

An Empirical Comparison of Support Vector Machines Versus Nearest Neighbour Methods for Machine Learning Applications

Mori Gamboni, Abhijai Garg, Oleg Grishin, Seung Man Oh, Francis Sowani, Anthony Spalvieri-Kruse, Godfried T. Toussaint^(✉), and Lingliang Zhang

Faculty of Science, New York University Abu Dhabi,
P.O. Box 129188, Saadiyat Island, Abu Dhabi, United Arab Emirates
{mg3794, ag3754, og402, smo304, fts215, ask417, gt42, lz781}@nyu.edu

Abstract. Support vector machines (SVMs) are traditionally considered to be the best classifiers in terms of minimizing the empirical probability of misclassification, although they can be slow when the training datasets are large. Here SVMs are compared to the classic k-Nearest Neighbour (k-NN) decision rule using seven large real-world datasets obtained from the University of California at Irvine (UCI) Machine Learning Repository. To counterbalance the slowness of SVMs on large datasets, three simple and fast methods for reducing the size of the training data, and thus speeding up the SVMs are incorporated. One is blind random sampling. The other two are new linear-time methods for guided random sampling which we call Gaussian Condensing and Gaussian Smoothing. In spite of the speedups of SVMs obtained by incorporating Gaussian Smoothing and Condensing, the results obtained show that k-NN methods are superior to SVMs on most of the seven data sets used, and cast doubt on the general superiority of SVMs. Furthermore, random sampling works surprisingly well and is robust, suggesting that it is a worthwhile preprocessing step to either SVMs or k-NN.

Keywords: Machine learning · Data mining · Support vector machines · SMO · Training data condensation · k-Nearest neighbour methods · Blind and guided random sampling · Wilson editing · Gaussian Condensing

1 Introduction

One of the most attractive learning machine models for pattern recognition applications, from the point of view of high classification accuracy, appears to be the support vector machine (SVM) [1]. There exists empirical evidence that SVMs yield lower rates of misclassification than even the classical k-Nearest Neighbour (k-NN) rule [2], in spite of the fact that (at least in theory) the latter is asymptotically Bayes optimal for all underlying probability distributions [3]. A major drawback of SVMs is their worst-case complexity, which is $O(N^3)$, where N is the

number of instances in the training set, so that for very large datasets the training time may become prohibitive [4]. Therefore much effort has been devoted to finding ways to speed up SVMs [5–12]. The simplest approach is to select a small random sample of the data for training [13]. This method is called *blind random sampling* because it uses no explicit information about the underlying structure of the data. This approach may be trivially implemented in $O(N)$ worst-case time. Non-blind random sampling techniques such as Progressive Sampling (PS) and Guided Progressive Sampling (GPS) have also been investigated with some success [14–16]. Non-random sampling methods attempt to use intelligent data analysis such as genetic algorithms [17] or proximity graphs [2, 11] to preselect a supposedly better representative subset of the training data, which is then fed to the SVM, in lieu of the large original set of data. However, the use of guided data condensation methods usually incurs an additional worst-case cost of $O(N \log N)$ to $O(N^3)$. Since 1968, the literature contains a plethora of such algorithms and heuristics of varying degrees of computational complexity, for preselecting small subsets of the training data that will perform well under a variety of circumstances [18–20]. Although such techniques naturally speed up the training phase of the SVMs, by virtue of the smaller size of the training data, many studies primarily focus on (and report) only the number of support vectors retained, ignoring the additional time taken to perform the pre-selection. Indeed, it has been shown empirically, for methods that used proximity graphs for training data condensation, that if the additional time taken by the pre-selection step is taken into consideration, the overall training time is generally much worse than that of simple blind random sampling [11].

Some hybrid methods that combine blind random sampling with structured search for good representatives of datasets have also been tried. An original method combining blind random sampling with SVM and near neighbour search has recently been suggested by Li, Cervantes, and Yu [10]. Their approach first uses blind random sampling to select a small subset of the data, from which the support vectors are extracted using a preliminary SVM. These support vectors are then used to select points from the original training set (the data recovery step) that are near the preliminary support vectors, thus yielding the condensed training set on which the final SVM is applied.

In this paper several methods for speeding up the running time of SVMs are compared in terms of the speed-up factor and the classification accuracy, using seven large real world datasets taken from the University of California at Irvine (UCI) Machine Learning Repository [21]. All the methods are based on efficiently reducing the size of the training data that is subsequently fed to an SVM with sequential minimal optimization (SMO), whilst maintaining a high classification accuracy. Three probabilistic methods are investigated that run in $O(N)$ worst-case time. The first is blind random sampling and the second and third are new methods proposed here for guided random sampling (called Gaussian Condensing and Gaussian Smoothing), as well as an algorithm which can combine these classifiers that can sometimes yield better results. These are compared with nearest neighbour methods for reducing the size of the training

set (k-NN condensation) and for smoothing the decision boundary (Wilson editing), both of which run in $O(N^2)$ worst-case time. One aim of this paper was also to compare SVMs with their leading competitor, the k-NN rule; thus, every method for reducing the size of the training data was also tested on a k-NN classifier.

2 The Classifiers Tested

2.1 Blind Random Sampling

Blind random sampling is the simplest method for reducing the size of the training set, both conceptually and computationally, with a running time of $O(N)$. Its possible drawback is that it is blind with respect to the quality of the resulting reduced training set, although this need not result in poor performance. In the experiments reported here the percentages of training data randomly selected for training the SVM were varied from 10% to 90% in increments of 10%.

2.2 Wilson Editing (Smoothing)

Wilson's editing algorithm was used for smoothing the decision boundary [22]. Each instance in the training set is classified using the k-Nearest Neighbour rule by means of a *majority* vote. If the instance is misclassified it is *marked*. After all instances have been classified, all the marked points are deleted. This condensed set is then used as the testing set. Wilson editing was not designed to significantly reduce the size of the training set; its goal is rather to improve classification accuracy, and is used here as a pre-processing step before reducing the training set further with methods tailored for that purpose. Using packages available in the Weka Machine Learning Software [23], a value of $k = 3$ was chosen, and thus Wilson editing runs in $O(N^2)$ worst-case time with a straightforward, naïve implementation. Wilson editing was preceded by blind random sampling in a second set of experiments (see Gaussian Smoothing below).

2.3 k-Nearest-Neighbour Condensation

When all k nearest neighbours of a point X belong to the class of X, the k-NN rule makes a decision with very high confidence. In other words the point X is located relatively far from the decision boundary. This suggests that many points with this property could be safely deleted. Before classifying each testing set, the corresponding training set is condensed as follows. Each instance in the training set is classified using the k-NN rule. If the instance is correctly classified with *very high confidence* it is marked. After all instances are classified, all marked points are deleted. This condensed set is then used to classify the testing set. High confidence in the classification of X is measured by the proportion of the k nearest neighbours of X that belong to the class of X. The standard k-NN rule uses a *majority* vote as its measure of confidence. In our approach we use the

unanimity vote (all the k nearest neighbours belong to the same class) and select a good value of k . This algorithm runs in $O(N^2)$ worst-case time using the naïve straightforward implementation and packages available in Weka. Note that when data of different classes are widely separated it may happen (at least in theory) that for every point X its k nearest neighbours all belong to the class of X . In such a situation unbridled k -NN condensation might discard the entire training set. For such an eventuality, if for some pattern class all training instances are marked for deletion, the mean of those instances is retained as the representative of that class. Experiments were also performed with k -NN condensation preceded by Wilson editing.

