Clustering

Instructor: Jing Wang
Department of ISOM
Spring 2023

HKUST BUSINESS SCHOOL
香港科大商學院

Recap: Supervised vs. Unsupervised Learning

- **Supervised learning (prediction)**: learns a model that predicts target outcome based on a set of other attributes (i.e., training data where target value is known).
  - Stock price prediction (numerical target variable)
  - Credit card default (binary target variable)
- **Unsupervised learning (relationship mining)**: finds relationships in the data without reference to target variable.
  - Beer and diaper

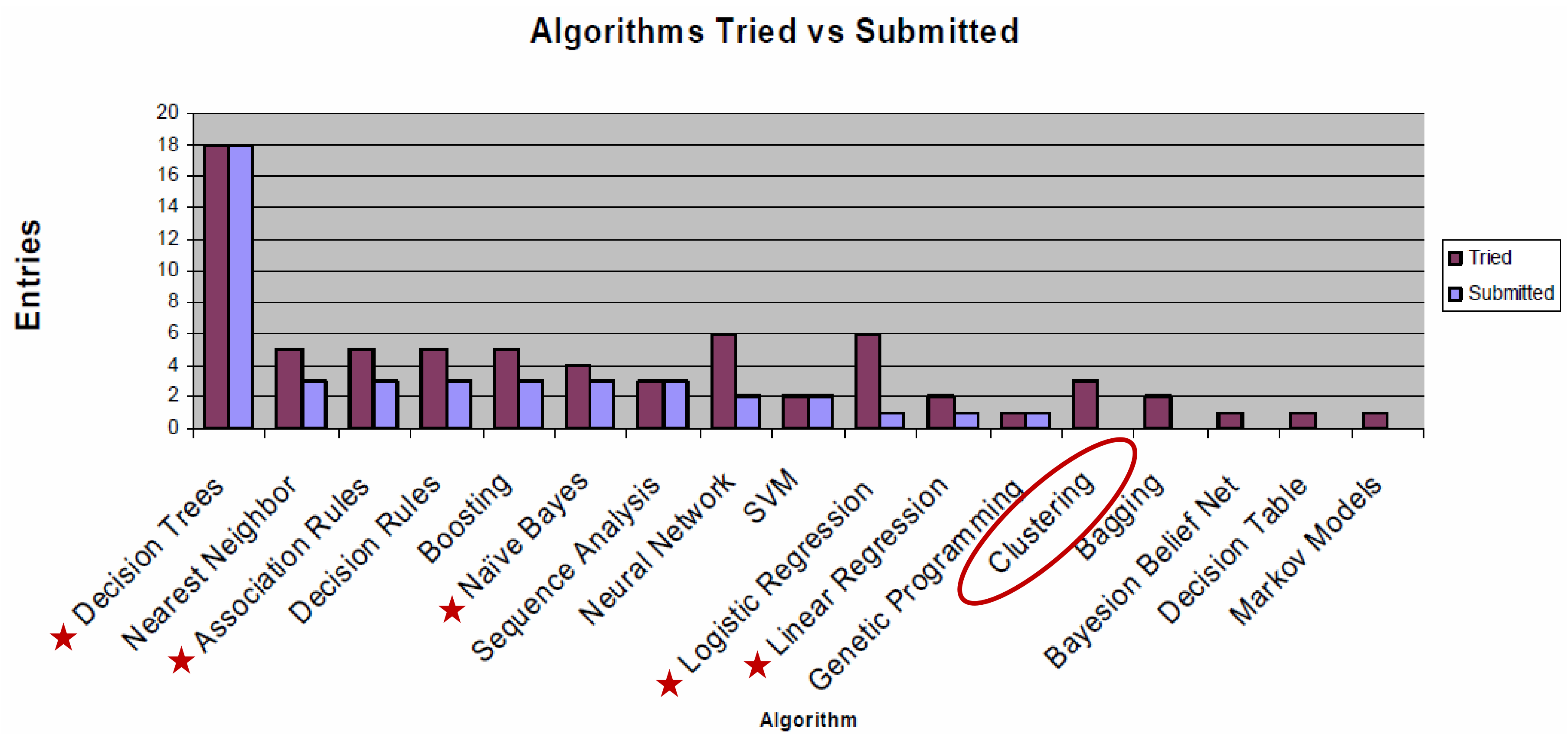
Key: is there a target that we are trying to predict?

Unsupervised DM

- Q: How do I find attributes that occur together more than I might expect by chance?
- A: Associations (Which attributes occur together? Relationship between columns.)

- Q: How do I find natural groupings of data instances?
- A: Clustering (Which examples are similar? Grouping rows together)

Commonly Used Induction Algorithms



Clustering

- How do I find groupings of **"similar"** things?
  - Here, things are multidimensional objects as with most other DM techniques

Corporate data on 22 public utilities in the US

Company	*Fixed*	*RoR*	*Cost*	*Load*	*Demand*	*Sales*	*Nuclear*	*Fuel Cost*
Arizona Public Service	1.06	9.2	151	54.4	1.6	9,077	0	0.628
Boston Edison Co.	0.89	10.3	202	57.9	2.2	5,088	25.3	1.555
Central Louisiana Co.	1.43	15.4	113	53	3.4	9,212	0	1.058
Commonwealth Edison Co.	1.02	11.2	168	56	0.3	6,423	34.3	0.7
Consolidated Edison Co. (NY)	1.49	8.8	192	51.2	1	3,300	15.6	2.044
Florida Power & Light Co.	1.32	13.5	111	60	−2.2	11,127	22.5	1.241
Hawaiian Electric Co.	1.22	12.2	175	67.6	2.2	7,642	0	1.652
Idaho Power Co.	1.1	9.2	245	57	3.3	13,082	0	0.309
Kentucky Utilities Co.	1.34	13	168	60.4	7.2	8,406	0	0.862
Madison Gas & Electric Co.	1.12	12.4	197	53	2.7	6,455	39.2	0.623
Nevada Power Co.	0.75	7.5	173	51.5	6.5	17,441	0	0.768
New England Electric Co.	1.13	10.9	178	62	3.7	6,154	0	1.897
Northern States Power Co.	1.15	12.7	199	53.7	6.4	7,179	50.2	0.527
Oklahoma Gas & Electric Co.	1.09	12	96	49.8	1.4	9,673	0	0.588
Pacific Gas & Electric Co.	0.96	7.6	164	62.2	−0.1	6,468	0.9	1.4
Puget Sound Power & Light Co.	1.16	9.9	252	56	9.2	15,991	0	0.62
San Diego Gas & Electric Co.	0.76	6.4	136	61.9	9	5,714	8.3	1.92
The Southern Co.	1.05	12.6	150	56.7	2.7	10,140	0	1.108
Texas Utilities Co.	1.16	11.7	104	54	−2.1	13,507	0	0.636
Wisconsin Electric Power Co.	1.2	11.8	148	59.9	3.5	7,287	41.1	0.702
United Illuminating Co.	1.04	8.6	204	61	3.5	6,650	0	2.116
Virginia Electric & Power Co.	1.07	9.3	174	54.3	5.9	10,093	26.6	1.306

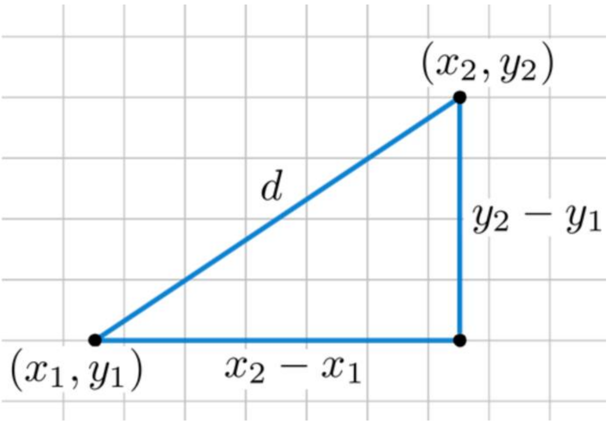
Fixed = fixed-charge covering ratio (income/debt); RoR = rate of return on capital; Cost = cost per kilowatt capacity in place; Load = annual load factor; Demand = peak kilowatthour demand growth from 1974 to 1975; Sales = sales (kilowatthour use per year); Nuclear = percent nuclear; Fuel Cost = total fuel costs (cents per kilowatthour).

Distance Measure: Euclidean Distance

- The Euclidean distance, d_{ij}, which between two records, i and j, with k attributes, is defined by

$$d_{ij} = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ik} - x_{jk})^2}$$

lower distance higher similarity



The Euclidean distance between Arizona Public Service and Boston Edison is:

$$d_{12} = \sqrt{(1.06 - 0.89)^2 + (9.2 - 10.3)^2 + (151 - 202)^2 + \cdots + (0.628 - 1.555)^2}$$
$$= 3989.408.$$

Euclidean Distance

- What's the distance of a data point to itself? 0
- Is Euclidean distance symmetric? Yes
- Is Euclidean distance sensitive to the scale of each attribute? Yes

Sales dominating distance computation

Company	*Fixed*	*RoR*	*Cost*	*Load*	*Demand*	*Sales*	*Nuclear*	*Fuel Cost*
Arizona Public Service	1.06	9.2	151	54.4	1.6	9,077	0	0.628
Boston Edison Co.	0.89	10.3	202	57.9	2.2	5,088	25.3	1.555
Central Louisiana Co.	1.43	15.4	113	53	3.4	9,212	0	1.058
Commonwealth Edison Co.	1.02	11.2	168	56	0.3	6,423	34.3	0.7
Consolidated Edison Co. (NY)	1.49	8.8	192	51.2	1	3,300	15.6	2.044
Florida Power & Light Co.	1.32	13.5	111	60	−2.2	11,127	22.5	1.241
Hawaiian Electric Co.	1.22	12.2	175	67.6	2.2	7,642	0	1.652

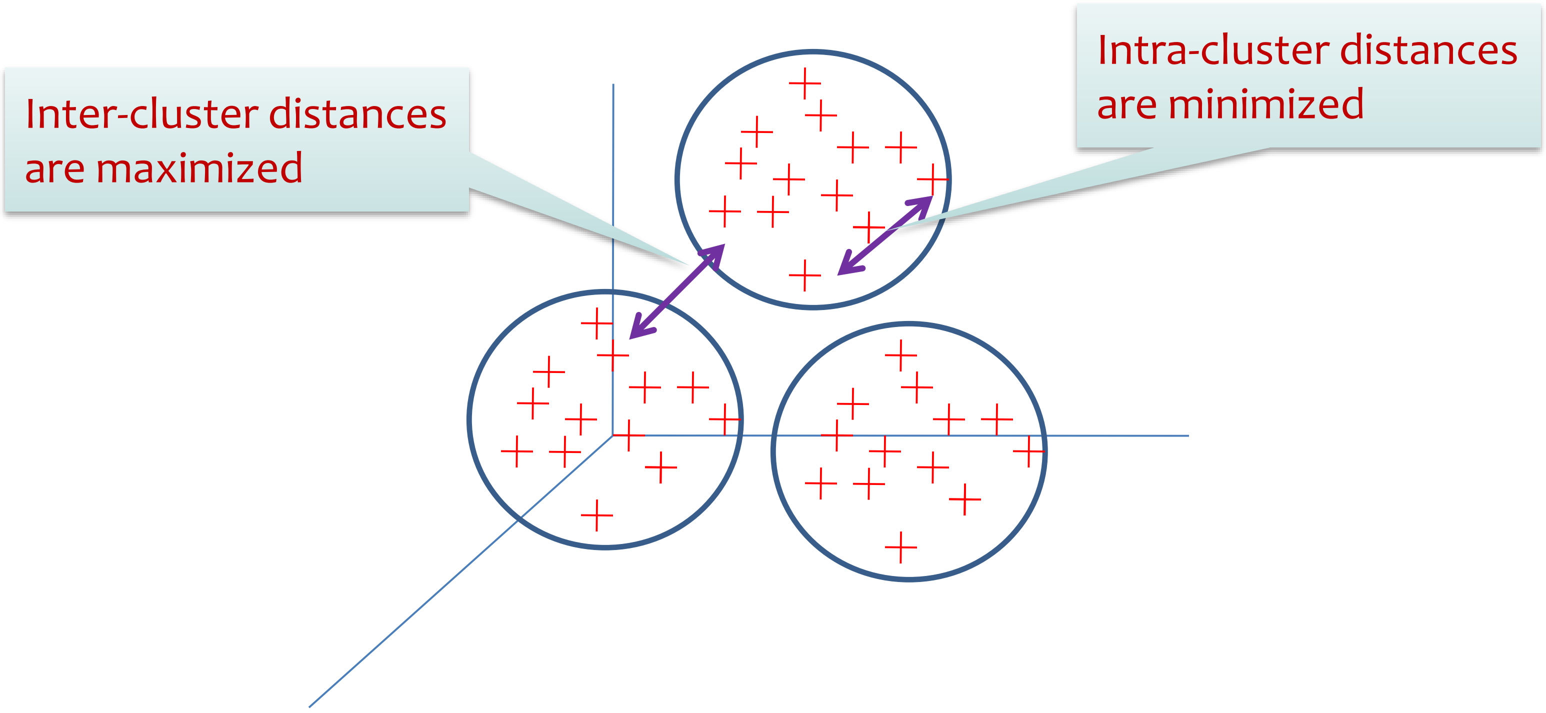
Normalizing Numerical Attributes

- Euclidean distance is highly influenced by the scale of each attribute, so that attributes with larger scales have a much greater influence over the total distance.
- Normalize numerical attributes, e.g. z-score

