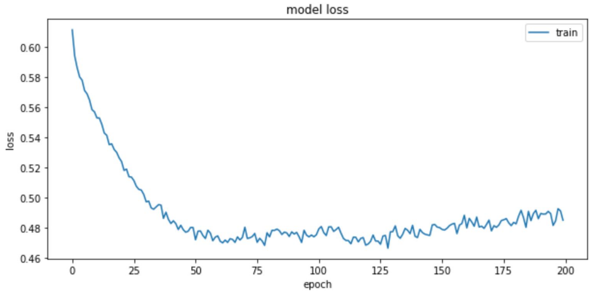
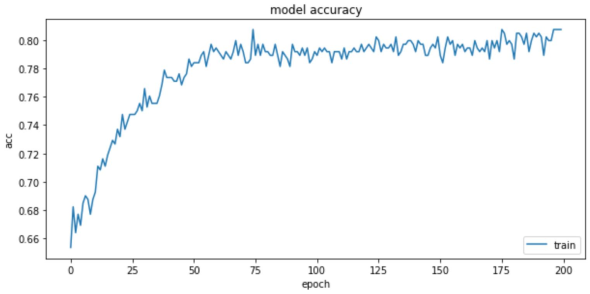
Update the pcn.py using Keras MLP model, and do the following tasks:

1.1 Demonstrate your Pima Results Here (Cycles, Accuracy)

1.2 Compare the batch training and sequential (single-instance) training.

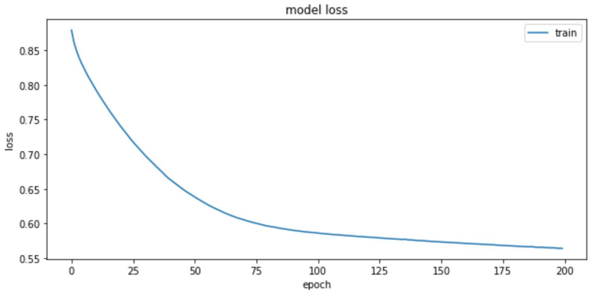
Epoch:200, Batch:1 (best acc:0.8, time=215.589s)

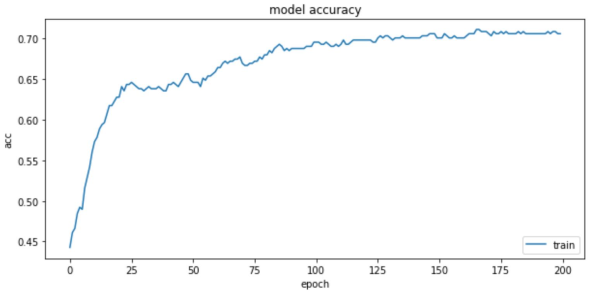




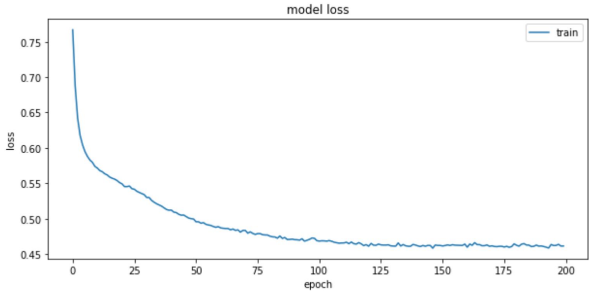
2. Compare the batch training and sequential (single-instance) training.

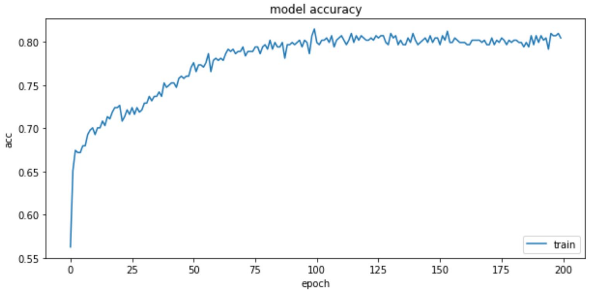
Epoch:200, Batch:256 (best acc:0.7, time=3.271s)



