分类算法

什么是分类

一些常见的概念：

- 训练集(Training data)：用于训练模型(的参数，如神经网络的各层权重和偏置，线性回归分析的系数);
- 验证集(Validation data)：用于调整超参数(Hyper-Parameters，如神经网络的宽度和深度、学习率等);
- 测试集(Test data)：用于评价模型本身的有效性（准确率等）
- 训练误差(Training error)：分类器在训练集上的误差。
- 泛化误差(Generalization error，out-of-sample error)：分类器在未见样本（不在训练集中的样本）上的误差。

对于数据集的划分

- 划分法: 训练集与测试集
  把样本划分成2个独立的数据集合, 如, 训练集 (2/3), 测试集(1/3)。
  适用于大规模的数据样本。

- 交叉验证(Cross-validation)
  把数据集合划分成k 个子样本;
  使用k - 1 个子样本作为训练集，另一个作为测试样本—k-折交叉验证。
  适用于中等规模的数据。

- 留一测试(Leave One Out， k = n)
  适用于小规模数据。

泛化误差的偏差/方差分解、过拟合、欠拟合

以回归为例

$$E[(h(x^*) - y^*)^2]$$

$$= E[h(x^*)^2 - 2h(x^*)y^* + y^{*2}]$$

$$= E[h(x^*)^2] - 2E[h(x^*)]E[y^*] + E[y^{*2}]$$

$$= E[(h(x^*) - \bar{h}(x^*))^2] + \bar{h}(x^*)^2$$

$$-2\bar{h}(x^*)f(x^*)$$

$$+E[(y^* - f(x^*))^2] + f(x^*)^2$$

$$= E[(h(x^*) - \bar{h}(x^*))^2] + (\bar{h}(x^*) - f(x^*))^2 + E[(y^* - f(x^*))^2]$$

$$= E[(h(x^*) - \bar{h}(x^*))^2] + (\bar{h}(x^*) - f(x^*))^2 + E[\varepsilon^2]$$

$$= E[(h(x^*) - \bar{h}(x^*))^2] + (\bar{h}(x^*) - f(x^*))^2 + \sigma^2$$

- 我们来看上式的每一项的意义：
- 第1项为 $E[(h(x^*) - \bar{h}(x^*))^2]$：模型的方差，它度量了在不同训练集上学习到的不同模型之间的性能差异。
- 第2项为 $(\bar{h}(x^*) - f(x^*))^2$： 模型的期望预测值与真实函数值的偏差的平方，它度量了学习到的平均模型与真实函数之间的差异
- 第3项为σ^2：为固有观察噪声的方差。

- 期望泛化误差=模型方差+模型偏差^2+噪声^2.
  - 其中噪声为固有特性，不可优化，任何模型的期望误差都将大于等于σ^2！
- 我们当然希望同时最小化模型方差与模型偏差，但往往是鱼与熊掌不可兼得！

过拟合与欠拟合

- 过拟合：模型过于复杂（模型的表达力或capacity过高：例如神经网络的参数过多，决策树过深、过宽等等，而训练样本相对较少）。
  考虑变量数大于方程数的线性方程组。
- 欠拟合：模型过于简单（模型的表达力或capacity不够）
  考虑以线性模型去拟合非线性模型。

判别模型与生成模型

关于判别模型：

判别（Discriminative）模型：$Pr(w|\mathbf{x})$

- 针对w，选择一个合适的概率分布$Pr(w)$（先验分布），其中该分布的参数是关于$\mathbf{x}$的函数。
  - 例如，若类状态w是连续的，则我们可以假设$Pr(w)$是一个高斯分布（正态分布，见下式），其中均值是一个关于$\mathbf{x}$的函数。
- 因为$Pr(w|\mathbf{x})$的值还依赖于分布的参数集合θ，因此我们记为$Pr(w|\mathbf{x}, \theta)$，并称之为后验分布（the posterior distribution）。

$$Pr(w) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(w-\mu)^2}{2\sigma^2}}$$

判别(Discriminative)模型：$Pr(w|\mathbf{x})$

- 然后我们利用训练数据 $\{\mathbf{x}_i, w_i\}$ $i=1, 2, \ldots, n$，利用最大似然法 (the maximum likelihood , ML), 最大后验概率法 (maximum a posteriori, MAP), 或贝叶斯方法来学习参数集合θ.
- 推导：给定新数据$\mathbf{x}$，直接确定其类状态的后验概率 $Pr(w|\mathbf{x}, \theta)$.

关于生成模型：

生成(Generative)模型 ： $Pr(\mathbf{x}|w)$

- 针对$\mathbf{x}$，选择一个合适的概率分布$Pr(\mathbf{x})$ 。其中该分布的参数是关于w的函数。
  - 例如，若$\mathbf{x}$是离散的，则我们可以假设$Pr(\mathbf{x})$是一个离散分布函数(a categorical distribution，见下式)，其中参数向量是一个关于w的函数。
- 同样，因为$Pr(\mathbf{x}|w)$的值还依赖于分布的参数集合θ，因此我们记为 $Pr(\mathbf{x}|w, \theta)$，并称之为似然(the likelihood)。

$$Pr(x = e_k) = \prod_{j=1}^{K} \lambda_j^{x_j} = \lambda_k$$

生成(Generative)模型： $Pr(\mathbf{x}|w)$

- 然后我们利用训练数据 $\{\mathbf{x}_i, w_i\}$ $i=1, 2, \ldots, n$，利用ML法，MAP法，或贝叶斯方法来学习参数集合 θ.
- 推导：给定新的数据**x**，计算其类状态的后验概率，这里用到了贝叶斯定理：

