Weekly Study Report

Wang Ma

2024-12-16

Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute

Outline

1. Fixed-Mean Gaussian Processes for Post-hoc Bayesian Deep Learning

1.1 Treat DNN as Black-Box Function

A *function-space* perspective:

- A prior on $p(f)$, target posterior: $p(f|\ \mathbf{y}) \approx q(f)$.
- The predictions $p(y^*|x^*, \mathbf{y}) = \mathbb{E}_{p(f|\mathbf{y})}[p(y^*|x^*, f)] \approx \mathbb{E}_{q(f)}[p(y^*|x^*, f)]$

Gaussian Process.

Given the network f_θ, we assume that

$$f_\theta(X) \sim N(m(X), k(X, X)),$$

where X is the total data set, $m(\cdot)$ and $k(\cdot, \cdot)$ are human-defined. In inference time, we use the properties of conditional multivariate normal distribution:

$$P(f(x^*)|f(X))) \sim N(\mu(x), \sigma^2(x)),$$

where $\mu(\cdot)$ and $\sigma^2(\cdot)$ have closed-form solutions.

- $f(x) \sim N(\mu(x), \sigma^2(x))$, but the complexity is $O(N^3)$.

Sparse Variational Gaussian Process (SGVP).

- M inducing points $z_1, z_2, ..., z_M$, instead of the whole data set. $u_i = f(z_i)$
- The posterior $p(f, u|\mathrm{y}) \approx q(f, u) = p(f|u)q(u)$,

$$p(f|u) = N(\mu_Z(x), \sigma_Z^2(x))$$

and $q(u)$ is the posterior of the inducing points (the learned points), while $p(u)$, the prior, is the starting inducing point.

Learning Objective.

$$L_{ELBO} = \mathbb{E}_{q(f)}[\log(p(y|f))] - KL(q(f)\|p(f))$$

To train a related parameters A and the inducing points set Z.suo y

Inference Equation: $f(x^*|\mathbf{y}) \sim N(m^*(x^*), v^*(x^*))$, where

$m(x^*) = f(x^*)$

$v^*(x^*) = K(x^*, x^*) - K_{x^*,Z} A^{-1} K_{Z,x^*}$.

My Decomposition:

- Total Uncertainty: $K(x^*, x^*)$ (the variance in x^* point itself)
- Epistemic Uncertainty: $-K_{x^*,Z} A^{-1} K_{Z,x^*}$, measures the distance between x^* and Z.
- Aleatoric Uncertainty: $v^*(x^*)$, the variance of prediction $y^*|x^*, \theta$

1.4 Conclusion

Strength.

1. It's a good idea to model the output space by Bayesian Optimization.
2. The paper provide complete, closed-form solutions for the prediction.
3. Applicable on classification tasks, and scale to large dataset like ImageNet.

Limitation.

1. The computation of inverse Kernel matrix, cubic cost of the inducing points. Bayesian Optimization itself can not scale well. For a complicated problem, we may need thousands of inducing points.
2. Although post-hoc, it requires additional optimization/training.if the additional training data has a different distribution with initial training data of the DNN (same distribution), can it work well?
3. Choosing the Kernel can be a strength making it flexible, but it may be difficult to efficiently use an effective kernel in complicated tasks.

