X.mean(axis=0)] # 各列的均值
features_var = [X.var(axis=0)]
features_scale =[X.max(axis=0)-X.min(axis=0)]

df = pd.DataFrame({'特征均值':[np.min(features_mean),np.max(features_mean)],
                   '特征方差':[np.min(features_var),np.max(features_var)],
                   '特征区间范围':[np.min(features_scale),np.max(features_scale)]},
                  index = ['最小','最大'])
df
```

	特征均值	特征方差	特征区间范围
最小	0.003795	0.003795	0.028945
最大	880.583128	880.583128	4068.800000

2.数据规范化预处理

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train,X_test,y_train,y_test=train_test_split(X,y,stratify=y,random_state=42)

scaler = StandardScaler()
Xtrain_new=
Xtest_new = 
```

3.用规范化后的训练集数据训练模型

```
from sklearn.linear_model import LogisticRegression
models = []
for i in [0.1,1,10,100]:
    logreg=LogisticRegression(C=i,max_iter=1000).fit(Xtrain_new,y_train)
    print("C={},训练得分：{:.3f}".format(i,logreg.score(Xtrain_new,y_train)))
    print("C={},测试得分：{:.3f}\n".format(i,logreg.score(Xtest_new,y_test)))
    models.append((i,logreg))
```

```
C=0.1,训练得分：0.984
C=0.1,测试得分：0.979

C=1,训练得分：0.988
C=1,测试得分：0.986

C=10,训练得分：0.991
C=10,测试得分：0.972

C=100,训练得分：0.995
C=100,测试得分：0.951
```

分析：与例3.10 乳腺癌预测_v1.ipynb中的模型相比，该模型的训练集精度更高，测试集精度也略有提高，说明数据预处理后效果更好。

4.用最优C=1重新训练一个新模型，预测测试数据所属类别和概率

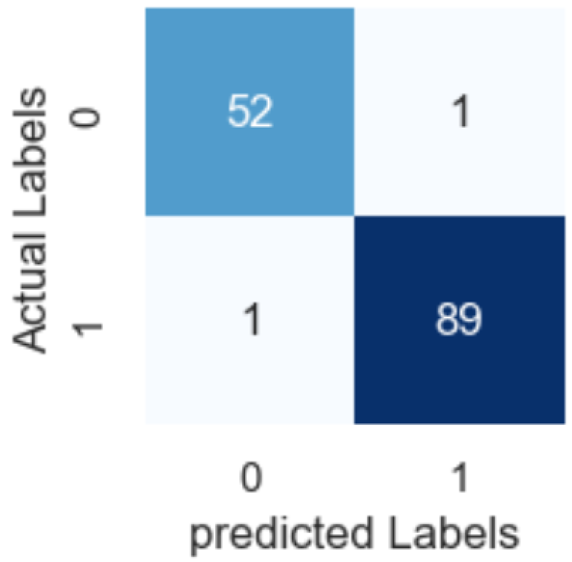
```python
best_model=LogisticRegression(C=1,max_iter=1000).fit(Xtrain_new,y_train)

predicted_prob = best_model.                    #生成在测试集上的预测概率
y_pred = best_model.                            #生成在测试集上的预测类别
pred_df = pd.DataFrame({'malignant预测概率':predicted_prob[:,0],
                        'benign预测概率':predicted_prob[:,1],
                        '预测类别':y_pred})
print('0类名:{},1类名:{}'.format(cancer.target_names[0],cancer.target_names[1]))
pred_df.head(5)
```

0类名:malignant,1类名:benign

	malignant预测概率	benign预测概率	预测类别
0	0.030954	0.969046	1
1	0.999652	0.000348	0
2	0.440761	0.559239	1
3	0.060111	0.939889	1
4	0.824483	0.175517	0

5. 使用混淆矩阵评估性能

```
import matplotlib.pyplot as plt
import seaborn as sns;sns.set()
from sklearn.metrics import confusion_matrix

conf_mx = confusion_matrix(                )
plt.figure(figsize=(2,2))
sns.heatmap(conf_mx,annot=True,cmap='Blues',cbar=False)
plt.xlabel('predicted Labels')
plt.ylabel('Actual Labels')
```



生成分类报告

```
from sklearn.metrics import classification_report
print(classification_report(                ))
```

