Multimodal Optimization

- Intro to optimization
- Genetic Algorithms
- Other techniques: hillclimbing, simulated annealing
- Paper: Genetic Algorthms and Technical Analysis in fx data

Almost Everything is Optimization

Basic optmization framework:

- Some set of parameters that define a parameter space
- A fitness function that maps every point in the parameter space onto a fitness value
- Optmization is simply the process of adjusting these parameters until fitness is maximized (or some other criteria is reached).

Almost Everything is Optimization, Continued

Almost all machine learning algorithms contain have some element of optimization at their core. Examples:

- Simple: maximize $-((x-10)^2)$
- Neural networks: parameters are weights, fitness function is error on learning task
- Programs: parameters are various primitives in programming languae, fitness function is peformance on desired task

Appropriate Optimization Techniques

- Smooth functions with one local optimum can use calculus based methods (Lagrange, derivatves)
- With neural networks, all we have is a local gradient; pick the optimal direction an go
- When the landscape is not smooth, we must use other methods (like genetic algorithms)

How to Optimize with No Derivatives

Hillclimbing:

- Look one step away in random direction in parameter space
- Did we improve in fitness?
  - Yes: Move, and look again
  - No: Stay, and look again

Multiple Optima

- Sometimes, we have multiple optima
- Hillclimbing gets stuck in 'basin of attraction'.
- Solution: Multiple Restarts
- Or: Genetic Algorithms

Genetic Algorithms

Slides from book go here

Simulated Annealing

Examine fitness of neighboring point in parameter space

- If better than current point, accept move
- If worse then than current point, accept with the following probability (called the 'Metropolis Criterion'): $e^{(f(y)-f(x)/T)}$

  Where

  - $f(y)$ is the fitness of the current point
  - $f(x)$ is the fitness of the neighboring point
  - T is an annealing schedule parameter that decreases with time

Essentially, this is hillclimbing, but with a small probability of accepting a downhill move. This probability decreases with time.

Simulated Annealing Continued

- This is based on very well worked out physics.
- Theoretically, if the annealing is 'slow' enough, the algorithm is guaranteed to converge
- Works very well in practice – usually better than GAs.

Which Method?

Unfortunately, much more of an art than a science

- GA-like methods are good for parameters spaces of unknown size (like genetic programming trees)
- Hillclimbing (and multi-restart hillclimbing) are simple to implement with few parameters.
- Simulated Annealing is also simple, and in general the most effective.

Bottom Line: Genetic Algorithms have their uses, but don't fall prey to STD (sexy terminology disease). The evolutionary metaphor means nothing – pay attention to the underlying computation.

If somebody is using GAs, ask them why hillclimbing or simulated annealing aren't appropriate.

GAs and Forex Trading Rules

"Is Technical Analysis in the Foreign Exchange Market Profitable?"
Neely, Weller, Ditmar

Basic idea:
Evolve technical analysis style trading rules using genetic programming.

Authors are economists and treat genetic algorithms as a black box. Much more interested in statistical validation of results than in tweaking the algorithm

Overview

- Parameter Space: Program trees that map past price data onto long/short signals in fx trading
- Fitness function: excess returns provided by those strategies

Structure of GP Tree

- Operations on price time series – moving averages, max, min, etc
- Arithmetic operators
- Boolean operations
- if-then, if-then else
- numerical constants
- Boolean constants: 'true', 'false'

Fitness function

Tree is a function: maps past price data onto an 'long' (borrow dollars to buy marks, say) or 'short' (borrow marks to buy dollars) signal. Metric of success: excess returns over cost of capital.

Daily returns

$r_t = lnS_{t+1} - lnS_t + ln(1 + i_t^*) - ln(1 + i_t)$

- r_t is daily log return
- S_t is exchange rate at time t
- i_t^* foreign risk-free rate at time t
- i_t US risk-free rate at time t

Fitness function, continued

Total returns:

$r = \Sigma_{t=0}^{T-1} z_t r_t + n \cdot ln((1 - c)/(1 + c))$

- c is transaction cost
- n is number of transactions
- r_t is daily log returns
- z_t is long/short signal (provided by tree) $z_t = 1$ for long, $z_t = -1$ for short,

Data

Data is daily forex closes, for dollar, mark, swiss frank, and yen, with accompanying risk-free rates.

Split data into 3 chunks:

- Training: 1975-77; Train initial population of rules
- Validation: 1978-80; Pick best rule over validation set
- Test: 1981-1995; Report results on test set

(This terminology differs from the paper, which confusingly refers to the 'validation' set as a 'selection' set, and the 'test' set as the 'validation' set).

Why Validation?

- Optimization problems often 'noiseless'; see block stacking problem in book.
- But, this data is extremely noisy (efficient market hypothesis, anyone?)
- Thus, need to prevent overfitting.
- After training on training set, pick only the best rule on validation set

Results

Excess profits across all currencies

Annualized returns on test set :

- 6.05% for dollar/mark
- 2.34% for dollar/yen
- 2.27% for dollar/pound
- 4.1% for mark/yen

Benchmark?

The real question: How to interpret the results?

What benchmark?

- Raw excess returns?
- Buy-and-hold?
- Sharpe Ratios?
- Simple Moving Average Rule?

Statistical Tests: The Bootstrap

Statistical Tests:

- Normal hypothesis testing? No
- Bootstrapping? Yes, allows our null-models to have arbitrary distributions
  - random walk/sampling with replacement
  - ARMA(2,2) model
  - ARMA(2,2) + GARCH(1,1)
- Examine out-of-sample profits of generated rules on bootstrapped data
- Result: Bootstrapped data sets do *not* produce excess returns
- Bootstrapping is the gold standard for this sort of work

Questions?

- Only tested one set of rules (dollar/DM) with bootsrap
- Is this really better than a simple 150 day moving average rule?
- Is validation really helpful?

Implications and Future Directions

- If ARMA + GARCH doesn't explain everything, what does this say about pricing options?
- What about other markets?
- Can we tweak the algorithm – esp, better guards against overfitting?

What do you make of the results?