Generative Posterior Networks for Approximately Bayesian Epistemic Uncertainty Estimation

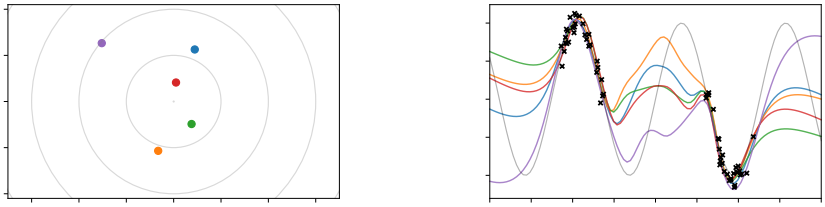
Anonymous Author(s)
Affiliation
Address
email

Abstract

In many real-world problems, there is a limited set of training data, but an abundance of unlabeled data. We propose a new method, Generative Posterior Networks (GPNs), that uses unlabeled data to estimate epistemic uncertainty in high-dimensional problems. A GPN is a generative model that, given a prior distribution over functions, approximates the posterior distribution directly by regularizing the network towards samples from the prior. We prove theoretically that our method indeed approximates the Bayesian posterior and show empirically that it improves epistemic uncertainty estimation and scalability over competing methods.

1 Introduction

In supervised learning tasks, the distribution of labeled training data often does not match exactly with the distribution of data the model will see at deployment. This distributional shift can cause significant problems in safety-critical environments where mistakes can be catastrophic. Ideally, we want our learned models to be able to estimate their epistemic uncertainty – uncertainty deriving from lack of data samples. Unfortunately, deep learning models struggle to estimate this type of uncertainty. While there are many proposed methods for addressing this problem, epistemic uncertainty estimation in deep learning remains an open problem due to out of distribution (OOD) performance and scalability.



(a) 2D embedding samples

(b) Corresponding posterior samples

Figure 1: Samples from a GPN using a 2D embedding trained on a simple sine-function. On the top are samples from the embedding with corresponding posterior samples underneath. Black ‘x’s represent observed data points.

Concurrently, many recent works have shown incredible performance using unlabeled data (Radford et al., 2019; Chen et al., 2020). While labeled training data is expensive to collect, in many real-world

Submitted to the Workshop on Distribution Shifts, 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

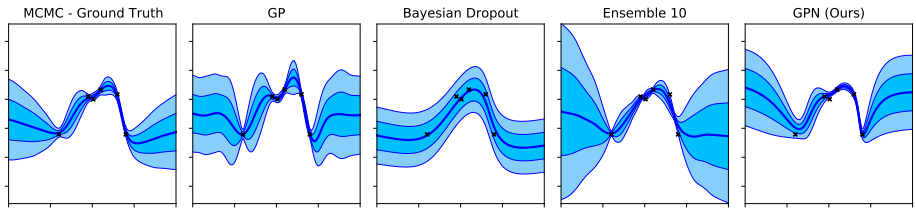


Figure 2: Predicted posterior distributions of different methods using the same observed data.

problems, such as image classification, there is an abundance of unlabeled data. Moreover, this unlabeled data is often much more representative of the deployment distribution. In this work, we propose a new method for estimating epistemic uncertainty that leverages this unlabeled data.

Our work builds off of the extensive ensembling literature. Neural network ensembling is a method that predicts epistemic uncertainty by looking at the disagreement between the ensemble members. Additionally, under the right regularization, ensembles approximate samples from the Bayesian posterior Pearce et al. (2020); He et al. (2020). However, these methods have one main drawback, namely that each new sample from the posterior requires training a new network from scratch.

To address this challenge, we introduce Generative Posterior Networks (GPNs), a generative neural network model that directly approximates the posterior distribution by regularizing the output of the network towards samples from the prior distribution. By learning a low-dimensional latent representation of the posterior, our method can quickly sample from the posterior and construct confidence intervals (CIs). We prove that our method approximates the Bayesian posterior over functions and show empirically that our method can improve uncertainty estimation over competing methods. While not motivated by the Bayesian Inference problem, Osband et al. (2021) (a concurrent work) propose Epistemic Neural Networks, which is very similar to our method. The key difference between our method and Epistemic NNs is in the regularization: our method takes advantage of unlabeled data to regularize towards *outputs* as opposed to the *parameters* of the prior networks. In our experiments we show empirically our method improves posterior sampling and OOD detection.

2 Generative Posterior Networks

Randomized MAP Sampling We consider a slight variation on the usual Bayesian Inference problem: instead of finding the posterior of the *parameters* of a function, we will instead find the posterior of the *outputs* of that function. Specifically, consider the transformed random variable $\hat{\mathbf{Y}} = f(\mathbf{x}_{\text{sample}}; \boldsymbol{\theta})$ for some function f parameterized by $\boldsymbol{\theta}$ and some set of unlabeled sample points $\mathbf{x}_{\text{sample}}$. We assume a known prior over parameters $P(\boldsymbol{\theta}) \sim \mathcal{N}(\mu_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ as well as access to a noisy dataset $(\mathbf{x}_{\text{obs}}, \mathbf{y}_{\text{obs}}) = \{(x^1_{\text{obs}}, y^1_{\text{obs}}) \ldots, (x^N_{\text{obs}}, y^N_{\text{obs}})\}$, where the observations $y^i_{\text{obs}} = f(x^i_{\text{obs}}; \boldsymbol{\theta}) + \epsilon$ depend on an unknown parameter $\boldsymbol{\theta}$ and are corrupted by Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon)$. The goal of this Bayesian Inference problem is to approximate samples from the posterior,

$$P(\hat{\mathbf{Y}} \,|\, \mathbf{y}_{\text{obs}}) \propto P(\mathbf{y}_{\text{obs}} \,|\, \hat{\mathbf{Y}}) P(\hat{\mathbf{Y}}). \tag{1}$$

Pearce et al. (2020) showed how we can use a method called Randomized MAP Sampling (RMS) to approximately sample from the posterior distribution. The idea behind RMS is to use the fact that the parameters that minimize the regularized MSE loss are equal to the maximum a posteriori (MAP) solution for the posterior. Thus, using optimization techniques, like gradient descent, we can easily find MAP solutions. However, the challenge of sampling from the posterior remains. Instead, RMS randomly samples shifted prior *distributions* such that the resulting MAP solutions for each of these shifted distributions form samples from the desired posterior.

