Which resampling methods can tame ill-behaved gradients in chaotic systems?

Paavo Parmas
OIST
Okinawa, Japan
paavo.parmas@oist.jp

Jan Peters
TU Darmstadt
Darmstadt, Germany
mail@jan-peters.net

Kenji Doya
OIST
Okinawa, Japan
doya@oist.jp

Abstract

Learning by gradient descent with derivatives backpropagated through model predictions is an attractive concept; however, in our previous work we showed that when the dynamics are chaotic—a property of nonlinear systems often found in nature—then gradient estimation can become erratic turning the optimization into a random walk. We solved the problem with a new gradient estimation algorithm called total propagation [6], but some work leading up to the final solution was left out of the article. Here, we provide additional illustrations, and discuss an alternative solution of resampling the predictions at each time step from a distribution fitted onto the particles. We show that resampling from a Gaussian stabilizes the gradients, while resampling from a mixture of Gaussians fitted via EM does not. Moreover, we interpolate between fully resampled particles and the original trajectory distribution, and show a gradual transition to ill-behaved gradients. Finally we explain undesirable properties of resampling based methods.

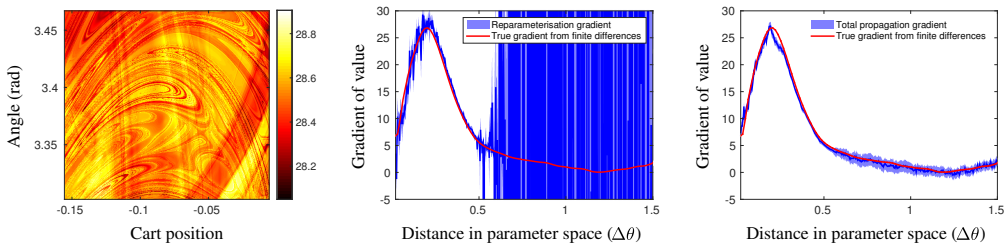
1 Introduction

Motivated by the desire to overcome the restrictions of the PILCO [1] algorithm, we attempted to swap out the approximate Gaussian distributions in PILCO with Monte Carlo particle-based predictions [6]. The key issue we encountered is depicted in Fig. 1b. In the cart-pole swing-up and balancing task, we chose a direction in the policy parameter space, and plotted the gradient of the objective function against the perturbation magnitude $\Delta\theta$. For some regions of the policy parameter space, the gradient behaves well, but in other regions the gradient variance explodes by over 10^6 times. This problem was caused a by a chaos-like nature of the dynamics illustrated by the fractal pattern in the long-term predictions in Fig. 1a. One method to stabilize the gradients is by resampling the predictions at each time step from a Gaussian distribution (see Sec. 3). Our more recent work showed that both the gradient stabilizing effect as well as an effect on smoothing out the reward contributed to the success of Gaussian resampling [5]. A remaining question is why resampling stabilizes the gradients, and whether resampling from multimodal distributions may also work. We first provide additional illustrations about the problem, then explain other resampling methods.

2 A closer look at the curse of chaos

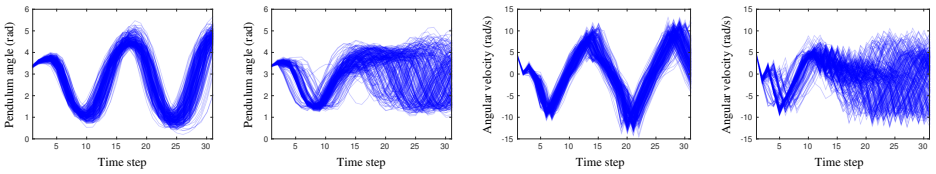
Previously, we had explained the curse of chaos in terms of the value landscape in Fig. 1a. In Fig. 2, we show what the predicted trajectory distributions look like for $\Delta\theta = 0$ (the well-behaved case) and $\Delta\theta = 1.5$ (the chaotic case). Due to the chaotic dynamics, there is a mixing of the trajectories, and the derivative of any individual trajectory does not provide information about the derivative of the whole distribution. Notice that if one compares individual trajectories between the two cases, the trajectories do not look that different, but the distributions have clearly different behavior.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.



(a) $\Delta\theta = 1.5$ (b) (c)

Figure 1



(a) $\Delta\theta = 0$ (b) $\Delta\theta = 1.5$ (c) $\Delta\theta = 0$ (d) $\Delta\theta = 1.5$

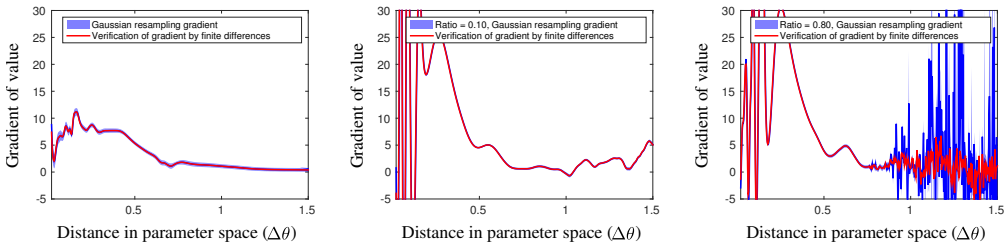
Figure 2: Bifurcations and a chaos-like mixing of trajectories leads to poor gradients computed using backpropagation at $\Delta\theta = 1.5$ in Fig. 1b.

