Toggle code

drawing

Week 10 - Clustering

Dr. David Elliott

1. Introduction
2. K-Means
3. Picking K

1. Introduction

Most of this course has focused on supervised learning methods such as regression and classification.

Here we look at a set of statistical tools intended for the setting in which we have a set of set of p features $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$ measured on n observations, but no response $\mathbf{y}$ also measured on those same n observations. Rather than prediction, the goal is to discover interesting things about the measurements on $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$.

- Is there an informative way to visualize the data?
- **Can we discover subgroups among the variables or among the observations?**

Notes

- _"In the supervised learning setting, we typically have access to a set of p features $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$, measured on n observations, and a response $\mathbf{y}$ also measured on those same n observations. The goal is then to predict $\mathbf{y}$ using $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$."_[1]

Unsupervised learning is often much more challenging than supervised learning.

It tends to be more subjective, as it can be hard to assess the results obtained from unsupervised learning methods, with a less straight forward goal for the analysis.

Unsupervised learning is often performed as part of an exploratory data analysis.

Notes

- Due to being more subjective, we wont be using any "real" data for now while learning about it... we'll leave most of that for the applications notebook.
- *"The reason for this difference is simple. If we fit a predictive model using a supervised learning technique, then it is possible to check our work by seeing how well our model predicts the response y on observations not used in fitting the model. However, in unsupervised learning, there is no way to check our work because we don’t know the true answer—the problem is unsupervised."*[1]

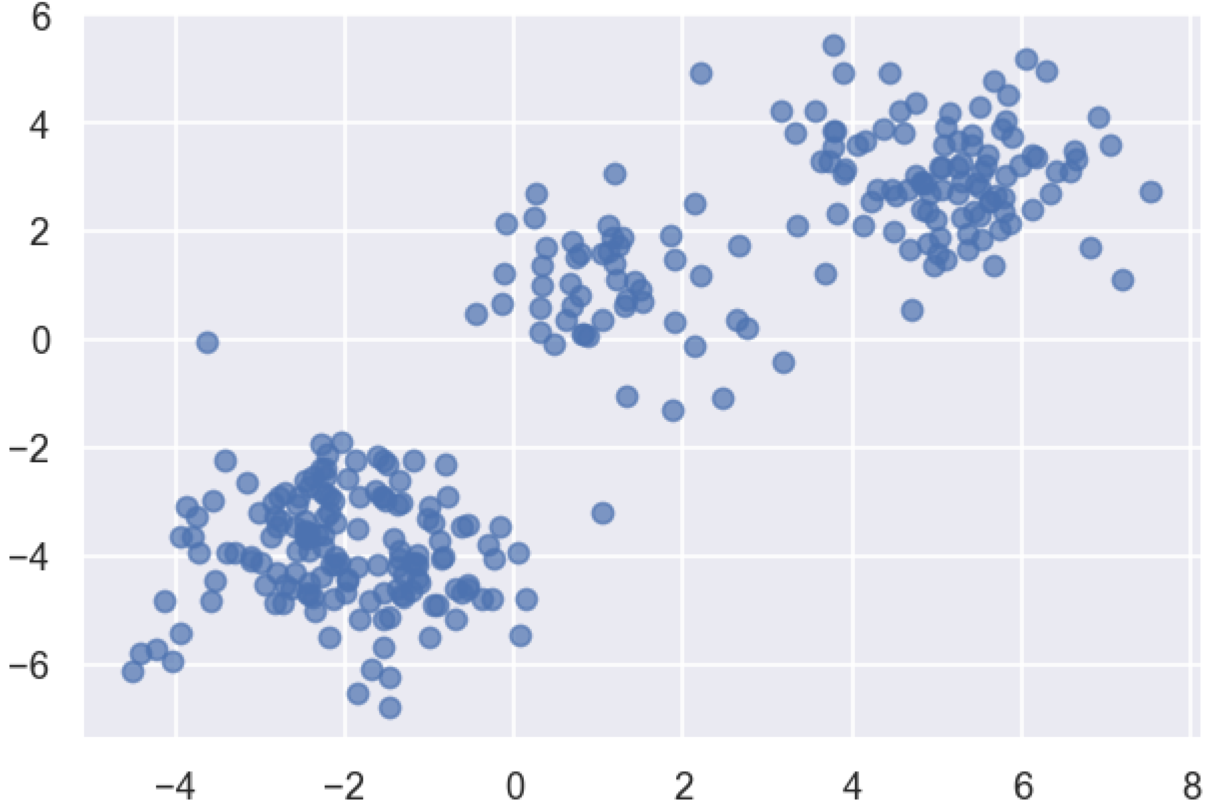
Clustering

Clustering covers a a broad class of methods for discovering unknown subgroups in data.

Clustering obervations of a dataset means seeking to partition them into distinct groups so observations in each group are similar, while observations in different groups are dissimilar.

Humans can identify clusters easily. For example, how many clusters are in the following plot?

Figure 1: Example Clusters



... if you guessed 3 you are correct, because I made the data have three centres with some noise around.

However for an algorithm to cluster data we must define what it means for two or more observations to be similar or different, with this decision typically being a domain-specific consideration[1].

