Introduction to Machine Learning.
CSCI-UA 9473, Lecture 7.

Augustin Cosse

Ecole Normale Supérieure, DMA & NYU
Fondation Sciences Mathématiques de Paris.

2018

Supervised Learning (I)

- Linear regression
  - Bias variance trade-off (Linear and non linear data)
  - Regularization (Ridge, Lasso, Subset Selection)
- Linear classification
  - Separating hyperplane, LDA, logistic regression
  - Perceptron
  - Discriminative vs Generative classifiers
- Non parametric regression/classification
  - Kernel methods
  - Support vector machines
- Neural Networks, convolutional neural networks

Today: Unsupervised Learning

- So far : predictions based on training samples for which joint values $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ are known.
- Problem: costly. Most datasets are not labeled.
- Today: Unsupervised learning = learning without a teacher
- In unsupervised Learning we are given samples $(x_1, x_2, \ldots, x_N)$ from a distribution $P(X)$ and the goal is to infer the properties of the distribution without the help of the teacher.
- One approach is to characterize the values X for which $P(X)$ is large. This includes a variety of methods which attempt to locate low dimensional manifolds within the X space

Samsung AI Forum Offers a Roadmap for the Future of AI

on September 18, 2018

Unsupervised Learning Takes Center Stage

LeCun used training self-driving cars as a key example of unsupervised learning's potential. "A lot of people who are working on autonomous driving are hoping to use reinforcement learning to get cars to learn to drive by themselves by trial and error," said LeCun. "The problem with this is that, because of [reinforcement learning's inherent inefficiencies], you'd have to get a car to drive off a cliff several thousand times before it figures out

"As new unsupervised learning algorithms are discovered, the data efficiency of deep learning will be greatly augmented in the years ahead, and its potential applications in healthcare and other fields will increase rapidly," according to Hinton.

Science Home News Journals Topics Careers

What artificial brains can teach us about how our real brains learn

By Matthew Hutson | Sep. 29, 2017 , 3:10 PM

Q: Why focus on unsupervised learning, which is much less common in AI?

A: With supervised learning, you are assuming that you have a teacher providing the correct label at each learning event. Think about how we humans learn. This very rarely happens.

Alexa is Now Even Smarter-

naturally; and used active learning and unsupervised learning to improve foundational wake word detection, speech recognition, and natural language understanding," said Rohit Prasad, Vice President and Head Scientist, Amazon Alexa. "We've only scratched the surface of A.I.-powered inventions and we'll continue to invent ways to make Alexa more useful for our customers."

7 factors that will push implementation of AI in healthcare

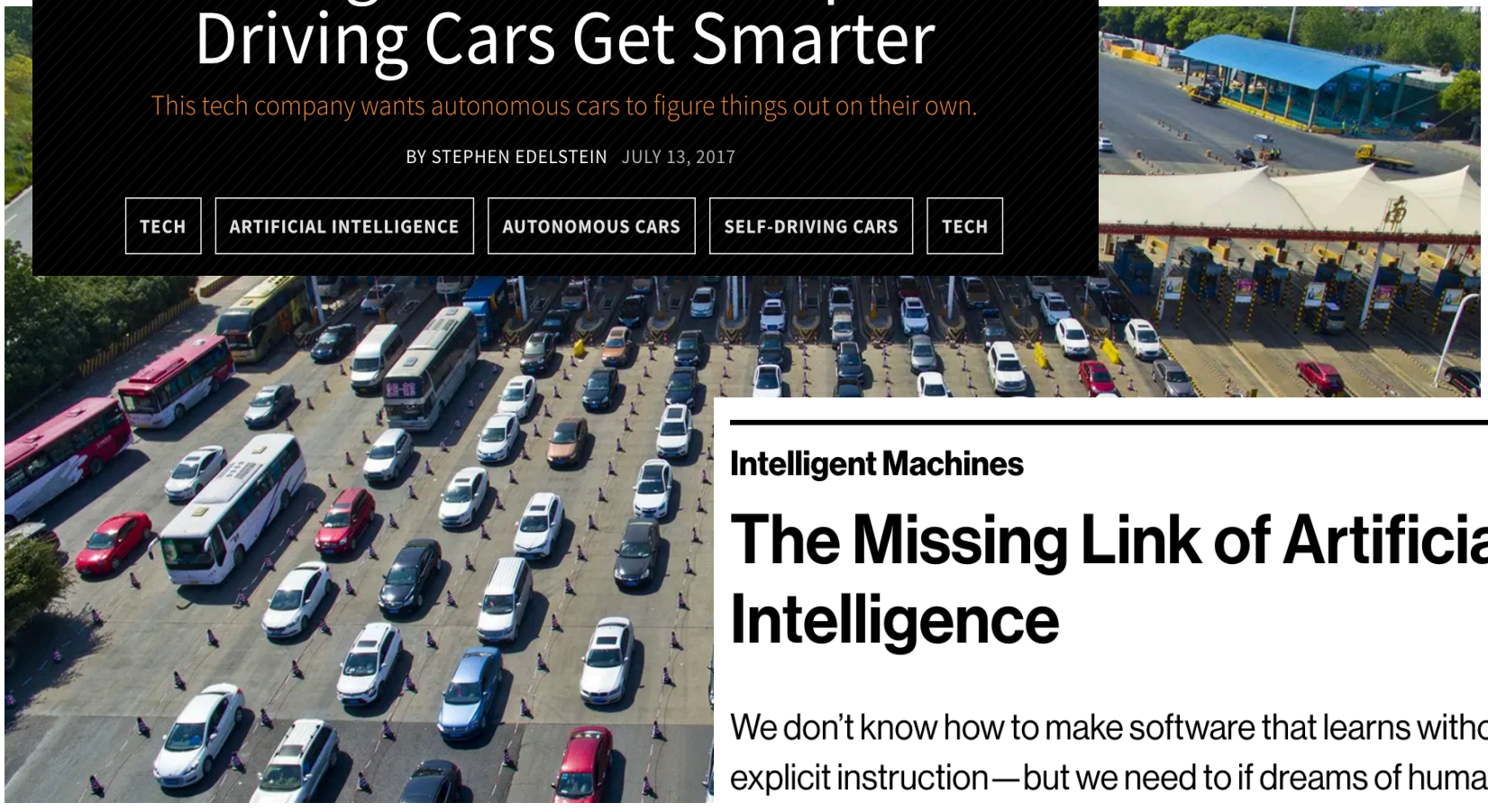
August 31, 2018 | Melissa Rohman | Artificial Intelligence

Cortica Will Apply 'Unsupervised Learning' AI Tech to Help Self-Driving Cars Get Smarter

This tech company wants autonomous cars to figure things out on their own.

