D.1. Where does the performance advantage of different models come from?

The results in Figure 2 illustrate that Multi-layer Perceptrons (MLPs) and MLP ensembles trained with SGD appear to have subpar performance compared to other methods, especially GPs, in terms of the joint log-likelihood. This might be somewhat surprising, especially given the model mismatch of the true function for the GP models. We might expect that the MLP models should eventually outperform GPs if the training size was large enough. In this subsection, we break down the performance metric in Figure 2 to try and gauge what's going on.

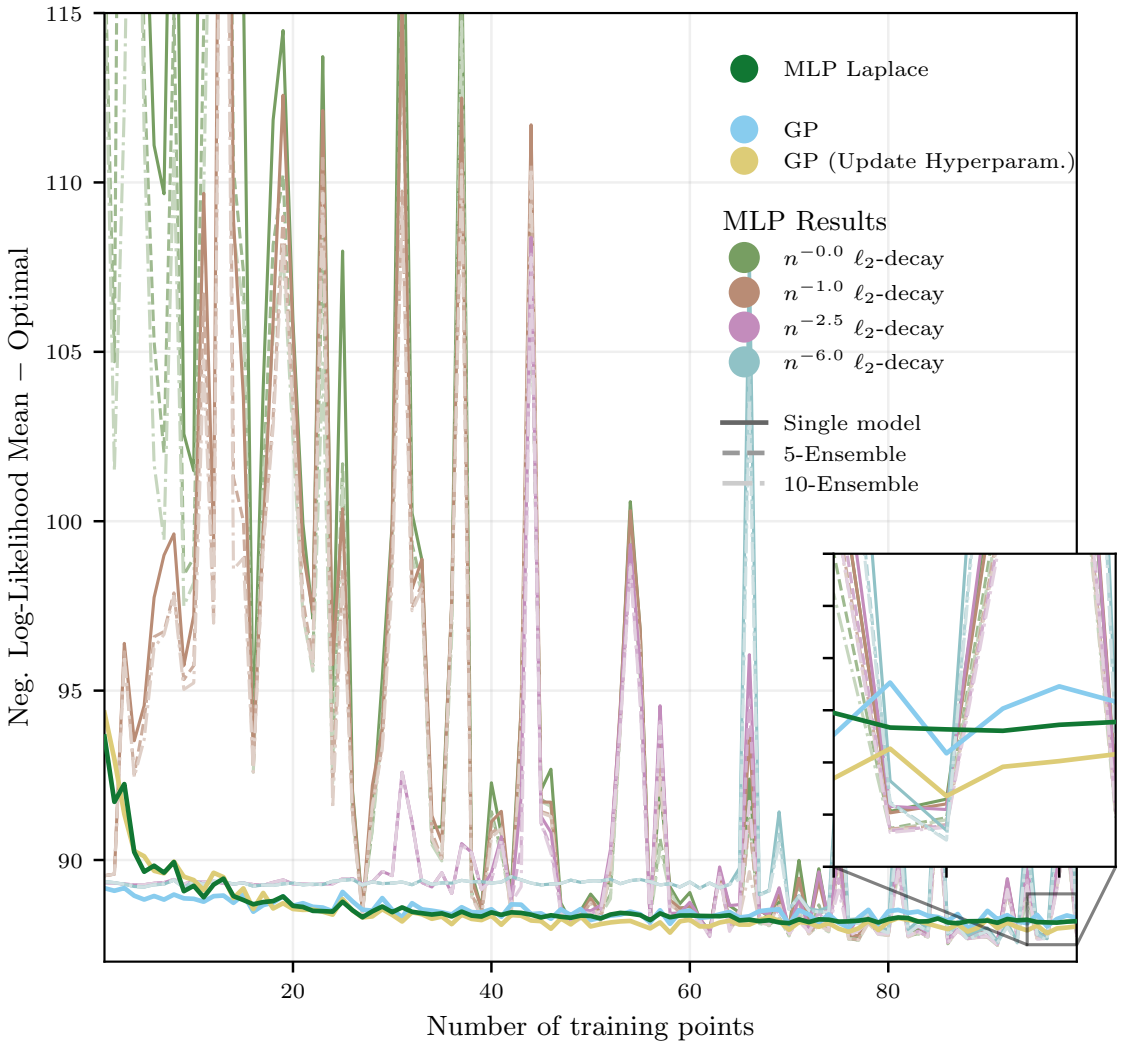
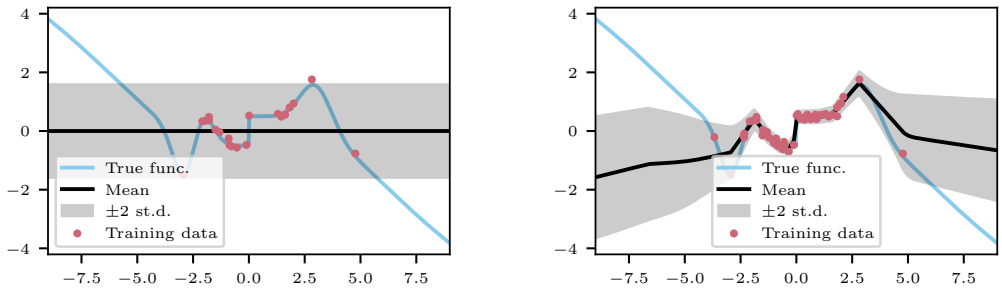


Figure 8: Comparison of the average "test" log-likelihoods of the next datapoint in a sequence for the different models with different training data sizes (number of conditioning datapoints).

