GENERAL ASSEMBLY

# DAT SCIENCE

## CLASS 11: CLUSTERING AND DIMENSIONALITY REDUCTION

# I. CLUSTERING (K-MEANS AND HIERARCHICAL)
# II. PRINCIPAL COMPONENTS ANALYSIS
# IV. SUPPORT VECTOR MACHINES

# I. CLUSTERING

- Clustering is an **unsupervised method** where observations are grouped together due to their feature similarity, but in a way not optimized to predict a certain class or feature.

- You can think of clustering as jut another form of dimensionality reduction – we are reducing k features used to make the cluster to just 1 feature – the clusters themselves.

- Clustering is useful for (a) recommendation algorithms (especially if the customer does not show clear intent to buy a specific product); (b) reducing dimensionality ahead of prediction, (c) grouping or binning data (such as customer behavior) in an objective, machine-driven way; and (d) visualizing data.

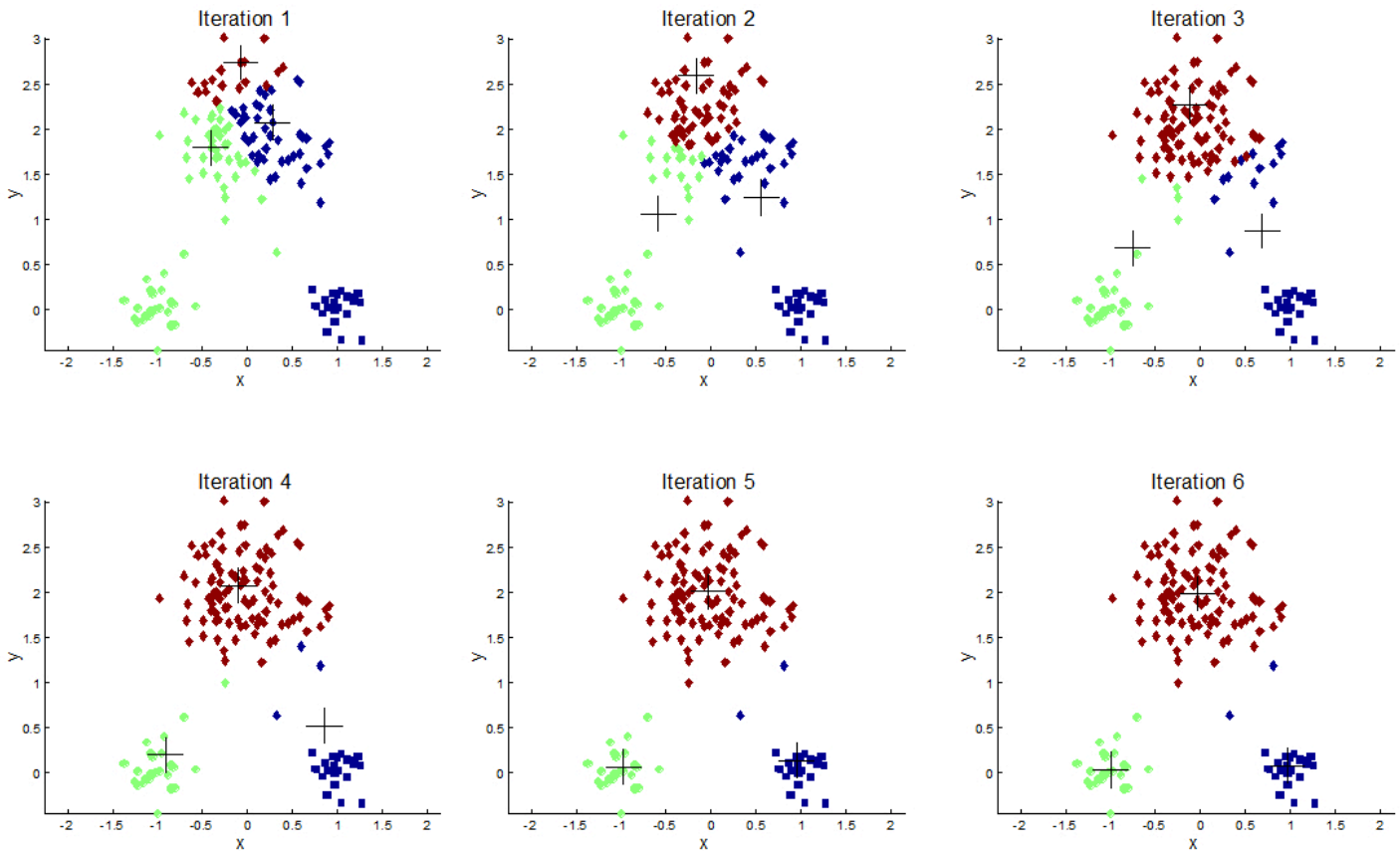
- There are many type of clustering depending on your application. The most common are **k-Means** and **Hierarchical** clustering.

