A Bayesian LSTM Approach to Earthquake Prediction in CAT Bonds Pricing

Travaux d'Etude et de Recherche

Fournier Damien
Doumbia Ibrahima
Boussim Inès

Grande Ecole d'Actuariat et de Gestion des Risques
Institut de Science Financière et d'Assurance (ISFA)
M1 Actuariat

Supervisor: Couloumy Aurélien

16.03.2022

Abstract

With the advent of big data and the Internet of Things, machine learning and deep learning have become increasingly popular tools for their predictive abilities as well as their capacity to handle large volume of data. The insurance industry is no stranger to such tools which are already being used to help drive improvements in various areas such as property analysis, fraud detection or claims processing. However, as it is hard to gain a comprehensive understanding of their inner working after they have been trained, many ML systems — especially deep neural networks — are essentially considered black boxes, which can limit their usage in a prudential sector such as insurance. Bayesian statistics offer a formalism to understand and quantify the uncertainty associated with deep neural network predictions in the form of Bayesian Neural Networks. Thus, we propose to use Bayesian neural networks to the prediction of intermediate term earthquake in Japan using recurrent neural networks. We compare our results to density-based prediction methods and explore how Bayesian neural networks can be used to better asses the risk associated with a parametric CAT Bond, from an investor's perspective.

keywords: Bayesian Neural Networks, Deep Learning, Cat bond, Earthquake Prediction, LSTM

Contents

List of Figures

List of Tables

1. Introduction

On the 30th of March 2021, Palomar Holdings announced that, through its wholly-owned subsidiaries (Palomar Specialty Insurance Company and Palomar Excess and Surplus Insurance Company), the Company has successfully closed a $400 million 144A catastrophe bond GlobeNewswire (2021) which provide protection against earthquakes in the covered area over a three-year risk period. This is just one example among many of the high level of activity of the cat bond market since its inception in the mid-1990s.

The advent of Insurance Linked Securities(ILS) followed Cyclone Andrews and the Northridge Earthquake to which re-insurers were unable to honour their commitments. The search for an alternative form of risk transfer has led insurers to turn to the financial markets. Less than a year later the first issue of cat bonds was made. It was the Cat Bond Kover issued in March 1994 by Hannover Re. With a capacity of $ 85M, it protected against any type of natural disasters. In 2017, the amount of emissions has reached $ 71,243 million Djekadom (2016). Today, there are a multitude of forms of ILS, the best known of which are Mortality and Longevity Bonds and Cat Bonds. In the framework of our study we focus only on Cat bonds, specifically on how NTBs can contribute to better pricing of these ILSs.

What is a Cat bond?

CAT Bonds are inherently risky, multi-period, non-compensation-based transactions that pay investors regular coupons at the end of each period, and the final principal on the maturity date. 1.1 shows the structure of a CAT bond, including the flow of funds from one party to another. The issuer does not issue the CAT directly but uses a special purpose vehicle (SPV) for the transaction. SPVs can be interpreted as a focused insurer whose sole purpose is to provide one insurance policy. Their presence minimizes capital friction costs. In addition, a sufficiently high allocation to SPV eliminates counterparty risk. SPV reinsurance contracts with sponsors or counterparties (insurance companies, reinsurance companies, governments, etc.) by issuing CAT bonds to investors and receiving premiums from sponsors in return for providing a given coverage.
Therefore, the sponsor can transfer some of the risk to the investor. Investors take risks in exchange for higher return expectations. SPVs collect capital (principal and premium) and invest the proceeds in collateral accounts (usually escrow accounts that are closely related to short-term securities such as government bonds). Yields generated from collateral accounts are exchanged for variable yields based on London Inter-bank Offered Rate.

The investors coupon payments are made up of SPV investment returns, plus the premiums from the sponsor. If no trigger event occurs during the term time of the CAT bond, then the collateral is liquidated at maturity date of CAT bond and investors are repaid principal plus a compensation for bearing the catastrophe risks. However, if a trigger event occurs before the maturity, the SPV will liquidate collateral required to make the payment and reimburse the counterparty according to the terms of the catastrophe bond transaction, and CAT bond investors will only receive part of the capital. The difficulty in issuing Cat

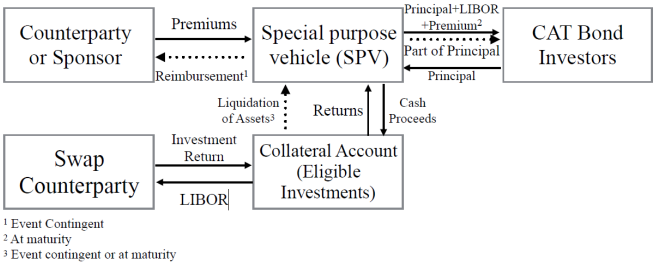


Figure 1.1.: Structure of Cat bonds Shao and Papaioannou (2015)

Bonds lies in the definition of the hedge trigger. Cat Bonds generally take the place of high tranches of a reinsurance program for which the realisation of the risk could lead to a default of re-insurers. Thus, we distinguish different types of Cat Bonds according to the trigger:

- Cat Bonds indemnified: The assignor is compensated for its losses.
- Parametric Cat Bonds based on the physical characteristics of the event to trigger coverage. These are for example the speed of the wind or the magnitude of an earthquake.
- Index Cat Bonds: the trigger is an indicator of the losses incurred by the entire market during the disaster.

There are also Cat Bonds that use multiple triggers. Also, during the issue, investors and (re)insurers also agree on the choice of a modelling agency or agencies always with the aim of clarifying and facilitating the understanding of risk by investors. Despite increasingly recurrent and costly disasters, the expansion of cat bonds market is restricted, and one of the reasons of this is the complexity of traded products. Indices or models used for cat bonds are hard to understand. In theory, these indices must be transparent, simple, viable and available and published by independent structure. In practice, they are very technical and difficult for investors to understand. In this context, the modelling of natural disasters and the pricing of insurance risk transfer products is a major issue for all market players. Indeed, investors are not keen to buy cat bonds, whose characteristics remain uncertain and unknown. Thus, our project focuses on modelling natural disasters, in particular earthquakes. Specifically, we will elaborate on how BNNs could help to make this modelling more accessible and transparent about its uncertainties.

Earthquakes modeling

Earthquakes can be generated by a sudden dislocation of large rock masses along "fault lines" fractures within the crust of the earth. Their main parameters are their locations, depths, fault rupture planes, and magnitudes.

The location, or "epicenter", of an earthquake is defined as the point on the surface of the earth directly above the "hypocenter" — the initial point of rupture within the earth. The "depth" is the distance between the hypocenter and the epicenter. A quake starts at a single point (the hypocenter), and then propagates through a fault rupture plane.

