Last lecture: Classification

- regression, density estimation, kNN, logistic regression

---

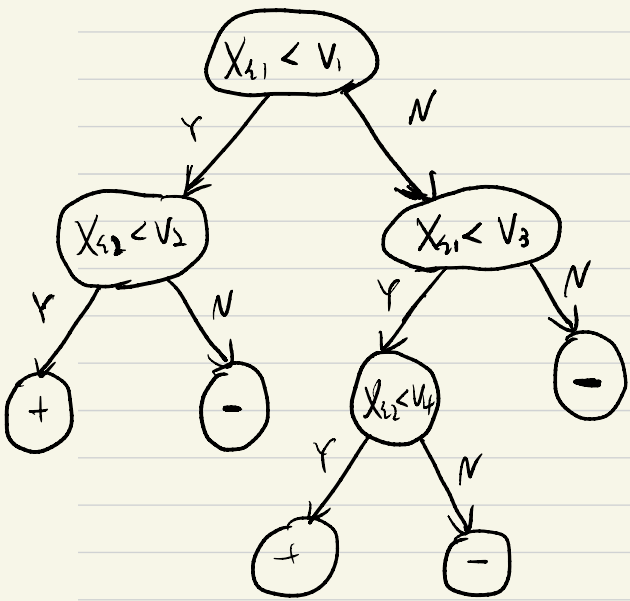
Decision Tree

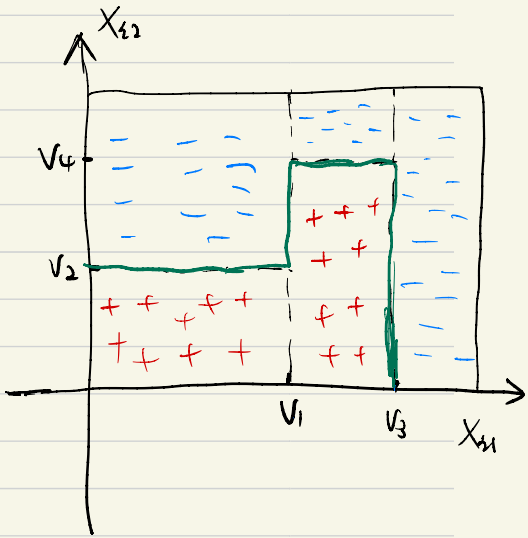
$\{(X_i, Y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{0, 1\}$ (− +)

$X_i = (X_{i1}, X_{i2})$

$d = 2$

features / predictors




$X^* = (X_1^*, X_2^*)$

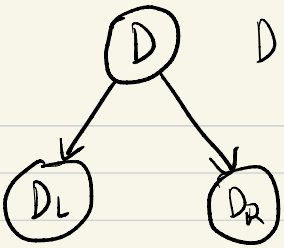
$X_1^* < V_1$, $X_2^* \geq V_2$

$\hat{Y}^* \leftarrow -$

① How to determine the split?

↳ regression tree: use "purity", find a split that maximizes

$$|D| \cdot \mathrm{Var}(D) - \left[ |D_L| \cdot \mathrm{Var}(D_L) + |D_R| \cdot \mathrm{Var}(D_R) \right]$$




$D = \{(X_i, Y_i)\}_{i}^{I_D}$

$|D|$ = # of samples

$$Var(D) = \frac{1}{|D|} \sum_{i \in D} (Y_i - \bar{y})^2$$

↑ variance of $Y_i$ in D

↳ $Y_i \in \{0, 1, \dots, K-1\}$ classification

use { Gini-index / Gini-impurity
information gain, i.e.

maximize $IG(Y|X) = H(Y) - H(Y|X)$

- $H(Y)$ is the entropy of $Y$

$$= -\sum_{j=0}^{K-1} P(Y_j) \cdot \log P(Y_j)$$

- $H(Y|X)$ is the conditional entropy of $Y|X$

↳ greedy searching

② When to stop?

