

Improving Structural Estimation with Sequential Neural Posterior Estimation

By Cameron Fen

Motivation

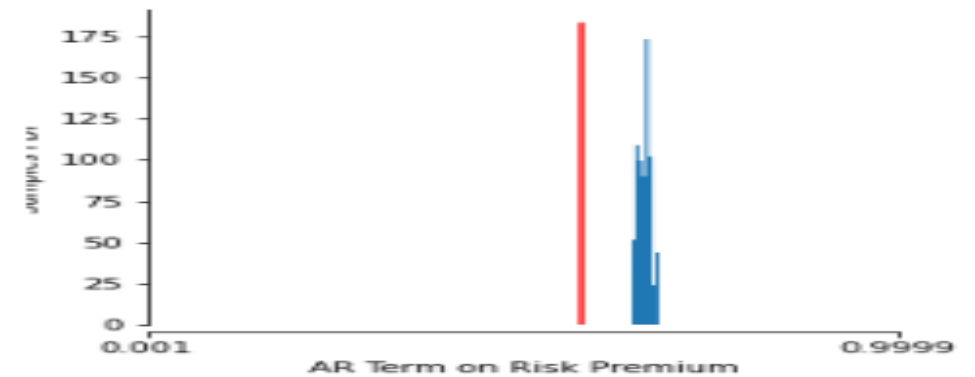
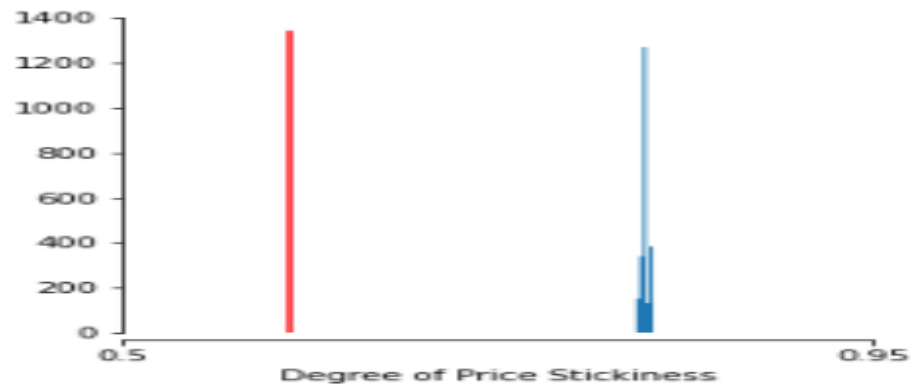
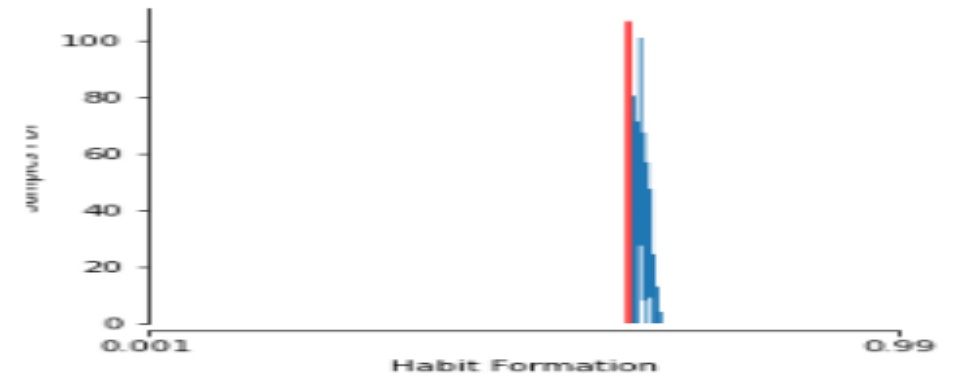
- With the curse of dimensionality, it is difficult to estimate macro models in a Bayesian fashion
- The workhorse Metropolis-Hastings Markov Chain Monte Carlo (MCMC) has difficulty converging effectively to posteriors for models with more than 10-15 parameters
- Sequential Neural Posterior Estimation (SNPE), a technique borrowed from machine learning, can estimate posteriors much more effectively

Contribution

- I introduce the SNPE machine learning algorithm to improve the Bayesian estimation of structural macroeconomic models
- SNPE can also handle likelihood free estimation in both a Bayesian and MLE framework
- Within my SNPE procedure, I also introduce normalizing flows, for high dimensional density estimation

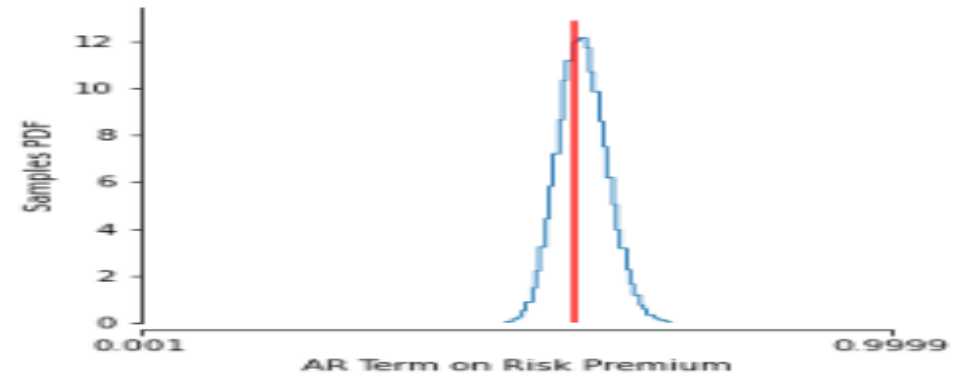
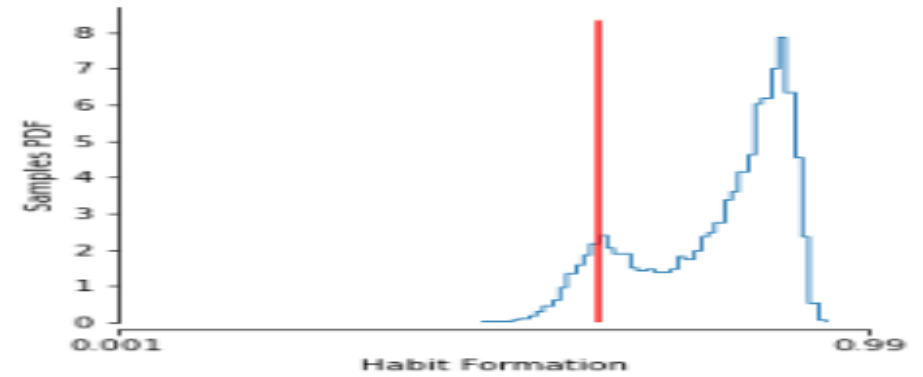
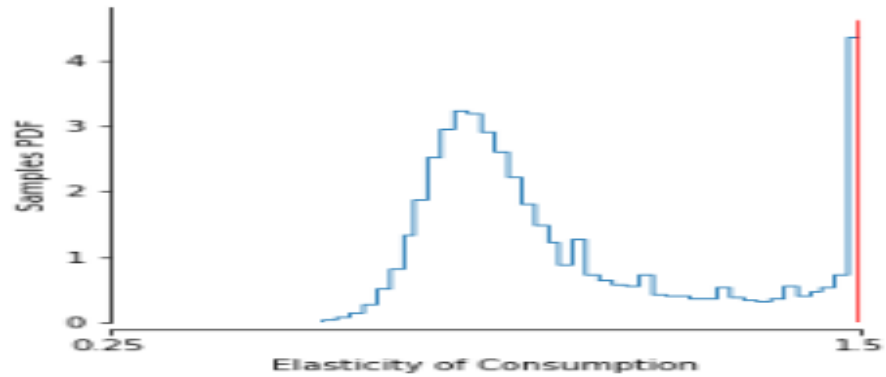
SNPE

Metropolis-Hastings MCMC on Smets-Wouters after Ten Million Iterations



[Additional Parameters](#)

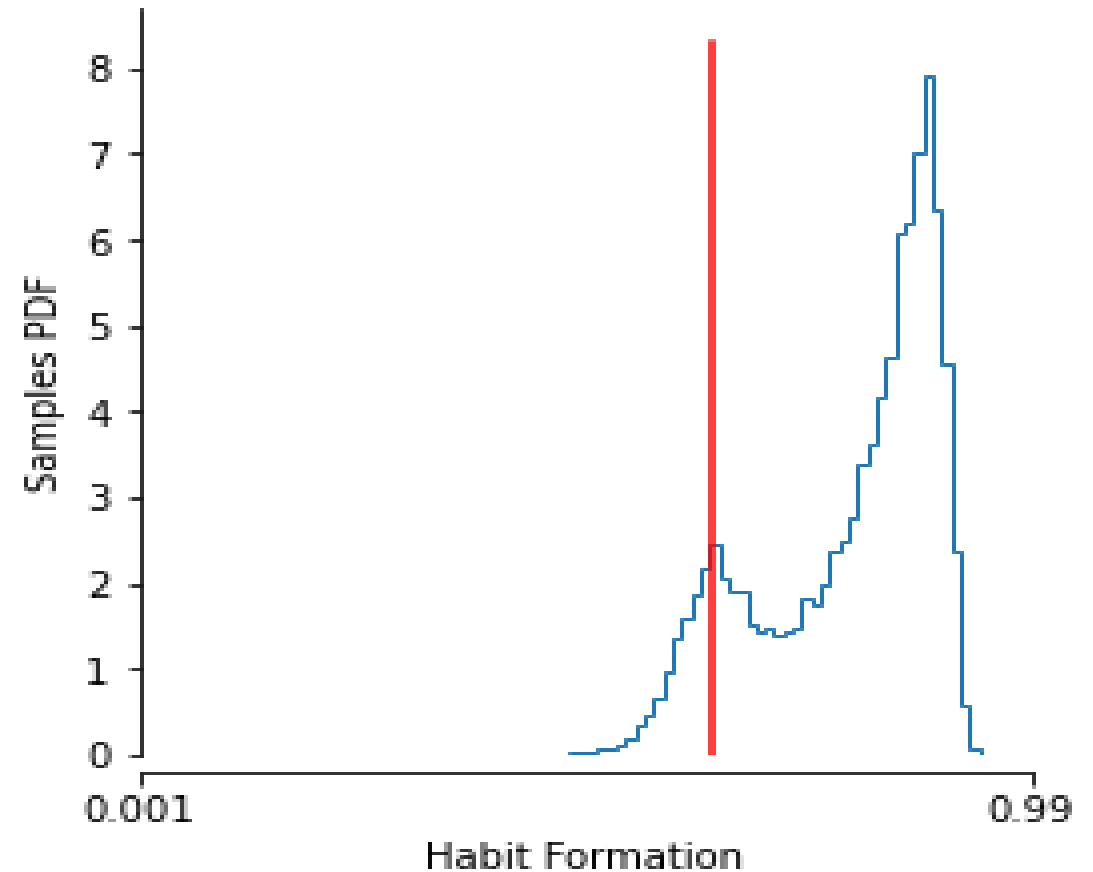
SNPE after Five Hundred Thousand Iterations



[Additional Parameters](#)

Theoretical Underpinning for SNPE Speed vs Monte Carlo

- Sampling from non-log-concave distributions—for example multimodal—with MCMC and related methods don't have polynomial computational guarantees (Chewi 2023)
- Using a parametric density estimator via maximum likelihood are roughly quadratic/polynomial in cost (Farrell 2021)



An Explanation of SNPE

Bayes Rule

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$$

Variable	Description
$P(\Theta X)$	Posterior
$P(X \Theta)$	Likelihood
$P(\Theta)$	Prior
$P(X)$	Normalization Term

MCMC and SNPE

- High level differences of MCMC and SNPE
- Whereas MCMC attempts to sample from the posterior by randomly walking to different points in proportion to the posterior probability...
- ...SNPE samples from the joint distribution of data and the parameters and then attempts to estimate a conditional density from the joint samples

What if the Likelihood is Intractable?

