# Practicum III: Unsupervised Learning (Dimension_Reduction_and_Clustering)

已開始： 12月17日 11:43

## 測驗說明

**Practicum III: Introduction to unsupervised learning: Dimension Reduction and Clustering analysis**

(Not required in ML class, but encouraged to practice after the course finishes)

Useful Reference:
1. Slides: **CSCI5750_Lecture20_unsupervised_learning.pdf (https://canvas.slu.edu/courses/28036/files/2162134?wrap=1)**
2. **https://github.com/ageron/handson-ml2/ (https://github.com/ageron/handson-ml2/)**

Note:
**Unsupervised learning** is a class of machine learning techniques for discovering patterns in data (without looking into labels).
For instance, finding the “clusters” of texts/words based on their semantic meanings (e.g., home & house, Saint Louis & Columbia).

Two representative examples in unsupervised learning is called **clustering** and **dimension reduction**. A good visualization example is here: https://projector.tensorflow.org/

The notebook is organized as follows:
- We need to get the features or vector representations of each example (e.g., embedding features of words, pixels of images).
- All examples will be saved in a data matrix, where each row is the example (e.g., words, image), and each column is the feature of each example (e.g., embeddings, pixels).
- We can explore dimension reduction techniques on training data
- We can cluster the examples using **k-means clustering**. K-means finds a specific number of clusters in the samples based on the distance measurements among data. You can call this function in the 'sklearn' library.

**Note:**
- This hands-on activity reflects the key contents in Chapter 8 & 9 of the textbook

- Please follow the tutorial to practice the codes. The work will be counted as bonus credits to part of course participation together (with reading sessions)
- We suggest everyone practice the codes and submit the results. You can copy texts/upload images from Google Colab and paste into the textbox.
- You can always leave this survey anytime without clicking submit button. Canvas will automatically save your work for latter use.

---

**Practicum III: Introduction to unsupervised learning: Dimension Reduction and Clustering analysis**

Useful Reference:
1. Slides: **CSCI5750_Lecture20_unsupervised_learning.pdf (https://canvas.slu.edu/courses/28036/files/2162134?wrap=1)**
2. **https://github.com/ageron/handson-ml2/ (https://github.com/ageron/handson-ml2/)**

Note:
**Unsupervised learning** is a class of machine learning techniques for discovering patterns in data (without looking into labels).
For instance, finding the "clusters" of texts/words based on their semantic meanings (e.g., home & house, Saint Louis & Columbia).

Two representative examples in unsupervised learning is called **clustering** and **dimension reduction**. A good visualization example is here: https://projector.tensorflow.org/

The notebook is organized as follows:
- We need to get the features or vector representations of each example (e.g., embedding features of words, pixels of images).
- All examples will be saved in a data matrix, where each row is the example (e.g., words, image), and each column is the feature of each example (e.g., embeddings, pixels).
- We can explore dimension reduction techniques on training data

- We can cluster the examples using **k-means clustering**. K-means finds a specific number of clusters in the samples based on the distance measurements among data. You can call this function in the 'sklearn' library.

**Note:**

- This hands-on activity reflects the key contents in Chapter 8 & 9 of the textbook
- Please follow the tutorial to practice the codes. The work will be counted as bonus credits to part of course participation together (with reading sessions)
- We suggest everyone practice the codes and submit the results. You can copy texts/upload images from Google Colab and paste into the textbox.
- You can always leave this survey anytime without clicking submit button. Canvas will automatically save your work for latter use.

## 問題 1

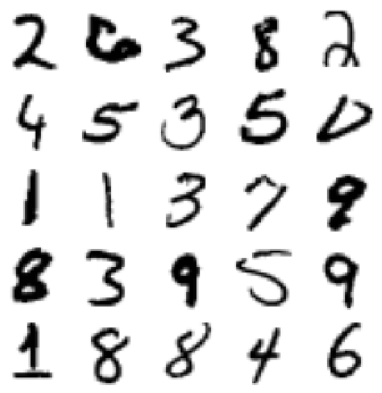
**5 分數**

### Step 1: Data Loading

```
# Step 1.1: Data Loading
from keras.datasets import mnist # MNIST dataset is included in Keras
import numpy as np

# The MNIST data is split between 60,000 28 x 28 pixel training images and 10,000
28 x 28 pixel images
(X_train, y_train), (X_test, y_test) = mnist.load_data()

print("X_train original shape", X_train.shape)
print("y_train original shape", y_train.shape)

X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

print("X_train shape", X_train.shape)
print("y_train shape", y_train.shape)
print("X_test shape", X_test.shape)
print("y_test shape", y_test.shape)
```

```
# Step 1.2: Data Visualization
def plot_digits(instances, images_per_row=5, **options):
    import matplotlib.pyplot as plt
    import matplotlib as mpl
    size = 28
    images_per_row = min(len(instances), images_per_row)
    images = [instance.reshape(size,size) for instance in instances]
    n_rows = (len(instances) - 1) // images_per_row + 1
    row_images = []
    n_empty = n_rows * images_per_row - len(instances)
    images.append(np.zeros((size, size * n_empty)))
    for row in range(n_rows):
       rimages = images[row * images_per_row : (row + 1) * images_per_row]
       row_images.append(np.concatenate(rimages, axis=1))
    image = np.concatenate(row_images, axis=0)
    plt.imshow(image, cmap = mpl.cm.binary, **options)
    plt.axis("off")
```

```
plot_digits(X_train[0:25])
```



**Task: run the above codes, check if you can get similar image. Write codes to check the dimension of X_train, and X_test**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p 0 個字

問題 2 5 分數

**Step 2: Find M PCs to reduce the dimensionality of the data**

```
# Step 2.1: Find M PCs to reduce the dimensionality of the data
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)  # Let M = 2
pca.fit(X_train)
```

```
# Step 2.2: Check the principal component matrix
print(pca.components_)
```

```
# Step 2.3: Check the shape of principal component matrix
print(pca.components_.shape)  # dimension: 2* 784
```

**Task: Explain what does each dimension of principal component matrix mean. What if we change n_components = 2 to n_components = 5**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p

0 個字 </>

## 問題 3

5 分數

### Step 3: Transform the original data into lower-dimension using PCA

```
# Step 3.1: Transform the original data into lower-dimension (M=2)
Reduced_X_train = pca.transform(X_train)
# Reduced_X_train.shape # i.e., (52500, 2)
```

**Task: Print out shape of Reduced_X_train. Explain what does each dimension in Reduced_X_train represent.**

p

0 個字 </>

## 問題 4

5 分數

### Step 4:Examine the PCA transformation process (review course slides)

```
# Step 4a: Transform the original data into lower-dimension (M=2) using PCA
reduced_X_train = pca.transform(X_train)

# Step 4b: Transform the original data into lower-dimension (M=2) using dot produ
ct
reduced_X_train_compare = X_train.dot(pca.components_.T)
# added transpose to pca.components_ make sure each column represent the PC
```

```
# Step 4c: Transform the original data into lower-dimension (M=2) using data cent
ering dot product
X_train_center = X_train - X_train.mean(axis=0)
reduced_X_train_reproduce = X_train_center.dot(pca.components_.T)
```

