7.01 K-Means Clustering

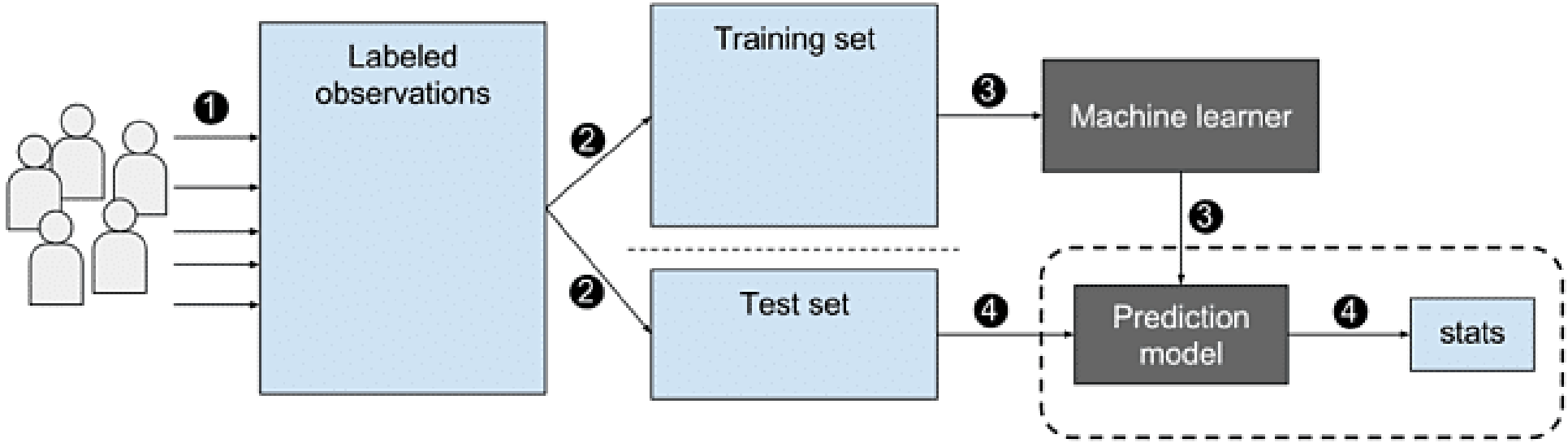
When would clustering be useful? E.g. detect covid cases

Segmenting customers, detecting fraud, genetics, recommenders, social networks

Recap: Supervised Learning

- If you're learning a task under supervision, someone is present judging whether you're getting the right answer. Similarly, in supervised learning, that means having a full set of labeled data while training an algorithm.
- Fully labeled means that each example in the training dataset is tagged with the answer the algorithm should come up with on its own. So, a labeled dataset of flower images would tell the model which photos were of roses, daisies and daffodils. When shown a new image, the model compares it to the training examples to predict the correct label.

Recap: Supervised Learning



Recap: Supervised Learning

There are two main areas where supervised learning is useful:

1. Regression problems
2. Classification problems

Recap: Supervised Learning

Regression problems look at continuous data.

- One use case, linear regression - given a particular x value, what's the expected value of the y variable?
- A more realistic machine learning example is one involving lots of variables, like an algorithm that predicts the price of an apartment in San Francisco based on square footage, location and proximity to public transport.

Recap: Supervised Learning

Classification problems ask the algorithm to predict a discrete value, identifying the input data as the member of a particular class, or group.

- In a training dataset of animal images, that would mean each photo was pre-labeled as cat, koala or turtle.
- The algorithm is then evaluated by how accurately it can correctly classify new images of other koalas and turtles.

Recap: Supervised Learning

Supervised learning is, thus, best suited to problems where there is a set of available reference points or a ground truth with which to train the algorithm.

But these aren’t always available.

Introducing Unsupervised Learning

Clean, perfectly labeled datasets aren't easy to come by. And sometimes, researchers are asking the algorithm questions they don't know the answer to. That's where unsupervised learning comes in.

In unsupervised learning, a model is handed a dataset without explicit instructions on what to do with it.

- The training dataset is a collection of examples without a specific desired outcome or correct answer. The model then attempts to automatically find structure in the data by extracting useful features and analyzing its structure.

Introducing Unsupervised Learning

Depending on the problem at hand, the unsupervised learning model can organize the data in different ways:

1. Clustering
2. Anomaly detection
3. Association

Introducing Unsupervised Learning

- **Clustering**
  - Without being an expert, it’s possible to look at a collection of bird photos and separate them roughly by species, relying on cues like feather color, size or beak shape.
  - That’s how the most common application for unsupervised learning, clustering, works: the deep learning model looks for training data that are similar to each other and groups them together.

Introducing Unsupervised Learning

- **Anomaly detection:**
  - Banks detect fraudulent transactions by looking for unusual patterns in customer's purchasing behavior.
  - For instance, if the same credit card is used in California and Denmark within the same day, that's cause for suspicion.
  - Similarly, unsupervised learning can be used to flag outliers in a dataset.

