Haining

Bayesian Sampling

Importance Sampling

- Sampling from a surrogate distribution q to produce an unbiased estimation for p (that is hard to sample from)

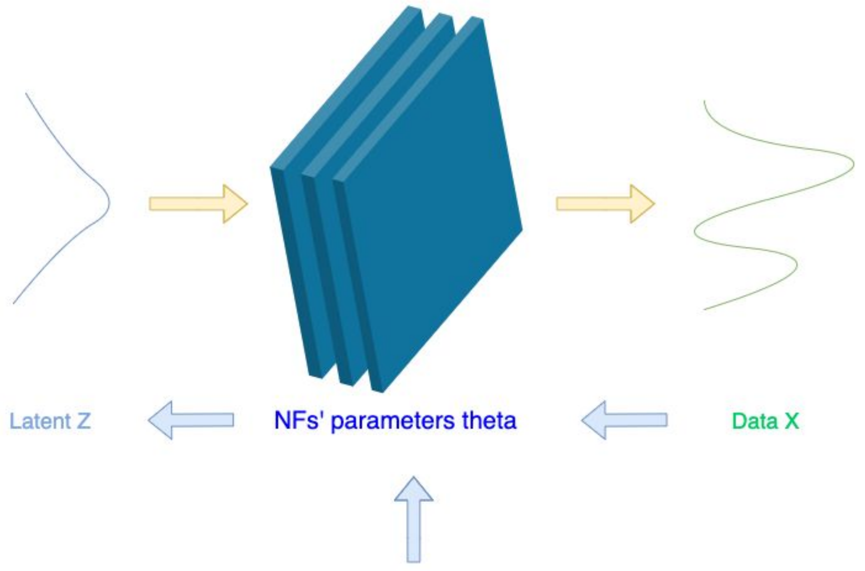
$$E_p[f(x)] = E_q\left[f(x)\frac{p(x)}{q(x)}\right]$$

- Bayesian IS: use prior as the surrogate distribution

  The weight p/q is proportional to the likelihood $\mathrm{IW}(\theta) = \frac{p(\theta|x)}{p(\theta)} \propto f(x|\theta)$

- In practice, need to normalize over sampled theta's, which leads to biased result but not bad
- IS for variance reduction

Our model

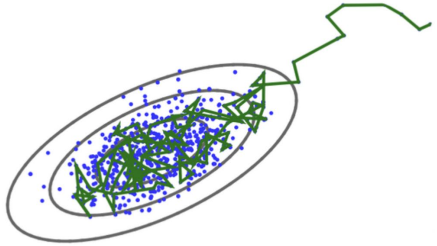


IS from approximated SWAG
transform the data back and get the likelihood
compute IW by normalizing f(x|theta)

MCMC

- MCMC is an important sampling method that can be used to approximate the posterior distributions in Bayesian analysis.
- The idea is to start from some random distribution and move towards the desired distribution (i.e. posterior)
- MCMC generates a dependent sequence, where each sample has a distribution conditioned on the previous value
- As opposed to sampling from posterior like MCMC, VI approximates the posterior using a feasible set of distributions Q, where the target is to minimize the KL / maximize the ELBO.
- Can do MC estimation for large enough time T

$$\mathbb{E}_{x\sim p}[f(x)] \approx \frac{1}{T}\sum_{t=1}^{T} f(x^{(t)})$$



https://carlessanchezalonso.github.io/2020/mcmc/

Metropolis-Hasting

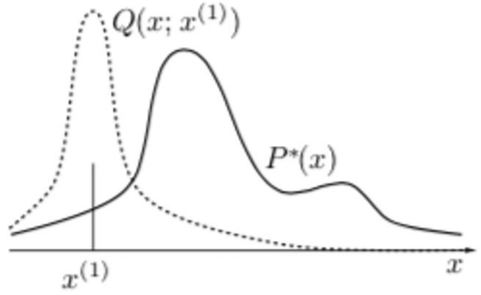
- Metropolis-Hasting incorporates the idea of MCMC by using a proposal distribution q that depends on the current state x_t, which defines a conditional transition distribution.
- Accept a new sample with probability max(a, 1), where

$$a = \frac{\tilde{p}(x')q(x^{(t)}|x')}{\tilde{p}(x^{(t)})q(x'|x^{(t)})}$$

- This is derived from the detailed balance sheet condition

$$p(X_1).\,T(X_1 \rightarrow X_2) = p(X_2).\,T(X_2 \rightarrow X_1)$$

- Can be slow when going to high-dimensional sample space



(Stochastic Gradient) Langevin Dynamics

- An optimization technique based on 1. Stochastic gradient descent 2. Langevin Dynamics
- Langevin dynamics models a molecular system based on Langevin equation, which describes physical property of Brownian motion
- SGD -> SGLD: $$\Delta\theta_t = \frac{\epsilon}{2}\left(\nabla \log p(\theta_t) + \sum_{i=1}^{N} \nabla \log p(x_i|\theta_t)\right) + \eta_t$$

  This is a stochastic differential equation and its solution/equilibrium is the posterior

- Take this as the proposal distribution and do Metropolis-Hasting!
- Interestingly, when step sizes decrease to zero with specific convergence properties, the MH rejection rates go to zero.
- Can be trained in mini-batched and quantify model uncertainty

$$\Delta\theta_t = \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla \log p(x_{ti}|\theta_t)\right) + \eta_t \qquad \eta_t \sim N(0, \epsilon_t)$$

Max Welling, Yee Whye Teh. 2011

Other sampling methods

- Gibbs sampling

  Gibbs sampling is based on Metropolis Hastings where its proposals are always accepted.

  Iteratively update xi ~ p(xi | x-i) in (x1, …, xn) until convergence

- Hamiltonian MC

  $$E(\mathbf{x}, \mathbf{v}) = U(\mathbf{x}) + K(\mathbf{v}), \qquad K(\mathbf{v}) = \sum_i \frac{m v_i^2}{2}$$

  HMC employs a physical trick describing the potential and kinetic energy of a system.

  MH in high-dimensional sampling becomes inefficient, while HMC is more efficient

Variational inference

Variational methods

- Approximate the posterior through a tractable parametric distribution q_phi(theta)
- Objective: minimizing the KL-divergence between q and posterior
- Equivalent to maximizing the ELBO

$$\min_q KL[\, q(\theta) \,||\, p(\theta|y) \,]$$

$$E_{q(\theta)}\left[\log \frac{p(y,\theta)}{q(\theta)}\right]$$

- The evidence log p(y) can be further triangulated into three terms:

$$\log p(y) \;=\; \int \log p(y|\theta) q(\theta) d\theta - KL[\, q(\theta) \,||\, p(\theta) \,] + KL[\, q(\theta) \,||\, p(\theta|y) \,]$$

