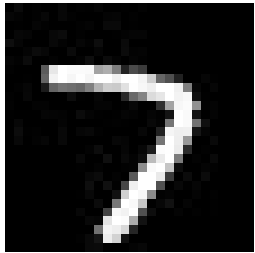
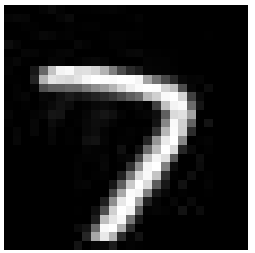
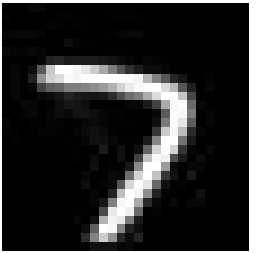
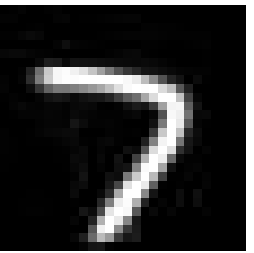


1(a). all training data

origin	Sparsity = 5	Sparsity = 10	Sparsity = 150
			

```
(1.a) sparsity: 5, 30 testing data Euclidean distance sum: 20788.30393722378
(1.a) sparsity: 10, 30 testing data Euclidean distance sum: 18522.996400206488
(1.a) sparsity: 150, 30 testing data Euclidean distance sum: 16214.04815112598
(1.a) sparsity: 5, idx: 17 error: 444.86134683365214
(1.a) sparsity: 10, idx: 17 error: 394.31988119496003
(1.a) sparsity: 150, idx: 17 error: 350.0239502468041
```

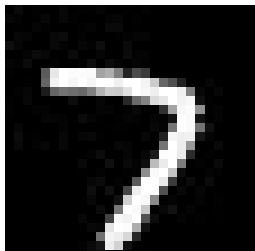
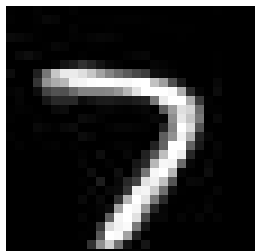
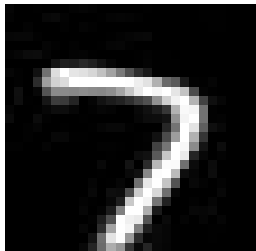
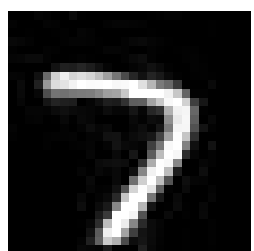
上圖的前三行是 30 筆資料的歐式距離總和

上圖的後三行是任意抽一筆資料的歐式距離

基本上可以看到，這筆資料的效果非常的好，基本上是根本沒有什麼誤差了，就是在 **sparsity** 由小到大都表現很好。

然後看歐式距離的時候，其實還是可以發現，隨著 **sparsity** 越來越大，歐式距離還是有下降，代表這個方法還是有效，只是一開始效果就不錯了。

1(b). base 縮小

Origin	Sparsity = 5	Sparsity = 10	Sparsity = 150
			

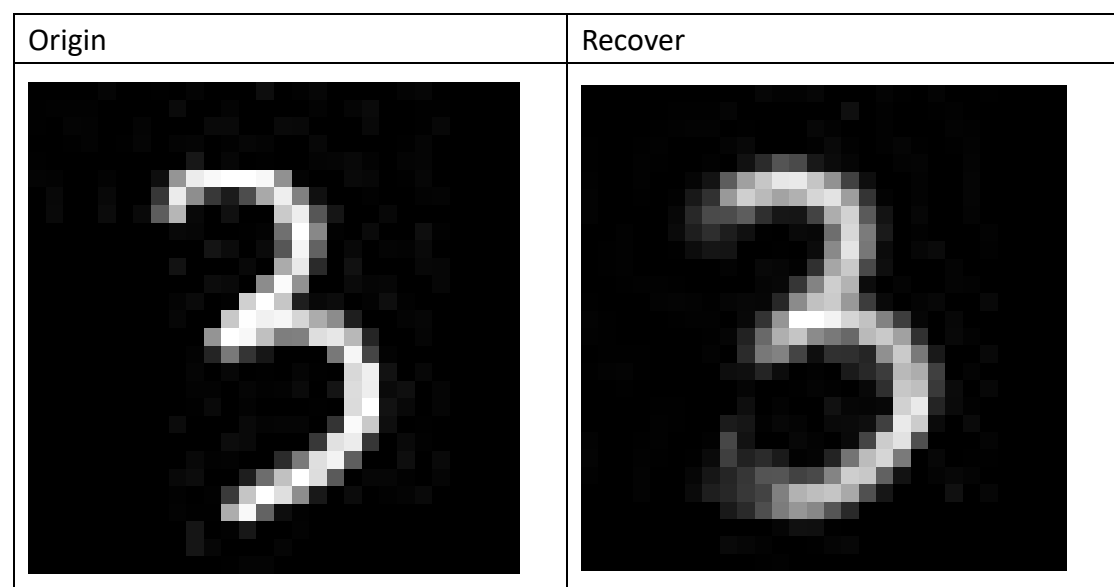
```
(1.b) sparsity: 5, 30 testing data Euclidean distance sum: 23742.787827405173
(1.b) sparsity: 10, 30 testing data Euclidean distance sum: 21728.770296397946
(1.b) sparsity: 150, 30 testing data Euclidean distance sum: 19478.734517943536
(1.b) sparsity: 5, idx: 17 error: 505.90431479624885
(1.b) sparsity: 10, idx: 17 error: 432.2241426915782
(1.b) sparsity: 150, idx: 17 error: 396.15028746482017
```

