作业一：MLQP

超并行机器学习与海量数据挖掘　EI328
2022 年春季工程实践与科技创新课程 IV-J

姓名：李子龙　学号：518070910095　日期：2022 年 3 月 4 日

1　推导

问题 1. Suppose the output of each neuron in a multi- layer perceptron is:

$$x_{kj} = f\left(\sum_{i=1}^{N_{k-1}} (u_{kji} x_{k-1,i}^2 + v_{kji} x_{k-1,i}) + b_{kj}\right) \tag{1}$$

where both u_{kji} and v_{kji} are the weights connecting the i^{th} unit in the layer $k-1$ to the j^{th} unit in the layer $k \in [2, M]$, b_{kj} is the bias of the j^{th} unit in the layer $k \in [2, M]$, N_k is the number of units if $1 \leq k \leq M$ and $f(\cdot)$ is the sigmoidal activation function.

Please derive a back-propagation algorithm for multilayer quadratic perceptron (MLQP) in on-line or sequential mode.

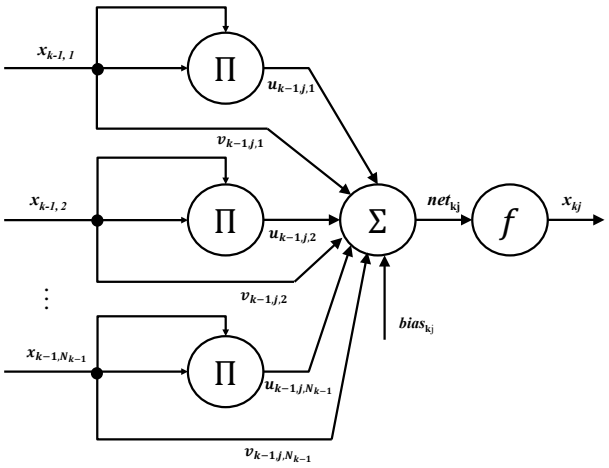

图 1: MLQP

1.1　输出神经元

对于输出神经元信号 x_{Mj} 来说，令 d_{Mj} 为目标值，则对应的误差为

$$e_{Mj} = d_{Mj} - x_{Mj}$$

使用均方误差计算损失函数

$$\mathcal{E} = \sum_{j=1}^{N_M} \frac{1}{2} e_{Mj}^2 \tag{2}$$

则对于 net_{Mj} 的梯度为

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial net_{Mj}} &= \frac{\partial \mathcal{E}}{\partial e_{Mj}} \frac{\partial e_{Mj}}{\partial x_{Mj}} \frac{\partial x_{Mj}}{\partial net_{Mj}} \\ &= -e_{Mj} f'(net_{Mj})\end{aligned}$$

定义对应的局部梯度（local gradient）为

$$\delta_{Mj} = -\frac{\partial \mathcal{E}}{\partial net_{Mj}} = e_{Mj} f'(net_{Mj}) \tag{3}$$

1.2 隐藏层

对于隐藏神经元上的 $net_{k-1,j}$ 而言，假设后一层反向传播来的局部梯度已知：

$$\delta_{kj} = -\frac{\partial \mathcal{E}}{\partial net_{kj}}$$

则该层的局部梯度为

$$\begin{aligned}\delta_{k-1,i} = -\frac{\partial \mathcal{E}}{\partial net_{k-1,i}} &= -\frac{\partial \mathcal{E}}{\partial x_{k-1,i}} \frac{\partial x_{k-1,i}}{\partial net_{k-1,i}} \\ &= -f'(net_{k-1,i}) \sum_{j=1}^{N_j} \frac{\partial \mathcal{E}}{\partial net_{kj}} \frac{\partial net_{kj}}{\partial x_{k-1,i}} \\ &= f'(net_{k-1,i}) \sum_{j=1}^{N_j} \delta_{kj} (2u_{kji} x_{k-1,i} + v_{kji})\end{aligned} \tag{4}$$

1.3 更新权值

而根据公式 (1)，

$$\begin{aligned}\frac{\partial net_{kj}}{\partial u_{kji}} &= x_{k-1,i}^2 \\ \frac{\partial net_{kj}}{\partial v_{kji}} &= x_{k-1,i}\end{aligned}$$

设定学习率分别为 η_1, η_2，则对应权重的修正值

$$\Delta u_{kji} = -\eta_1 \frac{\partial \mathcal{E}}{\partial u_{kji}} = -\eta_1 \frac{\partial \mathcal{E}}{\partial net_{kj}} \frac{\partial net_{kj}}{\partial u_{kji}} = \eta_1 \delta_{kj} x_{k-1,i}^2 \tag{5}$$

$$\Delta v_{kji} = -\eta_2 \frac{\partial \mathcal{E}}{\partial v_{kji}} = -\eta_2 \frac{\partial \mathcal{E}}{\partial net_{kj}} \frac{\partial net_{kj}}{\partial v_{kji}} = \eta_2 \delta_{kj} x_{k-1,i} \tag{6}$$

这个结果与 [1] 基本一致。

1.4 更新偏移

类似地由于

$$\frac{\partial net_{kj}}{\partial b_{kj}} = 1$$

则设定学习率为 η_3，则对应的偏移修正值

$$\Delta b_{kj} = -\eta_3 \frac{\partial \mathcal{E}}{\partial b_{kj}} = -\eta_3 \frac{\partial \mathcal{E}}{\partial net_{kj}} \frac{\partial net_{kj}}{\partial b_{kj}} = \eta_3 \delta_{kj} \tag{7}$$

2 实现

问题 2. Please implement an on-line BP algorithm for MLQP (you can use any programming language), train an MLQP with one hidden layer to classify two spirals problem, and compare the training time and decision boundaries at three different learning rates.

