k-nearest neighbors algorithm

Contents hide

From Wikipedia, the free encyclopedia

This article is about supervised classification/regression, but NOT a clustering algorithm. For algorithms for finding nearest neighbors, see Nearest neighbor search.
Not to be confused with k-means clustering.

In statistics, the ***k*-nearest neighbors algorithm** (***k*-NN**) is a non-parametric supervised learning method first developed by Evelyn Fix and Joseph Hodges in 1951,[1] and later expanded by Thomas Cover.[2] It is used for classification and regression. In both cases, the input consists of the *k* closest training examples in a data set. 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 plurality 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 *k* nearest neighbors. If *k* = 1, then the output is simply assigned to the value of that single nearest neighbor.

k-NN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.[3]

Both for classification and regression, a useful technique can be to assign weights 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.[4]

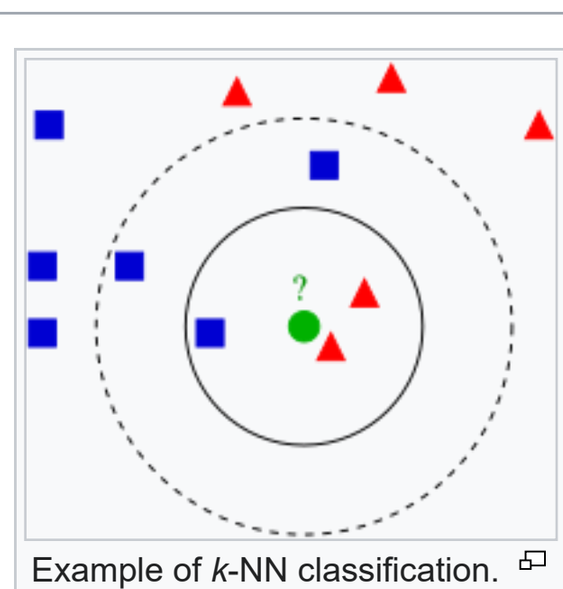
The neighbors are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A peculiarity of the *k*-NN algorithm is that it is sensitive to the local structure of the data.

Statistical setting [edit]

Suppose we have pairs $(X_1,Y_1),(X_2,Y_2),\dots,(X_n,Y_n)$ taking values in $\mathbb{R}^d \times \{1,2\}$, where Y is the class label of X, so that $X|Y=r \sim P_r$ for $r=1,2$ (and probability distributions P_r). Given some norm $\|\cdot\|$ on $\mathbb{R}^d$ and a point $x \in \mathbb{R}^d$, let $(X_{(1)},Y_{(1)}),\dots,(X_{(n)},Y_{(n)})$ be a reordering of the training data such that $\|X_{(1)}-x\| \leq \dots \leq \|X_{(n)}-x\|$.

Algorithm [edit]

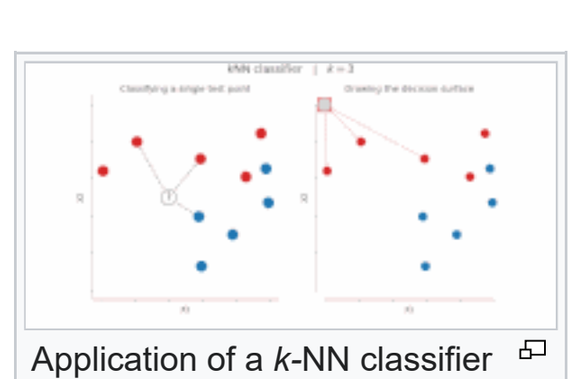


Example of *k*-NN classification. The test sample (green dot) should be classified either to blue squares or to red triangles. If *k = 3* (solid line circle) it is assigned to the red triangles because there are 2 triangles and only 1 square inside the inner circle. If *k = 5* (dashed line circle) it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

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.

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 been employed with correlation coefficients, such as Pearson and Spearman, as a metric.[5] 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.



Application of a *k*-NN classifier considering *k* = 3 neighbors. Left - Given the test point "?", the algorithm seeks the 3 closest points in the training set, and adopts the majority vote to classify it as "class red". Right - By iteratively repeating the prediction over the whole feature space (X1, X2), one can depict the "decision surface".