2.4 Gaussian Condensing

Gaussian Condensing is a novel heuristically guided random sampling algorithm introduced here. The heuristic implemented assumes that instances with feature values relatively close to the mean of their own class are likely to be furthest from the decision boundary, and therefore not expected to contain much discrimination information. Conversely, points relatively far from the mean are likely to be closer to the decision boundary, and expected to contain the most useful information. First, for each class, the mean value of each feature is calculated. Then, for each feature in every instance, the ratio between the Gaussian function of the feature value of that instance and the Gaussian function of the mean is computed. This determines a parameter termed the *partial discarding probability*. Finally, all instances are discarded probabilistically in parallel with a probability equal to the mean of the partial discarding probabilities of all their features. The main attractive attribute of this algorithm is that it runs in $O(N)$ worst-case time, where N is the number of training instances. It is therefore linear with respect to the size of the training data, and thus much faster than previous discarding methods that use proximity graphs, which are either quadratic or cubic in N . Indeed, the complexity of Gaussian Condensing is as low as that of blind random sampling.

The goal of Gaussian Condensing is to invert the probability distribution function of instances for all features of each class. Hence, points near the mean are certain to be thrown away, and points near the boundaries are almost never thrown away. If applied to data with a Gaussian distribution, the probability distribution function would result in an inverted bell curve, with the minimum point occurring at the center, and increasing towards the boundaries before decreasing again. A similar idea was introduced by Chen, Zhang, Xue, and Liu, [12], with strong results. However, their algorithm deletes a ratio of the total data closest to the mean. The approach proposed here is superior in two ways: (1) it does not require a method to decide the ratio of data that should be optimally kept, and (2) it does not create a hole? in the data, but rather preserves the entire distribution of points, by simply altering the density. Experiments were also performed with Gaussian Condensing preceded by Wilson editing.

2.5 Gaussian Smoothing

Gaussian Smoothing is another original method presented here that was added after experiments were done with Gaussian Condensing. This algorithm can be considered the opposite of Gaussian Condensing, in that it tries to discard points which are near the boundary. First, the mean feature value of every feature in every class is computed, then, lists of these values are grouped by feature and independently sorted (since the number of features is very small compared to the number of instances, sorting time does not dominate the total running time of the algorithm). For every feature of every instance, if the Gaussian function of the feature value of that instance is smaller than the Gaussian function of the closest mean feature value regardless of the class, the partial discarding probability is the ratio between the two values subtracted from one ($p = 1 - \frac{\text{instance}}{\text{neighbor}}$), otherwise, the partial discarding probability is zero. The mean of the partial discarding probabilities gives the final discarding probability and instances are discarded in parallel, exactly as in Gaussian Condensing. Thus, the partial discarding probability starts to increase for points when they are closer to the center of the neighboring class than the center of their own class (the partial discarding probability is zero where the two Gaussian functions intersect). Gaussian Smoothing thus runs in $O(N)$ worst-case time. k-NN condensation was not included in the second set of experiments as it performed poorly (see Sect. 4.2); Gaussian Smoothing was tested alone, in conjunction with Gaussian Condensing (in both possible orders), and preceded by blind random sampling.

2.6 Combined Gaussian Filter

In practice, using both Gaussian Condensing and Smoothing does not always give good accuracies (see Subsect. 4.2). However, a slight modification of the algorithm can sometimes give better results. To calculate the partial probabilities, the methods of condensing and smoothing are applied when the Gaussian function of the feature value of an instance is respectively higher than or lower than or equal to that of the closest mean feature value. In other words, if the Gaussian function of the feature value of the instance is lower than or equal to the closest mean feature value, then the partial discarding probability is given by the ratio of those two values subtracted by one (smoothing), if it is higher, the probability is given by the ratio between the Gaussian function of the feature value and the mean feature value of the class of the instance (condensing). The final discarding probability is calculated – and the points discarded – in the same manner as the other methods. This algorithm ideally causes the probability distribution to become inverted near the boundary of the classes, which is the case with perfect normal distributions, thus potentially keeping the best attributes of both the condensing and smoothing methods. The Combined Gaussian Filter runs in $O(N)$ for a constant number of classes, just as in Gaussian Smoothing.

3 The Datasets Tested

Year Prediction Million Song Data. The original dataset is extremely large, (515,345 instances) and therefore some of the data were randomly discarded. The pattern classes were converted from years to decades (1950 s through 2000 s) and then 3,000 instances of each class were chosen, comprising six classes with a total of 18,000 instances.

Letter Image Data. This dataset contains black-and-white rectangular pixel displays of the 26 upper-case letters in the English alphabet. The letter images were constructed from twenty different fonts. Each letter from the twenty fonts was randomly distorted to produce 20,000 unique instances. Each instance is described using 17 attributes: a letter category (A, B, C,..., Z) and 16 numeric features.

Wearable Computing Data. This dataset (PUC-Rio) contains information matching accelerometer readings from various parts of the human body, with the readings taken while the actions were performed. Accelerometers collected x , y , and z axes data from the waist, left-thigh, right ankle, and right upper-arm of the subjects. In each instance, the subjects were either sitting-down, standing-up, walking, in the process of standing, or in the process of sitting. Metadata about the gender, age, height, weight and BMI of each subject are also provided. In total 165,632 instances of such data are included in this dataset.

MAGIC Gamma Telescope Data. This data consist of Monte-Carlo generated simulations of high-energy gamma particles. There are ten attributes, each continuous, and two classes ('g' and 'h'). The number of instances was 12,332 for 'g' and 6,688 for 'h'. For the purpose of this study approximately half of class 'g' was removed at random, since the goal of the present research is the improvement of the running time of SVMs, rather than the minimization of the probability of misclassification for this particular application.

Spambase Data. The Spambase dataset provides information about email spam. The emails are classified into two categories: spam and non-spam. The data labelled spam were collected from postmasters and individuals who had reported spam, and the non-spam data were collected from filed work and personal emails. The dataset was created with the goal of designing a personal spam filter. It contains 4601 instances, of which 1813 (39.4%) are spam. These instances are characterized by 57 attributes (57 continuous features and one nominal class label). The class label is either 1 or 0, indicating that the email is either spam or non-spam, respectively.

Wine Quality Data. The white wine quality dataset includes over 2000 different *vinho verde* wines (instances). The dataset comprises twelve features that include acidity and sulphate content. There are ten classes defined in terms of quality ratings that vary between 1 and 10.

