法律声明

- 课程详情请咨询
  - 微信公众号：北风教育
  - 官方网址：*http://www.ibeifeng.com/*

人工智能之机器学习

多分类及多标签分类算法

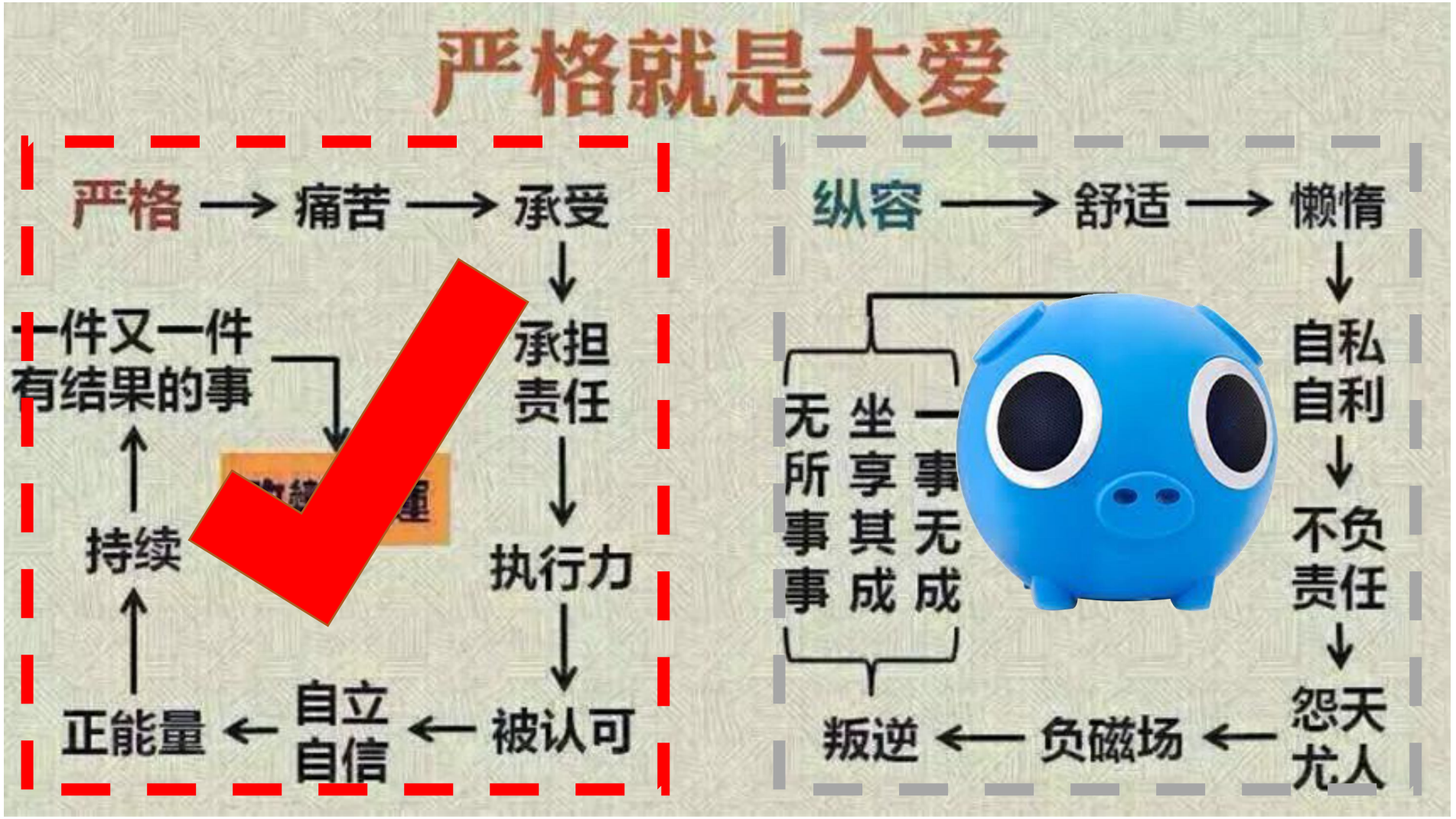
主讲人：Gerry

上海育创网络科技有限公司

课程要求

- 课上课下“九字”真言
  - 认真听，善摘录，勤思考
  - **多温故，乐实践**，再发散
- 四不原则
  - 不懒散惰性，不迟到早退
  - 不请假旷课，不拖延作业
- 一点注意事项
  - 违反“四不原则”，不包就业和推荐就业

严格是大爱


严格就是大爱
严格
痛苦
承受
承担责任
执行力
被认可
自立自信
正能量
持续
一件又一件有结果的事
纵容
舒适
懒惰
自私自利
不负责任
怨天尤人
负磁场
叛逆
一事无成
坐享其成
无所事事

寄语

做别人不愿做的事，

做别人不敢做的事，

做别人做不到的事。

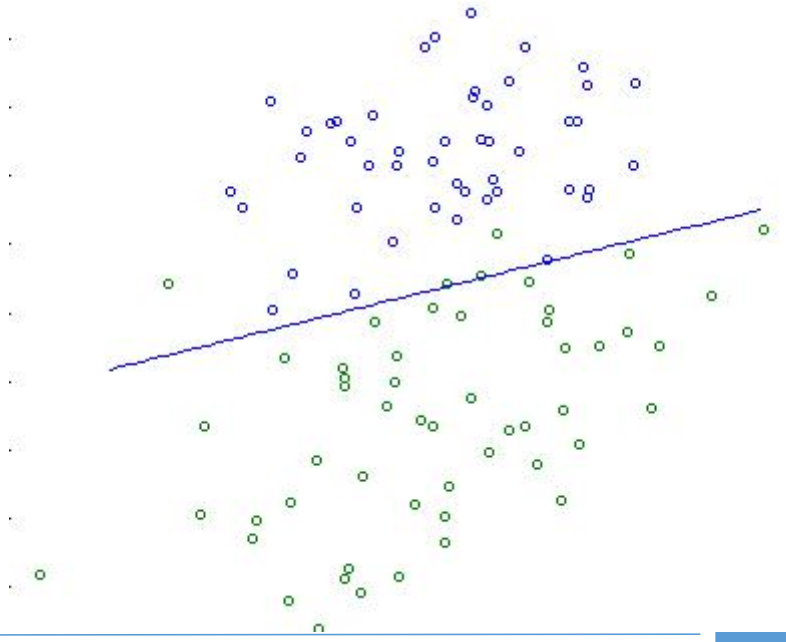
课程内容

- 单标签二分类问题
- 单标签多分类问题
- 多标签算法问题

单标签二分类算法原理

- 单标签二分类这种问题是我们最常见的算法问题，主要是指label标签的取值只有两种，并且算法中只有一个需要预测的label标签；直白来讲就是每个实例的可能类别只有两种(A or B)；此时的分类算法其实是在构建一个分类线将数据划分为两个类别。常见的算法：Logistic、SVM、KNN等

