BLACK BOX VARIATIONAL INFERENCE

David M. Blei
Departments of Computer Science and Statistics
Columbia University

We have ***complicated data***; we want to ***make sense*** of it.

What is ***complicated data?***

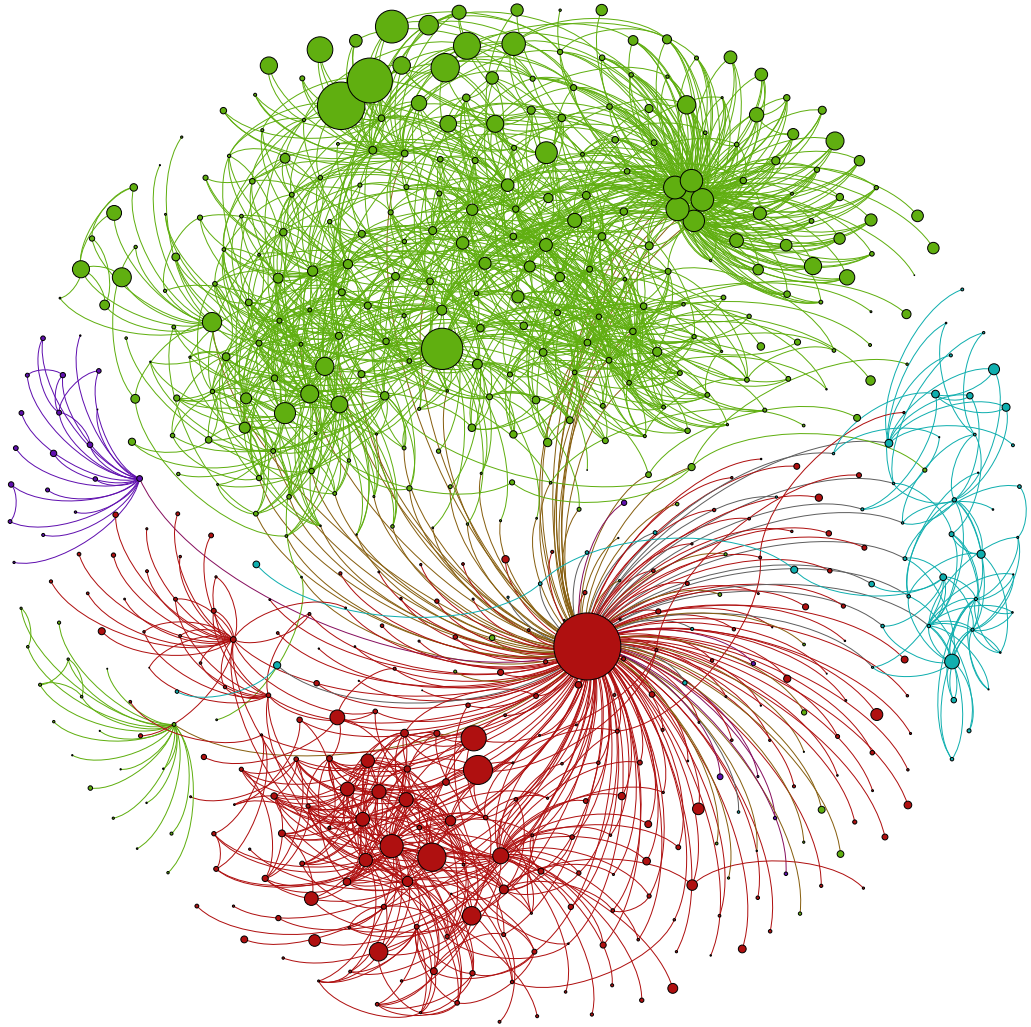
- many data points; many dimensions
- unstructured (e.g. text)
- multimodal and interconnected (e.g., images, links, text, clicks)

What is ***making sense of data***?

- make predictions about the future
- identify interpretable patterns
- do science: confirm, elaborate, form causal theories

PROBABILISTIC MACHINE LEARNING

- ML methods that *connect domain knowledge to data*.
- Provides a computational methodology for scalable modeling
- Goal: A methodology that is *expressive, scalable, easy to develop*

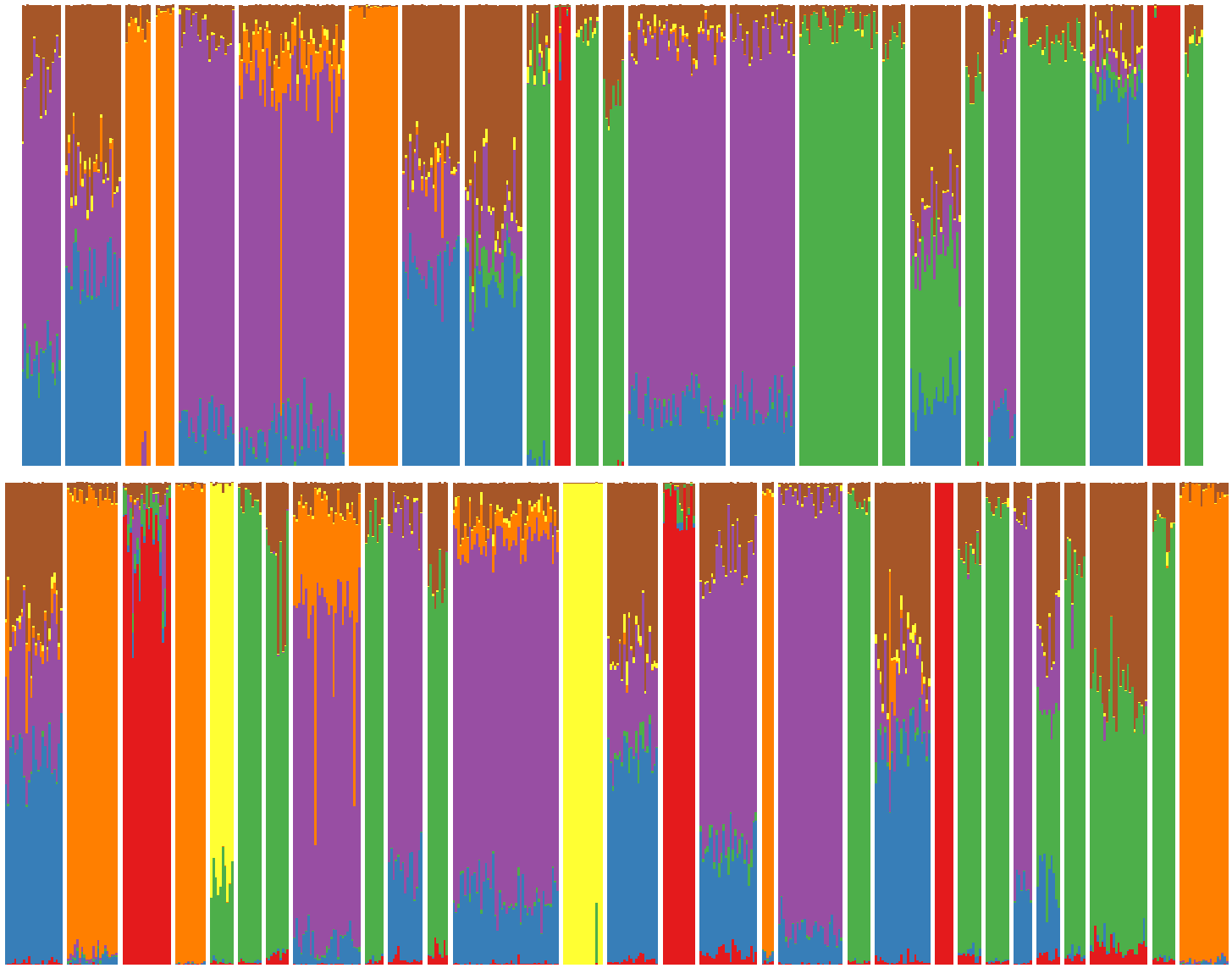


Communities discovered in a 3.7M node network of U.S. Patents

[Gopalan and Blei PNAS 2013]

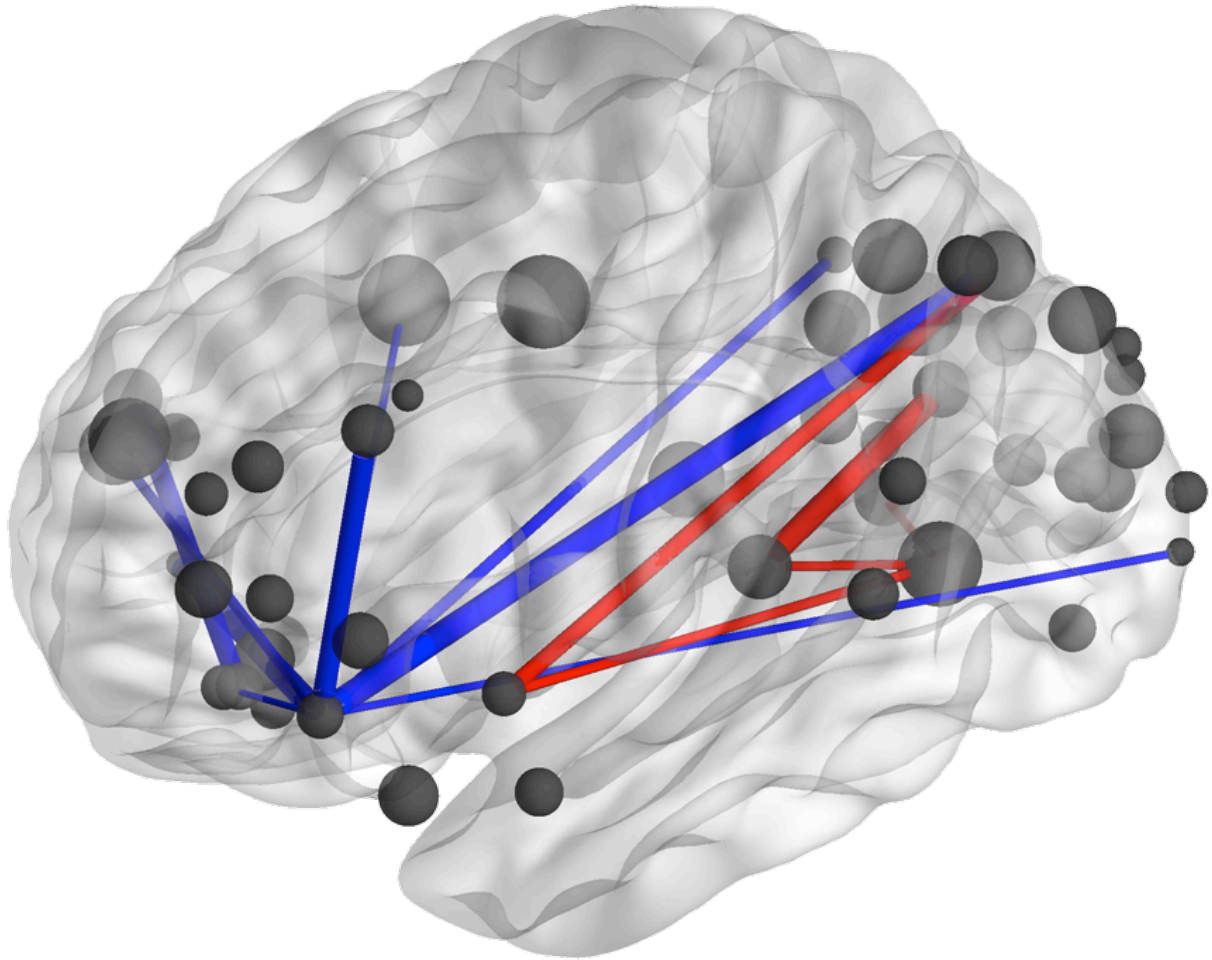
Topics found in 1.8M articles from the New York Times

[Hoffman+ JMLR 2013]



Population analysis of 2 billion genetic measurements

[Gopalan+ Nature Genetics 2016]

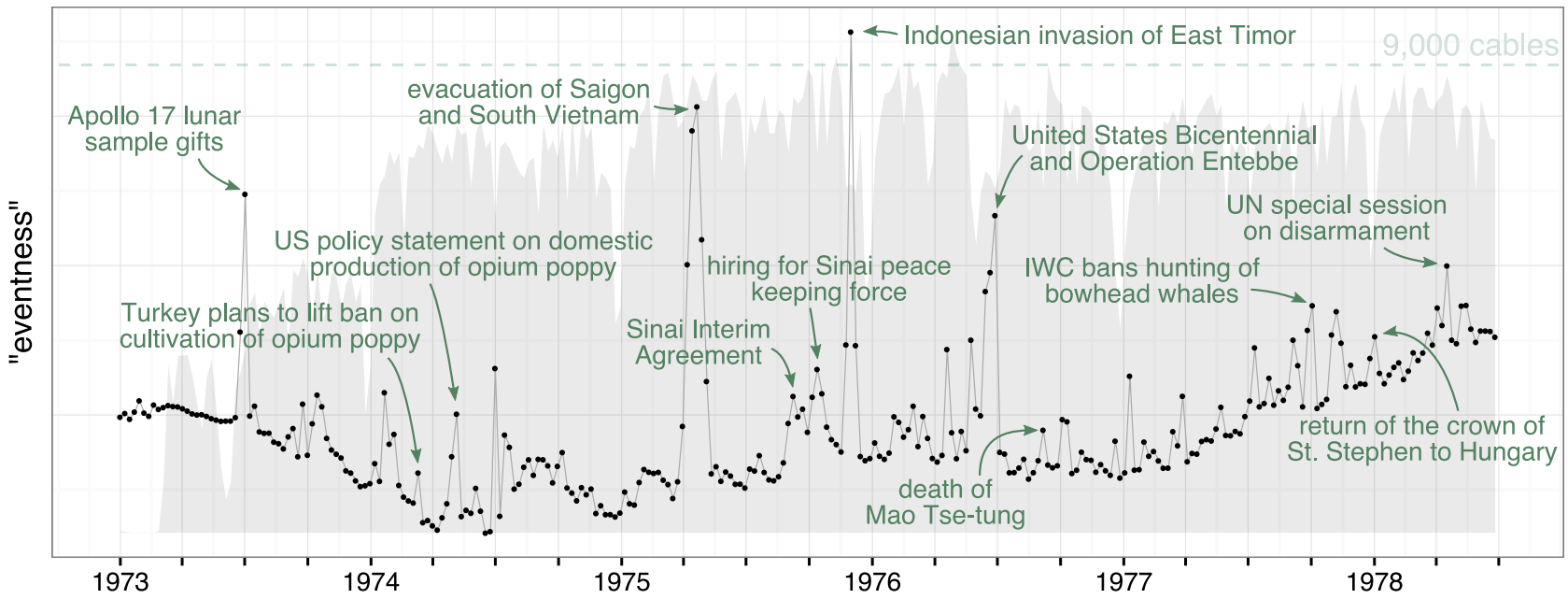


Neuroscience analysis of 220 million fMRI measurements

[Manning+ PLOS ONE 2014]

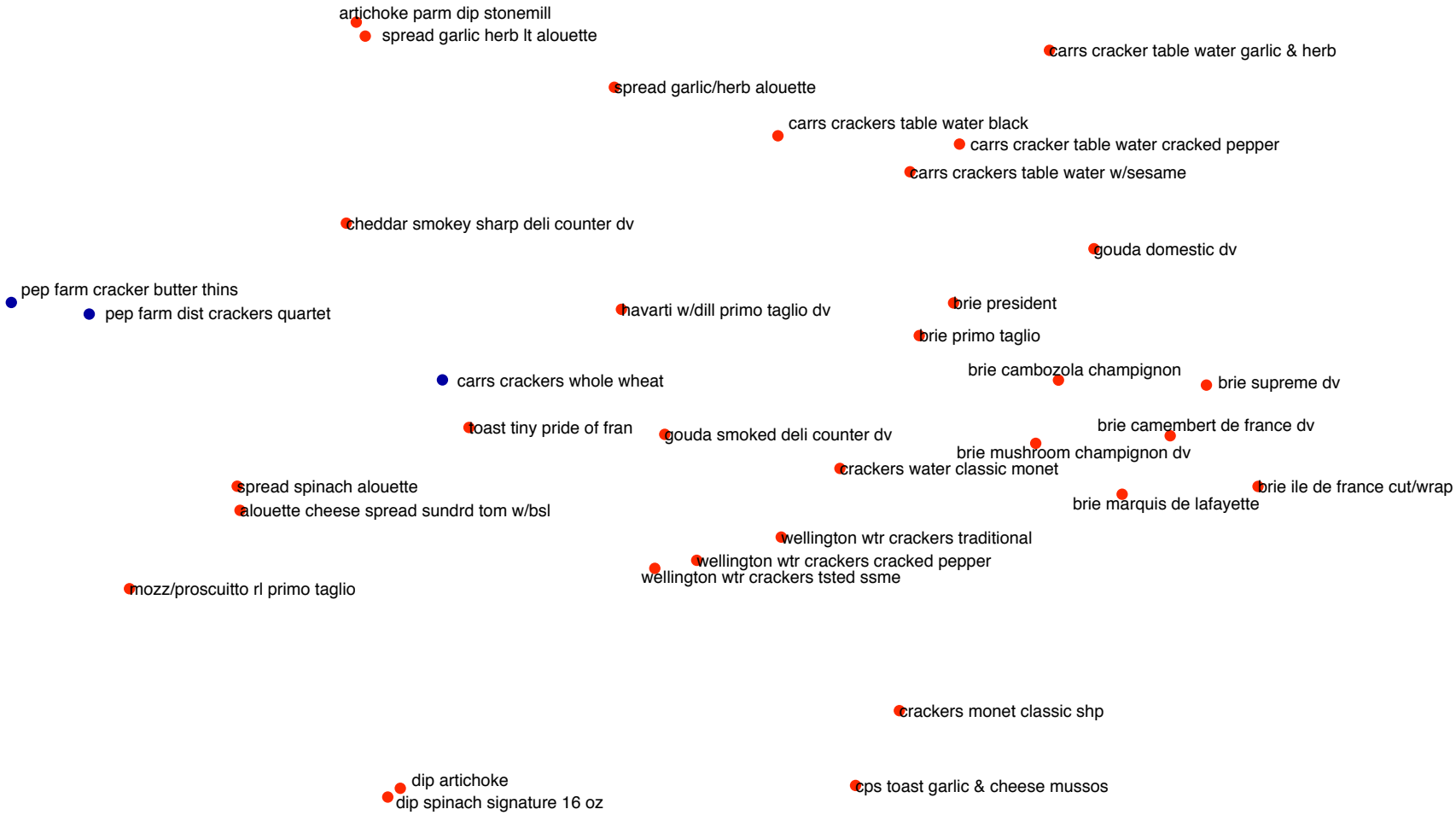
Analysis of 1.7M taxi trajectories, in Stan

[Kucukelbir+ JMLR 2017]



Analysis of 2M declassified cables from the State Dept

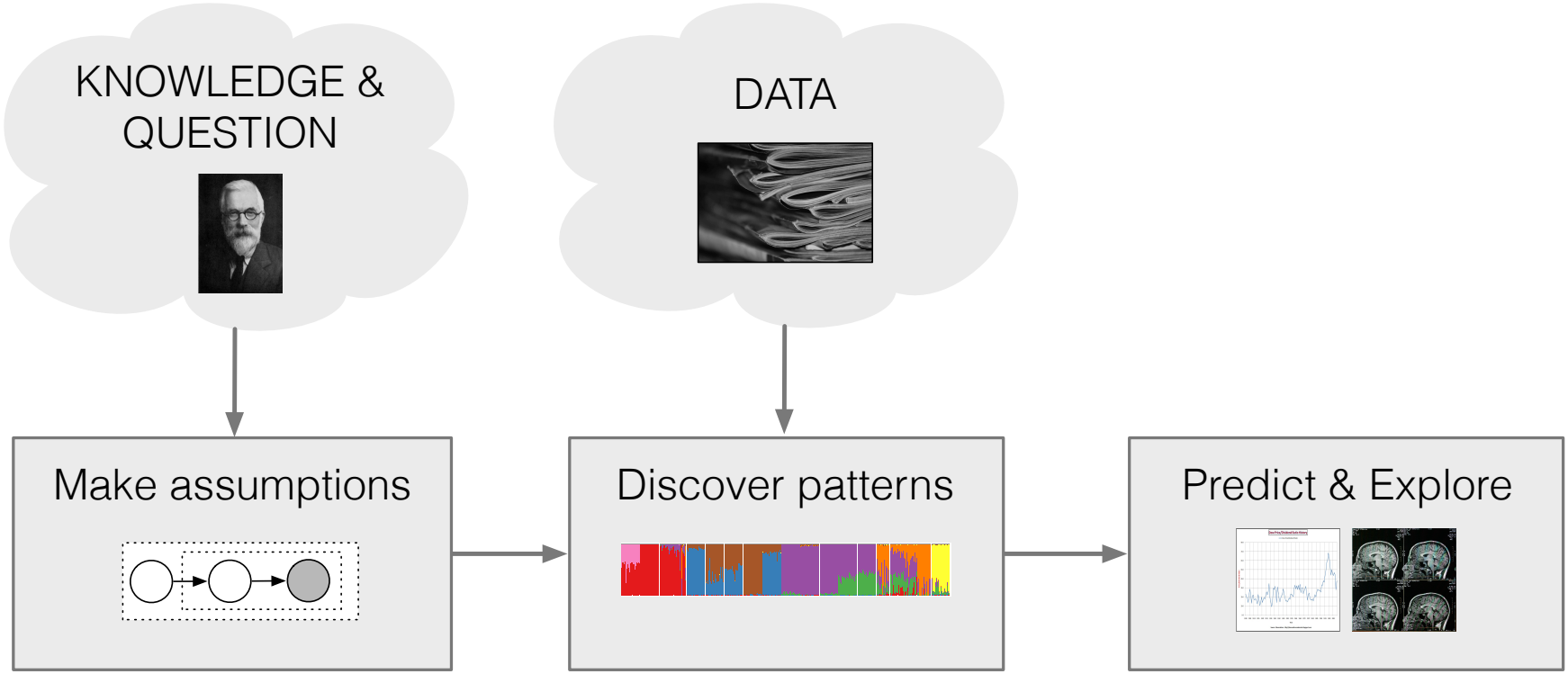
[Chaney+ EMNLP 2016]