Handwritten Digits Data. This dataset contains 32 by 32 bitmaps that have been obtained by centering and normalising the input images from 43 different people. The training set consisting of 5,620 instances and has data from 30 people, while the test set comes from the 13 others, so as to prevent learning algorithms from classifying digits based on the writing style rather than features of the shape of the digits themselves. To decrease the dimensionality of the data, the bitmaps are divided into 4 by 4 blocks and the number of pixels in each block is counted. The total number of features is thus 64 and the number of classes is 10, the digits 0 through 9.

4 Results and Discussion

Note: Due to space restrictions, only particularly illustrative graphs are shown.

4.1 The Computation Platform

The timing experiments were performed on the fastest high-performance computer available in the United Arab Emirates (second fastest in the Gulf region): *BuTinah*, operated by New York University Abu Dhabi. The computer consists of 512 nodes, each one equipped with 12 Intel XeonX5675 CPU's clocked at 3.07 GHz and 48 GB of RAM with 10 GB of swap memory. *BuTinah* operates at approximately 70 trillion floating-point operations per second (70 teraflops). The experiments utilized seven nodes, in total, consuming 9h of computation time and 12 GB of memory. The testing environment was programmed in Java, using the Weka data Mining Package, produced by the University of Waikato.

4.2 Blind Random Sampling

The SMO (Sequential Minimization Optimization) version of SVM – invented by John Platt [24] and improved by Keerthi, Shevade, Bhattacharyya, and Murthy [25] – that is installed in the Weka machine learning package was compared to the classical k-NN decision rule when both are preceded by blind random removal of data before feeding the remaining data to each classifier. A typical result obtained with the Song dataset is shown in Fig. 1, for the classification accuracy (left vertical axis) and the total running time (right vertical axis). Total time refers to the sum of the times taken for training data condensation, training time, and testing time (results for the three individual timings will be presented in a following section). In this and all other experiments the classification accuracies and timings were obtained by the method of K-fold cross-validation (or Π

method) with a value of $K = 10$ [26]. This means that for each of the classifiers and condensing methods tested the procedure for estimating the classification accuracy for each fold was the following. Let X denote the entire dataset. The i -th fold is obtained by taking the i -th 10% of $\{X\}$ as the testing set (denoted by $\{X_{TS-i}\}$), and the remaining 90% of the data as the training set (denoted by $\{X_{TR-i}\}$). Estimates of the misclassification accuracy of any classifier are then obtained by training the classifier on $\{X_{TR-i}\}$, and testing it on $\{X_{TS-i}\}$, for $i = 1, 2, \dots, 10$ yielding a total of ten estimates. Similarly, when estimating the classification accuracy of an editing (or condensing) method, the editing (or condensing) is first applied to $\{X_{TR-i}\}$, and the resulting edited (condensed) set is used to classify $\{X_{TS-i}\}$. Finally, in all cases the average of the ten estimates obtained in this way is calculated. Thus the results shown in the figures are the mean values over the ten folds. This method also permits the computation of standard deviations (over the ten folds) to serve as indicators of statistically significant differences between the means. The error bars in the figures indicate \pm one standard deviation.

All seven datasets exhibit similar behaviour to that depicted in Fig. 1 with respect to how the classification accuracy varies as a function of the percentage of training data removed. The classification accuracy results are not unanimous, but favour k-NN over SMO, the latter having significantly better accuracy than k-NN only for the Song data (Fig. 1). For the Letter Image, Wearable Computing, and MAGIC Gamma datasets k-NN did significantly better (e.g. Letter Image data set in Fig. 2). Furthermore, for the Wine and Handwritten Digits data no significant differences were observed between SMO and k-NN (Figure not shown). For the Spambase data SMO is significantly better only when 60%, 70% or 90% of the data are discarded (Figure not shown).

With respect to the total time taken, for all the datasets, SMO takes considerably less running time than k-NN, and all show similar behaviour. For example,

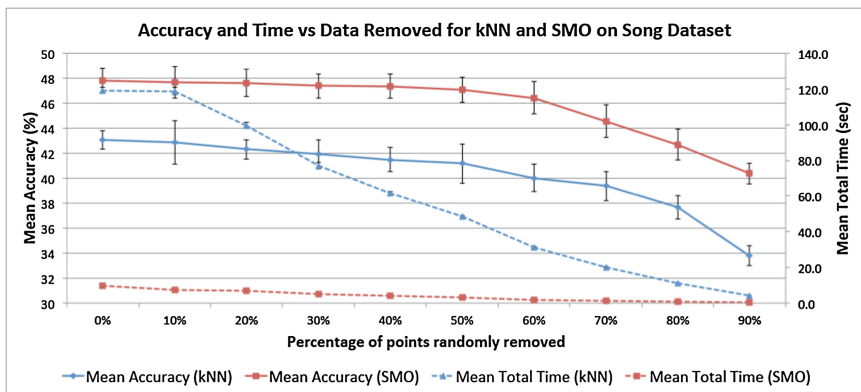


Fig. 1. Accuracy and total time vs. % of training data removed by blind random sampling for the Year Prediction Million Song data.

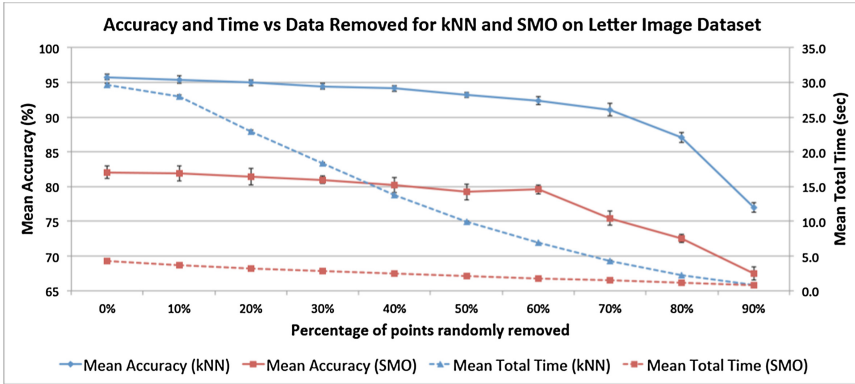


Fig. 2. Accuracy and total time vs. % of training data removed by blind random sampling for the Letter Image data.