上圖的前三行是 30 筆資料的歐式距離總和
上圖的後三行是任意抽一筆資料的歐式距離

基本上把 **base** 縮小之後，用肉眼看圖片跟上一題其實還是沒有什麼差距，但是如果細看歐式距離的話，會發現把 **base** 縮小真的有差，會讓歐式距離比較大一些，就是整體的 **error** 會比 **1.a** 做出來都要大，不過肉眼看真的圖片差距不大。

圖片看效果不彰有可能是我選資料選得不好，又或者是 **OMP** 在 **sparsity** 比較低的情況下表現就已經很不錯了，兩者都是原因。

2.



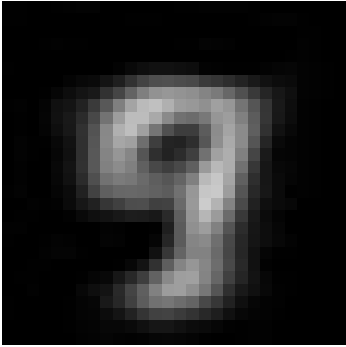
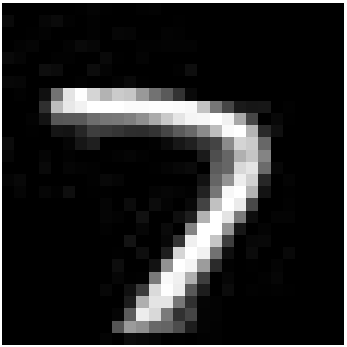
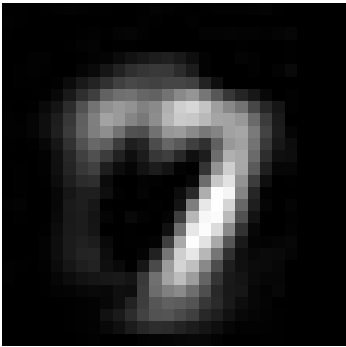
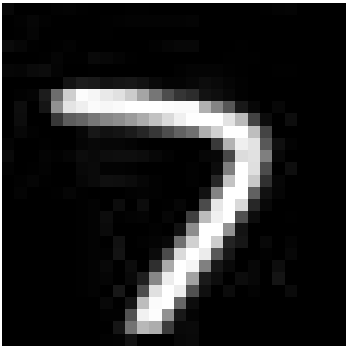
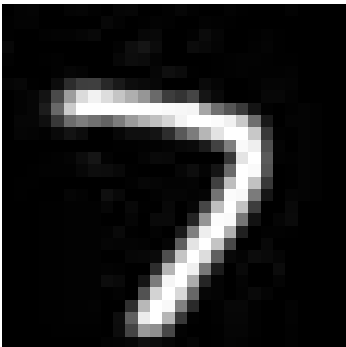
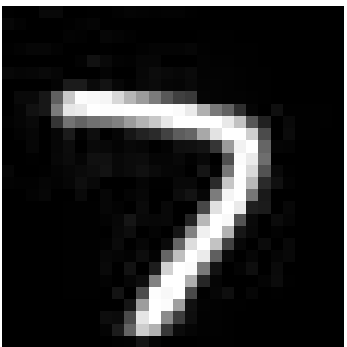
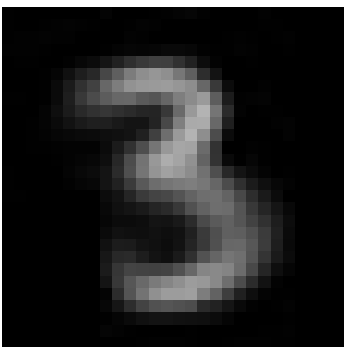

```
(base) johntseng@johntseng-Super-Server:~/文件/NTU-Digital-Signal-Processing-2020/Program_HW2$ python p2.py  
(2) sparsity: 10, 30 testing data Euclidean distance sum: 22565.901715043663  
(2) sparsity: 10, idx: 17 error: 595.4393195212391
```

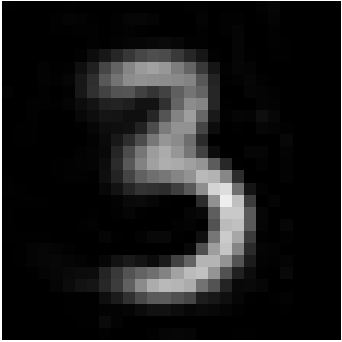
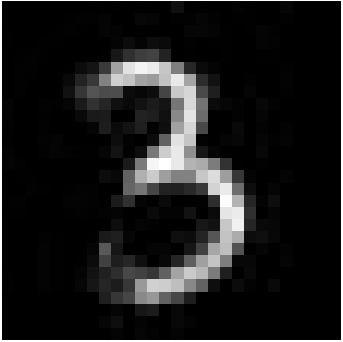

