Deep Learning

Monte Carlo Methods

Weizhu Qian

Email：wzqian@suda.edu.cn

Content

Preliminary

Let $D = \{X, Y\}$ a training dataset of i.i.d samples, $f_\theta\ (x)$ a neural network parameterized by the model parameters $\theta \in \mathbb{R}^d$, $p(\theta|D)$ the posterior, $q(\theta)$ the prior, $p(D|\theta)$ the likelihood(related to loss function).

NB: we only consider the solutions in weight (not function) space here.

Bayes' Theorem:

$$p(\theta|D) = \frac{p(D|\theta)q(\theta)}{Z}, \qquad \text{where } Z = \int_{\mathbb{R}^d} p(D|\theta)q(\theta)d\theta$$

Note: Z is the normalization constant which is intractable.

The logarithm of $p(\theta|D) = \frac{p(D|\theta)q(\theta)}{Z}$ is:

$$\log p(\theta|D) = \log p(D|\theta)q(\theta) - \log Z$$

$$\arg\max_{\theta} \log p(\theta|D) = \arg\max_{\theta} \log p(D|\theta)q(\theta) \cancel{- \log Z}$$

Above can be solved by minimizing the loss function via SGD.

Denote the approximated posterior $\pi(\theta) \coloneqq p(\theta|D)$, and

recall the target function (neural network) is:

$$y = f_{\theta}\,(x)$$

Bayesian Model Averaging (BMA)

Our goal is to estimate the **Expectation** of f_θ given a new sample x

$$\hat{y} = \int_{\mathbb{R}^d} f_\theta(x) d\pi(\theta) .$$

Now we can resort to the classic Monte Calo method:

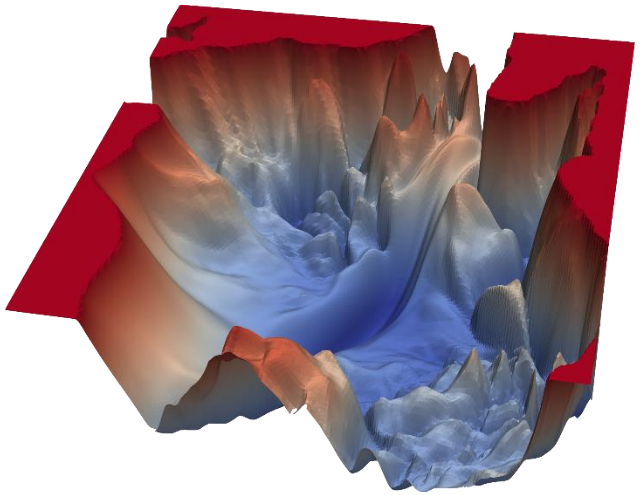
$$\hat{y} = \mathbb{E}_{\theta \sim \pi(\theta)}[f_\theta(x)] \\ \approx \frac{1}{M} \sum_{m=1}^{M} f_{\theta^m}(x),$$

Key point: Cast integration into expectation such that it can be solved numerically via sampling!

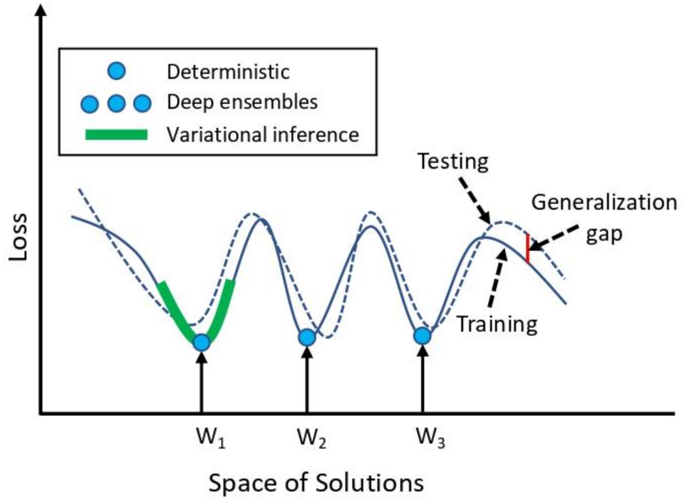
where M is the number of trained models, i.e., we have a set of different trained models and use the averaged predictions as the final prediction.

Solution Space of Deep Neural Networks

We rely on Stochastic Gradient Descent (SGD) to solve DNNs.



Loss landscape of a deep neural network, highly non-convex!



Solution space and loss, there are many (almost) equally good solutions!

