人工智能基础 作业 5

目录

1 问题 1 2

2 问题 2 2

3 问题 4 4

4 问题 3 5

5 问题 5 5

麻烦助教看看我做的后两个题有没有对，谢谢啦～

1 问题 1

使用线性模型对结果为 k 的概率的对数进行建模，并考虑到 $\sum_{k=1}^{K} P(y^{(i)}=k|x_i)=1$，添加归一化因子。

$$log\big[P(y^{(i)}=k|x_i)\big]=\beta_k x_i - logZ$$

因此

$$P(y^{(i)}=k|x_i)=\frac{e^{\beta_k x_i}}{Z}$$
$$\sum_{k=1}^{K} P(y^{(i)}=k|x_i)=\frac{\sum_{k=1}^{K} e^{\beta_k x_i}}{Z}=1$$
$$Z=\sum_{k=1}^{K} e^{\beta_k x_i}$$

故 softmax 模型为

$$P(y^{(i)}=k|x_i)=\frac{e^{\beta_k x_i}}{\sum_{k=1}^{K} e^{\beta_k x_i}}$$

2 问题 2

(a) 前馈神经网络如图1所示。

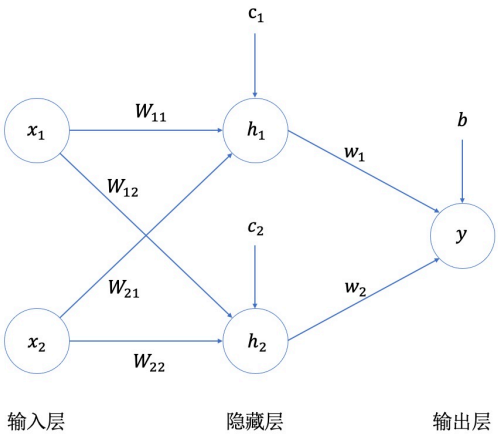

图 1: XOR 前馈神经网络

此时有

$$f(\boldsymbol{x};\boldsymbol{W},\boldsymbol{c},\boldsymbol{w},b)=ReLU\big[\boldsymbol{w}^T ReLU(\boldsymbol{W}^T\boldsymbol{x}+c)+b\big] \tag{1}$$

其中

$$\boldsymbol{W}=\begin{bmatrix}1 & -1\\ 1 & -1\end{bmatrix}, \boldsymbol{c}=\begin{bmatrix}-1\\ 1\end{bmatrix}, \boldsymbol{w}=\begin{bmatrix}1\\ 1\end{bmatrix}, b=0$$

下面验证参数正确性：当 $\boldsymbol{x}=\begin{bmatrix}0\\0\end{bmatrix}$ 时，$\boldsymbol{h}=\begin{bmatrix}0\\1\end{bmatrix}, \boldsymbol{y}=1$; 当 $\boldsymbol{x}=\begin{bmatrix}0\\1\end{bmatrix}$ 时，$\boldsymbol{h}=\begin{bmatrix}0\\0\end{bmatrix}, \boldsymbol{y}=0$;
当 $\boldsymbol{x}=\begin{bmatrix}1\\0\end{bmatrix}$ 时，$\boldsymbol{h}=\begin{bmatrix}0\\0\end{bmatrix}, \boldsymbol{y}=0$; 当 $\boldsymbol{x}=\begin{bmatrix}1\\1\end{bmatrix}$ 时，$\boldsymbol{h}=\begin{bmatrix}1\\0\end{bmatrix}, \boldsymbol{y}=1$。整理成表格如表1所示。

输入层 x_1	输入层 x_2	隐藏层 h_1	隐藏层 h_2	输出层 y
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

表 1: XOR 网络各节点参数计算结果

从表中可以看出所设计的网络对于计算 XOR 结果都是正确的，其中隐含层 h1 所确定的直线可以识别一个半平面，隐含层 h2 所确定的直线可以识别另一个半平面，同时可以看出输出层 y 所确定的直线相当于将隐含层 h1 和隐含层 h2 进行了或运算操作，因此它可识别由隐含层所识别的两个半平面的并集所构成凸多边形。因此，我们得到加入隐含层的神经网络可以让线性不可分的问题变得线性可分。

