k-NN Classifiers and Irrelevant Attributes

Manish Devana[1] and Prathmesh Matodkar [2]

[1]Rosenstiel School of Marine and Atmospheric Science
[2]University of Miami

September 24, 2018

Introduction

k-NN, also known as K nearest neighbor, is a classification algorithm used for assigning a class label to a particular example based on it's proximity to the trained classified example datasets(Kubat, 2015). The logic, understanding, and representation of this algorithm is fairly simple and is therefore an attractive and simple choice for classification problems. Here we investigate the effect of irrelevant attributes on the k-NN classification performance. Irrelevant attributes are ones which are not realistically related to the job or the activity in question. It should be noted that this is separate from redundant attributes, which provided relevant information in multiple ways rather than completely irrelevant information. Since k-NN algorithms are based on distances, adding more attributes directly affects distance calculations. Therefore, it is hypothesized that irrelevant attributes will significantly reduce the performance of k-NN algorithms.

k-NN Classifier Construction

For the experiments shown here, we constructed a simple k-NN classifier without weighting or scaling features. The model is designed to work with numeric attributes and classify boolean classes (i.e. *True* or *False*). We implement Euclidian distances to calculate the distance between nearest neighbors. For a vector $\vec{X} = [x_1, x_2, ...x_m]$, with m attributes, the distance between $\vec{X}$and a training example vector $\vec{E} = [e_1,e_2, ...e_m]$ is defined as :

$$d_i = \sqrt{(x_i - e_1)^2 + (x_i - e_2)^2 + ... + (x_i - e_m^2)} \tag{1}$$

In the most simple case of a single nearest neighbor (1-NN) the input vector is classified with the class of the example vector with the minimum distance. The addition of additional nearest neighbors introduces a "voting system" such that the most frequently occurring class among the k nearest neighbors found is the class assigned to the (201, 2017)input . Since the experiments presented here deal with strictly two class scenarios, k must always be an odd number to avoid a tie between classes. An overview of the k-NN voting is shown in table. 1 , where the three examples shown are the k nearest neighbors (k=3). Examples 1 and 2 are positive (class =1) and example 2 is negative, leading to a voting result of 2-1 in favor of positive The mathematical form of this logic is seen in Eq. 2 where c^1 and c^0 represent the positive and negative classes, respectively, and there

are m positive and j negative classes. Since the sum of positive classes is greater than the sum of negative classes, the result is a positive classification.

Table 1: k-NN voting process

example	distance	class
e_1	4	1
e_2	2	1
e_3	3	0

$$\sum_{i=1}^{m} c_i^1 > \sum_{i=1}^{j} c_i^0 \tag{2}$$

Example dataset construction

Example Dataset 1

In order to illustrate the effect of irrelevant attributes on the k-NN classifier, two sample datasets are constructed, both with 2 classes but varied numeric attribute ranges. The first dataset (Fig. 1a) is comprised of 1000 examples with 2 attributes and positive or negative class separated by a linear function f(x1). Attribute 1 (x_1) is a randomly generated number on the interval [0,1]. Attribute 2 is generated by randomly adding or subtracting a random number the interval [0,3] to each f(x), where final values greater than f(x) are deemed positive and those less than f(x) are deemed negative. It should be noted that the actual line function (in the form y=mx+b) was also generated randomly, meaning a randomly generated intercept and slop value. It was found that the random variations of the line function had no effect on the outcome of the results here.

Dataset 2 - Mock Test Scores

A second test dataset (Fig. 1b) was generated to simulate a more real-world dataset, where the average of 2 test scores ranging from 0-30, were used to determine whether a student passed or failed a class (each test score is an attribute and the average is only used to determine the class). Then a wide range of numeric irrelevant attributes is added, with values on the interval [0, 50]. While we do not explicitly name the irrelevant attributes, it is useful to describe a plausible example. In the case of classifying a pass or fail for a student, an irrelevant attribute might be a student's shoe size or the temperature in the north pole. While such examples are so absurd that they are unlikely to be found in datasets looking at test scores, it serves as a clear thought exercise for defining what is truly an irrelevant attribute. The larger range of values in the second example dataset also allows for examining the effects of numeric scaling in irrelevant attributes. In reality, irrelevant attributes are likely to be different scales than the relevant ones.

	X1 (relevant)	X2 (relevant)	X3	X4	X5
E1	18.94	12.21	8.4	23.57	4.34
E2	10.65	6.53	2.22	4.33	3.98
E3	29.91	19.8	3.11	0.1	4.03
E4	6.73	12.2	10.89	17.67	9.83
E5	19.57	5.2	18.3	5.57	2.51

Table 2: First 5 examples and first 5 attributes of test score dataset, where attributes 1 and 2 (X1 &X2) are the relevant attributes

Experiment Setup

Both example datasets were used in an identical experimental procedure, where each dataset was split 70/30 into training/testing subgroups. Classification is then conducted using the initial 2 (relevant) attributes, and repeated for the range of k values [1,50] (odds only). After each k value is utilized, a single irrelevant attribute is added and the classifications are repeated. In other words, an additional column of Table 2 is utilized in each subsequent classification, where the columns in the table represent an attribute. This procedure is then repeated until all of the randomly generated attributes are used in classification. It is important to note that the 70/30 split of datasets was maintained through all the tests. This prevents the examples from training and testing datasets being reshuffled, which would alter the results of the tests implemented here.