Note: you can also get the mean of features from pca.mean_, so the following codes would also work

```
X_train_center2 = X_train - pca.mean_
reduced_X_train_reproduce2 = X_train_center.dot(pca.components_.T)
```

**Task: Compare reduced_X_train, reduced_X_train_compare, reduced_X_train_reproduce, and reduced_X_train_reproduce2. Describe the observation and explain why reduced_X_train_compare didn't get same features as reduced_X_train**

p

0 個字 </>

## 問題 5

5 分數

### Step 5: Apply PCA on test set

```
# Step 5a: Transform the test set using PCA
reduced_X_test = pca.transform(X_test)
```

```
# Step 5b: Transform the test set using using centering and dot product
X_test_centered = X_test - X_train.mean(axis=0) # use mean from training set
reduced_X_test_compare = X_test_centered.dot(pca.components_.T)

# Step 5c: Transform the test set using using centering and dot product
X_test_centered2 = X_test - pca.mean_ # use mean from training set
reduced_X_test_compare2 = X_test_centered2.dot(pca.components_.T)
```

**Task: Do the three reduced datasets have same values?**

**From step 4 and step 5, summarize the process if we want to reproduce the data transformation using dot product to get same results as PCA in scikit-learn.**

p

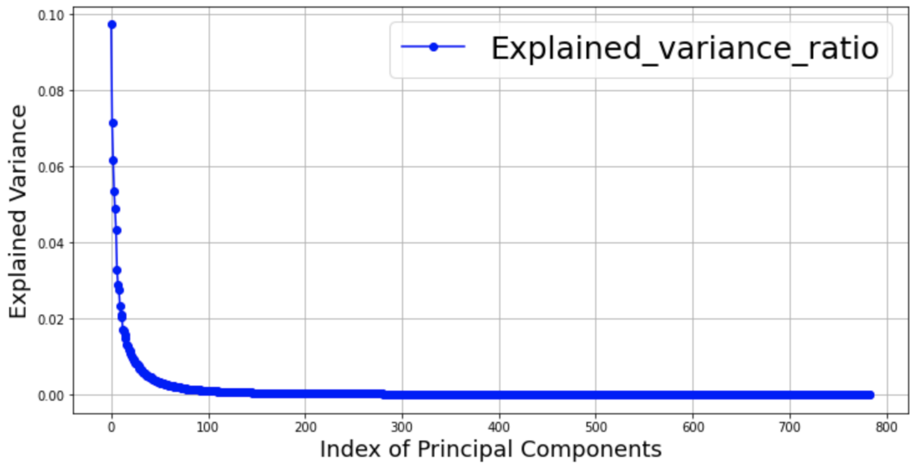
0 個字

## 問題 6

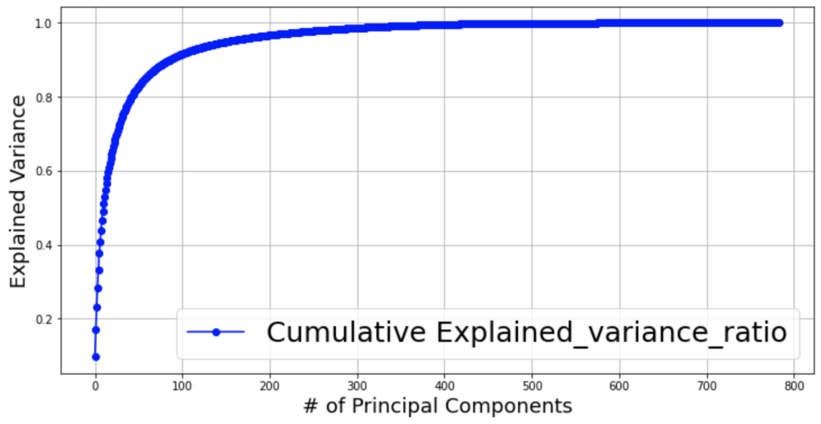
5 分數

### Step 6: Draw the Explained Variance Ratio

```
# Step 6.1: We first should fit PCA on the training set using all principal compo
nents (without setting n_component)
pca = PCA()
pca.fit(X_train)


# Step 6.2: Draw the explained variance ratio for each principal component
import matplotlib.pyplot as plt
plt.figure(figsize=(12,6))
plt.plot(pca.explained_variance_ratio_, marker='o', color='b', label='Explained_v
ariance_ratio')
plt.xlabel('Index of Principal Components', fontsize = 18)
plt.ylabel('Explained Variance', fontsize = 18)
plt.grid(True)
plt.legend(fontsize=25)
```



```
# Step 6.3: Draw the cumulative explained variance ratio for all principal compon
ents
import matplotlib.pyplot as plt
plt.figure(figsize=(12,6))
plt.plot(np.cumsum(pca.explained_variance_ratio_), marker='o', color='b', label
='Cumulative Explained_variance_ratio')
plt.xlabel('# of Principal Components', fontsize = 18)
plt.ylabel('Explained Variance', fontsize = 18)
plt.grid(True)
plt.legend(fontsize=25)
```



**Task: Run the above codes, guess where the minimum number of principal components should be to keep at least 90% variance.**

p

0 個字

問題 7

2 分數

## Step 7: Run PCA compression on images to get lower-dimension features

```
# Step 7.1: to identify the minimum number of PCs to achieve expected variance
(i.e., 95%), we can set a float value for n_components to get the expected number
pca = PCA(n_components=0.95)
X_reduced = pca.fit_transform(X_train)
```

```
# Step 7.2: check the expected number of PCs
print('Minimum number of PCs:', pca.n_components_)
print('Cumulative variance ratio: ',np.sum(pca.explained_variance_ratio_))
```

```
# Step 7.3: check the shape of X_reduced
print(X_reduced.shape)
```

**Task: Run the above codes, and describe the optimal number of PCs to achieve 95% variance. What's the new feature dimension of transformed data?**

