International Journal of Computer Applications (0975 - 8887)
Volume 135 - No.13, February 2016

An Evolutionary Bayesian Network Learning Algorithm using Feature Subset Selection for Bayesian Network Classifiers

Shefali K. Singhal
MBICT Women Engineering College
New Vallabh Vidyanagar
Anand, Gujarat, India

ABSTRACT

Classification is the process of constructing (learning) a model (classifier) to predict the class (labels) for given data. Bayesian belief network classifiers allow the representation of dependencies between subsets of attributes of a dataset. But learning an optimal Bayesian belief network for a Bayesian network classifier is a NP-hard problem. A number of heuristic-based algorithms have been proposed for supervised learning of bayesian belief network such as Tree-Augmented Naive Bayes (TAN), Hidden Naive Bayes (HNB). In the past decade, swarm intelligence (**SI**) based algorithms have been proposed for many optimization problems. Swarm intelligence based algorithms are characterized by the collective decentralized decision-making of several independent agents to search for the optimal solution in the solution search space.
In this paper, we propose a hybrid swarm intelligence based Bayesian network classifier combining Hunting Group search with Feature subset selections. The *Hunting Group Search - Feature Subset Selection (**HuGS-FSS**)* algorithm has been inspired by the behaviour of a pack of hunting animals such as wolves. The classification accuracy of the proposed HuGS-FSS algorithm is tested against other state of the art Bayesian network classifiers such as Naive Bayes, TAN, A2De, and HNB. Through comprehensive evaluation using 28 benchmark classification datasets from the UCI repository, we show that HuGS-FSS outperforms the other state of the art algorithms.

General Terms

Swarm Intelligence, Machine Learning

Keywords

Bayesian Network Classifier, Hunting Search Algorithm, Evolutionary Computation

1. INTRODUCTION

A classification algorithm takes a set of training data and creates a classifier. Once the classifier is learned from the training dataset, new data samples can be classified using the classifier. Let $< X_1, ..., X_n >$ be the attributes of the data. A data sample d can be described as $< x_1, ..., x_n >$ where x_i represents the values of attribute X_i. Learning a classifier from the training data can be viewed as learning the mapping $c = f(d)$, that can predict the class c of the data sample d.
A Bayesian belief network classifier comprises of two parts: (1) a Bayesian belief network graph *(directed acyclic graph)* that captures the dependencies between the attributes and (2) conditional probability tables.
Each node in the Bayesian belief network graph represents an attribute A_i of the dataset. An arc between 2 nodes indicates the attribute dependency. Figure 1 depicts a naive Bayes based Bayesian belief network graph for the iris dataset from the UCI machine learning datasets downloaded from the WEKA website [12]. In a naive Bayes network, every attribute is conditionally dependent only on the class attribute, as depicted in the above figure. Consequently, the class attribute is the parent of all the attributes of the iris dataset (i.e. sepallength, sepalwidth, petallength, petalwidth). Since all the attributes in a naive bayes bayesian network graph is conditionally independent of each other, there are no arcs between the attributes.
Each attribute in a Bayesian belief network classifier has an associated conditional probability table (CPT). The CPT for an attribute X gives the conditional probability distribution $P(X|Parents(X))$. The Bayesian belief network graph along with the CPT describe a joint probability over attributes X, given as:

$$p(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} p(x_i|\mathbf{Pa}(x_i)) \quad (1)$$

where $\mathbf{Pa}(x_i)$ are the parents of variable x_i in the Bayesian belief network graph.
A naive Bayes classifier makes the class conditional independence assumption. This implies that if the class of the data sample is known then the values of the attributes of the data sample are conditionally independent of each other. However, in reality, this assumption is often false. Bayesian belief network classifier enable the encoding of the dependency between the attributes. However learning an optimal Bayesian belief network classifier has been shown to an NP-hard problem [4, 3].

Many known state of the art algorithms for learning the optimal Bayesian Belief network for Bayesian network classifiers make limited relaxation of the class conditional independence assump-

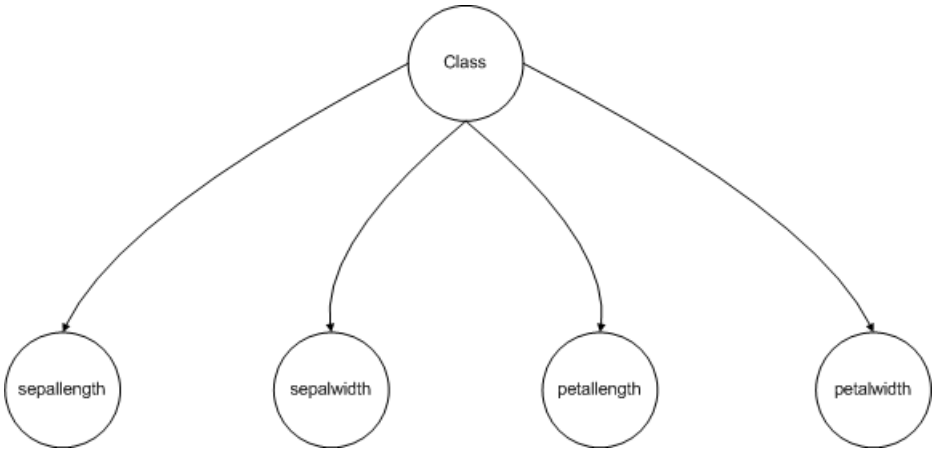

Fig. 1. A Naive Bayes Bayesian Network Graph for the iris dataset

tion. These algorithms relax the class independence assumption by adding limits (typically 1 or 2) on the number of parents permitted [9, 13]. While theoretically AnDe [25] enables learning of an ensemble of n-dependency classifiers, in practice as noted by the authors it expensive to learn n-dependency classifiers for $n \geq 3$. As learning the optimal Bayesian Network classifier has been shown to be NP-hard, and due to computational complexity of the current state of the art Bayesian Network learning algorithms for Bayesian network classifier, the existing algorithms are unable to accurately capture all the dependencies amongst all the attributes. The main motivation of this paper is to develop a computational simple model that can effectively search the solution space to find the best solution.

Hunting search algorithm [20, 23] is a bio-inspired metaheuristic algorithm for optimization problems and has been inspired by the behavior of animals such as wolves who hunt in packs and collaborate to catch a prey. Each member of the group positions itself based on the position of the other members in the group. In case the prey escapes, the hunting group repositions itself.

In our previous paper, we proposed a novel hunting group search based Bayesian network classifier [22]. Using preliminary experimental evaluation, we showed that our proposed classification algorithm outperformed other known Bayesian network classifiers. However, we observed that many known Bayesian network classifiers took a long time or large amounts of memory to classify high dimensional/large datasets. Therefore, in this paper, we extend our previous work and propose HuGS-FSS, a hybrid swarm intelligence based Bayesian network classifier combining Hunting Group search with Feature subset selection. The HuGS-FSS classifier learns the structure of the Bayesian network using the hunting search optimization technique. This paper is structured as follows: Section 2 gives a brief overview on the related work in Bayesian network classifiers, feature subset selection algorithms, swarm intelligence and hunting search algorithm. In section 3, we introduce the HuGS-FSS Classifier and explain the steps to learn the Bayesian Network structure using the Hunting Search meta-heuristic algorithm. Section 4 presents the experimental methodology and discusses the results. Finally our conclusions are stated in section 5 along with few remarks for future work.

2. RELATED WORK

2.1 Learning a Bayesian Network

Bayesian belief network classifier can be learned in two steps: (a) Learning the Bayesian belief network graph and (b) learning the CPTs for the graph learned in step (a). Learning the optimal Bayesian belief network for a Bayesian network classifier is an NP-hard problem [4, 3]. A number of heuristic-based algorithms have been proposed for learning of Bayesian belief network such as Naive-Bayes [10], K2 [5], TAN [9]. An overview of these methods is given by Cheng et al. in [2]. Learning the CPT is straightforward and can be done by estimating the likelihood of the value of the attribute given value of its parent attributes.

The class conditional independence assumption of Naive Bayes classifier is very strong and often does not hold for many real world datasets. Tree Augmented Naive Bayes [9] (TAN) classifier relaxes the class independence assumption of Naive Bayes classifier. It uses conditional mutual independence to add a single parent for each attribute.

Hidden Naive Bayes [13] is also a structure extension-based-algorithm which introduces a layer of hidden parents. A hidden parent for an attribute aggregates the influence of all other attributes by assigning higher weights to attributes with higher influence.

A2De [25] belongs to a general class of algorithms known as Averaged n-Dependence Estimators (AnDE). AnDe is an approach to probabilistic classification learning that learns by extrapolation from marginal to full-multivariate probability distributions. In A2De, the complete set of two-dependency classifiers are learned i.e. an ensemble of two-dependency classifiers is learned. The final prediction is made by averaging the predictions made by the ensemble. However, as noted in the referenced paper[25], learning AnDE for $n > 3$ is very expensive.

2.2 Feature Subset Selection

Feature subset selection refers to the process of selecting a subset of attributes from a given dataset for model learning. In the context of classification problems, the datasets can consist of a large number of features. May of these features are redundant or irrelevant. Eliminating these attributes can reduce the search space size and thereby making it easier and less expensive to build the classifier. As this is an active area of research, a large number of Feature

Subset Selection algorithms have been proposed. A complete review of all the literature in this area in beyond the scope of this paper. We refer the reader to [17] for more information on Feature Subset Selection. We compared the performance of HuGS-FSS with the following state of art Feature Subset Selection algorithm in our implementation: (a) wrapper based IWSSembeddedNB [1], (b) MultiObjectiveEvolutionarySearch [14], and (c) PSOSearch [18]. However, it must be noted that HuGS-FSS is not dependent on any particular feature subset selection algorithm.

2.3 Swarm Intelliegnce-Hunting Search Algorithm

In the past decade, many meta-heuristic algorithms based on swarm intelligence (SI) [26, 27] have been proposed to solve constraint optimization problems. SI algorithms incorporate a large number of simple homogeneous agents with simple behaviors who collaborate and share information to find the global optima. SI-inspired algorithms are characterized by their simplicity and the lack of a central management. Consequently, these algorithms are relatively fast, robust and effective in finding near optimal if not optimum solutions. For example, ants can locate food over long distances without the help of advanced communications and find the shortest path to the food. Even though each ant individually does not have a clue, collectively as a swarm, they are intelligent.
Hunting search algorithm [20, 23] is a bio-inspired meta-heuristic algorithm for optimization problems and has been inspired by the behavior of animals such as wolves who hunt in packs and collaborate to catch a prey. Each member of the group positions itself based on the position of the other members in the group. As a group they close in on the prey from different directions. In case the prey escapes, the hunting group repositions itself. This meta-heuristic-based algorithm has been used for constrained optimization problems such as transmission-constrained unit commitment [28], no-wait flow-shop scheduling [19] and other optimization problems [6, 24]. Hunting Search Algorithm has been also successfully applied to clustering [21], feature selection [7, 8] and high-dimensional function optimization problems [11]. However, to the best of our knowledge, HuGS is the first algorithm designed based on Hunting Search Algorithm to learn the Bayesian network structure for a bayesian network classifier.

In the last decade, there has been active research in the area of swarm intelligence inspired data ming algorithms. A comprehensive summary of the data mining/machine learning algorithms based on swarm intelligence has been compiled by Martens et al. [16]. As noted in this paper, most of the research in SI-inspired data mining algorithms has been concentrated on rule based classifiers, and clustering.
In the next section, we describe the HuGS-FSS, a hunting search inspired Bayesian network structure learning algorithm for a Bayesian network classifier. HuGS-FSS also incorporates a feature subset selection algorithm to remove irrelevant and redundant attributes in a classification dataset thereby making the classifier efficient and robust to handle large and high-dimensional datasets.

3. HUGS-FSS ALGORITHM

A naive Bayes classifier ignores the dependencies amongst the attributes. The Bayesian network classifiers overcomes this shortcoming by modeling the dependencies between the attributes in the form of a Bayesian network. The dependencies in the attributes are modeled by adding directed arcs between the attributes. The dependencies between attributes can be identified by a human expert

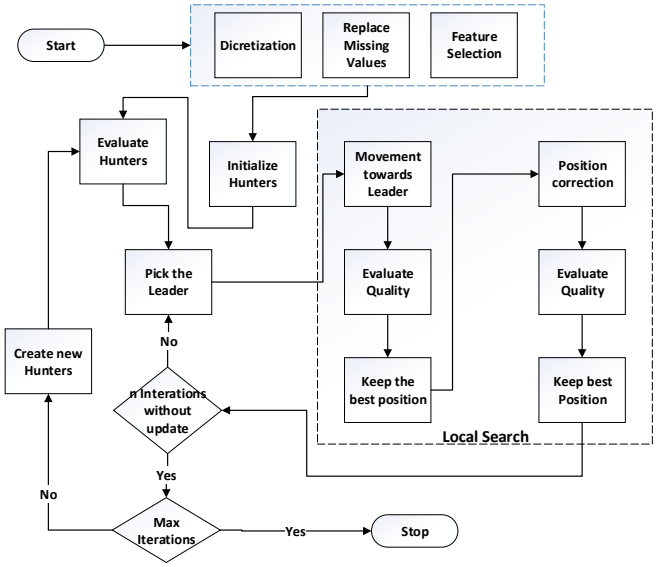

Fig. 2. Block diagram of the HuGS-FSS Bayesian Network Learning Algorithm

or in cases where expert knowledge of the problem domain is not known, a Bayesian network structure learning algorithm can learn the structure of Bayesian network directly from the training dataset. The proposed network learning algorithm, **Hu**nting **G**roup **S**earch-**F**eature **S**ubset **S**election (HuGS-FSS) Bayesian Network Classifier, is inspired from swarm intelligence based Hunting Search algorithm. The algorithm identifies the key attributes that contribute towards the classification accuracy and then builds Bayesian network classifier using the group hunting strategy of wolves.
Figure 2 depicts the block diagram of the HuGS-FSS algorithm. Initially, after discretization and replacement of missing values, an attribute selection algorithm selects the key attributes of the dataset that contribute to the classification accuracy. This is important for high-dimensional datasets as it reduces the complexity and size of the solution search space. Also by elimination of extraneous attributes that are irrelevant or redundant, the classification accuracy and efficiency can be increased as non-essential data is removed from training dataset.
The number of arcs e to be added and the maximum number of parents k of a node are the design parameters of HuGS-FSS Bayesian Network learning algorithm. As depicted in fig.2, feature subset selection is run on the training dataset and the extraneous attributes are removed from the training dataset. In our implementation, we compared the performance of HuGS-FSS using three state of the art feature subset selection algorithms: IWSSembeddedNB [1], MultiObjectiveEvolutionarySearch [14], and PSOSearch [18]. Our results noted in section 4 indicate that IWSSembeddedNB provides good performance when incorporated in HuGS-FSS. However, HuGS-FSS is not constrained to any particular feature subset selection algorithm.
Overall, the HuGS-FSS algorithm works as follows: After feature subset selection, the algorithm initializes a group of hunters. A hunter who is the best solution is selected as the leader. A neighborhood (local) search is performed by the hunters in the hunting group using two techniques: (1) moving towards the leader

Algorithm 1 HuGS-FSS Algorithm

Require: A dataset of training examples.

BEGIN
$AttributeSelection(dataset)$
$BestNetwork_{global} = \emptyset$
Initialize the parameters
for $i = 0$ **to** HG_Size **do**
  Initialize $hunter[i]$ {Create solution as per algo. 2}
end for
Identify the leader $hunter[i]$
$BestNetwork_{global} = hunter[i]$
repeat
  for $i = 0$ **to** HG_Size **do**
    Update $hunter[i]$ to $hunter[i]^{'}$ to **move towards the leader**
    if $Q(hunter[i]^{'}) > Q(hunter[i])$ **then**
      $hunter[i] = hunter[i]^{'}$
    end if
  end for
  for $i = 0$ **to** HG_Size **do**
    Update $hunter[i]$ to $hunter[i]^{'}$ **for position correction**
    if $Q(hunter[i]^{'}) > Q(hunter[i])$ **then**
      $hunter[i] = hunter[i]^{'}$
    end if
  end for
  Identify the leader $hunter[i]$
  $BestNetwork_{global} = hunter[i]$
  Reorganize the hunters
until max_iterations
return $BestNetwork_{global}$
END

and (2) position correction. Once the local search is completed, the algorithm performs global search by re-initializing the hunters. Re-initialization of hunters ensures that the algorithm does not get stuck in a local optima. Local search and global search are carried out till convergence or maximum number of iterations are reached. The details of HuGS-FSS Bayesian network learning algorithm are outlined in Algorithm 1.

Algorithm 2 Create Hunters Procedure

BEGIN CreateHunter()
$BNC \leftarrow \phi$
$k = hunter.selectMaxParents()$
$e = hunter.selectNumEdges()$
$i = 0$
while $i < e$ **do**
  $\{i \leftarrow j\} = hunter.addEdge()$
  $BNC \leftarrow BNC \cup \{i \leftarrow j\}$
  if $BNC.findCycle ==$ **true or** $BNC.exceedParentLimit ==$ **true then**
    $BNC \leftarrow BNC - \{i \leftarrow j\}$
  end if
end while
$BNC.LearnCPTs()$
return BNC

Once the Bayesian network structures corresponding to each hunter have been created, HuGS-FSS algorithm identifies the best solution, which is marked as the leader. Since HuGS-FSS algorithm is designed for solving classification problems, the leader is the solution (hunter) which has the best predictive performance i.e. solution that most accurately classifies maximum number of tuples. In this paper, we use $accuracy$ as the quality measure to test the quality of the solution generated by HuGS-FSS algorithm. The accuracy of a HuGS-FSS classifier on a given dataset is measured as the percentage of tuples that are correctly classified, and is given as follows:

$$Accuracy = \frac{TP + TN}{TC} \tag{2}$$

where, **True Positives (TP)**: Positive tuples correctly labeled by the classifier.
True Negatives (TN): Negative values correctly labeled by the classifier.
Total Count (TC): Total sample count in the dataset.

Step 3: Movement towards the leader: Once the leader has been identified (using equation 2), local search is carried out to improve the quality of the solution using maximum movement towards the leader. Maximum movement towards the leader mimics the wolves who surround a prey and try to capture the prey by moving in on the prey from different directions. All the hunters (candidate solutions) in the HuGS-FSS update their positions by moving towards the leader as per equation 3:

$$x_i^{'} = x_i + rand \times MML \times (x_i^l - x_i) \tag{3}$$

where x_i^l is the position of the leader and maximum movement toward the leader **(MML)** depends on the problem under consideration. We use a value of 0.5 in our implementation. In section 4, we test 3 datasets with different values of MML. As shown in table 5 and fig. 3, we note that the classifier is most accurate for $MML = 0.5$. After each hunter moves towards the leader, it's quality is evaluated as per equation 2. If accuracy of the solution is better, the hunter keeps the new position otherwise it moves back to it original position.
Step 4: Position Correction: The hunters perform $position\ correction$ based on the hunters current positions and the hunting group consideration rate **HGCR**. Based on the HGCR, the hunter either keeps the position from the HG or moves a new location based on the distance radius Ra. The new hunter position is set as shown in Eq. 4

$$x_i^j \longleftarrow \begin{cases} x_i^j \in \{HG\}, & p(HGCR) \\ x_i^j = x_i^j \pm Ra, & p(1 - HGCR) \end{cases} \tag{4}$$

where, x_i^j is the value of the i^{th} decision variable of the j hunter in the hunting group.
The parameter HGCR denotes the probability of picking the existing position from the HG. HGCR is set at 0.6 in our implementation. Ra denoted the fixed radius distance, that the hunter moves and is problem dependent parameter. In this step, the hunter searches in its own local neighborhood to look for a better solution. If a better solution is found, the hunter moves to the new location; otherwise it keeps its own position. In our implementation, Ra takes the value 2.
Step 5: Reorganization: After local search is completed using movement toward the leader (step 3) and position correction (step 4), the HuGS-FSS algorithm re-initializes the hunters. In this step, the hunters are reorganized to a new starting position. This is carried out to prevent the hunting group from being trapped in a local optimum. In our implementation, if there is no improvement in the

Table 1. Details of the datasets used for evaluation.

Sr. No.	Dataset	Number of data samples	Number of attributes	Number of classes
1	adult	32560	15	2
2	anneal	898	39	6
3	audiology	226	70	24
4	autos	205	26	7
5	balance-scale	625	5	3
6	breast-cancer	286	10	2
7	car	1728	7	4
8	cmc	1473	10	3
9	colic	368	23	2
10	credit-a	690	16	2
11	diabetes	768	9	2
12	glass	214	10	7
13	heart-c	303	14	5
14	heart-statlog	270	14	2
15	hepatitis	155	20	2
16	hypothyroid	3772	30	4
17	ionosphere	351	35	2
18	iris	150	5	3
19	kr-vs-kp	3196	37	2
20	labor	57	17	2
21	lymph	148	19	4
22	mushroom	8124	23	2
23	nursery	12960	9	5
24	primary-tumor	339	18	22
25	sick	3772	30	2
26	sonar	208	61	2
27	vehicle	846	19	4
28	vote	435	17	2

quality of the solution after 10 iterations of movement towards the leader and position correction, we re-initialize the hunting group as described in step 2. Even though the number of iterations of local search is fixed at 10 for our implementation, the problem search space and training time can be used as a guide to select the number of iterations of local search. However, in our experiments, 10 iterations of local search provide good accuracy for the trained classifier.

Step 6: Termination: Steps 3 to 5 are repeated until maximum number of iterations are completed. In our implementation, steps 3 to 5 are repeated for a maximum of 100 iterations. The selection of maximum number of iterations is problem specific. In order to conduct a more extensive search of the solution space for high dimensional datasets, the maximum number of iterations can be increased. The number of iterations provides a trade-off between the extend of search of solution space and computational resources required. For our experimental datasets, we found that 100 iterations provided acceptable solutions. Once maximum iterations are completed, the HuGS-FSS algorithm returns the best HuGS-FSS Bayesian network classifier learned by the algorithm.

In the next section, we evaluate the performance of the HuGS-FSS Bayesian network classifier introduced in this section. We also present the results of the detailed experimental evaluation and a discussion on selection of values of the various algorithmic parameters used in HuGS-FSS algorithm.

4. EXPERIMENTS AND RESULTS

We used the Weka machine learning library [12] to implement the HuGS-FSS classifier. The HuGS-FSS classifier is evaluated using 28 benchmark classification datasets selected from UCI (University of California, Irvine) repository [15]. These datasets were downloaded from the Weka website. The characteristics of these datasets have been summarized in table 1. These datasets represent a wide array of domains and varied characteristics. The datasets also include high-dimensional datasets (upto 70 attributes) as well as large datasets (having more than 30,000 data samples). As the classification algorithm requires a discretized dataset having no missing values, the following preprocessing steps were applied:

(1) Datasets having continuous attributes were discretized using the $Discretize$ filter implemented in Weka. It uses a 10-bin unsupervised discretization.

(2) Datasets with missing values were preprocessed using the $ReplaceMissingValues$ unsupervised filter in Weka. This filter replaces missing values with the modes and means from the training data.

The predictive accuracy of the HuGS-FSS classifier was compared against other widely used classifiers such as ZeroR (baseline), Naive Bayes [10], TAN [9], A2De [25], and HNB [13] search. ZeroR classifier chooses the most common class in the training set as the predicted class. ZeroR is used to provide a baseline to test the effectiveness of the classifiers.

Table 2 shows the predictive accuracy obtained by the classifiers. The performance of the classifier was evaluated by running a 10-fold cross-validation (CV) on the datasets listed in the table. A 10-fold cross validation divides the dataset into 10 partitions. 9 partitions are used to train the classifier whereas the remaining 1 partition is used to test the performance of the classifier learned. The 10-fold CV is run 10 times where in each iteration a different partition is used as the test set. Each classifier is evaluated via 10 runs of 10-fold cross validation procedure.

Table 2. Experimental Results: Predictive Accuracy and Standard Deviation

No.	DataSet	HuGS-FSS	ZeroR	NB	HNB	A2De	TAN
1	adult	83.32 ± 4.08	75.67 ± 0.0	80.36 ± 0.94	80.06 ± 2.35	**84.73 ± 3.80**	77.03 ± 1.52
2	anneal	**93.81 ± 2.36**	76.16 ± 0.0	93.34 ± 1.03	93.34 ± 1.03	93.00 ± 1.68	93.45 ±3.15
3	audiology	**81.10 ± 1.05**	25.22 ± 0.0	74.64 ±1.15	76.19 ± 1.81	78.05 ± 2.01	77.03 ± 1.52
4	autos	**84.92 ± 2.28**	32.68 ± 0.0	75.17 ± 2.02	82.68 ± 2.36	84.68 ± 1.77	82.04 ± 3.32
5	balance-scale	**91.50 ± 1.52**	45.76 ± 0.0	89.53 ± 4.08	89.53 ± 4.08	81.96 ± 6.32	85.40 ± 8.62
6	breast-cancer	**75.06 ± 1.33**	70.27 ± 0.0	74.37 ± 1.56	72.67 ± 2.04	71.99 ± 2.13	70.06 ± 3.06
7	car	93.14 ± 5.27	70.02 ± 0.0	85.82 ± 4.59	92.81 ± 3.33	**94.14 ± 6.66**	94.13 ± 3.05
8	cmc	**53.90 ± 6.25**	42.7 ± 0.0	53.57 ± 5.64	50.36 ± 7.41	52.21 ± 6.57	52.44 ± 9.75
9	colic	**86.00 ± 2.27**	63.04 ± 0.0	84.34 ± 1.26	83.94 ± 2.55	84.23 ± 2.35	83.91 ± 2.34
10	credit-a	**87.11 ± 1.96**	55.5 ± 0.0	86.13 ± 1.63	85.14 ± 2.95	85.75 ± 1.76	86.20 ± 2.85
11	diabetes	**77.85 ±3.07**	65.1 ± 0.0	77.77 ± 2.21	73.16 ± 4.04	71.32 ± 6.39	73.21 ± 6.84
12	glass	**64.25 ± 2.50**	35.51 ± 0.0	59.25 ± 3.42	58.83 ± 2.42	60.18 ± 3.04	58.27 ± 4.05
13	heart-c	**85.31 ± 1.26**	54.45 ± 0.0	84.98 ± 1.26	81.45 ± 4.54	81.02 ± 2.46	84.91 ± 1.63
14	heart-statlog	86.11 ± 1.08	55.55 ± 0.0	**86.18 ± 0.82**	82.70 ± 1.05	83.03 ± 2.82	83.03 ± 1.31
15	hepatitis	**89.03 ± 1.15**	79.35 ± 0.0	85.80 ± 1.33	82.96 ± 1.95	84.51 ± 2.0	88.58 ± 1.33
16	hypothyroid	**94.13 ± 1.03**	92.04 ± 0.0	92.89 ± 0.51	92.98 ± 3.24	93.18 ± 1.26	93.34 ± 1.47
17	ionosphere	**92.67 ± 1.56**	64.1 ± 0.0	92.25 ± 1.03	91.59 ± 2.12	91.50 ± 1.81	90.54 ± 3.15
18	iris	**97.19 ± 0.63**	33.33 ± 0.0	97.0 ± 1.08	88.4 ± 0.84	96.66 ± 0	**97.19 ± 0.63**
19	kr-vs-kp	**94.67 ± 0.31**	54.25 ± 0.0	93.24 ± 0.87	94.66 ± 0.63	94.48 ± 0.63	94.65 ± 0.69
20	labor	**94.73 ± 0.0**	64.91 ± 0.0	91.22 ± 0.0	87.01 ± 1.26	90.70 ± 0.82	90.52 ± 0.51
21	lymph	**87.43 ± 1.07**	54.72 ± 0.0	86.82 ± 1.26	83.31 ± 1.05	85.94 ± 1.54	86.89 ± 2.31
22	mushroom	**99.50 ± 0.0**	50.92 ± 0.0	96.61 ± 1.91	99.11 ± 0.84	**99.50 ± 0.0**	99.28 ± 1.17
23	nursery	**90.37 ± 3.74**	34.29 ± 0.0	88.65 ± 3.14	**90.37 ± 6.23**	91.27 ± 3.78	90.29 ± 3.10
24	primary-tumor	**48.64 ± 2.37**	24.77 ± 0.0	47.49 ± 2.94	47.84 ± 1.75	48.61 ± 2.09	45.89 ± 3.94
25	sick	**94.90 ± 0.42**	92.3 ± 0.0	94.66 ± 0.31	94.53 ± 0.51	94.48 ± 0.42	94.32 ± 1.13
26	sonar	78.02 ± 1.70	53.36 ± 0.0	**78.07 ± 1.64**	72.35 ± 2.27	71.39 ± 2.36	70.62 ± 4.48
27	vehicle	**72.64 ± 6.00**	25.47 ± 0.0	63.23 ± 4.10	72.21 ± 4.62	71.60 ± 7.06	68.81 ± 6.86
28	vote	96.32 ± 0.0	61.37 ± 0.0	94.57 ± 1.07	95.88 ± 0.73	95.44 ± 1.03	**95.97 ± 1.26**

Table 3. Win/Tie/Loss Summary Table

	ZeroR	**NB**	**HNB**	**A2De**	**TAN**
HuGS-FSS	28/0/0	26/0/2	27/1/0	25/1/2	27/1/0
ZeroR		0/0/28	0/0/28	0/0/28	0/0/28
NB			16/2/10	16/0/12	14/0/14
HNB				11/0/17	13/0/15
A2De					16/1/11

As noted in table 2, overall the classification accuracy of HuGS-FSS classifier is best amongst the classifiers compared in this paper. HuGS-FSS classifier outperforms all the other classifiers in the experiment for 23 out of the 28 datasets. For the remaining five datasets, its classification accuracy is close to the accuracy of the best classifier. Table 3 summarizes the performance of the algorithms as compared to each other.
The experimental results can be summarized as below:

(1) HuGS-FSS outperforms ZeroR in classification accuracy for all the datasets.

(2) HuGS-FSS outperforms Naive Bayes in classification accuracy for 26 datasets and its accuracy is close to Naive Bayes accuracy for 2 dataset.

(3) HuGS-FSS outperforms HNB in classification accuracy for 27 datasets and ties for 1 dataset.

(4) HuGS-FSS outperforms A2De in classification accuracy for 25 datasets. HuGS-FSS ties A2De for 1 dataset and underperforms for 2 datasets.

(5) HuGS-FSS outperforms TAN in classification accuracy for 27 datasets and ties for 1 dataset.

In the following subsections, we discuss how to tune values of the various HuGS-FSS algorithm parameters for specific applications.

4.1 Parameter Settings

4.1.1 Feature Subset Selection Algorithm. We compared the performance of various feature subset selection which gave the best accuracy when incorporated in HuGS-FSS. We compared the performance of three state of art feature subset selection algorithms:
(1) IWSSembeddedNB: Incremental Wrapper Subset Selection with embedded NB classifier
(2) MultiObjectiveEvolutionarySearch: An Multi-objective Evolutionary Algorithm (MOEA)
(3) PSOSearch: An implementation of the Particle Swarm Optimization (PSO) algorithm.

Table 4 shows the performance of HuGS-FSS classifier incorporating the above mentioned feature subset selection algorithm. From the results noted in the table 4, we found that IWSSembeddedNB provided the best performance for 9 out of 14 datasets. While the selection of the appropriate feature subset selection algorithm can be tuned for the characteristics of the particular dataset, in general, we found that IWSSembeddedNB provides good performance when incorporated into HuGS-FSS algorithm.

4.1.2 Hunting Group Size. The hunting group size parameter determines the number of hunters (candidate solutions) generated by the algorithm in each iteration. In our experimental setup we use hunting group size of 10 hunters. For most classification problems, a hunting group of size 10 gives good accuracy. However, for problem with very large search space, it will be useful to increase the

Table 4. Comparison of performance of Hugs-FSS using different feature subset selection algorithms

No	Dataset	IWSSembNB	MOEA	PSO
1	audiology	**81.10 ± 1.05**	77.30 ± 2.21	76.54 ± 2.26
2	autos	84.92 ± 2.28	**86.82 ± 3.21**	85.65 ± 1.83
3	breast-cancer	**75.06 ± 1.33**	74.79 ± 1.10	74.79 ± 1.52
4	colic	**86.00 ± 2.27**	85.38 ± 1.13	83.31 ± 2.17
5	diabetes	**77.85 ± 3.07**	76.17 ± 2.58	76.17 ± 2.58
6	heart-c	85.31 ± 1.26	**85.61 ± 2.63**	85.18 ± 0.73
7	hepatitis	**89.03 ± 1.15**	86.83 ± 0.96	87.09 ± 0.94
8	iris	**97.19 ± 0.63**	**97.19 ± 0.63**	**97.19 ± 0.63**
9	labor	**94.73 ± 0.0**	94.38 ± 0.63	94.03 ± 0.69
10	mushroom	99.50 ± 0.0	**100 ± 0.0**	99.01 ± 0.0
11	nursery	**90.37 ± 3.74**	70.79 ± 0.0	70.79 ± 0.0
12	sonar	78.02 ± 170	**78.70 ± 3.12**	78.41 ± 2.02
13	vehicle	**72.64 ± 6.00**	68.41 ± 7.88	66.33 ± 4.36
14	waveform	83.54 ± 6.26	83.13 ± 5.33	**84.48 ± 5.18**

hunting group size to enable faster convergence and better accuracy.

Table 5. Performance of HuGS-FSS for various MML values

MML	iris	cmc	hepatitis
0.1	96.13±0.78	52.6±7.67	87.29±1.15
0.2	95.86 ± 1.13	52.69 ± 7.39	87.29 ± 1.15
0.3	95.86 ± 1.13	52.89 ± 5.99	86.45 ± 1.24
0.4	95.86 ± 1.31	52.76 ± 7.14	87.03 ± 1.37
0.5	97.19 ± 0.63	53.89 ± 5.36	89.03 ± 1.15
0.6	95.86 ± 1.31	52.76 ± 7.14	87.61 ± 2.74
0.7	95.86 ± 1.31	53.38 ± 8.6	86.32 ± 1.31
0.8	95.86 ± 1.31	52.62 ± 7.67	87.61 ± 2.74
0.9	95.86 ± 1.31	52.76 ± 7.14	87.29 ± 1.15
1	95.86 ± 1.31	52.76 ± 7.14	87.03 ± 1.37

4.1.3 Maximum movement towards the leader (MML).. MML determines the distance the hunter moves towards the leader in one step. We ran experiments for different values of MML for iris, cmc and hepatitis datasets. Table 5 shows the results of the experimental evaluation. We found that the accuracy of the HuGS-FSS is the best when the value of MML is set at 0.5. This can be clearly observed from fig. 3.

4.1.4 HGCR and Ra. The values of $HGCR$ and Ra are dependent on the application and on the size of the search space (i.e. the number of attributes in our dataset). These parameters are used in position correction step of local search. During position correction step, we are search the solution space in close proximity to the current solution. The HGCR determines the likelihood of decision variable keeping its current position. Since we performing local search of the solution space, we would like to add/remove only a few arcs and evaluate the resultant Bayesian Network Classifier. Consequently we chose a value of 2 for Ra i.e. 2 arcs would be added or removed from the current Bayesian Network. If we pick a very large value of Ra, the resultant HuGS-FSS Bayesian Network generated will not be similar to the current solution and will not effectively carry out local search around the current solution. If we have a very large search space, then we can adjust the value to Ra to be a larger number.

5. CONCLUSIONS

In this paper, we proposed a novel swarm intelligence inspired Hunting Group Search - Feature Subset Selection (HuGS-FSS) algorithm to learn the Bayesian Network structure for a Bayesian belief network classification algorithm. HuGS-FSS has been inspired by the behavior of animals such as wolves who hunt in a group and collaborate to catch a prey. Our experimental results show that HuGS-FSS outperforms the other state of the art Bayesian network learning algorithms for Bayesian network classifiers. Also, due to the algorithmic and implementation simplicity of the algorithm, HuGS-FSS can be applied for real-world problems. As the algorithm consists of multiple simple computational agents that collaborate to search for the optimal solution, this algorithm can be easily parallelized.

We also systematically studied how the various algorithmic parameters affect the performance of the algorithm. We also provide general guidelines for tuning the values of these parameters. Through systematic evaluation on numerous datasets of varying characteristics i.e. large number of attributes (upto 70 attributes) or large datasets (having more than 30000 data samples), we have shown the effectiveness of HuGS-FSS classification algorithm for a wide range of classification problems. The resultant HuGS-FSS classifier is effective and easy to implement in most cases.

In this paper, we have shown the effectiveness of using a swarm intelligence inspired algorithm to design Bayesian network classifiers. More work is required to design and evaluate the effectiveness of using other swarm intelligence algorithms for developing novel and effective Bayesian network classifiers. Also, while HuGS-FSS has been proven effective for static dataset, more work needs to be done to adapt and evaluate HuGS-FSS for streaming datasets.

6. REFERENCES

[1] Pablo Bermejo, José A Gámez, and José M Puerta. Speeding up incremental wrapper feature subset selection with naive bayes classifier. *Knowledge-Based Systems*, 55:140–147, 2014.

[2] Jie Cheng and Russell Greiner. Comparing bayesian network classifiers. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 101–108. Morgan Kaufmann Publishers Inc., 1999.

[3] David Maxwell Chickering. Learning bayesian networks is np-complete. In *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.

[4] Gregory F Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2):393–405, 1990.

[5] Gregory F Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347, 1992.

[6] Erkan Doğan and Ferhat Erdal. Hunting search algorithm based design optimization of steel cellular beams. In *Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion*, pages 1729–1730. ACM, 2013.

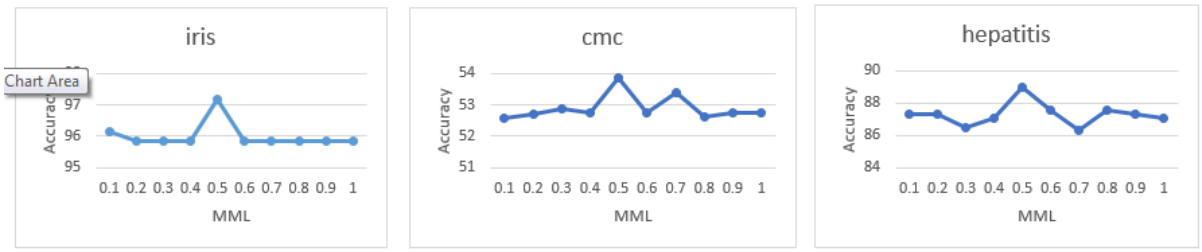

Fig. 3. HuGS-FSS Accuracy for various MML values

[7] Simon Fong, Kun Lan, and Raymond Wong. Classifying human voices by using hybrid sfx time-series preprocessing and ensemble feature selection. *BioMed research international*, 2013, 2013.

[8] Simon Fong, Xin-She Yang, and Suash Deb. Swarm search for feature selection in classification. In *Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on*, pages 902–909. IEEE, 2013.

[9] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers, 1997.

[10] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.

[11] Wu Husheng and Zhang Fengming. A uncultivated wolf pack algorithm for high-dimensional functions and its application in parameters optimization of pid controller. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 1477–1482, July 2014.

[12] Mark Hall Ian Witten, Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*.

[13] Liangxioa Jiang, Harry Zhang, and Zhihua Cai. A novel bayes model: Hidden naive bayes. *IEEE Transactions on Knowledge and Data Engineering*, 21(10), 2009.

[14] Fernando Jiménez, Gracia Sánchez, and José M Juárez. Multi-objective evolutionary algorithms for fuzzy classification in survival prediction. *Artificial intelligence in medicine*, 60(3):197–219, 2014.

[15] M. Lichman. UCI machine learning repository, 2013.

[16] David Martens, Bart Baesens, and Tom Fawcett. Editorial survey: swarm intelligence for data mining. *Machine Learning*, 82(1):1–42, 2011.

[17] Luis Carlos Molina, Lluís Belanche, and Àngela Nebot. Feature selection algorithms: A survey and experimental evaluation. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 306–313. IEEE, 2002.

[18] Alberto Moraglio, Cecilia Di Chio, Julian Togelius, and Riccardo Poli. Geometric particle swarm optimization. *Journal of Artificial Evolution and Applications*, 2008:11, 2008.

[19] B Naderi, Majid Khalili, and Alireza Arshadi Khamseh. Mathematical models and a hunting search algorithm for the no-wait flowshop scheduling with parallel machines. *International Journal of Production Research*, 52(9):2667–2681, 2014.

[20] R. Oftadeh, M.J. Mahjoob, and M. Shariatpanahi. A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search. *Computers Mathematics with Applications*, 60(7):2087 – 2098, 2010.

[21] Tang Rui, S. Fong, Xin-She Yang, and S. Deb. Nature-inspired clustering algorithms for web intelligence data. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on*, volume 3, pages 147–153, Dec 2012.

[22] Shefali K Singhal. Bio-inspired bayesian network learning algorithm for bayesian network classifiers. *International Journal of Advance Engineering and Research Development*, 2(3), 2015.

[23] Rui Tang, S. Fong, Xin-She Yang, and S. Deb. Wolf search algorithm with ephemeral memory. In *Digital Information Management (ICDIM), 2012 Seventh International Conference on*, pages 165–172, Aug 2012.

[24] Korakoch Waiyakan and Pongchanun Luangpaiboon. Heat treatment process optimization using hunting search and ant sense on path of steepest ascent. *Applied Mechanics and Materials*, 217:1475–1478, 2012.

[25] Geoffrey I. Webb, Janice R. Boughton, Fei Zheng, Kai Ming Ting, and Houssam Salem. Learning by extrapolation from marginal to full-multivariate probability distributions: Decreasingly naive bayesian classification. *Machine Learning*, 2012.

[26] Xin She Yang. *Nature-Inspired Optimization Algorithms*. Elsevier.

[27] Xin She Yang, Zhihua Cui, Renbin Xiao, Amir Hossein Gandomi, and Mehmet Karamanoglu. *Swarm Intelligence and Bio-Inspired Computation*. Elsevier, 2013.

[28] Kazem Zare and Sayyed Mohammad Hashemi. A solution to transmission-constrained unit commitment using hunting search algorithm. In *Environment and Electrical Engineering (EEEIC), 2012 11th International Conference on*, pages 941–946. IEEE, 2012.