Norwegian University of Science and Technology

DT8122 — **Project Report: Uncertainty Quantification**

PROBABILISTIC AI

Abdulmajid Abdullah Yahya Murad
abdulmajid.a.murad@ntnu.no
Student Number: 514455

1 Introduction

Deep learning models have been widely used in many real-world applications and quantifying their uncertainties are of crucial importance. However, most deep learning models operate in a "deterministic" fashion by having only point estimates of parameters and predictions, thus lacking credibility for decision making in real-life settings. In comparison, probabilist models place distributions over parameters and predictions, which provides more information about model uncertainty. In this project, we explore Bayesian methods and their approximate inference as techniques to capture model uncertainty in prediction tasks. We implement two deep generative modeling techniques: Dropout as a Bayesian Approximation to estimate uncertainty as well as Bayes by Backprop as a technique to approximate Bayesian neural networks using stochastic variational inference. We then demonstrate the inference methods and perform empirical evaluations on various benchmark regression dataset. Finally, we discuss the shortcomings of these two techniques, then suggest and implement possible improvement along with empirical evaluation and results comparison.

2 Dropout as a Bayesian Approximation

2.1 Implementation

This section describes our implementation of Dropout as a Bayesian Approximation [1] as a technique to build a deep probabilistic regression model. Then we used this model to estimate uncertainty by generating prediction intervals along with point predictions. We followed the notation from Pearce et al. [2], in which $\mathbf{x}_i \in \mathbb{R}^D$ is the ith D dimensional input features corresponding to target observation y_i, where $1 \leq i \leq n$ for n data points.

For building the model, we used a neural network that outputs a normal distribution (i.e., the NN has two output nodes: one represents the predicted mean $\hat{\mu}_i$, while the other node

represents the predicted variance $\hat{\sigma}_i$). Therefore, the Negative Log Likelihood (NLL) of the predicted distribution is used as training loss function (1).

$$Loss_{NLL} \approx -\sum_{i=1}^{n} \left[\log \hat{\sigma}_i - \frac{(y_i - \hat{\mu}_i)^2}{2\hat{\sigma}_i^2} \right] \tag{1}$$

We trained the NN with dropout and weight decay using 90% of a dataset, and evaluate it using the tail 10%. During the evaluation, we kept the dropout and used (T=1000) forward iterations for every data point to estimate the model's uncertainty. Because the model outputs a normal distribution for every single forward pass, multiple forward passes will result in a mixture of normal distributions with predictive mean as in (2), and predictive variance as in (3).

$$\hat{\boldsymbol{\mu}}_i = \frac{1}{T} \sum_{t=1}^{T} \hat{\mu}_{i_t} \tag{2}$$

$$\hat{\boldsymbol{\sigma}}_i = \sqrt{\frac{1}{T} \sum_{t=1}^{T} \hat{\sigma}_{i_t}^2 + \frac{1}{T} \sum_{t=1}^{T} \hat{\mu}_{i_t}^2 - \left(\frac{1}{T} \sum_{t=1}^{T} \hat{\mu}_{i_t} \right)^2} \tag{3}$$

Since we have a predicted mean and variance, we get a point prediction by $\hat{y}_i = \hat{\boldsymbol{\mu}}_i$, and can calculate the prediction interval (PI): $\hat{y}_{Ui} = \hat{\boldsymbol{\mu}}_i + z\hat{\boldsymbol{\sigma}}_i$, $\hat{y}_{Li} = \hat{\boldsymbol{\mu}}_i - z\hat{\boldsymbol{\sigma}}_i$, where z is the standard score. For a 95% prediction interval ($\gamma = 0.95$), $z = 1.96$.

Given a point prediction $\hat{y}_i$, predicted lower bound $\hat{y}_{Li}$, and predicted upper bound$\hat{y}_{Ui}$, we can calculate RMSE, PICP, and MPIW as in (4), (5), and (6), consecutively:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{4}$$

$$PICP = \frac{1}{n} \sum_{i=1}^{n} \max(0, sign(y_i - \hat{y}_{Li})) \ \odot \ \max(0, sign(\hat{y}_{Ui} - y_i)) \tag{5}$$

$$MPIW = \frac{1}{n} \sum_{i=1}^{n} \hat{y}_{Ui} - \hat{y}_{Li} \tag{6}$$

2.2 Empirical Evaluation

Table 1 presents the evaluation results of the implemented dropout models on ten regression datasets, in terms of RMSE, PICP, and MPIW. These results are comparable to those reported in [1, 2]. Prediction intervals are plotted in appendix A in terms of predicted upper and lower bounds, as well as point predictions in comparison the target (true) values.

Table 1: Evaluation results of models trained on ten regression datesets using Dropout as a Bayesian Approximation with NLL loss

Dataset	RMSE	PICP	MPIW
BOSTON	2.282	0.961	1.275
CONCRETE	6.598	0.971	1.738
ENERGY	2.280	1.000	0.944
KIN8NM	0.079	0.983	1.387
NAVAL	0.004	0.999	2.539
POWER	4.214	0.971	0.994
PROTEIN	4.694	0.962	2.841
WINE	0.631	0.925	2.891
YACHT	3.052	1.000	0.839
YEAR	8.569	0.950	2.652

2.3 Improvement: Dropout as a Bayesian Approximation Using a Quality-Driven Loss Function

One of the shortcomings of Dropout as a Bayesian Approximation approach is that it requires a strong assumption when casting dropout as approximate Bayesian inference in deep Gaussian process. As Osband [3] has shown analytically that the posterior we get from MC dropout does not concentrate asymptotically, which means that the Bayesian approximation is not exactly right.

Another shortcoming stems from the fact that current uncertainty modeling using BNNs, or dropout as a Bayesian approximation, ignore model misspecification, and only estimates training data uncertainty and parameter uncertainty. Pearce et al. [2] have proposed using a quality-driven (QD) loss function to overcome this shortcoming. This loss function is based on the idea that high-quality prediction intervals should be as narrow as possible while capturing a specified portion of data. Therefore, we implemented the QD loss function instead of NLL in a dropout model. The QD loss optimizes PI's width and coverage in (7), with a hyper-parameter λ controlling the importance of width vs. coverage.

$$Loss_{QD} = MPIW_{capt.} + \lambda \frac{n}{\alpha(1-\alpha)} \max(0,(1-\alpha) - PICP_{soft})^2 \tag{7}$$

Captured MPIW ($MPIW_{capt.}$) is the mean width of PIs that only capture their data points (8). It is defined instead of (6) to avoid encouraging PIs that fail to capture their data point

to shrink further.

$$MPIW_{capt.} := \frac{\frac{1}{c}\sum_{i=1}^{n}(\hat{y}_{Ui} - \hat{y}_{Li}) \ \odot \ k_{i_{hard}}}{\sum_{i=1}^{n} k_{i_{hard}}} \tag{8}$$

Additionally, $PICP_{soft}$ is defined in (8) instead of (5) to approximate the step function with the sigmoid function σ and a softening factor s, thus avoiding discontinuities and smoothing gradient descent.

$$PICP_{soft} = \frac{1}{n}\sum_{i=1}^{n} \sigma(s(y_i - \hat{y}_{Li})) \ \odot \ \sigma(s(\hat{y}_{Ui} - y_i)) \tag{9}$$

We built a neural network with two output modes: one representing the predicted lower bound $\hat{y}_{Li}$ and the other node representing the upper bounds $\hat{y}_{Ui}$. Additionally, we used the mean of PI's bounds as point prediction $\hat{y}_i = (\hat{y}_{Ui} + \hat{y}_{Li})/2$. We trained the NN with dropout and weight decay using 90% of a dataset, and evaluate it using the tail 10%. During the evaluation, we kept the dropout and used (T=1000) forward iterations for every data point to estimate the model's uncertainty.

2.4 Empirical Evaluation of Improvement

Table 2 presents the evaluation results of the QD-based models in terms of RMSE, PICP, and MPIW. It also compares the improvement gained from using QD-based models instead of NLL-based ones. Further, predictions intervals plots are shown in appendix B. The results follow what we had expected: there is a huge improvement in MPIW (QD-Models have smaller PI width than those with NLL), since minimizing the QD loss will lead to minimizing PI width. There is no substantial difference in PICP since in QD loss the penalty only occurs when $PICP < \gamma$. However, the RMSE increased in QD-Models, because the main objective of the QD loss is to decrease uncertainty, not accuracy (only indirectly).

3 Bayes by Backprop

3.1 Implementation

This section describes our implementation of Bayes by Backprop [4], as a technique to approximate Bayesian neural networks using stochastic variational inference and backpropagation. In BNN, a model's weights are treated as distributions rather than point estimates, and the training objective is simply calculating the posterior distribution $P(\mathbf{w})$ of the weights, given the data. Usually, a variational approximation $q(\mathbf{w}|\theta)$ to the posterior is used and the learning is reduced to finding the variational parameters θ that minimize the KL divergence with the true posterior, resulting in a loss function (ELBO) as in (10).

$$Loss = KL[q(\mathbf{w}|\theta)||P(\mathbf{w})] - \mathop{\mathbb{E}}_{q(\mathbf{w}|\theta)}[\log P(D|\mathbf{w}] \tag{10}$$

Table 2: Evaluation results of models trained on ten regression datasets using Dropout as a Bayesian Approximation with QD loss

Dataset	RMSE		PICP		MPIW	
	QD-Model	Improv.	QD-Model	Improv.	QD-Model	Improv.
BOSTON	2.282	0.%	0.980	1.98%	1.054	17.33%
CONCRETE	6.927	-4.99%	0.942	-2.99%	1.392	19.91%
ENERGY	2.580	-13.16 %	0.987	-1.3%	0.838	11.23%
KIN8NM	0.087	-10.13 %	0.960	-2.34%	1.271	8.36%
NAVAL	0.005	-25.%	0.951	-4.8%	2.054	19.1%
POWER	4.440	-5.36%	0.952	-1.96%	0.906	8.85%
PROTEIN	5.034	-7.24%	0.937	- 2.6%	02.242	21.08%
WINE	0.710	-12.52%	0.944	2.05%	2.183	24.49%
YACHT	3.969	-30.05%	1.000	0. %	0.616	26.58%
YEAR	9.219	-7.59 %	0.938	-1.26%	2.315	12.71%

Blundell et al. [4], approximated ELBO using sampled weights, which is backpropagation-compatible (11)

$$Loss \approx \sum_{i=1}^{n} \log q(\mathbf{w}^i|\theta) - \log P(\mathbf{w}^i) - \log P(D|\mathbf{w}^i) \tag{11}$$

In our implementation, we used a neural network that also predicts a distribution (i.e. the NN has two output nodes: one represents the mean μ_i, while the other node represents the variance σ_i). Therefore, the likelihood part of (11) is represented by the Negative Log Likelihood (NLL) of the predicted distribution given data (1).

