k-nearest neighbors algorithm

From Wikipedia, the free encyclopedia

In pattern recognition, the ***k*-Nearest Neighbors algorithm** (or ***k*-NN** for short) is a non-parametric method used for classification and regression.[1] In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression:

- In *k-NN classification*, the output is a class membership. 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 (*k* is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.
- In *k-NN regression*, the output is the property value for the object. This value is the average of the values of its *k* nearest neighbors.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all machine learning algorithms.

Both for classification and regression, it can be useful to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of $1/d$, where d is the distance to the neighbor.[2]

The neighbors are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

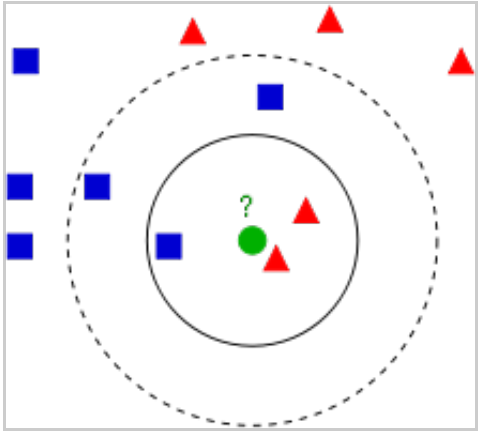
A shortcoming of the *k*-NN algorithm is that it is sensitive to the local structure of the data. The algorithm has nothing to do with and is not to be confused with *k*-means, another popular machine learning technique.

Contents

Algorithm

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point.



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

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the **overlap metric** (or Hamming distance). In the context of gene expression microarray data, for example, *k*-NN has also been employed with correlation coefficients such as Pearson and Spearman.[3] Often, the classification accuracy of *k*-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighbourhood components analysis.

A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the *k* nearest neighbors due to their large number.[4] One way to overcome this problem is to weigh the classification, taking into account the distance from the test point to each of its *k* nearest neighbors. The class (or value, in regression problems) of

each of the *k* nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. For example in a self-organizing map (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. *K*-NN can then be applied to the SOM.

Parameter selection

The best choice of *k* depends upon the data; generally, larger values of *k* reduce the effect of noise on the classification,[5] but make boundaries between classes less distinct. A good *k* can be selected by various heuristic techniques (see hyperparameter optimization). The special case where the class is predicted to be the class of the closest training sample (i.e. when $k = 1$) is called the nearest neighbor algorithm.

The accuracy of the *k*-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. Much research effort has been put into selecting or scaling features to improve classification. A particularly popular approach is the use of evolutionary algorithms to optimize feature scaling.[6] Another popular approach is to scale features by the mutual information of the training data with the training classes.

In binary (two class) classification problems, it is helpful to choose *k* to be an odd number as this avoids tied votes. One popular way of choosing the empirically optimal *k* in this setting is via bootstrap method.[7]

Properties

k-NN is a special case of a variable-bandwidth, kernel density "balloon" estimator with a uniform kernel.[8] [9]

The naive version of the algorithm is easy to implement by computing the distances from the test example to all stored examples, but it is computationally intensive for large training sets. Using an appropriate nearest neighbor search algorithm makes *k*-NN computationally tractable even for large data sets. Many nearest neighbor search algorithms have been proposed over the years; these generally seek to reduce the number of distance evaluations actually performed.

k-NN has some strong consistency results. As the amount of data approaches infinity, the algorithm is guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data).[10] *k*-NN is guaranteed to approach the Bayes error rate for some value of *k* (where *k* increases as a function of the number of data points). Various improvements to *k*-NN are possible by using proximity graphs.[11]

Metric learning

The K-nearest neighbor classification performance can often be significantly improved through (supervised) metric learning. Popular algorithms are neighbourhood components analysis and large margin nearest neighbor. Supervised metric learning algorithms use the label information to learn a new metric or pseudo-metric.

Feature extraction

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters) then the input data will be transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called feature extraction. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input. Feature extraction is performed on raw data prior to applying *k*-NN algorithm on the transformed data in feature space.

An example of a typical computer vision computation pipeline for face recognition using *k*-NN including feature extraction and dimension reduction pre-processing steps (usually implemented with OpenCV):

1. Haar face detection
2. Mean-shift tracking analysis
3. PCA or Fisher LDA projection into feature space, followed by *k*-NN classification

Dimension reduction

For high-dimensional data (e.g., with number of dimensions more than 10) dimension reduction is usually performed prior to applying the *k*-NN algorithm in order to avoid the effects of the curse of dimensionality.[12]

The curse of dimensionality in the *k*-NN context basically means that Euclidean distance is unhelpful in high dimensions because all vectors are almost equidistant to the search query vector (imagine multiple points lying more or less on a circle with the query point at the center; the distance from the query to all data points in the search space is almost the same).

