LARGE-SCALE FEATURE SELECTION USING MARKOV BLANKET INDUCTION FOR THE PREDICTION OF PROTEIN-DRUG BINDING

C.F. ALIFERIS, I. TSAMARDINOS, A. STATNIKOV

Department of Biomedical Inforrmatics, Vanderbilt University, 412 EBL, Informatics Center, 2209 Garland Ave, Nashville, TN 37232, USA
E-mail: {constantin.aliferis, ioannis.tsamardinos, alexander.statnikov}@vanderbilt.edu

Selecting appropriate features for classification is a pressing problem due to the emergence of extremely large bio-databases. In this paper we empirically evaluate a recently introduced Markov blanket induction algorithm (iterative association Markov blanket - IAMB) for the purpose of large-scale feature selection in the task of finding the optimal subset among 139,351 molecular structural properties that predict binding to thrombin (and thus the potential of these substances as anti-clotting agents). We also develop and evaluate parallel and chunked variants of IAMB for either high-performance parallel computing or for situations where the data exceeds the available RAM. As baseline feature selection comparison methods we use the state-of-the-art support vector machine–based recursive feature elimination (RFE) algorithm, as well as a basic univariate association filtering (UAF) method. The full set of features as well as the feature subsets chosen by each selection method are used to create KNN, linear SVM, polynomial SVM, RBF SVM, Simple Bayes, and Neural Network classifiers. The area under the ROC curve (AUC) is used as the classification performance metric. The results of these preliminary experiments are very encouraging: the chunked version of IAMB achieves superior classification performance to RFE for 4 out of 6 classifiers. The best overall classification model is achieved by combining IAMB and Simple Bayes; this model outperforms linear and non-linear SVM models in this task. In addition, IAMB's selected feature set is ~270 times smaller that RFE. We empirically demonstrate that the parallel versions run in minutes on a small (14 node) parallel computer cluster. The chunked version runs on a few hours on a single desktop CPU in the massive thrombin task.

1 Introduction

Machine Learning methods have found significant applications in many areas of biocomputing and are thus a valuable tool in the computational biologist's arsenal. A primary problem, especially when developing classification models, is how to select the minimal subset of the available features that yields optimal classification performance (we use "feature" and "variable" interchangeably). This is important for increasing classification performance, reducing the cost of measuring predictors, and focusing experimental efforts on a tractable number of instrumental variables that shed light on the structure of the biological domain.

Most research in feature selection is being conducted using *wrapper* algorithms, i.e., heuristic search procedures in the space of all possible feature subsets that use the classifier algorithm that will induce the final classifier as evaluation metric for assessing the quality of each feature subset. Wrapper algorithms have three considerable shortcomings: (a) they require training and evaluation of the performance of the classifier used for every feature subset considered during search, (which is computationally expensive). (b) The heuristic search is not guaranteed to return the optimal variable subset even with abundant samples. (c) It is necessary to repeat the feature selection for every different classifier used to solve the classification problem.

Consequently, what motivates our research is the desire to overcome these disadvantages by creating *filter* feature selections algorithm, i.e., algorithms that select features independently of the classifier algorithm that will be used to induce the final classifier. To do so we associate the problem of identifying the optimal predictor feature subset with discovering the Markov blanket of the target variable, and present algorithms that solve this problem under broad assumptions. The outline of the paper is as follows: section 2 gives necessary theoretical background discussing Bayesian Networks and connecting them to feature selection and causal induction. In section 3 we review a fundamental algorithm for MB induction we have introduced previously. We also introduce parallel and chunked variants of the basic algorithm designed to handle efficiently massive data sets and/or severe memory limitations. In section 4 we evaluate these algorithms in a massive dataset for the purpose of predicting drug-protein binding. Section 5 provides a discussion of related work, limitations and future research directions.

2 Theoretical Background

2.1 Bayesian Networks and Feature Selection

Definition 1. *Feature Selection Problem for Classification.*
Given: (i) a set of features F (including a target variable T) instantiated by some process P, *(ii)* a classification algorithm $A(D_{tr}, T, F_i) \rightarrow a_i$ (where D_{tr} is any set of training instances sampled from P, a_i is the model or class of models returned by $A()$ for some subset F_i of F); and (iii) a classification performance metric M, defined over the space of the outputs of $A()$ and the space of test sets D_{te}.
Find: a minimal subset of F that maximizes performance of $A()$ according to M.

