exp 7 - 差分隐私实验

神经网络

初始化

```python
def __init__(self, inputnodes, hiddennodes, outputnodes, learningrate):
    self.inodes = inputnodes
    self.hnodes = hiddennodes
    self.onodes = outputnodes
    self.wih = numpy.random.normal(0.0,pow(self.hnodes, -0.5),(self.hnodes, self.inodes)) #weight from input to hidden
    self.who = numpy.random.normal(0.0,pow(self.onodes, -0.5),(self.onodes, self.hnodes)) #weight from hidden to output
    self.lr = learningrate
    self.activation_function = lambda x: scipy.special.expit(x) #sigmoid(x)
```

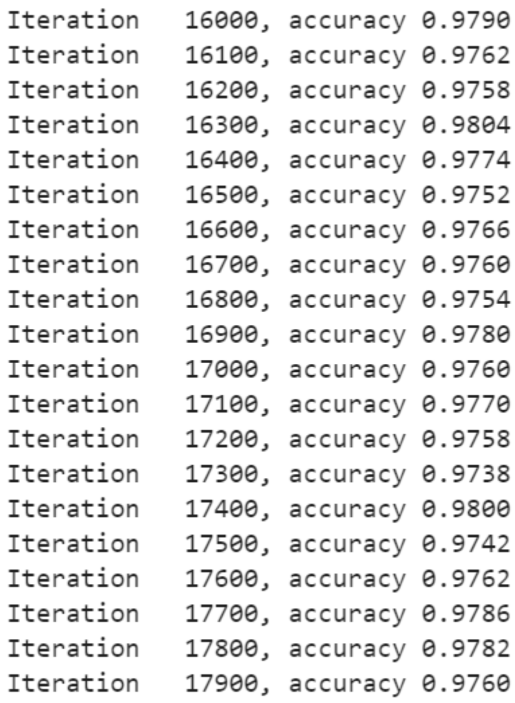
训练

```python
def train(self, inputs_list, targets_list):
    inputs = numpy.array(inputs_list, ndmin=2).T
    targets = numpy.array(targets_list, ndmin=2).T
    hidden_inputs = numpy.dot(self.wih, inputs)
    hidden_outputs = self.activation_function(hidden_inputs)
    final_inputs = numpy.dot(self.who, hidden_outputs)
    final_outputs = self.activation_function(final_inputs)
    output_errors = targets - final_outputs
    hidden_errors = numpy.dot(self.who.T, output_errors)
    # 反向传播
    self.who += self.lr * numpy.dot((output_errors*final_outputs*(1.0-final_outputs)),numpy.transpose(hidden_outputs))
    self.wih += self.lr * numpy.dot((hidden_errors*hidden_outputs*(1.0-hidden_outputs)), numpy.transpose(inputs))
```

预测

```python
def query(self, inputs_list):
    inputs = numpy.array(inputs_list, ndmin=2).T
    hidden_inputs = numpy.dot(self.wih, inputs)
    hidden_outputs = self.activation_function(hidden_inputs)
    final_inputs = numpy.dot(self.who, hidden_outputs)
    final_outputs = self.activation_function(final_inputs)
    return final_outputs
```

实验结果

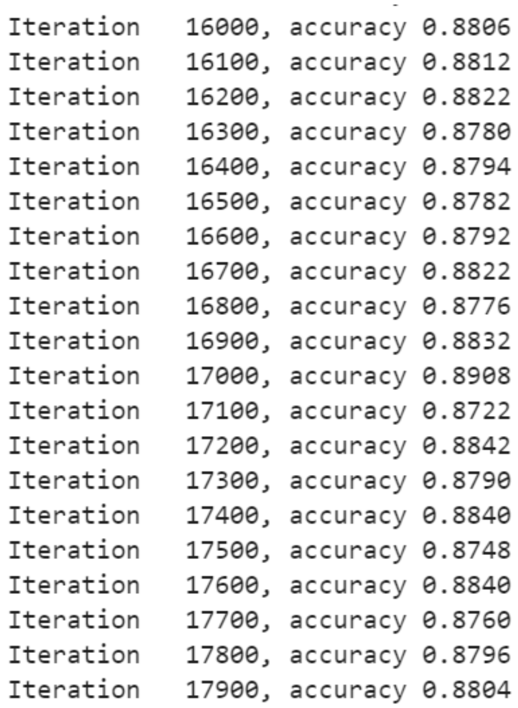


```
Iteration   16000, accuracy 0.9790
Iteration   16100, accuracy 0.9762
Iteration   16200, accuracy 0.9758
Iteration   16300, accuracy 0.9804
Iteration   16400, accuracy 0.9774
Iteration   16500, accuracy 0.9752
Iteration   16600, accuracy 0.9766
Iteration   16700, accuracy 0.9760
Iteration   16800, accuracy 0.9754
Iteration   16900, accuracy 0.9780
Iteration   17000, accuracy 0.9760
Iteration   17100, accuracy 0.9770
Iteration   17200, accuracy 0.9758
Iteration   17300, accuracy 0.9738
Iteration   17400, accuracy 0.9800
Iteration   17500, accuracy 0.9742
Iteration   17600, accuracy 0.9762
Iteration   17700, accuracy 0.9786
Iteration   17800, accuracy 0.9782
Iteration   17900, accuracy 0.9760
```

