# 11 Choosing $k$ in Clustering

A question that *always* arises with clustering, is how to choose $k$, the number of clusters one should have. There is unfortunately no great answer, no panacea. Remember, this is *unsupervised* learning and there is no rule to tell us what is good and what is bad.

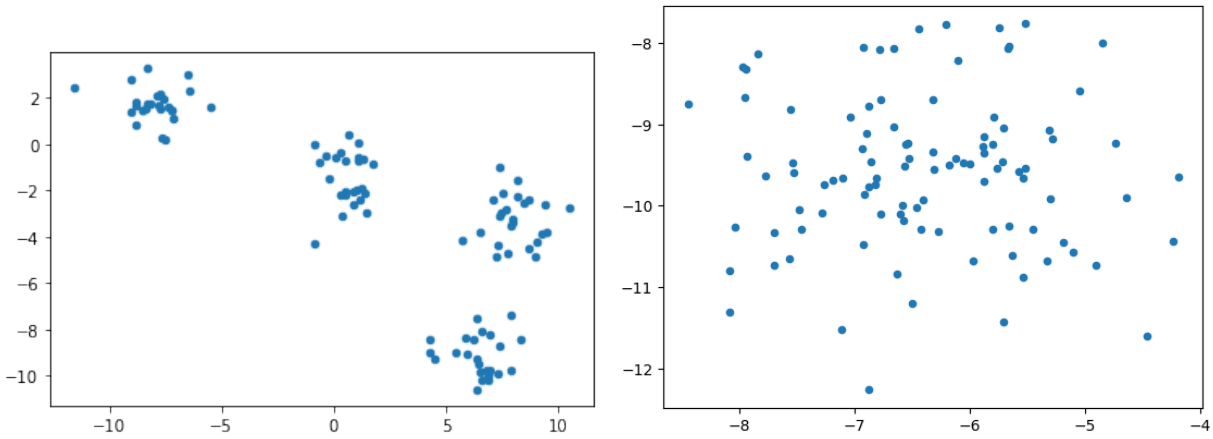
But data scientists demand answers ... or at least guidance. So lets dive in.

We'll describe methods that increase how much you to weight the decision on something you quantify, versus human judgement (which will decrease as we go). But, as we'll see, more complicated quantifications require more assumptions about the type of clustering used, and the assumptions on the data, and meaning of clusters.

## 11.1 Drawing Pictures

Whenever you can, especially when data $X \subset \mathbb{R}^2$, you should plot the data!

Consider the two data sets shown:




We can run whatever clustering algorithm we want on either dataset within any value $k$, and it will always return some "best" clustering. But the left one has 4 well-defined blobs, the right one does not.

More data sets than you might suspect are like the second one. They are often what I would call a *smear* (like cream cheese on a bagel): there may be some denser regions, but no well-defined separations and usually more diffuse around the edges. Clustering such data does not add much value in my opinion.

**What if your data is not in $\mathbb{R}^2$?** Then do some dimensionality reduction. We saw Laplacian Eigenmaps (as part of spectral clustering) in Lecture **10**. Later we will see PCA, MDS, and a variety of distance metric learning methods that will also be available.

Note that if you choose a different distance metric, this may change your plot. This change may update the plot from a smear to a set of blobs.

**Does this work with non-Assignment based clustering?** Sure. It can work for non-centrally symmetric blobs.

If you think the two-moons example is relevant, show me "real world" data that looks like that (which is not artificially created with t-SNE or UMAP). But it does not stop you from plotting, and circling clusters!

**Outliers?** Whatever you think best. Add to a group, or exclude them.

## 11.2 Elbow Method

But for clustering methods we study, we effectively design a score Cost of a clustering, and find (or attempt to find e.g., with Lloyds) the clustering which minimizes that cost. Why not just *consider this cost for each choice of $k$, and return the choice of $k$ that returns smallest cost?*

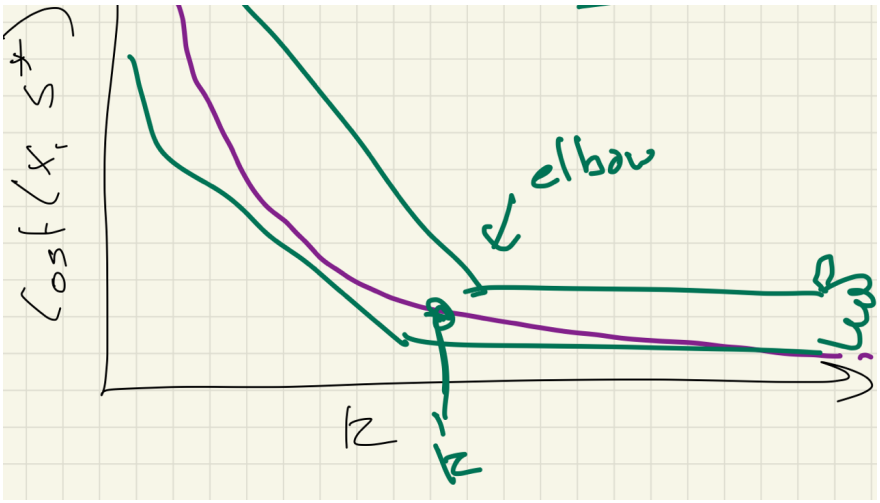
The reason is the cost (for most formulations we discussed) will always *decrease as $k$ increases*.