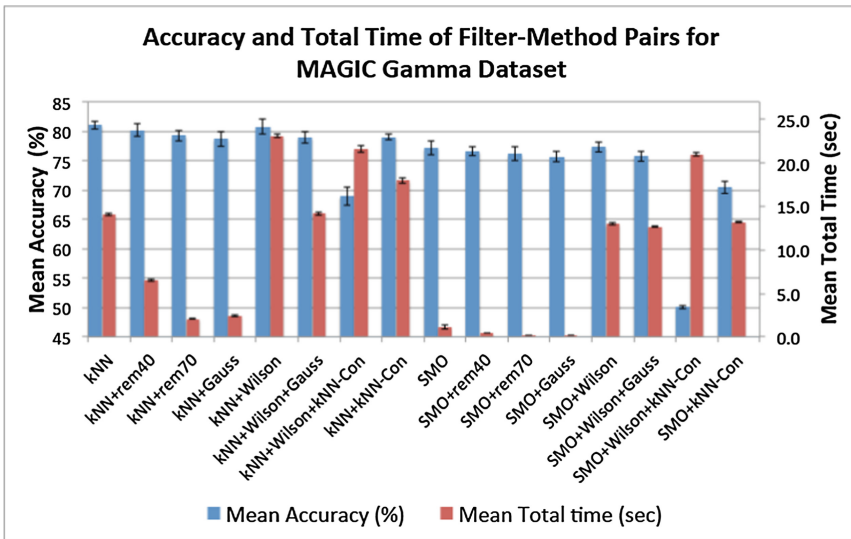


Fig. 3. Accuracy and total time of condensing algorithms for the MAGIC Gamma Telescope data.

if 70% of the data are discarded then k-NN runs about five times faster (and SMO about ten times faster) than when all the data is used for training. This is not too surprising since k-NN runs in $O(N^2)$ expected time and SMO is able to run faster in practice depending on the structure of the data.

4.3 The Condensing Classifiers

Experiments were done applying various training data condensation classifiers to reduce the size of the training dataset that was fed to both the SMO and

k-NN classifiers. The condensing classifiers tried in the first experiment were: (1) Wilson Editing, (2) Gaussian Condensation, (3) Wilson Editing + Gaussian Condensation, (4) Wilson Editing + k-NN Condensation, and (5) k-NN Condensation. In the second experiment, the classifiers tested were: (1) Gaussian Condensation + Gaussian Smoothing, (2) Gaussian Condensation + Wilson Editing, (3) Combined Gaussian Filters, (4) Gaussian Smoothing, (5) Gaussian Smoothing + Gaussian Condensation, and (6) Wilson Editing.

Figures 3–5 show the percent mean accuracy and mean total running times (in seconds) as well as the breakdown of the total time for all the five condensation classifiers in the first experiment, plus the results for blind random sampling obtained by discarding 40% and 70% of the training data. Figures 8–16 show the same variables for the six classifiers in the second set, as well as Gaussian Smoothing and Wilson Editing preceded by discarding 70% of the data. In the figures, ‘k-NN-Con’ indicates k-NN condensation, ‘Wilson’ denotes Wilson Editing, ‘Gauss’ stands for Gaussian Condensation, ‘GaussSmooth’ for Gaussian Smoothing, ‘GaussComb’ for the Combined Gaussian Filters, and ‘rem40’ and ‘rem70’/‘%’ are blind random removal of 40% and 70% of the data, respectively.

Experiment 1. Perusal of Figs. 3–5 reveals that none of the condensing methods improve the accuracy of the classifiers that do not use condensing. In fact, k-NN condensation performs the worst for all but two of the data sets – Song and Wine data – and for those two, other methods give poor results (e.g. see Fig. 4 for Wine data). With k-NN classification, random removal of 70% of the training set and Gaussian Condensing were the fastest methods for all the datasets (e.g. MAGIC Gamma data in Fig. 3) other than the Handwritten Digits (see Fig. 5). Similar relative behavior was observed with SMO, however, in all cases the running times with SMO were much smaller than those with k-NN.

In Figs. 3–5 the times plotted are *total* times: condensing time + training time + testing time. One of the main goals of this research is to compare the testing times of the various classifiers, since these reflect the speed of the classifier on all future data. However, if the condensing times and training times dominate the testing time, then the total times listed in the figures may hide the testing times. Therefore a breakdown of the individual times was also plotted for all the experiments. Note from Fig. 5 that the classifiers with the smallest total times are: SMO, SMO+rem40, SMO+rem70, and SMO+Gauss. Furthermore their accuracies are not significantly different. Therefore the classifiers with the fastest testing times would be preferred in this case.

Figure 6 shows the breakdown of condensing, training, testing, and total time for the Song data on a linear scale in seconds. This figure clearly shows how large the testing time for k-NN is compared to all other classifiers, thus making it difficult to compare the four classifiers of main interest. To zoom in on their performance the data from Fig. 6 are shown on a logarithmic scale in Fig. 7, where it can be clearly seen that SMO has shorter testing times with rem40, rem70, and Gaussian Condensing, but is slower when no filter is used (although total time is similar). All data sets show similar behaviour, and thus are not shown here.

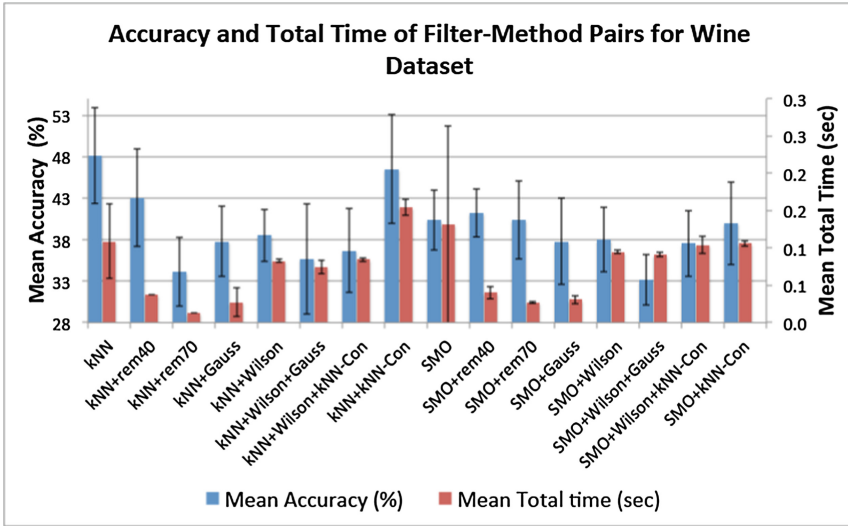


Fig. 4. Accuracy and total time of condensing algorithms for the Wine Quality data.

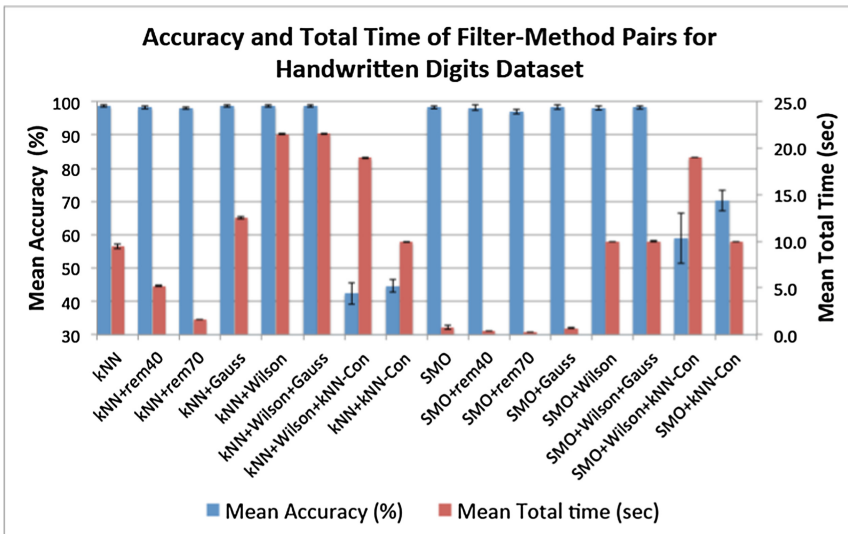


Fig. 5. Accuracy and total time of condensing algorithms for the Digits data.

