分类与预测

杨惠

2017/10/22

算法实现

我们通过举例说明：使用scikit-learn建立基于信息熵的决策树模型。

- 这个例子是经典的Kaggle101问题——泰坦尼克生还预测，部分数据如下：

Survived	PassengerId	Pclass	Sex	Age
0	1	3	male	22
1	2	1	female	38
1	3	3	female	26
1	4	1	female	35
0	5	3	male	35
0	6	3	male	

- 为了说明的方便，数据集有许多属性被删除了。通过观察可知：列Survived是指是否存活，是类别标签，属于预测目标；列Sex的取值是非数值型的。我们在进行数据预处理时应该合理应用Pandas的功能，让数据能够被模型接受。

算法实现

具体实现代码如下：

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier as DTC, export_graphviz
data = pd.read_csv('../data/titanic_data.csv', encoding='utf-8')
data.drop(['PassengerId'], axis=1, inplace=True)  # 舍弃ID列，不适合作为特征
# 数据是类别标签，将其转换为数，用1表示男，0表示女。
data.loc[data['Sex'] == 'male', 'Sex'] = 1
data.loc[data['Sex'] == 'female', 'Sex'] = 0
data.fillna(int(data.Age.mean()), inplace=True)  # 缺失值处理（均值插补）
print(data.head(6))   # 查看数据
```

算法实现

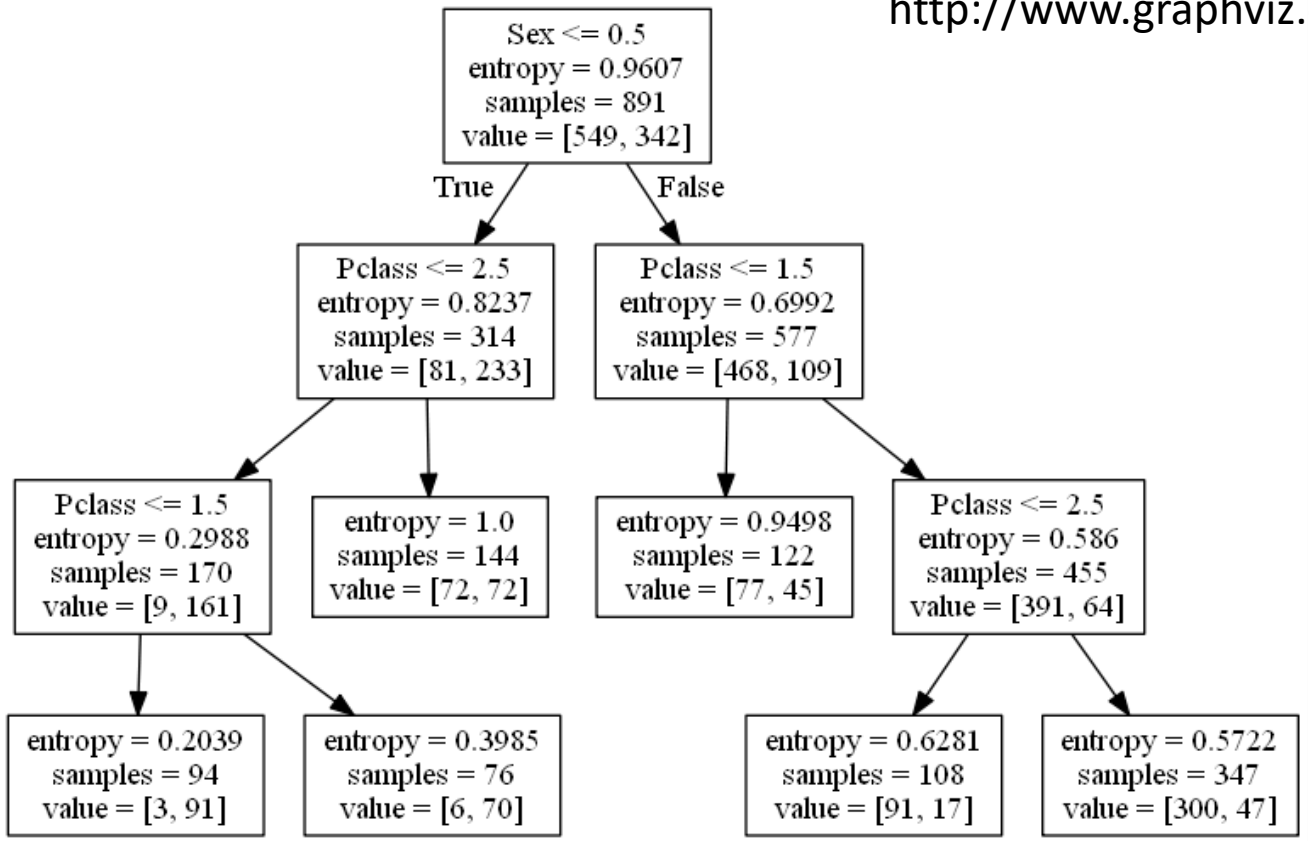
```
X = data.iloc[:, 1:3]     # 为便于展示，未考虑年龄（最后一列）
y = data.iloc[:, 0]
dtc = DTC(criterion='entropy')     # 初始化决策树对象，基于信息熵
dtc.fit(X, y)     # 训练模型
print('输出准确率：', dtc.score(X,y))
# 可视化决策树，导出结果是一个dot文件，需要安装Graphviz才能转换为.pdf或.png格式
with open('../tmp/tree.dot', 'w') as f:
    f = export_graphviz(dtc, feature_names=X.columns, out_file=f)
```

算法实现

运行代码后，将会输出一个tree.dot的文本文件。为了进一步将它转换为可视化格式，需要安装Graphviz（跨平台的、基于命令行的绘图工具），再在命令行中以如下方式编译。

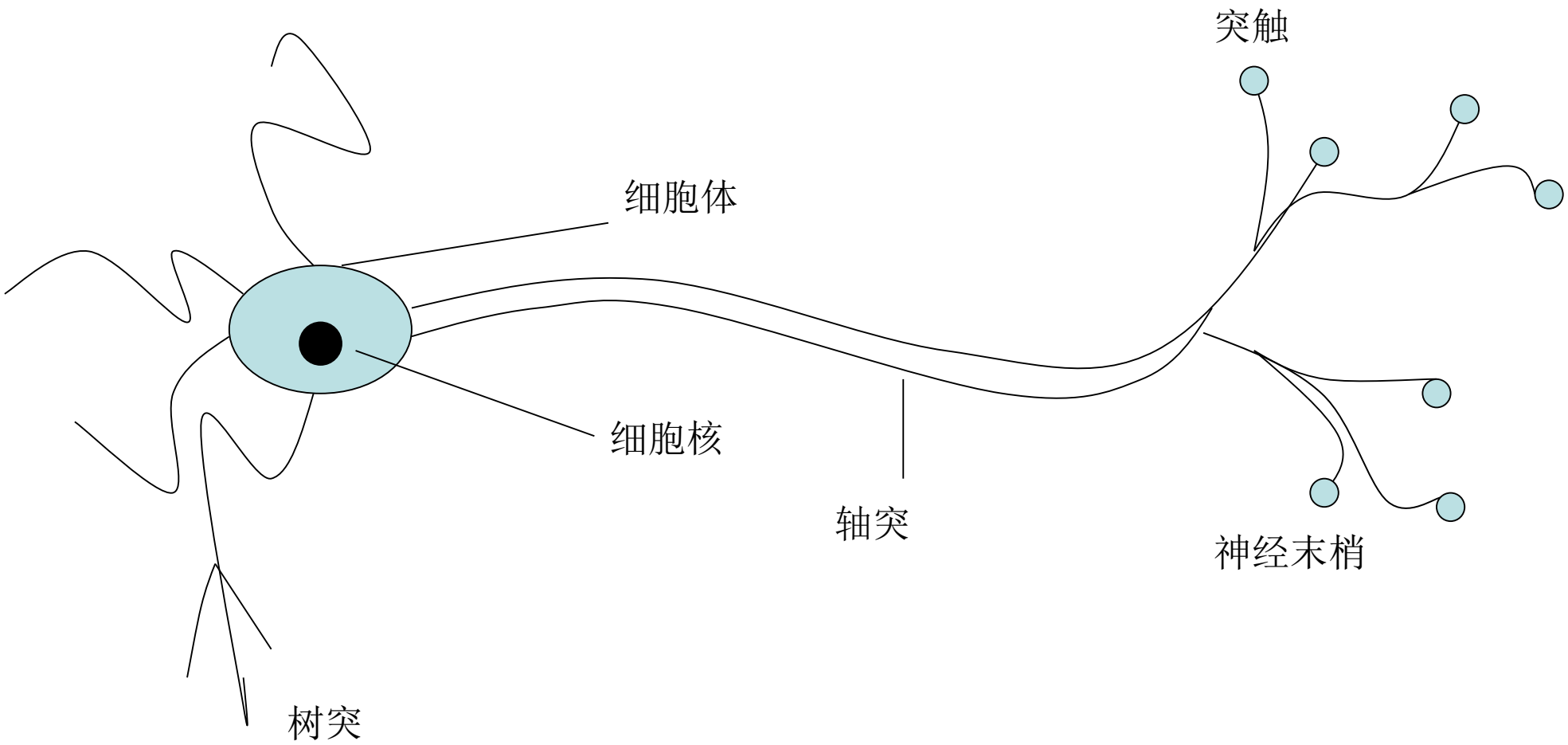
生成的效果图如下：

http://www.graphviz.org/Download_windows.php



目录

