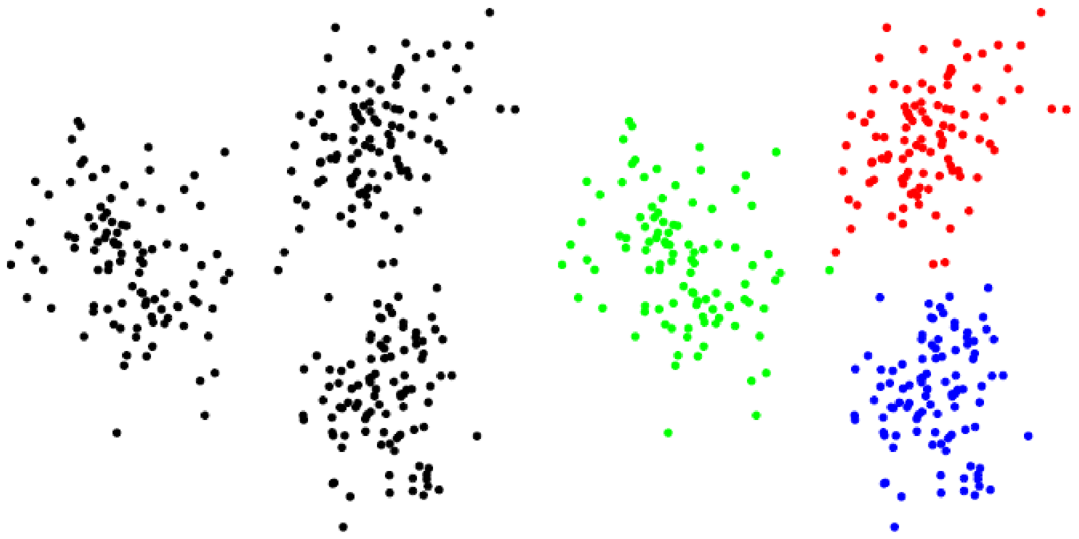
4.1 Introduction to Clustering

Clustering is the process of dividing the entire data into groups (also known as clusters) based on the patterns in the data.

Clustering is very widely used in many application areas, typically (but not always) when the vectors represent features of objects.



300 points in a plane. The points can be clustered in the three groups shown on the right.

Examples.

Survey response clustering. A group of N people respond to a survey with n questions. Each question contains a statement, such as `The movie was too long', followed by some ordered options such as

Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree.

(This is called a Likert scale, named after the psychologist Rensis Likert.) Suppose the n-vector xi encodes the selections of respondent i on the n questions, using the numerical coding -2, -1, 0, +1, +2 for the responses above. A clustering algorithm can be used to cluster the respondents into k groups, each with similar responses to the survey.

4.2 A clustering objective

Specifying the cluster assignments. We specify a clustering of the vectors by saying which cluster or group each vector belongs to. We label the groups 1; ....; k, and specify a clustering or assignment of the N given vectors to groups using an N-vector c, where ci is the group (number) that the vector xi is assigned to. As a simple example with N = 5 vectors and k = 3 groups, c = (3, 1, 1, 1, 2) means that x1 is assigned to group 3, x2, x3, and x4 are assigned to group 1, and x5 is assigned to group 2. We will also describe the clustering by the sets of indices for each group. We let Gj be the set of indices corresponding to group j. For our simple example above, we have

G1={2,3,4} G2={1} G3={5}

Formally, we can express these index sets in terms of the group assignment vector c as

Gj = {i| ci = j}

which means that Gj is the set of all indices i for which ci = j.

K-means Algorithm

It might seem that we can now solve the problem of choosing the group assignments and the group representatives to minimize Jclust, since we know how to do this when one or the other choice is Fixed. But the two choices are circular, i.e., each depends on the other. Instead we rely on a very old idea in computation: We simply iterate between the two choices. This means that we repeatedly alternate between updating the group assignments, and then updating the representatives, using the methods developed above. In each step the objective Jclust gets better (i.e., goes down) unless the step does not change the choice. Iterating between choosing the group representatives and choosing the group assignments is the celebrated k-means algorithm for clustering a collection of vectors.

Algorithm 4.1 k-means algorithm

given a list of N vectors x1; : : : ; xN, and an initial list of k group representative vectors z1; : : : ; zk repeat until convergence

1. Partition the vectors into k groups. For each vector i = 1; : : : ;N, assign xi to the group associated with the nearest representative.
2. Update representatives. For each group j = 1; : : : ; k, set zj to be the mean of the vectors in group j.

