# Self-Labeled Hidden Naive Bayes Algorithm for Semi-Supervised Classification

Nikos Fazakis
Department of Electrical Engineering, University of Patras
Patras, Greece
fazakis@ece.upatras.gr

Stamatis Karlos
Department of Mathematics, University of Patras
stkarlos@upatras.gr

Sotiris Kotsiantis
Department of Mathematics, University of Patras
sotos@math.upatras.gr

Kyrgiakos Sgarbas
Department of Electrical Engineering, University of Patras
sgarbas@upatras.gr

***Abstract*— Exploiting both labeled and unlabeled instances of various problems seems a really promising strategy, since useful information that is contained on the latter pool of data is discarded during supervised approaches. However, the size of the unlabeled data that needs to be examined is usually extremely large and efficient algorithms should be utilized in such cases. Hidden Naive Bayes (HNB) model constitutes a computational cheap variant of Bayesian networks. In this work, HNB has been used as the base classifier of Self-training scheme for classification problems. Its results over 36 UCI datasets prove that a robust behavior can be achieved with only one hidden layer even under strict time restrictions.**

***Keywords—Hidden Naive Bayes; Self-labeled methods; Classification task; Bayesian network; labeled/unlabeled instances***

## I. INTRODUCTION

Real-life applications constitute issues of utmost importance among Machine Learning (ML), Pattern Recognition (PR) and Statistical Learning fields the last years. Tasks like classification and clustering fit with the most dominant demands of them. Although supervised scenarios seem efficient enough for dealing with the most similar problems, they suffer from the demand of large amounts of labeled data. This fact is opposed with the reality, since labeled data are usually limited while vast amounts of unlabeled data are available. Moreover, the lack of any automated procedure that could annotate labels to the existing unlabeled data with high accuracy in the majority of these applications leads to the need of human expertise. But, this fact means that too much human effort needs to be spent under slow processes. On the contrary, semi-supervised learning (SSL) algorithms take advantage of the knowledge that is hidden in unlabeled data and are trying to combine both labeled and unlabeled examples for achieving more robust behaviors [1].

Simplicity of Bayesian theory has attracted the interest of scientific community over many fields, besides its inability to be highly harmonized with the most physical problems. Simple Naïve Bayes (NB) classifier [2], Tree-augmented Naïve Bayes (TAN) [3], generalized Bayesian Networks (BayesNet) [4], [5], Naïve Bayes Tree (NBTree) [6], Locally Weighted Naïve Bayes (LWNB) [7] and Averaged One-Dependence Estimators (AODE) [8] are some of the most well-known algorithms that have been produced combining NB theory with specific structures or methods. The main asset in this family of algorithms is the fact that the existence of only a few training instances do not prevent the corresponding classifier from its proper functionality, while the requirement of datasets whose features satisfy the conditional independence property is demanded for excellent performance. However, even if this prerequisite does not hold, the achieved performance can be maintained in equally well level, either by simpler or more sophisticated modifications.

Bayesian networks classifiers constitute the most general category of NB algorithms, since all the other classifiers are obtained by using networks with specific structure. They can be represented via directed graphs whose vertices and edges describe each different feature and each current correlation of the corresponding vertices that are linked through these, respectively. These correlations are expressed through conditional probabilities for each vertex in such networks given the state of its parents. The generalized formula that accompanies them is the following:

$$class(x) = \arg \max_{class \in C} P(class) * P(x^1\, x^2 \dots x^k | class) \qquad (1)$$

Equation 1 holds for each instance $x$ with $k$ separately features while $C$ symbolizes a vector with all the existing classes of the problem. The simplest Bayesian network corresponds to the simple NB classifier, where no dependencies among the features exist. Therefore, the only edges are these among the class vertex and each of the feature vertices. A more advanced network is this of TAN approach. Instead of assuming strong independency relationships, it is permitted to each feature node to be linked with one feature at most. This strategy is really affordable, since both improved performance is obtained and without highly increasing the computational complexity. More specific, the search time for these structures is bounded into polynomial time [9]. On the contrary, it has been proven that searching for an optimal Bayesian network classifier is NP-hard problem [10]. Consequently, this trade-off behavior should be respected when trying to build efficient Bayesian network classifiers. Hidden Naive Bayes (HNB) model constitutes such an approach [11]. Its structure is differentiated from the default Bayesian networks, since a hidden parent is produced for each contained feature variable and is connected with this. Each one of these incoming vertices is assigned with a value, coming from a related metric, and describes the dependency between the feature that is oriented to be attached and the rest of the other features. The addition of this layer merges the existing correlations into a strict framework, even for large scale datasets, without highly increasing the complexity of the network. Its better overall classification behavior against NB classifier has also been demonstrated by its authors.

