Decoupling

2020年11月9日 13:36

Nearest Neighbour classification

- Non-parametric (rely only on training data)
- K-d tree (not work for high dim).

⇒ Nearest → R-Near → (c, R)-Near.

<u>Def.</u> Locality-Sensitive hashing.

A family $\mathcal{H}$. (R, cR, p_1, p_2)-sensitive if:

$$\begin{cases} \text{if } \|p-q\| \le R, & Pr_{\mathcal{H}}[h(p)=h(q)] \ge p_1 \\ \text{if } \|p-q\| \ge cR, & Pr_{\mathcal{H}}[h(p)=h(q)] \le p_2 \end{cases} \qquad (p_1 > p_2).$$

Extreme Case. $p_1 = 1$. $p_2 = 0$

(not achieveable)

Amplify the gap between p_1 & p_2.

Preprocessing:

1. Choose L functions g_j, $j = 1, \dots L$, by setting $g_j = (h_{1,j}, h_{2,j}, \dots h_{k,j})$, where $h_{1,j}, \dots h_{k,j}$ are chosen at random from the LSH family $\mathcal{H}$.
2. Construct L hash tables, where, for each $j = 1, \dots L$, the j^{th} hash table contains the dataset points hashed using the function g_j.

Query algorithm for a query point q:

1. For each $j = 1, 2, \dots L$
   i) Retrieve the points from the bucket $g_j(q)$ in the j^{th} hash table.
   ii) For each of the retrieved point, compute the distance from q to it, and report the point if it is a correct answer (cR-near neighbor for Strategy 1, and R-near neighbor for Strategy 2).
   iii) (*optional*) Stop as soon as the number of reported points is more than L'.

<u>Theorem</u>

If there exists $p^* \in B(q, r)$, we will find a point that is cR-near to q with probability $\ge \frac{1}{2} - \frac{1}{e}$.

LSH Library

- $h_{r,b} = \left\lfloor \frac{\langle r, x \rangle + b}{w} \right\rfloor$

- $h_{r,b} = \lfloor \frac{\langle r, x\rangle + b}{w} \rfloor$

  w: hyper parameter

  $r \in \mathbb{R}^d \sim$ Gaussian

  $b \sim unif\ [0, w)$

$\Rightarrow \Pr[h_{r,b}(p) = h_{r,b}(q)]$

Let $c = \|p-q\|_2$. (because projection of gaussian is also gaussian)

$$= \int_0^{\frac{w}{c}} f_p(r) \cdot \frac{(w-rc)}{w}\, dr = \int_0^{w} \frac{1}{c} f_p\left(\frac{t}{c}\right)\left(1-\frac{t}{w}\right) dt$$

Metric Learning

- Searching nearest neighbour in $\mathbb{R}^d$ may not be the best (like kernel method)

Goal: Learn $f: \mathbb{R}^d \to \mathbb{R}^k$

NCA algorithm:

$$P_{ij} = \frac{e^{-\|f(x_i)-f(x_j)\|^2}}{\sum_{k\neq i} e^{-\|f(x_i)-f(x_k)\|^2}}$$

to avoid mapping to the same point

$$P_{ii} = 0$$

Suppose dataset associated with labels C.

Let $C_i = \{j \mid c_i = c_j\}$ the class containing i.

Loss function $\Rightarrow$ $P_i = \sum_{j\in C_i} P_{ij}$

optimize

$$f(A) = \sum_i \sum_{j\in C_i} P_{ij} = \sum_i P_i$$

LMNN: triplet loss.

$$L_{rank} = \max(0, \|f(x) - f(x^+)\|_2 - \|f(x) - f(x^-)\|_2 + r)$$

r: margin