(Fancy) discrete choice analysis of 5.7M purchases

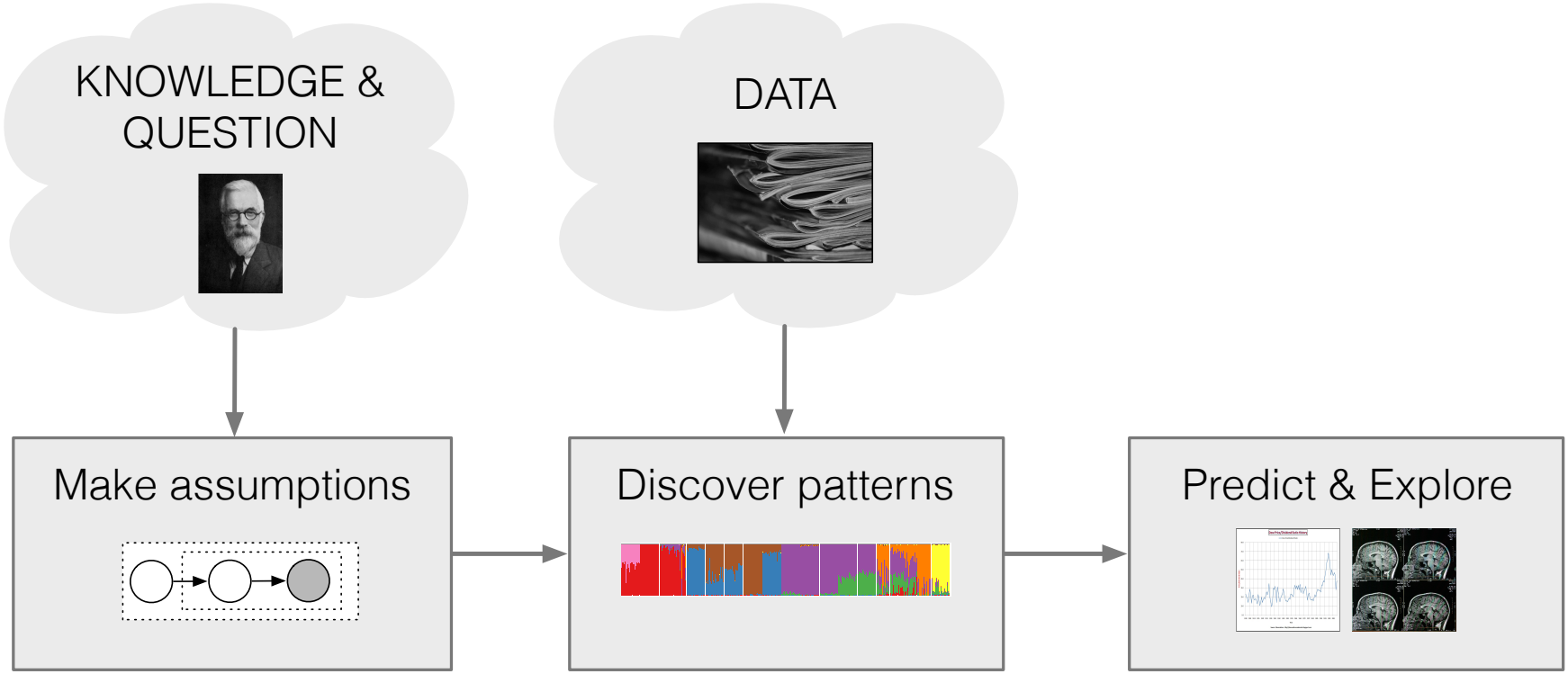
[Ruiz+ 2017]

The probabilistic pipeline



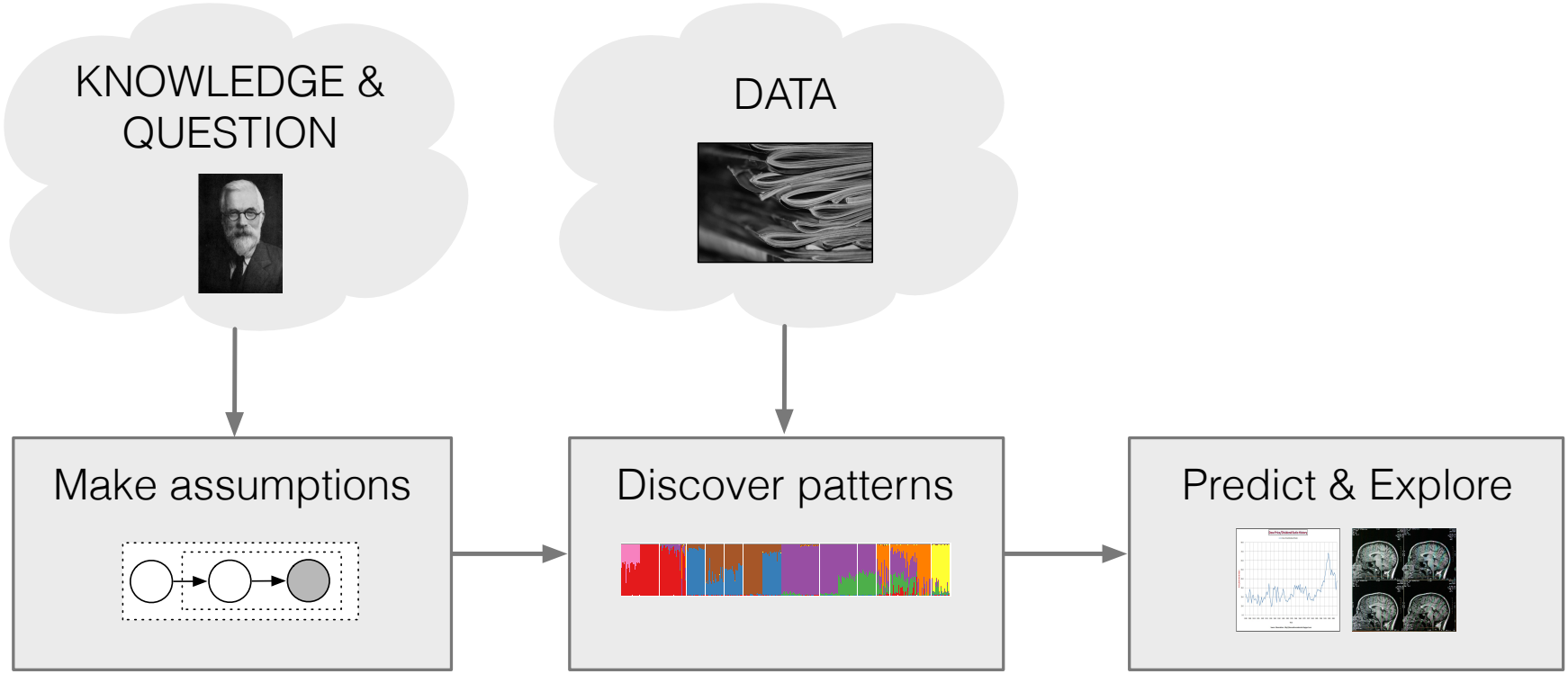
- Customized data analysis is important to many fields.
- Pipeline separates **assumptions**, **computation**, **application**
- Eases collaborative solutions to statistics problems

The probabilistic pipeline

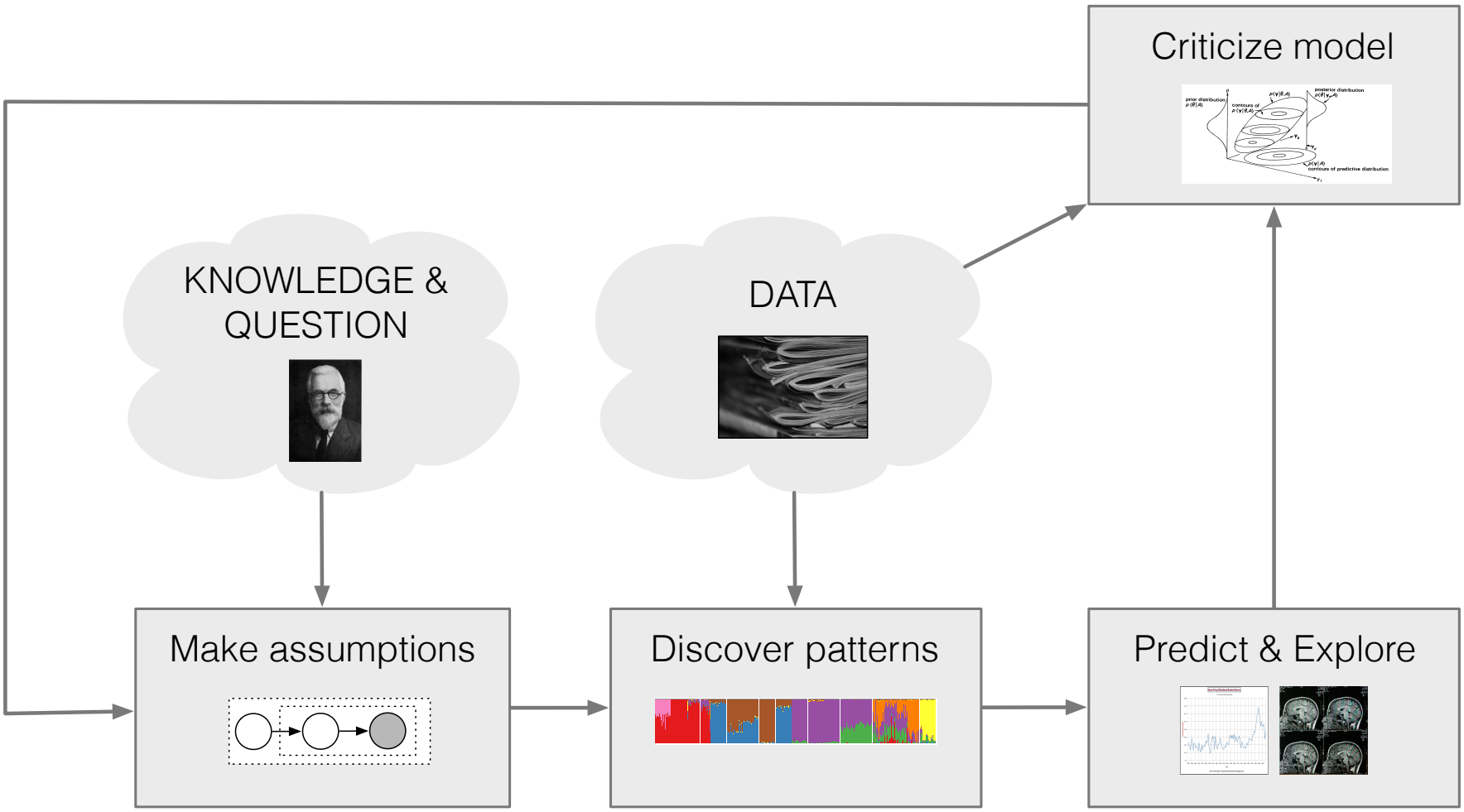


- **Posterior inference** is the key algorithmic problem.
- Answers the question: What does this model say about this data?
- Goal: **General** and **scalable** approaches to posterior inference

The probabilistic pipeline



- Probabilistic programming represents generative models as programs
- Inference engines compile models/programs to an inference executable.
- We can do this with **black box variational inference**.
  Key ideas: *autodifferentiation* and *stochastic optimization*.



[Box, 1980; Rubin, 1984; Gelman+ 1996; Blei, 2014]

Probabilistic machine learning

- A probabilistic model is a joint distribution of hidden variables $\mathbf{z}$ and observed variables $\mathbf{x}$,

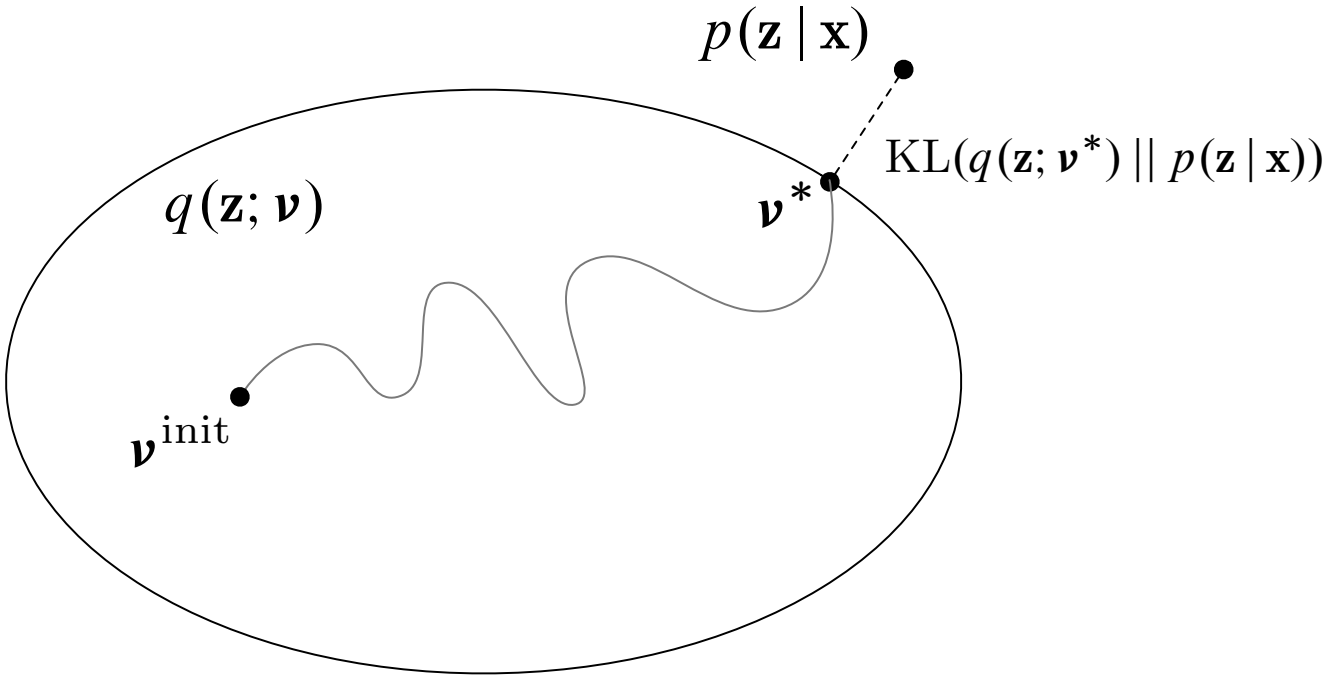
$$p(\mathbf{z}, \mathbf{x}).$$

- Inference about the unknowns is through the **posterior**, the conditional distribution of the hidden variables given the observations

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})}.$$

- For most interesting models, the denominator is not tractable.
  We appeal to **approximate posterior inference**.

Variational inference

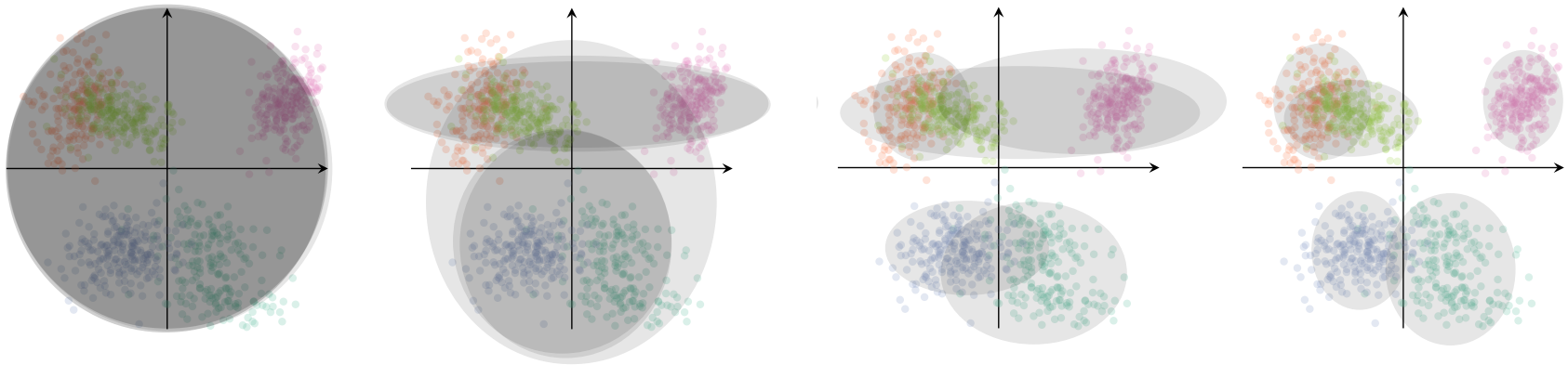


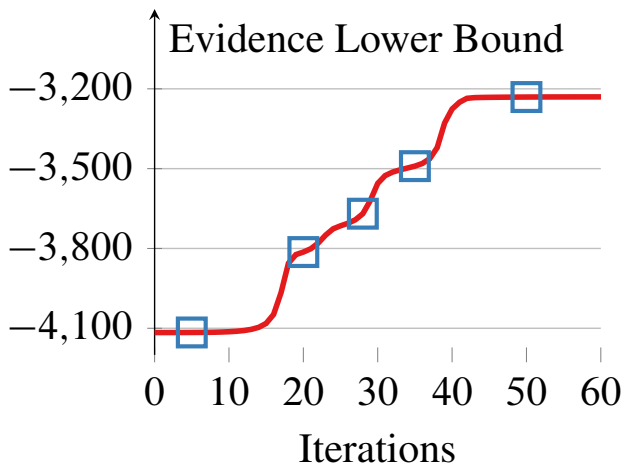
- VI solves **inference** with **optimization**. (Contrast with MCMC.)
- Posit a **variational family** of distributions over the latent variables,

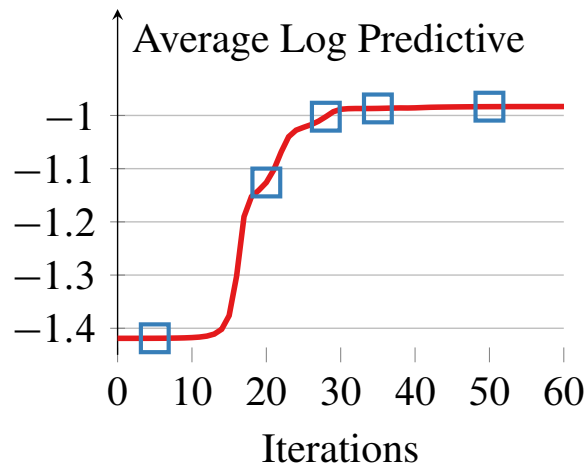
$$q(\mathbf{z}; \boldsymbol{\nu})$$

- Fit the **variational parameters** $\boldsymbol{\nu}$ to be close (in KL) to the exact posterior.
  (There are alternative divergences, which connect to algorithms like EP, BP, and others.)

Example: Mixture of Gaussians







[images by Alp Kucukelbir; Blei+ 2016]

A.1 Computing $\mathrm{E}[\log(\theta_i \mid \alpha)]$

The need to compute the expected value of the log of a single probability component under the Dirichlet arises repeatedly in deriving the inference and parameter estimation procedures for LDA. This value can be easily computed from the natural parameterization of the exponential family representation of the Dirichlet distribution.

Recall that a distribution is in the exponential family if it can be written in the form:

$$p(x \mid \eta) = h(x) \exp\left\{\eta^T T(x) - A(\eta)\right\},$$

where η is the natural parameter, $T(x)$ is the sufficient statistic, and $A(\eta)$ is the log of the normalization factor.

We can write the Dirichlet in this form by exponentiating the log of Eq. (1):

$$p(\theta \mid \alpha) = \exp\left\{\left(\sum_{i=1}^k (\alpha_i - 1)\log\theta_i\right) + \log\Gamma\left(\sum_{i=1}^k \alpha_i\right) - \sum_{i=1}^k \log\Gamma(\alpha_i)\right\}.$$

From this form, we immediately see that the natural parameter of the Dirichlet is $\eta_i = \alpha_i - 1$ and the sufficient statistic is $T(\theta_i) = \log\theta_i$. Furthermore, using the general fact that the derivative of the log normalization factor with respect to the natural parameter is equal to the expectation of the sufficient statistic, we obtain:

$$\mathrm{E}[\log\theta_i \mid \alpha] = \Psi(\alpha_i) - \Psi\left(\sum_{j=1}^k \alpha_j\right)$$

where Ψ is the digamma function, the first derivative of the log Gamma function.