2. Make Me a BNN: A Simple Strategy for Estimating Bayesian Uncertainty from Pre-trained Models

2.1 Illustration

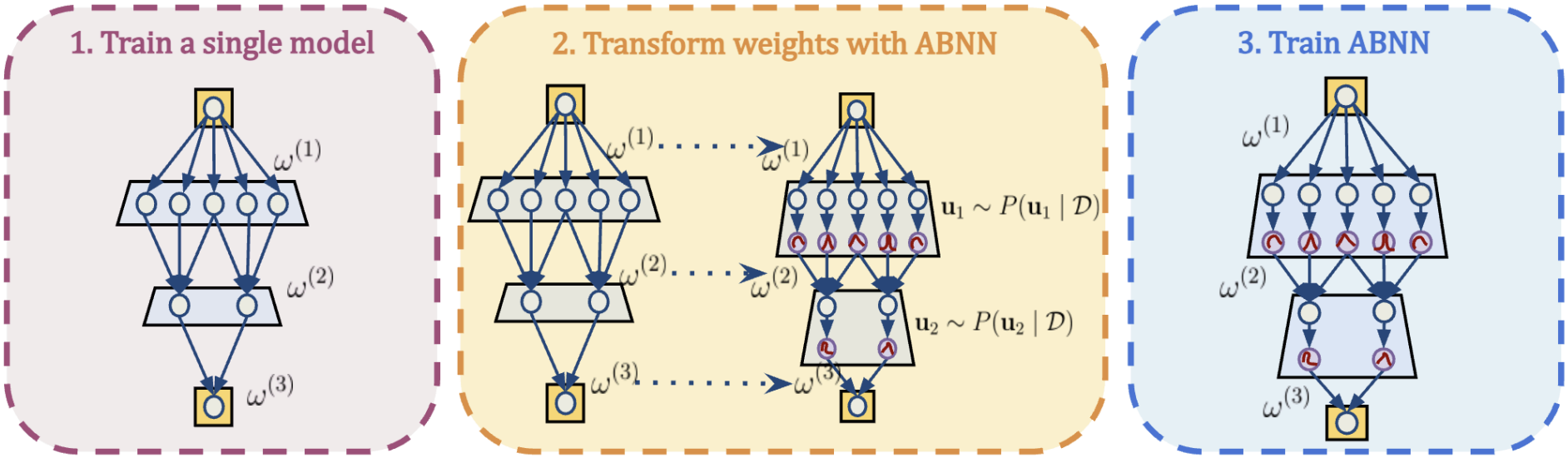


Figure 2. **Illustration of the training process for the ABNN.** The procedure begins with training a single DNN $\boldsymbol{\omega}_{\text{MAP}}$, followed by architectural adjustments to transform it into an ABNN. The final step involves fine-tuning the ABNN model.

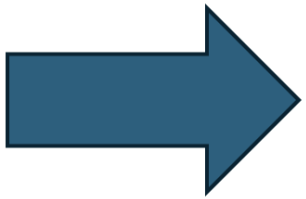
$$\mathbf{h}_1 = W^{(1)}\mathbf{x}$$

$$\mathbf{u}_1 = \text{norm}(\mathbf{h}_1, \beta_1, \gamma_1) = \frac{\mathbf{h}_1 - \hat{\mu}_1}{\hat{\sigma}_1} \times \gamma_1 + \beta_1$$

$$\mathbf{a}_1 = a(\mathbf{u}_1)$$

$$\mathbf{u}_2 = \text{norm}\left(W^{(2)}\mathbf{a}_1, \beta_2, \gamma_2\right), \text{ and } \mathbf{a}_2 = a(\mathbf{u}_2)$$

$$\mathbf{h}_3 = W^{(3)}\mathbf{a}_2, \text{ and } P(y \mid \mathbf{x}, \boldsymbol{\omega}) = \text{soft}(\mathbf{h}_3),$$



$$\mathbf{u}_j = \mathbf{BNL}\left(W^{(j)}\mathbf{h}_{j-1}\right), \text{ and}$$

$$\mathbf{a}_j = a(\mathbf{u}_j) \text{ with}$$

$$\mathbf{BNL}(\mathbf{h}_j) = \frac{\mathbf{h}_j - \hat{\mu}_j}{\hat{\sigma}_j} \times \gamma_j(1 + \boldsymbol{\epsilon}_j) + \beta_j.$$

Intuition: introducing perturbation on NL can reduce the variance of Gradients, giving a stable retraining process (stable than VI-BNN).

2.3 Procedure

1. Given pre-trained model f
2. Replace the NL, using BNL to introduce a gaussian noise ε:

$$BNL(h_j) = \frac{h_j - \hat{\mu}_j}{\hat{\sigma}_j} \times \gamma_j(1 + \varepsilon_j) + \beta_j$$

   .