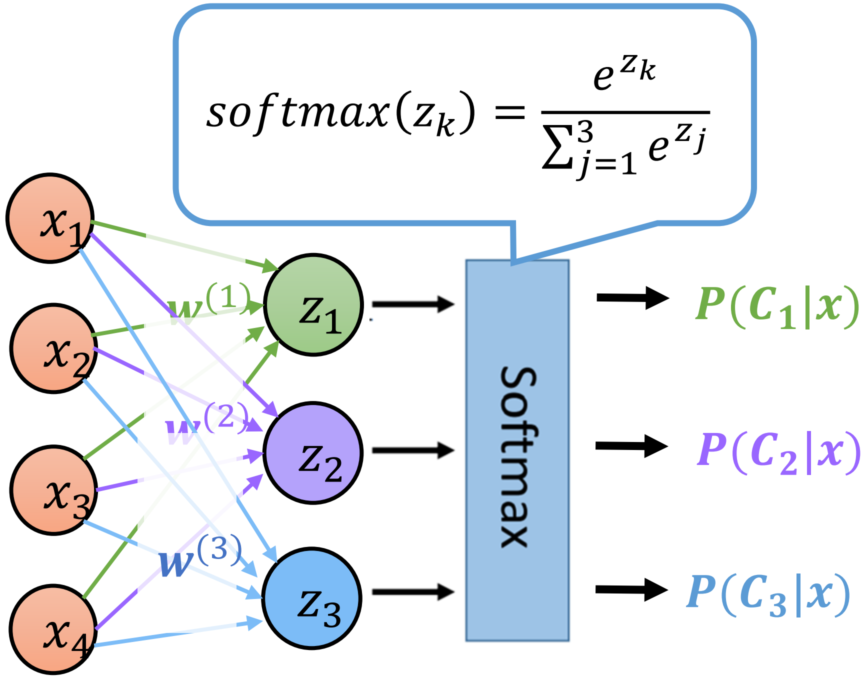
```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        53
           1       0.99      0.99      0.99        90

    accuracy                           0.99       143
   macro avg       0.99      0.99      0.99       143
weighted avg       0.99      0.99      0.99       143
```

3.3.4 Softmax回归

3.3.4 Softmax 回归 | Softmax函数

Softmax函数：

$$softmax(z_k) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$



- 用 Softmax 函数对 $z_k = (\boldsymbol{w}^{(k)})^T\boldsymbol{x}$, $k = 1,2, \ldots K$ 作变换, 结果为$K(K \geq 3)$个不同类上的概率分布。这种方法称为**softmax回归。**
- Softmax回归是Logistic回归的一般形式。Logistic回归是k=2时的Softmax回归。
- Logistic回归用于二分类，softmax回归用于多分类。

3.3.4 Softmax回归 | 工作原理

- **机器学习三步曲**：
  - Step1 **函数集**： $\hat{y}_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$ ， $z_k = (\boldsymbol{w}^{(k)})^T \boldsymbol{x}\ ,\ k = 1,2, \ldots K$ (式1)
  - Step2 **损失函数**：目标真实值 y 与输出值 $\hat{y}$ 的交叉熵，

$$L(\boldsymbol{W}) = L(y, \hat{y}\,) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log \hat{y}_k^{(i)} \qquad \boldsymbol{W} = (\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}, \cdots, \boldsymbol{w}^{(K)})$$

  - Step3 **优化算法**：梯度下降法，寻找最优参数 $\boldsymbol{w}^{(k)}$, $k = 1,2, \ldots K$ 。
- **分类规则**：确定 $\boldsymbol{w}^{(k)}$ 后，对一个输入 x，代入(式1)中，取最大输出值对应的类标签作为 x 类别。

梯度计算

交叉熵损失函数:

$$L(\boldsymbol{W}) = L(y, \hat{y}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log \hat{y}_k^{(i)}$$

$$\boldsymbol{W} = (\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}, \cdots, \boldsymbol{w}^{(K)})$$

m个样本(X,y) 梯度计算公式:

$$\frac{\partial L}{\partial \boldsymbol{w}^{(k)}} = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{\boldsymbol{y}}_k^{(i)} - y_k^{(i)}\right)\boldsymbol{x}^{(i)}$$

其中 $\hat{y}_k^{(i)} = softmax\left(z_k^{(i)}\right) = \frac{e^{z_k^{(i)}}}{\sum_{j=1}^{K} e^{z_j^{(i)}}}$， $z_k^{(i)} = (\boldsymbol{w}^{(k)})^T \boldsymbol{x}^{(i)}$， $k = 1,2, \ldots K$

