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 (timing), in which direction (long or short), on which market (underlying), with some predetermined level 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 have 4 up to 15 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 used in artificial intelligence (AI) like genetic algorithm and particle swarm optimization to be able to calibrate them in a very short time.

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 is Kalman filter (KF). Kalman filter is a recursive algorithm that has been invented in the 1960s to track a moving target, remove any noisy measurements of its position and predict its future position.

In finance, KF has been used by the asset management industry for various purposes. KF is an optimal choice in many cases and do much better than a simple moving average crossover. KF also has prediction power while moving averages suffer from lag. [Bruder-Dao-Richard-Roncalli(2011)] and [Dao(2011)] showed that for price following random walk with noise, KF is equivalent to the optimal exponential moving average with parameter equal to Kalman gain. [Chan(2013)] suggested using KF for pair correlation trading.

2. Kalman Filter

Consider a linear dynamic system given by

$$X_{t+1} = \phi X_t + c_t + w_t \qquad (1)$$
$$Y_t = H X_t + d_t + v_t \qquad (2)$$

where X_t and Y_t are the state and measurement vectors, ϕ and H the state transition and the measurement matrices, w_t and v_t the independent white noises with zero mean and their variance matrices given by Q and R. c_t and d_t are the drift of the state and the measurement vector.

Definition 1 *The Kalman filter is given by two states variables at time t: state vector estimate $X_{t|t}$, and error covariance matrix estimate $P_{t|t}$, whose dynamics are given by:*

$$X_{t+1|t} = \phi X_{t|t} + c_t \qquad (3)$$
$$P_{t+1|t} = \phi P_{t|t} \phi^T + Q \qquad (4)$$
$$X_{t+1} = X_{t+1|t} + K_{t+1}\left(Y_{t+1} - Y_{t+1|t}\right) \qquad (5)$$
$$Y_{t+1|t} = H X_t + d_t \qquad (6)$$
$$K_{t+1} = P_{t+1|t} H^T \left[H P_{t+1|t} H^T + R\right]^{-1} \qquad (7)$$
$$P_{t+1|t+1} = \left[I - K_{t+1} H\right] P_{t+1|t} \qquad (8)$$

Equation (3) and (4) are the prediction steps that provides next value for the estimates $X_{t+1|t}$ and $P_{t+1|t}$. Equations (5), (6), (7), (8) are the correction steps. The variable K_{t+1} is referred to as the Kalman gain.

The difficulty of using KF modelling lies in the specification of the underlying dynamics. [Benhamou(2017)] gives four different models:

- **model 1**: inspired from kinematics where the state vector evolves according the Newton equation with speed and acceleration.
- **model 2**: inspired from local linear trend model with an additional parameter compared to model one.
- **model 3**: a two factors model with contribution to price given by short and long term.
- **model 4**: a model inspired by a combination of oscillators and the previous model. In this model, we use price position with respect to its extrema.

Details of the models are summarized in table 1.

Model	ϕ	H	Q	R	$P_{t=0}$	c_t
1	$\begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} p_1^2 & p_1 p_2 \\ p_1 p_2 & p_2^2 \end{bmatrix}$	$[p_3]$	$\begin{bmatrix} p_4 & 0 \\ 0 & p_4 \end{bmatrix}$	0
2	$\begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} p_1^2 & p_1 p_2 \\ p_1 p_2 & p_2^2 \end{bmatrix}$	$[p_3]$	$\begin{bmatrix} p_4 & 0 \\ 0 & p_5 \end{bmatrix}$	0
3	$\begin{bmatrix} p_1 & p_2 \\ 0 & p_3 \end{bmatrix}$	$\begin{bmatrix} p_4 \\ p_5 \end{bmatrix}$	$\begin{bmatrix} p_6^2 & p_6 p_7 \\ p_6 p_6 & p_7^2 \end{bmatrix}$	$[p_8]$	$\begin{bmatrix} p_9 & 0 \\ 0 & p_{10} \end{bmatrix}$	0
4	$\begin{bmatrix} p_1 & p_2 \\ 0 & p_3 \end{bmatrix}$	$\begin{bmatrix} p_4 \\ p_5 \end{bmatrix}$	$\begin{bmatrix} p_6^2 & p_6 p_7 \\ p_6 p_6 & p_7^2 \end{bmatrix}$	$[p_8]$	$\begin{bmatrix} p_9 & 0 \\ 0 & p_{10} \end{bmatrix}$	$\begin{bmatrix} p_{11} & -p_{12} K_t \\ p_{13} & -p_{14} K_t \end{bmatrix}$