Experiment 2. Figures 8–13 show that the classifiers tested give greatly varying results across data sets. Only the Letter Image and Wearable Computing data show very similar behaviour (Wearable Computing data shown in Fig. 8). In general, k-NN classification with Gaussian Smoothing and Condensing together, in both orders (but not with the combined algorithm), Gaussian Condensing and

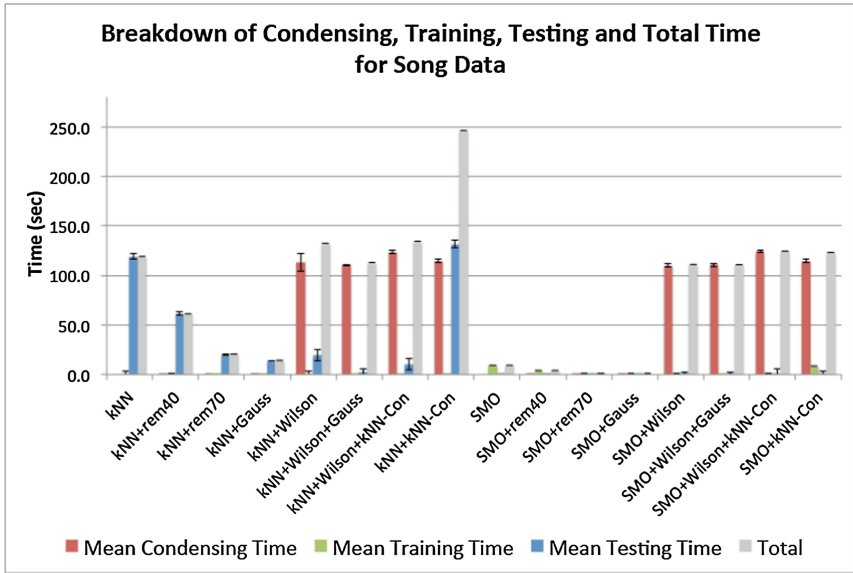


Fig. 6. Breakdown of condensing, training, testing, and total time for the Year Prediction Million Song data.

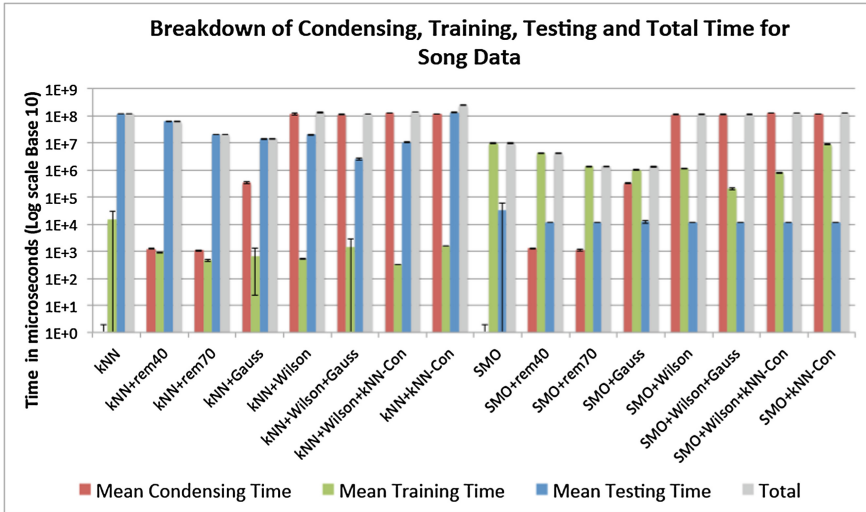


Fig. 7. The data of Fig. 8 on a logarithmic scale.

Wilson Editing, and random removal with Gaussian Smoothing or Wilson Editing are all fast, but give mixed results in terms of accuracy. In the Letter Image and Wearable Computing datasets, k-NN with Combined Gaussian Filters gives the greatest accuracy but is slightly slow, while Gaussian Smoothing and Con-

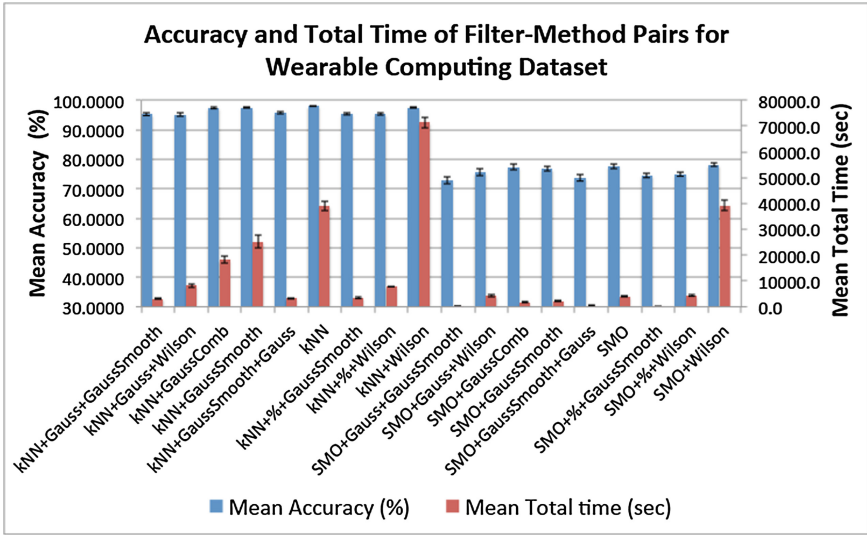


Fig. 8. Accuracy and total time of condensing algorithms for the Wearable Computing data.