- When the leaf is "pure", $Var(D_L) < \underline{\varepsilon}$ ← threshold
- When $|D| \leq \underline{\delta_N} = 20$
- stop at a fixed depth $D = 100$

"Bagging" = bootstrap aggregation

$\{(X_i, Y_i)\}_{i=1}^n$ —sample with replacement, $B=1000$→

$\{(X_i^{(1)}, Y_i^{(1)})\}_{i=1}^n \to \hat{C}_1$

$\vdots$

$\{(X_i^{(B)}, Y_i^{(B)})\}_{i=1}^n \to \hat{C}_B$

$$\hat{C}(x) = \begin{cases} 1 & \text{when } \frac{1}{B}\sum_{b=1}^{B} \hat{C}_b(x) \geq \frac{1}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

Random forest = Decision Tree + Bagging + Random Features

$\{(X_i, Y_i)\}_{i=1}^n$ —bootstrap→

$\{(X_i^{(1)}, Y_i^{(1)})\}_{i=1}^n \to \{(X_{i j_1^{(1)}}^{(1)}, \dots, X_{i j_m^{(1)}}^{(1)}, Y_i^{(1)})\}_{i=1}^n \to \hat{C}_1$

$\vdots$

$\{(X_i^{(B)}, Y_i^{(B)})\}_{i=1}^n \to \{(X_{i j_1^{(B)}}^{(B)}, \dots, X_{i j_m^{(B)}}^{(B)}, Y_i^{(B)})\}_{i=1}^n \to \hat{C}_B$

$X_i = (X_{i1}, \dots, X_{id}) \in \mathbb{R}^d$

$0 < m < d$

$m = \sqrt{d}$

$1 \leq j_1^{(b)} < \cdots < j_m^{(b)} \leq d, \quad b = 1, \dots, B$

$(X_{11}^{(1)}, X_{12}^{(1)}, \dots, X_{1d}^{(1)}, Y_1^{(1)})$

$d$

$\Downarrow \quad m < d$

$X_{1 j_1}^{(1)}, \dots, X_{1 j_m}^{(1)}$

$$\hat{C}(x) = \begin{cases} 1 & \text{when } \frac{1}{B}\sum_{b=1}^{B} \hat{C}_b(x) \geq \frac{1}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

# Kernel-based random forest

Caveat: It is motivated by random forest regression but is NOT a random forest.

- $A_j(x)$ be the cell/leaf that $x$ falls into in the $j$-th regression tree.

$\Rightarrow$ The $b$-th regression tree is

$$\hat{m}_b(x) = \frac{\sum_{i=1}^{n} Y_i \cdot \mathbb{1}\{X_i \in A_b(x)\}}{\sum_{j=1}^{n} \mathbb{1}\{X_j \in A_b(x)\}}$$

$\Rightarrow$ Random forest regression is

$$\tilde{m}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{m}_b(x)$$

$$= \frac{1}{B}\sum_{b=1}^{B} \frac{\sum_{i=1}^{n} Y_i \cdot \mathbb{1}\{X_i \in A_b(x)\}}{\sum_{j=1}^{n} \mathbb{1}\{X_j \in A_b(x)\}}$$

$$= \sum_{i=1}^{n} \sum_{b=1}^{B} \frac{Y_i \cdot \mathbb{1}\{X_i \in A_b(x)\}}{B \cdot \sum_{j=1}^{n} \mathbb{1}\{X_j \in A_b(x)\}}$$

Obs: In some extreme cases, $A_b(x)$ is too small to contain any $X_i$. (high prob.)

$$B \cdot \underbrace{\sum_{j=1}^{n} \mathbb{1}\{X_j \in A_b(x)\}}_{I(x)} = 0 \quad \text{(not good)}$$

$$B \cdot I(x) = \sum_{b=1}^{B} I(x) \quad \Longrightarrow \quad \sum_{b=1}^{B} I_b(x) = \sum_{b=1}^{B} \sum_{j=1}^{n} \mathbb{1}_{\{X_j \in A_b(x)\}}$$

