附录

激活函数

在机器学习（ML）中，激活函数是神经网络中非常重要的一部分，它们用于在神经元之间引入非线性特性，从而使网络能够学习和模拟复杂的非线性关系。以下是一些常见的激活函数及其适用场景：

1. Sigmoid函数

- **函数表达式**：$f(x) = \frac{1}{1+e^{-x}}$
- **适用场景**：Sigmoid函数常用于二分类问题的输出层，因为它可以将任意实值压缩到(0,1)区间内，表示概率。然而，由于其梯度消失问题（当输入值非常大或非常小时，梯度接近0），它并不适合在深度神经网络的隐藏层中使用。

2. 双曲正切（Tanh）函数

- **函数表达式**：$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
- **适用场景**：Tanh函数是Sigmoid函数的改进版，它将输出值压缩到(-1,1)区间内，并且以0为中心。这使得Tanh函数在某些情况下（如输入数据以0为中心时）比Sigmoid函数表现更好。然而，它同样存在梯度消失问题，因此也不适合在深度神经网络的隐藏层中广泛使用。

3. 线性整流单元（ReLU）

- **函数表达式**：$f(x) = max(0,x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$
- **适用场景**：ReLU函数是目前深度学习中最常用的激活函数之一。它对于正输入值保持原样，对于负输入值则输出0。这种特性使得ReLU函数在训练过程中能够更快地收敛，并且在一定程度上缓解了梯度消失问题。ReLU函数还促进了神经网络的稀疏性，即许多神经元在训练过程中会变为0，这有助于提升模型的泛化能力。然而，ReLU函数也存在“死亡神经元”问题，即某些神经元在训练过程中可能永远不会被激活。

4. 泄漏线性整流单元（Leaky ReLU）

- **函数表达式**：$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}$
  其中 α 是一个很小的常数（通常取 0.01）。
- **适用场景**：Leaky ReLU是ReLU的一个变体，它解决了ReLU函数中的“死亡神经元”问题。通过允许负输入值有一个小的梯度（由α控制），Leaky ReLU使得神经元在训练过程中更有可能被激活。这使得Leaky ReLU在某些情况下比ReLU表现更好。

5. 参数化线性整流单元（PReLU）

- **函数表达式**：$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}$
  与Leaky ReLU的核心区别在于：α是可学习的参数，而非人工设定的固定常数，会随模型训练自动优化。
- **适用场景**：PReLU进一步扩展了Leaky ReLU的概念，允许α在训练过程中被学习。这使得PReLU能够自适应地调整其行为，以适应不同的数据集和任务。因此，PReLU在某些情况下可能比Leaky ReLU表现更好。

6. 指数线性单元（ELU）

- **函数表达式**：$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha\left(e^{x}-1\right), & x < 0 \end{cases}$
  其中 α 是一个大于0的超参数，用于控制负区间输出的饱和程度。
- **适用场景**：ELU函数结合了ReLU和Sigmoid函数的优点，它在正输入值上保持线性，类似于ReLU；在负输入值上则有一个平滑的过渡，类似于Sigmoid函数。这使得ELU函数能够缓解梯度消失问题，并且在一定程度上减少了神经元的死亡。然而，ELU函数的计算复杂度略高于ReLU和Leaky ReLU。

7. 高斯误差线性单元（GELU）

- **函数表达式**：$f(x) = x \cdot \Phi(x)$
- 其中$\Phi(x)$是高斯函数，表达式为：

$\Phi(x) = \frac{1}{\sqrt{2pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt$

在工程实现中，常使用**近似公式**简化计算：

$$f(x) \approx 0.5x \left[1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right]$$

- **适用场景**：GELU函数是一种非线性激活函数，它结合了ReLU和Dropout的优点。GELU函数对于正输入值有一个平滑的过渡，类似于ReLU；同时，它还能够通过调整输入值的分布来减少过拟合的风险。这使得GELU函数在某些深度学习模型中表现良好。

8. Softmax函数

- **函数表达式**：$f(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (i = 1, 2, \ldots, K)$
- **适用场景**：Softmax函数通常用于多分类问题的输出层，它将神经网络的输出转换为概率分布。Softmax函数将每个输出值转换为正数，并且所有输出值的和为1，这使得Softmax函数非常适合于表示概率分布。

需要注意的是，以上列举的激活函数并不是全部，实际上还有很多其他的激活函数（如Maxout、Swish等）也被用于不同的机器学习场景中。在选择激活函数时，需要具体问题具体分析。