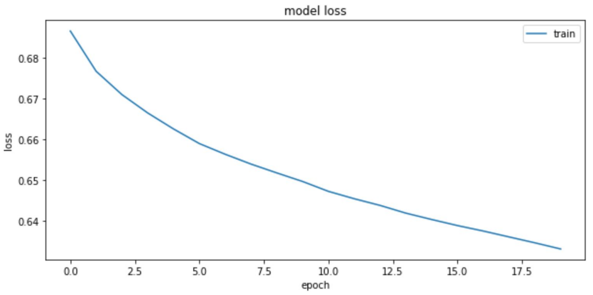


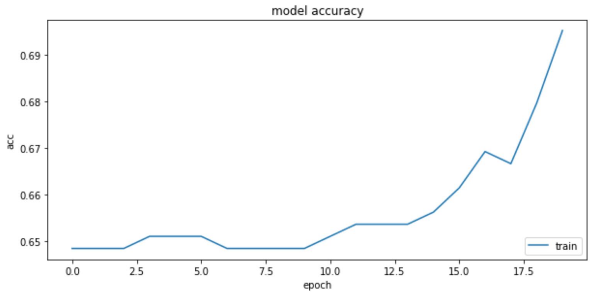
Epoch:200, Batch:2 (best acc:0.8, time=109.814s)

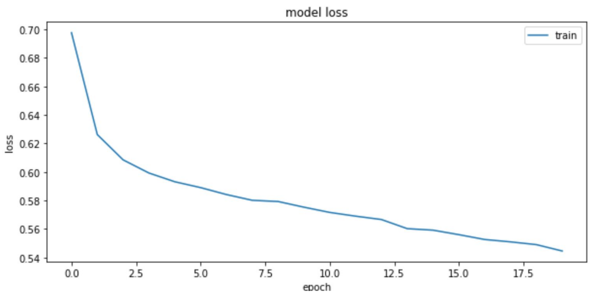


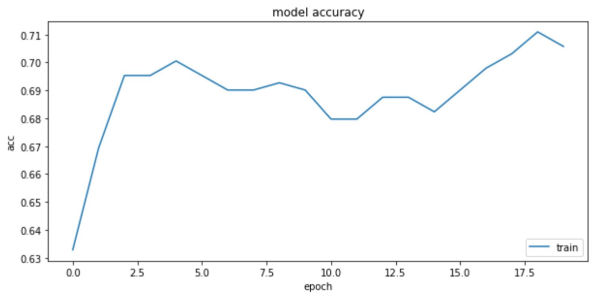


Epoch:20, Batch:256 (best acc:0.69, time=1.065s)

<table>
<tr>
<td>


</td>
<td>


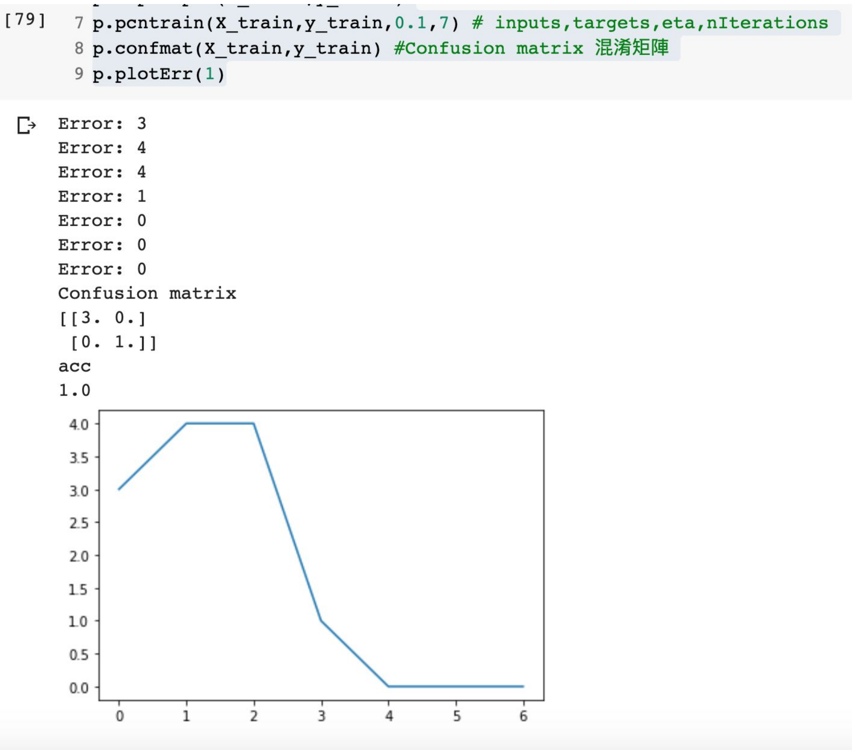
</td>
</tr>
<tr>
<td colspan="2">Epoch:20, Batch:2 (best acc:0.71, time=11.57s)</td>
</tr>
<tr>
<td>


</td>
<td>


</td>
</tr>
</table>

2. Run the pcn.py to the learning self-mapping problem (as shown in the following picture), and find the best result you can.