$W_1, \to W_B$ , $B \cdot W_b \approx \sum_{b=1}^{B} W_b$

$$\hat{m}_{\text{KeRF}}(x) = \sum_{i=1}^{n} \frac{Y_i \sum_{b=1}^{B} \mathbb{1}_{\{X_i \in A_b(x)\}}}{\sum_{b=1}^{B} \sum_{j=1}^{n} \mathbb{1}_{\{X_j \in A_b(x)\}}}$$

$$\overset{D}{=} \sum_{i=1}^{n} \frac{Y_i \cdot k(x, X_i)}{\sum_{j=1}^{n} k(x, X_j)} \qquad k(x,u) = \frac{1}{B} \sum_{b=1}^{B} \mathbb{1}_{\{u \in A_b(x)\}}$$

↑ Nadaraya-Watson kernel regression (linear smoother)

---

Imbalanced classification

. Imbalanced dataset $\{(X_i, Y_i)\}_{i=1}^{n}$ $n = 1000$ $\{Y_i = 1\}$ 10 $\{Y_i = 0\}$ 990

- $c : \mathcal{X} \to \mathcal{Y} = \{0, 1\}$ classifier
- $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ Loss function
- $R(c) = \mathbb{E}[L(c(X), Y)]$ risk
- Bayes classifier $c^* = \operatorname{argmin} R(c) = \operatorname{argmin}_c \mathbb{E}[L(c(X), Y)]$

$$= \operatorname{argmin}_c \int \mathbb{E}[L(c(X), Y) \mid X = x] \, dP(x)$$

0-1 loss → $= \begin{cases} 1 & \text{when } P(Y=1 \mid x) \geq P(Y=0 \mid x) \\ 0 & \text{otherwise.} \end{cases}$

| | | Actual label $Y=0$ | $Y=1$ |
|---|---|---|---|
| predicted label | $\hat{Y}=0$ | $n_{00}$ | $n_{01}$ |
| | $\hat{Y}=1$ | $n_{10}$ | $n_{11}$ |

$$\text{misclassification error} = \frac{n_{01}+n_{10}}{n_{00}+n_{01}+n_{10}+n_{11}}$$

$n_{00}$ = True negative , $n_{11}$ = True positive , $n_{10}$ = false positive

$n_{01}$ = false negative

- False positive rate (FPR) $= \frac{FP}{N} = \frac{n_{10}}{n_{10}+n_{00}}$ ← Type I error rate
- False negative rate (FNR) $= \frac{FN}{P} = \frac{n_{01}}{n_{01}+n_{11}}$ ← Type II error rate
- Recall $= \frac{TP}{P} = 1 - FNR$ (focus on the "$Y=1$" class)
- False discovery rate (FDR) $= \frac{FP}{PP}$ (focus on the "$\hat{Y}=1$" class)

$$= 1 - \text{precision}$$

- F1-score $= \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = \frac{2 \cdot \text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}}$

Q: why do we need all these additional metrics?

$n=1000$, $|\{Y_i=1\}| = 10$, $|\{Y_i=0\}| = 990$

$C_{naive}(x) = 0 \Rightarrow$ misclassification error of $C_{naive} = \frac{10}{1000} = 1\%$

recall of $C_{naive} = \frac{0}{10} = 0$ $\qquad \frac{0}{0} = 1$

precision of $C_{naive} = 1 - FDR = 1 - \frac{0}{0} = 0$

Example (finance) $\qquad Y = \begin{cases} 1 & \text{fraudulent transaction,} \\ 0 & \text{beligned transaction.} \end{cases}$

- $C_{naive}(x) = 0$ is problematic
- Missing a true fraud can cause some severe financial losses.