Despite the large appeal of Bayesian networks, only a few approaches have combined them with semi-supervised algorithms [12]. In our work, HNB classifier is exploited under a self-training wrapper method (Self-HNB) for classification tasks. Its performance was compared with 21 state-of-the art semi-supervised algorithms over 3 different Labeled Ratio scenarios and was evaluated with statistical tests, proving both its efficiency and its robust behavior. The rest of this work consists of a short presentation of the most known semi-supervised methods, while in Section 3 a description of the proposed algorithm is demonstrated. Experimental results are contained in Section 4. Finally, conclusions and future works are placed in Section 5.

## II. Semi-Supervised Techniques

Semi-Supervised algorithms are discriminated by supervised approaches because of the exploitation of unlabeled instances. Since every instance is described as a row vector with $k$ real different values $x = \{x^1\ x^2 \ldots\ x^k\}$, that represents an instance of the feature vector space ($x \subseteq R^k$), the difference between labeled and unlabeled instances is that the former category is extended by the class label – given from a predefined set of different classes – while this information is missing in the latter. Consequently, semi-supervised classification's (SSC) aim is to predict the class for the unlabeled examples, based on a few labeled examples. The parameter that defines the quantity of the available labeled ($L$) instances in comparison with the unlabeled ($U$) is called Labeled Ratio ($R$) and is expressed by the next formula:

$$R(\%) = \frac{numel(L)}{numel(L) + numel(U)} \tag{2}$$

Another definition of semi-supervised algorithms has been established by Schwenker and Trentin [13]. Thus, Partially Supervised Learning (PSL) term is used for describing the procedure according which these algorithms are trying to adjust their classification model. Moreover, in this work a taxonomy of several PSL approaches is recorded. Similar contribution has also been conducted by Triguero et al. in [14], mainly examining the self-labeled techniques. This characterization regards the methods that expand their available training data ($L$) extracting appropriate instances from the $U$ dataset. Their basic properties along with various useful criteria for comparing them are being reviewed in depth.

One of the most usual criterion for distinguishing SSC methods is the number of the views that are exploited. All the possible methods can be categorized into two wide families: single-view and multi-view methods. The most representative method of the former category is the self-training approach. This is an iterative wrapper method that does not make any specific assumption about the provided data. Given a classifier, which outputs class probabilities, it searches for the most confident instances that are included in the $U$ dataset, based on the learned hypothesis from the initial labeled instances. The most informative of them are removed from the $U$ and are joined to the $L$ subset at the end of each iteration. The final hypothesis is being formatted after a specific number of iterations or after a satisfying number of instances has been added to the $L$. This unbounded confidence that is performed by the default self-training scheme may lead to vulnerable classification behaviors, especially when the accuracy of the initial training data is poor. For this reason, many variants have been proposed trying to eliminate the looming misclassified instances that would appear. One of the most known is the SETRED algorithm [15]. Its restriction rule is based on graph theory. As a consequence, any observed instance that is theorized as misclassified is rejected from being added to the $L$.

On the other hand, multi-view methods require two or more redundant and sufficient views for achieving robust behaviors. The basic principle here is the training of each different view with the same or different classifiers for boosting the learning ability through disagreement-based theory [16]. Co-training [12] is the most well-known multi-view scheme. Its simplicity has contributed to be applied in many real-word applications, since only two views are needed, besides the fact that it is rare not to be violated the assumption about the conditional independence of the existing views, given the class annotation. RASCO [17] and Rel-RASCO [18] use random splits of the feature space ($F$) for training different learners. Co-training by Committee [19] is another famous tactic for combining different classifiers with co-training scheme.

Alternative approaches, which could be theorized as a hybrid strategy between single/multi-view methods, are using combinations of different learners. Tri-training scheme [20] employs three different classifiers trained on the $L$ subset through bootstrap sampling. Only if the decisions from two of them are the same about the label of the instances inside the $U$ subset, the $L$ subset is being expanded with them. Co-Forest [21] has also been proposed by the same authors as a more specialized algorithm. ADE-Co-Forest [22] constitutes a direct improvement of its ancestor, since an editing technique has been added. Democratic-co [23] also uses multiple algorithms over the same dataset and its final prediction is based on majority voting. More sophisticated semi-supervised techniques, such as Aggregation Pheromone density Semi-Supervised Classification (APPSC) [24] and Local Cluster Centers with a few labeled training examples [25] can be found in literature.

