神经网络简明教程——回归和分类任务概述

ARD Incubation - Double680

Microsoft AI-EDU　　2020 年 11 月 5 日

1　线性回归

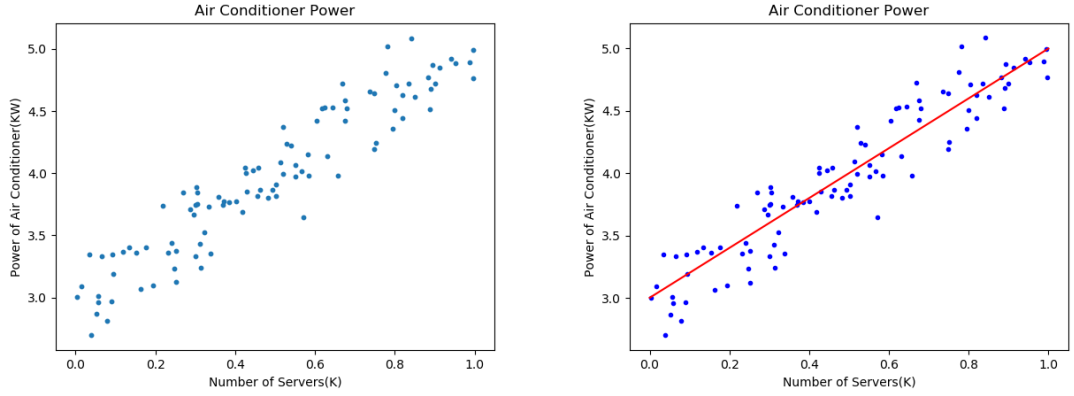
1.1　简单线性回归

从最简单的回归任务开始：给定训练集 $\{(x_i, y_i)\}_{i=1}^n$，采用简单线性模型

$$y = wx + b \tag{1}$$

拟合数据，得到相应的拟合值 $\hat{y}_i = wx_i + b, i = 1, 2, \ldots, n.$

教程中的空调功率数据集就可拟合简单线性模型，散点图和带拟合直线的散点图如下。



样本误差使用带系数的平方误差：

$$\text{loss}_i = \frac{1}{2}(\hat{y}_i - y_i)^2 \tag{2}$$

我们的目标是选取合适的 w, b，使得损失函数

$$J = \sum_{i=1}^{n} \text{loss}_i = \frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{n}(wx_i + b - y_i)^2 \tag{3}$$

取得最小值。

最小二乘法

由于损失函数 J 是关于 w, b 的凸函数，因而 “$\hat{w}, \hat{b}$ 使 J 取最小值” 等价于

$$\frac{\partial J}{\partial w}\Big|_{(\hat{w},\hat{b})} = \sum_{i=1}^{n} x_i(\hat{w}x_i + \hat{b} - y_i) = 0 \tag{4}$$

$$\frac{\partial J}{\partial b}\Big|_{(\hat{w},\hat{b})} = \sum_{i=1}^{n}(\hat{w}x_i + \hat{b} - y_i) = 0 \tag{5}$$

将式 (5) 变形后可得

$$\hat{b} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{w}x_i) \tag{6}$$

为了后序推导过程，首先给定样本均值 $\overline{x}$，样本均值 $\overline{y}$，样本协方差 s_{xy} 和样本方差 s_x^2 的公式表示：

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{7}$$

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{8}$$

$$s_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}) \tag{9}$$

$$s_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2 \tag{10}$$

将式 (6) 结果代入式 (4) 可得

$$\begin{aligned}\sum_{i=1}^{n} x_i(\hat{w}x_i + \hat{b} - y_i) &= \sum_{i=1}^{n} x_i\left(\hat{w}x_i + \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{w}x_j) - y_i\right)\\ &= \sum_{i=1}^{n} x_i(\hat{w}x_i + \overline{y} - \hat{w}\overline{x} - y_i)\\ &= \hat{w}\sum_{i=1}^{n} x_i(x_i - \overline{x}) - \sum_{i=1}^{n} x_i(y_i - \overline{y})\\ &= \hat{w}\sum_{i=1}^{n}(x_i - \overline{x})^2 - \sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}) = 0\end{aligned} \tag{11}$$

式 (11) 意味着

$$\hat{w} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2} = \frac{s_{xy}}{s_x^2} \tag{12}$$

因此我们只需按照式 (12) 求出 $\hat{w}$，再按式 (6) 求出 $\hat{b}$ 即可获得最优简单线性回归模型。

习题：请读者自行验证式 (11) 第四个等号前后逻辑关系的正确性。

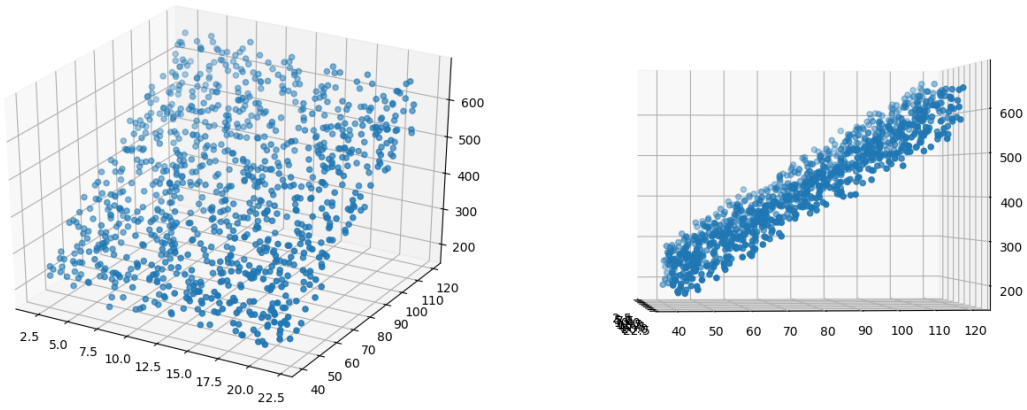
1.2 多变量线性回归模型

考虑多个特征的情形，给定训练集 $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$，其中每个 $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ 含有 p 个特征，采用多变量线性模型

$$y = w_0 + w_1x_1 + \ldots + w_px_p \tag{13}$$

拟合数据，得到相应的拟合值 $\hat{y}_i = w_0 + w_1x_{i1} + w_2x_{i2} + \ldots + w_px_{ip}, i = 1, 2, \ldots, n.$

