Conclusion

In the present paper an approach for optimal constrained control based on using of neural networks is suggested.

On the basis of the simulation experiments one can say that the proposed approach for optimal control is accurate enough for the engineering practice. The suggested approach can be applied for optimal control in real time, where the control is constrained.

Bibliography

[Ahmed 1998] M.S. Ahmed and M.A. Al-Djani. Neural regulator design. Neural Networks, Vol. 11, pp. 1695-1709, 1998

[Chjan 1961] Zh.-V.Chjan. A problem of optimal system synthesis through the maximum principle. Automatization and telemechanics, Vol. XXII, No.1, 1961, pp. 8-12

[Haykin 1999] S. Haykin. Neural Networks: A Comprehensive Foundation., 1999, 2nd ed, Macmillan College Publishing Company, New York

[Lewis 2002] F.L. Lewis and M. Abu-Khalaf. A Hamilton-Jacobi setup for constrained neural network control. Proceedings of the 2003 IEEE International Symposium on Intelligent Control Houston, Texas *October 5-8. 2003

[Pontryagin 1983] L.S. Pontryagin, V.G. Boltyanskiy, R.V. Gamkrelidze and E.F.Mishtenko. Methematical theory of the optimal processes. Nauka, Moskow, 1983, in Russian

Authors' Information

Georgi Toshkov –e-mail: g_toshkov2006@abv.bg
Daniela Toshkova – e-mail: daniela_toshkova@abv.bg
Technical University of Varna, 1, Studentska Str, Varna, 9010, Bulgaria
Todorka Kovacheva – Economical University of Varna, Kniaz Boris Str, e-mail: todorka_kovacheva@yahoo.com

LINEAR CLASSIFIERS BASED ON BINARY DISTRIBUTED REPRESENTATIONS

Dmitri Rachkovskij

***Abstract**: Binary distributed representations of vector data (numerical, textual, visual) are investigated in classification tasks. A comparative analysis of results for various methods and tasks using artificial and real-world data is given.*

***Keywords**: Distributed representations, binary representations, coarse coding, classifiers, perceptron, SVM, RSC*

***ACM Classification Keywords**: C.1.3 Other Architecture Styles - Neural nets*, I.2.6 Learning - Connectionism and neural nets, Induction, Parameter learning

Introduction

Classification tasks consist in assigning input data samples to one or more classes from a predefined set [1]. Classification in the inductive approach is realized on the basis of a training set containing labeled data samples. Usually, input data samples are represented as numeric vectors. Vector elements are real numbers (e.g., some measurements of object characteristics or their function) or binary values (indicators of some features in the input data).

This vector information often isn't explicitly relevant to the classification, therefore some kind of transformation is necessary. We have developed methods for transformation of input information of various kinds (such as numerical [2], textual [3], visual [4]) to binary distributed representations. These representations can then be classified by linear classifiers – such as *SVM* [5] or more computationally effective and naturally handling multiple classes perceptron-like classifiers [4, 6]. The objective of this paper is to investigate efficiency of the proposed methods for distributed information representation and classification using real and artificial data of different modalities.

Numeric vector data classification

For an experimental research of the abovementioned methods on numeric data the following well-known test problems have been selected: Leonard-Kramer *LK*, *XOR*, *Double Spiral*; datasets generated by *DataGen* [6]; and sample data from the *Elena* database [7]. The dimensionality A of data vectors varied from 2 to 36, number of classes C varied from 2 to 11, and the number of samples in the training and test sets varied from 75 to 3218.

All selected problems have essentially non-linear class boundaries. Therefore, non-linear transformation of input numeric vectors has been used - i.e., *RSC* and *Prager* [2] methods of encoding. Those methods extract binary features – indicators of input A-dimensional vector presence in s-dimensional ($s<A$) hyperrectangle receptive fields with random position and size.

To investigate the impact of code parameters on the classification quality, we chose the following experimental scheme. Input vectors were converted to *RSC* and *Prager* codes. Those codes were then used as input data for training and testing linear classifiers. The number (or percent) of test errors was chosen as a classification quality criterion. We used *SVM* [5] and modifications of perceptron-like classifiers [4] as linear classifiers for the obtained distributed representations. Besides, classification experiments with (non-linear) kernel *SVM* using *Prager*, *RSC* [2] and standard (Gaussian and polynomial) kernels were conducted.

It is well known [5] that *SVM* doesn't support online learning, requires solving computationally expensive non-linear programming problems, and constructs optimal separating hyperplane for two-class problems only. In this work we have also used a perceptron with an enlarged margin and multi-class learning rule developed by I.Misuno in order to overcome *SVM* drawbacks. In the resulting perceptron outputs of neurons that correspond to classes are determined as $y_c = \Sigma_i x_i w_{ic}$, where w_{ic} are the weights of modifiable connections, x_i is the i-th element value of the vector input to the connections. (In the present context x is the binary vector obtained by input transformation to distributed representation, but the original data vector could be used for linear tasks as well). For the "true" class neuron $y_{c\text{-}true} = y_{c\text{-}true}(1-T)$, where $0<T<1$ is the "defense margin parameter". The classification output is the index c^* of neuron with the maximum activation: $c^* = \text{argmax}_c\ y_c$. In case of an error ($c^* \neq c_{true}$) connections are modified in the following way: $w_{ic} = w_{ic} + \Delta w$ for $c=c_{true}$ and $w_{ic} = w_{ic} - f(\Delta w)$ for c: $y_c > y_{c\text{-}true}$, where c_{true} is the index of the correct class. E.g., $f(\Delta w) = \Delta w/|c|$. Our previous version of the enlarged margin perceptron had single-class (not multi-class) learning rule: unlearning with single class $c^* = \text{argmax}_c\ y_c$ was performed in case of error, and $f(\Delta w) = \Delta w$. For T=0 and single-class learning rule one obtains usual percepton, while for T=0 and multi-class learning rule one obtains usual percepton with multi-class learning. Multi-class learning extracts and uses more information from a single error and so provides a potential for faster learning and better generalization for essentially multi-class tasks, especially at early learning iterations of the training set. This can be critical for the on-line learning tasks.

Experimental results for numerical data

Figure 1 demonstrates Leonard-Kramer problem results: dependencies of classification errors percent *%err*, elementary cell size *cell* (the smaller is the *cell,* the larger is resolution), and average fields dimensionality $E\{s\}$ vs the code density p (the fraction of 1s in the code). For *Prager* and *RSC* coding, the results of *SVM* and of the perceptron with single-class ("Perc0") and multi-class learning ("Perc1") with no margin ("T0") and enlarged margin ("T0.75") were averaged by 10 realizations of codes at N=100. Results for *SVM* with kernels (*Kernel*) are also shown. For all cases (as well as for large Ns Figure 2) classification error reaches its minimum near p=0.25, which corresponds to the minimum *cell* and $E\{s\}$ = 2.

Figure 2 demonstrates %*error* and *cell* vs N at p=0.25. The results were averaged by 10 realizations of code generation trials. At N=500 the *SVM* results have already been close to the kernel results. For the enlarged margin perceptron (T=0.75) with multi-class learning the error for N>(300–1000) was lower than the *SVM* one. Training for perceptron was faster than that for *SVM* by 20 times, while testing was >100 times faster.

The experimental results for the *DataGen* data are given in Figure 3 (*A*=4, *S*=3, *C*=4, *R*=4, where *R* determines the complexity of the class regions [6]) and the number of samples per class is equal to 100. Averaging was conducted through 5 realizations of the *DataGen* samples and 5 realizations of codes. For these parameters the minimum cell value corresponds to *p*~0.3 (and close to it for *p*=0.125...0.5) and the error minimum for both *SVM* and the enlarged margin perceptron is also reached in this interval. For *N*=100 it is biased to the larger *p* values (which ensures a more stable number of 1s). For *N*=1000 the minimum is biased to the smaller *p* which corresponds to a larger mean dimensionality of receptive fields, while the number of 1s remains large enough and the *cell* is small enough. The training time for the perceptron is ~20 times less than for *SVM*, and the testing time is ~500 times less.

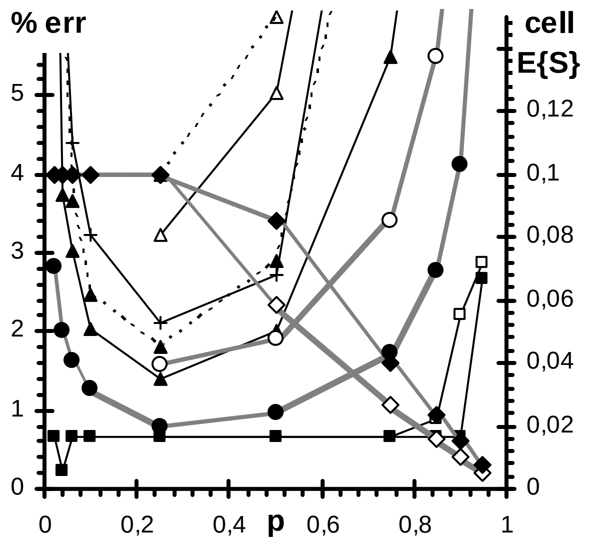

Figure 1

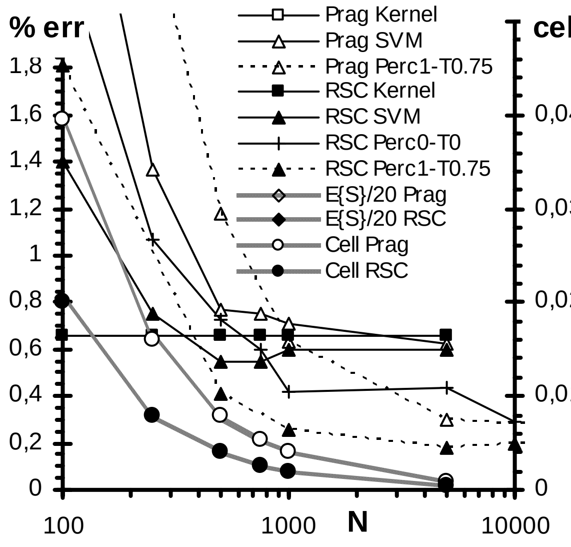

Figure 2

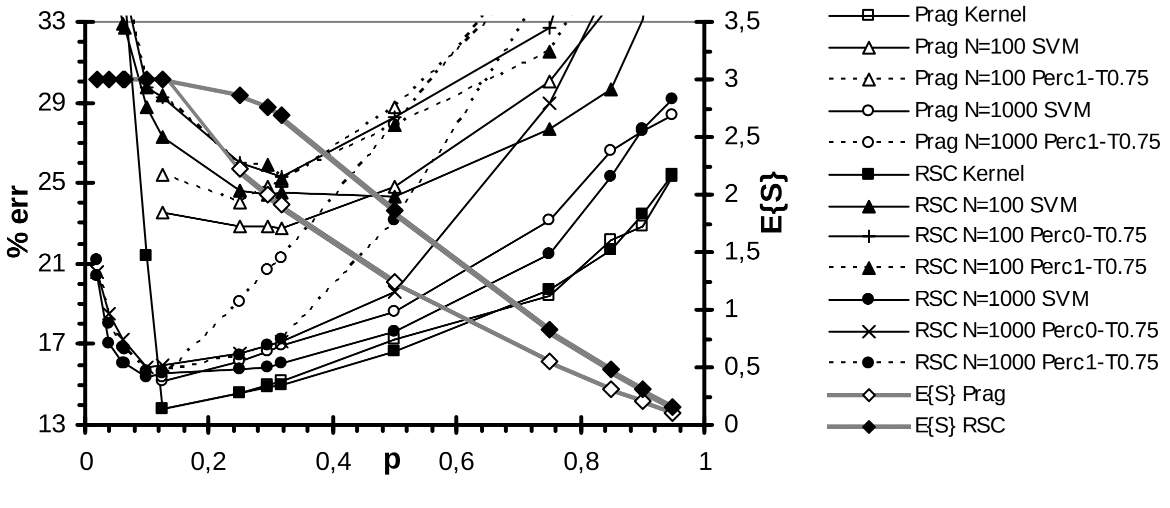

Figure 3

We have also obtained and compared experimental results for the multi-class and single class learning perceptrons. The error rate for multi-class learning perceptron was up to 1.5 times lower than for the usual one, whereas the error rate for multi-class perceptron with the enlarged margin was still lower and comparable with the error for single-class perceptron with the enlarged margin. Typically, the learning curves (test error vs training iteration number) were lower for multi-class learning than for single-class learning, and best results for multi-class learning were higher than those for single-class learning. The results for ordinary perceptron (no margin and single-class learning) were typically lower than those for perceptrons with the enlarged margin in all tests.

Table 1

Database	RSC SVM	RSC kern.	RSC Perc0 T0	RSC Perc1 T0.75	Prager SVM	Prager kern.	kNN	MLP	IRVQ
Clouds	12.68	14.84	17.84	–	12.4	14.8	11.8	12.2	11.7
Concentric	1.36	1.2	1.58	–	1.17	1.04	1.7	2.8	1.5
Gaussian2 S=2	28.12	35.12	38.42	–	27.83	35.64	27.4	26.8	27.2
Gaussian7 S=2	14.35	15.68	20.49	–	14.36	15.76	15.9	15.3	11.5
Gaussian7 S=5	14.69	14.64	19.62	–	13.36	15.12	–	–	–
Iris S=2	6.53	6.67	6.13	5.33	5.59	6.67	4	4.3	6.7
Iris S=4	4.27	6.67	7.47	5.73	6.13	6.67	–	–	–
Phoneme S=2	14.12	11.51	16.43	13.7	15.79	14.47	12.3	16.3	16.4
Phoneme S=5	13.61	11.62	15.74	13.19	14.82	12.62	–	–	–
Satimage S=2	10.06	10.13	10.69	9.15	10.82	10.79	9.9	12.3	11.4
Satimage S=5	10.11	–	10.89	9.1	10.64	–	–	–	–
Texture S=2	0.82	0.76	1.44	1.13	0.82	0.80	1.9	2.0	3.1
Texture S=5	0.73	–	1.65	1.07	0.74	–	–	–	–

