Evaluating High-Order Predictive Distributions in Deep Learning (Supplementary Material)

Ian Osband[1] **Zheng Wen**[1] **Seyed Mohammad Asghari**[1] **Vikranth Dwaracherla**[1] **Xiuyuan Lu**[1]
Benjamin Van Roy[1]

[1]DeepMind

A EVALUATING PREDICTIVE DISTRIBUTIONS

This section contains supplementary material for Section 2. Importantly, we provide the proof for Proposition 1and discuss why dyadic sampling is sufficient for Gaussian process.

A.1 PROOF FOR PROPOSITION 1

Proposition 1 (Small τ approximately marginal)**.** *If the agent defined above is applied to Example 1 with $\tau \ll M$,*

$$\mathbf{d}_{\mathrm{KL}}^{\tau} = \overline{\mathbf{d}}_{\mathrm{KL}}^{\tau} + O\left(\tau^3/M\right).$$

Proof. Note that by definition, $\mathbf{d}_{\mathrm{KL}}^{\tau} \geq \overline{\mathbf{d}}_{\mathrm{KL}}^{\tau}$. We now prove that $\mathbf{d}_{\mathrm{KL}}^{\tau} \leq \overline{\mathbf{d}}_{\mathrm{KL}}^{\tau} + O\left(\tau^3/M\right)$. Note that

$$\mathbf{d}_{\mathrm{KL}}^{\tau} = \mathbb{E}\left[\log\left(\mathbb{P}(Y_{1:\tau}|\mathcal{E}, X_{0:\tau-1})\right)\right] - \tau \log\left(\frac{1}{2}\right),$$

where $\tau \log\left(\frac{1}{2}\right)$ is the log-likelihood under the uniform agent, and

$$\overline{\mathbf{d}}_{\mathrm{KL}}^{\tau} = \mathbb{E}\left[\log\left(\mathbb{P}(Y_{1:\tau}|\mathcal{E}, X_{0:\tau-1})\right)\right] - \mathbb{E}\left[\log\left(\mathbb{P}(Y_{1:\tau}|X_{0:\tau-1})\right)\right].$$

Consequently, we have

$$\mathbf{d}_{\mathrm{KL}}^{\tau} - \overline{\mathbf{d}}_{\mathrm{KL}}^{\tau} = \tau \log(2) + \mathbb{E}\left[\log\left(\mathbb{P}(Y_{1:\tau}|X_{0:\tau-1})\right)\right].$$

We define the event $\mathcal{G}$ as

$$\mathcal{G} = \{\text{there are no repeated inputs in } X_{0:\tau-1}\}.$$

One key observation is that conditioning on $\mathcal{G}$, the posterior predictive distribution is i.i.d. across inputs, and

$$\log\left(\mathbb{P}(Y_{1:\tau}|X_{0:\tau-1})\right) = -\tau \log(2)$$

conditioning on $\mathcal{G}$. Hence

$$\begin{aligned}\mathbb{E}\left[\log\left(\mathbb{P}(Y_{1:\tau}|X_{0:\tau-1})\right)\right] &= -\mathbb{P}(\mathcal{G})\tau\log(2) + \mathbb{P}(\bar{\mathcal{G}})\mathbb{E}\left[\log\left(\mathbb{P}(Y_{1:\tau}|X_{0:\tau-1})\right)\middle|\bar{\mathcal{G}}\right] \\ &\leq -\mathbb{P}(\mathcal{G})\tau\log(2)\end{aligned}$$

where $\bar{\mathcal{G}}$ is the complement of $\mathcal{G}$, and the inequality follows from $\log\left(\mathbb{P}(Y_{1:\tau}|X_{0:\tau-1})\right) \leq 0$. Hence we have

$$\mathbf{d}_{\mathrm{KL}}^{\tau} - \overline{\mathbf{d}}_{\mathrm{KL}}^{\tau} \leq (1 - \mathbb{P}(\mathcal{G}))\tau\log(2) = \mathbb{P}(\bar{\mathcal{G}})\tau\log(2).$$

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

Finally, note that

$$\mathbb{P}(\mathcal{G}) = \prod_{k=1}^{\tau-1}\left(1-\frac{k}{M}\right) = 1 - \frac{1}{M}\sum_{k=1}^{\tau-1} k + O\left(\frac{1}{M^2}\right) = 1 - \frac{\tau(\tau-1)}{2M} + O\left(\frac{1}{M^2}\right).$$

Hence $\mathbb{P}(\bar{\mathcal{G}}) = O\left(\tau^2/M\right)$ and we have

$$\mathbf{d}_{\mathrm{KL}}^{\tau} - \overline{\mathbf{d}}_{\mathrm{KL}}^{\tau} \leq O\left(\tau^3/M\right).$$

The conclusion follows from

$$\overline{\mathbf{d}}_{\mathrm{KL}}^{\tau} \leq \mathbf{d}_{\mathrm{KL}}^{\tau} \leq \overline{\mathbf{d}}_{\mathrm{KL}}^{\tau} + O\left(\tau^3/M\right).$$

□

A.2 DYADIC SAMPLING AND GAUSSIAN PROCESSES

In this section, we discuss why dyadic sampling is sufficient for Gaussian processes (GPs). In particular, we show that when both the environment $\mathcal{E}$ and the imagined environment $\hat{\mathcal{E}}$ of an agent follow GP, then with sufficiently large τ and under suitable regularity conditions, performing well under $\mathbf{d}_{\mathrm{KL}}^{\tau,\kappa=2}$ is sufficient to ensure that the posterior distribution of $\mathcal{E}$ and the agent's belief over $\hat{\mathcal{E}}$ are close.

Assume that both $\mathcal{E}$ and $\hat{\mathcal{E}}$ are GPs with the same finite domain $\mathcal{X}$ and that the training input distribution is uniform over $\mathcal{X}$. Specifically, under the environment $\mathcal{E}$,

$$Y_{t+1} = f(X_t) + W_{t+1},$$

and under the imagined environment $\hat{\mathcal{E}}$,

$$\hat{Y}_{t+1} = \hat{f}(X_t) + \hat{W}_{t+1},$$

where W_{t+1}'s and $\hat{W}_{t+1}$'s are i.i.d. observation noises according to $N(0,\sigma^2)$, and f and $\hat{f}$ are functions over $\mathcal{X}$. We assume that $\mathbb{P}(f \in \cdot|\mathcal{D}_T) = N(\mu,\Sigma)$ and $\mathbb{P}(\hat{f} \in \cdot|\theta_T) = N(\hat{\mu},\hat{\Sigma})$. Note that by definition

$$\begin{aligned}
\mathbf{d}_{\mathrm{KL}}^{\tau,\kappa=2} =& \mathbb{E}\left[\mathbb{E}\left[\mathbf{d}_{\mathrm{KL}}\left(P^*_{T+1:T+\tau}\middle\|\hat{P}_{T+1:T+\tau}\right)\middle|X_{T:T+\tau-1} = \tilde{X}^{\kappa=2}_{T:T+\tau-1}\right]\right] \\
=& \underbrace{\mathbb{E}\left[\mathbb{I}\left(\mathcal{E}; Y_{T+1:T+\tau}\middle|\mathcal{D}_T, X_{T:T+\tau-1} = \tilde{X}^{\kappa=2}_{T:T+\tau-1}\right)\right]}_{\text{irreducible}} + \underbrace{\mathbb{E}\left[\mathbb{E}\left[\mathbf{d}_{\mathrm{KL}}\left(\overline{P}_{T+1:T+\tau}\middle\|\hat{P}_{T+1:T+\tau}\right)\middle|X_{T:T+\tau-1} = \tilde{X}^{\kappa=2}_{T:T+\tau-1}\right]\right]}_{\tilde{\mathbf{d}}_{\mathrm{KL}}^{\tau,\kappa=2}}.
\end{aligned}$$

Note that the first term in the above equation is irreducible and independent of the agent, hence, performing well under $\mathbf{d}_{\mathrm{KL}}^{\tau,\kappa=2}$ is equivalent to performing well under $\tilde{\mathbf{d}}_{\mathrm{KL}}^{\tau,\kappa=2}$. Under suitable regularity conditions, for sufficiently large τ, we have

$$\tilde{\mathbf{d}}_{\mathrm{KL}}^{\tau,\kappa=2} \approx \mathbb{E}\left[\mathbf{d}_{\mathrm{KL}}\left(\mathbb{P}\big(f(\tilde{X}_{1:2}) \in \cdot|\mathcal{D}_T, \tilde{X}_{1:2}\big)\middle\|\mathbb{P}\big(\hat{f}(\tilde{X}_{1:2}) \in \cdot|\theta_T, \tilde{X}_{1:2}\big)\right)\right],$$

