1. Consider a three-layer network for classification with n_H nodes in hidden layer, and c nodes in output layer. The patterns (also say samples) are in d dimensional space. The activation function (or transfer function) for the nodes in the hidden layer is the sigmoid function. Differently, the nodes in the output layer will employ the following softmax operation as their activation function:

$$z_j = \frac{e^{net_j}}{\sum_{m=1}^{c} e^{net_m}}, \quad j = 1, 2, ..., c,$$

where net_j stands for the weighted sum at the j-th node in the output layer.

(a) Derive the learning rule under the back propagation framework if the criterion function for each sample is the sum of the squared errors, that is （即分析每一层权重的更新方法）:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{c}(t_j - z_j)^2 ,$$

Where t_j is the known target value for the sample at the j-th node in the output layer.

(b) 结合课堂所学知识，对反向传播算法进行总结。

注意：本题只需要推导出单个样本对权重更新的贡献即可（因为多个样本只是简单地相加）

1.(a)

(1) 对于单个样本对隐藏层到输出层的权重:

单个样本的更新量:

$$\Delta w_{hj} = \eta \delta_j y_h$$
$$h = 1, 2, \cdots, n_H; \qquad j = 1, 2, \cdots, c$$
$$\delta_j = f'(net_j)(t_j - z_j)$$

其中，导数部分的计算为:

$$f'(net_j) = \frac{e^{net_j}(\sum_{m=1}^{L} e^{net_m}) - e^{net_j}e^{net_j}}{(\sum_{m=1}^{L} e^{net_m})^2} = \frac{e^{net_j}(\sum_{m=1,m\neq j}^{L} e^{net_m})}{(\sum_{m=1}^{L} e^{net_m})^2}$$

所以得出结果为:

$$\delta_j = f'(net_j)(t_j - z_j) = \frac{e^{net_j}(\sum_{m=1,m\neq j}^{L} e^{net_m})}{(\sum_{m=1}^{L} e^{net_m})^2}(t_j - z_j)$$

(2) 对于单个样本对输入层到隐藏层之间的权重:

单个样本的更新量:

$$\Delta w_{ih} = \eta x_i \delta_h$$
$$h = 1, 2, \cdots, n_H; \qquad i = 1, 2, \cdots, d$$
$$\delta_h = f'(net_h) \sum_{j=1}^{d} \delta_j w_{hj}$$

其中，导数部分的计算为 (sigmoid 函数)：

$$f(net_h) = \frac{1}{1 + e^{-net_h}}$$
$$f'(net_h) = \frac{e^{-net_h}}{(1 + e^{-net_h})^2}$$

所以得出结果为：

$$\delta_h = \frac{e^{-net_h}}{(1 + e^{-net_h})^2} \sum_{j=1}^{d} \delta_j w_{hj}$$

1.(b)
反向传播算法总结，结合多层的神经网络的情况下的方向算法传播。
(1) 隐藏层到输出层的权重 (这里的隐藏层 h 表示输出层 j 之前的那一层隐藏层):
对于单个样本：

$$\Delta w_{hj} = \eta \delta_j y_h$$
$$\delta_j = f'(net_j) \left(-\frac{dE}{df(net_j)} \right)$$

其中 $-\frac{dE}{df(net_j)}$ 为误差 E 对于输出节点的求导
对于多个样本：

$$\Delta w_{hj} = \sum_k \eta \delta_j^k y_h^k$$
$$\delta_j = f'(net_j^k) \left(-\frac{dE}{df(net_j^k)} \right)$$

其中 $-\frac{dE}{df(net_j^k)}$ 为误差 E 对于输出节点的求导.
2. 隐藏层, 这里以 h_1, h_2, h_3 三层的隐藏层用来展示在普通隐藏层之间的误差传递：
对于单个样本：

$$\Delta w_{h_1 h_2} = \eta y_{h_1} \delta_{h_2}$$
$$\delta_{h_2} = f'(net_{h_2}) \sum_{h_3} w_{h_2 h_3} \delta_{h_3}$$

对于多个样本:

$$\Delta w_{h_1 h_2} = \eta \sum_k y_{h_1}^k \delta_{h_2}^k$$
$$\delta_{h_2}^k = f'(net_{h_2}) \sum_{h_3} w_{h_2 h_3} \delta_{h_3}$$

