DOI: 10.15439/2014F498
ACSIS, Vol. 2

Feature Selection for Naive Bayesian Network Ensemble using Evolutionary Algorithms

Adam Zagorecki
Centre for Simulation and Analytics
Cranfield University
Defence Acedemy of the United Kingdom
Shrivenham, UK
Email: a.zagorecki@cranfield.ac.uk

***Abstract*—This document describes the winning method for the AAIA'14 Data Mining Competition: Key risk factors for Polish State Fire Service. The competition challenge was a feature selection problem for a set of three classifiers, each of them in a form of ensemble of naive Bayes classifiers. The method described in this paper uses a genetic algorithm approach to identify an optimal set of variables used by the classifiers. The optimal set of variables is found through a three-stage procedure that involves different settings for the genetic algorithm. The first step leads to reduction of attribute set under consideration from 11,582 to 200 attributes. The following two steps focus on finding an optimal solution by first exploring the solution space and then refining the best solution found in an earlier step.**

I. Introduction

THIS paper describes the winning method for the AAIA'14 Data Mining Competition: Key risk factors for Polish State Fire Service. The challenge was a feature selection problem for a set of three classifiers, each of them defined as ensemble of ten naive Bayes (NB) classifiers sharing the same set of features. The description of the competition and the task can be found at: http://challenge.mimuw.edu.pl/contest/view.php?id=83.

The method described here is based on genetic algorithm (GA) approach. I treated the challenge task as an optimisation problem, where the task was to find an optimal set of variables (it was divided into 10 sets of variables by the competition rules). The set was to optimise the objective function that was based on the score function defined for the competition. In my solutions, the objective function was slightly modified in order to avoid the overfitting phenomenon by using n-fold cross-validation (CV) in the process (with varying n).

The method described in this paper consisted of three different steps. For each step a different GA setup was used. The three steps can be summarised as follows:

- In the first step a small subset of *informative* variables from the original 11,852 variables was identified. For this task a single NB classifier was used rather than an ensemble of NBs.
- In the second step, solutions based on ensemble of NBs were identified, using only the informative variables subset identified in the first step.
- In the third step, the GA was used to improve the best solution obtained in the second step.

II. Genetic Algorithms

Genetic algorithms (GAs) introduced by Holland [1] are a category of evolutionary computation. Evolutionary computation is used to solve mathematical optimisation problems by means of heuristics, and therefore it can be viewed as a meta-heuristic optimisation algorithm. Evolutionary computation is concerned about developing algorithms and techniques that are inspired by the natural evolution. Concepts borrowed from the nature include generations, individuals and populations, genes, mating, natural selection, survival of the fittest, etc.

GAs are the most popular class of evolutionary algorithms [2] that uses *genes* and *chromosomes* to represent the individuals (which correspond to solutions). Genes are basically encoding of the solution by means of a string of numbers (typically binary, with possible other representations). New solutions are generated by means of combining typically pairs of individuals (existing solutions) form the population (working set of solutions). The combination mimics generic crossover with possible additional mutations. It has been believed that suitable chromosome representation can be critical to achieving satisfactory performance of the GA for the given problem. This premise was used in defining approach that I used in the competition.

III. The Challenge

The challenge was to predict three binary decision attributes based on a subset of 11,852 attributes. The challenge imposed a classifier model – for each of the three decision variables an ensemble of exactly 10 naive Bayes (NB) classifiers was to be used, with each NB having at least 3 attribute variables. The same set of NB classifiers was to be used for the three decision attributes.