We trained the NN with (T=20) samples, using 90% of a dataset, and evaluate it using the tail 10%. During the evaluation, we used (T=1000) samples for every data point to estimate the model's uncertainty. This in effect corresponds to sampling from infinite ensembles of neural networks, since we have distributions on weight. Therefore, combining the outputs from different samples gives information on the model's uncertainty.

3.2 Empirical Evaluation

Table 3 presents the evaluation results of the Bayes models on ten regression datasets, and figures are in appendix C shows graphs of prediction intervals of every dataset. These results are also comparable to those reported in [1, 2], but slightly inferior to dropout models. We also found that Bayes by Backprop requires high computation. Therefore we were not able to

make it work on last dataset "Year Prediction MSD" since it is a large dataset and requires a prohibitive computational cost (in terms of our limited resources).

Table 3: Evaluation results of models trained on ten regression datasets using Bayes by Backprop with NLL loss

Dataset	RMSE	PICP	MPIW
BOSTON	2.414	0.961	0.973
CONCRETE	6.680	0.990	1.409
ENERGY	1.065	0.948	0.496
KIN8NM	0.098	0.952	1.391
NAVAL	0.007	1.000	3.911
POWER	4.709	0.956	1.140
PROTEIN	6.162	0.931	3.372
WINE	0.633	0.925	2.787
YACHT	3.246	1.000	0.507
YEAR	NA	NA	NA

3.3 Improvement: Bayes by Backprop using a Quality-Driven Loss Function

The major limitation of the Bayes by Backprop approach is that it requires high computational cost and more time to converge. Additionally, it requires more parameters to represent uncertainty (double those in a standard NN). Further, by using BNN, it ignores model misspecification, and only estimates training data uncertainty and parameter uncertainty [2]. For this reason, we implemented the QD loss function (7) instead of NLL to improve upon its uncertainty estimation.

To leverage the QD loss, we built a neural network that has two output nodes: one representing the predicted lower bound $\hat{y}_{Li}$ and the other node representing the upper bounds $\hat{y}_{Ui}$. We trained the NN with (T=20) samples, using 90% of a dataset, and evaluate it using the tail 10%. During the evaluation, we used (T=1000) samples for every data point to estimate the model's uncertainty. Table 2 presents the evaluation results of the QD-based models in terms of RMSE, PICP, and MPIW. It also compares the improvement gained from using QD-loss instead of NLL. Further, predictions intervals plots are shown in appendix D.

3.4 Empirical Evaluation of Improvement

We found that Bayes by Backprop to be very sensitive to initial values and hyper-parameters, and as expected, there is much improvement in MPIW for most models (except for those trained on ENERGY and YACHT dataset, most probably, due to lack of thorough hyper-parameters tuning). The RMSE increased in QD-based Models, QD does not optimize accuracy directly. However, PICP decreased in most models, which was not expected (most probably due to lack of through hyper-parameters tuning).

Table 4: Evaluation results of models trained on ten regression datasets using Bayes by Backprop with QD loss

Dataset	RMSE		PICP		MPIW	
	QD-Model	Improv.	QD-Model	Improv.	QD-Model	Improv.
BOSTON	2.763	-14.46 %	0.941	-2.08%	0.978	-0.51 %
CONCRETE	6.928	-3.71 %	0.816	-17.58%	1.002	28.89%
ENERGY	3.106	-191.64%	0.896	-5.49%	0.905	-82.46%
KIN8NM	0.094	4.08%	0.910	-4.41%	1.204	13.44%
NAVAL	0.007	0.%	0.963	-3.7%	3.247	16.98%
POWER	4.517	4.08%	0.939	-1.78 %	0.874	23.33%
PROTEIN	5.550	9.93%	0.960	3.11%	2.909	13.73 %
WINE	0.691	-9.16%	0.900	-2.7%	1.916	31.25 %
YACHT	3.557	-9.58%	0.968	-3.2 %	0.722	-42.41 %
YEAR	NA	NA	NA	NA	NA	NA

4 Discussion and Conclusion

In this project, we explored uncertainty quantification using approximate Bayesian methods. We implemented two deep generative modeling techniques to estimate uncertainty by generating prediction intervals along with point predictions. Firstly, we implemented Dropout as a Bayesian Approximation, and empirically evaluated this method in a set of regression tasks. Furthermore, we discussed the shortcomings of Bayesian Approximation and implemented an improvement based on QD loss function instead of NLL. Our findings show that using QD decrease model's uncertainty (by decreasing MPIW), although increase RMSE. Secondly, we implemented Bayes by Backprop as a technique to approximate Bayesian neural networks using stochastic variational inference and empirically evaluated it ten benchmark dataset. Addition-

ally, we discussed the limitations of Bayes by Backprop and implemented an improvement based on QD loss. We found that using QD, generally, reduce MPIW but increase RMSE. We also found that Bayes by Backprop requires high computation costs and very sensitive to initial values and hyper-parameters. It requires careful hyper-parameters tuning to get to work on most datasets.

References

[1] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[2] Tim Pearce, Alexandra Brintrup, Mohamed Zaki, and Andy Neely. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4075–4084, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[3] Ian Osband. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS Workshop on Bayesian Deep Learning*, 2016.

[4] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France, 07–09 Jul 2015. PMLR.

