Lloyd Beaufils, Jerrry Bonnell, Gururaj Shriram

ECE-648

18 October 2017

Project 1 - Group 5

The Effect of Irrelevant Attributes on k-NN Classifiers

The k-NN classifier is a version of the nearest neighbor classifier that looks at k neighbors as opposed to simply one. For any given example, the classifier finds its k nearest neighbors and chooses a class label based upon those of those neighbors. Since this classifier makes use of geometric distances to other examples when deciding class labels, it is essential that those examples be relevant. Each attribute contributes equally to the distance, so attributes that should have no bearing on the class label can quickly overpower the relevant attributes and lead to improper classification. With the irrelevant attributes adding essentially random values to and dominating the distance, the classifier degenerates until it does little better than label classes at random. This behavior is dependent upon the ratio of relevant attributes to irrelevant attributes. The higher that ratio, the less of a negative impact the irrelevant attributes will have. The more irrelevant attributes, then the more terms negatively impacting the distance formula and the worse the classifier will perform, on average. Our group has run some tests to obtain evidence of these claims, and in this paper, we will outline our findings.

For this project, we used two separate k-NN classifiers to verify two different datasets. For the k-NN classifiers, we used a built-in Python algorithm known as Scikit and compared our results with an algorithm we wrote based upon the pseudocode in the book. The two algorithms have mostly the same output, but there are very slight differences. For the k=1 case, the two algorithms are actually identical- they behave exactly the same way in classifying the data. The differences arise when k is greater than one. We determine this difference to be in how we break ties that arise from having more than two possible classes. In our implementation, ties are essentially broken by the Python sort method. When determining classes, we sort classes based by number of neighbors indicating that class. If two happen to tie, then Python sorts alphanumerically by the name of the class. This was a bit of an oversight on our part, but since these domains are both so simple and ties are infrequent, it didn’t end up affecting the data in any real sense. To add irrelevant attributes, we used the normal distribution of the valid data to determine a range within which to generate random values. This resulted in the irrelevant attributes being random without dominating the data due to improper scaling. Finally, we randomly partitioned the datasets into testing and training sets with a ratio of 3:7, respectively. We employed random subsampling to make certain that our training set was representative of the data, and this ensured that our results were reliable. We ran these two algorithms on two separate datasets- the Iris dataset from the UCI repository and the Animals dataset that we fabricated. The results of our experiments are located below.

We obtained the Iris dataset from the UCI machine learning repository. The attributes are sepal and petal length and width, and the examples are classified into one of three species of iris. This domain is well-known for being simple, and that simplicity aids us in demonstrating our findings. With no irrelevant attributes, the k-NN classifiers (with k ranging from 1 to 10) all classify randomly-partitioned testing sets with at least 95% accuracy. Our data clearly show that adding irrelevant attributes immediately worsens our accuracy rate, plummeting about twenty points when two irrelevant attributes are added to the dataset. This downward trend continues all the way until the accuracy rate is around an abysmal 30% with 18 irrelevant attributes. This is the case for both k-NN classifiers, with varied values for k. The error rate is above 66%, indicating that having irrelevant attributes makes the classifier do worse than random; the bad attributes actively make worse decisions than selecting a class completely at random. The sharpest drop in accuracy occurs with the addition of the first 6 irrelevant attributes. This makes sense because the amount of bad values eclipse and overcome the amount of good values in the distance formula (3 relevant attributes), so at that point it reaches and dips slightly below 40%, which is slightly better than pure randomness because the relevant attributes contribute slightly to picking the correct class.

The Animals dataset yielded similar results to the Iris dataset. We generated this data programmatically, specifying ranges for valid attributes and generating random numbers within those ranges. There are 5 classes representing different animals, with both relevant (correlated) and irrelevant attributes. The k-NN classifiers do a slightly poorer job with no irrelevant attributes when compared to those classifying the Iris dataset. With only relevant attributes, both classifiers average about 91% accuracy across varying values for k. The higher the k value, the more accurate the classifier is. Again, as with the Iris dataset, the classifiers decrease in accuracy with the addition of irrelevant attributes. However, for this dataset, the decline is steadier and more gradual. The accuracy rate ends up at around 25%, which is slightly lower than the performance on the Iris dataset. However, this makes sense since there are more classes (5 vs 3), so when the classification approaches random selection, there is a lower chance that any specific class is selected.

To verify that the decrease in accuracy is a result of adding irrelevant attributes and not a result of simply adding attributes in general, we performed an additional test on the Wine dataset from the UCI repository. This dataset contains 13 attributes. Our test was as follows. First, we selected only 2 of the 13 attributes and ran our k-NN classifier on it. Then, we repeated this process, adding two of the valid attributes at a time until we were using 12 of them. This stands in contrast to our previous tests, as no irrelevant attributes were introduced. What we found was as expected- the accuracy did not decrease as a result of having many attributes. Besides a dip at the ‘6 attribute’ mark (it would appear that one attribute was less relevant than initially believed, or perhaps improperly scaled), the accuracy was either the same or better as a result of having more relevant attributes. This just reinforces the fact that it is the irrelevant attributes that are responsible for the increased error rate in the k-NN classifiers.

