机器学习

逻辑回归

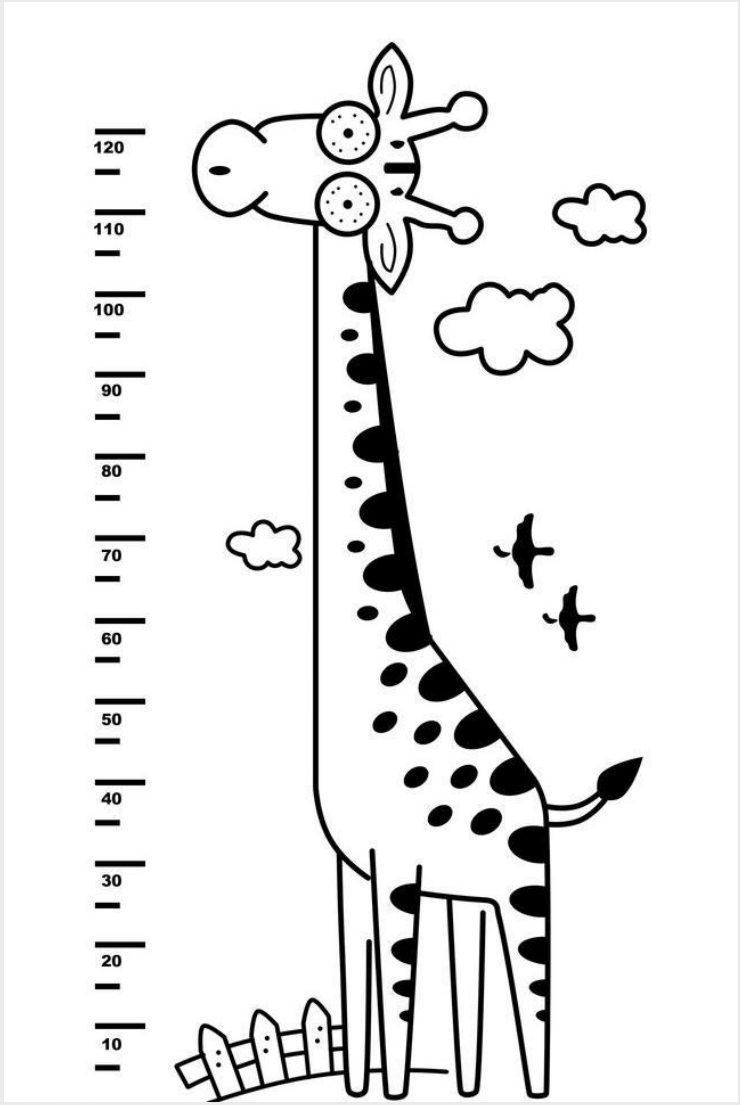
课程目录

Course catalogue

本课目标

- 学习逻辑回归原理
- 学习使用混淆矩阵、ROC、AUC
- 学习使用python解决不平衡集问题
- 学习使用python解决多分类问题
- 学习使用python解决非线性分类问题

逻辑回归原理



逻辑回归：名叫回归的分类算法

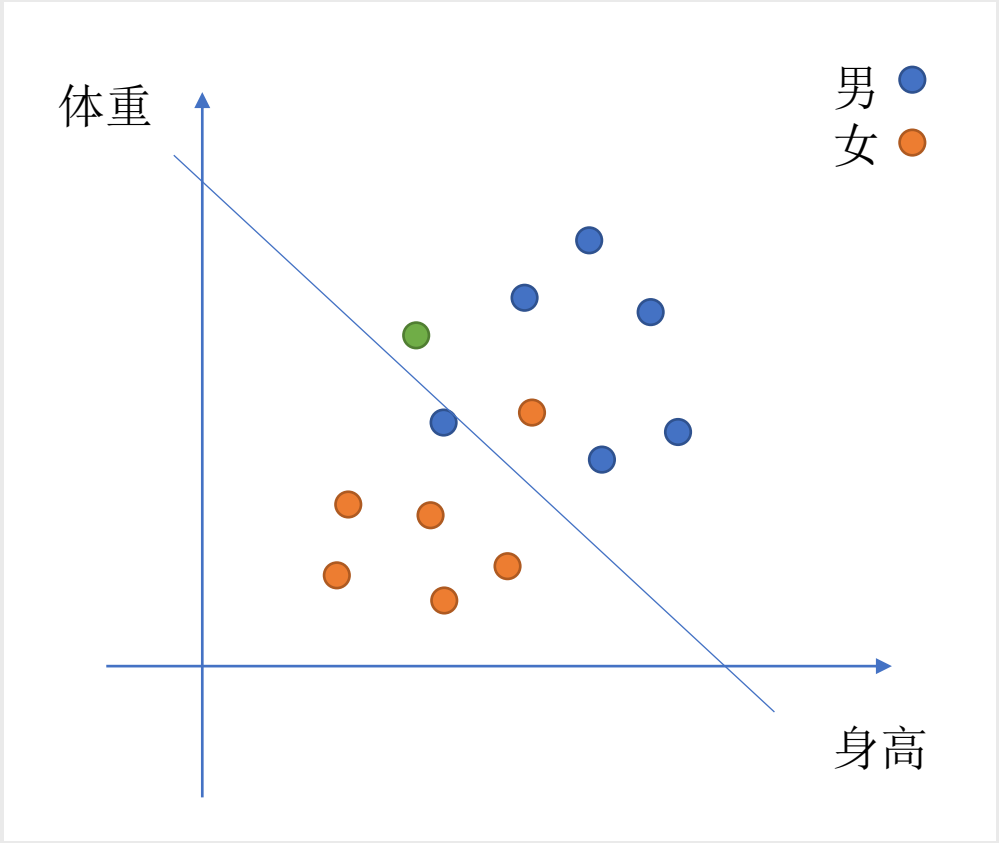
【问题】

一位成年人，身高160厘米，体重45公斤，ta的性别应该是男还是女呢？，如果ta的身高180厘米，体重70公斤呢？为什么？

【思考】

为什么逻辑回归的名字是回归，却可以用来解决分类问题？

逻辑回归原理



算法原理

逻辑回归通过预测属于不同类别的概率，从而实现分类的目的。

粗略来说，逻辑回归就是在找一条直线，这条直线能够尽可能的将不同类别分开。在这条线的一侧，属于某个类别的概率大一些，相对应的，在线的另一侧，则属于另一个类别的概率大一些。