The area of this fault rupture plane is an important determinant of the magnitude of the earthquake. The magnitude of an earthquake is a measure of the total energy that is radiated from its rupture plane. Various common magnitude scales are in use, such as the moment magnitude, the surface-wave magnitude, the body wave magnitude, and the local magnitude scales. These scales can empirically be related to one another.
The concept of "recurrence interval" of earthquakes, i.e., the characteristic period of time between events of similar magnitude, has to be distinguished from the concept of "return period" of hurricanes. For hurricanes, probabilities are generally assumed to be constant from year to year. For earthquakes, however, scientists generally believe that at least certain faults have time-dependent probabilities. They believe that such faults build up stress at a constant rate and will release it periodically when it reaches certain levels. This means that the probability of an event occurring at a given location increases as the time period since the last event at this location increases. Recurrence intervals increase with the magnitude of the earthquake and, for very large earthquakes, can be in the range of hundreds or even thousands of years. To give one examples, the recurrence interval for the 1906 San Andreas earthquake (magnitude 7.9) is estimated to be about 210 years.

Thus, our work consisted in using BNN to provide a model to simulate earthquakes. This model must be able to predict, for a given time interval (1months, 1 or 5 years in our case) and for each zone resulting from our subdivision, the magnitude and the frequency of future earthquakes.The aim is to enable the investor to better assess the risk over the period covered by the cat bond.

2. Density based approach

Cat bond pricing is not a new topic and various methods have already been studied to address this issue. In order to determine and best defend the contribution that our BNN model represents in the field, we have, initially, implemented methods already established in the literature.

2.1. Frequency modelling

For frequency modelling, we use our historical database of earthquakes in Japan and by comparing the properties of the base (mean, standard deviation etc. .) with the properties of the usual frequency laws, we can use a known law to predict losses for future periods. In the context of insurance, to model frequency, the laws generally used are the following:

Table 2.1.: Frequency modelling distributions

	BINOMIAL	POISSON	NEGATIVE BINOMIAL
P	$B(n,p), n \in \mathbb{N}, p \in]0,1[$	$P(\lambda), \lambda \in]0,1[$	$NB(r,p), r \in]0,\infty[, p \in]0,1[$
$P(N=k)$	$\binom{n}{k} p^k(1-p)^k, 0 \leq k \leq n$	$\frac{\lambda^k e^{-\lambda}}{k!}, k \in \mathbb{N}$	$\frac{\Gamma(r+k)p^r(1-p)^k}{k!\Gamma(r)}, k \in \mathbb{N}$
$E[N])$	np	λ	$\frac{r(1-p)}{p}$
$V([N])$	$np(1-p)$	λ	$\frac{r(1-p)}{p^2}$

However, the Poisson distribution or the Negative Binomial are more often used to model the number of insurance claims. Moreover, these laws make it possible to define different properties for claims. Poisson distribution is a counting distribution, which models the behavior of the number of events over a time interval. Often used to model wait times (E.g.: How long does it take on average to wait for a bus at a stop) because of its characteristics. Poisson distribution is particularly well suited to reinsurance problems because it can describe the occurrence of unlikely or rare events within a portfolio of many policies. This distribution is neither over-dispersed nor under-dispersed. Thus, when samples show signs of over-dispersion, especially when the ratio of variance to empirical expectancy is much greater than 1, it may be that simple Poisson distribution is rejected in favor of so-called over-dispersion laws. Moreover, the distribution tail of a Poisson's distribution is extremely fine, it underestimates rare frequency events.
To overcome these problems, it is possible to model the frequency with the help of Poisson distribution, but with a parameter that also has a randomness or a time-dependent function , we are faced with a mixed Poisson distribution.

The Negative Binomial distribution (noted NB) is a mixed Poisson distribution in the sense that it can be deduced from a simple Poisson distribution whose parameter is subject to a random λ which follows a Gamma distribution.It appears in the study of the number of events that can be realized in time, events that do not have the same probability of

being realized . This distribution is very useful in insurance for the study of the number of claims per policy during a fixed time in portfolios with heterogeneous risks.

For parameter estimation, we use the maximum likelihood method which is theoretically the best of parametric estimation methods. Indeed, the maximum likelihood estimator is asymptotically unbiased and converging, and its variance is minimal.

2.2. Magnitude modelling

The method used for the severity/magnitude modelling is the one described in (Shao & Papaioannou, 2015). There, a model with multiple catastrophes and financial risks in a discrete-time period is developed as an extension of the approach of Cox and Pedersen. The framework applied is an incomplete and no-arbitrage one and it is assumed that all risks are mutually independent, and that aggregate consumptions depend on financial and catastrophic risks. For our study, we focuse only on the estimation of the distribution of the annual maximum earthquake magnitude using extreme value theory.
To define extremes, we traditionally look into the statistical behavior of

$$M_k^q = \max(X_{1k}^q, X_{2k}^q, \ldots, X_{ok}^q)$$

with q=1,...,NZ (NZ being the number of areas considered) and $X_{1k}^q, X_{2k}^q, \ldots, X_{ok}^q$ the sequence of o independent random variables with a universal distribution function F that measures the monthly, yearly or quinquennial maximum-magnitude earthquake in each region for the period $]k, k+1[$.$X_{ok}^q = 0$ if no earthquake occurs in zone q on that considered period. Thus, the sequence $M_k^q = \max(M_k^1, \ldots, M_k^N Z)$ corresponds to the kth annual maximum-magnitude earthquake over the observation period. The distribution of M_k^q can be derived for each year using the generalized extreme value (GEV) distribution (Shao & Papaioannou, 2015). The rescaled sample $(M_k^q)^* = (M_k^q - b_k)/a_k$ is a heavy-tailed distribution and the possible distribution is provided by the well-known Fisher–Tippet–Gnedenko theorem (Shao & Papaioannou, 2015).

Theorem 1. *(Fisher–Tippet–Gnedenko) If there exist sequences of constants* $a_k : k > 0$ *and* $b_k : k > 0$ *such that:*

$$P\left\{\frac{M_k^q - b_k}{a_k} \leq z\right\} \to G(z) \; as \; k \to \infty$$

for a non-degenerate distribution function G, then G is a member of the GEV family

$$G(z) = exp\left\{-\left[1 + \varepsilon\left(\frac{z - \beta_4}{\sigma_4}\right)\right]^{\frac{-1}{\varepsilon}}\right\} \tag{2.1}$$

defined on $\{z: \; 1 + \varepsilon(z - \beta_4)/\sigma_4\}$, *where* $-\infty < \beta_4 < \infty$,$\sigma_4 > 0$, $-\infty < \varepsilon < \infty$ *and* $\beta_4 = E(M_k^q), \sigma_4 = \sqrt{Var(M_k^q)}$.

Indeed,

$$P\left\{\frac{M_k^q - b_k}{a_k} \leq z\right\} \to G(z) \; as \; k \to \infty \Longleftrightarrow P\{M_k^q \leq b_k - z a_k\} \to G(z) \; as \; k \to \infty$$

In practice we work directly with realizations of the max:

$$P\{M_k^q \leq z\} \approx G(\frac{x-b_k}{a_k}) = \exp\left\{-\left[1+\varepsilon\frac{x-b_k}{a_k}\right]^{\frac{-1}{\varepsilon}}\right\} (Toulemonde, 2017)$$

In practice, the parameters a_k, b_k *and* ε are not known and must therefore be estimated. To estimate the couple parameters a_k, b_k, we need ε. And conversely to calculate ε, we need a_k, b_k. This leaves us with a one-equation problem with two unknowns, which is impossible to solve. To get around this problem, the idea is to introduce 2 additional parameters β_4 and σ_4. The mathematical rigour behind this idea is well explained in the course "Théorie des valeurs extrêmes et gestion des risques extrême" by Gwladys Toulemonde, Lecturer at Polytech Montpellier, Computer Science and Management Department (IG). This link will take you directly to this course.

