02-KNN

Ethan Graham

20th June 2023

Basic Idea

Given a point x to be classified, find the k-nearest neighbors to it in the training set and find the max label from amongst them. This becomes x's predicted label.

Voronoi Cells

Given N training samples, we get N voronoi cells. They are locations that will be associated to each training point. The shape of these cells is dependant on the distance metric that we use (for example Euclidean distance or Manhattan distance).
$C_n = \{x \in X | \forall j \neq n, d(x, x_n) \leq d(x, x_j)\}$ Basically "all points that are closer to me than anyone else"

The desicion boundary is formed by selected edges of the Voronoi Diagram.

NN properties

1-NN is very sensitive to outliers. A point too close to an outlier will be misclasified. This is why we introduce k-NN so that we look at multiple neighbors instead of just one. -> **give the majority label** K-NN in particular is prone to misclassifying points near the decision boundary. If there is an unbalanced training set, then the better represented class is unfairly favoured over the other classes

Key Assumption:

We assums that the training set and test set are drawn from the same statistical distribution. Otherwise there is no reason for a decision boundary learned on the training set to be useful on the test set. *For example, if we are doing face detection. The training set must be representative of all faces the system is likely to encounter. Otherwise it might not be accurate for certain predictions*.

Overfitting and K-NN

- **k=1** we get perfect accuracy on the training set. Test set isn't great yet. This is overfitting as we fit exactly to the training set!
- **k=3** the boundary becomes smoother. Training error increases but testing error decreases.
- **k=7** ditto
- **k=21** The testing error decreases, but would increase if we chose an even larger k.

Moral of the story, choose k to avoid overfitting. **split the training set into a real training set and a validation set. Choose k that minimizes the classification error on the validation set**

This is known as **cross-validation** and is used in most supervised learning algorithms.

Recall underfitting/overfitting from nummet. Remember the Runge curve While the training error curve always decreases as the degree

Notes on training K-NN

We should run different values of K and use only training and validation sets during the training stage. We pick the value of k that yields the highest accuracy on the validation set.

KNN is simple but effective (96.8% accuracy on MNIST) and only has one meta-parameter k which can be tuned.

Data reduction

Instead of using all points, compute centers of gravity instead. This works in some cases, but not all *(the donut case for example where clases share a center of gravity)*.

Condensed NN

Let x be the set of training samples and P the set of Prototypes.

- `initialize`
- `repeat:`
    1. `look for x in X such that its nearest prototype in P has a different label than itself`
    2. `remove x from X and add it to P`

Application of K-NN

Recommender systems could use for example, where you are matched to other users with similar interests. Note that the complexity is $O(n^2)$ since we must check neighbors for all users in the recommender system. **KNN does not scale well at all**.

We can define a gossip-based algorithm that is highly parallelisable. Among random nodes, among friedns of friends etc... This is particularly well-adapted to peer-to-peer networks, and is robust to churn, partition, breakdowns.

In practice, between 25 and 100 neighbors are used.

Final words

KNN works well but isn't very scalable and therefore it is subject to performance issues.

The distance metric commonly used is Euclidean distance which becomes problemtic in high-dimensional space *(recall curse of dimensionality)*. In high-dimensional space, everything is far away from everything else with common distance metrics. However, real data often lies in a much smaller subspace. We call this dimesionality reduction. Data exists on a low-dimensional manifold. *example: faces are only a tiny subset of possible images that we can take with a camera*