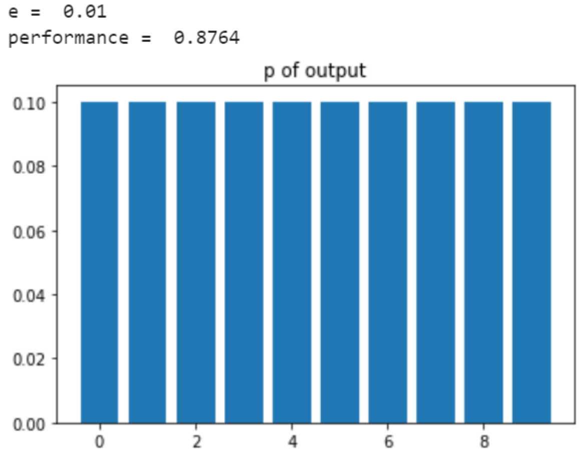
拉普拉斯噪声

给输出结果outputs添加拉普拉斯噪声：

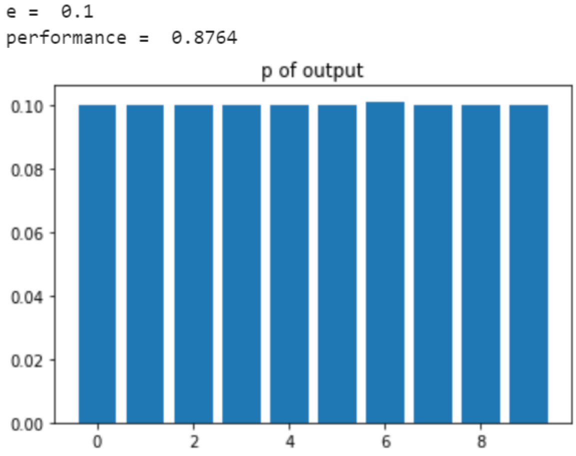
```
e = 1
b = 9/e
pred_ys = np.exp(-abs(np.subtract(to_numpy(model(to_tensor(xs))),np.argmax(ys, axis=1).reshape(5000,1)))/b)/(2*b)
acc = np.mean(np.argmax(ys, axis=1) == np.argmax(pred_ys, axis=1))
```



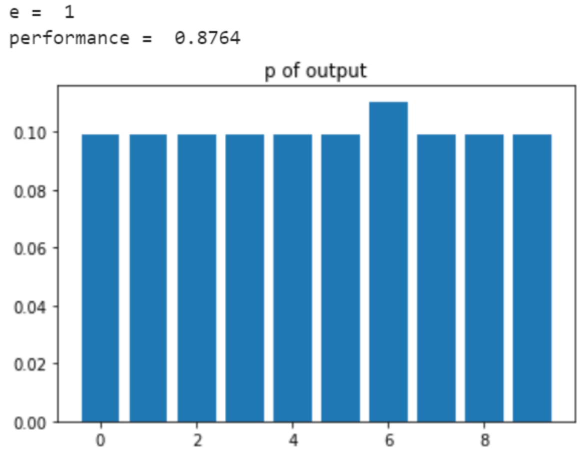
```
Iteration   16000, accuracy 0.8806
Iteration   16100, accuracy 0.8812
Iteration   16200, accuracy 0.8822
Iteration   16300, accuracy 0.8780
Iteration   16400, accuracy 0.8794
Iteration   16500, accuracy 0.8782
Iteration   16600, accuracy 0.8792
Iteration   16700, accuracy 0.8822
Iteration   16800, accuracy 0.8776
Iteration   16900, accuracy 0.8832
Iteration   17000, accuracy 0.8908
Iteration   17100, accuracy 0.8722
Iteration   17200, accuracy 0.8842
Iteration   17300, accuracy 0.8790
Iteration   17400, accuracy 0.8840
Iteration   17500, accuracy 0.8748
Iteration   17600, accuracy 0.8840
Iteration   17700, accuracy 0.8760
Iteration   17800, accuracy 0.8796
Iteration   17900, accuracy 0.8804
```



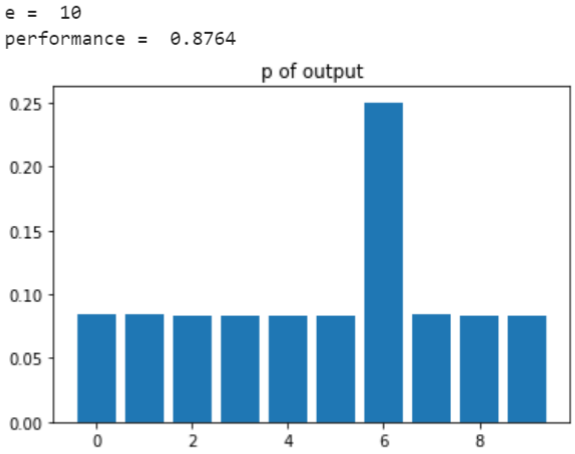
```
e =  0.01
performance =  0.8764
```



```
e =  0.1
performance =  0.8764
```



```
e =  1
performance =  0.8764
```



```
e =  10
performance =  0.8764
```

可见，添加噪声后，准确率降低到了88%左右。（条形图为自定义performance函数中按照最后一次预测值所绘制）

而且取到其他数值的概率随着e的升高的降低。而实验结果显示，只要添加了噪声，那么结果的准确性与e的大小无关,都在88%上下浮动，可能是由于mnist手写数据集结果为十个输出节点的最大值，故无论取到其他数值的概率如何，只要是同一个输出，那么当取最大值时，e不会改变概率的相对大小。