p

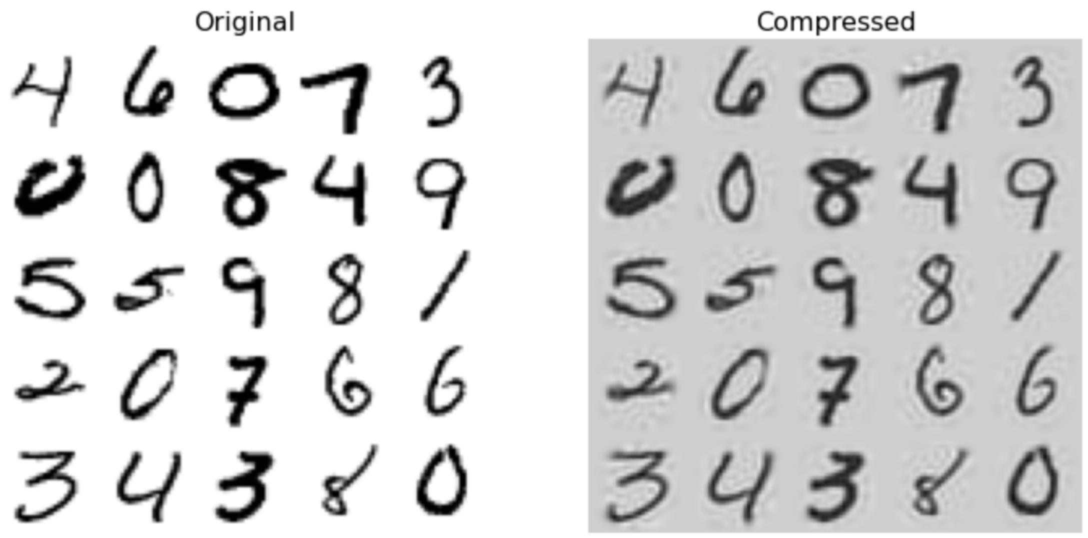
0 個字

**問題 8** **5 分數**

**Step 8: Reconstruct the original data from the lower-dimension data**

```
# Step 8.1: Reconstruct the original data from the lower-dime
nsion data
pca = PCA(n_components = 154)
X_reduced = pca.fit_transform(X_train)
X_recovered = pca.inverse_transform(X_reduced)
print("X_recovered: ", X_recovered.shape)
```

```
# Step 8.2: Visualize the original data and reconstructed dat
a
plt.figure(figsize=(12, 8))
plt.subplot(121)
plot_digits(X_train[0:25])
plt.title("Original", fontsize=16)
plt.subplot(122)
plot_digits(X_recovered[0:25])
plt.title("Compressed", fontsize=16)
plt.show()
```



**Task: Write codes to calculate the mean squared errors between the original data X_train and recovered data X_recovered.**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p 0 個字

## 問題 9

10 分數

**Step 9: Compare ML algorithms on original data and compressed data**

```
# Step 9.1: Load the library
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from keras.datasets import mnist # MNIST dataset is included in Keras
import numpy as np


# Step 9.2: Load the dataset
# The MNIST data is split between 60,000 28 x 28 pixel training images and 10,000
28 x 28 pixel images
(X_train, y_train), (X_test, y_test) = mnist.load_data()
print("X_train original shape", X_train.shape)
print("y_train original shape", y_train.shape)
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
print("X_train shape", X_train.shape)
print("y_train shape", y_train.shape)
print("X_test shape", X_test.shape)
print("y_test shape", y_test.shape)
```

```
# Step 9.3: Scale the data
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Step 9.4: Apply PCA to reduce the dimension
pca = PCA(n_components = 154)
X_train_reduced = pca.fit_transform(X_train_scaled)
X_test_reduced = pca.transform(X_test_scaled)
```

```
# Step 9.5: Fit random forest on the original data
from sklearn.ensemble import RandomForestClassifier
model_original = RandomForestClassifier()
model_original.fit(X_train_scaled, y_train)
print("(Original) Training accuracy: ", accuracy_score(y_train,model_original.pre
dict(X_train_scaled)))
print("(Original) Testing accuracy: ", accuracy_score(y_test,model_original.predi
ct(X_test_scaled)))
```

```
# Step 9.6: Fit random forest on the reduced data
from sklearn.ensemble import RandomForestClassifier
model_reduced = RandomForestClassifier()
model_reduced.fit(X_train_reduced, y_train)
print("(PCA) Training accuracy: ",accuracy_score(y_train,model_reduced.predict(X_
train_reduced)))
print("(PCA) Testing accuracy: ", accuracy_score(y_test,model_reduced.predict(X_t
est_reduced)))
```

**Task: Report the difference between the accuracy of ML models.**

**Optional Tasks:**

**Try different ML algorithms**
**Try Parameter Tuning**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p 0 個字

**問題 10** **5 分數**

### Step 10: Combine Dimension Reduction with Clustering analysis

To start the clustering analysis,

1). we can import kmeans from sklearn

2). Then create a kmeans model, **specifying the number of clusters** you want to find

3). Let's specify 10 clusters since we have 10 different groups of digits.
Here we use the same scaled dataset **X_train_scaled** in our Step 9.3:

```
# Step 10.1: fit the model on the scaled data (X_train_scale
d)
# Note: 1~2 minutes to run on this data to get the mean of op
```

```
timal clusters (centroid).
from sklearn.cluster import KMeans
kmeans_model = KMeans(init='k-means++', n_clusters=10)
kmeans_model.fit(X_train_scaled)
```

```
# Step 10.2: After the clustering is done, we can predict the
cluster labels.
# This will returns a cluster label for each sample, indicati
ng to which cluster a sample belongs.
# Note: this is the class label, not image label
```

```
train_cluster_labels = kmeans_model.predict(X_train_scaled)
print("train_cluster_labels: ", train_cluster_labels)
```

```
# Step 10.3: We can also use the model to predict cluster lab
els for any new test data.
# The new data points will be assigned to clusters whose cent
roid is closest.

X_test_scaled=scaler.transform(X_test)
test_cluster_labels = kmeans_model.predict(X_test_scaled)
print("test_cluster_labels: ", test_cluster_labels)
```

**Task: Run the above codes, and the check the labels in the variables train_cluster_labels and test_cluster_labels. Explain what does each label value represent in clustering analysis.**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p