The model has three parameters: location parameter , and shape parameter ε. When $\varepsilon = 0$ is the limit of Eq. (2.1) as $\varepsilon \to 0$, the model corresponds to the Gumbel family. For the cases $\varepsilon > 0$ and $\varepsilon < 0$, Eq.(2.1) leads to Frechét and Weibull family distributions, respectively.Then the GEV parameters can be estimated by maximizing the log-likelihood function.

3. Artificial Neural Networks

In order to price a parametric CAT bond, we need to estimate its parametric index trigger. These triggers are based on the occurrence of specific natural events whose probabilities can be estimated with observational time series. In the case of earthquakes, the seismic time series is a temporal stochastic point process marked by each earthquake's magnitude. One approach consists in dividing the time axis into equally spaced contiguous counting windows of duration Δ_t to produce a sequence of counts $\{N_t(\Delta_t)\}$ for $1 \leq t \leq T$, where Δ_t is called the counting time or timescale and T represents the number of counting time. (Telesca, 2018). A data based machine learning model should allow us to account for both spatial and temporal dependencies, in order to predict the probability of future earthquakes, both in frequency and intensity. The added bayesian framework will give us a measure of reliability in our model and predictions.

3.1. The Multilayer Perceptron

Artificial Neural Networks (ANNs) were first introduced back in 1943 by the neurophysiologist Warren McCulloch and the mathematician Walter Pitts. In their landmark paper "A Logical Calculus of Ideas Immanent in Nervous Activity," McCulloch and Pitts presented a simplified computational model of how biological neurons might work together in animal brains to perform complex computations using propositional logic (McCulloch & Pitts, 1990).

The Perceptron

Invented in 1957 by Frank Rosenblatt, the Perceptron is one of the simplest ANN architectures. To each input connection x_i is associated a weight w_i for $1 \leq i \leq n$. The threshold logic unit (TLU), computes a weighed sum of its input, then applies a step function h to that sum and outputs the result o as a binary. A bias term $b_0 = 1$ with a corresponding weight w_0 is added to the sum.

$$o = h\left(\mathbf{x}^\top w\right) \quad \text{with} \quad \mathbf{x} = \begin{pmatrix} b_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad \textit{and} \quad \mathbf{w} = \begin{pmatrix} w_0 \\ \vdots \\ w_n \end{pmatrix}$$

The perceptron is composed of a single layer of TLUs, each connected to all the inputs. When all the neurons in a layer are connected to every neuron in the previous layer, the layer is called a dense layer.

Figure 3.1 shows a Perceptron with two inputs and three outputs that can classify instances simultaneously into three different binary classes, making it a multioutput classifier.

To make accurate predictions, the perceptron needs to be trained. It is fed one training instance at a time, and it makes its prediction for each instance. For every output neuron

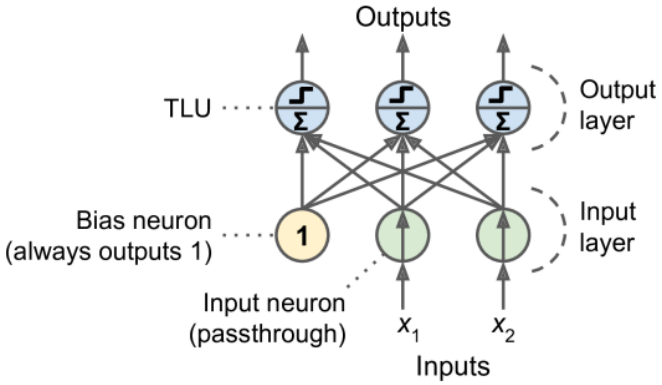


Figure 3.1.: Perceptron (Géron)

that produced a wrong prediction, it reinforces the connection weights from the inputs that would have contributed to the correct prediction.

$$w_{i,j}^{(k+1)} = w_{i,j}^{(k)} + \eta(y_j - \hat{y_j})x_i \tag{3.1}$$

Here, k is the k-th step of training, $w_{i,j}$ is the connection weight between the i-th input neuron and the j-th output neuron, x_i is the i-th input value of the current training instance, $\hat{y_j}$ is the output of the j-th output neuron, y_j is the target output of the j-th output neuron and η is the learning rate.

Equation 3.1, also known as the delta rule, is derived from the attempt to minimize the error in the output of the perceptron through gradient descent. The error is measured by the Mean Squared Error $E = \sum_j \frac{1}{2}(y_j - \hat{y_j})^2$.

However, perceptrons have serious weaknesses, in particular, their incapacity to solve trivial classification problems such as the XOR classification problem (Minsky & Papert, 2017). Its limitation is in its formulation, the percetron is a linear model. The parameter configuration $\mathbf{w}$ is a calibration of the slope of the decision boundary which is a straight line. A single layer perceptron can only separate classes if they are linearly separable. Figure3.2 illustrates the linearity of the percetron's decision frontier. The dataset is made up of classes of flowers: toxic and non-toxic.

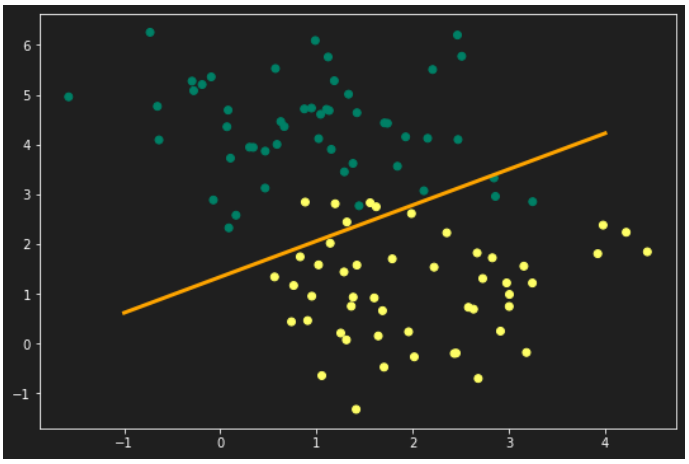


Figure 3.2.: Illustration of the adaptation of the perceptron to classes that are linearly separable

The Multilayer Perceptron (MLP)

Some of the limitations of the perceptron can be eliminated by stacking multiple perceptrons together in what is called a MLP. MLPs are composed of one input layer, one or more layers of TLUs, called hidden layers and one output layer. Every layers except the last one includes a bias neuron and is fully connected to the next layer (see figure 3.3).

Whereas the perceptron's TLUs used a step function to compute its (binary) output, different functions can be used with MLPs. These functions are called activation function and can be nonlinear. Some of the most widely used activation functions include the hyperbolic tangent $tanh : x \mapsto \frac{sinh(x)}{cosh(x)} = \frac{e^{2x}-1}{e^{2x}+1}$, the logistic function $\sigma : x \mapsto \frac{1}{1+e^{-x}}$ and the rectified linear unit (reLU) $reLU : x \mapsto max(0, x)$

These functions allow MLPs to perform classification or regression depending on its activation functions.