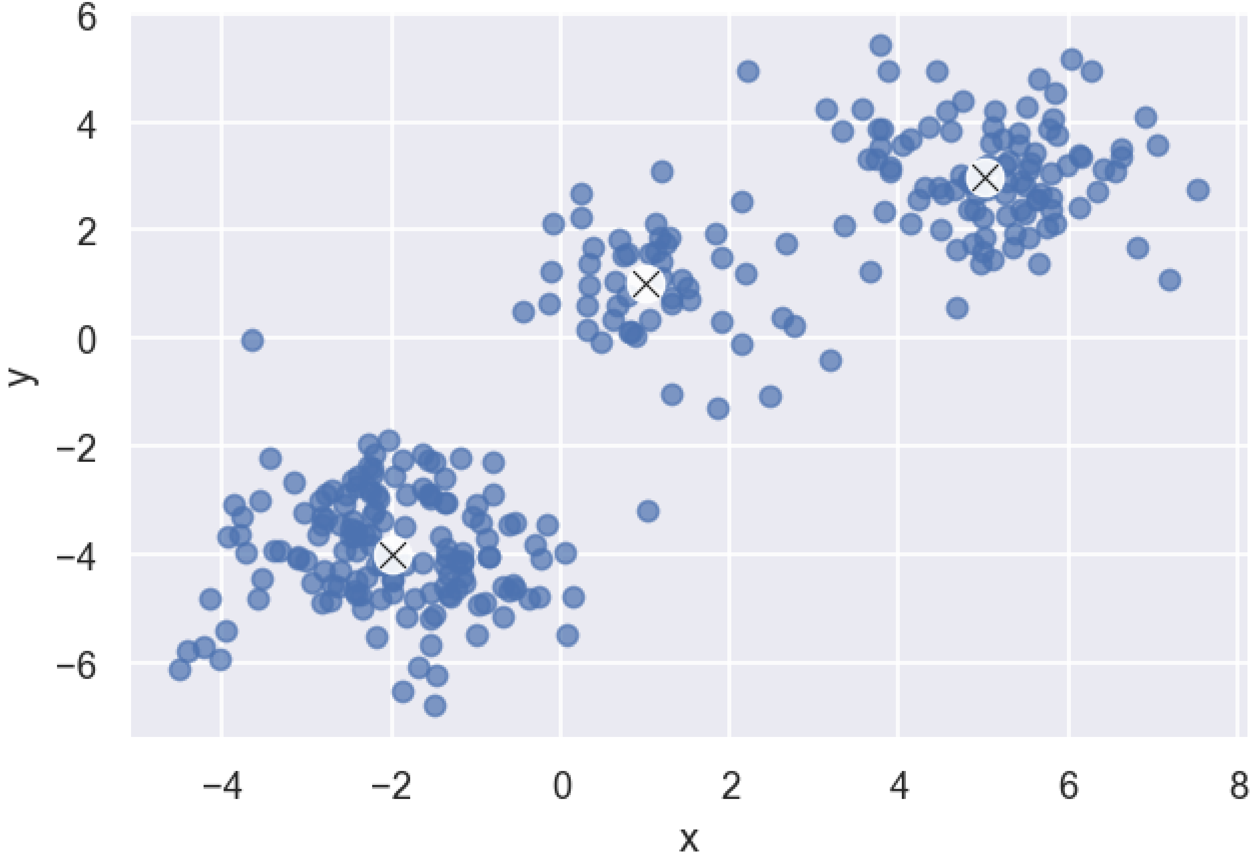


Figure 2: Example Cluster Centres

In the real world we dont have the ground-truth of what the categories are, so our goal is to group them based on feature similarities.

In this series of lectures we discuss three of the best-known clustering approaches: K-means clustering, hierarchical clustering, and density-based clustering[1]:

- In **K-means clustering**, we seek to partition the observations into a pre-specified number of clusters.
- In **Hierarchical clustering**, we use a tree-like visual representation of the observations (*dendrogram*) to view the clusterings obtained for each possible number of clusters.

- In **density-based clustering**, we can identify arbitary shapes by looking at the density of observations.

Notes

- Assigning data to clusters would be easy if we had the true...
  - ...center of the cluster (centeroids), as we would just assign each point to its closest centeroid.
  - ...labels, as we would just compute the mean of each label to find the centroids.
- There is a fourth advanced type of clustering we do not address, **graph-based clustering**.
  - An example of this is **Spectral Clustering**, which use eigenvectors, of a similarity or distance matrix to derrive the cluster relationships (see Von Luxburg, U. (2007). A tutorial on spectral clustering. Statistics and computing, 17(4), 395-416)[2].

2. K-Means

K-means clustering is a simple approach for partitioning a dataset into K distinct, non-overlapping clusters.

We can define our clusters as sets $C_1, \dots, C_K$ containing the indices of the observations in each cluster. These sets satisfy two properties[1]:

1. $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$.
2. $C_k \cap C_{k'} = \varnothing \text{ for all } k \neq k'$

If the ith observation is in the kth cluster, then $i \in C_k$.

Notes

1. Each observation belongs to at least one of the K clusters.
2. The clusters are nonoverlapping: no observation belongs to more than one cluster.

To perform K-means clustering, we must first standardized the input features, as without may lead to poor performance[5].

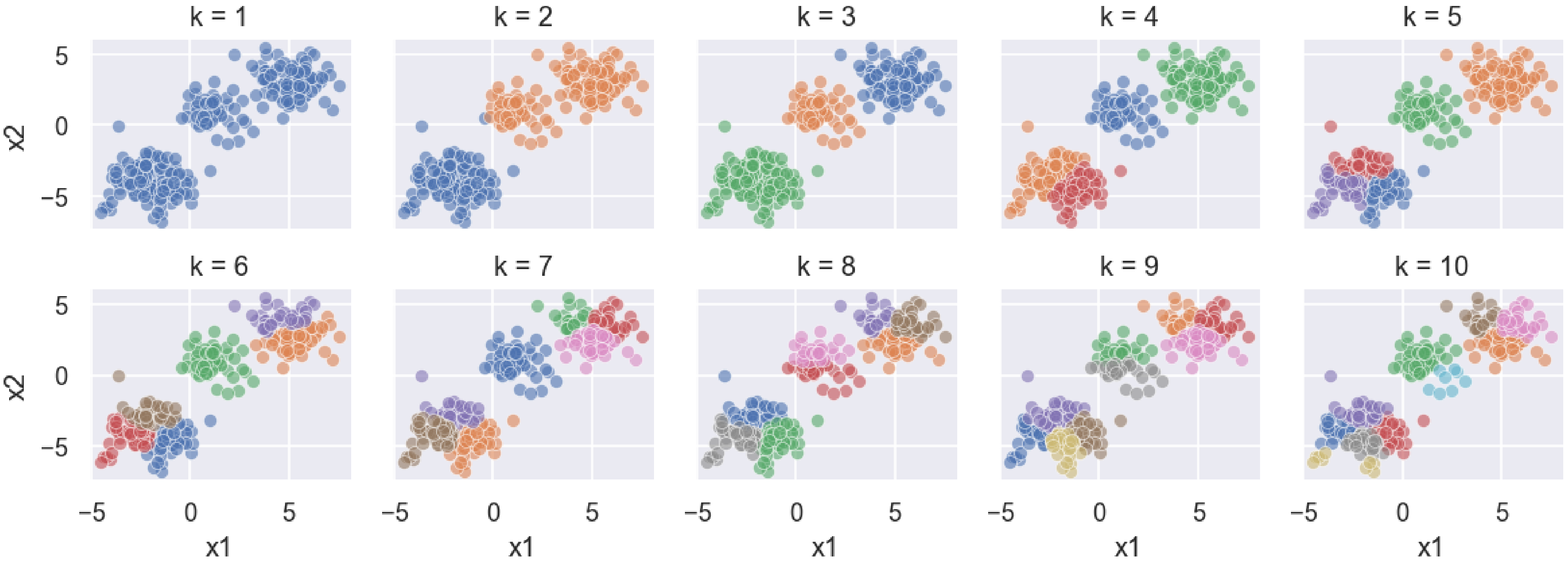
We must also specify the desired number of clusters K.

The algorithm will assign each observation to exactly one of the K clusters.

Notes

- standardized here meaning centered to have mean zero and scaled to have standard deviation one[1].

Figure 3: K-means clustering on a simulated two-dimensional dataset



How Does K-Means Work?

The traditional method can be summerised by the following steps[2]:

> 1. Randomly pick *k* centeroids from the sample points as initial cluster centers.
> 2. Assign each sample to the nearest centroid $\mu_j, j \in \{1, \ldots, k\}$.
> 3. Move the centroids to the center of the samples that were assigned to it.

> 4. Repeat steps 2 and 3 until the cluster assignments do not change or a user-defined tolerance or maximum number of iterations is reached.

