Branching Gaussian processes

Alexis Boukouvalas, James Hensman, Magnus Rattray

January 25, 2017

Contents

1 Introduction

Single-cell gene expression data can be used to uncover cellular progression through different states of a temporal transformation, e.g. during development, differentiation or disease. A single pseudotime parameter can be assigned to each cell to represent its cellular state. We propose a non-linear model to estimate the branching tree structure. The model provides a log likelihood ratio estimate of the evidence for branching and a posterior estimate of the most likely branching location as well as a confidence interval.

[8] developed a tractable GP model for the identification of a single perturbation point. They define a novel kernel that constrains two functions to intersect at a single point. The bifurcation point is identified by numerically approximating the posterior and selecting a point estimate. The model is used to identify when a gene becomes differentially expressed in time course gene expression data under a control and perturbed condition. In their approach

Table 1: Scenarios used to generate synthetic data. The specification of each scenario includes the number of genes branching at each location. A branching location of 1.1 refers to a non-branching gene.

Scenario	Branching	Description
0	[0.2, 20], [1.1, 20]	Single branching
1	[0.2, 20], [0.6, 20]	All genes branching
2	[0.2, 15], [0.6, 15], [1.1, 10]	Multiple branching points
3	[0.2, 15], [0.6, 15], [1.1, 10]	Short lengthscale
4	[0.1, 3], [0.7, 27], [1.1, 10]	Majority of late branching genes
5	[0.1, 5], [0.3, 5], [0.5, 5], [0.7, 5], [1.1, 20]	Many branching locations
6	[0.2, 20], [1.1, 20]	High branching variance

all data points have been labelled with the branch that generated them. The ordering of time points is also assumed known and fixed.

We extend their approach by removing the labelling assumption and include additional steps in the workflow in order to apply it to single-cell data. These include an estimate of the pseudotime using [7].

2 Synthetic data:

We sample from a branching GP. Samples where branches cross after bifurcation point are rejected to avoid penalizing linear methods like MFA. We fit the branching GP model as described in appendix D.

We evaluate three methods, the mixture of factors analysers [3], the BEAM approach [7] and the branching GP model. The synthetic scenarios are summarized in Table 1.

The log likelihood ratio of the branching GP can be used to rank the evidence of ranking for each. Similar measures exist for the MFA and BEAM method. We compare the three methods on their ability to discriminate branching genes from non-branching genes. The metric we have selected for this comparison is the area under the curve which has been classically used to evaluate classification models. In Table 2, the branching GP achieves consistently good performance whilst MFA performance varies significantly. The BEAM method is unable to discriminate well.

3 Application to droplet single-cell RNA-seq data

Our aim is to perform a gene by gene application of the probabilistic branching model to see if we can differentiate early and late branching genes. Our analysis is on the postmitotic normalised data. The data consists of 3299 cells and 2405 genes.

Firstly we apply the Monocle [7] method to get a global pseudotime, branching estimation as well as label allocation. Monocle identifies lots of segments and

Table 2: Evaluating the discriminative ability of each method to identify branching genes. The area under the curve (AUC) metric is reported for each scenario and we compare the mixture of factor analysers, BEAM and branching GP methods.

Scenario	MFA	BEAM	BGP
0	0.84	0.6	0.95
2	0.22	0.59	0.9
3	0.66	0.59	0.95
4	0.81	0.66	0.92
5	0.91	0.52	0.98
6	0.92	0.63	0.83

