source: https://www.pexels.com/photo/ask-blackboard-chalk-board-chalkboard-356079/

Probabilistic Machine Learning Series Post 1: Using Neural Networks as

part of a Bayesian model

Analytique Bourassa Aug 1, 2019 · 7 min read ★

This series will be about different experiments and examples in probabilistic machine learning. The advantages of probabilistic machine learning is that we will be able to provide probabilistic predictions and that the we can separate the contributions from different parts of the model. In this first post, we will experiment using a neural network as part of a Bayesian model. This allows us to use the feature learning aspect of deep learning with the uncertainty estimates provided by the Bayesian framework. For those not familiar with the Bayesian framework, the first chapter of Probabilistic Programming and Bayesian Methods for Hackers is suggested. In a nutshell, in the Bayesian framework, probabilities are seen as a degree of belief based on prior knowledge. The net result being that we will see the data as fixed and the parameters as random variables. Because of this, the parameters of our model will be represented by distributions. In comparison, in the frequentist framework, the parameters are fixed, but the data is random. The confidence interval representing the expected result of different samples. We will evaluate the posterior $P(\theta|y)$ using numerical methods. The posterior is proportional to the likelihood $P(y|\theta)$ times the prior $P(\theta)$.

$$P(\theta|y) \propto P(y|\theta)P(\theta)$$

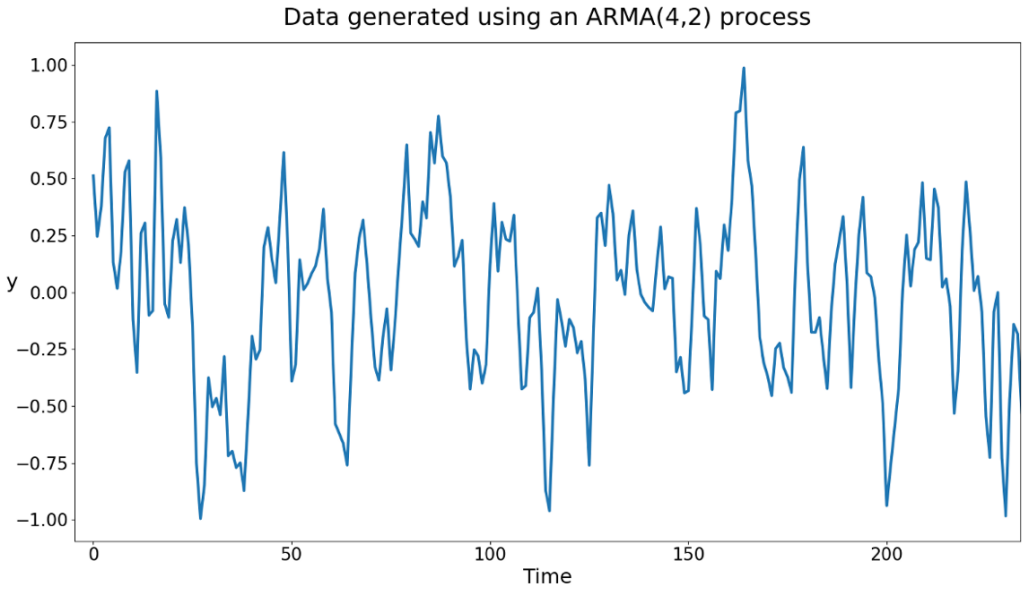
The posterior, left-hand side, is proportional to the likelihood times the prior.

For our experiment, the main assumption is to train a Neural Network, as a feature extractor, to be used in a Bayesian linear regression. The Neural Network architecture is chosen accordingly to the type of data. Our example will be about time series forecasting so we will use a LSTM (Long Short Term Memory) neural network since it will be able to extract time dependencies. The main advantage of this complete separation of the Neural Network from the Bayesian model is that a pretrained Neural Network giving good features can be used as is to make probabilistic predictions. One of the disadvantages is that we lose the regularization aspect of Bayesian Deep Learning for the Neural Net which need to be achieved otherwise.

