> 参考：
>
> https://blog.csdn.net/fsfjdtpzus/article/details/106256925

1 目标

学习《神经网络与深度学习》第二章反向传播，并在第一章练习的基础上，将训练集掺入我们自己的数据进行训练，看看泛化效果如何。例如在其中几个epoch中，将训练数据指定为我们的数据（随机选择），测试结果是否会更好。

2 添加训练集

切割数据集的时候，设定训练集的比例

```
def array_split(array_data,label_data,train_data_ratio):
    combined = list(zip(array_data, label_data))
    random.shuffle(combined)
    array_data[:], label_data[:] = zip(*combined)

    train_size = int(len(array_data) * train_data_ratio)
    test_size = len(array_data) - train_size

    # 提取训练集和测试集
    train_set_array, train_set_label = array_data[:train_size], label_data[:train_size]
    test_set_array, test_set_label = array_data[train_size:], label_data[train_size:]

    return train_set_array, train_set_label,test_set_array, test_set_label
```

在第j个epoch中，利用j来求余，调整添加的频率

```
if j % 1 == 0:
    mini_batches_mydata_tarin = [
    mydata_tarin[k:k+mini_batch_size]
    for k in range(0, n_mydata_tarin, mini_batch_size)]

    if len(mini_batches_mydata_tarin) != 0:
        mini_batches = mini_batches + mini_batches_mydata_tarin
```

设定为1，即为每个epoch都添加。mini_batch_size设为5，学习率为1.5，9的倍数的epoch添加自己的训练集，结果为18%。

```
6 : 0.0%
5 : 50.0%
9 : 0.0%
7 : 100.0%
8 : 100.0%
4 : 0.0%
2 : 25.0%
1 : 0.0%
0 : 0.0%
3 : 0.0%
Epoch 28 : 4 / 22  18.18%  18.18%
```

mini_batch_size设为10，学习率为3.0，1的倍数的epoch添加自己的训练集，结果为23%。

```
8 : 50.0%
7 : 20.0%
1 : 0.0%
5 : 0.0%
3 : 0.0%
2 : 0.0%
0 : 50.0%
6 : 0.0%
4 : 50.0%
Epoch 22 : 5 / 22  22.73%  22.73%
```

在上述参数下，将数据集分割的比例改为0.2，结果为27%。

在上述参数下，将数据集分割的比例改为0，也就是不掺入自己的数据，结果为27%。

有可能是掺入的数据污染了原始训练集，所以自己的数据加的越多，结果越差。

3 反向传播

3.1 四个方程

总结：反向传播的四个方程式

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{BP2}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{BP3}$$

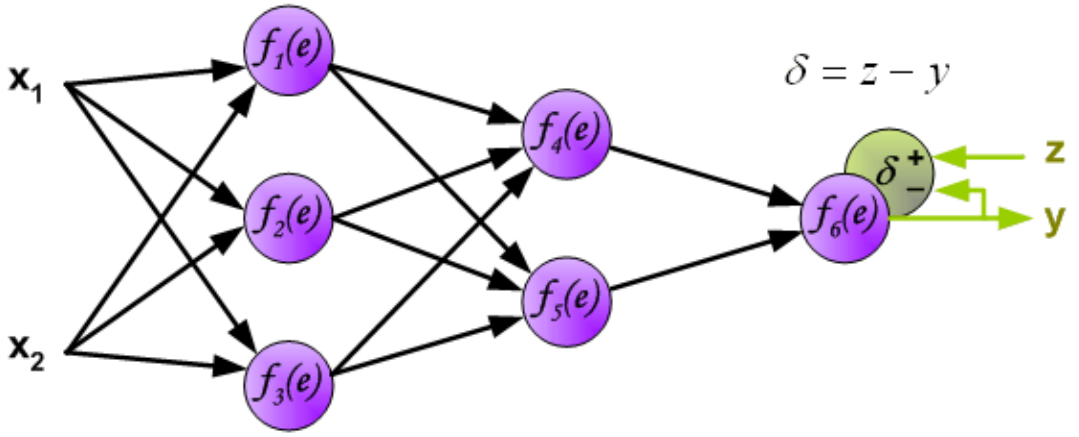
$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \tag{BP4}$$

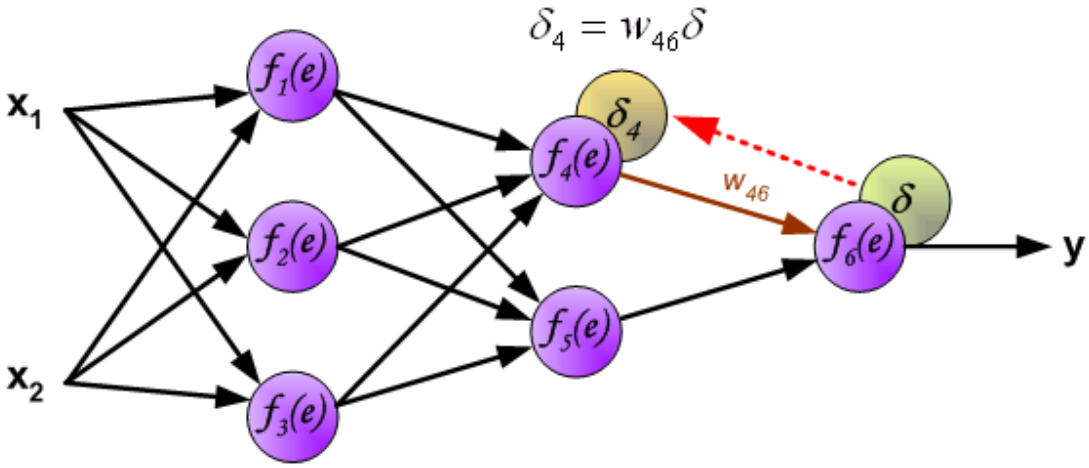
1. BP1：输出层误差的方程
2. BP2：使用下一层的误差来表示当前层的误差。
    1. 通过组合 (BP1) 和 (BP2)，我们可以计算任何层的误差 δl。首先使用 (BP1) 计算 δL，然后应用方程 (BP2) 来计算 δL−1，然后再次用方程 (BP2) 来计算 δL−2，如此一步一步地反向传播完整个网络。
3. BP3：代价函数关于网络中任意偏置的改变率

