Neurocomputing 80 (2012) 47–53

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Model based learning of sigma points in unscented Kalman filtering

Ryan Turner*, Carl Edward Rasmussen

University of Cambridge, Department of Engineering, Trumpington Street, Cambridge CB2 1PZ, UK

ARTICLE INFO

Available online 6 November 2011

Keywords:
Unscented Kalman filtering
Sigma points
State-space
Machine learning
Global optimization
Gaussian process

ABSTRACT

The unscented Kalman filter (UKF) is a widely used method in control and time series applications. The UKF suffers from arbitrary parameters necessary for sigma point placement, potentially causing it to perform poorly in nonlinear problems. We show how to treat sigma point placement in a UKF as a learning problem in a model based view. We demonstrate that learning to place the sigma points correctly from data can make sigma point collapse much less likely. Learning can result in a significant increase in predictive performance over default settings of the parameters in the UKF and other filters designed to avoid the problems of the UKF, such as the GP-ADF. At the same time, we maintain a lower computational complexity than the other methods. We call our method UKF-L.

1. Introduction

Filtering in linear dynamical systems (LDS) and nonlinear dynamical systems (NLDS) is frequently used in many areas, such as signal processing, state estimation, control, and finance/econometric models. Filtering (inference) aims to estimate the state of a system from a stream of noisy measurements. Imagine tracking the location of a car based on odometer and global positioning system (GPS) sensors, both of which are noisy. Sequential measurements from both sensors are combined to overcome the noise in the system and to obtain an accurate estimate of the system state. Even when the full state is only partially measured, it can still be inferred; in the car example the engine temperature is unobserved, but can be inferred via the nonlinear relationship from acceleration. To exploit this relationship appropriately, inference techniques in nonlinear models are required; they play an important role in many practical applications.

LDS and NLDS belong to a class of models known as state-space models. A state-space model assumes that there exists a sequence of latent states $\mathbf{x}_t$ that evolve over time according to a Markovian process specified by a transition function f. The latent states are observed indirectly in $\mathbf{y}_t$ through a measurement function g. We consider state-space models given by

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}) + \boldsymbol{\epsilon}, \quad \mathbf{x}_t \in \mathbb{R}^M,$$

$$\mathbf{y}_t = g(\mathbf{x}_t) + \boldsymbol{\nu}, \quad \mathbf{y}_t \in \mathbb{R}^D. \tag{1}$$

Here, the system noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\epsilon)$ and the measurement noise $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\nu)$ are both Gaussian. In the LDS case, f and g are linear functions, whereas the NLDS covers the general nonlinear case.

Kalman filtering [1] corresponds to exact (and fast) inference in the LDS, which can only model a limited set of phenomena. For the last few decades, there has been interest in NLDS for more general applicability. In the state-space formulation, nonlinear systems do not generally yield analytically tractable algorithms.

The most widely used approximations for filtering in NLDS are the extended Kalman filter (EKF) [2], the unscented Kalman filter (UKF) [3], and the cubature Kalman filter (CKF) [4]. The EKF linearizes f and g at the current estimate of $\mathbf{x}_t$ and treats the system as a nonstationary linear system even though it is not. The UKF and CKF propagate several estimates of $\mathbf{x}_t$ through f and g and reconstructs a Gaussian distribution assuming the propagated values came from a linear system. The locations of the estimates of $\mathbf{x}_t$ are known as the *sigma points*. Many heuristics have been developed to help set the sigma point locations [5]. Unlike the EKF, the UKF has free parameters that determine where to put the sigma points.

Our contribution is a strategy for improving the UKF through a novel learning algorithm for appropriate sigma point placement: we call this method UKF-L. The key idea in the UKF-L is that the UKF and EKF are doing exact inference in a model that is somewhat "perverted" from the original model described in the state space formulation. The interpretation of EKF and UKF as models, not merely approximate methods, allows us to better identify their underlying assumptions. This interpretation also enables us to *learn* the free parameters in the UKF in a model based manner from training data. If the settings of the sigma point are a poor fit to the underlying dynamical system, the UKF can make horrendously poor predictions.

2. Unscented Kalman filtering

We first review how filtering and the UKF works and then explain the UKF's generative assumptions. Filtering methods consist

* Corresponding author. Tel.: +44 1223 748511; fax: +44 1223 332662.
E-mail addresses: rt324@cam.ac.uk (R. Turner), cer54@cam.ac.uk (C.E. Rasmussen).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.07.029

of three steps: time update, prediction step, and measurement update. They iterate in a predictor-corrector setup. In the time update we find $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\, d\mathbf{x}_{t-1} \tag{2}$$

using $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$. In the prediction step we predict the observed space, $p(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ using $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})\, d\mathbf{x}_t. \tag{3}$$

Finally, in the measurement update we find $p(\mathbf{x}_t|\mathbf{y}_t)$ using information from how good (or bad) the prediction in the prediction step is

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1}). \tag{4}$$

In the linear case all of these equations can be done analytically using matrix multiplications. The EKF explicitly linearizes f and g at the point $\mathbb{E}[\mathbf{x}_t]$ at each step. The UKF uses the whole distribution on $\mathbf{x}_t$, not just the mean, to place sigma points and implicitly linearize the dynamics, which we call the *unscented transform* (UT). In one dimension the sigma points roughly correspond to the mean and α-standard deviation points; the UKF generalizes this idea to higher dimensions. The exact placement of sigma points depends on the unitless parameters $\{\alpha,\beta,\kappa\} \in \mathbb{R}^+$ through

$$\mathcal{X}^0 \coloneqq \boldsymbol{\mu}, \quad \mathcal{X}^i \coloneqq \boldsymbol{\mu} \pm (\sqrt{(D+\lambda)\boldsymbol{\Sigma}})_i, \tag{5}$$

$$\lambda \coloneqq \alpha^2(D+\kappa)-D, \tag{6}$$

where $\sqrt{\cdot}_i$ refers to the ith row of the Cholesky factorization.[1] The sigma points have weights assigned by

