K-Nearest Neighbours

Remarks

Procheta Sen

UNIVERSITY OF LIVERPOOL

Complexity of k-NN

Training dataset: n objects and d features

Time: Just store the data points (constant time for each object)
Space: size of the training data ($n \times d$)

- **Training**: store entire training set

- **Classification**: for an input object $\overline{X}'$,
  - find in the training set the **k closest (nearest) objects** to $\overline{X}'$
  - find the majority label among those nearest neighbours
  - the majority label is predicted to the test instances $\overline{X}'$

For a test example we need to find k nearest neighbours.

The test example needs to be compared with every single training example

Complexity of k-NN

- **Training** is computationally cheap
- **Classification** is computationally expensive

In practice the **classification** time is much more important than the **training** time

Possible solution: use Approximate Nearest Neighbor (ANN) algorithms that can accelerate the nearest neighbor lookup in a dataset at the expense of accuracy of the queries (see e.g. FLANN — Fast Library for Approximate Nearest Neighbors).

Inductive bias and feature importance

k-NN classifier assumes

- that nearby points should have the same label

- all features are equally important! (if there are only a few relevant and many irrelevant features in a dataset, the k-NN classifier is likely to do poorly)

Summary on k-NN classifier.

- Preprocess your data: normalise the features in your dataset to have zero mean and unit variance.
- If your data is very high-dimensional, consider using a dimensionality reduction
- Split your training data into training (50-90%) and validation (10-50%).
- If the validation data set is too small split your training data into folds and do cross-validation.
- Train and evaluate the k-NN classifier on the validation data for many choices of k and for different types of distances (start with L^1 and L^2).