(b) 证明：若使用线性激活函数，且为二输入问题，则根据式1输出必然可以写成如下形式

$$y=\boldsymbol{w}^T\Big[\boldsymbol{M}\big(\boldsymbol{W}^T\boldsymbol{x}+c\big)+\boldsymbol{N}\Big]+b=\boldsymbol{w'}^T\boldsymbol{x}+b'=w_1'x_1+w_2'x_2+b'$$

即输出 y 是关于 x_1、x_2 的线性组合。而输出的判断依据是找到一个阈值 c(此处 c=0.5)，判断输出是否大于 c。

故问题转化为

$$y=w_1'x_1+w_2'x_2+b'$$

能否满足 XOR 的真值表要求，即 $x_1=0,x_2=0$ 或 $x_1=1,x_2=1$ 时 $w_1'x_1+w_2'x_2+b'<c$，$x_1=0,x_2=1$ 或 $x_1=1,x_2=0$ 时 $w_1'x_1+w_2'x_2+b'>c$。即能否找到一条直线 (如图2所示) 将 x_1 和 x_2 对应取值所在区域划分出来。

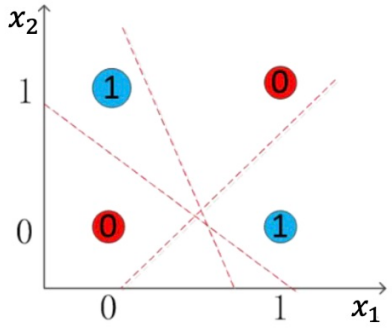

图 2: XOR 前馈神经网络

发现无法做到，故得证。

3 问题 4

(a) 该题的两输入的前馈神经网络如图3所示。

设

$$\boldsymbol{W_1} = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix}, \boldsymbol{W_2} = \begin{bmatrix} w_5 & w_7 \\ w_6 & w_8 \end{bmatrix}, \boldsymbol{b_1} = \begin{bmatrix} b_1 \\ b_1 \end{bmatrix}, \boldsymbol{b_2} = \begin{bmatrix} b_2 \\ b_2 \end{bmatrix}$$

则

$$\boldsymbol{h} = \sigma(z^{[1]}), z^{[1]} = \boldsymbol{W_1 i} + \boldsymbol{b_1}$$
$$\boldsymbol{y} = \sigma(z^{[2]}), z^{[2]} = \boldsymbol{W_2 h} + \boldsymbol{b_2}$$

其中 σ 为 sigmoid 函数，即 $\sigma(z) = \frac{1}{1+e^{-z}}$。

代入 $i = \begin{bmatrix} 0.05 \\ 0.1 \end{bmatrix}$ 解得 $h = \begin{bmatrix} 0.5933 \\ 0.5969 \end{bmatrix}, y = \begin{bmatrix} 0.7401 \\ 0.7729 \end{bmatrix}$。

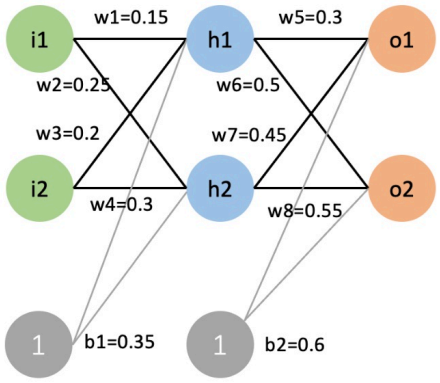

图 3: 两输入的前馈神经网络

(b) 设均方误差 MSE 为 $f = \frac{1}{2}\sum_{i=1}^{2}(y_i - o_i)^2$, 可以得到:

$$f = \frac{1}{2}(y_1 - o_1)^2 + (y_2 - o_2)^2$$
$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \sigma(z_1^{[2]}) \\ \sigma(z_2^{[2]}) \end{bmatrix} = \begin{bmatrix} w_5 h_1 + w_7 h_2 + b_2 \\ w_6 h_1 + w_8 h_2 + b_2 \end{bmatrix}$$

