AIML426P1Q4

Chad Kakau

2024-09-05

AIML426 Project 1 Q4 : Non-dominated Sorting Genetic Algorithm-II (NSGA-II)

Problem description

This problem replicates the problem from question two, where we want to reduce the amount of features used to fit a classification model. In this genetic algorithm we will use two objectives:
1. minimise the classification error rate
2. minimise the ratio of selected features

This experiment uses a slightly modified programme from that used in Question 2 (GA-Feature selection), which itself was a slight modification of the example script provided at the deap website.

Method Description

overall process

This genetic programme follows a generic process:

```
__Initialise__ (population
    create Fitness_class
      min_class:  classification_error
      min_feat:  (m=count_selected_features)/(n=count_features)
    create Individual_class
      inherit Fitness_class
    individual representation
      define chromosomes:
        'expr': boolean
      define individual:
        'indiv': 'expr'*n
    initialise population:
      'pop': initialise_'indiv' * pop_size=n
__Repeat__ until __Stop__
__Evaluate__ individuals
    classification_error:  MSE
    feat_ratio: m/n
__Crossover__
    parents_selected(prob=0.6)
      cxOnePoint(parent1, parent2)
        bit_selected(prob=0.4)
```

```
__Mutate__
    individual_selected(prob=0.4)
      mutFlipBit(indiv)
      bit_selected(prob=0.4)
__Selection (Tournament)__ offspring
__if__ stopping_criteria or max_generations=50:
  __Stop__
__else__ go_to __Evaluate__
```

The process takes a data set and performs the above alogrithm to determine an individual that comprises a reduced set of features from the initial data set. These selected features balance classification error against number of features to get the best classification for the least features.

The algorithm creates a fitness class, with two minimising objectives. It then creates an individual class that inherits from the fitness class. An Individual is represented as x, an n-length list of booleans, that indicate a feature at bit i is included $x_i = 1$ or excluded $x = 0$. The population comprises n individuals in the first generation.

Once the population is generated then individuals have fitness evaluated, by checking:
- the training error of the selected features using a classification model from sklearn - the ratio of selected features to total features.

Crossover of parent pairs occurs with probability of 0.6, and the cxOnePoint crossover function selects an individual bit as the crossover point with probability of 0.4. Mutation of individuals occurs with probability of 0.4 and the mutFlipBit function selects an individual bit for flipping $(0 \rightarrow 1)$ or $(1 \rightarrow 0)$ with probability 0.4.

I chose cxOnePoint crossover because it is reliable and straight-forward to understand. I chose 0.6 probability of crossover because I am more interested in transferring between parents than in mutating individuals, hence the complementary probability of mutation as 0.4. For both crossover and mutation, I used 0.4 as the probability an individual bit being selected so that the evolutions aren't too erratic or extreme between generations. I used the Tournament selection method, rather than elitism, to maintain some diversity in individuals while still maintining the likelihood of evolving good individuals.

I used the eaMuPlusLambda algorithm because it gives the broadest selection base for each generation.

I reduced the number of generations for each experiment from question 2, because it just took too long - a direct result of fitting and evaluating full models on each individual in the population. After spending ages waiting for evaluations during question two, I considered implementing a memory function within the programme that retains an individual's evaluated fitness and if an identical individual is encountered in the future, that previously evaluated fitness is used rather than fitting the model again. But, I never managed to implement.