模式识别：作业 #4

201928014629008
牛李金梁

2019 年 12 月 5 日

第一部分：计算与证明

Question 1

(a)

记第 k 个样本 $\boldsymbol{x}^k = \left[1, x_1^k, x_2^k, \cdots, x_d^k\right]^T$。设隐含层有 n 个神经元，输入层到隐含层的连接权重为 $(d+1)\times n$ 维的矩阵 W^H，其中某个元素记为 w_{ih}；输出层有 c 个神经元，隐含层到输出层的连接权重为 $(n)\times c$ 维的矩阵 W^c，其中某个元素记 w_{hj}。

在前馈过程中，隐含层 h 结点的输出为 $y_h^k = f_1\left(net_h^k\right) = f\left(\sum_i w_{ih}x_i^k\right)$，$f$ 为 sigmoid 函数；输出层 j 结点的输出为 $z_j^k = f_2\left(net_j^k\right) = f\left(\sum_h w_{hj}y_h^k\right) = f_2\left(\sum_h w_{hj}f_1\left(\sum_i w_{ih}x_i^k\right)\right)$，$f$ 为 softmax 函数。

误差函数为

$$\begin{aligned} E(w)^k = J(w)^k &= \frac{1}{2}\sum_j \left(t_j^k - z_j^k\right)^2 \\ &= \frac{1}{2}\sum_j \left(t_j^k - f\left(net_j^k\right)\right)^2 \\ &= \frac{1}{2}\sum_j \left(t_j^k - f\left(\sum_h w_{hj}f\left(net_h^k\right)\right)\right)^2 \end{aligned}$$

隐含层到输出层的连接权重调节量：

$$\begin{aligned} \Delta w_{hj} = -\eta\frac{\partial E}{\partial w_{hj}} &= -\eta\sum_k \frac{\partial E}{\partial net_j^k}\frac{\partial net_j^k}{\partial w_{hj}} \\ &= \eta\sum_k \left(t_j^k - z_j^k\right) f'\left(net_j^k\right) y_h^k \\ &= \eta\sum_k \delta_j^k y_h^k \end{aligned}$$

其中，$\delta_j^k = -\frac{\partial E}{\partial net_j^k} = f'\left(net_j^k\right)\left(t_j^k - z_j^k\right)$，$f$ 为 softmax 函数。softmax 的导数为雅可比矩阵，此处需要的是矩阵主对角线上的元素。设 softmax 主对角线上的第 j 个元素是 S_j，有 $S_j{}' = S_j\left(1 - S_j\right)$。

因此，$\Delta w_{hj} = \eta\sum_k \left(t_j^k - z_j^k\right) S_j\left(1 - S_j\right) y_h^k$，其中，$S_j = \frac{\exp\left(net_j^k\right)}{\sum_i \exp\left(net_i^k\right)}$

输入层到隐含层的连接权重调节量：

$$\begin{aligned}
\Delta w_{ih} = -\eta\frac{\partial E}{\partial w_{ih}} &= -\eta\sum_{k,j}\frac{\partial E}{\partial z_j^k}\frac{\partial z_j^k}{\partial w_{ih}} \\
&= \eta\sum_{k,j}\left(t_j^k - z_j^k\right)\frac{\partial z_j^k}{\partial w_{ih}} \\
&= \eta\sum_{k,j}\left(t_j^k - z_j^k\right)\frac{\partial z_j^k}{\partial net_j^k}\frac{\partial net_j^k}{\partial w_{ih}} \\
&= \eta\sum_{k,j}\left(t_j^k - z_j^k\right)f'\left(net_j^k\right)\frac{\partial net_j^k}{\partial y_h^k}\frac{\partial y_h^k}{\partial w_{ih}} \\
&= \eta\sum_{k,j}\left(t_j^k - z_j^k\right)f'\left(net_j^k\right)w_{hj}\frac{\partial y_h^k}{\partial w_{ih}} \\
&= \eta\sum_{k,j}\left(t_j^k - z_j^k\right)f'\left(net_j^k\right)w_{hj}\frac{\partial y_h^k}{\partial net_h^k}\frac{\partial net_h^k}{\partial w_{ih}} \\
&= \eta\sum_{k,j}\left(t_j^k - z_j^k\right)f'\left(net_j^k\right)w_{hj}f'\left(net_h^k\right)x_i^k \\
&= \eta\sum_{k}\delta_h^k x_i^k
\end{aligned}$$

