# Tradict - mathematical details

Surojit Biswas, Konstantin Kerner, Paulo José Pereira Lima Texeira,
Jeffery L. Dangl, Vladimir Jojic, Philip A. Wigge

## Contents



This document describes the full mathematical details for the concepts presented in the "Tradict algorithm" section, "Building a predictive Multivariate Normal Continuous-Poisson hierarchical model" subsection of the Materials and Methods in the Supplemental Information. Specifically, we present exactly how Tradict uses a selected set of markers to 1) complete the encoding, and 2) to perform decoding.

## 1 Preliminaries

For a matrix $A$, $A_{:i}$ and $A_{i:}$ index the $i^{th}$ column and row, respectively. For a set of indices, $q$, we use $-q$ to refer to all indices not specified by $q$.

We define the continuous relaxation of the Poisson distribution (hereafter, Continuous-Poisson) to have the following density function:

$$f(x|\lambda) = C_\lambda \frac{e^{-\lambda}\lambda^x}{\Gamma(x+1)}$$

where $C_\lambda$ is a normalization constant.

## 2 Model

Tradict uses a Continuous-Poisson Multivariate Normal (CP-MVN) hierarchical model to model the expression of transcriptional programs and all genes in the transcriptome. We begin by building a predictive model of gene expression, and thereafter discuss a predictive model for the expression of transcriptional programs.

Let $z_j$ denote the log-*latent abundance* of gene $j$, such that $\exp(z_j)$ is the *latent abundance* of that gene's TPM measurement, $t_j$. Let $T_j = t_j o$ be the measured number of transcripts of gene $j$. Here o is the

sequencing depth in millions of reads of the sample under consideration. We assume then,

$$z \sim \mathcal{N}(\mu, \Sigma)$$
$$T_j \sim \text{Continuous-Poisson}(\exp(z_j)o)$$

where $\mu$ and $\Sigma$ are $1 \times$ #-genes and #-genes $\times$ #-genes, respectively. In effect, we are assuming that the number of measured transcripts for gene $j$ is some noisy realization of an unmeasured latent abundance $\exp(z_j)$ times the sequencing depth, $o$. The log-latent abundances are then coupled across genes via the Multivariate Normal layer of the model. Note that we could model the TPM measurements directly in the second layer by assuming $t_j \sim \text{Continuous-Poisson}(\exp(z_j))$; however, this formulation does not consider sequencing depth, which can be a valuable source of information when inferring latent abundances for rare/poorly sampled genes [1].

During decoding, we are interested in building a predictive model between markers and all genes in the transcriptome, and are therefore not interested in working with the full model described above. Instead, we need to consider a conditional model of the transcriptome given the log-latent abundances of the markers. Let $m$ be the set of indices for the given panel of selected markers. We have,

$$z_m \sim \mathcal{N}(\mu^{(m)}, \Sigma^{(m)})$$
$$z_{-m}|z_m \sim \mathcal{N}(\mu_{z_{-m}|z_m}, \Sigma_{z_{-m}|z_m})$$
$$T_j \sim \text{Continuous-Poisson}(\exp(z_j)o)$$

Here, $\mu^{(m)}$ and $\Sigma^{(m)}$ refer to mean vector and covariance matrix of $z_m$. Given these, the conditional mean of the log-latent abundances for all non-marker genes can be obtained through Gaussian conditioning. Specifically, for two normally distributed row-vector variables $a$ and $b$ the conditional mean of $b$ given $a$ is given by $\mu_{b|a} = \mu_b + (a - \mu_a)\Sigma_a^{-1}\sigma_{ab}$ and $\Sigma_{b|a} = \Sigma_b - \sigma_{ab}^T\Sigma_a^{-1}\sigma_{ab}$, where $\sigma_{ab}$ is the cross-covariance between $a$ and $b$, and $\Sigma_a$ and $\Sigma_b$ are the covariance matrices of $a$ and $b$, respectively.

Given the expression of a transcriptional program is a linear combination of the latent abundances of its constituent genes, they will be normally distributed given 1) Central Limit Theorem, and 2) the latent abundances themselves are normally distributed (convolutions of normals are normals). Let $s$ be the expression of all transcriptional programs. We posit the following model,

$$z_m \sim \mathcal{N}\left(\mu^{(m)}, \Sigma^{(m)}\right)$$
$$s|z_m \sim \mathcal{N}(\mu_{s|z_m}, \Sigma_{s|z_m})$$

To use these models for prediction, we must learn their parameters from training data. This would complete the process of encoding described in the Supplemental Information. Specifically, we need to learn $\mu^{(m)}$, $\Sigma^{(m)}$, $\mu_s$, $\mu_{z_{-m}}$, $\sigma_{z_m,s}$ and $\sigma_{z_m,z_{-m}}$.

