Methods to Pre-Process Training Data for K-Nearest Neighbors Algorithm

John Cartmell, InterDigital Communications LLC

January 22, 2010

1 Abstract

The basic K-nearest neighbor classification algorithm performs a search through the training samples, computing the distance for each training sample from the sample to be classified. Once the distances, are computed the class of the majority of the k closest points is assigned as the classification of the test sample. The training phase of the algorithm is extremely efficient as no pre-processing of the training data is required. However, the phase where test samples are classified is very dear, since for every sample to be classified the entire training class must be traversed. In this paper, we explore three methods that will reduce the number of training samples to be traversed during the classification process. Each method reduces the number of samples in each class by averaging the training samples in each class using different techniques. Therefore, instead of having to compare a test sample against all of the training samples, a test sample is only compared against the reduced set of training samples. Once these methods are described, they are used, along with other classification algorithms, on real data sets to demonstrate their effectiveness from both fidelity and performance standpoints.

2 Keywords

KNN, K-Nearest Neighbors, classification, optimization methods

3 Introduction

The KNN Algorithm is a method used for classification that is unique, as the development of the model (training phase) requires almost no effort. However, the model requires significant real-time to classify a sample (classification phase). When a sample is to be classified, it is compared to all other samples within the training set, usually by computing the Euclidean distance from each sample. In cases where the samples are discrete valued, the Hamming distance can also be used. The sample to be classified is assigned a classification that is the majority of the nearest k-samples from the training set. If only one nearest neighbor is used, the classification of the new sample is the class of the single nearest neighbor. If more than one nearest neighbor is used, the class of the majority of these neighbors is the assigned class of the sample. Should there be a tie, it is implementation-dependent which classification is selected. The number of nearest neighbors to use has typically been selected heuristically and the determination of the optimal number of neighbors is not discussed further in this paper. As is shown in the following sections, the KNN algorithm takes a large amount of time to perform classification when compared to other classification methods. The time required to perform the classification phase is dependent on the size and complexity (dimensionality) of the training set.

Ideally, the algorithm can be modified such that the time required for the classification phase can be moved or shared with the training phase. Presumably, there is an "infinite" amount of time to create the model, but time is dear when the actual classifications are performed. Instead of having to compare each sample to be classified against every sample in the training set, this paper proposes three methods to reduce the

training set size, thereby reducing the real-time requirements of the classification phase. In addition, the methods proposed in this paper are compared with currently existing techniques. After describing the three pre-processing methods, the performance of each is demonstrated and analyzed against several data sets using the R Programming Language (which is similar to MATLAB) [1] running Windows XP on a Lenovo ThinkPad T500 with 4GB of RAM.

4 Background

There are several published variations of the KNN algorithm that are reviewed in this section. The applicability of each variant to solve the problem that we address in this paper is noted. Some methods attempt to make the distance computations of a test sample against all the training samples as fast as possible by employing parallel computing techniques [2]. While these methods are certainly valid, they are outside the scope of this paper and, therefore, not further explored.

Another variant of the KNN algorithm involves the weighting of the samples that are used in the classification process as a function of their distance from the sample to be classified [3]. Usually, in KNN, the classification assigned to the test sample is the simple majority of the k nearest neighbors. In a weighted version of KNN, the "vote" of the k nearest neighbors is weighted as a function of each nearest neighbor's distance from the sample to be classified. If a training sample is close to the sample to be classified, it is given a large weight, while if the training sample is far from the sample to be classified, it is given a low voting weight. No found variant using weighting would improve the time required to classify test samples.

Alternatively, instead of weighting neighbors based on distance, attributes can be weighted based on their relevance. As the number of attributes increase, the time required to perform the distance calculation increases. Therefore, pruning unused attributes can reduce the time required to perform the classification. In [4], several variations are shown where attributes that have little discriminatory value in the classification of test samples are removed or discounted.

In this vain, if Principal Component Analysis (PCA) is used to pre-process the data, perhaps the time required to perform classification will be reduced without reducing the fidelity of the developed model. PCA is intended to allow for the extraction of relevant information from a data set that may not be well understood or is very complicated. In performing PCA, the attributes of a given set of data are transformed into a new set, where the first component accounts for as much of the variability of the attribute data as possible [5]. Each successive principal component accounts for as much of the variability in the data as possible that remains after accounting for the preceding components. PCA employs a coordinate transformation to perform this "packing" of relevant information into the most significant components. PCA may improve the timing performance of KNN by removing dimensions from the data. The benefit of performing PCA prior to performing classification of data sets will be explored to determine if it is better than the pro-processing algorithms promulgated in the next section.

Discriminant Adaptive Nearest Neighbor (DANN) is another variation of KNN where discriminant analysis is performed between the classes of an attribute as KNN is bring applied. A discriminant is applied to segregate the classes within an attribute and is used during the classification process. When a sample is to be classified, the discriminant is used to shrink or expand the neighborhood in each attribute used when finding the nearest neighbors [6]. The intent is to prevent using nearest neighbors which may be of a wrong class, and encourage the use of neighbors which are of the proper class. This can be useful when the sample to be classified is between classes, or near the edge of several classes, and the classes in the training set are close together. Performing discriminant analysis in conjunction with a nearest neighbor classification will most likely not result in an improved performance from a time required to classify standpoint. It will, however, increase the fidelity of the model but that is not the goal of this paper.