神经网络

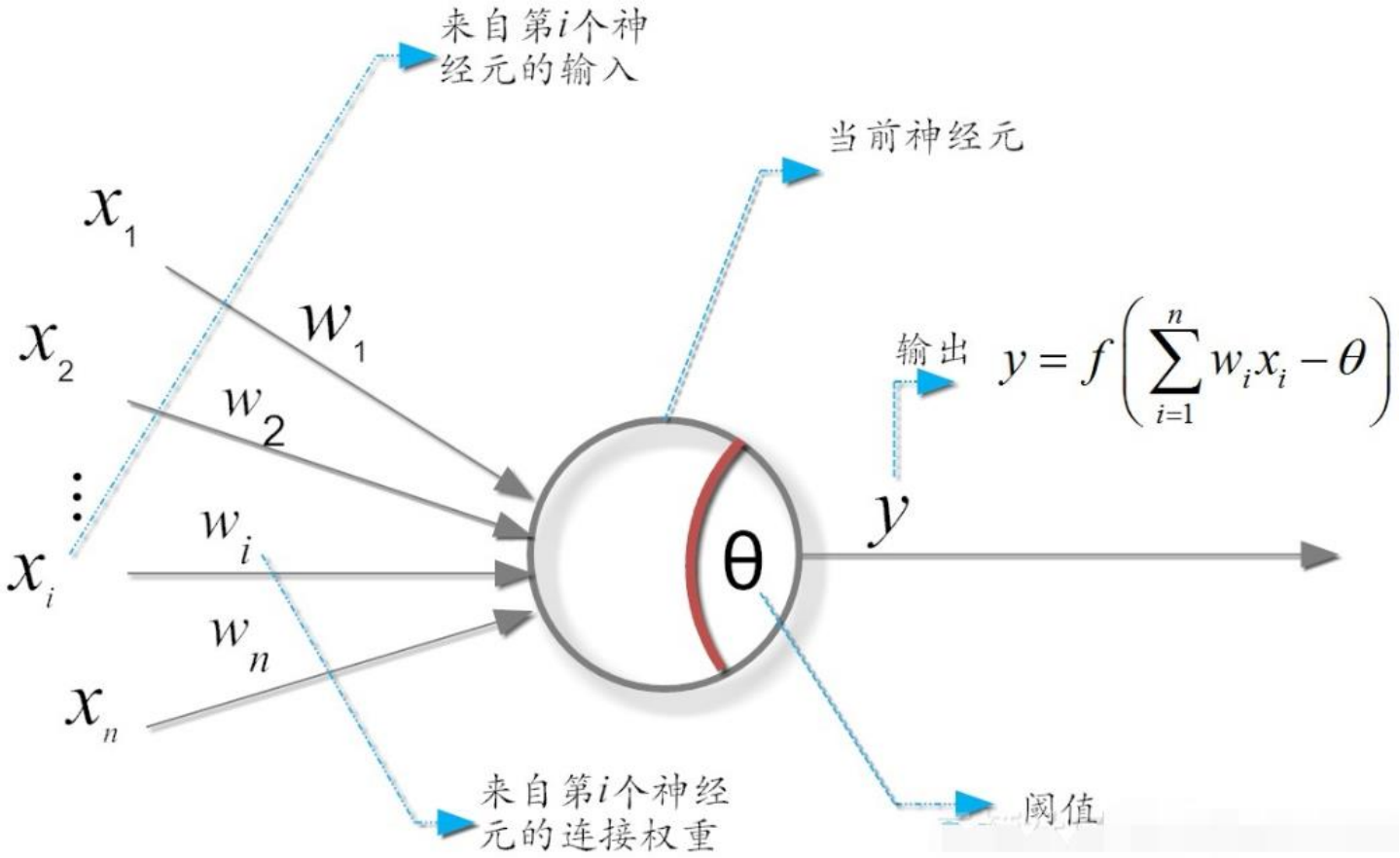


神经网络

- Def：“神经网络是有具有适应性的简单单元组成的广泛并行互连的网络，它的组织能够模拟生物神经系统对真实世界物体所做出的交互反应。”[Kohonen, 1988]
- 神经网络 (neural networks) 最重要的用途是分类。
  - 垃圾邮件识别
  - 疾病判断
  - 猫狗图片分类
- 这种能自动对输入的东西进行分类的机器，就叫做分类器。

神经网络

M-P神经元模型



其中，$f(\sum_{i=1}^{n} \omega_i x_i - \theta)$称为激活函数(activation function)。

神经网络

常用激活函数：

- 阶跃函数

$$sgn(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

- Sigmoid函数

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

若$f = sigmoid(x)$，则

$$f'(x) = f(x)(1 - f(x))$$

神经网络

感知器

- 感知器(Perceptron)由两层神经元组成，输入层接收外界输入信号后传递给输出层，输出层是M-P神经元。

$$y=f(\omega_1 x_1+\omega_2 x_2-\theta)$$

输出层

输入层

神经网络

感知器

- 给定训练数据集，权重$\omega_i(i = 1,2,\ldots,n)$以及阈值$\theta$可通过学习得到。
- 阈值θ可看作一个固定输入为-1.0的“哑节点”（dummy node）所对应的连接权重ω_{n+1}，这样权重与阈值的学习就可统一为权重的学习。
- 感知机的学习规则：

  对训练样本(x, y)，若当前感知机的输出为$\hat{y}$，则感知机权重调整：

$$\omega_i \leftarrow \omega_i + \Delta\ \omega_i$$

$$\Delta\ \omega_i = \eta(y - \hat{y})x_i$$

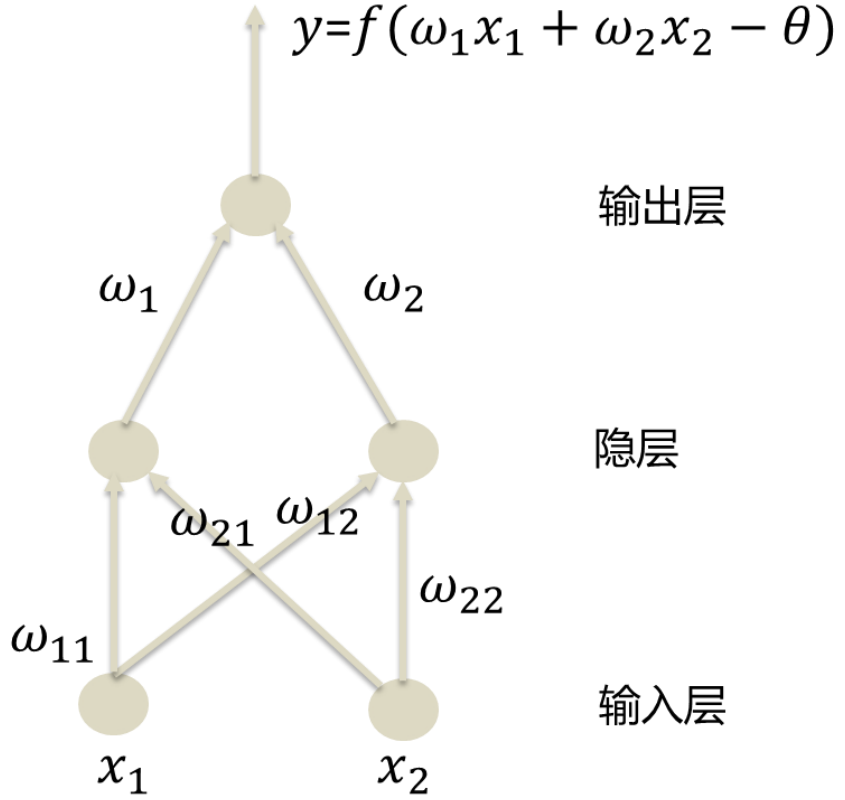
  其中，$\eta \in (0,1)$称为学习率（learning rate）。

  直到$y = \hat{y}$，感知机停止权重调整。

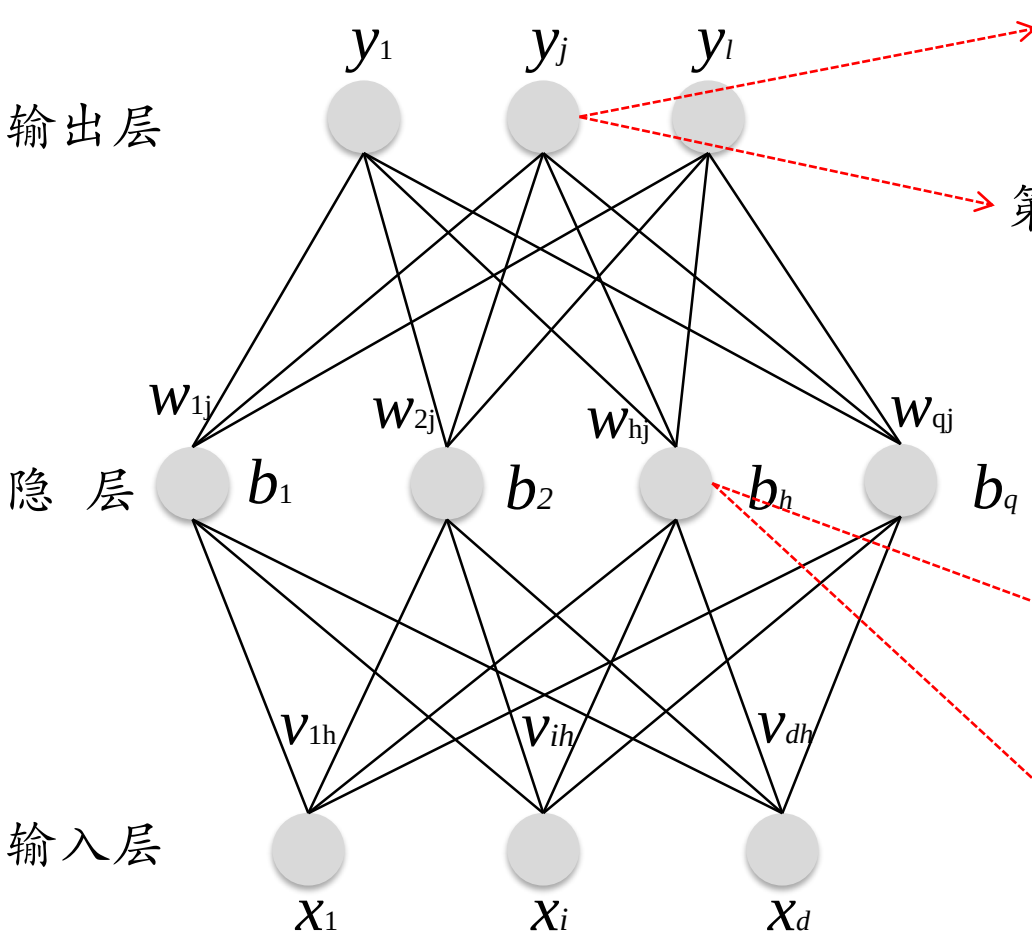
神经网络

多层网络

- 要解决非线性问题，需要考虑使用多层功能神经元。输入层与输出层之间的一层神经元，被称为隐层(hidden layer)。隐层与输出层神经元都是拥有激活函数的功能神经元。



神经网络



第j个输出层神经元的输出：

$$\hat{y}_j = f(b_j - q_j)$$

第j个输出层神经元的输入：

$$b_j = \sum_{h=1}^{g} w_{hj} b_h$$

第h个隐层神经元的输出：

$$b_h = f(a_h - g_h)$$

第h个隐层神经元的输入：