Our goal is to give a probabilistic forecast to a time series generated using only a short term time dependence, i.e. ARMA(4,2). The time series in seperated into two parts. The first one is the autoregressive AR(4) which means that the next value depends linearly on the last four values. The second part is the moving average part MA(2) which mean that the next value will also depend on the last two noise values. Those two parts combined constitute the ARMA(4,2) process. The LSTM will identify the structure in the time series while the Bayesian model will provided the probabilist estimates. In a first step, we will train a LSTM with a linear last layer which will mimic the Bayesian linear regression. Afterwards, we will include the LSTM as a feature extractor in our Bayesian model. We will describe the full model in the third section. The next three sections will be about

- Training the LSTM,
- Describing the Bayesian model and
- Making probabilistic predictions

but let's first take a look at the data generated.

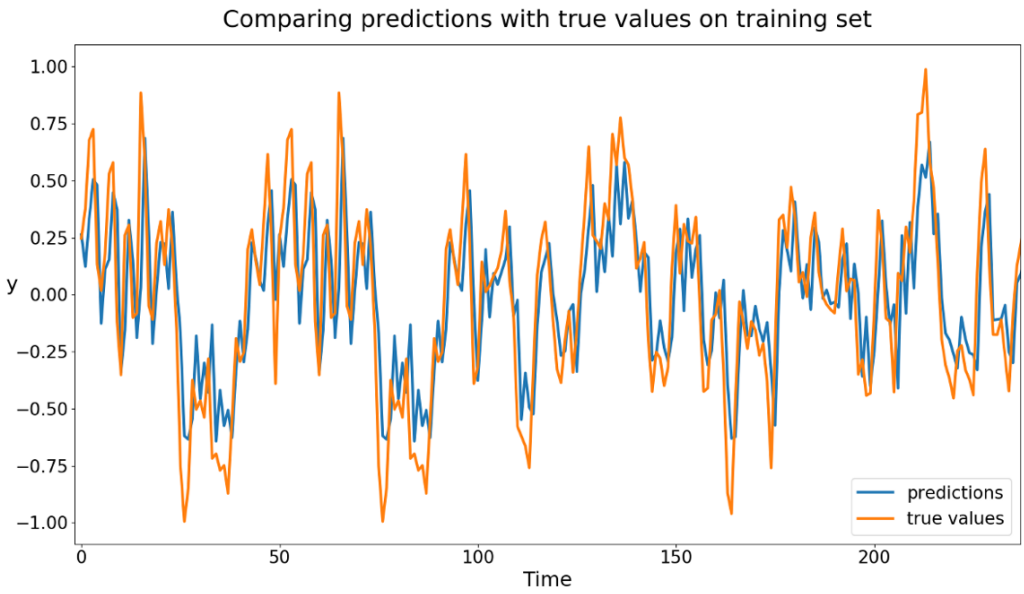


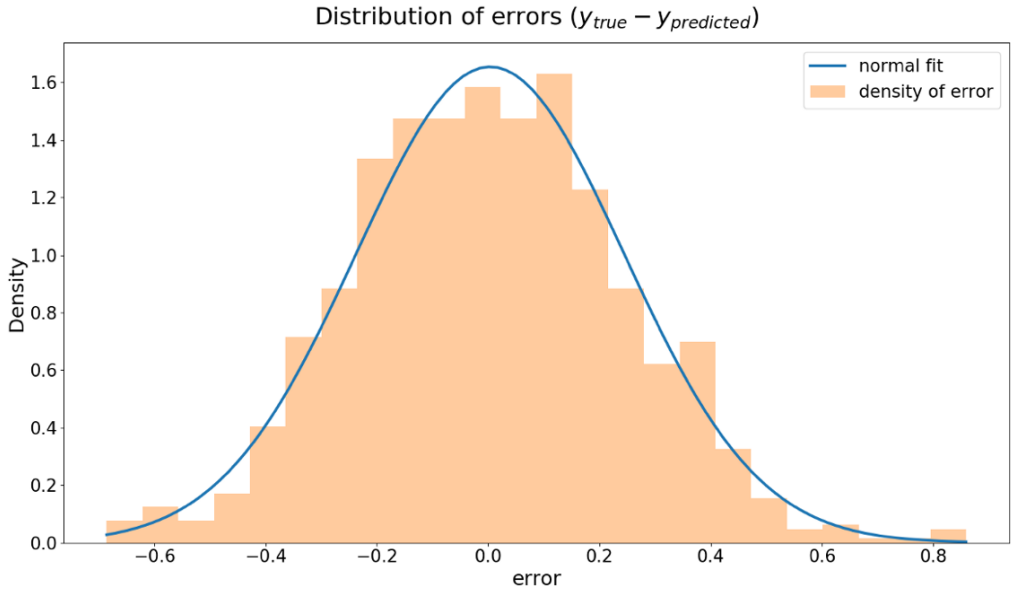
Since the process is ARMA (4,2), we have only a short term dependency, no seasonality (nor cycles) and no trend. If we had those two components, further preprocessing would have been needed, but we want to stay in the happy path for this example. Before moving forward, I want to mention that the LSTM was implemented in PyTorch and the Bayesian model was implemented in PyMC3.