Kd-trees are data structures used to arrange the training samples in a binary tree fashion that allows for a faster search when samples require classification [7] [8] [9]. In this method, a tree is constructed which contains all the samples in the training set, placed into the tree as a function of the relationship between the values of each attribute. At each level, a different attribute is used and the median found. The median is placed at the node and all points less than the median are pushed to one side of the tree, and all points greater than the median are pushed to the other side of the tree. In other words, each level in the tree splits the current plane. When classification is performed, the sample to be classified is compared against the root node of the tree. The distance is computed and stored. A comparison is also done between the sample to be classified, and the current node on the attribute used to position this node. This comparison will steer the search for the minimum down one side or the other of the tree. This continues until the end of the tree is reached or a minimum is found. This technique prevents the distance from being computed to every sample in the training set. This method does significantly reduce the number of comparisons that are performed, and poses significant risk to nullifying whatever benefit I can achieve through my ideas. However, it appears that kd-trees are only effective when the dimensionality of the data is small. As the number of dimensions of the data increases, the benefit of the kd-tree is obviated [10] [11].

From the research performed in the preparation of this paper, only PCA and kd-trees address the fundamental problem with all KNN-based algorithms, specifically, the amount of time required to perform classification. However, when introduced into the process of classifying data sets, they may or may not reduce the time required to perform classification. If they do reduce the time to perform classification, it is at the expense of accuracy. A method is required to perform classification while not being prohibitively expensive from a time required to classify standpoint.

5 Improved Performance Algorithms

When thinking about how to reduce the classification time at the expense of as little as possible classification accuracy, there is one characteristic that was evident. The smaller the training set, the less time it takes to perform classification using KNN. The solution to reduce the time to perform classification is "obvious". It is necessary to reduce the size of the training set as much as possible, with sacrificing as little model fidelity as possible. However, samples can not just be thrown away, as each training sample has information which is pertinent. The task is to find a method to extract as much information as possible from each training sample, while reducing the entire training set size. Three methods for performing this reduction are proposed below. Of these, one shows promise when applied to real world classification of data. While each method is described below, the performance of each method is explored later in the paper.

The first proposed solution requires the creation of a training set for classification that has one super sample per class, and this method is Average KNN (AKNN). This super sample is the average of all the training set samples of that particular class. An example is shown in Figure 1 for a 2-dimensional set of data that is comprised of two classes. The raw sample points are shown, along with the super sample created for each class. However, it is not expected that this algorithm would provide good performance unless, for each attribute, there is a clear delineation between each class. Should the data be well-segregated, other classification means can be employed. While this algorithm may not be a good candidate in all circumstances, it is a step along the way towards developing an acceptable algorithm and will be included in the analysis phase. Depending on the data to be classified, this method may be sufficient.

The second proposed solution is similar to the AKNN algorithm, but is not as drastic as distilling down all the samples in one class to one super sample, and is named Partial Average KNN (PAKNN). Similar to the AKNN, the training set is reduced by averaging, however, instead of one super sample per class, each n samples are averaged to create several super samples. The result of this processing is that the training samples have been reduced by n. There is no consideration given to the location of the data in the attribute space, i.e., the averaging process is applied to n samples, without regard for where any of those samples lie in the data space. They may be near each other or may not be near each other; the only requirement is that the n samples are from the same class. Similar to AKNN, PAKNN may yield acceptable results in certain

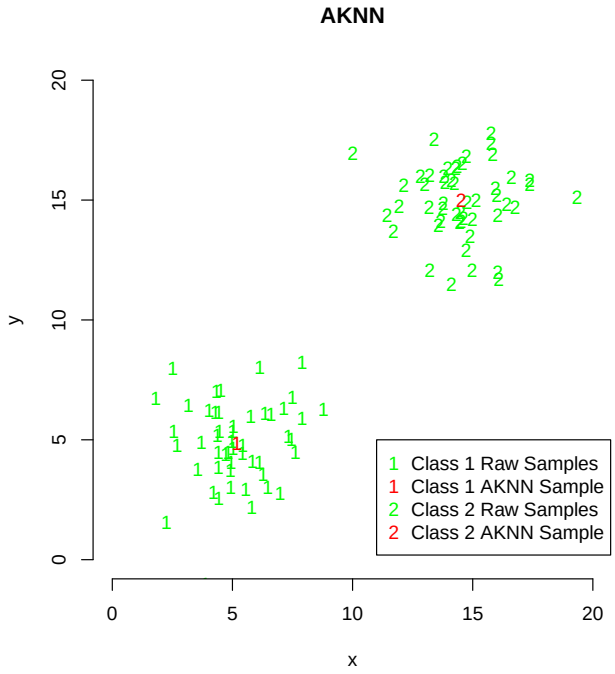

Figure 1: AKKN averaging process

circumstances. An example is shown in Figure 2 for a 2-dimensional set of data that is comprised of two classes. The raw sample points are shown along with the super samples created for each class performed by performing the partial averaging. For the purposes of this paper, the training set is segregated into *4* sub-groups, each containing *1/4* of the training samples. Each sub-group is averaged to create to a super sample.

The third proposed solution is similar to the PAKNN algorithm but is smarter about which samples to average, and is named adroitly Smart Average KNN (SAKNN). Similar to PAKNN, each n samples of a particular class are averaged to create a set of averaged super samples. However, where SAKNN differs from PAKNN is in the choice of which samples to average. PAKNN selects n consecutive samples in the training set to average, while SAKNN picks one sample in the training set and then finds the nearest *n-1* samples from the same class that have not already been used in this averaging process. These n samples are then averaged and marked as used. This averaging is continued until all samples in all classes have been used to create a set of super training samples. An example is shown in Figure 3 for a 2-dimensional set of data that is comprised of two classes. The raw sample points are shown, along with the super samples created for each class performed by smart averaging those samples near each other. This technique requires a long time to setup the data used to perform classification, as the distance of every sample from each other sample needs to be computed. However, the classification of the test data is done more quickly given the reduced training samples. Also, because similar training data was combined, the performance of the algorithm is still very reasonable in most circumstances as demonstrated below. For SAKNN, the selection of n, the number of samples to average, greatly affects the accuracy of the model and the time required to perform classification. In the following sections, the selection of n is allowed to vary and the model performance and time required to perform classification is shown. In future applications of the this model, the selection of n allows the user of this algorithm a degree of freedom in choosing whether they are more concerned with accuracy or the time to perform classification.

