Particle Swarm Optimization for Algorithmic Trading

Eric Benhamou
Université Paris Est
Laboratoire de Mathématiques
eric.famillebenhamou@gmail.com

Abstract
Automated trading systems make decisions on how to invest in financial markets. More precisely, these algorithms decide when to trade (buying or selling), in which direction (long or short), on which market (underlying), with sometimes predetermined levels of risk (stop loss level) and rewards (profit target) and in which quantity. These decisions depend on a variety of parameters that must be optimized to maximize returns and overall profits while minimizing risk.

We specifically look at four Kalman filter models. These models are parameters that need to be estimated according to a best fit function. This function is very irregular and non convex.

In this research, we investigate the use of various optimization algorithms like the simplex method to more heuristic techniques such as PSO and Monte Carlo. We specifically look at four Kalman filter models. These models are determined level of risk (stop loss level) and rewards (profit target).

We therefore look at particle swarm to be able to estimate the best parameters over one year of data.

1. Introduction
Investing in financial markets is not an easy task. As markets roll up and down, choosing the right time to invest or disinvest is a complex task. There is substantial risk of loss associated with trading securities. It can be emotionally stressful. Without an edge, it is like playing to the casino.

One tool that can help making an informed forecasting decision and that is widely used in artificial intelligence (AI) like genetic algorithm and particle swarm optimization to be able to calibrate them in a very short time.

2. Kalman Filter
Consider a linear dynamic system given by:
\[ X_{t+1} = AX_t + Q \]
where \( X_t \) is the state and \( Q \) is the noise.

\[ Y_{t+1} = HX_t + \epsilon \]
where \( Y_t \) is the measurement and \( \epsilon \) is the noise.

Definition 1 The Kalman filter is given by two state variables at time \( t \): state vector estimate \( \hat{X}_t \) and error covariance matrix estimate \( P_t \), whose dynamics are given by:
\[ \hat{X}_t = P_t \hat{X}_t + P_t Q \]
\[ P_{t+1} = (I - P_t H) P_t \]

A naive search with 20 possible values for each parameter will not be feasible for any model but one as shown in table 2. Calibration 1 is a simple calibration of the model. \[ 1 \] is the parameters in range between 0 and 1 (p1, p2, p3) and velocity (v1, v2, v3).

\[ P_{t+1} = (I - P_t H) P_t \]

We therefore look at particle swarm to be able to estimate the best parameters over one year of data.

3. Naïve Calibration
A naive search with 20 possible values for each parameter will not be feasible for any model but one as shown in table 2. Calibration 1 is a simple calibration of the model. \[ 1 \] is the parameters in range between 0 and 1 (p1, p2, p3) and velocity (v1, v2, v3).

\[ P_{t+1} = (I - P_t H) P_t \]

We therefore look at particle swarm to be able to estimate the best parameters over one year of data.

4. Particle Swarm Optimization
Particle Swarm Optimization (PSO) is a stochastic evolution algorithm based on swarm intelligence, which was first introduced by [Kennedy/Eberhart(1995)]. Since its inception, PSO has shown great success in solving function optimization problems and has been widely applied in a variety of engineering applications. PSO is motivated by the behavior of birds flocking in finding food. Suppose a flock of birds want to find food, but they do not know where the food is before they find it. PSO uses a swarm of particles to simulate these birds. Each particle is a possible solution of the optimization problem and has a random initial position \( x_0 \) and velocity \( v_0 \). The objective function targeted to be optimized \( f \) is used to evaluate each particle’s fitness. Higher fitness means a better position. For each particle, PSO uses \( v_t \) to record the best position this particle has arrived. For the whole swarm, \( x_t \) is used to record the global best position achieved by all particles. Velocity is updated according to an initial inertia weight \( w \) and two coefficients \( c_1 \) and \( c_2 \), that represent cognitive/local weight and social/global weight and two random numbers in the range between 0 and 1. The algorithm is simple and summarized by table 1 below:

5. Results and Discussion
Thanks to Particle Swarm, we are able to find the best parameters for Kalman filter models 1, 2, 3 and 4 for one year period for the Mini SP500 futures contracts. The results are summarized in table 3. More results can be found in [Benhamou(2017)].

6. Conclusion
We have shown that using particle swarm allows us to specify a complex Kalman filter model and find appropriate parameters in reasonable time.

Obviously, these optimization techniques can be applied to other trading rules, like ones inspired from technical analysis indicators like Moving Averages across overs, trading range break out systems, Relative Strong Index (RSI), Bollinger Bands, Stochastic Oscillator, Moving Average Convergence/Divergence (MACD), On Volume Average (OBV), etc... Fundamental rules can also be applied with Valuation Break out. Style (growth vs. value), size (large cap vs. small cap), Earning Quality, etc...

References