On semi-supervised Support Vector Machines for credit client classification with fictitious training

KLAUS BRUNO SCHEBESCH
Department of Economics and Department of Informatics
Vasile Goldiş Western University Arad
RO-310086 Arad, ROMANIA
kbschebesch@uvvg.ro

RALF STECKING
Department of Economics
Carl von Ossietzky University Oldenburg
D-26111 Oldenburg, GERMANY
ralf.w.stecking@uni-oldenburg.de

Abstract: - Fictitious training data points complementing empirical training examples have been observed within several contexts to enhance out-of-sample performance of classification methods. Investigating some simple generating rules for fictitious credit client scoring data, we showed in past work that the degree to which this desirable effect is achieved varies markedly. When using Support Vector Machines (SVM) as a modelling tool, effective performance enhancement depends on the combination of kernels and generating rules for fictitious training data. In past work fictitious training data were slight variations of some selected original training examples. For instance, they were chosen to be in the close vicinity of support vectors which belong to a SVM trained on the original data, while all the class labels of the fictitious data were assumed to be equal to those of the respective generating data points. In this paper we introduce more complex transformations of original data examples, which generate unlabelled fictitious training data. Hence we explore a generating mechanism for fictitious credit clients, without fixing their defaulting behavior in advance. In order to propose labels for the new examples for which no a priori assumption was placed on their class membership, we use transductive SVM, a semi-supervised classification method.We also discuss how to adapt model performance validation to the enlarged training data sets.

Key-Words: - Transductive SVM; Machine learning; Credit client classification, Fictitious training data.

1 Introduction

In past work on credit scoring by mainly using Support Vector Machines (SVM) in order to model different aspects of the data (as in [8], [10]), we find some hints at not yet detected possibly non-linear structure in the data.

Recently [11] we started investigating possibilities of extracting such additional structure by means of **fictitious training data**, i.e. data artificially generated by the modeler in order to hopefully reinforce some underrepresented but potentially useful aspects of the empirical data. Abu-Mostafa [1] describes the beneficial effects of reinforcing symmetries, e.g. in images by emphasizing certain *neighborhoods* via fictitious training examples.

In [9] and in [5] we find work which reinforces **manifold geometry** information, e.g. of "feasible" handwritten digits via fictitious training examples formed in the immediate vicinity of support vectors of pre-trained SVM. Finally, new approaches presented in [2] point towards the usefulness *instance based constraints*, i.e. "example *i* and *j* must (or cannot) be in the same class" – without specifying the classes, which can provide clues of how to construct useful fictitious training examples.

Our present interest in *credit scoring data* leads us into asking if similar principles of constructing fictitious training examples as those in the above cited applications can be found here too. Unfortunately, we do not know of any geometrically related input features or neighborhoods induced by symmetries in credit client data and we cannot (as yet) justify any instance based constraints applicable to pairs of credit clients.

ISBN: 978-960-474-311-7

2 Types of Fictitious Data in Credit Scoring

Given a training set $\{x_i, y_i\}_{i=1,...,N}$, with *m*-dimensional input features x_i and with associated label $y_i \in \{-1,+1\}$. Think of x_i as being the personal characteristics of client *i*, for instance age, income, demographic data, professional history and many other features which can be legally (if only confidentially) used in order to derive decisions regarding client crediting and of y_i as being a binary label, which denotes if this credit client was defaulting on the credit in the past. The input vectors are such that

$$x_i = [\underbrace{\boxed{1\,0} * * \overbrace{\boxed{1\,0\,0}}^{B_j} * * \boxed{1\,0}}_{\text{block binary}} \mid \underbrace{x_{i,b+1}, \ldots, x_{i,m}}_{\text{real valued}}],$$

where just a **subset** of all 2^b configurations with *b* entries can be realized due to an imposed binary block coding $\{B_j\}$, where $j = 1,...,d$, with $d < b$.

In a sense, all possible $\prod_{j=1}^{d} |B_j|$ configurations of the binary part of the above input representation are *feasible credit clients*.

The real part of the input vectors is not changed but transcribed from parents (original examples) to offspring (fictitious examples). In principle we can choose one of the following methods:

(a) Transform x_i into x_j by choosing x_j within a small distance $d(x_j, x_j) < \varepsilon$ (**local** mutation), a choice taken in [11].