Results

Dataset 1

As an initial test of the classifier, both example datasets are classified without the addition of any irrelevant attributes. Subsequent classifications are done, adding an additional random value irrelevant attribute each time. For example, dataset 1, where attributes all fall within a similar domain, a sharp increase in error rate is seen as irrelevant attributes are added. After the 10th irrelevant attribute is added, the error rate increases at a slower rate until saturating around 30% (Fig. 2). This trend is clearly unaffected by the number of neighbors used in the classification, shown by analogous trends in the grey shaded region of the plot which represents varying numbers of nearest neighbors.

Test Score Dataset

The experiment described above was then conducted on the Test Score dataset, again adding a new irrelevant attribute after each experiment until 50 irrelevant attributes were present. In this case, the rise in error rate was far more pronounced and sudden, reaching 35% errors after only several irrelevant attributes were introduced (Fig.3) Beyond this point, the increase in error saturated at a similar level to that of the previous example, around 30%. Here also the number of nearest neighbors utilized did not affect the overall trend. Additionally, while, the range of numeric values in the irrelevant attributes was larger, scaling also does not seem to have significantly altered the trend. It is interesting to note that the error rate reaches a similar constant, implying some underlying property of the euclidian distance formula is affecting the trends seen here.

Final words

Through the experiments shown here, we conclude that the addition of irrelevant attributes negatively impacts k-NN classifiers, regardless of the number of nearest neighbors used. While these results are for numeric attribute cases, it is unlikely to differ greatly in other cases. This is because the underlying use of Euclidean distances will be affected whenever you add a new term to the distance calculation. And in both cases, after an initial rise in error rate, there is an eventual saturation around 30% error. It would require further experiments to examine if this 30% saturation level is similar for a large range of datasets. These experiments highlight the importance of using some amount of intuition when applying a k-NN classifier, which would allow you to remove irrelevant attributes before the actual classifier implementation. Additionally, utilization of alternative distance methodologies as well as weighted nearest neighbors may help to reduce the error of irrelevant attributes.

References

Multiple Classifier Systems. In *Encyclopedia of Machine Learning and Data Mining*, pages 882–882. Springer US, 2017. doi: 10.1007/978-1-4899-7687-1_100318. URL https://doi.org/10.1007%2F978-1-4899-7687-1_100318.

Miroslav Kubat. Similarities: Nearest-Neighbor Classifiers. In *An Introduction to Machine Learning*, pages 43–64. Springer International Publishing, 2015. doi: 10.1007/978-3-319-20010-1_3. URL https://doi.org/10.1007%2F978-3-319-20010-1_3.

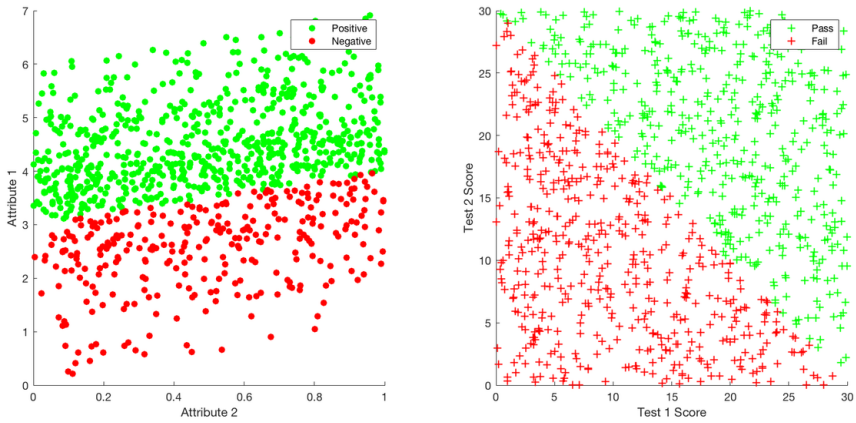

Figure 1: (**a -** *left*) Random Data sample Separated into 2 classes based on the equation f(x). (**b** -*right*) Dataset containing two test scores and the class separation based on Average test score

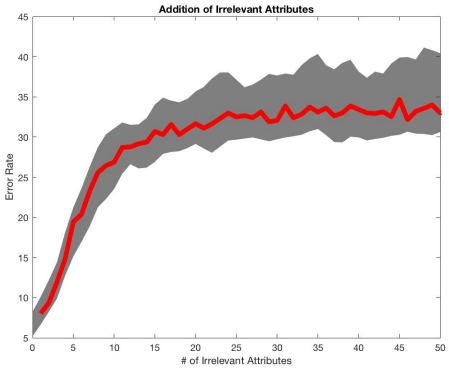

Figure 2: Error rate due to addition of Irrelevant Attributes for the above dataset

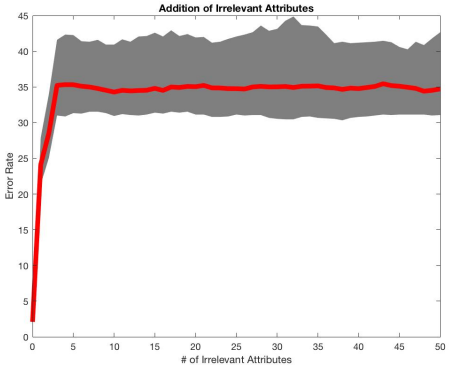

Figure 3: Error rate due to addition of Irrelevant Attributes for the Test score Dataset