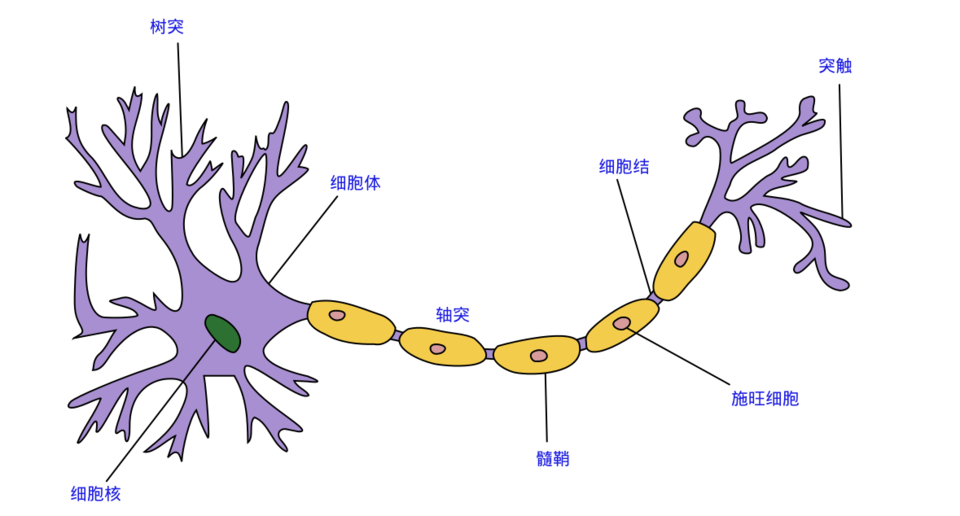
# 5.神经网络

## 5.1.神经元模型

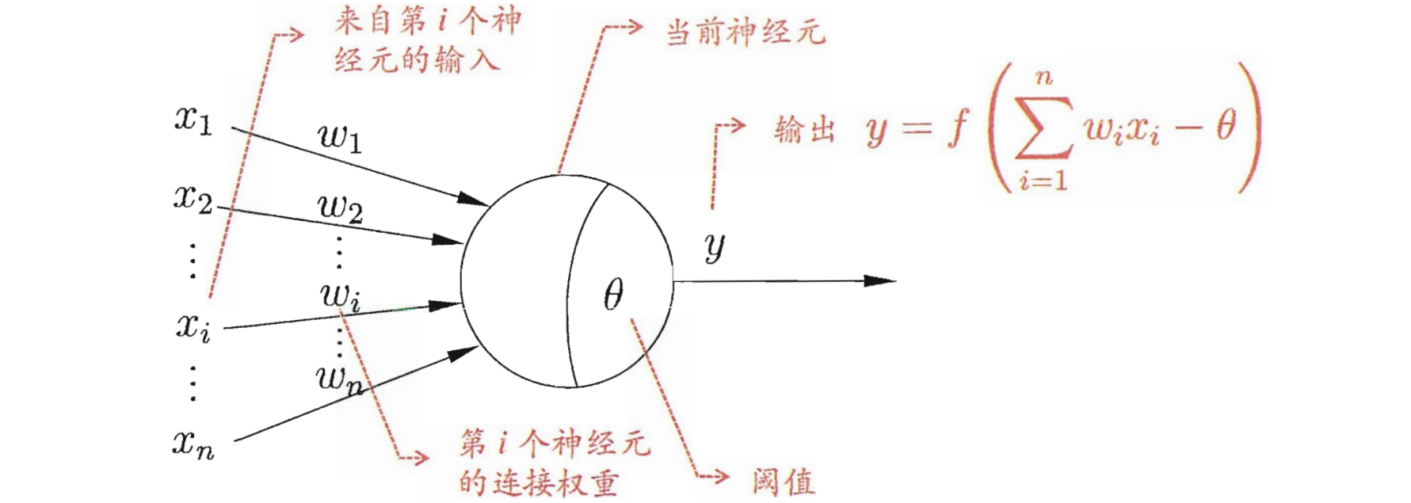
在机器学习中，神经网络一般指的是“神经网络学习”，是机器学习与神经网络两个学科的交叉部分。

所谓神经网络，目前用得最广泛的一个定义是“神经网络是由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”。

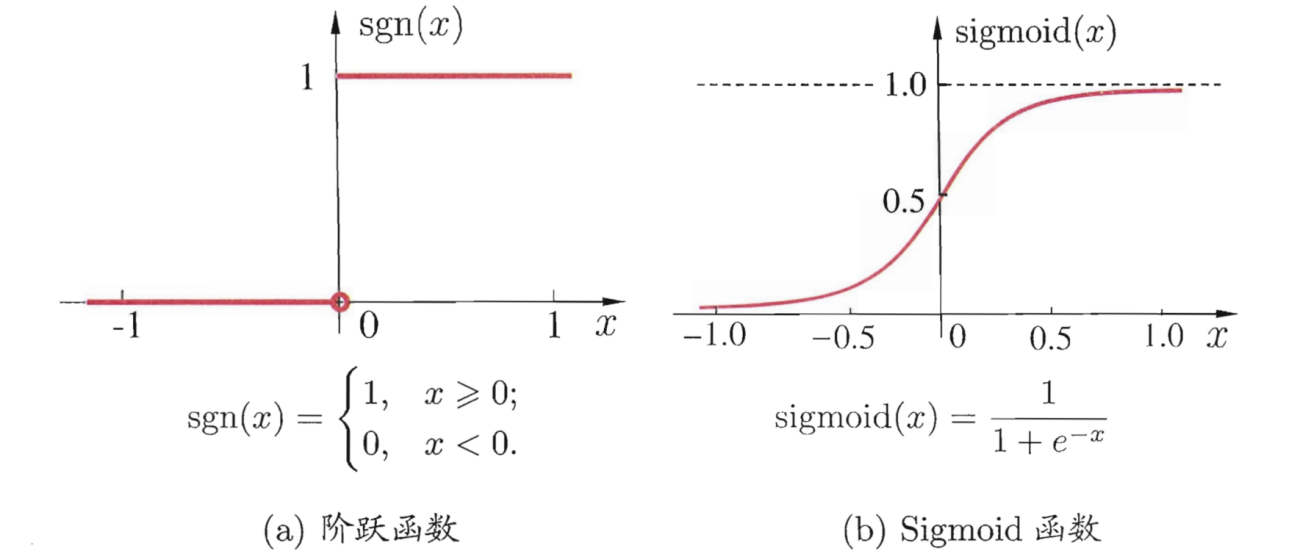
神经网络中最基本的单元是神经元模型（neuron）。在生物神经网络的中，每个神经元通常都有多个树突（dendrite），一个轴突（axon）和一个细胞体（cell body），神经元的生物学结构如下图所示：



一直沿用至今的“M-P神经元模型”正是对这一结构进行了抽象，每个神经元收到n个其他神经元传递过来的输入信号，这些信号通过带权重的连接进行传递，神经元接收到的总输入值将与神经元的阈值进行比较，然后通过“激活函数”（activation function）的处理产生输出如下图所示：



理想中的激活函数是阶跃函数，它将输入值映射为输出值“0”或“1”，显然“1”对应于神经元兴奋，“0”对应于神经元抑制。然而，阶跃函数具有不连续、不光滑等不太好的性质，因此实际常用Sigmoid函数作为激活函数。它把可能在较大范围内变化的输入值挤压到（0，1）输出值范围内，因此有时也称为“挤压函数”（squashing function）。

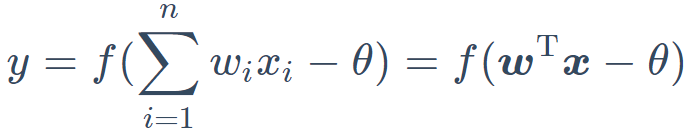


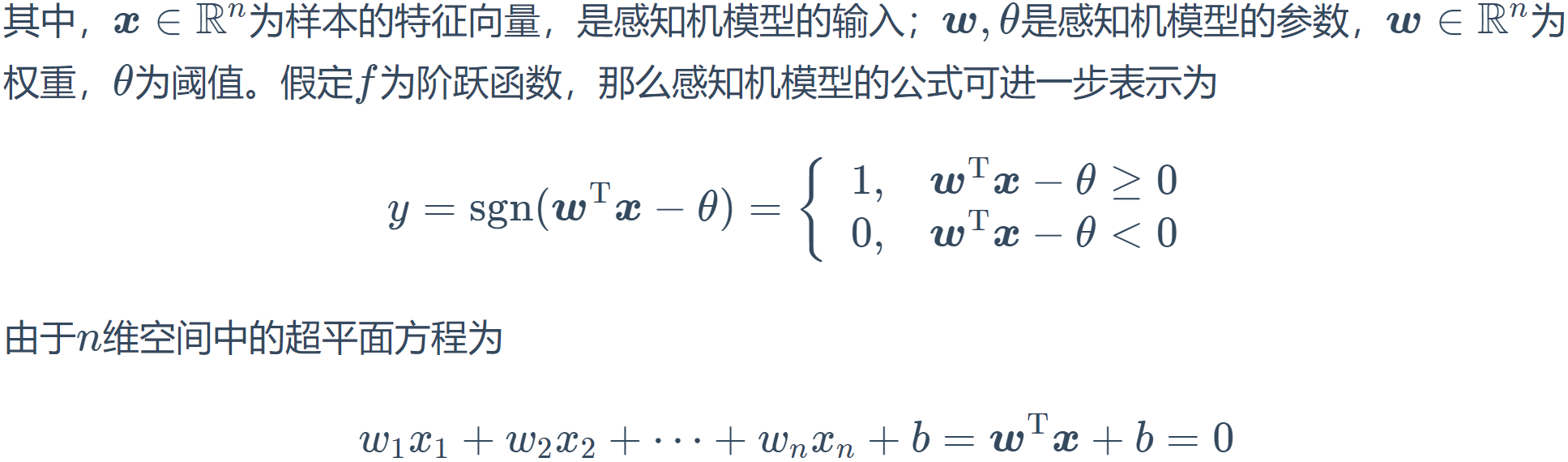
把许多个这样的神经元按一定的层次结构连接起来，就得到了神经网络.