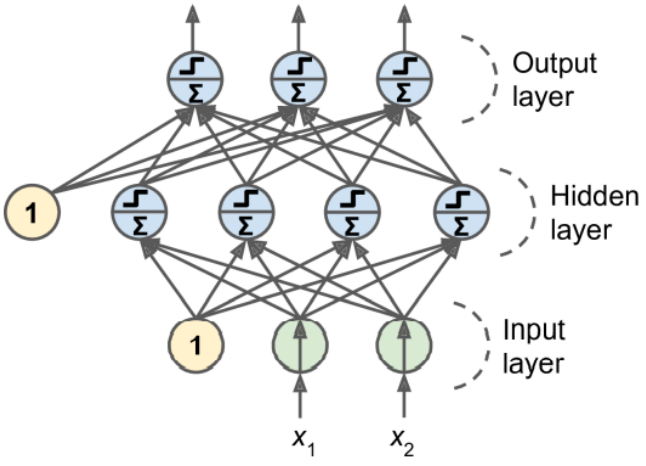


Figure 3.3.: A Multilayer Perceptron (Géron)

Backpropagation

Backpropagation is an algorithm for supervised learning of ANNs using gradient descent. Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural network's weights. It is a generalization of the delta rule for perceptrons 3.1 to multilayer feedforward neural networks.

3.2. Recurrent Neural Networks

Recurrent neural networks (RNNs) are artificial neural networks where the computation graph contains directed cycles. Unlike feedforward neural networks, where information flows strictly in one direction from layer to layer, in RNNs, information travels in loops from layer to layer so that the state of the model is influenced by its previous states. While feedforward neural networks can be thought of as stateless, RNNs have a memory which allows the model to store information about its past computations. This allows recurrent neural networks to exhibit dynamic temporal behavior and model sequences of input-output pairs, making them a viable model in time series prediction (Hewamalage et al., 2021).

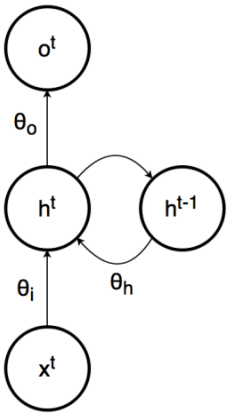


Figure 3.4.: A Simple RNN ("Recurrent Neural Networks", n.d.)

The following equations define how an RNN evolves over time:

$$\begin{aligned} o^t &= f\left(h^t; \theta\right) \\ h^t &= g\left(h^{t-1}, x^t; \theta\right) \end{aligned} \tag{3.2}$$

where o^t is the output of the RNN at time t, x^t is the input of the RNN at time t, h^t is the state of the hidden layer(s) at time t, f, g are activation functions and $\theta_i, \theta_h, \theta_o$ are the combined weight and bias parameters associated with the input in recurrent layers, the hidden units in recurrent layer and the output layer respectively.

Because of the recurrent nature of RNNs, the weights $\theta_i, \theta_h, \theta_o$ of the network are shared temporally. The difference in the output sequence arise from the context preserved by the previous hidden layer state h^{t-1}.

Figure 3.6 shows different types of RNNs. For seismic time series predictions, we will use many-to-one architecture for very short-term predictions (one period) and many-to-many

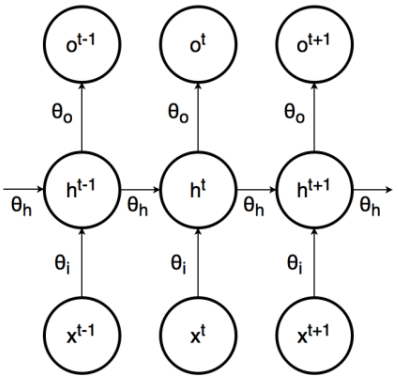


Figure 3.5.: Unrolled RNN ("Recurrent Neural Networks", n.d.)

architecture for long-term predictions (CAT Bond maturity period).

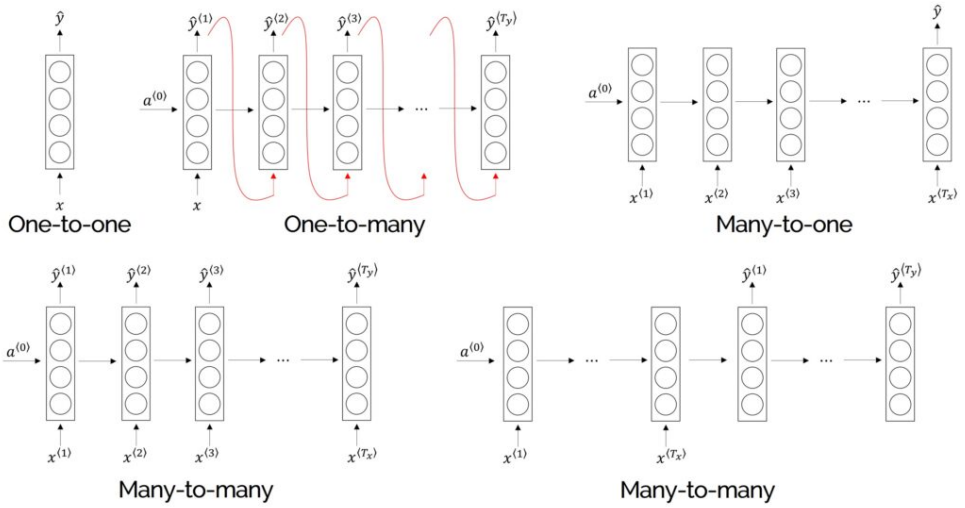


Figure 3.6.: Architectural Types of Different RNNs (Zivkovic)

Training a RNN

Training a RNN is very similar to training a regular feedforward ANN and use a backpropagation algorithm called backpropagation through time (BPTT). It works by applying the backpropagation algorithm to the unrolled RNN.

However, when using the chain rule to compute the gradients of the recurrent parameters for a standard RNN, there is an iterated product of matrices that can explode or vanish along some vector direction (in particular, the directions corresponding to the eigenvectors with the leading eigenvalues of the recurrent weight matrix). Thus, the gradient either becomes less and less useful over time due to vanishing, or causes massive instability due to explosion (Pascanu et al., 2013).

This problem is known as the vanishing / exploding gradient problem and is the motivation behind different RNNs architecture such as Gated Recurrent Units (GRU) and Long Short Term Memory (LSTM).

