Machine learning optimal parameters for a channel breakout system for trading commodities

Martin Neal
Becky Engley

May 16, 2009

1 Dataset

We have financial market data for eighteen commodities over a period of thirty years. The commodities include live cattle, pork bellies, corn, wheat, lean hogs, crude oil, sugar, unleaded gasoline, heating oil, soybeans, cotton, orange juice, and soybean oil. Some commodities are traded during multiple seasons. The data includes open, high, low, and closing values for each commodity on every trading day. The data was obtained from:

http://www.normanshistoricaldata.com

2 Background

A seasonal trading system leverages the normal fluctuations in supply and demand for certain commodities that occur every year in fairly consistent patterns. A commodity is either traded short, long, or bi-directionally. If a trader expects an uptrend in the value of a commodity, he will trade long, buying before selling. If a trader expects a downtrend, he will trade short, selling before buying.

In his book, Superstar Seasonals, John Momson defines a channel breakout system as a model for predicting uptrends and downtrends. According to his system, a commodity may be traded between a range of dates specified by the parameters *entry_window_open* and *entry_window_close*. A trade must be completed no later than the *exit_trade* date, a third parameter of the system.

A channel is defined by an upper channel line and a lower channel line. The upper channel line is determined by the maximum *high* value over the previous m days. The lower channel line is determined by the minimum *low* value over the previous n days. Here, m and n are also parameters of the system. If trading long, the upper channel line is the *entry* line and the lower channel line is the *trail-stop* line. If trading short, the channel lines are reversed.

The sixth and final system parameter, the *stop-loss* threshold, serves as a safety net to prevent large losses. When trading short, the stop-loss is defined by the maximum *high* value over the previous q days, where q is strictly less than m. When trading long, the stop-loss is defined by the minimum *low* value over the previous q days, where q is strictly less than n. The trail-stop line protects profits, while the stop-loss threshold minimizes losses.

According to the system, if the current value of the commodity crosses the entry line, it indicates the beginning of a new trend and the trade is begun. When the current value of the commodity crosses the trail-stop line, the trade is completed.

3 Project idea

We intend to machine learn optimal values for the six trading system parameters. Optimal values maximize the profit earned as a result of trading the system. We will approach this optimization problem using two different machine learning techniques: simulated annealing, and genetic algorithms.

3.1 Simulated Annealing

Simulated Annealing combines the best of hill climbing and random walk heuristic algorithms. The major problem with hill climbing algorithms is that they can get stuck on local maxima, because they never move downhill. Random walks, however, are guaranteed to find the global maximum but take far too long to do so. Simulated Annealing combines these approaches, yielding both efficiency and completeness. We present the psuedo-code for the Simulated Annealing algorithm below.

```
simulated-annealing()
    current ← 6 random parameter values
    for t ← 1 to ∞
        η ← f(t)
        next ← successor(current)
        ΔE ← value(next) - value(current)
        if(ΔE > 0)
            current ← next
        else
            current ← next only with probability e^(ΔEη)
    return current
```

The value function trades the system with the six parameters and returns the profit made (or lost). current and next are both nodes. In this context, a node is a set of values for the six parameters. η controls the probability of downward steps. $f(t)$ is a decreasing function of time; it decreases the probability of downward steps as time increases. This may be done in any number of ways (e.g. linearly, exponentially). The choice for the successor function greatly affects the learning speed of the algorithm. Our successor function will adjust exactly one of the six parameters in current by a random amount, δ, which may be either uniformly or normally distributed. As a variation on this function, we can change multiple parameters at once. We intend to experiment with all of these variations.

3.2 Genetic Algorithm

Genetic Algorithms model evolutionary processes. In our proposed implementation, a set of six system parameters represents an individual. Many individuals form a population which is repeatedly bred and then culled. Breeding swaps random parameters from two or more parents to create children. Culling evaluates each individual using a fitness function, and eliminates unfit individuals from the population. The fitness function returns the profit earned by trading the channel breakout system using the individuals' six parameters. We present the psuedo-code for the Genetic Algorithm below.

```
genetic-algorithm(population)
    do
        for i ← 1 to size(population)
            parents ← random-subset(population,size)
            children ← reproduce(parents)
            children ← mutate(children) with small random probability
            population.add(children)
        population ← cull(population,threshold)
    until (enough time has elapsed)
    return best-individual(population)
```

This generic algorithm leaves a lot to be decided by the implementer. The initial population size may vary. The mutate probability may be changed. The mutation of an attribute may be uniformly random or normally distributed. The reproduce function may take between two and six parents. These parents randomly swap attributes to produce one or more children, which are added to the population. The population is then culled, which preserves a threshold number of individuals, removing all others as unfit.

4 Required software and midway report goals

To implement our project idea, we will need to write a series of programs. One program will trade the channel break-out system using the six parameters and return the profit earned. The other two programs will implement the simulated annealing and genetic algorithms as described above. Our midway report goals are to have the trading system implemented along with at least one of the algorithms. We would also like to have some provisional data collected by then.