#h1 Chapter 17 - Clustering

#h2 Clusters, the bitter cold, and Emperor Penguins

#h3 17.1 - Introduction

#pg Whereas classification is the poster child of supervised learning, clustering undoubtedly is the best example of an unsupervised learning task. Clustering aims to identify data points in our data that group together and, if so, along which dimensions. Even though it is an unsupervised methodology, that does not mean we can blindly press a button and expect good results. Different clustering algorithms work with different assumptions, so a good understanding of the data through careful inspection is always recommended before committing to an algorithm. As pointed out, traditional machine learning methods are gradually replaced by their neural network implementation counterparts. However, the underlying ideas, principles, and pitfalls remain the same. Relying too much on neural networks to find a practical solution without understanding the data obscures serious flaws, biases, shortcomings, and erroneous interpretations of the outputs. We typically do not design neural networks to perform clustering. However, depending on the design and architectures used, they often allow us to see how a network, perhaps trained to perform a different task, was able to learn the task by grouping certain inputs together. However, given the large number of nodes of modern-day deep belief neural networks and the non-linearities it can introduce between inputs and outputs, it is not trivial how the clustering should be interpreted. As before, it is generally a good idea to start exploring a novel data set with some conventional tools, as they are easy to implement and can already provide critical insights.

#im ../../../assets/figures/017/017-01.png 50 256 Figure 17.1 - Male emperor penguins survive the Antarctic winter by huddling together into tight clusters. Members on the outside are exposed to the elements and will push themselves toward the center, making each cluster move and spin around its axis.

#h3 17.2 – A Geometric Interpretation

#pg As with all machine learning algorithms, it is important to understand the geometry of the space your data occupies. Intuitively, we usually imagine clusters as a group of points, a centroid, the origin of the cluster. What distinguishes clusters from one another is defined as a relative distance. It defines a cluster as a collection of points whose inter-distances (the distances between each point of the cluster and each other point of the cluster) are smaller than their intra-distances (the distances between the points of the cluster and points belonging to other clusters). A point is a member of a cluster if 1) its distance to that cluster is the shortest distance compared to the distances of the same point to all other identified clusters and 2) the point is close enough to that cluster. There are, however, other ways we can define clusters. For example, some algorithms (e.g., DBSCAN) consider clusters as regions of the data space where the density of points is greater than some other region or baseline. Other algorithms use a graph-centric approach and define clusters as data points that share greater connectivity than other parts of the graph. Different still, some algorithms use a distribution-based approach, defining clusters as the maximum likelihood that a point was generated by one of several parameterized distributions.

#h3 17.3 – Distance revisited

#pg We have already explored the concept of distance. Specifically, we showed that even though it might come naturally to us to view distance in terms of Euclidean geometry, such geometry can be misleading when applied to the space inhabited by our data. By approximation, Euclidean geometry is a useful and consistent framework for defining geometric relationships between objects in space. When you look up from reading this, you will see your surroundings as a space in which objects have a position, defined by orthogonal three dimensions, expressed in the same units. One unit of moving an object along one axis (say: upward) is the same as moving the same object along a different axis (say: forward). As the dimensions of our data can be expressed in different units, we should be cautious about assuming Euclidean distance is the right metric. If you got to this point without having worked through chapter 12, we would advise reading through it before continuing here. In it, we give a few examples of why defining the correct distance metric is not always trivial and requires some insights into the nature of your data. A final thing to

#im ../../../assets/figures/017/017-02.png 50 256 Figure 17.2 - The recommendations you are offered on services like Netflix and Amazon are based on the likes and dislikes of people like you.

#br

#pg Remember that increasing the dimensionality of our data space (by adding features) can make our distance metrics less meaningful. Increased dimensionality can affect our clustering algorithms substantially. Imagine, for example, a 100-dimensional space in which we have two vectors that are identical for 10 out of these 100 dimensions. The two vectors point in random directions in the remaining 90 dimensions. Suppose we were to compute a measure of the difference in angle between these two vectors using cosine dissimilarity. In that case, the 90 dimensions in which the two vectors are likely to point in very different directions will drown out and obscure the remaining dimensions for which the vectors have identical values. One way around this problem is to use dimensionality reduction, which we will discuss in detail in the next chapter. A similar situation occurs when most values are not random but simply undefined or zero. Such sparse matrices aren't uncommon inputs to your models. Consider Netflix, for example. They recommend certain movies based on your similarity with other Netflix users. If you and others have watched and liked the same movies, a movie the other person has seen (and rated favorably) will probably be a good recommendation for me. But each Netflix user has only seen a small fraction of all the available content. The resulting matrix of users x movie (or show) is thus very sparse. For example, not being particularly a fan of reality TV, for anything that falls in this category, I am of little help to the algorithm in recommending it to someone else. In cancer research, such missing or unspecified values might reflect missing data for that individual. Family histories of cancer might be missing entirely or untrustworthy and too anecdotal to be considered valid inputs to a machine learning model.

