Uncertainty Quantification and Model Selection in Neural Networks Using Adaptive Annealed SMC

Samir Arora

June 25th, 2025

Agenda

- Neural Network Models
- Problems
- A Bayesian Solution
- Adaptive Annealed SMC Algorithm
- Experiments

Neural Network Models

- **Goal**: Represent an arbitrary function:

$$\mathbf{y} = \Phi(\mathbf{x}).$$

- The simplest neural network architecture is **feedforward networks (FFN)**.
- The FFN is divided into a sequence of n layers: $\{\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3, \cdots, \mathbf{l}_n\}$.
- Each layer performs two steps:
  1. **Linear transformation**:

$$\mathbf{a}_i = \mathbf{W}_i \mathbf{l}_{i-1} + \mathbf{b}_i,$$

     where $\mathbf{W}_i$ and $\mathbf{b}_i$ are weight matrix and bias vector of layer i.
  2. **Non-linear operation**:

$$\mathbf{l}_i = s_i(\mathbf{a}_i),$$

     where s_i is some non-linear operation of layer i.

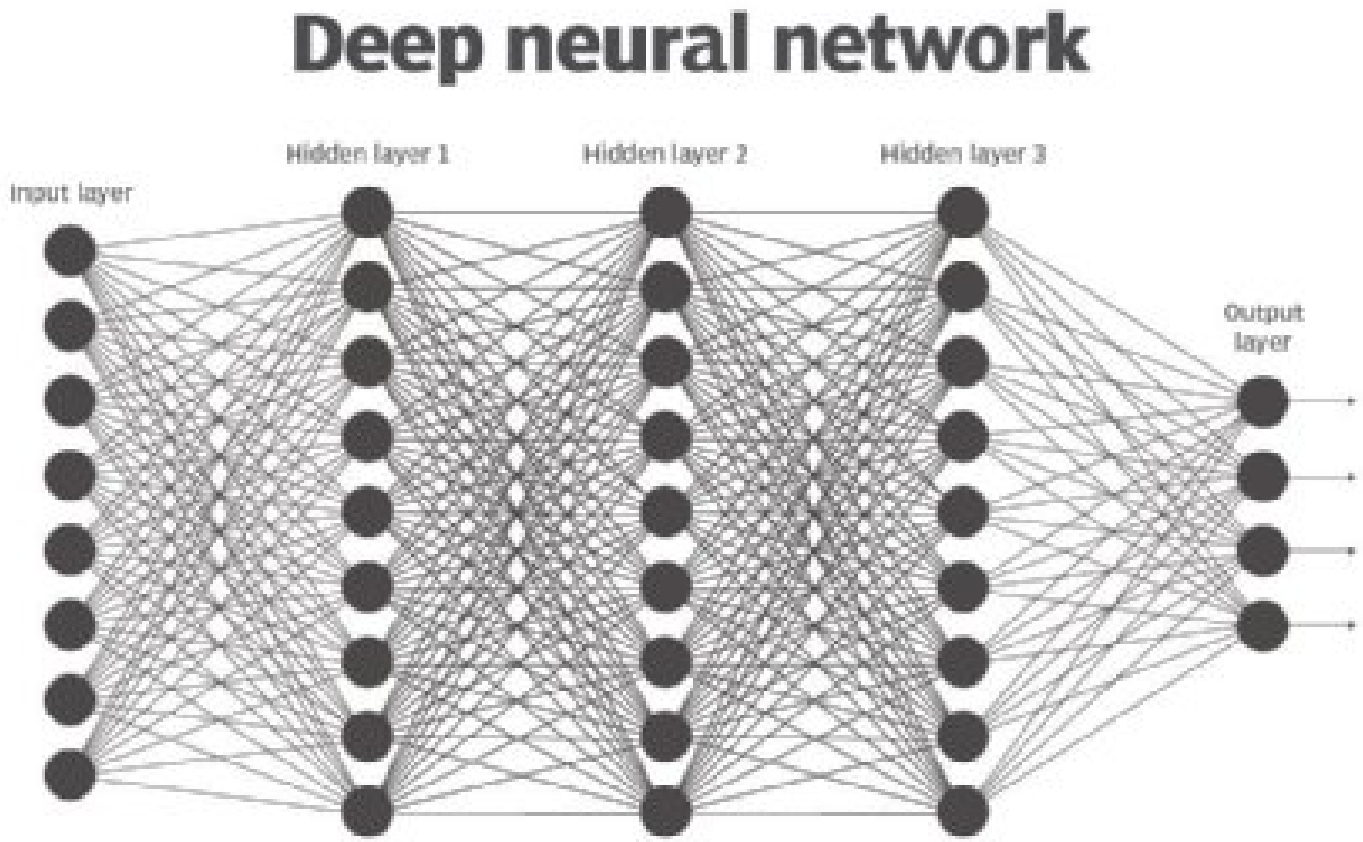
Neural Network Models

- Neural networks can be expressed as the following sequence of operations:

