Variational Inference

when you want to infer, but faster

Giovanni Santacatterina

19/05/2023

1. Intro to VI

From main goal…

$$\mathbf{x} = x_{1:n}$$ observations

$$\mathbf{z} = z_{1:m}$$ latent variables

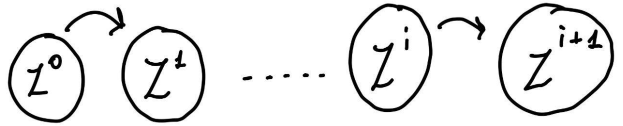
$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})} \longrightarrow p(\mathbf{x}) = \int p(\mathbf{z}, \mathbf{x}) dz$$

evidence

… to main problem

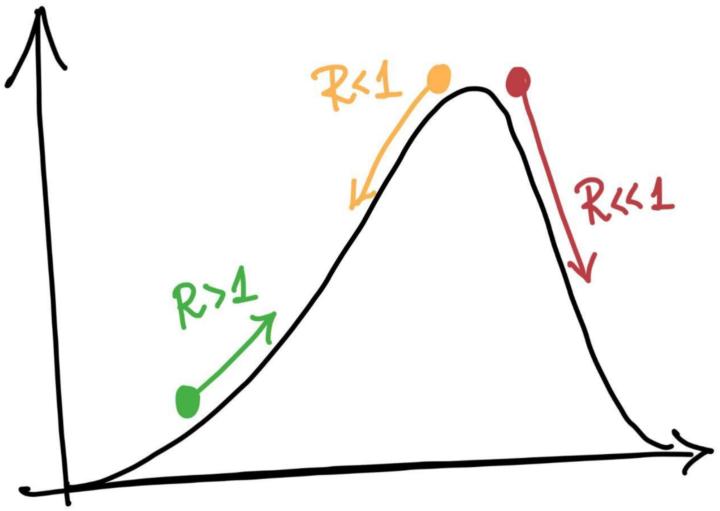
Possible solutions

In MCMC you construct an ergodic Markov chain on **z**



Get rid of the evidence and iterate!

$$R = \frac{p(\mathbf{z}^{t+1} \mid \mathbf{x})}{p(\mathbf{z}^{t} \mid \mathbf{x})}$$



VI first steps

Let’s approximate the posterior

But we need a measure of comparison first

$$\mathrm{KL}(q \,||\, p) = \int q(x) \log\left(\frac{q(x)}{p(x)}\right) dx$$

VI is an optimization problem

$$q^*(\mathbf{z}) = \underset{q(\mathbf{z}) \in \mathcal{D}}{\mathrm{argmin}} \mathrm{KL}(q(\mathbf{z}) \,||\, p(\mathbf{z} \mid \mathbf{x}))$$

From KL to ELBO

$$\mathrm{KL}(q(\mathbf{z}) \,||\, p(\mathbf{z},\mathbf{x})) = \mathbb{E}[\log q(\mathbf{z})] - \mathbb{E}[\log p(\mathbf{z} \mid \mathbf{x})]$$
$$= \mathbb{E}[\log q(\mathbf{z})] - \mathbb{E}[\log p(\mathbf{z},\mathbf{x})] + \log p(\mathbf{x})$$

We did not actually solved anything

Or did we?

$$\mathrm{ELBO}(q) = \mathbb{E}[\log p(\mathbf{z},\mathbf{x})] - \mathbb{E}[\log q(\mathbf{z})]$$

Nomen Omen! ELBO = Evidence Lower Bound

Role of the ELBO

$$\begin{aligned}\mathrm{ELBO}(q) &= \mathbb{E}[\log p(\mathbf{z})] + \mathbb{E}[\log p(\mathbf{x} \mid \mathbf{z})] - \mathbb{E}[\log q(\mathbf{z})] \\ &= \boxed{\mathbb{E}[\log p(\mathbf{x} \mid \mathbf{z})]} - \boxed{\mathrm{KL}(q(\mathbf{z}) \mid\mid p(\mathbf{z}))}\end{aligned}$$

Look at the two terms. Do they look familiar?

