Tight Prediction Intervals Using Expanded Interval Minimization

Dongqi Su
GoDaddy Inc
dsu5@godaddy.com

Ying Yin Ting
GoDaddy Inc
yying@godaddy.com

Jason Ansel
GoDaddy Inc
jansel@godaddy.com

Abstract

Prediction intervals are a valuable way of quantifying uncertainty in regression problems. Good prediction intervals should be both correct, containing the actual value between the lower and upper bound at least a target percentage of the time; and tight, having a small mean width of the bounds. Many prior techniques for generating prediction intervals make assumptions on the distribution of error which causes them to work poorly for problems with asymmetric distributions.

This paper presents, Expanded Interval Minimization (EIM), a novel loss function for generating prediction intervals using neural networks. This loss function uses minibatch statistics to estimate the coverage and optimize the width of the prediction intervals. It does not make the same assumptions on the distributions of data and error as prior work. We compare to three published techniques and show EIM produces on average $1.37x$ tighter prediction intervals and in the worst case $1.06x$ tighter intervals across two large real-world datasets and varying coverage levels.

1 Introduction

Prediction intervals are the preferred method of many people for quantifying uncertainty [5]. They work by providing a lower and upper bound for an estimated variable such that the value of predicted variable falls between the upper and lower bound for at least some target percentage (e.g., 90%) of holdout data. One can trivially achieve this correctness criterion with the bounds $[-\infty, \infty]$, so to be useful one also wants prediction intervals that are *tight*, having the minimum possible mean bound width while still satisfying the correctness criteria. There are three published techniques for generating prediction intervals with neural networks which we use as baselines to evaluate EIM.

Maximum likelihood estimation [16, 17] is a well-known method that can build prediction intervals based on two neural networks, one predicting the value and the second predicting the error. Ensemble method [2, 6] is another way of generating prediction intervals using variance between ensembles of many models to estimate error. Both of these techniques work well for problems with symmetric distributions because a Gaussian assumption on model error motivates their designs, however, they struggle to produce optimal bounds for the asymmetric distributions found in our datasets.

Quantile regression method [13, 14] is a modified version of least squares that converges to a given quantile of a dataset. One can use quantile regression to build prediction intervals that span between two given quantiles. For example, one could generate a prediction interval that spans from the 10% quantile regression as lower bound to the 90% quantile regression as an upper bound. We set these two upper and lower quantiles via exhaustive grid search to find the optimal values for each target coverage. This technique is more flexible than the prior two, in that it does not assume a specific data distribution. However, it is limited in that it is only able to express intervals in a quantile-to-quantile form.

This paper presents a novel technique for generating prediction intervals with neural networks called *expanded interval minimization* (EIM). We build a neural network structure that outputs both a lower

and upper bound directly and use a loss function that first scales the output bounds such that they cover the given target percentage of the minibatch, then minimizes the width of those scaled bounds during training. The scaling ensures that *correctness criteria* is met on the training data and then the training process optimizes only the *tightness* of those already correct bounds. Unlike maximum likelihood method and ensemble method, EIM can support asymmetric data and error distributions, and it is also able to create prediction intervals varying in the predicted quantiles between different samples. We empirically demonstrate that EIM produces tighter prediction intervals than prior techniques both on a real-word aftermarket domain name sales dataset and on a million song dataset. Compared to the next best method, EIM produces at least $1.26x$ tighter prediction intervals for a 70% coverage target, $1.21x$ tighter prediction intervals for and 80% target, and $1.06x$ tighter intervals for a 90% coverage target.

2 Related Work

Some past surveys focus on comparing techniques to construct the prediction interval for the neural network point estimation [12, 19]. The most common techniques to construct the prediction interval are the delta method (also known as analytical method) [3, 7, 12, 19], methods that directly predict the variance (maximum likelihood method and ensemble method) [2, 6, 12, 16, 17, 19] and quantile regression method [13, 14]. Our baseline methods exclude delta method because it is a poor fit for our datasets. Delta method assumes constant error variance in the dataset and noise homogeneity in all the samples, while our datasets has substantial differences in variance between different types of samples. We implement maximum likelihood method, ensemble method, quantile regression method and a naive baseline fixed bounds method to compare with our proposed technique EIM.

