PCA 降维案例：iris

1. 手动实施 PCA

```
>>> iris = datasets.load_iris()

>>> col_mean = np.mean(iris.data,axis=0)
>>> col_mean
array([5.84333333, 3.054     , 3.75866667, 1.19866667])

>>> centered_data = iris.data - col_mean

>>> cov_data = np.cov(centered_data.T)
>>> cov_data
array([[ 0.68569351, -0.03926846,  1.27368233,  0.5169038 ],
       [-0.03926846,  0.18800403, -0.32171275, -0.11798121],
       [ 1.27368233, -0.32171275,  3.11317942,  1.29638747],
       [ 0.5169038 , -0.11798121,  1.29638747,  0.58241432]])

>>> w,v = np.linalg.eig(cov_data)
>>> w
array([4.22484077, 0.24224357, 0.07852391, 0.02368303])

>>> v
array([[ 0.36158968, -0.65653988, -0.58099728,  0.31725455],
       [-0.08226889, -0.72971237,  0.59641809, -0.32409435],
       [ 0.85657211,  0.1757674 ,  0.07252408, -0.47971899],
       [ 0.35884393,  0.07470647,  0.54906091,  0.75112056]])

>>> np.dot(cov_data,v[:,0])
array([ 1.52765881, -0.34757296,  3.61888075,  1.51605845])
>>> w[0] * v[:,0]
array([ 1.52765881, -0.34757296,  3.61888075,  1.51605845])

>>> data_pca2c = np.dot(centered_data,v[:,0:2])
```

2. 利用 sklearn 实施 PCA

```
from sklearn.decomposition import PCA
```

保持 4 维

```
pca4c = PCA(n_components=4)
data_pca4c = pca4c.fit_transform(iris.data)
print data_pca4c.shape
(150, 4)
print pca4c.explained_variance_ratio_.sum()
```

```
1.0
print pca4c.components_
[[ 0.36158968 -0.08226889  0.85657211  0.35884393]
 [-0.65653988 -0.72971237  0.1757674   0.07470647]
 [ 0.58099728 -0.59641809 -0.07252408 -0.54906091]
 [ 0.31725455 -0.32409435 -0.47971899  0.75112056]]
```

降至 3 维

```
pca3c = PCA(n_components=3)
data_pca3c = pca3c.fit_transform(iris.data)
print data_pca3c.shape
(150,  3)
print pca3c.explained_variance_ratio_.sum()
0.994816914549810l
print pca3c.components_
[[ 0.36158968 -0.08226889  0.85657211  0.35884393]
 [-0.65653988 -0.72971237  0.1757674   0.07470647]
 [ 0.58099728 -0.59641809 -0.07252408 -0.54906091]]
```

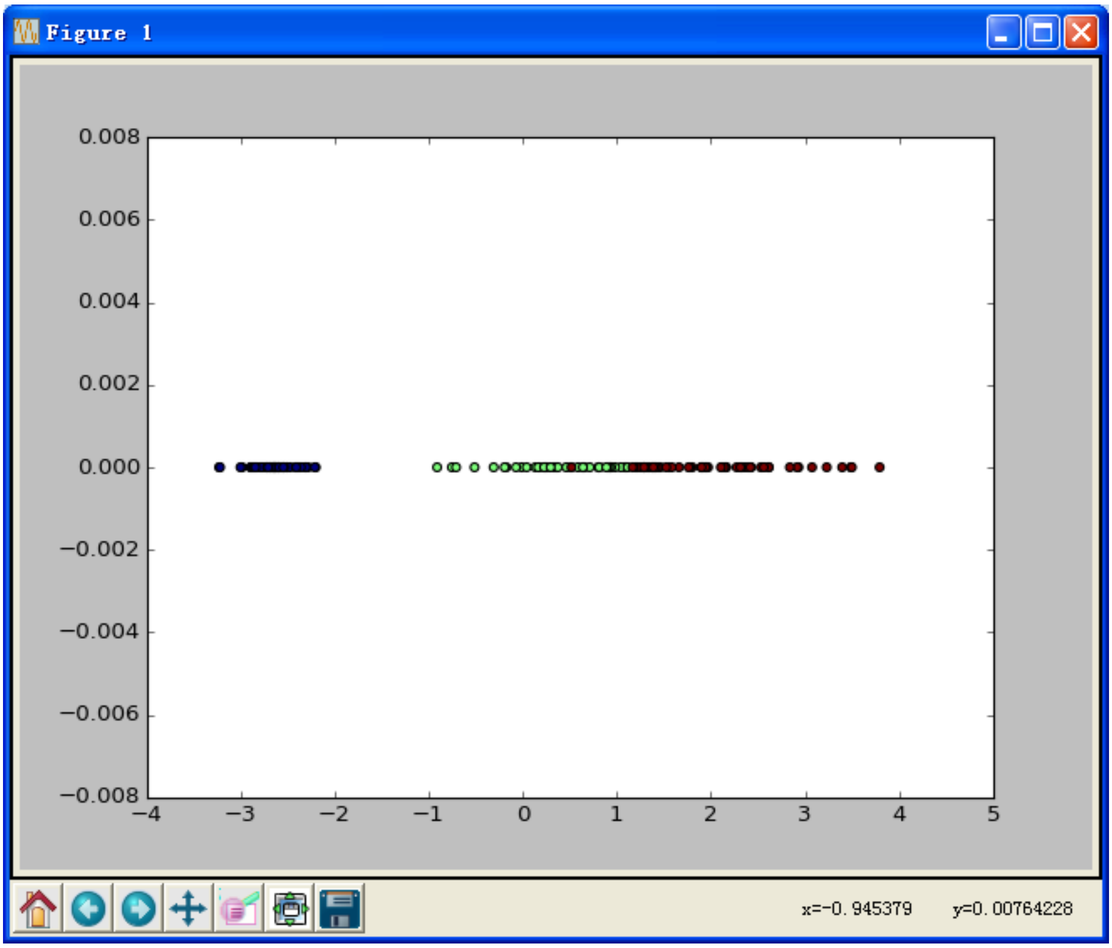
降至 2 维

```
pca2c = PCA(n_components=2)
data_pca2c = pca2c.fit_transform(iris.data)
print data_pca2c.shape
(150,  2)
print pca2c.explained_variance_ratio_.sum()
0.97763177502480 33
print pca2c.components_
[[ 0.36158968 -0.08226889  0.85657211  0.35884393]
 [-0.65653988 -0.72971237  0.1757674   0.07470647]]
```

