An Excellent Feature Selection Model Using Gradient-Based and Point Injection Techniques

D. Huang and Tommy W.S. Chow

Department of Electrical Engineering, City University of Hong Kong
dihuang@cityu.edu.hk, eetchow@cityu.edu.hk

Abstract. This paper focuses on enhancing the effectiveness of filter feature selection models from two aspects. One is to modify feature searching engines based on optimization theory, and the other is to improve the regularization capability using point injection techniques. The second topic is undoubtedly important in the situations where overfitting is likely to be met, for example, the ones with only small sample sets available. Synthetic and real data are used to demonstrate the contribution of our proposed strategies.

1 Introduction

As computer technology advances rapidly, data are accumulated in an enormous speed unprecedentedly experienced in human history. In some advanced engineering and physical science applications, most conventional computational methods have already experienced difficulty in handling the enormous data size. In handling these data sets, feature selection is an essential and widely used technique. It reduces the size of features through eliminating irrelevant and redundant features, and thus results with increased accuracy, enhanced efficiency, and improved scalability for classification and other applications such as data mining (Han, 2001). Feature selection is especially important when one is handling a huge data set with dimensions up to thousands.

A feature selection framework generally consists of two parts: a searching engine used to determine the promising feature subset candidates and a criterion used to determine the best candidate (Liu, 1998; Molina, 2002). Currently, there are several searching engines: ranking, optimal searching, heuristic searching and stochastic searching. Among these engines, heuristic searching, which can easily be implemented and is able to deliver respectable results (Pudil, 1994), is widely used. Feature selection models can be broadly categorized as filter model, wrapper model, and embedded model according to their evaluation criteria. Filter models explore various types of statistical information, such as distribution probabilities underlying data. Wrapper and embedded models are classifier-specified and the selected features may vary with different classifiers. Given a feature subset, say S, wrapper and embedded models firstly require to build a classifier based on S. Wrapper models then rely on the performance of the built classifier to determine the goodness of S, while embedded models make use of the parameters of the built classifier to assess S. Wrapper models are usually more computationally expensive than filter models.

I. King et al. (Eds.): ICONIP 2006, Part II, LNCS 4233, pp. 679 – 692, 2006.

In a filter model, good feature selection results rely on a respectable evaluation criterion and an appropriate searching strategy. The former issue has been heavily investigated. Various types of information, including mutual information (Battit, 1994; Bonnlander, 1996; Chow, 2005), correlation (Hall 1999), etc., have been explored for evaluating features. By comparison, there are fewer studies focused on searching engines. Also, most of those studies are completely designed in discrete feature domains. For example, sequential forward searching (SFS), a typical heuristic searching scheme, identifies k important features from unselected features and places them into a selected feature subset in each iteration. To improve SFS, a stepwise strategy is designed – in each iteration, selecting k "good" unselected features is followed by deleting r "worst" selected features ($r < k$) (Pudil, 1994). And Al-Ani *et al.*, (2000) employ "elite" selected features, not all of them, to identify important features from unselected ones. These algorithms, studied in a discrete feature space, depend on the testing of more feature combinations in order to deliver improved results. Clearly, more testing will increase the computational complexity.

Given a set of n samples $D = \{(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)\}$ which is drawn from a joint distribution P on $X \times Y$, a feature selection process is per se a learning process in the domain of $X \times Y$ to optimize the employed feature evaluation criterion, say $L_{(x,y)\sim P}(x, y)$. As P is unknown, $L_{(x,y)\sim P}(x, y)$ has to be substituted by $L_{(x,y)\in D}(x, y)$. Clearly, when D cannot correctly represent P, this substitution may cause overfitting in which the selected features are unable to deal with testing data satisfactorily despite performing splendidly on the training data D (Bishop, 1995). In many applications, a machine learning process suffers from insufficient learning samples. For instance, in most microarray gene profile expression based cancer diagnosis data sets that may only consist of tens samples. With small sample sets, overfitting is likely to happen and this issue must be addressed accordingly. A wrapper/embedded feature selection models always involve with classification learning processes. Thus, the regularization techniques developed for classification learning can be directly employed in a wrapper/embedded model. For example, support vector machine and penalized Cox regression model, which have been argued to have high generalization capability, are employed in embedded models (Guyon, 2002; Gui, 2005). And an embedded feature selection model is trained based upon with regularized classification loss functions (Perkins, 2003). On the other hand, since filter schemes do not explicitly include a classification learning process, the regularization techniques developed for classification learning cannot be explored. In this sense, it is a need to design specified regularization strategies. To our knowledge, only a few attempts are done on this topic. In order to address problem of overfitting, a bootstrap framework has been adopted for mutual information estimation (Zhou, 2003). Under this framework, mutual information estimation should be conducted several times in order to deliver one result. The bootstrap framework is thus highly computationally demanding, which precludes it from being widely used.

