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

```
1:  procedure KALMAN($\boldsymbol{s}, o, \alpha$)          ▷ $\boldsymbol{s}$ = index, $o$ = order, $\alpha$ = a constant
2:
3:      $N = size(\boldsymbol{s}, 1);$                          ▷ Number of months
4:      $\boldsymbol{Y} = zeros(N - o + 1, o);$                  ▷ Input data in $o$ order windows
5:      $\boldsymbol{W} = zeros(N, o);$                          ▷ Weights at every point
6:      $\boldsymbol{e} = zeros(N - o + 1, 1);$                  ▷ Prediction error at every point
7:
8:      $m = ar(s, o);$                                          ▷ Matlab's autoregression function
9:      $R = m.NoiseVariance;$                                   ▷ Observation noise variance
10:     $\boldsymbol{w} = (m.a(2 : o + 1) \times -1)^t;$          ▷ Current weight set to initial value
11:
12:     $\boldsymbol{A} = \boldsymbol{I}_o$                        ▷ $\boldsymbol{I}_o$ is the $o$ order Identity
13:     $\boldsymbol{Q} = \alpha\boldsymbol{I}_o$                  ▷ Process noise covariance
14:     $\boldsymbol{P} = \boldsymbol{I}_o$                        ▷ Uncertainity or covariance of parameters
15:
16:     for $n \leftarrow o + 1, N$ do
17:         $\boldsymbol{y}_n = \boldsymbol{Y}(n, :) = \boldsymbol{s}(n - o : n - 1)$     ▷ Input is last $o$ months' index
18:
19:         $\boldsymbol{P} = \boldsymbol{A}\boldsymbol{P}\boldsymbol{A}^t + \boldsymbol{Q}$     ▷ Uncertainity on parameters increases
20:
21:         $\boldsymbol{e}(n) = \boldsymbol{s}(n)$ - $\boldsymbol{w}^t\boldsymbol{y}_n$     ▷ Prediction Error
22:         $\boldsymbol{K} = \boldsymbol{P}\boldsymbol{y}_n/(R + \boldsymbol{y}_n^t\boldsymbol{P}\boldsymbol{y}_n)$     ▷ Kalman Gain
23:
24:         $\boldsymbol{w} = \boldsymbol{W}(n, :) = \boldsymbol{w} + \boldsymbol{K}\boldsymbol{e}(n)$     ▷ Correction
25:         $\boldsymbol{P} = (\boldsymbol{I}_o - \boldsymbol{K}\boldsymbol{y}_n^t)\boldsymbol{P}$
```

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.

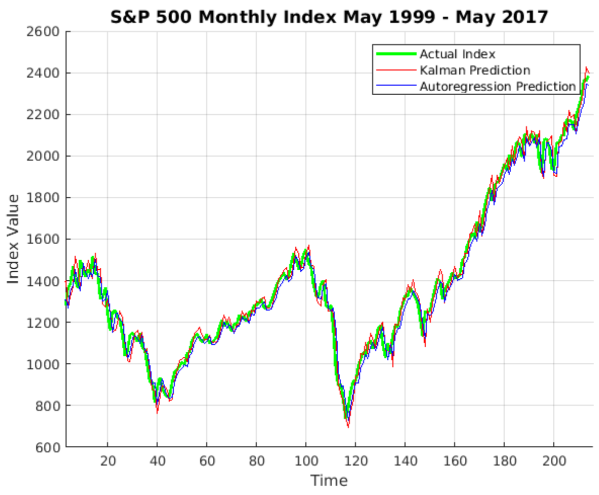
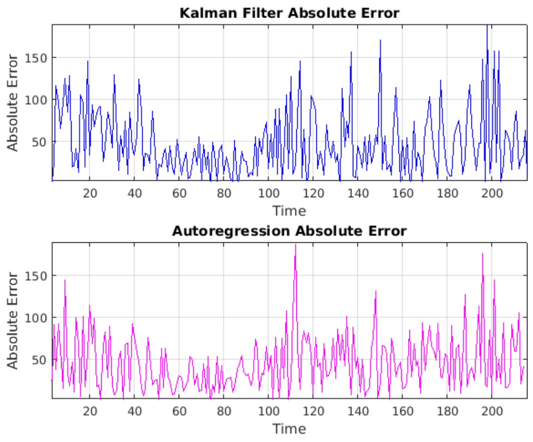
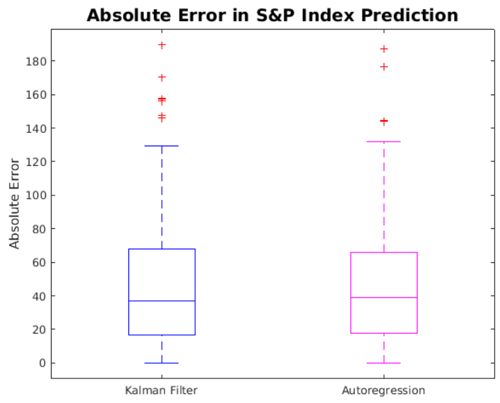