# 3 Encoding

As described in the Supplemental Information, given an estimate of $z_m$, $\hat{z}_m$, inference of $\mu_s$, $\mu_{z_{-m}}$, $\sigma_{z_m,s}$ and $\sigma_{z_m,z_{-m}}$ is straightforward. In lag transforming the entire training TPM expression matrix, $t \in \mathbb{R}^{\text{samples}\times\text{genes}}$, we have an estimate of $z$, $\hat{z} = \text{lag}(t)$ [1]. Thus, an estimate of $\mu_{z_{-m}}$ is given by the usual column-wise sample mean of $\hat{z}_{-m}$.

Let $\Lambda \in \mathbb{R}^{\text{genes}\times\text{transcriptional programs}}$ be a matrix of principal component 1 coefficients over genes for each transcriptional program. Note, that $\Lambda_{ij} = 0$ if gene $i$ is not in transcriptional program $j$. An estimate of $s$ is given by $\hat{s} = \hat{z}\Lambda$, and so an estimate for $\mu_s$, $\hat{\mu}_s$, is given by the usual column-wise mean of $\hat{s}$.

Given $\hat{z}_m$ the cross-covariances, $\sigma_{z_m,s}$ and $\sigma_{z_m,z_{-m}}$, are given by the usual sample cross-covariance between $\hat{z}_m$ and $\hat{s}$ and between $\hat{z}_m$ and $\hat{z}_{-m}$, respectively.

Now, though we could use the lag-transformed values of $t_m$ as our estimate for $z_m$, we have an opportunity to improve this estimate by virtue of having to estimate $\mu^{(m)}$ and $\Sigma^{(m)}$. More specifically, given $z_m$, estimates of $\mu^{(m)}$ and $\Sigma^{(m)}$ are given by – up to some regularization – the usual sample mean and covariance of $z_m$. Furthermore, given $\mu^{(m)}$ and $\Sigma^{(m)}$, we can update our estimate of $z_m$. This suggests an alternating iterative procedure in which we iterate 1) estimation of $\mu^{(m)}$ and $\Sigma^{(m)}$, and 2) inference of $z_m$ until convergence of their joint likelihood. It is the $\hat{z}_m$ that we obtain from this procedure that we use in the cross-covariance calculations above.

### 3.1 Inference of $z_m$ given $\mu^{(m)}$ and $\Sigma^{(m)}$

Suppose Tradict has estimates of $\mu^{(m)}$ and $\Sigma^{(m)}$ given by $\hat{\mu}^{(m)}$ and $\hat{\Sigma}^{(m)}$, and let $T_m = t_m(o \times \mathbf{1}_{1\times\text{markers}})$ be a matrix of the total measured number of transcripts for each marker. Here $o \in \mathbb{R}^{\text{samples}\times 1}$ is a vector of sample sequencing depths in millions of reads. Given these, we can calculate a maximum *a posteriori* (MAP) estimate of $z_m$.

The posterior distribution over $z_m$ is given by

$$\begin{aligned}
p(z_m|o, T_m, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) &= \frac{p(T_m|o, z_m, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})p(z_m|\hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})}{\int_k p(T_m|o, k, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})p(k|\hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})\mathrm{d}k} \\
&\propto \prod_{i=1}^{n} p(T_{im}|o, z_{im}, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})p(z_{im}|\hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) \\
&= \prod_{i=1}^{n}\left[\prod_{j=1}^{|m|} C_{[\exp(z_{ij})o_i]}[\exp(z_{ij})o_i]^{T_{ij}} e^{-[\exp(z_{ij})o_i]}/\Gamma(T_{ij}+1)\right] \\
&\times \frac{1}{\sqrt{2\pi|\hat{\Sigma}^{(m)}|}^{|m|}} \exp\left(-\frac{1}{2}(z_{i:} - \hat{\mu}^{(m)})\mathrm{inv}\left(\hat{\Sigma}^{(m)}\right)(z_{i:} - \hat{\mu}^{(m)})^T\right)
\end{aligned}$$

where for notational clarity we have used $\mathrm{inv}(\cdot)$ to represent matrix inverse.

Given $z$ is a matrix parameter, this may be difficult to solve directly. However, note that given $z_{ij}$, $T_{ij}$ is conditionally independent of $T_{i,-j}$. Additionally, given $z_{i,-j}$, $z_{ij}$ is normally distributed with mean and covariance

$$\begin{aligned}
a_{ij} &= \mu_j^{(m)} + \left(z_{i,-j} - \mu_{-j}^{(m)}\right)\mathrm{inv}\left(\Sigma_{-j,-j}^{(m)}\right)\Sigma_{-j,j}^{(m)} \\
\sigma_j &= \Sigma_{j,j}^{(m)} - \Sigma_{j,-j}^{(m)}\mathrm{inv}\left(\Sigma_{-j,-j}^{(m)}\right)\Sigma_{-j,j}^{(m)}
\end{aligned}$$

respectively. Taken together, this suggests an iterative conditional modes algorithm in which we maximize the posterior one column of $z$ at a time, while conditioning on all others.