#h3 17.4 - Clustering Methodologies

#pg Although distance seems a natural metric to find and define clusters, it is far from the only approach. The many clustering algorithms you may encounter differ mainly in how they define a cluster. For some data sets, this is not necessarily all that relevant, as all definitions seem to apply to the clustering at hand. In other cases, though, how we define a cluster does matter in how well our algorithm can perform. Typically, we can see clustering algorithms as 'clustering' into four different approaches: distance, distribution, graph, and density-based clustering methods. It is important to note that distance still plays a crucial role in all four methods. As we redefine what a cluster is, based on our method, we also redefine the role that distance plays, as we will see soon enough. We will use the three algorithms most closely associated with distance, distribution, and density-based clustering methods to differentiate between these approaches: K-means, gaussian mixture models (GMMs), and DBSCAN. We will only briefly discuss graph-based clustering methods, as they are discussed in much greater detail later in a chapter devoted entirely to graph and network theory. K-means and GMMs use an iterative fitting method called Expectation-Maximization (EM). Given that both clustering algorithms are fundamental and widely used and use the EM framework, we will discuss both these algorithms in greater detail. After that, we aim to build an intuition of when one algorithm thrives and another fails miserably, using a few artificially generated datasets. Finally, we will discuss how we can compute a meaningful success metric and how to set specific parameters to maximize such metrics. We will briefly discuss other clustering algorithms you might encounter or need.

#h3 17.5 - Expectation Maximization - Theory

#pg We use EM methods in many contexts outside of clustering and machine learning. Most notably, we find them in quantitative genetics, a field that studies phenotypes varying continuously (in characteristics such as height or mass) instead of discretely identifiable phenotypes like eye color. Previously, we discussed gradient descent as an iterative method to find the optimal parameters for a machine learning model. It allowed us to find such parameters when no closed-form solution was available. Similarly, the EM algorithm provides an iterative method to find the parameters on a statistical model when these cannot be estimated directly. When it comes to clustering, we assume that all the data has been generated by some fixed number of underlying processes (i.e., each cluster is the result of data generated by one process). But we do not know which data point should go with what process. This is the unsupervised aspect that makes clustering very different from supervised methods like classification. Imagine that I run an experiment in which I have a large number of people lined up to pass through some halfway with white walls. Before each individual entered, we gave them a marker and asked them kindly to place a dot anywhere on the wall, provided it was at their eye height while standing upright. This dot represents their body height, falling a few inches short (the distance between their eyes and the top of their head), but a good representation of their overall height. While I give out the markers, my assistant makes sure to log the sex at birth for each individual passing through. After a week of painstakingly measuring the height of each dot on the wall, we are ready to do our analysis, comparing it with the sex of the person. It is only then my assistant, and I realize our grave mistake. We have no idea who was responsible for each dot, for any of the dots. Usually, we would compute the mean height of the dot for the men and women separately, together with the variance. But we can't. Because we don't know for any of the points what the sex was of the individual creating it. This lack of information is the situation we are in when it comes to clustering.

#im ../../../assets/figures/017/017-03.png 50 256 Figure 17.3 - Our ingenious but severely flawed experiment had us collect plenty of data points, but no way of associating each data point to another variable of interest (sex)

#br

#pg Expectation-Maximization is a method to estimate the statistical parameters of a user-defined number of processes. In our dot experiment, we would set this number of processes to two for each sex. If there is a sufficient statistical difference between these two processes (i.e., on average, men are statistically taller than their female counterparts), EM can help estimate them. It will not, of course, know what these clusters represent, just that they are different statistically. That is up to us as humans to decipher.

#im ../../../assets/figures/017/017-04.png 50 256 Figure 17.4 - Multivariate Normal Distribution. In a univariate case, we can describe the data along a single dimension in our data by a mean and a standard deviation. Suppose we are modeling a statistical process that generates data across multiple dimensions, like the 2D example in this figure. In that case, we need a vector of n means (n = number of) and a covariance matrix of n x n, capturing the covariance between every two dimensions. The covariance between the dimension and itself along the diagonal of this matrix is the variance of that dimension itself.

