Adaptive Credit Scoring with Kernel Learning Methods

Dr Yingxu Yang

SHS Informationssystems
Fraunhoferstr. 12, 82152 Martinsried, Munich, Germany
yingxu.yang@shs.de

Abstract

Credit scoring is a method of modelling potential risk of credit applications. Traditionally, logistic regression, linear regression and discriminant analysis are the most popular approaches for building credit scoring models. Despite their popularity, quite a few limitations are known to be associated with these methods, such as instability with high-dimensional data (also known as combinatorial explosion) and small sample size, requirement for intensive data pre-processing through variable selection/reduction analysis and incapability of efficiently handling non-linear components. Most importantly, based on these algorithms, it is difficult to automate the modelling process and design a continuous workflow. When environment or population changes occur, the static models usually fail to adapt and may need to be rebuilt from scratch.

In this paper, a kernel learning method is used to derive a novel and practical adaptive credit scoring system capable of adjusting the model on-line. The kernel method is based on a two-layered structure (α form) and provides a powerful solution for non-linear systems by plugging in a kernel that maps patterns from non-linear input space to potentially high-dimensional feature space. A new kernel, the KGPF kernel, has been introduced and shown to be able to generate comparable results to those from the popular Gaussian kernel. The model is adjusted according to an on-line update rule and can always converge to the optimal solution. This approach is also robust for scoring data sets with a large number of attributes and does not require any variable selection or reduction effort. Experimental studies have demonstrated the effectiveness of the proposed approach.

I. Introduction

Credit scoring is a method of modelling potential risk of credit applications. In general it employs statistical techniques and historical data to produce a score that financial institutions can use to evaluate credit applicants in terms of risk. This helps them classify applicants into “good” and “bad” risk groups and assists risk managers to make risk related decisions. Essentially the problem credit scoring is trying to solve can be categorised into general population classification tasks. The statistical principle for discriminating different groups in a population can be traced back to 1936 in Fisher’s publication [2]. Durand [3] was the first to apply this technique to discriminate between good and bad loans. Based on the research development in this area, the founders of credit scoring, Bill Fair and Earl Isaac built the first credit

scoring system for American Investments in 1958. Although credit scoring has been used since then, it only became widely spread in the last two decades, with application areas expanding from customer lending business, small business loans and residential mortgages to direct marketing, corporate failure prediction, telecom risk management [5] and so on.

Credit institutions usually need to make two types of decisions. The first is, should a credit application be approved? Scoring approaches that help answer this type of questions are generally named as application scoring. The second is related to existing customers and is distinguished from the first type because this reflects the customer's payment pattern on the loan. The credit scoring that falls into this category is called behaviour scoring. The remaining sections of this paper will be focused on application scoring.

The most popular classification methods adopted in the credit scoring industry are linear discriminant analysis and logistic regression. They are relatively easy to understand and implement, and are able to generate straightforward results that can be readily interpreted. Nonetheless there are quite a few known limitations associated with their application in credit scoring. First of all, they entail intensive data pre-processing effort through manual variable selection/reduction analysis. This could be time-consuming and usually require domain expert knowledge and in-depth understanding of the data. In addition, these methods are not effective for systems with high-dimensional inputs (combinatorial explosion) and small sample size (e.g. new creditors). In real world application, assumptions regarding the data must be held, such as being linear separable and that the data should follow certain distributions. Most importantly, based on these algorithms, it is difficult to automate the modelling process and design a continuous workflow. When environment or population changes occur, the static models usually fail to adapt and may need to be rebuilt from scratch.

An adaptive credit risk modelling method is proposed in this paper to deal with the difficulties mentioned above. This method is based on a non-linear kernel mapping and an iterative constraint optimisation procedure. It considers the contributions of all the attributes simultaneously and does not require any variable selection effort. This allows to design a dynamic credit scoring process where the decision model can be updated and improved with the arrival of new customer information.

The rest of the paper is structured as follows. In section II, the conventional credit scoring methods, linear discriminant analysis and logistic regression are briefly introduced followed by a short discussion on their limitations. Section III focuses on non-linear scoring model development using the adaptive kernel learning method. Experimental studies using real world credit scoring data are presented in section IV where the performance of a new kernel, the KGPF kernel is also reported. Finally concluding remarks are given in section V.

II. Traditional methods: linear discriminant analysis and logistic regression

Traditionally, linear discriminant analysis and logistic regression methods have been the most widely used techniques in building credit scoring models. These two

methods are reviewed in the following subsections with a short discussion on their drawbacks when applied in credit scoring.

