神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

覃雄派

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

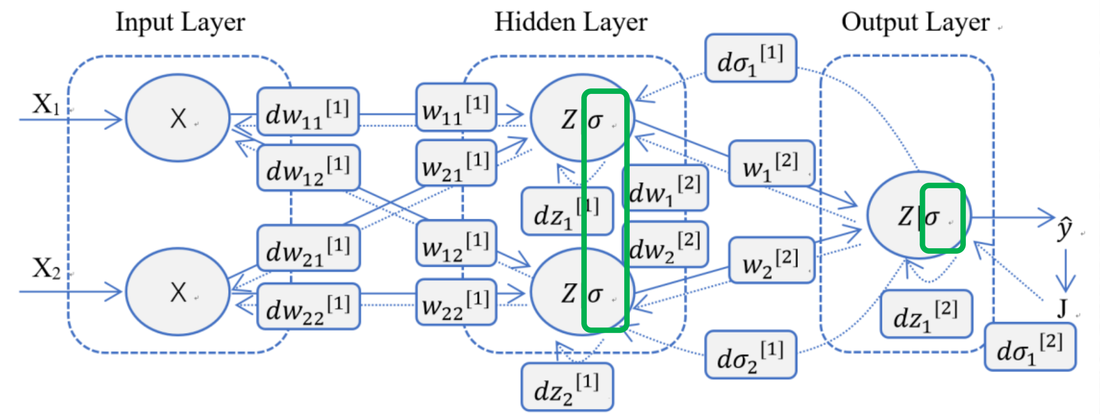
提纲

- 带隐藏层、非线性传导函数的3层MLP
  - 正向传播过程
  - 反向传导过程
- 从3层MLP到多层MLP
- MNIST数据集和样本分析
- 一个针对MNIST数据集二值分类的多层MLP
- 多类别分类的损失函数和反向传播算法
- 回归的损失函数和反向传播算法

要学习本讲，需要先学习“神经网络”上一讲

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

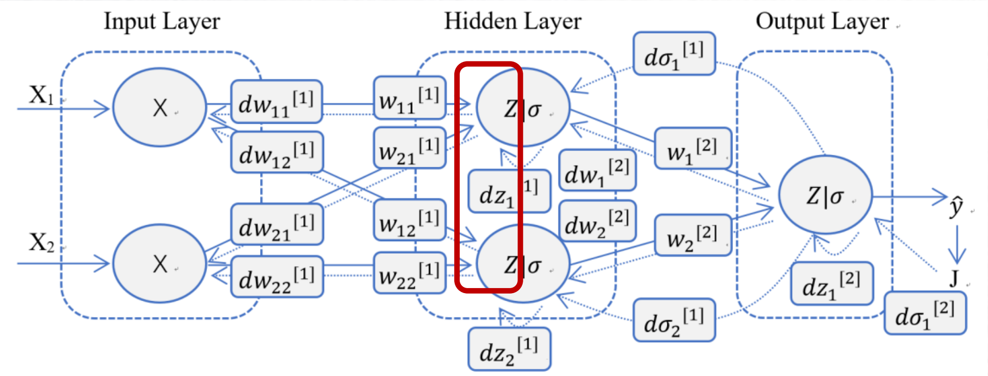
- 带隐藏层、非线性传导函数的3层MLP
  - 下图是一个比上一讲更加复杂的神经网络
  - 复杂性在于隐藏层的节点和输出层的节点都采用sigmoid函数作为激活函数
- 接下来了解其网络结构



采用 sigmoid 激活函数的 3 层神经网络结构

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 带隐藏层、非线性传导函数的3层MLP
  - 下图是一个比上一讲更加复杂的神经网络
  - 复杂性在于隐藏层的节点和输出层的节点都采用sigmoid函数作为激活函数
  - 正向传导过程如下
  - (1) $\boxed{Z^{[1]} = W^{[1]}X + b^{[1]}}$
  - (2) $A^{[1]} = \sigma(Z^{[1]})$
  - (3) $Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$
  - (4) $\hat{y} = A^{[2]} = \sigma(Z^{[2]})$
  - 即$X \rightarrow Z^{[1]} \rightarrow A^{[1]} \rightarrow Z^{[2]} \rightarrow A^{[2]}$
  - 最后用$A^{[2]}$构造损失函数

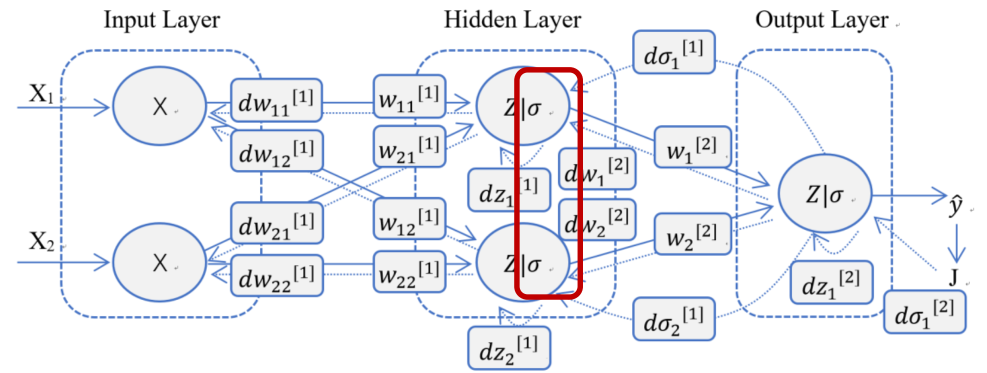


采用 sigmoid 激活函数的 3 层神经网络结构

展开的传导式 $\begin{bmatrix} Z_1^{[1]} \\ Z_2^{[1]} \end{bmatrix} = \begin{bmatrix} W_{11} W_{21} \\ W_{12} W_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \end{bmatrix}$

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 带隐藏层、非线性传导函数的3层MLP
  - 下图是一个比上一讲更加复杂的神经网络
  - 复杂性在于隐藏层的节点和输出层的节点都采用sigmoid函数作为激活函数
  - 正向传导过程如下
  - (1) $Z^{[1]} = W^{[1]}X + b^{[1]}$
  - $\boxed{(2)\ A^{[1]} = \sigma(Z^{[1]})}$
  - (3) $Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$
  - (4) $\hat{y} = A^{[2]} = \sigma(Z^{[2]})$
  - 即$X \rightarrow Z^{[1]} \rightarrow A^{[1]} \rightarrow Z^{[2]} \rightarrow A^{[2]}$
  - 最后用$A^{[2]}$构造损失函数



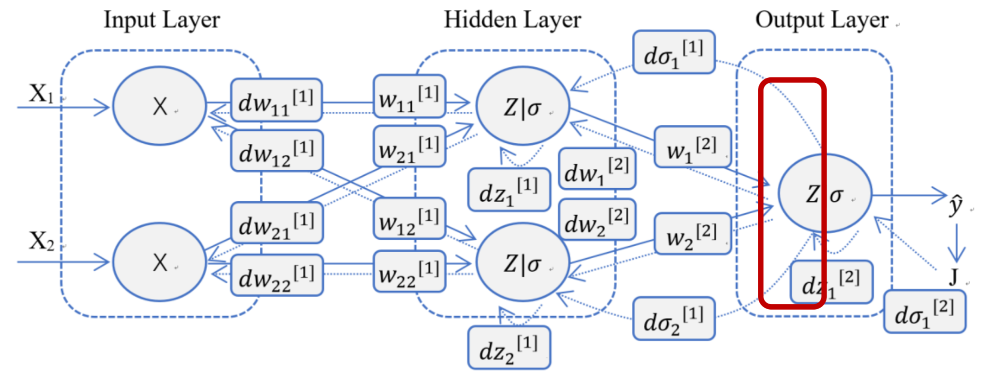
采用 sigmoid 激活函数的 3 层神经网络结构

$$\text{展开的传导式} \begin{bmatrix} A_1^{[1]} \\ A_2^{[1]} \end{bmatrix} = \sigma\left(\begin{bmatrix} Z_1^{[1]} \\ Z_2^{[1]} \end{bmatrix}\right)$$

Element-wise operation

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 带隐藏层、非线性传导函数的3层MLP
  - 下图是一个比上一讲更加复杂的神经网络
  - 复杂性在于隐藏层的节点和输出层的节点都采用sigmoid函数作为激活函数
  - 正向传导过程如下
  - (1) $Z^{[1]} = W^{[1]}X + b^{[1]}$
  - (2) $A^{[1]} = \sigma(Z^{[1]})$
  - (3) $\boxed{Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}}$
  - (4) $\hat{y} = A^{[2]} = \sigma(Z^{[2]})$
  - 即$X \rightarrow Z^{[1]} \rightarrow A^{[1]} \rightarrow Z^{[2]} \rightarrow A^{[2]}$
  - 最后用$A^{[2]}$构造损失函数

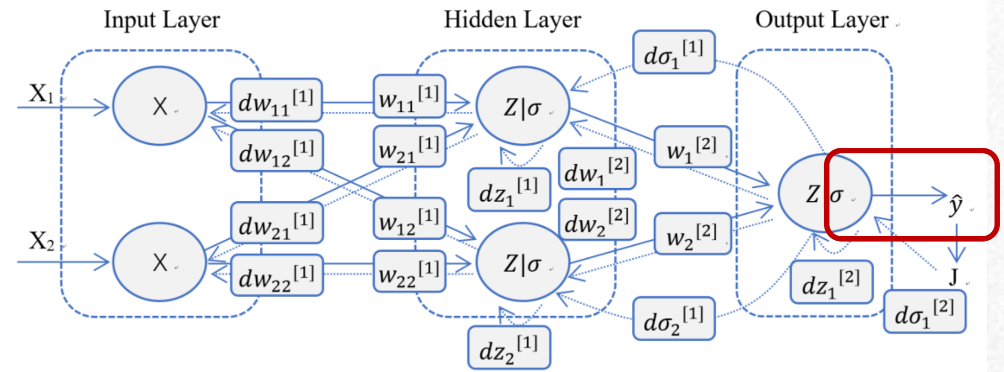


采用 sigmoid 激活函数的 3 层神经网络结构

$$\text{该传导式}[Z^{[2]}] = \begin{bmatrix} w_1^{[2]} & w_2^{[2]} \end{bmatrix} \begin{bmatrix} A_1^{[1]} \\ A_2^{[1]} \end{bmatrix} + \begin{bmatrix} b_1^{[2]} \\ b_2^{[2]} \end{bmatrix}$$

