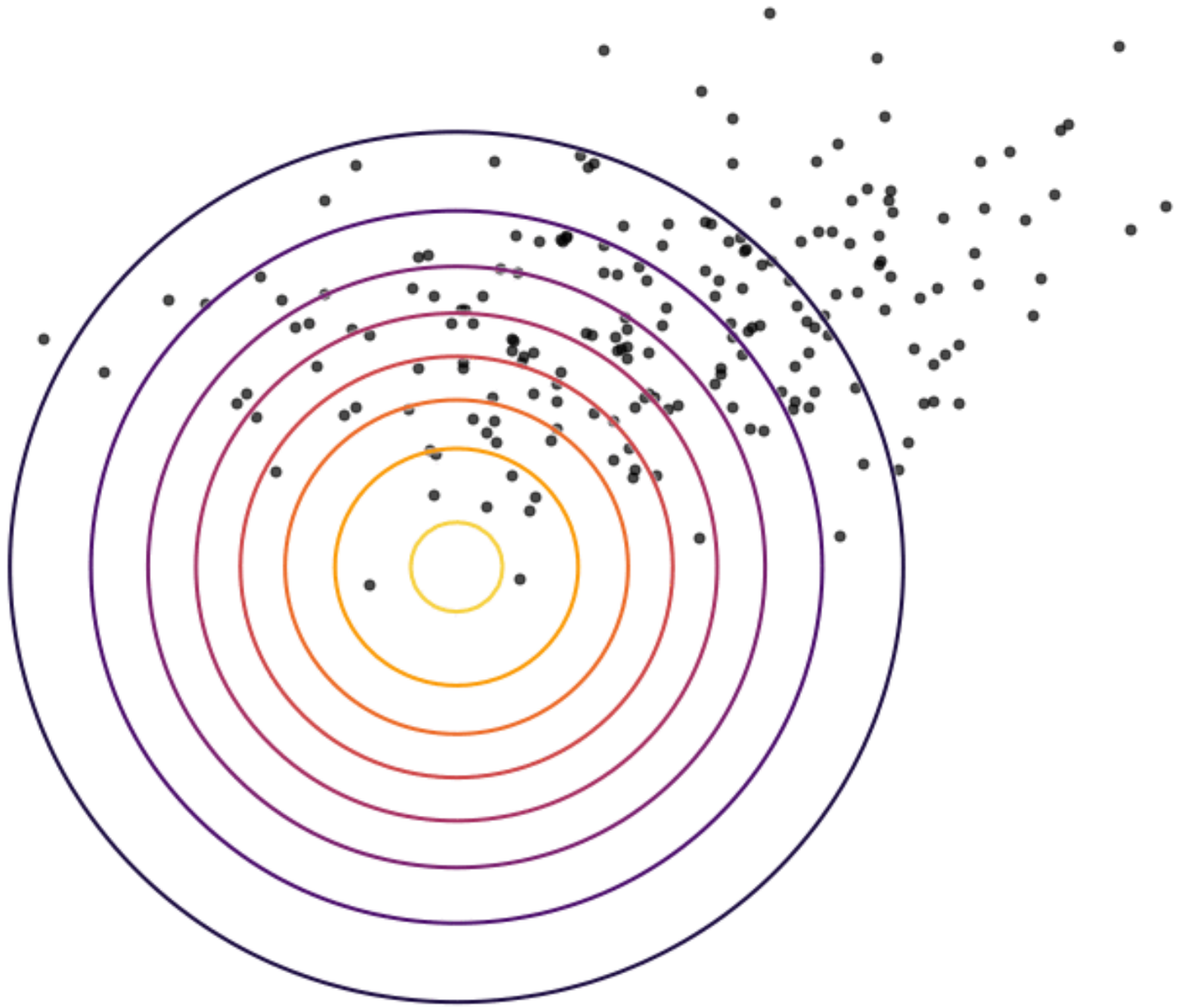
ENM 5310: Data-driven Modeling and Probabilistic Scientific Computing

Lecture #8

Black-Box Variational Inference

Variational inference



Variational inference

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta|)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta|)p(\theta)}{\int p(\mathcal{D}|\theta|)p(\theta)d\theta}$$

- Variational inference provides a computational framework for approximate Bayesian inference.
- The idea is that we'll approximate the posterior distribution with a family of distributions that is easy to work with.
- It will provide us with a set of tools for transforming the sampling problem (integration) to an optimization problem, that can be scaled to large models (i.e. with many parameters) and large data-sets.
- It also tends to favor approximations that underestimate the variance, and it usually will result in approximate distributions that get the means right but underestimate the variance.

