1 Kalman Filter on Stationary AR Process Parameter Estimation

Figure 1 shows the convergence of the Kalman filter on the problem of estimating a second order AR process parameters. Irrespective of the initial conditions, it takes about 200 iterations to converge to the true parameters. The estimates never settle and randomly move around the true values.

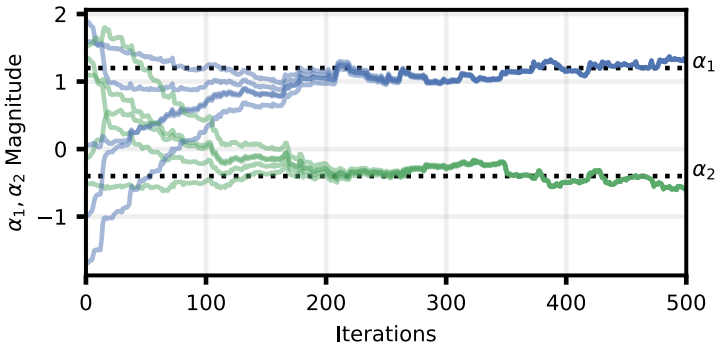

Figure 1. Convergence of the Kalman filter on AR parameter estimation for 5 different initial conditions. AR parameters are $\alpha_1 = 1.2, \alpha_2 = -0.4$. The Kalman filter parameters are $\beta = 10^{-4}, R = 0.2$.

In the next experiment, the behaviour of the filter was studied for different settings of the process and measurement noise covariances $\boldsymbol{Q}$ and R, where $\boldsymbol{Q} = \beta \boldsymbol{I}$. It was found experimentally that the ratio of β and R affect the convergence rather than their actual values. Hence, instead of changing each parameter individually, β was fixed at 10^{-4} and the ratio $r = \beta / R$ was varied (Figure 2). The results show that as the ratio increases, so does the convergence speed. However, the estimates become noisier and more unstable.

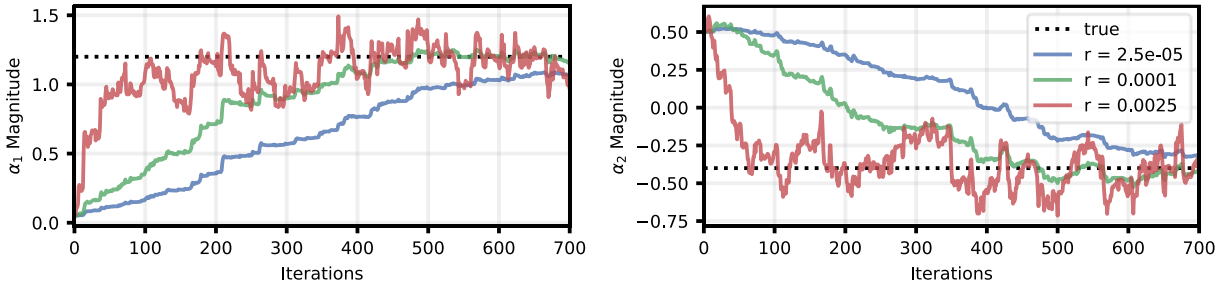

Figure 2. Convergence of the Kalman filter on stationary AR parameter estimation for different ratios r.

2 Particle Filter on Non-Stationary AR Process Parameter Estimation

The parameters for the non-stationary AR process have been chosen as described in Figure 3, to maximise the parameter dynamics and show the ability of the particle filter to find fast changing parameters, which are described by a non-linear function.

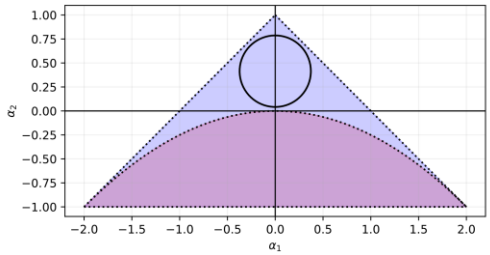

Figure 3. According to [1], an AR signal will be non-explosive (NE) and non-oscilatory (NO) when the parameters are in the blue region, non-explosive and oscilatory in the red region, and explosive everywhere else. To maximise the parameter dynamics whilst still having a NE-NO signal, α_1 and α_2 have been defined by $\alpha_1(n) = 0.9\, r \cos(2\pi n/N)$ and $\alpha_2(n) = r + 0.9\, r \sin(2\pi n/N)$, where $r = 1/(1+\sqrt{2})$ and N – the number of iterations (this corresponds to the circle in the figure).