(b) Transform the binary parts of two parents into two offspring by information exchange (crossover, **semi-local** conservative action, *this paper*):

$$x_{i_1} \otimes x_{i_2} \quad \text{leads to} \quad x_{j_1} \otimes x_{j_2}.$$

More specifically the follwing possibilities exist:

1. $y_{i_1} = y_{i_2} = a$ leads to $y_{j_1} = y_{j_2} = a$,
2. $y_{i_1} = y_{i_2} = a$ leads to unlabelled x_{j_1} and x_{j_2},
3. $y_{i_1} \neq y_{i_2}$ leads to unlabelled x_{j_1} and x_{j_2}.

(c) **Global**, unrestricted mutation within the feasible binary configurations $\{B_j\}$ leads to unlabeled input vector sections x_{j_1} and x_{j_2}.

2.1 The role of SVM classification

An SVM decision or classification rule derived from *N* training examples is then given by [12]:

$$y = \operatorname{sgn}\left(\sum_{i=1}^{N} \alpha_i^* y_i k(x_i, x) + b^* \right),$$

with $0 \le \alpha_i^* \le C$. The user selected constant C determines the amount of misclassification or error tolerance, which is permitted in order to avoid overtraining (no error tolerance at all would be detrimental in all cases of noisy training data, as is for instance the case with credit scoring data). The SVM optimization procedure uses the training examples $\{x_i, y_i\}_{i=1,...,N}$ as inputs to find *N* optimal dual variables $\alpha_i^* \ge 0$ and $b^* \in \boldsymbol{R}$, in order to predict a class membership y for a new (as yet unseen) input x.

The training examples verifying $0 < \alpha_i^* < C$ are called **essential support vectors** to be used in the sequel. Such support vectors separate regions in input features space which can be classified without error from those which are populated by both positive and negative cases, which hence cannot be functionally separated by a classification rule, which is deliberately kept too simple as to permit perfect classification (remember the role of C).

Furthermore, the SVM model uses a kernel function $k(x_j, x_j) \ge 0$, in general a nonlinear function related to the distance between x_i and x_j, which extracts some information about the vicinity of all training pairs in the *feature space*.

In the SVM literature (as in [7], [12]) this feature space refers to a hypothetical dimension blow-up of the input space until all differently labelled cases (classes) can be separated linearly, however subject to the constraint that the "distance order" between point pairs in the original input space is preserved in this new feature space. Kernel based methods like SVM can make implicily use of such feature spaces without requiring their explicit construction.

ISBN: 978-960-474-311-7

2.2 The use of the empirical credit client data

The empirical data set for our credit scoring models is a sample of N = 658 clients for a building and loan credit with seven metric, two ordinal and seven nominal input variables which are coded into a 40 dimensional input vector (m = 40). It contains 323 defaulting and 335 non defaulting credit clients.

Fictitious training data are generated in two different ways: Using a binary encoding of the categorical input variables, the genetic operators mutation and recombination are applied in order to generate offsprings, which are both feasible and new:

(a) Fictitious data generated by **mutation** are constructed by switching categories in nominal or ordinal input variables. A new nominal category is generated by switching the outcome of the original variable to any of the other possible categories.
For ordinal variables, only switches to the neighboring (or to those of type *not available*) categories are allowed.

(b) A **recombination** procedure of two existing data points is generating a fictitious training example by a simple crossover recombination of the two parents using a randomly chosen, but feasible crossover point.

In general, the label y of such a fictitious data point is unknown and has to be determined by semi-supervised classification using transductive SVM (see section 3).
Furthermore, for SVM model building we follow a two step procedure, proposed by [9]. In a first step, a SVM is trained using the initial data set of 658 clients. The resulting support vectors are then extracted and fictitious data points are generated by (*i*) mutation and (*ii*) crossover recombination, as outlined before.

3 Semi-supervised classification

As the data of a supervised classification problem are given by a set of N training examples $\{x_i, y_i\}_{i=1,...,N}$, with vector $x_i = (x_{i1}, x_{i2}, ..., x_{im})$ as $m-$dimensional input pattern of object (or credit client) i, and with $y_i \in \{-1,+1\}$ the associated class labels, inductive linear SVM has to solve the following quadratic optimization problem:

$$\min_{w,\zeta} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{N} \zeta_i$$

$$\text{subject to } y_i(\langle x_i, w \rangle + b) \geq 1 - \zeta_i \quad \text{and} \quad \zeta_i \geq 0 .$$

The (forecasted) SVM class label is then given by

$$\operatorname{sgn}(\langle x, w \rangle + b) = \operatorname{sgn}\left(\sum_{i=1}^{N} \alpha_i^* y_i \langle x, x_i \rangle + b^* \right),$$

with $w = \sum_{i=1}^{N} \alpha_i^* y_i x_i$ and with $0 \leq \alpha_i^* \leq C$ as before. With additional **unlabeled training examples** (where the value of the target variable y is unknown), the inductive SVM cannot be used anymore. For such cases **transductive SVM** were developed [12] as a method to learn a *large margin hyperplane* classifier using labeled training data, but simultaneously force this hyperplane to be far away from the unlabeled data [4].
In transductive learning some additional K unlabeled training examples $x_1^*, x_2^*, ..., x_K^*$ are given. In some cases, K may be much bigger then N, denoting that unlabeled data – being of less commercial value – are more readily available or can be collected more easily. The modified optimization problem to be solved for transductive learning then reads [7]:

$$\min_{w,\zeta,y^*,\zeta^*} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{N} \zeta_i + C^* \sum_{j=1}^{K} \zeta_j^*$$

$$\text{s. t.} \quad y_i(\langle x_i, w \rangle + b) \geq 1 - \zeta_i \quad \text{and} \quad \zeta_i \geq 0 ,$$
$$y_j^*(\langle x_j^*, w \rangle + b) \geq 1 - \zeta_j^* \quad \text{and} \quad \zeta_j^* \geq 0 ,$$
$$\text{and } y_j^* \in \{-1,+1\} \text{ for all } j = 1, ..., K .$$

Note that transductive SVM in general include solving a **combinatorial search problem** to determine the class labels y_j^* (as in [3], [4], [6]). The classification function which forecasts y from a new x is given by including the K unlabelled cases, namely by

$$\operatorname{sgn}\left(\sum_{i=1}^{N} \alpha_i^* y_i \langle x, x_i \rangle + \sum_{j=1}^{K} \alpha_j^* y_j^* \langle x, x_j^* \rangle + b^* \right).$$

In practice, using an appropriately modified kernel function as described in section 2.1, e.g. $k(x, x_j^*)$ instead of scalar product $\langle x, x_j^* \rangle$ allows for non-linear semi-supervised classification. In the sequel, SVM with different kernel functions are used for

ISBN: 978-960-474-311-7

classifying good and bad credit clients. Detailed information about diffrent kernels, hyperparameters and SVM tuning can be found in [10].

4 Empirical results

We trained SVM with four different kernels using three different data sets. The linear kernel is used as reference (benchmark) and the remaining kernels implement different non-linerities. The RBF kernel is the most flexible. This kernel can be easily adapted to complicated input-output maps but equally so to useless noise. In Table 1 for each kernel the number of essential support vectors (SV, see section 2), the number of bounded support vectors (BSV, those who's associated dual variables verify $\alpha_i^* = C$) and the tenfold cross validation classification error is shown for four different models trained on essential support vectors (where $0 < \alpha_i^* < C$) plus two types of fictitious data: mutated support vectors with labels preserved from the origin (L-SV Fict.) and (ii) unlabeled crossover recombinations of essential support vectors (UL-SV Fict.). Results from (i) and (ii) are compared to (iii) the empirical initial credit data set as a benchmark with 658 clients without any fictitious data points.

Table 1: Evaluation and comparison of four SVM with different kernel functions. Each model is trained and evaluated on two fictitious data sets that were generated from labeled and unlabeled support vectors (L-SV Fict. and UL-SV Fict.). Real data models are trained and evaluated without using any additional fictitious data points. Tenfold cross validation error for all SVM models is evaluated for real data points only.

SVM-Kernel	No. of Cases	No. of SV	No. of BSV	Alpha Error	Beta Error	Total Error*
Linear						
L-SV Fict.	768	43	0	32.2%	22.4%	27.2%
UL-SV Fict.	2041	43	0	39.3%	27.2%	33.1%
Real data	658	41	316	29.1%	29.9%	29.5%
Polyn. 2nd deg.						
L-SV Fict.	1060	94	96	19.5%	31.6%	25.7%
UL-SV Fict.	2031	131	237	22.5%	31.6%	26.0%
Real data	658	63	392	27.2%	28.1%	27.7%
Polyn. 3rd deg.						
L-SV Fict.	3715	532	119	23.8%	30.5%	27.2%
UL-SV Fict.	2216	419	155	23.8%	27.4%	25.7%
Real data	658	216	211	27.9%	29.0%	28.4%
RBF						
L-SV Fict.	3113	419	97	25.7%	31.6%	25.7%
UL-SV Fict.	2179	327	149	28.5%	31.6%	26.0%
Real data	658	179	252	28.2%	28.1%	27.7%

*computed on the *N*=658 real data set only.