Notes

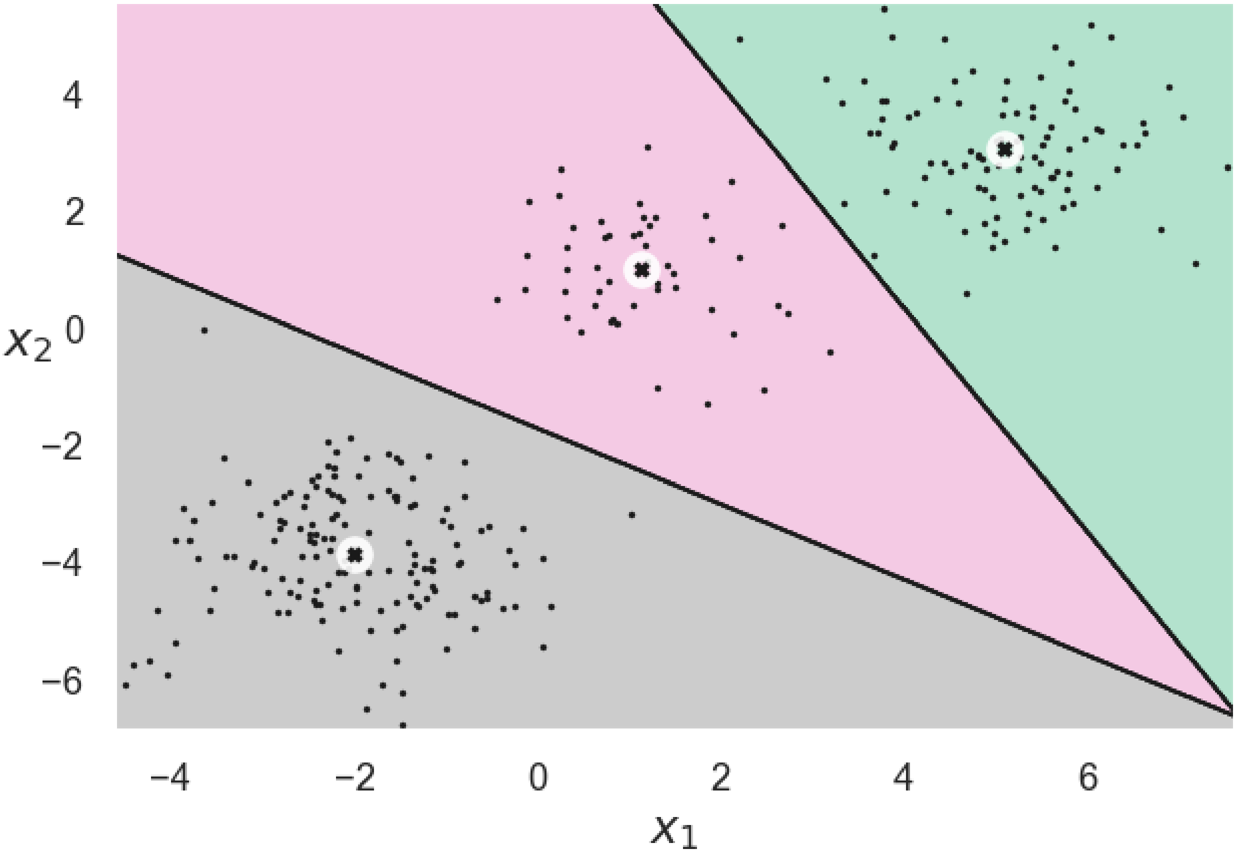
- K-Means is categorised as **Prototype-based Clustering**, meaning each cluster is represented by either a *centroid* (average) with continuous features, or *medoid* (most representative point), for categorical features[2].
- *"In general, we can cluster observations on the basis of the features in order to identify subgroups among the observations, or we can cluster features on the basis of the observations in order to discover subgroups among the features. In what follows, for simplicity we will discuss clustering observations on the basis of the features, though the converse can be performed by simply transposing the data matrix."*[1]

New observations are then assigned to a cluster based on which centeroid is closest.

Note

- A decision boundary for a K-Means algorithms can be called "Voronoi Tessellation"

Figure 4: K-means Decision Boundaries



How do we measure similarity?

Similarity can be defined as the opposite of distance, with distance often being measured using the *squared Euclidean distance*.

The squared Euclidean distance between two points x_{ij} and y_{ij} in p-dimensional space is[2]:

$$d(\mathbf{x}_i, \mathbf{y}_i)^2 = \sum_{j=1}^{p} (x_{ij} - y_{ij})^2 = ||\mathbf{x}_i - \mathbf{y}_i||_2^2,$$

where j refers to the jth dimension (e.g. feature column) of the sample points.

Notes

- Heres an explanation of $||\mathbf{x}_i - \mathbf{y}_i||_2^2$ by a collegue Colin Rundel:

  "Norm notation can be a bit funny - in general the p-norm of a vector is written as

  $$||x||_p = (|x_1|^p + |x_2|^p + \ldots + |x_n|^p)^{1/p}$$

  So the 2-norm of x would be $||x||_2 = (|x_1|^2 + |x_2|^2 + \ldots + |x_n|^2)^{1/2}$ which is just the euclidean distance in an n-dimensional space.

  Finally, the 2nd 2 is just a standard exponent so $||x||_2^2 = (||x||^2)^2 = |x1|^2 + |x2|^2 + \ldots + |x_n|^2$."

What makes a good cluster?

A good clustering is one for which the within-cluster variation is as small as possible.

The within-cluster variation for cluster C_k is a measure $W(C_k)$ of the amount by which the observations within a cluster differ from each other. We want to solve[1]:

$$\underset{C_1,\ldots,C_K}{\text{minimize}}\left\{\sum_{k=1}^{K} W(C_k)\right\}.$$

We can do this with the help of squared Euclidean distance:

$$\begin{align} W(C_k) &= \frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij} - x_{i'j})^2 \tag{1} \\ &= 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij} - \bar{x}_{kj})^2, \qquad \bar{x}_{kj} = \frac{1}{|C_k|}\sum_{i\in C_k} x_{ij} \tag{2} \end{align}$$

NOTES

- In other words, we want to partition the observations into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible.
- $|C_k|$ denotes the number of observations in the kth cluster
- *"In other words, the within-cluster variation for the kth cluster is the sum of all of the pairwise squared Euclidean distances between the observations in the kth cluster, divided by the total number of observations in the kth cluster."*[1]

The k-means algorithm can be described as an iterative approach which minimises the within-cluster *Sum of Squared Errors*[2].

The Sum of Squared Errors (SSE), or cluster inertia, is

$$SSE = \sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik} ||\mathbf{x}_i - \mu_k||_2^2$$

where i is the sample index, μ_k is the centroid (representative point) for the cluster c. $w_{ik} = 1$ if sample $\mathbf{x}_i$ is in the cluster k or otherwise $w_{ik} = 0$.

Notes

- *"inertia is the sum of the squared distances between each training instance and its closest centroid"*[3]

If we only run the algorithm once and randomly pick where the centroids start then performance will be affected by where this starting point is.

