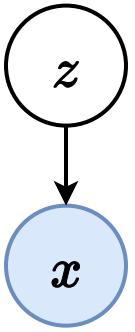
Foundations of Data Science

Lecture 8: Variational inference

Prof. Gilles Louppe
g.louppe@uliege.be

LIÈGE université



Without loss of generality, we consider latent variable models $p(z, x) = p(z)p(x|z)$ with observed variables x and latent variables z.

We want to compute the posterior distribution

$$p(z|x) = \frac{p(z)p(x|z)}{p(x)}$$

but the marginal likelihood $p(x) = \int p(z)p(x|z)dz$ is intractable.

Variational inference

We previously studied MCMC methods that provide asymptotically exact samples from the posterior distribution when the distribution is known up to a normalizing constant.

An alternative to MCMC methods is to cast posterior inference as **optimization**.

Problem statement

We consider a variational family $\mathcal{Q}$ of tractable distributions over the latent variables z. We want to find the variational distribution $q \in \mathcal{Q}$ that is closest to the true posterior distribution $p(z|x)$,

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(p(z|x)||q(z)),$$

where $\mathrm{KL}(p(z|x)||q(z))$ is the forward Kullback-Leibler divergence.

Unfortunately, the forward KL divergence

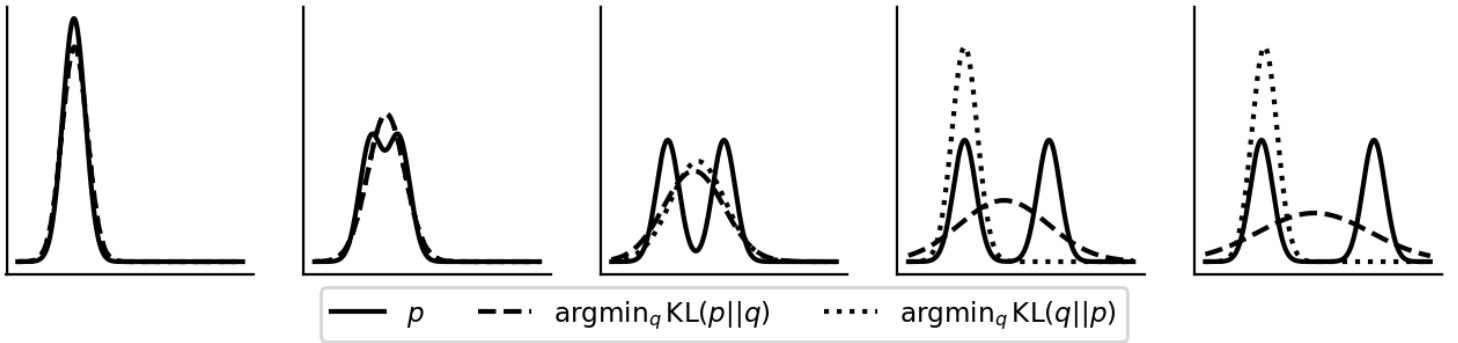
$$\mathrm{KL}(p(z|x)||q(z)) = \mathbb{E}_{p(z|x)}\left[\log \frac{p(z|x)}{q(z)}\right]$$

is not tractable since it requires samples from the true posterior $p(z|x)$ if we want to estimate the expectation.

Instead, we minimize the **reverse KL divergence**

$$\mathrm{KL}(q(z)||p(z|x)) = \mathbb{E}_{q(z)}\left[\log \frac{q(z)}{p(z|x)}\right]$$

where the expectation is taken with respect to the variational distribution $q(z)$, which we can sample from.



Minimizing the forward KL divergence is mode-covering,
while minimizing the reverse KL divergence is mode-seeking.

Evidence lower bound (ELBO)

As for EM, we can re-express the optimization problem in terms of the evidence lower bound as

$$\begin{aligned}
\mathrm{KL}(q(z)||p(z|x)) &= \mathbb{E}_{q(z)}\left[\log \frac{q(z)}{p(z|x)}\right] \\
&= \mathbb{E}_{q(z)}\left[\log q(z)\right] - \mathbb{E}_{q(z)}\left[\log p(z|x)\right] \\
&= \mathbb{E}_{q(z)}\left[\log q(z)\right] - \mathbb{E}_{q(z)}\left[\log p(z,x)\right] + \log p(x) \\
&= \mathbb{E}_{q(z)}\left[\log \frac{q(z)}{p(z,x)}\right] + \log p(x) \\
&= -\mathcal{L}(q) + \log p(x),
\end{aligned}$$

where $\mathcal{L}(q) = \mathbb{E}_{q(z)}\left[\log \frac{p(z,x)}{q(z)}\right]$ is the ELBO.

Therefore, minimizing the reverse KL divergence is equivalent to maximizing the ELBO.

Mean-field variational inference

The mean-field approximation consists in choosing a variational family where the latent variables $z = (z_1, \ldots, z_m)$ are independent under the variational distribution. That is,

$$q(z) = \prod_{j=1}^{m} q_j(z_j).$$

This assumption is often unrealistic, as the target posterior distribution would typically exhibit complex dependencies between the latent variables. However, it can lead to tractable inference algorithms.

Coordinate ascent variational inference (CAVI)

Under the mean-field assumption, we can derive **coordinate ascent updates** for each factor $q_j(z_j)$ by maximizing the ELBO with respect to $q_j(z_j)$ while keeping the other factors fixed.

Proposition. The optimal factor $q_j^*(z_j)$ that maximizes the ELBO is given by

$$q_j^*(z_j) \propto \exp\left(\mathbb{E}_{q_{-j}(z_{-j})}\left[\log p(z, x)\right]\right),$$

where z_{-j} denotes all latent variables except z_j and $q_{-j}(z_{-j})$ is the product of all factors except $q_j(z_j)$, all of which are kept fixed.

Proof. The ELBO can be written as

$$\begin{aligned}\mathcal{L}(q) &= \mathbb{E}_{q(z)}\left[\log p(z,x) - \log q(z)\right] \\ &= \mathbb{E}_{q(z)}\left[\log p(z,x)\right] - \sum_{j=1}^{m} \mathbb{E}_{q_j(z_j)}[\log q_j(z_j)].\end{aligned}$$

Focusing on the j-th factor while keeping the others fixed, we have

$$\mathcal{L}(q_j) = \mathbb{E}_{q_j(z_j)}\left[\mathbb{E}_{q_{-j}(z_{-j})}\left[\log p(z,x)\right]\right] - \mathbb{E}_{q_j(z_j)}[\log q_j(z_j)] + \text{const},$$

which we aim to maximize with respect to $q_j(z_j)$.

By the method of Lagrange multipliers, we introduce a multiplier λ and consider the functional

$$\mathcal{L}(q_j, \lambda) = \mathcal{L}(q_j) - \lambda\left(\int q_j(z_j)dz_j - 1\right)$$

where the constraint ensures that $q_j(z_j)$ integrates to 1.

The functional derivative of $\mathcal{L}(q_j, \lambda)$ with respect to $q_j(z_j)$ is given by

$$
\begin{aligned}
&\frac{\delta \mathcal{L}(q_j, \lambda)}{\delta q_j(z_j)} \\
&= \frac{\delta \mathcal{L}(q_j)}{\delta q_j(z_j)} - \lambda \frac{\delta}{\delta q_j(z_j)} \left(\int q_j(z_j) dz_j - 1 \right) \\
&= \frac{\delta \mathcal{L}(q_j)}{\delta q_j(z_j)} - \lambda \\
&= \frac{\delta}{\delta q_j(z_j)} \left(\mathbb{E}_{q_j(z_j)} \left[\mathbb{E}_{q_{-j}(z_{-j})} \left[\log p(z, x) \right] \right] - \mathbb{E}_{q_j(z_j)} [\log q_j(z_j)] \right) - \lambda \\
&= \frac{\delta}{\delta q_j(z_j)} \left(\int q_j(z_j) \mathbb{E}_{q_{-j}(z_{-j})} \left[\log p(z, x) \right] dz_j - \int q_j(z_j) \log q_j(z_j) dz_j \right) - \lambda \\
&= \mathbb{E}_{q_{-j}(z_{-j})} \left[\log p(z, x) \right] - \log q_j(z_j) - 1 - \lambda.
\end{aligned}
$$

Setting this derivative to zero and rearranging terms yields

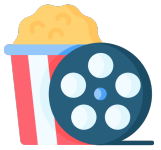
$$\log q_j^*(z_j) = \mathbb{E}_{q_{-j}(z_{-j})} \left[\log p(z, x) \right] + \text{const},$$

where the constant absorbs $-1 - \lambda$ and ensures that $q_j^*(z_j)$ integrates to 1.