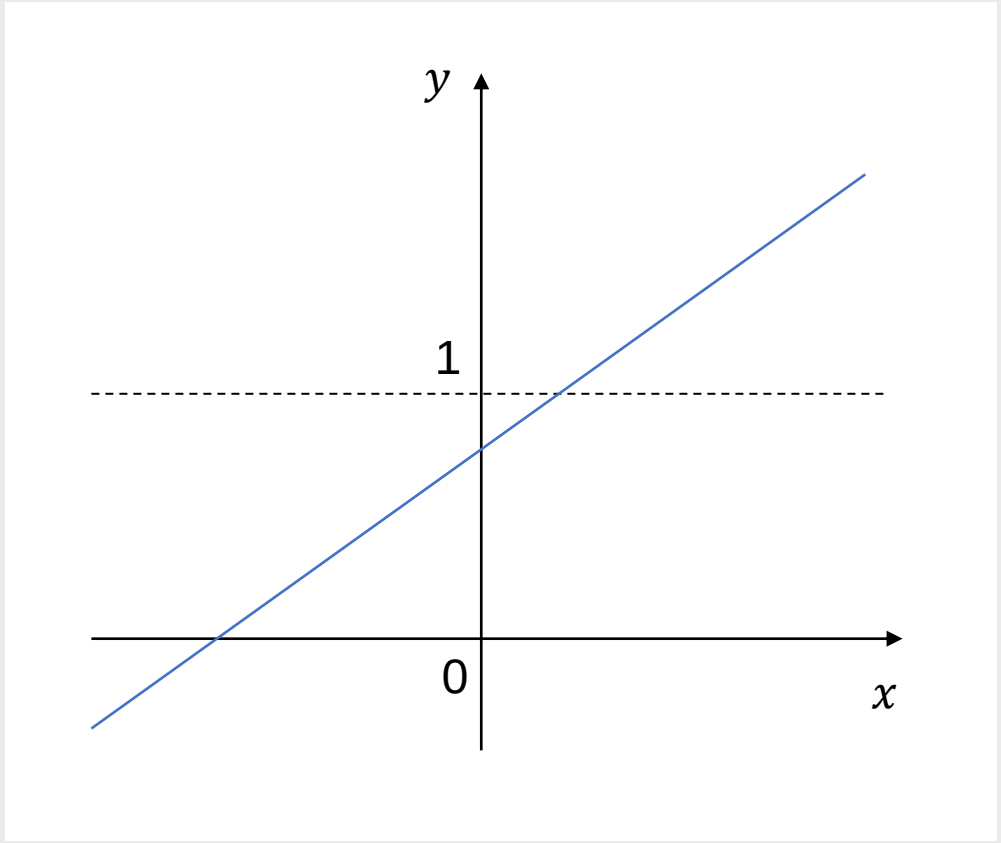
【问题】

绿色样本点是男生还是女生？

【思考】

线性回归模型可不可以用来预测概率？

逻辑回归原理



算法原理

线性回归预测函数：

$$y = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b$$

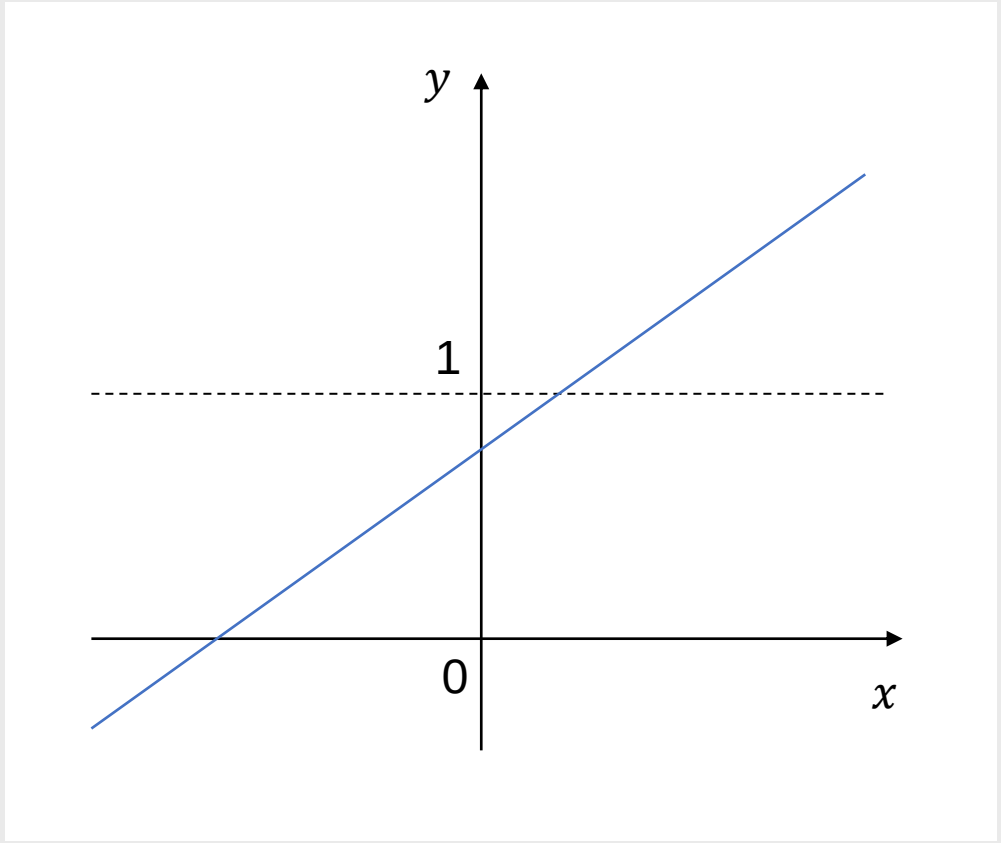
期望的逻辑回归预测函数：

$$prob(y = 1) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b$$

【思考】

逻辑回归预测函数的问题在哪里？有什么解决办法？

逻辑回归原理



算法原理

期望的逻辑回归预测函数：

$$prob(y = 1) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b$$

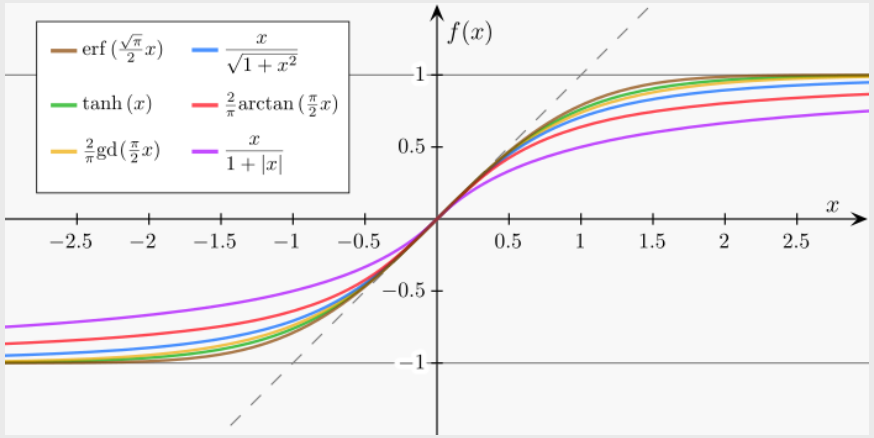
【等式问题】

- 左侧值域 $[0,1]$，右侧值域$(-\infty, +\infty)$

【解决思路】

1. 把左侧值域放大到$(-\infty, +\infty)$
2. 把右侧值域缩小到$[0,1]$

逻辑回归原理



第一种表达方式： $h(x)=\sigma(t)=\frac{1}{1+e^{-t}}$

其中：

$$t=\boldsymbol{w^T x}+b$$

$h(x)$刻画的是属于“1”的概率

思路一：右侧值域缩小到[0,1]

期望的逻辑回归预测函数：

$$P\ (y=1|x)=\boldsymbol{w^T x}+b$$

令 $t=\boldsymbol{w^T x}+b$，则可以通过Sigmoid函数将 t 转换为有限区间，如[0,1]区间，Sigmoid函数并非是某个特定的函数，而是具有S形的数学函数统称，记作$\sigma(t)$

如果没有特殊说明，通常来说，sigmoid函数是指如下形式：

$$\sigma(t)=\frac{1}{1+e^{-t}}$$

逻辑回归原理

第二种表达方式： $\log(odds) = \boldsymbol{w^T x} + b$

其中：

$$odds = p/(1-p)$$

思路二：左侧值域放大到$(-\infty, +\infty)$

期望的逻辑回归预测函数：

$$P\ (y = 1|x) = \boldsymbol{w^T x} + b$$

步骤一，将值域转化为[0,+∞)：

$$odds = p/(1-p)$$

步骤二，将值域转化为(-∞,+∞)：

$$\log(odds)$$

逻辑回归原理

代价函数

第一种“可能”的形式：

$$Cost(h(x), y) = \frac{1}{2}(h(x) - y)^2$$

第二种“可能”的形式：

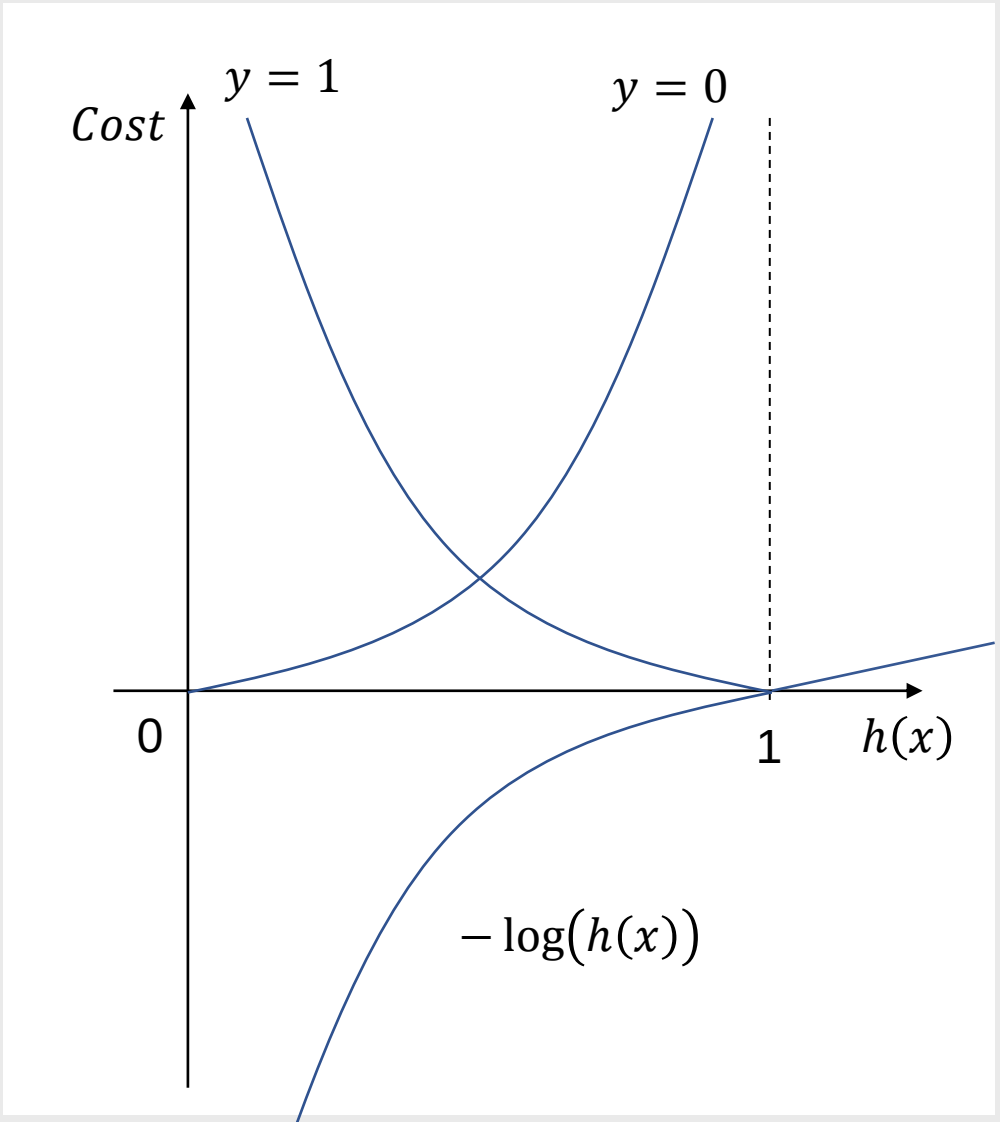
$$Cost(h(x), y) = \begin{matrix} 1 - h(x) & if\ y = 1 \\ h(x) & if\ y = 0 \end{matrix}$$

第三种“可能”的形式：

$$Cost(h(x), y) = \begin{matrix} -\log\big(h(x)\big) & if\ y = 1 \\ -\log\big(1 - h(x)\big) & if\ y = 0 \end{matrix}$$

【思考】第三种形式中，损失函数的图形怎样？

逻辑回归原理



代价函数

第三种“可能”的形式：

$$Cost(h(x),y)=\begin{cases}-\log(h(x)) & if\ y=1\\ -\log(1-h(x)) & if\ y=0\end{cases}$$