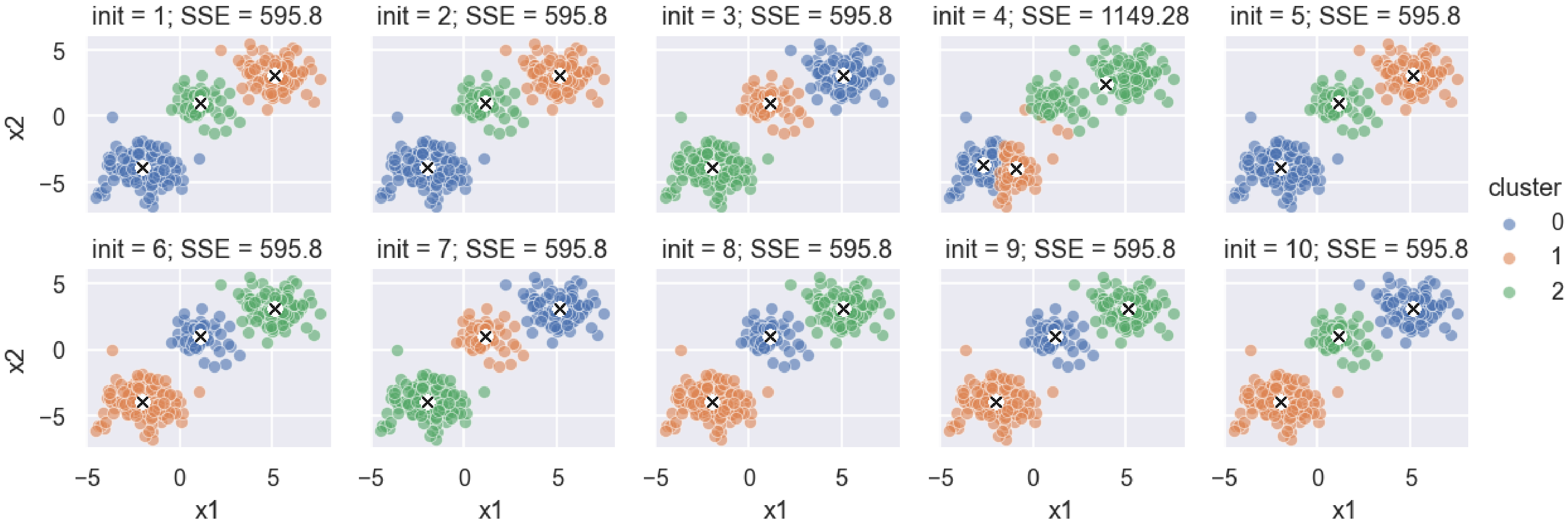
Indeed the algorithm may converge to a *local optimum*.

Notes

- Because the K-means algorithm finds a local rather than a global optimum, the results obtained will depend on the initial (random) cluster assignment of each observation.

There are therefore two improvements commonly used on the baseline method, the first is running the k-means algorithm with different centroid seeds and picking the best one based on inertia.

Figure 5: Repeated K-means clustering



K-Means++

The second, and the default method in Sklearn, is to use K-Means++[4]. Instead of initializing the centroids randomly, it is preferable to initialize them first[5]:

> 1. Take one centroid c_1, chosen uniformly at random from the dataset.
> 2. Take a new center c_i, choosing an instance $\mathbf{x}_i$ with probability: $D(\mathbf{x}_i)^2$ / $\sum_{j=1}^{m} D(\mathbf{x}_j)^2$ where $D(\mathbf{x}_i)$ is the distance between the instance $\mathbf{x}_i$ and the closest centroid that was already chosen. This probability distribution ensures that instances that are further away from already chosen centroids are much more likely be selected as centroids.
> 3. Repeat the previous step until all k centroids have been chosen.

After using this for initializing centroids, we use the K-Means algorithm as before.

As this initialisation means we are less likely to converge at a suboptimal solution, we can reduce `n_init` to compensate for the computational cost incurred by this extra step.

Other Improvements[5]

Accelerated K-Means[6]

Another important improvement was accelerating the runtime of K-Means using triangle inequality.

By keeping track of the lower and upper bounds for distances between instances and centeroids, the number of required distance caclulations could be reduced.

However it’s more memory intensive as it requires the allocation of an extra array[7].