$$y = f(x) \quad y \in \{-1, +1\}$$

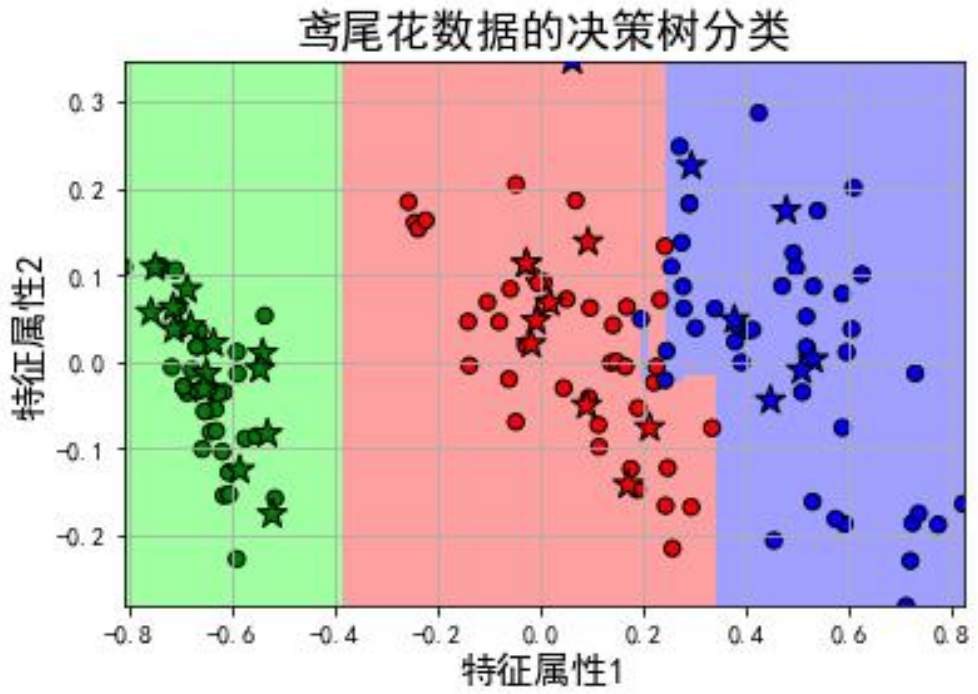


单标签多分类算法原理

- 单标签多分类问题其实是指待预测的label标签只有一个，但是label标签的取值可能有多种情况；直白来讲就是每个实例的可能类别有K种($t_1,t_2,...t_k,k\geq 3$)；常见算法：Softmax、KNN等；

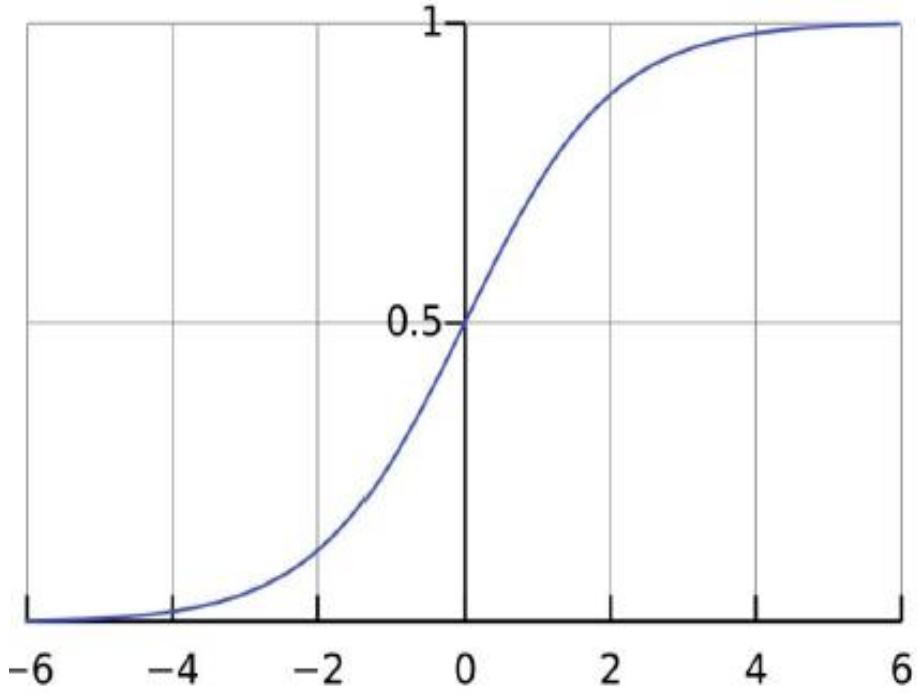
$$y = f(x)$$

$$y \in \{t_1, t_2, \ldots, t_k\}$$



Logistic算法原理

$$sigmoid = h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$



$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right] \quad y^{(i)} \in \{0,1\}$$

Softmax算法原理

$$p(y=k \mid x;\theta)=\frac{e^{\theta_k^T x}}{\sum_{l=1}^{K} e^{\theta_l^T x}}, k=1,2\cdots,K$$

$$h_\theta(x)=\begin{bmatrix} p(y^{(i)}=1 \mid x^{(i)};\theta) \\ p(y^{(i)}=2 \mid x^{(i)};\theta) \\ \cdots \\ p(y^{(i)}=k \mid x^{(i)};\theta) \end{bmatrix}=\frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}}\begin{bmatrix} e^{\theta_1^T x} \\ e^{\theta_2^T x} \\ \cdots \\ e^{\theta_k^T x} \end{bmatrix} \Rightarrow \theta=\begin{bmatrix} \theta_{11} & \theta_{12} & \cdots & \theta_{1n} \\ \theta_{21} & \theta_{22} & \cdots & \theta_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \theta_{k1} & \theta_{k2} & \cdots & \theta_{kn} \end{bmatrix}$$

$$J(\theta)=-\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} I(y^{(i)}=j)\log\left(\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right) \qquad I(y^{(i)}=j)=\begin{cases} 1, & y^{(i)}=j \\ 0, & y^{(i)}\neq j \end{cases}$$

