KNN background review

June 2020
Università della Svizzera italiana

Amirehsan Davoodi

Supervisors: Cesare Alippi - Daniele Zambon

Contents

- Neighbors-based learning
  - KD tree
  - Ball tree
  - Parametric vs Non-parametric learning
  - Decision tree
  - Bagging – variance problem
  - Boosting – bias problem
- Spectral Clustrering
- Graph Edit Distance
- Munkres Algorithm to solve GED
- Local outlier Factor (LOF)

Neighbors-based learning

Neighbors-based learning

- **Definition:** Find a predefined number of training samples closest in distance to the new point and predict the label from these.
- Neighbors-based learning is known as **non-generalizing machine learning** methods, since they simply "remember" all its training data.

❖ Unsupervised o Manifold learning o Spectral clustering	❖ Supervised o Classification: for data with discrete labels o Regression: for data with continuous labels

- **Number of closest samples** can be:

❖ User-defined constant: o K-nearest neighbor learning	❖ Vary based on the local density of points o Radius-based neighbor learning

- All the training samples can be transformed into a fast indexing structure such as **Ball tree** or **KD tree**
- Being a **non-parametric method**, it is often successful in classification situations where the decision boundary is very irregular.

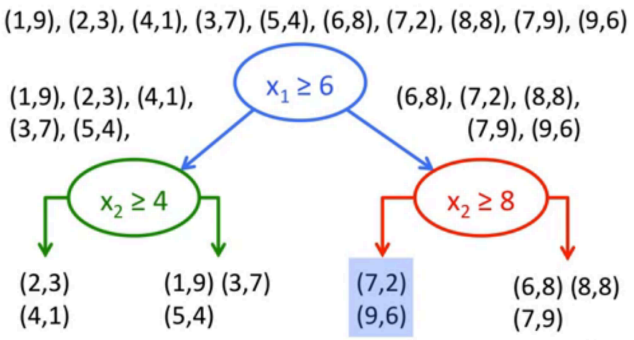
Neighbors-based learning
KD tree

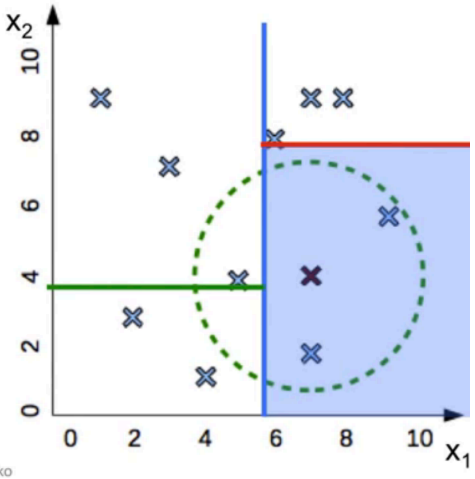
Making kNN fast

- Training: O(1), but testing: O(*nd*)
- Reduce *d*: dimensionality reduction
  - simple feature selection, other methods O(d^3)
- Reduce *n*: don't compare to **all** training examples
  - idea: quickly identify *m*<<*n* potential near neighbors
    - compare only to those, pick *k* nearest neighbors → O(*md*) time
  - **K-D trees**: low-dimensional, real-valued data
    - O (*d* log *n*), only works when *d* << *n*, inexact: may miss neighbors
  - **inverted lists**: high-dimensional, discrete data
    - O (*d'n'*) where *d'*<<*d*, *n'*<<*n*, only for sparse data (e.g. text), exact
  - **fingerprinting**: high-d, sparse or dense
    - O(*dn'*), *n'*<<*n* … bits in fingerprint, inexact: may miss near neighbors

K-D tree example

- Building a K-D tree from training data:
  - pick random dimension, find median, split data, repeat
- Find NNs for new point (7,4)
  - find region containing (7,4)
  - compare to all points in region