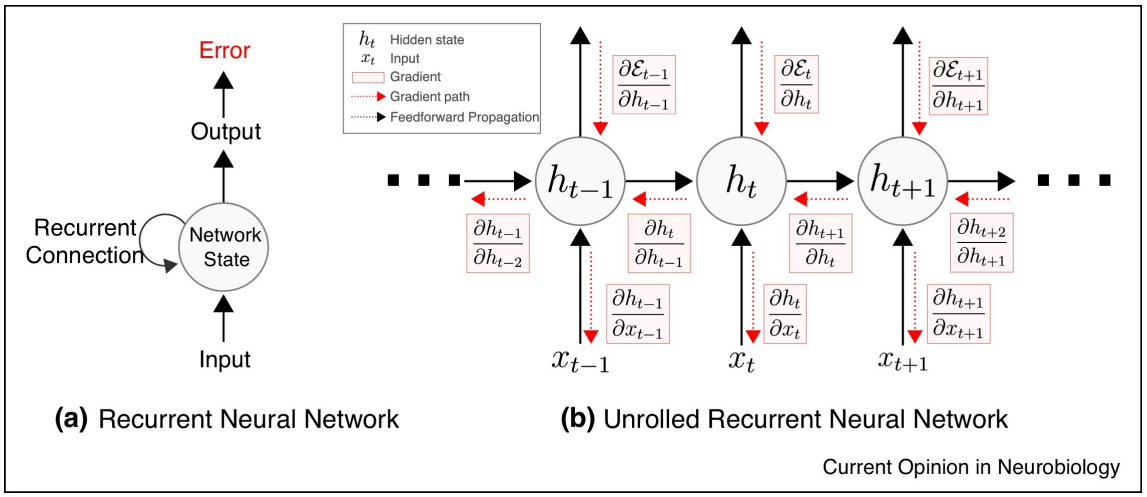


Figure 3.7.: Backpropagation through time (Lillicrap & Santoro, 2019)

3.3. Long Short Term Memory

The Long Short Term Memory (LSTM) cell was introduced in 1997 by Hochreiter and Schmidhuber because of the incapacity of RNNs to deal with long-term dependencies. They improved the remembering capacity of the standard recurrent cell by introducing a "gate" into the cell.

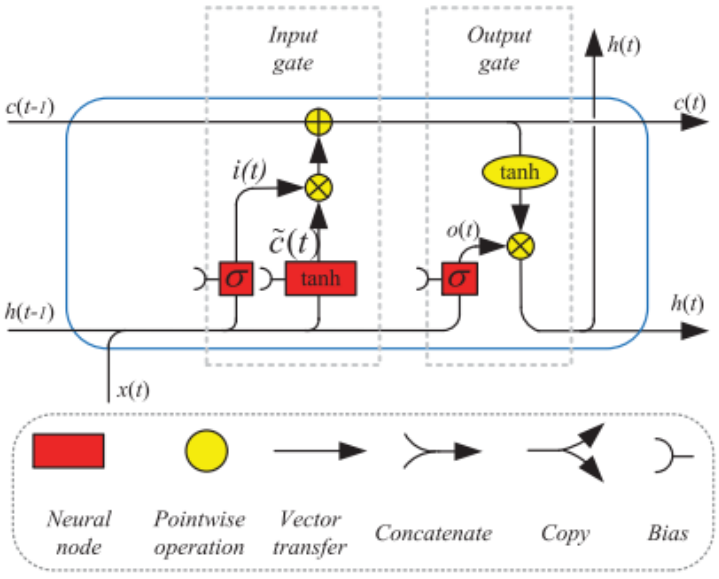


Figure 3.8.: LSTM without a Forget Gate (Yu et al., 2019)

LSTM networks are composed of an input layer, one or more hidden layers, and an output layer. The main characteristic of LSTM networks is contained in the hidden layer(s) consisting of "memory cells". Each of the memory cells has three gates maintaining and adjusting its cell state $c(t)$: a forget gate $f(t)$, an input gate $i(t)$, and an output gate $o(t)$.

At every timestep t, each of the three gates is presented with the input $x(t)$ as well as the

output of the memory cells at the previous timestep $t-1$. Hereby, the gates act as filters, each fulfilling a different purpose:

- The forget gate defines which information is removed from the cell state.
- The input gate specifies which information is added to the cell state.
- The output gate specifies which information from the cell state is used as output.

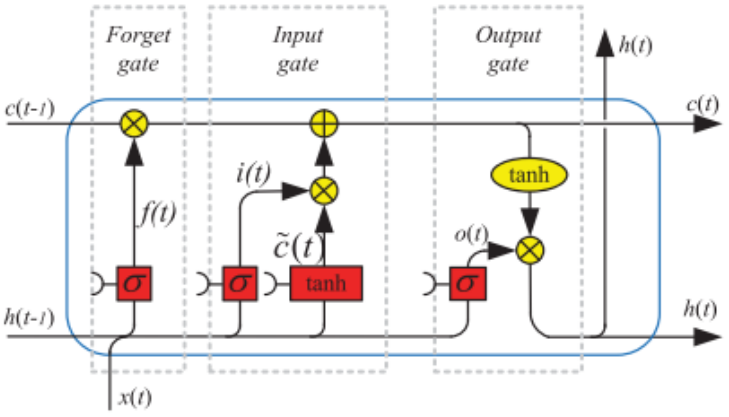


Figure 3.9.: LSTM cell architecture (Yu et al., 2019)

4. Bayesian Neural Networks

Modern deep learning methods constitute incredibly powerful tools to tackle a myriad of challenging problems. However, since deep learning methods operate as black boxes, the uncertainty associated with their predictions is often challenging to quantify (Nguyen et al., 2014). The Bayesian paradigm provides a rigorous framework to analyze and train uncertainty-aware neural networks.

Two main idea differentiate the bayesian paradigm from the frequentist paradigm.

- Probability is a measure of belief in the occurrence of events, rather than the limit in the frequency of occurrence when the number of samples goes toward infinity, as assumed by frequentists.
- Prior beliefs influence posterior beliefs.

This interpretation of probabilities is summarized in Bayes' theorem:

$$P(H \mid D) = \frac{P(D \mid H)P(H)}{P(D)} = \frac{P(D, H)}{\int_H P(D, H')\, dH'} \tag{4.1}$$

Equation 4.1 is still true in the frequentist interpretation with H and D considered as a set of outcomes. The Bayesian interpretation on the other hand considers H to be a hypothesis about which one holds some prior belief, and D to be some data that will update one's belief about H.

The probability distribution $P(D \mid H)$ is called the likelihood. It encodes the aleatoric uncertainty in the model, i.e., the uncertainty due to the noise in the process. $P(H)$ is the prior and $P(D) = \int_H P(D, H')\, dH'$ the evidence. $P(H \mid D)$ is called the posterior. It encodes the epistemic uncertainty, i.e., the uncertainty due to the lack of data. $P(D \mid H)P(H) = P(D, H)$ is the joint probability of D and H.

4.1. What is a Bayesian Neural Network

A Bayesian Neural Network (BNN) is a stochastic ANN trained using Bayesian inference. The stochastic part is due to the network either having a stochastic activation or stochastic weights (see figure 4.1) as opposed to a classic neural network whose parameters are point estimate.

Let $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b})$ be the parameters of a network, where $\mathbf{W}$ are the weights of the neural connections between layers and $\mathbf{b}$ the biases. Then, in a non Bayesian neural network, the standard approach is to approximate a minimal cost point estimate of the network parameters $\hat{\boldsymbol{\theta}}$, using the training data D comprised of a series of input $\boldsymbol{x}$ and their corresponding labels $\boldsymbol{y}$, and the backpropagation algorithm with respect to a cost function. This can be viewed as finding the best function $\Phi_{\boldsymbol{\theta}}$ such that $\boldsymbol{y} \approx \Phi_{\theta}(\boldsymbol{x})$ with respect to $\boldsymbol{\theta}$.