Notes

- triangle inequality is the idea that a straight line is always the shortest distance between two points
- $AC \leq AB + BC$, where A, B, C are points and AB, AC, BC are distances between points.
- this is the default in Scikit-learn, `algorithm="elkan"`

```
Algorithm: full
131 ms ± 2.11 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

Algorithm: elkan
15 ms ± 1.02 ms per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Mini-batch K-means[8]

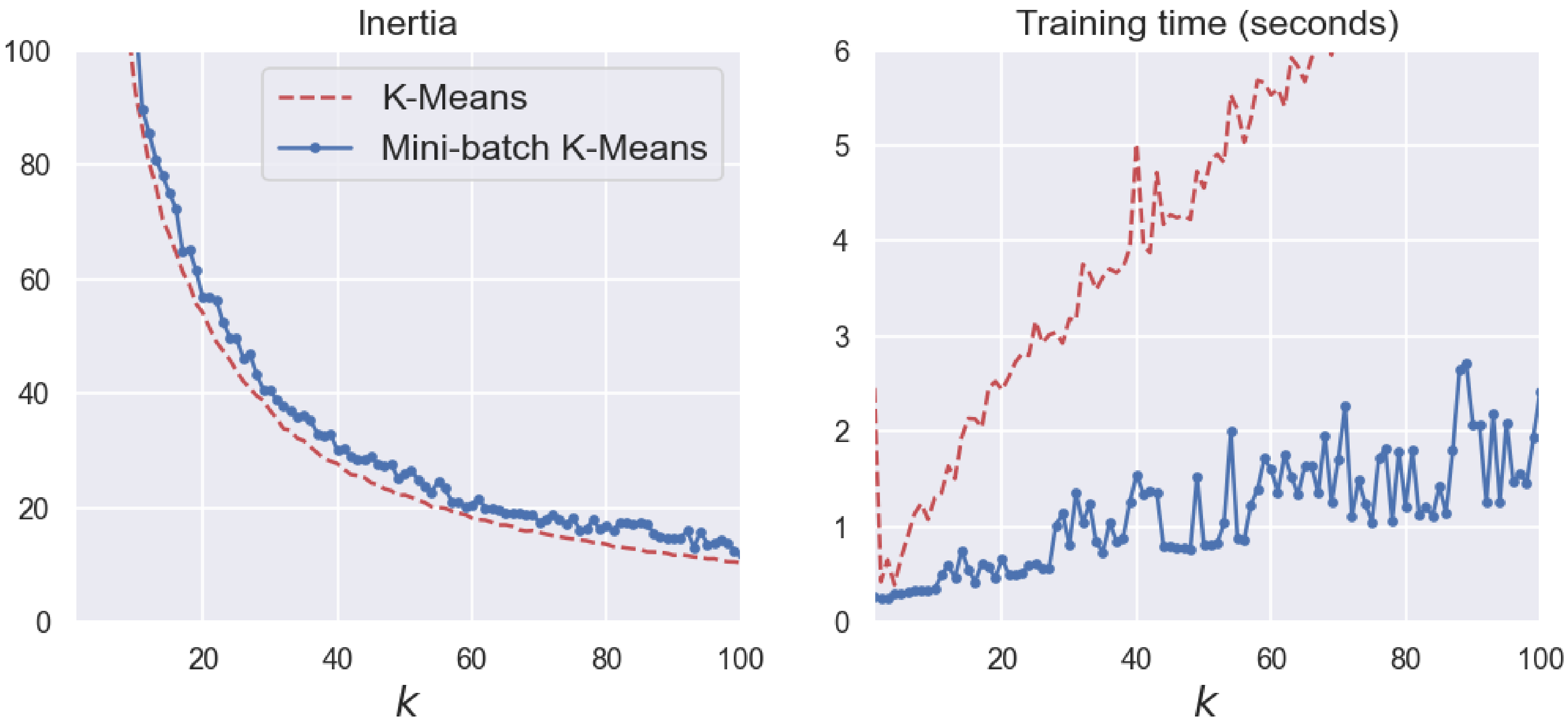
Instead of using the full dataset at each K-means iteration, the algorithm uses mini-batches of data, moving the centeroids slightly at each iteration.

This speeds up the algorithm as well as making it possible to cluster large datasets that do not fit into memory.

However the inertia is generally slightly worse, particularly when the number of clusters increase.

Notes

- For an example using data larger than memory see https://github.com/ageron/handson-ml2/blob/master/09_unsupervised_learning.ipynb.

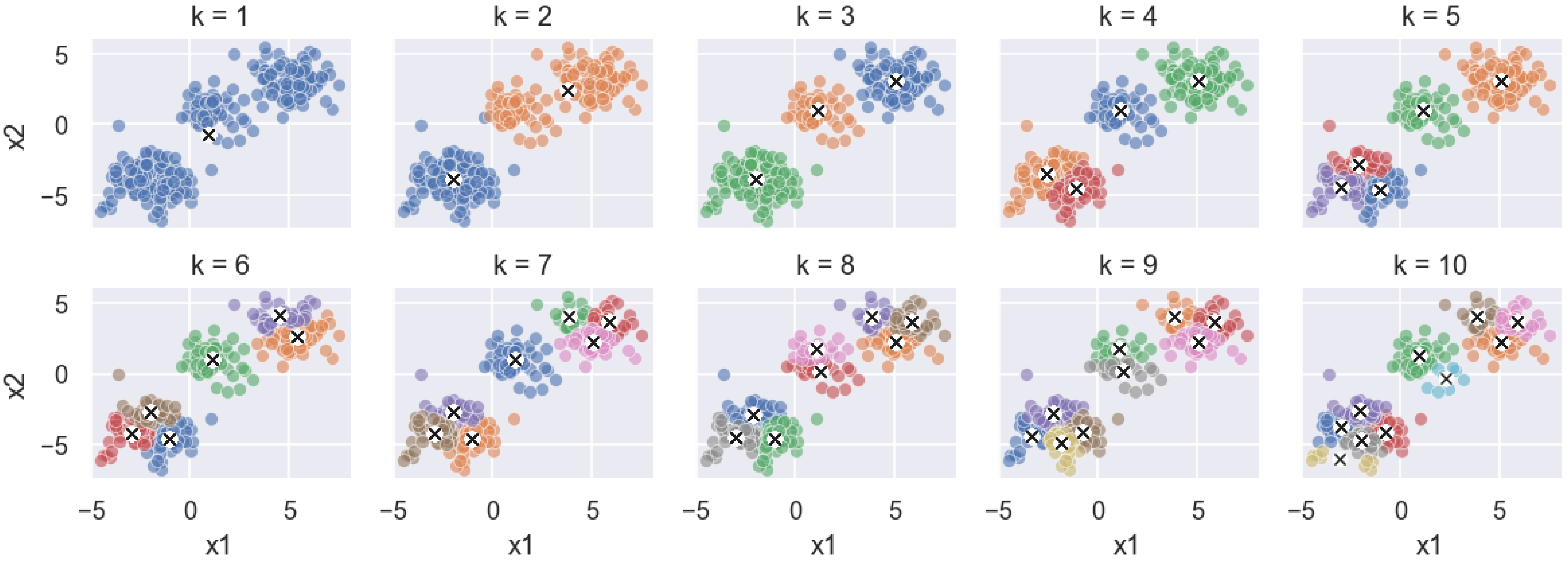


3. Picking K

To perform K-means clustering, we must decide how many clusters we expect in the data. The problem of selecting this K can be tricky[1].

A simple approach is to try several different choices and look for the one with the most useful or interpretable solution.

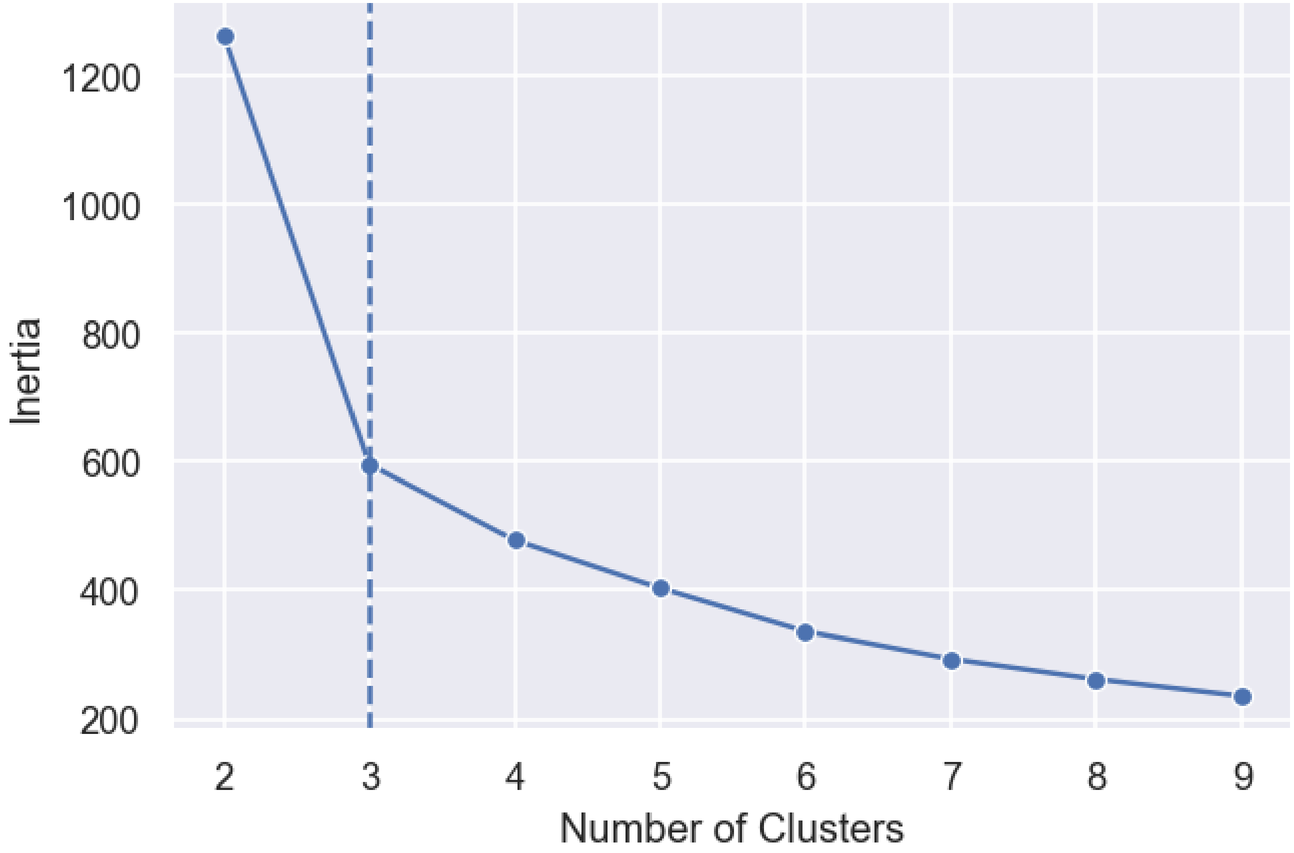
Figure 6: K-means clustering (and centres) on a simulated two-dimensional dataset



Instead we could base this on the within-cluster SSE. However we cannot just pick the lowest inertia as it keeps getting lower as we increase K. This is bcause the more clusters there are, the closer instance will be to its centeroid[5].

Instead we can use a graphical representation of within-cluster SSE and estimate the number of clusters using the *elbow method*[2]. As if K increases, distortion will decrease, we need to find where the distortions decrease less rapidly.

Figure 7: K-means Inertia



Silhouette Score

Using the within-cluster SSE is rather course so we may wish to use a *silhouette score* instead; another graphical tool used to measure how tightly grouped samples are in clusters are[2].

To calculate it we can apply the following steps[2]:

> 1. Calculate the cluster cohesion a_i as the average distance between a sample x_i and all other points in the same cluster.

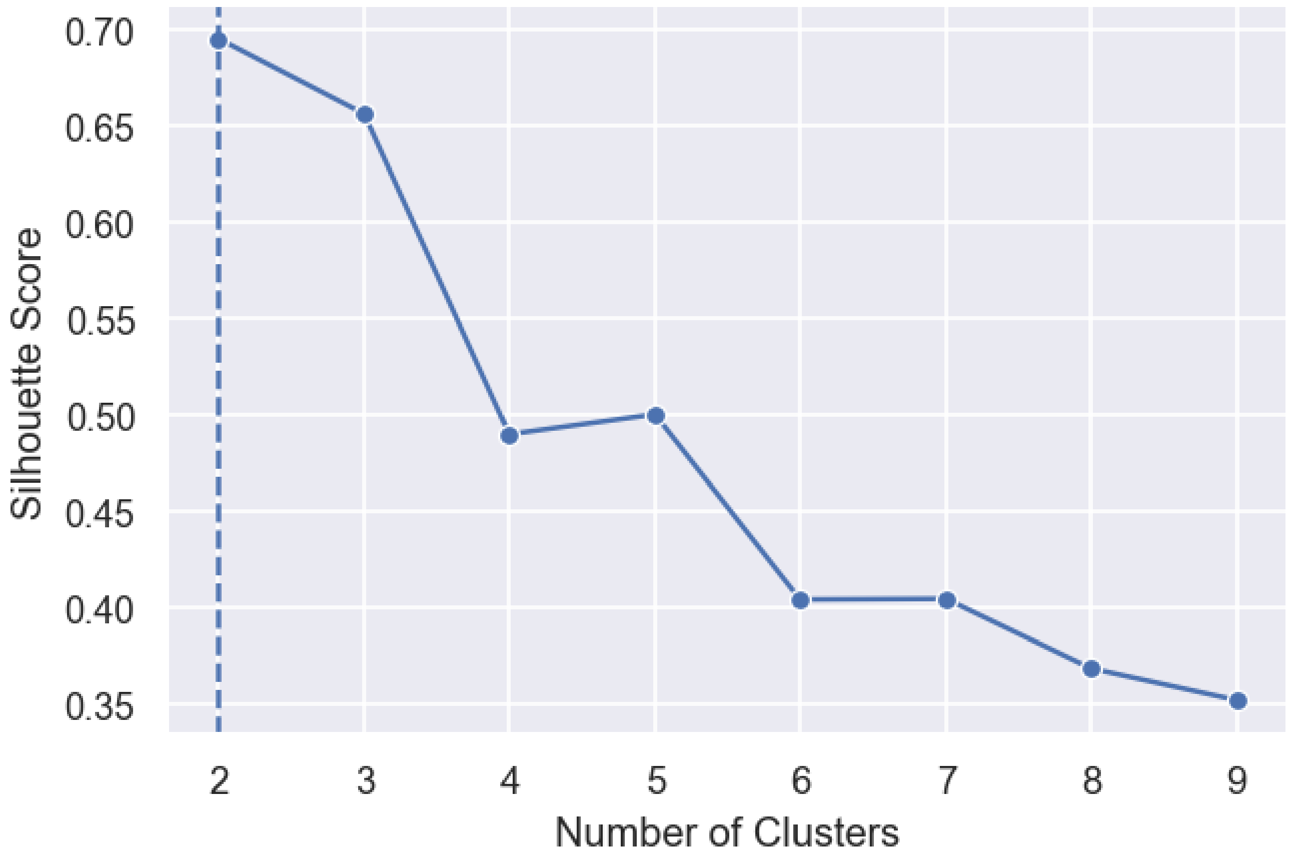
