hCSI - Implementation Information

This supplementary readme describes the exact details of the Python hCSI implementation, going into more technical detail than the algorithm outline in Penfold et al. (2012). It is not necessary for using the iPlant app, but it is here to document the implementation in greater depth.

1 The hCSI Algorithm

This subsection is largely redundant with the information in Penfold et al. (2012), but is here to provide the exact forms of the sampling formulas that are used in the implementation.

Given d datasets $D = \left\{D^1, ..., D^d\right\}$, a set of Gaussian process hyperparameters $\Theta = \left\{\theta^1, ..., \theta^d\right\}$ and temperature parameters $\beta = \left\{\beta^1, ..., \beta^d\right\}$, the joint distribution can be written as

$$\begin{aligned}\mathbb{P}(Pa^1(i), ..., Pa^d(i), Pa^*(i), \Theta, \beta|D) = \mathbb{P}(\Theta)\mathbb{P}(\beta)\mathbb{P}(Pa^*(i))\times \\ \prod_{j=1}^{d} \mathbb{P}(D^j|Pa^j(i), \theta^j)\mathbb{P}(Pa^j(i)|Pa^*(i), \beta^j)\end{aligned} \tag{1}$$

with $Pa^j(i)$ being the parents of gene i in dataset j and $Pa^*(i)$ being the parents of gene i in the hypernetwork. The priors $\mathbb{P}(\Theta)$ and $\mathbb{P}(\beta)$ are gamma-distributed, while $\mathbb{P}(Pa^*(i))$ is uniform across all possible parental sets. We have

$$\mathbb{P}(D^j|Pa^j(i), \theta^j) = \frac{1}{Z}\mathbb{P}(\mathbf{y}^j|\mathbf{X}^j, Pa^j(i), \theta^j)$$

where $\mathbb{P}(\mathbf{y}^j|\mathbf{X}^j, Pa^j(i), \theta^j)$ is the marginal likelihood of a zero-mean Gaussian process model with input $\mathbf{X}^j$ (the time shifted expression of the parent genes) and output $\mathbf{y}^j$ (the expression of the child gene), as described in Rasmussen and Williams (2005). Z is a normalising constant. We also have

$$\mathbb{P}(Pa^j|Pa^*(i), \beta^j)) = \frac{1}{Z}\exp\left(-\beta^j\mathcal{E}(Pa^j(i), Pa^*(i))\right)$$

with $\mathcal{E}(Pa^j(i), Pa^*(i))$ representing the Hamming distance between $Pa^j(i)$ and $Pa^*(i)$ – treating the parental gene sets as actual sets, the Hamming distance between them is the size of the union minus the size of the intersection between them. Once again, Z is a normalising constant.

All sampling performed in hCSI is based on Equation 1, with most of the terms in the product remaining unchanged for any individual sampling. For example, proposing a new parental set in a single dataset j will only see $\mathbb{P}(D^j|Pa^j(i),\theta^j)\mathbb{P}(Pa^j(i)|Pa^*(i),\beta^j)$ for that particular j change, with all the other components of the product staying the same.

The four sampling steps performed as part of hCSI are:

1. Gibbs updates of the parental sets in each of the d datasets

$$Pa^1(i) \sim \frac{1}{Z_1}\mathbb{P}(\mathbf{y}^1|\mathbf{X}^1, Pa^1(i), \theta^1)\exp\left(-\beta^1\mathcal{E}(Pa^1(i), Pa^*(i))\right)$$

$$\vdots$$

$$Pa^d(i) \sim \frac{1}{Z_d}\mathbb{P}(\mathbf{y}^d|\mathbf{X}^d, Pa^d(i), \theta^d)\exp\left(-\beta^d\mathcal{E}(Pa^d(i), Pa^*(i))\right)$$

2. Gibbs update of the hypernetwork

$$Pa^*(i) \sim \frac{1}{Z}\prod_{j=1}^{d}\exp\left(-\beta^j\mathcal{E}(Pa^j(i), Pa^*(i))\right)$$