The prior work maximum likelihood method [16, 17] and ensemble method [2, 6], make an assumption that the target distribution ($y(x)$) can be broken into two independent terms representing the true regression and noise. This assumption leads these techniques estimating the *total prediction variance*:

$$\sigma_p^2(x) = \sigma_m^2(x) + \sigma_\epsilon^2(x) \tag{1}$$

where $\sigma_m^2(x)$ is the model uncertainty variance and $\sigma_\epsilon^2(x)$ is the data noise variance. These methods then add and subtract a constant k times the estimate of $\sigma_p(x)$ from the estimated regression $f(x)$ to construct the prediction intervals:

$$[f(x) - k\sigma_p(x), f(x) + k\sigma_p(x)] \tag{2}$$

2.1 Maximum Likelihood Method

Maximum likelihood method [16, 17] builds two neural networks. The first neural network estimates the regression and the second neural network estimates the total prediction variance $\sigma_p^2(x)$. The loss function E_y for the first neural network, is simply mean squared error:

$$E_t = \sum_{i=1}^{N} (y_i - f_{mle}(x_i))^2 \tag{3}$$

where y_i is the target value correspond to x_i and $f_{mle}(x_i)$ is the output of the first neural network. The loss function $E_{\sigma_p^2}$, for the second neural network is the mean squared error of the squared error of the first neural network:

$$E_{\sigma_p^2} = \sum_{i=1}^{N} \Big((y_i - f_{mle}(x_i))^2 - var_{mle}(x_i) \Big)^2 \tag{4}$$

where $var_{mle}(x_i)$ is the output of the second neural network. The final prediction interval is $[f_{mle}(x_i) - k\sqrt{var_{mle}(x_i)}, f_{mle}(x_i) + k\sqrt{var_{mle}(x_i)}]$, where k depends on the desired level of confidence motivated by Gaussian distribution. Following the recommendations of the authors, we train these two neural networks by splitting our training data in half using disjoint training data for each network.

2.2 Ensemble Method

Ensemble method [2, 6] tries to estimate $\sigma_p^2(x)$ by estimating $\sigma_m^2(x)$ and $\sigma_\epsilon^2(x)$ separately and then combining them. It prepares M bootstrap neural networks each trained on a subsample of the full dataset using (3). It estimates the regression by taking the average output of all M bootstrap neural networks. We denoted this estimated value as $f_{emb}(x)$

To estimate the model uncertainly $\sigma_m^2(x)$, it splits all M bootstrap neural work into M_2 group and calculated an initial set L of M_2 average predictions for each group. Given this initial set L, it then estimates $\sigma_m^2(x)$ by taking the average of variances of P bootstrap set sampled with replacement from the initial set L. We denoted this estimated value as $\hat{\sigma}_{emb}^2(x)$

Given $f_{emb}(x)$ and $\hat{\sigma}_{emb}^2(x)$, a new model is trained on the same dataset to estimate $\sigma_\epsilon^2(x)$ directly in quantity $\hat{\sigma}_\epsilon^2(x)$ using maximum likelihood [16] with the assumption that $\epsilon(x) \sim \mathcal{N}(0, \sigma_\epsilon^2(x))$. For observed residual r, the loss function is therefore the negative probability density function of $\epsilon(x)$, as follow

$$E_{\sigma_\epsilon^2} = -\sum_{i=1}^{N} log(\frac{1}{\sqrt{2\pi\hat{\sigma}_\epsilon^2(x_i)}} exp(-\frac{r_i^2}{2\hat{\sigma}_\epsilon^2(x_i)})) \tag{5}$$

where $r_i^2 = (y_i - f_{emb}(x_i))^2 - \hat{\sigma}_{emb}^2(x_i)$, in which we need to remove the uncertainty generated by the estimated true regression. Similar to maximum likelihood method, ensemble method constructs prediction intervals using the estimated total prediction variance by summing $\hat{\sigma}_{emb}^2(x)$ and $\hat{\sigma}_\epsilon^2(x)$ and the estimated true regression $f_{emb}(x)$. For our implementation, we used $M = 200, M_2 = 8, P = 1000$ just like the original paper. We applied the log operation to cancel out the exponential term in (5) before we used it as a loss function to prevent exploding gradients.

