Using PGAS for factorial finite state machines

Lennart Svensson

Weight calculations in SMC and PGAS

Consider a Markov chain $\mathbf{z}_1, \mathbf{z}_2, \dots$ and a measurement sequence $\mathbf{y}_1, \mathbf{y}_2, \dots$ such that

$$p(\mathbf{y}_{1:T}|\mathbf{z}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{z}_t, \mathbf{z}_{t-1}, \dots, \mathbf{z}_{t-L+1}), \tag{1}$$

where $\mathbf{y}_{1:T} = [\mathbf{y}_1, \dots, \mathbf{y}_T]$, $\mathbf{z}_{1:T} = [\mathbf{z}_1, \dots, \mathbf{z}_T]$ and $\mathbf{z}_{t-L+1}$ is an empty set (that should be ignored) when $t - L + 1 < 1$. The primary purpose with this note is to derive the expression for the weights in equations (2) and (3) in [1].

The first weight is needed for both sequential Monte Carlo (SMC) and particle Gibbs with ancestor sampling (PGAS) algorithms:

$$W_{\theta,t}(\mathbf{z}_{1:t}) = \frac{\gamma_{\theta,t}(\mathbf{z}_{1:t})}{\gamma_{\theta,t-1}(\mathbf{z}_{1:t-1}) r_{\theta,t}(\mathbf{z}_t|\mathbf{z}_{1:t-1})}, \tag{2}$$

where $\gamma_{\theta,t}(\mathbf{z}_{1:t}) = p(\mathbf{z}_{1:t}|\mathbf{y}_{1:t})$. From here on, we omit the parameter θ from the notation. For simplicity we also assume that $r_{\theta,t}(\mathbf{z}_t|\mathbf{z}_{1:t-1}) = p(\mathbf{z}_t|\mathbf{z}_{t-1})$, i.e., that particles are propagated across time using the transition model in a simple bootstrap particle filter manner. Based on these assumptions we can simplify the expression for the weights in [1, Eq.(2)] to

$$\begin{aligned} W_{\theta,t}(\mathbf{z}_{1:t}) &= \frac{p(\mathbf{z}_{1:t}|\mathbf{y}_{1:t})}{p(\mathbf{z}_{1:t-1}|\mathbf{y}_{1:t-1}) p(\mathbf{z}_t|\mathbf{z}_{t-1})} \\ &\propto \frac{p(\mathbf{y}_{1:t}|\mathbf{z}_{1:t}) p(\mathbf{z}_{1:t})}{p(\mathbf{y}_{1:t-1}|\mathbf{z}_{1:t-1}) p(\mathbf{z}_{1:t-1}) p(\mathbf{z}_t|\mathbf{z}_{t-1})} \\ &= p(\mathbf{y}_t|\mathbf{z}_{t-L+1:t}). \end{aligned} \tag{3}$$

That is, to find these weights it is sufficient to evaluate the likelihood at time t.

The weights used to draw a random ancestor to particle N are

$$\tilde{w}^i_{t-1|T} = w^i_{t-1} \frac{p(\mathbf{z}^i_{1:t-1}, \mathbf{z}'_{t:T}|\mathbf{y}_{1:T})}{p(\mathbf{z}^i_{1:t-1}|\mathbf{y}_{1:t-1})}, \tag{4}$$

see [1, Eq.(3)]. We can simplify these weights using a similar strategy to what we did above:

$$\begin{aligned}\tilde{w}^i_{t-1|T} &\propto w^i_{t-1}\frac{p(\mathbf{y}_{1:T}\big|\mathbf{z}^i_{1:t-1},\mathbf{z}'_{t:T})p(\mathbf{z}^i_{1:t-1},\mathbf{z}'_{t:T})}{p(\mathbf{y}_{1:t-1}\big|\mathbf{z}^i_{1:t-1})p(\mathbf{z}^i_{1:t-1})}\\ &\propto w^i_{t-1}p(\mathbf{z}'_t\big|\mathbf{z}^i_{t-1})\prod_{\tau=t}^{t+L-2}p(\mathbf{y}_\tau\big|\mathbf{z}^i_{1:t-1},\mathbf{z}'_{t:T}).\end{aligned}\tag{5}$$

To obtain these equations we have made use of the fact that the state sequence is a Markov chain and the properties of the measurement model. We have also ignored factors that do not depend on the variable i.

It may be interesting to note that (5) takes a significantly simpler form for state space models (SSMs); for SSMs we have $L = 1$ which implies that the last factor does not depend on i. The corresponding weights for SSMs are therefore simply $\tilde{w}^i_{t-1|T} \propto w^i_{t-1}p(\mathbf{z}'_t\big|\mathbf{z}^i_{t-1})$. We also note that the transition probabilities, $p(\mathbf{z}'_t\big|\mathbf{z}^i_{t-1})$, factorise over the symbols in the state vectors.

Improved sampling

It is clear that it will not be sufficient to sample from the transition density in all scenarios. A particularly challenging setting is when both N_t and Q are large since the number of possible states is $(Q+1)^{N_t}$ at each time instant. If we sample from these states without using our observations we are likely to obtain very low update rates for difficult problems. If the update rate is too low it will simply not be feasible to use a Gibbs sampler for these problems since it is no longer mixing.

Brief ideas/reflections:

- We would like to make use of data in order to design a better proposal distribution for the particles. However, if N_t and Q are large it will also be very costly to evaluate the likelihood of the state for all values that it may take.

- One possibility is to assume that the particle that we condition is "good" and try to generate particles that resemble that particle. Exactly what we mean by "resemble" is not clear but it should somehow be close in space.

- It may be possible to use the output from an EP algorithm to obtain a better proposal. At least we may avoid proposing symbols in completely unreasonable areas.

- It generally appears difficult to take all dependencies across time and in between symbols in to account, but perhaps we can think of something clever. Whatever we propose does not have to be perfect as long as it yields a proposal density with a reasonable update rate.

References

[1] F. Lindsten, M. Jordan and T. Schön, Particle Gibbs with ancestor sampling, *Journal of Machine Learning Research*, (15):2145–2184, 2014.