For the artificial data of the *Elena* database the code parameters were *N*=1000, *A*=*S*=2, *p*=0.25; for the real data (*Iris*, *Phoneme*, *Satimage*, *Texture*) *N*=10000, *S*=2,5(4), *p*=0.1 and 0.25. Table 1 demonstrates percentage of classification errors. For *SVM* and perceptron the results were obtained by averaging over 10 realizations of *RSC* and *Prager* codes. The best results of the known methods *kNN*, *MLP*, *IRVQ* are also given [7]. The comparison of results shows that *RSC* and *Prager* coding provided the best result to *Concentric*, *Phoneme*, *Texture* and the second best result for *Satimage* and *Gaussian 7D*. Perceptron training time is (on the average) several times less, and test time is dozens of times less than that for SVM.

Classification of texts and images

Traditional approaches to text classification use functions of word occurrence frequencies as elements of their vector representations. Methods for informative feature selection can be used to reduce vectors' dimensionality, [4]; however, even simplified methods that consider features as independent have quadratic computational complexity. We propose and investigate the use of distributed representations for dimensionality reduction of vector text representation. *N*-dimensional binary code with *m* 1s in random positions is used to represent each word. *N*-dimensional text representation is formed by adding of its word vectors, with the following mapping to binary space performed by a threshold operation, or by context-dependent thinning *CDT* (see [3]).

Testing in the classification task has been conducted using *Reuters*-21578 text collection [3] by means of *SVM*. For the *TOP*-10 categories *BEP* (break even point of recall/precision characteristic) for the initial vector representation of *N**=20000 was 0.920/0.863 (micro/macro averaging). Using of the distributed representations with *N*=1000, *m*=2 made it possible to obtain 0.861/0.775 (micro/macro averaging), and usage of *CDT* in some experiments increased it by several percents.

The analogously formed distributed representations were studied for classification of handwritten digit images of the *MNIST* database [4], where images were coded by the extracting binary features. The presence of each feature corresponded to the combination of white and black points in some positions of retina (*LIRA* features [4]). As a result, a "primary" binary code was obtained. Then it was transformed to the "secondary" representation using the same procedures as for text information.

Classification results with dimensionality reduction from *N** to *N* are shown in Table 2. Line "sel" contains the error percent obtained using selection of informative features [4]. Line "distr" contains classification results for the "secondary" binary distributed representations. Results for the distributed representations considerably exceed the results of initial representations for the same *N* and are similar to the results of feature selection methods [4].

We have also obtained and compared *MNIST* experimental results for multi-class and single class learning perceptrons. Here we used original *LIRA* features without transformation to secondary distributed representations, for *N*={1000, 10000, 50000}, and both with and without feature selection. We observed the same tendencies as for numerical data, however the advantage of multi-class learning was more pronounced for weaker classifiers (at *N*=1000) than for the better ones (at *N*=50000).

Table 2

N (err)	5000(667)	10000 (407)		50000 (195)			128000 (160)		
*N**	1000	1000	5000	1000	5000	10000	1000	5000	10000
sel	820	578	420	492	264	242	474	261	218
distr	904	727	415	632	274	213	826	264	204

Conclusions

The developed binary distributed representations of vector data (numeric, text, images) were investigated in the classification tasks. A comparative analysis of various method results for the tasks with artificial and real data was carried out. The study showed that analytical expressions for the characteristics of the *RSC-Prager* codes of the numerical vectors obtained in [2] make it possible to select code parameters that provide high results in the non-linear classification tasks using linear classifiers. Results obtained with the proposed perceptron with an enlarged margin are comparable to the results of the state-of-the-art *SVM* classifiers, however a significant decrease in training and recognition time has been observed. The results obtained with the *RSC-Prager* kernels also make it possible to reduce training and testing time for small *S*.

Application of distributed encoding for representation of binary features in texts and images also made it possible to obtain computationally effective solutions of classification tasks preserving classification quality. A promising direction of further studies could consist in developing computationally efficient *RSC* and *Prager* kernels, as well as developing distributed representations and kernels that provide a more adequate account for structural information in the input data.

Acknowledgements

Author expresses his gratitude to Ivan Misuno and Serge Slipchenko for the support and collaboration during preparation of this paper.

Bibliography

[1] R. Duda, P. Hart, D. Stork. Pattern Classification, 2nd ed. – New York: John Wiley & Sons, 2000.

[2] S.V. Slipchenko, I.S. Misuno, D.A. Rachkovskij. Properties of coarse coding with random hyperrectangle receptive fields. Mathematical machines and systems, N 4, pp. 15-29, 2005 (in Russian).

[3] I.S. Misuno. Distributed vector representation and classification of texts. USIM, N 1, pp. 85-91, 2006 (in Russian).

[4] I.S. Misuno, D.A. Rachkovskij, S.V. Slipchenko. The experimental research of handwritten digits classification. System technologies, N 4 (39), pp. 110–133, 2005 (in Russian).

[5] V.N. Vapnik. Statistical Learning Theory. – New York: John Wiley & Sons, 1998.

[6] I.S. Misuno, D.A. Rachkovskij, E.G. Revunova, S.V. Slipchenko, A.M. Sokolov, A.E. Teteryuk. Modular software neurocomputer SNC - implementation and applications. USiM, N 2, pp. 74–85, 2005 (in Russian).

[7] D. Zhora. Evaluating Performance of Random Subspace Classifier on ELENA Classification Database. Artificial Neural Networks: Biological Inspirations – ICANN 2005 – Springer–Verlag Berlin Heidelberg, pp. 343–349, 2005.

Authors' Information

Dmitri A. Rachkovskij – International Research and Training Center of Information Technologies and Systems; Pr. Acad. Glushkova, 40, Kiev, 03680, Ukraine; e-mail: dar@infrm.kiev.ua.