0 個字 </>

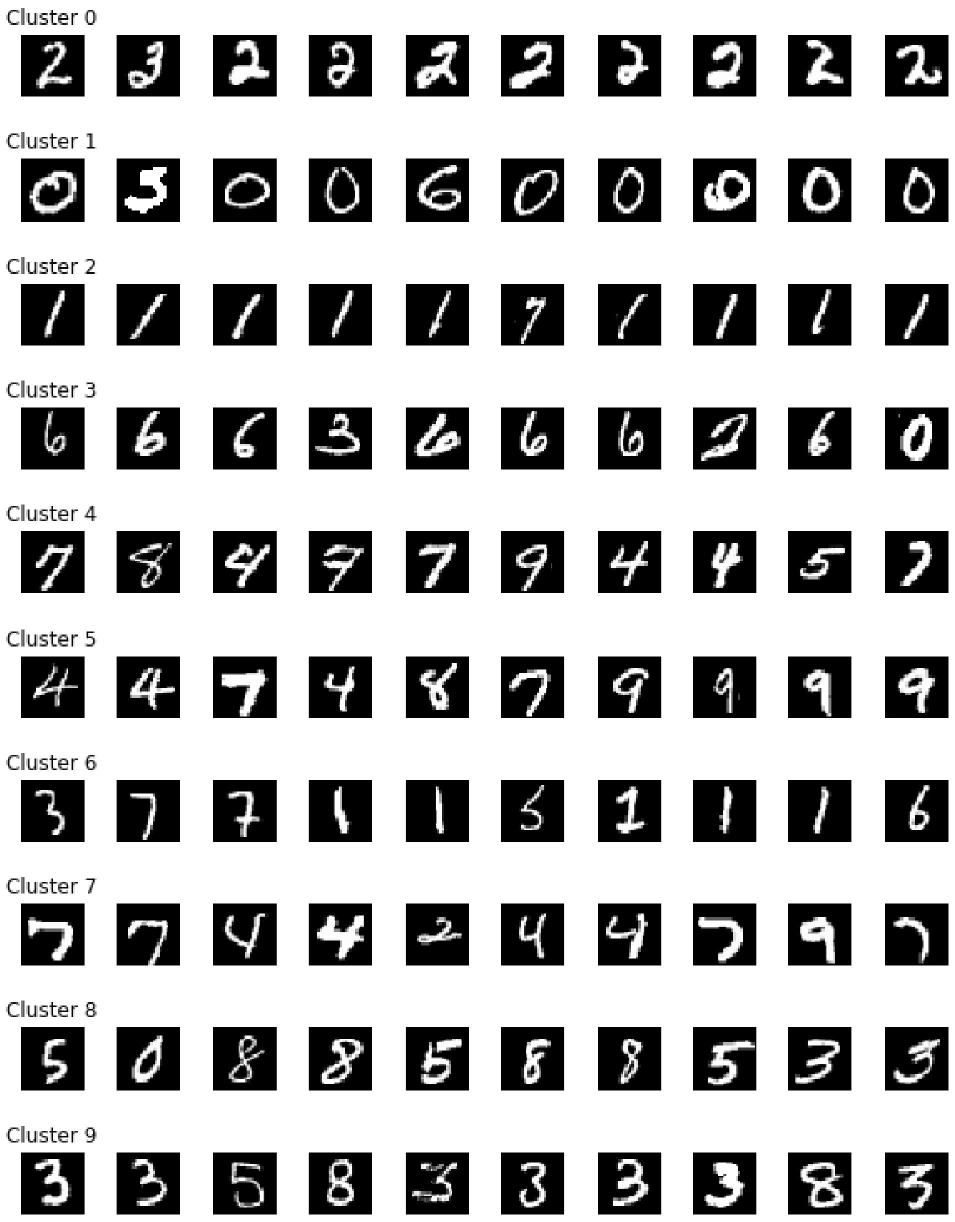
## 問題 11

5 分數

### Step 10.4 Visualize the images in each cluster from K-means

Note: If the clustering is accurate, we expect to see same digit images are grouped into same cluster

```
# Step 10.4: visualize the images in each cluster from K-means
img_idx = 0
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(10,10))
plt.subplots_adjust(top = 0.99, bottom=0.01, hspace=1, wspace=0.4)
for cluster_id in set(train_cluster_labels):
    cluster_images = X_train_scaled[train_cluster_labels==cluster_id] ## get imag
es within same clusters

    #select first 10 for visualization

    for i in range(10):
        plt.subplot(10,10,img_idx+1)
        plt.imshow(cluster_images[i].reshape(28,28), cmap='gray', interpolation
='none')
        plt.axis('off')
        if i == 0: plt.title("Cluster {}".format(cluster_id))
        img_idx += 1
```



**Task: Run the above codes, and summarize whether the clustering makes sense or not.**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p

0 個字

問題 12

2 分數

**Step 10.5: Evaluate the clustering accuracy (Assume the true labels are known)**

Use contingency table to count the number of mismatches among clusters.
The cross tabulations can tell us which sort of samples are in which clusters.

```
# Step 10.5 Evaluate the clustering accuracy (Assume the true
labels are known)
import pandas as pd
evaluation_df = pd.DataFrame({'cluster_labels':train_cluster_
labels, 'true_labels': y_train })
print(evaluation_df)
contingency_table = pd.crosstab(evaluation_df['cluster_label
s'],evaluation_df['true_labels'])
contingency_table
```

| true_labels | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| cluster_labels | | | | | | | | | | |
| **0** | 34 | 8 | 3685 | 181 | 19 | 13 | 80 | 32 | 37 | 8 |
| **1** | 4145 | 0 | 62 | 23 | 0 | 50 | 70 | 11 | 24 | 26 |
| **2** | 24 | 2604 | 341 | 67 | 179 | 594 | 169 | 204 | 338 | 67 |
| **3** | 180 | 9 | 142 | 54 | 111 | 115 | 4078 | 6 | 52 | 7 |
| **4** | 24 | 4 | 56 | 36 | 1309 | 308 | 1 | 2374 | 210 | 1400 |
| **5** | 9 | 8 | 37 | 114 | 1402 | 231 | 1 | 2124 | 166 | 2172 |
| **6** | 0 | 3319 | 272 | 309 | 90 | 90 | 219 | 170 | 225 | 104 |
| **7** | 59 | 3 | 162 | 62 | 2044 | 183 | 338 | 532 | 110 | 1314 |
| **8** | 474 | 6 | 130 | 1044 | 8 | 1638 | 132 | 2 | 2773 | 45 |
| **9** | 197 | 10 | 369 | 3423 | 0 | 1533 | 24 | 2 | 1178 | 72 |

**Task: Run the above codes. Could you tell which representative digit for each cluster to be labeled?**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p

0 個字

**問題 13** **5 分數**

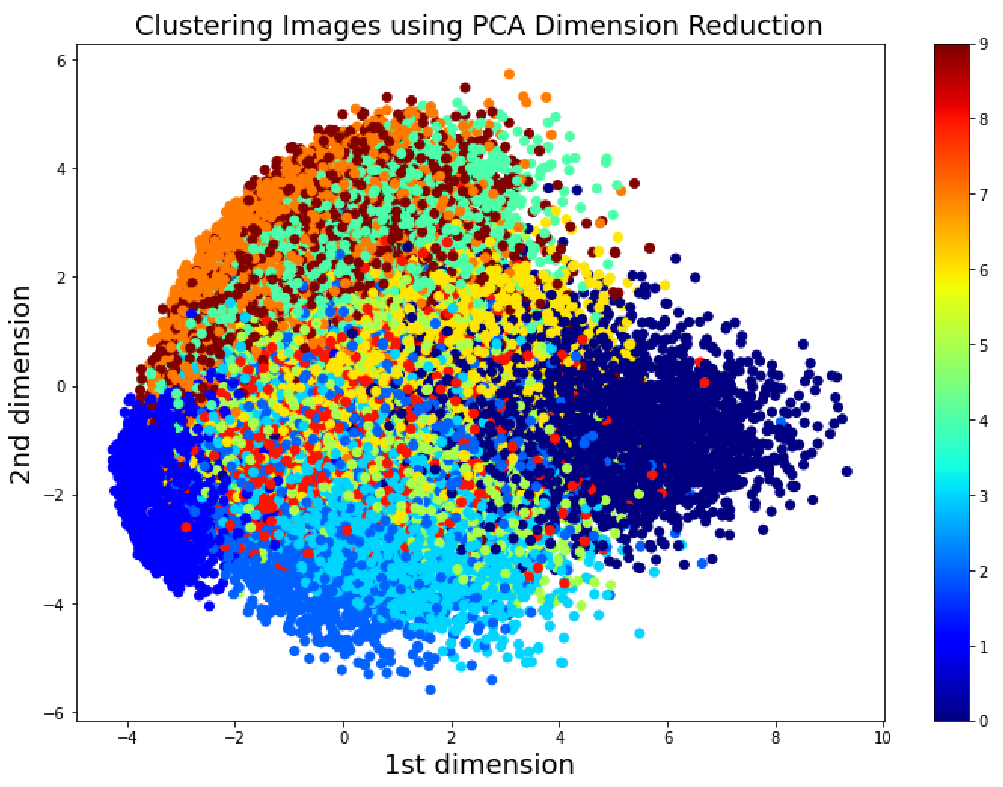
## Step 10.6. Then we can visualize the clustering of the data using scatter plots.