单标签多分类算法原理

- 在实际的工作中，如果是一个多分类的问题，我们可以将这个待求解的问题转换为二分类算法的延伸，即将多分类任务拆分为若干个二分类任务求解，具体的策略如下：
  - One-Versus-One(ovo)：一对一
  - One-Versus-All / One-Versus-the-Rest(ova/ovr)： 一对多
  - Error Correcting Output codes(纠错码机制)：多对多

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

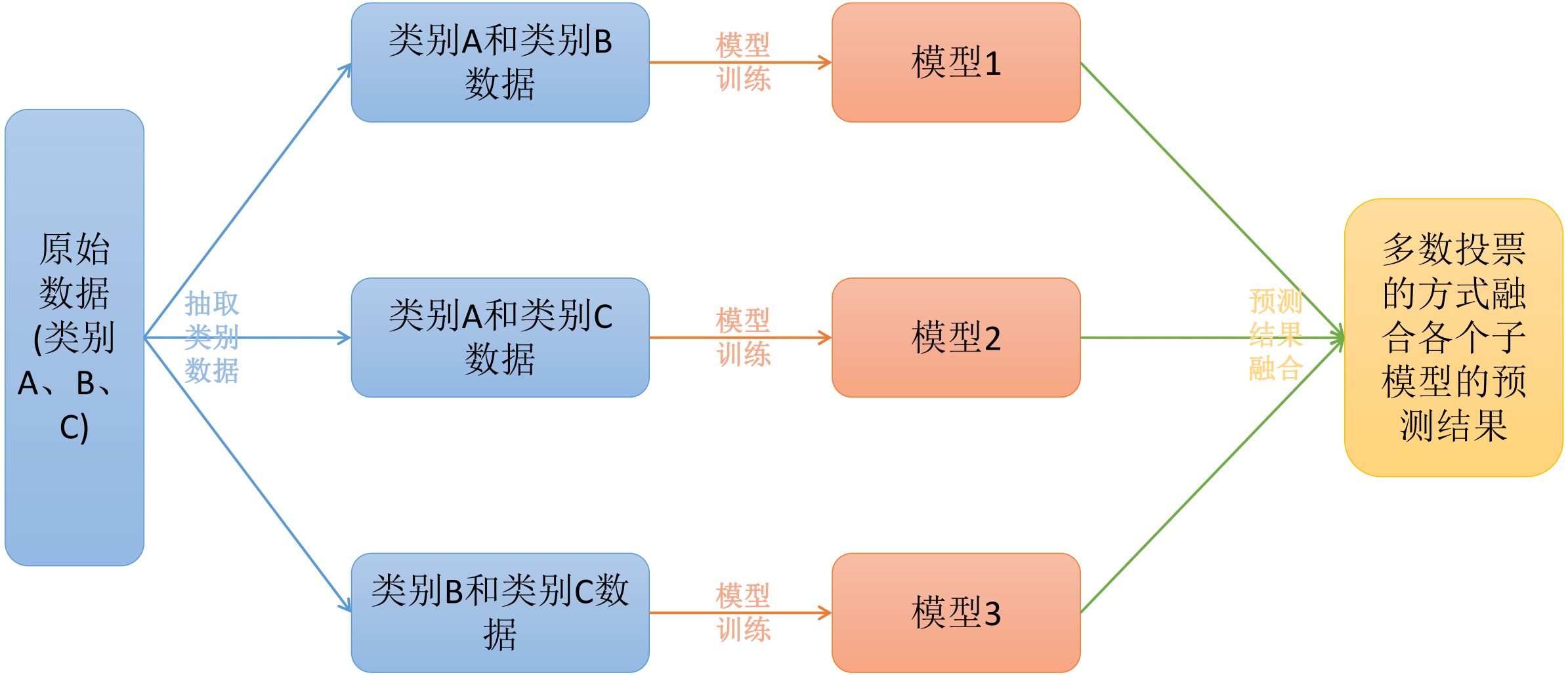
$$y_i = j, i = 1,2 \ldots n, j = 1,2 \ldots k$$

单标签多分类算法原理-ovo

- 原理：将K个类别中的两两类别数据进行组合，然后使用组合后的数据训练出来一个模型，从而产生K(K-1)/2个分类器，将这些分类器的结果进行融合，并将分类器的预测结果使用**多数投票**的方式输出最终的预测结果值。

1. for $(k, \ell) \in \mathcal{Y} \times \mathcal{Y}$
   obtain $\mathbf{w}_{[k,\ell]}$ by running linear binary classification on
   $$\mathcal{D}_{[k,\ell]} = \{(\mathbf{x}_n, y'_n = 2[\![y_n = k]\!] - 1) : y_n = k \text{ or } y_n = \ell\}$$
2. return $g(\mathbf{x}) = \text{tournament champion}\left\{\mathbf{w}_{[k,\ell]}^T \mathbf{x}\right\}$

单标签多分类算法原理-ovo



单标签多分类算法原理-ovr

- 原理：在一对多模型训练中，不是两两类别的组合，而是将每一个类别作为正例，其它剩余的样例作为反例分别来训练K个模型；然后在预测的时候，如果在这K个模型中，只有一个模型输出为正例，那么最终的预测结果就是属于该分类器的这个类别；如果产生多个正例，那么则可以选择根据分类器的置信度作为指标，来选择置信度最大的分类器作为最终结果，常见置信度：精确度、召回率。