教程中的通州房价数据集就可拟合多变量线性模型，其正向和侧向图展示如下：



正规方程法

为了简化推导，同时考虑多样本的情况，采用以下矩阵表示：

$$X = \begin{pmatrix} 1 & \mathbf{x}_1 \\ 1 & \mathbf{x}_2 \\ \vdots & \vdots \\ 1 & \mathbf{x}_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \tag{14}$$

$$W = \begin{pmatrix} w_0 & w_1 & \cdots & w_p \end{pmatrix}^\top \tag{15}$$

$$Y = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \end{pmatrix}^\top \tag{16}$$

于是有下式成立：

$$\hat{Y} = XW \tag{17}$$

当 $p = 1$ 时即为简单线性回归模型。

样本误差依然采用式 (2)，损失函数为

$$J = \frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{2}(\hat{Y} - Y)^\top(\hat{Y} - Y) = \frac{1}{2}(XW - Y)^\top(XW - Y) \tag{18}$$

严格来说，为使之取最小值，可用矩阵求导的方法求解；但可以通过设

$$Y = X\hat{W} \tag{19}$$

并在两边同时乘以 $X^\top$ 后得到正规方程：

$$X^\top Y = X^\top X\hat{W} \tag{20}$$

当 $X^\top X$ 可逆时可以求得

$$\hat{W} = (X^\top X)^{-1}X^\top Y \tag{21}$$

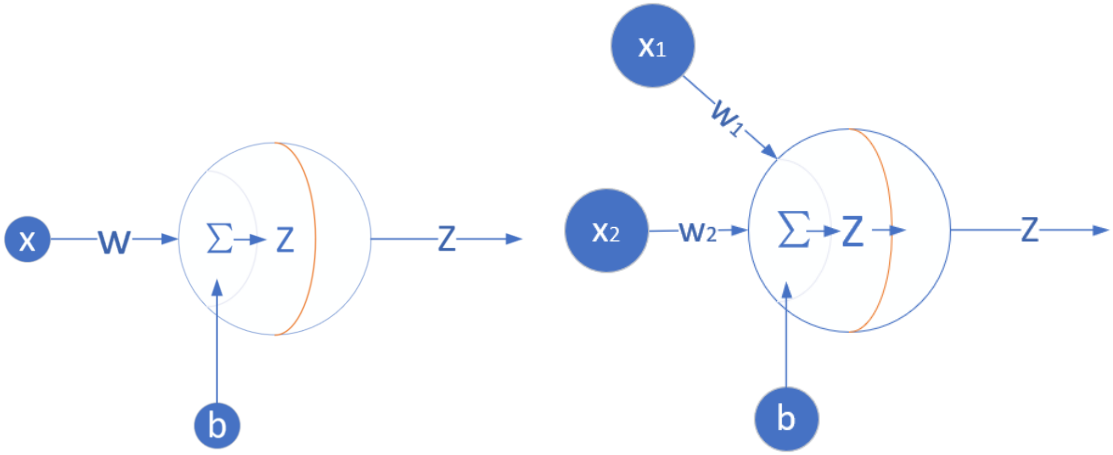
矩阵求导同样可以得到正规方程，可以参考教程的推导过程。

习题：“方阵 $X^\top X$ 可逆” 等价于 “ X 的列向量线性无关”，在什么情况下这一条件无法被满足？

1.3 单层无激活函数的神经网络

为训练线性回归模型，采用单层无激活函数的神经网络即可。输入神经元个数等于样本记录的特征个数，输出含 1 个神经元，表示预测结果。该模型本质上与式 (17) 表示的模型相同。

样本记录 $\mathbf{x}$ 含一或两个特征时，建立的单层神经网络模型分别展示如下：



梯度下降法

我们的目的是寻求合适的 W，使损失函数 J 取得最小值。在神经网络模型中，可以使用梯度下降法更新网络参数。

全样本梯度下降

每次使用全部数据集的误差更新梯度。此方法受单个样本的影响最小，一次计算全体样本速度快，损失函数值没有波动，到达最优点平稳，方便并行计算。但当数据量较大时不能实现（内存限制），训练过程变慢。若初始值不同，可能导致陷入局部最优。

$$W \leftarrow W - \alpha \nabla J(W) \tag{22}$$

单样本随机梯度下降

每次使用单个样本元素的误差更新梯度。此方法训练开始时损失值下降很快，随机性大，找到最优解的可能性大。但受单个样本的影响大，损失函数值波动大，到后期徘徊不前，在最优解附近震荡；且不能并行计算。

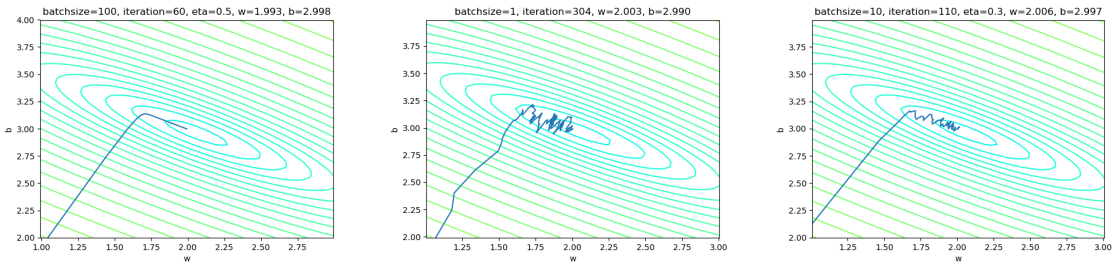
$$W \leftarrow W - \alpha \nabla \mathrm{loss}_i(W) \tag{23}$$

小批量样本梯度下降

每次选取部分样本元素的误差之和更新梯度。此方法介于以上两种方法之间，它不受单样本噪声影响，训练速度较快；但是批大小的选择很关键，直接影响训练结果。

$$W \leftarrow W - \alpha \nabla \left(\sum_{i \in A} \mathrm{loss}_i(W) \right), A \subset \{1, 2, \ldots, n\} \tag{24}$$

以下三图是在一个回归任务中，采用三种不同梯度下降法进行参数更新的过程。



神经网络参数更新过程

对多变量回归情形，采用全批量梯度下降方法。总损失函数表示为：

$$J=\sum_{i=1}^{n}\text{loss}_i=\frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i-y_i)^2=\frac{1}{2}\sum_{i=1}^{n}(w_0+w_1x_{i1}+\ldots+w_px_{ip}-y_i)^2 \tag{25}$$

那么就有

$$\frac{\partial J}{\partial w_0}=\sum_{i=1}^{n}(w_0+w_1x_{i1}+\ldots+w_px_{ip}-y_i)=\sum_{i=1}^{n}(\hat{y}_i-y_i) \tag{26}$$

$$\frac{\partial J}{\partial w_k}=\sum_{i=1}^{n}x_{ik}(w_0+w_1x_{i1}+\ldots+w_px_{ip}-y_i)=\sum_{i=1}^{n}x_{ik}(\hat{y}_i-y_i) \tag{27}$$