$$\begin{aligned}
\mathbf{l}_0 &= \mathbf{x}, \\
\mathbf{l}_i &= s_i(\mathbf{W}_i \mathbf{l}_{i-1} + \mathbf{b}_i), \quad \forall i \in [1, n], \\
\mathbf{y} &= \mathbf{l}_n.
\end{aligned}$$

- Thus, $\theta = (\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \mathbf{W}_3, \mathbf{b}_3, \cdots, \mathbf{W}_n, \mathbf{b}_n)$ is a vector of parameters in this model.

Neural Network Models



Neural Network Models

- The standard approach to fit a neural network:
  1. Define a loss function $\mathcal{L}(\boldsymbol{\theta}|\mathcal{D})$ - often its a function of the log-likelihood function defined over the training set, where $\mathcal{D}$ is the training data.
  2. **Get a point estimate $\hat{\boldsymbol{\theta}}$** by finding the value that minimize the loss function.

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}|\mathcal{D})$$

- This is done using **forward and backward propagation algorithm**.

Problems

1. What is the **uncertainty** associated with predictions?
2. Too many hyperparameters to decide:
   - number of hidden layers
   - number of hidden units in layer i.
   - type of non-linear operation in layer i.
   - and many more depending on the type of the neural network.

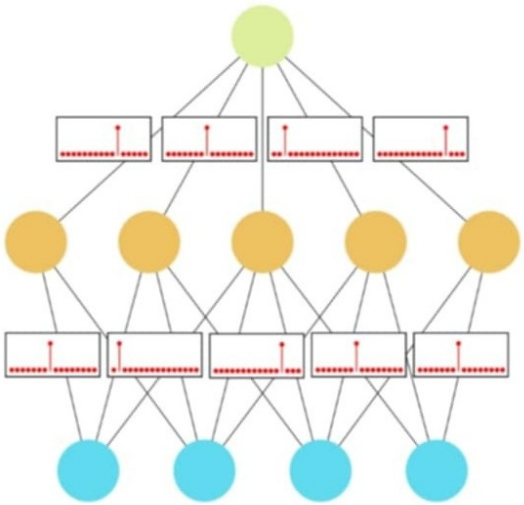
A Bayesian Solution

For the first problem:

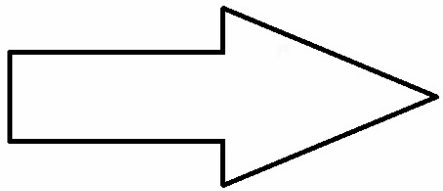
- Compute the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ over the model parameters $\boldsymbol{\theta} = (\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \mathbf{W}_3, \mathbf{b}_3, \cdots, \mathbf{W}_n, \mathbf{b}_n)$.

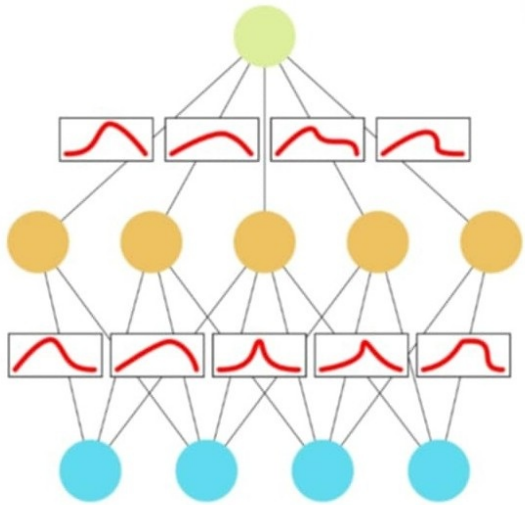
$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta}')p(\boldsymbol{\theta}')d\boldsymbol{\theta}'},$$

where $\mathcal{D}$ is the training set, $\mathcal{D}_x$ denotes the training inputs, and $\mathcal{D}_y$ is the training labels.