Q: Would Bayes classifier $C^*(x)$ work here?

$$C^*(x) = \begin{cases} 1 & \text{when } \mathbb{P}(Y=1|X) \geq \mathbb{P}(Y=0|X), \\ 0 & \text{when } \mathbb{P}(Y=1|X) < \mathbb{P}(Y=0|X). \end{cases}$$

$$= \begin{cases} 1 & \text{when } \mathbb{P}(X|Y=1) \cdot \mathbb{P}(Y=1) \geq \mathbb{P}(X|Y=0) \cdot \mathbb{P}(Y=0), \\ 0 & \text{when } \mathbb{P}(X|Y=1) \cdot \mathbb{P}(Y=1) < \mathbb{P}(X|Y=0) \cdot \mathbb{P}(Y=0). \end{cases}$$

The estimated classifier will be

$$\hat{C}(x) = \begin{cases} 1 & \text{when } \hat{\mathbb{P}}(X|Y=1) \cdot \hat{\mathbb{P}}(Y=1) \geq \hat{\mathbb{P}}(X|Y=0) \cdot \underline{\hat{\mathbb{P}}(Y=0)}, \\ 0 & \text{when } \hat{\mathbb{P}}(X|Y=1) \cdot \mathbb{P}(Y=1) < \hat{\mathbb{P}}(X|Y=0) \cdot \underline{\hat{\mathbb{P}}(Y=0)} \end{cases}$$

$\approx 0$ ↑ with high prob.

$\hat{\mathbb{P}}(Y=0) = \frac{990}{1000} = 0.99$

Standard 0-1 loss and its associated Bayes classifier are not suitable for imbalanced dataset!

Solution: Use a weighted 0-1 loss:

$$L_w(c(X), Y) = w_1 \cdot \mathbb{1}\{c(X)=0, Y=1\} + w_2 \cdot \mathbb{1}\{c(X)=1, Y=0\}, \quad w_1 >> w_2$$

e.g. $w_1 = 100$, $w_2 = 1$

$$\mathbb{E}\left[ L_w(c(X), Y) \mid X=x \right] = w_1 \cdot \mathbb{1}\{c(x)=0\} \cdot \mathbb{P}(Y=1 \mid X=x) + w_2 \cdot \mathbb{1}\{c(x)=1\} \cdot \mathbb{P}(Y=0 \mid X=x)$$

$$= \begin{cases} w_1 \cdot \mathbb{P}(Y=1 \mid X=x) & \text{if } c(x)=0, \\ w_2 \cdot \mathbb{P}(Y=0 \mid X=x) & \text{if } c(x)=1. \end{cases}$$

Bayes classifier

$$c^*(x) = \begin{cases} 0 & \text{when } w_1 \cdot \mathbb{P}(Y=1 \mid x) \le w_2 \cdot \mathbb{P}(Y=0 \mid x), \\ 1 & \text{when } w_1 \cdot \mathbb{P}(Y=1 \mid x) > w_2 \cdot \mathbb{P}(Y=0 \mid x). \end{cases}$$

$$= \begin{cases} 0 & \text{when } w_1 \cdot \mathbb{P}(Y=1) \cdot \mathbb{P}(x \mid Y=1) \le w_2 \cdot \mathbb{P}(x \mid Y=0) \cdot \mathbb{P}(Y=0), \\ 1 & \text{when } w_1 \cdot \mathbb{P}(Y=1) \cdot \mathbb{P}(x \mid Y=1) > w_2 \cdot \mathbb{P}(Y=0) \cdot \mathbb{P}(x \mid Y=0). \end{cases}$$

$w_1 >> w_2$

Optimization perspective of classification

$$c^* = \underset{c \in \mathcal{F}}{\operatorname{argmin}} \; \mathbb{E}\left[ L(c(X), Y) \right]$$