$$w_m^0 \coloneqq \frac{\lambda}{D+\lambda}, \quad w_c^0 \coloneqq \frac{\lambda}{D+\lambda} + (1-\alpha^2+\beta), \tag{7}$$

$$w_m^i \coloneqq w_c^i \coloneqq \frac{1}{2(D+\lambda)}, \tag{8}$$

where w_m is used to reconstruct the predicted mean and w_c used for the predicted covariance. We interpret the unscented transform as approximating the input distribution by $2D+1$ point masses at $\mathcal{X}$ with weights $\mathbf{w}$. Once the sigma points $\mathcal{X}$ have been calculated the filter accesses f and g as black boxes to find $\mathcal{Y}_t$, either $f(\mathcal{X}_t)$ or $g(\mathcal{X}_t)$ depending on the step. The UKF reconstructs a Gaussian predictive distribution with the mean and covariance determined from $\mathcal{Y}_t$ pretending the dynamics *had* been linear. In other words, the equation used to find the approximating Gaussian is such that the UKF is exact for linear dynamics model f and observation model g. It does not guarantee the moments will match the moment of the true non-Gaussian distribution. The UKF is a *black box filter* as opposed to the EKF which requires derivatives of f and g.

Both the EKF and the UKF approximate the nonlinear state-space as a nonstationary linear system. The UKF defines its own *generative process*, which linearizes the nonlinear functions f and g wherever in $\mathbf{x}_t$ a UKF filtering the time series would expect $\mathbf{x}_t$ to be. Therefore, it is possible to sample synthetic data from the UKF by sampling from its one-step-ahead predictions as seen in Algorithm 1. The sampling procedure augments the filter: predict-sample-correct. If we use the UKF with the same $\{\alpha,\beta,\kappa\}$ used to generate synthetic data, then the one-step-ahead predictive distribution will be the exact same distribution the data point was sampled from. Note that in an LDS, if we sample from the one-step-ahead predictions of a Kalman filter we are sampling from the same process as if we sampled from the latent states $\mathbf{x}$ and then sampled from the observation model to get $\mathbf{y}$.

[1] If $\sqrt{\mathbf{P}} = \mathbf{A} \Rightarrow \mathbf{P} = \mathbf{A}^\top\mathbf{A}$, then we use the rows in (5). If $\mathbf{P} = \mathbf{A}\mathbf{A}^\top$, then we use the columns.

Algorithm 1. Sampling data from UKF's implicit model

1: $p(\mathbf{x}_1|\emptyset) \leftarrow (\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$
2: **for** $t=1$ to T **do**
3: Prediction step: $p(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ using $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$
4: Sample $\mathbf{y}_t$ from prediction step distribution
5: Measurement update: $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ using $\mathbf{y}_t$
6: Time update: find $p(\mathbf{x}_{t+1}|\mathbf{y}_{1:t})$ using $p(\mathbf{x}_t|\mathbf{y}_{1:t})$
7: **end for**

2.1. Setting the parameters

We summarize all the parameters as $\theta \coloneqq \{\alpha,\beta,\kappa\}$. For any setting of θ the UKF will give identical predictions to the Kalman filter if f and g are both linear. Many of the heuristics for setting θ assume f and g are linear (or close to it), which is not the problem the UKF solves. For example, one of the heuristics for setting θ is that $\beta = 2$ is optimal if the state distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is exactly Gaussian [6]. However, the state distribution will seldom be Gaussian unless the system is linear, in which case any setting of θ gives exact inference! It is often recommended to set the parameters to $\alpha = 1$, $\beta = 0$, and $\kappa = 3-D$ [7,8]. The CKF can be constructed as a special case of a UKF with $\alpha = 1$, $\beta = 0$, and $\kappa = 0$.

3. The Achilles' heel of the UKF

The UKF can have embarrassingly poor performance because its predictive variances can be far too small if the sigma points are placed in unlucky locations. Insufficient predictive variance will cause observations to have too much weight in the measurement update, which causes the UKF to fit to noise. Meaning, the UKF will perform poorly even when evaluated on root-mean-square-error (RMSE), which only uses the predictive mean. On the NLL, the situation is even worse where too small predictive variances are heavily penalized.

In the most extreme case, the UKF can give a delta spike predictive distribution. We call this *sigma point collapse*. As seen in Fig. 1, when the sigma points are arranged together horizontally the UKF has no way to know the function varies anywhere. We aim to learn the parameters θ in such a way that collapse

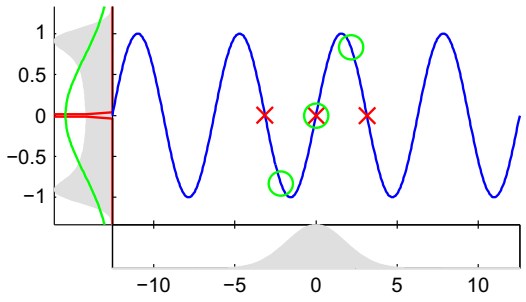

Fig. 1. An illustration of a good and bad assignment of sigma points. The lower panel shows the true input distribution. The center panel shows the sinusoidal system function f (blue) and the sigma points for $\alpha = 1$ (red crosses) and $\alpha = 0.68$ (green rings). The left panel shows the true predictive distribution (shaded), the predictive distribution under $\alpha = 1$ (red spike) and $\alpha = 0.68$ (green). Using a different set of sigma points we get either a completely degenerate solution (a delta spike) or a near optimal approximation within the class of Gaussian approximations. The parameters β and κ are fixed at their defaults in this example. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