2.3 Quantile Regression Method

Quantile regression [13, 14] is similar to the ordinary least squares regression, however unlike the ordinary least square regression that estimates the mean of conditional target distribution, quantile regression estimates the τ quantile of the conditional target distribution. For example, the 50th quantile of a target distribution is the condition median of that target distribution, which is above 50% of the target values. To train a neural network that estimate the τ quantile of the target distribution, quantile regression uses the loss function E_τ according to

$$E_\tau = \sum_{i=1}^{N} L_\tau(y(x_i) - q_\tau(x_i)) \quad \text{where} \quad L_\tau(e_i) = \begin{cases} \tau e_i & \text{if } e_i \geq 0 \\ (\tau - 1)e_i & \text{otherwise} \end{cases} \tag{6}$$

where the output of the model is $q_\tau(x)$. Given a desired quantile range from τ_l to τ_u, we construct prediction interval by training two quantile regression models one for the lower bound τ_l and one for the upper bound τ_u. For our implementation, we use a single neural network that output both the lower and upper quantile regressions.

A naive heuristic to set the hyper-parameters τ_l and τ_u would be, for example, to use 10% and 90% to capture 80% of target coverage. Unfortunately, our experiment show that this produces sub-optimal results. Instead we set τ_l and τ_u via exhaustive grid search at training time.

2.4 Fixed Bounds Method

As naive baseline, we include a fixed bounds method in our experiments. One can view the output of this model as a ceiling, or the score one could trivially get without using any technique. This technique takes a model that is trained to predict a regression ($f(x)$) and creates prediction intervals that are plus or minus a fixed percentage (α) of the target value, yielding:

$$[(1 - \alpha)f(x), (1 + \alpha)f(x)] \tag{7}$$

3 Assessment Metrics

Assessing the quality of a prediction interval can be difficult because there are two competing objectives. One wants intervals that are both *correct* the target amount of the time and *tight*, having a

narrow mean width. These two objectives were formalized in [18] as prediction interval coverage probability (PICP) and mean prediction interval width (MPI [18] or MPIW [9, 10, 12]).

Prediction interval coverage probability (PICP) representing the percentage of the time the prediction interval is correct

$$\text{PICP}_{l(x),u(x)} = \frac{1}{N}\sum_{i=1}^{N} h_i \quad \text{where} \quad h_i = \begin{cases} 1 & \text{if } l(x_i) \leq y_i \leq u(x_i) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where N is the size of the test set. $l(x_i)$ and $u(x_i)$ are the lower and upper bounds of the prediction interval for sample i, and y_i is the observed target value. Mean prediction interval width (MPIW), measures the average size of all prediction intervals.

$$\text{MPIW}_{l(x),u(x)} = \frac{1}{N}\sum_{i=1}^{N} |u(x_i) - l(x_i)| \tag{9}$$

Given Target T, one wants to minimize MPIW, while maintaining PICP $\geq T$. Some literature discards cases where PICP $< T$ as invalid because they fail to meet the correctness criteria of the problem. [11] propose a combined metric that applies an exponentially exploding penalty to cases that do not meet the PICP target. In practice, if the penalty is sufficiently large, this eliminates incorrect intervals, but one can construct still pathological datasets to defeat this combined metric. Another way to address this issue, which we use in evaluation (see section 5.3), linearly scales the output of each technique's output by a constant factor to make it hit the target PICP exactly.