BY STEPHEN EDELSTEIN JULY 13, 2017

TECH | ARTIFICIAL INTELLIGENCE | AUTONOMOUS CARS | SELF-DRIVING CARS | TECH



THE DRIVE

MIT Technology Review

Intelligent Machines

The Missing Link of Artificial Intelligence

We don't know how to make software that learns without explicit instruction—but we need to if dreams of humanlike AI are to come true.

by Tom Simonite February 18, 2016

Topics+ The Download Magazine Events More+

Today: Unsupervised Learning

- Finding lower dimensional representation of the data gives an intuition on the association among the variables and whether the variables can be considered as functions of a smaller set of "latent" variables, i.e. $X = X(\theta)$ for some θ.
- Examples include
  - Associations rules,
  - Factor Analysis, including principal and independent component Analysis,
  - Manifold learning methods including Multidimensional scaling, Locally Linear Embedding, Self organizing maps, ..
  - Principal curves,...

Association Rules

- Let us use $X = (X_1, \ldots, X_d)$ to encode the set of items purchased in a store with $X_i = \{0, 1\}$ depending on whether item i is purchased (1) or not (0).
- The idea underlying association rules is to try to find values of the variables X_1 to X_d that frequently appear simultaneously.
- This approach is found in product organization, cross marketing (i.e. sales promotion) or even finance.

Association Rule Analysis

- In Association rule analysis, we want to find subsets s_j of the prototypes that maximize the probability

$$P\left(\cap_{j=1}^{p} X_j \in s_j\right)$$

The intersection $\cap_{j=1}^{p}(X_j \in s_j)$ is called conjunctive rule.

- As an example, the association rule could hep us detect two clusters
  - Cluster 1 is defined from the observation that a (sufficiently large) group of customers always purchase items 1 and 3 simultaneously $((1, 0, 1, 0))$
  - Cluster 2 is defined from the observation that a (sufficiently large) group of customers always purchase items 2 and 4 simultaneously $(X_i = (0, 1, 0, 1))$.

Market Basket Analysis

- General approaches at finding subsets of variable values $(s_1, \ldots, s_p)$ with relatively high occurence are not feasible for large databases
- Instead, we turn to a more tractable algorithm known as Market Basket Analysis
- Market Basket Analysis simplifies the problem by only considering two types of subsets. Either s_j consists of a single value of X_j, or it consists of the entire set of values that X_j can assume $(s_j = \mathcal{S}_j)$
- The problem is then reduced to finding subsets of indices $\mathcal{J}$ and associated values v_{0j} such that the probability

$$P\left(\bigcap_{j \in \mathcal{J}} (X_j = v_{0j})\right)$$

Market Basket Analysis

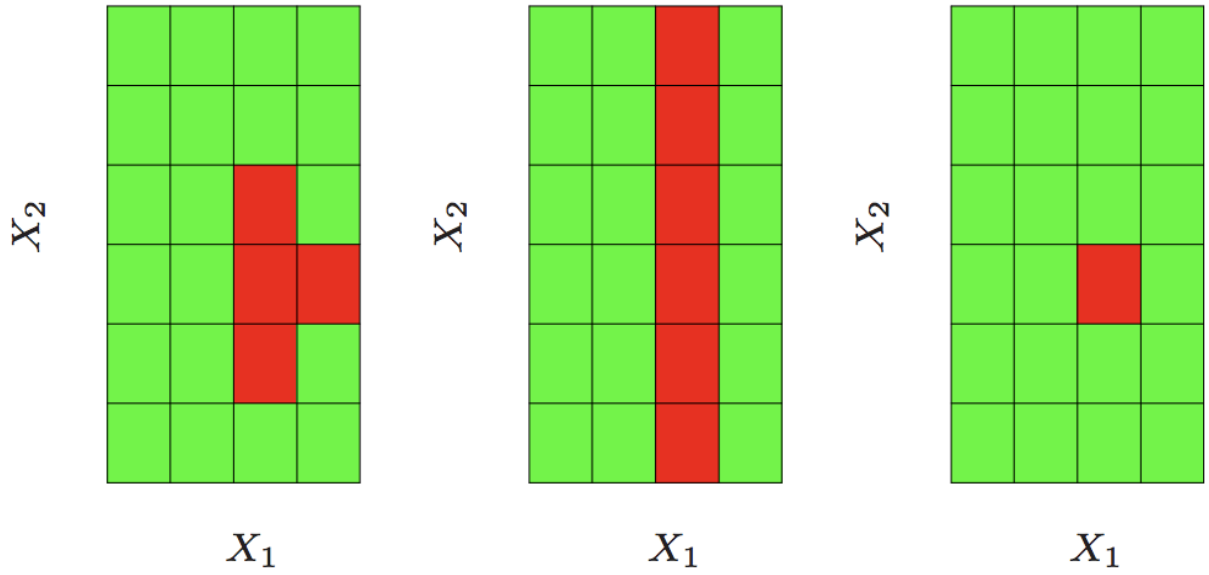


FIGURE 14.1. *Simplifications for association rules. Here there are two inputs X_1 and X_2, taking four and six distinct values, respectively. The red squares indicate areas of high density. To simplify the computations, we assume that the derived subset corresponds to either a single value of an input or all values. With this assumption we could find either the middle or right pattern, but not the left one.*

From HTF, The Elements of Statistical Learning.

Market Basket Analysis

- Market Basket Analysis also relies on the use of dummy variables $Z_k \in \{0, 1\}$ that each represent one possible value of the variables X_ℓ.

- The first value $v_{1,1}$ that X_1 can take is encoded through the variable Z_1 ($Z_1 = 1$ if this value appears and 0 otherwise), and so on for every value and every variable

- We thus have $K = \sum_{j=1}^{p} |\mathcal{S}_j|$ dummy variables and we can try to determine subsets of indices $\mathcal{K}$ for which the probability

$$P\left(\bigcap_{k\in\mathcal{K}} (Z_k = 1)\right) = P\left(\prod_{k\in\mathcal{K}} Z_k = 1\right)$$

  is maximized.

