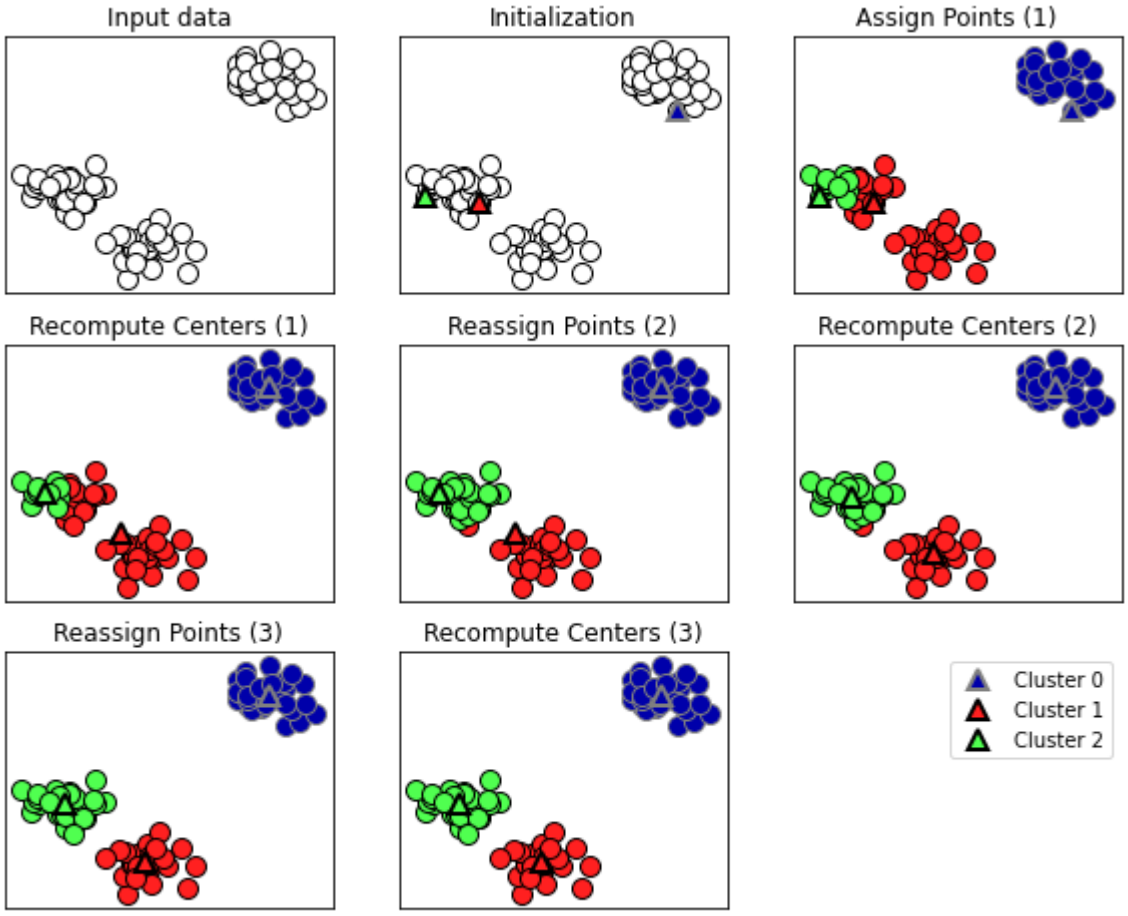
Практическая работа 4. Задача кластеризации и метод kMeans

Задача кластеризации - это задача разделения набора данных на группы, называемые кластерами, так чтобы точки в одном кластере были очень похожи, а точки в разных кластерах были разными.

Метод k средних

Метод кластеризации k-средних - один из простейших и наиболее часто используемых алгоритмов кластеризации. Он пытается найти центры кластеров, для чего последовательно выполняются два действия: присвоение каждой точки данных ближайшему центру кластера, а затем установка каждого центра кластера как среднее значение точек данных, которые ему назначены. Алгоритм завершен, когда назначение экземпляров кластерам больше не меняется. Следующий пример иллюстрирует работу алгоритма на синтетическом наборе данных:

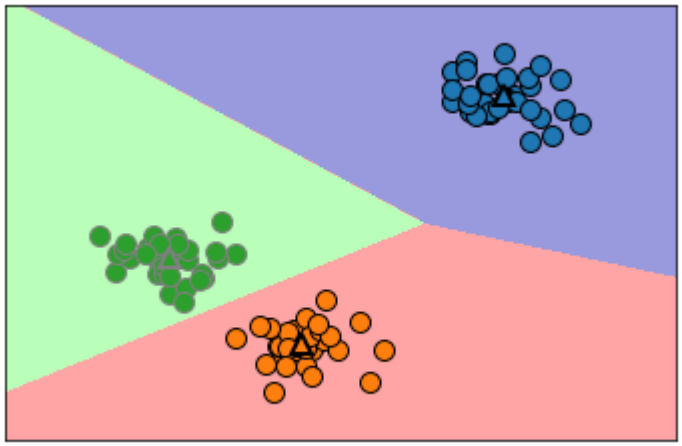
In [13]:
```
import mglearn
mglearn.plots.plot_kmeans_algorithm()
```



В примере центры кластеров показаны треугольниками, а точки данных кружками. Цвета указывают на принадлежность к тому или иному кластеру. Мы ищем три кластера. В начале три точки данных случайным образом взяты в качестве центров кластеров. Затем запускается итерационный алгоритм. Сначала каждая точка данных назначается ближайшему к ней центру кластера. Затем центры кластеров обновляются до среднего значения назначенных точек. Затем процесс повторяется еще два раза. После третьей итерации присвоение точек центрам кластеров не изменилось, поэтому алгоритм останавливается.