Normalized(sales) = (9,077-mean_sales)/std_sales

Clustering: Main Idea

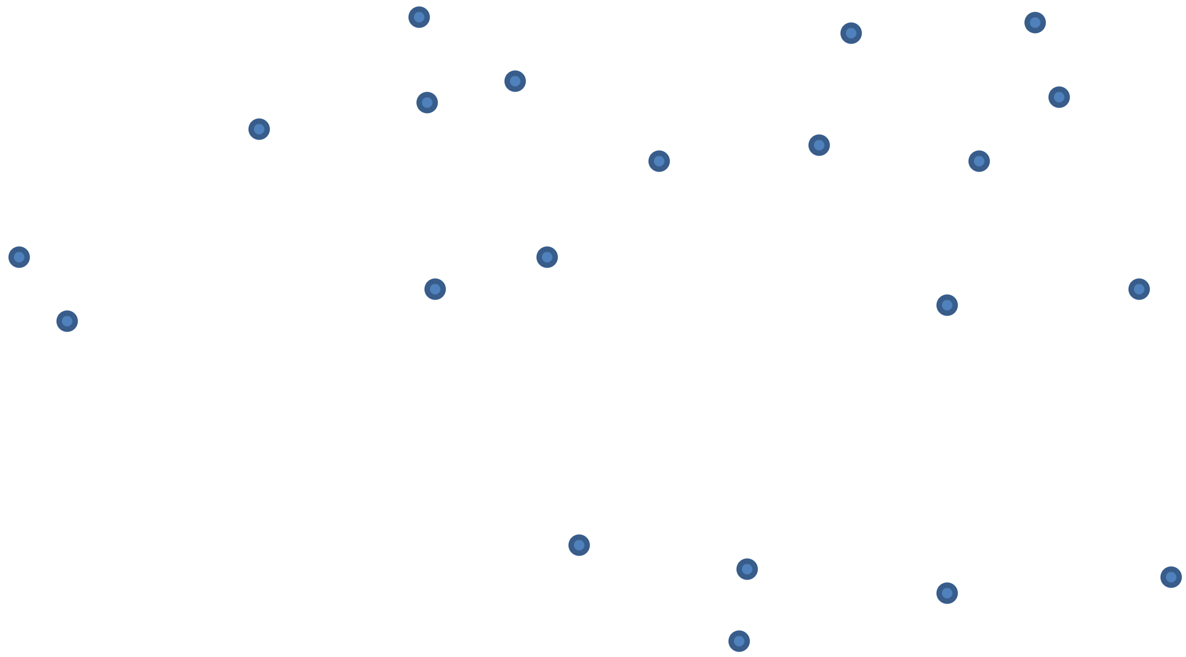
- Create clusters of records to achieve:
  - Maximum similarity between records within a cluster
  - Maximum dissimilarity between records of different clusters



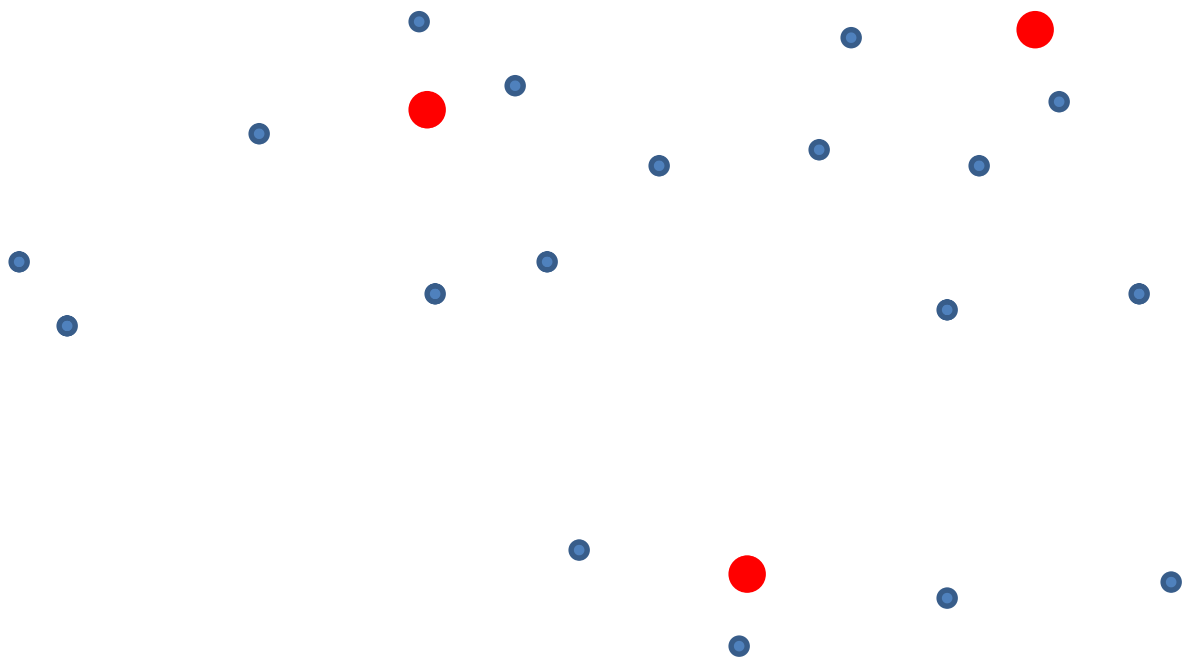
K-Means Clustering

- Most popular and simplest: *k*-means
- Each cluster is associated with a centroid (center point).
- Each point is assigned to the cluster with the closest centroid.
- Number of clusters *k* must be specified.
- Objective: minimize the within-cluster sum of squared distances (SSD) to the *k* centers.

An Example: *k*-Means (k=3)

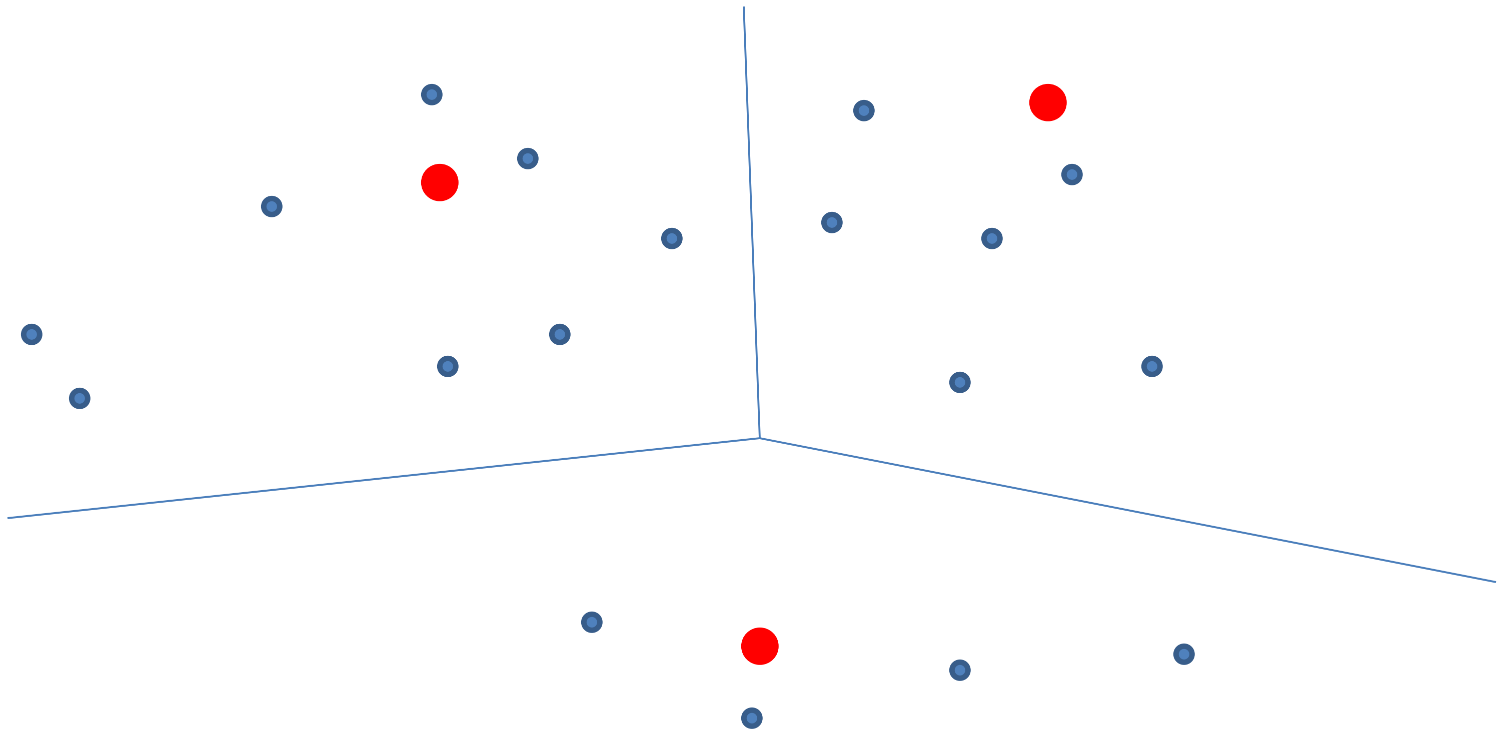


An Example: *k*-Means (*k*=3)