How to Form Clusters

Identify and discover K-categories from the data.

here the important parameter you must figure out is the value of 'K'

Some common methods to find value of K

1.Visual EDA (pairplot)

2.The concept of linearity(weather the data is linearly seperable or not.

3.Elbow method.

algorithm (k-means)

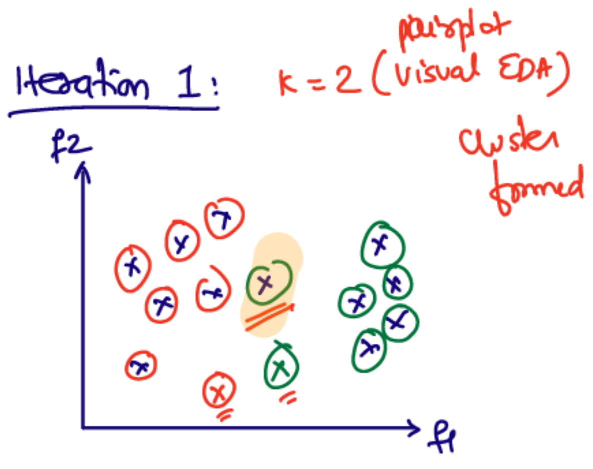
1. consider value of k (by above method or else take any value of K)

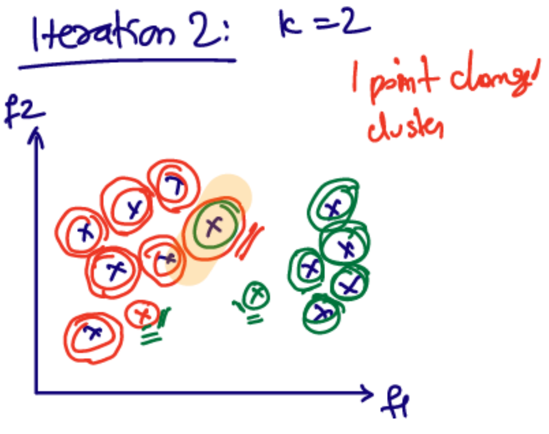
2.create K cluster center point randomly in given dataset

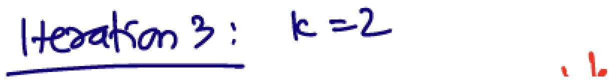
3.using distance formula identify and group each data point to the cluster center

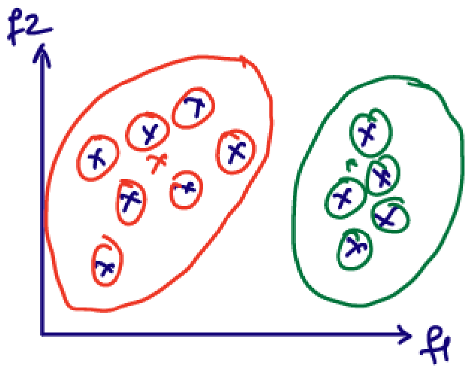
4.change the cluster center based on nearest perpendicularity

1. do this until you dont find any change in group.









no points changed the cluster

∴ our iteration stops resulting finalized cluster

In [6]:

```
#Clustering
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [3]:

```
data= pd.read_csv('Mall_Customers.csv')
```

In [4]:

```
data.head()
```

In [5]:

```
data.info()
```

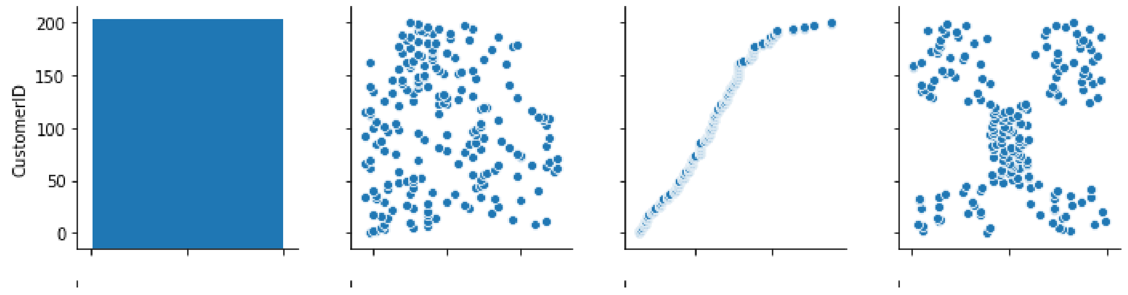
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
CustomerID                200 non-null int64
Genre                     200 non-null object
Age                       200 non-null int64
Annual Income (k$)        200 non-null int64
Spending Score (1-100)    200 non-null int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

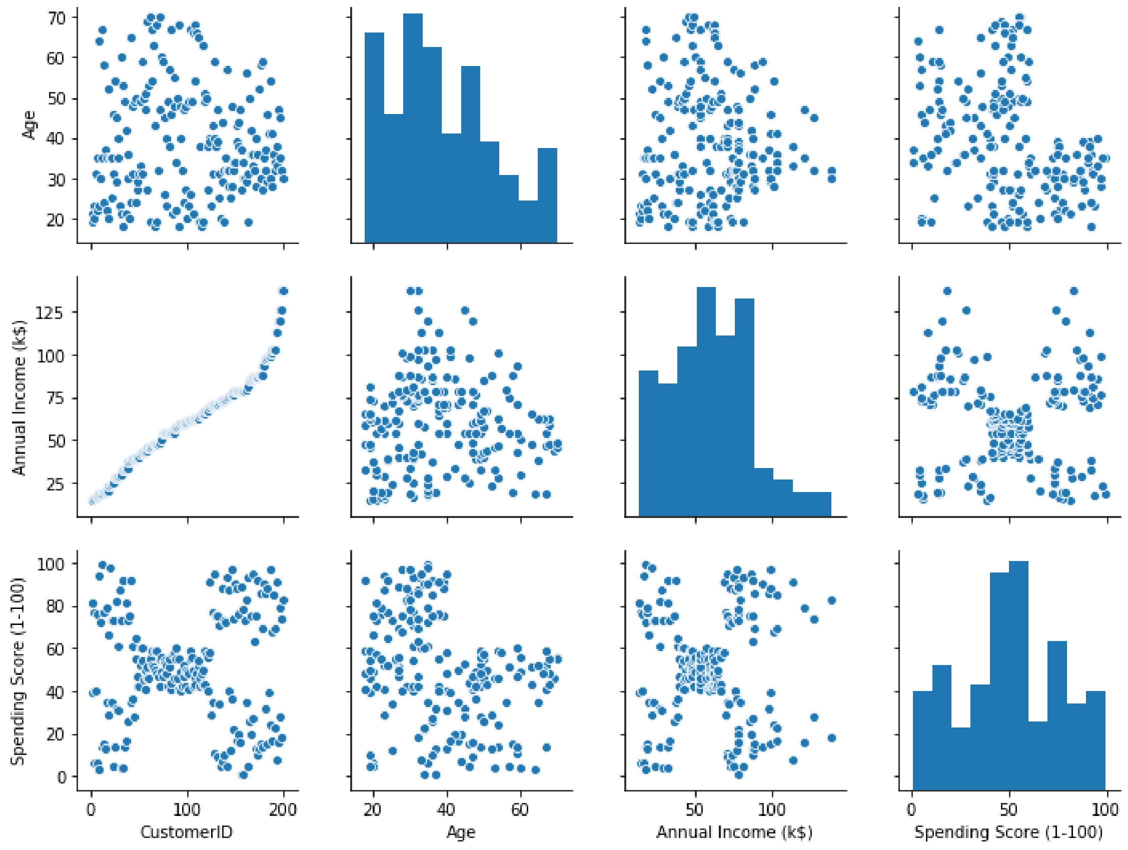
In [7]:

```
sns.pairplot(data)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x2ab4e97b940>
```



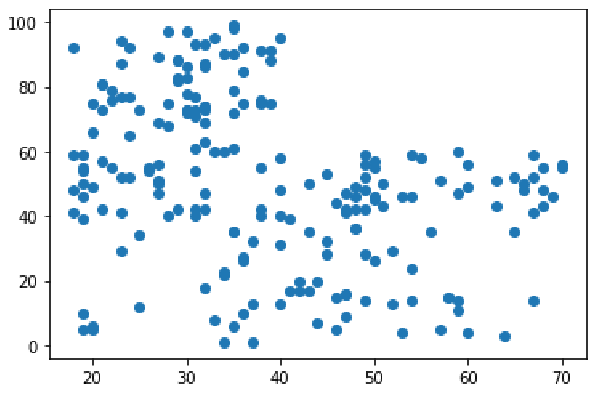


In [8]:

```
plt.scatter(data['Age'],data['Spending Score (1-100)'])
```

Out[8]:

```
<matplotlib.collections.PathCollection at 0x2ab4f47f400>
```



In [9]:

```
features = data.iloc[:,[2,4]].values
```

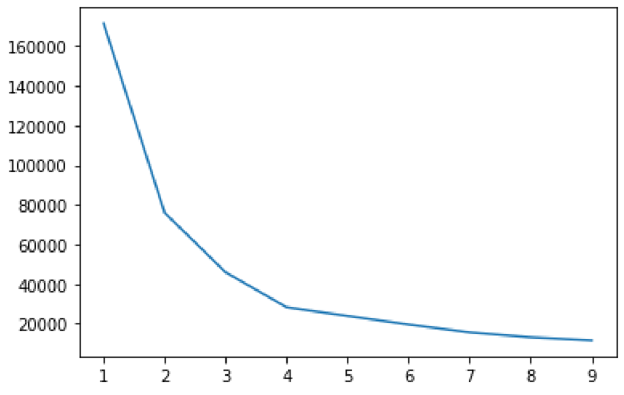
In [14]:

```
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,10):
    model = KMeans(n_clusters=i)
    model.fit(features)
    wcss.append(model.inertia_)

plt.plot(range(1,10),wcss)
```

Out[14]:

```
[<matplotlib.lines.Line2D at 0x2ab50cb5550>]
```



In [15]:

```
from sklearn.cluster import KMeans
finalModel = KMeans(n_clusters=4)
finalModel.fit(features)
```

Out[15]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [18]:

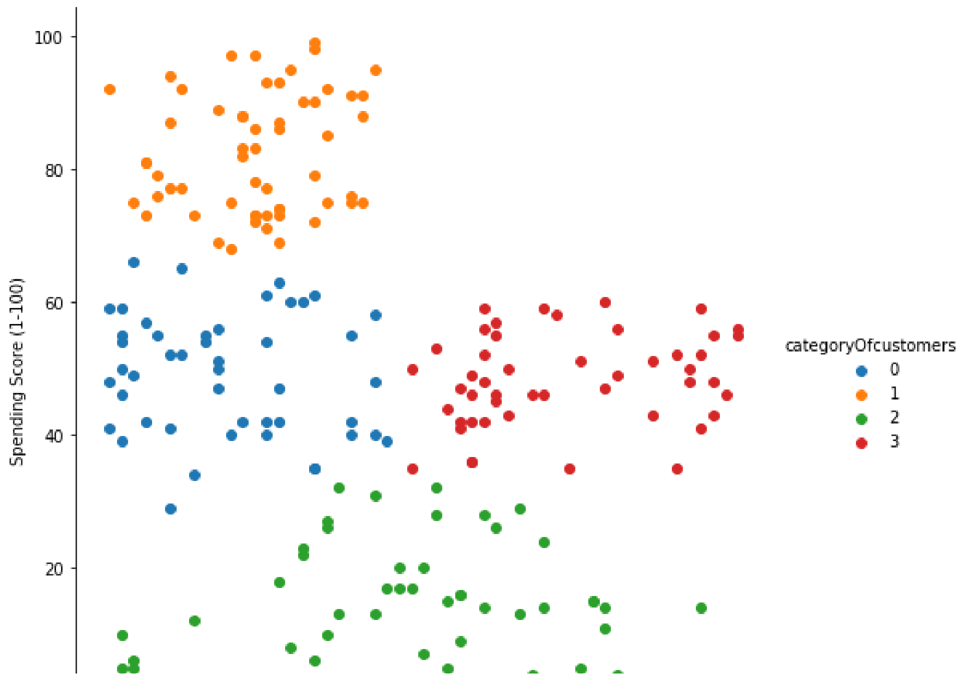
```
categoryOfcustomers=finalModel.predict(features)
```

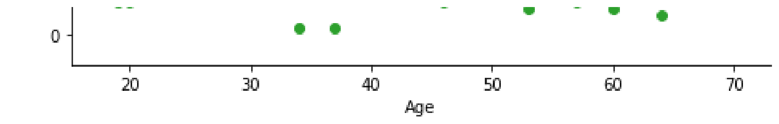
In [19]:

```
finalData  = data.iloc[:,[1,2,3,4]]
finalData['categoryOfcustomers'] = categoryOfcustomers
finalData.head()
```

In [21]:

```
sns.FacetGrid(finalData, hue='categoryOfcustomers', size = 7) \
    .map(plt.scatter, 'Age', 'Spending Score (1-100)' ) \
    .add_legend()
plt.show()
```





In [22]:

```
from sklearn.cluster import KMeans
finalModel1 = KMeans(n_clusters=3)
finalModel1.fit(features)
categoryOfcustomers1=finalModel1.predict(features)
finalData1  = data.iloc[:,[1,2,3,4]]
finalData1['categoryOfcustomers1'] = categoryOfcustomers1
finalData1.head()
```

In [23]:

```
sns.FacetGrid(finalData1, hue='categoryOfcustomers1', size = 7) \
    .map(plt.scatter, 'Age', 'Spending Score (1-100)' ) \
    .add_legend()
plt.show()
```

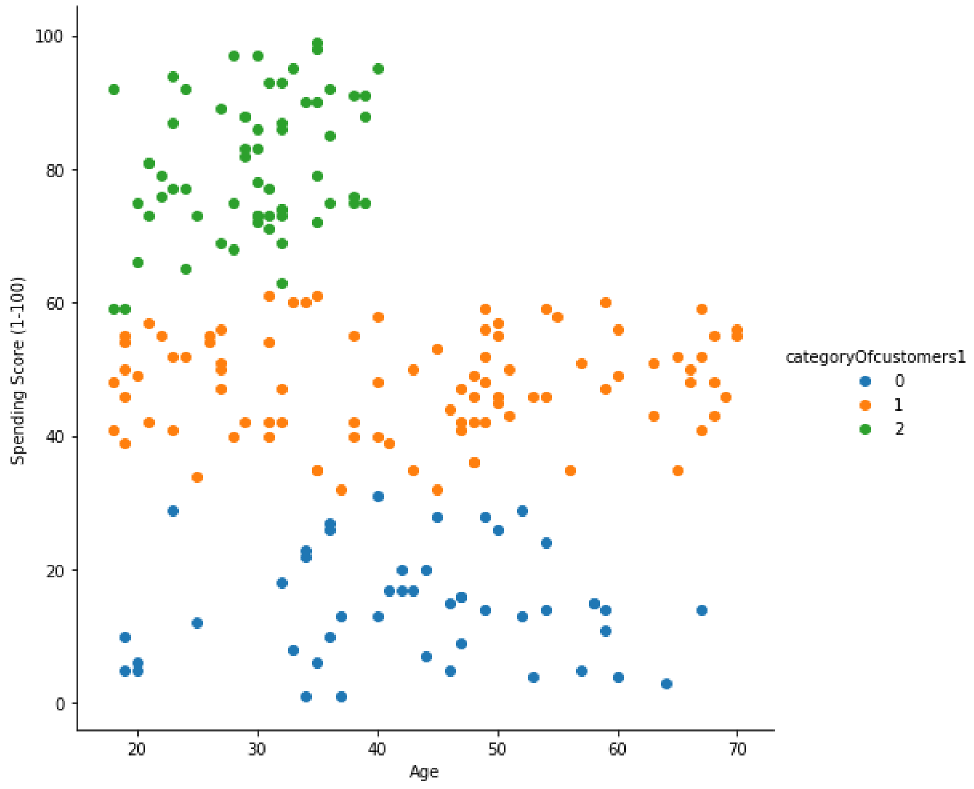
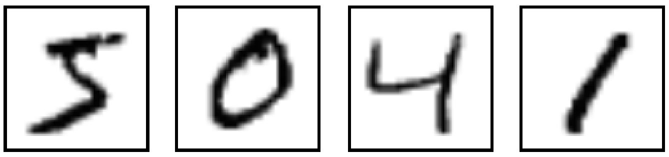


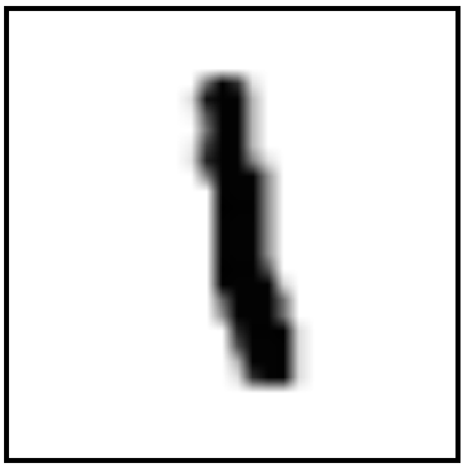
Image Clustering

MNIST