However, point estimate neural networks are unable to assess the reliability of their predictions and can generalize in unforeseen and overconfident ways on out-of-training distribution data points (Nguyen et al., 2014). Stochastic neural networks on the other hand

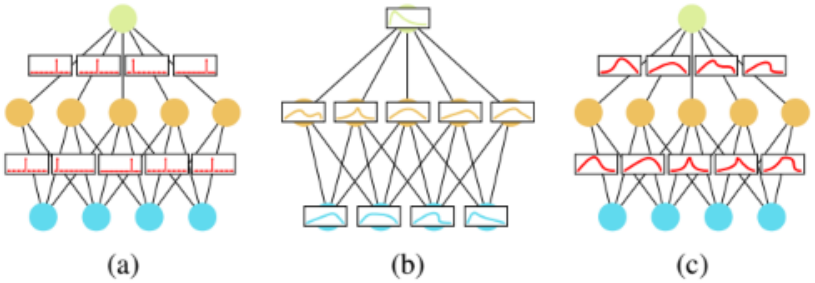


Figure 4.1.: (a) Point estimate neural network, (b) stochastic neural network with a probability distribution for the activations, and (c) stochastic neural network with a probability distribution over the weights. (Jospin et al., 2020)

simulate multiple possible models $\boldsymbol{\theta}$ with their associated probability distribution $p(\boldsymbol{\theta})$. By comparing the predictions of the different sampled model parametrizations, the network is then able to return a measure of uncertainty in its predictions. If different models agree, then the uncertainty is low. If they disagree, then the uncertainty is high. This process can be summarized as follows:

$$\begin{aligned} \boldsymbol{\theta} &\sim p(\boldsymbol{\theta}) \\ \boldsymbol{y} &= \Phi_{\boldsymbol{\theta}}(\boldsymbol{x}) + \boldsymbol{\epsilon} \end{aligned} \tag{4.2}$$

where $\boldsymbol{\epsilon}$ represents random noise.

Designing a BNN requires choosing a functional model, that is, a network architecture, and a stochastic model, that is, a prior distribution over the possible model parametrization $p(\boldsymbol{\theta})$ and a prior confidence in the predictive power of the model $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})$. The model then infers a posterior distribution $p(\boldsymbol{\theta} \mid D)$ over the parameters θ of the model after observing the data D. From Eq. 4.1, we have:

$$p(\boldsymbol{\theta} \mid D) = \frac{p\left(D_y \mid D_{\boldsymbol{x}}, \boldsymbol{\theta}\right) p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p\left(D_{\boldsymbol{y}} \mid D_{\boldsymbol{x}}, \boldsymbol{\theta}'\right) p\left(\boldsymbol{\theta}'\right) d\boldsymbol{\theta}'} \propto p\left(D_{\boldsymbol{y}} \mid D_{\boldsymbol{x}}, \boldsymbol{\theta}\right) p(\boldsymbol{\theta}) \tag{4.3}$$

where D_x represents the training inputs and D_y represents the training labels. $p(\boldsymbol{\theta} \mid D)$ is called the **epistemic uncertainty** and represents the uncertainty towards the model.

The predictions $\boldsymbol{y}$ of the model on a new test example $\boldsymbol{x}$ are given by the Bayesian model average (BMA) $p(\boldsymbol{y} \mid \boldsymbol{x}, D)$, also called the marginal

$$p(\boldsymbol{y} \mid \boldsymbol{x}, D) = \int_{\boldsymbol{\theta}} p\left(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}'\right) p\left(\boldsymbol{\theta}' \mid D\right) d\boldsymbol{\theta}' \tag{4.4}$$

The BMA quantifies the model's uncertainty on its prediction. Unfortunately, the integral in Eq. 4.4 is computationally intractable for most non-trivial neural networks structure. To address this problem, two broad approaches have been introduced: Markov Chain Monte Carlo (MCMC) methods and Variational Inference methods.

4.2. Variational Inference Methods

Variational inference is not an exact method. Rather than allowing sampling from the exact posterior $p(H \mid D)$, the idea is to have a distribution $q_\phi(H)$, called the variational distribution, parametrized by a set of parameters ϕ. The goal is to find the parameters ϕ that minimises the Kullback-Leibler (KL) divergence between the variational distribution and the exact posterior (Graves, 2011).

$$\begin{aligned} \phi^\star = \arg\min_\phi D_{KL}(q_\phi \| P) &= \arg\min_\phi \int_H q_\phi\left(H'\right) \log\left(\frac{q_\phi\left(H'\right)}{P\left(H' \mid D\right)}\right) dH' \\ &= \arg\min_\phi \mathcal{F}(D, H) \end{aligned} \tag{4.5}$$

where the KL divergence D_{KL}, measures the difference or the 'closeness' between probability distributions and $\mathcal{F}(D, H)$ the cost function known as the variational free energy.

In the case of variational inference on a neural network, our hypothesis H corresponds to the model's weight parameters $\boldsymbol{\theta}$ to approximate. From Eq. 4.5 , we can rewrite the variational free energy as:

$$\mathcal{F}(D, \boldsymbol{\theta}) = D_{KL}(q_\phi(\boldsymbol{\theta}) \| P(\boldsymbol{\theta})) - \mathbb{E}_{q_\phi(\boldsymbol{\theta})}\left(\log P(D \mid \boldsymbol{\theta})\right) \tag{4.6}$$

This cost function is a trade-off between satisfying the complexity of the data and having an accurate representation of the data given our prior.

4.2.1. Bayes by Backprop

Bayes by Backprop seeks to minimize the variational free energy (Eq. 4.6). However, exactly minimizing this cost is computationally prohibitive and gradient descent and various approximations are used instead. We approximate the exact cost as (Blundell et al., 2015):

$$\mathcal{F}(D, \boldsymbol{\theta}) \approx \sum_{i=1}^{n} \log q_\phi(\boldsymbol{\theta}^{(i)}) - \log P(\boldsymbol{\theta}^{(i)}) - \log P(D \mid \boldsymbol{\theta}^{(i)}) \tag{4.7}$$

where $\boldsymbol{\theta}^{(i)}$ denotes the ith Monte Carlo sample drawn from the variational posterior $q_\phi(\boldsymbol{\theta})$.

Furthermore, we can suppose that the variational posterior follows a Gaussian distribution. Then a sample of the weights $\boldsymbol{\theta}$ can be obtained by sampling a unit Gaussian, shifting it by a mean μ and scaling by a standard deviation σ. We parameterise the standard deviation pointwise as $\sigma = \log(1+\exp(\rho))$ so that σ is always non-negative. The variational parameters are $\phi = (\mu, \rho)$. Thus, we get the following posterior sample of the weights: $\boldsymbol{\theta} = \mu + \log(1 + \exp(\rho)) \circ \epsilon$ where $\circ$ is point-wise multiplication and $\epsilon \sim N(0, I)$. The following algorithm summarizes the optimisation process.