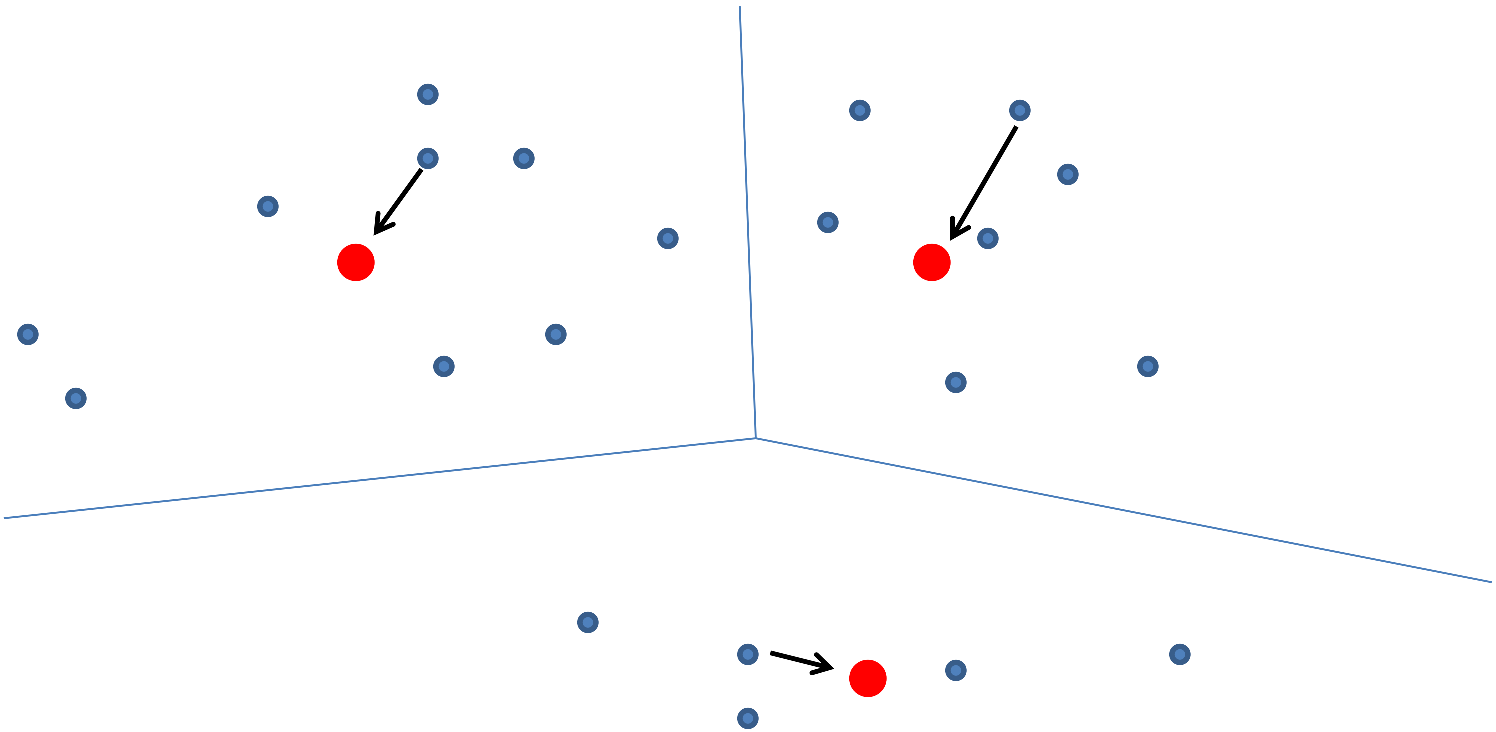
random assign *k* centroids

An Example: *k*-Means (*k*=3)



assign each example to the closest centroid

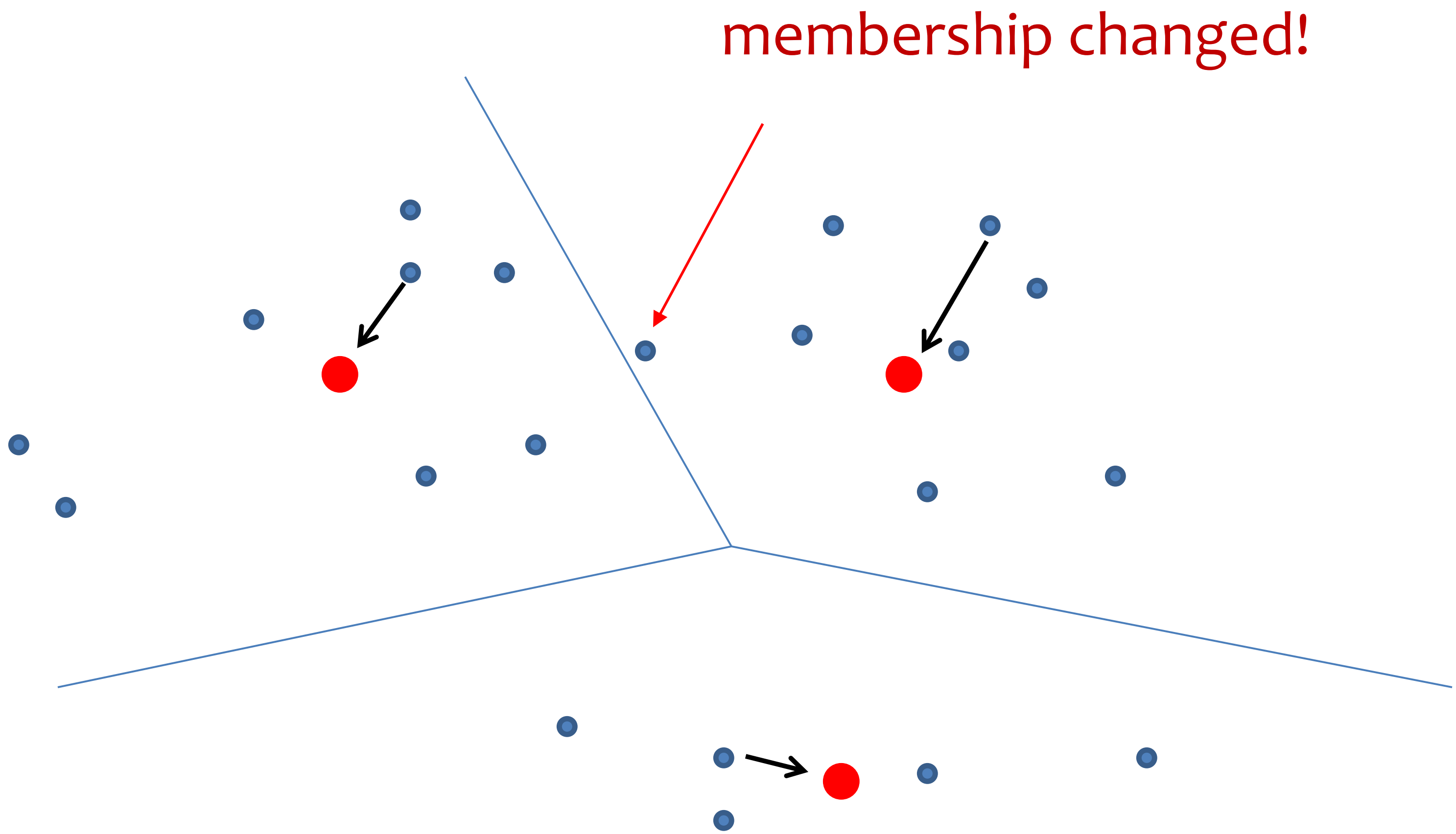
An Example: *k*-Means (*k*=3)



compute new centroids

Note: the new centroids may not be any points in the data.

An Example: *k*-Means (*k*=3)



assign each example to the closest centroid

k-Means Algorithm

Randomly choose *k* examples as initial centroids.

Repeat:

1. create *k* clusters by assigning each example to the closest centroid.
2. compute *k* new centroids by averaging examples in each cluster.
3. if centroids don't change, exit.

One iteration

k-Means Visualization

Online demo:

http://syskall.com/kmeans.js/

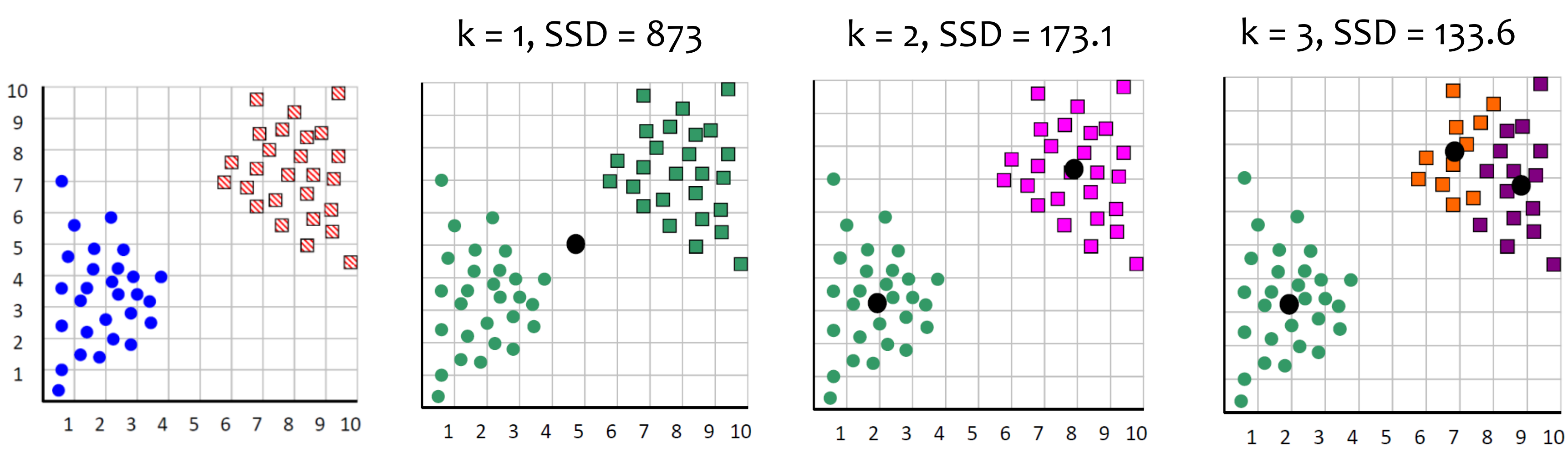
K-Means clustering

- Key notion: define a numeric space to represent objects.
- Some cons:
  - Sensitive to the selection of initial centers
  - Sensitive to noisy data and outliers
  - Not good in handling categorical variables
  - Need to specify k, the number of clusters, in advance

How do I choose k for k-means?

What is the Right Number of Clusters?

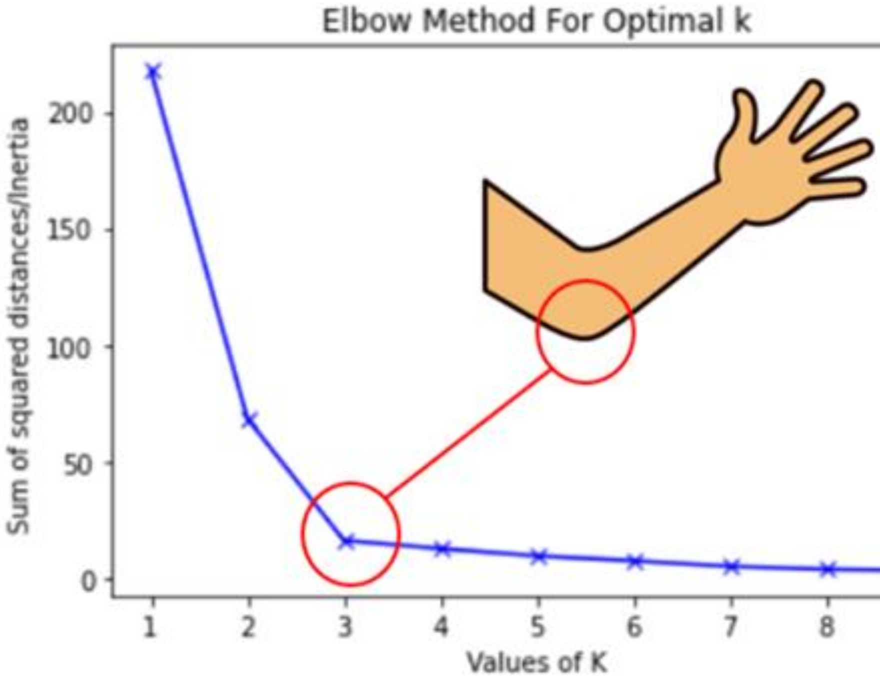
- In general, this is a unsolved problem. However, there are many approximate methods.
- Below is one example: using within-cluster sum of squared distances (SSD) as the objective and select the best *k* to minimize it.