综合起来看，采用矩阵表示即可写为：

$$\frac{\partial J}{\partial W}=\begin{pmatrix}\frac{\partial J}{\partial w_0}\\ \frac{\partial J}{\partial w_1}\\ \vdots\\ \frac{\partial J}{\partial w_p}\end{pmatrix}=\begin{pmatrix}1 & 1 & \cdots & 1\\ x_{11} & x_{21} & \cdots & x_{n1}\\ \vdots & \vdots & \ddots & \vdots\\ x_{1p} & x_{2p} & \cdots & x_{np}\end{pmatrix}\begin{pmatrix}\hat{y}_1-y_1\\ \hat{y}_2-y_2\\ \vdots\\ \hat{y}_n-y_n\end{pmatrix}=X^\top(\hat{Y}-Y) \tag{28}$$

因此欲使损失函数最小化，只需事先设置好合适的学习率，每次进行前向计算后即可利用梯度表达式 (28) 进行梯度下降，采用式 (22) 更新参数，重复至收敛条件满足时停止。

数据标准化

在梯度下降过程中，容易遇到损失函数值收敛速度很慢或者发散的情况，有时甚至可能会往无穷大扩散。一个容易想到的思路是设置合适的学习率，但当数据特征较多时，可以考虑进行数据标准化。

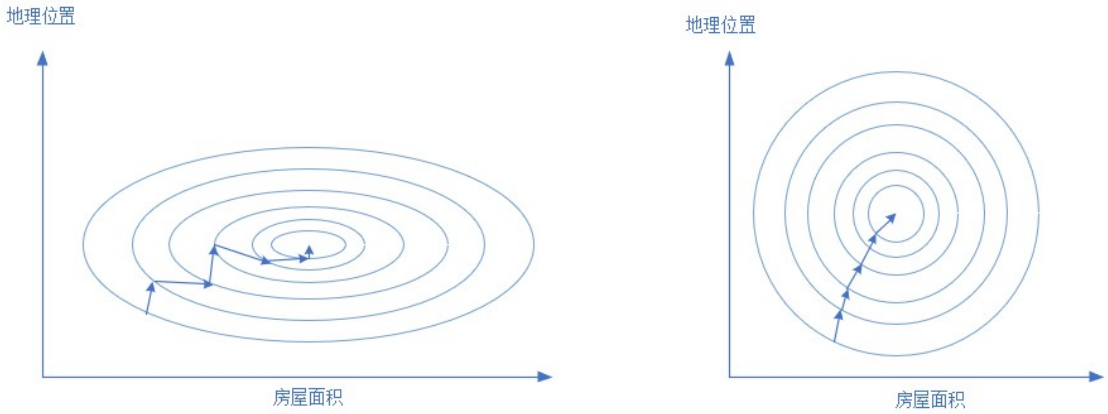
Min-Max 标准化方法是数据分析过程中最常用的标准化方法之一，表达式和逆过程表达式为：

$$z=\frac{x-x_{\min}}{x_{\max}-x_{\min}} \tag{29}$$

$$x=(x_{\max}-x_{\min})z+x_{\min} \tag{30}$$

在训练模型之前，记录训练数据各个特征的两个最值，即可按照式 (29) 进行标准化，然后再投入模型训练；也可以同时对标签数据进行标准化，训练得到一个神经网络模型，但在之后执行预测任务时，需执行一步逆标准化过程，以得到与原始数据量级相同的输出。

对房价数据集而言，执行样本特征数据标准化与否对网络学习梯度下降的过程颇有影响，如下图。



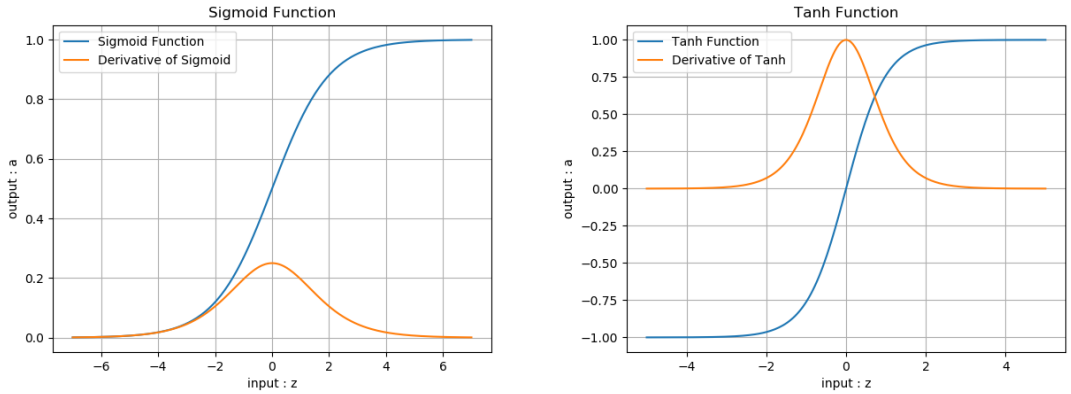
习题：房价数据集的每个训练样本元素含有两个特征，假设第一个特征的取值范围是 [0,10000]，第二个特征的取值范围是 [0,0.001]，如果不进行数据标准化而直接训练神经网络模型，可能会产生怎样的后果？