Unlike Pearce et al. (2020), we consider using the RMS technique on our modified Bayesian Inference problem, to estimate the posterior of the *outputs* $\hat{\mathbf{Y}}$ as opposed to the *parameters* $\boldsymbol{\theta}$. Specifically, RMS in this setting samples "anchor points" $\hat{\mathbf{Y}}_{\text{anc}}$ from some distribution $\hat{\mathbf{Y}}_{\text{anc}} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{anc}}, \boldsymbol{\Sigma}_{\text{anc}})$. Each anchor point corresponds to the mean of a shifted prior distribution $P_{\text{anc}}(\hat{\mathbf{Y}}) = \mathcal{N}(\hat{\mathbf{Y}}_{\text{anc}}, \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}})$.

Table 1: Superconductor regression (Hamidieh, 2018) results.

Methods	In Dist. Loss	In Dist. CI-width	OOD CI-correct	OOD-detect. AUC
Dropout	**0.10**	0.32	17.7%	0.56
DKL GP	0.11	0.74	37.9%	0.52
SNGP	**0.10**	1.45	87.8%	0.82
Ensemble - PR	0.12	**0.23**	83.8%	0.93
Ensemble - OR	0.11	**0.23**	75.4%	0.95
Epistemic NN	0.12	1.07	**95.3%**	0.92
GPN (Ours)	0.11	0.31	86.2%	**0.97**

Table 2: CIFAR-10 results with Out of Distribution (OOD) evaluation on SVHN.

Methods	In Dist. Accuracy	In Dist. Entropy	OOD Entropy	OOD-detect. AUC
Dropout	80.7%	0.579	1.214	0.76
DKL GP	79.3%	**0.027**	0.372	0.76
SNGP	77.2%	0.386	0.794	0.77
Ensemble - PR	**80.9%**	0.882	1.344	0.68
Ensemble - OR	80.7%	0.706	2.018	0.98
Epistemic NN	77.9%	0.630	1.145	0.76
GPN (Ours)	79.4%	0.621	**2.256**	**1.00**

Now we define the MAP function which finds the MAP solution using this shifted prior:

$$Y_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}}) = \arg\max_{\hat{\mathbf{Y}}} P(\mathbf{y}_{\text{obs}} \mid \hat{\mathbf{Y}}) P_{\text{anc}}(\hat{\mathbf{Y}}). \tag{2}$$

By sampling different anchor points, we can then construct a probability distribution over MAP solutions $P(Y_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}}))$. If we assume that the prior over $\hat{\mathbf{Y}}$ is Gaussian, this distribution is equivalent to the posterior distribution $P(\hat{\mathbf{Y}} \mid \mathbf{y}_{\text{obs}})$ for a specific anchor distribution. While this Gaussian prior assumption does not hold exactly in general, it will hold exactly when f is linear or has an infinitely wide final layer.

Theorem 2.1. *Assume that the prior distribution of $\hat{\mathbf{Y}}$ is Gaussian, $P(\hat{\mathbf{Y}}) = \mathcal{N}(\boldsymbol{\mu}_{\hat{\mathbf{Y}}}, \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}})$. If we choose the distribution over $\hat{\mathbf{Y}}_{anc}$ to be $P(\hat{\mathbf{Y}}_{anc}) = \mathcal{N}(\boldsymbol{\mu}_{anc}, \boldsymbol{\Sigma}_{anc})$, where $\boldsymbol{\mu}_{anc} = \boldsymbol{\mu}_{\hat{\mathbf{Y}}}$ and $\boldsymbol{\Sigma}_{anc} = \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}} + \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}\boldsymbol{\Sigma}_{like}^{-1}\boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}$, then $P(\hat{Y}_{MAP}(\hat{\mathbf{Y}}_{anc})) = P(\hat{\mathbf{Y}} \mid \mathbf{y}_{obs})$.*

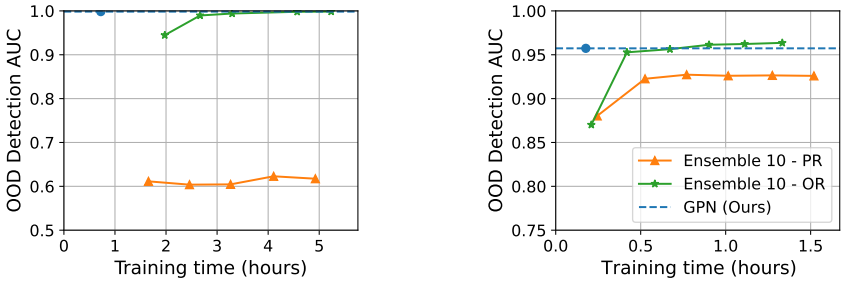
Our proof follows a similar technique to Pearce et al. (2020) and is provided in Appendix E.1. Just as in Pearce et al. (2020), we use the approximation $\boldsymbol{\Sigma}_{\text{anc}} \approx \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}$ to make it tractable to sample from the anchor distribution. [1] With this assumption, the anchor point sampling distribution $\hat{\mathbf{Y}}_{\text{anc}} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{anc}}, \boldsymbol{\Sigma}_{\text{anc}})$ becomes approximately equal to the output prior distribution $P(\hat{\mathbf{Y}})$.

Generative Model While RMS can be used to sample from the posterior, every new sample requires solving a new optimization problem. Instead, we propose to learn a generative model to approximate the MAP function itself. Let g be a neural network parameterized by some vector, ϕ, that takes as input a sample from the anchor distribution, $\hat{\mathbf{Y}}_{\text{anc}}$, and outputs the MAP solution $\hat{\mathbf{Y}}_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}})$. Since we are using the prior distribution as our anchor distribution, we can sample anchor points using $\hat{\mathbf{Y}}_{\text{anc}} = f(\mathbf{x}_{\text{sample}}; \boldsymbol{\theta}_{\text{anc}})$, where $\boldsymbol{\theta}_{\text{anc}} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$. If we assume that g has enough expressive power to represent the MAP function, then $g(\hat{\mathbf{Y}}_{\text{anc}}; \boldsymbol{\phi}) = \hat{Y}_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}})$, if $\boldsymbol{\phi}$ is equal to:

$$\arg\min_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{\theta}_{\text{anc}}} \sum_{i=1}^{N} \left(\|y_{\text{obs}}^i - g(x_{\text{obs}}^i, \boldsymbol{\theta}_{\text{anc}}; \boldsymbol{\phi})\|_2^2 + \sigma_\epsilon^2 \boldsymbol{\delta}^T \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1} \boldsymbol{\delta} \right) \tag{3}$$

where $\delta_j = g(x_{\text{sample}}^j, \boldsymbol{\theta}_{\text{anc}}; \boldsymbol{\phi}) - f(x_{\text{sample}}^j; \boldsymbol{\theta}_{\text{anc}})$. See Appendix E.2 for a step-by-step derivation.

[1] Pearce et al. (2020) argue that this approximation, in general, causes RMS to only over-estimate the posterior variance.



(a) Classification - CIFAR-10

(b) Regression - Superconductor

Figure 3: OOD detection AUC vs. training time on the Superconductor and CIFAR-10 datasets for parameter and output-regularized ensembles and our method (GPN).

In practice, of course, it is expensive to compute the regularization term, $\sigma_\epsilon^2 \boldsymbol{\delta}^T \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1} \boldsymbol{\delta}$, for a large set of sample points $\mathbf{x}_{\text{sample}}$. Instead, we approximate this term with $\beta \|\delta\|_2^2$ where β is a hyper-parameter. This assumes independence and uniform variance of the prior, but we found it to work well in practice. Finally, optimizing the generative function g over the the space of all anchor parameters $\boldsymbol{\theta}_{\text{anc}}$ is impractical, as this space is too high dimensional. Instead, we construct a low-dimensional embedding for $\boldsymbol{\theta}_{\text{anc}}$. Details for how we construct this embedding are provided in Appendix B.

3 Experiments

The goal of our experiments is to illustrate the ability of our method to accurately model epistemic uncertainty for OOD data while retaining high performance on in distribution data. To do so, similar to related work (Liu et al., 2020; Van Amersfoort et al., 2020), we consider two separate datasets, "In Distribution" and "Out of Distribution" (OOD). At training time, we provide labeled data from the In Distribution dataset and unlabeled data from both datasets. We evaluate each model on a hold-out set of In Distribution and OOD data. Our first experiment is a small 1-dimensional regression problem where we can easily compute the ground truth (Figure 2). For our high-dimensional regression task, we use a Superconductivity prediction task (Hamidieh, 2018), splitting the data into In Distribution and OOD based on target values (Table 1). For our classification tasks, we use the MNIST and CIFAR data for In Distribution data and Fashion MNIST and SVHN for OOD data (Tables 2 and 3).

One desired property of posterior approximators is that they have low variance on in distribution data and high variance on OOD data. Thus, along with test loss/accuracy, we report confidence interval width/correctness (for regression) and posterior entropy (for classification). We also use posterior sample variance to construct ROC curves for OOD-detection and measure the area under the curve (AUC). Finally, to evaluate scalability of our method, we trained increasingly larger ensembles and measured their OOD-detection performance vs. training time. See Appendix D for full details.

While all methods are able to achieve high In Distribution performance (loss or accuracy), the GPN is able to consistently achieve the highest OOD-detection AUC. The SNGP and OR-ensemble are also able to achieve high OOD-detection AUC, but have other drawbacks. Specifically, SNGP has a very high CI-width on In Distribution data for the regression task and a very small contrast in sample entropy between the In Distribution and OOD data for the classification tasks. And, while the OR-ensemble performs well, Figure 3 shows GPN scales much better.

4 Conclusion

In this paper we introduce Generative Posterior Networks (GPNs), a method that uses unlabeled data to learn a generative model of the Bayesian posterior distribution. We prove that under mild assumptions, GPNs approximate samples from the true posterior. We then show empirically that our method significantly outperforms competing epistemic uncertainty predictions techniques on high-dimensional classification tasks and scales much better than ensembling methods, the closest performing baseline.

References

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pp. 1613–1622. PMLR, 2015.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017.

Hamidieh, K. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154:346–354, 2018.

He, B., Lakshminarayanan, B., and Teh, Y. W. Bayesian deep ensembles via the neural tangent kernel. *arXiv preprint arXiv:2007.05864*, 2020.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

Liu, J., Lin, Z., Padhy, S., Tran, D., Bedrax Weiss, T., and Lakshminarayanan, B. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in Neural Information Processing Systems*, 33:7498–7512, 2020.

Osband, I., Wen, Z., Asghari, M., Ibrahimi, M., Lu, X., and Van Roy, B. Epistemic neural networks. *arXiv preprint arXiv:2107.08924*, 2021.

Pearce, T., Leibfried, F., and Brintrup, A. Uncertainty in neural networks: Approximately bayesian ensembling. In *International conference on artificial intelligence and statistics*, pp. 234–244. PMLR, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Tanner, M. A. and Wong, W. H. The calculation of posterior distributions by data augmentation. *Journal of the American statistical Association*, 82(398):528–540, 1987.

Titsias, M. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pp. 567–574. PMLR, 2009.

Van Amersfoort, J., Smith, L., Teh, Y. W., and Gal, Y. Uncertainty estimation using a single deep deterministic neural network. In *International conference on machine learning*, pp. 9690–9700. PMLR, 2020.

Wilson, A. and Nickisch, H. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International Conference on Machine Learning*, pp. 1775–1784. PMLR, 2015.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016.

A Related Work

There are two primary methods of formalizing epistemic uncertainty: (1) as a Gaussian Process (GP) or (2) as a Bayesian Inference problem. The GP framework is often used in practice, specifically in low-dimensional, low-data settings where the problem can be solved exactly (Hensman et al., 2013; Titsias, 2009). Many GP approximation methods have been proposed to improve scalability (Wilson & Nickisch, 2015; Wilson et al., 2016), but each have their drawbacks. We highlight Spectral-normalized Neural Gaussian Processes (SNGP) (Liu et al., 2020) as one of the best GP approximation methods for high dimensional problems. While SNGP shows impressive out of distribution prediction performance, we show in our experiments that the learned posterior does not adequately capture the true posterior.