- This is likely to happen in the case of a macro model
- For example, both value function iteration and projection take normal shocks and transforms them with nonlinear functions
 - Distribution likely will not have a parametric form and so likelihood evaluations are impossible
- With these model solutions, it's much harder to do MCMC or even maximum likelihood estimation

If You can't Evaluate, Just Simulate

- SNPE can estimate these models, even when the likelihood is intractable, just by simulating from the likelihood
- Typically, simulation-based techniques like Method of Simulated Moments (MSM) aren't full information techniques, but can estimate more flexible models (ie ones without tractable likelihoods)

SNPE: Best of Both Worlds

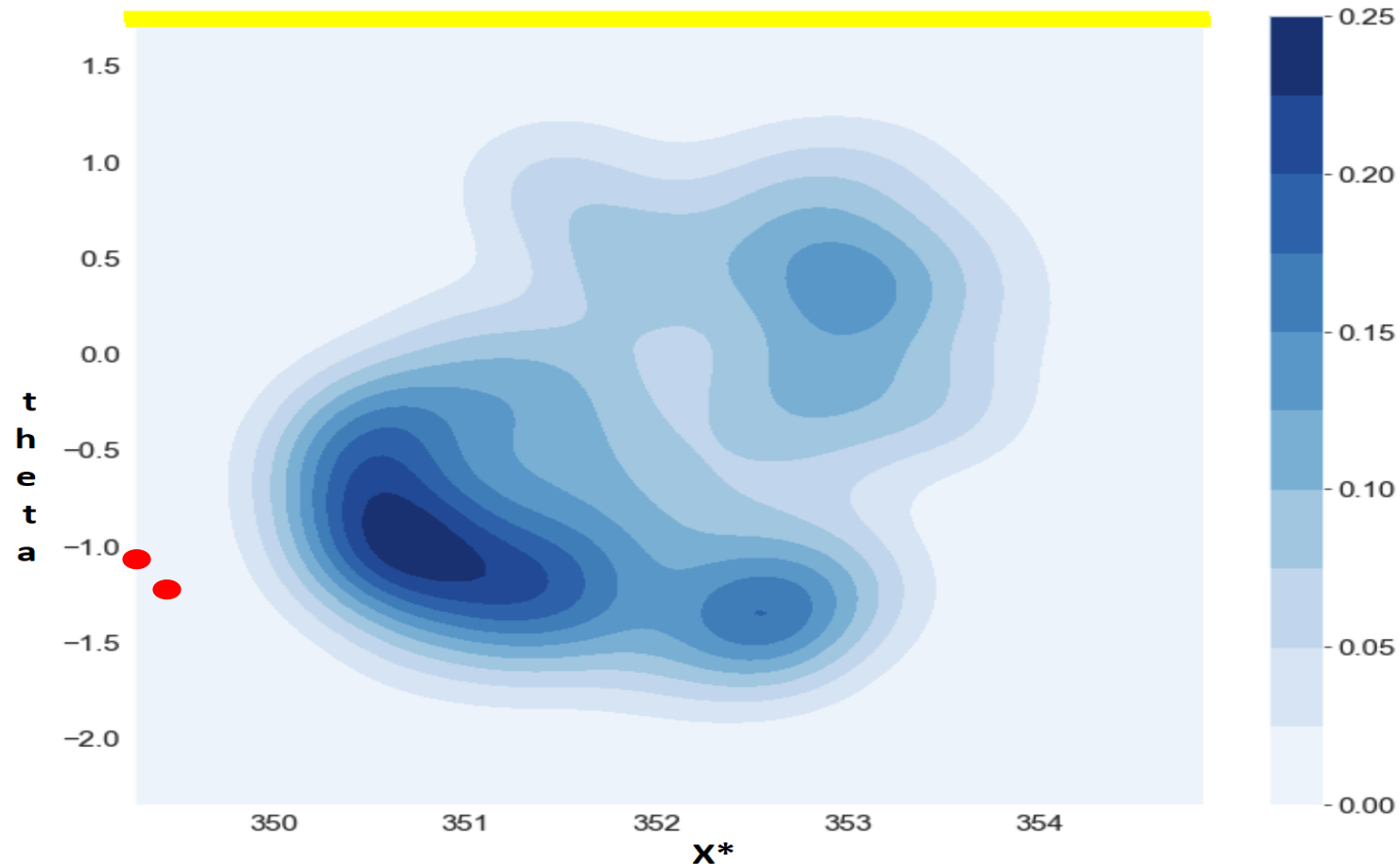
- SNPE combines the best of both worlds:
 - Can estimate any model that MSM can, but performs likelihood/Bayesian estimation
- Additionally, [adheres](#) to the state-space assumption for the data generating process rather than a Markovian assumption for MSM

SNPE: Estimation Exposition

SNPE Toy Model: $X^* = \theta + \epsilon$

- X = data, X^* = model simulations, $p(\theta)$ = prior over θ , $\epsilon \sim N(0,1)$
- Objective: $p(\theta|X^* = X)$
- **Step 1:** $\{X_i^*, \theta_i^*\}_{i=1}^I$ = simulated via Monte Carlo [Simulate diagram](#)
 1. Sample θ_i^* from $p(\theta)$
 2. Sample X_i^* from simulated model conditional on θ_i^* , $X_i^* = \theta_i^* + \epsilon$
- **Step 2:** Estimate the conditional density $p(\theta|X^*)$ [Estimate diagram](#)
 1. Use a conditional density estimator
 - Many different CDEs can be used, I will discuss a Conditional Mixture of Gaussians and Normalizing Flows
- **Step 3:** Condition X^* on the real data, X , forming the posterior:
 $p(\theta|X^* = X)$ [Condition diagram](#)

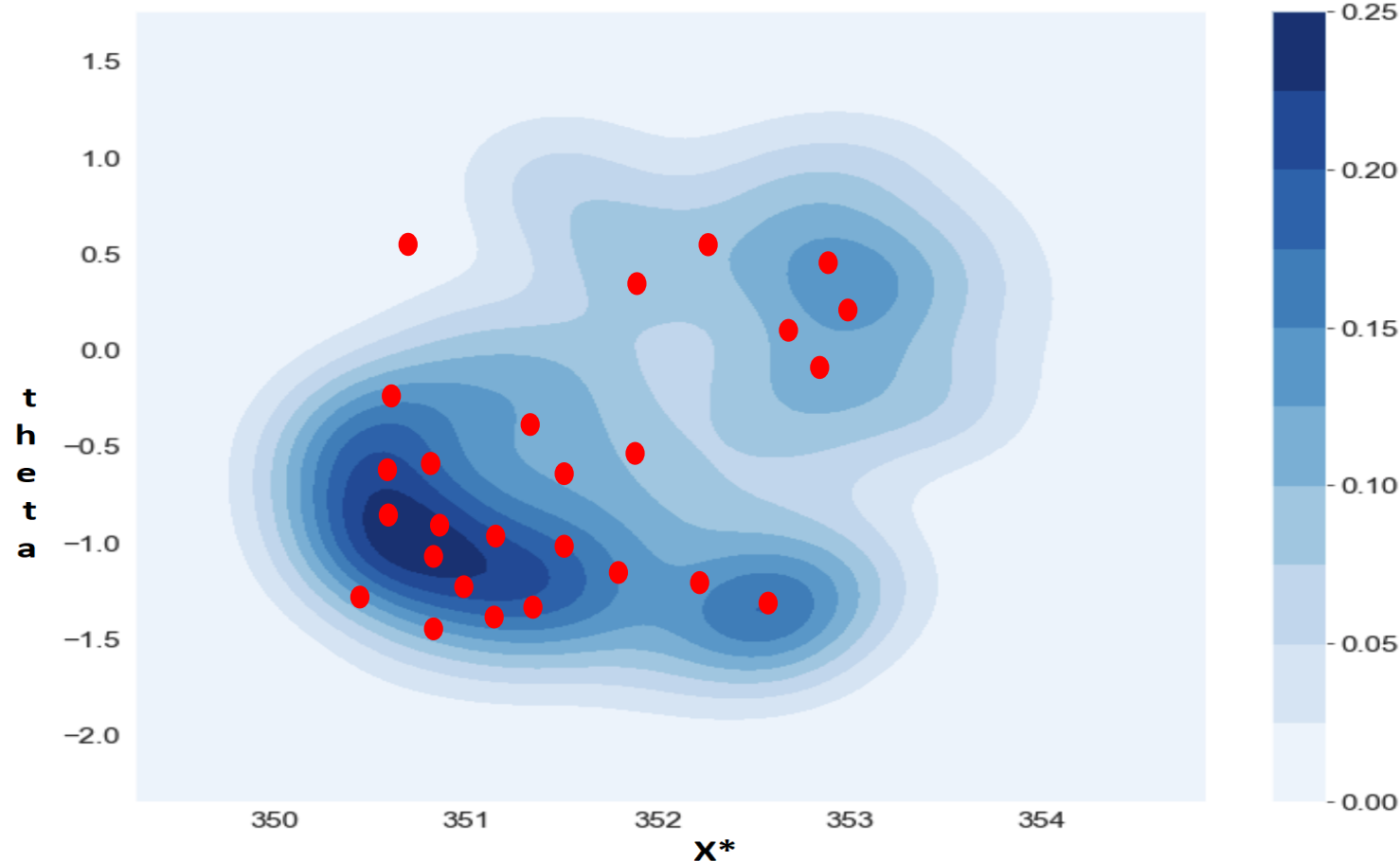
Step 1: Sample $\{\theta_i^*, X_i^*\}_{i=1}^I \sim p(\theta, X^*)$ via Monte Carlo Simulation



[Back](#)