MNIST is a simple computer vision dataset. It consists of 28x28 pixel images of handwritten digits, such as:



Every MNIST data point, every image, can be thought of as an array of numbers describing how dark each pixel is. For example, we might think of as something like:



$$\cong \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

Since each image has 28 by 28 pixels, we get a 28x28 array. We can flatten each array into a 28∗28=784 dimensional vector. Each component of the vector is a value between zero and one describing the intensity of the pixel. Thus, we generally think of MNIST as being a collection of 784-dimensional vectors.

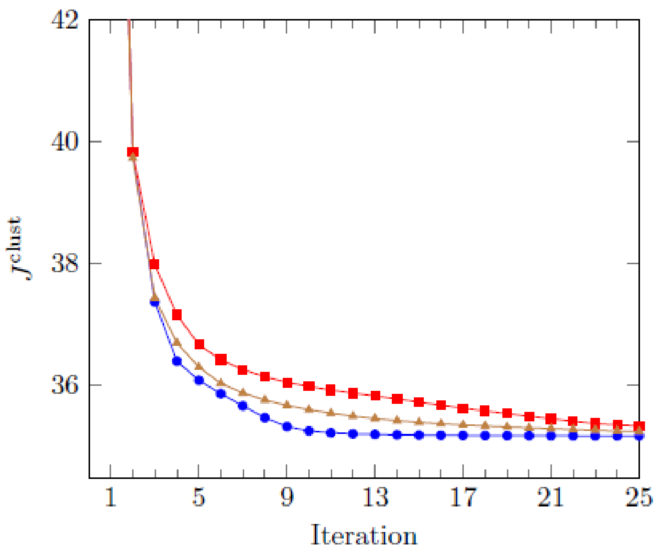
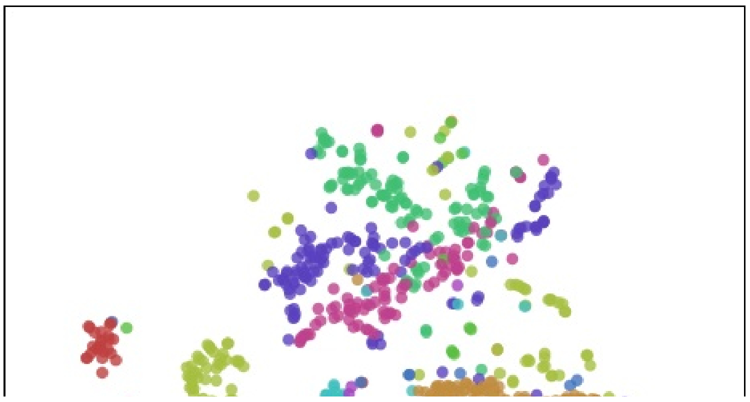
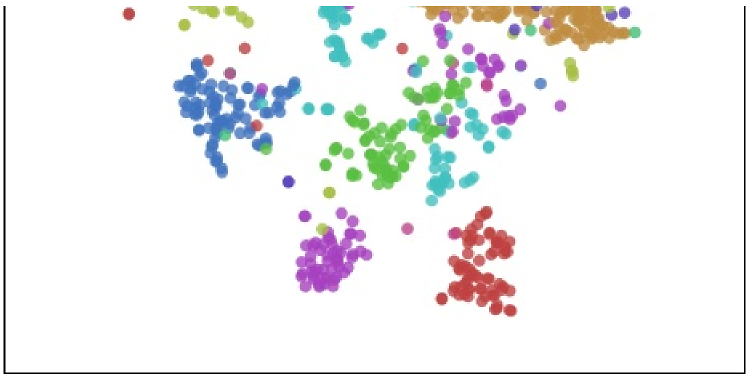


Figure 4.7 Clustering objective J^{clust} after each iteration of the k-means algorithm, for three initial partitions, on digits of the MNIST set.




Visualizing MNIST with t-SNE