Answer of Assignment 4

第 6 章：神经网络

2020E8017782032_ 蒲尧

第一部分：计算与证明

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:

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

Where net_j stands for the weighted sum at the j-th node in the output layer.
Please 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）:

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

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

2. 请对反向传播算法的训练步骤进行总结；结合三层网络给出不超过三个有关权重更新的公式，并用文字描述所述公式的含义；指出哪些因素会对网络的性能产生影响。

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。

第一部分回答

1. 推到如下：令对隐藏层节点的激励函数 sigmoid 为 f(x)，对输出层节点的激励函数 softmax 为 g(x)

$$
\begin{aligned}
f(s) &= sigmoid(s) = \frac{1}{1+e^{-s}} \\
f'(s) &= \frac{e^{-s}}{(1+e^{-s})^2} = \frac{1}{1+e^{-s}}\left(1-\frac{1}{1+e^{-s}}\right) = y(1-y) \\
g(net_j) &= z_j = softmax(net_j) = \frac{e^{net_j}}{\sum_{m=1}^{c} e^{net_m}} \\
g'(net_j) &= \frac{\partial g(net_j)}{\partial net_m} = \frac{\frac{\partial e^{net_j}}{\partial net_m}\sum - \frac{\partial \sum}{\partial net_m} e^{net_j}}{\sum^2} \\
&= \frac{e^{net_j}\sum - e^{net_j}e^{net_m}}{\sum^2} \; OR \; \frac{0\sum - e^{net_j}e^{net_m}}{\sum^2} \\
&= \frac{e^{net_j}}{\sum}\frac{\sum - e^{net_m}}{\sum} \; OR \; -\frac{e^{net_j}}{\sum}\frac{e^{net_m}}{\sum} \\
&= z_j(1-z_j) \; OR \; -z_j z_m \\
&= z_j(\delta_m - z_m) \\
Where\ \delta_m &= \begin{cases} 1, & j=m; \\ 0, & j \neq m \end{cases} \\
J(w)^k &= \frac{1}{2}\sum_{j=1}^{c}\left(t_j^k - z_j^k\right)^2 \\
&= \frac{1}{2}\sum_{j=1}^{c}\left(t_j^k - g\left(net_j^k\right)\right)^2 \\
&= \frac{1}{2}\sum_{j=1}^{c}\left(t_j^k - g\left(\sum_{h=1}^{n_H} w_{hj} y_h^k\right)\right)^2 \\
&= \frac{1}{2}\sum_{j=1}^{c}\left(t_j^k - g\left(\sum_{h=1}^{n_H} w_{hj} f\left(net_h^k\right)\right)\right)^2 \\
&= \frac{1}{2}\sum_{j=1}^{c}\left(t_j^k - g\left(\sum_{h=1}^{n_H} w_{hj} f\left(\sum_{i=1}^{d} w_{ih} x_i^k\right)\right)\right)^2
\end{aligned}
$$

隐含层到输出层

$$
\begin{aligned}
\Delta w_{hj} &= -\eta\frac{\partial J}{\partial w_{hj}} = -\eta\sum_k \frac{\partial J}{\partial z_j^k}\frac{\partial z_j^k}{\partial net_j^k}\frac{\partial net_j^k}{\partial w_{hj}} \\
&= \eta\sum_k\left(t_j^k - z_j^k\right)g'\left(net_j^k\right)y_h^k \\
&= \eta\sum_k \delta_j^k y_h^k \\
Where \quad \delta_j^k &= \frac{\partial J}{\partial net_j^k} = \left(t_j^k - z_j^k\right)g'\left(net_j^k\right) = \Delta_j^k z_j^k(\delta_m - z_m) = \left(t_j^k - z_j^k\right)z_j^k(\delta_m - z_m)
\end{aligned}
$$

Single sample contribution:

$$\Delta w_{hj}^k = \eta \delta_j^k y_h^k = \eta \underbrace{\left(t_j^k - z_j^k\right) z_j^k \left(\delta_m - z_m\right)}_{\delta_j^k} y_h^k$$