- For tractability, we need a nice set of q. For better approximation, we can proceed by using mean-field approach, which assumes

$$q(x) = \prod_{i \in V} q_i(x_i)$$

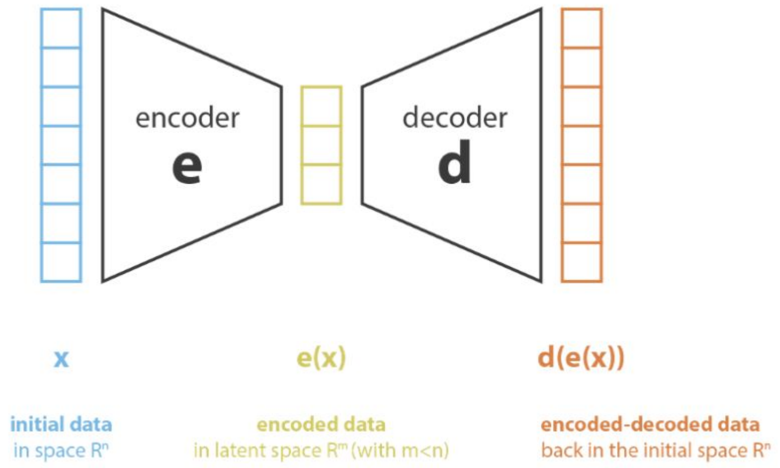
Variational Autoencoder

- Encodes inputs into distributions in a latent space
- Apply VI to approximate the posterior of the latent
- Run forward and compute the loss = - ELBO

$$L(x;\theta,\phi) = -E_{z_\phi \sim q_\phi}\Big[\log p_\theta(x|z)\Big] + KL(q_\phi(z|x)||p(z))$$

- Backpropagate enc, dec for optimization
- Since all the encoder decoder parameters are point estimates, VAE is not Bayes in our context.

Kingma/Welling 2014



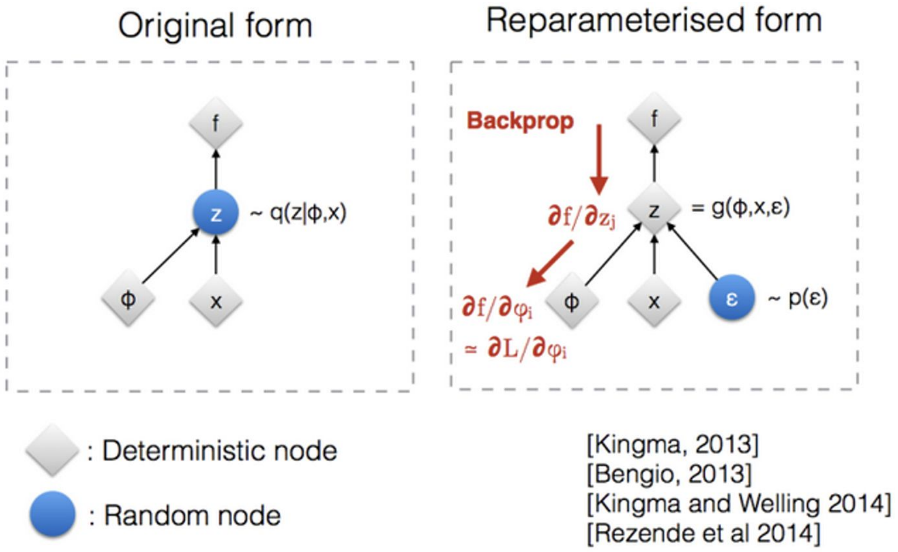
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Bayes by Backprop

The reparameterization trick:

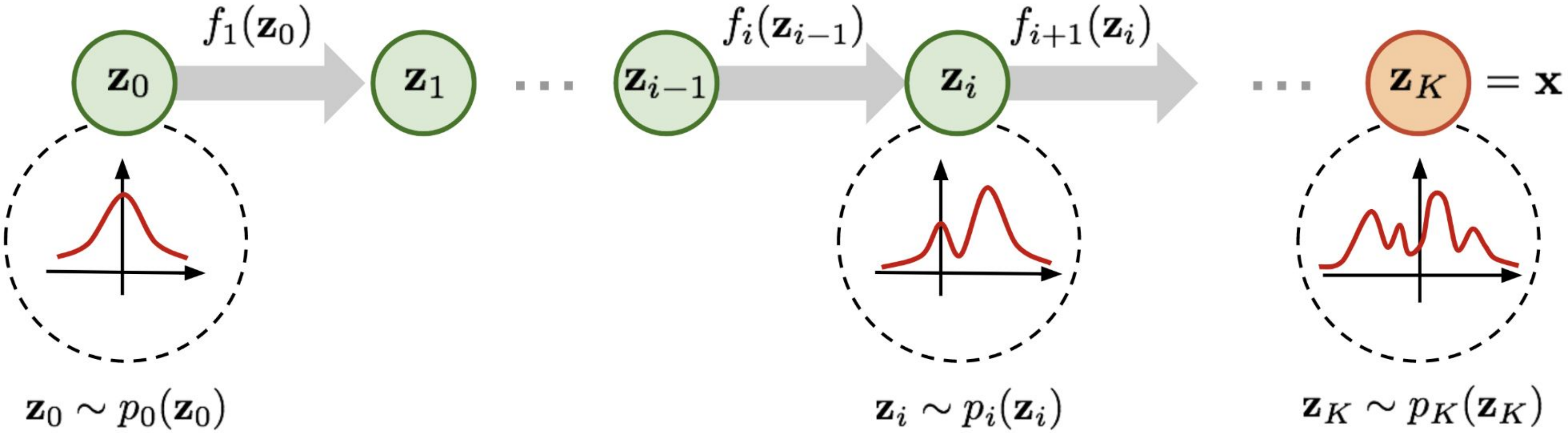
Makes the density independent from the set of parameters phi

$$\begin{aligned}\nabla_\phi \mathcal{L}(\phi) &= \nabla_\phi \mathbb{E}_{z\sim q_\phi(z)}\Big[\log p(x,z) - \log q_\phi(z)\Big]\\ &= \nabla_\phi \mathbb{E}_{\epsilon\sim p(\epsilon)}\Big[\log p(x,T(\phi,\epsilon)) - \log q_\phi(T(\phi,\epsilon))\Big]\\ &= \mathbb{E}_{\epsilon\sim p(\epsilon)}\nabla_\phi\Big[\log p(x,T(\phi,\epsilon)) - \log q_\phi(T(\phi,\epsilon))\Big]\end{aligned}$$

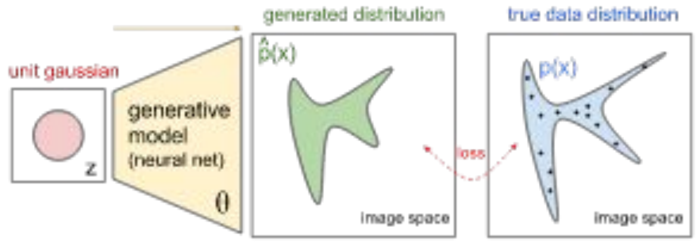


[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

Normalizing Flows


$f_1(\mathbf{z}_0)$
$f_i(\mathbf{z}_{i-1})$
$f_{i+1}(\mathbf{z}_i)$
$\mathbf{z}_0$
$\mathbf{z}_1$
$\mathbf{z}_{i-1}$
$\mathbf{z}_i$
$\mathbf{z}_K = \mathbf{x}$
$\mathbf{z}_0 \sim p_0(\mathbf{z}_0)$
$\mathbf{z}_i \sim p_i(\mathbf{z}_i)$
$\mathbf{z}_K \sim p_K(\mathbf{z}_K)$