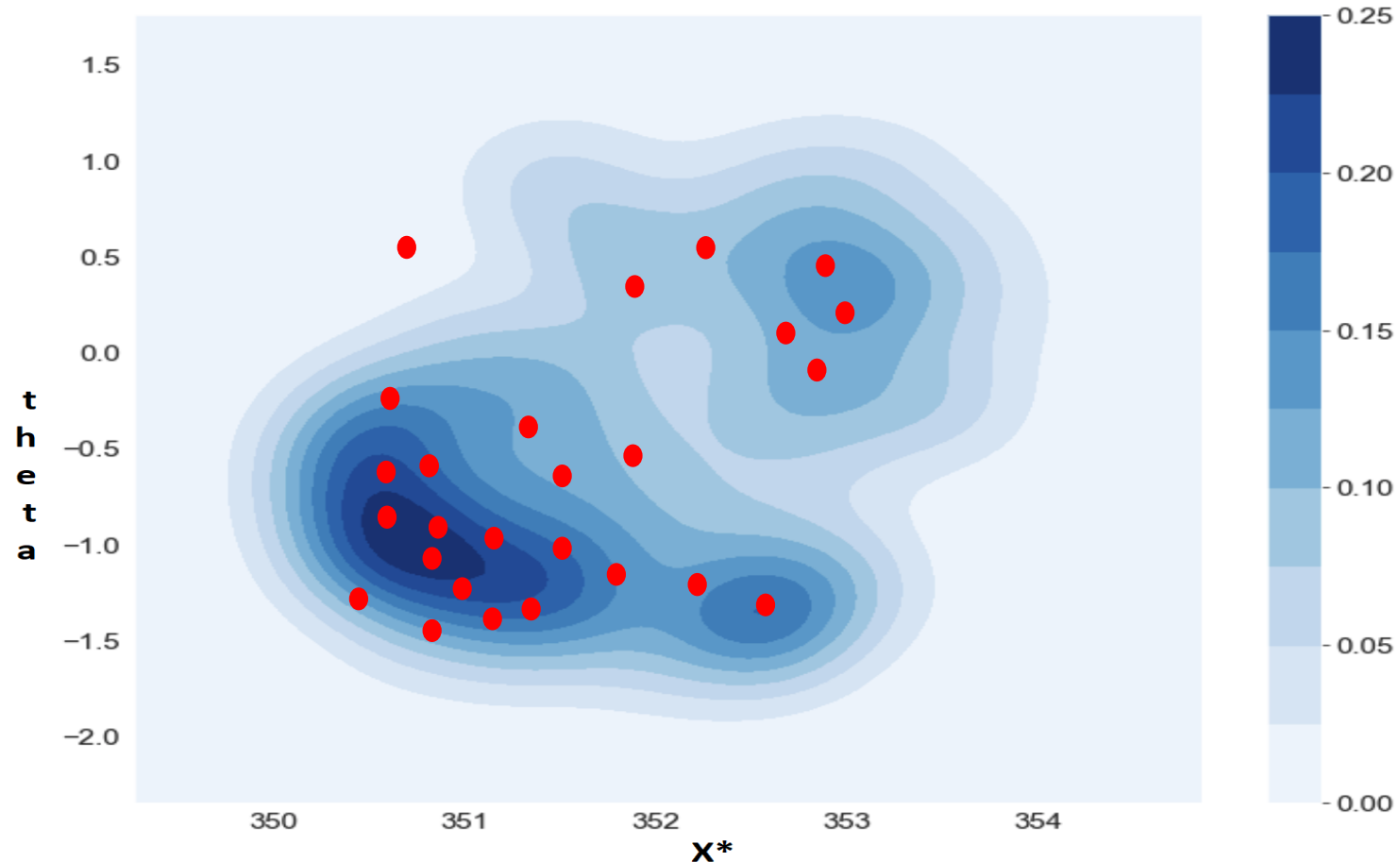
In this way one samples from the joint distribution of the prior and simulated data from the model:

$$\theta^*, X^* \sim P(\theta, X^*) = P(X^*|\theta)P(\theta)$$



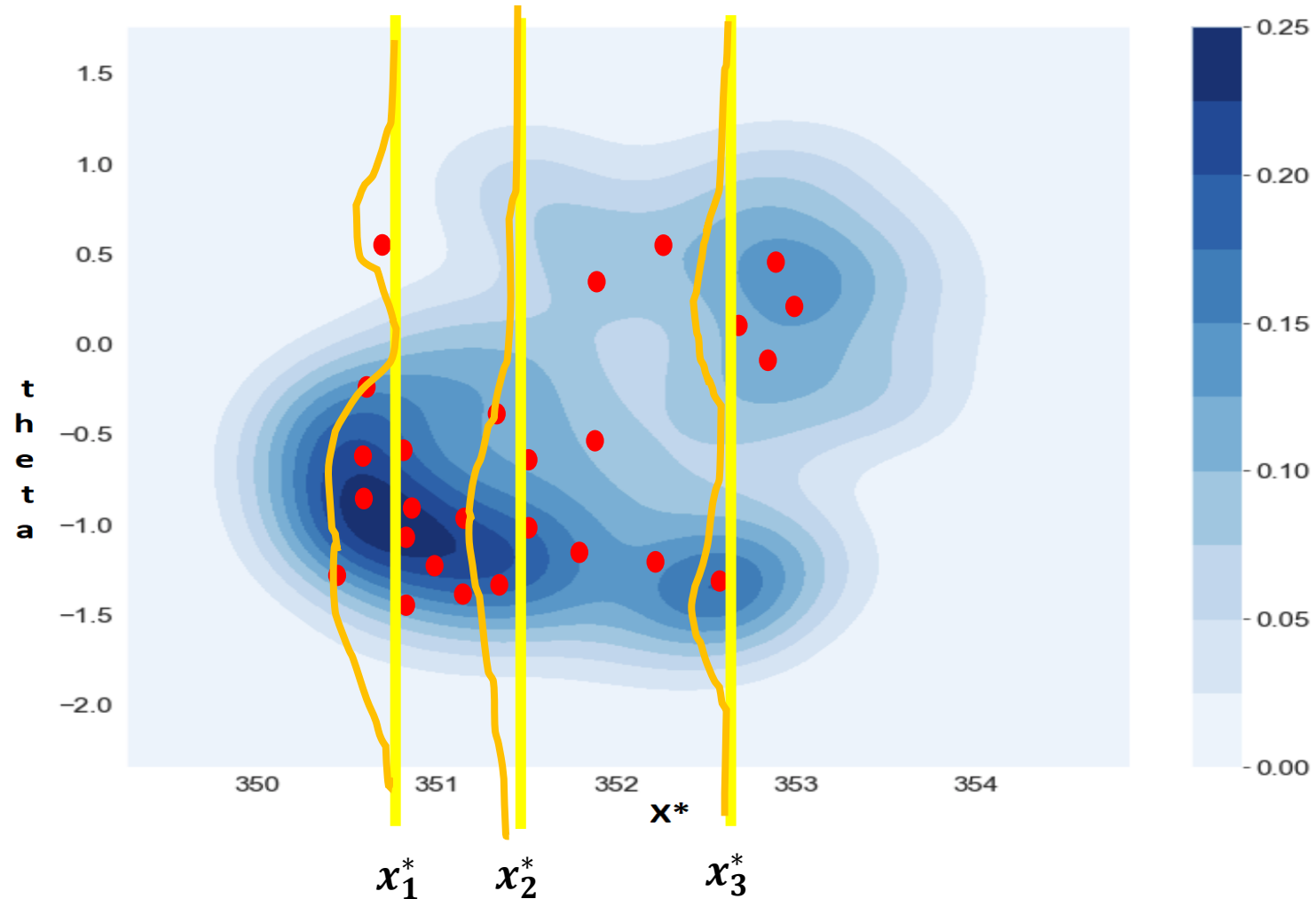
[Back](#)

Step 2: Estimate the conditional density $p(\theta|X^*)$



[Back](#)

Step 2: Estimate the conditional density $p(\theta|X^*)$

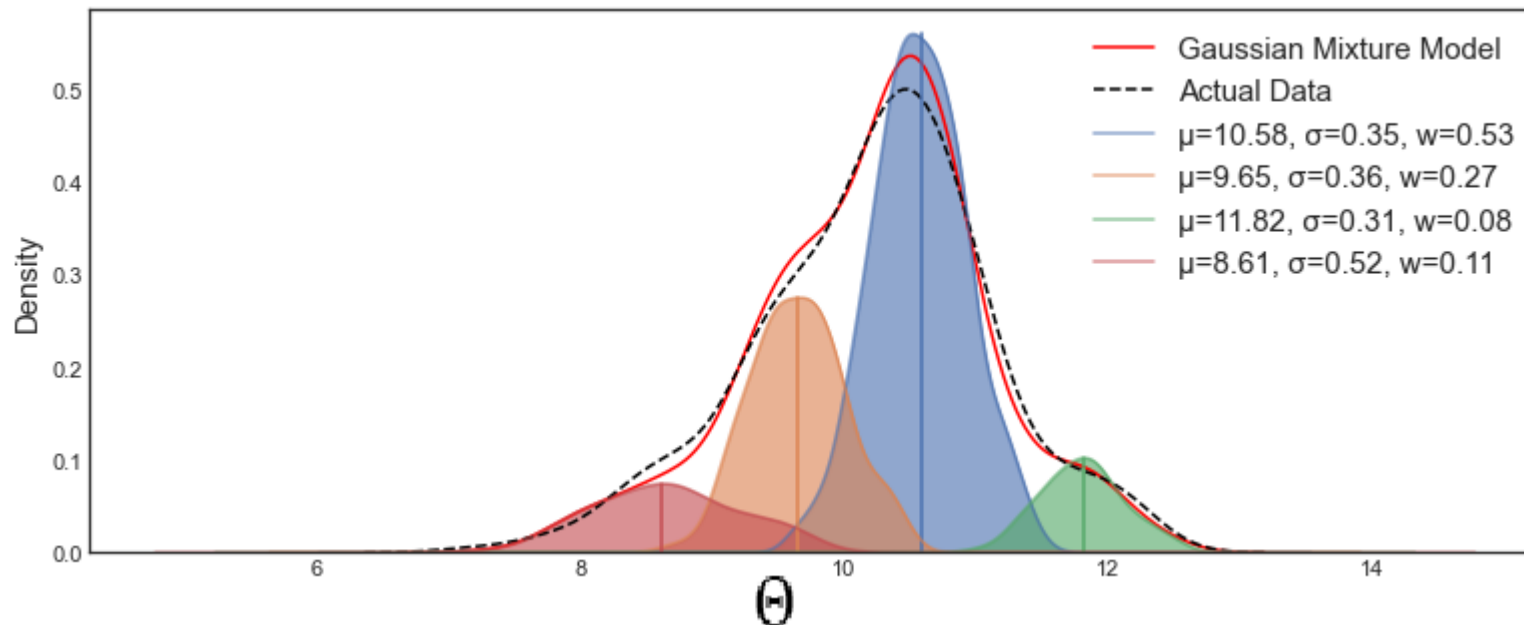


[Back](#)

