KNN

20 September 2024 09:33

Whole thing is based off the idea of "Birds of a feather flock together"

↓ "Similar data points are close together" — Assumptⁿ

But how do you measure how similar the birds are?

Distance

↳ Can be metric or non-metric

$d(x,x) = 0$ — +ve reflexivity

$d(x,y) = d(y,x)$ — symmetry

$d(x,y) \leq d(x,z) + d(z,y)$ — Δ^{ar} inequality.

① Distance 1: Minkowski

$$d^{(r)}(p,q) = \left(\sum_{k=1}^{L} |p_k - q_k|^r\right)^{1/r}$$

r → tells us the type of metric

$\bar{p}, \bar{q}$ are L dimensional vectors

② Distance 2: Manhattan

$r = 1$

L_1 norm

$$d(p,q) = \sum_{k=1}^{L} |p_k - q_k|$$

③ Distance 3: Euclidean

$r = 2$

L_2 norm

$$d(p,q) = \left[\sum |p_k - q_k|^2\right]^{\frac{1}{2}}$$

④ Distance 4: α??

$r = \alpha$

L_α norm

$$d(p,q) = \max_k (|p_k - q_k|)$$

$k = 1, 2 \cdots L$

But how do we normalize when we have to?

① Method 1: $x' = \frac{x - x_{min}}{x_{max} - x_{min}}$] Works when
- no outliers
- uniform distribution.

② Method 2: $x' = \frac{x - \mu}{\sigma}$] z score normalization

③ Method 3: $x' = \log(x)$] log scaling — for when you plot the feature distribution & it follows the power law.

Now some features are more important.

↳ weighted distance.

$$d(x,y) = \left[\sum_{k=1}^{L} w_k (x_k - y_k)^r\right]^{1/2}$$

(k^{th} dimension has weight w_k.)

You can have non-metric similarity measures too.

① Cosine similarity

$$S(x,y) = \frac{\bar{x}^T \bar{y}}{|x|\,|y|} \rightarrow d(x,y) = 1 - S(x,y)$$

② Measure of distance b/w two strings. → min no. of mutations (insert, delete, change) to transform S1 to S2

Levenshtein / Edit distance.

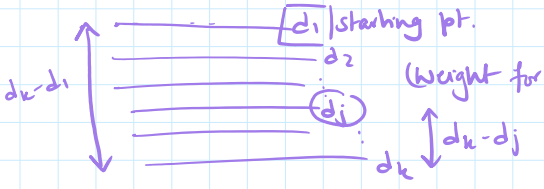
CAT
RAT ← C→R } one change

③ Hamming distance: no. of mismatched bits b/w two binary patterns.

Variations of KNN — weighted KNN (Variation 1)

↳ weighted majority rule — nearest neighbor = highest value (1)

farthest neighbor = lowest value (0)



(weight for neighbor j) $w_j = \frac{d_k - d_j}{d_k - d_1}$

d_k — distance b/w farthest & neighbor j

d_1 — distance b/w farthest and closest

↳ if $d_k \neq d_1$

$d_k - d_1$ — distance b/w farthest and closest
d_k d_1
↳ if $d_k \neq d_1$
↳ if $d_k = d_1$, your farthest is your closest, so 0.

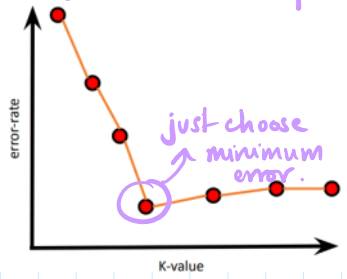
Eliminating outliers. (Variation 2)
- take point p
- take radius r (hyperparam)
- if anything outside radius r, outlier
- output majority class.

if empty subset, consider full dataset.

Assumption = "Similar points are closer together" = Inductive bias. → Set of assumptions that learner uses to predict o/p of i/p it has NOT encountered.

Algorithm:
1. Load data
2. Initialize k to chosen no. of neighbors — NOT even — cuz chances of ties.
3. For each example in data:
   a. Calculate dist. b/w query + current example.
   b. add distance + index of example to ordered collection
4. Sort the collection of distances + indices — ascending.
5. Pick first k entries.
6. If regression:
   return mean
   If classification:
   return mode.

But how would you choose k?
ELBOW METHOD.
Plot error vs. k-value. Choose optimal as elbow value of curve.



Low k: high variance + overfitting
High k: high bias + underfitting

Q1. Consider the following data set. Apply kNN algorithm to assess the risk for a patient whose BP is 100, sugar is 135, Haemoglobin is 12 and WBC count is 8 thousand. Take k =3 and use Euclidean distance measure.

#	B.P.	Sugar	Haemoglobin	WBC(in thousands)	Risk
A	100	120	12	6	No
B	110	130	14	5	Yes
C	120	110	11	7	Yes
D	100	140	13	7	No
E	115	140	11	6	Yes

$X_q = (100, 135, 12, 8)$
$k = 3$

Query	Datapoint	Euclidean distance
X_q	A	$\sqrt{(100-100)^2+(120-135)^2+(12-12)^2+(6-8)^2} = 15.13$
X_q	B	11.74
X_q	C	32.04
X_q	D	5.19
X_q	E	15.96

Now sort.

Query	Datapoint	Euclidean distance	label
X_q	D	5.19	N
X_q	B	11.74	Y
X_q	A	15.13	N
X_q	E	15.96	Y
X_q	C	52.04	Y

Mode = No.
So classify X_q as No.