Gallery of Generative Models

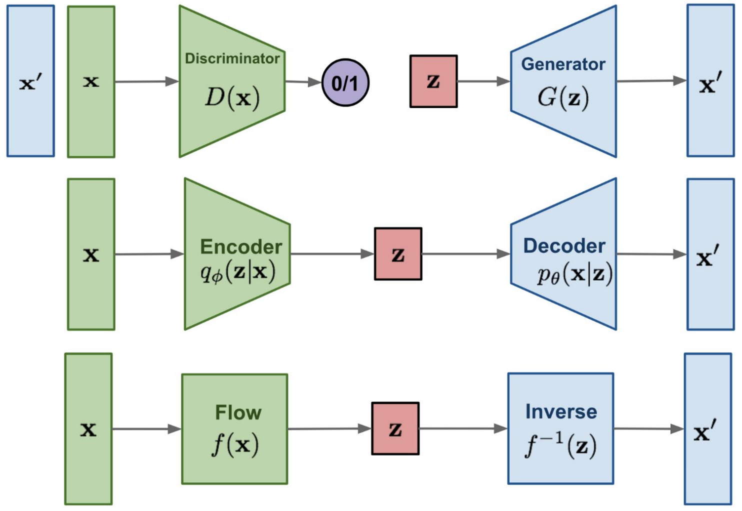


- Gaussian Mixture Models (GMM)
- Hidden Markov Models (HMM)
- Latent Dirichlet Allocation Models (LDA)
- Boltzmann Machine (BM)
- Variational Autoencoder (VAE)
- Generative Adversarial Networks (GAN)
- Flow-based Models e.g. **Normalizing Flows**
- Energy-based Models

GAN: minimax the classification error loss.

VAE: maximize ELBO.

Flow-based generative models: minimize the negative log-likelihood



$$\mathbf{z}_K = f_K \circ \ldots \circ f_2 \circ f_1(\mathbf{z}_0)$$

$$\ln q_K(\mathbf{z}_K) = \ln q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|$$

Normalizing Flows: Definition

A normalizing flow describes the transformation of a probability density through a sequence of invertible mappings.

By repeatedly applying the rule for change of variables, the initial density ‘flows’ through the sequence of invertible mappings.

At the end of this sequence we obtain a valid probability distribution and hence this type of flow is referred to as a *normalizing flow*.

Concept Decomposition

- ***Normalizing***

  The change of variables gives a normalized density after applying an invertible transformation.

- ***Flow***

  The path traversed by the random variables *z_k = f_k(z_k−1)* with initial distribution *q_0(z_0)*. In other words, the invertible transformations can be composed with each other to create more complex invertible transformations.

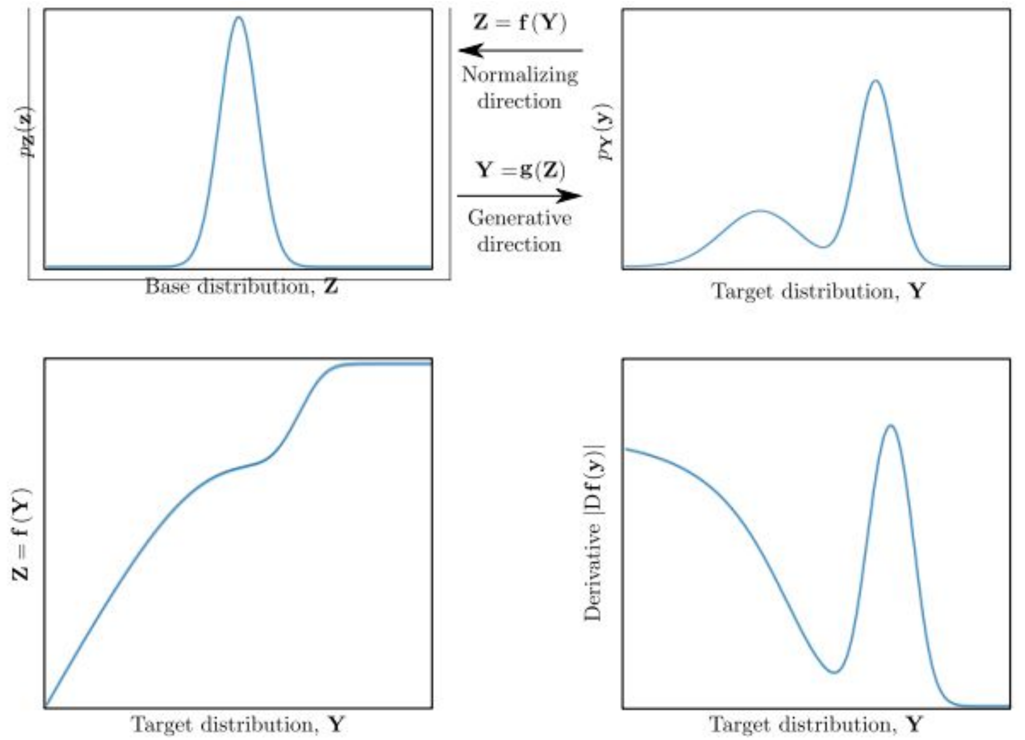
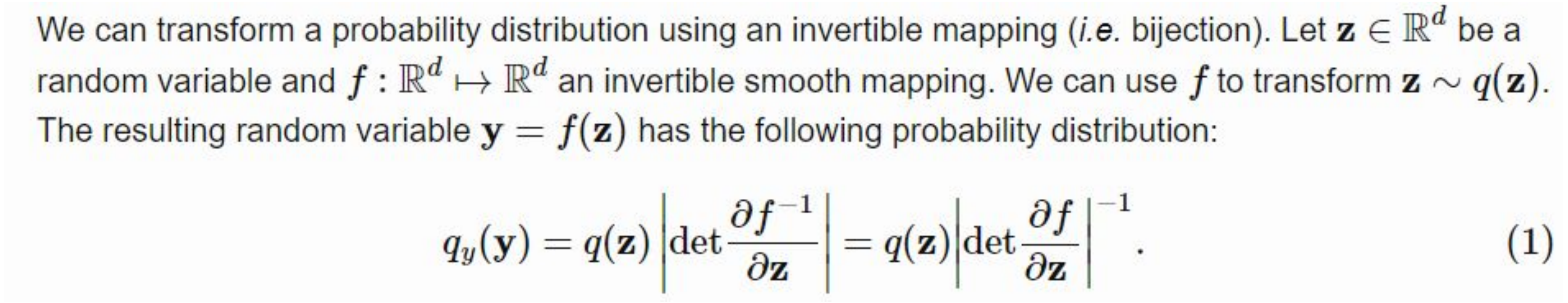


Fig. 1. Change of variables (Equation (1)). Top-left: the density of the source $p_{\mathbf{Z}}$. Top-right: the density function of the target distribution $p_{\mathbf{Y}}(\mathbf{y})$. There exists a bijective function $\mathbf{g}$, such that $p_{\mathbf{Y}} = \mathbf{g}_* p_{\mathbf{Z}}$, with inverse $\mathbf{f}$. Bottom-left: the inverse function $\mathbf{f}$. Bottom-right: the absolute Jacobian (derivative) of $\mathbf{f}$.