Here we can apply the **principal component analysis (PCA)** to reduce the dimension for visualization.

We can try 2-dimension visualization and 3-dimension visualization

```
# Step 10.6  For visualization, we need to apply PCA for dimension reduction
# https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.htm
l

import numpy as np
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(X_train_scaled)
pca_data = pca.transform(X_train_scaled)
```

```
# treat the first two PCs as x and y axis
import matplotlib.pyplot as plt
plt.figure(figsize=(12,8))
plt.subplots_adjust(bottom = 0.1)
plt.scatter(pca_data[:, 0], pca_data[:, 1], c= y_train, cmap='jet')
plt.xlabel("1st dimension", fontsize=18)
plt.ylabel("2nd dimension", fontsize=18)
plt.title("Clustering Images using PCA Dimension Reduction", fontsize=18)
plt.colorbar()
plt.show()
```



**Task: Run the above codes, describe your observations. Does PCA make any number distinguishable?**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p

0 個字

## 問題 14

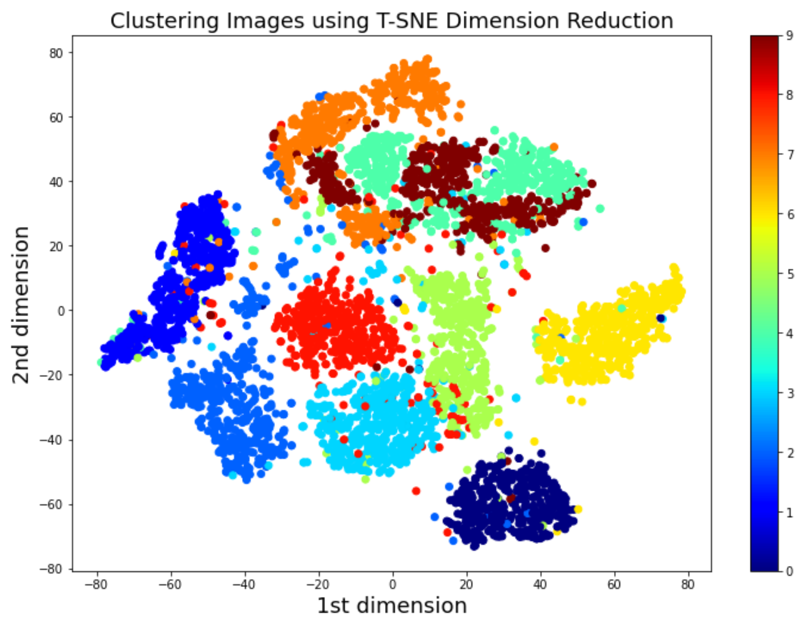
5 分數

### Step 10.7 Visualize the clusters using T-SNE dimension reduction

```
# Step 10.7 Visualize the clusters using T-SNE dimension reduction
from sklearn.manifold import TSNE
tsne = TSNE(n_components=2, random_state=42)

# Dimensionality reduction on the full 60,000 images takes a very long time, so l
et's only do this on a random subset of 10,000 images:
subset_X_train_scaled = X_train_scaled[0:5000]
subset_y_train = y_train[0:5000]
X_tsne_reduced = tsne.fit_transform(subset_X_train_scaled)
```

```
### visualize the T-SNE coordinates
import matplotlib.pyplot as plt
plt.figure(figsize=(12,8))
plt.subplots_adjust(bottom = 0.1)
plt.scatter(X_tsne_reduced[:, 0], X_tsne_reduced[:, 1], c= subset_y_train, cmap
='jet')  # color by real labels
```

```
plt.colorbar()
plt.xlabel("1st dimension", fontsize=18)
plt.ylabel("2md dimension", fontsize=18)
plt.title("Clustering Images using T-SNE Dimension Reduction", fontsize=12)
plt.show()
```



**Task: Run the above codes, describe the following observations:**

- **Does T-SNE make number more distinguishable than PCA?**
- **Could you tell which numbers are well separated?**
- **Or which numbers are hard to get distinguished based on the visualization?**
- **Observe any difference in running time compared to PCA?**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p

0 個字 </>

## Step 11: Combine feature extraction, clustering, and dimension reduction

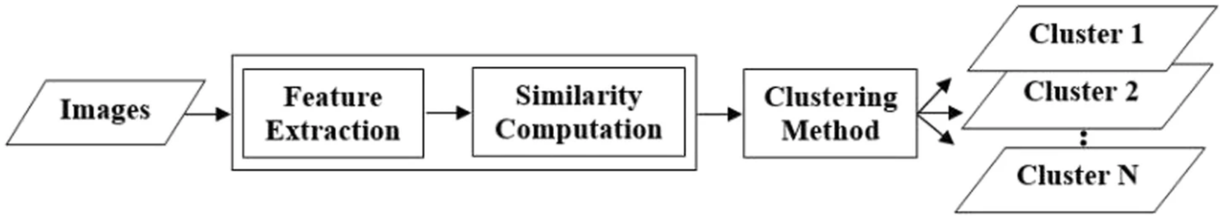
**Observation**: Directly applying kmeans on MNIST image data is not working well.

Good features/measurements of samples for clustering are important. In other words, the accuracy of a clustering algorithm depends on the features you are using.

**Another option is to use deep learning methods to extract hidden features from the image data, and then apply the clustering algorithm on those extract features.**

The general strategy is followed as:

1. Pre-trained deep learning model(e.g., Neural network, CNN) for image feature extraction. The deep learning model can be derived from other image-based dataset.
2. Extracted features(neuron values for each image) from the last second layer which best describe digit images.
3. Clustered using KMeans algorithm from the extracted feature vector.



https://keras.io/api/applications/

**問題 15** **5 分數**

## Step 11.1 Train one neural network on the training images.

For demonstration purpose, here we can train one neural network -based  image-classification model on MNIST data first

(For any other color images or using convolution-based neural network, you can also use pre-trained deep learning models provides on https://keras.io/api/applications/

Let's use the codes in [MNIST tutorial](https://github.com/wxs/keras-mnist-tutorial/blob/master/MNIST%20in%20Keras.ipynb) to train the model.

```
# Step 11.1: Load the library
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from keras.datasets import mnist # MNIST dataset is included in Keras
from keras.utils import np_utils
import numpy as np


# Step 11.2: Load the dataset
# The MNIST data is split between 60,000 28 x 28 pixel training images and 10,000
28 x 28 pixel images
(X_train, y_train), (X_test, y_test) = mnist.load_data()
print("X_train original shape", X_train.shape)
print("y_train original shape", y_train.shape)
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
print("X_train shape", X_train.shape)
print("y_train shape", y_train.shape)
print("X_test shape", X_test.shape)
print("y_test shape", y_test.shape)
```

```
# Step 11.3: Scale the data
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Step 11.4: Generate one-hot encoding for labels
nb_classes = 10
Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)
```

```
# Step 11.5: Create one neural network using Keras
import numpy as np
import matplotlib.pyplot as plt
from keras.utils import np_utils
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.utils import np_utils

model = Sequential()
model.add(Dense(512, input_shape=(784,)))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dense(10))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=["accura
cy"])
model.summary()
```