Definition 2. *Bayesian Network* $< V, G, J >$. Let V be a set of discrete variables and J be a joint probability distribution over all possible instantiations of V. Let G be a directed acyclic graph over a set of variables S. Let all nodes of G correspond one-to-one to members of V. We require that for every node A $\in$ V, A is probabilistically independent of all non-descendants of A, given the parents of A (*Markov Condition*). Then we call the triplet [V, G, J] a Bayesian Network (BN), or equivalently a Belief Network or Probabilistic Network [16,17].

Definition 3. *Faithfulness*. The graph *G* of some BN *B* is faithful to a joint probability distribution *J* over feature set *V* if and only if every dependence entailed by *G* is also present in *J*. In the terminology of of Spirtes et al, *G* and *J* are faithful to one another, and in the terminology of Pearl *G* is a perfect-map of *P* and *P* is DAG-isomorph of *G*. We also say that a data-generating process *K* is faithfully represented by *B*, if *K* in the sample limit produces data with joint probability distribution *P*, and *C* is faithful to *P* [8,17,19].

Property 1. *Markov blanket of a feature T, MB(T) of a BN.* The MB(T) in a BN is the set of parents, children, and parents of children of *T* [16]. MB(T) is the minimal set of features conditioned on which all other features are independent of T, i.e. for any feature set S, P(T | MB(T), S) = P(T | MB(T)) [16].

By combining property 1 and definition 3 it follows that knowledge of MB(T) is sufficient for perfectly estimating the distribution of T and thus for classifying T. When calibrated classification is required (and thus the exact conditional probability of T) then MB(T) is also a minimal set of optimal predictors. Hence:

Property 2. In processes that can be faithfully represented by a BN *C*, the Markov blanket of a target feature *T*, MB(T) is the solution to the optimal feature selection problem[*].

Therefore, in order to solve the feature selection problem, under these assumptions, one could induce the BN that generates the data. Methods for inducing BNs from data have been investigated vigorously in the last decade [5,8,10,19]. Unfortunately these approaches are also intractable for large (>100 feature sets). To this end we develop specialized MB-induction algorithms that are more efficient than inducing the *full* BN. The algorithms are provably optimal under the following assumptions:

1. Data is generated by processes that can be faithfully represented by Bayesian Networks.
2. There exist reliable tests of statistical independence. This assumption subsumes the requirement for enough sample size to detect conditional independence with high probability.
3. The observed data is independently sampled (i.e., any pair of instance vectors are sampled with same probability) and identically distributed (i.e., the generating process is time-invariant). We will refer to this assumption as "*iid*" in the remainder of the paper.

[*] When the metric *M* in definition 1 is sensitive to calibration and *A* is any well-calibrated classifier (e.g., Neural Networks optimizing mean squared error loss, KNN, Bayesian Classifiers, etc.).

3 Algorithms for Inducing the Markov Blanket

3.1 Iterative Associative Markov Blanket Algorithm (IAMB)

The algorithm we present next will indeed discover MB(T) if assumptions 1 to 3 hold [20]. Briefly, the algorithm (as outlined in Figure 1) consists of two phases. In the first phase an approximation to MB(T) is built by using a greedy maximum multivariate conditional association heuristic strategy. MB(T) is guaranteed to be contained in this set at the end of phase 1. In the second phase, the initial approximation to MB(T) is pruned down by a backward multivariate conditional independence strategy.

IAMB runs in $O(D\times N\times M)$ worst case time (N is number of features, D the number of instances and M the size of the maximum tentative MB which in the worst case is N) assuming an association metric and test of conditional independence that are linear to D (such as thresholded conditional mutual information [20]). Note that although more sophisticated algorithms are presented in [20] for induction of MB(T), in the present paper we examine the more basic IAMB algorithm as it is more efficient and more amenable to parallelization.

We introduce variants of IAMB that address two problems: (a) the data does not fit into fast memory (RAM), and (b) even if the data fits, processing time is unacceptably long. We allow for the possibility that the user may have access to just one node or, alternatively, have access to several processors arranged in a parallel cluster.

3.2 Fine-Grain Parallel IAMB (fgIAMB)

Initially the algorithm partitions the feature set *F* into *ch* equal splits. Each split *i* corresponds to a chunk C_i of data containing only the variables in the split and the target variable. In phase I of the regular IAMB algorithm the step:

```
  Find the feature F that maximizes:
 abs(association(F ; T | CurrentMB))
```

is collectively executed by the parallel processors: each processor calculates the association of the features in C_i with T and returns the feature with the maximum association; subsequently the global maximum is calculated. Soundness is not compromised with this modification. In terms of asymptotic complexity, the algorithm is $O((D\times N\times M/ch)$ time where *ch* is the number of chunks/processors.