II.1. Linear discriminant analysis

Linear discriminant analysis is a technique for classifying a set of input vectors into predefined classes based on the linear combination of the input variables. The linear discrimination rule is given by:

$$f(x_i) = +1 \quad \text{if } w_0 + w_1 x_{i1} + \cdots + w_d x_{id} > 0$$
$$f(x_i) = -1 \quad \text{otherwise}$$

The weight parameters w_i $(i = 0, \cdots, d)$ can be estimated by maximizing the following objective function:

$$F(w) = \frac{w^T M_B w}{w^T M_W w} \tag{2.1}$$

where $w = [w_0 \cdots w_d]$, M_B is the scatter matrix between different classes and M_W is the scatter matrix within the same class. The two scatter matrices are defined as:

$$M_B = \sum_{s=1}^{2} N_s (\mu_s - \bar{x})(\mu_s - \bar{x})^T$$
$$M_W = \sum_{s=1}^{2} \sum_{i \in s} (x_i - \mu_s)(x_i - \mu_s)^T$$

where

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \quad \text{(overall mean)}$$
$$\mu_s = \frac{1}{N_s} \sum_{i=1}^{N_s} x_i \quad (s = 1, 2) \quad \text{(class mean)}$$

and N is the number of all patterns, N_1 is the number of patterns in the class with label +1; N_2 is the number of patterns in the class with label –1.

Maximizing function (2.1) is equivalent to minimizing the following cost function with equality constraint:

$$-\frac{1}{2} w^T S_B w = 1$$
$$\text{subject to} \quad w^T S_W w = 1$$

The corresponding Lagrangian is constructed as:

$$L(\lambda, w) = -\frac{1}{2} w^T S_B w + \frac{1}{2} \lambda (w^T S_W w - 1)$$

Enforcing the KKT conditions gives:

$$S_B w = \lambda S_W w \tag{2.2}$$

This can finally be solved as a generalized eigenvalue problem.

Linear discriminant analysis implicitly assumes that each class follows a Gaussian distribution with a different mean vector but a common covariance matrix, and that the discriminating factor is the mean, not the variance.

II.2. Logistic regression

Logistic regression is considered to be a special class of linear regression and is currently one of the most popular methods in building credit scoring models. This type of model assumes that the input variables time their weights are linearly correlated to the natural log of the odds that the outcome event will happen (for example, the odds that a loan is going to default). While linear regression may calculate a probability that is out of the [0,1] interval, logistic regression can avoid this by taking the natural log of the odds.

Logistic regression uses the following decision model:

$$\ln(\frac{p}{1-p}) = w_0 + w_1 x_1 + \cdots + w_d x_d \qquad (2.3)$$

where p is the probability that the outcome y will occur and $\frac{p}{1-p}$ is the odds that y will occur. Maximum likelihood can be applied to compute the estimate of $w_i, \{i = 0,..., d\}$ given the historical data vectors x and y. This modelling technique solves a concave optimisation problem and is numerically robust, which has doubtlessly contributed to the success and popularity of logistic regression. Assume the regression model in (2.3) is obtained, the total score can be calculated using

$$\frac{M}{\ln(2)}(w_1 x_1 + \cdots w_d x_d)$$

where M represents the expected change in the score when the odds that y will happen doubles and is usually defined by the model developer. The probability p can be computed using

$$p = \frac{1}{1+\exp(-w_0 - w_1 x_1 - \cdots - w_d x_d)}$$

Although not necessary, some credit scoring model users prefer to divide the values of each input attribute into a number of discrete intervals. This can be implemented by creating a separate 0/1 categorical variable for each group of each attribute and including all but one of the variables corresponding to the same attribute in the model.

Note that the structure of a logistic regression model is equivalent to that of a single layer perceptron with a *logsig* output function. It is obvious that the *logit* function $y = \ln(\frac{x}{1-x})$ in logistic regression is the inverse of the *logsig* function $y = \frac{1}{1+\exp(-x)}$. The difference between these two methods is that logistic regression employs maximum likelihood estimation to calculate the weight parameters, while the single layer perceptron uses Rosenblatt's perceptron learning rule [6].

Similar to linear discriminant analysis, logistic regression is intrinsically a linear classification method and is most suitable for linearly separable data.

II.3. Drawbacks of the present models for developing an adaptive credit scoring system

Both methods described above result in linear models. To consider non-linear effects in real world applications, very often a variable combination is manually created and entered into the model on the basis of the knowledge or assumption of the model developer. A number of this type of variables are included and the resulting model performances from different combinations are compared before a decision is made regarding which ones should finally be in the model. This is close to an arbitrary trial and error process and the number of various combinations can grow astronomically even with a reasonable increase in the number of input variables. In addition, these linear models are sensitive to redundancy or collinearity in the input variables. When the variable selection/reduction process fails to remove these correlations, the model runs into the risk of over-fitting as well as requires an excessive amount of computation time.

With conventional credit scoring, usually more than fifty percent of the time and effort is spent for data pre-processing and variable selection/creation. This is a highly manual process requiring significant amount of domain expert knowledge and is prone to errors introduced through subjective interpretation of the data. Typically a scoring model is static and is not updated on a regular basis (time between the development and update can be months or years) given the re-development expenses or company policies. This is particularly challenging for new creditors who may not have enough matured data at the time of development and later with more and more new information becoming available need to update the model to improve predictive accuracy. Ideally, a dynamic decision model should be created which is able to continuously adapt to changes both in the environment and the system, and in real time. The modelling process should have a much higher degree of automation with minimum manual adjustment.

It is difficult to develop an adaptive credit scoring model based on the current linear methods due to their aforementioned constraints. The following section describes a dynamic modelling process using state-of-the-art kernel learning techniques.

