Support Vector Machines for Credit Scoring and discovery of significant features

Support vector machines for credit scoring and discovery of significant features

Tony Bellotti and Jonathan Crook

Credit Research Centre
School of Management and Economics
University of Edinburgh#
William Robertson Building
50 George Square
Edinburgh EH8 9JY
UK

7 May 2007

Research funded by EPSRC grant EP/D505380/1, working as part of the Quantitative Financial Risk Management Centre

Abstract

The assessment of risk of default on credit is important for financial institutions. Logistic regression and discriminant analysis are techniques traditionally used in credit scoring for determining likelihood to default based on consumer application and credit reference agency data. We test support vector machines against these traditional methods on a large credit card database. We find that they are competitive and can be used as the basis of a feature selection method to discover those features that are most significant in determining risk of default.

1. Introduction

Credit scoring is the set of decision models and techniques that aid lenders in granting consumer credit by assessing the risk of lending to different consumers. It is an important area of research that enables financial institutions to develop lending strategies to optimise profit. Additionally, bad debt is a growing social problem that could be tackled partly by better informed lending enabled by more accurate credit scoring models. A range of different data mining and statistical techniques have been used since the 1930's when numerical score cards were first introduced by mail-order companies (Thomas et al. 2002, Section 1.3). It is now common for financial institutions to use statistical methods such as logistic regression (LR) and linear discriminant analysis (LDA) to build credit scoring models. Potential borrowers are

classified according to their probability to default on a loan, based on application and credit reference agency data collected about them. Such models are used by setting a threshold on the probability to default and rejecting loan applications that fall below this level.

Support vector machines (SVMs) have been applied successfully in many classification problems such as text categorisation, image recognition and gene expression analysis (eg see Cristianini and Shawe-Taylor (2000), chapter 8). Experiments using SVM for credit scoring are relatively new, however. Several papers have recently been published assessing the performance of SVM for credit scoring. Baesens et al. (2003) apply SVMs, along with other classifiers to several data sets. They report that SVMs perform well in comparison with other algorithms, but do not always give the best performance. Schebesch and Stecking (2005) apply SVM to a database of applicants for building and loan credit. They conclude that SVMs perform slightly better than LR, but not significantly so. They also use SVMs, with its capacity to output support vectors, to discover typical and critical regions of the problem space. Both papers report using linear SVM and a Gaussian radial basis function (RBF) kernel. In both cases, the size of the credit database is much smaller than would typically be used in a real application. Van Gestel et al (2006) use least squares SVMs with a Bayesian kernel to derive classifiers for corporate bankruptcy. They find no significant difference between SVM, LR and LDA in terms of proportion of test cases correctly classified and no difference between LR and SVM in terms of area under the ROC curve. Li et al (2006) find SVMs outperform multi-layer perceptrons for consumer credit data, but their results are also based on a small sample size. Huang et al (2004) compare SVMs with a back-propagation neural network to predict corporate credit ratings but find inconsequential differences in performance. Lee (2007) find a similar result for corporate loans. Dikken (2005) finds SVMs to be inferior to LR when modelling corporate credit risk. Huang et al (2007) find SVMs classify credit applications no more accurately than artificial neural nets (ANN), decision trees or genetic algorithms (GA), and compared the relative importance of using features selected by GA and SVM along with ANN and genetic programming. However, they use data sets far smaller and with fewer features than would be used by a financial institution and do not compare the features selected by SVM alone, nor do they compare with methods used in practice such as LR.

In this paper, our general framework is to compare the performance of SVM against several other well-known algorithms: LR, LDA and *k*-nearest neighbours (*k*NN). We extend the work on assessing SVM for credit scoring in several ways.

1. SVM is tested against a much larger database of credit card customers than has been considered in the literature so far. We restrict our attention to those accounts opened in the same three month period. Hand (2006) points out that for many classification problems, the data suffers from population drift, in that the class distributions shift over time. This is particularly true of credit data with customer behaviour changing over time due to economic circumstances or changes in product development and marketing. For this reason a clearer model can be developed if it is based on data taken from a narrow time period within which there is likely to be less variability in these circumstances.

2. SVM is tested with a polynomial kernel to determine if a non-linear polynomial decision space yields better performance than linear SVM or using the Gaussian RBF kernel.

3. SVM performance is assessed in light of the number of support vectors required to model the data.

4. Financial institutions are primarily interested in determining which consumers are most likely to default on loans. However, they are also interested in knowing which characteristics of a consumer are most likely to affect their likelihood to default. For example, is a home-owner less likely to default than a tenant? This information allows credit modellers to stress test their predictions. Traditionally a test of significance of features is used to discover these characteristics. When a LR problem is solved using maximum likelihood estimation, a Wald statistic can be computed for each feature which is then used to determine significance. As an alternative, we follow Guyon et al. (2002) who select significant features in the data using the square of weights on features output by SVM. We apply this approach to select the top ranking

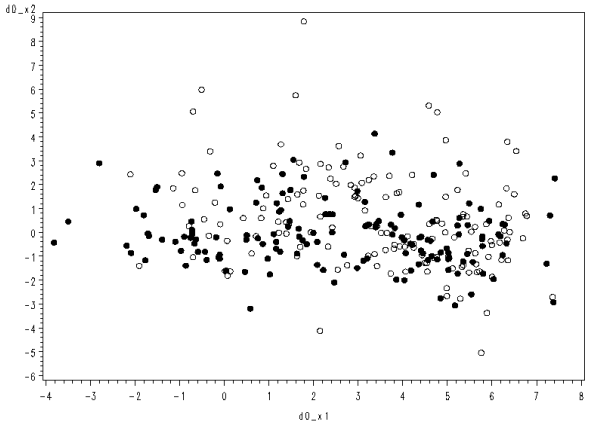
features that are significant for credit scoring. This selection is compared with that given using LR.

2. Data

A data set of approximately 25,000 customers with credit cards opened in the same three month period of 2004 was provided by a major financial institution. Four different credit card products are represented in the data. A customer is defined to have *defaulted* if he or she has become three months or more behind in their payments within the first 12 months after the account is opened. Other definitions of default can be used, but this one is common in credit scoring (Thomas, et al. 2002, Section 8.3). Defaulting customers are referred to as *bad* cases and all others as *good* cases. The data includes 34 features taken from each customer's original application along with features extracted from a credit reference agency at the time of application. The data is standardized before use, so each feature has the same mean (0) and variance (1).

Typically, credit data is not easily separable by any decision surface. This is natural since the data at time of application cannot capture the complexities in each individual customer's life that may lead to default. The application data can at best only provide an indication of default. Consequently, it is usual for the rates of misclassification on credit data to be between around 20% and 30% (eg see Baesens et al, 2003). This would be considered a poor result for many other classification problems but is typical of credit data. The poor separability of the credit data is illustrated in Figure 1. The good cases tend to cluster towards the bottom-right and the bad towards the top-left, but this is only a very general trend and there is no clear separation.

Figure 1. Illustration the of poor separability of the credit data.



Partial least squares was used to transform data for 100 good cases (black) and 100 bad cases (white) selected randomly from the data into two factors given as the *x* and *y* axis of the graph.

3. Methods

The SVM is a relatively new learning algorithm that can be used for classification. We compare its performance against three older statistical classification methods: LR, LDA and *k*NN. All algorithms are described briefly below for a sequence of *n* training examples $(\mathbf{x}_1, y_1, \ldots, \mathbf{x}_n, y_n)$ with feature vectors $\mathbf{x}_i$ and class labels y_i. For credit scoring, the class label is either *bad* or *good*.

3.1. Support Vector Machine (SVM) classifier

SVM separates binary classified data by a hyperplane such that the margin width between the hyperplane and the examples is maximized. Statistical learning theory shows that maximizing the margin width reduces the complexity of the model, consequently reducing the expected general risk of error. For problems where data is not separable by a hyperplane, typical of most real-world classification problems, a soft margin is used. In this case, training examples are allowed some slack to be on the wrong side of the margin. However, they accrue a penalty proportional to how far they are on the wrong side. The sum of the penalties is minimized whilst maximizing the margin width. A parameter *C* controls the relative cost of each goal in the overall optimization problem. The SVM optimization problem can be expressed algebraically as a dual form quadratic programming problem.

Let $y_i \in \{-1,+1\}$ for all *i*=1 to *n*. Then the SVM optimization problem is

$$\max_{\boldsymbol{\alpha}} \left(\sum_{i=1}^{n} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \right)$$

subject to constraints

$$0 \le \alpha_i \le C \text{ for all } i=1 \text{ to } n \text{ and } \sum_{i=1}^{n} y_i \alpha_i = 0$$

where α_i is a Lagrange multiplier for each training example *i*. The kernel function *k* can be used to implement non-linear models of the data. For this paper, we consider three commonly used kernels.

Linear model	$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
Polynomial model, with degree *d*	$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$
Gaussian radial basis function (RBF), with parameter σ	$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right)$

Using non-linear kernels means that it is not feasible to extract an explicit scorecard, although predictions can be made using them.

The vector of Lagrange multipliers $\boldsymbol{\alpha}$ is sufficient to define the output decision rule. A classification prediction is made on a new example **x** as

$$\hat{y} = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \right)$$

where *b* is a threshold term computed as

$$b = \sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_j) \text{ for any } j \in \{1, \ldots, n\} \text{ such that } 0 < \alpha_j < C.$$

Training examples are called “support vectors” (SVs) if they are on the margin or are on the wrong side of the margin. This is because together they are sufficient to “support” the optimal separating hyperplane, since only SVs are such that $\alpha_i > 0$. It

follows that the decision rule can be expressed simply in terms of SVs. See Vapnik (1998) and Cristianini and Shawe-Taylor (2000) for details about SVMs.

3.2. Logistic Regression (LR)

Let $y_i \in \{0,1\}$ for all *i*=1 to *n*. LR estimates the probability that the label is 1 for a given example **x** using the model

$$P(y=1 \mid \mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}\cdot\mathbf{x}-b)}.$$

Parameters **w** and *b* can be estimated using the maximum likelihood procedure to maximize the log-likelihood function, with respect to **w** and *b*,

$$L(\mathbf{x}_1,\ldots,\mathbf{x}_n \mid \mathbf{w},b) = \sum_{i=1}^{n} y_i \log p_i + (1-y_i)\log(1-p_i) \text{ where } p_i = P(y=1 \mid \mathbf{x}_i).$$

Non-linear LR models are allowed by including interaction variables chosen for inclusion using a likelihood ratio test.

3.3. Linear Discriminant Analysis (LDA)

Let $y_i \in \{0,1\}$ for all *i*=1 to *n*. The Fisher discriminant criterion is given by the function

$$J(\mathbf{w},(\mathbf{x}_1,y_1,\ldots,\mathbf{x}_n,y_n)) = \frac{(m_1-m_0)^2}{s_1^2+s_0^2}$$

where m_y and s_y^2 are in-class means and variances given by

$$m_y = \frac{1}{|C_y|}\sum_{i\in C_y}\mathbf{w}\cdot\mathbf{x}_i,\ s_y^2 = \frac{1}{|C_y|}\sum_{i\in C_y}(\mathbf{w}\cdot\mathbf{x}_i - m_y)^2 \text{ where } C_y = \{i=1,\ldots,n \mid y_i = y\}.$$

Maximizing the Fisher discriminant criterion yields a hyperplane $0=\mathbf{w}\cdot\mathbf{x}$ which maximizes the distance between the means of the two classes in relation to their variance. This is the optimization problem for LDA. Once a hyperplane is computed, a prediction is made on a new example **x** as $\hat{y} = \text{sgn}(\mathbf{w}\cdot\mathbf{x})$. Assuming a Gaussian distribution of examples about the class mean, this method can also yield probabilities for each class label prediction. Duda et al. (2001), Section 3.8.3, give further details.

3.4. *k*-Nearest Neighbours (*k*NN)

*k*NN is a nonparametric classification algorithm that uses a distance measure to make predictions without building a model. The prediction for a new example **x** is given by the majority class label within a neighbourhood of **x** in the training data. Formally, $\hat{y} = \arg\max_y k_y$ where k_y is the number of cases of class *y* amongst the *k* nearest neighbours taken from the training set. The probability of the example belonging to class *y* is estimated as $\hat{p} = k_y / k$ (Hand 1981, Section 2.4). We use the usual Euclidean distance measure to determine the neighbourhood of an example. Henley and Hand (1997) use kNN for credit scoring and compare with other methods including LR.

3.5. Validation and assessment

The data set is randomly divided into a training and test set in the ratio 2:1, whilst preserving the same proportion of bad cases in each. A hold-out procedure is then used to assess each classification method. Experiments are repeated 10 times for different random splits of the data to test that the performance measures are stable with low standard deviation across the 10 permutations.

Error rates are reported on the test set as the proportion of test examples wrongly classified. It is possible to set a threshold term for the decision rule output by each algorithm to control the distribution of cases classified as good or bad. For example, in LR, we set a threshold *t* and classify all examples **x** with $P(y=1|\mathbf{x}) < t$ as good ($\hat{y} = 0$). Otherwise **x** is classified as a bad case. The threshold setting depends on a prior assumption of the relative cost of misclassifying good or bad cases. For example, we expect that a bad case misclassified as good, and so given a loan, would yield a greater loss – ie the loss of a substantial part of the loan value – than a good case misclassified as bad, leading to a loan not being made and the subsequent loss of interest payments. However, it is not reasonable to assume this relative cost for assessment. Also, using error rates makes it difficult to compare algorithms with different threshold terms that would lead to different distributions of misclassification of good and bad cases. Therefore it is usual to measure performance with a receiver operating characteristic (ROC) curve which plots sensitivity (true positive rate) against 1-specificity (false negative rate) for the full range of possible threshold

values. This is a typical performance measure for credit scoring (eg Engelmann et al. 2003, Baesens et al. 2003). The area under the ROC curve (AUC) is used as a single summary statistic for measuring performance and comparing algorithms (DeLong et al. 1988). Note that a ROC curve is constructed for SVM by varying the threshold term *b*. Reducing this threshold will increase the number of cases classified as bad ($\hat{y} = +1$).

3.6. Parameter Tuning

Both SVM and *k*NN require parameters to be set prior to classification. These parameters are tuned against a separate data set, independent of the data set used for classification as described in Section 2. The tuning data set comprises 17,585 customers taken from 2003 for the same products. It is split into a training and test set, again in a 2:1 ratio. A grid search is used to determine those parameter values that maximize the AUC on the test set. SVM is tuned for a range of values of C from 10^{-9} to 10^{4}. Additionally the polynomial kernel is tuned for degrees d=2, 3 and 4; and the Gaussian RBF kernel is tuned for values of σ from 10^{-8} to 10^{-3}. *k*NN is tuned for values of *k* from 50 to 6000.

3.7. Discovering significant features

The maximum likelihood method used for LR is typically used to determine the most significant features in a model. The process of maximizing the likelihood function yields a Wald statistic for each coefficient to test the null-hypothesis that its true value is zero. A p-value is computed based on the Wald statistic using a chi-square test. The lower the p-value, the less likely the true value is zero, hence the more significant the feature with the coefficient is likely to be for the model. We set a significance level of 0.01 and select those features with p-values less than this.

Guyon et al. (2002) propose using the square of the weights from the hyperplane generated by SVM as a feature selection criterion. They show this will minimize generalized risk and apply the technique to cancer classification. They used a recursive procedure, removing a few features at a time. However, since for the credit scoring problem, there are relatively few features to begin with, we do not need to apply a recursive procedure. We simply use the magnitude of weights on features as a feature selection criterion. We set a threshold of 0.1 and all features with weights

greater than this will be selected as significant features. This threshold level is chosen since we found it yields approximately the same number of features as the LR method described above. Since the data is standardized, it is reasonable to directly compare the magnitudes of weights on different features.

4. Results

Results are given in this section for pre-classification parameter tuning, algorithm comparison and significant feature discovery using LR and SVM.

4.1. Parameter tuning

Table 1 shows the optimal parameter values that maximize test AUC for SVM on the tuning data set.

Table 1. Optimal parameter values.

SVM kernel	*Optimal parameters*	*#SV*	*Training AUC*	*Test AUC*
Linear	C=0.001	7961	0.796	0.791
Polynomial	C=10^{-5}, d=2	9458	0.808	0.758
Gaussian RBF	C=1000, $\sigma = 10^{-6}$	7803	0.796	0.791

#SV = number of support vectors

It is interesting that the non-linear kernels do not perform better than the simple linear model. In particular, the polynomial kernel performs poorly. This may be because this non-linear model is over-fitting the data. This is evident in the difference between the relatively high training AUC and low test AUC. However, the results are not sufficient to assert this conclusively.

The best results are achieved when large numbers of SVs are extracted. Over 50% of training examples are SVs. This is due to the fact that credit data is not easily separable by any decision surface as explained in Section 2, so many of the training examples remain misclassified.

For *k*NN, test AUC was stable at over 0.760 for values of *k* between 500 and 4000. It is usual for performance to be stable across a wide range of values of *k* for large

training sets (Olsson 2006). We choose the mid-range figure *k*=2000 as an optimal value for the following comparative experiments.

4.2. Comparing algorithms

Table 2 shows results on the test set. Mean and standard deviations of AUC are given for the 10 experiments with different random splits of the data into training and test sets. The optimal parameter settings found in the previous section are used for these experiments.

Table 2. Performance for different algorithms

Classification algorithm	*Test AUC (mean)*	*Standard deviation*
LR	0.779	0.0063
LR with interaction variables	0.777	0.0076
SVM: linear	0.783	0.0055
SVM: polynomial	0.755	0.0068
SVM: Gaussian RBF	0.783	0.0053
LDA	0.781	0.0058
*k*NN	0.756	0.0063

The standard deviations are relatively low (less than 1% of mean AUC) indicating that the measured performance is stable.

SVM with a linear or Gaussian model performs best yielding the highest AUC. However, the differences in performance are small and are not significant. Schebesch and Stecking (2005) reach a similar conclusion with their experiments.

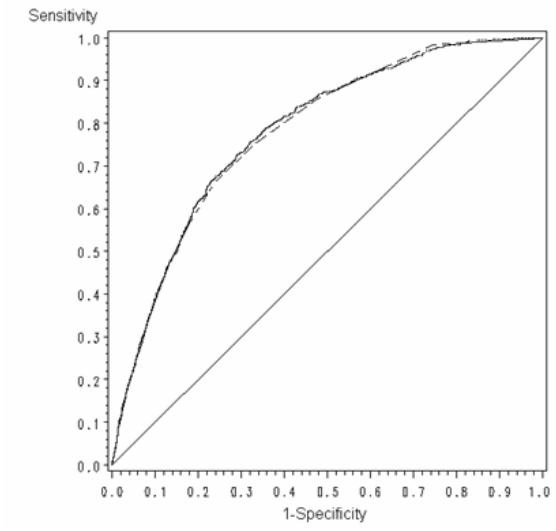
The only algorithms that stand out as particularly poor are SVM with polynomial kernel and *k*NN. As mentioned in Section 4.1, we suspect the polynomial kernel over-fits the training data. The poor result with *k*NN corroborates the results given by Baesens et al (2003).

Neither LR with interaction variables nor SVM with non-linear kernels give an improvement over the simpler models. This indicates that the data is broadly linearly

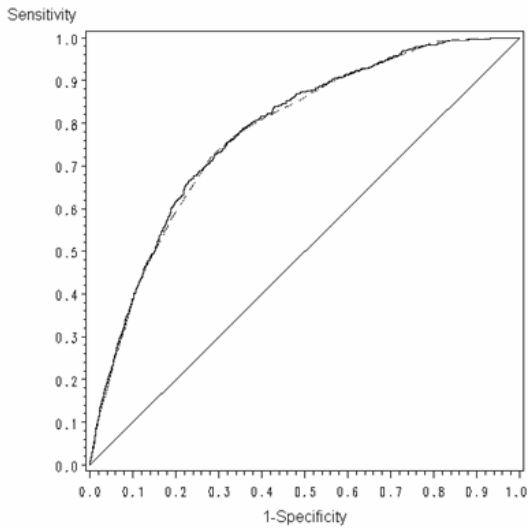
separable. Gayler (2006) has argued that interaction variables are less stable than the main effects and they would usually only be included in a model if the modeller has prior belief in their relevance to credit scoring. Our results tend to support this view.