where $\tilde{X}_{1:2} = \big(\tilde{X}_1, \tilde{X}_2\big)$ and $\tilde{X}_1$ and $\tilde{X}_2$ are i.i.d. sampled from P_X. Thus, if the RHS of the above equation is small, then it implies that

$$\mathbf{d}_{\mathrm{KL}}\left(\mathbb{P}\big(f(\tilde{X}_{1:2}) \in \cdot|\mathcal{D}_T, \tilde{X}_{1:2}\big)\middle\|\mathbb{P}\big(\hat{f}(\tilde{X}_{1:2}) \in \cdot|\theta_T, \tilde{X}_{1:2}\big)\right) \tag{1}$$

is small for all $\tilde{X}_{1:2}$. Let $\mu(\tilde{X}_{1:2}) \in \Re^2$ and $\Sigma(\tilde{X}_{1:2}) \in \Re^{2\times 2}$ respectively denote μ and Σ restricted to $\tilde{X}_{1:2}$, and $\hat{\mu}(\tilde{X}_{1:2})$ and $\hat{\Sigma}(\tilde{X}_{1:2})$ are defined similarly, then we have

$$f(\tilde{X}_{1:2}) \sim N\big(\mu(\tilde{X}_{1:2}), \Sigma(\tilde{X}_{1:2})\big) \quad \text{and} \quad \hat{f}(\tilde{X}_{1:2}) \sim N\big(\hat{\mu}(\tilde{X}_{1:2}), \hat{\Sigma}(\tilde{X}_{1:2})\big).$$

Consequently, if equation 1 is small, then $\mu(\tilde{X}_{1:2})$ is close to $\hat{\mu}(\tilde{X}_{1:2})$ and $\Sigma(\tilde{X}_{1:2})$ is close to $\hat{\Sigma}(\tilde{X}_{1:2})$. Since this holds for all $\tilde{X}_{1:2}$, this further implies that μ is close to $\hat{\mu}$ and Σ is close to $\hat{\Sigma}$. In other words, the posterior distribution of $\mathcal{E}$ and the agent's belief over $\hat{\mathcal{E}}$ are close.

B LOGISTIC REGRESSION

This appendix provides supplementary details for Section 3. We include all of the code necessary to generate Figures 1 and 2 as part of our opensource submission `github.com/deepmind/neural_testbed`. Results are averaged over 10 random seeds per problem setting.

Figure 8 provides another kind of insight to the scaling observed in Figure 1. In these plots we show the KL ratio of a perfect `prior` agent when compared to `uniform`. We can see that, for any input dimension, the empirical KL ratio decreases with τ. However, as the input dimension grows reasonably large $(D = 10)$, that even large $\tau = 10,000$ are not enough to observe this ratio under 0.5. We know that, as $\tau \to \infty$ this ratio will tend to zero for these two agents. By contrast, dyadic sampling is able to clearly distinguish these agents even for moderate values of τ.

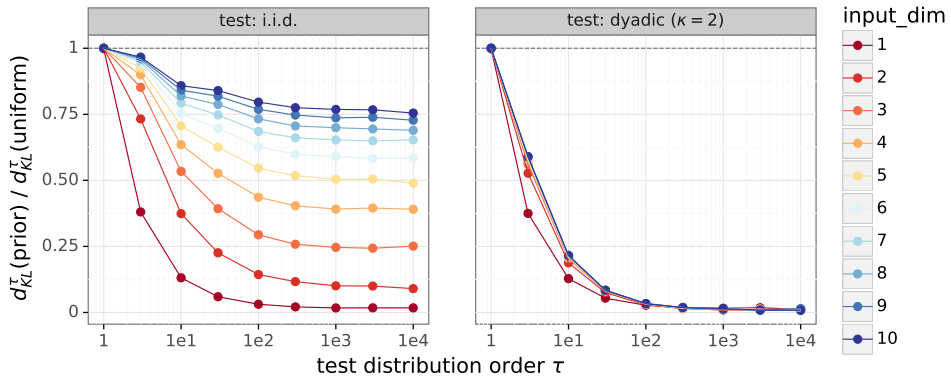

Figure 1: Global input sampling can eventually separate prior samples from uniform, but the required τ grows exponentially with input dimension. Local $\kappa = 2$ sampling can distinguish these agents without exponential τ.

