Chapter 6

Feature Selection

Tony Bellotti, Ilia Nouretdinov, Meng Yang, and Alex Gammerman

In this chapter, we consider the implementation of feature selection approaches within the Conformal Predictor framework. We begin with a review of feature selection, and then consider several approaches to implementation. Firstly, using existing feature selection methods within conformal predictors, which raise some computational issues. Secondly, using techniques specifically designed for conformal predictors: (1) the strangeness minimisation feature selection (SMFS) method and (2) the average confidence maximisation (ACM) method. SMFS minimises the overall non-conformity values of a sequence of examples, whilst ACM maximises the average confidence output by the conformal predictor using a subset of features.

6.1 Introduction

Feature selection is the process of selecting a subset of features from a given space of features with the intention of meeting one or more of the following goals.

1. Choose the feature subset that maximises the performance of a learning algorithm.
2. Reduce the requirement for computer storage and/or computational time to classify data without significantly reducing the performance of a learning algorithm on a learning problem.
3. Detect a subset of features that are related to the natural problem being studied.

Feature selection is of most value when only a small number of significantly different features are expected to be relevant. Sometimes, reduction of features can improve the quality of prediction and even be a necessary, embedded, step of the prediction algorithm. Such algorithms, related to the first goal, are discussed in Chapter 11 (Biomedical Applications: Diagnostic and Prognostic) of this book. In this chapter, we mostly consider the second goal as prior: selection of a feature subset either of a fixed size or of a size as small as possible without an essential decrease of performance. The third goal is partially taken into account in the structure of the method.

Some example application areas for which feature selection has proved valuable are gene expression analysis and document classification. In the first example, we may want to model a disease or biological effect based on data containing thousands of gene expression measurements, of which we may expect less than 100 to be relevant. In the second, classification is often conducted on lists of counts of words in a document. Clearly, many words will not be relevant e.g. 'the' and 'a'. These will need to be filtered out. The goals listed above can be in conflict, since the feature subset that gives optimal performance is not necessarily the smallest that still gives good performance, or may not be computationally efficient. The decision regarding which is the most important goal depends on the outcome required by the user of the learning algorithm. Sometimes a small set of most important features is sought to facilitate better understanding or visualisation of the problem. This may direct further research within the problem domain. This is often the case in gene expression analysis where the particular set of genes selected through analysis is important to direct further biological research. Gene expression analysis is also an example where the third goal is actually a biological task. On the other hand, if the user is primarily interested in achieving the best learning performance then the first goal is more important [158].

For most learning problems there is an implicit feature selection process that is undertaken by researchers. For example, for document classification, researchers will choose features that are expected to be pertinent to the problem, such as the language and words in documents or the structure of the document. Researchers would not suppose the source of the paper or the kind of ink the documents are printed with as useful indicators of the document class, so these would not be included as features in the learning problem. However, in this work, we are interested in the task of explicit automated feature selection. Both [38] and [158] provide reviews of feature selection methods. Feature selection remains an open area of research within the machine learning community. Since the first review of these methods in [38], there have been many developments. Not least, is the fact that the number of features in a typical machine learning problem has leapt from tens to tens or hundreds of thousands. Hence, the problem of feature selection has become acute.

6.1.1 Feature selection methods

Typically a feature selection algorithm will use a given training dataset in order to make a decision about which features to select. There are three general methods for feature selection: filters, wrappers and embedded feature selection. These are discussed in the following sections.

Filters

The filter method employs a feature ranking function to choose the best features. The ranking function gives a relevance score based on a sequence of examples. Intuitively, the more relevant the feature, the higher its rank. Either a fixed number of features with the highest rank are selected, or a variable number of features above a preset threshold value of the rank are selected.

When the outcome variable is binary and the features are numeric, typical ranking functions are the signal-to-noise ratio (SNR) and the Fisher discriminant ratio (FDR) given for each feature as:

$$\text{SNR} = \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2} \text{ and } \text{FDR} = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$

where μ_i and σ_i are the sample mean and standard deviation, respectively, of the feature for each class label i.

Figure 6.1 illustrates the use of this approach. For small numbers of features (say, 5), performance is poor, but as the number of selected features increases, the performance improves. However, the performance peaks at about 600 features. When more features are included, the performance gets worse. If all features are included then performance deteriorates markedly. This demonstrates that including too many irrelevant features can actually worsen the performance of a learning algorithm, and hence shows the need for feature selection.