4 Expanded Interval Minimization

To motivate the design of Expanded Interval Minimization (EIM), imagine how one would find the parameters (θ) of a neural network to output a prediction interval lower bound $l(x;\theta)$, and an upper bound $u(x;\theta)$ that optimizes the objective directly:

$$\theta_{min} = \underset{\theta}{\text{argmin}}\, \text{MPIW}_{l(x;\theta),u(x;\theta)} \quad \text{subject to} \quad \text{PICP}_{l(x;\theta),u(x;\theta)} = T \tag{10}$$

This formulation would be desirable, but it is difficult in practice for two reasons. First, it is expensive because it requires a global calculation and second it is difficult to optimize directly or extract a gradient from.

A critical insight is that one can use the PICP and MPIW of each minibatch as a noisy estimate of the population PICP and MPIW. Calculating MPIW in a minibatch is straight-forward, but the minibatch PICP is unlikely to match the target T during training. We fix this issue by applying automatic scaling of the bounds so they cover the target PICP on the minibatch. For each minibatch (B), we calculate a scaling factor k_B to expand or shrink the predicted bounds such that the minibatch PICP equal T throughout the training. The EIM loss function (E) for minibatch B, is as follows:

$$E_B = k_B \sum_{i \in B} |u(x_i) - l(x_i)| \tag{11}$$

Here we present a closed form solution to calculate k'_B, the scaling required to hit the PICP target on the minibatch. k'_B is a simplified version of the full k_B which will be introduce later. First, we to calculate a set of minimum scaling factors $\{k_i\}_{i=1}^{|B|}$ for all instance in the minibatch such that each scaled output bound is just wide enough to capture the target value y_i, according to:

$$k_i = \left| \frac{u(x_i) + l(x_i) - 2y_i}{u(x_i) - l(x_i)} \right| \tag{12}$$

Then we select the Tth percentile[1] scaling factor from $\{k_i\}_{i=1}^{|B|}$ to scale all the output bounds in minibatch B. This will make the output bounds in the minibatch achieve exactly T PICP.

$$k'_B = \sum_{i \in B} c_i k_i \quad \text{where} \quad c_i = \begin{cases} 1 & \text{if } k_i \text{ is the } T\text{th percentile of } B \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

[1] "Tth percentile" represents selecting the value T percent of the way through the set when sorted by value. In this context we interpret T as a percentage between 0 and 100, while in other places it is 0 to 1.

While the above loss function works and directly represents our primary objective, it produces suboptimal results since only a single value from the minibatch receive nonzero gradients from the loss function. An improved version of the above loss function selects an average of multiples k_i values that are within δ (we use δ from 1% to 3%) of the Tth percentile:

$$k_B = \frac{\sum_{i \in B} c_i k_i}{\sum_{i \in B} c_i} \quad \text{where} \quad c_i = \begin{cases} 1 & \text{if } k_i \text{ is within } \delta \text{ of the } T\text{th percentile of } B \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

4.1 Training Procedure and Testing

We have found that to get the best results training EIM models that one should use larger minibatches than other techniques. Since EIM uses the minibatch to estimate its coverage, larger minibatches produce more stable estimates that improve convergence and performance. Another procedure we found profitable was to pretrain EIM on fixed bounds. This pretraining uses mean squared error loss on the target value plus and minus a constant on a minibatch B, with the following pretraining loss function:

$$E'_B = \sum_{i \in B} (y_i - \alpha - l(x_i))^2 + \sum_{i \in B} (y_i + \alpha - u(x_i))^2 \tag{15}$$

Where y_i is target value, $[l(x_i), u(x_i)]$ is the predicted range, and α is a constant depends on the dataset. This pretraining was used mainly for preventing early divergence when training with the EIM loss. To use EIM in a production environment the output bounds from the trained model should be scaled about their center in a similar process to how k_b is computed. After training the model, we use a holdout set to compute the population scaling factor k required to hit the PICP target. We then grow or shrink all future model outputs by this constant factor k (see section (5.3)).