Finally, taking the exponential of both sides gives

$$q_j^*(z_j) \propto \exp\left(\mathbb{E}_{q_{-j}(z_{-j})}\left[\log p(z, x)\right]\right).$$

This optimal factor maximizes the ELBO with respect to $q_j(z_j)$ while keeping the other factors fixed. It can therefore be used in a coordinate ascent algorithm to iteratively update each factor until convergence, leading to the **coordinate ascent variational inference** algorithm.



Example: Matrix factorization

We assume we have observed a matrix of movie ratings $R \in \mathbb{R}^{N \times M}$, where R_{ij} is the rating given by user i to movie j, N is the number of users and M is the number of movies. Most entries of R are missing and we want to predict them.

We model the ratings R_{ij} using a latent variable model with latent user and movie factors, such that

$$\begin{aligned} u_i &\sim \mathcal{N}(0, \sigma_u^2 I), \\ v_j &\sim \mathcal{N}(0, \sigma_v^2 I), \\ R_{ij} | u_i, v_j &\sim \mathcal{N}(u_i^T v_j, \sigma^2), \end{aligned}$$

where $u_i \in \mathbb{R}^K$ and $v_j \in \mathbb{R}^K$ are latent factors for user i and movie j, respectively, and σ^2 is the observation noise variance.

We want to approximate the posterior distribution $p(\{u_i\}, \{v_j\} | R)$ over the latent factors given the observed ratings.

We choose a mean-field variational family where the latent factors are independent under the variational distribution, i.e.,

$$q(\{u_i\}, \{v_j\}) = \prod_{i=1}^{N} q(u_i) \prod_{j=1}^{M} q(v_j).$$

Due to Gaussian-Gaussian conjugacy, each optimal factor will be Gaussian:

$$q(u_i) = \mathcal{N}(u_i | \mu_{u_i}, \Sigma_{u_i}),$$
$$q(v_j) = \mathcal{N}(v_j | \mu_{v_j}, \Sigma_{v_j}).$$

Let $q_{-u_i}(\{u_{i'}\}, \{v_j\})$ denote the product of all factors except $q(u_i)$. The coordinate ascent update is

$$q^*(u_i) \propto \exp\left(\mathbb{E}_{q_{-u_i}}\left[\log p(\{u_i\}, \{v_j\}, R)\right]\right).$$

The joint log-probability factorizes as

$$\begin{aligned}
\log p(u_i, v_j, R) &= \sum_{i=1}^{N} \log p(u_i) + \sum_{j=1}^{M} \log p(v_j) + \sum_{(i,j)\in\mathcal{O}} \log p(R_{ij}|u_i, v_j) \\
&= \log p(u_i) + \sum_{j\in\mathcal{O}_i} \log p(R_{ij}|u_i, v_j) + \text{const w.r.t. } u_i \\
&= -\frac{1}{2\sigma_u^2}|u_i|^2 - \frac{1}{2\sigma^2}\sum_{j\in\mathcal{O}_i}(R_{ij} - u_i^T v_j)^2 + \text{const}
\end{aligned}$$

where $\mathcal{O}$ denotes the set of observed entries (i, j) and $\mathcal{O}_i = \{j : (i, j) \in \mathcal{O}\}$.

Taking the expectation over q_{-u_i}, we have

$$\mathbb{E}_{q_{-u_i}}[\log p(u_i, v_j, R)] = -\frac{1}{2\sigma_u^2}|u_i|^2 - \frac{1}{2\sigma^2}\sum_{j \in \mathcal{O}_i} \mathbb{E}_{q(v_j)}\left[(R_{ij} - u_i^T v_j)^2\right] + \text{const.}$$

Expanding the squared term inside the expectation yields

$$\mathbb{E}_{q(v_j)}\left[(R_{ij} - u_i^T v_j)^2\right] = R_{ij}^2 - 2R_{ij}u_i^T \mu_{v_j} + u_i^T(\mu_{v_j}\mu_{v_j}^T + \Sigma_{v_j})u_i,$$

where μ_{v_j} and Σ_{v_j} are the mean and covariance of $q(v_j)$.

Substituting back and absorbing R_{ij}^2, we get

$$\begin{aligned}
&\mathbb{E}_{q_{-u_i}}[\log p(u_i, v_j, R)] \\
&= -\frac{1}{2\sigma_u^2}|u_i|^2 - \frac{1}{2\sigma^2}\sum_{j\in\mathcal{O}_i}\left[-2R_{ij}u_i^T\mu_{v_j} + u_i^T(\mu_{v_j}\mu_{v_j}^T + \Sigma_{v_j})u_i\right] + \text{const.}
\end{aligned}$$

Collecting terms quadratic and linear in u_i, we have

$$\begin{aligned}
&\mathbb{E}_{q_{-u_i}}[\log p(u_i, v_j, R)] \\
&= -\frac{1}{2}u_i^T\left[\frac{1}{\sigma_u^2}I + \frac{1}{\sigma^2}\sum_{j\in\mathcal{O}_i}(\mu_{v_j}\mu_{v_j}^T + \Sigma_{v_j})\right]u_i + u_i^T\left[\frac{1}{\sigma^2}\sum_{j\in\mathcal{O}_i}R_{ij}\mu_{v_j}\right] + \text{const.}
\end{aligned}$$

We can identify this expression as the log of a Gaussian distribution in u_i with covariance and mean given by

$$\Sigma_{u_i} = \left[\frac{1}{\sigma_u^2} I + \frac{1}{\sigma^2} \sum_{j \in \mathcal{O}_i} (\mu_{v_j} \mu_{v_j}^T + \Sigma_{v_j}) \right]^{-1},$$

$$\mu_{u_i} = \Sigma_{u_i} \left[\frac{1}{\sigma^2} \sum_{j \in \mathcal{O}_i} R_{ij} \mu_{v_j} \right].$$

By symmetry, the updates for $q(v_j)$ are similar and defined as

$$\Sigma_{v_j} = \left[\frac{1}{\sigma_v^2} I + \frac{1}{\sigma^2} \sum_{i \in \mathcal{O}_j} (\mu_{u_i} \mu_{u_i}^T + \Sigma_{u_i}) \right]^{-1},$$

