"A Multi-Horizon Quantile Recurrent Forecaster" - Paper Review

Elad Eatah

Introduction & Setup

The paper [4] was written by researchers from the "Forecasting Data Science" team and "Amazon AI Lab" of Amazon Inc, and presented on the 31st "Conference on Neural Information Processing Systems (NeurIPS 2017)".

This work deals with probabilistic multi-step, long-horizon forecasting of multivariate time-series. Formally, given t past values samples, this technique estimates consecutive $k > 1$ values, rather than a single future estimation. Unlike point-forecasting, where each future estimation is single value estimation, the shown technique estimates some arbitrary Q quantiles of the conditional-distribution the the k future-values, given t historical values. This Seq2Seq framework can incorporate any known information regarding any future events, such as seasonal-patterns or any static time-invariant feature. Consequently, this framework may handle multivariate time-series with long-term dependencies (as found in long seasonal patterns), which may involve some known information about the future values.

This framework is named *Multi-horizon Quantile RNN* (MQ-RNN). The authors then propose variants for the encoder of the MQ-RNN architecture, and compare their performance against the MQ-RNN.

Advantages of The Framework

- The usage of quantiles may be more relevant to some applications instead of exact parametric description of the conditional-distribution, or even the expectation of this conditional-distribution. An example for such application is risk-management for stock prices estimation. While a possible decrease in a stock's price is surely a risk whose cost could be high, an increase of its price may not be viewed as a risk at all. Thus the symmetric mean-squared error of these estimation do not reflect the true risk of the estimation. Instead, this paper utilizes the quantile-loss function.
- The estimation of quantiles requires no assumptions at all regarding the conditional probability (for instance, being Gaussian).
- This Seq2Seq framework deals with a problem, which received little attention by then (according to the authors): accounting for any known information regarding any future events.

Framework Architecture

The general Seq2Seq framework is an Encoder-Decoder structure. The encoder is a vanilla-LSTM RNN, with as many recurrent layers as the number of input time-steps, whereas the decoder consists

of two Multi-Layer Perceptrons (MLP). This architecture is illustrated in Figure 1b in [4].

The first MLP (a.k.a the global MLP) receives the encode output, along with all the given future information, and generates a collection of contexts. The first context is time-independent, which is later utilized for all requested horizons. The other contexts are time-specific (a single context for each requested horizon). The second MLP (a.k.a the local MLP) creates, for each horizon, all the requested quantiles estimations, using the time-independent context, the specific context for this horizon and the available future information for this horizon. The parameters of the local MLP are shared among all requested horizons. This MLP allows alighning with future seasonality and events, and possible sudden-spikes forecasts. Suppose no future information is known, or spiky forecast are not desired, the authors suggest on removing the local MLP (and the time-independent context), and simplifying the global MLP to produce the quantiles estimations directly. This alternative retains all other advantages.

Training Scheme & Loss Function

The training is performed w.r.t the quantile-loss. Formally, let y_t and $\hat{y}_t^{(q)}$ be a ground-truth value and an estimation of its q-quantile, respectively. Then the quantile-loss is defined by

$$L_q\left(y_t, \hat{y}_t^{(q)}\right) := q\left(y_t - \hat{y}_t^{(q)}\right)_+ + (1-q)\left(\hat{y}_t^{(q)} - y_t\right)_+, \tag{1}$$

and the total-loss between a ground-truth sequence $\{y_t\}_t$ and the estimated-quantiles $\left\{\hat{y}_t^{(q)}\right\}_{t,q}$ is defined by

$$\sum_t \sum_q \sum_k L_q\left(y_{t+k}, \hat{y}_{t+k}^{(q)}\right), \tag{2}$$

where the summation is performed over all time-steps, all forecast horizon and all requested quantiles.

Being an RNN, this model can be trained efficiently, using the forking-sequences scheme. For each recurrent layer in the encode, a decoder is placed on top of it. Then, for every input sequence, the quantiles estimations from all decoders are gathered, and the total-loss w.r.t the input sequence. Afterwards, a single backpropagation-through-time collects the multi-horizon error gradients, using a single pass for each sequence sample (and a small additional computational cost). Since these gradients are updated simultaneously, the learning remains stable, despite the fact that consecutive forecast creation times (FCT) are highly-correlated. Each training sequence serves as a single training sample. Using this method, there is no need for splitting sequences or augmentation of the dataset, and thus the training time is reduced, comparing to such training methods.