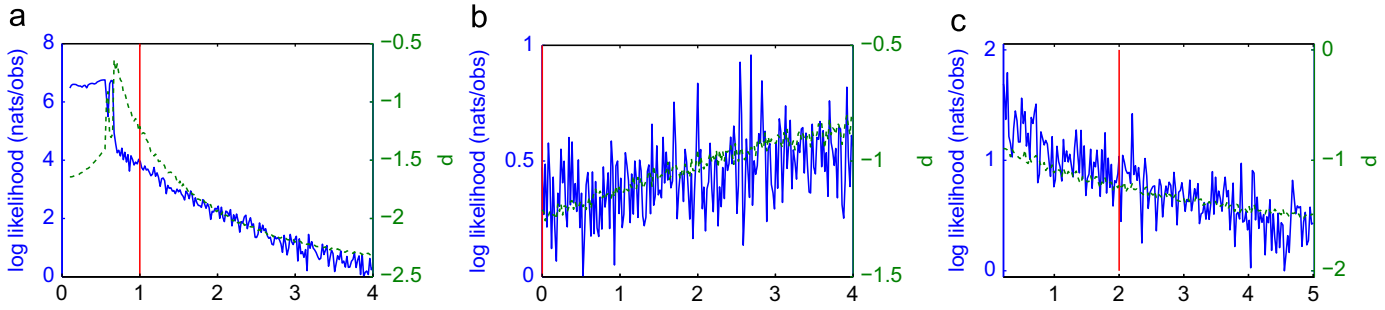

Fig. 2. Illustration of the UKF when applied to a pendulum system. Cross-section of the marginal likelihood (blue line) varying the parameters one at a time from the defaults (red vertical line). We shift the marginal likelihood, in nats per observation, to make the lowest value zero. The dashed green line is the total variance diagnostic $d \coloneqq \mathbb{E}[\log(|\boldsymbol{\Sigma}|/|\boldsymbol{\Sigma}_0|)]$, where $\boldsymbol{\Sigma}$ is the predictive variance in one-step-ahead prediction. We divide out the variance $\boldsymbol{\Sigma}_0$ of the time series when treating it as iid to make d unitless. Values of θ with small predictive variances closely track the θ with low marginal likelihood. Note that the "noise" is a result of the sensitivity of the predictions to parameters θ due to sigma point collapse, not randomness in the algorithm as is the case with a particle filter. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

becomes unlikely. Anytime collapse happens in training the marginal likelihood will be substantially lower. Hence, the learned parameters will avoid anywhere this delta spike occurred in training. Maximizing the marginal likelihood is tricky since it is not well-behaved for settings of θ that cause sigma point collapse.

4. Model based learning

A common approach to estimating model parameters θ in general is to maximize the log marginal likelihood

$$\ell(\theta) \coloneqq \log p(\mathbf{y}_{1:T}|\theta) = \sum_{t=1}^{T} \log p(\mathbf{y}_t|\mathbf{y}_{1:t-1},\theta). \tag{9}$$

Hence we can equivalently maximize the sum from the one-step-ahead predictions. One might be tempted to apply a gradient based optimizer on (9), but as seen in Fig. 2 the marginal likelihood can be very unstable. The instability in the likelihood is likely the result of the phenomenon explained in Section 3, where a slight change in parameterization can avoid problematic sigma point placement. This makes the application of a gradient-based optimizer hopeless.

We could apply Markov chain Monte Carlo (MCMC) and integrate out the parameters. However, this is usually "over kill" as the posterior on θ is usually highly peaked unless T is very small. Tempering must be used as mixing will be difficult if the chain is not initialized inside the posterior peak. Even in the case when T is small enough to spread the posterior out, we would still like a single point estimate for computational speed on the test set.[2]

We focus on learning using a Gaussian process (GP) based optimizer [10,11]. Since the marginal likelihood surface has an underlying smooth function but contains what amounts to additive noise, a probabilistic regression method seems a natural fit for finding the maximum.

5. Gaussian process optimizers

Gaussian processes form a prior over functions. Estimating the parameters amounts to finding the maximum of a structured function: the log marginal likelihood. Therefore, it seems natural to use a prior over functions to guide our search. Given that Gaussian processes form a prior over functions we can use them for *global optimization*. The same principle has been applied to integration in Rasmussen and Ghahramani [12].

GP optimization (GPO) allows for effective *derivative free* optimization. We consider the maximization of a likelihood function $\ell(\theta)$. GPs allow for derivative information $\partial_\theta \ell$ to be included as well, but in our case that will not be very useful due to the function's instability.

GPO treats optimization as a sequential decision problem in a probabilistic setting, receiving reward r when using the right input θ to get a large function value output $\ell(\theta)$. This setup is also known as a Markov decision process (MDP) [13, Chapter 1]. At each step GPO uses its posterior over the objective function $p(\ell(\theta))$ to look for θ it believes has a large function value $\ell(\theta)$. A maximization strategy that is *greedy* will always evaluate the function $p(\ell(\theta))$ where the mean function $\mathbb{E}[\ell(\theta)]$ is the largest. A strategy that trades-off *exploration* with *exploitation* will take into account the posterior variance $\mathrm{Var}[\ell(\theta)]$. Areas of θ with high variance carry a possibility of having a large function value or high reward r. The optimizer is programmed to evaluate at the maxima of

$$J(\theta) \coloneqq \mathbb{E}[\ell(\theta)] + K\sqrt{\mathrm{Var}[\ell(\theta)]}, \tag{10}$$

where K is a constant to control the exploration exploitation trade-off. The objective J is known as the upper confidence bound (UCB), since it optimizes the maximum of a confidence interval on the function value. The UCB has recently been analyzed theoretically by Srinivas et al. [11]. The optimizer must also find the maximum of J, but since it is a combination of the GP mean and variance functions it is easy to optimize with gradient methods.

We summarize the method in Algorithm 2, the subroutine to compute J is shown in Algorithm 3.[3] GPO assumes that we provide a feasible set of θ to search within. We can do the optimization using ℓ as the objective h in Algorithm 2. Algorithm 3 optionally adds a barrier to make sure that new candidate points for evaluation stay within the feasible set. For instance, if the feasible set is the unit cube, the 20th norm $\|\theta\|_{20}$ forms an effective barrier. Alternatively, a constrained optimization routine could be used. Every iteration we consider another set of C candidate points to evaluate $\ell(\theta)$.