5 Evaluation Datasets

Property	Domain Valuation Dataset	Million Song Dataset
Size	634,328	515,345
Target Range	\$0-\$25,000	1922-2011
Target Mean	\$1,971.2	1998.4
Target Median	\$1,000.0	2002
Target Standard Deviation	\$3,379.8	10.93
Number of Attributes	269	90
Base Model R^2	56.83	20.67

Figure 1: Properties of each dataset.

This section describes the two real world datasets used to evaluate EIM. Figure 1 shows some summary statistics about each of these datasets.

5.1 Aftermarket Domain Name Value Dataset

For our first dataset, we use a set of aftermarket sale prices of domain names. The goal of our models is, given a domain name that has sold in the past, provide prediction intervals on the prices that it sold at. This is a real word data set that is being used to build models available to millions of customers. A challenge in working with domain names is tokenizing them into words because domains do not contain the spaces found in most text. We built a domain name tokenizer based on a vector embedding of words [15] language model that estimates the probability of every possible tokenization. We built a training set for this model by extracting the tokenization of a domain name from its crawled website content. This language model for domains is also used as an input to our neural network, as we find the way people use words in domain names differs from how they use them in other text.

Our model also pulls in external data and other features as inputs to the neural network. These additional inputs include:

- For each top-level domain (TLD) with enough historical sale data, we create a vector embedding. The TLD is one of the most important features.
- The usage of other TLDs of the domain (e.g., when looking at $foo.com$, we inspect $foo.net$ and others to see which hosting provider, if any, serves it.).

- Domain statistics from other aftermarket datasets, just as active listings and expiry auctions.
- Word matches and statistics from various dictionaries, including English, English part of speech, French, German, Italian, Spanish, Japanese, names, first names, last names, female names, male names, places, acronyms, brands, products, adult-related, currencies, phrases, countries, and Wikipedia.
- Other handcrafted features to identify specific non-word based patterns of interest to domain investors [8].

5.2 Year Prediction Million Song Dataset

The second dataset we used is a published dataset [4] that is a subset of the Million Song Dataset [1]. This dataset is used for building regression model that predict the release year of a song from audio features. Each instance has 90 attributes which consist of 12 timbre average and 78 timbre covariance encoded as floats.

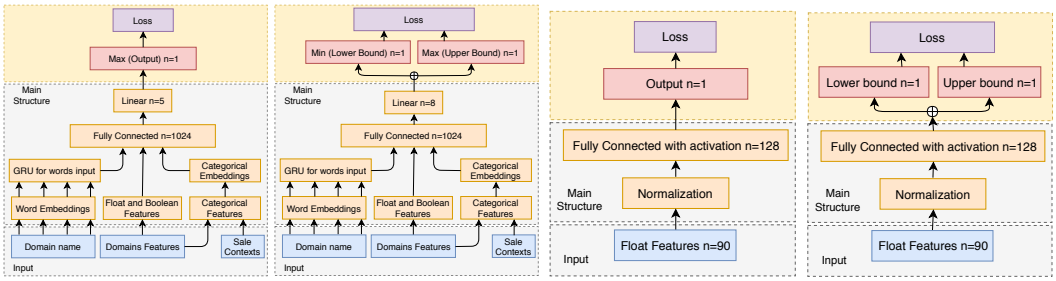
5.3 Scaled Intervals

Some of the models we compare against are not able to customize themselves to a specific target correctness rate, or may be under or over the target PICP. To provide a fair playing field and avoid comparing models with different PICP rates, we scale the output of each model so it exactly hits the target ($PICP = T$). To do this, we define a new interval $[u'(x_i), l'(x_i)]$ based on the raw interval $[u(x_i), l(x_i)]$:

$$\begin{aligned} u'(x_i) &= \frac{u(x_i) + l(x_i) + k|u(x_i) - l(x_i)|}{2} \\ l'(x_i) &= \frac{u(x_i) + l(x_i) - k|u(x_i) - l(x_i)|}{2} \end{aligned} \tag{16}$$

where k is a constant scaling factor. Note that if $k = 1$, then $[u'(x_i), l'(x_i)] = [u(x_i), l(x_i)]$. If $k < 1$, the intervals are shrunk and if $k > 1$ they are expanded in a linear way. We compute the k required to bring each model into exact PICP compliance.

