COMP226: Slides 19

Parameter optimization

Rahul Savani

rahul.savani@liverpool.ac.uk

Overview

- choosing **parameter ranges**
- **optimization** of parameters by **grid search** (brute force)
- **visualizing** backtest results (for two-parameter strategies)
- **other optimization methods**

What are good results?

Need an **optimization criterion** (**ordering** of trading results)

Achieved via a **fitness function**

- Maps trading results to fitness (usually a single number)
- Higher fitness means better trading strategy results

The simplest fitness function is just **profit** or **return**

We have already seen other examples, such as the **Sharpe ratio** etc.

Optimization approaches

The are many relevant optimization approaches such as:

- **Grid search** (exhaustive, or brute forece), i.e., we define allowable parameter values for each parameter individually, and then **look at all combinations**

- **Evolutionary algorithms (e.g. genetic programming)**

In these slides we mainly focus on grid search (so consequently we can't have too many parameter combinations)

At the end of the slides we discuss other approaches

Parameter ranges for grid search

We focus on **regularly-spaced sequences** as returned by the `seq` function in R

We need to decide on:

- **Endpoints** (the `from` and `to` arguments to `seq`)
- **Increments** (either via the `by` or `lenght.out` argument to `seq`)

The choice of endpoints

Some parameters have natural upper or lower **bounds**, e.g.

The size of a moving average window is

- lower bounded by 0
- upper bounded by the length of historical data available

Beyond some value a parameter may not effect trading results

- multiple of standard deviation for BBands

Heuristic for picking increments

Suppose you **have already decided endpoints**

1. Estimate backtest time for one parameter combination
2. Based on how long you have to run the backtest determine the number of parameter combinations to test
3. Test the same number of parameter values for each parameter to achieve a total close to the target

Example: picking increments

- Suppose you have three parameters:

$$p_1,\ p_2,\ p_3$$

- All three can take on any rational value
- You have decided on the following endpoints:

$$p_1 \in [0, 5]$$

$$p_2 \in [10, 20]$$

$$p_3 \in [-10, 10]$$

Example

- Use 2x2x2=8 combinations of endpoints to **estimate iteration time**
- Suppose this indicates average iteration time is 20 seconds
- **Target time** for backtest 8 hours

$$\rightarrow 8 \times 60 \times 60 / 20 = 1440$$

- Each would be allowed

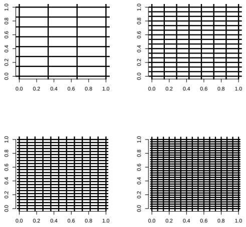
$$\sqrt[3]{1440} = 11.292\ldots$$

- So we would have 11 values for each parameter

```
> p1 <- seq(from=0,to=5,length.out=11)
> p2 <- seq(from=10,to=20,length.out=11)
> p3 <- seq(from=-10,to=10,length.out=11)

> p1
 [1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> p2
 [1] 10 11 12 13 14 15 16 17 18 19 20
> p3
 [1] -10  -8  -6  -4  -2   0   2   4   6   8  10

> nrow(expand.grid(p1=p1,p2=p2,p3=p3))
[1] 1331

> 11*11*11
[1] 1331
```

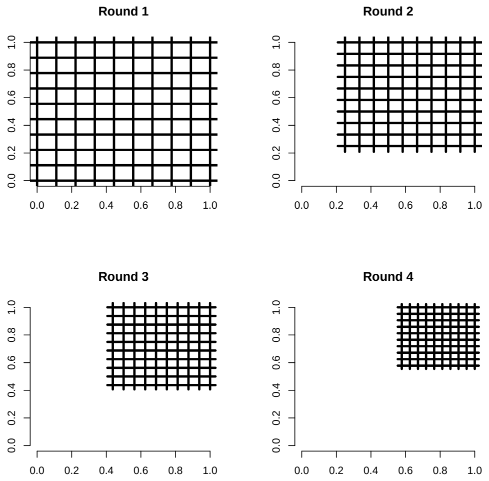
Grid search

- Evaluate **all parameter combinations**
- For two parameters this is a 2d grid, in general a **hypergrid**
- With **n** parameters, a **k**-fold increase in number of parameter values for each results in a **k^n**-fold increase in the number of parameter combinations



Iterative approach

1. Start with a **coarse grid**
2. Pick promising region and repeat with corresponding
    - **smaller ranges** and
    - **finer increments**

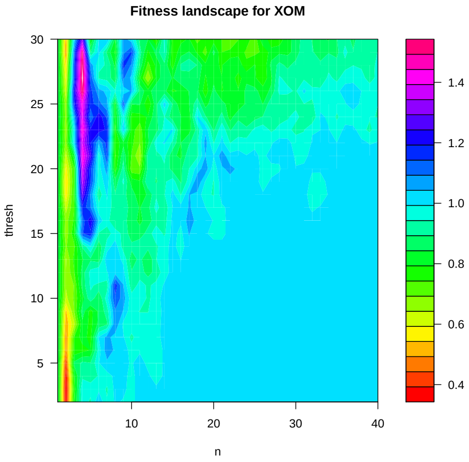

Round 1
Round 2
Round 3
Round 4
0.0
0.2
0.4
0.6
0.8
1.0

How to understand results?

- For 2 parameters the parameter space is 2d grid
- Then we can easily visualize the **fitness landscape**
- Fitness can be represented as colours on this 2d grid

In the practical you will use fitness plots to pick parameter values

Fitness landscape example



Other optimization methods

We mention, without going in to details, other methods you should be aware of. All are **gradient-free**, black-box optimizers:

- **Local-search based methods**: simmulated annealing; tabu search (short-term memory prevents cycling);
- **Population-based methods**: particle swarm optimization; ant colony optimization; genetic algorithms (the first 3 are all bio inspired); differential evolution
- **Bayesian optimization** (trades off exploitation and exploration)

More details on several of these are covered in **COMP305** Biocomputation and **COMP532** Machine Learning and BioInspired Optimisation (Masters level)