$$Pr(w|x) = \frac{Pr(x|w)\ Pr\ (w)}{\sum_{w=0}^{i} Pr\ (x|w) Pr\ (w)}$$

用哪种模型：

- 区分模型推导时更简单；
- 生成模型更加复杂一些：x比w往往高维，因此参数往往更多，训练困难；
- 生成模型反映了数据的实际产生过程，如果想把数据的产生过程集成到模型中，则应该考虑生成模型；(考虑如日中天的生成对抗网络Generative Adversarial Networks, GAN)
- 如果训练数据中有大量的遗失数据，则应该考虑生成模型；
- 生成模型更容易集成背景（专家知识）。

决策树

相关概念

属性选择度量：信息增益(ID3/C4.5)

- **启发式策略**：选择具有最高信息增益的属性。
- **期望信息**：设样本集合s含有s_i个类为C_i的元组，$i = \{1, \ldots, m\}$，则对一个给定的样本分类所需的期望信息是：

$$I(s_1, s_2, \ldots, s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{s_i}{s}$$

- **熵**（Entropy）：具有值$\{a_1, a_2, \ldots, a_v\}$的属性$A$的熵E(A)为属性A导致的s的划分的期望信息的加权平均和。：

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + \ldots + s_{mj}}{s} I(s_{1j}, \ldots, s_{mj})$$

- **信息增益**：在A上分枝将获得的信息增益是

$$Gain(A) = I(s_1, s_2, \ldots, s_m) - E(A)$$

一个例子

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

计算分类所需的期望信息：

$$I(s_1,s_2,...,s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{s_i}{s}$$

$$= -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$$

$$= 0.4098 + 0.5305$$

$$= 0.9403$$

- **属性 age的熵的计算：**

$$E(A)=\sum_{j=1}^{v}\frac{s1j+\ldots+smj}{s}I(s1j,\ldots,smj)=\frac{s11+s21}{s}I(s11,s21)+\frac{s12+s22}{s}I(s12,s22)+\frac{s13+s23}{s}I(s13,s23)$$

$$=\frac{5}{14}\times 0.971+\frac{4}{14}\times 0+\frac{5}{14}\times 0.971=0.6946$$

$$I(s1j,s2j)=-\sum_{i=1}^{2}\frac{sij}{Sj}\log_2\frac{sij}{Sj}$$

$$I(s11,s21)=-\frac{s11}{s1}\log_2\frac{s11}{s1}-\frac{s21}{s1}\log_2\frac{s21}{s1}=-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5}=0.971$$

$$I(s12,s22)=-\frac{s12}{s2}\log_2\frac{s12}{s2}-\frac{s22}{s2}\log_2\frac{s22}{s2}=-\frac{4}{4}\log_2\frac{4}{4}-\frac{0}{4}\log_2\frac{0}{4}=0$$

$$I(s13,s23)=-\frac{s13}{s3}\log_2\frac{s13}{s3}-\frac{s23}{s3}\log_2\frac{s23}{s3}=-\frac{3}{5}\log_2\frac{3}{5}-\frac{2}{5}\log_2\frac{2}{5}=0.971$$

属性age的信息增益：

$$Gain(Age)=I(S1,S2)-E(Age)$$

$$=0.940-0.6946=0.246$$

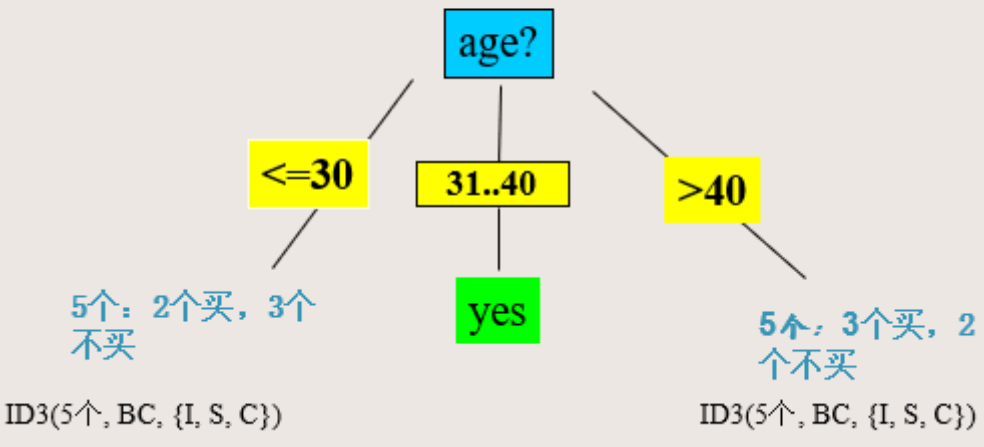
同样，对其它属性有：

$$Gain(income)=0.029$$

$$Gain(student)=0.151$$

$$Gain(credit\_rating)=0.048$$

所以我们有决策树：



下一步递归调用ID3： 针对年龄≤30的样本集测试剩余的属性

计算对≤30的样本集进行分类所需的信息：其中2个买（类Yes或1），3个不买（类No或2），所以有：

$$I(s_1,s_2)=-\frac{s_1}{s}\log_2\frac{s_1}{s}-\frac{s_2}{s}\log_2\frac{s_2}{s}=-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5}=0.971$$