In this paper, we propose two strategies – the one is for improving the effectiveness of searching engines, and the other is for addressing the problem of overfitting. And we choose a typical filter feature selection model as example to demonstrate these strategies. In this filter model, the searching engine and the feature evaluation index are SFS (Devijver, 1982) and Bayesian discriminant criterion (BD) (Huang, 2005), respectively. We firstly analyze SFS according to the well-established

optimization theory (Bishop, 1995). The analysis of this type, which has been overlooked in previous studies, can reveal the shortcoming of conventional SFS – SFS is unable to perform optimization in a maximal way. To address this issue, we naturally come to the optimization theory. As a result, a modified SFS is proposed, which conducts the feature searching along the possible steepest optimization direction. To enhance the regularization capability, a point injection approach is proposed. This approach generates certain points according to the distribution of given samples, which is similar to the ones developed for classification learning. In our proposed approach, the injected points are just employed for evaluating the feature subsets. This mechanism is able to minimize the undesired side-effect of injected points.

In the next section, the *BD* sequential forward searching (SFS) feature selection model is briefed. After that, our proposed strategies are described in section 3. Finally the proposed strategies are extensively evaluated.

2 Bayesian Discriminat Based Sequential Forward Feature Searching Process

Assume that the feature set of n-sample dataset D is $F = \{f_1, f_2, \ldots, f_M\}$. Also, each pattern (say, x_i) falls into one of L categories, i.e., $y_i = \omega_k$ where $1 \le i \le n$ and $1 \le k \le L$.

2.1 Bayesian Discriminant Feature Evaluation

In filter models, probability based feature evaluation criteria are commonly used. Bayesian discriminant criterion (BD), a typical probability based approach, is developed by Huang *et al.* (2005). With the dataset D, BD is defined as

$$BD(S) = \frac{1}{n}\sum_{i=1}^{n} \log \frac{p_S(y_i \mid x_i)}{p_S(\bar{y}_i \mid x_i)} = \frac{1}{n}\sum_{i=1}^{n} \log \frac{p_S(y_i \mid x_i)}{1 - p_S(y_i \mid x_i)}, \quad (1)$$

where $\bar{y}_i$ means all the classes but class y_i, and $p_S(.)$ represents a probability which is estimated in the data domain defined by S. As shown in (1), $BD(S)$ directly measures the likelihood of given samples being correctly recognized in the data domain defined by S. A large BD(S), which indicates that most given samples can be correctly classified, is preferred.

And in our study, the probabilities required by $BD(S)$ are estimated with Parzen window (Parzen, 1962) which is modeled as

$$p(x, y) = \sum_{all\ (x_i, y_i) \in \text{class } y} p(x_i)\, p(x \mid x_i) = \sum_{y_i = y} p(x_i)\, \kappa(x - x_i, h_i), \quad (2)$$

$$p(x) = \sum_{all\ classes} p(x, y) = \sum_{all\ (x_i, y_i) \in D} p(x_i)\, \kappa(x - x_i, h_i), \quad (3)$$

where κ and h_i are the kernel function and the width of window, respectively. The parzen window estimator (2) or (3) has been shown to be able to converge the real probability when κ and h_i are selected properly (Parzen, 1962). κ is required to be a finite-value nonnegative function and satisfies $\int \kappa(x - x_i, h_i)dx = 1$. And the width of

κ, i.e. h_i, is required to have $\lim_{n\to\infty} h = 0$ where n is the number of given samples. Following the common way, we choose Gaussian function as κ. That is,

$$\kappa(x - x_i, h_i) = G(x - x_i, h_i) = \frac{1}{(2\pi h_i^2)^{M/2}} \exp\left(-\frac{1}{2h_i^2}(x - x_i)(x - x_i)^T \right),$$

where M is the dimension of x. And the window width h_i is set with h_i=2*distance*(x_i,x_j) where x_j is the 3rd nearest neighbor of x_i. We use Euclidean distance, i.e., $\text{distance}(x_i, x_j) = \sqrt{(x_i - x_j)(x_i - x_j)^T}$ for two data vectors x_i and x_j. As to $p(x_i)$ of the equations (2) and (3), it is estimated with $p(x_i) = 1/n$. With the equations (2) and (3) and based on $p(y|x)=p(x,y)/p(x)$, $p(y|x)$ required by *BD(S)* is finally obtained.

2.2 Sequential Forward Searching

In a BD based feature selection process, the aim is to determine the feature subset S that can maximize *BD(S)* (1). In general, *BD*(S) is optimized in the following way: after a pool of feature subsets is suggested by a searching engine, BD of each suggested feature subset is calculated, and one with the largest BD is either outputted as the finial feature selection result or remembered as the reference to guide the subsequent feature selection process. Many schemes for determining feature subset pools have been developed to trade the quality of optimization results with computational consumption. Among these schemes, the sequential forward searching (SFS) is the most popular one.

The SFS firstly sets the selected feature set (denoted by S, below) empty and enriches S through iteratively adding k important features into it. In each iteration, to select the k features, all the feature combinations {S, k unselected features} are examined, and the one with the largest BD is selected out to remember as a new S. Based upon this S, another iteration of feature selection is conducted. This process continues until certain stopping criteria are met.