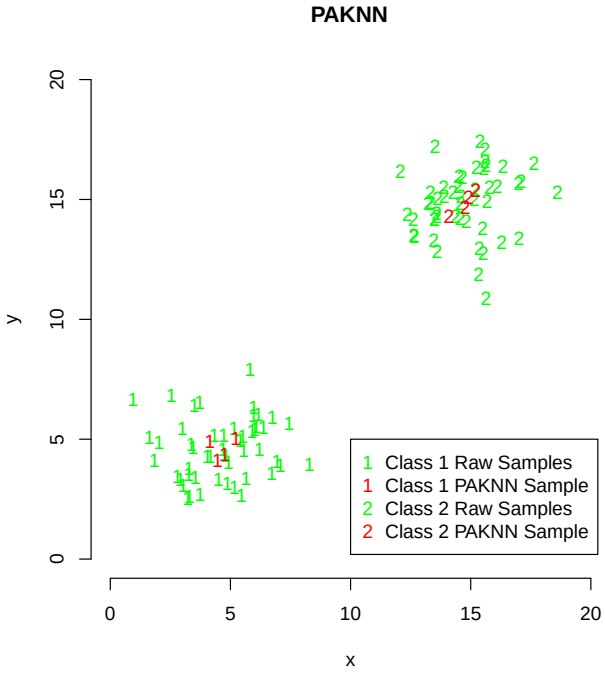

Figure 2: PAKKN averaging process

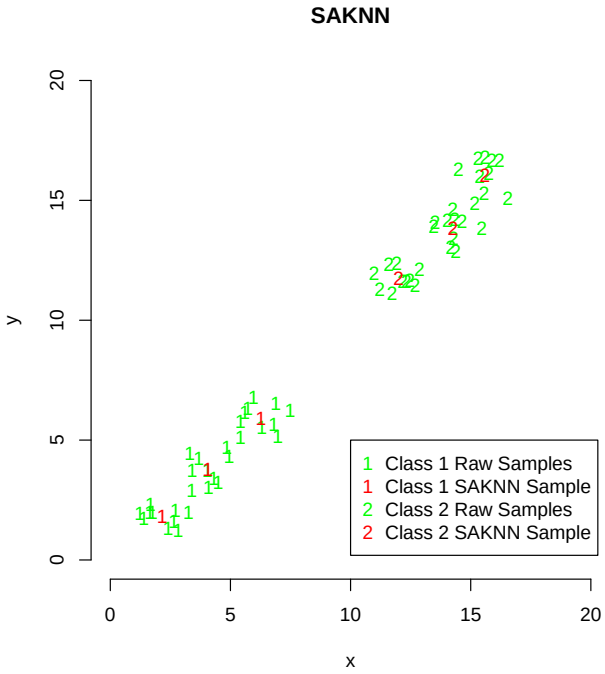

Figure 3: SAKKN averaging process

6 Results

The AKNN, PAKNN, and SAKNN algorithms were realized using the R Programming Language and are not included in this document for the sake of everyone's sanity. The performance of each method proposed is compared against the performance of a standard realization of the KNN algorithm, also in the R Programming Language. Additionally, these performance metrics are also shown for other standard classification methods, such as Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA) and Logistic Regression. Each of these classification methods is tested against several well-established published data sets that are freely available to be used in classification research [12]. Three data sets were chosen to get a cross section of sample size and attributes. Finally, both PDA and kd-trees are compared against the SAKNN method described above.

The Wine data is available at the UCI Machine Learning Repository [13]. The data is a collection of data samples from three classes of wine. For each sample there are *13* attributes of measured parameters which may or may not be good indicators for classification. The group that collected and posted this data stated that it is well-suited for classification, therefore, making it a good candidate upon which to try various KNN incarnations. The data set is comprised of about *175* samples fairly distributed amongst all three classes. This data will be segregated between training and test data sets, ensuring an equal split of each classification among the training and test data set, and models will be developed using the variants of KNN proposed in this paper and other classification algorithms. The results, both the model fidelity and the time required to perform the classification, will be presented so observations about the results may be made.

The Parkinson's data is available at the UCI Machine Learning Repository [14]. The data is a collection of samples from two classes, specifically subjects with Parkinson's and those who do not exhibit the disease. For each sample there are *23* attributes of voice information which may or may not be good indicators for classification. The group that collected and posted this data is trying to use measurable voice information as a classifier, as they feel Parkinson's disease has a profound effect on subtle voice attributes of those who suffer from the disease. The data set is comprised of almost *200* samples from *31* subjects, *23* with Parkinson's, *8* without the disease. As with the Wine data set, this data will be segregated into both a training and test data set, and the various classification methods will be used and then discussed.

The Zip code data set includes two sets of data, a training set and a test set [15]. The training set includes *7,291* samples while the test set includes *2,007* samples. The data to be classified are hand drawn digits. Each digit is classified, *0* through *9*, and *256* attributes are included with each sample which represent a *16*-by-*16* pixel image of the digit. Because the data is already segregated into a training and test data set, this data set does not require partitioning. The already-partitioned data was used by the various classification algorithms and the results are presented below.

Unless otherwise noted, the training fraction used for the wine and Parkinson's data sets was *80*% of the samples devoted to the training models and *20*% devoted to the test phase of the developed models. For the Zip code data, it was already segregated into roughly the same split. This is the typical split employed when presented with a data set.