Table 1: *Kalman filter model specifications*

3. Naive Calibration

A naive search with 20 possible values for each parameters will not be feasible for any model but one as shown in table 2. Calibration 1 is a simple calibration of the model parameters while calibration 2 includes in addition the calibration of a profit target and stop loss. It adds two extra parameters (profit target and stop loss level). On a standard computer, each iteration takes about 10 ms. Because of the curse of the dimension, calibration times explodes as we reach more than dimension 5. In table 2, t. years stands for thousands of years, b. for billions of years, and u.l. for the universe life. Basically, a brute force optimization is not feasible for model KF 3 and KF 4.

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

model	dimension	calibration 1	calibration 2
KF 1	4	0.4 hours	7.4 days
KF 2	5	8.9 hours	148.2 days
KF3	10	3 t. years	1.3 m. years
KF4	14	520 m. years	15 u.l.

Table 2: *Timing for brute force optimization t. years stands for thousands of years, m. for millions of years, u.l. for the univers life*

4. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a stochastic evolution algorithm based on swarm intelligence, which was first introduced by [KennedyEberhart(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 behaviour of bird flocks 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_i and velocity v_i . The objective function targeted to be optimized f is used to evaluate each particle position's fitness. Higher fitness means a better position. For each particle, PSO uses p_i to record the best position this particle has arrived. For the whole swarm, g is used to record the global best position achieved by all particles. Velocity is updated according to an initial inertia weight ω and two coefficients φ_p and φ_g that represent cognitive/local weight and social/global weight and two random numbers in the range between 0 and 1 r_p and r_g
The algorithm is simple and summarized by table 1 below:

```
foreach( particle in Swarm )
{
  // initialize particle's position x_i and velocity v_i
  x_i = ..., v_i = ...;
  // initialize particle's best position to initial position
  p_i = x_i;
  // update swarm's best position
  if( f(p_i) < f(g) )
    g = x_i;
}

while( termination criterion not met )
{
  foreach( particle in Swarm )
  {
    foreach( dimension d = 1, ..., n)
    {
      // update particle's velocity
      v_{i,d} = ω v_{i,d} + φ_p r_p (p_{i,d} − x_{i,d}) + φ_g r_g (g_d − x_{i,d});
      // update particle's position
      x_i = x_i + v_i ;
      // update particle's best known position
      if( f(x_i) < f(p_i )
        p_i = x_i;
      // update swarm's best known position:
      if( f(p_i) < f(g) )
        g = p_i;
    }
  }
}
```

Algorithm 1: *Particle Swarm Optimization pseudo code*

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 a one year period for the Mini SP500 futures contracts. The results are summarized in table 3. More results can be found in [Benhamou(2017)]

Model	Profit	Drawdown	# Trades	Recovery ratio
1	18,755	- 7,348	55	2.55
2	22,380	- 7,348	55	3.05
3	29,022	- 3,800	57	7.64
4	39,558	- 2,600	48	15.21

Table 3: *Results for the various models*

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 cross overs, trading range break out systems, Relative Strength Index (RSI), Bollinger Bands, Stochastic Oscillator, Moving Average Convergence/Divergence (MACD), On Volume Average (OBVA), 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

[Benhamou(2017)] E. Benhamou Trend without Hiccups, a Kalman filter approach, IFTA Journal 2017, pp. 38-46.

[Bruder-Dao-Richard-Roncalli(2011)] B. Bruder, T.L. Dao, J.C. Richard, T. Roncalli. Trend Filtering Methods for Momentum Strategies SSRN paper. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2289097, 2011

[Chan(2013)] E. Chan. Algorithmic Trading: Winning Strategies and Their Rationale, *Wiley Finance*, 2013.

[Dao(2011)] T.L. Dao. Momentum Strategies: From Novel Estimation Techniques to Financial Applications, SSRN White Paper http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2358988, 2011

[KennedyEberhart(1995)] J. Kennedy. R. Eberhart. Particle swarm optimization. In Neural Network, 1995 IEEE International Conference, *Conference Proceedings* vol.4, p. 1942 –1948 Nov/Dec 1995.