3 Modified Sequential Forward Searching Scheme

3.1 Weighting-Sample

The objective of feature selection is to optimize the employed evaluation criterion, for example, *BD*(S) (1) in this study, through adjusting S. To clearly explain our idea, we recast *BD*(S) (1) as

$$BD(S) = \frac{1}{n}\sum_{i=1}^{n} \log \underbrace{\frac{p_S(y_i \mid x_i)}{1 - p_S(y_i \mid x_i)}}_{f((x_i, y_i), S)} = \frac{1}{n}\sum_{i=1}^{n} \log f((x_i, y_i), S) . \qquad (4)$$

According to the optimization theory, the steepest direction of adjusting S to maximize (4) is determined by

$$\frac{\partial BD(S)}{\partial S}=\frac{1}{n}\sum_{i=1}^{n}\frac{\partial BD(S)}{\partial f((x,y),S)}\frac{\partial f((x,y),S)}{\partial S}\bigg|_{(x_i,y_i)} \tag{5}$$

It shows that, to optimize $BD(S)$, the updating of S depends on the two terms, $\partial BD(S)/\partial f((x,y),S)$ and $\partial f((x,y),S)/\partial S$. The former one happens in a continuous domain, while the latter one is related to S and has to be tackled in a discrete feature domain. In this sense, (5) cannot be solved directly. To maximize $BD(S)$, SFS tests all combinations of S and an unselected feature, and remains the one having the maximal BD. Clearly, SFS only considers the second term of (5), but overlooks the first term. It means that the searching direction of SFS is not in accordance with the steepest optimization one. This shortcoming may reduce the optimization effectiveness, and thus motivates our modification.

Naturally, our proposed strategy is based on the optimization theory, i.e., equation (5). The second term of (5) is resolved by using any conventional discrete-domain searching scheme. We use SFS for this purpose. The first term of (5) can be directly calculated in the way of

$$\frac{\partial BD(S)}{\partial f((x,y),S)}=\frac{\partial \log(f(x,y),S)}{\partial f((x,y),S)}=\frac{1}{f((x,y),S)}=\frac{1-p_S(y\mid x)}{p_S(y\mid x)}. \tag{6}$$

This shows that $\partial BD(S)/\partial f((x,y),S)$, which is only related to x, is independent of the change making on S. With this observation, we use (6) as weights to samples. In such way, feature searching is conducted with the weighted samples, not the original ones.

Assume that the dataset D is weighted by $\{w_1,w_2,\ldots,w_n\}$. With this weighted dataset, the criterion BD (1) and the probability estimations (2) and (3) are adjusted accordingly. The rule of $p(x_i)=1/n$ is replaced by $p(x_i)=w_i/n$. Also, we have

$$p(x,y)=\sum_{all\,(x_i,y_i)\in \text{class } y}\frac{w_i}{n}G(x-x_i,h_i). \tag{7}$$

And the criterion BD is modified as

$$BD(S)=\frac{1}{n}\sum_{i=1}^{n}w_i\log\frac{p_S(y_i\mid x_i)}{1-p_S(y_i\mid x_i)}=\frac{1}{n}\sum_{i=1}^{n}w_i\log\frac{p_S(x_i,y_i)}{\sum_{all\ y_j\neq y_i}p_S(x_i,y_j)}. \tag{8}$$

Apparently, it is natural to regard different samples may have different contributions to the learning processes. Currently, most machine learning algorithms have already incorporated this idea. For instance, the classification learning aims to minimize the mean square error $L(\Lambda)=\sum_{\text{all}\,(x_i,y_i)}(f(x_i,\Lambda)-y_i)^2$ by training the model f, i.e., adjusting the parameter set $\wedge$ of f. The steepest decent type algorithm, which is commonly used for classification learning, determines the updating direction with

$$-\frac{\partial E}{\partial \Lambda}=\sum_{\text{all}\,(x_i,y_i)}-\frac{\partial E}{\partial f}\frac{\partial f(x,\Lambda)}{\partial \Lambda}\bigg|_{x=x_i,y=y_i}=\sum_{\text{all}\,(x_i,y_i)}-\big(f(x,\Lambda)-y\big)\frac{\partial f(x,\Lambda)}{\partial \Lambda}\bigg|_{x=x_i,y=y_i}, \tag{9}$$

where (x_i,y_i) is a given training sample. It is noted that the contribution of (x_i,y_i) is penalized by $|f(x_i,\wedge)-y_i|$. Another example is AdaBoosting (Hastie et al., 2001), a typical boosting learning algorithm. During the course of learning, AdaBoosting repeats weighting the sample (x_i,y_i) with $w_i e^{-y_i f(x_i)}$ where w_i is the current weight to (x_i,y_i). Also, in order to reduce the risk of overfitting, it is intuitively expected that the negative samples (i.e., incorrectly-recognized ones) have more influence to the subsequent learning than positive ones do. In such a way, the convergence rate can be speeded up, and the problem of overfitting can be alleviated (Lampariello *et al*., 2001). AdaBoosting clearly can meet this expectation. The equation (9), however, indicates that the steepest decent algorithm fell short on tackling overfitting in a way that the correctly-recognized patterns still carry large weights. This fact has motivated modifications on the gradient-based algorithms (Lampariello *et al*., 2001). Consider our proposed weighting-sample strategy, defined by equation (6). It penalizes the negative patterns heavily. Thus it will be helpful in alleviating the problem of overfitting.