$\mathcal{F}$ is a class of candidate classifiers

$\Longrightarrow$ Empirical risk minimization

$$\hat{c}_n = \underset{c \in \mathcal{F}}{\operatorname{argmin}} \left[ \frac{1}{n} \sum_{i=1}^{n} L(c(X_i), Y_i) \right] = \underset{c \in \mathcal{F}}{\operatorname{argmin}} \; \hat{R}_n(c)$$

- Unbiasedness: $\mathbb{E}\left[ \hat{R}_n(c) \right] = \mathbb{E}\left[ \frac{1}{n} \sum_{i=1}^{n} L(c(X_i), Y_i) \right] = R(c)$
- generalization: $c^* = \underset{c \in \mathcal{F}}{\operatorname{argmin}} \; R(c)$, $\hat{c}_n = \underset{c \in \mathcal{F}}{\operatorname{argmin}} \; \hat{R}_n(c)$

Ideally, we want the excess risk $R(\hat{c}_n) - R(c^*)$ to be small.

$$R(\hat{c}_n) - R(c^*) = R(\hat{c}_n) - \hat{R}_n(\hat{c}_n) + \underbrace{\hat{R}_n(\hat{c}_n) - \hat{R}_n(c^*)}_{\leq 0 \text{ because } \hat{c}_n \text{ is a minimizer of } \hat{R}_n} + \hat{R}_n(c^*) - R(c^*)$$

$$\leq \left| R(\hat{c}_n) - \hat{R}_n(\hat{c}_n) \right| + 0 + \left| \hat{R}_n(c^*) - R(c^*) \right|$$

$$\leq 2 \sup_{c \in \mathcal{F}} \left| R(c) - \hat{R}_n(c) \right|$$

In order for $\hat{c}_n = \underset{c \in \mathcal{F}}{\operatorname{argmin}} \; \hat{R}_n(c)$ to generalize well, we need

$$\sup_{c \in \mathcal{F}} \left| R(c) - \hat{R}_n(c) \right| = o_p(1)$$

- Case 1: if $|\mathcal{F}|$ is finite, then

$$\sup_{c \in \mathcal{F}} \left| \hat{R}_n(c) - R(c) \right| = O_p\left( \sqrt{\frac{\log |\mathcal{F}|}{n}} \right) \quad \text{by Hoeffding's inequality.}$$

- Case 2: if $|\mathcal{F}| = \infty$ but $\mathcal{F}$ has a finite VC-dimension $V$,

then $$\sup_{c \in \mathcal{F}} \left| \hat{R}_n(c) - R(c) \right| = O_p\left( \sqrt{\frac{V \cdot \log n}{n}} \right)$$

e.g. for linear regression / classifier, VC-dimension $= O(\text{dimension})$

for neural networks, a fully connected NN with width $W$ and depth $H$ has

$$\text{VC-dimension} = \Omega(W \cdot H)$$

Peter Bartlett et al. 2017

"Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks", COLT 2017

VC-dimension bounds are not useful when $W \cdot H \geq n$ !!

- Rademacher complexity

For a given classifier $c$, $\hat{RC}_n(c) = \mathbb{E}_{\xi} \sup_{c \in \mathcal{F}} \sum_{i=1}^{n} \xi_i \cdot c(X_i)$

where $\sigma_i = \begin{cases} 1 & \text{with prob. } \frac{1}{2}, \\ -1 & \text{with prob. } \frac{1}{2}. \end{cases}$

$\sigma_1, \dots, \sigma_n$ are IID.