Lets be concrete. For a clustering defined by sites $s = \{s_1, s_2, \ldots, s_k\}$ for a data set $X$, consider the cost

$$\mathsf{Cost}_2(S, X) = \frac{1}{|X|} \sum_{x \in X} (x - \phi_S(x))^2.$$

Recall, that $\phi_S(x) = \arg\min_{s_j \in S} \|x - s\|$.

For some choice of $k$, if we let $S^*$ be the optimal clustering in terms of $\mathsf{Cost}_2(S^*, X)$, then we can plot the score for each $k$. It will look something like this:




Now that as $k$ increases the curve (in purple) decreases. It should start high, and then decrease towards $0$. When $k = n$ it will be exactly $0$, since each $x \in X$ can be a center, and the distance to the closest one is $0$.

So then how does this help? When if there are well-defined clusters for some choice of $k^*$, then we should see a certain structure. With small $k < k^*$ then we must assign all points in multiple clusters to a single site $s_j$, and so the cost is high. For large $k > k^*$, then we split these clusters into assignments into multiple sites. This has less effect, since the points are already assigned to a compact cluster, so splitting it does not reduce the cost that much (although still some).

Then the *elbow point* in this plot is when the $\mathsf{Cost}_2(X, S^*)$ transitions from rapidly decreasing to slowing decreasing. Like a bent arm (in green!). This transition point can be a good choice for $k$.

**Caveats.** Its not always so cut-and-dry where the elbow point is. Even in well-clustered data, some points are really well separated. And splitting these off into their own cluster causes a large decrease in cost. Whereas some clusters are not so far apart, and the effect of splitting these is more moderate. Also, some clusters themselves will be more spread out, and so splitting them into two still gives a decent decrease in cost. That is to say, *even on well-clustered data*, the elbow point is not always so obvious.

On the positive side, one could argue that this means then getting it slightly wrong may be ok.

But to make matters more complicated, data may have multiple levels of clusters – a natural hierarchy of clusters within clusters. This may induce 2 elbows points. Then which should we choose? Either may be fine! You can choose which gives a more appropriate scale for the purpose of the analysis.

**Extensions.** This works directly for any assignment-based clustering model. Just update the cost function on the $y$-axis.

One can also use with spectral clustering if you map to $\mathbb{R}^t$ and then solve under the $k$-means formulation. Similarly, single-link or other HAC formulations can look at the cost of the merge.

Other clustering algorithms have *some parameter*. For instance, DBScan has $\varepsilon$ and minPts. As either of these vary, the number of clusters will change in a fairly predictable way (usually monotonically). And some similar analysis can be done. Choose a measure to optimize that seems appropriate for the task at hand.

## 11.3 Silhouette Score

OK, the elbow method started to quantify this, but the ultimately resorted to drawing a plot, and using human judgement. The Silhouette score helps quantify this choice.

This assumes some model similar to $k$-means, $k$-medioid, or mean-link HAC. We want to quantify a disjoint clustering $S_1, S_2, \ldots, S_k$. We then quantify the average *inter-cluster distance* for a point $x_i \in X$ in cluster $j$ as

$$a(i) = \frac{1}{|S_j| - 1} \sum_{x \in S_j; x \neq x_i} \mathbf{d}(x_i, x).$$

We also consider the average *replacement cluster score* again for a point $x_i \in X$ in cluster $j$ where

$$b(i) = \min_{j' \neq j} \frac{1}{|S_{j'}|} \sum_{x \in S_{j'}} \mathbf{d}(x_i, x).$$

This is what the score $a(i)$ would be for $x_i$ if it could not use cluster $S_j$ and instead had to use the next best option (which would be cluster $S_{j'}$).

With these values, we can define the Silhouette score for a point $x_i \in X$ as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}.$$

Note that $a(i)$ is not defined if the cluster for $x_i$ is of size 1. In this case, we define $s(i) = 0$.

Note that $s(i) \in [-1, 1]$, and that if $s(i) > 0$, then its "better off" in its cluster than the replacement one scored by $b(i)$.

