Classification Problems

Examples:
- ▷ Fraud detection in credit card payments
- ▷ Categorize digital images: dog, cat, human

Supervised Learning:
- ▷ $X \subset \mathbb{R}^k$: domain set
- ▷ $Y \subset N$: finite target set, $|N| = q$

1. (Multiclass) Logistic Regression

$\mathcal{X} \subset \mathbb{R}^k$, $\mathcal{Y} = \{0,1\}^q$ $\mathcal{F} = \{ h : \mathbb{R}^k \to \mathbb{R}^q : x \mapsto \text{softmax}(Wx),\ W \in \mathbb{R}^{q \times k} \}$

with $\underset{\mathbb{R}^q}{\text{softmax}(z)} = \left[\frac{\exp(z_1)}{\sum_{i=1}^q \exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^q \exp(z_i)}, \dots, \frac{\exp(z_q)}{\sum_{i=1}^q \exp(z_i)} \right]$

- loss function $\ell(y,z) := -\sum_{i=1}^q y_i \log(z_i) = -\langle y, \log(z) \rangle$

LinReg: $\ell(y,z) = (y-z)^2$

- empirical risk $L_S(W) = \frac{1}{n} \sum_{j=1}^n \ell(y^j, \text{softmax}(Wx^j))$

if $S = \{x^j, y^j\}_{j=1}^n$ with hot encoding s.t. $y^j = (0,0,\ ,0,1,0,\dots,0)$ — if j-th data point is in class i, 1 at i-th position

- Add ridge/lasso type regularization term is an option $[W \mapsto L_S(W)$ is convex$]$
- Optimization is non-trivial, but convex optimization can be used.

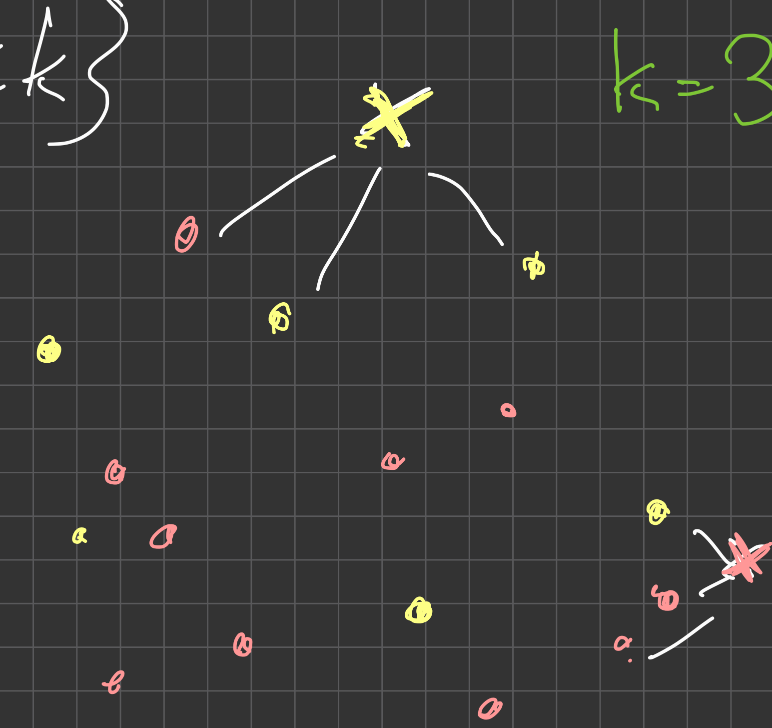
2. K-Nearest Neighbors

Let $X \subset \mathbb{R}^k$, $Y \subset \{0,1\}^q$. Let $d: X \times X \to \mathbb{R}$ be a metric, e.g. $d(x,x') := \|x-x'\|_2$.

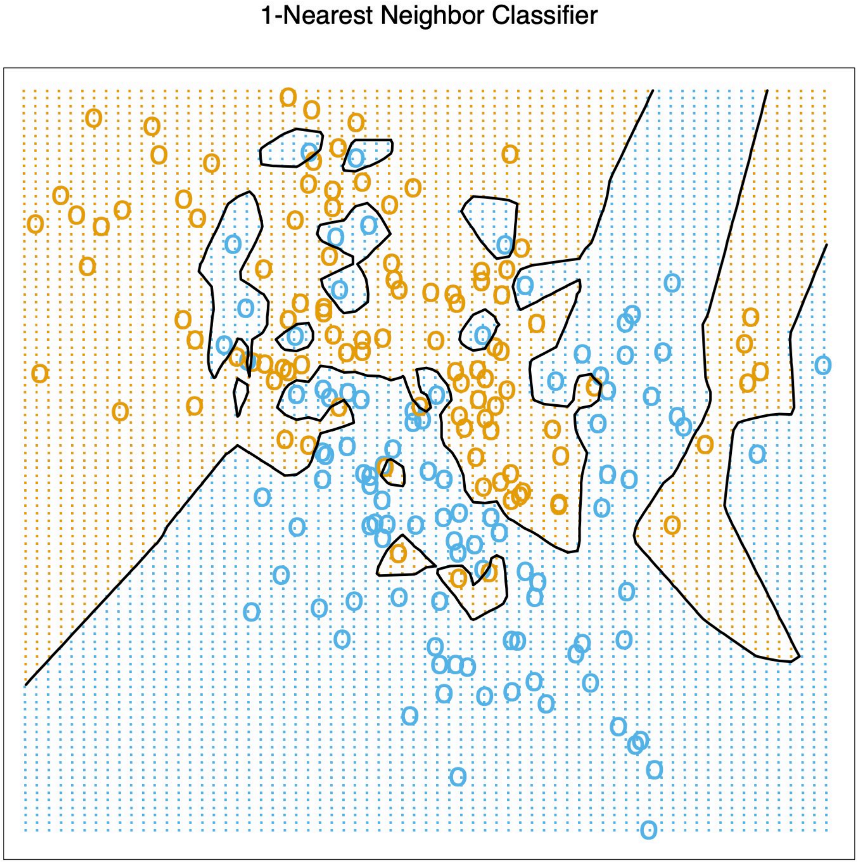
If $S_x = \{x^1, \dots, x^n\}$ is a set, define $\pi_i(x)$ as the i-th closest member of S_x to x (w.r.t. metric d)

