Practical Session 2: Gene Regulatory Network Inference

In this practical session, we will explore **different approaches** to infer gene regulatory networks (GRNs) from simulated mRNA and protein data. Participants have to connect to the following github repo:

`https://github.com/eliasventre/TP_inference`

The session is divided into three parts:

1. Presentation of the challenge of GRN inference and 3 different approaches to be explored (corresponding to the sections of this document) $\sim$ 20mns;
2. Discovering the repo with data simulations (*simulation_distributions.py* and *simulation_trajectories.py*) for different network structures (*Network4* and *Network8*), inference implementation (*inference.py*) with basic method already provided (ex: *methods/correlations.py*), and evaluation (*evaluation.py*) $\sim$ 20mns;
3. Exploring **one of the approaches** described below: model-agnostic inference, protein-based model-specific inference using NeuralODEs on the deterministic limit of the model, and combined mRNA-protein model-specific inference usign the closest method to the state-of-the-art on this model. $\sim$ 1h20.

Starting with the code

- In each file, you can define the variable*outfile = "Network4"* or *outfile = "Network8"* to choose to simulate, infer or evaluate methods on this network.
- in the file **inference.py**, you can also define the variable `type_data = "traj"` or `type_data = "distrib"` depending on the type of data you want to use for the inference (real trajectories or sequence of independent snapshots).
- After running the simulation files (*simulation_distributions.py* and *simulation_trajectories.py*), you can use the jupyter notebook `visualize_data.ipynb` to see what the network and the simulations look like. You can in particular verify that in terms of UMAP, real trajectories or distributions are similar, and that if single trajectories are very noisy the mean trajectories are already almost determinist with 100 cells per timepoint.
- For running the inference, you have to run: **python inference.py -m method** where *method* is the name of one of the python files in the folder **methods** where you have defined a new class *NetworkInference()* and an associated function *fit()*. You can also run **inference.py -m all** to infer with all the methods successively. each time, the algorithm will create or write in a subfolder *method* in the folder corresponding to the network.

1 Option 1: Model-agnostic inference

This first part focuses on general-purpose methods that do not rely on mechanistic knowledge of the model.

1.1 Information-theory based methods

Compare different information-theory based tools. Data can be chosen either from mRNA or protein datasets (or together) for comparison. Most of these methods aim to estimate a quantity $I(g1, g2)$ for each pair of genes characterizing the strenght of the interaction.

Exemples for the predictor model include:

- Classical correlations: Pearson, Spearman
- Mutual information
- Conditional mutual information to remove indirect links (ex: PIDC [2]).

1.2 Machine learning-based methods

Compare different ML-based tools. Data can be chosen either from mRNA or protein datasets (or together) for comparison. These methods generally consist in following this recipe for every gene g:

1. Use the other genes as predictors by choosing a model $Pr(X_{-g}) = X_g$ (P is for example the GRN matrix for simple linear models;
2. Solve a problem of the form:

$$\min ||X_g - Pr(X_{-g})||^2 + \lambda ||Pr||_1$$

3. Take the non-zero coefficients of Pr as potential regulators.

Exemples for the predictor model include:

- Generalized Linear Model
- Random Forest (ex: GENIE3 [4])

1.3 Incorporating temporal information

Now try to extend your favorite approach to include the temporal dimension. For example, Transfer entropy has been used for identfying GRN from time-series [6] and Genie3 has been extended in DynGenie3 [3].

2 Option 2: Model-specific inference from protein data

This part leverages mechanistic knowledge and focuses on proteins, which directly reflect regulatory dynamics.

2.1 Neural ODE on protein trajectories

Implement a *neuralODE* corresponding to the deterministic limit of the mechanistic model, which becomes (for a parameter s_1 well chosen):

$$\frac{\partial}{\partial t} P_t = d_1(k^\theta_{on}(P_t) - P_t).$$

where for each gene i, $k^\theta_{on,i}(P_t)$ est une sigmoïde.

Calibrate the network using protein **trajectories** in a first place.

2.2 Pseudotrajectories from single-cell snapshots

In practice, full cellular trajectories are not observed. Pseudotrajectories can be reconstructed using optimal transport algorithms. In this case, you can reconstruct pseudotrajectories prior to the neuralODEs reconstruction using optimal transport algorithms. This framework has been applied for example in the method FLECS [1].

