COMP226: Slides 18

Strategy parameters

Rahul Savani

rahul.savani@liverpool.ac.uk

Overview

- Recap definition of trading strategy **parameters**
- BBands Overbought/sold strategy: **changing parameters**
- **Enumerating** and **counting** parameter combinations

Recap

For us trading strategy **parameters** are

- **inputs** (often numerical) required to define the strategy

For example:

The parameters of the vanilla **BBands overbought/oversold strategy** we looked are:

- `n`: the size of the window of a moving average
- `sd`: the multiple of standard deviations for the Bollinger Bands

Recall strategy

Strategy

- Long when below lower Bollinger band line
- Short when above upper Bollinger band line

```
bbands  <- BBands(prices,n=50,sd=2)
long    <- ifelse(prices<bbands$dn,1,0)
short   <- ifelse(prices>bbands$up,-1,0)
pos     <- lag(long + short)
```

Effect of changing parameters

- For **fixed** `n`, **lower** `sd` gives **less stringent** trading condition
- Therefore the condition will likely be met more times so there will be **more active days**
- Let's confirm this **empirically**

Exploratory code

```
run <- function(prices,n,sd,plotEquity=FALSE) {
    bbands   <- BBands(prices,n=n,sd=sd)
    long     <- ifelse(prices<bbands$dn,1,0)
    short    <- ifelse(prices>bbands$up,-1,0)
    pos      <- long + short
    pos      <- lag(pos)
    pos[is.na(pos)] <- 0
    active_days <- sum(abs(pos)) # number of non-flat days
    equity <- getEquityCurve(getLogReturn(prices),pos)
    if (plotEquity) print(plot(equity,main="Equity curve"))
    simple_return <- round(last(as.numeric(equity)),2)
    return(c(simple_return, active_days))
}
```

Varying sd parameter

```
source('../utilities.R')
source('run_bbands.R')
library(quantmod)

prices <- getPrices(readCsvData('../GSPC.csv'))

n <- 10
sd <- seq(0.5,by=0.5,to=2.5)

results <- sapply(sd,function(x) run(prices,n=n,sd=x))
results <- t(results) # transpose
colnames(results) <- c("simple_ret","active_days")
results <- cbind(sd,results)
print(results)
```

Varying sd parameter

```
> results
      sd simple_ret active_days
[1,] 0.5       1.76        1020
[2,] 1.0       1.21         727
[3,] 1.5       1.15         355
[4,] 2.0       0.35          75
[5,] 2.5      -0.01           6
```

- This confirms that a **lower** `sd` **results in more trades**
- This is a logical consequence of the strategy definition
- We would find the **same thing for other values of** `n`

expand.grid

Now let's consider varying **both** parameters

```
n   <- seq(10 ,by=10 ,to=30)
sd  <- seq(0.5,by=0.5,to=2)

params <- expand.grid(n=n,sd=sd)
```

`expand.grid` returns a `data.frame` of **all parameter combinations**

expand.grid

```
> params
    n  sd
1  10 0.5
2  20 0.5
3  30 0.5
4  10 1.0
5  20 1.0
6  30 1.0
7  10 1.5
8  20 1.5
9  30 1.5
10 10 2.0
11 20 2.0
12 30 2.0
```

apply

Now we run the strategy with each parameter combination

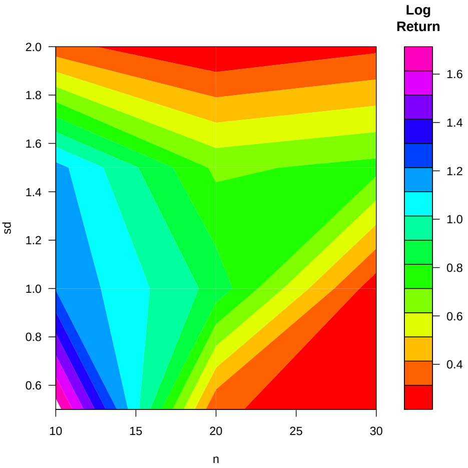
```
res <- apply(params,1,function(x) run(prices,
                                        n=x['n'],
                                        sd=x['sd']))
results <- t(res)
colnames(results) <- c("simple_ret","active_days")
results <- cbind(params,results)
results
```

