Optimizing Parameter Searches using Gaussian Process Regression

David P. Fleming
Department of Astronomy
University of Washington, Seattle
Seattle, WA 98105
dflemin3@uw.edu

Matthew C. Wilde
Department of Astronomy
University of Washington, Seattle
Seattle, WA 98105
mwilde@uw.edu

Abstract

For simulations involving a large number of initial conditions, it is infeasible to sufficiently sample parameter using convential methods such as grid searches or random sampling due to the curse of dimensionality. Other methods must be employed in order sufficiently sample parameter space to make statistical inferences about simulation results. We model the results of 10,000 simulations involving tidal and dynamical interactions between theoretical planets in the Proxima Centauri system ([1]) as a function of initial conditions. Using fit uncertainties derived from bootstrapping and Gaussian process (GP) regression, we identify regions in parameter space where our models are uncertain and hence where additional simulations are required. We find our linear regression bootstrapping heuristic decently approximates the more robust GP uncertainties. We consider a parameter search complete if a model can accurately predict the results of unseen data above some user-defined threshold. We test numerous models and find that ensemble methods perform well with XGBoost ([2]) yielding the best performance.

1 Introduction

Astronomers recently discovered a planet orbiting the closest star to the Sun, Proxima Centauri b (hereafter Proxima b) [1]. Proxima b's proximity to Earth and the fact that it resides in its star's habitable zone, the region around a star where liquid water could exist, provides an excellent opportunity to potentially detect life on another world. Therefore, characterizing this system is a major focal point for exoplanet astronomy moving forward. The discovery data hints at the possibility of another planet exterior to Proxima b's orbit, Proxima c. Dynamical interactions between Proxima b and the theoretical Proxima c could have interesting implications for the dynamics and potential habitability of Proxima b.

We simulated such a system using the code `VPLANET` [3] which models the tidal and orbital interactions of Proxima b, a theoretical Proxima c and the central star in order to constrain the orbit of Proxima c if it exists. If the interactions between Proxima b and c perturb the orbit of Proxima b such that its simulated orbit is not consistent with the properties derived from observations, we can rule out the orbital properties of Proxima c for that realization. Hence, we seek to maximize the time in which Proxima b lies within 3 standard deviations of its currently observed orbital parameters given all these interactions in order to place constraints on the orbital properties of Proxima c if it exists. In order to simulate such a system, we need to search over properties such as orbital eccentricity and inclination, mass, distance from the star, tidal coefficients and more for each planet and physical

properties for the star results in 14 parameters to search over even after conservative assumptions are made about the system. Given that these simulations require a large parameter space and that the simulations are computationally expensive since we simulate the system's evolution over 7 billion years, it is infeasible to simulate a representative sample of all planetary configurations in order to place any statistical constraints on the existence of Proxima c and its potential orbit.

We seek to fit a model which predicts the time in which Proxima b lies within 3 standard deviations of its currently observed orbital parameters as a function of the initial conditions. Here, we fit the results of these simulations with simple linear models and use bootstrapping and GP regression to identify regions in parameter space where the fit performs poorly, identifying points where new simulations are required. We then seek to fit the results of our simulations to see if models can learn the underlying interactions to replace the model evaluation and complete the parameter search.

2 Dataset

We have 10,000 simulations of the tidal and dynamical orbital interactions of planets in a theoretical Proxima Centauri planetary system. These computationally expensive simulations produce times series data of how each planet's parameters evolve as a function of time over 7 billion years. We parse the dataset to yield the target variable, what fraction of time Proxima b lies within 3σ of its observed orbit, as a function of 14 of initial simulation conditions that are randomly sampled from physically-informed priors on the dynamics of the posited planetary system. The data produced by a given simulation lives in a file, one for each body, in a simulation directory. For our entire dataset, this results in 10,000 directories each with three output files for a total of 30,000 files which require parsing. We iterated through each simulation directory, read each body's output files for its parameters as a function of time using pandas ([4]), and stored all the time-series results in an HDF5 file ([5]). Once the HDF5 file had been assembled, we processed the data to a matrix containing the target variable as a function of initial conditions.

3 Feature Creation

In order to maximize the the accuracy of our models, we created an additional 12 features that are physically meaningful functions of initial conditions to complement the original 14. For example, for each planet, we create total and normal angular momentum proxy features of the form $\sqrt{1-e^2}$ and $\sqrt{1-e^2}\cos i$ for orbital eccentricity e and inclination i. We call this augmented feature set Physical. We synthesized more features to probe the effects of model complexity by transforming the Physical set to all monomials of degree 2 including cross-terms. This transformation, deemed Polynomial, yielded about 200 total features and allows us to examine the bias-variance trade-off for our models.