Gaussian Mixture Model

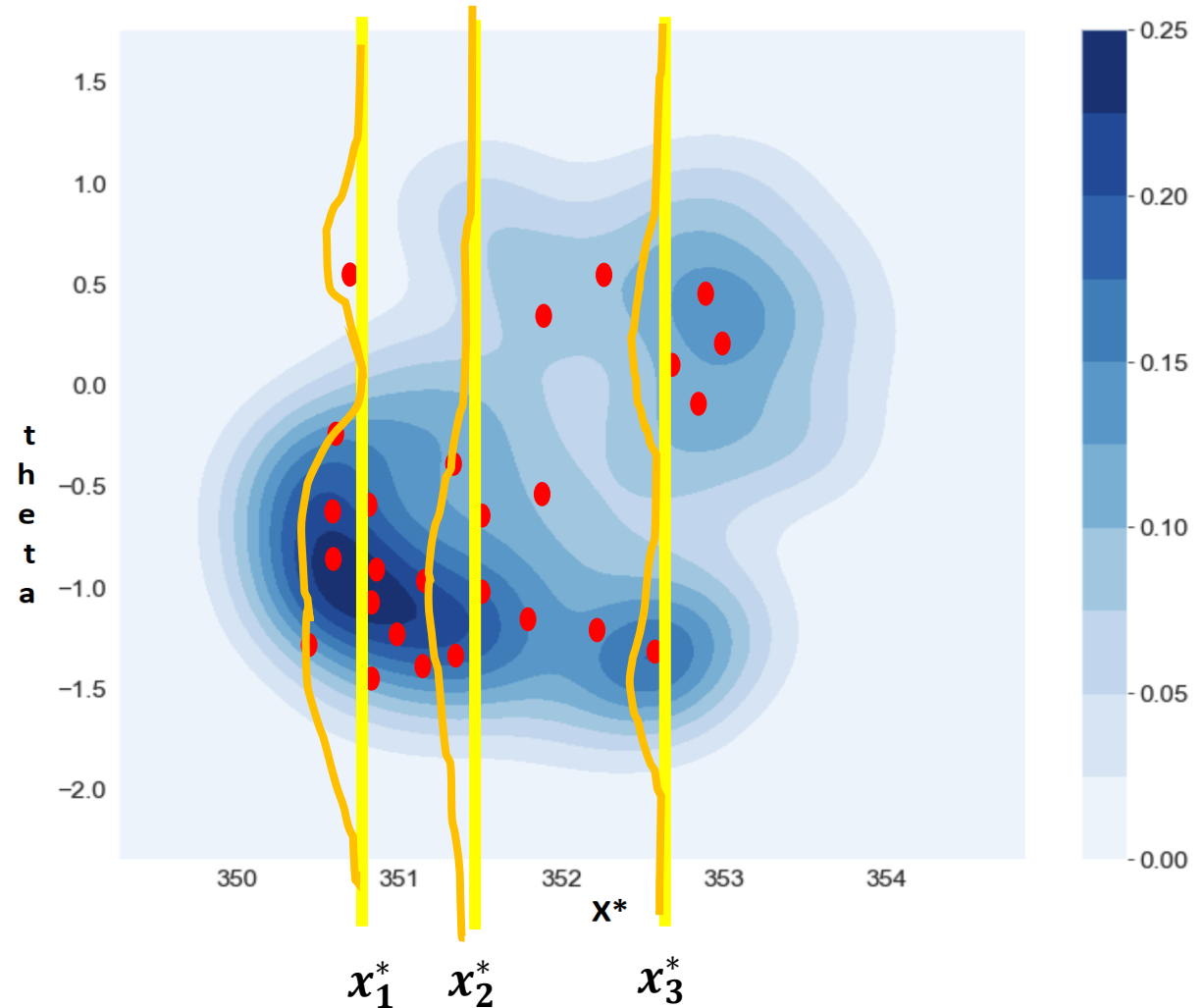
- One way to do density estimation is with a Gaussian Mixture Model (first unconditional):

- $P(\theta) = \sum_{i=1}^n \alpha_i * N(\theta|\mu_i, \Omega_i)$
 - $N(\theta|\mu_i, \Omega_i) = \frac{1}{\det(\Sigma_i)^{1/2} (2\pi)^{k/2}} \exp(-\frac{1}{2}(\theta - \mu_i)^T \Omega_i^{-1} \frac{1}{2}(\theta - \mu_i))$



From Unconditional to Conditional Mixture of Gaussians

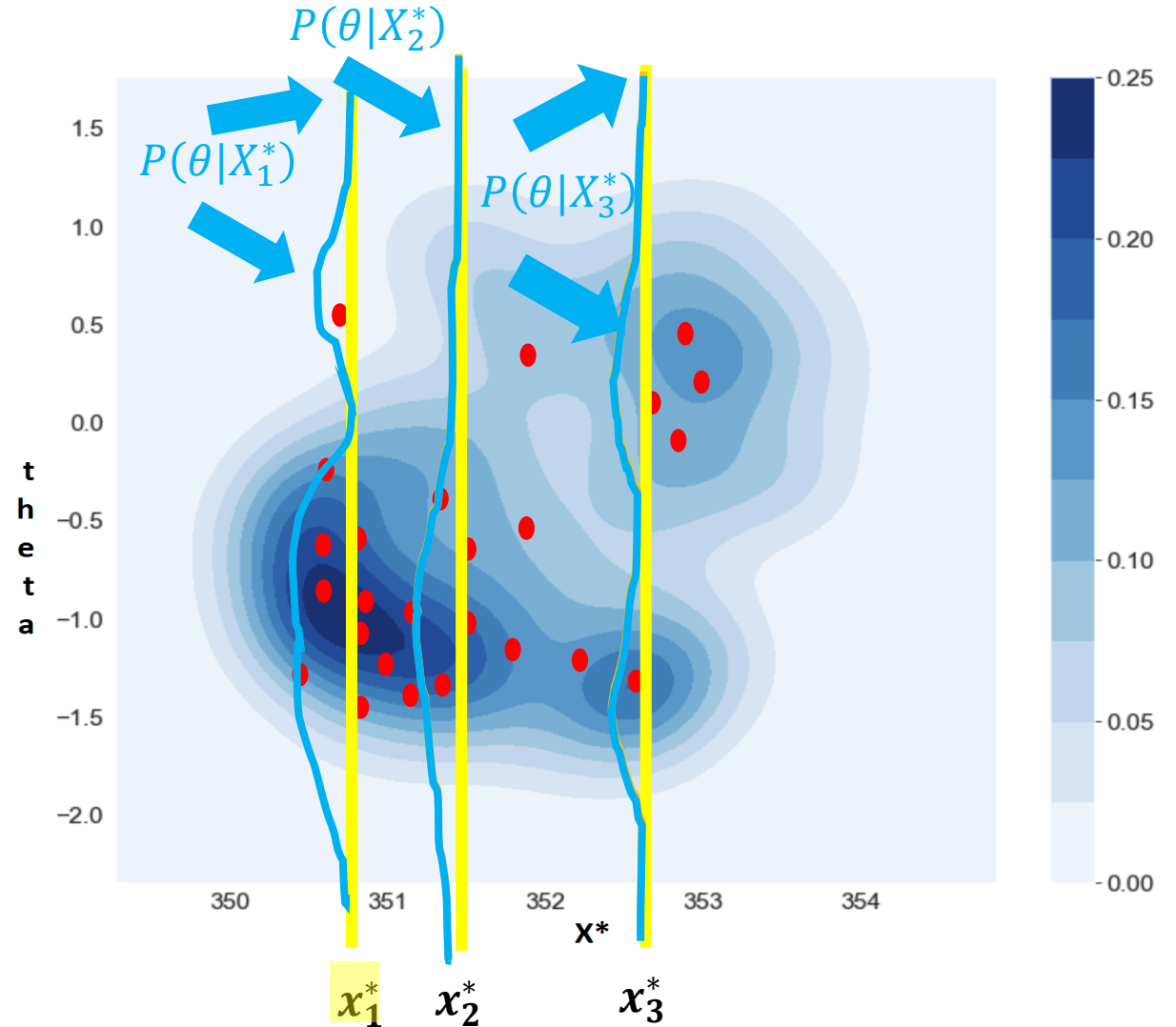
- Condition all the parameters in the mixture on X^* :
 - $P(\theta|X^*) = \sum_j^k \alpha_j(X^*) * N(\theta|\mu_j(X^*), \Omega_j(X^*))$
- In practice this means building a flexible but optimizable function F that maps X^* to $\alpha_j(X^*)$, $\mu_j(X^*)$, and $\Omega_j(X^*)$
 - Usually, a feedforward neural network is used



Conditional Mixture of Gaussians

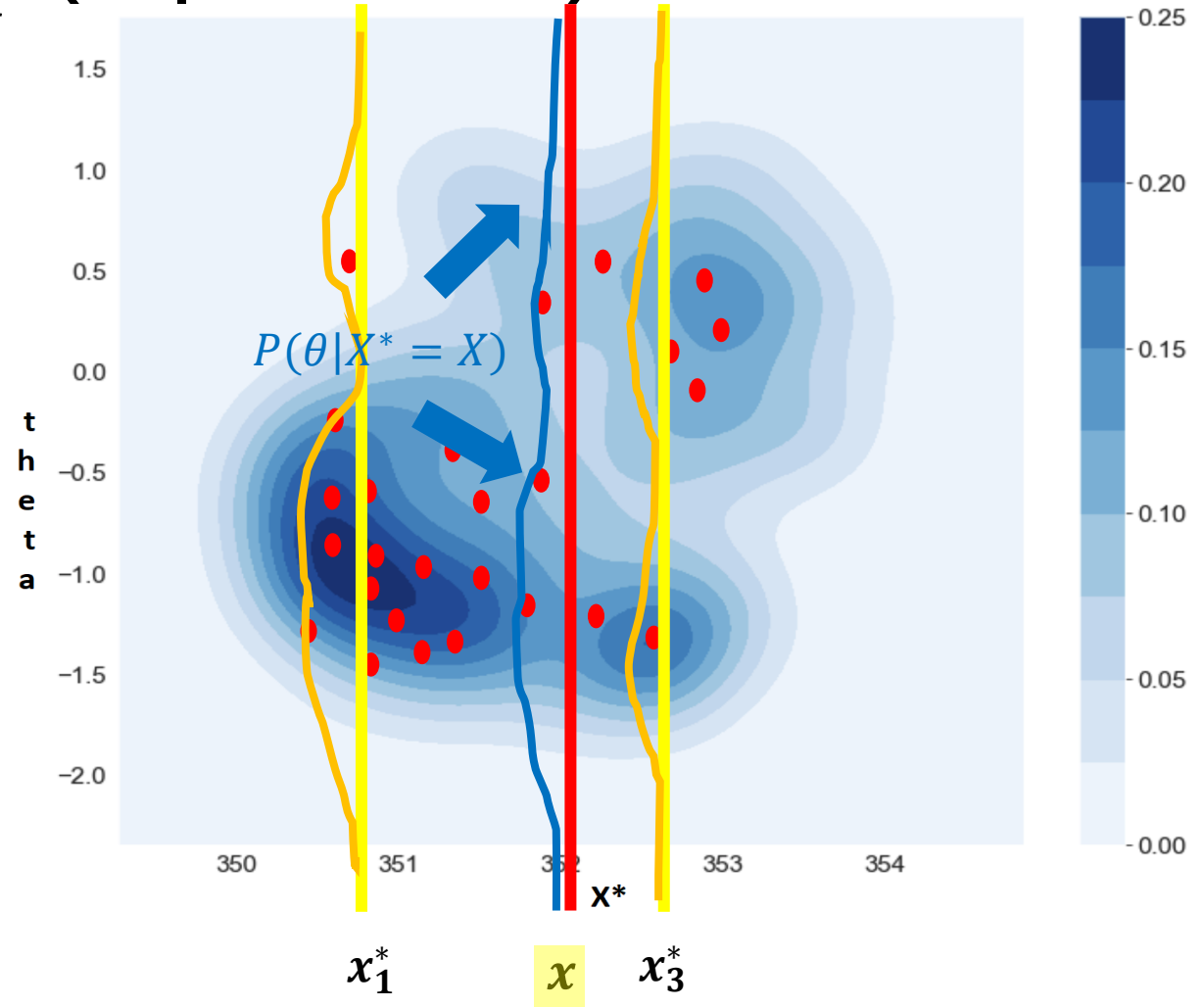
- Condition all the parameters in the mixture on X_1^* :
 - $P(\theta|X_1^*) = \sum_{j=1}^J \alpha_j(X_1^*) * N(\theta|\mu_j(X_1^*), \Omega_j(X_1^*))$
- One performs density estimation via the maximum likelihood objective function:

$$\max_{\text{wrt params of } \alpha_j, \mu_j, \Sigma_j} \sum_{i=1}^I \log(P_{\alpha_j, \mu_j, \Sigma_j}(\theta_i|X_i^*))$$