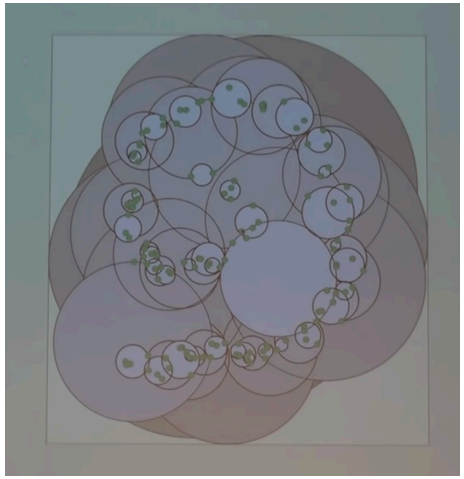


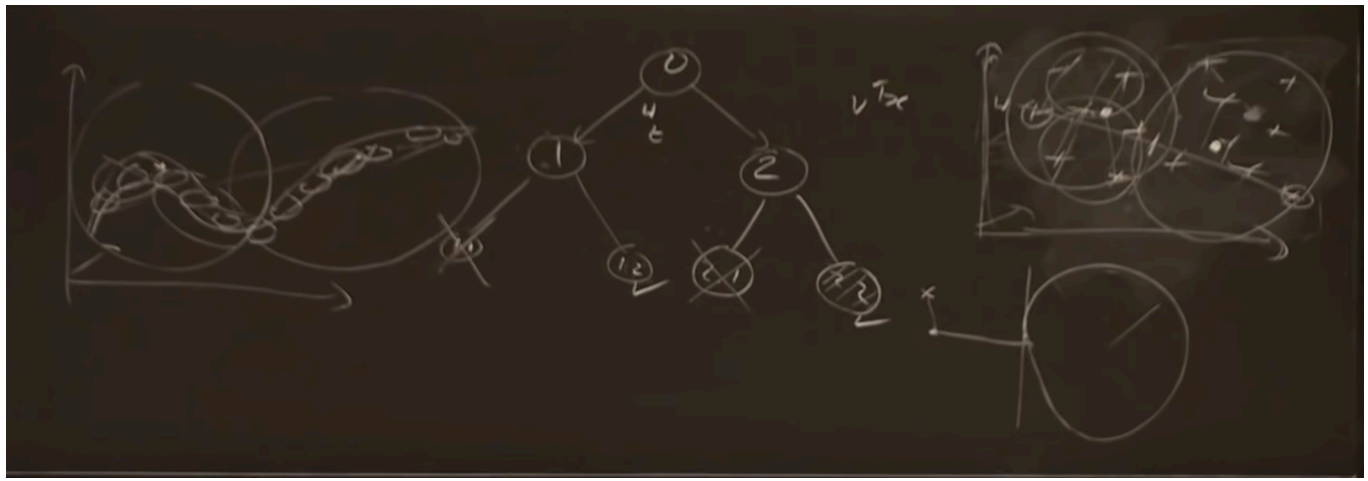


Neighbors-based learning
Ball trees

Steps:

- Find the line (take one random point and find the first point with the longest distance and find the second point which is having the longest distance to the first point.
- Map every datapoints to the line.
- Find the median and balls with the centre of the mean for right and left side of the median of the point.
- Repeat to have smaller ball.