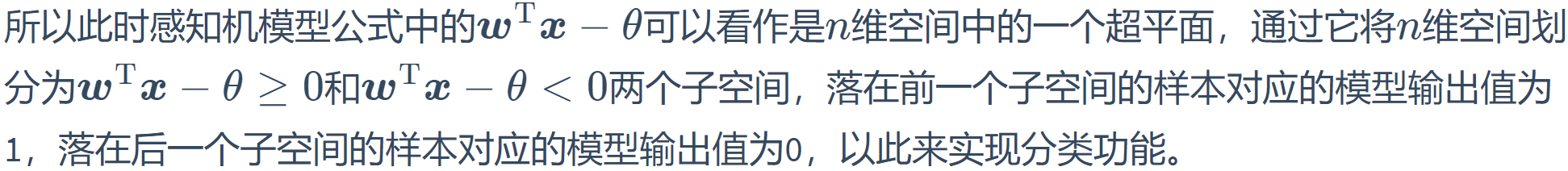
## 5.2.感知机与多层网络

**感知机模型：**

已知感知机由两层神经元组成，故感知机模型的公式可表示为



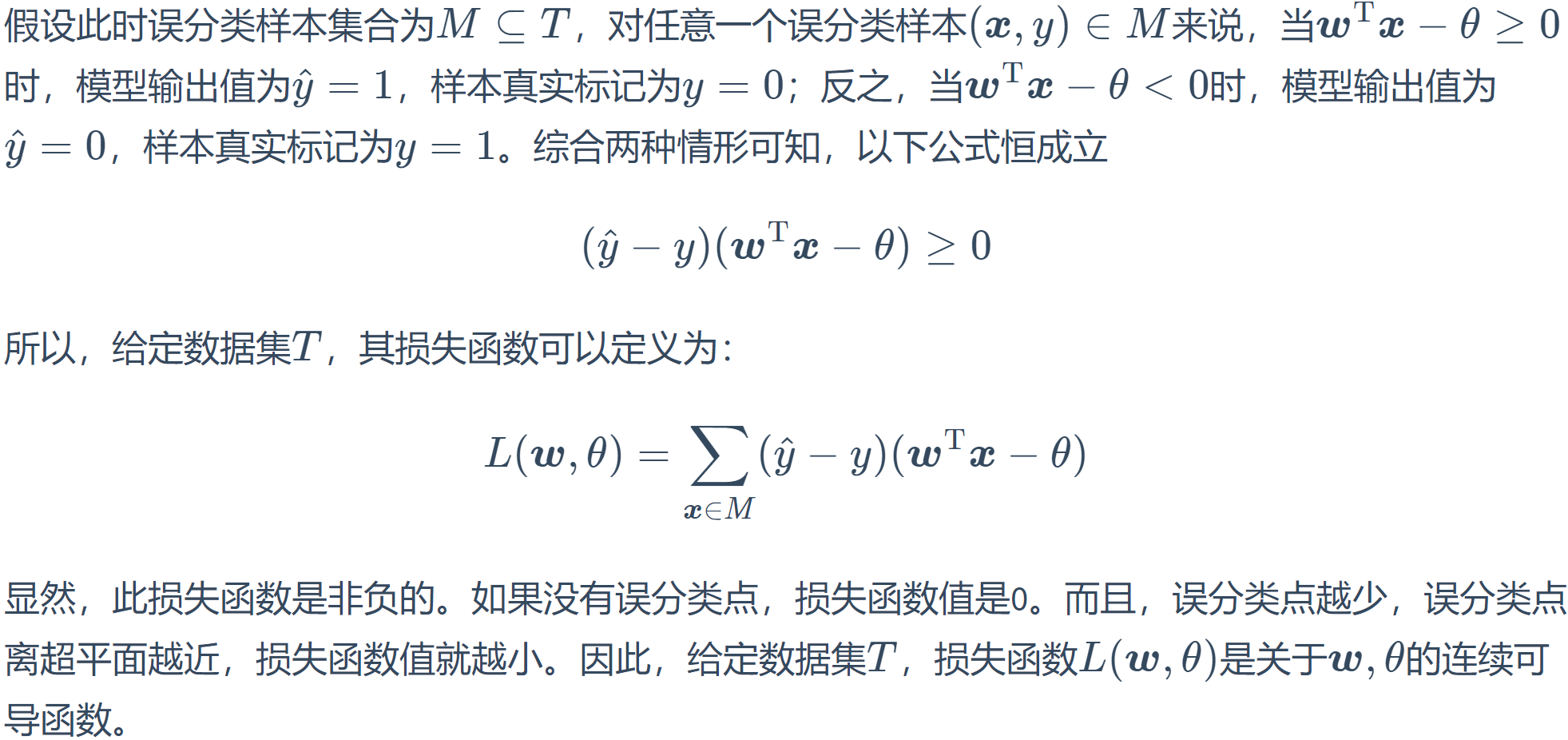


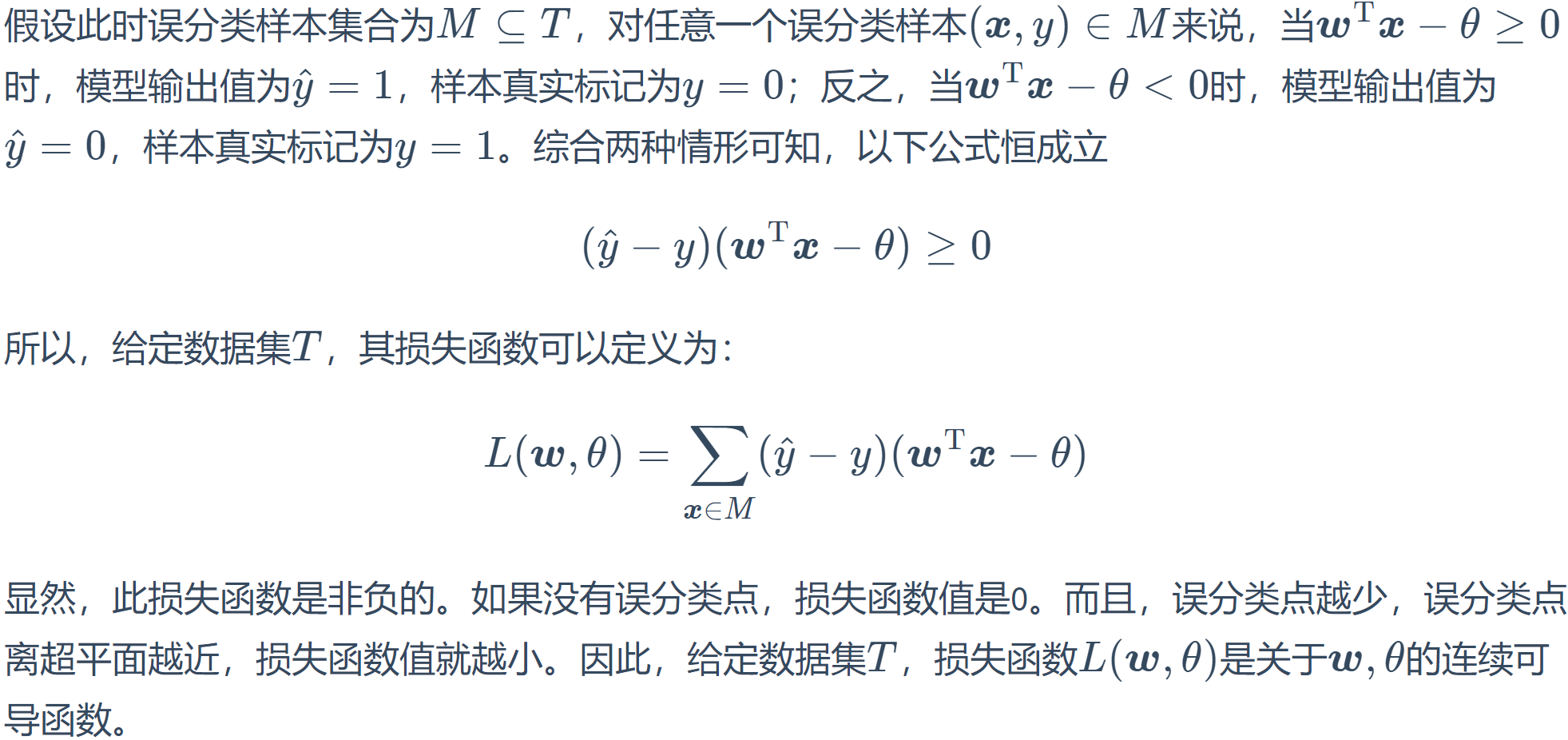


**[感知机学习策略](https://datawhalechina.github.io/pumpkin-book/" \l "/chapter5/chapter5?id=%e6%84%9f%e7%9f%a5%e6%9c%ba%e5%ad%a6%e4%b9%a0%e7%ad%96%e7%95%a5)：**