Elbow Method

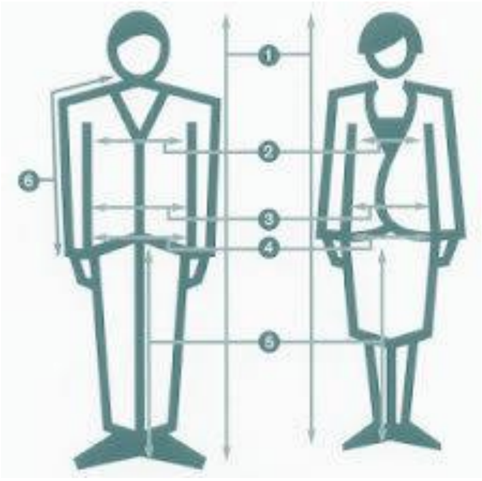
We can plot SSD values for incremental *k*.

The abrupt change at k = 3, is highly suggestive of three clusters in the data. This technique for determining the number of clusters is known as "elbow method".



Clustering Problem: Customers

- Finding clothing sizes
  - A customer is a data instance with values in multiple dimensions such as sizes for chest, waist, inseam, hips, neck, sleeve, etc.
  - Task: group people of similar sizes together to determine the number of sizes to offer (e.g., “small”, “medium”, “large”)

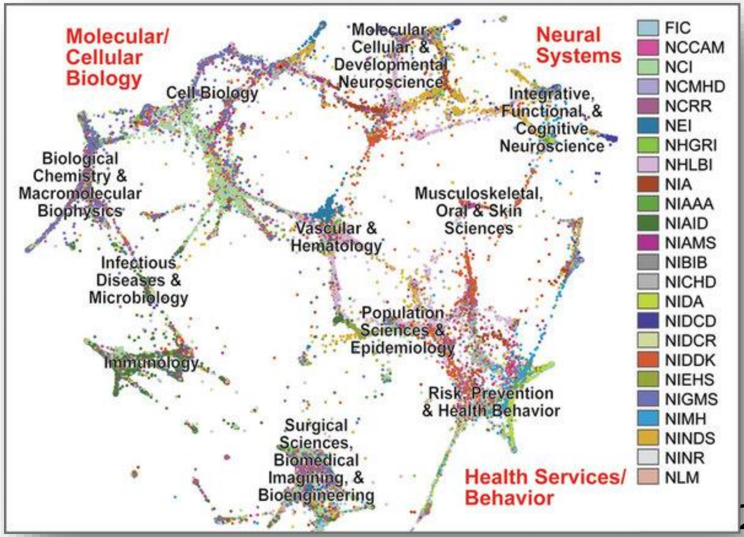


Clustering Problem: Music CDs

- Finding musical genres
  - Think of a customer as one attribute
    - Values in each dimension may be 0 or 1 only;
    - A CD is a point in this space $(x_1, x_2, \ldots, x_n)$, where $x_i = 1$ iff the i^{th} customer bought the CD
  - Task: CDs simultaneously bought by a large number of customers could be considered similar. Find clusters of similar CDs

Clustering Problem: Documents

- Finding document topics
  - Represent a document by a vector $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ if and only if the i^{th} word appears in the document
  - Task: find documents with similar sets of words; they may be about the same topic



Lab: Clustering

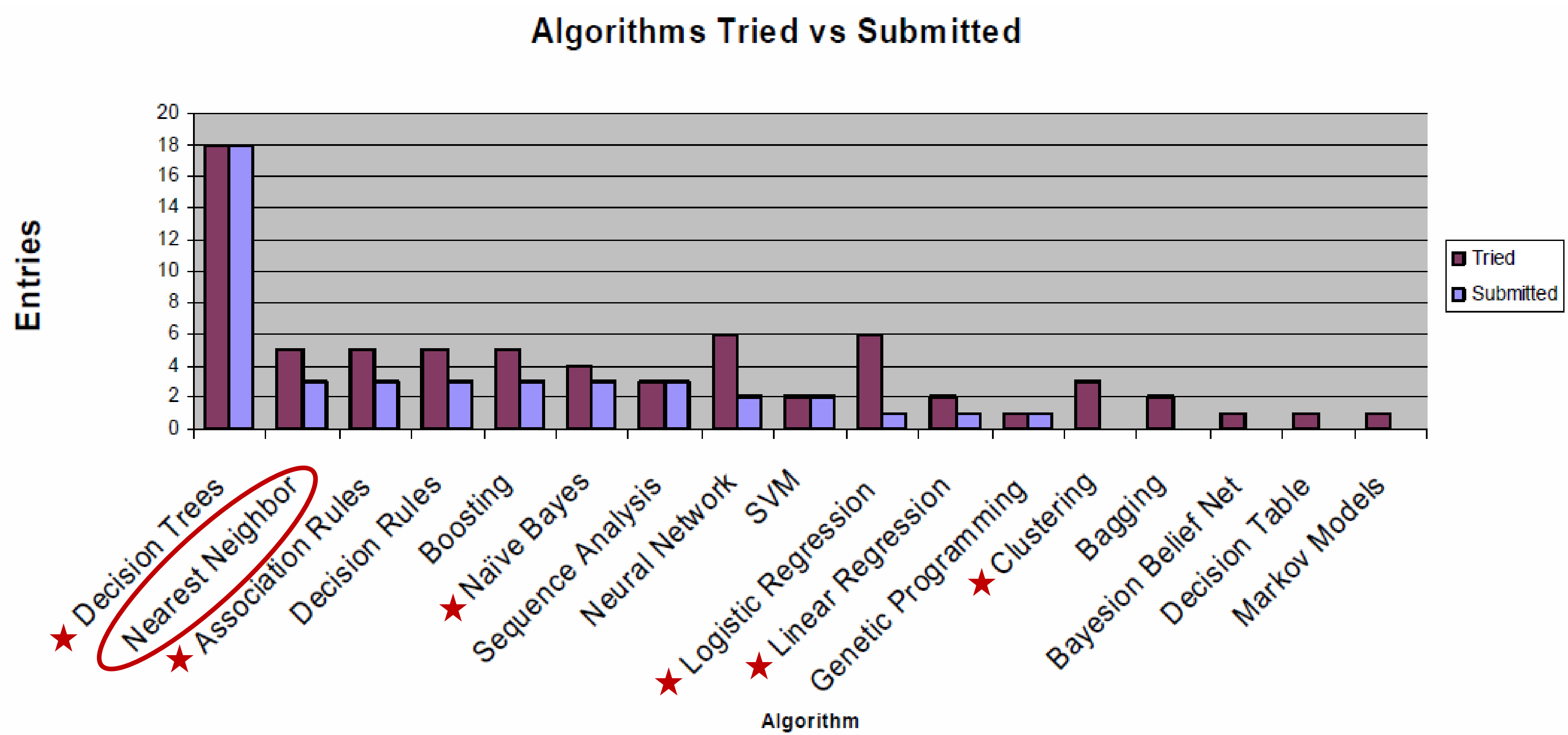
- Files needed
  - Clustering.ipynb
  - Mall_Customers.csv (dataset)

K-Nearest Neighbors

Instructor: Jing Wang
Department of ISOM
Spring 2023

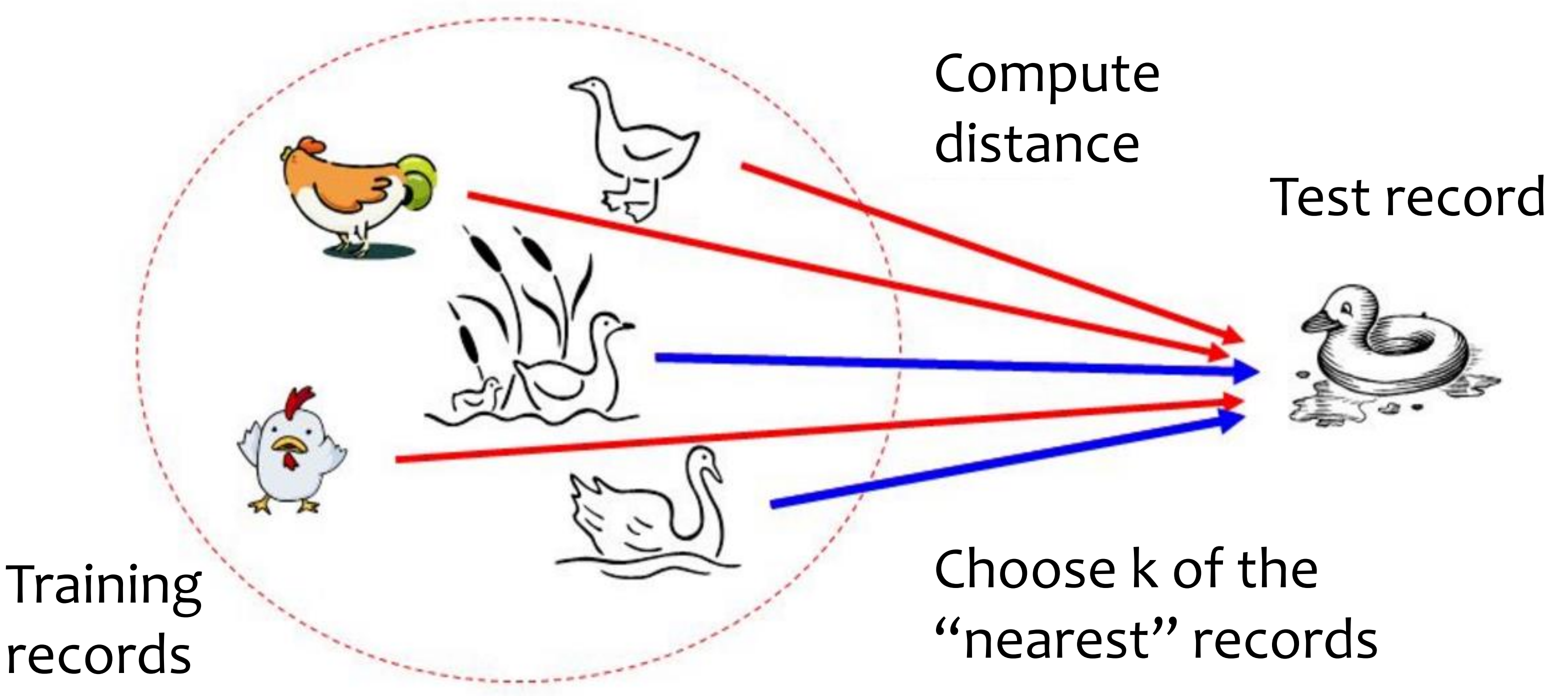
HKUST
BUSINESS SCHOOL
香港科大商學院

Commonly Used Induction Algorithms

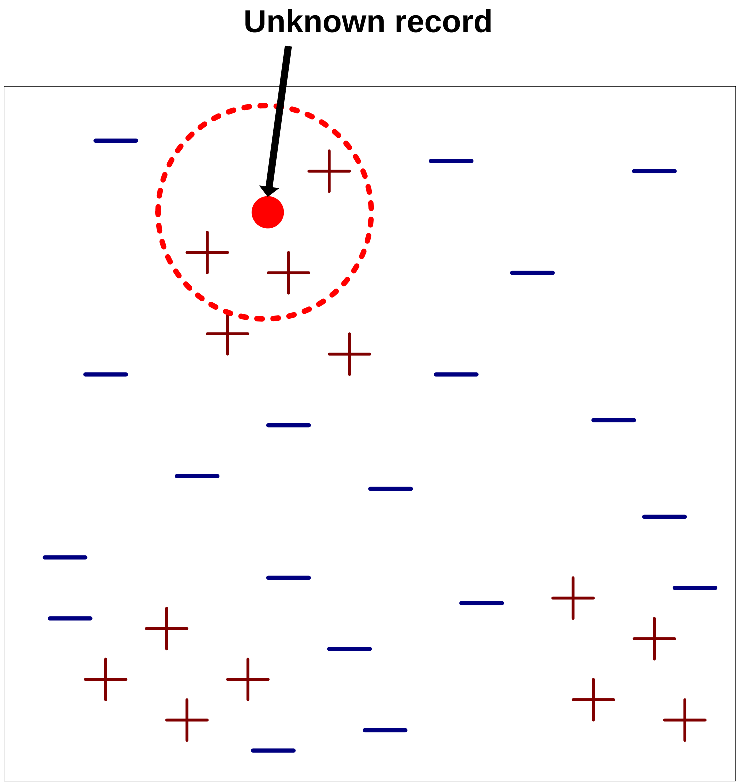


Nearest Neighbor Classification: The Idea

- If it walks like a duck, quacks like a duck, then it's probably a duck.

