K-NN review

Elistratov Semyon

MMF CMC MSU

2021

Introduction

- K-NN - family of metric algorithms for various tasks.
  $(x_1, y_1)..., (x_N, y_N)$ - training sample.,
  $(x_{N+1}, y_{N+1})..., (x_{N+M}, y_{N+M})$ - test sample.
  $x_i \in \mathbb{R}^D$
- Key idea
  - predict the response based on the responses of the K nearest neighbors.
- Assumption
  - similar objects yield similar outputs
- Hyperparameters
  - k - number of nearest neighbors
  - $\rho(x, y)$ - distance function

Features

Features

- simple to implement
- interpretable
- small number of hyperparameters
- can work without direct vector representation of objects
- memory-based, lazy learning
- curse of dimensionality

Metric

We should determine metric function for finding nearest neighbors.

Most popular metrics:

- L2 (Euclidean): $\rho(x, y) = ||x - y||_2^2 = \sum_{i=1}^{D}(x_i - y_i)^2$
- L1 (Manhattan): $\rho(x, y) = ||x - y||_1 = \sum_{i=1}^{D}|x_i - y_i|$
- Cosine: $\rho(x, y) = \frac{\langle x, y \rangle}{||x||\,||y||}$
- Chebyshev: $\rho(x, y) = ||x - y||_\infty = \max_i |x_i - y_i|$
- Mahalanobis: $\rho(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}$

Besides, it is not really required for function $\rho(x, y)$ to satisfy all algebraic metric axioms (triangle rule).

For some cases (categorial features) more special distance functions are needed.

Euclidean metric computation

Let $X \in \mathbb{R}^{N \times D}$, $Z \in \mathbb{R}^{M \times D}$ - train and test samples respectively.

Computation $\rho(x, y) = ||x - z||_2^2$ for each pair require

Euclidean metric computation

Let $X \in \mathbb{R}^{N \times D}$, $Z \in \mathbb{R}^{M \times D}$ - train and test samples respectively.

Computation $\rho(x, y) = ||x - z||_2^2$ for each pair require $3NMD$ operations.

Simple trick:

$$||x - z||_2^2 = \langle x - z,\ x - z \rangle = \langle x,\ x \rangle + \langle z,\ z \rangle - 2\langle x,\ z \rangle$$

$$\rho(x, y) = ||x||_2^2 + ||z||_2^2 - 2\langle x,\ z \rangle$$

Thus, now the computation require

Euclidean metric computation

Let $X \in \mathbb{R}^{N \times D}$, $Z \in \mathbb{R}^{M \times D}$ - train and test samples respectively.

Computation $\rho(x, y) = ||x - z||_2^2$ for each pair require $3NMD$ operations.

Simple trick:

$$||x - z||_2^2 = \langle x - z,\ x - z \rangle = \langle x,\ x \rangle + \langle z,\ z \rangle - 2\langle x,\ z \rangle$$

$$\rho(x, y) = ||x||_2^2 + ||z||_2^2 - 2\langle x,\ z \rangle$$

Thus, now the computation require $2D(N + M) + 2NMD$. Besides, last part could be effectively compute by matrix multiply (so $O(N^{2.3727})$).

Regression

- Find k nearest neighbors:

$$\rho(z, x_1) \leq \rho(z, x_2) \leq ... \leq \rho(z, x_k)$$

- Predict:
  - Mean: $\hat{y}(z) = \frac{1}{k}\sum_{i=1}^{k} y_i$
  - Median: $\hat{y}(z) = median\{y_1, ..., y_k\}$
- Weighted algorithm:

$$\hat{y}(z) = \frac{\sum_{i=1}^{i=k} w(i, \rho(z, x_i)) y_i}{\sum_{i=1}^{i=k} w(i, \rho(z, x_i))}$$

Classification

- Find k nearest neighbors:

$$\rho(z, x_1) \leq \rho(z, x_2) \leq ... \leq \rho(z, x_k)$$

- Predict:

$$g_c(z) = \sum_{i=1}^{k} [y_i = c]$$
$$\hat{y}(z) = \mathrm{argmax}_c\, g_c(z)$$

- Weighted algorithm:

$$g_c(z) = \sum_{i=1}^{k} w(i, \rho(z, x_i))[y_i = c]$$

Weights

Distance do not influence the predict directly.

So we can use weighted algorithm.

The most common used weight:

- $w_i = \frac{1}{\rho(z, x_i) + \epsilon}$, $\epsilon > 0$
- $w_i = \alpha^i$, $\alpha \in (0, 1)$

MNIST classification

Consider classification task on MNIST dataset.

Dataset contains 70k images. Each image has size 28×28. Train sample: first 60k objects, test sample: 10k objects.

Each object is image of handwritten digit from 0 to 9. Therefore, this is 10 class classification task.

Model

Hyperparameters of model:

- K
- metric (only Euclidean or cosine)
- weights (weighted prediction or not)
- strategy ('my_own', brute, kd-tree, and ball-tree)
- test_block_size

For such strategies as brute, kd-tree, and ball-tree we should use it sklearn implementation.

Cross-validation

You should realize CV algorithm by yourself.

One of the advantages of using CV with K-NN is effective simultaneous computation for different values of K.

So, for an array of k: $k_1 < k_2 < ... < k_n$, we compute distance matrix only once!

Augmentation

You should implement two strategies of augmentation.

- Train augmentation
  - augment only train data
  - fit model on original and augmented training sample
  - implement model on test data
- Test augmentation
  - fit model on original train data
  - for each test object create it augmented copies
  - get forecast for original and augmented test objects
  - get the final prediction by voting