An Analysis of Feature Selection Techniques

Matthew Shardlow

Abstract

In this paper several feature selection methods are explored. These are analysed to see what effect they have on the accuracy of a simple svm. Several filter and wrapper techniques are investigated. Hybrid methods which use combinations of filter and wrapper techniques are also investigated. Many filter methods are found to give no increase in accuracy for the classifier. The most effective method is found to be a hybrid method which is called 'Ranked Forward Search'. This gives an increase in accuracy for the classifier when using only a small subset of the possible features.

1 Introduction

Data in the modern world is becoming more and more high dimensional. One especially prevalent example is the world of bioinformatics [7]. Here, machine learning is employed to build classifiers which are able to infer classifications about patients' health. 20 years ago the data for this would have been simple - environmental factors such as smoking, age, weight etc. would be used to classify a patient's risk. However in the modern genomic era, it is possible to sequence genomes (in entirety or in part) on a patient-by-patient basis. The result is that the data available to make inferences over patients increases drastically to thousands of features per data instance.

The challenge that this presents to the machine learner is what to do with all this data? The obvious approach is to try to reduce the number of dimensions in which we are computing. This is done by feature selection [3, 5]. It is very likely that different features will carry different amounts of information. Some will be more useful than others. In feature selection we have several aims:

- To reduce the size of the problem - reducing compute time and space required to run our algorithms.
- To improve classifiers. Firstly by removing noisy or irrelevant features. Secondly by reducing the likelihood of overfitting to noisy data.
- To identify which features may be relevant to a specific problem. E.G. to demonstrate which gene expressions are relevant in a certain disease.

So our aim is: to reduce the number of features whilst maintaining high performance by discarding those least useful. To discard the least useful features whilst keeping the most useful ones, we must have some measure of what is 'useful' to us. This can be done on a feature by feature basis looking at how useful each feature is on it's own and then selecting some arbitrary amount of best features. However, features work together [3]. Ranking features on their individual usefulness does not take this into account, and so more complex methods may be required. We can see this in figure 1 where we observe that as more features are included, the test error of the classification scheme varies. Whilst there is an initial minima at around 10 features, there is also another lower minima at around 100 features. Between and around this the inclusion of more features often appears to reduce the efficacy of the classifier. This shows that it is the combination of features that is important, not necessarily the individual features which are included.

2 Methods

We wish to select a set of features which will in some way give us more information about our data than any other combination. Detailed below are several ways which have been explored to do this.

2.1 Filter Methods

Filter methods apply some ranking over features. The ranking denotes how 'useful' each feature is likely to be for classification. Once this ranking has been computed, a feature set composing of the best N features is created. For each of the filter methods described below N was set to 10, 30 and 100. These values were chosen to give an indication of the importance of feature set size.

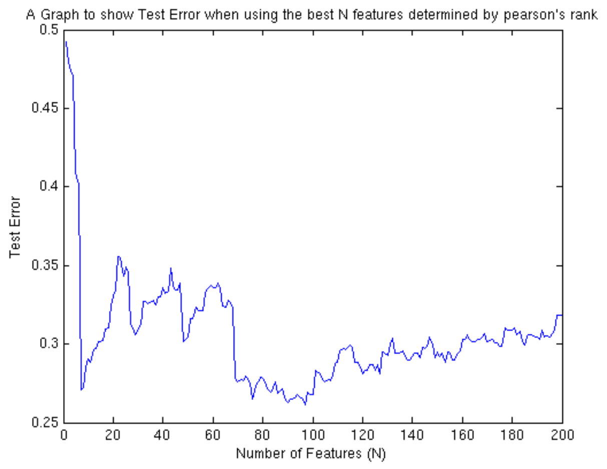


Figure 1: This graph was produced using a 1-nn classifier. The features were ranked using pearson's correlation. The N best features were incrementally used for classification. The test error (y-axis) is shown to vary with the number of features used (x-axis). 2 minima are observed. one at around 10 features and another at around 100 features.

2.1.1 Pearson's correlation coefficient.

This method looks at how well correlated two sets of data are [2]. That is, how much does a variation in one data set affect the variation in another.

We might want to use this as if there is a high correlation between one feature and the class of the data we are using then this will most likely be good at separating the data and will be useful for classification purposes.

2.1.2 Mutual Information.

This is an indicator of the shared information between two variables and "measures how much knowing one of these variables reduces uncertainty about the other" [1]. This works similar to pearson's coefficient and is included to see how well it performs in comparison. Again, if there is a lot of shared information between a feature and the class of our labelled data then this is a good indicator that this feature is important and useful in distinguishing members of one class from another.