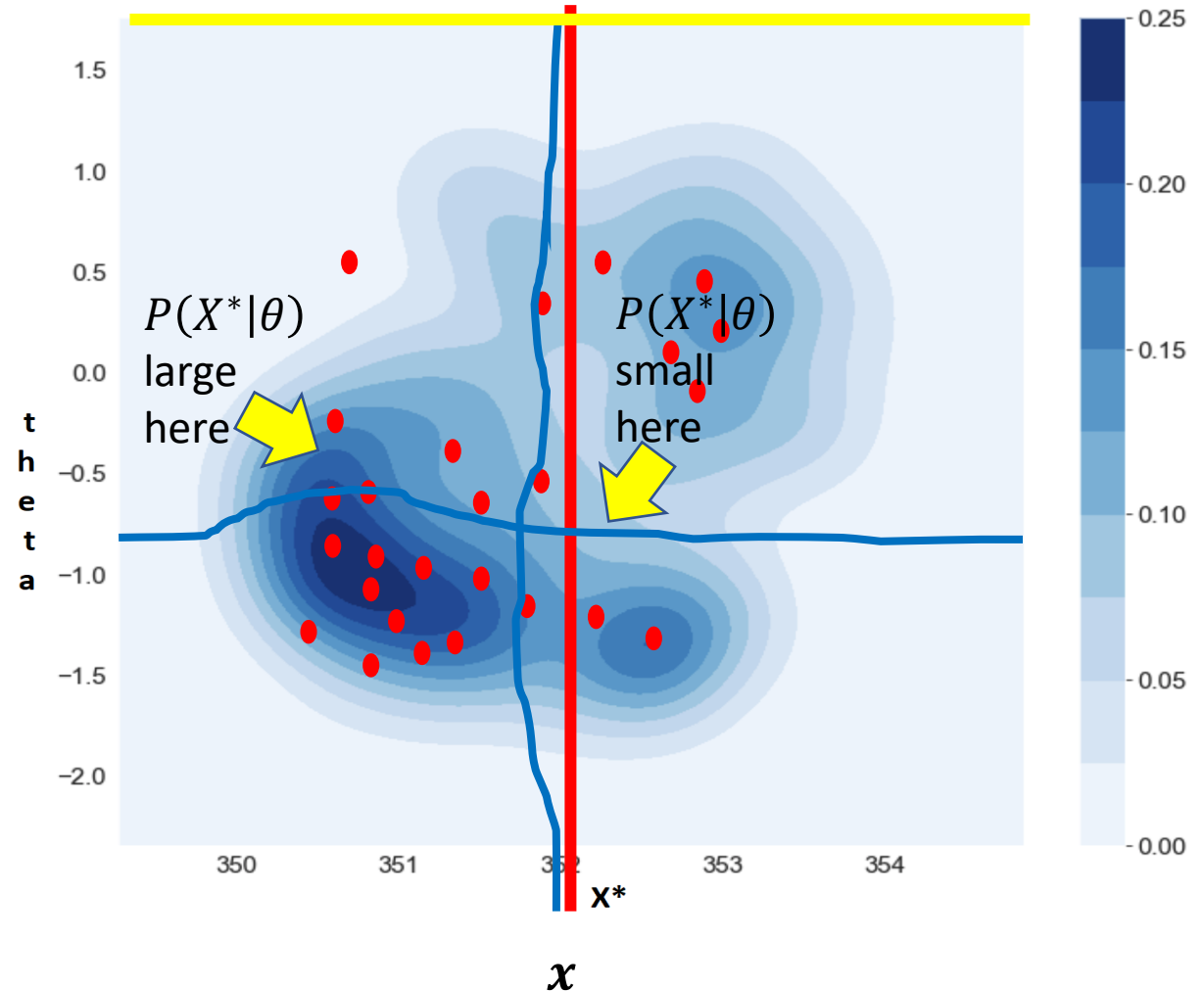
Step 3: Condition X^* on the real data, X , forming the posterior: $p(\theta|X^* = X)$

- Now that you have $p(\theta|X^*)$...
- ...The posterior is a conditional distribution of θ implied by the model conditioned on the specific X^* that corresponds to the real-world data, X :
 - $P(\theta|X^* = X)$



Step 3.5: Multi-Round Inference

- When sampling from the prior $p(\theta)$, can sample in places where the likelihood of the real data is small:
 $P(X^* = X|\theta)$ small
- Thus going to sample X far away from the posterior giving little information about $P(\theta|X^* = X)$
- The solution is to perform multi-round inference where the best guess of the posterior becomes the proposal distribution for sampling θ in the next round



Alternative to Mixture of Gaussians: Normalizing Flows

SNPE Example: $X^* = \theta + \epsilon$

- X = data, X^* = model simulations, $p(\theta)$ = prior over θ , $\epsilon \sim N(0,1)$
- Objective: $p(\theta|X^* = X)$
- **Step 1:** $\{X_i^*, \theta_i^*\}_{i=1}^I$ = simulated via Monte Carlo
 1. Sample θ_i^* from $p(\theta)$
 2. Sample X_i^* from simulated model conditional on θ_i^* , $X_i^* = \theta_i^* + \epsilon$
- **Step 2:** Estimate the conditional density $p(\theta|X^*)$
 1. Use a conditional density estimator
 - Many different CDEs can be used, I will discuss a conditional mixture of Gaussians and normalizing flows
- **Step 3:** Condition X^* on the real data, X , forming the posterior: $p(\theta|X^* = X)$

Smets-Wouters 2007

- In Smets-Wouters 2007, θ is 36 dimensional
- For Gaussian Mixture: $P(\theta|X^*) = \sum_j^{j=k} \alpha_j(X^*) * N(\theta|\mu_j(X^*), \Sigma_j(X^*))$
- Each Σ_j is $R^{36*37/2} = R^{666}$
- Infeasible for large θ
- Alternative: Use normalizing flow for Step 2: Conditional Density Estimator

Change of Variables

- Given a random variable: $\theta \sim p(\theta)$, and a function f :

$$q(f(\boldsymbol{\theta})) = p(\boldsymbol{\theta}) \left| \det \frac{\delta f}{\delta \boldsymbol{\theta}} \right|^{-1}$$

- Given a series of transformations: $\boldsymbol{\theta}_K = f_K \circ f_{K-1} \circ \cdots f_1(\boldsymbol{\theta})$:

$$\ln(q(\boldsymbol{\theta}_K)) = \ln(p(\boldsymbol{\theta}_0)) - \sum_k^{k=K} \ln\left(\left| \det \frac{\delta f_k}{\delta \boldsymbol{\theta}_{k-1}} \right| \right)$$

- Given this information, we can create a series of transformations f_i that can take a parametric distribution like $p(\boldsymbol{\theta}_0) = N(\bar{\mathbf{0}}, \bar{\mathbf{1}})$, and morph it into a broad family of distributions

Talking Syntax

- Some syntax:
 - Call f_k a bijector (because it will be a bijection...more to follow), a stack of f_k 's is a flow
 - Subscript variables like the k in f_k and z_k correspond to the composition level in the equations: $\theta_K = f_K \circ f_{K-1} \circ \dots \circ f_1(\theta_0)$:
 - Superscript variables, θ_k^j and f_k^j which I introduce here, indicate the element in the vector θ_k or f_k , as θ_k is a vector and f_k is a vector valued function
 - Next, will discuss the structure of individual f_k with the understanding that we will compose many f_k 's together to form a flow

Properties of a good f_k

- In order to sample from θ_K need to go from $\theta_0 \rightarrow \theta_K \dots$
- In order to calculate the probability of θ_K , need to go from $\theta_K \rightarrow \theta_0$
- Thus, f_k must be a bijection
- A simple, but inflexible function that is bijection is an affine transformation: $\theta_{k+1} = a_k * \theta_k + b_k$ and $\theta_k = \frac{\theta_{k+1} - b_k}{a_k}$

Masked Autoregressive Flows

- The following will discuss Masked Autoregressive Flows
- In order to make the bijector more expressive, we can condition the affine parameters on θ_i^j
- To preserve invertibility condition on θ_k like this:

$$\theta_{k+1}^0 = a_k^0 * \theta_k^0 + b_k^0$$
$$\theta_{k+1}^{j>0} = a_k^j(\theta_k^{j-1} \dots \theta_k^0) * \theta_k^j + b_k^j(\theta_k^{j-1} \dots \theta_k^0)$$

Masked Autoregressive Flows: Preserving Invertibility

- Given the forward equations

$$\theta_{k+1}^0 = a_k^0 * \theta_k^0 + b_k^0$$

$$\theta_{k+1}^{j>0} = a_k^j(\theta_k^{j-1} \dots \theta_k^0) * \theta_k^j + b_k^j(\theta_k^{j-1} \dots \theta_k^0)$$

- Finding θ_{k+1} from θ_k requires evaluating the forward equations
- If one knows only knows θ_{k+1} , to find θ_k

$$\theta_k^0 = \frac{\theta_{k+1}^0 - b_k^0}{a_k^0}$$

$$\theta_k^1 = \frac{\theta_{k+1}^1 - b_k^1(\theta_k^0)}{a_k^1(\theta_k^0)}$$

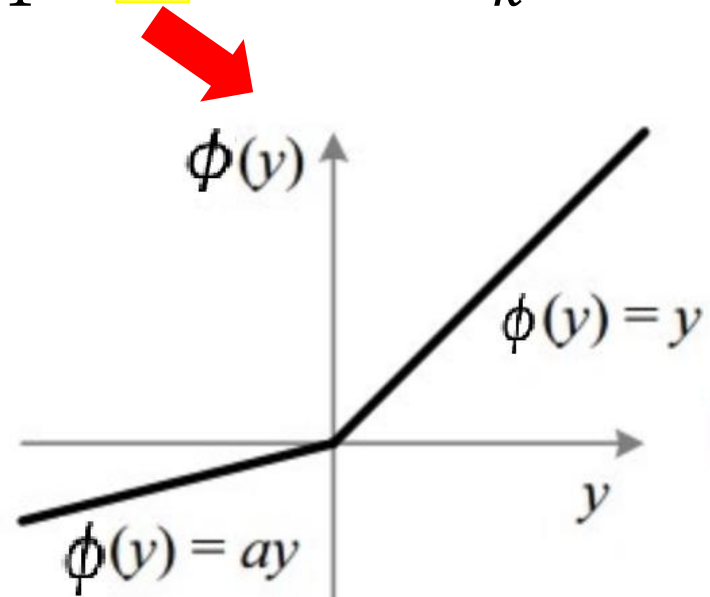
.

.