5.4 Dataset Specific Neural Network Structures



(a) Single Output + Domain Valuation (b) Dual Output + Domain Valuation (c) Single Output + Million Song (d) Dual Output + Million Song

Figure 2: Neural network structures for each dataset. Figures 2(a) and 2(c) show the structures used by the techniques fixed bounds, MLE, and ensemble. Figures 2(b) and 2(d) show the structures used by EIM and quantile regression.

To compare the performance of different methods on the domain valuation dataset and million song dataset, we first implemented the regression models that can output a single target value given the input features. Figures 2(a) and figures 2(c) show the single output neural network structures for both domain valuation dataset and million song dataset respectively. Note that Figures 2(a) is a more complex model because of the features introduced in section 5.1. Figure 2(b) and figure 2(d) show the neural network structures based on the single neural network structure but modified to output both lower and upper bounds predictions.

Method	PICP=70%	PICP=80%	PICP=90%
MLE	2488	2735	3689
Ensemble	1908	2480	3437
EIM 70	**1185**	1889	3715
EIM 80	1452	**1692**	3076
EIM 90	2183	2362	**2756**
Quantile 70	1804	2509	4802
Quantile 80	1969	2402	3519
Quantile 90	2108	2562	3485
Fixed bounds	2642	3077	3654

(a) MPIW for Domain Valuation Dataset

Method	PICP=70%	PICP=80%	PICP=90%
MLE	15.50	18.93	27.37
Ensemble	16.92	19.91	25.74
EIM 70	**11.54**	16.53	32.30
EIM 80	12.18	**15.00**	26.23
EIM 90	14.81	17.35	**21.32**
Quantile 70	12.59	16.40	27.39
Quantile 80	12.70	15.99	25.47
Quantile 90	14.80	17.50	22.46
Fixed Bounds	23.57	26.20	28.96

(b) MPIW for Million Song Dataset

Figure 3: Mean prediction interval width (MPIW) at 70%, 80%, and 90% prediction interval coverage percent (PICP) for all techniques and both datasets. EIM and quantile regression are parameterized by each target PICP, where EIM 80 indicates EIM trained to hit a PICP=80%. Lower is better and bold values are the best found.

For our implementations, maximum likelihood method used two networks to predict the target value and the total prediction variance respectively. Ensemble method used 200 networks to obtain the mean prediction and the model uncertainty variance, and trained an additional network to predict the data noise variance. Both quantile regression method and EIM method used only one network to output both lower and upper bounds and we built 3 different models for 3 different PICP targets 70%, 80% and 90%. For EIM, we can specify different PICP targets in the loss function for different models; for quantile regression method, we run the exhaustive grid search to find the best possible quantiles range (τ_l, τ_u) for different PICP targets.

6 Results and Discussion

Figure 3 shows that EIM produces significantly tighter prediction interval than other techniques for both the domain name valuation dataset and for the million song dataset. EIM produces $1.33x$ to $2.23x$ tighter bounds than fixed bounds method, $1.26x$ to $2.1x$ tighter bounds than MLE, $1.21x$ to $1.61x$ tighter bounds than ensemble, and $1.06x$ to $1.52x$ tighter bounds than quantile regression. Interestingly the gap between EIM and the next best technique decreases with higher PICP targets, going from $1.26x$ tighter prediction intervals for a 70% coverage target to $1.06x$ a 90% target. We believe that these lower targets provide EIM more flexibility in the choice of generated ranges which it is better able to utilize than other techniques.