Other ranking functions can be constructed based on the t and χ^2 statistics. If the features are categorical, then criteria related to information gain can be used. If the learning problem involves a multi-class outcome variable, then FDR can be generalized or the ANOVA (ANalysis Of VAriance) test can be used. If a linear classifier is used, the magnitude of weights can also be used as ranks for filter feature selection [28]. Recursive feature elimination (RFE) is an iterative method using this approach and has been shown to give optimal results when applied to microarray data [159, Figure 5].

Filter methods have been successful in a number of problem domains and are very efficient. Nevertheless, the disadvantage of this method is that each feature is assessed independently. Correlations between features are not taken into consideration. It is possible that two features are individually useless in classification, but taken together become useful [158, Section 3.3].

Wrappers

Wrapper methods are general-purpose algorithms that search the space of feature subsets, testing performance of each subset using a learning algorithm. The

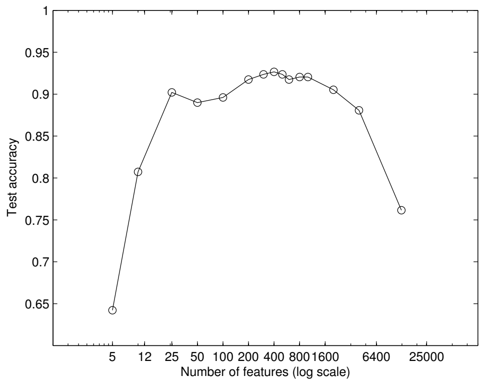

Figure 6.1: Feature selection: Accuracy on an independent test data set against the number of features selected using SNR (log width). Classification of childhood leukaemia using a naive Bayes classifier, based on 12,600 gene expression measures. Data source: [433].

feature subset that gives the best performance is selected for final use. Clearly, if there are m features in total, then there are 2^m possible subsets to search. Such an exhaustive search is impractical, so most wrapper algorithms incorporate a heuristic to narrow the search space. For example, one simple method is *backward elimination* that starts the search with the full set of features and eliminates them one at a time using some form of feature scoring, until an optimal subset is found. This search is linear on the number of features.

Wrapper methods have a long history in statistical modelling, in the form of *stepwise variable selection* [108]. This approach involves either forward selection, adding features one at a time or backward selection, removing features one at a time, until some criterion is reached. Additionally, a bidirectional selection method is available that involves adding *or* removing a feature at each step. A common wrapper criterion is to use the Akaike information criterion (AIC) of model fit. The application of wrappers in machine learning is relatively recent [38, Section 2.5].

The disadvantage of the wrapper technique is that it tends to be computationally intensive and the use of a heuristic to refine the search space can be ad hoc. There has been much research into improving the performance of wrappers. For example, [423] proposed a principled approach for optimal feature selection specifically for Support Vector Machines (SVM), based on the goal of reducing risk of generalization error.

Embedded Feature Selection

Some learning algorithms include an embedded feature selection method. Selecting features is then an implicit part of the learning process. This is the case, for example, with decision tree learners such as CART [53] that use an information measure or loss function to choose the best features. Other learning algorithms have been developed with embedded feature selection in mind. For example, Littlestone's WINNOW algorithm is an adaptation of the Perceptron algorithm that uses multiplicative weight updates instead of additive. This has the effect of rapidly degrading the weights on irrelevant features quickly, leaving only a relatively few features with non-zero weights [38, Section 2.6].

6.1.2 Issues in feature selection

We have already discussed computational issues with wrapper methods and the need for multi-class feature selection methods. However, there are three key issues for feature selection: the problem of selection bias - which is about the correct use of feature selection, false discovery rates and questions about relevance and redundancy. We discuss each of these issues below.

Selection Bias

It is important that feature selection is applied correctly. Some published research, using feature selection for microarray classification, has been criticized for using the entire dataset for feature selection prior to dividing the data into training and test sets. Presumably, the researchers supposed that the involvement of the test data in the derivation of the feature selection would not have a great impact on their results. However, [6] have shown that a significant *selection bias* does occur as a consequence of this approach. For two published datasets, for which no errors are reported, errors emerge when the feature subset is constructed on the training set alone. They conclude that “the test set must play no role in the feature-selection process for an unbiased estimate to be obtained”. They recommend that feature selection is only conducted on the training dataset. Or, if that is not possible, perhaps due to a small sample size, then a procedure involving cross-validation should be used.