In Figure 8, we plot the average log-likelihoods for the different methods broken down by the training set size. The final 'joint' log-likelihood metric in Figure 2 will be the sum of the per-training-size log-likelihoods. The figure illustrates that the SGD-trained MLP models can overfit dramatically when trained with very few training datapoints (< 40). This seems to be the biggest contributor to their subpar aggregate performance. Surprisingly, for the standard ℓ_2-decay of n^{-1} (with n being the training data size), the performance doesn't seem to monotonically improve, and is actually the worst after observing around 15 to 20 datapoints.

Increasing the regularisation strength through ℓ_2-decay seems to have the effect of averting the worst of the overfitting in MLPs in the low-data regime by essentially forcing the network to ignore the training data until a certain training dataset size is reached. For example, predictions from an MLP with $n^{-2.5}$ ℓ_2-decay are shown in Figure 9.

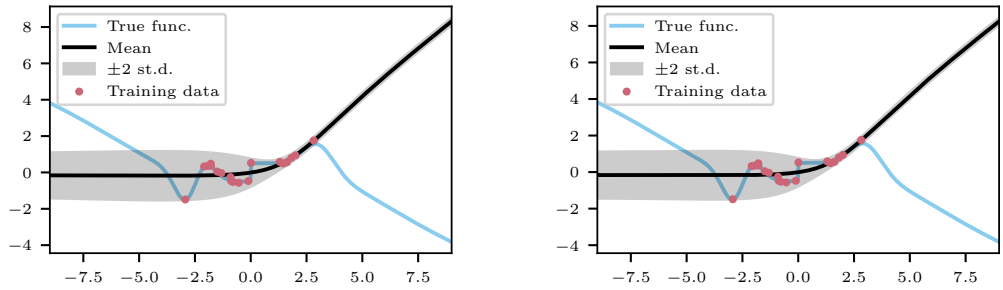


(*a*) Dataset size=20 (*b*) Dataset size=50

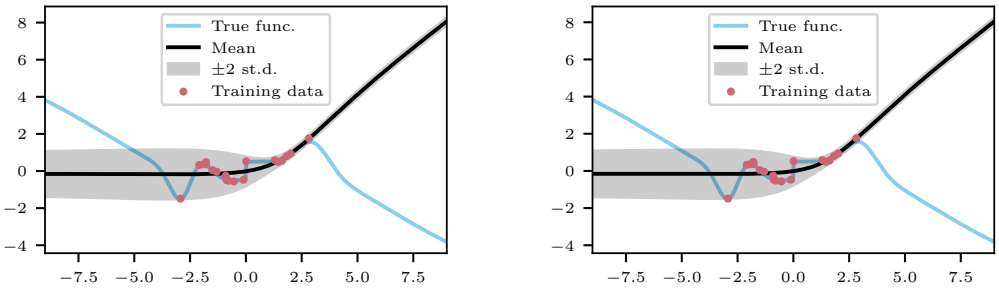
Figure 9: MLP trained with SGD on the discontinuous regression task with ℓ_2-decay of $n^{-2.5}$ for different training data sizes.

Ensembling the MLP predictions seems to help surprisingly little with performance in the low-data regime. This appears to be because, for some training data configurations, the MLPs trained with different seeds overfit in the same way. An example of this is shown in Figure 10.

It appears that both MLPs fit with the Laplace approximation and a GP with kernel hyperparameters updated throughout training eventually outperform the GP with fixed hyperparameters. This is at the cost of the initially subpar performance in the low-data regime (< 20 datapoints), where updating the kernel hyperparameters in the GP might lead to overfitting.



(a) Seed=0 (b) Seed=1



(c) Seed=2 (d) Seed=3

Figure 10: MLP trained with SGD on the same 19 datapoints on the discontinuous regression task with different seeds (different initialisation). The model consistently fits the data in the same manner, regardless of initialisation, extrapolating the data unfavourably in the interval $(-2.5, \infty)$.

Appendix E. Can a prediction rule at a fixed step be extended to an implicitely Bayesian prediction rule?

In deep learning, we would often taylor our approach to the amount of data available at hand. For example, we might use a small neural network for a small dataset, and a large neural network for a large dataset. If we have fewer than hundreds of datapoints, we might not even consider using a neural network at all. Hence, viewing training of a neural network as a prediction rule for all possible data sizes might not be the most natural lens.

One might wonder, assuming that one has only specified a prediction rule at step n only, can it be extended to a prediction rule for all n to satisfy some of the previously mentioned properties?

One reasonable guess might be that, if the prediction rule at step n is invariant to the ordering of the data, i.e. $s_n(\cdot|x_1, \ldots, x_n) = s_n(\cdot|x_{\pi(1)}, \ldots, x_{\pi(n)})$ for any permutation π of $\{1, \ldots, n\}$[9], then maybe it can be extended to an exchageable prediction rule. If that was the case, that would be good news: as long as we're making the prediction at step n only, we can claim that our prediction rule is implicitly Bayesian; once we start making predictions at other time-steps, we just need to figure out what an implicitely Bayesian extension is.

9. This is a weaker condition than exchangeability, as it only requires invariance to the ordering of the conditioning sequence.