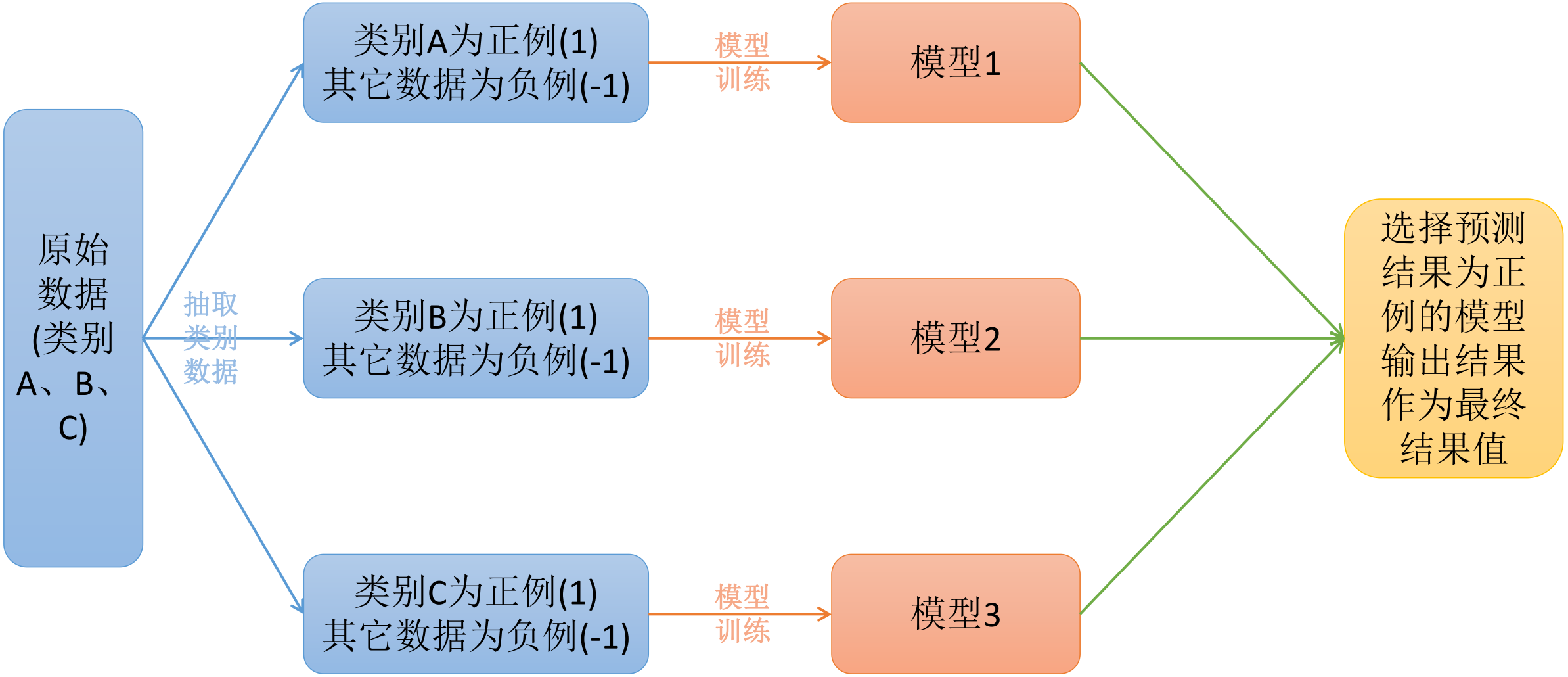
1. for $k \in \mathcal{Y}$

   obtain $\mathbf{w}_{[k]}$ by running logistic regression on

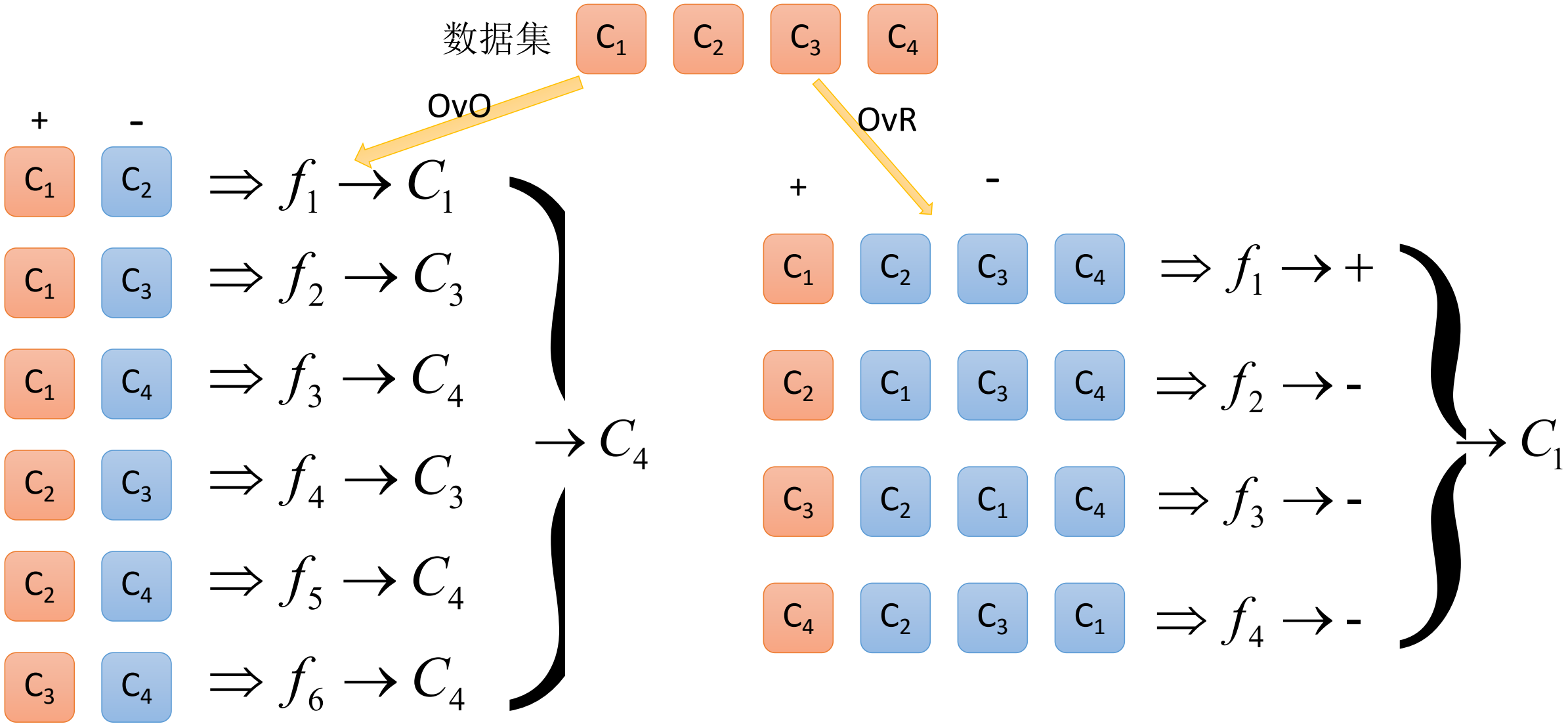
   $$\mathcal{D}_{[k]} = \{(\mathbf{x}_n, y'_n = 2[\![y_n = k]\!] - 1)\}_{n=1}^{N}$$