4 Where to simulate next

Since it is infeasible to run simulations that span all of parameter space, we develop a heuristic that allows us to approximately span this space based on fitting models to the results of simulations as a function of initial conditions. When we learn the model to the data, we can use methods discussed below to associate an uncertainty to the model's predictions on unseen data. A large uncertainty for a prediction indicates that that region of parameter space requires additional simulations for the space to be sufficiently sampled. Below, we test GP regression-derived uncertainties and bootstrapped uncertainties to see how well they identify where future simulations should be ran and how their performances compare.

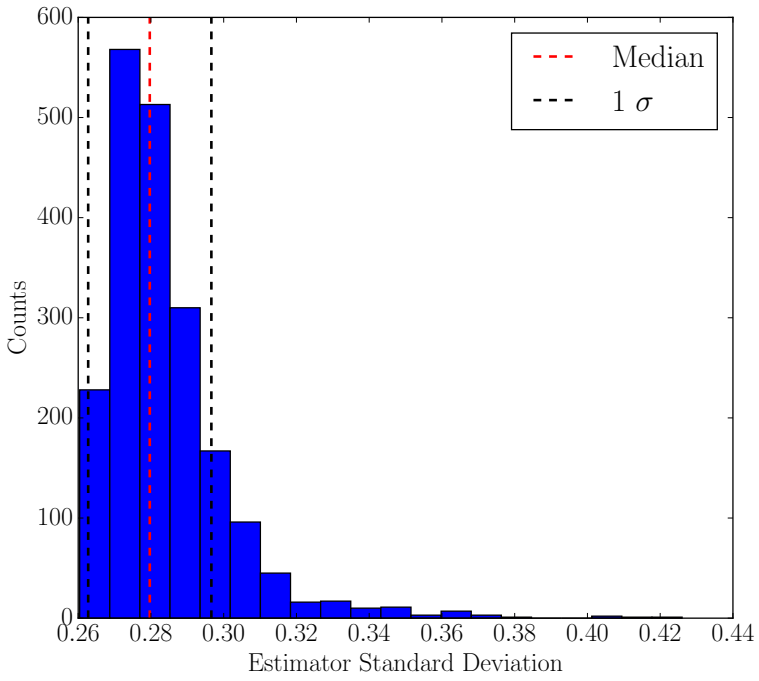

Figure 1: Histogram of the standard deviations from GP regression using the rational quadratic kernel.

4.1 Gaussian Process Regression

GPs are a natural choice for associating uncertainty with a prediction given their Bayesian nature. The predictions from GPs have an associated posterior of the following form

$$P(f_*|X_*, X, f) = N(f_*|\mu_*, \Sigma_*) \tag{1}$$

where f_* is the predicted function outputs, X_* is the unseen testing set, and X, f compose the training data ([6]). In this formalism,

$$\mu_* = \mu(X_*) + K_*^T K^{-1}(f - \mu(X)) \tag{2}$$

and

$$\Sigma_* = K_{**} - K_*^T K^{-1} K_* \tag{3}$$

for covariance kernel $K_{ij} = k(X_i, X_j)$.

We fit GPs with RBF, Matern and Rational Quadratic kernels on the training data and derive uncertainties on the testing set predictions via the standard deviation of the predictive posterior distribution. If the uncertainty on a prediction is greater than 1σ above the median uncertainty, we say that that region of parameter space requires additional simulations. Figure 1 displays a histogram of the testing set uncertainties for a GP with a Rational Quadratic kernel and shows that our heuristic identifies the few points in parameter space where more simulations are required. With such a heuristic, we can identify where to simulate next in parameter space, run the simulations and use those results to retrain the model, improving its performance and increasing the estimators confidence in those regions and predictive ability. With this retrained model, we then again predict where it is uncertain and repeat the process.