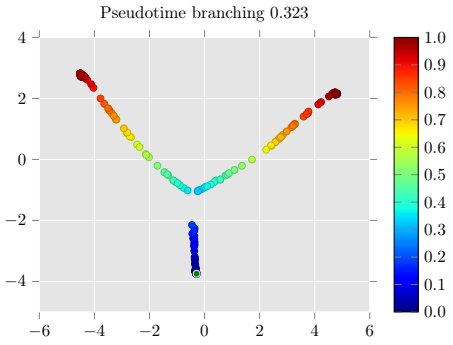
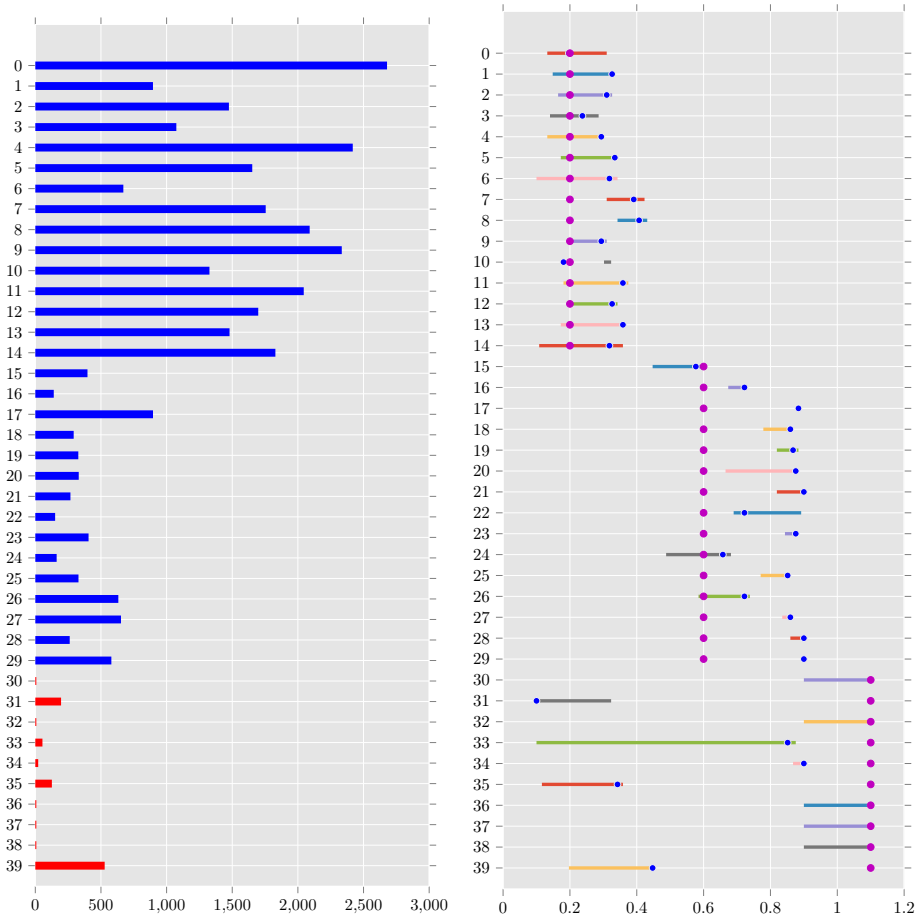


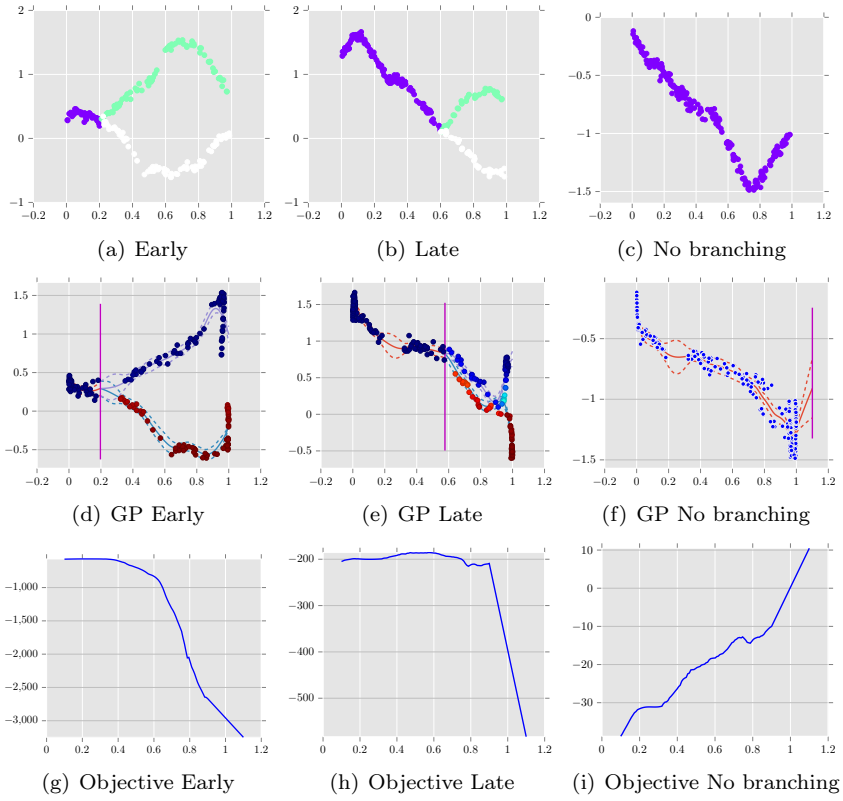
Figure 1: Synthetic data, scenario 3. Monocle2 latent space and pseudotime estimation. The global branching estimated by Monocle is also shown. The green circle denotes the starting cell.