3 Resampling from a unimodal Gaussian

Originally, McHutchon [3] had unsuccessfully attempted particle-based methods in PILCO, and suggested trying to replicate PILCO's Gaussian moment-matching effect by fitting a Gaussian on the particles and resampling to enforce unimodal trajectories. This was later tested in a Deep PILCO article [2], which found that resampling indeed improved the learning performance. We found that rather than the unimodality a more important effect of resampling is that the gradients are stabilized (compare Fig. 1b to Fig. 3a). This effect is most likely due to an averaging out of the value landscapes, i.e. it would smooth out the high-frequency components in Fig. 1a.

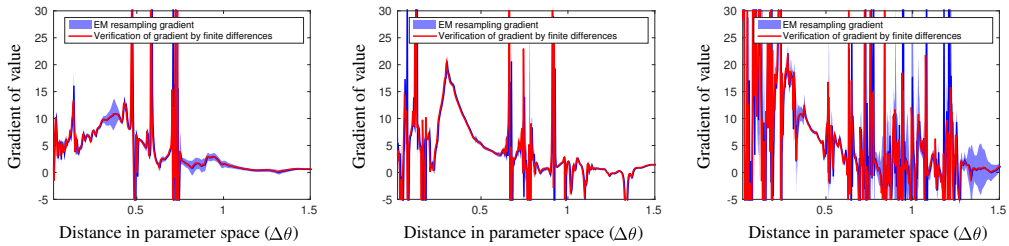
This method works by fitting a Gaussian on the particles at each time step, i.e. $\hat{\mu} = \sum_{i=1}^{P} \mathbf{x}_i / P$ and $\hat{\Sigma} = \sum_{i=1}^{P} (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T / (P-1)$. The particles are resampled from the fitted distribution $\mathbf{z}_i \sim \hat{\mu} + L\epsilon_i \mid \epsilon_i \sim \mathcal{N}(0, I)$, where L is the Cholesky factor of $\hat{\Sigma}$. See [4] for how to compute $\frac{\mathrm{d}L}{\mathrm{d}\hat{\Sigma}}$.

In this work we further considered a method to smoothly interpolate between fully resampling, and keeping the particles on their original trajectory. The method works by sampling from the fitted Gaussian, but only moving the original samples a portion of the distance towards the resampled particles, i.e. for each each particle $\mathbf{z}'_i = (1-r)\mathbf{z}_i + r\mathbf{x}_i$, where $r \in [0, 1]$ is a ratio. Fig. 3 shows that the gradients gradually transition from stable to unstable when one deviates from fully resampling.



(a) Full resampling (b) Ratio = 0.1 (c) Ratio = 0.8

Figure 3: The gradients gradually transition from unstable to stable when the particles are moved closer towards being completely resampled from a Gaussian distribution.



(a) $K = 2, M = 2$ (b) $K = 2, M = 5$ (c) $K = 5, M = 2$

Figure 4: Increasing the # of mixture components K or EM iterations M leads to worse gradients

4 Resampling from a mixture of Gaussians via backprop through EM

Sec. 3 explained that resampling can smooth out the gradients. An interesting question is then whether resampling not from a unimodal Gaussian, but from a mixture distribution could allow stabilizing the gradients, while still allowing for multimodal trajectory distributions. We propose to fit mixture distributions using the EM algorithm, then backpropagate through the algorithm to obtain the gradient. Algorithm 1 explains the method. The results are in Fig. 4. We can see that that the gradients were not stabilized. The result is not entirely conclusive, as the gradient behaves poorly in the full range of $\Delta\theta$, not just in the unstable region in Fig. 1b; however, the gradients become worse, as the method deviates further from a unimodal Gaussian, and it does not appear straight-forward to achieve good performance with a multimodal resampling method.

5 Why resampling based methods are undesirable

Even if resampling from a multimodal distribution were to work, we believe it is not ideal, because it destroys the temporal dependence in the particles, which is undesirable for the reasons listed below.

- It is difficult to model sampling dependencies, i.e. if the uncertainty is caused by a lack of knowledge about the dynamics model, then the sampling should be correlated, because sampling a prediction restricts the probability space of the underlying dynamics function.
- If the controller depends on past history, e.g. a recurrent neural network, resampling makes it more difficult to model such a dependency.
- Parallel computation becomes more difficult, as it is necessary to exchange information between the particles to make predictions.
- The trajectory distribution is modified, and does not correspond to the true trajectory, even if the model is perfect. This can lead to learning controllers with lower performance, or even worse, the task may even become impossible if the approximation is too conservative.