Standard Way





Bayesian Way

A Bayesian Solution

Algorithm 1 Inference procedure for Bayesian Neural Networks

1: Define posterior:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta}')p(\boldsymbol{\theta}')\, d\boldsymbol{\theta}'}$$

2: **for** $i = 0$ to N **do**
3: Sample $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}|\mathcal{D})$
4: Compute $\boldsymbol{y}_i = \Phi_{\boldsymbol{\theta}_i}(\boldsymbol{x})$
5: **end for**
6: **return** $Y = \{\boldsymbol{y}_i \mid i \in [0, N]\}$ and $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i \mid i \in [0, N]\}$.

A Bayesian Solution

Based on $Y = \{\boldsymbol{y}_i \mid i \in [0, N]\}$ and $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i \mid i \in [0, N]\}$,

- Get predictions using **model averaging** as:

$$\hat{\boldsymbol{y}} = \frac{1}{|\boldsymbol{\Theta}|} \sum_{\boldsymbol{\theta}_i \in \boldsymbol{\Theta}} \Phi_{\boldsymbol{\theta}_i}(\boldsymbol{x}_i).$$

- Quantify uncertainty by computing the covariance matrix as:

$$\boldsymbol{\Sigma}_{\boldsymbol{y}|\boldsymbol{x},\mathcal{D}} = \frac{1}{|\boldsymbol{\Theta}| - 1} \sum_{\boldsymbol{\theta}_i \in \boldsymbol{\Theta}} (\Phi_{\boldsymbol{\theta}_i}(\boldsymbol{x}) - \hat{\boldsymbol{y}})(\Phi_{\boldsymbol{\theta}_i}(\boldsymbol{x}) - \hat{\boldsymbol{y}})^T.$$

A Bayesian Solution

For the model selection problem,

- Start by defining a denumerable collection of models $\{M_k\}_{k \in K}$ with model M_k having parameter space $\boldsymbol{\Theta}_k$.
- Perform Bayesian inference by specifying a prior distribution over the collection of models, $\pi(M_k)$, a prior distribution for the parameters of each model, $\pi(\boldsymbol{\theta}_k|M_k)$, and the (model-specific) likelihood $p(\boldsymbol{y}|\boldsymbol{\theta}_k, M_k)$ as follows:

$$\pi(M_k|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|M_k)\pi(M_k)}{p(\boldsymbol{y})}, \tag{1}$$

  where $p(\boldsymbol{y}|M_k) = \int_{\boldsymbol{\theta}_k} p(\boldsymbol{y}|\boldsymbol{\theta}_k, M_k)\pi(\boldsymbol{\theta}_k|M_k)d\boldsymbol{\theta}_k$ is called the **evidence** for model M_k.
- Typically, **evidence is used in deciding which model is better. Higher the evidence, better the model**.

A Bayesian Solution

- How to compute the posterior: $p(\theta|\mathcal{D})$?
- An intractable problem.
- We will approximate it by drawing samples from it using **Adaptive Annealed Sequential Monte Carlo** method.

Adaptive Annealed Sequential Monte Carlo Method

- **Annealed SMC algorithm**
- **Idea**: Assuming $\pi(\boldsymbol{\theta})$ as prior and $\pi_T(\boldsymbol{\theta})$ as the target, create a sequence of distributions, given by $\pi(\boldsymbol{\theta})$ up to $\pi_T(\boldsymbol{\theta})$, which we hope will assist in sampling from $\pi_T(\boldsymbol{\theta})$ and which satisfy $\pi_t(\boldsymbol{\theta}) \neq 0$ wherever $\pi_{t-1}(\boldsymbol{\theta}) \neq 0$.
- Annealing sequence of distribution:

$$\pi_t(\boldsymbol{\theta}) \propto \pi(\boldsymbol{\theta}|M_k)p(\boldsymbol{y}|\boldsymbol{\theta}, M_k)^{\lambda_t}, \tag{2}$$

where $0 = \lambda_0 < \cdots < \lambda_t < \cdots < \lambda_T = 1$.