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 带隐藏层、非线性传导函数的3层MLP
  - 下图是一个比上一讲更加复杂的神经网络
  - 复杂性在于隐藏层的节点和输出层的节点都采用sigmoid函数作为激活函数
  - 正向传导过程如下
  - (1) $Z^{[1]} = W^{[1]}X + b^{[1]}$
  - (2) $A^{[1]} = \sigma(Z^{[1]})$
  - (3) $Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$
  - (4) $\boxed{\hat{y} = A^{[2]} = \sigma(Z^{[2]})}$
  - 即$X \rightarrow Z^{[1]} \rightarrow A^{[1]} \rightarrow Z^{[2]} \rightarrow A^{[2]}$
  - 最后用$A^{[2]}$构造损失函数



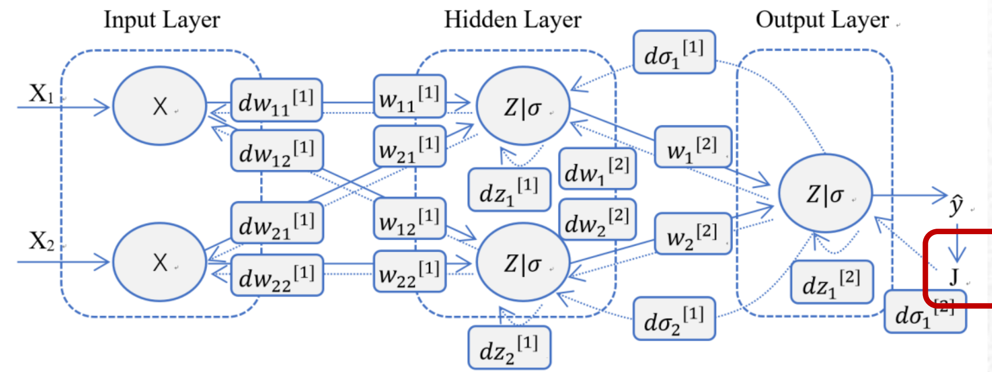
采用 sigmoid 激活函数的 3 层神经网络结构

$$\boxed{\hat{y} = A^{[2]} = \sigma(Z^{[2]})}$$

Element-wise operation

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 带隐藏层、非线性传导函数的3层MLP
  - 下图是一个比上一讲更加复杂的神经网络
  - 复杂性在于隐藏层的节点和输出层的节点都采用sigmoid函数作为激活函数
  - 正向传导过程如下
  - (1) $Z^{[1]} = W^{[1]}X + b^{[1]}$
  - (2) $A^{[1]} = \sigma(Z^{[1]})$
  - (3) $Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$
  - (4) $\hat{y} = A^{[2]} = \sigma(Z^{[2]})$
  - 即$X \rightarrow Z^{[1]} \rightarrow A^{[1]} \rightarrow Z^{[2]} \rightarrow A^{[2]}$
  - 最后用$A^{[2]}$构造损失函数



采用 sigmoid 激活函数的 3 层神经网络结构

二值分类器(0/1)的交叉熵损失函数的形式为

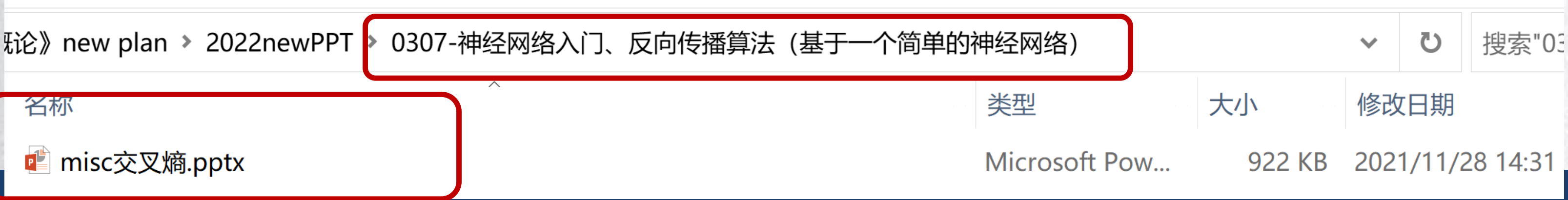
$$J = -\frac{1}{n}((Ylog(A^{[2]}) + (1-Y)\log(1-A^{[2]}))$$

注意$A^{[2]}$即预测值$\hat{y}$

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- **二值分类交叉熵损失函数**
  - $J = -(ylog(\hat{y}) + (1-y)\log(1-\hat{y})) = -ylog(\hat{y}) - (1-y)\log(1-\hat{y})$
  - 对$\hat{y}$求导得到导数
    - $\boxed{-\frac{Y}{\hat{y}} + \frac{1-Y}{1-\hat{y}}}$
  - 在一个样本的情况下$-\frac{Y}{\hat{y}} + \frac{1-Y}{1-\hat{y}}$=-(1*1)/(1*1)+ (1*1)/(1*1) = (1*1)
  - 在n个样本的情况下$-\frac{Y}{\hat{y}} + \frac{1-Y}{1-\hat{y}}$=-(1*n)/(1*n)+ (1*n)/(1*n) = (1*n)

请参考上一节的“交叉熵”补充材料

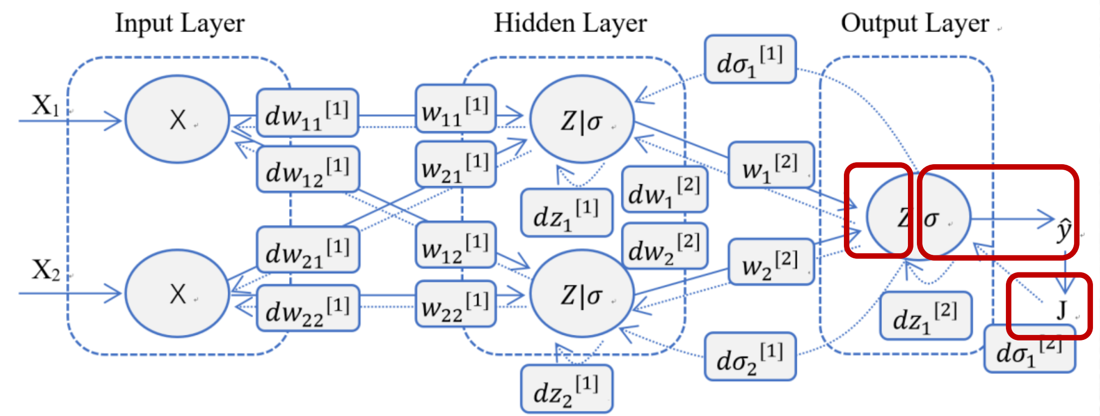


神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 针对该网络结构以及前向传导过程
- 推导反向传播的式子

二值分类器的交叉熵损失函数的形式为$J = -\frac{1}{n}((Ylog(A^{[2]}) + (1-Y)\log(1-A^{[2]}))$
注意$A^{[2]}$即预测值$\hat{y}$



采用 sigmoid 激活函数的 3 层神经网络结构

$$\hat{y} = A^{[2]} = \sigma(Z^{[2]})$$

$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$

第一个部分用到了对数函数的导数公式
第二个部分用到了sigmoid函数的导数公式
第三个部分用到了传导式$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$

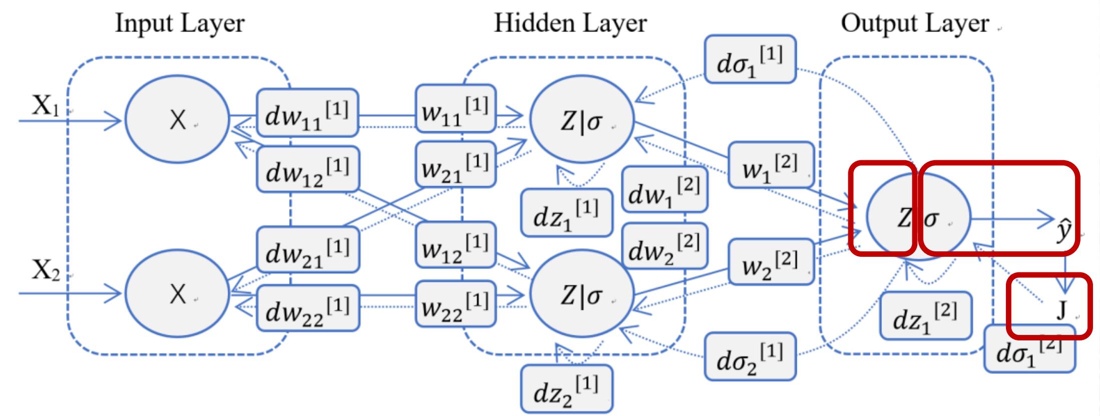
计算$dW^{[2]}$(1/n在这里先省略) $\frac{dJ}{dW^{[2]}} = \frac{dJ}{dA^{[2]}}\frac{dA^{[2]}}{dZ^{[2]}}\frac{dZ^{[2]}}{dW^{[2]}} = \left[-\frac{Y}{A^{[2]}} + \frac{1-Y}{1-A^{[2]}}\right]\left[A^{[2]}(1-A^{[2]})\right]\left[A^{[1]}\right]$

