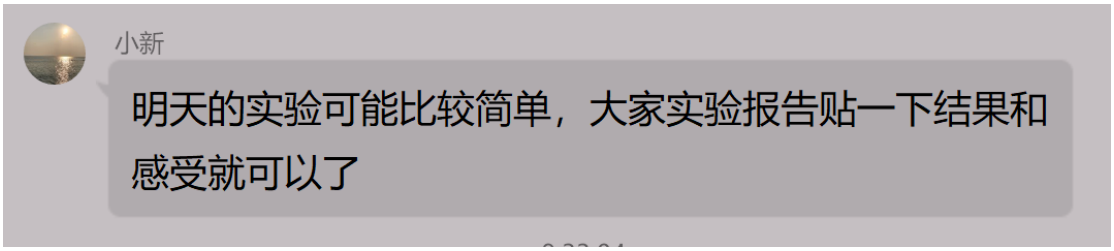
机器学习课程实验六

2022 年 10 月 13 日　苏博南　202000460020

首先先贴老师原话：

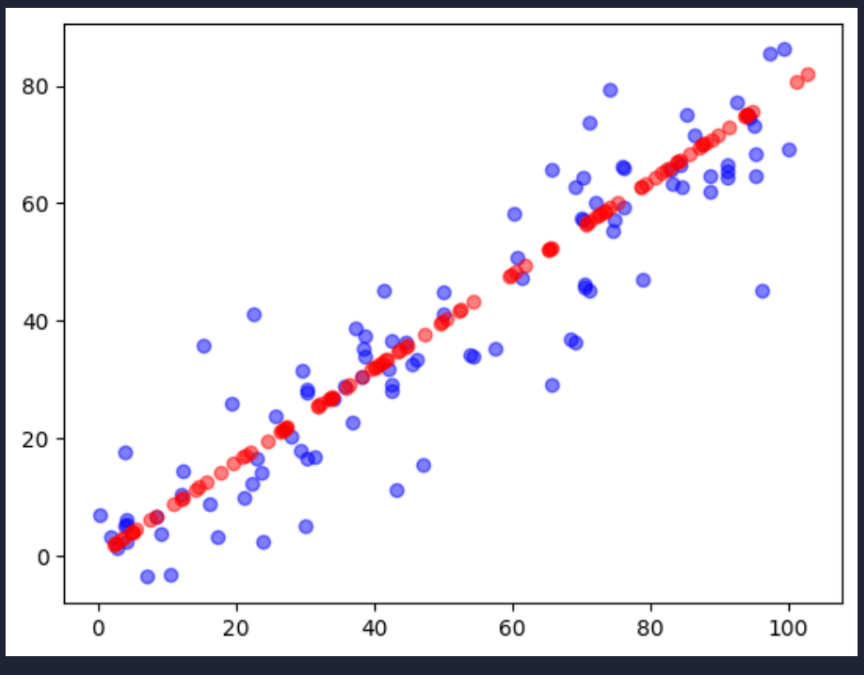


然后一看代码，PCA 部分不是 import 自 sklearn.decomposition 就是 import 自 playML。好嘛那我就贴结果和感受了。

1 结果

```
[10]  pca = PCA(n_components=1)
      pca.fit(X)
      ✓ 0.1s

...   PCA(n_component=1)

[11]  X_reduction = pca.transform(X)
      ✓ 0.1s

[12]  X_reduction.shape
      ✓ 0.8s

...   (100, 1)

[13]  X_restore = pca.inverse_transform(X_reduction)
      ✓ 0.6s

[14]  X_restore.shape
      ✓ 0.7s

...   (100, 2)
```

```
[15]  plt.scatter(X[:,0], X[:,1], color='b', alpha=0.5)
      plt.scatter(X_restore[:,0], X_restore[:,1], color='r', alpha=0.5)
      plt.show()
      ✓ 0.6s
```



scikit_learn中的PCA

```
[16] from sklearn.decomposition import PCA
```
✓ 1.7s

```
[17] pca = PCA(n_components=1)
     pca.fit(X)
```
✓ 0.1s

```
PCA(n_components=1)
```

```
[18] pca.components_
```
✓ 0.1s

```
array([[0.78143604, 0.62398534]])
```

```
[19] X_reduction = pca.transform(X)
```
✓ 0.6s

```
[20] X_reduction.shape
```
✓ 0.9s

```
(100, 1)
```