2.1 结构设计

如图 2，输入为两个维度上的坐标 x, y。采用单输出与硬阈值分类，阈值设定为 0.5，按照 [2] 的说法，识别时将大于等于 0.5 的设为正类，小于 0.5 的定义为负类，但这并不影响训练过程，因为采用的是原始输出作为误差的计算。

由于只需要一个隐藏层，隐藏层神经元个数为 N_2，那么参数个数为

$$2(2N_2 + N_2) + N_2 + 1 = 7N_2 + 1$$

由于训练集样本数为 300，假设我们想要保持 10% 的分类错误误差，那么为了好的泛化能力，参数个数应当满足

$$300 = O\left(\frac{7N_2 + 1}{10\%}\right)$$

N_2 至少应当是 5 的倍数级别，仿照 [1]，这里取 10。

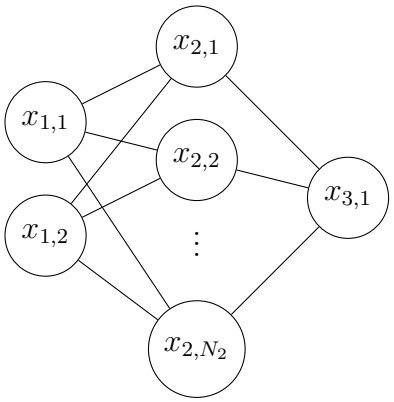

图 2: 网络结构

2.2 算法小结

下面将采用矩阵记号简化运算，以下 $\times$ 均表示矩阵的逐点相乘。设第 $k-1$ 层的数值矩阵为 $\mathbf{X}_{k-1}=(x_{k-1,i})_{N_{k-1}\times 1}$，其逐项平方矩阵为 $\mathbf{Y}_{k-1}=(x_{k-1,i}^2)_{N_{k-1}\times 1}=\mathbf{X}_{k-1}\times\mathbf{X}_{k-1}$，向后的权重矩阵为 $\mathbf{U}_k=(u_{kji})_{N_k\times N_{k-1}}$, $\mathbf{V}_k=(v_{kji})_{N_k\times N_{k-1}}$，偏移量矩阵 $\mathbf{B}_k=(b_{kj})_{N_k\times 1}$。

2.2.1 前向运算

公式 (1) 将被改写为

$$\mathbf{X}_k=f\left(\mathbf{U}_k\mathbf{Y}_{k-1}+\mathbf{V}_k\mathbf{X}_{k-1}+\mathbf{B}_k\right)$$

为了简单起见，保留中间结果

$$\mathbf{N}_k=\mathbf{U}_k\mathbf{Y}_{k-1}+\mathbf{V}_k\mathbf{X}_{k-1}+\mathbf{B}_k \tag{8}$$

$$\mathbf{X}_k=f(\mathbf{N}_k) \tag{9}$$

2.2.2 后向运算

令 $\mathbf{D}_M$ 为目标输出矩阵，将公式 (3) 改写，输出层的梯度计算为

$$\boldsymbol{\delta}_M=(\delta_{M,i})_{N_M\times 1}=(\mathbf{D}_M-\mathbf{X}_M)\times f'(\mathbf{N}_M) \tag{10}$$

改写公式 (4)，隐含层的梯度计算为

$$\boldsymbol{\delta}_{k-1}=\left(\mathbf{U}_k^T\boldsymbol{\delta}_k\times 2\mathbf{X}_{k-1}+\mathbf{V}_k^T\boldsymbol{\delta}_k\right)\times f'(\mathbf{N}_{k-1}) \tag{11}$$

改写公式 (5) 和公式 (6)，权重修正矩阵

$$\boldsymbol{\Delta}\mathbf{U}_k=(\Delta u_{kji})_{N_k\times N_{k-1}}=\eta_1\boldsymbol{\delta}_k\mathbf{Y}_{k-1}^T \tag{12}$$

$$\boldsymbol{\Delta}\mathbf{V}_k=(\Delta v_{kji})_{N_k\times N_{k-1}}=\eta_2\boldsymbol{\delta}_k\mathbf{X}_{k-1}^T \tag{13}$$

改写公式 (7)，偏移修正矩阵

$$\boldsymbol{\Delta}\mathbf{B}_k=(\Delta b_{kj})_{N_k\times 1}=\eta_3\boldsymbol{\delta}_k \tag{14}$$

2.3 训练过程

根据 [3]，将参数初始化为 $\mathcal{U}\left(-\sqrt{\frac{1}{2}},\sqrt{\frac{1}{2}}\right)$ 上的随机值，其中 $\mathcal{U}$ 为一致分布。
简便起见，将三个参数的学习率设置为一个相等的值 $\eta_1=\eta_2=\eta_3=\eta$。
数据范围是 $x\in[-6,6], y\in[-6,6]$。

参考文献

[1] LU B L, BAI Y, KITA H, et al. An efficient multilayer quadratic perceptron for pattern classification and function approximation[C/OL]//Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan): volume 2. IEEE, 1993: 1385-1388. DOI: 10.1109/ijcnn.1993.716802.

[2] 胡晓武, 秦婷婷, 李超, 等. 智能之门[M]. 高等教育出版社, 2020.

[3] Linear — pytorch master documentation[M/OL]. PyTorch, 2022. https://pytorch.org/docs/master/generated/torch.nn.Linear.html#torch.nn.Linear.