情况1：$y=1\ \&\ h(x)=1$，$Cost$取值如何？

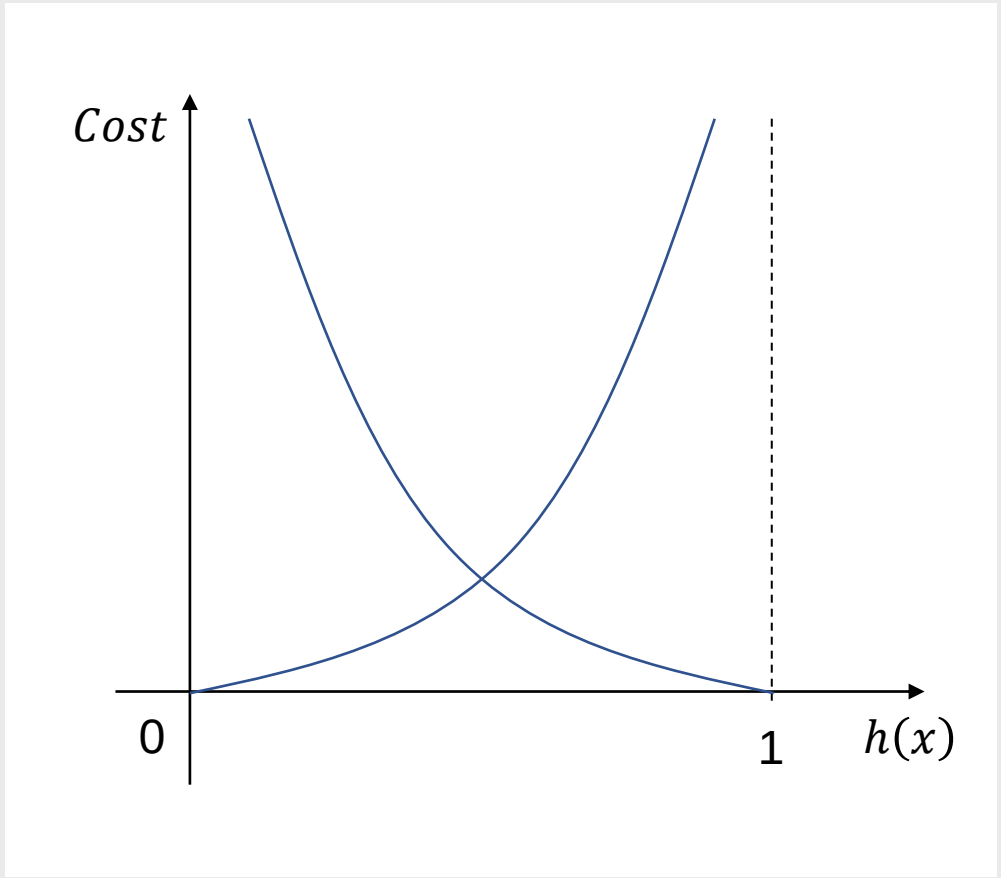
情况2：$y=1\ \&\ h(x)=0$，$Cost$取值如何？

情况3：$y=0\ \&\ h(x)=1$，$Cost$取值如何？

结论：如果模型预测错误，将导致很大的惩罚。

【问题】这种形式存在什么问题？

逻辑回归原理



代价函数

第三种“可能”的形式：

$$Cost(h(x),y)=\begin{cases}-\log(h(x)) & if\ y=1\\ -\log(1-h(x)) & if\ y=0\end{cases}$$

这种形式的代价函数，问题在于不方便求导，因此需要将代价函数合并。

合并后最终的逻辑回归代价函数：

$$Cost=-y\log(h(x))-(1-y)\log(1-h(x))$$

【思考】y=0 和 y=1时，代价函数分别是怎样的？

逻辑回归原理

	petal_l	petal_w	classes
0	1.4	0.2	0
1	1.4	0.2	0
2	1.3	0.2	0

```
# 载入数据
import pandas as pd
df = pd.read_csv('datas/iris_partial.csv',index_col=0)

# 划分自变量和因变量
X = df.loc[:,df.columns != 'classes']
y = df.loc[:,df.columns == 'classes']

# 划分训练集和测试集
from sklearn.model_selection import train_test_split
X_tr,X_ts,y_tr,y_ts = train_test_split(X,y)

# 建立逻辑回归模型
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_tr,y_tr.values.ravel())

# 查看预测结果
model.predict(X_ts)
```

```
array([0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0,
       1, 0, 1], dtype=int64)
```

Python解决方案

读取数据：pandas.read_csv()

Hold-out：sklearn.model_selection.train_test_split

Sklearn三板斧：

建模

model = LogisticRegression()

拟合

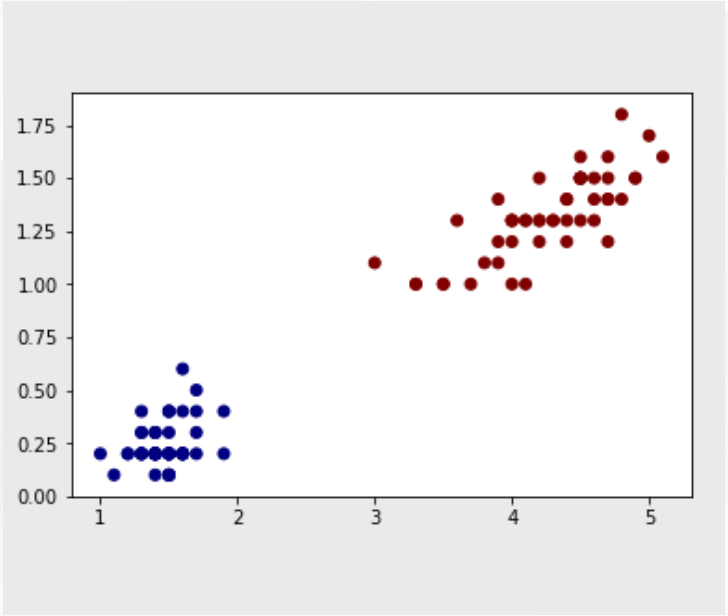
model.fit()

预测

model.predict model.predict_prob

逻辑回归原理

逻辑回归原理的直观理解

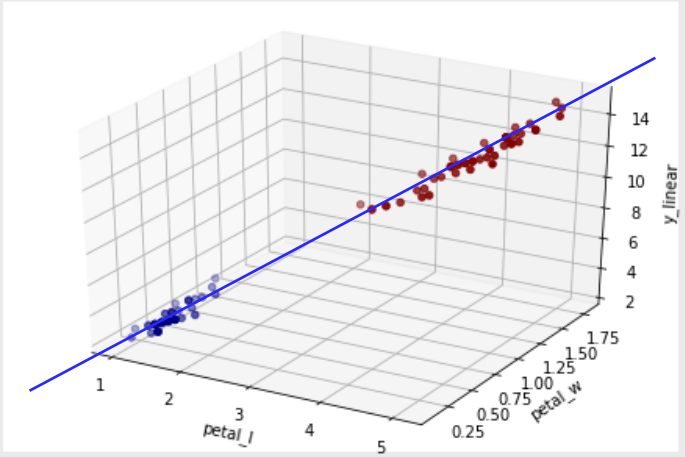


原始数据

横坐标为花瓣长度

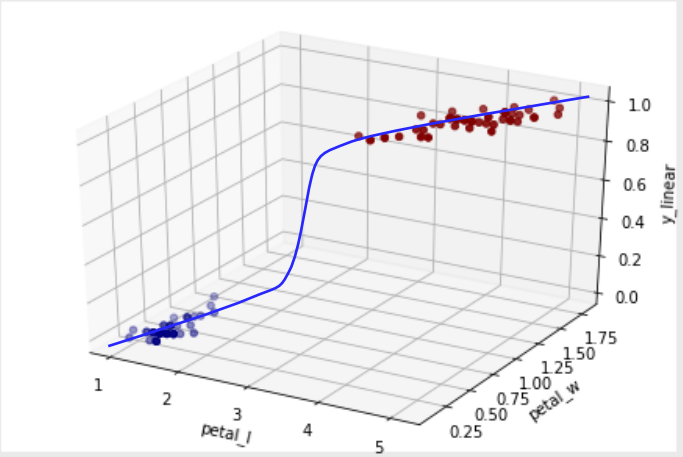
纵坐标为花瓣宽度

颜色划分为花瓣类型



线性变换

找到最优参数，对特征值组合进行线性变换，得到线性组合的“超平面”



Sigmoid变换

对“超平面”进行sigmoid变换，将“超平面”的阈值压缩到（0,1）之间

逻辑回归原理

```
# 将数值型离散变量转换为哑变量
# 只能转换一个特征
from sklearn.preprocessing import OneHotEncoder
oe = OneHotEncoder(sparse=False)
oe.fit_transform(df.var3.values.reshape(-1,1))
```

```
array([[1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.]])
```

```
# 将字符型离散变量转换为序号，隐含序号信息
# 一次只能转换一个特征
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
le.fit_transform(df.var1)
```

```
array([2, 2, 0, 1], dtype=int64)
```

```
# 批量将字符离散变量转换为哑变量
pd.get_dummies(df)
```

	var3	var1_初中	var1_大学	var1_小学	var2_x	var2_y
0	0	0	0	1	1	0
1	0	0	0	1	1	0
2	1	1	0	0	0	1
3	1	0	1	0	0	1

离散变量处理

离散变量转码的三种方式：

One-hot、Label-encoding、get_dummies

One-hot: sklearn.preprocessing. OneHotEncoder

Label-encoding: sklearn.preprocessing. LabelEncoder

Get_dummies: pandas.get_dummies

var1	var2	var3
小学	x	0
小学	x	0
初中	y	1
大学	y	1

课程目录

Course catalogue

模型评价

			预测结果	
			反例 Negative	正例 Positive
			0	1
真实情况	反例 Negatives	0	真反例 True Negatives	假正例 False Positives
	正例 Positives	1	假反例 False Negatives	真正例 True Positives

