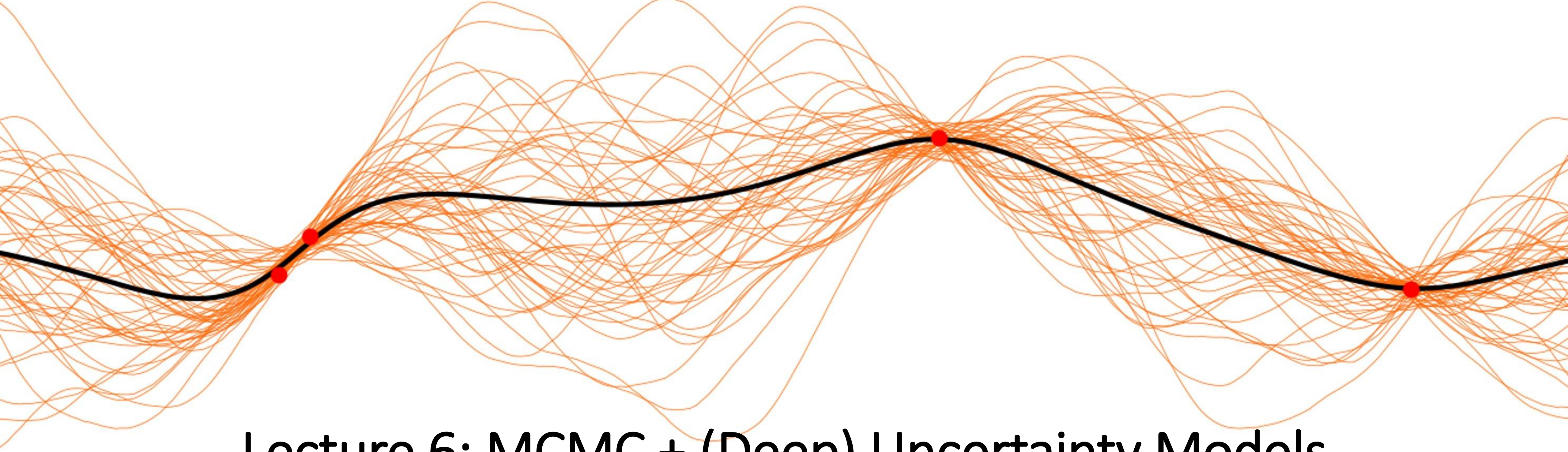


Lecture 6: MCMC + (Deep) Uncertainty Models

CS/CNS/EE/IDS 159: Advanced Topics in Machine Learning
Caltech, Spring 2023

James Bowden

Lecture Outline

- Approximate Inference: SVI vs. MCMC
- MCMC: approximately sampling from the “true” posterior
- (Deep) Uncertainty models (inference-method agnostic?)
    - Gaussian processes
    - (Bayesian) Neural networks
    - Deep kernels
    - (Deep) ensembles
    - Neural processes + more

Refresher: Approximate Posterior Inference

- Why posterior inference?
  - Learn a distribution over parameters, θ, for our model

$$p(\theta \mid D) = \frac{p(\theta)p(D \mid \theta)}{p(D)}$$

  - Given posterior: sample parameters, compute predictive distribution, compute predictive uncertainties for any test point, …
- Why approximate?
  - Evidence ("marginal likelihood") is generally intractable: "marginalize" out **all** parameters, i.e., compute **high-dimensional integral** w/ dim(θ)
  - Want complex models / high-dim θ, have data from complex distribution

Summary: Approximate Posterior Inference

SVI

- How can we use **backprop** to **learn** an approximate posterior?
- Focus on formulating a **variational objective**
- Takes advantage of autograd packages, **GPU**
- Generally faster!
- What if true posterior $\notin$ chosen family? Not even close?

MCMC

- How can we “directly” **sample from** the posterior?
- Focus on your **sampler** matching the true posterior, and **quickly**!
- Takes advantage of compilation packages, **CPU** / **parallel proc.**
- Generally slower ☹
- Exact in the limit!

When to use…? (from Blei et al. 2018)

