STAT 452/652: Statistical Learning and Prediction

Owen G. Ward
owen_ward@sfu.ca

Fall 2023

These notes are largely based on previous iterations of this course, taught by Prof. Tom Loughin and Prof. Haolun Shi.

19 Unsupervised Learning and Clustering

(Reading: ISLR 12.2, 12.4)

Goals for this section

- Lots of data where you don't care about prediction
- Instead just want to understand the structure in the data
- Identify similar observations, or important variables
- Unsupervised Learning a way to solve this
- Will see some common methods, how to implement them

19.1 Unsupervised Learning

Almost every problem we have seen so far has been a supervised learning problem.

- Supervised means you have a response Y which you want to predict using X
- Can compute a measure of how good a model is based on this
- In an unsupervised problem, no clear outcome you want to optimise for
- PCA on it's own an example of this which we will come back to briefly
- Qualitative rather than quantitative, but has an important role in a statistical toolkit

We saw this already briefly when we talked about Principal Component Analysis (PCA). When we use PCA we want to provide a low dimensional summary of each observation in X which explains a large amount of the variance in the data. This is unsupervised as there is no response we want to predict, we just wanter to summarise the data in a smaller number of dimensions. For a dataset X with n observations and p predictors, so that each row of the data is

$$X_i = (X_{i1}, X_{i2}, , \ldots, X_{ip}),$$

for potentially large p. PCA aims to find linear combinations of these predictors, with the k-th principal component written as

$$Z_k = \sum_{j=1}^{p} X_j \phi_{kj},$$

where we have a coefficient ϕ_{kj} for principal component k and predictor j. To be clear, the first principal component of the first observation ($i = 1$) will be $Z_{i1} = \sum_{j=1}^{p} X_{1j}\phi_{1j}$. These ϕ are chosen in a specific way such that each principal component explains as much variance in the original data as possible.

When using PCA we need to decide how many principal components to use. This question is somewhat poorly defined. We need to decide "how much variance" of X is enough, and whether adding an additional principal component captures "enough" variance to be worth adding. There are no hard or fast rules for answering these questions. Before we saw that we can use a Scree plot to answer this question.

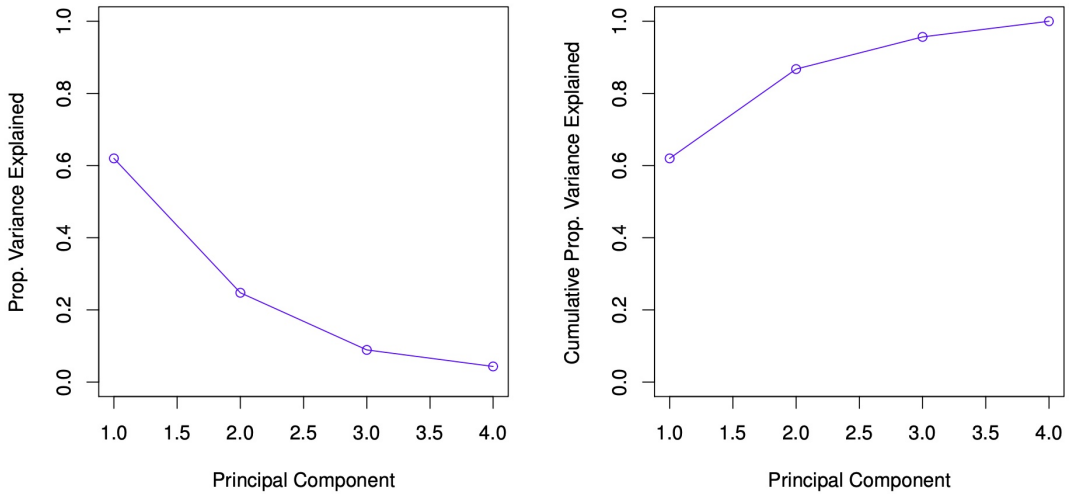

Figure 1: Two ways to construct a scree plot to determine how many principal conponents to use. Any choice here will be somewhat subjective. Taken from ISLR.

19.2 Key ideas for Clustering

- Reasonable to think that in any dataset there are some observations which are "more similar" than others
- Want to have a way to identify these groups
- In a company, maybe you want to identify similar customers Or you collect geological data and want to identify similar groups for further analysis
- Need to have a definition of "similar"?
- Need to define "how similar also" or how may similar groups there are

- Will see two ways to do this

Why do you want to do clustering? What does it solve?