3. 输入层到隐藏层，这里输入层是 i，输入层后面的隐藏层是 h, 隐藏层后面的一层隐藏层是 h_1 对于单个样本：

$$\Delta w_{ih} = \eta x_i \delta_{h_1}$$
$$\delta_h = f'(net_h) \sum_{h_1} w_{hh_1} \delta_{h_1}$$

对于多个样本:

$$\Delta w_{ih} = \eta \sum_k x_i^k \delta_h^k$$
$$\delta_h^k = f'(net_h) \sum_{h_1} w_{hh_1} \delta_{h_1}$$

以上是反向传播的算法，这里分了三类进行讨论总结。

2. 请描述自组织算法的计算步骤，给出训练算法的框图。

前向计算：对于输入向量，对于其每一个分量，会有一个特定的输出神经元节点（胜出神经元节点）与之对应。
在训练初期，对于样本中的每一个分量，其对应的胜出神经元和其附近的神经元距离接近，形成粗略映射。但是训练之后，胜出邻域会变窄，胜出神经元附近的神经元数会变少，最后只会有胜出的神经元对于输入的对应的样本有响应。训练算法步骤：

step1. 初始化、归一化所有权向量 w_j，建立初始优胜邻域 $N_j, j = 1, 2, \cdots, c$，学习率 η
step2. 随机选取一个样本输入 x, 其分量 $x_i i = 1, 2, \cdots, d$
step3. 计算每一个权重对应样本的响应（距离形式）$d_j = \sqrt{\sum_{i=1}^{d} (x_i - w_{ij})^2}$
step4. 选择胜者神经元（距离最小）记录其位置（index）为 j^*，记录胜者神经元的邻域 $h(., j^*)$
step6. 更新胜出节点和其邻接节点的权重的值:$w_{ij}(t+1) = w_{ij}(t) + \eta h(j, j*)[x_i - w_{ij}(t)]$
step7. 判断是否达到预先设定的收敛的要求，不是的话返回 step2.

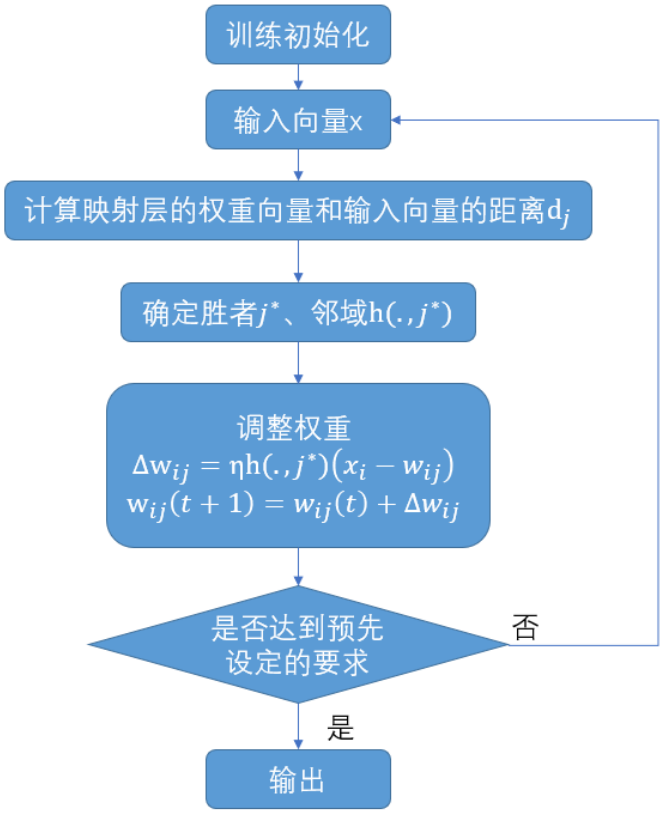

图 1: SOM 训练算法框图

参考了 PPT 那一章节的自组织算法的计算步骤，以及 SOM 训练过程的算法过程。

3. 拟考虑对 400×400 大小的图像数据集运用卷积神经网络。假定共有 4 个卷积层。第一隐含层采用 5×5 大小的滤波器，其它卷积层采用 3×3 大小的滤波器。第 1 至第 4 卷积层分别包含的图像数目为 20，30，20，10。Pooling 操作均采用在 2×2 大小的局部窗口内取最大值(即 max pooling)，同时假定在 max pooling 之后完成激励操作。假定连接到最后一个 pooling 操作之后的前向神经网络为单层前向神经网络，且输出层的结点个数为 10。（1）请按层指出该网络需要计算的权重数量；相对于全连接和非权值共享，请指出当同时采用权值共享和局部连接时所减少的权重数量；（2）请指出在遇到 max pooling 操作时，在进行反向传播时错误如何传？（3）请讨论你所能想到的对网络结构的改变。（本题可在下一章讲完之后再做，本次作业先布置）