A.3.2 Variational Dirichlet

Next, we maximize Eq. (15) with respect to γ_i, the ith component of the posterior Dirichlet parameter. The terms containing γ_i are:

$$\begin{aligned} L_{[\gamma]} = &\sum_{i=1}^k (\alpha_i - 1)\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right) + \sum_{n=1}^N \phi_{ni}\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right) \\ &- \log\Gamma\left(\sum_{j=1}^k \gamma_j\right) + \log\Gamma(\gamma_i) - \sum_{i=1}^k (\gamma_i - 1)\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right). \end{aligned}$$

This simplifies to:

$$L_{[\gamma]} = \sum_{i=1}^k \left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right)\left(\alpha_i + \sum_{n=1}^N \phi_{ni} - \gamma_i\right) - \log\Gamma\left(\sum_{j=1}^k \gamma_j\right) + \log\Gamma(\gamma_i).$$

We take the derivative with respect to γ_i:

$$\frac{\partial L}{\partial \gamma_i} = \Psi'(\gamma_i)\left(\alpha_i + \sum_{n=1}^N \phi_{ni} - \gamma_i\right) - \Psi'\left(\sum_{j=1}^k \gamma_j\right)\sum_{j=1}^k \left(\alpha_j + \sum_{n=1}^N \phi_{nj} - \gamma_j\right).$$

Setting this equation to zero yields a maximum at:

$$\gamma_i = \alpha_i + \sum_{n=1}^N \phi_{ni}. \tag{17}$$

Since Eq. (17) depends on the variational multinomial ϕ, full variational inference requires alternating between Eqs. (16) and (17) until the bound converges.

Finally, we expand Eq. (14) in terms of the model parameters (α, β) and the variational parameters (γ, ϕ). Each of the five lines below expands one of the five terms in the bound:

$$\begin{aligned} L(\gamma, \phi; \alpha, \beta) = &\log\Gamma\left(\sum_{j=1}^k \alpha_j\right) - \sum_{i=1}^k \log\Gamma(\alpha_i) + \sum_{i=1}^k (\alpha_i - 1)\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right) \\ &+ \sum_{n=1}^N \sum_{i=1}^k \phi_{ni}\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right) \\ &+ \sum_{n=1}^N \sum_{i=1}^k \sum_{j=1}^V \phi_{ni} w_n^j \log\beta_{ij} \\ &- \log\Gamma\left(\sum_{j=1}^k \gamma_j\right) + \sum_{i=1}^k \log\Gamma(\gamma_i) - \sum_{i=1}^k (\gamma_i - 1)\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right) \\ &- \sum_{n=1}^N \sum_{i=1}^k \phi_{ni}\log\phi_{ni}, \end{aligned} \tag{15}$$

where we have made use of Eq. (8).

In the following two sections, we show how to maximize this lower bound with respect to the variational parameters ϕ and γ.

A.3.1 Variational multinomial

We first maximize Eq. (15) with respect to ϕ_{ni}, the probability that the nth word is generated by latent topic i. Observe that this is a constrained maximization since $\sum_{i=1}^k \phi_{ni} = 1$.

We form the Lagrangian by isolating the terms which contain ϕ_{ni} and adding the appropriate Lagrange multipliers. Let β_{iv} be $p(w_n^v = 1 \mid z^i = 1)$ for the appropriate v. (Recall that each w_n is a vector of size V with exactly one component equal to one; we can select the unique v such that $w_n^v = 1$):

$$L_{[\phi_{ni}]} = \phi_{ni}\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right) + \phi_{ni}\log\beta_{iv} - \phi_{ni}\log\phi_{ni} + \lambda_n\left(\sum_{j=1}^k \phi_{ni} - 1\right),$$

where we have dropped the arguments of L for simplicity, and where the subscript ϕ_{ni} denotes that we have retained only those terms in L that are a function of ϕ_{ni}. Taking derivatives with respect to ϕ_{ni}, we obtain:

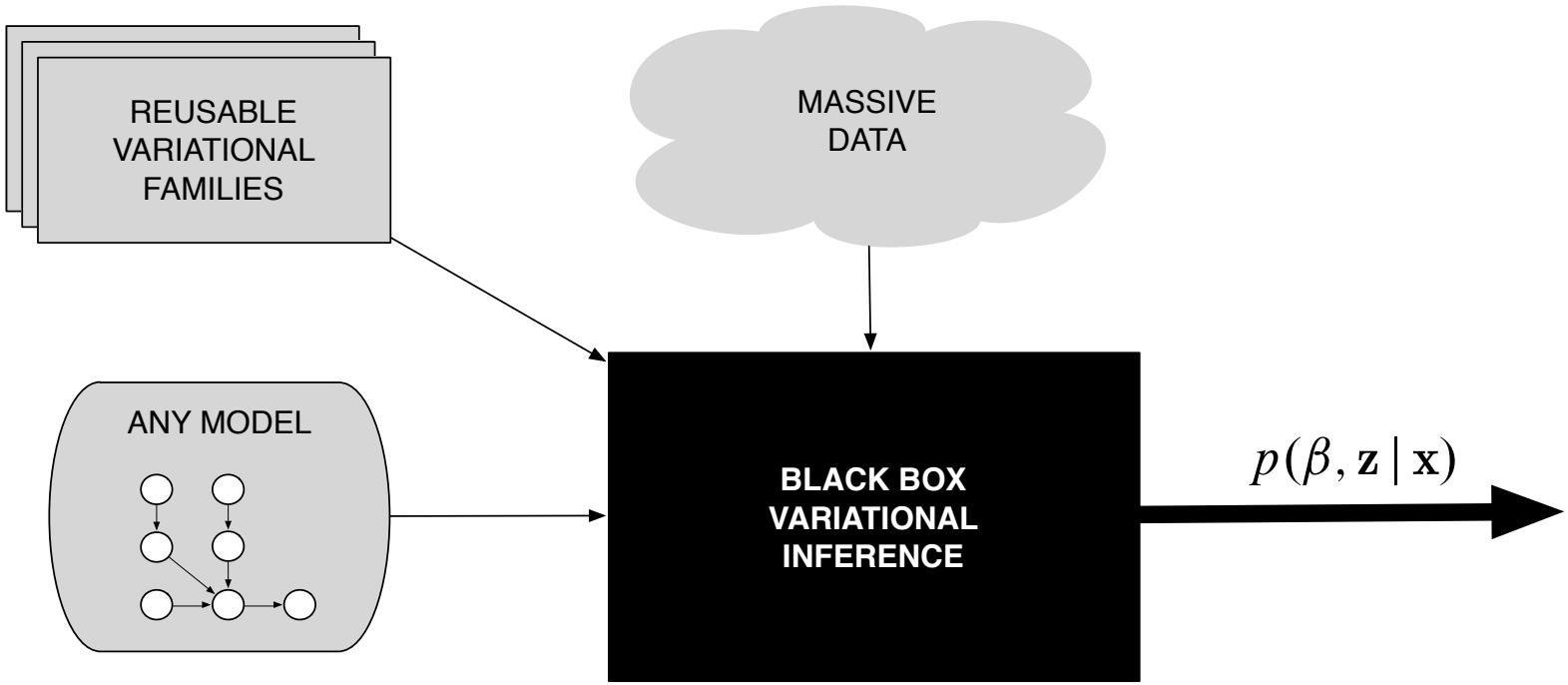
$$\frac{\partial L}{\partial \phi_{ni}} = \Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right) + \log\beta_{iv} - \log\phi_{ni} - 1 + \lambda.$$

Setting this derivative to zero yields the maximizing value of the variational parameter ϕ_{ni} (cf. Eq. 6):

$$\phi_{ni} \propto \beta_{iv}\exp\left(\Psi(\gamma_i) - \Psi\left(\sum_{j=1}^k \gamma_j\right)\right). \tag{16}$$

[from Blei+ 2003]

Black box variational inference

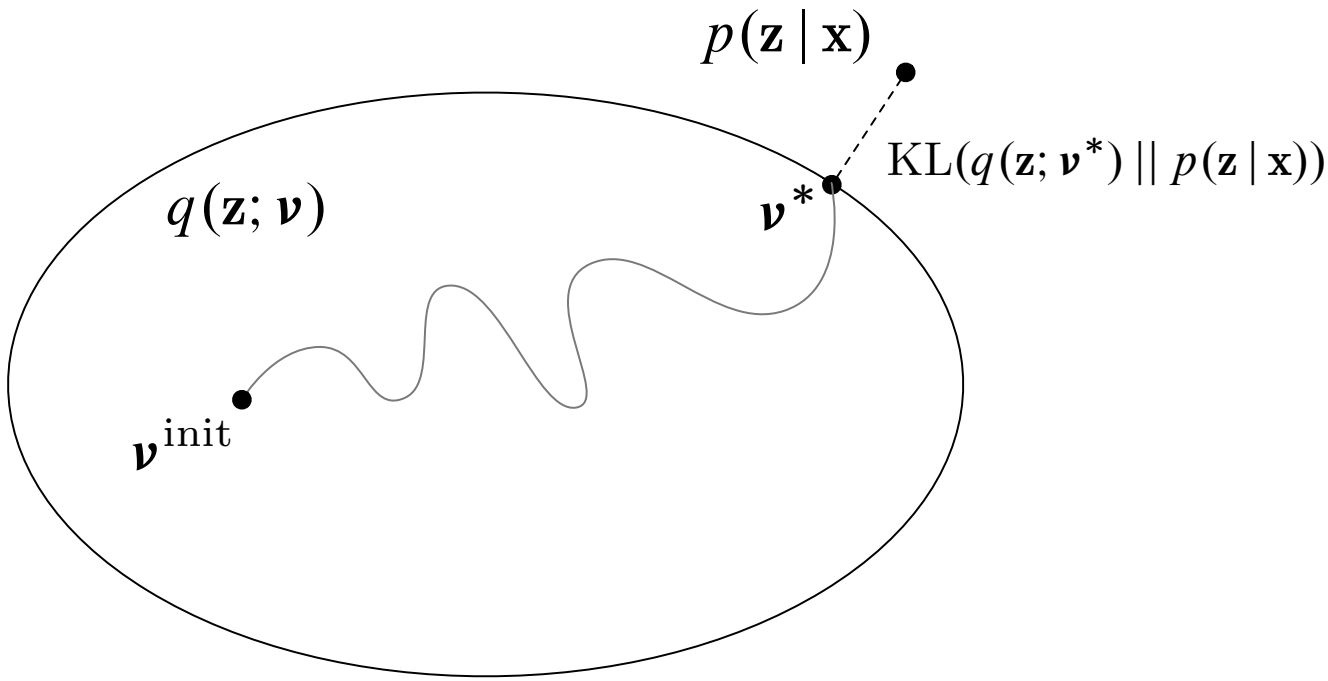


- Easily use variational inference with **any model**; no more appendices!
- Perform inference with **massive data**
- **No mathematical work** beyond specifying the model

BBVI enables probabilistic programming

- Anglican
- Edward/Tensorflow Probability
- PyMC3
- Pyro
- Stan
- Venture
- WebPPL
- (& others...)

Variational inference



- VI solves **inference** with **optimization**.
- Posit a **variational family** of distributions over the latent variables.
- Fit the **variational parameters** $\boldsymbol{\nu}$ to be close (in KL) to the exact posterior.

The evidence lower bound

$$\mathcal{L}(\boldsymbol{\nu}) = \underbrace{\mathbb{E}_q\left[\log p(\mathbf{z}, \mathbf{x})\right]}_{\text{Expected complete log likelihood}} - \underbrace{\mathbb{E}_q\left[\log q(\mathbf{z}; \boldsymbol{\nu})\right]}_{\text{Negative entropy}}$$

- KL is intractable; VI optimizes the **evidence lower bound** (ELBO) instead.
  - It is a lower bound on $\log p(\mathbf{x})$.
  - Maximizing the ELBO is equivalent to minimizing the KL.
- The ELBO trades off two terms.
  - The first term prefers $q(\cdot)$ to place its mass on the MAP estimate.
  - The second term encourages $q(\cdot)$ to be diffuse.
- Caveat: The ELBO is not convex.

Black box variational inference

$$\mathcal{L}(\boldsymbol{\nu}) = \underbrace{\mathbb{E}_q\left[\log p(\mathbf{z}, \mathbf{x})\right]}_{\text{Expected complete log likelihood}} - \underbrace{\mathbb{E}_q\left[\log q(\mathbf{z}; \boldsymbol{\nu})\right]}_{\text{Negative entropy}}$$

- Black box variational inference (BBVI)
  - Write the *gradient as an expectation* with respect to $q(\cdot)$.
  - Sample from $q(\cdot)$ to take a *Monte Carlo estimate of the gradient*.
  - Use the MC estimate in a *stochastic optimization*.
- Keep in mind the black box criteria:
  - sample from $q(\mathbf{z}; \boldsymbol{\nu})$
  - evaluate $q(\mathbf{z}; \boldsymbol{\nu})$
  - evaluate $\log p(\mathbf{z}, \mathbf{x})$
- Two main ideas: **score gradients** and **reparameterization gradients**

The score gradient

$$\nabla_{\boldsymbol{\nu}} \mathcal{L} = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\underbrace{\nabla_{\boldsymbol{\nu}} \log q(\mathbf{z};\boldsymbol{\nu})}_{\text{score function}} \underbrace{(\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\boldsymbol{\nu}))}_{\text{instantaneous ELBO}}]$$

- Use the score function to write the gradient as an expectation.
- Also called the likelihood ratio or REINFORCE gradient
  [Glynn 1990; Williams 1992; Wingate+ 2013; Ranganath+ 2014; Mnih+ 2014]
- Satisfies the black box criteria — no model-specific analysis needed.
  - sample from $q(\mathbf{z};\boldsymbol{\nu})$
  - evaluate $\nabla_{\boldsymbol{\nu}} \log q(\mathbf{z};\boldsymbol{\nu})$
  - evaluate $\log p(\mathbf{x},\mathbf{z})$ and $\log q(\mathbf{z})$

Algorithm 1: Basic Black Box Variational Inference

Input: data $\mathbf{x}$, model $p(\mathbf{z}, \mathbf{x})$.

Initialize $\boldsymbol{\nu}$ randomly.
Set ρ_j appropriately.

while *not converged* **do**

Take S samples from the variational distribution

$$\mathbf{z}[s] \sim q(\mathbf{z}; \boldsymbol{\nu}) \quad s = 1 \ldots S$$

Calculate the noisy score gradient

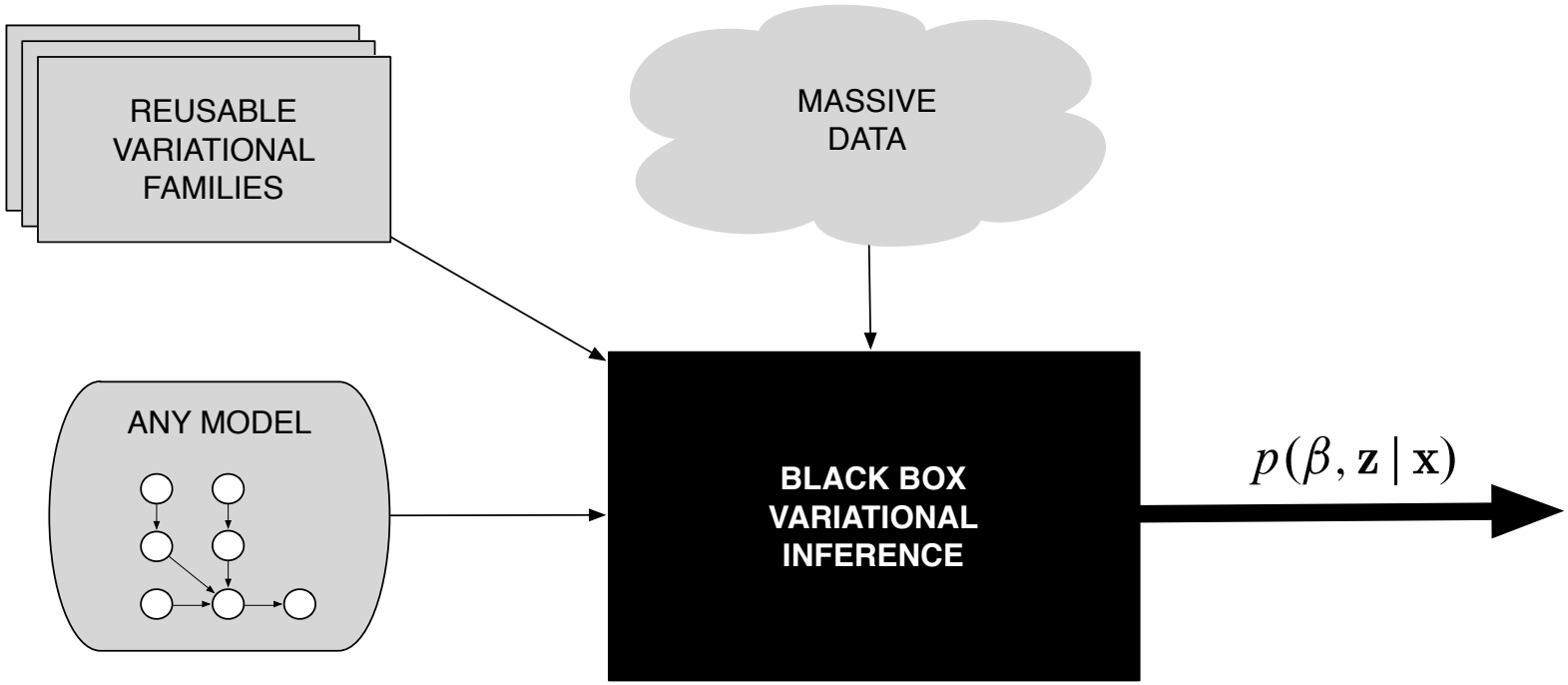
$$\tilde{g}_j = \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\nu}} \log q(\mathbf{z}[s]; \boldsymbol{\nu}_t)(\log p(\mathbf{x}, \mathbf{z}[s]) - \log q(\mathbf{z}[s]; \boldsymbol{\nu}_t))$$

Update the variational parameters

$$\boldsymbol{\nu}_{j+1} = \boldsymbol{\nu}_j + \rho_j \tilde{g}_j$$

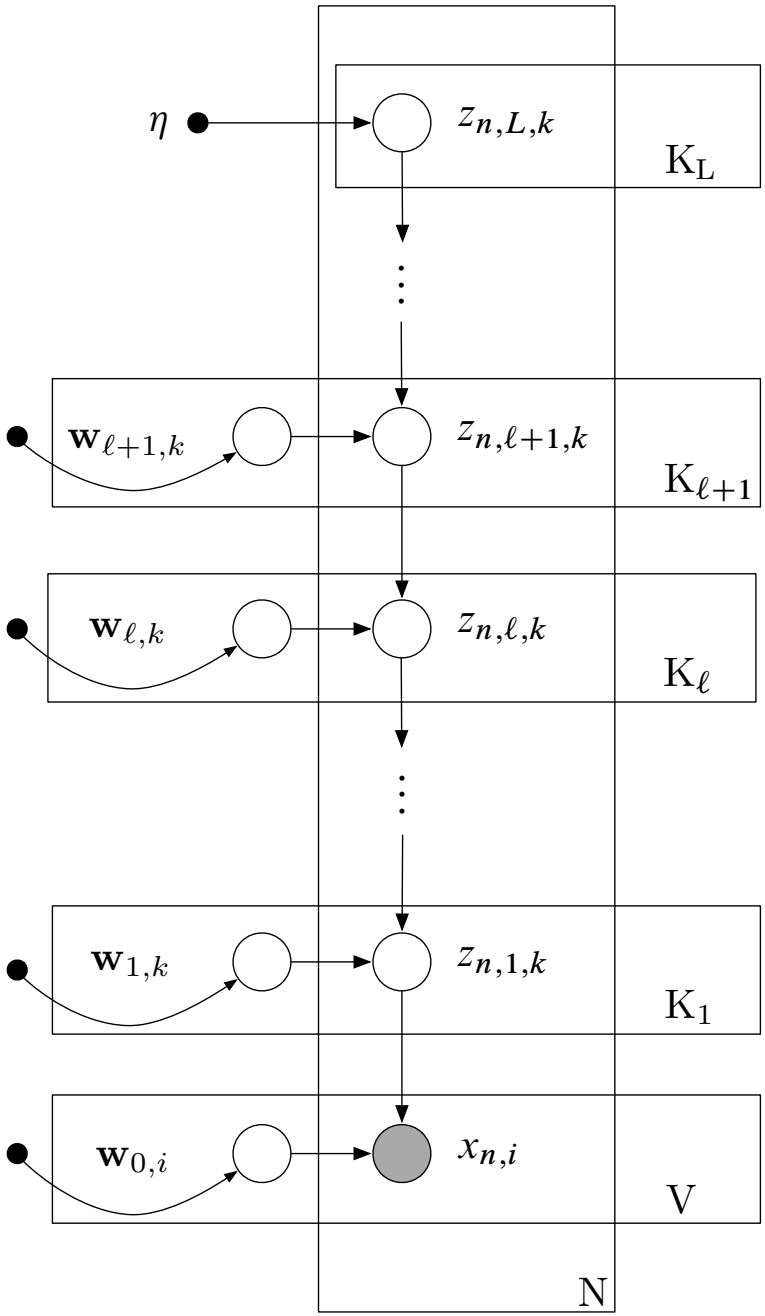
end

BBVI: Making it work



- Control the variance of the gradient
  - Rao-Blackwellization, control variates, importance sampling, ...
- Adaptive step sizes
  [e.g., Duchi+ 2011; Tieleman and Hinton 2012; Kingma and Ba 2014]
- Data subsampling, for massive data
  [Hoffman+ 2013]

Example: Deep exponential families



$z_{n,L,k} \sim \text{Exp-Fam}(\eta)$

$z_{n,\ell,k} \sim \text{Exp-Fam}(g(\mathbf{w}_{\ell,k}^\top \mathbf{z}_{n,\ell+1}))$

$x_{n,i} \sim \text{Exp-Fam}(g(\mathbf{w}_{0,i}^\top \mathbf{z}_{n,1}))$

Example: Deep exponential families

Model	$p(\mathbf{w})$	*NYT*	*Science*
LDA [Blei+ 2003]		2717	1711
DocNADE [Larochelle+ 2012]		2496	1725
Sparse Gamma 100	$\emptyset$	2525	1652
Sparse Gamma 100-30	Γ	2303	1539
Sparse Gamma 100-30-15	Γ	**2251**	1542
Sigmoid 100	$\emptyset$	2343	1633
Sigmoid 100-30	$\mathcal{N}$	2653	1665
Sigmoid 100-30-15	$\mathcal{N}$	2507	1653
Poisson 100	$\emptyset$	2590	1620
Poisson 100-30	$\mathcal{N}$	2423	1560
Poisson 100-30-15	$\mathcal{N}$	2416	1576
Poisson log-link 100-30	Γ	2288	**1523**
Poisson log-link 100-30-15	Γ	2366	1545

The reparameterization gradient

- Express the variational distribution with a *transformation*,

$$\begin{aligned}\boldsymbol{\epsilon} &\sim s(\boldsymbol{\epsilon}) \\ \mathbf{z} &= t(\boldsymbol{\epsilon}, \boldsymbol{\nu}) \\ &\rightarrow \mathbf{z} \sim q(\mathbf{z}; \boldsymbol{\nu})\end{aligned}$$

- For example,

$$\begin{aligned}\epsilon &\sim \text{Normal}(0, 1) \\ z &= \epsilon\sigma + \mu \\ &\rightarrow z \sim \text{Normal}(\mu, \sigma^2)\end{aligned}$$

- Also assume $\log p(\mathbf{x}, \mathbf{z})$ and $\log q(\mathbf{z})$ are *differentiable with respect to* $\mathbf{z}$.

