深度之眼
deepshare.net

法律声明

课程详情请咨询

- 微信公众号：深度之眼
- 客服微信号：deepshare0920

公众号

微 信

deepshare.net
深度之眼

西瓜书公式推导

导师：Sm1les（Datawhale南瓜书项目负责人）

deepshare.net
深度之眼

神经网络公式推导

Derivation of Neural Networks

本节大纲

Outline

deepshare.net
深度之眼

先修内容：西瓜书5.1、5.2、5.3

1.感知机

2.神经网络

感知机

Perceptron

1.定义

2.感知机的几何解释

3.学习策略

4.算法

感知机

Perceptron

定义：

假设输入空间是 $\mathcal{X} \subseteq R^n$ 输出空间是 $\mathcal{Y} = \{1, 0\}$ 。输入 $\boldsymbol{x} \in \mathcal{X}$ 表示实例的特征向量，对应于输入空间的点；输出 $y \in \mathcal{Y}$ 表示实例的类别。由输入空间到输出空间的如下函数

$$f(\boldsymbol{x}) = \operatorname{sgn}(\boldsymbol{w}^T \boldsymbol{x} + b)$$

称为感知机。其中 $\boldsymbol{w}$ 和b为感知机模型参数，sgn是阶跃函数，即

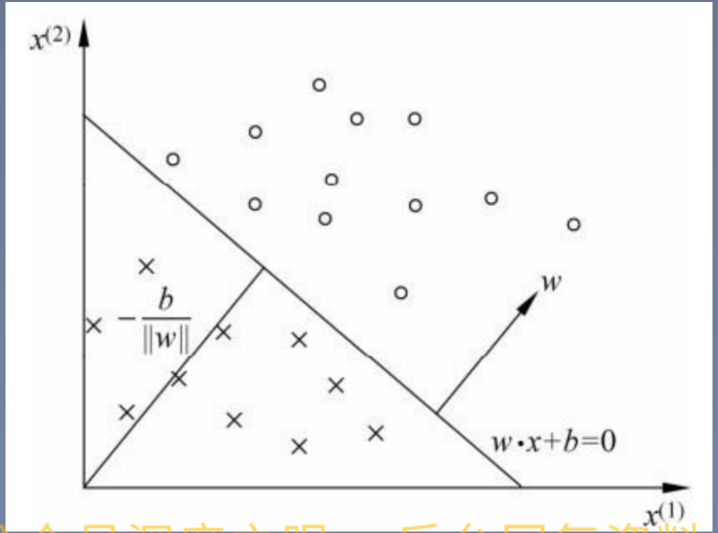
$$\operatorname{sgn}(z) = \begin{cases} 1, & z \geqslant 0 \\ 0, & z < 0 \end{cases}$$

感知机

Perceptron

感知机的几何解释:

线性方程 $\boldsymbol{w}^T\boldsymbol{x}+b=0$ 对应于特征空间（输入空间）R^n 中的一个超平面S，其中 $\boldsymbol{w}$ 是超平面的法向量，b是超平面的截距。这个超平面将特征空间划分为两个部分。位于两边的点（特征向量）分别被分为正、负两类。因此，超平面S称为分离超平面，如图所示



感知机

Perceptron

学习策略：

假设训练数据集是线性可分的，感知机学习的目标是求得一个能够将训练集正实例点和负实例点完全正确分开的超平面。为了找出这样的超平面S，即确定感知机模型参数 w 和b，需要确定一个学习策略，即定义损失函数并将损失函数极小化。损失函数的一个自然选择是误分类点的总数。但是，这样的损失函数不是参数 w 和b的连续可导函数，不易优化，所以感知机采用的损失函数为误分类点到超平面的总距离。

感知机

Perceptron

输入空间 R^n 中点 $\boldsymbol{x}_0$ 到超平面S的距离公式为

$$\frac{\left|\boldsymbol{w}^T \boldsymbol{x}_0 + b\right|}{\|\boldsymbol{w}\|}$$

其中，$\|\boldsymbol{w}\|$ 表示向量 $\boldsymbol{w}$ 的 L_2 范数，也即模长。若将b看成哑结点，也即合并进w可得

$$\frac{\left|\hat{\boldsymbol{w}}^T \hat{\boldsymbol{x}}_0\right|}{\|\hat{\boldsymbol{w}}\|}$$

感知机

Perceptron

deepshare.net
深度之眼

设误分类点集合为M，那么所有误分类点到超平面S的总距离为

$$\sum_{\hat{\boldsymbol{x}}_i \in M} \frac{\left|\hat{\boldsymbol{w}}^T \hat{\boldsymbol{x}}_i\right|}{\|\hat{\boldsymbol{w}}\|}$$