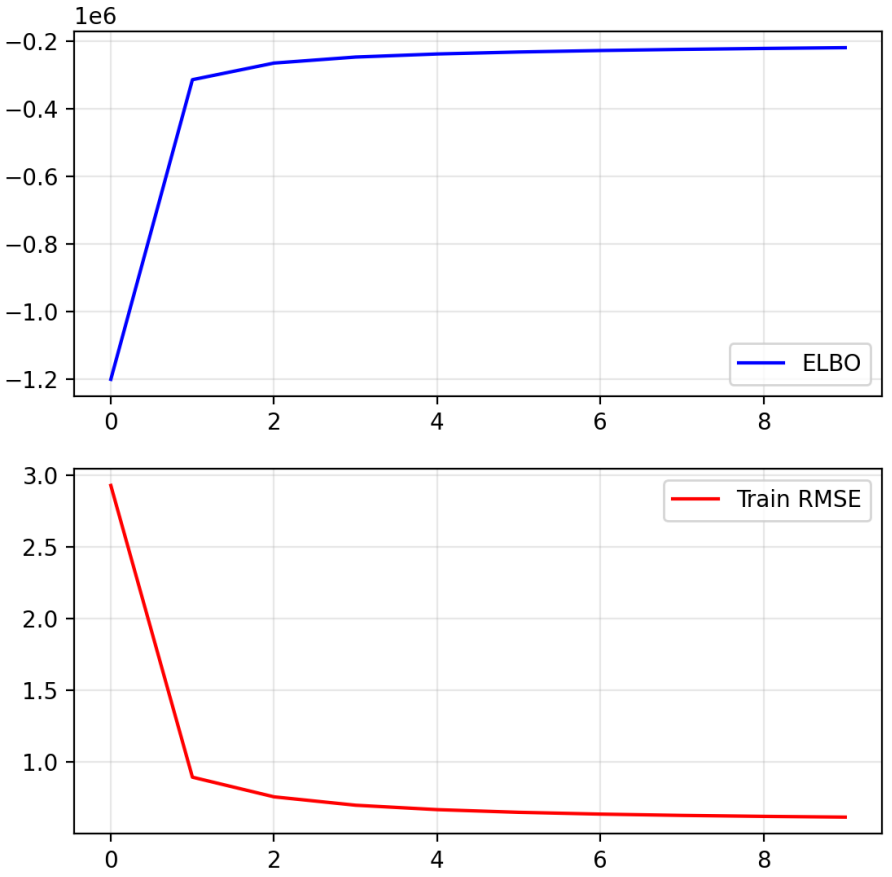
$$\mu_{v_j} = \Sigma_{v_j} \left[\frac{1}{\sigma^2} \sum_{i \in \mathcal{O}_j} R_{ij} \mu_{u_i} \right],$$

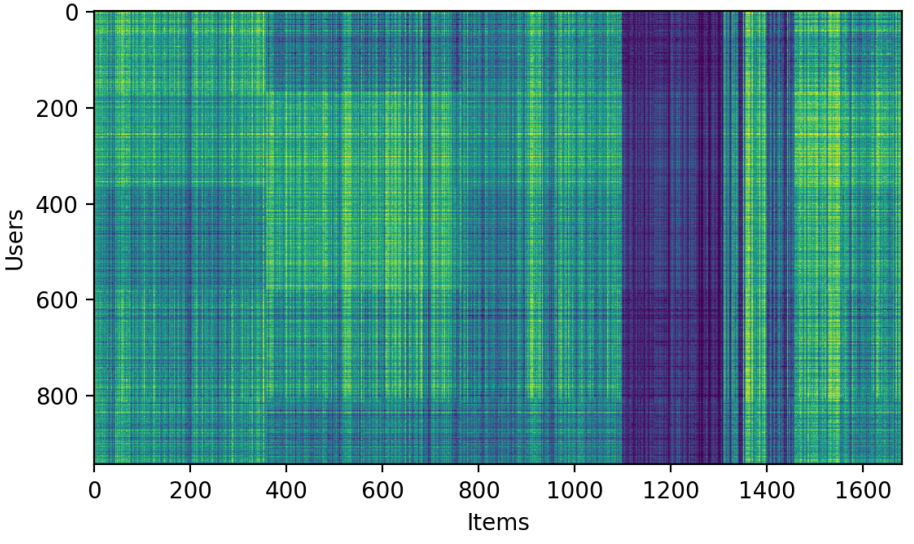
where $\mathcal{O}_j = \{i : (i, j) \in \mathcal{O}\}$.

In summary, CAVI for matrix factorization can be implemented as follows:

- Initialize the variational parameters $\{\mu_{u_i}, \Sigma_{u_i}\}$ and $\{\mu_{v_j}, \Sigma_{v_j}\}$ randomly.
- Repeat until convergence:
  - For each user $i = 1, \ldots, N$, update Σ_{u_i} and μ_{u_i} using the derived formulas.
  - For each movie $j = 1, \ldots, M$, update Σ_{v_j} and μ_{v_j} using the derived formulas.

(Step-by-step code example in `nb08-cavi.ipynb`.)


1e6
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
0
2
4
6
8
ELBO
3.0
2.5
2.0
1.5
1.0
0
2
4
6
8
Train RMSE



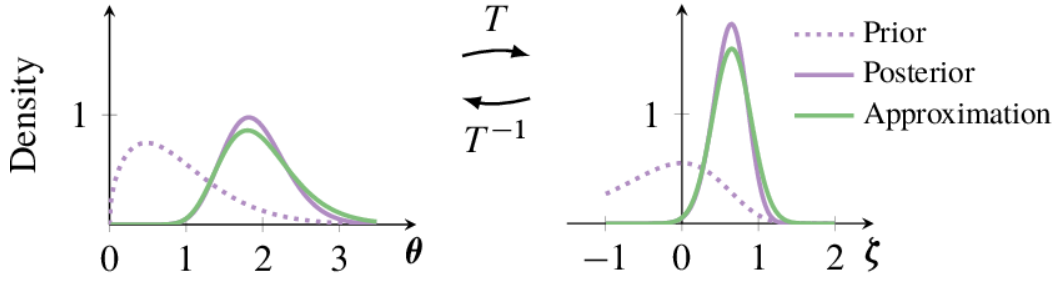
Predicted ratings matrix (using biclustering for visualization).

Strengths and weaknesses

- CAVI provides a general framework for variational inference under the mean-field assumption.
- CAVI requires **closed-form expressions for the coordinate updates**. This is only possible for certain models (e.g., conditionally conjugate models). Their derivation can be **tedious** and **error-prone**.
- CAVI is restricted to mean-field variational families, which may be too simplistic to capture the true posterior distribution.

Automatic differentiation variational inference (ADVI)

ADVI is a **black-box variational inference** method that overcomes the limitations of CAVI by leveraging automatic differentiation and stochastic optimization.



(a) Latent variable space **(b)** Real coordinate space

Since latent variables may be constrained (e.g., positive or bounded), the first step in ADVI is to transform the latent variables z to an unconstrained space using a differentiable bijection

$$T : \text{supp}(z) \to \mathbb{R}^m,$$

such that $\zeta = T(z)$ are unconstrained variables in $\mathbb{R}^m$.

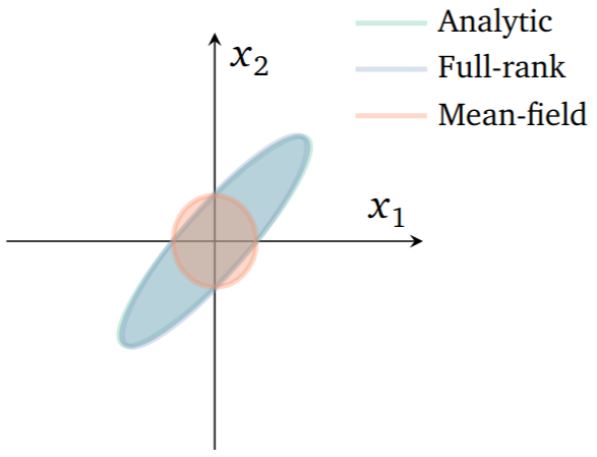
By the change of variables theorem, the joint distribution in the unconstrained space is given by

$$p(\zeta, x) = p(T^{-1}(\zeta), x) \left|\det J_{T^{-1}}(\zeta)\right|,$$

where $J_{T^{-1}}(\zeta) = \frac{dT^{-1}(\zeta)}{d\zeta}$ is the Jacobian of the inverse transformation.

—

Credits: Kucukelbir et al, 2016.



We then define a variational family $q(\zeta;\phi)$ in the unconstrained space, parameterized by variational parameters ϕ.