参考第9页的导数计算

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 针对该网络结构以及前向传导过程
- 推导反向传播的式子
  - 对该式子进行约减

$$\frac{dJ}{dW^{[2]}} = \frac{dJ}{dA^{[2]}}\frac{dA^{[2]}}{dZ^{[2]}}\frac{dZ^{[2]}}{dW^{[2]}} = \left[-\frac{Y}{A^{[2]}} + \frac{1-Y}{1-A^{[2]}}\right][A^{[2]}(1-A^{[2]})][A^{[1]}]$$



采用 sigmoid 激活函数的 3 层神经网络结构

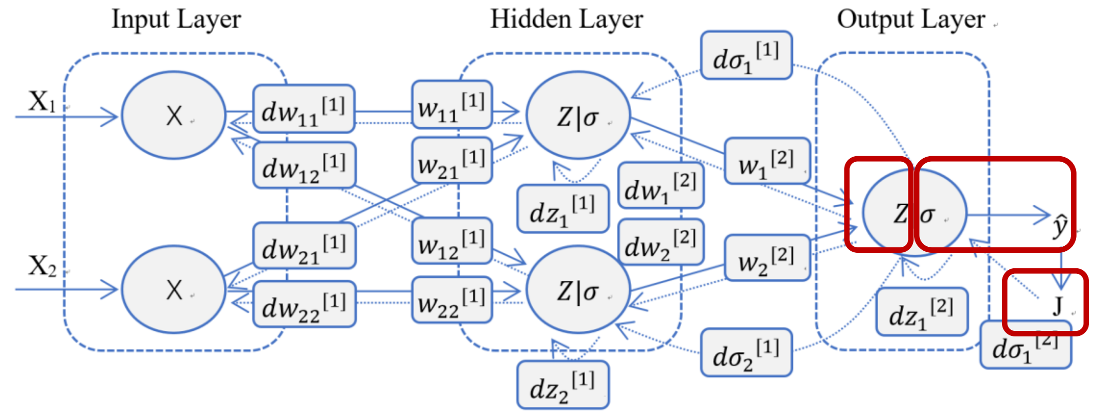
现在，已有

$$\frac{dJ}{dW^{[2]}} = [-Y + YA^{[2]} + A^{[2]} - YA^{[2]}][A^{[1]}] = [A^{[2]} - Y][A^{[1]}] = \mathbf{dZ^{[2]}[A^{[1]}]}$$

因为$dZ^{[2]} = \frac{dJ}{dZ^{[2]}} = \frac{dJ}{dA^{[2]}}\frac{dA^{[2]}}{dZ^{[2]}} = [A^{[2]} - Y]$，这是链式求导的一部分

- 针对该网络结构以及前向传导过程
- 推导反向传播的式子
  - 目标函数对$b^{[2]}$的导数

$$\frac{dJ}{dW^{[2]}}=\frac{dJ}{dA^{[2]}}\frac{dA^{[2]}}{dZ^{[2]}}\frac{dZ^{[2]}}{dW^{[2]}}=\left[-\frac{Y}{A^{[2]}}+\frac{1-Y}{1-A^{[2]}}\right]\left[A^{[2]}(1-A^{[2]})\right]\left[A^{[1]}\right]$$



采用 sigmoid 激活函数的 3 层神经网络结构

现在，已有

$$\frac{dJ}{dW^{[2]}}=\left[-Y+YA^{[2]}+A^{[2]}-YA^{[2]}\right]\left[A^{[1]}\right]=\left[A^{[2]}-Y\right]\left[A^{[1]}\right]=\mathbf{dZ^{[2]}}\left[\boldsymbol{A^{[1]}}\right]$$

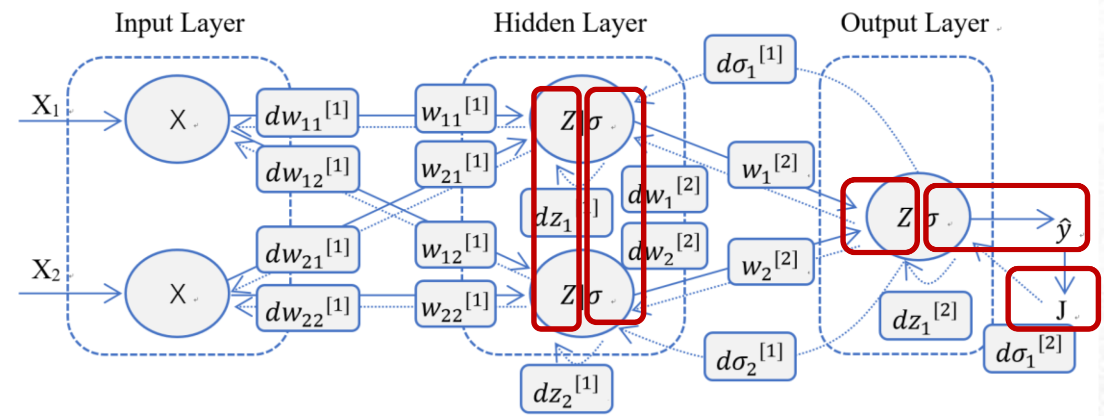
类似地，我们得到

$$\frac{dJ}{db^{[2]}}=\frac{dJ}{dA^{[2]}}\frac{dA^{[2]}}{dZ^{[2]}}\frac{dZ^{[2]}}{db^{[2]}}=\left[A^{[2]}-Y\right][1]=\left[A^{[2]}-Y\right]=\mathbf{dZ^{[2]}}$$

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 针对该网络结构以及前向传导过程
  - 继续反向传播式的推导
  - 继续计算$\frac{dJ}{\mathrm{d}W^{[1]}}$和$\frac{dJ}{\mathrm{d}b^{[1]}}$



采用 sigmoid 激活函数的 3 层神经网络结构

参考$X \xrightarrow{W^{[1]}} Z^{[1]} \xrightarrow{\sigma} A^{[1]} \xrightarrow{W^{[2]}} Z^{[2]} \xrightarrow{\sigma} A^{[2]}$

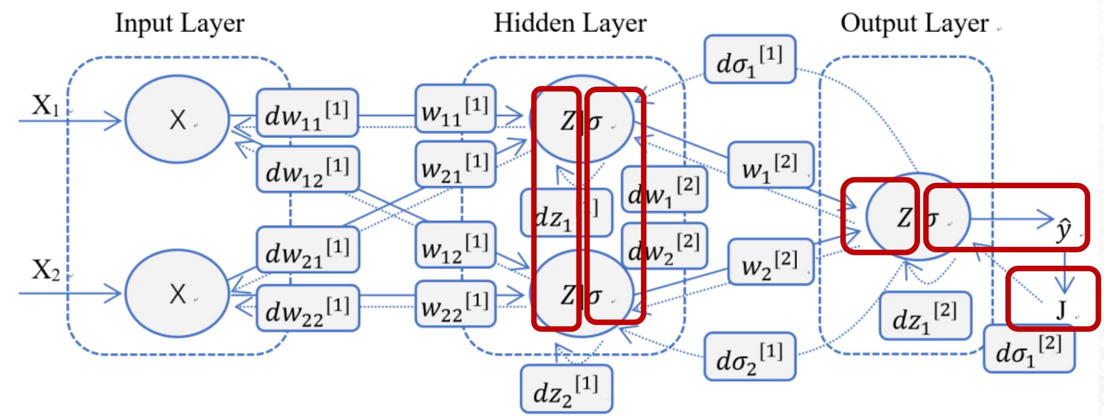
$$\mathrm{d}W^{[1]} = \frac{dJ}{\mathrm{d}W^{[1]}} = \frac{\boldsymbol{dJ}}{\mathbf{dA}^{[2]}}\frac{\boldsymbol{dA}^{[2]}}{\mathbf{dZ}^{[2]}}\frac{\mathrm{d}Z^{[2]}}{dA^{[1]}}\frac{dA^{[1]}}{\mathrm{d}Z^{[1]}}\frac{dZ^{[1]}}{\mathrm{d}W^{[1]}} = \frac{\boldsymbol{dJ}}{\mathbf{dZ}^{[2]}}\frac{dZ^{[2]}}{dA^{[1]}}\frac{dA^{[1]}}{\mathrm{d}Z^{[1]}}\frac{dZ^{[1]}}{\mathrm{d}W^{[1]}} = [\boldsymbol{A}^{[2]} - \boldsymbol{Y}]\, \boldsymbol{W}^{[2]}\mathbf{g}'(\mathbf{Z}^{[1]})A^{[0]} = \mathbf{d}\boldsymbol{Z}^{[1]}\boldsymbol{A}^{[0]}$$

注意$A^{[0]}$即X

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 针对该网络结构以及前向传导过程
  - 继续反向传播式的推导
  - 继续计算$\frac{dJ}{dW^{[1]}}$和$\frac{dJ}{db^{[1]}}$

参考$X \xrightarrow{W^{[1]}} Z^{[1]} \xrightarrow{\sigma} A^{[1]} \xrightarrow{W^{[2]}} Z^{[2]} \xrightarrow{\sigma} A^{[2]}$