Step 1: Training the LSTM

The LSTM was trained with sequences of seven time steps with the mean squared error loss. We used early stopping to prevent overfitting. We can see in the next figures that our predictions

on the training set are close to the true values and that the errors can be well fitted using the Normal distribution.





We now have an accurate predictor for our time series that gives only point-wise predictions. In the next section, we will include it in a Bayesian model to obtain probabilistic predictions.

Step 2: The Bayesian model

First of all, we will look at a graphical representation of our model. White circles are stochastic nodes, shaded circle are observations and square node are the results of deterministic transformations. The full arrows points to the parameters of the stochastic node and the dashed line are deterministic transformations. We will start at y which is the value that we want to predict.

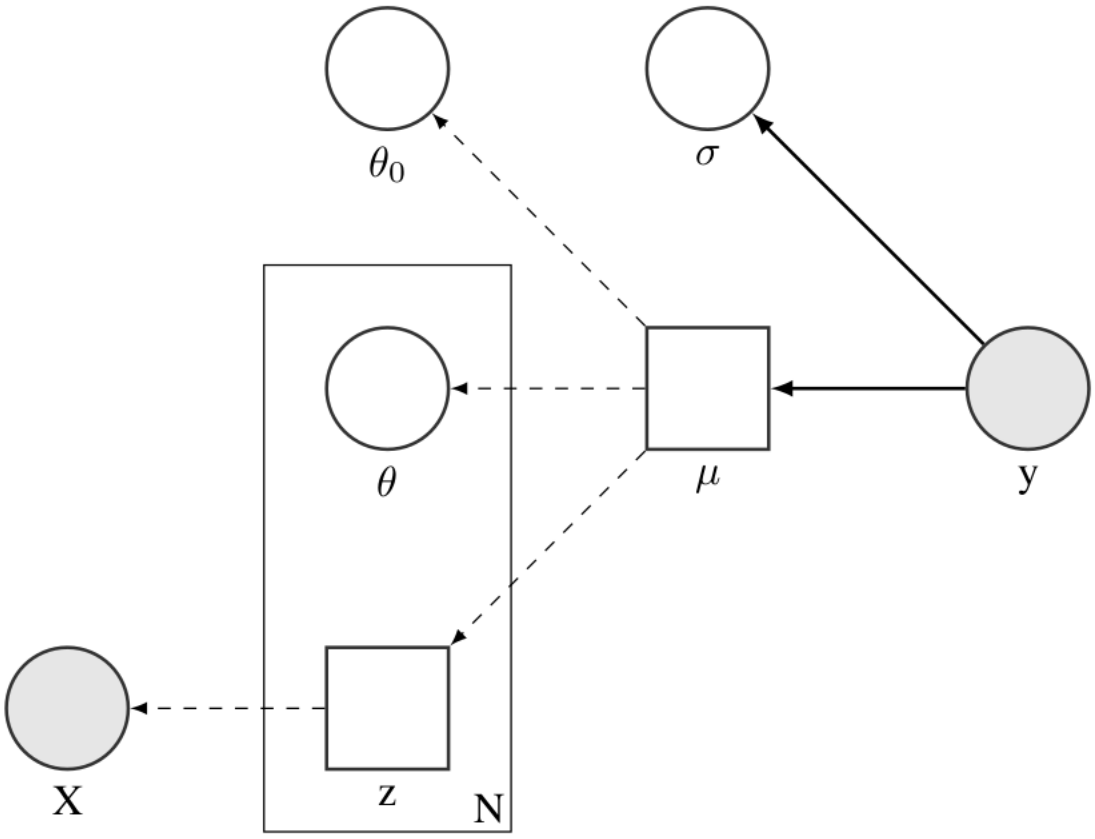


Figure 3: Graphical representation of our Bayesian model

We suppose that y follows a Normal(μ, σ). μ is the equivalent of the predicted value of our LSTM which is defined by

$$n$$

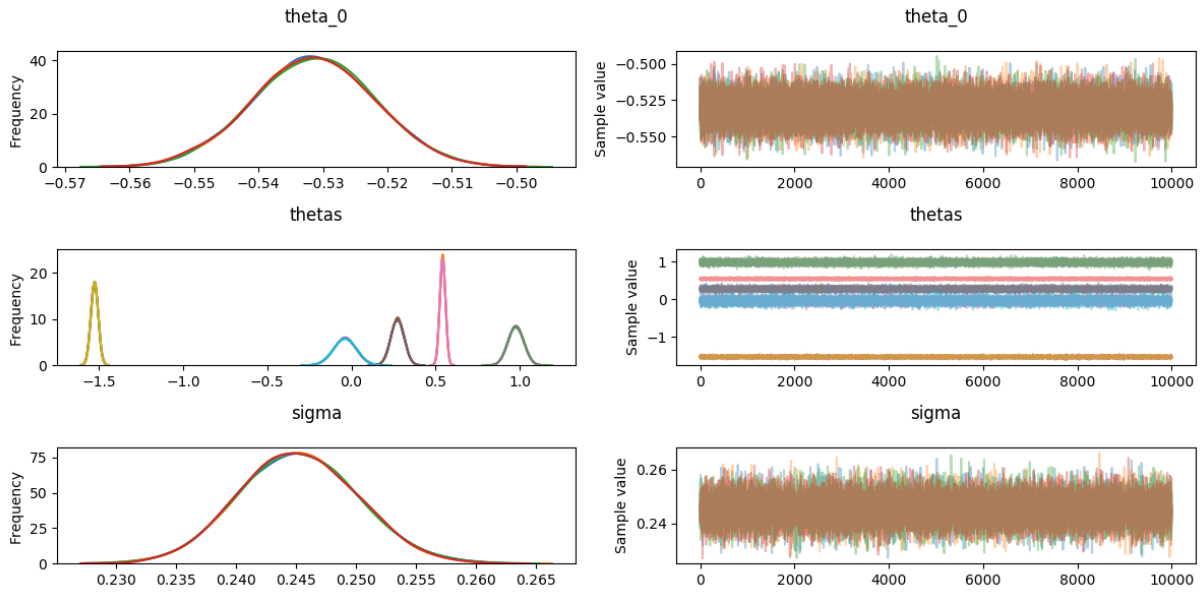
$$\mu = \sum_{i=1} z_i \theta_i + \theta_0$$

where z_i are the last hidden state of our LSTM, θ_i the weights of the linear layer, θ_o is the bias and n is the number of last hidden states of the LSTM. The main difference between the last layer of the LSTM model and the Bayesian model is that the weights and bias will be represented by a Normal distribution instead of being point estimates (i.e. single values). σ is the random error of the predictor that could not be captured by the uncertainty on the weights. At this point, z can be considered like a deterministic transformation done by the LSTM of the observed data. Since the uncertainty of the weights will not depend on the specific data values themselves, we only characterise the model uncertainty. The steps to include the LSTM in our Bayesian model are as follows:

- We trained the LSTM with a linear last layer
- We remove the last layer and used the LSTM as a feature extractor
- We replaced the original linear layer by a Bayesian linear regression

Now that we know all the parameters of our model that we want to evaluate, let's look at their distributions obtained using ADVI (Automatic Differentiation Variational Inference) and some fine tuning with MCMC (Markov Chain Monte Carlo).