Encoder Alternatives

The authors imply that the simple vanilla-LSTM might not always be favorable. This encoder might suffer from memory-loss during recurrent forward propagation for sequences with long periodicity (for example, annual events). Therefore they suggest replacing the encode with either of the following alternatives:

1. A **NARX-RNN** (as in [2]) - Here each hidden-state in the model is computed, using not only the previous hidden-state, but some former hidden-states as well. These connection are known as skip-connections. In practice, these connections can be implemented by placing an extra linear layer on top of the LSTM. Equivalently, one may just feed these past hidden-states as input, in addition to the existing inputs.

2. **Any NN with sequential or temporal structure** - The authors suggest, for instance, the *WaveNet* CNN in [3] . The convolutions in this CNN have stride 1 and the local structure is step-invariant, this model is compatible with the forking-sequences scheme. Hence it can be integrated easily as the encoder. The authors name such alternatives a **MQ-CNN** network.

A closely-related SOTA Solution - Seq2SeqC

A very similar approach was already suggested by [1], and is known as *Seq2SeqC* (Seq2Seq with Context). This encoder-decoder architecture uses a vanilla-LSTM for both the encoder (as in MQ-RNN) and the decoder (in contrast to the two MLPs of MQ-RNN). Seq2SeqC contains only a single time-independent context, while MQ-RNN utilizes time-specific contexts as well. Horizon awareness in the decoder is achieved, instead, by recursively feeding predictions for the LSTM cell.

Rather then minimizing the estimation error, the training for this model is performed by maximizing the log-likelihood of the conditional-distribution, rather then minimizing the estimation error (for numeric values, for example, this distribution is assumed to be log-Gaussian).

Performance Comparison

The paper compares the performance of various models, w.r.t some metrics (the datasets are described in details in Section 4 in the paper). First, a comparison is performed between MQ-RNN, MQ-RNN-Cut (the same model, where training is performing by cutting and splitting input sequences), ML-RNN (analougous to MQ-RNN, where training is performed by maximizing log-Gaussian likelihood and the prediction is replaced by prediction of the distribution mean and variance) and Seq2SeqC (as in [1]). The training time of these models are first compared, against the number of performed epochs, with quantiles $\{0.1, 0.5, 0.9\}$. The training loss curve of the MQ-RNN flattens more quickly of the MQ-RNN-Cut.

Then the testing loss for these models is compared, as a function of the time-step, w.r.t each of the quantiles separately. In all these cases, MQ-RNN outperforms all other models.

An additional comparison between MQ-RNN and ML-RNN is made w.r.t the calibration metric

$$\mathbb{E}\left(I\left(y_{t+k} \leq \hat{y}_{t+k}^{(q)}\right)\right), \tag{3}$$

for each $q \in \{0.1, 0.5, 0.9\}$ and the sharpness metric

$$\mathbb{E}\left|\hat{y}_{t+k}^{(0.9)} - \hat{y}_{t+k}^{(0.1)}\right|, \tag{4}$$

for all time-steps t and horizons k. The presented values are in percents, after dividing these metrics by the sharpness of MQ-RNN. When the calibration for the quantile q is perfect, the metric should be equal to q percents. MQ-RNN is closer to perfect calibration than ML-RNN, making it preferable.

The last comparison is conducted between the variants of MQ-RNN, where the encoder is replaced by any of the aforementioned alternatives. The compared metric is the total sum of testing loss, for quantiles $q \in \{0.1, 0.5, 0.9\}$. Here the MQ-CNN (i.e the model with WaveNet as the encoder) outperforms all other alternatives.

Guidelines for Prototyping

The current objective is creating a prototype of this model in a very short time. In order to finish this task on-time, the implemented variant of MQ-RNN will be the simplest of them all - the vanilla LSTM, for the encoder, and a single global MLP for the decoder. This simple model requires substantially less time for prototyping, on one hand, but at the expense of a few key-features of this model, on the other hand. Such features include support for long periodicity, because of the simple encoder, and the support for future information accounting which is achieved when both MLPs are used.

Another practical consideration is the run-time for both training the prototype model and its inference-time. Thus one must use a much smaller datasets than the ones used in that research.

As a short-cut, when the dataset for the training is created, no check for overlapping between training sequences and testing sequences will be performed. Consequently, some testing time-steps may be known to the model during training. This obviously leads to a bias in the test stage. However, assuming this dataset is large enough, comparing to the input sequences length, than the probability for this event is unlikely (though not necessarily negligible).

References

[1] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, *Learning phrase representations using rnn encoder-decoder for statistical machine translation*, 2014.

[2] Robert DiPietro, Christian Rupprecht, Nassir Navab, and Gregory D. Hager, *Analyzing and exploiting narx recurrent neural networks for long-term dependencies*, 2018.

[3] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, *Wavenet: A generative model for raw audio*, 2016.

[4] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka, *A multi-horizon quantile recurrent forecaster*, 2018.