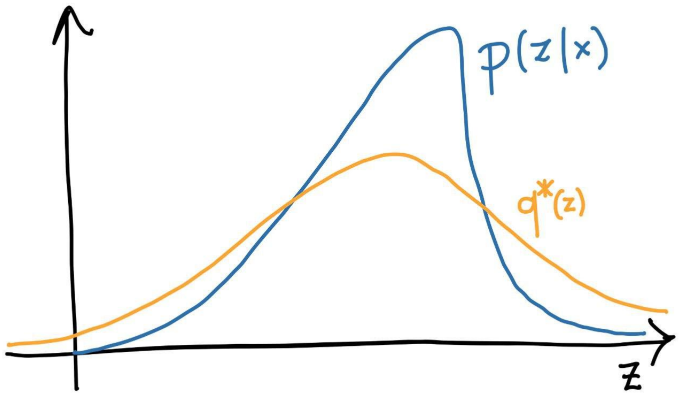
Here comes the typical bayesian “battle”

Prior vs. Likelihood

Variational family

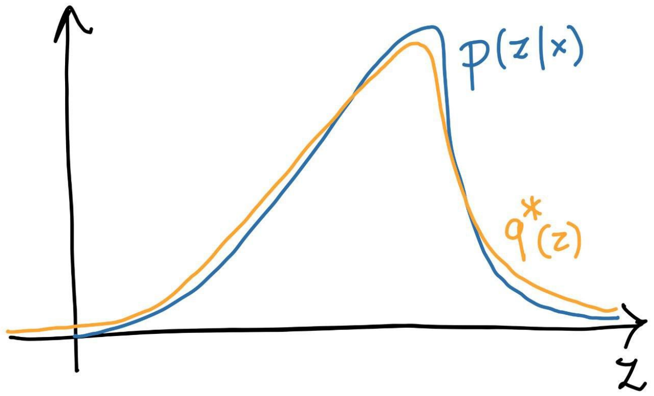
How good will be our approximation?

It will depend on $\mathcal{D}$



increase complexity

$\xrightarrow{\qquad\qquad}$



But how do you define $\mathcal{D}$

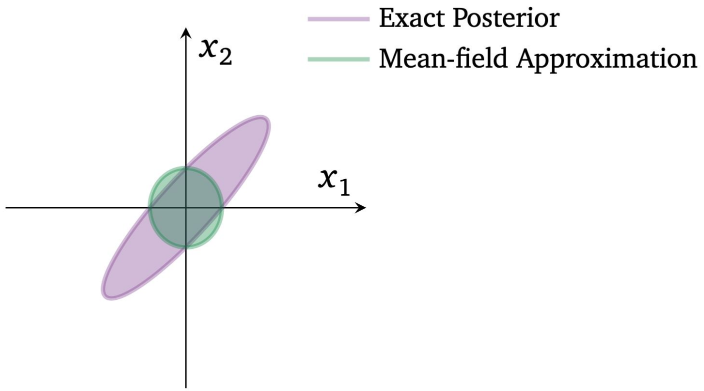
Mean-field Variational Family

Easy, consider the latent variables as mutually independent

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

Powerful, yet with some limitations

- correlation between latent variables is lost
- marginal variances are underestimated



Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe. "Variational inference: A review for statisticians." *Journal of the American statistical Association* 112.518 (2017): 859-877.

2.VI algorithms

Coordinate Ascent

Main idea

- initialize the variational factors
- until convergence of the ELBO repeat
    - fix all factors but one and optimize the latter
    - do it for all factors

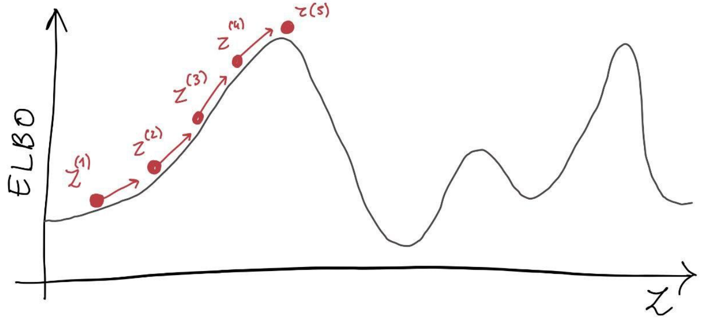
$$\begin{aligned}\log q_j^*(z_j) &= \mathbb{E}_{i\neq j}[\log p(\mathbf{x},\mathbf{z})] + const. \\ &\propto \mathbb{E}_{i\neq j}[\log p(\mathbf{x},\mathbf{z})]\end{aligned}$$

It has one main issue: it scales BADLY

From: Bishop, Christopher M., and Nasser M. Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. No. 4. New York: springer, 2006.