Figure 9 provides some insight to the robustness of Algorithm 1 under varying number of agent samples. We make use of the *epistemic neural network* notation introduced by Osband et al. [2021]. We can see that these monte carlo estimates converge empirically as we increase the number of samples. Therefore, for the purposes of our experiments in this section our choice of $10,000$ ENN samples is sufficient.

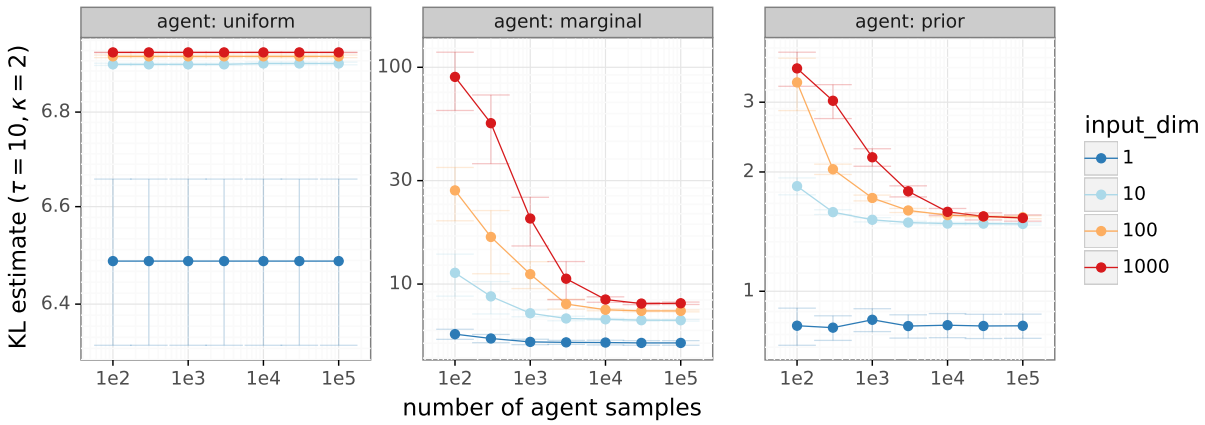

Figure 2: For the agents that we consider, 10,000 ENN samples is sufficient to get reasonable KL estimates across all input dimensions.

C NEURAL TESTBED

This appendix provides supplementary details for Section 4

C.1 PROBLEM FORMULATION

We build on the opensource code of the Neural Testbed `github.com/deepmind/neural_testbed`. Our testbed sweep is defined over input dimensions $D \in \{2, 10, 100\}$, number of training pairs $T = \lambda D$ for $\lambda \in \{1, 10, 100, 1000\}$, temperature $\rho \in \{0.01, 0.1, 0.5\}$ with 5 random seeds in each setting. We replace the $\mathbf{d}_{\mathrm{KL}}^{10}$ evaluation with dyadic sampling $\mathbf{d}_{\mathrm{KL}}^{10,\kappa=2}$. We release all of our code and implementation at `github.com/deepmind/neural_testbed`.

C.2 BENCHMARK AGENTS

We make use of the benchmark agents introduced in Osband et al. [2022] and opensourced at `github.com/deepmind/neural_testbed`. Since our testbed includes settings with number of training pairs as small as 2 (when $D = 2$, $\lambda = 1$) and as large as 100,000 (when $D = 100$, $\lambda = 1000$), in order to improve agent performance over all settings, we allow agents to adjust their number of training steps based on the problem setting. Agents implementation can be found in our open source code under the path `/agents/factories`.

We make small alterations to the tuning sweeps proposed in Osband et al. [2022] in an effort to improve agent performance in high dimension problems. This change strictly improved the agent performance as we only *added* hyperparameter choices and did not restrict them. Our sweeps can be found in our open source code under the path `/agents/factories/sweeps/testbed`, but we highlight the differences that helped to improve agent performance. For **`mlp`**, **`ensemble`**, **`dropout`**, **`bbb`**, **`hypermodel`**, **`ensemble+`** agents, we found out that their performance improves by allowing them to adjust their default number of training steps based on the problem setting: increase it by 5x when $\lambda = 1000$ and decrease it by 5x when $\lambda = 1$. For **`sgmcmc`** agent, we found out that we can improve the performance of this agent by allowing it to increase prior variance parameter by 2x when $D = 100$.