- K-means clustering creates clusters of data around **centroids** the ‘average’ points of all the points in the data.

- Hierarchical clustering groups data together by absolute distance, and then further groups up the hierarchy when distances cross a given threshold.

- The most popular implementation of K-means (called Lloyd's algorithm) uses the following process to 'lock in' on the data's proper centroids:

1. Pick a number of clusters you want to create, k.
2. Assign a random k observations as the centroids of the data set.
3. Calculate the distance of each observation to each k centroids.
4. Assign the observation to the cluster of the nearest k centroid.
5. Re-calculate each centroid to the 'average' value of its cluster.
6. Reassign each observation to the cluster corresponding to its nearest centroid.
7. Continue steps 5-6 until no observations are re-assigned after new centroids are calculated OR the reassignment no longer decreases mean cluster loss (as defined by average Euclidian distance of all points to their centroids).


Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
x
y

# K-MEANS PROS AND CONS



- Pros:
  - Simple, intuitive algorithm
  - Fast execution
  - Effective for two-dimensional or geospatial data

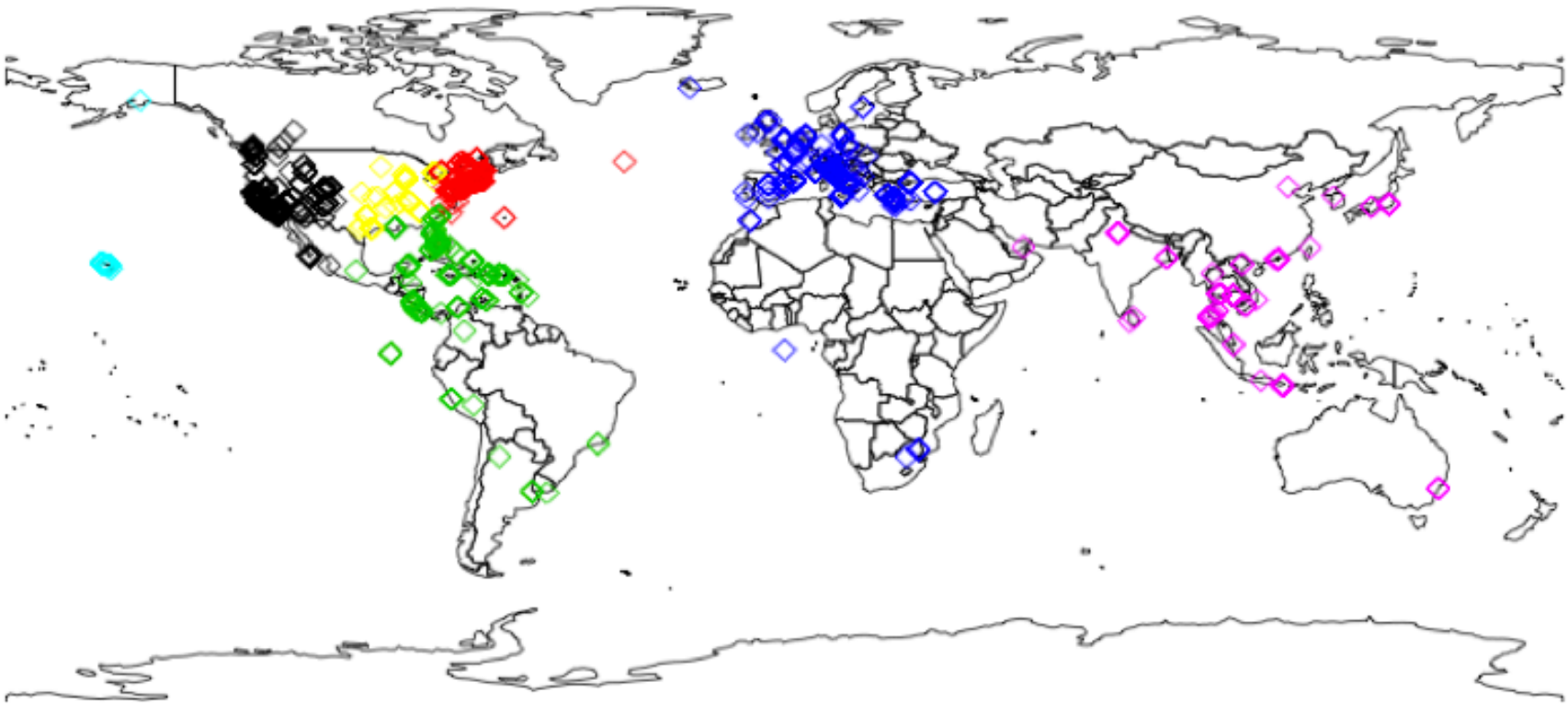
- Cons:
  - Tendency to converge to local minima or dense regions of data (especially if you pick your starting points at random)
  - Produces nonsensical centroids if data is not closely and tightly dispersed