(a) Log likelihood. Branching (blue) vs non-branching (red)

(b) Posterior summary

Figure 2: Synthetic data, scenario 3. Posterior summary



(a) Early (b) Late (c) No branching

(d) GP Early (e) GP Late (f) GP No branching

(g) Objective Early (h) Objective Late (i) Objective No branching

Figure 3: Synthetic data, scenario 3. Samples, model predictions and objective function shown for scenario 3 representing early, late and no branching. The data shown in the GP plots do not match the samples due to the estimation of pseudotime by Monocle.

we simplify the problem to two main branches. We have selected the starting cell by looking at the dimension reduction visualization. However this should be further validated, perhaps by looking at known marker genes. We then score genes by difference in the t-statistic for branch end states. The full pipeline is described in appendix D.

We apply the branching Gaussian process (GP) model to identify early and late branching genes. We apply the model on the top 500 genes from the previous step. There are multiple source of errors such as dropout and misclassification of cells by the global branch estimation. We therefore treat the labels as uncertain and the branching GP model may reassign cells to the other branch. This also allows for more flexible noise modeling including dropout without the need for a complex probabilistic noise model.

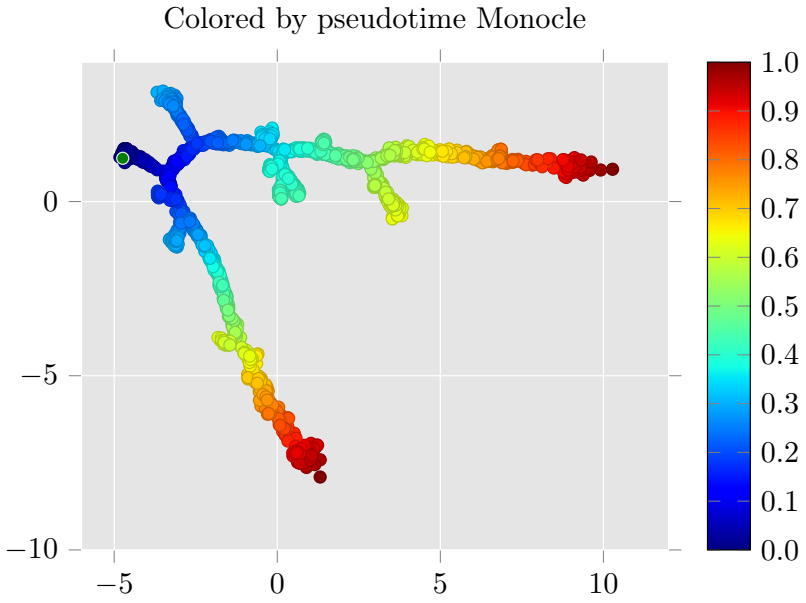


Figure 4: Dropseq data. Monocle reduce spaced coloured by pseudotime.

4 Theory

We can define a kernel that describes two functions crossing at a single point [8][1]. In this work however we cannot assume the data points are labelled with the function the generated them. [5, 4] define this is the data association problem and propose the overlapping mixture of GPs to address it.

However using the OMG model to identify branches in pseudotime would be wasteful since we the functions are not independent as they are constrained

[1] This result is re-derived in Appendix A.

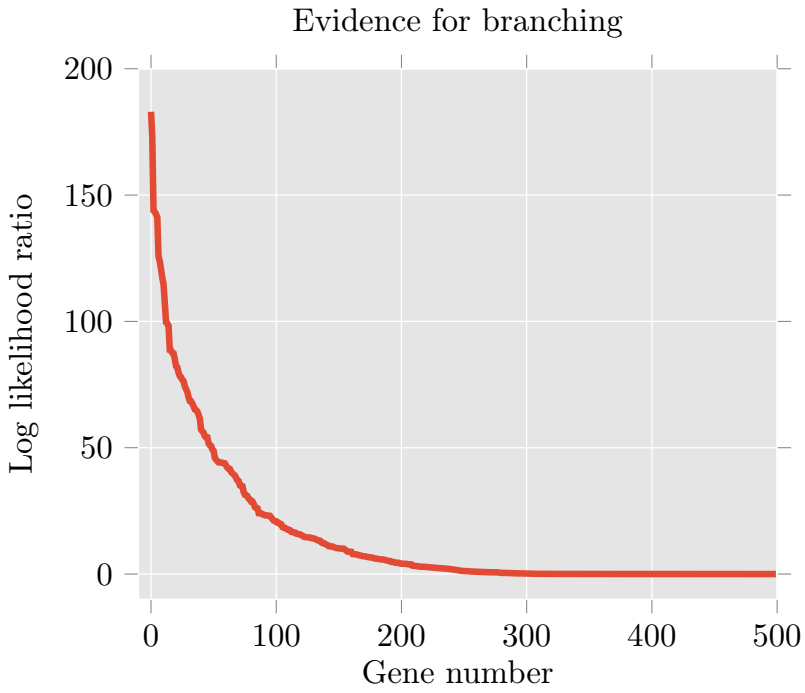


Figure 5: Dropseq data. Log-likelihood rank of top 500 genes.

