Announcements

- Homework 3 due in **Wednesday, October 11 at 8pm**
- Quiz 4 due **Wednesday, October 11 at 8pm**
  - We'll leave it open for a bit longer

Boosting as Gradient Descent

- Note that

$$F_{t+1}(x_i) \approx y_i$$

Boosting as Gradient Descent

- Note that

$$F_t(x_i) + f_{t+1}(x_i) = F_{t+1}(x_i) \approx y_i$$

- Rewriting this equation, we have

$$f_{t+1}(x_i) = F_{t+1}(x_i) - F_t(x_i)$$

Boosting as Gradient Descent

- Note that

$$F_t(x_i) + f_{t+1}(x_i) = F_{t+1}(x_i) \approx y_i$$

- Rewriting this equation, we have

$$f_{t+1}(x_i) = F_{t+1}(x_i) - F_t(x_i) \approx \underbrace{y_i - F_t(x_i)}_{\text{“residuals”, i.e., error of the current model}}$$

Boosting as Gradient Descent

- In other words, at each step, boosting is training the next model f_{t+1} to approximate the residual:

$$f_{t+1}(x_i) \approx \underbrace{y_i - F_t(x_i)}_{\text{"residuals", i.e., error of the current model}}$$

Boosting as Gradient Descent

- **Algorithm:** For each $t \in \{1, \ldots, T\}$:
  - **Step 1:** Train f_{t+1} using dataset

$$Z_{t+1} = \left\{\left(x_i, y_i - F_t(x_i)\right)\right\}_{i=1}^{n}$$

  - **Step 2:** Take

$$F_{t+1}(x) = F_t(x) + f_{t+1}(x)$$

- Return the final model F_T

Boosting as Gradient Descent

- Residuals are the gradient of the squared error $\tilde{L}(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$:

$$-\frac{\partial \tilde{L}}{\partial \hat{y}}(F_t(x_i); y_i)$$

Boosting as Gradient Descent

- Residuals are the gradient of the squared error $\tilde{L}(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$:

$$-\frac{\partial \tilde{L}}{\partial \hat{y}}(F_t(x_i); y_i) = y_i - F_t(x_i)$$

Boosting as Gradient Descent

- Residuals are the gradient of the squared error $\tilde{L}(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$:

$$-\frac{\partial \tilde{L}}{\partial \hat{y}}(F_t(x_i); y_i) = y_i - F_t(x_i) = \mathrm{residual_i}$$

- For general $\tilde{L}$, instead of $\{(x_i, y_i - F_t(x_i))\}_{i=1}^n$ we can train f_{t+1} on

$$Z_{t+1} = \left\{\left(x_i, -\frac{\partial \tilde{L}}{\partial \hat{y}}(F_t(x_i); y_i)\right)\right\}_{i=1}^n$$

Boosting as Gradient Descent

- **Algorithm:** For each $t \in \{1, \dots, T\}$:
  - **Step 1:** Train f_{t+1} using dataset

$$Z_{t+1} = \{(x_i, y_i - F_t(x_i))\}_{i=1}^{n}$$

  - **Step 2:** Take

$$F_{t+1}(x) = F_t(x) + f_{t+1}(x)$$

- Return the final model F_T

Boosting as Gradient Descent

- **Algorithm:** For each $t \in \{1, \ldots, T\}$:
  - **Step 1:** Train f_{t+1} using dataset

$$Z_{t+1} = \left\{ \left(x_i, -\frac{\partial \tilde{L}}{\partial \hat{y}} (F_t(x_i); y_i) \right) \right\}_{i=1}^{n}$$

  - **Step 2:** Take

$$F_{t+1}(x) = F_t(x) + f_{t+1}(x)$$

- Return the final model F_T

Boosting as Gradient Descent

- Casts ensemble learning in the **loss minimization framework**
  - **Model family:** Sum of base models $F_T(x) = \sum_{t=1}^{T} f_t(x)$
  - **Loss:** Any differentiable loss expressed as

$$L(F; Z) = \sum_{i=1}^{n} \tilde{L}(F(x_i), y_i)$$

- Gradient boosting is a general paradigm for training ensembles with specialized losses (e.g., most NLL losses)

Gradient Boosting in Practice

- Gradient boosting with depth-limited decision trees (e.g., depth 3) is one of the most powerful off-the-shelf classifiers available
  - **Caveat:** Inherits decision tree hyperparameters

- XGBoost is a very efficient implementation suitable for production use
  - A popular library for gradient boosted decision trees
  - Optimized for computational efficiency of training and testing
  - Used in many competition winning entries, across many domains
  - https://xgboost.readthedocs.io

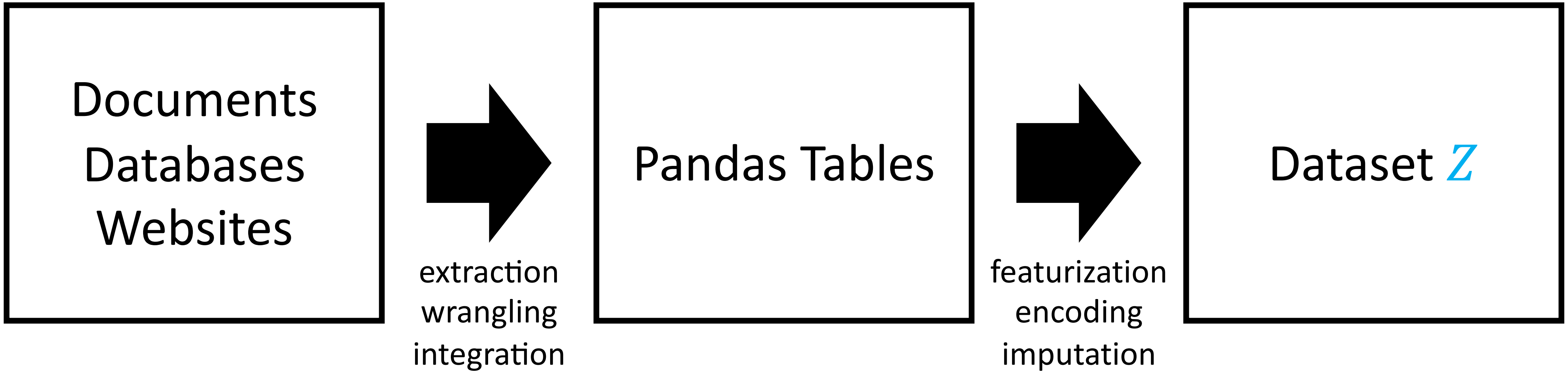
Data Engineering

- We have been assuming that the dataset Z is given

- For many problems, building Z is >80% of the work!
  - What is the prediction task we want to solve?
  - **Data integration:** Integrate data across many data sources

- Focus of CIS 5450, but we give a summary

Typical Data Engineering Pipeline



Data Collection Challenges

- Even gathering the relevant data can be a huge challenge
  - Proprietary/private data
  - Data must be labeled
  - Web scraping
  - Unclear what data is even needed

- Data must be converted into tables
  - CSV, JSON, XML, etc.
  - Images, Excel files, MATLAB, etc.
  - Text data in documents and webpages

Data Integration

tracks

id	name	album_id	media_type_id	genre_id	composer	milliseconds	bytes	unit_price
1	For Those Ab	1	1	1	Angus Young	343719	11170334	0.99
2	Balls to the V	2	2	1		342562	5510424	0.99
3	Fast As a Sha	3	2	1	F. Baltes, S. K	230619	3990994	0.99
4	Restless and	3	2	1	F. Baltes, R.A	252051	4331779	0.99
5	Princess of th	3	2	1	Deaffy & R.A.	375418	6290521	0.99
6	Put The Finge	1	1	1	Angus Young	205662	6713451	0.99
7	Let's Get It U	1	1	1	Angus Young	233926	7636561	0.99
8	Inject The Ve	1	1	1	Angus Young	210834	6852860	0.99
9	Snowballed	1	1	1	Angus Young	203102	6599424	0.99
10	Evil Walks	1	1	1	Angus Young	263497	8611245	0.99
11	C.O.D.	1	1	1	Angus Young	199836	6566314	0.99
12	Breaking The	1	1	1	Angus Young	263288	8596840	0.99
13	Night Of The	1	1	1	Angus Young	205688	6706347	0.99
14	Spellbound	1	1	1	Angus Young	270863	8817038	0.99

albums

id	title	artist_id
1	For Those About To Ro	1
2	Balls to the Wall	2
3	Restless and Wild	2
4	Let There Be Rock	1
5	Big Ones	3
6	Jagged Little Pill	4
7	Facelift	5
9	Plays Metallica By Fou	7
10	Audioslave	8
11	Out Of Exile	8
12	BackBeat Soundtrack	9
13	The Best Of Billy Cobha	10
14	Alcohol Fueled Brewta	11

artists

id	name
1	AC/DC
2	Accept
3	Aerosmith
4	Alanis Morissette
5	Alice In Chains
7	Apocalyptica
8	Audioslave
9	BackBeat
10	Billy Cobham
11	Black Label Society
12	Black Sabbath
13	Body Count

Image: https://dataschool.com/learn-sql/joins/

Data Integration Challenges

- **Merged table may be too large for memory**
  - Incrementally load and join data, using SGD or mini-batches
  - Use online learning techniques
- **Encoding issues**
  - Inconsistent data formats or terminology
  - Key aspects mentioned in cell comments or auxiliary files
- **Record linking problem**
  - Inconsistent column values

Record Linking Strategies

- String similarity above a threshold
  - Edit distance (“J Smith” → “Jon Smithee” with 4 edits)
  - String overlap (n-grams)

- Can tokenize and compare tokens, not just strings

- Can consider multiple fields (e.g., name, address)

Ins ID	Name
203342	J Smith
123452	Mao Y

Student ID	Name
3432432	Jon Smithee
9734783	Jane Smyth
8273737	Ying Mao

Encoding Features

- **Column types**
  - **Categorical:** Unordered finite set
  - **Ordinal:** Finite set with order
  - **Numerical:** Number (**warning:** numbers are not always numerical, e.g., ID)

Categorical **Numerical** **Ordinal**

MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	...	MoSold	YrSold	SaleType	SaleCondition	SalePrice
20	RL	80.0	10400	Pave	NaN	Reg	...	5	2008	WD	Normal	174000
180	RM	35.0	3675	Pave	NaN	Reg	...	5	2006	WD	Normal	145000
60	FV	72.0	8640	Pave	NaN	Reg	...	6	2010	Con	Normal	215200
20	RL	84.0	11670	Pave	NaN	IR1	...	3	2007	WD	Normal	320000
60	RL	43.0	10667	Pave	NaN	IR2	...	4	2009	ConLw	Normal	212000
80	RL	82.0	9020	Pave	NaN	Reg	...	6	2008	WD	Normal	168500

Data from: De Cock. Journal of Statistics Education 19(3), 2011

Encoding Features

- **Encoding categorical features**
  - Encode as one-hot vector
  - **Example:** Expand $X_j \in \{1,2,3\}$ into $[1, 0, 0]$ or $[0, 1, 0]$ or $[0, 0, 1]$

- **Encoding ordinal features**
  - Convert to a number, preserving the order
  - **Example:** [low, medium, high] → [1, 2, 3]
  - Encoding as categorical sometimes works better (try both!)