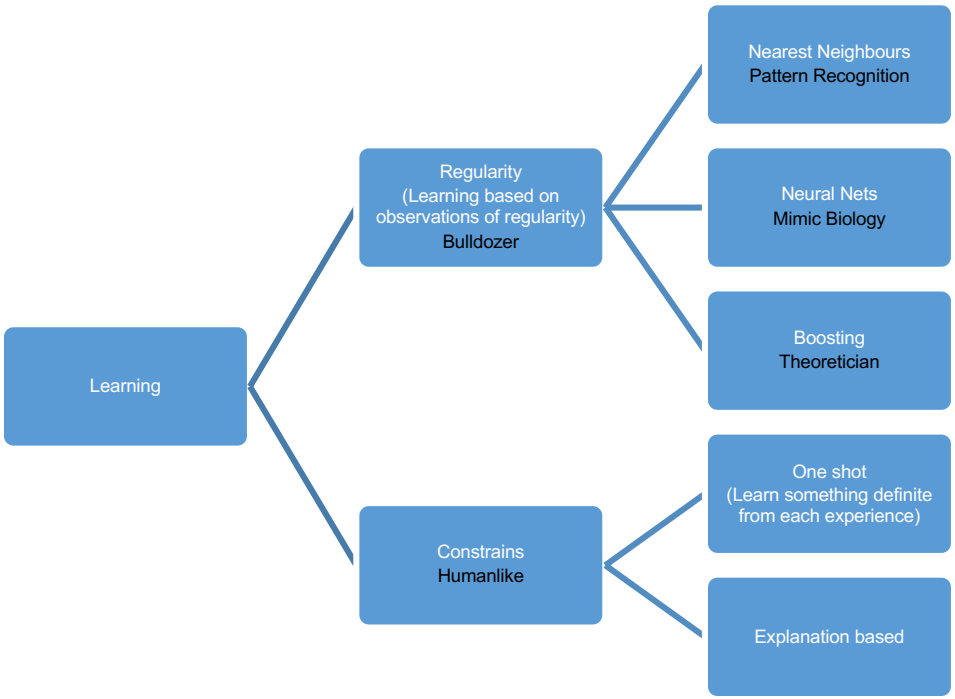




Parametric vs Non-parametric

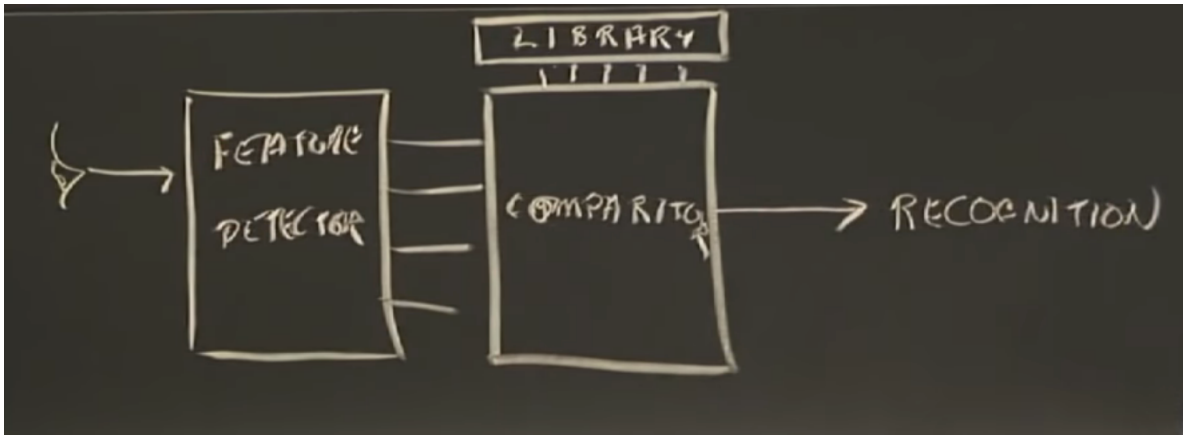
- **Parametric Machine Learning Algorithms:** Assumptions can greatly simplify the learning process, but can also limit what can be learned. Algorithms that simplify the function to a known form are called parametric machine learning algorithms.
  - A learning model that summarizes data with a set of parameters of fixed size (independent of the number of training examples) is called a parametric model. No matter how much data you throw at a parametric model, it won't change its mind about how many parameters it needs. (Artificial Intelligence: A Modern Approach, page 737)
- **Nonparametric Machine Learning Algorithms**: Algorithms that do not make strong assumptions about the form of the mapping function are called nonparametric machine learning algorithms. By not making assumptions, they are free to learn any functional form from the training data.
  - Nonparametric methods are good when you have a lot of data and no prior knowledge, and when you don't want to worry too much about choosing just the right features. (Artificial Intelligence: A Modern Approach, page 757)

Introduction to Learning, Nearest Neighbours



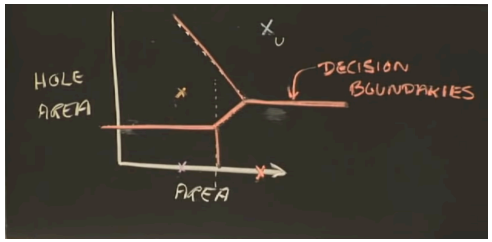
Nearest Neighbours learning

Feature detector: Mechanism to generates vector of features



Nearest Neighbours learning

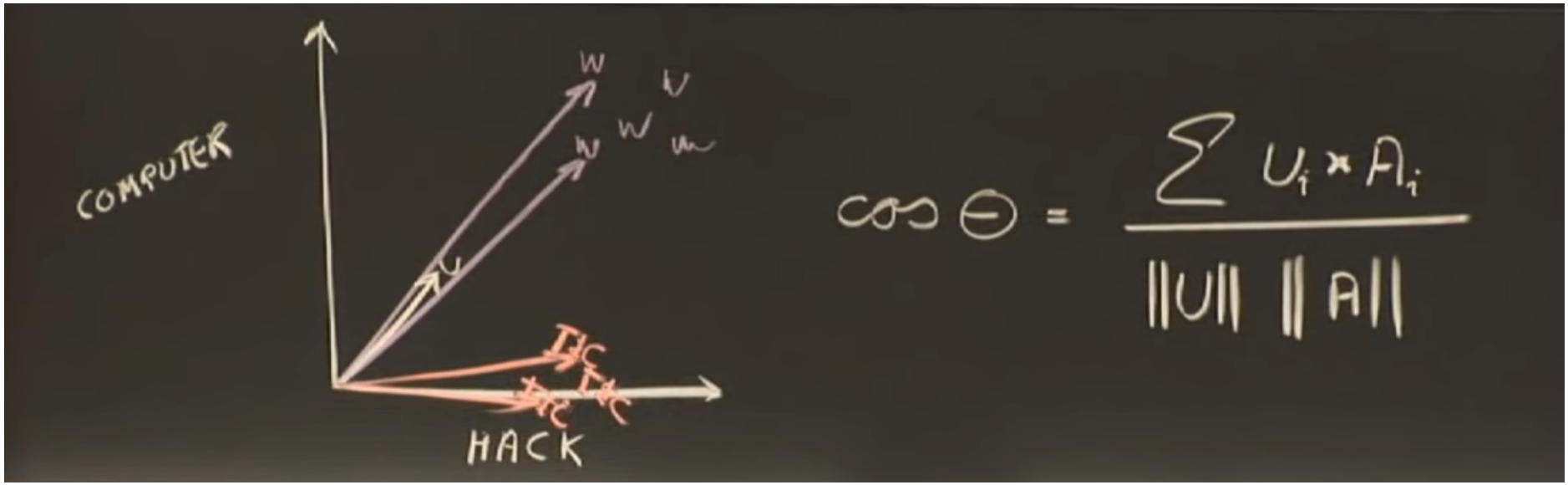
- Decision boundaries:



- Principles:
  - If something is similar in some respects, it's likely to be similar in other respects.