3.3 Chunked Parallel IAMB (cpIAMB)

The data are split exactly as in fgIAMB. IAMB is run on each chunk, the resulting Markov blankets are combined to form the set M1 (Figure 2), and a similar process is repeated once again using M1. Specifically IAMB is run on each chunk returning $MB(T \mid C_i)$ (i.e, MB(T) relative to C_i). In the next step, IAMB is run with the union

```
Input: a  MxN array of data with N number of features (variables) and M
number of instances, a target feature T

Phase I (forward):

CurrentMB = ∅
Continue = True
While Continue= True
   Continue = False
   Find the feature F that maximizes: abs(association(F ; T | CurrentMB))
   If (association(F ; T | CurrentMB)) → 0
       Add F to CurrentMB
       Continue = True
   End If
End While

PhaseII (backward):

For each variable F in V-{T}
   If independent(F; T | CurrentMB – {F})
       Remove F from CurrentMB
   EndIf
EndFor
Return CurrentMB
```

Figure 1. The Iterative Associative Markov Blanket (IAMB) algorithm.

$\cup_i$ (*MB(T | C_i)*) and returns *M1* thus completing the first phase of the algorithm. The first phase guarantees that every parent or child of T will be in M1. However, it may miss a parent *P* of some children *S* of T if *P* and *S* happen to belong in different chunks. If we repeat the same process one more time (phase II) starting from *ch* new chunks each of which is $C_i' = C_i \cup M1$ then all members of MB(T) will be included in the output (set *M2*) therefore cpIAMB is sound. Figure 2 gives the high-level pseudocode. Steps 2 in both phase I and II are the parallel steps.

We note that a potential problem with cpIAMB is that the conditioning set (tentative MB) in some chunk may become larger than the largest conditioning set in IAMB/fgIAMB thus the conditioning tests in these chunks may become less reliable, potentially leading to errors in estimation of MB(T). Also the possibility of overflow (i.e., data not fitting in a single node) increases accordingly. Hence cpIAMB is expected to be more sensitive to small sample size than the regular or fine-grain parallel IAMB and may return a different estimate of MB(T).

If the full set of data does not fit in RAM and only one processor is used, both cpIAMB and fgIAMB can be used to alleviate the problem. Each algorithm only requires a chunk of data in RAM to be processed at a time. cpIAMB loads a different chunk $ch\times 2$ times (once per phase). fgIAMB loads each chunk $ch\times$|max(currentMB)| times. Thus for most cases, cpIAMB, when run on a single CPU and when the data do not fit in the RAM, should be faster than fgIAMB. The complexity of cpIAMB is O(D×N×M’/ch) where M’ is the maximum size of tentative MBs over all chunks and phases.

```
Input: a  MxN array of data with N number of features (variables) and
       M number of instances, a target feature T, a desired number ch
       of data chunks

Phase I:

1.  Split the data into ch arrays Ci of equal size, such that each
      array contains a non-overlapping subset of the features plus T
2. For all i let MB(T|Ci)← IAMB(T, Ci)
3. M1= IAMB(T, ∪i (MB(T | Ci ))

PhaseII:

1.  Split the data into ch arrays Ci of equal size, such that each
      array contains a non-overlapping subset of the features plus M1
      and T
2. For all i let MB(T|Ci)← IAMB(T, Ci)
3. M2= IAMB(∪i (MB(T | Ci ))


Return M2
```

Figure 2. The cpIAMB algorithm.

4 Thrombin Experiment

4.1 Task

We apply the IAMB variants to the Thrombin dataset from KDD Cup 2001 [4]. The problem involves classification of chemical substances as biologically active (i.e., binding to Thrombin, a key receptor in blood clotting) on the basis of molecular properties of each substance. This problem involves a very large number of features (139,351), and an equally large feature-to-sample ratio (up to 107 in our experiments depending on the data split), forming an “acid test” for any feature selection algorithm. Furthermore, this is a publicly available dataset thus enabling future comparisons of our methods with other feature selection algorithms. We note that since in the original KDD Cup problem statement the test set was not derived from the same population as the train set (the included substances were created by chemists after analysis of the properties of instances in the train set) this violates assumption 3 (iid). Classification when iid is violated is highly controversial given a cross-sectional dataset and in the absence of any other knowledge about the domain and/or the characteristics of the test set. As expected, the best of the 114 entrants to the KDD Cup competition that analyzed this dataset exhibited a poor performance. An independent subsequent analysis using transduction and testset-tailored methods[*] performed better [21]. Since we wished to examine the performance of our