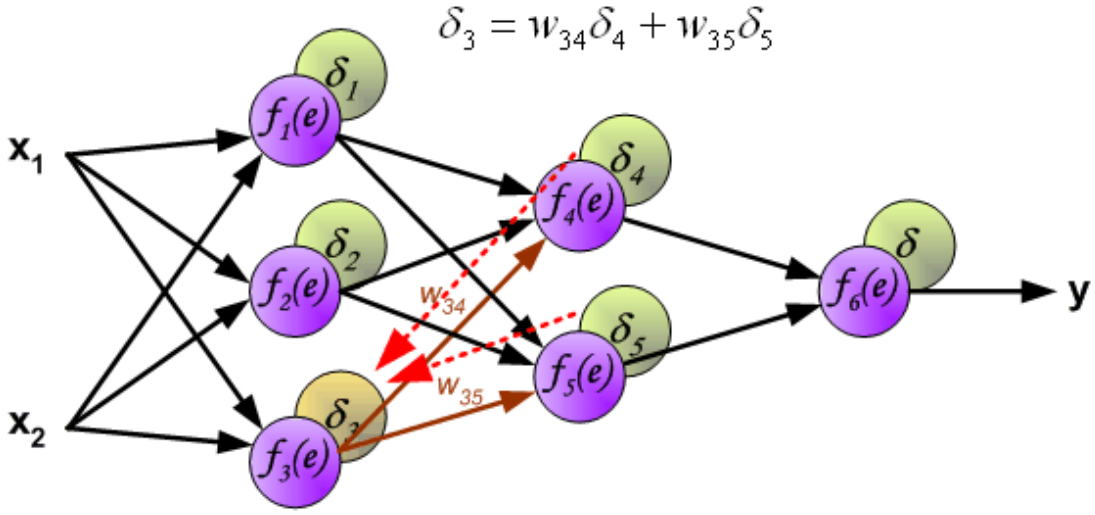
4. BP4：代价函数关于任何一个权重的改变率

3.2 计算误差

1. 计算神经网络的输出(预测值)和真值的误差。
2. 计算完误差后，需要将这个误差向不断的向前一层传播。**向前一层传播时，需要考虑到前一个神经元的权重系数**(因为不同神经元的重要性不同，因此回传时需要考虑权重系数)。
3. 与前向传播时相同，**反向传播时后一层的节点会与前一层的多个节点相连，因此需要对所有节点的误差求和。**
4. 计算出每个神经元的误差，接下来就更新权重。

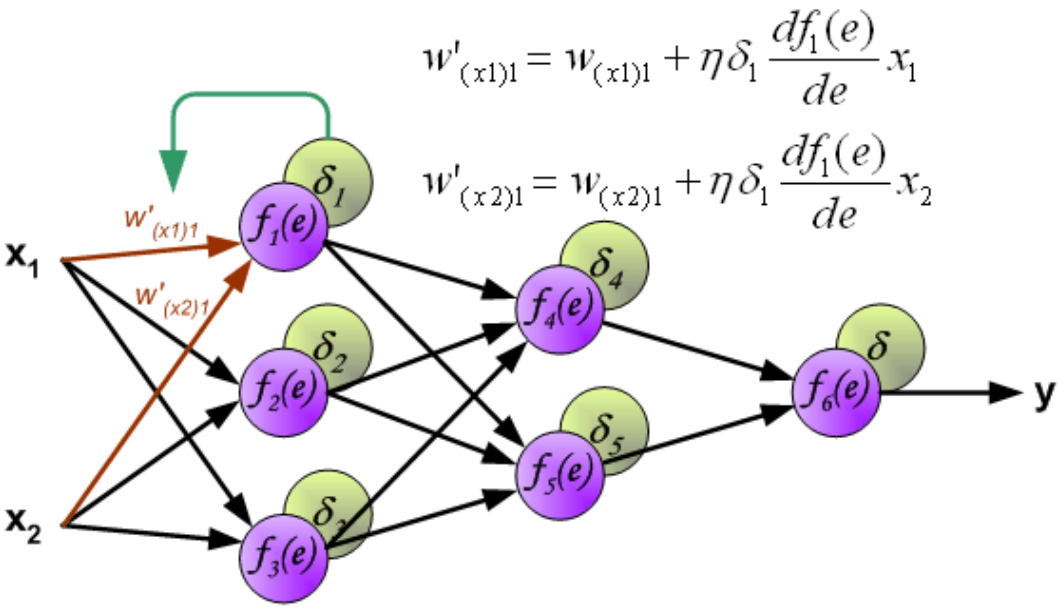






3.3 更新权重

η代表学习率，w'是更新后的权重，通过这个式子来更新权重。



计算好误差，并且更新权重，反向传播就结束了。

将这个过程不断重复，就可以不断减小误差，提高正确率，获得比较好的模型了。