2 非线性回归

2.1 激活函数

在上一部分，我们介绍了无激活函数的单层神经网络，但是这样的网络仅能模拟线性模型。对非线性模型而言，激活函数不可或缺，它为网络模型引入了非线性因素。

常用的激活函数有很多种。挤压型激活函数包含 Sigmoid 函数、tanh 函数等，如下图。



Sigmoid 函数的值介于 $(0,1)$ 间，可以模拟一些概率分布，它的表达式和导数公式如下：

$$\text{Sigmoid}(z) = \frac{1}{1+e^{-z}} \to a \tag{31}$$

$$\text{Sigmoid}'(z) = a(1-a) \tag{32}$$

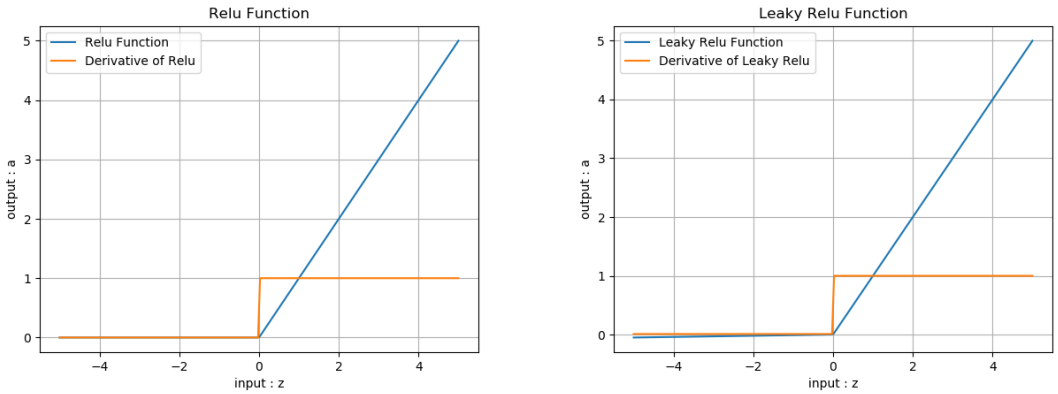
而 tanh 激活函数可以视为 Sigmoid 函数的一个变种，它的表达式和导数公式如下：

$$\tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} = 2\cdot\text{Sigmoid}(2z) - 1 \to a \tag{33}$$

$$\tanh'(z) = 1 - a^2 \tag{34}$$

两个函数对中央区的信号增益较大，对两侧区的信号增益小，在信号的特征空间映射上，有很好的效果；但容易产生梯度消失的问题，使得网络梯度下降时收敛速度慢。

半线性激活函数包括 ReLU 函数、Leaky ReLU 函数等，如下图。



ReLU 函数的表达式和导数公式如下：

$$\mathrm{ReLU}(z) = \max(z, 0) \tag{35}$$

$$\mathrm{ReLU}(z) = \mathbf{1}_{\{z>0\}} \tag{36}$$

它的梯度简单易算，避免了梯度消失的问题；但是在训练过程中容易因为学习率过大等因素，使预测单元落入负半轴，从而不再传播梯度，导致神经元“死亡”。

Leaky ReLU 函数的表达式和导数公式如下：

$$\mathrm{LReLU}(z) = \begin{cases} z & z > 0 \\ \alpha z & z \leq 0 \end{cases} \tag{37}$$

$$\mathrm{LReLU}'(z) = \begin{cases} 1 & z > 0 \\ \alpha & z \leq 0 \end{cases} \tag{38}$$

可见该激活函数在一定程度上缓解了神经元“死亡”的问题。

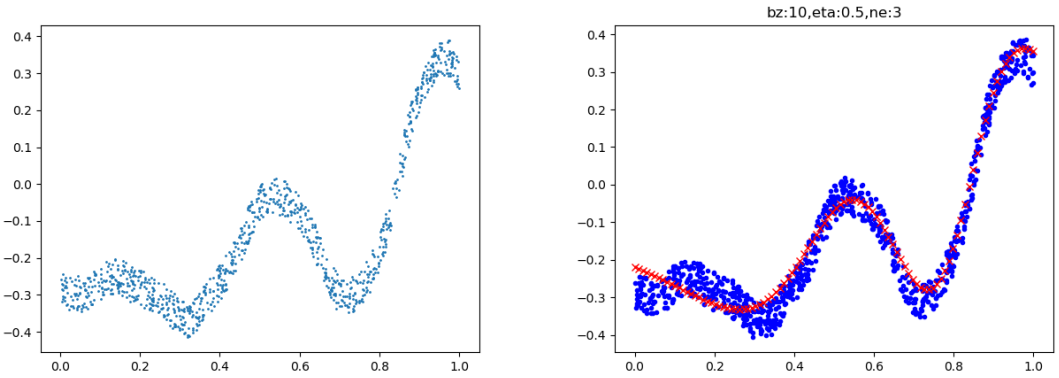
激活函数用于神经网络层与层之间的连接，输出层无需激活函数，下一节将作详细说明。

习题：你认为好的激活函数应具有哪些基本性质？

2.2 双层神经网络实现非线性回归

实际应用场景中，很多数据集 $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ 使用线性模型拟合的效果并不好。

某非线性数据散点图和使用双层网络（隐藏层含 3 个神经元）的拟合模型如下图，很显然，线性模型无法很好地解决这一回归问题。



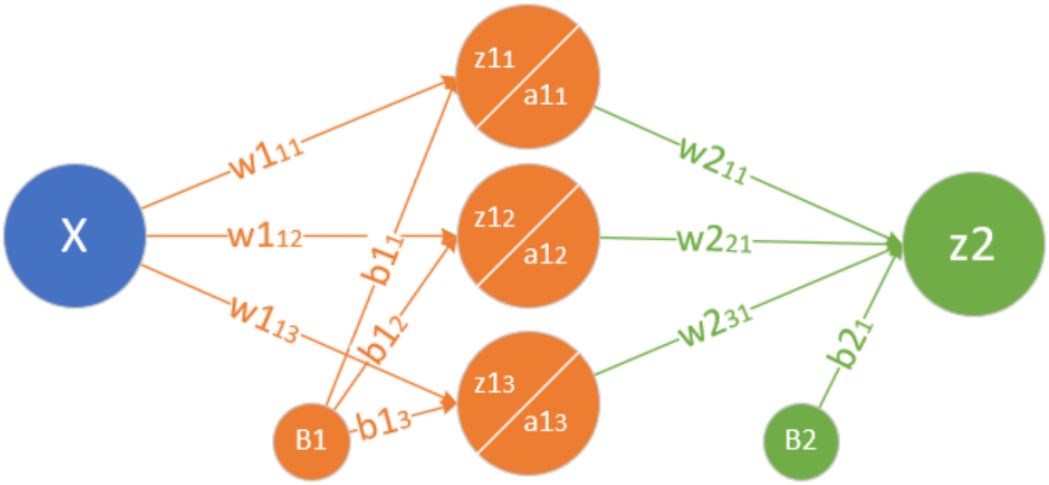
万能近似定理

在引入多层神经网络模型之前，需要先提及一个非常重要的定理——万能近似定理，它是深度学习最根本的理论依据。它证明了在给定网络具有足够多的隐藏单元的条件下，配备一个线性输出层和一个带有任何 “挤压” 性质的激活函数（如 Sigmoid 激活函数）的隐藏层的前馈神经网络，能够以任何想要的误差量近似任何从一个有限维度的空间映射到另一个有限维度空间的 Borel 可测的函数。

