Scalable backpropagation for Gaussian Processes using celerite

Daniel Foreman-Mackey

1Center for Computational Astrophysics, Flatiron Institute, New York, NY

Keywords: methods: data analysis — methods: statistical

INTRODUCTION

This research note presents a derivation and implementation of efficient and scalable gradient computations using the celerite algorithm for Gaussian Process (GP) modeling. The algorithms are derived in a “reverse accumulation” or “backpropagation” framework and they can be easily integrated into existing automatic differentiation frameworks to provide a scalable method for evaluating the gradients of the GP likelihood with respect to all input parameters. The algorithm derived in this note uses less memory and is more efficient than versions using automatic differentiation and the computational cost scales linearly with the number of data points.

GPs (Rasmussen & Williams 2006) are a class of models used extensively in the astrophysical literature to model stochastic processes. The applications are broad-ranging and some examples include the time domain variability of astronomical sources (Brewer & Stello 2009; Kelly et al. 2014; Haywood et al. 2014; Rajpaul et al. 2015; Foreman-Mackey et al. 2017; Angus et al. 2018), data-driven models of light curves or stellar spectra (Wang et al. 2012; Luger et al. 2016; Czekala et al. 2017), and the cosmic microwave background (Bond & Efstathiou 1987; Wandelt & Hansen 2003). In all of these applications, the calculation and optimization of the GP marginalized likelihood function (here we follow the notation of Foreman-Mackey et al. 2017)

\[
\log L(\theta, \alpha) = -\frac{1}{2} \left[ y - \mu_\theta \right]^T K_\alpha^{-1} \left[ y - \mu_\theta \right] - \frac{1}{2} \log \det K_\alpha + \text{constant}
\]  

is generally the computational bottleneck. The details of these models are omitted here (see Rasmussen & Williams 2006 and Foreman-Mackey et al. 2017 for details), but the key point is that, for a dataset with \(N\) data points, every evaluation of a GP model requires computing the log-determinant and multiplying a vector by the inverse of the \(N \times N\) covariance matrix \(K_\alpha\). The computational cost of these operations scales as \(O(N^3)\) in the general case, but the celerite method was recently introduced in the astronomical literature to compute the GP likelihood for a class of one-dimensional models with \(O(N)\) scaling (Ambikasaran 2015; Foreman-Mackey et al. 2017).

The details of the celerite method can be found in Foreman-Mackey et al. (2017) and I will not repeat them here. The only difference in notation is that all the matrices in what follows are the “pre-conditioned” matrices that are indicated with a tilde by Foreman-Mackey et al. (2017). The tilde is not included here for simplicity and to improve the readability of the algorithms. I will also use the symbol \(P\) for the \((N-1) \times J\) pre-conditioning matrix that was called \(\phi\) by Foreman-Mackey et al. (2017). The Cholesky factorization algorithm derived by Foreman-Mackey et al. (2017, their Equation 46) is as follows:

```matlab
function celerite_factor(U, P, d, W)
    # At input d = \(a\) and W = V
    S ← zeros(J, J)
    w_1 ← w_1/d_1
    for n = 2, \ldots, N:
        S ← diag(p_{n-1}) [S + d_{n-1} w_{n-1}^T w_{n-1}] diag(p_{n-1})
        d_n ← d_n - u_n S u_n^T
        w_n ← [w_n - u_n S] / d_n
    return d, W, S
```

In this algorithm, the `zeros(J, K)` function creates a \(J \times K\) matrix of zeros, the `diag` function creates a diagonal matrix from a vector, and \(x_n\) indicates a row vector made from the \(n\)-th row of the matrix \(X\). The computational cost
of this algorithm scales as $O(NJ^2)$. Using this factorization, the log-determinant of $K$ is

$$\log \det K = \sum_{n=1}^{N} \log d_n .$$  (2)