False Discovery Rate

When selecting relevant features from a large pool of possible features based on sample data, it may be that some of those features are only *apparently* relevant, in the sense that their relevance in the sample is only the result of chance. These are known as *false discoveries*. The larger the pool of features to choose from, the more likely false discoveries will be. In order for the selected feature subset to be relevant to the population and more importantly, future examples, it is important to reduce the number of false discoveries. We define the *False Discovery Rate* (FDR) as the percentage of false discoveries amongst the features selected [33]. Controlling FDR is an important area of feature selection research, for problems with large feature spaces such as analysis of bioinformatics data [321].

Using the t-statistic filter, we can readily compute an expected FDR based on the p-value. If we set a significance level α as a threshold and select only features with a p-value less than α, then we can expect that the FDR will be approximately $\alpha/2$, assuming distribution of p-values is approximately uniform. Even for low values of α, we can expect a large number of features to be false discoveries. For example, if we have 10,000 features, $\alpha = 0.01$ will still give an expected 50 false discoveries [392].

Significance analysis of microarrays (SAM) is a popular method developed to overcome this problem. It was designed with microarrays in mind, but could be used with other problem domains. Feature selection is conducted using a variation of the SNR for different permutations of the data. The number of false discoveries is then estimated as the average number of falsely significant features corresponding to each permutation based on a user-defined threshold of significance. This then yields an estimate of FDR. As the threshold decreases, the number of significant features discovered increases, but at the cost of increasing FDR. Experiments show that SAM works well in controlling FDR compared to conventional methods [392].

Relevance and Redundancy

Feature selection was first considered as a search for the most *relevant* features where, broadly, a relevant feature is one that provides some information about the target label [38]. In contrast, [158] takes a more pragmatic approach to feature selection, considering the goal of feature selection to discover a feature subset that is most *useful* to build a good predictor[1]. Being relevant is not necessarily the same as being useful in this sense.

Some researchers have argued that the inclusion of *all* relevant features may not be necessary and may actually reduce performance. This is particularly the case for features that represent the same *factor*, or are correlated and so are *redundant*. The inclusion of a large number of redundant features will over-represent that factor in the classification algorithm and may mean other factors, that are just as important but under-represented, have less weight. This will skew the decision process. Therefore, feature selection algorithms have been developed that have the dual optimization goal of maximizing relevance and minimizing redundancy (MRMR) by measuring information content [101, 119] or by analysis of correlation coefficients between features [189]. These all show improvements in performance over other methods that do not consider redundancy. However, these results are not conclusive. [158] suggested that removing redundant features may not always improve performance and provide artificial data to demonstrate this assertion.

> Noise reduction and consequently better class separation may be obtained by adding variables that are presumably redundant. Variables that are independently and identically distributed are not truly redundant [158].

[1]They quite deliberately focus on the first feature selection goal to maximize performance.

Hence, for data sets that include much noise, the removal of redundant features may actually be a disadvantage.

6.2 Feature Selection for Conformal Predictors

In this section, we consider feature selection in the context of conformal predictors. In particular, we address the following three problems.

1. Ensure we do not incur a selection bias, as discussed in Section 6.1.2.
2. Ensure that the NCM is *exchangeable* as required by Equation 1.3.
3. Ensure reasonable computation time.

To meet the first condition, feature selection should be independent of the test data. Therefore, as a first attempt to implement feature selection, we might design the feature selection process on only the training examples. Consider examples of the form $z = (x, y)$ where x is a vector of features and y is a class label. Let $(z_1, \ldots, z_{n-1})$ be a sequence of $n - 1$ labeled examples given to the conformal predictor to make a prediction for a new unlabeled example x_n. Then feature selection is based only on the first $n-1$ examples. However, this will not work for standard (transductive) conformal predictors (as described in Chapter 1.3) since this procedure breaks the exchangeability requirement. Recall that the value of the NCM does not change with different ordering of the sequence (Equation 1.3). The intuitive meaning of the NCM is that it measures how unusual example i is relative to the whole sequence of n examples. We then see that a feature selection procedure on just the first $n-1$ examples would, in general, lead to a different feature space for different sequences of examples which in turn would, in general, mean that the NCM is not well-defined. However, this is not a problem for inductive conformal predictors (Chapter 2.3) since in the inductive setting, feature selection would be restricted to the training data, therefore giving the same feature subset for the sequence of examples, regardless of ordering.