Non-Linearity

- A composition of affine functions is still affine
- In order to make stacks of bijectors more expressive, add a bijective nonlinear 'link' function between each bijector:

$$\theta_{k+1}^j = \phi(a^j(.) * \theta_k^j + b^j(.))$$



Universal Approximating Theorems

- With some small tweaks to the non-linearity and changing the conditioning order of each variable between bijectors, one can show that this normalizing flow can approximate any continuous distribution (Huang et. al. 2018)
- If you add additional conditioning variables into the input of $a(\cdot)$ and $b(\cdot)$ you can make this into a conditional density estimator
- Flow density estimation involves maximizing $\sum_i^{i=I} \ln(q(\{\boldsymbol{\theta}_K\}_i))$, where $\ln(q(\{\boldsymbol{\theta}_K\}_i))$ comes from:

$$\ln(q(\boldsymbol{\theta}_K)) = \ln(p(\boldsymbol{\theta}_0)) - \sum_k^{k=K} \ln(|\det \frac{\delta f_k}{\delta \boldsymbol{\theta}_{k-1}}|)$$

Normalizing Flows as Density Estimators

- In this case, we can estimate $P(\theta|X^*)$ using the normalizing flow as a density estimator for the conditional distribution:

$$\ln(q(\boldsymbol{\theta}_K|X^*)) = \ln(p(\boldsymbol{\theta}_0|X^*)) - \sum_k^{k=K} \ln(|\det \frac{\delta f_k(\boldsymbol{\theta}_{k-1}|X^*)}{\delta \boldsymbol{\theta}_{k-1}}|)$$

- One advantage of a flow-based density estimation is you can both sample from the flow and calculate probabilities without resorting to rejection sampling techniques like you would for a kernel density estimator

SNPE Example Redux: $X^* = \theta + \epsilon$

- X = data, X^* = model simulations, $p(\theta)$ = prior over θ , $\epsilon \sim N(0,1)$
- Objective: $p(\theta|X^* = X)$
- **Step 1:** $\{X_i^*, \theta_i^*\}_{i=1}^I$ = simulated via Monte Carlo [Simulate diagram](#)
 1. Sample θ_i^* from $p(\theta)$
 2. Sample X_i^* from simulated model conditional on θ_i^* , $X_i^* = \theta_i^* + \epsilon$
- **Step 2:** Estimate the conditional density $p(\theta|X^*)$ [Estimate diagram](#)
 1. Use a conditional density estimator
 - Many different CDEs can be used, I will discuss a conditional mixture of Gaussians and normalizing flows
- **Step 3:** Condition X^* on the real data, X , forming the posterior:
 $p(\theta|X^* = X)$ [Condition diagram](#)

Results

Comparison of Estimation Algorithms Ability to Handle a Selection of Solution Methods

Symbol	Meaning
✓	Model can do this without difficulties
~	Model can do inaccurately or with some modifications
?	Unknown based on lack of research
✗	Model not designed to do this

Solution Method/Algorithm	Details	SNPE	MCMC	SMC	MSM
Vanilla Perturbation	Posterior Quality	✓	~	✓	✗
	Ability to Estimate	✓	✓	✓	~
Vanilla Projection	Posterior Quality	✓	~	✓	✗
	Ability to Estimate	✓	~	~	~
Value Function Iteration	Posterior Quality	✓	~	✓	✗
	Ability to Estimate	~	~	~	~
HANK Reiter	Posterior Quality	✓	~	~	✗
	Ability to Estimate	✓	~ / ✗	~ / ✗	~
HANK Winberry	Posterior Quality	✓	~	~ / ?	✗
	Ability to Estimate	✓	~	~ / ?	~

[Perturbation](#)

[Projection](#)

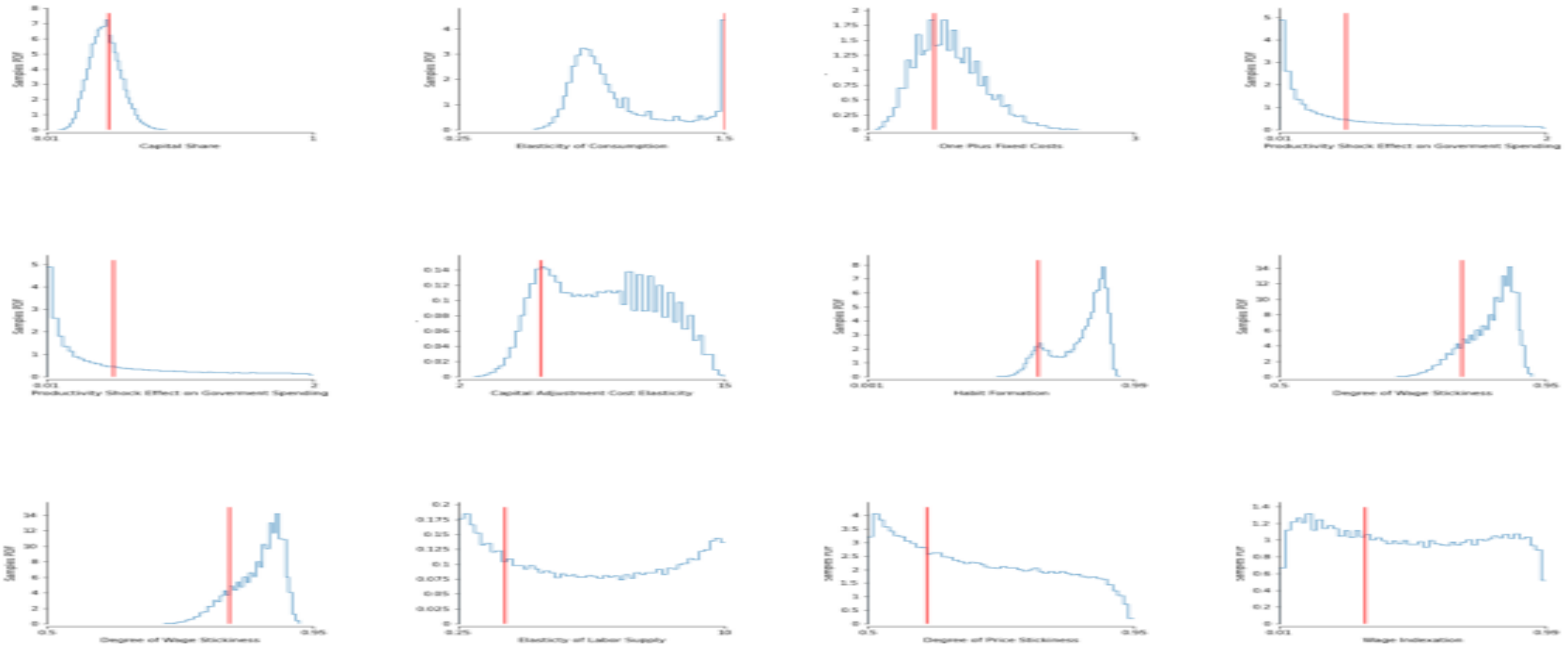
[Value Function Iteration](#)

[HANK Reiter](#)

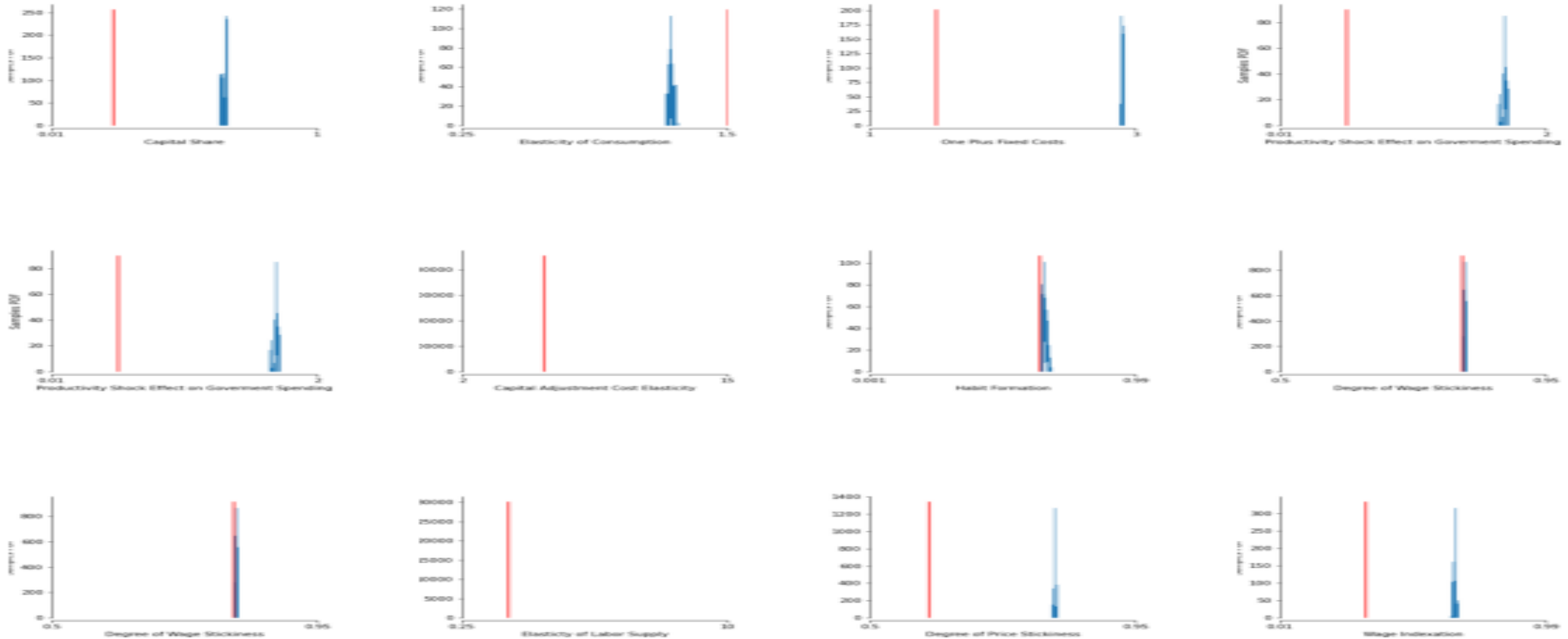
[HANK Winberry](#)

SNPE/MCMC Performance on Smets-Wouters

SNPE after Five Hundred Thousand Iterations



Metropolis-Hastings MCMC after Ten Million Iterations



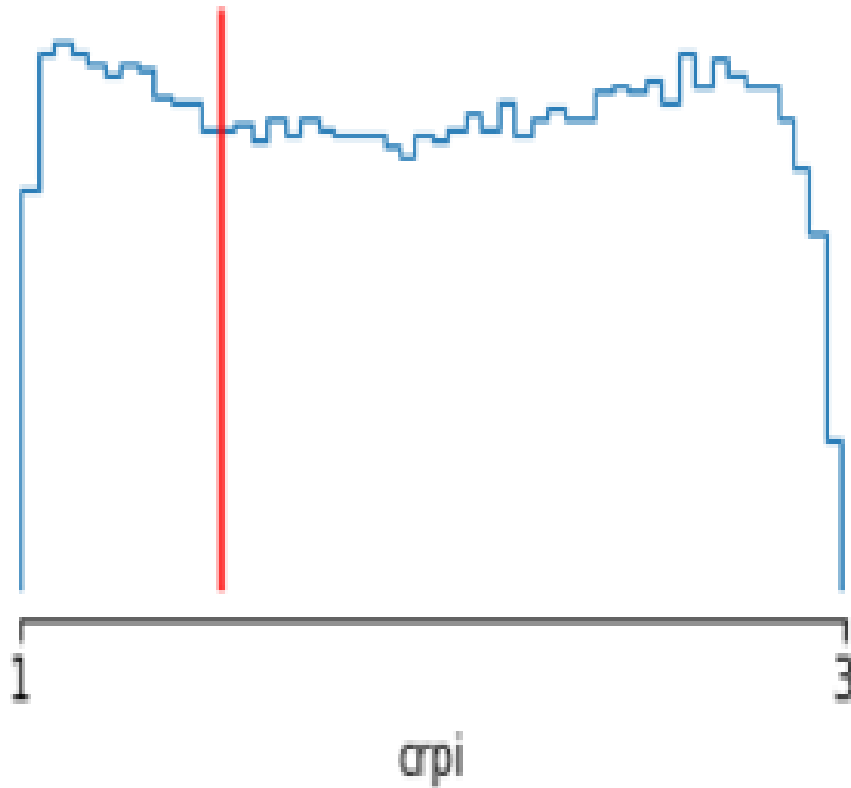
[Back](#)