2. return $g(\mathbf{x}) = \text{argmax}_{k \in \mathcal{Y}} \left(\mathbf{w}_{[k]}^T \mathbf{x}\right)$

单标签多分类算法原理-ovr



OvO和OvR的区别


数据集
C1
C2
C3
C4
OvO
OvR
+
-
C1 C2 ⇒ f1 → C1
C1 C3 ⇒ f2 → C3
C1 C4 ⇒ f3 → C4
C2 C3 ⇒ f4 → C3
C2 C4 ⇒ f5 → C4
C3 C4 ⇒ f6 → C4
→ C4
+
-
C1 C2 C3 C4 ⇒ f1 → +
C2 C1 C3 C4 ⇒ f2 → -
C3 C2 C1 C4 ⇒ f3 → -
C4 C2 C3 C1 ⇒ f4 → -
→ C1

OvR和OvO案例代码

class sklearn.multiclass. **OneVsRestClassifier** (*estimator*, *n_jobs=1*) ¶ [source]

class sklearn.multiclass. **OneVsOneClassifier** (*estimator*, *n_jobs=1*) [source]

In [4]:
```
# 模型创建
clf = OneVsRestClassifier(LinearSVC(random_state=0))
# 模型构建
clf.fit(X, y)
```

Out[4]:
```
OneVsRestClassifier(estimator=LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=0, tol=0.0001,
     verbose=0),
          n_jobs=1)
```

In [9]:
```
# 模型构建
clf = OneVsOneClassifier(LinearSVC(random_state=0))
# 模型训练
clf.fit(X, y)
```

Out[9]:
```
OneVsOneClassifier(estimator=LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=0, tol=0.0001,
     verbose=0),
          n_jobs=1)
```

单标签多分类算法原理-Error Correcting

- 原理：将模型构建应用分为两个阶段：编码阶段和解码阶段；编码阶段中对K个类别中进行M次划分，每次划分将一部分数据分为正类，一部分数据分为反类，每次划分都构建出来一个模型，模型的结果是在空间中对于每个类别都定义了一个点；解码阶段中使用训练出来的模型对测试样例进行预测，将预测样本对应的点和类别之间的点求距离，选择距离最近的类别作为最终的预测类别。

单标签多分类算法原理-Error Correcting

	f_1	f_2	f_3	f_4	f_5	距离
$C_1 \rightarrow$	-1	+1	-1	+1	+1	$\rightarrow 2\sqrt{2}$
$C_2 \rightarrow$	+1	-1	-1	+1	-1	$\rightarrow 2\sqrt{5}$
$C_3 \rightarrow$	-1	-1	+1	-1	+1	$\rightarrow 2$
$C_4 \rightarrow$	-1	-1	+1	+1	-1	$\rightarrow 2\sqrt{3}$
测试样本	-1	+1	+1	-1	+1	

Error Correcting案例代码

class sklearn.multiclass. **OutputCodeClassifier**(*estimator*, *code_size=1.5*, *random_state=None*, *n_jobs=1*) ¶ [source]

In [11]:
```
# 模型对象创建
clf = OutputCodeClassifier(LinearSVC(random_state=0), code_size=2, random_state=0)
# 模型构建
clf.fit(X, y)
```

Out[11]:
```
OutputCodeClassifier(code_size=2,
                     estimator=LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=0, tol=0.0001,
          verbose=0),
                     n_jobs=1, random_state=0)
```

多标签算法概述

- Multi-Label Machine Learning(MLL算法)是指预测模型中存在多个y值，具体分为两类不同情况：1) 多个待预测的y值；2)在分类模型中，一个样例可能存在多个不固定的类别。根据多标签业务问题的复杂性，可以将问题分为两大类：1)待预测值之间存在相互的依赖关系；2)待预测值之间是不存在依赖关系的。对于这类问题的解决方案可以分为两大类：1) 转换策略(Problem Transformation Methods)；2)算法适应(Algorithm Adaptation)。

多标签算法概述

x_1	x_2	x_3	x_4	y_1	y_2
1.1	1.5	1.8	1.2	1	1.5
2.1	2.8	2.4	2.1	2.1	2.45
-1.2	-0.5	0.25	-0.12	-0.5	-1
0	0.2	0.89	1.2	0	0.5

x_1	x_2	x_3	x_4	y_1
1.1	1.5	1.8	1.2	a
2.1	2.8	2.4	2.1	b,c
-1.2	-0.5	0.25	-0.12	a,c
0	0.2	0.89	1.2	e

Problem Transformation Methods

- Problem Transformation Methods又叫做策略转换或者问题转换，是一种将多标签的分类问题转换成为单标签模型构造的问题，然后将模型合并的一种方式，主要有以下几种方式：
  - Binary Relevance(first-order)
  - Classifier Chains(high-order)
  - Calibrated Label Ranking(second-order)

Problem Transformation Methods
Binary Relevance

- Binary Relevance的核心思想是将多标签分类问题进行分解，将其转换为q个二元分类问题，其中每个二元分类器对应一个待预测的标签。

$$D_j = \{(x_i, \phi(Y_i, y_j)) | 1 \le i \le m\} \quad \phi(Y_i, y_j) = \begin{cases} +1, y_j \in Y_i \\ -1, otherwise \end{cases}$$

$$g_j = f(D_j)$$

$$Y = \{y_j \mid g_j(x) > 0, 1 \le j \le q\} \cup \left\{ y_{j^*} \mid j^* = \arg\max_{1 \le j \le q} g_j(x) \right\}$$

Problem Transformation Methods
Binary Relevance

Y=BinaryRelevance($\mathcal{D}$, $\mathcal{B}$, $\boldsymbol{x}$)

1. **for** $j = 1$ to q **do**
2. Construct the binary training set $\mathcal{D}_j$ according to Eq.(3);
3. $g_j \leftarrow \mathcal{B}(\mathcal{D}_j)$;
4. **endfor**
5. Return Y according to Eq.(5);