Eta = 0.1, Iter = 7



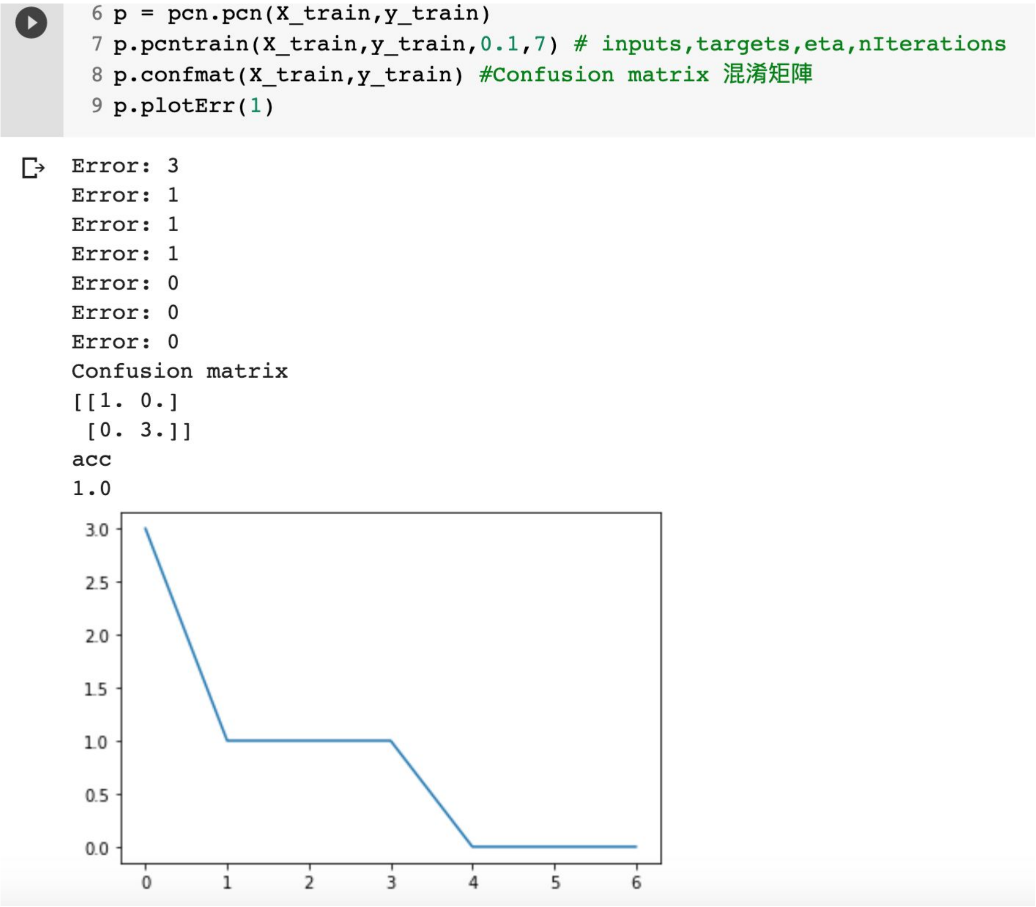
```
[79]
p.pcntrain(X_train,y_train,0.1,7) # inputs,targets,eta,nIterations
p.confmat(X_train,y_train) #Confusion matrix 混淆矩陣
p.plotErr(1)
```

```
Error: 3
Error: 4
Error: 4
Error: 1
Error: 0
Error: 0
Error: 0
Confusion matrix
[[3. 0.]
 [0. 1.]]
acc
1.0
```

3. Run the pcn.py to the learning OR mapping problem (as shown in the following picture), choose the parameters (e.g. epochs) to find the best result you can.