Common choices include:

- Mean-field Gaussian: $q(\zeta;\phi) = \mathcal{N}(\zeta|\mu, \mathrm{diag}(\sigma^2)) = \prod_{i=1}^{m} \mathcal{N}(\zeta_i|\mu_i, \sigma_i^2)$ with $\phi = (\mu \in \mathbb{R}^m, \sigma \in \mathbb{R}^m)$;
- Full-rank Gaussian: $q(\zeta;\phi) = \mathcal{N}(\zeta|\mu, \Sigma = LL^T)$ with $\phi = (\mu \in \mathbb{R}^m, L \in \mathbb{R}^{m(m+1)/2})$, where L is the Cholesky factor of Σ.

—

Credits: Kucukelbir et al, 2016.

Finally, ADVI proceeds by maximizing the ELBO with respect to the variational parameters ϕ using stochastic gradient ascent. The ELBO in the unconstrained space is given by

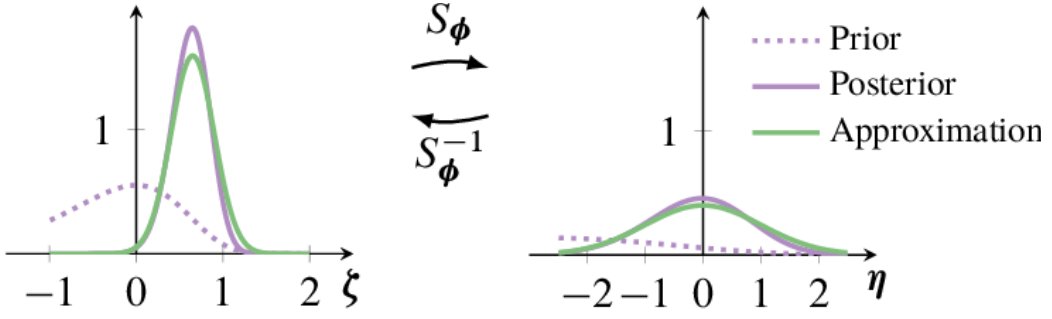
$$\mathcal{L}(\phi) = \mathbb{E}_{q(\zeta;\phi)} \left[\log p(T^{-1}(\zeta), x) + \log \left| \det J_{T^{-1}}(\zeta) \right| - \log q(\zeta; \phi) \right].$$

To compute gradients of the ELBO with respect to ϕ, we can use the **reparameterization trick** to express the expectation over $q(\zeta;\phi)$ in terms of a fixed distribution independent of ϕ.

If $\zeta = g(\epsilon;\phi)$ with $\epsilon \sim p(\epsilon)$, then

$$\mathcal{L}(\phi) = \mathbb{E}_{p(\epsilon)}\left[\log p(T^{-1}(g(\epsilon;\phi)), x) + \log\left|\det J_{T^{-1}}(g(\epsilon;\phi))\right| - \log q(g(\epsilon;\phi);\phi)\right].$$

In particular, for a Gaussian variational family, we can write $\zeta = \mu + L\epsilon$ where L is the Cholesky factor of Σ and $\epsilon \sim \mathcal{N}(0, I)$.



(a) Real coordinate space **(b)** Standardized space

—

Credits: Kucukelbir et al, 2016.

In this form, we can compute unbiased estimates of the gradient of the ELBO with respect to ϕ using Monte Carlo sampling and automatic differentiation,

$$\begin{aligned}&\nabla_\phi \mathcal{L}(\phi) \\ &\approx \frac{1}{B}\sum_{i=1}^{B} \nabla_\phi \left[\log p(T^{-1}(g(\epsilon_i;\phi)), x) + \log\left|\det J_{T^{-1}}(g(\epsilon_i;\phi))\right| - \log q(g(\epsilon_i;\phi);\phi)\right]\end{aligned}$$

where $\epsilon_i \sim p(\epsilon)$.

Finally, these gradient estimates can be used in a stochastic optimization algorithm to maximize the ELBO and learn the variational parameters ϕ.

Example: The Pitcher's example (from Lecture 1)

(Step-by-step code example in `nb08-advi.ipynb`.)

Strengths and weaknesses

- ADVI does not require closed-form updates.
- ADVI does not require mean-field assumptions.
- ADVI relies on automatic differentiation and stochastic optimization, which avoids tedious derivations.
- ADVI is limited by the choice of variational family (e.g., Gaussian), which may not capture complex posterior distributions.

Amortized variational inference

In all previous examples, we considered variational inference for a single observation x. In practice, we often have a dataset of N observations $\mathbf{d} = \{x_1, \ldots, x_N\}$ and want to perform inference for each observation.

A naive approach would be to run variational inference separately for each observation, which can be computationally expensive.

Amortized variational inference

Amortized variational inference addresses this issue by learning a shared inference model that maps each observation x_i to its corresponding variational distribution $q(z|x_i)$.

Mathematically, this can be done by parameterizing $q(z|x_i)$ as

$$q(z|f(x_i;\varphi)),$$

where $f(x;\varphi)$ is a function with shared parameters φ that takes an observation x as input and outputs the parameters of the variational distribution for the latent variables z.

The parameters φ of the inference model could be learned by maximizing the ELBO averaged over the dataset.

However, because we want to amortize inference, we can rewrite the objective as the minimization of the expected forward KL divergence over the observations,

$$\varphi^* = \arg\min_{\varphi} \mathbb{E}_{p(x)} \left[\mathrm{KL}(p(z|x)||q(z|f(x;\varphi))) \right].$$

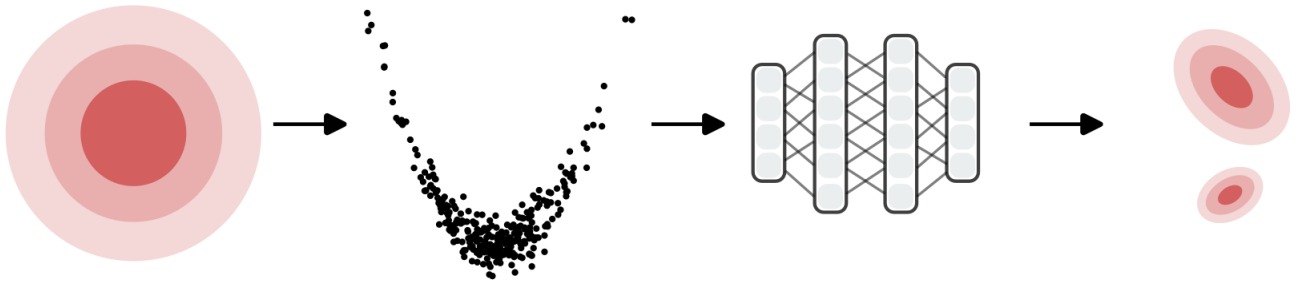
While the forward KL divergence was previously inaccessible due to the lack of samples from the true posterior, in the amortized setting we can use samples from the joint distribution $p(z, x)$ to estimate the expected KL divergence!

$$\begin{aligned}\mathbb{E}_{p(x)}\left[\mathrm{KL}(p(z|x)||q(z|f(x;\varphi)))\right] &= \mathbb{E}_{p(x)}\left[\mathbb{E}_{p(z|x)}\left[\log \frac{p(z|x)}{q(z|f(x;\varphi))}\right]\right] \\ &= \mathbb{E}_{p(x,z)}\left[\log \frac{p(z|x)}{q(z|f(x;\varphi))}\right] \\ &= -\mathbb{E}_{p(x,z)}\left[\log q(z|f(x;\varphi))\right] + \mathrm{const.}\end{aligned}$$

In other words, we can learn the inference model by maximizing the expected posterior log-density under the joint distribution!

Neural posterior estimation (NPE)

Neural posterior estimation is an amortized variational inference method where the inference model is parameterized using neural networks.



—

Credits: Adapted from Deistler et al, 2025.

The variational distribution can be defined in different ways, such as

- $q(z|f(x;\varphi))$, using a simple distribution (e.g., Gaussian) and a **neural network** $f(x;\varphi)$ that outputs the parameters of the variational distribution given an observation x,
- or as $q(z|f(x;\varphi);\phi)$ using a flexible density estimator, such as (conditional) **normalizing flows**, with parameters ϕ, and a neural network $f(x;\varphi)$ that outputs a sufficient statistic of x.