III. Adaptive Credit Scoring with Kernel Learning Methods

One of the most powerful kernel learning method in machine learning is the kernel large margin classifier: the support vector machine (SVM) [7,12]. In the last decade there has been a surge of interest in SVM's in many different application areas [8-11] and they are shown to be able to produce good generalization performance.

III.1. Kernel Learning Methods

This paper focuses on SVM's with L_1 soft margin and two C penalty parameters to deal with non-separable case and imbalanced data. In the linear case without bias, this is formulised as the following constraint optimisation problem:

$$\textbf{min}\ \ \frac{1}{2}\| w\|^2 + C_{-1}\sum_{i\in S_{-1}}^{N_{-1}}\xi_i + C_{+1}\sum_{i\in S_{+1}}^{N_{+1}}\xi_i \qquad (3.1)$$

$$\text{subject to } y_i(w\cdot x_i)\geq 1-\xi_i$$

where $y_i \in \{-1,+1\}$ is the class label for pattern x_i, w is the weight vector, ξ_i's are slack variables that allow margin failure, S_{-1} is the set of patterns with class label –1 and C_{-1} is the corresponding penalty parameter indicating a trade off between large margin and a small number of margin failures. S_{+1} contains the set of patterns with class label +1 and C_{+1} is the corresponding penalty parameter. N_{-1} and N_{+1} are the number of the patterns in S_{-1} and S_{+1} respectively.

After constructing a Lagrangian and enforcing the KKT conditions, the optimisation problem can be transformed into a dual form. This Lagrange dual is a typical QP problem and the dual cost function is dependent on the Lagrange multipliers α_i:

$$\textbf{min}\ D(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N}\alpha_i \qquad (3.2)$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C_{-1},\ i \in S_{-1}$$

$$0 \leq \alpha_i \leq C_{+1},\ i \in S_{+1}$$

where N is the total number of training patterns. The weight vector is expressed as:

$$w = \sum_{i=1}^{N}\alpha_i y_i x_i \qquad (3.3)$$

The objective function in (3.2) can be extended to the non-linear case by replacing the linear dot product using a non-linear kernel. This can then be written as:

$$\textbf{min}\ D(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{N}\alpha_i \qquad (3.4)$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C_{-1},\ i \in S_{-1}$$

$$0 \leq \alpha_i \leq C_{+1},\ i \in S_{+1}$$

Note that the introduced non-linearity does not change the fact that the Lagrange dual is quadratic in α_i and that these Lagrange multipliers can still be calculated using a quadratic program. To ensure that the kernel matrix is positive definite, the kernel function K must satisfy Mercer's conditions [12].

The non-linear decision function is finally given as:

$$f(x_p) = \sum_{i=1}^{N}\alpha_i y_i K(x_i, x_p) \qquad (3.5)$$

$$\hat{y}(x_p) = \text{sgn}(f(x_p))$$

where $\hat{y}(x_p)$ is the predicted class label for pattern x_p.

The equation in (3.3) can be considered as a link between the "perceptron representation" [6] (α form) and explicit (w) form of the decision function. In the linear case, the weight vector w can be back calculated using equation (3.3) after obtaining α_i. However for non-linear kernels, it is difficult to retrieve $\phi(x)$ given

$\phi(x_i)\phi(x_j) = K(x_i, x_j)$. The corresponding w expression therefore only exists in the feature space and is theoretically written as:

$$w = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i)$$

III.2. Adaptive Scoring

One of the most important properties an adaptive credit scoring system should posses is that the model can be updated incrementally in real time. Traditional scoring methods require that a manual variable reduction/creation process based on trial and error is repeated for each update after model development. This makes them not suitable for adaptive system. In contrast, the kernel approach allows all the attributes in the scoring model to be simultaneously considered without having to independently analyse each variable or select a smaller subset from all the candidates. The decision value calculated using equation (3.5) is in fact a linear combination of similarity measures in the non-linear feature space. The non-linearity in the data is represented in the model through a concise kernel mapping instead of arbitrary inclusion of variable combinations. This helps reduce the amount of manual adjustment and increase the degree of automation in model development, and enables kernel learning to be an ideal candidate for the adaptive credit scoring system.

The focus of adaptive scoring is to obtain a model update rule given the existing model and the new customer data. For kernel learning, this should ideally be expressed as:

$$\alpha_{i,new} = \alpha_{i,old} + \Delta\alpha_i \qquad (3.6)$$

Originally, however, the model parameters α_i, $i \in \{1, \cdots, N\}$ are computed by running a classical QP routine, which cannot be easily modified to derive an adaptive form. Taking a closer look at the quadratic cost function in equation (3.4) reveals that although it is non-linear in the input patterns x_i, the parameters α_i can be iteratively calculated using linear modelling techniques [15,16] including constraint least squares, non-negative conjugate gradient method and so on. In particular, the change in the model parameter $\Delta\alpha_i$ can be computed using derivatives of the Lagrange dual and through simultaneously enforcing the constraints:

$$0 \le \alpha_i \le C_{-1},\ i \in S_{-1}$$
$$0 \le \alpha_i \le C_{+1},\ i \in S_{+1}$$

However with linear online methods, the number of parameters to be estimated usually remains the same when new training patterns are added. This is different for kernel learning where each new pattern introduces a new parameter. The question now is how to update the existing α_i, $i \in \{1, \cdots, N\}$ and compute the new α_u, given the current model f_{old} in equation (3.5) and the new pattern x_u with its class label y_u. Before answering this question, the KKT (Karush Kuhn Tucker) conditions for the Lagrange dual need to be discussed.