采用 sigmoid 激活函数的 3 层神经网络结构

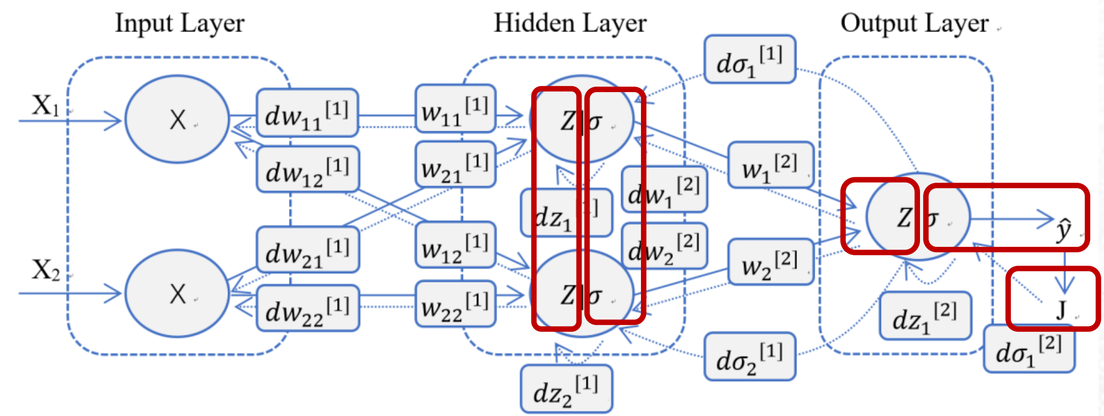
$$dW^{[1]} = \frac{dJ}{dW^{[1]}} = \frac{dJ}{dA^{[2]}}\frac{dA^{[2]}}{dZ^{[2]}}\frac{dZ^{[2]}}{dA^{[1]}}\frac{dA^{[1]}}{dZ^{[1]}}\frac{dZ^{[1]}}{dW^{[1]}} = \frac{dJ}{dZ^{[2]}}\frac{dZ^{[2]}}{dA^{[1]}}\frac{dA^{[1]}}{dZ^{[1]}}\frac{dZ^{[1]}}{dW^{[1]}} = [A^{[2]} - Y]\, W^{[2]} g'(Z^{[1]}) A^{[0]} = dZ^{[1]} A^{[0]}$$

注意$A^{[0]}$即X

- 此处涉及矩阵求导，先按如上形式(标量)理解
- 后续再验证矩阵可以相乘即可

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 针对该网络结构以及前向传导过程
  - 继续反向传播式的推导
  - 继续计算$\frac{dJ}{dW^{[1]}}$和$\frac{dJ}{db^{[1]}}$



采用 sigmoid 激活函数的 3 层神经网络结构

参考$X \xrightarrow{W^{[1]}} Z^{[1]} \xrightarrow{\sigma} A^{[1]} \xrightarrow{W^{[2]}} Z^{[2]} \xrightarrow{\sigma} A^{[2]}$

现在，已有

$$dW^{[1]} = \frac{dJ}{dW^{[1]}} = \frac{dJ}{dA^{[2]}}\frac{dA^{[2]}}{dZ^{[2]}}\frac{dZ^{[2]}}{dA^{[1]}}\frac{dA^{[1]}}{dZ^{[1]}}\frac{dZ^{[1]}}{dW^{[1]}} = \frac{dJ}{dZ^{[2]}}\frac{dZ^{[2]}}{dA^{[1]}}\frac{dA^{[1]}}{dZ^{[1]}}\frac{dZ^{[1]}}{dW^{[1]}} = [A^{[2]} - Y]\, W^{[2]} g'(Z^{[1]}) A^{[0]} = dZ^{[1]} A^{[0]}$$

注意$A^{[0]}$即X

类似地，我们得到

$$\frac{dJ}{db^{[1]}} = \frac{dJ}{dA^{[2]}}\frac{dA^{[2]}}{dZ^{[2]}}\frac{dZ^{[2]}}{dA^{[1]}}\frac{dA^{[1]}}{dZ^{[1]}}\frac{dZ^{[1]}}{db^{[1]}} = \frac{dJ}{dZ^{[2]}}\frac{dZ^{[2]}}{dA^{[1]}}\frac{dA^{[1]}}{dZ^{[1]}}\frac{dZ^{[1]}}{db^{[1]}} = [A^{[2]} - Y] W^{[2]} g'(Z^{[1]})[1] = dZ^{[2]} W^{[2]} g'(Z^{[1]}) = dZ^{[1]}$$

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 观察一下规律性

$$\frac{dJ}{dW^{[2]}} = [-Y + YA^{[2]} + A^{[2]} - YA^{[2]}][A^{[1]}] = [A^{[2]} - Y][A^{[1]}] = \mathbf{dZ^{[2]}[A^{[1]}]}$$

$$\frac{dJ}{db^{[2]}} = \frac{dJ}{dA^{[2]}}\frac{dA^{[2]}}{dZ^{[2]}}\frac{dZ^{[2]}}{db^{[2]}} = [A^{[2]} - Y][1] = [A^{[2]} - Y] = \mathbf{dZ^{[2]}}$$

$$dW^{[1]} = \frac{dJ}{dW^{[1]}} = \frac{\mathbf{dJ}}{\mathbf{dA^{[2]}}}\frac{\mathbf{dA^{[2]}}}{\mathbf{dZ^{[2]}}}\frac{dZ^{[2]}}{dA^{[1]}}\frac{dA^{[1]}}{dZ^{[1]}}\frac{dZ^{[1]}}{dW^{[1]}} = \frac{\mathbf{dJ}}{\mathbf{dZ^{[2]}}}\frac{dZ^{[2]}}{dA^{[1]}}\frac{dA^{[1]}}{dZ^{[1]}}\frac{dZ^{[1]}}{dW^{[1]}} = [\mathbf{A^{[2]}} - \mathbf{Y}]\, \mathbf{W^{[2]}g'(Z^{[1]})}A^{[0]} = \mathbf{dZ^{[1]}A^{[0]}}$$

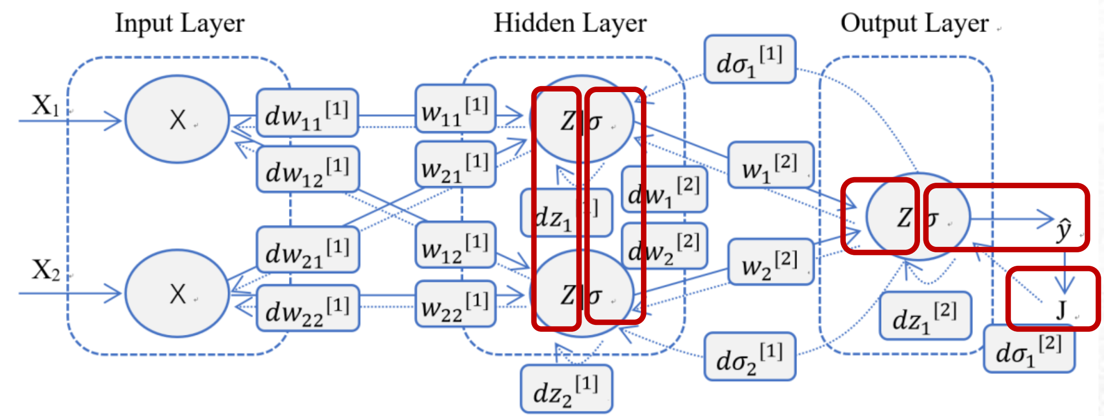
注意$A^{[0]}$即X

$$\frac{dJ}{db^{[1]}} = \frac{\mathbf{dJ}}{\mathbf{dA^{[2]}}}\frac{\mathbf{dA^{[2]}}}{\mathbf{dZ^{[2]}}}\frac{dZ^{[2]}}{dA^{[1]}}\frac{dA^{[1]}}{dZ^{[1]}}\frac{dZ^{[1]}}{db^{[1]}} = \frac{\mathbf{dJ}}{\mathbf{dZ^{[2]}}}\frac{dZ^{[2]}}{dA^{[1]}}\frac{dA^{[1]}}{dZ^{[1]}}\frac{dZ^{[1]}}{db^{[1]}} = [A^{[2]} - Y]W^{[2]}g'(Z^{[1]})[1] = dZ^{[2]}W^{[2]}g'(Z^{[1]}) = \mathbf{dZ^{[1]}}$$

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 针对该网络结构以及前向传导过程
  - 继续反向传播式的推导
  - 准备推导式$\frac{dJ}{dA^{[1]}}$



采用 sigmoid 激活函数的 3 层神经网络结构

参考$X \xrightarrow{W^{[1]}} Z^{[1]} \xrightarrow{\sigma} A^{[1]} \xrightarrow{W^{[2]}} Z^{[2]} \xrightarrow{\sigma} A^{[2]}$

此外，我们准备如下的推导式(后续通用算法用到)

$$\frac{dJ}{dA^{[1]}} = \frac{dJ}{dA^{[2]}} \frac{dA^{[2]}}{dZ^{[2]}} \frac{dZ^{[2]}}{dA^{[1]}} = \frac{dJ}{dZ^{[2]}} W^{[2]} = dZ^{[2]} W^{[2]}$$

求导的链式法则的运用

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 从3层MLP到多层MLP
  - 利用前述总结的规律性

$$\frac{dJ}{\mathrm{dW}^{[2]}} = \mathbf{d}\boldsymbol{Z}^{[2]}[\boldsymbol{A}^{[1]}]$$

$$\frac{dJ}{\mathrm{db}^{[2]}} = \mathbf{d}\boldsymbol{Z}^{[2]}$$

$$\mathrm{dW}^{[1]} = \mathbf{d}\boldsymbol{Z}^{[1]}\boldsymbol{A}^{[0]}$$

注意$A^{[0]}$即X

$$\frac{dJ}{\mathrm{db}^{[1]}} = \mathbf{d}\boldsymbol{Z}^{[1]}$$

此外，我们准备如下的推导式(通用算法用到)

$$\frac{dJ}{\mathrm{d}A^{[1]}} = \mathrm{dZ}^{[2]}W^{[2]}$$

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 从3层MLP到多层MLP
  - 利用前述总结的规律性
  - 推广到L层的神经网络(除了输入层的)，前向传导和反向传播的算法如下

输入	1层	2层	…	L层	代价函数
X 即 $A^{[0]}$	$Z^{[1]} = W^{[1]}A^{[0]} + b^{[1]}$ $A^{[1]} = \sigma(Z^{[1]})$	$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$ $A^{[2]} = \sigma(Z^{[2]})$	…	$Z^{[L]} = W^{[L]}A^{[L-1]} + b^{[L]}$ $A^{[L]} = \sigma(Z^{[L]})$ $A^{[L]}$即$\hat{y}$ 即预测值	$J = -\frac{1}{n}((Y log(A^{[L]}) + (1-Y)\log(1-A^{[L]}))$ 注意$A^{[L]}$即预测值$\hat{y}$ y为实际值，取值为0或者1