3.2 Point Injection

Overfitting is caused by the deviation between the real optimization goal and the actual achievable optimization objective. The real goal of the BD based feature selection process is to maximize $BD_P(S)$ where P is the underlying probability. Since P is unknown in most cases, $BD_P(S)$ can not be actually defined, and thus has to be substituted with its empirical estimate $BD_D(S)$ (simplified as $BD(S)$, like equation (1) does). When $BD(S)$ cannot always reflect $BD_P(S)$ correctly, overfitting is caused. To avoid overfitting, it is preferred that $BD_P(S)$ varies smoothly enough.

In the area of classification/regression, overfitting can be tackled through modifying the employed empirical objective function with regularization terms. These regularization terms penalized the complex models. With them, the simple learned models can be obtained, and the likelihood of overfitting happens will thus be decreased (Bishop, 1995). The penalty terms, however, cannot always be built without thorough theoretical analysis. This is especially the case when the parameters or factors controlling smoothness of a training model are hard to determine. Another widely used regularization technique is point injection. It is known that smooth means that samples near to each other should correspond to similar performance, which is the rationale behind the techniques of point injection. In many literatures, this technique is referred as *noise injection* (Matsuoka 1992; Skurichina *et al.*, 2000; Zagoruiko *et al.*, 1976), but it is certainly expected that injected points are not real noise. Thus, to avoid the confusion, we use the term *point injection* instead of *noise injection* in this paper.

Under the frameworks of classification/regression learning, injected points are always treated just like the original samples – a classifier/regression model is built upon the original samples as well as the injected points. This working mechanism requires high-quality points. Spherical Gaussian distributed points are generated around each training object (Bishop, 1995; Matsuoka 1992). Then, the undesirable fact that the added points may increase the complexity of the solved problem is revealed. To avoid this, high quality injected points, such as, k-NN direction points (Skurichina *et al.*, 2000) and eigenvector direction points (Zagoruiko *et al.*, 1976), are

suggested to replace Gaussian distributed points. Also, points are generated in a way of feature-knock-out (Wolf *et al.*, 2004). With the injected points of improved quality, contributions of injected point techniques are naturally enhanced. In this study, we reduce the risk caused by point injection through adopting a different working mechanism. Under our mechanism, only the given samples are used for building the probability estimators required by our feature evaluation criterion BD, and the given samples as well as the injected points are employed for evaluating feature subsets. Without participating in the process of model-building, the undesirable impacts of injected points must be reduced.

Around a pattern x_i, a point injection technique adds v points which are generated from a distribution $b(x\text{-}x_i)$. v and $b(x\text{-}x_i)$ play important parts in a point injection scheme (Kim, 2002; Skurichina, 2000). In order to strike the balance between performance stability and computational efficiency, v can be determined. Also, it has been argued that, for the reasonable choice of v, such as v = 8, 10 or 20, the effect of point injection is slightly different (Skurichina, 2000). We thus set v = 10. As to $b(x\text{-}x_i)$, the "width" of $b(x\text{-}x_i)$, which determines the variance of the injected points, is crucial. Since the aim of point injection is to test the properties of the region around x_i, a large width of $b(x\text{-}x_i)$ is not expected. And a small width of $b(x\text{-}x_i)$ must correspond to the insignificant contribution.

To determine an appropriate width, the simulation based strategies can be used (Skurichina, 2000). We develop an analytic approach to determine the width of $b(x\text{-}x_i)$. This approach is inspired by the ideas mentioned in (Glick, 1985; Kim, 2002). Aiming to reduce the bias intrinsic to the re-substitution error estimation as much as possible (Glick, 1985), our approach depends on the joint distribution (X,C) to determine the width of $b(x\text{-}x_i)$. Around a given pattern, say x_i, we generate several points around from Gaussian distribution $N(x_i,\sigma_i)$ where $\sigma_i = d_i/2$ and d_i is the distance of x_i to the nearest samples, i.e.,

$$d_i = \arg \min_{j,\, j \neq i} \left\| x_i - x_j \right\|. \tag{10}$$

In this way, it can be guaranteed that x' having $\|x_i\text{-}x'\| = d_i$ occurs with the close-zero probability.

The given sample set D cannot cover each part of the whole data domain very well. In turn, the probability estimators built with these samples cannot describe every part of the data domain. In detail, there may exist the parts where the conditional probabilities $p(x|y)$ for all classes are very small. According to the equation (8), it is that $\left|\frac{\partial BD(S)}{\partial p(x,\omega)}\right| \propto \frac{1}{p(x,\omega)}$ for all classes ω. It indicates that, when all $p(x|y)$ are small, a very little change of x will cause a large change of $BD(S)$. The points of such type are not expected.