## III. Proposed Algorithm

Our proposed algorithm uses HNB as a base classifier under self-training scheme. The general idea of HNB is to improve the behavior of simple NB theory by inserting new variables that should enrich the relationship between the class category and each initial feature, since the assumption about the feature independency is not verified in the majority of the occasions that ML algorithms have to tackle. Furthermore, the rules for constructing HNB model do not overcome the polynomial complexity that is also needed from others variants of NB theory, leading in this way to an efficient learner.

```
Algorithm Self-HNB
  Input:
  HNB – Hidden Naive Bayes as base classifier
  D – Initial training dataset
  R – Ratio of labeled instances along D
  L – Initial labeled instances, $L \subseteq D$
  U – Initial unlabeled instances, $U \subseteq D$
  T – Predefined threshold for accepted accuracy
  $x_{MCP}$ – Instances with Most Confident Predictions
  MaxIter – number of maximum iterations performed
      1. Initialization
          a. Replace Missing Values in L and Discretize
          b. Train HNB as base model on L
      2. Loop for a number of iterations (MaxIter is equal to 40
         for our implementation)
          a. Use HNB classifier to select the instances with Most
             Confident Predictions per iteration ($x_{MCP}$)
          b. Remove $x_{MCP}$ from U and add them to L.
          c. In each iteration a few instances per class are selected
             from U and added to L.
          d. Re-train HNB as base model on new enlarged L.
 Output:
    Use HNB trained on L to predict class labels of the unknown test
    cases.
```

Fig. 1. The Self-training HNB Algorithm

To be more specific, a layer of new variables is being constructed for describing the correlation of each initial feature with the rest of them. This "hidden" layer helps the existing Bayesian network to represent more accurately the structure of each tested problem. Consequently, for a general dataset with $N$ instances and a feature set ($F$) that consists of $k$ variables, another $k$ new variables will be computed ($f^i_{hid\_var}, i = 1,2 \dots k$) and will be connected only with the corresponding initial feature ($f^i$). The joint distribution of the structure that HNB supports is presented here:

$$P(x^1, x^2 \dots x^k, class) = P(class) * \prod_{i=1}^{k} P(x^i \,|x^i_{hid\_var}, class) \quad (3)$$

The term inside the product factor is defined as follows:

$$P\left(x^i \middle| x^i_{hid_{var}}, class\right) = \sum_{j=1}^{k} W_{ij} * P\left(x^i \middle| x^j, class\right), \forall j \neq i \quad (4)$$

Because the generated variables have to summarize the influence from all over the features except from the feature itself that is going to be connected, the computation of the $W_{ij}$ weights play a cardinal role on the whole model. The suggested strategy by the authors of HNB is to introduce the conditional mutual information ($I_P$) as the weighting function for capturing more precisely the current relationships. The following formulas define the appropriate functions:

$$W_{ij} = \frac{I_P(x^i;x^j|class)}{\sum_{j=1}^{k} I_P(x^i;x^j|class)}, \ \forall j \neq i \quad (5)$$

$$I_P(x^i; x^j \middle| class) = \sum_{v_i,v_j,class} P(v_i, v_j, class) \log \frac{P(v_i,v_j|class)}{P(v_i|class)*P(v_j|class)} \quad (6)$$

Another comment about the previous formulas is the fact that probabilities $P(class)$ and $P(x^i \middle| x^j, class)$ are computed according to M-estimation [11]. The computation time for estimating all the necessary weights and variables still remains under polynomial boundary, being comparable with the rest of the variants of simple NB classifier. However, its framework seems to express the underlying dependencies with a more successful manner, since each hidden variable stems from a filtering stage that examines all the initial features of the dataset.

The properties of the self-training scheme that has been combined with HNB are demonstrated in Fig. 1. During the initialization step, we discretize a range of numeric attributes in the $L$ into nominal attributes. Discretization is done with [26]. No specific restriction has been added to the default scheme. However, for avoiding the insane self-confidence of self-training, the value of the threshold ($T$) has been set equal to 0.9. This means that only the instances that have been annotated with a confidence value greater that $T$ would be permitted to be added on the $L$. In this way, the rate of misclassifying instances may be restricted enough. The whole procedure is repeated over MaxIter times. At the end, the classification model is constructed based on the expanded $L$.

## IV. Experiments

The experiments are based on 36 standard classification datasets extracted from the KEEL-dataset repository [27] covering a wide range of scientific fields. These datasets have been partitioned using the 10-fold cross-validation theory and its training partition is split into labeled and unlabeled subsets according to the selected $R$ value. In order to study the influence of the amount of labeled data, three different ratios were used: 10%, 20%, and 30%.