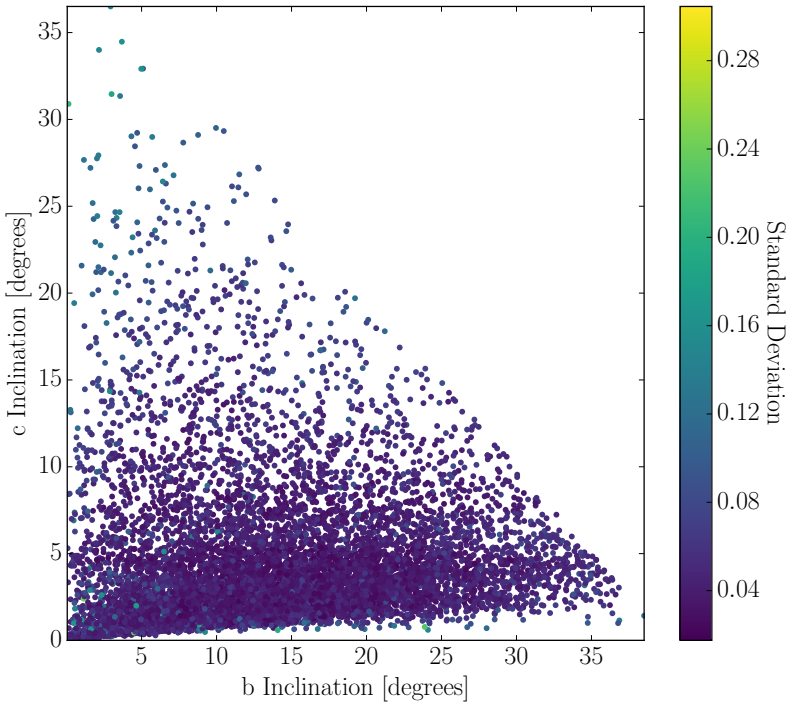

Figure 2: Two-dimensional projection of simulation initial conditions projected into inclination space for Proxima b and c. The colors of each point are the uncertainties derived from ridge regression bootstrapping.

4.2 Bootstrapping

Gaussian processes are quite computationally expensive to train, scaling as $O(N^3)$ for N samples. Given this cost, we explored faster, less expensive methods such as bootstrapping, e.g. [7]. Bootstrapping, or fitting on various re-sampled realizations of the training set with replacement, allows us to derive robust uncertainties on the testing set. For each bootstrap, we predicted on the testing set and compute the standard deviation of all the bootstrapped model predictions as our uncertainty proxy using the online method of [8]. We performed bootstrapping with linear regression and ridge regression using 100 bootstraps. We explored how the number of bootstraps influenced our standard deviation calculation and found that using an order of magnitude more realizations did not significantly impact the fit.

To gauge the performance of our bootstrapping method, we plotted the inclinations of Proxima b and Proxima c shown in the Figure 2 and colored each simulation by the bootstrapping-derived standard deviation of the fit. The linear cut off is due to a mutual inclination prior of 40 degrees stemming from dynamical stability constraints and the numerical stability of our secular models. In general, systems with high mutual inclination are dynamically active and causing large orbital fluctuations and thus are unlikely to be consistent with the observations of Proxima b. Both inclinations are sampled from separate Gaussian priors. As expected, we observe a lower standard deviation where there is a higher density of simulations which happens at lower mutual inclination angles and a higher standard deviation where the simulations are sampled sparsely. At high eccentricity and high mutual inclinations where the model is uncertain, planet-planet interactions become more important. Our large model uncertainties in this region of this area of parameter space indicate that we need to simulate more.

4.3 GP, bootstrapped uncertainties comparison

We gauge the performance of our bootstrapping uncertainties heuristic by comparing it with the more robust GP method by computing the fractional disagreement between a truth estimator and another evaluated on the testing set. Given the uncertainties inherent in GP regression, we consider those models truth. We compute fractional disagreement as the 0/1 loss between which testing set points the bootstrapping method predicts as too uncertain (one standard deviation) and requiring more simulations and which points the true GP model predicts as too uncertain. We do not prefer one GP model over another since they all perform similarly on the testing data (see Table 2). The results of the comparison are given in Table 1. We find that the GP fits tend to agree with each other while the linear models perform surprisingly well with less than 10% disagreement with the RBF kernel fit! This result indicates the simple linear bootstrapping could be a feasible replacement for the slower, more computationally expensive GP method.

Table 1: Model Estimator Error Comparison.