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分离超平面

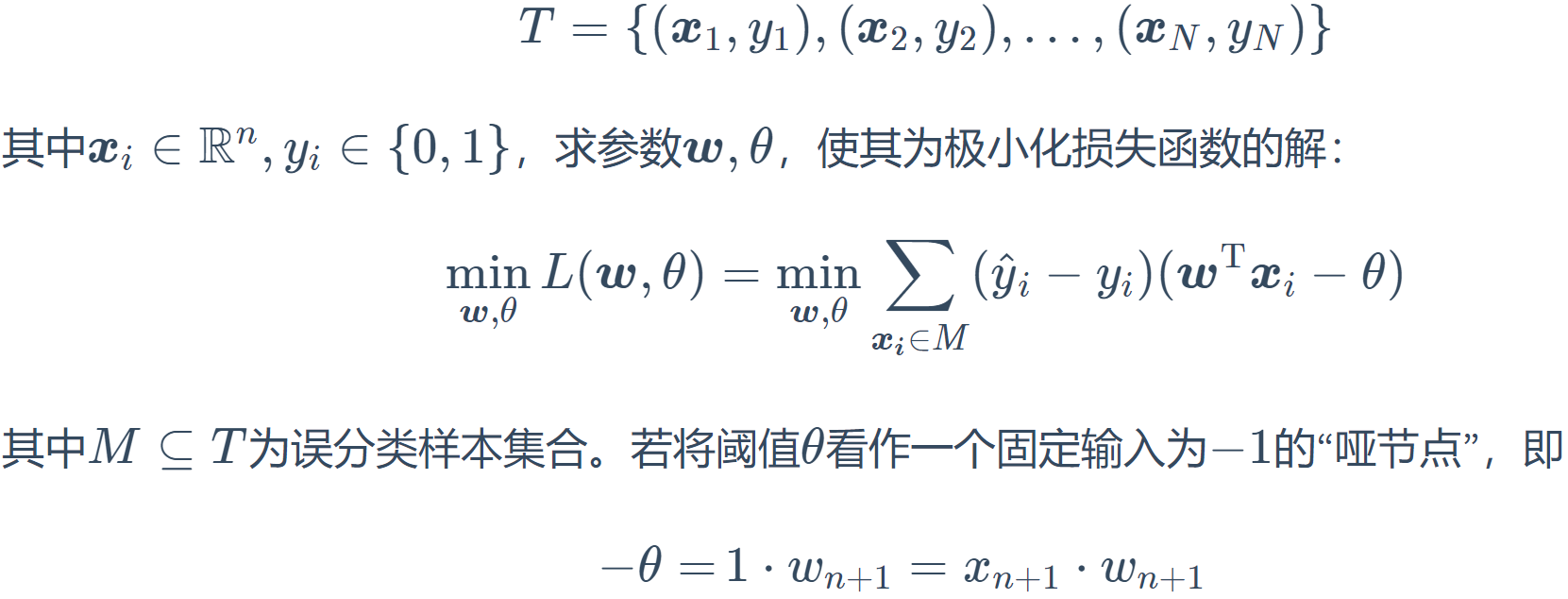


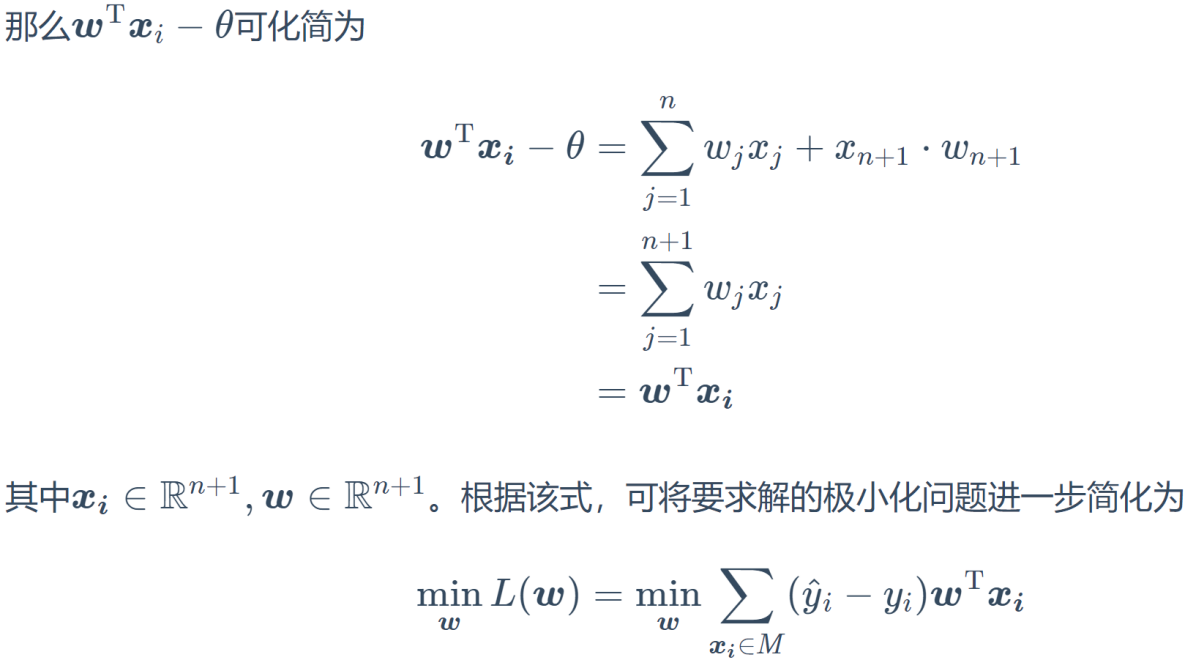


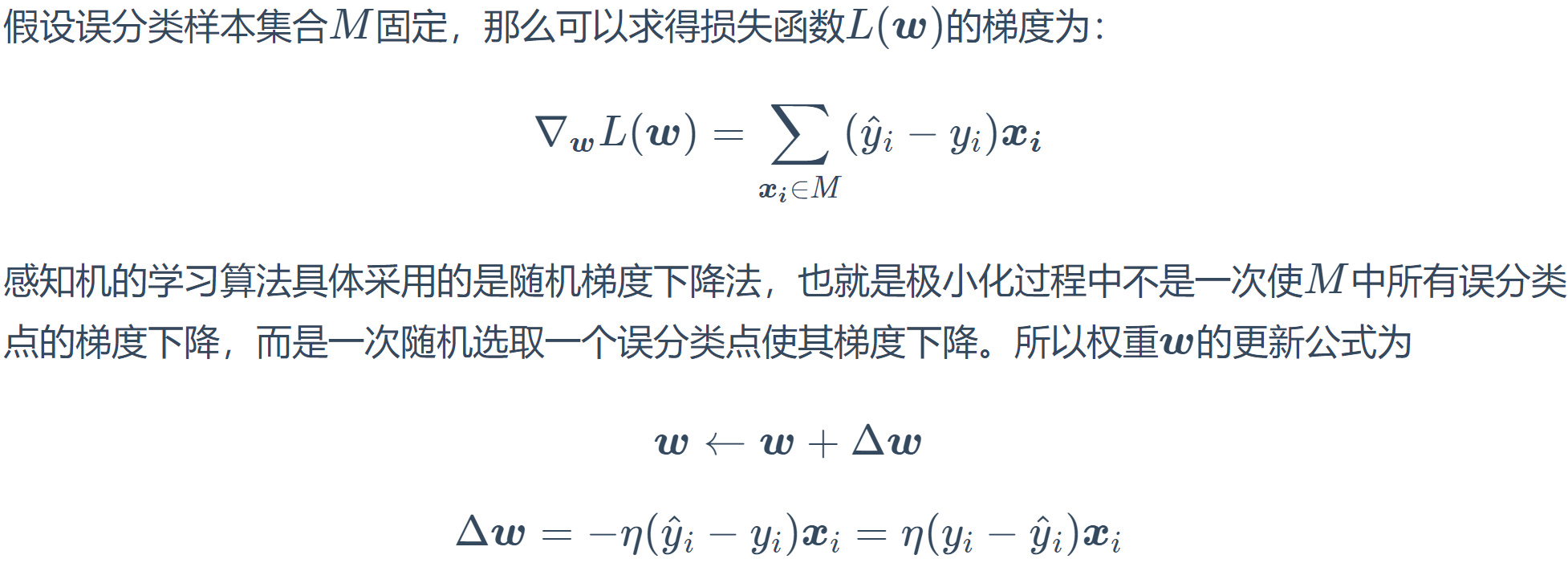


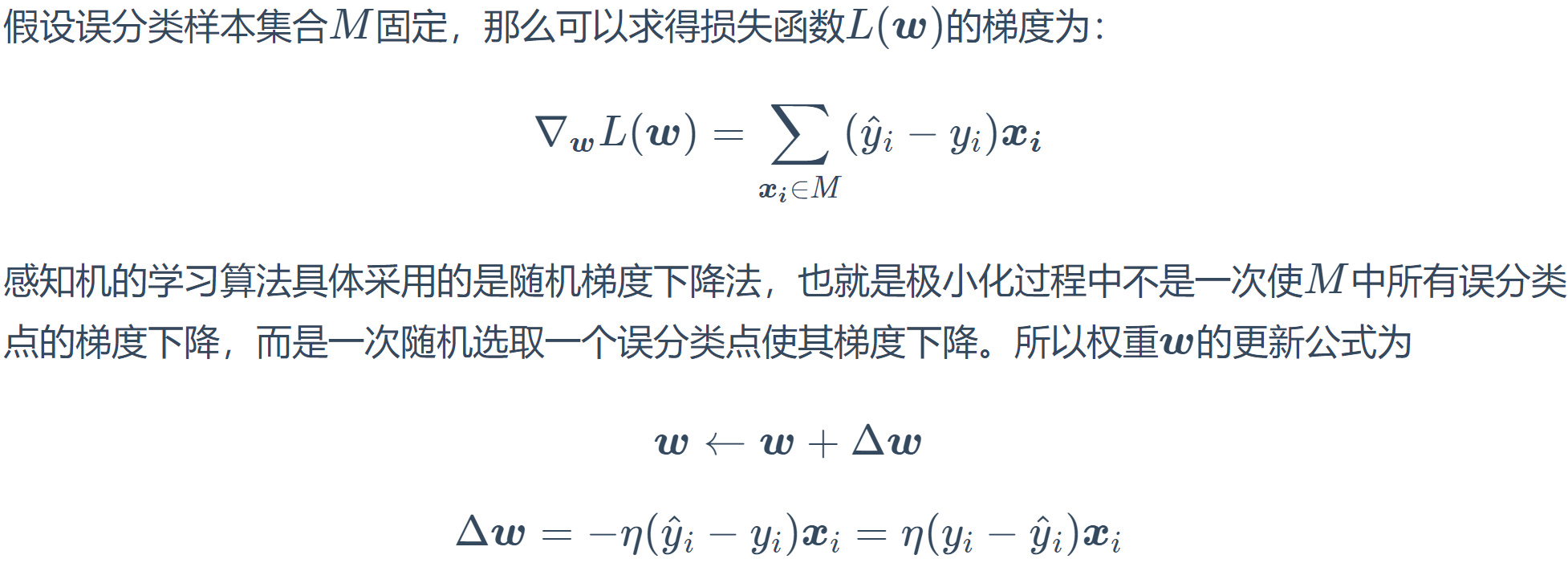
**[感知机学习算法](https://datawhalechina.github.io/pumpkin-book/" \l "/chapter5/chapter5?id=%e6%84%9f%e7%9f%a5%e6%9c%ba%e5%ad%a6%e4%b9%a0%e7%ae%97%e6%b3%95)：**

感知机模型的学习问题可以转化为求解损失函数的最优化问题，具体地，给定数据集



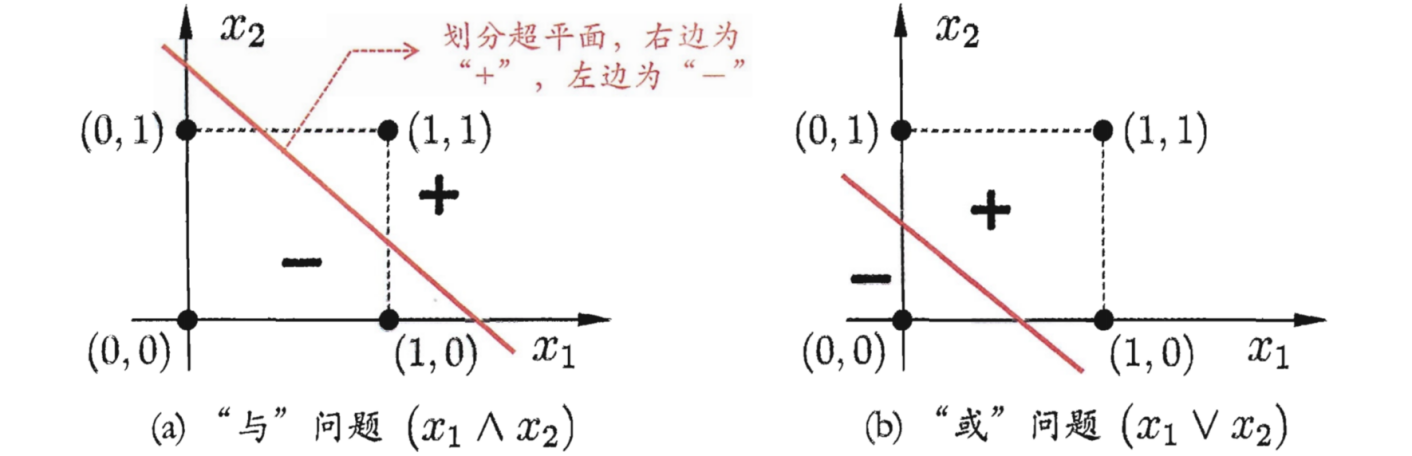


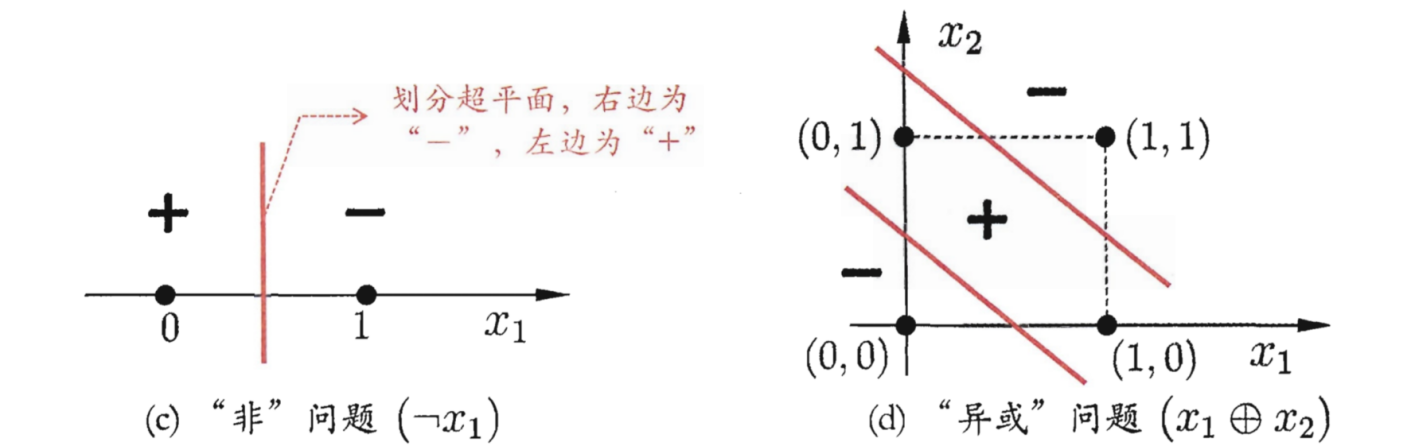




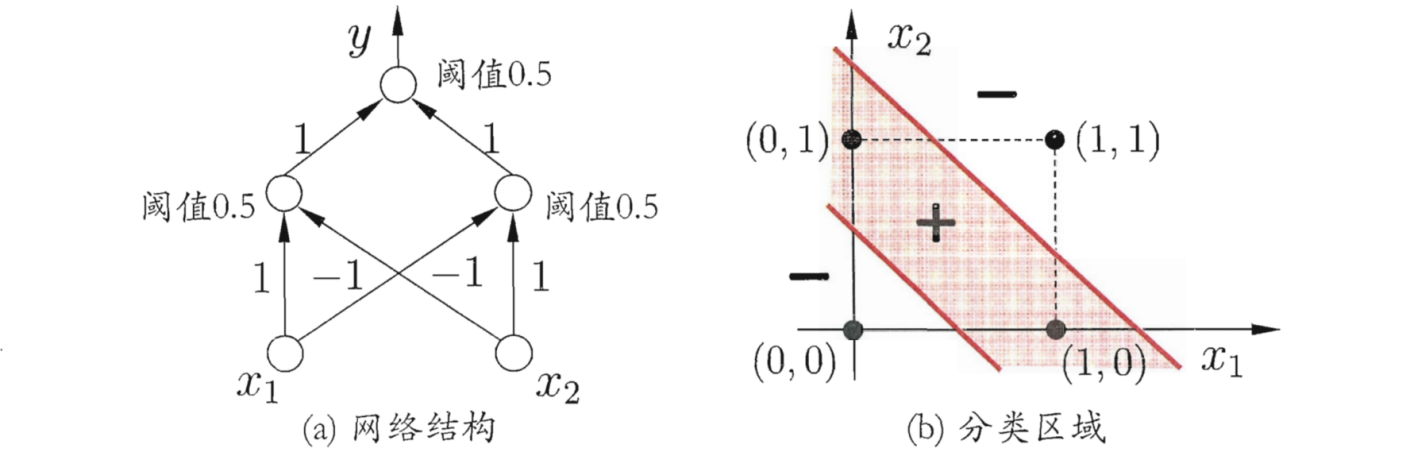
其中称为学习率（learning rate）。

感知机只拥有一层功能神经元，其学习能力非常有限。事实上，上述与、或、非问题都是线性可分（linearly separable）的问题。若两类模式是线性不可分的，则感知机学习过程将会发生振荡，w难以稳定下来，不能求得合适解