**Task 1: Examine the data shape of variables X_train_scaled and X_test_scaled. Make sure the data shape fits the required dimension to the neural network.**
**Task 2: Check the number of neurons in the hidden layers. We will re-use these hidden neurons in the later step.**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p 0 個字

**問題 16** 5 分數

## Step 11.2: Train the neural network using training dataset.

```
#step 11.2: training the network
model.fit(X_train_scaled, Y_train, validation_split = 0.1, ba
tch_size=128, epochs=10, verbose=1)
```

```
# Step 11.2: Evaluate on the test set
score = model.evaluate(X_test_scaled, Y_test)
print('Test score:', score)
```

**Task: Examine the training accuracy.**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p 0 個字

**問題 17** 5 分數

## Step 11.3: Feature extraction from Neural network model

Let's create a function to get the last hidden layers from the model for feature extraction

```
### Step 11.3a: get all layers of the model and save into a list
model_layers = model.layers
print("model_layers:", model_layers)
```

```
###Step 11.3b: Let's select last second layer as the output for feature extractor
# model.layers[-3].output corresponds to the output of last second layer
print("model.layers[-3].output:", model.layers[-3].output)
```

```
###Step 11.3c: select the first layer as the input of feature extractor
# model.layers[0].input corresponds to the input of the first layer
print("model.layers[0].input: ", model.layers[0].input)
```

```
###Step 11.3d: Then we can build a feature extractor function to map the input to
feature vectors
from keras import backend as K
Get_Hidden_Layered_Output = K.function([model.layers[0].input], [model.layers[-
3].output])

###  Let's apply the feature extractor on one image, and check the feature vector
X_test_featureVector = Get_Hidden_Layered_Output([X_test_scaled])[0]

print("extracted_features:", X_test_featureVector)
print("shape of original features:", X_test_scaled.shape)
print("shape of extracted features:", X_test_featureVector.shape)
```

**Task: Double-check what's the new dimension of the hidden features. We will use this extracted feature for clustering analysis on test set**

編輯　檢視　插入　格式　工具　表格

12pt　段落

p

0 個字

問題 18

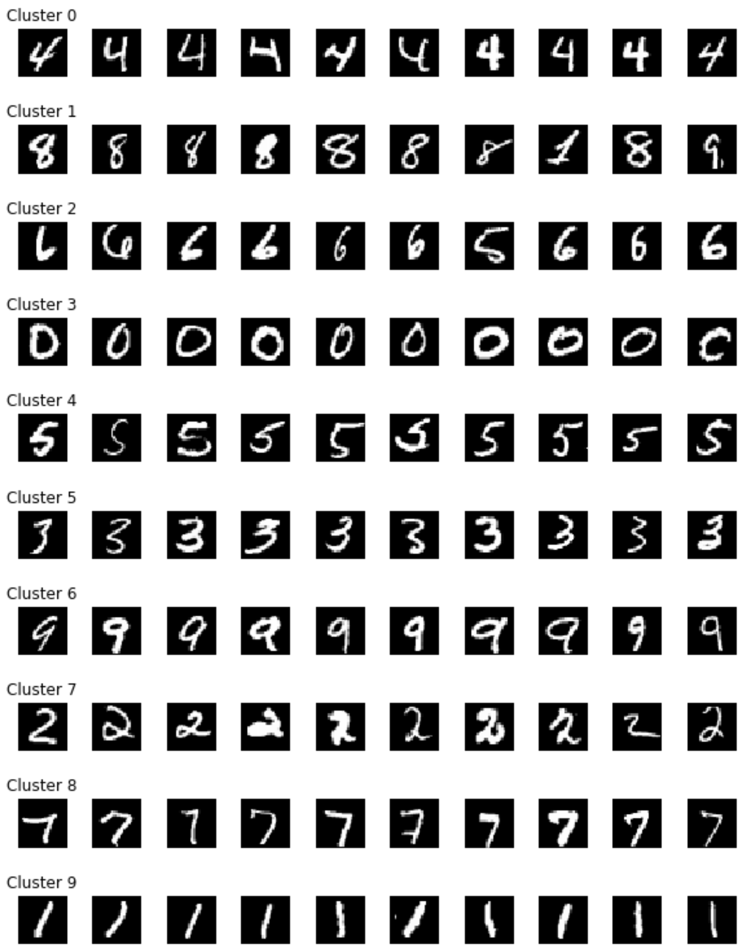
5 分數

## Step 11.4. Apply the clustering algorithm on the hidden features of the images

As you can see, the dimension of test data points are reduced from 784 to 128 after applying the feature extraction.
We can apply k-means on the new features

```
# Step 11.4a Apply the clustering algorithms on the hidden features of the images
from sklearn.cluster import KMeans
kmeans_model = KMeans(init='k-means++', n_clusters=10)
kmeans_model.fit(X_test_featureVector)
```

```
# Step 11.4b  Get clustering labels and visualize the clusters
pred_cluster_labels = kmeans_model.predict(X_test_featureVector)
img_idx = 0
fig = plt.figure(figsize=(10,10))
plt.subplots_adjust(top = 0.99, bottom=0.01, hspace=1, wspace=0.4)
for cluster_id in set(pred_cluster_labels):
    cluster_images = X_test_scaled[pred_cluster_labels==cluster_id]
    #select first 10 for visualization
    for i in range(10):
        plt.subplot(10,10,img_idx+1)
        plt.imshow(cluster_images[i].reshape(28,28), cmap='gray', interpolation
='none')
        plt.axis('off')

        if i == 0: plt.title("Cluster {}".format(cluster_id))
        img_idx += 1
```



**Task: Compared to the output in Step 10.4, does each cluster have more consistent instances?**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p 0 個字 </>