$$a_h = \sum_{i=1}^{d} v_{ih} x_i$$

神经网络

BP算法（误差逆传播算法）

- 假设隐层与输入层神经元的激活函数均为sigmoid函数
- 对训练例(x_k, y_k)，假记神经网络的输出为$\hat{y}_k = (\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_l^k)$，即

$$\hat{y}_j^k = f(\beta_j - \theta j)$$

则网络在(x_k, y_k)上的均方误差为

$$E_k = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_j^k - y_j^k)$$

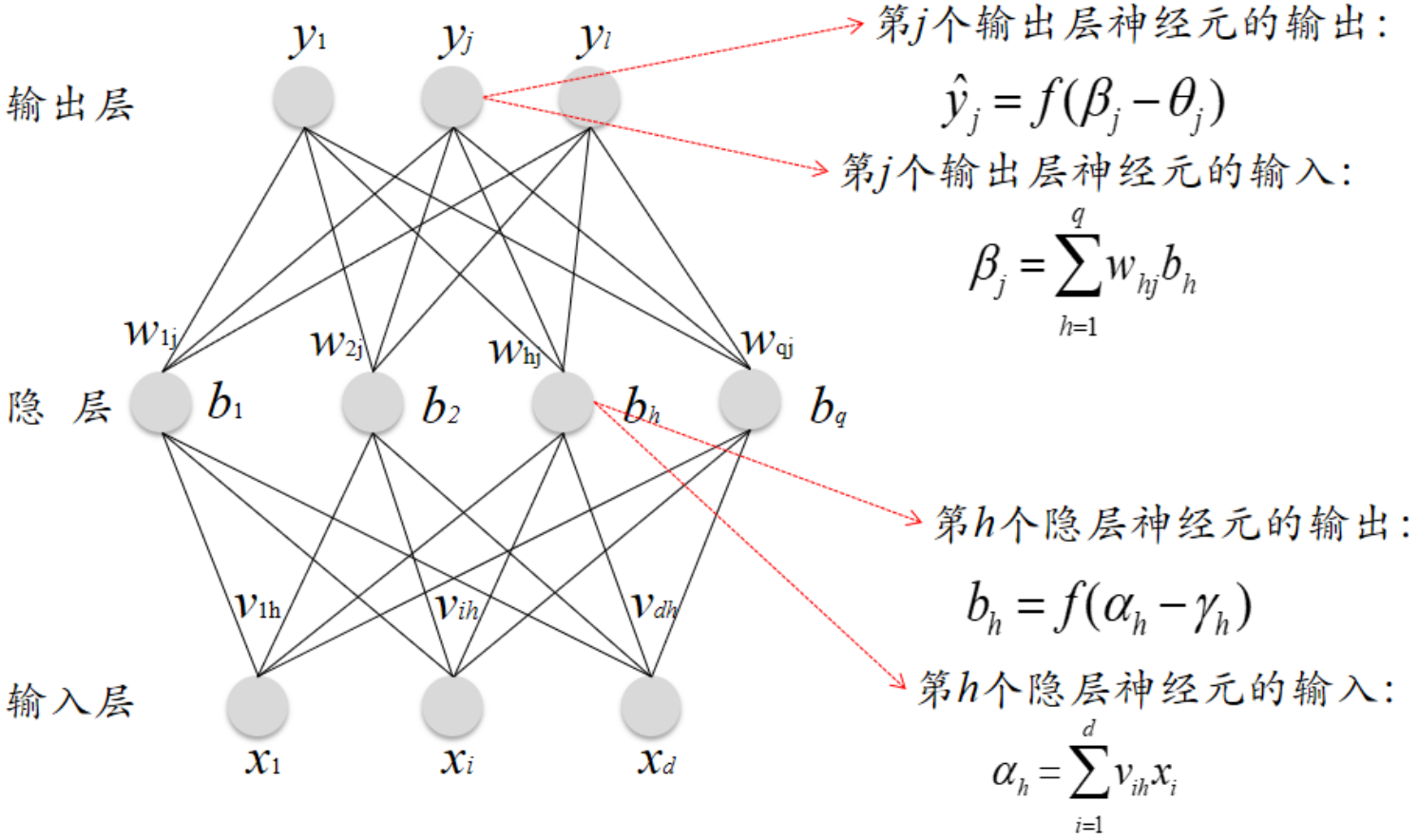
神经网络

BP算法（误差逆传播算法）推导

- $f(x) = sigmoid(x) = \frac{1}{1+e^{-x}}$
- $\hat{y} = f(\beta_j - \theta_j)$
- $E = \frac{1}{2}\sum_{j=1}^{l}(\hat{y}_j - y_j)^2$
- $\omega_i \longleftarrow \omega_i + \Delta\omega_i$

梯度下降策略：

$$\Delta\omega_i = -\eta \frac{\partial E_k}{\partial w_{hj}}$$



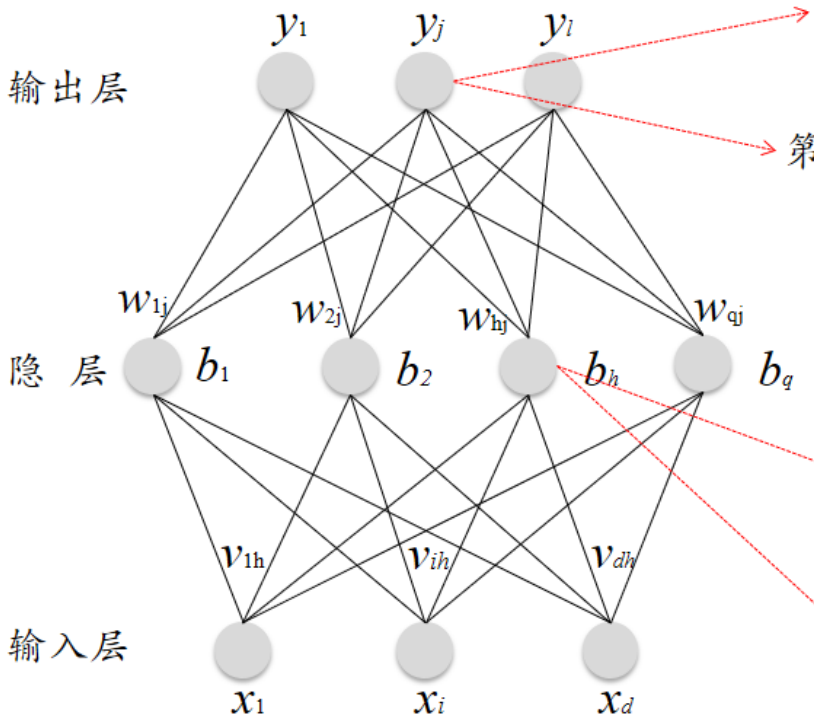
神经网络

BP算法（误差逆传播算法）推导

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial \beta_j}\frac{\partial \beta_j}{\partial w_{hj}}$$

$$\frac{\partial \beta_j}{\partial w_{hj}} = b_h$$

$$\frac{\partial f}{\partial x} = f(x)(1-f(x))$$



第j个输出层神经元的输出：

$$\hat{y}_j = f(\beta_j - \theta_j)$$

第j个输出层神经元的输入：

$$\beta_j = \sum_{h=1}^{q} w_{hj} b_h$$

第h个隐层神经元的输出：

$$b_h = f(\alpha_h - \gamma_h)$$

第h个隐层神经元的输入：

$$\alpha_h = \sum_{i=1}^{d} v_{ih} x_i$$

$$g_j = -\frac{\partial E_k}{\partial \hat{y}_j}\frac{\partial \hat{y}_j}{\partial \beta_j} = -(\hat{y}_j - y_j)f'(\beta_j - \theta_j) = \hat{y}_j(1-\hat{y}_j)(y_j - \hat{y}_j)$$

$$\Rightarrow \Delta w_{hj} = -\eta\frac{\partial E_k}{\partial w_{hj}} = \eta g_j b_h$$