输入层到隐含层

$$\begin{aligned}
\Delta w_{ih} &= -\eta \frac{\partial J}{\partial w_{ih}} \\
&= -\eta \sum_{k,j} \frac{\partial J}{\partial z_j^k} \frac{\partial z_j^k}{\partial net_j^k} \frac{\partial net_j^k}{\partial y_j^k} \frac{\partial y_j^k}{\partial net_h^k} \frac{\partial net_h^k}{\partial w_{ih}} \\
&= \eta \sum_{k,j} \left(t_j^k - z_j^k\right) g'\left(net_j^k\right) w_{hj} f'\left(net_h^k\right) x_i^k \\
&= \eta \sum_{k,j} \delta_j^k w_{hj} f'\left(net_h^k\right) x_i^k \\
&= \eta \sum_{k} \left[f'\left(net_h^k\right) \sum_j \delta_j^k w_{hj} \right] x_i^k \\
&= \eta \sum_{k} \delta_h^k x_i^k
\end{aligned}$$

$$Where \quad \delta_h^k = \frac{\partial J}{\partial net_h^k} = f'\left(net_h^k\right) \sum_j \delta_j^k w_{hj} = f'\left(net_h^k\right) \Delta_h^k = \underbrace{y_h^k \left(1 - y_h^k\right)}_{f'\left(net_h^k\right)} \sum_j w_{hj} \underbrace{\left(t_j^k - z_j^k\right) z_j^k \left(\delta_m - z_m\right)}_{\delta_j^k}$$

Single sample contribution:

$$\Delta w_{ih}^k = \eta \delta_h^k x_i^k = \eta \underbrace{\underbrace{y_h^k \left(1 - y_h^k\right)}_{f'\left(net_h^k\right)} \sum_j w_{hj} \underbrace{\left(t_j^k - z_j^k\right) z_j^k \left(\delta_m - z_m\right)}_{\delta_j^k}}_{\delta_h^k} x_i^k$$

2. (1) 步骤：

Step1: 将训练集数据输入到输入层，经过隐藏层，最后达到输出层并输出结果，这是 ANN 的前向传播过程；

Step2: 由于 ANN 的输出结果与实际结果有误差，则计算估计值与实际值之间的误差，并将该误差从输出层向隐藏层反向传播，直至传播到输入层；

Step3: 在反向传播的过程中，根据误差调整各级网络节点之间的权值 $w_{ih1}, w_{h1h2}, ..., w_{h(last)j}$；不断迭代上述过程，直至收敛或梯度下降小于某阈值。