计算属性student的熵E(student)： 学生（属性值1）有2个，都买，即属于类Yes或1，非学生（属性值2）有3个，属于类No或2。

$$E(Stu)=\sum_{j=1}^{v}\frac{s_{1j}+\ldots+s_{mj}}{s}I(s_{1j},\ldots,s_{mj})=\frac{s_{11}+s_{21}}{s}I(s_{11},s_{21})+\frac{s_{12}+s_{22}}{s}I(s_{12},s_{22})$$

$$=\frac{2}{5}\times I(s_{11},s_{21})+\frac{3}{5}\times I(s_{12},s_{22})=0$$

$$I(s_{11},s_{21})=-\frac{s_{11}}{s_1}\log_2\frac{s_{11}}{s_1}-\frac{s_{21}}{s_1}\log_2\frac{s_{21}}{s_1}=-\frac{2}{2}\log_2\frac{2}{2}-\frac{0}{2}\log_2\frac{0}{2}=0$$

$$I(s_{12},s_{22})=-\frac{s_{12}}{s_2}\log_2\frac{s_{12}}{s_2}-\frac{s_{22}}{s_2}\log_2\frac{s_{22}}{s_2}=-\frac{0}{3}\log_2\frac{0}{3}-\frac{3}{3}\log_2\frac{3}{3}=0$$

计算属性student的熵E(student)： 学生（属性值1）有2个，都买，即属于类Yes或1，非学生（属性值2）有3个，属于类No或2。

$$E(Stu)=\sum_{j=1}^{v}\frac{s_{1j}+\ldots+s_{mj}}{s}I(s_{1j},\ldots,s_{mj})=\frac{s_{11}+s_{21}}{s}I(s_{11},s_{21})+\frac{s_{12}+s_{22}}{s}I(s_{12},s_{22})$$

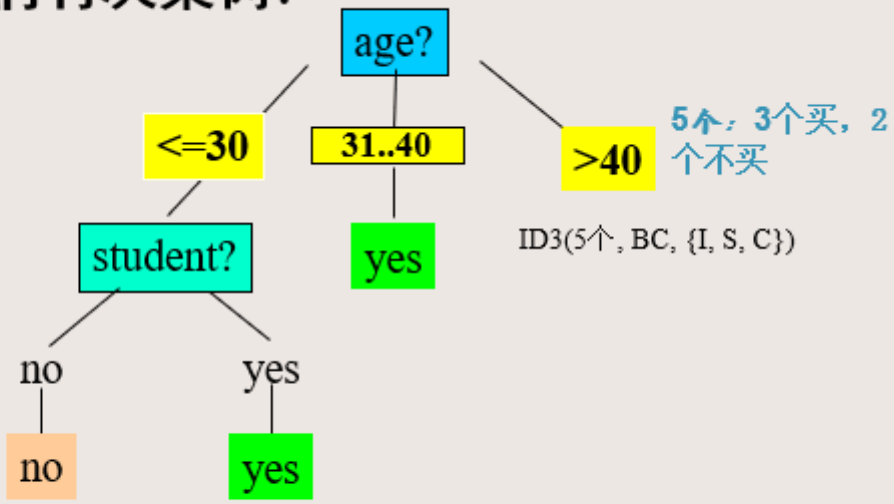
$$=\frac{2}{5}\times I(s_{11},s_{21})+\frac{3}{5}\times I(s_{12},s_{22})=0$$

$$I(s_{11},s_{21})=-\frac{s_{11}}{s_1}\log_2\frac{s_{11}}{s_1}-\frac{s_{21}}{s_1}\log_2\frac{s_{21}}{s_1}=-\frac{2}{2}\log_2\frac{2}{2}-\frac{0}{2}\log_2\frac{0}{2}=0$$

$$I(s_{12},s_{22})=-\frac{s_{12}}{s_2}\log_2\frac{s_{12}}{s_2}-\frac{s_{22}}{s_2}\log_2\frac{s_{22}}{s_2}=-\frac{0}{3}\log_2\frac{0}{3}-\frac{3}{3}\log_2\frac{3}{3}=0$$

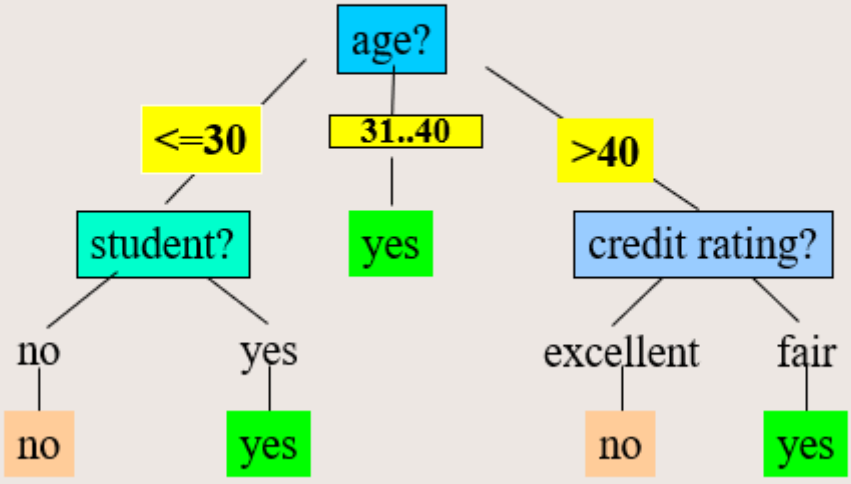
- 因此，*Gain*(student)=I(s_1,s_2)-E(stu)=0.971
- 同样，计算其它几个属性的增益：
  - Gain(income)=略
  - Gain(credit_rating)=略
- 属性Student的增益最大。

