Lecture 5 Feb 11, 2021

Why We Don't Use LS For Classification?

LS weights the model based on the sum of residuals. For the case of classification, we don't care about that distance, we only care if the point is being classified correctly or not.

K-NN Classification

Take the k (odd) nearest neighbours of x_{pred}

let $y_{pred} = \text{sign}\left(\sum_{N_k(x_{pred})} y_i\right)$.

Eg. the sign of the k-NN. K is odd for 2-D so there are no ties.

Problems with K-NN

- doesn't provide us with a way to understand how the data was generated or provide a model to generate new data.

- slower since we need to do calculation for every new prediction whereas logis. regreg. creates a model to interpolate from.
  - we need to find k-NN from all data set.

Curse of Dimentionality

Imagine a $c \times c$ box in $\mathbb{R}^2$ with k data points with data density $\frac{k}{c \times c}$.

If we were to increase the dimensionality from $\mathbb{R}^2$ to $\mathbb{R}^{10}$, we now have $\underbrace{c \times \dots \times c}_{10\times}$ box. To keep the same data density, we would need to have $\frac{k \times c^8}{c^{10}}$ or

c^8 times more data for only increasing 8 dimensions. A small increase in c in $\mathbb{R}^2$ would not affect the amount of data needed to match the density but a small increase in c in $\mathbb{R}^{10}$ would be huge. Eg $(c)^8$ vs. $(c+0.1)^8$

Solving Problems with K-NN

- Increase data size
- Decrease dimensionality $\Rightarrow$ Feature Extraction
  - Map $\mathbb{R}^n \longrightarrow \mathbb{R}^m$ $m < n$
  - Compute features and use those.
- Using weighted average

Decision Trees

Given training data set with features, create a model (tree) where each node performs a test

finally resulting in classification at the end of the tree.

How to choose good splits?

At each node:

1) keep track of which data points belong there
2) the histogram of class distributions

Until all nodes are leaf nodes:

1) pick a non-leaf node
2) choose a test to apply on this node
3) split data into subsets based on test
4) insert nodes for each new subset in the tree

Leaf node:

1) all data points are those of the same class
2) to little data
3) all points have the same features

How to quantify uncertainty?

We use entropy.

$$H = -\sum_{c=1}^{k} p_c \log p_c$$

Entropy is maximized when classes are uniformly distributed.

How to choose a test?

Given subset of data

- choose random feature
- run random test from test bank
- split data and evaluate reduction of uncertainty
    - does test reduce entropy?

How to avoid overfitting?

Decision trees can perfectly fit training data when no limits are in place.

To avoid overfitting:

- limit height of the tree
- limit minimum size of a node's data set

Random Tree Forests (OP)

Build large number of decision trees

- each with randomized subset of training data
- limit height and minimum size of node's date set for each tree
- each tree will have randomized tests,

Bootstrap aggregating

To classify new data point,

- run it through forest
- choose majority class