Lab 2: Star spots, stellar rotation, and the age distribution of the Galactic disk

Astro 128 / 256 (UC Berkeley)

Final lab report due 4:00 pm, Tuesday March 17, 2020

Introduction

In this lab, you will use Gaussian processes to model the light curves of stars observed by the NASA *Kepler* mission. The first goal is to infer rotation periods of stars from their light curves. This can be nontrivial, because (a) light curves often exhibit several different kinds of variability, only some of which are attributable to rotation, and (b) it may not be obvious what the best kind of predictive model for rotation-driven variability is. Once you have a serviceable model for Kepler light curves and can use it to infer periods, your will apply it to a large sample of stars. Finally, you will compare the distribution of observed periods to theoretical age–rotation rate relations, and in doing so, learn something about the age distribution of stars in the Milky Way.

Technical Components

- Gaussian processes: theory and implementation
- Efficient Gaussian process packages in python
- Model selection; cross validation
- Hyperparameter tuning, MCMC
- Algorithmic complexity, "Big O" notation
- Managing memory usage
- Stellar ages, gyrochronology

Problem 0

Read the following papers:

1. Borucki et al. 2010: description of the Kepler mission.
2. McQuillan et al. 2013 and 2014: measuring stellar rotation periods using the autocorrelation function of Kepler light curves.
3. Angus et al. 2018: inferring stellar rotation periods with Gaussian processes.
4. Foreman-Mackey et al. 2017: fast Gaussian process algorithms.

These papers are for your own edification, so spend as much or as little time as you need on them. Based on our experience, time spent reading them is well spent. We also recommend this 50-minute video for a general primer on GPs.

Problem 1

Before applying them to real data, let's start with a hands-on introduction to Gaussian processes.

(a) Simulate a fake dataset with $N = 20$ datapoints. Assume that your true signal is given by a model $y = \sin(t) + \cos(2t)$. Choose N random observation times, distributed uniformly over $0 < t < N/2$. Assign each point a random uncertainty σ_y, drawn from a uniform distribution $0 < \sigma_y < 0.5$. Throughout this lab, use a fixed random seed so that your results are reproducible. Make a plot of your synthetic data and the true model over $0 < t < N/2$.

(b) Build a naive Gaussian process model from scratch, and use it to predict the values of $y(t)$ over $0 < t < 10$ (on a finely sampled grid) based on the 10 datapoints. Use a squared-exponential kernel, $k(x_i, x_j) \propto \exp\left[(x_i - x_j)^2 / (2\ell^2)\right]$, where x_i and x_j represent different datapoints, and ℓ is a free parameter. Start with $\ell = 1$. Your model should be built on simple python and numpy functions – matrix inversion is OK, but no fancy GP packages are allowed. Plot the predictions of your Gaussian process model, including the $\pm 1\,\sigma$ uncertainties, on top of the data and true model. Comment on the model's performance. Does the true model prediction generally fall within the uncertainties?

(c) Now try varying the correlation length ℓ in your kernel function. Try $\ell = 0.2$, $\ell = 1$, and $\ell = 5$, and make a plot comparing the data, GP model, and true model for both cases. Comment on your results. Does this give you a sense of what the "best" value of ℓ is?

(d) To be more quantitative, we'll use cross-validation to choose the value of ℓ. Generate another 10 datapoints to use as a validation set. Now try a log-spaced grid of 50 ℓ values between 0.1 and 10. For each value of ℓ, calculate the cross-validation loss, which for this problem we'll define as χ^2/N for the points in the cross validation set. That is, $\chi^2/N = \frac{1}{N}\sum\left(\frac{(y_{\rm GP} - y_{\rm obs})^2}{(\sigma^2_{\rm obs} + \sigma^2_{\rm GP})}\right)$, where the sum is over points in the cross-validation set, $y_{\rm GP}$ is the value predicted by the GP model, $\sigma_{\rm GP}$ is the corresponding error, $\sigma_{\rm obs}$ is the measurement error, and $y_{\rm obs}$ is the measured value. Plot this loss as a function of ℓ. Comment on this behavior. What does this suggest is the "best" value of ℓ?