C.3 OVERALL RESULTS

Figure 3 provides an overview of the agent performance on the testbed in terms of $\mathbf{d}_{\mathrm{KL}}$. These numbers are normalized so that the baseline MLP has a value of 1. In classification problems it is common to also consider the classification accuracy, or the percentage of inputs for which the agent correctly labels the input. Figure 3 confirms that, after tuning, none of the agents perform significantly differently from baseline MLP.

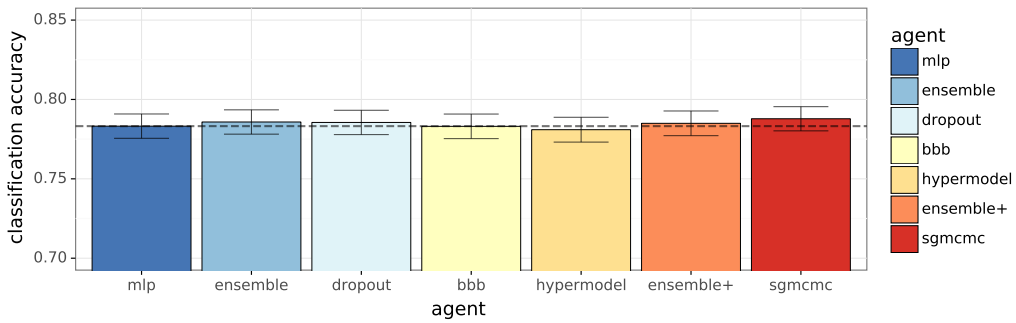

Figure 3: After tuning, none of the agents perform signficantly differently from the baseline MLP in terms of classification accuracy.

D REAL DATA

This section provides supplementary details regarding the experiments in Section **??**. As before, we include full implementation and source code in our open source code under the path `/real_data`.

D.1 PROBLEM FORMULATION

Table 2 outlines the datasets included in our experiments. For each dataset, we perform a standard preprocessing on inputs to be mean zero and unit variance. Full details are available in our open source code under the path `/real_data/utils.py`.

In the testbed we are able to evaluate a wide range of SNR regimes by varying temperature. This means that we can query a given input X_t multiple times and potentially obtain different class labels Y_t. For these fixed dataset there is only one testing dataset, with deterministic labels given for each input. We map this setting to the low temperature limit (and high SNR) setting of our testbed. As such, we evaluate the negative log-likelihood in place of $\mathbf{d}_{\mathrm{KL}}^{\tau}$. This is equivalent to assuming the underlying world model was deterministic at these testing points, and is standard practice in deep learning.

We note that this 'high SNR' assumption appears to be reasonable in practice, since for all of the datasets considered in Table 2 the benchmark **`mlp`** agent is able to obtain high classification accuracy on held out data. This would not be possible if the underlying system was fundamentally stochastic, due to the irreducible error due to chance.

D.2 RESULTS

In this section we provide some supplementary results that analyze the performance of our benchmark agents on real data. To allow for hyperparameter tuning separately on the testbed and real datasets, we included different sweeps for the testbed and real datasets. Our sweeps for real data can be found in our open source code under the path `/agents/factories/sweeps/real_data`.

One of the headline results in our paper is Figure 7, which shows that the quality of joint predictions on the testbed is highly correlated with performance in real data. Figure 4 shows that this result is still true when you restrict the evaluation to the 'full training data' setting in each dataset. Further, this aggregate correlation is not driven by just one outlier dataset, but actually occurs in each dataset individually. In fact, after bootstrapping only the results on Iris were not significant at the 95% confidence levels. This gives some additional reassurance that the relationship between joint performance on testbed and real data is robust.