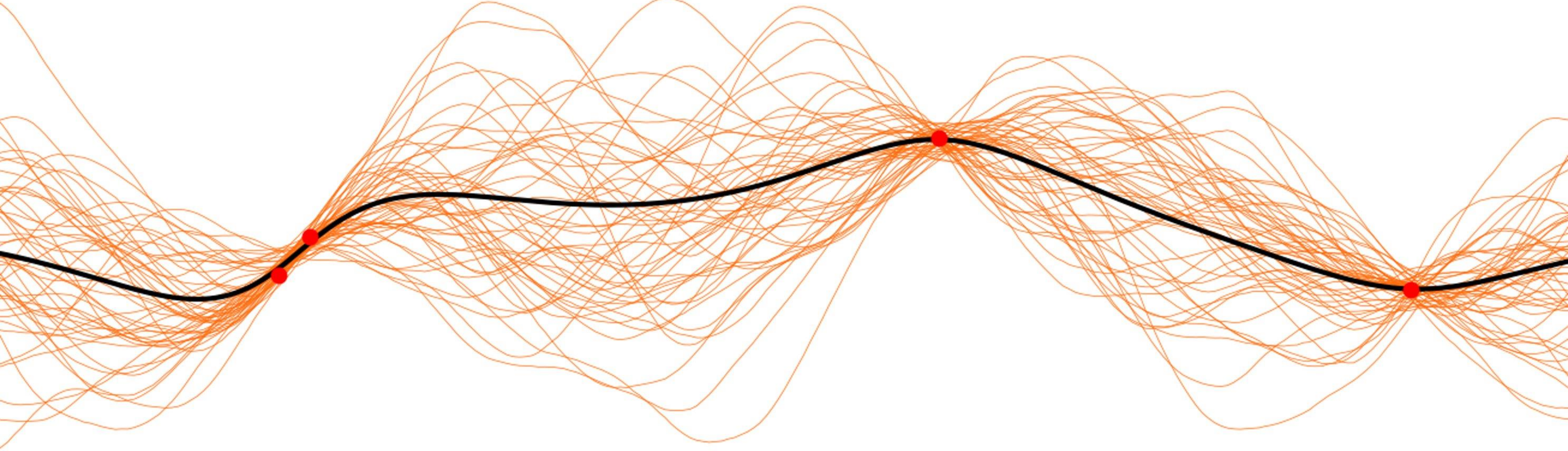
SVI

- Larger datasets
- Quicker exploration of many models
- Generally underestimates posterior variance
  - Perspective as regularizing?
- Assumes independence of vars (mean-field)

MCMC

- Smaller datasets
- Have model class we're pretty confident in already
- Require precise inference
  - More complex underlying dist.?
  - Overfit data, noise?
- Works on highly-correlated vars

- Depends heavily on downstream tasks!



Questions?

MCMC: Markov Chain Monte Carlo

- Why Markov chains?
  - If you can get chain to converge to desired stationary distribution…
  - Very cheap* to draw samples from this distribution!
- Why Monte Carlo?
  - Customary name for simulation of random processes
- So, if we can arbitrarily draw samples from the posterior, good enough!
  - In the limit, we'll recover the "true posterior"*

Markov chains

A Markov chain is defined by:

- A state space, **X**, e.g., all possible parameter vectors
- An initial distribution, $P(X_1)$
- A transition probability distribution, $P(X_{n+1} \mid X_n)$
  - Remove dependence on *n*

We get a chain of random elements $X_1, X_2, \ldots, X_n$

- Each sample in the chain is drawn from $P(X_{n+1} \mid X_n)$
- Would like $P(X_{n+1} \mid X_n) = P(\theta|D)$, the posterior → posterior sampling!

MCMC for Bayesian inference

What do we want for Bayesian inference?

- Generally, predictive mean and variance:

$$\mathbb{E}[y_*|Y;X] = \mathbb{E}_{\theta\sim p(\theta|Y;X)}[f_\theta(x_*) + \epsilon]$$

$$\mathbb{V}[y_*|Y;X] = \mathbb{E}_{\theta\sim p(\theta|Y;X)}[(f_\theta(x_*) + \epsilon)^2] - \mathbb{E}_{\theta\sim p(\theta|Y;X)}[f_\theta(x_*) + \epsilon]^2$$

What do we need from MCMC?

- Posterior samples, $\theta \sim p(\theta|Y;X)$
- Hopefully: accurate* + lots of them

Relevant Markov chain properties

Markov chain: sequence of random elements X_1, X_2, … where:

- **Markov property**: $P(X_{n+1} \mid X_1, \ldots X_n) = P(X_{n+1} \mid X_n)$
- **Ergodicity**: can get to any state $X \in \mathbf{X}$ (not necessarily in 1 step)
  - Long-term, each state is independent of start state
- **Reversibility**: forward chain and reversed chain have same distr.
  - Implies stationarity, not other way around
- **Stationarity**: $P(X_n)$ does not depend on *n* (along w/ MP)
  - Stationary distribution, $\pi_i = P(X_n = i)$

Reversibility + ergodicity => distribution of samples → $\pi(X)$ in limit

MCMC samplers

Intuition: spend time at any $\theta \in \boldsymbol{\theta}$ proportional to target density

- Metropolis-Hastings (1953 / 1970, physicists/chemists)
- Gibbs (1984) – MH special case
  - Brought to statistics, Bayesian community in 1990
- Hamiltonian MC (1987, Neal 1996, Stan 2012)
- NUTS (2014) – HMC extension
- Slice sampling – fun bonus content
- …? your sampler?

Metropolis-Hastings

Algorithm sketch:

- Choose arbitrary starting state, θ_1
- Choose arbitrary* proposal distribution, $P(\theta_{cand} \mid \theta_n)$
  - Generally chosen to be symmetric, e.g., $N(\theta_n, \sigma^2)$, σ^2 is a hyperparameter
- For *t* samples:
  - Pick $\theta_{cand} \sim P(\theta_{cand} \mid \theta_n)$
  - Compute acceptance ratio, $\alpha = P(\theta_{cand} \mid D) / P(\theta_n \mid D)$
  - Generate uniform random number $r \in [0,1]$
  - Accept $\theta_{n+1} = \theta_{cand}$ if $\alpha \geq r$; o/w reject and $\theta_{n+1} = \theta_n$

Metropolis-Hastings

- Acceptance ratio includes posterior…

$$\alpha = \frac{P(\theta_{cand}|D)}{P(\theta_n|D)} = \frac{\frac{P(\theta_{cand})*P(D|\theta_{cand})}{P(D)}}{\frac{P(\theta_n)*P(D|\theta_n)}{P(D)}} = \frac{P(\theta_{cand}) * P(D|\theta_{cand})}{P(\theta_n) * P(D|\theta_n)}$$

