pca_prob_python

May 9, 2019

1 Solutions to PCA Problems

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
```

1.1 Problem 1(a)

Sample mean

```
In [3]: X = np.array([[3,2,1], [2,4,5], [1,2,3], [0,2,5]])
        mu = np.mean(X, axis=0)
        print(mu)
```

```
[1.5 2.5 3.5]
```

1.2 Problem 1(b)

Sample covariance

```
In [4]: n = X.shape[0]
        Xm = X - mu[None,:]   # Python broadcasting
        Q = (1/n)*Xm.T.dot(Xm)
        print(Q)
```

```
[[ 1.25  0.25 -1.25]
 [ 0.25  0.75  0.75]
 [-1.25  0.75  2.75]]
```

You can also compute it with the `np.cov` function. Note that you must set `bias=True` for the correct scaling.

```
In [11]: print(np.cov(X.T, bias=True))
```

```
[[ 1.25  0.25 -1.25]
 [ 0.25  0.75  0.75]
 [-1.25  0.75  2.75]]
```

1.3 Problem 1(c)

Eigenvalues

```
In [5]: lam, V = np.linalg.eigh(Q)
        print(lam)
        print(V)
```

```
[0.01495506 1.1733803  3.56166464]
[[ 0.59363515  0.66677184 -0.45056922]
 [-0.66472154  0.72187235  0.19247228]
 [ 0.45358856  0.18524476  0.87174641]]
```

1.4 Problem 1(d) and (e)

Transform and reconstruct `X`. You see that `Xhat` matches `X`

```
In [8]: # Get the coefficients
        Z = (X - mu[None,:]).dot(V)

        # Reconstruct the X.
        Xhat = Z.dot(V.T) + mu[None,:]
        print(X)
```

```
[[3 2 1]
 [2 4 5]
 [1 2 3]
 [0 2 5]]
```

1.5 Problem 1(f).

Approximate with the two largest PCs. Note that the `eigh` function sorts the eigenvalues in ascending order, so we select the last two columns.

```
In [19]: k = 2
         Xhat = Z[:,k-1:].dot(V[:,k-1:].T) + mu[None,:]
         print(Xhat)
```

```
[[ 2.94726021  2.05905526  0.95970224]
 [ 2.0118026   3.98678407  5.0090182 ]
 [ 1.11353336  1.87287129  3.0867493 ]
 [-0.07259617  2.08128939  4.94453025]]
```

1.6 Problem 1(g)

Approximation error

```
In [20]: Xerr = np.mean(np.sum((Xhat-X)**2,axis=1))
         print(Xerr)
         lam
```

```
0.014955061432806494
```

```
Out[20]: array([0.01495506, 1.1733803 , 3.56166464])
```

1.7 Problem 2

```
In [21]: mu = np.array([1, 0, 2])
         v1 = 1/np.sqrt(2)*np.array([1,1,0])
         v2 = 1/np.sqrt(2)*np.array([1,-1,0])
         x = np.array([2,3,4])
```

```
In [22]: # Part 2(a)

         z1 = v1.dot(x-mu)
         z2 = v2.dot(x-mu)
         print([z1,z2])
```

```
[2.82842712474619, -1.414213562373095]
```

```
In [23]: # Part 2(b)
         xhat = z1*v1 + z2*v2 + mu
         print(xhat)
```

```
[2. 3. 2.]
```

```
In [24]: # Part 2(c)
         np.sum((x-xhat)**2)
```

```
Out[24]: 4.0
```

```
In [ ]:
```