Новые точки данных метод k-means отнесет к кластеру с ближайшим к ней центром. Ниже показаны границы кластеров:

In [14]:
```
mglearn.plots.plot_kmeans_boundaries()
```



Использование метода k-средних с помощью пакета scikit-learn довольно просто. Здесь мы применяем его к синтетическим данным, которые мы использовали для предыдущих графиков. Мы создаем экземпляр класса KMeans и устанавливаем количество кластеров, которые мы ищем. Затем мы вызываем метод fit с данными:

In [15]:
```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans

# generate synthetic two-dimensional data
X, y = make_blobs(random_state=1)

# build the clustering model
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
```

Out[15]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

Во время алгоритма каждой точке обучающих данных в X присваивается метка кластера. Эти метки можно найти в атрибуте **kmeans.labels_**:

In [16]:
```
print("Cluster memberships:\n{}".format(kmeans.labels_))
```

```
Cluster memberships:
[0 1 1 1 2 2 2 1 0 0 1 1 2 0 2 2 2 0 1 1 2 1 2 0 1 2 2 0 0 2 0 0 2 0 1 2 1
 1 1 2 2 1 0 1 1 2 0 0 0 0 1 2 2 2 0 2 1 1 0 0 1 2 2 1 1 2 0 2 0 1 1 1 2 0
 0 1 2 2 0 1 0 1 1 2 0 0 0 0 1 0 2 0 0 1 1 2 2 0 2 0]
```

Поскольку мы указали количество кластеров, равное трем, кластеры пронумерованы от 0 до 2.

Для определение кластера новых точек, можно использовать метод **predict**.

In [17]:
```
print(kmeans.predict([[-1, -4], [-1, -2]]))
```

```
[2 0]
```

Вы можете видеть, что кластеризация похожа на классификацию, поскольку каждый элемент получает метку кластера (класса). Повторный запуск алгоритма k средних может привести к другой нумерации кластеров из-за случайного характера инициализации. Мы также можем попробовать указать другое число кластеров.

In [18]:
```
from matplotlib import pyplot as plt
fig, axes = plt.subplots(1, 2, figsize=(10, 5))
```

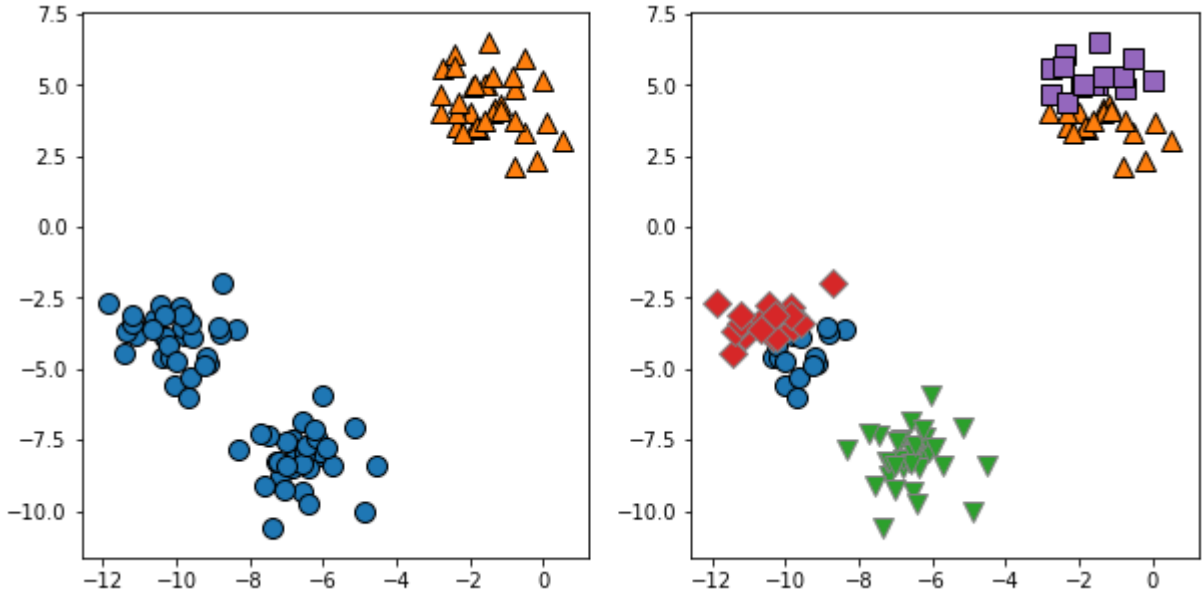
```
# using two cluster centers:
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
assignments = kmeans.labels_

mglearn.discrete_scatter(X[:, 0], X[:, 1], assignments, ax=axes[0])

# using five cluster centers:
kmeans = KMeans(n_clusters=5)
kmeans.fit(X)
assignments = kmeans.labels_

mglearn.discrete_scatter(X[:, 0], X[:, 1], assignments, ax=axes[1])
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x7fd73883d650>,
 <matplotlib.lines.Line2D at 0x7fd73883dc10>,
 <matplotlib.lines.Line2D at 0x7fd738846190>,
 <matplotlib.lines.Line2D at 0x7fd7388466d0>,
 <matplotlib.lines.Line2D at 0x7fd738846c90>]
```