The KKT conditions specify necessary and sufficient conditions for an optimal solution to a QP problem. For the objective function in equation (3.4) these are defined as follows:

$$\frac{\partial D}{\partial \alpha_i} = y_i f(x_i) - 1$$

$$\Downarrow$$

$$y_i f(x_i) - 1 \le 0 \Leftrightarrow \alpha_i = C_{-1} \ (\text{for } y_i = -1) \ \text{ or } \ \alpha_i = C_{+1} \ (\text{for } y_i = +1)$$
$$y_i f(x_i) - 1 = 0 \Leftrightarrow 0 < \alpha_i < C_{-1} \ (\text{for } y_i = -1) \ \text{ or } \ 0 < \alpha_i < C_{+1} \ (\text{for } y_i = +1)$$
$$y_i f(x_i) - 1 \ge 0 \Leftrightarrow \alpha_i = 0$$

An adaptive update procedure can be derived by evaluating the KKT conditions on each new pattern:

1. Use the existing model f_{old} to calculate the decision value for the new pattern x_u and evaluate $y_u f_{old}(x_u)$.
2. If the KKT conditions are satisfied, then the new pattern does not participate in forming the decision boundary. It is not necessary to update the current model. However keep the new sample in the pattern pool for possible update at a later stage.
3. If this pattern violates the KKT conditions, then compute the new parameter $\alpha_u = \Delta\alpha_u$. The response in equation (3.5) is recalculated as
   $f_{new}(x_p) = \sum_{i=1}^{N} y_i \alpha_i K(x_i, x_p) + y_u \alpha_u K(x_u, x_p)$.
4. Update the existing parameters α_i, $i \in \{1, \cdots, N\}$ using f_{new} and the formula in (3.6).
5. Update the new parameter α_u using f_{new}.
6. Repeat steps 4 and 5 until all the parameters α_i, $i \in \{1, \cdots, N, u\}$ converge.

This update procedure does not differentiate between training patterns bounded at $(0, C_{-1}, C_{+1})$ and those within the bounds, and might be slow with large training sets. To speed up the learning process, the following update rule can be implemented. It is based on the heuristics to choose multipliers for optimisation in [13].

1. Use the existing model f_{old} to calculate the decision value for the new pattern x_u and evaluate $y_u f_{old}(x_u)$.
2. If the KKT conditions are satisfied, then the new pattern does not participate in forming the decision boundary. It is not necessary to update the current model. However keep the new sample in the pattern pool for possible update at a later stage.
3. If this pattern violates the KKT conditions, then compute the new parameter $\alpha_u = \Delta\alpha_u$. The response in equation (3.5) is recalculated as
   $f_{new}(x_p) = \sum_{i=1}^{N} y_i \alpha_i K(x_i, x_p) + y_u \alpha_u K(x_u, x_p)$.
4. Iterate through all (x_i, y_i), $i \in \{1, \cdots, N, u\}$ and check if the corresponding response violates the KKT conditions. Put the data that violate the KKT conditions into a separate set named as S_v.

5. Update the corresponding parameters for patterns in S_v using f_{new} and the formula in (3.6).
6. Iterate through S_v, identify non-bound patterns and update the corresponding parameters.
7. Repeat step 6 until all the non-bound patterns satisfy the KKT conditions. At this point, go back to step 4 and repeat steps 4-7 until all the patterns in the training set satisfy the KKT conditions.

Both procedures provide the flexibility to adjust the model with each new pattern. These can be easily extended to updating the model using a set of new patterns (instead of each single pattern) with a user-defined set size. For simplicity this is not detailed in the paper.

III.3. Model Interpretation

One of the reasons that linear classification methods are popular in credit scoring is that the decision models can be easily explained. The contribution of each attribute is clearly shown in the associated weight parameter vector w. As discussed earlier in section IV.1, although it is possible to derive the w form decision function for linear kernels, the non-linear kernel model cannot be explicitly expressed in the w form in the input space. In particular for RBF and KGPF kernels, both the mapping $\phi(x)$ and the w vector ($w = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i)$) are infinite dimensional. However, would it then be useful to consider contribution measures based on w, for example, the impact an attribute has on $\frac{1}{2}\| w \|^2$?