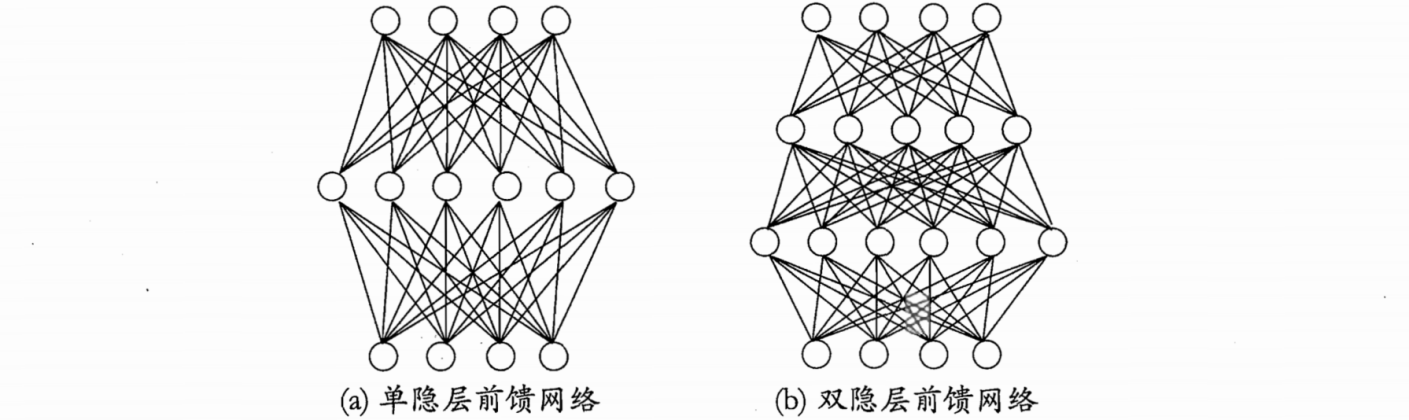
要解决非线性可分问题，需考虑使用多层功能神经元.例如可以用两层感知机解决异或问题。其中，输出层与输入层之间的一层神经元，被称为隐层或隐含层（hidden layer）。



更一般的，常见的神经网络是形如下图所示的层级结构，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.这样的神经网络结构通常称为“多层前馈神经网络”，“前馈”并不意味着网络中信号不能向后传，而是指网络拓扑结构上不存在环或回路。

其中输入层神经元接收外界输入，隐层与输出层神经元对信号进行加工，最终结果由输出层神经元输出；换言之，输入层神经元仅是接受输入，不进行函数处理，隐层与输出层包含功能神经元。

神经网络“学”到的东西，蕴涵在连接权与阈值中.

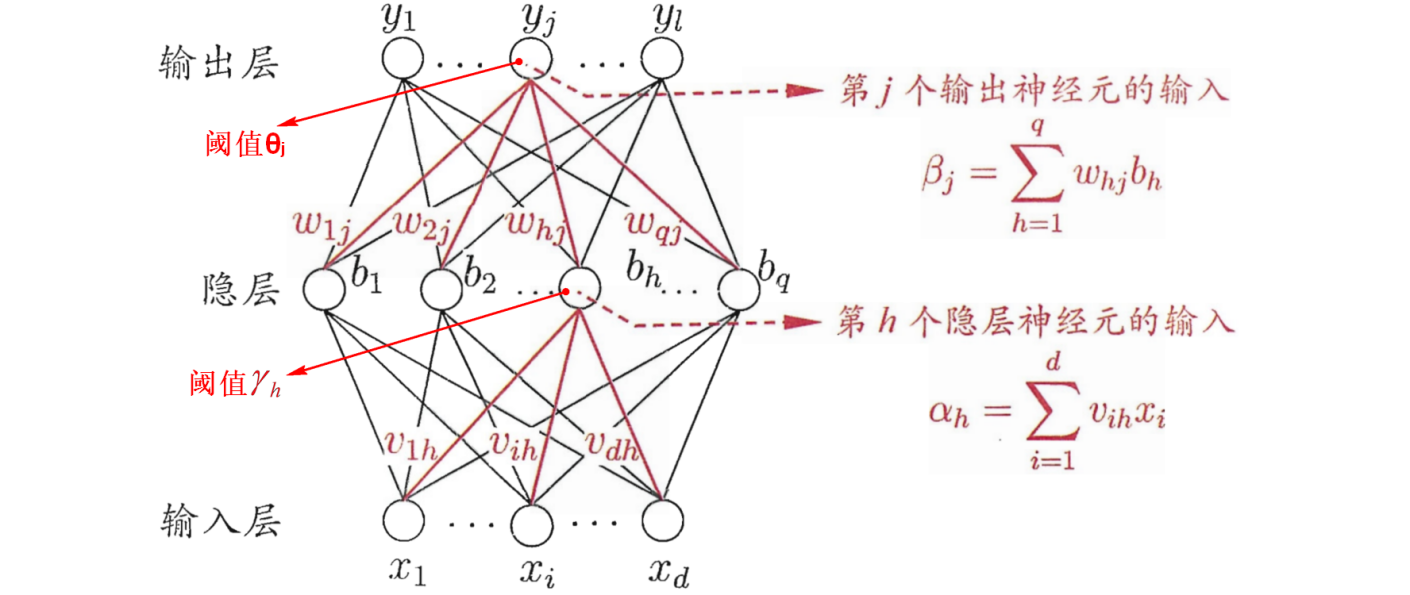


## 5.3.反向传播算法

多层网络的学习能力比单层感知机强得多.欲训练多层网络，需要更强大的学习算法.误差逆传播（error BackPropagation，简称BP）算法就是其中最杰出的代表，它是迄今最成功的神经网络学习算法.

现实任务中使用神经网络时，大多是在使用BP算法进行训练。值得指出的是，BP算法不仅可用于多层前馈神经网络，还可用于其他类型的神经网络，例如训练递归神经网络（RNN），但通常说“BP网络”时，一般是指用BP算法训练的多层前馈神经网络.

一般而言，只需包含一个足够多神经元的隐层，就能以任意精度逼近任意复杂度的连续函数[Hornik et al.,1989]，下面以训练单隐层的前馈神经网络为例，介绍BP神经网络的算法思想。





给定训练集，即输入示例由个属性描述，输出维实值向量.为便于讨论，下图给出了一个拥有d个输入神经元、个输出神经元、个隐层神经元的多层前馈网络结构，其中输出层第个神经元的阈值用表示，隐层第个神经元的阈值用表示.输入层第个神经元与隐层第个神经元之间的连接权为，隐层第个神经元与输出层第个神经元之间的连接权为。记隐层第个神经元接收到的输入为，输出层第个神经元接收到的输入为，其中为隐层第个神经元的输出。假设隐层和输出层神经元都使用Sigmoid函数。

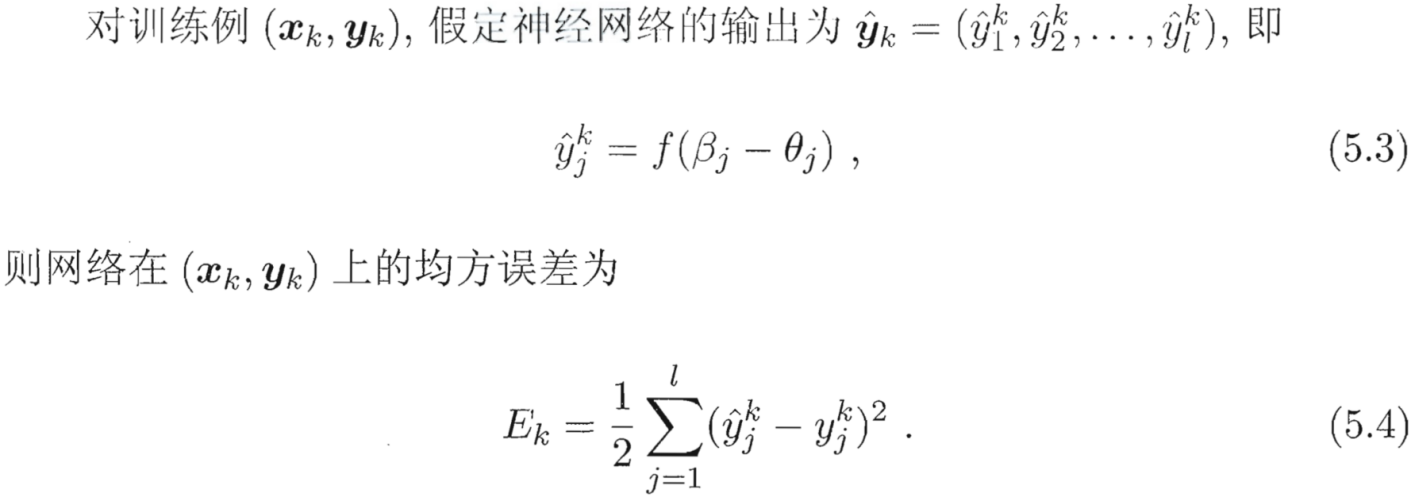
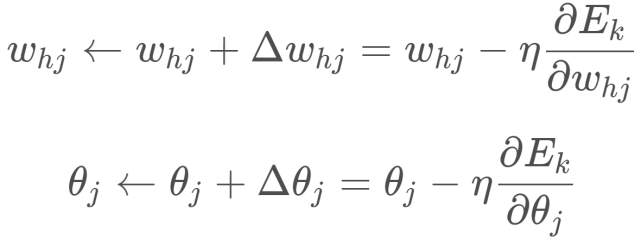
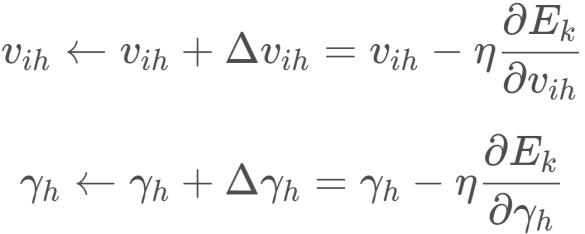