x1	x2	Output
0	0	0
0	1	1
1	0	1
1	1	1

Eta = 0.1, Iter = 7



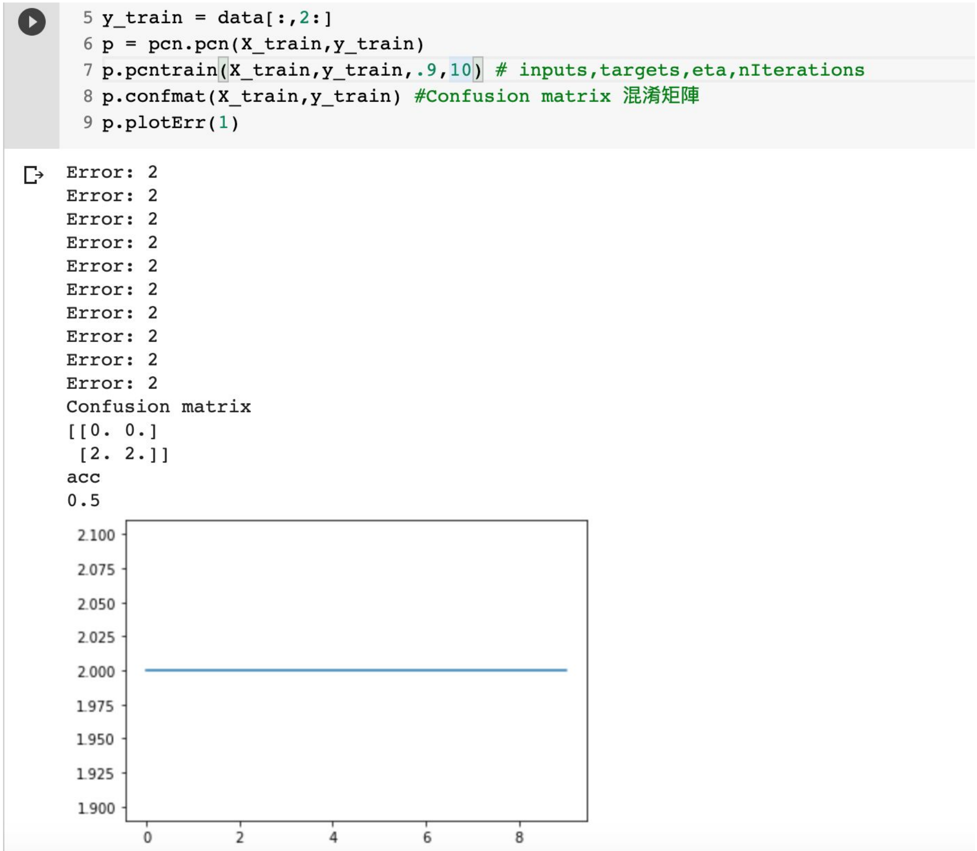
```
p = pcn.pcn(X_train,y_train)
p.pcntrain(X_train,y_train,0.1,7) # inputs,targets,eta,nIterations
p.confmat(X_train,y_train) #Confusion matrix 混淆矩陣
p.plotErr(1)
```

```
Error: 3
Error: 1
Error: 1
Error: 1
Error: 0
Error: 0
Error: 0
Confusion matrix
[[1. 0.]
 [0. 3.]]
acc
1.0
```

4. Run the pcn.py to the learning XOR mapping problem (as shown in the following picture), choose the parameters (e.g. epochs) to find the best result you can.

