实现一个手写数字识别的算法(使用神经网络算法)

MNIST数据集:

训练(train)	50,000
验证(validation)	10,000
测试(test)	10,000

```
#神经网络
class Network(object):
def __init__(self, sizes):
        self.num_layers = len(sizes)
        self.sizes = sizes
        self.biases = [np.random.randn(y, 1) for y in sizes[1:]]
        self.weights = [np.random.randn(y, x)
                        for x, y in zip(sizes[:-1], sizes[1:])]
```

sizes: 每层神经元的个数, 例如: 第一层2个神经元,第二层3个神经元:

> ```
> net = Network([2, 3, 1]);
> ```

`np.random.rand(y, 1)`: 随机从正态分布(均值0, 方差1)中生成 `net.weights[1]`: 存储连接第二层和第三层的权重 (Python索引从0开始数)

$${\alpha}'=\sigma(\omega *\alpha + b)$$

> 正向传播

```
    def feedforward(self, a):
        """Return the output of the network if "a" is input."""
        for b, w in zip(self.biases, self.weights): zip函数把两个矩阵（实际上是多个维度不同的向量组成）组合
            a = sigmoid(np.dot(w, a)+b)
        return a
```

```
#随机梯度下降算法
def SGD(self, training_data list形式, epochs, mini_batch_size 每次用于梯度下降的实例的大小, eta 学习率,
            test_data=None):
```

```
        """Train the neural network using mini-batch stochastic
        gradient descent.  The "training_data" is a list of tuples
        "(x 输入特征向量, y x数据所属label)" representing the training inputs and
the desired
        outputs.  The other non-optional parameters are
        self-explanatory.  If "test_data" is provided then the
        network will be evaluated against the test data after each
        epoch, and partial progress printed out.  This is useful for
        tracking progress, but slows things down substantially."""
        if test_data: n_test = len(test_data)
        n = len(training_data)
        for j in xrange(epochs):
            random.shuffle(training_data)
            mini_batches = [
                training_data[k:k+mini_batch_size]
                for k in xrange(0, n, mini_batch_size)]
            for mini_batch in mini_batches:
                self.update_mini_batch(mini_batch, eta)
            if test_data:
                print "Epoch {0}: {1} / {2}".format(
                    j, self.evaluate(test_data), n_test)
            else:
                print "Epoch {0} complete".format(j)
```

Backprogation:反向传播

```

        #随机梯度下降更新公式

        # 反向传播代码
        def update_mini_batch(self, mini_batch, eta):
        """Update the network's weights and biases by applying
        gradient descent using backpropagation to a single mini batch.
        The "mini_batch" is a list of tuples "(x, y)", and "eta"
        is the learning rate."""
        nabla_b = [np.zeros(b.shape) for b in self.biases]
        nabla_w = [np.zeros(w.shape) for w in self.weights]
        for x, y in mini_batch:
            delta_nabla_b, delta_nabla_w = self.backprop(x, y)
            nabla_b = [nb+dnb for nb, dnb in zip(nabla_b, delta_nabla_b)]
            nabla_w = [nw+dnw for nw, dnw in zip(nabla_w, delta_nabla_w)]
        self.weights = [w-(eta/len(mini_batch))*nw
                        for w, nw in zip(self.weights, nabla_w)]
        self.biases = [b-(eta/len(mini_batch))*nb
                       for b, nb in zip(self.biases, nabla_b)]
```

手写识别的demo

####实现一个手写数字识别程序

> 1.加载mnist数据集 2.查看下返回的训练集、测试集和验证集的数据类型和长度（数据结构搞清楚） 3.创建神经网络 4.进行梯度下降，获得优化后的神经网络模型参数 5.对算法进行准确性评估

```
#coding=utf-8

import mnist_loader
from network import Network

trainDataset,validationDataset,testDataset = mnist_loader.load_data_wrapper()
# 训练集是一个50000长度的list，每个元素是一个元祖(x,y),x表示输入特征(768),y表示所属数字label
# print(len(trainDataset))
# print(len(trainDataset[0]))
# # x的结构
# print(trainDataset[0][0].shape)
# # y的结构
# print(trainDataset[0][1].shape)

# list里面是每层神经网络的神经数量
network = Network([784,50,10])
# 进行梯度下降，并测试
network.SGD(trainDataset,30,10,3.0,test_data=testDataset)
```