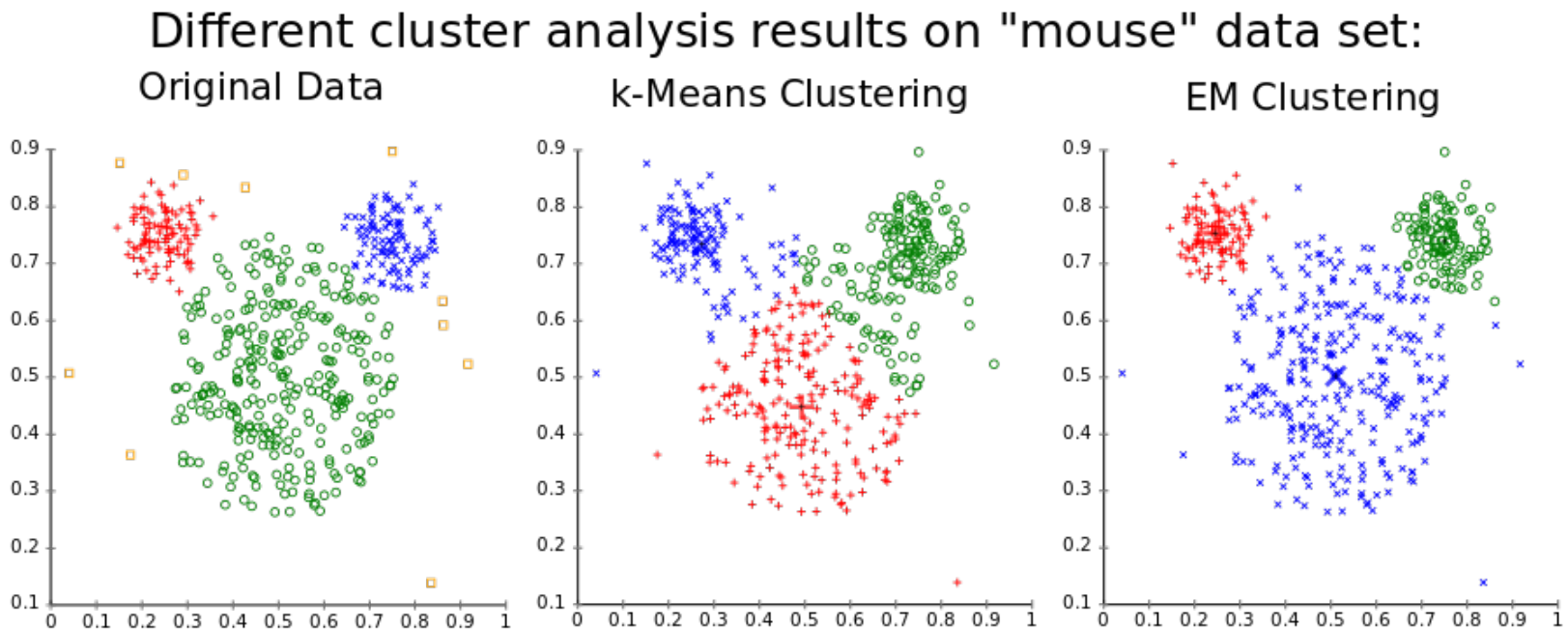
# K-MEANS EXTENSIONS

- Due to the limitations of K-means, a number of related methods are more commonly used to derive cluster meaning:
  - One simple extension of K-means is to repeatedly run the algorithm with different initialization sets, and average the results.
  - K-medoids assigns centroids to actual observations in your dataset.
  - Expectation-Maximization (EM) clustering derives clusters by calculating confidence that the found centroids are the ‘true’ centroids for the dataset.
  - Density Based Scanning (DBSCAN) looks at the median difference *between* points in the cluster, so is robust to non-linear cluster combinations.