其中，$\delta_h^k = -\frac{\partial E}{\partial net_h^k} = f'\left(net_h^k\right)\sum_j w_{hj}\delta_h^k$，$\delta_j^k = f'\left(net_j^k\right)\left(t_j^k - z_j^k\right)$。

sigmoid 函数的导数 $S_h{}' = S_h\left(1 - S_h\right)$，于是，$\Delta w_{ih} = \sum_j w_{hj}\delta_j^k S_h\left(1 - S_h\right)x_i^k$，$S_h = \frac{1}{1+\exp\left(-net_h^k\right)}$

(b)

通过 (a) 中的推导可以看出，反向传播算法将各层间的连接权重分隔开，使之相互独立，传播时输出层的误差沿着前馈网络的反方向逐层更新层间的连接权重，从而避免了深层网络中复合函数链式求导中过长的链。这一算法对简化神经网络优化过程有着重要的意义。

Question 2

计算步骤：

- 初始化网络。通常使用随机初始化的方法。
- 输入训练样本，每个样本为一个 d 维向量。
- 计算映射层的权重向量和输入向量的距离。$d_j = \sqrt{\sum_{i=1}^{d}\left(x_i - w_{ij}\right)^2}$。
- 选择与权重向量距离最小的神经元作为胜出神经元 j^*，并给出其邻接神经元集合 $h\left(., j^*\right)$
- 调整胜出神经元和其邻接神经元的权重，按下式更新：

$$\Delta w_{ij} = \eta h\left(j, j^*\right)\left(x_i - w_{ij}\right)$$

$$w_{ij}\left(t+1\right) = w_{ij}\left(t\right) + \Delta w_{ij}$$

- 检查是否达到预设要求，是则结束算法，否则进行迭代。

算法流程图：

图 1: 自组织算法流程图

Question 3

(1)

输入 图像大小：400×400

第一卷积层 滤波器大小：5×5

结点图像个数：20

图像大小：396×396

pooling 后图像大小：198×198

包含偏置的训练权重个数：$5 \times 5 \times 20 + 20 = 520$

第二卷积层 滤波器大小：$3 \times 3 \times 20$

结点图像个数：30

滤波器总数：30

图像大小：196×196

pooling 后图像大小：98×98

包含偏置的训练权重个数：$20 \times 3 \times 3 \times 30 + 30 = 5430$

第三卷积层 滤波器大小：$3 \times 3 \times 30$

结点图像个数：20

滤波器总数：20

图像大小：96×96

pooling 后图像大小：48×48

包含偏置的训练权重个数：$30 \times 3 \times 3 \times 20 + 20 = 5420$

第四卷积层 滤波器大小：$3 \times 3 \times 10$

结点图像个数：10

滤波器总数：10

特征图尺度：46×46

pooling 后图像大小：23×23

包含偏置的训练权重个数：$20 \times 3 \times 3 \times 10 + 10 = 1810$

全连接层 输入图像大小：23×23

输入图像个数：10

输出结点个数：10

包含偏置的训练权重个数：$10 \times 23 \times 23 \times 10 + 10 = 52910$

采用权值共享和局部连接的总权重个数为 65290。

采用全连接网络：

输入层到第一隐层：$400 \times 400 \times 396 \times 396 \times 20 + 396 \times 396 \times 20 = 501814336320$
第一隐层到第二隐层：$198 \times 198 \times 20 \times 196 \times 196 \times 30 + 196 \times 196 \times 30 = 903637670880$
第二隐层到第三隐层：$98 \times 98 \times 30 \times 96 \times 96 \times 20 + 96 \times 96 \times 20 = 53106462720$
第三隐层到第四隐层：$48 \times 48 \times 20 \times 46 \times 46 \times 10 + 46 \times 46 \times 10 = 975073960$
第四隐层到输出层：$23 \times 23 \times 10 \times 10 + 10 \times 10 = 53000$
总权重个数约为 1.459×10^{12}。