Once the wine data set was read in, the data was randomly divided between training and test data sets, ensuring an equal distribution of each class, multiple times. For each instantiation of training and test data sets, a number of classification algorithms were used to develop models based on the training data. Once the models were created, the fidelity of each model was computed on both the training and test data sets. The results for each instantiation were saved and then averaged. In addition to the fidelity, the time to perform the classification of both the training and test data sets were saved and then averaged. The results are shown in Table 1 and Table 2. For this data set, sub-groups of *5* samples were used in the SAKNN. In other sections this parameter changes and is noted when changed.

With regard to the model fidelity, there are several observations related to the three KNN variants proposed herein. The model fidelity is better against the training data than against the test data. This is to

Algorithm	Training % Correct	Test % Correct
LDA	90.12	87.84
QDA	99.50	98.92
AKNN	97.73	96.76
PAKNN	97.59	96.76
SAKNN - Sub-groups of 5	98.72	95.97
KNN - 1 Neighbor	100.00	94.59
KNN - 10 Neighbors	97.66	94.59
Logistic Regression	99.22	97.57

Table 1: Model Fidelity for Wine Data Set Using Different Classification Algorithms

Algorithm	Training Class Time (secs)	Test Class Time (secs)
LDA	0.00	0.02
QDA	0.03	0.03
AKNN	0.03	0.01
PAKNN	0.03	0.00
SAKNN - Sub-groups of 5	0.02	0.02
KNN - 1 Neighbor	0.58	0.14
KNN - 10 Neighbors	0.56	0.14
Logistic Regression	0.02	0.01

Table 2: Time to Classify for Wine Data Set Using Different Classification Algorithms

be expected since the three variants use the training data to create the model, so this is not a surprise and should provide a feeling of comfort that the implementations are correct. The fidelity of the three models developed with the proposed variants are as good as the standard classification algorithms and better than the standard KNN algorithm.

Reviewing the time to perform classification, there are also several observations, limited to the AKNN, PAKNN and SAKNN variants offered in this document. The hope is that the proposed KNN variants will require less time to perform robust classification than the original KNN algorithms. As expected, the three variants require significantly less time to classify the test data than the original KNN algorithm. Surprisingly, the three KNN variants require the same order of magnitude of processing time to classify the test data as LDA, QDA, and Logistic Regression.

Next, several experiments were performed where the sub-group size is changed to determine the impact on the performance. Given the small sample size of this data set, there was not much change in the time required to classify the training and test data as the sub-group size changed. Therefore, that data is not presented. However, Figure 4 shows the model fidelity for both the training and test data sets as the sub-group sample size is increased from *5* to *100* in varying steps. As can be seen by this figure, as the sub-group size increases the performance of the model against the test data samples improves. This is a result of the fact that the three classes in the data set are fairly well segregated. This thought is buoyed by this figure and that all the classification methods perform very well for this data set except LDA. Therefore, the methods proposed in this paper appear to have some value.

Once the Parkinson's data set was read in, the data was randomly divided between training and test data sets, ensuring an equal distribution of each class, multiple times. For each instantiation of training and test data sets, a number of classification algorithms were used to develop models based on the training data. Once the models were created, the fidelity of each model was computed on both the training and test data sets. The results for each instantiation were saved and then averaged. In addition to the fidelity, the time to perform the classification of both the training and test data sets were saved and then averaged. The results are shown in Table 3 and Table 4. For this data set, sub-groups of *10* samples were used for the SAKNN method.

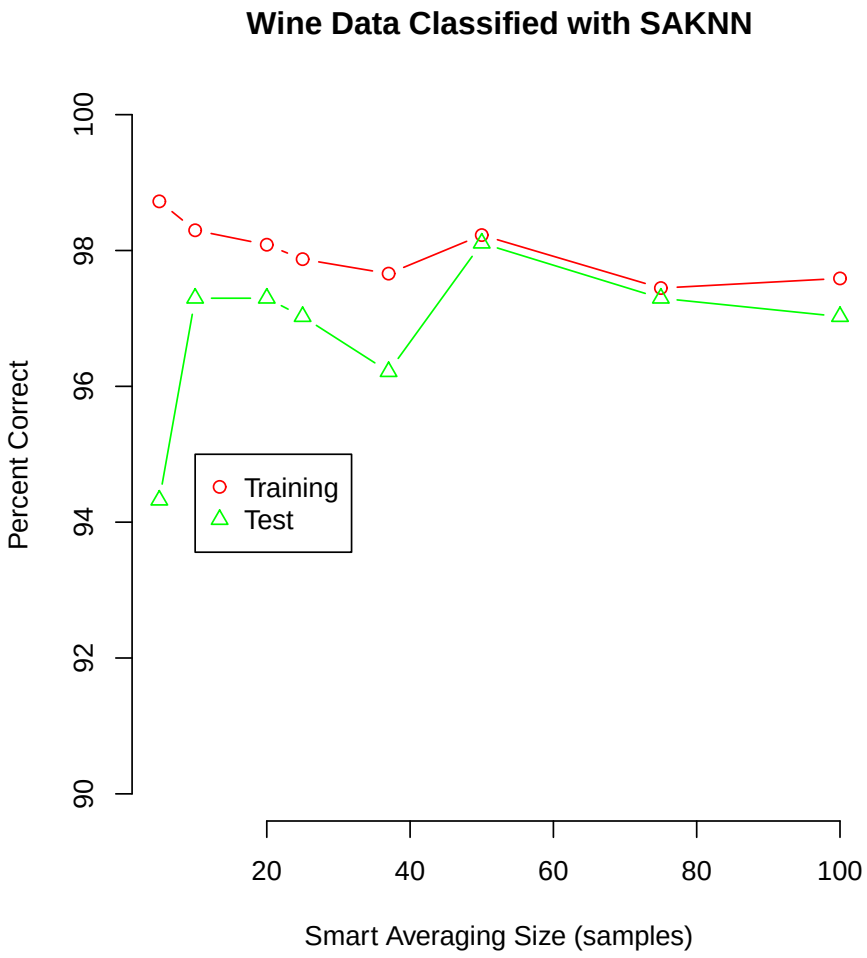

Figure 4: Wine Data Classified with SAKNN, varying sub-group size

Algorithm	Training % Correct	Test % Correct
LDA	74.39	73.25
QDA	98.52	90.00
AKNN	73.87	74.00
PAKNN	73.74	73.75
SAKNN - Sub-groups of 10	90.65	89.25
KNN - 1 Neighbor	100.00	94.75
KNN - 10 Neighbors	90.52	90.75
Logistic Regression	84.19	86.00

