Concept Learning and Classification with Prime Implicants Applied to Numerical Domains

Z. Shevked, L. Dakovski

Key Words: *Concept learning; learning from examples; complementing; numerical data.*

Abstract. *In this paper we discuss an algorithm for logical function minimization and its application to the problem of concept learning from examples. The algorithm is based on complementing of the available negative examples. The goal is to find a more compact representation of classification function and use it for further prediction of unknown cases. This is accomplished by an innovatory strategy for logical function minimization. The method can be applied in every domain where observations might be described by attributes with nominal values (valued in a finite set). Here we extend this approach to handle numerical data (valued in a linear interval) as well.*

1. Introduction

Concept learning from examples aims at building a model that represents the examples in a more compact way as the purpose is to effectively determine whether new unseen cases belong to the learned concept or not. This problem is very similar to the problem of logical function minimization, where again the output for some of the input combinations is known and a shorter representation consistent with them is looked for.

Logical minimization is one of the oldest problems in computer science. It is proved to be NP-complete, but its numerous and varied applications stimulate the research for effective solution till nowadays. The exact minimization algorithms consist of two main phases: generation of function prime implicants and choosing the irredundant ones (more popular as covering problem). Exact minimization algorithms are computationally expensive either in time and space, because the number of implicants increases exponentially with the number of inputs. The most successive representatives of this group as Scherzo [3] and Rondo [14] use the so called Binary Decision Diagrams to facilitate the simultaneous operating with large number of objects.

Another direction in logical function minimization tries to handle the intractability drawback of the exact algorithms for large problems by using the heuristics. Such algorithms, rather than first generating all prime implicants and after that engaging in the covering problem, directly take up the covering problem and then only the necessary prime implicants are generated. Combining these two steps results in powerful heuristic based practical tools as ESPRESSO [7] and its modifications [13,16], BOOM [8], FC-Min [6] which almost always produce near-minimum solution.

In this paper we discuss an algorithm for logical function minimization which combines the advantages of both upper types, namely the exact solution and the optimization by coverage directed search. This is achieved by finding only these prime implicants of the compliment of the function OFF-set that cover its ON-set. Here this algorithm is adapted for multi-valued logic and applied for concept learning task.

The concept learning from examples aims at finding a classifier that describes the examples in a more compact way as the purpose is to effectively classify new previously unknown cases. The most learning algorithms use a representation of the available observations by a set of characteristics that best describes an instance from the domain of interest. For every particular example each of these attributes (characteristics) takes a concrete value. In most real-world problems an attribute might be nominal - when it is valued in a finite set, or linear - when it is valued in an interval. Some of the learning algorithms handle only nominal attributes, some handle only numerical attributes and others handle both of them. As is to be expected, it is always better to support both of the possible attribute types especially for learning on datasets including both types - these datasets are called mixed or hybrid.

We have developed an algorithm for learning from examples based on logical function minimization. It results in a set of prime implicants that represent a more general concept describing the learned class. The highlight of this minimization algorithm inspired by two-valued logic is the extensive use of the complement of a value. Finding the complement of a value in two-valued logic is simple - when the value is 1, its complement is 0 and vice versa. This principal easily can be applied to nominal data when an attribute takes one of predefined set of values. The complement of a particular value from such a set is its complement - a subset consisting of all remaining possible values for this attribute. Our goal here is to extend this algorithm to handle not only nominal type of attributes but also numerical attributes (numerical data) as the main question is interpreting the concept of a numerical value complement .

The paper is organized as follows: Section 2 discusses handling nominal attributes in the algorithm for learning from examples based on function minimization; section 3 represents a way to apply this algorithm for numerical data; section 4 mentions some experimental results from applying the new extended approach. The conclusion follows and future tasks are discussed at the end.