根据反向传播算法可以得到:

$$\frac{\partial f}{\partial w_5} = \frac{\partial f}{\partial y_1}\frac{\partial y_1}{\partial z_1^{[2]}}\frac{\partial z_1^{[2]}}{\partial w_5} = (y_1 - o_1)y_1(1 - y_1)h_1 = 0.0708$$
$$\frac{\partial f}{\partial w_6} = \frac{\partial f}{\partial y_2}\frac{\partial y_2}{\partial z_2^{[2]}}\frac{\partial z_2^{[2]}}{\partial w_6} = (y_2 - o_2)y_2(1 - y_2)h_1 = -0.0184$$

故

$$grad(w_5) = 0.0708, grad(w_6) = -0.0184$$

(c)

$$w_5 = 0.3 - grad(w_5) \cdot 0.1 = 0.2929$$
$$w_6 = 0.5 - grad(w_6) \cdot 0.1 = 0.5018$$

4 问题 3

(a) 模型对于输入的微小改变产生了输出的较大差异，这是因为模型的“曲率”太大，而模型的曲率是由 w 决定的，b 不贡献曲率（对输入进行求导，b 是直接约掉的）。

(b) 采用梯度下降算法，有:

$$\frac{\partial C}{\partial \omega} = \frac{\partial C_0}{\partial \omega} + 2\lambda\omega$$
$$\frac{\partial C}{\partial b} = \frac{\partial C_0}{\partial b}$$

可以发现 L2 正则化项对 b 的更新没有影响，但是对于 ω 的更新有影响:

$$\omega^{(new)} = \omega - \alpha\frac{\partial C}{\partial \omega} = (1 - 2\alpha\lambda)\omega - \alpha\frac{\partial C_0}{\partial \omega}$$

在不使用 L2 正则化时，求导结果中 ω 前系数为 1，现在 ω 前面系数为 $1 - 2\alpha\lambda$，因为 ω, α, λ 都是正数，所以 $(1 - 2\alpha\lambda) < 1$，它的效果是减小 ω，这也就是权重衰减（weight decay）的由来。当然考虑到后面的导数项，$\omega^{(new)}$ 最终的值可能增大也可能减小。

更小的权值 ω，表示网络的复杂度更低，对数据的拟合刚刚好。而正则化是通过约束参数的范数使其不要太大，所以可以在一定程度上减少过拟合情况，故能够提高模型泛化能力。

(c) 首先将原始训练集划分为训练集和验证集，然后通过选择不同的 λ 作为模型进行训练，使用梯度下降法求出不同 λ 对应的最小损失函数

$$C(\lambda, \omega) = \frac{1}{2m}\sum_{i=1}^{m}(h_\omega(x_{cv}^{(i)}) - y_{cv}^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\omega_j^2$$

对应的向量 ω，使用交叉验证集来评价，即测出每个参数 ω 在交叉验证集上的平均的误差平方和

$$C'(\omega) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\omega(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

然后取所有模型中交叉验证集最小的那个模型的 λ 作为最终选择。

5 问题 5

(a) 考虑一个 k 层的神经网络如图4所示。

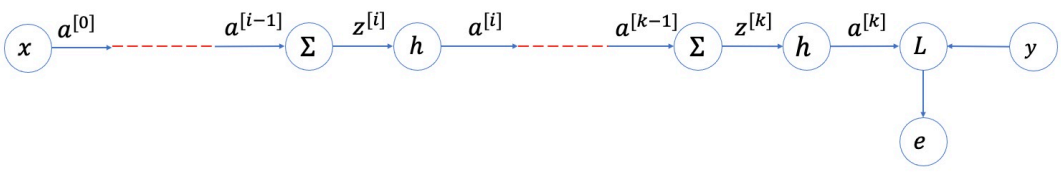

图 4: 多层的神经网络

现在欲计算第 i 层的参数梯度 $W^{[i]}$，有

$$\frac{\partial e}{\partial \boldsymbol{W}^{[i]}} = \frac{\partial e}{\partial \boldsymbol{a}^{[k]}} \frac{\partial \boldsymbol{a}^{[k]}}{\partial \boldsymbol{z}^{[k]}} \frac{\partial \boldsymbol{z}^{[k]}}{\partial \boldsymbol{a}^{[k-1]}} \cdots \frac{\partial \boldsymbol{z}^{[i+1]}}{\partial \boldsymbol{a}^{[i]}} \frac{\partial \boldsymbol{a}^{[i]}}{\partial \boldsymbol{z}^{[i]}} \frac{\partial \boldsymbol{z}^{[i]}}{\partial \boldsymbol{W}^{[i]}}$$