Problem Transformation Methods
Binary Relevance

- Binary Relevance方式的优点如下：
  - 实现方式简单，容易理解；
  - 当y值之间不存在相关的依赖关系的时候，模型的效果不错。
- 缺点如下：
  - 如果y直接存在相互的依赖关系，那么最终构建的模型的泛化能力比较弱；
  - 需要构建q个二分类器，q为待预测的y值数量，当q比较大的时候，需要构建的模型会比较多。

Problem Transformation Methods
Classifier Chains

- Classifier Chains的核心思想是将多标签分类问题进行分解，将其转换成为一个二元分类器链的形式，其中链后的二元分类器的构建式在前面分类器预测结果的基础上的。在模型构建的时候，首先将标签顺序进行shuffle打乱排序操作，然后按照从头到尾分别构建每个标签对应的模型。

y_1	y_2	y_3	y_4	y_5

shuffle sorted

y_2	y_4	y_1	y_5	y_3

Problem Transformation Methods
Classifier Chains模型构建

$$\tau : shuffle\_sorted\{1,2,...,q\}$$

$$D_{\tau(j)} = \left\{\left([x_i, pre^i_{\tau(j)}], \phi\left(Y_i, y_{\tau(j)}\right)\right) \middle| 1 \le i \le m\right\}$$

$$pre^i_{\tau(j)} = \left(\phi\left(Y_i, y_{\tau(1)}\right), \phi\left(Y_i, y_{\tau(2)}\right), ..., \phi\left(Y_i, y_{\tau(j-1)}\right)\right)^T \quad pre^i_{\tau(1)} = \phi$$

$$g_{\tau(j)} = f\left(D_{\tau(j)}\right)$$

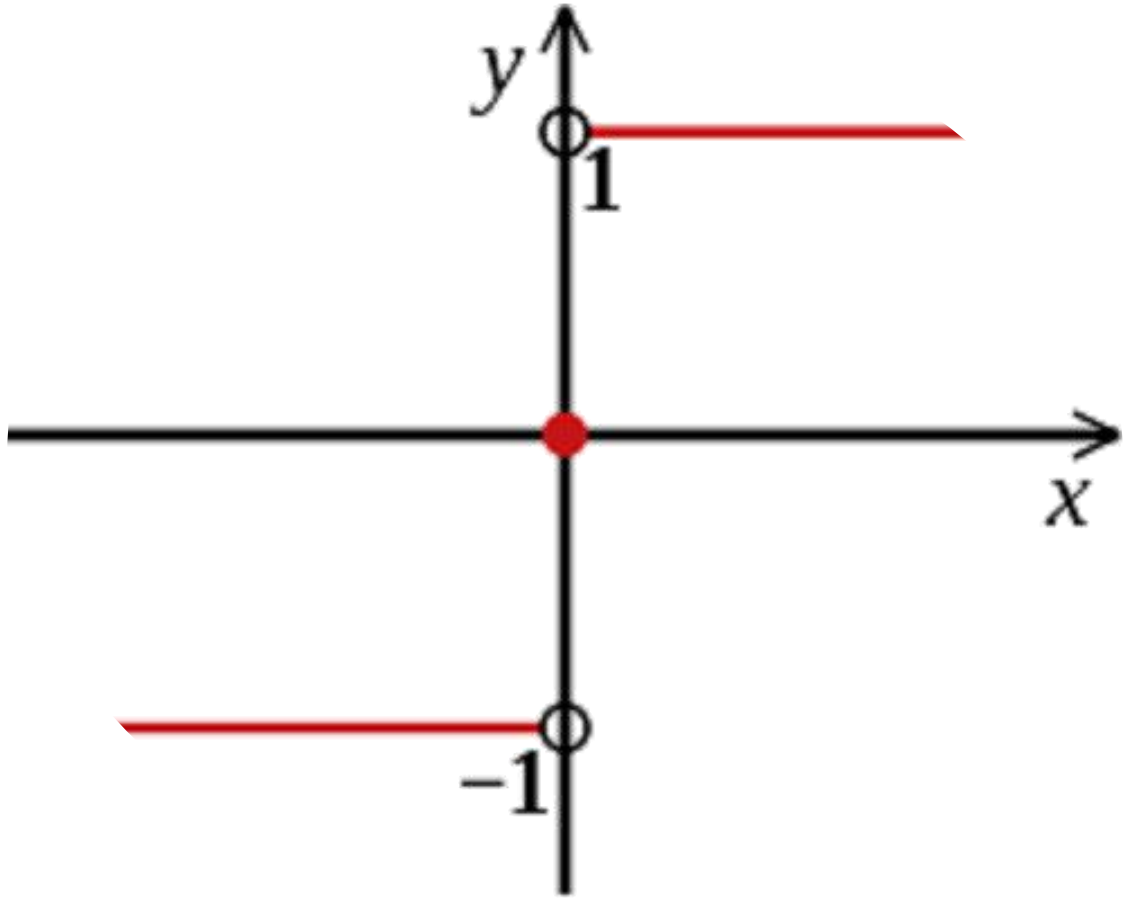
Problem Transformation Methods
Classifier Chains模型预测

$$\lambda_{\tau(1)}^{x} = sign\left(g_{\tau(1)}(x)\right)$$

$$\lambda_{\tau(j)}^{x} = sign\left(g_{\tau(j)}\left(\left[x, \lambda_{\tau(1)}^{x}, \lambda_{\tau(2)}^{x} \cdots \lambda_{\tau(j-1)}^{x}\right]\right)\right), 2 \le j \le q$$

$$Y = \left\{ y_{\tau(j)} \mid \lambda_{\tau(j)}^{x} = +1, 1 \le j \le q \right\}$$

Sign函数



Problem Transformation Methods
Classifier Chains

Y=ClassifierChains($\mathcal{D}$, $\mathcal{B}$, τ, $\boldsymbol{x}$)

1. **for** $j = 1$ to q **do**

2. Construct the chaining binary training set $\mathcal{D}_{\tau(j)}$ according to Eq.(6);

3. $g_{\tau(j)} \leftarrow \mathcal{B}(\mathcal{D}_{\tau(j)})$;

4. **endfor**

5. Return Y according to Eq.(8) (in conjunction with Eq.(7));

Problem Transformation Methods
Classifier Chains

- Classifier Chains方式的优点如下：
  - 实现方式相对比较简单，容易理解；
  - 考虑标签之间的依赖关系，最终模型的泛化能力相对于Binary Relevance方式构建的模型效果要好。
- 缺点如下：
  - 很难找到一个比较适合的标签依赖关系。