First, the problem of degeneracy in Algorithm 1 of [2] is shown (Figure 4). There are a few indicators to the problem: 1) the effective number of particles N_{eff} (1) drops to 1 from 100; 2) all particles have a near-zero weight apart from one which has a near-one weight; 3) the mean of all particles remains unchanged throughout the iterations.

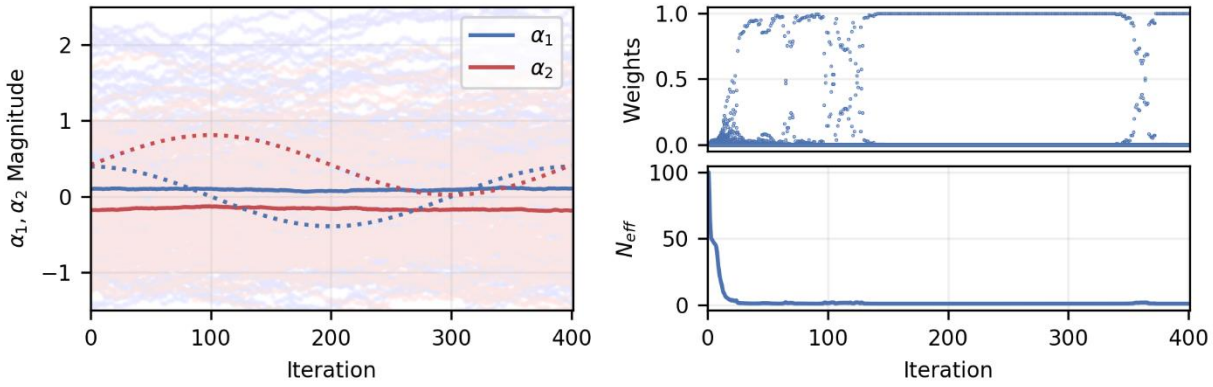

Figure 4. A particle filter tracking the parameters of a non-stationary AR process (100 particles; $\beta = 0.02$; $R = 1$). The left figure shows the true parameters (dotted), the individual particles (solid, translucent), and the average of all particles (solid, opaque). Taking a weighted average (not shown) yields slightly better predictions of the parameters, but the effect of degeneracy still persists. The top right is a scatter plot of the weights for all particles, and the bottom right shows the effective number of particles throughout the iterations.

To remedy the effect of degeneracy, a resampling step (Algorithm 2 in [2]) is performed every time the effective number of particles (1) drops below a certain threshold $\beta_r N_p$, where N_p is the total number of particles and $\beta_r \in [0,1]$ is a constant (Algorithm 3 in [2]).

$$N_{eff} = \frac{1}{\sum_{i=1}^{N_p} (\omega_i)^2} \tag{1}$$

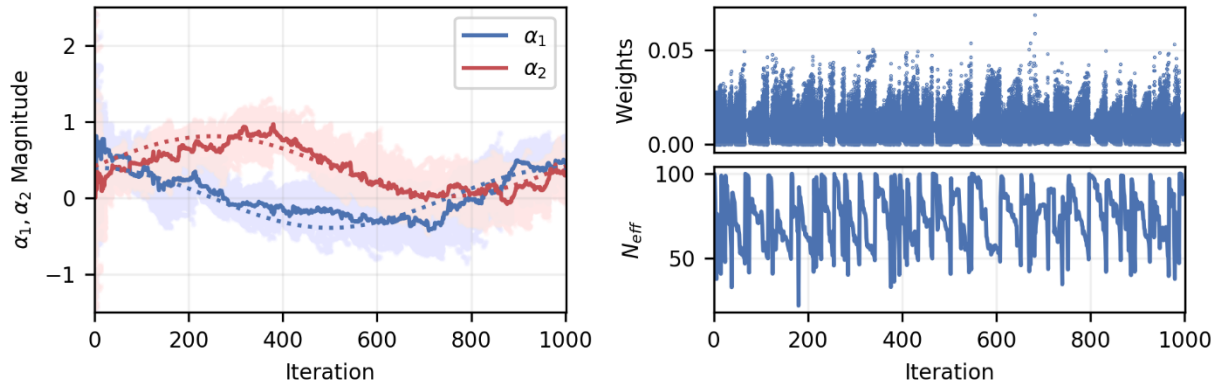

Figure 5. A particle filter with resampling, tracking the parameters of a non-stationary AR process (100 particles; $\beta = 0.02$; $R = 1$; $\beta_r = 0.5$). The meanings of the plots is the same as in Figure 4 (apart from using a weighted average for the particles).

The results of this remedy for $\beta_r = 0.5$ are shown in Figure 5. There are a few indicators that show the effect of the remedy: 1) the effective number of particles N_{eff} gradually and periodically drops to 50, after which resampling is triggered and it goes back to 100; 2) at each resampling, the particle weights are reset to $1/N_p$; 3) the weighted mean particle successfully tracks the time-varying AR parameters.

3 Extended Kalman Filter for Logistic Regression

The extended Kalman filter (EKF) is described by Algorithm 1 below. The difference from the normal Kalman filter is that the derivative of the measurement update function is used in lines 8,9 and 12, instead of the function itself [3], [4]. This results in a linear approximation of the function from the first term of its Taylor series expansion.

Algorithm 1. EKF

1	**Input:** $\mathbf{X}, \boldsymbol{y}, \beta, R$	7	$\boldsymbol{P}_{(n\|n-1)} = \boldsymbol{P}_{(n-1\|n-1)} + \boldsymbol{Q}$
2	$\boldsymbol{P}_{(0\|-1)} = \boldsymbol{P}_{(0\|0)} = 0.001\ \boldsymbol{I}^{d+1\times d+1}$	8	$\boldsymbol{S} = R + \hat{\boldsymbol{\sigma}}_{\boldsymbol{\theta}}^{T}\ \boldsymbol{P}_{(n\|n-1)}\ \hat{\boldsymbol{\sigma}}_{\boldsymbol{\theta}}$
3	$\boldsymbol{\theta}_{(0\|-1)} = \boldsymbol{\theta}_{(0\|0)} = \mathbf{0}^{d+1\times d+1}$	9	$\boldsymbol{k} = \boldsymbol{P}_{(n\|n-1)}\ \hat{\boldsymbol{\sigma}}_{\boldsymbol{\theta}}\ \boldsymbol{S}^{-1}$
4	$\boldsymbol{Q} = \beta\ \boldsymbol{I}^{d+1\times d+1}$	10	$e = y_{(n)} - \hat{\sigma}$
5	**for** $n = 1, \dots, N$ **do**	11	$\boldsymbol{\theta}_{(n\|n)} = \boldsymbol{\theta}_{(n\|n-1)} + \boldsymbol{k}\ e$
6	$\boldsymbol{\theta}_{(n\|n-1)} = \boldsymbol{\theta}_{(n-1\|n-1)}$	12	$\boldsymbol{P}_{(n\|n)} = \left(\boldsymbol{I} - \boldsymbol{k}\ \hat{\boldsymbol{\sigma}}_{\boldsymbol{\theta}}^{T}\right)\ \boldsymbol{P}_{(n\|n-1)}$

where

$$\hat{\sigma} = \frac{1}{1 + e^{-\boldsymbol{\theta}^{T}(n|n-1)^{\boldsymbol{x}_{(n)}}}} \tag{2}$$

$$\hat{\boldsymbol{\sigma}}_{\boldsymbol{\theta}} = \hat{\sigma}(1-\hat{\sigma})x_{(n)} \tag{3}$$

One way to observe the dynamics of the decision boundary as data comes in is shown in Figure 6, left. However, we could also observe how the individual components of $\boldsymbol{\theta}$ change. The parameter to be learnt is $\boldsymbol{\theta} = [\theta_0 \quad \theta_1 \quad \theta_2]^T$. In the two-dimensional case, the decision boundary line for input $\boldsymbol{x} = [x_1 \quad x_2]^T$ is defined by:

$$x_2 = \frac{-\theta_1 x_1 - \theta_0}{\theta_2} \tag{4}$$

Because of the way the data was generated, it is reasonable to assume that the best decision boundary is defined by $x_2 = x_1$. In other words, we could assume that $\boldsymbol{\theta}$ should converge, such that $\theta_1/\theta_2 = -1$ and $\theta_0/\theta_2 = 0$ (shown in Figure 6, right).