Next, the proposed method was compared with 21 state of the art algorithms into the KEEL tool: Self-Training (NB) [14], Self-Training (C45) [28], SETRED [15], Co-Training (NB) and Co-Training (C45) [14], Democratic-Co [23], TriTraining (NB) [14], TriTraining (C45) [29], DE-TriTraining (NB) [14], DE-TriTraining (C45) [30], CoForest [21], Rasco (NB) and Rasco (C45) [17], Rel-Rasco (NB) and Rel-Rasco (C45) [18], Co-Bagging (NB) [14], Co-Bagging (C45) [19],Ade-Co-Forest [22]. For all tested algorithms, the default parameters of KEEL were used. Here, we present only 5 algorithms, including Self-HNB. Their classification accuracy is recorded in Tables I, II and III. The supplemental file with the total results can be found in http://ml.math.upatras.gr/wp-content/uploads/2016/04/Self_HNB_Results.xlsx. The maximum achieved score for any dataset has been formatted in bold style.

Beyond the average accuracy, a statistical comparison of the performance of the tested algorithms has been also applied for all the selected values of $R$. For this reason, Friedman test together with two similar post hoc statistical tests (Holm/Hochberg) described in [31] have been selected. According to the former, a ranking of the algorithms for each one of the contained datasets is produced. Next, a comparison of the average ranks of the algorithms is executed. The null hypothesis states that all the algorithms are equivalent, getting same average ranking over any dataset. The results are presented in tables IV and V.

TABLE I. LABELED RATIO 10%

| Datasets | Algorithms | | | | |
|---|---|---|---|---|---|
| | *Self (HNB)* | *Self (C4.5)* | *Co Train (NB)* | *Democ* | *Tri Train (C45)* |
| abalone | 0.2187 | 0.2041 | 0.2024 | 0.2106 | 0.2161 |
| automobile | 0.4396 | 0.4052 | 0.3541 | 0.3601 | 0.3889 |
| breast | 0.7028 | 0.7216 | 0.7401 | 0.7287 | 0.7216 |
| contraceptive | 0.4202 | 0.4886 | 0.4623 | 0.4358 | 0.4813 |
| dermatology | 0.9357 | 0.8562 | 0.6670 | 0.8760 | 0.8816 |
| flare | 0.7158 | 0.7214 | 0.6952 | 0.7214 | 0.7158 |
| german | 0.7000 | 0.7060 | 0.7190 | 0.7160 | 0.7170 |
| haberman | 0.7255 | 0.7054 | 0.7448 | 0.7156 | 0.7088 |
| heart | 0.7593 | 0.6778 | 0.8000 | 0.8000 | 0.7148 |
| hepatitis | 0.8434 | 0.8343 | 0.6115 | 0.8343 | 0.8343 |
| lymphography | 0.7303 | 0.6312 | 0.4734 | 0.4901 | 0.6118 |
| magic | 0.8100 | 0.8217 | 0.7208 | 0.7842 | 0.8245 |
| mammographic | 0.8146 | 0.8025 | 0.7441 | 0.7963 | 0.8183 |
| marketing | 0.2833 | 0.2845 | 0.2865 | 0.2710 | 0.2694 |
| mushroom | 0.9970 | 0.9966 | 0.9204 | 0.9927 | 0.9955 |
| nursery | 0.8390 | 0.9064 | 0.8941 | 0.8951 | 0.9039 |
| page-blocks | 0.9501 | 0.9523 | 0.8785 | 0.9077 | 0.9561 |
| penbased | 0.9380 | 0.8916 | 0.8534 | 0.9474 | 0.9027 |
| pima | 0.7045 | 0.6643 | 0.7033 | 0.6967 | 0.6564 |
| ring | 0.9468 | 0.8396 | 0.9801 | 0.8741 | 0.8542 |
| saheart | 0.6603 | 0.6516 | 0.6864 | 0.6819 | 0.6776 |
| satimage | 0.8553 | 0.8045 | 0.7932 | 0.8462 | 0.8224 |
| segment | 0.8814 | 0.8900 | 0.7719 | 0.9026 | 0.9000 |
| sonar | 0.6619 | 0.6433 | 0.6436 | 0.6005 | 0.7019 |
| spambase | 0.9002 | 0.8669 | 0.8369 | 0.8777 | 0.8810 |
| spectfheart | 0.7534 | 0.6819 | 0.7115 | 0.7379 | 0.7574 |
| splice | 0.8843 | 0.8266 | 0.9125 | 0.8978 | 0.8254 |
| texture | 0.8844 | 0.8305 | 0.7607 | 0.8944 | 0.8524 |
| thyroid | 0.9592 | 0.9922 | 0.9282 | 0.9393 | 0.9918 |
| titanic | 0.7665 | 0.7751 | 0.7706 | 0.7756 | 0.7765 |
| twonorm | 0.9597 | 0.8136 | 0.9770 | 0.9645 | 0.8616 |
| vehicle | 0.5225 | 0.5792 | 0.4670 | 0.5023 | 0.6194 |
| wine | 0.9438 | 0.7405 | 0.8980 | 0.9493 | 0.8203 |
| wisconsin | 0.9284 | 0.9093 | 0.9446 | 0.9650 | 0.9312 |
| yeast | 0.4307 | 0.4616 | 0.4784 | 0.4886 | 0.4907 |
| zoo | 0.9231 | 0.6794 | 0.8831 | 0.9314 | 0.7192 |