- 从3层MLP到多层MLP
  - 利用前述总结的规律性
  - 推广到L层的神经网络(除了输入层的)，前向传导和反向传播的算法如下

.初始化$W^{[1]} \dots W^{[L]}, b^{[1]} \dots b^{[L]}$
.设置$A^{[0]} = X$，L为总的网络层数
.执行如下迭代过程(直到最大迭代次数)
    .前向传导
    .计算代价函数
    .反向传播
    .更新参数

神经网络的反向传播算法总体框架

- 从3层MLP到多层MLP
  - 利用前述总结的规律性
  - 推广到L层的神经网络(除了输入层的)，前向传导和反向传播的算法如下

.初始化$W^{[1]} \dots W^{[L]}, b^{[1]} \dots b^{[L]}$
.设置$A^{[0]} = X$，L为总的网络层数
.执行如下迭代过程(直到最大迭代次数)
  .前向传导
  .计算代价函数
  .反向传播
  .更新参数

For l=1 to L-1
  $Z^{[l]} = \boldsymbol{W}^{[l]} A^{[l-1]} + b^{[l]}$，注意$A^{[0]}$为X
  $\boldsymbol{A}^{[l]} = \sigma(Z^{[l]})$
  保存$Z^{[l]}, W^{[l]}, A^{[l]}$在内存里，备用

最后$Z^{[L]} = W^{[L]} A^{[L-1]} + b^{[L]}$
$A^{[L]} = \sigma(Z^{[L]})$, $A^{[L]}$即$\hat{y}$

- 从3层MLP到多层MLP
  - 利用前述总结的规律性
  - 推广到L层的神经网络(除了输入层的)，前向传导和反向传播的算法如下

.初始化$W^{[1]} \dots W^{[L]}, b^{[1]} \dots b^{[L]}$
.设置$A^{[0]} = X$，L为总的网络层数
.执行如下迭代过程(直到最大迭代次数)
  .前向传导
  .计算代价函数
  .反向传播
  .更新参数

$$J=-\frac{1}{n}((Y log\left(A^{[L]}\right)-(1-Y)\log\left(1-A^{[L]}\right))$$

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

输入	1层	2层	...	L层	代价函数
X 即 $A^{[0]}$	$Z^{[1]} = W^{[1]}A^{[0]} + b^{[1]}$ $A^{[1]} = \sigma(Z^{[1]})$	$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$ $A^{[2]} = \sigma(Z^{[2]})$	...	$Z^{[L]} = W^{[L]}A^{[L-1]} + b^{[L]}$ $A^{[L]} = \sigma(Z^{[L]})$ $A^{[L]}$即$\hat{y}$ 即预测值	$J = -\frac{1}{n}((Y log(A^{[L]}) + (1-Y)\log(1-A^{[L]}))$ 注意$A^{[L]}$即预测值y y为实际值，取值为0或者1

.初始化$W^{[1]} \dots W^{[L]}, b^{[1]} \dots b^{[L]}$

.设置$A^{[0]} = X$，L为总的网络层数

.执行如下迭代过程(直到最大迭代次数)

　　.前向传导

　　.计算代价函数

　　.反向传播

　　.更新参数

$$dA^{[L]} = -\frac{Y}{A^{[L]}} + \frac{1-Y}{1-A^{[L]}}$$

$$dZ^{[L]} = dA^{[L]}g'(Z^{[L]})$$

$$\left.\begin{array}{l} dW^{[L]} = dZ^{[L]}A^{[L-1]} \\ db^{[L]} = dZ^{[L]} \end{array}\right\}$$

$\mathbf{dA^{[L-1]}} = dZ^{[L]}W^{[L]}$(请参考上文推导)

for l=L-1 to 1

$$dZ^{[l]} = \mathbf{dA^{[l]}}g'(Z^{[l]})$$

$$dW^{[l]} = dZ^{[l]}A^{[l-1]}$$

$$db^{[l]} = dZ^{[l]}$$

$$\mathbf{dA^{[l-1]}} = dZ^{[l]}W^{[l]}$$

此外，我们准备如下的推导式(后续通用算法用到)$\frac{dJ}{dA^{[1]}} = dZ^{[2]}W^{[2]}$

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 从3层MLP到多层MLP
  - 利用前述总结的规律性
  - 推广到L层的神经网络(除了输入层的)，前向传导和反向传播的算法如下

.初始化$W^{[1]} \dots W^{[L]}, b^{[1]} \dots b^{[L]}$
.设置$A^{[0]} = X$，L为总的网络层数
.执行如下迭代过程(直到最大迭代次数)
　　.前向传导
　　.计算代价函数
　　.反向传播
　　$\boxed{\text{.更新参数}}$

For l=1 to L
　　$W^{[l]} = W^{[l]} - \eta\, dW^{[l]}$
　　$b^{[l]} = b^{[l]} - \eta\, db^{[l]}$
　　η为学习率

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- MNIST数据集和样本分析
  - 手写数字数据集

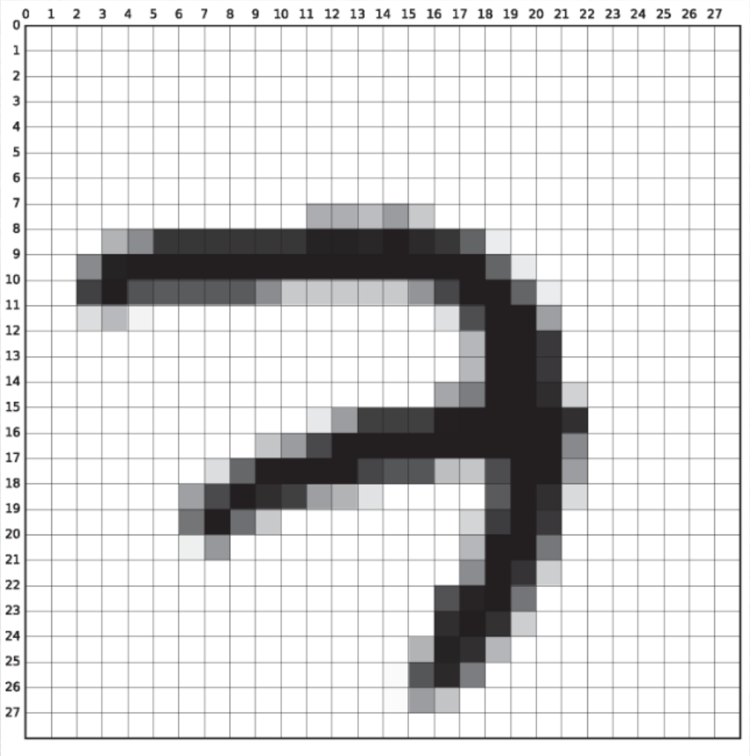
THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- MNIST数据集和样本分析
  - 手写数字数据集
  - 每个样本28×28的黑白图片
  - 压扁则可以表示为
  - 1*784的向量



神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集二值分类的多层MLP
  - MNIST数据集有60000个样本，10个类别
    - 在这里仅仅选取数字5和数字8的样本，共11272个样本
    - 由于只有两个类别，是一个2值分类问题
    - 每个样本是28×28的图片(矩阵)，这些图片经过转化，变成784个分量的一维向量形式

细节请参考如下文档

名称	类型	大小	修改日期
2021-new-反向传播算法-MLP二值分类.docx	Microsoft Word ...	149 KB	2022/2/18 10:38

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集二值分类的多层MLP
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为1个神经元

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集二值分类的多层MLP
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为1个神经元
  - Layer1的前向传导过程具体如下

输入层 784 ⇨ 隐藏层 196 ⇨ 输出层1

X=(11272, 784)

11272个样本，每个样本784维

$W^{[1]}$=(196, 784)

$b^{[1]}$=(196,1)

$A^{[0]} = X^T$=(784, 11272)

$Z^{[1]} = W^{[1]}X + b^{[1]} = W^{[1]}A^{[0]} + b^{[1]}$

=(196,784)×(784, 11272) +(196,1)

=(196, 11272)

$A^{[1]} = \sigma(Z^{[1]})$=(196, 11272)

注意(196, 11272)的每一列都加(196,1)的$b^{[1]}$

- 一个针对MNIST数据集二值分类的多层MLP
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为1个神经元
  - Layer2的前向传导过程具体如下

输入层 784 ⇨ 隐藏层 196 ⇨ 输出层1

$W^{[2]}$=(1, 196)
$b^{[2]}$=(1,1)
$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$
=(1,196)×(196, 11272) +(1,1)
=(1,11727)
$A^{[2]} = \sigma(Z^{[2]})$=(1, 11272)

注意(1,11727)的每一列都加(1,1)的$b^{[2]}$

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集二值分类的多层MLP
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为1个神经元
  - Layer2的反向传播过程具体如下

输入层 784 → 隐藏层 196 → 输出层1 → 损失函数

Y^T=(1,11727)

构造损失函数，计算损失函数对各个参数的导数

$$dA^{[2]} = -\frac{Y^T}{A^{[2]}} + \frac{1-Y^T}{1-A^{[2]}}$$ =(1, 11272)

注意，矩阵的各个位置相除（$A^{[2]}$参考上页）

$dZ^{[2]} = dA^{[2]}g'\left(Z^{[2]}\right)$ =(1, 11272)* (1, 11272) = (1, 11272)

注意，矩阵的各个位置相乘

$dW^{[2]} = dZ^{[2]}(A^{[1]})^T$=(1, 11272)×(11727, 196) = (1, 196)

注意，是矩阵乘法

$db^{[2]} = dZ^{[2]}[1]$ = (1,1)

