

深度学习

输出层与代价函数

概览

1. 线性神经元输出层与均方误差代价函数。
2. logistic神经元输出层与均方误差代价函数。
3. 对数似然代价函数。
4. logistic神经元输出层与对数似然代价函数。
5. softmax神经元输出层与交叉熵代价函数。
6. 三种代价函数对比。

1. 线性神经元输出层与均方误差代价函数。

线性神经元输出层

线性神经元激活函数： $y = x$

线性神经元输出层： $h = [a_1^{(L)}, a_2^{(L)}, \dots, a_n^{(L)}]^T$

作用：拟合线性或近似线性分布的数据。

均方误差代价函数

公式：

$$J = \frac{1}{2m} \sum_{i=1}^m \| y^{(i)} - h^{(i)} \|^2$$

以单样本代价函数为例，输出层残差为：

$$\delta_j^{(L)} = \frac{\partial J}{\partial z_j^{(L)}} = \frac{\partial J}{\partial h_j} \frac{\partial h_j}{\partial z_j^{(L)}} = h_j - y_j$$

2. logistic神经元输出层 与均方误差代价函数。

logistic神经元

logistic神经元激活函数: $y = \frac{1}{1 + e^{-x}}$

logistic神经元输出层: $h = [a_1^{(L)}]^T$

作用: 输出二分类任务中正负类的概率。

logistic神经元的意义

若某神经元的输出值表示为正类的概率： $P_1 = h = a_1^{(L)}$

则负类的概率： $P_0 = 1 - h = 1 - a_1^{(L)}$

则几率 (odds)： $\frac{P_1}{P_0} = \frac{h}{1-h}$

对数几率 (logit odds)： $\ln \frac{P_1}{P_0} = \ln \frac{h}{1-h}$

对数几率拥有很好的数学特征，对数几率是高阶可导的凸函数。

logistic神经元的意义

对数几率 (logit odds) : $\ln \frac{P_1}{P_0} = \ln \frac{h}{1-h}$

令对数几率的值为 z ，求 h 关于 z 的表达式，则：

$$z = \ln \frac{P_1}{P_0} = \ln \frac{h}{1-h} \quad \longrightarrow \quad h = \frac{1}{1 + e^{-z}}$$

可以得到：输出层的输入值 z 表示的是对数几率。

logistic的作用是将几率转化为结果为正类的概率。

均方误差代价函数

公式：

$$J = \frac{1}{2m} \sum_{i=1}^m \|y^{(i)} - h^{(i)}\|^2$$

以单样本代价函数为例，输出层残差为：

$$\delta_j^{(L)} = \frac{\partial J}{\partial z_j^{(L)}} = \frac{\partial J}{\partial h_j} \frac{\partial h_j}{\partial z_j^{(L)}} = (h_j - y_j) \sigma'(z_j^{(L)})$$

可能导致梯度消失

3. 对数似然代价函数

使用最大似然估计得到代价函数

神经网络模型可以表示为： $h_{w,b}(x^{(i)})$

当h的结果为0~1之间的概率时，可以表示为：

$$p(x^{(i)}; w, b) = h_{w,b}(x^{(i)})$$

使用最大似然估计得到代价函数

若 $h_{w,b}(x^{(i)})$ 表示输出为正类的概率，
则 $1 - h_{w,b}(x^{(i)})$ 表示输出为负类的概率。

若将标记 y 记为类后验概率估计，则输出与输入一致的概率为：

$$p(y^{(i)} | x^{(i)}; w, b) = h_{w,b}(x^{(i)})^{y^{(i)}} * (1 - h_{w,b}(x^{(i)}))^{1-y^{(i)}}$$

使用最大似然估计得到代价函数

带入所有样本，可得到似然函数为：

$$L(w, b | x_1, x_2, \dots, x_m) = \prod_{i=1}^m p(y^{(i)} | x^{(i)}; w, b)$$

对似然函数取对数，可得对数似然函数为：

$$l = \ln(L(w, b | x_1, x_2, \dots, x_m)) = \sum_{i=1}^m \ln(p(y^{(i)} | x^{(i)}; w, b))$$

使用最大似然估计得到代价函数

将 $p(y^{(i)} | x^{(i)}; w, b) = h_{w,b}(x^{(i)})^{y^{(i)}} * (1 - h_{w,b}(x^{(i)}))^{1-y^{(i)}}$