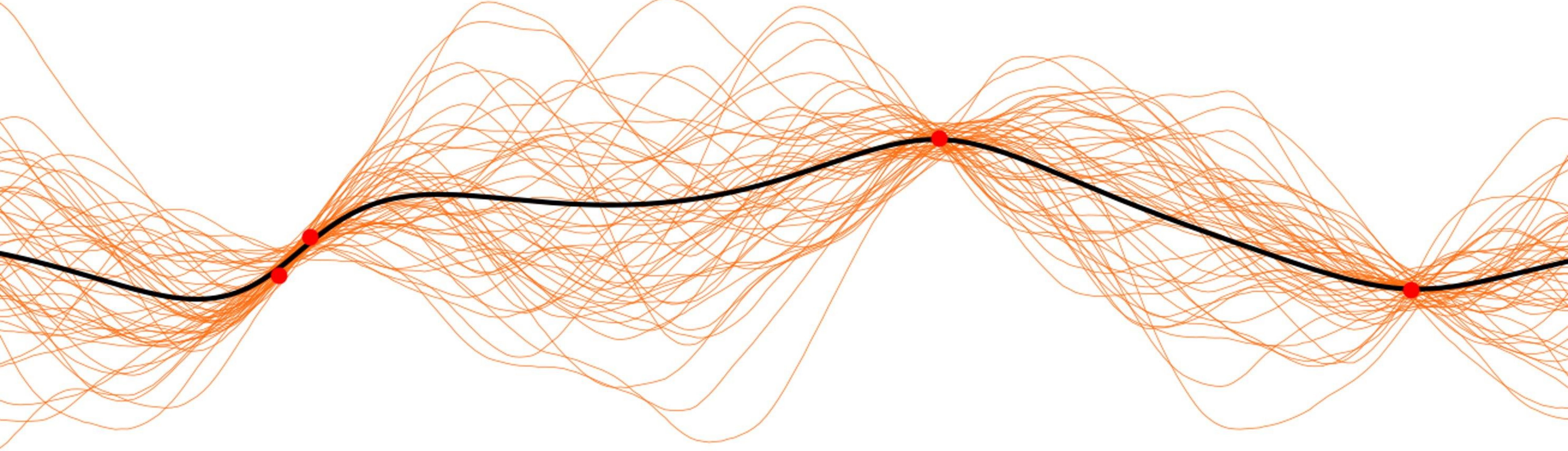
- Key insight: only need **unnormalized density** (prior * likelihood)
- i.e., don't need to calculate the (intractable) marginal likelihood!
    - Marginal is **not** dependent on choice of θ

MH: Properties satisfied?

- **Markov property**: $P(X_{n+1} \mid X_1, \ldots X_n) = P(X_{n+1} \mid X_n)$
  - Yes, proposal distribution only relies on θ_n
- **Ergodicity**: can get to any state $X \in \mathbf{X}$ (not necessarily in 1 step)
  - Yes, Gaussian can get to any vector in $\boldsymbol{\theta}$
- **Reversibility**: forward chain and reversed chain have same distr.
  - Gaussian update: symmetric probability between θ_{n+1} and θ_n
- **Stationarity**: $P(X_n)$ does not depend on *n* (along w/ MP)
  - By reversibility
  - Stationary distribution: the posterior!

MH conclusions

- Fun visualization (1 and 2)
- Pretty intuitive
- What proposal distribution to choose?
  - If Gaussian, what σ^2 ?
  - How to cover high-dim $\boldsymbol{\theta}$ space well?
- "Too random" – high rejection rates, wasted compute
- In practice, nearby samples are correlated
  - May take many samples to reach limit behavior



Questions?

Gibbs (special case of MH)

Algorithm sketch:

- Choose arbitrary starting state, θ_1
- For *t* samples:
  - Sample each component of θ_{n+1} , θ_{n+1}^{i} , $i \in [1, m]$, holding others fixed
  - $\theta_{n+1}^{i} \sim P(\theta_{n+1}^{i} \mid \theta_{n+1}^{1}, \ldots, \theta_{n+1}^{i-1}, \theta_{n}^{i+1}, \ldots, \theta_{n}^{m})$
- Iteratively sample from conditional posterior, $P(\theta^{i} \mid D, \theta^{-i})$
  - $P(\theta \mid D) = P(\theta^{m}i\ \theta^{-i} \mid D) = P(\theta^{i} \mid D, \theta^{-i}) * P(\theta^{-i} \mid D)$
  - Good for e.g., PGMs specified as collection of conditionals, conjugate priors
- MH special case: acceptance ratio is always 1

Hamiltonian (or hybrid) MC

- Metropolis-Hastings with **gradient-based proposals**
- Idea: include position, momentum information over density surface
  - Starting at θ_n , run *L* steps of particle simulation (via Hamiltonian dynamics)
  - Lands at θ_{cand} , which we accept using similar acceptance ratio criteria
- Better at exploring farther from last state
- More likely to yield candidates that are actually accepted
  - Tends to "converge" to target distribution in fewer (more expensive) samples
- Fun visualization (3 and 4)

NUTS (extension of HMC)

- What if *L* is too small?
  - Behaves like random walk, similar to original MH algo.
- What if *L* is too large?
  - Brings us back near θ_n , bad for efficient exploration!
  - Fun visualization (5)
- Idea: adaptively set *L* to prevent **U-turns**
  - Run Hamiltonian dynamics both **forward** and **backward**
  - Stop when hit a U-turn condition!
  - Randomly sample from path (both fwd, bwd)
- Fun visualization (6)