- `apply` **applies a function over an array** (here `params`)
- second argument says: work over the **rows** of `params`, i.e., execute for each `i`: `run(prices,params[i,"n"],params[i,"sd"])`

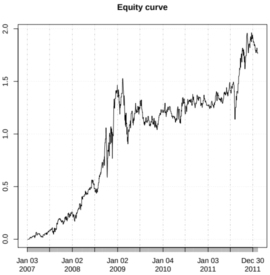
```
> results
    n  sd simple_ret active_days
1  10 0.5       1.76        1020
2  20 0.5       0.32        1033
3  30 0.5       0.28        1008
4  10 1.0       1.21         727
5  20 1.0       0.88         758
6  30 1.0       0.25         752
7  10 1.5       1.15         355
8  20 1.5       0.69         407
9  30 1.5       0.75         423
10 10 2.0       0.35          75
11 20 2.0       0.21         127
12 30 2.0       0.29         138
```

The relationship between active days and n is not so clearcut

Fitness landscape



Best result



- **Question**: How **representative** are the best parameters?
- We will return to this issue when we discuss **backtesting**

Counting parameter combinations

2 parameters:

```
> param1   <- c(10,20,30)
> param2   <- c(0.5,1,1.5,2)
```

The **cardinality** (size) of the two sets

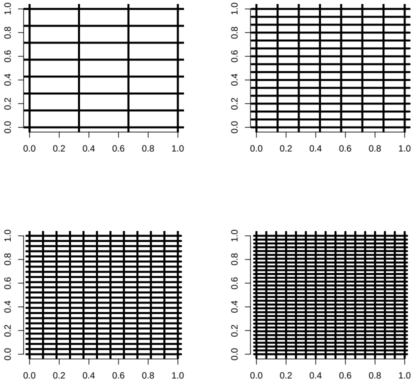
$$|\{10, 20, 30\}| = 3, \quad |\{0.5, 1, 1.5, 2\}| = 4$$

So the total numer of **parameter combinations** is:

$$3 \times 4 = 12$$

The underlying grid

These parameter combinations can be represented on a **2d grid**:



Counting parameter combinations

```
> param1   <- c(10,20,30,40)
> param2   <- c(5,6,7)
> param3   <- c(0,1,2)
```

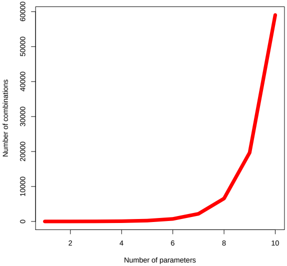
Now we have the product of three **cardinalities**

$$4 \times 3 \times 3 = 36$$

```
> params <- expand.grid(p1=param1,p2=param2,p3=param3)
```

Warning

The total number of parameter combinations grows **exponentially** with the number of parameters



Counting parameter combinations

- If there are **n** parameters and each can take at least **k** different values
- Then there are at least **k^n** parameter combinations
- To be precise:

If parameter **i**, for **i=1,...,n**, can take on **p(i)** different values, the total number of parameter combinations is

$$\prod_{i=1,...,n} p(i)$$

Selecting parameters

- Trading strategies are typically **parameterized**
- How should one choose **which parameter values to use?**
- Typically **parameter optimization** is used

Parameter optimization

Pick parameter values that are **likely** to produce good results **in the future**. In the terminology of **machine learning**, we want a model (strategy) that will **generalise** (to the future).

Warning

Optimization is important but **dangerous**

We will carefully consider how to avoid **over-optimization** and how to test the **robustness** of a strategy