3. Retrain the model to update γ and β, sample ε in every forward process.
4. Step training at the max epochs, normally $(2 - 5)$.
5. During the Inference time, also randomly sample ε.
6. A single ABNN only has ε as the r.v., while it can do UQ with MC sampling. The author propose that it is better to train multiple ABNNs.

Strength.

1. Simple trick, easy to use, workable, extendable to many tasks. (on CIFAR-10, ABNN takes 2 hrs, while DE takes 6.8 hours, both for 3 instances)
2. It is stable because the low variance of the gradients.

Limitations.

1. The adding ε_j introduces some diversity, but it can not meet the level of deep ensemble. It can introduce diversity locally, does not provide a good posterior estimation globally. The author introduced an extra random loss term to introduce more diversity.
2. ABNN may be overfitting if previous DNN is overfitting.
3. Retraining part can be time consuming sometimes. And it requires the same data from training set (default setting in the paper).

3. Efficient Uncertainty Quantification and Reduction for Over-Parameterized Neural Networks

3.1 Three sources of Epistemic Uncertainty

1. **Model Approximation Error**: $\mathrm{UQ}_{\mathrm{AE}} = h^* - h_B^*$, where h^* is the ideal network when data in infinite, h_B^* is the best predictor - *bayesian predictor*. We call it best, because it can minimize the global risks $\mathbb{E}_{(X,Y)\sim\pi}[L(f(x), y)]$. It can be ignored because the perfect approximation probability of NN.
2. **Data Variability**: $\mathrm{UQ}_{\mathrm{DV}} = \hat{h}_n^* - h^*$, where $\hat{h}_n^* = \mathbb{E}_{\gamma\sim P_{\gamma|D}}\left[\hat{h}_{n,\gamma}\right]$, the ideal ensemble results with all possible initialization. This measures the representativeness of the training data, whihc is the most standard epistemic uncertainty in classical statistics.
3. **Procedural Variability**: $\mathrm{UQ}_{\mathrm{PV}} = \hat{h}_{n,\gamma} - \hat{h}_n^*$. This arises from the randomness in the training process for a single network $\hat{h}_{n,\gamma}$, which is present even with deterministic or infinite data. The randomness comes from the *initialization* of the network parameters, and also *data ordering* and possibly training time when running *stochastic gradient descent* with finite training epochs.

3.2 Quantifying Epistemic Uncertainty

1. With assumptions, training a single NN $\hat{h}_{n,\theta_b}(x)$ is equivalent to

$$\hat{h}_{n,\theta_b}(x) = s_{\theta_b}(x) + K(x,x)^T(K(x,x) + \lambda_n I)^{-1}\Big(y - s_{\theta_b}(x)\Big).$$

   This means Procedural Variability is strongly related to initialization.

2. For a DE $\hat{h}_n^m(x) = \frac{1}{m}\sum_{i=1}^{m} \hat{h}_{n,\theta_b^i}(x)$, they proved that

$$\mathrm{Var}\Big(\hat{h}_n^m(x)\Big) = \mathrm{Var}\Big(\hat{h}_n^*(x)\Big) + \frac{1}{m}\mathbb{E}\Big[\mathrm{Var}\Big(\hat{h}_{n,\gamma}(x)|D\Big)\Big].$$

   The reveals that DE can reduce Procedural Variability and cannot reduce data variabilty.

3. To reduce the procedural variability directly whether using DE, so the author proposed to use an auxillary network(procedural noise predictor) to reduce procedural variablity. And it is proved that $\hat{h}_n^{\mathrm{PNC}}(x) = \hat{h}_n^*(x)$ exactly.

3.3 Overall Steps

Goal: Decouple the Data Variability and Procedural Variability.

1. Trined NN $h_{n,\theta_b}(x)$, where θ_b is the initialization. Training data $\{(x_i, y_i)\}$
2. Construct an artificial Dataset $\{(x_i, \hat{s}(x_i))\}$, where

$$\hat{s}(x) = \mathbb{E}_{\theta_b}\left[s_{\theta_b}(x)\right].$$

3. Train an auxillary NN $\varphi'_{n,\theta_b}(x)$ on $\{(x_i, \hat{s}(x_i))\}$, then define

$$\varphi_{n,\theta_b}(x) = \varphi'_{n,\theta_b}(x) - \hat{s}(x_i))$$

4. Then the modified output $\hat{h}^*_n = h_{n,\theta_b}(x) - \varphi_{b,\theta_b}(x)$.

Proved: $\hat{h}^*_n(x) = \mathbb{E}_{\theta_b}\left[h_{n,\theta_b}(x)\right]$.