Nearest Neighbours learning

- Instead of decision boundaries, we can use the cosine of the angle which is the dot product of the two vector



Identification trees and disorder

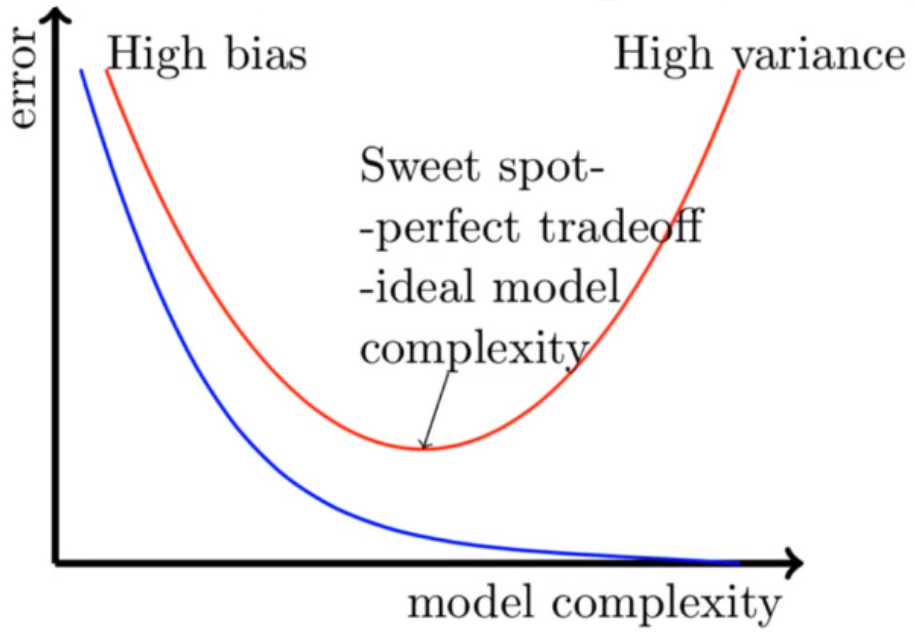
- Occam’s Razor: The simplest explanation is often the best explanation.
- Build the tree which is simples to help us predict the unknown class of the datapoint.

KD trees vs Decision trees

- **Decision trees** are same as **KD trees** except in decision trees you only **split on one dimensions** but you different **splitting criteria**.
  - **KD trees**: split using **median**
  - **Decision trees**: using Gini Index or Entropy to compute the Impurity of the splitting options which are D(# dimensions) * N (# datapoints)

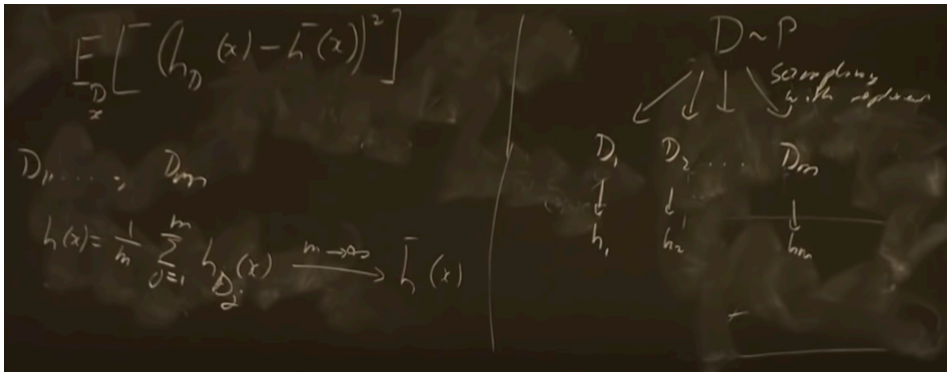
Decision trees Variance and Biased problem

- Using **Bagging** for **variance** problem
- Using **Boosting** for **bias** problem