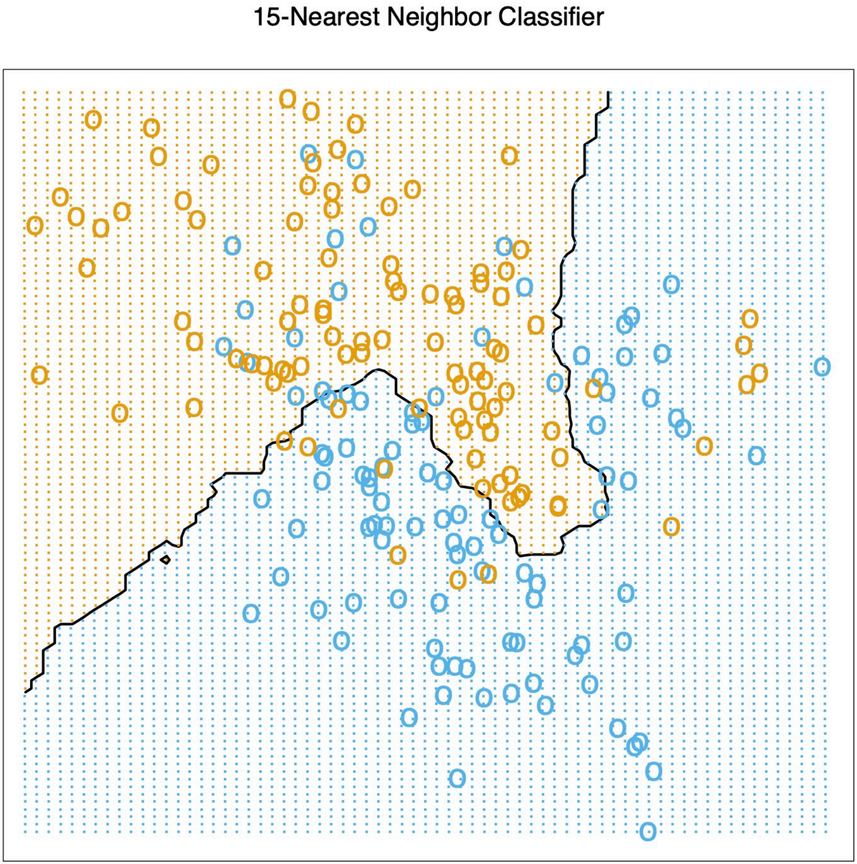
Algorithm: Input: Training set $S = \{x^j, y^j\}_{j=1}^n$, parameter k.

Output: Function $h_S : X \to Y$ such that $h_S(x)$ is the majority label among $\{y^{\pi_i(x)}, i \le k\}$



- (+) "local" method, simple
- (−) Needs all pairwise comparison ($O(kn)$ computations)
- (−) Suffer from a "curse of dimensionality" if k is large


1-Nearest Neighbor Classifier



Feature selection: Natural language Processing

Q: How to work with text data?

→ spam/no detector

→ categorizing text items according to topics

"document"

$x =$ "Michael likes walking his dog in his neighborhood."

1. Tokenize:

"Michael", "likes", "walking", "dog", "in", "his", "neighbhoorhood"

2. Build dictionary: Do this for all documents x of interest.

"Aaron", "Amsterdam", "am", ..., "his", ..., "Michael", ..., "walking", ..., "zebra".

List of d words

3. Encoding: a) Count how often each word/token in x occurs, create sparse vector in $\mathbb{R}^d$:

$$\Phi : \mathcal{D} \to \mathbb{R}^d, \quad \Phi(x) = \frac{\tilde{\Phi}(x)}{\|\tilde{\Phi}(x)\|}, \text{ where } \tilde{\Phi}(x) = (0, 0, 0, \dots, \overset{\tilde{\Phi}(x)_w}{\overset{\shortparallel}{2}}, \dots 1, \dots, 1, \dots, 0)$$

Modifications can include:

- Removal of very common words such as "in", "the"
- "n-grams": Use "Michael likes", "likes walking", etc. as tokens ("2-gram")
  - ⊕ Better semantic understand
  - ⊖ Computationally more challenging as dictionary dimension d is larger.

b) Term-Frequency - Inverse Document Frequency (TF-IDF)

Choose $\phi(x)_w = \text{freq}_w \cdot \left(\log\left(\frac{d}{N_w}\right) + 1\right)$

freq_w : frequency of word w in document x

d : total number of words

N_w : nr. of documents containing word w.

⊕ scales down importance of words that are common across documents.