# K-MEDOIDS

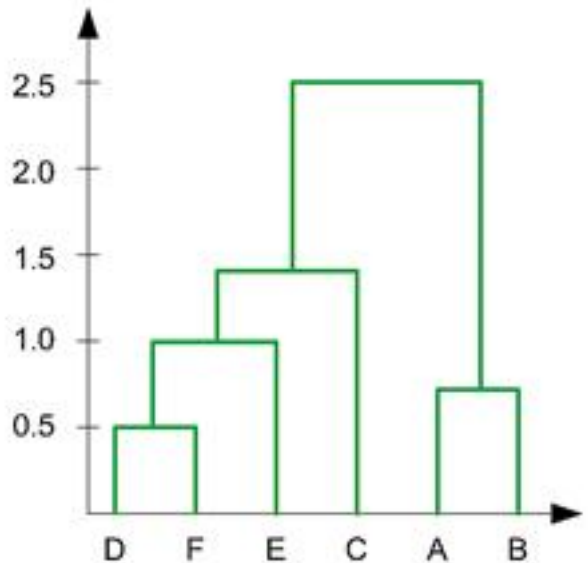





Different cluster analysis results on "mouse" data set:
Original Data
k-Means Clustering
EM Clustering
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

- While k-means clustering tries to determine a given set of discrete clusters, hierarchical clustering attempts to determine the relationship between each observation and cluster.

- Hierarchical clustering is typically visualized via a *dendrogram* (seen to your right), a representation of the relationship of each point against some sort of dissimilarity measure (typically Euclidian distance).