#h3 17.6 - Expectation Maximization - Practice

#pg As the name suggests, an EM algorithm consists of two phases, expectation and maximization. Like gradient descent, we start with some random set of parameters or perhaps some educated first guess. And like gradient descent, those initial parameters matter and can lead to different outcomes. Therefore, running the algorithm multiple times under different initial conditions is always a good idea, with computational resources permitting. The first phase is the expectation phase, where we compute the output of our current model, given its current parameters. For example, an EM clustering algorithm will compute the predicted cluster label for each data point, given our current parameters. In K-means, these estimated parameters are just the centroids of each cluster. For something more advanced like GMMs, those parameters describe the parameters of the multivariate normal distribution underlying each cluster. We realize that 'Multivariate normal distribution' is a big word to throw at you causally, the reader. Hopefully, by now, you are familiar with the normal or gaussian distribution and its characteristic bell-shaped appearance. We usually use it in a univariate context. Univariate simply means the distribution only captures the probability of observing a particular value (for example, an IQ of 100) along a single dimension. We parameterize the distribution by a mean mu and a variance of sigma. Multivariate normal distributions describe the probability of observing values across multiple dimensions, for example, IQ and SAT scores. To model this distribution, we need two means, mu (one for IQ and one for SAT scores. But we need more than just two variances. In an early chapter, we encountered this scenario where we observed that the dimensions our distribution describes might not be independent. I imagine this is the case for SAT scores and IQ. The higher the IQ, the higher the SAT score. So, in addition to their variance in isolation, we also need to estimate their covariance. A multivariate normal distribution is thus parameterized. by a vector of means of size n (n = the number of different dimensions the distribution describes) and a covariance matrix of size n x n where the elements of the diagonal represent the covariance of each dimension with itself, which is the variance. The off-diagonal elements capture the covariance or correlation between each paring of 2 out of n dimensions. During maximization, we change the parameters based on the expectations from the previous phase. In the case of K-means, we would recompute the location of the centroids based on what data points we predicted to be part of each cluster in the expectation phase. Suppose these new parameters are still substantially different from those we had in the previous phase. In that case, we return to the expectation phase, computing the output based on these new parameters. We continue to cycle through these two phases until the change in our parameters falls below some threshold, in which case we decide the algorithm has converged. We realize this is perhaps too abstract to convey an intuition of how this method yields useful results, especially without any visuals to help the reader. Fear not, the following paragraph will provide such visuals in abundance. Not only to convey the intuition behind EM but also the geometric interpretation of the most widely used distance-based clustering algorithm: K-means.

#im ../../../assets/figures/017/017-05.png 50 128 Figure 17.5 - Of the four points A, B, C, and D, A and B belong to the same cluster, whereas B and D belong to the other cluster. However, the Euclidean distance between the pairs of points informs us that A is closer to A

#h3 17.7 - Distance Based Clustering - K-means

#pg K-means is widely used as a first foray into finding and understanding whether your data shows meaningful clustering. It is partly because it is easy to implement and partly because it is a distance-based clustering method. As discussed above, we generally think of clustering as things grouping in space (or time). We further typically assume or at least have a hard not imagining that a cluster of points is somewhat homogeneous across its dimensions. If this assumption holds to some degree for the space our data lives in, K-means can be a useful tool. But, for example, take the data in Figure 17-5. To a human observer, the points clearly show a clustering, with a closer affinity to some points compared to other points. The human visual system is extremely good at detecting such groupings of elements together. For an algorithm lacking this skill, things are far more ambiguous. For example, take points A, B, C, and D. To you, hopefully, A and B are part of the same cluster, as are C and D. But to an algorithm computing distance, it would find A and C, and B and D to be closer together than A is to B or C is to D.

#im ../../../assets/figures/017/017-06.png 50 256 Figure 17.6 - A map based on the research by Dr. Snow, with each dot representing an individual showing symptoms of cholera. Snow argued that the culprit in the middle of this cluster was the water pump at Broad Street. Although he did not understand how the disease would spread through the water into the nearby human population, it was clear that some aspects of the water and pump were at the root of the Cholera outbreak. Sure enough, officials shut down the pump and the epidemic stopped shortly after that.

