Datawhale 异步社区

《机器学习公式详解》（南瓜书）

第5章 神经网络

本节主讲：谢文睿

本节大纲

西瓜书对应章节：5.1、5.2、5.3

1. M-P神经元
2. 感知机
3. 神经网络

M-P神经元

M-P神经元（一个用来模拟生物行为的数学模型）：接收n个输入(通常是来自其他神经元)，并给各个输入赋予权重计算加权和，然后和自身特有的阈值θ进行比较（作减法），最后经过激活函数（模拟“抑制”和“激活”）处理得到输出（通常是给下一个神经元）

$$y = f\left(\sum_{i=1}^{n} w_i x_i - \theta\right) = f\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b\right)$$

单个M-P神经元：感知机（sgn作激活函数）、对数几率回归（sigmoid作激活函数）

多个M-P神经元：神经网络

感知机

感知机模型：激活函数为sgn（阶跃函数）的神经元

$$y=\operatorname{sgn}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta)=\begin{cases}1, & \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta \geqslant 0\\ 0, & \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta < 0\end{cases}$$

其中，$\boldsymbol{x}\in\mathbb{R}^n$为样本的特征向量，是感知机模型的输入，$\boldsymbol{w},\theta$是感知机模型的参数，$\boldsymbol{w}\in\mathbb{R}^n$为权重，$\theta$为阈值。

感知机

再从几何角度来说，给定一个线性可分的数据集T，感知机的学习目标是求得能对数据集T中的正负样本完全正确划分的超平面，其中$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}-\theta$即为超平面方程。

n维空间的超平面（$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0$，其中$\boldsymbol{w},\boldsymbol{x}\in\mathbb{R}^n$）：

- 超平面方程不唯一
- 法向量$\boldsymbol{w}$垂直于超平面
- 法向量$\boldsymbol{w}$和位移项b确定一个唯一超平面
- 法向量$\boldsymbol{w}$指向的那一半空间为正空间，另一半为负空间

感知机

感知机学习策略：随机初始化$\boldsymbol{w}, b$，将全体训练样本代入模型找出误分类样本，假设此时误分类样本集合为$M \subseteq T$，对任意一个误分类样本$(\boldsymbol{x}, y) \in M$来说，当$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} - \theta \geqslant 0$时，模型输出值为$\hat{y} = 1$，样本真实标记为$y = 0$；反之，当$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} - \theta < 0$时，模型输出值为$\hat{y} = 0$，样本真实标记为$y = 1$。综合两种情形可知，以下公式恒成立

$$(\hat{y} - y)(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} - \theta) \geq 0$$

所以，给定数据集T，其损失函数可以定义为：

$$L(\boldsymbol{w}, \theta) = \sum_{\boldsymbol{x} \in M} (\hat{y} - y)(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} - \theta)$$

显然，此损失函数是非负的。如果没有误分类点，损失函数值是0。而且，误分类点越少，误分类点离超平面越近，损失函数值就越小。

感知机

具体地，给定数据集

$$T=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\dots,(\boldsymbol{x}_N,y_N)\}$$

其中$\boldsymbol{x}_i\in\mathbb{R}^n, y_i\in\{0,1\}$，求参数$\boldsymbol{w},\theta$，使其为极小化损失函数的解：

$$\min_{\boldsymbol{w},\theta} L(\boldsymbol{w},\theta)=\min_{\boldsymbol{w},\theta}\sum_{\boldsymbol{x}_i\in M}(\hat{y}_i-y_i)(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i-\theta)$$

其中$M\subseteq T$为误分类样本集合。若将阈值θ看作一个固定输入为-1的“哑节点”，即

$$-\theta=-1\cdot w_{n+1}=x_{n+1}\cdot w_{n+1}$$

根据该式，可将要求解的极小化问题进一步简化为

$$\min_{\boldsymbol{w}} L(\boldsymbol{w})=\min_{\boldsymbol{w}}\sum_{\boldsymbol{x}_i\in M}(\hat{y}_i-y_i)\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i$$