The Bayesian Inference formulation, considers a different problem: given a prior distribution over functions and some labeled data, the goal is to construct a posterior distribution in the Bayesian sense. Because this posterior distribution is usually intractable to compute exactly, approximate sampling methods are used instead (Tanner & Wong, 1987; Blei et al., 2017). For high-dimensional problems, Bayesian Neural Networks (BNNs) (Blundell et al., 2015) and Bayesian Dropout (Blundell et al., 2015) are alternative methods for approximating Bayesian Inference on neural networks.

Neural network ensembling is a method that predicts out-of-distribution data well in practice, requires very little fine-tuning and, under the right regularization, approximates samples from the Bayesian posterior Pearce et al. (2020); He et al. (2020). However, these methods have one main drawback, namely that each new sample from the posterior requires training a new network from scratch. Our method, on the other hand, seeks to construct a generative posterior model using similar regularization techniques to Pearce et al. (2020), allowing for quick sampling from the posterior. Epistemic Neural Networks Osband et al. (2021) are a concurrent work to ours that, while not motivated by the Bayesian Inference problem, arrived at a very similar technique to ours. The key difference between our method and Epistemic NNs is in the regularization: our method requires unlabeled data from the test-distribution for our regularization, but results in improved posterior sampling.

B Algorithm details

B.1 Low-Dimensional Embedding

As mentioned in the main paper, optimizing the generative function g over the the space of all anchor parameters $\boldsymbol{\theta}_{\text{anc}}$ is impractical, as this space is too high dimensional. Instead, we seek to learn a low-dimensional representation of the anchor parameters. There are two key reasons why we would expect to be able to decrease the representational power of the anchor distribution and maintain the same of fidelity in the posterior estimation. (1) Because of the nature of neural networks, there are many settings of parameters $\boldsymbol{\theta}$ that would result in the same output vector $\hat{\mathbf{Y}}$. Because we are focusing on the posterior of $\hat{\mathbf{Y}}$, we need less representational power to model $\hat{\mathbf{Y}}$. (2) And, maybe more importantly, because the posterior parameters are highly correlated, we would expect to need less expressive power to represent the posterior distribution than the prior distribution.

Thus, we would like to construct a simpler estimate of the prior space using a low-dimensional embedding vector, $\mathbf{z}$. That is, we want our generative model to take as input the embedding vector $\mathbf{z} \sim \mathcal{N}(0, I)$, instead of $\boldsymbol{\theta}_{\text{anc}}$, in order to estimate the MAP function. To do this, we need to learn a mapping from anchor parameters $\boldsymbol{\theta}_{\text{anc}}$ to embedding vectors $\mathbf{z}$. For our experiments, we used a 1-1 embedding scheme where we sample k parameters from the true prior $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k \sim P_{\text{prior}}(\boldsymbol{\theta})$ and k independent samples from our embedding $\mathbf{z}_1, \ldots, \mathbf{z}_k$. We then jointly optimize over $\boldsymbol{\phi}$ and $\mathbf{z}_1, \ldots, \mathbf{z}_k$ as follows:

$$\underset{\boldsymbol{\phi}, \mathbf{z}_1, \ldots, \mathbf{z}_k}{\arg\min}\ \mathbb{E}_{\mathbf{x}_{\text{sample}}} \sum_{j=1}^{k} \sum_{i=1}^{N} \|y^i_{\text{obs}} - g(x^i_{\text{obs}}, \mathbf{z}_j + \boldsymbol{\epsilon}; \boldsymbol{\phi})\|_2^2 + \beta \|\delta\|_2^2 + \mathcal{L}_{\text{reg}}(\mathbf{z}_1, \ldots, \mathbf{z}_k), \tag{4}$$

where $\delta_j = g(x^j_{\text{sample}}, \mathbf{z}_j + \boldsymbol{\epsilon}; \boldsymbol{\phi}) - f(x^j_{\text{sample}}; \boldsymbol{\theta}_j)$, $x^j_{\text{sample}} \sim P_{\text{sample}}(x)$, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ is a noise injection vector that improves the smoothness of interpolations between embedding vectors, and $\mathcal{L}_{\text{reg}}(\mathbf{z}_1, \ldots, \mathbf{z}_k)$ is a regularizer to keep $\mathbf{z}_1, \ldots, \mathbf{z}_k$ roughly normally distributed, which allows us to easily sample from the embedding space. For our experiments we use the KL divergence between $\mathbf{z}$

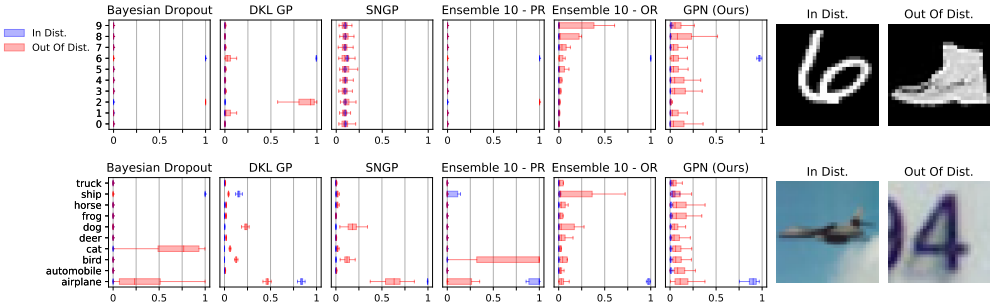
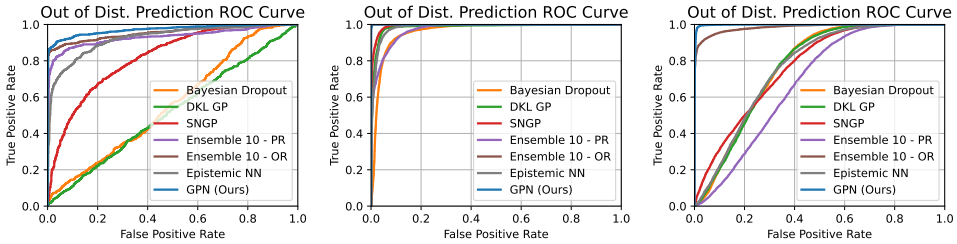


Figure 4: Boxplots of 100 posterior samples from every method for 2 test images, one from the In Distribution dataset and one from the OOD dataset.