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.[6] One way to overcome this problem is to weight 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 [edit]

The best choice of *k* depends upon the data; generally, larger values of *k* reduces effect of the noise on the classification,[7] 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[citation needed] approach is the use of evolutionary algorithms to optimize feature scaling.[8] Another popular approach is to scale features by the mutual information of the training data with the training classes.[citation needed]

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.[9]

The 1-nearest neighbor classifier [edit]

The most intuitive nearest neighbour type classifier is the one nearest neighbour classifier that assigns a point x to the class of its closest neighbour in the feature space, that is $C_n^{1nn}(x) = Y_{(1)}$.

As the size of training data set approaches infinity, the one nearest neighbour classifier guarantees an error rate of no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data).

The weighted nearest neighbour classifier [edit]

The k-nearest neighbour classifier can be viewed as assigning the k nearest neighbours a weight $1/k$ and all others 0 weight. This can be generalised to weighted nearest neighbour classifiers. That is, where the ith nearest neighbour is assigned a weight w_{ni}, with $\sum_{i=1}^n w_{ni} = 1$. An analogous result on the strong consistency of weighted nearest neighbour classifiers also holds.[10]

Let C_n^{wnn} denote the weighted nearest classifier with weights $\{w_{ni}\}_{i=1}^n$. Subject to regularity conditions, which in asymptotic theory are conditional variables which require assumptions to differentiate among parameters with some criteria. On the class distributions the excess risk has the following asymptotic expansion[11]

$$\mathcal{R}_\mathcal{R}(C_n^{wnn}) - \mathcal{R}_\mathcal{R}(C^{\text{Bayes}}) = \left(B_1 s_n^2 + B_2 t_n^2\right)\{1+o(1)\},$$

for constants B_1 and B_2 where $s_n^2 = \sum_{i=1}^n w_{ni}^2$ and $t_n = n^{-2/d}\sum_{i=1}^n w_{ni}\left\{i^{1+2/d} - (i-1)^{1+2/d}\right\}$.

The optimal weighting scheme $\{w_{ni}^*\}_{i=1}^n$, that balances the two terms in the display above, is given as follows: set $k^* = \lfloor B n^{\frac{4}{d+4}} \rfloor$,

$$w_{ni}^* = \frac{1}{k^*}\left[1 + \frac{d}{2} - \frac{d}{2{k^*}^{2/d}}\{i^{1+2/d} - (i-1)^{1+2/d}\}\right]$$

for $i = 1,2,\dots,k^*$ and

$$w_{ni}^* = 0$$

for $i = k^*+1,\dots,n$.

With optimal weights the dominant term in the asymptotic expansion of the excess risk is $\mathcal{O}(n^{-\frac{4}{d+4}})$. Similar results are true when using a bagged nearest neighbour classifier.

Properties [edit]

k-NN is a special case of a variable-bandwidth, kernel density "balloon" estimator with a uniform kernel.[12][13]

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 approximate 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 two-class *k*-NN 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).[2] Various improvements to the *k*-NN speed are possible by using proximity graphs.[14]

For multi-class *k*-NN classification, Cover and Hart (1967) prove an upper bound error rate of

$$R^* \ \leq \ R_{k\mathrm{NN}} \ \leq \ R^* \left(2 - \frac{MR^*}{M-1}\right)$$

where R^* is the Bayes error rate (which is the minimal error rate possible), R_{kNN} is the asymptotic *k*-NN error rate, and M is the number of classes in the problem. This bound is tight in the sense that both the lower and upper bounds are achievable by some distribution.[15] For $M=2$ and as the Bayesian error rate R^* approaches zero, this limit reduces to "not more than twice the Bayesian error rate".

Error rates [edit]

There are many results on the error rate of the k nearest neighbour classifiers.[16] The k-nearest neighbour classifier is strongly (that is for any joint distribution on (X,Y)) consistent provided $k := k_n$ diverges and k_n/n converges to zero as $n \to \infty$.

Let C_n^{knn} denote the k nearest neighbour classifier based on a training set of size n. Under certain regularity conditions, the excess risk yields the following asymptotic expansion[11]

$$\mathcal{R}_\mathcal{R}(C_n^{knn}) - \mathcal{R}_\mathcal{R}(C^{\text{Bayes}}) = \left\{B_1\frac{1}{k} + B_2\left(\frac{k}{n}\right)^{4/d}\right\}\{1+o(1)\},$$

for some constants B_1 and B_2.