- The set $\mathcal{K}$ is called an item set. Together those two assumptions define the general Market Basket Analysis formulation.

Market Basket Analysis

- Note that two dummy variables Z_k associated to the same variable X_j cannot simultaneously take the value 1
- One way to estimate the probability of occurence of given item sets is to count the number of instances in each subset $\mathcal{K}$, i.e

$$\hat{P}\left(\prod_{k\in\mathcal{K}}(Z_k = 1)\right) = \frac{1}{N}\sum_{i=1}^{N}\prod_{k\in\mathcal{K}} z_{ik}$$

- Where the variable z_{ik} is 1 if the corresponding variable X_k^i in the i^{th} prototype $X^i = (X_1^i, \ldots, X_d^i)$ takes the value encoded by this z_{ik}.

Market Basket Analysis

- Given a model for the probability P, the clusters are defined by setting a threshold $\{\mathcal{K}_\ell \mid T(\mathcal{K}_\ell) > t\}$
- How do we solve the Market Basket Analysis problem in practice? One possibility is the A priori algorithm
  - Start with single item sets and discard the sets for which the support is less than the threshold
  - As a second step, form all subsets of size 2 that can be formed by pairing subsets of size 1. Then discard those size 2 subsets that are less than the threshold.
  - In other words, forming subsets $\mathcal{K}$ of size M, only requires looking at the points whose size $(M-1)$ ancestors all share the same value (i.e are part of a same subset)
  - In terms of computational complexity, the A priori algorithm requires one pass over the data

From the a priori algorithm to association rules

- Once the A priori algorithm has returned high support item sets $\mathcal{K}$, Association Rule Analysis partition each set $\mathcal{K}$ into disjoint subsets A and B from which it defines association rules of the form $A \Rightarrow B$ where A is called antecedent and B is called the consequent.
- Association rules are ways of studying how much the purchase of one item influence the purchase of another item.

From the a priori algorithm to association rules

- To characterize those association rules, we introduce 3 quantities,
  - The support $T(A \Rightarrow B)$ is the fraction of items in the union of antecedent and the consequent (that is the number of item in the item set $\mathcal{K}$)
  - The confidence or predictability $C(A \Rightarrow B)$ is its support divided by its antecedent.

    $$C(A \Rightarrow B) = \frac{T(A \Rightarrow B)}{T(A)}$$

    The confidence $C(A \Rightarrow B)$ can be considered as an estimate of the probability $P(B|A)$
  - The Lift of the rule $A \Rightarrow B$ is defined as the ratio of the confidence over the expected confidence

    $$L(A \Rightarrow B) = \frac{C(A \Rightarrow B)}{T(B)}$$

From the a priori algorithm to association rules

- The expected confidence $T(B)$ can be computed as $T(B) = Pr(\prod_{k \in A} Z_k = 1)$ and is an estimate of the unconditional probability $P(B)$.
- The Lift can be considered as an estimate of the dependence/association measure $Pr(A \cap B)/P(A)P(B)$
- Once the item sets $\mathcal{K}$ with a sufficiently large support have been returned by the A priori algorithm, a confidence threshold c is set and all the association rules with confidence above the threshold are formed, i.e.

$$\{A \Rightarrow B \mid C(A \Rightarrow B) > c\}$$

From the a priori algorithm to association rules

- The output of the Analysis is thus a set of association rules satisfying the constraints

$$T(A \Rightarrow B) > t \quad \text{and} \quad C(A \Rightarrow B) > c$$

- Once the analysis is completed the results are stored in a database which can be accessed to receive specific information on particular items
- Most often we will be interested in retrieving all the transactions (i.e prototypes) in which one particular item appeared as consequent
- The analysis will then indicate the antecedents which might reveal valuable in predicting future sales for the consequent.

Clustering Algorithms

- There are three main types of clustering algorithms (HTF)
  - Combinatorial algorithms (Work directly on the data without assuming an underlying probability distribution)
  - Mixture Models (Assume that the prototypes are i.i.d. samples from some probability distribution. The probability distribution is assumed to be a mixture of simpler densities, each one of the corresponding to the distribution of a cluster. The distribution is fit to the data through MLE)
  - The last type, called Mode seekers or Bump hunting algorithms try to estimate the modes of the distribution (and hence the clusters) direclty from the data (non-parametric)

K-means and K-medoid

- The most popular clustering algorithms are combinatorial algorithms which assign every observation to a given cluster without regard to any predefined probabilistic model.
- The number of clusters K is usually predefined
- One approach is to introduce a loss that will drive the assignment. If we let $\mathcal{C}_k$ to denote the k^{th} cluster, we get

$$\ell(\mathcal{C}) = \frac{1}{2} \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \sum_{j \in \mathcal{C}_k} d(x_i, x_j)$$

K-means

- When the dissimilarity is chosen to be the Euclidean distance,

$$d(x_i, x_i') = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = \|x_i - x_{i'}\|^2$$

- The loss then reads as

$$\ell(\mathcal{C}) = \frac{1}{2} \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \sum_{j \in \mathcal{C}_k} \|x_i - x_j\|^2$$

K-means

- In particular, developing, we get

$$\frac{1}{2}\sum_{k}\sum_{i\in\mathcal{C}_k}\sum_{j\in\mathcal{C}_k}\|x_i - x_j\|^2$$

$$= \frac{1}{2}\sum_{k}\sum_{i\in\mathcal{C}_k}\sum_{j\in\mathcal{C}_k}\langle x_i, x_i\rangle + \langle x_j, x_j\rangle - 2\langle x_i, x_j\rangle$$

$$= \frac{1}{2}\sum_{k}\sum_{i\in\mathcal{C}_k} 2\langle x_i, x_i\rangle N_k - \frac{1}{2}\sum_{k}\sum_{i\in\mathcal{C}_k} 2\langle x_i, \sum_{j\in\mathcal{C}_k} x_j\rangle$$

$$= \left(\sum_{k=1}^{K} N_k \sum_{i\in\mathcal{C}_k}\langle x_i, x_i\rangle - \sum_{k=1}^{K} N_k \sum_{i\in\mathcal{C}_k}\langle x_i, \sum_{j\in\mathcal{C}_k} x_j\rangle \frac{1}{N_k}\right)$$