1. Sample $\epsilon \sim N(0, I)$.
2. Let $\boldsymbol{\theta} = \mu + \log(1 + \exp(\rho)) \circ \epsilon$

3. Let $\phi = (\mu,)$
4. Let $f(\boldsymbol{\theta}, \phi) = \log q_\phi(\boldsymbol{\theta}) - \log P(\boldsymbol{\theta})P(D \mid \boldsymbol{\theta})$
5. Calculate the gradient with respect to the mean

$$\Delta_\mu = \frac{\partial f(\boldsymbol{\theta}, \phi)}{\partial \boldsymbol{\theta}} + \partial f(\boldsymbol{\theta}, \phi)\mu \tag{4.8}$$

6. Calculate the gradient w.r.t. the standard deviation parameter ρ

$$\Delta_\rho = \frac{\partial f(\boldsymbol{\theta}, \phi)}{\partial \boldsymbol{\theta}} \frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(\boldsymbol{\theta}, \phi)}{\partial \rho} \tag{4.9}$$

7. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha\Delta_\mu \tag{4.10}$$

$$\rho \leftarrow \rho - \alpha\Delta_\rho \tag{4.11}$$

Bayes by Backprop (BBB) can be adapted to RNNs in what is called Bayes by Backprop Through Time (Fortunato et al., 2017). Recall from 3.2 that a RNN can be trained on a sequence of length T by backpropagation through by unrolling T times into a feedforward network. From Eq. 3.2, let's denote h_t as the state of the hidden layer of a RNN at step t and $h_{1:T}$ as the state of an RNN core unrolled for T steps. Crucially, the weights at each unrolled steps are shared. Thus, each weight of the RNN core receives T gradient contributions and the variational free energy becomes:

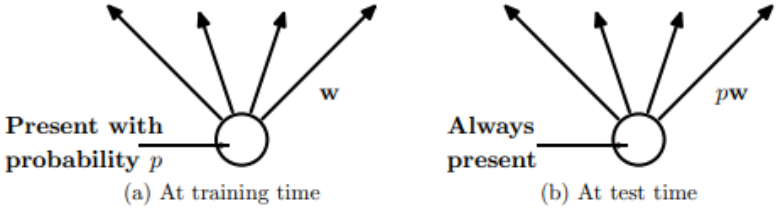
$$\mathcal{F}(D, \boldsymbol{\theta}) = D_{KL}(q_\phi(\boldsymbol{\theta}) \| P(\boldsymbol{\theta})) - \mathbb{E}_{q_\phi(\boldsymbol{\theta})} \left(\log P(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}, \boldsymbol{x}_{1:T})\right) \tag{4.12}$$

The same BBB algorithm 4.2.1 can then be used on this cost function to take gradient steps.

4.2.2. Monte Carlo-Dropout

Dropout is a regularization technique used to prevent deep neural network from overfitting. The key idea is to randomly drop units and their connections from the neural network during training (Srivastava et al., 2014), thus, sampling from different versions of the same 'thinned' network. At test time, predictions are made with the unthinned network, whose weights are scaled-down versions of the training weights. If a unit is retained with probability p during training, the outgoing weights of that unit are multiplied by p at test time.

Monte Carlo-Dropout (MC Dropout) uses the same regularization technique at training time and keeps it at test time, resulting in a distribution for the output predictions (Gal & Ghahramani, 2015). It is variational inference with a variational distribution defined for each weight matrix as:



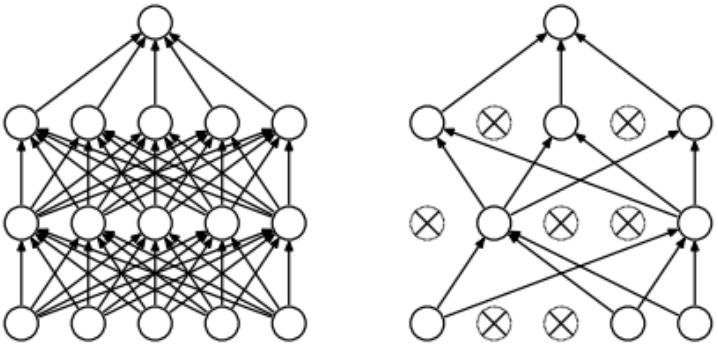
(a) At training time (b) At test time

Figure 4.2.: Neural Network unit with dropout at training and test time (Srivastava et al., 2014)

$$\begin{aligned} \boldsymbol{z}_{i,j} &\sim \text{Bernoulli}\,(p_i) \\ \boldsymbol{W}_i &= \boldsymbol{M}_i \cdot \text{diag}\,(\boldsymbol{z}_i) \end{aligned} \tag{4.13}$$

with $\boldsymbol{z_i}$ being the random activation coefficients and M_i the matrix of weights before dropout is applied.

MC-Dropout is a very convenient technique to perform Bayesian deep learning. It is straightforward to implement and requires little additional knowledge or modeling effort compared to traditional methods. It often leads to a faster training phase compared to other variational inference approaches.



(a) Standard Neural Net (b) After applying dropout.

Figure 4.3.: Dropout (Srivastava et al., 2014)

The model uncertainty can then be obtained by running multiple forward passes through the network and averaging the results.

5. Application to Earthquake Prediction in Japan

5.1. Problem Statement

We consider earthquakes records from 1960 to 2021 in Japan. The data was collected on the United States Geological Survey (USGS) site with their API. Each entry has four parameters: the occurrence time of the earthquake t, its latitude, longitude and its magnitude M, with latitudes ranging from 30°N to 46°N and longitudes ranging from 129°E to 149°E.

We then split the data in different zones according to the earthquake's location and the seismic faults around Japan as shown in figure 5.1. This division allows to discover potential spatial relations in our model and better reflects real life as some zones are more vulnerable than other.

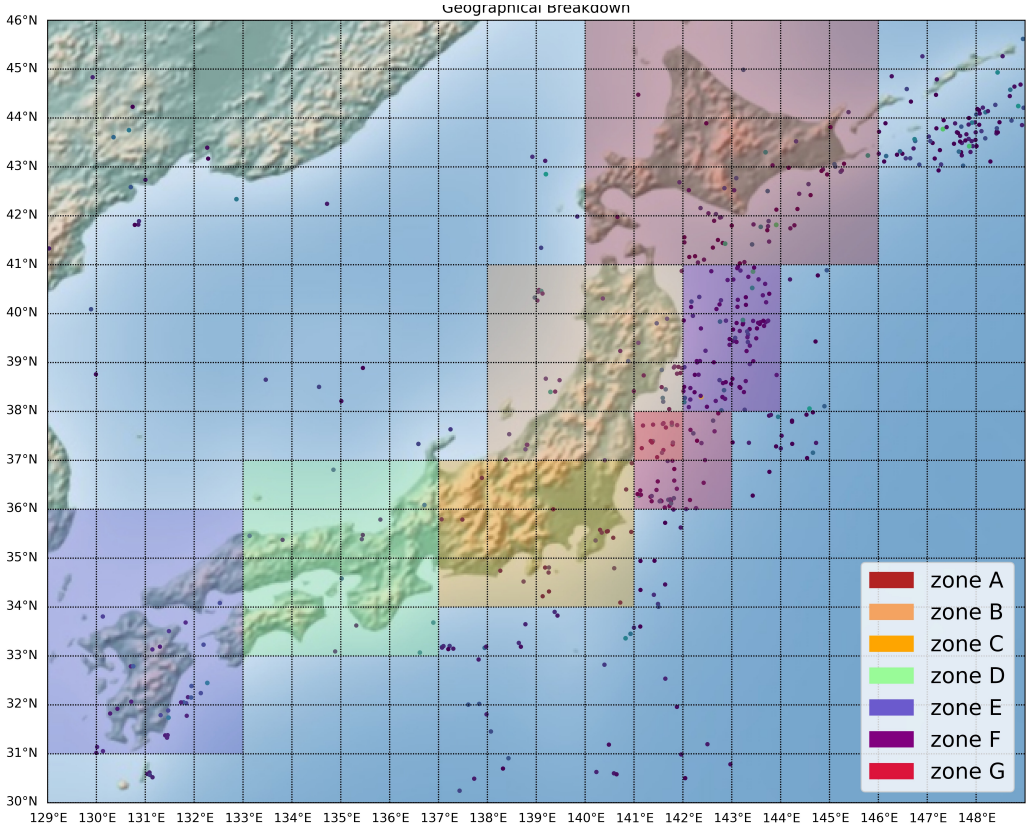


Figure 5.1.: Geograpical Breakdown. The points represent earthquakes' location of magnitude $M > 6$