Appendices

A Prediction Intervals of Dropout Models with Log-Likelihood Loss Function

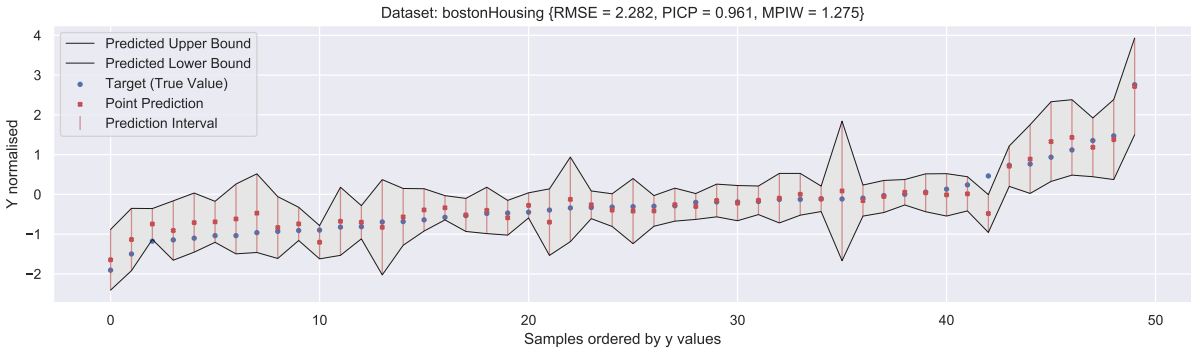

Figure 1: PI of a dropout model with NLL loss: Boston Housing Dataset

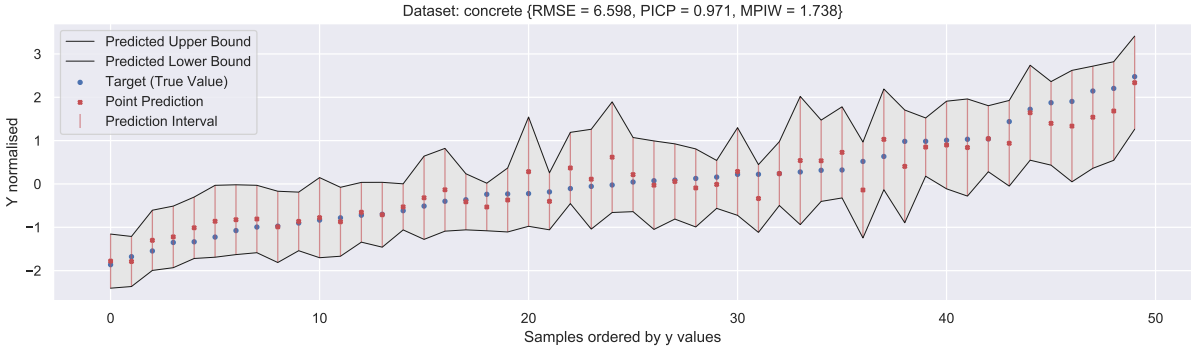

Figure 2: PI of a dropout model with NLL loss: Concrete Dataset

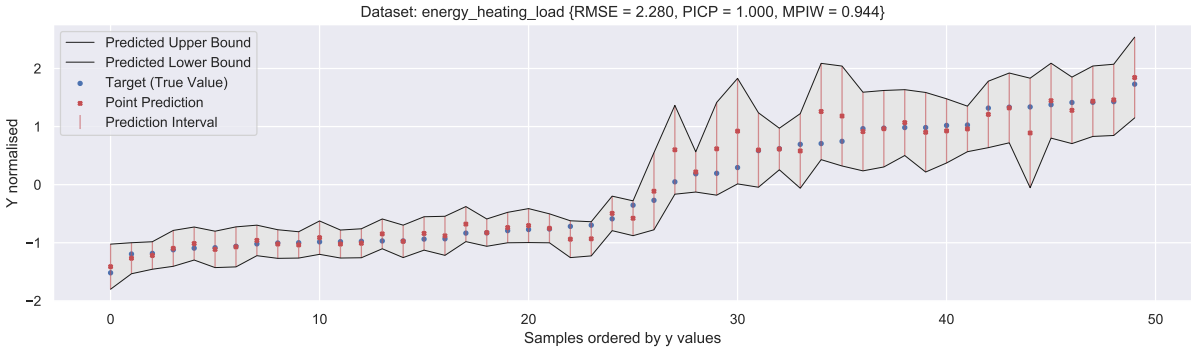

Figure 3: PI of a dropout model with NLL loss: Energy Dataset

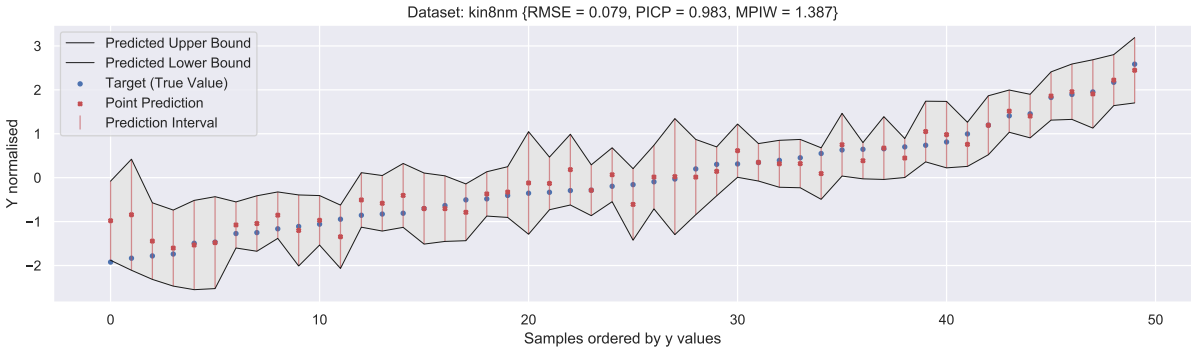

Figure 4: PI of a dropout model with NLL loss: Kin8nm Dataset

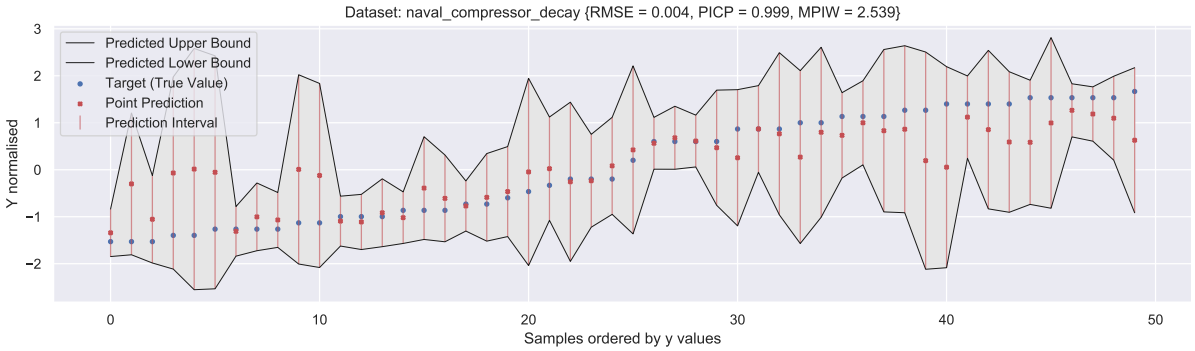

Figure 5: PI of a dropout model with NLL loss: Naval Dataset

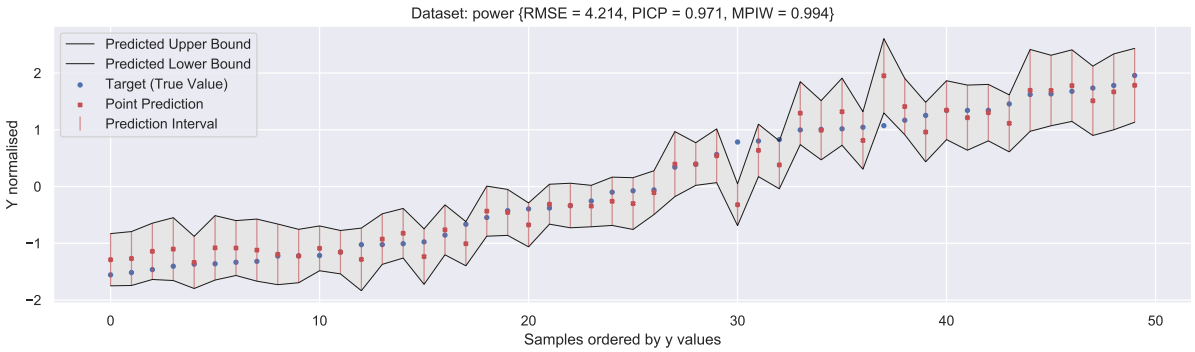

Figure 6: PI of a dropout model with NLL loss: Power Dataset

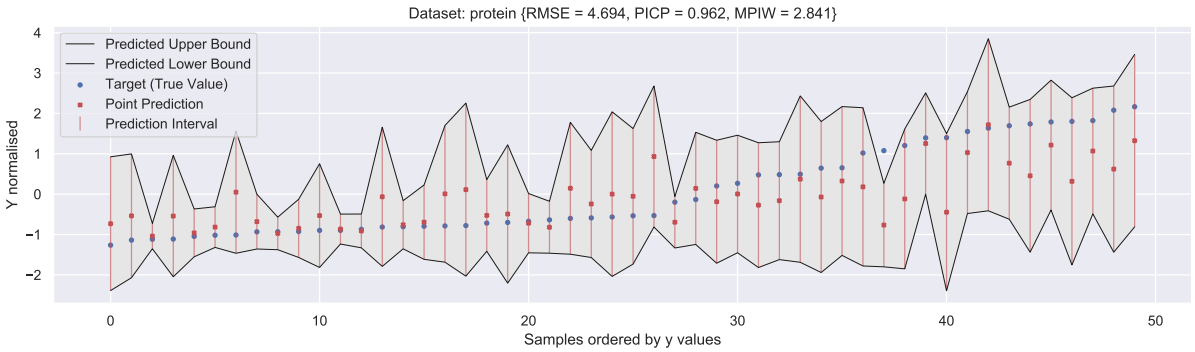