K-means

$$\frac{1}{2}\sum_{k}\sum_{i\in\mathcal{C}_k}\sum_{j\in\mathcal{C}_k}\|x_i - x_j\|^2$$

$$= \left(\sum_{k=1}^{K} N_k \sum_{i\in\mathcal{C}_k}\langle x_i, x_i\rangle - \sum_{k=1}^{K} N_k \sum_{i\in\mathcal{C}_k}\langle x_i, \sum_{j\in\mathcal{C}_k} x_j\rangle \frac{1}{N_k}\right)$$

$$= \sum_{k=1}^{K} N_k \sum_{i\in\mathcal{C}_k}\left(\langle x_i, x_i\rangle - \langle x_i, \sum_{j\in\mathcal{C}_k} x_j\rangle \frac{1}{N_k}\right)$$

$$= \sum_{k=1}^{K} N_k \sum_{i\in\mathcal{C}_k}\left(\langle x_i, x_i\rangle - \langle x_i, \sum_{j\in\mathcal{C}_k} x_j\rangle \frac{1}{N_k} + \frac{1}{N_k^2}\langle \sum_{j\in\mathcal{C}_k} x_j, \sum_{j\in\mathcal{C}_k} x_j\rangle\right)$$

$$- \sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}\left(\frac{1}{N_k}\sum_{j\in\mathcal{C}_k}\langle x_j, x_i\rangle\right)$$

K-means

$$\frac{1}{2}\sum_{k}\sum_{i\in\mathcal{C}_k}\sum_{j\in\mathcal{C}_k}\|x_i - x_j\|^2$$

$$= \sum_{k=1}^{K} N_k \sum_{i\in\mathcal{C}_k}\left(\langle x_i, x_i\rangle - \langle x_i, \sum_{j\in\mathcal{C}_k} x_j\rangle\frac{1}{N_k} + \frac{1}{N_k^2}\langle\sum_{j\in\mathcal{C}_k} x_j, \sum_{j\in\mathcal{C}_k} x_j\rangle\right)$$

$$- \sum_{k=1}^{K} N_k \sum_{i\in\mathcal{C}_k}\left(\frac{1}{N_k}\sum_{j\in\mathcal{C}_k}\langle x_j, x_i\rangle\right)$$

$$= \sum_{k=1}^{K} N_k \sum_{i\in\mathcal{C}_k}\left(\langle x_i, x_i\rangle - 2\langle x_i, \frac{1}{N_k}\sum_{j\in\mathcal{C}_k} x_j\rangle + \langle\frac{1}{N_k}\sum_{j\in\mathcal{C}_k} x_j, \frac{1}{N_k}\sum_{j\in\mathcal{C}_k} x_j\rangle\right)$$

$$= \sum_{k=1}^{K} N_k \sum_{i\in\mathcal{C}_k}\|x_i - \frac{1}{N_k}\sum_{j\in\mathcal{C}_k} x_j\|^2$$

K-means

- In other words, when using the Euclidean distance, one can write the clustering objective/loss as

$$\ell(\mathcal{C}) = \sum_{k=1}^{K} N_k \sum_{i \in \mathcal{C}_k} \|x_i - \overline{x}^k\|^2$$

  Where $\overline{x}^k$ is the center of mass of the k^{th} cluster.

- the optimal clustering in that framework is thus the clustering that minimizes the average dissimilarity

K-means

- Given a set of centers $\bar{x}_k$, the optimal clustering is thus defined by solving the minimization

$$\mathcal{C}^* = \min_{\mathcal{C}} \sum_{k=1}^{K} N_k \sum_{i \in \mathcal{C}_k} \|x_i - \overline{x}_k\|^2$$

- On the other hand, for a subset of observations x_S, the center of mass can also read as a minimization

$$x_S = \operatorname*{argmin}_m \sum_{i \in S} \|x_i - m\|^2$$

- We can thus solve the clustering problem by iterating between the two steps. This gives the Kmeans algorithm.

K-means

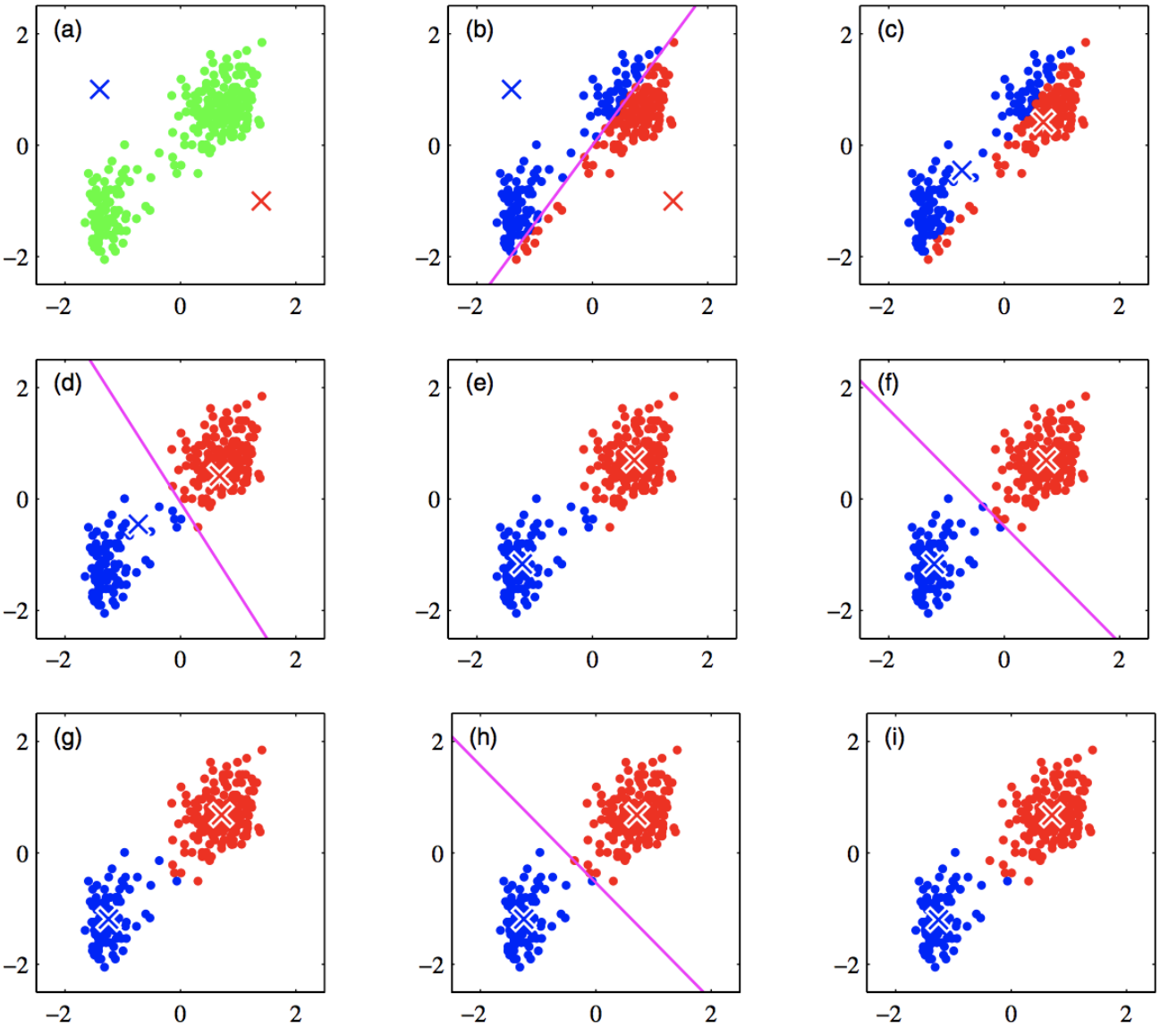
- **1.** Given a cluster assignement $\mathcal{C}$, compute the center of mass of each cluster