感知机

感知机学习算法：当误分类样本集合M固定时，那么可以求得损失函数$L(\boldsymbol{w})$的梯度为

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w})=\sum_{\boldsymbol{x}_{\boldsymbol{i}} \in M}\left(\hat{y}_i-y_i\right) \boldsymbol{x}_{\boldsymbol{i}}$$

感知机的学习算法具体采用的是随机梯度下降法，也就是极小化过程中不是一次使M中所有误分类点的梯度下降，而是一次随机选取一个误分类点使其梯度下降。所以权重$\boldsymbol{w}$的更新公式为

$$\boldsymbol{w} \leftarrow \boldsymbol{w}+\Delta \boldsymbol{w}$$

$$\Delta \boldsymbol{w}=-\eta\left(\hat{y}_i-y_i\right) \boldsymbol{x}_i=\eta\left(y_i-\hat{y}_i\right) \boldsymbol{x}_i$$

相应地，$\boldsymbol{w}$中的某个分量w_i的更新公式即为西瓜书公式(5.2)，最终解出来的$\boldsymbol{w}$通常不唯一。

神经网络

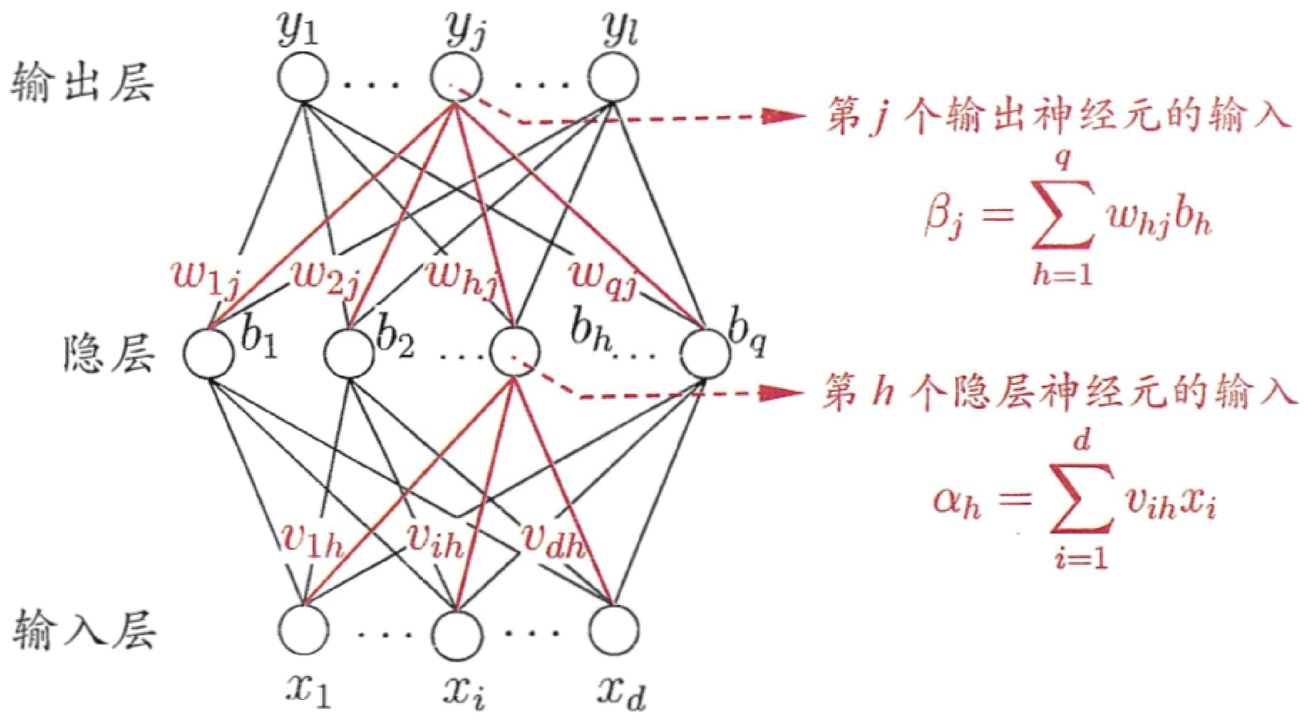
由于像感知机这种单个神经元分类能力有限，只能分类线性可分的数据集，对于线性不可分的数据集则无能为力，但是多个神经元构成的神经网络能够分类线性不可分的数据集（西瓜书上异或问题的那个例子），且有理论证明（通用近似定理）：只需一个包含足够多神经元的隐层，多层前馈网络（最经典的神经网络之一）就能以任意精度逼近任意复杂度的连续函数。因此，神经网络既能做回归，也能做分类，而且不需要复杂的特征工程。