即(1, 11272)×(11272,1)，相当于每行的各列累加

$dA^{[1]} = (W^{[2]})^T dZ^{[2]}$=(196,1)(1,11272) = (196, 11272)

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集二值分类的多层MLP
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为1个神经元
  - Layer1的反向传播过程具体如下

输入层 784 → 隐藏层 196 → 输出层1

$\mathrm{dZ}^{[1]} = \mathrm{dA}^{[1]} g'(\mathrm{Z}^{[1]})$ = (196, 11272)* (196, 11272) = (196, 11272)　注意，矩阵的各个位置相乘

$\mathrm{dW}^{[1]} = \mathrm{dZ}^{[1]} (A^{[0]})^T$ = (196, 11272)×(11272, 784) = (196*784)　注意，是矩阵乘法

$\mathrm{db}^{[1]} = \mathrm{dZ}^{[1]}[1]$ = (196,1)，即(196, 11272)×(11272,1)　相当于每行的各列累加

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集二值分类的多层MLP
  - 参考实现

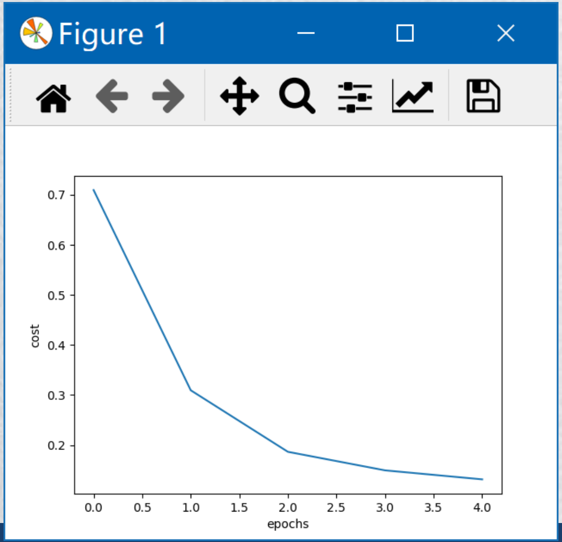
2021-07-18《数据科学概论》new plan › 2022newPPT › 0308-神经网络MLP二分类、多分类、回归（不同损失函数，反向传播算法）

名称	类型	大小	修改日期
01ann_from_scratch.py	Python File	8 KB	2021/11/20 19:54

https://www.adeveloperdiary.com/data-science/machine-learning/understand-and-implement-the-backpropagation-algorithm-from-scratch-in-python

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

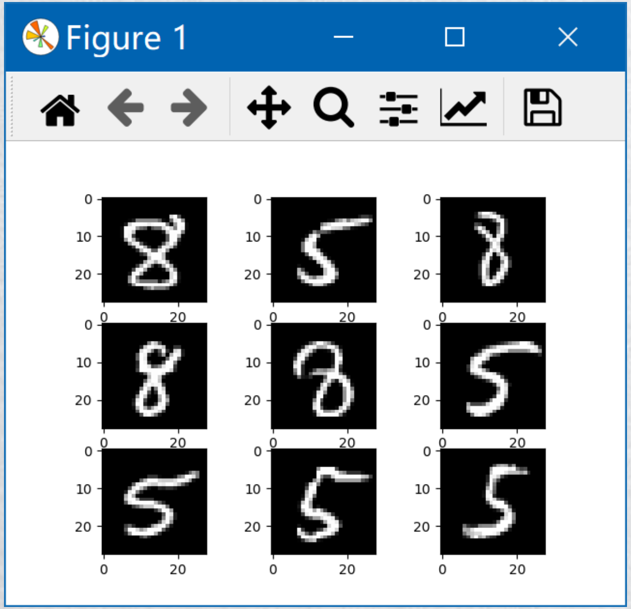
- 一个针对MNIST数据集二值分类的多层MLP
  - 参考实现
  - 损失函数值的变化
  - 训练集、测试集上的准确率




```
cost 0.7094374236626867
cost 0.30940282755807685
cost 0.186483392138912
cost 0.14942029764298495
cost 0.13145182661752475
Accuracy: 0.9532469836763663
Accuracy: 0.956591639871383
```

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集二值分类的多层MLP
  - 参考实现
    - 测试集的8个图片和分类结果




```
nine_y [[1. 0. 1. 1. 1. 0. 0. 0. 0.]]
```

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- Simple MLP的损失函数和反向传导
  - $\mathrm{i} = \begin{bmatrix} \mathrm{i}_1 \\ i_2 \end{bmatrix}$
  - $W_1 = \begin{bmatrix} \mathrm{w}_1 & \mathrm{w}_3 \\ \mathrm{w}_2 & \mathrm{w}_4 \end{bmatrix}$
  - $h = \begin{bmatrix} \mathrm{h}_1 \\ \mathrm{h}_2 \end{bmatrix} = W_1^T \mathrm{I}$
  - $W_2 = \begin{bmatrix} \mathrm{w}_5 \\ \mathrm{w}_6 \end{bmatrix}$
  - $\mathrm{o} = W_2^T \mathrm{h}$
  - $j = \frac{1}{2}(o - y)^2$

输入层2 ⇒ 隐藏层2 ⇒ 输出层1

改写

X=(1, 2) 1个样本，每个样本2维

$\mathrm{A}^{[0]} = X^T = \begin{bmatrix} \mathrm{i}_1 \\ i_2 \end{bmatrix}$=(2, 1)

$\mathrm{W}^{[1]} = \begin{bmatrix} \mathrm{w}_1 & \mathrm{w}_2 \\ \mathrm{w}_3 & \mathrm{w}_4 \end{bmatrix}$=(2, 2)

$\mathrm{Z}^{[1]} = \mathrm{W}^{[1]}\mathrm{X} = \mathrm{W}^{[1]}\mathrm{A}^{[0]} = \begin{bmatrix} \mathrm{w}_1 & \mathrm{w}_2 \\ \mathrm{w}_3 & \mathrm{w}_4 \end{bmatrix} \begin{bmatrix} \mathrm{i}_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} \mathrm{h}_1 \\ \mathrm{h}_2 \end{bmatrix}$=(2,2)×(2, 1)=(2, 1)

$\mathrm{A}^{[1]} = \sigma(\mathrm{Z}^{[1]}) = \begin{bmatrix} \mathrm{h}_1 \\ \mathrm{h}_2 \end{bmatrix}$=(2, 1)，不做非线性变化

$\mathrm{W}^{[2]} = [\mathrm{w}_5 \quad \mathrm{w}_6]$ =(1, 2)

$\mathrm{Z}^{[2]} = \mathrm{W}^{[2]}\mathrm{A}^{[1]} = [\mathrm{w}_5 \quad \mathrm{w}_6] \begin{bmatrix} \mathrm{h}_1 \\ \mathrm{h}_2 \end{bmatrix}$=(1,2)*(2,1) =(1,1)

$\mathrm{A}^{[2]} = \sigma(\mathrm{Z}^{[2]})$=(1, 1)，不做非线性变化

$j = \frac{1}{2}(\mathrm{A}^{[2]} - y)^2$

上一讲内容
转换成矩阵形式

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- Simple MLP的损失函数和反向传导

输入层2 ⇨ 隐藏层2 ⇨ 输出层1

X=(1, 2) 1个样本，每个样本2维

$$A^{[0]} = X^T = \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = (2, 1)$$

$$W^{[1]} = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} = (2,2)$$

$$Z^{[1]} = W^{[1]}X = W^{[1]}A^{[0]} = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}\begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = (2,2)\times(2, 1) = (2, 1)$$

$$A^{[1]} = \sigma(Z^{[1]}) = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}) = (2, 1)$$

$$W^{[2]} = [w_5 \quad w_6]$$

$$Z^{[2]} = W^{[2]}A^{[1]} = [w_5 \quad w_6]\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

$$A^{[2]} = \sigma(Z^{[2]})$$

$$j = \frac{1}{2}(A^{[2]} - y)^{2}$$

正向传导(1)

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- Simple MLP的损失函数和反向传导

输入层2 ⇨ 隐藏层2 ⇨ 输出层1

X=(1, 2) 1个样本，每个样本2维

$$A^{[0]} = X^T = \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}$$

$$W^{[1]} = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}$$

$$Z^{[1]} = W^{[1]}X = W^{[1]}A^{[0]} = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}\begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

$$A^{[1]} = \sigma(Z^{[1]}) = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

$$W^{[2]} = [w_5 \quad w_6] = (1, 2)$$

$$Z^{[2]} = W^{[2]}A^{[1]} = [w_5 \quad w_6]\begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = (1,2)*(2,1) = (1,1)$$

$$A^{[2]} = \sigma(Z^{[2]}) = (1, 1)$$

$$j = \frac{1}{2}(A^{[2]} - y)^{2}$$

正向传导(2)

- **均方误差损失函数**
  - $\frac{1}{2}(\hat{y}-y)^2$
    - Y为实际值，$\hat{y}$为预测值
    - $A^{[2]}$即$\hat{y}$
  - 对$\hat{y}$求导得到
    - $\frac{1}{2}2(\hat{y}-y)=(\hat{y}-y)$
    - 在一个样本的情况下
    - $(\hat{y}-y)$=(1*1) – (1*1) = (1*1)
    - 在n个样本的情况下
    - $(\hat{y}-y)$= (1*n) – (1*n) = (1*n)

损失函数

输入层2 → 隐藏层2 → 输出层1

X=(1, 2) 1个样本，每个样本2维

$$A^{[0]}=X^T=\begin{bmatrix} i_1 \\ i_2 \end{bmatrix}$$

$$W^{[1]}=\begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}$$

$$Z^{[1]}=W^{[1]}X=W^{[1]}A^{[0]}=\begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}\begin{bmatrix} i_1 \\ i_2 \end{bmatrix}=\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

$$A^{[1]}=\sigma(Z^{[1]})=\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

$$W^{[2]}=[w_5 \quad w_6]$$

$$Z^{[2]}=W^{[2]}A^{[1]}=[w_5 \quad w_6]\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$$

$$A^{[2]}=\sigma(Z^{[2]})$$

$$j=\frac{1}{2}(A^{[2]}-y)^2$$

- Simple MLP的损失函数和反向传导

输入层2 ⇒ 隐藏层2 ⇒ 输出层1

X=(1, 2) 1个样本，每个样本2维

$A^{[0]} = X^T = \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}$

$W^{[1]} = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}$

$Z^{[1]} = W^{[1]}X = W^{[1]}A^{[0]} = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}\begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$