$$\bar{x}_S = \underset{m}{\operatorname{argmin}} \sum_{i \in S} \|x_i - m\|^2$$

- **2.** Update the assignement by setting

$$x_i \in \mathcal{C}_k \quad \text{if} \quad k = \underset{k}{\operatorname{argmin}} \|x_i - m_k\|^2$$

- Repeat Steps 1 and 2 until the assignement does not change



source: Bishop, Pattern Recognition and Machine Learning.

K-means and K-medoid

- The use of the Euclidean distance provides an easy way to compute the center of each cluster.
- It is however possible to get an extension to general more general similarity by using explicit optimization.
- This gives the K-medoid algorithm which iterates over the two steps
  - For a given assignement, find the points that are minimizing the distances to any other points in the cluster (the centroid is thus taken as one of the prototypes)

$$c_k \leftarrow \underset{x_i \in \mathcal{C}_k}{\operatorname{argmin}} \sum_{j \in \mathcal{C}_k} d(x_i, x_j)$$

  - For a given set of centroids, c_k, $k = 1, \ldots, K$, assign each points to its closest centroid,

$$x_\ell \in \mathcal{C}_i \quad \text{with} \quad i = \underset{1 \leq k \leq K}{\operatorname{argmin}}\, d(x_i, c_k)$$

K-means and K-medoid

- Because it relies on the Euclidean distance, K means is often much more sensitive to outliers (i.e it places the highest influence on the largest distances)
- Conversely, the number of operations needed to derive the centroid in K-means was only $\mathcal{O}(N_k)$. For K-medoid, it is now $\mathcal{O}(N_k^2)$ (i.e. comparing every pair of points in the clusters).
- K-means and K-medoid both suffer from several drawbacks among which
  - Both methods assume the number of clusters to be known beforehand (not true in practice)
  - As iterative techniques, they are both sensitive to initial conditions
  - Finally, both K-means and K-medoid converge to local minimas.

K-means and K-medoid: Initialization

- Random partitioning. The approach divides the dataset in K distinct clusters chosen at random.
- The Forgy or *Lloyd-Forgy* approach (FA). The method picks K feature vectors at random to define the centroids and assigns remaining feature vectors to their nearest centroid.
- MacQueeen's approach. Instead of considering a batch algorithm where the whole dataset if used to update the centroids, the MacQueeen approach is the extension of the Forgy approach to a recalculation of the centroids after each assignment.
- Kaufman's approach. The Kauffman initialization is a deterministic initialization which places the centroids in the areas where there is a higher density of prototypes.

MacQueen's approach

- Initialize the centroids, m_k and set the cluster sizes to zero, $n_k = 0$, for all $k = 1, \ldots, K$.
- Repeat
  - Pick an observation $\boldsymbol{x}_i$ and determine the cluster following

$$k = \underset{x \quad \ell}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{m}_k\|^2$$

  - Update the centroid $\boldsymbol{m}_k$ following

$$\boldsymbol{m}_k \leftarrow \boldsymbol{m}_k + \frac{1}{n_k}(\boldsymbol{x}_i - \boldsymbol{m}_k)$$

    where n_k is the number of prototypes in cluster k.

Kauffman's approach

- Set $\overline{\boldsymbol{x}}_0$ to be the "median" feature vector (i.e the one that is the most centrally located)
- Set $\mathcal{S} = \{\overline{\boldsymbol{x}}_0\}$ and let $\mathcal{S}^c$ denote the set of remaining feature vectors.
- For all the remaining feature vectors $\boldsymbol{x}_i \in \mathcal{S}^c$,
  - compute $\overline{d}_i \equiv \min_{k \in \mathcal{S}} d(\overline{\boldsymbol{x}}_k, \boldsymbol{x}_i)$
  - Set $C_{i,j} \equiv \max(\overline{d}_i - d(\boldsymbol{x}_j, \boldsymbol{x}_i), 0)$
- Choose the next centroid $\overline{\boldsymbol{x}}_{k+1}$ from the index j that solves $\max_i C_{ij}$ (The result will be a prototype with high density of points around him (i.e the prototype with the largest number of points closer to him than to the previous centroids))
- If there are K points in $\mathcal{S}$ stop. Otherwise set $\mathcal{S} \leftarrow \mathcal{S} \cup \overline{x}_{k+1}$ and go back to step 1.

Hierarchical clustering (I)

- As we saw, an inconvenient aspect of K-means or K-medoid is that those methods require the user to explicitely provide a number of clusters.
- Hierarchical clustering does not require such initialization. Instead it only relies on a measure of dissimilarity between groups of observations
- The algorithm then iteratively defines the clusters by either merging (bottom up) or dividing (top down) the set of measurements
  - Agglomerative strategies start at the bottom (n clusters) and successively merge a selected pair of clusters ($n-1$ clusters)
  - Divisive approaches start at the top (1 cluster) and succesively split the previous clusters into two new clusters.
- It is then up to the user to decide which level represents the most natural clustering.