图5.7的网络中有个参数需确定：输入层到隐层的个权值、隐层到输出层的个权值、个隐层神经元的阈值、个输出层神经元的阀值。任意参数的更新估计式为

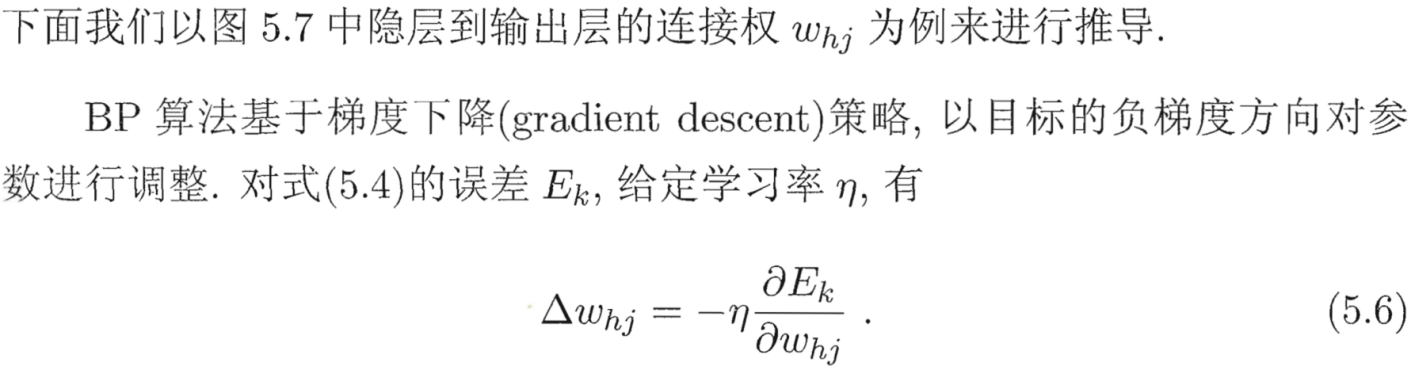


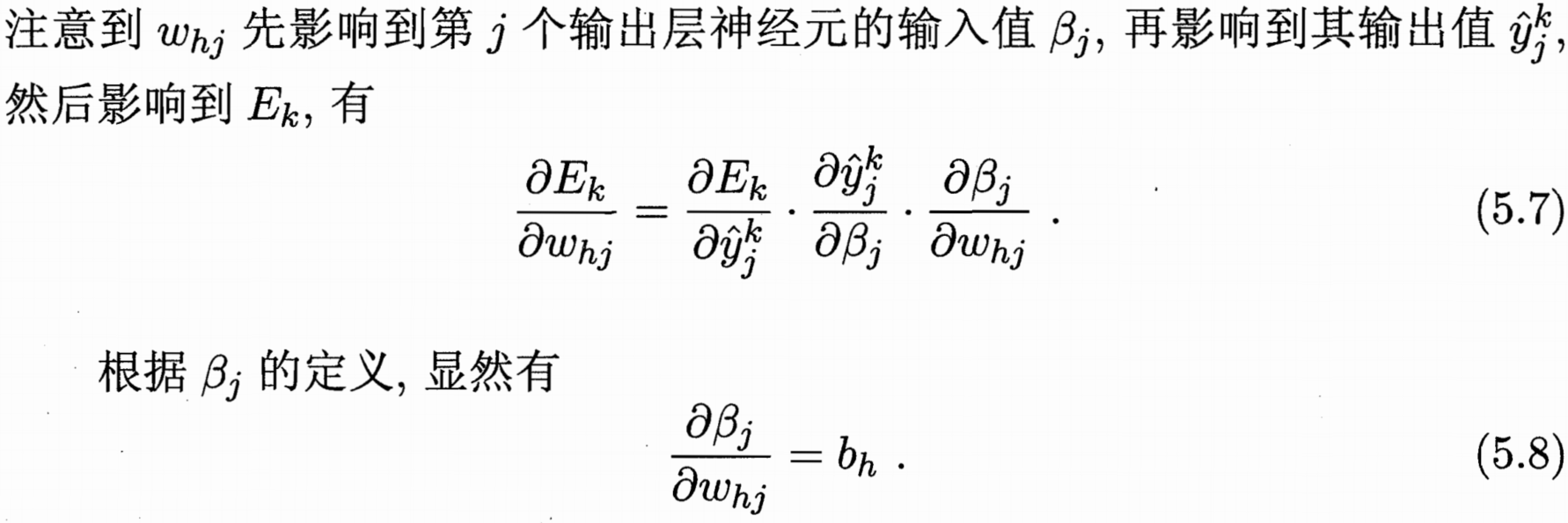
如果按照梯度下降策略，以目标的负梯度方向对参数进行调整，给定学习率，那么各个参数的更新公式为：



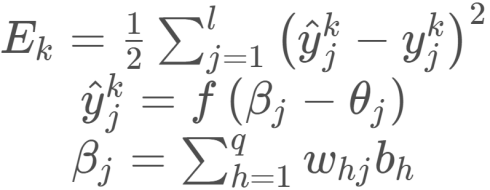


下面我们以为例来进行推导.