简单来说，只要神经网络足够复杂，就可以逼近一切函数。

双层神经网络

对于上面的数据集，可以定义一个两层的神经网络（一个隐藏层，含 3 个神经元），如下图所示。



对多样本情形，写出前向计算公式，即：

$$Z1 = X \cdot W1 + B1 \tag{39}$$

$$A1 = \text{Sigmoid}(Z1) \tag{40}$$

$$Z2 = A1 \cdot W2 + B2 \tag{41}$$

这个模型与上一章介绍的神经网络模型，不同点在于隐藏层 $Z1$ 经过了 Sigmoid 激活函数，各个分量均经过 Sigmoid 激活单元，最终成为 $A1$，亦即式 (40)。其余部分属于线性运算。

该模型的损失函数为

$$J = \frac{1}{2}\sum_{i=1}^{n}(z2_i - y_i)^2 = \frac{1}{2}(Z2 - Y)^\top (Z2 - Y) \tag{42}$$

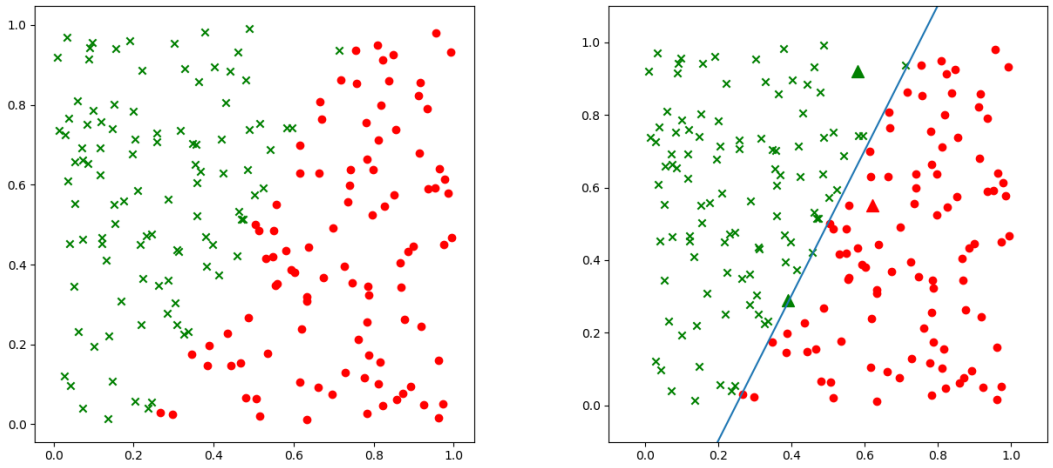
此模型中，待学习的参数有 $W1, B1, W2, B2$；通过梯度下降法更新参数，采用求梯度的链式法则容易获得损失函数 J 对各参数矩阵的梯度，重复更新参数至收敛条件满足即可。值得注意的是，网络隐藏层的数量、各层神经元的数量、学习率的选取等因素均会影响模型训练的效果，此处不详述调参方法。

习题：参照该网络的前向计算公式，请根据链式法则和 Sigmoid 函数的导数公式，计算损失函数对各参数矩阵的梯度。

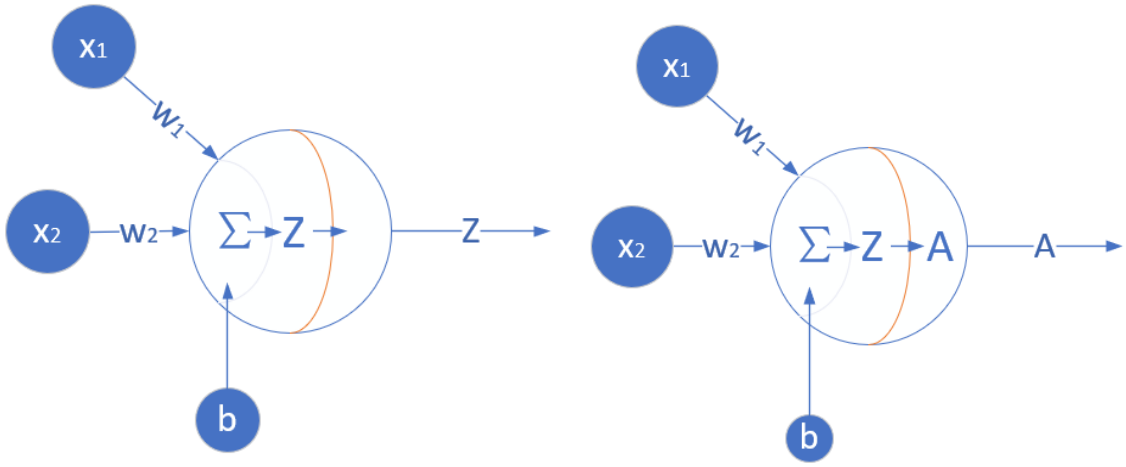
3 线性分类

3.1 线性二分类

从最简单的线性二分类开始：给定训练集 $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$，$y_i \in \{0, 1\}, i = 1, 2, \ldots, n$，要求完成线性二分类任务。教程中的“楚河汉界”训练样本可视化后具有明显的线性边界，如下图。



建立无激活函数的单层神经网络进行分类模型的训练。然而，之前提到的模型在输出层均为连续的实数预测值，而非离散数值；解决这一问题的方法是将线性预测值通过二分类函数，然后计算损失。以下两图展示了线性回归模型与线型分类模型的差别。