[*] This bias is acknowledged by the authors of that study, however.

algorithms when assumptions 1-3 *do* hold and assuming no knowledge of the test set (as we believe is the case in most practical settings) we “restored” iid as follows in our experiment: we took the union of all available data and divided it randomly in a train and test set; we further divided the train set into a “traintrain” and a validation set. The traintrain set was used to develop classifiers under different parameter settings; the validation set was used to obtain the best parameter set for each classifier family; then the full train set was used to develop classifiers employing the best parameter set found in the previous step; finally unbiased estimates of absolute and comparative performance were derived by application to the test set. Table 1 shows the characteristics of the data splits.

Table 1. Summary of the data splits.

DATA SET	SIZE (% OF split)	ACTIVE	INACTIVE	% OF ACTIVE IN SPLIT
TOTAL	2543 (-)	192	2351	7.6
TRAIN	2000 (78.7)	151	1849	7.6
TRAIN-TRAIN	1300 (65)	90	1210	6.9
VALIDATION	700 (35)	61	639	8.7
TEST	543 (21.3)	41	502	7.6

4.2 Methods

The classifier families used in our experiments were: *Support Vector Machines* (SVMs) [1], *Neural Networks* (NNs), *K-Nearest Neighbors* (KNN), and the *Simple Bayes Classifier* (SBC) [7]. Next, we review the two methods used as baseline comparison references[*]. Table 2 presents the parameter ranges and the total number of models (number of parameters×number of feature selection methods) built per classifier family.

[*] The winning entry in the KDD Cup competition (from 114 total submissions) was derived by choosing the 200 strongest univariate predictors and then running a full-network induction algorithm, then choosing the MB and classifying. The best model found using this process had four predictors. Although our original intent was to replicate the same methodology as one of the baseline comparisons, we found it impossible to replicate the original results using the publicly available software, (even after contacting the authors) and thus this baseline was abandoned.

Recursive Feature Elimination (RFE). RFE uses linear SVMs [1,9] to select features. The method consists of first developing a linear SVM model on the full predictor set. The linear SVM model is simply:

$$\text{Sign}\,(\mathbf{w} \bullet \mathbf{x} + b)$$

where **w** is a vector of weights for each feature, **x** is an input instance vector, and b is a threshold. Clearly, if $\mathbf{w}_i$ is zero, then the feature $\mathbf{X}_i$ does not influence classification and thus can be eliminated from the set of predictors. Once **w** has been computed for the full predictor set, the predictor variables are sorted according to their weights in descending order and the lower half is discarded. A new linear SVM is built using this new set of features and the whole process is repeated until the set of predictors is non-divisible by two. Finally the best predictor subset (i.e., the one with the highest classification performance) is chosen [9]. In case of ties, the subset with the smallest number of predictors was used.

Strongest Univariate Associations Filtering (UAF). This is a method widely used in statistics and data mining. The method consists of sorting all predictor variables in descending order according to their univariate association to the target and include in the classification model only the first *m* ones. In this experiment we did not assume an arbitrary *m* but instead optimized it like any other regular parameter. It turned out that optimal *m* was the full train set in each split. Thus, in the reported results, *full data* and *UAF* produce identical results.

4.3 Performance Metric and Evaluation

In all reported experiments we used area under the receiver operator characteristic curve (ROC) (abbreviated as "AUC") to evaluate the quality of the produced models [18]. The ROC is a plot of the sensitivity of a classifier against 1-specificity for several decision thresholds as follows: for KNN we picked decision thresholds on the percentage of votes for the active target class among the *k* closest training instances; for SVMs we picked thresholds on the distance from the optimal hyperplane; for SBC we chose thresholds on the probability of the active class, and for NNs on the activation level of the output unit.

We used the SVMlight library [11], the NN toolbox of Matlab [15] and our own Matlab implementations for the remaining methods.