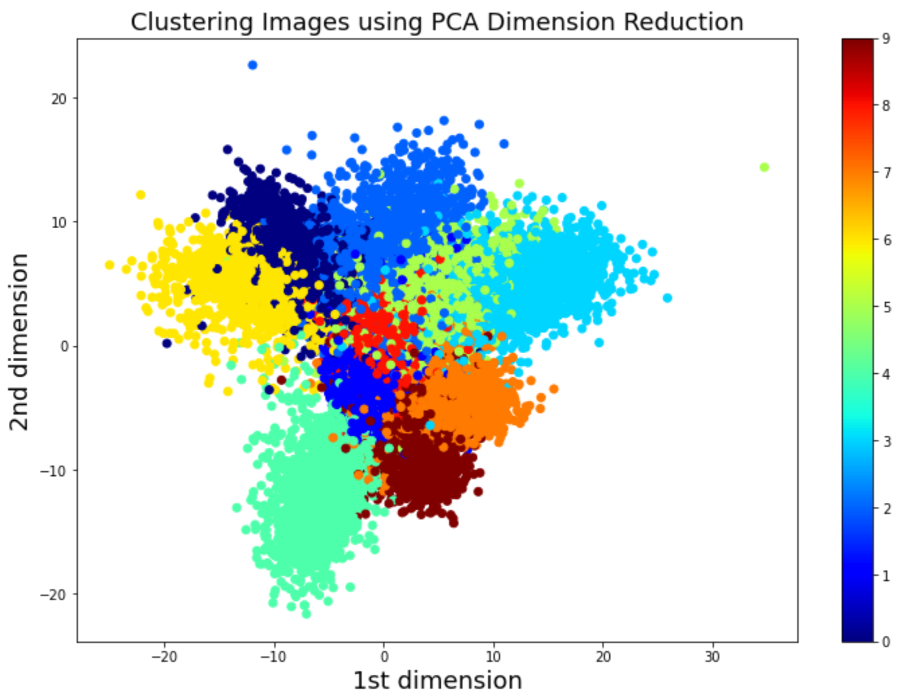
## 問題 19

3 分數

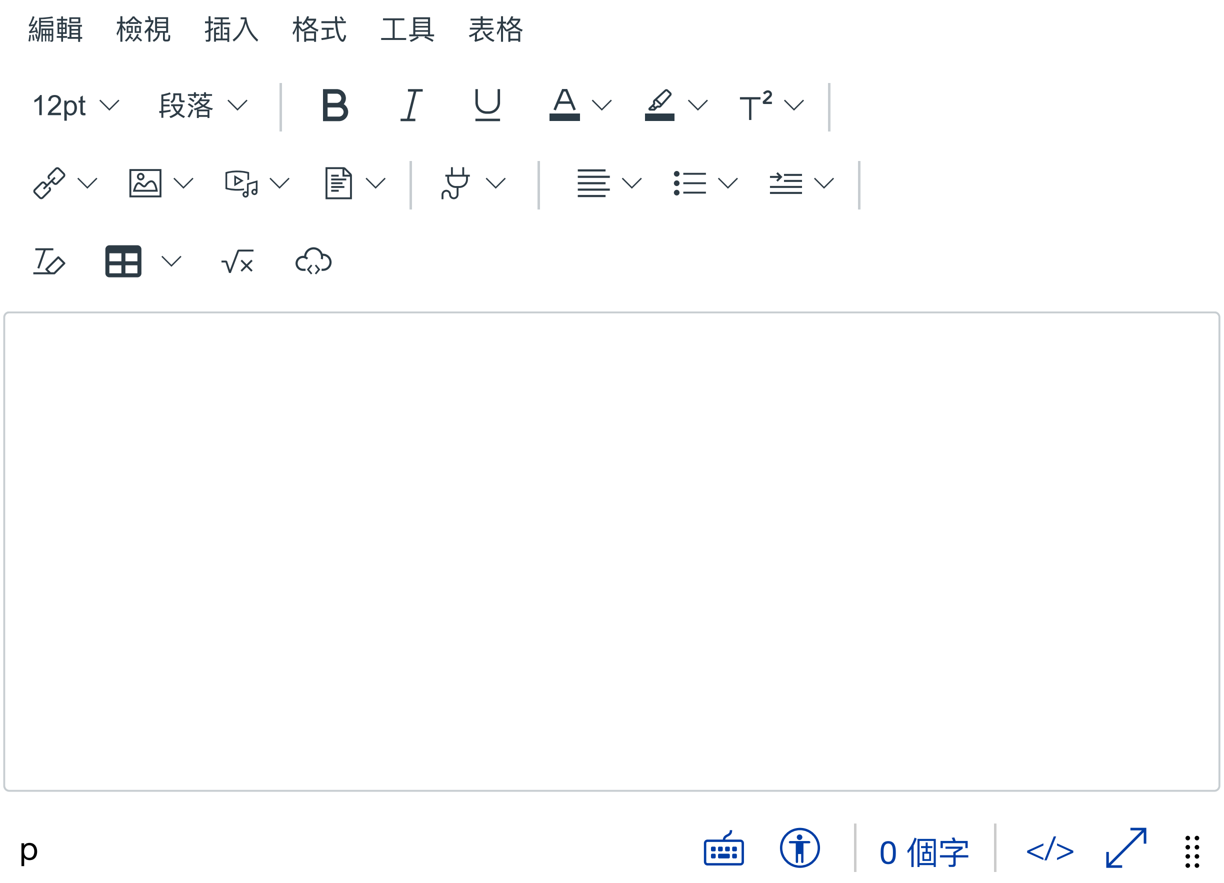
**Step 11.5: Apply PCA for dimension reduction on this extracted features**

```
# Step 11.5: Apply PCA for dimension reduction on this extracted
features
# https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.htm
l

import numpy as np
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(X_test_featureVector)
pca_data = pca.transform(X_test_featureVector)
```

```
## Step 11.5: visualize the PCA clusters
import matplotlib.pyplot as plt
plt.figure(figsize=(12,8))
plt.subplots_adjust(bottom = 0.1)
plt.scatter(pca_data[:, 0], pca_data[:, 1], c= y_test, cmap='jet') # colored by r
eal label
plt.xlabel("1st dimension", fontsize=18)
plt.ylabel("2nd dimension", fontsize=18)
plt.title("Clustering Images using PCA Dimension Reduction", fontsize=18)
plt.show()
```



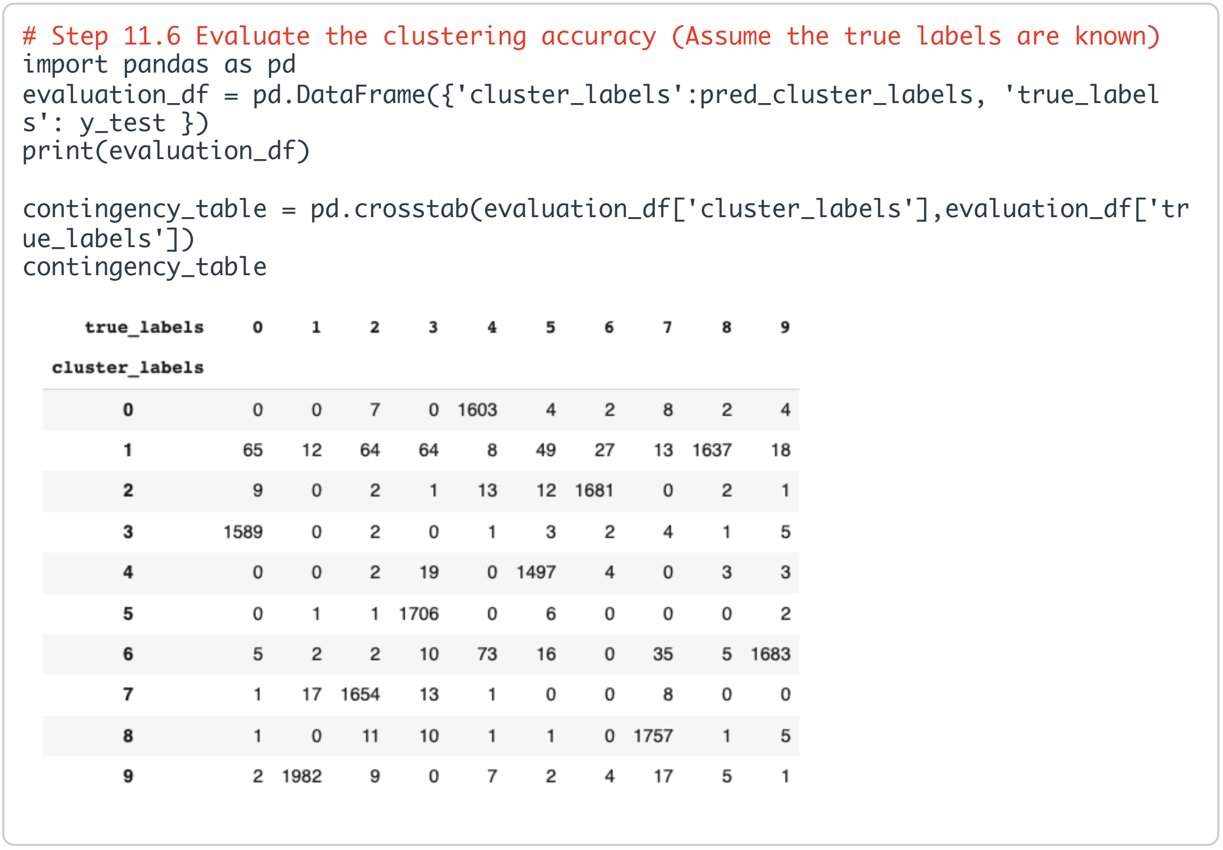
**Task: Compared to the output in Step 10.6, does each cluster have more consistent instances?**