Stochastic Variational Inference

1. Substitute coordinate ascent with gradient-based optimizations

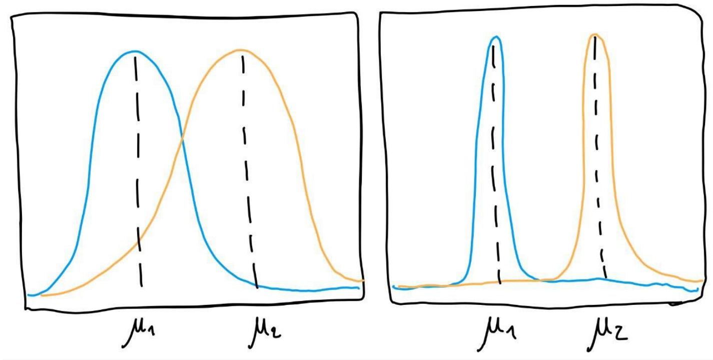


$$\mathbf{z}^{(t+1)} = \mathbf{z}^{(t)} + \epsilon \nabla_{\mathbf{z}} \text{ELBO}$$

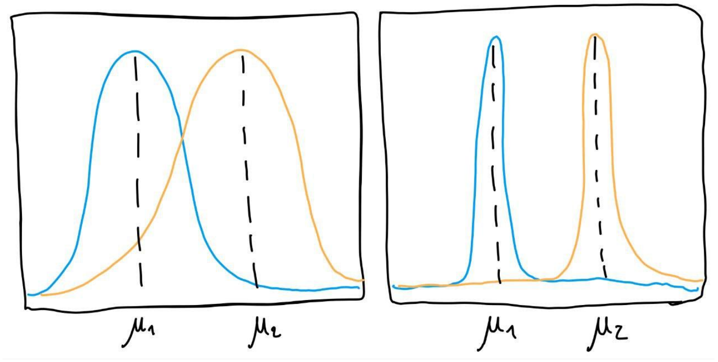
where ϵ represents the learning rate

2. You can scale up quite easily

3. Maybe the standard gradient is not the smartest choice



Natural gradient



Gradient based update come from

$$x^{(t+1)} = \arg\min_x f(x^{(t)}) + \nabla f(x^{(t)})(x - x^{(t)}) + \frac{1}{2}\left\|x - x^{(t)}\right\|_2^2$$

choose a better norm/distance (i.e. KL divergence)

$$\mathrm{KL}(p(x \mid z) \,||\, p(x \mid z^{(t)})) \approx F$$
$$F = \mathbb{E}_{x\sim p}[(\nabla_z \log p(x \mid z))(\nabla_z \log p(x \mid z))^T]$$

$$\mathbf{z}^{(t+1)} = \mathbf{z}^{(t)} - \epsilon F^{-1} \nabla_{\mathbf{z}} \mathrm{ELBO}$$

SVI and Pyro

SVI can used in Pyro, a probabilistic programming language

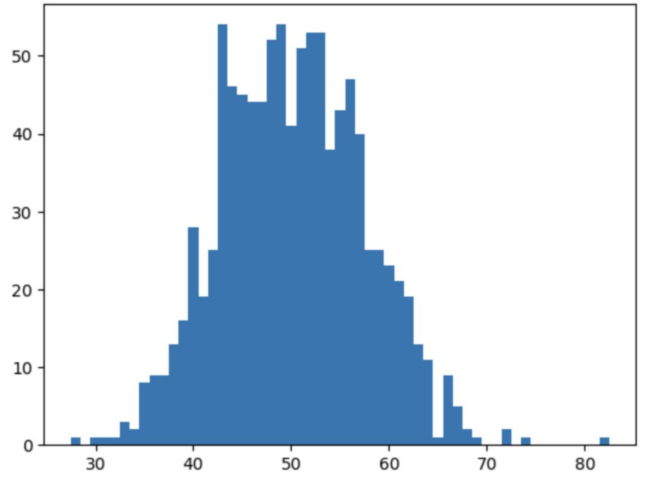
PPL are languages where probabilistic models are the main protagonists.

```
[2]: def poisson_sampling_model(n, _lambda):
         with pyro.plate("data", n):
             samples = pyro.sample("samples", dist.Poisson(_lambda))
         return samples

     def normal_sampling_model(n, mu, sigma):
         with pyro.plate("data", n):
             samples = pyro.sample("samples", dist.Normal(mu, sigma))
         return samples
```