Table 3: Model Fidelity for Parkinson's Data Using Different Classification Algorithms

Algorithm	Training Class Time (secs)	Test Class Time (secs)
LDA	0.00	0.00
QDA	0.00	0.00
AKNN	0.02	0.00
PAKNN	0.02	0.00
SAKNN - Sub-groups of 10	0.02	0.00
KNN - 1 Neighbor	0.09	0.04
KNN - 10 Neighbors	0.05	0.02
Logistic Regression	0.00	0.00

Table 4: Time to Classify for Parkinson's Data Using Different Classification Algorithms

There are several observations that should be made about the fidelity of the models. First, all of the models performed worse against this data set than for the wine data set. This indicates that the data is not well-segregated. Additionally, the SAKNN algorithm, when used with sub-groups of *10*, performed almost as well as KNN with *10* neighbors. Otherwise, when compared to the other classification techniques, the performance was adequate, being better than LDA and Logistic Regression, being about the same as QDA and KNN with *10* neighbors, and being worse than KNN with only *1* neighbor. Both the AKNN and PAKNN algorithms performed very poorly and were only marginally better than LDA.

Given the lack of time required to perform classifications, there is little to be gleaned from this data other than standard KNN requires a large time to classify the training and test data as compared to all the other methods employed.

As with the wine data set, several experiments were executed where the sub-group size was changed and the results recorded. Given the similarity to the wine data set, there was not much change in the time required to classify the training and test data as the sub-group size changed. Therefore, that data is not presented. However, Figure 5 shows the model fidelity for both the training and test data sets as the sub-group sample size is increased from *5* to *100* in varying steps. As can be seen by this figure, there is definitely a correlation between the sub-group size and the model fidelity. As the sub-group size is increased, the model performance is worse. This is to be expected since the more and more samples we average, the more distorted the averaged sample is from the true underlying data. Also, this is a result of the data not being as well-segregated as it was for the wine data set. Therefore, the proposed SAKNN method may have some value as long as the sub-group size is correctly chosen. When the sub-group size is small, the model is between *85* and *90*% accurate. However, both AKNN and PAKNN do not provide sufficient fidelity based on the results against this data set.

Once the Zip code data was read in, it was already segregated between a training and test data set. Fortunately, each algorithm was only used once to develop a model, and then twice to perform classification (once with training data and once with the test data). Given the similarity of the data samples, both LDA and QDA could not be executed since it resulted in a singularity when each tried to compute the covariance matrix. The Logistic Regression algorithm has also been excluded since the ratio of attributes to samples per class is very high, as there are *256* attributes and about *700* samples per class. Also, a number of the attributes appear to be collinear, which affects not only Logistic Regression but also LDA and QDA. Therefore, only KNN based algorithms are presented. Once the models were created, the fidelity of each model was computed on both the training and test data sets. In addition to the fidelity, the time to perform the classification of both the training and test data sets were saved. Given that this data set dwarfs the Parkinson's and wine data sets, the time to perform classification is significant and the effects of pre-processing the training data for all three proposed KNN variants should be plainly evident. The results are shown in Table 5 and Table 6.

There are several observations that should be made about the fidelity of the models developed for the Zip code data. These results are surprising, as the *1* nearest neighbor version of KNN is especially accurate.