神经网络

BP算法（误差逆传播算法）推导

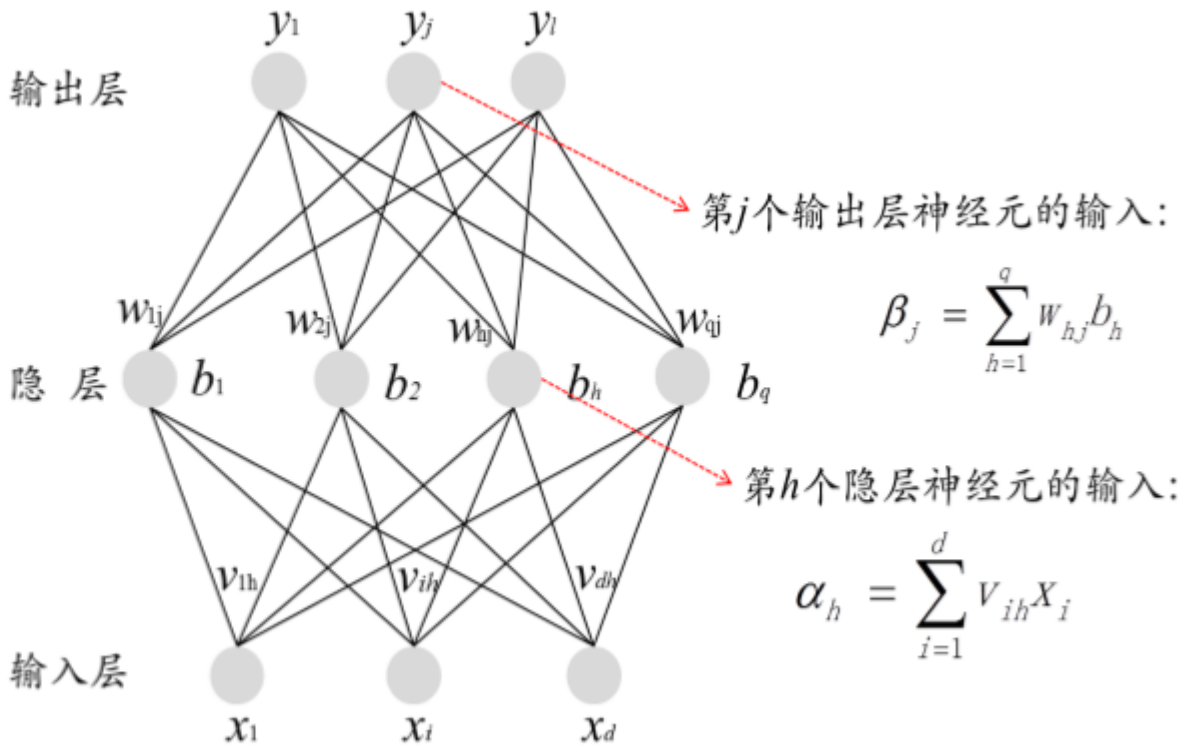
同理可得：

$$\Delta w_{hj} = \eta \hat{y}_j (1 - \hat{y}_j)(y_j - \hat{y}_j) b_h$$

$$\nabla \theta_j = -\eta \hat{y}_j (1 - \hat{y}_j)(y_j - \hat{y}_j)$$

$$\nabla v_{ih} = \eta b_h (1 - b_h) \sum_{j=1}^{l} w_{hj} g_j x_i$$

$$\nabla \gamma_h = -\eta b_h (1 - b_h) \sum_{j=1}^{l} w_{hj} g_j$$



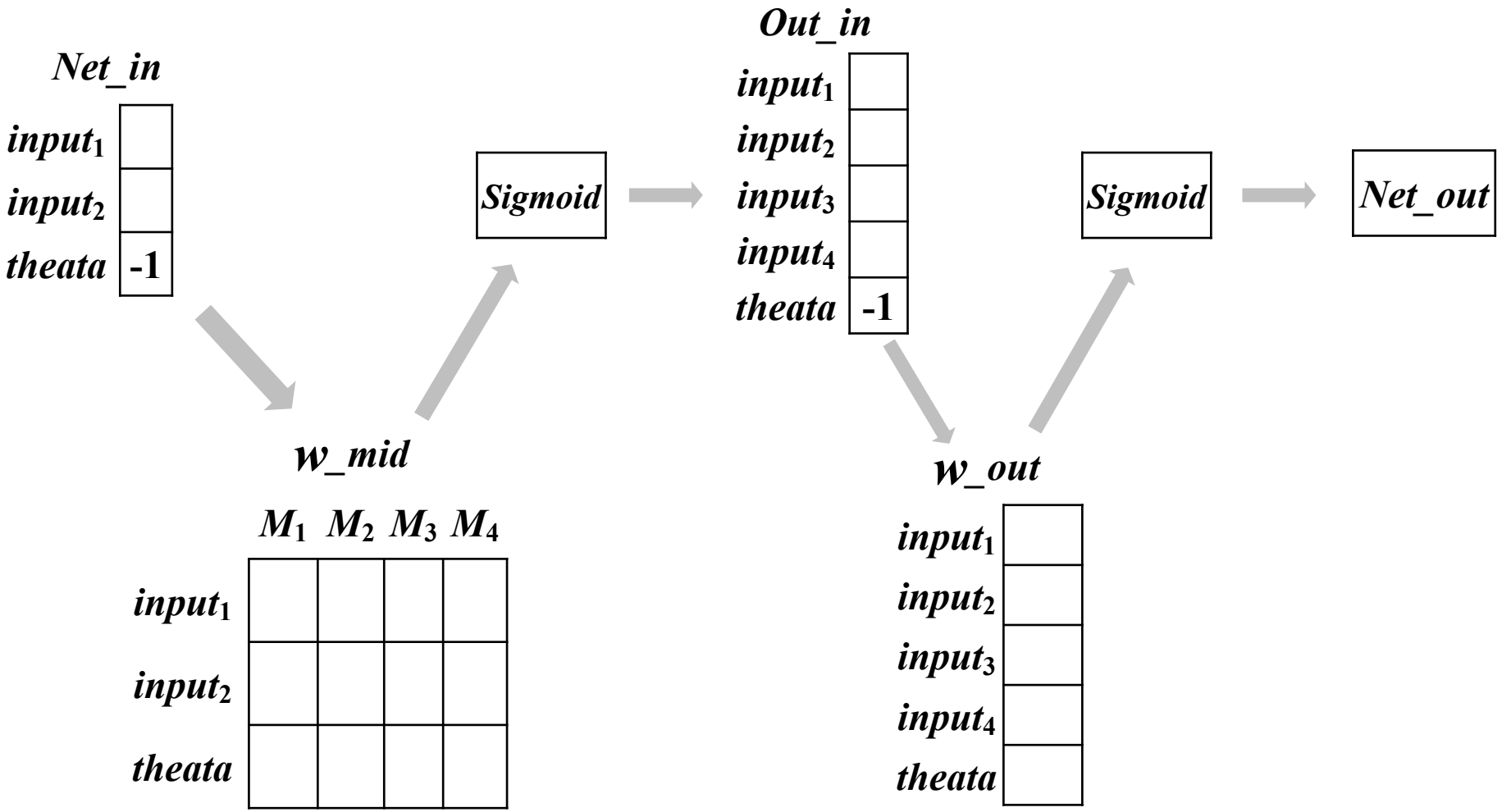
神经网络

BP算法步骤

- **输入**：训练集$D = \{(x_k, y_k)\}_{k=1}^{m}$；
  学习率η
- **过程**：
  在(0,1)范围内随机初始化网络中所有连接权和阈值
  repeat
  for all $(x_k, y_k) \in D$ do
  根据网络输入和当前参数计算当前样本的输出$\hat{y}_k$
  计算输出层神经元的梯度项g_j
  计算隐层神经元的梯度项e_h
  更新连接权ω_{hj}, v_{ih}与阈值θ_j, γ_h
  end for
  Until 达到停止条件
- **输出**：连接权与阈值确定的多层前馈神经网络

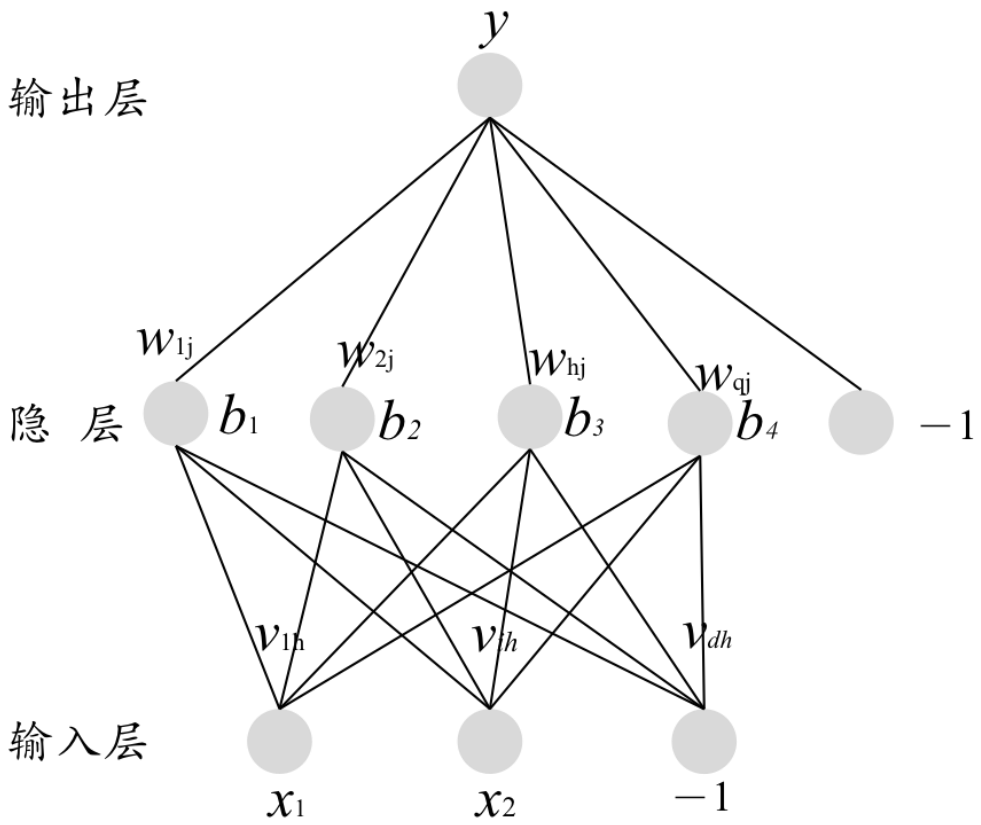
神经网络

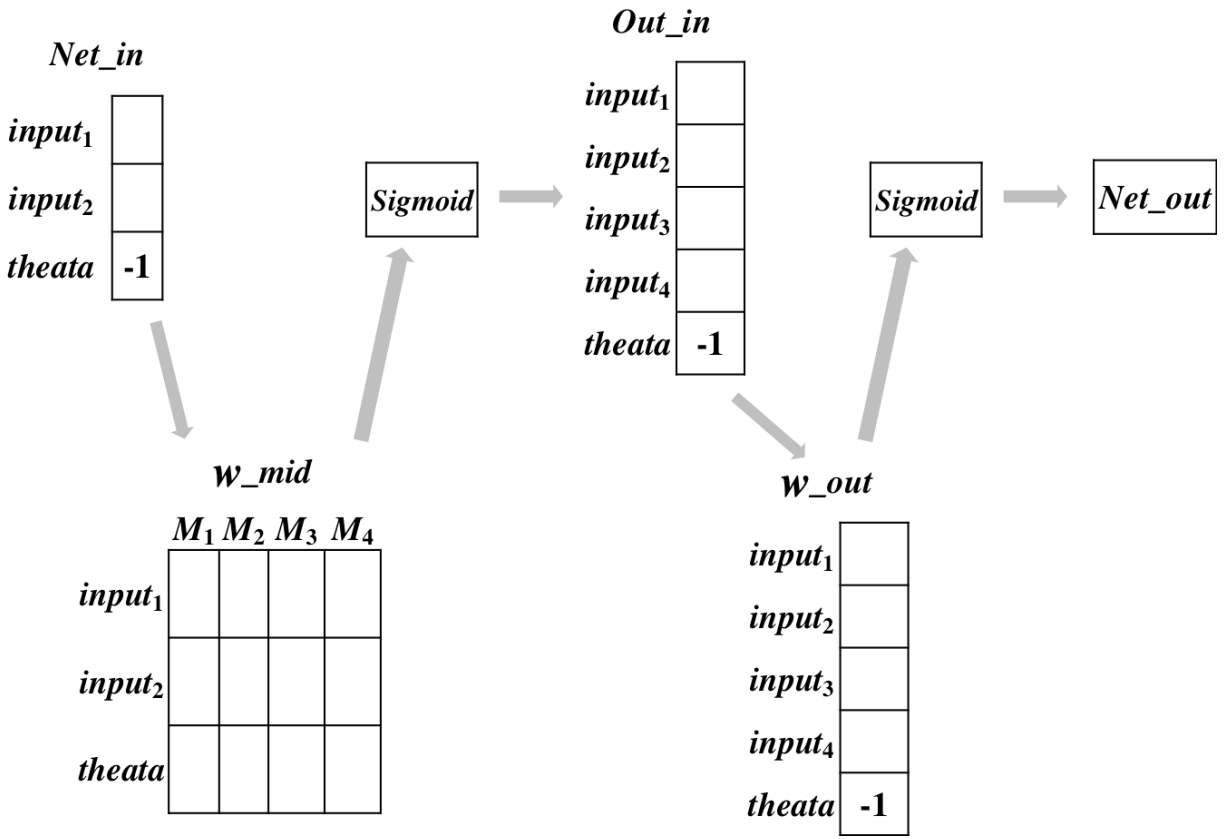
网络训练过程


Net_in
input1
input2
theata
-1
Sigmoid
Out_in
input1
input2
input3
input4
theata
-1
Sigmoid
Net_out
w_mid
M1 M2 M3 M4
input1
input2
theata
w_out
input1
input2
input3
input4
theata

神经网络

网络训练过程


y
输出层
w_{1j}
w_{2j}
w_{hj}
w_{qj}
隐 层
b_1
b_2
b_3
b_4
−1
v_{1h}
v_{ih}
v_{dh}
输入层
x_1
x_2
−1


Net_in
input1
input2
theata
-1
Sigmoid
Out_in
input1
input2
input3
input4
theata
-1
Sigmoid
Net_out
w_mid
M1 M2 M3 M4
input1
input2
theata
w_out
input1
input2
input3
input4
theata
-1

神经网络

Python实现

- from sklearn.neural_network import MLPClassifier