所以我们有决策树：



不断递归调用，直到所有叶子都是相同属性的。

所有样本已经被分类，算法停止，我们得到最终的决策树如下：



最近邻分类器

朴素贝页斯分类器

- **x**是一个类标识未知的数据样本。
- $P(c_i)$: 类c_i的先验概率(prior probability) (即在我们观测任何数据之前的初始概率, 它反映了背景知识)
- $P(\mathbf{x})$: 样本数据被观测的概率。
- $P(\mathbf{x}|c_i)$: 在属于类c_i的前提下，观测到样本**x**的概率，又称为似然。
- 确定后验概率$P(c_i|\mathbf{x})$: 给定观测数据样本**x**，确定**x**属于类c_i的概率。

- 给定训练集, **x**属于类c_i的后验概率(posteriori probability), $P(c|\mathbf{x})$ 遵守如下贝叶斯定理

$$P(c_i|\mathbf{x})=\frac{P(\mathbf{x}|c_i)P(c_i)}{P(\mathbf{x})}$$

- MAP (Maximum a posteriori，极大后验) 假设

$$\underset{c_i\in C}{\operatorname{argmax}}P(c_i|\mathbf{x})=\underset{c_i\in C}{\operatorname{argmax}}P(\mathbf{x}|c_i)P(c_i).$$

- 实际应用的困难: 需要许多概率的初始知识, 需要较多的计算时间。

- 一个简化假设: 属性之间是条件独立的:

$$P(\mathbf{x}\,|\,c_i)=\prod_{k=1}^{n}P(x_k|\,c_i)$$

- 一旦知道了概率$P(\mathbf{x}|c_i)$, 把X赋予使得 $P(\mathbf{x}|c_i)*P(c_i)$具有极大值的类c_i。

一个例子

一个例子

类:
c_1:buys_computer= 'yes'
c_2:buys_computer= 'no'

数据样本
x =(age<=30, Income=medium, Student=yes Credit_rating= Fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