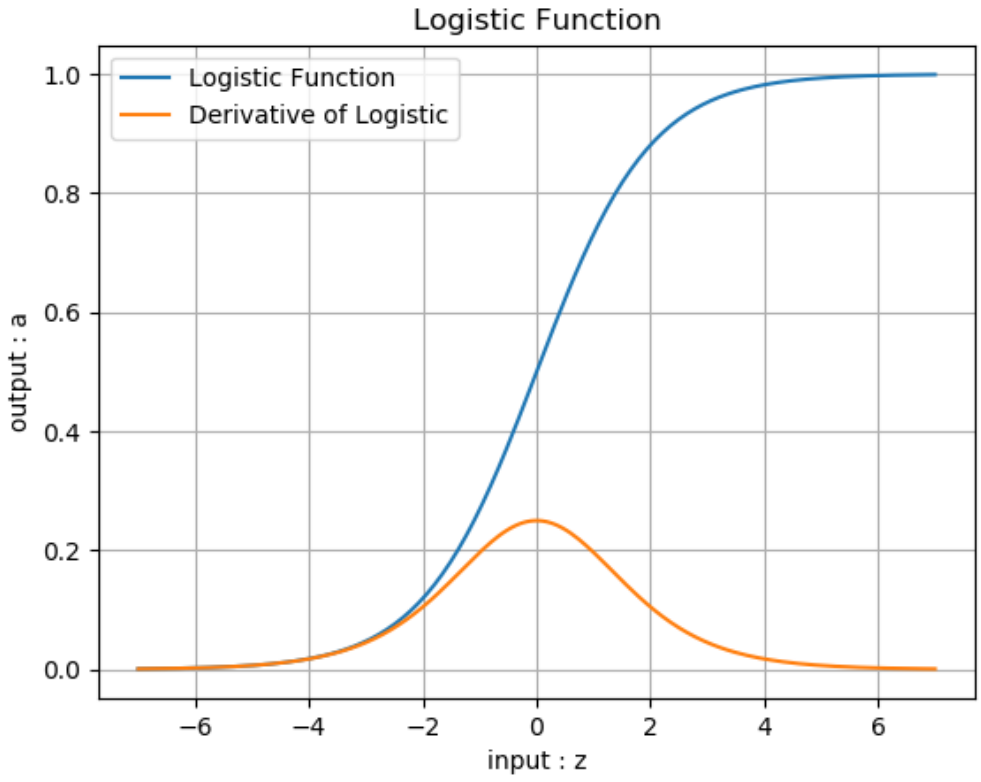
二分类函数

对率函数 Logistic 函数可用于二分类任务，其表达式和导数公式如下：

$$\text{Logistic}(z) = \frac{1}{1+e^{-z}} \to a \tag{43}$$

$$\text{Logistic}'(z) = a(1-a) \tag{44}$$

Logistic 函数的值介于 $(0,1)$ 间，且从图像上看存在一定的对称关系。我们在得到 A 时可以 0.5 为阈值：大于 0.5 归为类 1，小于 0.5 归为类 0。



二分类的反向传播

网络的前向计算公式为：

$$Z = XW + B \tag{45}$$

$$A = \text{Logistic}(Z) \tag{46}$$

对分类任务需选用交叉熵作为损失函数，单样本误差和损失函数表达式为：

$$\text{loss}_i = -[y_i \ln a_i + (1 - y_i) \ln(1 - a_i)] \tag{47}$$

$$J = \sum_{i=1}^{n} \text{loss}_i \tag{48}$$

计算单样本误差对线性预测值的梯度，采用链式法则：

$$\frac{\partial \text{loss}_i}{\partial z_i} = \frac{\partial \text{loss}_i}{\partial a_i} \frac{\partial a_i}{\partial z_i} = -\left[\frac{y_i}{a_i} - \frac{1 - y_i}{1 - a_i}\right] a_i(1 - a_i) = a_i - y_i \tag{49}$$

采取多样本矩阵运算即可得到

$$\frac{\partial J}{\partial Z} = A - Y \tag{50}$$

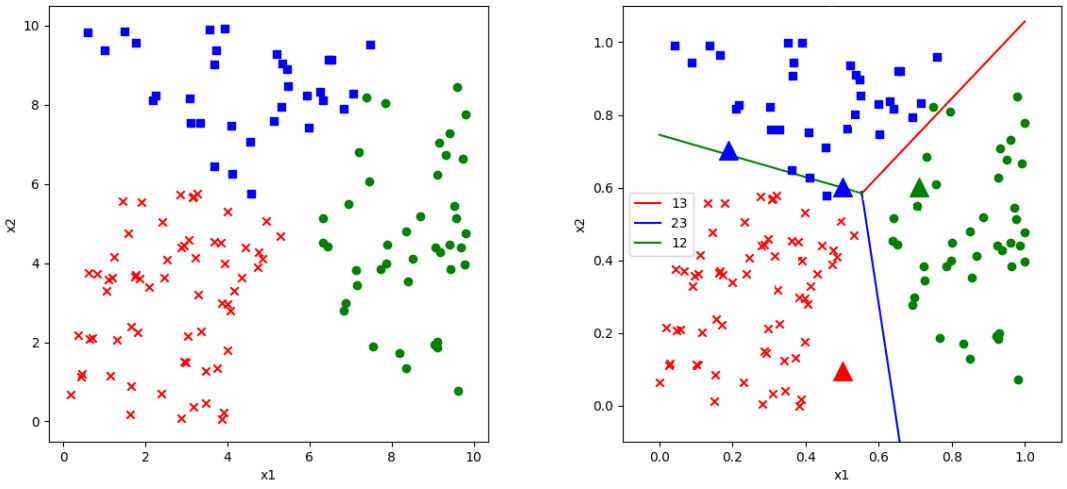
随后对线性部分求梯度的运算可参考第一章的计算部分，结合链式法则即可求得损失函数对各网络权重参数的梯度，执行梯度下降法更新参数即可。

习题：Logistic 二分类函数与 Sigmoid 激活函数的表达式完全一致，那么为何有两个不一样的名称，而非用一个名称统一代替？

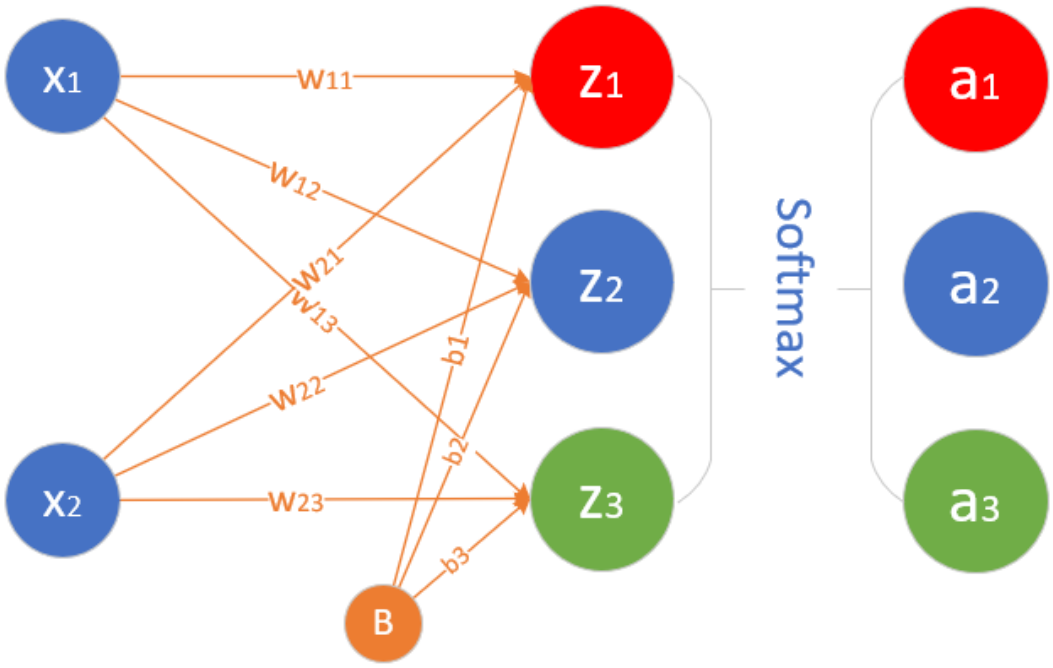
3.2 线性多分类

线性多分类任务的数据集略有不同：给定训练集 $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$，$y_i$ 为长度与类别个数相同的 One-Hot 向量——假若样本属于类别 k，则 y_i 的第 k 个位置上的值为 1，其余位置上的值为 0。