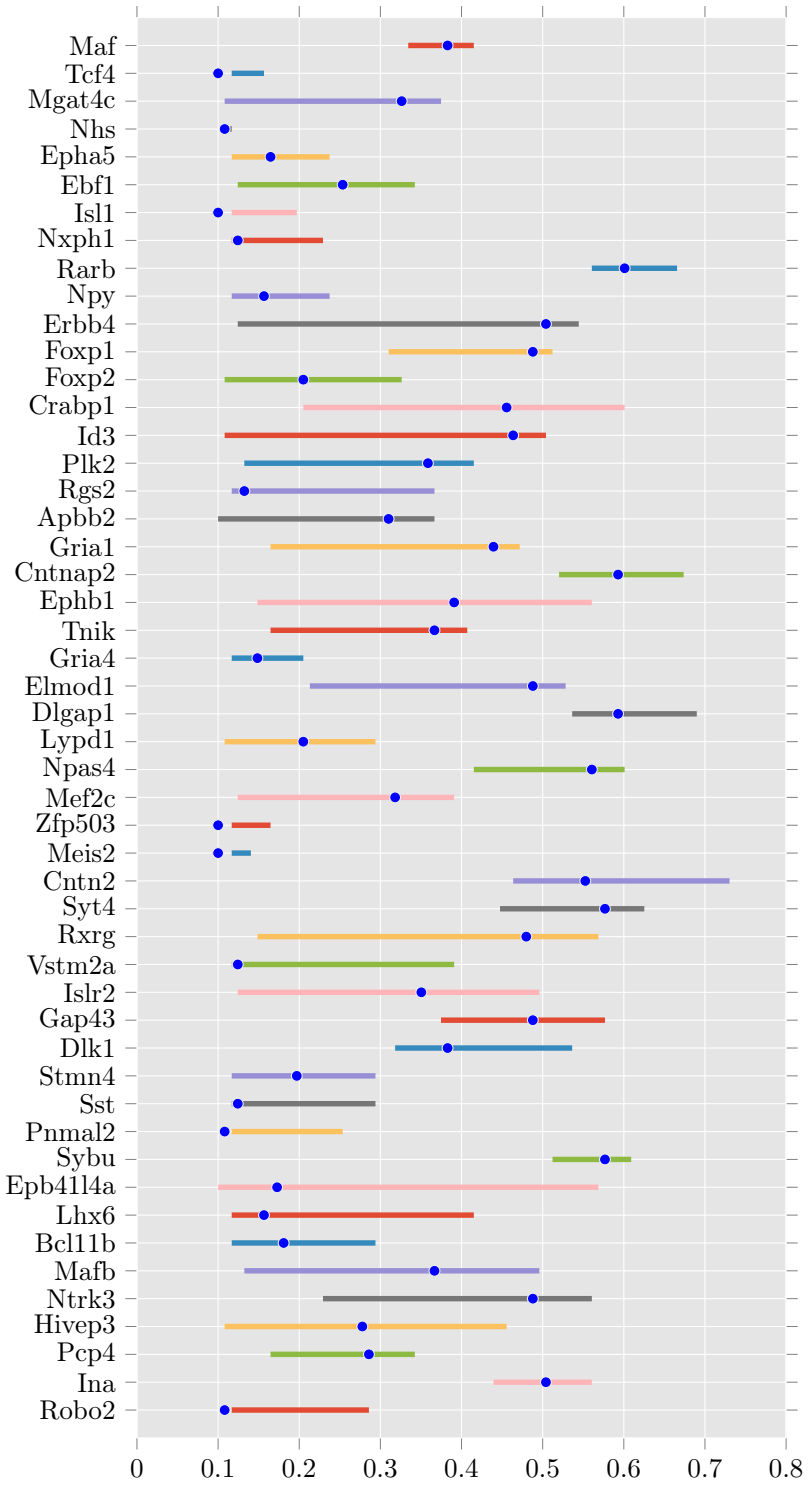


Figure 6: Dropseq data. Posterior branching summary.