Adaptive Annealed Sequential Monte Carlo Method

Algorithm 2 Adaptive Annealed SMC

Input: Number of particles N, sequence of distributions $\pi_t(\boldsymbol{\theta})$, and rules for constructing HMC kernel $\mathcal{K}_t^h$.

Output: Weighted particle set $\{(\boldsymbol{\theta}_t^{(i)}, w_t^{(i)})\}_{i=1}^N$, evidence estimate Z

Initialize $t \leftarrow 1$, $\lambda_0 \leftarrow 0$, evidence $E_0 = 1$, sample $\boldsymbol{\theta}_0^{(i)} \sim \pi_0$, weight $W_0^{(i)} = 1/N \; \forall i \in 1 : N$

while $\lambda_{t-1} < 1$ **do**

    Tune HMC kernel parameters h using Algorithm 4.

    Move particles $\boldsymbol{\theta}_t^{(i)} \sim \mathcal{K}_t^h(\boldsymbol{\theta}_{t-1}^{(i)})$ using Algorithm 3.

    Choose next exponent $\lambda_t \in (\lambda_{t-1}, 1]$ using Algorithm 5.

    Update Weight $W_t^{(i)} \propto W_{t-1}^{(i)} p(\boldsymbol{y}|\boldsymbol{\theta}_{t-1}^{(i)}, M_k)^{\lambda_t - \lambda_{t-1}}$.

    Resample particles if necessary.

    Update Evidence $E_t = E_{t-1} \sum_{i=1}^N W_{t-1}^{(i)} p(\boldsymbol{y}|\boldsymbol{\theta}_{t-1}^{(i)}, M_k)^{\lambda_t - \lambda_{t-1}}$

    $t \leftarrow t + 1$.

end while

Adaptive Annealed Sequential Monte Carlo

Algorithm 3 Hamiltonian Monte Carlo (HMC)

Input: Gradient function $\nabla_{\boldsymbol{\theta}} \pi_{t+1}(\cdot)$, initial state $\boldsymbol{\theta}_t$, step size ϵ, number of leapfrog steps L

Output: Next state $\boldsymbol{\theta}_{t+1}$

Sample momentum $\boldsymbol{p}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{M}_t)$

Perform leapfrog integration:

$$(\hat{\boldsymbol{p}}_{t+1}, \hat{\boldsymbol{\theta}}_{t+1}) \leftarrow \hat{\Phi}_{\epsilon, L}(\boldsymbol{p}_t, \boldsymbol{\theta}_t)$$

Sample $u \sim \mathcal{U}[0, 1]$

if $\log u < H(\boldsymbol{p}_t, \boldsymbol{\theta}_t) - H(\hat{\boldsymbol{p}}_{t+1}, \hat{\boldsymbol{\theta}}_{t+1})$ **then**

$\boldsymbol{\theta}_{t+1} \leftarrow \hat{\boldsymbol{\theta}}_{t+1}$

else

$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t$

end if

Adaptive Annealed Sequential Monte Carlo

- Tuning HMC kernel $\implies$ Finding suitable step size ϵ and number of leap frog steps L.
- **Idea**: Use information from the previous iteration - see how previously used ϵ and L performed.
- Rao-Blackwellized estimator of **Estimated Squared Jumping Distance (ESJD)**:

$$\tilde{\Lambda}(\tilde{\boldsymbol{\theta}}_{t-1}^{(i)}, \hat{\boldsymbol{\theta}}_t^{(i)}) = \frac{\|\tilde{\boldsymbol{\theta}}_{t-1}^{(i)} - \hat{\boldsymbol{\theta}}_t^{(i)}\|_{M_t}^2}{L} \times \min\left(1, \exp\left[H(\boldsymbol{p}_{t-1}^{(i)}, \boldsymbol{\theta}_{t-1}^{(i)}) - H(\hat{\boldsymbol{p}}_t^{(i)}, \hat{\boldsymbol{\theta}}_t^{(i)})\right]\right)$$

where $\hat{\boldsymbol{\theta}}_t^{(i)}$ is the proposed position, i.e., based on Hamiltonian flow $\hat{\Phi}_{\epsilon,L}(\boldsymbol{p}_{t-1}^{(i)}, \tilde{\boldsymbol{\theta}}_{t-1}^{(i)})$, $\tilde{\boldsymbol{\theta}}_{t-1}$ is the particle after any resampling step, $||x-y||_M^2 = (x-y)^T M^{-1}(x-y)$, and $M_t = diag(\hat{Var}_{\pi_t}[\boldsymbol{\theta}_t])^{-1}$.

Adaptive Annealed Sequential Monte Carlo

Algorithm 4 Tuning of HMC

1: **Input:** Previous parameters $h_{t-1}^{(i)} = (\epsilon_{t-1}^{(i)}, L_{t-1}^{(i)})$, Rao-Blackwellized estimator of ESJD $\tilde{\Lambda}(\tilde{\boldsymbol{\theta}}_{t-2}^{(i)}, \hat{\boldsymbol{\theta}}_{t-2}^{(i)})$, and perturbation kernel:

$$R(h_t^{(i)}; h_{t-1}^{(i)}) = \mathcal{TN}(\epsilon; \epsilon_{t-1}^{(i)}, 0.015^2) \otimes \left\{ \frac{1}{3}\mathbb{1}_{L=L_{t-1}^{(i)}-1} + \frac{1}{3}\mathbb{1}_{L=L_{t-1}^{(i)}} + \frac{1}{3}\mathbb{1}_{L=L_{t-1}^{(i)}+1} \right\}$$

2: **Output:** Sample of $h_t^{(i)} = (\epsilon_t^{(i)}, L_t^{(i)})$, for $i \in \{1, \ldots, N\}$

3: **for** $i = 1$ to N **do**

4: Sample $h_t^{(i)} \sim \chi_t(h) \propto \sum_{j=1}^{N} \tilde{\Lambda}(\tilde{\boldsymbol{\theta}}_{t-2}^{(j)}, \hat{\boldsymbol{\theta}}_{t-1}^{(j)}) \cdot R(h; h_{t-1}^{(j)})$

5: **end for**

Adaptive Annealed Sequential Monte Carlo

Algorithm 5 Choice of the next exponent based on the effective sample size.

1: **Input:** Target value α, likelihood $I(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta}_t^{(i)})$ for the N particles, and current exponent λ_{t-1}.
2: **Result:** Next exponent λ_t.
3: Define $\beta^{(i)}(\lambda) = I(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta}_t^{(i)})^{\lambda-\lambda_{t-1}}$
4: and compute:

$$ESS(\lambda) = \frac{\left(\sum_{i=1}^{N} \beta^{(i)}(\lambda)\right)^2}{\sum_{i=1}^{N} \left(\beta^{(i)}(\lambda)\right)^2}$$

5: **if** $ESS(1) \geq \alpha N$ **then**
6: Set $\lambda_t = 1$
7: **else**
8: Solve $ESS(\lambda) = \alpha N$ for $\lambda \in (\lambda_{t-1}, 1]$ using bisection
9: $\lambda_t \leftarrow \lambda$.

Experiments

- **A naive example** - Model Selection and accuracy via model averaging.
- The iris dataset.
- Four predictors - Sepal.Length, Sepal.Width, Petal.Length, and Petal.Width $\implies$ input layer has four units.
- Categorical response variable with three categories - setosa, versicolor, and virginica $\implies$ output layer has three units.
- Number of hidden layers - ?, Number of activation units in each hidden layer - ?, activation function - ?

Experiments

- Suppose I am considering this set of models to choose from: {[hidden layers = (1), relu], [hidden layers = (2), relu], [hidden layers = (3), relu], [hidden layers = (4), relu], [hidden layers = (5), relu], [hidden layers = (1, 1), relu], [hidden layers = (2, 2), relu], [hidden layers = (3, 3), relu], [hidden layers = (4, 4), relu], [hidden layers = (2, 2), sigmoid]}.
- Number of particles = 20 and π_0 = Multivariate Logistic Distribution with location = 0.
- $\epsilon_0^{(i)} \sim \mathcal{U}(0, 0.1)$ and $L_0^{(i)}$ is chosen randomly from $\{1, 2, \cdots, 50\}$.
- Train-Test Split = 80:20.