For a clustering problem we will observe X, containing n observations of p predictors and no response variable. We want to find some number of groups in the data, such that observations in the same group are more similar than observations in a different group. Call these groups **clusters**. We will see two popular ways to do this.

19.3 K-Means Clustering

- As seen in the name, decide in advance how many clusters there are in a dataset
- Want to put each of n observations into exactly one of the K clusters
- Observations in the same cluster should differ as little as possible
- Will see a way to formulate this mathematically
- Figure 2 shows a simple example. For a given K, we want to create clusters of similar data. When we change K these clusters can change in quite arbitrary ways

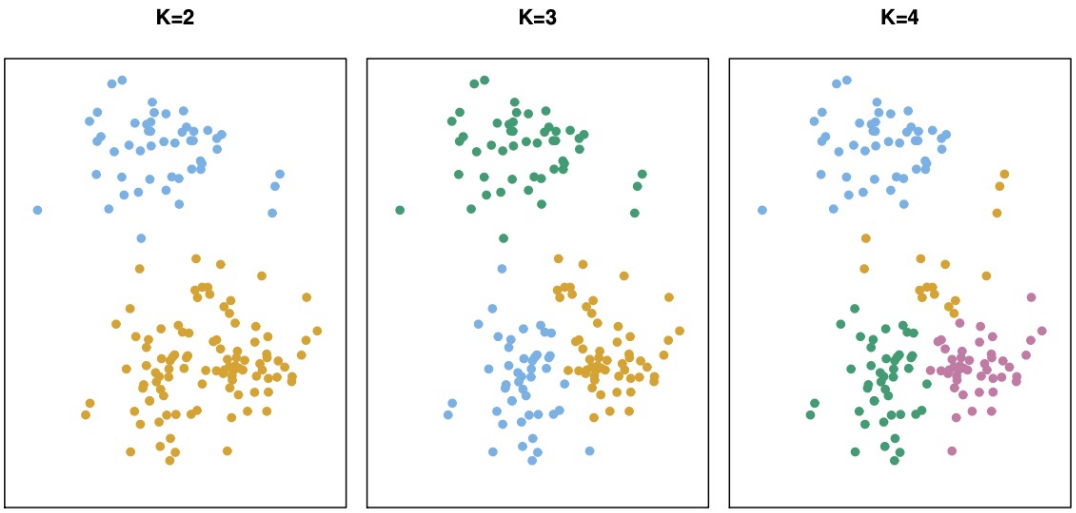

Figure 2: An example of K-means clustering for $p = 2$. As K changes, the composition of the clusters changes. The colours are arbitrary here and can vary from one plot to the next. Taken from ISLR.

The K-means clustering algorithm

- For each cluster at a given K, we want the variability within each cluster to be small
- For cluster k call this $W(C_k)$, where C_k tells you which observations are in the k-th cluster
- Want to put nodes in clusters to make each $W(C_k)$ small

We can write this mathematically as

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

We need to define what each $W(C_k)$ actually is, and how we compute it. To do this we need a way to define a "similarity" between points. Normally this is the **distance** between points. We want all points in a cluster to be close, which gives us

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \tag{1}$$

Lets go through this equation.

- This is saying that the variablity within a cluster is the sum of all pairwise squared distances between points in a cluster
- The $\frac{1}{|C_k|}$ is dividing by the number of observations in the cluster
- $\sum_{i,i' \in C_k}$ means we consider all pairs of nodes in a cluster
- $\sum_{j=1}^{p}(x_{ij} - x_{i'j})^2$ means for each pair, we get the distance between each predictor for each pair, square it and add it up.

So we want to put each node in a cluster such that we minimize this quantity across all K clusters. This is a very hard problem but thankfully there is an easy way to try solve this which can usually give good results.

Algorithm 1 K-Means Clustering

1: Randomly assign each observation to one of K clusters
2: While cluster labels are changing after each iteration:
3: For all observations in each of the current clusters, compute the cluster *centroid*, the mean of each of the p predictors
4: Compute the distance from each observation to these K controids, reassign that observation to the cluster it's nearest to

This algorithm is relatively simple, and the animation on the wikipedia page helps illustrate what is going on, see here.

- We said above that this is a hard problem and that's true
- This simple method, is guaranteed to improve the quantity we want to minimize each time
  - Will put points in the cluster that they are closest too
  - This means how much the clusters are spread out will decrease each time
- However, because we started with the points in random clusters, we may not get to the **best overall** solution
- What it finds is a *local optimum*, the best possible solution near to the random starting point. How can we try get the best overall solution then?