Similarly, Foreman-Mackey et al. (2017) derived a $O(NJ)$ algorithm to apply the inverse of $K$ (i.e. compute $Z = K^{-1} Y$) as follows (Equations 47 and 48 in Foreman-Mackey et al. 2017):

```python
function celerite_solve(U, P, d, W, Z)
    # At input $Z = Y$
    F ← zeros(J, Nrhs)
    for $n = 2, \ldots, N$:
        $F ← \text{diag}(p_{n-1}) \left[ F + w_{n-1}^T z_{n-1} \right]$
        $z_n ← z_n - u_{n-1} F$
    for $n = 1, \ldots, N$:
        $z_n ← z_n / d_n$
    G ← zeros(J, Nrhs)
    for $n = N - 1, \ldots, 1$:
        $G ← \text{diag}(p_n) \left[ G + u_{n+1}^T z_{n+1} \right]$
        $z_n ← z_n - w_n G$
    return $Z, F, G$
```

The empirical scaling of these algorithms is shown in Figure 1.

**GRADIENTS OF GP MODELS USING CELERITE**

It is standard practice to make inferences using Equation (1) by optimizing or choosing a prior and sampling with respect to $\theta$ and $\alpha$. Many numerical inference methods (like non-linear optimization or Hamiltonian Monte Carlo) can benefit from efficient calculation of the gradient of Equation (1) with respect to the parameters. The standard method of computing these gradients uses the identity (Rasmussen & Williams 2006)

$$\frac{d \log L}{d \alpha_k} = \frac{1}{2} \text{Tr} \left[ \tilde{r} \tilde{r}^T - K_\alpha^{-1} \right] \frac{dK_\alpha}{d\alpha_k}$$  (3)

where

$$\tilde{r} = K_\alpha^{-1} [y - \mu_\theta] .$$  (4)

Similar equations exist for the parameters $\theta$. Even with a scalable method of applying $K_\alpha^{-1}$, the computational cost of Equation (3) scales as $O(N^2)$. This scaling is prohibitive when applying the celerite method to large datasets and I have not found a simple analytic method of improving this scaling for semi-separable matrices. However, it was recently demonstrated that substantial computational gains can be made by directly differentiating Cholesky factorization algorithms even in the general case (Murray 2016).

Following this reasoning and using the notation from an excellent review of matrix gradients (Giles 2008), I present the reverse-mode gradients of the celerite method. While not yet popular within astrophysics, “reverse accumulation” of gradients (also known as “backpropagation”) has recently revolutionized the field of machine learning (see LeCun et al. 2015, for example) by enabling the non-linear optimization of models with large numbers of parameters. The review (Giles 2008) provides a thorough overview of these methods and the interested reader is directed to that discussion for details and for an explanation of the notation.

Using the notation from Giles (2008), after some tedious algebra, the reverse accumulation function corresponding to celerite_factor is found to be:
function celerite_factor_rev(U, P, d, W, S, Š, ā, Ģ)
    # At input ā = d and Ŵ = W
    Ū ← zeros(N, J)
    Ź ← zeros(N - 1, J)
    ĕN ← ĕN/dN
    for n = N, ..., 2:
        ān ← ān - wn ĕnT
        ūn ← [vn + 2 ān un] S
        Š ← Š - unT [vn + ān un]
        źp−1 ← diag(Š Š diag(pn−1)−1 + diag(pn−1)−1 Š Š)
        Š ← diag(pn−1) Š diag(pn−1)
        Źd−1 ← Žd−1 + wn−1 Š wTn−1
        ťn−1 ← ťn−1/dn−1 + wn−1 [Š ŠT]
        Š ← diag(pn−1)−1 Š diag(pn−1)−1 - Žd−1 wTn−1 wn−1
        ā1 ← ā1 - ť1 wT1
    return Ū, Ź, ā, Ģ

