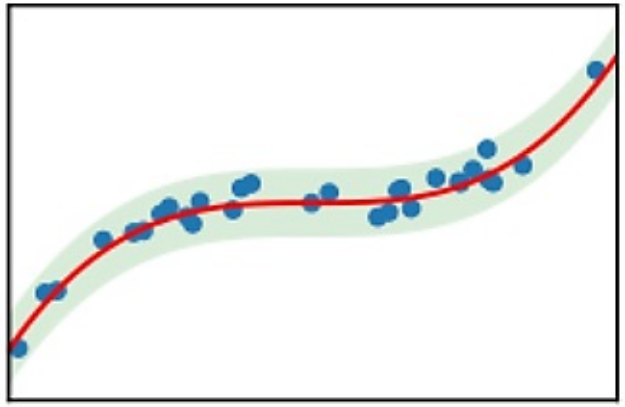
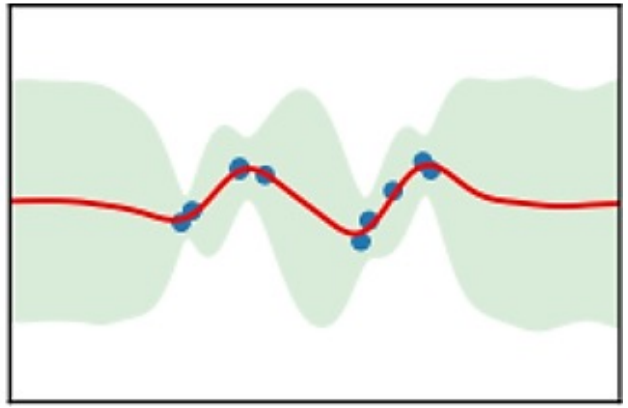
Fast ε-free Inference of Simulation Models with Bayesian Conditional Density Estimation

Types of Uncertainty

- There are two major kinds of uncertainty
- **Epistemic** Uncertainty describes what the model doesn't know. It is attributed to inadequate knowledge of the model. This is the uncertainty which can be reduced by having more data or increasing the model complexity.
- **Aleatoric** Uncertainty is the inherent uncertainty which is part of the data generating process. For example, a paper plane which is launched by a high precision equipment, which maintains the same degree of release, speed of release and a thousand other parameters will still not fall in the same place each trial. This inherent variability is Aleatoric Uncertainty.



Aleatoric Uncertainty



Epistemic Uncertainty

Illustration of aleatoric and epistemic uncertainty. Blue dots are the data points, red lines are the predictions, and the green shades is the ±3 Standard deviation around the prediction. Aleatoric uncertainty captures the noise in the dataset and is thus constat in the case if a data set with homoscedastic noise pictured above. Meanwhile, epistemic uncertainty captures the uncertainty of the model and thus decrease when more data points are observed

Bayesian Theorem

Conditional probability of a particular parameter value θ given data D to the probability of D given θ

- Refers to the process of determine the best data distribution given a specific situation in data.
- Is used to generally maximize the chance of a particular situation to occur.
- It is computationally expensive or sometimes completely infeasible to evaluate

- Represents beliefs about θ before D is available
- Often specified by choosing a tractable distribution such that random generation of values of θ are straight forward.

Likelihood

Prior

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

Posterior

The Evidence

- The probability of an event after taking in consideration the evidence.
- Can be calculated be approximated using ABC

- A collection of observations.
- It is what is being observed and measured

Posterior in Practice

Approximation / Simulation methods

- Maximum A Posteriori (MAP) Estimation
- Full Predictive Distribution
- Approximate Predictive Distribution
- Monte Carlo Dropout
- Stochastic Weight Averaging - Gaussian (SWAG)

Introduction

- Many Statistical models can be simulated forwards but have intractable likelihood as likelihood function is of central importance.
- For Simple models, an analytical formula for the likelihood function can typically be derived
- For complex models, an analytical formula might be very costly to evaluate computationally
- Approximate Bayesian Computation (ABC) methods are used to infer properties of these models from data
- Traditionally these methods approximate the posterior over parameters by conditioning on data being inside an ε-ball around the observed data, which is only correct in the limit $\varepsilon \to 0$.
- Monte Carlo methods can then draw samples from the approximate posterior to approximate predictions or error bars on parameters. These algorithms critically slow down as $\varepsilon \to 0$, and in practice draw samples from a broader distribution than the posterior

Simulation Based Modeling

- A simulator-based model is a data-generating process described by a computer program, usually with some free parameters we need to learn from data.
- Simulator-based modelling lends itself naturally to scientific domains (y biology, ecology, disease epidemics, economics and cosmology)
- The application domains mentioned can require properly calibrated distributions that express uncertainty over plausible parameters, rather than just point estimates, to reach scientific conclusions or make decisions.
- As an analytical expression for the likelihood of parameters given observations is typically not available for simulator-based models, conventional likelihood-based Bayesian inference is not applicable.
- An alternative family of algorithms for likelihood-free inference has been developed, referred to as **Approximate Bayesian Computation (ABC)**.
- These algorithms simulate the model repeatedly and only accept parameter settings which generate synthetic data like the observed data, typically gathered in a real-world experiment

Simulation Based Modeling

- Rejection ABC, the most basic ABC algorithm, simulates the model for each setting of proposed parameters, and rejects parameters if the generated data is not within a certain distance from the observations.
- The accepted parameters form a set of independent samples from an approximate posterior. Markov Chain Monte Carlo ABC (MCMC-ABC) is an improvement over rejection ABC which, instead of independently proposing parameters, explores the parameter space by perturbing the most recently accepted parameters.
- Sequential Monte Carlo ABC (SMC-ABC) uses importance sampling to simulate a sequence of slowly-changing distributions, the last of which approximates the parameter posterior
- Conventional ABC algorithms such as the above suffer from three drawbacks.
  - Only represent the parameter posterior as a set of (possibly weighted or correlated) samples. A sample-based representation easily gives estimates and error bars of individual parameters, and model predictions. However these computations are noisy, and it is not obvious how to perform some other computations using samples, such as combining posteriors from two separate analyses.
  - Second, the parameter samples do not come from the correct Bayesian posterior, but from an approximation based on assuming a pseudo-observation that the data is within an ε-ball centered on the data observed.
  - Third, as the ε-tolerance is reduced, it can become impractical to simulate the model enough times to match the observed data even once. When simulations are expensive to perform, good quality inference becomes impractical

likelihood-Free Inference

- Proposed a parametric approach to likelihood-free inference
- which unlike conventional ABC does not suffer from the above three issues.
- Instead of returning samples from an ε-approximation to the posterior, This approach learns a parametric approximation to the exact posterior, which can be made as accurate as required.
- Preliminary fits to the posterior are used to guide future simulations, which can reduce the number of simulations required to learn an accurate approximation by orders of magnitude
- This approach uses conditional density estimation with Bayesian neural networks, and draws upon advances in parametric density estimation, stochastic variational inference, and recognition networks, as discussed in the related work section

Bayesian Conditional Density Estimation For likelihood-free Inference

Simulator-based models and ABC

Θ → vector of parameters controlling a simulator-based model

X → be a data vector generated by the model.

p(x | ϑ) → likelihood

X_0 -> Given Observation

$$p(\theta \mid X - X_0 \propto P(x = X_0 \mid \theta)\, p(\theta)$$

- likelihood $p(x = x_o \mid \theta)$ is unavailable
- Conventional Bayesian inference cannot be carried out

Bayesian Conditional Density Estimation For likelihood-free Inference

Simulator-based models and ABC

- The principle behind ABC is to approximate $p(x = X_0 \mid \vartheta)$ *by* $p(|| X - X_0 < \varepsilon \mid \vartheta)$ for a sufficiently small value of ε ,and then estimate the latter
- (by Monte Carlo—using simulations)
- ABC approximates the posterior by $p(\vartheta || X - X_0 < \varepsilon \mid \vartheta)$, which is typically broader and more uncertain
- ABC can trade off computation for accuracy by decreasing ε , which improves the approximation to the posterior but requires more simulations from the mode
- the approximation becomes exact only when $\varepsilon \rightarrow 0$, in which case simulations never match the observations, $p(|| X - X_0 < \varepsilon \mid \vartheta) \rightarrow 0$, and existing methods break down

Bayesian Conditional Density Estimation For likelihood-free Inference

Learning the posterior

- $p(||X - X_0|| < \varepsilon | \vartheta)$
- use the simulations to directly estimate $p(\theta \mid X = X_0)$
- run simulations for parameters drawn from a distribution, $\tilde{p}(\theta)$
- form a consistent estimate of the exact posterior, using a flexible family of conditional densities, $q_\phi(\theta \mid x)$, parameterized by a vector ϕ

Proposition 1. *We assume that each of a set of N pairs $(\boldsymbol{\theta}_n, \mathbf{x}_n)$ was independently generated by*

$$\boldsymbol{\theta}_n \sim \tilde{p}(\boldsymbol{\theta}) \quad \text{and} \quad \mathbf{x}_n \sim p(\mathbf{x} \mid \boldsymbol{\theta}_n). \tag{1}$$

In the limit $N \to \infty$, the probability of the parameter vectors $\prod_n q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_n \mid \mathbf{x}_n)$ is maximized w.r.t. $\boldsymbol{\phi}$ if and only if

$$q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{x}) \propto \frac{\tilde{p}(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}\, p(\boldsymbol{\theta} \mid \mathbf{x}), \tag{2}$$

provided a setting of $\boldsymbol{\phi}$ that makes $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \mid \mathbf{x})$ proportional to $\frac{\tilde{p}(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}\, p(\boldsymbol{\theta} \mid \mathbf{x})$ exists.