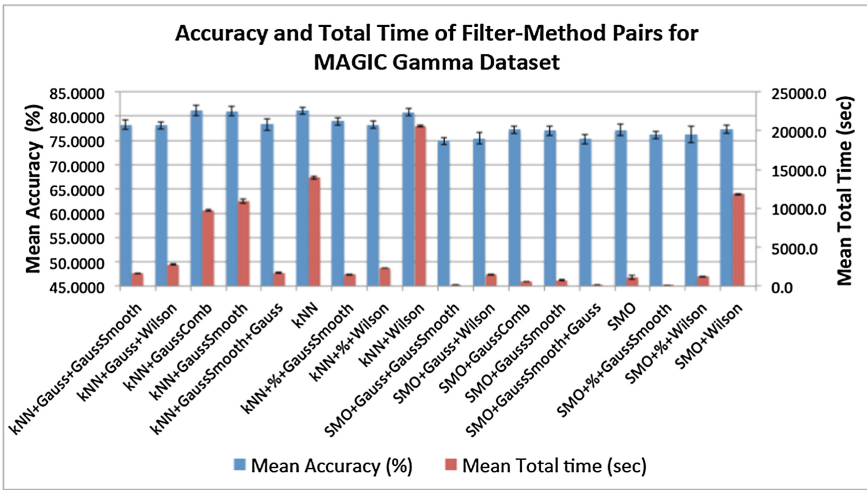


Fig. 9. Accuracy and total time of condensing algorithms for the MAGIC Gamma Telescope data.

condensing together in either order, and random removal with Gaussian Smoothing both give good accuracy (see Fig. 8). The MAGIC Gamma data show similar behaviour, although the latter three classifiers are instead better with SMO (see Fig. 9). On average, k-NN classification gives better accuracy than SMO classification for these data sets, as well as the Song data (See Fig. 10). The latter is

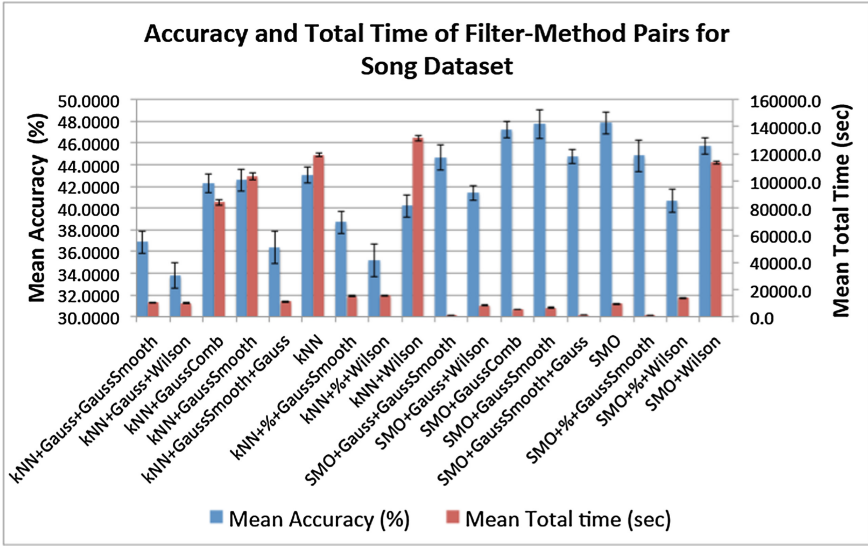


Fig. 10. Accuracy and total time of condensing algorithms for the Year Prediction Million Song data.

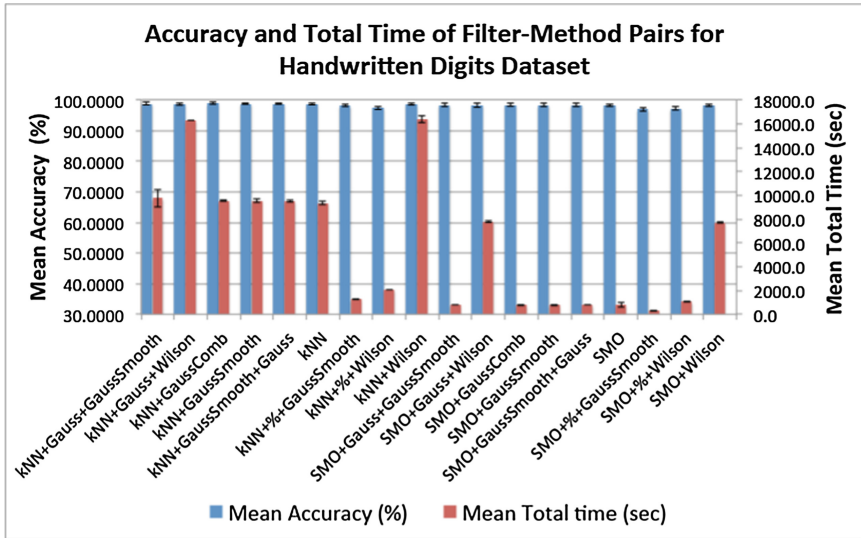


Fig. 11. Accuracy and total time of condensing algorithms for the Digits data.

also similar to MAGIC Gamma data since they both have the same three well performing classifiers, however in the Song data, SMO with Combined Gaussian Filters is overall the best method, SMO with Gaussian Condensation and Wilson Editing or random removal and Wilson Editing give poor accuracy, while

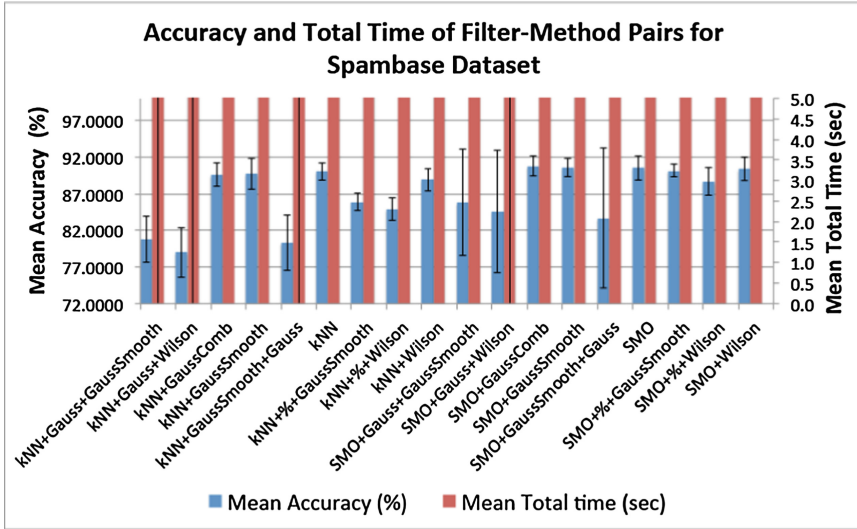


Fig. 12. Accuracy and total time of condensing algorithms for the Spambase data.

