Week 10 Lecture Transcript

Recap (slide 1)

Last week, we covered the syntax and basic usage of regular expressions. While this won't be on the exam, I highly recommend becoming familiar with them, as they are powerful and widely used in real-world text processing for data analysis. We also discussed the steps involved in text analysis and explored several R packages, with a focus on the `tidytext` package. We introduced the concept of stop words, explaining why and how we remove them from data. Additionally, we looked at sentiment analysis to determine whether a text is positive or negative. Finally, we covered word importance, which uses formulas to measure how significant a word is to a document relative to a collection of documents.

Outline (slide 2)

This week, we will focus on cluster analysis. We'll begin by explaining what cluster analysis is and why it's important, with some applications and examples. Next, we'll discuss distance measures, including how distance is defined and the different types of distance measures available. Two clustering algorithms will be covered: k-means clustering and hierarchical clustering. Finally, we will introduce dendrograms, which are used in hierarchical clustering.

Cluster Analysis (slide 5)

Cluster analysis is a statistical technique used to group similar data points into clusters according to their similarity on the variables. Each cluster consists of data points that are more similar to each other than to those in other clusters. It helps identify groupings in data without predefined labels.

It is a misconception to view cluster analysis as a visualization technique. While it assigns labels to data points, it does not inherently provide any visual representation of the data. However, visualization can be employed to explore and examine the results of a cluster analysis.

Cluster Analysis (slide 6)

There are several real-world applications of cluster analysis that can help clarify the concept.

One example is market segmentation. A retail company can group customers based on buying behavior, demographics, or preferences to tailor its marketing strategies. For instance, the company may identify a group of customers with high purchasing power and customize products or services for them to maximize profit. On the other hand, for a group of price-sensitive customers, the company might send discount emails to encourage higher sales.

The second application is social network analysis. Cluster analysis can be used to detect groups of users with similar connections or interactions on social media platforms. Some analyses have used scraped Twitter data to identify people who are connected in real life, such as friends and family, by grouping users based on publicly available information. Similarly, websites like LinkedIn can recommend people you may know in the physical world. These recommendations are often highly accurate because they use sophisticated cluster analysis techniques.

Similarly, recommender systems can be developed using cluster analysis. Streaming services like Netflix or Spotify use it to group users with similar preferences, allowing the system to provide personalized recommendations. For example, if a user in a cluster watches a movie or listens to a song, the platform may suggest the same content to other users in the same cluster, as they likely share similar tastes.

Cluster analysis is also applied to genetic data to find groups of genes with similar expression patterns.

Example: 2D data (slide 7)

Let's begin with a simple example using a 2D dataset. This dataset has only two columns, x and y, with about 300 to 500 observations. Since it's in two dimensions, we can easily visualize it using a scatter plot, and this also allows us to manually identify clusters. How many clusters can you spot?

When we refer to a cluster, we typically mean a group of data points that are close to each other and separated from other groups. The first thing we usually focus on is the gap between data points. In the plot, you can see that the bottom-left group is separated from the top group, and the top group has a gap from the bottom-right group. Therefore, this dataset contains three clusters. You can also see this from the next slide.

Do you notice that the bottom-right cluster has a spherical shape, while the other two have elliptical shapes? This is because they were simulated from different multivariate normal distributions, illustrating that clusters can take on various shapes.

Example: Olive Oil Samples from Italy (slide 8)

Visualizing and manually clustering data in 2D may not be too difficult, but what about high-dimensional datasets? For instance, the olive oil dataset has eight variables, so it exists in 8D space. Even with such a high-dimensional dataset, we can still visualize it using methods like a scatter plot matrix or a data tour.

Let's start with the scatter plot matrix. It displays scatter plots for each pair of variables in the lower triangle of the matrix, density plots for each variable on the diagonal, and correlation values in the upper triangle. This gives us various 2D projections of the high-dimensional data. However, this method is limited because it only shows a finite number of 2D projections. To overcome this, we can use the tour approach, which involves rotating the dataset and capturing continuous 2D projections. The tour provides a more comprehensive view of the dataset, and given enough time, it can explore the entire high-dimensional space.

How many clusters can you identify from the scatter plot matrix and the tour? In the bottom-right of the scatter plot matrix, there is a distinct cluster and a vertical line of points that are clearly separated, indicating at least two clusters in the dataset.

For the tour, if you spot a clear cluster, you can pause the tour, use the mouse to brush that cluster, and then resume the tour. For the remaining unbrushed points, if no further separation is visible, they form a single cluster. Otherwise, if a separation can still be found, there are multiple clusters. In this example, although it's challenging to detect, waiting long enough during the tour reveals that there are actually three clusters identifiable by the human eye.

The code for running the tour and creating the scatter plot matrix is fairly straightforward. Please refer to the code section for details.

Example: Olive Oil Samples from Italy (slide 9)

The olive oil samples come from three regions: North, Sardinia, and South of Italy. From the scatter plot matrix, you can see that the cluster we identified earlier corresponds to the North, which is also the largest cluster in the tour.