常见术语：数学部分

标量，向量，矩阵和张量的关系

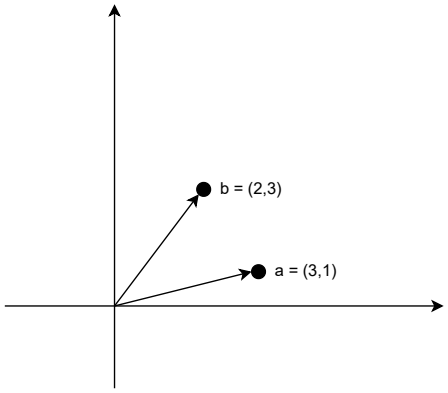
- 标量(scalar): 就是**0维**张量，代码里里用**变量**表示
- 向量(vertor): 就是**1维**张量，代码里里用**一维数组**表示
- 矩阵(matrix): 就是**2维**张量，代码里里用**二维数组**表示

几何向量

向量拥有大小和方向。

几何向量的表示法

- 用几何方法来表示向量，如下图这样用箭头来表示的二位向量。



- 用纵向排列方式表示向量，这样的向量被称为列向量。

$$\vec{a} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \vec{b} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

向量的四则运算

$$\vec{a} + \vec{b} = \begin{bmatrix} 3 \\ 1 \end{bmatrix} + \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 3+2 \\ 1+3 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$$

$$\vec{a} - \vec{b} = \begin{bmatrix} 3 \\ 1 \end{bmatrix} + \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 3-2 \\ 1-3 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$\vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 = 3 \times 2 + 1 \times 3 = 9$
Tips: 向量**点积**（dot product, 也称**内积**）后得到的已经不是向量了，而是一个**标量**，所以点积也称**标量积**。

余弦定理的向量表示法

我们先介绍几个概念，然后在推导出余弦定理。

- $\|a\|$表示向量的**长度**（也可以理解为**距离**）。
  - 假如二维向量$\vec{a} = (a_1, a_2)$，那么 $\|a\| = \sqrt{a_1^2 + a_2^2}$
- 假设向量$\vec{a}$和$\vec{b}$之间的夹角为θ, 如何计算$\vec{a} \cdot \vec{b}$呢?
  - 先做个方向的转换，我们把$\vec{b}$投影到$\vec{a}$上，这样$\vec{b}$在$\vec{a}$方向上的投影就变成了$\left\|b\right\| \cos\theta$
  - $\vec{a}$在自己方向上的投影就是$\|a\|$
  - 这样，$\vec{a} \cdot \vec{b}$ 就等价于 $\|a\|\|b\| \cos\theta$，即$\vec{a} \cdot \vec{b} = \|a\|\|b\| \cos\theta$

简单做一个等式变化，我们就得到了二维向量的余弦定理。

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|a\|\|b\|} = \frac{a_1 b_1 + a_2 b_2}{\sqrt{a_1^2 + a_2^2}\sqrt{b_1^2 + b_2^2}}$$

推广到N维向量空间，就得到公式

$$\vec{a} = (a_1, a_2, ..., a_n); \qquad \vec{b} = (b_1, b_2, ..., b_n)$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|a\|\|b\|} = \frac{a_1 b_1 + a_2 b_2 + ... + a_n b_n}{\sqrt{a_1^2 + a_2^2 + ... + a_n^2}\sqrt{b_1^2 + b_2^2 + ... + b_n^2}}$$

通常，我们用余弦定理来进行相似度的计算。

- 如果两个向量夹角很小，cos值大于0，接近1，说明他们很相似，即**正相关。**
- 如果两个向量夹角是90度，cos值为0，说明他们不相似，是正交的。
- 如果两个向量夹角大于90度，cos值为小于0，说明他们不相似，即**负相关。**

均方误差(MSE:Mean Square Error)

公式： $\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - f_\theta(x^{(i)}))^2$

Accuracy，Precision和Recall

在分类问题中，我们经常需要计算Accuracy的值来评估模型训练的结果。

分类	结果标签-True	结果标签-False
Positive	True Positive(TP)	False Positive(FP)
Negative	False Negative(FN)	True Negative(TN)

- 正确率(Accuracy)