4.4 Experimental Results

Table 3 shows the relative performance of all IAMB variants for all classifiers and feature selection methods. Table 4 summarizes the relative performance in terms of time of the IAMB variants when run on a single node and a 14-node parallel Beowulf cluster [22] using MPI as message passing interface. We note that for comparison purposes we used a slow single-CPU machine and that for highpower workstations the time of running regular IAMB on a single CPU is actually an order of magnitude faster. In addition, whenever possible we use dense arrays (as they are much faster to process), but in the case of one-CPU IAMB sparse arrays had to be used to fit the data in memory. We also consider two distinct application scenarios for when data is distributed among several computer nodes: in the first scenario the data has to be distributed before processing begins, and in the second scenario the data is already distributed (thus the related data transfer costs are zero).

Table 2. Summary of parameter ranges and total models examined per classifier (distributed equally over all feature selection methods).

CLASSIFIER	PARAMETER RANGES	# OF MODELS EXAMINED
KNN	K=1 to #cases in training data	5,204
LINEAR SVM	C=(10^{-5}, 10^{-3}, 0.01, 0.1, 1, 2, 5, 10, 100)	40

POLYNOMIAL SVM	C=(10^{-5}, 10^{-3}, 0.01, 0.1, 1, 2, 5, 10, 100)
	D=(2, 3, 4)
		112

RBF SVM	C=(10^{-5}, 10^{-3}, 0.01, 0.1, 1, 2, 5, 10, 100)
	g=(0, 10^{-5}, 10^{-3}, 0.01, 0.1, 1, 2, 5, 10)
		328

SIMPLE BAYES	-	4
FEED -FORWARD NNs	1 to 2 hidden layers 1 to number of inputs hidden units (up to a maximum of 50) momentum (0, 0.001, 0.01, 0.1, 0.5, 1)	720

Conclusions: cpIAMB selects roughly equal-quality feature subsets as fgIAMB. The cpIAMB (a) enables one to run the algorithm on a single CPU when the data do not fit in the fast memory, (b) achieves an order of magnitude faster execution on a single CPU than IAMB (the latter requires a sparse array representation of the data to fit in memory), and (c) achieves 135 times faster execution in this dataset and 14-node parallel platform over one-CPU IAMB. fgIAMB allows a 180-fold speedup in this dataset and parallel platform. The actual reported times are especially encouraging if one takes into account that we used a small parallel cluster with low-performance nodes and unoptimized (interpreted) Matlab implementations of the algorithms.

It appears that IAMB variants, on the average, and over 6 widely-used classifier families, perform better than RFE and UAV when used as filter feature selection algorithms in the thrombin task and dataset. Surprisingly, the highest-performing model was not obtained using some SVM model (either with or without feature selection) but instead when coupling IAMB with the simple Bayes classifier. Also notice that the selected feature sets for IAMB variants contain only 8 variables, whereas the best RFE subset contains 2177 variables.

5 Discussion

Koller and Sahami [13], Cooper et al [6] and Cheng and Greiner [3] have explored Markov blanket methods for feature selection. All these methods are heuristic and/or intractable for very large datasets. Li et al have investigated the use of Genetic Algorithm wrappers over KNN for feature selection in gene expression

Table 3. Overall relative classification performance (as measured by AUC) of chunked, non-chunked IAMB, RFE and All features/UVF in test set selection.

	FILTERS			UAF/ALL FEATURES	
CLASSIFIER	IAMB	CHUNKED IAMB	RFE		AVERAGE OVER ALL FILTERS
KNN	0.91	0.90	0.84	0.87	0.88
LINEAR SVM	0.85	0.84	0.89	0.89	0.86
POLYNOMIAL SVM	0.86	0.89	0.88	0.90	0.88
RBF SVM	0.82	0.83	0.92	0.92	0.86
SIMPLE BAYES	0.96	0.92	0.60	0.82	0.83
FEED -FORWARD NNs	0.84	0.88	0.87	-	0.86
AVERAGE OVER ALLCLASSIFIERS	0.87	0.88	0.83	-	
# OF SELECTED FEATURES	**8**	**8**	**2177**	**139,351**	

Table 4. Relative performance (in terms of time) of parallel and chunked IAMB variants.

ALGORITHM	TOTAL TIME (HRS)	NOTES
IAMB	72	1 CPU, data in sparse array, 128MB RAM, 600 MHz PIII (100% load)
Chunked IAMB	6.69	1 CPU, data in dense array, 256MB RAM, 600 MHz PIII (100% load)
Fine-Grain Parallel IAMB	0.5	14 CPUs, data in dense arrays, 256MB RAM, 600 MHz PIII (100% load)
Fine-Grain Parallel IAMB distributed data	0.4	14 CPUs, data in dense arrays, 256MB RAM, 600 MHz PIII (100% load)
Chunked Parallel IAMB	0.53	14 CPUs, data in dense arrays, 256MB RAM, 600 MHz PIII (100% load)
Chunked Parallel IAMB distributed data	0.53	14 CPUs, data in dense arrays, 256MB RAM, 600 MHz PIII (100% load)

data [14]. Caruana et al have explored greedy wrapper methods over decision tree induction [2]. As discussed earlier, however, heuristic search approaches do not provide any guarantees of finding an optimal subset (even in the sample limit), and need be repeated for all desired classifiers.