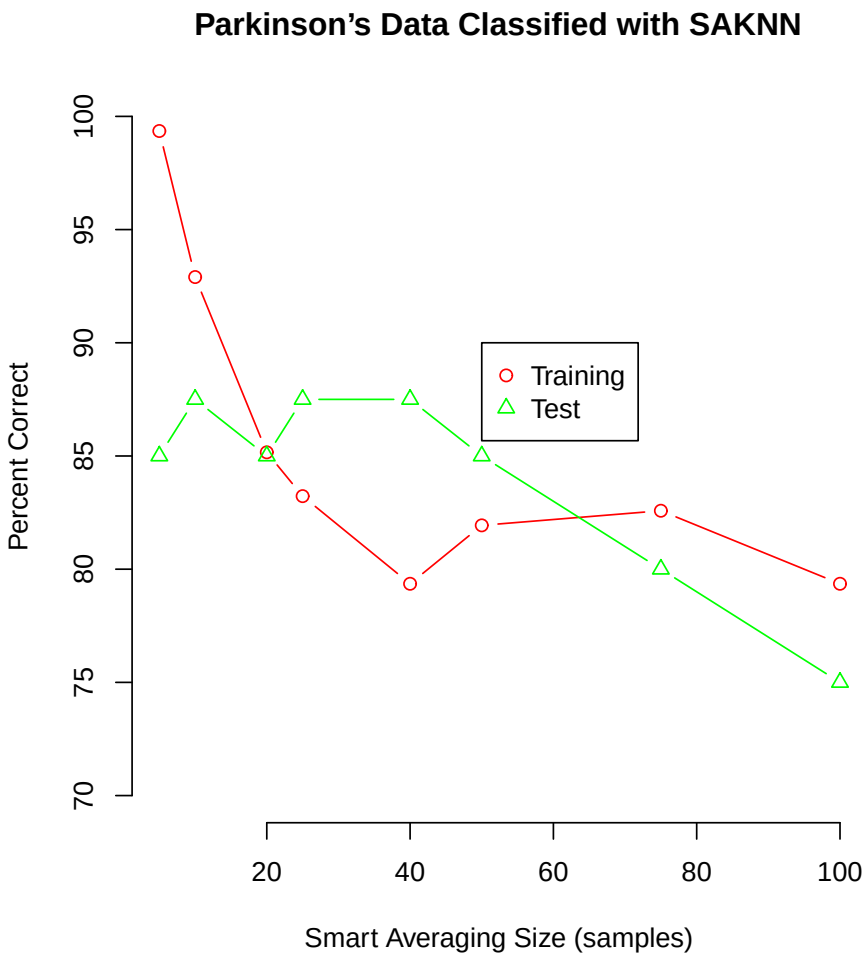

Figure 5: Parkinson's Data Classified with SAKNN, varying sub-group size

Algorithm	Training % Correct	Test % Correct
AKNN	82.61	82.36
PAKNN	83.01	82.81
SAKNN - Sub-groups of 10	97.72	98.70
KNN - 1 Neighbor	100.00	100.00
KNN - 10 Neighbors	96.06	96.41

Table 5: Model Fidelity for Zip Code Data Using Different Classification Algorithms

Algorithm	Training Class Time (secs)	Test Class Time (secs)
AKNN	2.96	0.85
PAKNN	6.14	1.69
SAKNN - Sub-groups of 10	74.97	20.59
KNN - 1 Neighbor	3239.88	869.57
KNN - 10 Neighbors	3178.06	874.22

Table 6: Time to Classify for Zip Code Data Using Different Classification Algorithms

Also surprising is how well the SAKNN algorithm performed. The AKNN and PAKNN performance is not good but this data set is very complex, has a huge number of attributes, and the averaging is being done over a large group of samples. While the performance is not good, there may be some applications which only require *80*% accuracy. According to the website that provides this data, getting an error of less than *2.5*% is considered excellent. Based on these results, SAKNN shows promise.

Reviewing the time to perform classification of this data set also provides some exciting results. Standard KNN algorithms took *40* times longer to classify the data with no appreciable gain in model fidelity.

As with the other two data sets, several experiments were conducted where the sub-group size was changed and the results recorded for analysis. Given the massive size of this data set, it is expected to see a real difference in time required to classify data as the sub-group size is changed. Figure 6 shows six plots. The top two plots are the percent correct for the training and test data sets as a function of execution time. I would expect that the longer the execution time, the better model performance. In this case, this is true given the displayed curves. The middle two plots are the model fidelity versus sub-group sample size. As expected, for both data sets the model fidelity worsens as the sub-group size is increased. This is to be expected, since as the sub-group size increases, we are averaging across samples that are spread further. The bottom two plots show the time to classify versus the sub-group size. As expected, for both training and test data, the larger the sub-group size, the faster the classification. Larger sub-groups yield a smaller sample space that must be searched when classification is performed. SAKNN performs as well as the other classification algorithms on this data set. The results on this data set are the "smoking gun" that reveals real potential in the SAKNN algorithm.

This section will compare the performance of the SAKNN algorithm against both PCA and Kd-trees. For PCA, the previously used data sets will be re-used and models will be developed after the raw data has been pre-processed with PCA. The question to answer is whether or not SAKNN yields superior performance (fidelity and real-time) when compared to PCA. All the classification algorithms are used after PCA has been applied. For the three previously used data sets, the Eigen values were determined and any attribute with an Eigen value of less than *1* was discarded, as its contribution is considered to be non substantial.

First to be pre-processed via PCA was the wine data. The results are shown in Table 7 and Table 8. There are three interesting items that should be noted from the data in these tables. First, all the classification algorithms performed worse once PCA was applied, except LDA. This is to be expected, since we are removing some level of information within the data when we whisk away attributes. In this case, the models were not over-fitting the data since removing attributes did lower each models fidelity (except LDA). The second item to note is that the time required to perform the classification was greatly reduced for the standard KNN algorithm. In fact, from a time required to classify standpoint, the PCA pre-processed KNN algorithm performed as well as the SAKNN algorithm. The third item of note is that, for this data set, the SAKNN algorithm provided a truer model as compared to the PCA pre-processed KNN algorithm. From Table 1, the SAKNN was almost *96*% accurate while the standard KNN algorithm, pre-processed by PCA, was about *92*% accurate.

Next to be pre-processed via PCA was the Parkinson's data. The results are shown in Table 9 and Table 10. These results are eerily similar to the results seen with the wine data. All of the classification