Figure 7: PI of a dropout model with NLL loss: Protein Dataset

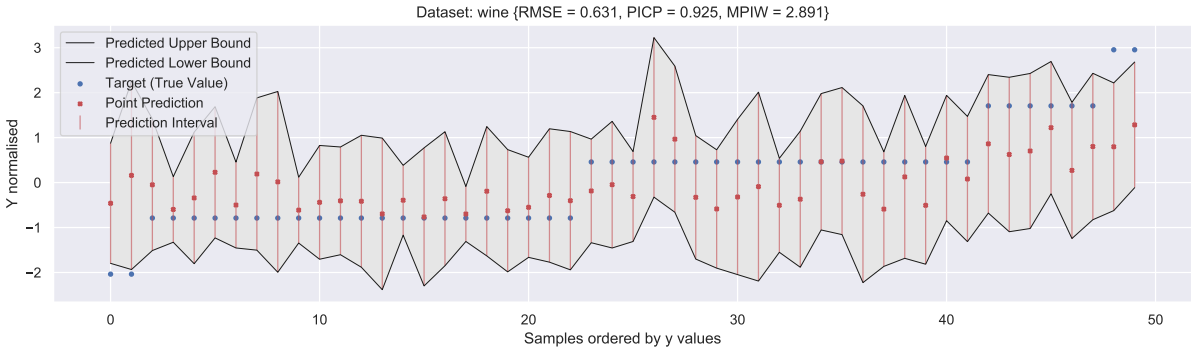

Figure 8: PI of a dropout model with NLL loss: Wine Dataset

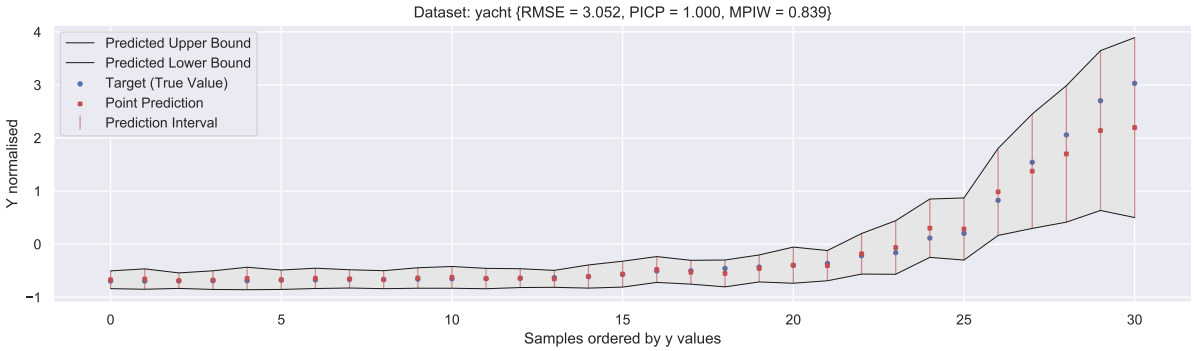

Figure 9: PI of a dropout model with NLL loss: Yacht Dataset

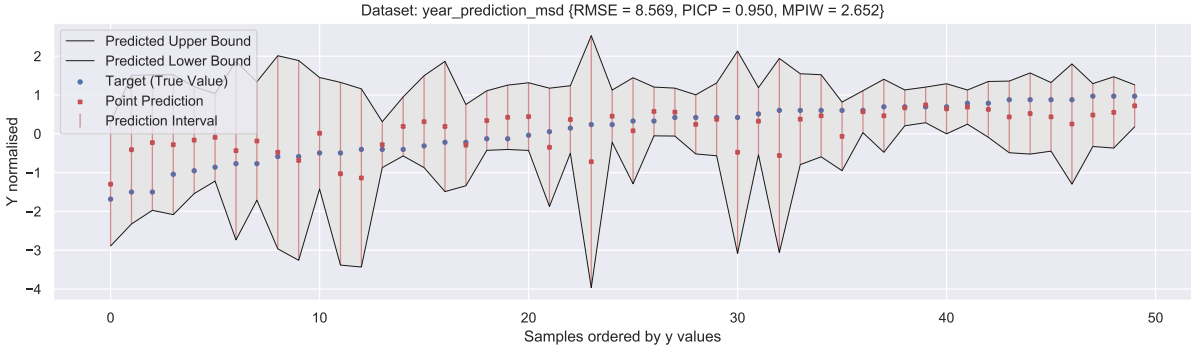

Figure 10: PI of a dropout model with NLL loss: Year Prediction Dataset

B Prediction Intervals of Dropout Models with Quality-Driven Loss Function

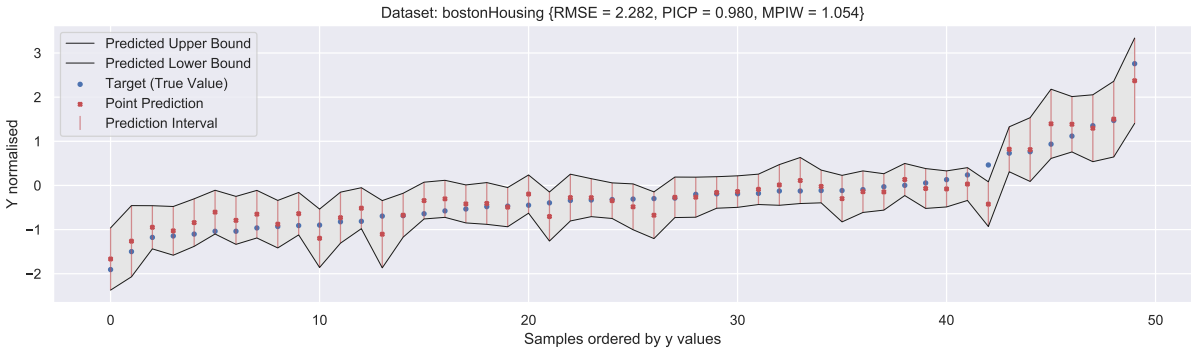

Figure 11: PI of a dropout model with QD loss: Boston Housing Dataset

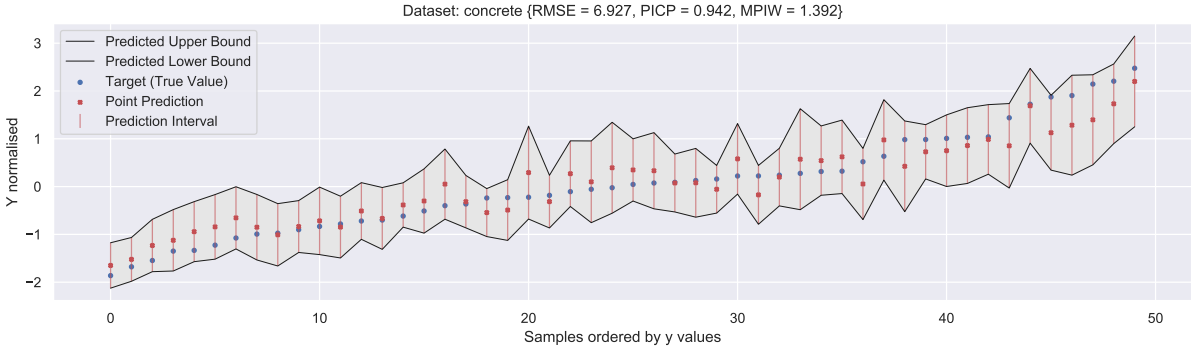

Figure 12: PI of a dropout model with QD loss: Concrete Dataset

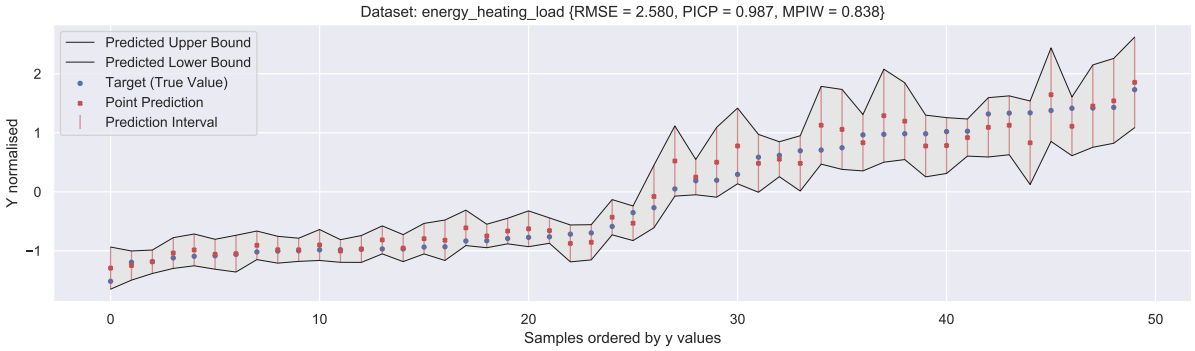

Figure 13: PI of a dropout model with QD loss: Energy Dataset