Bayesian Conditional Density Estimation For likelihood-free Inference

Learning the posterior

Intuition:

if we simulated enough parameters from the prior, the density estimator q_ϕ would learn a conditional of the joint prior model over parameters and data, which is the posterior $p(\theta \mid x)$. If we simulate parameters drawn from another distribution, we need to "importance reweight" the result.

The proposition above suggests the following procedure for learning the posterior:

- propose a set of parameter vectors $\{\theta n\}$ from the proposal prior;
- for each θn run the simulator to obtain a corresponding data vector xn;
- train $q\phi$ with maximum likelihood on $\{\theta n, xn\}$; and
- estimate the posterior by

$$\hat{p}(\theta \mid X = X_0) \propto \frac{p(\theta)}{\tilde{P}(\theta)}\, q\phi(\theta \mid X_0)$$

Bayesian Conditional Density Estimation For likelihood-free Inference

Choice of conditional density estimator and proposal prior

conditional neural density estimation with ***Mixed density network (MDN)***

$$q_\phi(\theta \mid x) = \Sigma_k \alpha_k \, N(\theta \mid m_k, S_k)$$

- $\{\alpha_k\}$ = mixing coefficients
- $\{m_k\}$ = means
- $\{S_k\}$ = covariance matrices
- computed by a feedforward neural network parameterized by ϕ, taking x as input.

Mixture Density Networks

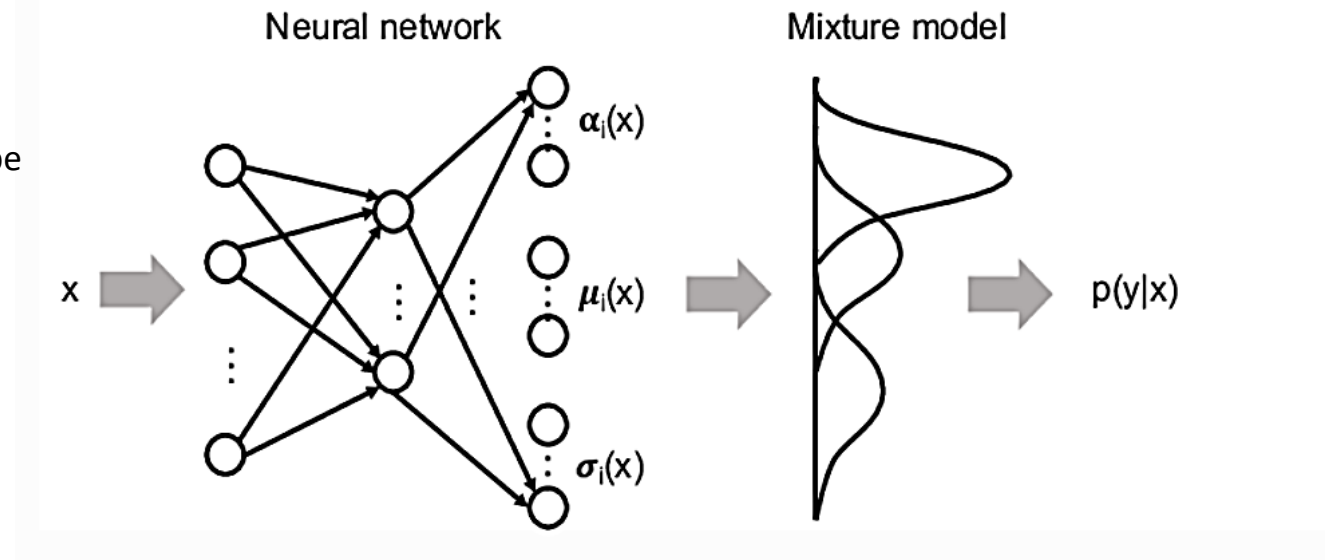
- Mixture Density Networks are built from two components – a Neural Network and a Mixture Model.
- The Neural Network can be any valid architecture which takes in the input \mathbf{X} and converts into a set of learned features

$\prod = \{\pi_0, \pi_1, \ldots, \pi_m - 1\}$ by the following equation:

$p(x) = \sum_{j=0}^{m-1} \pi_j p_j(x|\theta_j)$, where θ_j are the parameters of the distribution describing the shape and location of the distribution.

Mixture Density Networks

$$p(x|\Pi,\Theta) = \sum_{j=0}^{m-1} \pi_j \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\left(\frac{(x-\mu_j)^2}{2\sigma_j^2}\right)}$$



Mixture Density Network: The output of a neural network parametrizes a Gaussian mixture model

Sufficient Conditions

- The mixing coefficients (**π or α**) are probabilities and have to be less than one and sum to unity. This can be easily achieved by passing the outputs of the mixing coefficients through a *SoftMax layer.*
- The variance (**σ**) should be strictly positive. Bishop suggested that we use the exponential function to the raw logits of the sigma neuron. He suggested that this had the same effect as assuming an uninformative prior and avoids the pathological configurations in which one or more of the variances goes to zero.
- The center parameters **(μ)** represent location parameters, and this should be the raw logits of the mean neuron.

Loss Function

- loss function we are minimizing is the Negative Log Likelihood, which is equivalent to the Maximum Likelihood Estimation

Bayesian Conditional Density Estimation For likelihood-free Inference

Choice of conditional density estimator and proposal prior

To choose density estimator $q_\phi(\theta \mid x)$ and proposal prior $\tilde{p}(\theta)$, below criteria needs to be meet

- q_ϕ should be flexible enough to represent the posterior but easy to train with maximum likelihood
- $\tilde{p}(\theta)$ should be easy to evaluate and sample from; and
- the right-hand side expression in Posterior learning Eq should be easily evaluated and normalized $\left(\hat{p}(\theta \mid X = X_0) \propto \frac{p(\theta)}{\tilde{P}(\theta)}\, q\phi(\theta \mid X_0)\right)$

Bayesian Conditional Density Estimation For likelihood-free Inference

Choice of conditional density estimator and proposal prior

conditional neural density estimation with ***Mixed density network (MDN)***

$$q_\phi(\theta \mid x) = \sum_k \alpha_k \, N(\theta \mid m_k, S_k)$$

- provided the number of components K and number of hidden units in the neural network are sufficiently large—while remaining trainable by backpropagation.
- take the proposal prior to be a single Gaussian $\tilde{p}(\theta) = N(\theta \mid m_0, S_0)$, with mean m_0 and full covariance matrix S_0.
- Assuming the prior $p(\theta)$ is a simple distribution (uniform or Gaussian, as is typically the case in practice), then this choice allows us to calculate $\hat{p}(\theta \mid x = x_o)$ in $\left(\hat{p}(\theta \mid X = X_0) \propto \frac{p(\theta)}{\tilde{P}(\theta)} \, q\phi(\theta \mid X_0)\right)$
- That is, $\hat{p}(\theta \mid x = x_o)$ will be a mixture of K Gaussians, whose parameters will be a function of $\{\alpha_k, m_k, S_k\}$ evaluated at x_o

Bayesian Conditional Density Estimation For likelihood-free Inference

Learning the proposal prior

- Simple rejection ABC is inefficient because the posterior $p(\theta \mid x = x_o)$ is typically much narrower than the prior $p(\theta)$.
- A parameter vector θ sampled from $p(\theta)$ will rarely be plausible under $p(\theta \mid x = x_o)$ and will most likely be rejected.
- Practical ABC algorithms attempt to reduce the number of rejections by modifying the way they propose parameters
- for instance, MCMC-ABC and SMC-ABC propose new parameters by perturbing parameters they already consider plausible, in the hope that nearby parameters remain plausible

Bayesian Conditional Density Estimation For likelihood-free Inference

Algorithm 1: Training of proposal prior

initialize $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x})$ with one component
$\tilde{p}(\boldsymbol{\theta}) \leftarrow p(\boldsymbol{\theta})$
repeat
  for $n = 1..N$ **do**
    sample $\boldsymbol{\theta}_n \sim \tilde{p}(\boldsymbol{\theta})$
    sample $\mathbf{x}_n \sim p(\mathbf{x} \,|\, \boldsymbol{\theta}_n)$
  end
  retrain $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x})$ on $\{\boldsymbol{\theta}_n, \mathbf{x}_n\}$
  $\tilde{p}(\boldsymbol{\theta}) \leftarrow \frac{p(\boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta})} q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x}_o)$
until $\tilde{p}(\boldsymbol{\theta})$ *has converged*;

Algorithm 2: Training of posterior

initialize $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x})$ with K components
// if $q_{\boldsymbol{\phi}}$ available by Algorithm 1
// initialize by replicating its
// one component K times
for $n = 1..N$ **do**
  sample $\boldsymbol{\theta}_n \sim \tilde{p}(\boldsymbol{\theta})$
  sample $\mathbf{x}_n \sim p(\mathbf{x} \,|\, \boldsymbol{\theta}_n)$
end
train $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x})$ on $\{\boldsymbol{\theta}_n, \mathbf{x}_n\}$
$\hat{p}(\boldsymbol{\theta} \,|\, \mathbf{x} = \mathbf{x}_o) \leftarrow \frac{p(\boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta})} q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x}_o)$

Bayesian Conditional Density Estimation For likelihood-free Inference

Learning the proposal prior

Idea to set up a fixed-point system.

strategy - learn an efficient proposal prior that closely approximates the posterior as follows

- initially take $\tilde{p}(\theta)$ to be the prior $p(\theta)$
- propose N samples $\{\theta_n\}$ from $\tilde{p}(\theta)$ and corresponding samples $\{x_n\}$ from the simulator, and train $q_\phi(\theta \mid x)$ on them
- approximate the posterior using Equation (deck ref) and set $\tilde{p}(\theta)$ to it
- repeat until $\tilde{p}(\theta)$ has converged.