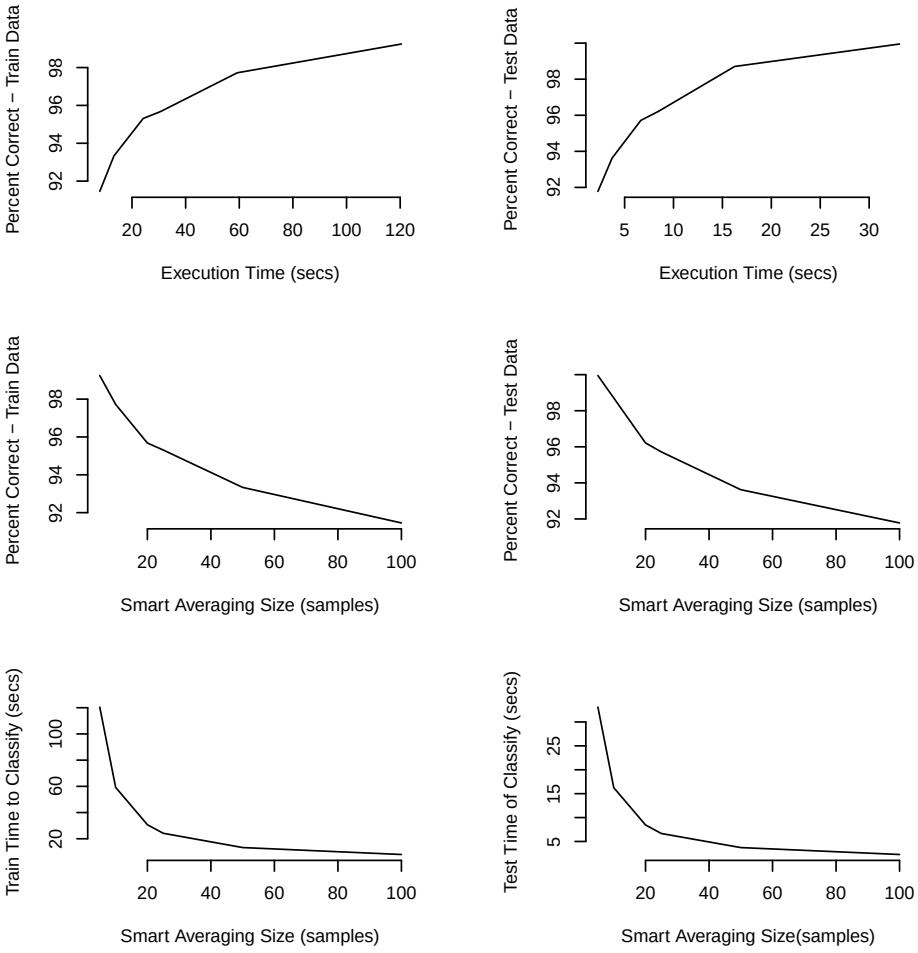

Figure 6: Zip Code Data Classified with SAKNN, varying sub-group size

Algorithm	Training % Correct	Test % Correct
LDA	92.20	90.54
QDA	96.24	93.51
AKNN	95.04	93.51
PAKNN	94.33	92.70
SAKNN - Sub-groups of 5	94.68	92.16
KNN - 1 Neighbor	100.00	92.43
KNN - 10 Neighbors	95.46	91.89
Logistic Regression	95.53	94.86

Table 7: Model Fidelity for Wine Data Set Using PCA to pre-process data

Algorithm	Training Class Time (secs)	Test Class Time (secs)
LDA	0.00	0.00
QDA	0.00	0.00
AKNN	0.01	0.00
PAKNN	0.02	0.00
SAKNN - Sub-groups of 5	0.01	0.00
KNN - 1 Neighbor	0.02	0.01
KNN - 10 Neighbors	0.01	0.00
Logistic Regression	0.02	0.00

Table 8: Time to Classify for Wine Data Set Using PCA to pre-process data

Algorithm	Training % Correct	Test % Correct
LDA	77.48	77.25
QDA	76.26	75.75
AKNN	71.29	70.25
PAKNN	72.58	71.50
SAKNN - Sub-groups of 5	84.97	77.25
KNN - 1 Neighbor	100.00	85.00
KNN - 10 Neighbors	85.03	81.75
Logistic Regression	82.71	82.50

Table 9: Model Fidelity for Parkinson's Data Set Using PCA to pre-process data

algorithms performed worse once PCA was applied, which as noted above, is to be expected. Again, this tells us the pre-PCA models were not over-fit to the data. Also, the time required to perform the classification was greatly reduced for the standard KNN algorithm. From a time required to classify standpoint, the PCA pre-processed KNN algorithm performed as well as the SAKNN algorithm, however, the SAKNN algorithm provided a truer model as compared to the PCA pre-processed KNN algorithm. From Table 3, the SAKNN was almost *89*% accurate while the standard KNN algorithm, pre-processed by PCA, was approximately *85*% accurate or less depending on the number of neighbors used.

Algorithm	Training Class Time (secs)	Test Class Time (secs)
LDA	0.00	0.00
QDA	0.00	0.00
AKNN	0.02	0.00
PAKNN	0.02	0.01
SAKNN - Sub-groups of 5	0.01	0.00
KNN - 1 Neighbor	0.04	0.01
KNN - 10 Neighbors	0.02	0.00
Logistic Regression	0.00	0.00

Table 10: Time to Classify for Parkinson's Data Set Using PCA to pre-process data

Algorithm	Training % Correct	Test % Correct
AKNN	88.45	88.59
PAKNN	88.70	88.79
SAKNN - Sub-groups of 10	98.09	99.15
KNN - 1 Neighbor	100.00	100.00
KNN - 10 Neighbors	96.85	96.61

Table 11: Model Fidelity for Zip Code Data Set Using PCA to pre-process data

The Zip Code data set was pre-processed with PCA and then evaluated. The results are shown in Table 11 and Table 12. All of the classification algorithms used performed better once PCA was applied to the data to be classified. Note that even SAKNN improved to over *99*% accuracy. The time to perform classification behaves similarly to the other data sets when PCA was applied, specifically, less time was required to classify. This is to be expected since the dimensionality of the data was reduced. One important item to note is that the time to classify using SAKNN was *5* times less than standard KNN, and provided very similar accuracy.

Based on the above data, it is demonstrated that while PCA can be used to make the time required to perform classification using KNN less, it still requires much more time to perform classifications than if SAKNN was used either with or without PCA, while not providing an appreciable fidelity improvement. And in the above experiments, the time savings gained by using SAKNN leads to less fidelity loss of the model than if PCA was applied.