Data from type (i) and (iii) is used for supervised classification via inductive SVM, whereas data from type (ii) is used for semi-supervised classification via transductive SVM. Three classification measures are reported: The total error is the percentage of those clients classified incorrectly relative to all credit clients. The alpha (beta) error is the percentage of accepted bad (rejected good) relative to all bad (good) clients. In all cases error rates are recorded for 658 real credit clients only. We detect higher classification accuracy in six out of eight models trained with fictitious data when compared to the Real data performance. Labeled and unlabeled fictitious data points, however, do not seem to con-

ISBN: 978-960-474-311-7

tribute to out-of-sample performance in the same way.
Whereas inductive learning (supervised) leads to smaller error rates for every single model, transductive learning (semi-supervised) just improves two out of four models. The structure of the SVM models differs vastly as can be seen by the varying alpha and beta errors as well as by the varying numbers of support vectors of the different types.

5 Conclusion

After investigating the effect of the simplest placement of fictitious training points, namely seeding the vicinity of each training point with randomly drawn points, having the same label as the original data point (see also [11]) we now turn to the problem of whether new training points generated by semi-local methods do still express feasible domain data. This would in fact be the case (it would be indirectly confirmed) when the **validated** misclassification error can be reduced. First we generate fictitious data points by switching categories of nominal or ordinal variables.

In the context of supervised SVM model training the validation error computed on the original data decreases, when adding fictitious training data. Semi-supervised classification with more involved crossover recombinations of two parent support vectors also leads to improvement of some models. Classification results in this case, however, are ambiguous and need further investigation.

References:

[1] Y.S. Abu-Mostafa, Hints. *Neural Computation* vol. 7, 1995, pp. 639-671.

[2] S. Basu, I. Davidson, K.L. Wagstaff (eds.), *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, Chapman & Hall / CRC Press 2009.

[3] Y. Chen, G. Wang, and G. Dong, Learning with Progressive Transductive Support Vector Machine. *Second IEEE International Conference on Data Mining ICDM'02*, 2002

[4] R. Collobert, F. Sinz, J. Weston and L. Bottou, Large Scale Transductive SVMs. *Journal of Machine Learning Research*, vol. 7, 2006, pp. 1687-1712.

[5] D. DeCoste and B. Schölkopf, Training Invariant Support Vector Machines. *Machine Learning*, vol. 46, 2002, pp. 161-190.

[6] T. Joachims, Transductive Inference for Text Classification using Support Vector Machines. *International Conference on Machine Learning (ICML)*, 1999, pp. 200-209.

[7] T. Joachims, *Learning to classify text using support vector machines: methods, theory, and algorithms*. Kluwer, Boston 2002.

[8] K.B. Schebesch and R. Stecking, Support vector machines for credit applicants: detecting typical and critical regions. *Journal of the Operational Research Society*, vol. 56(9), 2005, pp. 1082-1088.

[9] B. Schölkopf, C. Burges and V. Vapnik, Incorporating Invariances in Support Vector Learning Machines. In: C. von derMalsburg et al. (eds.): *Artificial Neural Networks – ICANN96, Springer Lecture Notes in Computer Science*, Vol. 1112, Berlin 1996, pp. 47-52.

[10] R. Stecking and K:B. Schebesch, Comparing and Selecting SVM-Kernels for Credit Scoring. In: M. Spiliopoulou et al. (eds.), *From Data and Information Analysis to Knowledge Engineering*. Springer, Berlin 2006, pp. 542-549.

[11] R. Stecking and K.B. Schebesch, Improving Classifier Performance by Using Fictitious Training Data? A Case Study. In: Kalcsics, J., Nickel, S. (Eds.): *Operations Research Proceedings 2007*. Springer, Berlin 2008, pp. 89-94.

[12] V. Vapnik, *Statistical Learning Theory*. Wiley, New York 1998.