This procedure is summarized in *Algorithm 1*. In the procedure above, as long as $q_\phi(\theta \mid x)$ has only one Gaussian component (K = 1) then $\tilde{p}(\theta)$ remains a single Gaussian throughout. Moreover, in each iteration we initialize $q\phi$ with the density estimator learnt in the iteration before, thus we keep training q_ϕ throughout. This initialization allows us to use a small sample size N in each iteration, thus making efficient use of simulations.

Bayesian Conditional Density Estimation For likelihood-free Inference

Use of Bayesian neural density estimators

To make Algorithm 1 as efficient as possible, the number of simulations per iteration N should be kept small, while at the same time it should provide a sufficient training signal for q_ϕ. With a conventional MDN, if N is made too small, there is a danger of overfitting, especially in early iterations, leading to over-confident proposal priors and an unstable procedure. Early stopping could be used to avoid overfitting; however, a significant fraction of the N samples would have to be used as a validation set, leading to inefficient use of simulations.

Bayesian Conditional Density Estimation For likelihood-free Inference

Use of Bayesian neural density estimators

Bayesian version of the MDN using Stochastic Variational Inference (SVI) for neural networks.

- An MDN-SVI has two sets of adjustable parameters of the same size, the means ϕ_m and the log variances ϕ_s.
- The means correspond to the parameters ϕ of a conventional MDN.
- During training, Gaussian noise of variance $\exp \phi_s$ is added to the means independently for each training example (θ_n, x_n).
- The Bayesian interpretation of this procedure is that it optimizes a variational Gaussian posterior with a diagonal covariance matrix over parameters ϕ.
- At prediction time, the noise is switched off and the MDN-SVI behaves like a conventional MDN with $\phi = \phi_m$.

Bayesian Conditional Density Estimation For likelihood-free Inference

Use of Bayesian neural density estimators

using an MDN-SVI instead of an MDN improves the robustness and efficiency of Algorithm 1 because

- MDN-SVI is resistant to overfitting, allowing us to use a smaller number of simulations N
- no validation set is needed, so all samples can be used for training;
- since overfitting is not an issue, no careful tuning of training time is necessary.

Experiments : Mixture of two Gaussians

$$p(\theta) = \mathcal{U}(\theta \,|\, \theta_\alpha, \theta_\beta) \quad \text{and} \quad p(x \,|\, \theta) = \alpha \mathcal{N}\left(x \,|\, \theta, \sigma_1^2\right) + (1-\alpha)\mathcal{N}\left(x \,|\, \theta, \sigma_2^2\right),$$

where $\theta_\alpha = -10$, $\theta_\beta = 10$, $\alpha = 0.5$, $\sigma_1 = 1$, $\sigma_2 = 0.1$ and $x_o = 0$. The posterior can be calculated analytically, and is proportional to an equal mixture of two Gaussians centred at x_o with variances σ_1^2 and σ_2^2, restricted to $[\theta_\alpha, \theta_\beta]$.

Experiments : Mixture of two Gaussians

- shows the results of neural density estimation using each strategy.
- All MDNs have one hidden layer with 20 tanh units and 2 Gaussian components
- except for the proposal prior MDN which has a single component.
- Both MDN with prior and MDN with proposal learn good parametric approximations to the true posterior, and the proposal prior is a good Gaussian approximation to it.
- It used 10K simulations to train the MDN with prior, whereas the prior proposal took 4 iterations of 200 simulations each to train, and the MDN with proposal took 1000 simulations on top of the previous 800.
- The MDN with prior learns the posterior distributions for a large range of possible observations x (*middle plot of Figure 1*),
- whereas the MDN with proposal gives accurate posterior probabilities only near the value observed (*right plot of Figure 1)*

Experiments : Mixture of two Gaussians

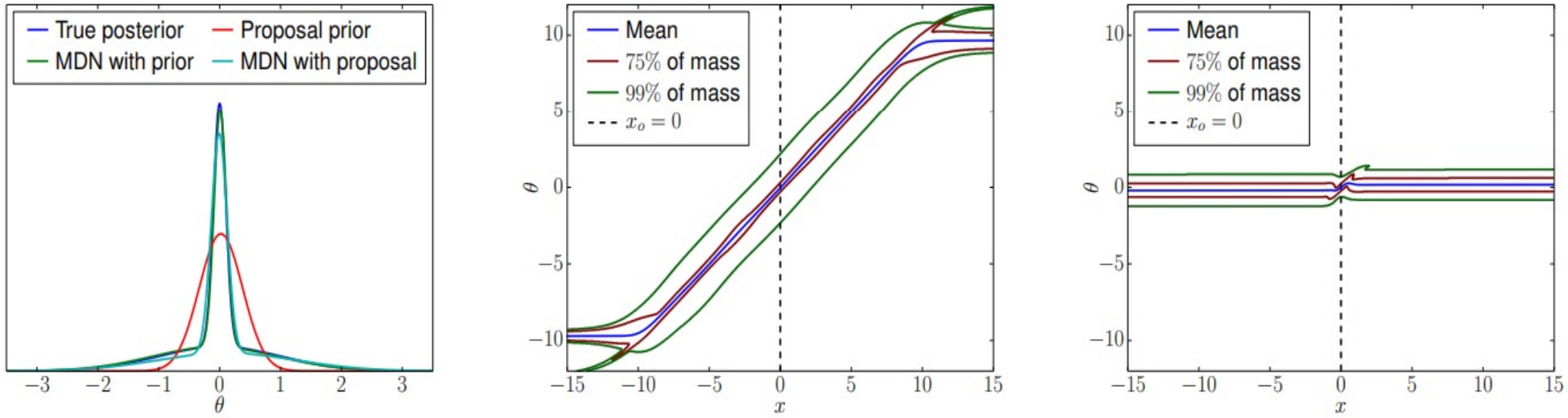


Figure 1: Results on mixture of two Gaussians. **Left**: approximate posteriors learnt by each strategy for $x_o = 0$. **Middle**: full conditional density $q_\phi(\theta|x)$ leant by the MDN trained with prior. **Right**: full conditional density $q_\phi(\theta|x)$ learnt by the MDN-SVI trained with proposal prior. Vertical dashed lines show the location of the observation $x_o = 0$.

Experiments : Bayesian Liner Regression

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} \,|\, \mathbf{m}, \mathbf{S}) \quad \text{and} \quad p(\mathbf{x} \,|\, \boldsymbol{\theta}) = \textstyle\prod_i \mathcal{N}\left(x_i \,|\, \boldsymbol{\theta}^T \mathbf{u}_i, \sigma^2\right),$$

where we took $\mathbf{m} = \mathbf{0}$, $\mathbf{S} = \mathbf{I}$, $\sigma = 0.1$, randomly generated inputs $\{\mathbf{u}_i\}$ from a standard Gaussian, and randomly generated observations $\mathbf{x}_o$ from the model.

Experiments : Bayesian linear regression

- As ε is decreased, ABC methods sample from an increasingly better approximation to the
- true posterior, however, they eventually reach their failing point or take prohibitively long. The best
- approximations are achieved by MDN with proposal and a very long run of SMC-ABC
- The middle of Figure 2 shows the increase in number of simulations needed to improve approximation quality (as decreases).
- We quote the total number of simulations for MDN training, and the number of simulations per effective sample for ABC.
- Section E of the supplementary material describes how the number of effective samples is calculated.
- The number of simulations per effective sample should be multiplied by the number of effective samples needed in practice.
- Moreover, SMC-ABC will not work well with only one particle, so many times the quoted cost will always be needed.
- Here, MDNs make more efficient use of simulations than Monte Carlo ABC methods. Sequentially fitting a prior proposal was more than ten times cheaper than training with prior samples, and more accurate

Experiments : Bayesian linear regression

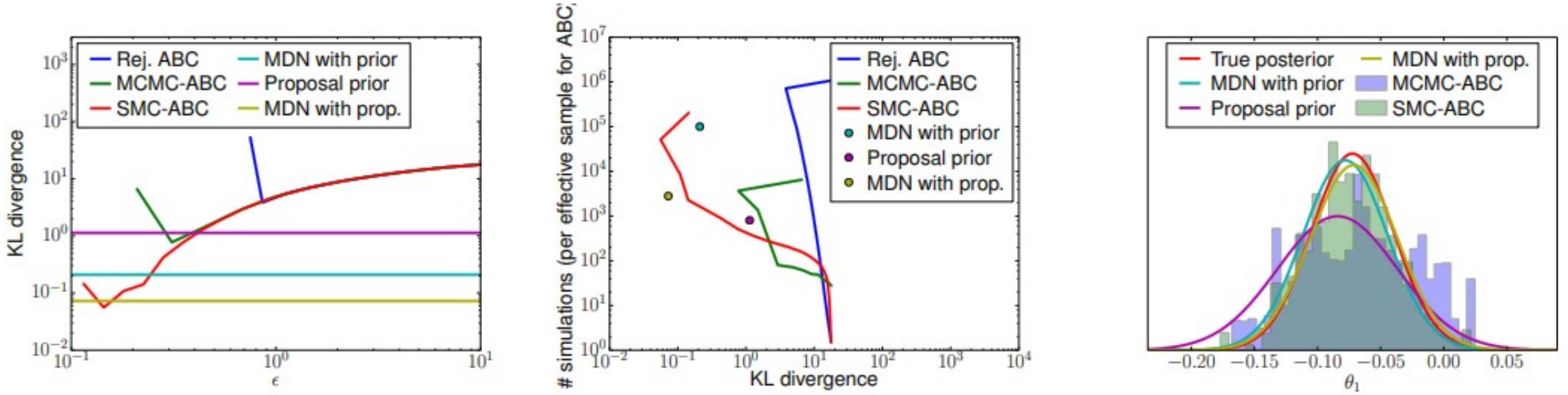


Figure 2: Results on Bayesian linear regression. **Left**: KL divergence from true posterior to approximation vs ϵ; lower is better. **Middle**: number of simulations vs KL divergence; lower left is better. Note that number of simulations is total for MDNs, and per effective sample for ABC. **Right**: Posterior marginals for θ_1 as computed by each method. ABC posteriors (represented as histograms) correspond to the setting of ϵ that minimizes the KL in the left plot.