(a) Regression - Superconductor (b) Classification - MNIST (c) Classification - CIFAR-10

Figure 5: ROC curves for out of distribution prediction based on sample variance from 100 samples.

and the normal distribution, $\mathcal{L}_{\text{reg}}(\mathbf{z}_1, \ldots, \mathbf{z}_k) = D_{\text{KL}}(\mathcal{N}(\bar{\mathbf{z}}, s_{\mathbf{z}}), \mathcal{N}(0, 1))$. Figure 1 illustrates how we can sample from this embedding space to construct posterior functions.

B.2 Classification

Our method uses the assumption that the prior over $\hat{\mathbf{Y}}$ is approximately normally distributed. Of course, this approximation is only exact when either the network is linear or the final layer is infinitely wide and does not have an activation function (assuming the prior parameters are normally distributed). Thus, in classification tasks, where it is beneficial to a soft-max to the final output, this assumption is violated. To get around this, we add the anchor loss to the pre-softmax outputs for classification tasks.

C Additional Experiment Results

Figure 4 illustrates our methods ability to form credible posterior distributions. Specifically, the figure shows two informative test images, one from the In Distribution dataset and one from the OOD dataset, for both MNIST/Fashion MNIST and CIFAR-10/SVHN, along with box plots of sampled PMFs from each of the learned models. While all methods predict the correct label with high precision on the in distribution example, ours has a uniquely wide sample distribution on the OOD example, illustrating high predicted epistemic uncertainty.

Figures 5 show the full ROC curves for the OOD prediction tasks.

D Experiment Details

Small Scale Regression Our first experiment is a small 1-dimensional regression problem where we can easily compute the ground truth. We provide each method with the same 6 observations. We compute 100 samples from the approximate posterior for each method. For ground truth, we use the Metropolis-Hastings MCMC algorithm.

Table 3: MNIST results with Out of Distribution (OOD) evaluation on Fashion MNIST.

Methods	In Dist. Accuracy	In Dist. Entropy	OOD Entropy	OOD-detect. AUC
Dropout	99.0%	0.018	0.581	0.96
DKL GP	98.9%	0.091	1.223	0.99
SNGP	98.5%	2.300	2.300	0.99
Ensemble - PR	98.1%	0.062	0.952	0.97
Ensemble - OR	99.1%	0.085	2.066	**1.00**
Epistemic NN	98.8%	0.024	0.833	0.99
GPN (Ours)	99.1%	0.272	2.076	**1.00**

Table 4: Training and test datasets used for each experiment task.

Task	Training		Testing	
	Labeled	Unlabeled	In Dist.	OOD
Small Scale	6 points	$\mathcal{U}[-2, 2]$	N/A	$\mathcal{U}[-2, 2]$
Superconductor	train set where $y_i \in [13.9, +\infty)$	full train set	test set where $y_i \in [13.9, +\infty)$	test set where $y_i \in [0, 13.9)$
MNIST	MNIST train	50% MNIST train 50% F-MNIST train	MNIST test	F-MNIST test
CIFAR-10	CIFAR-10 train	50% CIFAR-10 train 50% SVHN train	CIFAR-10 test	SVHN test

High-Dimensional Regression For a high-dimensional regression task, we used a Superconductivity prediction dataset (Hamidieh, 2018), the goal of which is to predict the critical temperature of different superconductive materials based on 81 features. For the In Distribution and OOD datasets, we split the full dataset based on the target values. Specifically, the In Distribution and OOD datasets contained data with a target values in the intervals $[13.9, +\infty)$ and $[0, 13.9)$, which is roughly 58% of the data in distribution and 42% out of distribution. The unlabeled dataset we provide our method and the Output-Regularized ensemble method is the full training set. Regularization hyper-parameters were chosen by running each method with 5 different settings, then choosing the model with both a low validation loss and high CI-correct on a validation set.

Classification We run two different classification experiments: for one, we use the MNIST as our In Distribution dataset and Fashion MNIST as the OOD dataset, and for the other, we use the CIFAR-10 dataset as our In Distribution dataset and SVHN as the OOD dataset. The unlabeled dataset we provide our method and the Output-Regularized ensemble method consists of unlabeled data 50% from the In Distribution and 50% from the OOD datasets. As in the regression experiments, we ran each method with 5 different regularization parameters, then chose the model with both a high validation accuracy and OOD prediction performance on a validation set.

D.1 Baselines

We compare our method against 4 competing Bayesian methods: Bayesian Dropout (Gal & Ghahramani, 2016), GPs, and anchor-regularized neural-network ensembles (Pearce et al., 2020) with 10 ensemble members. For the high-dimensional problems, we need to use approximate GPs. Specifically, we use a grid-interpolated GP with Deep Kernel Learning (DKL) (Wilson et al., 2016) implemented with the GPyTorch library (Gardner et al., 2018) and the Spectral-normalized Neural Gaussian Process (SNGP) method (Liu et al., 2020). For the ensembles, since every member requires a unique set of parameters (and anchor points) the number of ensemble members was limited by the memory capacity of our GPUs.

As described in Equation 3, our method a method to sample unlabeled data points. In low-dimensional problems, this can simply be a uniform sample from the interval of interest, for example $[-2, 2]$ in

Figure 2. However, in high-dimensional problems, this requires access to an additional dataset of unlabeled training data. From a practical perspective, this is a reasonable setup as many real-world datasets have far more unlabeled data than labeled data.