Figure 14: PI of a dropout model with QD loss: Kin8nm Dataset

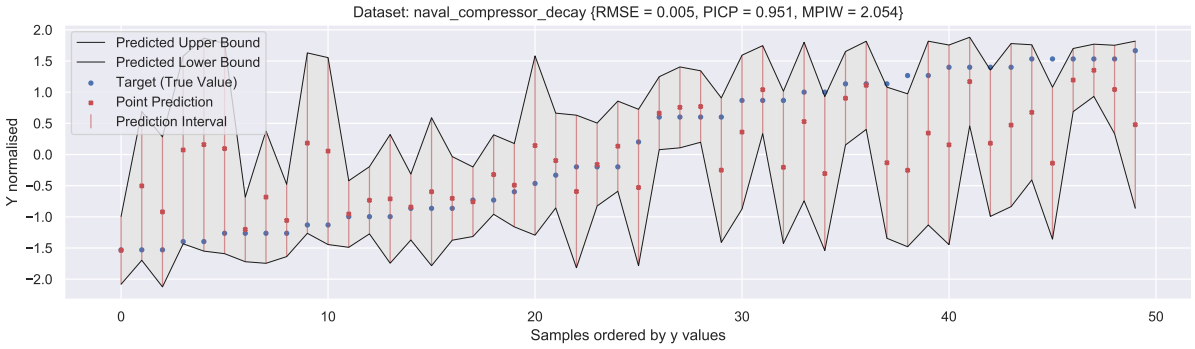

Figure 15: PI of a dropout model with QD loss: Naval Dataset

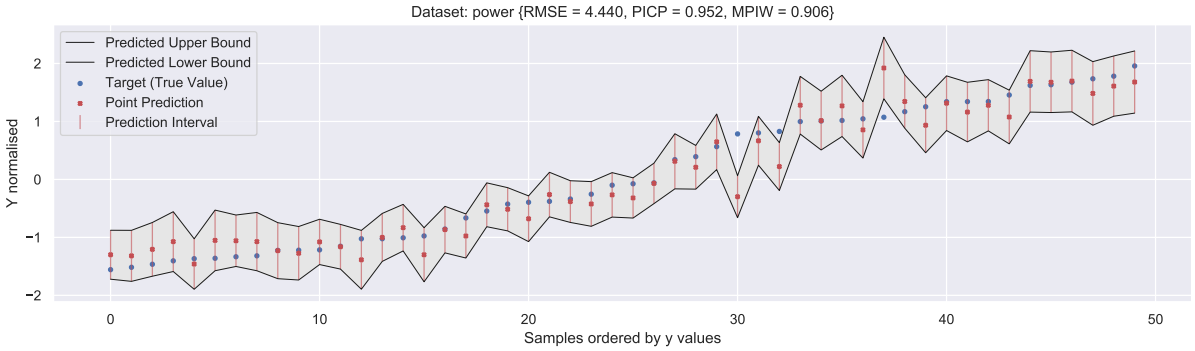

Figure 16: PI of a dropout model with QD loss: Power Dataset

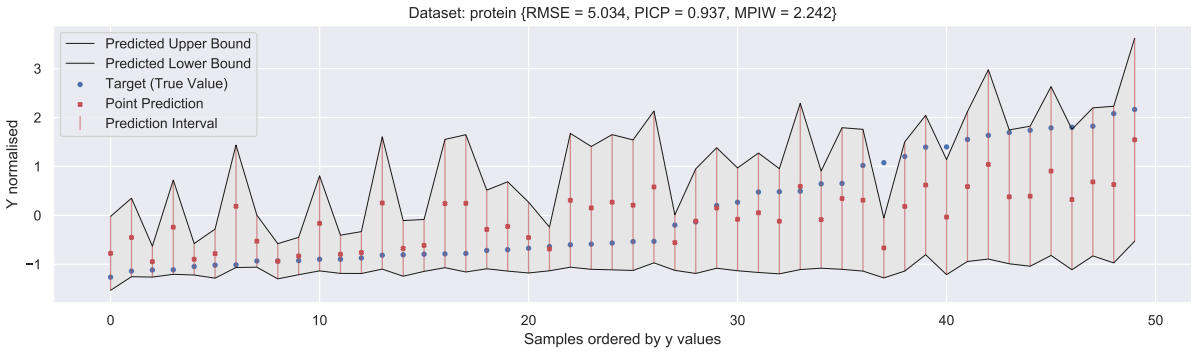

Figure 17: PI of a dropout model with QD loss: Protein Dataset

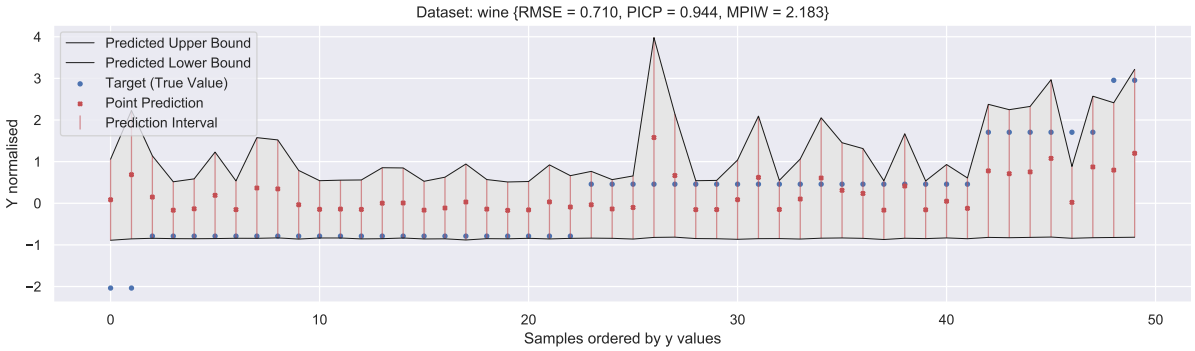

Figure 18: PI of a dropout model with QD loss: Wine Dataset

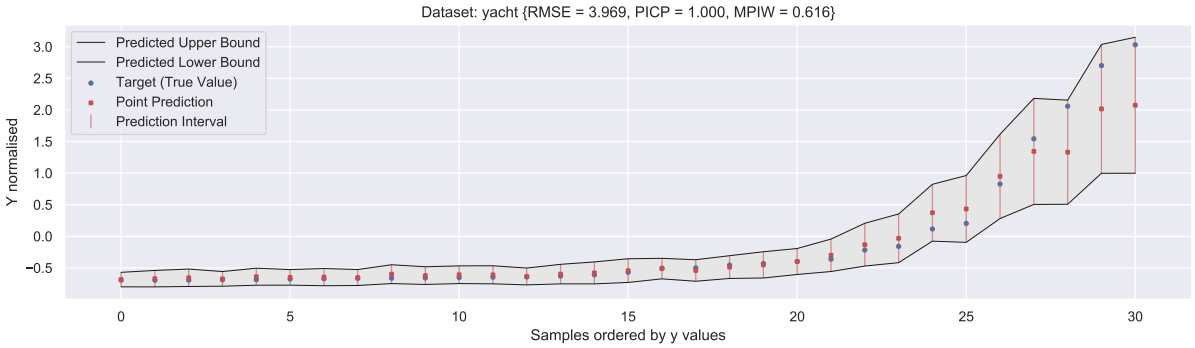

Figure 19: PI of a dropout model with QD loss: Yacht Dataset

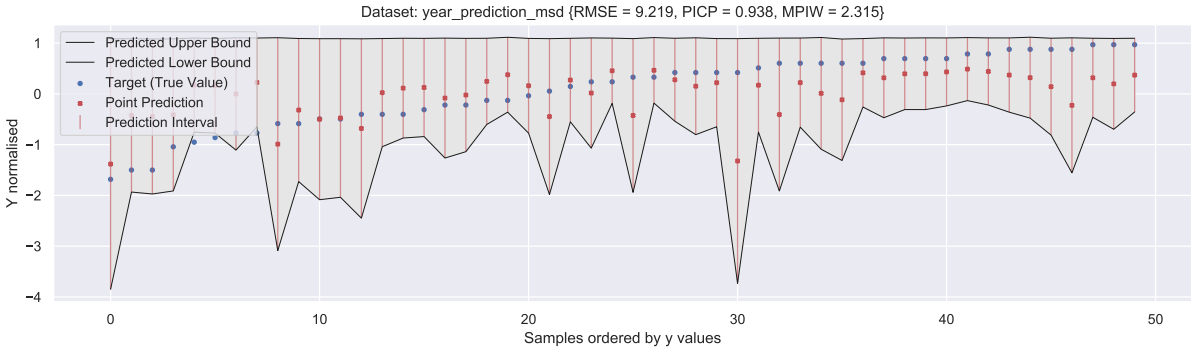

Figure 20: PI of a dropout model with QD loss: Year Prediction Dataset

C Prediction Intervals of Bayes Models with Log-Likelihood Loss Function

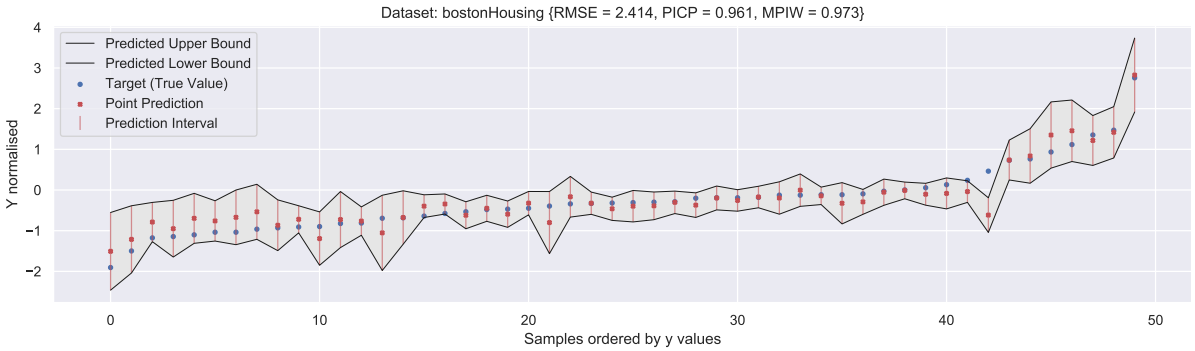