$A^{[1]} = \sigma(Z^{[1]}) = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$

$W^{[2]} = [w_5 \quad w_6]$

$Z^{[2]} = W^{[2]}A^{[1]} = [w_5 \quad w_6]\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$

$A^{[2]} = \sigma(Z^{[2]})$

$j = \frac{1}{2}(A^{[2]} - y)^{2}$

反向传导

Y^T=(1,1)

$dA^{[2]} = \frac{1}{2}2(A^{[2]} - Y)$ =(1, 1)

$dZ^{[2]} = dA^{[2]}g'(Z^{[2]})$ =(1, 1)* [1]= (1, 1)，矩阵的各个位置相乘，[1]为一个矩阵

$dW^{[2]} = dZ^{[2]}(A^{[1]})^T = \boxed{(A^{[2]} - Y)[h_1 \quad h_2]}$=(1, 1)×(1, 2) = (1, 2)，注意，是矩阵乘法

$dA^{[1]} = (W^{[2]})^T dZ^{[2]} = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix}(A^{[2]} - Y)$=(2,1)(1,1) = (2, 1)

这个结果和上一讲PPT的结果是一样的

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- Simple MLP的损失函数和反向传导

输入层2 ⇨ 隐藏层2 ⇨ 输出层1

X=(1, 2) 1个样本，每个样本2维

$A^{[0]}=X^T=\begin{bmatrix} i_1 \\ i_2 \end{bmatrix}$

$W^{[1]}=\begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}$

$Z^{[1]}=W^{[1]}X=W^{[1]}A^{[0]}=\begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}\begin{bmatrix} i_1 \\ i_2 \end{bmatrix}=\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$

$A^{[1]}=\sigma(Z^{[1]})=\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$

$W^{[2]}=[w_5 \quad w_6]$

$Z^{[2]}=W^{[2]}A^{[1]}=[w_5 \quad w_6]\begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$

$A^{[2]}=\sigma(Z^{[2]})$

$j=\frac{1}{2}(A^{[2]}-y)^2$

参考$dA^{[1]}=(W^{[2]})^T dZ^{[2]}=\begin{bmatrix} w_5 \\ w_6 \end{bmatrix}(A^{[2]}-Y)$=(2,1)(1,1) = (2, 1)

$dZ^{[1]}=dA^{[1]}g'(Z^{[1]})=\begin{bmatrix} w_5 \\ w_6 \end{bmatrix}(A^{[2]}-Y)*\begin{bmatrix} 1 \\ 1 \end{bmatrix}$=(2, 1)* (2, 1) = (2, 1)，

注意，矩阵的各个位置相乘

$dW^{[1]}=dZ^{[1]}(A^{[0]})^T=\boxed{\begin{bmatrix} \boldsymbol{w_5} \\ \boldsymbol{w_6} \end{bmatrix}(\boldsymbol{A^{[2]}}-\boldsymbol{Y})[\boldsymbol{i_1} \quad \boldsymbol{i_2}]}$=(2, 1)×(1, 2) = (2*2)，

注意，是矩阵乘法

这个结果和上一讲PPT的结果是一样的

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 回归的损失函数和反向传播算法
  - 我们这里采用和前述对MNIST的5和8两类样本进行分类的网络结构
    - 前文进行二值分类
    - 这里进行回归
  - 看看其主要区别在哪里?

- 回归的损失函数和反向传播算法
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为1个神经元

输入层 784 $A^{[0]}$ ⇨ 隐藏层196 $Z^{[1]}$ | $A^{[1]}$ ⇨ 输出层1 $Z^{[2]}$ | $A^{[2]}$

二值分类 ⇨ 回归

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 回归的损失函数和反向传播算法
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为1个神经元
  - Layer1的前向传导过程具体如下

输入层 784 $A^{[0]}$ ⇨ 隐藏层196 $Z^{[1]}$ | $A^{[1]}$ ⇨ 输出层1 $Z^{[2]}$ | $A^{[2]}$

X=(11272, 784) 11272个样本，每个样本784维
$W^{[1]}$=(196, 784)
$b^{[1]}$=(196,1)
$A^{[0]} = X^T$=(784, 11272)
$Z^{[1]} = W^{[1]}X + b^{[1]} = W^{[1]}A^{[0]} + b^{[1]}$
=(196,784)×(784, 11272) +(196,1)，注意(196, 11272)的每一列都加(196,1)的$b^{[1]}$
=(196, 11272)
$A^{[1]} = \sigma(Z^{[1]})$=(196, 11272)

X=(11272, 784) 11272个样本，每个样本784维
$W^{[1]}$=(196, 784)
$b^{[1]}$=(196,1)
$A^{[0]} = X^T$=(784, 11272)
$Z^{[1]} = W^{[1]}X + b^{[1]} = W^{[1]}A^{[0]} + b^{[1]}$
=(196,784)×(784, 11272) +(196,1)，注意(196, 11272)的每一列都加(196,1)的$b^{[1]}$
=(196, 11272)
$A^{[1]} = \sigma(Z^{[1]})$=(196, 11272)

一样

- 一个针对MNIST数据集二值分类的多层MLP
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为1个神经元
  - Layer2的前向传导过程具体如下

输入层 784 ⇨ 隐藏层196 $Z^{[1]}$ | $A^{[1]}$ ⇨ 输出层1 $Z^{[2]}$ | $A^{[2]}$

$W^{[2]}$=(1, 196)
$b^{[2]}$=(1,1)
$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$
=(1,196)×(196, 11272) +(1,1)，注意(1,11727)的每一列都加(1,1)的$b^{[2]}$
=(1,11727)
$A^{[2]} = \sigma(Z^{[2]})$=(1, 11272)

$W^{[2]}$=(1, 196)
$b^{[2]}$=(1,1)
$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$
=(1,196)×(196, 11272) +(1,1)，注意(1,11727)的每一列都加(1,1)的$b^{[2]}$
=(1,11727)
$A^{[2]} = \sigma(Z^{[2]})$=(1, 11272)

一样

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集二值分类的多层MLP
  - 经网络为输入层有784个神经元
  - 层有196个神经元
  - 出层为1个神经元
  - ver2的反向传播过程具体如下

交叉熵损失函数的导数

均方差损失函数的导数

输入层 784 ⇨ 隐藏层196 $Z^{[1]}$ | $A^{[1]}$ ⇨ $Z^{[2]}$ | $A^{[2]}$

Y^T=(1,11727)

$dA^{[2]} = -\frac{Y^T}{A^{[2]}} + \frac{1-Y^T}{1-A^{[2]}}$ =(1, 11272)，注意，矩阵的各个位置相除（$A^{[2]}$参考上页）

$dZ^{[2]} = dA^{[2]} g'(Z^{[2]})$ =(1, 11272)* (1, 11272) = (1, 11272)，注意，矩阵的各个位置相乘

$dW^{[2]} = dZ^{[2]}(A^{[1]})^T$=(1, 11272)×(11727, 196) = (1, 196)，注意，是矩阵乘法

$db^{[2]} = dZ^{[2]}[1]$ = (1,1)，即(1, 11272)×(11272,1)，相当于每行的各列累加

$dA^{[1]} = (W^{[2]})^T dZ^{[2]}$=(196,1)(1,11272) = (196, 11272)

Y^T=(1,11727)

$dA^{[2]} = \frac{1}{2}\, 2(A^{[2]} - Y)$=(1, 11272)

$dZ^{[2]} = dA^{[2]} g'(Z^{[2]})$ =(1, 11272)* (1, 11272) = (1, 11272)，注意，矩阵的各个位置相乘

$dW^{[2]} = dZ^{[2]}(A^{[1]})^T$ =(1, 11272)×(11727, 196) = (1, 196)，注意，是矩阵乘法

$db^{[2]} = dZ^{[2]}[1]$ = (1,1)，即(1, 11272)×(11272,1)，相当于每行的各列累加

$dA^{[1]} = (W^{[2]})^T dZ^{[2]}$=(196,1)(1,11272) = (196, 11272)

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集二值分类的多层MLP
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为1个神经元
  - Layer1的反向传播过程具体如下

输入层 784 → 隐藏层196 $Z^{[1]}$ | $A^{[1]}$ → 输出层1 $Z^{[2]}$ | $A^{[2]}$

$dZ^{[1]} = dA^{[1]} g'(Z^{[1]})$ = (196, 11272)* (196, 11272) = (196, 11272)，注意，矩阵的各个位置相乘
$dW^{[1]} = dZ^{[1]}(A^{[0]})^T$ = (196, 11272)×(11272, 784) = (196*784)，注意，是矩阵乘法
$db^{[1]} = dZ^{[1]}$[1] = (196,1)，即(196, 11272)×(11272,1)，相当于每行的各列累加

$dZ^{[1]} = dA^{[1]} g'(Z^{[1]})$ = (196, 11272)* (196, 11272) = (196, 11272)，注意，矩阵的各个位置点乘
$dW^{[1]} = dZ^{[1]}(A^{[0]})^T$ = (196, 11272)×(11272, 784) = (196*784)，注意，是矩阵乘法
$db^{[1]} = dZ^{[1]}$ [1] = (196,1)，即(196, 11272)×(11272,1)，相当于每行的各列累加

