A C T A U N I V E R S I T A T I S L O D Z I E N S I S
FOLIA OECONOMICA 286, 2013

Michał Trzęsiok[*]

EXTRACTING CLASS DESCRIPTION FROM SUPPORT VECTOR MACHINES

Abstract. Support Vector Machines (SVMs) belong to the group of Data Mining and Machine Learning methods. SVMs are considered to be one of the best classification methods in terms of performance measure. The biggest disadvantage of SVMs is their lack of interpretability. Additional procedures can be applied that enable knowledge extraction. We present such a procedure that uses the information embedded in *support vectors* – the observations that define the classification function. We use recursive partitioning applied to support vectors to increase the interpretability of SVMs.

Keywords: Support Vector Machines, knowledge extraction, model interpretability.

I. INTRODUCTION

Support Vector Machines (SVMs) are the state-of-the-art classification method. Ever since they were introduced by Vapnik *et al.* (1992) they have attracted attention and are still developed by many scientists representing different fields. SVMs gained a very strong position within *Pattern Recognition* methods, due to their high classification accuracy level [see Abe (2005)], still their largest disadvantage is their lack of interpretability. There are different approaches to the problem of knowledge extraction from SVMs [see Diederich (2008)]. The paper presents a method that combines two independent techniques for increasing model interpretability – *backward elimination* for measuring the discrimination power of input variables, and *recursive partitioning* for extracting classification rules from SVMs. After building the SVMs classification model the first step towards extracting knowledge from the model is to reduce its complexity. This is done by performing feature elimination and finding the set of variables that would give comparable accuracy, but with only influential inputs [see Guyon *et al.* (2006), Trzęsiok (2009)]. The motivation for reducing the complexity of the model is natural, since its interpretability decreases with the complexity.

SVMs identify the *observations* that are crucial for the form of discrimination function. These observations are called *support vectors*. In the second step

[*] Ph.D., Department of Mathematics, Katowice University of Economics.

we replace the true classifications assigned to support vectors by the values predicted by SVMs model and train a *classification tree* (i.e. we build a recursive partitioning model) on such a dataset. This is how we force the classification tree to imitate the mechanics of SVMs. In the case of high similarity between what the classification tree has learned and SVMs predictions, the classification rules obtained from the tree may be used as a good representation of SVMs rules.

II. SVMs – THE SHORT OVERVIEW

In this section we briefly present the main ideas of Support Vector Machines. The comprehensive description can be found in Vapnik (1998), Cristianini and Shawe-Taylor (2000), or Abe (2005). In the case of the two-class classification, first the data points from the training set are transformed to a higher dimensional feature space by non-linear mapping. Then we find the optimal hyperplane (i.e. the hyperplane that maximises the margin) separating the images of the data in the feature space. This hyperplane (linear boundary) in the feature space corresponds with the nonlinear classifier in the original data space.

Following Cristianini and Shawe-Taylor (2000) we introduce the mathematical formulation of the SVMs' algorithm. In the case of the two-class classification problem we are given the training set $\left\{\left(\mathbf{x^1}, y^1\right), \ldots, \left(\mathbf{x^N}, y^N\right)\right\}$, where $\mathbf{x^i} \in \mathbf{R}^d$ and $y^i \in \{-1,1\}$, $i = 1, \ldots, N$. The goal of supervised learning is to find a "good" predictive classification function $y = f(\mathbf{x})$, based on the available training set. In order to obtain a model with a good generalization ability SVMs use the *structural risk minimization principle* [see Vapnik (1998)] as a criterion for finding a "good" decision function (that would not overfit training data). To handle a case when the two classes are not linearly separable, the non-linear mapping $\varphi : \mathbf{R}^d \to \mathbf{Z}$ is used. Thus we search for the optimal separating hyperplane:

$$\boldsymbol{\beta} \cdot \varphi(\mathbf{x}) + \beta_0 = 0, \tag{1}$$

where $\boldsymbol{\beta} \in \mathbf{Z}$, $\beta_0 \in \mathbf{R}$, in the new feature space **Z**. This hyperplane separates the classes and defines the decision function:

$$f(\mathbf{x}) = \operatorname{sign}\left(\boldsymbol{\beta} \cdot \varphi(\mathbf{x}) + \beta_0\right). \tag{2}$$

The problem of finding the optimal separating hyperplane (i.e. finding the normal vector $\boldsymbol{\beta}$ and the intercept β_0) can be formulated as a convex optimization problem with linear inequality constraints. The solution to this problem can be computed using the method of Lagrange multipliers and the decision function takes the final form:

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i y^i \mathrm{K}(\mathbf{x^i}, \mathbf{x}) + \beta_0 \right), \tag{3}$$

where $\mathrm{K}(\mathbf{u}, \mathbf{v}) = \varphi(\mathbf{u}) \cdot \varphi(\mathbf{v})$ is a kernel function that represents an inner product in the feature space.

The Karush-Kuhn-Tucker conditions imply that many of the Lagrange multipliers α_i in the solution (3) are equal zero. Since the decision function (3) uses the linear combination of the images of the observations, only the observations corresponding to nonzero Lagrange multipliers have discriminative power. These observations are called *support vectors*.

In the case of *m* classes ($m \geq 3$), the most common approach is to build $\frac{m(m-1)}{2}$ binary classifiers (one-against-one multi-class SVMs) and use the majorization scheme (voting) when classifying a new observation.

III. MODEL COMPLEXITY REDUCTION AND CLASSIFICATION RULES EXTRACTION

As mentioned in the introduction in order to extract knowledge from SVMs we perform an additional procedure that is based on two steps: model complexity reduction by backward elimination [as presented in Trzęsiok (2009)], and classification rule extraction with the use of recursive partitioning [the description of the recursive partitioning method, known also as *classification trees* can be found in Gatnar (2001)]. In more detail the recursive partitioning method is applied to the set of support vectors with class assignment replaced by the SVMs model predictions. The biggest advantage of using recursive partitioning is that the model has a clear interpretation that may be presented in two forms: textual (as classification rules written in natural language) and graphical (as classification tree). We present the whole knowledge extraction procedure in the form of an algorithm in Table 1.

Table 1. Algorithm for knowledge extraction from Support Vector Machines

Step 1.	**Model complexity reduction by backward elimination procedure:** a) Tune the SVMs model on training set *D* using all the predictors. b) Apply the backward elimination procedure (an iterative procedure where we start with all the features and delete one feature at a time) with the cross-validation (*CV*) classification error computed in every iteration [as in Trzęsiok (2009)]. c) As a result of b) you get a sequence of models with decreasing number of input variables. Identify the model with the smallest number of input variables, which has a *CV* classification error less or equal $\min(CVerr) + SE$, where $\min(CVerr)$ is the minimal classification error in the sequence and *SE* is the standard error of this estimator. d) Consider the variables not included in the model identified in c) as *redundant variables* and delete them from dataset *D*. Build the SVMs model on the reduced dataset D' and proceed to the next step.
Step 2.	**Classification rules extraction based on recursive partitioning results:** e) Identify the support vectors from the model SVMs built in d). f) Replace the true class assignment for the observations (support vectors) from e) by SVMs model predictions (from d)). Denote this new set by D''_{SV}. g) Apply the recursive partitioning on dataset D''_{SV}. h) Check the classification agreement between classification tree from g) and SVMs predictions for the support vectors. If the agreement is not met significantly, stop the procedure and use different approach to knowledge extraction, otherwise consider the classification tree to be a good representation of SVMs model and proceed with the interpretation. i) Since the classification tree in g) is built on SVMs fitted values for the influential observations (support vectors) it imitates the classification rules embedded in SVMs. Interpret the rules obtained from the classification tree as representing SVMs model.

Source: own results.

Building the classification tree on the dataset including support vectors only, means that we use only the knowledge embedded in the influential observations, thus we do not take into account information that is irrelevant or information repeated (doubled) in many objects in the dataset. It is beneficial to examine precisely only the information which is essential for the given classification task.

IV. AN EXAMPLE ILLUSTRATING THE PROCEDURE

We use a real-world dataset `german credit` shared by prof. dr hab. Hans Hofmann from the Institute of Statistics and Econometrics in University of Hamburg. This dataset set is available in the UCI Repository of Machine Learning Databases[1] (University of California, Irvine). The dataset includes information about short term loans (taken in German currency [DM]). The task is the classical *credit scoring* problem – given a dataset representing the credit history of 1000 bank customers, find the classification function that would classify a new client into one of two groups: "good clients" who represent low credit risk and "bad clients" with high credit risk. This function should be an automatic support for the decision making process whether or not accept an application form for granting a loan. The general information about the analysed dataset is presented in Table 2.

Table 2. General information about the German credit dataset

No. of observations	No. of input variables	
	interval	nominal
1 000	7	13

Source: own results.

The set of input variables consist of: status of the checking account, loan duration in month, credit history (no credits taken, all credits paid back duly, delay in paying off in the past, other credits existing – not at this bank), purpose of the loan, credit amount, savings account/bonds, present employment since, instalment rate in percentage of disposable income, personal status and sex, other debtors/guarantors, present residence since, property, age in years, other instalment plans, housing, number of existing credits at this bank, job, no. of people being liable to provide maintenance for, telephone, foreign worker. The dependent variable is a categorical one and has two levels: "good" and "bad".

Because the dataset included some categorical inputs, these variables were transformed and represented by dummy variables. Thus the objects in the analysed training set were described by 7 interval input variables and 54 categorical predictors (some of them – dummy variables). In the first step the SVMs model with the complete set of inputs was built. Then the backward elimination procedure allowed to reduce the number of inputs (61) by 42. It means that the model built on the specific group of 19 inputs has classification accuracy similar to the saturated model (with 61 inputs). Since the ranking of predictors is long, we only mention here that the input with largest discrimination power is the dummy variable reflecting *negative status of checking account*. The SVMs model built on the 19 inputs identified 568 support vectors. This number is relatively large compared with the dataset size (1000), but this is normal when there are so many categorical inputs. These 568 objects with the class assignment replaced by the

[1] ftp://ftp.ics.uci.edu/pub/machine-learning-databases, http://mlearn.ics.uci.edu/

SVMs predictions created a new training set D''_{SV} which were further analysed by applying recursive partitioning. The classification agreement between classification tree and SVMs predictions is equal 91%, thus the tree was considered to represent well the SVMs.

The classification rules for SVMs obtained from the classification tree (after pruning, i.e. after reducing the complexity of the tree) are presented in Table 3. and Figure 1.

Remembering that going left on the tree means that the condition on that level is met, makes it easy to write down the classification rules, e.g. "*If* the customer has a negative status of his/her checking account *and* has no credits at other banks *and* the purpose of the loan is not furniture *nor* a used car *then* the credit risk is too high (bad credit) and the application for the loan should be rejected". Another example of the rule is "*If* the customer has a negative status of his/her checking account *and* he/she has got credits at other banks *and* the credit amount is less than 7 340 DM *then* the credit risk is low (good credit) and the credit should be granted". The alternative of every rule that is represented by a path on the tree from the root to the leaf "good" is the description of the class "good".

Table 3: Textual form of classification rules for SVMs obtained by the use of classification tree imitating what SVMs have learned from the training set `german credit` (the rule extraction included only the influential input variables and influential observations [support vectors])

```
n= 568
node), split, n, loss, yval, (yprob)
      * denotes terminal node
1) root 568 162 good (0,285 0,715)
2) negative.status.of.checking.account>=0.5 222  94 bad (0,577 0,423)
4) history_credits.at.other.banks< 0.5 175  59 bad (0,663 0,337)
8) credit.purpose_furniture< 0.5 125  34 bad (0,728 0,272)
16) credit.purpose_used.car< 0.5 112  25 bad (0,777 0,223) *
17) credit.purpose_used.car>=0.5 13   4 good (0,308 0,692) *
9) credit.purpose_furniture>=0.5 50  25 bad (0,500 0,500)
18) credit.amount>=3989.5 11   1 bad (0,909 0,091) *
19) credit.amount< 3989.5 39  15 good (0,385 0,615) *
5) history_credits.at.other.banks>=0.5 47  12 good (0,255 0,745)
10) credit.amount>=7340 4   1 bad (0,750 0,250) *
11) credit.amount< 7340 43   9 good (0,209 0,791) *
3) negative.status.of.checking.account< 0.5 346  34 good (0,098 0,902)
6) credit.amount>=10294 20  10 bad (0,500 0,500)
12) history_credits.paid.back.duly>=0.5 11   2 bad (0,818 0,182) *
13) history_credits.paid.back.duly< 0.5 9   1 good (0,111 0,889) *
7) credit.amount< 10294 326  24 good (0,074 0,926) *
```

Source: own results.