Figure 21: PI of a Bayes model with NLL loss: Boston Housing Dataset

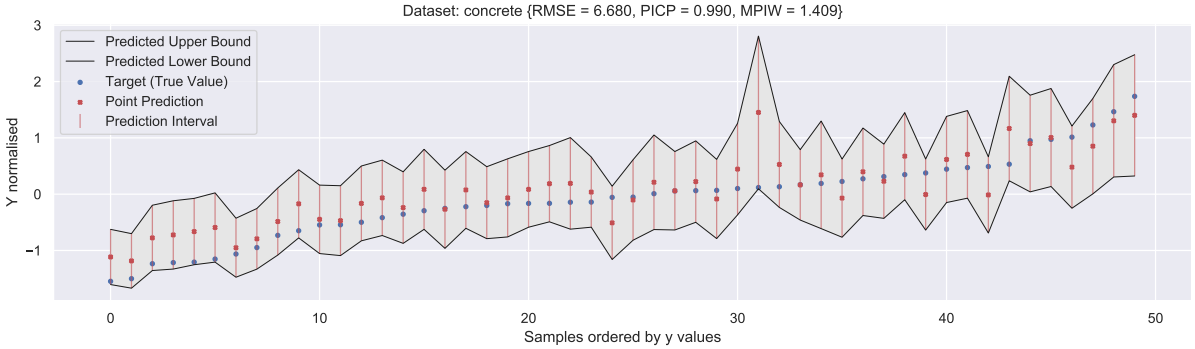

Figure 22: PI of a Bayes model with NLL loss: Concrete Dataset

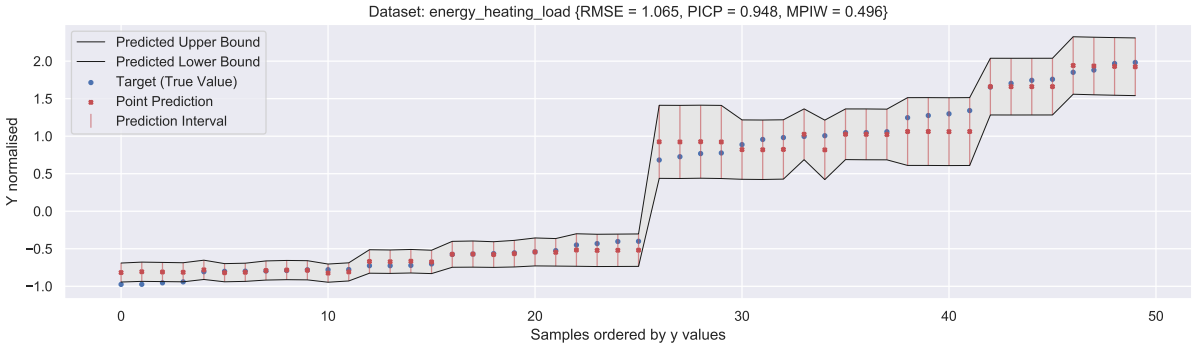

Figure 23: PI of a Bayes model with NLL loss: Energy Dataset

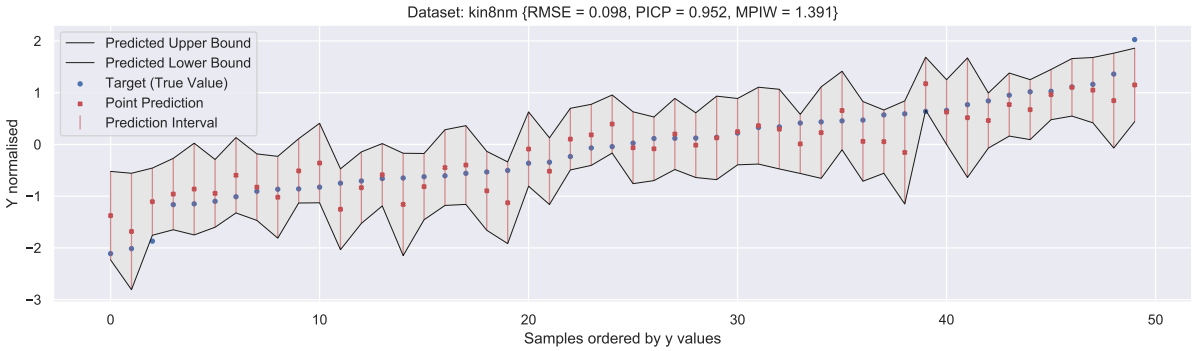

Figure 24: PI of a Bayes model with NLL loss: Bayes Dataset

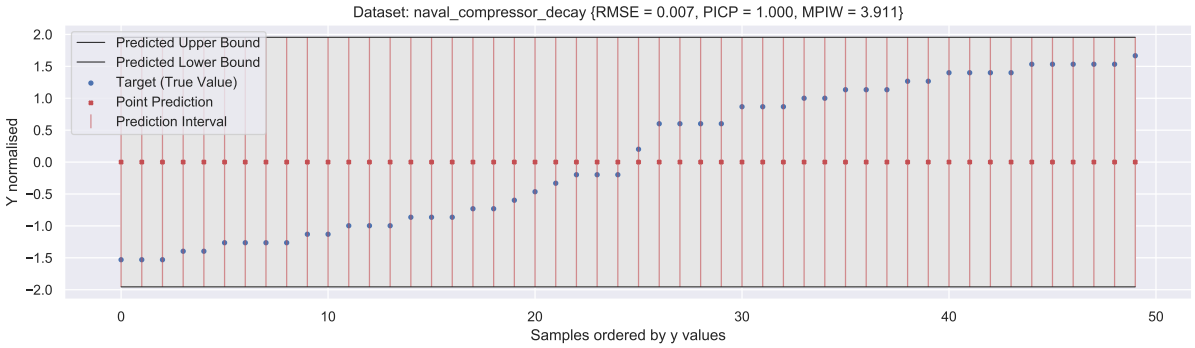

Figure 25: PI of a Bayes model with NLL loss: Naval Dataset

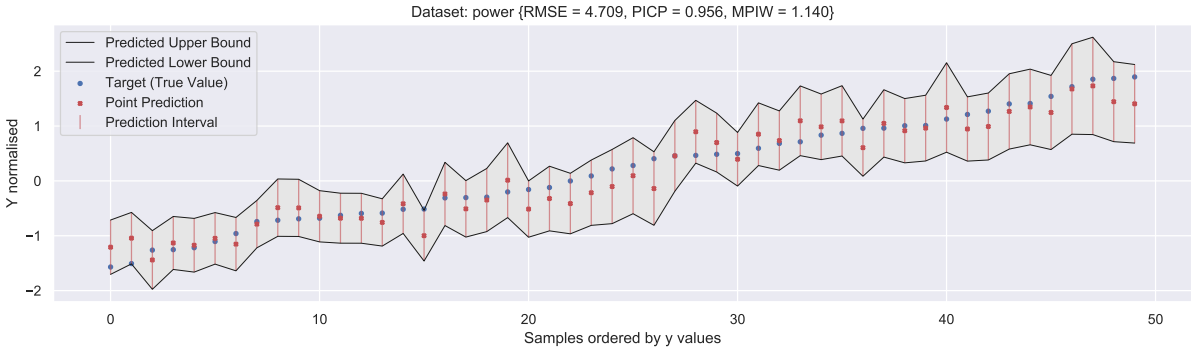

Figure 26: PI of a Bayes model with NLL loss: Power Dataset

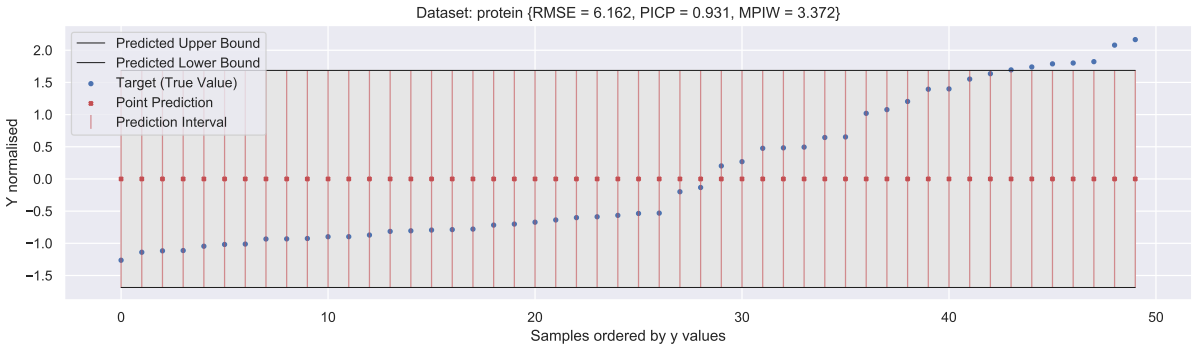

Figure 27: PI of a Bayes model with NLL loss: Protein Dataset

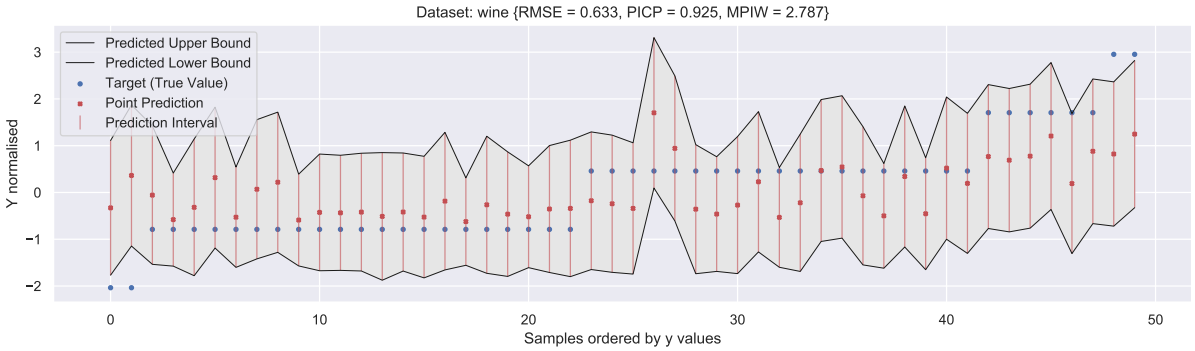

Figure 28: PI of a Bayes model with NLL loss: Wine Dataset