Optimal transport in a nutshell: given distributions x_i and x_j at two time points t_i and t_j, one can compute a transport plan P minimizing:

$$\min_P \sum_{k,l} P_{kl} \|x_i^k - x_j^l\|^2$$

subject to marginal constraints. For each x_i^k, its pseudo-descendant will be defined as $x_j^{l_k}$ where l is defined as:

$$l_k := argmin_l P_{kl}.$$

Python package: POT (Python Optimal Transport). A code snippet is provided to interpolate and reconstruct pseudotrajectories with this method.

A code will be provided in *align_data.py* in the github repo.

3 Option 3: Model-specific inference combining mRNA and protein data

This part explores the basis of state-of-the-art reverse-engineering methods for the mechanistic model (Harissa, CARDAMOM).

3.1 Inference from paired mRNA-proteins distributions

This approach relies on a key observation: marginal distributions of mRNA are well-approximated by mixtures of negative binomial distributions:

$$M(t) \sim \sum_{z \in Z} \mu_t(z) \prod_{i=1}^{n} \text{Beta}\left(\frac{k_{z,i}}{d_{0,i}}, \frac{k_{off,i}}{s_{0,i}}\right).$$

under the assumption that for all gene i, $k_{on,i}$ is approximately piecewise constant:

$$\forall P,\, k_{on,i}^{\theta}(P) \in \{k_{z,i}\}_{z \in \mathbb{N}}.$$

Thus, for each cell c, we can attribute to each mRNA level M_c a frequency mode vector k_{z_c} by finding the one maximizing the probability:

$$M_c(t) \sim \text{Beta}\left(\frac{k_{z_c,i}}{d_{0,i}}, \frac{k_{off,i}}{s_{0,i}}\right).$$

The inference method can be then performed in two steps:

1. Step 1: Find for each cell c its corresponding frequency mode vector k_{z_c}. A code will be provided for Step 1 in *binarize_data.py* in the github repo.

2. Step 2: use the corresponding protein values P_c to maximize a loss between the k_{z_c} and the $k_{\text{on}}(P_c)$), e.g:

$$\theta^* = \text{argmin}_\theta \sum_c ||k_{on}^{\theta}(P_c) - k_{z_c}||^2.$$

   You can also try different loss and compare.

3.2 Inference from only mRNA distributions (more advanced)

In reality, proteins are often not observed. The principle of the first version of CARDAMOM consists in approximating proteins by their equilibrium conditional on the inferred frequency modes [5].

A second version use optimal transport with a mechanistic-specific cost, alternating between GRN and protein trajectories reconstruction. You can try these version by implementing the *cardamom* package as a separate method.

Option: For interested people (but it's very unlikely to have time to do it within the 2h), you could implement a mix of Options 2 and 3 by implementing a neuralODEs fitting the evolution of the frequency modes, and then compute both plausible protein trajectories and GRNs.

Conclusion

This practical session allowed participants to compare different approaches for gene network inference:

- General-purpose methods based on information theory or machine learning.
- Model-specific methods exploiting protein dynamics and knowledge of the Harissa model.
- Advanced methods using CARDAMOM to approximate proteins and infer regulatory parameters from mRNA data alone.

References

[1] Paul Bertin, Joseph D Viviano, Alejandro Tejada-Lapuerta, Weixu Wang, Stefan Bauer, Fabian J Theis, and Yoshua Bengio. A scalable gene network model of regulatory dynamics in single cells. *arXiv preprint arXiv:2503.20027*, 2025.

[2] Thalia E Chan, Michael PH Stumpf, and Ann C Babtie. Gene regulatory network inference from single-cell data using multivariate information measures. *Cell systems*, 5(3):251–267, 2017.

[3] Vân Anh Huynh-Thu and Pierre Geurts. dyngenie3: dynamical genie3 for the inference of gene networks from time series expression data. *Scientific reports*, 8(1):3384, 2018.

[4] Vân Anh Huynh-Thu, Alexandre Irrthum, Louis Wehenkel, and Pierre Geurts. Inferring regulatory networks from expression data using tree-based methods. *PloS one*, 5(9):e12776, 2010.

[5] Elias Ventre, Ulysse Herbach, Thibault Espinasse, Gérard Benoit, and Olivier Gandrillon. One model fits all: combining inference and simulation of gene regulatory networks. *PLoS Computational Biology*, 19(3):e1010962, 2023.

[6] Stephen Y Zhang and Michael PH Stumpf. Inferring cell-specific causal regulatory networks from drift and diffusion. In *The 2022 ICML Workshop on Computational Biology. Baltimore, Maryland, USA*, 2022.