Change of Variables Theorem



We can transform a probability distribution using an invertible mapping (*i.e.* bijection). Let $\mathbf{z} \in \mathbb{R}^d$ be a random variable and $f : \mathbb{R}^d \mapsto \mathbb{R}^d$ an invertible smooth mapping. We can use f to transform $\mathbf{z} \sim q(\mathbf{z})$. The resulting random variable $\mathbf{y} = f(\mathbf{z})$ has the following probability distribution:

$$q_y(\mathbf{y}) = q(\mathbf{z}) \left| \det \frac{\partial f^{-1}}{\partial \mathbf{z}} \right| = q(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1}. \tag{1}$$

http://akosiorek.github.io/ml/2018/04/03/norm_flows.html

Specific Architectural Structures of Deep NFs

- The input and output dimensions must be the same.
- The transformations must be invertible.
- The transformations should be computationally efficient, both in terms of the calculation of the determinant of the Jacobian and of the transformation **f** + its inverse

$$\mathbf{z}_K = f_K \circ \ldots \circ f_2 \circ f_1(\mathbf{z}_0)$$

$$\ln q_K(\mathbf{z}_K) = \ln q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|$$

Ryan Wang

Applications of normalizing flows

Where do tractable likelihoods and sampling come in handy?

General references:
Kobyzev, I 2020: Normalizing Flows: An Introduction and Review of Current Methods
Papamakarios, G. 2021: Normalizing Flows for Probabilistic Modeling and Inference
Schwartz, S 2022: NormalizingFlows.ipynb

Density estimation:

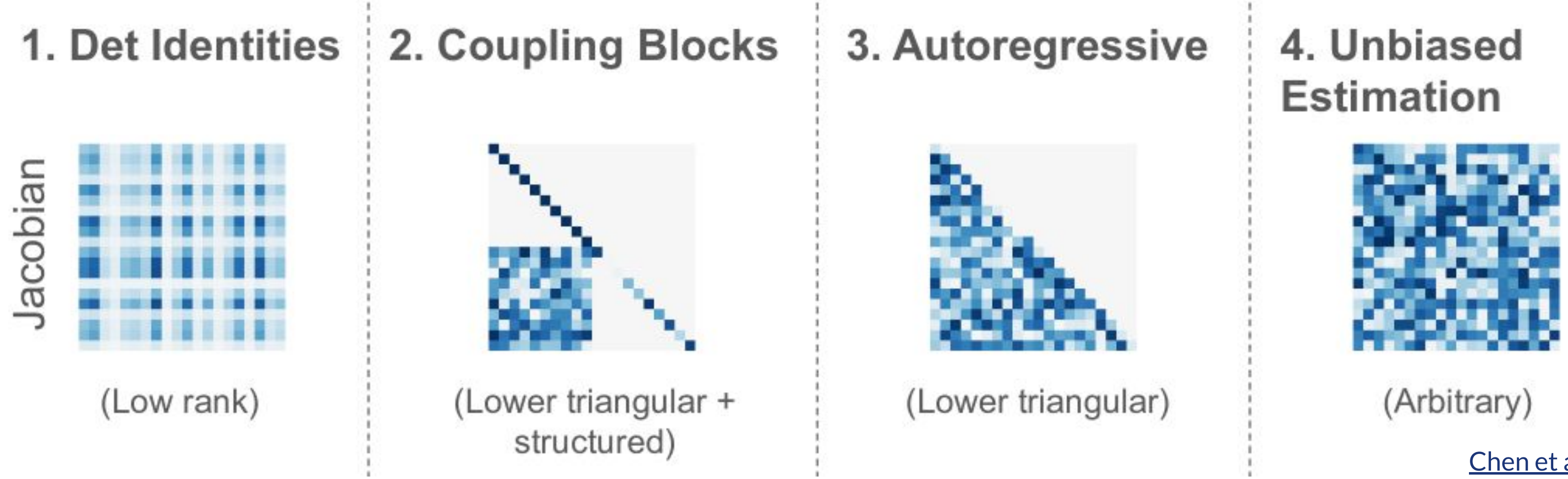
- Model likelihood where **f** is the normalizing function and **Df** is the Jacobian of the transformation

$$\begin{aligned}\log p(\mathcal{D}|\Theta) &= \sum_{i=1}^{M} \log p_{\mathbf{Y}}(\mathbf{y}^{(i)}|\Theta) \\ &= \sum_{i=1}^{M} \log p_{\mathbf{Z}}(\mathbf{f}(\mathbf{y}^{(i)}|\theta)|\phi) + \log|\det \mathbf{Df}(\mathbf{y}_i|\theta)|\end{aligned}$$

Probabilistic Modelling and Inference

- Variational inference using the flow as an approximate posterior/target (reverse KL)
  - Reparameterization using base distribution
  - MCMC
- Sampling and evaluation needed for IS
  - e.g. NFs as proposals (Müller et al., 2019)

Popular normalizing flows and their variations



Chen et al., 2019

A detour into autoregressive model specification

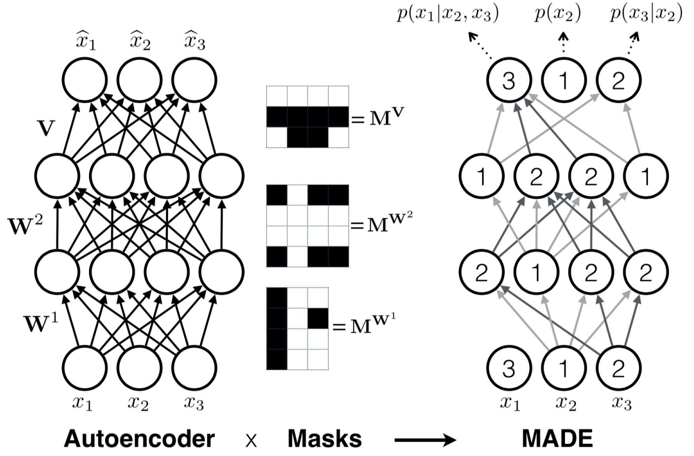
The Chain Rule of Probability:

$$p_{\mathrm{x}}(\mathbf{x}) = \prod_{i=1}^{D} p_{\mathrm{x}}(\mathrm{x}_i \mid \mathbf{x}_{<i}).$$

MADE (Masked Autoencoder for Distribution Estimation)

- Autoregressive neural network such that:

$$\begin{aligned} p(x_1, \cdots, x_k) &= \prod_{j=1}^{k} p_{X_j|X_{j-1},\cdots,X_1}(x_j|x_{j-1},\cdots,x_1) \\ &= \prod_{j=1}^{k} p_{\theta_j(x_1,\cdots,x_{j-1})}(x_j) \end{aligned}$$



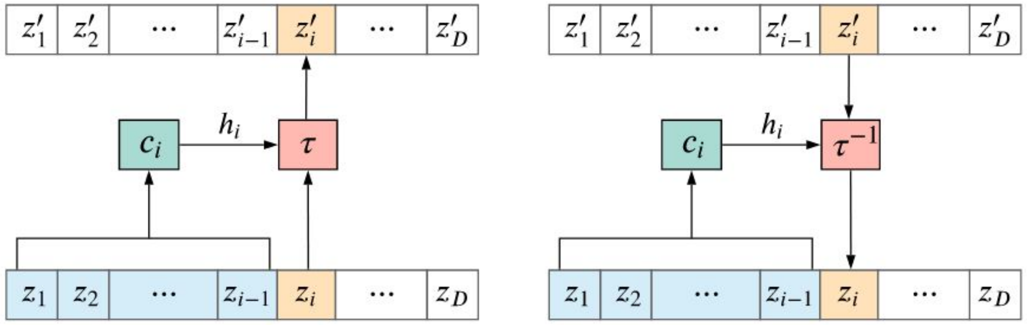
Germain et al., 2015

Autoregressive Flows

Model Specification:

$$z'_i = \tau(z_i; \boldsymbol{h}_i) \quad \text{where} \quad \boldsymbol{h}_i = c_i(\mathbf{z}_{<i})$$

$$z_i = \tau^{-1}(z'_i; \boldsymbol{h}_i) \quad \text{where} \quad \boldsymbol{h}_i = c_i(\mathbf{z}_{<i})$$