We illustrate the iterations of GPO in Fig. 3. The function in the figure is highly non-convex and the global approach of the GP helps greatly. We consider a feasible region of $\theta \in [0, 10]$ and initialize the search by evaluating the function at the edges and

[2] If we want to integrate the parameters out we must run the UKF with each sample of $\theta|\mathbf{y}_{1:T}$ during test and average. To get the optimal point estimate of the posterior we would like to compute the *Bayes' point* [9].

[3] We use the notation $\mathbf{A}\backslash\mathbf{y}$ equivalently to $\mathbf{A}^{-1}\mathbf{y}$ except that we expect the computation to be done using matrix back substitution rather than explicit calculation of the inverse and then multiplying.

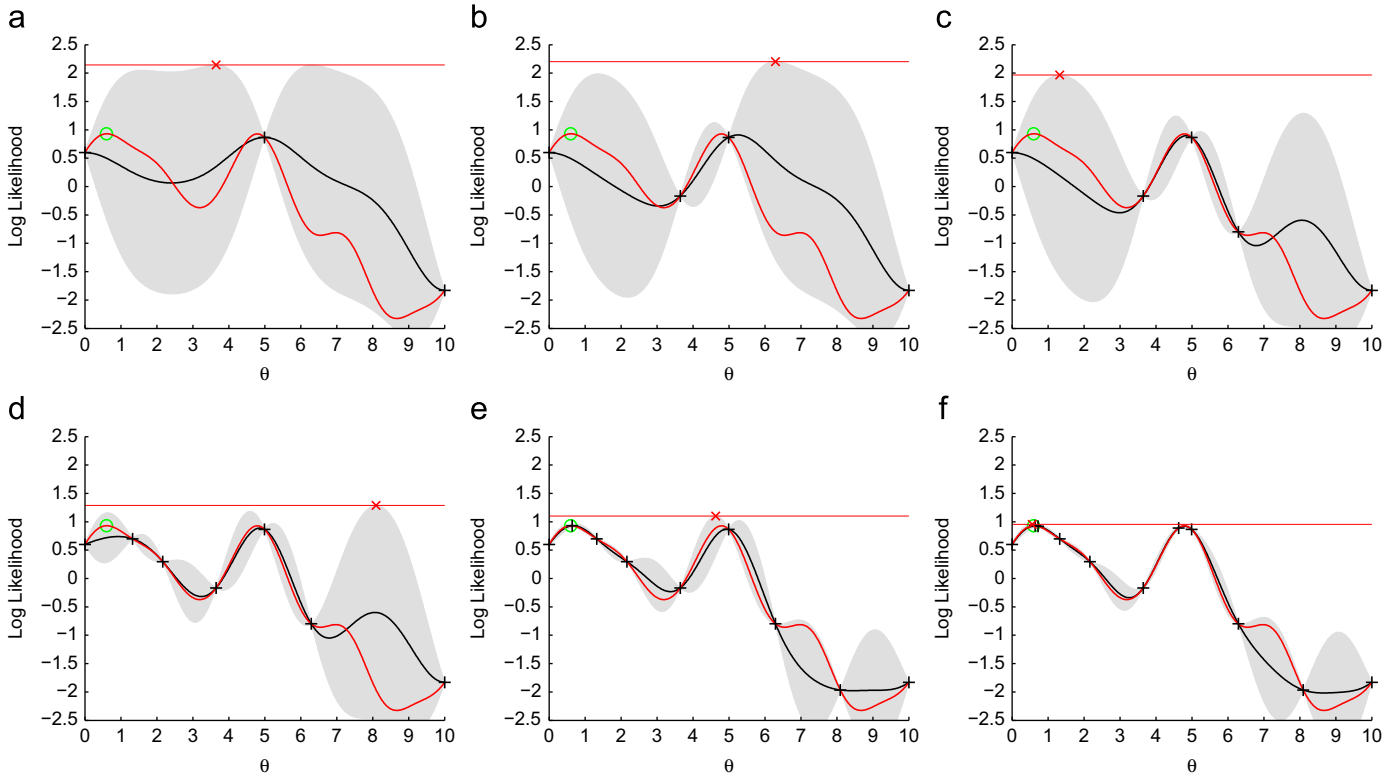

Fig. 3. Illustration of GPO in one dimension. The red line represents the (highly non-convex) function we are attempting to optimize with respect to its input θ. Its true maximum is shown by the green circle. The black line and the shaded region represent the GP predictive mean and two standard deviation error bars. The black + shows the points that have already been evaluated. The red $\times$ is the maximum of $J(\theta)$ shown by the red horizontal line. Here we use $K=2$ so the UCB criterion J corresponds to the top of the two standard deviation error bars shown in the plot. Every iteration GPO merely evaluates the function where the top of the error bars is highest. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the midpoint: {0,5,10}. The figure illustrates how the UCB criterion J trades-off exploitation with exploration. For instance, a purely explorative strategy would evaluate the function at $\theta=9$ in Fig. 3(f), since the error bars are the largest even though the optima does not occur there. Likewise, a purely (greedy) exploitative strategy would get stuck evaluating the function at $\theta=5$ in Fig. 3(a) and never discover the maximum at $\theta=1$.

