Particle Filters for Non-Linear Filtering

Simon Maskell
University of Liverpool
Liverpool, UK
smaskell@liverpool.ac.uk

PROBLEMS

We often encounter problems where we need to process an incoming stream of data to make inferences about a time-evolving quantity of interest. Examples of such problems emerge in a diverse range of applications spanning tracking, GPS-free navigation, robotics, epidemiology, and finance.

To make statistical inferences in such contexts, we capitalise on models that capture our understanding of both the time-evolution of the quantities of interest and the relationship between these quantities and the observed data. Were these models to be linear and Gaussian, the uncertainty can be exactly characterised using a sequence of analytic calculations. Unfortunately, the models are rarely linear or Gaussian. One widespread strategy is to approximate the models as being (locally) linear and Gaussian and/or to decompose the problem into a set of sub-problems each of which involves linear and Gaussian models: the result are approaches typified by the extended and unscented Kalman filters, multi-hypothesis tracker, and the interacting multiple model. Given that it is the models that capture our understanding, approximating the models necessarily compromises our ability to use that understanding to inform the inferences that we make. Particle filters adopt a different approach whereby we explicitly approximate the result of the inference but fully capitalise on the model fidelity when we do so. Particle filters achieve this via Sequential Monte Carlo, i.e., at each time step they characterise the uncertainty using a set of (weighted) sampled values for the inferred quantities of interest.

When the models concerned are well approximated as linear and Gaussian, particle filters may offer some benefit relative to alternative techniques, but this benefit is often (sensibly) argued to not be warranted by the computational expense required to propagate the (potentially large number of) samples over time. Perhaps as a result, while particle filters were initially hoped to offer improved performance in contexts where existing filters were struggling (typified by bearing-only tracking), they arguably failed to make significant gains in these applications: it transpires that the problems were limiting performance in these contexts, not the filters.

However, there are many important problems where the models are not well approximated as linear or Gaussian. It is these contexts where particle filters have shone as a result of their ability to solve problems that other approaches simply cannot tackle. Examples are diverse and range from GPS-free navigation [1] to localising earthquakes using data extracted from social media [2].

PARALLELISM

A fundamental strategy when developing faster processors is to make the processors smaller. However, when processors switch state, they generate heat and it becomes increasingly challenging to dissipate this heat as the devices shrink in size. The result is that single processors have failed to deliver on Moore's law since approximately 2013. Since then, increases in processing power (typified by the GPUs being used for deep learning) have been achieved by maximising the number of processors on a chip. If algorithms are to exploit such hardware, parallelism is a necessity.

When a particle filter processes each datum, it propagates each particle and calculates each particle's weight. These operations are independent from one particle to another such that particle filters are often claimed to be readily parallelisable. However, there is an issue with this claim. As a particle filter iterates through time, a weight is recursively updated for each particle: this weight is the extent to which the particle will contribute to any inference. It is inevitable that the weights for different particles will come to differ significantly and inferences will become dominated by a small subset of the particles. To address this wastefulness, particle filters employ a 'resampling' step. This involves removing the particles with low weights and replacing them with replications of the particles with high weights. It is the introduction of resampling that gave rise to the first working particle filter [3]. However, this same resampling step is non-trivial to parallelise. This has motivated research into approaches to both modifying the resampling step to make it amenable to parallel implementation [4] and approaches to defining a parallel implementation without such modifications [5].

PROPOSAL DISTRIBUTIONS

However much we exploit parallelism, we will be limited by how efficiently each particle is processed and, more specifically, the "proposal distribution", how the state associated with each particle is proposed. Research has focused on how to design efficient proposals. One exemplar such approach is the use of Kalman filter techniques inside the proposal [6]. Another closely related approach, particle flow [7], involves defining a process that takes the place of the proposal, and is highly reminiscent of numerical approaches (e.g., Hamiltonian Monte Car-

lo) that have proven both effective and popular in the context of other numerical Bayesian algorithms.

It transpires that one can achieve further improvements by proposing refinements to historic samples retrospectively in the light of recent data. Note that such an a fixed-lag approach [8] is also known as "blocking" in the context of particle filters.

PARAMETER ESTIMATION

As already explained, particle filters capitalise on models to make their inferences. Developing models and fine-tuning their parameters is time consuming. Techniques have been developed to learn such parameters from data: see, for example, [9].

As well as offering the potential to use high fidelity models, this capacity to learn models' parameters from data also makes it possible to apply particle filtering with the models used by deep learning algorithms such as long short-term memory (LSTM) networks and transformers. In this context, particle filters can be seen as a machine learning approach that enables users to understand the uncertainty associated with such deep learning approaches.

PROSPECTS

It is currently challenging for an applied researcher to capitalise on the advances to particle filtering that have happened in the last 25 years and are exemplified above. This contrasts with neighbouring domains where probabilistic programming languages (PPLS) such as Carptenter et al. [10] have made it straightforward to define and then use probabilistic models to make inferences from (fixed) data. One exciting avenue for future research into particle filters is to extend such PPLS to make it straightforward to perform parameter estimation and use parallel implementations of particle filters with high-performance proposal distributions.

REFERENCES

1. Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R., et al. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, Vol. 50, 2 (2002), 425–437.
2. Sakaki, T., Okazaki, M., and Matsuo, Y. Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web*, Raleigh, NC, 2010, 851–860.
3. Gordon, N. J., Salmond, D. J., and Smith, A. F. M. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEEE Proceedings F (Radar and Signal Processing)*, Vol. 140, 2 (1993), 107–113.
4. Bolic, M., Djuric, P. M., and Hong, S. Resampling algorithms and architectures for distributed particle filters. *IEEE Transactions on Signal Processing*, Vol. 53, 7 (2005), 2442–2450.
5. Varsi, A., Maskell, S., and Spirakis, P. G. An O(log2n) fully-balanced resampling algorithm for particle filters on distributed memory architectures. *Algorithms*, Vol. 14, 12 (2021), 342.
6. Van Der Merwe, R., Doucet, A., De Freitas, N., and Wan, E. The unscented particle filter. *Advances in Neural Information Processing Systems*, Vol. 13, 2000.
7. Daum, F., and Huang, J. Particle flow for non-linear filters with log-homotopy. *SPIE Signal and Data Processing of Small Targets 2008*, Vol. 6969, (2008), 414–425.
8. Doucet, A., and Senecal, S. Fixed-lag sequential monte carlo. In *Proceedings 2004 12th IEEE European Signal Processing Conference*, Vienna, Austria, 2004, 861–864.
9. Chopin, N., Jacob, P. E., and Papaspiliopoulos, O. Smc2: an efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Vol. 75, 3 (2013), 397–426.
10. Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., et al. Stan: A probabilistic programming language. *Journal of Statistical Software*, Vol. 76, 1 (2017), 1–32.