Lec 12

Thursday, October 24, 2019 10:54

Clustering using (dis)similarity measures

Both k-means & EM work on vector data
What if we only have dissimilarity data?

K-medoids

Medoid is the pt that's most similar to all other pts
(cf. mean is pt closest in sq dist to all other pts)

pts: $1, \dots, m$ dissimilarities: d_{ij}

medoid of these pts is

$$i^* = \underset{i=1,\dots,m}{\operatorname{argmin}} \sum_{j=1}^{m} d_{ij}$$

K-medoids is like k-means but finds clusters s.t. within each cluster all pts are similar to their cluster's medoid.

Start w/ some $C: \{1, \dots, n\} \to \{1, \dots, k\}$

1. Find the medoid for each cluster

$$i_j^* = \underset{i: C(i)=j}{\operatorname{argmin}} \sum_{i': C(i')=j} d_{ii'}$$

2. Assign each observation to its most similar medoid

$$c(i) = \underset{j=1,\dots,k}{\operatorname{argmin}} \; d_{i i_j^*}$$

Repeat

Hierarchical Clustering

Another algo working directly w/ dissimilarity measures
(but also very popular to apply to Euclidean dists)

Let a clustering be denoted by

$$\mathcal{C} = \{C_1, \dots, C_k\} \qquad C_j \subseteq \{1, \dots, n\}$$

$$C_j \cap C_{j'} = \emptyset \quad j \neq j'$$

$$\{1, \dots, n\} = \bigcup_{j=1}^{k} C_j$$

1. Init: Start w/ every pt as a singleton cluster

$$\mathcal{C} = \{\{1\}, \{2\}, \dots, \{n\}\}$$

2. Merge the "closest" two clusters
$(|\mathcal{C}_{new}| = |\mathcal{C}_{old}| - 1)$

3. Repeat step 2 until we have one super cluster $(|\mathcal{C}| = 1, \; \mathcal{C} = \{\{1, \dots, n\}\})$

How to define "closest"?

How to define cluster dissim:

We'll define fns $d(G,H)$ that take two subsets of pts & return dissimilarity of the clusters.

1. Single linkage

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

The dissimilarity b/w the two most similar pts in G, H

2. Complete linkage

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

The dissim b/w the two most <u>dis</u>sim pts in G, H

3. Avg linkage

$$d_{AL}(G,H) = \frac{1}{|G|\cdot|H|} \sum_{i\in G}\sum_{j\in H} d_{ij}$$

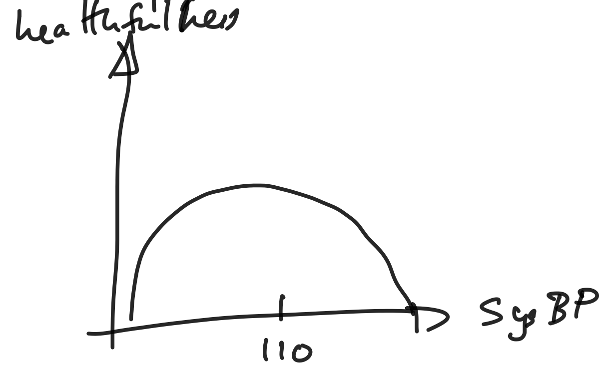
Back to supervised learning:

<u>Tree learning:</u>

So far in supervised learning:
we (for the most part) considered two models:

- linear: $\hat{f}(x) = \hat{\beta}^T x$ (or, $\hat{f}(x) = g(\hat{\beta}^T x)$)
- local avging: $\hat{f}(x) = $ weighted avg $\{K(x_i, x) \to Y_i\}$

linear: pros: simple, interpertable, can
deal w/ hi-dim x (esp using regularizatio
Cons: parametric & restrictive



local avging: pros: flexible, can estimate $\mathbb{E}[Y|X=x]$
w/o specifying a model

Cons: hard to interpret,
curse of dim,
high var

New model: trees! interpertable + flexible

$$\hat{f}(x) = \sum_{l=1}^{L} \mathbb{I}[x \in R_l] \beta_l$$

$$R_l \subseteq \mathbb{R}^p, \quad R_l \cap R_{l'} = \emptyset \quad l \neq l'$$

$$\mathbb{R}^p = R_1 \cup \cdots \cup R_L$$

(i.e. a partition of $\mathbb{R}^p$)

& the partition corresponds to a tree

Regression trees: $\hat{\beta}_\ell \in \mathbb{R}$ predicted value

Classification trees: $\hat{\beta}_\ell \in [0,1]$ predicted prob