2.1.3 Relief.

This method (first proposed in [4]) works through a data set looking at the separation capabilities of randomly selected instances. It does this by selecting for each instance both the nearest same-class instance and the nearest opposite-class instance. These are then used to calculate a weighting for each feature which is iteratively updated with each stochastically chosen data point.

This method can be highly useful when there is a large amount of data. Time complexity is not an issue as a constant number of trials is performed. This means that the relief algorithm may complete quicker than other methods which require all the data to be taken into account.

2.1.4 Ensemble with data permutation.

As described in [6], the concept of ensemble techniques may be applied to feature selection. In a typical ensemble method, many weak classifiers are created which are combined together to give a result much better than any single classifier could achieve on it's own. This approach may be applied to feature selection by producing many 'weak' rankings which can then be combined together to give a much stronger and more accurate ranking.

This was applied by creating a bootstrap of the testing data and performing a ranking on this subset of the data. This process was repeated 10 times with a new random bootstrap each time. After this the rankings were combined as proposed in [6]. This was done by adding all the rankings for each data point together to create a score. The order of these scores was then used as a new ranking over the features. [6] proposes a method for weighting the initial rankings to give more significance to high ranked features when creating the scores, however that was not pursued in this paper.

This was performed for each ranking method, selecting the best 10, 30 and 100 features each time in order to compare results.

2.1.5 Ensemble of methods.

The final filter method explored is mentioned, but not explored in [6]. Rather than permuting the data to create several different rankings, we instead change the method being used to rank the data. The same data (i.e. the training data) is used for each ranking method and the results of each ranking are combined as explained in the section above.

It is hoped that different ranking methods will pick out different important qualities in the data and that the combination of these will perform better than any one ranking method on it's own.

This was applied by performing ranking with the pearson's coefficient, mutual information and the relief algorithm. These 3 results were then combined to give an overall ranking.

2.2 Wrapper Methods

Wrapper methods are so called because they wrap a classifier up in a feature selection algorithm [5]. Typically: a set of features is chosen; the efficacy of this set is determined; some perturbation is made to change the original set and the efficacy of the new set is evaluated. The problem with this approach is that feature space is vast and looking at every possible combination would take a large amount of time and computation. This means that some heuristic search methods must be developed to find optimum sets of features.

2.2.1 Greedy Forward Search

This method is a greedy finite difference calculation as explained in [3]. It works by making changes to the set of features and only keeping the new set if there is an increase in accuracy. Greedy Forward Search works by starting with just one feature and incrementally adding in all the other features. As each feature is added in, the classifier is evaluated with the feature set and the new feature is only kept if there is a notable increase in accuracy.

This is a greedy solution and may not find the absolute optimum feature set, however by looking at which features cause an increase in accuracy it will pick out useful features to the classification scheme. Also, because the accuracy of the classifier is evaluated with all the features in a set, this method will pick out features which work well together for classification. Features are not assumed to be independent and so advantages may be gained from looking at their combined effect.

2.2.2 Exhaustive Search

In this method, we explore a brute force approach as mentioned in [3]. This means looking at every possible combination of features to find which one gives the best result. It is of course only possible to do this with a small number of features and so some simplification of this problem must be done.

To quantify the scale of attempting exhaustive search, lets look at the numbers. Say we wished to look at every single possible combination of a dataset with 325 features (see section 3 for specifications of data used). There must exist one set for every possible combination of features, each feature has 2 states - it's either included or not, so there are $2^{325} = 6.8 * 10^{97}$ different possibilities. This is clearly too many possibilities to consider. So let's simplify, we will only look at every group of 10 features out of the total 325. This gives us $325C10 = 3.15 * 10^{18}$ possibilities. Although this problem is easily parallelisable, we don't have the resources to do that, so lets say we want to reduce the size of the problem further. Say we somehow chose a pool of 50 ideal candidate features from our original 325. From these 50, we want to identify 10 to use together in our classification scheme. This gives us $50C10 = 10,272,278,170$ possibilities.

This is certainly a smaller number than before, however it is still clearly much larger than we can hope to compute with. 10 billion trials is a lot. This shows that exhaustive search, if it is going to work needs to be highly constrained and that constraining must be done in an intelligent way. A full brute force approach was not explored for any data set in this paper, however the exhaustive methods employed are discussed in the following section.