#h3 17.8 - The Birth of Epidemiology

#pg In epidemiology, outbreaks happen in time and space, with an infectious disease manifesting as an unusually high number of cases in close geographical proximity of each other, indicating some source or starting point of the disease, with cases clustering around it (initially). We didn't always consider infectious diseases clustering and spreading in time and space. Primarily because, for a long time, we didn't quite understand that a disease can be contagious and that it usually spreads through some form of transmission (it appears that even after two years of the pandemic, some still do not subscribe to this well-established fact). Cue John Snow, not the wall-defending hero from Game of Thrones, but instead an obstetrician living in mid-19th century London. He believed that contaminated water was the cause of the cholera outbreaks that happened quite frequently in his city, contrary to the established idea that 'bad' vapors or smells caused cholera and similar diseases. He did not base this on the knowledge that water can be contaminated with a viral or bacterial disease because germ theory hadn't even been proposed yet. Instead, he based his theory purely on his observations during an 1853 outbreak of cholera that killed hundreds of people. He created a map of the known cases by interviewing residents and hospital staff in the neighborhood. And in the very middle of this cluster of points was the Broad Street water pump. Luckily, local officials were sufficiently convinced and closed the water pump, and the outbreak quickly died out. It did not change completely how we now think about infectious disease, contamination, and transmission, more research by those unconvinced by the established science was needed to do that. However, it is commonly regarded as the founding event of epidemiology. In medieval times, the plague decimating the European continent was moving much slower, gradually moving outward from its initial starting point. It could take weeks, months, or even years before the plague had reached all the farthest corners of the European continent. In our modern world, with air travel and other fast means of transportation, geography appears to be less important in the spread of infectious disease, allowing Covid to spread so rapidly throughout the entire world.

#h3 17.9 - K-means pseudo code walkthrough

#pg [Short Step-by-Step Walkthrough of K-means]

#im ../../../assets/figures/017/017-07.gif 50 256 Figure 17.7 - Animation of Kssmeans at work, iterating over the EM steps to find the 4 clusters.

#h3 17.10 - Metrics of success

#pg Once our clustering algorithm convergences, there isn't a natural way for us to assess the quality of the solution (i.e., the clusters) it has found. Remember that clustering is an unsupervised method, suggesting we do not have any ground truth data. However, it is possible that at least some subset of our data do have labels. In these cases, quite a few metrics of clustering can compute. Mutual information, completeness, homogeneity, and the Rand Index are a few examples that give us some insight into the amount of clustering in our data. In the case of truly unsupervised learning, these metrics are of no use, as they all require ground truth labels to be provided for the data set at hand. The silhouette measure is among the few that are informative without needing ground truth labels. Without any ground truth, we can't rely on accuracy to inform us how well our clustering algorithm or neural network performed compared to other algorithms, the same algorithm with different parameters, or the same algorithm with a different data set. Instead, we typically rely on our earlier definitions of a cluster. Whether we use a density, graph, distance, or distribution-based approach, we expect some deviation from a ground truth of a completely homogenous space. For distance-based metrics, we expect the overall distances between points of the same cluster to be smaller than the distances between points of different clusters. For distribution-based methods, we expect that we observe points as of the distributions

#h3 17.11 - Silhouette Metric

#h3 17.12 - Choosing K to be the correct number of clusters

#pg In some, but not all, clustering methods, we will need to specify how many clusters we expect to find in the data. Our approach is unsupervised, and it is very likely we don't necessarily have a reasonable estimate of the number of clusters for our data. So how do we determine, based on the output of our cluster algorithms, the number of clusters the data contains? It turns out there is an objective measure of performance we can use for this. After K-means has converged for a particular combination of user-defined K number of clusters and initial random starting values for the means of these K clusters, we can compute what is known as the cost. Perhaps somewhat like the cost functions we used in an earlier chapter to perform linear and logistic regression, and classification, the cost in clustering (K-means for now) captures the difference between our solution after convergence and how well, in theory, we could have done. In supervised learning, like regression and classification, we can use the actual outcomes in our training data to compute this cost. In clustering, where we have no such outcome data available, we can nevertheless still define an objective optimal solution. In clustering, we compute cost as the squared distance between all the points to their closest cluster center. To see why this yields a metric that is useful as a measure of performance in the absence of having any ground truth outcome data about the actual labels, let us consider a few edge cases.