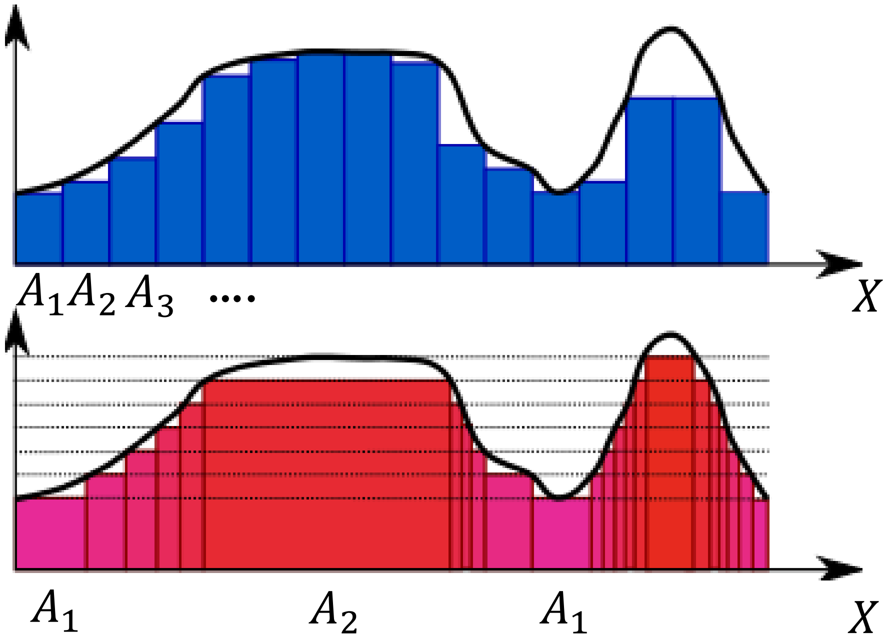
Note: Trained models with high losses are not valid solutions, which results in disconnected paths and multi-modality.

Monte Carlo Integration

We now need to compute $\hat{y} \approx \frac{1}{M} \sum_{\theta \in \Theta} f_\theta(x)$.

Riemann Integral:

Lebesgue Integral:



We only need to train a set of models to compute the Lebesgue integral.

Question: Why not Riemann Integral?

Answer: The partitioning is hard when the measure space is discrete and high dimensional.

Measure space (X, A, μ),
where X: set, $\mathbb{R}^d$;
A: Borel σ-algebra(minimum collections of subsets of X);
μ: probability measure such that $\mu: A \to [0,1]$.

A Typical Realization: Bayesian Neural Networks (BNNs)

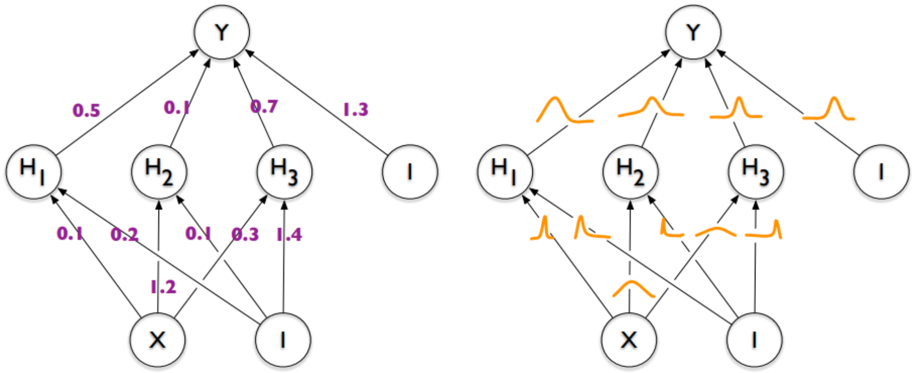


Figure 1. Left: each weight has a fixed value, as provided by classical backpropagation. Right: each weight is assigned a distribution, as provided by Bayes by Backprop.

Key point: Each weight θ_i is with respect to some distribution, e.g., a (mixed) Gaussian.

Alternatives: Langevin and Hamiltonian dynamics, Dropout, SVGD, etc.

References

1. Deep Learning, Ian Goodfellow, Yoshua Bengio and Aaron Courville, MIT Press, 2016.

2. Probabilistic Machine Learning: An Introduction, Kevin Patrick Murphy, MIT Press, March 2022.

3. Blundell, Charles, et al. "Weight uncertainty in neural network." *International conference on machine learning*. PMLR, 2015.

4. Sampling as First-Order Optimization over a space of probability measures, Anna Korba, Adil Salim, ICML 2022 tutorial.

5. Fort, Stanislav, Huiyi Hu, and Balaji Lakshminarayanan. "Deep ensembles: A loss landscape perspective." *arXiv preprint arXiv:1912.02757* (2019).