Algorithm 2. Gaussian process optimization

function GPO(Range,h,C)
  Set E to be the dimension of inputs $\mathbf{x}$
  ▷ Use end points and midpoints of feasible set
  $\mathbf{x} \leftarrow \text{multiGrid(Range)} \in \mathbb{R}^{E\times 3E}$
  ▷ Evaluate $h(\mathbf{x})$ at all $3\cdot E$ grid points
  $\mathbf{y} \leftarrow h(\mathbf{x}) \in \mathbb{R}^{3E}$
  for $i=1$ to maximum function evaluations **do**
    Pre-compute $\mathbf{L}$, Cholesky of GP cov. matrix
    for $j=1$ to C **do**
      ▷ Sample a random initial point
      $\mathbf{x}_0 \sim \mathcal{N}(\mathbb{E}[\mathbf{x}], \text{Cov}\,\mathbf{x}) \in \mathbb{R}^E$
      ▷ Maximize the UCB criterion J w.r.t. $\mathbf{x}_\star$
        ▷ $\mathbf{x}_\star$ initialized at $\mathbf{x}_0$
      $(\mathbf{S}_j, F_j) \leftarrow \max_{\mathbf{x}_\star} \text{UCB}(\mathbf{x}_\star, \mathbf{x}, \mathbf{y}, K, \mathbf{L})$
    end for
    ▷ Find best init. $z \in \{1,\ldots,C\}$ from list of candidates solutions $\mathbf{S}$ and values F
    append $\mathbf{S}(\text{argmax}_z F)$ to $\mathbf{x}$ ▷ $\mathbf{x}$ now $\mathbb{R}^{E\times 3E+i}$
    append $h(\mathbf{S}(\text{argmax}_z F))$ to $\mathbf{y}$ ▷ $\mathbf{y}$ now $\mathbb{R}^{3E+i}$
  end for
  Return $\mathbf{x}(\text{argmax}\,\mathbf{y})$
end function

Algorithm 3. Upper confidence bound

function UCB($\mathbf{x}_\star$,$\mathbf{x}$,$\mathbf{y}$,K,$\mathbf{L}$)
  ▷ compute $\mathbb{E}[\mathbf{x}_\star|\mathbf{x},\mathbf{y}]$ and $\text{Var}[\mathbf{x}_\star|\mathbf{x},\mathbf{y}]$ using a GP in numerically stable way
  find $\mathbf{K}_{\star\star} \in \mathbb{R}^+$ and $\mathbf{K}_{\mathbf{x},\star} \in \mathbb{R}^{|\mathbf{x}|\times 1}$
  $\alpha \leftarrow \mathbf{L}^\top \backslash \mathbf{L} \backslash \mathbf{y}$
  $\overline{h} \leftarrow \mathbf{K}_{\mathbf{x},\star}^\top \alpha$ ▷ $\mathbb{E}[\mathbf{x}_\star|\mathbf{x},\mathbf{y}]$
  $\beta \leftarrow \mathbf{L}^\top \backslash \mathbf{L} \backslash \mathbf{K}_{\mathbf{x},\star}$
  $h_{\text{sd}} \leftarrow \sqrt{\mathbf{K}_{\star\star} - \mathbf{K}_{\mathbf{x},\star}^\top \beta}$ ▷ $\sqrt{\text{Var}[\mathbf{x}_\star|\mathbf{x},\mathbf{y}]}$
  $J \leftarrow h + K h_{\text{sd}}$ ▷ find the UCB
  Add a barrier to J, e.g. $\|\mathbf{x}_\star\|_{20}$ ▷ Optional
  Compute dJ w.r.t. $\mathbf{x}_\star$
  Return J and its derivatives dJ
end function

6. Experiments and results

We test our method on filtering in three dynamical systems: the sinusoidal dynamics used in Turner et al. [14], the Kitagawa dynamics used in Deisenroth et al. [15], Kitagawa [16], and pendulum dynamics used in Deisenroth et al. [15]. The sinusoidal dynamics are described by

$$x_{t+1} = 3\sin(x_t) + w, \quad w \sim \mathcal{N}(0, 0.1^2), \tag{11}$$

$$y_t = \sigma(x_t/3) + v, \quad v \sim \mathcal{N}(0, 0.1^2), \tag{12}$$

where $\sigma(\cdot)$ represents a logistic sigmoid. The Kitagawa model is described by

$$x_{t+1} = 0.5x_t + \frac{25x_t}{1+x_t^2} + w, \quad w \sim \mathcal{N}(0, 0.2^2), \tag{13}$$

$$y_t = 5\sin(2x_t) + v, \quad v \sim \mathcal{N}(0, 0.01^2). \tag{14}$$

The Kitagawa model was presented as a filtering problem in Kitagawa [16]. The pendulum dynamics is described by a discretized ordinary differential equation (ODE) at $\Delta_t = 400$ ms. The pendulum possesses a mass $m = 1$ kg and a length $l = 1$ m. The pendulum angle φ is measured anti-clockwise from the upward position. The state $\mathbf{x} = [\varphi \ \dot{\varphi}]^{\mathrm{T}}$ of the pendulum is given by the angle φ and the angular velocity $\dot{\varphi}$. The ODE is

$$\frac{d}{dt}\begin{bmatrix}\dot{\varphi}\\ \varphi\end{bmatrix} = \begin{bmatrix}\dfrac{-mlg\sin\varphi}{ml^2}\\ \dot{\varphi}\end{bmatrix}, \tag{15}$$

where g is the acceleration of gravity. This model is commonly used in stochastic control for the inverted pendulum problem [17]. The measurement function is

$$\mathbf{y}_t = \begin{bmatrix}\arctan\left(\dfrac{p_1 - l\sin(\varphi_t)}{p_1 - l\cos(\varphi_t)}\right)\\ \arctan\left(\dfrac{p_2 - l\sin(\varphi_t)}{p_2 - l\cos(\varphi_t)}\right)\end{bmatrix}, \quad \begin{bmatrix}p_1\\ p_2\end{bmatrix} = \begin{bmatrix}1\\ -2\end{bmatrix}, \tag{16}$$

which corresponds to *bearings only measurement* since we do not directly observe the velocity. We use system noise $\mathbf{\Sigma}_w = \mathrm{diag}([0.1^2 \ \ 0.3^2])$ and $\mathbf{\Sigma}_v = \mathrm{diag}([0.2^2 \ \ 0.2^2])$ as observation noise.

