搭建一个神经网络

这一部分会不用PyTorch搭建一个神经网络并训练它。但是为了代码的简洁，我们会用到Numpy模块用于基本的矩阵计算。

数据集

正如之前说的，我们直接导入一个数据集(本文用Pendigits数据集为例)

In [5]:
```
import pickle
import os
datasets = os.listdir('./others/datasets/')
datasets = [f for f in datasets if (f.startswith('Dataset') and f.endswith('.p'))]
datasets.sort()
datapath = os.path.join(f'./others/datasets/{datasets[20]}')
with open(datapath, 'rb') as f:
    data = pickle.load(f)
X_train    = data['X_train']
y_train    = data['y_train']
X_valid    = data['X_valid']
y_valid    = data['y_valid']
X_test     = data['X_test']
y_test     = data['y_test']
data_name  = data['name']

N_class    = data['n_class']
N_feature  = data['n_feature']
N_train    = X_train.shape[0]
N_valid    = X_valid.shape[0]
N_test     = X_test.shape[0]

print(f'Dataset "{data_name}" has {N_feature} input features and {N_class} classes.\nThere are {N_train} training examp
```

```
Dataset "Pendigits" has 16 input features and 10 classes.
There are 6595 training examples, 2198 valid examples, and 2199 test examples in the dataset.
```

定义基本计算

加权求和：为了完善的展现反向传播的计算，我们这里不把权重W和偏差b合并，然后再把X扩展一列。

In [6]:
```
def LinearForward(A, W, b):
    return np.dot(W, A) + b
```

激活函数：这里定义了2种激活函数

In [7]:
```
def Activation(Z, activation):
    if activation == 'relu':
        A = np.maximum(Z, 0)
    if activation == 'sigmoid':
        A = 1 / (1 + np.exp(-Z))
    return A
```