Introducing Unsupervised Learning

- **Association**:
  - Fill an online shopping cart with diapers, apple sauce and sippy cups and the site just may recommend that you add a bib and a baby monitor to your order.
  - This is an example of association, where certain features of a data sample correlate with other features.
  - By looking at a couple key attributes of a data point, an unsupervised learning model can predict the other attributes with which they're commonly associated.

K-Means Clustering Overview

- It is an unsupervised clustering algo where you may already have a gauge of how many clusters are appropriate
- It is a simple unsupervised algorithm that is used for quickly predicting groupings from within an unlabelled data set
- Predictions are based on the number of centroids present (K) and nearest mean values, given an Eculidean distance measurement between observations

K-Means Clustering Use Cases

- Market Price and Cost Modelling
- Dog Breed Classification
- Insurance Claim Fraud Detection
- Customer Segmentation

K-Means Clustering Algorithm

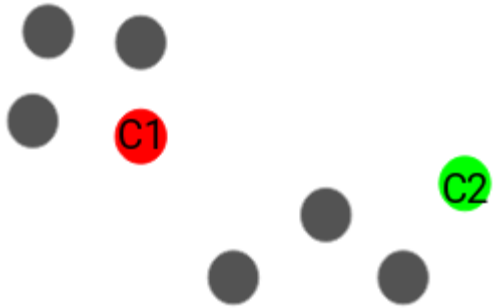
- To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids
- It halts creating and optimizing clusters when either:
- The centroids have stabilized — there is no change in their values because the clustering has been successful.
- The defined number of iterations has been achieved.

K-Means Clustering Algorithm

- To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids
- It halts creating and optimizing clusters when either:
  - The centroids have stabilized — there is no change in their values because the clustering has been successful.
  - The defined number of iterations has been achieved.

K-Means Clustering Algorithm

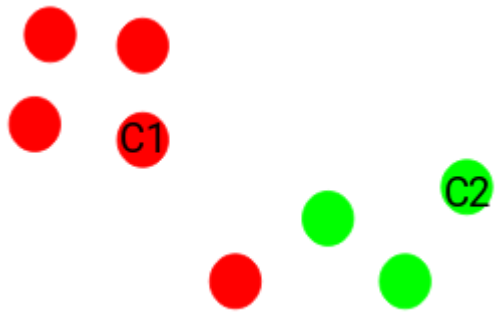
- Step 1: Choose the number of clusters k
  - The first step in k-means is to pick the number of clusters, k.
- Step 2: Select k random points from the data as centroids
  - Next, we randomly select the centroid for each cluster. Let's say we want to have 2 clusters, so k is equal to 2 here. We then randomly select the centroid:



Here, the red and green circles represent the centroid for these clusters.

K-Means Clustering Algorithm

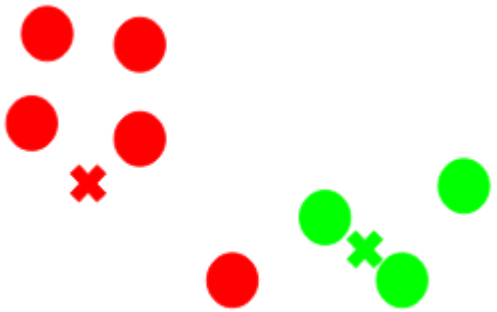
- Step 3: Assign all the points to the closest cluster centroid
  - Once we have initialized the centroids, we assign each point to the closest cluster centroid:



Here you can see that the points which are closer to the red point are assigned to the red cluster whereas the points which are closer to the green point are assigned to the green cluster.

K-Means Clustering Algorithm

- Step 4: Recompute the centroids of newly formed clusters
  - Now, once we have assigned all of the points to either cluster, the next step is to compute the centroids of newly formed clusters:



Here, the red and green crosses are the new centroids.

K-Means Clustering Algorithm

- Step 5: Repeat steps 3 and 4
  - We then repeat steps 3 and 4:

The step of computing the centroid and assigning all the points to the cluster based on their distance from the centroid is a single iteration.

Stopping Criteria for K-Means Clustering

There are two key stopping criteria that can be adopted to stop the K-means algorithm:

1. Centroids of newly formed clusters do not change significantly

   kmeans = KMeans(n_clusters=3, tol=0.00001) # Stop when change < 0.00001
   kmeans.fit(data)

2. Maximum number of iterations are reached

   kmeans = KMeans(n_clusters=3, max_iter=100) # Stop after 100 iterations
   kmeans.fit(data)

K-Means Clustering

- To-Do List
  - Scale your variables
  - Review scatterplot or the data table to estimate the appropriate number of centroids to use for the K parameter value

K-Means Clustering Quality Evaluation

- The goal here isn’t just to make clusters, but to make good, meaningful clusters. Quality clustering is when the datapoints within a cluster are close together, and afar from other clusters.

- The two methods to measure the cluster quality are described below:
  - **Inertia**: Inertia tells how far away the points within a cluster are. Therefore, a small of inertia is aimed for. The range of inertia’s value starts from zero and goes up.
  - **Silhouette score**: Silhouette score tells how far away the data points in one cluster are, from the datapoints in another cluster. The range of silhouette score is from -1 to 1. Score should be closer to 1 than -1.