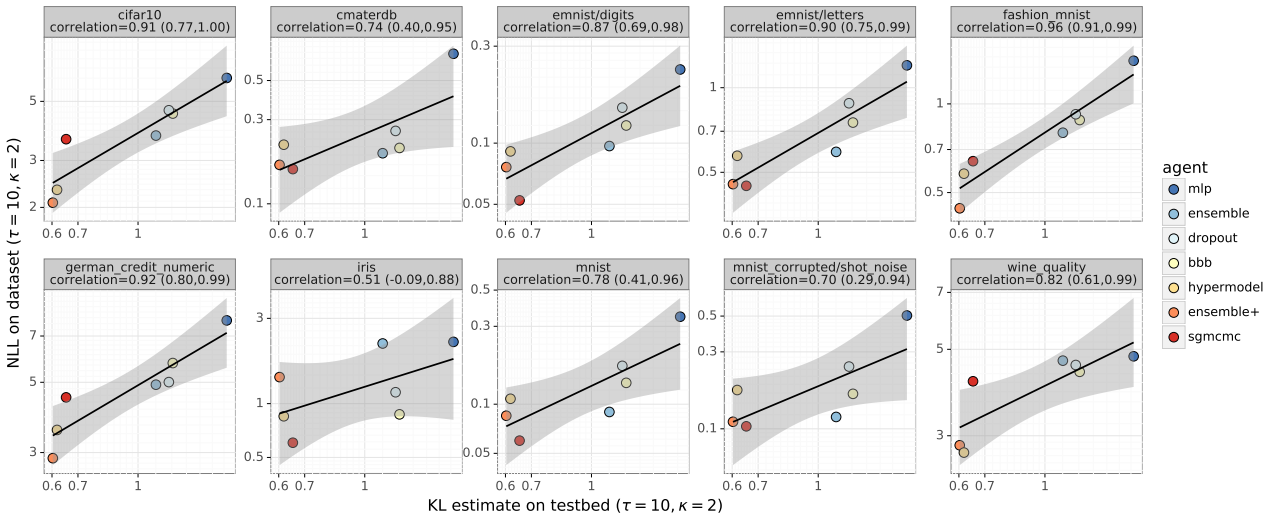

Figure 4: The quality of joint predictions on the testbed is highly correlated with performance in real data.

Our results in this paper allow for hyperparameter tuning separately on the testbed and real datasets. We believe that this is reasonable practice, and reflects the way machine learning algorithms are usually used in practice. However, one natural question might be if tuning an agent's performance on the testbed leads to good hyperparameter settings on real data. Figure 5 shows the results of this analysis across a wide range of agent-hyperparameter pairs. Agent-hyperparameter pairs that perform better on the testbed generally also perform better on real data. This result is statistically significant in both $\tau = 1$ and $\tau = 10$ dyadic sampling. However, we do see a stronger correlation in joint predictions rather than marginals. So while we do not necessarily recommend tuning your agent for real datasets using the Neural Testbed, these results say that it will provide a better answer on average than random chance.

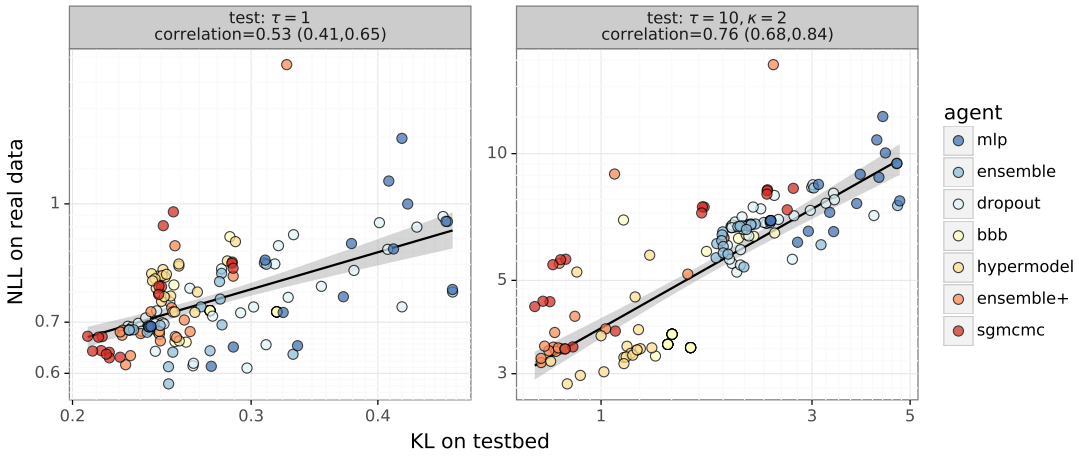

Figure 5: Agent-hyperparameter pairs that perform better on the testbed generally also perform better on real data.

References

Ian Osband, Zheng Wen, Mohammad Asghari, Morteza Ibrahimi, Xiyuan Lu, and Benjamin Van Roy. Epistemic neural networks. *arXiv preprint arXiv:2107.08924*, 2021.

Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Botao Hao, Morteza Ibrahimi, Dieterich Lawson, Xiuyuan Lu, Brendan O'Donoghue, and Benjamin Van Roy. The neural testbed: Evaluating predictive distributions, 2022.