True Positives(真正例)：实际为P，预测也为P

True Negatives(真反例)：实际为N，预测也为N

False Positives(假正例)：实际为N，预测为P

False Negatives(假反例)：实际为P，预测为N

TP+TN+FP+FN=样例总数

二分类中：样本根据真实类别与预测类别组合分成四类

混淆矩阵

TPR/recall/查全率：$\frac{TP}{TP+FN}$　　FPR：$\frac{FP}{N}$

PPV/Precision/查准率：$\frac{TP}{TP+FP}$

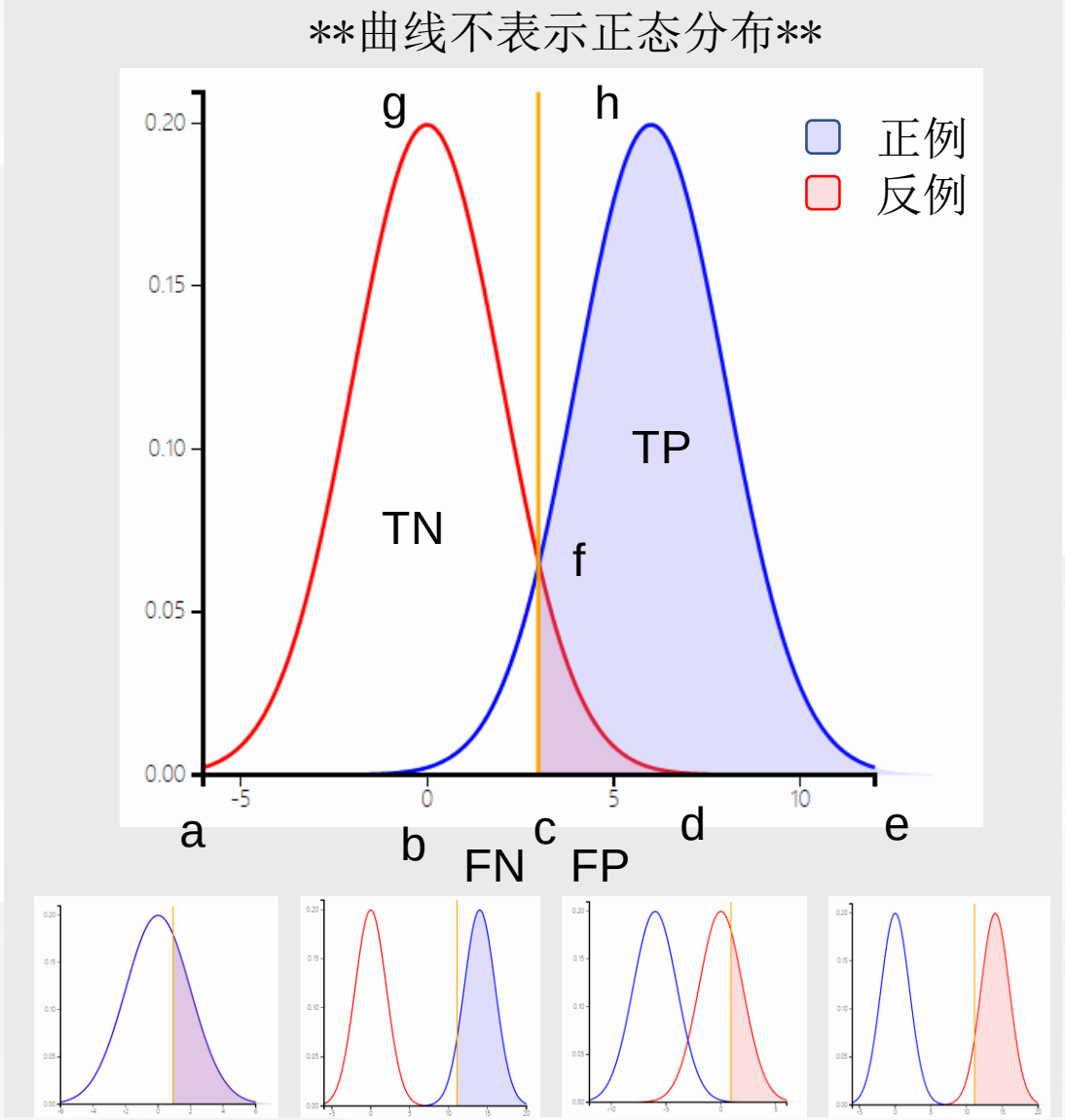
ACC/准确率：$\frac{TP+TN}{P+N}$

$$F_1 - score = \frac{2}{\frac{1}{recal}+\frac{1}{precision}}$$

如果记不住，那就记住最关键的一句：“TPR大，FPR小，模型比较，F1-score”。

【思考】TPR、FPR、ACC可以作为评价指标吗？

模型评价



混淆矩阵的直观理解

红线代表负类的分布曲线

蓝线代表正类的分布曲线

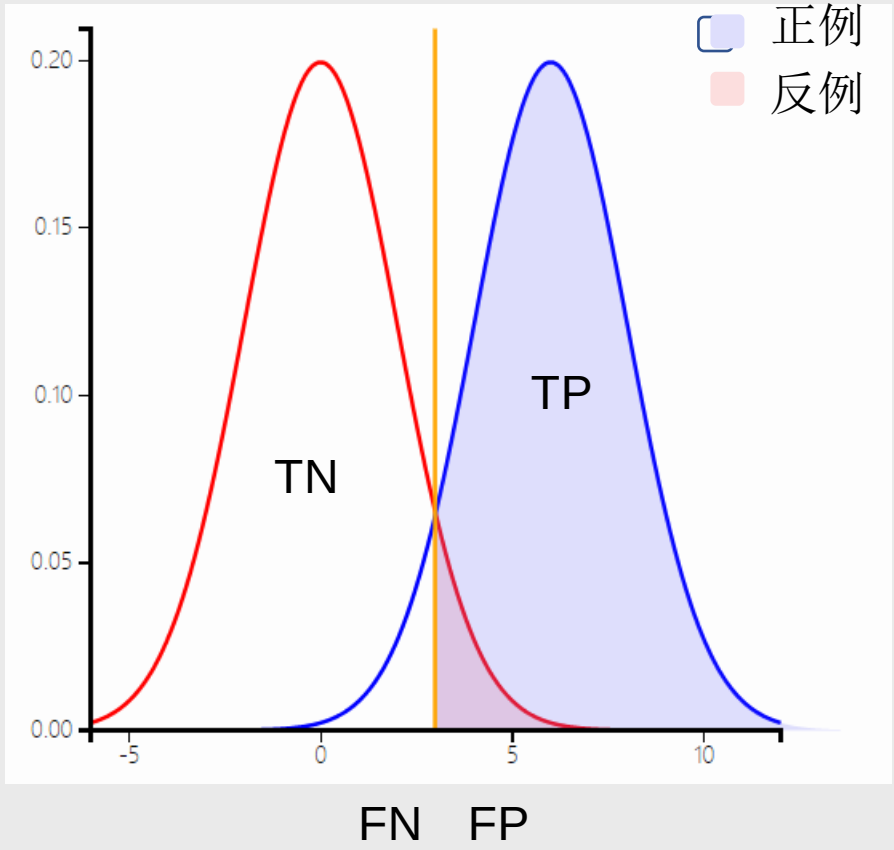
黄线代表模型划分标准，左侧为负右侧为正

【问题】

TP、FP、TN、FN分别对应哪个区域？

TPR和FPR分别对应那部分比例？变化规律怎样？

右下四幅图形是在描述什么？



逻辑回归的概率阈值选择

模型优化的目的是让TPR尽可能大，FPR尽可能小。

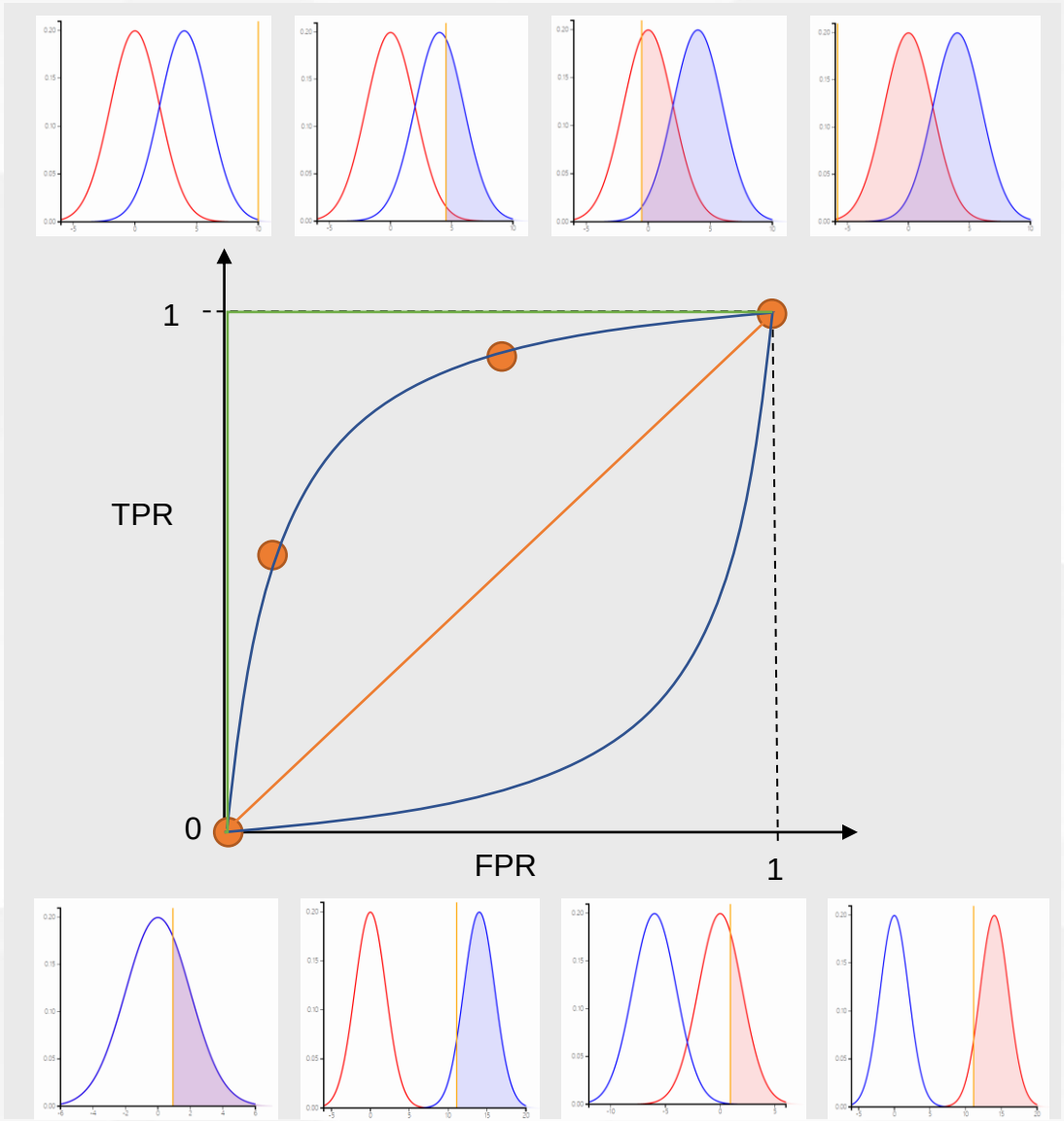
学习器产生的概率预测，和分类阈值进行比较，大于阈值则分为正类，否则为反类。

我们可以通过调节概率阈值来改变TPR和FPR，但TPR和FPR总是同向变化的，因此我们说：

“逻辑回归的概率阈值选择，就是在平衡TPR和FPR之间的代价关系。”

