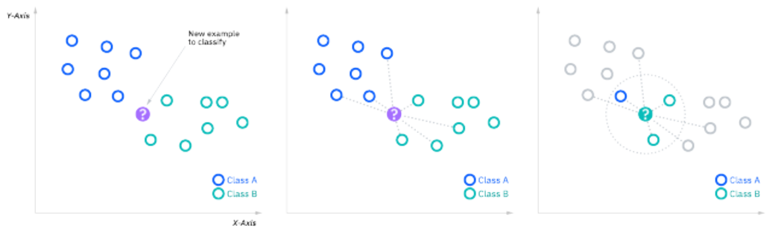
is the k-nearest
oors algorithm?

e k-nearest neighbors algorithm, one
and simplest classification and
sifiers used in machine learning

K-Nearest Neighbors Algorithm

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

For classification problems, a class label is assigned on the basis of a majority vote—i.e. the label that is most frequently represented around a given data point is used. While this is technically considered “plurality voting”, the term, “majority vote” is more commonly used in literature. The distinction between these terminologies is that “majority voting” technically requires a majority of greater than 50%, which primarily works when there are only two categories. When you have multiple classes—e.g. four categories, you don’t necessarily need 50% of the vote to make a conclusion about a class; you could assign a class label with a vote of greater than 25%. The University of Wisconsin-Madison summarizes this well with an example here (PDF, 1.2 MB) (link resides outside of ibm.com).



KNN diagram

Regression problems use a similar concept as classification problem, but in this case, the average the k nearest neighbors is taken to make a prediction about a classification. The main distinction here is that classification is used for discrete values, whereas regression is used with continuous ones. However, before a classification can be made, the distance must be defined. Euclidean distance is most commonly used, which we'll delve into more below.
It's also worth noting that the KNN algorithm is also part of a family of "lazy learning" models, meaning that it only stores a training dataset versus undergoing a training stage. This also means that all the computation occurs when a classification or prediction is being made. Since it heavily relies on memory to store all its training data, it is also referred to as an instance-based or memory-based learning method.
Evelyn Fix and Joseph Hodges are credited with the initial ideas around the KNN model in this 1951 paper (PDF, 1.1 MB) (link resides outside of ibm.com) while Thomas Cover expands on their concept in his research (PDF 1 MB) (link resides outside of ibm.com), "Nearest Neighbor Pattern Classification." While it's not as popular as it once was, it is still one of the first algorithms one learns in data science due to its simplicity and accuracy. However, as a dataset grows, KNN becomes increasingly inefficient, compromising overall model performance. It is commonly used for simple recommendation systems, pattern recognition, data mining, financial market predictions, intrusion detection, and more.

Compute KNN: distance metrics

To recap, the goal of the k-nearest neighbor algorithm is to identify the nearest neighbors of a given query point, so that we can assign a class label to that point. In order to do this, KNN has a few requirements:

Determine your distance metrics

In order to determine which data points are closest to a given query point, the distance between the query point and the other data points will need to be calculated. These distance metrics help to form decision boundaries, which partitions query points into different regions. You commonly will see decision boundaries visualized with Voronoi diagrams.

While there are several distance measures that you can choose from, this article will only cover the following:

Euclidean distance (p=2): This is the most commonly used distance measure, and it is limited to real-valued vectors. Using the below formula, it measures a straight line between the query point and the other point being measured.

Euclidean distance formula

Manhattan distance (p=1): This is also another popular distance metric, which measures the absolute value between two points. It is also referred to as taxicab distance or city block distance as it is commonly visualized with a grid, illustrating how one might navigate from one address to another via city streets.

Manhattan distance formula

Minkowski distance: This distance measure is the generalized form of Euclidean and Manhattan distance metrics. The parameter, p, in the formula below, allows for the creation of other distance metrics. Euclidean distance is represented by this formula when p is equal to two, and Manhattan distance is denoted with p equal to one.

Minkowski distance formula

Hamming distance: This technique is used typically used with Boolean or string vectors, identifying the points where the vectors do not match. As a result, it has also been referred to as the overlap metric. This can be represented with the following formula:

Hamming distance formula

As an example, if you had the following strings, the hamming distance would be 2 since only two of the values differ.