- Try many different random starting points, compare the results
  - We want to find the clusters which give us the smallest value of

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

  - Each random starting value will give us a "local" best clustering, and a value of

$$\text{WSS} = \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

  - Pick the clustering which gives the smallest overall value

Notes on this

- When using kmeans want to start from several different random starting values. The `kmeans` function only does 1 by default, probably want to make that closer to 20
- The algorithm above assumes that K is known in advance. Choosing K can be a hard problem! There are some ways to do this, but somewhat subjective, like in PCA.
  - Maybe a domain expert can help you know how many clusters to expect for your problem?
  - Can plot the quantity you want to minimize as we increase K, see where the plot starts to "level out"
  - Will see this in the example below
- As this clustering procedure uses the distance between the predictors, the scale of these predictors can make a big difference!
  - If one predictor has much larger variability that all others, will have a big influence in the clusters
  - Rescaling them all to have mean 0 and sd 1 will remove that
  - But maybe you want this predictor to be important.
  - Whether to scale your data or not can depend on the problem!
- If we have very large p and/or n then these distances can be computationally expensive to compute.
  - Maybe you could do PCA to reduce p to a smaller size, then do clustering on the PCA output
  - This is common practice in lots of fields
- Clustering can give poor estimates if some of the data is very different. For example, clustering may not deal with *outliers* correctly. This is the case in a lot of statistical methods.

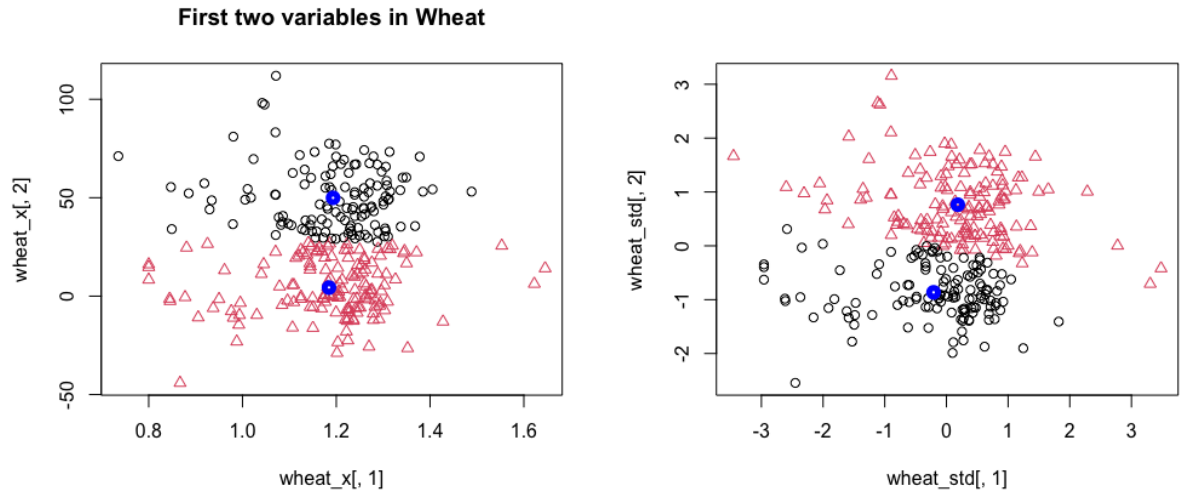

Figure 3: Clustering the first two numeric predictors in the `wheat` data into $K = 2$ clusters. The left shows the original data and the right shows the clusters after standardizing the variables. Here we colour the point based on their cluster. The blue dots correspond to the centers of the clusters.

Example We will use k-means to cluster the Wheat data, using the numeric variables. We will just use the first two predictors, `density, hardness`. The clustering with $K = 2$ is shown in Fig 3. On the left we cluster the raw predictors and on the right we standardize the predictors before clustering. Note the difference in the range on the y-axis, which allows the cluster centers to move more on the x-axis with the standardized data.

We can also see how many clusters seems accurate for this data. To do this, we vary K from $K = 1$ to $K = 10$, each time computing the minimum value of WSS across 20 random starts of the clustering algorithm. We show these in Figure 4. As also this measure is subjective, but for convenience we will use $K = 2$.

The program also performs PCA on all numeric predictors before doing K-means clustering. Each of these procedures will give slightly different clusters of the data. We will see below how to compare them, but there is no ground truth! So all we can check is if they cluster the data in a similar way.