#h3 17.13 - Choosing K - edge case 1 - Randomness

#pg In the first scenario, we literally through n points at the wall. We also define n cluster centroids. From these cluster centroids and the data, we compute the cost. This will be high because, on average, points aren't necessarily close to the centroids, as they are randomly and uniformly distributed. But the number is arbitrary and depends on 1) where we ended up putting the clusters, how many points we had in our data and the range of values on each dimension of our input data.

#im ../../../assets/figures/017/017-08.png 50 128 Figure 17.8 - Animation of K-means at work, iterating over the EM steps to find the 4 clusters.

#h3 17.14 - Choosing K - edge case 2 - Perfection to chaos

#pg For cost to be relevant, we will have to look at the relative difference between the compute cost between different scenarios. Imagine first that we start with a set of points for which half are close to one or the two clusters we defined, and the other half very close to the other. We can compute the cost in that scenario, and compute the cost we obtain when we start moving the points outward and away from their respective cluster centroids. We can compare computed costs in this case since all other things are held equal.

#h3 17.15 - What's good for the goose is good for the gander

#pg The above figure shows a steady increase in cost as our clusters become less defined and points spread out outward from the centroid representing the cluster. It also shows the theoretical zero cost, where all points are equal to either centroid, something we do not expect to happen for real data, and something that wouldn't necessarily be that informative if it did. But as luck would have it, the exact relationship we observe between cost and clustering is found for instances where we do not vary the distance between points and the centroids. Still, also when we vary the number of clusters we specify as an input to our K-means clustering algorithm. We will use the scikit-learn make\_blobs tool to generate data with n number of clusters to serve as synthetic input data to our K-means algorithm. In addition, plotting the cost as a function of the number we specify K-means needs to detect can help us determine the optimal number of clusters.

#h3 17.16 - Limitations of Distance-Based Clustering Methods

#pg As we have already discussed, a distance-based clustering method depends on its distance metric providing a suitable measure of which points go together and which points do not. When the space formed by the variables (dimensions) or our data are not independent but instead covary substantially, Euclidean distances between points become misleading and do not align with the natural grouping we nevertheless can easily observe in some cases, as have already seen in Figure 17.5.

#br

#pg Most implementations of K-means will use Euclidean distance as the core metric. However, in principle we could use other distance metrics that consider covariance, such as the Mahalanobis distance. Alternatively, we can turn to distribution-based algorithms, such as gaussian mixture models. Rather than considering the relative distance between points to group them, distribution-based algorithms assume the data is generated by a set of n distributions, resulting in n clusters. By estimating the parameters that describe this distribution we can compute the likelihood that any of these distributions generated any point in our data. Although there are many distributions, the most commonly used is a multivariate normal or gaussian distribution, parametrized by a vector of means and a square covariance matrix. We refer to the clustering method as a gaussian mixture model in this case. Estimating the parameters on the distribution can be done iteratively using the same EM approach we used for K-means, the only difference being that we we

#h3 17.17 - Distribution Based Clustering - Gaussian Mixture Models

#h3 17.18 - GMM pseudo code walkthrough

#h3 17.19 - Density Based Clustering - DBSCAN

#pg Although a distribution-based approach such as a Gaussian Mixture Model offered an improvement over the distance-based approach implemented by K-means, it too can fail to adequately capture what appears to us as natural grouping of points. First, both methods assume that a cluster can be adequality represented by a centroid. This isn't necessarily always true. Clusters may have complex N-dimensional patterns and curvature, representing some interesting underlying non-linear covariance between the variables that form the axes of our data space, see Figure 17.4B. It is here where perhaps the word 'cluster' stops being an adequate descriptor, and instead we should consider describing these sets of interrelated points as manifolds insteas? They are no longer clusters with a localized core or centroid, but rather represent a subspace within our space to which our data points are confined. We will return to the topic of manifolds in two chapters, where we argue that manifolds are the overarching superclass of subspaces, and clusters, classification, regression methods work on a juxtaposition of data point that can be described as specific types of manifolds. In addition, both methods will assign each and every points to a cluster, which not only always makes sense, but also leads to suboptimal results in the clustering it finds for points that do genuinely cluster closely together, relative to other points. Since each point contributes new estimate for cluster centroids (in the case of k-means) and the cluster mean and covariance (in the case of GMM), points that are clearly outliers can have a negative effect on these estimations.

#h3 17.20 - Other clusters algorithms - Hierarchical Clustering

#h3 17.21 - Conclusion

#h3 17.22 - Notebooks

#h3 17.23 - References