x1	x2	Output
0	0	0
0	1	1
1	0	1
1	1	0

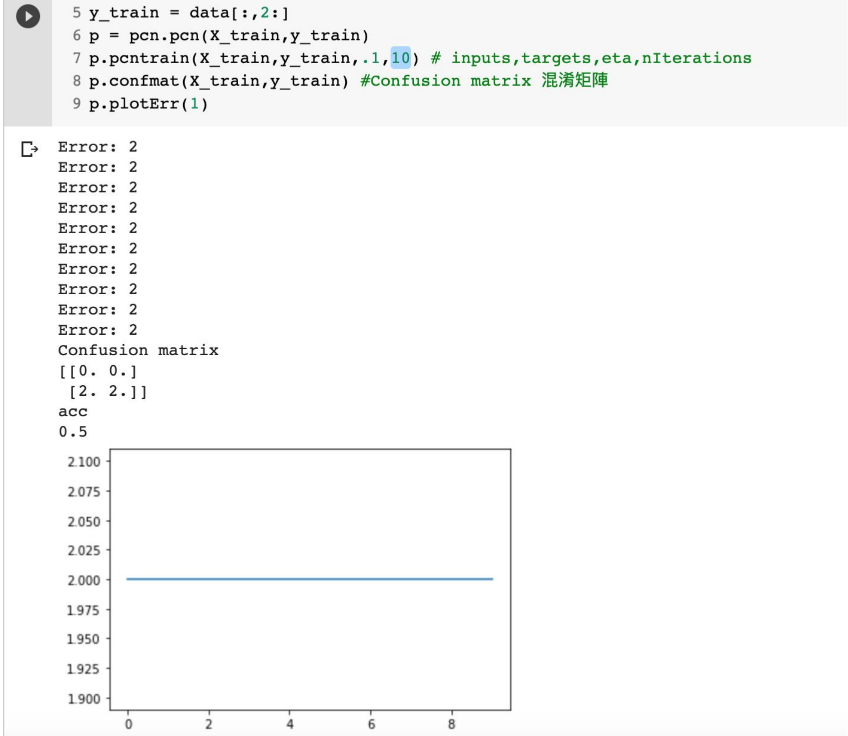
Eta = 0.9, Iter = 10



```
y_train = data[:,2:]
p = pcn.pcn(X_train,y_train)
p.pcntrain(X_train,y_train,.9,10) # inputs,targets,eta,nIterations
p.confmat(X_train,y_train) #Confusion matrix 混淆矩陣
p.plotErr(1)
```

```
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Confusion matrix
[[0. 0.]
 [2. 2.]]
acc
0.5
```

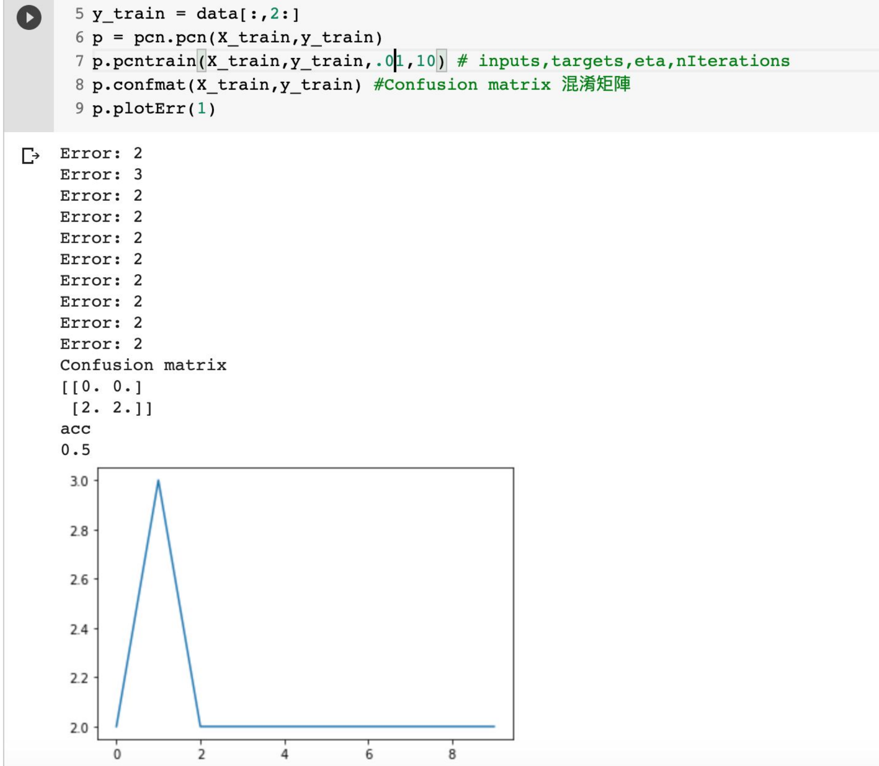
Eta = 1, Iter = 10



```
y_train = data[:,2:]
p = pcn.pcn(X_train,y_train)
p.pcntrain(X_train,y_train,.1,10) # inputs,targets,eta,nIterations
p.confmat(X_train,y_train) #Confusion matrix 混淆矩陣
p.plotErr(1)
```

```
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Confusion matrix
[[0. 0.]
 [2. 2.]]
acc
0.5
```

Eta = .01, Iter = 10



```
y_train = data[:,2:]
p = pcn.pcn(X_train,y_train)
p.pcntrain(X_train,y_train,.01,10) # inputs,targets,eta,nIterations
p.confmat(X_train,y_train) #Confusion matrix 混淆矩陣
p.plotErr(1)
```

```
Error: 2
Error: 3
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Error: 2
Confusion matrix
[[0. 0.]
 [2. 2.]]
acc
0.5
```