(e) Now check out the python GP package George. Discuss the possible advantages of using George over own GP implementation. Use George to implement a GP model with an exponential-squared kernel and $\ell = 1$. Verify that it gives *identical* model predictions and uncertainties to your explicit model in part (b). Now repeat this, but using the "HODLRSolver" in George.

(f) What happens when you increase the number of datapoints? Try log-spaced values of N between 2^0 and 2^{13}. Plot the total computational time required to evaluate the GP model as a function of N. Compare three models: your naive GP, the default GP implementation in George, and George with the HODLRSolver. Comment on the scalings with each model. Give a one-paragraph explanation what George is doing that allows it to scale better than your naive GP implementation. See this paper for more information.

Problem 2

Now we'll move on to modeling real data.

(a) Download the catalog of stars with periodic light curves attributed to rotation by McQuillan et al. 2014 (table 1). It can be found here.

(b) For the first 100 objects in the table, download their Kepler light curves (for all quarters). You can use the kplr package.[1] Use the "PDCSAP_FLUX" and its error. Explain briefly what this is and how it is computed. How is it different from the raw flux? Rescale the fluxes and errors so that they have 0 mean

[1] This package can be loaded on Datahub or colab if you add the following to the beginning of your notebook: `!pip install fitsio==1.0.5; pip install kplr`

and order unity variance. This will make the range of optimal kernel hyperparameters more constant across all the light curves. Plot flux as a function of time for several objects. Rather than plotting the full 4 years' of data, set the time limits to be comparable to the characteristic evolution timescale, so you can see a few periods. Show plots for the following KIC ids: 892376, 1026146, 1026474.

(c) Do all light curves have clear periods? Plot an example of one light curve in which the period is visually obvious, and one in which it is not (perhaps because the shape changes on a timescale comparable to the period itself). Discuss what, astrophysically, causes the changes in light curve shape over time.

Required checkpoint for Monday, March 9: By 4:00 pm, submit a pdf to bCourses containing the plots you made in 1.c, 1.f, and 2.c.

(d) We'll model the light curves using a quasi-period kernel, similar to a damped simple harmonic oscillator. Explain the physical motivation for using this kind of kernel. The introduction of Angus et al. 2018 should be very relevant.

(e) A common problem when dealing with periodic signals is *aliasing*, which can cause false periods at harmonics of the true period (i.e., $2P$, $P/2$, etc.). Calculate and plot Lomb-Scargle periodograms for a few stars. Do you see evidence of aliasing?

(f) To minimize aliasing, we'll use a kernel that is a sum of simple harmonic oscillators with periods P and $2P$. A fast version of such a kernel is implemented in the exoplanet package. A tutorial that shows how to implement a GP model in `exoplanet` can be found here.

(g) Implement a GP model for the full light curve of one of your objects using `exoplanet` and a sum of two simple harmonic oscillators as a kernel. This kernel has several free parameters. One is the period, P, which is one we care most about. Once you have a model, you can assess how "good" the model is using a traditional goodness of fit indicator, such as χ^2. You can then adjust the free parameters of the kernel to maximize the likelihood (minimizing χ^2, assuming Gaussian uncertainties). Carry out such a fit for one of your light curves, either using one of the optimizers in `scipy` or the built-in optimizer in `exoplanet`. How does your best-fit P compare to the estimate from McQuillan et al. 2014?

(h) Assuming Gaussian uncertainties is not formally the right thing to do for light curves. Explain (i) why this is, and (ii) why it's still an acceptable approximation for Kepler data, and (iii) what a more principled choice might be.