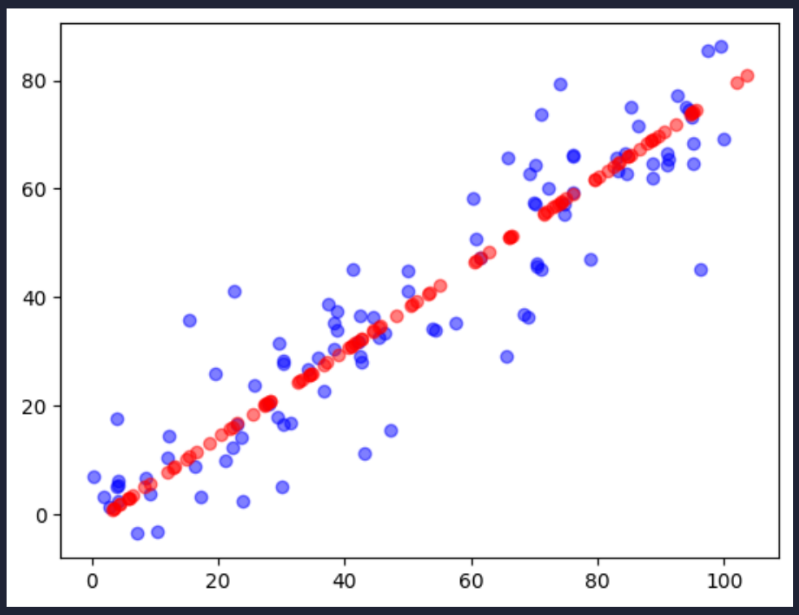
```
[21] X_restore = pca.inverse_transform(X_reduction)
```
✓ 0.1s

```
[22] X_restore.shape
```
✓ 0.7s

```
(100, 2)
```

```
plt.scatter(X[:,0], X[:,1], color='b', alpha=0.5)
plt.scatter(X_restore[:,0], X_restore[:,1], color='r', alpha=0.5)
plt.show()
```

[23] ✓ 0.3s



scikit-learn中的PCA

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
```

[2] ✓ 3.8s

```
digit = datasets.load_digits()
X = digit.data
y = digit.target
```

[3] ✓ 0.1s

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=666)
```

[4] ✓ 0.1s

```
X_train.shape
```

[5] ✓ 0.9s

```
(1347, 64)
```

```
%%time
from sklearn.neighbors import KNeighborsClassifier

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)
```

[6] ✓ 0.6s

```
Wall time: 492 ms

KNeighborsClassifier()
```

```
knn_clf.score(X_test, y_test)
```

[7] ✓ 0.1s

```
0.9866666666666667
```

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca.fit(X_train)
X_train_reduction = pca.transform(X_train)
X_test_reduction = pca.transform(X_test)      #用训练集训练得到的pca来直接对X_test降维，不要对X_test再进行fit处理
```

[8] ✓ 0.8s

```
%%time

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train_reduction, y_train)
```

[9] ✓ 0.1s

```
Wall time: 14.4 ms

KNeighborsClassifier()
```

```
knn_clf.score(X_test_reduction, y_test)
```

[10] ✓ 0.1s

```
0.6066666666666667
```

```
pca.explained_variance_ratio_      ##拟合原数据的多少
```

[11] ✓ 0.6s

```
array([0.14566817, 0.13735469])
```

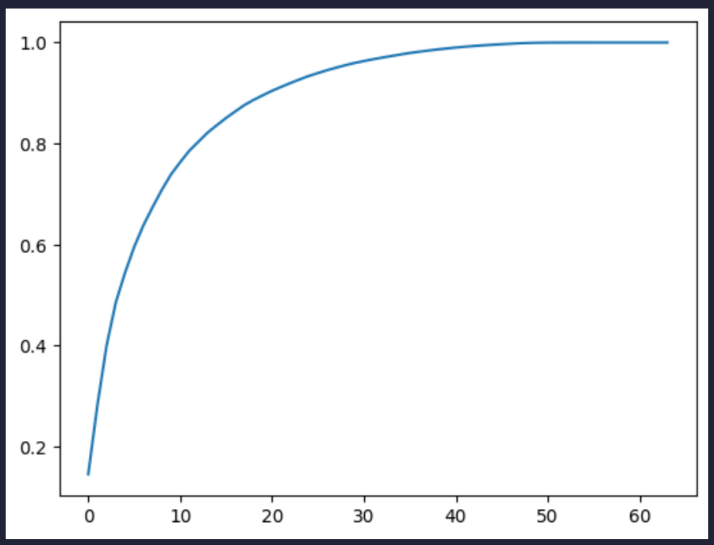
```
pca = PCA(n_components=X_train.shape[1])
pca.fit(X_train)
pca.explained_variance_ratio_
```

[12] ✓ 0.1s