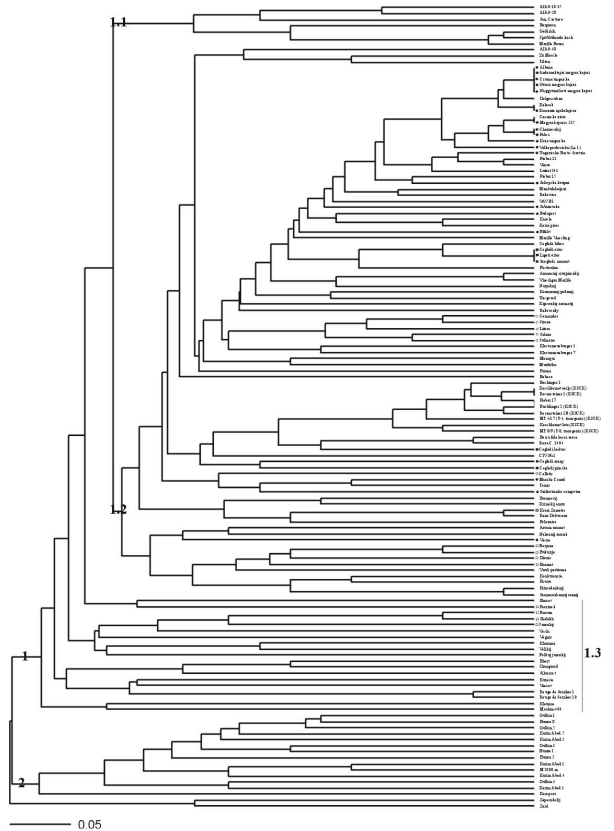
# HIERARCHICAL CLUSTERING



- Hierarchical clustering typically shows you a more accurate representation of similarity between your data than most other techniques.

- However, as  you must chose an arbitrary cutoff point to split your data, you may get highly unbalanced clusters or markedly different numbers of clusters as the data changes through time.

- Examples of use cases include identifying customer behavior groupings or similarities between genes.



# IN-CLASS EXERCISE: CLUSTERING

- Using the Baseball dataset, we will try to cluster teams by two dimensions: their average salaries and their number of hits:
  1. Create an annotated plot of the data showing team name.
  2. Perform K-means clustering and plot the results.
  3. Perform DBSCAN clustering and plot the results.
  4. Create a hierarchical cluster of the data and plot a dendrogram.
  5. Create a cutoff in the dendrogram to create a discrete number of clusters.

# II. PRINCIPAL COMPONENT ANALYSIS

- Recall that in previous classes, you learned feature selection, i.e. a recursive process to determine the bag of variables that allow your model to optimize predictive accuracy.

- But, recall the problems with recursive feature elimination:
  - Current implementations do not explore all possible feature interactions.
  - Elimination is done in a rank-ordered way, which can be misleading as rank and/or significance of a feature can change as you eliminate other features.
  - Implementation of RFE is very computationally intensive.
  - There is no guarantee that you will eliminate linearly related variables if they make it through your initial preprocessing.

- Moreover, recursive feature elimination can become unstable if you have multiple correlated or related features, each of which has a weak contribution to overall accuracy.
- This shows the **curse of dimensionality:** as you add related features to your dataset, it is march harder for a machine learning algorithm to determine which ones are truly predictive, and which are just correlated to the predictive features.
- This typically gives you nonsensical results, where, for example, you see highly positive and negative coefficients if you use linear methods.
- Most social science and business data suffers from this – for example, when you see a rise in revenue, most of your other features rise as well – but which one drove the rise in revenue?

- To get around these issues, data scientists often use principal component analysis to ‘decompose’ the data into n (typically 3 or fewer) dimensions.

- However, the simpler methods you will learn today (such as PCA) will only decompose your data properly if they have a **linear** relationship with one another.

- In addition, once you have transformed your data, it becomes **un-interpretable** as it no longer has a direct connection to any one feature. As such, you will have no way of identifying a problem with an underlying feature by looking at the PCA output.

- In addition, PCA is dependent on the initial scaling of the variables – so if one variable is scaled to have a larger magnitude, it will dominate your decomposition.