BUT，理想很丰满，现实很骨感，神经网络存在如下问题待屏幕前的你来解决：

- 面对一个具体场景，神经网络该做多深？多宽？
- 面对一个具体场景，神经网络的结构该如何设计才最合理？
- 面对一个具体场景，神经网络的输出结果该如何解释？

神经网络

多层前馈网络：每层神经元与下一层神经元全互连，神经元之间不存在同层连接，也不存在跨层连接。（隐层阈值γ_h，输出层阈值θ_j）



神经网络

将神经网络（记为NN）看作一个特征加工函数

$$\boldsymbol{x} \in \mathbb{R}^d \to \mathrm{NN}(\boldsymbol{x}) \to \boldsymbol{y} = \boldsymbol{x}^* \in \mathbb{R}^l$$

（单输出）回归：后面接一个$\mathbb{R}^l \to \mathbb{R}$的神经元，例如：没有激活函数的神经元

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

分类：后面接一个$\mathbb{R}^l \to [0,1]$的神经元，例如：激活函数为sigmoid函数的神经元

$$y = \frac{1}{1 + e^{-(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}^* + b)}}$$

在模型训练过程中，神经网络（NN）自动学习提取有用的特征，因此，机器学习向“全自动数据分析”又前进了一步。

神经网络

假设多层前馈网络中的激活函数全为sigmoid函数，且当前要完成的任务为一个（多输出）回归任务，因此损失函数可以采用均方误差（分类任务则用交叉熵）。对于某个训练样本$(\boldsymbol{x}_k, \boldsymbol{y}_k)$，其中$\boldsymbol{y}_k = \left(y_1^k, y_2^k, \ldots, y_l^k\right)$，假定其多层前馈网络的输出为$\hat{\boldsymbol{y}}_k = \left(\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_l^k\right)$，则该单个样本的均方误差（损失）为