带入 $l = \sum_{i=1}^m \ln(p(y^{(i)} | x^{(i)}; w, b))$ 可得：

$$\begin{aligned} l &= \sum_{i=1}^m \ln(h_{w,b}(x^{(i)})^{y^{(i)}} * (1 - h_{w,b}(x^{(i)}))^{1-y^{(i)}}) \\ &= \sum_{i=1}^m y^{(i)} \ln(h_{w,b}(x^{(i)})) + (1 - y^{(i)}) \ln(1 - h_{w,b}(x^{(i)})) \end{aligned}$$

使用最大似然估计得到代价函数

所以平均对数似然为：

$$\hat{\ell} = \frac{1}{m} \sum_{i=1}^m y^{(i)} \ln(h_{w,b}(x^{(i)})) + (1 - y^{(i)}) \ln(1 - h_{w,b}(x^{(i)}))$$

目标为最大化平均对数似然： $\arg \max \hat{\ell}(w, b | x_1, x_2, \dots, x_m)$

等价于最小化负的平均对数似然： $\arg \min(-\hat{\ell}(w, b | x_1, x_2, \dots, x_m))$

即代价函数为： $J = -\frac{1}{m} \sum_{i=1}^m y^{(i)} \ln(h_{w,b}(x^{(i)})) - (1 - y^{(i)}) \ln(1 - h_{w,b}(x^{(i)}))$

4. logistic神经元输出层 与对数似然代价函数

对数似然代价函数作为输出层

logistic神经元激活函数: $y = \frac{1}{1 + e^{-x}}$

logistic神经元输出层: $h = [a_1^{(L)}]^T$

代价函数: $J = -\frac{1}{m} \sum_{i=1}^m y^{(i)} \ln(h(x^{(i)})) - (1 - y^{(i)}) \ln(1 - h(x^{(i)}))$

对数似然代价函数作为输出层

logistic神经元: $y = \frac{1}{1 + e^{-x}}$

以单样本代价函数为例，输出层残差为：

$$\delta_j^{(L)} = \frac{\partial J}{\partial z_j^{(L)}} = \frac{\partial J}{\partial h_j} \frac{\partial h_j}{\partial z_j^{(L)}} = h_j - y_j$$

5. softmax神经元输出层 与交叉熵代价函数

softmax神经元输出层

softmax激活函数: $y_j = \frac{e^{x_j}}{\sum_i e^{x_i}}$

作用：输出二分类或多分类任务中某一类的概率。

意义：将输入排序，并转换为概率表示。

softmax求导

$$\frac{\partial y_j}{\partial x_k} = \frac{\partial}{\partial x_k} \left(\frac{e^{x_j}}{\sum_i e^{x_i}} \right)$$

当 $k=j$ 时，导数为： $y_j(1-y_j)$

当 $k \neq j$ 时，导数为： $-y_j y_k$

交叉熵代价函数

交叉熵代价函数：

$$J = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^{N^{(L)}} y_j^{(i)} \ln h_j(x^{(i)})$$

作用：比较两个概率分布的相似性。误差越大，代价越大。

softmax输出层与交叉熵代价函数

通常多分类任务中，标记使用one-hot编码。例如：

一个5分类的任务，表示每一类的方法如下：

$[1, 0, 0, 0, 0]$ 表示第一类的概率为100%，其它为0。

$[0, 1, 0, 0, 0]$ 表示第二类的概率为100%，其它为0。

$[0, 0, 1, 0, 0]$ 表示第三类的概率为100%，其它为0。

$[0, 0, 0, 1, 0]$ 表示第四类的概率为100%，其它为0。

$[0, 0, 0, 0, 1]$ 表示第五类的概率为100%，其它为0。

softmax输出层与交叉熵代价函数

所以可以使用交叉熵衡量标记的概率分布与模型输出的概率分布之间的误差：

$$J = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^{N^{(L)}} y_j^{(i)} \ln h_j(x^{(i)})$$