为了直观的描述TPR和FPR的关系，由此引入ROC曲线（receiver operating characteristic curve，受试者工作特征曲线）。

模型评价



ROC曲线

以FPR为x轴，以TPR为y轴，当阈值变化时，画出TPR和FPR的变化关系，得到的曲线就是ROC曲线。

考虑以下几种情况下的ROC曲线形状：

1、模型在一定程度上能够分别不同类别

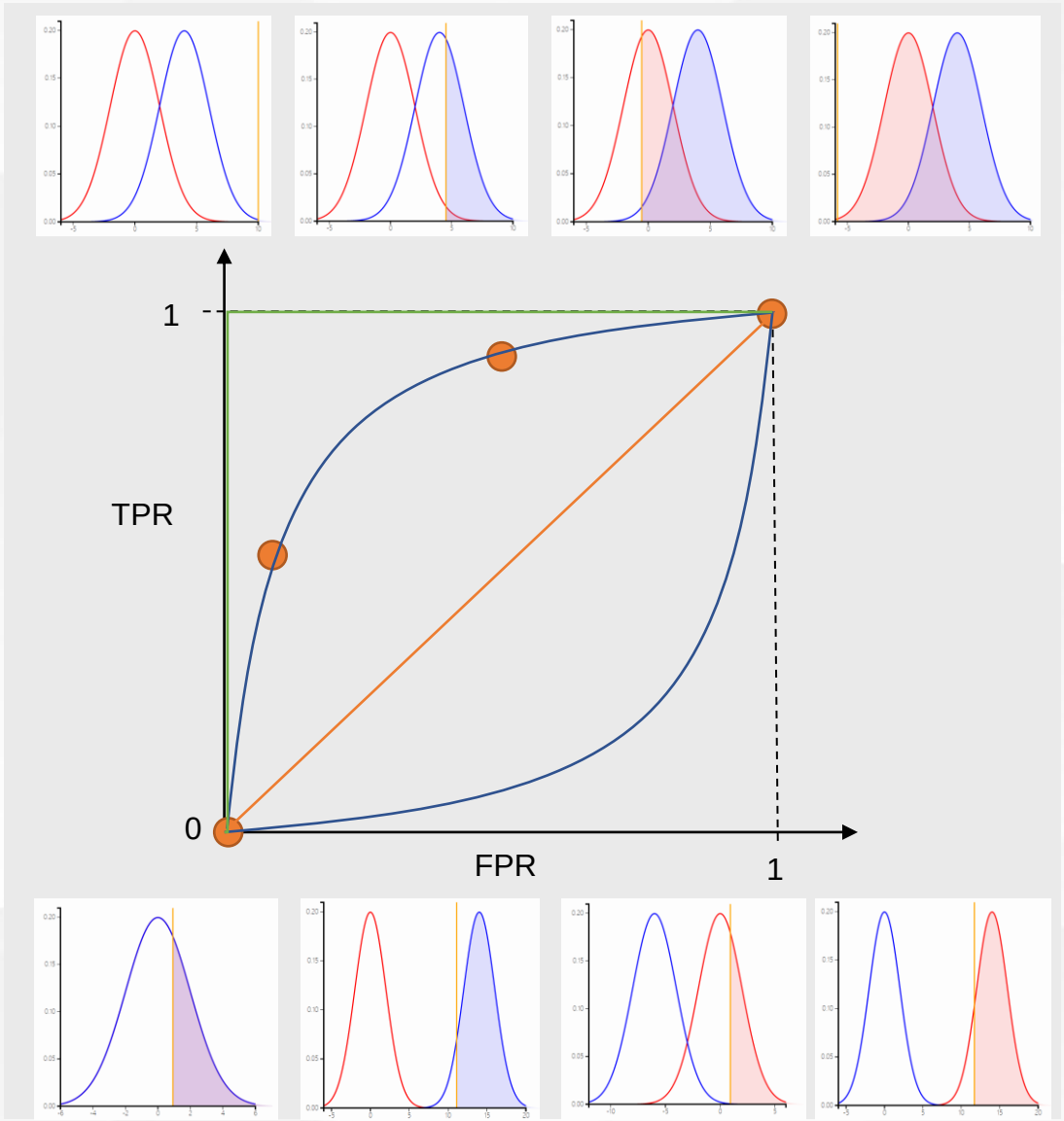
2、模型完全不能分别不同类别（等价于随机猜测）

3、模型能够完全分别不同类别

4、模型经常猜错不同类别（不如随机猜测）

5、模型完全猜错了

模型评价



ROC与AUC

AUC是ROC曲线下的面积，0≤AUC ≤1，AUC越大，说明模型预测能力越好。

1. 当模型预测结果和随机猜测没有区别时，ROC曲线为左下到右上的对角线，AUC=0.5
2. 当模型预测结果好于随机猜测时，ROC曲线为向左上凸起的曲线，0.5<AUC ≤1
3. 当模型预测结果不如随机猜测时，ROC曲线为向右下凹陷的曲线，0 ≤AUC<0.5

推荐概率阈值为距离左上角最近的点所对应的阈值，即$p = \arg\ max(TPR - FPR)$

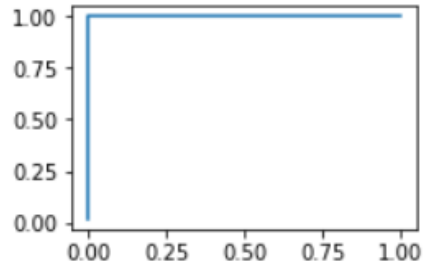
【思考】怎样和一位局外人解释ROC和AUC？

模型评价

```
# 混淆矩阵
from sklearn.metrics import confusion_matrix
confusion_matrix(y,model.predict(X))
```

```
array([[47,  3],
       [ 0, 50]], dtype=int64)
```

```
# ROC
from sklearn.metrics import roc_curve,roc_auc_score
fpr,tpr,prob = roc_curve(y,model.predict_proba(X)[:,1])
import matplotlib.pyplot as plt
plt.figure(figsize=(3,2))
plt.plot(fpr,tpr)
plt.show()
print(roc_auc_score(y,model.predict_proba(X)[:,1]))
```




```
1.0
```

```
from sklearn.metrics import f1_score
f1_score(y,model.predict(X))
```

```
0.970873786407767
```

模型评价的python实现

混淆矩阵

sklearn.metrics.confusion_matrix

confusion_matrix(y_true,y_hat)

ROC与AUC

sklearn.metrics.roc_curve,roc_auc_score

roc_curve(y_true,y_prob)

F1-score

sklearn.metrics.f1_score

f1_score(y_true,y_hat)

课程目录

Course catalogue

不平衡集问题

		预测		
		0	1	
实际	0	99	0	FPR=0/99=0
	1	1	0	TPR=0/1=0

什么是不平衡集

某个样本集共有100个样本集，有99个反例和1个正例，现在有一个逻辑回归分类器，不管喂给分类器什么数据，都会被预测为反例，那么模型的准确率ACC=99/100=99%。

