Semester Project: Improve Prediction Model of Trading Agent.

Contents

Abstract

This report describes three different filters for Trading Agent.
Also I will show here compare of these filters and in conclusion I will assume which filter works better and why.

Introduction

TAC Ad Auction is realistic simulator in which participants can develop strategies that could apply to real sponsored-search auctions.

Internet advertising has become a very important part of today's marketing processes. Internet advertising provides a substantial source of revenue for online publishers, amounting to billions of dollars annually. Sponsored search is a popular form of targeted advertising, in which query-specific advertisements are placed alongside organic search-engine results. Advertisers compete among each other to get the best promotion of their products on the internet. Sponsored search is a popular facet of internet advertising where each advertiser buys, in an ad auction, their ad position on a page for a given query.
Advertisers represent retailers in a simplified home entertainment market, bidding to place ads before users searching on product keywords.

Prediction models.

In this report we will compare 3 methods which we can use to predict impressions.

1) Particle Filter
2) Kalman filter
3) Grid-Based filter

Particle filter

Particle filters analogue of Markov chain Monte Carlo (MCMC) batch methods and are often similar to importance samplingmethods. Particle filters can often be much faster than MCMC. They are often an alternative to the Extended Kalman filter (EKF) or Unscented Kalman filter (UKF) they can be made more accurate than either the EKF or UKF. However, when the simulated sample is not sufficiently large, they might suffer from sample impoverishment. The approaches can also be combined by using a version of the Kalman filter as a proposal distribution for the particle filter.

For each population of the game agent simulate different “population particles” that represent a possible states distribution of the real population. Then agent compute the probability that this particle exists given some observations of the game, then agent assign a weight to each particle given its probability of occurrence.

The parameters of the filter:

For any population all the F2 users generate F2 queries, and each IS user may generate with the same probability an F0, F1 or F2 query. So from the impressions we get for an F2 query we can calculate the probability of existence of each particle, in fact:

$$Imp_{F2}(q) \approx N_{F_2}(q) + \frac{1}{3}N_{IS}(q) \quad (1)$$

$$Imp_{F_2}(q) = \#\ of\ Impressions\ for\ query\ q\ of\ type\ F_2$$

$$N_{F_2}(q) = \#\ of\ F_2\ users\ from\ the\ population\ that\ has\ preferences\ defined\ like\ q$$

$$N_{IS}(q) = \#\ of\ IS\ users\ from\ the\ population\ that\ has\ preferences\ defined\ like\ q$$

Expression (1) becomes: $Imp_{F2}(q) - N_{F_2}(q) \approx \frac{1}{3}N_{IS}(q)$ we thus can get a binomial distribution $Bin\sim X(n,p)$ with parameters: $n = N_{IS}(q)$ and $p = \frac{1}{3}$ and we calculate the desired probability:

$$\boldsymbol{prob_{n_{IS},1/3}(n_{F_2}|Impressions_{F_2})} := P\left(X = Imp_{F_2}(q) - N_{F_2}(q)\right)$$

Kalman filter:

The Kalman filter assumes that the posterior density at everytime step is Gaussian and, hence, parameterized by a mean andcovariance. Kalman filter produces estimates of the true values of measurements and their associated calculated values by predicting a value, estimating the uncertainty of the predicted value, and computing a weighted average of the predicted value and the measured value. The most weight is given to the value with the least uncertainty. The estimates produced by the method tend to be closer to the true values than the original measurements because the weighted average has a better estimated uncertainty than either of the values that went into the weighted average.

The Kalman filter is applied recursively through time to construct prediction and prediction variances. Each step of the process allows the next observation to be prediction based on the previous observation and the prediction of the previous observation. That is, each consecutive prediction is found by updating the previous prediction. The update rules for each prediction are weighted averages of the previous observation and the previous prediction error. These update rules resemble those of an allied approach to predictioning called exponential smoothing. The intriguing feature of the Kalman filter is that the weights in the update rules are chosen to ensure that the prediction variances are minimised. These weights, referred to collectively as the Kalman gain, play a similar role to the so-called smoothing constants in exponential smoothing.