The reparameterization gradient

$$\nabla_\nu \mathcal{L} = \mathbb{E}_{s(\epsilon)} \left[\underbrace{\nabla_{\mathbf{z}} [\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \nu)]}_{\text{gradient of instananeous ELBO}} \quad \underbrace{\nabla_\nu t(\epsilon, \nu)}_{\text{gradient of transformation}} \right]$$

- This is the reparameterization gradient.
  [Glasserman 1991; Fu 2006; Kingma+ 2014; Rezende+ 2014; Titsias+ 2014]
- Can use autodifferentiation to take gradients (especially of the model)
- Can use and reuse different transformations [e.g., Naesseth+ 2017]

Algorithm 2: Reparameterization Black Box Variational Inference

Input: data $\mathbf{x}$, model $p(\mathbf{z}, \mathbf{x})$.

Initialize $\boldsymbol{\nu}$ randomly.
Set ρ_j appropriately.

while *not converged* **do**

Take S samples from the auxillary variable

$$\boldsymbol{\epsilon}_s \sim s(\epsilon) \quad s = 1 \ldots S$$

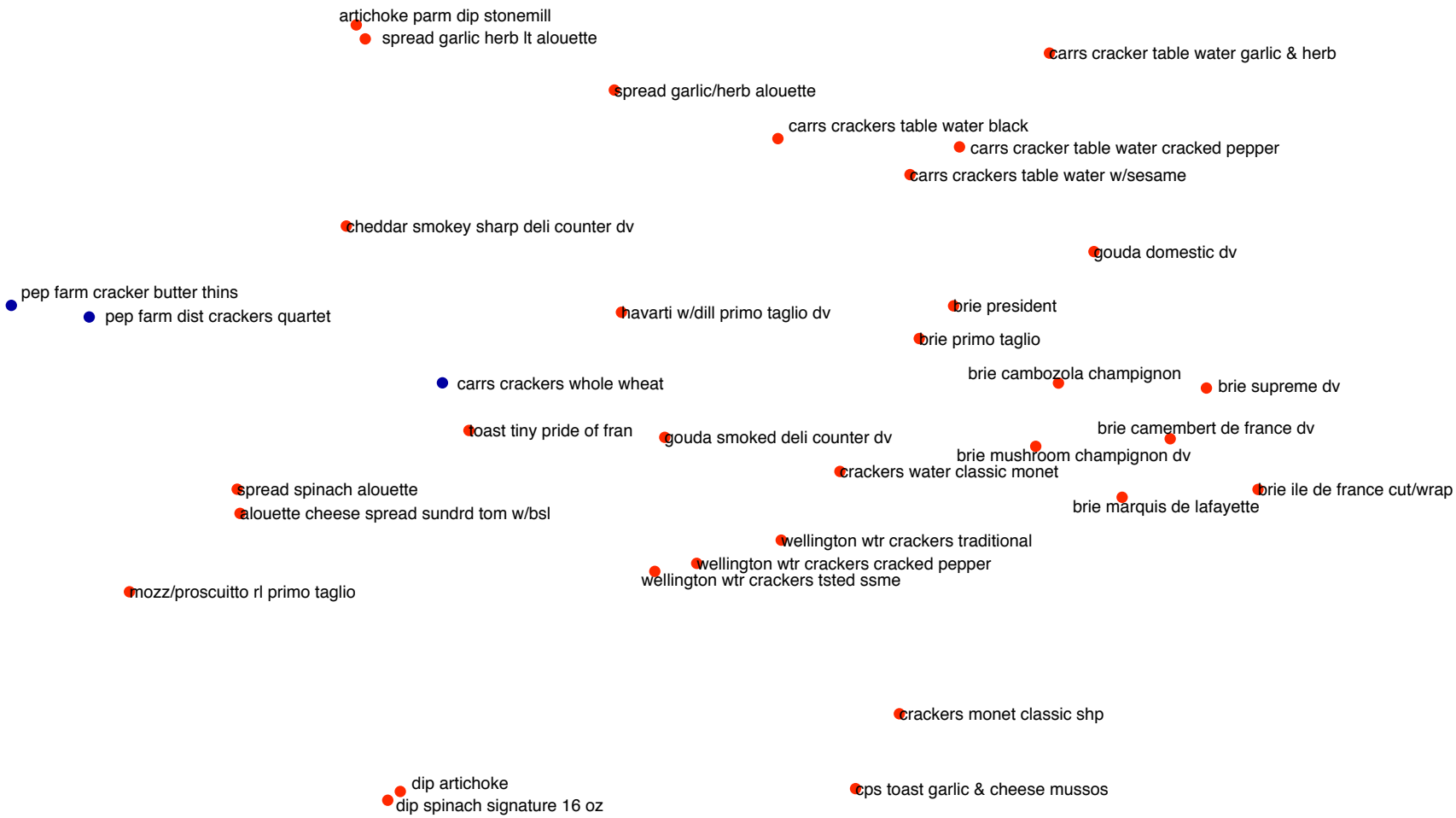
Calculate the noisy gradient

$$\tilde{g}_j = \frac{1}{S} \sum_{s=1}^{S} \nabla_{\mathbf{z}} [\log p(\mathbf{x}, t(\boldsymbol{\epsilon}_s, \boldsymbol{\nu}_n)) - \log q(t(\boldsymbol{\epsilon}_s, \boldsymbol{\nu}_n); \boldsymbol{\nu}_n)] \nabla_{\boldsymbol{\nu}} t(\boldsymbol{\epsilon}_s, \boldsymbol{\nu}_n)$$

Update the variational parameters

$$\boldsymbol{\nu}_{j+1} = \boldsymbol{\nu}_j + \rho_j \tilde{g}_j$$

end



Discrete choice model on 5.7M purchases.

[Ruiz+ 2017]

Analysis of 1.7M taxi trajectories, in Stan

[Kucukelbir+ 2017]

Score gradient

$$\mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\nabla_{\boldsymbol{\nu}} \log q(\mathbf{z};\boldsymbol{\nu})(\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\boldsymbol{\nu}))]$$

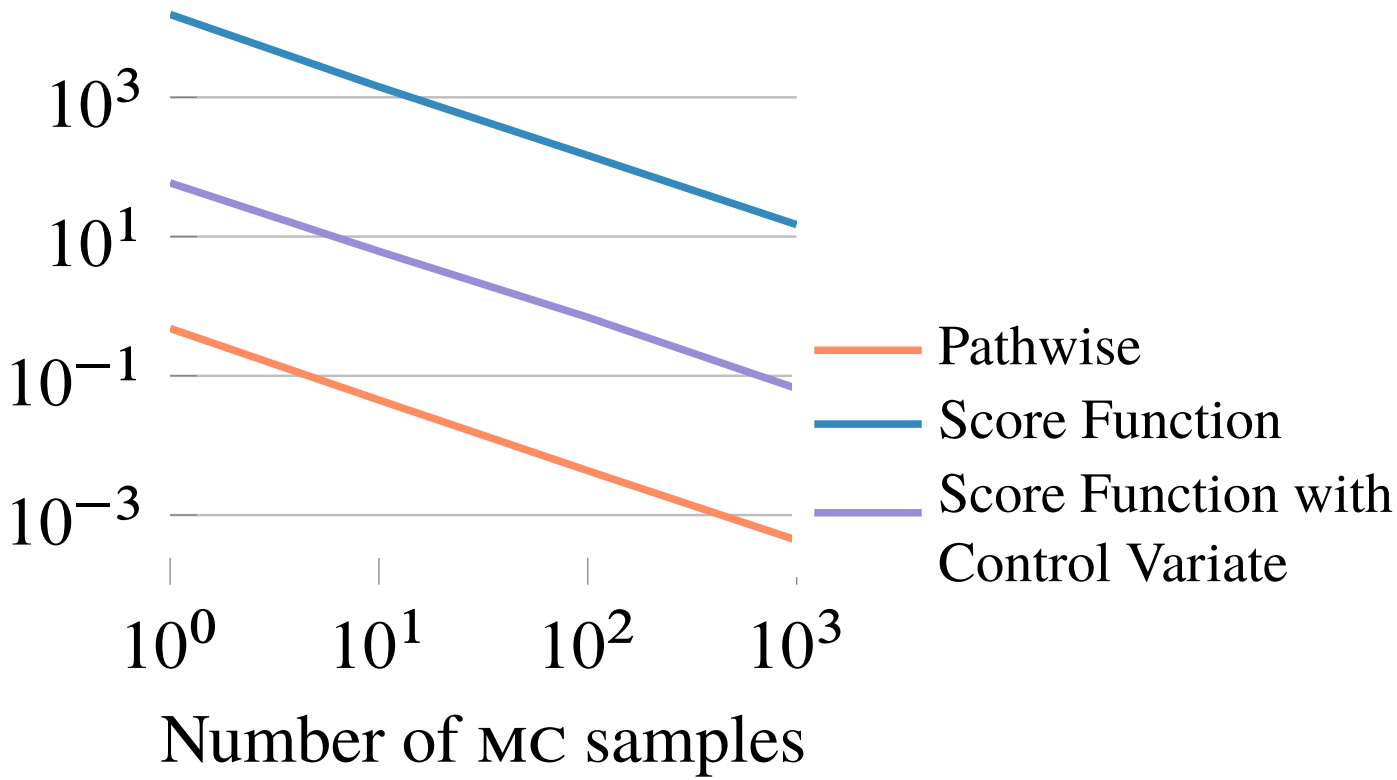
- Works for discrete and continuous models
- Works for a large class of variational approximations
- But the variance of the noisy gradient can be large

Reparameterization gradient

$$\mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\mathbf{z}}[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\boldsymbol{\nu})]\nabla_{\boldsymbol{\nu}} t(\boldsymbol{\epsilon},\boldsymbol{\nu})]$$

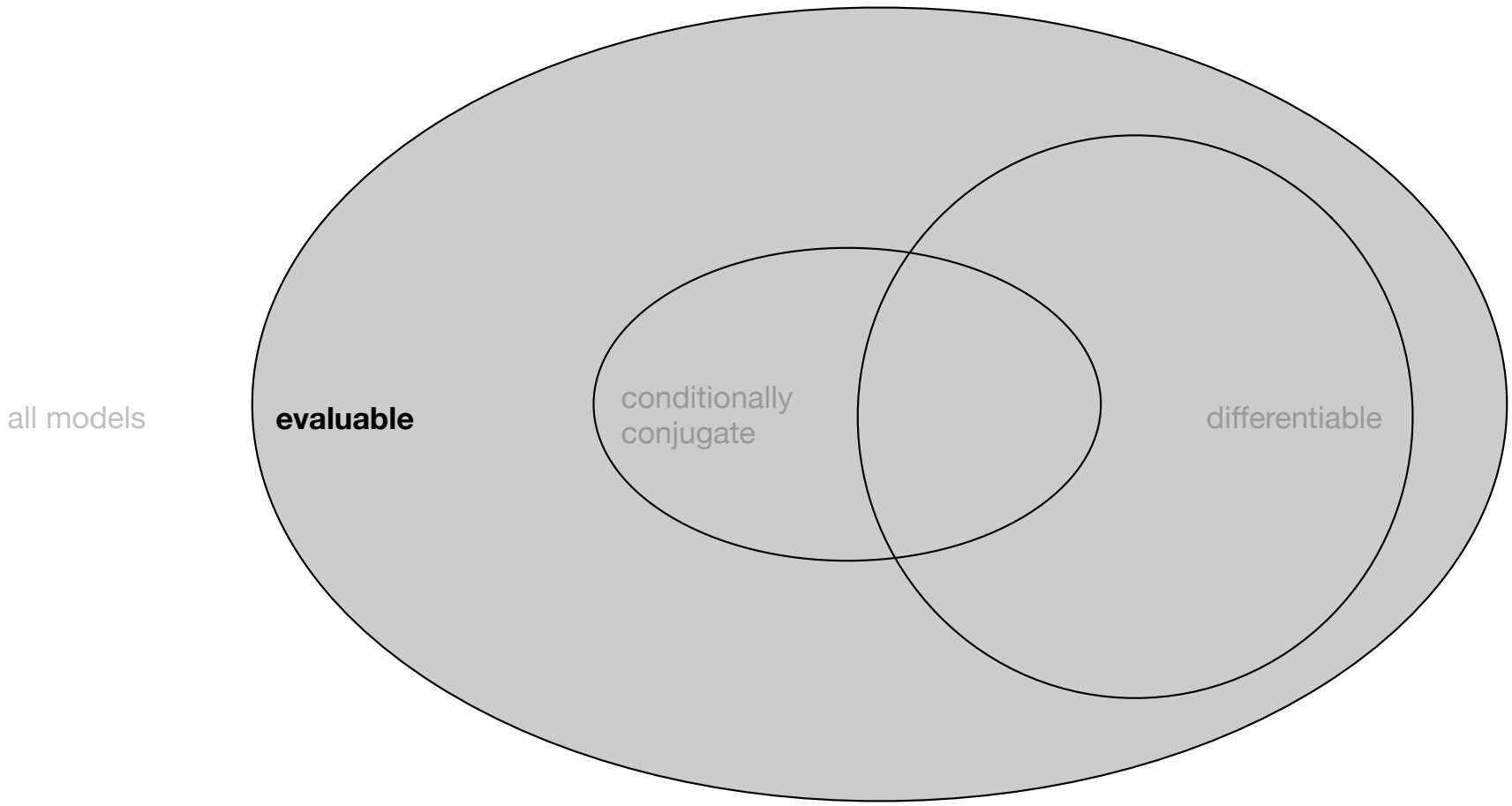
- Requires differentiable models, i.e., no discrete variables
- Requires variational approximation to have form $\mathbf{z} = t(\boldsymbol{\epsilon},\boldsymbol{\nu})$
- Better behaved variance

Variance comparison

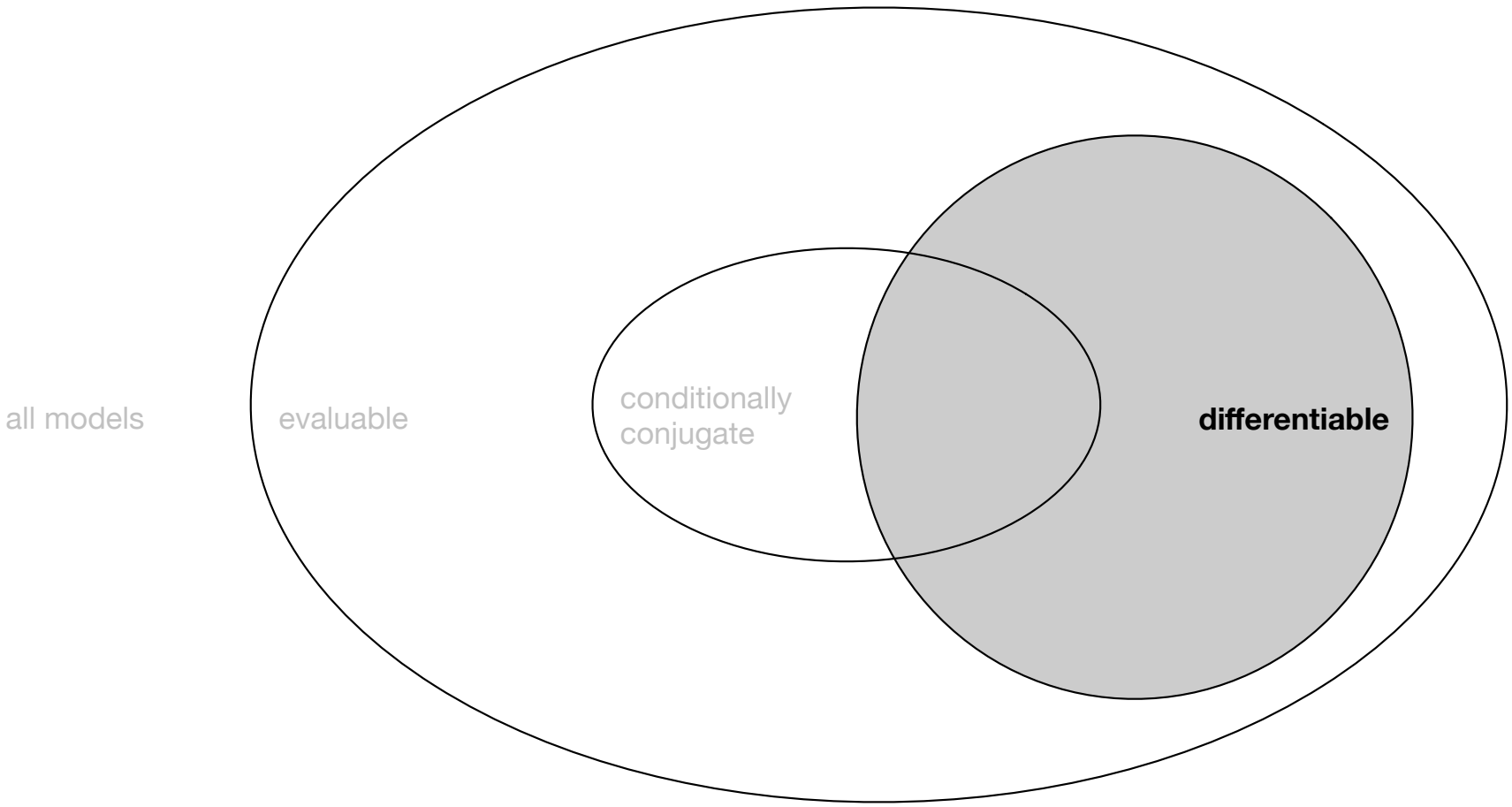


[Kucukelbir+ 2017]

Models that can use the score gradient



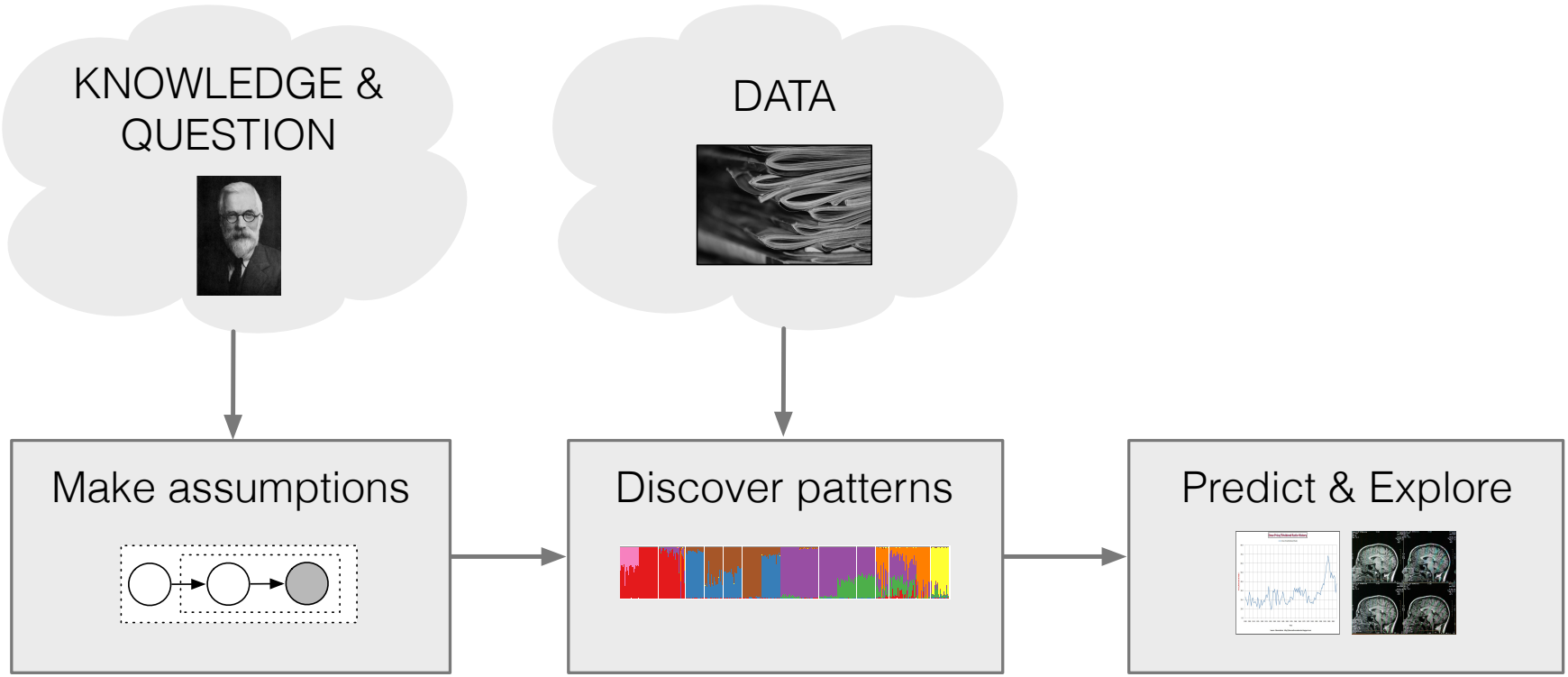
Models that can use the reparameterization gradient