(2) 权重更新公式：

$$w_{hj} = w_{hj} + \eta \delta_j^k y_h^k$$

$$w_{ih} = w_{ih} + \eta \delta_h^k x_i^k$$

w_{hj} 隐含层和输出层节点权值，η 学习率，δ_j^k 第 k 个样本对输出层第 j 个输出应该更新权重的贡献，y_h^k 隐含层的第 h 个输出；

w_{ih} 输入层和隐含层节点权值，η 学习率，δ_h^k 第 k 个样本对隐含层第 h 个输出应该更新权重的贡献，x_i^k 输入层的第 i 个输出。

具体算法见第一题。

(3) 会对网络性能产生影响的因素

准则函数；激励函数；隐含层数；隐含层节点个数；初始权重；正则化技术；学习率；停止训练准则；等等。

第二部分回答

1. 主要代码

```
def forward(self, x_i):
    """
    Parameter:
        x_i: single sample
    Return:
        y_h: outputs of hidden-layer for single sample
        z_j: outputs of output-layer for single sample
    """
    # net_h = np.matmul(x_i.T, w_ih)
    net_h = x_i.T @ self.w_ih
    y_h = self.tahn_func(net_h)
    # net_j = np.matmul(y_h.T, w_hj)
    net_j = y_h.T @ self.w_hj
    z_j = self.sigmoid_func(net_j)
    return y_h, z_j

def backward(self, z_j, y_h, x_i, t, eta):
    """
    Parameters:
        z_j: outputs of output-layer for single sample
        y_h: outputs of hidden-layer for single sample
        x_i: inputs of input-layer for single sample
        t: target's label
        eta: learn rate
    Return:
        Delta_w_hj: hidden-layer-output-layer weight updating matrix
        Delta_w_ih: input-layer-hidden-layer weight updating matrix
        error: sample square error for ploting
```

```
    """
    # reshape the dimension of matrix (3, 1) (n_h, 1) (3, 1) (3, 1)
    z_j = np.reshape(z_j, (z_j.shape[0], 1))
    y_h = np.reshape(y_h, (y_h.shape[0], 1))
    x_i = np.reshape(x_i, (x_i.shape[0], 1))
    t = np.reshape(t, (t.shape[0], 1))
    # The following matrixs' shape: (1, 1)(3, 1)(n_h, 3)(n_h, 1)(3, n_h)
    # output error
    error = ((t - z_j).T @ (t - z_j))[0][0]
    # sigmoid'(s) = sigmoid(s) * (1-sigmoid(s)) = z_j * (1-z_j)
    delta_j = (t - z_j) * z_j * (1-z_j)
    Delta_w_hj = eta * (y_h @ delta_j.T)
    # tanh'(s) = 1-tanh(s)^2 = 1-y_h**2
    delta_h = (((t - z_j) * z_j * (1-z_j)).T @ self.w_hj.T).T * (1-y_h**2)
    Delta_w_ih = eta * (x_i @ delta_h.T)
    return Delta_w_hj, Delta_w_ih, error

def train(self, bk_type, eta, epoch_num):
    """
    Parameters:
        bk_type: 'single' or 'batch'
        eta: learn rate
        epoch_num: maximum iretation number
    Return:
        w_ih:input-layer-hidden-layer weight finally matrix
        w_hj:hidden-layer-output-layer weight finally matrix
    """
    # Stochastic Backpropagation
    if bk_type == 'single':
        E = []
        for _ in range(epoch_num):
            e = []
            for idx, x_i in enumerate(self.train_data):
                # forward
                y_h, z_j = self.forward(x_i)
                # backward
                Delta_w_hj, Delta_w_ih, error = self.backward(
                    z_j, y_h, x_i, self.train_label[idx], eta)
```

```
                # weight update each sample
                self.w_hj += Delta_w_hj
                self.w_ih += Delta_w_ih
                e.append(error)
            E.append(np.mean(e))

    # Batch Backpropagation
    if bk_type == 'batch':
        E = []
        for _ in range(epoch_num):
            e = []
            Batch_Delta_w_hj = 0
            Batch_Delta_w_ih = 0
            for idx, x_i in enumerate(self.train_data):
                # forward
                y_h, z_j = self.forward(x_i)
                # backward
                Delta_w_hj, Delta_w_ih, error = self.backward(
                    z_j, y_h, x_i, self.train_label[idx], eta)
                # template weight matrix update each sample
                Batch_Delta_w_hj += Delta_w_hj
                Batch_Delta_w_ih += Delta_w_ih
                e.append(error)
            # weight matrix update each iretation
            self.w_hj += Batch_Delta_w_hj
            self.w_ih += Batch_Delta_w_ih
            E.append(np.mean(e))
```