- hidden_layer_sizes 参数：输入为元组，表示在神经网络中每层的神经元数量，其中元组中的第n个元素表示MLP模型第n层的神经元数量
- max_iter参数：最大迭代次数

- fit训练模型
- predict预测

神经网络

Python实现

- coefs_是权重矩阵的列表，其中索引i处的权重值表示第i层到第i+1层之间的权重
- intercepts_是偏差向量的列表，其中索引i处的向量表示添加到第i+1层之间的偏差值

- 另，sklearn中自带混淆矩阵及分类报告
- from sklearn.metrics import classification_report,confusion_matrix

神经网络

优点

- 有良好的自组织学习功能。
- 类神经网络可以建构非线性的模型，模型的准确度高。
- 类神经网络有良好的推广性，对于未知的输入亦可得到正确的输出。
- 类神经网络可以接受不同种类的变量作为输入，适应性强。
- 对异常值不敏感。
- 对噪声数据有比较高的承受能力。
- 以用于分类预测问题。
- 对数据的基本关系几乎不需要做出假设。

神经网络

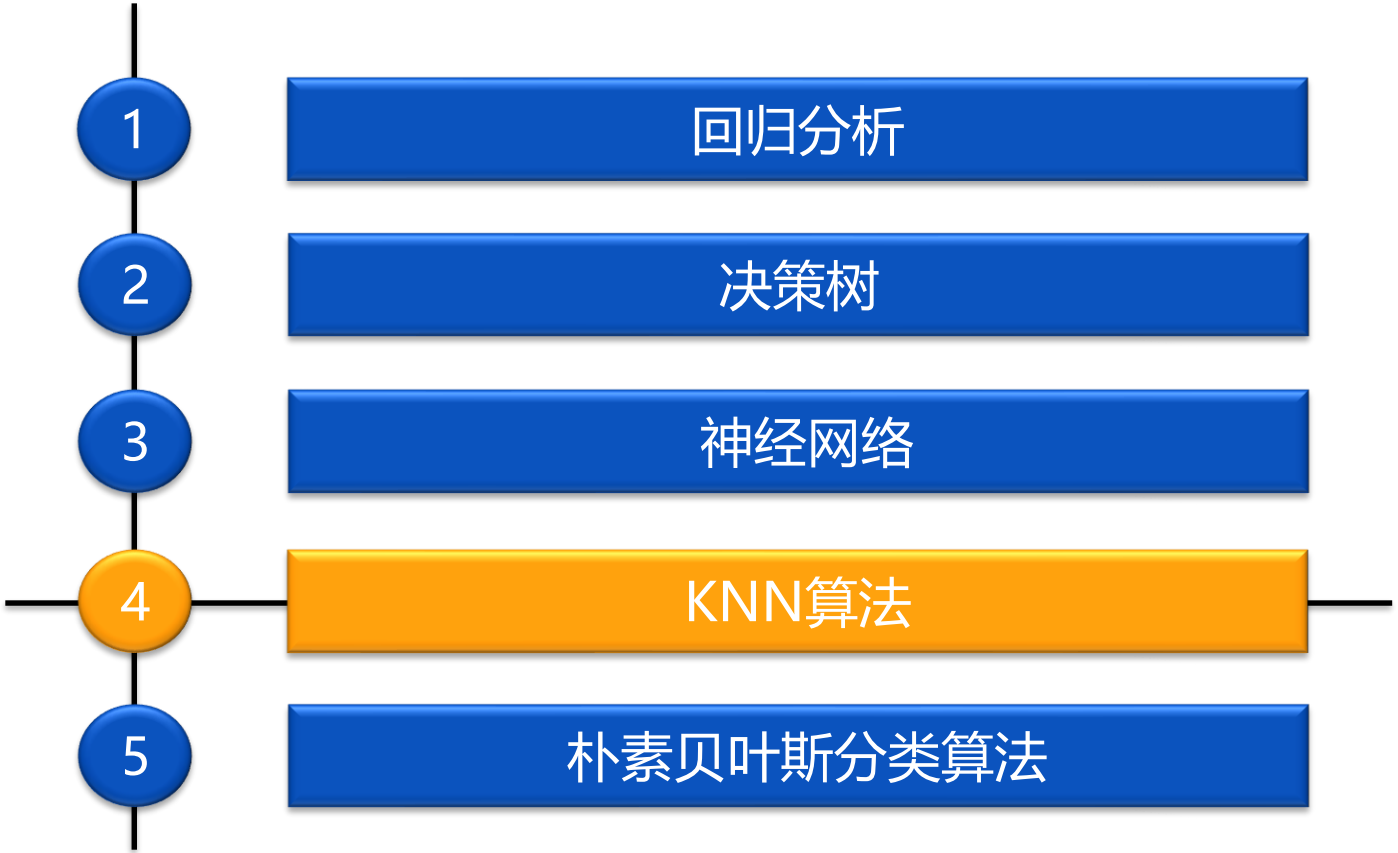
缺点

- 计算量大，训练速度慢，特别是在网络拓扑结构复杂的情况下。
- 容易出现过度拟合。
- 由于其结构的复杂性和结论的难以解释性，神经网络在商业实践中远远没有回归和决策 树应用的广泛，人们对它的理解、接纳还有待提高

其他神经网络

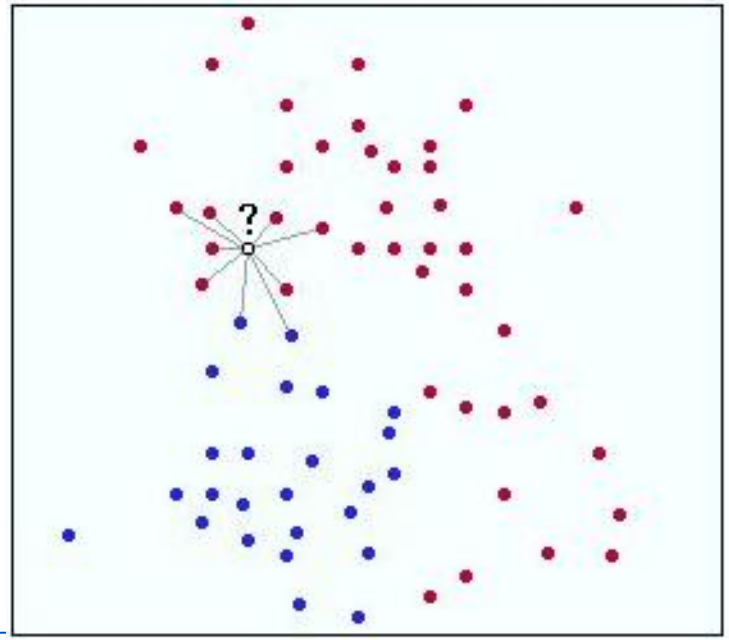
- 卷积神经网络(Convolutional Neual Network,CNN)是一种前馈神经网络，与BP神经网络不同的是，它包括卷积层(alternating convolutional layer)和池层(pooling layer)，在图像处理方面有很好的效果，经常用作解决计算机视觉问题。
- 递归神经网络(RNN)分为时间递归神经网络(Recurrent Neural Network)和结构递归神经网络(Recursive Neural Network)。
- RNN主要用于处理序列数据。在BP神经网络中，输入层到输出层各层间是全连接的，但同层之间的节点却是无连接的。这种网络对于处理序列数据效果很差，忽略了同层间节点的联系，而在序列中同层间节点的关系是很密切的。

目录



kNN算法概述

- kNN算法是k-Nearest Neighbor Classification的简称，即k-近邻分类算法。它的思想很简单：一个样本在特征空间中，总会有k个最相似（即特征空间中最邻近）的样本。其中，大多数样本属于某一个类别，则该样本也属于这个类别。
- 近朱者赤，近墨者黑。
- 是理论上比较成熟的方法，也是最简单的机器学习算法之一。
- 行业应用：
  - 客户流失预测
  - 欺诈侦测等（更适合于稀有事件的分类问题）



kNN算法

- 计算步骤如下：

  1）算距离：给定测试对象，计算它与训练集中的每个对象的距离

  2）找邻居：圈定距离最近的k个训练对象，作为测试对象的近邻