(a) Forward (b) Inverse

- $\tau(\cdot; \mathbf{h}_i)$ the **transformer**, representing a generative or normalizing transformation
  - Strictly monotonic bijection such that we can invert them, satisfying the change of variables
- $\mathbf{h}_i = c_i(\mathbf{z}_{<i})$ is the **i-th conditioner**, parameterizing the transformer
  - e are no restrictions on how the conditioner is defined as long as it follows the autoregressive dependency structure
  - Autoregressive dependency allows triangular Jacobian

Mixing-and-matching

Transformers

Affine: $\tau(\mathrm{z}_i; \boldsymbol{h}_i) = \alpha_i \mathrm{z}_i + \beta_i \quad \text{where} \quad \boldsymbol{h}_i = \{\alpha_i, \beta_i\}$

$$\log\left|\det J_{f_\phi}(\mathbf{z})\right| = \sum_{i=1}^{D} \log|\alpha_i| = \sum_{i=1}^{D} \tilde{\alpha}_i$$

Combinations: $\tau(\mathrm{z}) = \sum_{k=1}^{K} w_k \tau_k(\mathrm{z})$, where $w_k > 0$ for all k

- Not analytically invertible; can correspond to 1-layer MLP with Jacobian through back-prop
- Weights must be positive with monotone activations

Others include splines, piecewise-bijective, sum-of-squares polynomial, etc.

Conditioners

RNNs: $\boldsymbol{h}_i = c(\boldsymbol{s}_i) \quad \text{where} \quad \begin{aligned} \boldsymbol{s}_1 &= \text{initial state} \\ \boldsymbol{s}_i &= \mathrm{RNN}(\mathrm{z}_{i-1}, \boldsymbol{s}_{i-1}) \text{ for } i > 1 \end{aligned}$

- Requires sequential computation

Masking:

- Avoids sequential computation; all $\boldsymbol{h}_i$ in one pass as long as autoreg. dep. holds (e.g. **MADE**)

Coupling:
$$(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_d) = \text{constants, either fixed or estimated}$$
$$(\boldsymbol{h}_{d+1}, \ldots, \boldsymbol{h}_D) = F(\mathbf{z}_{\leq d}).$$

- Partitions inputs and parameters of transformer are conditioned on one partition

MAF vs. IAF

Masked Autoregressive Flow: Density Estimation

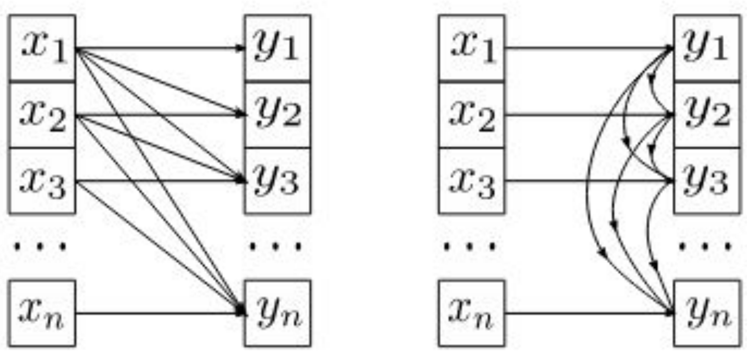
- Using the affine transformer with scale and location defined by a neural network like MADE for one-pass likelihood evaluation

$$X_j^{(t)} = g^{(t)}(X_{0:j-1}^{\boxed{(t)}}) = \mu_j^{(t)}(X_{0:j-1}^{\boxed{(t)}}) + \sigma_j^{(t)}(X_{0:j-1}^{\boxed{(t)}}) \cdot X_j^{(t-1)}$$

Inverse Autoregressive Flow: Sampling

- Theoretically the same as MAF, but focuses on the reverse transform

$$X_j^{(t)} = g^{(t)}(X_{0:j-1}^{\boxed{(t-1)}}) = \mu_j^{(t)}(X_{0:j-1}^{\boxed{(t-1)}}) + \sigma_j^{(t)}(X_{0:j-1}^{\boxed{(t-1)}}) \cdot X_j^{(t-1)}$$



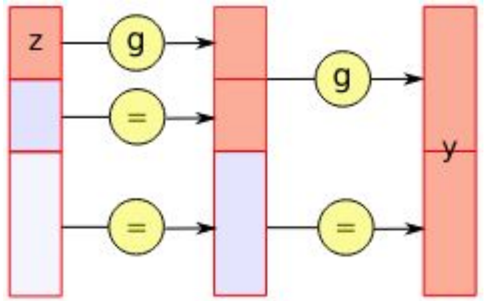
Notice the importance of flow direction, MAF forward transform requires autoreg. dep. from current transform, while IAF doesn't!

Ryan Wang

Improving autoregressive efficiency

- Recall the coupling conditioner previously specified relies on less inputs, improving the computation (e.g. RealNVP)

- This can be extended to change the size of the partition over the flow to get intrinsic dimension with computation advantages (Multi-scale)



- Batch normalization between layers; requires Jacobian estimate

$$T_k \circ \mathrm{BN} \circ T_{k-1}$$

Other flows (linear, residual, infinitesimal)

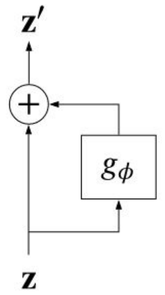
Linear Flows

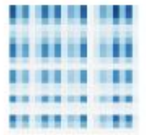
$$\mathbf{z}' = \mathbf{W}\mathbf{z}$$

- Weight matrix is invertible and Jacobian is just the weight matrix!
- Weight matrix can be structured
- Permutations are useful to employ in autoregressive flows
- Invertible 1x1 convolutions are a generalization of permutations (e.g. Glow combines these with coupling layers from RealNVP)

Residual Flows

$$\mathbf{z}' = \mathbf{z} + g_\phi(\mathbf{z})$$




(a) Det. Identities (Low Rank)

(d) **Unbiased Est.** (Free-form)

Chen et al., 2019.