Missing Values

- **Basic solutions**
  - Delete features with mostly missing values
  - Delete instances with missing features

- **Imputation**
  - Fill missing features with mean (for numeric) or mode (for categorical)
  - Alternatively, predict missing values using supervised learning
  - Good practice to add binary feature indicating missingness for **each** feature that has missing values
  - **Example:** Medical history might be missing from a new patient

Outliers

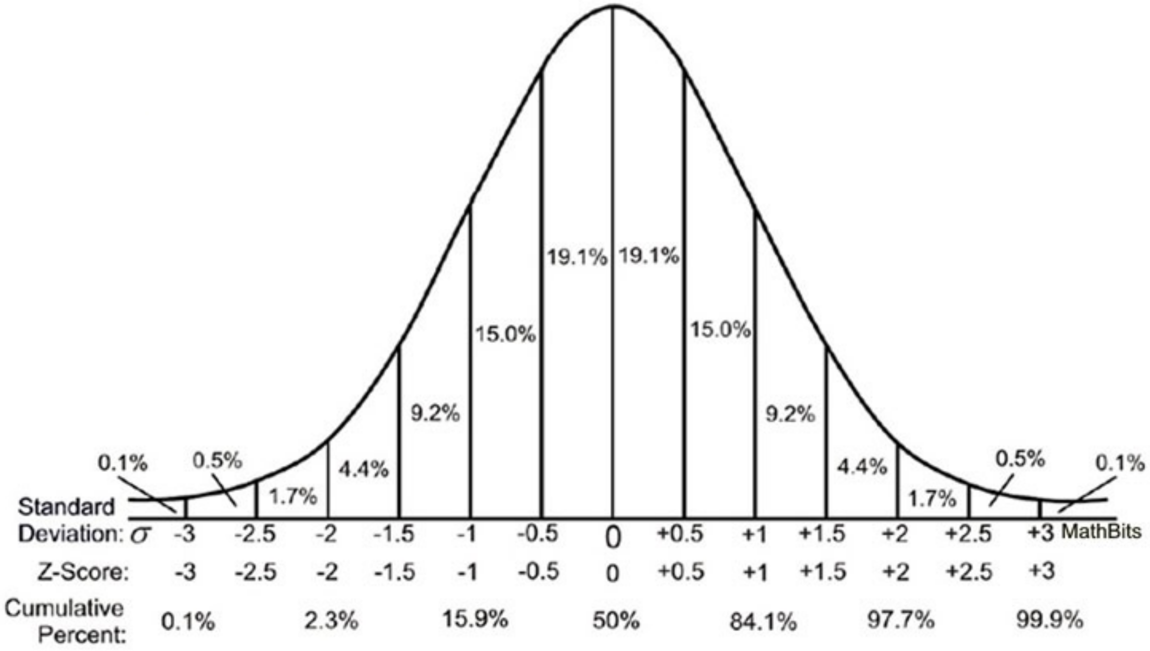
- **Causes**
  - Human error in data collection or data entry
  - Measurement/instrumentation errors
  - Experimental errors
  - Data merge errors (e.g., merging datasets with different scales)
  - Data preprocessing errors
  - Naturally from data generating process

Outliers

- Assume feature values are **Gaussian**

- **Removing outliers**
  - Discard points more than k standard deviations from mean
  - E.g., $k \in \{2.5, 3, 3.5\}$

- **Alternative:** Use loss that is robust to outliers (e.g., L_1 error)



https://mathbitsnotebook.com/Algebra2/Statistics/STzScores.html

Other Data Quality Issues

- **Incorrect feature values**
  - Typos (e.g., color = “bleu”, “gren”, “redd”)
  - Inconsistent spelling (e.g., “color”, “colour”)
  - Inconsistent abbreviations (e.g., “Oak St.”, “Oak Street”)
  - Garbage (e.g., color = “w┐r--śïį”)
  - **Potential solution:** Compare against a dictionary

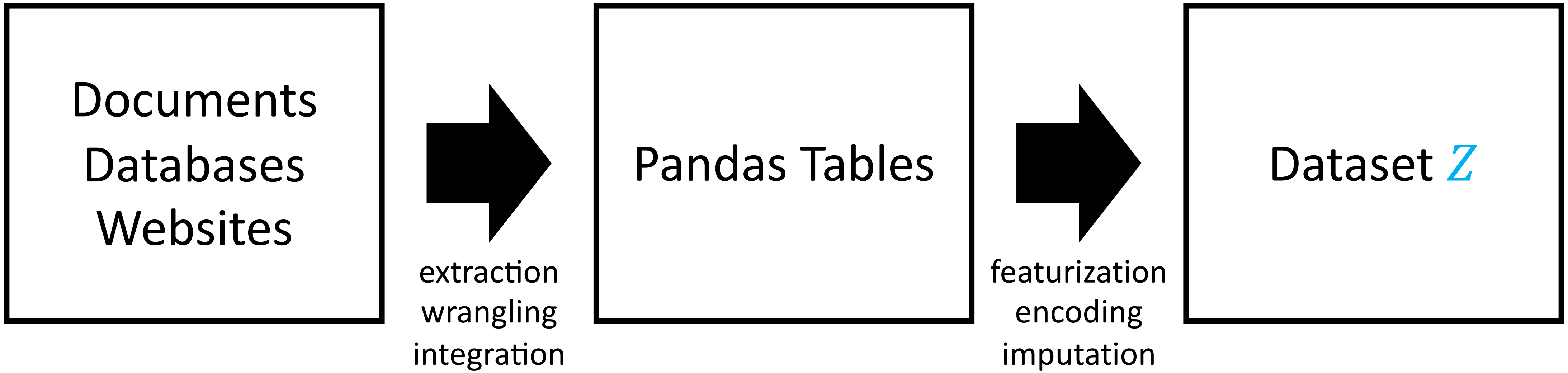
- **Missing instance labels**
  - Delete instances with missing labels
  - Can use semi-supervised learning techniques that leverage unlabeled data

Script Your Data Preprocessing!

- **Don’t manually edit**
  - No history of changes
  - Very easy to introduce mistakes
  - Hard to change earlier decisions

- **Write a script to load and preprocess data**
  - Documents all steps
  - Incremental debugging
  - Easy to make changes to earlier steps
  - Repeatable

Typical Data Engineering Pipeline



Understand Your Data!

- **Basic statistics**
  - Feature distribution
  - Feature-label correlations
  - Feature-feature correlations
  - “describe” function in pandas

- Data dictionary

- Can we do more?
  - Unsupervised learning!

Lecture 11: K-Means Clustering

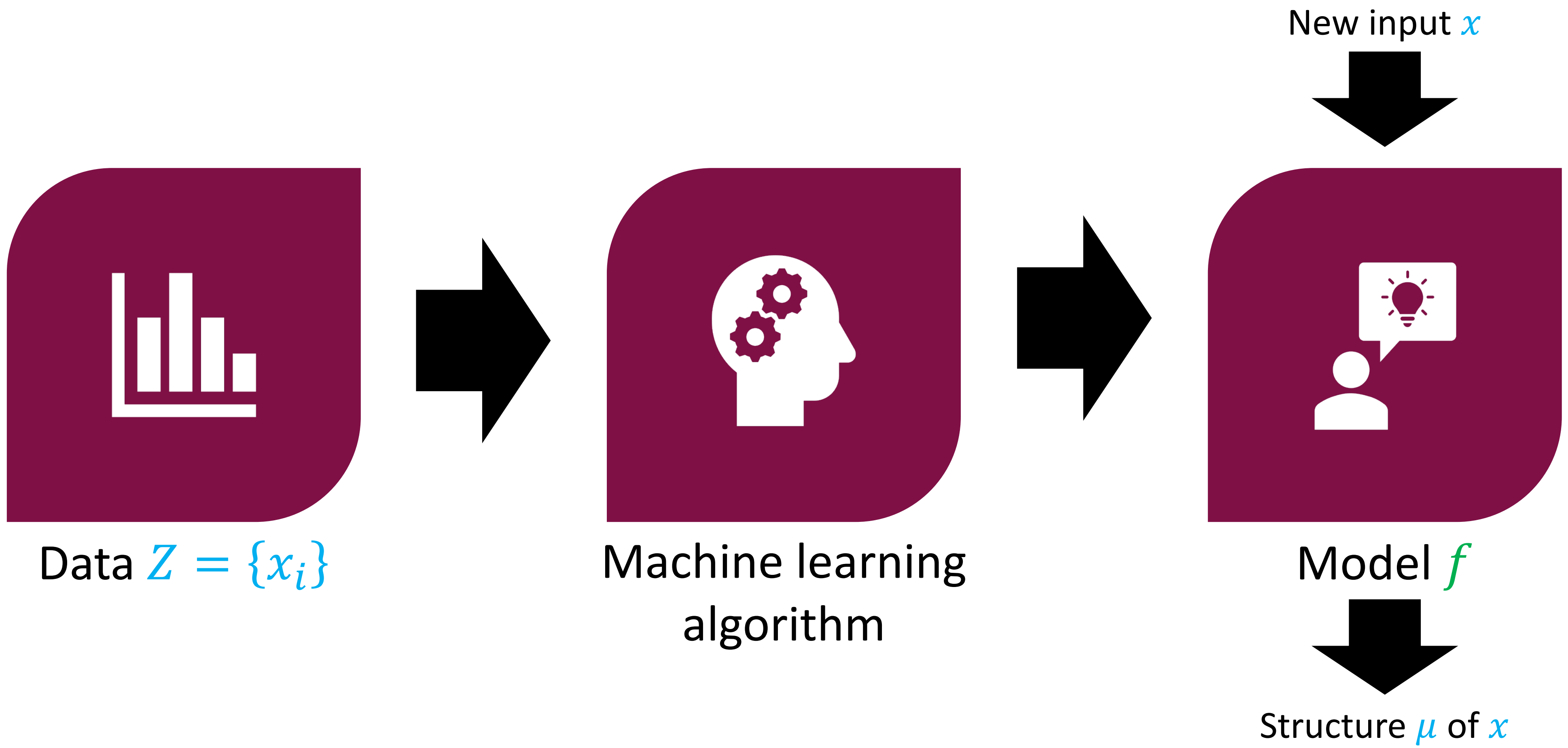
CIS 4190/5190

Fall 2023

Types of Learning

- **Supervised learning**
  - **Input:** Examples of inputs and desired outputs
  - **Output:** Model that predicts output given a new input
- **Unsupervised learning**
  - **Input:** Examples of some data (no "outputs")
  - **Output:** Representation of structure in the data
- **Reinforcement learning**
  - **Input:** Sequence of interactions with an environment
  - **Output:** Policy that performs a desired task

Unsupervised Learning



Applications of Unsupervised Learning

- **Visualization**
  - Exploring a dataset, or a machine learning model's outputs

- **Feature Learning**
  - Automatically construct lower-dimensional features
  - Especially useful with a lot of unlabeled data and just a few labeled examples

- **Compression (for storage)**
  - E.g., JPEG is adopting unsupervised machine learning approaches
  - https://jpeg.org/items/20190327_press.html

Applications of Unsupervised Learning

- “Based on our polling data, there are three main voting blocs, based on age, race, education level, income, political beliefs, and home-ownership. Features like marital status and # children are irrelevant.”

- “Our model says our company’s profits actually vary systematically based on the weather, is this actually the case?”

