Classification Problems

Examples:
- Fraud detection in credit card payments
- Categorize digital images : dog, cat, bus, house ....

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

Methods:
- Logistic Regression
- k-nearest neighbors
- Support vector machines

(Multiclass) Logistic Regression

$X \subset \mathbb{R}^k$, $Y = \{0,1\}^q$, q: nr. of classes/categories

$$\mathcal{F} = \{ h : X \to \mathbb{R}^q, \; x \longmapsto \text{softmax}(Wx), \; W \in \mathbb{R}^{q \times k} \}$$

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

- loss function: $\ell(y, z) := -\sum_{i=1}^q y_i \log(z_i)$
- empirical risk: $L_S(W) = \frac{1}{n} \sum_{j=1}^n \ell(y^j, \text{soft}(Wx^j))$

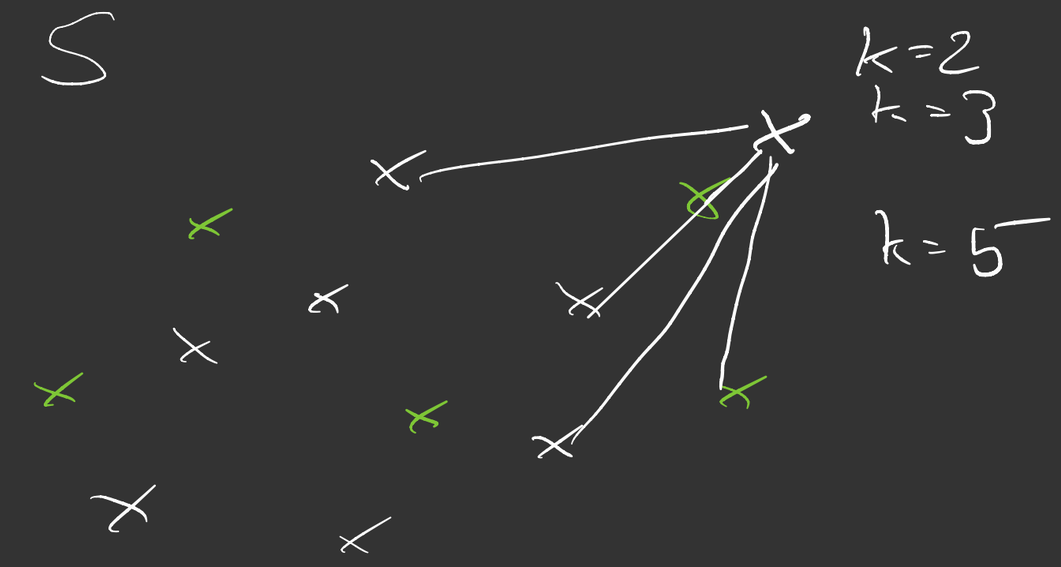
with hot encoded $y^j = (0, 0, 1, 0, \dots, 0)$

↑ i-th coordinate if label of j-th data point is equal to class i.

Options:
- Add regularization terms (e.g., as in Ridge / Lasso regression).

Observation: Optimization is non-trivial, but ERM is still convex.

K-Nearest Neighbor Classification



Parameter k: Determines with how many neighbors to be compared

Given $|S| = n$ data point.

If $S_x = \{x_1, \dots, x_n\}$ is a set, $\pi_i(x)$ denote i-th closest member of S_x to x with respect to d.

Algorithm: Input: Training set $S = (x_i, y_i)_{i=1}^n$, parameter k

Output: Function $h_S: X \to Y$ s.t. $h_S(x)$ is the majority label $\{y_{\pi_i(x)} : i \le k\}$

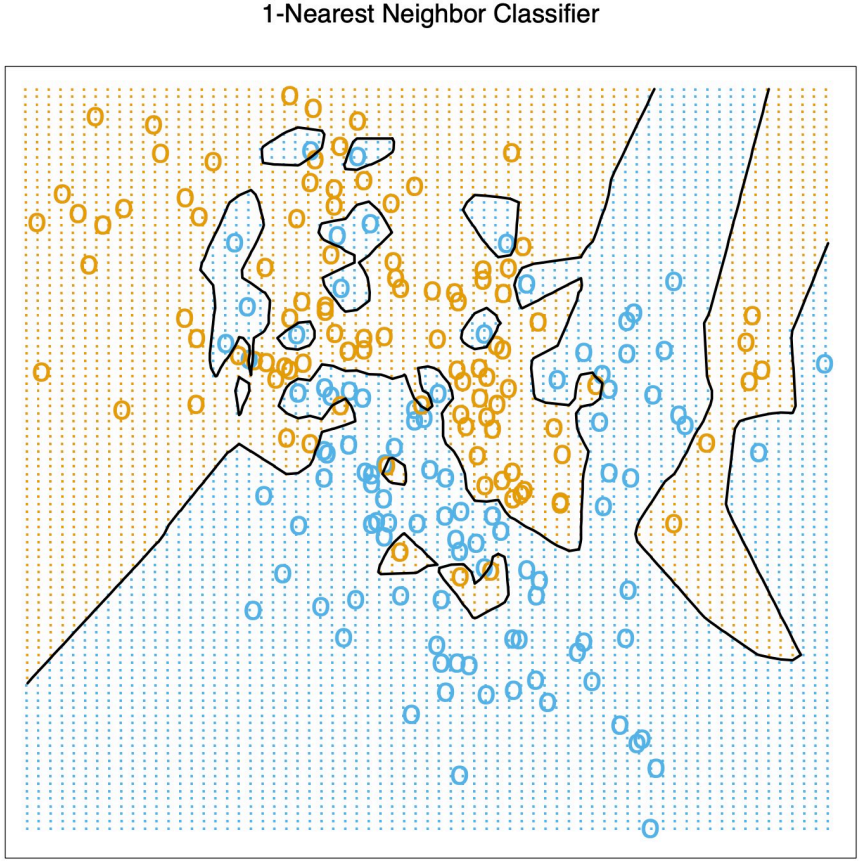
Properties:

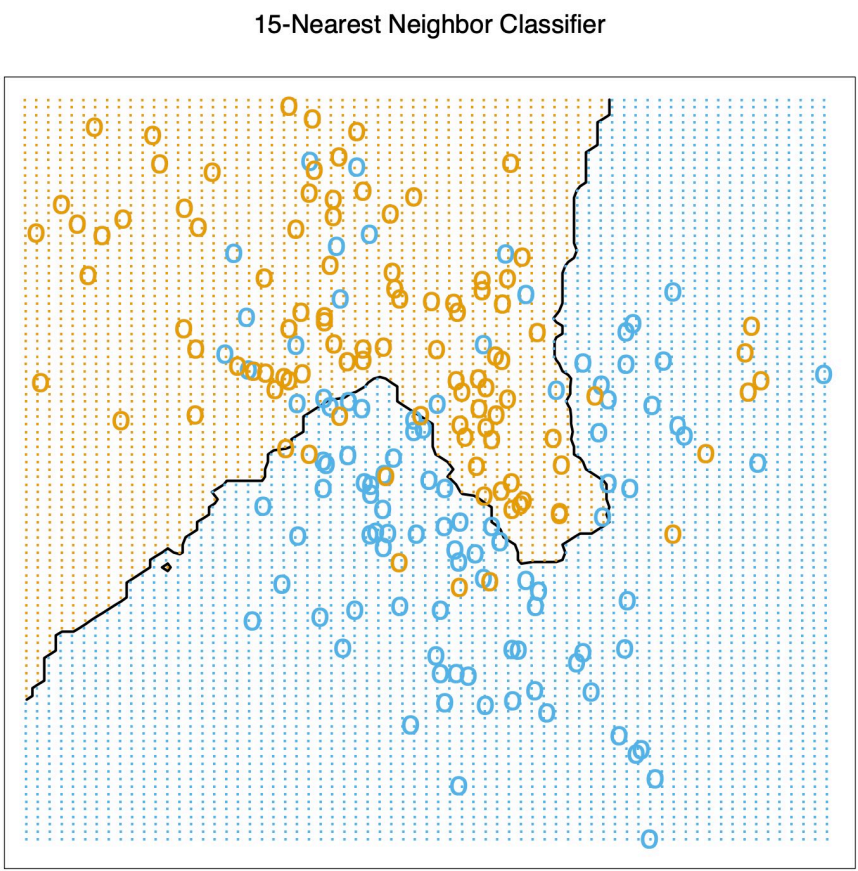
(+) "local method", simple

(−) Needs all pairwise comparisons on n points

$O(k \cdot n)$ computations even if locality of distances taken into account

(−) Suffer from "curse of dimensionality"


1-Nearest Neighbor Classifier


15-Nearest Neighbor Classifier

Natural Language Processing

Q: How to process test data?

Example for "2-grams": Tokenization such that $z \mapsto$ {'Michael likes', 'likes walking', 'walking his', 'his dog' ...}

$z :=$ "Michael likes walking his dog is his neighborhood"

1. Tokenize — "1-grams"

"Michael", "likes", "his", "dog", "is", "neighborhood"

2. Build vocabulary: Do this for all data points (here: sentences)

"Aaron", "Amsterdam" ... "zebra"

3. Encoding: a) Count encoding / vectorizer: List of k words.

$$\Phi : \mathcal{D} \longrightarrow \mathbb{R}^k$$

$$\Phi(z) = \frac{\tilde{\Phi}(z)}{\|\tilde{\Phi}(z)\|}, \text{ with } \tilde{\Phi}(z) = (0, 0, 0, \ldots, 2, 0, 1, 0, \ldots)$$

"his"

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

$$\mathcal{L}(z)_w = freq_w(z) \cdot \left(\log\left(\frac{K}{N_w}\right) + 1\right)$$

$freq_w$: frequency of word w in document z

K : nr. of of words in vocabulary

N_w : nr. of documents containing word w.

Count Vectorizer

⊖ simple

TF-IDF

⊕ scales down importance of words that are common across documents