Sidenote: SGMCMC

- Gradient-based MCMC steps are expensive
  - Have to compute gradient of log posterior, P(θ_n | D)
  - Generally requires summing over all of the context data
- After SGD: can we **subsample** and make a stochastic approximation to the gradient?
- Produces consistent estimates, scales with data, slower convergence

MCMC practicalities

- How many samples until convergence to stationary distribution?
  - **Mixing**: how quickly your chain reaches $\pi(X)$
  - Approximate in practice b/c still affected by starting position to some extent
  - Some theory on this: see Markov chain central limit thm.
- How can you tell that you've converged? That you haven't?
  - Run multiple chains
  - Various diagnostics (e.g., effective # samples, inter:intra-chain var. ratio)

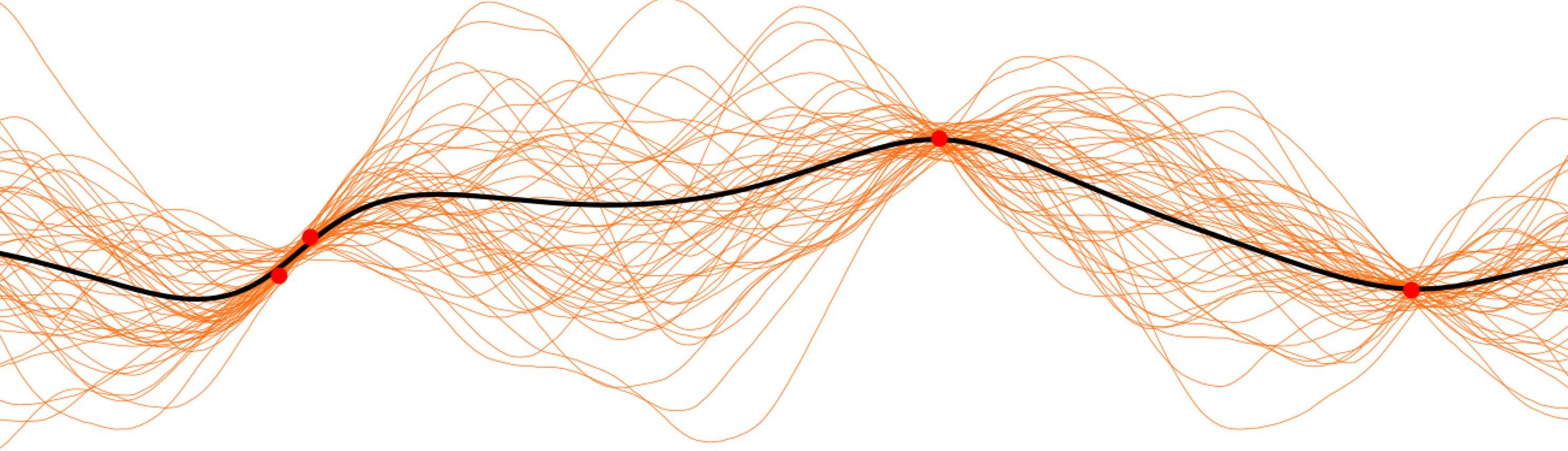
MCMC practicalities

- Pseudo-convergence / multimodality
  - Appears to converge, but eq. distribution is still conditioned on initialization
  - Stuck in one mode due to hyperparameter settings, sampler?
  - Fun visualization (7)
- Trade-off between **chain length** (limit convergence) and **# chains** (convergence detection? avg. across initializations?)

https://chi-feng.github.io/mcmc-demo/

MCMC: Implementation

- Python packages like NumPyro (jax-based), Pyro (torch-based), etc.
  - Still need gradient capabilities for gradient-based samplers! Hence jax, torch
- R packages like stan, mcmc, etc.
- Write your own sampler! (just kidding, probably don't)
- Embarrassingly parallelizable (Neiswanger et al. 2014)
- Generally, you provide:
  - Target distribution [prior over θ_n , likelihood function] + data
  - # chains, # samples, # burn-in, [thinning, initialization strategy, etc.]



Uncertainties?

Lecture Outline

- Approximate Inference: SVI vs. MCMC
- MCMC: approximately sampling from the “true” posterior
- (Deep) Uncertainty models (inference-method agnostic?)
    - Gaussian processes
    - (Bayesian) Neural networks
    - Deep kernels
    - (Deep) ensembles
    - Neural processes + more

High-Level: Approximate Posterior Inference

- We’ve seen how to do exact inference in special cases:
  - BLR if prior, likelihood are Gaussian
  - Exact Gaussian Processes “
- Otherwise: use either SVI, MCMC, or something else to approximate the posterior or its samples
  - In theory, you can be “Bayesian” about the parameters of any model
- What can we do with? Survey of models that produce uncertainties…

Refresher: Uncertainty in Bayesian inference

What do we want for Bayesian inference?

- Generally, predictive mean and variance:

$$\mathbb{E}[y_*|Y;X] = \mathbb{E}_{\theta\sim p(\theta|Y;X)}[f_\theta(x_*) + \epsilon]$$

$$\mathbb{V}[y_*|Y;X] = \mathbb{E}_{\theta\sim p(\theta|Y;X)}[(f_\theta(x_*) + \epsilon)^2] - \mathbb{E}_{\theta\sim p(\theta|Y;X)}[f_\theta(x_*) + \epsilon]^2$$

What do we need from MCMC (or SVI)?

