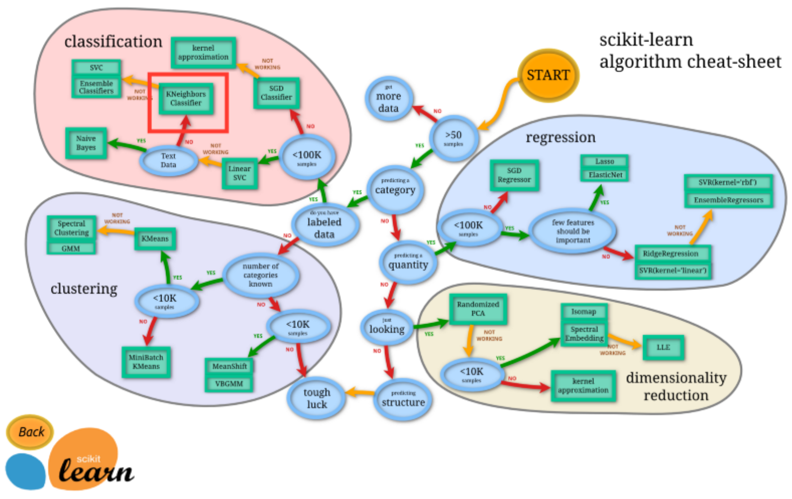
A Quick Introduction to K-Nearest Neighbors Algorithm

Adi Bronshtein Follow
Apr 11, 2017 · 4 min read

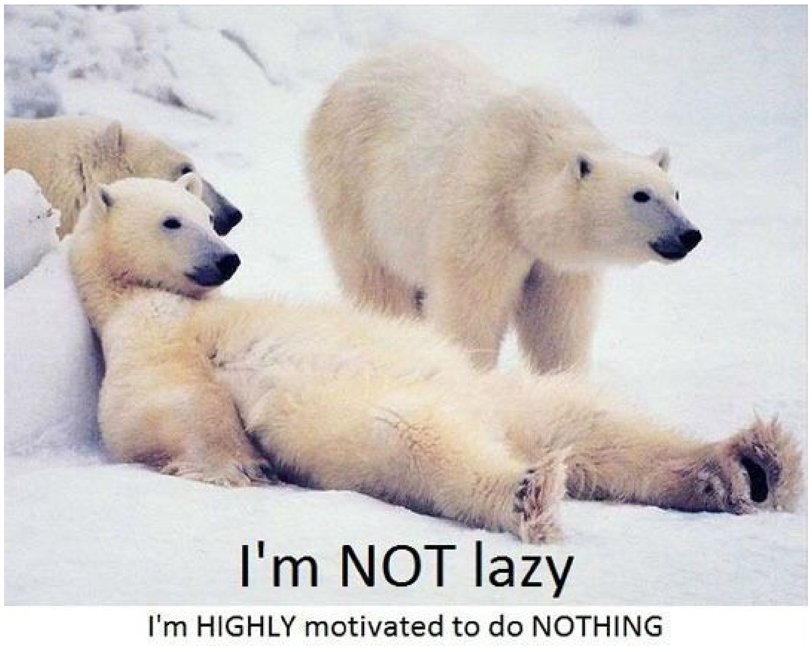
Unfortunately, it's not that kind of neighbor! :)

Hi everyone! Today I would like to talk about the K-Nearest Neighbors algorithm (or KNN). KNN algorithm is one of the simplest classification algorithm and it is one of the most used learning algorithms. So what is the KNN algorithm? I'm glad you asked! **KNN** is a **non-parametric, lazy** learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.

Just for reference, this is "where" KNN is positioned in the algorithm list of scikit learn. BTW, scikit-learn documentation (clickable link) is amazing! Worth a read if you're interested in machine learning.