Applications of Unsupervised Learning

- Framing an ML problem (Mitchell's P, T, E)
- Data curation (sourcing, scraping, collection, labeling)
- Data analysis / visualization
- ML Design (hypothesis class, loss function, optimizer, hyperparameters, features)
- Train model
- Validate / Evaluate
- Deploy (and generate new data)
- Monitor performance on new data

Loss Minimization Framework

- **To design an unsupervised learning algorithm:**
  - **Model family:** Choose a model family $F = \{f_\beta\}_\beta$, where $\mu = f_\beta(x)$ encodes the structure of x
  - **Loss function:** Choose a loss function $L(\beta; Z)$

- **Resulting algorithm:**

$$\hat{\beta}(Z) = \arg\min_\beta L(\beta; Z)$$

Types of Unsupervised Learning

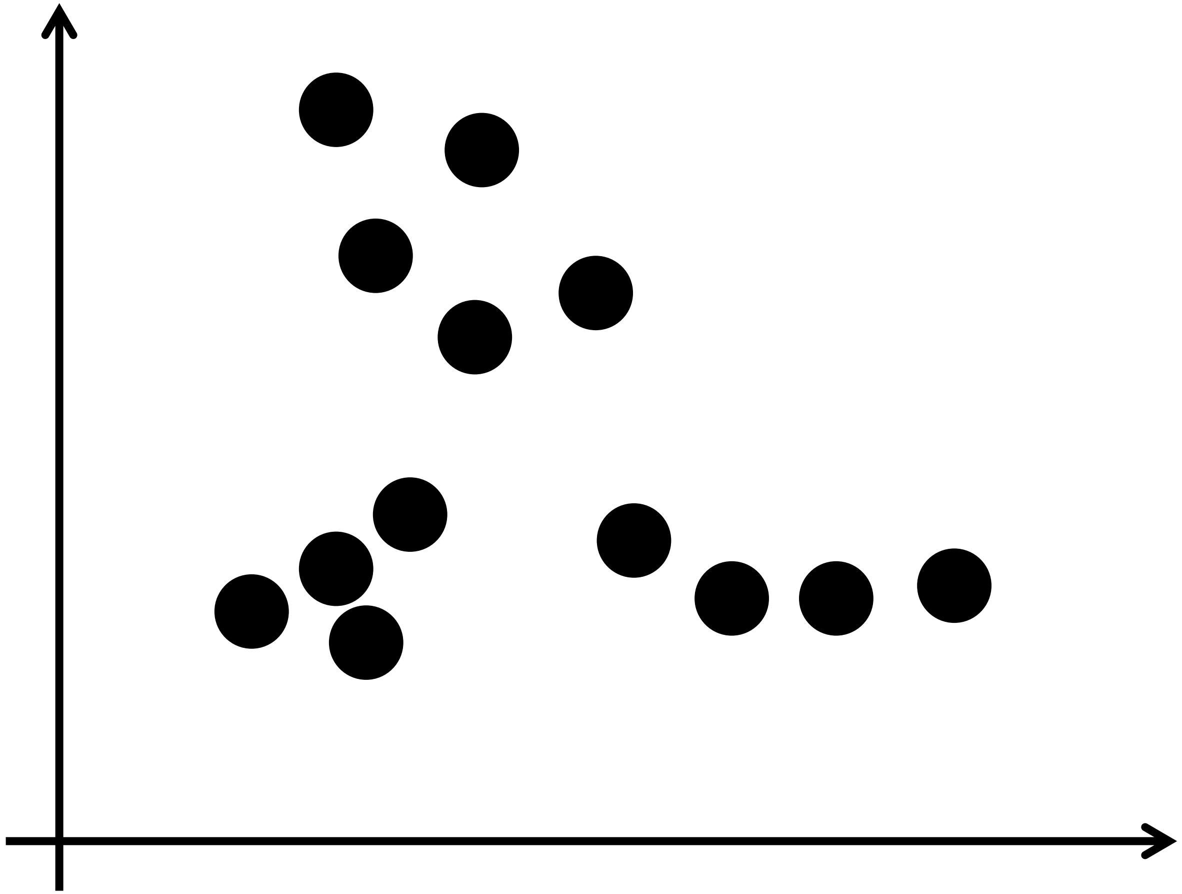
- **Clustering**
  - Map samples $x \in \mathbb{R}^d$ to $f(x) \in \mathbb{N}$
  - Each $k \in \mathbb{N}$ is associated with a representative example $x_k \in \mathbb{R}^d$
  - **Examples:** K-means clustering, greedy hierarchical clustering

- **Dimensionality reduction**
  - Map samples $x \in \mathbb{R}^d$ to $f(x) \in \mathbb{R}^{d'}$, where $d' \ll d$
  - **Example:** Principal components analysis (PCA)
  - Modern deep learning is based on this idea

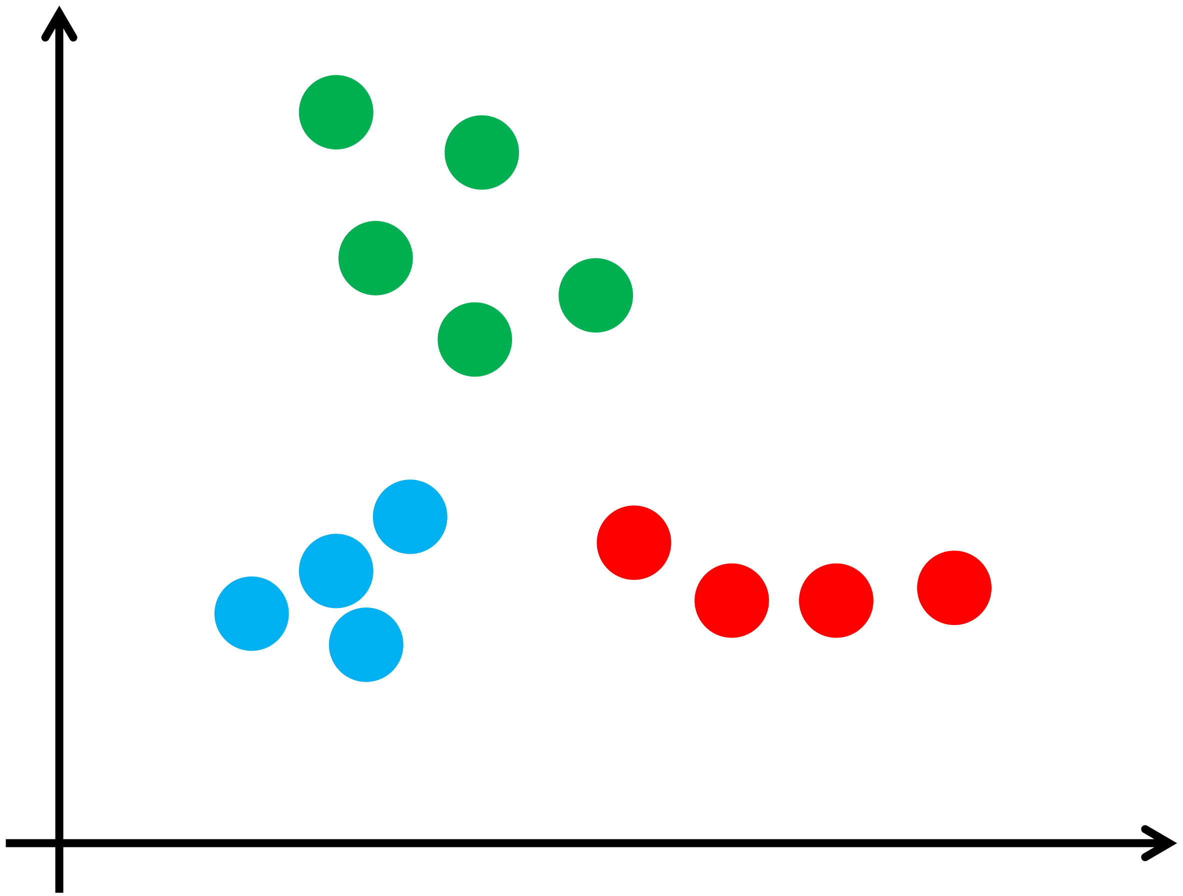
The Clustering Problem

- **Input:** Dataset $Z = \{x_i\}_{i=1}^{n}$
- **Output:** Model $f(x) \in \{1, \ldots, K\}$
  - **Intuition:** Predictions should encode "natural" clusters in the data
  - Here, $K \in \mathbb{N}$ is a hyperparameter