For training, we can generate samples from the joint distribution $p(z, x)$ by first sampling $z \sim p(z)$ from the prior and then sampling $x \sim p(x|z)$ from the likelihood.

The parameters φ (and ϕ if applicable) can then be learned by maximizing the expected posterior log-density using stochastic gradient ascent,

$$\varphi^*, \phi^* = \arg\max_{\varphi,\phi} \mathbb{E}_{p(x,z)} \left[\log q(z|f(x;\varphi);\phi)\right].$$

Strengths and weaknesses

- NPE leverages neural networks to learn flexible inference models that can capture complex posterior distributions.
- NPE only requires samples from the joint distribution $p(z, x)$, making it applicable to a large class of models.
- NPE amortizes inference across multiple observations, making inference as fast as a forward pass through the neural network. However, this comes at the cost of an upfront training phase.

Simulation-based inference

NPE is an example of simulation-based inference (SBI) methods, which are designed for scenarios where the likelihood $p(x|z)$ (resp. $p(x|\theta)$) is intractable but we can still generate samples from the joint distribution $p(z, x)$ (resp. $p(\theta, x)$) using a simulator.

SBI algorithms are a major evolution in Bayesian inference, enabling posterior inference in complex models where traditional methods fail, **without simplifying assumptions**.

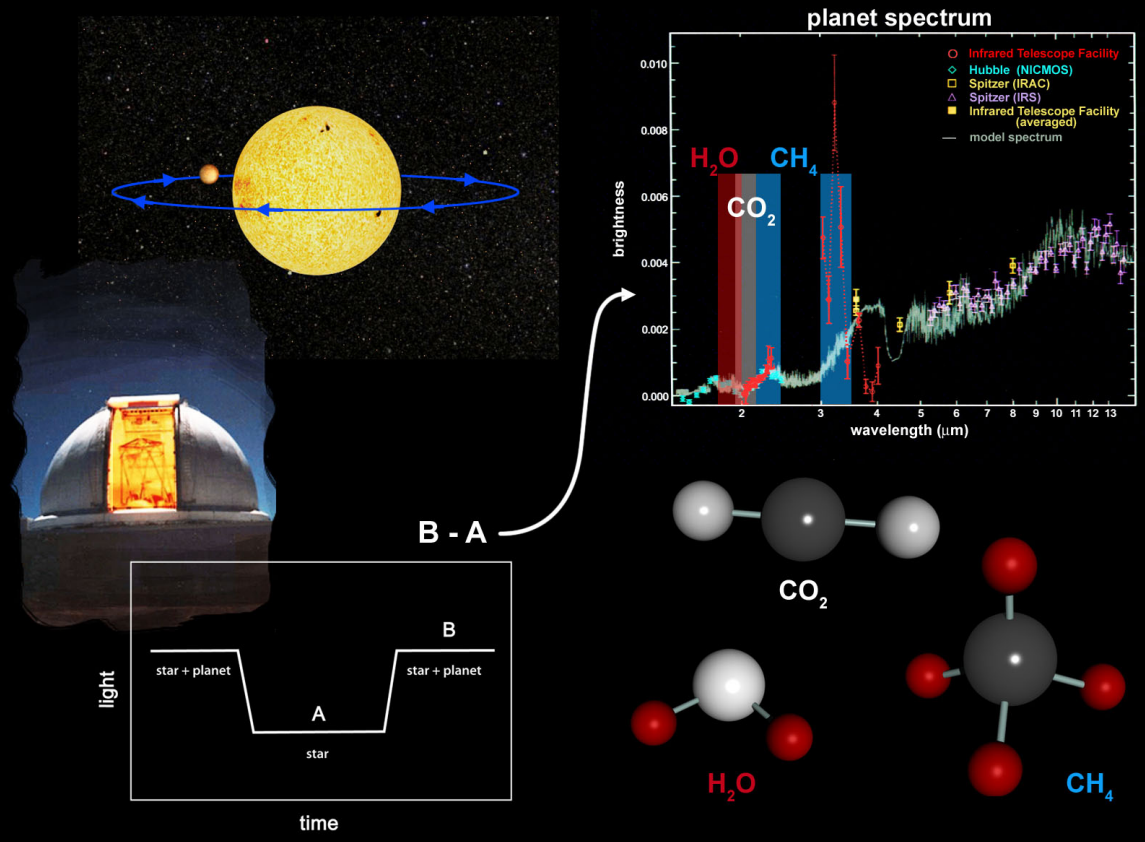
Alternative simulation-based inference algorithms include:

- Neural likelihood estimation (NLE): learns a surrogate likelihood model $q(x|\theta;\phi)$ using samples from $p(\theta, x)$, then performs inference using standard methods (e.g., MCMC, VI) with the surrogate likelihood.
- Neural ratio estimation (NRE): learns a likelihood-to-evidence ratio model $r(x, \theta; \phi) = \frac{p(x|\theta)}{p(x)}$ using samples from $p(\theta, x)$, then performs inference using standard methods with the learned ratio.
- Neural score estimation (NSE): learns a score function model $s(\theta|x;\phi) = \nabla_\theta \log p(\theta|x;\phi)$ using samples from $p(\theta, x)$, then performs inference using score-based methods.

These methods are not variational inference methods per se, but they share the same goal of approximating the posterior distribution in complex models.