TABLE II. LABELED RATIO 20%

| Datasets | Algorithms | | | | |
|---|---|---|---|---|---|
| | *Self (HNB)* | *Self (C4.5)* | *Co Train (NB)* | *Democ* | *Tri Train (C45)* |
| abalone | 0.2135 | 0.2022 | 0.2209 | 0.2010 | 0.2029 |
| automobile | 0.5820 | 0.5158 | 0.4375 | 0.4744 | 0.4938 |
| breast | 0.7302 | 0.7156 | 0.7399 | 0.7184 | 0.7225 |
| contraceptive | 0.4515 | 0.4779 | 0.4575 | 0.4841 | 0.4881 |
| dermatology | 0.9607 | 0.9100 | 0.8371 | 0.9300 | 0.9214 |
| flare | 0.7327 | 0.7280 | 0.7383 | 0.7402 | 0.7270 |
| german | 0.7220 | 0.6970 | 0.7340 | 0.7290 | 0.6840 |
| haberman | 0.7154 | 0.7089 | 0.7415 | 0.7222 | 0.7089 |
| heart | 0.8333 | 0.7519 | 0.8259 | 0.8222 | 0.7926 |
| hepatitis | 0.8309 | 0.8434 | 0.7543 | 0.8343 | 0.8434 |
| lymphography | 0.7909 | 0.7055 | 0.4210 | 0.4612 | 0.7545 |
| magic | 0.8182 | 0.8304 | 0.7246 | 0.8017 | 0.8335 |
| mammographic | 0.8195 | 0.8229 | 0.7794 | 0.8084 | 0.8157 |
| marketing | 0.3023 | 0.2891 | 0.3016 | 0.2879 | 0.2844 |
| mushroom | 0.9995 | 0.9991 | 0.9366 | 0.9980 | 0.9989 |
| nursery | 0.8731 | 0.9235 | 0.8971 | 0.9108 | 0.9258 |
| page-blocks | 0.9572 | 0.9602 | 0.8949 | 0.9115 | 0.9611 |
| penbased | 0.9532 | 0.9241 | 0.8511 | 0.9632 | 0.9287 |
| pima | 0.7004 | 0.6810 | 0.7100 | 0.7319 | 0.6939 |
| ring | 0.9595 | 0.8658 | 0.9791 | 0.8969 | 0.8795 |
| saheart | 0.6584 | 0.6515 | 0.6884 | 0.6972 | 0.6777 |
| satimage | 0.8637 | 0.8238 | 0.7977 | 0.8612 | 0.8353 |
| segment | 0.9165 | 0.9264 | 0.7853 | 0.9307 | 0.9247 |
| sonar | 0.6821 | 0.6636 | 0.6740 | 0.6474 | 0.6631 |
| spambase | 0.9071 | 0.8912 | 0.8286 | 0.8941 | 0.8941 |
| spectfheart | 0.7685 | 0.7192 | 0.6858 | 0.7305 | 0.7491 |
| splice | 0.9257 | 0.8834 | 0.9379 | 0.9119 | 0.8843 |
| texture | 0.9185 | 0.8669 | 0.7669 | 0.9176 | 0.8929 |
| thyroid | 0.9714 | 0.9935 | 0.9267 | 0.9419 | 0.9939 |
| titanic | 0.7651 | 0.7824 | 0.7706 | 0.7797 | 0.7815 |
| twonorm | 0.9662 | 0.8165 | 0.9769 | 0.9707 | 0.8674 |
| vehicle | 0.5899 | 0.6489 | 0.4539 | 0.4833 | 0.6620 |
| wine | 0.9379 | 0.8369 | 0.9487 | 0.9542 | 0.8422 |
| wisconsin | 0.9536 | 0.9343 | 0.9578 | 0.9637 | 0.9253 |
| yeast | 0.5088 | 0.5263 | 0.5276 | 0.5492 | 0.5486 |
| zoo | 0.9172 | 0.8311 | 0.9067 | 0.8900 | 0.8264 |