PROBABILISTIC MACHINE LEARNING

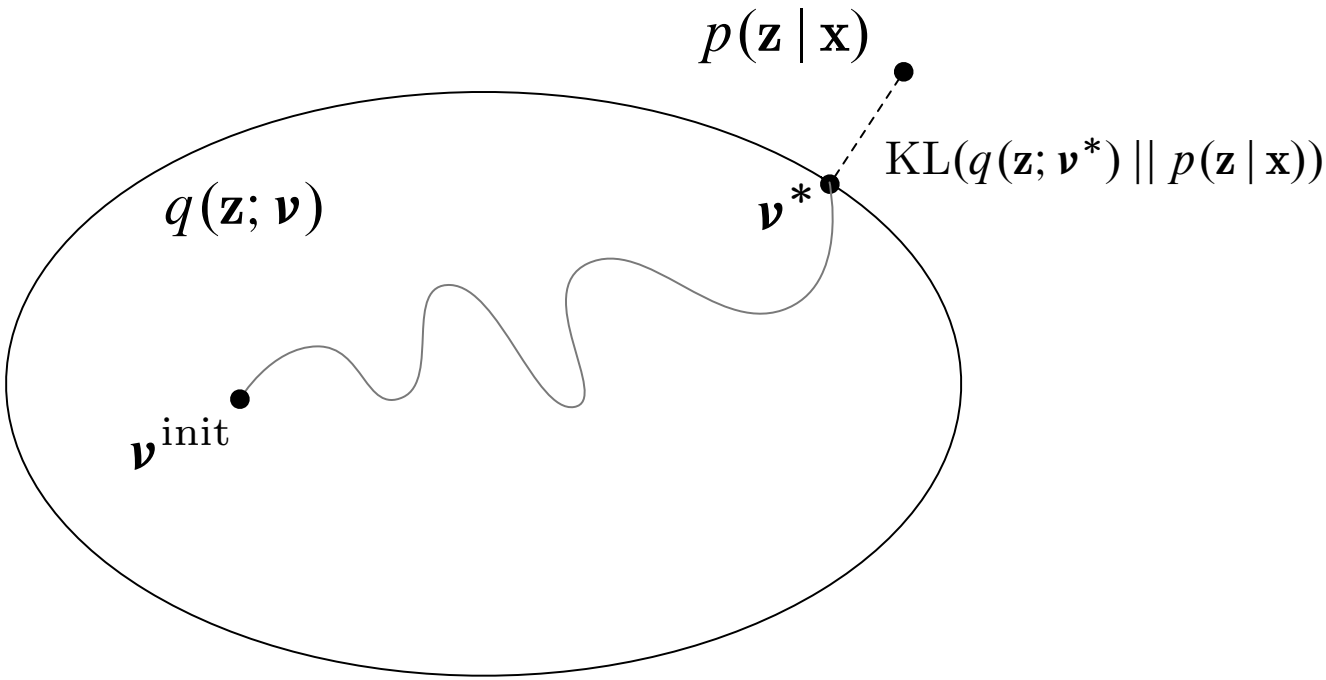
- ML methods that *connect domain knowledge to data*.
- Provides a computational methodology for scalable modeling
- Goal: A methodology that is *expressive, scalable, easy to develop*

The probabilistic pipeline



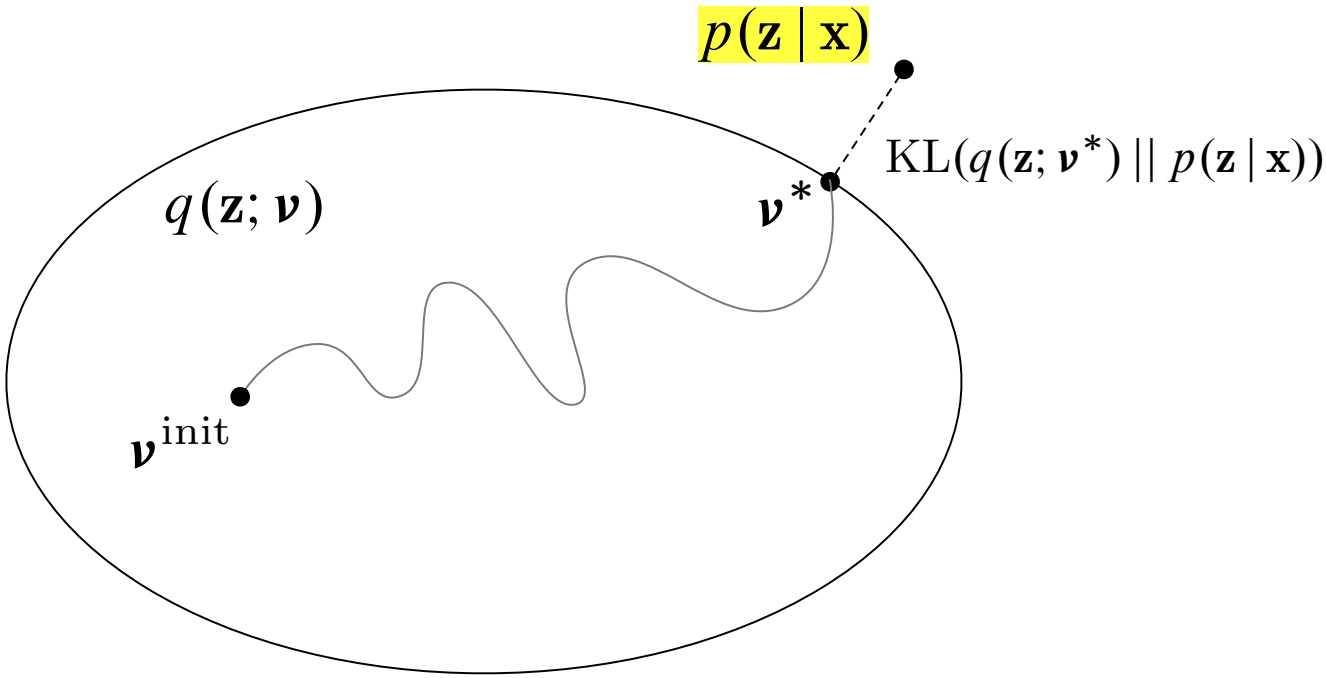
- **Posterior inference** is the key algorithmic problem.
- Answers the question: What does this model say about this data?
- BBVI provides **general** and **scalable** approaches to posterior inference

A tour of research in VI



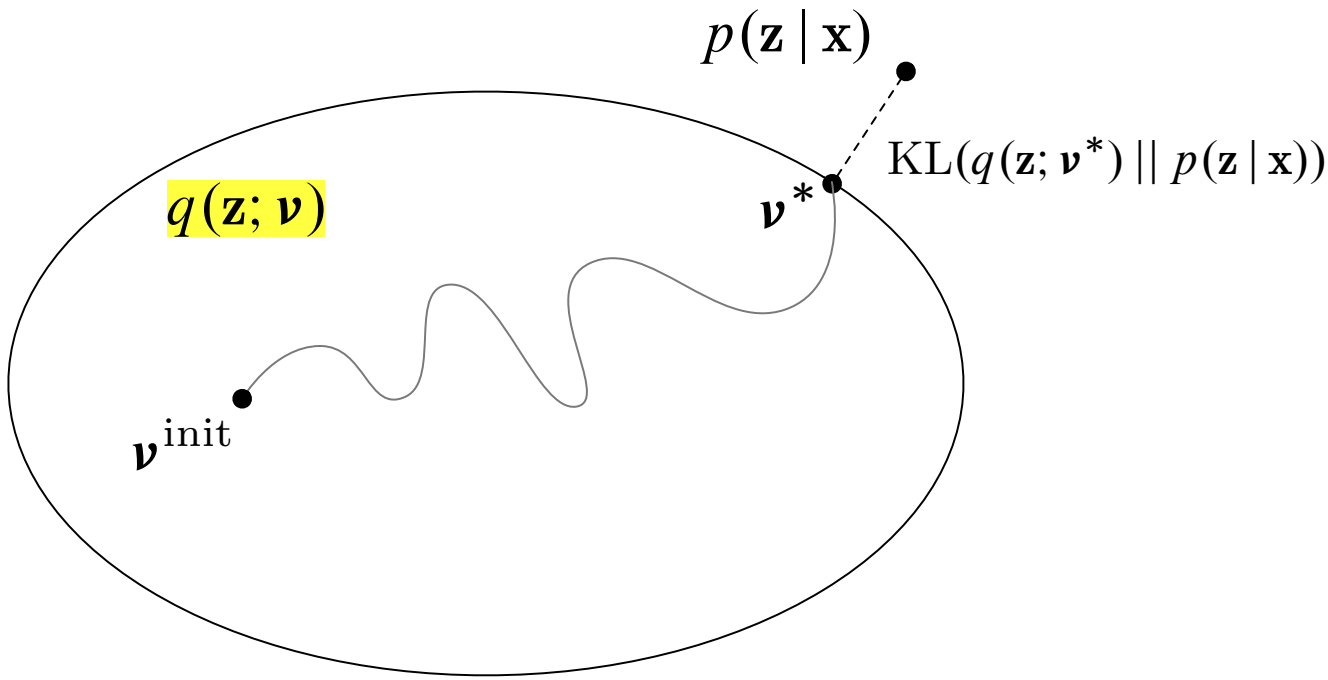
This picture helps situate the flurry of research on VI.

The class of models



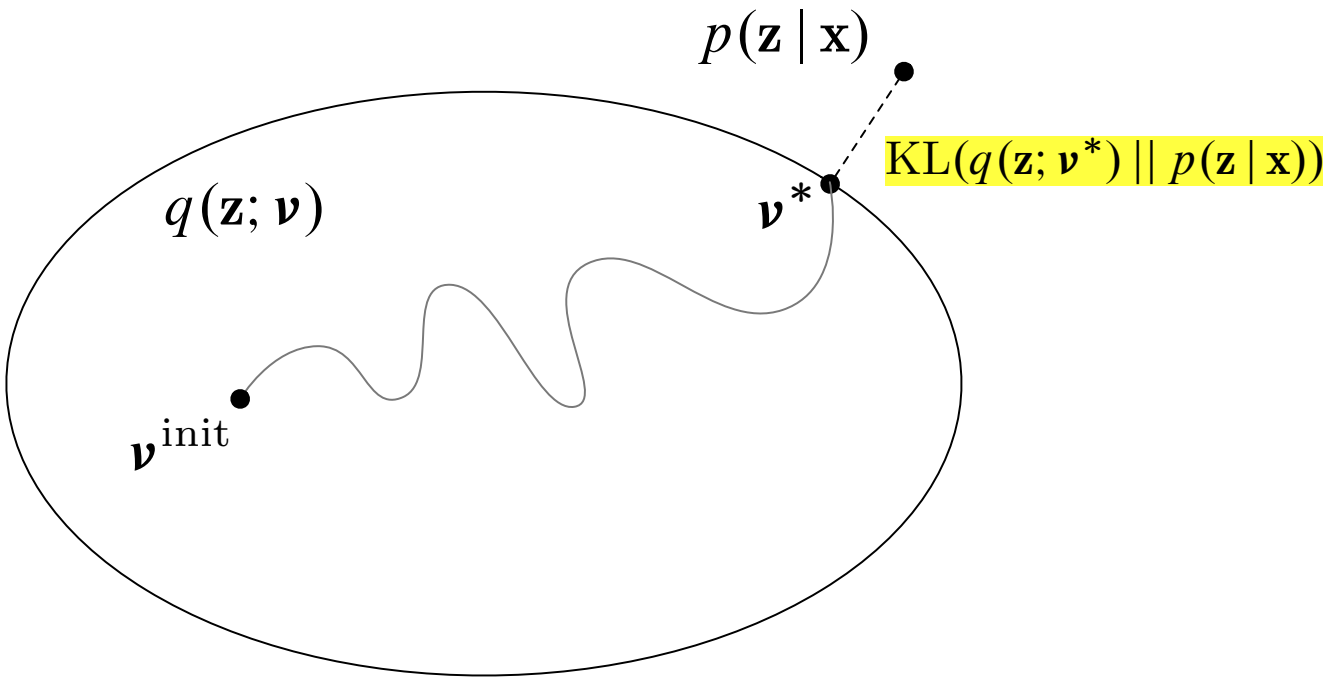
- Conditionally conjugate [Gharamani and Beal 2001; Hoffman+ 2013]
- Not ↑, but can differentiate the log likelihood [Kucukelbir+ 2017]
- Not ↑, but can calculate the log likelihood [Ranganath+ 2014]
- Not ↑, but can sample from the model [Ranganath, Tran+ 2017]

The family of variational approximations



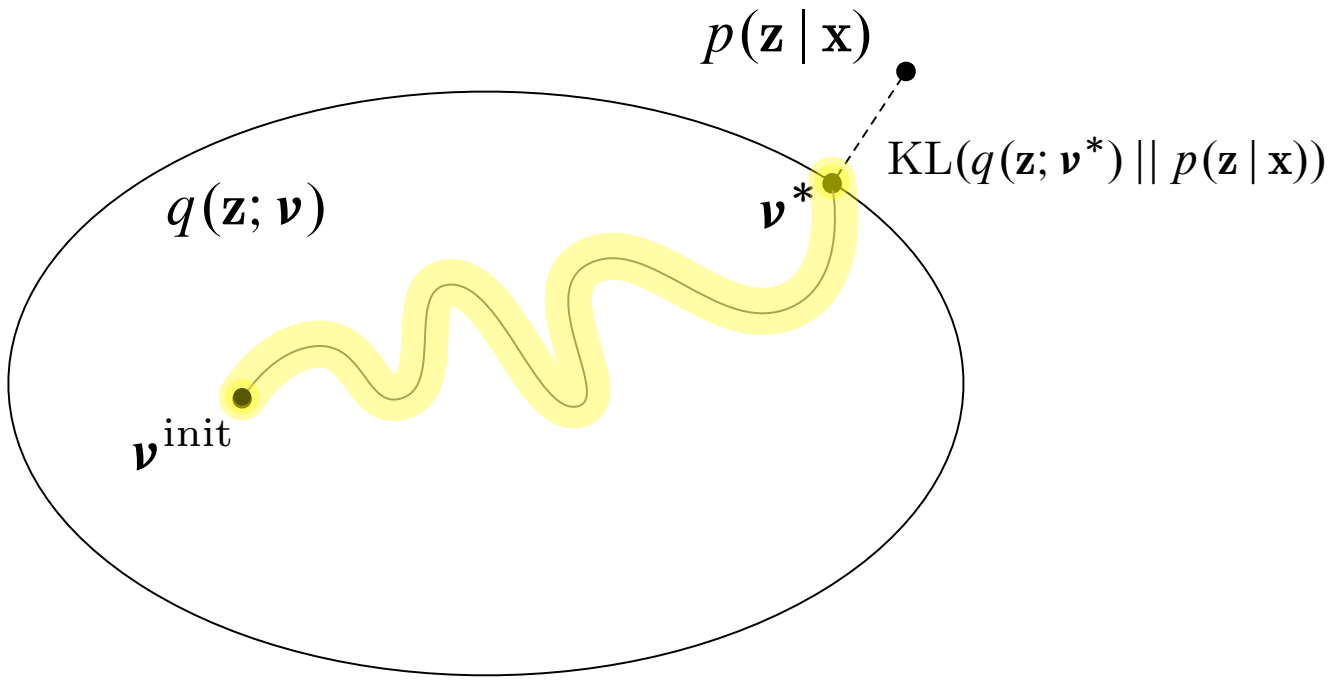
- Structured variational inference [Saul and Jordan 1996; Hoffman and Blei 2015]
- Variational models [Lawrence 2001; Ranganath+ 2015; Tran+ 2015]
- Amortized inference [Kingma and Welling 2014; Rezende+ 2014]
- Sequential Monte Carlo [Naesseth+ 2018; Maddison+ 2017; Le+ 2017]

The distance function



- Expectation propagation [Minka 2001]
- Belief propagation [Yedidia 2001]
- Operator variational inference [Ranganath+ 2016]
- χ-variational inference [Dieng+ 2017]

The algorithm



- SVI and structured SVI [Hoffman+ 2013; Hoffman and Blei 2015]
- Proximity VI [Altosaar+ 2018]
- SGD as VI [Mandt+ 2017]
- Adaptive rates, averaged and biased gradients, etc. [Many papers]

Some open research problems in VI

- **Theory**
  MCMC has been widely analyzed and studied; VI is less explored.

- **Optimization**
  Can we find better local optima? Can we accelerate convergence?

- **Alternative divergences**
  KL is chosen for convenience; can we use other divergences?

- **Better approximations**
  VI underestimates posterior variance. Can we do better?

- **Combining methods**
  Can VI be combined with other methods? Can flavors of VI be combined?

References (from our group)

- D. Blei, A. Kucukelbir, J. McAuliffe. **Variational inference: A review for statisticians**. Journal of American Statistical Association, 2017.

- M. Hoffman, D. Blei, C. Wang, J. Paisley. **Stochastic variational inference**. Journal of Machine Learning Research, 2013.

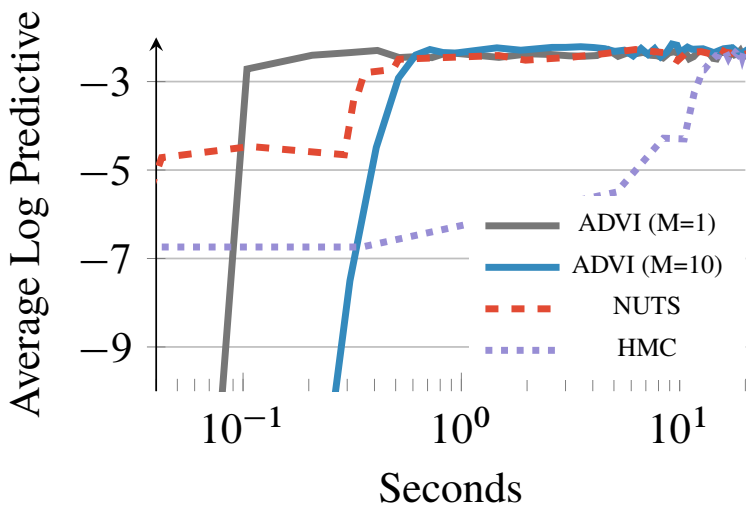
- R. Ranganath, S. Gerrish, D. Blei. **Black box variational inference**. Artificial Intelligence and Statistics, 2014.

- A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, D. Blei. **Automatic differentiation variational inference**. Journal of Machine Learning Research, 2017.

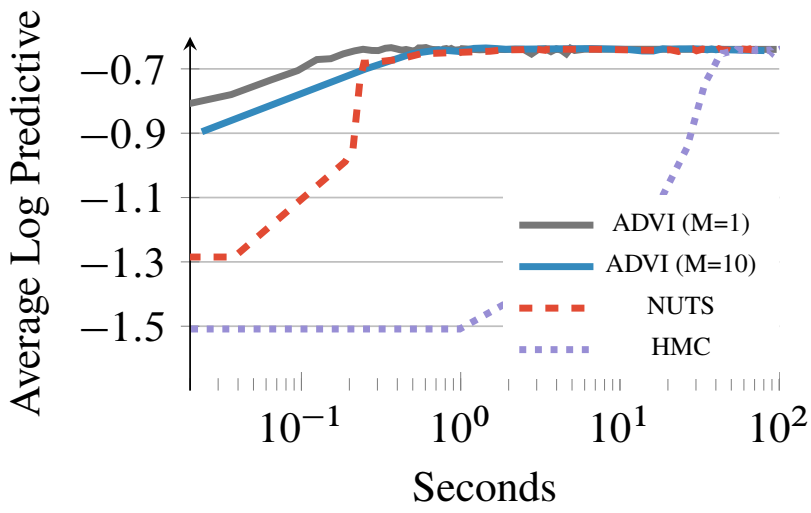
- Y. Wang and D. Blei. **Frequentist consistency of variational Bayes.** Journal of the American Statistical Association, to appear.

Extra slides

Should I be skeptical about variational inference?