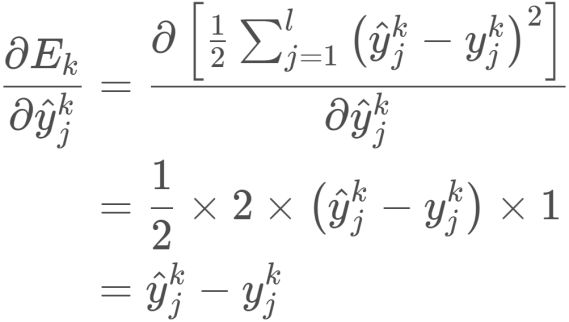


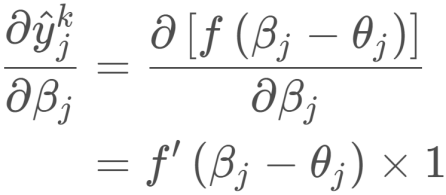


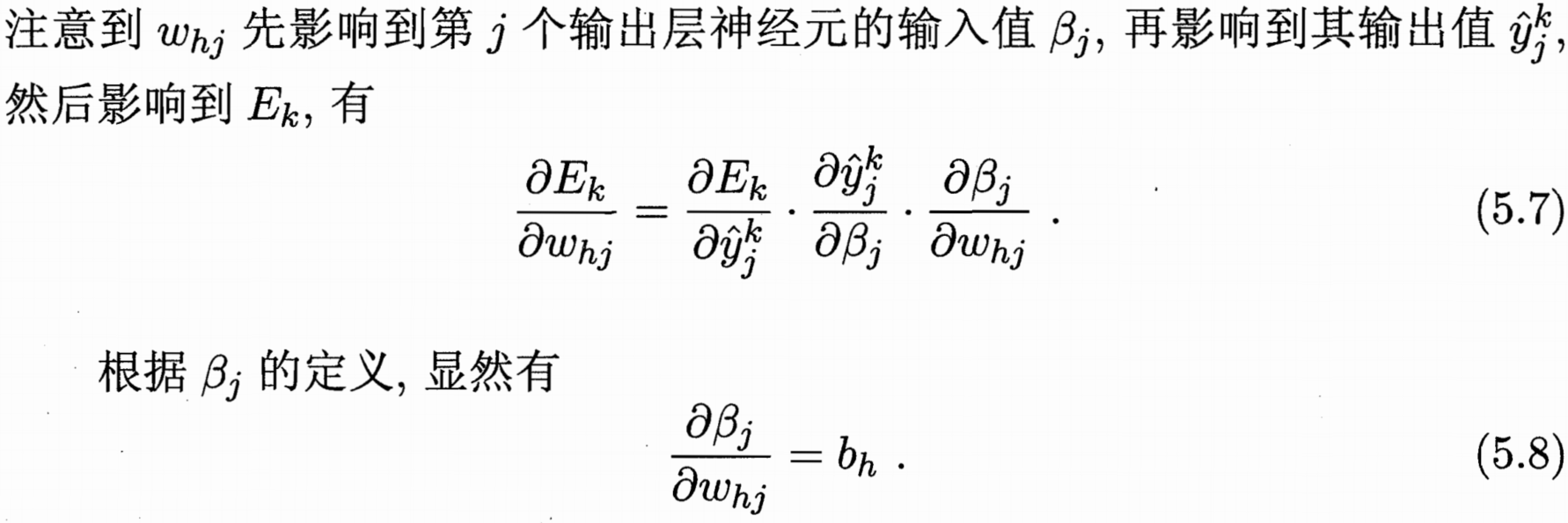
其中



所以



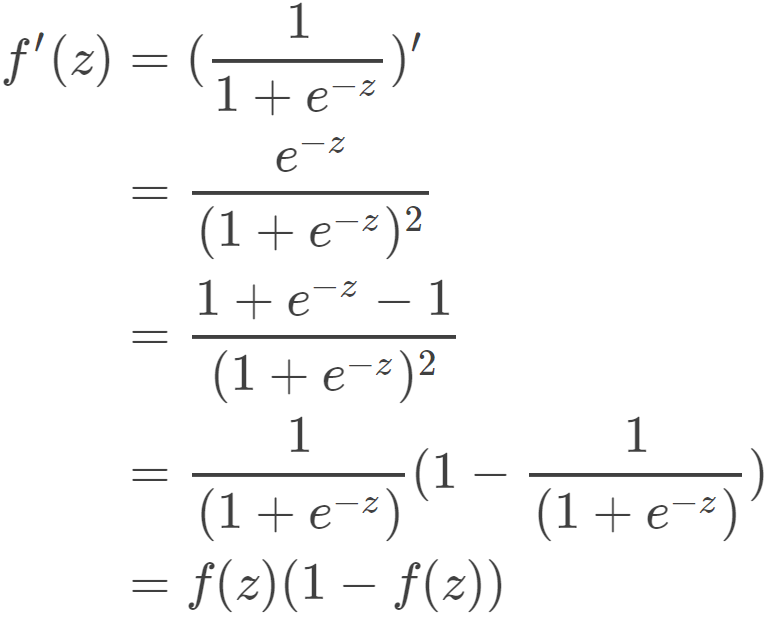




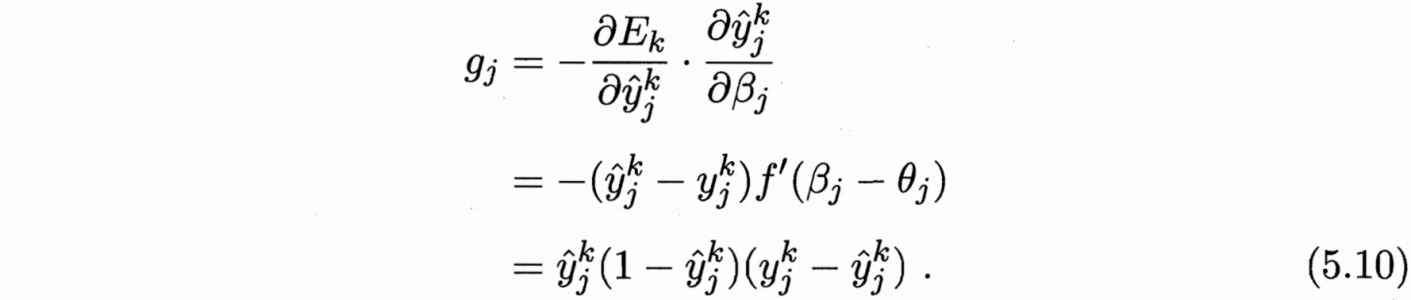
Sigmoid函数有一个很好的性质：

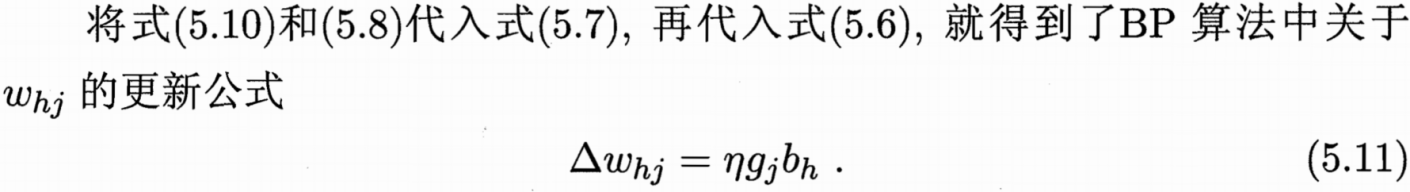


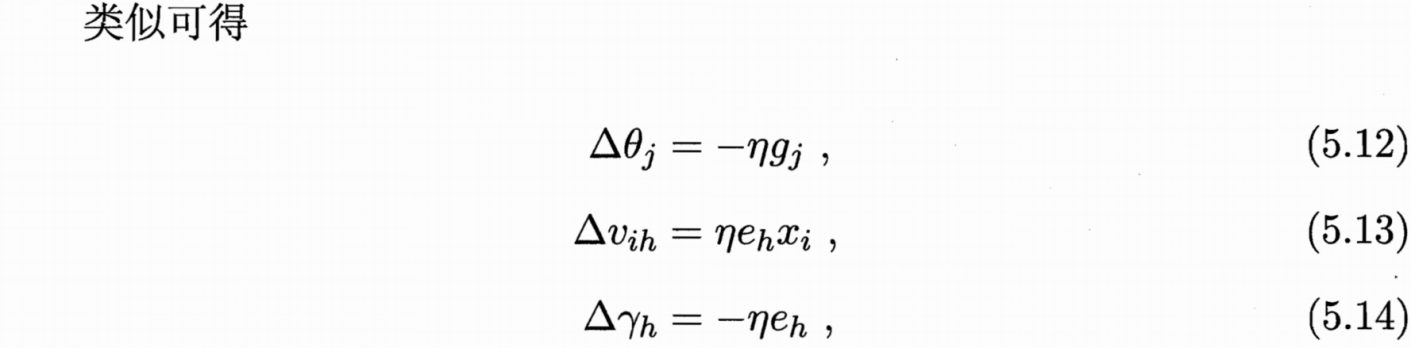
具体求导过程：

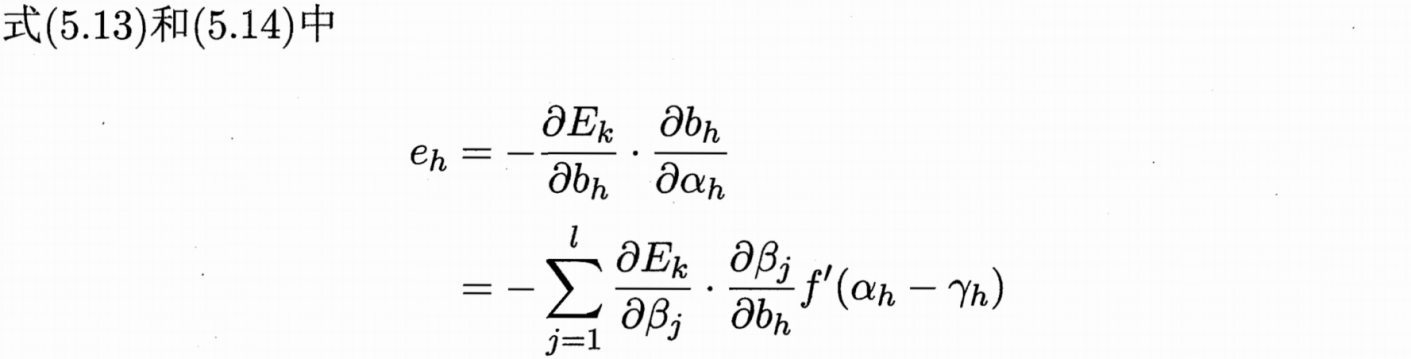


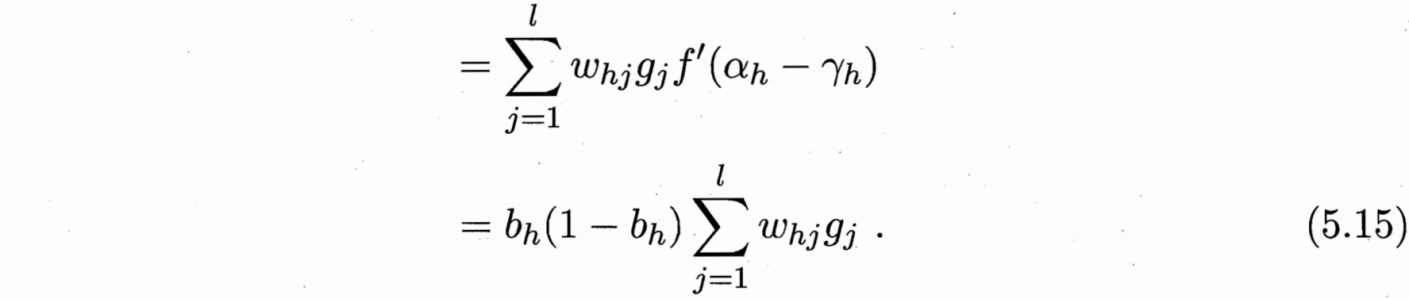
所以，记



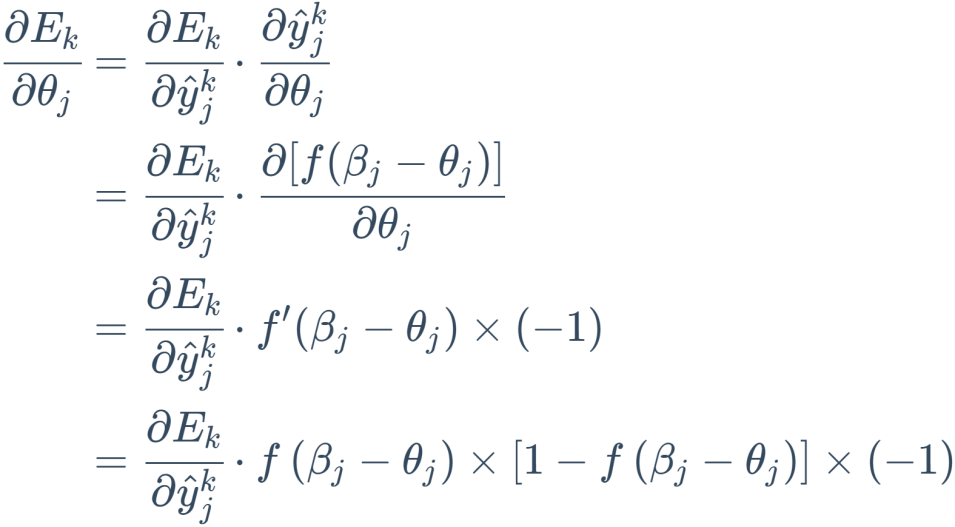


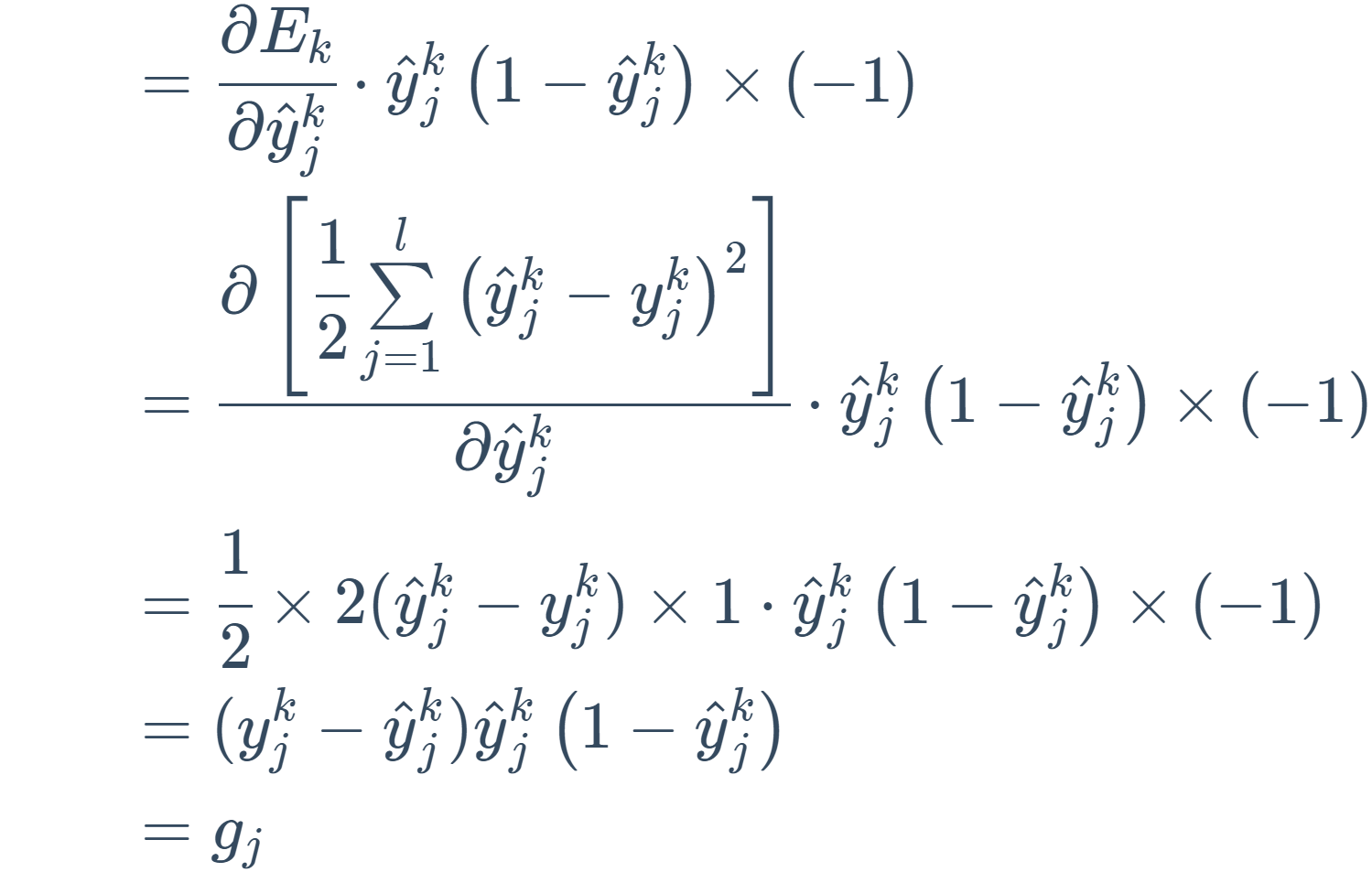


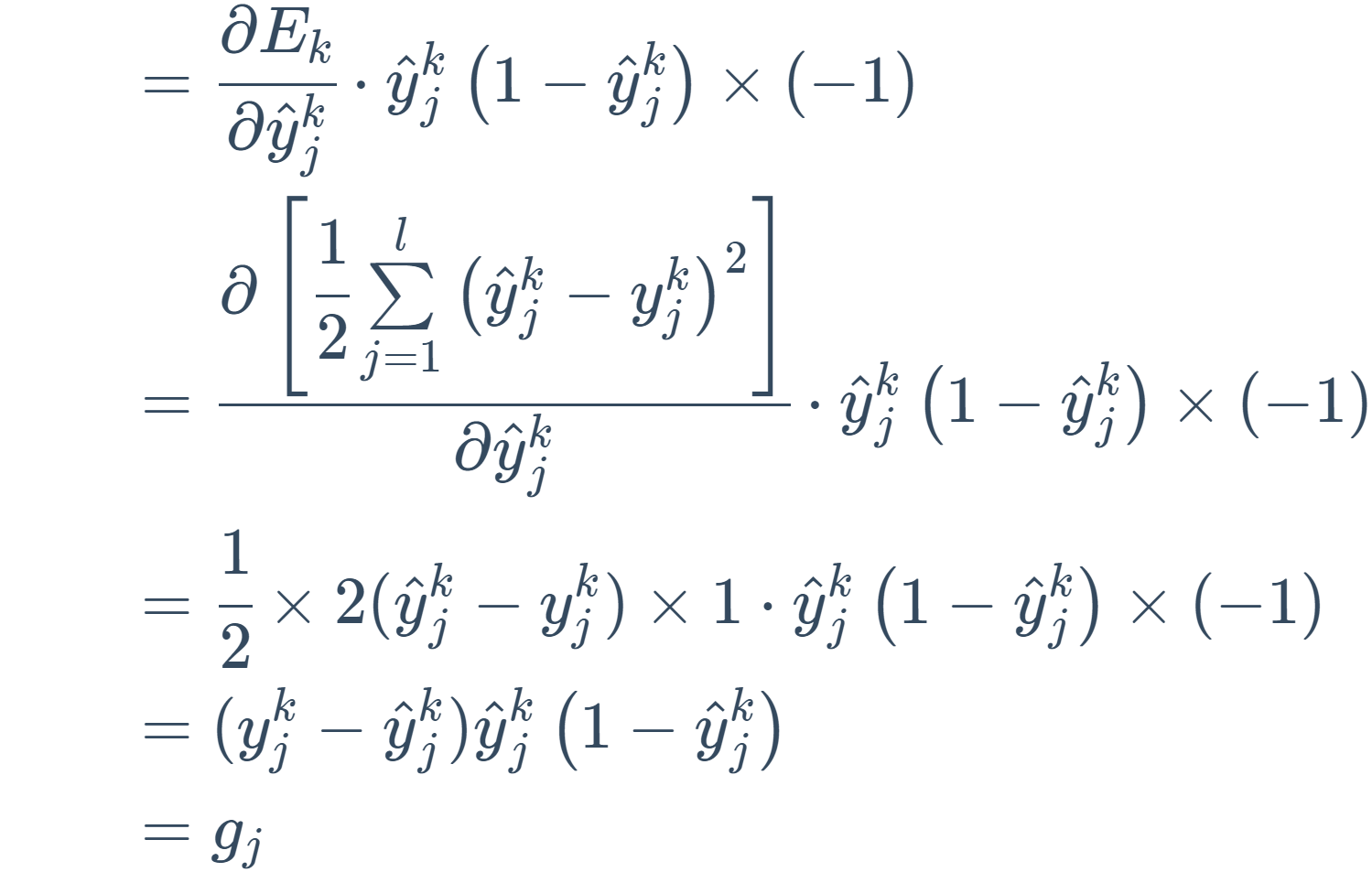




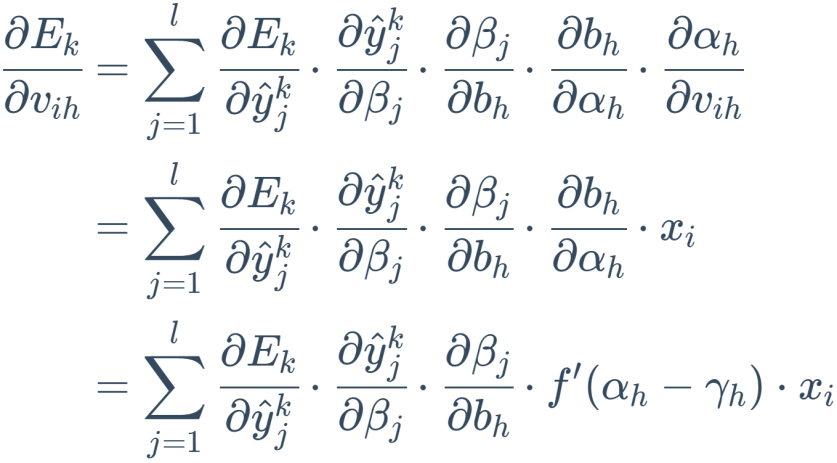
式5.12推导过程：

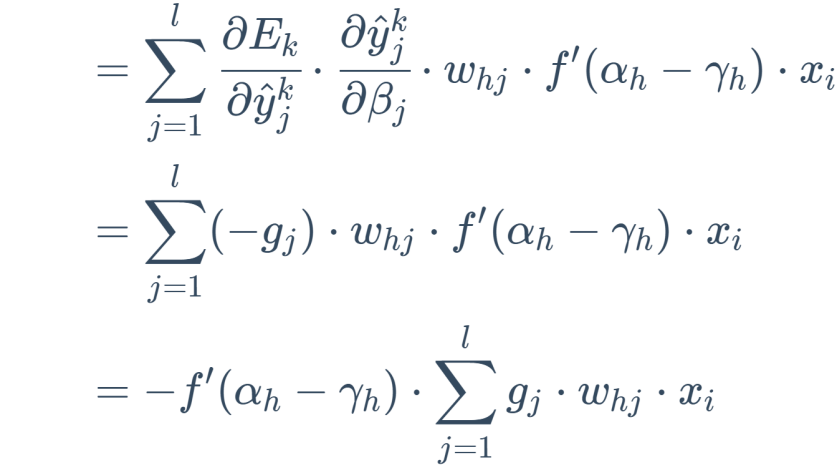


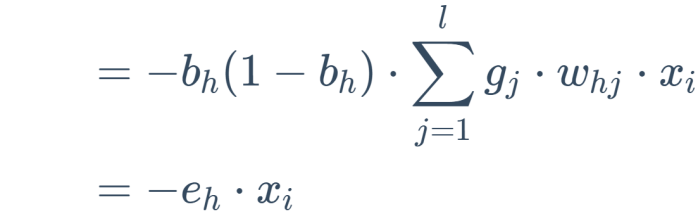




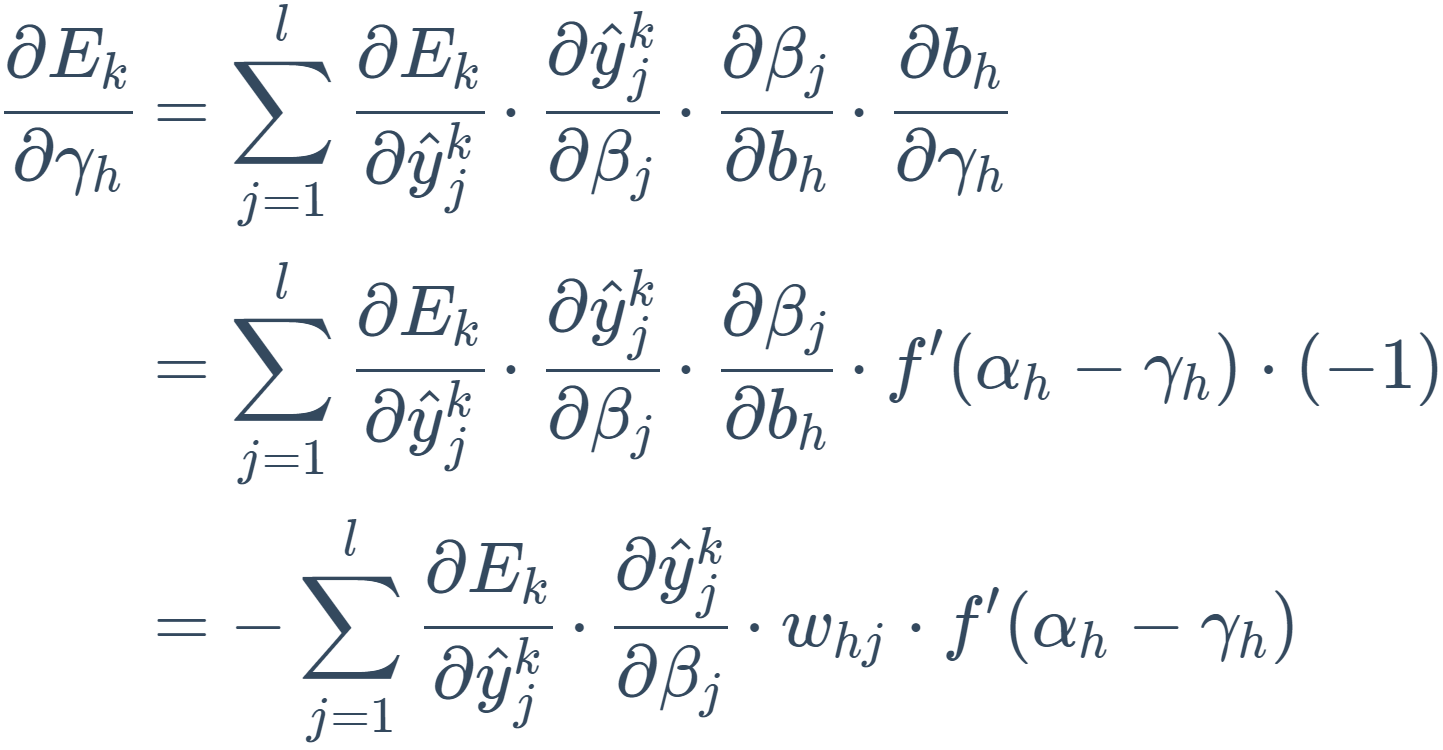
式5.13推导过程：

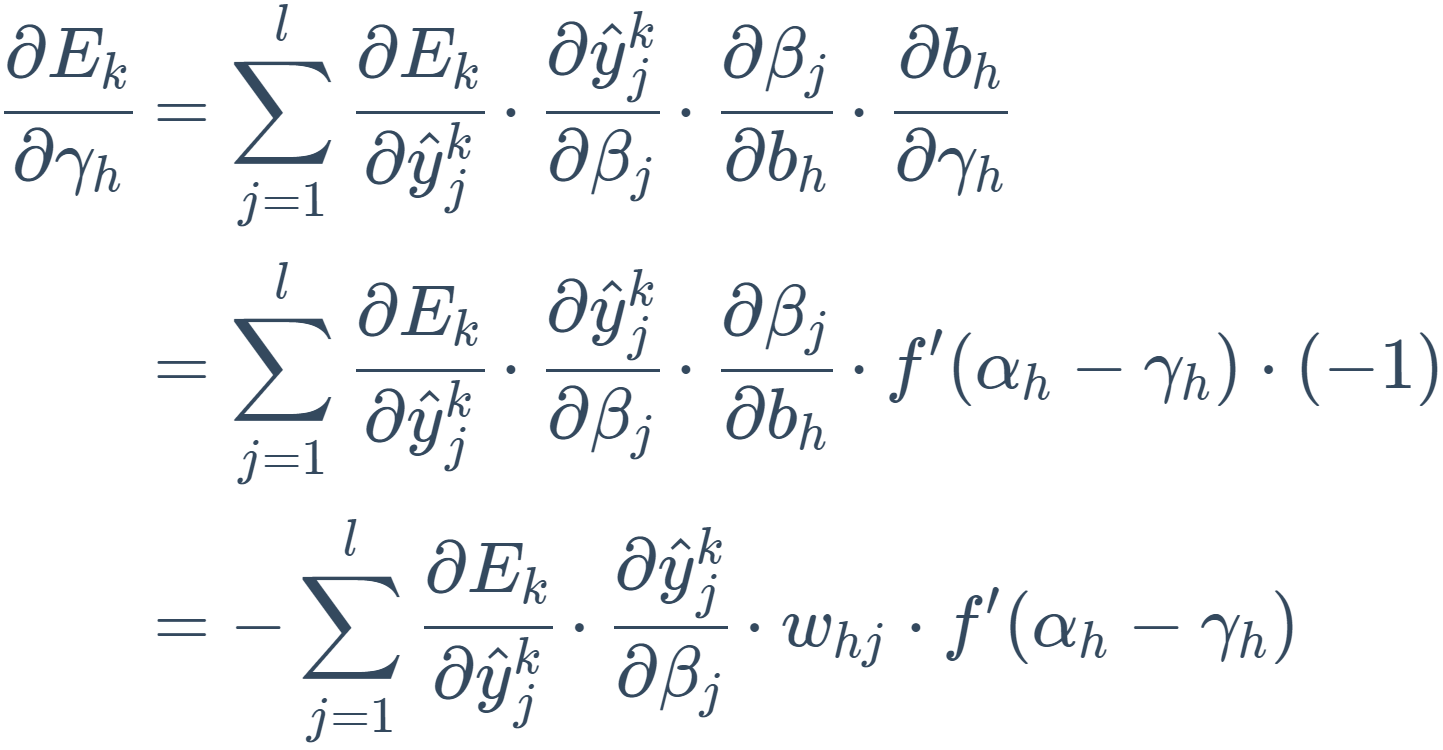