This additional data obviously favors our method over the baseline methods as our method is the only method that is able to take advantage of this unlabeled data. To address this, we added an additional ensemble regularized using the same regularization as Equation 3; in other words, we regularize the outputs of the ensemble members towards outputs of sampled prior networks. We denote this new method as an "Output-regularized (OR)" Ensemble to distinguish it from the "Parameter-regularized" (PR) Ensemble. We show that despite having the access to the same additional unlabeled data, our method outperforms and scales better than this OR Ensemble method.

Each model uses the same 4-layer (Superconductor), 5-layer (MNIST), or 7-layer (CIFAR) architecture, except the ensembles, which use slightly smaller networks for each ensemble member (to fit on a single GPU). Full experimental details are in Appendix D.

D.2 Metrics

Because we are interested in epistemic uncertainty estimation, we cannot use common aleatoric uncertainty evaluation metrics, such as Expected Calibration Error (ECE) (Guo et al., 2017). Instead, we evaluate each methods ability to construct useful posterior estimates using the following metrics.

OOD Detection One desired property of an epistemic uncertainty estimators is in predicting when a data point is outside the training distribution and, thus, any prediction made on such a data point will likely be incorrect. For this reason, OOD detection is a common metric used in epistemic uncertainty literature (Liu et al., 2020; Van Amersfoort et al., 2020). For an ideal posterior distribution, we expect high sample variance on OOD data and low sample variance from In Distribution data. Thus, in our experiments, we use posterior sample variance to detect OOD data. Specifically, for each model, we use posterior variance over 100 samples from each model as a score to construct ROC curves and measure the measure the area under the curve (AUC).

Confidence Intervals / Entropy Another desired property of an epistemic uncertainty estimator for a regression task is in constructing confidence intervals. We would expect such confidence intervals (CIs) to contain the true class with high probability. We also expect that, on in distribution examples, confidence intervals are narrow. To construct these intervals, we take 100 samples from each model on every data point in the test data and compute the range of the middle 95% of the samples. *CI-correct* refers to the percentage of true labels that fall in this interval. Note that, unlike the other methods tested, ensembles are not a generative model. Thus, when trying to sample from the ensemble method, we can only sample as many functions as there are members of the ensemble; for our experiments this number is 10.

For classification, CIs are not as informative. Instead, we look at the posterior sample entropy. For an ideal posterior distribution, we expect entropy to be low for data inside the training set (data the model should have confidence on) and high for data outside the training set (which we expect the model to be uncertain on). To measure this sample entropy, we take 100 samples from each model on every data point and take the average of the post-softmax outputs. We then average this sample entropy for both the In Distribution and OOD datasets.

Scalability One key benefit of a generative model over an ensemble is efficiency: every sample from an ensemble method requires learning an entire new network from scratch. Generative models, on the other hand, are able to produce new samples with a single pass through the network. To illustrate the benefit of this efficiency in practice, we ran an experiment to measure OOD detection AUC vs. computation time for both the Superconductor and CIFAR-10 tasks. We trained 2 (Superconductor) and 5 (CIFAR) ensemble members at a time and computed the OOD-detection performance of the cumulative combined ensembles (both parameter-regularized and output-regularized). Each method is trained using an Nvidia 1080TI. Figure 3 shows our method is able to achieve high out of distribution prediction performance in significantly less computation time.

Table 5: GPN Hyper-parameters used for our experiments.

Hyper-parameter	Value
# of sampled embeddings (k)	100
Embedding dimension	10
Regularization coefficient (β)	0.1
Bootstrap network ($\boldsymbol{\theta}$)	2-layer linear model
Prior variance ($\boldsymbol{\Sigma}_{\text{prior}}$)	40 (layer 1), 10 (layer 2)
Bootstrap activation	Tanh

D.3 Experiment datasets

Table 4 details the labeled and unlabeled data provided to the agent at training time and the In Distribution and OOD data used for evaluation at test time.

D.4 Network Architectures

For the superconductor regression problem, we used a 4-layer network with 4 128-unit wide linear layers for the ensemble methods and a 4-layer network with 4 512-unit wide linear layers for the other methods. For the MNIST classification problem, we used a 5-layer network with 2 convolution layers and 3 128-unit wide linear layers for our the ensemble methods and a 5-layer network with 2 convolution layers and 3 512-unit wide linear layers for our the other methods. For the CIFAR classification problem, we used a 7-layer network with 3 convolution layers and 4 256-unit wide linear layers for our the ensemble methods and a 7-layer network with 3 convolution layers and 4 512-unit wide linear layers for our the other methods.

D.5 Hyper parameters

The non-network architecture hyper-parameters we used for GPN were roughly consistent across all experiments, except for the regularization coefficient, β. As stated in our experiments section, we performed a small search over 5 values of β for each experiment. The full set of hyper-parameters are shown in Table 5

E Proofs

E.1 Proof of Theorem 2.1

Proof. We will start by showing that $P(\mathbf{y}_{\text{obs}}|\hat{\mathbf{Y}})$ is normally distributed.

Let $\hat{Y}_i' = f(x^i_{\text{obs}}; \boldsymbol{\theta})$ be a transformation of the random variable $\boldsymbol{\theta}$ corresponding to the outputs of the function f parameterized by $\boldsymbol{\theta}$ evaluated at observation points $\mathbf{x}_{\text{obs}}$. Note that by definition of $\mathbf{y}_{\text{obs}}$, for each $i \in 1, \ldots, N$:

$$y^i_{\text{obs}} = f(x^i_{\text{obs}}; \boldsymbol{\theta}) + \epsilon_i = \hat{Y}_i' + \epsilon_i$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon)$.

By assumption: $[\hat{\mathbf{Y}}, \hat{\mathbf{Y}}']^T$ is normally distributed. Since $\boldsymbol{\epsilon}$ is normally distributed and independent of $[\hat{\mathbf{Y}}, \hat{\mathbf{Y}}']^T$, then the joint distribution $[\hat{\mathbf{Y}}, \hat{\mathbf{Y}}', \boldsymbol{\epsilon}]^T$ is also normally distributed.