19.4 Hierarchical Clustering (Optional)

- K-means clustering requires you to specify K and then cluster the data to match that choice of K
- Could isntead try a different approach
  - Start with n clusters in the data
  - Gradually merge "similar" data into clusters
  - Eventually, merge all the data into **one** cluster
  - Go back and see where you should have stopped merging the data
- This is known as HIERARCHICAL CLUSTERING

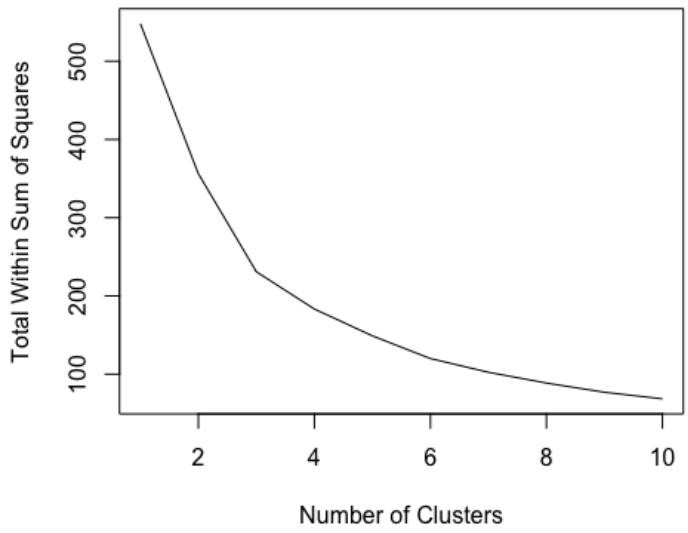

Figure 4: The value of WSS as the number of clusters fit to the data increases. Here $K = 2$ or $K = 3$ might be a reasonable number of clusters.

- Its a bit like growing a regression tree, but in reverse
- This reverse tree structure is known as a DENDROGRAM, and an example for a very small dataset is shown in Fig 5
- There are a few questions that we need to understand
  - As we go from the bottom to the top, we merge this data
  - Observations which are joined lower in the dendrogram are merged first
  - How do we merge clusters? Why do 1 and 6 merge first, rather than 1 and 3?
  - As we go up the tree the number of clusters changes. How many clusters should we have?

Merging Clusters

Looking again at Figure 5, we want to explain how these clusters form. We do the following:

- At each point in the tree, we have some number of clusters.
- We consider all possible pairs of these clusters
- For each pair, we compute how SIMILAR each pair are
  - The most similar pair is merged together to form a single larger cluster
  - Then this procedure is repeated, where now there is one less cluster
- We repeat this until all data is in a single cluster

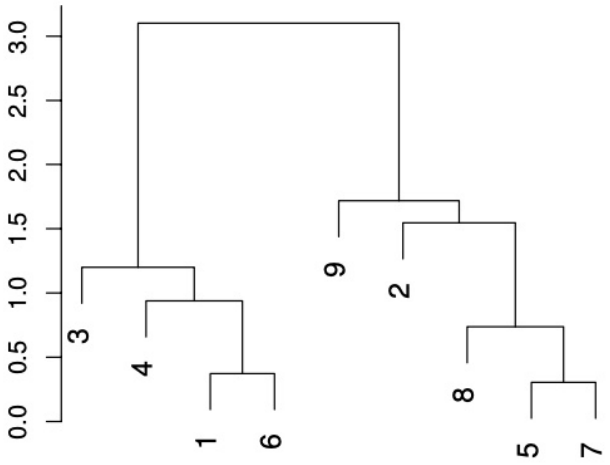

Figure 5: A very simple dendrogram. As we go up the tree clusters merge, until eventually all the data is in a single cluster.

There are several ways to compute how similar two clusters are. These are known as the LINKAGE of two clusters. We will only consider similarity based on the Euclidean distance between two points, but others can make sense in other problems.

- Complete Linkage
  - For all observations in cluster A and cluster B, find the pair which has the largest distance between them.
  - That largest distance is the complete linkage between the two clusters
- Single Linkage
  - Instead of computing the largest possible distance between points in the clusters, instead compute the smallest possible distance
- Average Linkage
  - Take the average of all pairwise distances between all points in the two clusters
- Centroid Linkage
  - Compute the CENTROID of each cluster and get the distance between these centroids

Average and complete linkage are perhaps the most practical of these two metrics. The other two can lead to strange results in certain settings. But using a different linkage will lead to a completely different dendrogram.