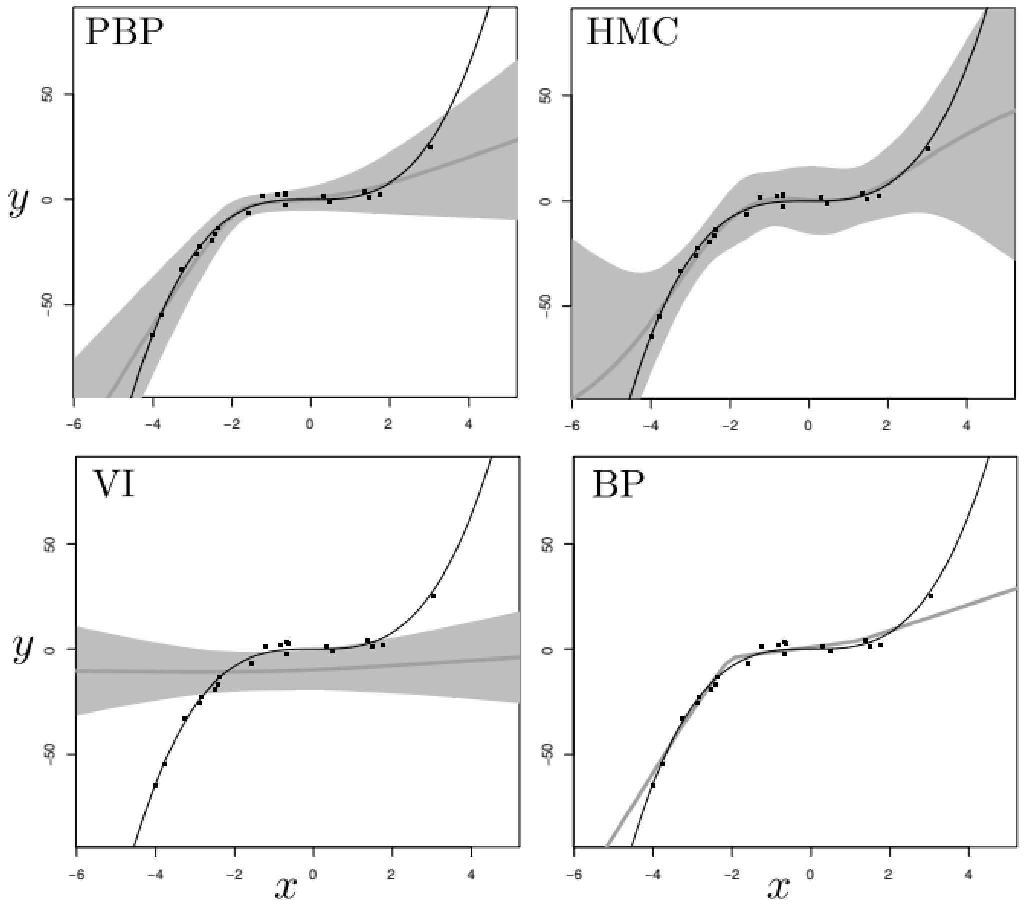
- Posterior samples, $\theta \sim p(\theta|Y;X)$
- Hopefully: accurate* + lots of them

Gaussian Processes

- Exact GP gives closed-form posterior computation
  - Can sample from, get predictive mean, variance
- Approximate GP inference…when should we do?
  - GP hyperparameters are pretty minimal
  - GP inference scales cubically with data
  - Sparse GPs, non-Gaussian likelihoods, classification: **no closed-form**
- Variational GPs, MCMC GPs
  - Define prior over hp (e.g., lengthscale, noise)
- SVI/MCMC doesn't buy much in the way of UQ – GP already has var.

Neural networks

- Deterministic NN: scalar weights, point estimate predictions
- Probabilistic Backpropagation (PBP):
  - Distributions over weights like in a standard BNN
  - Instead of prediction error, compute marginal log likelihood as loss
  - Gradient update minimizes KL div.
  - + approximations, implementation details, etc.



(Hernandez-Lobato and Adams 2015)