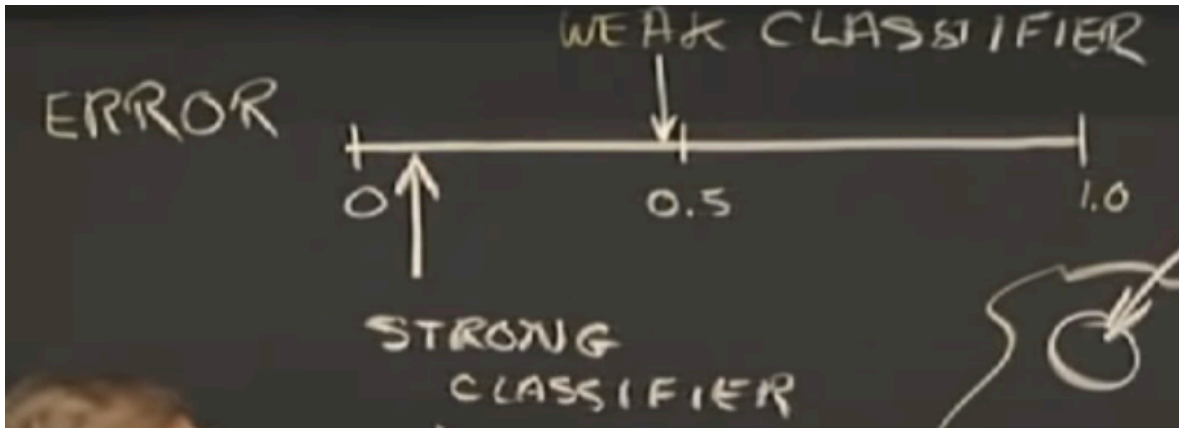
Bagging

- **Bagging:** solving the problem of variance of the dataset when Decision trees doesn’t work.
  - Build **M** dataset from the original dataset by sampling replacement
  - Compute the decision trees for each of the created **M** datasets
  - Take the average of the **M** computed decision trees.



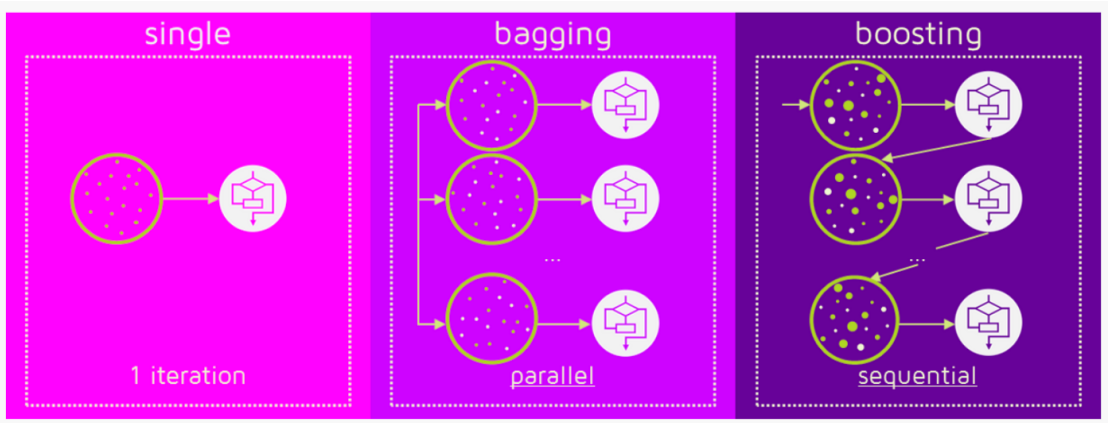
Weak learner vs Strong learner

- **Weak** learners: algorithms that cannot learn training error down to zero
- **Strong** learners: can learn training error down to zero



Boosting

- In a nutshell, we use the number of classifiers to build the strong classifier and we use the **incremental** approach like **gradient descent** to solve the **bias** problem



https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d

https://www.youtube.com/watch?v=UHBmv7qCey4

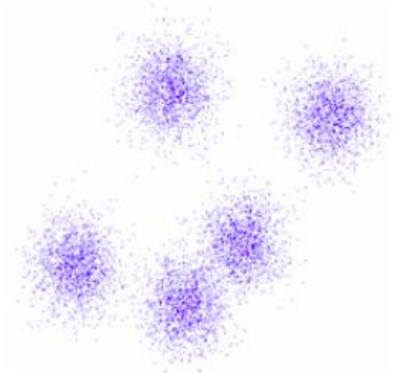
http://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote19.html

Spectral Clustering

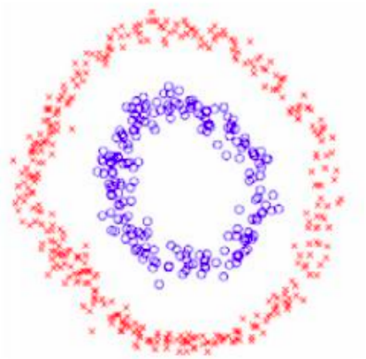
(Spectral clustering) vs (K-mean and Mixture model)

The goal of spectral clustering is to cluster data that is connected but not necessarily compact or clustered within convex boundaries.

- Compactness, e.g., k-means, mixture models
- Connectivity, e.g., spectral clustering