  3）做分类：根据这k个近邻归属的主要类别，来对测试对象分类

- kNN是一种懒惰算法，平时不好好学习，考试（对测试样本分类）时才临阵磨枪（临时去找k个近邻）。懒惰的后果：构造模型很简单，但在对测试样本分类地的系统开销大，因为要扫描全部训练样本并计算距离。

kNN算法

算距离

- 什么是合适的距离衡量？

  距离越近应该意味着这两个点属于一个分类的可能性越大。

- 计算的距离衡量包括欧式距离、夹角余弦等。

  欧式距离：

$$d(x,y)=\sqrt{\sum_{k=1}^{n}(x_k-y_k)^2}$$

  注：有些距离测量方法会受维数的影响，特别是欧氏距离在属性数增加时判断的能力减弱。

- 对于文本分类来说，使用余弦(cosine)来计算相似度就比欧式(Euclidean)距离更合适。

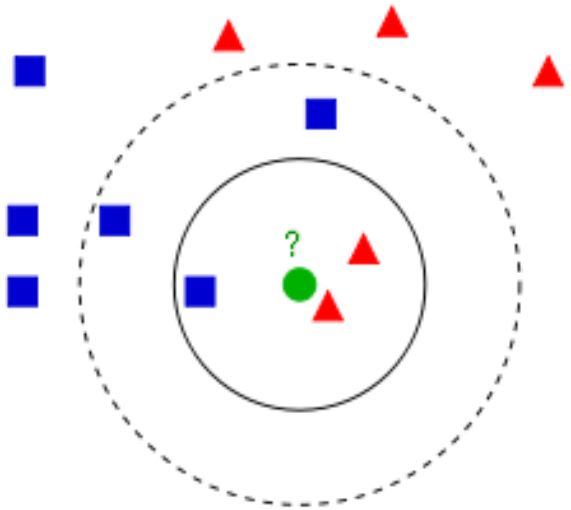
kNN算法

做分类

- 投票决定：少数服从多数，近邻中哪个类别的点最多就分为该类。

- 加权投票法：根据距离的远近，对近邻的投票进行加权，距离越近则权重越大（权重为距离平方的倒数）

- 投票法没有考虑近邻的距离的远近，距离更近的近邻也许更应该决定最终的分类，所以加权投票法更恰当一些。



kNN算法实现流程

a
- 计算已知类别数据集中的点与当前点之间的距离；

b
- 按照距离递增次序排序；

c
- 选取与当前点距离最小的k个点；

d
- 确定前k个点所在类别对应的出现频率；

e
- 返回前k个点出现频率最高的类别作为当前点的预测分类。

kNN算法

- 1、优点

  简单，易于理解，易于实现，无需估计参数，无需训练

  适合对稀有事件进行分类（例如当流失率很低时，比如低于0.5%，构造流失预测模型）

  特别适合于多分类问题(multi-modal,对象具有多个类别标签)，例如根据基因特征来判断其功能分类，kNN比SVM的表现要好

- 2、缺点

  懒惰算法，对测试样本分类时的计算量大，内存开销大，评分慢

  可解释性较差，无法给出决策树那样的规则。

kNN算法

Python实现

- sklearn库中提供KNeighborsClassifier实现kNN算法，此外，还提供RadiusNeighborsClassifier（非均匀采样时比较合适，以半径为选取方法）做最近邻分类
- sklearn.neighbors.KNeighborsClassifier(n_neighbors=5 #邻居数，默认为5
  , weights='uniform' #用于预测的权重方法
  , algorithm='auto' #用于计算最近邻的算法（ball_tree、kd_tree、brute、auto）
  , leaf_size=30 #传递给BallTree 或KDTree 叶大小
  , p=2 #
  , metric='minkowski' #用于树的度量距离
  , metric_params=None #度量参数
  , **kwargs)
- from sklearn.neighbors import KNeighborsClassifier

kNN算法

练习