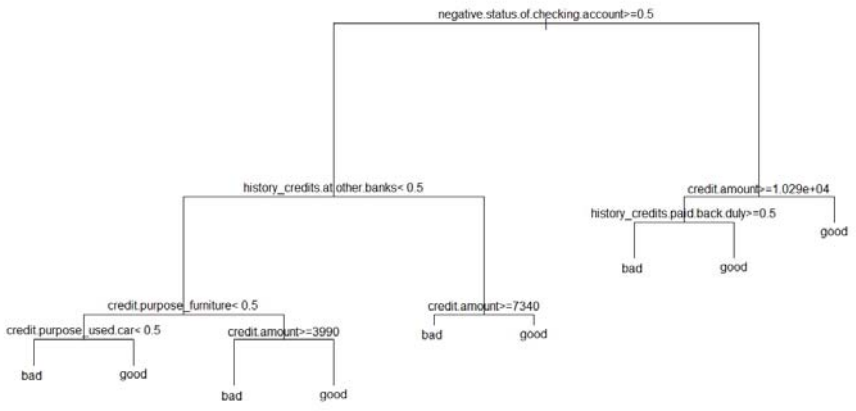
negative.status.of.checking.account>=0.5
history_credits.at.other.banks< 0.5
credit.amount>=1.029e+04
history_credits.paid.back.duly>=0.5
good
bad
good
credit.purpose_furniture< 0.5
credit.amount>=7340
credit.purpose_used.car< 0.5
credit.amount>=3990
bad
good
bad
good
bad
good

Fig. 1: Graphical form of classification rules for SVMs obtained by the use of classification tree imitating what SVMs have learned from the training set `german credit` (the rule extraction included only the influential input variables and influential observations [support vectors])

Source: own results.

V. CONCLUSION

Support Vector Machines are very powerful classification method, but they are considered to build black-box models. Sometimes it is very important not only to have a prediction tool with high accuracy level, but also a model that can be interpreted and give some knowledge about the analysed phenomenon. In the paper we have presented a procedure for knowledge extraction from SVMs. The procedure combines model complexity reduction by backward elimination and classification rules extraction by recursive partitioning. The main motivation of combining SVMs with classification trees was to get advantage from the recursive partitioning which is the graphical form of the model, that enables clear interpretation of classification rules. The example on the real-world dataset from the credit scoring area demonstrates the usefulness of the procedure.

REFERENCES

Abe S. (2005), *Support Vector Machines for Pattern Classification*, Springer.

Boser B., Guyon I., and Vapnik V. (1992). *A training algorithm for optimal margin classifiers*. In Haussler D. (ed.), 5th Annual ACM Workshop on COLT, ACM Press, Pittsburgh, p. 144-152.

Cristianini N., Shawe-Taylor J. (2000), *An Introduction to Support Vector Machines (and other kernel-based learning methods)*, Cambridge University Press, Cambridge.

Diederich J. (ed.) (2008), *Rule Extraction from Support Vector Machines*, Springer.

Gatnar E. (2001), *Nonparametric Method for Classification and Regression* (in Polish), PWN, Warszawa.
Guyon I., Gunn S., Nikravesh M., Zadeh L. (eds) (2006), *Feature Extraction, Foundations and Applications*. Springer.
Trzęsiok M. (2009), *Variable Selection in Support Vector Machines* (in Polish), In: Jajuga K., Walesiak M. (eds), „*Taksonomia 16*", Wrocław University of Economics Publishing House, Wrocław, p. 214-222.
Vapnik V. (1998), *Statistical Learning Theory*, John Wiley & Sons, N.Y.

Michał Trzęsiok

PROFILOWANIE KLAS W METODZIE WEKTORÓW NOŚNYCH

Metoda wektorów nośnych (SVM) należy do grupy statystycznych metod uczących się. Jak większość metod z tej grupy, metoda SVM buduje modele o bardzo dobrych własnościach predykcyjnych, lecz niewielkiej interpretowalności. W celu uzyskania dodatkowej wiedzy –stosuje się dodatkowe procedury wspomagające interpretowanie wyników modelowania. W artykule przedstawiono procedurę wykorzystującą informacje zawarte w *wektorach nośnych* – obserwacjach istotnie wpływających na postać wyznaczonej funkcji dyskryminującej. Intepretowalność modelu końcowego uzyskano dzięki zastosowaniu modelu rekurencyjnego podziału do dyskryminacji wyznaczonych wektorów nośnych.