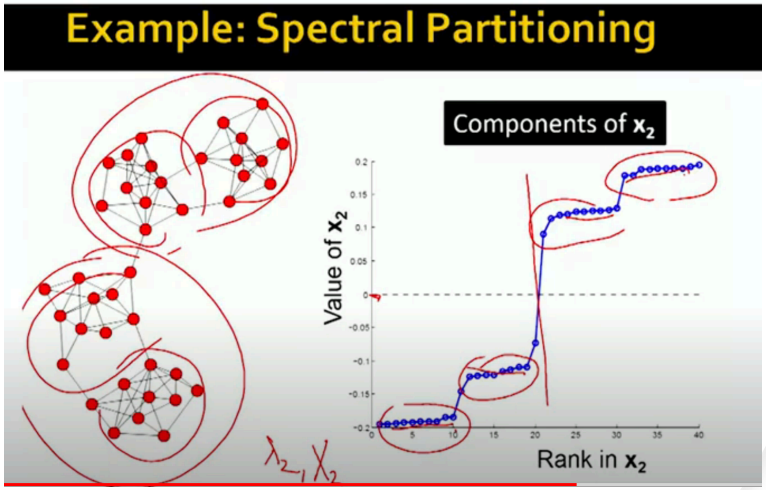
Compactness

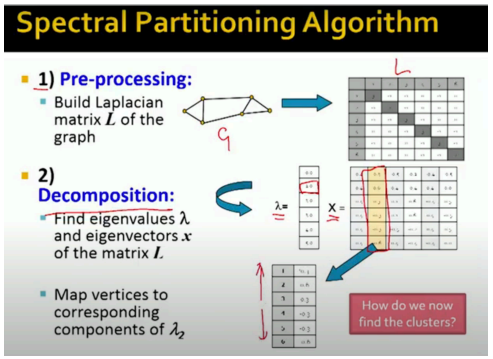


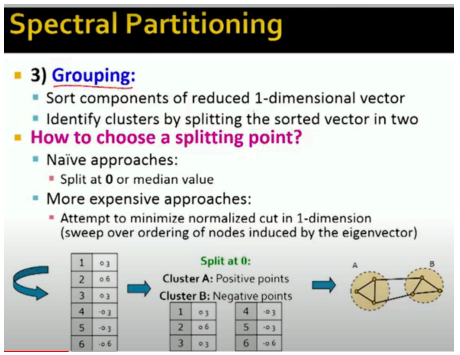
Connectivity

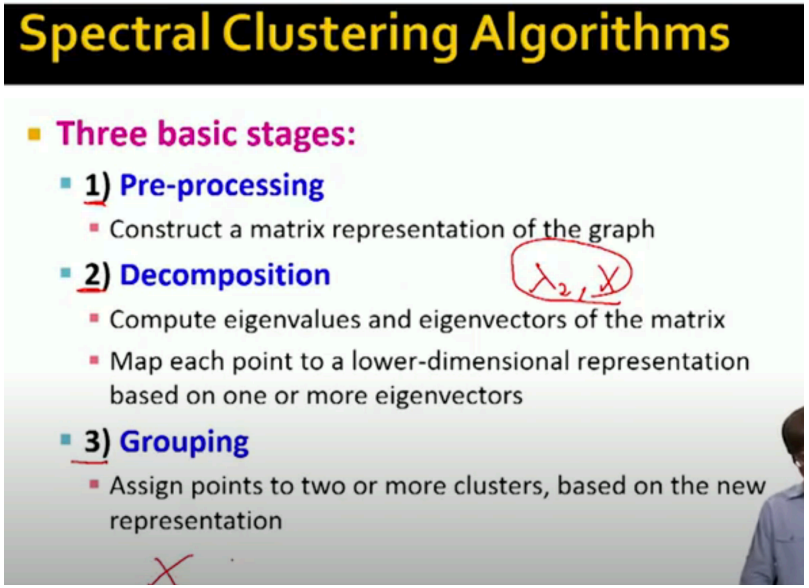
Spectral Clustering

- To partition a graph or datapoints into multiple clusters.









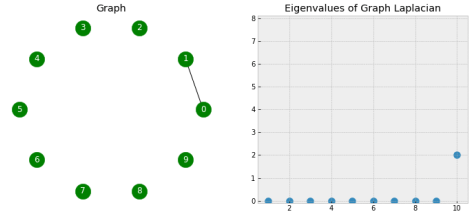
Spectral clustering - (Terminologies)

Adjacency Matrix (A): if the entry in row 0 and column 1 is 1, it would indicate that node 0 is connected to node 1

Affinity Matrix (A): the value for a pair of points expresses how similar those points are to each other. the affinity acts like the weights for the edges on our graph.

Degree Matrix (D): a diagonal matrix, where the degree of a node (i.e. values) of the diagonal is given by the number of edges connected to it. (I think of this as a local (**nearest neighbour**) mean-field average of the Affinity)

Laplacian Matrix (L): Simple Laplacian L=D-A



Graph edit distance

graph edit distance (GED)

GED is a measure of similarity (or dissimilarity) between two graphs.

$$GED(g_1, g_2) = \min_{(e_1,\ldots,e_k)\in \mathcal{P}(g_1,g_2)} \sum_{i=1}^{k} c(e_i)$$

Where $\mathcal{P}(g_1, g_2)$ denotes the set of edit paths transforming g_1 into (a graph isomorphic to) g_2 and $c(e) \geq 0$ is the cost of each graph edit operation e.

GED – graph edit operators

The set of elementary graph edit operators typically includes:

- **vertex insertion** to introduce a single new labelled vertex to a graph.
- **vertex deletion** to remove a single (often disconnected) vertex from a graph.
- **vertex substitution** to change the label (or colour) of a given vertex.
- **edge insertion** to introduce a new coloured edge between a pair of vertices.
- **edge deletion** to remove a single edge between a pair of vertices.
- **edge substitution** to change the label (or colour) of a given edge.