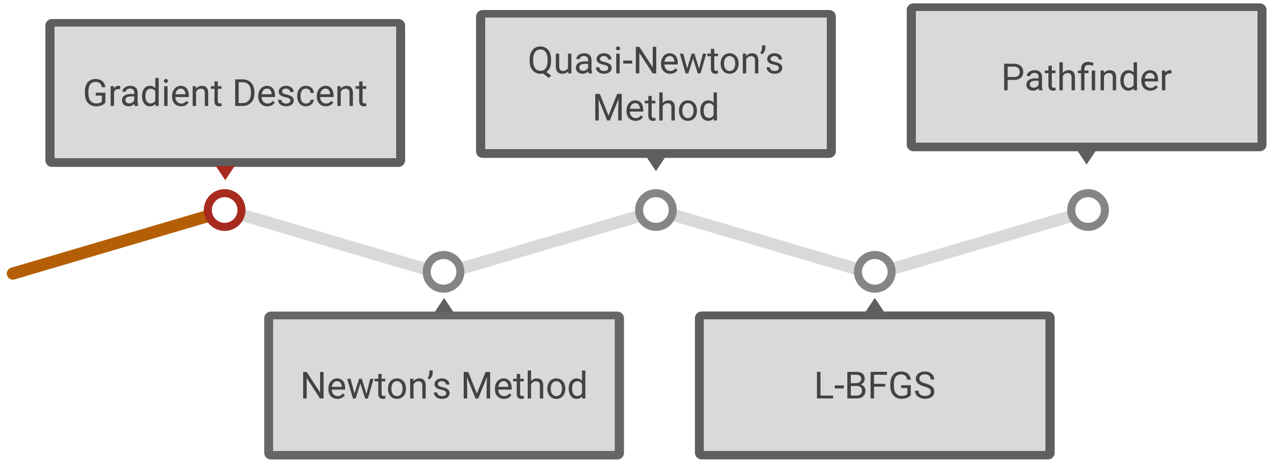
```
[3]: poisson_data = poisson_sampling_model(1000, 50)
     d = np.diff(np.unique(poisson_data)).min()
     left_of_first_bin = poisson_data.min() - float(d)/2
     right_of_last_bin = poisson_data.max() + float(d)/2

     plt.hist(poisson_data, np.arange(left_of_first_bin, right_of_last_bin + d, d))
     plt.show()
```



3. Pathfinder

Pathfinder, what's that?



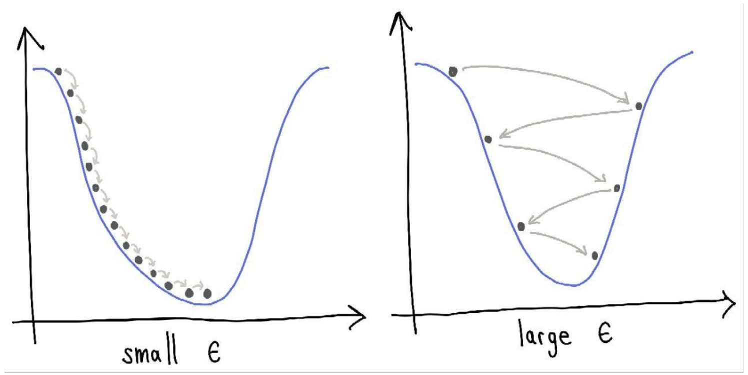
Gradient descent

we want to minimize $f(\mathbf{x})$

let's follow the direction that points towards the minimum

$$\mathbf{x}^{t+1} = \mathbf{x}^{t} - \epsilon f'(\mathbf{x}^{t})$$

1. It's a first order approximation
2. Learning rate must be tuned
3. Might be quite inefficient



Newton’s method

consider the second order expansion

$$f(\mathbf{x}^t + \epsilon) = f(\mathbf{x}^t) + f'(\mathbf{x}^t)\epsilon + \frac{1}{2}f''(\mathbf{x}^t)\epsilon^2$$