To implement feature selection in a standard conformal predictor, it is necessary to build the feature selection into the NCM. To do this, an auxiliary nonconformity measure $\tilde{A}$ is introduced which includes a function f that performs feature selection on a sequence of examples and returns a subset of features:

$$\alpha_i = \tilde{A}\,(z_i, (z_1, \ldots, z_n), f(z_1, \ldots, z_n))$$

To ensure exchangeability of $\tilde{A}$, it is therefore necessary that f is exchangeable. That is, the feature subset output by f should be the same for any ordering of the sequence of example. Most feature selection methods are exchangeable in this sense. For example, this is the case for the filter methods described in Section 6.1.1 using means and variances in computing ranks for each feature, such as SNR and FDR. The use of the entire sequence, including the unlabeled nth example, for feature selection does not lead to selection bias, since in TCM the last example is $z_n = (x_n, y)$ where the label y is a conjecture made by a standard conformal predictor to compute p-values for all possible class labels and the feature selection process is never informed of the true label for example n.

This procedure unfortunately increases computational time since it requires the feature selection procedure to be called every time the NCM is used. For large problems, in the online setting, using a computationally intensive feature selection method such as a wrapper, this may make it impractical to implement. In such cases, it may be better to use an inductive conformal predictor. Nevertheless, for simple feature selection methods, such as filter or embedded approaches, the added computational requirement may be small. In the online setting, some feature selection methods allow the selection to be updated iteratively with each new example. For example, filter methods that use summary statistics, such as SNR and FDR, simply need an iterative update to in-class sample means and standard deviations.

We have so far looked at how to implement existing feature selection methods within the context of conformal predictors. However, it is possible to derive feature selection, by taking advantage of properties of conformal predictors. In the next two sections, we describe two such approaches: the Strangeness Minimization Feature Selection and Average Confidence Maximization methods.

6.2.1 Strangeness Minimization Feature Selection

Strangeness Minimization Feature Selection (SMFS) selects the subset of features that minimizes the overall non-conformity value of a sequence of examples. The intuition for this approach is that reducing overall strangeness implies an increase in conformity amongst the examples in the sequence. Therefore, the set of features that minimizes overall strangeness are most relevant to maximizing measured conformity between training examples.

The SMFS goal is defined in relation to any underlying NCM. In common with wrapper feature selection methods, it requires a search over the space of all feature subsets, although restricted to subsets of some fixed size t. For m features, this search has computational complexity $O(m^t)$. The number of selected features t does not need to be very large for this to become impractical. Fortunately, practical implementations of SMFS are possible if we restrict our attention to a subclass of *linear NCMs*. The problem becomes tractable without the need for ad-hoc heuristics that are often required with wrapper methods and, within this framework, we can still implement useful versions of conformal predictors based on a wide range of learning algorithms. We find that SMFS is a principled, broad and practical feature selection framework, for which distinct feature selection methods are determined by NCMs. We can also apply the principle in reverse to derive novel NCMs based on existing feature selection methods. [30] have shown that it has practical application, with results in classifying leukaemia diagnosis based on gene expression data.

Given an auxiliary NCM $\tilde{A}$, SMFS can be defined formally as the feature

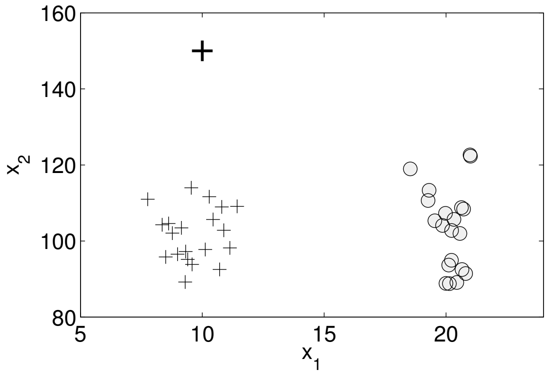

Figure 6.2: Simulated data to illustrate SMFS

selection function

$$f_{\mathrm{SMFS}}(z_1, \ldots, z_n) = \arg\min_{S \in G} \sum_{i=1}^{n} \tilde{A}(z_i, (z_1, \ldots, z_n), S) \tag{6.1}$$

where $G = \{R : R \subseteq F,\ |R| = t\}$ and F is the total feature space available. So long as $\tilde{A}$ is exchangeable for any fixed S, it follows that f_{SMFS} is also exchangeable. Hence a conformal predictor using the auxiliary NCM $\tilde{A}(z_i, (z_1, \ldots, z_n), f_{\mathrm{SMFS}}(z_1, \ldots, z_n))$ will be well-calibrated.