2.3 Hybrid Methods

So far, we have described methods for ranking the features of a data set in order and methods for searching the space containing all possible subsets of features. In this section we present two ways of using these together in order to gain a better classification scheme.

2.3.1 Ranked Forward Search

Here we introduce a ranking to the features and then perform a greedy forward search over that ranking. It is hoped that only including the variables which increase the accuracy will lead to a better result than thresholding (as in the filter approach).

For this to work we are assuming that there will be some co-dependence between features which are highly ranked. The ranking methods we have used so far have all assumed that features are independent and so the ranking of each feature is not affected by it's correlation with other features. Evaluating the ranked set of features with a greedy algorithm should discover some correlatory behaviour between the highly ranked features.

2.3.2 Refined Exhaustive Search

As previously suggested it is an incomprehensibly massive task to explore the entire feature space of any problem big enough for feature selection to be worthwhile. That is to say that if a problem had a low number of features then it would be relatively easy to look at every possible combination of those and determine which combination was best. But if a problem has a low number of features, it will possibly be obvious which to choose or more likely be the case that all the features are relevant and necessary to classification. Ironically, it is only when the feature space gets large that we want to start employing expensive methods such as a brute force search.

The aim of this method is to constrain the feature space to some useful subset over which an exhaustive search may be performed. To do this, we first rank the variables according to some filter method criterion. We then select the best n variables to be our new feature space. we search in this feature space for every subset of r variables. We can work out the number of possible combinations we will compute as:

$$nCr = \frac{n!}{r!(n-r)!}$$

Using this we can select sensible values for n and r. it should be noted that this does not limit n and r to small values. The distribution of the values is symmetrical so for example $20C5 = 20C15 = 15504$ This means that there are as many combination of 5 variables as there are of 15 variables. So as long as n and r and kept close to each other we can search over relatively high values for r. Of course, if these values are close then there will not be many combinations searched, so some distance should be maintained. The appropriate amount may be discovered experimentally.

3 Experiments

3.1 Data

The data set chosen for this experiment was the lung cancer examples. This contained 73 examples each with 325 features. This data was selected due to the large number of features. The features are discrete, but this does not affect our classification scheme. This data was kept constant to facilitate comparison of results.

3.2 Classifier

The classifier used in all experiments is a support vector machine with a linear kernel. This was chosen as it is simple to use and quick to run. Whilst this will not necessarily give the best classification results, it was considered acceptable to use as the focus of the research is the analysis of the feature selection techniques employed.

3.3 Train / Test Protocol

N-fold cross validation was used for training and testing the data. The number of trials was initially set to 10 however this took a long time, so more computationally intensive experiments (such as exhaustive search) used $N = 5$. This was found to increase the speed of computation and still gave valid results. Mean accuracies and standard deviations are reported where relevant.

3.4 Results

The efficacy of the selected feature subset was measured by creating a classifier which only used the selected features. Many methods have been compared and (unless stated) use the same data and classifier, so the only difference is the feature subset selected. Each method chose different features and different numbers of features were chosen. Comparisons are displayed for the different types of methods employed below. Explanations of these methods are give in section 2.

For comparison a control experiment was performed where a classifier using all 325 potential features was trained and tested. The accuracy of this classifier is displayed with the accuracies for the chosen subsets of features to make it easy to display the effect of feature selection on classifier accuracy.

3.4.1 Filter Methods

Several filter methods were employed to rank the variables. Three trials were performed with each ranking method, selecting the best 10, 30 and 100 variables in each case. Results are presented in figure 2.

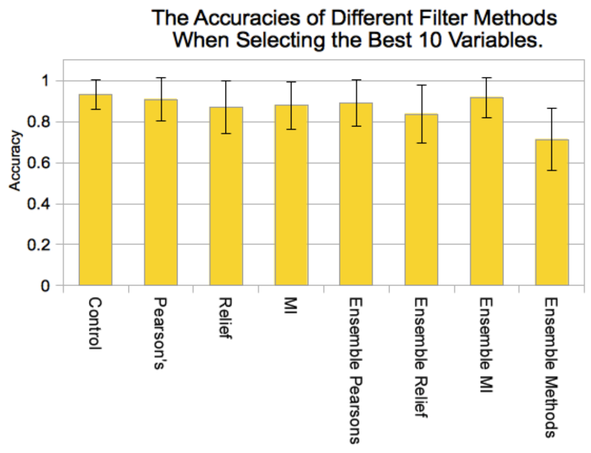
The ensemble methods appear to perform better than the standard methods, however no methods appear to select feature subsets which give higher accuracy than the control (first column in each chart). The control accuracy was 0.9329 with a standard deviation of 0.0719, this was equalled by the "ensemble pearson's" method when selecting 30 features. This fulfils the dimensionality reduction aim as the same accuracy as for all features was achieved with 30 features.