Examples and case studies

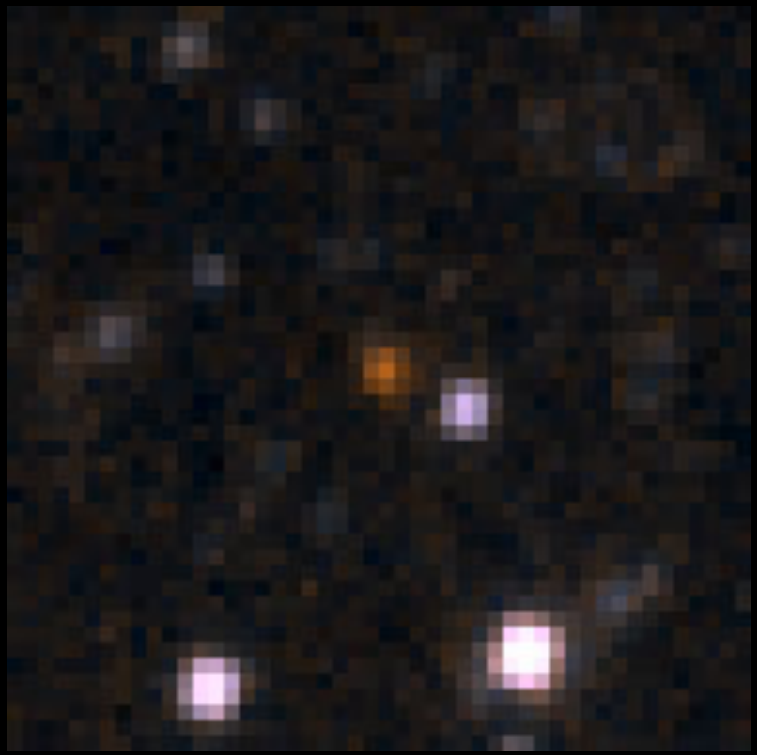
Exoplanet atmosphere characterization



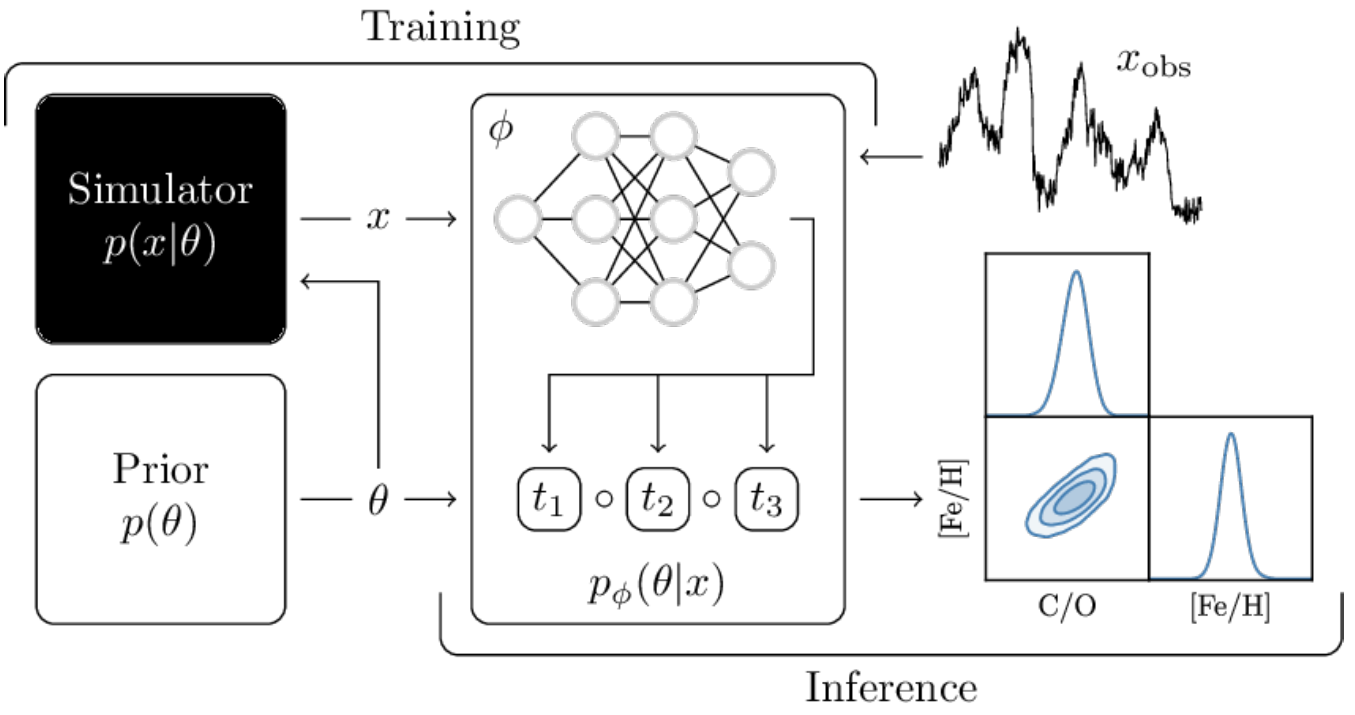
What are the atmospheres of exoplanets made of?
How do they form and evolve? Do they host life?

—

Credits: NSA/JPL-Caltech, 2010.



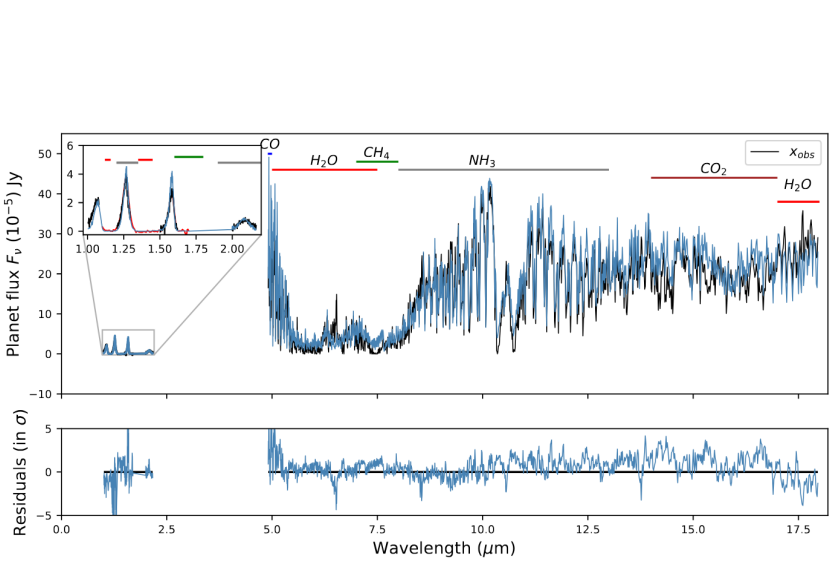
WISE 1738+2732, a brown dwarf 25 light-years away.

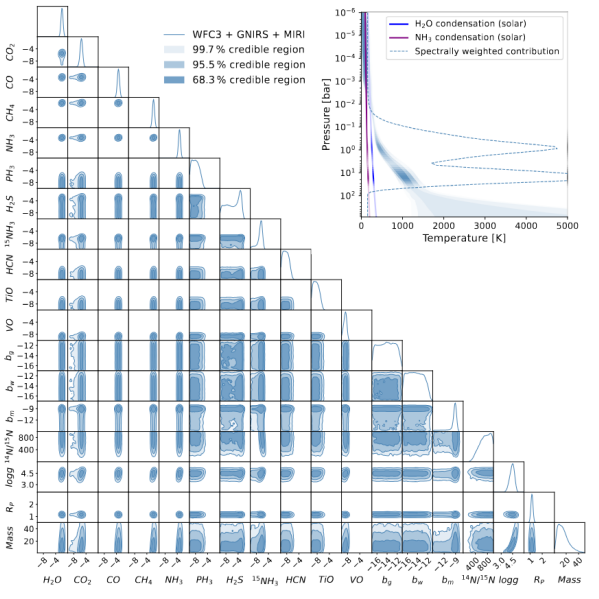


Using **Neural Posterior Estimation** (NPE), we approximate the posterior distribution $p(\theta|x)$ of atmospheric parameters θ with a **normalizing flow** trained on pairs (θ, x) simulated from a physical model of exoplanet atmospheres.

—

Credits: Vasist et al, 2023 (arXiv:2301.06575).



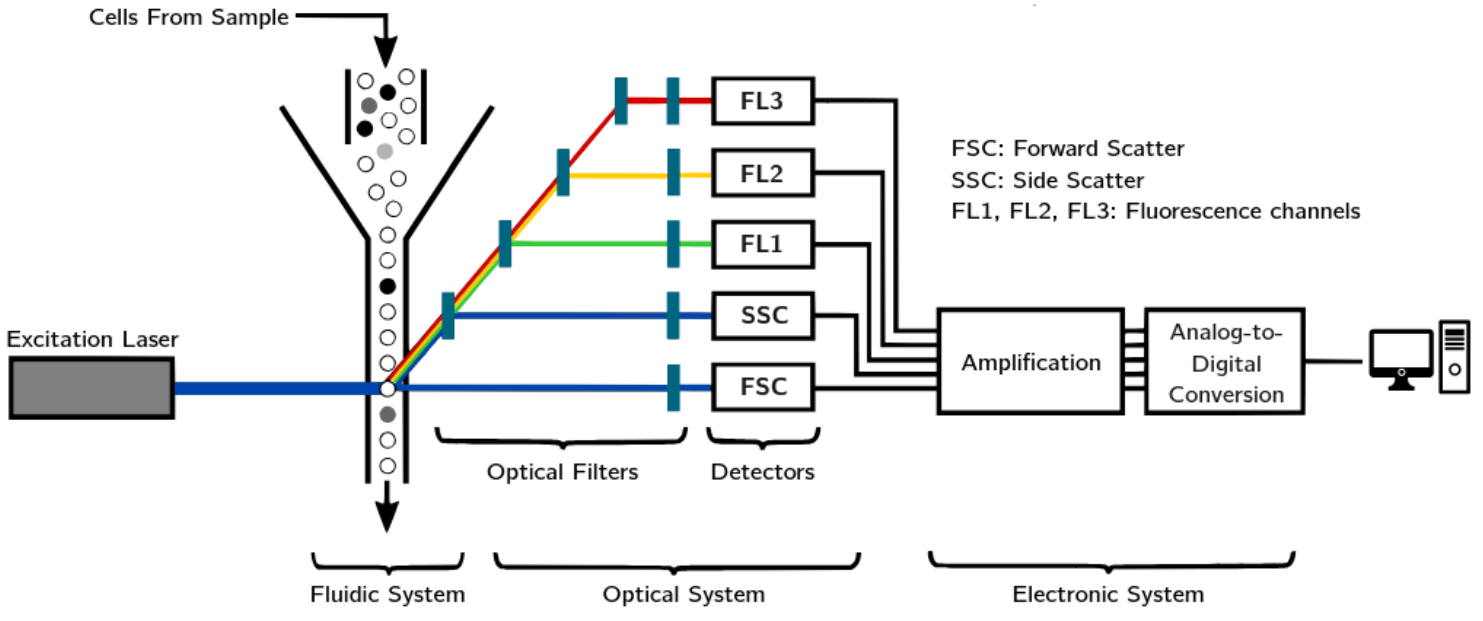


Panchromatic characterization of WISE 1738+2732 using JWST/MIRI.