The performance of the classifiers was determined by means of the receiver operating characteristic (ROC) curve. The goal was to maximise the average area under the ROC curve for the three classifiers. Additionally, a penalty for using a large number of attributes (beyond 30) was introduced. The penalty term p was defined as:

$$p = (\frac{n-30}{1000})^2$$

where n is the number of attributes in the solution. Fig. 1 visualises the effect of penalty as the function of the number

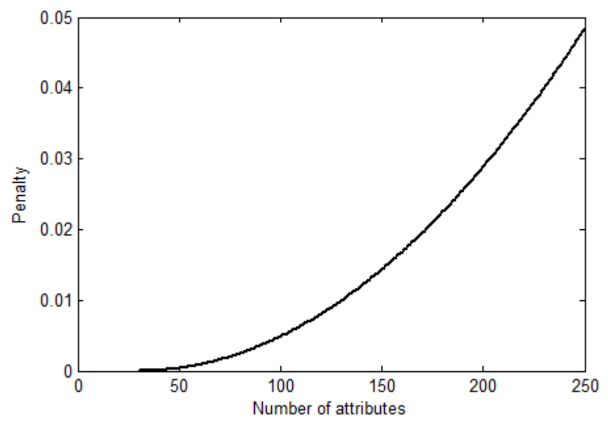

Fig. 1. Penalty as a function of the number of attributes included in the model

of attributes included in the ensemble. Taking into account that the area under the ROC curve should range between 0.5 and 1 and that the baseline solution provided by the competition organisers achieved the score of approximately 0.91, one should conclude that the number of attributes in the ensemble should not be greater than 200 to 300.

The data provided for competition was comprised of 11,582 variables and approximately 50,000 records. For each record a complete set of attributes is available. The distributions of values for the most of variables are heavily unbalanced and that applies to the decision variables as well.

IV. Cross-Validation

One of the basic challenges in data mining is overfitting [3]. To account for overfitting, I used k-fold cross-validation (CV) [4] embodied in the objective function. I used a simple k-fold CV schema, where data records were assigned to appropriate folds using modulo k operator and the original order of the records in the dataset. I used k-1 folds as a training set and the remaining fold as the test set, repeating it for each fold (round robin). The objective function was calculated based on the average of results for k folds.

I experimented with different values for k, as generally there is no widely accepted way to determine a *good* value of k. It is believed, that *good* values depend on the problem and particular dataset. In fact, I varied values for k for different steps. For the first step I used k=10, for the second step I used k=5, while for the third step I used k=3. While the change for the change between the first and second step was dictated just by the desire to avoid artefacts coming from the k selection, the reason for the change to k=3 was a result of simple investigation I decided to do after the second step: I evaluated performance of the same algorithm run with different CV settings: n=3,4,5,7, and 13 using the competition submission website that allowed to get feedback based on the evaluation set. I noticed that the results for n=3 and n=4 produced the same solution that scored better on the competition leaderboard than the solutions achieved using greater values of n. Therefore, for the final step I decided to use n=3.

V. The Method

In this section I will describe the method to find solutions to the competition challenge. The approach taken consisted of three steps that varied in the details of the experimental setup. The idea was to in the first step to identify attributes that can lead to good solutions. For the following steps I decided to use only a subset of attributes to improve the performance of the search heuristic.

My approach was dictated by two observations: (1) initial data analysis indicated that there are many variables that are in *present* state very rarely (and therefore arguably are not contributing much information or leading to overfitting) and (2) that initial experiments suggested that the *good* solutions quite consistently preferred certain variables over others. Based on those observations I decided first to narrow a subset of candidate attributes to 200.

In the three steps the same simple GA was used. However the chromosome and operator definitions, method of constructing initial population and the population size, and other parameters were varied throughout the steps. Below I provide the details of the algorithm for each step.

VI. The First Step – Identifying Informative Attributes

In the first step, the goal was to reduce an attribute set by identifying a subset of attributes that was *most informative*. In order to do that, I used a GA that would identify a NB (note: a single NB, not an ensemble of ten NBs) that otherwise would be optimising the problem as stated in the competition.

A. Chromosome Definition

Chromosome was defined as a list of integer values that were allowed to take values from 1 to 11,852. The length of chromosome was constrained to be at least 30, with upper limit set to 250 (but effectively solutions never exceeded 200 attributes anyway).