For the originally given samples on which the probability models are built, at least one $p(x|y)$ must be large enough. On the other hand, an injected point may be uncertain. That is, all the probabilities about it are very small. It is better to minimize the impact of uncertain points, although it can be argued that they may equally affect the quality of different feature subset candidates. With this idea, the way of

calculating $BD(S)$ of injected points is modified. Suppose that, according to the given D, we generate dataset D' for which we have

$$BD(S)\big|_{D'} = \frac{1}{|D'|}\sum_{\text{all }(x'_i,y_i')\in D'} w'_i \log\frac{p_S(y'_i \mid x'_i)}{p_S(\overline{y'_i} \mid x'_i)}\underbrace{\left(\arg\max_{all\ \omega}(p_S(x'_i \mid \omega))\right)}_{A} \tag{11}$$

where $|D'|$ means the cardinality of D'. y'_i and w'_i are the weight and class label of x'_i and are inherited from the corresponding sample in D. With part A, the impact of uncertain points will be limited, which satisfies our expectation.

Below, contributions of the point injection strategy are assessed on a group of 3-class and 8-feature synthetic datasets. In these data, the first four features are generated according to

Class 1 ~ m samples from $N((1, 1, -1, -1), \sigma)$,
Class 2 ~ m samples from $N((-1, -1, 1, 1), \sigma)$,
Class 3 ~ m samples from $N((1, -1, 1, -1), \sigma)$.

And the other four features are randomly determined from normal distribution with zero means and unit variance. Clearly, among totally eight features, the first four are equally relevant to the classification task, and the others are irrelevant. Three feature selection methods are applied to this data to determine four salience features. They are the conventional SFS, SFS with the feature-knock-out and the proposed point injection approaches. Only if all relevant features are selected out, the selection results can be considered correct.

Table 1. Comparisons on a synthetic data. These results demonstrate the merits of the proposed point injection strategy.

		SFS	SFS with feature-knock-out strategy	SFS with the point injection strategy
$\sigma = 0.3$	m = 3	0.980	0.941	0.991
	m = 9	1.000	1.000	1.000
$\sigma = 0.8$	m = 3	0.357	0.358	0.392
	m = 9	0.933	0.930	0.931

Different settings of σ and m are investigated. For reliable estimation, in each setting, three feature selection methods are run on 10,000 datasets independently generated. And the correct results over 10,000 trials are counted. In Table 1, the correctness ratios are presented. It shows that the feature-knock-out point injection strategy cannot bring the improved feature selection results in this example. This may be because this strategy is originally designed for classification learning, not for feature selection. Turn to the proposed point injection strategy. Its advantage becomes more significant either when the sample size becomes small or when σ becomes large. All these conditions actually mean there is a high likelihood of overfitting since

a larger σ means a more complex problem. Thus, the presented results suggest that our approach can improve the generalization capability of SFS.

3.3 Procedure

With the above described weighting-sample and point injection strategies, the conventional BD based SFS feature selection models are modified as follows.

Step 1. (Initialization) Set the selected feature set S empty. Also set the injected point set D' empty. Also, for each sample, assign a weight of 1, i.e., $w_i = 1$, $1 \le i \le n$.

Step 2. (Feature selection) From the feature set F, identify the feature f_m which satisfying

$$f_m = \arg\max_{f \in F} \left[BD(f + S)|_D + BD(f + S)|_{D'} \right].$$

The probability estimators required by BD are established with (7) based on the dataset D. And the BDs on D and D' are defined in (8) and (11) respectively. Put the feature f_m into S and delete it from F at the same time.

Step 3. (Update the sample weights) Set w_i based on equation (6). Then normalize w_i as $w_i = w_i \Big/ \sum_{j=1}^{n} w_j$.

Step 4. (Point-injection) Set D' with empty. In the data domain described by S, conduct point injection around each sample in the following way.

Around the pattern x_i, produce 10 points based on the distribution $N(x_i, d_i/2)$ where d_i is defined by the equation (10). Place these points into D'. Also, the class label and the weight of these injected points are set with y_i and w_i, respectively.

Step 5. If the size in S has reached the desired value, then Stop the whole process and output S, otherwise Go to step 2.

4 Experimental Results

Our modified SFS, called gradient and point injection based SFS (gp-SFS), is evaluated through comparing with several related methods, namely, the conventional SFS, support machine learning recursive feature elimination scheme (SVM RFE) (Guyon, 2002), and the conventional SFS with the feature-knock-out regularization technique (fko-SFS) (Wolf, 2004). SVM RFE, a typical embedded feature selection model, begins with the training of an SVM (of linear kernel) with all the given features. Then according to the parameters of the trained SVM, features are ranked in terms of importance, and half of the features are eliminated. The training-SVM-eliminating-half-of-features process repeats until no feature is left. The feature-knock-out point injection scheme is designed for classification learning in which a point x' is added in each learning iteration. To generate x', two samples (say x_1 and x_2) are randomly selected and a feature f is specified according to the newly-built model. And all information about x' is set with that of x_1, except that $x'(f) = x_2(f)$. We adopt this point injection scheme to modify the conventional SFS as fko-SFS.