The choice $k^* = \left\lfloor B n^{\frac{4}{d+4}} \right\rfloor$ offers a trade off between the two terms in the above display, for which the k^*-nearest neighbour error converges to the Bayes error at the optimal (minimax) rate $\mathcal{O}\left(n^{-\frac{4}{d+4}}\right)$.

Metric learning [edit]

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 [edit]

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 [edit]

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.[17]

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. This process is also called low-dimensional embedding.[18]

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",[19] "sketches"[20] or other high-dimensional similarity search techniques from the VLDB toolbox might be the only feasible option.

Decision boundary [edit]

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.[21]

Data reduction [edit]

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 [edit]

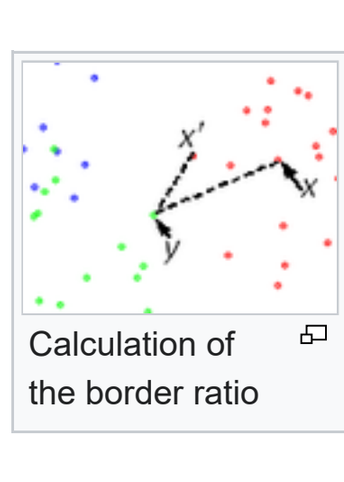
A training example surrounded by examples of other classes is called a class outlier. Causes of class outliers include:

- 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 don't 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.

Condensed Nearest Neighbor for data reduction [edit]

Condensed nearest neighbor (CNN, the *Hart algorithm*) is an algorithm designed to reduce the data set for *k*-NN classification.[22] 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.



Calculation of the border ratio

-Prototype
-Outlier
-Absorbed

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

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.[23] 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*.

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*.

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.[23]

CNN model reduction for k-NN classifiers

Fig. 1. The dataset.

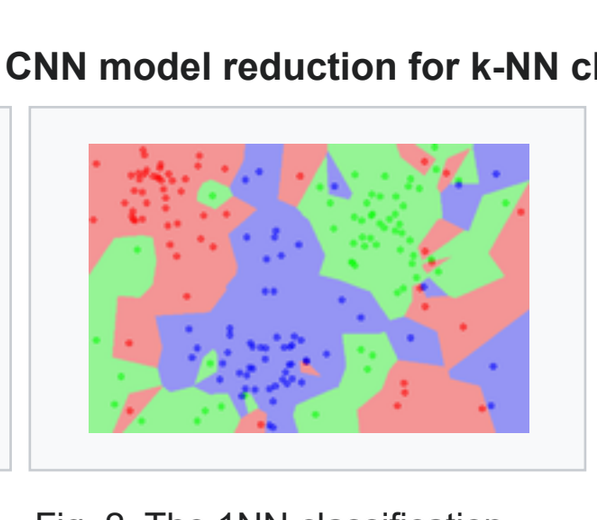


Fig. 2. The 1NN classification map.

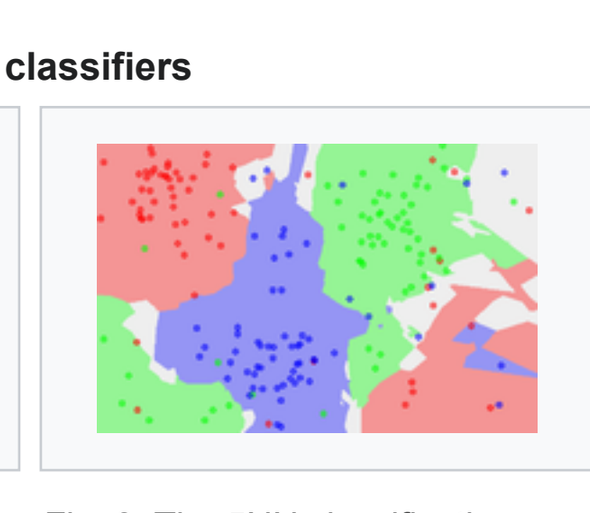


Fig. 3. The 5NN classification map.