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

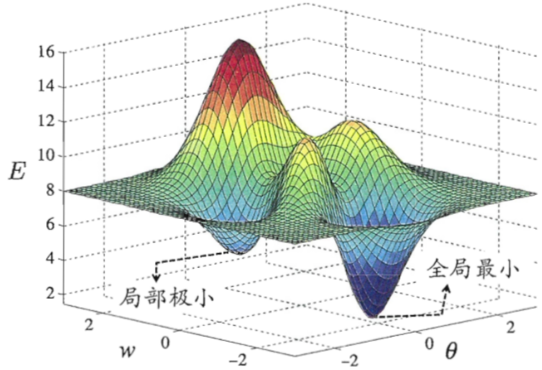
神经网络

误差逆传播算法（BP算法）： 基于随机梯度下降的参数更新算法

$$w \leftarrow w + \Delta w$$

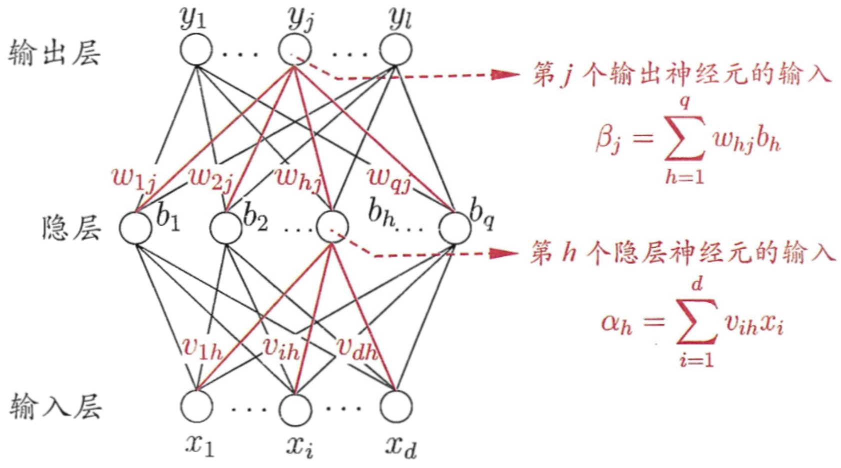
$$\Delta w = -\eta \nabla_w E$$

其中只需推导出$\nabla_w E$这个损失函数E关于参数w的一阶偏导数（梯度）即可（链式求导）。值得一提的是，由于$\mathrm{NN}(\boldsymbol{x})$通常是极其复杂的非凸函数，不具备像凸函数这种良好的数学性质，因此随机梯度下降不能保证一定能走到全局最小值点，更多情况下走到的都是局部极小值点。



神经网络

下面以输入层第i个神经元与隐层第h个神经元之间的连接权v_{ih}为例推导一下:



$$E_k = \frac{1}{2}\sum_{j=1}^{l}\left(\hat{y}_j^k - y_j^k\right)^2, \quad \Delta v_{ih} = -\eta\frac{\partial E_k}{\partial v_{ih}}$$

$$\frac{\partial E_k}{\partial v_{ih}} = \sum_{j=1}^{l}\frac{\partial E_k}{\partial \hat{y}_j^k}\cdot\frac{\partial \hat{y}_j^k}{\partial \beta_j}\cdot\frac{\partial \beta_j}{\partial b_h}\cdot\frac{\partial b_h}{\partial \alpha_h}\cdot\frac{\partial \alpha_h}{\partial v_{ih}}$$

神经网络

$$
\begin{aligned}
g_j &= -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \\
&= -\left(\hat{y}_j^k - y_j^k\right) f'\left(\beta_j - \theta_j\right) \\
&= \hat{y}_j^k \left(1 - \hat{y}_j^k\right) \left(y_j^k - \hat{y}_j^k\right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E_k}{\partial v_{ih}} &= \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \cdot \frac{\partial \alpha_h}{\partial v_{ih}} \\
&= \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \cdot x_i \\
&= \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot f'(\alpha_h - \gamma_h) \cdot x_i \\
&= \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot w_{hj} \cdot f'(\alpha_h - \gamma_h) \cdot x_i \\
&= \sum_{j=1}^{l} (-g_j) \cdot w_{hj} \cdot f'(\alpha_h - \gamma_h) \cdot x_i \\
&= -f'(\alpha_h - \gamma_h) \cdot \sum_{j=1}^{l} g_j \cdot w_{hj} \cdot x_i \\
&= -b_h(1 - b_h) \cdot \sum_{j=1}^{l} g_j \cdot w_{hj} \cdot x_i \\
&= -e_h \cdot x_i
\end{aligned}
$$

神经网络

其他参数的更新公式推导参见《机器学习公式详解》（南瓜书）第5章相应部分~

预告

下一节：支持向量机

西瓜书对应章节：6.1、6.2

结束语

欢迎加入【南瓜书读者交流群】，我们将在群里进行答疑、勘误、本次直播回放、本次直播PPT发放、下次直播通知等最新资源发放和活动通知。

加入步骤：

1. 关注公众号【Datawhale】，发送【南瓜书】三个字获取机器人“小豚”的微信二维码
2. 添加“小豚”为微信好友，然后对“小豚”发送【南瓜书】三个字即可自动邀请进群、