Discussion on Multimodality

Multimodality and Frequentism

- Weak identification and multimodality is a problem for frequentist statistics → what is the right parameter value...
- ...but is just another posterior in the Bayesian approach
- Furthermore, multimodality might provide important feedback to economic research

Taylor Rule Parameter on Inflation



- $Crpi$, the Taylor rule parameter on inflation is unidentified
- SW and current data cannot distinguish between the two stories of central bank inflation fighting ability

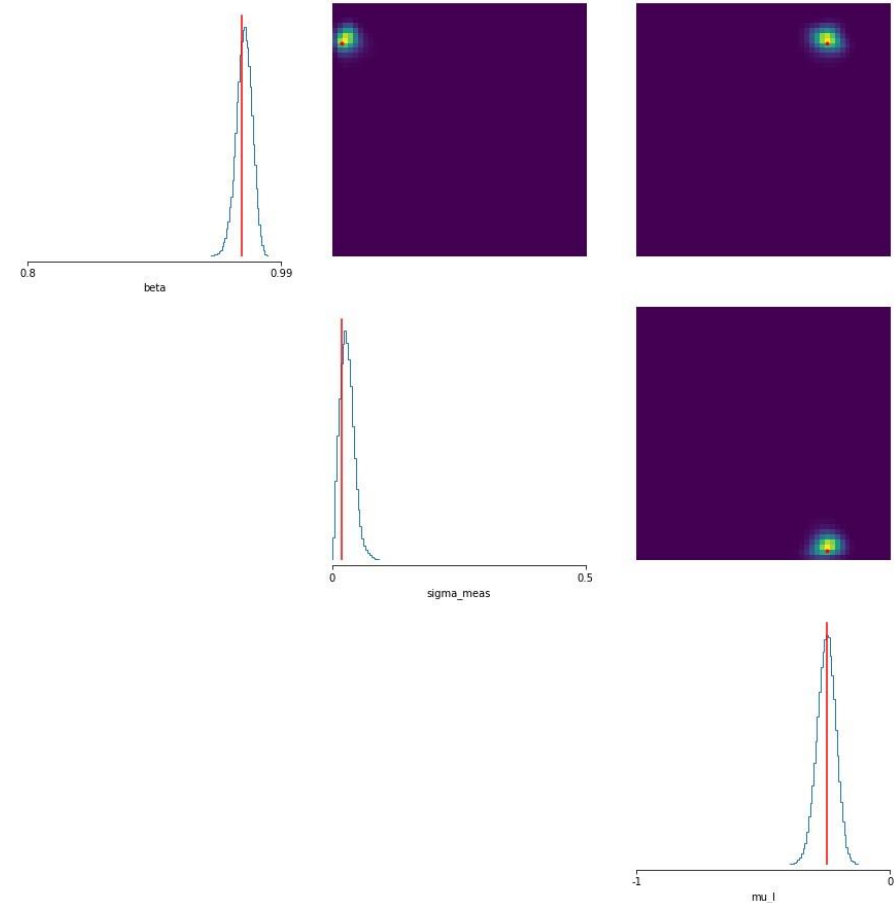
Multimodality and economic cycles

- Like the Diamond Dybvig (1983) model of bank runs, it doesn't require much imagination to think we coordinate on multiple equilibria
- In good times we coordinate on parameter values that lead to high output, but perhaps there is a low consumption, low production equilibrium that we could coordinate on if we get knocked out of the good equilibrium
- A deeper understanding of equilibrium coordination, by its nature involves deeper understanding of the multiple modes of parameter posteriors

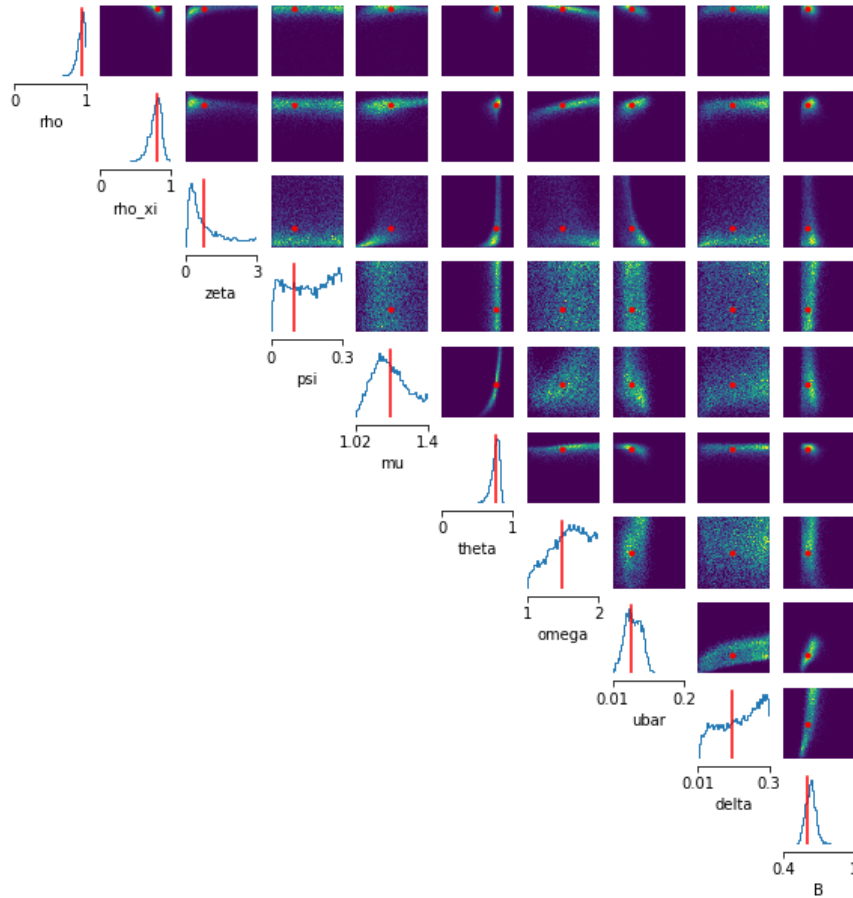
SNPE Performance on Heterogenous Agent Models

SNPE and MCMC Performance on Krussell Smith with Winberry Solution Method

- Following Liu and Plagborg-Moller (2021), likelihood estimate KS with all micromoments
- Liu and Plagborg-Moller (2021) is state of the art, taking 6.5 min. per CPU iteration
- My approach that doesn't have to solve for a likelihood function takes 30 seconds per CPU iteration



HANK Model Solved via Reiter



- Using Reiter's method, this HANK model has 1000+ states
- Easy to simulate from, but takes 20-30 mins to use the Kalman filter for a single likelihood evaluation, which is intractable
- Only other option is to use dimensionality reductions which loses information (Ahn et. al. 2018)

Appendix


Literature

- Many models are non-smooth so one can't use perturbation and thus can't likelihood estimated without measurement error
 - **s-S models:** Arrow et. al. 1951, Caplin and Spulber 1987, Bertola and Caballero 1990, House and Leahy 2004, Caplin and Leahy 2006, Kahn and Thomas 2008, Caplin et. al. 2020, Badoczy 2022, etc.
 - **Structural Finance:** He, Whited, and Guo 2021, Terry, Whited, and Zakolyukina 2021, Taylor 2012, Whited and Guo 2006
- With non-smooth models: One generally must resort to simulation-based estimation:
 - **Method of Simulated Moments:** McFadden 1989, Pakes and Pollard 1989, Duffie and Singleton 1990, etc
 - **Machine Learning:** Kaji, Manresa, and Pouliot 2020

[Back](#)

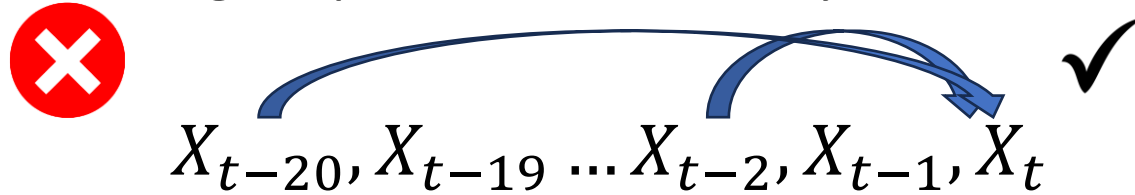
Markov Assumption Versus State-Space Assumption

- With MSM, we impose the Markov Assumption on the data
 - A small number of lags captures all necessary historical information:

$$X_{t-20}, X_{t-19} \dots X_{t-2}, X_{t-1}, X_t$$


Markov Assumption Versus State-Space Assumption

- With MSM, we impose the Markov Assumption on the data
 - A small number of lags captures all necessary historical information:



Markov Assumption Versus State-Space Assumption

- With MSM, we impose the Markov assumption on the data
 - A small number of lags captures all necessary historical information:



$X_{t-20}, X_{t-19} \dots X_{t-2}, X_{t-1}, X_t$

- With the state-space assumption, no information, no matter how back the data is, is not being conditioned, using the state of the model:



$X_{t-20}, X_{t-19} \dots X_{t-2}, X_{t-1}, X_t$



Flow Universal Approximator Intuition

- From Huang 2018
- First a sum of step functions \rightarrow universal approximator of monotonic functions
- A logit can approximate a step function arbitrarily well \rightarrow a sum of logit can do the same with monotonic functions
- A neural autoregressive flow can universally approximate any set of parameters and thus when using a logit nonlinearity, can approximate logits \rightarrow step functions \rightarrow monotonic functions arbitrarily well

Monotonic Functions and Universal Approximators

- One can map any uniform density to any distribution via the monotonic CDF
- One can map any distribution to the uniform density via the inverse CDF which also is monotonic
- Since you can map any distribution to a uniform via monotonic transformations and any uniform to any other distributions via monotonic transformation, then monotonic distributions can map any distribution to any other → universal approximation with flows