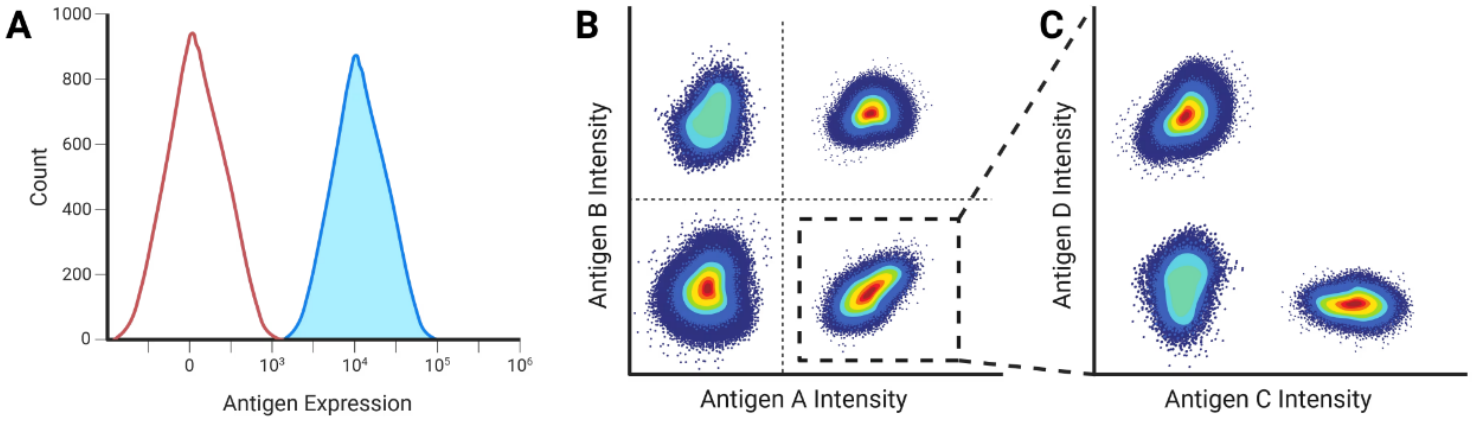
—

Credits: Vasist et al, 2025 (arXiv:2507.12264).

Representation learning for cytometry data



Flow cytometry is used to measure the chemical characteristics of cells as they flow in a fluid stream through a beam of light. It is used across biology and medicine for immune profiling, disease diagnosis, and drug development.



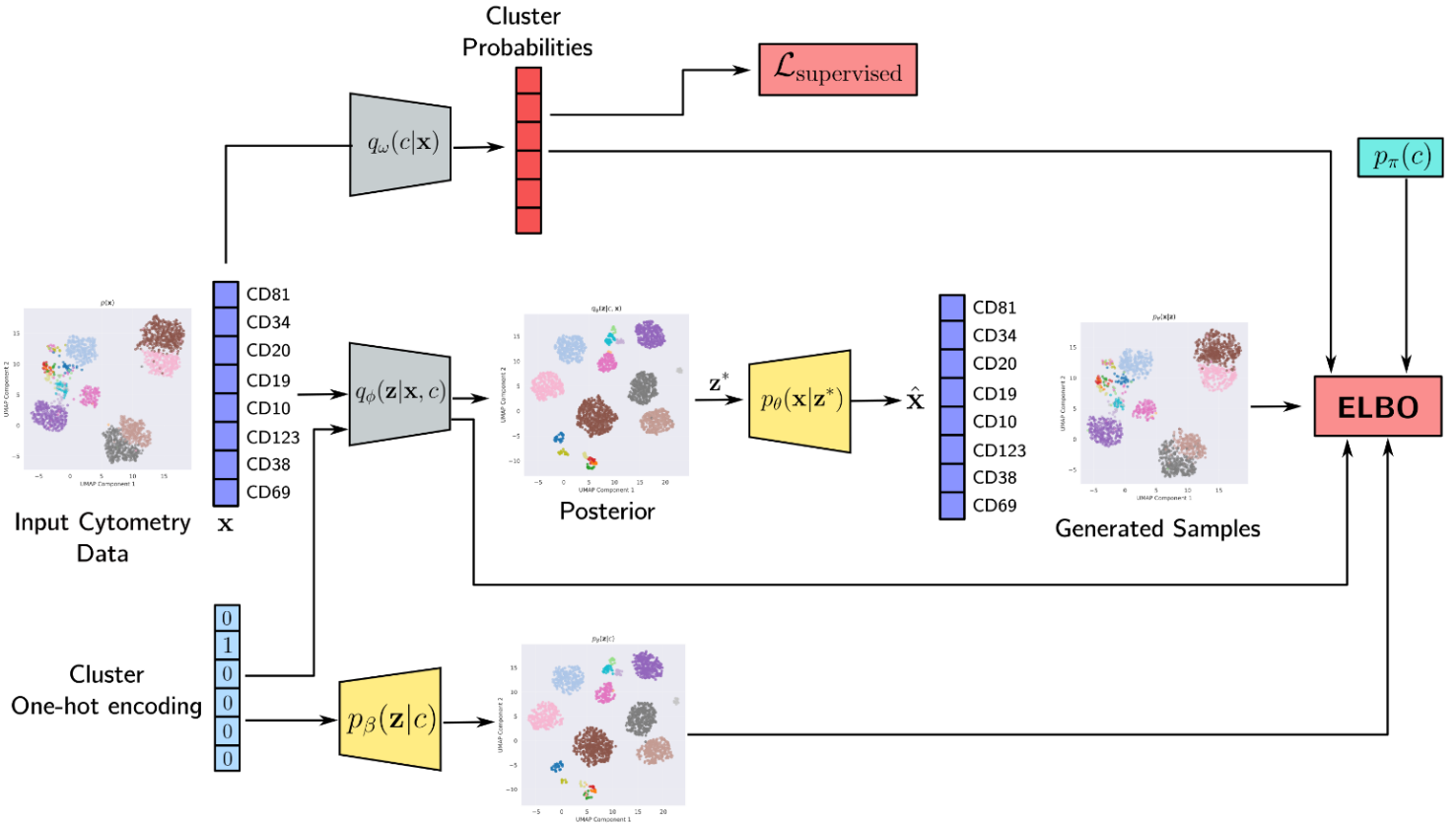
However, analyzing flow cytometry data can be challenging due to the high dimensionality and complexity of the data. Traditional methods often rely on manual gating, which is time-consuming and subjective.