Example

Simulated data is shown in Figure 6.2, forming a simple classification problem. The large cross (+) is clearly far from the clustering of other crosses and so we would expect it to have a relatively high strangeness value in relation to all the others.

We use an auxiliary NCM based on the Euclidean distance from a centroid restricted to a feature subset S,

$$\alpha_i^S = \sqrt{\sum_{j \in S} \left(x_{ij} - \mu_{(y_i)j}\right)^2 \ \sigma_j^{-2}}.$$

where the index j refers to the jth feature in a feature vector x_i and σ_j is the sample standard deviation of feature j. Using this auxiliary NCM, we find that all examples have a non-conformity value less than 1.5, except the large cross which has the relatively high non-conformity value 4.5 as we had expected, since it is so different from all the others. We can compute the sum of non-conformity values restricted to feature subsets of size $t = 1$:

$$\sum_{i=1}^{n} \alpha_i^{\{1\}} = 1.0, \qquad \sum_{i=1}^{n} \alpha_i^{\{2\}} = 6.4$$

Therefore, using the SMFS method, the first feature would be chosen as most relevant since it minimises the total non-conformity value. By observation, it is clear from Figure 6.2 that this is the correct choice since the first feature (x_1-axis) has more discriminatory power than the second (x_2-axis).

β-non-conformity values

The use of linear NCMs is central to implementing SMFS efficiently. In conformal predictors, non-conformity is computed as α-non-conformity values for each example. By using linear NCMs, non-conformity values are computed for each feature instead. We call them *β-non-conformity values* and the SMFS function is reformulated in terms of them.

An auxiliary NCM is defined as a *linear NCM* if

$$\tilde{A}(z, (z_1, \ldots, z_n), S) = \sum_{j \in S} \phi(z, j, (z, (z_1, \ldots, z_n))$$

for all examples z, sequences $(z_1, \ldots, z_n)$ and $S \subseteq F$, for some exchangeable transformation function ϕ. The β-non-conformity value for feature j is defined as

$$\beta_j = \sum_{i=1}^{n} \phi(z_i, j, (z_1, \ldots, z_n)).$$

It follows that

$$f_{\mathrm{SMFS}}(z_1, \ldots, z_n) = \arg\min_{S \in G} \sum_{j \in S} \beta_j.$$

Hence, the SMFS optimisation problem is solved by (1) computing β-non-conformity values for each feature, (2) sorting the β-non-conformity values in ascending order, (3) choosing the top t features in this sorted list. This algorithm is $O(m \log m)$ in the number of features m.

Although the SMFS goal is defined in a way similar to a wrapper feature selection method as a search to minimise the non-conformity value across all possible feature subsets, the implementation for linear NCM involves searching for the subset of features with smallest β-non-conformity values. As such, the implementation is like filter feature selection. In fact, $-\beta_j$ can be used as a feature ranking function. We can also reverse this relationship and build new NCMs based on existing ranking functions. In the next three sections we explore these approaches for different underlying learning methods: the nearest centroid (NC) and support vector machine (SVM) classifiers, and for the ranking function SNR.

Nearest centroid linear NCM

The NC classifier computes centroids for each class and for a new example will predict the class with the nearest centroid based on a given distance metric. Using Euclidean distance, a plausible NCM is given by

$$\tilde{A}_{\mathrm{NC}}((x_0, y_0), (z_1, \ldots, z_n), S) = \sum_{j \in S} \left(x_{0j} - \mu_{(y_0)j}\right)^2 \sigma_j{}^{-2}$$

with μ_{yj} and σ_j computed on the sample sequence $(z_1, \ldots, z_n)$. Then

$$\phi_{\mathrm{NC}}((x_0, y_0)), j, (z_1, \ldots, z_n)) = \left(x_{0j} - \mu_{(y_0)j}\right)^2 \sigma_j{}^{-2}.$$

and, where Y is the set of all class labels and $C_y = \{i : z_i = (x_i, y)\}$,

$$\beta_j = \sum_{i=1}^{n} \left(x_{0j} - \mu_{(y_0)j}\right)^2 \sigma_j{}^{-2} = \sigma_j^{-2} \sum_{y \in Y} \left[(|C_y| - 1)\, \sigma^2_{(y)j} \right]. \tag{6.2}$$

The negation of this β-NCM forms a natural ranking function for feature filtering since it measures the ratio of the weighted sum of within-class variances and total variance. A low ratio implies greater variance between classes than noise within classes.