不采用权值共享：

输入层到第一隐层：$5 \times 5 \times 396 \times 396 \times 20 + 396 \times 396 \times 20 = 105304320$
第一隐层到第二隐层：$3 \times 3 \times 20 \times 196 \times 196 \times 30 + 196 \times 196 \times 30 = 208598880$
第二隐层到第三隐层：$3 \times 3 \times 30 \times 96 \times 96 \times 20 + 96 \times 96 \times 20 = 49950720$
第三隐层到第四隐层：$3 \times 3 \times 20 \times 46 \times 46 \times 10 + 46 \times 46 \times 10 = 3829960$
第四隐层到输出层：$23 \times 23 \times 10 \times 10 + 10 \times 10 = 53000$
总权重个数约为 3.67×10^{8}。

与二者相比，采用权值共享和局部连接的总权重个数基本可以忽略。

(2)

max pooling 的前向传播是舍去其他元素，把区域内最大的值传给后一层。反向传播则把梯度直接传给前一层该区域的某一个像素，而其他像素为 0。确定这个像素需要记录下 pooling 时取的是哪个像素，即最大值所在位置。

(3)

能改变网络结构的因素：

- 改变网络层数
- 采用不同大小的滤波器
- 激励函数的选择
- 池化操作的选择
- 能量函数的选择

第二部分：计算机编程

环境为 python3.7
依赖库：numpy、matplotlib
运行方式：直接在 cmd 中执行相应 py 文件

Question 1

对于 3 层前向神经网络并使用反向传播进行训练，要确定以下参数：

- 批大小
- 隐层结点个数
- 梯度更新步长
- 训练次数

还要根据已知参数确定输入层结点个数（样本维度），输出层类别个数。

具体训练中，先随机初始化层间权重，接着进行多次迭代。将数据分批送入网络前向传播，并根据误差函数（平方误差）计算误差，然后将误差反向传播来修正层间权值。经过迭代，误差越来越小，直到迭代次数满足预设的训练次数，则根据预测判断正确率。

Question 2

(a)

隐含层结点个数从 1 到 10 逐步增加，计算误差和训练集准确率，得到下面的图。

图 2: 隐含层结点个数对训练精度的影响

从图可知，随着隐含层结点数目的增加，训练误差整体趋势是减小，分类准确率整体趋势是增加。但也可以看出，结点数并不需要无限制的增加，只需要采用适当的结点数，便可以有效地进行训练了。

(b)

设定隐含层结点数为 8，梯度更新步长从 0.002 到 0.2 以 0.002 的步长变化，得到下面的图。

图 3: 梯度更新步长对训练的影响

从图中可以看出，随着更新步长的增加，训练误差呈先减小后增加的态势，训练准确率也呈先提高后减小的态势。

出现这种情况是因为以较小的更新步长进行训练时，在有限迭代次数内仍未收敛，所以此时的误差较高准确率较低。而对于较大的更新步长，由于更新步长过大，所得的解一直在最优值附近徘徊却无法到达，因此误差和准确率出现了震荡。

因此，网络训练中应采取合适的更新步长，过小则收敛慢，过大则效果不好。

(3)

本实验将 epoch 设为 1000，采用批量更新的方法，batch size 设为 10，即总共进行 3000 次迭代，隐含层结点数为 8，学习率为 0.1，结果如下图所示。从图中可以看出，本程序有良好的收敛特性。

图 4: 目标函数随迭代次数增加的变化