Manually clustering this high-dimensional dataset is possible, but it is quite difficult and time-consuming. We need a clustering algorithm that can be run and computed numerically by software.

Distance Measure (slide 12)

We want to group similar data points into clusters, but how do we determine if two data points are "similar" or "close"?

A common choice is Euclidean distance, which measures the dissimilarity between two points. It is calculated by summing the squared differences along each axis and then taking the square root. In vector notation, it is the square root of the inner product of the difference between two vectors. The example at the bottom of this slide shows how to compute Euclidean distance for 2D or 3D points.

Distance Measure (slide 13)

There are other distance measures we can use, and I'll provide two examples. One is the Mahalanobis distance, which is similar to Euclidean distance but includes an additional term—the inverse of the S matrix. The S matrix is the variance-covariance matrix of the dataset. This means the Mahalanobis distance first scales the dataset based on this matrix, and then applies Euclidean distance to the scaled data points. The scaling is determined by the variance-covariance matrix, which represents the shape of the dataset.

For example, in the plot on the left, there are around 5000 data points shown with low alpha transparency. The dataset has an elliptical shape, and the variance along the X-axis is clearly larger than that along the Y-axis. Mahalanobis distance scales the data points according to this shape. I've drawn an ellipse representing the 95% confidence region, which reflects the variance-covariance matrix and the shape of the data.

Point A is at the center of the shape, so it remains unchanged by the scaling. The distance between A and C is the same as between A and B because it accounts for the variances of X and Y. Any points inside the ellipse, like D, will have a shorter distance to A than A to B or A to C. Similarly, any points outside the ellipse will have a greater distance from A than B or C.

Another distance measure is the Manhattan distance, which is simple to compute. It sums the absolute differences between each axis. For example, in the plot on the right, the distance between A and B is 2, between A and C is also 2, and between A and D is 3, since the distance between B and D is 1.

Rules of Distance Metric (slide 14)

We've discussed distance measures, but not all distance measures qualify as distance metrics. A metric has a strict mathematical definition and must meet certain conditions. First, the distance from point A to itself must be zero, and the distance from point A to any other point must always be positive. Additionally, the distance from A to B must be the same as from B to A, and the final condition is the triangle inequality.

It's important to note that some divergences in statistics, while sometimes referred to as "statistical distances", are not true distance metrics. Also, keep in mind that Euclidean distance is not suitable for categorical variables.

With these rules in mind, can you come up with your own distance metrics?

K-means Clsutering (slide 16)

K-means is an iterative clustering algorithm that requires specifying the number of clusters (k) in advance.

The goal is to create k clusters that minimize within-cluster variance, which can be understood as the total volume of all clusters. If a cluster is dense and compact, its variance will typically be smaller.

K-means Clsutering (slide 17)

Smaller within-cluster variance indicates that data points within each cluster are more tightly grouped.

Here are two clustering solutions for the same dataset. In the solution on the left, the two clusters are not tightly grouped, with noticeable gaps within each cluster. In contrast, the clusters in the solution on the right are more compact. As shown in the plot titles, the within-cluster variance is significantly larger in the left solution compared to the right.

Thus, within-cluster variance can be used to compare the quality of clustering solutions.

Lloyd's Algorithm (slide 19)

There are several implementations of K-means clustering, and in this unit, we will focus on Lloyd's algorithm due to its simplicity. The algorithm follows four steps:

1. Randomly divide the data into k groups.
2. Calculate the mean for each group.
3. Assign each observation to the group with the closest mean.
4. Repeat steps 2 and 3 until no further observations are reassigned.

Example (slide 20 - 26)

We will use an example dataset with two columns and eight observations to illustrate K-means clustering.

First, we set the desired number of clusters—in this case, 2. The algorithm will then randomly divide the dataset into two groups, which I have labeled with different colors in the plot.

Next, we compute the mean for each cluster, represented by the red dot and the blue dot.

Then, we determine the nearest mean for each observation. In the accompanying table, columns d1 and d2 represent the distances to the first and second means, respectively, while the “near” column indicates which mean is closest. I’ve also drawn dashed lines from each observation to its nearest mean for clarity.

Next, we update the membership for each observation based on the closest mean. If the nearest mean belongs to the first cluster, the membership is updated to 1; if it belongs to the second cluster, the membership is updated to 2.

With these two new clusters, we recompute the means for each cluster. I’ve drawn dashed lines to indicate the movement of the means.

Next, we repeat the previous steps to determine the nearest mean for each observation. If no observations need to be reassigned, the algorithm stops.

Try k-means on olive (slide 27)

In R, you can use the `kmeans` function to run K-means clustering. However, since the number of clusters in the dataset is typically unknown, you need to experiment with different values of k. Be sure to use `set.seed` to ensure the results are reproducible.

