

Symbolic data transformations for continuous data oriented models

Krzysztof Grąbczewski & Norbert Jankowski

Department of Informatics, Nicolaus Copernicus University
ul. Grudziądzka 5, 87-100 Toruń, Poland
{kgrabcze|norbert}@phys.uni.torun.pl, <http://www.phys.uni.torun.pl/kmk>

Abstract. Most of Computational Intelligence models (e.g. neural networks or distance based methods) are designed to operate on continuous data and provide no tools to adapt their parameters to data described by symbolic values. Two new conversion methods which replace symbolic by continuous attributes are presented and compared to two commonly known ones. The advantages of the continuousification are illustrated with the results obtained with a neural network, SVM and a kNN systems for the converted data.

1 Introduction

The majority of the Computational Intelligence (CI) systems are designed to deal with continuous data. The adaptive processes of neural networks and most of similarity based models operate in R^n space to perform their approximation or distance calculation tasks. Building such models for data described by symbolic attributes requires an embedding of the sets of symbols into some sets of real numbers. The simplest (and most commonly used) mapping arbitrarily replaces subsequent symbolic values with subsequent natural numbers. The order of the symbols is random and different randomizations may lead to significantly different results obtained with CI systems, so finding an appropriate mapping from symbols to real numbers is mostly advisable.

A simple way to get rid of symbolic features is to replace each of them by a number of binary features.

Distance based systems may use some similarity measures which are designed for symbolic data like Value Difference Metric (VDM), Heterogeneous Euclidean-Overlap Metric [13] or Minimum Risk Metric [2].

A simple observation that instead of using VDM metric, one can replace each symbolic value with a number of probabilities and use Minkovski measure on the converted data, leads to a conclusion that thanks to the data conversion any system designed for continuous data may also take advantage of the VDM measure [7].

Using another conditional probability yields very similar (but of different behavior in applications) MDV continuousification scheme.

A conversion from symbolic to continuous data can also be done with the usage of a separability criterion dedicated for decision tree systems.

CI systems augmented by some continuousification methods become yet more powerful tools, which take advantage of different methods of information extraction and offer very good predictive accuracy (see section 5) which can be significantly better (see sections 4 and 5) than arbitrary coding of symbols.

2 Continuousification methods

The arbitrary coding of symbols may lead to very different orders and distances between attribute values. As a consequence of that the placement of the training data in the feature space is different for each mapping and this significantly affects the distances between the data vectors. The goal of continuousification is to find such a representation of symbols in the set of real numbers, that makes the classification or approximation problem easier.

NBF continuousification Some of CI systems convert symbolic features consisting of n symbols with n binary features (NBF). For each $i = 1, \dots, n$ the i 'th new feature indicates whether the value of the original feature of given vector is the i 'th symbol or not. Such data conversion results in a dataset of dimensionality strongly dependent on the numbers of symbols representing the features and does not depend on the number of classes (in opposition to the VDM and MDV methods presented below).

VDM continuousification In the space of symbolic features $X = X_1 \times \dots \times X_n$, for the set of classes $C = \{c_1, \dots, c_k\}$ the Value Difference Metric (VDM) for $x, y \in X \times C$ and a parameter q is defined as:

$$D_{\text{VDM}}^q(x, y) = \sum_{i=1}^n \sum_{j=1}^k |P(c_j|X_i = x_i) - P(c_j|X_i = y_i)|^q \quad (1)$$

where $P(c_j|X_i = z_i)$ is a shortened form of $P(C(u) = c_j|u_i = z_i \wedge u \in X \times C)$.

Duch *et al.* [7] presented the idea of mapping each of the x_i symbolic values of a vector x with k real numbers $P(c_1|X_i = x_i), \dots, P(c_k|X_i = x_i)$.

In two class problems the dimensionality of the space may remain unchanged, since only one probability may be used without any change of the relative distances between data vectors (using two probabilities instead of one would just double each distance).

MDV continuousification A natural alternative to VDM continuousification is to use the other conditional probability binding the feature values with class labels. Replacing $P(c_j|X_i = x_i)$ by $P(X_i = x_i|c_j)$ we obtain a mapping of an idea similar to VDM (hence we call it MDV), but with several important differences.

The main dissimilarity is that in the case of VDM the feature values are ordered according to how specific they are for given class and in the case of

MDV according to the frequency of given value among the vectors belonging to the class.

The nature of VDM causes that in two class tasks, only one of the probabilities is necessary to preserve the whole information of two values. MDV's nature is different - both probabilities are not so closely related – they are different sources of information, however assuming some similarity one can use probabilities for one of the classes to reduce the dimensionality of the resulting data.

SSV criterion and SSV based continuousification The SSV criterion is one of the most efficient among criteria used for decision tree construction [8,9]. It's basic advantage is that it can be applied to both continuous and discrete features. The *split* value (or *cut-off point*) is defined differently for continuous and symbolic features. For continuous features it is a real number and for symbolic ones it is a subset of the set of alternative values of the feature. The *left side* (*LS*) and *right side* (*RS*) of a split value s of feature f for a given dataset D is defined as:

$$\begin{aligned} \text{LS}(s, f, D) &= \begin{cases} \{x \in D : f(x) < s\} & \text{if } f \text{ is continuous} \\ \{x \in D : f(x) \notin s\} & \text{otherwise} \end{cases} \\ \text{RS}(s, f, D) &= D - \text{LS}(s, f, D) \end{aligned} \quad (2)$$

where $f(x)$ is the f 's feature value for the data vector x . The definition of the *separability of a split value* s is:

$$\begin{aligned} \text{SSV}(s) &= 2 * \sum_{c \in C} |\text{LS}(s, f, D) \cap D_c| * |\text{RS}(s, f, D) \cap (D - D_c)| \\ &\quad - \sum_{c \in C} \min(|\text{LS}(s, f, D) \cap D_c|, |\text{RS}(s, f, D) \cap D_c|) \end{aligned} \quad (3)$$

where C is the set of classes and D_c is the set of data vectors from D which belong to class $c \in C$. A similar criterion has been used for design of neural networks by Bobrowski *et al.* [3].

Decision trees are constructed recursively by searching for best splits (with the largest SSV value) among all the splits for all the features. At each stage when the best split is found and the subsets of data resulting from the split are not completely pure (i.e. contain data belonging to more than one class) each of the subsets is being analyzed in the same way as the whole data. The decision tree built this way gives maximal possible accuracy (100% if there are no contradictory examples in the data) which usually means that the created model overfits the data. To remedy this a cross validation training is performed to find the optimal parameters for pruning the tree. Optimal pruning produces a tree capable of good generalization of the patterns used in the tree construction process.

SSV based continuousification The SSV criterion can also be a successful tool for symbolic to real-valued feature mapping. The following algorithm presents the method:

Algorithm 1 (SSV based continuousification)

Input: The classification space X , set of classes C , training set $T \subseteq X \times C$, symbolic feature F (of space X).

Output: Mapping $F \rightarrow R$.

1. Build a decision tree D using $T' = \{(x_F, c) \in F \times C : (x, c) \in T\}$ where x_F is the value of the feature F for vector x .
2. For each node W of the tree D , such that W is not the root or a direct subnode of the root, calculate SSV_W as the SSV criterion value for the split between W and the sibling of its parent (the split of the set of vectors belonging to the two nodes, to the two sets determined by the nodes).
3. For each node W of the tree D , such that W is not the root or a leaf (starting with the root's direct subnodes, through their subnodes to the parents of the leaves) order the children W_i of W :
 - with decreasing (from left to right) values of SSV_{W_i} if W is the left child of its parent.
 - with increasing (from left to right) values of SSV_{W_i} if W is the right child of its parent.
4. Create the list L_1, \dots, L_n of all the leaves of the tree D with the order of visiting them with the depth first search (where the left child is visited before the right one).
5. Calculate the criterion values $\text{SSV}_{i,i+1}$ for $i = 1, \dots, n-1$ for the pairs of leaves that are neighbors in the list.
6. For each $i = 1, \dots, n$ assign to the L_i leave the real value

$$\frac{\sum_{j=1}^{i-1} \text{SSV}_{j,j+1}}{\sum_{j=1}^{n-1} \text{SSV}_{j,j+1}}. \quad (4)$$

7. The output is the mapping which maps each possible value f of the F feature to the real number calculated in the preceding step for the leaf which contains the vectors with f value of the F feature.

The algorithm takes advantage of the fact, that SSV tree usually puts each symbolic value of the feature into a separate leaf. Two different values may end up in a single leaf only if all the vectors with any of that values belong to the same class – in such a case the two values (from the classification point of view) need not be distinguished and the above algorithm maps them to the same real value.

3 Adaptive models tested

We have tested the algorithms with three different kinds of adaptive models: a neural network (FSM), an SVM and a minimal distance system (kNN).

FSM neural network. Feature Space Mapping (FSM) is a neural network system based on modelling probability distribution of the input/output data vectors [6,1]. The learning algorithm facilitates growing and shrinking of the network structure, which makes the method flexible and applicable to classification problems of miscellaneous domains.

SVM method. The SVM algorithm we have used is the Platt’s Sequential Minimal Optimization (SMO) [12] augmented by the ideas presented in [10]. Such version yields very fast and accurate solutions.

kNN method. The *k Nearest Neighbours (kNN)* algorithm we used is a method with automated selection of the *k* parameter. For given training dataset the *k* is determined by means of cross validation training performed inside the training set (the winner *k* is the one that gives the smallest average validation error).

4 Statistical significance test

When comparing the performances of different classification systems it is important not only to see the average accuracies, but to answer the question of the probability, that the average accuracy of a number of tests for one system will be higher than the average for the other. Assuming normal distribution of the accuracies, we estimate the probability with Student’s *t* test [4,5].

In our experiments we repeated a cross validation (CV) test 10 times. Each competing system was run for the same data sample, so we are justified to estimate the statistical significance with paired *t* test with 10 degrees of freedom. The estimation of the variance of the CV mean is done on the basis of the 10 results.

5 Results

There is no point in continuousification of binary features, so the datasets containing continuous and binary features only are not eligible for the test. Also the results obtained for a dataset containing just one or two symbolic features do not allow for any conclusions.

We have tested the continuousification algorithms on three datasets from the UCI repository [11], defined in spaces consisting of symbolic features only: *Promoters*, *Soybean* and *DNA*. The tables 1 and 2 present the results of the 10 repetitions of 10 fold CV for each of the tested models, and table 3 shows average results for 10 training and test runs (the data is divided into training and test parts). The first row of each table shows the result obtained on raw data (arbitrary coding of symbols) while the other rows show the results with continuousification noted in the first column. A “k” before the method’s name means that *one-against-rest* technique was used, i.e. the data was classified by a committee of *k* experts (where *k* is the number of classes) – each specializing in the recognition of one of the classes against all the others. For each continuousifier the columns P_1, P_2, \dots show the probabilities (calculated with the *t* test) that its averaged test accuracy is higher then that of method 1, 2, etc.

FSM	Acc.	Std.dev.	P_1	P_2	P_3	P_4	P_5
1: None	0.673	0.034	—	0.000	0.000	0.000	0.000
2: SSV	0.878	0.029	1.000	—	0.175	0.308	0.540
3: NBF	0.912	0.013	1.000	0.825	—	0.734	0.968
4: VDM	0.893	0.024	1.000	0.692	0.266	—	0.706
5: MDV	0.874	0.017	1.000	0.460	0.032	0.294	—
SVM	Acc.	Std.dev.	P_1	P_2	P_3	P_4	P_5
1: None	0.478	0.014	—	0.000	0.000	0.000	0.000
2: SSV	0.903	0.015	1.000	—	1.000	0.091	0.045
3: NBF	0.695	0.040	1.000	0.000	—	0.000	0.000
4: VDM	0.930	0.016	1.000	0.909	1.000	—	0.376
5: MDV	0.936	0.006	1.000	0.955	1.000	0.624	—
kNN	Acc.	Std.dev.	P_1	P_2	P_3	P_4	P_5
1: None	0.725	0.026	—	0.001	0.069	0.000	0.000
2: SSV	0.860	0.020	0.999	—	0.992	0.008	0.014
3: NBF	0.771	0.022	0.931	0.008	—	0.000	0.000
4: VDM	0.929	0.019	1.000	0.992	1.000	—	0.587
5: MDV	0.924	0.011	1.000	0.986	1.000	0.413	—

Table 1. Results for Promoters (106 instances, 57 attributes, 2 classes).

FSM	Acc.	Std.dev.	P_1	P_2	P_3	P_4	P_5		
1: None	0.868	0.010	—	0.539	0.012	0.434	0.261		
2: SSV	0.867	0.007	0.461	—	0.000	0.407	0.198		
3: NBF	0.894	0.006	0.988	1.000	—	0.987	0.926		
4: VDM	0.870	0.010	0.566	0.593	0.013	—	0.320		
5: MDV	0.877	0.013	0.739	0.802	0.074	0.680	—		
SVM	Acc.	Std.dev.	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1: None	0.664	0.007	—	0.000	0.000	0.000	0.845	1.000	0.000
2: SSV	0.762	0.007	1.000	—	0.001	0.999	1.000	1.000	0.001
3: NBF	0.787	0.007	1.000	0.999	—	1.000	1.000	1.000	0.134
4: VDM	0.729	0.007	1.000	0.001	0.000	—	1.000	1.000	0.000
5: MDV	0.656	0.005	0.155	0.000	0.000	0.000	—	1.000	0.000
6: k VDM	0.487	0.007	0.000	0.000	0.000	0.000	0.000	—	0.000
7: k MDV	0.796	0.005	1.000	0.999	0.866	1.000	1.000	1.000	—
kNN	Acc.	Std.dev.	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1: None	0.831	0.006	—	0.000	0.000	0.000	0.000	0.000	0.000
2: SSV	0.894	0.005	1.000	—	0.039	0.003	0.011	0.953	0.517
3: NBF	0.909	0.005	1.000	0.961	—	0.059	0.415	0.992	0.942
4: VDM	0.923	0.007	1.000	0.997	0.941	—	0.938	0.999	0.999
5: MDV	0.910	0.004	1.000	0.989	0.585	0.062	—	0.991	0.958
6: k VDM	0.878	0.008	1.000	0.047	0.008	0.001	0.009	—	0.086
7: k MDV	0.894	0.008	1.000	0.483	0.058	0.001	0.042	0.914	—

Table 2. Results for Soybean (290 instances, 35 attributes, 15 classes).

FSM	Acc.	Std.dev.	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1: None	0.906	0.007	—	0.001	0.000	0.000	0.000	0.000	0.000
2: SSV	0.936	0.004	0.999	—	0.058	0.007	0.060	0.005	0.099
3: NBF	0.948	0.005	1.000	0.942	—	0.620	0.715	0.238	0.500
4: VDM	0.946	0.002	1.000	0.993	0.380	—	0.699	0.052	0.405
5: MDV	0.944	0.003	1.000	0.940	0.285	0.301	—	0.032	0.282
6: k VDM	0.953	0.003	1.000	0.995	0.762	0.948	0.968	—	0.721
7: k MDV	0.948	0.007	1.000	0.901	0.500	0.595	0.718	0.279	—
SVM	Acc.	Std.dev.	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1: None	0.611	0.000	—	0.000	0.000	0.000	0.000	0.000	0.000
2: SSV	0.927	0.000	1.000	—	1.000	0.000	0.000	0.000	1.000
3: NBF	0.633	0.000	1.000	0.000	—	0.000	0.000	0.000	0.000
4: VDM	0.948	0.000	1.000	1.000	1.000	—	1.000	1.000	1.000
5: MDV	0.947	0.000	1.000	1.000	1.000	0.000	—	1.000	1.000
6: k VDM	0.935	0.000	1.000	1.000	1.000	0.000	0.000	—	1.000
7: k MDV	0.895	0.000	1.000	0.000	1.000	0.000	0.000	0.000	—
kNN	Acc.	Std.dev.	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1: None	0.676	0.001	—	0.000	0.000	0.000	0.000	0.000	0.000
2: SSV	0.876	0.003	1.000	—	1.000	0.000	0.000	0.000	0.000
3: NBF	0.827	0.004	1.000	0.000	—	0.000	0.000	0.000	0.000
4: VDM	0.949	0.001	1.000	1.000	1.000	—	0.883	0.000	0.889
5: MDV	0.947	0.003	1.000	1.000	1.000	0.117	—	0.001	0.822
6: k VDM	0.958	0.001	1.000	1.000	1.000	1.000	0.999	—	0.991
7: k MDV	0.941	0.006	1.000	1.000	1.000	0.111	0.178	0.009	—

Table 3. Results for DNA (2000 instances for training, 1186 for test, 60 attributes, 3 classes).

6 Conclusions

The presented results clearly show that commonly used continuousification methods do not perform very well. Whether the results are good is the matter of luck. The VDM method as well as the new MDV and SSV methods are significantly more reliable. The need for appropriate data preparation confirms, that combinations of different kinds of information retrieval (hybrid methods) are necessary to obtain good results.

All the presented algorithms are fast, however NBF, VDM and MDV may produce high dimensional data, which may significantly slower the learning of the final models. In the case of Soybean data consisting of 35 features NBF produced 97 new features and each of VDM and MDV 525 features. For DNA data (60 features) NBF gave 240 features and each of VDM and MDV 180 features.

Instead of producing large spaces with VDM or MDV, sometimes it is reasonable to use the *one-against-rest* technique - although it requires the final classifier to be trained several times, it may be faster than training a single final model - it depends on how efficient the model is in high dimensional spaces.

The SSV method does not enlarge the dataset by features multiplication, regardless the number of classes and feature values. Hence it is very efficient for complex data. Although some other continuousification methods may give higher accuracies, the difference is usually small in comparison to the difference between SSV and an arbitrary symbols coding.

It must be pointed out, that the VDM, MDV and SSV methods use the information about the classes of the vectors, so in the case of the tests like cross validation it should not be used at the stage of data preprocessing. It must be run separately for each fold of the test. Used at the preprocessing stage they yield overoptimistic results.

References

1. R. Adamczak, W. Duch, and N. Jankowski. New developments in the feature space mapping model. In *Third Conference on Neural Networks and Their Applications*, pages 65–70, Kule, Poland, October 1997. Polish Neural Networks Society.
2. E. Blanzieri and F. Ricci. Advanced metrics for class-driven similarity search. In *Proceedings of the International Workshop on Similarity Search*, Firenze, Italy, September 1999.
3. L. Bobrowski, M. Krętowska, and M. Krętowski. Design of neural classifying networks by using dipolar criterions. In *Third Conference on Neural Networks and Their Applications*, Kule, Poland, October 1997.
4. S. Brandt. *Data Analysis*. Springer, New York, 1999.
5. T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.
6. W. Duch and G. H. F. Dierksen. Feature space mapping as a universal adaptive system. *Computer Physics Communications*, 87:341–371, 1995.
7. W. Duch, K. Grudziński, and G. Stawski. Symbolic features in neural networks. In *Proceedings of the 5th Conference on Neural Networks and Their Applications*, pages 180–185, Zakopane, Poland, June 2000.
8. K. Grąbczewski and W. Duch. A general purpose separability criterion for classification systems. In *Proceedings of the 4th Conference on Neural Networks and Their Applications*, pages 203–208, Zakopane, Poland, June 1999.
9. K. Grąbczewski and W. Duch. The separability of split value criterion. In *Proceedings of the 5th Conference on Neural Networks and Their Applications*, pages 201–208, Zakopane, Poland, June 2000.
10. S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt’s SMO algorithm for SVM classifier design. *Neural Computation*, 13:637–649, 2001.
11. C. J. Merz and P. M. Murphy. UCI repository of machine learning databases, 1998. <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
12. J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA., 1998.
13. D.R. Wilson and T.R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 11:1–34, 1997.