Similarly, the reverse accumulation function for to celerite_solve is:

function celerite_solve_rev(U, P, d, W, Z, F, G, Ģ, Ź)
    # At input Ź = Ź
    Ū ← zeros(N, J)
    Ź ← zeros(N - 1, J)
    ā ← zeros(N)
    Ŵ ← zeros(N, J)
    for n = 1, ..., N - 1:
        Ŵn ← Ŵn G T
        Ŭ ← Ū - wn Ŵn T Ŧn
        Ŷn ← Ŷn + wn G
        ŶG ← diag(pn)−1 ŶG
        Ŵp ← diag(G G T)
        ŪG ← diag(pn) ŪG
        ŶG ← ŶG - Ŷn+1 Ŷn+1 T Ŷn+1
        Ŭn+1 ← Ŷn+1 G T
        ĥy ← Ŧn+1 Ŷn+1 G
    for n = 1, ..., N:
        Ťn ← Ťn/dn
        Ŭd ← Ŷn Ťn T
    for n = N, ..., 2:
        Ūn ← Ūn - Ťn F T
        Ģ ← Ģ - Ťn Ťn T Ťn
        ĢF ← diag(pn−1)−1 ĢF
        Ťp−1 ← Ťp−1 + diag(Ģ F T)
        ĢF ← diag(pn−1) ĢF
        ĢF ← ĢF - Ťn−1 Ťn−1 Ťn−1 T Ťn−1
        Ťn−1 ← Ťn−1 + Ťn−1 ĢF T
        Ģn−1 ← Ģn−1 + wn−1 ĢF
    return Ū, Ź, ā, Ģ

A reference C++ implementation of this algorithm can be found online (Foreman-Mackey 2018) and Figure 1 shows the performance of this implementation.
DISCUSSION

This research note presents the algorithms needed to efficiently compute gradients of GP models applied to large datasets using the *celerite* method. These developments increase the performance of inference methods based on *celerite* and improve the convergence properties of non-linear optimization routines. Furthermore, the derivation of reverse accumulation algorithms for *celerite* allow its integration into popular model building and automatic differentiation libraries like Stan (Carpenter et al. 2015), TensorFlow (Abadi et al. 2016), and others.

It is a pleasure to thank Eric Agol, Sivaram Ambikasaran, and Victor Minden for conversations that inspired this work. A reference implementation of these algorithms, benchmarks, and tests can be found at https://github.com/dfm/celerite-grad and Foreman-Mackey (2018).

Figure 1. The empirical computational scaling of the algorithms presented in this note. (top row): The cost of computing Equation (1) as a function of the number of data points ($N$; left) and the model complexity ($J$; right). (bottom row): The cost of computing the gradient of Equation (1) with respect to the vector $a$ and the matrices $U$, $V$, and $P$. In the left panels, each line corresponds to a different value of $J$ as indicated in the legend (with $J$ increasing from bottom to top). Similarly, in the right panels, the lines correspond to different values of $N$ increasing from bottom to top. In both cases, the theoretical scaling is $O(N, J^2)$. 
Abadi, M., et al. 2016, in OSDI, Vol. 16, 265–283

Ambikasaran, S. 2015, Numer. Linear Algebra Appl., 22, 1102

Angus, R., et al. 2018, MNRAS, 474, 2094

Bond, J. R., & Efstathiou, G. 1987, MNRAS, 226, 655

Brewer, B. J., & Stello, D. 2009, MNRAS, 395, 2226

Carpenter, B., et al. 2015, ArXiv, 1509.07164

Czekala, I., et al. 2017, ApJ, 840, 49

Foreman-Mackey, D. 2018, celerite-grad, ,

doi:10.5281/zenodo.1160614

Foreman-Mackey, D., et al. 2017, AJ, 154, 220

Giles, M. 2008, An extended collection of matrix derivative results for forward and reverse mode automatic differentiation, Tech. Rep. NA-08/01, Oxford University Computing Laboratory

Haywood, R. D., et al. 2014, MNRAS, 443, 2517

Kelly, B. C., et al. 2014, ApJ, 788, 33

LeCun, Y., et al. 2015, Nature, 521, 436

Luger, R., et al. 2016, AJ, 152, 100

Murray, I. 2016, ArXiv, 1602.07527

Rajpaul, V., et al. 2015, MNRAS, 452, 2269

Rasmussen, C. E., & Williams, K. I. 2006, Gaussian Processes for Machine Learning (MIT Press)

Wandelt, B. D., & Hansen, F. K. 2003, PhRvD, 67, 023001

Wang, Y., et al. 2012, ApJ, 756, 67