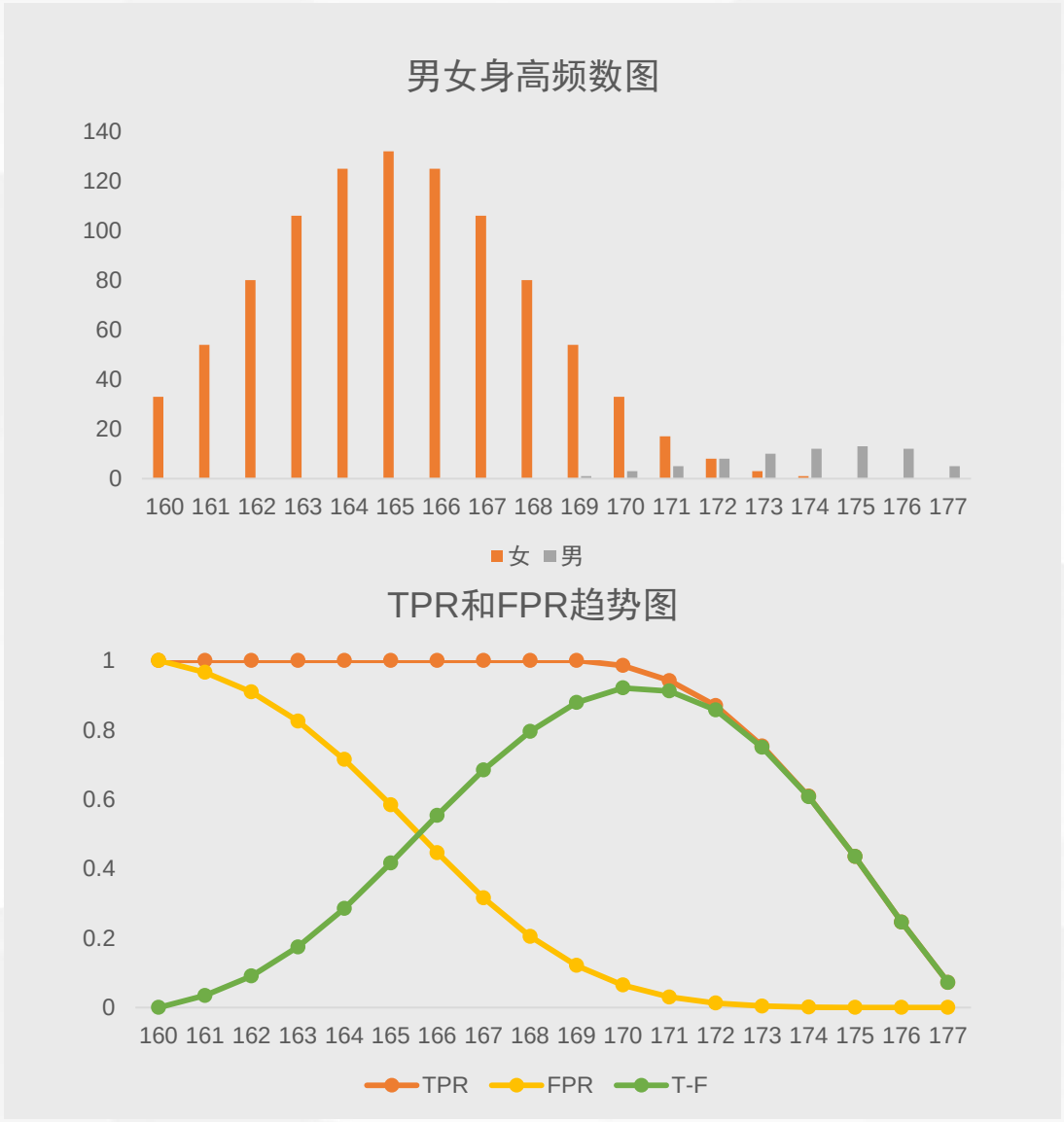
【问题】你怎样评价这个分类器模型？

不平衡集广泛存在与现实应用中，比如：

信用卡欺诈分析中，欺诈客户往往占比很小。肿瘤诊断中，最终确诊为肿瘤的患者占比也很小。营销推广，被筛选为目标客户的，也应该占比较小。

这些案例有个共同点，那就是为了不放跑“坏人”，我们愿意在一定程度上忍受错杀“好人”的风险。

不平衡集问题



不平衡集带来的问题

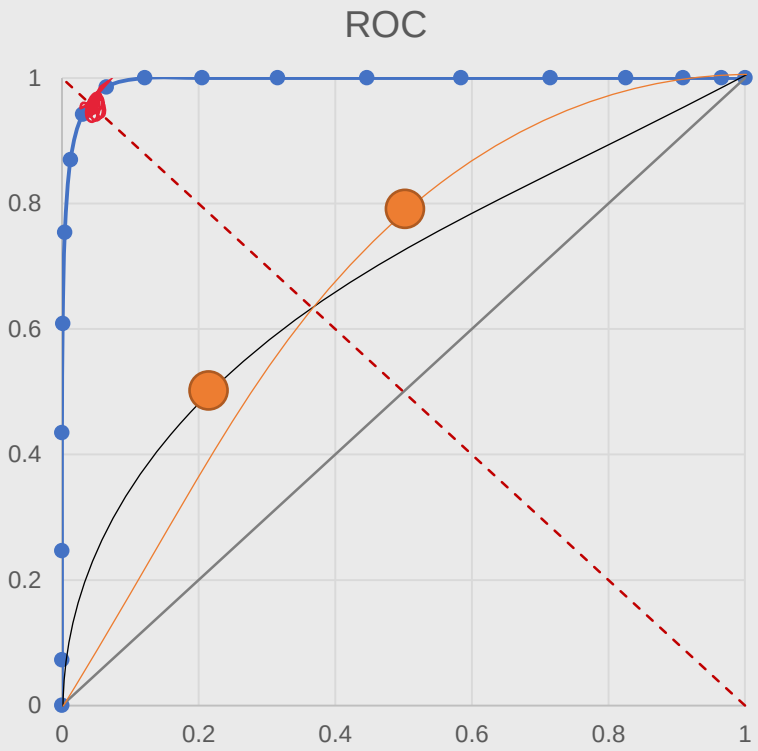
身高	女	男	TPR	FPR	T-F	t	sigmoid
160	33		1	1	0	-2.60604	0.068751
161	54		1	0.965517	0.034483	-2.60132	0.069053
162	80		1	0.909091	0.090909	-2.59661	0.069357
163	106		1	0.825496	0.174504	-2.59189	0.069662
164	125		1	0.714734	0.285266	-2.58717	0.069968
165	132		1	0.584117	0.415883	-2.58246	0.070276
166	125		1	0.446186	0.553814	-2.57774	0.070585
167	106		1	0.315569	0.684431	-2.57303	0.070895
168	80		1	0.204807	0.795193	-2.56831	0.071206
169	54	1	1	0.121212	0.878788	-2.56359	0.071519
170	33	3	0.985507	0.064786	0.920721	-2.55888	0.071832
171	17	5	0.942029	0.030303	0.911726	-2.55416	0.072147
172	8	8	0.869565	0.012539	0.857026	-2.54944	0.072464
173	3	10	0.753623	0.00418	0.749443	-2.54473	0.072781
174	1	12	0.608696	0.001045	0.607651	-2.54001	0.0731
175		13	0.434783	0	0.434783	-2.53529	0.073421
176		12	0.246377	0	0.246377	-2.53058	0.073742
177		5	0.072464	0	0.072464	-2.52586	0.074065

【思考】使用0.5作为概率阈值，会导致什么问题？

【思考】既然求TPR-FPR的最大值，为什么还需要sigmoid变换？

【思考】此时ROC曲线是什么样子的？

不平衡集问题



加权混淆矩阵

不平衡集的ROC曲线与问题解决方案

观察不平衡数据集及ROC，可以得到以下结论：

- 不平衡数据集的ROC曲线顶点偏离左上右下对角线
- 不平衡数据集的概率阈值不等于0.5

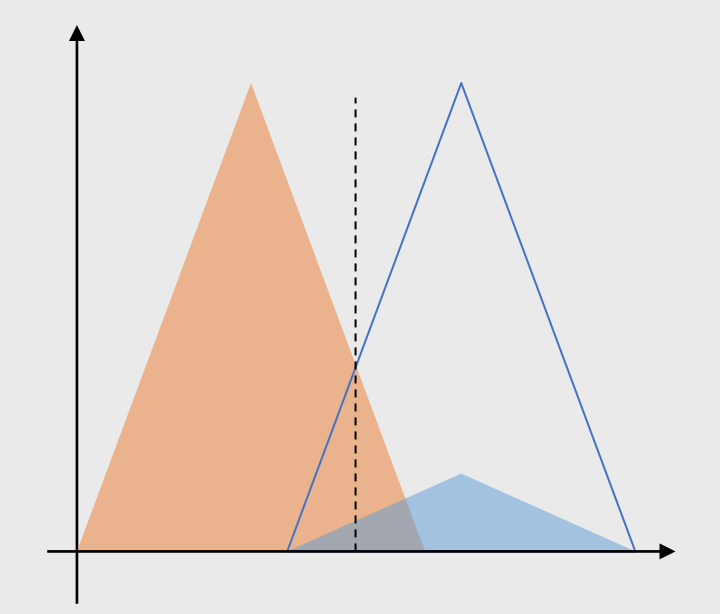
由此得出不平衡集问题的以下几种解决方案：

- 换个对不平衡集更加鲁棒的算法，比如随机森林
- 人为增加某类样本，重新平衡样本标签的分布
- ROC选择”恰当”的概率阈值

【思考】分别结合以下两种假设，思考什么叫“恰当”的阈值？

- 放跑一个负样本会引起地球毁灭
- 放跑一个正样本会引起地球毁灭

不平衡集问题

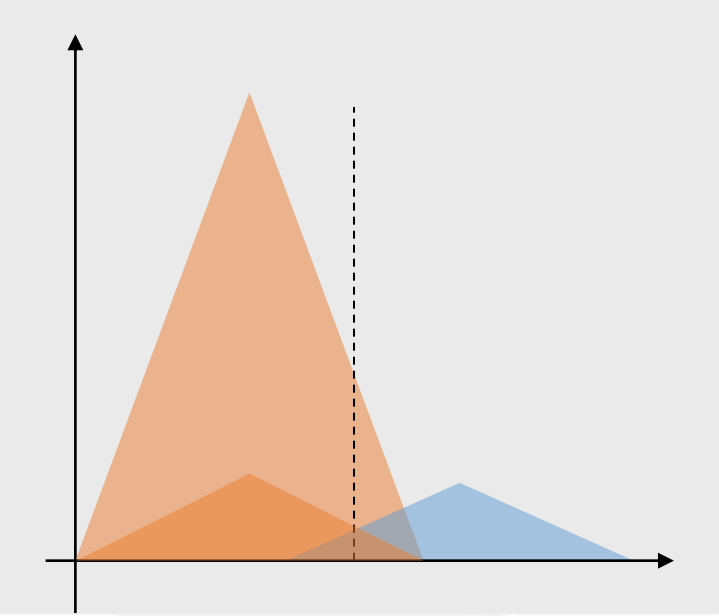


过采样（Oversampling）

对少数标签的样本进行重抽样

会增加样本数据

不会新增样本点

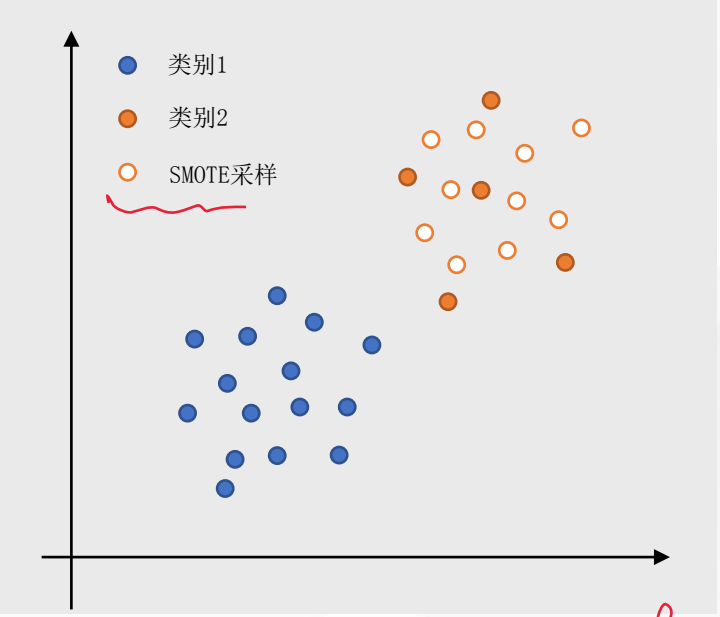


欠采样（undersampling）

对多数标签的样本进行重抽样

会减少样本数据

不会新增样本点



SMOTE

对少数标签的样本进行重抽样

会增加样本数据

会新增样本点

不平衡集问题

```
# 读取数据，查看类标签偏斜程度
import pandas as pd
df = pd.read_csv('datas/creditcard.csv')
sum(df.Class==1)/df.shape[0]
```

```
0.001727485630620034
```

```
# 数据预处理
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
df['normal_amount'] = ss.fit_transform(df['Amount'].values.reshape(-1,1))
df = df.drop(['Amount','Time'], axis=1)
X = df.loc[:,df.columns != 'Class']
y = df.loc[:,df.columns == 'Class']

# 建立模型并预测
from sklearn.linear_model import LogisticRegression
logr = LogisticRegression(class_weight='balanced').fit(X,y.values.ravel())
y_pred = logr.predict(X)

# 绘制混淆矩阵
from sklearn.metrics import confusion_matrix
confusion_matrix(y,y_pred)
```

```
array([[277780,   6535],
       [    39,    453]], dtype=int64)
```