For all the problems we compare the UKF-L with the UKF-D, CKF, EKF, GP-UKF, and GP assumed density filter (GP-ADF), and time independent model (TIM); we use UKF-D to denote a UKF with default parameter settings, and UKF-L for learned parameters. The TIM treats the data as iid normal and is inserted as a reference point. The GP-UKF and GP-ADF use GPs to approximate f and g and exploit the properties of GPs to make tractable predictions. The Kitagawa and pendulum dynamics were used by Deisenroth et al. [15] to illustrate the performance of the GP-ADF and the very poor performance of the UKF. Deisenroth et al. [15] used the default settings of $\alpha = 1$, $\beta = 0$, $\kappa = 2$ for all of the experiments; we use $\kappa = 3 - D$. We used exploration trade off $K = 2$ for the GPO in all the experiments. Additionally, GPO used the squared-exponential with automatic relevance determination (SE-ARD) covariance function,

$$k_\xi(\theta_i, \theta_j) = \sigma_0^2 \exp(-\tfrac{1}{2}(\theta_i - \theta_j)^\top \mathbf{M}(\theta_i - \theta_j)), \tag{17}$$

$$\mathbf{M} \coloneqq \mathrm{diag}(\ell)^{-1}, \tag{18}$$

$$\xi \coloneqq \{\sigma_0^2, \ell\} \in (\mathbb{R}^+)^{|\theta|+1} = (\mathbb{R}^+)^4, \tag{19}$$

plus a noise term with standard deviation 0.01 nats per observation. We set the GPO to have a maximum number of function evaluations of 100, even better results can be obtained by letting the optimizer run longer to hone the parameter estimate. We show that by *learning* appropriate values for θ we can match, if not exceed, the performance of the GP-ADF and other methods.

The models were evaluated on their one-step-ahead predictions. The evaluation metrics were the negative log-predictive likelihood (NLL), the mean squared error (MSE), and the mean absolute error (MAE) between the mean of the prediction and the true value. Note that unlike the NLL, the MSE and MAE do not account for uncertainty. The MAE will be more difficult for approximate methods than MSE. For MSE, the optimal action is to predict the mean of the predictive distribution, while for the MAE it is the median [18, Chapter 1]. Most approximate methods attempt to moment match to a Gaussian and preserve the mean; the median of the true predictive distribution is implicitly assumed to be the same as mean. Quantitative results are shown in Table 1.

Table 1
Comparison of the methods on the sinusoidal, Kitagawa, and pendulum dynamics. The measures are supplied with 95% confidence intervals and a p-value from a one-sided t-test under the null hypothesis UKF-L is the same or worse as the other methods. NLL is reported in nats per observation, while MSE and MAE are in the units of $\mathbf{y}^2$ and $\mathbf{y}$, respectively. Since the observations in the pendulum data are angles we projected the means and the data to the complex plane before computing MSE and MAE.

Method	NLL	p-Value	MSE	p-Value	MAE	p-Value
Sinusoid ($T=500$ and $R=10$)						
UKF-D	$10^{-1} \times -4.58 \pm 0.168$	< 0.0001	$10^{-2} \times 2.32 \pm 0.0901$	< 0.0001	$10^{-1} \times 1.22 \pm 0.0253$	< 0.0001
UKF-L*	$\mathbf{-5.53 \pm 0.243}$	N/A	$\mathbf{1.92 \pm 0.0799}$	N/A	$\mathbf{1.09 \pm 0.0236}$	N/A
EKF	-1.94 ± 0.355	< 0.0001	3.03 ± 0.127	< 0.0001	1.37 ± 0.0299	< 0.0001
CKF	-2.07 ± 0.390	< 0.0001	2.26 ± 0.100	< 0.0001	1.17 ± 0.0260	< 0.0001
GP-ADF	-4.13 ± 0.154	< 0.0001	2.57 ± 0.0940	< 0.0001	1.30 ± 0.0261	< 0.0001
GP-UKF	-3.84 ± 0.175	< 0.0001	2.65 ± 0.0985	< 0.0001	1.32 ± 0.0266	< 0.0001
TIM	-0.779 ± 0.238	< 0.0001	4.52 ± 0.141	< 0.0001	1.78 ± 0.0323	< 0.0001
Kitagawa ($T=10$ and $R=200$)						
UKF-D	$10^{0} \times 3.78 \pm 0.662$	< 0.0001	$10^{0} \times 5.42 \pm 0.607$	< 0.0001	$10^{0} \times 1.32 \pm 0.0841$	< 0.0001
UKF-L*	$\mathbf{2.24 \pm 0.369}$	N/A	$\mathbf{3.60 \pm 0.477}$	N/A	$\mathbf{1.05 \pm 0.0692}$	N/A
EKF	617 ± 554	0.0149	9.69 ± 0.977	< 0.0001	1.75 ± 0.113	< 0.0001
CKF	> 1000	0.1083	5.21 ± 0.600	< 0.0001	1.30 ± 0.0830	< 0.0001
GP-ADF	2.93 ± 0.0143	0.0001	18.2 ± 0.332	< 0.0001	4.10 ± 0.0522	< 0.0001
GP-UKF	2.93 ± 0.0142	0.0001	18.1 ± 0.330	< 0.0001	4.09 ± 0.0521	< 0.0001
TIM	48.8 ± 2.25	< 0.0001	37.2 ± 1.73	< 0.0001	4.54 ± 0.179	< 0.0001
Pendulum ($T = 200 = 80$ s and $R=100$)						
UKF-D	$10^{0} \times 2.611 \pm 0.0750$	< 0.0001	$10^{-1} \times 5.03 \pm 0.0760$	< 0.0001	$10^{-1} \times 10.6 \pm 0.0940$	< 0.0001
UKF-L*	$\mathbf{0.392 \pm 0.0277}$	N/A	$\mathbf{1.93 \pm 0.0378}$	N/A	6.14 ± 0.0577	N/A
EKF	0.660 ± 0.0429	< 0.0001	1.98 ± 0.0429	0.0401	$\mathbf{6.11 \pm 0.0611}$	0.779
CKF	1.12 ± 0.0500	< 0.0001	3.01 ± 0.0550	< 0.0001	7.79 ± 0.0760	< 0.0001
GP-ADF	1.18 ± 0.00681	< 0.0001	4.34 ± 0.0449	< 0.0001	10.3 ± 0.0589	< 0.0001
GP-UKF	1.77 ± 0.0313	< 0.0001	5.67 ± 0.0714	< 0.0001	11.6 ± 0.0857	< 0.0001
TIM	0.896 ± 0.0115	< 0.0001	4.13 ± 0.0426	< 0.0001	10.2 ± 0.0589	< 0.0001

6.1. Sinusoidal dynamics

The models were trained on $T=1000$ observations from the sinusoidal dynamics, and tested on $R=10$ restarts with $T=500$ points each. The initial state was sampled from a standard normal $x_1 \sim \mathcal{N}(0,1)$. The UKF optimizer found the optimal values $\alpha=2.0216$, $\beta=0.2434$, and $\kappa=0.4871$.