In the reported experiments we have used thresholded mutual information as a test of conditional independence. We did so because this measure is an efficient *heuristic* approximation of more principled statistical association metrics and independence tests and admits any form of functional dependence. The threshold was not optimized in the reported experiments (instead we used a threshold that was found to be successful in previous simulated studies). We plan to explore a rigorous optimization of this threshold, as well as more statistically principled measures such as the G^2 test and Fisher's z-test as employed by Spirtes et al [19]. We are also in the process of developing variants of IAMB that operate on sample sizes small relative to the size of the MB(T) by restricting the class of distributions to a subset of BN-faithful processes. Finally, we have applied here IAMB in a single task but several additional evaluations using simulated and real data from gene expression, clinical, and text categorization domains are being pursued.

6 Acknowledgements

The authors acknowledge DuPont Pharmaceuticals and the KDD Cup 2001 organizers for making the Thrombin dataset available.

References

1. Burges C.J.C. A tutorial on support vector machines for pattern recognition. Data Minining and Knowledge Discovery. 1998, 2(2):1-47.
2. Caruana, R. and Freitag D. Greedy attribute selection. In International Conference on Machine Learning, 1994.
3. Cheng J., Greiner R. Comparing bayesian network classifiers. In Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI), 1999.
4. Cheng J. et al. KDD Cup 2001 Report. SIGKDD Explorations. 2002, **3** (2): P.1-18.
5. Cooper G.F., Herskovits E.: A Bayesian method for the induction of probabilistic networks from Data. Machine Learning 9: 309-347 (1992)
6. Cooper, G.F., et al., An evaluation of machine-learning methods for predicting pneumonia mortality. Artificial Intelligence in Medicine, 1997. **9**: p. 107-138.
7. Duda, R.O., Hart P.E., and Stork D.G., Pattern Classification. (Second ed.) John Wiley and Sons, 2001.
8. Glymour, C. and Cooper G.F., eds. Computation, Causation, and Discovery. AAAI Press/The MIT Press, 1999.

9. Guyon, I., Weston J., Barnhill S., et al., Gene selection for cancer classification using support vector machines. Machine Learning, 2002, 46: 389-422.
10. Heckerman D. A Bayesian approach to learning causal networks. Technical Report MSR-TR-95-04, Microsoft Research, March, 1995.
11. Joachims T. Learning to classify text using support vector machines. Dissertation, Kluwer, 2002.
12. Kohavi, R. and John G.H. Wrappers for feature subset selection. Artificial Intelligence, 1997. **97**(1-2): p. 273-324.
13. Koller, D. and Sahami M. Toward optimal feature selection. In: Thirteenth International Conference in Machine Learning, 1996.
14. Li L. et al. Computational analysis of leukemia microarray expression data using the GA/KNN method. In: Lin A.M., and Johnson K.F., editors. Methods of microarray data analysis. Kluwer Academic Publishers, 2002.
15. MATLAB 6.1, The MathWorks Inc.
16. Neapolitan, R.E. Probabilistic reasoning in expert systems: theory and algorithms. John Wiley and Sons, 1990.
17. Pearl, J. Probabilistic reasoning in intelligent systems. Morgan Kaufman, 1988.
18. Provost F., Fawcet T., Kohavi R. The case against accuracy estimation for comparing induction algorithms. In: Fifteenth International Conference on Machine Learning, 1998.
19. Spirtes, P., Glymour C., and Scheines R. Causation, prediction, and search. (Second ed.) The MIT Press, 2000.
20. Tsamardinos I., Aliferis C.F. Algorithms for local causal discovery. Technical Report DSL-02-03 (7-01-2002), Vanderbilt University. (Available on request from first author).
21. Weston J. et al. Feature selection and transduction for prediction of molecular bioactivity for drug design. Bioinformatics, 2002; 1(1):1-8.
22. VAMPIRE. URL: http://www.vampire.vanderbilt.edu/