Introduction to Machine Learning in Geosciences GEO371T/GEO39D.1

K Nearest Neighbor

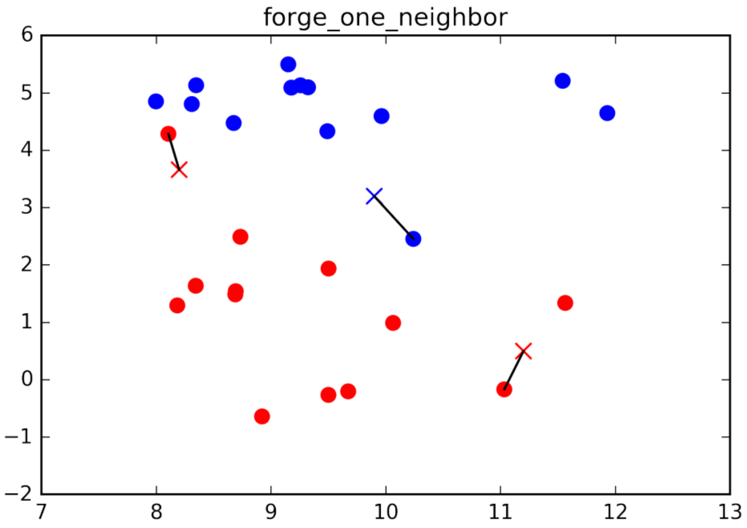
Mrinal K. Sen
Geosciences
UT Austin

October 12, 2023

Introduction

- Arguably the **simplest supervised machine learning** algorithm.
- Building the model only consists of storing the training dataset.
- To make a prediction for a new data point, the algorithm finds the closest data points in the training dataset, its “nearest neighbors”.
- In its simplest version, the algorithm only considers exactly one nearest neighbor, which is the closest training data point to the point we want to make a prediction for.
- The prediction is then simply the known output for this training point.

Introduction

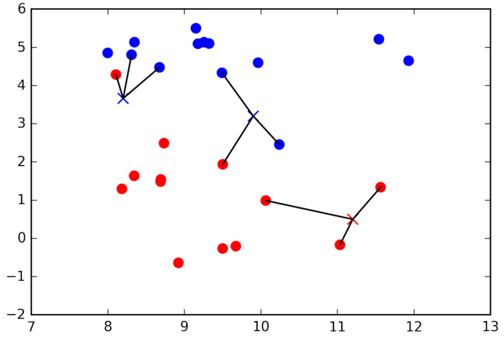


source:

Muller and Guido:Binary Classification

Introduction

- For more classes, we count how many neighbors belong to each class, and again predict the most common class.



source: Muller and Guido:Binary Classification

The Algorithm

- We do not make any assumption about the functional form of the kNN algorithm, a kNN model is also considered a **non-parametric** model.
- kNN does not have an explicit training step and defers all of the computation until prediction,
- Since the prediction is based on a comparison of a query point with data points in the training set (rather than a global model), kNN is also categorized as **instance-based** (or "memory-based") method.
- While kNN is a lazy instance-based learning algorithm, an example of an eager instance-based learning algorithm would be the **support vector machine**.

`https://github.com/rasbt/stat479-machine-learning-fs19/blob/master/02_knn/02-knn__notes.pdf`

Parametric vs. Non-parametric Models

- **Parametric**
  - Fixed Number of Parameters.
- **Non-parameteric**
  - No fixed number of parameters
  - parameters may grow with the number of training data points. Non-parametric does not mean no parameters!