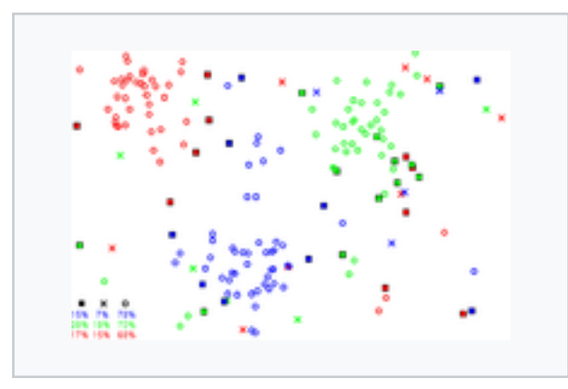


Fig. 4. The CNN reduced dataset.

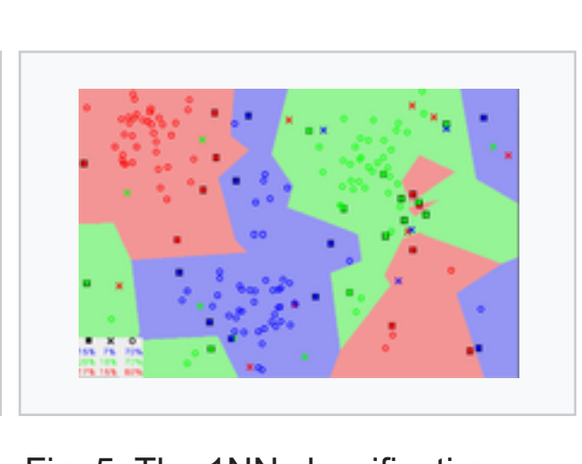


Fig. 5. The 1NN classification map based on the CNN extracted prototypes.

k-NN regression [edit]

In *k*-NN regression, the *k*-NN algorithm[citation needed] 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.

k-NN outlier [edit]

The distance to the *k*th nearest neighbor can also be seen as a local density estimate and thus is also a popular outlier score in anomaly detection. The larger the distance to the *k*-NN, the lower the local density, the more likely the query point is an outlier.[24] Although quite simple, this outlier model, along with another classic data mining method, local outlier factor, works quite well also in comparison to more recent and more complex approaches, according to a large scale experimental analysis.[25]

Validation of results [edit]

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.[how?]

See also [edit]

- Nearest centroid classifier
- Closest pair of points problem
- Nearest neighbor graph
- Segmentation-based object categorization

Mathematics portal

References [edit]