B. Crossover

As the crossover operator I used the uniform crossover method with mixing probability 0.5.

C. Mutation

Mutation had two operators, each of them applied with 0.5 probability:

- Remove – in that case the attribute would be removed from the list of attributes for that NB
- Replace – it would replace an attribute with a random attribute sampled from the uniform distribution. The addition of attributes was achieved by replacing an *empty* (denoted as 0) gene with a value related to one of the attributes (denoted as an integer 1 to 11,852).

The probability of mutation was set initially to the value 0.01 and was designed to decline after each generation. After each iteration (generation), the probability of mutation was multiplied by a constant delta equal to 0.999. The parameters were set in such a way that a mutation would occur for a new individual with probability over 0.9.

D. Initial Population

Initial population consisted of 200 individuals. The initial number of attributes for individuals from the initial population was sampled from the uniform distribution with lower and upper limit 30 and 130 correspondingly. The attributes were sampled from the complete set of 11,582 attributes using the uniform distribution. The size of the initial population of 200 individuals was determined based on the size of the problem. The total number of possible attributes in the population (200 multiplied by 80, where 80 is an average chromosome size in the initial population) would exceed the total number of attributes in the data (11,852).

E. Simulations

In order to identify the *informative* attributes I run 200 independent simulations and collected the best result (an individual with the highest score at the end of simulations) for each of the 200 simulations. Each simulation terminated after 2500 generations. For each generation I generated 50 new individuals (25%) and subsequently rejected 50 individuals with the lowest score.

To perform simulations I used my own implementation of the algorithm written in C++ programming language. The implementation was intended to provide optimised code for the task. Rather than using the raw data, I used pre-calculated conditional probability tables for each attribute (given decision attribute and CV fold).

F. Result

As the result of simulations I obtained a set of 200 solutions. Please note that those solutions assumed a single NB model, rather than an ensemble of 10 NB models. To evaluate those solutions using the competition leaderboard I used a simple technique: I assumed that the first 27 attributes will be assigned for the 9 NBs in the ensemble with each contacting only 3 attributes (the minimum required). The remaining attributes were assumed to belong to the last NB in the ensemble. Additionally I sorted the attributes for the evaluation purpose. Solutions obtained using this method resulted in scores in the range of 0.93 to 0.945 on the competition leaderboard.

But the most valuable result for this step was the set of *informative* variables. It turned out that the 200 solutions shared a lot of common attributes, suggesting that there were clearly attributes that were more informative than the others.

The attributes regarded *informative* as were (sorted according to the ranking): 5270, 143, 2887, 8179, 10062, 460, 2924, 8914, 258, 2182, 7187, 7980, 5306, 1880, 7999, 11266, 1509, 11463, 7299, 5909, 3273, 5835, 1244, 72, 142, 8985, 6961, 10114, 6660, 2835, 7055, 8959, 304, 7148, 8039, 8107, 6779, 10880, 5446, 11359, 3294, 3492, 3951, 4323, 11459, 1335, 2684, 401, 4347, 3257, 7755, 8990, 675, 5519, 1772, 6949, 2883, 6196, 766, 6238, 9846, 6534, 5990, 5996, 7425, 10446, 1111, 3598, 3735, 8110, 8114, 9873, 691, 759, 10902, 2380, 3324, 4415, 8249, 40, 2971, 4959, 9926, 8972, 2635, 4962, 11802, 8772, 3290, 5572, 6163, 8529, 3402, 9715, 10019, 27, 6430, 7172, 8216, 8794, 9657, 9771, 1161, 7243, 7520, 10771, 271, 1540, 4153, 4469, 5655, 8034, 8853, 11196, 3166, 5229, 5350, 8159, 9848, 11701, 73, 799, 3890, 5861, 6474, 6653, 480, 2966, 3706, 6388, 11217, 11231, 327, 896, 3479, 5393, 5778, 9725, 8397, 648, 840, 1954, 2538, 7158, 10300, 11604, 833, 2697, 4594, 7046, 8314, 10740, 11088, 2273, 2402, 2868, 4137, 6162, 7304, 8553, 10014, 11483, 2539, 5262, 6750, 10638, 11421, 135, 220, 1307, 3751, 4443, 6222, 6498, 6507, 7103, 7414, 8093, 8663, 9055, 11594, 282, 1289, 1767, 2568, 4198, 5041, 5143, 5859, 6579, 6602, 7069, 7356, 8807, 11587, 11636, 364, 426, 958, 2218. I arbitrarily decided to use 200 attributes for the further experiments. The 200th attribute was close to 0.05 probability of occurring in the solution.