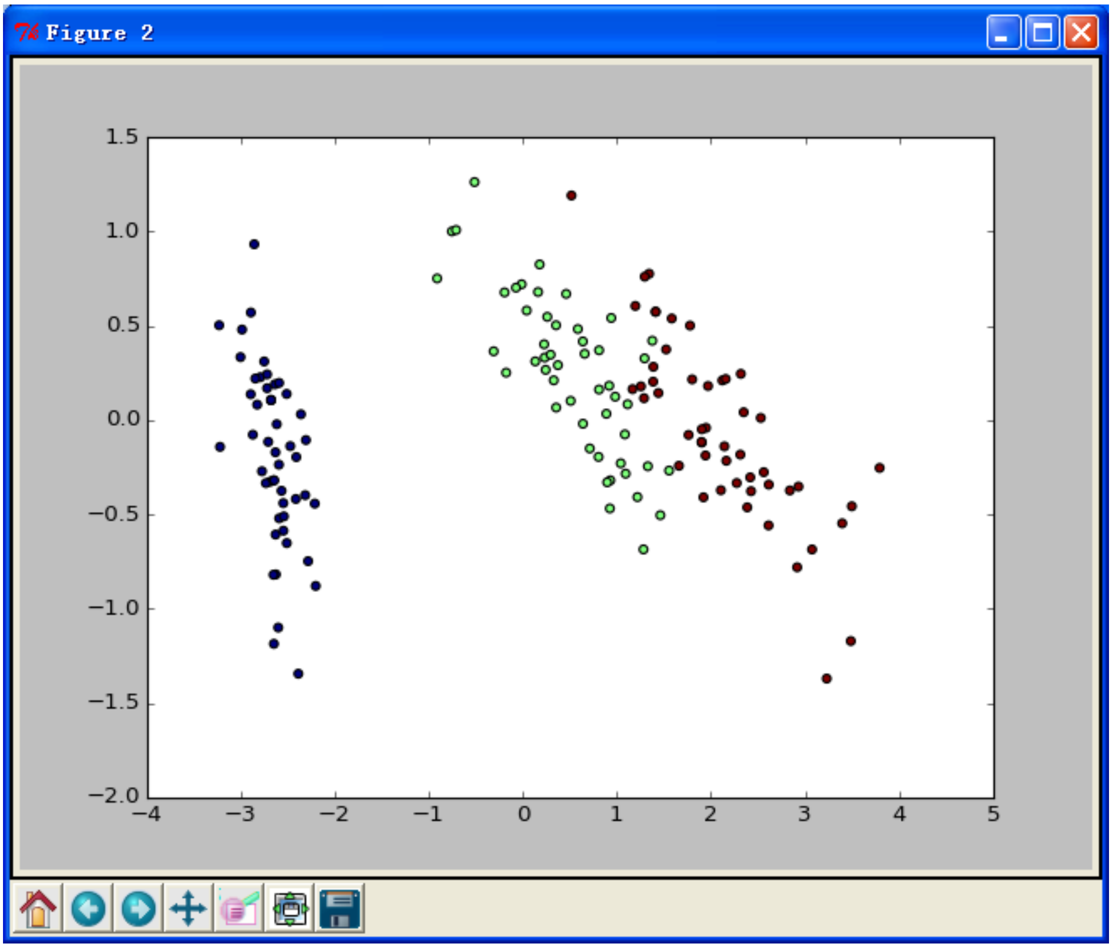
降至 1 维

```
pca1c = PCA(n_components=1)
data_pca1c = pca1c.fit_transform(iris.data)
print data_pca1c.shape
(150,  1)
print pca1c.explained_variance_ratio_.sum()
0.924616207174268 3
print pca1c.components_
[[ 0.36158968 -0.08226889  0.85657211  0.35884393]]

plt.scatter(data_pca1c[:,0],np.zeros(data_pca1c.shape),c=iris.targ
et)
```