- Invertibility depends on constraints like **contractive mapping theorem** (spectral normalization, element-wise nonlinearities) and the **matrix determinant lemma** (planar flows, Sylvester flows, radial flows, etc.)
- Contractive residual flows require estimates of log absolute Jacobian determinants but sampling/density est. are iterative and expensive
- Matrix determinant lemma flows are generally used for VI because no analytical inverse

Continuous Infinitesimal Flows

Instead of using a specific number of transformations, we can represent infinite transformations as an ODE that we can learn

$$\frac{d\mathbf{z}_t}{dt} = g_\phi(t, \mathbf{z}_t)$$

Integrate forward in time:

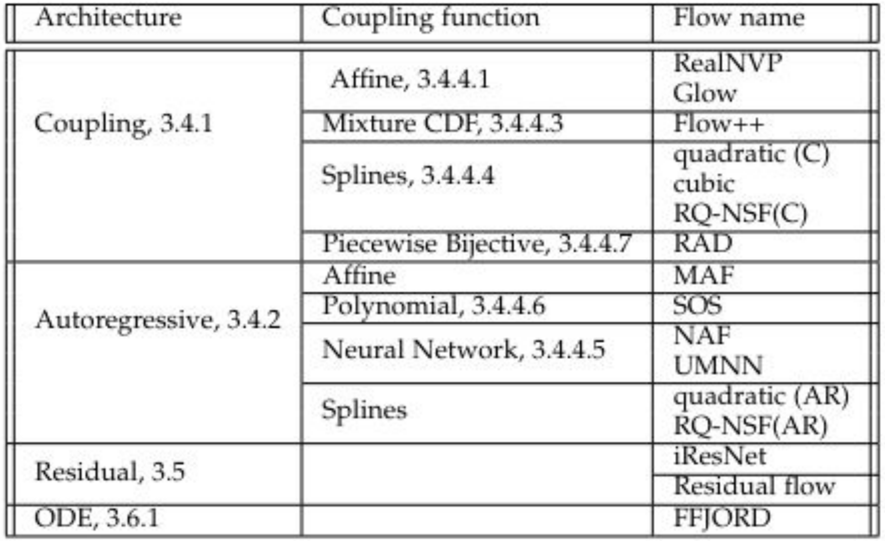
$$\mathbf{x} = \mathbf{z}_{t_1} = \mathbf{u} + \int_{t=t_0}^{t_1} g_\phi(t, \mathbf{z}_t)\, dt$$

Integrate backward in time:

$$\mathbf{u} = \mathbf{z}_{t_0} = \mathbf{x} - \int_{t=t_0}^{t_1} g_\phi(t, \mathbf{z}_t)\, dt$$

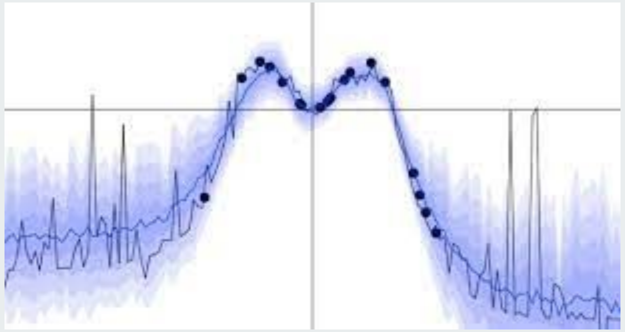
- Like a continuous version of the residual flow
- Forward and inverse have same computational cost
- Estimates require ODE solvers (e.g. Runge-Kutta methods or Adjoint Sensitivity methods)
- Langevin flows were SDE-based mixing of base distribution and complicated distribution however doesn't have many practical applications

And many more



Architecture	Coupling function	Flow name
Coupling, 3.4.1	Affine, 3.4.4.1	RealNVP
		Glow
	Mixture CDF, 3.4.4.3	Flow++
	Splines, 3.4.4.4	quadratic (C)
		cubic
		RQ-NSF(C)
	Piecewise Bijective, 3.4.4.7	RAD
Autoregressive, 3.4.2	Affine	MAF
	Polynomial, 3.4.4.6	SOS
	Neural Network, 3.4.4.5	NAF
		UMNN
	Splines	quadratic (AR)
		RQ-NSF(AR)
Residual, 3.5		iResNet
		Residual flow
ODE, 3.6.1		FFJORD

Overview of Bayesian Deep Learning (BDL)



"While deep learning has been revolutionary for machine learning, most modern deep learning models cannot represent their uncertainty nor take advantage of the well-studied tools of probability theory. This has started to change following recent developments of tools and techniques combining Bayesian approaches with deep learning. The intersection of the two fields has received great interest from the community, with the introduction of new deep learning models that take advantage of Bayesian techniques, and Bayesian models that incorporate deep learning elements."

—— 2019 NeurIPS BDL Workshop

"Instead of training a single network, the proposed method trains an ensemble of networks, where each network has its weights drawn from a shared, learnt probability distribution. Unlike other ensemble methods, our method typically only doubles the number of parameters yet trains an infinite ensemble using unbiased Monte Carlo estimates of the gradients."

—— Blundell, et. al 'Bayes by Backprop'

$$V(y|x) = E[V(y|x,\theta)] + V(E[y|x,\theta])$$

Uncertainty Quantification in Neural Networks (NNs)

Aleatoric Uncertainty:

- Captures noise inherent in the observations
- Modeled by placing a distribution over the output of NNs
- Cannot be decreased with more data
- Encoded in the likelihood *P(D|H)*

Epistemic Uncertainty:

- Accounts for model uncertainty (structure + parameters)
- Modeled by placing a prior distribution over the weights of NNs
- Can be decreased with more data
- Encoded in the posterior *P(H|D)*

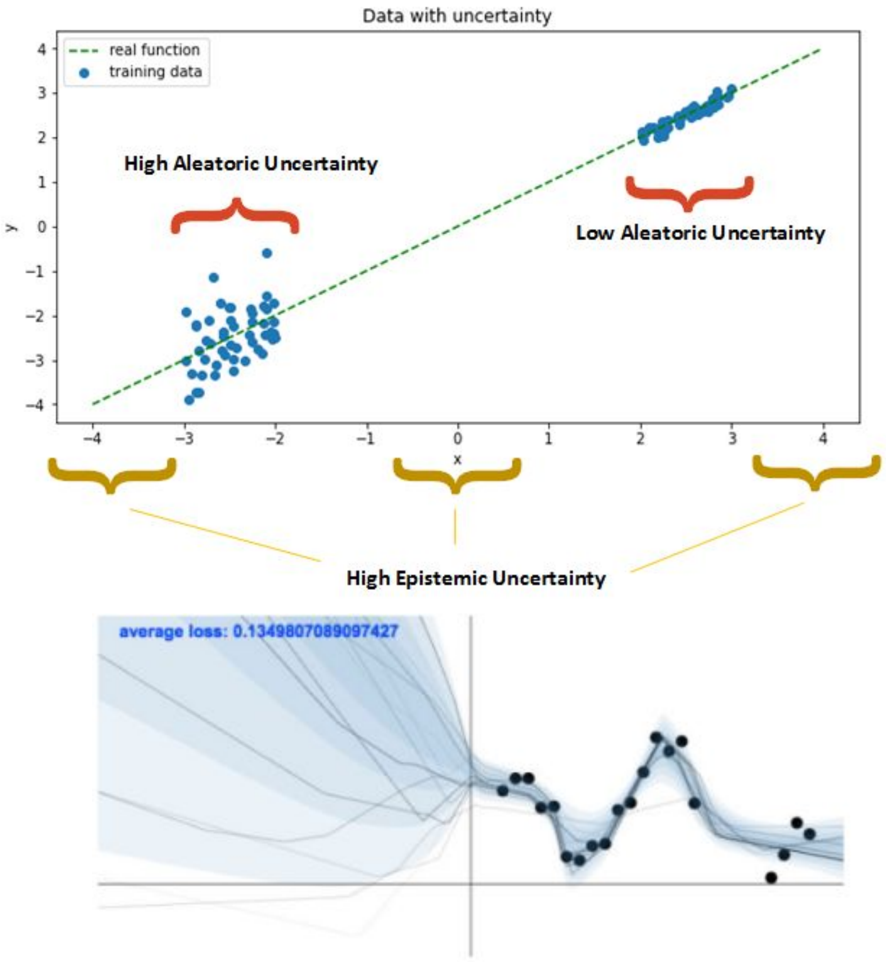
H: hypothesis about which one holds some prior belief *P(H)* i.e. model weights in our case
D: data that will update one's belief about H

Kendall, A., & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision?. *Advances in neural information processing systems*, *30*.