Experiments

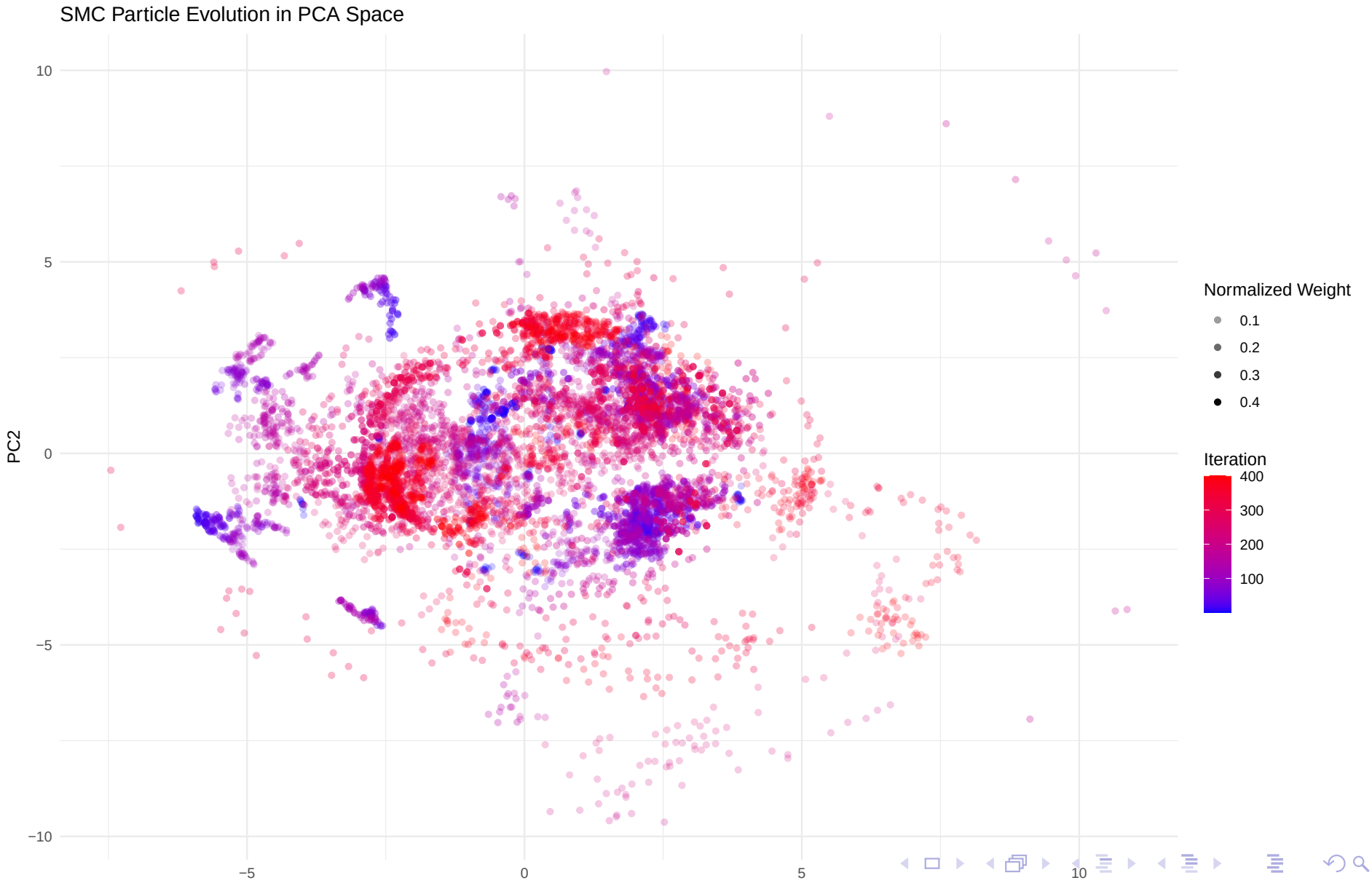
Model	No. of parameters	log evidence estimate
hidden layers = (1), relu	11	-3121.444
hidden layers = (2), relu	19	-3937.799
hidden layers = (3), relu	27	-2571.448
hidden layers = (4), relu	35	-1832.163
hidden layers = (5), relu	43	-1479.915
hidden layers = (1, 1), relu	13	-1196.42
hidden layers = (2, 2), relu	25	-582.9204
hidden layers = (3, 3), relu	39	-589.8898
hidden layers = (4, 4), relu	55	-1214.106
hidden layers = (2, 2), sigmoid	25	-1102.685