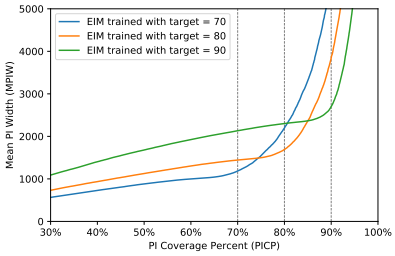
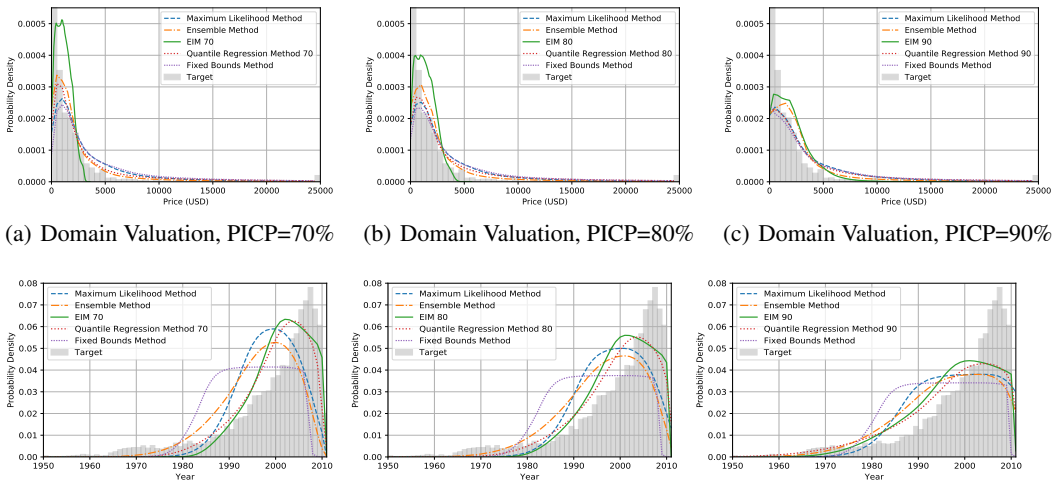


Figure 4: MPIW for the versions of EIM trained for one PICP target scaled to hit other PICP targets on the domain valuation dataset.

Figure 3 also shows that techniques that train for the specific PICP targets (EIM and quantile regression) have a significant advantage over the techniques that only support symmetric bounds (fixed, MLE, and ensemble). One can note that the versions of these techniques with training targets matching the measured target perform better than those trained with other targets. Figure 4 expands on this finding for EIM by showing how MPIW changes as one scales versions EIM trained for one PICP target to other PICP targets on the domain valuation dataset. We can see that each version of EIM has specialized itself for its specific target.



(a) Domain Valuation, PICP=70% (b) Domain Valuation, PICP=80% (c) Domain Valuation, PICP=90%

(d) Million Song, PICP=70% (e) Million Song, PICP=80% (f) Million Song, PICP=90%

Figure 5: Prediction and target distribution for each target and dataset. Note that the first bar in (a)-(c) is 0.00068, and was truncated for clarity in showing the predicted distributions.

Figure 5 shows the prediction and target distribution for each target and dataset. The target distribution is a histogram of actual values in the holdout set. The lines for each technique show the normalized probability density corresponding to how often the predicted range contains each point.

Looking at Figure 5(d-f) one notices that the predicted distributions for EIM and quantile regression are skewed more towards where the majority of the data is compared to the other techniques. The other methods construct their bounds by starting at the predicted mean and expanding equally in both directions. EIM, on the other hand, can strategically expand bounds more in the direction with more density of data. For Figure 5(a-c), the data distribution is more concentrated at the lower values, and the prediction distribution of EIM stands out from all the other techniques in that it is able to change its distribution more dramatically between the different target percentiles.

Comparing the three target percentages in Figure 5 one sees that the techniques need to spread out their prediction distributions to achieve higher coverage percents. One also sees a shift in the mean which is more pronounced for EIM. We see larger differences between techniques for the lower PICPs target and all the techniques cluster closer together at the higher targets.