(1) 参考 PPT 中的写法，为了方便计算，这里参数忽略了阈值的参数
第一层 $5 \times 5 \times 20$
第二层 $3 \times 3 \times 20 \times 30$
第三层 $3 \times 3 \times 30 \times 20$
第四层 $3 \times 3 \times 20 \times 10$
全连接输出层 $23 \times 23 \times 10 \times 10$

如果是使用全连接和非权重共享
第一层 $400 \times 400 \times 396 \times 396 \times 20$
第二层 $198 \times 198 \times 196 \times 196 \times 20 \times 30$
第三层 $98 \times 98 \times 96 \times 96 \times 30 \times 20$
第四层 $48 \times 48 \times 46 \times 46 \times 20 \times 10$
全连接输出层 $23 \times 23 \times 10 \times 10$

上面两种情况做差值，得出减少的权重为：1459529036500，可以看到同时采用权值共享和局部连接减少了非常大的数量的权重。

(2) 构造传播的问题，一定要遵循传播梯度综合保持不变的原则在反向传播的时候，框架需要将该区域的梯度直接分配到最大神经元，这里由于在池化层后面添加了激活操作，所以要将梯度再乘上激活函数在池化最大值处的导数，其他神经元的梯度被分配为 0 且是被舍弃不参与反向传播的，但是如何确认最大神经元，这个还是需要框架再进行前向传播的时候记录下最大神经元的 id 位置。(参考资料 https://zhuanlan.zhihu.com/p/258604402)

(3) 网络结构的改变：
根据训练情况，减少或者增加每层的卷积层数，或者修改卷积核的大小
跨层连接 (skip connections)，跨卷积层的连接，对于不同层提取到的不同 resolution 的 features 进行整合。

第二部分：计算机编程

本题使用的数据如下：

第一类 10 个样本（三维空间）：
[1.58, 2.32, -5.8], [0.67, 1.58, -4.78], [1.04, 1.01, -3.63],
[-1.49, 2.18, -3.39], [-0.41, 1.21, -4.73], [1.39, 3.16, 2.87],
[1.20, 1.40, -1.89], [-0.92, 1.44, -3,22], [0.45, 1.33, -4.38],
[-0.76, 0.84, -1.96]

第二类 10 个样本（三维空间）：
[0.21, 0.03, -2.21], [0.37, 0.28, -1.8], [0.18, 1.22, 0.16],
[-0.24, 0.93, -1.01], [-1.18, 0.39, -0.39], [0.74, 0.96, -1.16],
[-0.38, 1.94, -0.48], [0.02, 0.72, -0.17], [0.44, 1.31, -0.14],
[0.46, 1.49, 0.68]

第三类 10 个样本（三维空间）：
[-1.54, 1.17, 0.64], [5.41, 3.45, -1.33], [1.55, 0.99, 2.69],
[1.86, 3.19, 1.51], [1.68, 1.79, -0.87], [3.51, -0.22, -1.39],
[1.40, -0.44, -0.92], [0.44, 0.83, 1.97], [0.25, 0.68, -0.99],
[0.66, -0.45, 0.08]

1. 请编写两个通用的三层前向神经网络反向传播算法程序，一个采用批量方式更新权重，另一个采用单样本方式更新权重。其中，隐含层结点的激励函数采用双曲正切函数，输出层的激励函数采用 sigmoid 函数。目标函数采用平方误差准则函数。
2. 请利用上面的数据验证你写的程序，分析如下几点：
   (a) 隐含层不同结点数目对训练精度的影响；
   (b) 观察不同的梯度更新步长对训练的影响，并给出一些描述或解释；
   (c) 在网络结构固定的情况下，绘制出目标函数随着迭代步数增加的变化曲线。

1. 详细部分见代码，代码中 HMDataTrain.py 是针对题目的运行、训练函数，直接运行就好，但是运行时间会比较长的，代码中的 network.py 是网络架构，主要存放的就是根据题目写的相关的网络架构和反向传播算法，网络只是设计了三层，每一层的节点数量可以改变，不能增加新的层数，如果要增加新的层数的话，需要修改代码。