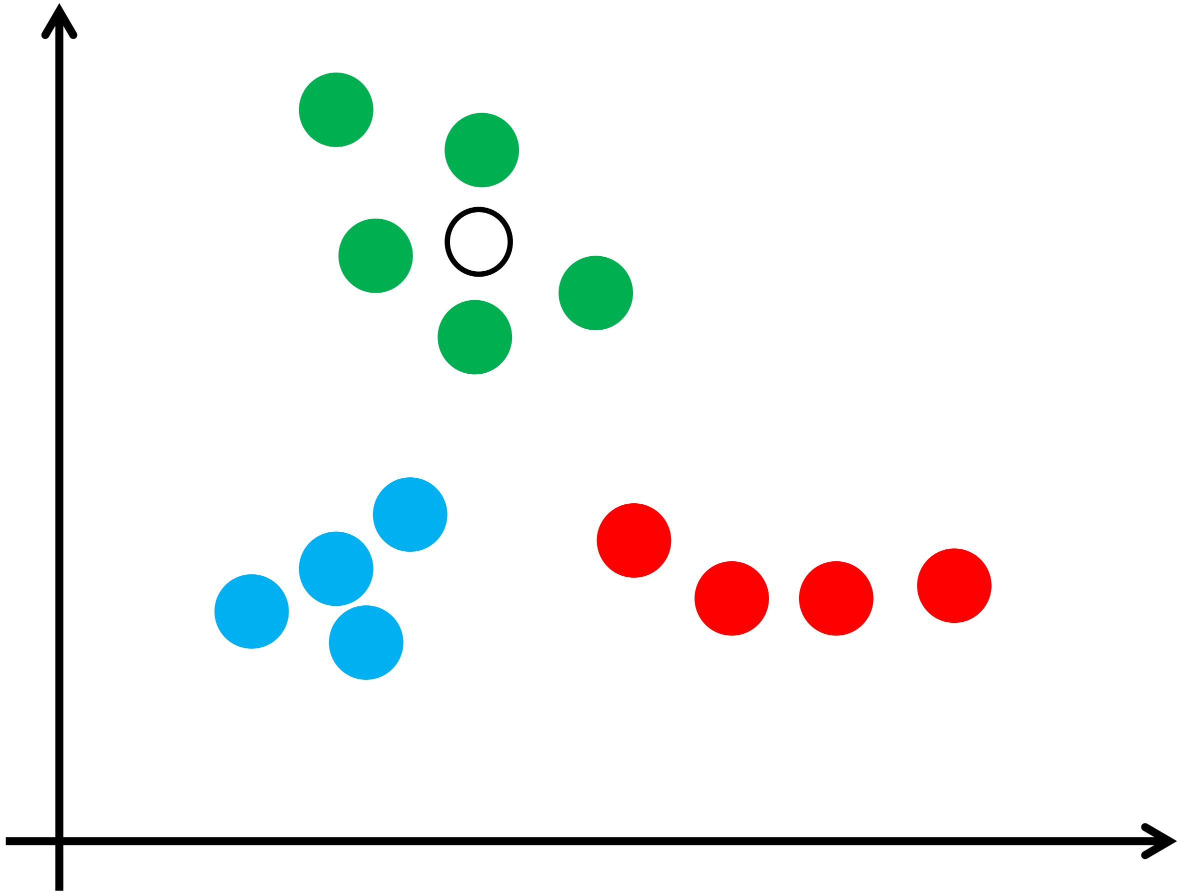
The Clustering Problem



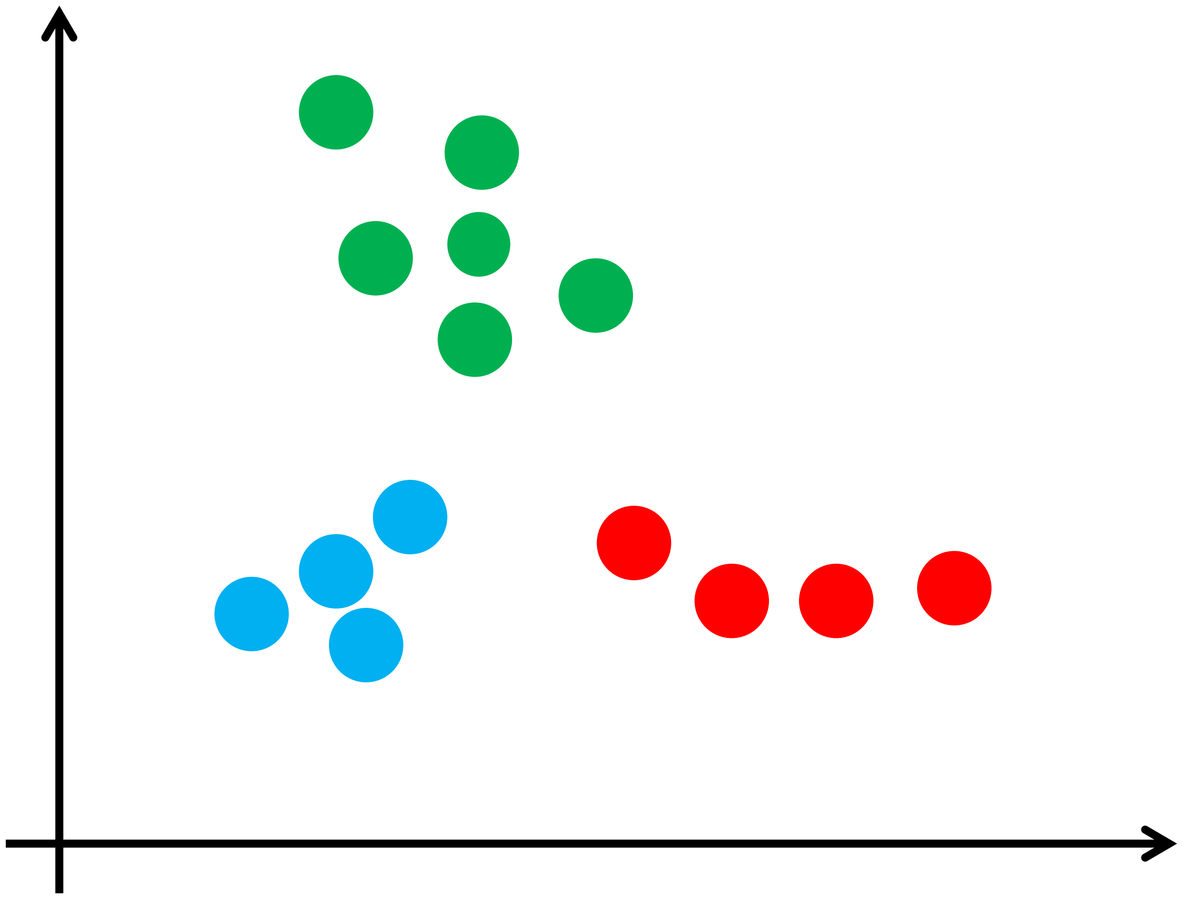
The Clustering Problem



The Clustering Problem



The Clustering Problem

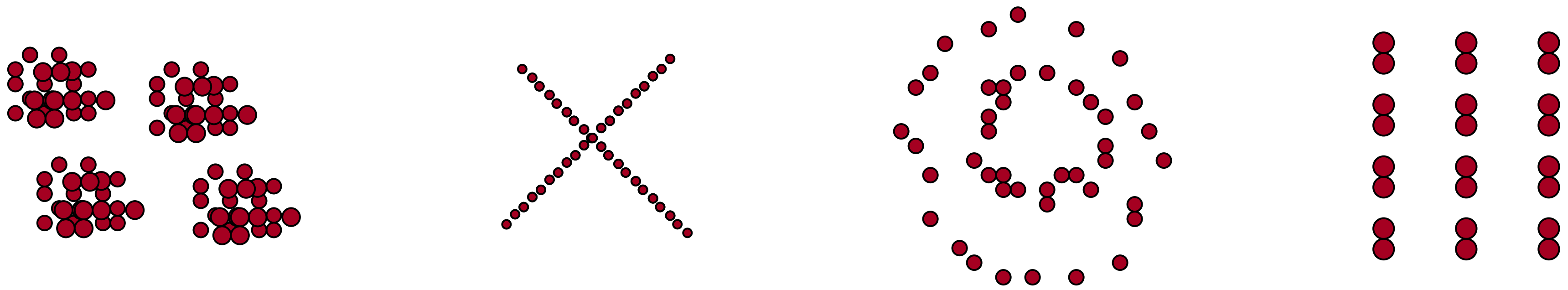


The Clustering Problem

- **Input:** Dataset $Z = \{x_i\}_{i=1}^{n}$
- **Output:** Model $f(x) \in \{1, \dots, K\}$
  - **Intuition:** Predictions should encode "natural" clusters in the data
  - Here, $K \in \mathbb{N}$ is a hyperparameter
- How to formalize "naturalness"?
  - Using a loss function!

Clustering Loss

- Loss depends on the structure of the data we are trying to capture



- K-Means clustering aims to minimize specific loss over a specific model family

K-Means Clustering Model Family

- **Parameters:** Set of **centroids** μ_j (for $j \in \{1, \dots, K\}$)
  - One for each cluster (K is a hyperparameter)
  - **Intuition:** μ_j is the "center" of cluster j

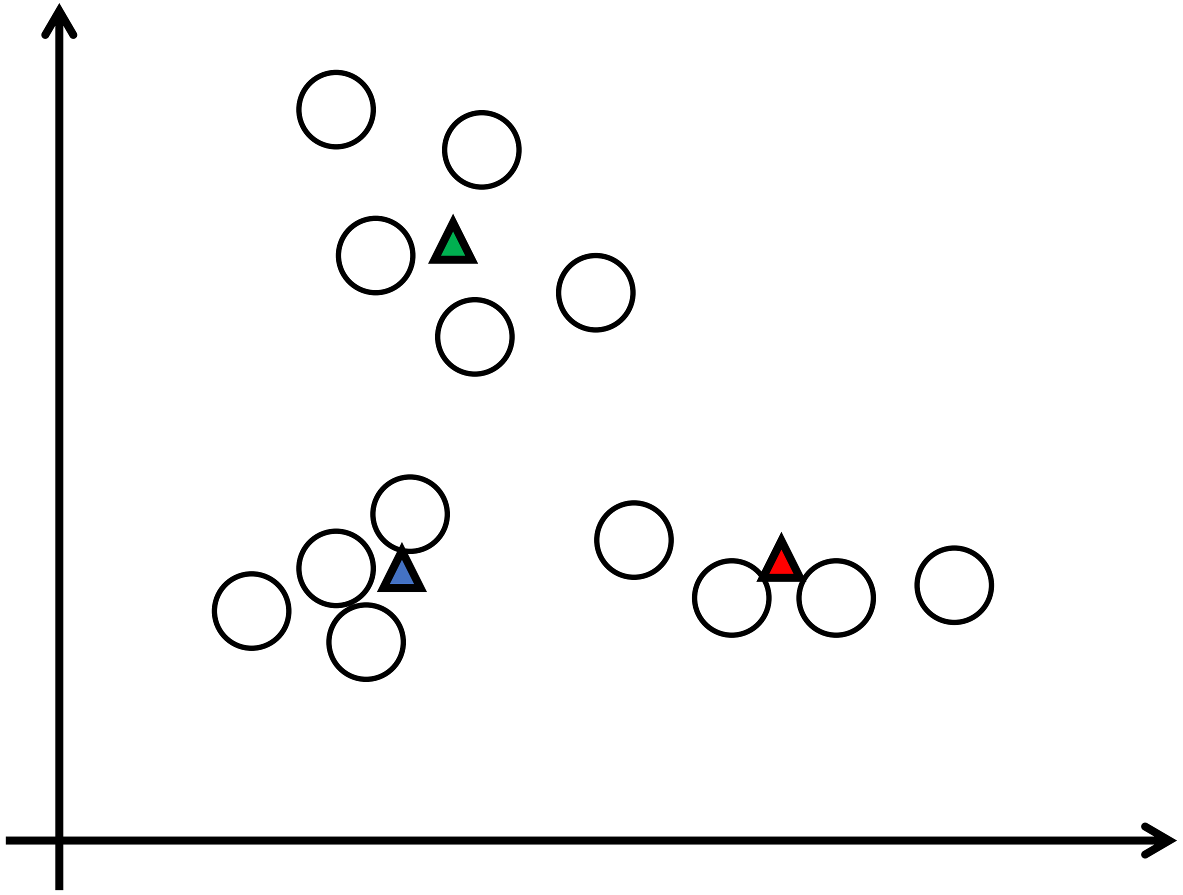
- Given a new example x, assign it to the nearest cluster:

$$f_\mu(x) = \arg\min_j \left\| x - \mu_j \right\|_2^2$$

- Can use other distance functions

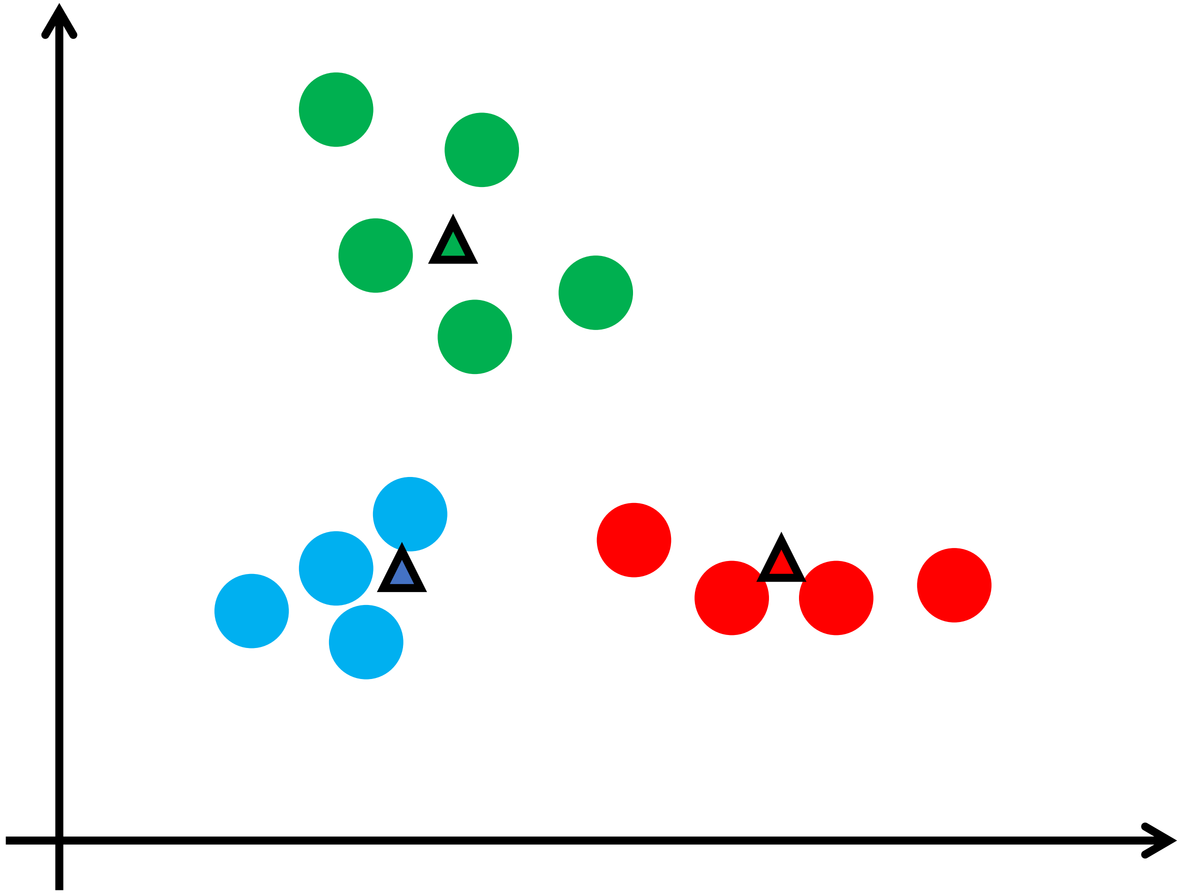
K-Means Clustering Loss

- Compute MSE of each point in the training data to its centroid



K-Means Clustering Loss

- Compute MSE of each point in the training data to its centroid



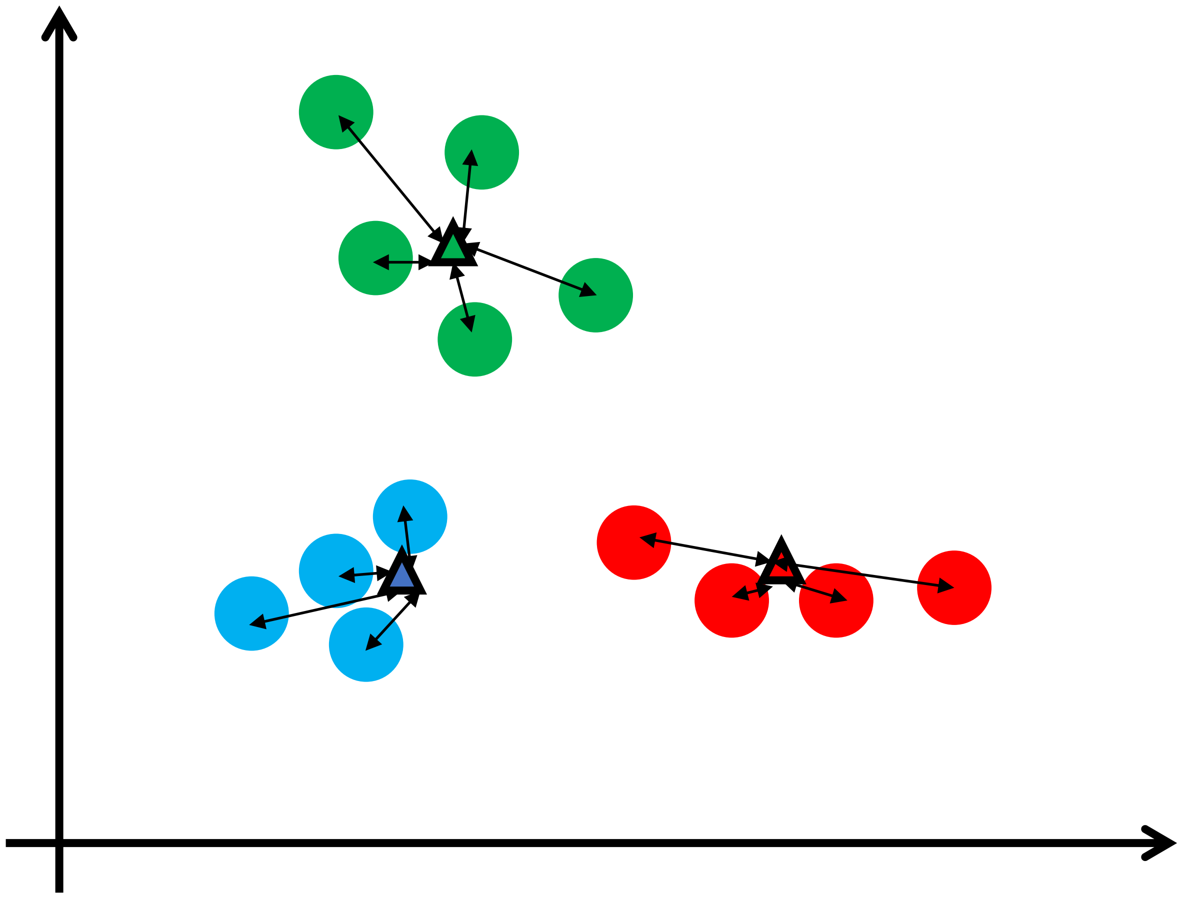
K-Means Clustering Loss

- K-means clustering chooses centroids that minimize loss of training examples Z

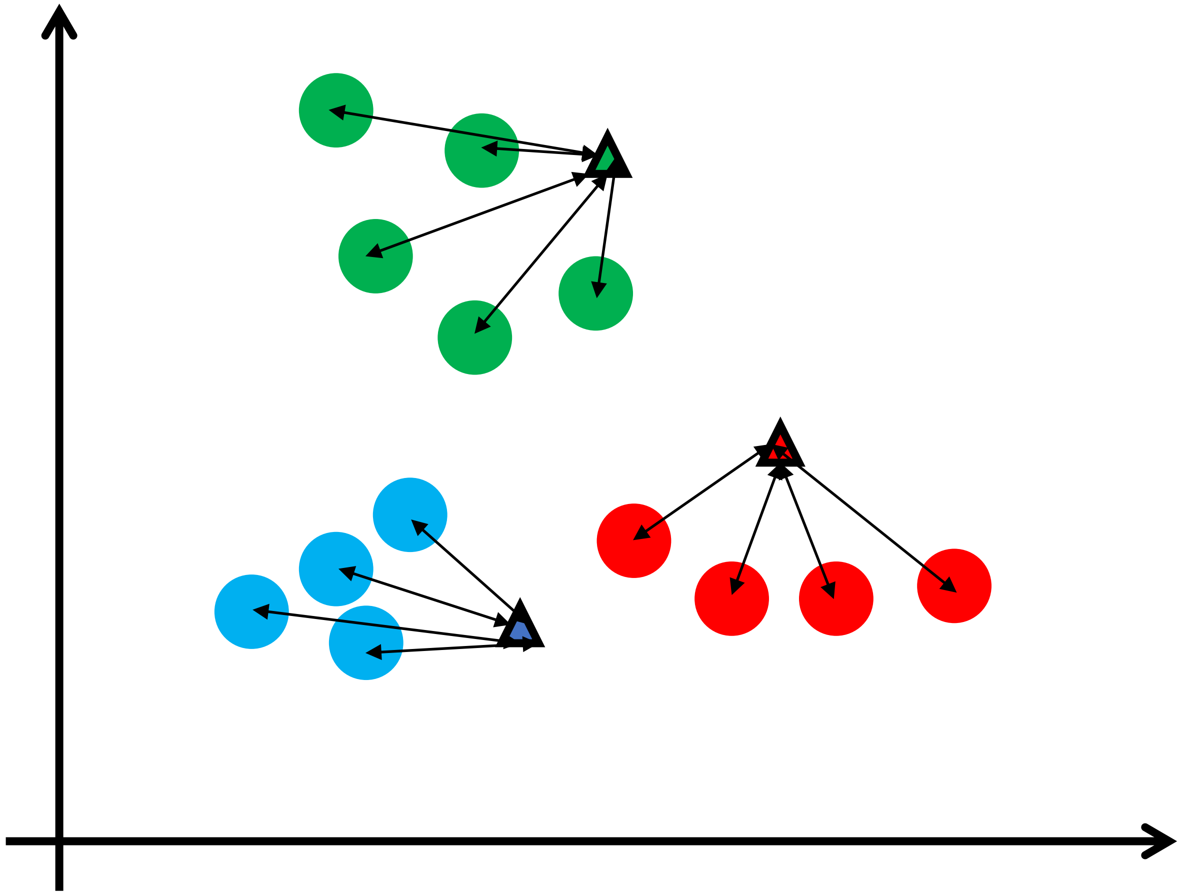
- Compute MSE of each point in the training data to its centroid:

$$L(\mu; Z) = \sum_{i=1}^{n} \left\| x_i - \mu_{f_\mu(x_i)} \right\|_2^2$$

K-Means Clustering Loss



K-Means Clustering Loss



K-Means Clustering Optimization

- Minimizing the loss exactly is hard due to local minima
- Use an “alternating minimization” heuristic
  - Works better than gradient descent in practice
  - Provably converges to local minimum

K-Means Clustering Algorithm

Kmeans(Z):

  for $j \in \{1, \ldots, k\}$:

    $\mu_{1,j} \leftarrow \text{Random}(Z)$

  for $t \in \{1,2, \ldots\}$:

    for $i \in \{1, \ldots, n\}$:

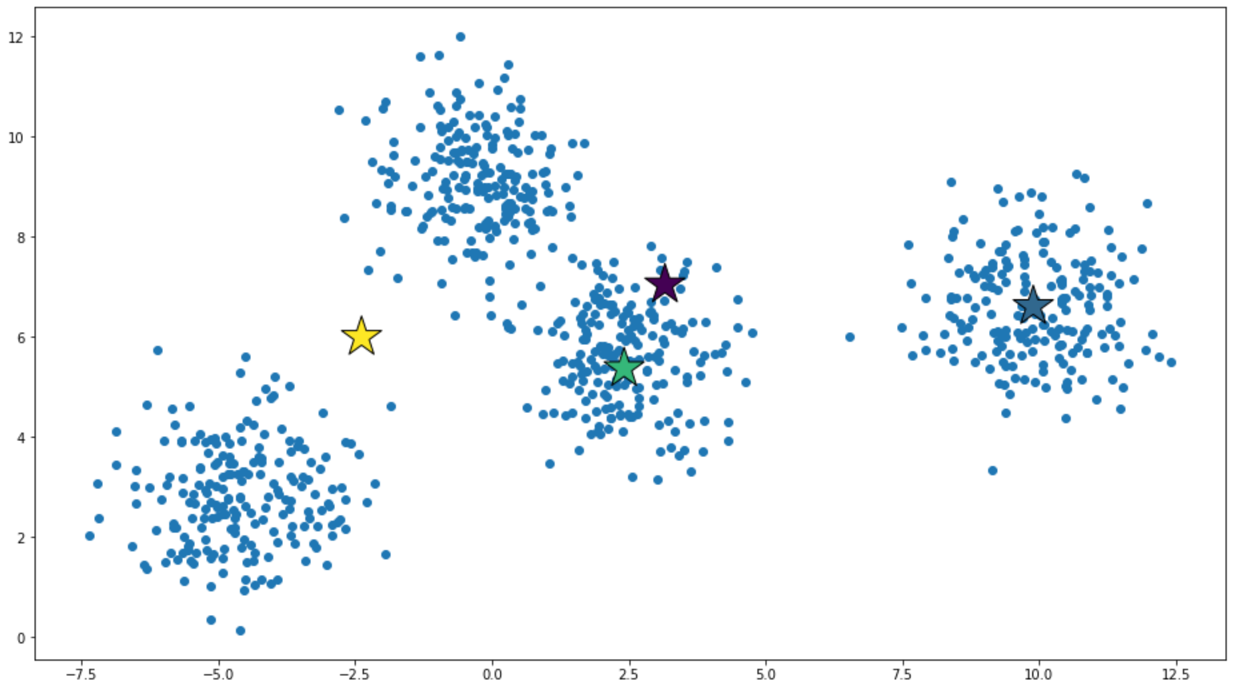
      $j_{t,i} \leftarrow f_{\mu_t}(x_i)$

    for $j \in \{1, \ldots, k\}$:

      $\mu_{t,j} \leftarrow \text{mean}(\{\, x_i \mid j_{t,i} = j \,\})$

    if $\mu_t = \mu_{t-1}$:

      return μ_t



K-Means Clustering Algorithm

Kmeans(Z):

$\boxed{\begin{array}{l}\textbf{for } j \in \{1, \ldots, k\}: \\ \quad \mu_{1,j} \leftarrow \text{Random}(Z)\end{array}}$

for $t \in \{1,2, \ldots\}$:

for $i \in \{1, \ldots, n\}$:

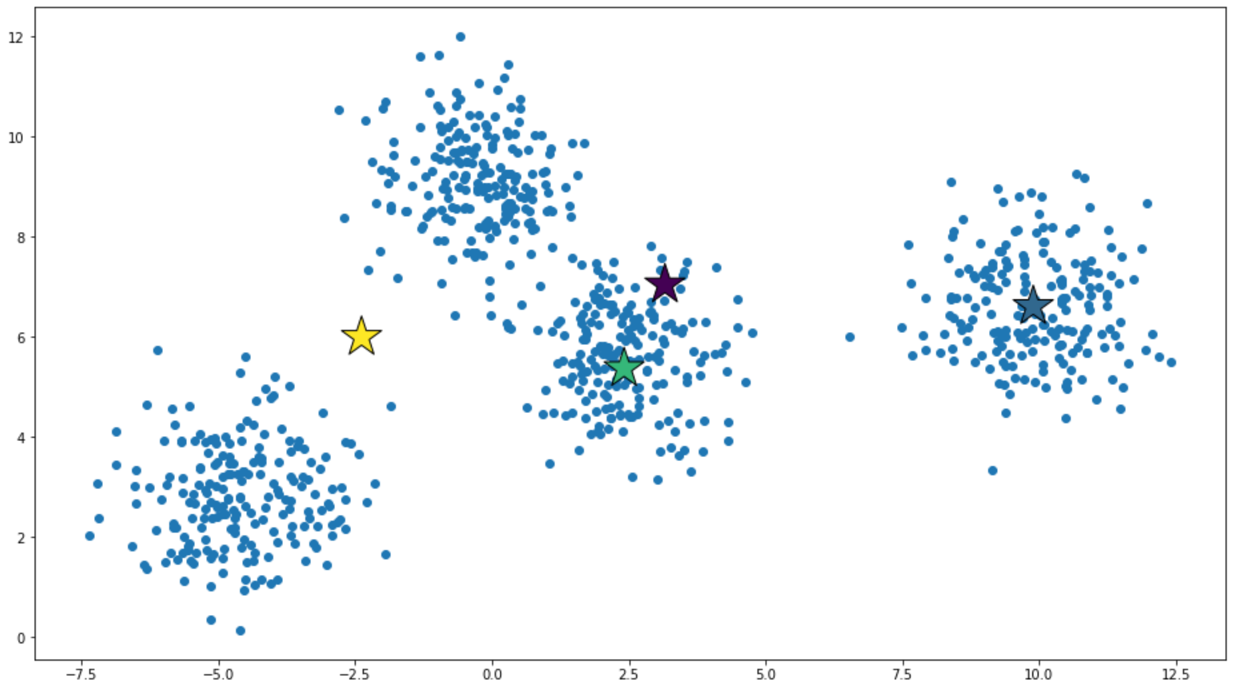
$j_{t,i} \leftarrow f_{\mu_t}(x_i)$

for $j \in \{1, \ldots, k\}$:

$\mu_{t,j} \leftarrow \text{mean}(\{\, x_i \mid j_{t,i} = j \,\})$

if $\mu_t = \mu_{t-1}$:

return μ_t



K-Means Clustering Algorithm

Kmeans(Z):

for $j \in \{1, \dots, k\}$:

$\mu_{1,j} \leftarrow \text{Random}(Z)$

for $t \in \{1,2, \dots\}$:

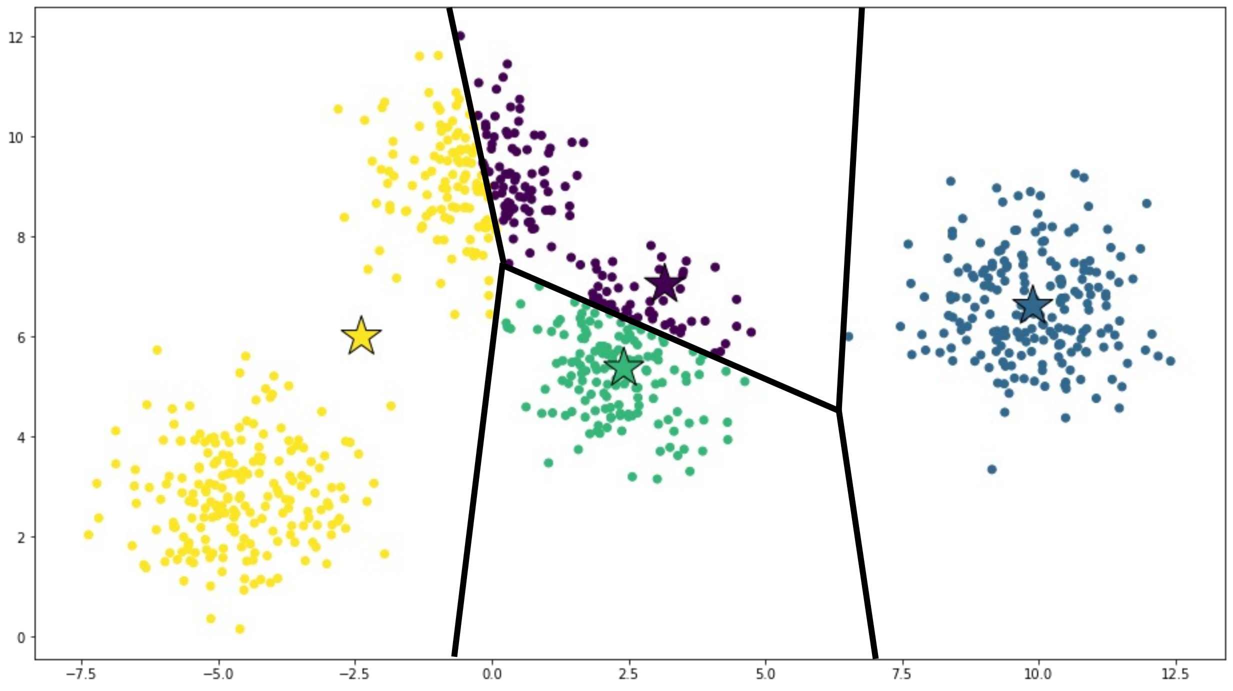
$\boxed{\begin{array}{l} \textbf{for } i \in \{1, \dots, n\}: \\ \quad j_{t,i} \leftarrow f_{\mu_t}(x_i) \end{array}}$

for $j \in \{1, \dots, k\}$:

$\mu_{t,j} \leftarrow \text{mean}(\{\, x_i \mid j_{t,i} = j \,\})$

if $\mu_t = \mu_{t-1}$:

return μ_t



K-Means Clustering Algorithm

Kmeans(Z):

for $j \in \{1, \ldots, k\}$:

$\mu_{1,j} \leftarrow \text{Random}(Z)$

for $t \in \{1,2, \ldots\}$:

for $i \in \{1, \ldots, n\}$:

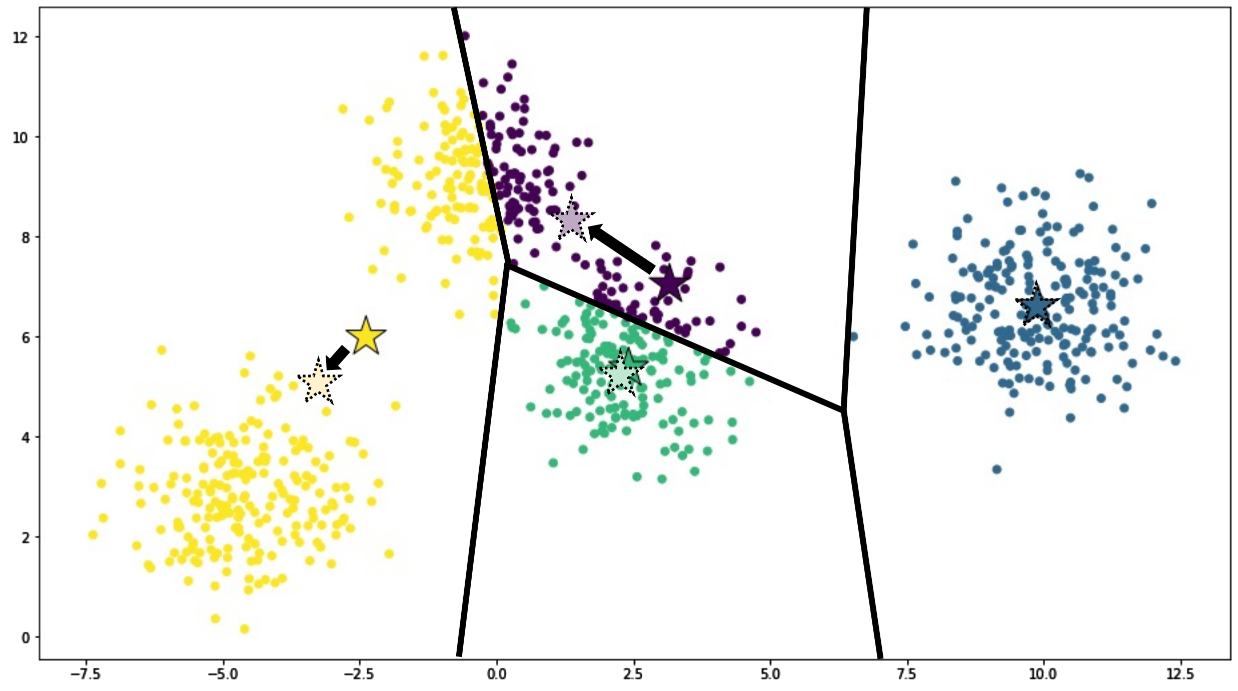
$j_{t,i} \leftarrow f_{\mu_t}(x_i)$

for $j \in \{1, \ldots, k\}$:

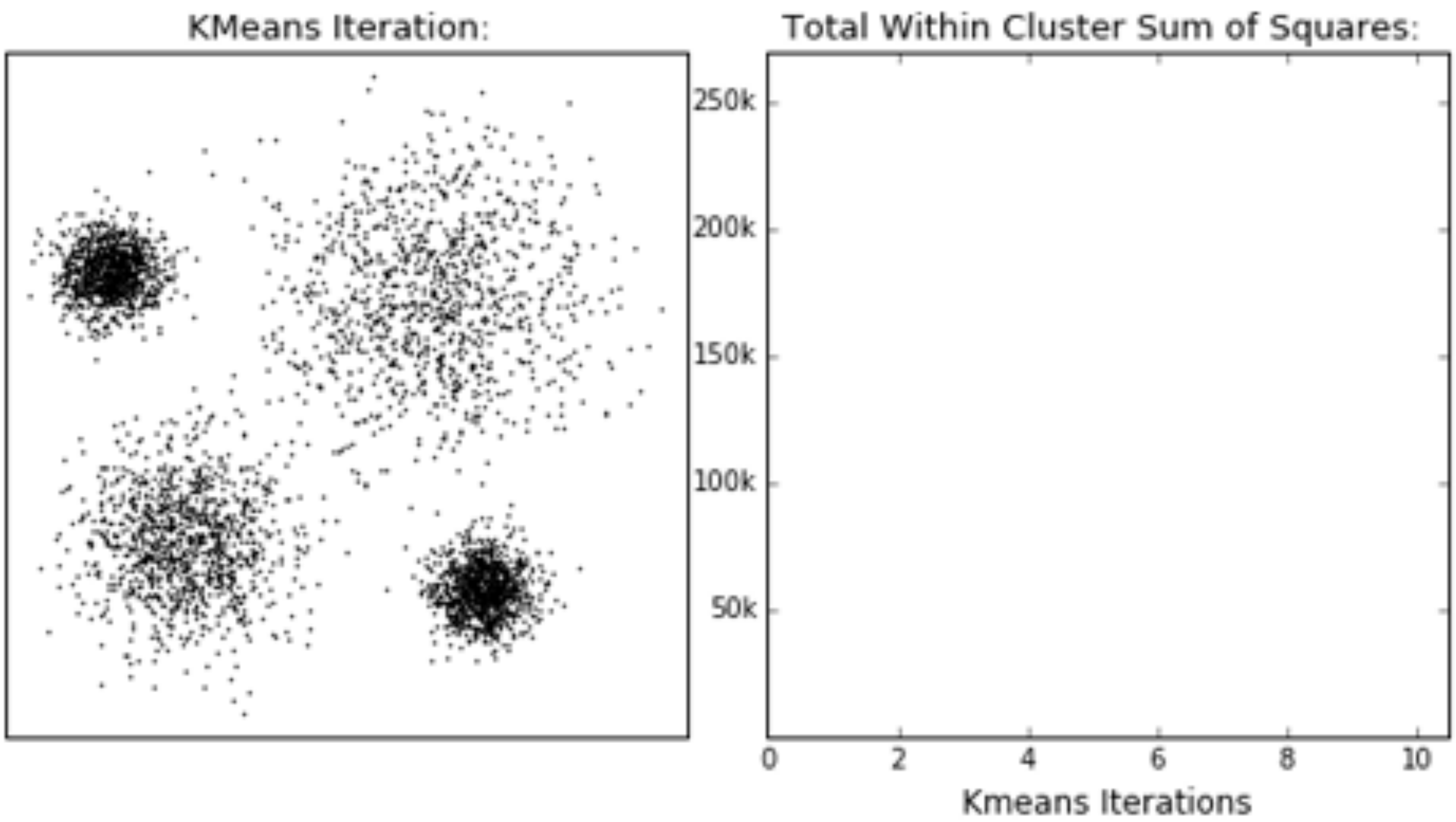
$\mu_{t,j} \leftarrow \text{mean}(\{\, x_i \mid j_{t,i} = j \,\})$

if $\mu_t = \mu_{t-1}$:

return μ_t



K-Means Clustering Algorithm



https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

Random Initialization

- Sensitive to initialization

- One strategy is to run multiple times with different random centroids and choose the model with lowest MSE

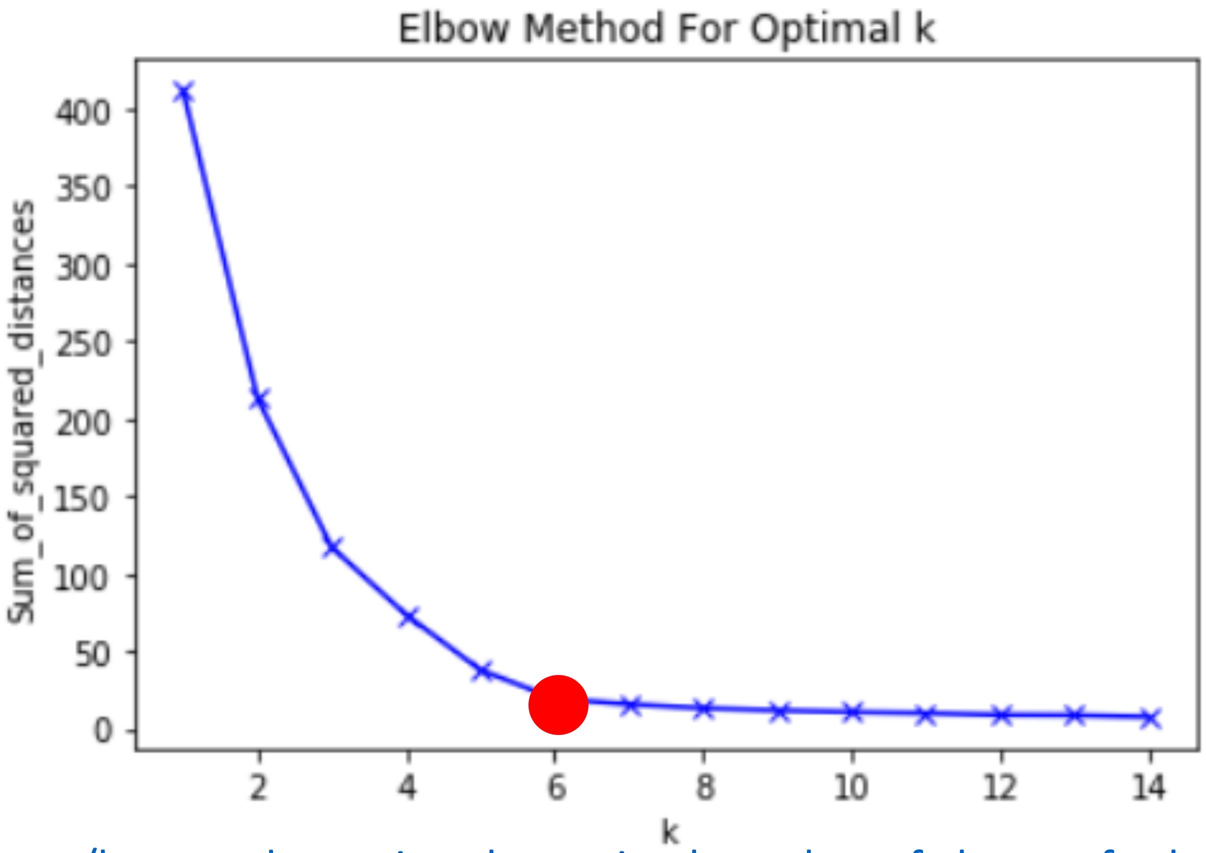
- **Alternative:** K-means++
  - Randomly initialize first centroid to some $x \in Z$
  - Subsequently, choose centroid randomly according to $p(x) \propto d_x^2$, where d_x is the distance to the nearest centroid so far
  - Upweights points that are farther from existing centroids

Number of Clusters

- As K becomes large
  - MSE becomes small
  - Many clusters → might be less useful

- Choice of K is subjective

Number of Clusters



https://blog.cambridgespark.com/how-to-determine-the-optimal-number-of-clusters-for-k-means-clustering-14f27070048f

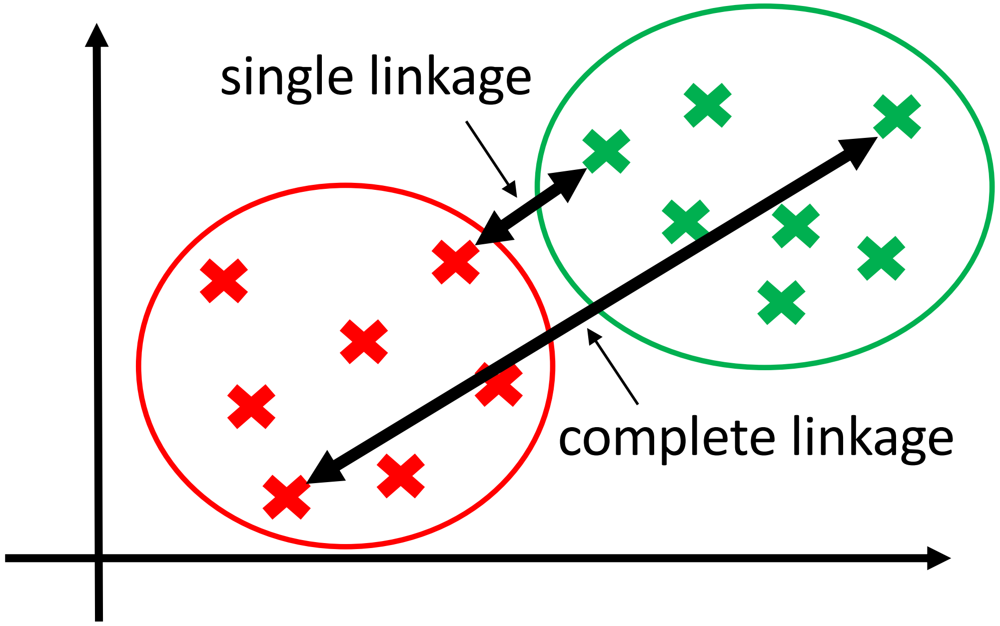
Hierarchical Clustering

- Alternative approach to clustering that makes local changes
- **Agglomerative clustering**
  - Initialize each example to its own cluster
  - Iteratively agglomerate “closest” clusters
- **Divisive clustering**
  - Initialize all examples in a single cluster
  - Iteratively divide “most distant” sub-clusters
- Incremental nature results in hierarchical clusters

Selecting Clusters

- **Single linkage**
  - Compute distances between most similar members of pair of clusters
  - Merge pair of clusters with smallest minimum distance