> 2. Calculate the cluster separation b_i from the next closest cluster as the average distance between the sample x_i and all samples in the nearest cluster.
> 3. Calculate the silhoutte s_i as the difference between cluster cohesion and separation divided by the greater of the two:
>
> $$s_i = \frac{b_i - a_i}{\max\{b_i, a_i\}}$$

It can vary between -1 and +1 with coefficients close to +1 indicating instances are generally inside their own cluster and away from other ones, and -1 meaning they may be assigned to the wrong cluster.

Notes

- The *"silhouette score"* is the *"mean Silhouette Coefficient over all samples"*[1][2].
- A _"Silhouette Coefficient is only defined if number of labels is 2 <= n_labels <= n*samples - 1"*[1][2] so this prevents the situation where every observation is its own cluster.
- Slides 20 and 21 (Internal Measures: Cohesion and Separation and Internal Measures: Silhouette Coefficient) in this presentation may help with a visual aid.

Figure 8: K-means Silhouette Scores



However, this plot shows 2 being the highest and 3 (our chosen number of clusters) second. Doesn't this indicate that we should pick 2 instead?

Well its just highlighting one of the limitations of K-means: it performs poorly when there are clusters of different sizes, densities, or the shapes are nonsperical.

If we ballance the classes and increase the number of samples in each cluster we can see it appears more as we would expect.

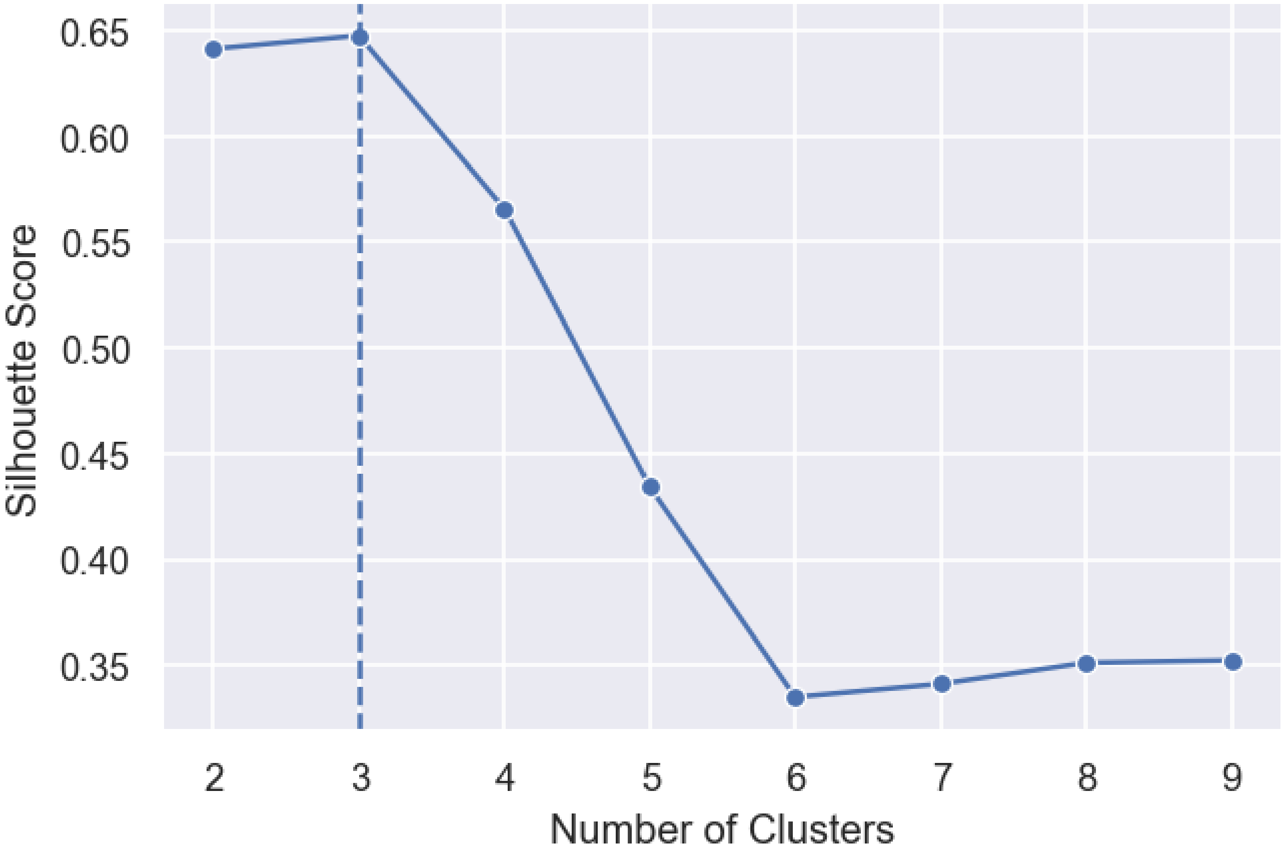


Figure 9: K-means Silhouette Scores (Ballanced Classes)

Silhouette Diagram

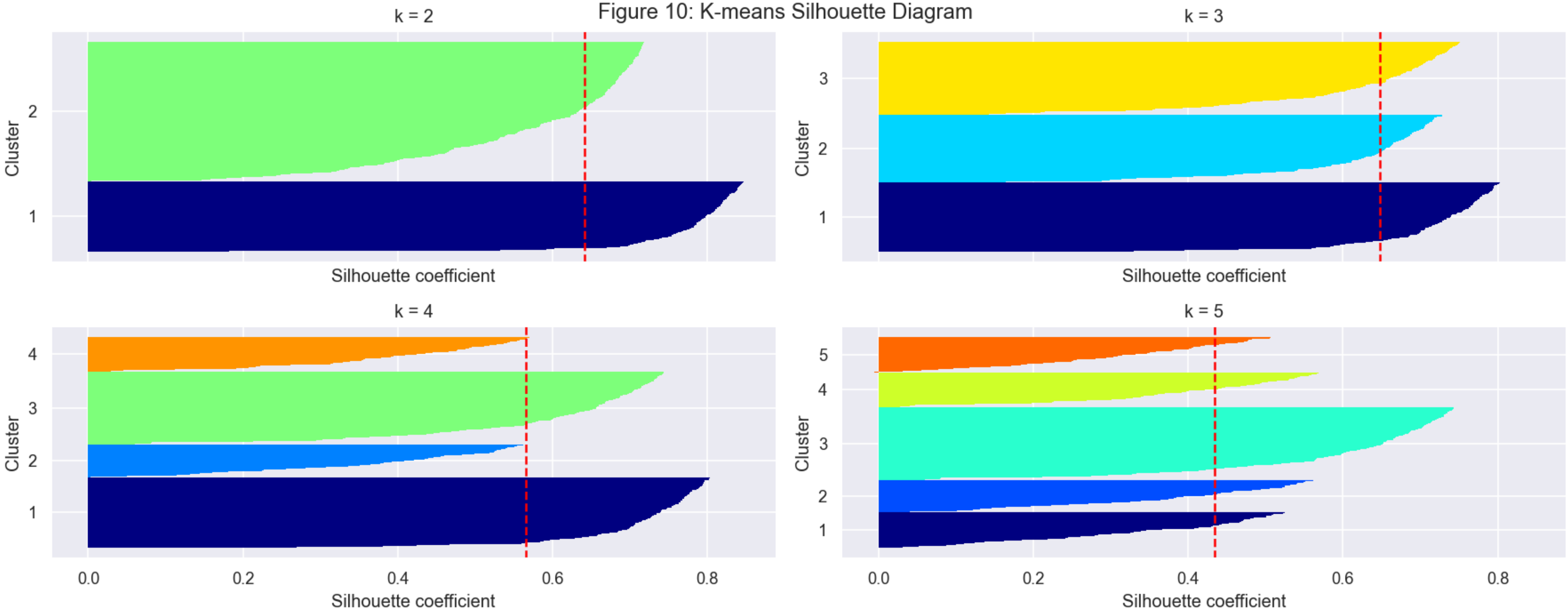
We could also plot every instance's silhouette coefficient^5:

- The **height** indicates the number of instances the cluster contains.
- The **width** is the sorted silhouette coefficient of the instances in the cluster (wider the better).
- The **dashed line** is the silhouette score for the clusters.

Notes

- Silhouettes with similar lengths and widths typically are refective of more optimal clustering[2].
- When most instances in a cluster have a lower coefficient than the score, then the cluster is bad as instances are much too close to other clusters[5].

Figure 10: K-means Silhouette Diagram



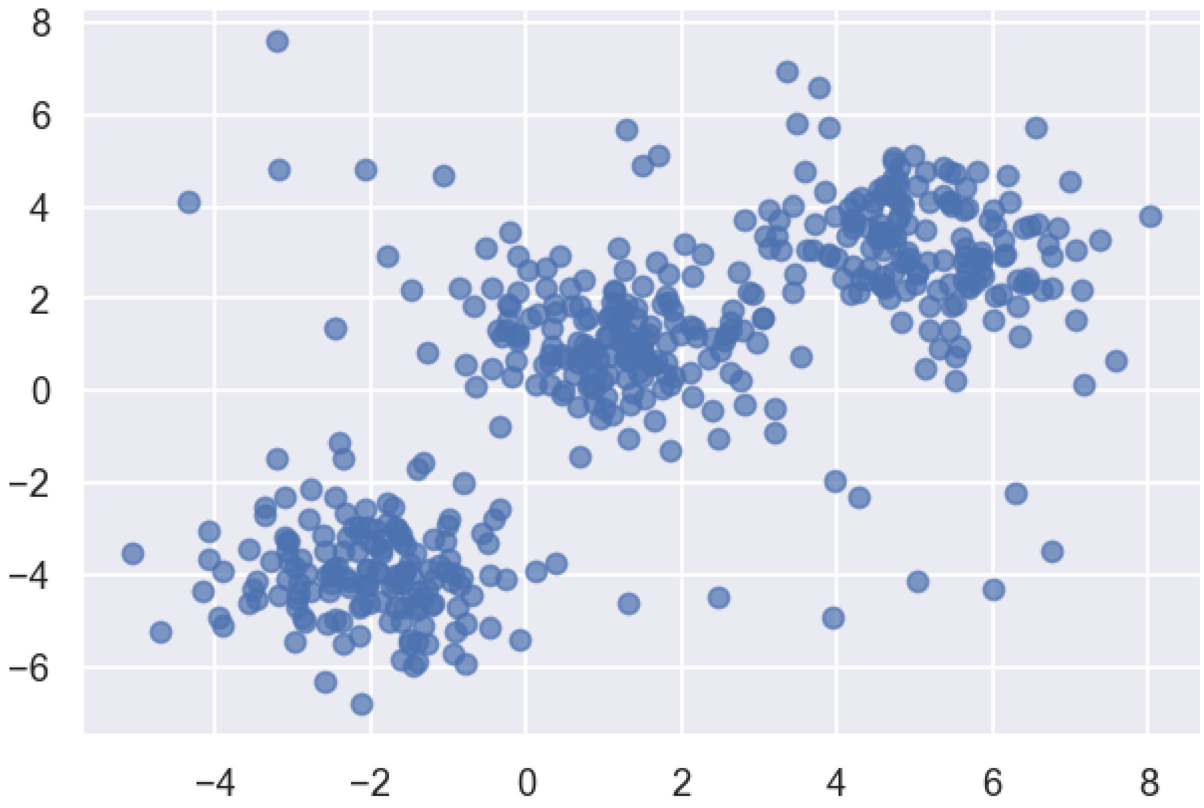
Extra

Soft Clustering

K-means will assign each observation to a cluster, however, sometimes this might not be appropriate.