To automate the classification of cells and the discovery of new cell populations, we consider a latent variable model of cell populations

$$
\begin{aligned}
p_{\theta,\beta,\pi}(x, z, c) &= p_\theta(x|z)p_\beta(z|c)p_\pi(c), \\
p_\pi(c) &= \text{Categorical}(c|\pi), \\
p_\beta(z|c) &= \mathcal{N}(z|\mu_\beta(c), \text{diag}(\sigma^2_\beta(c))), \\
p_\theta(x|z) &= \mathcal{N}(x|\mu_\theta(z), \text{diag}(\sigma^2_\theta(z))),
\end{aligned}
$$

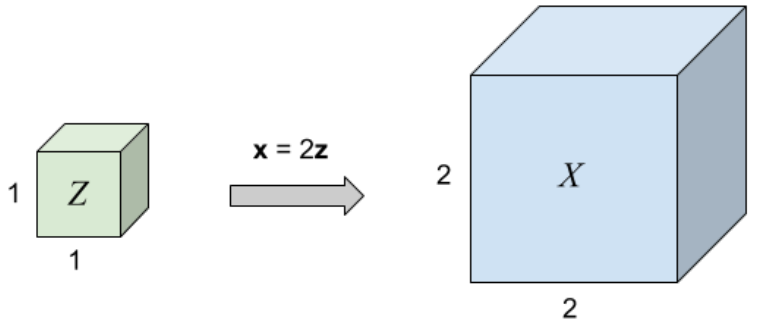
where x are the observed cell measurements, z are latent representations of cells, and c are discrete cell population memberships.

We use **deep neural networks** to parameterize the functions $\mu_\theta(z)$, $\sigma^2_\theta(z)$, $\mu_\beta(c)$, and $\sigma^2_\beta(c)$ and rely on **amortized variational inference** to learn the model parameters and infer the latent variables.


Cluster
Probabilities
$\mathcal{L}_{\text{supervised}}$
$q_\omega(c|\mathbf{x})$
$p_\pi(c)$
CD81
CD34
CD20
CD19
CD10
CD123
CD38
CD69
$q_\phi(\mathbf{z}|\mathbf{x}, c)$
$\mathbf{z}^*$
$p_\theta(\mathbf{x}|\mathbf{z}^*)$
$\hat{\mathbf{x}}$
ELBO
Input Cytometry
Data
$\mathbf{x}$
Posterior
Generated Samples
Cluster
One-hot encoding
0
1
0
0
0
0
$p_\beta(\mathbf{z}|c)$

The end.

Change of variables



Assume $p(\mathbf{z})$ is a uniformly distributed unit cube in $\mathbb{R}^3$ and $\mathbf{x} = f(\mathbf{z}) = 2\mathbf{z}$. Since the total probability mass must be conserved,

$$p(\mathbf{x}) = p(\mathbf{x} = f(\mathbf{z})) = p(\mathbf{z})\frac{V_{\mathbf{z}}}{V_{\mathbf{x}}} = p(\mathbf{z})\frac{1}{8},$$

where $\frac{1}{8} = \left|\det \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}\right|^{-1}$ represents the inverse determinant of the Jacobian of the linear transformation f.

What if f is non-linear?

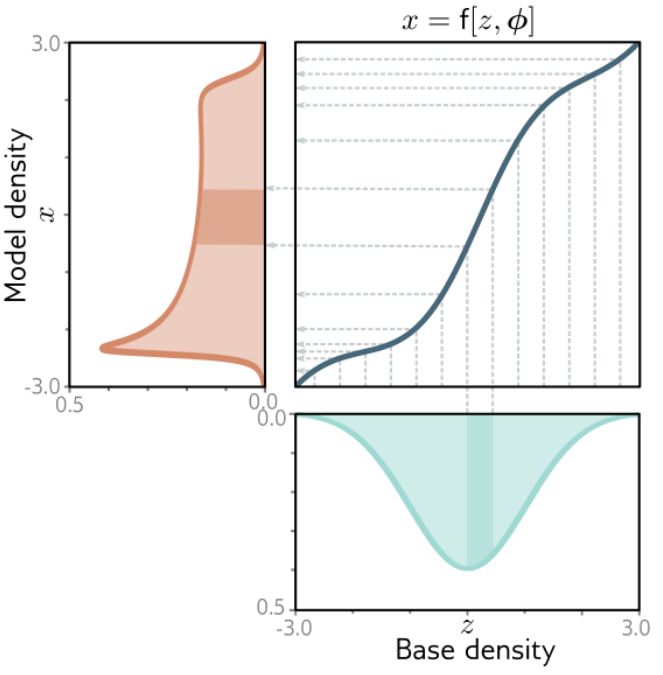


Figure 16.2 Transforming distributions. The base density (cyan, bottom) passes through a function (blue curve, top right) to create the model density (orange, left). Consider dividing the base density into equal intervals (gray vertical lines). The probability mass between adjacent lines must remain the same after transformation. The cyan-shaded region passes through a part of the function where the gradient is larger than one, so this region is stretched. Consequently, the height of the orange-shaded region must be lower so that it retains the same area as the cyan-shaded region. In other places (e.g., $z = -2$), the gradient is less than one, and the model density increases relative to the base density.

—

Image credits: Simon J.D. Prince, Understanding Deep Learning, 2023.

Change of variables theorem

If f is non-linear,

- the Jacobian $J_f(\mathbf{z})$ of $\mathbf{x} = f(\mathbf{z})$ represents the infinitesimal linear transformation in the neighborhood of $\mathbf{z}$;
- if the function is a bijective map, then the mass must be conserved locally.

Therefore, the local change of density yields

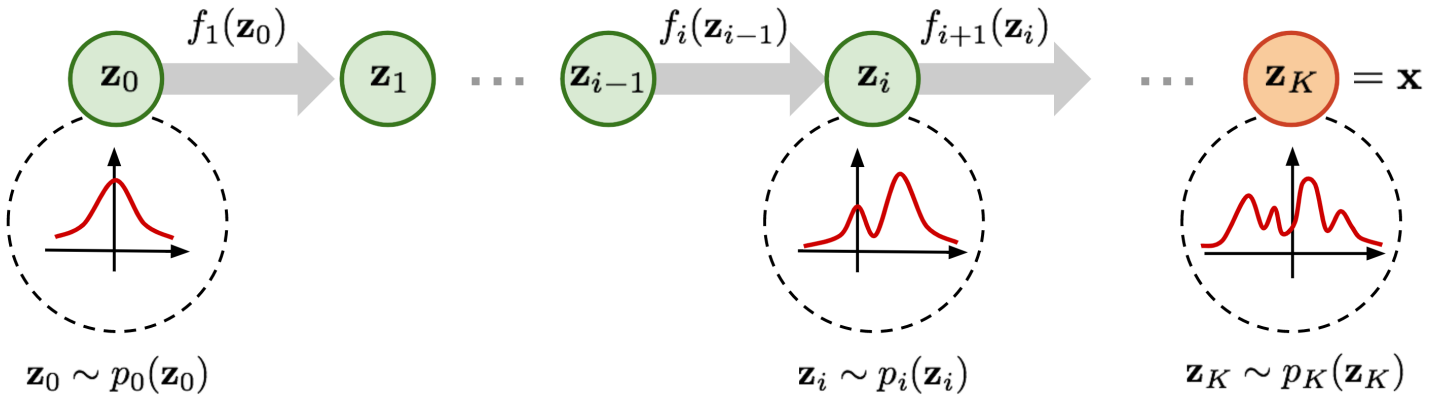
$$p(\mathbf{x} = f(\mathbf{z})) = p(\mathbf{z}) \left|\det J_f(\mathbf{z})\right|^{-1}.$$

Similarly, for $g = f^{-1}$, we have

$$p(\mathbf{x}) = p(\mathbf{z} = g(\mathbf{x})) \left|\det J_g(\mathbf{x})\right|.$$

Normalizing flows

A normalizing flow is a sequence of invertible transformations f_k that map a simple distribution p_0 to a more complex distribution p_K. Each transformation f_k is parameterized by an invertible neural network.



By the change of variables formula, the log-likelihood of a sample x is given by

$$\log p(\mathbf{x}) = \log p(\mathbf{z}_0) - \sum_{k=1}^{K} \log \left| \det J_{f_k}(\mathbf{z}_{k-1}) \right| .$$