Example:

An exhibit of the different kinds of uncertainty in a linear regression context

- Noisy measurements of the underlying process lead to high ***aleatoric uncertainty***
- High ***epistemic uncertainty*** arises in regions where there are few or no observations for training.

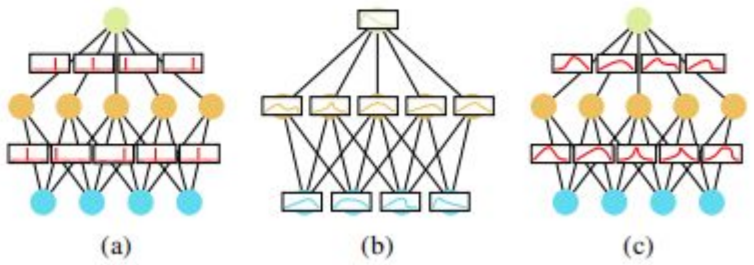


Kendall, A., & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision?. *Advances in neural information processing systems*, *30*.

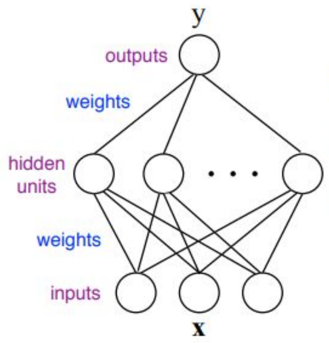
Stochastic Neural Networks = NNs + stochastic components into the network

(a) Point estimate neural network
(b) Stochastic neural network with a probability distribution for the activations
(c) Stochastic neural network with a probability distribution over the weights.

Bayesian Neural Networks = Stochastic Neural Networks + Bayesian training approach



Jospin, Laurent Valentin, et al. "Hands-on Bayesian neural networks—A tutorial for deep learning users." *IEEE Computational Intelligence Magazine* 17.2 (2022): 29-48.



Bayesian neural network

Data: $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N} = (X, \mathbf{y})$

Parameters $\boldsymbol{\theta}$ are weights of neural net

prior $p(\boldsymbol{\theta}|\boldsymbol{\alpha})$

posterior $p(\boldsymbol{\theta}|\boldsymbol{\alpha}, \mathcal{D}) \propto p(\mathbf{y}|X, \boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\alpha})$

prediction $p(y'|\mathcal{D}, \mathbf{x}', \boldsymbol{\alpha}) = \int p(y'|\mathbf{x}', \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}, \boldsymbol{\alpha})\, d\boldsymbol{\theta}$

http://bayesiandeeplearning.org/2016/slides/nips16bayesdeep.pdf

$$p(y \mid x, \mathcal{D}) = \int p(y \mid x, \theta) p(\theta \mid \mathcal{D}) d\theta$$

Deep Ensembles & Bayesian Model Averaging (BMA)

Motivation: aggregating the predictions of a large set of average-performing but independent predictors can lead to better predictions than a single well-performing expert predictor

Deep Ensembles

Find different MAP or maximum likelihood solutions, corresponding to different basins of attraction in the loss sphere, starting from different random initialization.

Some ensembling methods instead work by enriching the hypothesis space, and thus do not collapse when the posterior concentrates.

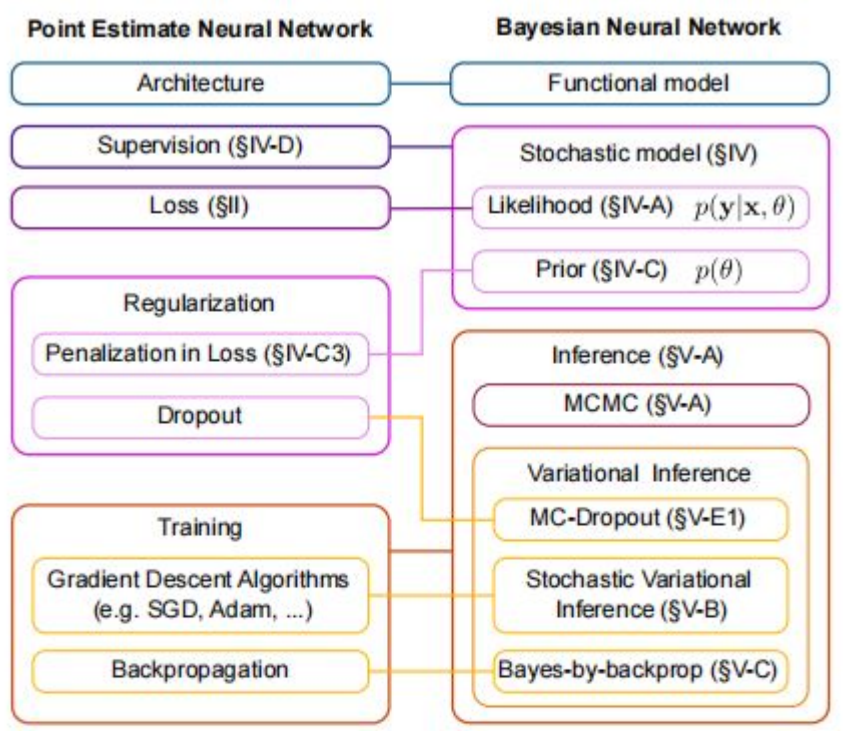
Bayesian Model Averaging (BMA)

Assume that only one hypothesis (one setting of the weights) is correct, and averages over models due to an inability to distinguish between hypotheses given limited data.

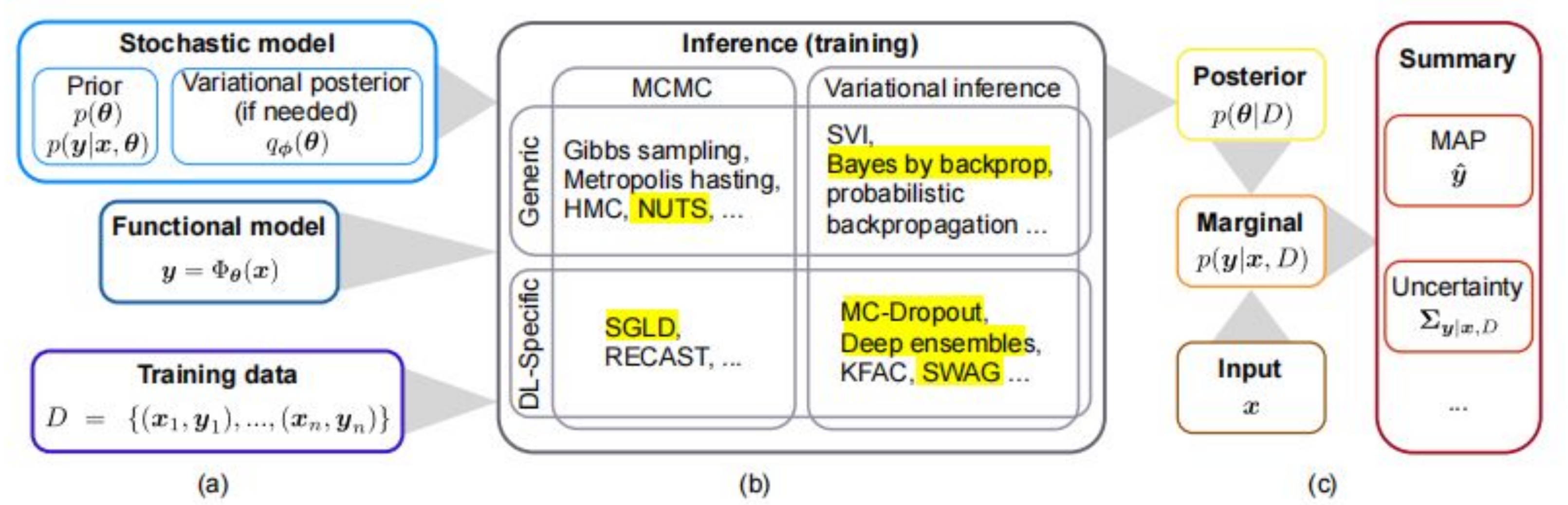
As we observe more data, the posterior collapses,and the Bayesian model average converges to the maximum likelihood solution.