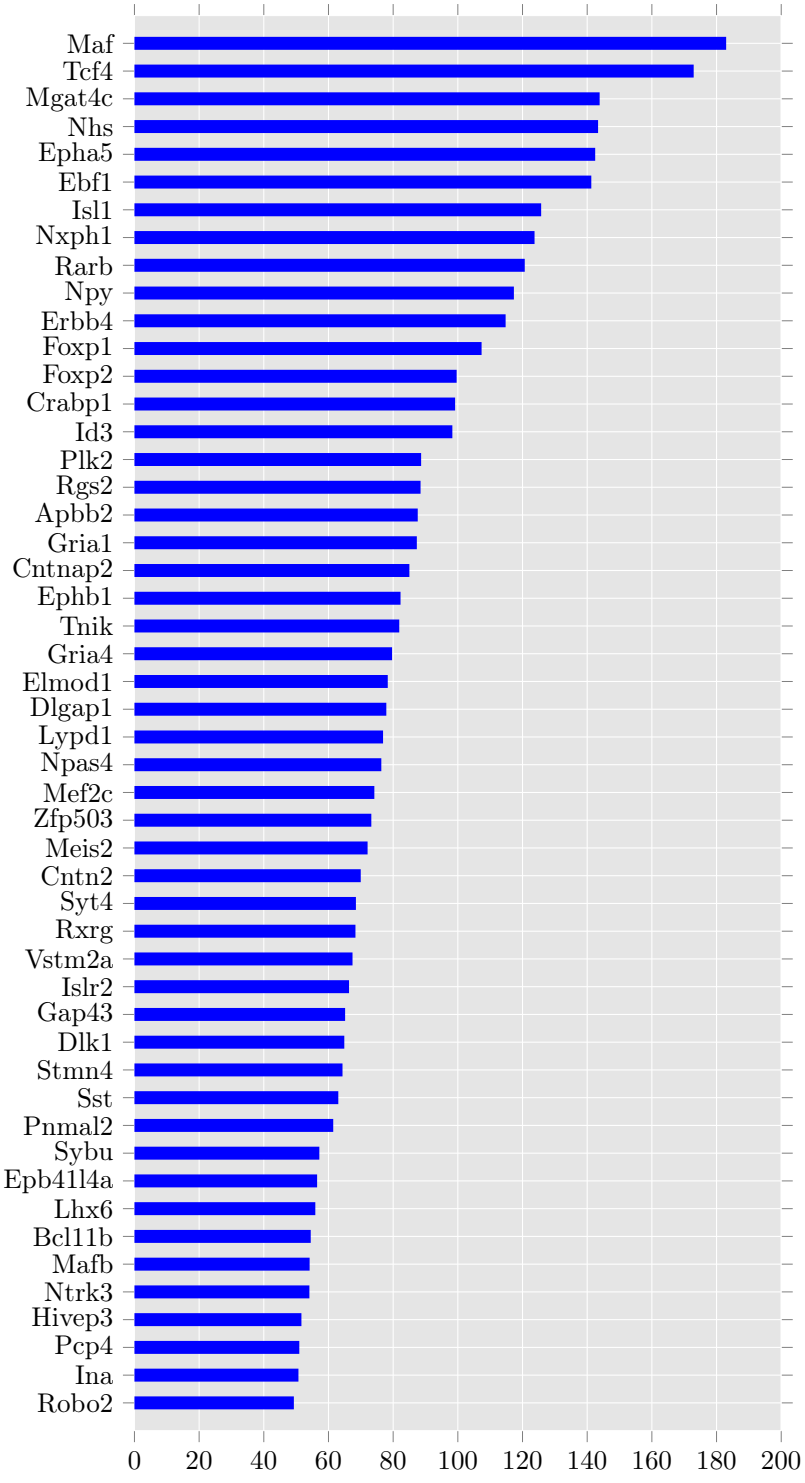
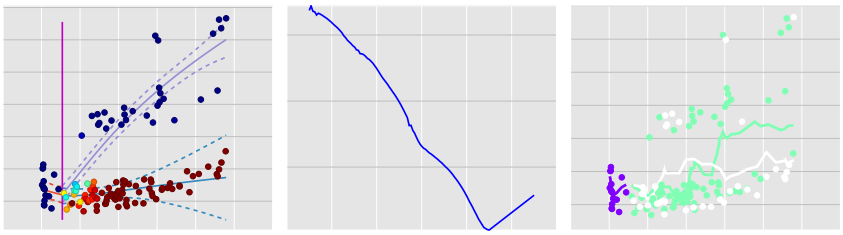
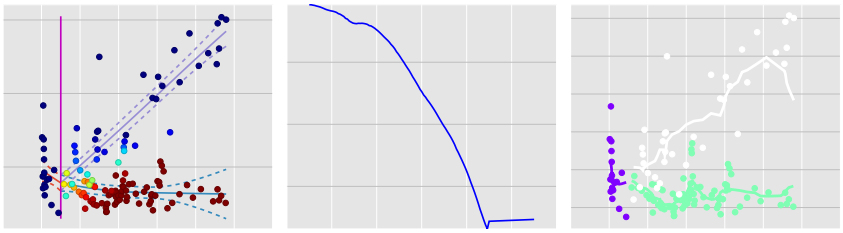