Feature extraction and dimension reduction can be combined in one step using principal component analysis (PCA), linear discriminant analysis (LDA), or canonical correlation analysis (CCA) techniques as a pre-processing step, followed by clustering by *k*-NN on feature vectors in reduced-dimension space. In machine learning this process is also called low-dimensional embedding.[13]

For very-high-dimensional datasets (e.g. when performing a similarity search on live video streams, DNA data or high-dimensional time series) running a fast **approximate** *k*-NN search using locality sensitive hashing, "random projections",[14] "sketches" [15] or other high-dimensional similarity search techniques from the VLDB toolbox might be the only feasible option.

Decision boundary

Nearest neighbor rules in effect implicitly compute the decision boundary. It is also possible to compute the decision boundary explicitly, and to do so efficiently, so that the computational complexity is a function of the boundary complexity.[16]

Extension of *k-NN* (ENN) for classification

Unlike the classic *k-NN* methods in which only the nearest neighbors of an object are used to estimate its group membership, an extended *k-NN* method, termed ENN,[17] makes use of a *two-way communication* for classification: it considers not only who are the nearest neighbors of the test sample, but also who consider the test sample as their nearest neighbors. The idea of ENN method is to assign a group membership to an object by maximizing the *intra-class coherence*, which is a statistic measuring the coherence among all classes. Empirical studies have shown that ENN can significantly improve the classification accuracy in comparison with the *k-NN* method.

Data reduction

Data reduction is one of the most important problems for work with huge data sets. Usually, only some of the data points are needed for accurate classification. Those data are called the *prototypes* and can be found as follows:

1. Select the *class-outliers*, that is, training data that are classified incorrectly by *k*-NN (for a given *k*)
2. Separate the rest of the data into two sets: (i) the prototypes that are used for the classification decisions and (ii) the *absorbed points* that can be correctly classified by *k*-NN using prototypes. The absorbed points can then be removed from the training set.

Selection of class-outliers

A training example surrounded by examples of other classes is called a class outlier. Causes of class outliers include:

- random error
- insufficient training examples of this class (an isolated example appears instead of a cluster)
- missing important features (the classes are separated in other dimensions which we do not know)
- too many training examples of other classes (unbalanced classes) that create a "hostile" background for the given small class

Class outliers with *k*-NN produce noise. They can be detected and separated for future analysis. Given two natural numbers, $k>r>0$, a training example is called a *(k,r)*NN class-outlier if its *k* nearest neighbors include more than *r* examples of other classes.

CNN for data reduction

Condensed nearest neighbor (CNN, the *Hart algorithm*) is an algorithm designed to reduce the data set for *k*-NN classification.[18] It selects the set of prototypes *U* from the training data, such that 1NN with *U* can classify the examples almost as accurately as 1NN does with the whole data set.

Given a training set *X*, CNN works iteratively:

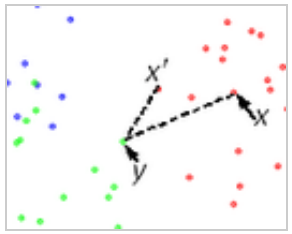
1. Scan all elements of *X*, looking for an element *x* whose nearest prototype from *U* has a different label than *x*.
2. Remove *x* from *X* and add it to *U*
3. Repeat the scan until no more prototypes are added to *U*.

Use *U* instead of *X* for classification. The examples that are not prototypes are called "absorbed" points.

It is efficient to scan the training examples in order of decreasing border ratio.[19] The border ratio of a training example x is defined as

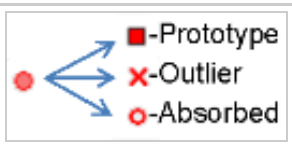
$$a(x) = \frac{||x'-y||}{||x-y||}$$

where $||x-y||$ is the distance to the closest example y having a different color than x, and $||x'-y||$ is the distance from y to its closest example x' with the same label as x.



Calculation of the border ratio.

The border ratio is in the interval [0,1] because $||x'-y||$never exceeds $||x-y||$. This ordering gives preference to the borders of the classes for inclusion in the set of prototypes$U$. A point of a different label than x is called external to x. The calculation of the border ratio is illustrated by the figure on the right. The data points are labeled by colors: the initial point is x and its label is red. External points are blue and green. The closest to x external point is y. The closest to y red point is x' . The border ratio $a(x) = ||x'-y|| / ||x-y||$is the attribute of the initial point x.



Three types of points: prototypes, class-outliers, and absorbed points.