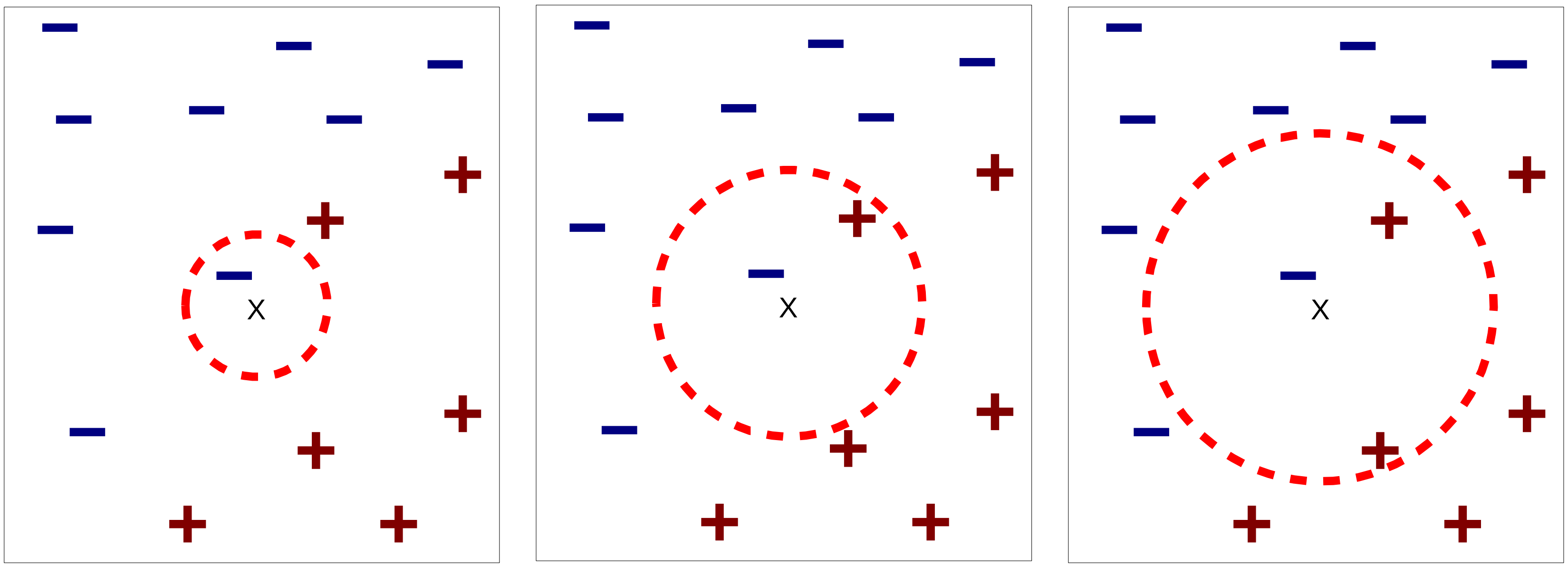


K-Nearest Neighbor Classification (*K*-NN)



- To classify an unknown example:
  - Compute distances between the example and all examples in training data
  - Identify the *k* nearest neighbors
  - Use class labels of *k* nearest neighbors to determine the class label of unknown example (e.g., by taking majority vote, proportion, or weighted average)

Changing *K* For Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

Changing *k* in the *k*-nearest neighbors may change the predicted class label of the example.

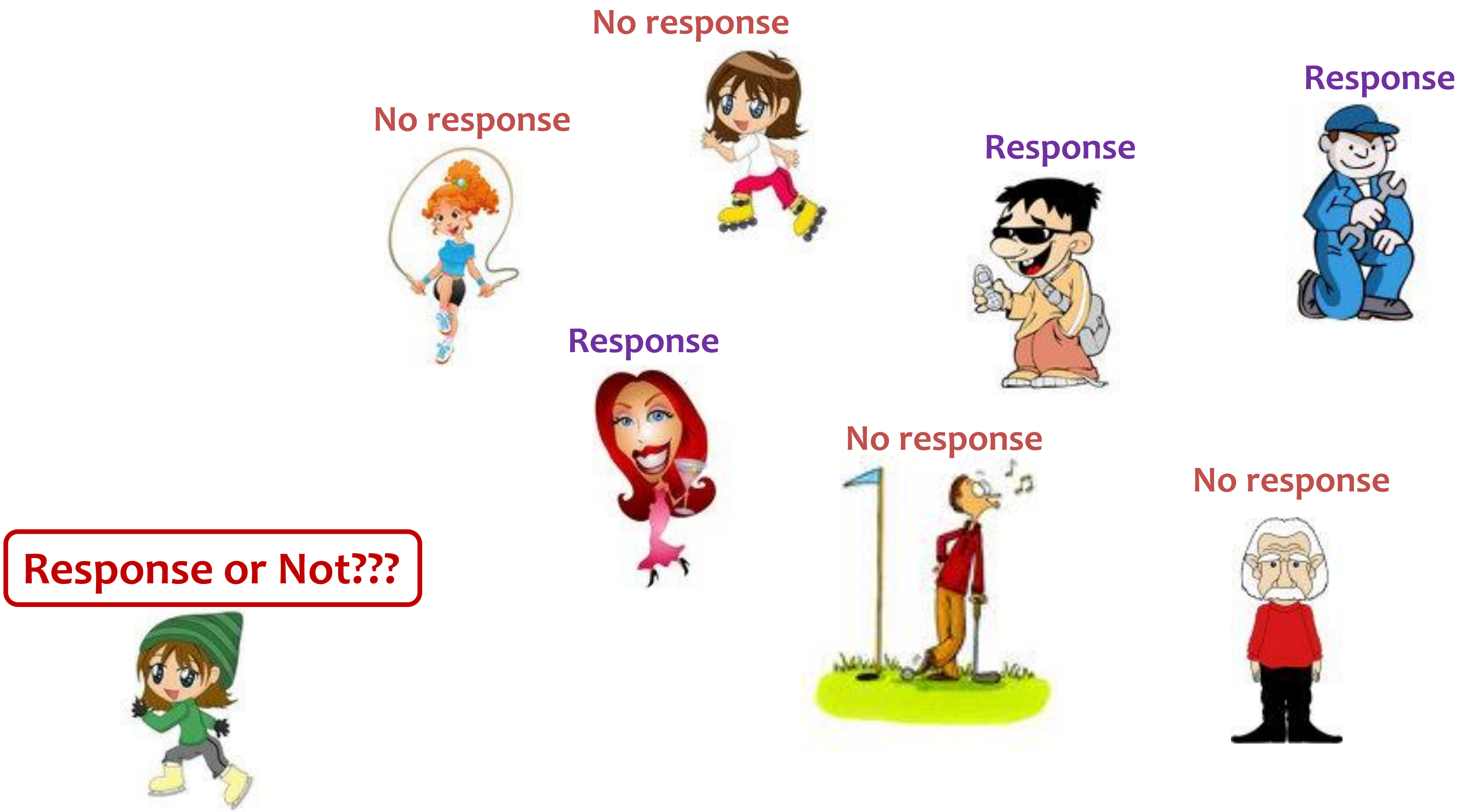
Nearest Neighbor Classification

- Compute distance between two points
  - Euclidean distance

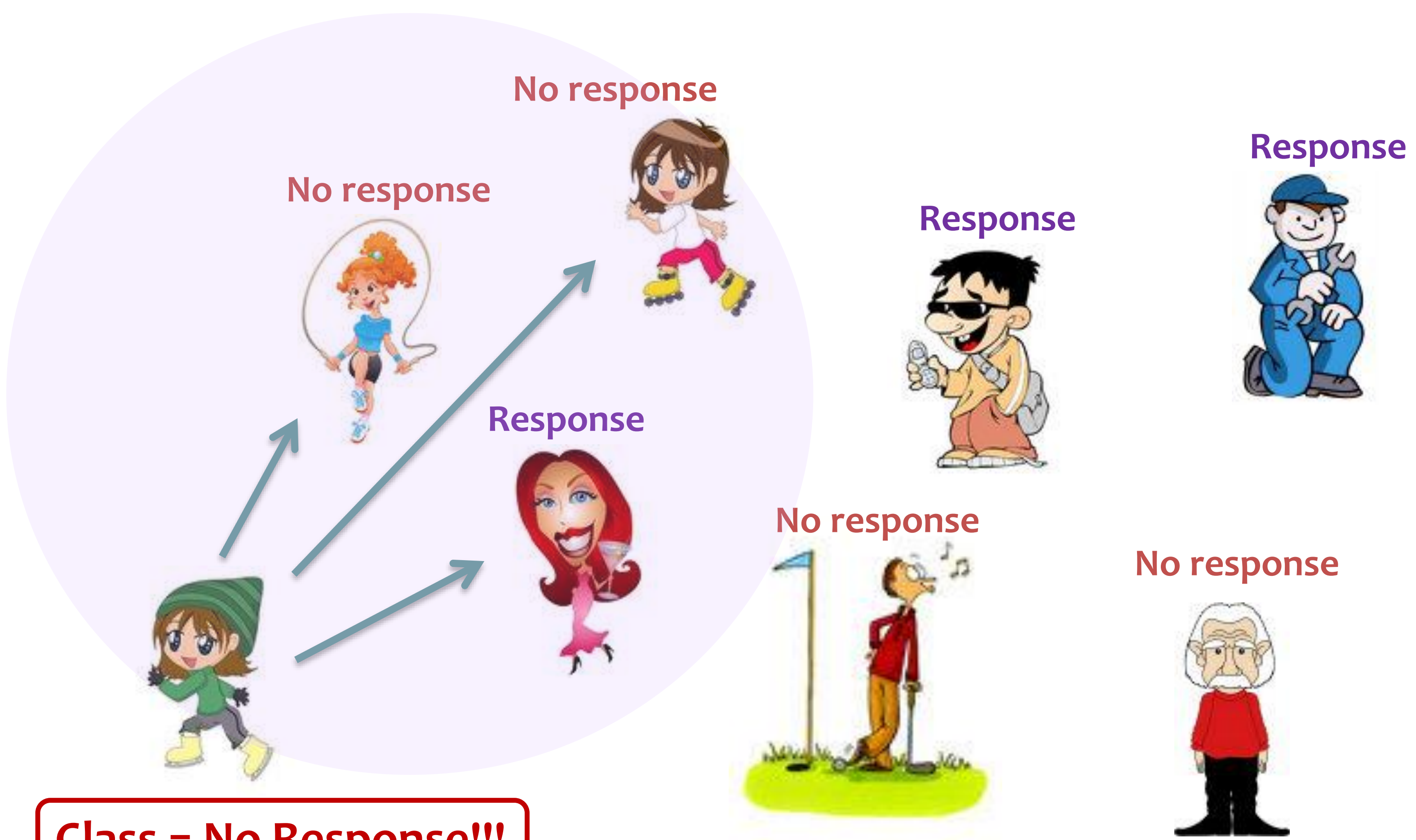
$$d_{ij} = \sqrt{\sum_k (x_{ik} - x_{jk})^2}$$

- Determine the class based on nearest neighbor list
  - Take the majority vote of class labels among the *k*-nearest neighbors
  - Weigh the vote according to distance
    - Weight factor, w = 1/d_{ij} or 1/d_{ij}^2

A Classification Example



3-Nearest Neighbor Algorithm



Class = No Response!!!