教程中的“三国”训练样本可视化后，两不同类的数据之间具有明显线性边界，如下图。



与上一节一样，单层神经网络足以解决线性分类问题，如下图。但是 Logistic 函数只能作为二分类函数的分类函数，对于多分类问题，常用的分类函数为 Softmax 函数。

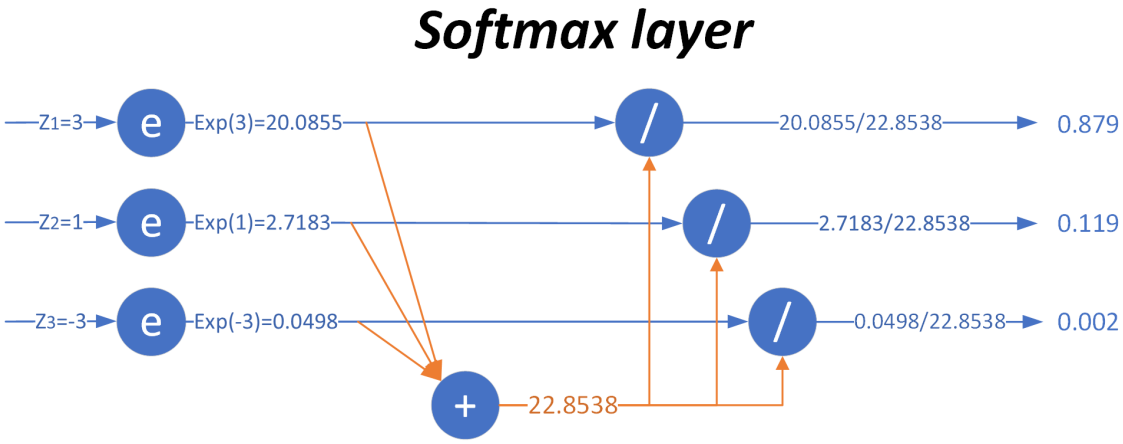


多分类函数

Softmax 函数可用于多分类任务，其表达式如下：

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{c} e^{z_j}} \to a_i \tag{51}$$

可见向量 Z 经过 Softmax 函数成为向量 A 后，其值总和为 1，我们可以将其理解为分数各类的概率。下图例子说明了一个向量通过 Softmax 函数后的变化。



相较于简单的 Max 函数而言，Softmax 函数依旧保留了各原始线性预测值相对大小的信息，且可求导以进行梯度反向传播。

多分类的反向传播

网络的前向计算公式为：

$$Z = XW + B \tag{45}$$

$$A = \text{Softmax}(Z) \tag{52}$$

仍选用交叉熵作为损失函数，单样本误差和损失函数表达式为：

$$\text{loss}_i = -\sum_{j=1}^{c} y_{ij} \ln a_{ij} \tag{53}$$

$$J = \sum_{i=1}^{n} \text{loss}_i \tag{54}$$

计算单样本误差对线性预测值的梯度，采用链式法则：

$$\begin{aligned} \frac{\partial \text{loss}_i}{\partial z_{ij}} &= \frac{\partial \text{loss}_i}{\partial a_{ij}} \frac{\partial a_{ij}}{z_{ij}} + \sum_{k \neq j} \frac{\partial \text{loss}_i}{\partial a_{ik}} \frac{\partial a_{ik}}{z_{ij}} \\ &= -\frac{y_{ij}}{a_{ij}} \cdot a_{ij}(1 - a_{ij}) - \sum_{k \neq j} \frac{y_{ik}}{a_{ik}} \cdot (-a_{ij} a_{ik}) \\ &= a_{ij} \left(\sum_{j=1}^{c} y_{ij} \right) - y_{ij} = a_{ij} - y_{ij} \end{aligned} \tag{55}$$

于是就有了以下的相似结果

$$\frac{\partial J}{\partial Z} = A - Y \tag{56}$$

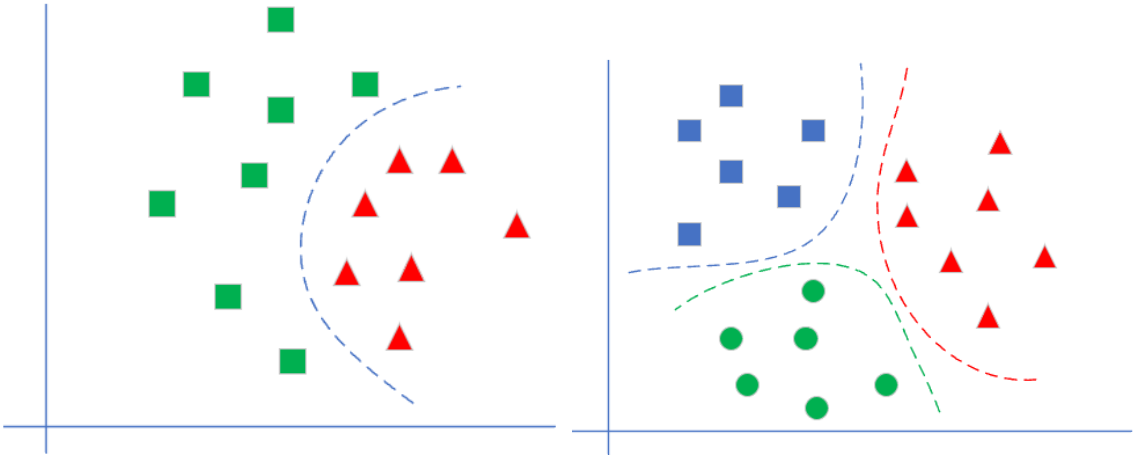
随后对线性部分求梯度的运算可参考第一章的计算部分，结合链式法则即可求得损失函数对各网络权重参数的梯度，执行梯度下降法更新参数即可。