**問題 20** **3 分數**

**Step 11.6: Evaluate the clustering accuracy (Assume the true labels are known)**

Use contingency table to count the number of mismatches among clusters.
The cross tabulations can tell us which sort of samples are in which clusters.



```
# Step 11.6 Evaluate the clustering accuracy (Assume the true labels are known)
import pandas as pd
evaluation_df = pd.DataFrame({'cluster_labels':pred_cluster_labels, 'true_label
s': y_test })
print(evaluation_df)

contingency_table = pd.crosstab(evaluation_df['cluster_labels'],evaluation_df['tr
ue_labels'])
contingency_table
```

| true_labels | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **cluster_labels** | | | | | | | | | | |
| **0** | 0 | 0 | 7 | 0 | 1603 | 4 | 2 | 8 | 2 | 4 |
| **1** | 65 | 12 | 64 | 64 | 8 | 49 | 27 | 13 | 1637 | 18 |
| **2** | 9 | 0 | 2 | 1 | 13 | 12 | 1681 | 0 | 2 | 1 |
| **3** | 1589 | 0 | 2 | 0 | 1 | 3 | 2 | 4 | 1 | 5 |
| **4** | 0 | 0 | 2 | 19 | 0 | 1497 | 4 | 0 | 3 | 3 |
| **5** | 0 | 1 | 1 | 1706 | 0 | 6 | 0 | 0 | 0 | 2 |
| **6** | 5 | 2 | 2 | 10 | 73 | 16 | 0 | 35 | 5 | 1683 |
| **7** | 1 | 17 | 1654 | 13 | 1 | 0 | 0 | 8 | 0 | 0 |
| **8** | 1 | 0 | 11 | 10 | 1 | 1 | 0 | 1757 | 1 | 5 |
| **9** | 2 | 1982 | 9 | 0 | 7 | 2 | 4 | 17 | 5 | 1 |

**Task: Compared to the output in Step 10.5, does each cluster have more consistent instances?**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p

0 個字

## 問題 21

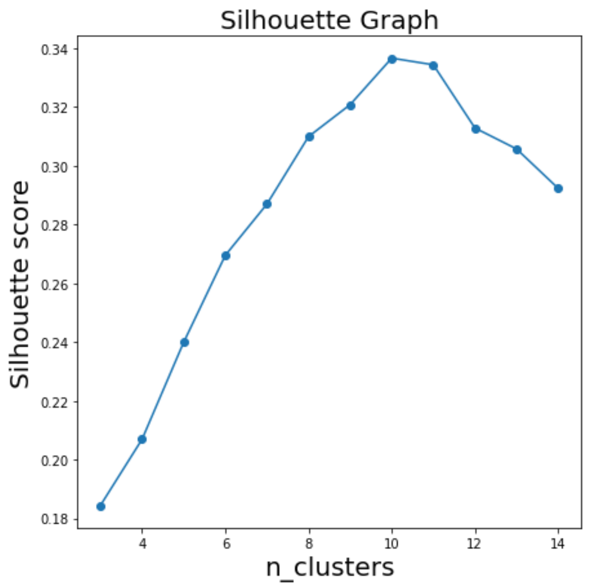
5 分數

### Step 11.7: Explore different clusters

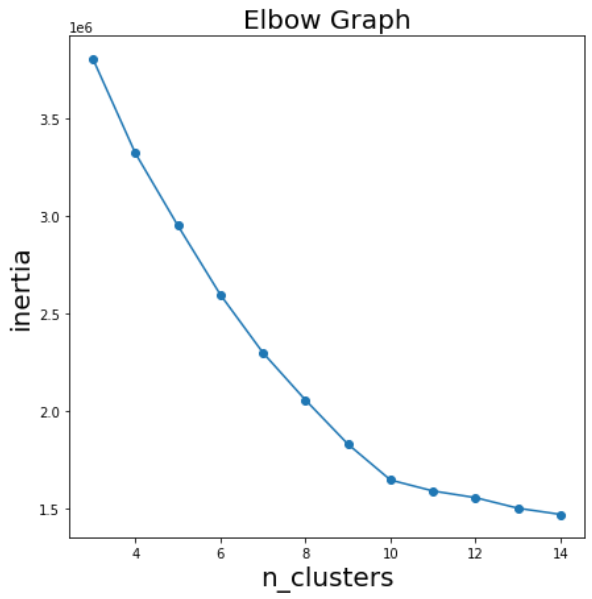
Above steps assume the default number of clusters as 10. We can evaluate the clustering performance when using different number of clusters.

```
# Step 11.7a: Try different number of clusters, and evaluate the clusters using s
ilhousett
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

elbow={"inertia":[],"clusters":[]}
silhouette={"silhouette":[],"clusters":[]}

for i in range(3,15):
   print("Clustering data into ", i, ' groups')
   kmeans_model=KMeans(i)
   elbow["clusters"].append(i)
   kmeans_model.fit(X_test_featureVector)
   elbow["inertia"].append(kmeans_model.inertia_)
   silhouette["silhouette"].append(silhouette_score(X_test_featureVector,kmeans_m
odel.labels_))
```

```
# Step 11.7b:  Visualize the optimal clusters based on Silhou
ette
plt.scatter(x=elbow["clusters"],y=silhouette["silhouette"])
plt.plot(elbow["clusters"], silhouette["silhouette"])
plt.xlabel("n_clusters", size=20)
plt.ylabel("Silhouette score", size=20)
plt.title("Silhouette Graph", size=20)
plt.show()
```



```
# Step 11.7c:  Visualize the optimal clusters based on Inerti
a
plt.scatter(x=elbow["clusters"],y=elbow["inertia"])
plt.plot(elbow["clusters"], elbow["inertia"])
plt.xlabel("n_clusters", size=20)
plt.ylabel("inertia", size=20)
plt.title("Elbow Graph", size=20)
plt.show()
```



**Task: Describe what's the optimal clusters from the above analysis.**

編輯 檢視 插入 格式 工具 表格

12pt 段落

p

0 個字

**Extra Exercise (optional)**

Please apply what you have learned in this demo to cluster flowers in the iris data. The reference link is attached.
[Clustering iris data](https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html (https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html) )

測驗保存於 11:44

提交測驗