After all our experiments, we conclude that irrelevant attributes very clearly increase the error rate of nearest neighbor classifiers. The intuition is straightforward, and our experimental results back it up. The irrelevant attributes add terms to the distance formula that end up obfuscating the result and causing classifiers to deteriorate to randomness. We tested two classifiers on two datasets and found the same results throughout every test. Furthermore, we tested adding only relevant attributes to see if it would have the same effect and it did not. The cause of the decrease in accuracy is the irrelevant attributes, and these tests show just how dangerous they can be. To reduce a classifier from near-perfect accuracy to random performance is a very serious matter, and great care must be taken to ensure that no irrelevant attributes are used when making any classifications with machine learning.

Figure 1: This graph shows our 1-NN and 7-NN classifiers on the Iris dataset. Except for the bump at 12 irrelevant attributes, the irrelevant attributes have more of a negative effect when k has a higher value.

Figure 2: This shows our classifier with all values of k we tested on the Iris dataset. For all classifiers, the steepest drop in accuracy is with the introduction of the first 6 irrelevant attributes, after which our classifier converges at around 30% accuracy.

Figure 3: Similarly to before, irrelevant attributes very negatively impact the accuracy rate of our classifier. However, in this dataset, the irrelevant attributes have more of a negative effect on the 1-NN classifier than the 7-NN classifier.

Figure 4: Irrelevant attributes again are shown to decrease the accuracy rate for all values of k in our classifier. Here, the descent is a bit more gradual, but once more the accuracy ends up at around 30%.

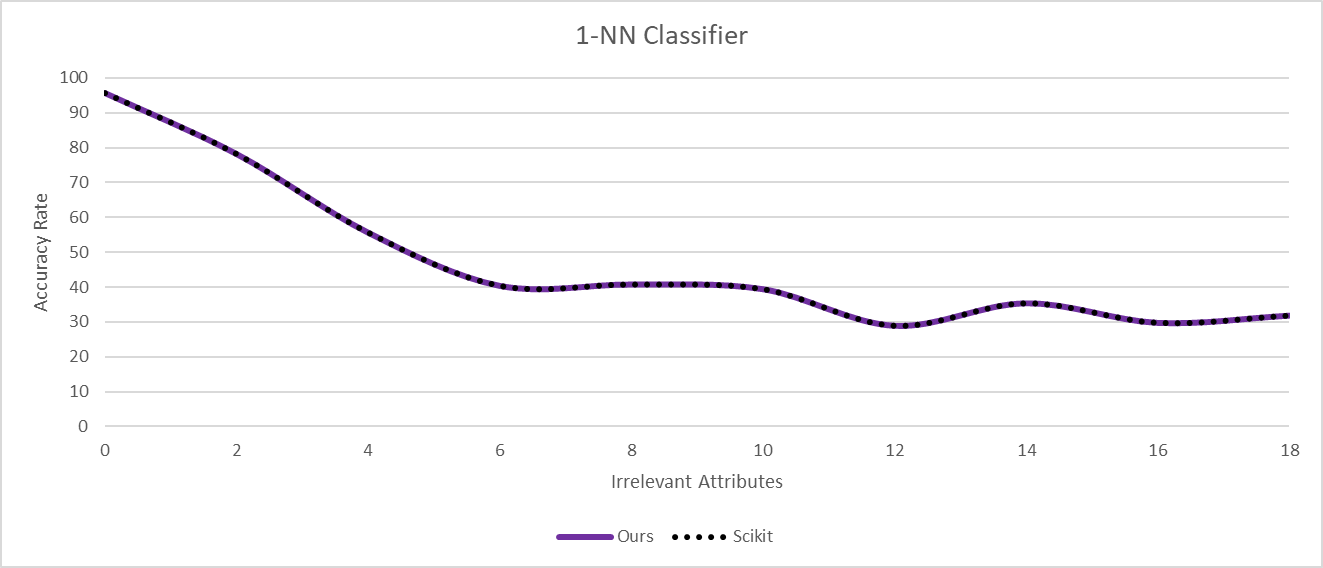


Figure 5: These two figures show the performance of our classifier against Python’s built-in k-NN classifier, Scikit. They behave identically for k=1 but begin to vary at k-values higher than one. We believe this to be due to a difference in how class ties are broken.

Figure 6: This graph compares the performance of our 1-NN classifier on both datasets. It demonstrates how the irrelevant attributes cause a sharper decline in the Iris dataset and how they decrease the classifier’s accuracy on both datasets.