其中

$$\begin{aligned} e &= ||\boldsymbol{a}^{[k]} - \boldsymbol{y}||_2 \\ \boldsymbol{a}^{[k]} &= h(\boldsymbol{z}^{[k]}) \\ \boldsymbol{z}^{[k]} &= \begin{bmatrix} \boldsymbol{b}^{[i]} & \boldsymbol{W}^{[i]} \end{bmatrix} \begin{bmatrix} 1 \\ \boldsymbol{a}^{[i-1]} \end{bmatrix} \end{aligned}$$

故

$$\frac{\partial e}{\partial \boldsymbol{W}^{[i]}} = 2(\boldsymbol{a}^{[k]} - \boldsymbol{y})\ diag\{1 - (\boldsymbol{a}^{[k]})^2\}\ \boldsymbol{W}^{[k]} \cdots\ diag\{1 - (\boldsymbol{a}^{[i]})^2\}\ \ diag\{(\boldsymbol{a}^{[i-1]})^T\}$$

其可能导致梯度消失问题的原因：对激活函数进行求导，发现 $tanh'(x) = 1 - tanh^2(x)$，即此部分小于 1。

神经网络的反向传播是逐层对函数偏导相乘，因此当神经网络层数非常深的时候，最后一层产生的偏差就因为乘了很多的小于 1 的数而越来越小，最终就会变为 0，从而导致层数比较浅的权重没有更新，即求出的梯度更新信息将会以指数形式衰减，也就是发生了梯度消失。解决方案：

(a) 重新设计网络模型

梯度爆炸可以通过重新设计层数更少的网络来解决。使用更小的批尺寸对网络训练也有好处。另外也许是学习率过大导致的问题，减小学习率。

(b) 使用 ReLU 激活函数

梯度爆炸的发生可能是因为激活函数，如之前很流行的 Sigmoid 和 Tanh 函数。使用 ReLU 激活函数可以减少梯度爆炸。采用 ReLU 激活函数是最适合隐藏层的，是目前使用最多的激活函数。

(c) 使用梯度截断（Gradient Clipping）

梯度剪切这个方案主要是针对梯度爆炸提出的，其思想是设置一个梯度剪切阈值，然后更新梯度的时候，如果梯度超过这个阈值，那么就将其强制限制在这个范围之内。这可以防止梯度爆炸。

(d) 使用权重正则化（Weight Regularization）

如果梯度爆炸仍然存在，可以尝试另一种方法，即检查网络权重的大小，并惩罚产生较大权重值的损失函数。该过程被称为权重正则化，通常使用的是 L1 惩罚项（权重绝对值）或 L2 惩罚项（权重平方）。

(e) 批量归一化

Batchnorm 具有加速网络收敛速度，提升训练稳定性的效果，Batchnorm 本质上是解决反向传播过程中的梯度问题。batchnorm 全名是 batch normalization，简称 BN，即批规范化，通过规范化操作将输出信号 x 规范化保证网络的稳定性。

(f) 残差结构

残差网络通过加入 shortcut connections，变得更加容易被优化。这样的结构在反向传播中具有很大的好处，见下式：

$$\frac{\partial loss}{\partial x_l} = \frac{\partial loss}{\partial x_L} \frac{\partial x_L}{\partial x_l} = \frac{\partial loss}{\partial x_L} \left(1 + \frac{\partial}{\partial x_L} \sum_{i=l}^{L-1} F(x_i, W_i)\right) \tag{2}$$

式 (2) 的第一个因子表示的损失函数到达 L 的梯度，小括号中的 1 表明短路机制可以无损地传播梯度，而另外一项残差梯度则需要经过带有 weights 的层，梯度不是直接传递过来的。残差梯度不会那么巧全为-1，而且就算其比较小，有 1 的存在也不会导致梯度消失。所以残差学习会更容易。