To assess the quality of feature selection results, we rely on experimental classification results. In details, given a feature subset for examining, say *S*, certain classifiers are constructed using training data which is also used for feature selection. Then, based on the performance of these classifiers on a test dataset, the quality of *S* is evaluated. Respectable feature subsets should correspond to good classification results. For this evaluation purpose, four typical classifiers are employed. They are multiply percepton model (MLP), support vector machine model with linear kernel (SVM-L), the support vector machine model with RBF kernel (SVM-R) and the 3-NN rule classifier. The MLP used in our study is available at http://www.ncrg.aston.ac.uk/netlab/. For convenience, we set 6 hidden neurons of MLP for all examples. It is worth noting that slightly different number of hidden neurons will not have effect on the overall performance. The number of training cycles is set with 100 in order to ease the concerns on overfitting. And other learning parameters are set with default values. SVM models are available at http://www.isis.ecs.soton.ac.uk/resources/ svminfo.

4.1 Data

Sonar classification. It consists of 208 samples. Each sample is described with 60 features and falls into one of two classes, metal/rock. From 208 samples, 40 ones are randomly selected for training and the others are used for test.

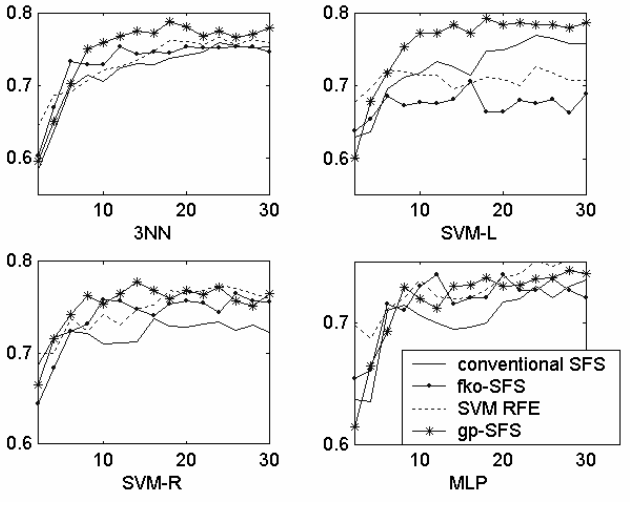
Vehicle classification. This is 4-class dataset for distinguishing the type of vehicle. There are totally 846 samples provided. Each sample is described with 18 features. We randomly select 80 samples for training. The remained 766 samples are used for testing.

Colon tumor classification. This is a microarray data set and is built for colon tumor classification, which contains 62 samples collected from colon-cancer patients (Alon, 1999). Among these samples, 40 samples are tumor, and 22 are labeled "normal". There are 2,000 genes (features) selected based on the confidence in the measured expression levels. We randomly split the 62 samples into two disjoint groups – one group with 31 samples for training and the other one with 31 samples for test.

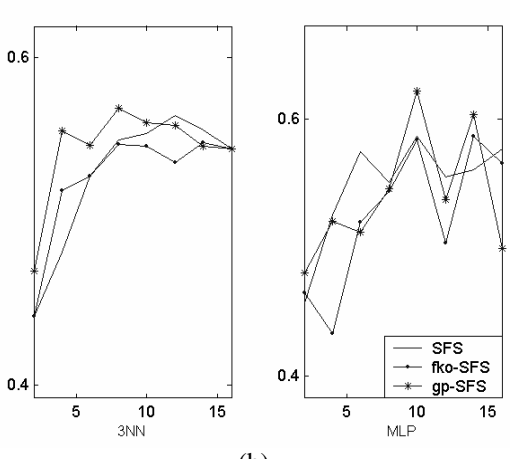
Prostate cancer classification. This is another microarray dataset, which are collected with the aims to prostate cancer cases from non-cancer cases (Singh, 2002). This dataset consists of 102 samples from the same experimental conditions. And each sample is described by using 12600 genes (features). We split the 102 samples into two disjoint groups – one group with 60 samples for training and the other with 42 samples for testing.

4.2 Results

In each example, we repeat investigation on 10 different sets of training and test data. The presented results are the statistics of 10 different trials. Also, in each training data, the original ratios between different classes are roughly remained. For example,



(a)



(b)

Fig. 1. Comparisons on UCI datasets. (a) sonar classification. (b) vehicle classification.

during the investigation on the colon cancer classification, the original ratio between tumor and normal class, i.e., 40 normal vs. 22 tumor, is roughly kept in each training dataset. For each training dataset, we preprocess it so that each input variable has zero means and unit variance. And the same transformation is then applied to the corresponding test dataset.