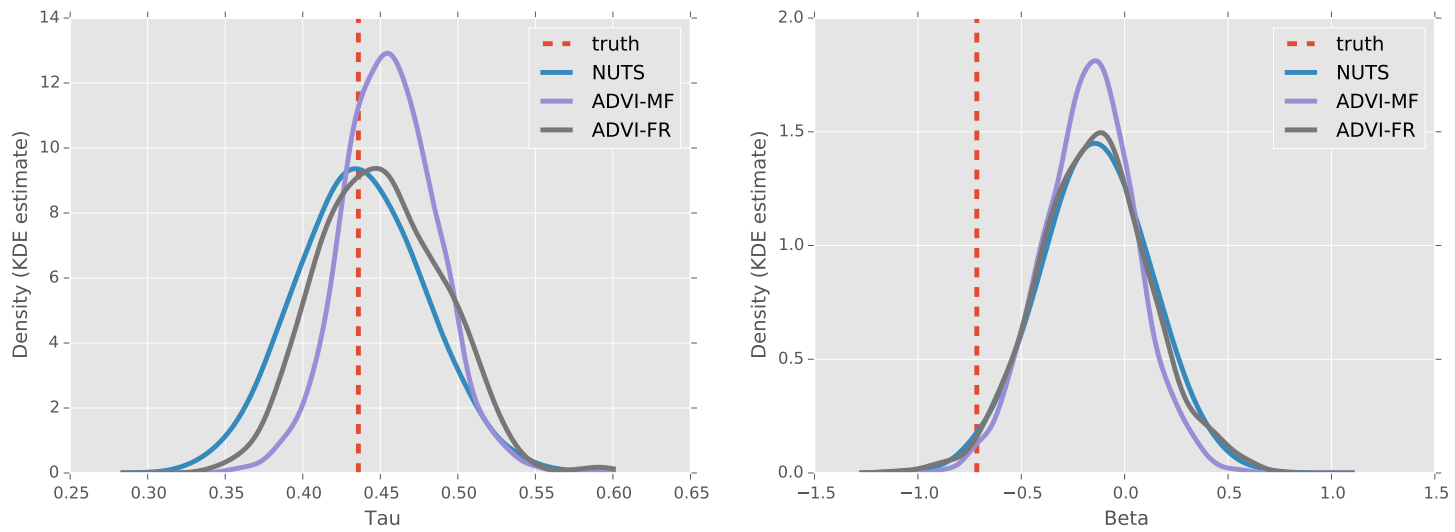
(a) Linear Regression with ARD



(b) Hierarchical Logistic Regression

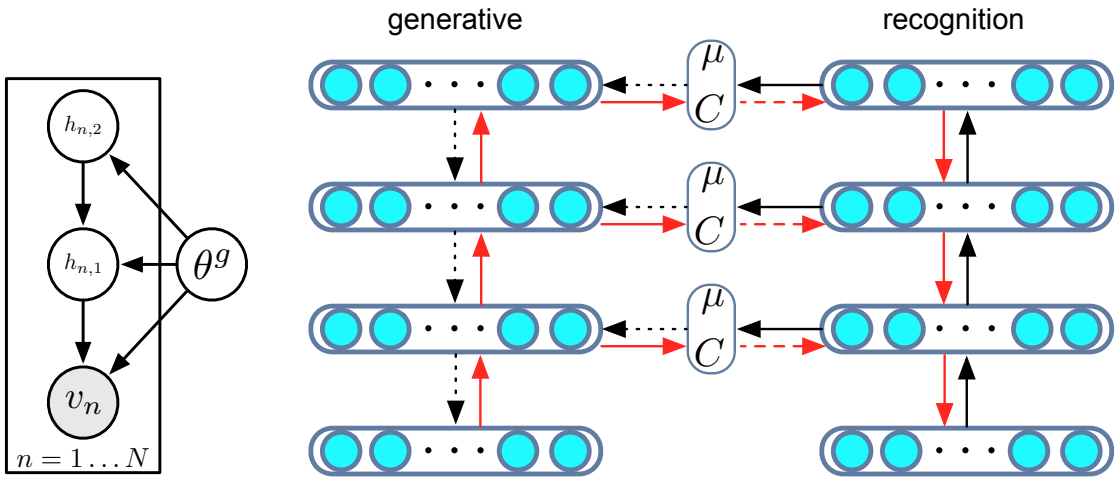
- **MCMC enjoys theoretical guarantees.**
- But they usually get to the same place. [Kucukelbir+ 2016]
- We need more theory about variational inference.

Should I be skeptical about variational inference?



- **Variational inference underestimates the variance of the posterior.**
- Relaxing the mean-field assumption can help.
- Here: A Poisson GLM [Giordano+ 2015]

Amortized inference and the variational autoencoder

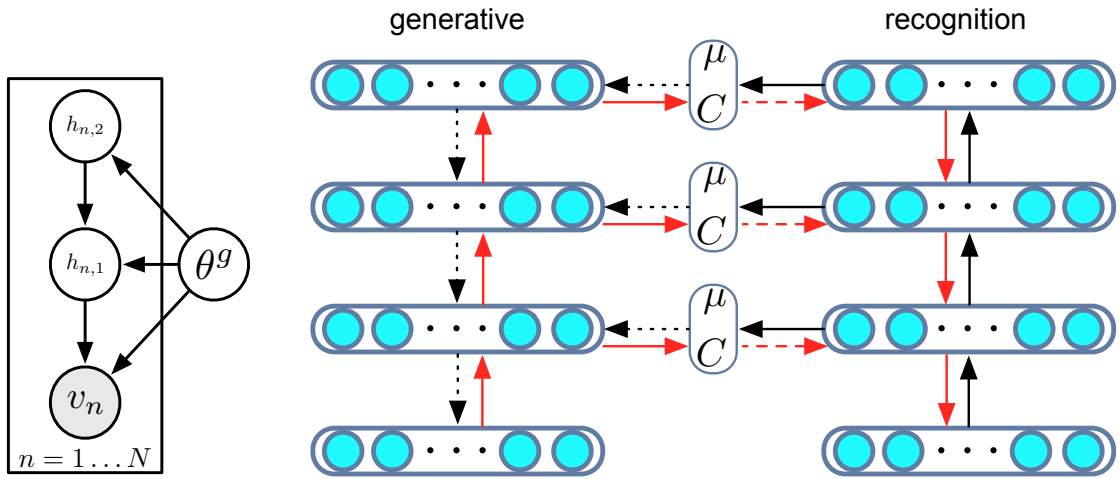


- **Amortization**:
  Variational parameters a parameterized function of the input $\nu_\eta(\mathbf{x})$
  [Gershman and Goodman 2014]

- This lets us "learn to infer." Test-time inference is fast.
  (Open question: There seems to be more to the story.)

- Plays nicely with autodifferentiation and reparameterization gradients.

Amortized inference and the variational autoencoder

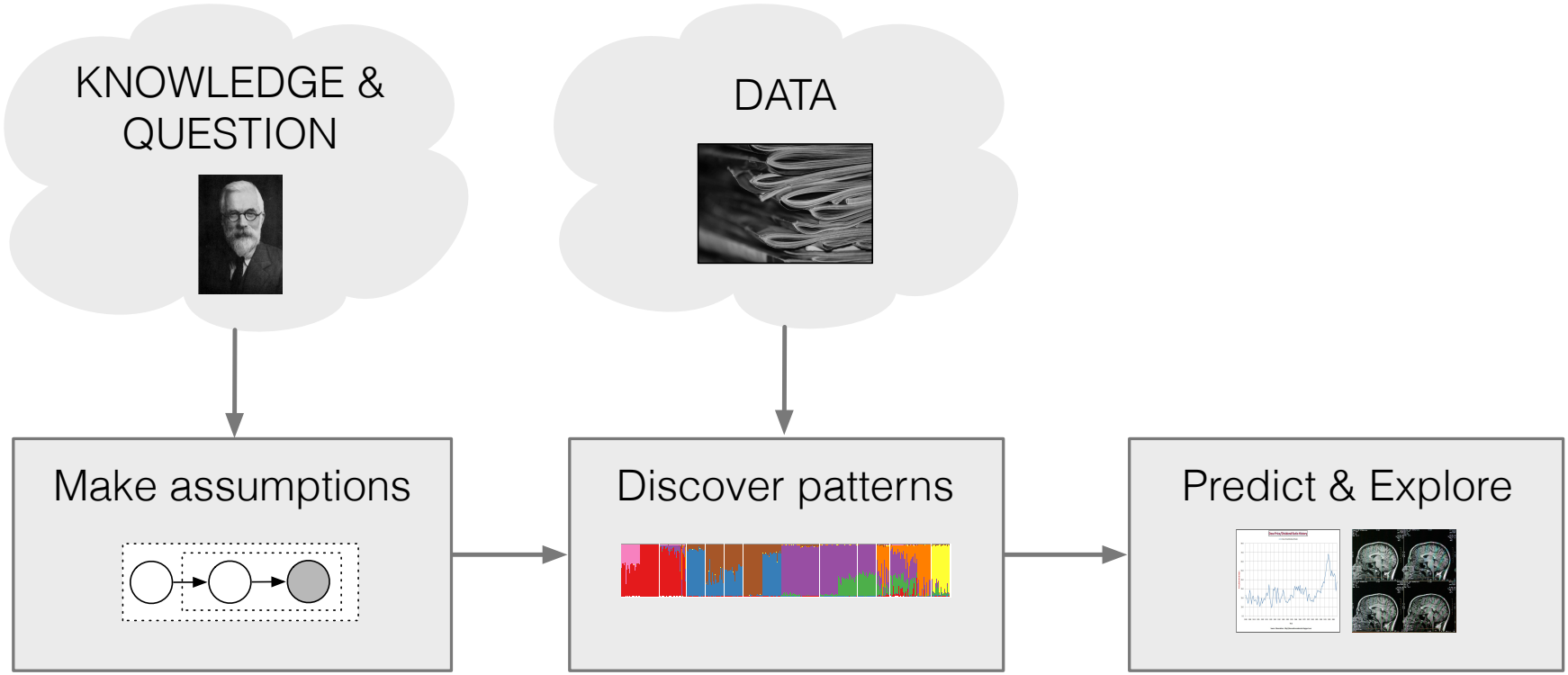


- Model: Deep generative model [Kingma+ 2013; Rezende+ 2014]

$$\mathbf{z}_i \sim \mathcal{N}(0, I)$$
$$\mathbf{x}_i \sim \mathcal{N}(f_\theta(\mathbf{z}_i), \sigma^2)$$

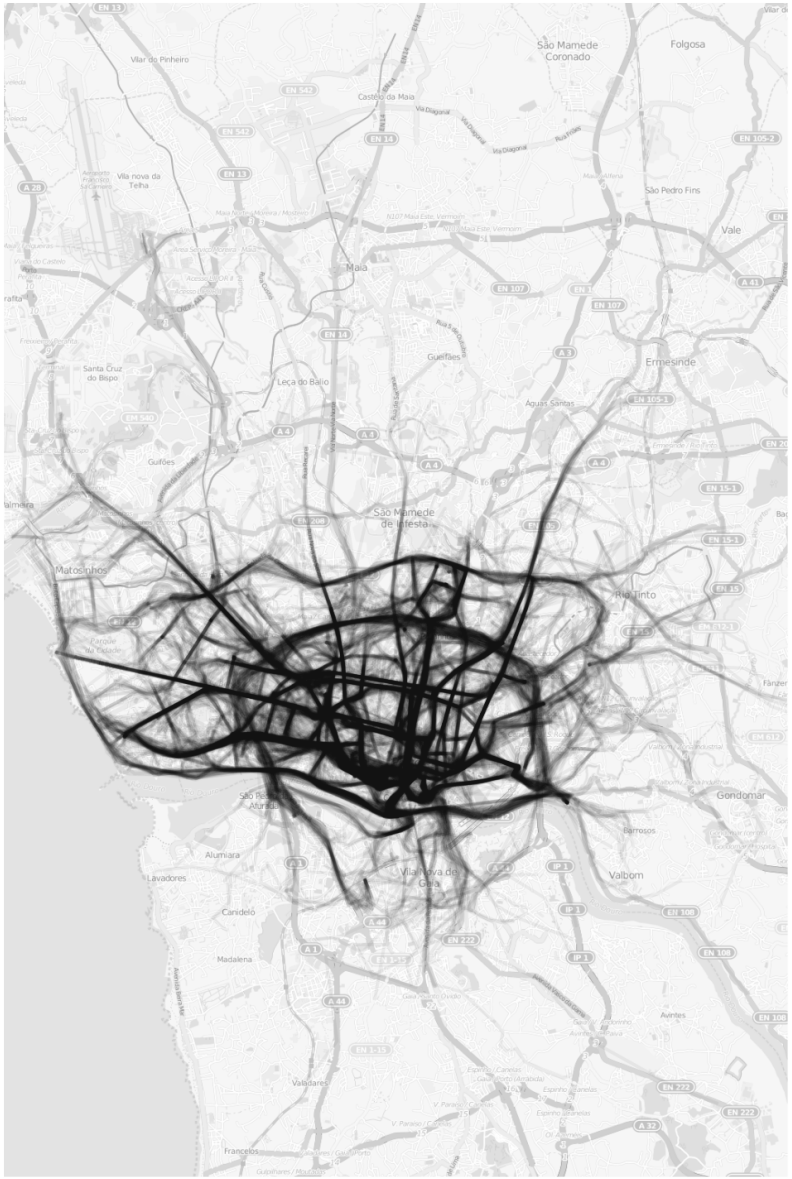
- Variational parameters $\boldsymbol{\nu}_\eta(\mathbf{x})$, also a deep neural network
- Algorithm:
  - Update η with reparameterization gradient
  - Update θ with gradient

Probabilistic programming



- Generative models are programs.
  Probabilistic programming takes this idea seriously.
- Languages for expressing models as programs
  Engines to compile models/programs to an inference executable.
- We can do this with BBVI.
  Key ideas: **autodifferentiation** and **stochastic optimization**.

Example: Taxi rides in Portugal



Example: Taxi rides in Portugal

- Data: 1.7M taxi rides in Porto, Portugal
- Multimodal probabilistic PCA with automatic relevance determination

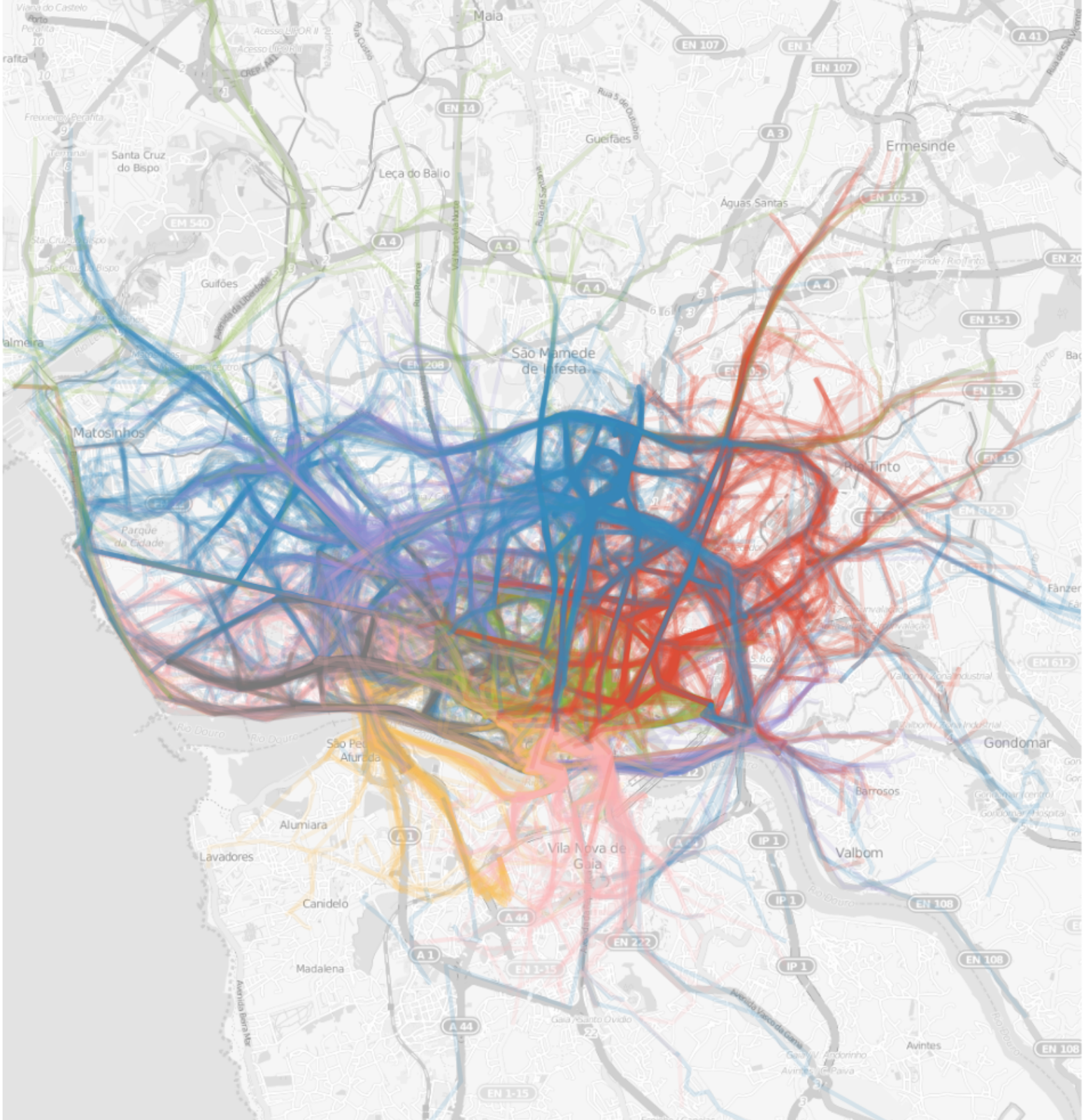
$$
\begin{aligned}
\sigma &\sim \text{log-normal}(0, 1) \\
\\
\alpha_j &\sim \text{inv-gamma}(1, 1) && j = 1 \dots k \\
w_{x,j} &\sim \mathcal{N}(0, \sigma \cdot \alpha_j) \\
w_{y,j} &\sim \mathcal{N}(0, \sigma \cdot \alpha_j) \\
\\
z_i &\sim \mathcal{N}(0, I) && i = 1 \dots n \\
x_i &\sim \mathcal{N}(w_x^\top z_i, \sigma) \\
y_i &\sim \mathcal{N}(w_y^\top z_i, \sigma)
\end{aligned}
$$

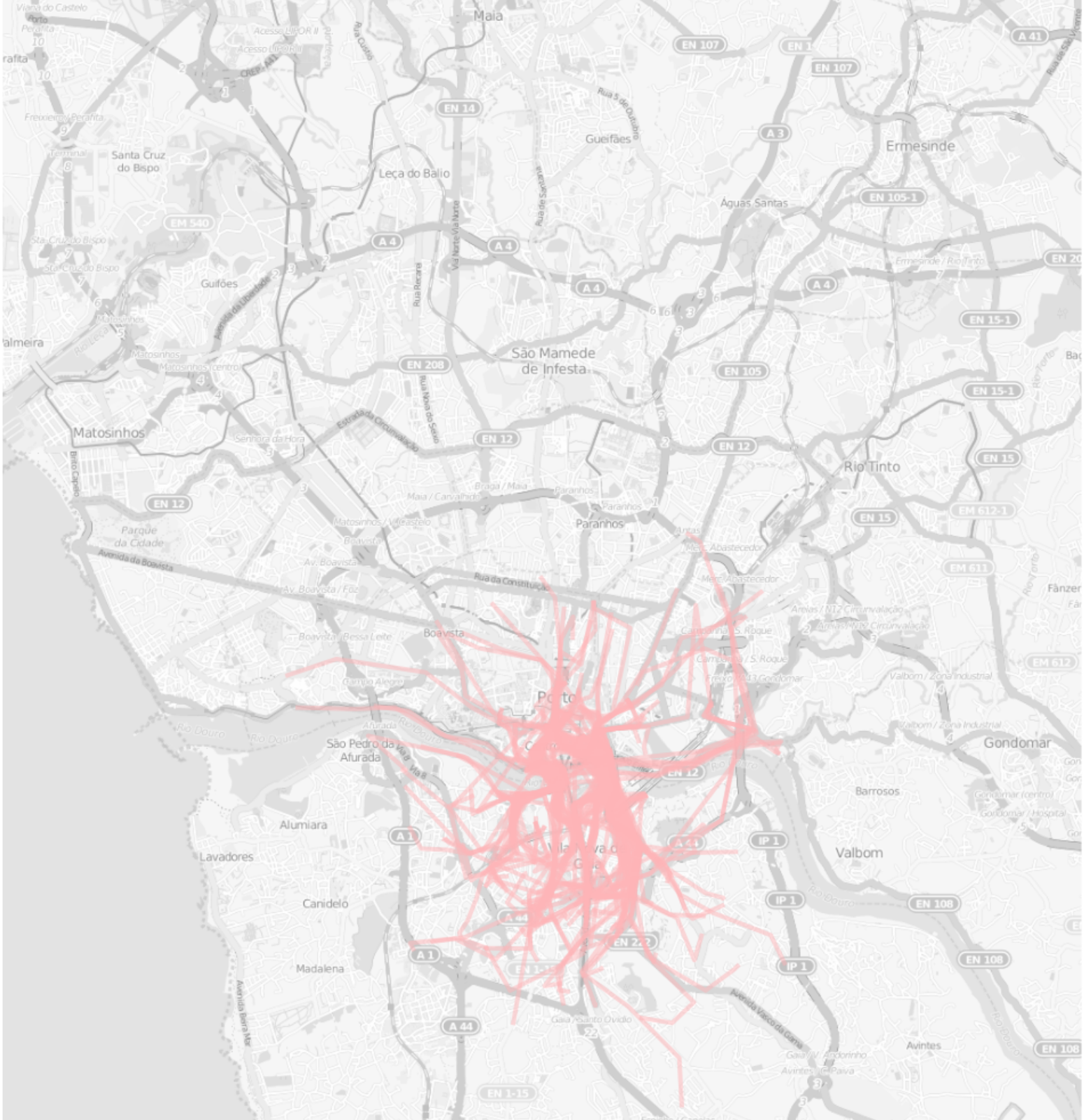
- The generative process looks like a program.