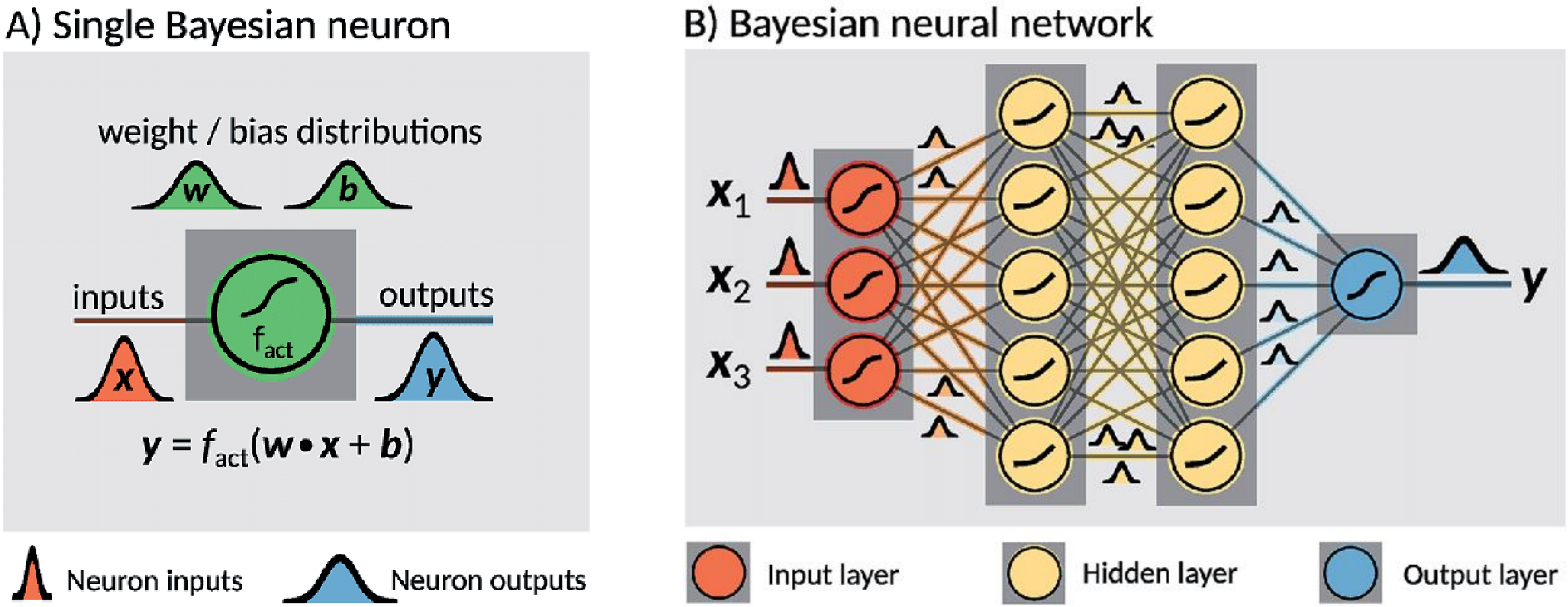
Neural networks

- Laplace approximation: 2nd order Taylor approx. about MAP
  - Compute inverse Hessian of log likelihood: scales very poorly
- Monte Carlo Dropout: test-time dropout, viewed as approximate posterior sampling
  - Argument: approximate intractable posterior with q(w), a distribution over matrices whose columns are randomly set to 0 == dropped out neuron
  - Min. KL div. in this setting recovers L2 regularization loss that dropout uses
  - Intuition: view instance of dropout as a sample, buys us predictive variance
  - Super popular for its simplicity, has fallen out of favor of late (?)

(Gal and Ghahramani 2016)

Neural networks

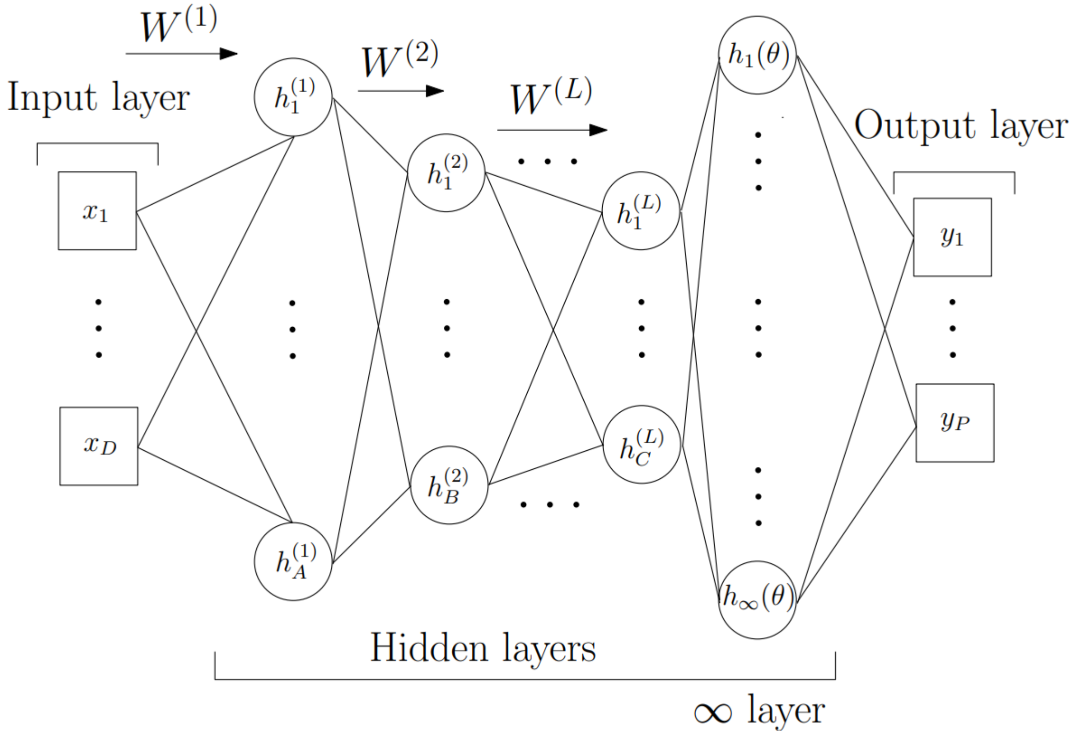
- “Standard” BNN training: SVI or MCMC!
  - See [Izmailov et al. 2021]



[Hase et al. 2019]

Deep Kernels

- Representational power of NN
- UQ from GP
- Backprop through GP + NN
  - Can also use SVI, MCMC, MC dropout
- Nice properties from both sides, avoids custom kernel design, allows for transfer / unsupervised (VAE?) learning
- Overfitting, see [Ober et al. 2021]
- My current research! Ask me about :)



[Wilson et al. 2015]