6 Conclusions

We have provided additional evidence towards explaining that chaotic properties of physical systems can make it difficult to optimize controllers by differentiating through the model predictions due to an exploding gradient variance. While we focused on control tasks, it is probable that the same issues would emerge if one tries to train a predictive model of a chaotic system, where the model includes hidden states which are propagated forwards along a trajectory to perform inference. Blending the trajectories by resampling from a unimodal Gaussian distribution stabilizes the gradients; however, in the experiments which we tested, deviating from a unimodal distribution causes the issues with gradient variance to re-emerge. While our work does not fully rule out the possibility that some other multimodal resampling based method may work, we explained several downsides to resampling methods, and believe that other methods, such as total propagation [6] are preferable.

Algorithm 1 Particle-based trajectory predictions while resampling from a mixture of Gaussians fitted by the Expectation-Maximization Algorithm (EM)

Input: policy π with parameters θ, episode length T, initial Gaussian state distribution $\mathrm{p}(\mathbf{x}_0)$, cost function $c(\mathbf{x})$, learned dynamics model $\hat{f}$, number of particles P, number of mixture components K, number of EM iterations M.
Initialize: Set initial mixture component weights equally $W_k = {}^1\!/_K$, set initial mixture distributions to the initial distribution $\mathrm{g}_k = \mathrm{p}(\mathbf{x}_0)$ for each k, sample ${}^P\!/_K$ initial particles separately from each mixture component $\{\mathbf{z}_{i,k,0}\}_{i=1}^{P} \sim \mathrm{g}_k(\mathbf{z}_0)$.
for $t = 0$ **to** $T - 1$ **do**
1. **Predict next timestep:**
for each particle i **do**
Compute controls: $\mathbf{u}_{i,t} = \pi(\mathbf{z}_{i,t}; \theta)$
Predict next state distribution: $\mathcal{N}(\mathbf{x}_{i,t+1}; \mathbf{m}_{i,t+1}, \mathbf{S}_{i,t+1}) = \hat{f}(\mathbf{z}_{i,t}, \mathbf{u}_{i,t})$
Set particle weight: $w_{i,t+1} = W_{k,t}$ ▷ Based on which g_k, $\mathbf{z}_{i,t}$ was sampled from
end for
2. **Compute new initialization:**
for each mixture component k **do**
$\mu_k = \frac{1}{{}^P\!/_K} \sum_{i=1}^{{}^P\!/_K} \mathbf{m}_{i,k,t+1}$ ▷ The k index for $\mathbf{m}_{t+1}$ means the particle $\mathbf{z}_t$ came g_k
$\Sigma_k = \frac{1}{{}^P\!/_K - 1} \sum_{i=1}^{{}^P\!/_K} (\mathbf{m}_{i,k,t+1} - \mu_k)(\mathbf{m}_{i,k,t+1} - \mu_k)^T + \frac{1}{{}^P\!/_K} \sum_{i=1}^{{}^P\!/_K} \mathbf{S}_{i,k,t+1}$
end for
3. **Run EM with weighted samples to fit the Gaussian components:**
$\{\mu_k, \Sigma_k, W_k\}_{k=1}^{K} = \mathrm{EM}\left(M, (\mathbf{m}_{t+1}, \mathbf{S}_{t+1}, \mathbf{w}_{t+1}), \{\mu_k, \Sigma_k, W_k\}_{k=1}^{K}\right)$ ▷ The update equations in EM are weighted versions of the equations in step 2. $\mathbf{S}$ is ignored when computing the responsibilities for the samples, but is used when updating the covariance of the mixture component.
4. **Resample particles and compute cost:**
for each mixture component k **do**
Sample ${}^P\!/_K$ new particles: $\mathbf{z}_{i,k,t+1} \sim \mathrm{g}_k(\mathbf{z}_{i,k,t+1}; \mu_k, \Sigma_k)$
end for
Average the cost: $c_{t+1} = \frac{1}{P} \sum_{i=1}^{P} c(\mathbf{z}_{i,t+1})$
end for
Gradient computation: $\frac{\mathrm{d}}{\mathrm{d}\theta} \left(\sum_{t=1}^{T} \mathbb{E}\left[c(\mathbf{z}_{t+1})\right] \right)$ is stochastically approximated from the particles. Each computation can be differentiated, and the full gradient can be obtained by backpropagation.

Acknowledgments

This work was supported by OIST Graduate School funding and by JSPS KAKENHI Grant Numbers JP16H06563 and JP16K21738.

References

[1] Deisenroth, M. P. and Rasmussen, C. E. (2011). PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, pages 465–472.

[2] Gal, Y., McAllister, R., and Rasmussen, C. (2016). Improving PILCO with bayesian neural network dynamics models. In *Workshop on Data-efficient Machine Learning, ICML*.

[3] McHutchon, A. (2014). *Modelling nonlinear dynamical systems with Gaussian Processes*. PhD thesis, University of Cambridge.

[4] Murray, I. (2016). Differentiation of the Cholesky decomposition. *arXiv preprint arXiv:1602.07527*.

[5] Parmas, P. (2018). Total stochastic gradient algorithms and applications in reinforcement learning. In *Advances in Neural Information Processing Systems*.

[6] Parmas, P., Rasmussen, C. E., Peters, J., and Doya, K. (2018). PIPPS: Flexible model-based policy search robust to the curse of chaos. In *International Conference on Machine Learning*.