Linear Classifier NCM

A linear classifier can be characterized by a score, linear on weighted features, giving a prediction of outcome:

$$\hat{y} = g(w \cdot x)$$

where w is a vector of feature weights and g is a monotonically increasing function. For example, in logistic regression, g is the logit function, and in SVM, it is the sign function with label space $Y = \{-1, +1\}$. A plausible NCM is based on the distance of an example i from the separating hyperplane:

$$\tilde{A}_{\mathrm{LC}}((x_i, y_i), (z_1, \ldots, z_n), S) = -y_i \sum_{j \in S} w_j x_{ij}$$

where the weight vector w has been computed using an inductive learner l, such as SVM. That is, $w = l(z_1, \ldots, z_n)$. A version of this NCM was originally derived for logistic regression by [411]. This NCM then yields

$$\beta_j = -w_j \sum_{i=1}^{n} y_i x_{ij}$$

SNR linear NCM

We can use feature ranking functions, such as SNR, to form a β-NCM that can be used to build a new implementation of NCM.

Consider a learning environment with binary label space $Y = \{-1, +1\}$, and transformation functions with the general form

$$\phi_{\mathrm{F}}((x_i, y_i), j, (z_1, \ldots, z_n)) = -\frac{y_i}{|C_{y_i}|} w_j x_{ij} \tag{6.3}$$

This gives the NCM

$$\alpha_i^S = -\frac{y_i}{|C_{y_i}|} \sum_{j \in S} w_j x_{ij}$$

and

$$\beta_j = -w_j(\mu_{(+1)j} - \mu_{(-1)j}).$$

Taking

$$w_j = \frac{\mathrm{sgn}(\mu_{(+1)j} - \mu_{(-1)j})}{\sigma_{(+1)j} + \sigma_{(-1)j}}$$

gives $-\beta_j$ as SNR. Other forms of w give the FDR and t-statistic ranking functions. Therefore, we find that the general form of transformation function (6.3) is versatile. It is interesting that it has the same form as the linear classifier NCM, except for normalizing each example by the number of training examples in its class.

Results for microarray classification

NCM with feature selection was used to classify microarray data [27, 29]. In this section we give results using NCM to classify different forms of childhood leukaemia based on microarray data. The two major forms of acute leukaemia are ALL (acute lymphoblastic leukaemia) and AML (acute myeloblastic leukaemia). A data set of 132 children with ALL and 130 with AML was available for analysis [327, 326]. Each observation had 22,283 gene probe features. Classification was performed using different implementations of NCM to compare predictive performance. Off-line learning was used with 10-fold cross-validation. The same division of data into folds was used for each algorithm to ensure fair comparison. Results are given in Table 6.1. Accuracy (Acc.) measures the fraction of predictions that are correct. In all cases, the accuracy is greater than the confidence level demonstrating the results are well-calibrated, as we expect. The NCM classifier works by allowing region predictions. That is, the prediction is a *set* of possible outcomes [411]. In this way, accuracy can be controlled. A prediction is called *certain* if it is a single class label and is called *uncertain* otherwise. In Table 6.1, efficiency (Eff.) measures the fraction of certain predictions. Therefore, larger values represent more efficient algorithms. Table 6.1 shows that using no feature selection yields poor efficiency at all confidence levels. Although using SNR for feature selection improves efficiency, SMFS gives the best results with efficiency close to 1 even at a 97.5%

confidence level (both methods were used to select the top 40 features). The choice of underlying classifier (SVM or NC) makes little difference to the results.

Table 6.1: Performance of NCM classifiers for childhood leukaemia classification.

