伯禹学习笔记篇壹

2020年2月14日 19:07

线性回归

1. 线性回归可以看作是没有非线性部分(激活函数) 的单层神经网络；
2. 线性回归的损失函数通常选取2范数；
3. 基于梯度下降法降低损失函数的原理，可以衍生出批量梯度下降方法；
4. 广播运算的高效性；

Softmax回归

1. Softmax运算符的介绍：

softmax运算符（softmax operator）解决了以上两个问题。它通过下式将输出值变换成值为正且和为1的概率分布：

$$\hat{y}_1, \hat{y}_2, \hat{y}_3 = \text{softmax}(o_1, o_2, o_3)$$

其中

$$\hat{y}1 = \frac{\exp(o_1)}{\sum_{i=1}^{3} \exp(o_i)}, \quad \hat{y}2 = \frac{\exp(o_2)}{\sum_{i=1}^{3} \exp(o_i)}, \quad \hat{y}3 = \frac{\exp(o_3)}{\sum_{i=1}^{3} \exp(o_i)}.$$

容易看出$\hat{y}_1 + \hat{y}_2 + \hat{y}_3 = 1$且$0 \le \hat{y}_1, \hat{y}_2, \hat{y}_3 \le 1$，因此$\hat{y}_1, \hat{y}_2, \hat{y}_3$是一个合法的概率分布。这时候，如果$\hat{y}_2 = 0.8$，不管$\hat{y}_1$和$\hat{y}_3$的值是多少，我们都知道图像类别为猫的概率是80%。此外，我们注意到

$$\arg\max_i o_i = \arg\max_i \hat{y}_i$$

因此softmax运算不改变预测类别输出。

2. Softmax回归是**多分类模型**，而不是回归模型；
3. Logistic回归用于二分类，softmax回归用于多分类，都可以看作是加入了非线性部分的单层神经网络，损失函数都采用**交叉熵损失函数**；

$$H\left(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}\right) = -\sum_{j=1}^{q} y_j^{(i)} \log \hat{y}_j^{(i)},$$

其区别在于一个是复合了sigmoid函数，一个是复合Softmax运算符。

多层感知机

1. 数据流

$$\boxed{\boldsymbol{a}_{i-1} = (\boldsymbol{a}_{i-1}^{1}, \boldsymbol{a}_{i-1}^{2}, \dots \boldsymbol{a}_{i-1}^{l_{i-1}})} \quad \boldsymbol{l}_{i-1}\text{维}$$

$$\boldsymbol{a_{i-1} = (a_{i-1}^{1},\ a_{i-1}^{2}, \dots\ a_{i-1}^{l_{i-1}})} \qquad \boldsymbol{l_{i-1}}\text{维}$$

$$\downarrow \quad \boldsymbol{w_i = (w_i^{\alpha\beta})_{l_{i-1}\times l_i}} \quad \boldsymbol{b_i = (b_i^{\beta})_{l_i\times 1}}$$

$$\boldsymbol{z_i = (z_i^1,\ z_i^2, \dots\ z_i^{l_i})} \qquad \boldsymbol{l_i}\text{维}$$

$$\downarrow \quad \boldsymbol{f_i}$$

$$\boldsymbol{a_i = (a_i^1,\ a_i^2, \dots\ a_i^{l_i})} \qquad \boldsymbol{l_i}\text{维}$$

$$\downarrow \quad \boldsymbol{w_{i+1} = (w_{i+1}^{\alpha\beta})_{l_i\times l_{i+1}}} \quad \boldsymbol{b_{i+1} = (b_{i+1}^{\beta})_{l_{i+1}\times 1}}$$

$$\boldsymbol{z_{i+1} = (z_{i+1}^1,\ z_{i+1}^2, \dots\ z_{i+1}^{l_{i+1}})} \qquad \boldsymbol{l_{i+1}}\text{维}$$

$$\downarrow \quad \boldsymbol{f_{i+1}}$$

$$\boldsymbol{a_{i+1} = (a_{i+1}^1,\ a_{i+1}^2, \dots\ a_{i+1}^{l_{i+1}})} \qquad \boldsymbol{l_{i+1}}\text{维}$$

2. 前向传播

$$z_i = w_i^T \cdot a_{i-1} + b_i \quad \text{且有} \frac{\partial z_i}{\partial a_{i-1}} = w_i^T$$

$$a_i = f_i(z_i)$$

3. 反向传播

假设 $f(x)=\frac{1}{1+e^{-x}}$ ，而 $J=\frac{1}{2}\sum_{m=1}^{n}(\hat{y}_m-y_m)^2$ 故有 $\frac{\partial J}{\partial a_{end}}=\frac{\partial J}{\partial \hat{y}}=(\hat{y}_1,\hat{y}_2,\dots,\hat{y}_n)$

$$\frac{\partial J}{\partial z_i}=\frac{\partial J}{\partial z_{i+1}}\cdot\frac{\partial z_{i+1}}{\partial a_i}\cdot\frac{\partial a_i}{\partial z_i}$$

即

$$\boldsymbol{\frac{\partial J}{\partial z_i}=\left(\frac{\partial J}{\partial z_{i+1}}\cdot w_{i+1}^{T}\right)\circ\left(a_i(1-a_i)\right)}$$

根据上述公式反向迭代直到$a_{end}=\hat{y}$ 为止，从而可以推出$\frac{dJ}{dz_i}$

$$\boldsymbol{\frac{dJ}{dw_i}=\frac{dJ}{dz_i}\cdot\frac{dz_i}{dw_i}=a_{i-1}\otimes\frac{dJ}{dz_i}}$$

$$\boldsymbol{\frac{dJ}{db_i}=\frac{dJ}{dz_i}\cdot\frac{dz_i}{db_i}=\frac{dJ}{dz_i}}$$

4. 激活函数

ReLU函数

$$\mathrm{ReLU}(x)=\max(x,0).$$

sigmoid函数

$$\mathrm{sigmoid}(x)=\frac{1}{1+\exp(-x)}.$$

tanh函数

$$\tanh(x)=\frac{1-\exp(-2x)}{1+\exp(-2x)}.$$

如何选择?

ReLu函数是一个通用的激活函数，目前在大多数情况下使用。但是，ReLU函数只能在隐藏层中使用。

用于分类器时，sigmoid函数及其组合通常效果更好。由于梯度消失问题，有时要避免使用sigmoid和tanh函数。

在神经网络层数较多的时候，最好使用ReLu函数，ReLu函数比较简单计算量少，而sigmoid和tanh函数计算量大很多。

在选择激活函数的时候可以先选用ReLu函数如果效果不理想可以尝试其他激活函数。