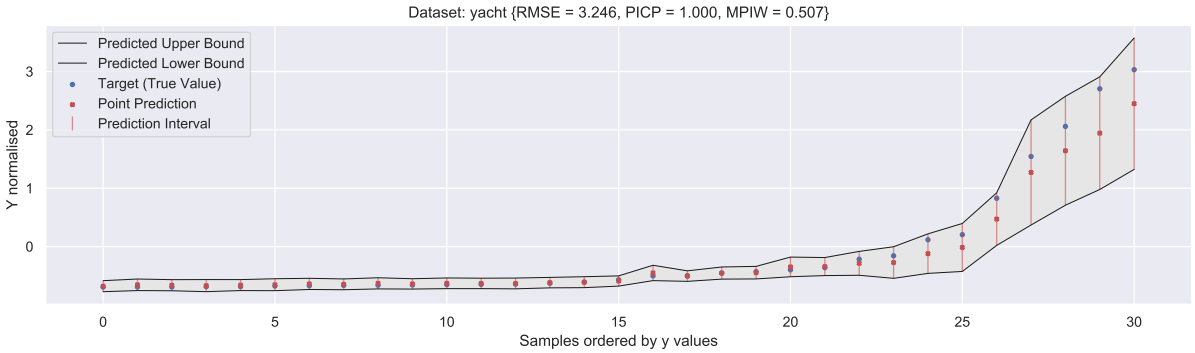

Figure 29: PI of a Bayes model with NLL loss: Yacht Dataset

D Prediction Intervals of Bayes Models with Quality-Driven Loss Function

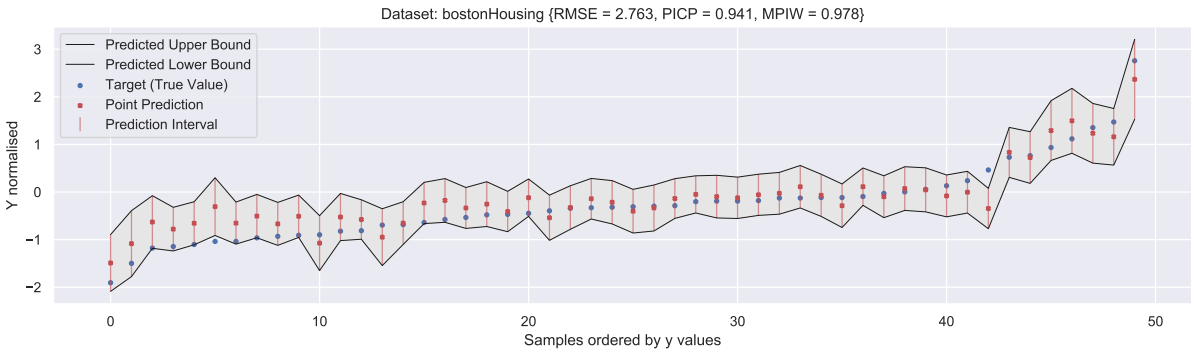

Figure 30: PI of a Bayes by Backprop model with QD loss: Boston Housing Dataset

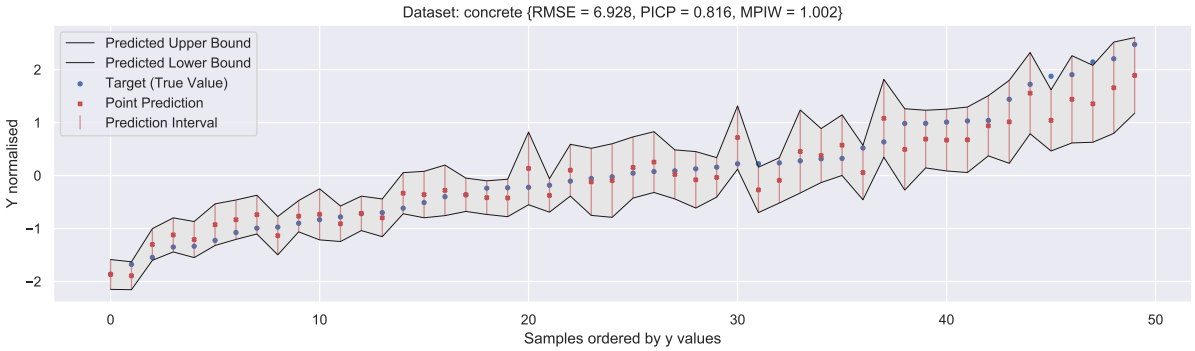

Figure 31: PI of a Bayes by Backprop model with QD loss: Concrete Dataset

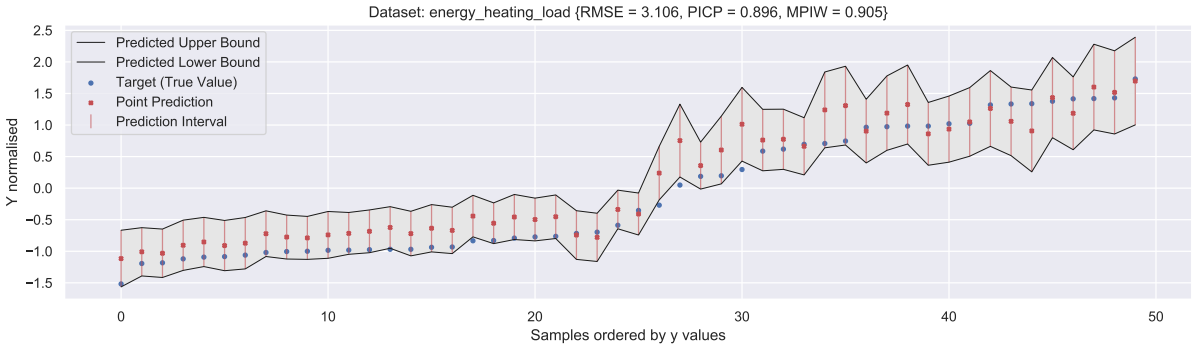

Figure 32: PI of a Bayes by Backprop model with QD loss: Energy Dataset

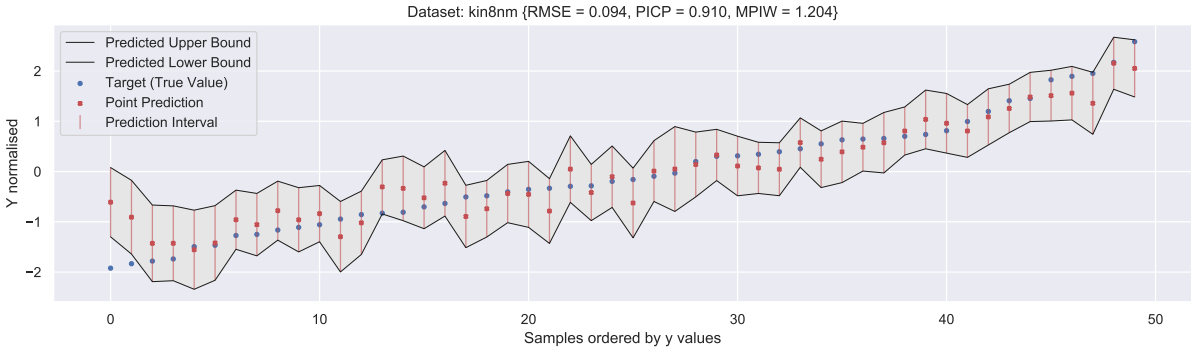

Figure 33: PI of a Bayes by Backprop model with QD loss: Kin8nm Dataset

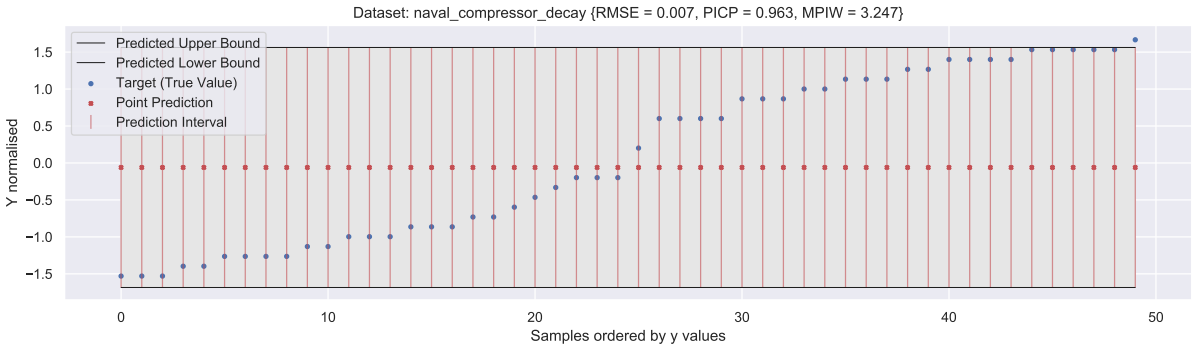

Figure 34: PI of a Bayes by Backprop model with QD loss: Naval Dataset