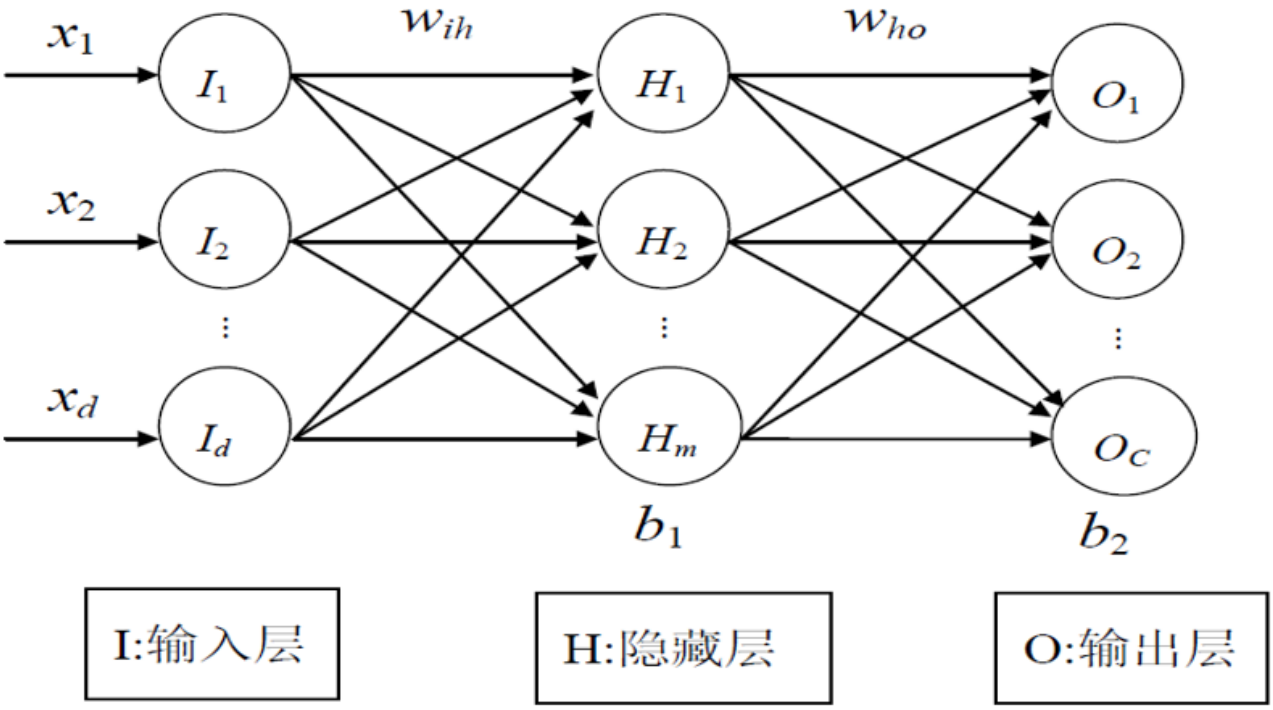
一个例子

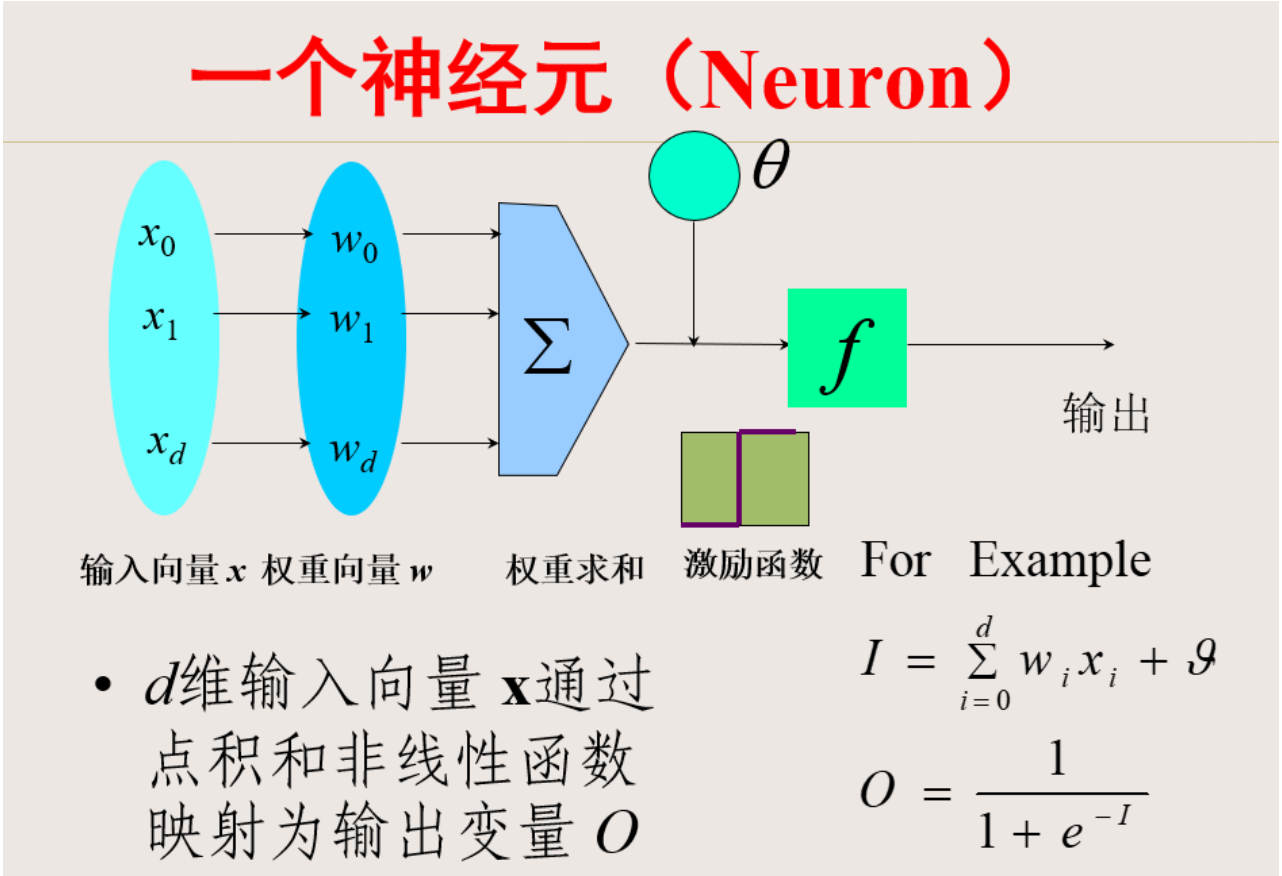
- 对每一个类，计算$P(\mathbf{x}|c_i)$
  P(age="<30" | buys_computer="yes") = 2/9=0.222
  P(age="<30" | buys_computer="no") = 3/5 =0.6
  P(income="medium" | buys_computer="yes")= 4/9 =0.444
  P(income="medium" | buys_computer="no") = 2/5 = 0.4
  P(student="yes" | buys_computer="yes)= 6/9 =0.667
  P(student="yes" | buys_computer="no")= 1/5=0.2
  P(credit_rating="fair" | buys_computer="yes")=6/9=0.667
  P(credit_rating="fair" | buys_computer="no")=2/5=0.4
- **x = (age<=30 ,income =medium, student=yes,credit_rating=fair)**
  $P(\mathbf{x}|c_1)$: P(**x**|buys_computer="yes")= 0.222 × 0.444 × 0.667 × 0.0.667 =0.044
  P(**x**|buys_computer="no")= 0.6 × 0.4 × 0.2 × 0.4 =0.019
  $P(\mathbf{x}|c_2) \times P(c_2)$: P(**x** | yes) × P(yes) = 0.028
  P(**x** | no) × P(no) = 0.007
- 所以：**x** 属于类 **"buys_computer=yes"**

深度学习

三层前馈神经网络

- 大致结构：





- 结构设计：

1. 神经网络的结构是指它具有多少单元以及它们如何连接：多少层？每层多少节点单元？单元之间如何连接？

2. 大多数神经网络被组织成称为层的单元组。大多数神经网络结构将这些层布置成**链式结构**，其中每一层都是前一层的函数。在这种结构中，第一层由下式给出：

$$\boldsymbol{h}^{(1)} = g^{(1)}\left(\boldsymbol{W}^{(1)\top}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right),$$

第二层由

$$\boldsymbol{h}^{(2)} = g^{(2)}\left(\boldsymbol{W}^{(2)\top}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}\right)$$

