# DJS Synapse Learning Period
# Week 4

---

## Instructions:

1. We will explore unsupervised learning (Dimensionality reduction & Clustering) this week.
2. Try to go through the resource below before working on the tasks.
3. Make sure you try to understand the algorithms. A complete mathematical understanding is not necessary; an intuition will do.
4. You are given 4 text files as a dataset. Size of the dataset is about 90MB.
5. Directions to use the dataset are mentioned in the notebook.

---

## Resources:

### Unsupervised Learning

Theory: https://www.ibm.com/cloud/learn/unsupervised-learning

### Principal Component Analysis (PCA)

Theory : https://www.youtube.com/watch?v=fkf4IBRSeEc
Implementation : https://www.youtube.com/watch?v=QdBy02ExhGI

### K-Means

Theory+Implementation:
https://towardsdatascience.com/k-means-clustering-explained-4528df86a120

### Agglomerative Clustering

Theory+Implementation:
https://towardsdatascience.com/hierarchical-clustering-explained-e58d2f936323

### DBSCAN

Theory+Implementation: https://youtu.be/C3r7tGRe2eI
Theory+Implementation:
https://towardsdatascience.com/dbscan-clustering-explained-97556a2ad556

# 1.2 Unsupervised learning

Unsupervised was invented a bit later, in the '90s. It is used less often, but sometimes we simply have no choice.

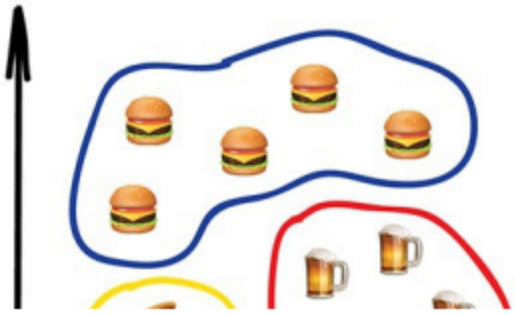
Labeled data is luxury. But what if I want to create, let's say, a bus classifier? Should I manually take photos of million fucking buses on the streets and label each of them? No way, that will take a lifetime, and I still have so many games not played on my Steam account.

There's a little hope for capitalism in this case. Thanks to social stratification, we have millions of cheap workers and services like Mechanical Turk who are ready to complete your task for $0.05. And that's how things usually get done here.

Or you can try to use unsupervised learning. But I can't remember any good practical application for it, though. It's usually useful for exploratory data analysis but not as the main algorithm. Specially trained meatbag with Oxford degree feeds the machine with a ton of garbage and watches it. Are there any clusters? No. Any visible relations? No. Well, continue then. You wanted to work in data science, right?

[redacted]

## Clustering




*"Divides objects based on unknown features. Machine chooses the best way"*

Nowadays used:

- For market segmentation (types of customers, loyalty)
- To merge close points on a map
- For image compression
- To analyze and label new data
- To detect abnormal behavior

Popular algorithms: K-means_clustering, Mean-Shift, DBSCAN

[redacted]

Clustering is a classification with no predefined classes. It's like dividing socks by color when you don't remember all the colors you have. Clustering algorithm trying to find similar (by some features) objects and merge them in a cluster. Those who have lots of similar features are joined in one class. With some algorithms, you even can specify the exact number of clusters you want.

An excellent example of clustering — markers on web maps. When you're looking for all vegan restaurants around, the clustering engine groups them to blobs with a number. Otherwise, your browser would freeze, trying to draw all three million vegan restaurants in that hipster downtown.

Apple Photos and Google Photos use more complex clustering. They're looking for faces in photos to create albums of your friends. The app

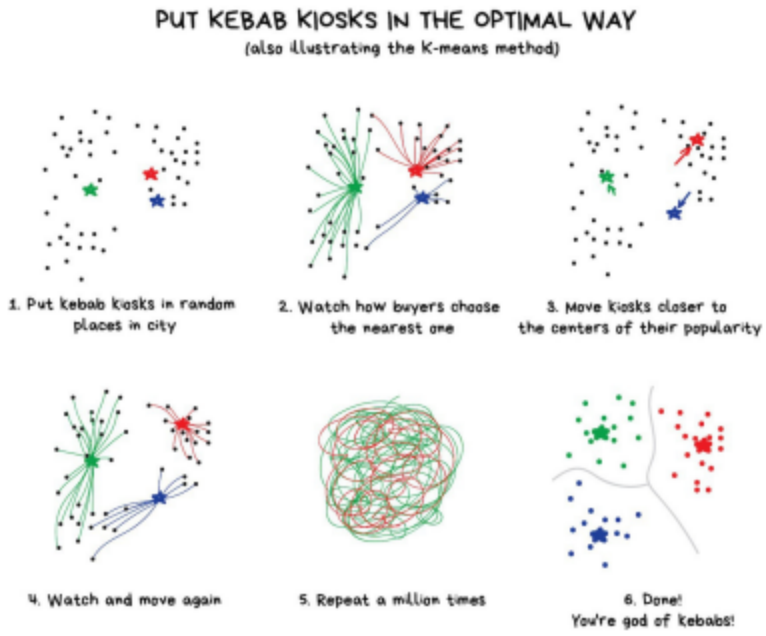
doesn't know how many friends you have and how they look, but it's trying to find the common facial features. Typical clustering.

Another popular issue is image compression. When saving the image to PNG you can set the palette, let's say, to 32 colors. It means clustering will find all the "reddish" pixels, calculate the "average red" and set it for all the red pixels. Fewer colors — lower file size — profit!

However, you may have problems with colors like Cyan■-like colors. Is it green or blue? Here comes the K-Means algorithm.