过采样的Python解决方案

载入逻辑回归

from sklearn.linear_model import LogisticRegression

设置class_weight参数调整标签权重

LogisticRegression(class_weight='balanced')

数据标准化

from sklearn.preprocessing import StandardScaler

不平衡集问题

身高	女	男	TPR	FPR	T-F	t	sigmoid
160	33		1	1	0	-2.60604	0.068751
161	54		1	0.965517	0.034483	-2.60132	0.069053
162	80		1	0.909091	0.090909	-2.59661	0.069357
163	106		1	0.825496	0.174504	-2.59189	0.069662
164	125		1	0.714734	0.285266	-2.58717	0.069968
165	132		1	0.584117	0.415883	-2.58246	0.070276
166	125		1	0.446186	0.553814	-2.57774	0.070585
167	106		1	0.315569	0.684431	-2.57303	0.070895
168	80		1	0.204807	0.795193	-2.56831	0.071206
169	54	1	1	0.121212	0.878788	-2.56359	0.071519
170	33	3	0.985507	0.064786	0.920721	-2.55888	0.071832
171	17	5	0.942029	0.030303	0.911726	-2.55416	0.072147
172	8	8	0.869565	0.012539	0.857026	-2.54944	0.072464
173	3	10	0.753623	0.00418	0.749443	-2.54473	0.072781
174	1	12	0.608696	0.001045	0.607651	-2.54001	0.0731
175		13	0.434783	0	0.434783	-2.53529	0.073421
176		12	0.246377	0	0.246377	-2.53058	0.073742
177		5	0.072464	0	0.072464	-2.52586	0.074065

ROC与概率阈值选择

观察不平衡集的sigmoid值，可以注意到sigmoid值发生了“偏移”，在合理的身高区间内，sigmoid值变小。

此时如果继续使用0.5作为阈值，可以看到模型会将所有的样本均划分为“女”，而没有“男”。因此需要对模型的概率阈值进行调整。

找到TPR-FPR最大值为0.920721，对应的sigmoid值为0.071832，因此概率阈值应当调整为0.071832，即 $p(y=1|\boldsymbol{x})$>0.071832时，即预测为1，否则为0。

不平衡集问题

```
# 读取数据，查看类标签偏斜程度
import pandas as pd
df = pd.read_csv('datas/creditcard.csv')
sum(df.Class==1)/df.shape[0]
```

```
0.001727485630620034
```

```
# 数据预处理
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
df['normal_amount'] = ss.fit_transform(df['Amount'].values.reshape(-1,1))
df = df.drop(['Amount','Time'], axis=1)
X = df.loc[:,df.columns != 'Class']
y = df.loc[:,df.columns == 'Class']
```

```
# 建立模型并预测
from sklearn.linear_model import LogisticRegression
logr = LogisticRegression().fit(X,y.values.ravel())
y_pred_prob = logr.predict_proba(X)[:,1]

# 绘制ROC，计算概率阈值
from sklearn.metrics import roc_curve
fpr,tpr,thresh = roc_curve(y,y_pred_prob)
import numpy as np
idx = np.argmax(tpr-fpr)
Thresh = thresh[idx]

# 使用概率阈值进行预测
y_pred = (y_pred_prob>Thresh)*1

# 绘制混淆矩阵
from sklearn.metrics import confusion_matrix
confusion_matrix(y,y_pred)
```

```
array([[277653,    6662],
       [    49,     443]], dtype=int64)
```

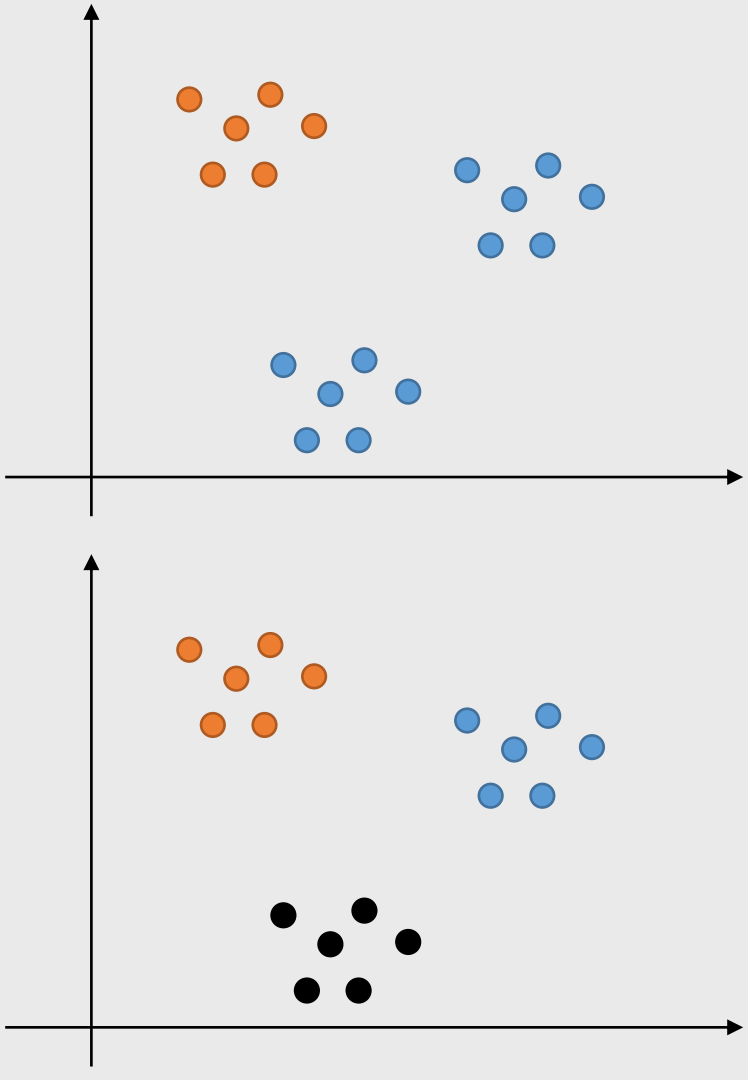
概率阈值选择的Python解决方案

```
# 预测概率

model = LogisticRegression().fit(X_tr,y_tr)

y_pred_prob = model.predict_proba(X_ts)

# 使用ROC计算概率阈值

fpr,tpr,thresh = roc_curve(y_true,y_score)

idx = np.argmax(tpr-fpr)

Thresh = thresh[idx]

# 重新预测分类

y_pred = y_pred_prob > Thresh
```

课程目录

Course catalogue

解决多分类问题



多分类问题的解决方法

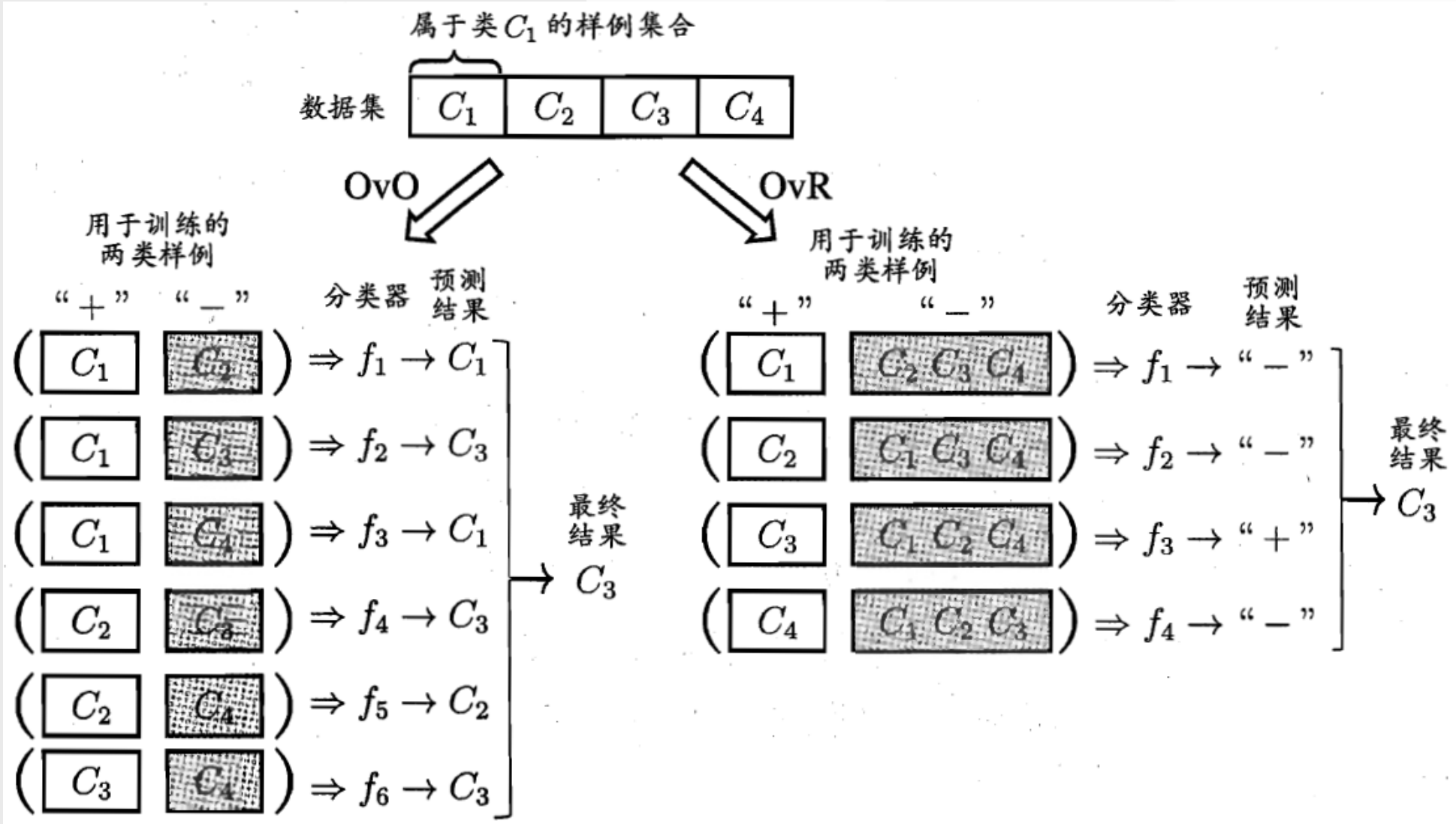
在二分类问题中，我们通过预测0，1的概率来实现类别的预测，同理，对一个类别数为k的多分类问题，如果能够预测到属于类k的概率，就可以解决多分类预测问题。