It can be seen from equation (3.3) that

$$\frac{1}{2}\| w \|^2 = \frac{1}{2}(\sum_{i=1}^{N} \alpha_i y_i x_i)^2 = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \qquad (3.7)$$

For non-linear cases, this is written as:

$$\begin{aligned}\frac{1}{2}\| w \|^2 &= \frac{1}{2}(\sum_{i=1}^{N} \alpha_i y_i \phi(x_i))^2 = \frac{1}{2}\sum_{i=1}^{N} \alpha_i y_i \phi(x_i) \sum_{j=1}^{N} \alpha_j y_j \phi(x_j) \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \phi(x_i)\cdot\phi(x_j) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j)\end{aligned} \qquad (3.8)$$

This indicates that although the weight vector w itself is not directly applicable in non-linear models, the corresponding quadratic form can be used to explain the contribution of each attribute. The impact an attribute θ_t has on the outcome variable is reflected in the change between the $\frac{1}{2}\| w \|^2$ computed using all the attributes and that using all the attributes except θ_t. A feature ranking measure can then be given as:

$$R(\theta_t) = \frac{1}{2}\| w \|^2 - \frac{1}{2}\| w_- \|^2 = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (K(x_i, x_j) - K_-(x_i, x_j)) \qquad (3.9)$$

Note that this is also interpreted as the contribution θ_t made to the dual cost function [14].

The rank computation time grows linearly with the number of attributes. This could be relatively expensive for scoring systems with a large number of variables. However the ranking measure is based on the computed model after training. Calculating the changes in kernel values is therefore only necessary for patterns with non-zero α (the support vectors). This makes it particularly suitable for models with a relatively small number of support vectors.

IV. Experimental Studies

The adaptive credit scoring procedure has been tested using a real world data set provided by a German creditor and the results are presented in this section. Two kernel functions: RBF and KGPF are used in the experiments and have performed differently. Based on the computed model, all the 102 attributes are ranked according to their contribution to the quadratic w measure. The ranked list is provided in the Appendix.

The data set used for the experiments contains about 6000 samples, 102 attributes and a binary outcome (Good/Bad). The two classes are well balanced with Good/Bad ratio 52%-48%. For simplicity a description of the data is not given in this paper. The Good/Bad class labels are coded as +1 and –1 respectively. All the attributes are linearly scaled to [0, 1] to avoid the dominance of attributes with greater numeric values over those with smaller values. No variable analysis/selection/reduction techniques have been applied to this data set. No domain expert knowledge is known.

The whole data set is randomly shuffled 100 times and split into two sets: development (2/3) and test (1/3). The development set is again divided into batch learning (the first 2000 samples) and on-line learning sets (the rest). To choose the suitable kernel parameter and soft margin penalty parameters 10-fold cross training-validation has been conducted on the batch development data. After fixing these parameters, the whole development set is used for training (first batch then on-line) and the results are finally validated on the test.

IV.1. RBF Kernel Results

Firstly, an RBF potential function [12] is used as the kernel function:

$$K(x_i, x_j) = \exp\left(\frac{-\| x_i - x_j \|^2}{2\sigma^2}\right)$$

Because the Good/Bad ratio is well balanced, the soft margin penalty parameters are set to be the same: $C_{-1} = C_{+1}$. The 10-fold cross validation on the batch development set chooses the following parameter combination: $\sigma = 5.66$, $C_{-1} = 20$, $C_{+1} = 20$. The scoring model is then re-trained on the whole development set (first batch then on-line) and tested on the test set. The following confusion matrices are produced for development (δ) and test (τ) respectively:

$$\delta = \begin{bmatrix} 1580 & 449 \\ 1609 & 362 \end{bmatrix} \qquad \tau = \begin{bmatrix} 745 & 208 \\ 783 & 185 \end{bmatrix}$$

The α and β errors are 22.13% and 18.37% for development, and 21.83% and 19.11% for test. Without discarding intermediate non-support-vectors or chunking implementation, these results are the same as those produced using only batch learning.

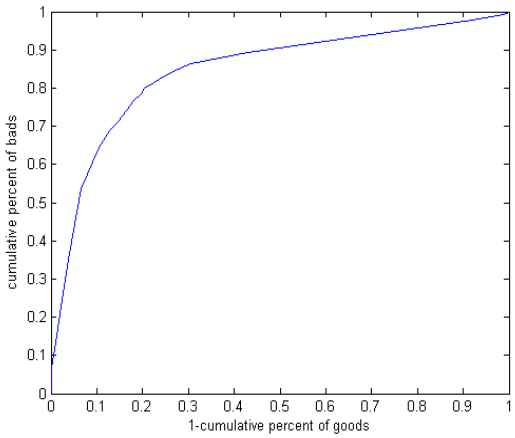

Figure 4.1 ROC curve (RBF)

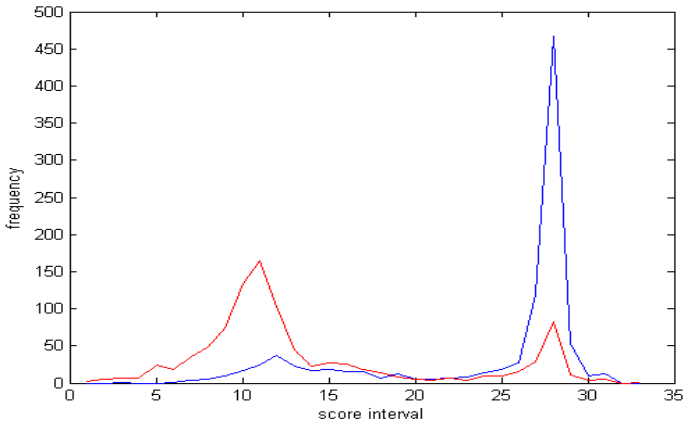

Figure 4.2 Frequency distribution of goods and bads (RBF)