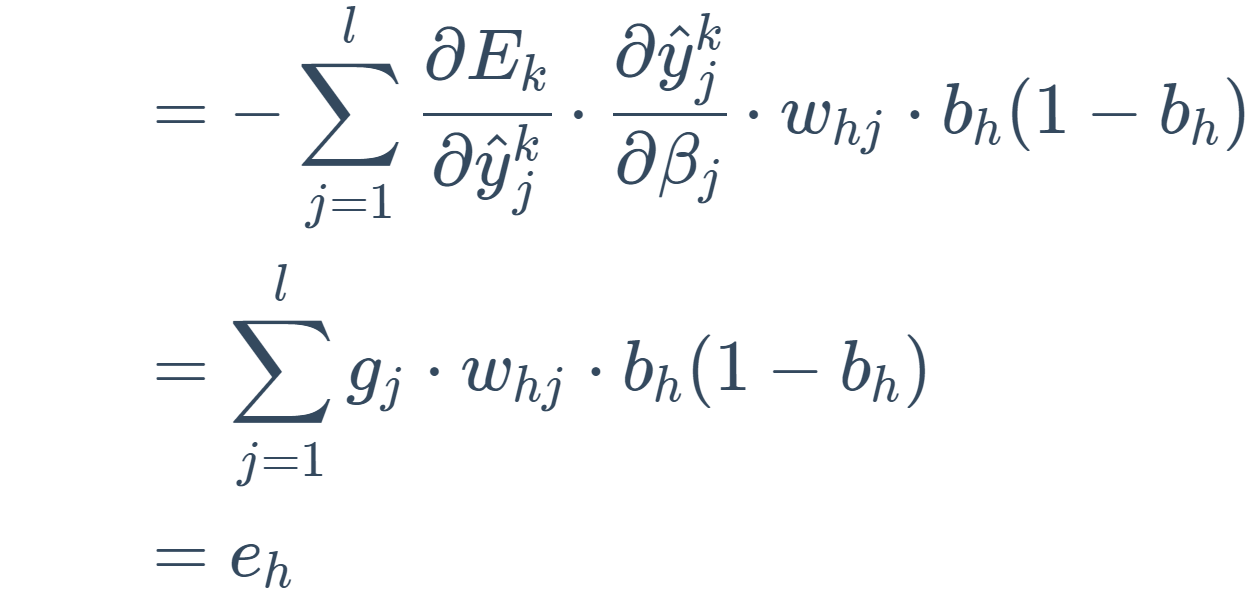


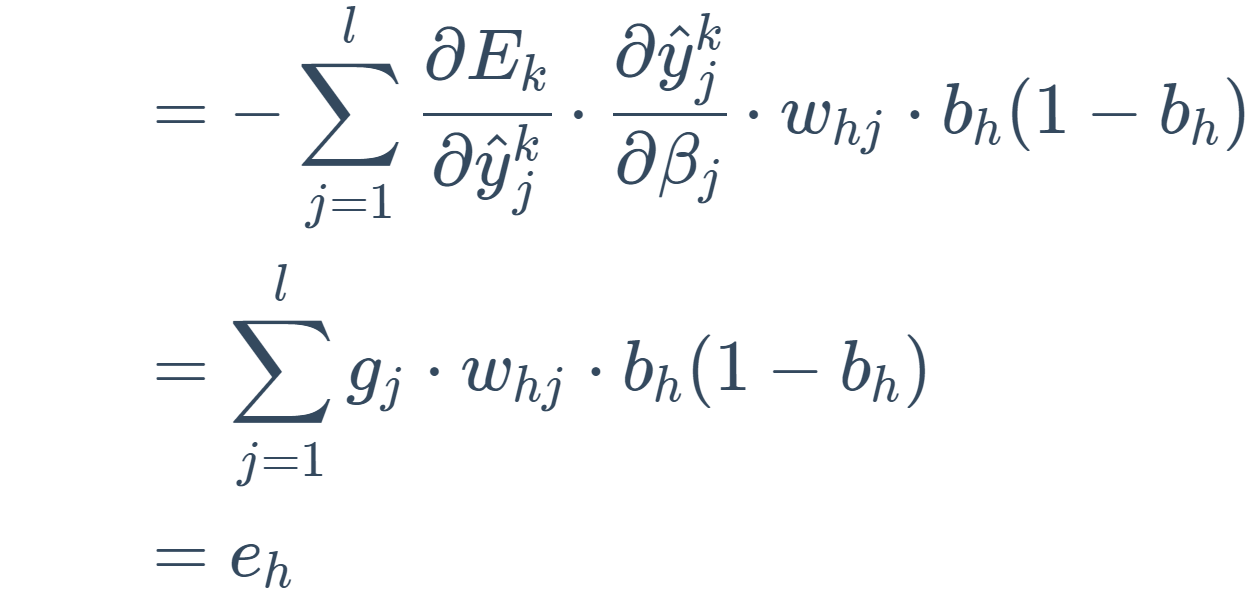


式5.14推导过程：



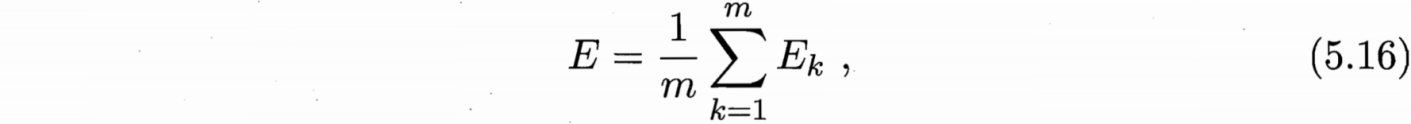






BP算法的更新规则是基于每个样本的预测值与真实类标的均方误差来进行权值调节，即BP算法每次更新只针对于单个样例。

需要注意的是：BP算法的最终目标是要最小化整个训练集D上的累积误差，即：



如果基于累积误差最小化的更新规则，则得到了累积误差逆传播算法（accumulated error backpropagation）。

标准BP算法和累积BP算法的区别类似于随机梯度下降（stochastic gradient descent，简称SGD）与标准梯度下降之间的区别.

标准BP算法每次更新只针对单个样例，参数更新得非常频繁，而且对不同样例进行更新的效果可能出现“抵消”现象.因此，为了达到同样的累积误差极小点，标准BP算法往往需进行更多次数的迭代。

累积BP算法每次读取全部的数据集一遍，进行一轮学习，从而基于当前的累积误差进行权值调整，因此参数更新的频率相比标准BP算法低了很多。

但在很多任务中，累积误差下降到一定程度之后，进一步下降会非常缓慢，这时标准BP往往会更快获得较好的解，尤其是在训练集D非常大时更明显。

累积BP算法与标准BP算法都很常用。

另外，如何设置隐层神经元的个数仍是个未决问题，实际应用中通常靠“**试错法**”（trial-by-error）调整.

BP神经网络强大的学习能力常常容易造成过拟合问题，有以下两种策略来缓解BP网络的过拟合问题：

（1）早停（early stoping）：将数据分为训练集与测试集，训练集用于学习，测试集用于评估性能，若在训练过程中，训练集的累积误差降低，而测试集的累积误差升高，则停止训练。