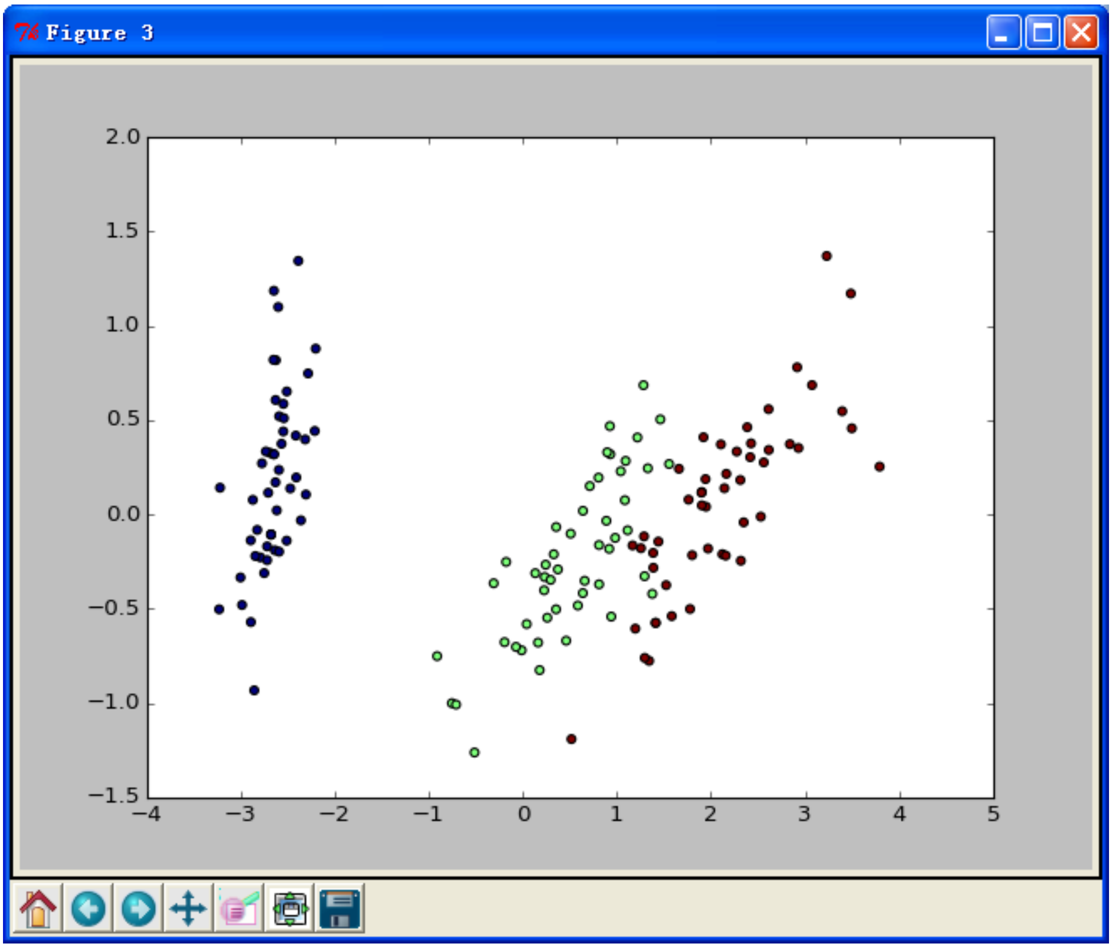

```
plt.figure(2)
plt.scatter(data_pca2c[:,0],data_pca2c[:,1],c=iris.target)
```



3. RandomizedPCA 适合于大数据集

```
rpca2c = RandomizedPCA(n_components=2)
data_rpca2c = rpca2c.fit_transform(iris.data)
print rpca2c.explained_variance_ratio_.sum()
0.977631775024804   # 随机：每次运行略有不同

plt.figure(3)
plt.scatter(data_rpca2c[:,0],data_rpca2c[:,1],c=iris.target)
```




```
plt.show()
```