其中：y为one-hot向量，模型的输出层为softmax层。

softmax输出层与交叉熵代价函数

以单样本为例，代价函数为：
$$J = - \sum_{j=1}^{N^{(L)}} y_j \ln h_j(x)$$

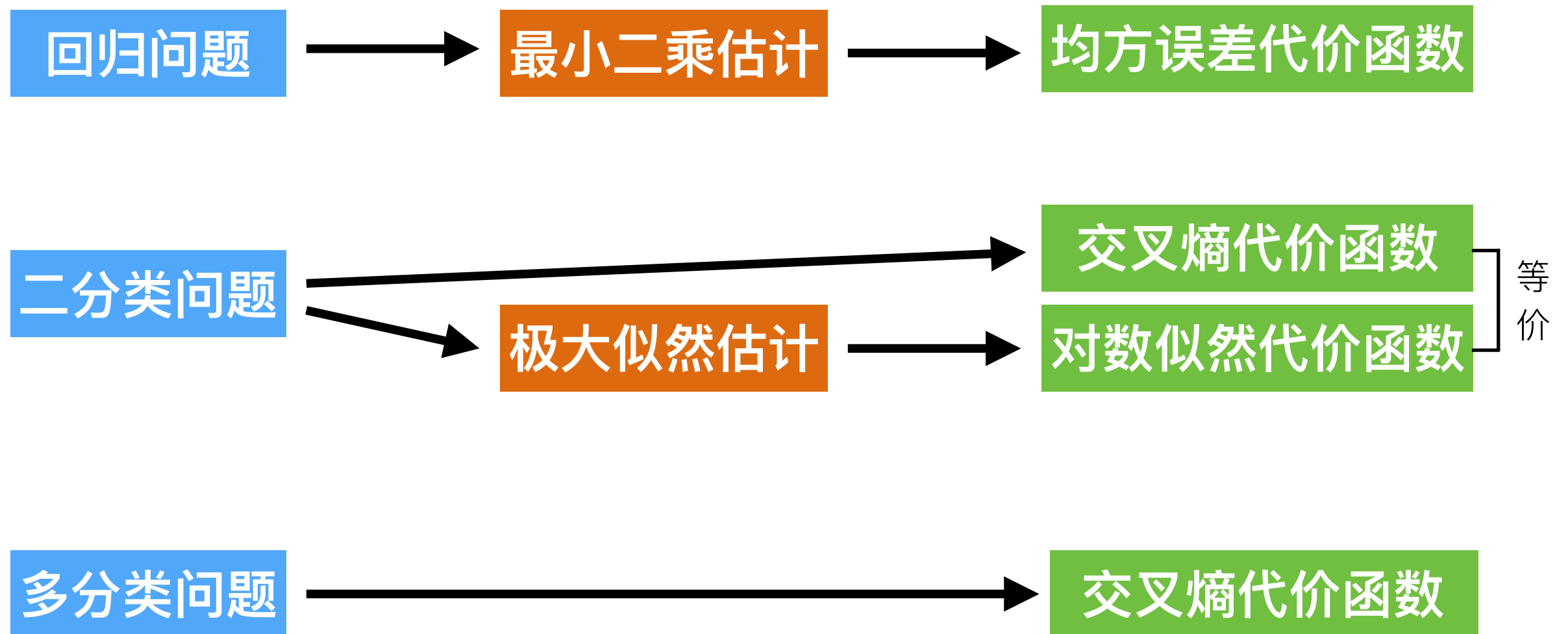
对输出层求残差：

$$\begin{aligned}\delta_j^{(L)} &= \frac{\partial J}{\partial z_j^{(L)}} = \sum_{k=1}^{N^{(L)}} \frac{\partial J}{\partial a_k^{(L)}} \frac{\partial a_k^{(L)}}{\partial z_j^{(L)}} \\ &= -\frac{y_j}{a_j^{(L)}} (a_j^{(L)} (1 - a_j^{(L)})) - \sum_{k \neq j}^{N^{(L)}-1} \frac{y_k}{a_k^{(L)}} * (-a_j^{(L)} a_k^{(L)}) \\ &= a_j \sum_{k=1}^{N^{(L)}} y_k - y_j = a_j - y_j\end{aligned}$$

注意：y的和为1

6. 三种代价函数对比

代价函数适用范围



小节

- 均方误差代价函数可以作为回归任务的代价函数，此时输出层通常使用线性激活函数。
- 均方误差代价函数可在分类任务中使用，但非最优。
- **logistic**函数在输出层的作用是将几率转化为概率。
- **softmax**函数在输出层的作用是将多个输出归一化，并转化为概率。输入越大，概率越大。
- 当遇到二分类任务时，可以选择**sigmoid**函数或者**softmax**函数作为输出层的激活函数。
- 当解决多分类任务时，选择**softmax**函数作为输出层激活函数。
- 输出层使用**sigmoid**激活函数时，输出层有且仅有一个神经元。其代价函数是交叉熵代价函数或者对数似然代价函数。
- 输出层使用**softmax**激活函数时，最适合的代价函数是对数似然代价函数。
- 对数似然代价函数可以看做是交叉熵代价函数在二分类任务中的特例。

THANKS