Example To compute this hierarchical clustering we need to store the distances between all data. We use the `dist` function to do this. In Figure 6 we show the dendrogram of the first 20 rows of data in the `wheat` dataset, which are created using the `hclust` function. Note that different linkages can result in different clusters. Also, the X-axis on this plot should not be overinterpreted!

The location of the observations here is determined after the cluster. In the left plot, the fact that 9 and 17 are *close* on the x-axis doesn't mean anything. The height at which they split gives an indication of how similar they are. Then, 9 and 17, are only placed in the same cluster when all data is merged into one cluster, so not very similar at all!

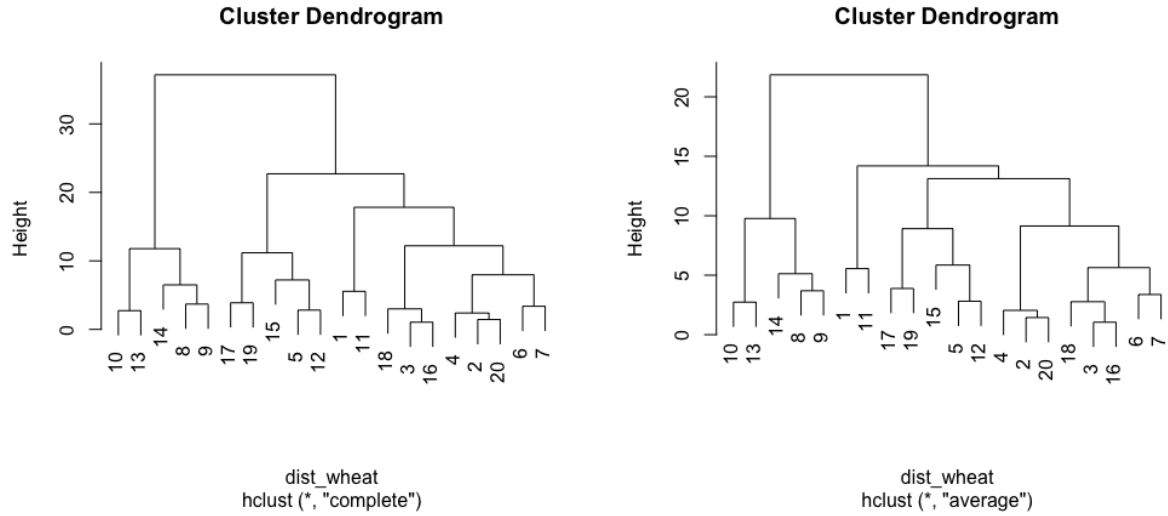

Figure 6: The clustering of the first 20 observations in the wheat data, using complete(left) and average (right) linkage.

Choosing the number of clusters

Just like in K-means, we need to determine how many clusters there should be in the data.

- We can specify how many clusters we want, and see how the data is clustered in that many clusters
- This corresponds to deciding what height to split the tree at
- Alternatively, there is a heuristic for where you should cut the tree
- Each time we merge to clusters we get a height h_i
- This height corresponds to how the similarity of the two clusters that were merged
- Cut the tree at $\bar{h} + 3 \times sd(h)$
- However many clusters the data is in at this height is the estimated for K
- This is largely subjective

19.5 Comparing Different Clustering Algorithms (Optional)

Suppose we perform K-means clustering and hierarchical clustering on a dataset. How do we compare these clusterings?

The simplest thing we could do is look at a table of the clusters from each method and see how they agree. Here we will show it for K-means and hierarchical clustering for the wheat data, finding two clusters for both.

```
> table(km_fit3$cluster, hclusters1)
hclusters1
    1  2
1 89 85
2 78 23
```

- This says that 89 observations were placed in cluster 1 by both methods
- 23 were placed in cluster two by both methods
- Is this good?
- The labels don't mean anything here! The clustering procedures label the data arbitrarily
- If you ran the procedure again all the labels could be swapped
- So, if you consider cluster 1 from K-means to match cluster 2 from hierarchical clustering, then 85 points were placed in the same cluster
- There are methods to check the agreement, which deal with this issue, but we won't have time to discuss these here
- Can still do this if the two methods give you a different number of clusters!

What to take away from this

- Clustering is an important tool for any statistician or data scientist
- Clustering and other unsupervised methods require some skill to use them. There are commonly subjective steps that you need to take
  - Should you standardize the data first or not
  - How many clusters should there be
  - What type of linkage is appropriate for the data you have
- This is where an understanding of the data is essential
- Can't use these methods blindly, need to understand how they work to get something useful from them