The parameters of the filter:

$$x_k = F_k x_{k-1} + V_{k-1}$$

Where

$F_k x_{k-1}$ – Matrix $\begin{pmatrix} \#IS_{k-1} \\ \#F2_{k-1} \end{pmatrix}$

V_{k-1}-is the control-input model which is applied to the control vector

Grid-Based filter

Grid-based methods provide the optimal recursion of the filtered density $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ if the state space is discrete and consists of a finite number of states. Suppose the state space at time $k-1$ consists of discrete states $\mathbf{x}^i_{k-1}, i = 1, \ldots, N_s$. For each state $\mathbf{x}^i_{k-1}$, let the conditional probability of that state, given measurements up to time $k-1$ be denoted by $w^i_{k-1|k-1}$, that is, $\Pr(\mathbf{x}_{k-1} = \mathbf{x}^i_{k-1}|\mathbf{z}_{1:k-1}) = w^i_{k-1|k-1}$. Then, the posterior pdf at $k-1$ can be written as

$$p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}) = \sum_{i=1}^{N_s} w^i_{k-1|k-1}\delta(\mathbf{x}_{k-1} - \mathbf{x}^i_{k-1})$$

Compare three prediction filters

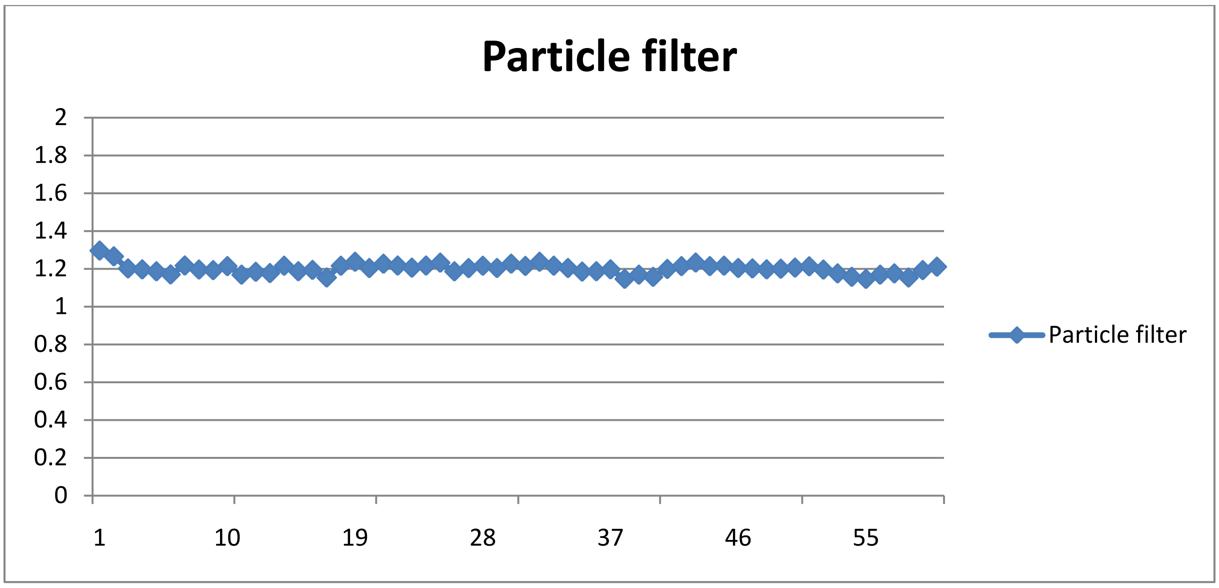
To compare three different filters we started 36 games for each of filter to get statistic data of prediction models.

In all these games we took only data with parameter F2 (F2: Focus level 2-searching users, they generate (Manufacturer specialty, Component Specialty) queries, these users may transact).

