COMP6212 Assignment 4 : Shakib-Bin Hamid 25250094 sh3g12

1 Kalman Filter

Lets suppose a data generative process is a linear combination of its past values, i.e. the current process process parameters are a linear combination of previous process parameters. In this respect we can calculate the process parameters by minimising the least squared error between the prediction and the true values, $\boldsymbol{\omega} = (\boldsymbol{Y}^t\boldsymbol{Y})^{-1}\boldsymbol{Y}^t\boldsymbol{f}$, where $\boldsymbol{Y}$ is the input data and $\boldsymbol{f}$ is the target. But this method requires calculating a new $\boldsymbol{Y}^{-1}$ (a $O(n^3)$ operation) when we receive any new data. If the process is generating a time series data (getting data one at a time), then online updating the parameters can be incredibly costly.

However, it can be shown that we can calcuate new parameters without calculating a new inverse, $\boldsymbol{\omega}_n = \boldsymbol{\omega}_{n-1} + L \times (f_n - \boldsymbol{\omega}_{n-1}^t \boldsymbol{y_n})$. It means that we can calculate new parameters by updating the old parameters with the scaled prediction error on new data using the old parameters. Of course we need to assume some initial values for our parameters. But, we still do not account for any noise in our process.

In a Kalman Filter, we may assume a process $\boldsymbol{\omega}_n = \boldsymbol{A}\boldsymbol{\omega}_{n-1} + \boldsymbol{\epsilon}(n)$, where the process noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{Q})$. Here we can assume that our parameters are a linear transformation of previous parameters corrupted by Gaussian noise. Similarly, we can assume that our observation $f_n = \boldsymbol{\omega}_n \boldsymbol{y}_n + \xi(n)$, where observation noise $\xi \sim \mathcal{N}(0, R)$.

[1] claims that we can predict the monthly S&P 500 index using a Kalman filter. In order to verify his claim, I have implemented Algorithm 1 -

Algorithm 1 S&P 500 Index Prediction using a Kalman Filter

procedure KALMAN($\boldsymbol{s}, o, \alpha$) ▷ $\boldsymbol{s}$ = index, o = order, α = a constant

$N = size(\boldsymbol{s}, 1)$; ▷ Number of months
$\boldsymbol{Y} = zeros(N - o + 1, o)$; ▷ Input data in o order windows
$\boldsymbol{W} = zeros(N, o)$; ▷ Weights at every point
$\boldsymbol{e} = zeros(N - o + 1, 1)$; ▷ Prediction error at every point

$m = ar(s, o)$; ▷ Matlab's autoregression function
$R = m.NoiseVariance$; ▷ Observation noise variance
$\boldsymbol{w} = (m.a(2 : o + 1) \times -1)^t$; ▷ Current weight set to initial value

$\boldsymbol{A} = \boldsymbol{I}_o$ ▷ $\boldsymbol{I}_o$ is the o order Identity
$\boldsymbol{Q} = \alpha \boldsymbol{I}_o$ ▷ Process noise covariance
$\boldsymbol{P} = \boldsymbol{I}_o$ ▷ Uncertainity or covariance of parameters

for $n \leftarrow o + 1, N$ **do**
$\boldsymbol{y}_n = \boldsymbol{Y}(n, :) = \boldsymbol{s}(n - o : n - 1)$ ▷ Input is last o months' index

$\boldsymbol{P} = \boldsymbol{A}\boldsymbol{P}\boldsymbol{A}^t + \boldsymbol{Q}$ ▷ Uncertainity on parameters increases

$\boldsymbol{e}(n) = \boldsymbol{s}(n)$ - $\boldsymbol{w}^t \boldsymbol{y}_n$ ▷ Prediction Error
$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{y}_n / (R + \boldsymbol{y}_n^t \boldsymbol{P}\boldsymbol{y}_n)$ ▷ Kalman Gain

$\boldsymbol{w} = \boldsymbol{W}(n, :) = \boldsymbol{w} + \boldsymbol{K}\boldsymbol{e}(n)$ ▷ Correction
$\boldsymbol{P} = (\boldsymbol{I}_o - \boldsymbol{K}\boldsymbol{y}_n^t)\boldsymbol{P}$

The input at each point in time is past o index values. Observation noise R is taken as the residual variance from an autoregression model using `ar` function of Matlab's System Identification Toolbox. Process noise covariance needed to be tuned via α. Figure 4a (where I have plotted the cumulative sum of prediction error as a function of α for a 3rd order filter), indicates that 10^{-3} is a reasonable value for α. Unless otherwise specified, it can be assumed that a 3rd order filter and autoregression was in use. The parameters were initialised in Line 10 and covariance in Line 14.