Hierarchical clustering (II)

- Agglomerative and divisive Hierarchical Clustering approaches can be defined to possess a monotonicity property which means that the dissimilarity between the clusters that are merged increase monotonically with the level in the hierarchy.
- Recursive splitting/agglomeration happening in hierarchical clustering can be represented by a tree (a.k.a dendogram) in which the nodes represent the clusters. The root node encode the whole dataset and the leafs represent each of the prototypes.
- Furthermore, when the clustering approach satisfies the monotonicity property, the height in the tree represents the dissimilarity between merged clusters.

Hierarchical clustering (III)

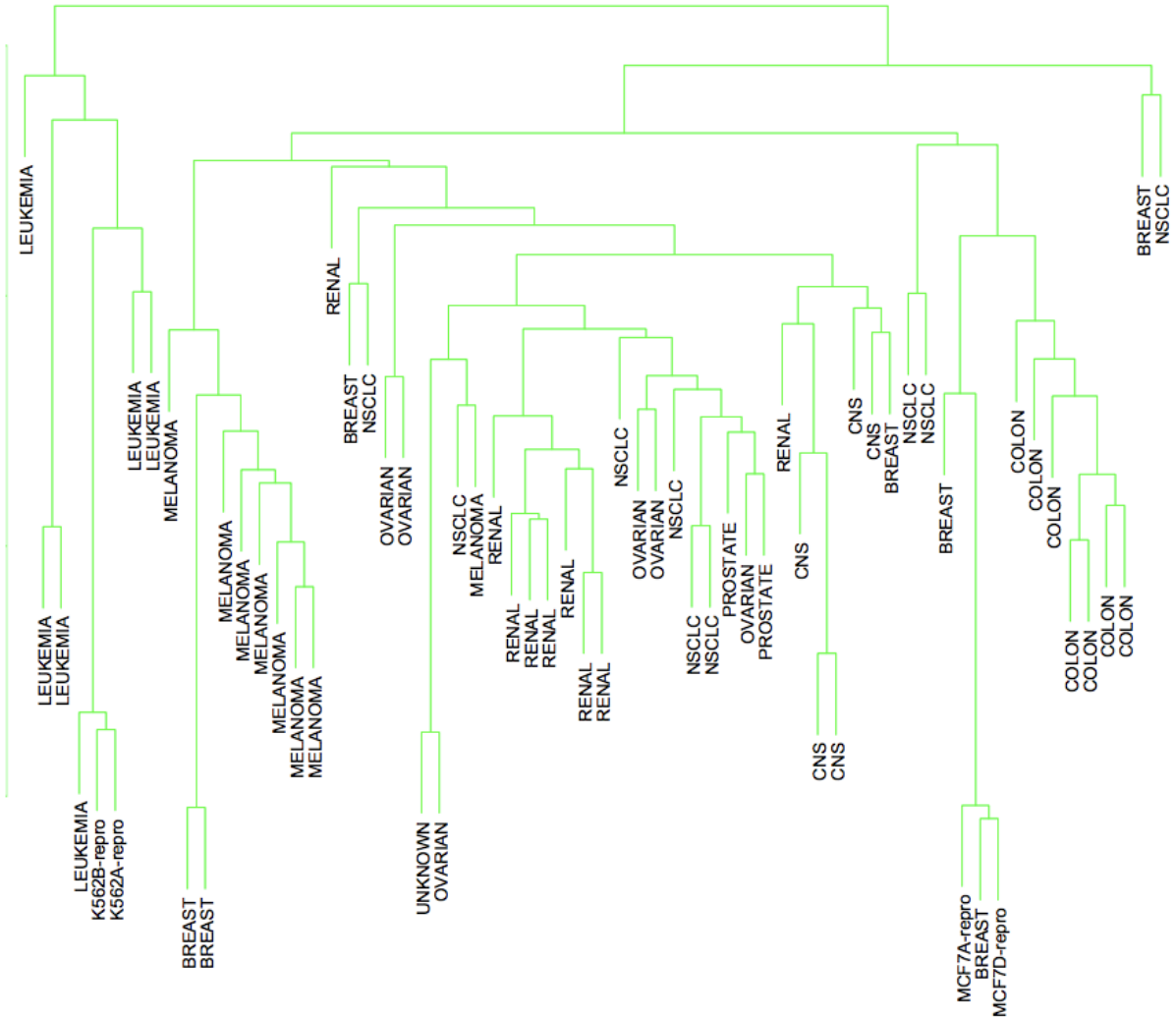
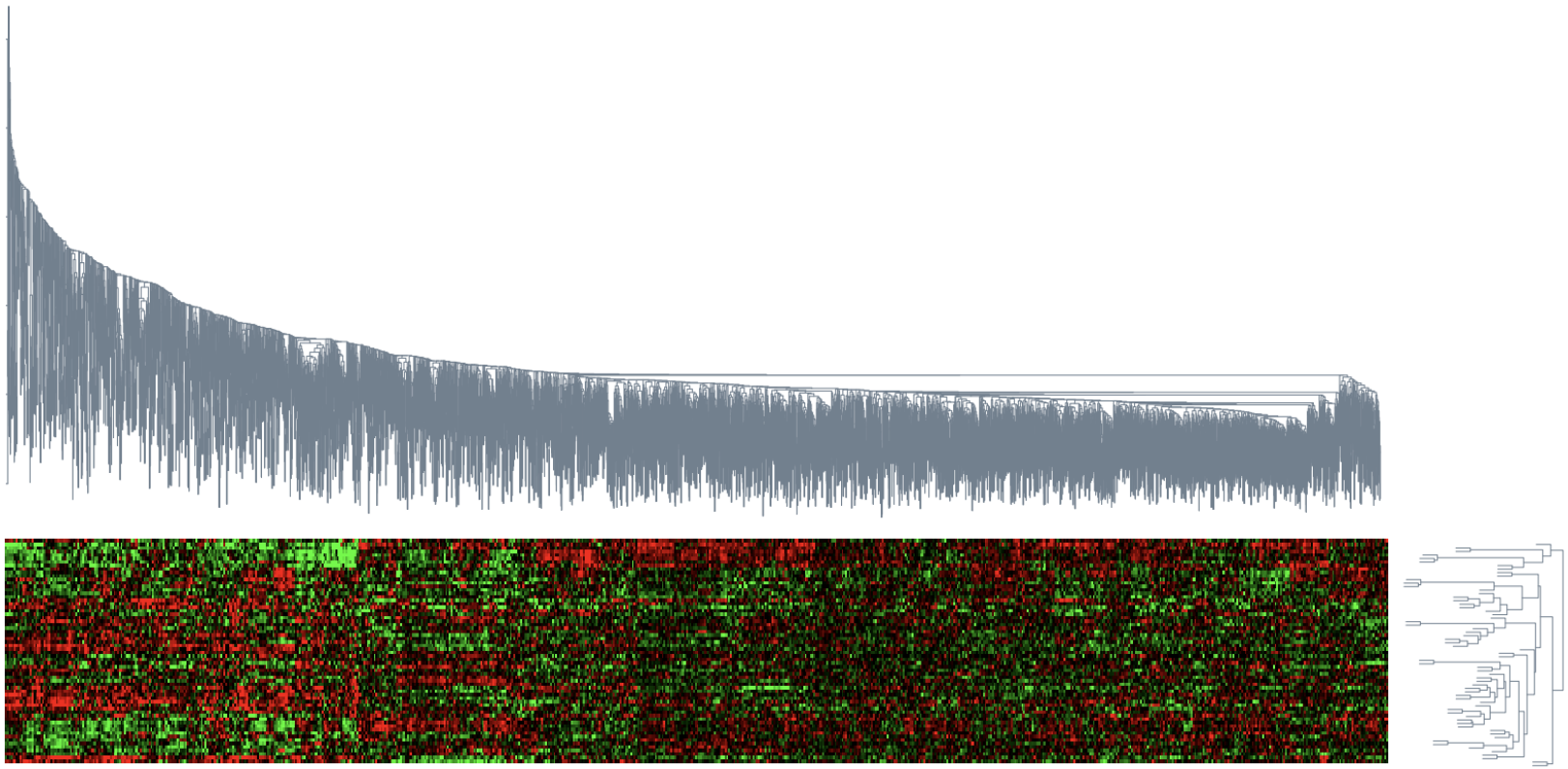