Можно видеть, как метод k-средних, нашел два и пять кластеров в наших данных.

Пример:

Применим метод k-средних для определения цифр одного класса.

Набор данных состоит из 1797 образцов с 64 характеристиками, каждая из которых представляет собой яркость одного пикселя в изображении 8 × 8:

In [19]:
```
from sklearn.datasets import load_digits
digits = load_digits()
digits.data.shape
```

Out[19]:
```
(1797, 64)
```

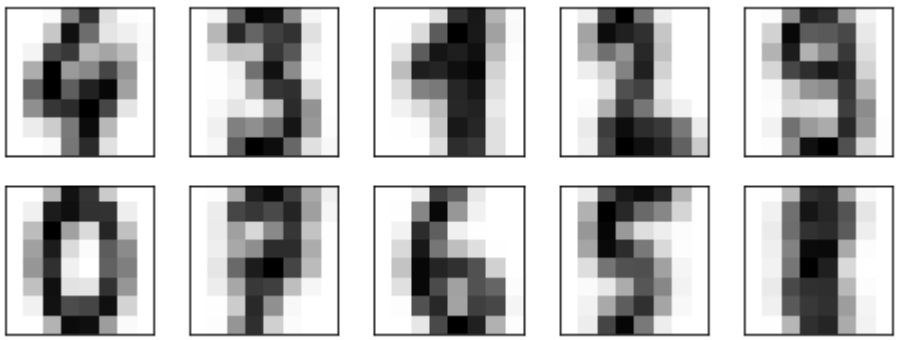
Выполним кластеризацию, указав 10 классов:

In [20]:
```
kmeans = KMeans(n_clusters=10, random_state=0)
clusters = kmeans.fit_predict(digits.data)
kmeans.cluster_centers_.shape
```

Out[20]:
```
(10, 64)
```

Центры кластеров могут быть интерпретированы как «типичная» цифра внутри кластера. Посмотрим, как выглядят эти кластерные центры:

In [21]:
```
fig, ax = plt.subplots(2, 5, figsize=(8, 3))
centers = kmeans.cluster_centers_.reshape(10, 8, 8)
for axi, center in zip(ax.flat, centers):
    axi.set(xticks=[], yticks=[])
    axi.imshow(center, interpolation='nearest', cmap=plt.cm.binary)
```



Отметим, что метод не использовал меток класса.

Следующий код проверяет, какова доля правильных ответов.

In [22]:
```
from scipy.stats import mode
import numpy as np

labels = np.zeros_like(clusters)
for i in range(10):
    mask = (clusters == i)
    labels[mask] = mode(digits.target[mask])[0]
```

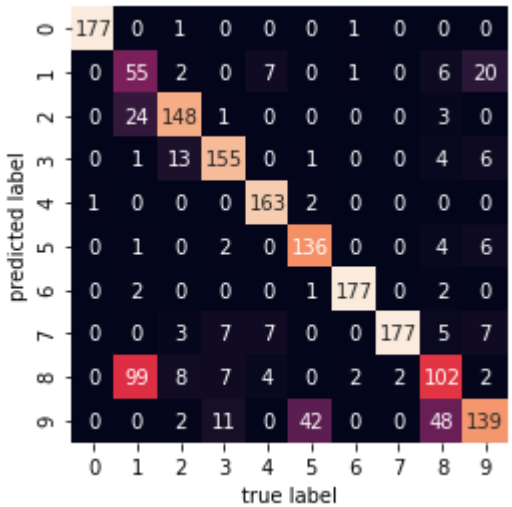
In [23]:
```
from sklearn.metrics import accuracy_score
accuracy_score(digits.target, labels)
```

Out[23]:
```
0.7952142459654981
```

Итак, алгоритм k средних дает 80% правильных ответов! Давайте посмотрим также на матрицу ошибок:

In [24]:
```
import seaborn as sns
from sklearn.metrics import confusion_matrix
mat = confusion_matrix(digits.target, labels)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=digits.target_names,
            yticklabels=digits.target_names)
plt.xlabel('true label')
plt.ylabel('predicted label')
```

Out[24]:
```
Text(91.68, 0.5, 'predicted label')
```



Как мы видим, основная путаница возникает между восьмерками и единицами. Но это по-прежнему показывает, что с помощью простого метода k-средних мы можем построить классификатор цифр без ссылки на какие-либо известные метки!