- 使用kNN算法对iris数据集进行分类

目录

朴素贝叶斯

定义

- 朴素贝叶斯分类器是基于贝叶斯定理与特征条件独立同分布假设的分类方法。
- 贝叶斯公式：

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

- 简单的说，贝叶斯定理是基于假设的先验概率、给定假设下观察到不同数据的概率，提供了一种计算后验概率的方法。
- 独立同分布：指随机过程中，任何时刻的取值都为随机变量，如果这些随机变量服从同一分布，并且互相独立，那么这些随机变量是独立同分布。
- 即：

$$P(x|c_k)=P(x_1,\mathrm{L}\ ,x_n|c_k)=\prod_{j=1}^{n}P(x_j|c_k)$$

朴素贝叶斯

原理（一）

- 对于贝叶斯网络分类器，若某一待分类的样本D，其分类特征值为 $x=(x_1,x_2,...,x_n)$，则样本D 属于类别yj 的概率 $P(Y=y_j|X=x_1,\mathrm{L}\ X=x_n),(j=1,\mathrm{L},m)$ 应满足下式： $\max P(Y=y_j|X=x)$
- 而由贝叶斯公式：

$$P(Y=y_j|X=x)=\frac{P(X=x/Y=y_j)P(Y=y_j)}{P(X=x)}$$

- 得

$$\max P(Y=y_j|X=x)\ \Rightarrow \max P\left(Y=y_j\right)P(X=x|Y=y_j)$$

- 因此估计 $P(Y=y_j|X=x)$ 的问题就转化为如何基于训练集数据D来估计先验概率 $P\left(Y=y_j\right)$ 和条件概率 $P(X=x|Y=y_j)$

朴素贝叶斯

原理（二）：$P\left(Y=y_j\right)$的计算

- 大数定理：

设μ_n是n次独立试验中事件A发生的次数，且事件A在每次试验中发生的概率为P，则对任意正数ε，有：

$$\lim_{n\to\infty} P(|\frac{\mu_n}{n}-p<\varepsilon|)=1$$

该定律是切贝雪夫大数定律的特例，其含义是，当n足够大时，事件A出现的频率将几乎接近于其发生的概率，即频率的稳定性。

- 当n足够大时，事件A出现的频率将几乎接近于其发生的概率，即频率的稳定性。
- 根据大数定理，当训练集包含充足的独立同分布样本时，$P\left(Y=y_j\right)$可通过各类样本出现的频率来进行估计。

朴素贝叶斯

原理（三）： $P(X=x|Y=y_j)$

- 在朴素贝叶斯算法中，假设样本独立同分布，此时，

$$P\left(X=x|Y=y_j\right)=P\left(X=x_1,\mathrm{L}\ ,X=x_n|Y=y_j\right)$$

$$=\prod_{i=1}^{n}P\left(X=x_i|Y=y_j\right)$$

- 所以，

$$\max P\left(Y=y_j\right)P(X=x|Y=y_j)$$

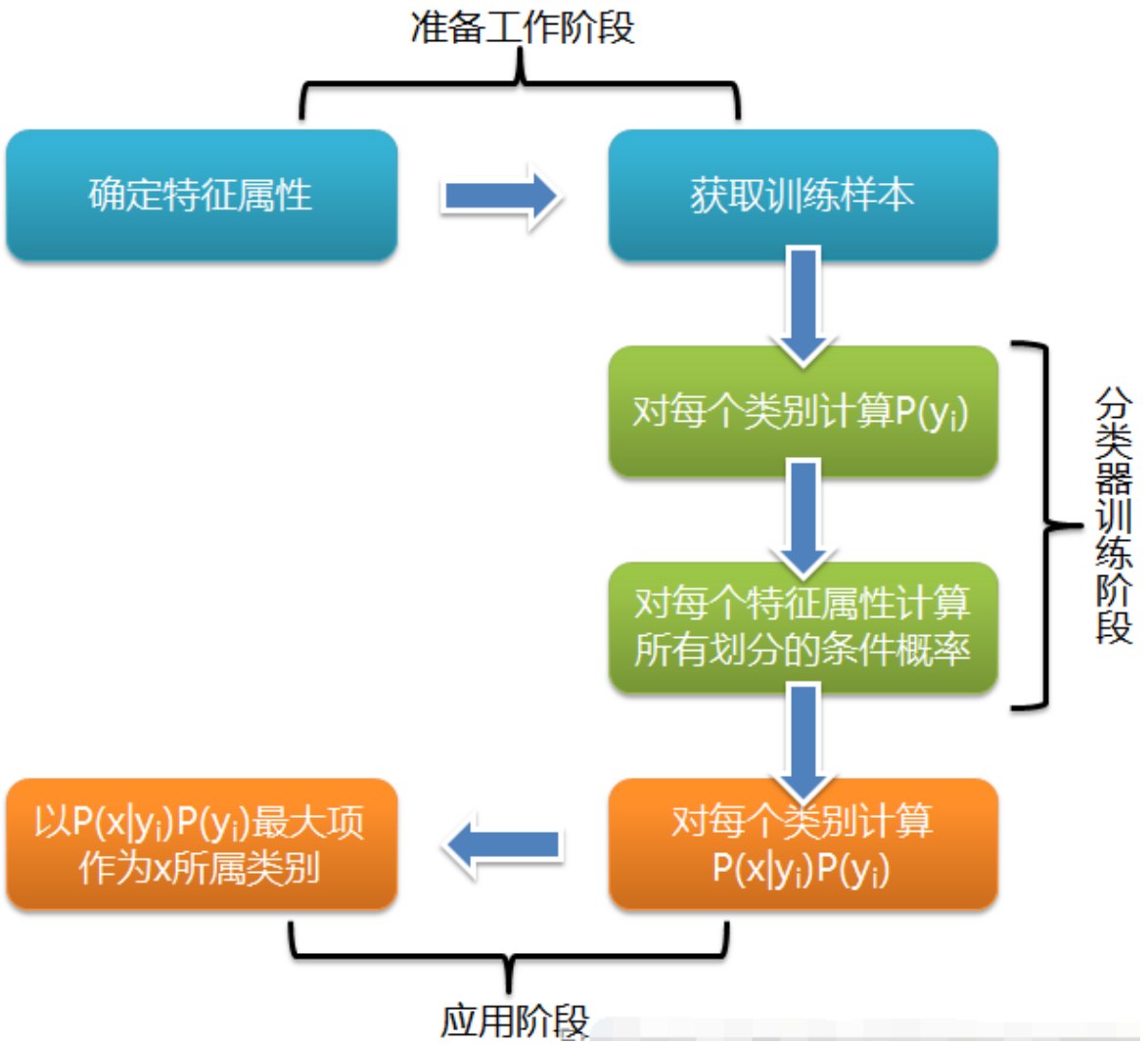
$$=\max P\left(Y=y_j\right)\prod_{i=1}^{n}P\left(X=x_i|Y=y_j\right)$$