2. (a) 隐含层不同结点数目对训练精度的影响，节点数 n_h 越大，收敛越快，相同迭代步数情况下，误差越小。

   (b) 梯度更新步长较小时，误差变化较小，图线较光滑，越大变化越大，图线不光滑。

   (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如下图。

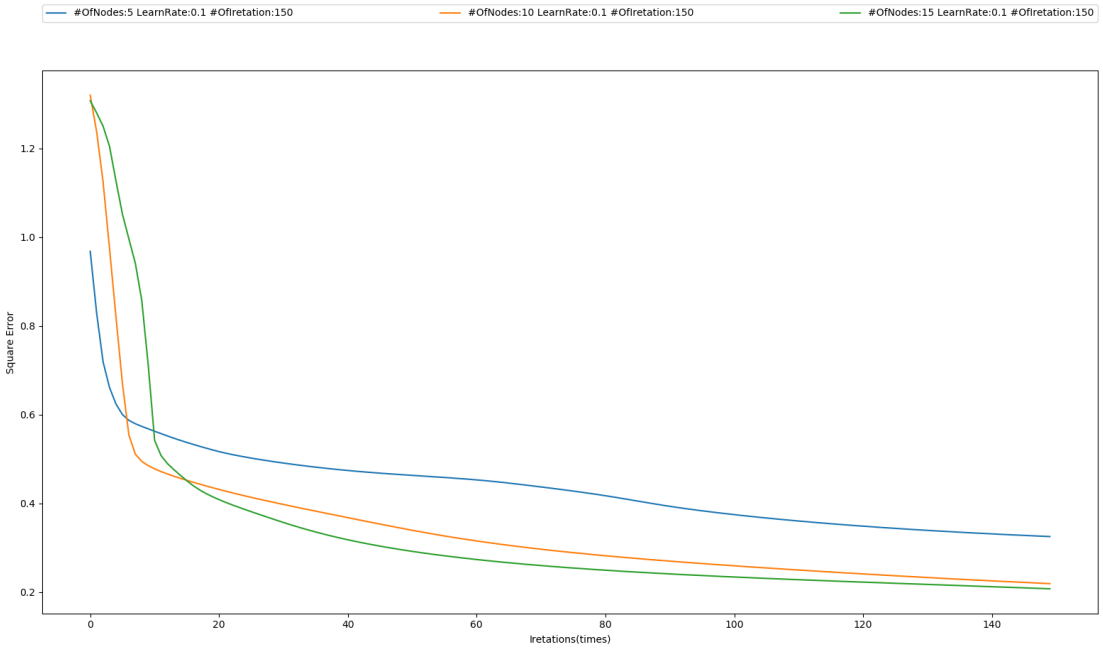

图 1: 隐含层结点数目 n_h 变化对 $Error$ 影响对比图

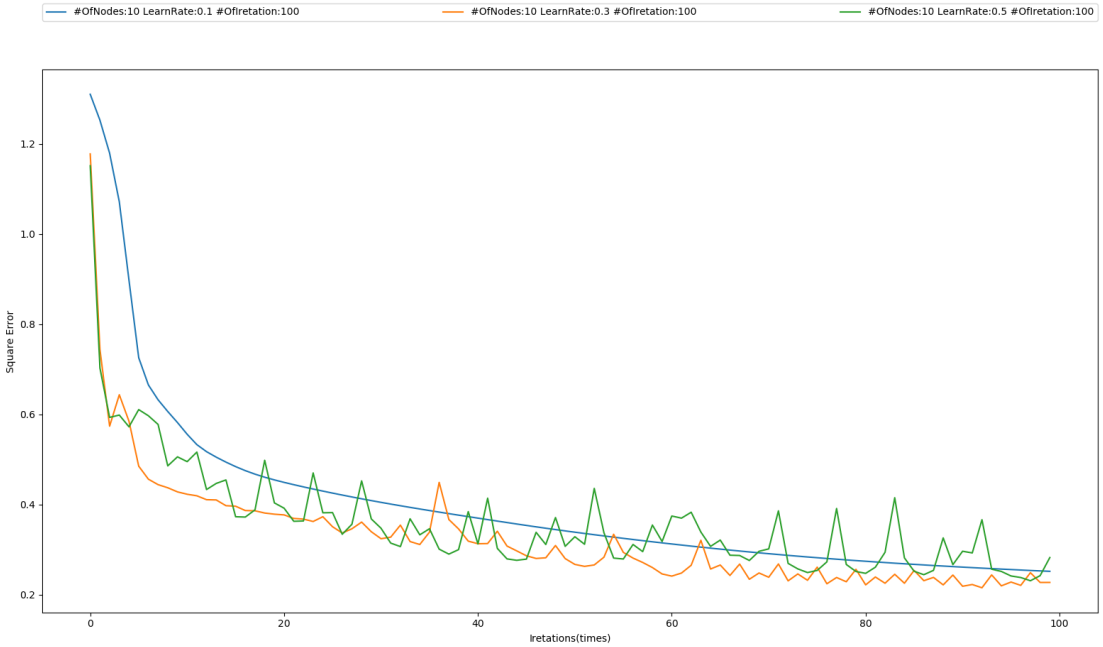

图 2: 学习率 η 变化对 $Error$ 影响对比图

代码见如下文件 Python 文件