Below is an illustration of CNN in a series of figures. There are three classes (red, green and blue). Fig. 1: initially there are 60 points in each class. Fig. 2 shows the 1NN classification map: each pixel is classified by 1NN using all the data. Fig. 3 shows the 5NN classification map. White areas correspond to the unclassified regions, where 5NN voting is tied (for example, if there are two green, two red and one blue points among 5 nearest neighbors). Fig. 4 shows the reduced data set. The crosses are the class-outliers selected by the (3,2)NN rule (all the three nearest neighbors of these instances belong to other classes); the squares are the prototypes, and the empty circles are the absorbed points. The left bottom corner shows the numbers of the class-outliers, prototypes and absorbed points for all three classes. The number of prototypes varies from 15% to 20% for different classes in this example. Fig. 5 shows that the 1NN classification map with the prototypes is very similar to that with the initial data set. The figures were produced using the Mirkes applet.[19]

CNN model reduction for k-NN classifiers

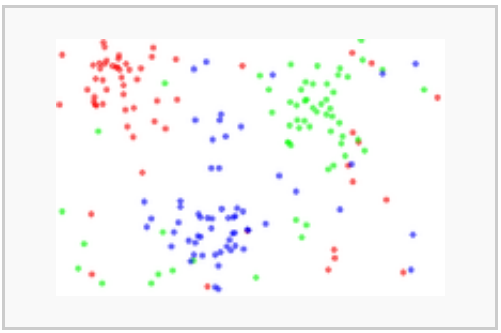


Fig. 1. The dataset.

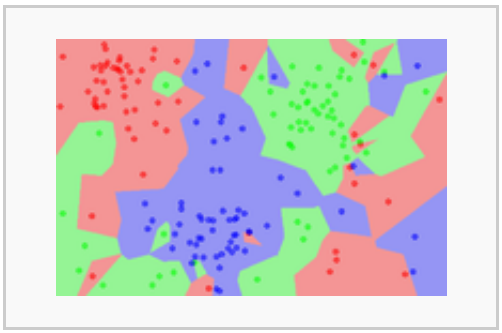


Fig. 2. The 1NN classification map.

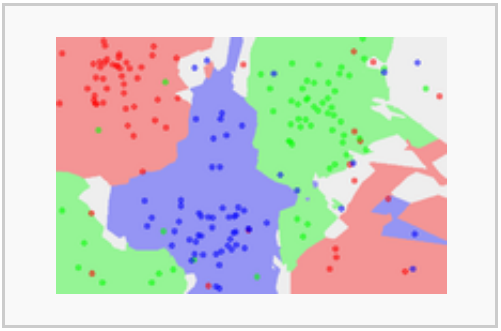


Fig. 3. The 5NN classification map.

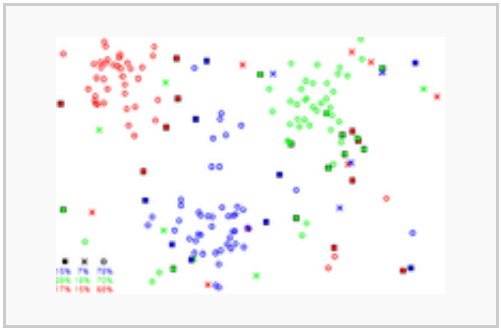


Fig. 4. The CNN reduced dataset.

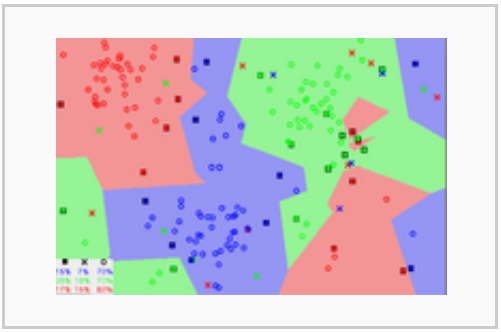


Fig. 5. The 1NN classification map based on the CNN extracted prototypes.

k-NN regression

In *k*-NN regression, the *k*-NN algorithm is used for estimating continuous variables. One such algorithm uses a weighted average of the *k* nearest neighbors, weighted by the inverse of their distance. This algorithm works as follows:

1. Compute the Euclidean or Mahalanobis distance from the query example to the labeled examples.
2. Order the labeled examples by increasing distance.
3. Find a heuristically optimal number *k* of nearest neighbors, based on RMSE. This is done using cross validation.
4. Calculate an inverse distance weighted average with the *k*-nearest multivariate neighbors.

Validation of results

A confusion matrix or "matching matrix" is often used as a tool to validate the accuracy of *k*-NN classification. More robust statistical methods such as likelihood-ratio test can also be applied.