Additional, but less common operators, include operations such as **edge splitting** that introduces a new vertex into an edge (also creating a new edge), and **edge contraction** that eliminates vertices of degree two between edges (of the same colour).

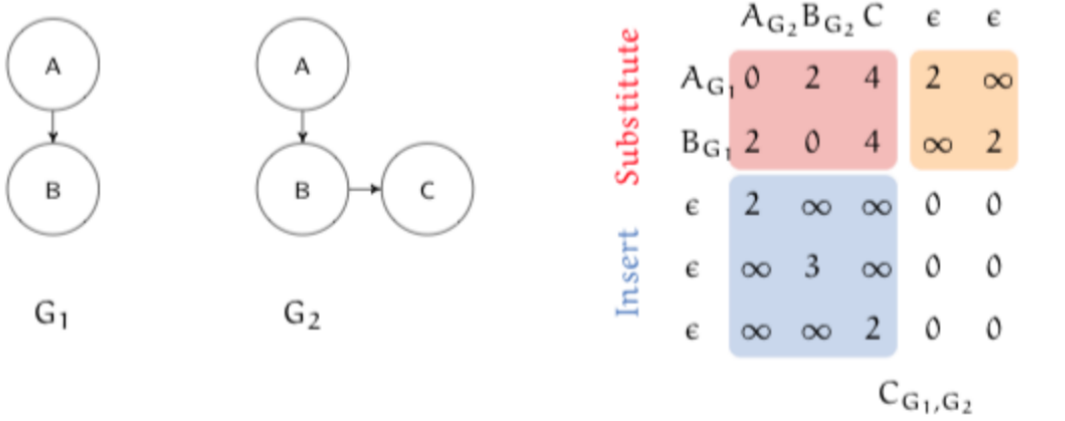
Munkres algorithm

vs

GED

Munkres algorithm

Solve **GED** problem as an algorithm that transpose the graph edit distance to an **Assignment Problem**. (GMatch4py)



https://github.com/Jacobe2169/GMatch4py/issues/7

Munkres algorithm vs GED

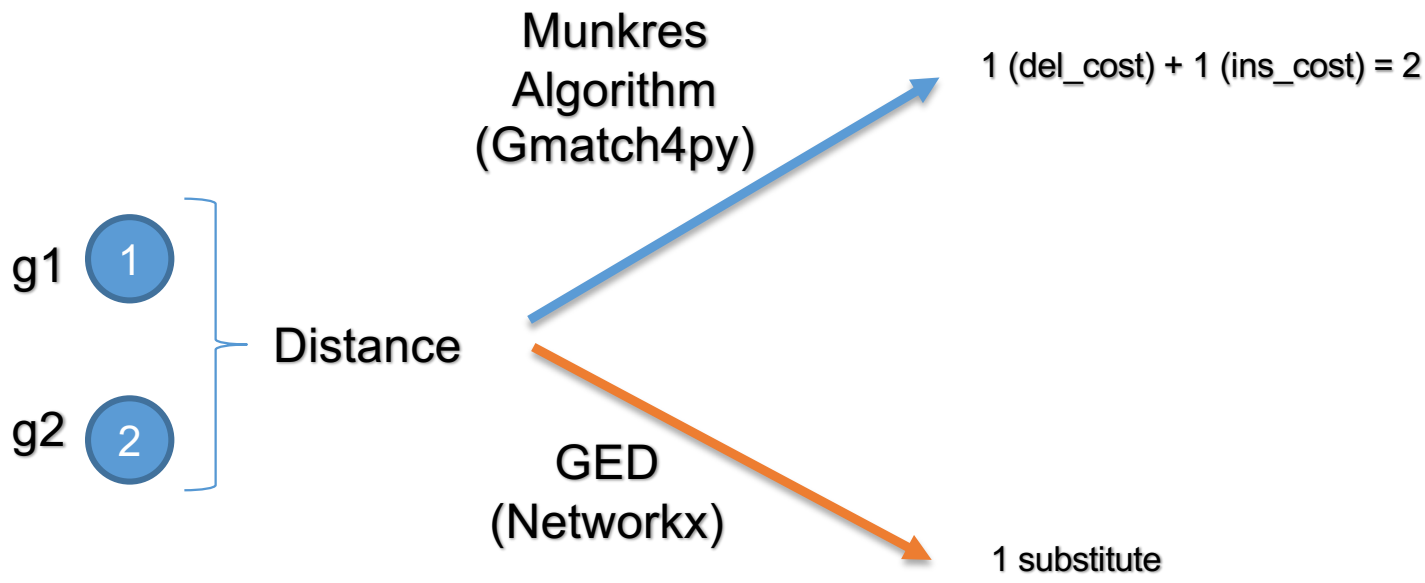
We have Two main difference:

- **Node substitution:** In Munkres algorithm, node substitution or (relabeling) is considered as (delete_cost + insert_cost). However, in GED algorithm node relabeling is considered 0 cost.