Now, by applying a linear transformation, we obtain the joint variable $\begin{bmatrix} \hat{\mathbf{Y}} \\ \hat{\mathbf{Y}}' \\ \mathbf{y}_{\text{obs}} \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & I & I \end{bmatrix} \begin{bmatrix} \hat{\mathbf{Y}} \\ \hat{\mathbf{Y}}' \\ \boldsymbol{\epsilon} \end{bmatrix}$, which is also normally distributed (as it is a linear transformation of a Gaussian random variable).

Now, since the conditional and marginal distributions of a Gaussian distribution are also Gaussian, $P(\mathbf{y}_{\text{obs}}, \hat{\mathbf{Y}}'|\hat{\mathbf{Y}})$ and $P(\mathbf{y}_{\text{obs}}|\hat{\mathbf{Y}})$ are also Gaussian.

Now, using Bayes' rule, we can show $P(\hat{\mathbf{Y}}|\mathbf{y}_{\text{obs}})$ is also normally distributed.

$$\begin{aligned} P(\hat{\mathbf{Y}} \mid \mathbf{y}_{\text{obs}}) &\propto P(\mathbf{y}_{\text{obs}} \mid \hat{\mathbf{Y}}) P(\hat{\mathbf{Y}}) \\ &= \mathcal{N}(\hat{\mathbf{Y}}|\boldsymbol{\mu}_{\text{like}}, \boldsymbol{\Sigma}_{\text{like}}) \mathcal{N}(\hat{\mathbf{Y}}|\boldsymbol{\mu}_{\hat{\mathbf{Y}}}, \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}) \end{aligned}$$

for some mean and covariance $\boldsymbol{\mu}_{\text{like}}, \boldsymbol{\Sigma}_{\text{like}}$. Since the product of two multivariate Gaussians is a Gaussian, we know that:

$$P(\hat{\mathbf{Y}} \mid \mathbf{y}_{\text{obs}}) = \mathcal{N}(\boldsymbol{\mu}_{\text{post}}, \boldsymbol{\Sigma}_{\text{post}}) \tag{5}$$

where

$$\boldsymbol{\Sigma}_{\text{post}} = \left(\boldsymbol{\Sigma}_{\text{like}}^{-1} + \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1}\right)^{-1}, \quad \boldsymbol{\mu}_{\text{post}} = \boldsymbol{\Sigma}_{\text{post}} \boldsymbol{\Sigma}_{\text{like}}^{-1} \boldsymbol{\mu}_{\text{like}} + \boldsymbol{\Sigma}_{\text{post}} \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1} \mu_{\hat{\mathbf{Y}}}. \tag{6}$$

Now we consider at the distribution $P(\hat{Y}_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}}))$ where $\hat{\mathbf{Y}}_{\text{anc}} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{anc}}, \boldsymbol{\Sigma}_{\text{anc}})$. We will show that, when we set

$$\boldsymbol{\mu}_{\text{anc}} = \boldsymbol{\mu}_{\hat{\mathbf{Y}}}, \quad \boldsymbol{\Sigma}_{\text{anc}} = \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}} + \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}} \boldsymbol{\Sigma}_{\text{like}}^{-1} \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}$$

the distribution $P(\hat{Y}_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}}))$ becomes equal to the posterior distribution $\mathcal{N}(\boldsymbol{\mu}_{\text{post}}, \boldsymbol{\Sigma}_{\text{post}})$. There are three steps needed to show equality between these distributions:

1. Show that $P(\hat{Y}_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}}))$ is normally distributed for some mean and variance, $\boldsymbol{\mu}_{\text{post}}^{\text{RMS}}, \boldsymbol{\Sigma}_{\text{post}}^{\text{RMS}}$.
2. Show that $\boldsymbol{\mu}_{\text{post}}^{\text{RMS}} = \boldsymbol{\mu}_{\text{post}}$.
3. Show that $\boldsymbol{\Sigma}_{\text{post}}^{\text{RMS}} = \boldsymbol{\Sigma}_{\text{post}}$.

Using the same reasoning as above, in the limit as $M \to \infty$,

$$\begin{aligned} P(\hat{Y}_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}})) &= \arg\max_{\hat{\mathbf{Y}}} P(\mathbf{y}_{\text{obs}}|\hat{\mathbf{Y}}) P_{\text{anc}}(\hat{\mathbf{Y}}) \\ &= \arg\max_{\hat{\mathbf{Y}}} \mathcal{N}(\hat{\mathbf{Y}}|\boldsymbol{\mu}_{\text{like}}, \boldsymbol{\Sigma}_{\text{like}}) \mathcal{N}(\hat{\mathbf{Y}}|\boldsymbol{\mu}_{\text{anc}}, \boldsymbol{\Sigma}_{\text{anc}}) \end{aligned}$$

Since the max of a Gaussian is the mean, then

$$F_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}}) = A\hat{\mathbf{Y}}_{\text{anc}} + b \tag{7}$$

where we define:

$$A = \boldsymbol{\Sigma}_{\text{post}} \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1} \tag{8}$$

$$b = \boldsymbol{\Sigma}_{\text{post}} \boldsymbol{\Sigma}_{\text{like}}^{-1} \boldsymbol{\mu}_{\text{like}} \tag{9}$$

We will now show that $\mathbb{E}[\hat{Y}_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}})] = \boldsymbol{\mu}_{\text{post}}$. Because we set $\boldsymbol{\mu}_{\text{anc}} = \boldsymbol{\mu}_{\hat{\mathbf{Y}}}$:

$$\begin{aligned} \mathbb{E}[\hat{Y}_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}})] &= \mathbb{E}[A\hat{\mathbf{Y}}_{\text{anc}} + b] \\ &= A\mathbb{E}[\hat{\mathbf{Y}}_{\text{anc}}] + b \\ &= A\boldsymbol{\mu}_{\hat{\mathbf{Y}}} + b \\ &= \boldsymbol{\Sigma}_{\text{post}} \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1} \boldsymbol{\mu}_{\hat{\mathbf{Y}}} + \boldsymbol{\Sigma}_{\text{post}} \boldsymbol{\Sigma}_{\text{like}}^{-1} \boldsymbol{\mu}_{\text{like}} \\ &= \boldsymbol{\mu}_{\text{post}} \end{aligned}$$