又因为，对于任意误分类点 $\hat{\boldsymbol{x}}_i \in M$ 来说都有

$$(\hat{y}_i - y_i)\hat{\boldsymbol{w}}^T \hat{\boldsymbol{x}}_i > 0$$

其中，$\hat{y}_i$ 为当前感知机的输出。于是所有误分类点到超平面S的总距离可改写为

$$\sum_{\hat{\boldsymbol{x}}_i \in M} \frac{(\hat{y}_i - y_i)\hat{\boldsymbol{w}}^T \hat{\boldsymbol{x}}_i}{\|\hat{\boldsymbol{w}}\|}$$

感知机

Perceptron

不考虑 $\frac{1}{\|\hat{\boldsymbol{w}}\|}$ 就得到感知机学习的损失函数

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

显然，损失函数 $L(\hat{\boldsymbol{w}})$ 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，在误分类时是参数 $\hat{\boldsymbol{w}}$ 的线性函数，在正确分类时是0。因此，给定训练数据集，损失函数 $L(\hat{\boldsymbol{w}})$ 是$\hat{\boldsymbol{w}}$的连续可导函数。

感知机

Perceptron

deepshare.net 深度之眼

算法：

感知机学习算法是对以下最优化问题的算法，给定训练数据集

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

其中 $\hat{\boldsymbol{x}}_i \in R^{n+1}, y_i, \in \{1,0\}$ 求参数 $\hat{\boldsymbol{w}}$ 使其为以下损失函数极小化问题的解

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

其中M为误分类点的集合。

感知机

Perceptron

感知机学习算法是误分类驱动的，具体采用随机梯度下降法。首先，任意选取一个超平面 $\hat{\boldsymbol{w}}_0^T\hat{\boldsymbol{x}}=0$ 用梯度下降法不断地极小化损失函数 $L(\hat{\boldsymbol{w}})$ 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。已知损失函数的梯度为

$$\begin{aligned}\nabla L(\hat{\boldsymbol{w}})=\frac{\partial L(\hat{\boldsymbol{w}})}{\partial \hat{\boldsymbol{w}}}&=\frac{\partial}{\partial \hat{\boldsymbol{w}}}\left[\sum_{\hat{\boldsymbol{x}}_i\in M}(\hat{y}_i-y_i)\hat{\boldsymbol{w}}^T\hat{\boldsymbol{x}}_i\right]\\&=\sum_{\hat{\boldsymbol{x}}_i\in M}\left[(\hat{y}_i-y_i)\frac{\partial}{\partial \hat{\boldsymbol{w}}}(\hat{\boldsymbol{w}}^T\hat{\boldsymbol{x}}_i)\right]\end{aligned}$$

由矩阵微分公式 $\frac{\partial \boldsymbol{x}^T\boldsymbol{a}}{\partial \boldsymbol{x}}=\boldsymbol{a}$ 可得

$$\nabla L(\hat{\boldsymbol{w}})=\frac{\partial L(\hat{\boldsymbol{w}})}{\partial \hat{\boldsymbol{w}}}=\sum_{\hat{\boldsymbol{x}}_i\in M}(\hat{y}_i-y_i)\hat{\boldsymbol{x}}_i$$

感知机

Perceptron

deepshare.net
深度之眼

那么随机选取一个误分类点 $\hat{\boldsymbol{x}}_i$ 进行梯度下降可得参数 $\hat{\boldsymbol{w}}$ 的更新公式为

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

$$\Delta\hat{\boldsymbol{w}} = -\eta\nabla L(\hat{\boldsymbol{w}})$$

$$\hat{\boldsymbol{w}} \leftarrow \hat{\boldsymbol{w}} - \eta\nabla L(\hat{\boldsymbol{w}})$$

$$\hat{\boldsymbol{w}} \leftarrow \hat{\boldsymbol{w}} - \eta(\hat{y}_i - y_i)\hat{\boldsymbol{x}}_i = \hat{\boldsymbol{w}} + \eta(y_i - \hat{y}_i)\hat{\boldsymbol{x}}_i$$

$$\Delta\hat{\boldsymbol{w}} = \eta(y_i - \hat{y}_i)\hat{\boldsymbol{x}}_i$$ 此即为式5.2

