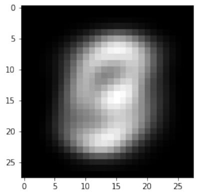
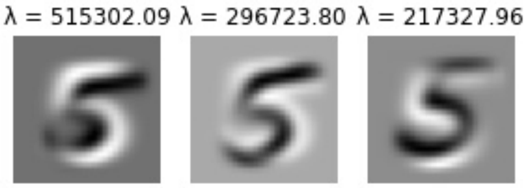
Part 1.PCA

Q1.



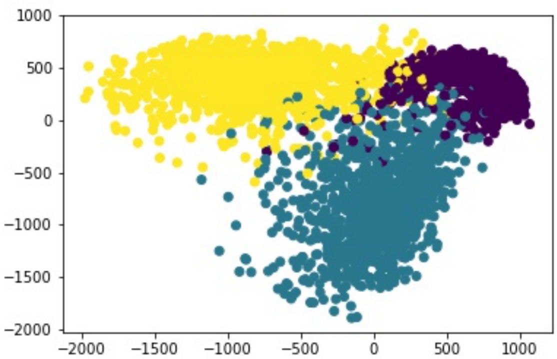
Q2.



Q3.

隨著用於重建的 eigenvector 數越多，重建出來的圖會越來越像原圖。只用 3 和 10 個 eigenvector 的重建效果都還很差，到用 30 個時就可以看到原圖的大致輪廓，到 100 個時除了雜訊和模糊，基本上可以認出是同一張圖片。

Q4.



基本上可以三個數字用前兩個 eigenvector 去表示的係數已經可以分的滿開，不過邊緣的地方還是有重疊，若要更好的分類效果可能還要再加維度。

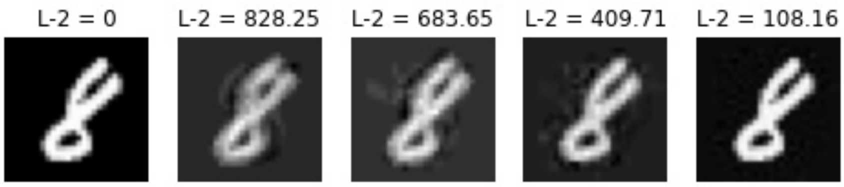
Part 2.OMP

Q5.

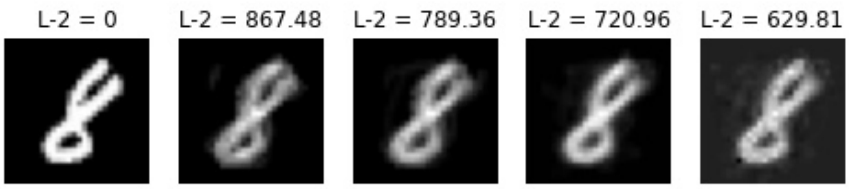
找到的基底幾乎都是 3，用同為 3 當做基底能夠較好地重建原始影像。特別的是，在這題如果挑選和 original 內積最大的基底時，不取絕對值，反而可以得到五個 3 的基底，並且獲得更小的 L2-norm，與同學討論得出的結論是，可能是因為 OMP 並非 optimal 的解法才會有此結果。

內積不取絕對值得到的 5 個基底：	重建：有(L)沒(R)取絕對值	
		

Q6.



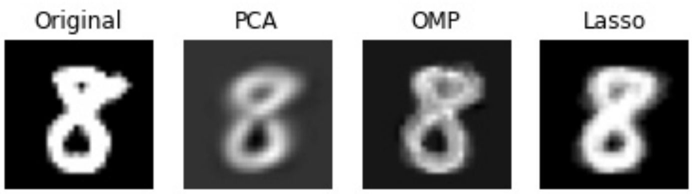
隨著 sparsity 越大重建效果越好，L2-norm 越小，不過和上一題相反，這題若不取絕對值結果會變差很多。下圖是不取絕對值所重建的結果。



Part 3.Lasso

Q7.

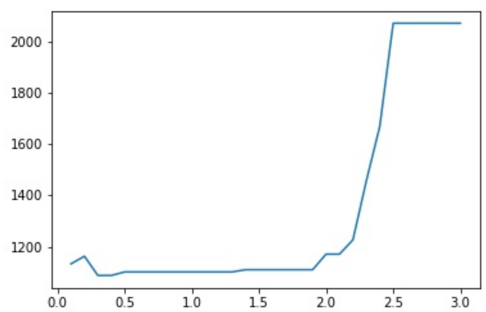
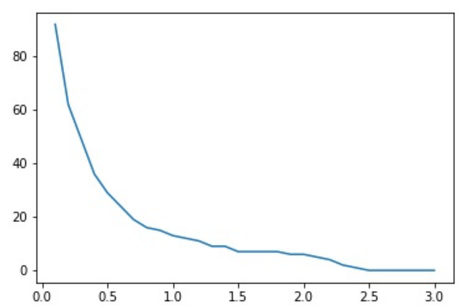
1.2.3.



　　上圖為 PCA, OMP, Lasso 三種方法使用五個基底重建 8 的結果。Lasso 的的效果感覺是最好的，其次是 OMP，最差是 PCA。

4.

　　上圖為調整 alpha 值所得到的重建結果。圖片上方的數字代表（alpha 值, L2-norm），可以發現 alpha 再 0.5~1.3 時有最小的 L2-norm(1101.21)，當 alpha=2.1 時係數不為零的基底個數是五個，到 alpha=2.5 以上，全部的係數皆為零。

L2-norm	Coef 不等於零的基底個數
	

（X 軸為 alpha）

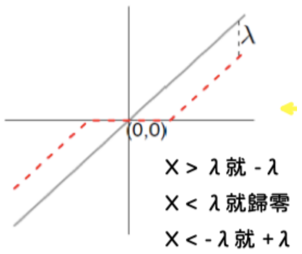
Bonus:

```python
def soft_threshold(tmp, lamda):
    if tmp < (-1)*lamda:
        return (tmp+lamda)
    elif tmp > lamda:
        return (tmp-lamda)
    return 0
def coordinate_descent(A, y, x, lamda, nIter):
    nfeature, nSample = A.shape
    for k in range(nIter):
        for i in range(nSample):
            Ai = A[:,i].reshape(-1,1)
            tmp = Ai.T @ (y - (A@x) + x[i]*Ai)
            x[i] = soft_threshold(tmp, lamda)
    return x
```

Explain :

soft_threshold 即下圖 function 的實現

而 coordinate_descent 則是根據最佳解公式算參數



最佳解公式

$$x_i = S_{\lambda/\|A_i\|^2}\left(\frac{A_i^T(y - A_{-i}x_{-i})}{A_i^T A_i}\right)$$

```python
lamda = np.linspace(0.1, 3, num=30)

A = nor[:6824].T        # A:784x6824
y = test.reshape(-1,1)# y:784x1
x = np.ones((6824, 1))# x:6824x1

for i in range(30):
    x = coordinate_descent(A, y, x, lamda[i], 1)
```

根據上題，設定 λ 在[0.1, 3]區間，以便和套件的 Lasso 做比較，權重 x 初始設定 1，接著即 loop 每個 λ 去跑 coordinate_descent。

下圖為重建結果。圖片上方的數字代表（λ ，L2-norm），在 λ=0.5-0.9 區間 L2-norm 最小，和套件 Lasso 同，但是手刻的 L2-norm 比較大（我的：1176.01 > Lasso：1101.21）