No model is built: store all training examples!

In-Class Exercise

- Lisa has lost gender information of one of her customers, and does not know whether to make a skirt or trousers. She is planning to throw a coin. Can you help her to make a better decision using a 3NN-classifier? The customer who is missing gender information: Gender=NA, Waist=28, Hip=34.

Gender	waist (cm)	hip (cm)	distance	ranking number	belongs to the neighborhood (Yes or No)
Male	28	32	Sqrt((28-28)^2+(34-32)^2)) 2	2	
Male	33	35		4	
Female	27	33	1.4	1	
Female	31	36	3.	3	

Class based on the majority vote, gender that gets most votes: __________

Feature Normalization in *K*-NN

- Examples of an attribute dominating distance computation
  - Age of a customer may vary from 10 to 100
  - Income of a customer may vary from $0 to $1M
  - No. of credit cards of a customer may vary from 0 to 20

Rachel:
Age=41
Income=$215K
No. of credit cards=3

John:
Age=35
Income=$95K
No. of credit cards=2

- Attributes have to be scaled to prevent distance measures from being dominated by one of the attributes
- Feature normalization (z-score): rescale features to have zero mean and unit variance.

Question

Does feature normalization always give us better model performance?

Not All Features Are Equally Informative

- In high dimensions if there are a lot of irrelevant features, normalization (scaling) may not help.

$$d_{ij} = \sqrt{\sum_{p} (x_{ip} - x_{jp})^2 + \sum_{q} (x_{iq} - x_{jq})^2}$$

informative features

noisy features

- If the number of informative features is smaller than that of noisy features, Euclidean distance is dominated by noise.

Any solutions to this problem based on what you have learned so far?

Solution: Feature Selection

- Perform feature selection to remove not informative features first, then apply K-NN.
  - Filter methods: use a proxy measure to score features.
  - Wrapper methods: use the error rate of a predictive model to score feature subsets.
  - Embedded methods: a catch-all group of techniques which perform feature selection as part of the model construction process.

But selected features may not be equally informative.

Solution: Feature Weighting (Optional)

- Feature weighting: weight each feature by its importance

$$d_{ij} = \sqrt{\sum_k w_k (x_{ik} - x_{jk})^2}$$

- How to determine weights w_k? (optional)
  - Performance bias methods: find a set of weights through an iterative procedure that uses the classifier's performance to select the next set of weights (best weights give best model performance).
  - Preset bias methods (filters): use a pre-determined function that measures e.g., mutual information and correlation between each feature and the target variable.

***Feature weighting is currently not implemented in python sklearn library.*

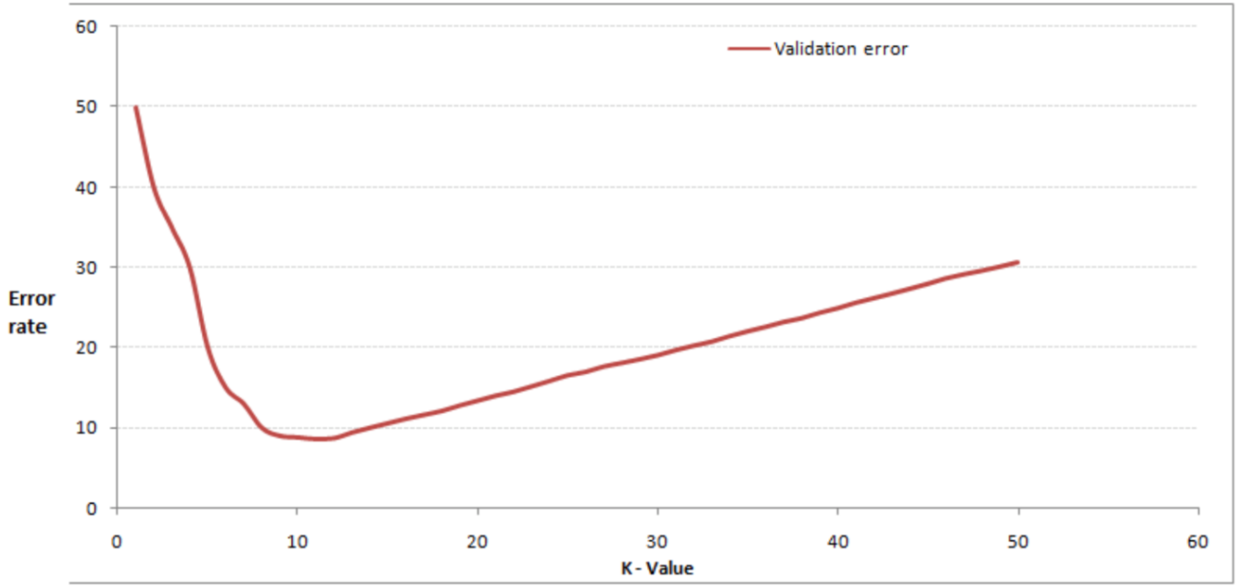
The Selection of *K*

- Think about two extreme cases:
  - *K*=1
  - *K*=N where N is the number of examples in the training data Benchmark model

Can they give us good classification performance?

The Selection of *K*

- We want to balance between overfitting (k=1) and no learning (k=N).
- Simple answer: we choose the k with the best classification performance!



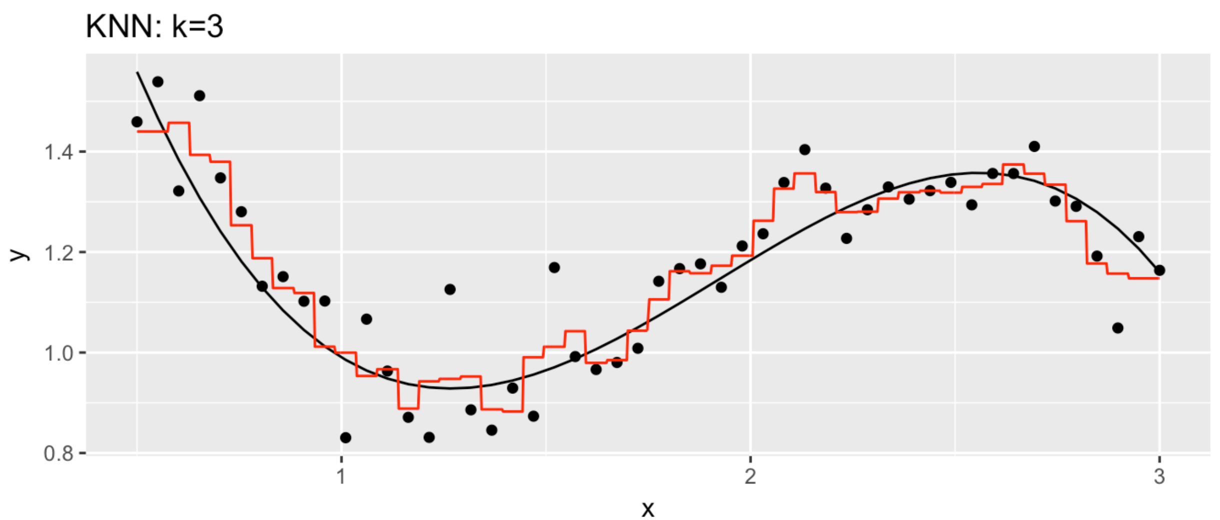
Questions

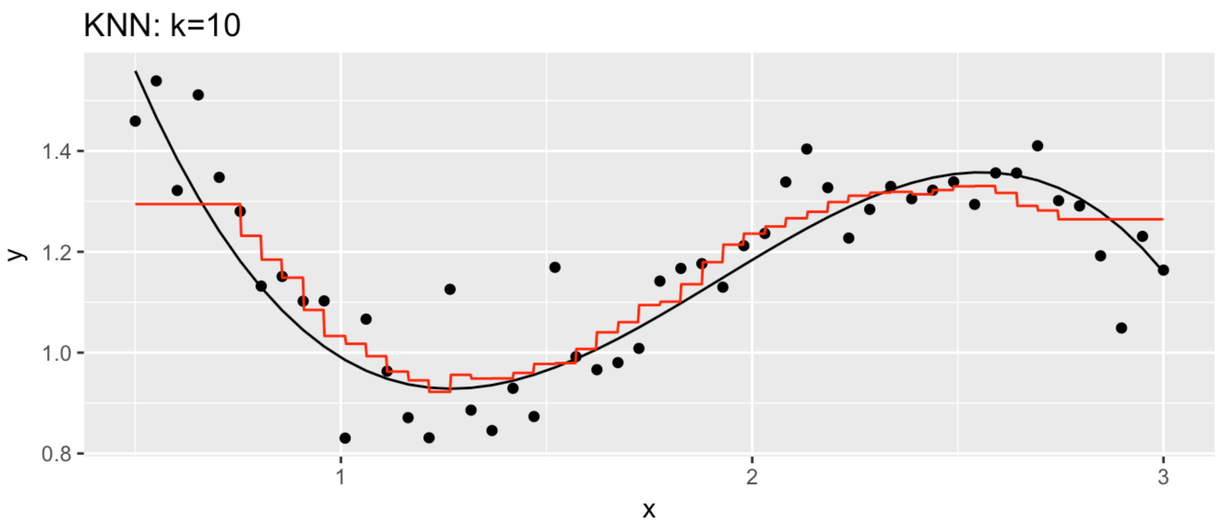
Can we use k-NN as for regression? Can we use k-NN for multi-class classification? If yes, how?

Yes

K-NN for Regression

- x: value of predictor attribute; y: value of target attribute





Strengths of *K*-NN

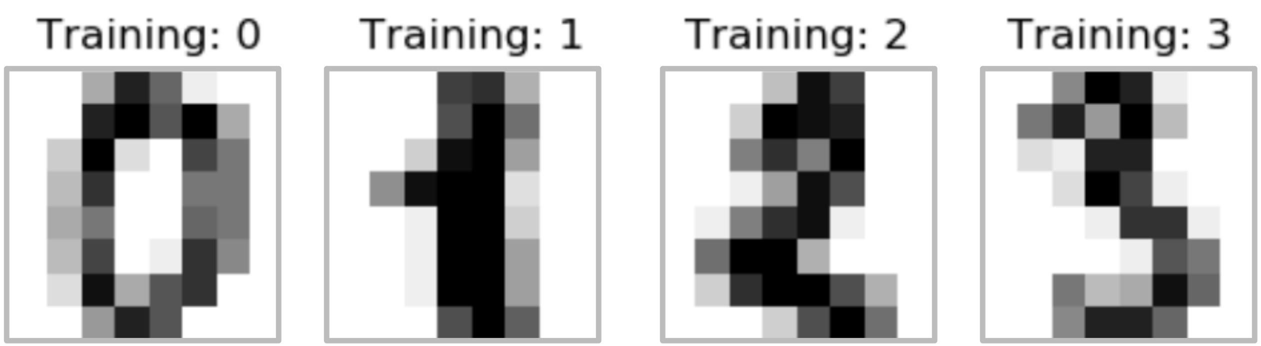
- Simple to implement and use
- Comprehensible – easy to explain prediction
- Robust to noisy data by averaging *k*-nearest neighbors (overfitting control).
- Some appealing applications
  - Collaborative filtering for recommender systems (next topic)

Weaknesses of *K*-NN

- The model can not be interpreted (there is no description of the learned model)
- Takes much longer time to classify/predict a new example
  - K-NN does not build models explicitly (lazy learners)
  - Need to calculate and compare distance from new example to all examples in training data
  - Prohibitively expensive for large number of examples
- Prone to the curse of dimensionality (feature selection/weighting, principal component analysis)

A KNN Application: Digit Recognition

- Handwritten digit recognition is an example of KNN application
  - Each data is a 8X8 (pixel) image of a digit (=64 attributes)
  - Feature values: 0->16 (white -> black)
  - There are 10 classes: 0, 1, 2, 3, 4, …, 8, 9



KNN idea: calculate the Euclidian distance (square root of sum of the squared differences of values in each pixel) and predict based on k-nearest training examples.