多分类预测方法主要包括：

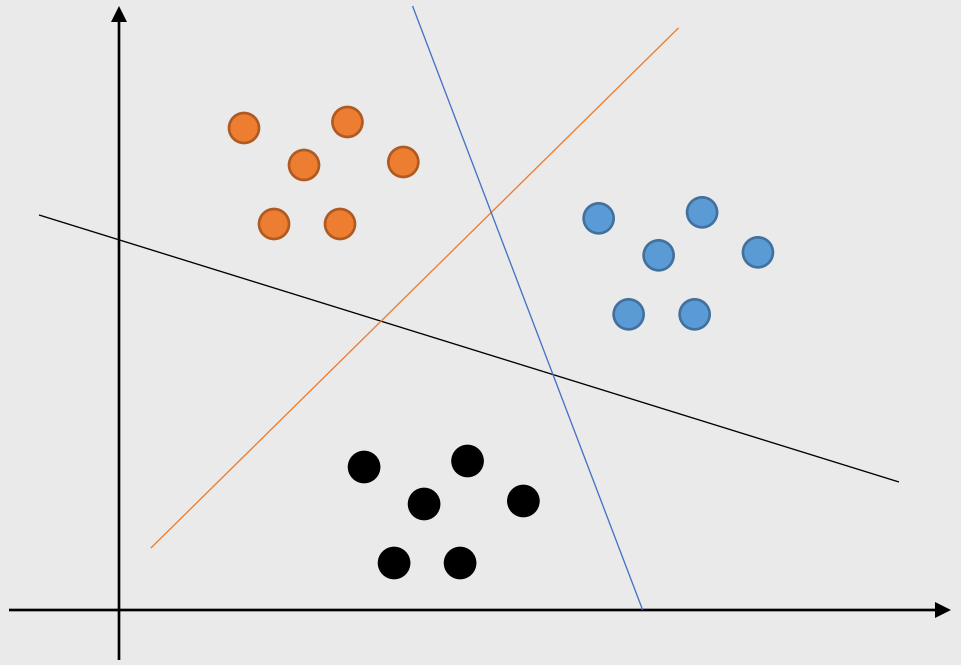
- 一对一 OvO(One vs One)
- 一对余 OVR/OVA(One vs Rest / One vs All)
- Softmax

不平衡集问题

OvO 与 OvR示意图



解决多分类问题



OVR/OVA

假定有某数据黄蓝黑三类，那么OVR计算如下：

1、将蓝黑视为一类，计算黄色概率 $h^{yellow}(x)$

2、将黄黑视为一类，计算蓝色概率 $h^{blue}(x)$

3、将黄蓝视为一类，计算黑色概率 $h^{black}(x)$

4、计算$h(x) = [h^{yellow}(x), h^{yellow}(x), h^{yellow}(x)]$的最大值，作为该样本的预测分类

【思考】OVR方法会遇到什么问题？

解决多分类问题

	sepal_l	sepal_w	petal_l	petal_w	classes
73	6.1	2.8	4.7	1.2	1.0
18	5.7	3.8	1.7	0.3	0.0
118	7.7	2.6	6.9	2.3	2.0

```
# 读取数据
import pandas as pd
df = pd.read_csv('datas/iris.csv',index_col=0)
X = df.iloc[:,:4]
y = df.iloc[:,4]
```

```
# 建立OVR和SoftMax模型
from sklearn.linear_model import LogisticRegression
model_ovr = LogisticRegression(multi_class='ovr').fit(X,y)
model_sm = LogisticRegression(multi_class='multinomial',
                              solver='lbfgs').fit(X,y)
```

```
# 查看预测结果
from sklearn.metrics import confusion_matrix
confusion_matrix(y,model_ovr.predict(X))
```

```
array([[50,  0,  0],
       [ 0, 45,  5],
       [ 0,  1, 49]], dtype=int64)
```

```
confusion_matrix(y,model_sm.predict(X))
```

```
array([[50,  0,  0],
       [ 0, 47,  3],
       [ 0,  1, 49]], dtype=int64)
```

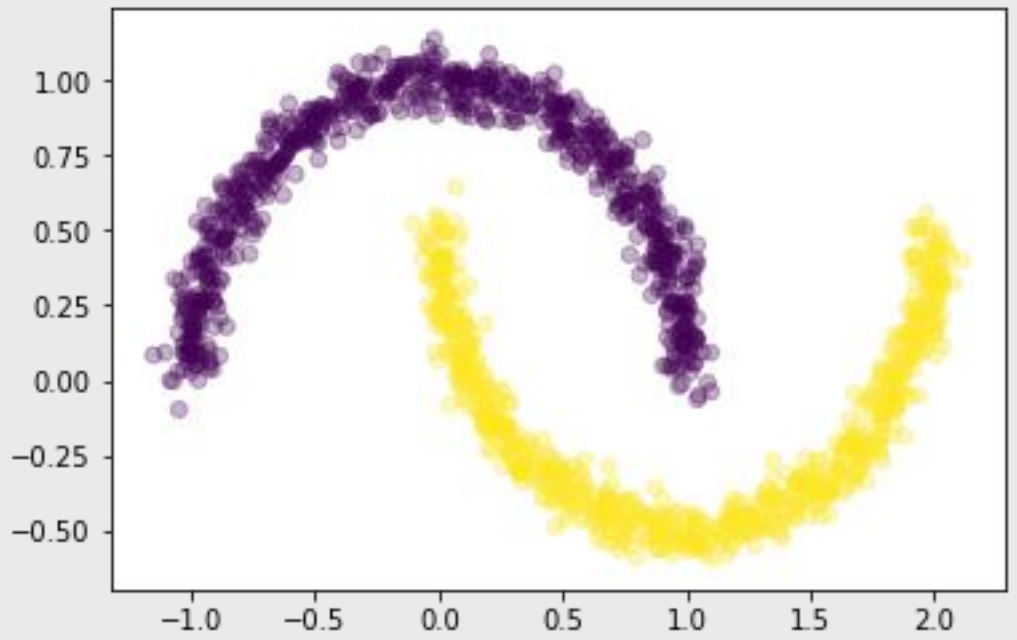
OVR与SoftMax的python实现

```
# OVR

model_ovr = LogisticRegression(multi_class='ovr')

model_ovr.fit(X_tr,y_tr)

# SoftMax

model_sm = LogisticRegression(multi_class='multinomial')

model_sm.fit(X_tr,y_tr)

# 参数solver

solver : {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}
```

课程目录

Course catalogue

非线性分类问题



非线性分类问题解决思路

对于逻辑回归而言，我们的目的是找到一条“直线”，将不同类别尽可能分开。根据逻辑回归原理，我们知道，这条“直线”和特征值的线性组合有关。

【思考】如图所示的数据，该如何进行分类？

借鉴线性回归中，多项式回归的做法，如果增加n次项，则能够构建出这样一条“曲线分类边界”。和多项式一样，我们可以通过添加正则项来避免过拟合。

非线性分类问题

```python
# 生成数据
from sklearn import datasets
noisy_moons = datasets.make_moons(n_samples=1500, noise=.05)
X = noisy_moons[0]
y = noisy_moons[1]
```

```python
# 数据可视化
import matplotlib.pyplot as plt
plt.scatter(X[:,0],X[:,1],c=y,alpha=0.3)
plt.show()
```

```python
# 生成多项式数据
from sklearn.preprocessing import PolynomialFeatures
X_poly = PolynomialFeatures(degree=9).fit_transform(X)
```

```python
# 划分训练集和测试集
from sklearn.model_selection import train_test_split
X_poly_tr,X_poly_ts,y_tr,y_ts = train_test_split(X_poly,y)
```

```python
# 建模并预测，设置正则化及正则化系数防止过拟合
from sklearn.linear_model import LogisticRegression
logr = LogisticRegression(penalty='l2',C=1).fit(X_poly_tr,y_tr)
y_pred = logr.predict(X_poly_ts)
```

```python
# 查看混淆矩阵
from sklearn.metrics import confusion_matrix
confusion_matrix(y_ts,y_pred)
```

非线性分类的python解决方案

生成多项式数据

sklearn.preprocessing.PolynomialFeatures

PolynomialFeatures(degree=9)

添加正则项，设置正则项系数

LogisticRegression(penalty='l2',C=1)

混淆矩阵

sklearn.metrics.confusion_matrix

Confusion_matrix(y_true,y_predict)