Figure 4.1 shows the Receiver Operating Characteristic (ROC) curve on the test set. The area under this curve is a measure of how well the obtained model separates the good customers from the bad ones: the faster the curve rises to reach the "1"s , the better the separation. This measure uses information in the entire good/bad distributions and summarizes the capability of the model to assign relatively more low scores to bad than to good applications. Figure 4.2 displays the frequency distribution of the good (blue) and bad (red) customers. The score interval is scaled by a factor of 25. Similar to ROC curve, these two plots are used to visually evaluate the separation performance of the model: the further away they are apart, the better the separation.

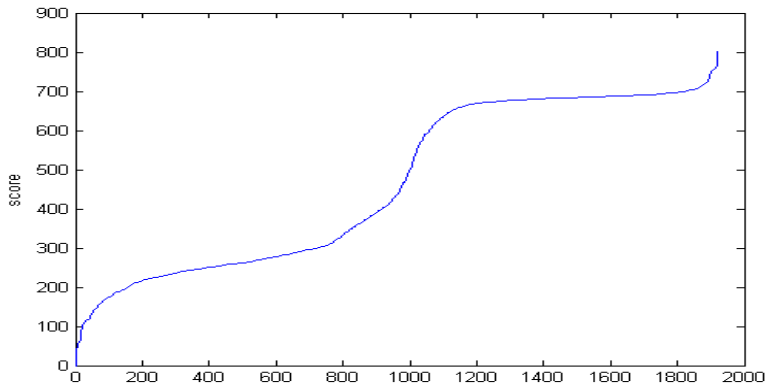

Figure 4.3 Score distribution (RBF)

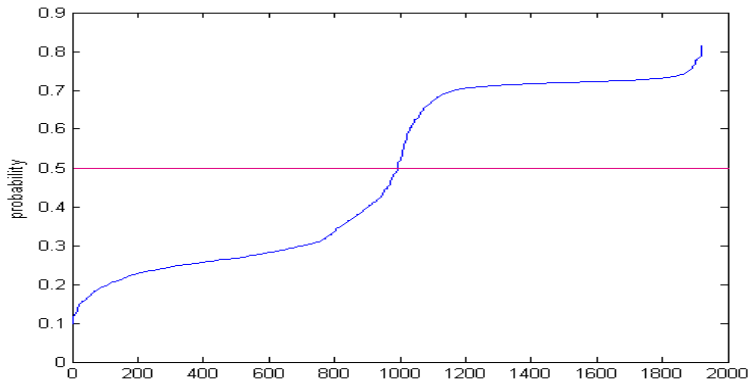

Figure 4.4 Probability distribution (RBF)

The score distribution on the validation set is shown in Figure 4.3, where the M parameter for score calculation from the response value is defined as 150. The corresponding probability distribution in Figure 4.4 displays almost the same shape as the curve in Figure 4.3. This is correct because both are computed using the model response value. Note that the probability indicates level of confidence in the prediction, the further away the value is from 0.5, the more trustworthy the prediction.

Based on the obtained kernel model, the input variables are ranked according to their contributions to $\frac{1}{2}\| w \|^2$. The results are presented in the Appendix.

It can be seen from the confusion matrices produced using the RBF kernel that the alpha and beta errors are slightly imbalanced. To equalize this difference, an alternative kernel is introduced and tested using the same development and validation sets.

IV.2. KGPF Kernel Results

In some cases the performance of a kernel model can be improved by using a different kernel function. This subsection uses another bell-shaped kernel, KGPF, for the non-linear mapping. This function is one of the so called “fat tail” kernels adopted in risk analysis in financial time series modelling to deal with extreme data. Figure

4.5 shows the difference between the two bell shapes, where the KGPF response falls with less momentum to zero than RBF, hence the name “fat tail”.

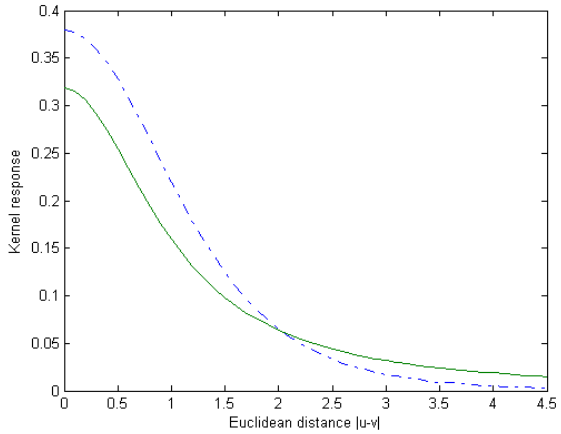

Figure 4.5 Comparison between RBF and KGPF kernels (-˙ RBF, - KGPF)