Problem Transformation Methods
Calibrated Label Ranking

- Calibrated Label Ranking的核心思想是将多标签分类问题进行分解，将其转换为标签的排序问题，最终的标签就是排序后最大的几个标签值。

$$D_{jk} = \left\{ \left(x_i, \ell\left(Y_i, y_j, y_k \right) \right) \middle| \phi\left(Y_i, y_j \right) \neq \phi\left(Y_i, y_k \right), 1 \leq i \leq m \right\}$$

$$\ell\left(Y_i, y_j, y_k \right) = \begin{cases} +1, if\ \phi\left(Y_i, y_j \right) = +1\ and\ \phi\left(Y_i, y_k \right) = -1 \\ -1, if\ \phi\left(Y_i, y_j \right) = -1\ and\ \phi\left(Y_i, y_k \right) = +1 \end{cases}$$

$$g_{jk} = f\left(D_{jk} \right)$$

Problem Transformation Methods
Calibrated Label Ranking

$$g_{jk} = f(D_{jk}) \Rightarrow \hat{y} = y_j \quad if \quad g_{jk}(x) > 0 \quad else \quad y_k$$

$$\zeta(x, y_j) = \sum_{k=1}^{j-1} \left\| g_{kj}(x) \le 0 \right\| + \sum_{k=j+1}^{q} \left\| g_{jk}(x) > 0 \right\|$$

$$Y = \left\{ y_j \middle| \zeta(x, y_j) > threshold, 1 \le j \le q \right\}$$

Problem Transformation Methods
Calibrated Label Ranking

$$D_{jV} = \left\{ \left(x_i, \phi\left(Y_i, y_j \right) \right) \middle| 1 \le i \le m \right\}$$

$$g_{jV} = f\left(D_{jV} \right) \Rightarrow \hat{y} = y_j \quad if \quad g_{jV}(x) > 0$$

$$\zeta^*\left(x, y_j \right) = \zeta\left(x, y_j \right) + \left\| g_{jV}(x) > 0 \right\| \qquad \zeta^*(x) = \sum_{j=1}^{q} \left\| g_{jV}(x) \le 0 \right\|$$

$$Y = \left\{ y_j \mid \zeta^*\left(x, y_j \right) > \zeta^*(x), 1 \le j \le q \right\}$$

Problem Transformation Methods
Calibrated Label Ranking

Y=CalibratedLabelRanking($\mathcal{D}$, $\mathcal{B}$, $\boldsymbol{x}$)

1. **for** $j = 1$ to $q - 1$ **do**
2. **for** $k = j + 1$ to q **do**
3. Construct the binary training set $\mathcal{D}_{jk}$ according to Eq.(9);
4. $g_{jk} \leftarrow \mathcal{B}(\mathcal{D}_{jk})$;
5. **endfor**
6. **endfor**
7. **for** $j = 1$ to q **do**
8. Construct the binary training set $\mathcal{D}_{j\mathrm{V}}$ according to Eq.(11);
9. $g_{j\mathrm{V}} \leftarrow \mathcal{B}(\mathcal{D}_{j\mathrm{V}})$;
10. **endfor**
11. Return Y according to Eq.(14) (in conjunction with Eqs.(10)-(13));

Problem Transformation Methods
Calibrated Label Ranking

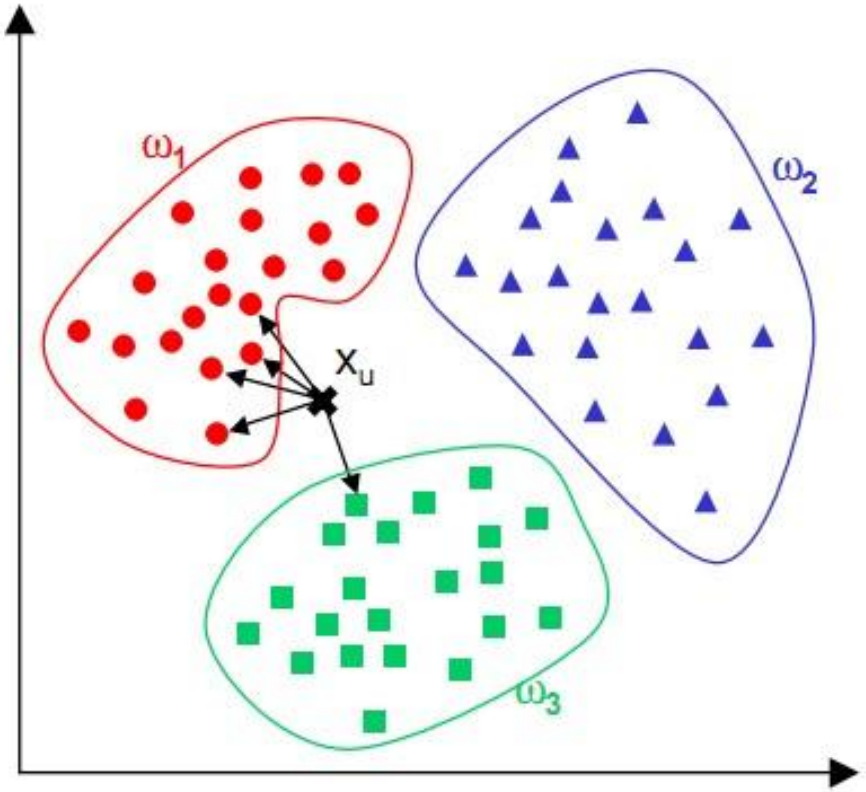
- Calibrated Label Ranking 方式的优点如下：
  - 考虑了标签两两组合的情况，最终的模型相对来讲泛化能力比较好。
- 缺点如下：
  - 只考虑两两标签的组合，没有考虑到标签与标签之间的所有依赖关系。

Algorithm Adaptation

- Algorithm Adaptation又叫做算法适应性策略，是一种将现有的单标签的算法直接应用到多标签上的一种方式，主要有以下几种方式：
  - ML-kNN
  - ML-DT

kNN

- k近邻算法(k-Nearest Neighbour, KNN)的思想：如果一个样本在特征空间中的k个最相似(即特征空间中距离最近)的样本中的大多数属于某一个类别，那么该样本属于这个类别。