Experiments : Lotka–Volterra predator-prey population model

The Lotka–Volterra model is a stochastic Markov jump process that describes the continuous time evolution of a population of predators interacting with a population of prey.
There are four possible reactions:

a predator being born
a predator dying
a prey being born
a prey being eaten by a predator.

Positive parameters $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$ control the rate of each reaction.

Given a set of statistics x_o calculated from an observed population time series, the objective is to infer θ.
We used a flat prior over $\log \theta$, and calculated a set of 9 statistics x.

The Lotka–Volterra model is commonly used in the ABC literature as a realistic model which can be simulated, but whose likelihood is intractable. One of the properties of Lotka–Volterra is that typical nature-like observations only occur for very specific parameter settings, resulting in narrow, Gaussian-like posteriors that are hard to recover

Experiments : Lotka–Volterra predator-prey population model

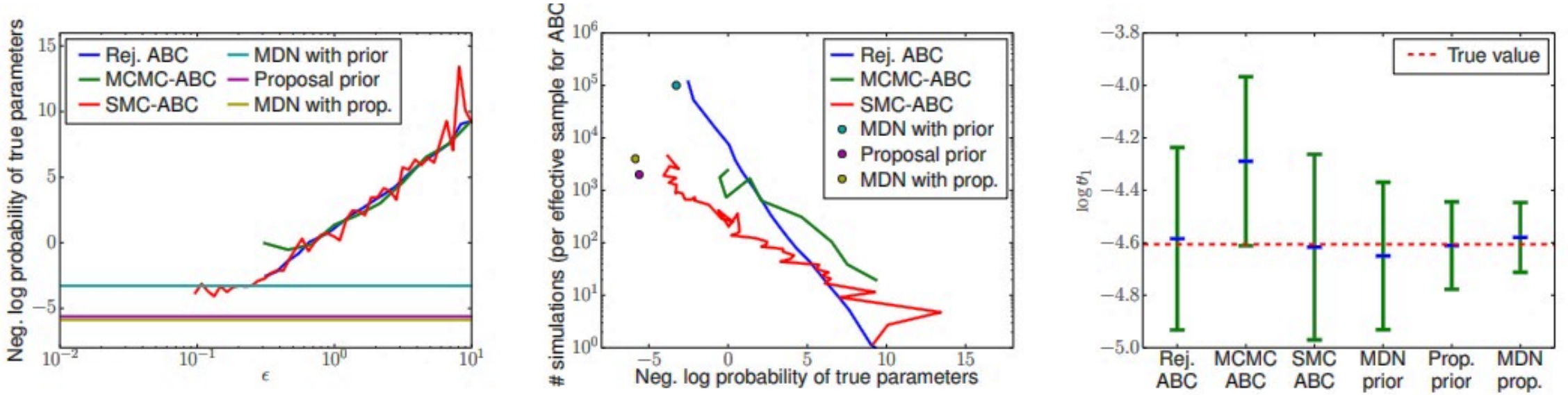


Figure 3: Results on Lotka–Volterra. **Left**: negative log probability of true parameters vs ϵ; lower is better. **Middle**: number of simulations vs negative log probability; lower left is better. Note that number of simulations is total for MDNs, but per effective sample for ABC. **Right**: Estimates of $\log \theta_1$ with 2 standard deviations. ABC estimates used many more simulations with the smallest feasible ϵ.

Experiments : M/G/1 queue model

- The M/G/1 queue model describes the processing of a queue of continuously arriving jobs by a single server.
- In this model, the time the server takes to process each job is independently and uniformly distributed in the interval $[\theta_1, \theta_2]$.
- The time interval between arrival of two consecutive jobs is independently and exponentially distributed with rate θ_3.
- The server observes only the time intervals between departure of two consecutive jobs. Given a set of equally-spaced percentiles xo of inter-departure times, the task is to infer parameters $\theta = (\theta_1, \theta_2, \theta_3)$.
- This model is easy to simulate but its likelihood is intractable, and it has often been used as an ABC benchmark
- Unlike Lotka–Volterra, data x is weakly informative about θ, and hence the posterior over θ tends to be broad and non-Gaussian. In our setup, we placed flat independent priors over θ_1, $\theta_2 - \theta_1$ and θ_3, and we took x to be 5 equally spaced percentiles

Related work : Regression adjustment

An early parametric approach to ABC is regression adjustment, where a parametric regressor is trained on simulation data in order to learn a mapping from x to θ. The learnt mapping is then used to correct for using a large , by adjusting the location of posterior samples gathered by e.g. rejection ABC. Beaumont et al. used linear regressors, and later Blum and François used neural networks with one hidden layer that separately predicted the mean and variance of θ. Both can be viewed as rudimentary density estimators and as such they are a predecessor to our work. However, they were not flexible enough to accurately estimate the posterior, and they were only used within some other ABC method to allow for a larger . In this work, make conditional density estimation flexible enough to approximate the posterior accurately

Related work : Regression adjustment

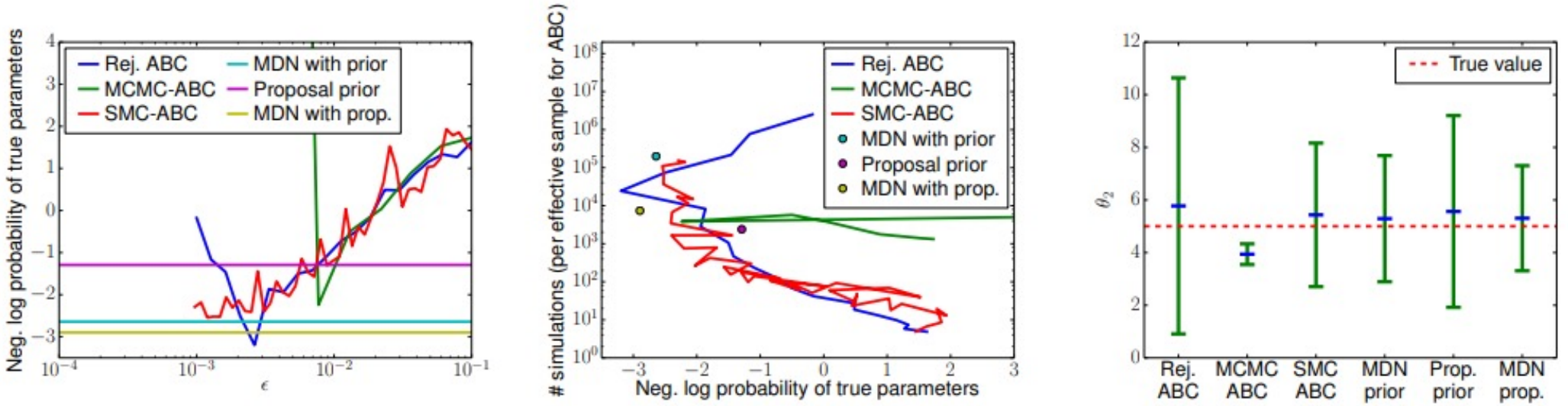


Figure 4: Results on M/G/1. **Left**: negative log probability of true parameters vs ϵ; lower is better. **Middle**: number of simulations vs negative log probability; lower left is better. Note that number of simulations is total for MDNs, and per effective sample for ABC. **Right**: Estimates of θ_2 with 2 standard deviations; ABC estimates correspond to the lowest setting of ϵ used.

Related work : Synthetic likelihood

Another parametric approach is synthetic likelihood, where parametric models are used to estimate the likelihood $p(x \mid \theta)$. Wood used a single Gaussian, and later Fan et al. used a mixture Gaussian model. Both of them learnt a separate density model of x for each θ by repeatedly simulating the model for fixed θ. More recently, Meeds and Welling used a Gaussian process model to interpolate Gaussian likelihood approximations between different θ's. Compared to learning the posterior, synthetic likelihood has the advantage of not depending on the choice of proposal prior. Its main disadvantage is the need of further approximate inference on top of it in order to obtain the posterior. In our work we directly learn the posterior, eliminating the need for further inference, and we address the problem of correcting for the proposal prior

Related work : Efficient Monte Carlo ABC

Recent work on ABC has focused on reducing the simulation cost of sample-based ABC methods. Hamiltonian ABC improves upon MCMC-ABC by using stochastically estimated gradients in order to explore the parameter space more efficiently. Optimization Monte Carlo ABC explicitly optimizes the location of ABC samples, which greatly reduces rejection rate. Bayesian optimization ABC models $p(||X - X_0|| \mid \theta)$ as a Gaussian process and then uses Bayesian optimization to guide simulations towards the region of small distances $||X - X_0||$. In our work we show how a significant reduction in simulation cost can also be achieved with parametric methods, which target the posterior directly.