3.4.2 Wrapper and Hybrid Methods

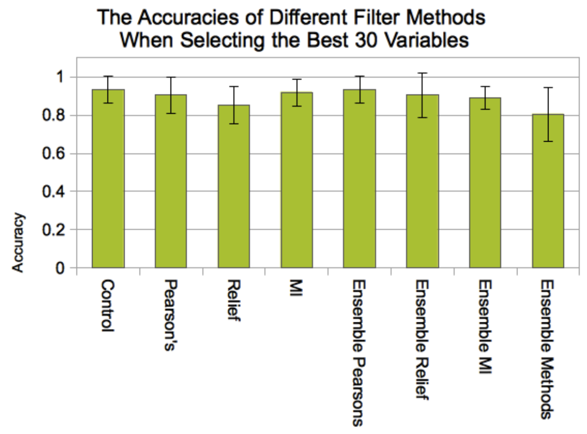
Several wrapper and hybrid methods as described in section 2 were employed. A naive forward search was developed and a brute force exhaustive wrapper method was developed. The exhaustive search was too computationally expensive to run in any sensible time, and so there are no results for this.

These wrapper methods were then combined with the pearson's coefficient filter method. Pearson's coefficient was chosen as it is quick and is shown in fig 2 to be an effective feature ranking method. The exhaustive search selected the top 20 features and then looked at every possible combination of 10 features from this. These values were set by experimentation with how long each trial took and the number of trials to be performed. The values used resulted in a search time of about 20 minutes, which was deemed to be acceptable.

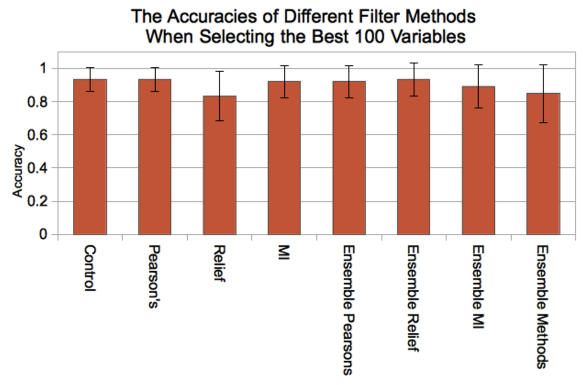
The results of these experiments are shown in table 1. We can see that each method employed provides a marked improvement over the control (which used



(a) 10 features



(b) 30 features



(c) 100 features

Figure 2: The accuracies of different methods when selecting the top 10, 30 and 100 features. In each chart, the first column is the control experiment and uses all 325 features.

no feature selection). This improvement is evident on two fronts. Firstly, The accuracy of the classifier is improved and secondly, the number of features used is reduced. The ranked forward search appears to be the best method attempted as this has a high accuracy, with a low standard deviation, implying that the high accuracy is reliable. This method uses 33 features out of a possible 325, this is a dimensionality reduction by a factor of 10.

4 Discussion

Feature selection has certain advantages and disadvantages which have been clearly shown in this project. It is not necessarily a means to improving the accuracy rate of a classifier as shown by the filter methods. In fact in all cases reducing the number of features too far will always result in a decrease in accuracy. However, it has been shown that with some techniques and especially with the high dimensional data used in this paper, filter selection can help to improve the accuracy of a classifier. Further to this it has been shown that in all cases feature selection can select the features which are most important to classification and so reduce the total number of dimensions necessary for classification.

We have shown that wrapper methods (such as the Greedy Forward Search) yield subsets with higher accuracy than filter methods. We have also shown that hybrid methods (such as Refined Forward Search) yield subsets with even higher accuracy.

The filter methods which were employed gave varying results. No methods were found to give a higher accuracy than the control experiment - which used all 325 features. This does not necessarily mean that the method was useless. There may be some situations where reducing the dimensionality is a key factor and some loss of accuracy may be sustained in order to do this. The filter methods reduced the number of features to 10, 30 and 100 features, which is a reduction by a factor of 30, 10 and 3 respectively. These different thresholds were found to be similarly accurate.