（2）引入正则化（regularization）：基本思想是在累积误差函数中增加一个用于描述网络复杂度的部分，例如所有权值与阈值的平方和



其中λ∈（0,1）用于对累积经验误差与网络复杂度这两项进行折中，常通过交叉验证法来估计。

防止过拟合的方法常用的还有dropout，增加数据集等

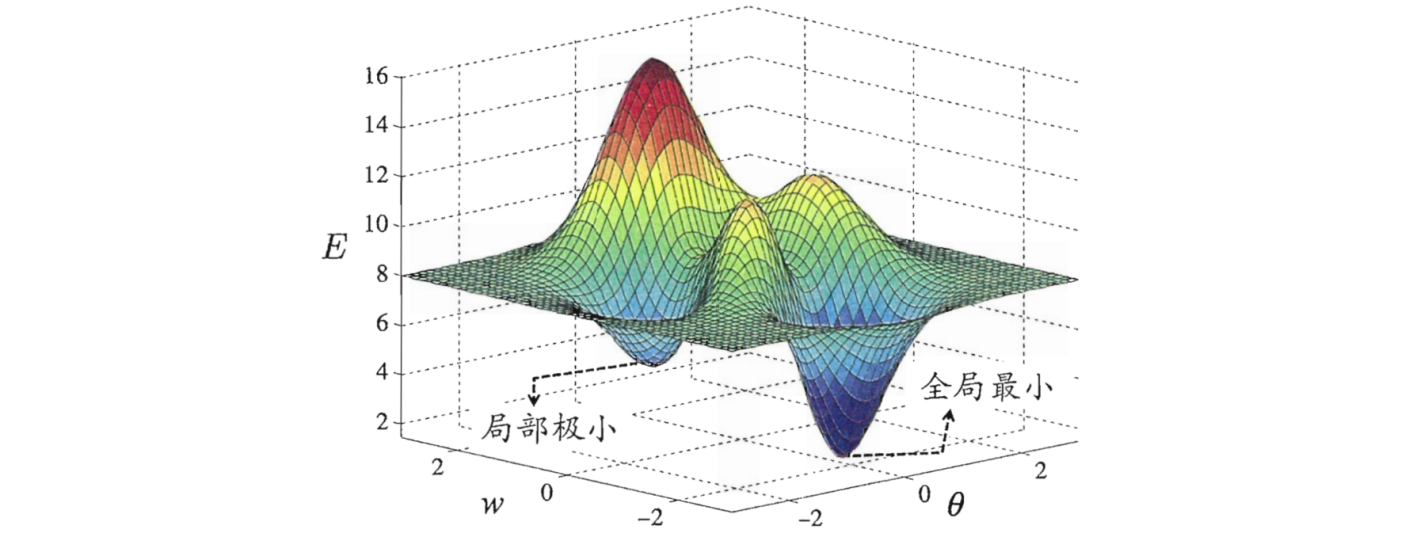
## 5.4.全局最小与局部最小

模型学习的过程实质上就是一个寻找最优参数的过程，例如BP算法试图通过最速下降来寻找使得累积经验误差最小的权值与阈值。

在谈到最优时，一般会提到局部极小（local minimum）和全局最小（global minimum）。

\* 局部极小解：参数空间中的某个点，其邻域点的误差函数值均不小于该点的误差函数值。

\* 全局最小解：参数空间中的某个点，所有其他点的误差函数值均不小于该点的误差函数值。



要成为局部极小点，只要满足该点在参数空间中的梯度为零。局部极小可以有多个，而全局最小只有一个。全局最小一定是局部极小，但局部最小却不一定是全局最小。显然在很多机器学习算法中，都试图找到目标函数的全局最小。

梯度下降法的主要思想就是沿着负梯度方向去搜索最优解，负梯度方向是函数值下降最快的方向，若迭代到某处的梯度为0，则表示达到一个局部最小，参数更新停止。因此在现实任务中，通常使用以下策略尽可能地去接近全局最小。

\* 以多组不同参数值初始化多个神经网络，按标准方法训练，迭代停止后，取其中误差最小的解作为最终参数。

\* 使用“模拟退火”技术，这里不做具体介绍。

\* 使用随机梯度下降，即在计算梯度时加入了随机因素，使得在局部最小时，计算的梯度仍可能不为0，有助于跳出局部极小，但是可能也会跳出全局最小。