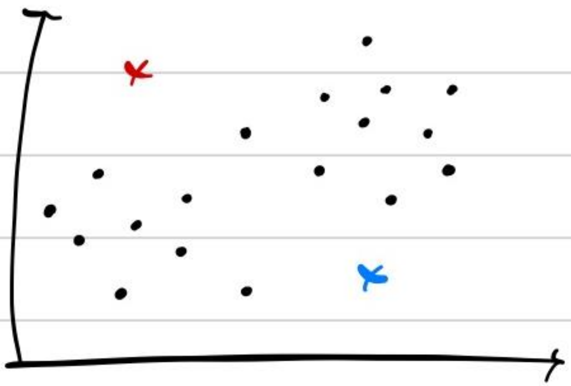
Unsupervised Learning : K – Mean Clustering

2017010698
수학과 오서영

• K - means algorithm



x, x : clustering centroids

① Clustering Assignment Step

: It's going to assign each of the data points one of the two cluster cluster centroids (depending on whether it is closer to them)

red or blue

② Move Centroid Step

: We are going to move two centroids to the average of the points colored the same colour. blue centroid $\xrightarrow{\text{move}}$ blue dot's mean

③ Go to back ① (depending on whether it's closer red or blue)

: If cluster centroids do not change and the colours of the points will not change then at this point k-means has converged

- K - means Algorithm

→ Input
- K (number of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$

$x^{(i)} \in \mathbb{R}^n$ (Drop $x_0 = 1$ convention)

→ Randomly initialize k cluster centroids

① For i=1 to m

$c^{(i)}$ = index (from 1 to k) of cluster centroid closest to $x^{(i)}$ → $\min \| x^{(i)} - \mu_k \|^2$ ... (*)

② For k=1 to K

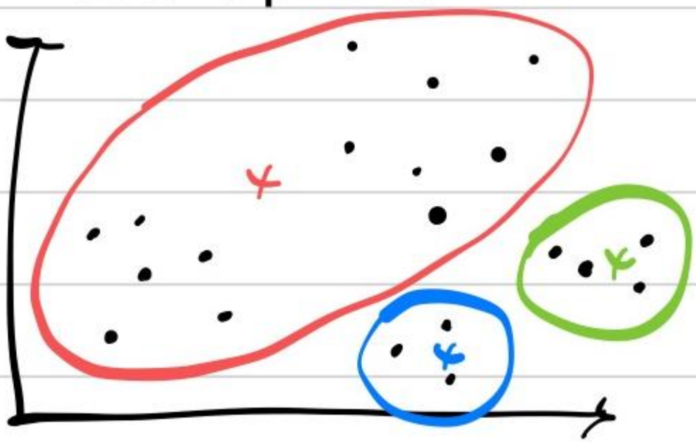
μ_k = average (mean) of points assigned to cluster k

(*) : minimize $J = \frac{1}{m} \sum_{i=1}^{m} \| x^{(i)} - \mu_{c^{(i)}} \|^2$ with $c^{(i)}$ (hold $\mu_1, \cdots, \mu_k$ fixed)

ex) $x^{(1)}, x^{(5)}, x^{(6)} \xrightarrow[\text{assigned}]{} 2 \Rightarrow c^{(1)} = 2, c^{(5)} = 2, c^{(6)} = 2$

$\mu_2 = \frac{1}{3}(x^{(1)}, x^{(5)}, x^{(6)}) \in \mathbb{R}^n$

- Local Optima



$\Rightarrow$ bad!

Initialize k-means lots of times and run k-means lots of times, and try to make sure get as good a solution.