Related work : Recognition networks

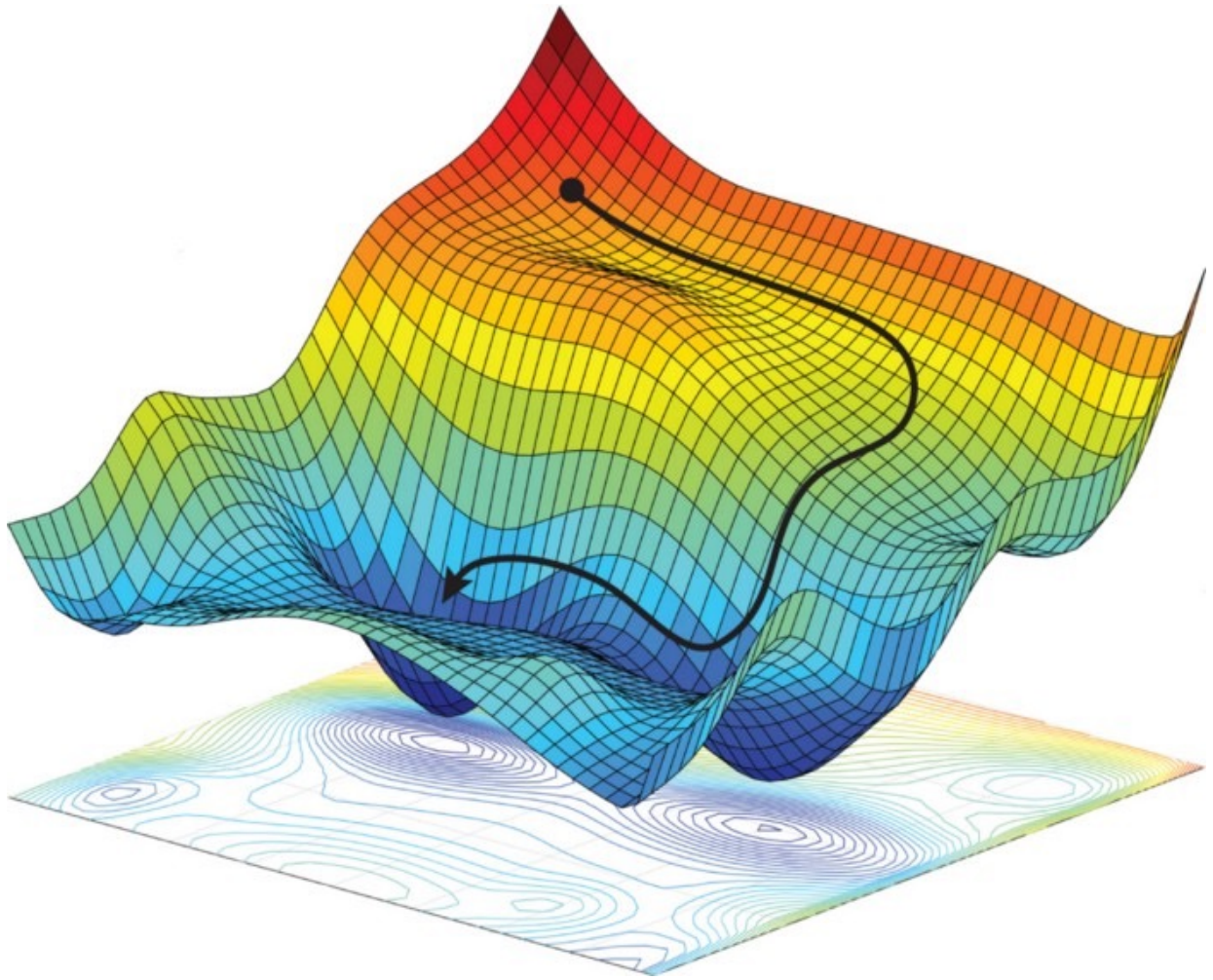
Use of neural density estimators for learning posteriors is reminiscent of recognition networks in machine learning. A recognition network is a neural network that is trained to invert a generative model. The Helmholtz machine, the variational auto-encoder and stochastic backpropagation are examples where a recognition network is trained jointly with the generative network it is designed to invert. Feedforward neural networks have been used to invert black-box generative models and binary-valued Bayesian networks, and convolutional neural networks have been used to invert a physics engine. Our work illustrates the potential of recognition networks in the field of likelihood-free inference, where the generative model is fixed, and inference of its parameters is the goal.

Bayesian Deep Learning

Frequentist perspective

- The frequentist approach to machine learning is to *optimize* a *loss function* to obtain an optimal setting of the model parameters.
- An example loss function is *cross-entropy*, used for *classification* tasks such as object detection or machine translation.
- From a probabilistic perspective, frequentists are trying to maximize the likelihood ***p(D|w,M)***

maximum likelihood estimation (MLE)



Navigating a loss space in the direction of steepest descent using Gradient Descent.

Bayesian Perspective

- Realistically **Quantify Uncertainty**
- Instead of a parameter point estimate, **probability distribution** over parameters
- The posterior represents our **belief / hypothesis / uncertainty**
- **Bayes' Theorem** to compute the posterior

$$posterior = \frac{likelihood * prior}{evidence}, \ or \ p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})}$$

$$evidence = p(\mathcal{D}) = \int p(\mathcal{D}|w)p(w)dw$$

Bayesian Perspective

$$posterior = \frac{likelihood * prior}{evidence}, \ or \ p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})}$$

$$evidence = p(\mathcal{D}) = \int p(\mathcal{D}|w)p(w)dw$$

- Start with specifying a prior distribution p(w) over the parameter to capture our **belief** about, what our model parameter should look like **prior to** observing any data.
- using our dataset, we can update (multiply) our prior **belief** with the likelihood **p(D|w)**
- To obtain a valid posterior probability distribution, however, the product between the likelihood and the prior must be evaluated for each parameter setting and normalized. This means **marginalizing** (summing or integrating) over all parameter settings. The normalizing constant is called the Bayesian (model) **evidence** or **marginal likelihood p(D)**
- **p(D)** provides evidence for how good our model
- We sometimes explicitly include the model choice M in the evidence as **p(D|M)**. This enables us to compare different models with different parameter spaces

Maximum A Posteriori (MAP) Estimation

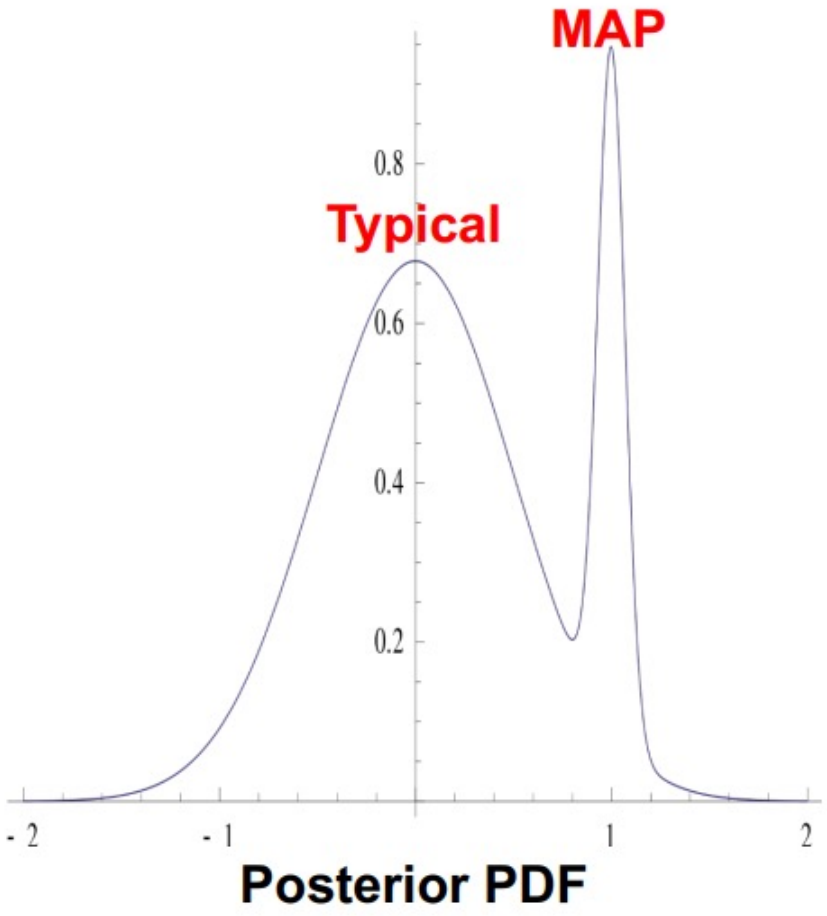
- Very common to produce maximum probability estimates

$$\hat{\theta}^{\mathrm{MAP}} = \arg max\ p(\theta \mid y)$$

- MAP is the mode (highest probability outcome) of the posterior
- MAP (mode) may not be representative of typical outcomes, Also, not a Bayes estimator (unless discrete)

$$\lim_{c \to 0} L(\theta, \hat{\theta}) = \begin{cases} 0, & \text{if } |\hat{\theta} - \theta| < c \\ 1, & \text{otherwise} \end{cases}$$