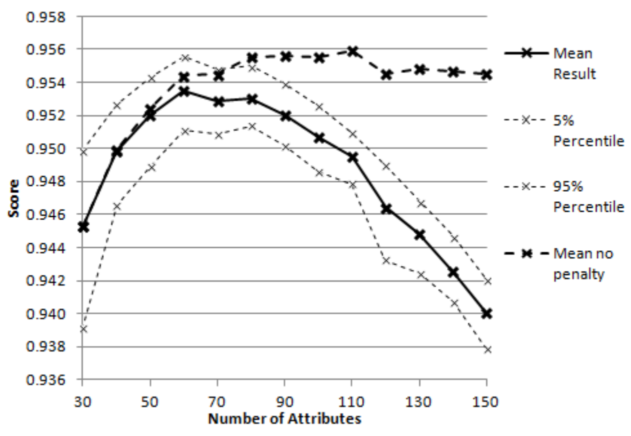

Fig. 2. Performance as a function of the number of attributes

As well, at this step I determined that the optimal number of parameters ranged between 60 and 80 (assuming single NB, not an ensemble). It should be noted that the definition of the chromosome and the simulation allowed the algorithm for adjusting the number of attributes in the solution, and therefore I did not need to be concerned about specifying the right number of the attributes in the solution.

In fact I run the experiments for which I used constant chromosome size, in order to ensure that the results I get are consistent with the fixed chromosome size experiments. Those experiments shed some light on the problem of the optimal number of attributes in the solution. The results are shown in Fig. 2. This analysis conformed that the optimal number of attributes should be approximately in range of 60 to 80 attributes if the penalty term is included.

VII. The Second Step – Finding Optimal Ensemble

In the second step, the task was to come up with a *good* solution for NB ensemble.

A. Chromosome Definition

For this step, the chromosome encoded the structure of the classifier ensemble, as stated in the competition requirements. The chromosome was defined as a vector of 10 lists of integers, where each list corresponded to one NB in the ensemble. The list was constrained to have at least three elements, with no upper limit on the number of elements.

B. Crossover

For both crossover and mutation, I treated each list of 10 chromosomes as individual chromosomes, applying crossover to i-th chromosome from the first individual with the i-th chromosome form the other individual. I indeed tried to crossover i-th chromosome from the first individual with j-th chromosome from the second individual, but performance of such algorithm was inferior to the one used, therefore I rejected the idea.

This time I used single point cross-over with the point randomly generated from the uniform distribution.

C. Mutation

The probability of mutation for a single NB classifier definition (repeated for each list in the vector) was set initially to be 0.05, and it was decreased at each generation with factor of 0.999. The mutation algorithm used one of three mutation operators applied with different probabilities:

- Add – adding a new element to the list (sampled from 200 attributes using uniform distribution), applied with probability 25%
- Remove – remove an attribute from the list (if there are more than 3 attributes in the list), applied with probability 25%
- Replace – replace an existing attribute with a randomly selected attribute (sampled from 200 attributes using uniform distribution), applied with probability 50%

D. Initial Population

Initial population consisted of 100 individuals. This time the chromosomes were randomly initialised using 200 attributes identified in the first step. The initial chromosome size was assumed to be 3 for all individual lists within a chromosome (hence started with 30 attributes in each solution).