Let $\hat{z}$ denote our current estimate of $z$. Our objective is given by,

$$\begin{aligned}
\hat{z}_{ij} &= \underset{z_{ij}|z_{i,-j}}{\mathrm{argmax}} \log p(z_{ij}|T_{ij}, o_i, \hat{z}_{i,-j}, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) \\
&= \underset{z_{ij}|z_{i,-j}}{\mathrm{argmax}} \log p(T_{ij}|z_{ij}, o_i, \hat{z}_{i,-j}, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})p(z_{ij}|\hat{z}_{i,-j}, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) \\
&= \underset{z_{ij}|z_{i,-j}}{\mathrm{argmax}} \log p(T_{ij}|z_{ij}, o_i)p(z_{ij}|\hat{z}_{i,-j}, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) \\
&= \underset{z_{ij}|z_{i,-j}}{\mathrm{argmax}} \log\left[[\exp(z_{ij})o_i]^{T_{ij}} e^{-[\exp(z_{ij})o_i]} \exp\left(-\frac{1}{2\sigma_j}(z_{ij} - a_{ij})^2\right)\right] \\
&= \underset{z_{ij}|z_{i,-j}}{\mathrm{argmax}}\, T_{ij}\exp(z_{ij})o_i - \exp(z_{ij})o_i - \frac{1}{2\sigma_j}(z_{ij} - a_{ij})^2
\end{aligned}$$

Differentiating we get,

$$\begin{aligned}\frac{\partial}{\partial z_{ij}} T_{ij} z_{ij} o_i - \exp(z_{ij}) o_i - \frac{1}{2\sigma_j}(z_{ij} - a_{ij})^2 \\ = T_{ij} o_i - \exp(z_{ij}) o_i - \frac{1}{\sigma_j}(z_{ij} - a_{ij})\end{aligned}$$

Because $z_{ij}$ appears as a linear and exponential term, we cannot solve this gradient analytically. We therefore utilizes Newton-Raphson optimization. For this we also require the Hessian, which is given by,

$$\begin{aligned}\frac{\partial}{\partial z_{ij}} T_{ij} o_i - \exp(z_{ij}) o_i - \frac{1}{\sigma_j}(z_{ij} - a_{ij}) \\ = -\exp(z_{ij}) o_i - \frac{1}{\sigma_j} < 0\end{aligned}$$

Notice the Hessian is always negative-definite, which implies each update has a single, unique optimum.

In practice, the Newton-Raphson updates can be performed in vectorized fashion iteratively for each column of $z$. We generally find that this optimization takes 5-15 iterations (full passes over all columns of $z$) and less than a minute to converge. We refer to the program that performs these calculations as $\hat{z}_m = \texttt{MAP\_z}\left(t, o, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}\right)$.

## 3.2 Complete inference of $\mu^{(m)}$, $\Sigma^{(m)}$, and $z_m$

For complete inference we use the following algorithm

- Initialize $T_m = t_m(o \times \mathbf{1}_{1\times\text{markers}})$, $\hat{z}_m = \text{lag}(t_m)$.
- Until convergence of $\log p(T_m | o, \hat{z}_m, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) + \log p(\hat{z}_m | \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})$, iterate:
  - Update $\hat{\mu}^{(m)}$ and $\hat{\Sigma}^{(m)}$:

$$\begin{aligned}\hat{\mu}^{(m)} &= \frac{1}{\#\text{samples}} \sum_i \hat{z}_{im} \\ \hat{\Sigma}^{(m)} &= \frac{1}{\#\text{samples} - 1} \sum_i (\hat{z}_{im} - \hat{\mu}^{(m)})^T (\hat{z}_{im} - \hat{\mu}^{(m)}) + \lambda \text{diag}\left[\text{cov}\left(\hat{z}_m^{(\text{init})}\right)\right]\end{aligned}$$

  - Update $\hat{z}_m = \texttt{MAP\_z}\left(t, o, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}\right)$.

Here $\text{diag}(x)$ of the square matrix $x$ returns an equivalently sized matrix with only the diagonal of $x$ preserved and 0's for the off-diagonal terms. $\text{cov}(\cdot)$ denotes the usual sample covariance matrix.

