First Whole Cell Parameter Estimation sub-Challenge

Alex Williams

August, 2013

1 Summary of Overall Approach

The challenge boils down to a complex, high-dimensional regression problem. We are asked to infer 15 perturbations that were delivered to a subset of 30 identified model parameters. These parameters (which can be though of as dependent variables) are estimated based on large amounts of "high-throughput data." In this document I describe the statistical techniques I have used to solve this problem. Importantly, the techniques I outline below are complementary to an analysis of the "sub-models." That is, the general search strategy I describe can be constrained and thus improved if one were to gain insight from the sub-models.

I have written code to estimate the parameters of the whole model, given the high-throughput data that is generated for each simulation. This model can be stated very simply:

$$\vec{p} = f(\vec{x}),$$

where $\vec{p}$ is the estimated "perturbation vector" (which encodes the perturbation delivered to each of the 30 parameters of interest), f is a non-linear function, and $\vec{x}$ is a vector that contains all of the highthroughput data for the mutant wildtype model (provided freely to contest participants). The vector $\vec{p}$ has 30 elements, each of which represents the proportional change in each parameter (e.g. if the third parameter in the list - the kcat of Tmk - is halved, then the third element of $\vec{p}$ is equal to 3).

In practice, I found that the above problem is intractable because of the large number of variables in the high throughput data leading to a very large

vector $\vec{x}$. Thus I perform a principal components analysis of the highthroughput data before fitting the model. This reduces the dimensionality of $\vec{x}$ to be on the order of 50 components.

The nonlinear function f is fit by a collection of regression trees using the Random Forests technique. This is a popular technique in the field of machine learning. Its popularity stems from the fact that we do not need to have an initial guess for the form of the non-linear function f. Additionally, the algorithm cleverly avoids the overfitting problem by probabilistically sampling from the training data. The random forest was fit based on the highthroughput data of 1128 whole cell simulations.

2 Improvements - Compressed Sensing

A substantial improvement, which I have not had the time to implement yet, would be to incorporate the constraint that the perturbation vector $\vec{p}$ is sparse. This piece of information is critical, and is widely studied in the context of "compressed sensing". Essentially, one should be able to improve the fit by penalizing the L1 norm of the estimated perturbation vector $\vec{p}$ (in theory, at least).

Additionally, I am in the process of simulating more simulations which I am now doing in triplicate. Averaging the highthroughput data across these replicates has the potential to improve the fit, since the stochasticity of the model can be quite influential (especially in terms of the data stored in "rxnFluxes"). In fact, I found some models that seemed to fit the data quite well, but failed when submitted to Bitmill. This was apparantly due to trial-to-trial variability in the rxnFlux data, which was revealed by averaging over 8 trials.

3 Explanation of Code

Running the script "s007_simplified_random_forest_script" should generate some estimates of the perturbations to the cell. This script can be found in the "analysis" folder. I have added the prefix "alex_" to almost all of my written functions to distinguish them from Jonathan Karr's original code. Due to severe time constraints, I have not been able to comment and clean up all of my code. Please email me at alex.h.willia@gmail.com if you have further questions.

Full code can be found at: https://github.com/ahwillia/WholeCell