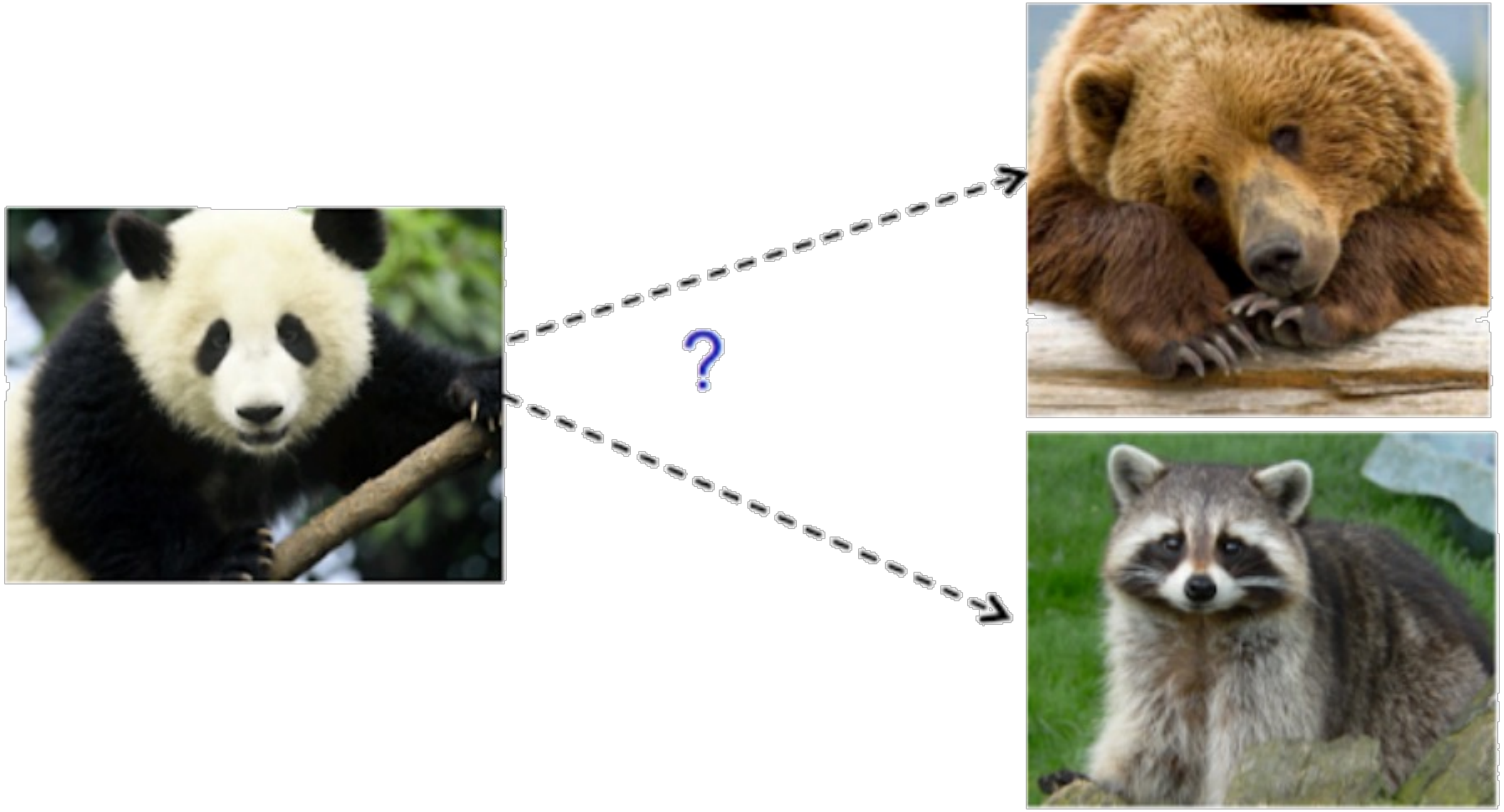
- **Complete linkage**
  - Compute distances between most distant members of pair of clusters
  - Merge pair of clusters with smallest maximum distance



Optimization Algorithm

- Computing pairwise distances is $O(n^2)$, which can be expensive
- **Solution**
  - Precompute pairwise distances d_{ij} between clusters i and j
  - Update d_{ij} with every merge/divide

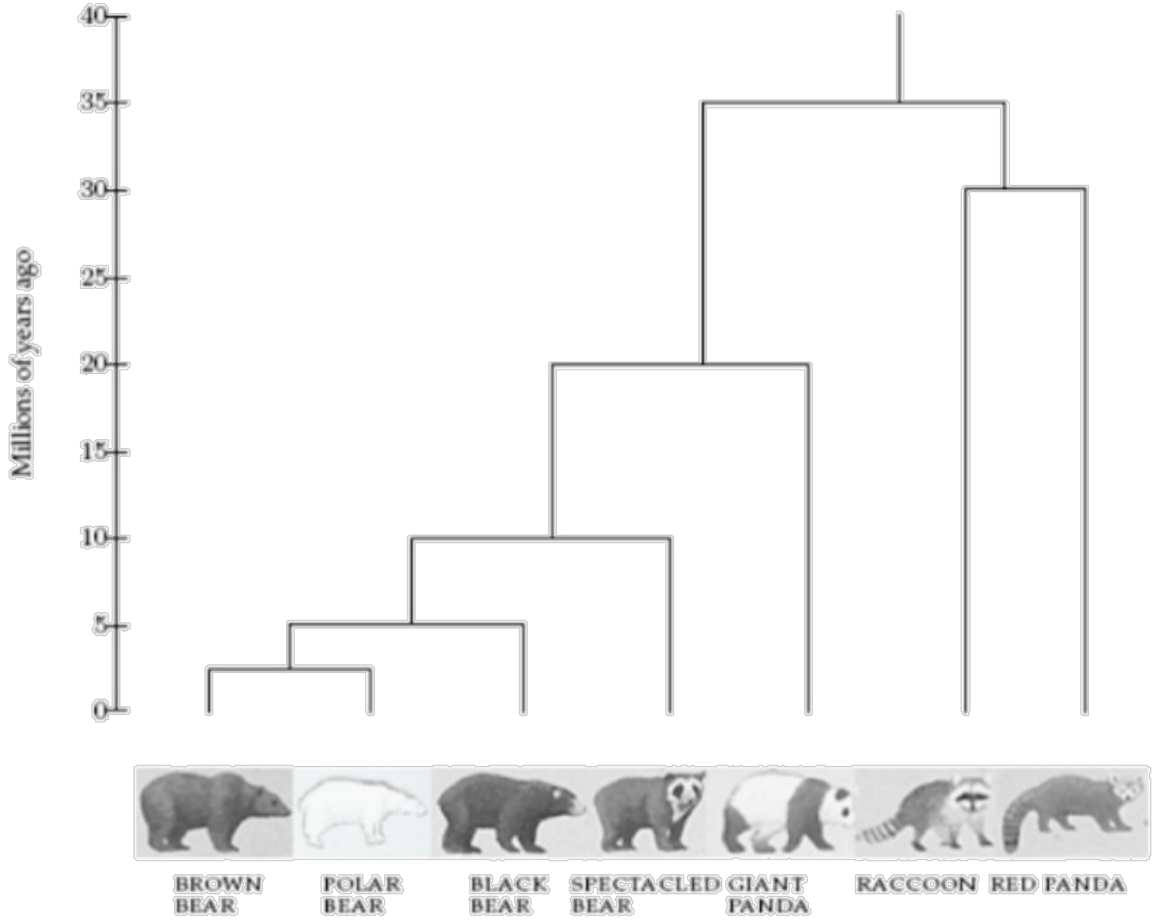
Example: Phylogenetic Trees



https://towardsdatascience.com/hierarchical-clustering-and-its-applications-41c1ad4441a6

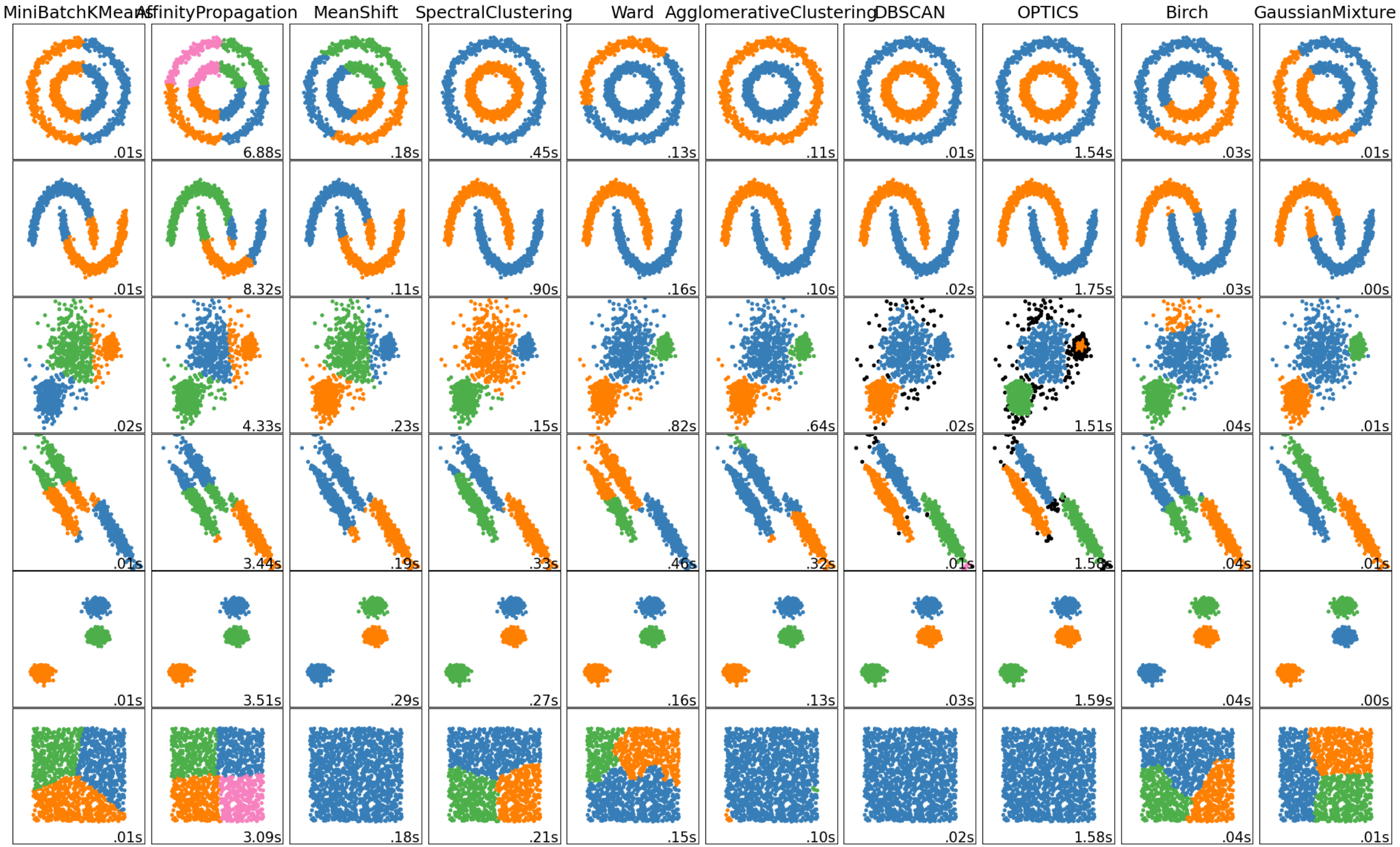
Example: Phylogenetic Trees

- **Features:** Gene sequences
- **Distance:** Edit distance
- Use agglomerative clustering to compute hierarchical clusters, which form phylogenetic trees



https://towardsdatascience.com/hierarchical-clustering-and-its-applications-41c1ad4441a6

Many Clustering Algorithms



https://scikit-learn.org/stable/modules/clustering.html#clustering