- Random initialization

For i=1 to 100 { 50 ~ 1000

Randomly initialize k-means

Run k-means. Get $c^{(1)}, \cdots, c^{(m)}, \mu_1, \cdots, \mu_k$

Compute cost function (distortion)

$J(c^{(1)}, \cdots, c^{(m)}, \mu_1, \cdots, \mu_k)$

Pick clustering that gave lowest cost J $k = 2 \sim 10$

K-mean Clustering with Patterns based on turing model representing chemical morphogens

1. import packages

```
from keras.datasets import mnist
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import random
```

2. Make Dataset

```
# Make dataset
x_orig = []
y_orig = np.zeros((1,120))
for i in range(0, 36):
    for j in range(i*120 + 1, i*120 + 121) :
        img = Image.open('dataset/{0}/pattern_{1}.jpg'.format(i, j))
        data = np.array(img)
        x_orig.append(data)

for i in range(1,36):
    y_orig = np.append(y_orig, np.full((1, 120),i), axis = 1)
```

```
# Random shuffle
x_orig = np.array(x_orig)
s = np.arange(x_orig.shape[0])
np.random.shuffle(s)

x_shuffle = x_orig[s,:]
y_shuffle = y_orig[:,s]

print(x_shuffle.shape)
print(y_shuffle.shape)
```

```
(4320, 64, 64)
(1, 4320)
```

```
# Split train and test datasets
x_train_orig, x_test_orig, y_train, y_test = train_test_split(x_shuffle,y_shuffle.T,
                                                              test_size=0.3,  shuffle=True, random_state=500)
```

```
# Flatten the training and test images
x_train_flatten = x_train_orig.reshape(x_train_orig.shape[0], -1
x_test_flatten = x_test_orig.reshape(x_test_orig.shape[0], -1)

# Normalize image vectors
x_train = (2/255) * x_train_flatten - 1
x_test = (2/255) * x_test_flatten - 1

# # Normalize image vectors
# x_train = (1/255) * x_train_flatten
# x_test = (1/255) * x_test_flatten

# Explore dataset
print ("number of training examples = " + str(x_train.shape[1]))
print ("number of test examples = " + str(x_test.shape[1]))
print ("x_train shape: " + str(x_train.shape))
print ("y_train shape: " + str(y_train.shape))
print ("x_test shape: " + str(x_test.shape))
print ("y_test shape: " + str(y_test.shape))
```

```
number of training examples = 4096
number of test examples = 4096
x_train shape: (3024, 4096)
y_train shape: (3024, 1)
x_test shape: (1296, 4096)
y_test shape: (1296, 1)
```

3. K-Means

```
model = KMeans(init="k-means++", n_clusters=6, random_state=500)
model.fit(x_train)
y_pred = model.labels_
```