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

- 精确率(Precision)

$$Precision = \frac{TP}{TP + FP}$$

- 召回率(Recall)

$$Recall = \frac{TP}{TP + FN}$$

归一化（Normalization）

归一化是一种常用的数据预处理技术，主要用于消除数据特征之间的量纲和数值范围差异，使得不同特征具有相同的尺度。

归一化的基本思想是将原始数据按比例缩放，使之落入一个小的特定区间。这样做可以使得模型训练更加稳定，收敛更快，同时可以防止模型在训练过程中产生过大或过小的数值。

常见的归一化方法有以下几种：

1. **最大最小归一化**：这种方法将原始数据线性变换到[0,1]区间，计算公式为：

$$x' = \frac{x - min}{max - min}$$

其中x是原始数据，x'是归一化后的数据，min和max分别是数据集中的最小值和最大值。
2. **Z-Score归一化**：这种方法将原始数据变换为均值为0，标准差为1的数据，计算公式为：

$$x' = \frac{x - \mu}{\sigma}$$

其中x是原始数据，x'是归一化后的数据，μ是数据集的均值，σ是数据集的标准差。
3. **单位长度归一化**：这种方法将原始数据变换为单位长度，即每个数据点都在单位球面上。
4. **批归一化**（Batch Normalization）是一种在深度学习中常用的技术，主要用于解决深度神经网络训练过程中的梯度消失和梯度爆炸问题。Batch Normalization的基本思想是对每个小批量（mini-batch）的数据进行归一化处理，使得结果（输出信号各个维度）的均值为0，方差为1。Batch Normalization的计算过程如下：

- 计算均值和方差。
  - 均值公式：

$$\mu = \frac{1}{D}\sum_{i=1}^{D} x_i$$

  - 方差公式：

$$\sigma = \sqrt{\frac{1}{D}\sum_{i=1}^{D}(x_i - \mu)^2}$$

- 进行归一化：通过均值和方差，可以得到归一化后的值, 公式：

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

其中，ε是一个很小很小的数，用于防止分母为0这种情况。

- 线性变换：在Layer Normalization中，我们还需要一组参数来保证归一化操作不会破坏之前的信息。这组参数叫做增益（gain）γ和偏置（bias）β。
  输出公式：

$$y = \gamma\hat{x} + \beta$$

其中γ和β是可学习的参数，可以通过反向传播进行优化的。
5. **层归一化**（Layer Normalization）与Batch Normalization不同，Layer Normalization是在特征维度上进行标准化的，而不是在数据批次维度上。具体的计算过程如下：

- 计算均值和方差。
  - 均值公式：

$$\mu = \frac{1}{D}\sum_{i=1}^{D} x_i$$

  - 方差公式：

$$\sigma = \sqrt{\frac{1}{D}\sum_{i=1}^{D}(x_i - \mu)^2}$$

- 进行归一化：通过均值和方差，可以得到归一化后的值, 公式：

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

其中，ε是一个很小很小的数，用于防止分母为0这种情况。

- 线性变换：在Layer Normalization中，我们还需要一组参数来保证归一化操作不会破坏之前的信息。这组参数叫做增益（gain）g和偏置（bias）b（等同于Batch Normalization中的γ和β）。
  输出公式：

$$h = f(\frac{g}{\sqrt{\sigma^2 + \varepsilon}} \odot (x - \mu) + b)$$

其中f是激活函数，$\odot$ 表示Hadamard Product，也就是操作矩阵中对应的元素相乘，因此要求两个相乘矩阵是同型的。

常见术语：机器学习部分

机器学习非常擅长以下三个任务：

- 回归(regression)
- 分类(classification)
- 聚类(clustering)

回归

简单的说，回归就是处理**连续数据**，如**时间序列数据**时使用的技术。
例子：过去几天的股价数据，如下表所示：

日期	股价
昨天	￥1000
2天前	￥1100
3天前	￥1070

从这样的数据中学习它的趋势，求出“明天的股价会变成多少？”的方法就是**回归。**

分类

检查邮件的内容，然后判断它是不是垃圾邮件，就是一个典型的分类问题。

例子：根据邮件内容，以及这封邮件是否属于垃圾邮件这些数据来进行学习。

邮件内容	是否为垃圾邮件
辛苦啦！下个周日我们去玩吧.....	No
加我为好友吧。这里有我的照片哟！http://..	Yes
恭喜您赢得夏威夷旅游大奖...	Yes

在开始学习之前，我们必须像上述这张表这样，先手动标记邮件是否为垃圾邮件。这样打标签的工作，需要人工介入。

聚类

聚类与分类相似，又有不同。
例子：假设在100名学生的学校进行摸底考试，然后根据考试成绩把100名学生分成几组，根据分组结果，我们能得出某组偏重理科、某组偏重文科这样有意义的结论。

学生编号	英语分数	数学分数	语文分数	物理分数
A-1	100	98	89	96

学生编号	英语分数	数学分数	语文分数	物理分数
A-2	77	98	69	98
A-3	99	56	99	61

Tips: 聚类与分类的区别在于数据带不带**标签**.