We can evaluate the clustering solution using a two-way table. Make sure that large values are positioned on the diagonal by switching the labels, as the labels themselves have no inherent meaning, allowing you the freedom to rename them as needed.

Check Group Means (slide 28)

Here’s some code you can use to check the mean for each group. If the solution is effective, you should observe a clear separation of the means.

Examine Results in 2D (slide 29)

Another way to evaluate the results is to plot them in the data space, such as a scatter plot using two variables from the dataset. You can experiment with different combinations of variables to determine if the clusters are well separated.

Examine Results Using Tour (slide 30)

You can also use tours to examine cluster results.

The K-means algorithm tends to segment the data into roughly equal-sized or spherical clusters, making it effective when the clusters are well-separated and exhibit a similar spherical shape.

If gaps appear within a single cluster, it suggests that K-means has failed to capture important underlying cluster structures. In the tour, we can observe a gap in one of the clusters, indicating a potential issue with the K-means approach.

For more information on visualizing clustering results, check out the book *Interactively Exploring High-Dimensional Data and Models in R* by Di Cook and Ursula Laa.

Choosing Clusters (slide 31)

There is no correct number of clusters. Choosing the number of clusters depends on the context.

Do not choose too many clusters and do not choose too few clusters.

Cluster Statistics (slide 32)

Choosing clusters can be approached numerically. The `fpc` package provides various cluster statistics that can help you compare different clustering solutions and determine the optimal number of clusters.

For more details, you can refer to `?cluster.stats()`. The slides include some commonly used cluster statistics for you to explore.

Hierarchical Clustering (slide 35)

There are two types of hierarchical clustering:

- **Agglomerative**: This method starts with each observation in its own cluster and progressively merges them until all observations are grouped into a single cluster.
- **Divisive** (not covered in this unit): This approach begins with all observations in one cluster and sequentially splits them until each observation is in its own cluster.

Agglomerative clustering is a bottom-up approach, while divisive clustering is a top-down approach. The divisive method is computationally intensive, so it will not be covered in this unit.

Hierarchical Clustering (slide 36)

Hierarchical clustering is a recursive algorithm:

1. Identify the two closest data points in the dataset.
2. Merge them to create a new "data point".
3. Repeat steps 1 and 2 until only one "data point" remains.

The algorithm sounds simple, but how do we measure the distance between two clusters? or a point to a cluster?

Linkage (slide 37 and 38)

Linkage methods are used to measure dissimilarity between sets of observations, such as clusters.

In the diagram, you can see that single linkage measures the minimum distance, while complete linkage measures the maximum distance. Average linkage takes the average of all pairwise distances, and centroid linkage measures the distance between the means of two clusters. Ward's linkage is particularly unique; instead of simply finding the closest points to merge, it identifies the pair that will minimize within-cluster variance and merges those clusters.

Example (slide 39 - 43)

What we display in the table is the distance between each data point, known as the distance matrix. We start by identifying the two closest points in the distance matrix, which is 1, corresponding to the distance between A and C. We then merge these points, as indicated by the red dashed line.

After merging, we need to update the distance matrix. Since we are using single linkage, we calculate the minimum distance between clusters or between a cluster and a data point. For example, the distance between the merged cluster (A, C) and point D is determined by the distance between A and D, as the distance from C to D is greater than from A to D. Using the updated distance matrix, we find the closest pair again and merge them. This process continues until all data points are merged into a single cluster.

Difference Linkage Results (slide 44)

Using different linkage methods will often yield varying results. We can visualize these differences using a dendrogram. In the dendrogram, you can observe that complete, average, and Ward's linkage produce similar results. However, single and centroid linkage generate completely different outcomes. This discrepancy arises possibly because complete, average, and Ward's linkage methods all consider the volume of the clusters, while single and centroid linkage methods are less sensitive to this factor.

Dendrogram (slide 45 and 46)

- Each leaf of the dendrogram represents an individual observation.
- Leaves merge into branches, and branches can merge with either leaves or other branches.
- Merges that occur lower in the tree indicate that the groups of observations are merged earlier and are more similar to one another.

We can cut the tree to partition the data into k clusters.

You can use the `hclust` function to perform hierarchical clustering. To specify where you want to cut the tree, you can use `geom_hline`. The number of clusters corresponds to the number of intersection points between the horizontal line and the vertical branches.

Stability of the Clustering Solution (slide 47 and 48)

One criterion for stability is that the number of clusters remains consistent across a wide range of tolerance levels. The tolerance level refers to the distance between each horizontal line and the next horizontal line, or the distance to the next merge.

In general, you should look for a prolonged stretch of tolerance where the number of clusters does not change.

The hierarchical clustering on the olive data indicates that the longest tolerance level occurs at k = 2. The tolerance level for k = 3 is shorter, while k = 4 has an even shorter tolerance level. However, k = 5 features a longer tolerance level, suggesting that you can choose either k = 2 or k = 5.

Conclusions (slide 50)

Same as the slide.