Workshop

Day 1: Introduction to artificial intelligence and optimisation

Functional Programming and Intelligent Algorithms:
Genetic Algorithms
Spring 2015
Faculty of Engineering and Natural Sciences
Aalesund University College

Robin T. Bye*

Monday 16 February 2015

*email: roby@hials.no; website: www.robinbye.com; phone: 70 16 15 49; room: B314.

Contents

1 Workshop overview 2
1.1 Topics . . . 2
1.2 Reading material . . . 2
1.3 Specific learning outcomes . . . 2
1.4 Schedule . . . 3

2 Exercises 3
2.1 Introduction to AI and optimisation . . . 3
2.2 Minimum-seeking algorithms . . . 4

3 Homework 6

1 Workshop overview

1.1 Topics

Today's topics include:

- Introduction to artificial intelligence (AI)
- Introduction to optimisation
- Minimum-seeking algorithms

1.2 Reading material

Compulsory reading to be studied *before* this workshop is Chapter 1 in Haupt & Haupt (2004) and Chapter 9 in Marsland (2015).

Supplementary reading include Chapter 9.5 in Negnevitsky (2005), Chapter 1 in Goldberg (1989), and Part I in Russell & Norvig (2010).

1.3 Specific learning outcomes

After completing this workshop, including self-study, reading and exercises, the students should be able to

- provide a proper definition of artificial intelligence (AI) and list several topics and tools in AI.

- explain what is meant by optimisation both in general terms and from a systems point of view (inputs → system → output) and provide examples of *absolute* and *relative* optimisation.
- find exact global optima of simple analytical functions such as the parabola using calculus (analytical methods such as finding derivatives) and plot the same functions.
- write small Haskell programmes able to implement mathematical functions as well as simple minimum-seeking algorithms for optimising such functions.
- categorise problems as solvable analytically or not and give examples of methods that may be used in each case.
- explain and exemplify common terminology and methodology from the field of optimisation and computational intelligence.
- recognise the problem of getting stuck in local optima and suggest methods for avoiding this problem.

1.4 Schedule

We begin at 8.15 with a recap of the last couple of weeks' activities and questions. Today's workshop will roughly follow the schedule below:

08.15 Status update/recap.

08.45 Introduction to AI and optimisation.

10.15 Workshop rest of the day.

2 Exercises

2.1 Introduction to AI and optimisation

Exercise 2.1: Explain what is meant by AI.

Exercise 2.2: Do a literature search and find a real-world problem that has been solved using AI techniques. Write a half- to one-page summary of the problem justifying *why* the solution belongs to the field of AI.

Exercise 2.3: Explain what is meant by optimisation.

Exercise 2.4: Consider optimisation of the functions

$$f_1 = |x| + \cos x, \qquad -\infty \leq x \leq \infty \tag{2.1}$$
$$f_6 = (x^2 + x)\cos x \qquad -10 \leq x \leq 10 \tag{2.2}$$
$$f_7 = x \sin 4x + 1.1y \sin 2y, \qquad 0 \leq x, y \leq 10 \tag{2.3}$$

For each function, is the optimisation problem

(a) constrained or unconstrained?

(b) single-variable or multivariable? Give the number of dimensions of the problem.

(c) static or dynamic?

(d) discrete or continuous?

(e) solvable analytically (using calculus and derivative methods)?

Provide explanations to all your answers.

Exercise 2.5: Two functions are given by

$$g(z) = -(z + 3)^2 \tag{2.4}$$
$$h(y) = (y + 1)^2 - 2 \tag{2.5}$$
$$\tag{2.6}$$

For each function, answer the following:

(a) Explain why the function can be optimised analytically.

(b) Use calculus to find the optimum (ignore values at infinity) and determine if it is a maximum or a minimum.

(c) Draw a diagram by hand and indicate the optimum.

Exercise 2.6: Consider the function f_2 given by

$$f_2 = |x| + \sin x \qquad -20 \leq x \leq 20 \tag{2.7}$$

Implement the function f_2 and the other functions f_1, f_6, and f_7 defined previously as Haskell functions. Test your functions in the interpreter.

2.2 Minimum-seeking algorithms

Exercise 2.7: You have written a computer program that finds the *minimum* of a given cost function. One day you are asked to write another computer program that finds the *maximum* a given fitness function. How can you easily modify your initial program to maximise a fitness function instead of minimising a cost function? Give an example of a cost function and convert it to a fitness function.

Exercise 2.8: Investigate the Nelder-Mead local minimisation algorithm by performing a search on the Internet. Is the algorithm able to find a global minimum in each case (within the given constraints)? How does the choice of initial guess affect your results?

Exercise 2.9: Write your own simple version of a local minimiser as a function `fminsimple` that given a single-variable cost function `costfunction` starts at an initial guess `x0` in the search space and moves in fixed moves with an increment of `stepsize` in a downhill direction until it cannot move downwards anymore. You should also include lower and upper boundaries, `l` and `u`, respectively.

The minimiser should return the found input variable value, `xmin`, and the minimum function value evaluated at `xmin`, namely `costfunction xmin`, for example in a tuple.

A possible function signature is given by

```
fminsimple :: (Double -> Double) -- costfunction
           -> Double             -- x0
           -> Double             -- stepsize
           -> Double             -- lower boundary l
           -> Double             -- upper boundary u
           -> (Double,           -- xmin (optimal value of x)
               Double)           -- costfunction xmin (minimum cost)
fminsimple costfunction x0 stepsize l u = -- <... your code here ...>
```

Test your local minimiser function on some one-dimensional cost functions such as those presented above, which can be found with plots and solutions in Appendix I of Haupt & Haupt (2004). Observe when the minimiser succeeds and when it fails in finding the global minimum.

Exercise 2.10: Expand your function to also display the number of iterations to complete the search for a minimum. A possible function signature is given by

```
fminsimple' :: (Double -> Double)   -- costfunction
            -> Double               -- x0
            -> Double               -- stepsize
            -> Double               -- lower boundary l
            -> Double               -- upper boundary u
            -> Int                  -- counter
            -> (Double,             -- xmin
                Double,             -- costfunction xmin
                Int)                -- counter + 1
fminsimple' costfunction x0 stepsize l u c = -- <... your code here ...>
```

How does choice of initial guess `x0` and the stepsize affect the number of iterations?

Exercise 2.11: Both the Nelder-Mead algorithm and your homemade function `fminsimple` can get stuck in local minima. Suggest how you could make your algorithm above more sophisticated so that there is a greater chance of finding a global minimum.

3 Homework

- Complete all the exercises above.
- Read through (again!) the specific learning outcomes in Section 1.3 to check which outcomes you have not attained yet. Study today's material and prepare questions for tomorrow about learning outcomes you have missed.
- Skim through the lecture notes for tomorrow's lecture and the suggested literature.

References

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley Longman, Inc.

Haupt, R. L., & Haupt, S. E. (2004). *Practical Genetic Algorithms*. Wiley, 2nd ed.

Marsland, S. (2015). *Machine learning: an algorithmic perspective*. CRC press, 2nd ed.

Negnevitsky, M. (2005). *Artificial Intelligence: A Guide to Intelligent Systems*. Addison-Wesley, 2nd ed.

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Pearson, 3rd ed.