```
array([1.45668166e-01, 1.37354688e-01, 1.17777287e-01, 8.49968861e-02,
       5.86018996e-02, 5.11542945e-02, 4.26605279e-02, 3.60119663e-02,
       3.41105814e-02, 3.05407804e-02, 2.42337671e-02, 2.28700570e-02,
       1.80304649e-02, 1.79346003e-02, 1.45798298e-02, 1.42044841e-02,
       1.29961033e-02, 1.26617002e-02, 1.01728635e-02, 9.09314698e-03,
       8.85220461e-03, 7.73828332e-03, 7.60516219e-03, 7.11864860e-03,
       6.85977267e-03, 5.76411920e-03, 5.71688020e-03, 5.08255707e-03,
       4.89020776e-03, 4.34888085e-03, 3.72917505e-03, 3.57755036e-03,
       3.26989470e-03, 3.14917937e-03, 3.09269839e-03, 2.87619649e-03,
       2.50362666e-03, 2.25417403e-03, 2.20030857e-03, 1.98028746e-03,
       1.88195578e-03, 1.52769283e-03, 1.42823692e-03, 1.38003340e-03,
       1.17572392e-03, 1.07377463e-03, 9.55152460e-04, 9.00017642e-04,
       5.79162563e-04, 3.82793717e-04, 2.38328586e-04, 8.40132221e-05,
       5.60545588e-05, 5.48538930e-05, 1.08077650e-05, 4.01354717e-06,
       1.23186515e-06, 1.05783059e-06, 6.06659094e-07, 5.86686040e-07,
       1.71368535e-33, 7.44075955e-34, 7.44075955e-34, 7.15189459e-34])
```

```
plt.plot([i for i in range(X_train.shape[1])],
         [np.sum(pca.explained_variance_ratio_[:i+1]) for i in range(X_train.shape[1])])
plt.show()
```

[13] ✓ 0.3s



```
[14]  pca = PCA(0.95)      #解释原来数据95%的方差
      pca.fit(X_train)
      ✓ 0.1s

...   PCA(n_components=0.95)
```

```
[15]  pca.n_components_
      ✓ 0.5s

...   28
```

```
[16]  X_train_reduction = pca.transform(X_train)
      X_test_reduction = pca.transform(X_test)
      ✓ 0.5s
```

```
      %%time

      knn_clf = KNeighborsClassifier()
      knn_clf.fit(X_train_reduction, y_train)
[17]  ✓ 0.1s

...   Wall time: 1.53 ms

      KNeighborsClassifier()
```

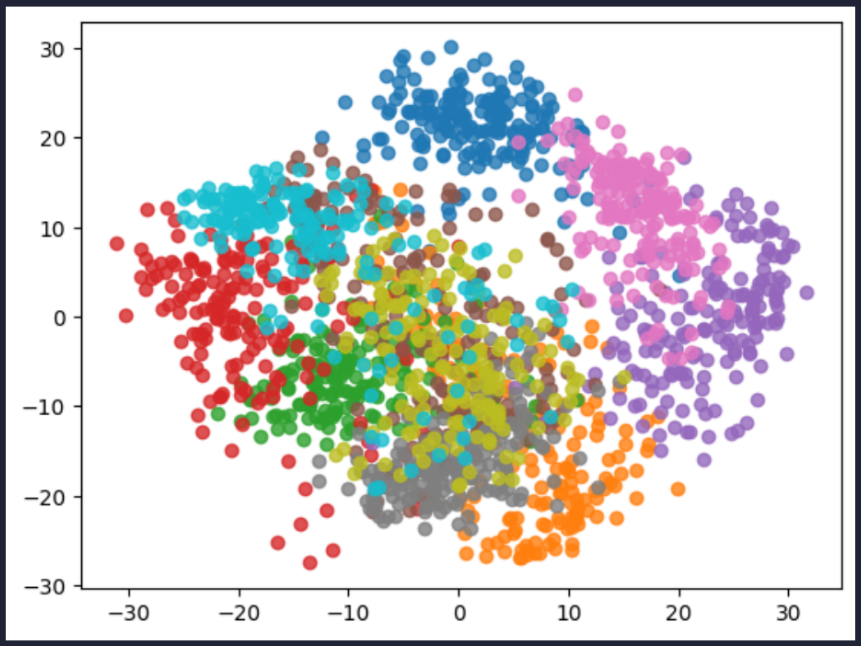
```
[18]  knn_clf.score(X_test_reduction, y_test)
      ✓ 0.1s

...   0.98
```

```
[19]  pca = PCA(n_components=2)
      pca.fit(X)
      X_reduction = pca.transform(X)
      ✓ 0.7s
```

```
[20]  X_reduction.shape
      ✓ 0.9s

...   (1797, 2)
```

```
for i in range(10):
    plt.scatter(X_reduction[y==i,0], X_reduction[y==i,1], alpha=0.8)
plt.show()
```

[21] ✓ 0.3s



2 感受

没有啥感受。不知道代码在干嘛，反之运行了一遍，也不用写 PCA，KNeighboursClassifier 也不知道在干嘛，认为实验设置不合理，对非机器学习专业同学不友好，体验挺差，不如之前写个 pdf 用 matlab 求一下协方差和本征矢更像回事，或者用 spss 因子分析看一下相关系数，总之感觉 python 确实意义不明了。