给出，以此类推

3. 具有一层隐藏层的前馈神经网络足以表示任何函数，但是网络层可能非常大（不可实现）并且可能无法正确地学习和泛化。
   但在很多情况下，使用更深的模型能够减少表示期望函数所需的单元的数量，并且可以减少泛化误差。

- 网络结构：

1. 输入层节点数：对应样本的维数：有多少个属性，就设置多少个输入节点
2. 隐藏层节点数：隐藏层节点数的设置没有很具体的建议（视为超参数），要通过实验验证设置多少为好。
3. 输出层节点数：若为2元分类, 可设为1个, 若输出范围为[0,1], 则可采用softmax函数, 若大于0.5，则类标签为1, 否则为0。也可设为2个节点；若是多元分类(标签数C > 2), 则用C个输出节点(对应着C维输出向量), 某个样本标签为i, 则训练时对应的目标向量T为(0,0,...,1,0,...,0), 其中第i个为1,其它为0.

- 深度的影响：
  更深的网络能够更好地泛化。

- 参数数量的影响

激活函数

- 为什么要使用非线性激活函数：
  线性模型，例如逻辑回归和线性回归，是非常吸引人的，因为无论是封闭形式还是使用凸优化，它们都能高效而可靠地拟合。
  线性模型也有明显的缺陷，那就是该模型的能力被局限在线性函数里，所以它无法理解任何两个输入变量间的相互作用。
  即使多层的线性函数组合在一起，本质上也只有线性函数的表达能力。在激活函数是线性的情况下，相比于单隐藏层神经网络，包含多隐藏层的深度网络并没有增加表达能力。

- 各种激活函数：

sigmoid：

$$\sigma(x) = \frac{1}{1+\exp(-x)}.$$

softmax:

$$\operatorname{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n}\exp(x_j)}.$$

对sigmoid的扩展，常用于预测与多项式分布

ReLU:

$$g(z) = \max\{0, z\}$$

tanh:

$$\tanh(z) = 2\sigma(2z) - 1.$$

后向传播算法（BP）

P、NP、NP完全与NP难

1. P类问题：存在多项式时间算法的问题。
2. NP类问题：能在多项式时间内验证得出一个正确解的问题。（P类问题是NP问题的子集，因为存在多项式时间解法的问题，总能在多项式时间内验证他。）
3. NPC问题:是一个NP问题，并且如果所有NP问题都能在多项式时间内转化为他，则称该NP问题为NPC问题。（是一类问题）
4. NPH问题：NP-Hard问题是这样一种问题，它满足NPC问题定义的第二条（所有NP问题都能在多项式时间内转化为他）但不一定要满足第一条（是一个NP问题），就是说，NP-Hard问题要比 NPC问题的范围广。即使NPC问题发现了多项式级的算法，NP-Hard问题有可能仍然无法得到多项式级的算法。事实上，由于NP-Hard放宽了限定条件，它将有可能比所有的NPC问题的时间复杂度更高从而更难以解决。

基于梯度的优化算法

1. 批量梯度下降（Batch GD，BGD）：更新参数时使用所有的样本。训练速度慢，不能以在线的形式更新我们的模型。

2. 随机梯度下降(Stochastic GD, SGD)：仅仅选取一个样本i来进行梯度更新。能够在线学习，速度快。但是更新值的方差很大，在频繁的更新之下，它的目标函数有着剧烈波动。

3. 小批量梯度下降（Mini Batch GD，MBGD）：我们采用m (1< m << n)个样本来迭代。收敛过程更为稳定

后向传播算法

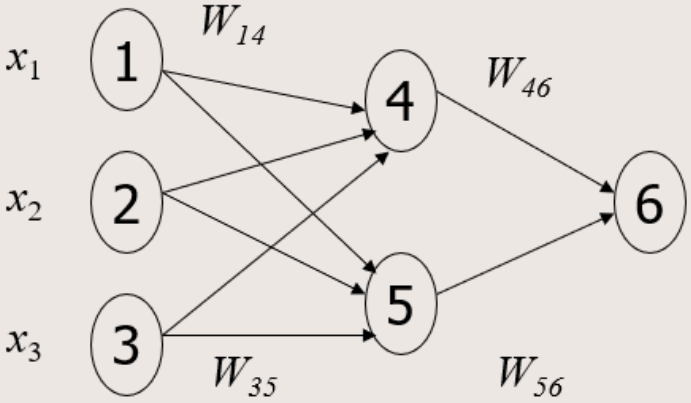
后向传播算法

- **训练目标**
  - 获得一组权重，使得能够对所有的训练样例进行正确分类。
- **步骤**
  - 用随机数初始化所有权重和偏置。
  - 把每个训练样例逐个给网络输入层
  - 对于每个网络单元
    - 利用所有输入的线性组合，计算其获得的净输入（对于输入单元，它的输出等于它的输入值）。
    - 利用激励函数和净输入计算其输出。
    - 计算错误值
    - 更新权重和偏置。

实例：

训练示例

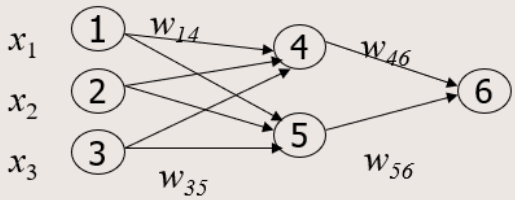
- 给定如下二元分类网络，其中输入向量：$\mathbf{x}=(x_1, x_2, x_3)=(1, 0, 1)$，其类标签为1。