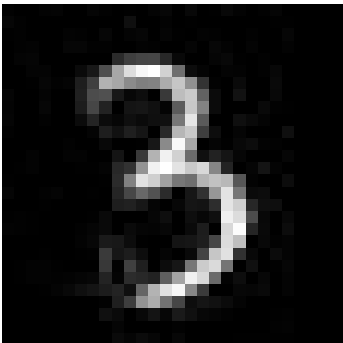
根據題目要求計算 label data 的前 30 筆資料歐式距離的和為
22565.0901715043663，

然後從中抽取一筆資料，歐式距離為 595.4393195212391

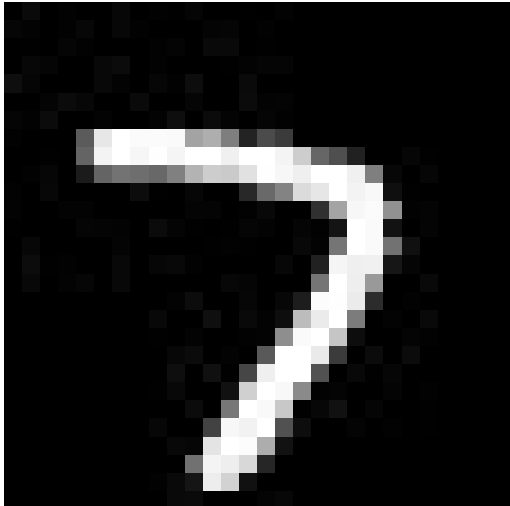

基本上 **recover** 的圖片和原圖還算相近，在 **sparsity=10** 的情況下就可以看出來 3 的輪廓非常清晰而且好辨識，因為 **data** 都是數字為 3 的資料，所以抽出來的效果看起來不錯，而且電腦計算也很快。

3.

	Center PCA	OMP
All data 25%		
All data 50%		
All data 95%		
Label data 25%		

Label data 50%		
Label data 95%		

這是原始的圖片：

All data	Label data
	

```
(base) johntseng@johntseng-Super-Server:~/文件/NTU-Digital-Signal-Processing-2020/Program_HW2$ python p3_part.py
(3) idx: 17 center PCA, energy preserve: 25 euclidean distance: 1152.4316056815428
(3) idx: 17 center PCA, energy preserve: 50 euclidean distance: 940.7573172745067
(3) idx: 17 center PCA, energy preserve: 95 euclidean distance: 283.57174255259395
(3) idx: 17 OMP, sparsity: 3 euclidean distance: 765.4033400222659
(3) idx: 17 OMP, sparsity: 9 euclidean distance: 594.4708693134166
(3) idx: 17 OMP, sparsity: 138 euclidean distance: 491.9932651084196
(base) johntseng@johntseng-Super-Server:~/文件/NTU-Digital-Signal-Processing-2020/Program_HW2$ python p3_all.py
(3) idx: 17 center PCA, energy preserve: 25 euclidean distance: 1680.733779568277
(3) idx: 17 center PCA, energy preserve: 50 euclidean distance: 1346.2598453363269
(3) idx: 17 center PCA, energy preserve: 95 euclidean distance: 385.6395513966063
(3) idx: 17 OMP, sparsity: 4 euclidean distance: 553.4399931318729
(3) idx: 17 OMP, sparsity: 11 euclidean distance: 379.3977044004049
(3) idx: 17 OMP, sparsity: 154 euclidean distance: 317.09514960502486
```

兩種不同演算法和原圖計算出來的歐式距離都在上面了

上圖 p3_part.py 是對 label data 的計算結果

我的 label 是選擇 3

可以看到 PCA 的 energy 在 25%,50%,95%時分別對應的 sparsity 是 3,9,138

上圖 p3_all.py 是對全部資料的計算結果

可以看到 PCA 的 energy 在 25%,50%,95%時分別對應的 sparsity 是 4,11,154

我覺得直接從圖片來看兩個演算法效果比較好，可以發現在 PCA 的 energy 比較低的時候，PCA 的效果都非常差，但是 OMP 在 sparsity 比較低的時候就會非常近似原圖了，就是肉眼已經清楚可以辨認出數字，而且也算清晰，然而 PCA 看起來很糟糕。

隨著 energy preserve 變大，可以發現 PCA 的清晰度就變好非常多，但是 OMP 的效果我覺得並沒有差異到非常大。

其實這個現象從演算法本身就可以發現了，PCA 是要找出 eigenvector 去 fit 整體的資料，但是 OMP 是拿整體的資料，找最像的來 fit 我要表示的 data，所以有這樣的結果也不太意外。

另外 label data 和 all data 的差別，我覺得在 PCA 影響比較大，資料越大，PCA 在能量保留低的時候表現很糟，其他看起來差異並不大。