The Algorithm

NN Classifier

Training Algorithm

```
for $i = 1, 3, \dots, n$
    store training examples $< \mathbf{x}^i, \mathbf{y}^i >$
```

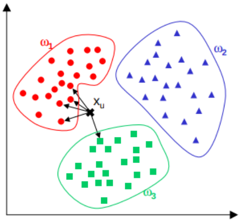
Prediction Algorithm

```
For a given $\mathbf{x}^k$
closest_point = none
closest_distance = $\infty$
for i=1,2,...,n:
    current_distance = $||\mathbf{x}^i - \mathbf{x}^k||_2$
    if current_distance < closest_distance:
        closest_distance = current_distance
        closest_point = $\mathbf{x}^i$
```

https://github.com/rasbt/stat479-machine-learning-fs19/blob/master/02_knn/02-knn__notes.pdf

Example

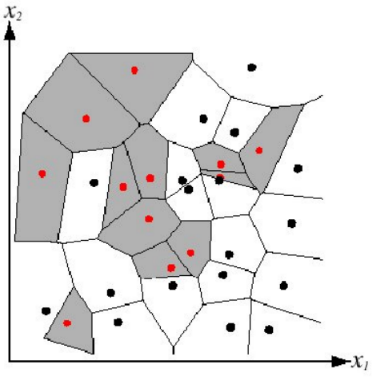
- The k-NN only requires
  - An integer k
  - A set of labeled examples (training data)
  - A metric to measure “closeness”
- 3 classes and goal: find a class label for the unknown X_u.
- The example uses k=5 neighbors and the Euclidean distance.
- Of the 5 closest neighbors, 4 belong to W1 and 1 belongs to W3, so the test point is assigned to W1, the predominant class.



source: Ricardo Gutierrez-Osuna, Wright State University

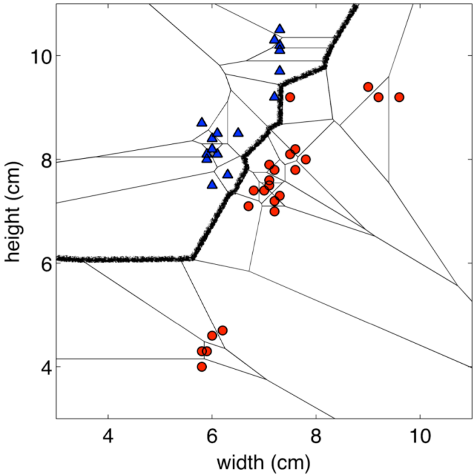
Decision Boundarlies

- Nearest neighbor algorithm does not explicitly compute decision boundaries, but these can be inferred.
- **Voronoi diagram** visualization



https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec5_handout.pdf

Decision Boundarlies



https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec5_handout.pdf

k-Nearest Neighbors

- Nearest neighbors sensitive to noise or mis-labeled data ("class noise"). Solution?
- Smooth by having k nearest neighbors vote

Algorithm (kNN):

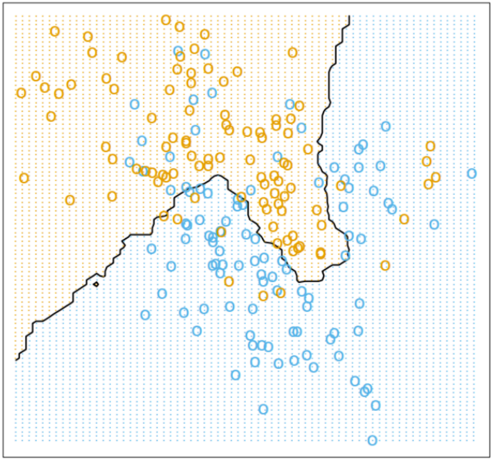
1. Find k examples $\{\mathbf{x}^{(i)}, t^{(i)}\}$ closest to the test instance $\mathbf{x}$
2. Classification output is majority class

$$y = arg \max_{t^{(z)}} \sum_{r=1}^{k} \delta(t^{(z)}, t^{(r)})$$

`https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec5_handout.pdf`

k-Nearest Neighbors

k=15



[Image credit: "The Elements of Statistical Learning"]

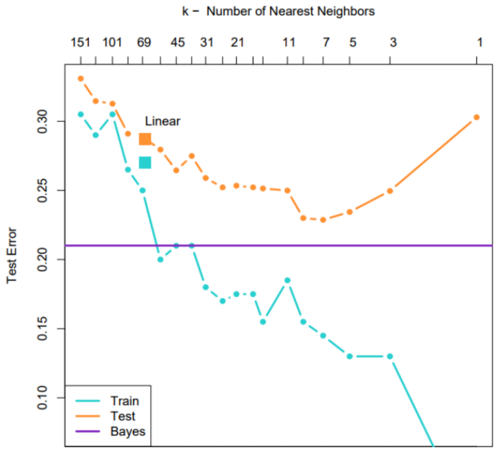
https://www.cs.toronto.edu/ jlucas/teaching/csc411/lectures/lec5$_h$*andout.pdf*

k-Nearest Neighbors

How do we choose k?