Ensembles

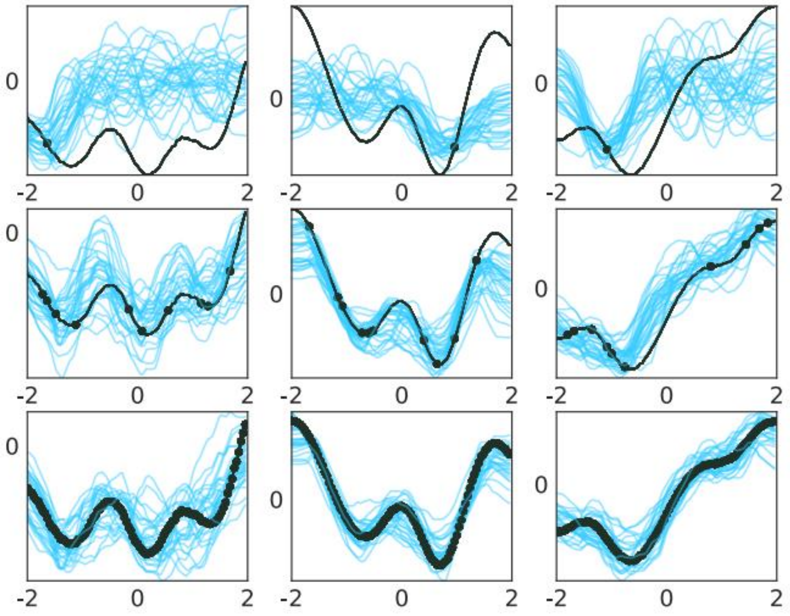
- Idea: train many independent models and **combine**
  - Model combination makes sense when true model $\notin$ hypothesis class
- **Bagging**: fit models on different subsamples, average predictions
  - e.g., random forests, extra trees, etc.
- **Stacking**: fit models on same data, use meta-model to learn combo
- **Boosting**: add models sequentially to correct predictions as you go
  - Output weighted average
  - e.g., XGBoost, AdaBoost, etc.
- Have *n* predictions: compute sample variance in naïve way, or other

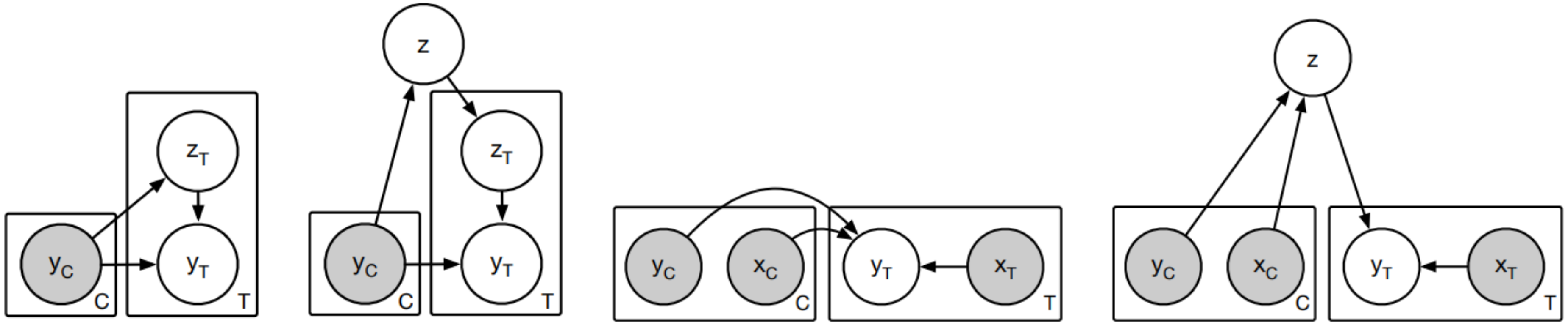
Deep Ensembles

- Can apply bagging, stacking, boosting to DNNs
- Empirically: quite good predictive performance (competitions)
- Computationally: easily parallelized
- Implementation: much simpler!
- Lakshminarayanan et al. 2017: uniform stacking of DNNs
  - Train using a proper scoring rule
  - Adversarial training to smooth predictive distribution
  - Output: Gaussian w/ sample mean, variance
- View of MC dropout as deep ensemble w/ shared parameters
  - "Implicit" ensemble: sampled networks with randomly dropped neurons