4. Conclusion

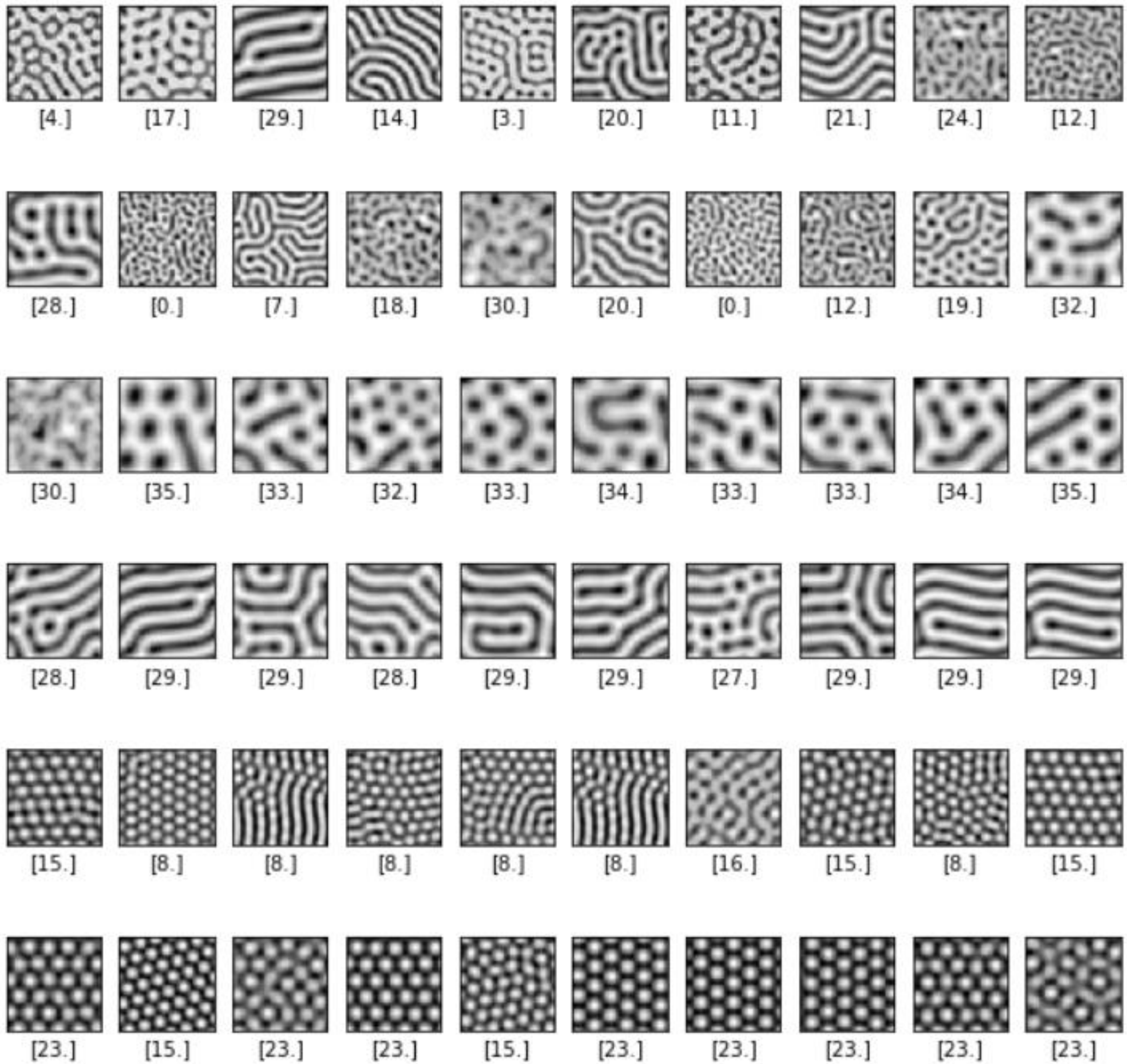
```
np.where(y_pred == 0)
```

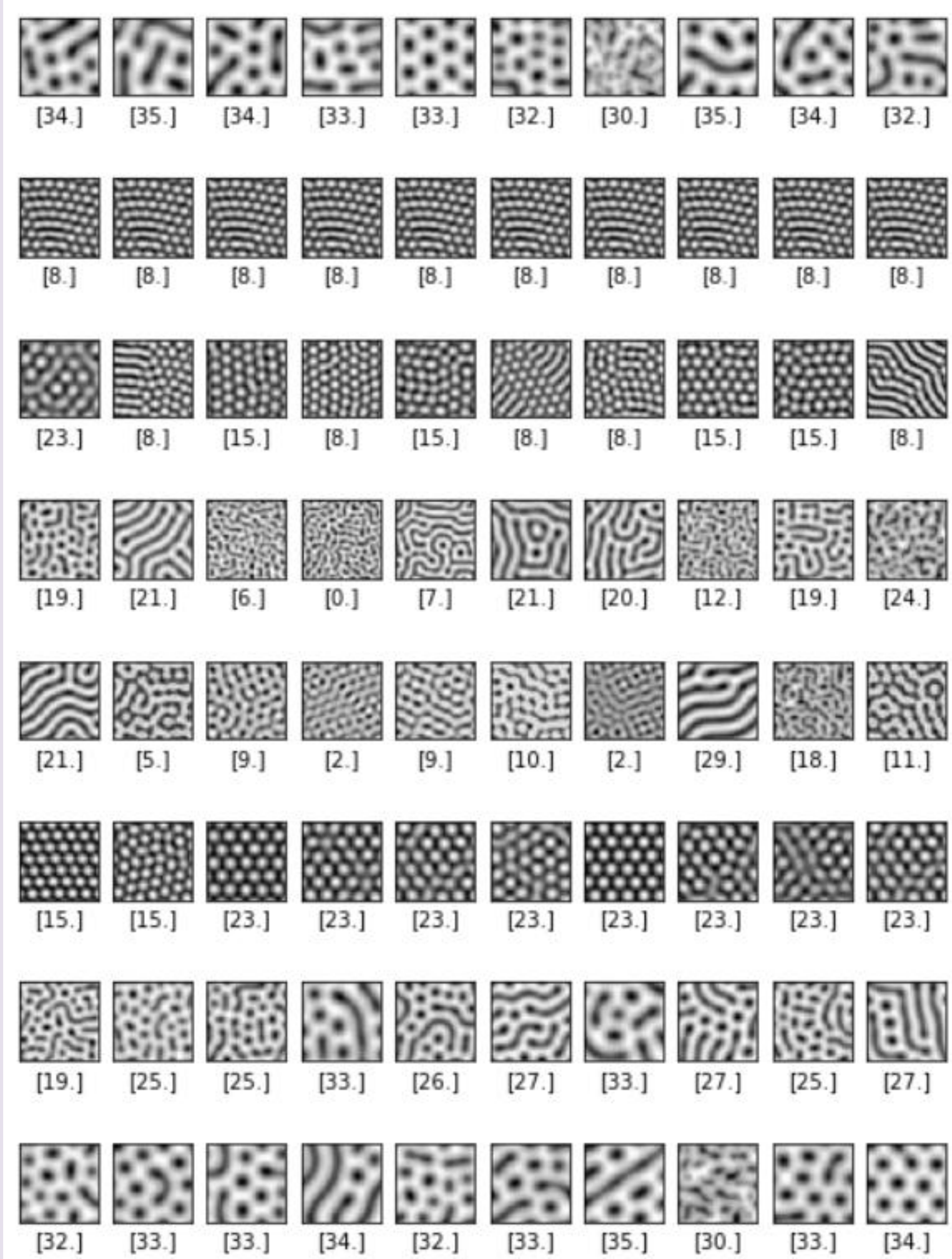
```
(array([   0,    2,    4, ..., 3015, 3016, 3023], dtype=int64),)
```

```
np.random.choice(np.where(y_pred == 0)[0].tolist(), 10, replace=False)
```

```
array([ 903, 2670,  480, 2843,  879, 2035,  417,  267, 1609, 1985])
```

```
plt.figure(figsize=(10,10))
n = 10
row = 1
for cluster in range(6):
    result = np.where(y_pred == cluster)
    for i in range(10):
        rand_index = np.random.choice(result[0].tolist(), n, replace=False)
        plt.subplot(6, 10, 10*cluster+i+1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        plt.imshow(x_train[rand_index[i]].reshape(64,64), cmap='Greys', interpolation='nearest')
        plt.xlabel(y_train[rand_index[i]])
plt.show()