Hamming distance example

Compute KNN: defining k

The k value in the k-NN algorithm defines how many neighbors will be checked to determine the classification of a specific query point. For example, if k=1, the instance will be assigned to the same class as its single nearest neighbor. Defining k can be a balancing act as different values can lead to overfitting or underfitting. Lower values of k can have high variance, but low bias, and larger values of k may lead to high bias and lower variance. The choice of k will largely depend on the input data as data with more outliers or noise will likely perform better with higher values of k. Overall, it is recommended to have an odd number for k to avoid ties in classification, and cross-validation tactics can help you choose the optimal k for your dataset.

k-nearest neighbors and python

To delve deeper, you can learn more about the k-NN algorithm by using Python and scikit-learn (also known as sklearn). Our tutorial in Watson Studio helps you learn the basic syntax from this library, which also contains other popular libraries, like NumPy, pandas, and Matplotlib. The following code is an example of how to create and predict with a KNN model:

```
from sklearn.neighbors import KNeighborsClassifier
model_name = ‘K-Nearest Neighbor Classifier’
knnClassifier = KNeighborsClassifier(n_neighbors = 5, metric = ‘minkowski’, p=2)
knn_model = Pipeline(steps=[(‘preprocessor’, preprocessorForFeatures), (‘classifier’
, knnClassifier)])
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
```

Applications of k-NN in machine learning

The k-NN algorithm has been utilized within a variety of applications, largely within classification. Some of these use cases include:

- Data preprocessing: Datasets frequently have missing values, but the KNN algorithm can estimate for those values in a process known as missing data imputation.

- Recommendation Engines: Using clickstream data from websites, the KNN algorithm has been used to provide automatic recommendations to users on additional content. This research (link resides outside of ibm.com) shows that the a user is assigned to a particular group, and based on that group's user behavior, they are given a recommendation. However, given the scaling issues with KNN, this approach may not be optimal for larger datasets.

- Finance: It has also been used in a variety of finance and economic use cases. For example, one paper (PDF, 391 KB) (link resides outside of ibm.com) shows how using KNN on credit data can help banks assess risk of a loan to an organization or individual. It is used to determine the credit-worthiness of a loan applicant. Another journal (PDF, 447 KB)(link resides outside of ibm.com) highlights its use in stock market forecasting, currency exchange rates, trading futures, and money laundering analyses.

- Healthcare: KNN has also had application within the healthcare industry, making predictions on the risk of heart attacks and prostate cancer. The algorithm works by calculating the most likely gene expressions.

- Pattern Recognition: KNN has also assisted in identifying patterns, such as in text and digit classification (link resides outside of ibm.com). This has been particularly helpful in identifying handwritten numbers that you might find on forms or mailing envelopes.

Advantages and disadvantages of the KNN algorithm

Just like any machine learning algorithm, k-NN has its strengths and weaknesses. Depending on the project and application, it may or may not be the right choice.

Advantages

- Easy to implement: Given the algorithm's simplicity and accuracy, it is one of the first classifiers that a new data scientist will learn.

- Adapts easily: As new training samples are added, the algorithm adjusts to account for any new data since all training data is stored into memory.

- Few hyperparameters: KNN only requires a k value and a distance metric, which is low when compared to other machine learning algorithms.

Disadvantages

- Does not scale well: Since KNN is a lazy algorithm, it takes up more memory and data storage compared to other classifiers. This can be costly from both a time and money perspective. More memory and storage will drive up business expenses and more data can take longer to compute. While different data structures, such as Ball-Tree, have been created to address the computational inefficiencies, a different classifier may be ideal depending on the business problem.

- Curse of dimensionality: The KNN algorithm tends to fall victim to the curse of dimensionality, which means that it doesn't perform well with high-dimensional data inputs. This is sometimes also referred to as the peaking phenomenon (PDF, 340 MB) (link resides outside of ibm.com), where after the algorithm attains the optimal number of features, additional features increases the amount of classification errors, especially when the sample size is smaller.

- Prone to overfitting: Due to the "curse of dimensionality", KNN is also more prone to overfitting. While feature selection and dimensionality reduction techniques are leveraged to prevent this from occurring, the value of k can also impact the model's behavior. Lower values of k can overfit the data, whereas higher values of k tend to "smooth out" the prediction values since it is averaging the values over a greater area, or neighborhood. However, if the value of k is too high, then it can underfit the data.

Related solutions

IBM Cloud Pak for Data

IBM Cloud Pak for Data is an open, extensible data platform that provides a data fabric to make all data available for AI and analytics, on any cloud.

Explore IBM Cloud Pak for Data →

IBM Watson Studio

Build, run and manage AI models. Prepare data and build models on any cloud using open source code or visual modeling. Predict and optimize your outcomes.

Explore IBM Watson Studio →

IBM Db2 on Cloud

Learn about Db2 on Cloud, a fully managed SQL cloud database configured and optimized for robust performance.

Explore IBM Db2 on Cloud →

Resources

Background of KNN →

Usage of KNN →

Functions for KNN →

Next steps

k-NN Node and IBM Cloud Pak for Data
The Cloud Pak for Data is a set of tools that helps to prepare data for AI implementation. k-NN node is a modeling method available in the IBM Cloud Pak for Data, which makes developing predictive models very easy. The plugin deploys on any cloud and integrates seamlessly into your existing cloud infrastructure.

To learn more about k-NN, sign up for an IBMid and create your IBM Cloud account. →

Let’s talk