Functional Programming and Intelligent Algorithms

Introduction and Overview

Prof Hans Georg Schaathun

Høgskolen i Ålesund

18th January 2016

Outline

Overview and Motivation

The Module

Functional Programming

Why the title of this course?

Functional Programming is our toolbox

— start from scratch
— prior experience with other programming paradigms is a double-edged sword

Intelligent Algorithms are our solution techniques

1. Genetic Algorithms (GA)
2. Artificial Neural Networks (ANN)

Two types of problems

Optimisation Problems

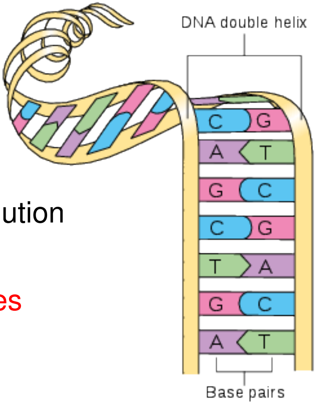
1. $\max_x f(x)$
2. $\min_x f(x)$

Image from NASA via Wikipedia. No copyright protection.

Classification Problems

1. Is this photo computer generated?
2. Does this fingerprint belong to Person X?
3. Whose face does this image show?

Genetic Algorithms

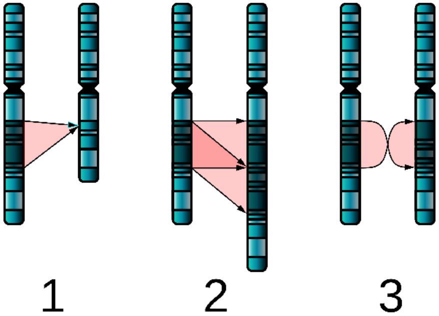


Genetic algorithms mimic biological evolution

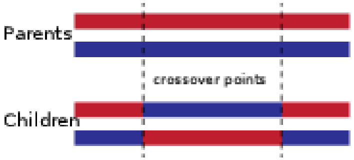
The candidate solution is a chromosomes

Image by Cancer Research UK [CC BY-SA 4.0], via Wikimedia Commons

Evolution

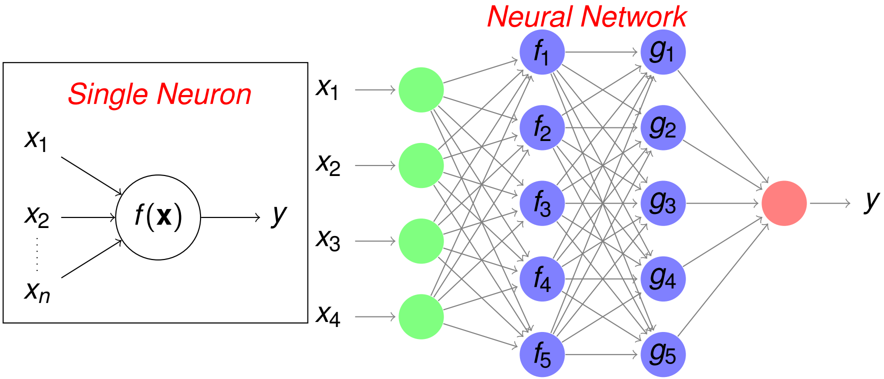


"Single Chromosome Mutations" by Richard Wheeler (Zephyris) Licensed under CC BY-SA 3.0 via Commons



By R0oland (Own work) [CC BY-SA 3.0], via Wikimedia Commons

Neural Networks



Commonly used for classification problems.

Outline

Overview and Motivation

The Module

Functional Programming

Topics and teachers

- Functional programming: Hans Georg Schaathun
- Neural networks: Que Tran/Hans Georg Schaathun
- Genetic algorithms: Robin T. Bye

The module

1. 100% written exam
2. Learning by doing
   - practical and theoretical exercises
   - collaborative work — individual learning
   - problem based learning
3. Memorising is essential
4. $7\frac{1}{2}$ credit = $187\frac{1}{2}$h–225h
   - i.e. approx. 10h+ days 5 days a week

Learning approach

1. Iterative approach
2. Start with practical tutorials
   - require a large part of the curriculum
   - get started – get it working
   - look up theory and solutions as required
3. Revisit theoretical material later
   - reflect over previous solutions
   - why does it work?
   - can it be done better?
4. Move on to more complex and advanced exercises
   - search for more theory as required
5. Reflect again. What more do we need to learn

... and so it goes on ...