Algorithm Adaptation ML-kNN

- ML-kNN的思想：对于每一个实例来讲，先获取距离它最近的k个实例，然后使用这些实例的标签集合，通过**最大后验概率(MAP)**来判断这个实例的预测标签集合的值。
- 最大后验概率（MAP）：其实就是在最大似然估计(MLE)中加入了这个要估计量的先验概率分布。

$$\hat{\theta}_{MLE}(x) = \arg\max_{\theta} f(x \mid \theta)$$

$$\hat{\theta}_{MAP}(x) = \arg\max_{\theta} \frac{f(x \mid \theta) g(\theta)}{\int_{\Theta} f(x \mid \theta') g(\theta') d\theta'} = \arg\max_{\theta} f(x \mid \theta) g(\theta)$$

Algorithm Adaptation ML-kNN

$$C_j = \sum_{(x^*, Y^*) \in N(x)} \left\| y_j \in Y^* \right\|$$

$$Y = \left\{ y_j \mid P\left(H_j \mid C_j\right) / P\left(\neg H_j \mid C_j\right) > 1, 1 \le j \le q \right\}$$

$$\frac{P\left(H_j \mid C_j\right)}{P\left(\neg H_j \mid C_j\right)} = \frac{P\left(H_j\right) P\left(C_j \mid H_j\right)}{P\left(\neg H_j\right) P\left(C_j \mid \neg H_j\right)}$$

Algorithm Adaptation
ML-kNN

$$P(H_j)=\frac{1}{n}\sum_{i=1}^{n}\|y_j \in Y_i\| \qquad P(\neg H_j)=1-P(H_j)$$

$$k_j[r]=\sum_{i=1}^{n}\|y_j \in Y_i\|\cdot\|C_j=r\|,0\le r\le k$$

$$\tilde{k}_j[r]=\sum_{i=1}^{n}\|y_j \notin Y_i\|\cdot\|C_j=r\|,0\le r\le k$$

$$P(C_j \mid H_j)=\frac{k_j[C_j]}{\sum_{r=1}^{k} k_j[r]}$$

$$P(C_j \mid \neg H_j)=\frac{\tilde{k}_j[C_j]}{\sum_{r=1}^{k} \tilde{k}_j[r]}$$

Algorithm Adaptation ML-kNN

Y=ML-kNN($\mathcal{D}$, k, $\boldsymbol{x}$)

1. **for** $i = 1$ to m **do**

2. Identify k nearest neighbors $\mathcal{N}(\boldsymbol{x}_i)$ for $\boldsymbol{x}_i$;

3. **endfor**

4. **for** $j = 1$ to q **do**

5. Estimate the prior probabilities $\mathbb{P}(H_j)$ and $\mathbb{P}(\neg H_j)$ according to Eq.(24);

6. Maintain frequency arrays κ_j and $\tilde{\kappa}_j$ according to Eq.(25);

7. **endfor**

8. Identify k nearest neighbors $\mathcal{N}(\boldsymbol{x})$ for $\boldsymbol{x}$;

9. **for** $j = 1$ to q **do**

10. Calculate statistic C_j according to Eq.(21);

11. **endfor**

12. Return Y according to Eq.(22) (in conjunction with Eqs.(23), (24) and (26));

Algorithm Adaptation
ML-DT

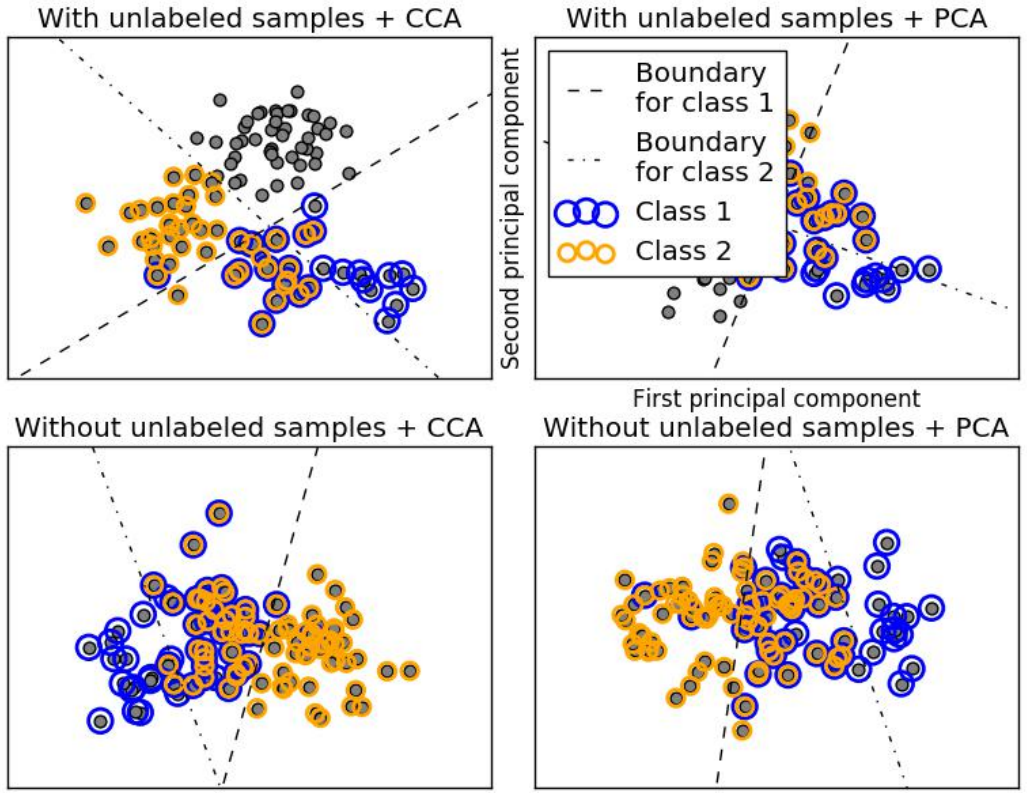
- ML-DT是使用决策树处理多标签内容，核心在于给予更细粒度的信息殇增益准则来构建这个决策树模型。

$$entry = \sum_{j=1}^{q} \left[- p_j \log p_j - \left(1 - p_j\right) \log\left(1 - p_j\right) \right]$$

$$p_j = \frac{\sum_{i=1}^{n} \left\| y_j \in Y_i \right\|}{n}$$

多标签分类在scikit-learn中的实现方式

- 在scikit-learn中使用OneVsRestClassifier对多标签进行分类操作，内部其实是将多标签问题转换为多类别的区分问题。



THANK YOU

上海育创网络科技有限公司