一样

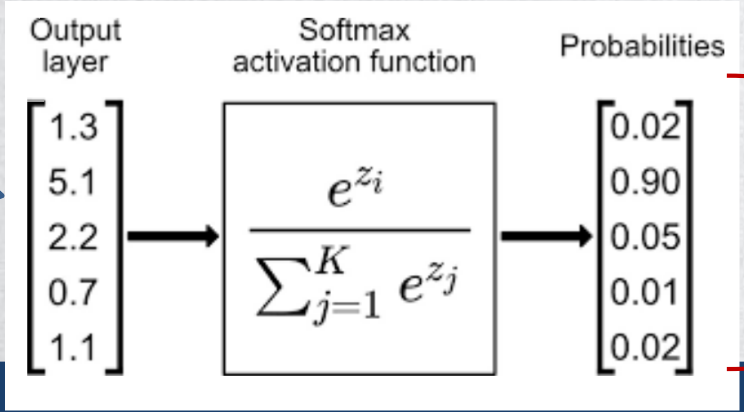
神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 多类别分类的损失函数和反向传播算法
  - 对于MNIST数据集
    - 每个样本是一个图片，总共60 000个样本
    - 样本的类别有10个，即0-9的10个数字
  - 前面的模型只能进行二值分类，只有5和8两种样本
  - 如何实现多类别分类呢?
    - 比如MNIST数据集的10类别分类
    - 这里要用到Softmax函数

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

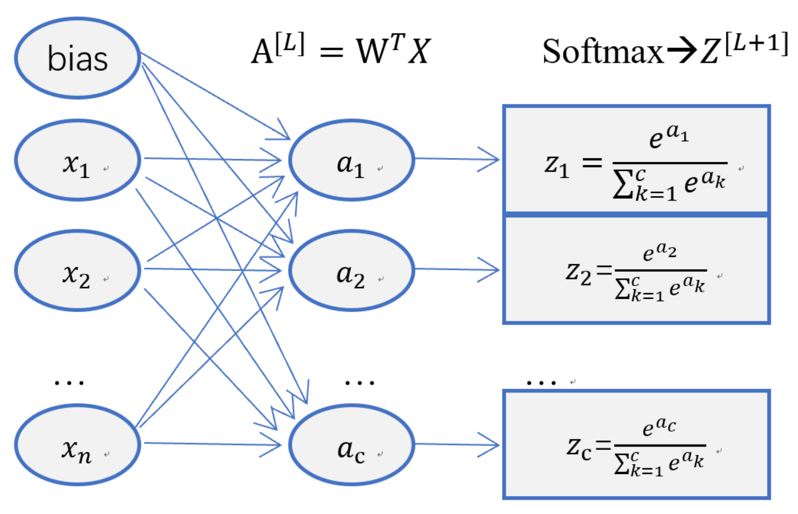
- 多类别分类的损失函数和反向传播算法
  - 对于MNIST数据集
    - 每个样本是一个图片，总共60 000个样本
      - 样本的类别有10个，即0-9的10个数字
  - 在神经网络的最后输出层有10个神经元
    - 我们需要进行Softmax函数转换
    - 把10个神经元的输出，转换成10个类别的概率分布

Softmax
示例



- 这里是5个类别
- 输出层的神经元输出
- 经过Softmax变换，得到5个类别的概率分布

- 多类别分类的损失函数和反向传播算法
  - Softmax函数的具体形式
  - $z_i = \frac{e^{a_i}}{\sum_{k=1}^{c} e^{a_k}}$，这里c为类别的数量，$a_k$为对应各个类别的原始数值
    - 可以看出$\sum_{i=1}^{c} z_i = 1$
  - a_k是神经网络最后一层的激活值



Softmax 函数示意图

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 多类别分类的损失函数和反向传播算法
  - 针对多类别分类，采用交叉熵损失函数，具体形式如下
    - $\hat{y} = Z^{[L+1]}$ =Softmax($A^{[L]}$)
    - Loss = L = $-\sum_{i=1}^{c} y_i log\hat{y}_i = -\sum_{i=1}^{c} y_i log z_i$
    - $\hat{y}_i$为$\hat{y}$的各个分量，有多少个类别就有多少个分量
    - y为实际值，有多少个类别就有多少个分量，其分量取值为0或者1
  - 根据Softmax的导数
    - 有$\frac{dL}{dA^{[L]}} = Z^{[L+1]} - Y = = \hat{y} - y$
  - 在一个样本的情况下$\hat{y} - y$ =(10*1)-(10*1)=(10*1)，假设有10个类别
  - 在n个样本的情况下$\hat{y} - y$ =(10*n)-(10*n)=(10*n)，假设有10个类别

推导请参考如下附加Word文档
损失函数Loss是$Z^{[L+1]}$的函数
现在计算损失函数Loss针对$A^{[L]}$的导数

2021-new-反向传播算法-MLP多类别分类Softmax.docx   Microsoft Wor...   70 KB   2021/11/24 20:55

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 多类别分类的损失函数和反向传播算法
  - 针对多类别分类，采用交叉熵损失函数，具体形式如下
    - $\mathrm{L} = -\sum_{i=1}^{c} y_i log z_i$
  - 根据Softmax的导数
    - 有$\frac{dL}{dA^{[L]}} = Z^{[L+1]} - Y$
  - L-1层到1层的参数的导数的推导
    - 和前文所述是一样的（接下来看实例）

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集10类别分类的多层MLP
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为10个神经元

输入层 784 ⇒ 隐藏层196 $Z^{[1]} \mid A^{[1]}$ ⇒ 输出层10 $Z^{[2]} \mid A^{[2]}$ Softmax $Z^{[3]}$

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集多类别分类的多层MLP
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为10个神经元
  - Layer1的前向传导过程具体如下

输入层 784 → 隐藏层196 $Z^{[1]}$ | $A^{[1]}$ → 输出层10 $Z^{[2]}$ | $A^{[2]}$ | Softmax $Z^{[3]}$

X=(60000, 784) 60000个样本，每个样本784维

$W^{[1]}$=(196, 784)

$b^{[1]}$=(196,1)

$A^{[0]} = X^T$=(784, 60000)

$Z^{[1]} = W^{[1]}X + b^{[1]} = W^{[1]}A^{[0]} + b^{[1]}$

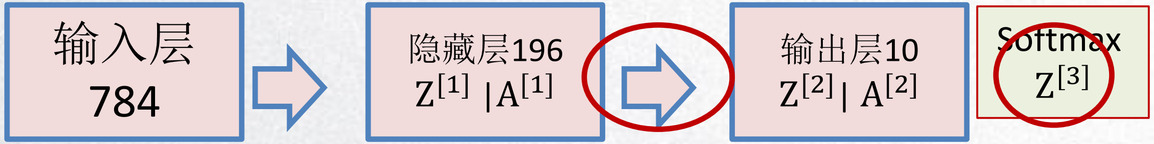
=(196,784)×(784, 60000) +(196,1)，注意(196, 60000)的每一列都加(196,1)的$b^{[1]}$

=(196, 60000)

$A^{[1]} = \sigma(Z^{[1]})$=(196, 60000)

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集多类别分类的多层MLP
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为10个神经元
  - Layer2的前向传导过程具体如下



$W^{[2]}$=(10, 196)
$b^{[2]}$=(10,1)
$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$
=(10,196)×(196, 60000) +(10,1)，注意(10,60000)的每一列都加(10,1)的$b^{[2]}$
=(10,60000)
$A^{[2]} = \sigma(Z^{[2]})$=(10, 60000)

接着进行Softmax转换
$Z^{[3]}$=Softmax($A^{[2]}$)=(10, 60000)

损失函数$L = -\sum_{i=1}^{c} y_i log z_i^{[3]}$

根据Softmax的导数
有$\frac{dL}{dA^{[2]}} = Z^{[3]} - Y$=(10,60000)

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集二值分类的多层MLP
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为10个神经元
  - Layer2的反向传播过程具体如下

输入层 784 ⇨ 隐藏层196 $Z^{[1]}\ |A^{[1]}$ ⇨ 输出层10 $Z^{[2]}|\ A^{[2]}$ | Softmax $Z^{[3]}$

Y^T=(10, 60000)

$dA^{[2]} = \frac{dL}{dA^{[2]}} = Z^{[3]} - Y$=(10, 60000)，注意，矩阵的各个位置相减（$A^{[2]}$参考上页）

$dZ^{[2]} = dA^{[2]}g'(Z^{[2]})$=(10, 60000)* (10, 60000) = (10, 60000)，注意，矩阵的各个位置相乘

$dW^{[2]} = dZ^{[2]}(A^{[1]})^T$=(10, 60000)×(60000, 196) = (10, 196)，注意，是矩阵乘法

$db^{[2]} = dZ^{[2]}$[1] = (10,1)，即(10, 60000)×(60000,1)，相当于每行的各列累加

$dA^{[1]}$ = $(W^{[2]})^T dZ^{[2]}$=(196,10)(10,60000) = (196, 60000)

元素都是1的列向量

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）

- 一个针对MNIST数据集二值分类的多层MLP
  - 神经网络为输入层有784个神经元
  - 隐藏层有196个神经元
  - 输出层为10个神经元
  - Layer1的反向传播过程具体如下

输入层 784 ⇨ 隐藏层196 $Z^{[1]}$ | $A^{[1]}$ ⇨ 输出层10 $Z^{[2]}$ | $A^{[2]}$ | Softmax $Z^{[3]}$

$dZ^{[1]} = dA^{[1]} g'(Z^{[1]})$ = (196, 60000)* (196, 60000) = (196, 60000)，注意，矩阵的各个位置相乘

$dW^{[1]} = dZ^{[1]} (A^{[0]})^T$ = (196, 60000)×(60000, 784) = (196*784)，注意，是矩阵乘法

$db^{[1]} = dZ^{[1]}$[1] = (196,1)，即(196, 60000)×(60000,1)，相当于每行的各列累加

元素都是1的列向量

神经网络MLP二值分类、多类别分类、回归（不同损失函数，反向传播算法）