See also

- Nearest centroid classifier
- Closest pair of points problem

References

1. Altman, N. S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". *The American Statistician* **46** (3): 175–185. doi:10.1080/00031305.1992.10475879.
2. This scheme is a generalization of linear interpolation.
3. Jaskowiak, P. A.; Campello, R. J. G. B. "Comparing Correlation Coefficients as Dissimilarity Measures for Cancer Classification in Gene Expression Data". *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.208.993*. Brazilian Symposium on Bioinformatics (BSB 2011). pp. 1–8. Retrieved 16 October 2014. External link in |website= (help)
4. D. Coomans; D.L. Massart (1982). "Alternative k-nearest neighbour rules in supervised pattern recognition : Part 1. k-Nearest neighbour classification by using alternative voting rules". *Analytica Chimica Acta* **136**: 15–27. doi:10.1016/S0003-2670(01)95359-0.
5. Everitt, B. S., Landau, S., Leese, M. and Stahl, D. (2011) Miscellaneous Clustering Methods, in Cluster Analysis, 5th Edition, John Wiley & Sons, Ltd, Chichester, UK.
6. Nigsch F, Bender A, van Buuren B, Tissen J, Nigsch E, Mitchell JB (2006). "Melting point prediction employing k-nearest neighbor algorithms and genetic parameter optimization". *Journal of Chemical Information and Modeling* **46** (6): 2412–2422. doi:10.1021/ci060149f. PMID 17125183.
7. Hall P, Park BU, Samworth RJ (2008). "Choice of neighbor order in nearest-neighbor classification". *Annals of Statistics* **36** (5): 2135–2152. doi:10.1214/07-AOS537.
8. D. G. Terrell; D. W. Scott (1992). "Variable kernel density estimation". *Annals of Statistics* **20** (3): 1236–1265. doi:10.1214/aos/1176348768.
9. Mills, Peter. "Efficient statistical classification of satellite measurements". *International Journal of Remote Sensing*.
10. Cover TM, Hart PE (1967). "Nearest neighbor pattern classification". *IEEE Transactions on Information Theory* **13** (1): 21–27. doi:10.1109/TIT.1967.1053964.
11. Toussaint GT (April 2005). "Geometric proximity graphs for improving nearest neighbor methods in instance-based learning and data mining". *International Journal of Computational Geometry and Applications* **15** (2): 101–150. doi:10.1142/S0218195905001622.
12. Beyer, Kevin, et al.. 'When is “nearest neighbor” meaningful?. Database Theory—ICDT’99, 217-235|year 1999
13. Shaw, Blake, and Tony Jebara. 'Structure preserving embedding. Proceedings of the 26th Annual International Conference on Machine Learning. ACM,2009
14. Bingham, Ella, and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM | year 2001
15. Shasha, D High Performance Discovery in Time Series.Berlin: Springer, 2004, ISBN 0-387-00857-8
16. Bremner D, Demaine E, Erickson J, Iacono J, Langerman S, Morin P, Toussaint G (2005). "Output-sensitive algorithms for computing nearest-neighbor decision boundaries". *Discrete and Computational Geometry* **33** (4): 593–604. doi:10.1007/s00454-004-1152-0.
17. Tang B, He H (2015). "ENN: Extended Nearest Neighbor method for pattern recognition". *IEEE Computational Intelligence Magazine* **10** (3): 52–60. doi:10.1109/MCI.2015.2437512.
18. P. E. Hart, The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory 18 (1968) 515–516. doi: 10.1109/TIT.1968.1054155
19. E. M. Mirkes, KNN and Potential Energy: applet. (http://www.math.le.ac.uk/people/ag153/homepage/KNN/KNN3.html) University of Leicester, 2011.

Further reading

- When Is "Nearest Neighbor" Meaningful? (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.1422)
- Belur V. Dasarathy, ed. (1991). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. ISBN 0-8186-8930-7.
- Shakhnarovish, Darrell, and Indyk, ed. (2005). *Nearest-Neighbor Methods in Learning and Vision*.

MIT Press. ISBN 0-262-19547-X.
- Mäkelä H Pekkarinen A (2004-07-26). "Estimation of forest stand volumes by Landsat TM imagery and stand-level field-inventory data". *Forest Ecology and Management* **196** (2–3): 245–255. doi:10.1016/j.foreco.2004.02.049.
- Fast k nearest neighbor search using GPU. In Proceedings of the CVPR Workshop on Computer Vision on GPU, Anchorage, Alaska, USA, June 2008. V. Garcia and E. Debreuve and M. Barlaud.
- Scholarpedia article on *k*-NN (http://www.scholarpedia.org/article/K-nearest_neighbor)
- google-all-pairs-similarity-search (https://code.google.com/p/google-all-pairs-similarity-search/)