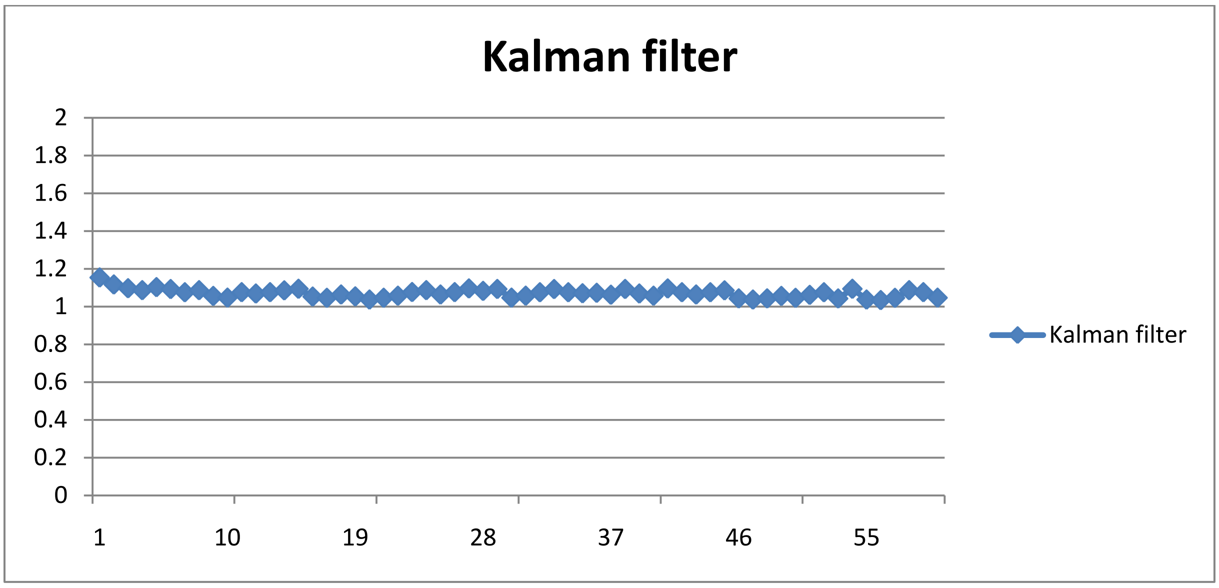
For made statistics data we used formula which you can see below to calculate value of error (difference between prediction impressions and real impressions)