需要的环境：

numpy 1.16.5

matplotlib 3.1.1

argparse

代码中的注释已经写的比较清楚，可以打开。

还写了一个用于生成网络架构并且训练的通用的程序 train.py，使用 argparse 进行调试，如果需要使用的话可以参照其中的–help。

2. (a) 隐藏层的不同的节点数，根据实验，实验结果、实验中变量的控制如图 2 所示，其中，网络都设置了 mini-batch 小块下降的方法，从图中可以看出，在其他条件不变的情况下，网络隐藏层数越多，网络收敛的速度相对较慢，但是最终收敛的结果比较好，尤其是当网络节点数量超过了训练数据的数量，拟合网络会出现问题。所以网络中间隐藏层数量需要根据实际任务设计，如果太大的话会出现问题。

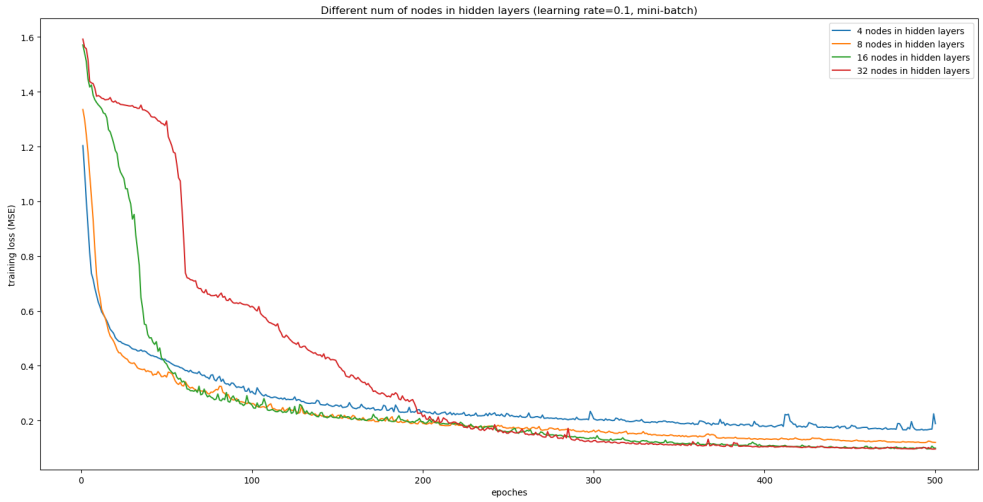

图 2: 隐含层不同结点数目对训练精度的影响

(b) 观察不同的梯度更新步长对训练的影响，根据实验，实验结果和实验中变量控制都如图 3 所示，其中训练步长如果设置的太大的话，会引起震动，这样会影响训练的收敛结果，但是又是训练步长设置的太小的话，会影响训练的收敛的速度，收敛速度可能会比较的慢。

训练步长需要在合适的区间内选择，步长太大太小，都会有问题，在优化理论中，优化目标函数，我们通过梯度下降找到了相关的迭代方向，但是对于迭代步长有多种的选择，有很多种的不同的方法来确定步长。可以把神经网络看成一个非线性的函数，对于训练优化的目标函数，通过梯度下降法确定方向，但是对于步长需要合适的设计，有很多的设计的方法，可以借鉴优化中的一些方法，比如说黄金分割法、裴波那契分割法，还有其他的一些方法，这一块现在依旧在不断的研究。