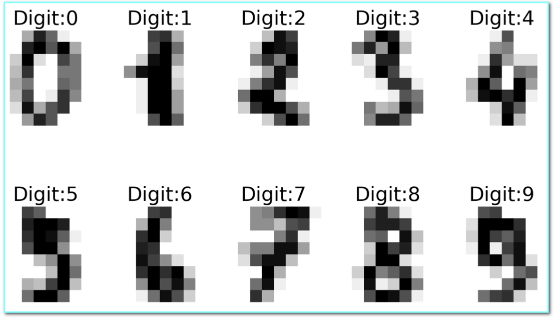
例3.11 Softmax手写字符识别　　多类情况

Scikit-learn自带的手写数字图片数据集有1797个样本，每个样本是8*8尺寸的灰度图片。

(1) 用datasets.load_digits() 载入数据，显示前10个数字的图像。

(2) 数据集分成两部分，80%样本用于训练，20%留做测试。 在训练集训练一个**softmax回归分类器**。

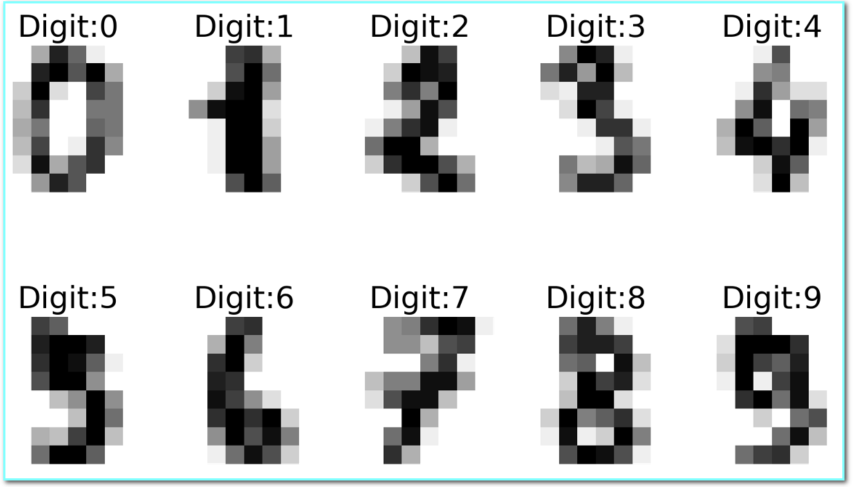
(3) 输出分类器在测试集上的得分，以及分类性能报告。



模型建立过程：

1.读取数据

```
from sklearn import datasets
from matplotlib import pyplot as plt

digits=datasets.load_digits()

# 显示数字所代表的图片
images_and_labels=list(zip(digits.images,digits.target))
plt.figure(figsize=(10,6),dpi=200)
for index,(image,label)in enumerate(images_and_labels[:10]):
    plt.subplot(2,5,index+1)
    plt.axis('off')
    plt.imshow(image,cmap=plt.cm.gray_r,interpolation='nearest')
    plt.title('Digit:%i'%label,fontsize=20)
```



2.创建并训练Softmax模型

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
Xtrain,Xtest,Ytrain,Ytest=train_test_split(digits.data,digits.target,
                                            test_size=0.2,random_state=42);

# 默认multi_class='auto'自适应类别数
clf = LogisticRegression(C=0.01, random_state=42,max_iter=10_000)
clf.fit(Xtrain, Ytrain)
```

3.评估模型

```
print("在测试集上的准确度为：{:.3f}".format(clf.score(Xtest,Ytest)))
print("在训练集上的准确度为：{:.3f}".format(clf.score(Xtrain,Ytrain)))
```

```
训练得分：0.991
测试得分：0.975
```

输出分类性能报告

```
from sklearn.metrics import classification_report

Y_pred = clf.predict(Xtest)
print(classification_report(Ytest,Y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.97      1.00      0.98        28
           2       0.97      1.00      0.99        33
           3       1.00      0.97      0.99        34
           4       1.00      0.98      0.99        46
           5       0.94      0.94      0.94        47
           6       0.97      0.97      0.97        35
           7       1.00      0.97      0.99        34
           8       0.97      0.97      0.97        30
           9       0.95      0.97      0.96        40

    accuracy                           0.97       360
   macro avg       0.98      0.98      0.98       360
weighted avg       0.98      0.97      0.98       360
```

小结

基于Logistic/softmax回归的机器学习算法构建，三步：

- 模型：Logistic函数/softmax函数
- 损失函数：交叉熵
- 优化算法：梯度下降法

作业5　　5.1 逻辑回归鉴别红酒的种类

Scikit-learn内置数据集：wine, 178个样本，13个特征，分3个类别（0、1、2）

要求:

(1) 用softmax回归创建一个分类器，用训练数据训练模型，测试其分类准确率。

输出测试数据属于3个类别的概率、类别，注意要求构成dataframe对象

(2) 对红酒数据进行缩放处理 StandardScaler 。选择合适的超参数(max_iter,C)重建softmax回归分类器，并用缩放后的数据训练模型，评估新模型的测试准确率，输出分类性能报告。

作业提示:

```
from sklearn.datasets import load_wine
wine = load_wine()
```

补充：Logistic回归 与 logit回归

对于两类问题，用Logistic函数可估计样本x的类别概率：

$\boldsymbol{x}$ 属于正类 (C_1) 的概率 $$P(C_1|\boldsymbol{x}) = \frac{1}{1+e^{-(\boldsymbol{w}^T\boldsymbol{x}+b)}}$$

$\boldsymbol{x}$ 属于负类 (C_2) 的概率 $$P(C_2|\boldsymbol{x}) = 1 - P(C_1|\boldsymbol{x}) = \frac{1}{1+e^{(\boldsymbol{w}^T\boldsymbol{x}+b)}}$$

$$\boxed{\log \boxed{\frac{P(C_1|\boldsymbol{x})}{1-P(C_1|\boldsymbol{x})}}} = \log e^{(\boldsymbol{w}^T\boldsymbol{x}+b)} = \boldsymbol{w}^T\boldsymbol{x} + b$$

对数几率函数　　几率

$logit(P(C_1|\boldsymbol{x}))$

- 一事件的几率是指该事件发生的概率p与该事件不发生的概率1-p的比值，即$\frac{p}{1-p}$。
- Logistic函数是对数几率函数 (logit函数) 的反函数。
- logistic回归也称为logit回归。