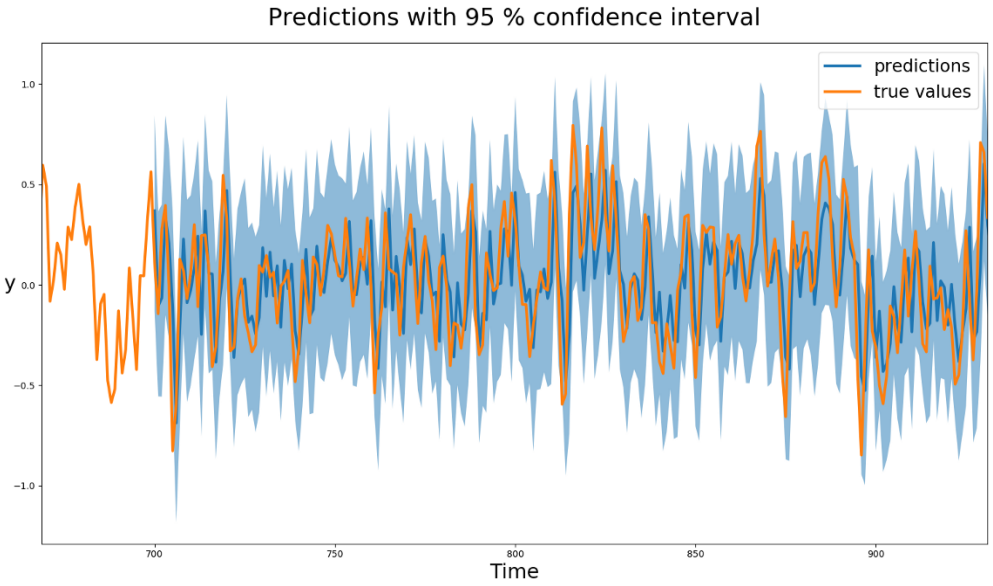
Those inference numerical methods were combined by using the ADVI posterior as the MCMC prior which can be easily done in PyMC3. The ADVI method has the advantage of being scalable, but only gives an approximate posterior. The MCMC method is slower but converges to the exact posterior. In a further post, we will examine those in more details.



In this step, we have used the trained LSTM as part of a Bayesian model. We have obtained distributions for the parameters which were point estimates in the previous step. We are now ready to make probabilistic predictions.

Step 3: Making probabilistic forecasts

The forecasts are made using the Posterior Predictive Checks (i.e. the parameters of the model given the features are sampled to obtain the probabilistic forecast). We can see in the next figure the results with the 95% confidence interval. We can notice that most predictions are close to the true value and that most predictions fall in the confidence interval.



Time series predictions (blue) with 95% confidence interval (shaded blue) and the true values (orange)

Let's do a small recap. We trained at LSTM on a time series to obtain accurate prediction. We could have stopped there, but we wanted a probabilistic prediction. Because of this, we used the LSTM as a feature extractor to be used in a Bayesian model. Since the Bayesian model parameters are represented by distributions, we could characterise the model uncertainty. The Bayesian model is then used to make probabilistic prediction using the posterior predictive checks. A lot of ideas are put together in this post. As stated earlier, this is part of a series of posts on probabilistic modeling so we will tackle some of the pieces individually in further posts.

Thanks for reading!

References and suggested readings:

[1] A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari and D. Rubin*, Bayesian Data Analysis (2013),* Chapman and Hall/CRC

[2] C. Davidson-Pilon, *Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Inference (2015)*, Addison-Wesley Professional

[3] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman and D. M. Blei, *Automatic Differentiation Variational Inference* (2016), ArXiv

Thanks to Mazid Osseni and Roxane Debruyker.

Machine Learning | Bayesian | Modeling | Deep Learning | Time Series Forecasting