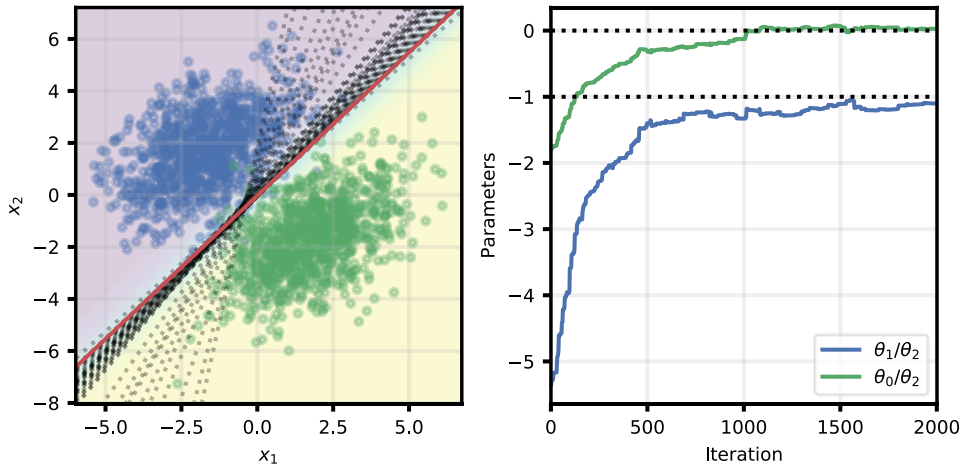

Figure 6. The left plot shows the dynamic of the decision boundary every 50 iterations (dashed black lines), and the final decision boundary (red line). The right plot shows the dynamics of the different components of $\boldsymbol{\theta}$. The class means are defined by $\alpha = 1.7$. The EKF parameters were empirically set to $\beta = 10^{-4}$ and $R = 0.2$.

Figure 7 shows how the classification accuracy of the logistic regression changes for different class distances (defined by α). At $\alpha = 0$ the classifier makes random guesses. By contrast, it achieves full accuracy, when $\alpha > 2$. An interesting observation was that the extended Kalman filter sometimes fails to optimise $\boldsymbol{\theta}$, when the classes are far apart (e.g. $\alpha > 5$, and the classifier is 100% wrong).

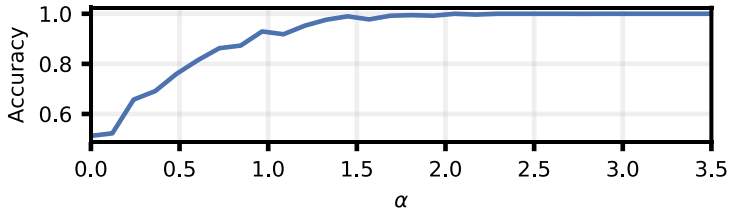

Figure 7. Accuracy (avg. of 10 runs) of learnt logistic regression parameters for different between-class distances.

4 Online Principal Component Analysis

I will comment on the use of OPCA [5] by considering the URL Reputation Dataset [6], which consists of 2.4 million examples and 3.2 million features. The dataset was designed to train real-time classifiers to detect spam, phishing, etc. Since URLs evolve over time, the classifier should be trained online. Due to the scale of the internet (many new examples are continiously generated) and the large number of features, it is impractical to use the data directly. Benefit 1) – OPCA would allow for a reduction in the number of features of data as it arrives, and would potentially remove noise by only preserving the directions with high variance from the data.

Algorithm 2 from [5] provides an efficient implementation of OPCA. During initialization, the user specifies an accuracy parameter $\epsilon \in (0, 1/15)$, and a target dimension k. However, the actual output dimension will be k/ϵ^3; i.e. there is a trade-off between accuracy and dimensionality reduction (due to the online nature). As a concrete example, suppose $\epsilon = 1/16$ and $k = 2$. The algorithm would reduce the dimensionality from 3.2 million to 8192. This should hint that OPCA is only useful at high dimensions. Benefit 2) – since OPCA processes examples one at a time, it is not required to store the full dataset in memory (if at all possible); the space complexity of the algorithm is $\mathcal{O}(dk/\epsilon^3)$, where d is the original dimension size. I.e. the space required is independent of the size of the dataset. Benefit 3) – depending on the time complexity of the classifier, OPCA will likely provide computational savings too as it only spends $\mathcal{O}(dk/\epsilon^3)$ floating point operations per new input. Subsequent model will work with much less dimensions (k/ϵ^3).

5 References

[1] P. J. Robinson and A. Zellner, "An Introduction to Bayesian Inference in Econometrics," *J. Mark. Res.*, 1974.

[2] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, 2002.

[3] M. Niranjan, "Sequential Bayesian computation of logistic regression models," 2008.

[4] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," *In Pract.*, 2006.

[5] C. Boutsidis, D. Garber, Z. Karnin, and E. Liberty, "Online principal components analysis," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2015.

[6] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URLs: an application of large-scale online learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 681–688.