& more…

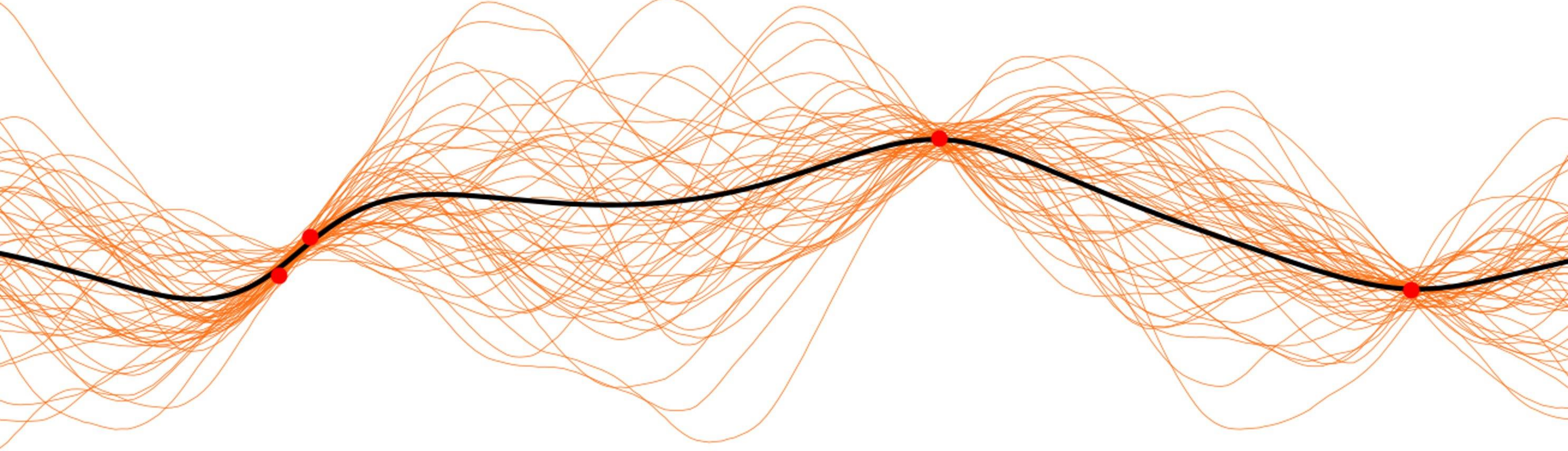
- Variational Autoencoders
- Neural processes: VAE, but for prediction
  - Multi-fidelity hierarchical NPs
- More generally: Bayesian models, hierarchical models, PGMs, etc.





(a) Conditional VAE (b) Neural statistician (c) Conditional neural process (d) Neural process

Ganelo et al. 2018



Questions?

Summary: Approximate Posterior Inference

SVI

- How can we use **backprop** to **learn** an approximate posterior?
- Focus on formulating a **variational objective**
- Takes advantage of autograd packages, **GPU**
- Generally faster!
- What if true posterior $\notin$ chosen family? Not even close?

MCMC

- How can we "directly" **sample from** the posterior?
- Focus on your **sampler** matching the true posterior, and **quickly**!
- Takes advantage of compilation packages, **CPU** / **parallel proc.**
- Generally slower ☹
- Exact in the limit!

Summary: When to use…? (from Blei et al. 2018)

SVI

- Larger datasets
- Quicker exploration of many models
- Generally underestimates posterior variance
  - Perspective as regularizing?
- Assumes independence of vars (mean-field)

MCMC

- Smaller datasets
- Have model class we're pretty confident in already
- Require precise inference
  - More complex underlying dist.?
  - Overfit data, noise?
- Works on highly-correlated vars

- Depends heavily on downstream tasks!

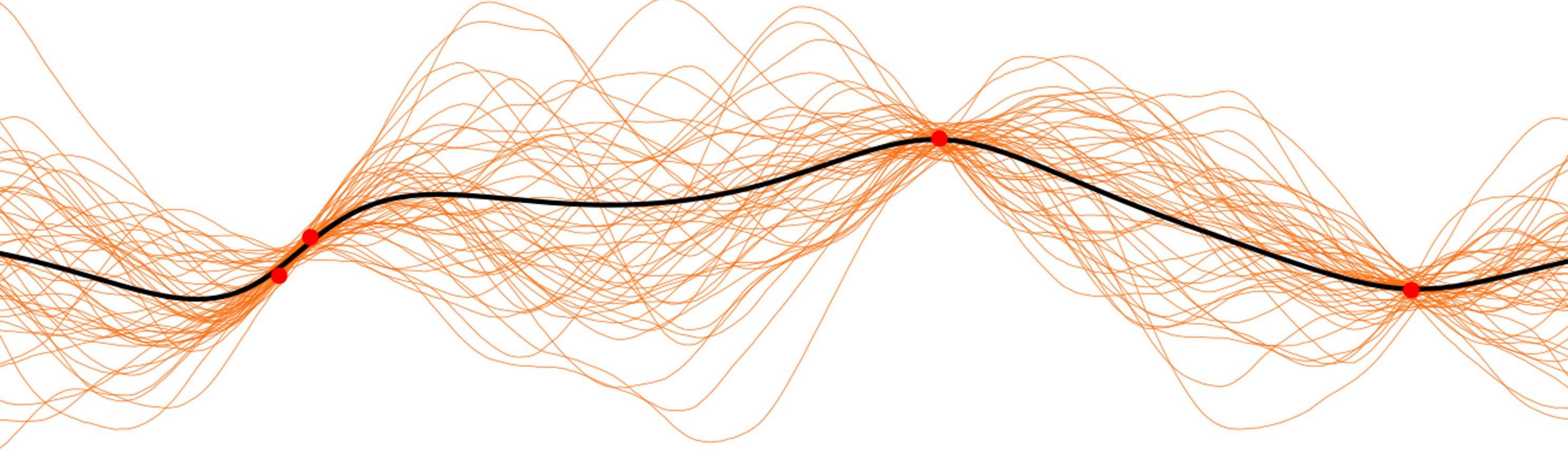
Summary: Uncertainty modeling

Many different ways of modeling predictive uncertainty, some Bayesian, some not, some neural networks, …

- Gaussian processes
- (Bayesian) Neural networks
- Deep kernels
- (Deep) ensembles
- Neural processes + more

Other orders of business

- HW 3 out tonight
  - Focuses more on practical implementation, as these are likely the methods you'll use in your projects / The Real World ™
  - Due Wed 04/26
- Looking forward: how do we use UQ methods in larger decision-making frameworks?
  - Tuesday: adaptive experimentation w/ Yisong
  - Thursday: TBD
  - Next Tuesday: Bayesian optimization w/ Raul



Questions?