其中 $\eta \in (0,1)$ 称为学习率。

神经网络

Neural Networks

deepshare.net
深度之眼

1.模型结构

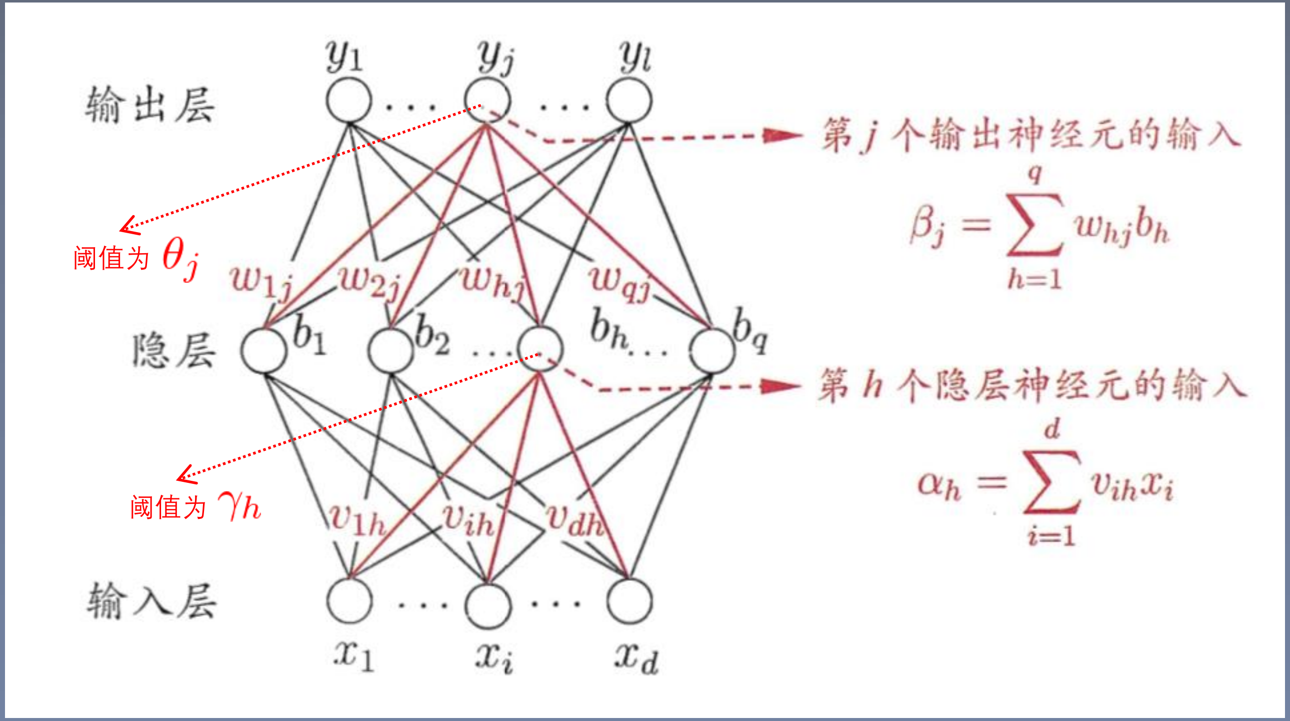
2.标准BP算法

神经网络

Neural Networks

deepshare.net 深度之眼

单隐层前馈网络模型结构图如下：



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

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

神经网络

Neural Networks

标准BP算法：

给定一个训练样本 $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ 假设模型输出为 $\hat{\boldsymbol{y}}_k = \left(\hat{y}_1^k, \hat{y}_2^k, \ldots, \hat{y}_l^k\right)$ 则均方误差为

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

书上式5.4

如果按照梯度下降法更新模型的参数，那么各个参数的更新公式为

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

$$\theta_j \leftarrow \theta_j + \Delta \theta_j = \theta_j - \eta \frac{\partial E_k}{\partial \theta_j}$$

$$v_{ih} \leftarrow v_{ih} + \Delta v_{ih} = v_{ih} - \eta \frac{\partial E_k}{\partial v_{ih}}$$

$$\gamma_h \leftarrow \gamma_h + \Delta \gamma_h = \gamma_h - \eta \frac{\partial E_k}{\partial \gamma_h}$$

神经网络

Neural Networks

已知 E_k 和 w_{hj} 的函数链式关系为

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

$$\downarrow$$

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

f 为Sigmoid函数

$$\downarrow$$

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

所以

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

神经网络

Neural Networks

deepshare.net
深度之眼

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

$$= \frac{1}{2} \times 2 \times (\hat{y}_j^k - y_j^k) \times 1$$

$$= \hat{y}_j^k - y_j^k$$

$$\frac{\partial \hat{y}_j^k}{\partial \beta_j} = \frac{\partial [f(\beta_j - \theta_j)]}{\partial \beta_j}$$

$$= f'(\beta_j - \theta_j) \times 1$$

由于 $f'(x) = f(x)(1 - f(x))$

$$= f\left(\beta_j - \theta_j\right) \times \left[1 - f\left(\beta_j - \theta_j\right)\right]$$

$$= \hat{y}_j^k\left(1 - \hat{y}_j^k\right)$$

$$\frac{\partial \beta_j}{\partial w_{hj}} = \frac{\partial(\sum_{h=1}^{q} w_{hj} b_h)}{\partial w_{hj}}$$

$$= b_h$$

令 $g_j = -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} = -(\hat{y}_j^k - y_j^k) \cdot \hat{y}_j^k\left(1 - \hat{y}_j^k\right) = \hat{y}_j^k\left(1 - \hat{y}_j^k\right)\left(y_j^k - \hat{y}_j^k\right)$ 此即为式5.10