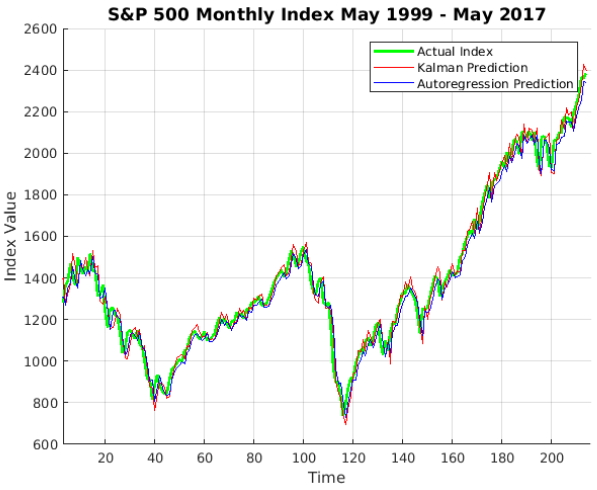
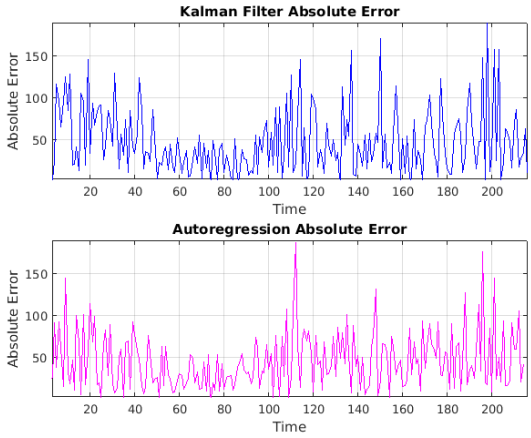
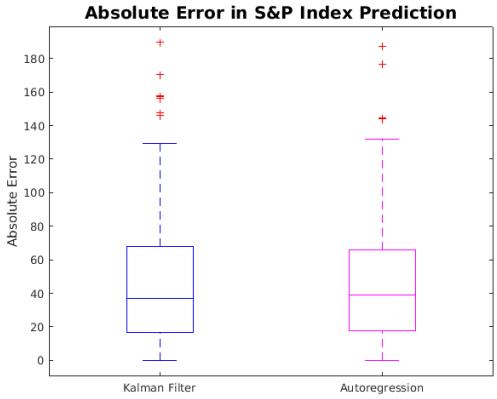


Figure 1: Kalman & Autoregression Prediction



(a) Index Prediction Absolute Error



(b) Index Prediction Error Comparison

Figure 2: Prediction Error

I began by generating an artificial time series using white Gaussian noise (Matlab's `wgn` function) and $(0.5, 0.6, 0.1)$ weights. The kalman filter quickly stabilises on this data at the expected weights (Figure 3a). After confirming the correctness of the kalman filter, I applied it to real data.

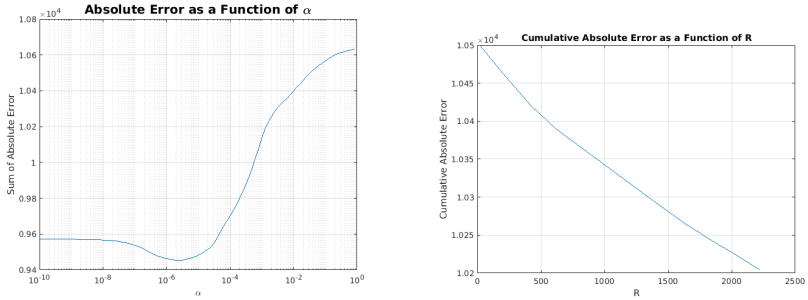
Figure 1 is showing the monthly index values, as well as prediction made from an autoregression and the implemented Kalman filter. Clearly both are very close to the true values of the index. We can also see from Figure 2a that both methods produce very similar error patterns. Figure 2b shows that the errors have very similar spread.

Figure 3b shows the convergence of the parameters on real data. It demonstrates the recursive correction of weights.

Now we can also observe from Figure 4a that error changes as the choice of α changes. The α choice is rather arbitrary and needs tuning. As I mentioned earlier, I chose $\alpha = 10^{-3}$ for all orders, since it seemed reasonable at most orders. Similarly, I plotted the error by varying $R = (20 : 200 : 2000)$. Figure 4b shows the effect of tuning R. I have chosen the noise variance of an autoregression model to be the effective R after this observation.

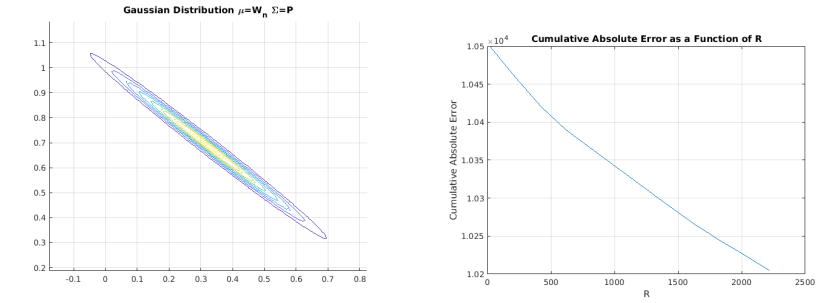
(a) Artifical (b) $R \approx 53$ (c) $R = 420$

Figure 3: Parameter Convergence



(a) $\boldsymbol{e} = f(\alpha)$ (b) $\boldsymbol{e} = f(R)$

Figure 4: Tuning α and R



(a) 2nd Order Parameter Contour (b) $\boldsymbol{e} = f(R)$

Figure 5: Tuning α and R

References

[1] N. Mahler, "Modeling the S & P 500 index using the Kalman filter and the LagLasso," in *Machine Learning for Signal Processing, 2009. MLSP 2009. IEEE International Workshop on, Sept 2009*, pp. 16.