TABLE III. LABELED RATIO 30%

| Datasets | Algorithms | | | | |
|---|---|---|---|---|---|
| | ***Self (HNB)*** | ***Self (C4.5)*** | ***Cotrain (NB)*** | ***Democ*** | ***TriTrain (C45)*** |
| abalone | 0.2362 | 0.2139 | 0.2228 | 0.2147 | 0.2048 |
| automobile | 0.6246 | 0.5484 | 0.4993 | 0.5432 | 0.5707 |
| breast | 0.7365 | 0.7179 | 0.7433 | 0.7219 | 0.7000 |
| contraceptive | 0.4820 | 0.4922 | 0.4650 | 0.4847 | 0.5105 |
| dermatology | 0.9663 | 0.9209 | 0.8796 | 0.9180 | 0.9240 |
| flare | 0.7299 | 0.7299 | 0.7392 | 0.7485 | 0.7298 |
| german | 0.7260 | 0.7100 | 0.7320 | 0.7390 | 0.7030 |
| haberman | 0.7218 | 0.7153 | 0.7482 | 0.7447 | 0.7089 |
| heart | 0.8259 | 0.7556 | 0.8481 | 0.8296 | 0.7556 |
| hepatitis | 0.7827 | 0.8192 | 0.7975 | 0.8168 | 0.8334 |
| lymphography | 0.8019 | 0.7587 | 0.4067 | 0.7442 | 0.7315 |
| magic | 0.8164 | 0.8380 | 0.7259 | 0.8016 | 0.8411 |
| mammographic | 0.8101 | 0.8438 | 0.8024 | 0.8300 | 0.8425 |
| marketing | 0.3120 | 0.2872 | 0.3014 | 0.2902 | 0.2893 |
| mushroom | 1.0000 | 0.9996 | 0.9445 | 0.9995 | 0.9995 |
| nursery | 0.8897 | 0.9377 | 0.9005 | 0.9212 | 0.9362 |
| page-blocks | 0.9600 | 0.9635 | 0.8949 | 0.9289 | 0.9635 |
| penbased | 0.9614 | 0.9409 | 0.8542 | 0.9729 | 0.9431 |
| pima | 0.7411 | 0.7252 | 0.7307 | 0.7305 | 0.7045 |
| ring | 0.9642 | 0.8754 | 0.9796 | 0.9089 | 0.8881 |
| saheart | 0.7080 | 0.6753 | 0.6886 | 0.7080 | 0.6797 |
| satimage | 0.8706 | 0.8270 | 0.7953 | 0.8693 | 0.8438 |
| segment | 0.9290 | 0.9303 | 0.7918 | 0.9416 | 0.9381 |
| sonar | 0.7119 | 0.6762 | 0.6348 | 0.7310 | 0.6912 |
| spambase | 0.9113 | 0.8999 | 0.8284 | 0.9052 | 0.9073 |
| spectfheart | 0.7570 | 0.7496 | 0.6670 | 0.7121 | 0.7644 |
| splice | 0.9480 | 0.9166 | 0.9492 | 0.9188 | 0.9157 |
| texture | 0.9380 | 0.8891 | 0.7722 | 0.9331 | 0.9055 |
| thyroid | 0.9822 | 0.9946 | 0.9313 | 0.9521 | 0.9944 |
| titanic | 0.7678 | 0.7787 | 0.7733 | 0.7792 | 0.7783 |
| twonorm | 0.9693 | 0.8255 | 0.9774 | 0.9701 | 0.8743 |
| vehicle | 0.6183 | 0.6584 | 0.4350 | 0.5652 | 0.6702 |
| wine | 0.9320 | 0.8415 | 0.9549 | 0.9660 | 0.9039 |
| wisconsin | 0.9607 | 0.9443 | 0.9607 | 0.9666 | 0.9502 |
| yeast | 0.5324 | 0.5209 | 0.5357 | 0.5371 | 0.5405 |
| zoo | 0.9206 | 0.8231 | 0.9089 | 0.9133 | 0.7781 |

TABLE IV. FRIEDMAN RANKING FOR DIFFERENT LABELED RATIOS

| **Algorithms** | **R=10%** | **R=20%** | **R=30%** |
|---|---|---|---|
| *HNB* | *7.625* | *6.833333* | *6.625* |
| TriTraining (C45) | 7.75 | 8.388889 | 9.25 |
| Democratic-Co | 8.027778 | 7.402778 | 6.861111 |
| Co-Bagging (C45) | 8.611111 | 8.486111 | 8.791667 |
| CoForest | 8.916667 | 9.125 | 9.916667 |
| Self-Training (C45) | 9.75 | 9.638889 | 9.833333 |
| DE-TriTraining (C45) | 10.15278 | 10.11111 | 11.44444 |
| ADE-CoForest | 10.51389 | 10.98611 | 11.04167 |
| Co-Training (C45) | 10.65278 | 8.486111 | 9.930556 |
| Co-Training (NB) | 10.66667 | 10.29167 | 11.23611 |
| DE-TriTraining (NB) | 10.81944 | 10.09722 | 12.05556 |
| TriTraining (NB) | 10.84722 | 9.875 | 10.18056 |
| SETRED | 11.15278 | 11.81944 | 12.08333 |
| Co-Bagging (NB) | 11.40278 | 9.763889 | 11.23611 |
| Rel-Rasco (NB) | 12.19444 | 14.16667 | 10.83333 |
| Rasco (NB) | 12.30556 | 14.25 | 11.43056 |
| APSSC | 12.31944 | 12.83333 | 13.45833 |
| CLCC | 13.72222 | 14.91667 | 15.36111 |
| Self-Training (NB) | 14.11111 | 12.93056 | 13.70833 |
| Rasco (C45) | 14.55556 | 15.56944 | 12.54167 |
| Rel-Rasco (C45) | 14.90278 | 15.79167 | 13.04167 |