- 假设共有 m种标签，我们只需计算 $P(y_k)P(x_1,\cdots,x_n|y_k),k=1,2,\cdots,m$，取最大值作为预测的分类标签，即：

$$\hat{y}=\arg\max_k P(y)\prod_{i=1}^{n}P(x_i|y_k)$$

朴素贝叶斯

算法处理流程



高斯朴素贝叶斯

- 原始的朴素贝叶斯只能处理离散数据，当 $X_1,\cdots,X_n$ 是连续变量时，我们可以使用高斯朴素贝叶斯（Gaussian Naive Bayes）完成分类任务。当处理连续数据时，一种经典的假设是：与每个类相关的连续变量的分布是基于高斯分布的，故高斯贝叶斯的公式如下：

$$P(x_i = v \mid y_k) = \frac{1}{\sqrt{2\pi\sigma_{y_k}^2}} \exp\left(-\frac{(v-\mu_{y_k})}{2\sigma_{y_k}^2}\right)$$

- 其中 μ_y ， $\sigma_{y_k}^2$ 表示表示全部属于类 y_k 的样本中变量 X_i 的均值和方差

多项式朴素贝叶斯

- 多项式朴素贝叶斯（Multinomial Naïve Bayes）经常被用于处理多分类问题，比起原始的朴素贝叶斯分类效果有较大的提升。其公式如下：

$$P(x_i \mid y_k) = \frac{N_{y_k i} + \alpha}{N_y + \alpha n}$$

- 其中 $N_{y_k i} = \sum_{x \in T} x_i$ 表示在训练集T 中类 y_k 具有特征 i的样本的数量， $N_y = \sum_{i=1}^{|T|} N_{yi}$ 表示训练集 T中类 y_k 的特征总数。平滑系数 $\alpha > 0$ 防止零概率的出现，当 $\alpha = 1$ 称为拉普拉斯平滑，而 $\alpha < 1$ 称为Lidstone平 滑。

朴素贝叶斯

Python实现

- naive Bayes is a decent classifier, but a bad estimator
- 高斯朴素贝叶斯
- 构造方法：sklearn.naive_bayes.GaussianNB
- GaussianNB 类构造方法无参数，属性值有：
  - class_prior_ #每一个类的概率
  - theta_ #每个类中各个特征的平均
  - sigma_ #每个类中各个特征的方差
- 注：GaussianNB 类无score 方法

朴素贝叶斯

Python实现

- 多项式朴素贝叶斯——用于文本分类
- 构造方法：

```
sklearn.naive_bayes.MultinomialNB(alpha=1.0 #平滑参数
                                  , fit_prior=True #学习类的先验概率
                                  , class_prior=None) #类的先验概率
```

Thank you!

泰迪科技：www.tipdm.com
热线电话：40068-40020