The *average silhouette score* for the entire clustering is reported as

$$\mathsf{sil}(S, X) = \frac{1}{|X|} \sum_{x_i \in X} s(i)$$

as the average of the silhouette scores of all points.

Finally, we can choose $k$ as the value which results in the clustering with highest average silhouette score.

This is a popular way to fully automate the decision on $k$, but note that it assumes a specific model of what constitutes a good cluster. For instance, this may not be meaningful for density based clustering or single-link HAC.

## 11.4 Bayesian Information Criteria (BIC)

The silhouette is a popular one, but why is it the right measure to judge how effective a clustering is, and compare various values of $k$. Yes, it compares inter-cluster distance with a replacement, but I am sure you could imagine other ways to quantify this.

To resolve this, we can add more modeling, and fit each cluster $S_j$ with a *generative likelihood model* $f_j$. Given a data element $x$, we can measure $f_j(x)$ which evaluates how likely a point is to come from the model. It is a positive probability density function, so it is normalized so its integral is 1.

Lets make this concrete by fitting each cluster $S_j \subset X$ with an isotropic Gaussian (multi-dimensional normal) distribution. The *isotropic* term means we consider the same variance $\sigma$ in each direction; for simplicity, assume $\sigma$ is fixed. To define this we need a center parameter $s_j \in \mathbb{R}^d$. Then

$$f_j(x) = \frac{1}{(\sqrt{2\pi}\sigma)^d} \exp\left(-\frac{\|x - s_j\|^2}{2\sigma^2}\right).$$

And the the likelihood for a cluster, assuming the data is iid from $f_j$ is

$$f_j(S_j) = \prod_{x \in S_j} f_j(x) = \prod_{x \in S_j} \frac{1}{(\sqrt{2\pi}\sigma)^d} \exp\left(-\frac{\|x - s_j\|^2}{2\sigma^2}\right).$$

Indeed if we were to maximize this for $S_j$ over the choice of center $s_j$, the mean $\frac{1}{|S_j|}\sum_{x \in S_j} x$ is the optimal choice: the *maximum likelihood estimator*.

One may assume that the likelihood for an entire $k$-means clustering is then $\prod_{j=1}^{k} f_j(S_j)$, but this is more complicated because of how clusters interact and uncertainty of assignments. See for instance discussion on *Mixture of Gaussians* for both how to resolve this, and how to choose the maximum likelihood $\sigma$. But for our purposes we can assume we have defined a likelihood for a set $f(\mathcal{S}_k)$ for the best clustering $\mathcal{S}_k = \{S_1, S_2, \ldots, S_k\}$.

Moreover, it is typically more numerically stable to work with the *negative log-likelihood*:

$$\ell(\mathcal{S}_k) = -\ln(f(\mathcal{S}_k)).$$

Specifically for one cluster

$$\ell(S_j) = -\ln(f_j(S_j)) = -\sum_{x \in S_j} \ln(f_j) = \sum_{x \in S_j} \left(\frac{\|x - s_j\|^2}{2\sigma^2} - d\ln(\frac{1}{\sqrt{2\pi}\sigma})\right)$$

Because we negated it, we seek to minimize $\ell(\cdot)$ when we sought to maximize the likelihood $f(\cdot)$.

However, the likelihood suffers from the same challenges as other settings: as we increase $k$, we can get a higher likelihood, and smaller negative log-likelihood.

**Penalize more parameters via information theory.** To address this, we can penalize solutions that have more parameters. A model of $k$-means clustering in $\mathbb{R}^d$ has $kd$ parameters, since each site $s_j$ has $d$ parameters.

Through, some information theoretical arguments about a Bayesian model, and considering it in the limit ... we can derive the *Bayesian Information Criteria* (BIC) for a model $M$ with $m$ parameters on $n$ observations as

$$\mathsf{BIC}(M) = -2\ln(f(M)) + m\ln(n).$$

Since our $k$-means algorithm has $kd$ parameters, for a best-fit clustering $\mathcal{S}_k$ of size $k$ we have

$$\mathsf{BIC}(\mathcal{S}_k) = -2\ln(f(\mathcal{S}_k)) + kd\ln(|X|) = \ell(\mathcal{S}_k) + kd\ln(|X|).$$

Now the first term is twice the negative log-likelihood, and so decreases with $k$ increasing. On the other hand, the second term $kd\ln(|X|)$ has a fixed quantity $d\ln(|X|)$ and so increases linearly with $k$.

The value $k$ which minimizes $\mathsf{BIC}(\mathsf{S}_k)$ provides a choice for $k$.

**Extensions.** This method is general for any model with a well-defined likelihood function. This is well-defined for Mixture of Gaussians, and can be made rigorous for $k$-means (sometimes called $X$-means). For HAC models it might be possible (debatable, but some have tried), and difficult for settings like Spectral or DBScan.