初始输入、权重与偏置

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

- 网络净输入与输出计算
- Unit j Net input, I_j Output O_j
- 4 0.2*1+0.4*0-0.5*1-0.4=-0.7 $1/(1+e^{0.7})=0.332$
- 5 -0.3*1+0.1*0+0.2*1+0.2=0.1 $1/(1+e^{-0.1})=0.525$
- 6 (-0.3)(0.332)+(-0.2)(0.525)+0.1=-0.105 $1/(1+e^{0.105})=0.474$
- 计算误差
- Unit j Err_j
- 6 (0.474)(1-0.474)(1-0.474)=0.1311
- 5 (0.525)(1-0.525)(0.1311)(-0.2)=-0.0065
- 4 (0.332)(1-0.332)(0.1311)(-0.3)=-0.0087



$$Err_j = O_j(1-O_j)(T_j-O_j)$$

$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$$

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

- 计算权重与偏置更新
- 权重或偏置 新值

$$\theta_j = \theta_j + (l)Err_j$$

$$w_{ij} = w_{ij} + (l)Err_j O_i$$

- w_{46} -0.3+(0.9)(0.1311)(0.332)=-0.261
- w_{56} -0.2+(0.9)(0.1311)(0.525)=-0.138
- w_{14} 0.2+(0.9)(-0.0087)(1)=0.192
- w_{15} -0.3+(0.9)(-0.0065)(1)=-0.306
- w_{24} 0.4+(0.9)(-0.0087)(0)=0.4
- w_{25} 0.1+(0.9)(-0.0065)(0)=0.1
- w_{34} -0.5+(0.9)(-0.0087)(1)=-0.508
- w_{35} 0.2+(0.9)(-0.0065)(1)=0.194
- θ_6 0.1+(0.9)(0.1311)=0.218
- θ_5 0.2+(0.9)(-0.0065)=0.194
- θ_4 -0.4+(0.9)(-0.0087)=-0.408

O_1	O_2	O_3
1	0	1
O_4	O_5	O_6
0.332	0.525	0.474
Err_6	Err_5	Err_4
0.1311	-0.0065	-0.0087

深度网络及其训练

深度网络的优势：

1. 高的表达力(Capacity) - 使用深度网络最主要的优势在于，它能以更加紧凑简洁的方式来表达比浅层网络大得多的函数集合。
2. 深度学习= 学习（特征的）层次化表示。低层特征 → 中层特征 → 高层特征
3. 深度网络通常能够获取到输入的“层次型分组”或者“部分-整体分解”结构：第一层会学习得到原始输入的一阶特征，第二层会学习得到二阶特征，该特征对应一阶特征里包含的一些模式。更高层还会学到更高阶的特征。

例如：
对于图像数据，我们能够使用深度网络学习到“部分-整体”的分解关系。
第一层可以学习如何将图像中的像素组合在一起来检测边缘。
第二层可以将边缘组合起来检测更长的轮廓或者简单的“目标的部件”。
在更深的层次上，可以将这些轮廓进一步组合起来以检测更为复杂的特征。

深度网络训练面临的挑战

1. 数据获取问题：标记数据少,无标记数据多。
2. 局部极值问题：容易陷入局部最优。
3. 梯度消失问题：除了最高的几层之外，梯度很小或消失; 如果不进行pretraining，效果有时还不如浅层网络。
4. 复杂性太高：如果对所有层同时训练，时间复杂度会太高。训练时间和资源花费非常大。

逐层贪婪预训练

从底层(输入层)开始，一层一层的往顶层(输出层)训练，原始输入作为训练第一个隐藏层的输入；训练好一层后，参数固定，再在其上堆积（Stacking，又称栈化）一层，然后用前一层的输出作为训练该层的输入。
这些各层单独训练所得到的权重被用来初始化最终（或者说全部）的深度网络的权重，然后使用后向传播算法对整个网络进行“微调”（即把所有层放在一起来优化有标签训练集上的训练误差）。

上述训练可以是无监督的或有监督的

- 无监督：如使用受限玻尔兹曼机(Restricted Boltzmann Machines, RBM)，自编码器(auto-encoder)等；
- 有监督：直接用训练集的标识作为训练一层时的输出（将每一步的分类误差作为目标函数）。

无监督：自编码器

自编码神经网络是一种无监督学习算法，它使用了反向传播算法，并让目标值等于输入值。自编码神经网络尝试学习一个f(x)□x的函数。换句话说，它尝试逼近一个恒等函数，从而使得输出接近于输入。

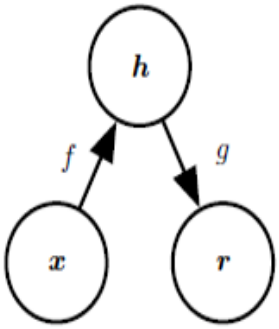


图 14.1: 自编码器的一般结构，通过内部表示或编码 $\boldsymbol{h}$ 将输入 $\boldsymbol{x}$ 映射到输出（称为重构）$\boldsymbol{r}$。自编码器具有两个组件：编码器 f（将 $\boldsymbol{x}$ 映射到 $\boldsymbol{h}$）和解码器 g（将 $\boldsymbol{h}$ 映射到 $\boldsymbol{r}$）。