The population-level Rademacher complexity is defined as

$$R(\mathcal{C}) = \mathbb{E}\left[\hat{R}_n(\mathcal{C})\right]$$

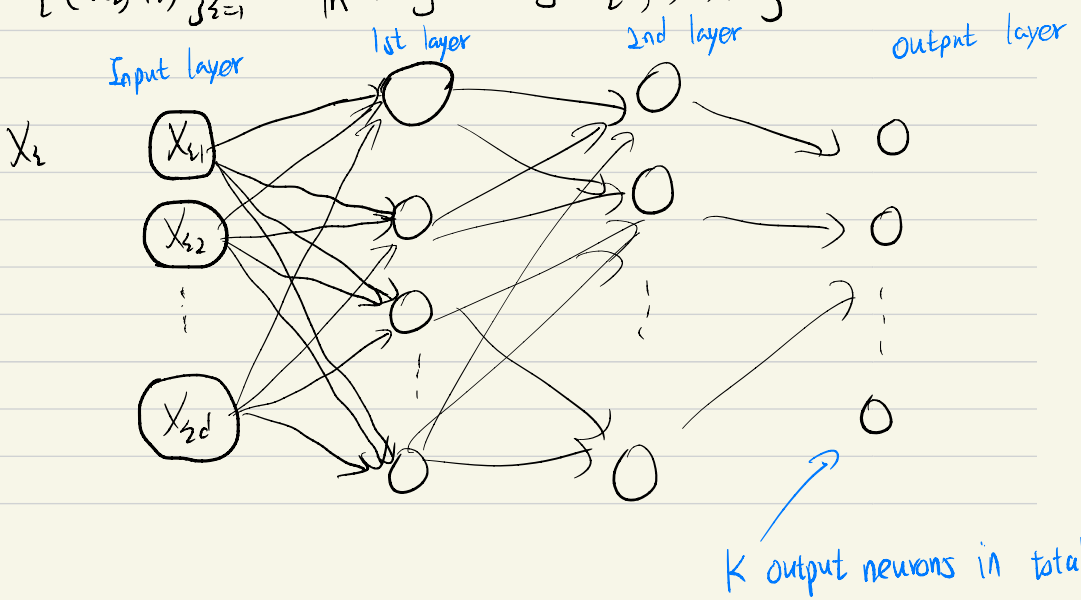
$\mathbb{E}$: over $X_1, \dots, X_n$

$$\sup_{c \in \mathcal{C}} \left| \hat{R}_n(c) - R(c) \right| = O_p\left( \frac{R(\mathcal{C})}{n} + \sqrt{\frac{\log\log n}{n}} \right).$$

---

Neural network classification

$$\{(X_i, Y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathcal{Y} \qquad \mathcal{Y} = \{0, 1, \dots, k-1\}$$



$X_1 = x \in \mathbb{R}^d$, $W_L \in \mathbb{R}^{d_L \times d_{L-1}}$, $L = 1, \dots, H$ and $d_0 = d$

$b_L \in \mathbb{R}^{d_L \times 1}$ $\quad d_H = K$

$$f(x) = \sigma_L(W_L Z_L + b_L), \quad Z_L = \sigma_{L-1}(W_{L-1} Z_{L-1} + b_{L-1}), \quad L = 1, \dots, H$$

with $Z_1 = x$

$\sigma_L$'s are some activation function

ReLU $\quad \sigma(u) = \begin{cases} u & \text{when } u \geq 0, \\ 0 & \text{otherwise.} \end{cases}$

sigmoid $\quad \sigma(u) = \frac{1}{1+e^{-u}}$

For K-classes classification problem, the last layer will use the softmax function

$$\sigma_L(u) = \left( \frac{e^{u_1}}{\sum_{j=1}^K e^{u_j}}, \dots, \frac{e^{u_K}}{\sum_{j=1}^K e^{u_j}} \right) = (f(x)_1, \dots, f(x)_K)$$

$\Rightarrow$ the output of this NN is $f(x) \in [0,1]^K$, $\sum_{j=1}^K f(x)_j = 1$.

How do we train NN in practice?