Next, we need to compare the SAKNN algorithm proposed herein with using kd trees to perform the standard KNN algorithm. After much reading, it seems that kd trees suffer from the "curse of dimensionality" where, as the number of dimensions in the data increases, the less beneficial the tree structure is in reducing the search time [16]. It appears that after approximately six dimensions, the performance advantage provided by kd-trees is nullified. One reason for this is that as the number of dimensions increases, the training sample density decreases as the training samples are farther from each other. This results in more nodes of the kd-tree having to be searched to find the minimum. A good rule of thumb is that kd-trees are beneficial when:

$$N >> 2^D \tag{1}$$

Algorithm	Training Class Time (secs)	Test Class Time (secs)
AKNN	1.05	0.27
PAKNN	1.42	0.36
SAKNN - Sub-groups of 10	11.15	3.05
KNN - 1 Neighbor	61.94	16.66
KNN - 10 Neighbors	60.86	16.74

Table 12: Time to Classify for Zip Data Set Using PCA to pre-process data

Data Set	Training Set Size	$2^{Dimensions}$
Wine Data	142	8192
Parkinson's Data	156	4194304
Zip Code Data	7291	1.157921e+77

Table 13: Training set size versus Dimensionality for various data sets

where N is the number of samples in the training set and D is the dimensionality of the data. In the case of the three data sets used in this paper, Table 13 shows the relationship between the dimensions and the samples in the training set.

All three data sets are woefully short of training samples to make good use of the kd-tree structure, based on this rule of thumb. It would appear, from this table, that the training data sets are sparsely populated given the dimensionality. The problem with kd-trees and dimensionality relates to the need to perform more comparisons as the number of dimensions increases. Assuming a balanced kd-tree (knowing the training data beforehand, which in the examples presented in this paper is true), every level of the kd-tree will be sorted or segregated, based on the value in a particular dimension. Just because the sample to be classified may be near (considering distance) the samples in a particular portion of the tree, based on traversing the tree, the other portions of the tree may need to be searched. For example, if we assume a very simple example that has two attributes, the kd-tree would use each dimension to segregate the data at each level of the tree, alternating until all the samples were placed in the tree. Now when a sample is to be classified, the tree needs to be traversed to find the closest neighbor. The samples to be traversed may be on different sides of the kd-tree. The sample to be classified is far from all the samples. Imagine many more dimensions, where the sample to be classified is close to some training samples in some dimensions, and close to other samples in other dimensions. All three KNN variants proposed in this paper do not suffer from this problem because the number of training samples has been reduced, hence, there is no way that all the training samples have to be traversed.

7 Conclusion

In this paper, three methods have been demonstrated that can be used prior to performing classification with the standard KNN algorithm. All of these methods greatly reduce the time required to perform KNN classification of test samples. While each of the algorithms reduces the amount of time required to perform classification, the fidelity of the developed model remains high for the SAKNN method. The fidelity of the model developed using KNN after applying either ANN or PAKNN is not as true, however, there may be applications where this level of fidelity is sufficient. In the future, more "smart" averaging can be performed to find the best combination of training samples to minimize the distance between averaged training samples, while maximizing the fidelity of the algorithm.

References

[1] *The R Project for Statistical Computing*, `http://www.r-project.org/`, Retrieved November 30th, 2009.

[2] *cell-knn*, `http://code.google.com/p/cell-knn/wiki/kNN`, Retrieved November 30th, 2009.

[3] *Image Restoration using a KNN-variant of the Mean-Shift*, Angelino, Cesario Vincenzo, `http://www-video.eecs.berkeley.edu/Proceedings/ICIP2008/pdfs/0000573.pdf`, Retrieved November 30th, 2009.

[4] *Introduction to Machine Learning, Lecture 7, Instance Based Learning*, Puig, Albert, `http://www.slideshare.net/aorriols/lecture7-ibk-1048631`, Retrieved November 30th, 2009.

[5] *Principal Component Analysis*, http://ordination.okstate.edu/PCA.htm, Palmer, Michael, Retrieved November 30th, 2009.

[6] *Discriminant Function Analysis*, http://faculty.chass.ncsu.edu/garson/PA765/discrim.htm, Garson, G. David, Retrieved November 30th, 2009.

[7] *Introduction to kd-trees*, http://www.cse.iitb.ac.in/~sharat/current/cs663/notes/pKdtrees.pdf, Retrieved November 30th, 2009.

[8] *Nearest Neighbor Classification*, Fleming, Jesse, www.cs.uvm.edu/~xwu/kdd/kNN-09.ppt, Retrieved November 30th, 2009.

[9] *An Introductory Tutorial of kd Trees*, Moore, Andrew, http://www.autonlab.org/autonweb/14665/version/2/part/5/data/moore-tutorial.pdf?branch=main&language=en, Retrieved November 30th, 2009.

[10] *Nearest Neighbor Search in General Metric Spaces Using a Tree Data Structure with a Simple Heuristic*, Xu, Huafeng, http://www.dimitris-agrafiotis.com/Papers/ci034150f.pdf, Retrieved December 1st, 2009.

[11] *Machine Learning: Nearest Neighbor*, Ho, Ricky, http://horicky.blogspot.com/2009/05/machine-learning-nearest-neighbor.html, Retrieved December 1st, 2009.

[12] *UCI Machine Learning Repository*, http://archive.ics.uci.edu/ml/, Retrieved November 30th, 2009.

[13] *Wine Data Set*, http://archive.ics.uci.edu/ml/datasets/Wine, Retrieved December 1st, 2009.

[14] *Parkinson's Data Set*, http://archive.ics.uci.edu/ml/datasets/Parkinsons, Retrieved December 1st, 2009.

[15] *Normalized Handwritten Digits*, http://www-stat.stanford.edu/~tibs/ElemStatLearn/index.html, Retrieved December 1st, 2009.

[16] *Optimizing Search Strategies in k-d Trees*, Sample, Neal, http://www-graphics.stanford.edu/~tpurcell/pubs/search.pdf, Retrieved December 3rd, 2009.