Correspondence between the concepts in deep learning for

- Left: point-estimate neural networks
- Right: Bayesian neural networks

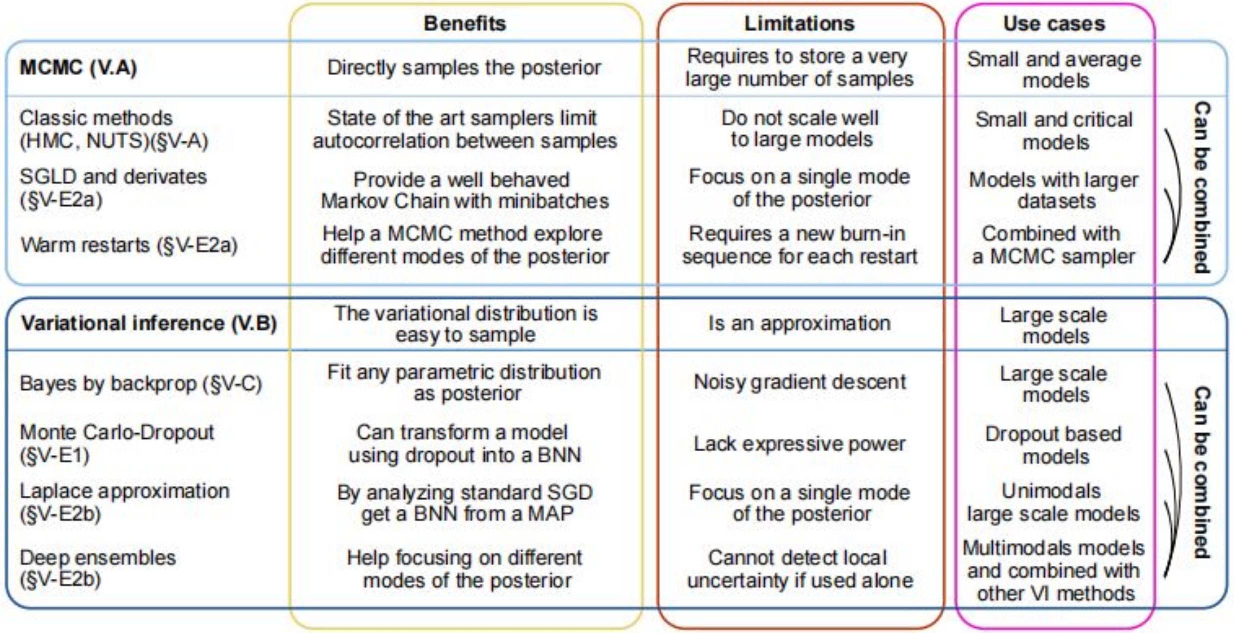


Workflow to (a) design, (b) train and (c) use a BNN for predictions


Stochastic model
Prior $p(\boldsymbol{\theta})$ $p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})$
Variational posterior (if needed) $q_\phi(\boldsymbol{\theta})$
Functional model $\boldsymbol{y} = \Phi_\theta(\boldsymbol{x})$
Training data $D = \{(\boldsymbol{x}_1,\boldsymbol{y}_1),...,(\boldsymbol{x}_n,\boldsymbol{y}_n)\}$
(a)
Inference (training)
MCMC
Variational inference
Generic
DL-Specific
Gibbs sampling, Metropolis hasting, HMC, NUTS, ...
SVI, Bayes by backprop, probabilistic backpropagation ...
SGLD, RECAST, ...
MC-Dropout, Deep ensembles, KFAC, SWAG ...
(b)
Posterior $p(\boldsymbol{\theta}|D)$
Marginal $p(\boldsymbol{y}|\boldsymbol{x},D)$
Input $\boldsymbol{x}$
Summary
MAP $\hat{\boldsymbol{y}}$
Uncertainty $\boldsymbol{\Sigma}_{\boldsymbol{y}|\boldsymbol{x},D}$
...
(c)

To design a BNN, need to choose a functional model (DNN architecture) and a stochastic model (prior and likelihood)

Summary of different inference approaches used to train a BNN with their benefits, limitations and use cases



	Benefits	Limitations	Use cases	
MCMC (V.A)	Directly samples the posterior	Requires to store a very large number of samples	Small and average models	
Classic methods (HMC, NUTS)(§V-A)	State of the art samplers limit autocorrelation between samples	Do not scale well to large models	Small and critical models	**Can be combined**
SGLD and derivates (§V-E2a)	Provide a well behaved Markov Chain with minibatches	Focus on a single mode of the posterior	Models with larger datasets	
Warm restarts (§V-E2a)	Help a MCMC method explore different modes of the posterior	Requires a new burn-in sequence for each restart	Combined with a MCMC sampler	
Variational inference (V.B)	The variational distribution is easy to sample	Is an approximation	Large scale models	
Bayes by backprop (§V-C)	Fit any parametric distribution as posterior	Noisy gradient descent	Large scale models	**Can be combined**
Monte Carlo-Dropout (§V-E1)	Can transform a model using dropout into a BNN	Lack expressive power	Dropout based models	
Laplace approximation (§V-E2b)	By analyzing standard SGD get a BNN from a MAP	Focus on a single mode of the posterior	Unimodals large scale models	
Deep ensembles (§V-E2b)	Help focusing on different modes of the posterior	Cannot detect local uncertainty if used alone	Multimodals models and combined with other VI methods	

Jospin, Laurent Valentin, et al. "Hands-on Bayesian neural networks—A tutorial for deep learning users." *IEEE Computational Intelligence Magazine* 17.2 (2022): 29-48.

Critiques on BNNs... or just misunderstanding?

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

- Heavier computational cost and model complexity = HARD to implement?
  - Not a critique but a challenge to work on
- [Un]informative (parameter) prior p(w): how to view and choose the prior?
  - Prior as a regularizer $\hat{w} = \arg\max_w \log p(w|\mathcal{D}) = \arg\max_w (\log p(\mathcal{D}|w) + \log p(w) + \text{constant})$
  - Prior as proposal distribution in IS: uninformative = BAD to sample from?
  - Is uninformative prior good or bad?
- What did we gain from regularization being Bayesian?

https://jacobbuckman.com/2020-01-22-bayesian-neural-networks-need-not-concentrate/