		Confidence level					
	FS	97.5%		95%		90%	
Classifier	method	Acc.	Eff.	Acc.	Eff.	Acc.	Eff.
SVM	None	0.989	0.034	0.966	0.065	0.939	0.160
SVM	SNR	0.989	0.496	0.977	0.710	0.973	0.966
SVM	SMFS	0.989	0.996	0.989	1	0.989	1
NC	SMFS	0.992	0.992	0.992	0.992	0.992	0.992

The same data was also used for the more difficult task of sub-classifying AML. There are 6 subtypes of AML in the data with small sample sizes: MLL (23), AML1-ETO (21), FAB-M7 (10), PML-RARα (15), CBFβ-MYH11 (14) and Other (47). Figure 6.3 shows results using NCM-NC with SNR feature selection. Changing confidence level is shown on the horizontal axis. The calibration line shows the maximum number of errors expected at each confidence level. We see that the actual number of prediction errors is always less than this and for higher confidence levels is close to the calibration line. This is a clear demonstration of the calibration property of NCM, and that this property is preserved with feature selection. The figure also shows that with rising confidence level, beyond a 90% level, although number of errors are controlled, the number of uncertain predictions rises quite sharply.

6.2.2 Average Confidence Maximisation (ACM)

As one can see, SMFS combined ideas of filters and wrappers in the feature selection: the features were sorted by β-non-conformity values (like filtering) but the calculation of these values was based on the whole feature set which makes the method a wrapper. An alternative approach based on conformal prediction (Algorithm 8) was originally presented in [432]. It also uses elements of the embedded feature selection methodology because the process of selection is connected with the process of confident prediction. It is a step-wise method of increasing feature set size with average confidence as the optimization goal, which can be also used as the stopping criterion.

Suppose we try to separate the two classes just by one feature. We take all examples from the dataset that belong to one of these two classes and process it in the online mode in order to average the results over different sizes of the training set. This is done for each feature, so the feature with the largest average confidence will be the first important feature of the set of useful features. We then continue in the same way with the new question: what is the second feature that can be added to this one in order to maximize average confidence? On answering this question, we will have a list of two features; we then look for

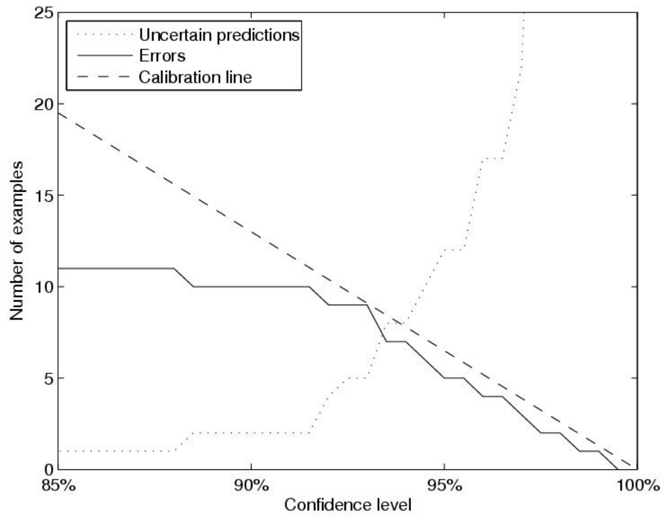

Figure 6.3: NCM-NC results classifying AML subtypes.

the third feature and so on. Adding new features can be stopped when we reach either a desired number of features or at a selected confidence level when it is reached. If it is never reached, it might be reasonable to stop when it stabilizes and stops growing.

A typical result follows in Figure 6.4 [432]. The sample dataset was part of the Abdominal Pain dataset [300] containing 126 Appendicitis (APP) and 173 Dyspepsia (DYS) patients with 33 features that are clinical symptoms. The figure shows the dependence of the average confidence level on the number of selected features as it increases according to the method. Based on this figure, a user can decide at what number of features to stop, depending on the desired level of confidence: 10 features are enough for 90% confidence, 20 are needed for 95%. Alternatively, one can stop when the confidence stabilizes and stops increasing: this is at about 25 features.