It randomly sets 32 color dots in the palette. Now, those are centroids. The remaining points are marked as assigned to the nearest centroid. Thus, we get kind of galaxies around these 32 colors. Then we're moving the centroid to the center of its galaxy and repeat that until centroids stop moving.

All done. Clusters defined, stable, and there are exactly 32 of them. Here is a more real-world explanation:

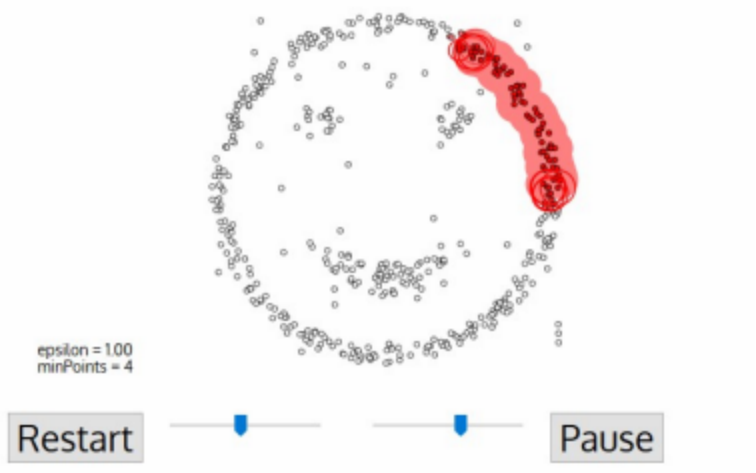


Searching for the centroids is convenient. Though, in real life clusters not always circles. Let's imagine you're a geologist. And you need to find some similar minerals on the map. In that case, the clusters can be weirdly shaped and even nested. Also, you don't even know how many of them to expect. 10? 100?

K-means does not fit here, but DBSCAN can be helpful. Let's say, our dots are people at the town square. Find any three people standing close to each other and ask them to hold hands. Then, tell them to start grabbing hands of those neighbors they can reach. And so on, and so on until no one else can take anyone's hand. That's our first cluster. Repeat the process until everyone is clustered. Done.

A nice bonus: a person who has no one to hold hands with — is an anomaly.

It all looks cool in motion:




Interested in clustering? Check out this piece The 5 Clustering Algorithms Data Scientists Need to Know

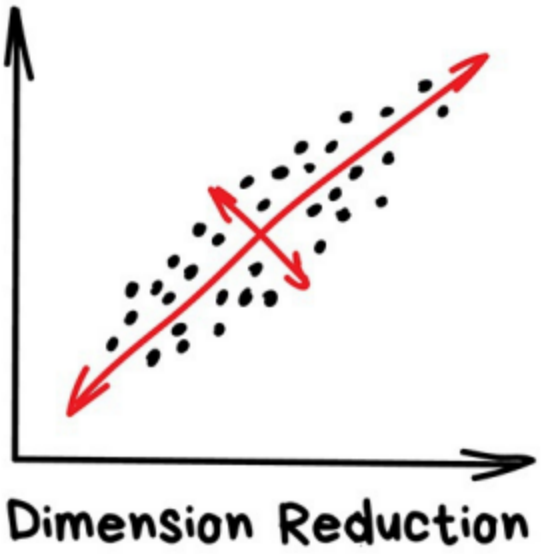
Just like classification, clustering could be used to detect anomalies. User behaves abnormally after signing up? Let the machine ban him temporarily and create a ticket for the support to check it. Maybe it's a bot. We don't even need to know what "normal behavior" is, we just

upload all user actions to our model and let the machine decide if it's a "typical" user or not.

This approach doesn't work that well compared to the classification one, but it never hurts to try.

[redacted]

## Dimensionality Reduction (Generalization)




*"Assembles specific features into more high-level ones"*

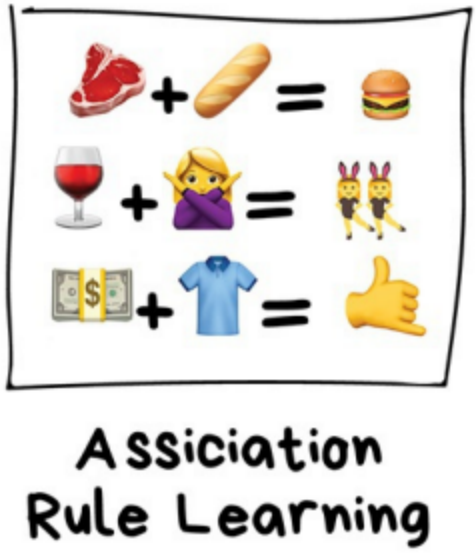
Nowadays is used for:

- Recommender systems (★)
- Beautiful visualizations
- Topic modeling and similar document search
- Fake image analysis

- Risk management

Popular algorithms: Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Latent Dirichlet allocation (LDA), Latent Semantic Analysis (LSA, pLSA, GLSA), t-SNE (for visualization)

[redacted]




This includes all the methods to analyze shopping carts, automate marketing strategy, and other event-related tasks. When you have a sequence of something and want to find patterns in it — try these thingys.

Say, a customer takes a six-pack of beers and goes to the checkout. Should we place peanuts on the way? How often do people buy them together? Yes, it probably works for beer and peanuts, but what other sequences can we predict? Can a small change in the arrangement of goods lead to a significant increase in profits?

Same goes for e-commerce. The task is even more interesting there — what is the customer going to buy next time?

No idea why rule-learning seems to be the least elaborated upon category of machine learning. Classical methods are based on a head-on look through all the bought goods using trees or sets. Algorithms can only search for patterns, but cannot generalize or reproduce those on new examples.

In the real world, every big retailer builds their own proprietary solution, so nooo revolutions here for you. The highest level of tech here — recommender systems. Though, I may be not aware of a breakthrough in the area. Let me know in the comments if you have something to share.