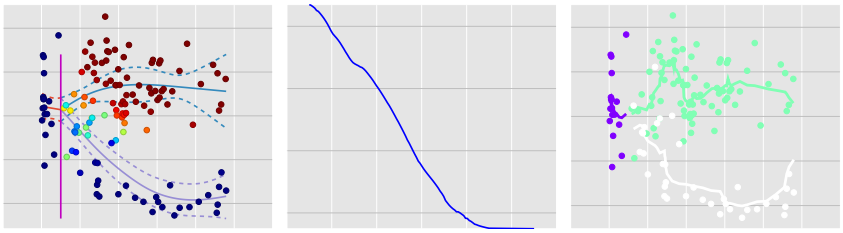
Figure 7: Dropseq data. Log-likelihood ratio



(a) Nhs (b) Nhs (c) Nhs

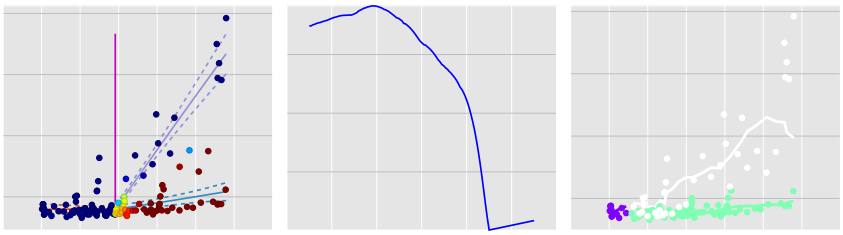


(d) Tcf4 (e) Tcf4 (f) Tcf4

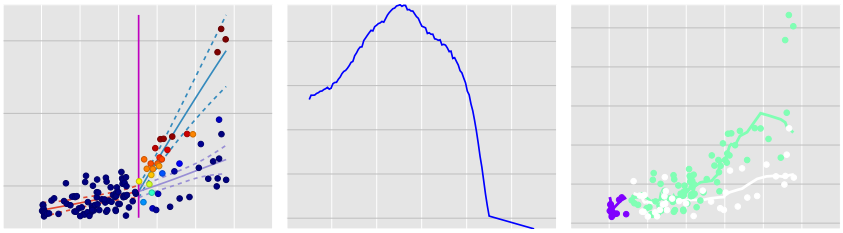


(g) Meis2 (h) Meis2 (i) Meis2

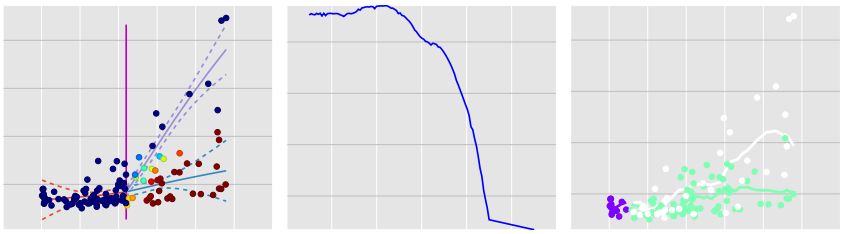
Figure 8: Dropseq data. Early branching genes



(a) Maf (b) Maf (c) Maf

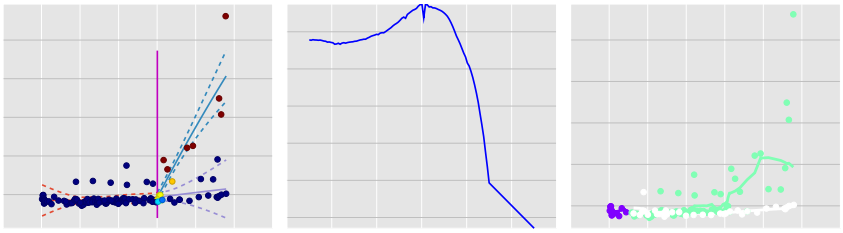


(d) Ina (e) Ina (f) Ina

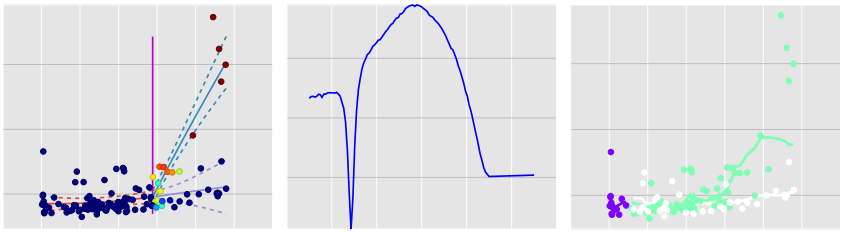


(g) Gria1 (h) Gria1 (i) Gria1

Figure 9: Dropseq data. Medium branching genes



(a) Rarb (b) Rarb (c) Rarb



(d) Sybu (e) Sybu (f) Sybu

Figure 10: Dropseq data. Late branching genes

to intersect at the branching point. Rather we believe there is underlying tree structure where branches occur at bifurcation points.