(i) You likely found that evaluating and optimizing the GP model with full Kepler light curves (which have about 60,000 points) is fairly slow. The times-spacing of Kepler light curves is pretty dense. You can likely get reasonably good measurements of periods with only a subset of these points, since the periods of most rotating stars will be significantly longer than the typical exposure time. Try reducing the number of points by (a) downsampling the data (i.e., taking only every N-th datapoint) and/or (b) taking a shorter subset of the light curve instead of the full 4 years. How much sub-sampling can you get away with before you period measurements get degraded? Justify your choice by plotting some quantification of period accuracy versus how much you subsample. You can use whatever subsampling you find to be appropriate for the rest of the lab.

(j) To get uncertainties on P (and other kernel parameters), we can use MCMC. Using pymc3 and the same likelihood function as in part (g), with the addition of sensible priors, sample from the posterior of kernel parameters. Make some diagnostic plots to verify that your sampler has converged. Use `corner.py` to visualize your parameter constraints. Note: due to the large plausible dynamic range, sampling (and optimization) will likely be more efficient if you sample in the log of the kernel parameters.

(k) Once you have a well-fit GP model of a light curve, you can use it to predict the light curve where there is no data. Kepler light curves all have gaps, due to periods when the telescope was downlinking data, and other technical glitches. Let's examine one of these gaps and plot the model prediction and uncertainty. Consider the light curve of KIC 892376 over $700 < t < 770$. First, plot the prediction and 1σ uncertainties for the maximum-likehoood model found in part (g). Then repeat this on a

separate plot, now plotting predictions for models with different kernel hyperparameters drawn from the samples obtained in part (j). Comment. Do these two approaches yield comparable uncertainties in the prediction? How can you interpret the difference between the maximum-likelihood and MCMC uncertainties?

(l) Now take the first 100 stars in table 1 and fit a GP model to estimate the rotation period, P. Use whatever sub-sampling scheme you found to work in part (i). Just using an optimizer is OK here; no need for full MCMC. Make a plot comparing your best-fit periods to those reported by McQuillan et al. 2014, on a log scale. Comment on the origin of any discrepancies.

(m) Once you are convinced your period-finding scheme is robust (which hopefully will mean you get the same period as McQuillan et al. in a majority of cases), try fitting the light curves of all stars in a narrow temperature range, $3600 < T_{\rm eff}/{\rm K} < 3650$. Plot a histogram of the rotation periods you obtain, and overplot a histogram of the McQuillan et al. periods for the same stars. One of the main claims in the McQuillan paper is that the period distribution is bimodal at fixed temperature. Do you still find a bimodal distribution when using your periods? Discuss what, physically, a bimodal period distribution might imply.

Required checkpoint for Monday, March 16. By 4:00 pm, submit a pdf to bCourses containg the plots you made for parts 2.k and 2.m.

(n) At least under certain conditions, the rotation period of a star can be used to constrain its age. Measuring ages of stars from rotation periods is called "gyrochronology". Why does this (sometimes) work, and what are the conditions under which it is expected to work? When might it not work?

(o) Use the empirical color – rotation period – age relation from Meibomb et al. 2009 (specifically, the functional form proposed by Barnes 2007, with free parameters calibrated to M35) to estimate ages of the sample of stars whose periods you fit in part (m). Plot period versus color for a range of ages as predicted by this model, and then plot age versus period for stars in the narrow temperature range you are considering. You can estimate the color corresponding to this temperature range using isochrones. Assuming most stars are on the main sequence, what mass does this color and temperature range correspond to?

Plot the inferred age distribution. What can you infer about the age of the Galactic disk? Is the age of the oldest stars you find similar to what you might expect given what we know about the Milky Way? If not, what might account for this?

(p) You likely found that some stars have short rotation periods ($P < 1\,{\rm week}$), which would imply very young ages. There are significantly more of these fast-rotating stars than predicted from age–rotation rate models, given other constraints on the Milky Way's recent star formation history. Check out Simonian et al. 2018 for an investigation of this issue, and briefly discuss the proposed solution.

Now also read through Section 5 of Meibom et al. 2009. What is *their* proposed explanation from the rapidly-rotating low-mass stars? Are the two papers' explanations compatible?