Finally, we will show that $\mathbb{V}\text{ar}[\hat{Y}_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}})] = \boldsymbol{\Sigma}_{\text{post}}$.

$$\begin{aligned} \mathbb{V}\text{ar}[\hat{Y}_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}})] &= \mathbb{V}\text{ar}[A\hat{\mathbf{Y}}_{\text{anc}} + b] \\ &= A\mathbb{V}\text{ar}[\hat{\mathbf{Y}}_{\text{anc}}]A^T \\ &= (\boldsymbol{\Sigma}_{\text{post}} \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1})(\boldsymbol{\Sigma}_{\hat{\mathbf{Y}}} + \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}} \boldsymbol{\Sigma}_{\text{like}}^{-1} \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}})(\boldsymbol{\Sigma}_{\text{post}} \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1})^T \\ &= (\boldsymbol{\Sigma}_{\text{post}} + \boldsymbol{\Sigma}_{\text{post}} \boldsymbol{\Sigma}_{\text{like}}^{-1} \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}})(\boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1} \boldsymbol{\Sigma}_{\text{post}}) \\ &= \boldsymbol{\Sigma}_{\text{post}} \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1} \boldsymbol{\Sigma}_{\text{post}} + \boldsymbol{\Sigma}_{\text{post}} \boldsymbol{\Sigma}_{\text{like}}^{-1} \boldsymbol{\Sigma}_{\text{post}} \\ &= \boldsymbol{\Sigma}_{\text{post}} (\boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1} + \boldsymbol{\Sigma}_{\text{like}}^{-1}) \boldsymbol{\Sigma}_{\text{post}} \\ &= \boldsymbol{\Sigma}_{\text{post}} \end{aligned}$$

So, $P(\hat{Y}_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}})) \sim \mathcal{N}(\boldsymbol{\mu}_{\text{post}}^{\text{RMS}}, \boldsymbol{\Sigma}_{\text{post}}^{\text{RMS}})$ where $\boldsymbol{\mu}_{\text{post}}^{\text{RMS}} = \boldsymbol{\mu}_{\text{post}}$ and $\boldsymbol{\Sigma}_{\text{post}}^{\text{RMS}} = \boldsymbol{\Sigma}_{\text{post}}$.

Thus, in the limit as $M \to \infty$, $P(\hat{Y}_{\text{MAP}}(\hat{\mathbf{Y}}_{\text{anc}})) = P(\hat{\mathbf{Y}}|\mathbf{y}_{\text{obs}})$.

□

E.2 Derivation of loss function

We start with

$$\boldsymbol{\phi} = \arg\max_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{\theta}_{\text{anc}} \sim P_{\text{prior}}(\boldsymbol{\theta}_{\text{anc}})} \log P_{\text{like}}(\mathbf{y}_{\text{obs}} \mid g(\mathbf{x}_{\text{sample}}, \boldsymbol{\theta}_{\text{anc}}; \boldsymbol{\phi})) + \log P_{\text{anc}}(g(\mathbf{x}_{\text{sample}}, \boldsymbol{\theta}_{\text{anc}}; \boldsymbol{\phi}))$$

If we choose $\boldsymbol{\mu}_{\text{anc}} = \boldsymbol{\mu}_{\hat{\mathbf{Y}}}$ and $\boldsymbol{\Sigma}_{\text{anc}} = \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}$, we can simplify the logarithm of the likelihood and anchor distributions as follows:

$$\begin{aligned}
\log P_{\text{like}}(\mathbf{y}_{\text{obs}} \mid g(\mathbf{x}_{\text{sample}}, \boldsymbol{\theta}_{\text{anc}}; \boldsymbol{\phi})) &= \sum_i \log \mathcal{N}(y_{\text{obs}}^i | g(x_{\text{sample}}^i, \boldsymbol{\theta}_{\text{anc}}; \boldsymbol{\phi}), \sigma_\epsilon) \\
&= -\frac{1}{2\sigma_\epsilon^2} \sum_i \|y_{\text{obs}}^i - g(x_{\text{sample}}^i, \boldsymbol{\theta}_{\text{anc}}; \boldsymbol{\phi})\|_2^2
\end{aligned}$$

$$\begin{aligned}
\log P_{\text{anc}}(g(\mathbf{x}_{\text{sample}}, \boldsymbol{\theta}_{\text{anc}}; \boldsymbol{\phi})) &= \log \mathcal{N}(g(\mathbf{x}_{\text{sample}}, \boldsymbol{\theta}_{\text{anc}}; \boldsymbol{\phi}) | \boldsymbol{\mu}_{\hat{\mathbf{Y}}}, \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}) \\
&= -\frac{1}{2} \boldsymbol{\delta}^T \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1} \boldsymbol{\delta}
\end{aligned}$$

where $\delta_j = g(x_{\text{sample}}^j, \boldsymbol{\theta}_{\text{anc}}; \boldsymbol{\phi}) - f(x_{\text{sample}}^j; \boldsymbol{\theta}_{\text{anc}})$.

Putting these together, we get:

$$\begin{aligned}
\boldsymbol{\phi} &= \arg\max_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{\theta}_{\text{anc}} \sim P_{\text{prior}}(\boldsymbol{\theta}_{\text{anc}})} - \frac{1}{2\sigma_\epsilon^2} \sum_i \|y_{\text{obs}}^i - g(x_{\text{sample}}^i, \boldsymbol{\theta}_{\text{anc}}; \boldsymbol{\phi})\|_2^2 - \frac{1}{2} \boldsymbol{\delta}^T \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1} \boldsymbol{\delta} \\
&= \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{\theta}_{\text{anc}} \sim P_{\text{prior}}(\boldsymbol{\theta}_{\text{anc}})} \frac{1}{N} \sum_i \|y_{\text{obs}}^i - g(x_{\text{sample}}^i, \boldsymbol{\theta}_{\text{anc}}; \boldsymbol{\phi})\|_2^2 + \frac{1}{N} \sigma_\epsilon^2 \boldsymbol{\delta}^T \boldsymbol{\Sigma}_{\hat{\mathbf{Y}}}^{-1} \boldsymbol{\delta}
\end{aligned}$$