and minimize it

$$\frac{d}{d\epsilon}\left[f(\mathbf{x}^t) + f'(\mathbf{x}^t)\epsilon + \frac{1}{2}f''(\mathbf{x}^t)\epsilon^2\right] = f'(\mathbf{x}^t) + f''(\mathbf{x}^t)\epsilon = 0$$

hence

$$\epsilon = -\frac{f'(\mathbf{x}^t)}{f''(\mathbf{x}^t)} \qquad \mathbf{x}^{t+1} = \mathbf{x}^t - \frac{f'(\mathbf{x}^t)}{f''(\mathbf{x}^t)}$$

Newton's method

Looks easy right? Let's generalize to *n* dimensions

$$f'(x) \rightarrow \nabla f \qquad f''(x) \rightarrow H_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

The new update scheme becomes $\mathbf{x}^{t+1} = \mathbf{x}^t - H_f(\mathbf{x}^t)^{-1} \nabla f(\mathbf{x}^t)$

1. Hessian must be positive-definite

2. Computationally expensive

Quasi-Newton's method to the rescue

Let's' go back in one dimension. Assume that

$$f''(\mathbf{x}^{t+1}) \sim \frac{f'(\mathbf{x}^{t+1}) - f'(\mathbf{x}^{t})}{\mathbf{x}^{t+1} - \mathbf{x}^{t}}$$

then

$$\mathbf{x}^{t+1} = \mathbf{x}^{t} - f'(\mathbf{x}^{t}) \frac{\mathbf{x}^{t} - \mathbf{x}^{t-1}}{f'(\mathbf{x}^{t}) - f'(\mathbf{x}^{t-1})}$$

If we find an approximation B of the Hessian matrix such that

$$B^{t+1} = \frac{\nabla f(\mathbf{x}^{t+1}) - \nabla f(\mathbf{x}^{t})}{\mathbf{x}^{t+1} - \mathbf{x}^{t}}$$

We have an efficient update! Do we?

BFGS method to the final rescue

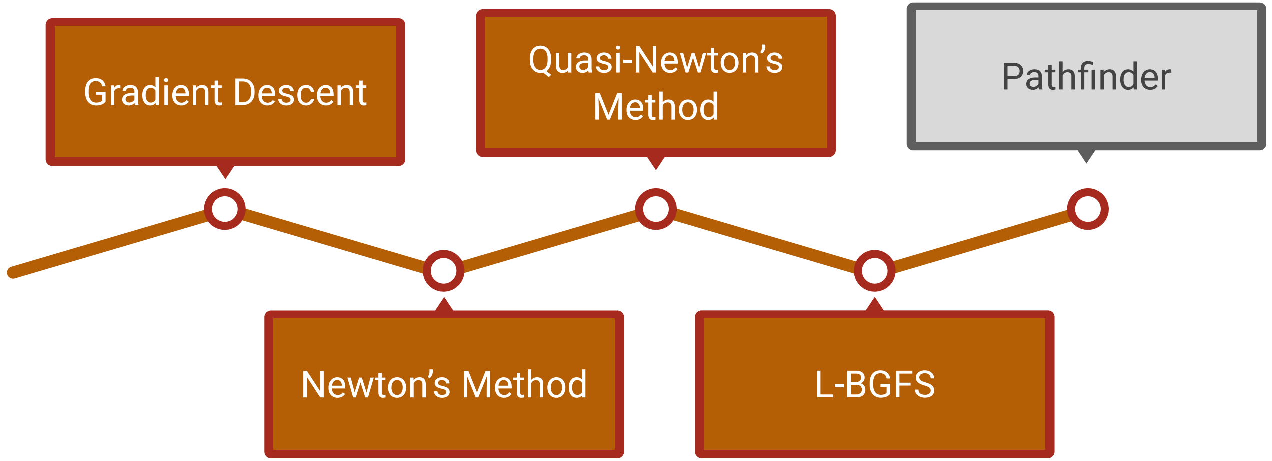
The approximate Hessian B has to many parameters $\longrightarrow$ add constraints on B

B must

1. be positive-definite
2. be symmetric
3. updating B should not change it too much $\longrightarrow$ $\min \left\| B^{t+1} - B^{t} \right\|$

Under such constraints, Broyden, Fletcher, Goldfarb and Shanno produces an efficient update equation.