6.2. Kitagawa

The Kitagawa model has a tendency to stabilize around $x=\pm 7$ where it is linear. The challenging portion for filtering is away from the stable portions where the dynamics are highly non-linear. Deisenroth et al. [15] evaluated the model using $R=200$ independent starts of the time series allowed to run only $T=1$ time steps, which we find somewhat unrealistic. Therefore, we allow for $T=10$ time steps with $R=200$ independent starts. In this example, $x_1 \sim \mathcal{N}(0, 0.5^2)$.

The learned value of the parameters were $\alpha=0.3846$, $\beta=1.2766$, $\kappa=2.5830$.

6.3. Pendulum

The models were tested on $R=100$ runs of length $T=200$ each, with $\mathbf{x}_1 \sim \mathcal{N}([-\pi\ 0],[0.1^2\ 0.2^2])$. The initial state mean of $[-\pi\ 0]$ corresponds to the pendulum being in the downward position. The models were trained on $R=5$ runs of length $T=200$. We found that in order to perform well on NLL, but not on MSE and MAE, multiple runs of the time series were needed during training; otherwise, TIM had the best NLL. This is because if the time series is initialized in one state the model will not have a chance to learn the needed parameter settings to avoid rare, but still present, sigma point collapse in other parts of the state-space. A short period single sigma point collapse in a long time series can give the models a worse NLL than even TIM due to incredibly small likelihoods. The MSE and MAE losses are more bounded so a short period of poor performance will be hidden by good performance periods. Even when $R=1$ during training, sigma point collapse is much rarer in UKF-L than UKF-D. The UKF found optimal values of the parameters to be $\alpha=0.5933$, $\beta=0.1630$, $\kappa=0.6391$. This is further evidence that the correct θ is hard to proscribe a priori and must be learned empirically. We compare the predictions of the default and learned settings in Fig. 4.

6.4. Analysis of sigma point collapse

We find that the marginal likelihood is extremely unstable in regions of θ that experience sigma point collapse. When sigma point collapse occurs, the predictive variances become far too small making the marginal likelihood much more susceptible to noise. Hence, the marginal likelihood is smooth near the optima, as seen in Fig. 2. As a diagnostic d for sigma point collapse we look at the mean $|\boldsymbol{\Sigma}|$ of the predictive distribution.

6.5. Computational complexity

The UKF-L, UKF, and EKF have test set computational time $\mathcal{O}(DT(D^2+M))$. The GP-UKF and GP-ADF have complexity $\mathcal{O}(DT(D^2+DMN^2))$, where N is the number of points used in training to learn f and g. This means that if $N^2 \gg D$ then UKF-L will have a much smaller computation complexity than the GP-ADF, which also attempts to avoid sigma point collapse. Typically it will take much more than N points to approximate a function in D dimensions, which implies that we will almost always have $N^2 \gg D$. The D^3 term in GP-ADF comes from the covariance calculations in the GP. This is usually not problematic as D is typically small, unlike T. If a large number of training points N is needed to approximate f and g well the GP-ADF and GP-UKF can become much slower than the UKF.

6.6. Discussion

The learned parameters of the UKF performed significantly better than the default UKF for all error measures and data sets. Likewise, it performed significantly better than all other methods except against the EKF on the pendulum data on MAE, where the two methods are essentially tied. We found that results could be improved further by averaging the predictions of the UKF-L and the EKF.

There exists another set of parameters rarely addressed in the UKF. The Cholesky factorization is typically used for $\sqrt{\boldsymbol{\Sigma}}$. However, any matrix square-root can be used. Meaning, we can apply a rotation matrix to the Cholesky factorization $\boldsymbol{\Sigma}$ and get a valid UKF, which gives us $\mathcal{O}(D^2)$ extra degrees of freedom (parameters). However, the beauty of the UKF-L is that the number of parameters to learn is three regardless of D.

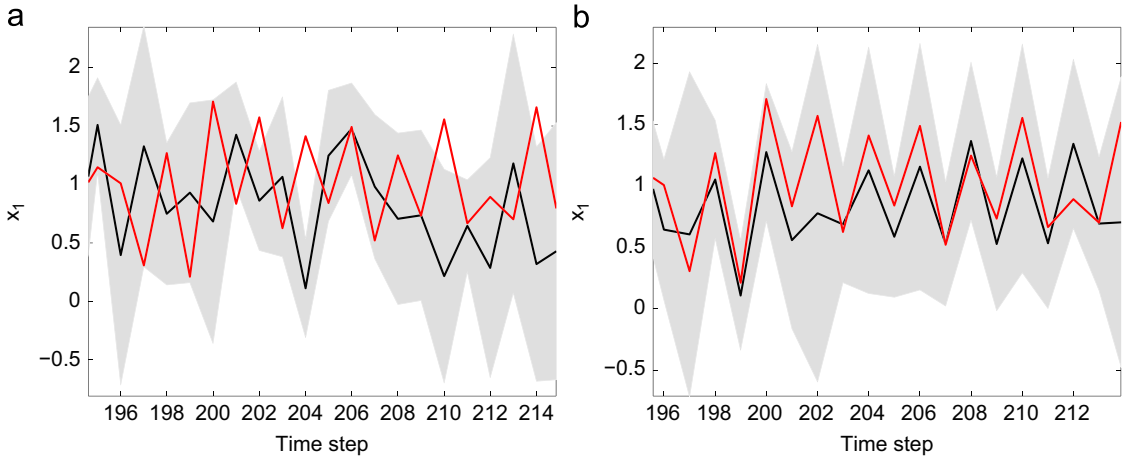

Fig. 4. Comparison of default and learned for one-step-ahead prediction for first element of $\mathbf{y}_t$ in the Pendulum model. The red line is the truth, while the black line and shaded area represent the mean and 95% confidence interval of the predictive distribution. (a) Default θ, (b) learned θ. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7. Conclusions