Let the likelihood be $p(\mathcal{Y}|\mathcal{Z}, X)$ where $\mathcal{Z}$ is the $N \times M$ binary indicator matrix, X is the pseudotime $(N \times 1)$ and $\mathcal{Y}$ the data $(N \times D)$. $M \gg N$ is the number of possible allocations for all points; for example if we have a single branching point whose location is unknown each point has 3 possible allocations and therefore $M = 3 \times N$.

We assume a Gaussian error model with common observation noise σ^2, independent dimensions d and we place a GP prior on the latent function values

$$p(f|X) = \mathcal{GP}(0, K(X)) \tag{1}$$

The likelihood can then be written as

$$p(\mathcal{Y}|\mathcal{Z}, X) = \int_f p(f|X) \prod_d \mathcal{N}(\mathcal{Y}_d | \mathcal{Z}f, \sigma^2 I_N) \tag{2}$$

where f is the $M \times 1$ vector of latent noise free function values. Let X_* the expanded version of X where is entry of the latter is replicated M times resulting in a $(N \times M) \times 1$ vector. Then the marginal likelihood is

$$p(\mathcal{Y}|\mathcal{Z}, X) = \prod_d \mathcal{N}(\mathcal{Y}_d | 0, vec(Z) K(X_*) + \sigma^2 I_N) \tag{3}$$

5 Extensions

Extending to multiple branching points is possible by discretizing the pseudotime space and trying all possible trees. Although computationally expensive, this is not intractable as pseudotime is a one-dimensional space. The number of permutations governed by Catalan number [2].

We also constrain tree to be rooted at start of pseudotime, that is the branching grows with pseudo-time. In theory we could allow for a full graph in pseudotime but prefer the simpler approach as it is also simpler to elicit the number of leafs in a binary tree. We could also use K-trees, where at each bifurcation point we could have $K > 2$ branches but again for reasons of simplicity we select the binary tree, that is $K = 2$.

A Joint distribution of two functions constrained at a point

We prove the result in Section 2.3 [**?**]. The following Gaussian Identity will be useful (Section 2.3.3, page 93, Equations 2.113-2.117 [2]). Let $p(x) = N\left(x|\mu, \Lambda^{-1}\right)$ and $p(y|x) = N\left(y|Ax + b, L^{-1}\right)$ then we have

$$p(y) = N\left(y|A\mu + b, L^{-1} + A\Lambda^{-1}A^T\right) \tag{4}$$

.

As [**?**] discuss the predictive mean of a GP conditional on a single point (u, x_p) is $\mu(X) = \frac{k_X}{k_p}u$ and $C(X, X) = K_X - \frac{k_X k_X^T}{k_p}$where we denote $K_X = K(X, X)$ the $N \times N$ test covariance, $k_X = K(X, x_p)$ the $N \times 1$ train-test matrix and $k_p = K(x_p, x_p)$ the scalar. Note that the latter does not depend on the kernel length scale but only on the variance terms (process+nugget).

We now integrate out the latent response value at the bifurcation point by assuming

$$p(u) = N(u\,|0, k_p) \tag{5}$$

We also assume conditional independence between the responses f and g, that is $p(f, g|u) = p(f|u)\,p(g|u)$. Then we have

$$p(f, g) = \int p(f, g|u)\,p(u)\,du \tag{6}$$

If we observe process f at points X and process g at point Z then we have

$$\begin{aligned} p(f(X), g(Z)|u) &= \mathcal{N}\left(\begin{pmatrix} f \\ g \end{pmatrix} \middle| \begin{pmatrix} \mu(X) \\ \mu(Z) \end{pmatrix}, \begin{pmatrix} C(X, X) & 0 \\ 0 & C(Z, Z) \end{pmatrix}\right) & (7) \\ &= \mathcal{N}\left(\begin{pmatrix} f \\ g \end{pmatrix} \middle| \begin{pmatrix} \frac{k_X}{k_p} \\ \frac{k_Z}{k_p} \end{pmatrix} u, \begin{pmatrix} C(X, X) & 0 \\ 0 & C(Z, Z) \end{pmatrix}\right) & (8) \end{aligned}$$

[2] See `https://en.wikipedia.org/wiki/Catalan_number`.

an $2N \times 2N$ matrix where $C(.,.)$ as defined previously.