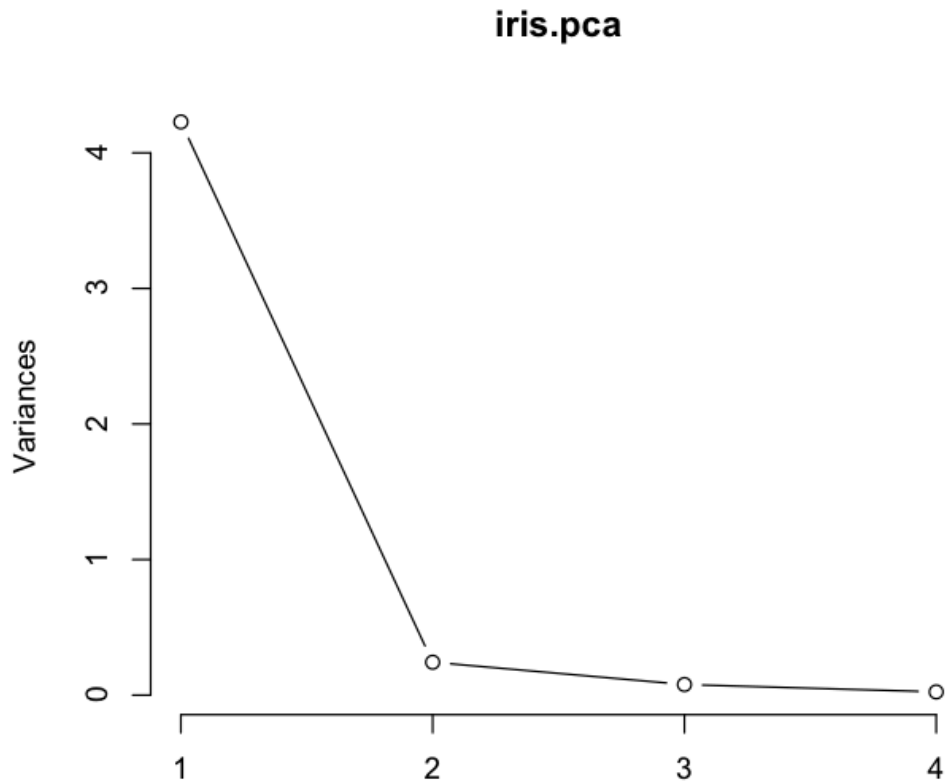
- Principal Component Analysis (PCA) decomposes your (sometimes-correlated) variables into a set of linearly **uncorrelated variables** called principal components.
- Here's how principal component analysis works:
  1. Scale each feature around 0 by subtracting the mean from each observation.
  2. Calculate a covariance matrix between each scaled variable in the data.
  3. Calculate eigenvalues and eigenvectors of the matrix.
     - Eigenvectors effectively work like OLS regressions, by best fitting the data. Each subsequent eigenvector fits the residuals of the previous eigenvector.
  4. Sort the eigenvectors by the size of their corresponding eigenvalues and and determine a cutoff (typically around 0) below which you discard the eigenvector.
  5. Fit each eigenvector to your original data. The first eigenvector you fit is called your **first principal component.**
- See http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf for more information.

- Eigenvectors are in essence a linear system that solve each row of a matrix to zero. Eigenvalues are the scaler or 'fit' on the eigenvector.
- Say you have the following matrix: $$A = \begin{pmatrix} 1 & 2 & 1 \\ 6 & -1 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$
- You can solve for your eigenvalues via the following equation: $$\begin{vmatrix} 1-\lambda & 2 & 1 \\ 6 & -1-\lambda & 0 \\ -1 & -2 & -1-\lambda \end{vmatrix} = 0.$$
- Eigenvectors are solved this way: (using the value of lambda as zero) $$\begin{cases} x + 2y + z &= 0 \\ 6x - y &= 0 \\ -x - 2y - z &= 0 \end{cases}$$

# HOW MANY PRINCIPAL COMPONENTS SHOULD YOU KEEP?



- A **scree plot** lets you look at how much variance is explained by each principal component.
- Apply the **elbow test** to the plot: only take those components to the left of the ‘elbow’ in explained variance.



- Besides PCA, there are a number of other related techniques that rely on matrix decomposition:

  - Singular Value Decomposition
    - Pros: Less prone to overfitting than PCA, faster to compute
    - Cons: Same as PCA: assumes linear relationships between variables, no guarantee that principal components will be fit on features that are most important to dividing classes or helping predict your regressand.

  - Linear Discriminant Analysis
    - Pros: Dimensionality reduction is **supervised**, so differences returned by their nature are relevant to your classification or regression problem.
    - Cons: Assumes input features are normally distributed, assumes output classification follows a functional (linear, quadratic. etc.) decision boundary

- Multidimensional Scaling
  - Pros: does not assume linear relationships between variables or normal distributions among them.
  - Cons:  longer computation time, less available documentation to properly tune, still no guarantee features extracted will be relevant to your regression/ classification question.