Truth	OLS	RR	GP Matern	GP RBF	GP RQ
GP Matern	0.125	0.125	0	0.045	0.031
GP RBF	0.081	0.081	0.045	0	0.075
GP RQ	0.156	0.156	0.031	0.075	0

5 When can we stop simulating?

Since the goal of this project is to identify when a parameter search is complete and we can then use a learned model instead of running additional simulations, our feature vector for a given sample must be a function of simulation initial conditions, such as planet orbital properties. Ideally, an appropriately trained model will fit the data and could replace running additional computationally expensive simulations. We fit our data set using numerous methods such as ordinary least squares (OLS), ridge regression (RR), ensemble methods like Random Forest (RF) and XGBoost [2], and GP regression using scikit-learn [9]. To quantify model performance we used scikit-learn's score (R^2) where a score of 1 is the best possible and also the mean square loss (MSE). We trained each model on 8,000 simulations and tested on the remaining 2,000. We tuned all hyperparameters using k=5 fold cross-validation. For both XGBoost and Random Forest Regression, we regularized by limiting the maximum depth of the decision trees where this hyperparameters was tuned using k-fold cross-validation. The results of these models on the training and testing set for both the Physical and Polynomial feature sets are shown in Tables 2 and 3.

Table 2: Physical feature set fit results.

Model	Train MSE	Test MSE	Train R^2	Test R^2	Estimator Median Std
OLS	0.097308	0.099976	0.575670	0.561866	0.013340
RR	0.097311	0.099923	0.575658	0.562102	0.016322
RF	0.014023	0.033080	0.941388	0.861286	-
GP Matern	0.050634	0.081951	0.788365	0.656356	0.279772
GP RBF	0.065208	0.083292	0.727450	0.650734	0.281678
GP RQ	0.046516	0.081516	0.805579	0.658183	0.279678
XGBoost	0.005621	0.028202	0.976505	0.88174	-

We found that both linear models, Ordinary Least Squares and Ridge Regression, performed comparably as seen in Tables 2 and 3 implying, at least for these models, that regularization is not

Table 3: Polynomial feature set fit results.

Model	Train MSE	Test MSE	Train R^2	Test R^2	Estimator Median Std
OLS	0.083643	0.093350	0.650397	0.608556	0.049163
RR	0.083747	0.092761	0.649963	0.611030	0.046600
RF	0.013843	0.033862	0.942140	0.858009	-
XGBoost	0.00871	0.028451	0.963597	0.880698	-

important. We do notice a somewhat significant improvement in the MSE and the R^2 values when using the Polynomial feature set compared to the Physical feature set. As expected, a more complex model such as GPs perform better than simple linear regression models while we find that ensemble methods perform the best. As an example, we show in Figure 3 the case of training GP with a Matern kernel with only the Physical features. We clearly see that the GP is a biased estimator and could benefit from additional features and more data as the cross-validation scores continues to increase as more samples are added. Training the GP methods with the Polynomial feature set could help improve its performance but this is prohibitively slow. We find that XGBoost consistently yields the best performance. Even with cross-validation tuned regularization parameters, more complex models seem to overfit as seen and regularization of the depth of decision trees is important. With more data to train the XGBoost method, we could perhaps get to an improved performance around a score of 0.9 - 0.95 where overfitting is less of an issue and we would no longer have to run simulations. Using XGBoost to train on the dataset while performing bootstrapping to get the uncertainties requires running both, however XGBoost can be trivially run on a cluster to speed up performance ([2]).

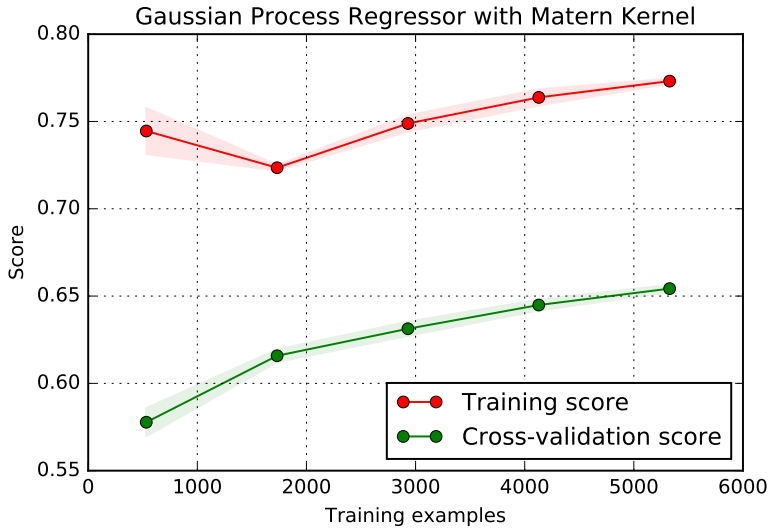

Figure 3: Learning curve using GP with a Matern kernel on the Physical feature set. The low score indicates that GPs are biased. Given the large discrepancy between the training and validation curve, adding more training samples will most likely increase generalization. The shaded 1σ errors are the standard deviation of the k-fold cross-validation iterations.