- Larger k may lead to better performance.
- But if we set k too large we may end up looking at samples that are not neighbors (are far away from the query)
- We can use validation set / cross-validation to find k.
- Rule of thumb is $k < \sqrt{n}$, where n is the number of training examples.

k-Nearest Neighbors



[Image credit: "The Elements of Statistical Learning"]

https://www.cs.toronto.edu/~jlucas/teaching/csc411/lectures/lec5_handout.pdf

k-Nearest Neighbors: Regression

- The general concept of kNN for regression is the same as for classification.
- We find the k nearest neighbors in the dataset.
- we find the k nearest neighbors in the dataset.
- In regression, the target function is a real- instead of discrete-valued function.
- A common approach for computing the continuous target is to compute the mean or average target value over the k nearest neighbors

Curse of Dimensionality

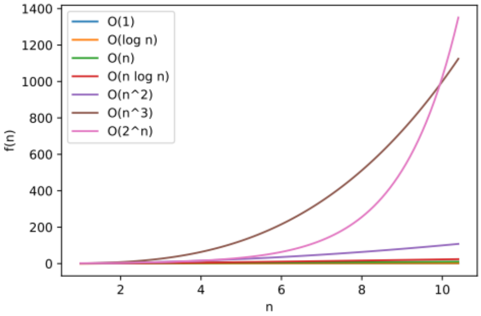
- The curse of dimensionality refers to scenarios with a fixed number of training examples but an increasing number of dimensions and range of feature values in each dimension in a high-dimensional feature space.
- The kNN algorithm is particularly susceptible to the curse of dimensionality.
- In kNN an increasing number of dimensions becomes increasingly problematic because the more dimensions we add, the larger the volume in the hyperspace needs to be to capture a fixed number of neighbors.
- s the volume grows larger and larger, the "neighbors" become less and less "similar" to the query point as they are now all relatively distant from the query point considering all different dimensions that are included when computing the pairwise distances!
- A common approach for computing the continuous target is to compute the mean or average target value over the k nearest neighbors

Computational Complexity and the Big-O Notation

- The curse of dimensionality refers to scenarios with a fixed number of training examples but an increasing number of dimensions and range of feature values in each dimension in a high-dimensional feature space.
- The kNN algorithm is particularly susceptible to the curse of dimensionality.
- In kNN an increasing number of dimensions becomes increasingly problematic because the more dimensions we add, the larger the volume in the hyperspace needs to be to capture a fixed number of neighbors.
- s the volume grows larger and larger, the "neighbors" become less and less "similar" to the query point as they are now all relatively distant from the query point considering all different dimensions that are included when computing the pairwise distances!
- A common approach for computing the continuous target is to compute the mean or average target value over the k nearest neighbors

Computational Complexity and the Big-O Notation

For the brute-force neighbor search of the kNN algorithm, we have a time complexity of O(nm), where n is the number of training examples and m is the number of dimensions in the training set. For simplicity, assuming $n >> m$, the complexity of the brute-force nearest neighbor search is O(n).



Sebastian Raschka STAT479 FS19. L02: Nearest Neighbor Methods

Knn: Shortcomings

- If some attributes (coordinates of x) have larger ranges, they are treated as more important. Normalize scale.
- Irrelevant, correlated attributes add noise to distance measure. eliminate some attributes or vary and possibly adapt weight of attributes
- High Dimensional Data: "Curse of Dimensionality" - Computational cost also increases!
- Expensive at test time. Use subset of dimensions; Remove redundant data.
- Storage Requirements!