Nearest Neighbor Algorithm

Version 1.0, August 2024

1 Nearest neighbor algorithm

1.1 Remember all training examples

- Find the nearest training example to it using a distance measure
- The class label of the nearest neighbor will be the predicated label for the new example

1.2 Computational complexity in Classification

- Compare each unseen example with each training example
- if m training examples with dimensionality n, lookup for 1 unseen example takes $m \times n$ computations, i.e. $O(mn)$

1.3 Decision boundary of 1-nearest neighbor

- Nearest neighbor classification produced decision boundaries with an arbitrary shape
- The 1-nearest neighbor boundary is formed by the edges of the Voronoi diagram that separate the points of the two classes
- Voronoi region: Each training example has an associated Voronoi region; it contains the data points for which this is the close example

2 K-nearest neighbor

2.1 K-nearest neighbor

- K-nearest Neighbor is very sensitive to the value of k.
  - rule of thumb: $k \leq \sqrt{m}$, where m is the number of training examples
  - commercial package typically use $k = 10$
- Using more nearest neighbors increases the robustness to noisy examples.
- It can be used not only for classification, but also for regression. The prediction will be the average value of the class values of the k nearest neighbors

Algorithm 1: K-nearest Neighbor Algorithm

Data: training data: X_t; training label y_t; test data X_e; nearest number k

Result: test label: y_e

Compute distance $d = \sqrt{(X_e - X_t)^2}$;

Ranking the distance from small to large: $sort(d)$;

Pick the first k points, s.t. $P_x = min_k(X_t)$;

Map the points $P_y = y_t[Px]$;

Give the output as $y_e = count_max(P_y)$;

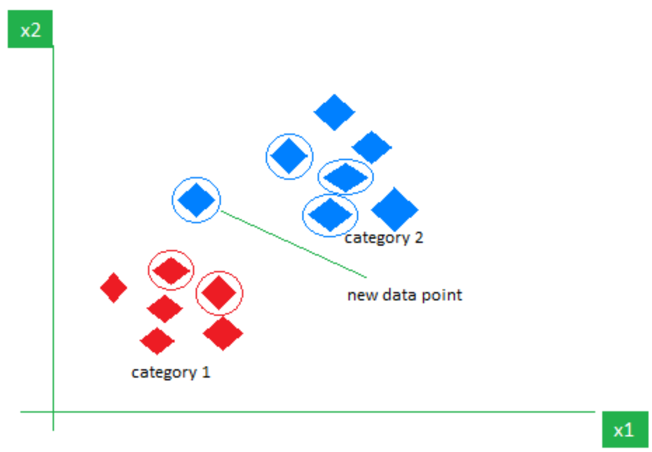

Figure 1: Schematic diagram of k-nearest neighbor algorithm

2.2 Weighted nearest neighbor

- idea: Closer neighbors should count more than distant neighbors
- Distance-weighted nearest -neighbor algorithm
  - bigger wight if they are closer
  - smaller wight if they are further(i.e. $w = \frac{1}{d^2}$)

Algorithm 2: Weighted nearest neighbor Algorithm

Data: training data: X_t; training label y_t; test data X_e; nearest number k

Result: test label: y_e

Compute distance $d = \sqrt{(X_e - X_t)^2}$;

Ranking the distance from small to large: $sort(d)$;

Pick the first k points, s.t. $P_x = min_k(X_t)$;

Map the points $P_y = y_t[Px]$;

Give the output as $y_e = max \sum w * P_y$;