Figure 2 shows typical ROC curves taken from one experiment. It is clear that the ROC curve for SVM, LR and LDA are all very similar. The only algorithm which gives a distinctly poor ROC curve is *k*NN which is outperformed by SVM across the whole range of the graph.

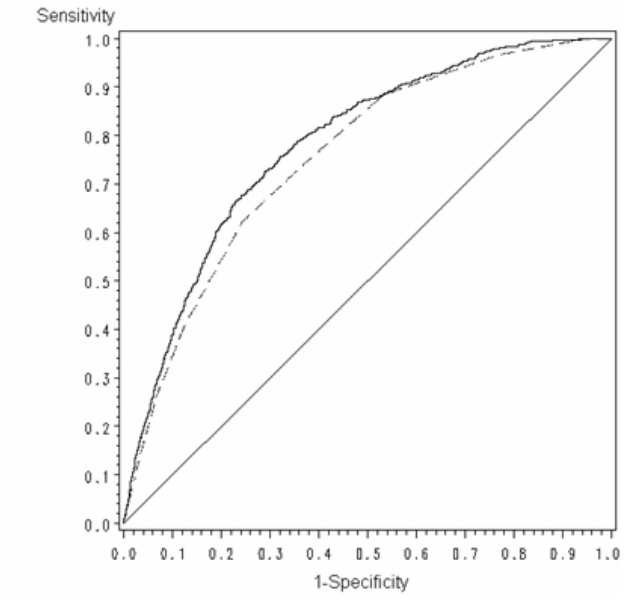
Figure 2. ROC curves for performance on test data, comparing the performance of linear SVM with LR, LDA and *k*NN.



SVM (unbroken line) and LR (broken line)



SVM (unbroken line) and LDA (broken line)



SVM (unbroken line) and *k*NN (broken line)

Error rates can be derived by setting a cut-off threshold for each model and predicting those test cases with a score computed from the model below the cut-off as bads and those above as goods. Error rates are given by comparing predicted against actual classifications across the test set. However, error rates are not comparative since each classifier will yield a different distribution of errors on good and bad cases. This is why using AUC is a better comparative measure, since it measures predictive performance across all possible chosen cut-off thresholds. Nevertheless, it is interesting to review error rates on good and bad cases for SVM to ensure that they represent an acceptable performance. The natural cut-off for SVM is the threshold term *b* described in Section 3.1. For linear SVM the error rates for good and bad cases in the test set are 27.4% and 29.6% respectively and for SVM with a Gaussian RBF kernel the error rates for good and bad cases are 27.2% and 30.1%. These outcomes are within the range of error rates we would expect from predicting default in credit data, as we discussed in Section 2.

4.3. Significant features

Table 3 shows significant features selected using LR and SVM with a linear model as described in Section 3.7. A count of the number of times each feature is selected in each of the 10 experiments is made. Only those features with a count greater than 4 for either the LR or SVM selection methods are reported. Typical weights and coefficient estimates are also given for each variable.

Table 3. Significant features.

Feature	*SVM count*	*LR count*	*Typical SVM weight*	*Typical LR coefficient estimate*
F1: Home owner	10	10	-0.280	-0.357
F2: Time with bank	10	10	-0.180	-0.240
F3: Insurance required	10	10	+0.346	+0.337
F4: No. settled non-mail order CAIS accounts	10	10	-0.187	-0.230

F5: Total outstanding balance excluding mortgages on all active CAIS accounts	10	10	+0.325	+0.357
F6: Total no. of credit searches in last 6 months	10	10	+0.208	+0.221
F7: Worst account status (0 to 99)	10	10	+0.248	+0.211
F8: Age	10	10	*	*
F9: Product (type of credit card)	10	10	*	*
F10: Time since most delinquent account	7	7	+0.139	+0.112
F11: UK Mosaic code	7	5	+0.103	+0.129

*These variables are course classified into several indicator variables. Therefore there are several weights and coefficient estimates associated with each of them.

The direction of SVM weights and LR coefficient estimates is the same and indicates how each feature contributes to the risk of default. A positive value indicates higher risk and a negative value a lower risk. For example, an applicant who has already applied for credit several times (F6) will be more likely to default and a home owner (F1) is less likely to default.