FIGURE 14.12. *Dendrogram from agglomerative hierarchical clustering with average linkage to the human tumor microarray data.*

from H.,T.,F., The elements of Statistical Learning

Hierarchical clustering (IV)



from H.,T.,F., The elements of Statistical Learning

Agglomerative Clustering

- Agglomerative clustering starts with every prototype representing a singleton.
- At each step, it then chooses the closest two clusters and merge them into a single cluster.
- For any two clusters H and G, there are three common ways to define the dissimilarity between those clusters

Agglomerative Clustering

- In Single linkage (SL), the dissimilarity between two clusters H and G is defined from the closest pair of points

$$d_{SL}(G, H) = \min_{i \in G, j \in H} d(i, j)$$

- In Complete Linkage (CL), the inter-cluster dissimilarity is defined from the dissimilarity of the furthest pair,

$$d_{CL}(G, H) = \max_{i \in G, j \in H} d(i, j)$$

- Finally, in Group Average clustering (GA), the criterion is the average dissimilarity

$$d_{GA}(G, H) = \frac{1}{N_G N_H} \sum_{i \in G} \sum_{j \in H} d(i, j)$$

A couple of observations (I)

- Single linkage only requires the dissimilarity to be small for one pair of prototypes from each clusters
- It therefore leads to clusters that violates the compactness assumption (i.e clusters are compact if all the observations within them are relatively close to each other)
- If we define the diameter of a group of observations as

$$D_G = \max_{i,j \in G} d(i,j),$$

single linkage can produce large diameters clusters

A couple of observations (II)

- On the opposite, complete linkage will generate compact clusters with small diameters
- However CL can violate the closeness property. Two points from one cluster can be closer to prototypes from another cluster than they are to the prototypes in their own cluster.
- E.g. Assume C_1 and C_2 where C_2 has a single prototype that is very far from the points in C_1 and all the others which are very close (i.e closer than the closest distance between prototypes in C_1)