(b) 当学习率 α 过大时，根据 $\omega^{(new)} = \omega - \alpha\Delta\omega$，权重参数很可能变为负值，输入网络的正值会和权重相乘后也会变为负值，负值通过 relu 后就会输出 0；如果 ω 在后期有机会被更新为正值也不会出现大问题，但是当 relu 函数输出值为 0 时，relu 的导数也为 0，因此会导致后边 $\Delta\omega$ 一直为 0，进而导致 ω 一直不会被更新，因此会导致这个神经元永久性死亡（一直输出 0）。

解决方法：

(a) 减小学习率

(b) 使用 ELU、PReLU、LeakyRelu 等激活函数代替 Relu

ELU、PReLU、LeakyRelu 都是让 Relu 在自变量小于 0 时函数值也是一个小于 0 的值，从而让 $\omega < 0$ 在后期有机会更新为正值。

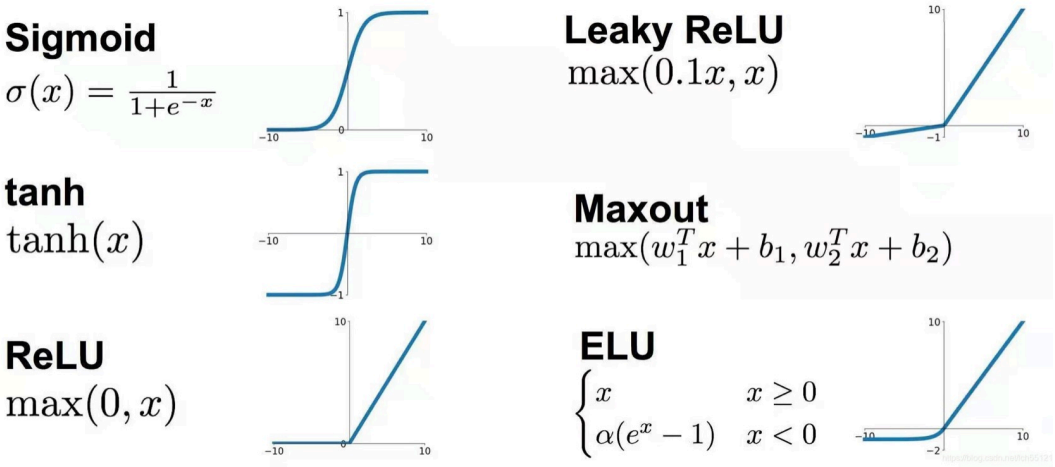

图 5: 各种激活函数

(c)

$$switch'(x) = sigmoid(\beta x) + \beta x sigmoid(\beta x)(1 - sigmoid(\beta x))$$

$$GELU'(x) = \Phi(x) + x f_x(x)$$

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

比较 GELU 与 ReLU 激活函数：如图6所示，GELU 函数在 $x > 0$ 时和 ReLu 函数非常相似，在 $x < 0$ 时函数值也是一个小于 0 的值，且在 0 处有明确的导数值。

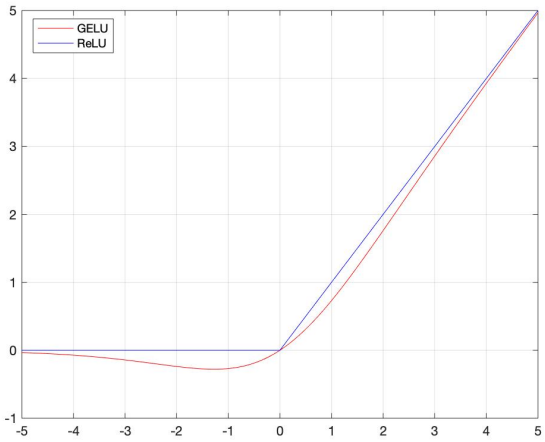

图 6: ReLU 和 GELU 激活函数

GELU 激活函数的优势：在自变量小于 0 时 GELU 函数值也是一个小于 0 的值，从而让 $\omega < 0$ 在后期有机会更新为正值，避免梯度消失。且结合了 ReLU 函数和 drop out 二者的优势，drop out 概率 p 能够随着输入 x 的不同而不同，在 x 较小时以较大概率将其置 0。