```


[4.] [17.] [29.] [14.] [3.] [20.] [11.] [21.] [24.] [12.]
[28.] [0.] [7.] [18.] [30.] [20.] [0.] [12.] [19.] [32.]
[30.] [35.] [33.] [32.] [33.] [34.] [33.] [33.] [34.] [35.]
[28.] [29.] [29.] [28.] [29.] [29.] [27.] [29.] [29.] [29.]
[15.] [8.] [8.] [8.] [8.] [8.] [16.] [15.] [8.] [15.]
[23.] [15.] [23.] [23.] [15.] [23.] [23.] [23.] [23.] [23.]


[34.] [35.] [34.] [33.] [33.] [32.] [30.] [35.] [34.] [32.]
[8.] [8.] [8.] [8.] [8.] [8.] [8.] [8.] [8.] [8.]
[23.] [8.] [15.] [8.] [15.] [8.] [8.] [15.] [15.] [8.]
[19.] [21.] [6.] [0.] [7.] [21.] [20.] [12.] [19.] [24.]
[21.] [5.] [9.] [2.] [9.] [10.] [2.] [29.] [18.] [11.]
[15.] [15.] [23.] [23.] [23.] [23.] [23.] [23.] [23.] [23.]
[19.] [25.] [25.] [33.] [26.] [27.] [33.] [27.] [25.] [27.]
[32.] [33.] [33.] [34.] [32.] [33.] [35.] [30.] [33.] [34.]