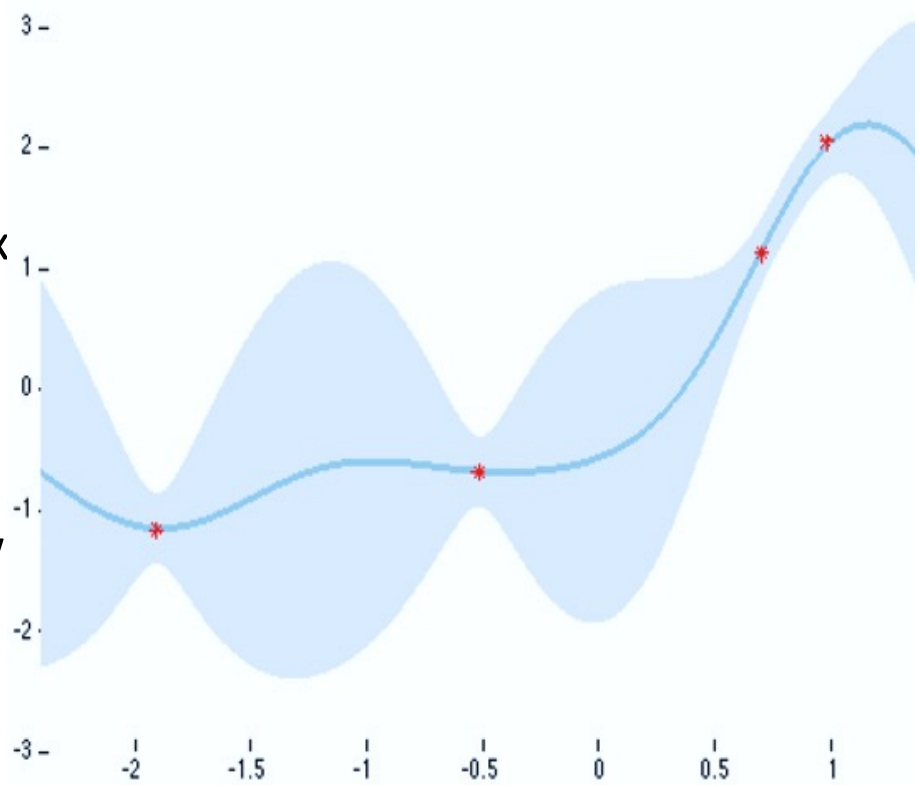
Degenerate loss function



Full Predictive Distribution

$$p(y|\mathcal{D}, x) = \int p(y|w, x)p(w|\mathcal{D})dw$$

- The full-fledged Bayesian approach is to specify a predictive distribution **p(y|D,x)**
- This defines the probability for class label y given new input x and dataset D
- marginalize over our parameter settings
- **Bayesian Model Averaging, or BMA**
- multiply the posterior probability of each setting w, with the probability of label y, given input x using parameter setting w
- **Prior Predictive** distribution - predict of *y* before seeing *x*
- **Posterior Predictive** distribution - we observe *x* we can predict future observations *y*



Approximate Predictive Distribution

- sampling a few parameters settings and combining the resulting models
- **Monte Carlo Approximation** of the predictive distribution
- Monte Carlo Estimation
- Sequential Monte Carlo
- Markov Chain Monte Carlo
- The Metropolis-Hastings algorithm
- The Gibbs sampler

Density Estimation

- In statistics, density estimation is the procedure of estimating an unknown density p(y) from observed data
- The very early stage of density estimation techniques traces back to the usage of histograms, later followed by kernel density estimation in which the shape of the data is approximated through a kernel function with a smoothing parameter (bandwidth)
- Due to the difficulty in specifying the bandwidth in kernel density estimation, mixture models have become a popular alternative approach
- Mixture Densities

$$p(y|\theta) = \sum_{k=1}^{K} \omega_k p_k(y|\theta_k)$$

where $\sum_{k=1}^{K} \omega_k = 1$ for non-negative mixture weights ω_k and $p_k(x|\theta_k)$ are the component densities. When $n < \infty$, the mixture is said to be finite. If $K = \infty$, it is called an infinite mixture

Density Estimation

One important property is that the moments of the mixture density are easily obtained through the moments of its mixture components. If the m:th central moment exists for all of its component densities, the m:th central moment for the finite mixture density exists and is of the form

$$E((y-\mu)^m|\theta) = \sum_{k=1}^{K}\sum_{i=1}^{m}\omega_k\binom{m}{i}E((y-\mu_k)^i|\theta_i)$$

- Conditional Density Estimators
- Multivariate density estimation
- Copula density estimation

Conditional Density Estimation

- Conditional density estimation (CDE) is a general framing of supervised learning problems, subsuming both classification and regression.
- Modeling complex and heteroscedastic noise distributions is useful in a variety of settings for which the full predictive distribution rather than its mean is of inherent interest or informs subsequent decisions.

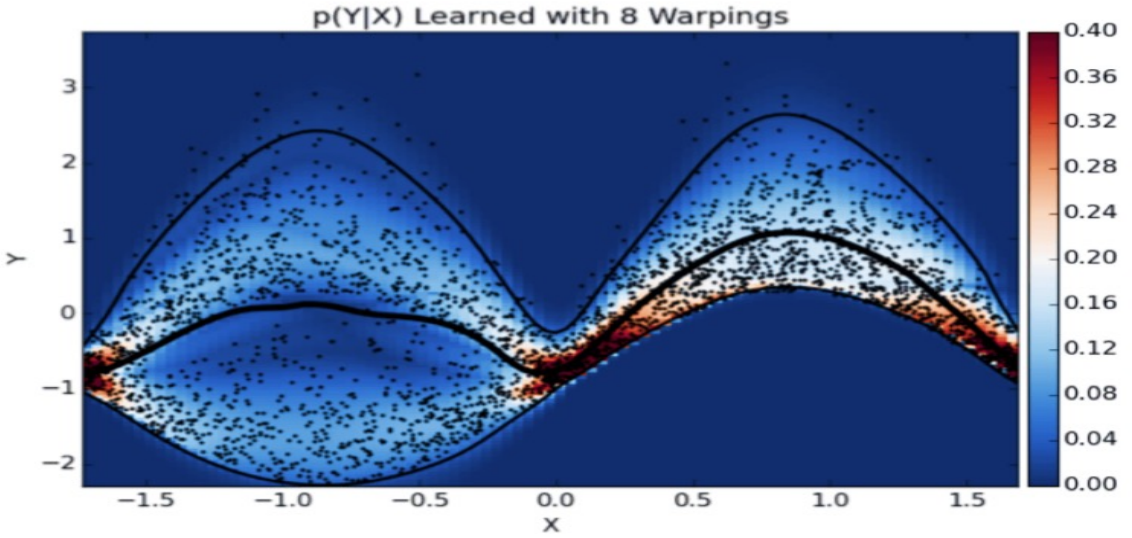


Figure 1. Complex, heteroscedastic conditional densities are learned with normalising flows on a toy dataset using the the proposed method (N= 5000). Color reflects the predicted conditional probability, $p(y|x)$. Black lines represent the median and 95% confidence intervals of the conditionals. Best viewed in color.

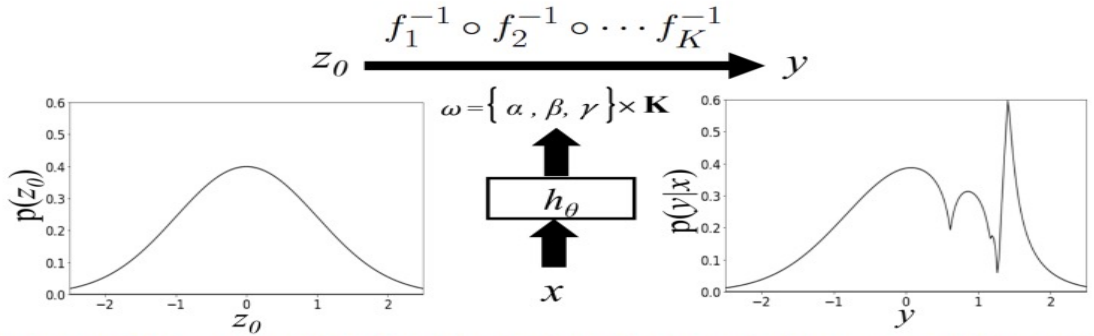
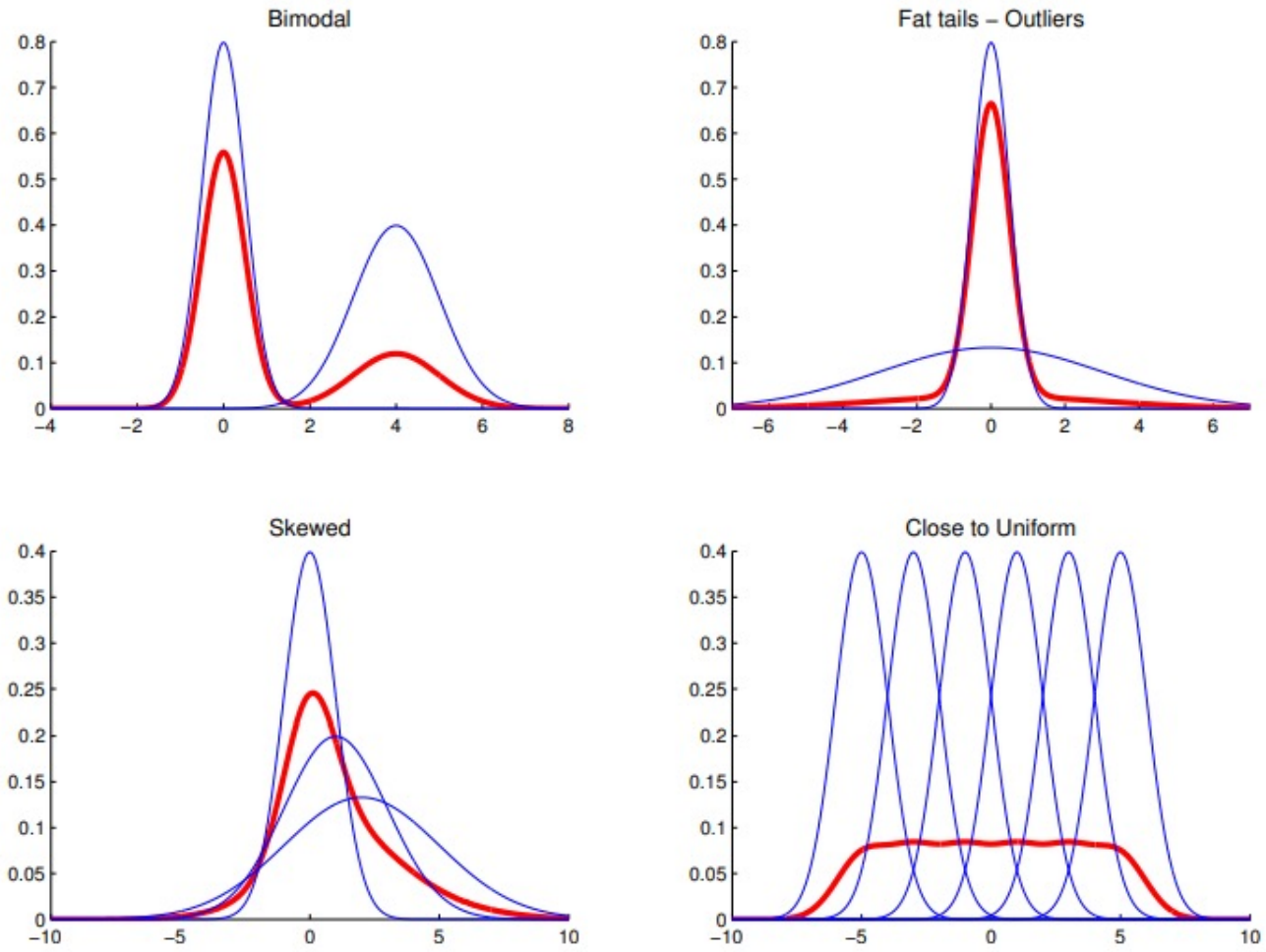


Figure 2. A normally distributed random variable passed through a $K-$stage normalising flow gives a complex derived distribution, $p(y = f^{-1}(z)|x)$. In CDE the parameters of the flow are $\omega = h_\theta(x)$.

Conditional Density Estimation

The conditional density estimation concentrates on modeling the relationship between a response and set of covariates *x* through a conditional density function *p(y|x).*



Using mixture of normal densities (thin lines) to mimic a flexible density (bold line)

Conditional Density Estimation

Gaussian linear regression $y = x'\beta + \varepsilon$ with $\varepsilon \sim N(0, \sigma^2)$ is trivially equivalent to modeling $p(y|x)$ by a Gaussian density with mean function $\mu = x'\beta$ and constant variance σ^2.

Mixtures of conditional densities is the obvious extension of mixture models to the conditional density estimation problem:

$$p(y|x) = \sum_{k=1}^{K} \omega_k p_k(y|x)$$

where $p_i(y|x)$ is the conditional density in i:th mixture component. A simple case is the mixture of homoscedastic Gaussian regression models with constant mixture weights. The limitation of this model is that it restricts the shape of the distribution to be the same for all x. A smooth mixture is a finite mixture density with weights that are smooth functions of the covariates

$$\omega_k(x) = \frac{\exp(x'\gamma_k)}{\sum_{i=1}^{K} \exp(x'\gamma_i)}.$$

Conditional Density Estimation

In conditional density estimation, an important focus is modeling the regression mean $E(y|x)$. A spline is a popular approach for nonlinear regression that models the mean as a linear combination of a set of nonlinear basis functions of the original regressors,

$$y = f(x) + \varepsilon = x'\beta + \sum_{i=1}^{k} x(\xi_i)'\beta_i + \varepsilon$$

where k is number of basis functions $x(\xi)$ used and ξ_i is the location of i:th basis function, often referred to as a knot. Each basis function is defined by a knot ξ_i in covariates space and the knots determine the points of flexibility of the fitted regression function. In the case with multiple covariates $x_1, \ldots, x_q$ it is common to assume additivity

$$y = \sum_{j=1}^{q} f_j(x_j) + \varepsilon,$$

where $f_j(x_j)$ are spline functions. The more general surface model does not assume additivity and uses a multi-dimensional basis function with interactions among the covariates. It is possible to have both additive and interactive splines in the regression.

Conditional Density Estimation - Notation and Terminology

- Conditional density estimation (CDE) refers to the problem of modeling the conditional, p(y|x, D)
- Parametric methods for CDE propose a class of densities, {p, ω ∈ Ω}, and a class of functions, h, indexed by θ ∈ Θ, and use D to choose, hθ : X → Ω, xi → ωi, which is then used to model p(yi|xi) as p(yi|ωi = hθ(xi)).
- The choice of Ω determines the sort of distributional complexity which may be learned from D, and Θ sets the array of input dependent changes which can be learned, ranging from simple global translations of a stationary predictive distribution to complex heteroscedastic behavior

Method	Distribution	Input Dependence	Inference Scheme
Least Squares Regression	Gaussian	$h(x) = \theta^{\mathrm{T}} x$	$\theta = (X^{\mathrm{T}} X)^{-1} X^{\mathrm{T}} Y$
Generalized Linear Models	Exponential Family	$h(x) = \theta^{\mathrm{T}} x$	Iteratively reweighted least squares
Neural Network Classification	Categorical	$h(x) = \mathrm{NN}_\theta(x)$	Stochastic gradient descent
Mixture Density Networks	Mixture of Gaussians	$h(x) = \mathrm{NN}_\theta(x)$	Stochastic gradient descent
Conditional Density Estimation	Normalising Flows	$h(x) = \mathrm{NN}_\theta(x)$	Stochastic variational inference

Table 1. Bayesian normalising flows in relation to some common methods for conditional density methods

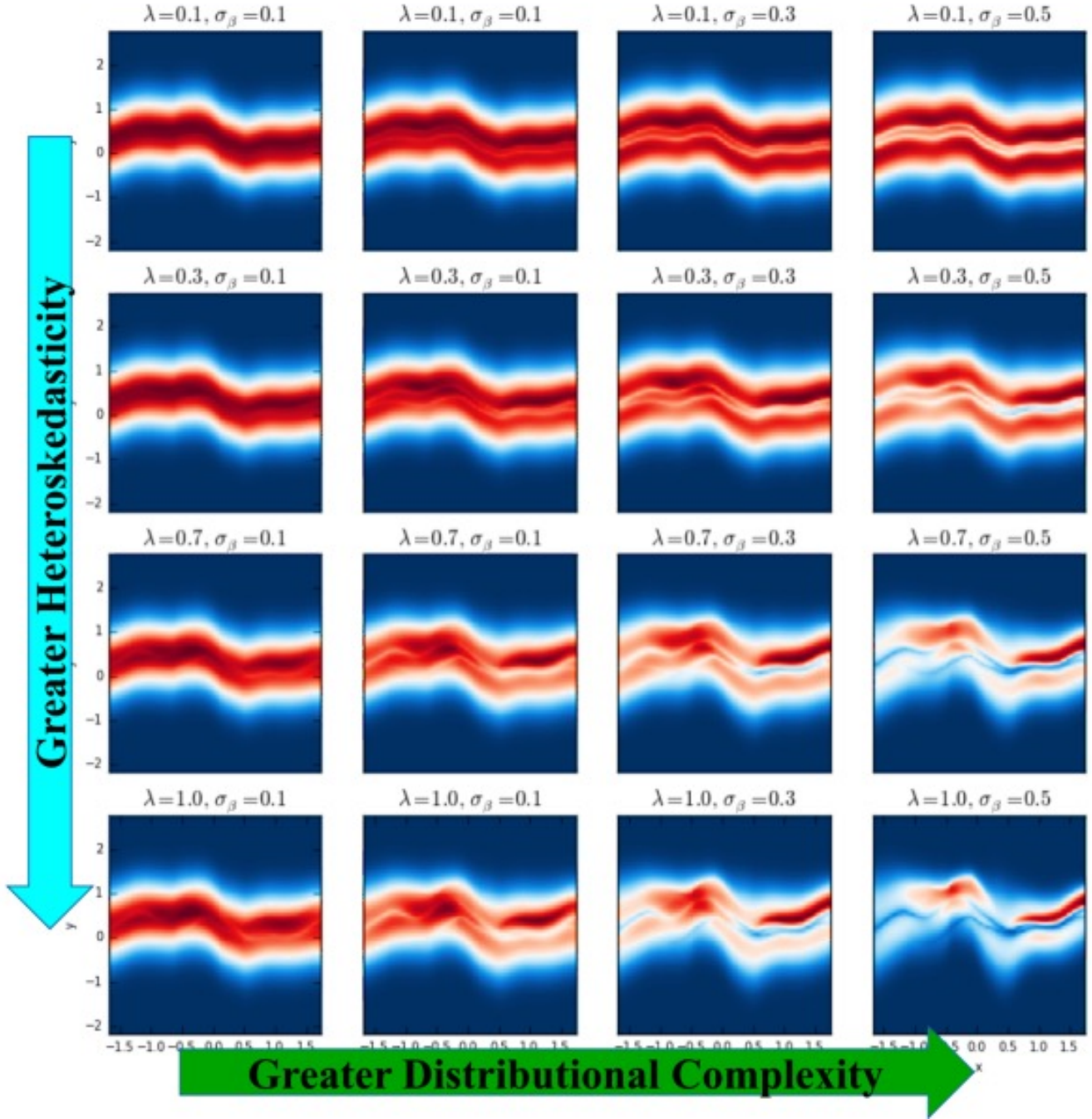


Figure 4. The characteristics of conditional density estimators sampled from different priors vary with the prior parameters. Each panel is a heatmap depicting the conditional density estimator defined by an MLP mapping to the parameters of a 5-stage normalising flow. The same random seed for each sample, and the reparameterization trick is used to interpolate between different Gaussian priors. Moving left to right, we increase the prior standard deviation over the parameter $\hat{\beta}$, which controls the magnitude of the warping. Moving top to bottom, we increase the value of the parameter λ, which controls the extent of heteroscedasticity (with larger values reflecting greater heteroscedasticity). Best viewed in color.

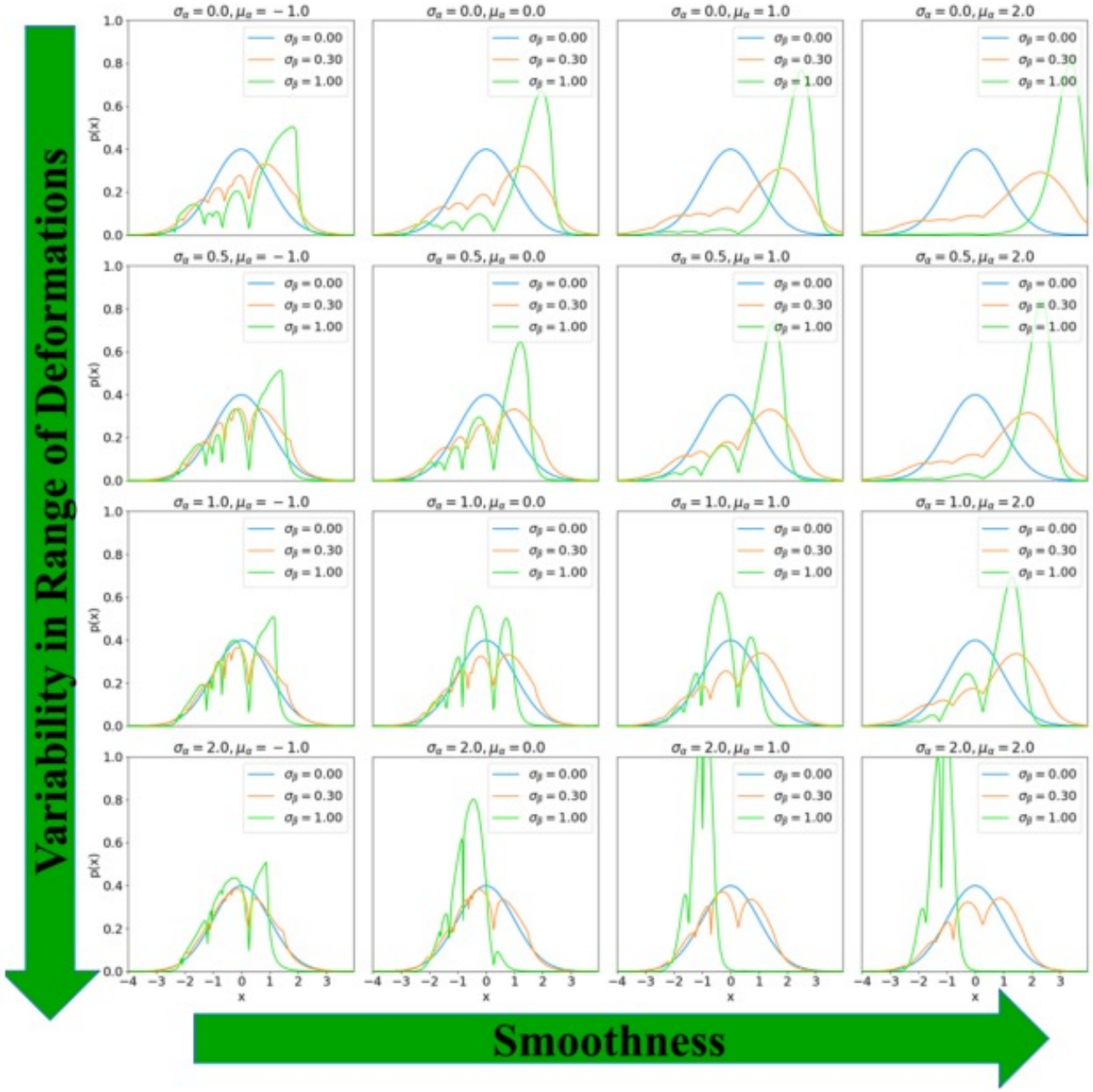
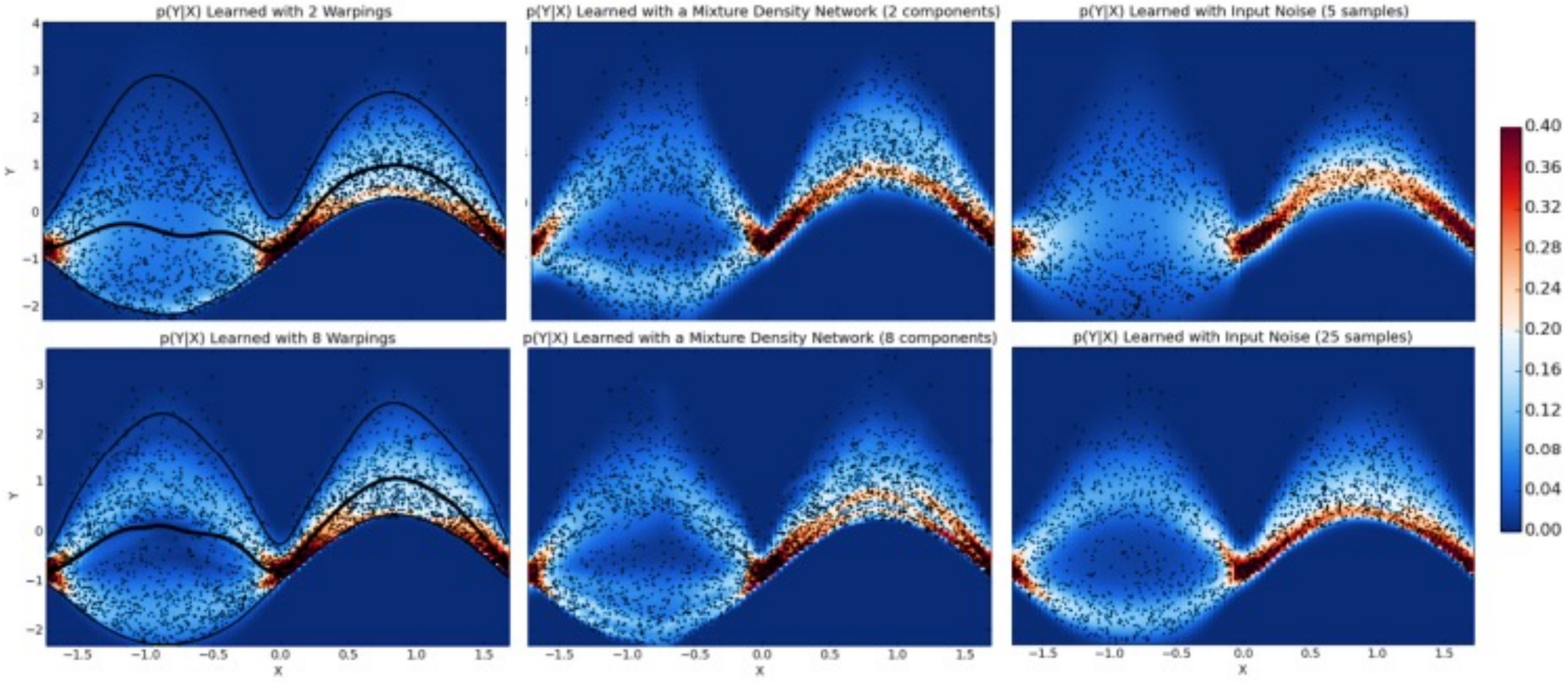


Figure 8. A manifold representing probability densities sampled from different priors over a 10-stage normalizing flow. The same random seed is used to interpolate between different choices of priors to demonstrate the impact of different choices of priors on the resultant distribution. The variance in maximum magnitudes of the distortions are controlled by $\sigma_{\hat{\beta}}$, which varies across the densities within each subplot. The sharpness of the distortions is controlled by $\mu_{\hat{\alpha}}$, which varies from sharpest to smoothest across the columns. The variance of the sharpness of the distortions is controlled by $\sigma_{\hat{\alpha}}$, and is increased in successive rows. The remaining parameters are fixed at $\sigma_{\hat{\beta}} = 1.0$, $\mu_z = 0.0$ and $\mu_{\hat{\beta}} = 0.0$. Best viewed in color.



Heatmaps representing toy densities learned with three conditional density estimation methods. LEFT) Normalizing flows, MIDDLE) Mixture density networks and RIGHT) input noise. The top row demonstrates the performance of simple models and the bottom show performance for higher capacity models. Heat represents the conditional probability, p(y|x). Normalizing flows additionally allow us easily find confidence intervals of this conditional distribution; we plot the 95% confidence interval in black. The training set consisted of 5000 points. Best viewed in color

Qs

https://openai.com/research/deep-double-descent

- mixture of homoscedastic Gaussian regression models

References

1- https://jorisbaan.nl/2021/03/02/introduction-to-bayesian-deep-learning.html#Basics
2. https://arxiv.org/pdf/1802.04908.pdf
3. https://www.diva-portal.org/smash/get/diva2:618973/FULLTEXT01.pdf
4. https://pytorch.org/blog/stochastic-weight-averaging-in-pytorch/
5. https://arxiv.org/pdf/1902.02476.pdf
6. https://arxiv.org/pdf/1902.02476.pdf
7. https://arxiv.org/pdf/1803.05407.pdf
8. https://deep-and-shallow.com/2021/03/20/mixture-density-networks-probabilistic-regression-for-uncertainty-estimation/