3. Metropolis-Hastings updates of the Gaussian process hyperparameters in each of the d datasets (with θ_{new} obtained by adding a Gaussian random variable to θ)

$$p_1(accept) = \min\left(1, \frac{\mathbb{P}(\mathbf{y}^1|\mathbf{X}^1, Pa^1(i), \theta^1_{new})\mathbb{P}(\theta^1_{new})}{\mathbb{P}(\mathbf{y}^1|\mathbf{X}^1, Pa^1(i), \theta^1)\mathbb{P}(\theta^1)}\right)$$

$$\vdots$$

$$p_d(accept) = \min\left(1, \frac{\mathbb{P}(\mathbf{y}^d|\mathbf{X}^d, Pa^d(i), \theta^d_{new})\mathbb{P}(\theta^d_{new})}{\mathbb{P}(\mathbf{y}^d|\mathbf{X}^d, Pa^d(i), \theta^d)\mathbb{P}(\theta^d)}\right)$$

4. Metropolis-Hastings updates of the temperature parameters in each of the d datasets (with β_{new} obtained by adding a Gaussian random variable to β)

$$p_1(accept) = \min\left(1, \frac{\frac{1}{Z^1_{new}}\exp\left(-\beta^1_{new}\mathcal{E}(Pa^1(i), Pa^*(i))\right)\mathbb{P}(\beta^1_{new})}{\frac{1}{Z^1}\exp\left(-\beta^1\mathcal{E}(Pa^1(i), Pa^*(i))\right)\mathbb{P}(\beta^1)}\right)$$

$$\vdots$$

$$p_d(accept) = \min\left(1, \frac{\frac{1}{Z^d_{new}}\exp\left(-\beta^d_{new}\mathcal{E}(Pa^d(i), Pa^*(i))\right)\mathbb{P}(\beta^d_{new})}{\frac{1}{Z^d}\exp\left(-\beta^d\mathcal{E}(Pa^d(i), Pa^*(i))\right)\mathbb{P}(\beta^d)}\right)$$

Like previously, Z always represents a normalising constant. These are computed as the sums of the unnormalised distributions, and dividing by them ensures that the distribution adds up to 1 (and is fit for sampling).

2 Technical Implementation Details

hCSI makes use of the Python CSI implementation due to its optimised likelihood computation. $\sim 99\%$ of hCSI's computational time goes to likelihood evaluation in the Gibbs updates of the individual dataset network structures. A feature not in use in the iPlant version of the app (due to the resource limitations) is the `--LikelihoodPool` command line argument, which opens up a second level of parallelisation when performing likelihood computations. The option is provided for users who want to speed up run time on resource-heavy local computational platforms.

The model is initialised completely randomly. Each perturbation has a random parental set combination assigned to it as the starting value with equal weighting, and the same happens for the hypernetwork. The starting Gaussian process hyperparameters are sampled from U(0,1) for each dataset independently (in contrast to the original Matlab implementation, where a single set of sampled values are copied over for each dataset as the starting hyperparameters), all of the temperatures are initialised at 0.1.

In contrast to the original Matlab implementation, all four sampling procedures (individual networks, hypernetwork, hyperparameters, temperatures) are performed in tandem, with a default total of 25,000 steps. In Matlab, one of those operations was chosen randomly as a single step, with 100,000 default total steps.

In order to avoid the same RNG chains being used, the RNG seed is initiated based on the row number of the gene being used as the child at that point.

The hyperparameter and temperature sampling steps feature the addition of a random Gaussian variable, which is sampled from N(0,1) and multiplied by a scaling constant. The scaling constant is initialised at 0.1 and re-evaluated every 100 steps, aiming to keep the Metropolis-Hastings acceptance rate at around 0.25. In the case of less than 15 accepted jumps (out of 100), the transition operator is deemed to be not localised enough and the scaling constant is multiplied by 0.9. If over 35 jumps get accepted, the scaling constant is multiplied by 1.1 to make the sampling be less localised. This, as well as the actual sampling of the hyperparameter/temperature values, is performed independently for hyperparameters and temperatures and each dataset.

References

Penfold, C. A., Buchanan-Wollaston, V., Denby, K. J., and Wild, D. L. Nonparametric Bayesian inference for perturbed and orthologous gene regulatory networks. *Bioinformatics*, 28(12):i233–i241, 2012.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. The MIT Press, 2005. ISBN 0-262-18253-X.