- 分类：
  欠完备(under complete)自编码器：隐藏编码维数小于输入维数。

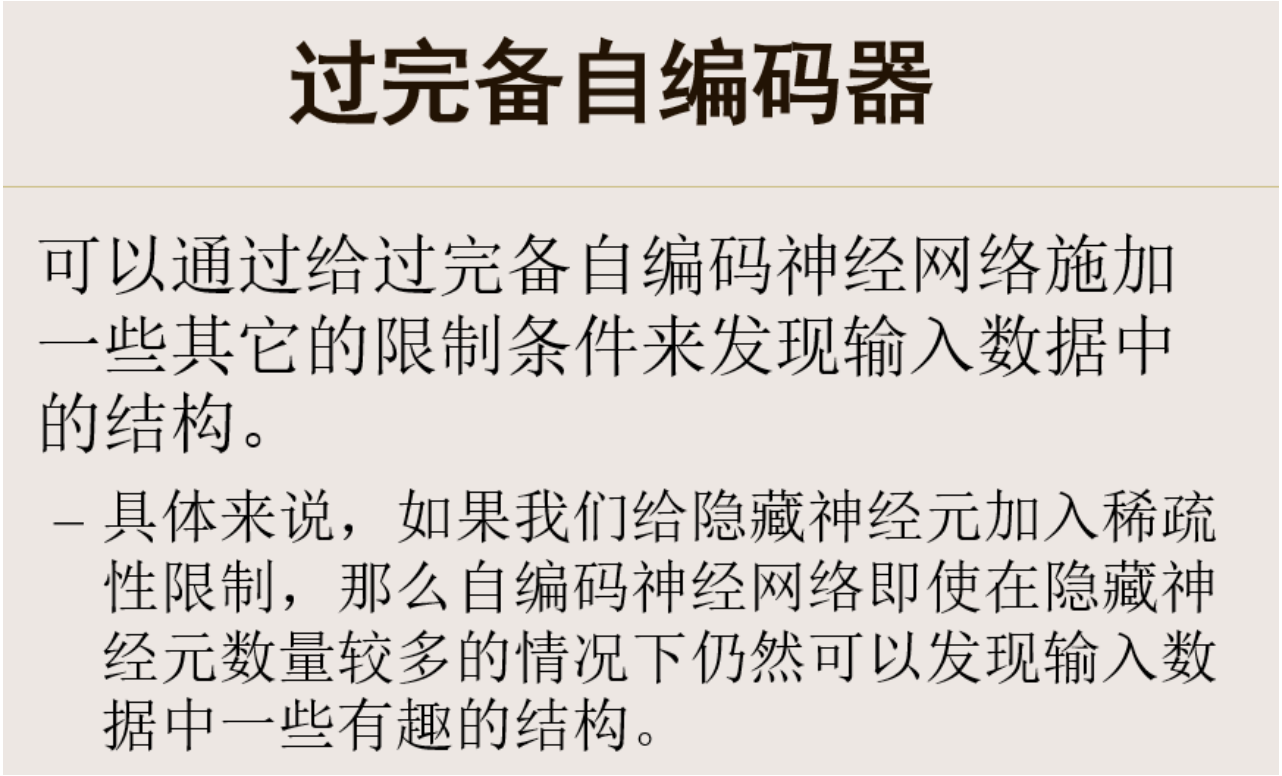
欠完备自编码器

欠完备自编码器可以学习数据分布最显著的特征

– 假设某个自编码神经网络的输入**x**是一张10×10图像（共100个像素）的像素灰度值，于是n=100，其隐藏层中有50个隐藏神经元。注意，输出也是100维的$y \in R^{100}$。由于只有50个隐藏神经元，我们迫使自编码神经网络去学习输入数据的压缩表示，也就是说，它必须从50维的隐藏神经元激活度向量$y \in R^{50}$中重构出100维的像素灰度值输入x。

过完备(over-complete)自编码器：如果隐藏编码的维数允许与输入相等，或隐藏编码维数大于输入

的维数。



深度网络的正则化技术

以增大训练误差为代价来减少测试误差。这些策略统称为正则化。

常见的正则化策略

1. 添加额外的约束，如参数的约束(硬约束);
2. 有些向目标函数增加额外项，对应于参数值的软约束;
3. 提前终止策略;
4. Dropout

提前终止

当训练有足够的表示能力甚至会过度拟合的大模型时，我们经常观察到，训练误差会随着时间的推移逐渐降低但验证集的误差会再次上升。

在每次验证集误差有所改善后，我们存储模型参数的副本。当训练算法终止时，我们返回这些参数而不是最新的参数。当验证集上的误差在事先指定的循环内没有进一步改善时，算法就会终止。这种策略被称为提前终止(early stopping)。这可能是深度学习中最常用的正则化形式。

Dropout

dropout是指在深度学习网络的训练过程中，对于神经网络单元，按照一定的概率将其暂时从网络中丢弃。注意是暂时，对于随机梯度下降来说，由于是随机丢弃，故而每一个mini-batch都在训练不同的网络。

没有免费的午餐定理（NFL定理）

1）**对所有可能的的目标函数求平均**，所有学习算法的“非训练集误差”的期望值相同;
2）对任意固定的训练集，对所有的目标函数求平均，所有学习算法的“非训练集误差”的期望值也相同;
3）**对所有的先验知识求平均**，所有学习算法的“非训练集误差”的期望值也相同;
4）对任意固定的训练集，对所有的先验知识求平均，所有学习算法的的“非训练集误差”的期望值也相同;

NFL定理表明没有一个学习算法可以在任何数据集上总是学习到最准确的分类器。