There are two potential reasons for the drop in accuracy. Firstly, the filter methods explored look at the correlation between each feature and the class label. Features are evaluated independently and so no information about features which influence each other may be captured. Secondly, The threshold value was set naively. The threshold is a parameter of the feature selection algorithm which can be fine tuned for each problem.

The forward search method appeared to marginally improve the accuracy of the classifier, whilst reducing the number of features to 30. The reason these 30 features worked better than the original 325 is be-

Method	Accuracy	Standard Deviation	Number of Features Used
Control	0.9329	0.0719	325
Naive Forward Search	0.9446	0.0566	30
Ranked Forward Search	0.9833	0.0373	33
Refined Exhaustive Search	0.9667	0.0745	10

Table 1: The results of the experiments with wrapper and hybrid methods.

cause the extra 295 features which were not selected did not improve the classification scheme. This is because some features are not effective at splitting the data. If we include features which do not split the data well then our classifier will have to work in a higher dimensional space, with less examples per dimension. The forward search algorithm only kept features which improved classification rate. This meant that the final set of features worked well together to split the data.

This method was improved by first ranking the variables. This led to an increase in classification accuracy with 33 features selected. This helped as it meant that features which were independently most correlated to the data labels were evaluated first. The search started with the best possible candidates and so was more likely to find a set of features which were also individually more effective at splitting the data.

The exhaustive search method showed that sometimes a brute force approach can be effective. 10 features were selected, with an accuracy which was an improvement on the control. Whilst this method is naive, brute-force and time-consuming the results show that it is effective. No other method which selected only 10 features improved the accuracy of the classifier and no other method with a higher accuracy had as few as 10 features.

The method showed that a very small subset of features could be found which separates the data well. This was only found by evaluating all the possibilities and selecting the most accurate. Whilst there are some heuristic methods for doing this such as genetic algorithms, this exhaustive approach means that all subsets in a given specification were evaluated. The constraints placed on the problem meant that only a useful subspace was exhaustively searched. This helped to reduce the time the search took making it possible. The parameters of this search were fixed, and it may be possible to improve this algorithm further by experimenting with these parameters.

This algorithm has some limitations for larger feature sets. Say we wish to select 30 features. If we select a space of 40 features we have to evaluate $40C30 = 847,660,528$ potential possibilities. The result of this is only to eliminate the 10 least useful features, which means we are unlikely to gain much from this approach anyway. This is probably best used in problems where there are only a small number of features to begin with and we only want to select a few important features from these. However, feature selection may not always be appropriate in the case with smaller feature sets, so the usefulness of this technique is questionable.

There are many ways that the techniques in this report could be expanded. One key avenue for future work would be the exploration of the ranked forward search. Different ranking methods could be applied and a deeper investigation into the nature of the feature selection over the ranked variables could be performed. This would give a deeper understanding of the method and why it appears to work well. The algorithm could also be modified to select only a certain number of features as currently every 'useful feature' is included in the final feature set.

5 Conclusion

Many problems currently exist where the data encountered has many features, for example 'omics' data in bioinformatics. This makes it difficult to process and comprehend as many features are often not relevant. Feature selection may be employed to find which features will be useful so as computation may be focussed on these. The desired effect of this is to speed up algorithms and also to make them more effective by only focussing on the relevant features in the data.

Several feature selection techniques have been evaluated to determine their effectiveness for reducing the number of dimensions in a dataset and for improving the accuracy of a classifier which uses only these features. This paper has found Ranked Forward Search to be a highly effective method for feature selection which has both increased accuracy and reduced the number of dimensions for the dataset used. Other methods have been evaluated but found not to be as effective. An exhaustive search was trialed experimentally, it was found to be very effective for reducing the number of dimensions, and marginally improved the accuracy of the classifier being used.

References

[1] Mutual information. `http://en.wikipedia.org/wiki/Mutual_information`.

[2] Pearson product-moment correlation coefficient. `http://en.wikipedia.org/wiki/Pearson_product_moment_correlation_coefficient`.

[3] Guyon and Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 2003.

[4] L. A. R. K Kira. A practical approach to feature selection. *ML92 Proceedings of the ninth international workshop on Machine learning*, 1992.

[5] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 1997.

[6] Y. Saeys, T. Abeel, and Y. V. de Peer. Robust feature selection using ensemble feature selection techniques. *MACHINE LEARNING AND KNOWLEDGE DISCOVERY IN DATABASES*, 2008.

[7] Tarczy and P. Hornoch. Bioinformatics challenges and opportunities. *Medical Informatics*, 2005.