For example, there could be a situation where most of the observations truly belong to a small number of (unknown) subgroups, and a small subset of the observations are quite different from each other and from all other observations.

Then since Kmeans clustering force every observation into a cluster, the clusters found may be heavily distorted due to the presence of outliers that do not belong to any cluster[1].



Fuzzy C-means

hard clustering is where algorithms assign each sample to exactly one cluster.

soft clustering (or **fuzzy clustering**[9]) is where algorithms assign each sample to one or more clusters.

Fuzzy C-means (FCM)[10] is a popular soft clustering algorithm.

Rather than a hard cluster assignment, each point has probabilities regarding its cluster membership.

The FCM algorithm is similar to k-means[2]:

> 1. Specify the number of k centroids and randomly assign the cluster memberships for each point.
> 2. Compute the cluster centroids $\mu_j \in \{1, \dots, k\}$.
> 3. Update the cluster memberships for each point.
> 4. Repeat steps 2 and 3 until the membership coefficients do not change, or a user-defined tolerance or maximum number of iterations.

The objective function is also similar,

$$J_m = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{m(i,j)} ||\mathbf{x}_i - \mu_j||_2^2.$$

The cluster probability is calculated as:

$$w_{i,j} = \left[\sum_{p=1}^{k} \left(\frac{||\mathbf{x}_i - \mu_j||_2}{||\mathbf{x}_i - \mu_p||_2} \right)^{\frac{2}{m-1}} \right]^{-1}.$$

The center, μ_j, of a cluster is the mean of all samples weighted by the degree that each sample belongs to a cluster ($w_{m(i,j)}$):

$$\mu_j = \frac{\sum_{i=1}^{n} w_{m(i,j)} \mathbf{x}_i}{\sum_{i=1}^{n} w_{m(i,j)}}$$

Notes

- the membership indicator $w_{i,j}$ is not binary ($w_{i,j} \in \{0, 1\}$), iot is a real value representing membership probability ($w_{i,j} \in [0, 1]$).
- m is an exponent, $m \geq 1$, which controls the degree of *fuzziness*. The larger the m, the smaller the cluster membership $w_{i,j}$.
- its not currently availble in `scikit-learn` but I did find a package called `skfuzzy` if you want to try this.

Kernelized K-medoids clustering[11]

Euclidean distance is not always best to apply to structured objects, so we instead may want to use Kernelized K-medoids clustering.

We can kernelize Euclidean distance by observing that:

$$||x_i - x_{i'}||_2^2 = < x_i, x_1 > + < x_{i'}, x_{i'} > -2 < x_i - x_{i'} >$$