Variational inference

Black-Box Stochastic Variational Inference in Five Lines of Python

David Duvenaud
dduvenaud@seas.harvard.edu
Harvard University

Ryan P. Adams
rpa@seas.harvard.edu
Harvard University

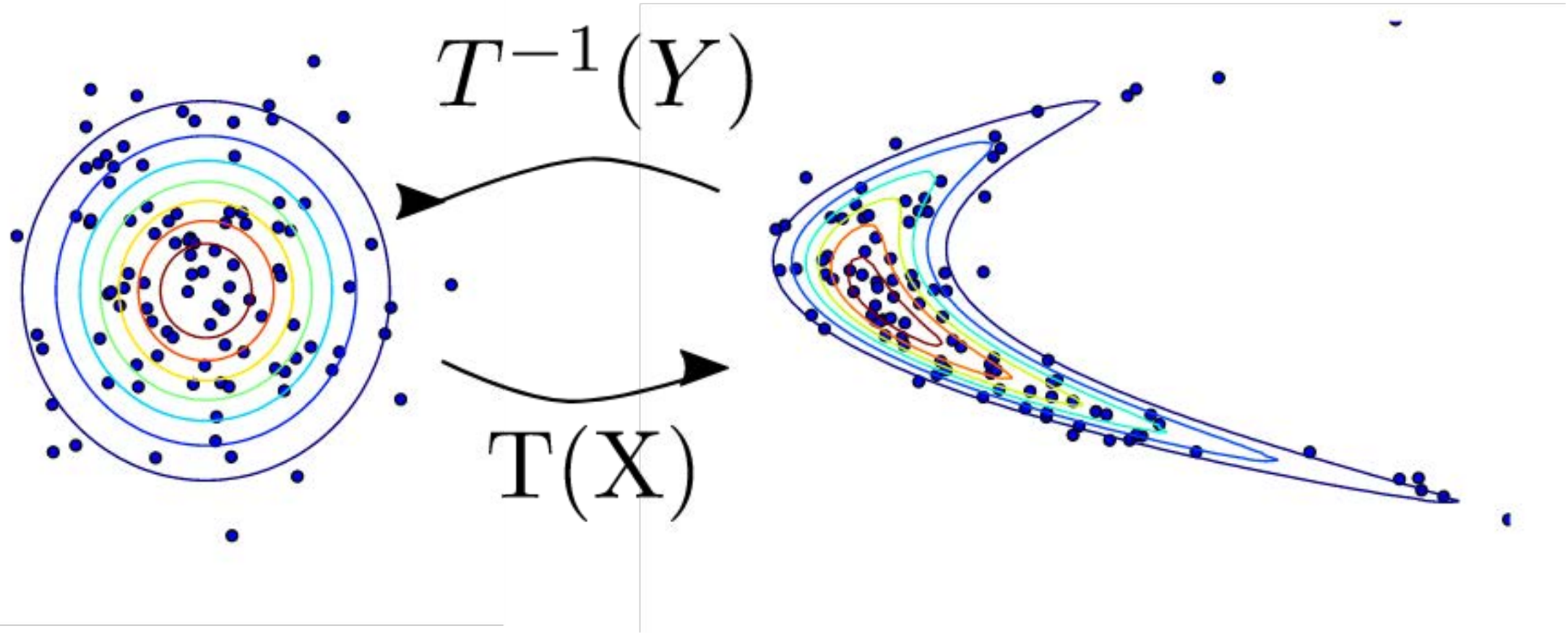
Abstract

Several large software engineering projects have been undertaken to support black-box inference methods. In contrast, we emphasize how easy it is to construct scalable and easy-to-use automatic inference methods using only automatic differentiation. We present a small function which computes stochastic gradients of the evidence lower bound for any differentiable posterior. As an example, we perform stochastic variational inference in a deep Bayesian neural network.

```python
def lower_bound(variational_params, logprob_func, D, num_samples):
    # variational_params: the mean and covariance of approximate posterior.
    # logprob_func:       the unnormalized log-probability of the model.
    # D:                  the number of parameters in the model.
    # num_samples:        the number of Monte Carlo samples to use.

    # Unpack mean and covariance of diagonal Gaussian.
    mu, cov = variational_params[:D], np.exp(variational_params[D:])

    # Sample from multivariate normal using the reparameterization trick.
    samples = npr.randn(num_samples, D) * np.sqrt(cov) + mu

    # Lower bound is the exact entropy plus a Monte Carlo estimate of energy.
    return mvn.entropy(mu, np.diag(cov)) + np.mean(logprob(samples))

# Get gradient with respect to variational params using autograd.
gradient_func = grad(lower_bound)
```

https://www.cs.toronto.edu/~duvenaud/papers/blackbox.pdf

Normalizing Flows



Variational Inference with Normalizing Flows

Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Shakir Mohamed SHAKIR@GOOGLE.COM
Google DeepMind, London

Monte Carlo approximation

$$\mathbb{E}_{x\sim p(x)}[f(x)] = \int f(x)p(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i),$$

where x_i are drawn iid from $p(x)$