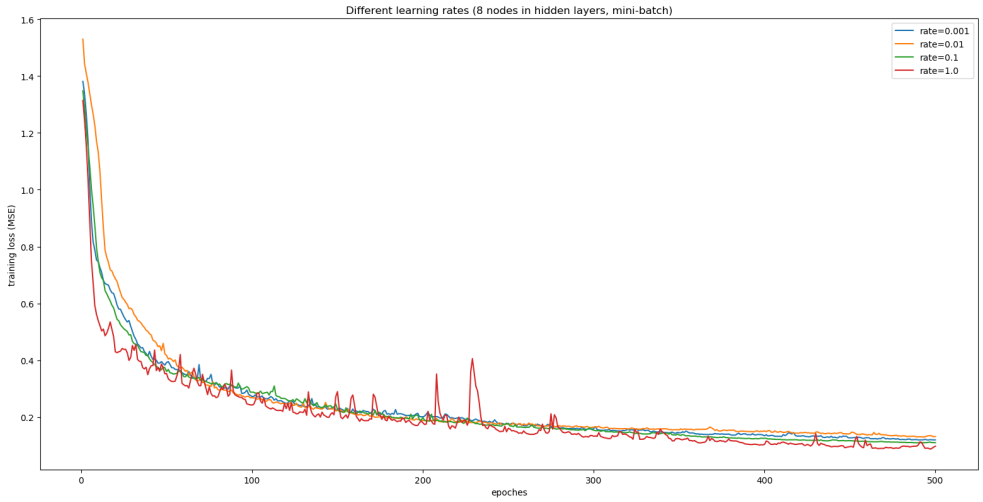

图 3: 不同的梯度更新步长对训练的影响

(c) 图 4 和图 5 分别绘制两种方法的目标函数随着迭代步数增加的变化曲线。这里将整个数据集作为训练集，因为数据集实在是太小了，这里不考虑训练集和验证集了。一个采用批量方式更新权重，另一个采用单样本方式更新权重。目标函数采用平方误差准则函数，这里为了方便统一，展示使用的是平均平方误差准则函数的值，也就是 mean square error，MSE（在网络训练的时候，目标函数采用的是平方误差准则函数，这里真实的是使用的平均值的）。

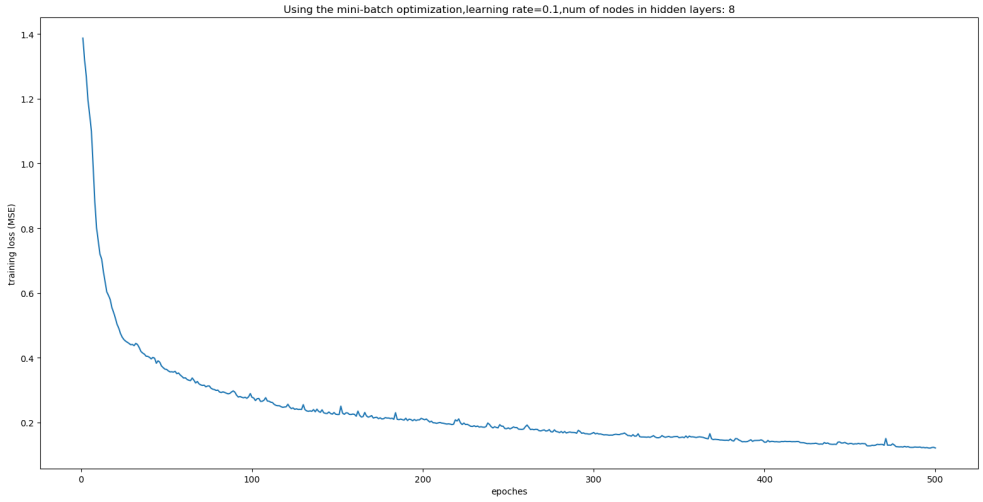

图 4: 采用批量方式更新权重

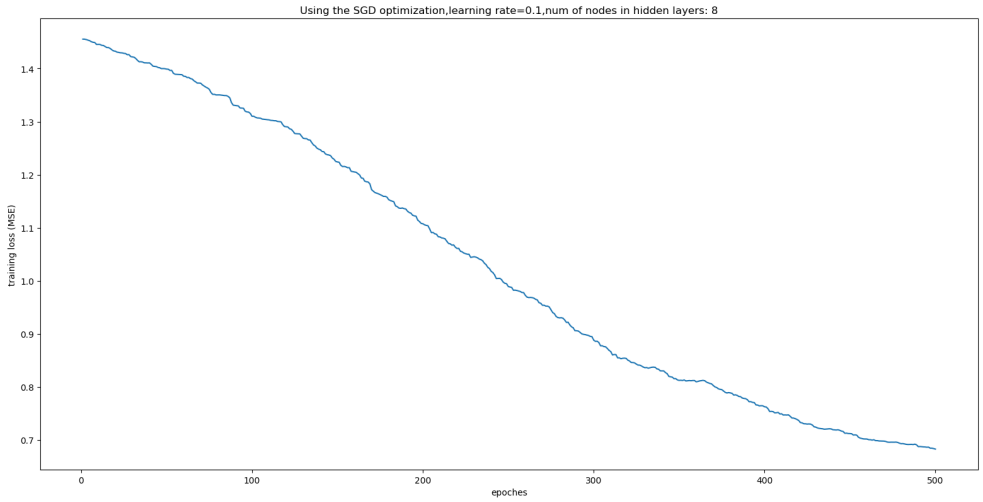

图 5: 采用单样本方式更新权重