TABLE V. RANKING ACCORDING TO HOLM\HOCHBERG (ALPHA=0.05)

| **Algorithms** | **R=10%** | **R=20%** | **R=30%** |
|---|---|---|---|
| TriTraining (C45) | 0.05 | 0.016667 | 0.016667 |
| Democratic-Co | 0.025 | 0.05 | 0.05 |
| Co-Bagging (C45) | 0.016667 | 0.025 | 0.025 |
| CoForest | 0.0125 | 0.01 | 0.01 |
| Self-Training (C45) | 0.01 | 0.008333 | 0.0125 |
| DE-TriTraining (C45) | 0.008333 | 0.005 | 0.003846 |
| ADE-CoForest | 0.007143 | 0.004167 | 0.005556 |
| Co-Training (C45) | 0.00625 | 0.0125 | 0.008333 |
| Co-Training (NB) | 0.005556 | 0.004545 | 0.005 |
| DE-TriTraining (NB) | 0.005 | 0.005556 | 0.003571 |
| TriTraining (NB) | 0.004545 | 0.00625 | 0.007143 |
| SETRED | 0.004167 | 0.003846 | 0.003333 |
| Co-Bagging (NB) | 0.003846 | 0.007143 | 0.004545 |
| Rel-Rasco (NB) | 0.003571 | 0.003125 | 0.00625 |
| Rasco (NB) | 0.003333 | 0.002941 | 0.004167 |
| APSSC | 0.003125 | 0.003571 | 0.002778 |
| CLCC | 0.002941 | 0.002778 | 0.0025 |
| Self-Training (NB) | 0.002778 | 0.003333 | 0.002632 |
| Rasco (C45) | 0.002632 | 0.002632 | 0.003125 |
| Rel-Rasco (C45) | 0.0025 | 0.0025 | 0.002941 |

## V. Conclusions

Semi-supervised algorithms act as autonomous self-restricted schemes demanding only a little knowledge produced by human expertise. Moreover, the necessity for elaborating with vast amount of data in many synchronous applications can be tackled, inside tolerable time and space restrictions, only with fast learners. Bayesian networks that represent a map of the joint probabilities of any tested dataset seems an attractive solution for being exploited under semi-supervised methods.

The specific model of general Bayesian networks that has been used here is the HNB. The language of its structure allows to be built a hidden layer that describes the dependencies of each initial feature with the rest. The performance of the proposed algorithm (Self-HNB) as it concerns both the accuracy and the needed computational resources achieved great results compared with several other state of the art algorithms.

Some promising ideas that may perform well enough would be either a pre-process stage of feature selection for keeping the most informative features during the computation of any selected weighing function inside the HNB algorithm or the introduction of an adaptive procedure that could expand the depth of the hidden layers analog to the time tolerance that is needed, since a similar structure with double layer Bayesian classifier has been proven also adequate robust [32]. Finally, use of alternative approaches during the discretization [33] or the replacement of missing values may lead to better results under certain circumstances.

## APPENDIX A

A java software tool implementing the proposed algorithm and some basic run instructions can be found at: http://ml.math.upatras.gr/wp-content/uploads/2016/04/SelfHNB-Experiment.zip.

## References

[1] D. Yarowsky, "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods," *Proc. 33rd Annu. Meet. Assoc. Comput. Linguist.*, pp. 189–196, 1995.

[2] G. H. G. John and P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers," *Proc. Elev. Conf. Uncertain. Artif. Intell. Montr. Quebec, Canada*, vol. 1, pp. 338–345, 1995.

[3] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Mach. Learn.*, vol. 29, pp. 131–163, 1997.

[4] R. R. Bouckaert, "Voting massive collections of bayesian network classifiers for data streams," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4304 LNAI, pp. 243–252, 2006.

[5] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.

[6] R. Kohavi, "Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid," *Proc. Second Int. Conf. Knowl. Discov. Data Min.*, vol. 7, no. Utgo 1988, pp. 202–207, 1996.

[7] E. Frank, M. Hall, and B. Pfahringer, "Locally Weighted Naive Bayes," *Proc. 19th Conf. Uncertain. Artif. Intell.*, pp. 249–256, 2003.

[8] G. I. Webb, J. R. Boughton, and Z. Wang, "Not so naive Bayes: Aggregating one-dependence estimators," *Mach. Learn.*, vol. 58, no. 1, pp. 5–24, 2005.

[9] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data," *Mach. Learn.*, vol. 20, no. 3, pp. 197–243, 1995.

[10] D. M. Chickering, "Learning Bayesian networks is NP-complete," *Learn. from data*, pp. 121–130, 1996.

[11] L. Jiang, H. Zhang, and Z. Cai, "A novel bayes model: Hidden naive bayes," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 10, pp. 1361–1371, 2009.

[12] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory - COLT' 98*, 1998, pp. 92–100.

[13] F. Schwenker and E. Trentin, "Pattern classification and clustering: A review of partially supervised learning approaches," *Pattern Recognit. Lett.*, vol. 37, no. 1, pp. 4–14, 2014.

[14] I. Triguero, S. García, and F. Herrera, "Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study," *Knowl. Inf. Syst.*, vol. 42, no. 2, pp. 245–284, 2013.

[15] M. Li and Z. Zhou, "S ETRED : Self-training with Editing," *LNAI*, vol. 3518, pp. 611–621, 2005.

[16] Z. H. Zhou and M. Li, "Semi-supervised learning by disagreement," *Knowl. Inf. Syst.*, vol. 24, no. 3, pp. 415–439, 2010.

[17] J. Wang, S. Luo, and X. Zeng, "A random subspace method for co-training," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 195–200.

[18] Y. Yaslan and Z. Cataltepe, "Co-training with relevant random subspaces," *Neurocomputing*, vol. 73, no. 10–12, pp. 1652–1661, Jun. 2010.

[19] M. Hady and F. Schwenker, "Co-Training by Committee: A Generalized Framework for Semi-Supervised Learning with Committees," *Int J Softw. Informatics*, vol. 2, no. 2, pp. 95–124, 2008.

[20] Z.-H. Zhou and M. Li, "Tri-training: exploiting unlabeled data using three classifiers," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1529–1541, Nov. 2005.

[21] M. Li and Z. H. Zhou, "Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples," *IEEE Trans. Syst. Man, Cybern. Part ASystems Humans*, vol. 37, no. 6, pp. 1088–1098, 2007.

[22] C. Deng and M. Z. Guo, "A new co-training-style random forest for computer aided diagnosis," *J. Intell. Inf. Syst.*, vol. 36, no. 3, pp. 253–281, 2011.

[23] Y. Zhou and S. Goldman, "Democratic Co-Learning," *16th IEEE Int. Conf. Tools with Artif. Intell.*, pp. 594–602, 2004.

[24] A. Halder, S. Ghosh, and A. Ghosh, "Ant Based Semi-supervised Classification Proposed Methodology : Aggregation Pheromone Density Based Semi-Supervised Classification ( APSSC )," pp. 376–383, 2010.

[25] T. Huang, Y. Yu, G. Guo, and K. Li, "A classification algorithm based on local cluster centers with a few labeled training examples," *Knowledge-Based Syst.*, vol. 23, no. 6, pp. 563–571, 2010.

[26] U. Fayyad and K. Irani, "Multi-Interval Discretization of Continuous-Valued Attributes for Classification learning," *Proc. Natl. Acad. Sci. U. S. A.*, pp. 1022–1027, 1993.

[27] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Mult. Log. Soft Comput.*, vol. 17, no. 2–3, pp. 255–287, 2011.

[28] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-Supervised Self-Training of Object Detection Models," *Proc. Seventh IEEE Work. Appl. Comput. Vis.*, vol. 1, pp. 29–36, 2005.

[29] Z.Zhou and M.Li, "Tri-Training: Exploiting Unlabled Data Using Three Classifiers," *IEEE Trans.Data Eng.*, vol. 17, no. 11, pp. 1529–1541, 2005.

[30] C. Deng and M. Z. Guo, "Tri-training and data editing based semi-supervised clustering algorithm," *Micai 2006 Adv. Artif. Intell. Proc.*, vol. 4293, pp. 641–651, 2006.

[31] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci. (Ny).*, vol. 180, no. 10, pp. 2044–2064, May 2010.

[32] J. Sun, C. Wang, and S. Chen, "A Double Layer Bayesian Classifier," in *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, 2007, vol. 1, pp. 540–544.

[33] J. Lu, Y. Yang, and G. Webb, "Incremental discretization for naive-bayes classifier," *Adv. Data Min. Appl.*, pp. 223–238, 2006.