Curiously, we see an order of magnitude difference in the median bootstrapped uncertainty between the linear models and the Gaussian processes-derived uncertainty. Despite this difference, both methods seem to predict the same regions in parameter space in which to run further simulations which is all we need. This discrepancy is perhaps due to the use of a white kernel in our Gaussian processes which sets a noise floor and allows the matrix inversion to be numerically stable.

6 Conclusions

Our goals for this project were twofold: determine where in parameter space to run further simulations and to determine if we could learn the model well enough to no longer require more simulations. To this effect, we tried a suite of machine learning techniques to determine which performed the best. We found that complex ensemble methods such as XGBoost sufficiently learn the model as a function of simulation intial conditions and with more data we can perform well enough such that we no longer need to run simulations. In addition, running simple linear regression and using a bootstrapping scheme to determine the uncertainty of points in parameter space performs nearly as well as much more computationally expensive methods such as Gaussian processes at measuring uncertainty. This is a particularly exciting result given how relatively computationally-inexpensive training a linear regression model is relative to GPs with a complexity of $O(d^2 N N_{boots})$ for d features, N samples and N_{boots} bootstraps. Therefore using such a method to explore parameter space will mainly be limited by the time it takes to evaluate the forward model. Using our heuristic, we have found that we require more simulations in the high eccentricity-high mutual inclination space as not only was this space sparsely sampled given our priors, but also at high eccentricities and mutual inclinations, the planets more strongly dynamically interact making this a physically interesting region of parameter space.

In this work, we have identified a work-flow for exploring a high-dimensional parameter space that is more computationally efficient than a massive grid search. First, one evaluates the forward model to generate an initial dataset that can be split into a training and testing set. With the training and testing set, one learns a sufficiently complex model on the forward model initial conditions, perhaps using XGBoost, to see if this model can perform well enough on unseen data to replace the forward model, `VPLANET` ([3]) in our case. Next, one performs bootstrapped linear regression as described above to identify points in the testing set where the predictor is uncertain. This step identifies regions in parameter space where additional simulations are required. Then use the forward model to run these simulations, and repeat until a learned model can appropriate predict the results of the forward model. Schemes such as the one presented in this work are necessary for any study that hopes to map a high-dimensional parameter space.

This parameter-space exploring scheme can also be generalized to classification problems. For example, a logistic or softmax regression model can produce classification probabilities which can be transformed into an uncertainty proxy. For the case of binary classification, if the logistic model predicts a point is a certain class with a probability of 50%, for example, this prediction could be deemed uncertain and hence this part of parameter space could require more simulations allowing for a similar work-flow as discussed above to be used to map out simulation space.

References

[1] G. Anglada-Escudé, P. J. Amado, J. Barnes, Z. M. Berdiñas, R. P. Butler, G. A. L. Coleman, I. de La Cueva, S. Dreizler, M. Endl, B. Giesers, S. V. Jeffers, J. S. Jenkins, H. R. A. Jones, M. Kiraga, M. Kürster, M. J. López-González, C. J. Marvin, N. Morales, J. Morin, R. P. Nelson, J. L. Ortiz, A. Ofir, S.-J. Paardekooper, A. Reiners, E. Rodríguez, C. Rodríguez-López, L. F. Sarmiento, J. P. Strachan, Y. Tsapras, M. Tuomi, and M. Zechmeister. A terrestrial planet candidate in a temperate orbit around Proxima Centauri. , 536:437–440, August 2016.

[2] T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. *ArXiv e-prints*, March 2016.

[3] R. Barnes, R. Deitrick, R. Luger, P. E. Driscoll, T. R. Quinn, D. P. Fleming, B. Guyer, D. V. McDonald, V. S. Meadows, G. Arney, D. Crisp, S. D. Domagal-Goldman, A. Lincowski, J. Lustig-Yaeger, and E. Schwieterman. The Habitability of Proxima Centauri b I: Evolutionary Scenarios. *ArXiv e-prints*, August 2016.

[4] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[5] The HDF Group. Hierarchical data format version 5, 2000-2010.

[6] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[7] A. K. Jain, R. C. Dubes, and C. C. Chen. Bootstrap techniques for error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):628–633, Sept 1987.

[8] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.