

CS 445/545
Machine Learning
Winter, 2010

Homework 6:
Genetic Algorithms for Feature Selection

Due Monday March 15.

In this homework assignment, you will use a genetic algorithm to do feature selection for a Naive Bayes algorithm on the UCI “Spam” dataset. You may either modify the “Simple GA in C” from the class web page:

<http://www.cs.pdx.edu/~mm/MachineLearningWinter2010/simple-ga-in-c.tar.gz>

or if you are very ambitious (and have a lot of free time), you may write your own GA code. In this writeup, I will assume you are modifying the Simple GA in C code.

Un-gzip the tar file, extract the files, and follow the instructions in README.pdf to run the program.

This code will run on Linux machines. If you want to run it on other platforms, you may need to change the random number generation code.

If you have any questions about this code (or if you are not familiar with C), feel free to e-mail or come see myself or the TA, Mick Thomure.

Genetic algorithm for feature selection

You will be writing a fitness function for the genetic algorithm to perform feature selection for your Naive Bayes algorithm as applied to the spam data on the UCI Machine Learning Repository. The data is at

<ftp://ftp.ics.uci.edu/pub/machine-learning-databases/spambase/>

Divide this data into a training set, a validation set, and a test set. The validation and test sets should each have approximately the same ratio of spam and ham examples as the training set.

Each example has 57 features, as described in the “spambase.names” file.

Your GA will evolve bit strings of length 57, where a 1 means “use the corresponding feature” and the 0 means “don’t use the corresponding feature”.

Suppose you have a training set `spambase.train`, a validation set `spambase.val`, and a test set `spambase.test`. The fitness of individual c is calculated as follows:

1. Train your naive Bayes classifier using `spambase.train`, but only use the attributes whose corresponding value in c is 1.
2. Run the resulting naive Bayes classifier on `spambase.val`. The fitness of c is the accuracy of the naive Bayes classifier on the validation set (i.e., a fraction between 0 and 1).

Don't use the test set until the "testing step" (step 5) below.

What you need to do

1. Starting with the Simple GA in C code, modify the `fitness.c` file to implement the fitness function described above.
2. Modify the "params" file to use the following parameters:

```
NUM_RUNS 10
FITNESS_FUNCTION_NAME "feature selection"
MAX_NUM_FUNCTION_EVALS 1000      # Maximum of 50 generations
POP_SIZE 20
CHROM_LENGTH 57
FITNESS_FUNCTION_OPTIMUM_KNOWN TRUE
FITNESS_FUNCTION_OPTIMUM 1.0
RUN_NUM_DIR "<fill in pathname here>"
OUTPUT_DIR "<fill in pathname here>"
LONG_PRINT TRUE
```

All other parameters can stay the same as in the original params file.

4. Running the GA: Run the GA (it will do 10 independent runs, as indicated in the params file). For each run, record the best fitness found in the run, and the individual that produced that fitness. (If there is more than one best individual, choose one of them arbitrarily to record.)

5. Testing the best feature sets For each of your 10 best individuals (from the 10 different runs), run the corresponding naive Bayes classifier on the test set, `spam.test`. Record the number of attributes used and the classification accuracy of each on the test set. Compare with the classification accuracy on the test set of the naive Bayes classifier using all of the 57 attributes.

Give the recorded information from steps 4 and 5 in table form in your writeup, and write a one-paragraph discussion.

6. Which GA parameters do you think could be modified to improve your results? Design and run an experiment (10 runs of the GA) with these modified parameters. Describe your experiment, why you thought modification of those parameters might help, and the results.
7. *This step is required for graduate students, optional for undergrads.* Devise a version of the GA (by changing either the algorithm or the fitness function) that will penalize individuals for using too many features. Write a description of your new method, and repeat the experiments in steps 4-6 above using your new method. As above, write a one-paragraph discussion of your results.

Here is what you need to turn in: an electronic version of your writeup as described above, along with your (well-commented) `fitness.c` file. E-mail this to thomure@cs.pdx.edu.