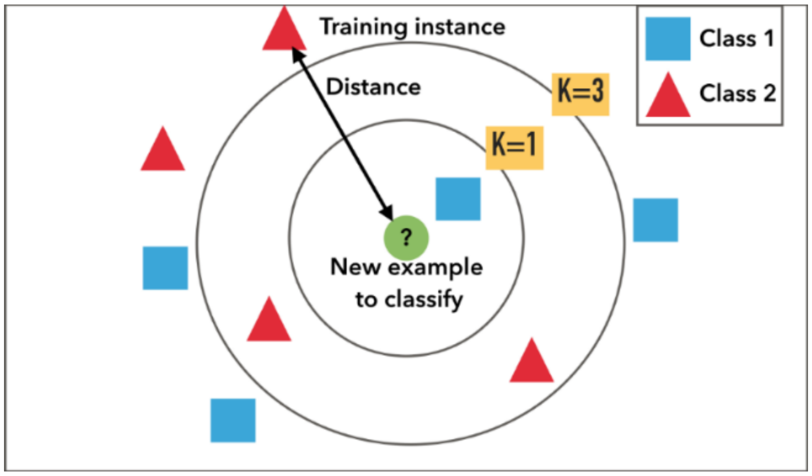


When we say a technique is **non-parametric** , it means that it does not make any assumptions on the underlying data distribution. In other words, the model structure is determined from the data. If you think about it, it's pretty useful, because in the "real world", most of the data does not obey the typical theoretical assumptions made (as in linear regression models, for example). Therefore, KNN could and probably should be one of the first choices for a classification study when there is little or no prior knowledge about the distribution data.



KNN is also a **lazy** algorithm (as opposed to an *eager* algorithm). Does that mean that KNN does nothing, like these polar bears imply??? Not quite. What this means is that it does not use the training data points to do any *generalization*. In other words, there is *no explicit training phase* or it is very minimal. This also means that the training phase is pretty fast . Lack of generalization means that KNN keeps all the training data. To be more exact, all (or most) the training data is needed during the testing phase.

KNN Algorithm is based on **feature similarity**: How closely out-of-sample features resemble our training set determines how we classify a given data point:



Example of k-NN classification. The test sample (inside circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (outside circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If, for example k = 5 it is assigned to the first class (3 squares vs. 2 triangles outside the outer circle).

KNN can be used for **classification**—the output is a class membership (predicts a class—a discrete value). An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. It can also be used for **regression**—output is the value for the object (predicts continuous values). This value is the average (or median) of the values of its k nearest neighbors.

. . .

A few Applications and Examples of KNN

- Credit ratings—collecting financial characteristics vs. comparing people with similar financial features to a database. By the very nature of a credit rating, people who have similar financial details would be given similar credit ratings. Therefore, they would like to be able to use this existing database to predict a new customer's credit rating, without having to perform all the calculations.

- Should the bank give a loan to an individual? Would an individual default on his or her loan? Is that person closer in characteristics to people who defaulted or did not default on their loans?
- In political science—classing a potential voter to a "will vote" or "will not vote", or to "vote Democrat" or "vote Republican".
- More advance examples could include handwriting detection (like OCR), image recognition and even video recognition.

. . .

Some pros and cons of KNN

Pros:

- No assumptions about data—useful, for example, for nonlinear data
- Simple algorithm—to explain and understand/interpret
- High accuracy (relatively)—it is pretty high but not competitive in comparison to better supervised learning models
- Versatile—useful for classification or regression

Cons:

- Computationally expensive—because the algorithm stores all of the training data
- High memory requirement
- Stores all (or almost all) of the training data
- Prediction stage might be slow (with big N)
- Sensitive to irrelevant features and the scale of the data

. . .

Quick summary of KNN

The algorithm can be summarized as:

1. A positive integer k is specified, along with a new sample
2. We select the k entries in our database which are closest to the new sample
3. We find the most common classification of these entries
4. This is the classification we give to the new sample

A few other features of KNN:

- KNN stores the entire training dataset which it uses as its representation.
- KNN does not learn any model.
- KNN makes predictions just-in-time by calculating the similarity between an input sample and each training instance.

I hope this helps a little in understanding what the K-Nearest Neighbor algorithm is. As always, I welcome questions, notes, suggestions etc. See you next time!