The Procedural-Noise-Correcting (PNC) Predictor's output equals to an ideal Deep Ensemble.

3.4 Constructing the Confidence Interval for h^*

The $1-\alpha$-level confidence interval:

$$\left[h_{n,\theta_b}(x) - \varphi_{b,\theta_b}(x) - \frac{\sigma(x)}{\sqrt{n}} q_{1-\alpha}, h_{n,\theta_b}(x) - \varphi_{b,\theta_b}(x) + \frac{\sigma(x)}{\sqrt{n}} q_{1-\alpha}\right].$$

Algorithm 2 PNC-Enhanced Batching

Input: Training dataset $\mathcal{D}$ of size n. The number of batches $m' \geq 2$.

Procedure: **1.** Split the training data $\mathcal{D}$ into m' batches and input each batch in Algorithm 1 to output $\hat{h}^j_{n',\theta^b}(x) - \hat{\phi}^j_{n',\theta^b}(x)$ for $j \in [m']$, where $n' = \frac{n}{m'}$.

2. Compute $\psi_B(x) = \frac{1}{m'} \sum_{j=1}^{m'} \left(\hat{h}^j_{n',\theta^b}(x) - \hat{\phi}^j_{n',\theta^b}(x)\right)$,

and $S_B(x)^2 = \frac{1}{m'-1} \sum_{j=1}^{m'} \left(\hat{h}^j_{n',\theta^b}(x) - \hat{\phi}^j_{n',\theta^b}(x) - \psi_B(x)\right)^2$.

Output: At point x, output $\psi_B(x)$ and $S_B(x)^2$.

($m' \in [5, 10]$)

3.4 Constructing the Confidence Interval for h^*

Algorithm 3 PNC-Enhanced Cheap Bootstrap

Input: Training dataset $\mathcal{D}$ of size n. The number of replications $R \geq 1$.

Procedure: **1.** Input $\mathcal{D}$ in Algorithm 1 to output $\hat{h}_{n,\theta^b}(x) - \hat{\phi}_{n,\theta^b}(x)$.

2. For each replication $j \in [R]$, resample $\mathcal{D}$, i.e., independently and uniformly sample with replacement from $\{(x_1, y_1), ..., (x_n, y_n)\}$ n times to obtain $\mathcal{D}^{*j} = \{(x_1^{*j}, y_1^{*j}), ..., (x_n^{*j}, y_n^{*j})\}$. Input $\mathcal{D}^{*j}$ in Algorithm 1 to output $\hat{h}^{*j}_{n,\theta^b}(x) - \hat{\phi}^{*j}_{n,\theta^b}(x)$.

3. Compute $\psi_C(x) = \hat{h}_{n,\theta^b}(x) - \hat{\phi}_{n,\theta^b}(x)$,

and $S_C(x)^2 = \frac{1}{R}\sum_{j=1}^{R}\left(\hat{h}^{*j}_{n,\theta^b}(x) - \hat{\phi}^{*j}_{n,\theta^b}(x) - \psi_C(x)\right)^2$.

Output: At point x, output $\psi_C(x)$ and $S_C(x)^2$.

(R can be small and reach a good coverage)

3.5 Conclusion

Strength.

1. Interesting way to make a single model with an auxillary model to get same results from Deep Ensemble.
2. Reduce the procedure variability via a simple auxillary model. Achieve an equal results with DE with much less computation.

Limitation.

1. PNC method needs the assumption of over-parametrization of NN and NTK theory. So the NN width and initializaiton can be significant.
2. This paper only focus on Regression.
3. But the CI construction part needs more training (why just use Ensemble?) Can we do like first paper, find the most "representable" points serve as the inducing points.