Lab: KNN for Digit Recognition

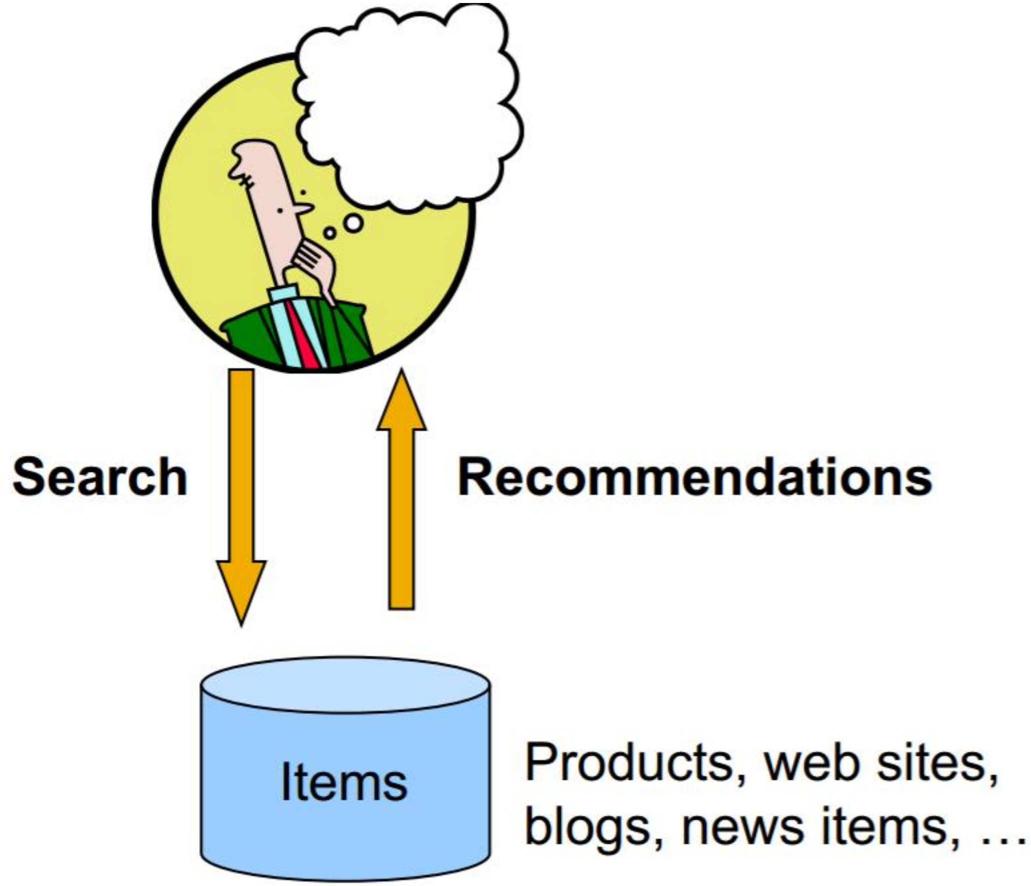
- Files needed
  - KNN.ipynb (python file)
  - digits.csv (dataset)

Recommender Systems Using Collaborative Filtering

Instructor: Jing Wang
Department of ISOM
Spring 2023

HKUST BUSINESS SCHOOL
香港科大商學院

Recommender Systems



Why Using Recommender Systems?

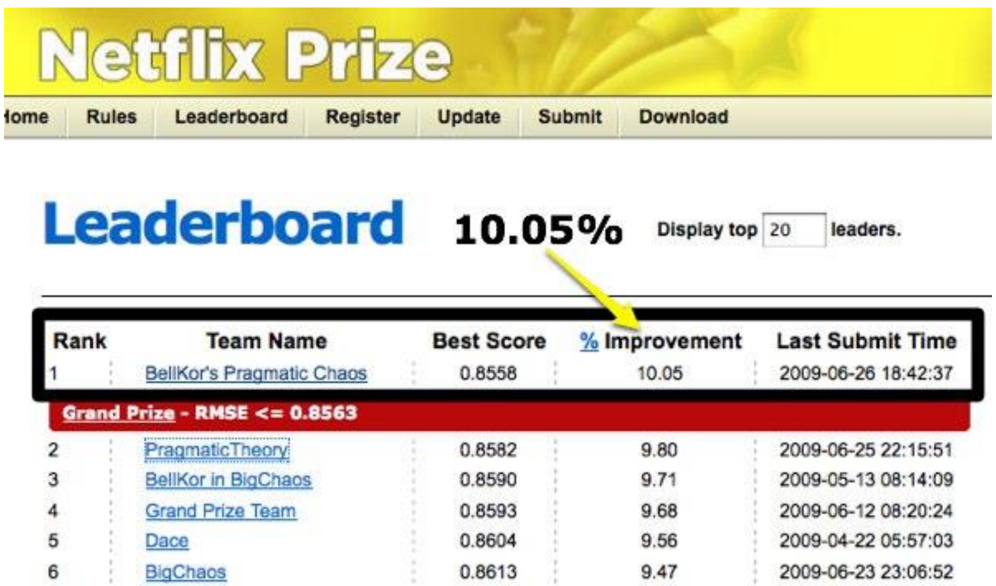
“A lot of times, **people don’t know what they want** until you show it to them…”

Steve Jobs

- Value for customers
  - Find things they like
  - Help explore the space of options
  - Reduce search and navigation time
  - …
- Value for e-commerce companies
  - Increase sales, click through rates, conversion, etc.
  - Opportunities for promotion, persuasion
  - Increase customer loyalty
  - …

The Netflix Prize

- Oct 2nd, 2006 --- an open competition for the best collaborative filtering algorithm to predict user ratings for films
- Grand prize of US$1,000,000 for improving the accuracy by 10%



Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	BellKor's Pragmatic Chaos	0.8558	10.05	2009-06-26 18:42:37
Grand Prize - RMSE <= 0.8563				
2	PragmaticTheory	0.8582	9.80	2009-06-25 22:15:51
3	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
4	Grand Prize Team	0.8593	9.68	2009-06-12 08:20:24
5	Dace	0.8604	9.56	2009-04-22 05:57:03
6	BigChaos	0.8613	9.47	2009-06-23 23:06:52

Netflix thinks its personalized recommendation engine is worth $1 billion per year

DM for Personalized Recommendations

- Association rules
  - Recommendation based-on occurrence of items purchased together
- Nearest-neighbor based collaborative filtering (CF)
  - **User-based**: find similar users to the target user and recommend what they liked
  - **Item-based**: find similar items to those that the target user have previously liked
- Other techniques
  - Out of the scope of this course.

Data Type and Format

- For n users and p items, we can think of the rating data as an $n \times p$ table

	Item ID				
User ID	I_1	I_2	I_3	…	I_p
U_1	$r_{1,1}$	$r_{1,2}$	$r_{1,3}$		$r_{1,p}$
U_2	$r_{2,1}$	$r_{2,2}$	$r_{2,3}$		$r_{2,p}$
…					
U_n	$r_{n,1}$	$r_{n,2}$	$r_{n,3}$		$r_{n,p}$

$r_{u,i}$ is missing is user u didn't rate item i.

Example: Movie Recommendation

- Given a set of ratings (e.g., 1-5), can we recommend the next set of movies to a user?

	Movie ID								
Customer ID	1	5	8	17	18	28	30	44	48
30878	4	1			3	3	4	5	
124105	4								
822109	5								
823519	3		1	4		4	5		
885013	4	5							
893988	3						4	4	
1248029	3					2	4		3
1503895	4								
1842128	4						3		
2238063	3								

Sample of records from the Netflix Prize contest

User-Based CF (Someone Like You)

Step 1: Find users who are most similar to the user of interest (neighbors). This is done by comparing the ratings of this user to the ratings of other users.

Step 2: Consider only the items that the user has not yet purchased, recommend the ones that are most preferred by the user's neighbors.

- How do we measure similarity?
- How many neighbors should we consider?
- How do we generate recommendations?

Measure User Similarity

- A popular similarity measure in user-based CF: Pearson correlation

$$\boldsymbol{sim(a,b)} = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

- a, b: users
- $r_{a,p}$: rating of user a for item p
- P: set of items, rated both by a and b
- $\bar{r}_a$: average of the user a's ratings (take into account the differences in rating scale between different users).

Example: Measure User Similarity

	Movie ID								
Customer ID	1	5	8	17	18	28	30	44	48
30878	4	1			3	3	4	5	
124105	4								
822109	5								
823519	3		1	4		4	5		
885013	4	5							
893988	3						4	4	
1248029	3					2	4		3
1503895	4								
1842128	4						3		
2238063	3								

$$\overline{r}_{30878} = (4+1+3+3+4+5)/6 = 3.333$$

$$\overline{r}_{823519} = (3+1+4+4+5)/5 = 3.4$$

$$\boldsymbol{sim(U_{30878}, U_{823519})} = \frac{(4-3.333)(3-3.4)+(3-3.333)(4-3.4)+(4-3.333)(5-3.4)}{\sqrt{(4-3.333)^2+(3-3.333)^2+(4-3.333)^2}\sqrt{(3-3.4)^2+(4-3.4)^2+(5-3.4)^2}}$$

$$= 0.6/1.75 = 0.34$$

Exercise

					Movie ID				
Customer ID	**1**	**5**	**8**	**17**	**18**	**28**	**30**	**44**	**48**
30878	4	1			3	3	4	5	
124105	4								
822109	5								
823519	3		1	4		4	5		
885013	4	5							
893988	3						4	4	
1248029	3					2	4		3
1503895	4								
1842128	4						3		
2238063	3								

Can you calculate the similarity between user_823519 and user_1842128?

Make Predictions

- So now we calculate the similarity between a user and all other users in the database.
- We look only at the k nearest users.
- Among all the other items that they rated, choose the best ones and recommend them to the user.
- The best ones are those with the highest predicted ratings, where the predicted rating of each item is given by the weighted average of the ratings given by the k nearest users on that item.