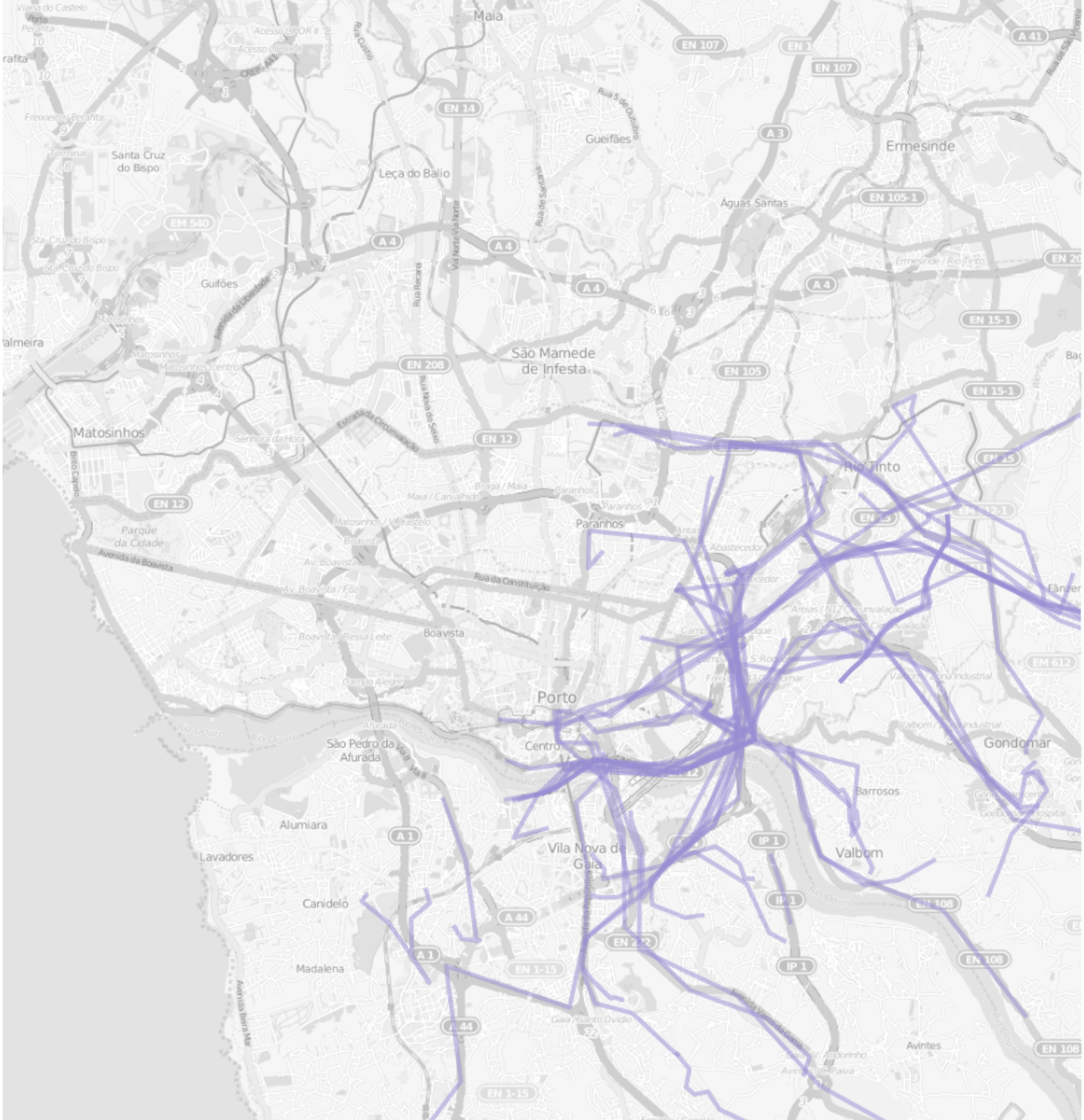
Supervised pPCA with ARD (Stan)

```
data {
  int<lower=0> N;              // number of data points in dataset
  int<lower=0> D;              // dimension
  int<lower=0> M;              // maximum dimension of latent space to cinsider

  vector[D] x[N];
  vector[N] y;
}

parameters {
  matrix[M,N] z;               // latent variable
  matrix[D,M] w_x;             // weights parameters
  vector[M]   w_y;             // variance parameter
  real<lower=0> sigma;
  vector<lower=0>[M] alpha;    // hyper-parameters on weights
}

model {
  // priors
  to_vector(z) ~ normal(0,1);
  for (d in 1:D)
    w_x[d] ~ normal(0, sigma * alpha);
  w_y ~ normal(0, sigma * alpha);

  sigma ~ lognormal(0,1);
  alpha ~ inv_gamma(1,1);

  // likelihood
  for (n in 1:N) {
    x[n] ~ normal(w_x * col(z, n), sigma);
    y[n] ~ normal(w_y' * col(z, n), sigma);
  }
}
```


Maia
Guelfães
Ermesinde
Santa Cruz do Bispo
Leça do Balio
Aguas Santas
Guifões
São Mamede de Infesta
Matosinhos
Rio Tinto
Gondomar
Barrosos
Alumiara
Lavadores
Vila Nova de Gaia
Valbom
Canidelo
Madalena
Avintes


Maia
Ermesinde
Guelfães
Leça do Balio
Santa Cruz do Bispo
Águas Santas
Guifões
São Mamede de Infesta
Matosinhos
Rio Tinto
Parque da Cidade
Gondomar
Barrosos
Alumiara
Lavadores
Vila Nova de Gaia
Valbom
Canidelo
Madalena
Avintes
EN 107
EN 14
A 3
A 41
EN 105-1
A 4
EM 540
EN 15-1
EN 208
EN 15
EM 612-1
EM 612
IP 1
EN 108
A 1
A 44
EN 222
EN 1-15
Rua 5 de Outubro
Rua de Santa Maria
Avenida da Liberdade
Rua Roberto
Via Norte-Via Norte
Avenida Barra Mar
Avenida Vieira de Castro
Rio Douro


Maia
Ermesinde
Santa Cruz do Bispo
Leça do Balio
Guelfães
Águas Santas
Guifões
São Mamede de Infesta
Matosinhos
Rio Tinto
Paranhos
Boavista
Parque da Cidade
São Pedro da Afurada
Alumiara
Lavadores
Canidelo
Madalena
Gondomar
Barrosos
Valbom
Avintes
EN 107
EN 14
A 3
A 4
EN 105-1
EN 540
EN 208
EN 105
EN 15-1
EN 12
EN 15
EM 612-1
EM 613
EM 612
A 1
IP 1
EN 108
A 44
EN 222
EN 1-15


Maia
Ermesinde
Santa Cruz do Bispo
Leça do Balio
Guelfães
Águas Santas
Guifões
São Mamede de Infesta
Matosinhos
Rio Tinto
Paranhos
Boavista
Porto
São Pedro da Afurada
Gondomar
Barrosos
Alumiara
Lavadores
Vila Nova de Gaia
Valbom
Canidelo
Madalena
Avintes
EN 14
EN 107
A 3
EN 105-1
A 4
EN 540
EN 208
EN 105
EN 15-1
EN 12
EN 15
A 1
IP 1
A 44
EN 108
EN 1-15
EN 612

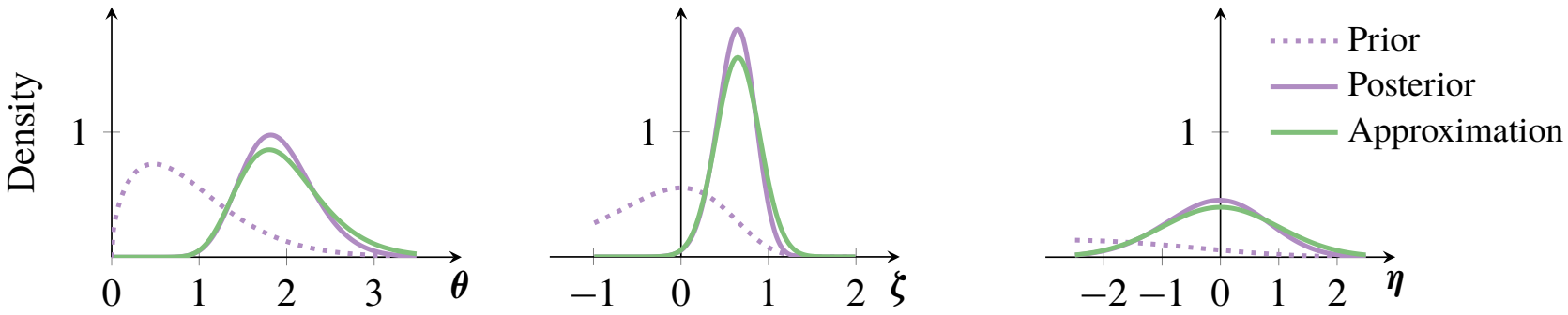
Exploring Taxi Trajectories

- Write down a supervised pPCA model (∼minutes).
- Use VI to fit the model in Stan (∼hours).
- Estimate latent representation z_i of each taxi ride (∼minutes).

- Write down a mixture model (∼minutes).
- Use VI to cluster the latent representations (∼minutes).

What would take weeks → a single day.

Automatic differentiation variational inference



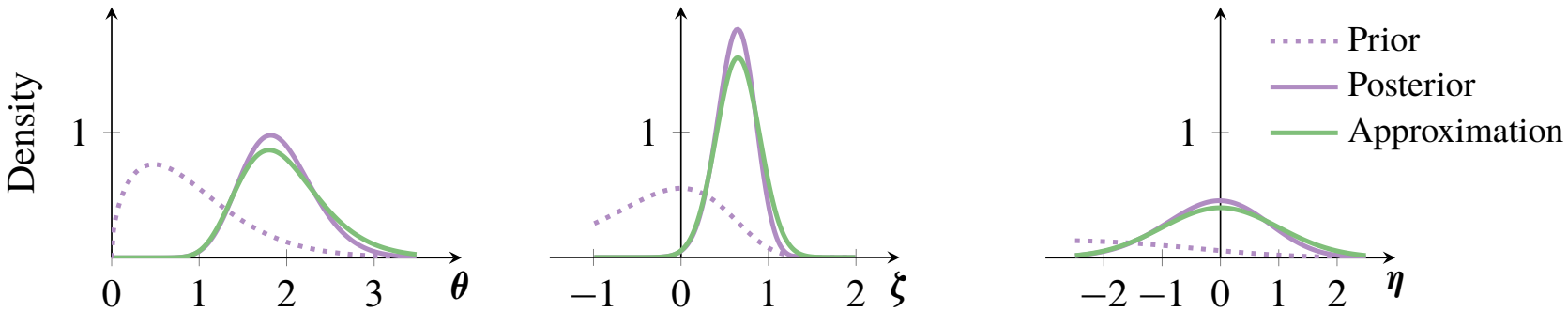
(a) Latent variable space **(b)** Real coordinate space **(c)** Standardized space

1. Transform the model.

- Transform from $p(\mathbf{z}, \mathbf{x})$ to $p(\zeta, \mathbf{x})$, where $\zeta \in \mathbb{R}^d$.
- The mapping is in the joint,

$$p(\zeta, \mathbf{x}) = p\left(\mathbf{x}, s(\zeta)\right) \left| \det J_s(\zeta) \right|.$$

Automatic differentiation variational inference



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

2. Redefine the variational problem.

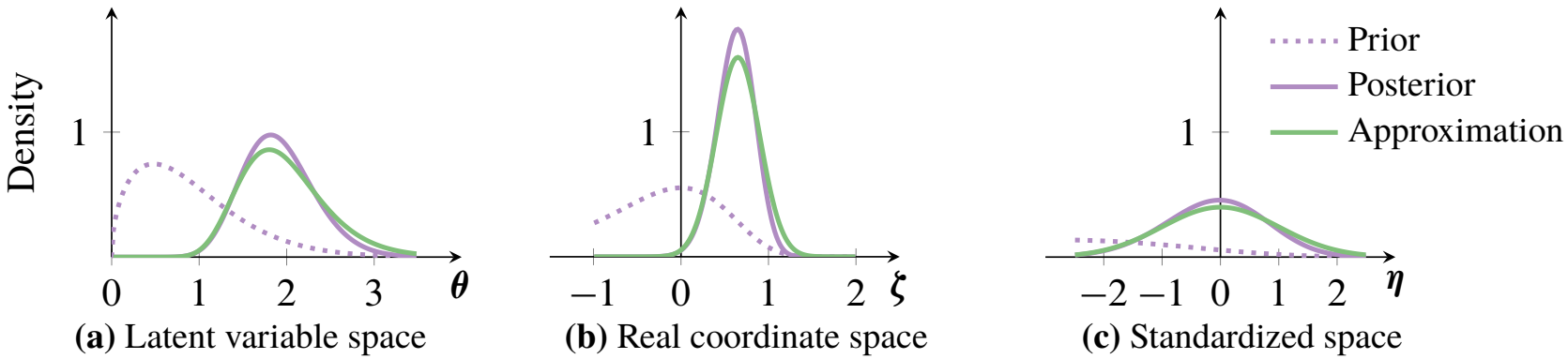
- The variational family is mean-field Gaussian

$$q(\boldsymbol{\zeta}\,;\,\boldsymbol{\nu}) = \prod_{k=1}^{K} \varphi\left(\zeta_k\,;\,\nu_k\right),$$

- The ELBO is

$$\mathscr{L} = \mathbb{E}_{q(\zeta)}\Big[\log p\left(\mathbf{x}, s(\boldsymbol{\zeta})\right) + \log\left|\det J_s(\boldsymbol{\zeta})\right|\Big] + \mathbb{H}(q)$$

Automatic differentiation variational inference

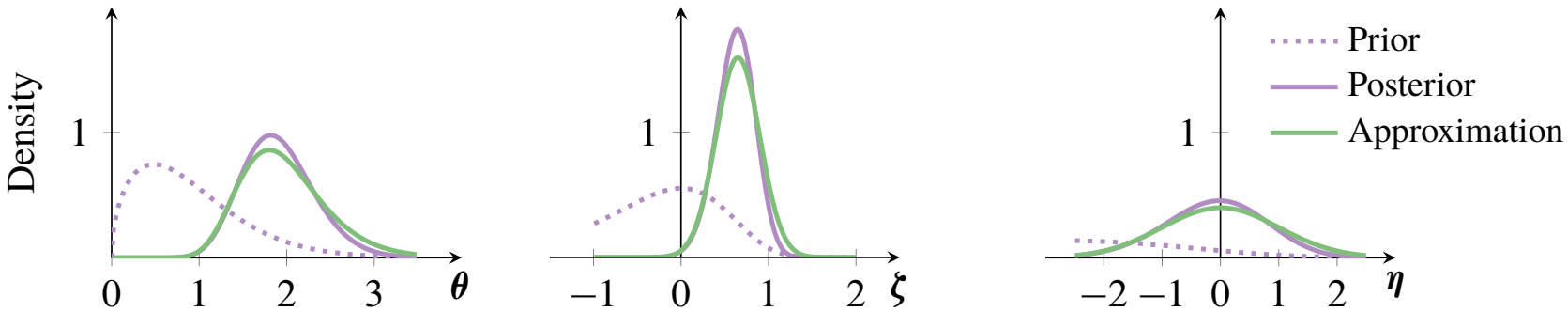


(a) Latent variable space **(b)** Real coordinate space **(c)** Standardized space

3. Use the reparameterization gradient

- Transform ζ using a standard normal $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ to a general normal.
- This is a second transformation of the original latent variable.
- Autodifferentiation handles the reparameterization gradient.

Automatic differentiation variational inference



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

Implementation

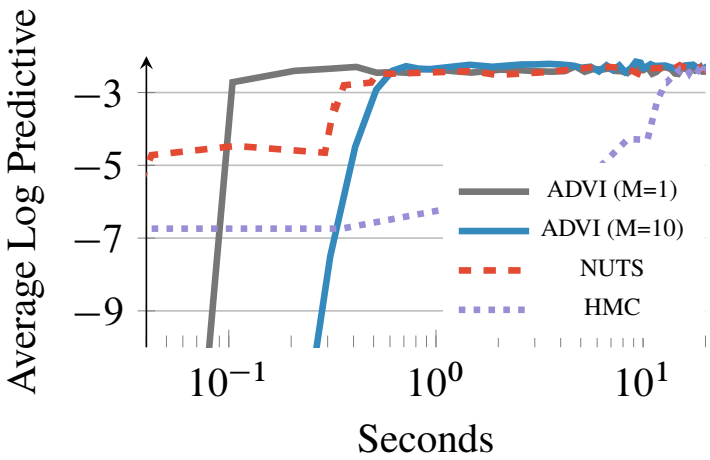
- Stan automates going from $\log p(\mathbf{x}, \mathbf{z})$ to

$$\log p(\mathbf{x}, s(\zeta)) + \log \left| \det J_s(\zeta) \right|$$

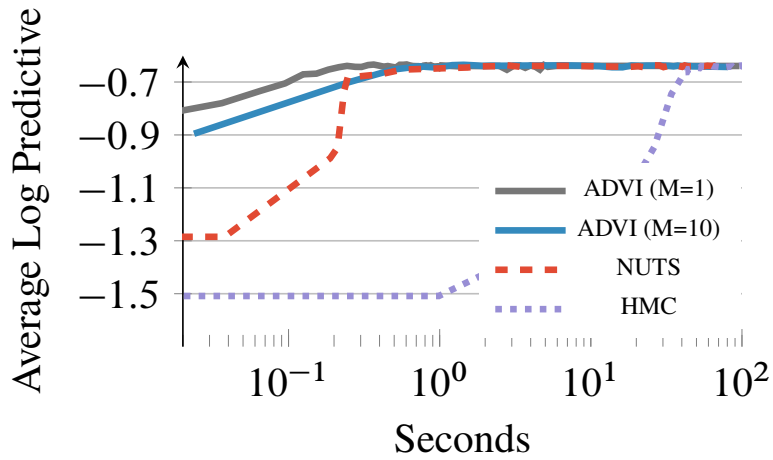
$$\nabla_\zeta (\log p(\mathbf{x}, s(\zeta)) + \log \left| \det J_s(\zeta) \right|)$$

- Use reparameterization BBVI (with the Gaussian transformation)
- Can incorporate SVI and other innovations

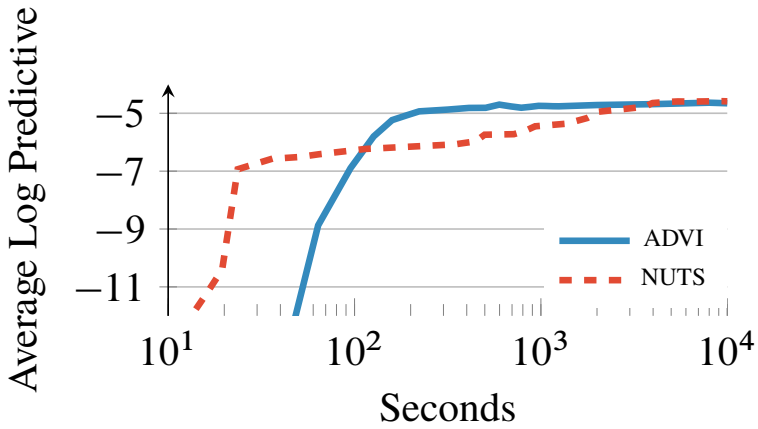
Some benchmarks



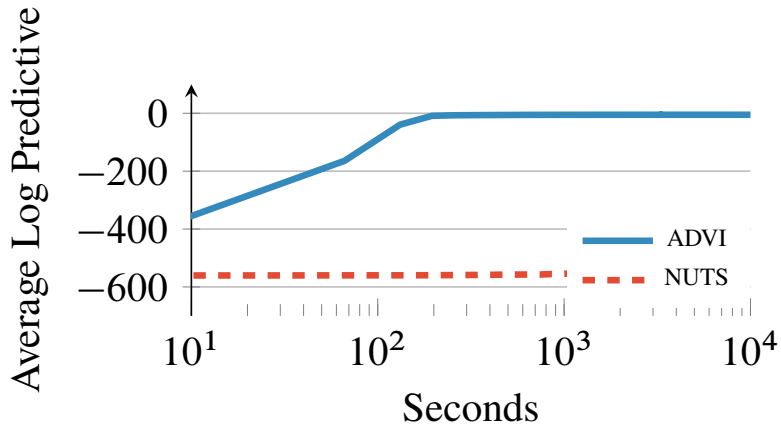
(a) Linear Regression with ARD



(b) Hierarchical Logistic Regression



(a) Gamma Poisson Predictive Likelihood



(b) Dirichlet Exponential Predictive Likelihood

[Kucukelbir+ 2016]

Score gradient

$$\nabla_{\nu}\mathcal{L} = \mathbb{E}_{q(\mathbf{z};\nu)}[\underbrace{\nabla_{\nu}\log q(\mathbf{z};\nu)}_{\text{score function}}\ \underbrace{(\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\nu))}_{\text{instantaneous ELBO}}]$$

Reparameterization gradient

$$\nabla\mathcal{L} = \mathbb{E}_{s(\epsilon)}\left[\underbrace{\nabla_{\mathbf{z}}[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\nu)]}_{\text{gradient of instananeous ELBO}}\ \underbrace{\nabla_{\nu}t(\epsilon,\nu)}_{\text{gradient of transformation}}\right]$$

A recipe for variational inference

$$p(\mathbf{z}, \mathbf{x})$$

Posit a model, a joint distribution of hidden and observed variables.