Figure 1: Kalman & Autoregression Prediction



(a) Index Prediction Absolute Error



(b) Index Prediction Error Comparison

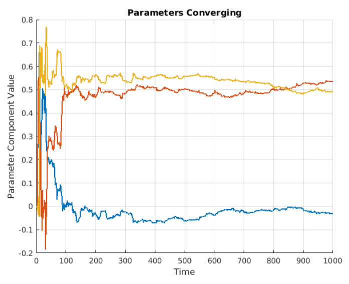
Figure 2: Prediction Error

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.

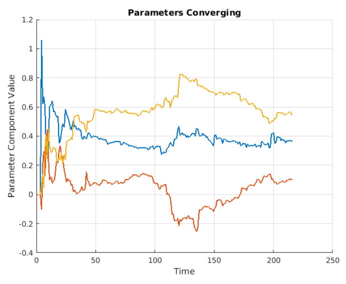
Figure 1 is showing the monthly index values, as well as prediction made from an autoregression and the implemented Kalman filter. Clearly both are very close to the true values of the index. We can also see from Figure 2a that both methods produce very similar error patterns. Figure 2b shows that the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

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

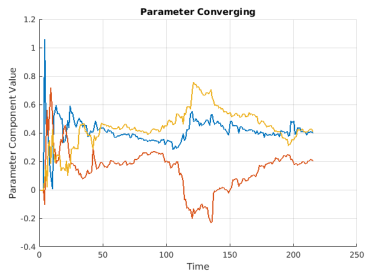
Now we can also observe from Figure 4a that error changes as the choice of α changes. The α choice is rather arbitrary and needs tuning. As I mentioned earlier, I chose $\alpha = 10^{-3}$ for all orders, since it seemed reasonable at most orders. Similarly, I plotted the error by varying $R = (20 : 200 : 2000)$. 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

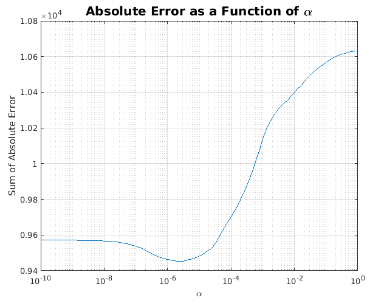


(b) $R \approx 53$

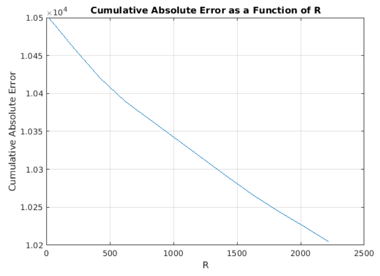


(c) $R = 420$

Figure 3: Parameter Convergence

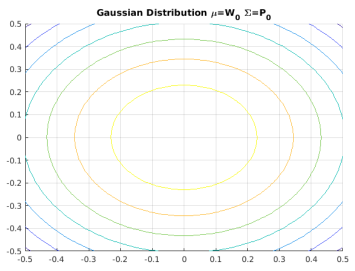


(a) $\boldsymbol{e} = f(\alpha)$

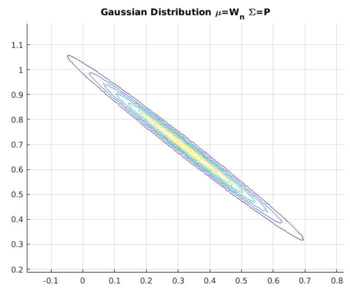


(b) $\boldsymbol{e} = f(R)$

Figure 4: Tuning α and R



(a) Initial Parameter Contour



(b) Final Parameter Contour

Figure 5: Parameter Contour Updates

Finally, from Figure 5, we can clearly see the updates on the parameters. This diagram shows the spread of the parameters at the start and at the end of the kalman filter. We can clearly see how the spread has been shrinked into the final shape from the initial uniform shape. This is exactly as expected.

Overall, the benefit of building a kalman filter to deal with financial time series is that we can isolate the noise and tune the process of doing so. At the same time we receive a distribution of our solution, which tells us the confidence in the parameters. This way we may be able to make better judgements. However, we are yet to explain the residual noise.

2 Predicting the Residual in Kalman Filter

The paper tries to explain the residual in the Kalman filter by the following variables - S&P 500 Price to Earnings Ratio (PER), Spot Oil Price, West Texas Intermediate (OIL), NAPM, ISM Manufacturing, Purchasing Managers' Composite Index (PMI), Disposable Personal Income (INCOME), Corporate Profits After Tax (CORP PROFIT), US Population (POPULATION) and US Unemployment Rate (UNEMPLOYMENT). The monthly data (May 1999 - February 2017) for these were collected from [2]. It was normalised via `zscore` function of Matlab's Statistics and Machine Learning Toolbox. The missing months' data were padded by the last month's data. The S&P 500 index, Kalman Filter and Augoregression of 2nd order were also calculated.

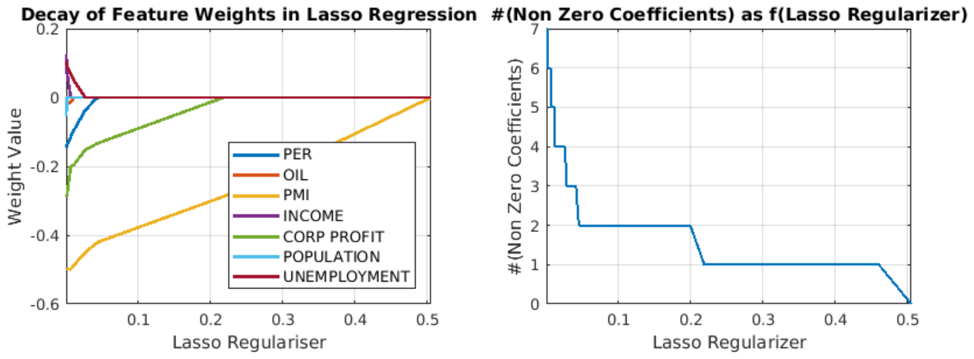
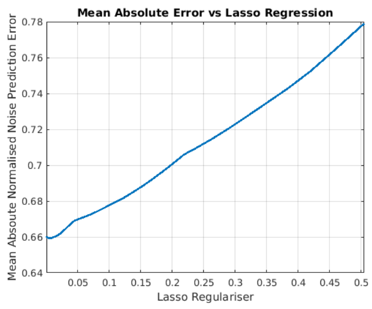


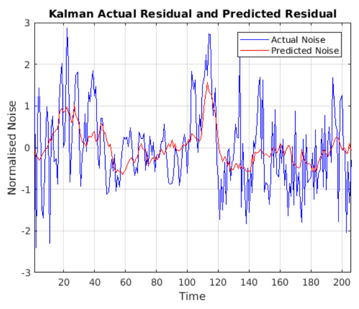
Figure 6: Decaying Features in Lasso Regulariser

Matlab's `lasso` function was used to run a lasso regularisation with 5 fold validation. The input was a 7 column matrix with the features mentioned in the last paragraph. The target output was the residual from a 2nd order Kalman Filter index prediction. It ran a lasso regularisation by solving the optimisation $min_{w_0,w}(\frac{1}{2N}\Sigma_{i=1}^{N}(y_i - w_0 - x_i^T w)^2 + \lambda\Sigma_{j=1}^{p}|w_j|)$. The optimisation was done for λ in range of $(0, 0.5)$ for 100 values. The values for w_0 were approximately 10^{-6}. So, its effect can be ignored.

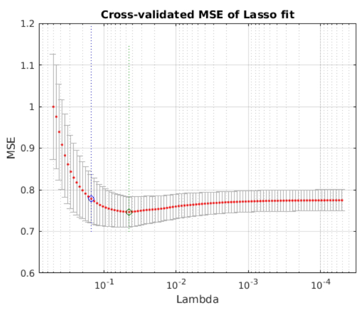
As expected (from Figure 6), the number of non-zero weights for the features reduced as λ or the regularisation penalty increased. It seems that it was most stable for 1 or 2 features. From Figure 6 left, we can see that PMI and CORP PROFIT were the last two features to be eliminated. Also, from Figure 7a we can see that, the least error was when λ was set to 0, i.e. we only wanted to match the target residual. As usual the choice of λ is one of tuning. If we want the weights to be sparse (as we do), we would like to choose a λ that will set all but 2 or 3 weights to 0.



(a) Lambda vs Noise Prediction Error



(b) Kalman Noise Prediction by Lasso



(c) Lasso Plot with Cross-Validated Fits

Figure 7: Lasso Plots

If the entire residual data is used for the lasso regression, we can fit the residual fairly well (Figure 7b). I also plotted the cross validated fit in Figure 7c. The blue dot is the value of λ that gives the min MSE (Mean Squared Error). The green dot is one standard deviation away. This plot gives us an idea of how the residual prediction progressed as λ values were changed.

However, our final goal is to not just to run the lasso regression on all of the residual data in one go. Just like we performed a 2nd order Kalman Filter or Autoregression, we would like to only predict the residual error by performing the lasso regularisation over the previous 2 months' features. Since at this point, we are confident that our lasso set up works and effectively predicts the residual, we only supply as input the past 2 month' features for each subsequent month from July 1999.

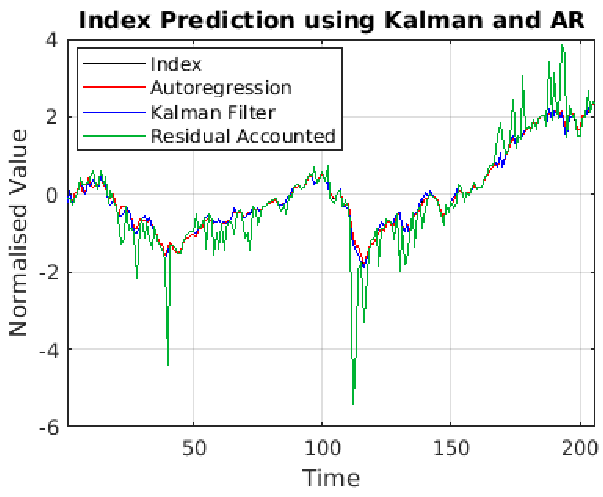


Figure 8: Two Months Lagged Residual Prediction Added to 2nd Order Kalman Filter Prediction

Just as expected, the weights for the features for each month were sparse, leaving only 1-3 non-zero values - the PMI, CORP PROFIT and PER. I added the predicted residual to Kalman Filter's index prediction. Finally I plotted the actual index values and all the predictions in Figure 8. It still predicts fairly well. However, it has to be noted that the residual matching is sensitive around extreme times. Take the crests in time series for example. At the extremes when we see upward trends for extended periods, it does not forecast well (consider the end of the time series). This precisely coincides with the claim made in the paper. It is more prominent if we use, for each month, all previous features data as input to the lasso regularisation in Figure 9. I believe at that point the exogenous features try to overtake the index data in the recent window, i.e. it amplifies the 'good fortune'. We would need to be cautionary in these cases.

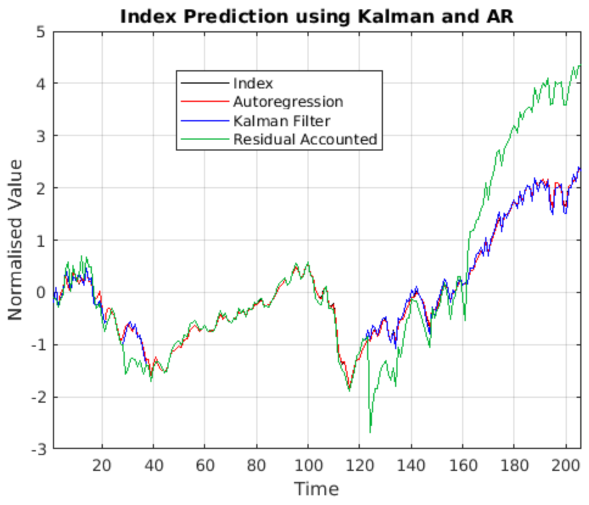


Figure 9: All Previous Months' Features are Used in Residual Forecast

Finally, it can be said that we can efficiently forecast the S&P 500 index from a recent window of previous index data. When a Kalman Filter is used, we can also try to model the residual using exogenous financial data, so that we can have more information in making financial decision. It must be said that the methods require regular tuning of certain parameters and may not work well in extreme upheavals. But it does give us a mathematical framework in efficiently explaining the financial markets. This is the point of computational finance.

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.

[2] Federal Reserve Bank of St. Louis, "Economic Research," at *https://research.stlouisfed.org/* [ONLINE - Last Accessed 19 May 2017].