神经网络

Neural Networks

$$\begin{aligned}\Delta w_{hj} &= -\eta \frac{\partial E_k}{\partial w_{hj}} \\ &= -\eta \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}} \\ &= \eta g_j b_h\end{aligned}$$

此即为式5.11

神经网络

Neural Networks

已知 E_k 和 θ_j 的函数链式关系为

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

$$\downarrow$$

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

f 为Sigmoid函数

所以

$$\frac{\partial E_k}{\partial \theta_j} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \theta_j}$$

神经网络

Neural Networks

deepshare.net
深度之眼

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

神经网络

Neural Networks

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

此即为式5.12

神经网络

Neural Networks

已知 E_k 和 v_{ih} 的函数链式关系为

$$\begin{tikzcd}[column sep=small]
E_k = \frac{1}{2}\sum_{j=1}^{l}\left(\hat{y}_j^k - y_j^k\right)^2 \arrow[d] & & & & \\
\hat{y}_j^k = f(\beta_j - \theta_j) \arrow[dr] & & & & \\
& \beta_j = \sum_{h=1}^{q} w_{hj} b_h \arrow[dr] & & & \\
& & b_h = f(\alpha_h - \gamma_h) \arrow[dr] & & \\
& & & \alpha_h = \sum_{i=1}^{d} v_{ih} x_i &
\end{tikzcd}$$

f 为Sigmoid函数

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

神经网络

Neural Networks

deepshare.net
深度之眼

$$\frac{\partial \beta_j}{\partial b_h} = \frac{\partial \left(\sum_{h=1}^{q} w_{hj} b_h \right)}{\partial b_h} = w_{hj}$$

$$\frac{\partial b_h}{\partial \alpha_h} = \frac{\partial [f(\alpha_h - \gamma_h)]}{\partial \alpha_h} = f'(\alpha_h - \gamma_h) \times 1 = f(\alpha_h - \gamma_h) \times [1 - f(\alpha_h - \gamma_h)] = b_h(1 - b_h)$$

$$\frac{\partial \alpha_h}{\partial v_{ih}} = \frac{\partial \left(\sum_{i=1}^{d} v_{ih} x_i \right)}{\partial v_{ih}} = x_i$$

令 $e_h = -\frac{\partial E_k}{\partial \alpha_h} = -\sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} = b_h\,(1 - b_h) \sum_{j=1}^{l} w_{hj} g_j$ 此即为式5.15

神经网络

Neural Networks

$$
\begin{aligned}
\Delta v_{ih} &= -\eta \frac{\partial E_k}{\partial v_{ih}} \\
&= -\eta \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \cdot \frac{\partial \alpha_h}{\partial v_{ih}} \\
&= \eta e_h x_i
\end{aligned}
$$

此即为式5.13

神经网络

Neural Networks

已知 E_k 和 γ_h 的函数链式关系为

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

$$\downarrow$$

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

f 为Sigmoid函数

$$\downarrow$$

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

$$\downarrow$$

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

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

神经网络

Neural Networks

$$
\begin{aligned}
\frac{\partial E_k}{\partial \gamma_h} &= \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot \frac{\partial b_h}{\partial \gamma_h} \\
&= \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot \frac{\partial [f(\alpha_h - \gamma_h)]}{\partial \gamma_h} \\
&= \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot f'(\alpha_h - \gamma_h) \cdot (-1) \\
&= \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot f(\alpha_h - \gamma_h) \times [1 - f(\alpha_h - \gamma_h)] \cdot (-1) \\
&= \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} \cdot b_h(1 - b_h) \cdot (-1)
\end{aligned}
$$

神经网络

Neural Networks

$$\frac{\partial E_k}{\partial \gamma_h} = \sum_{j=1}^{l} \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot w_{hj} \cdot b_h(1-b_h) \cdot (-1)$$

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

$$= e_h$$

所以

$$\Delta\gamma_h = -\eta \frac{\partial E_k}{\partial \gamma_h}$$

$$= -\eta e_h$$ 此即为式5.14

结 语

在这次课程中，我们学习了西瓜书

神经网络的公式推导

那么在下次课程中，我们将会学习西瓜书

EM算法的公式推导

deepshare.net
深度之眼

联系我们:

电话: 18001992849

邮箱: service@deepshare.net

Q Q: 2677693114

公众号

客服微信