$$\frac{Pred\#F2 + \frac{1}{3}Pred\ \#IS}{\#F2real\_impr}$$

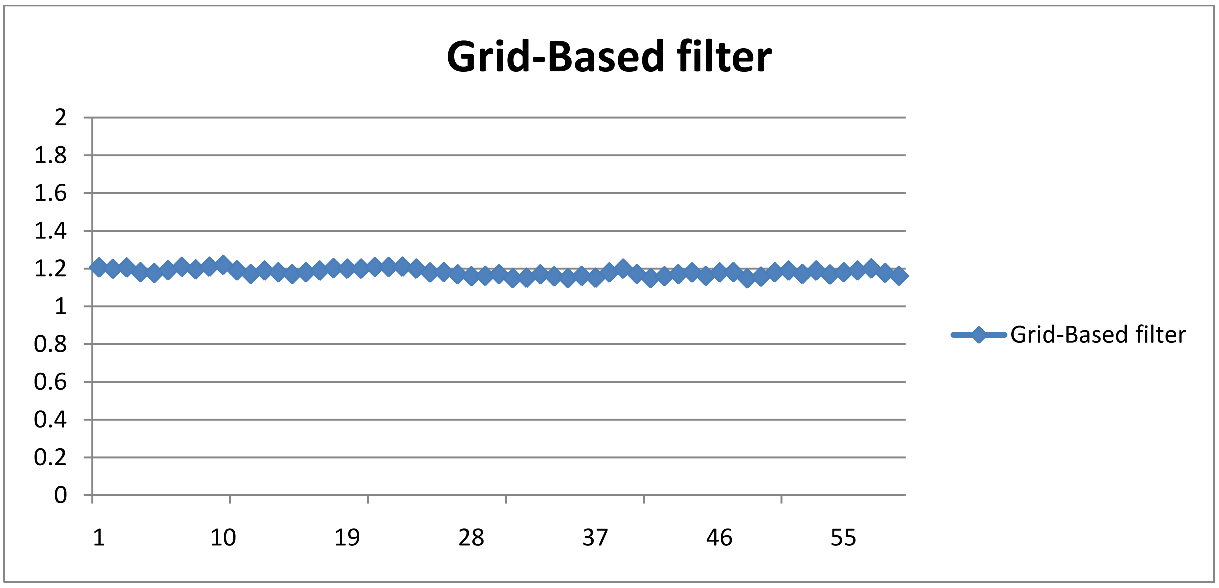
1) Particle filter



2) Kalman filter



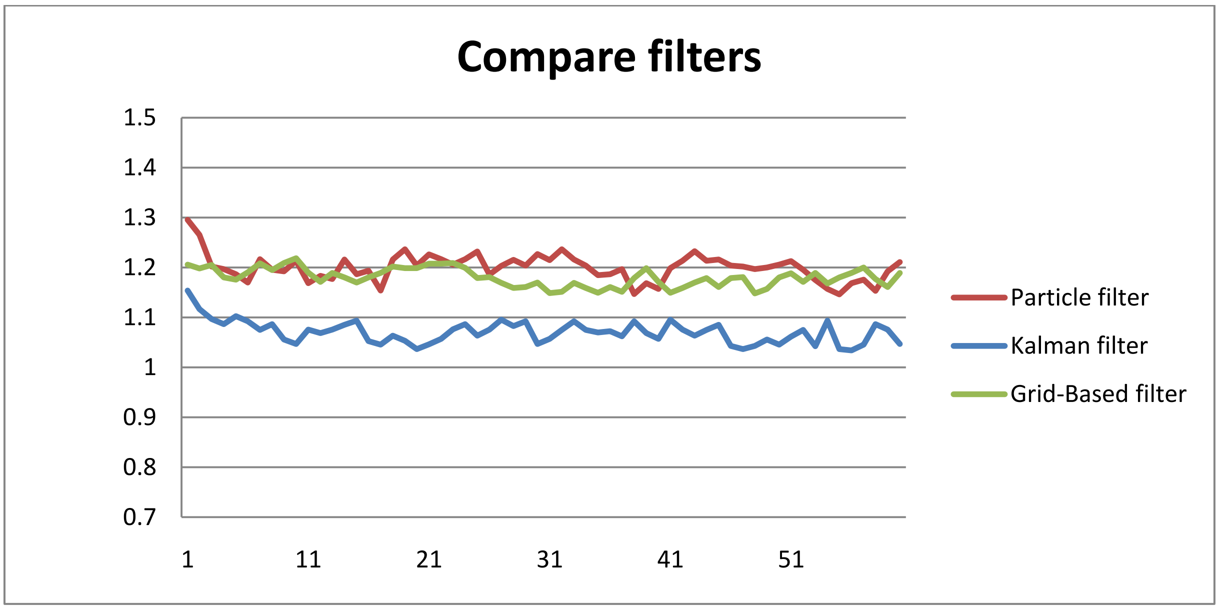
3) Grid-Based filter



Compare table

Filter	Runtime	Average value for 36 games
Particle Filter	10sec	1.199641
Kalman Filter	18sec	1.070303
Grid-Based Filter	12sec	1.180124

How we can see in table Kalman filter have best results but it also have highest runtime. Also we can see that Grid-Based filter have pretty same value like Particle filter but it have higher runtime.



Also if we look at graph we can assume that Kalman filter it’s best decision for prediction impression because it have average error value for 36 games equal 1,07.

Filter	+	-
Particle filter	Lowest runtime (10 sec)	Highest error value (1.19)
Kalman filter	Good runtime (18 sec), Lowest error value(1.07)	-
Grid - based filter	Good runtime (12 sec)	Error value (1.18)

We can assume that Kalman filter have more pluses then other filters and also this filter theoretically better than other.

Conclusion

In this report we can see description of all three prediction models also I describe how I used it in Agent. Also I compare all methods and describe all pluses and minuses for each filter. And in the end of report I made conclusion which of this filters is better and why.

References

1) D. Pardoe, D. Chakraborty, and P. Stone. TacTex09: A champion bidding agent for ad auctions, May 2010.

2) L. Cigler. Bidding Agent for Advertisement Auctions, 2009.

3) M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. February 2002.

4) Wikipedia. Kalman Filter.

5) Dan Simon. Kalman Filter 2001.