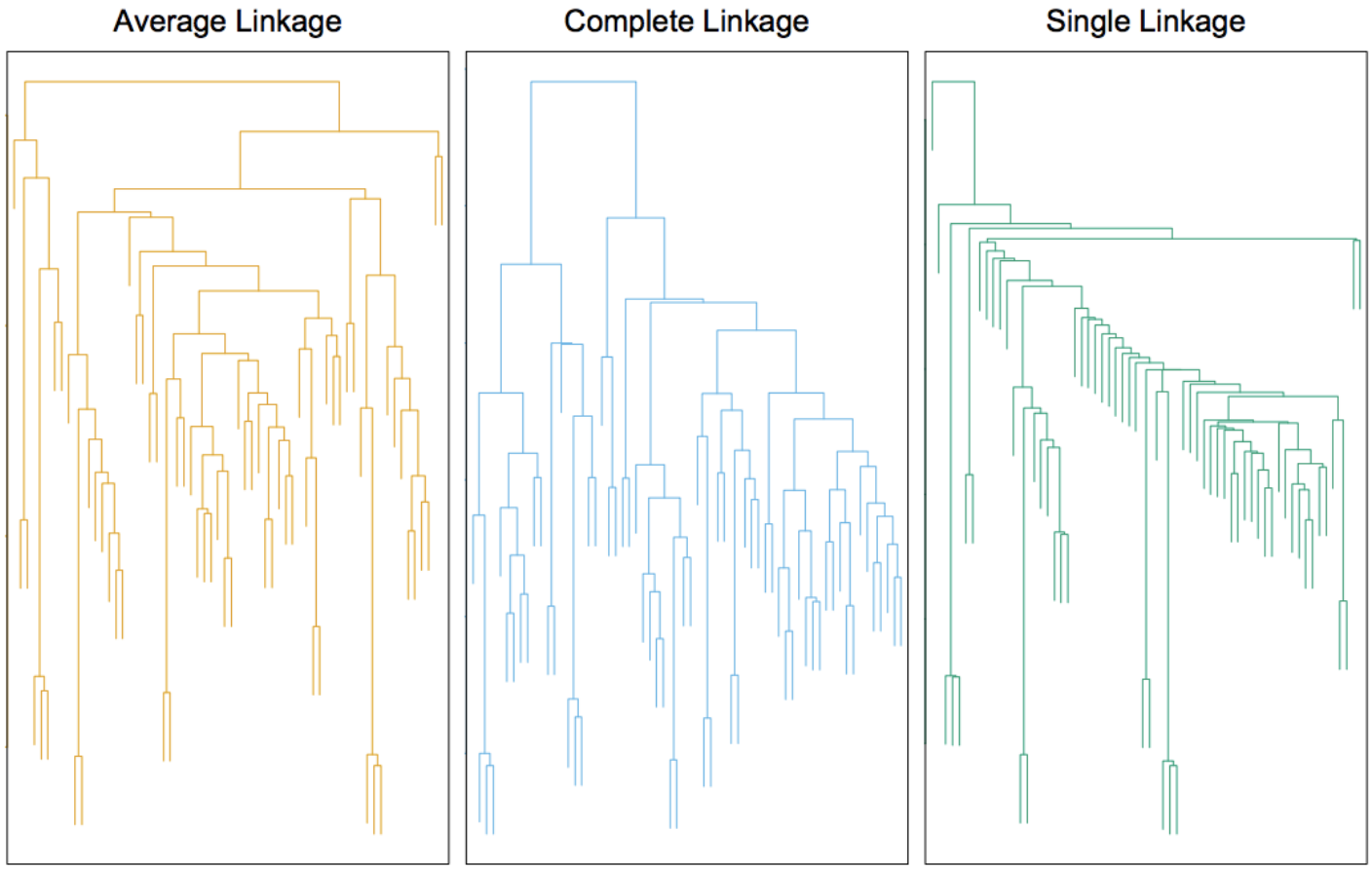


FIGURE 14.13. *Dendrograms from agglomerative hierarchical clustering of human tumor microarray data.*

from HTF, The elements of Statistical Learning

How about divisive clustering?

- One possibility is to use a K-means algorithm (with $K=2$) to split one cluster at each step. But this approach would depend on the initialization of $K-means$ and does not satisfy the monotonicity property

- One alternative is the Macnaughton-Smith algorithm. Starting from a single cluster $\mathcal{C}_0$ with $|\mathcal{C}_0| = n$, remove that prototype x_1 which has the largest average dissimilarity,

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\text{argmax}}\ \overline{D_0}(\boldsymbol{y}) = \underset{\boldsymbol{y}}{\text{argmax}}\ \frac{1}{n-1} \sum_{\boldsymbol{x}_i \in \mathcal{C}_0} d(\boldsymbol{x}_i, \boldsymbol{y})$$

Then define $\mathcal{C}_1 = \{\boldsymbol{y}^*\}$, $\mathcal{C}_0 = \mathcal{C}_0 \setminus \{\boldsymbol{y}^*\}$. For the second step, remove the prototype $\boldsymbol{y} \in \mathcal{C}_0$ that maximizes the difference

$$D_{C_0} - D_{C_1} = \frac{1}{|C_0| - 1} \sum_{\boldsymbol{x}_j \in C_0} d(\boldsymbol{y}, \boldsymbol{x}_j) - \frac{1}{|C_1|} \sum_{\boldsymbol{x}_j \in C_1} d(\boldsymbol{x}_j, \boldsymbol{y})$$

How about divisive clustering?

- Other alternatives include Kaufman and Rousseeuw: choose at each step the cluster that maximizes the diameter and split this particular cluster
- Largest average dissimilarity: split the cluster that maximizes

$$\bar{d}_G = \frac{1}{N_G^2} \sum_{\boldsymbol{x}_i \in G} \sum_{\boldsymbol{x}_j \in G} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- In any of these approaches, recursive splitting is then applied until each cluster is singleton or members of the cluster have zero dissimilarity.

Spectral clustering I

- Given N prototypes $\boldsymbol{x}_i$, $i = 1, \ldots, N$, spectral clustering starts with the $N \times N$ matrix encoding the pairwise similarities between points, $s(\boldsymbol{x}_i, \boldsymbol{x}_j)$
- Spectral clustering might be particularly interesting on non convex data where traditional clustering methods such as K-means might underperform
- The data is then represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{E}$ denotes the set of edges and $\mathcal{V}$ denotes the set of vertices.
- Each edge w_{ij} in the graph is given the weight $s(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Spectral clustering II

- The whole idea of spectral clustering is to partition the graph into clusters, such that edges between clusters have low weights and edges inside each cluster have higher weights
- Given a set of prototypes $\boldsymbol{x}_i$, let $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to denote the Euclidean distance between those prototypes.
- We can define the similarity from the Radial kernel $s(\boldsymbol{x}, \boldsymbol{x}_j) = \exp(-d(\boldsymbol{x}_i, \boldsymbol{x}_j)\gamma)$ where $\gamma > 0$ is a scale parameter.
- Given the similarity matrix, one can either keep all interactions into account, or only retain those interactions corresponding to the K nearest neighbors

Spectral clustering III

- The matrix, $\boldsymbol{A}_{ij}$ of edge weights, $\boldsymbol{A}_{ij} = w_{i,j}$ is called the adjacency matrix of the graph. We call degree of a vertex i, the sum $d_i = \sum_j A_{i,j}$.
- If we define the diagonal matrix $\boldsymbol{D}$ encoding the degrees as

$$\boldsymbol{D} = \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_N \end{bmatrix}$$

  Then the (unnormalized) Laplacian of the graph is defined as the matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$
- One can equivalently work with the normalized graph Laplacian $\tilde{\boldsymbol{L}} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A}$

Spectral clustering IV

- Spectral clustering then works by first computing the eigendecomposition of the graph Laplacian
- It then selects the m smallest eigenvalues and their corresponding eigenvectors
- This gives a matrix $\boldsymbol{V}$ of size $N \times m$ on which one can apply traditional clustering algorithms such as K-means to find a clustering of the original set of prototypes
- Finding the smallest eigenpair implies that the representation given by $\boldsymbol{V}$ will preserve the similarity encoded in the w_{ij}.

$$\begin{aligned}\boldsymbol{v}^T \boldsymbol{L}\boldsymbol{v} &= \sum_{i=1}^{N} d_i v_i^2 - \sum_{i=1}^{N}\sum_{j=1}^{N} v_i v_j w_{ij} \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(v_i - v_j)^2\end{aligned}$$