1. ^ Fix, Evelyn; Hodges, Joseph L. (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties (PDF) (Report). USAF School of Aviation Medicine, Randolph Field, Texas. Archived (PDF) from the original on September 26, 2020.
2. ^ *a* *b* Cover, Thomas M.; Hart, Peter E. (1967). "Nearest neighbor pattern classification" (PDF). *IEEE Transactions on Information Theory*. **13** (1): 21–27. CiteSeerX 10.1.1.68.2616. doi:10.1109/TIT.1967.1053964. S2CID 5246200.
3. ^ Hastie, Trevor. (2001). *The elements of statistical learning : data mining, inference, and prediction : with 200 full-color illustrations*. Tibshirani, Robert., Friedman, J. H. (Jerome H.). New York: Springer. ISBN 0-387-95284-5. OCLC 46809224.
4. ^ This scheme is a generalization of linear interpolation.
5. ^ Jaskowiak, Pablo A.; Campello, Ricardo J. G. B. (2011). "Comparing Correlation Coefficients as Dissimilarity Measures for Cancer Classification in Gene Expression Data". *Brazilian Symposium on Bioinformatics (BSB 2011)*: 1–8. CiteSeerX 10.1.1.208.993.
6. ^ Coomans, Danny; Massart, Desire L. (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.
7. ^ Everitt, Brian S.; Landau, Sabine; Leese, Morven; and Stahl, Daniel (2011) "Miscellaneous Clustering Methods", in *Cluster Analysis*, 5th Edition, John Wiley & Sons, Ltd., Chichester, UK
8. ^ Nigsch, Florian; Bender, Andreas; van Buuren, Bernd; Tissen, Jos; Nigsch, Eduard; Mitchell, John B. O. (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.
9. ^ Hall, Peter; Park, Byeong U.; Samworth, Richard J. (2008). "Choice of neighbor order in nearest-neighbor classification". *Annals of Statistics*. **36** (5): 2135–2152. arXiv:0810.5276. Bibcode:2008arXiv0810.5276H. doi:10.1214/07-AOS537. S2CID 14059866.
10. ^ Stone, Charles J. (1977). "Consistent nonparametric regression". *Annals of Statistics*. **5** (4): 595–620. doi:10.1214/aos/1176343886.
11. ^ *a* *b* Samworth, Richard J. (2012). "Optimal weighted nearest neighbour classifiers". *Annals of Statistics*. **40** (5): 2733–2763. arXiv:1101.5783. doi:10.1214/12-AOS1049. S2CID 88511688.
12. ^ Terrell, George R.; Scott, David W. (1992). "Variable kernel density estimation". *Annals of Statistics*. **20** (3): 1236–1265. doi:10.1214/aos/1176348768.
13. ^ Mills, Peter (2012-08-09). "Efficient statistical classification of satellite measurements". *International Journal of Remote Sensing*.
14. ^ Toussaint, Godfried T. (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.
15. ^ Devroye, L., Gyorfi, L. & Lugosi, G. A Probabilistic Theory of Pattern Recognition. Discrete Appl Math 73, 192–194 (1997).
16. ^ Devroye, Luc; Gyorfi, Laszlo; Lugosi, Gabor (1996). *A probabilistic theory of pattern recognition*. Springer. ISBN 978-0-3879-4618-4.
17. ^ Beyer, Kevin; et al. "When is "nearest neighbor" meaningful?" (PDF). *Database Theory—ICDT'99*. **1999**: 217–235.
18. ^ Shaw, Blake; Jebara, Tony (2009), "Structure preserving embedding" (PDF), *Proceedings of the 26th Annual International Conference on Machine Learning* (published June 2009), pp. 1–8, doi:10.1145/1553374.1553494, ISBN 9781605585161, S2CID 8522279
19. ^ Bingham, Ella; Mannila, Heikki (2001). "Random projection in dimensionality reduction". *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*. pp. 245–250. doi:10.1145/502512.502546. ISBN 158113391X. S2CID 1854295.
20. ^ Ryan, Donna (editor); *High Performance Discovery in Time Series*, Berlin: Springer, 2004, ISBN 0-387-00857-8
21. ^ Bremner, David; Demaine, Erik; Erickson, Jeff; Iacono, John; Langerman, Stefan; Morin, Pat; Toussaint, Godfried T. (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.
22. ^ Hart, Peter E. (1968). "The Condensed Nearest Neighbor Rule". *IEEE Transactions on Information Theory*. **18**: 515–516. doi:10.1109/TIT.1968.1054155.
23. ^ *a* *b* Mirkes, Evgeny M.; *KNN and Potential Energy: applet*, University of Leicester, 2011
24. ^ Ramaswamy, Sridhar; Rastogi, Rajeev; Shim, Kyuseok (2000). "Efficient algorithms for mining outliers from large data sets". *Proceedings of the 2000 ACM SIGMOD international conference on Management of data - SIGMOD '00*. Proceedings of the 2000 ACM SIGMOD international conference on Management of data – SIGMOD '00. pp. 427–438. doi:10.1145/342009.335437. ISBN 1-58113-217-4.
25. ^ Campos, Guilherme O.; Zimek, Arthur; Sander, Jörg; Campello, Ricardo J. G. B.; Micenková, Barbora; Schubert, Erich; Assent, Ira; Houle, Michael E. (2016). "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study". *Data Mining and Knowledge Discovery*. **30** (4): 891–927. doi:10.1007/s10618-015-0444-8. ISSN 1384-5810. S2CID 1952214.

Further reading [edit]

- Dasarathy, Belur V., ed. (1991). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. ISBN 978-0818689307.
- Shakhnarovich, Gregory; Darrell, Trevor; Indyk, Piotr, eds. (2005). *Nearest-Neighbor Methods in Learning and Vision*. MIT Press. ISBN 978-0262195478.