Pathfinder, almost there



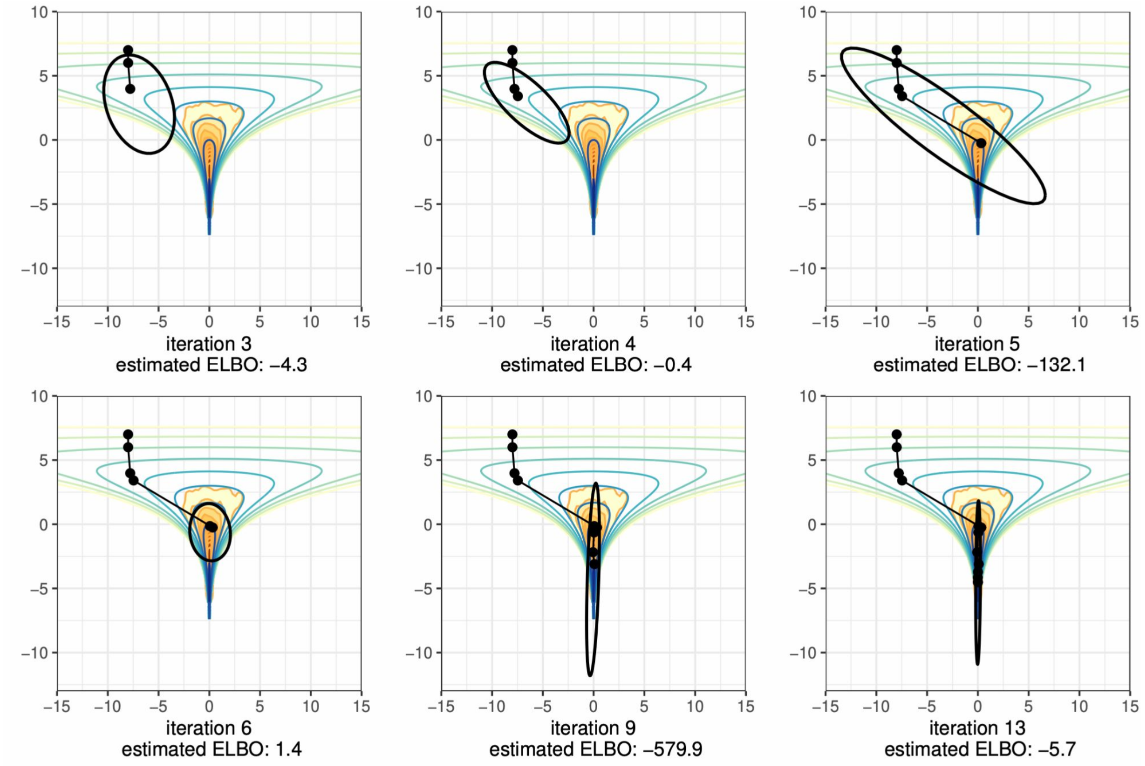
Pathfinder in all its glory

Algorithm 1: Pathfinder

```
Data: $\log p$, $\pi_0$, L, M, K
Result: $\psi_{1:M}$ draws, $\log q(\psi_{1:M})$ log density in ELBO-maximizing normal
        approximation
Compute $L$ normal approximation along the optimization path using L-BFGS
for $l \in 1 : L$ do
    sample $K$ draws $\psi$ and log densities $\log q(\psi)$ for the $l^{th}$ approximation
    for $k \in 1 : K$ do
        evaluate and store $\log p(\psi_k)$
    let $\lambda_l = \mathrm{ELBO}(\log p(\psi_l), \log q(\psi_l)$
let $l^* = \arg\max_l \lambda$
sample $M$ draws $\psi$ and log densities $\log q(\psi)$ for the best approximation
```

From: Zhang, Lu, et al. "Pathfinder: Parallel quasi-Newton variational inference." *Journal of Machine Learning Research* 23.306 (2022): 1-49.

Pathfinder visualized



From: Zhang, Lu, et al. "Pathfinder: Parallel quasi-Newton variational inference." *Journal of Machine Learning Research* 23.306 (2022): 1-49.

Thanks for your attention!

and now let’s do some practice