Finally, we should mention the different complexity of the top three techniques: EIM, quantile, and ensemble. EIM is by far the fastest and most straightforward of them, requiring training only a single neural network. Ensemble is the most complex and slower, requiring training 200 neural networks and using sophisticated but challenging to implement methods to combine them. Basic quantile regression has a certain elegance to it but contains two hyper-parameters which must be set with exhaustive grid search to achieve optimal results. This grid search made the implementation more complicated and made quantile regression the slowest technique to train. The resulting values for lower and upper quantiles found by the grid search were non-obvious. For example, for the million song dataset they were (0.2, 0.65) for PICP=70%, (0.2, 0.75) for PICP=80%, and (0.5, 0.75) PICP=90%. When scaled to hit the coverage, these outperformed the more naive symmetric choices.

7 Conclusions

This paper presented Expanded Interval Minimization (EIM), a novel technique for generating prediction intervals with neural networks. We showed that compared to the next best technique, EIM produces $1.26x$ tighter prediction intervals for a 70% coverage target, $1.21x$ tighter prediction intervals for a 80% target, and $1.06x$ tighter intervals for a 90% coverage target. EIM is a natural fit for any application using prediction intervals and having asymmetrically distributed error. Using EIM, we hope that others will be able to generate tighter prediction intervals and advance the state of the art of what machines can do with deep learning.

References

[1] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[2] J. G. Carney, P. Cunningham, and U. Bhagwan. Confidence and prediction intervals for neural network ensembles. In *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, volume 2, pages 1215–1218. IEEE, 1999.

[3] R. D. De VlEAUX, J. Schumi, J. Schweinsberg, and L. H. Ungar. Prediction intervals for neural networks via nonlinear regression. *Technometrics*, 40(4):273–282, 1998.

[4] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017.

[5] A.-M. Halberg, K. H. Teigen, and K. I. Fostervold. Maximum vs. minimum values: Preferences of speakers and listeners for upper and lower limit estimates. *Acta psychologica*, 132(3):228–239, 2009.

[6] T. Heskes. Practical confidence and prediction intervals. In *Advances in neural information processing systems*, pages 176–182, 1997.

[7] J. G. Hwang and A. A. Ding. Prediction intervals for artificial neural networks. *Journal of the American Statistical Association*, 92(438):748–757, 1997.

[8] J. Iles. The origin of chips: Why do we use this term? https://www.namepros.com/blog/the-origin-of-chips-why-do-we-use-this-term.899340/, 2015.

[9] A. Khosravi, S. Nahavandi, and D. Creighton. Construction of optimal prediction intervals for load forecasting problems. *IEEE Transactions on Power Systems*, 25(3):1496–1503, 2010.

[10] A. Khosravi, S. Nahavandi, and D. Creighton. A prediction interval-based approach to determine optimal structures of neural network metamodels. *Expert systems with applications*, 37(3):2377–2387, 2010.

[11] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya. Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE Transactions on Neural Networks*, 22(3):337–346, 2011a.

[12] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya. Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on neural networks*, 22(9):1341–1356, 2011b.

[13] R. Koenker and G. Bassett Jr. Regression quantiles. *Econometrica: journal of the Econometric Society*, pages 33–50, 1978.

[14] R. Koenker and K. F. Hallock. Quantile regression. *Journal of economic perspectives*, 15(4):143–156, 2001.

[15] A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273, 2013.

[16] D. A. Nix and A. S. Weigend. Estimating the mean and variance of the target probability distribution. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, volume 1, pages 55–60 vol.1, June 1994.

[17] D. A. Nix and A. S. Weigend. Learning local error bars for nonlinear regression. In *Advances in neural information processing systems*, pages 489–496, 1995.

[18] D. L. Shrestha and D. P. Solomatine. Machine learning approaches for estimation of prediction interval for the model output. *Neural Networks*, 19(2):225–235, 2006.

[19] A. Zapranis and E. Livanis. Prediction intervals for neural network models. In *Proceedings of the 9th WSEAS International Conference on Computers*, page 76. World Scientific and Engineering Academy and Society (WSEAS), 2005.