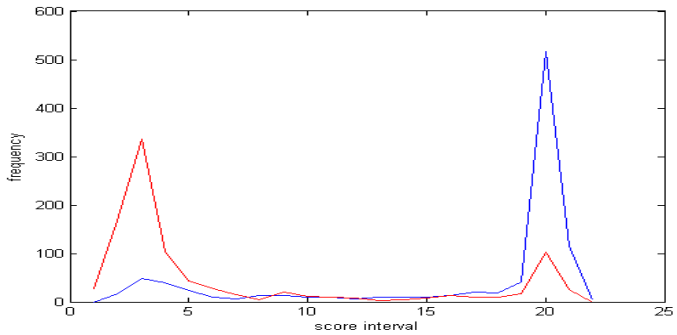

Figure 4.6 Frequency distribution of goods and bads (KGPF)

The experiments in the previous subsection are repeated using the new kernel function. The kernel parameter k is chosen as 2.001 and the soft margin penalty parameters are again set to be $C_{-1} = 20$, $C_{+1} = 20$ after carrying out the 10-fold cross training-validation on the development data. Finally the confusion matrices generated for the development (δ) and test (τ) sets are as follows:

$$\delta = \begin{bmatrix} 1641 & 388 \\ 1588 & 383 \end{bmatrix} \qquad \tau = \begin{bmatrix} 760 & 193 \\ 771 & 197 \end{bmatrix}$$

The alpha and beta errors are 19.1% and 19.4% for development, and 20.3% and 20.4% for test. The alpha and beta errors are now much more balanced and close to being equal on the test set. This improvement is also reflected in the frequency distribution plots in Figure 4.6. The overall classification error has also been slightly reduced. Examining the present support vector set shows that there is an increase of 20% in the number of support vectors compared to the RBF model. Figure 4.9 shows that more than 60% of the alpha parameters are bounded at 0 or $C_{-1} = C_{+1} = 20$ (with a small discrepancy of $e = 0.001$).

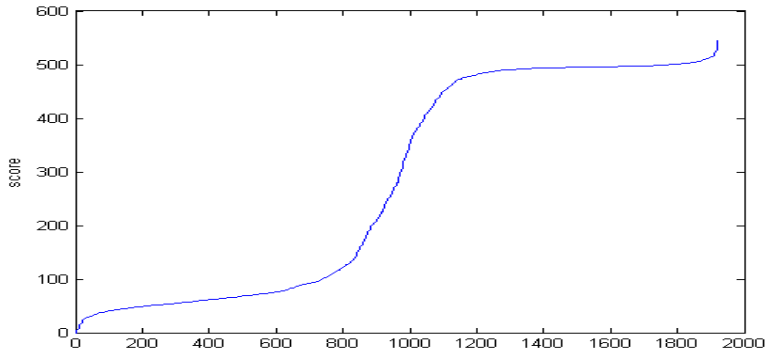

Figure 4.7 Score distribution (KGPF)

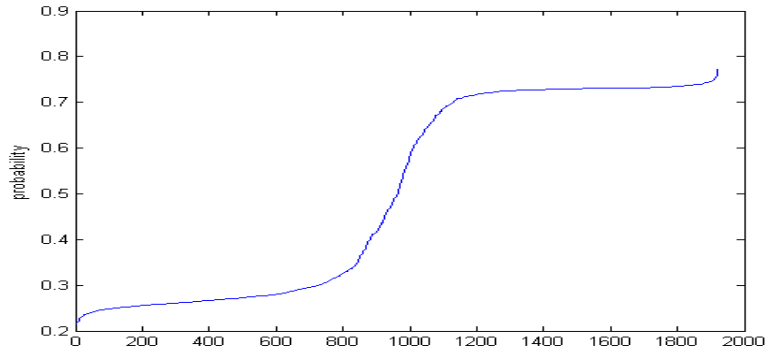

Figure 4.8 Probability distribution (KGPF)

Figures 4.7 and 4.8 display a shrink in the range of both the score and probability distributions in comparison to those from RBF kernel. The ROC curve stays however almost unchanged (not shown here).

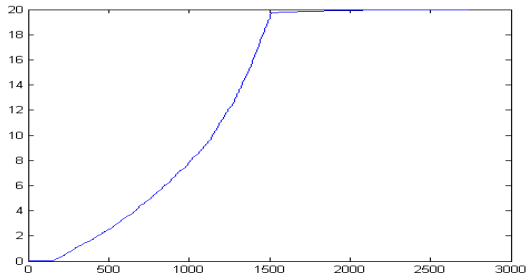

Figure 4.9 Alpha distribution

V. Conclusions

In this paper, a new adaptive credit scoring technique has been presented on the basis of a kernel learning algorithm. This method provides an effective solution to real world non-linear scoring tasks and helps reduce the time, effort and expenses for data pre-processing and variable analysis. The on-line update procedure allows each (set of) new information to be incrementally added in the scoring model as the data become available. The predictive performance of the adaptive model increases with time, while that of a traditional static model deteriorates gradually. Experimental studies using real world credit scoring data show that the results generated using the adaptive procedure are comparable to that using a batch method where all the data samples are presented to the learning system at once. Experimental results also demonstrate the effectiveness of a potential kernel function KGPF, with which better results have been produced than with the RBF kernel using the real data. The adaptive

approach is particularly useful for new creditors that have limited number of matured samples at hand at the time of model development.