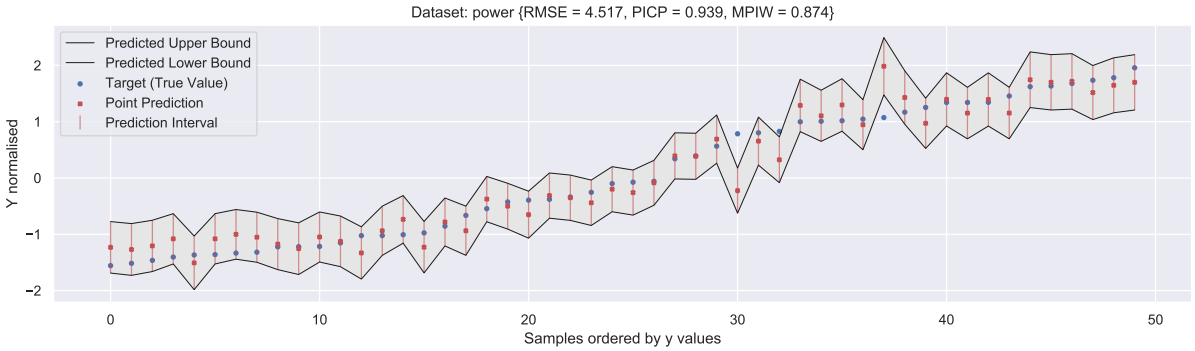

Figure 35: PI of a Bayes by Backprop model with QD loss: Power Dataset

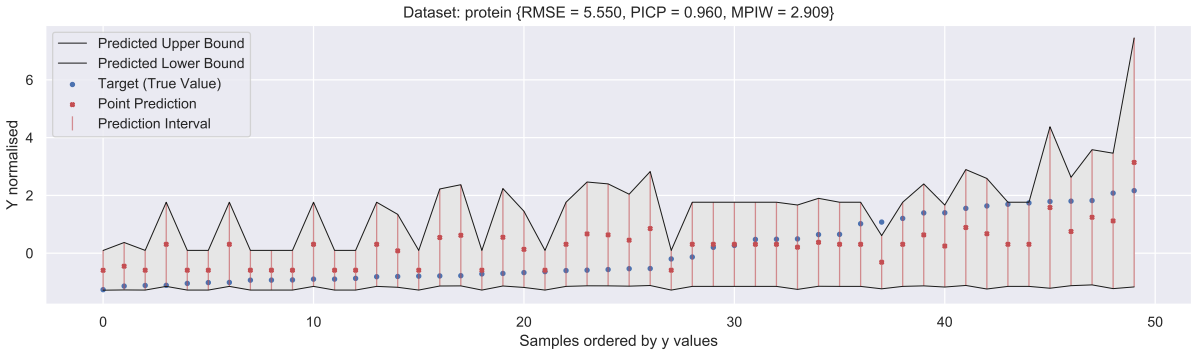

Figure 36: PI of a Bayes by Backprop model with QD loss: Protein Dataset

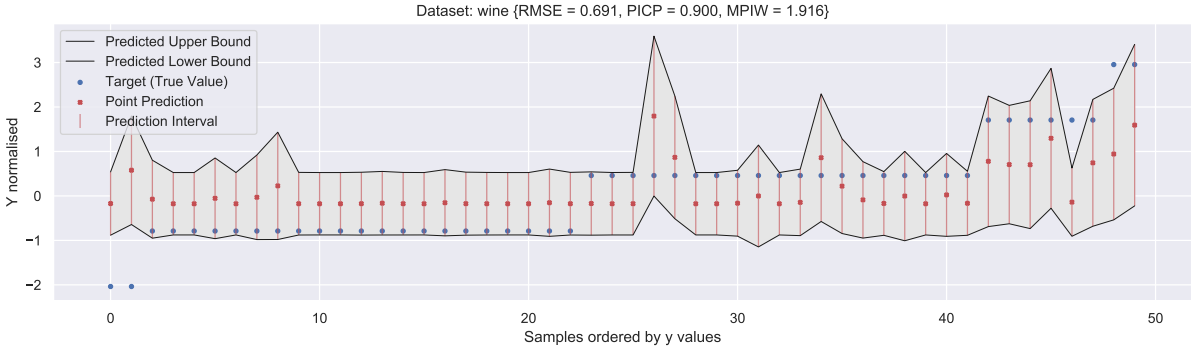

Figure 37: PI of a Bayes by Backprop model with QD loss: Wine Dataset

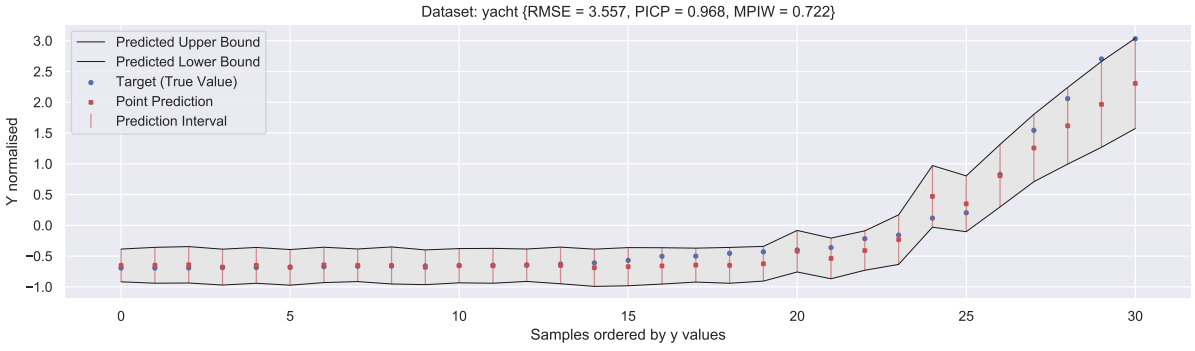

Figure 38: PI of a Bayes by Backprop model with QD loss: Yacht Dataset

E Sample of a Training Process

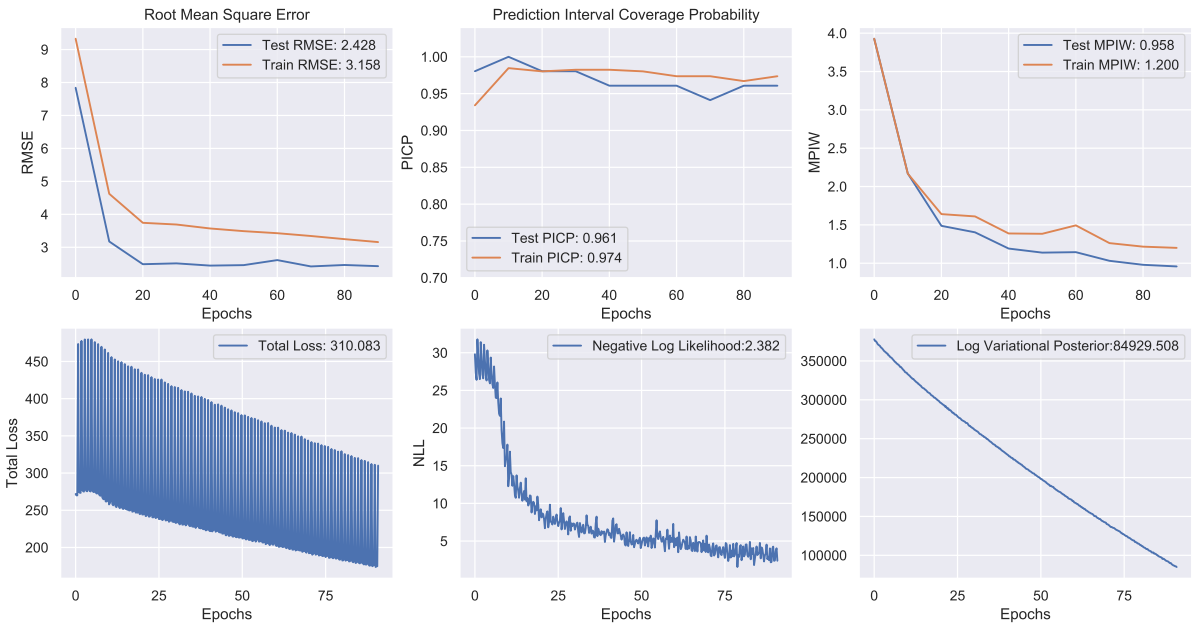

Figure 39: Training progress over epochs of a Bayes model with NLL loss using Boston Housing Dataset

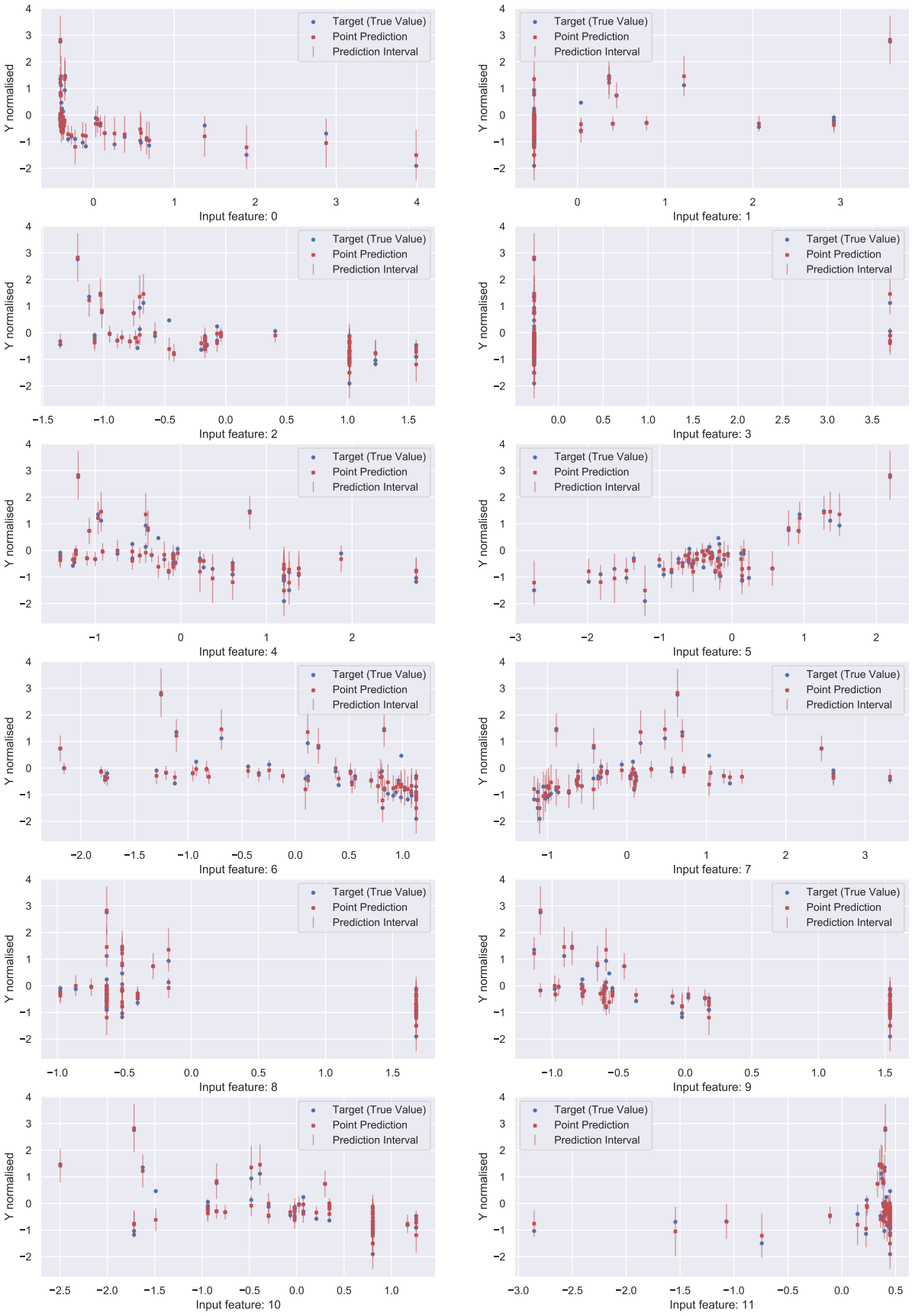

Figure 40: PI with regard to every input feature (Bayes model with NLL loss using Boston Housing Dataset)