$Y_i \in \{0, 1, \dots, K-1\}$ $\xrightarrow{\text{one-hot encoding}}$ $\tilde{Y}_i = (0, \dots, 1, \dots 0)$

$\tilde{Y}_{ij} = 1$ if $Y_i = j-1$

Define the cross-entropy loss

$$L_c(f(x), \tilde{Y}) = -\sum_{j=1}^{k} \tilde{Y}_j \cdot \log\left[f(x)_j\right] \qquad \sum_{j=1}^{k} f(x)_j = 1, \quad f(x)_j \geq 0$$

$$\hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} L_c(f(X_i), Y_i)$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} \tilde{Y}_{ij} \cdot \log\left[f(x)_j\right]$$

- special case (binary classification): $Y_i \in \{0, 1\}$

$$L_c(f(X), Y) = -Y \cdot \log f(x) - (1-Y)\log\left[1 - f(x)\right]$$

↳ This is indeed the log-likelihood of a binomial distribution.

$Y_i | X_i \sim \text{Bernoulli}(p_i)$ with $p_i = f(X_i)$

$$\hat{R}_n(f) = -\frac{1}{n}\sum_{i=1}^{n}\left(Y_i \log f(X_i) + (1-Y_i)\log\left[1 - f(X_i)\right]\right)$$

penalized method:

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}}\left[\hat{R}_n(f) + P_\lambda(f)\right]$$

e.g. (L1-penalized logistic regression)

$$-\frac{1}{n}\sum_{i=1}^{n}\left[Y_i(\beta_0 + \beta_1^T X_i) - \log\left(1 + e^{\beta_0 + \beta_1^T X_i}\right)\right] + \lambda \sum_{j=0}^{d} |\beta_j|$$

# Clustering (Unsupervised Learning)

↳ Separate observations into groups so that observations within the same group are similar.

Marina Meila 2018 "How to tell when a clustering is (approximately) correct using convex relaxations"

Density-based clustering
- mode clustering (mean shift clustering)
- level-set clustering ⟷ dynamical system
- DBSCAN

Observation-based clustering
- hierarchical clustering ⟷ graph theory
- K-Means clustering ⟷ transportation theory
- spectral clustering ⟷ harmonic analysis

- Spectral clustering

$G = (V, E)$, $V = \{X_1, \dots, X_n\} \subset \mathbb{R}^d$

How do we define the edge set $E$?

↳ Construct a similarity matrix $S \in \mathbb{R}^{n \times n}$

- $\varepsilon$-neighbor graph $S_{ij} = 1$ when $d(X_i, X_j) < \varepsilon$ ← some prespecified threshold
- $kNN$ graph $S_{ij} = 1$ when $X_i$ is among the $kNN$ of $X_j$ or $X_j$ is among the $kNN$ of $X_i$

- Fully connected weighted graph $S_{ij} = \exp\left[-\frac{d(x_i, x_j)^2}{2\sigma^2}\right]$,

  $\sigma > 0$ is a tuning parameter

$\Rightarrow$ (Unnormalized) graph Laplacian

$L_{un} = D - S$, where $D$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} S_{ij}$

- $V^T L_{un} V = \frac{1}{2} \sum_{i,j=1}^{n} S_{ij} (V_i - V_j)^2$ for any $V \in \mathbb{R}^n$
- $L_{un}$ is positive semi-definite and $L_{un}^T = L_{un}$ (symmetric)
- The smallest eigenvalue of $L_{un}$ is 0, whose corresponding eigenvector is $\mathbb{1}_n = (1, 1, \ldots, 1)^T \in \mathbb{R}^n$.
- $0 = \lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_1$.

★ Why is Graph Laplacian useful for clustering the data?

Suppose that a graph $G = (V, E)$ with its adjacency matrix as $S$ has $V_1, \ldots, V_k \subset V$ connected components.

$\Rightarrow$ Then, $L_{un} = D - S$ has $k$ zero eigenvalues.

Those eigenvectors of these $k$ zero eigenvalues are $\mathbb{1}_{V_1}, \mathbb{1}_{V_2}, \ldots, \mathbb{1}_{V_k}$.

($\mathbb{1}_{V_1}$ means that those components corresponding to $V_1$ are 1, and others are 0.)

Implementation: (Python / Pyspark)

https://github.com/zhangyk8/Spectral-Clustering