References

[1] Mays, E. (2004). Credit Scoring for Risk Managers, The handbook for Lenders, Thomson Learning.

[2] Fisher, R.A. (1936). The use of multiple measurements in taxonomic problems, Annals of Eugenics, vol.7, pp.179-188.

[3] Durand, D. (1941). Risk elements in consumer instalment financing, National Bureau of Economic Research, New York.

[4] Back, B. et al (1996). Choosing Bankruptcy Predictors Using Discriminant Analysis, Logit Analysis, and Genetic Algorithms, Proceedings of the 1st International Meeting on Artificial Intelligence in Accounting, Finance and Tax, pp. 337-356.

[5] A Fair Issac White Paper (2004). Telecom Risk Management.

[6] Rosenblatt, F. (1962). Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms, Washington, DC: Spartan.

[7] Cortes,C. and Vapnik,V. (1995). Support Vector Networks. Machine Learning, vol.20, pp.273-297.

[8] Bomhardt, C. (2004). NewsRec, a SVM-driven Personal Recommendation System for News Websites. In: Web Intelligence, IEEE/WIC/ACM International Conference on (WI'04)

[9] Kwang, I. K. et al (2002). Support Vector Machines for Texture Classification, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no.11.

[10] Dror, G. et al (2005). Accurate identification of alternatively spliced exons using Support Vector Machine, Bioinformatics, 21(7), pp.897-901.

[11] Osuna, E. et al (1997). Training Support Vector Machines: An Application to Face Detection, Proc. Computer Vision and Pattern Recognition'97, pp.130-136.

[12] Aizerman, M. et al (1964). Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control, vol. 25, pp. 821—837.

[13] Platt, J.C. (1998). Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Technical Report MSR-TR-98-14.

[14] Guyon, I. etc (2000). Gene Selection for Cancer Classification Using Support Vector Machines. Machine Learning.

[15] Yang, Y.X. and Billings, S.A. (2000). Neighbourhood Detection and Rule Selection. IEEE Transactions on Systems, Man and Cybernetics Pt. A, vol.30, no.6, pp.840-847.

[16] Yang, Y.X. and Billings, S.A. (2000). Extracting Boolean Rules from Cellular Automata Patterns. IEEE Transactions on Systems, Man and Cybernetics Pt. B, vol.30, no.4, pp.573-580.

Appendix

Attribute ranking according to the quadratic w measure

Rank	VarIdx	*R*%	Rank	VarIdx	*R*%	Rank	VarIdx	*R*%
102	69	0.071381	68	31	0.85896	34	16	2.2731
101	64	0.095112	67	32	0.85908	33	13	2.3017
100	67	0.095229	66	97	0.88588	32	94	2.3346
99	71	0.17748	65	83	0.88588	31	14	2.436
98	70	0.17881	64	60	0.8894	30	77	2.4385
97	3	0.19355	63	55	0.96384	29	76	2.4385
96	63	0.20674	62	96	0.98711	28	75	2.4386
95	23	0.24381	61	85	1.0243	27	74	2.4386
94	56	0.24427	60	53	1.0592	26	73	2.4386
93	40	0.24427	59	21	1.2732	25	78	2.4386
92	61	0.24429	58	33	1.2858	24	79	2.4386
91	52	0.24429	57	82	1.3263	23	93	2.5095
90	46	0.24429	56	98	1.3263	22	18	2.5279
89	59	0.24429	55	29	1.4275	21	22	2.5603
88	89	0.2443	54	30	1.4344	20	6	2.6086
87	37	0.24431	53	44	1.5448	19	12	2.643
86	43	0.24432	52	41	1.5448	18	4	2.7494
85	49	0.24432	51	50	1.5448	17	19	2.9982
84	17	0.2641	50	47	1.5448	16	91	3.3005
83	57	0.28131	49	38	1.5448	15	35	3.4352
82	72	0.33849	48	87	1.5448	14	81	3.4947
81	26	0.41444	47	102	1.6035	13	7	3.808
80	15	0.43648	46	5	1.7075	12	62	3.8484
79	2	0.45936	45	42	1.7997	11	11	3.8704
78	27	0.48732	44	36	1.7998	10	20	4.2349
77	65	0.49569	43	51	1.7998	9	58	4.8312
76	24	0.50614	42	45	1.7998	8	10	4.9274
75	25	0.53272	41	48	1.7998	7	34	6.0799
74	66	0.5644	40	88	1.7998	6	8	6.5537
73	68	0.59339	39	39	1.7998	5	99	7.0495
72	28	0.67678	38	92	2.1574	4	100	7.0495
71	86	0.70566	37	90	2.1586	3	9	7.5284
70	95	0.73381	36	80	2.1668	2	84	9.7024
69	101	0.74812	35	1	2.272	1	54	9.9947