6.3 Comparison and conclusions

We presented two feature selection methods, SMFS and ACM, based on conformal prediction in the previous sections. Our experience has revealed that SMFS is usually better in the sense that it agrees with the selection of features given by human experts. On the other hand ACM has some advantages from a theoretical point of view. It is based, not on an NCM directly as SMFS is,

Algorithm 8 Feature Selection by Conformal Predictors

```
Input: feature subset $U$, candidate feature $\overline{U}$, $z_1 = (x_1, y_1), z_2 = (x_2, y_2), ..., z_l = (x_l, y_l)$
Input: a set of new object $\{x_{l+1}, x_{l+2}, ..., x_{l+n}\}$
Input: strangeness measure $A$
  for $U = U \bigcup \overline{u}$, $\overline{u} \in \overline{U}$ do
    for i=1:n do
      for $y \in Y$ do
        $z_{l+i} = (x_{l+i}, y)$
        $(\alpha_1, \ldots, \alpha_{l+1}) = A(\wr z_1, z_2......z_{l+i} \wr / z_j, z_j)$
        $p(y) = \frac{\#\{j=1,...,l+i: \alpha_j \geq \alpha_{l+i}\}}{l+i}$
      end for
      confidence$(i) = 1-$ the second largest $p$
      $z_{l+i} = (x_{l+i}, y_{true})$
      $Z = Z \bigcup z_{l+i}$
    end for
  end for
  $U = U \bigcup \overline{u}$ which achieve highest average confidence
```

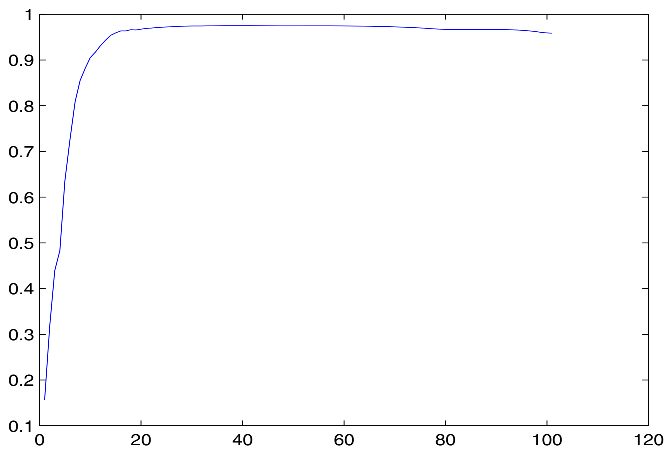

Figure 6.4: Confidence level for different number of features (APP vs DYS)

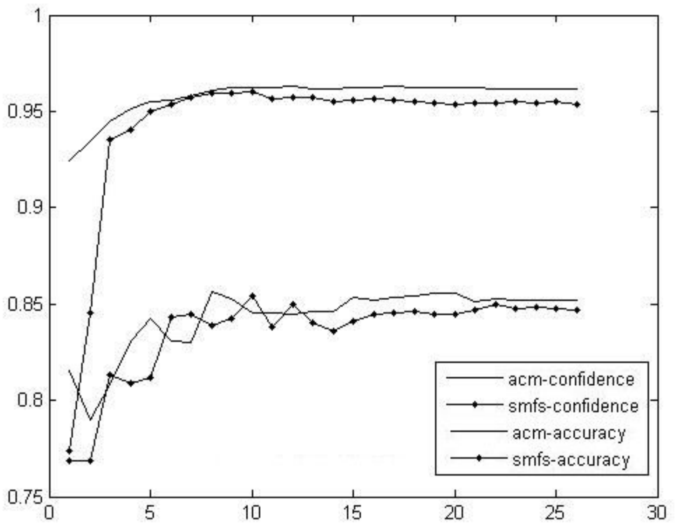

Figure 6.5: accuracy and average confidence for APP

but on the conformal prediction output (confidence). This means that a bad choice of NCM will be better reflected in the results (low average confidence) giving a warning to the user before the results of feature selection are assessed. Another advantage of ACM is the flexibility of the stopping criterion: instead of fixing the number of selected features, one can select a level of confidence to be achieved. This was illustrated in the example above.

In order to have a direct comparison between the two algorithms, we applied both of them to the same dataset using the NCM (Nearest Centroid) method. The Nearest Centroid for an example x_i classified as y_i is

$$\alpha_i = \frac{d(\mu_{y_i}, x_i)}{min_{y \neq y_i} d(\mu_y, x_i)}$$

. The two methods are applied to the problem of classification of appendicitis in the abdominal pain dataset (Section 6.2.2). The comparison shows that both methods have about the same quality in accuracy and average confidence (Fig. 6.5). Although further empirical work is required, results so far suggest that in the context of conformal predictors, good results are obtained using both SMFS and ACM, in comparison to standard methods.