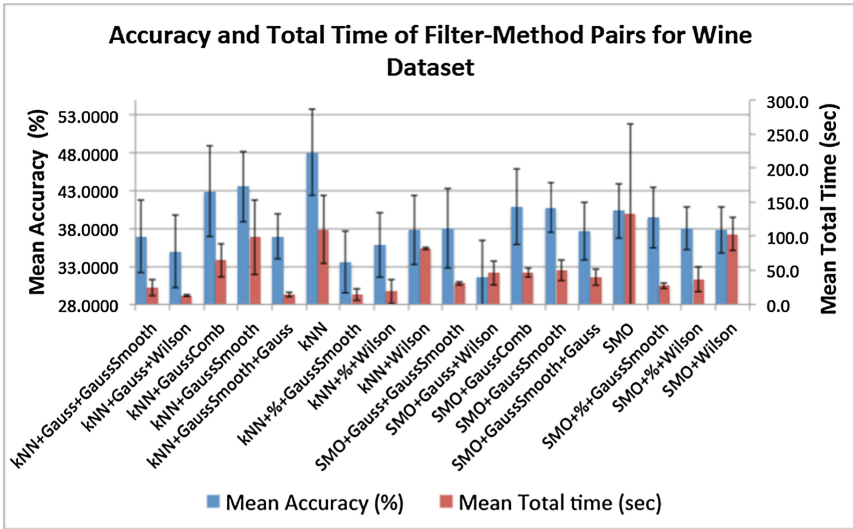


Fig. 13. Accuracy and total time of condensing algorithms for the Wine Quality data.

only Gaussian Condensing with Wilson Editing gives good accuracy and short running times out of all k-NN classifiers. The Handwritten Digit data gives generally excellent accuracy with all classifiers, but SMO with random removal and Gaussian Smoothing is the fastest (see Fig. 11). In the Spambase data (Fig. 12) SMO classification is much better than k-NN classification. Finally, the Wine

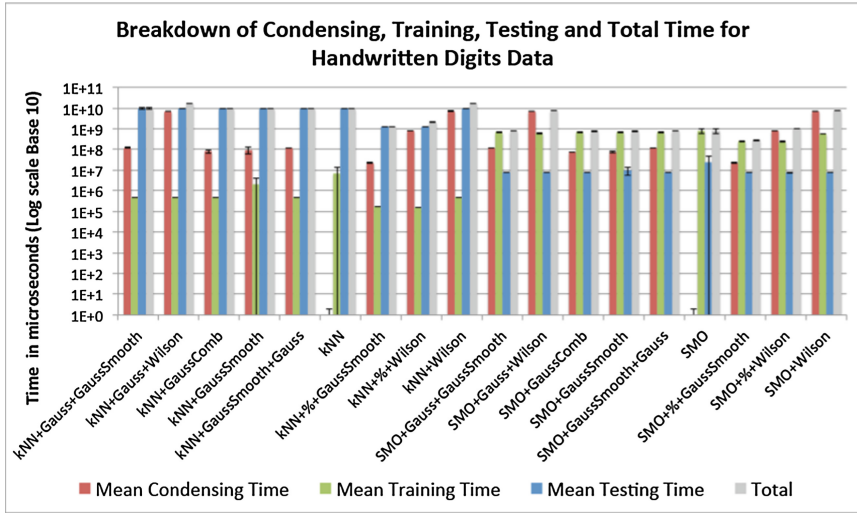


Fig. 14. Breakdown of condensing, training, testing, and total time on a logarithmic scale for the Digits data.

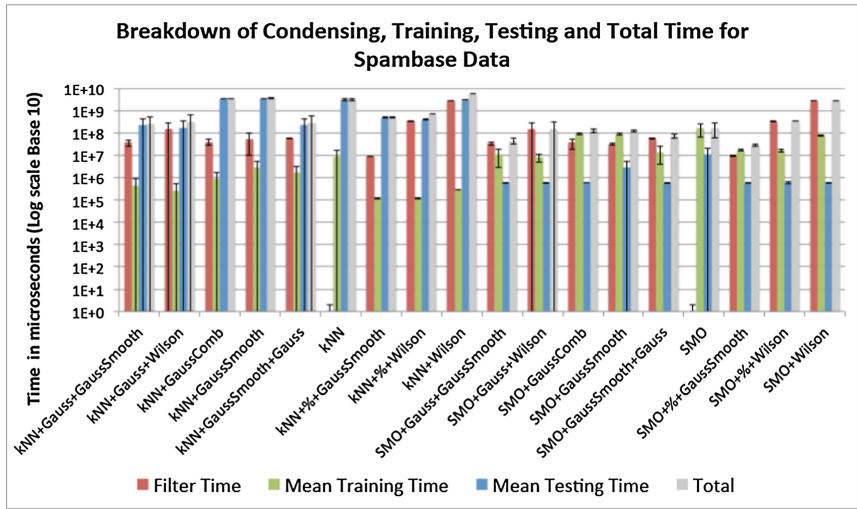


Fig. 15. Breakdown of condensing, training, testing, and total time on a logarithmic scale for the Spambase data.

data (Fig. 13) shows very little similarity with other data sets, and yields generally poor results for every classifier.

As in Experiment 1, a breakdown of the condensing, training, testing, and total time was also graphed (only the logarithmic versions are shown for the same reasons outlined above). The Handwritten Digits data (see Fig. 14) show

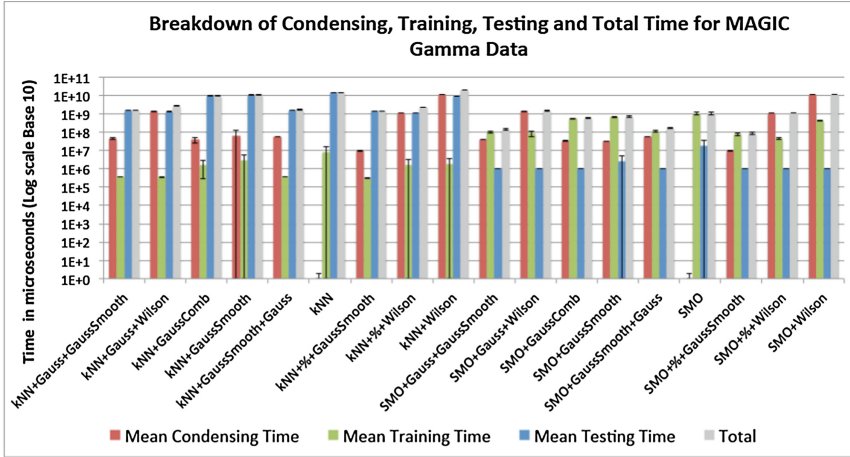


Fig. 16. Breakdown of condensing, training, and total time on a logarithmic scale for the MAGIC Gamma Telescope data.

random removal with Gaussian Smoothing as the classifier with the shortest combined testing and total time, which gives a good approximation of the best classifier in general. The Spambase data (see Fig. 15) also include Gaussian Condensation followed by Gaussian Smoothing as a top classifier. Finally, the top classifiers of the rest of the data sets comprise the last two classifiers along with Gaussian Smoothing and Gaussian Condensation (the opposite order as the second classifier mentioned), shown in Fig. 16 using the Magic GAMMA data as an example.

5 Conclusions

One of the main conclusions that can be made from the experiments reported here is that blind random sampling is surprisingly good and robust. For all the datasets, as much as 70% to 80% of the data may be discarded, without incurring any significant decrease in the classification accuracy. Furthermore, for six of the seven datasets, discarding 70% of the data at random in this way made k-NN run about five times faster, and SMO about ten times faster. Since this method is so simple and requires so little computation time we believe that it should play a role as a pre-processing step for speeding up SVMs.