Learning material

- `http://www.hg.schaathun.net/FPIA/`
- Backup: `http://kerckhoffs.schaathun.net/FPIA/`
- Piazza for Q&A:
  `https://piazza.com/hials.no/spring2015/ie501614/home`

Books on Functional Programming

1. Simon Thompson: Haskell. The craft of functional programing. Third edition 2011. (core)
2. Bryan O'Sullivan, John Goerzen and Don Stewart: Real World in Haskell
3. Miran Lipovača: Learn You a Haskell for Great Good!
4. Simon Marlow: Parallel and Concurrent Programming in Haskell

Books on Intelligent Algorithms

1. Stephen Marsland: *Machine Learning. An Algorithmic Perspective.* (core)
2. Stuart Russell and Peter Norvig: *Artificial Intelligence: A Modern Approach* (Third edition) (elective on AI next semester)
3. Randy L. Haupt & Sue Ellen Haupt: *Practical Genetic Algorithms.*

1. Reading list
   `http://www.hg.schaathun.net/FPIA/reading.php`
   - secondary textbooks
   - research papers
   - on-line references

Outline

Overview and Motivation

The Module

Functional Programming

Functional Programming

Exercise

Get familiar with the necessary software on your computer system.

1. *set up the Haskell platform*
2. *get familiar with a suitable editor*
3. *get familiar with the interpreter*
4. *write and test your first simple program*
5. *signup and test out Piazza by submitting a question and answering someone else's question*

Haskell is declarative language

- The programme is a list of definitions
- We can define constants
  1. `r = 10`
- We can use expressions in definitions
  1. `a = pi*r^2`
- Definitions are made once and for all
  1. symbols can be used before their definitions
  2. definitions cannot change
  3. there are no variables and no assignment
- The heart of the programme is some expression to be evaluated

Haskell is strongly typed

— All objects have types
- `r :: Double`
- `r = 10`
- `a :: Double`
- `a = pi*r^2`

— Haskell is very picky about the type.

— You will explore a few types in the tutorial
1. Boolean
2. integer types: Int and Integer
3. floating point: Double

Haskell is functional language

- Functions are first class objects
- defined like a constant, with a function type
  - `area ::  Double -> Double`
  - `area r = pi*r^2`
- used to form new expressions
  - `myArea ::  Double`
  - `myArea = area 10`
- Note that we do not need parenthesis to mark function calls
- Functions behave in any way like other objects
  - can be passed as arguments to other functions

Haskell has no side effects

- An expression, say **`area 10`**, always has the same value
  - it can be replaced with the result (314.1592653589793) every time (referential transparency)
- Referential transparency is not common in programming
  - **`getchar`** in C has a new return value every time
    - returns a character read from the keyboard
  - functions could use or change global variables
    - Haskell has no variables
- Haskell functions only communicate via input and output arguments
  - i.e. no side effects

What is Functional Programming?

- Not well-defined
  - many hybrid languages
  - functional ideas in combination with other paradigms
- Functional programming does not have to be as pure as it is in Haskell

Common features of a functional programming style are:

1. Functions are first-class objects
   - used as parameters to other functions
   - used as data structures
2. Focus on evaluation of expression (incl. functions)
   - rather than execution of operations

Haskell

Declarative
: The program is a list of definitions which cannot be changed.

Lazy and non-strict
: A value is only calculated if it is used.

Functional
: Functions are first-class objects

Pure

- No side effects.
- No state.
- Referential transparency.

The tools

The editor	to write your definitions. We do demos with `gedit` and `vim`. You use whatever you like.
The interpreter	to evaluate expressions. We use `ghci` on the command line.
Compiler	to compile standalone programs. We will be using `ghc` later.
The Haskell Platform	(includes the interpreter and compiler) Includes libraries, package manager, etc.

Operating system

— The Haskell Platform is available on many platforms
  - Linux and FreeBSD (we can help and supervise)
  - Mac OS X (difficult, but we will try)
  - MS-Windows (you are on your own)
  - can be compiled from source (George might be able to help on Unix)

— Editor is separate

VirtualBox

1. Download and install virtualbox
2. Download the fpia VM from `http://www.hg.schaathun.net/FPIA/fpiavm.zip`
3. Unzip the archive
4. Start virtualbox and import the VM.
5. Username `fpia`; password `NeuralNetwork`

Programme today

1. Lecture
2. now: Tutorial 1
3. Lunch at will
4. 12.15: Recap of Tutorial 1 + new lecture
5. Tutorials 2-4
6. Tomorrow 8.15: Recap of Tutorial 2 + new lecture

Exercise

Learning material on the web pages:

1. *Tutorial exercises. Do the exercises.*
2. *Short demo videos. Watch them when you need them.*