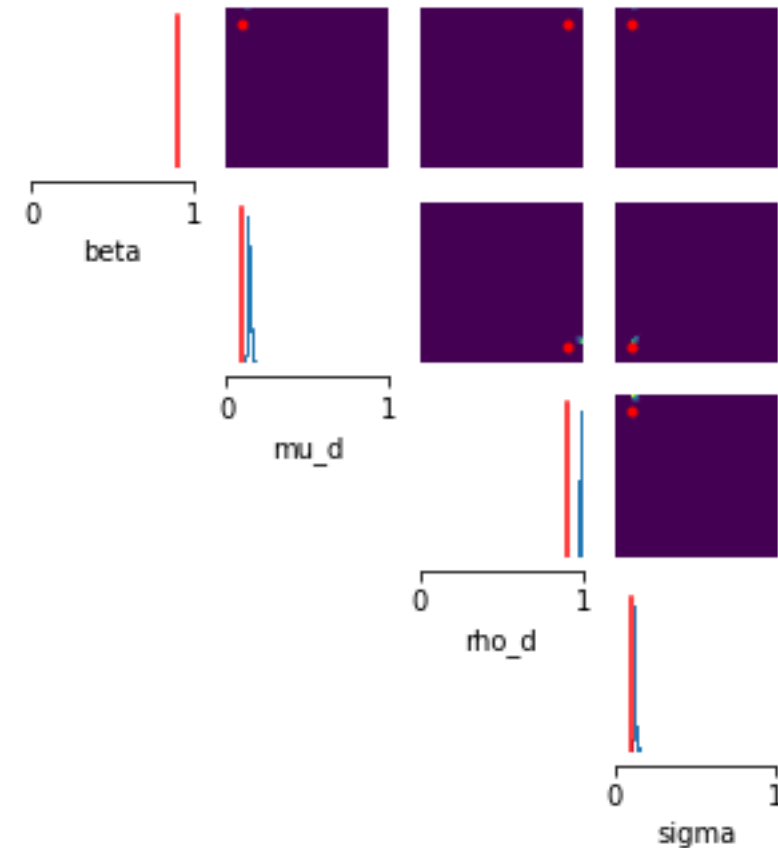
[Back](#)

Lossless Embedding

- I pointed out that in comparison to (Ahn et. al. 2018), the use of an end to end embedding generally leads to lossless compression
- Since a small number of parameters generate many sample data points, there likely exists a sufficient statistic the size of the parameter set
 - Modulo continuity and other regularity conditions
- By Rao-Blackwell, if there exists a sufficient statistic, it will maximize likelihood
 - Caveat regarding global versus local mins
- A universal approximator embedding neural network, can arbitrarily approximate any function so given enough parameters can approximate the sufficient statistic and will approximate the sufficient statistic as that will minimize MLE loss
- Using PCA like in Ahn et.al. (2018) will not be lossless as it's not trained end to end with the Bayesian objective

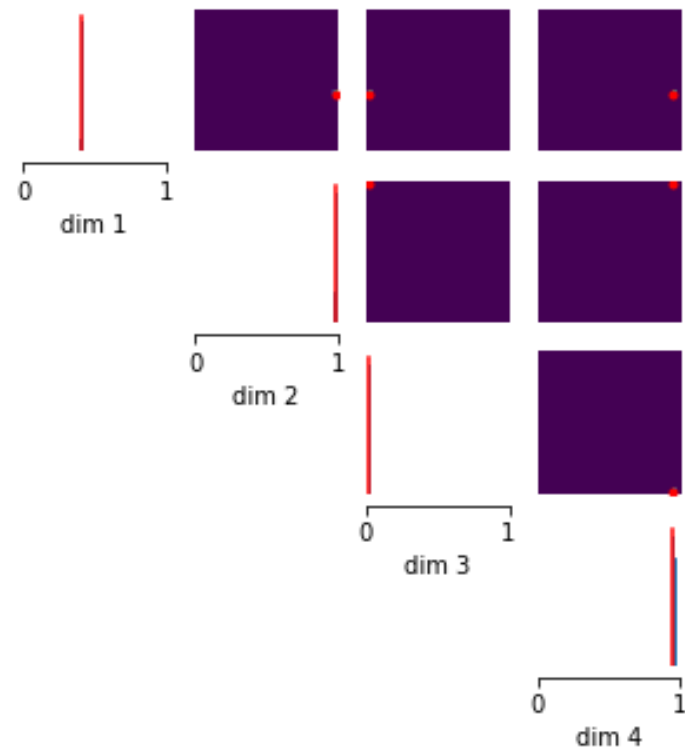
Projection Model

- Lucas-Tree estimated via projection
- Can see some errors with the posterior not quite aligned with the data generating parameters
- No measurement error used when estimating

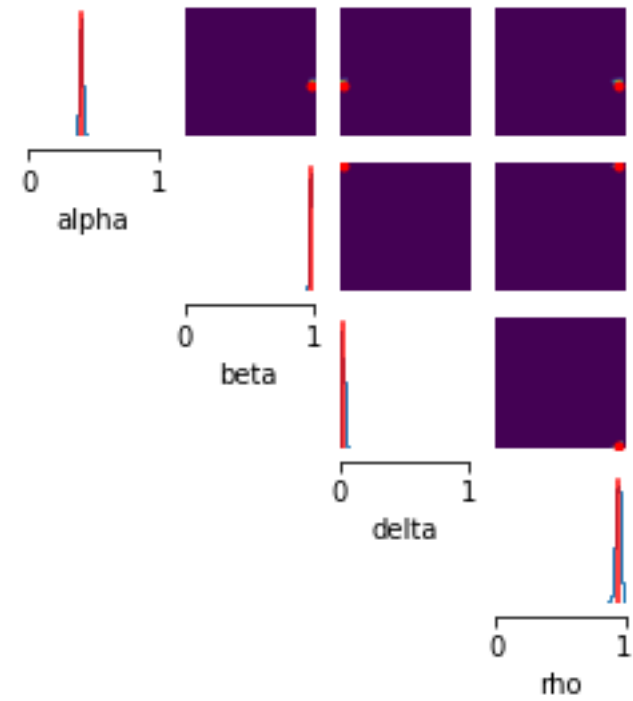


RBC Model Estimated Two Ways

MCMC Ground Truth Posterior

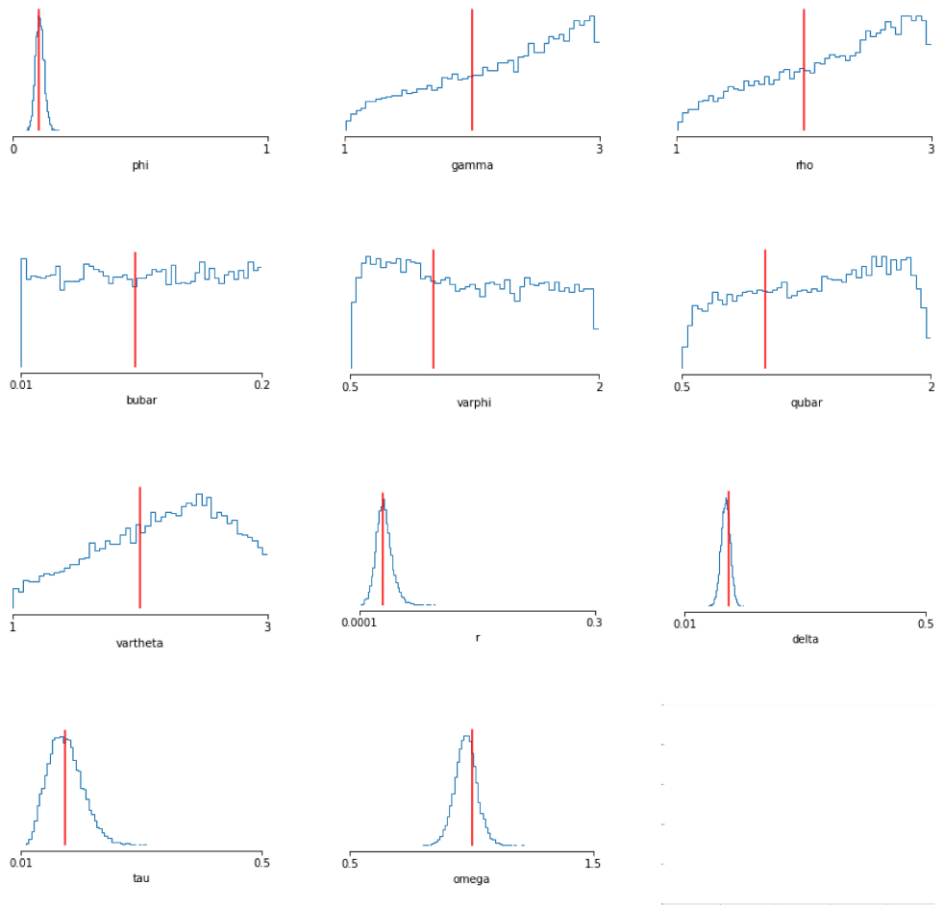


SNPE Posterior



[Back](#)

Bequests Value Function Iteration Model



- Liquid and Illiquid asset
 - Kink at the transition point → No perturbation
- Solved via value function iteration
- Using measurement error

[Back](#)

SNPE

Algorithm 1: SNPE Algorithm

Input: Simulator $p(X^*|\theta)$, prior $p(\theta)$, data X , Density Estimator $p_\phi(X^*|\theta)$, Rounds R , Samples S ;

Initialize: Posterior $p^{(0)}(\theta) = p(\theta)$, data set $D = \{\}$;

for $i \leftarrow 1$ **to** R **do**

 Sample $\theta^{*(n)} \sim p^{(i-1)}$ for $n = 1 \dots S$ with Monte Carlo;

 Simulate $X^{*(n)} \sim P(X^*|\theta^{*(n)})$ for $n = 1 \dots S$;

 Concatenate data $D = D \cup \{X^{*(n)}, \theta^{*(n)}\}_{n=1}^S$;

while $p_\phi(\theta|X^*)$ *not converged* **do**

 Sample $\{X^{*(i)}, \theta^{*(i)}\}_i^B \sim D$ from D ;

 Train $p_\phi(\theta|X^*) \frac{p^{(i-1)}(\theta)}{p(\theta)}$ on $\{X^{*(i)}, \theta^{*(i)}\}_i^B$;

end

 Update posterior $p^{(i)}(\theta) = p_\phi(\theta|X^* = X)$;

end

[Back](#)

Alternative Algorithm: Sequential Neural Variational Inference

Algorithm 3: SNVI Algorithm

Input: Simulator $p(x|\theta)$, prior $p(\theta)$, data x_0 , discriminator $d_\phi(x, \theta)$, Rounds R , Samples S ;

Initialize: Posterior $p^{(0)} = p(\theta)$, data set $D = \{\}$;

for $i \leftarrow 1$ **to** R **do**

 Sample $\theta^{(n)} \sim f_{\phi'}^{(i-1)}(\theta)$ for $n = 1 \dots S$;

 Simulate $x^{(n)} \sim P(x|\theta^{(n)})$ for $n = 1 \dots S$;

 Concatenate data $D = D \cup \{x^{(n)}, \theta^{(n)}\}_{n=1}^S$;

while $d_\phi(x, \theta)$ not converged **do**

 Sample $\{x^{(i)}, \theta^{(i)}\}_i^B \sim D$ from D ;

 Sample K contrasting samples for each i sample

$\{x'^{(k)}\}_k^{B*K}, \{\theta'^{(k)}\}_k^{B*K} \sim D\{x\}, D\{\theta\}$, breaking the joint distribution of D into independent marginals;

 Train the GAN using the GAN loss function from the two distributions x, θ and x', θ' and a softmax/multinomial logit objective $\sum_i^B \log\left(\frac{\exp(d_\phi(x^{(i)}, \theta^{(i)}))}{\sum_k^K \exp(d_\phi(x^{(k)}, \theta^{(k)}))}\right)$;

end

 Train normalizing flow with variational inference

$f_{\phi'}^i = \operatorname{argmin}_{\phi'} (KL(f_{\phi'}^i(\theta) || \exp(d_\phi(x, \theta))p(\theta))$;

end

Table Runtimes

- Clean up slides
- SMM comparison table
- What does the model do (Reiter)? Background, what can I do with the posteriors