We have presented an automatic and model based mechanism to learn the parameters of a UKF, $\{\alpha,\beta,\kappa\}$, in a principled way. The UKF can be reinterpreted as a generative process that performs inference on a slightly different NLDS than desired through specification of f and g. We demonstrate how the UKF can fail arbitrarily badly in very nonlinear problems through sigma point collapse. Learning the parameters can make sigma point collapse less likely to occur. When the UKF learns the correct parameters from data it can outperform other filters designed to avoid sigma point collapse, such as the GP-ADF, on common benchmark dynamical system problems.

Acknowledgements

We thank Steven Bottone, Zoubin Ghahramani, and Andrew Wilson for advice and feedback on this work.

References

[1] R.E. Kalman, A new approach to linear filtering and prediction problems, Trans. ASME—J. Basic Eng. 82 (1960) 35–45.
[2] P.S. Maybeck, Stochastic Models, Estimation, and Control. Mathematics in Science and Engineering, vol. 141, Academic Press, Inc., 1979.
[3] S.J. Julier, J.K. Uhlmann, A new extension of the Kalman filter to nonlinear systems, in: Proceedings of AeroSense: 11th Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, FL, USA, 1997, pp. 182–193.
[4] I. Arasaratnam, S. Haykin, Cubature Kalman filters, IEEE Trans. Autom. Control 54 (6) (2009) 1254–1269.
[5] S.J. Julier, J.K. Uhlmann, Unscented filtering and nonlinear estimation, Proc. IEEE 92 (3) (2004) 401–422. doi:10.1109/JPROC.2003.823141.
[6] E.A. Wan, R. van der Merwe, The unscented Kalman filter for nonlinear estimation, in: Symposium 2000 on Adaptive Systems for Signal Processing, Communication and Control, IEEE, Lake Louise, AB, Canada, 2000, pp. 153–158.
[7] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, The MIT Press, Cambridge, MA, USA, 2005.
[8] S.J. Julier, J.K. Uhlrnann, H.F. Durrant-Whyte, A new approach for filtering nonlinear systems. in: American Control Conference, vol. 3, IEEE, Seattle, WA, USA, 1995, pp. 1628–1632. doi:10.1109/ACC.1995.529783.
[9] E. Snelson, Z. Ghahramani, Compact approximations to Bayesian predictive distributions, in: Proceedings of the 22nd International Conference on Machine Learning, Omnipress, Bonn, Germany, 2005, pp. 840–847.
[10] M.A. Osborne, R. Garnett, S.J. Roberts, Gaussian processes for global optimization, in: 3rd International Conference on Learning and Intelligent Optimization (LION3), Trento, Italy, 2009.
[11] N. Srinivas, A. Krause, S. Kakade, M. Seeger, Gaussian process optimization in the bandit setting: no regret and experimental design, in: J. Fürnkranz, T. Joachims (Eds.), Proceedings of the 27th International Conference on Machine Learning, Omnipress, Haifa, Israel, 2010, pp. 1015–1022.
[12] C.E. Rasmussen, Z. Ghahramani, Bayesian Monte Carlo, in: S. Becker, S. Thrun, K. Obermayer (Eds.), Advances in Neural Information Processing Systems, vol. 15, The MIT Press, Cambridge, MA, USA, 2003, pp. 489–496.
[13] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, The MIT Press, Cambridge, MA, USA, 1998.
[14] R. Turner, M.P. Deisenroth, C.E. Rasmussen, State-space inference and learning with Gaussian processes, in: the 13th International Conference on Artificial Intelligence and Statistics, vol. 9, Sardinia, Italy, 2010, pp. 868–875.
[15] M.P. Deisenroth, M.F. Huber, U.D. Hanebeck, Analytic moment-based Gaussian process filtering, in: Proceedings of the 26th International Conference on Machine Learning, Omnipress, Montreal, QC, Canada, 2009, pp. 225–232.
[16] G. Kitagawa, Monte Carlo filter and smoother for non-Gaussian nonlinear state space models, J. Computat. Graphical Stat. 5 (1) (1996) 1–25.
[17] M.P. Deisenroth, C.E. Rasmussen, J. Peters, Model-based reinforcement learning with continuous states and actions, in: Proceedings of the 16th European Symposium on Artificial Neural Networks (ESANN 2008), Bruges, Belgium, 2008, pp. 19–24.
[18] C.M. Bishop, Pattern Recognition and Machine Learning, first ed., Springer, 2007.

Ryan Turner received his Bachelors of Science degree from the University of California, Santa Cruz, CA, USA in Computer Engineering in 2007 (highest honors). He has recently completed his Ph.D. in Machine Learning at the University of Cambridge in the Computational and Biological Learning Lab. His research interests are Gaussian processes, state space models, and change point detection.

Carl Edward Rasmussen received his Masters degree in Engineering from the Technical University of Denmark and his Ph.D. degree in Computer Science from the University of Toronto in 1996.

Since 1996 he has been a post doc at the Technical University of Denmark, a senior research fellow at the Gatsby Computational Neuroscience Unit at University College London from 2000 to 2002 and a junior research group leader at the Max Planck Institute for Biological Cybernetics in Tübingen, Germany, from 2002 to 2007. He is a reader in the Computational and Biological Learning Lab at the Department of Engineering, University of Cambridge, Germany. His main research interests are Bayesian inference and machine learning.