习题：请读者自行验证式 (55) 和式 (56) 的正确性。

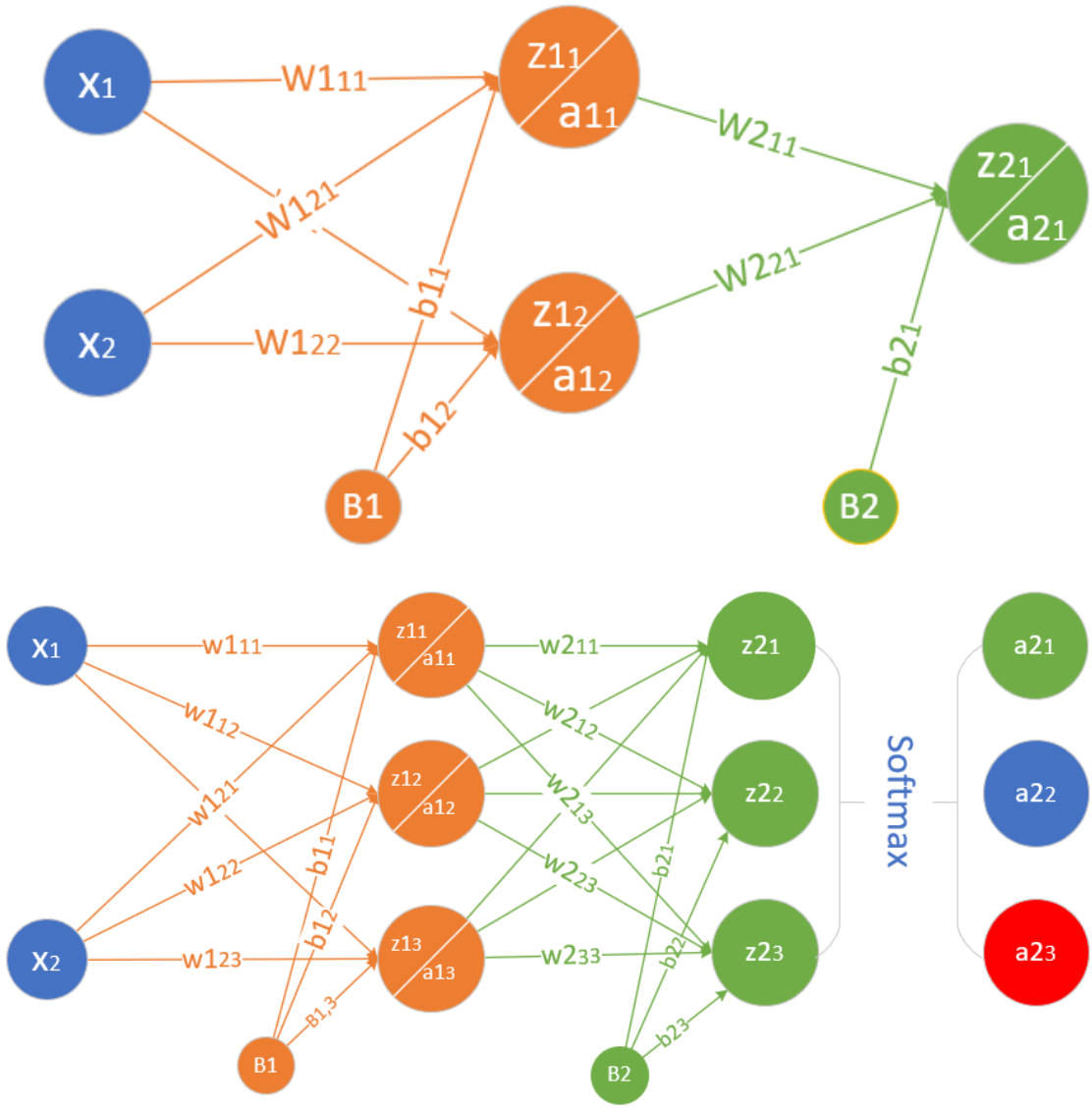
4 非线性分类

4.1 多层神经网络完成分类任务

下图中的两个数据散点图，分别对应线性二分类任务和线性多分类任务：



根据非线性回归模型和线性分类模型中学到的经验，自然的想法是建立多层神经网络，中间隐藏层含有激活函数，输出层需经过分类函数，执行梯度下降训练网络参数即可。



习题：写出以上两个非线性分类任务的前向计算和反向传播公式。

4.2 分类任务的一些其他问题

对于分类任务，除了损失函数之外，还有一些其它的更为直观的评价指标；此外，分类任务有时会出现训练样本不平衡的问题。

评价指标

评价分类模型的指标有很多，基本的几个指标除了预测准确率之外，还有查准率、查全率和综合两指标的 F1-Score。

以二分类问题为例，将样本分为以下四类：

（1）TP：预测为正类，结果预测正确的样本数；
（2）TN：预测为负类，结果预测正确的样本数；
（3）FP：预测为正类，结果预测错误的样本数；
（4）FN：预测为负类，结果预测错误的样本数。
则上述几个评价指标的公式表示如下：

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{57}$$

$$\text{Presicion} = \frac{TP}{TP + FP} \tag{58}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{59}$$

$$\text{F1-Score} = \frac{1}{\frac{1}{\text{Presicion}} + \frac{1}{\text{Recall}}} \tag{60}$$

这些评价指标均可扩展到多分类问题的评价中。

更多的评价指标，诸如 ROC 曲线和 AUC 等，此处不再赘述。

训练样本不平衡问题

在分类任务中，经常遇到训练样本不平衡的问题。类别不均衡问题是现实中很常见的问题，大部分分类任务中，各类别下的数据个数基本上不可能完全相等，但是一点点差异不会产生任何影响与问题。但如果差距过于悬殊，就会使模型的预测效果大打折扣。

一个极端的例子是，在疾病预测时，有 98 个正例，2 个反例，那么分类器只要将所有样本预测为正类，就可以得到 98% 的准确度，但是我们并不会认为这是一个良好的分类器。

平衡数据集

最容易想到的方法就是平衡训练数据集，先想办法扩充样本数量少的类别的数据。有一句话叫做“更多的数据往往战胜更好的算法”。常用的经验法则有：

（1）考虑对大类下的样本（超过 1 万、十万甚至更多）进行欠采样，即删除部分样本；
（2）考虑对小类下的样本（不足 1 万甚至更少）进行过采样，即添加部分样本的副本；
（3）考虑尝试随机采样与非随机采样两种采样方法；
（4）考虑对各类别尝试不同的采样比例，有时候 1:1 的分类反而不好，因为与现实情况相差甚远；
（5）考虑同时使用过采样（over-sampling）与欠采样（under-sampling）。
更多的方法论，包括尝试其它评价指标、修改现有算法和集成学习等，此处不再赘述。

– THE END –