A recipe for variational inference

$$q(\mathbf{z}; \boldsymbol{\nu})$$

Choose the variational family, distributions of the hidden variables.

A recipe for variational inference

$$\mathcal{L}(\boldsymbol{\nu}) = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\boldsymbol{\nu})]$$

Write the ELBO, the objective function for finding a $q(\mathbf{z};\boldsymbol{\nu})$ close to $p(\mathbf{z}|\mathbf{x})$.

A recipe for variational inference

$$\mathscr{L}(v) = x v^2 + \log v \quad \text{(example)}$$

Integrate: The ELBO is a function of data and variational parameters.

A recipe for variational inference

$$\nabla_v \mathcal{L}(\boldsymbol{v}) = 2x\boldsymbol{v} + 1/\boldsymbol{v} \quad \text{(example)}$$

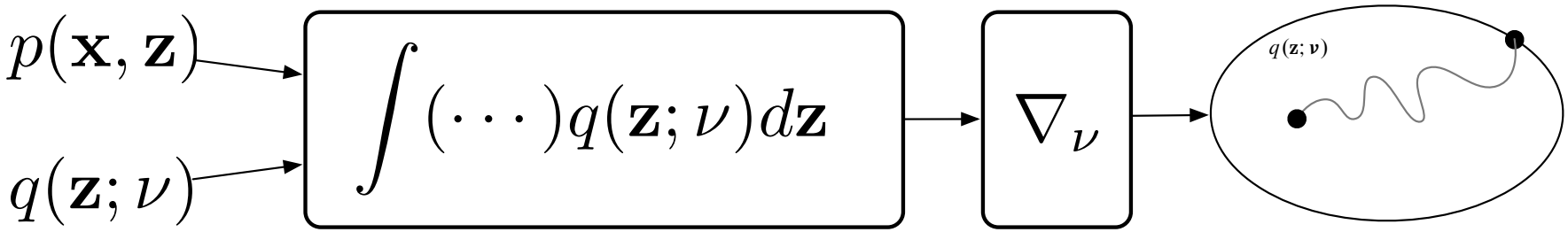
Take derivatives.

A recipe for variational inference

$$\boldsymbol{\nu}_{t+1} = \boldsymbol{\nu}_t + \rho_t \nabla_{\boldsymbol{\nu}} \mathcal{L}$$

Optimize.

A recipe for variational inference



1. Posit a model
2. Choose a variational family
3. Integrate (calculate the ELBO)
4. Take derivatives
5. Optimize

Bayesian logistic regression

- Data are pairs (x_i, y_i)
  - x_i is a covariate
  - $y_i \in \{0, 1\}$ is a binary label
  - z are the regression coefficients

- Conditional on covariates, Bayesian LR posits a generative process of labels

$$z \sim N(0, 1)$$
$$y_i \,|\, x_i, z \sim \text{Bernoulli}(\sigma(z x_i)),$$

  where $\sigma(\cdot)$ is the logistic function, mapping reals to $(0, 1)$.

VI for Bayesian logistic regression

- Consider one data point (x, y).
- Our goal is to approximate the posterior coefficient $p(z \mid x, y)$.
- The variational family $q(z; \boldsymbol{\nu})$ is a normal; $\boldsymbol{\nu} = (\mu, \sigma^2)$
- The ELBO is

$$\mathcal{L}(\mu, \sigma^2) = \mathbb{E}_q[\log p(z) + \log p(y \mid x, z) - \log q(z)]$$

VI for Bayesian logistic regression

$$\mathcal{L}(\mu, \sigma^2) \quad = \quad \mathbb{E}_q[\log p(z) - \log q(z) + \log p(y \mid x, z)]$$

VI for Bayesian logistic regression

$$
\begin{aligned}
\mathcal{L}(\mu, \sigma^2) &= \mathbb{E}_q[\log p(z) - \log q(z) + \log p(y \mid x, z)] \\
&= -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + \mathbb{E}_q[\log p(y \mid x, z)] + C
\end{aligned}
$$

VI for Bayesian logistic regression

$$
\begin{aligned}
\mathcal{L}(\mu, \sigma^2) &= \mathbb{E}_q[\log p(z) - \log q(z) + \log p(y \mid x, z)] \\
&= -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + \mathbb{E}_q[\log p(y \mid x, z)] + C \\
&= -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + \mathbb{E}_q[yxz - \log(1 + \exp(xz))]
\end{aligned}
$$

VI for Bayesian logistic regression

$$
\begin{aligned}
\mathcal{L}(\mu, \sigma^2) &= \mathbb{E}_q[\log p(z) - \log q(z) + \log p(y \mid x, z)] \\
&= -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + \mathbb{E}_q[\log p(y \mid x, z)] + C \\
&= -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + \mathbb{E}_q[yxz - \log(1 + \exp(xz))] \\
&= -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + yx\mu - \mathbb{E}_q[\log(1 + \exp(xz))]
\end{aligned}
$$

VI for Bayesian logistic regression

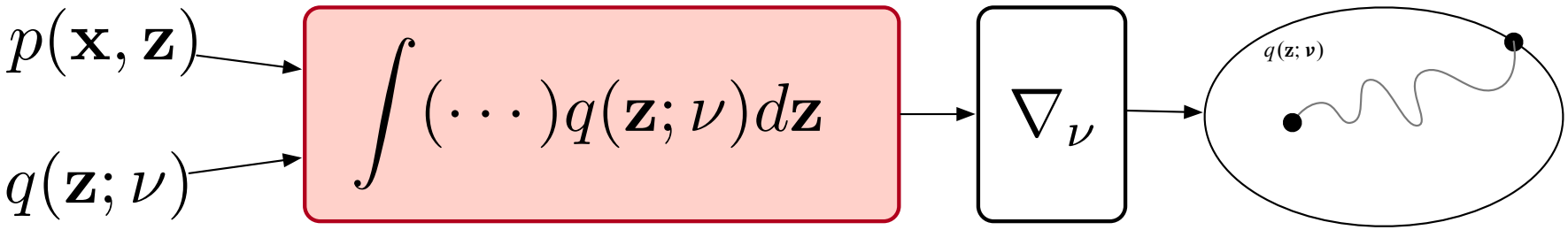
$$
\begin{aligned}
\mathcal{L}(\mu, \sigma^2) &= \mathbb{E}_q[\log p(z) - \log q(z) + \log p(y \mid x, z)] \\
&= -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + \mathbb{E}_q[\log p(y \mid x, z)] + C \\
&= -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + \mathbb{E}_q[yxz - \log(1 + \exp(xz))] \\
&= -\frac{1}{2}(\mu^2 + \sigma^2) + \frac{1}{2}\log \sigma^2 + yx\mu - \mathbb{E}_q[\log(1 + \exp(xz))]
\end{aligned}
$$

We are stuck—we cannot analytically take the expectation.

Options?

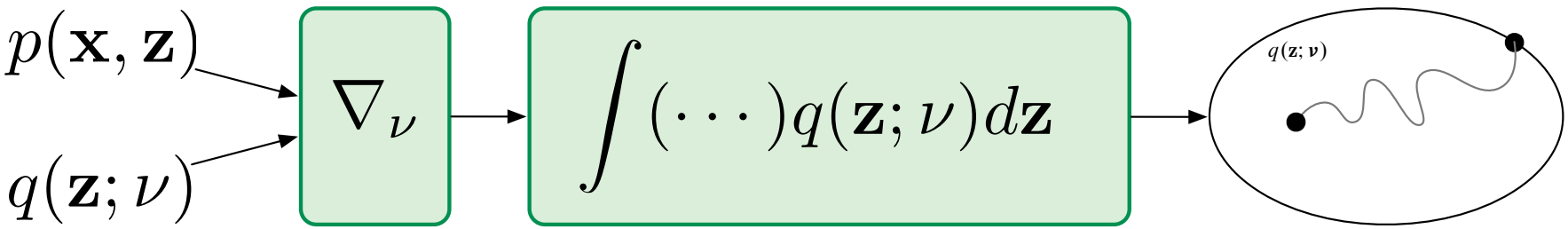
- Derive a model-specific bound
  [Jordan and Jaakola 1996], [Braun and McAuliffe 2008], others
- Use other approximations (that require model-specific analysis)
  [Wang and Blei 2013], [Knowles and Minka 2011]
- But neither satisfies the *black box criteria*.

The problem with the VI recipe



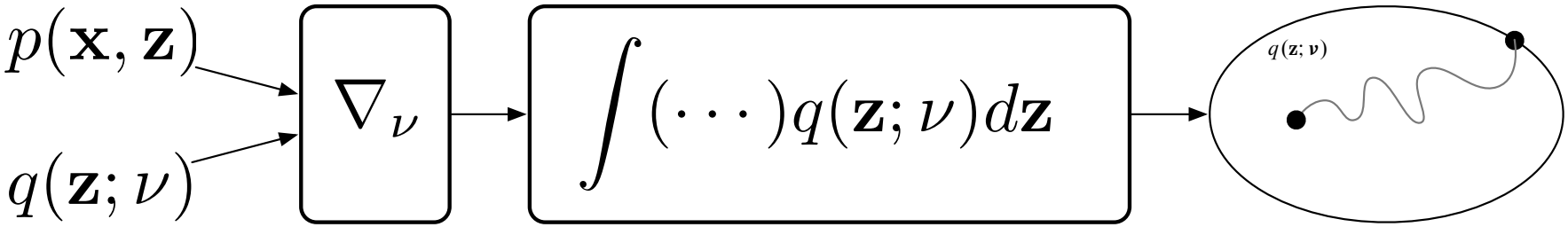
The integral is hard to take.

Solution: Swap integration and differentiation



Swap! Now we can use MC gradients and stochastic optimization.

The new recipe



- This is the key idea behind modern methods in variational inference
- It has enabled score gradients, reparameterization gradients, amortized inference, probabilistic programming, complex variational families, and alternative divergences.
- How do we reverse differentiation and integration?

Reversing the gradient and the expectation

- Denote the *instantaneous ELBO*

$$g(\mathbf{z}, \boldsymbol{\nu}) = \log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\nu}).$$

- The ELBO is its expectation

$$\mathscr{L} = \mathbb{E}_q \left[g(\mathbf{z}, \boldsymbol{\nu}) \right] = \int q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) d\mathbf{z}$$

- We want to calculate $\nabla_{\boldsymbol{\nu}} \mathscr{L}$.

Reversing the gradient and the expectation

Fact:

$$\nabla_{\boldsymbol{\nu}} q(\mathbf{z}; \boldsymbol{\nu}) = q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\boldsymbol{\nu}} \log q(\mathbf{z}; \boldsymbol{\nu}).$$

Reversing the gradient and the expectation

Fact:

$$\nabla_{\nu} q(\mathbf{z}; \boldsymbol{\nu}) = q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} \log q(\mathbf{z}; \boldsymbol{\nu}).$$

With this,

$$\nabla_{\nu} \mathcal{L} \quad = \quad \nabla_{\nu} \int q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) d\mathbf{z}$$

Reversing the gradient and the expectation

Fact:

$$\nabla_{\nu} q(\mathbf{z}; \boldsymbol{\nu}) = q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} \log q(\mathbf{z}; \boldsymbol{\nu}).$$

With this,

$$\begin{aligned} \nabla_{\nu} \mathcal{L} &= \nabla_{\nu} \int q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) d\mathbf{z} \\ &= \int \nabla_{\nu} q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) + q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} g(\mathbf{z}, \boldsymbol{\nu}) d\mathbf{z} \end{aligned}$$

Reversing the gradient and the expectation

Fact:

$$\nabla_{\nu} q(\mathbf{z}; \boldsymbol{\nu}) = q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} \log q(\mathbf{z}; \boldsymbol{\nu}).$$

With this,

$$\begin{aligned}
\nabla_{\nu} \mathcal{L} &= \nabla_{\nu} \int q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) d\mathbf{z} \\
&= \int \nabla_{\nu} q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) + q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} g(\mathbf{z}, \boldsymbol{\nu}) d\mathbf{z} \\
&= \int q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} \log q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) + q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} g(\mathbf{z}, \boldsymbol{\nu}) d\mathbf{z}
\end{aligned}$$

Reversing the gradient and the expectation

Fact:

$$\nabla_{\nu} q(\mathbf{z}; \boldsymbol{\nu}) = q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} \log q(\mathbf{z}; \boldsymbol{\nu}).$$

With this,

$$\begin{aligned}
\nabla_{\nu} \mathcal{L} &= \nabla_{\nu} \int q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) d\mathbf{z} \\
&= \int \nabla_{\nu} q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) + q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} g(\mathbf{z}, \boldsymbol{\nu}) d\mathbf{z} \\
&= \int q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} \log q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) + q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} g(\mathbf{z}, \boldsymbol{\nu}) d\mathbf{z} \\
&= \mathbb{E}_{q(\mathbf{z}; \boldsymbol{\nu})} [\nabla_{\nu} \log q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) + \nabla_{\nu} g(\mathbf{z}, \boldsymbol{\nu})]
\end{aligned}$$

Reversing the gradient and the expectation

Fact:

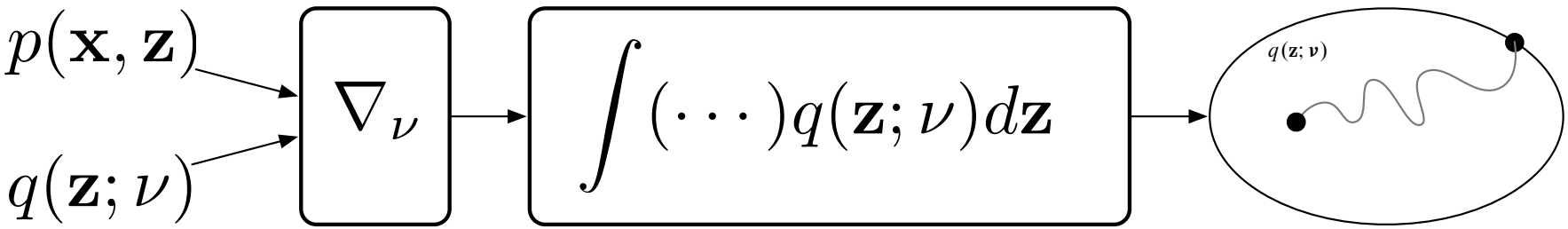
$$\nabla_{\nu} q(\mathbf{z}; \boldsymbol{\nu}) = q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} \log q(\mathbf{z}; \boldsymbol{\nu}).$$

With this,

$$\begin{aligned}
\nabla_{\nu} \mathcal{L} &= \nabla_{\nu} \int q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) d\mathbf{z} \\
&= \int \nabla_{\nu} q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) + q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} g(\mathbf{z}, \boldsymbol{\nu}) d\mathbf{z} \\
&= \int q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} \log q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) + q(\mathbf{z}; \boldsymbol{\nu}) \nabla_{\nu} g(\mathbf{z}, \boldsymbol{\nu}) d\mathbf{z} \\
&= \mathbb{E}_{q(\mathbf{z}; \boldsymbol{\nu})} [\nabla_{\nu} \log q(\mathbf{z}; \boldsymbol{\nu}) g(\mathbf{z}, \boldsymbol{\nu}) + \nabla_{\nu} g(\mathbf{z}, \boldsymbol{\nu})]
\end{aligned}$$

We have written the gradient as an expectation.

Black box variational inference



- Derive the score gradient
- Derive the reparameterization gradient

The score gradient

- The black-box gradient is

$$\nabla_{\boldsymbol{\nu}} \mathscr{L} = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\nabla_{\boldsymbol{\nu}} \log q(\mathbf{z};\boldsymbol{\nu}) g(\mathbf{z},\boldsymbol{\nu}) + \nabla_{\boldsymbol{\nu}} g(z,\boldsymbol{\nu})]$$

The score gradient

- The black-box gradient is

$$\nabla_{\boldsymbol{\nu}} \mathscr{L} = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\nabla_{\boldsymbol{\nu}} \log q(\mathbf{z};\boldsymbol{\nu}) g(\mathbf{z},\boldsymbol{\nu}) + \nabla_{\boldsymbol{\nu}} g(z,\boldsymbol{\nu})]$$

- Simplify the second term

$$\mathbb{E}_q[\nabla_{\boldsymbol{\nu}} g(\mathbf{z},\boldsymbol{\nu})] = \mathbb{E}_q[\nabla_{\boldsymbol{\nu}} \log q(\mathbf{z};\boldsymbol{\nu})] = 0$$

The score gradient

- The black-box gradient is

$$\nabla_{\boldsymbol{\nu}} \mathcal{L} = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\nabla_{\boldsymbol{\nu}} \log q(\mathbf{z};\boldsymbol{\nu}) g(\mathbf{z},\boldsymbol{\nu}) + \nabla_{\boldsymbol{\nu}} g(z,\boldsymbol{\nu})]$$

- Simplify the second term

$$\mathbb{E}_q[\nabla_{\boldsymbol{\nu}} g(\mathbf{z},\boldsymbol{\nu})] = \mathbb{E}_q[\nabla_{\boldsymbol{\nu}} \log q(\mathbf{z};\boldsymbol{\nu})] = 0$$

- This gives the score gradient

$$\nabla_{\boldsymbol{\nu}} \mathcal{L} = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\nabla_{\boldsymbol{\nu}} \log q(\mathbf{z};\boldsymbol{\nu})(\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\boldsymbol{\nu}))]$$

The reparameterization gradient

- Assume that we can express the variational distribution with a transformation, where

$$\begin{aligned}\boldsymbol{\epsilon} &\sim s(\boldsymbol{\epsilon}) \\ \mathbf{z} &= t(\boldsymbol{\epsilon}, \boldsymbol{\nu}) \\ &\rightarrow \mathbf{z} \sim q(\mathbf{z}; \boldsymbol{\nu})\end{aligned}$$

- For example,

$$\begin{aligned}\epsilon &\sim \text{Normal}(0, 1) \\ z &= \epsilon\sigma + \mu \\ &\rightarrow z \sim \text{Normal}(\mu, \sigma^2)\end{aligned}$$

- Also assume $\log p(\mathbf{x}, \mathbf{z})$ and $\log q(\mathbf{z})$ are differentiable with respect to $\mathbf{z}$

The reparameterization gradient

- The black box gradient is

$$\nabla_{\boldsymbol{\nu}} \mathscr{L} = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\nabla_{\boldsymbol{\nu}} \log q(\mathbf{z};\boldsymbol{\nu}) g(\mathbf{z},\boldsymbol{\nu}) + \nabla_{\boldsymbol{\nu}} g(\mathbf{z},\boldsymbol{\nu})]$$

The reparameterization gradient

- The black box gradient is

$$\nabla_{\boldsymbol{\nu}} \mathcal{L} = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\nu})}[\nabla_{\boldsymbol{\nu}} \log q(\mathbf{z};\boldsymbol{\nu}) g(\mathbf{z},\boldsymbol{\nu}) + \nabla_{\boldsymbol{\nu}} g(\mathbf{z},\boldsymbol{\nu})]$$

- Rewrite using using $\mathbf{z} = t(\boldsymbol{\epsilon},\boldsymbol{\nu})$,

$$\nabla_{\boldsymbol{\nu}} \mathcal{L} = \mathbb{E}_{s(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\nu}} \log s(\boldsymbol{\epsilon}) g(t(\boldsymbol{\epsilon},\boldsymbol{\nu}),\boldsymbol{\nu}) + \nabla_{\boldsymbol{\nu}} g(t(\boldsymbol{\epsilon},\boldsymbol{\nu}),\boldsymbol{\nu})]$$

The reparameterization gradient

- The black box gradient is

$$\nabla_{\nu} \mathcal{L} = \mathbb{E}_{q(\mathbf{z};\nu)}[\nabla_{\nu} \log q(\mathbf{z};\nu) g(\mathbf{z},\nu) + \nabla_{\nu} g(\mathbf{z},\nu)]$$

- Rewrite using using $\mathbf{z} = t(\epsilon, \nu)$,

$$\nabla_{\nu} \mathcal{L} = \mathbb{E}_{s(\epsilon)}[\nabla_{\nu} \log s(\epsilon) g(t(\epsilon,\nu),\nu) + \nabla_{\nu} g(t(\epsilon,\nu),\nu)]$$

- Note that $\nabla_{\nu} \log s(\epsilon) = 0$. Now use the chain rule:

$$\begin{aligned}
\nabla_{\nu} \mathcal{L} &= \mathbb{E}_{s(\epsilon)}[\nabla_{\nu} g(t(\epsilon,\nu),\nu)] \\
&= \mathbb{E}_{s(\epsilon)}[\nabla_{\mathbf{z}}[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\nu)] \nabla_{\nu} t(\epsilon,\nu) - \nabla_{\nu} \log q(\mathbf{z};\nu)] \\
&= \mathbb{E}_{s(\epsilon)}[\nabla_{\mathbf{z}}[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z};\nu)] \nabla_{\nu} t(\epsilon,\nu)]
\end{aligned}$$

This is the reparameterization gradient.

[Glasserman 1991; Fu 2006; Kingma+ 2014; Rezende+ 2014; Titsias+ 2014]