Instead of a cluster's centeroid being the mean of all data vectors assigned to a cluster, we can make each centeroid a data vector. The rest of the k-means algorithm is similar:

1. We assign objects to their closest ceneteroid,
2. update the centroids,
3. measure the sum distances to others in teh same cluster
4. pick one which has the smallest sum $m_k = \underset{i:z_i=k}{\arg\min} \sum_{i':z_{i'}=k} d(i,i')$ where $d(i,i')$ is the kernelized Euclidean distance as defined above.

Validating the Clusters[1]

Knowing if clusters represent "true" subgroups in the data, or whether a result of clustering the noise is tricky.

We could maybe look at obtaining an independent set of observations, then seeeing if those observations also display the same set of clusters.

A number of techniques exist for assigning a "p-value" to a cluster, however, there is no current consensus on a best approach.

Notes

- KMedoids is availble in `sklearn_extra.cluster.KMedoids`.
- `scikit-learn-extra` extends `scikit-learn` with useful algorithms that do not satisfy the inclusion criteria. Its part of the `scikit-learn-contrib` github organization which gathers high-quality `scikit-learn` compatible projects.

Notes

- This page on the Scikit-Learn documentation goes over a number of cluster validation approaches.

Associated Exercises

Now might be a good time to try exercises 1 & 2.

References

1. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.
2. Raschka, S., & Mirjalili, V. (2019). Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2. Packt Publishing Ltd.
3. https://github.com/ageron/handson-ml2/blob/master/09_unsupervised_learning.ipynb
4. Arthur, D., & Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding. Stanford.
5. Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media.
6. Elkan, C. (2003). Using the triangle inequality to accelerate k-means. In Proceedings of the 20th international conference on Machine Learning (ICML-03) (pp. 147-153).
7. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
8. Sculley, D. (2010, April). Web-scale k-means clustering. In Proceedings of the 19th international conference on World wide web (pp. 1177-1178).
9. Dunn, J. C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters.
10. Bezdek, J. C. (2013). Pattern recognition with fuzzy objective function algorithms. Springer Science & Business Media.
11. Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.
12. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html