Table: Results

Experiments

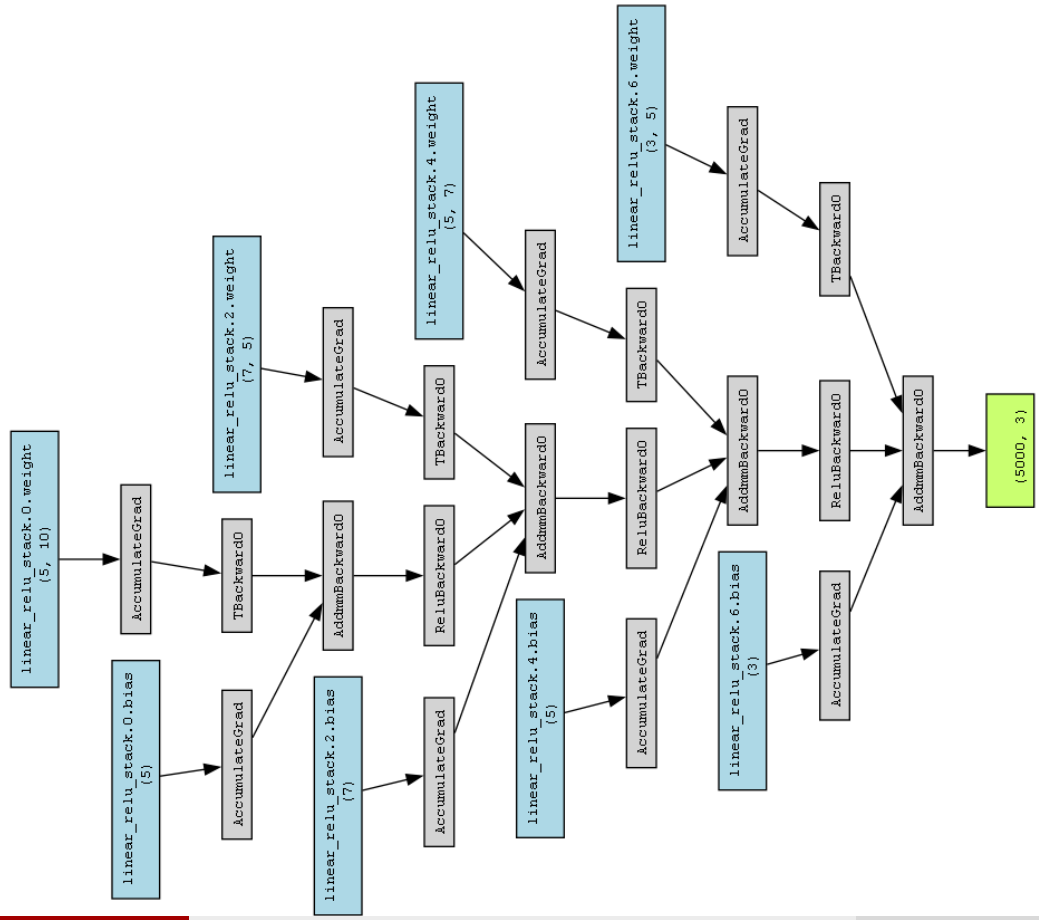
Using BMA in our example, model with hidden layers = (2, 2) and relu non-linearity gives an accuracy of 86.67%.

Experiments



Experiments

- **Simulation Study**
- Coverage probability of HPD intervals based on this approximation of posterior distribution.
- **True Model**:



Experiments

- $N = 400$
- $\pi_0 = N(0, 1)$
- $\epsilon_0^{(i)} \sim \mathcal{U}(0, 0.1)$ and $L_0^{(i)}$ is chosen randomly from $\{1, 2, \cdots, 50\}$.
- All HPD intervals are 95% HPD intervals.
- Total iterations taken were always between 20-25.
- Note: This experiment was performed only 20 times.

Experiments

- HPD intervals for selected parameters:

Param	True Value	Lower HPD	Upper HPD	Covered
39	0.32	-1.935272	0.680883	Yes
40	0.12	-1.719444	0.315818	Yes
41	0.05	-1.840880	0.986922	Yes
42	0.02	-2.517226	-0.075168	No
55	0.01	-0.811074	1.199212	Yes
56	0.10	-0.912664	0.538444	Yes
57	0.20	-0.738993	0.665472	Yes
58	0.40	-0.884004	0.584135	Yes

- In 18/20 runs, HPD intervals for all parameters included the true values.
- Even when not all were covered, about 95% of parameters were.
- **Note: Very limited experiments on a small network so far.**

Thank you!