E. Simulations

Each simulation included 5000 generations. For each generation I generated 25 new individuals (25%) and rejected 25 individuals with the lowest score. I run over 50 of simulations and collected the final best results for each of the simulation runs.

F. Results

The best solution identified at this step allowed me to achieve a score of 0.9512 at the competition leaderboard.

VIII. Step Three – Improving the Score

The final step was inspired by the idea of using evolutionary algorithms that alter the initial solution in order to improve it. But rather than using any specific approach, I decided to us the same GA framework I used previously, with the only difference that I decided to focus on the mutation operator as the search driving mechanism. The basic setup was the same as the algorithm in the second step, with some changes described below.

A. Chromosome Definition and Corssover

Chromosome definition and crossover operators were exactly the same as in the second step.

B. Mutation

The mutation operator was exactly the same as in the second step. The only difference was that the initial probability of mutation for each list element (individual NB in an ensemble) was increased to 0.15 to induce more mutations.

C. Initial Population

The key change was the initial population – this time I used seeding. I decided to use initial population that comprised of 50 copies of the same individual – the one that was achieving the highest score in the second step.

D. Simulations

Each simulation had 5000 generations, but in fact I used several simulations:

- I terminated the first simulation at around 500 generations, and I achieved the leaderboard score of 0.9522.
- I used the best result achieved so far as the seed for the initial population, and this time I allowed to run the complete 5000 generations. That allowed me to achieve the leaderboard result of 0.9561.
- Consequently, I used the best result so far to repeat the procedure. This time I decided to use several parallel runs with the same seeding. Most of the results were inferior to the initial (seeding) solution (0.9561) with most of leaderboard submissions ranging between 0.954 and 0.946. One outlier was the solution with the leaderboard result 0.9583 that I used at the final submission.

The solution that I used for the final submission was as follows:

- 11463, 11088, 8179, 6498, 2883, 460, 143, 6388, 2924
- 10880, 4415, 1880, 258, 2966, 5926, 6491
- 10446, 9771, 8249, 7980, 7187, 5270, 401, 5990, 8039, 8959, 8093, 3890
- 833, 72, 3751, 1266
- 8914, 7999, 6474, 5041, 4962, 10019, 947
- 10902, 5996, 2835, 748
- 8034, 7055, 6961, 5909, 1244, 27, 3324, 7172, 270
- 11701, 8110, 7356, 7187, 6222, 6162, 7158, 9873, 7560
- 11266, 10062, 220, 11231, 2971, 7521
- 11459, 7148, 7755, 3951, 2887, 766.

IX. Conclusion

In this paper I presented the description of the method used to optimise the set of variables used for classifiers based on ensembles of NB. I used the same GA framework, however in three different ways to address the challenge problem:

- First, I used GA to reduce the number of attributes in the problem in order to improve performance of the consequent simulations
- Then I used GA to find a set of optimal solutions to the competition problem,
- Finally, I used GA with focus on mutation to refine the solutions obtained from the previous step.

I would like to emphasise that I strongly believe that the winning solution I obtained can be improved. Similar applies to the algorithms used – they can be refined and improved in terms of convergence efficiency, chromosome encoding, etc.

Acknowledgment

I would like to thank my wife, Katarzyna Holownia, for encouragement and support during the competition.

References

[1] J. H. Holland, "Adaptation in natural and artificial systems", Ann Arbor: The University of Michigan Press, 1975
[2] M. Mitchell, "An Introduction to Genetic Algorithms", MIT Press, 1998
[3] T. Dietterich, Overfitting and undercomputing in machine learning, ACM Comput. Surv. 27, (3), 326-327, 1995
[4] S. Geisser, "The predictive sample reuse method with applications", J. Amer. Statist. Assoc., 70:320–328, 1975