Prediction Function (Optional)

- A common prediction function:

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} |sim(a,b)|}$$

  - Calculate, the difference between neighbors' ratings for the unseen item *p* and their average ratings
  - Combine the rating differences – use the similarity with a as a weight
  - Add the target user's average rating to the weighted average and use this as a prediction

Further Improvements

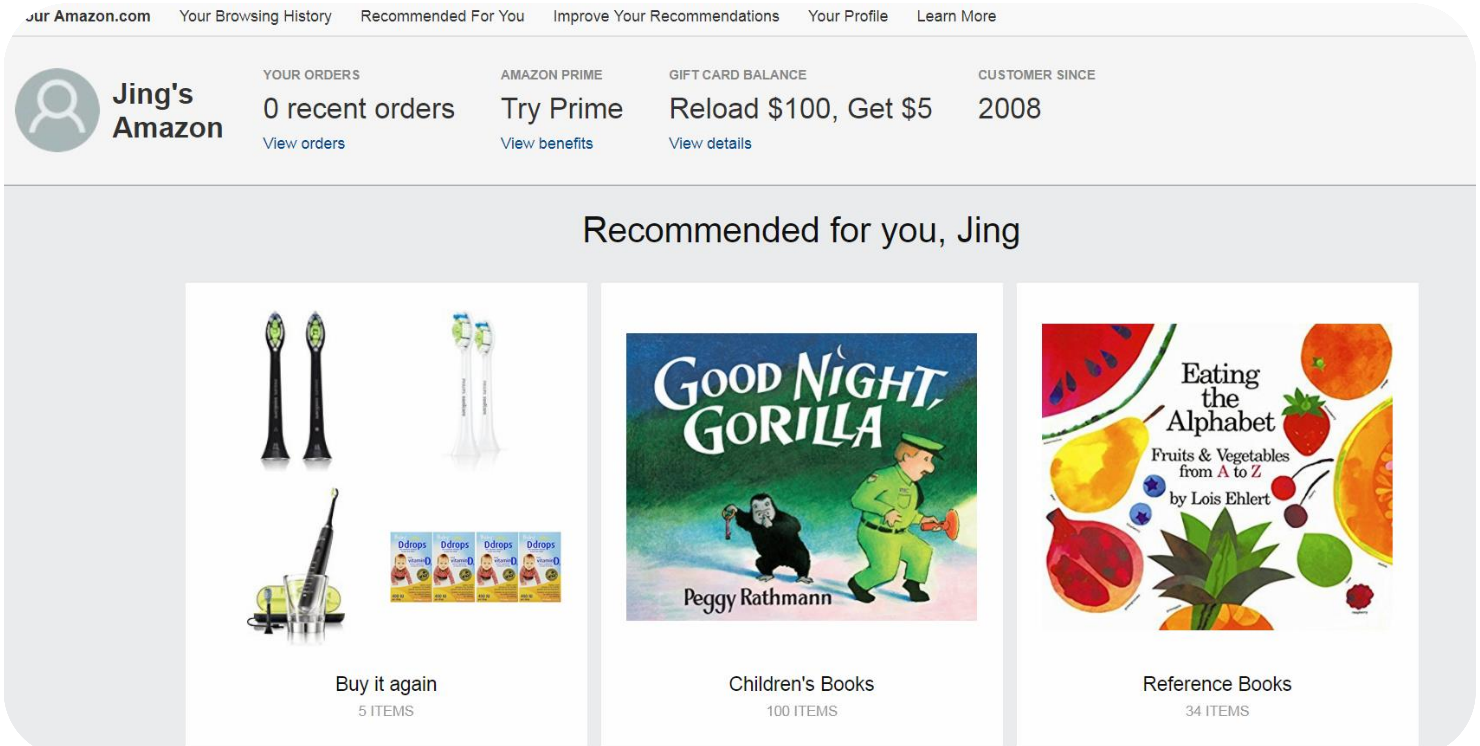
- Not all neighbor ratings might be equally “valuable”
  - Agreement on commonly liked items is not so informative as agreement on controversial items.
  - Possible solution: give more weight to items that have a higher variance.
- Number of co-rated items might be different
  - Use “significance weighting”, by e.g., linearly reducing the weight when the number of co-rated items is low.

Challenges of CF: Scalability

- Scalability
  - Nearest neighbor algorithms require computation that grows with both the number of users and the number of items. With millions of users and items, a recommender system will suffer serious scalability problems.
- Solution: pre-processing (calculate similarity beforehand)

When you have scalability issue, the item-based CF is more preferred in practice!

Amazon: Item-Based Collaborative Filtering



https://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf

Item-Based CF

- When the number of users is much larger than the number of items, it's faster to find similar items rather than similar users. -> Item-Based CF!

Step 1: For each item rated by the user of interest, calculate the similarity with all the other items.

Step 2: Consider only the items that the user has not yet purchased, recommend the ones whose neighbors are highly rated by the user of interest.

Measure Item Similarity

- Pearson correlation

$$\boldsymbol{sim(i,j)} = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_i)^2}\sqrt{\sum_{u \in U}(r_{u,j} - \bar{r}_j)^2}}$$

  - i, j: items
  - $r_{u,i}$: rating of user u for item i
  - U: set of users who have rated both i and j
  - $\bar{r}_i$: average of the item i's ratings across all users.

Measure Item Similarity

Customer ID	Movie ID 1	5	8	17	18	28	30	44	48
30878	4	1			3	3	4	5	
124105	4								
822109	5								
823519	3		1	4		4	5		
885013	4	5							
893988	3						4	4	
1248029	3					2	4		3
1503895	4								
1842128	4						3		
2238063	3								

$\bar{r}_1 = 3.7, \qquad \bar{r}_5 = 3$

$$sim(I_1, I_5) = \frac{(4-3.7)(1-3) + (4-3.7)(5-3)}{\sqrt{(4-3.7)^2 + (4-3.7)^2}\sqrt{(1-3)^2 + (5-3)^2}} = 0$$

Measure Item Similarity: Alternative

- Adjusted cosine similarity

$$\boldsymbol{sim(i,j)} = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{u \in U}(r_{u,j} - \bar{r}_u)^2}}$$

  - i, j: items
  - $r_{u,i}$: rating of user u for item i
  - U: set of users who have rated both i and j
  - $\bar{r}_u$: average of the user u's ratings on all items (take into account the differences in rating scale between different users).

Make Predictions

- So now we calculate the similarity between the items rated by a user and all other items in the database.
- Among all the other items that are not rated by the user, choose the best ones and recommend them to the user.
- The best ones are those with the highest predicted ratings, where the predicted rating of each item is given by the weighted average of the ratings given to the k nearest items by the user of interest.

Prediction Function (Optional)

- A common prediction function:

$$pred(a,p) = \frac{\sum_{i \in N} sim(i,p) * r_{a,i}}{\sum_{i \in N} |sim(i,p)|}$$

- Predict user *a*'s rating on item *p*, using the weighted average of the ratings given by user *a* on other items (across all the *i*'s), who are the neighbors of item *p*.
- Neighborhood size is typically also limited to a specific size (e.g., 10, 20, 50)

Pre-Processing for Item-Based CF

- Pre-processing approach by Amazon.com (in 2003)
  - Item similarities are supposed to be more stable than user similarities.
  - Calculate all pair-wise item similarities in advance
  - The neighborhood to be used at run-time is typically rather small, because only items which the user has rated are taken into account.
- Memory requirements
  - Based on an offline pre-processing or “model-learning” phase
  - Up to N^2 pair-wise similarities to be memorized (N = number of items) in theory
  - In practice, this is significantly lower (items with no co-ratings)

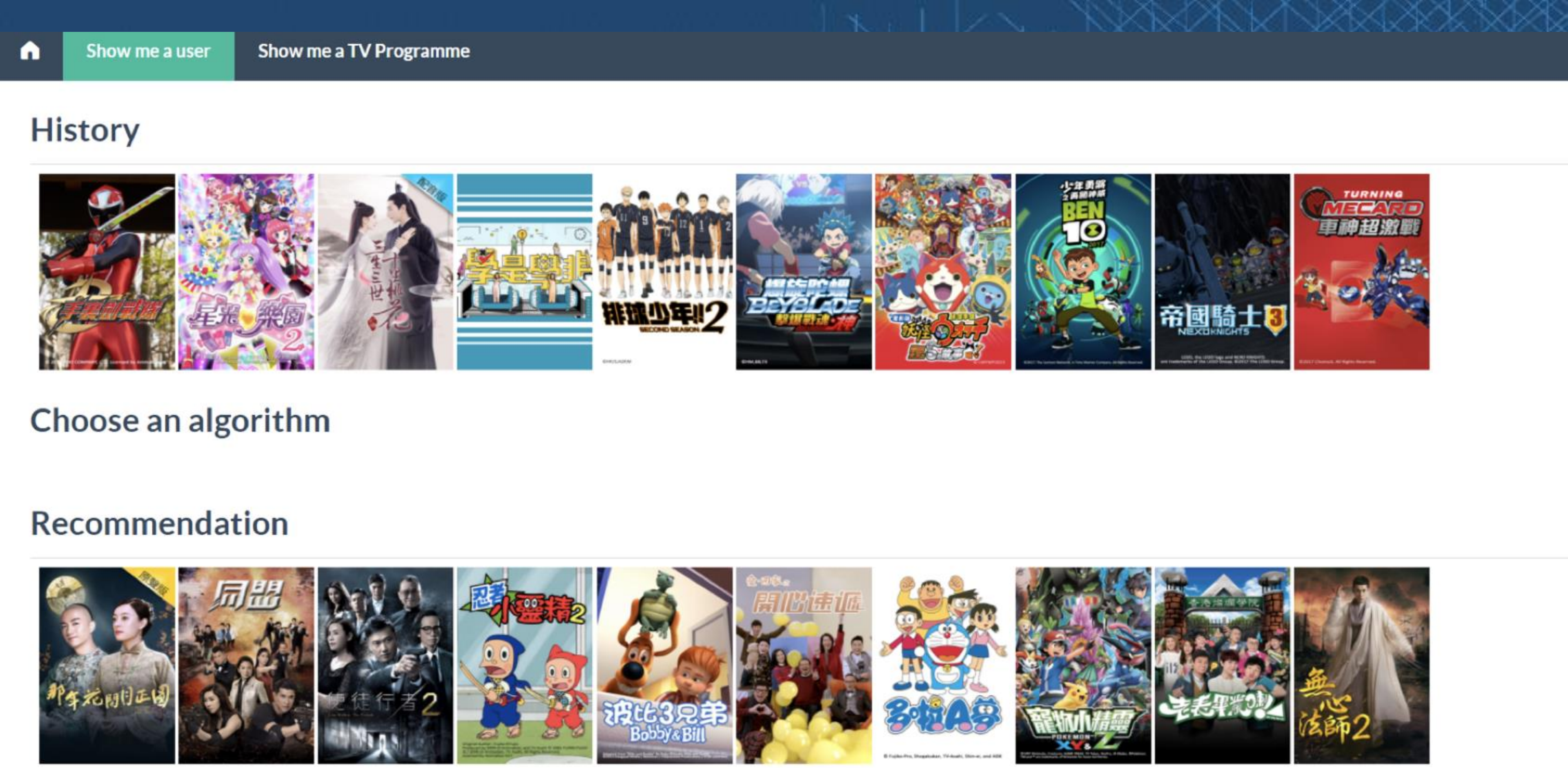
Pre-Processing: User-Based vs. Item-Based

- User-based similarity is more dynamic.
  - Precomputing user neighbourhood can lead to poor predictions.
- Item-based similarity is more static.
  - We can precompute item neighbourhood.

Item-based filtering does not solve the scalability problem itself.
It relies on pre-processing to reduce online computation time.

Challenge of CF: Cold Start

- Cold start problem
  - How to recommend new items?
  - What to recommend to new users?
- Straightforward approaches
  - Ask/force users to rate a set of items
  - Use another method (e.g., content-based, demographic or simply non-personalized) in the initial phase
- Alternatives
  - Use other algorithms (beyond nearest-neighbor approaches).



Recommendation system project with TVB myTVSuper service

Discussion

Let's say you are going to build a recommendation system for TVB myTVSuper service. This service has about 2 million users, and 10K TV programs. Think about how you can do it. Which of the following methods might be a better choice? Why?

1. Item-based collaborative filtering
2. User-based collaborative filtering

Evaluation (I)

- From data mining or user perspective:
  - Accurate rating prediction is better (MSE, RMSE, MAE)

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

  - Accurate prediction of the decision is better (precision, recall, AUC)

Evaluation (II)

- From business perspective:
  - Click-through rate
  - Conversion rate
  - Total viewing time
  - …

Somebody Took It After Three Years

Netflix Prize COMPLETED

Home | Rules | Leaderboard | Update

Leaderboard

Showing Test Score. Click here to show quiz score

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11

Much more complicated than the basic collaborative filtering!

Netflix provided a training data set of 100,480,507 ratings that 480,189 users gave to 17,770 movies. The competition began on October 2, 2006.
http://www.youtube.com/watch?v=ImpV70uLxyw

Lab: User-Based Collaborative Filtering

- Files needed
  - Movie Recommend.ipynb (python file)
  - movies.csv, ratings.csv (datasets)