These results show that the two methods agree strongly on the most significant features. The fact that two very different methods give the same results provides further confidence that these features can be taken forward for use in credit scorecards to determine the risk of default for individual applicants for credit. It shows that SVM can be used successfully for feature selection in credit scoring.

5. Conclusions

SVMs are a relatively new technique for application to credit scoring. We test them on a much larger credit data set than has been used in previous studies. We find that SVMs are successful in comparison to established approaches to classifying credit card customers who default. This corroborates the findings of previous researchers. In addition, we find that, unlike many other learning tasks, a large number of support vectors are required to achieve the best performance. This is due to the nature of the credit data for which the available application data can only be broadly indicative of default. Finally, we show that SVM can be used successfully as a feature selection

method to determine those application variables that can be used to most significantly indicate the likelihood of default.

There are several further lines of investigation. Firstly, we discovered that the type of product (F9 in Table 3) is an important indicator of default. It would be interesting to build separate models for each product to determine how performance and significant features vary between them. Secondly, we took data from just one three-month period to avoid the problem of population drift (Hand 2006). It would be interesting to see how models and performance change across time and how robust simple and complex models are when tested against test sets drawn from later dates.

Acknowledgements

We used *SVM light* for this project which is available at `http://svmlight.joachims.org` and is documented by Joachims (1999). This research is funded through EPSRC grant EP/D505380/1.

References

Baesens B, van Gestel T, Viaene S, Stepanova M, Suykens J and Vanthienen J (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society* **54**: 1082-1088.

Cristianini N and Shawe-Taylor J (2000). *Support vector machines and other kernel-based learning methods.* Cambridge University Press.

Duda RO, Hart PE, Stork DG (2001). *Pattern Classification.* Wiley.

Gayler RW (2006). Comment: Classifier Technology and the Illusion of Progress - Credit Scoring. *Statistical Science* **21**(1):19-23.

Guyon I, Weston J, Barnhill S, Vapnik V (2002). Gene selection for cancer classification using support vector machines. *Machine Learning* **46**(1-3):389-422.

Hand DJ (1981). *Discrimination and Classification.* Wiley.

Hand DJ (2006). Classifier Technology and the Illusion of Progress. *Statistical Science* **21**(1):1-14.

Henley WE, Hand DJ (1997). Construction of a k-nearest-neighbour credit-scoring system. *Journal of Management Mathematics* 1997 **8**(4):305-321.

Huang C-L, Chen M-C, Wang C-J (2007). Credit scoring with a data mining approach based on support vector machines. Expert Systems with Applications **33**(4):847-856

Huang Z, Chen H, Hsu C-J, Chen W-H, Wu S (2004). Credit rating analysis with support vector machines and neural networks: a market comparative study. *Decision Support Systems* (Special issue: *Data mining for financial decision making*) **37**(4):543-558.

Joachims T (1999). Making large-Scale SVM Learning Practical. *Advances in Kernel Methods - Support Vector Learning*, Schölkopf B, Burges C, Smola A (ed.), MIT-Press.

Lee Y-C (2007). Application of support vector machines to corporate credit rating prediction. *Expert Systems with Applications* **33**(1):67-74.

Li ST, Shiue W, Huang MH (2006). The evaluation of consumer loans using support vector machines. *Expert Systems with Applications* **30**(4):772-782.

Olsson J S (2006). An analysis of the coupling between training set and neighbourhood sizes for the *k*NN classifier. *29th annual international ACM SIGIR 2006*.

Schebesch K B and Stecking R (2005). Support vector machines for classifying and describing credit applicants: detecting typical and critical regions. *Journal of the Operational Research Society* **56**:1082-1088.

Thomas LC, Edelman DB and Crook JN (2002). *Credit Scoring and its Applications*. SIAM Monographs on Mathematical Modeling and Computation. SIAM: Philidelphia, USA.

Van Gestel T, Baesens B, Suykens JAK, Van den Poel D, Baestaens D, Willekens M (2006). Bayesian kernel based classification for financial distress detection. *European Journal of Operational Research* 172: 979-1003.

Vapnik V (1998). *Statistical Learning Theory*. Wiley: New York.