- **Edge insertion:** In Munkres algorithm, we only consider **substitute**, **delete** and **insert** nodes to change graph 1 to graph 2. Thus, we cannot add only edge to change graph 1 to graph 2 and we count twice for adding only on edge. (one for the source node of the edge and one for the destination node of the edge)

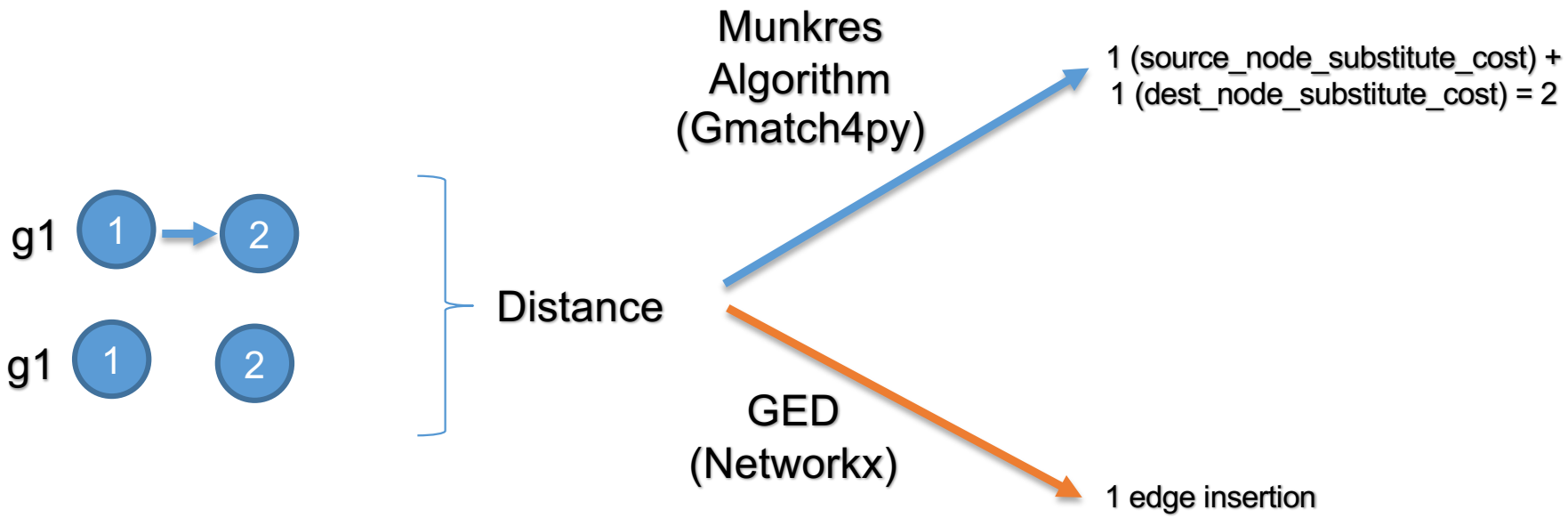
Munkres algorithm vs GED

Node substitution:



Munkres algorithm vs GED

Edge insertion:

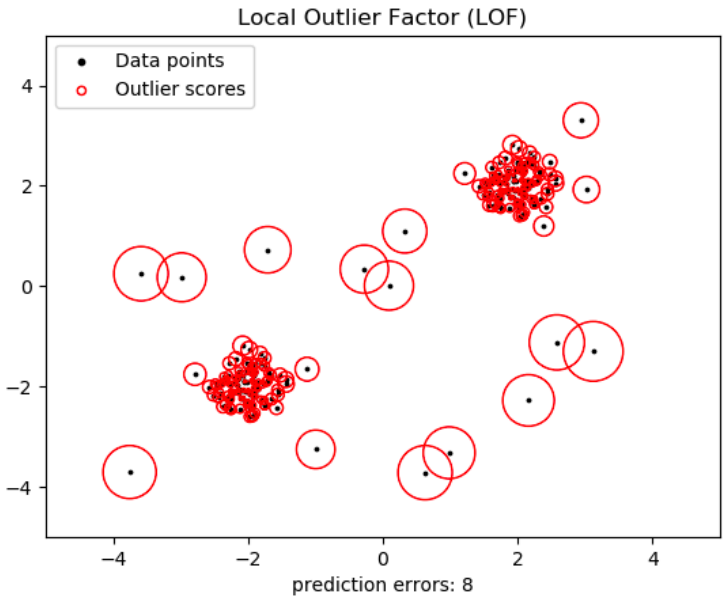


Local Outlier Factor (LOF)

Local Outlier Factor (LOF)

Local Outlier Factor (LOF):

Unsupervised anomaly detection method which computes the local density deviation of a given data point with respect to its neighbours. It considers as outliers the samples that have a substantially lower density than their neighbours.



https://scikit-learn.org/stable/auto_examples/neighbors/plot_lof_outlier_detection.html#:~:text=The%20Local%20Outlier%20Factor%20(LOF,lower%20density%20than%20their%20neighbors.

https://www.dbs.ifi.lmu.de/Publikationen/Papers/LOF.pdf