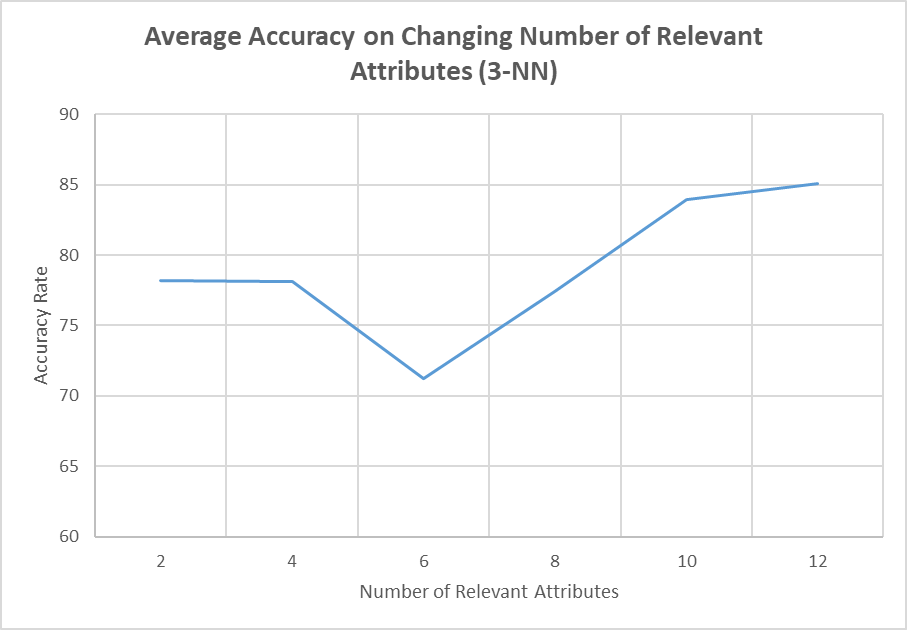


Figure 7: This graph shows our 3-NN on the Wine dataset. It demonstrates that the lower accuracy demonstrated in all other graphs is not merely a result of additional attributes, but of specifically irrelevant attributes. Here we add relevant attributes two at a time and the accuracy is improved.

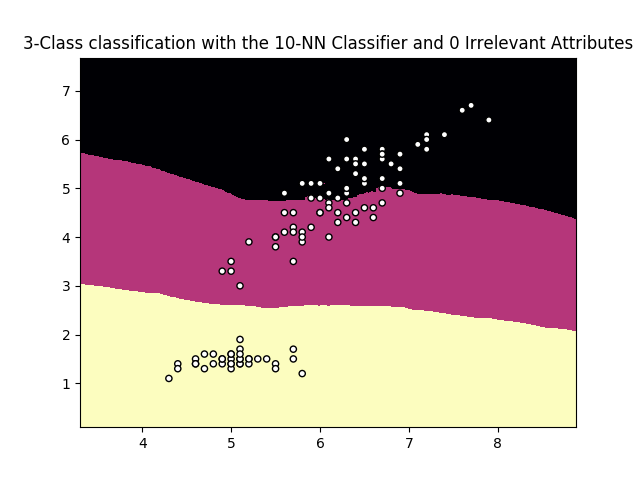


Figure 8: This colormap demonstrates where an example would have to land to be classified in the Iris dataset with the 10-NN classifier and 0 irrelevant attributes. The three colors represent the three classes, and this classifier performs with about 96% accuracy.

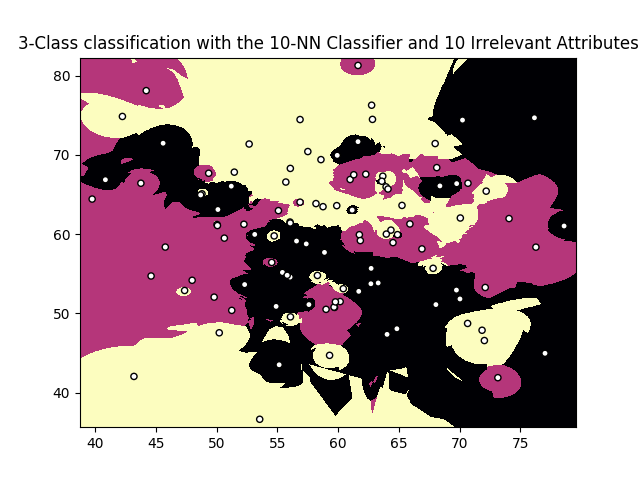


Figure 9: This colormap shows the effect of 10 irrelevant attributes on the classifier. The class boundaries are thrown into complete disarray,

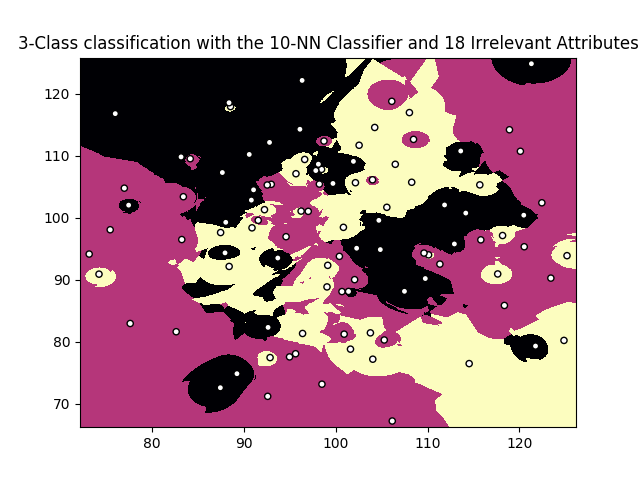


Figure 10: