Applied Soft Computing 11 (2011) 477–488

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Using GMDH-based networks for improved spam detection and email feature analysis

El-Sayed M. El-Alfy*, Radwan E. Abdel-Aal

College of Computer Sciences and Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

A R T I C L E I N F O

Article history:
Received 24 March 2008
Received in revised form 22 July 2009
Accepted 4 December 2009
Available online 16 December 2009

Keywords:
Group method of data handling
GMDH-based networks
Soft computing
Neural networks
Bayesian classifiers
Spam detection
Spam filtering
Feature selection
Network ensembles
Network committees

A B S T R A C T

Unsolicited or spam email has recently become a major threat that can negatively impact the usability of electronic mail. Spam substantially wastes time and money for business users and network administrators, consumes network bandwidth and storage space, and slows down email servers. In addition, it provides a medium for distributing harmful code and/or offensive content. In this paper, we explore the application of the GMDH (Group Method of Data Handling) based inductive learning approach in detecting spam messages by automatically identifying content features that effectively distinguish spam from legitimate emails. We study the performance for various network model complexities using spambase, a publicly available benchmark dataset. Results reveal that classification accuracies of 91.7% can be achieved using only 10 out of the available 57 attributes, selected through abductive learning as the most effective feature subset (i.e. 82.5% data reduction). We also show how to improve classification performance using abductive network ensembles (committees) trained on different subsets of the training data. Comparison with other techniques such as neural networks and naïve Bayesian classifiers shows that the GMDH-based learning approach can provide better spam detection accuracy with false-positive rates as low as 4.3% and yet requires shorter training time.

1. Introduction

Electronic mail has become a dominant means for personal and business communication. However, due to its low cost and ease of deployment, individuals and companies can misuse it for the rapid distribution of unsolicited (spam) messages to a large number of users. Spam emails are used for marketing of products and services, spreading rumors and other fraudulent advertisements, and distributing offensive content. Spam does not only annoy email users, but also exhausts network resources (bandwidth and storage), slows down email servers and provides a medium for phishing attacks and distributing harmful malicious codes. In addition, it increases the cost of computing and network infrastructure upgrades, and negatively affects the quality of service provided to other non-spam email traffic and legitimate Internet applications [20].

With the rapid increase in the volume and serious impact of spam, providing viable spam fighters has recently attracted considerable attention. In addition to regulations and legislations, several technical solutions including commercial and open-source products have been deployed. Installing anti-spam filters at the network gateway is the most commonly used mechanism to block or quarantine spam messages as early as possible. Spam filtering methods fall into two broad categories: non-machine-learning based and machine-learning based. Most of the early anti-spam tools belonged to the first category; for example using a blacklist of known spammers, a whitelist of safe senders, or a list of keywords such as “Get Rich” either in the subject line or the message content [27,43]. However, these list-based methods can be easily defeated by spammers through avoiding/misspelling certain words, forging the content, changing or spoofing the sender’s address or domain to bypass the filter. These methods also require periodic manual updates and the likelihood of filtering out a legitimate message as spam is high, which can be more harmful than not filtering at all. It is estimated that over five million working hours a year are wasted on checking that legitimate email messages were not mistakenly quarantined [8].

The similarity between spam filtering and text categorization problems and the success of machine-learning techniques in solving such problems have inspired several researchers to investigate their applicability in filtering spam. Unlike traditional approaches, machine-learning techniques analyze the message content and classify it accordingly. Therefore, they can be more effective in dealing with spammers’ tactics. Recently, various machine-learning

* Corresponding author at: P.O. Box 371, KFUPM Dhahran 31261, Saudi Arabia. Tel.: +966 3 860 1930; fax: +966 3 860 2174.
E-mail addresses: alfy@kfupm.edu.sa (E.-S.M. El-Alfy), radwan@kfupm.edu.sa (R.E. Abdel-Aal).

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.12.007

techniques have been used for spam filtering, including support vector machines [13,22,46], memory-based learning [6,35,41], Ripper rule-based learning [12,33], boosting decision trees [10], rough sets [49], neural networks [11,48], Bayesian classifiers [5,6,23,33,34,36,37], and fuzzy logic [17,32]. Among these methods, Bayesian classifiers have been widely applied as one of the most effective spam filters [19,23]. Recent reviews and taxonomy of current and potential solutions for the spam problem are presented in [9,18,21,25,28,44]. Although empirical studies on existing spam filters have reported promising results in terms of detection accuracy, the number of false positives (legitimate messages classified as spam) is still unacceptably high. Because of the continuous sophistication of spamming software tools, no single technique could completely solve this problem. This has led some researchers to propose the application of multiple techniques in several layers, as each may excel in some classification aspect [29,50].

In this study, we introduce the use of GMDH (group method of data handling) based networks for detecting spam messages and identifying the most effective content features for classifying emails. GMDH-based networks have emerged as a powerful supervised inductive learning approach in artificial intelligence. They were successfully used for data mining, knowledge discovery, system modeling, long-term and short-term forecasting, optimization and pattern recognition [1,3,14,15,31]. We report on experiments conducted to evaluate various network models for classifying emails using a publicly available dataset. We also show how performance was improved through using a network ensemble trained on different subsets of the training data. Results are compared with three other techniques; namely probabilistic neural network (PNN) with genetic training, multilayer perceptron (MLP), and naïve Bayesian (NB) classifiers.

The rest of this paper is organized as follows: Section 2 describes the GMDH-based learning methods used for building optimal network models that capture the input–output relationships in the training set. Section 3 defines the performance measures used for evaluating and comparing the effectiveness of the proposed methods. Section 4 describes the spambase dataset used. Section 5 describes the various experiments conducted using abductive machine learning. Section 6 compares the performance of the proposed methods with three other existing techniques. Finally, Section 7 concludes the paper and proposes directions for future work.

2. GMDH-based networks

2.1. AIM abductive networks

Abductory inductive mechanism (AIM) [2] is a supervised machine-learning approach for inductively synthesizing compact abductive models from a database of training examples. The models built take the form of layered feed-forward abductive networks of polynomial functional elements (nodes) [2]; see Fig. 1. The network size, element types, connectivity, and coefficients for the optimum model are automatically determined using well-proven optimization criteria, thus reducing the need for user intervention as compared to neural networks. In addition, the automation of model synthesis lessens the burden on the analyst and safeguards the model generated against influence by human biases and misjudgments. This simplifies model development and considerably reduces the learning/development time and effort. AIM builds networks of various types of functional elements. Elements in the first layer operate on various combinations of the independent input variables (x's) and the element in the final layer produces the predicted output for the dependent variable y. An input layer of normalizers convert the input variables into an internal representation as Z scores with zero mean and unity variance, and an output

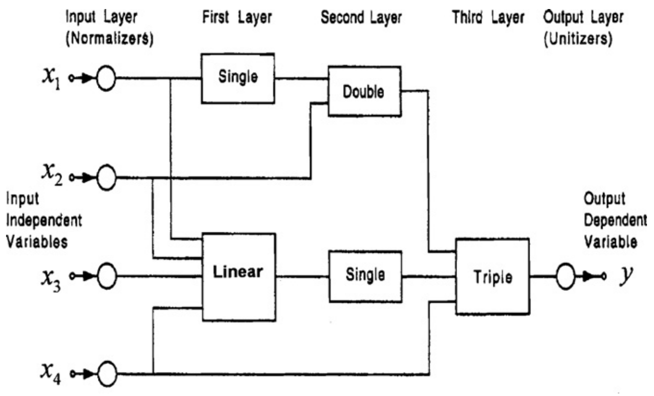


Fig. 1. A typical AIM abductive network model showing various types of functional elements.

unitizer unit restores the results to the original problem space. AIM supports the following main functional elements:

(i) A linear element which consists of a constant plus the linear weighted sum of all outputs of the previous layer, *i.e.*

$$\text{“Linear” Output} = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_qx_q \quad (1)$$

where $x_1, x_2, \ldots, x_q$ are the element inputs and $w_0, w_1, \ldots, w_q$ are the element weights.

(ii) Single, double, and triple elements which implement a third-degree polynomial expression for one, two, and three inputs respectively; *e.g.*,

$$\text{“Double” Output} = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2 + w_6x_1^3 + w_7x_2^3 \quad (2)$$

As a GMDH-type approach, AIM algorithm attempts to build an effective high-degree polynomial model in an iterative manner without overfitting the training dataset. It starts by using simple (myopic) regression relationships to derive more accurate representations in the next iteration. To prevent exponential growth and limit model complexity, the algorithm selects only relationships having good predicting powers within each phase. Iteration is stopped when the new generation regression equations start to have poorer prediction performance than that of the previous generation, at which point the model starts to become overspecialized and therefore unlikely to perform well with new data. The algorithm has three main components (representation, selection, and stopping) and applies abduction heuristics for making decisions concerning some or all of these three aspects. AIM uses the predicted squared error (PSE) criterion for selection and stopping to avoid model overfitting. This criterion minimizes the expected squared error that would be obtained when the network is used for predicting unseen data. PSE is expressed as [7]:

$$\text{PSE} = \text{FSE} + \text{CPM}\left(\frac{2K}{N}\right)\sigma_p^2 \quad (3)$$

where FSE is the fitting squared error on the training data, CPM is a complexity penalty multiplier selected by the user, K is the number of model coefficients, N is the number of samples in the training set, and σ_p^2 is a prior estimate for the variance of the error obtained with the unknown model, usually taken as half the variance of the dependent variable. PSE goes through a minimum at the optimum model size that strikes a balance between accuracy and simplicity (*a.k.a.* exactness and generality). The user may optionally control this trade-off using the CPM parameter that has a default value

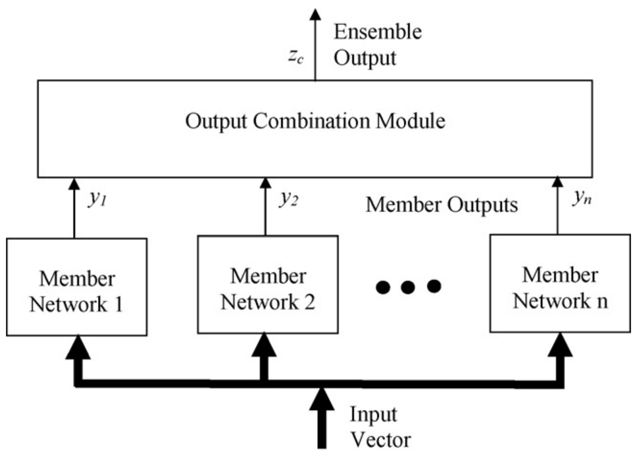


Fig. 2. A schematic diagram for a network ensemble.

of 1. Smaller values of CPM produce more complex networks that may overfit the training data, thus degrading the prediction performance. However, larger values lead to simpler models that can generalize well with new data but are less accurate.

2.2. Network ensembles (committees)

Network ensemble (committee) [40] is a relatively recent learning approach, in which n networks are trained to solve the same problem in a parallel independent fashion. During prediction the networks operate simultaneously on the input data and their outputs are combined to produce the final committee output; see Fig. 2. The output combination module often performs simple functions on the outputs of individual members, such as majority voting and simple or weighted averaging without involving the input features [26]. Alternatively, a gating network may use the input features to determine the optimum weighting factors used with individual member outputs for each case to be classified [38]. In the stacked generalization approach, the combiner takes the form of another higher level network trained on the outputs of individual members to generate the committee classification output [47].

Simple averaging often provides an effective method of combining continuous outputs from individual committee members using $z_c = (1/n)\sum_{i=1}^{n} y_i$, where y_i ($i = 1, 2, \ldots, n$) is the output of committee member i and z_c is the combined committee output. The above relationship assumes that outputs from all members are of equal accuracy. In practice, some outputs may have greater certainty than others, and individual outputs may be weighted to reflect this fact [26]. The committee output is then a weighted sum of the outputs of all members as given by $z_c = \sum_{i=1}^{n} \alpha_i y_i$, where $\sum_{i=1}^{n} \alpha_i = 1$. As a static measure, the certainty c_i of the output from member i can be expressed as the inverse of the variance of the error (σ_i^2) by that member over its training set [26], *i.e.* $c_i = 1/\sigma_i^2$, and the weight α_i is then determined by $\alpha_i = c_i / \sum_{j=1}^{n} c_j$, which satisfies the normalization condition on the weights, *i.e.* $\sum_{i=1}^{n} \alpha_i = 1$.

When member networks are independent, the resulting diversity in the decision making process is expected to boost generalization performance, thus improving the accuracy, robustness, and reliability of predictions. Combining the outputs of several identical networks produces no gain, and improvement is expected only when members err in different ways so that errors may cancel out [39]. An ideal committee would therefore consist of highly accurate networks that disagree as much as possible. Enhancing diversity among individual members of neural network ensembles has been achieved through training on different parts of the dataset, on different input features, or using different network architectures, learning paradigms or training parameters. In the "committee of experts" approach, members are developed using various machine-learning techniques that adopt different ways to build decision boundaries for the classification problem at hand, such as neural networks, nearest neighbor classifiers, classification and regression trees (CART), etc. This allows training adequate individual models on the full training dataset available while ensuring a good degree of diversity among them.

2.3. GMDH-based feature ranking and selection

This paper employs a novel GMDH-based approach for ranking the input features in a training set according to their predictive quality [1]. With this approach, the abductive learning algorithm is repeatedly forced to automatically select a small subset of optimum predictors at reduced model complexity settings. Selected features are then removed from the dataset and the process is repeated for the selection of a new subset of a lower quality. In this way, features are ranked in groups of different levels of predictive quality, with those selected earlier being better predictors. Depending on the problem at hand, it may be possible to further rank the features within each group by constructing models using only those features and repeating the selection process at greater simplicity settings.

Two approaches can be adopted for selecting a feature subset from the ranking list obtained above. In the first approach, a compact m-feature subset can be obtained by arbitrarily considering only the top m features in the ranking list. In the second approach, the optimum subset of features is determined by repeatedly forming subsets of k features, $k = 1, 2, 3, \ldots, n$, where n is the total number of available features, starting at the top of the ranking list. A classifier is then trained on each of the formed subsets. As k increases, the classification error rate for the resulting models on the training set is expected to monotonically decrease as the models fit the training data more accurately. However, performance on an out-of-sample evaluation dataset would first improve and then starts to deteriorate as the model overfits the training data. The optimum model corresponding to the optimum feature subset would correspond to the smallest value for k where the minimum classification error rate on the evaluation set is reached.

3. Performance measures

Several performance measures used in data mining and information retrieval can be adopted for evaluating and comparing the effectiveness of the proposed spam detection approaches [16,30]. These measures include confusion matrix, classification accuracy, false-positive rate, false negative rate, spam recall and precision, legitimate recall and precision, and F-measure. The confusion matrix or contingency table is commonly used in visualizing the performance of a binary classifier. It shows the actual and predicted classification of each category. Other performance measures can be calculated from the confusion matrix as shown in Table 1; where TP denotes the number of spam messages that are correctly detected (*i.e.* true positives), FP denotes the number of legitimate messages that are falsely classified as spam (*i.e.* false positives), FN denotes the number of spam messages that are falsely classified as legitimate (*i.e.* false negative) and TN denotes the number of legitimate messages that are correctly classified (*i.e.* true negatives).

The classifier performance can be also described by the Receiver Operating Characteristic (ROC) curve [16] which shows the trade-off between the true positive rate (TPR) and the false-positive rate (FPR) as the discrimination threshold between the two classified categories is varied. The ROC curve is useful for visualizing and comparing the performance of different classifiers. It is widely used in signal detection theory, medical decision making systems, machine learning and data mining research [16]. Equivalently, the ROC graph

Table 1
Performance measures.

Performance measure	Formula	Interpretation
Overall classification accuracy	$\text{Acc} = \frac{\text{TP+TN}}{\text{TP+FP+FN+TN}}$	Measures the effectiveness of the classifier in terms of the fraction of correctly classified messages; its complementary is *error rate*, err = 1 − Acc
False-negative rate	$\text{FNR} = \frac{\text{FN}}{\text{TotalPos}} = \frac{\text{FN}}{\text{TP+FN}}$	Measures the fraction of spam emails classified as legitimate; its complementary parameter is *sensitivity*, SNS = 1 − FNR
False-positive rate	$\text{FPR} = \frac{\text{FP}}{\text{TotalNeg}} = \frac{\text{FP}}{\text{FP+TN}}$	Measures the fraction of legitimate emails classified as spam; its complementary parameter is *specificity*, SPC = 1 − FPR
Spam recall	$\text{SR} = \frac{\text{TP}}{\text{TotalPos}} = \frac{\text{TP}}{\text{TP+FN}}$	Measures the fraction of spam messages rejected by the filter; also called *sensitivity* (SNS) or true positive rate (TPR)
"Spam" precision	$\text{SP} = \frac{\text{TP}}{\text{TP+FP}}$	Measures the degree to which messages rejected by the spam filter are truly spam; also called positive predictive value
"Legitimate" precision	$\text{LP} = \frac{\text{TN}}{\text{TN+FN}}$	Measures the degree to which messages accepted by the spam filter are truly legitimate; also called negative predictive value
F-measure	$\text{FM}_{\text{spam}} = \frac{2\text{SP}\cdot\text{SR}}{\text{SP+SR}} = \frac{2}{(1/\text{SP})+(1/\text{SR})}$	A combined measure of recall and precision

can be drawn to show the tradeoff between spam precision (SP) versus spam recall (SR) as the discrimination threshold is varied; this is also known as the precision–recall curve. An important scalar measure that can be calculated from the ROC curve is the area under the curve (AUC) [24]. The AUC is often used as an alternative to the classifier accuracy. In [16], Fawcett shows that the AUC is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. Hence, it is desirable to have a classifier with higher AUC values. AUC is 1.0 for an ideal classifier which gives 100% sensitivity at 100% specificity. Practically, useful classifiers would have AUC values in the range ($0.5 < \text{AUC} \leq 1.0$).

4. Description of the dataset

To evaluate and compare the performance of the proposed approach, we used the spambase dataset [42]. This database has been created in June–July 1999 by M. Hopkins, E. Reeber, G. Forman, and J. Suermondt at Hewlett-Packard Labs. It consists of 4601 instances of legitimate and spam email messages with 39.4% being spam. Each instance is characterized by 57 input attributes and is labeled as spam (represented as 1) or legitimate (represented as 0). Table 2 lists the attribute number and name, as well as the average and standard deviation values for each attribute in both the legitimate and spam populations. Attributes 1–48 give the percentage of words in the email message for the respective keyword indicated in the attribute name. Attributes 49–54 give the percentage of characters in the email message for the respective character indicated in the attribute name. Attributes 55 and 56 give the average and maximum lengths, respectively, of uninterrupted sequences of capital letters in the message. Attribute 57 gives the total number of capital letters in the message. The attribute number will be used as the variable number for models described throughout this paper. Attribute number 58 in the dataset is the true email class. The dataset has no missing attribute values. Table 2 also lists the mean and standard deviation of each attribute in both the legitimate population (μ_l and σ_l respectively) and spam population (μ_s and σ_s respectively). The last column in Table 2 gives the absolute value of the *Z* statistic for each attribute. The *z* parameter measures the magnitude of the absolute difference between the two means of a given attribute in the legitimate and spam populations relative to variations about the two means. The larger the value of *z*, the more significant is the difference between the two means and therefore the larger the value discriminatory power of the attribute in classifying the dataset as spam/legitimate. Sorting the list of input attributes in a descending order based on the values of their *Z* statistic gives the following ranking: {21, 25, 26, 19, 23, 7, 53, 52, 16, 5, 27, 17, 57, 6, 11, 9, 30, 37, 3, 24, 18, 35, 28, 8, 46, 56, 45, 42, 43, 29, 15, 36, 31, 20, 39, 33, 32, 10, 34, 1, 13, 41, 44, 48, 55, 50, 22, 40, 51, 49, 14, 54, 47, 4, 38, 2, 12}. Attributes toward the beginning of the list should prove more effective than attributes toward the end. However, this simple ranking approach does not take into account possible mutual correlation between various attributes when grouped together to classify the dataset. Since using several attributes is more common in discriminating spam from legitimate emails and leads to better results, forming subsets of attributes based on the ranking of individual attributes may not produce compact subsets of predictors and may miss complementary attributes that have low individual ranking.

5. Experiments and results

5.1. AIM abductive network modeling

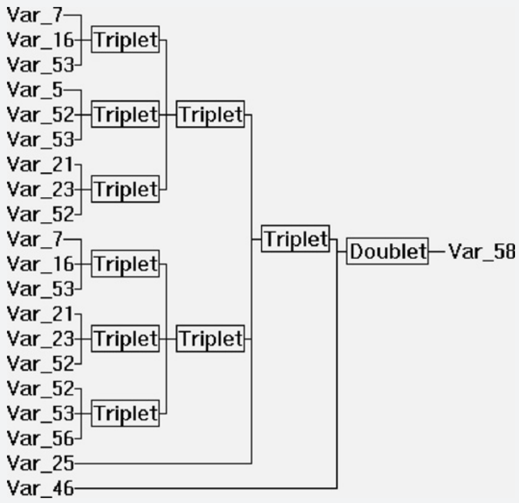
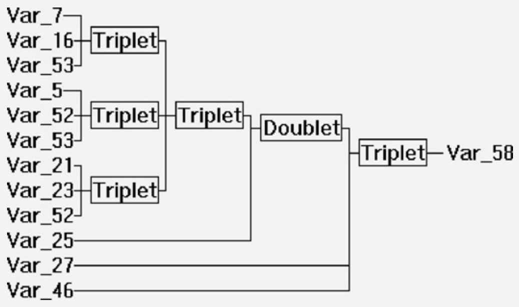
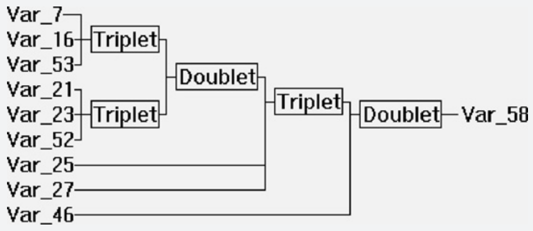
Abductive network models were developed for classifying the spambase dataset into spam or legitimate messages. The dataset was randomly split into a training set of 2844 cases and an evaluation set of 1757 cases. The percentage of spam in the two datasets was 38.4% and 41.1%, respectively. In the first modeling experiment, the full training set was used to develop abductive network models that utilize all available 57 attributes for spam classification. We refer to such models as monolithic models to contrast them with those developed later using the modular approach of abductive network ensembles to be described in Section 5.2. Table 3 shows the abductive model structures synthesized at three levels of model complexity as indicated by the CPM parameter specified prior to training, together with their classification accuracy on both the training and evaluation sets. The variable number indicated at a model input, *e.g.* Var_*i*, corresponds to the number of the attribute selected as input to the model during training. Var_58 is the binary classification model output (*i.e.* spam or legitimate). As the CPM value increased, simpler models are synthesized. In all three models developed with CPM = 0.3, 1, and 3; only a small subset of the 57 input attributes is selected automatically during training. The number of selected attributes is 10, 10, and 9, respectively. Eight attributes {7, 16, 21, 23, 25, 46, 52, and 53} are common to all three input subsets. This indicates the effectiveness of such attributes as spam predictors and the robustness of the modeling process. During model evaluation on the evaluation set, computed model output was rounded to 0 (legitimate) or 1 (spam) based on a simple threshold of 0.5. We will later consider operation at other thresholds by presenting the receiver operator characteristics in Section 5.4. Classification performance on the external evaluation set is approximately equal to that on the training set. Best classification

Table 2
List of names and statistical properties for the 57 input attributes of the spambase dataset consisting of 4601 cases for both the legitimate and spam populations. The attribute number is used to identify the attribute as a model input variable throughout the paper. The absolute value of the *Z* statistic is used to provide a tentative ranking of the 57 attributes. Rows shaded correspond to the most effective spam detection attributes; see text at the end of Section 6.1.

Attribute #	Attribute name	'Legitimate' population ($n_l = 2788$)		'Spam' population ($n_s = 1813$)		*Z* statistic, $z = \frac{\lvert\mu_l - \mu_s\rvert}{\sqrt{\frac{\sigma_l^2}{n_l} + \frac{\sigma_s^2}{n_s}}}$
		Mean, μ_l	Standard deviation, σ_l	Mean, μ_s	Standard deviation, σ_s	
1	word_freq_make:	0.073	0.298	0.152	0.311	8.55
2	word_freq_address:	0.244	1.633	0.165	0.349	2.49
3	word_freq_all:	0.201	0.503	0.404	0.481	13.76
4	word_freq_3d:	0.001	0.021	0.165	2.219	3.14
5	word_freq_our:	0.181	0.615	0.514	0.707	16.42
6	word_freq_over:	0.045	0.223	0.175	0.322	15.05
7	word_freq_remove:	0.009	0.110	0.275	0.572	19.56
8	word_freq_internet:	0.038	0.247	0.208	0.545	12.46
9	word_freq_order:	0.038	0.199	0.170	0.355	14.44
10	word_freq_mail:	0.167	0.643	0.351	0.631	9.55
11	word_freq_receive:	0.022	0.150	0.118	0.251	14.79
12	word_freq_will:	0.536	0.979	0.550	0.641	0.57
13	word_freq_people:	0.062	0.259	0.144	0.350	8.55
14	word_freq_report:	0.042	0.344	0.084	0.319	4.15
15	word_freq_addresses:	0.008	0.094	0.112	0.387	11.20
16	word_freq_free:	0.074	0.617	0.518	1.013	16.78
17	word_freq_business:	0.048	0.219	0.288	0.626	15.65
18	word_freq_email:	0.097	0.397	0.319	0.666	12.79
19	word_freq_you:	1.270	1.794	2.265	1.567	19.85
20	word_freq_credit:	0.008	0.097	0.206	0.788	10.64
21	word_freq_your:	0.439	1.025	1.380	1.227	27.09
22	word_freq_font:	0.045	0.610	0.238	1.441	5.39
23	word_freq_000:	0.007	0.068	0.247	0.519	19.57
24	word_freq_money:	0.017	0.272	0.213	0.601	13.04
25	word_freq_hp:	0.895	2.071	0.017	0.161	22.28
26	word_freq_hpl:	0.432	1.105	0.009	0.099	20.08
27	word_freq_george:	1.265	4.253	0.002	0.033	15.69
28	word_freq_650:	0.194	0.637	0.019	0.305	12.47
29	word_freq_lab:	0.163	0.755	0.001	0.013	11.33
30	word_freq_labs:	0.166	0.572	0.006	0.103	14.40
31	word_freq_telnet:	0.106	0.513	0.001	0.036	10.74
32	word_freq_857:	0.077	0.419	0.001	0.016	9.66
33	word_freq_data:	0.151	0.703	0.015	0.112	10.05
34	word_freq_415:	0.078	0.420	0.002	0.036	9.51
35	word_freq_85:	0.169	0.674	0.007	0.066	12.64
36	word_freq_technology:	0.142	0.498	0.030	0.148	11.15
37	word_freq_1999:	0.198	0.490	0.043	0.268	13.76
38	word_freq_parts:	0.019	0.280	0.005	0.050	2.58
39	word_freq_pm:	0.122	0.549	0.012	0.090	10.29
40	word_freq_direct:	0.083	0.432	0.037	0.152	5.20
41	word_freq_cs:	0.072	0.462	0.000	0.002	8.23
42	word_freq_meeting:	0.217	0.976	0.002	0.027	11.59
43	word_freq_original:	0.071	0.282	0.008	0.050	11.36
44	word_freq_project:	0.127	0.794	0.006	0.061	7.97
45	word_freq_re:	0.416	1.260	0.125	0.323	11.61
46	word_freq_edu:	0.287	1.153	0.015	0.134	12.35
47	word_freq_table:	0.008	0.097	0.001	0.019	3.70
48	word_freq_conference:	0.051	0.365	0.002	0.027	7.07
49	char_freq_;:	0.050	0.303	0.021	0.092	4.84
50	char_freq_(:	0.159	0.261	0.109	0.282	6.00
51	char_freq_[:	0.023	0.135	0.008	0.047	5.20
52	char_freq_!:	0.110	0.821	0.514	0.744	17.26
53	char_freq_$:	0.012	0.070	0.174	0.360	19.00
54	char_freq_#:	0.022	0.244	0.079	0.612	3.79
55	capital_run_length_average:	2.377	5.114	9.519	49.846	6.08
56	capital_run_length_longest:	18.214	39.085	104.393	299.285	12.19
57	capital_run_length_total:	161.471	355.738	470.619	825.081	15.07

performance on the evaluation set is obtained using the optimum model with CPM = 1 which gives a classification accuracy of 91.7%. This model will be considered as the optimum monolithic model throughout the paper. The subset of 10 input attributes selected by this model is {5, 7, 16, 21, 23, 25, 27, 46, 52, and 53}. Eight of these 10 attributes are among the top 10 positions of the *Z* statistic ranking given in Section 4, which confirms their superior explanatory qualities. Table 4(a) shows the resulting confusion matrix and Table 5(a) lists the parameters characterizing the classification performance of this optimum model on the evaluation set, in terms of sensitivity, specificity, "spam" precision, and "legitimate" precision, as well as classification accuracy. The results indicate a minimum value of approximately 88.2% for all performance parameters. A good aspect of this classifier is that specificity is greater than sensitivity. This means fewer false-positive (FP) classification errors where a legitimate email is wrongly classified as spam email, which is a more serious outcome compared to a spam being considered as legitimate *i.e.* a false negative (FN). Table 4(a) gives the number of FP and FN errors as 61 and 85, respectively, out of the 1757 evaluation cases.

Table 3
Structure and performance of the spam detection abductive models trained on 2844 cases at various levels of specified model complexity. Specified values for the other training parameters: number of layers = 4, and Size of first layer = 15. Evaluation set has 1757 cases.

CPM	Model structure	Input attributes selected	Classification accuracy, %	
			On training set	On evaluation set
0.3		5, 7, 16, 21, 23, 25, 46, 52, 53, 56	91.4	91.6
1		5, 7, 16, 21, 23, 25, 27, 46, 52, 53	91.6	91.7
3		7, 16, 21, 23, 25, 27, 46, 52, 53	91.5	91.5

5.2. *Improving classification accuracy using abductive network ensembles*

We have investigated improving the spam detection accuracy through the use of a classifier based on a modular network ensemble (committee) of the type described in Section 2.2. A network ensemble was constructed by combining the outputs of three abductive models trained on different subsets of the full training set using all the available 57 attributes. The training set was randomly split into three equal portions of 948 cases each (with approximately equal spam ratios of 38.1%, 38.2% and 38.8%) which were then used to train the three models at the same CPM value of 1. Table 6 shows the structure of the resulting models for the 3 ensemble members together with the classification accuracy for each individual model on both the training set (948 cases) and evaluation set (1757 cases). The models select between 9 and 11 attributes as inputs, with 7 attributes {7, 16, 25, 27, 46, 52, and 53} being common to all three models. Out of this subset of 7 attributes, six attributes {7, 16, 25, 46, 52, and 53} were also common to the models in Table 3. Performance measured on the training set appears somewhat superior to that on the evaluation set, which can be attributed to the fact that the training set for each model is smaller than the evaluation set for the ensemble.

Two methods were used to combine the outputs of the 3 ensemble members to form the final output of the ensemble classifier. The first approach uses simple averaging of the raw model outputs followed by simple rounding (at threshold = 0.5) to either 0 or 1. The second approach uses majority voting of the binary classification outputs of individual members. The confusion matrices for the resulting two classifiers are shown in Table 4(b) and (c) and the corresponding parameters characterizing the classification performance are given in Table 5. The results show that the ensemble approach improves classification performance beyond that of the optimum monolithic model, with more improvement obtained through combining the outputs of individual members prior to discretization. Further improvement with the majority voting approach was achieved by introducing more diversity among the ensemble members through the use of different rounding thresholds (0.4, 0.6, and 0.5 for ensemble members 1, 2, and 3, respectively), see results in Tables 4(d) and 5. This resulted in the best overall classification accuracy of 92.4%. It is also encouraging to note that improvements introduced by network ensembles

Table 4
Confusion matrices obtained when classifying the evaluation dataset of 1757 cases using: (a) monolithic optimum abductive model, (b) 3-member ensemble network classifier with simple averaging of the continuous member outputs, (c) 3-member ensemble network classifier with majority voting among the binary member outputs with the same output rounding threshold of 0.5 for all members, and (d) 3-member ensemble network classifier with majority voting among the binary member outputs with different output rounding thresholds of 0.4, 0.6, and 0.5 for the 3 members.

			Predicted
(a)			
		1 (698)	0 (1059)
Actual	1 (722)	637	85
	0 (1035)	61	974
(b)			
		1 (681)	0 (1076)
Actual	1 (722)	634	88
	0 (1035)	47	988
(c)			
		1 (676)	0 (1081)
Actual	1 (722)	628	94
	0 (1035)	48	987
(d)			
		1 (677)	0 (1080)
Actual	1 (722)	633	89
	0 (1035)	44	991

tend to reduce false positives, which helps reduce the number of legitimate messages discarded as spam. FP value is reduced from 61 with the optimum monolithic model to 44 for the ensemble model of Table 4(d), while FN is increased only from 85 to 89. This means that with this ensembling method, out of every 3 misclassified emails, one legitimate email is considered as spam for every 2 spam emails misclassified as legitimate.

5.3. GMDH-based ranking of the spam detection attributes

Approximate ranking of the 57 spam attribute comprising the dataset was carried out through model training in 18 steps using the procedure described in Section 2.3. Unless otherwise mentioned, all steps were performed at the same model complexity settings of CPM = 10 (maximum value permitted with the AIM version used), the maximum number of model layers = 1, and the size of the first layer = 3. Initially, all input features were enabled for selection as inputs for the synthesized model by the abductive learning algorithm. Following modeling step i; $i = 1, 2, \ldots, 17, 18$, inputs selected for the model synthesized during that step were disabled in order to prevent their use as model inputs in all succeeding steps: $i + 1$, $i + 2, \ldots, 17, 18$. This forces selection from lower quality inputs and allows partial ranking of the overall feature space in the form of the small groups of items which are sequentially selected. Inputs selected at lower values of i are expected to have superior predictive quality. Results of the 18 modeling steps are shown in Table 7. In addition to the inputs excluded from being selected at each step, the table shows also the structure of the synthesized abductive model and its classification accuracy on the evaluation set. In 14 out of the 18 steps, a 3-input triplet model was synthesized. Toward later steps, input features available for selection become progressively poorer in predictive quality, thus driving the training algorithm to select a larger subset of inputs to ensure adequate prediction performance by the synthesized model. With the AIM software used, this requirement could override the limit of 3 specified for the size of the first layer in the model. For example, the model generated at step 15 is a linear functional element with 11 inputs. At the last step (step 18), the predictive quality of the remaining two attributes, {38, 47} were so poor that the CPM value had to be reduced from 10 to 5 to allow a model to be synthesized. Due to the gradual degradation in the predictive quality of selected model inputs, there is a general trend of decreasing classification accuracy at later steps, with the accuracy dropping from 86.7% at step 1 to 58.9% at step 18. Based on the assumption that higher quality predictors are selected at earlier steps, the GMDH-based ranking of the groups is identical to the group number, with group 1 {attributes 7, 52, 53} having the highest predictive quality and group 18 {38, 47} having the lowest quality. This suggests the following partial ranking list for the 57 test items: {7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8, 20, 55, 6, 18, 30, 2, 9, 35, 12, 15, 28, 1, 29, 46, 10, 13, 31, 32, 40, 42, 33, 45, 54, 4, 14, 22, 36, 39, 41, 43, 44, 48, 49, 51, 34, 50, 38, 47}, where items within a group are listed in the order they appeared in the model structures shown in Table 7. The three attributes in group 1 alone explain 85.7% of the variations in the evaluation dataset. All 6 attributes in the first two groups in Table 7 are included in the 10 attributes used by the optimum monolithic model at CPM = 1 shown in Table 3. The 12 attributes forming the top 4 groups in Table 7 include 90% of the attributes used by that optimum model.

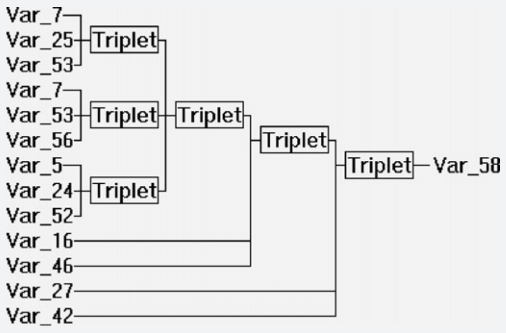
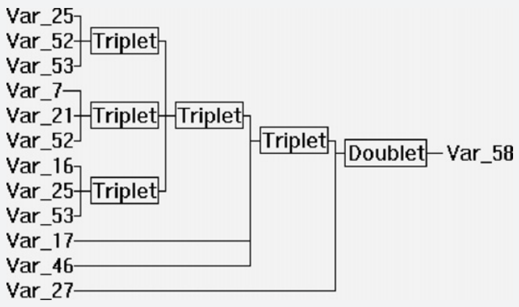
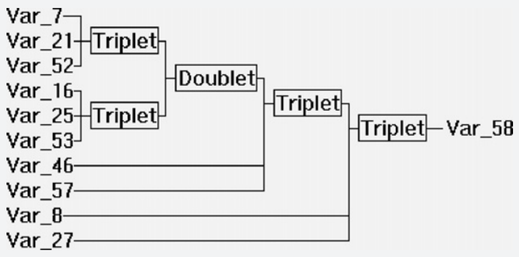
5.4. ROC analysis

Classifier performance results described above were carried out at a single cutoff point (0.5) marking the legitimate/spam transition in the continuous value of model output for generating the binary classification output. A more useful comparison would involve several such cutoff points over the range 0.0–1.0 using ROC analysis. ROC analysis was used to evaluate and compare the performance of the optimum monolithic at CPM = 1 in Table 3 and the ensemble network classifier using the 3 members of the ensemble in Table 6. We used the Analyse-it statistical software package [4] which employs the Hanley and McNeil method [24]. Fig. 3 plots the two ROC curves obtained and gives values of the AUC parameter and its standard error (SE) for each model. Results indicate that both models give adequate performance with the values for the

Table 5
Detailed classification characteristics obtained when classifying the evaluation dataset of 1757 cases using the four models (a)–(d) of Table 4. The code value will be used when referring to different models.

Code	Model		Sensitivity, SNS, %	Specificity, SPC, %	"Spam" precision, SP, %	"Legitimate" precision, LP, %	Classification accuracy, Acc, %
ABD1	a	Optimum monolithic model (CPM = 1)	88.2	94.1	91.3	92.0	91.7
ABD2	b	3-Member ensemble (simple averaging)	87.8	95.5	93.1	91.8	92.3
ABD3	c	3-Member ensemble (majority voting, same output rounding threshold of 0.5)	87.0	95.4	92.9	91.3	91.9
ABD4	d	3-Member ensemble (majority voting, different output rounding thresholds: 0.4, 0.6, 0.5)	87.7	95.7	94.5	91.8	92.4

Table 6
Structure and performance of the 3-member models constituting the abductive network ensemble classifier. Models are trained using all 57 features on three different training subsets, each of 948 cases, and evaluated on the same evaluation set of 1757 cases. Specified values for other training parameters: CPM = 1, number of layers = 4, and size of first layer = 15.

Ensemble member #	Model structure	Input attributes selected	Classification accuracy, %	
			On training set	On evaluation set
1	 Var_7 Var_25 Var_53 Var_7 Var_53 Var_56 Var_5 Var_24 Var_52 Var_16 Var_46 Var_27 Var_42 Triplet Triplet Triplet Triplet Triplet Triplet Var_58	5, 7, 16, 24, 25, 27, 42, 46, 52, 53, 56	93.4	91.2
2	 Var_25 Var_52 Var_53 Var_7 Var_21 Var_52 Var_16 Var_25 Var_53 Var_17 Var_46 Var_27 Triplet Triplet Triplet Triplet Triplet Doublet Var_58	7, 16, 17, 21, 25, 27, 46, 52, 53	93.8	90.7
3	 Var_7 Var_21 Var_52 Var_16 Var_25 Var_53 Var_46 Var_57 Var_8 Var_27 Triplet Triplet Doublet Triplet Triplet Var_58	7, 8, 16, 21, 25, 27, 46, 52, 53, 57	92.7	91.0

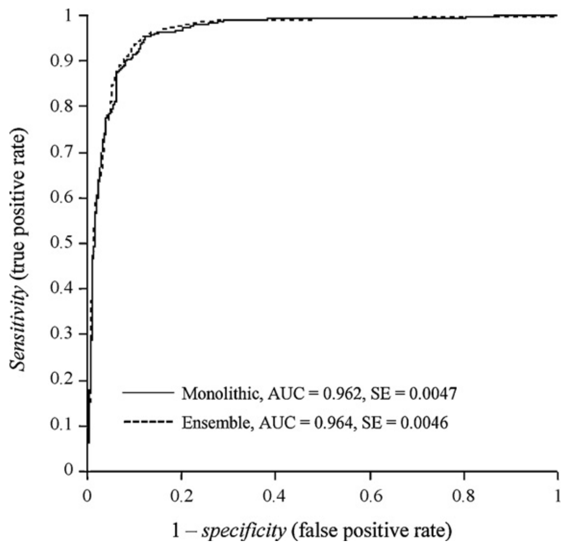


Fig. 3. Comparison of the ROC characteristics for two abductive network classifiers: the optimum monolithic model (CPM = 1 in Table 3) and the 3-member network ensemble using simple averaging of linear member outputs (member models shown in Table 6). Indicated on the figure are the values for the area under the curve (AUC) and the associated standard error (SE) in each case.

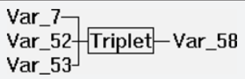
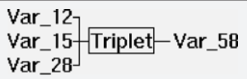
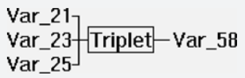
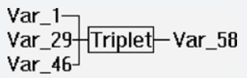
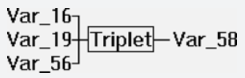
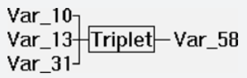
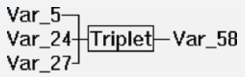
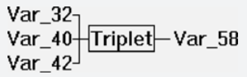
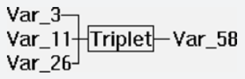
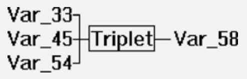
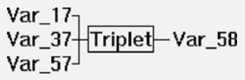
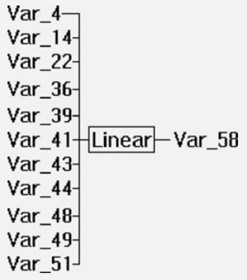
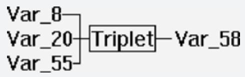
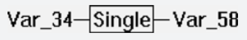
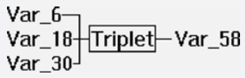
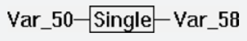
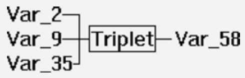
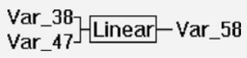
AUC being 0.962 and 0.964, respectively. The ensemble classifier shows a slightly larger AUC value and smaller SE.

6. Comparisons with other techniques

6.1. Comparison with probabilistic neural networks

We have compared the classification performance and the attribute ranking obtained above using abductive modeling with results obtained using probabilistic neural networks employing a genetic learning algorithm, which were trained and evaluated on the same datasets using the NeuroShell Classifier software [45]. Compared to conventional neural networks, such networks take longer time to train but have the advantage of generalizing well on external data [45]. Table 8 shows the results of comparing the optimum abductive network model (middle row of Table 3) with four PNNs with genetic learning. Prior to training, the maximum number of generations allowed without performance improvement was set to 20, and the goal of the genetic optimization during training was set to minimize the total number of incorrect classifications in both classification categories. The latter criterion was used in order to match that adopted in the training and selection of optimum abductive network models. Table 8 lists details of the networks considered including input attributes used for training, training

Table 7
Structure and performance of the simplest classification abductive models synthesized in a sequence of 18 steps, with inputs selected at a given step excluded in all subsequent steps. Training on 2844 cases and evaluation on 1757 cases. Specified training parameters for all steps: CPM = 10, number of layers = 1, and size of first layer = 3. The percentage classification accuracy shown was determined for the evaluation set.

Step	Attributes excluded from selection as inputs	Model synthesized	Classification accuracy, %	Step	Attributes excluded from selection as inputs	Model synthesized	Classification accuracy, %
1	None		85.7	10	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8, 20, 55, 6, 18, 30, 2, 9, 35		64.7
2	7, 52, 53		83.4	11	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8, 20, 55, 6, 18, 30, 2, 9, 35, 12, 15, 28		66.5
3	7, 52, 53, 21, 23, 25		80.8	12	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8, 20, 55, 6, 18, 30, 2, 9, 35, 12, 15, 28, 1, 29, 46		66.6
4	7, 52, 53, 21, 23, 25, 16, 19, 56		79.7	13	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8, 20, 55, 6, 18, 30, 2, 9, 35, 12, 15, 28, 1, 29, 46, 10, 13, 31		61.8
5	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27		74.7	14	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8, 20, 55, 6, 18, 30, 2, 9, 35, 12, 15, 28, 1, 29, 46, 10, 13, 31, 32, 40, 42		62.1
6	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26		74.4	15	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8, 20, 55, 6, 18, 30, 2, 9, 35, 12, 15, 28, 1, 29, 46, 10, 13, 31, 32, 40, 42, 33, 45, 54	 Var_4 Var_14 Var_22 Var_36 Var_39 Var_41 Var_43 Var_44 Var_48 Var_49 Var_51 Linear Var_58	61.6
7	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57		72.4	16	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8, 20, 55, 6, 18, 30, 2, 9, 35, 12, 15, 28, 1, 29, 46, 10, 13, 31, 32, 40, 42, 33, 45, 54, 4, 14, 22, 36, 39, 41, 43, 44, 48, 49, 51		58.9
8	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8, 20, 55		71.5	17	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8, 20, 55, 6, 18, 30, 2, 9, 35, 12, 15, 28, 1, 29, 46, 10, 13, 31, 32, 40, 42, 33, 45, 54, 4, 14, 22, 36, 39, 41, 43, 44, 48, 49, 51, 34		58.9
9	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8, 20, 55, 6, 18, 30		69.9	18	7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8, 20, 55, 6, 18, 30, 2, 9, 35, 12, 15, 28, 1, 29, 46, 10, 13, 31, 32, 40, 42, 33, 45, 54, 4, 14, 22, 36, 39, 41, 43, 44, 48, 49, 51, 34, 50		58.9

time, number of generations (for the PNN networks), as well as four aspects of classification performance, namely the area under the ROC curve (AUC), sensitivity (SNS), specificity (SPC), and classification accuracy (Acc). Models considered in Table 8 are given a serial number from 1 to 5.

The first PNN network considered (model 2 in Table 8) used all 57 attributes as inputs and training continued for 75 generations before it was stopped by the algorithm due to lack of performance improvement. Training took an excessively long time of nearly 12 h compared to a fraction of a minute by the abductive network model (model 1 in Table 8). Nevertheless, the abductive model outperforms the neural model nearly on all aspects of classification performance including a 0.6 percent point increase in classification accuracy, a 2.1 percent point improvement in specificity, and a larger AUC value. The larger specificity by the abductive model reduces the number of legitimate emails discarded as spam by the

Table 8
Comparison between the optimum monolithic abductive model and four probabilistic neural network (PNN) models with genetic training developed using the NeuroShell Classifier software.

Learning paradigm	Serial #	Model	Input attributes used for training	Number of training generations (genetic)	Training time, min	Classification performance			
						Area under ROC curve (AUC)	Sensitivity, SNS, %	Specificity, SPC, %	Classification accuracy, Acc, %
GMDH	1	Optimum monolithic model (CPM = 1)	All 57 attributes	–	0.25	0.962	88.2	94.1	91.7
Probabilistic neural network (PNN) with genetic training (NeuroShell Classifier software)	2	Using all attributes	All 57 attributes	75**	698	0.938	89.8	92.0	91.1
	3	Using best 19 attributes*	{7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8}	100	310	0.953	87.0	94.5	91.3
	4	Using middle 19 attributes*	{20, 55, 6, 18, 30, 2, 9, 35, 12, 15, 28, 1, 29, 46, 10, 13, 31, 32, 40}	64**	198	0.872	73.3	89.0	82.5
	5	Using poorest 19 attributes*	{42, 33, 45, 54, 4, 14, 22, 36, 39, 41, 43, 44, 48, 49, 51, 34, 50, 38, 47}	100	310	0.728	90.1	52.2	68.2

* As per the tentative GMDH-based ranking of the attributes in Table 7.
** Training stopped by the algorithm due to lack of improvement for 20 successive generations.

spam filter. It is noted, however, that we have not utilized other available genetic optimization options that optimize the average performance over all categories or take into account different costs for various types of misclassifications [45].

Models 3–5 are three PNN networks trained on three different feature subsets, each containing 19 attributes, of the 57 input features. Model 3 uses the top 19 attributes according to the GMDH-based tentative attribute ranking obtained in Table 7, while models 4 and 5 were trained on the middle and bottom 19 attributes in the ranking list, respectively. These models were developed in order to verify the attribute ranking results based on the GMDH approach through the use of a different learning paradigm. The overall classification accuracy and the AUC values exhibit significant deterioration moving from model 3 through model 5, which indicates gradual degradation in the predictive quality of the subsets of 19 attributes used to train the three models. Moreover, it is interesting to note that model 3 trained on the top 19 attributes outperforms model 2 trained on all 57 attributes, while requiring a much shorter training time. This indicates that insights gained from GMDH-based attribute ranking can be put to advantage in gaining data reduction benefits in other modeling and learning modalities.

The NeuroShell Classifier software provides an estimate of the relative importance of each input used to train the model, which is updated during training. It would be interesting to compare results using this feature with the ranking based on the *Z* statistic and the tentative GMDH-based ranking. Unfortunately, the NeuroShell ranking is reliable only for a small number of inputs, *e.g.* up to 20 inputs [45], and therefore could not be reliably used to give an overall ranking of the full set of 57 attributes of the spambase dataset. Fig. 4 is a bar chart depicting the importance of input attributes, which is produced by the software at the end of training for model 3 in Table 8 on the top 19 attributes. According to Fig. 4, the most important 12 of these attributes are {57, 27, 25, 53, 23, 52, 37, 56, 7, 5, 21, 16}. As shown in Fig. 5, this subset contains 9 out of the 12 attributes at the top of the *Z* statistic ranking list and 10 out of the 12 attributes at the top of the GMDH-based ranking list. This indicates reasonable agreement between the three ranking schemes considered. In all three ranking lists, attribute 23 occupies the 5th position from the top, while attribute 5 occupies the 10th position. The 9 attributes {5, 7, 16, 21, 23, 25, 27, 52, 53}, which are shaded in Fig. 5, are common to the top 12 attributes in all three ranking lists. It is interesting to note that all these 9 attributes are included in the subset of 10 input attributes selected automatically by the optimum monolithic abductive model with CPM = 1 (middle row in Table 3). Rows for these most effective spam detection attributes are shaded in Table 2. All these 9 attributes are characterized by a large value of the *Z* statistic. Seven attributes indicate a larger presence of the keywords “our”, “remove”, “free”, “your”, “000” and the characters “!” and “$” in spam messages. The remaining two attributes indicate a larger presence in legitimate messages of the words “george” and “hp”, which refer to the first name and the affiliation of the researcher who developed the spambase dataset.

6.2. *Comparison with multilayer perceptron and naïve Bayesian classifiers*

The effectiveness of using multilayer perceptron neural networks and naïve Bayesian classifiers has been recently evaluated for spam filtering on the spambase dataset used in this paper [48]. In [48], the dataset was randomly shuffled and then partitioned into five independent subsets using 5-fold cross validation. Five experiments were conducted. In each experiment, four subsets were used for training the classifier and the remaining subset was used for evaluation. The authors of [48] examined various MLP architectures to determine the best classifier model which was then tested for various cutoff threshold values (see Table 6 in [48]). They also tested

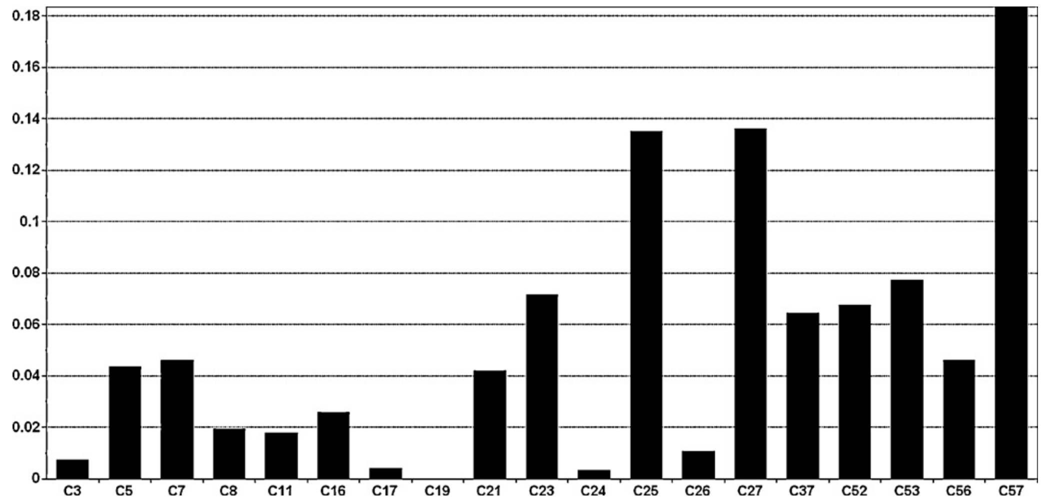
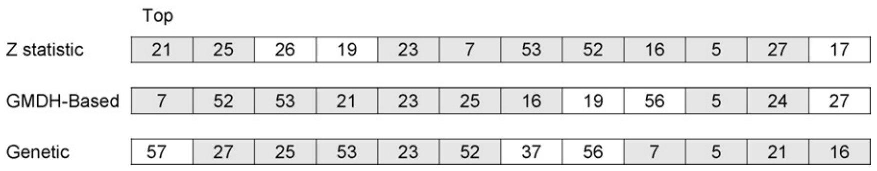


Fig. 4. Relative importance of the 19 input attributes {7, 52, 53, 21, 23, 25, 16, 19, 56, 5, 24, 27, 3, 11, 26, 17, 37, 57, 8} for model 3 in Table 8 as determined by the genetic learning algorithm. These inputs constitute the top third in the tentative GMDH-based ranking of the 57 spam detection attributes (Table 7).



	Top											
Z statistic	21	25	26	19	23	7	53	52	16	5	27	17
GMDH-Based	7	52	53	21	23	25	16	19	56	5	24	27
Genetic	57	27	25	53	23	52	37	56	7	5	21	16

Fig. 5. The top most effective twelve attributes as determined by the *Z* statistic measure, the GMDH-based ranking, and the genetic learning algorithm in NeuroShell Classifier software. Shaded cells indicate the nine attributes which are common to all three ranking lists.

Table 9
Performance comparison of abductive learning with MLP neural networks and naïve Bayesian (NB) classifiers in terms of accuracy (Acc), false-positive rate (FPR), false-negative rate (FNR), spam recall (SR), spam precision (SP) and F-measure (FM).

Method, reference	Model	Acc (%)	FPR (%)	FNR (%)	SR (%)	SP (%)	FM (%)
Abductive networks, Table 5	ABD1	91.7	5.9	11.8	88.2	91.3	89.7
	ABD2	92.3	4.5	12.2	87.8	93.1	90.4
	ABD3	91.9	4.6	13.0	87.0	92.9	89.8
	ABD4	92.4	4.3	12.3	87.7	93.5	90.5
Multi-layer perceptron neural networks (see Table 6 in [48] at threshold = 0.3)	MLP1	89.6	6.2	5.9	88.5	90.0	89.2
	MLP2	91.7	3.9	6.1	89.7	94.2	91.9
	MLP3	89.8	2.6	7.0	85.5	95.8	90.3
	MLP4	89.2	4.0	9.5	84.5	92.7	88.4
	MLP5	90.9	3.9	6.7	88.3	93.5	90.8
Naïve Bayesian classifiers (see Table 7 in [48])	NB1	69.5	14.7	50.4	47.7	68.5	56.2
	NB2	75.4	9.0	44.9	54.8	80.8	65.3
	NB3	70.8	15.8	44.1	53.5	70.6	60.9
	NB4	74.7	8.5	50.4	48.4	77.1	59.5
	NB5	75.1	12.9	43.6	56.1	72.4	63.2

the naïve Bayesian classifier (see Table 7 in [48]). Our comparisons with their best models are based on classification accuracy (Acc), false-positive rate (FPR), false-negative rate (FNR), spam recall (SR), spam precision (SP) and F-measure (FM). Table 9 summarizes the performance of our four abductive modeling experiments of Table 5 of this paper, and that of the five MLP experiments and five NB experiments reported in [48]. In contrast to their work, we label "spam" as positive and "legitimate" as negative, which is more common in anti-spam literature, and we recalculated the performance measures for their work according to the definitions in Section 3. The results indicate that our proposed approaches yield much better performance compared to the naïve Bayesian classifier in all cases and offer a slight improvement on the MLP models in most of the cases in terms of classification accuracy, spam recall and spam precision. Moreover, unlike the MLP that requires long training times (see Section 7 in [48]), the GMDH-based approach requires much shorter training time as shown in Table 8.

7. Conclusions

We have demonstrated the use abductive machine learning for spam detection and the ranking of 57 spam detection attributes of the spambase dataset according to their predictive value. In general, the proposed approach offers the following advantages: improved classification accuracy, greater insight into the email feature set, data reduction through automatic exclusion of insignificant input features, simpler model development, and reduced training time. Classification accuracies of about 91.7% were achieved with a single model that automatically selects only 10 input attributes, thus achieving data reduction by a ratio of approximately 6:1. Accuracy was further improved up to 92.4% through the use of a 3-member abductive network ensembles with the members trained on different subsets of the training set using all 57 attributes. Abductive modeling was compared with probabilistic neural networks employing a genetic learning algorithm and developed using the NeuroShell Classifier software. Results indicate that the abductive approach gives better classification accuracy and requires much shorter training times. Three methods were adopted for ranking the 57 attributes to determine the most effective spam predictors: (1) The *Z*-statistic measure of the significance of the difference between the two means of each attribute for the spam and legitimate subsets of the data, (2) A GMDH-based approach of iteratively forcing the synthesis of a simple model and excluding the inputs selected at each stage, and (3) Information provided by the NeuroShell Classifier on the relative importance of inputs to the genetic learning algorithm. The top most effective 12 predictors in each of the above three rankings have 9 attributes in common. The same 9 attributes are automatically selected by the 10-input optimum monolithic abductive network model. These attributes highlight the greater frequency of certain words or characters in either spam or legitimate messages. Performance was compared with that of MLP neural networks and naïve Bayesian classifiers reported in the literature for the same email dataset. The comparison indicates that the abductive network models provide better classification accuracy, spam recall and spam precision with false-positive rates as low as 4.3%. Future work would consider methods for finer ranking within attribute groups in the GMDH-based method to achieve complete ranking of the item set, and the use of ensembles with members trained using different learning paradigms to achieve further improvements in classification performance.

Acknowledgments

The authors are grateful to King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia for continuous support during this work.

References

[1] R.E. Abdel-Aal, GMDH-based feature ranking and selection for improved classification of medical data, Journal of Biomedical Informatics 38 (2005) 456–468.
[2] AIM User's Manual, AbTech Corporation, Charlottesville, VA, 1990.
[3] A. Agarwal, Abductive networks for two-group classification: a comparison with neural networks, The Journal of Applied Business Research 15 (1999) 1–12.
[4] Analyse-it Software Ltd. http://www.analyse-it.com/.
[5] I. Androutsopoulos, J. Koutsias, V. Chandrinos, D. Dpyropoulos, An experimental comparison of naive Baysian and keyword-based anti-spam filtering with personal email messages, in: Proc. of the 23rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2000, pp. 160–167.
[6] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos, P. Stamatopoulos, Learning to filter spam email: a comparison of a naïve Bayesian and a memory-based approach, in: Workshop on Machine Learning and Textual Information Access, 4th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD), Lyon, France, September, 2000.
[7] A.R. Barron, Predicted squared error: a criterion for automatic model selection, in: S.J. Farlow (Ed.), Self-organizing Methods in Modeling: GMDH Type Algorithms, Marcel-Dekker, New York, 1984, pp. 87–103.
[8] British Computer Society. http://www.bcs.org/server.php?show=conWebDoc.14617.
[9] J. Carpinter, R. Hunt, Tightening the net: a review of current and next generation spam filtering tools, Computers & Security 25 (2006) 566–578.
[10] X. Carreras, L. Marquez, Boosting trees for anti-spam email filtering, in: Proc. of Fourth Int. Conf. on Recent Advances in Natural Language Processing, Tzigov Chark, Bulgaria, September, 2001.
[11] J. Clark, I. Koprinska, J. Poon, A neural network based approach to automated email classification, in: Proc. of the IEEE/WIC Int. Conf. on Web Intelligence (WI'03), 2003.
[12] W.W. Cohen, Learning rules that classify email, in: Proc. of AAAI'96 Spring Sympos. on Machine Learning in Information Access, Stanford, California, April, 1996, pp. 18–25.
[13] H. Drucker, D. Wu, V.N. Vapnik, Support vector machines for spam categorization, IEEE Transactions on Neural Networks 10 (5) (1999) 1048–1054.
[14] E.M. El-Alfy, R.E. Abdel-Aal, Construction and analysis of educational tests using abductive machine learning, Computers & Education 51 (August (1)) (2008) 1–16.
[15] S.J. Farlow, The GMDH algorithm, in: S.J. Farlow (Ed.), Self-organizing Methods in Modeling: GMDH Type Algorithms, Marcel-Dekker, New York, 1984, pp. 1–24.
[16] T. Fawcett, An introduction to ROC analysis, Pattern Recognition Letters 27 (2006) 861–874.
[17] M.M. Fuad, D. Deb, M.S. Hossain, A trainable fuzzy spam detection system, in: Proc. of the 7th Int. Conf. on Computer and Information Technology, 2004.
[18] E. Georgioua, M.D. Dikaiakosa, A. Stassopoulou, On the properties of spam-advertised URL addresses, Journal of Network and Computer Applications 31 (4) (2008) 966–985.
[19] GFI Software, Why Bayesian filtering is the most effective anti-spam technology, White Paper (2007). http://www.gfi.com/.
[20] L.H. Gomes, C. Cazita, J.M. Almeida, V. Almeida, W. Meira Jr., Workload models of spam and legitimate emails, Performance Evaluation 64 (2007) 690–714.
[21] J. Gordillo, E. Conde, An HMM for detecting spam mail, Expert Systems with Applications 33 (2007) 667–682.
[22] R. Gordon, Z. Hongyuan, Exploring support vector machines and random forests for spam detection, in: Proc. of the First Conf. on Email and Anti-spam, Mountain View, CA, July, 2004.
[23] P. Graham, Better Bayesian filtering, in: Proc. of the First Annual Spam Conf., MIT Press, 2003.
[24] J.A. Hanley, B.J.A. McNeil, Method of comparing the areas under receiver operating characteristic curves derived from the same cases, Radiology 148 (1983) 839–843.
[25] W.-F. Hsiao, T.-M. Chang, An incremental cluster-based approach to spam filtering, Expert Systems with Applications 34 (3) (2008) 1599–1608.
[26] D. Jimenez, Dynamically weighted ensemble neural networks for classification, in: IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, 1998, pp. 753–756.
[27] J. Jung, E. Sit, An empirical study of spam traffic and the use of DNS black lists, in: Proc. of the Fourth ACM SIGCOMM Conf. on Internet Measurement, Taormina, Sicily, Italy, October, 2004.
[28] C.C. Lai, An empirical study of three machine learning methods for spam filtering, Knowledge-Based Systems 20 (2007) 249–254.
[29] B. Leiba, N. Borenstein, A multifaceted approach to spam reduction, in: Proc. of the First Conf. on Email and Anti-spam, Mountain View, CA, July, 2004.
[30] C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.
[31] G.J. Montgomery, K.C. Drake, Abductive reasoning networks, Neurocomputing 2 (3) (1991) 97–104.
[32] E.M. El-Alfy, F.S. Al-Qunaieer, A fuzzy similarity approach for automated spam filtering, in: Proc. of IEEE International Conf. on Computer Systems and Applications (AICCSA'08), Qatar, April, 2008.
[33] J. Provost, Naïve-Bayes vs. rule-learning in classification of email, The University of Texas at Austin, Department of Computer Sciences Rep. AI-TR-99-284 (1999).
[34] M. Sahami, S. Dumais, D. Heckerman, E.A. Horvitz, Bayesian approach to filtering junk email, in: Proc. of AAAI'98 Workshop on Learning for Text Categorization, Madison, WI, July, 1998, pp. 55–62.
[35] G. Sakkis, I. Androutsopoulos, G. Paliouras, A memory-based approach to anti-spam filtering, Information Retrieval 6 (2003) 49–73.
[36] K. Schneider, A comparison of event models for naïve Bayes anti-spam email filtering, in: Proc. of the 11th Conf. of the European Chapter of the Association for Computational Linguistics (EACL'03), Budapest, Hungary, April, 2003.
[37] S. Sinclair, Adapting Bayesian statistical spam filters to the server side, Journal of Computing Sciences in Colleges 19 (2004) 344–346.
[38] M. Su, M. Basu, Gating improves neural network performance, in: Proc. of the IEEE Int. Joint Conf. on Neural Networks, Washington, DC, USA, 2001, pp. 2159–2164.
[39] A. Swann, N. Allinson, Fast committee learning: preliminary results, Electronics Letters 34 (1998) 1408–1410.
[40] V. Tresp, Committee machines, in: Y.H. Hu, J.N. Hwang (Eds.), Handbook for Neural Network Signal Processing, CRC Press, 2001.
[41] D.C. Trudgian, Spam classification using nearest neighbour techniques, in: Proc. of the Fifth Int. Conf. on Intelligent Data Engineering and Automated Learning (IDEAL04), vol. 3177, UK, Lecture Notes in Computer Science (2004) 578–585.
[42] UCI Machine Learning Repository. http://mlearn.ics.uci.edu/databases/spambase/.
[43] C.C. Wang, Sender and receiver addresses as cues for anti-spam filtering, Journal of Research and Practice in Information Technology 36 (1) (2004) 3–7.
[44] E.M. El-Alfy, Learning methods for spam filtering, International Journal of Computer Research 16 (4) (2008).
[45] NeuroShell Classifier Software, Ward Systems Group, Inc. http://www.wardsystems.com/.
[46] M. Woitaszek, M. Shaaban, R. Czernikowski, Identifying junk electronic mail in Microsoft Outlook with a support vector machine, in: Sympos. on Applications and the Internet, Orlando, FL January, 2003.
[47] D.H. Wolpert, Stacked generalization, Neural Network 5 (1992) 241–260.
[48] Y. Yang, S. Elfayoumy, Anti-spam filtering using neural networks and Bayesian classifiers, in: Proc. of the IEEE Int. Sympos. on Computational Intelligence in Robotics and Automation, Jacksonville, FL, USA June, 2007.
[49] W. Zhao, Z. Zhang, An email classification model based on rough set theory, in: Proc. of the Int. Conf. on Active Media Technology, 2005.
[50] V. Zorkadisa, D.A. Karrasb, M. Panayotouc, Efficient information theoretic strategies for classifier combination, feature extraction and performance evaluation in improving false positives and false negatives for spam email filtering, Neural Networks 18 (2005) 799–807.