The computational complexity of SFS type models is $O(M^2)$ where M is the number of features. A microarray dataset generally contains information of thousands or ten thousands genes. Clearly, directly handling the huge gene sets cost SFSs unbearable computational burden. To improve the computational efficiency, and given by the fact that most genes originally given in a microarray dataset are

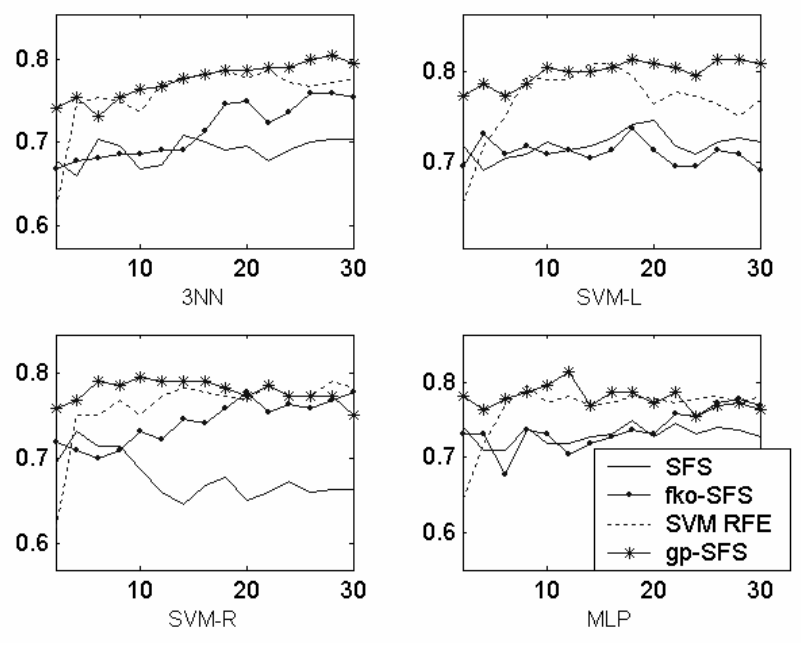


Fig. 2. Comparisons on the colon cancer classification data

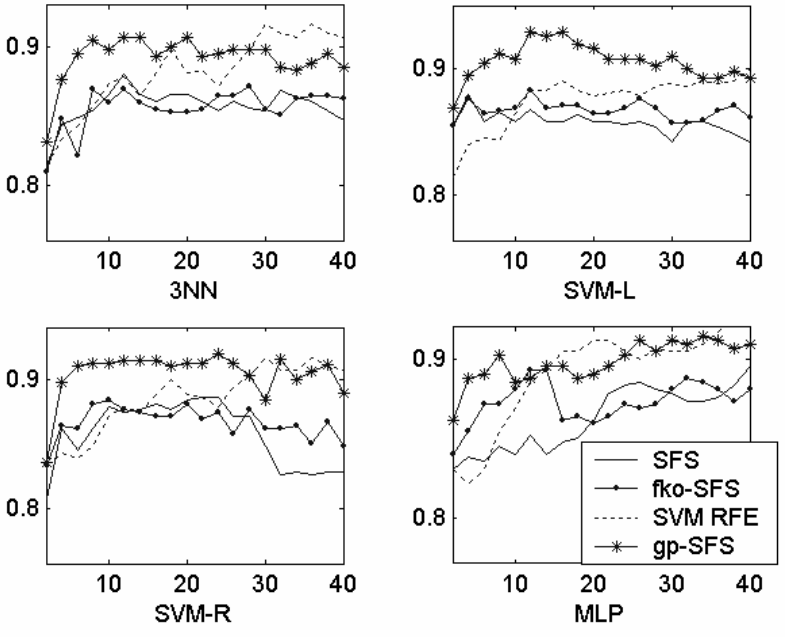


Fig. 3. Comparisons on the prostate cancer classification data

irrelevant to a specified task, a widely used pre-filtering-gene strategy is adopted in our study to eliminate the irrelevant and insignificantly relevant genes before the commencement of feature selection. In details, all the given features (genes) are ranked in a descend order of BD (8). And the one third top-ranked features are left behind for further feature selection.

The comparative results are presented in Figure 1 (for sonar classification and for vehiecle classification), Figure 2 (for colon cancer classification) and Figure 3 (for prostate cancer classification). In most cases, our modified SFS greatly outperform the conventional SFS. This is contributed by the gradient based and point injection strategies. Also, compared with fko-SFS and SVM RFE in which the problem of small-sample is tackled implicitly or explicitly, the proposed SFS still shows its advantages. The contributions of our study can thus be proved.

5 Conclusions

In this paper, two strategies are proposed to enhance the performance of filter feature selection models. The first one is a graident based strategy which is used to enhance the searching effectivenss, and another is a new point-injection approach which is aimed to improve generalization ability. The results obtained on synthetic data and real data obivously demonstrate that these proposed strategies can bring a remarkable improvement. The proposed strategies are only applied to one representative filter model – BD based sequential forward searching. In furture work, we will extend these strategies to other filter models and further evaluate their merits and limitations.