We can now use the result in Equation (4) to solve the integral in Equation (6).

$$\begin{aligned} p(f(X), g(Z)) &= \mathcal{N}\left(\begin{pmatrix} f \\ g \end{pmatrix} \middle| 0, \begin{pmatrix} C(X,X) & 0 \\ 0 & C(Z,Z) \end{pmatrix} + \begin{pmatrix} \frac{k_X}{k_p} \\ \frac{k_Z}{k_p} \end{pmatrix} k_p \begin{pmatrix} \frac{k_X}{k_p} \\ \frac{k_Z}{k_p} \end{pmatrix}^T \right) \\ &= \mathcal{N}\left(\begin{pmatrix} f \\ g \end{pmatrix} \middle| 0, \begin{pmatrix} C(X,X) & 0 \\ 0 & C(Z,Z) \end{pmatrix} + \begin{pmatrix} \frac{k_X k_X^T}{k_p} & \frac{k_X k_Z^T}{k_p} \\ \frac{k_Z k_X^T}{k_p} & \frac{k_Z k_Z^T}{k_p} \end{pmatrix} \right) \\ &= \mathcal{N}\left(\begin{pmatrix} f \\ g \end{pmatrix} \middle| 0, \begin{pmatrix} K_X & \frac{k_X k_Z^T}{k_p} \\ \frac{k_Z k_X^T}{k_p} & K_Z \end{pmatrix} \right) \end{aligned}$$

which proves the result in Equation 6 of [**?**].

B Crossing kernel

The crossing kernel is

$$p(f(X), g(Z)) = \int p(f|u)\, p(g|u)\, p(u)\, du = \mathcal{N}\left(\begin{pmatrix} f \\ g \end{pmatrix} \middle| 0, \begin{pmatrix} K_X & k_X k_p^{-1} k_Z^T \\ k_Z k_p^{-1} k_X^T & K_Z \end{pmatrix}\right)$$

where $k_X = K(X, x_p)$ the $N \times 1$ train-test matrix and $k_p = K(x_p, x_p)$ the branching kernel.

C Efficient implementation using TensorFlow.

We have implemented our approach in the GPflow framework [6] which leverages efficient computation using Tensorflow [1].

In the latter we build an executable graph using python that can then be executed across many CPU/GPUs or in a distributed environment. The goal is to easily transition from research prototype to production.

D Workflow

The workflow we use in details is given below.

1. Apply Monocle 2 method to get a global pseudotime, branching estimation as well as label allocation. Monocle identifies lots of 'segments' and we simplify the problem to two main branches.

2. Rank genes by median distance or t-statistic between the tips of the branches.

3. Normalise the data: $y = (y - y_{min})/(np.percentile(y, 99) - y_{min})$. This reduces the effective range of the gene expression reducing the effect of outliers and noise.

4. Smooth the data using a running mean for each each branch. This avoids more complex than Gaussian noise models.

5. Initialise allocation matrix Phi using Monocle. If branch 1, branch 2 and 3 [0.5 0.5], if branch 2 [0.75 0.25].

6. Stretch branches to be same length, as in Monocle 2.

7. Estimate a branching GP using the Monocle 2 branching point. We estimate the kernel hyperparameters as well.

8. We are using a Matern 3/2 kernel - perhaps a rougher kernel would be more appropriate.

9. Estimate hyperparameters at global branching location.

10. Perform a grid search on other branching locations keeping hyperparameters fixed.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1 edition, 2006.

[3] Kieran Campbell and Christopher Yau. Probabilistic inference of bifurcations in single-cell data using a hierarchical mixture of factor analysers. *bioRxiv*, 2016.

[4] Miguel Lázaro-Gredilla and Steven Van Vaerenbergh. A gaussian process model for data association and a semidefinite programming solution. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(11):1967–1979, 2014.

[5] Miguel Lázaro-Gredilla, Steven Van Vaerenbergh, and Neil D. Lawrence. Overlapping Mixtures of Gaussian Processes for the data association problem. *Pattern Recognition*, 45(4):1386–1395, apr 2012.

[6] Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *arXiv preprint 1610.08733*, October 2016.

[7] Xiaojie Qiu, Andrew Hill, Yi-An Ma, and Cole Trapnell. Single-cell mrna quantification and differential analysis with census. *Nat Meth*, page btw329, 2016.

[8] Jing Yang, Christopher A Penfold, Murray R Grant, and Magnus Rattray. Inferring the perturbation time from biological time course data. *Bioinformatics*, page btw329, 2016.