# IN-CLASS ASSIGNMENT: PCA



- Perform PCA on the features we used in last class's dataset.
- Plot the first two principal component of the explanatory data.
- Use a scree plot to determine how many principal components you should keep.
- Run a random forest classifier on the retained principal components.
- Evaluate out-of-sample accuracy against non-transformed data.

# III. SUPPORT VECTOR MACHINES

- Support Vector Machines (SVMs) as a set of classifiers that use similar techniques to PCA and other matrix-based dimensionality reduction techniques.

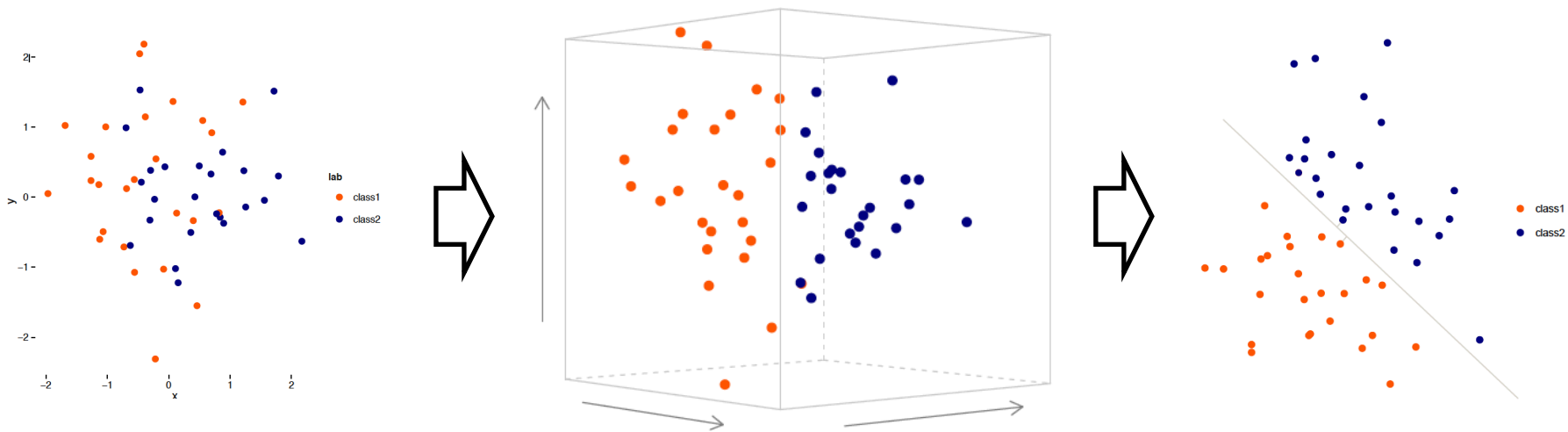
- SVMs apply a function (called a **kernel function**) on the independent features and find the best interaction of the function results that separate the classes (for classification) or best follow the variance of the response feature (for regression).

- SVMs work best if you have multiple ‘weak inputs’ that have some sort of strong underlying signal between them.

# SUPPORT VECTOR MACHINES



- Pros:
  - Accuracy on par with RFs GBMs
  - Very good for picture and text analysis
  - Works well with trending data (unlike RFs!)
  - No ‘jagged edges’ in regression.
- Cons:
  - Computationally intensive
  - Hard to debug
  - Typically no intuition on which kernel function works best



# KERNEL FUNCTIONS

- SVMs include the option of many different kernel functions, including:
  - Linear
  - Polynomial
  - RBF
  - Sigmoid
  - Any Python function you want!

- You'll have to use grid search to determine which kernel is best, which can take a very long time!

# IN-CLASS EXERCISE: SVMS



- Run a SVM with the kernel as ‘polynomial’
- Perform grid search to find the optimal kernel for our use case.
- Compare the accuracy of the final model to the PCA result and the raw random forest.


# ADIOS, AMIGOS!