References

Al-Ani A. and Deriche, M. (2000) Optimal feature selection using information maximisation: case of biomedical data, in *Proc. of the 2000 IEEE Signal Processing Society Workshop*, vol. 2, pp. 841-850.

Alon, U. *et al.* (1999) Broad pattern of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. U.S.A.* vol. 96(12), pp. 6745-6750.

Battiti, R. (1994) Using mutual information for selecting features in supervised neural net learning. *IEEE Trans. Neural Networks*, vol. 5, pp537-550.

Bishop, C.M. (1995) *Neural Networks for Pattern Recognition*, New York: Oxford University Press.

Bonnlander, B. (1996) *Nonparametric Selection of Input Variables for Connectionist Learning*, Ph.D. thesis, CU-CS-812-96, University of Colorado at Boulder.

Chow, T. W. S. and Huang, D. (2005) Estimating optimal feature subsets using efficient estimation of high-dimensional mutual information. *IEEE Trans. Neural Networks*, vol. 16, no. 1, pp. 213-224, January 2005.

Devijver, P. A. and Kittler, J. (1982) *Pattern Recognition: a Statistical Approach*, Englewood Cliffs: Prentice Hall.

Golub, T. R. *et al.* (1999) Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286, pp. 531-537.

Glick, N. (1985) Additive estimators for probabilities of correct classification, *Pattern recognition*, vol. 18 (2), pp. 151-159.

Gui, J. and Li, H. (2005) Penalized Cox regression analysis in the high-dimensional and low-sample size settings, with application to microarray gene expression data, *Bioinformatics*, 21(13), pp. 3001-3008.

Guyon, I., Weston J. and Barnhill, S. (2002) Gene selection for cancer classification using support vector machines, *Machine Learning*, vol. 46, pp. 389-422.

Hall, M. A. (1999) *Correlation-based Feature Selection for Machine Learning*, Ph.D. thesis, Department of Computer Science, Waikato University, New Zealand.

Han, J. W. and Kamber, M. (2001) *Data mining: concepts and techniques*. San Francisco: Morgan Kaufmann Publishers, 2001.

Hastie,T., Tibshirani,R. and Friedman,J. (2001) *The Elements of Statistical Learning*, pp. 308-312, Springe.

Huang, D. and Chow, T.W.S. (2005) Efficiently searching the important input variables using Bayesian discriminant*, IEEE Trans. Circuits and Systems*, vol. 52(4), pp.785-793.

Huang, D. , Chow, T.W.S. *et al.* (2005) Efficient selection of salient features from microarray gene expression data for cancer diagnosis, *IEEE Trans. Circuits and Systems, part I*, vol. 52 (9), pp.1909-1918.

Kim, S., Dougherty, E.R, Barrera, J.Y. *et al*. (2002) Strong feature sets from small samples, *Journal of Computational Biology,* 9, pp.127-146.

Lampariello, F. and Sciandrone, M. (2001) Efficient training of RBF neural networks for pattern recognition, *IEEE Trans. On Neural Networks*, vol. 12(5), pp.1235-1242.

Liu, H. and Motoda, H. (1998) *Feature Selection for Knowledge Discovery and Data Mining*, London, GB: Kluwer Academic Publishers.

Matsuoka, S. (1992) Noise injection into inputs in back-propagation learning, *IEEE Trans. Syst., Man, Cybern.,* vol. 22, pp. 436-440.

Molina, L. C., Belanche L. and Nebot, A. (2002) *Feature Selection Algorithms: a Survey and Experimental Evaluation*, available at: http://www.lsi.upc.es/dept/techreps /html/R02-62.html, Technical Report.

Parzen, E. (1962) On the estimation of a probability density function and mode, *Ann. Math. Statistics.,* vol. 33, pp. 1064-1076.

Perkins, S., Lacker K., Theiler, J. (2003) Grafting: Fast, Incremental feature selection by gradient descent in function space, *Journal of machine learning research*, vol. 3, pp. 1333-1356.

Pudil, P., Novovicova, J. and Kittler, J. (1994) Floating search methods in feature selection. *Pattern Recognition Letter*, vol. 15, pp. 1119-1125, Nov. 1994.

Singh, D. *et al.* (2002) Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, vol. 1, pp. 203-209.

Skurichina, M, Raudys, S. and Duin, R.P. (2000) K-nearest neighbours directed noise injection in multilayer perceptron training, *IEEE Trans. On Neural Networks,* vol. 11(2), pp. 504-511.

Wolf, L. and Martin, I. (2004) Regularization through feature knock out, AI memo 2004-2005, available at http://cbcl.mit.edu/cbcl/publications/ai-publications/2004/.

Zagoruiko, N. G., Elkina, V. N. and Temirkaev, V. S. (1976) ZET-an algorithm of filling gaps in experimental data tables. Comput. Syst. vol. 67, pp. 3-28.

Zhou, X., Wang X. and Dougherty, E. (2004) Nonlinear probit gene classification using mutual information and wavelet-based feature selection, *Journal of Biological Systems*, vol. 12 (3), pp. 371-386.