Note that in this algorithm we have added a regularization to the estimate of the covariance matrix. This is done in order to ensure stability and to avoid infinite-data-likelihood singularities that arise from singular covariance matrices. This is most often happens when a genes TPM abundance is mostly zero (i.e. there is little data for the gene), giving the multivariate normal layer an opportunity to tightly couple this genes latent abundance to that of another gene, thereby producing a singularity. This regularization is probabilistically equivalent to adding and Inverse-Wishart prior over $\Sigma^{(m)}$. The parameter $\lambda$ controls the strength of the regularization. In practice, we find $\lambda = 0.1$ leads to good predictive performance, stable (non-singular) covariance matrices, and reasonably quick convergence.

# 4 Decoding

During decoding we are given new measured TPM measurements for our markers, $t^*_m \in \mathbb{R}^{\text{query samples} \times |m|}$, and we must make predictions about the expression of all transcriptional programs and the remaining non-marker genes. To do this we first need an estimate of the log-latent abundances $\hat{z}^*_m$ associated with $t^*_m$. Given the estimates $\hat{\mu}^{(m)}$ and $\hat{\Sigma}^{(m)}$ obtained from the training data, we obtain these estimates as

$$\hat{z}^*_m = \texttt{MAP\_z}\left(t^*_m, \mathbf{1}_{\text{query samples} \times 1}, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}\right)$$

Given the learned latent marker abundances, we can calcualte the expected espression of all transcriptional programs as

$$\mathbb{E}[s^*|\hat{z}^*_m] = \mu_{s^*|\hat{z}^*_m} = \hat{\mu}_s + \left(\hat{z}^*_m - \hat{\mu}^{(m)}\right) \text{inv}\left(\hat{\Sigma}^{(m)}\right) \hat{\sigma}_{z_m,s}.$$

Here, $\hat{\mu}_s$ and $\hat{\sigma}_{z_m,s}$ represent estimates of the unconditional mean of $s$ and the cross-covariance matrix between $z_m$ and $s$ previously learned during encoding.

For the remaining, non-marker genes in the transcriptome we have,

$$\begin{aligned}
\hat{t}^*_{ij} = \mathbb{E}[t^*_{ij}|\hat{z}^*_{im}] &= \int_{-\infty}^{\infty} \mathbb{E}[t^*_{ij}|z^*_{ij}] p(z_{ij}|\hat{z}^*_{im}) \mathrm{d}z_{ij} \\
&= \int_{-\infty}^{\infty} \exp(z_{ij}) \mathcal{N}(z_{ij}|\mu_{z_{ij}|\hat{z}^*_{im}}, \Sigma_{z_{ij}|\hat{z}^*_{im}}) \mathrm{d}z_{ij} \\
&= \mathbb{E}_{\mathcal{N}}[\exp(z_{ij})|\hat{z}^*_{im}]
\end{aligned}$$

The Moment Generating Function of a Normal random variable $X$ with mean $\mu$ and variance $\sigma^2$ is given by $M(t) = \mathbb{E}[\exp(tX)] = \exp(\mu t + \sigma^2 t^2/2)$. Therefore we have,

$$\hat{t}^*_{ij} = \mathbb{E}[t^*_{ij}|\hat{z}^*_{im}] = \mathbb{E}_{\mathcal{N}}[\exp(z_{ij})|\hat{z}^*_{im}] = M(1) = \exp\left(\mu_{z_{ij}|\hat{z}^*_{im}} + \frac{1}{2}\Sigma_{z_{ij}|\hat{z}^*_{im}}\right)$$

where,

$$\begin{aligned}
\mu_{z_{ij}|\hat{z}^*_{im}} &= \hat{\mu}_j + \left(\hat{z}^*_{im} - \hat{\mu}^{(m)}\right) \text{inv}\left(\hat{\Sigma}^{(m)}\right) \hat{\sigma}_{z_m,z_j} \\
\Sigma_{z_{ij}|\hat{z}^*_{im}} &= \hat{\sigma}_{jj} - \hat{\sigma}^T_{z_m,z_j} \text{inv}\left(\hat{\Sigma}^{(m)}\right) \hat{\sigma}_{z_m,z_j}
\end{aligned}$$

Here, $\hat{\mu}_j$ and $\hat{\sigma}_{z_m,z_j}$ represent estimates of the unconditional mean of $z_j$ and the cross-covariance matrix between $z_m$ and $z_j$. These were learned from the training data during encoding.

Though this predictor is unbiased, it does not produce a good prediction for most samples. This is due to the right-skew of the Poisson, which drags its mean away from the most likely values. Therefore, in practice, we use as our estimate of $t^*_{ij}$,

$$\hat{t}^*_{ij} = \underset{t}{\text{argmax}}\, p(t|\hat{z}^*_{im}) = \exp\left(\mu_{z_{ij}|\hat{z}^*_{im}}\right).$$

# 5 References

[1] Surojit Biswas. The latent logarithm. *arXiv*, pages 1–11, 2016.