2. Learning from Examples by Minimization

In this paper the goal is to extend the algorithm for learning from examples by minimization introduced in [4]. Shortly it works as follows: all training examples are separated in two groups of the positive examples - $E^+ = \{E_1^+, E_2^+, \ldots E_p^+\}$ and negative examples - $E^- = \{E_1^-, E_2^-, \ldots E_n^-\}$; these groups are used to build two logical functions in sum-of-minterms form - one that corresponds to positive examples ($\mathbf{F_1}$) and other that corresponds to negative examples ($\mathbf{F_0}$):

$$(1)\quad F_1 = E_1^+ \cup E_2^+ \cup \ldots \cup E_p^+ = \bigcup_{i=1}^{p} E_i^+ ;$$

$$(2)\quad F_0 = E_1^- \cup E_2^- \cup \ldots \cup E_n^- = \bigcup_{i=1}^{n} E_i^- ;$$

Each example is described by a set of attributes $A = \{A_1, A_2, \ldots, A_k\}$ and an attribute A_i $(i \in (1,k))$ takes one of a predefined set of m_i values $V_i = \{v_i^1, v_i^2, \ldots, v_i^{m_i}\}$. This way an example E_j is represented by such an expression:

$$(3)\quad E_j = v_{j,1} \cap v_{j,2} \cap \ldots \cap v_{j,k} = \bigcap_{i=1}^{k} v_{j,i},$$

where $v_{j,h}$ is the value which takes the attribute h for the example j.

In a learning process the purpose is to find a good approximation of the target classification function that generalizes available training examples in order to successfully predict unknown cases. The two functions that we have are approximations of the target function, but they correspond only to known training instances - they do not generalize any. Also, each of them interprets all unknown cases respectively as belonging to class (F_1) or not (F_0). Because of these restrictions they would not give reliable results if used for classification of unknown examples. But they can be used to build a better classifier combining the advantages of both.

As we know in concept learning the aim is to find shorter concept describing the class. In our case - working with logical functions - the most suitable representation would be prime implicants, because they have equal meaning with the function and at the same time are fairly shorter. That is why we work with prime implicants.

Let us consider the complement of F_0:

$$(4)\quad \bar{F}_0 = \overline{\left(E_1^- \cup E_2^- \cup \ldots \cup E_n^-\right)} = \bar{E}_1^- \cap \bar{E}_2^- \cap \ldots \cap \bar{E}_n^- ;$$

$$(5)\quad \bar{F}_0 = \overline{\left(v_{1,1} \cap v_{1,2} \cap \ldots \cap v_{1,k}\right)} \cap \overline{\left(v_{2,1} \cap v_{2,2} \cap \ldots \cap v_{2,k}\right)} \cap \ldots \cap \overline{\left(v_{n,1} \cap v_{n,2} \cap \ldots \cap v_{n,k}\right)};$$

$$(6)\quad \bar{F}_0 = \left(\bar{v}_{1,1} \cup \bar{v}_{1,2} \cup \ldots \cup \bar{v}_{1,k}\right) \cap \ldots \cap \left(\bar{v}_{n,1} \cup \bar{v}_{n,2} \cup \ldots \cup \bar{v}_{n,k}\right);$$

$$(7)\quad \bar{F}_0 = \bigcap_{i=1}^{n} \bigcup_{j=1}^{k} \bar{v}_{i,j} ;$$

In the last expression $\bar{v}_{i,j}$ represents a set of all possible values for the attribute j excluding the value $v_{i,j}$ - the value for the j-th attribute in the i-th positive example E_i^-. Our goal is to optimize (7). Here the complication comes from three points:

- n - number of negative examples;
- k - number of attributes;
- $|\bar{v}_{i,j}|$ — power of sets $\bar{v}_{i,j}$.

Since the first two parameters if diminished, would affect in general the classifier correspondence to the training data, only the third parameter remains as a possible way for optimization.

According to the definition of $\bar{F}_0$, it corresponds to all positive examples but no one of negatives ones. Also $\bar{F}_0$ attaches all unknown cases positive classification, because it includes all but the negatives. Still this is not much, but if the $\bar{F}_0$ prime implicants are known we can choose from them for classification only those that correspond to positive examples. Even more, we can control how restrictive the classifier need be by increasing the number of positive examples that are required to be corresponded by a prime implicant in order to be used for classification. This way the classifier is build only by those prime implicants that correspond at least to the specified number of positive examples.

Having in mind these conditions the minimization of $\bar{F}_0$ aimed at getting its prime implicants might be optimized. $\bar{F}_0$ is processed as the goal is to reduce its set of prime implicants only to the ones that correspond to a positive example. For this reason, $\bar{F}_0$ is coupled separately with each positive example to get prime implicants corresponding exactly to this example. An example is corresponded by a concept only when all of the values in a concept are the same as in the example or are more general. Thus a prime implicant is required to have the same or more general value than the currently considered positive. It can not have a value for an attribute that differs from the one in the positive example or is not more general. This means that

when $\overline{F}_0$, is processed toward a positive example i all attribute values in the representing expression $\overline{F}_{0,i}$ different than corresponding values in the positive example are not necessary, they can be removed from the expression. Thereby each set $\overline{v}_{i,j}$ is replaced by a single value. This leads to a significant reduction and a great optimization is achieved because this way the most complicated computations of a function minimization are simplified. This is especially useful when the number of examples or the number of attributes is bigger and the standard minimization would involve far more memory and computational power. The presented approach decreases the amount of needed computational resources.

In general the expression of $\overline{F}_0$ consists of complements of values for attributes (6). When the attribute type is nominal there is a set of predefined values v_i that this attribute i can take. According to the set theory the complement of a value from a set is its complement or a subset containing all remaining values. In the presented approach, when forming $\overline{F}_{0,i}$ only these values, that do not exclude the value in the positive example, can remain in the expression. If a negative example n and a positive example p have the same value for a given attribute this means that both examples can not be differentiated by this attribute. The corresponding $\overline{F}_{0,n}$ expression part for the considered attribute will consist of a set of all other possible values and when compared with the value in the positive example for the current attribute this set will not contain the same value as the positive example. Therefore, for this couple of examples it is not useful and must not remain in the expression because will lead to prime implicants that correspond to both positive and negative sample. However this is not the goal - it is to get implicants that correspond to positives and do not to negatives, so would be used to separate the examples. For this reason when the expression in $\overline{F}_0$ does not contain the value from positive example it is removed. In the other case when the values in the positive and negative example differ, then in $\overline{F}_0$ the complement will be a set containing the value from the positive example. Here in $\overline{F}_{0,m}$ the value remains, because the two examples can be separated by this attribute value so the got implicants will correspond to this positive instance and will not to the negative ones.

Let us consider a part of $\overline{F}_0$ that represents a complement of the negative example i:

$$(7)\quad \overline{E}_i = \left(\overline{v}_{i,1} \cup \overline{v}_{i,2} \cup \ldots \cup \overline{v}_{i,k}\right)$$

and the positive example m:

$$(8)\quad E_m = v_{m,1} \cap v_{m,2} \cap \ldots \cap v_{m,k}$$

When comparing the values $\overline{v}_{i,h}$ and $v_{m,h}$ for the attribute h there are two cases:

- If $v_{i,h}$ differs from $v_{m,h}$ then the examples i and m might be differentiated by the attribute h, so $v_{m,h}$ remains in the expression for $\overline{F}_{0,m}$, because in the currently targeted prime implicants the attribute h should have this value or more general in order to correspond to the positive example m.
- If $v_{i,h}$ is the same as $v_{m,h}$ then the examples i and m can not be differentiated by the attribute h, so this is not needed in the expression for $\overline{F}_{0,m}$.

The expression for $\overline{F}_{0,m}$, which represents the part of the negative example complement that corresponds to the positive example E_m^+, is as follows:

$$(9)\quad \overline{F}_{0,m} = \bigcap_{i=1}^{n} \bigcup_{j=1}^{k} v_{i,j}^{m},$$

where $v_{i,j}^{m}$ is a single value from the set $\overline{v}_{m,j}$ or the empty set $\varnothing$.

$$(10)\quad v_{i,j}^{m} = \begin{cases} v_{i,j}, & (v_{i,j} \neq v_{m,j}) \\ \varnothing, & (v_{i,j} = v_{m,j}) \end{cases}$$

Here $v_{i,j}$ is the value for the j-th attribute in the i-th positive example and it is compared to the value $v_{m,j}$ for the same j-th attribute in the m-th negative example. In the first case (when the two values differ), the particular value $v_{i,j}$ must remain in the expression, because it is essential for distinguishing instances belonging to the class from not belonging ones. In the second case (when the two values are the same), the particular value $v_{i,j}$ can not remain in the expression, because comparing the two currently considered examples by this attribute has no use for determining the class, so it is omitted.

3.Handling Numerical Data

In case of numerical data the matter is how to form $\overline{F}_{0,m}$ for numerical attribute types. When considering the positive example m and an expression for a negative example from $\overline{F}_0$ for a particular attribute the goal is to find an expression that will be written for this pair in $\overline{F}_{0,m}$ and the only requirement to this

expression is to help differentiate this two examples. This is needed, because by definition $\overline{F}_{0,m}$ corresponds to the positive example m and does not correspond to all negatives. So it must consist of expressions that distinguish m from negatives. In general, a complement of a value is its complement. For a numerical attribute h which takes a value v_h from the interval (b_h^l, b_h^r) either finite or infinite, the complement is the whole interval except this particular value- $(b_h^l, v_h) \cup (v_h, b_h^r)$. Here b_h^l denotes the left boundary for the possible values for the h-th attribute and b_h^r denotes the right boundary for the possible values for the h-th attribute. But when forming $\overline{F}_{0,m}$ not all of this is required, we need only these parts of the interval that it is enough to distinguish the positive example m (8) from currently considered negative example i (7). The maximal interval that surely contains the value from the positive example m and excludes the value from the negative example is the interval from the second value to the respective bound. There are two cases:

- If $v_{i,h} > v_{m,h}$ then the maximally general interval that contains the example m and does not contain the example i and differentiates them by the attribute h, is the interval $(b_h^l, v_{m,h})$, so it is used in the expression for $\overline{F}_{0,m}$.

- If $v_{i,h} < v_{m,h}$ then the maximally general interval that contains the example m and does not contain the example i and differentiates them by the attribute h, is the interval $(v_{m,h}, b_h^r)$, so it is used in the expression for $\overline{F}_{0,m}$.

Exactly this is used when forming $\overline{F}_{0,m}$ - the condition that becomes a part of it is the value for this attribute need be in the interval between the value from the negative example and this bound that includes the value from the positive example. This interval will be a part of $\overline{F}_{0,m}$ and together with other restrictions to this particular attribute coming from other negative examples forms restrictions to this attribute composing wanted prime implicants. Usually, after applying this rule to all couples of the positive example m and the negatives the interval for the considered numerical attribute shrinks round the value presented in the positive example.

As in the case of nominal data when the value in the positive example and the value in the negative example are the same this attribute can not be used to distinguish the two examples so it is not presented in the respective expression in $\overline{F}_{0,m}$.

In the case of numerical data the expression for $\overline{F}_{0,m}$, which represents the part of the negative example complement that corresponds to the positive example E_m^+, is as follows:

$$(9) \qquad \overline{F}_{0,m} = \bigcap_{i=1}^{n} \bigcup_{j=1}^{k} v_{i,j}^m$$

where $v_{i,j}^m$ is an interval from the value $\bar{v}_{m,j}$ to the respective left b_h^l or right b_h^r boundary, or it is the empty interval $\varnothing$.

$$(10) \qquad v_{i,j}^m = \begin{cases} (b_h^l, v_h), (v_{m,j} < v_{i,j}) \\ (v_h, b_h^r), (v_{m,j} > v_{i,j}) \\ \varnothing, \quad (v_{i,j} = v_{m,j}) \end{cases}$$

It is clear that the interval to distinguish a pair of positive and negative example by a particular attribute can not be this part lying between the value from the negative and the bound and not containing the value from the positive example, because if so it is not possible to corresponds to the positive example in any way. We need such a part that generalizes the positive example without correspondings to the negative. The idea is to get exactly that sort of prime implicants that corresponds to this positive example and excludes the negative one. That is why only this part of the interval remains in the expression for $\overline{F}_{0,m}$ - it is simply because there is no way the other part to meet this requirement; it does not correspond to the positive example.

In the literature this approach is called maximally discriminant selector [12]. It is the maximal range for a linear attribute that includes the value from the positive example and excludes the value from the a negative example. This selector covers the positive example and discriminates the negative one. It is used in the learning process as the goal is to get such a generalization of the positive example that excludes negative examples. This procedure is applied to all pairs attributes for all couples of a positive and negative example. Because this selector has a value true or false depending whether the checked value is in the interval or not, this approach can be used not only for numerical datasets but for datasets with mixed type of attributes too. Processing numerical data does not change the type of the output. Again, the resulting classification function is a Boolean function with two possible outputs - true, which means „belongs to the class" and false, which means „does not belong to the class". Also as one can guess there is no matter whether the type of values is integer or real. The type of value (integer or real) involves only the type used in the program implementation of the algorithm; it is not related to the algorithm itself.

Thus the algorithm for learning from examples based in logical function minimization can be used successfully not only for datasets with nominal attribute types but also for numerical data and for mixed datasets.

4.Experimental Results

The presented algorithm for learning from examples is used for experiments on the Haberman dataset from the UCI machine learning repository [17]. The dataset contains cases from a study on the survival of patients who has undergone surgery for breast cancer at the University of Chicago's Billing Hospital for the period between 1958 and 1970. The number of examples in the dataset is 306 (255 positive and 51 negative examples). They are characterized by 3 numerical attributes and one class attributes:

- Age of patient at time of operation.

- Patient's year of operation.
- Number of positive auxiliary nodes detected.
- Survival status (class attribute) -

1 = the patient survived 5 years or more
2 = the patient died within 5 years.

In order to perform the experiments data is randomly separated in three parts as this separation is balanced i.e. the proportion positive examples count / negative examples count is kept in the got parts. One half of the data (154 examples) is used for training. The described process of minimization is performed on it and the result is a set of prime implicants. After this a process of validation is done on one fourth of data (76 examples) - the optimal value for the tuning parameter is determined at this stage. Tuning parameter is the required number of prime implicants that must correspond to a tested example in order to classify it positively. This parameter is studied in details in [5]. It avoids overfitting and can decrease the eventual effects of noisy data. After determining its optimal values the tuned classifier can be applied on the remaining one fourth of the data (76 examples) for testing. Classification accuracy is determined on this part as the percent of correctly predicted test instances from the test set.

Experimental results for Haberman dataset

No	Time for minimization (s)	Total count of PI	Optimal values for parameter	Classification accuracy (%)
1	7.03	449	1÷4	72.4
2	7.30	514	1÷4	75.0
3	5.89	432	1÷5	72.4
4	7.13	425	1÷6	73.7
5	9.58	573	1÷5	76.3
6	6.67	449	1÷4	72.4
7	6.35	488	1÷4	71.1
8	6.91	461	1÷3	72.4
9	6.69	455	1÷7	76.3
10	8.64	521	1÷2	71.1

This procedure is performed ten times with different data separations. The results from the folds are presented in the *table*. There are shown also the time for minimization on a PC with processor 846 MHz and 256 MB RAM; the total number of prime implicants found after minimization for each experiment; the optimal value for tuning parameter which is determined by classifying the validation examples with got prime implicants using different values for this parameter; the achieved classification accuracy of the resulting classifier consisted of the found prime implicants and the tuned parameter on the test set.

This dataset is often used for testing different machine learning algorithms, there are many of papers where experimental results on it are reported [1,2,9,10,11,15]. It appears that this is one of the relatively hard to predict datasets - the achieved results on average vary between 65% and 70%; the maximal classification accuracy that we found in the literature is 76.2% [15]. With our algorithm the maximal achieved classification accuracy is 76.3% and the average classification accuracy for all experimentations is 73.3%.

Our experiments with this data showed that usually the optimal values for tuning parameter are between 1 and 4 which means that for this dataset it can be said that there is no overfitting problem for this learning algorithm. Even using the resulting classifier in its clear mode (without tuning parameter, which is equal to a value 1 for it) would give good prediction accuracy.

5.Conclusions and Future Work

This paper continues investigating of a learning algorithm based on logical function minimization which results in a set of prime implicants that represent an approximation of the target concept. The algorithm can be applied to datasets containing nominal data. Here it is extended to handle numerical data too. The procedure for manipulating numerical attributes is described and grounded. In this way the algorithm might be used also for learning on numerical and mixed datasets.

The presented approach is applied to an experimental dataset. It has shown good time performance and high classification accuracy.

Future work includes examining of the algorithm performance for different datasets. Also a challenge to each learning algorithm is applying to large datasets consisting of thousands of records. Another direction of our study is handling missing values, because in the most real world problems not all values are known. Also we consider ways for incremental implementation of our algorithm which currently belongs to the group of batch learning algorithms.

References

1. Ataman, K., W. N. Street, Y. Zhang Learning to Rank by Maximizing AUC with Linear Programming. *IEEE International Joint Conference on Neural Networks*, 2006,123-129.
2. Bhattacharya, B., K. Mukherjee, G. Toussaint Geometric Decision Rules for Instance-based Learning Problems. *The First International Conference on Pattern Recognition and Machine Intelligence*, Kolkata, India, 2005.
3. Coudert, O.Two-level Logic Minimization: an Overview. - *Integration, the VLSI Journal,17:97-140*, 1995.
4. Dakovski, L., Z. Shevked Alternative Approach for Learning from Examples, *Proceedings of CompSysTech Conference*, Varna, Bulgaria, IIIB.5-1 - IIIB.5-6, 2005.
5. Dakovski, L., Z. Shevked Tuning Classification for Prime Implicant Based Learner, *Proceedings of CompSysTech Conference*, Veliko Tarnovo, Bulgaria, 2006.
6. Fiser P., J. Hlavicka. FC-Min: A Fast Multi-Output Boolean Minimizer. In: *Proceedings of Euromicro Symposium on Digital Systems Design*. Antalya, TR, 2003.
7. Hachtel G. D., F. Somenzi. *Logic Synthesis and Verification Algorithms, 564 pp. Kluwer Academic Publishers Boston, MA*, 1996.
8.Hlavicka J., P. Fiser. BOOM: A Heuristic Boolean Minimizer. In: *Proceedings of ICCAD-2001, San Jose, Cal., USA*, 2001, 439-442.
9.Kar, P., S. Sen (2003) Agent Teaching Agent Framework. *AAMAS'03*, Melbourne, Australia.
10. Kotsiantis, S., P. Pintelas An Online Ensemble of Classifiers. *The Fourth International Workshop on Pattern Recognition in Information Systems, In conjunction with 6th International Conference on Enterprise Information Systems*, Porto, Portugal,2004, 59-68.
11. Laurikkala, J. Improving Identification of Difficult Small Classes by Balancing Class Distribution. University of Tampere, Finland, 2001.
12.Michalski, R.S. A theory and Methodology if Inductive Learning.

Machine Learning: an artificial intelligence approach, volume 1. Morgan Kaufmann, 1983.
13.McGeer P. C., J. V. Sanghavi, R. K. Brayton, A. L. Sanciovanni-Vincentelli. ESPRESSO-SIGNATURE: A new Exact Minimizer for Logic Functions. In: *Proceedings of DAC'93*, 1993.
14. Mishchenko, A. http://web.cecs.pdx.edu/~alanmi/research/min/minSop.htm, 2001.
15. Perez, A., P. Larranaga, I. Inza Supervised Classification with Conditional Gaussian Networks: Increasing the structure complexity from naïve Bayes. University of The Basque Country, 2006.
16. Sapra S., M. Theobald, E. Clarke. SAT-Based Algorithms for Logic Minimization. In: *Proceedings of 21st ICCD-2003*, 2003, 510-518.
17.University of California, Irvine Machine Learning Repository - Haberman dataset, ftp://ftp.ics.uci.edu/pub/machine-learning-databases/haberman

Manuscript received on 08.03.2007

Zekie Shevked *- PhD student in Artificial Intelligence Systemes at Technical University of Sofia. She received MSD in Computer Systems and Technologies from Technical University of Sofia, branch - Plovdiv in 2005. Her works are in the fields of Machine Learning.*

Contacts:
e-mail: zekie_shevked@yahoo.com

Eng. Lyudmil Dakovski *-Prof. D.Sc. CLBME-BAN. His research interests include computer architecture, neural nets, artificial intelligence, affective computing.*

Contacts:
Centre of Biomedical Engineering
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., bl. 105
1113 Sofia, Bulgaria
e-mail: lgd@clbme.bas.bg

continuation from 30

12. Vidyasagar, M. Nonlinear Systems Analysis. Second Edition. Englewood Cliffs, NJ: Prentice Hall, 1993.
13. Luckjiff, G., I. Wallace, D. Divan. Feedback Linearization of Current Regulated Induction Motors. Power Electronics Specialists Conference, 2001. PESC. 2001 IEEE 32nd Annual, 2001, 1173–1178.
14.Benchaib, A., A. Rachid, E. Audrezet. Sliding Mode Input–Output Linearization and Field Orientation for Real-Time Control of Induction Motors. – *Power Electronics, IEEE Transactions on*, 14, January 1999,Issue 1, 3–13.
15. Sobczuk, D., M. Malinowski. Feedback Linearization Control of Inverter Fed Induction Motor – with Sliding Mode Speed and Flux Observers IEEE Industrial Electronics. IECON 2006 – 32nd Annual Conference on, November 6–10 2006 10.1109/IECON.2006.348089, 1299-1304, Paris, France.
16. Mitronikas, E. D., A. N. Safacas, E. C. Tatakis. A New Stator Resistance Tuning Method for Stator-Flux-Oriented Vector-Controlled Induction Motor Drive. – *IEEE Transactions on Industrial Electronics*, 48, December 2001, No. 6, 1148-1157.
17. Holtz, J. Sensorless Speed Control of Induction Motor Drives – A Tutorial. ISIE 2006, 9-13 July, Montréal, Canada.
18. Rehman, H. Elimination of the Stator Resistance Sensitivity and Voltage Sensor Requirement Problems for DFO Control of an Induction Machine. IEEE Trans. on Industrial Electronics, 52, February 2005, No 1, 263-269.

Manuscript received on 14.02.2008

Stanislav Enev *(born 1980) received the M.Sc. degree in electrical engineering from Technical University – Sofia in 2004. Currently he is working towards a Ph.D. degree at the French Language Department of Electrical Engineering, Technical University – Sofia. Since March, 2008 he is with the Department of Industrial Automation (FA), TU-Sofia as an assistant professor. His current research interests are in nonlinear control theory and applications.*

Contacts:
Technical University – Sofia, Faculty of Automation,
Department of Industrial Automation
e-mail: stanislav.enev@gmail.com