Previous research has shown that SVMs perform better than k-NN. However, some of the comparisons have used synthetically generated datasets that do not resemble real world data. On the other hand, the results of the present study with seven real-world datasets tell a different story. SMO is significantly better in terms of accuracy only for the Song data, whereas k-NN does better for the Letter Images, Wearable Computing, and Magic Gamma datasets. For the other three datasets (Spam, Wine, and Handwritten Digits) there are no significant

differences between SMO and k-NN. However, SMO has the advantage of running faster than k-NN, which may be more important in some applications. In future research we hope to discover structural features of the data that predict when SMO is expected to outperform k-NN.

One of the goals of this research was to test how much Wilson editing and k-NN condensation improve the overall performance of classifiers in practice. It was found that for all seven datasets using Wilson editing as a pre-processing step to either SVM or k-NN, yielded no statistically significant improvement in accuracy, and k-NN condensation led to poorer accuracy. Furthermore, both Wilson editing and k-NN condensation incur a considerable additional cost in the filter (editing+condensing) time. However, if used after Gaussian Condensing or blind random sampling, it runs quickly, and can sometimes give good levels of accuracy.

Another main goal of this research project was to compare the new proposed methods for condensing and smoothing training data in $O(N)$ worst-case time: Gaussian Condensing/Smoothing, along with the algorithm for combining both Gaussian filters. This probabilistic method falls in the category of guided (or intelligent) random sampling and is almost as fast as blind random sampling. The results of this study show that Gaussian Smoothing used alone or together with Gaussian Condensing can give surprisingly good results, and is especially fast with SMO classification. The Combined Gaussian Filters also performed quite well in general. Finding ways to predict which combination of random sampling methods will do best depending on the dataset is an area of research we would also like to investigate further.

Acknowledgements. This research was supported by a grant from the Provost's Office of New York University Abu Dhabi in the United Arab Emirates. The authors are grateful to the University of California at Irvine for making available their large collection of data at the Machine Learning Repository.

References

1. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
2. Toussaint, G.T., Berzan, C.: Proximity-graph instance-based learning, support vector machines, and high dimensionality: an empirical comparison. In: Perner, P. (ed.) MLDM 2012. LNCS (LNAI), vol. 7376, pp. 222–236. Springer, Heidelberg (2012)
3. Devroye, L.: On the inequality of cover and hart in nearest neighbour discrimination. IEEE Trans. Pattern Anal. Mach. Intell. **3**, 75–78 (1981)
4. Bordes, A., Ertekin, S., Weston, J., Bottou, L.: Fast kernel classifiers with online and active learning. J. Mach. Learn. Res. **6**, 1579–1619 (2005)
5. Almeida, M.B., Braga, A.P., Braga, J.P.: SVM-KM: speeding SVMs learning with a priori cluster selection and k-means. In: Proceedings of the 6th Brazilian Symposium on Neural Networks, pp. 162–167 (2000)
6. Chen, J., Chen, C.: Speeding up SVM decisions based on mirror points. In: Proceedings of the 6th International Conference Pattern Recognition, vol. 2, pp. 869–872 (2002)

7. Panda, N., Chang, E.Y., Wu, G.: Concept boundary detection for speeding up SVMs. In: Proceedings of the 23 International Conference on Machine Learning, Pittsburgh (2006)
8. Wang, Y., Zhou, C.G., Huang, Y.X., Liang, Y.C., Yang, X.W.: A boundary method to speed up training support vector machines. In: Liu, G.R., et al. (eds.) Computational Methods, pp. 1209–1213. Springer, Netherlands (2006)
9. Chen, J., Liu, C.-L.: Fast multi-class sample reduction for speeding up support vector machines. In: Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing, Beijing, China, 18–21 September (2011)
10. Li, X., Cervantes, J., Yu, W.: Fast classification for large datasets via random selection clustering and support vector machines. *Intell. Data Anal.* **16**, 897–914 (2012)
11. Liu, X., Beltran, J.F., Mohanchandra, N., Toussaint, G.T.: On speeding up support vector machines: proximity graphs versus random sampling for pre-selection condensation. In: Proceedings of the International Conference Computer Science and Mathematics, Dubai, United Arab Emirates, 30–31 January, vol. 73, pp. 1037–1044 (2013)
12. Chen, J., Zhang, C., Xue, X., Liu, C.-H.: Fast instance selection for speeding up support vector machines. *Knowl. Based Syst.* **45**, 1–7 (2013)
13. Lee, Y.L., Mangasarian, O.L.: RSVM: reduced support vector machines. In: Proceedings of the First SIAM International Conference on Data Mining, 5–7 April (CD-ROM). SIAM, Chicago (2001)
14. Provost, F., Jensen, D., Oates, T.: Efficient progressive sampling. In: Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, USA (1999)
15. Ng, W.Q., Dash, M.: An evaluation of progressive sampling for imbalanced datasets. In: Sixth IEEE International Conference on Data Mining Workshops, Hong Kong, China (2006)
16. Portet, F., Gao, F., Hunter, J., Quiniou, R.: Reduction of large training set by guided progressive sampling: application to neonatal intensive care data. In: Proceedings of Intelligent Data Analysis in Biomedicine and Pharmacology, Amsterdam, pp. 43–44 (2007)
17. Kawulok, M., Nalepa, J.: Support vector machines training data selection using a genetic algorithm. In: Gimel'farb, G.L., et al. (eds.) SSPR & SPR 2012. LNCS, vol. 7626, pp. 557–565. Springer, Heidelberg (2012)
18. Hart, P.E.: The condensed nearest neighbour rule. *IEEE Trans. Inf. Theory* **14**, 515–516 (1968)
19. Sriperumbudur, B.K., Lanckriet, G.: Nearest neighbour prototyping for sparse and scalable support vector machines. Technical report No. CAL-2007-02, University of California San Diego (2007)
20. Toussaint, G.T.: Geometric proximity graphs for improving nearest neighbour methods in instance-based learning and data mining. *Int. J. Comput. Geom. Appl.* **15**, 101–150 (2005)
21. Bache, K., Lichman, M.: UCI Machine Learning Repository (2013). <http://archive.ics.uci.edu/ml>
22. Wilson, D.L.: Asymptotic properties of nearest neighbour rules using edited-data. *IEEE Trans. Syst. Man Cybern.* **2**, 408–421 (1973)
23. Witten, I., Frank, E.: WEKA: machine learning algorithms in java. In: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, pp. 265–320. MorganKaufmann (2000)

24. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Scholkopf, B., Burges, C., Smola, A. (eds.) *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge (1998)
25. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Comput.* **13**, 637–649 (2001)
26. Toussaint, G.T.: Bibliography on estimation of misclassification. *IEEE Trans. Inf. Theory* **20**, 472–479 (1974)