Our goal is to predict the frequency and the intensity of the next earthquakes for a time frame similar to a CAT Bond's maturity time.

5.2. Data description

The dataset consists in records of 34,064 earthquake from 1960 to 2021 in Japan with magnitude $M > 2.5$. The sample has already been balanced for magnitude (Fig. 5.2).

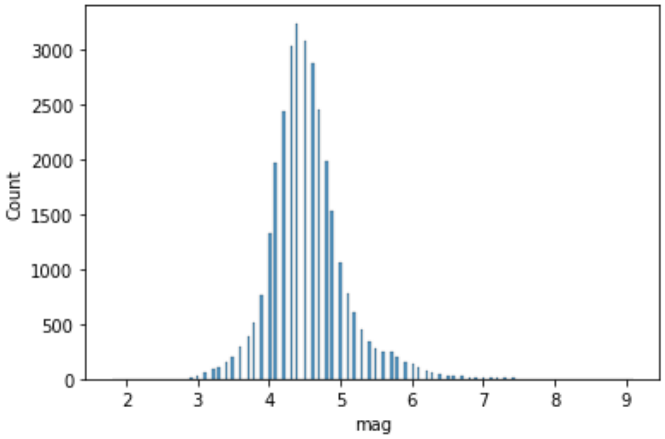


Figure 5.2.: Histogram of magnitudes

We then create two different data sets to work on frequency modelling and intensity modelling. The frequency data set is based on a monthly count of earthquake with magnitude $M \geq 4.5$ by zones. The intensity data set is created by taking the maximum magnitude by month and by zone. Thus, both data sets contains 732 rows representing each months from 1960 to 2021, and 7 columns representing each zones.

5.3. Density-based approach

The following paragraphs present the application of what we may call the 'usual method' of modelling the frequency and severity (magnitude) of natural disasters. Note that here, for parametric Cat bonds, only magnitude modeling is important since it is the one that defines the trigger.

Magnitude We take region 1 as an example for analysis. We maximize the GEV loglikelihood for these data and achieve the estimate

$$(\hat{\beta}_4, \hat{\sigma_4}, \hat{\varepsilon_4}) = (4.9347280, 0.3647488, 0.1729006)$$

for which the log-likelihood is 401.2018. The approximate variance–covariance matrix of the parameter estimates is

$$V = \begin{bmatrix} 0.0003004976 & 0.0001358776 & -0.0002505777 \\ 0.0001358776 & 0.0001882045 & -0.0001110949 \\ -0.0002505777 & -0.0001110949 & 0.0014669025 \end{bmatrix}$$

Therefore, we can easily obtain standard errors 0.01733487, 0.01371876 and 0.03830016 for β_4,σ_4 and ε_4 with approximate 95 % confidence intervals of $\hat{\beta}_4 \in [4.9007516, 4.9687043]$, $\hat{\sigma_4} \in [0.3307724, 0.3916375]$ and $\hat{\varepsilon} \in [0.1389243, 0.2479690]$.

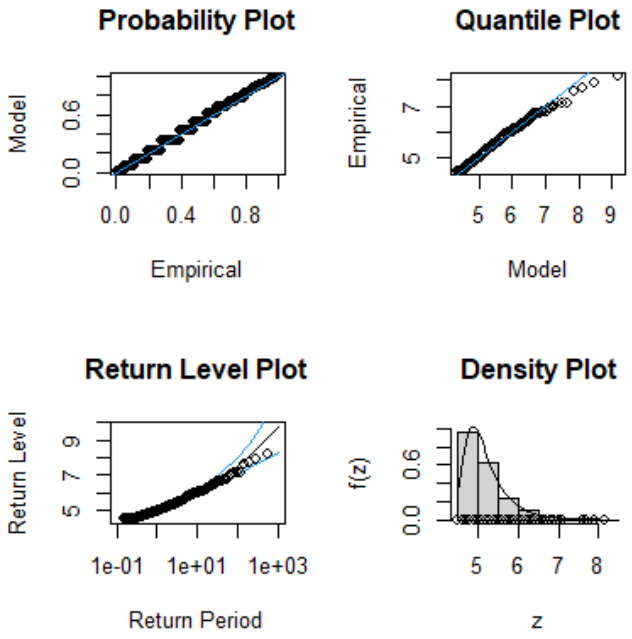


Figure 5.3.: Diagnostic plots for GEV fitting to the annual maximum-magnitude earthquakes in region C in Japan

To assess the accuracy of the GEV model fitted to the data, we show various diagnostic plots of M_k^3 in Fig. 5.3. The probability and quantile plots are close to linear, which confirms the validity of the fitted model. The estimate estimated curve in the return level plot is nearly linear.According to the histogram density plot of the data, the density estimate is consistent.Consequently,the GEV model provides a good fit.

Furthermore, the tail behaviour (Shao & Papaioannou, 2015) of the distribution displayed in Fig. 5.4 reflects the sample mean excess, and the downward trend suggests a very short tail behaviour for the annual maximum-magnitude earthquakes in region C in Japan.
Similar analysis can be conducted for the other regions and the exceeding probabilities intervals for the GEV distributions are listed in Table 5.1.

Table 5.1.: Exceeding probabilities for the model in regions A to G

	Region A	Region B	Region C	Region D	Region E	Region F	Region G
$P(X \leq 4.5)$	0.04117	0.029488	0.02240	0.04236	0.05322	0.040691	0.03124
$P(4.5 < X \leq 6.5)$	0.91647	0.91030	0.93805	0.91591	0.87946	0.91140	0.95377
$P(6.5 < X \leq 8.5)$	0.02963	0.048650	0.03629	0.03294	0.04304	0.03933	0.01455
$P(8.5 < X \leq 9.1)$	0.002825	0.00351	0.00144	0.00250	0.00467	0.00269	0.00025
$P(X > 9.1)$	0.00991	0.00805	0.00183	0.00628	0.01961	0.00589	0.00019

A prediction can then be made using the parameters found above.

Confidence Intervals are estimated by Monte Carlo methods using 100 samples per period.

Figure 5.4.: Sample mean excess for annual maximum-magnitude earthquakes in region C in Japan with 95 % confidence interval

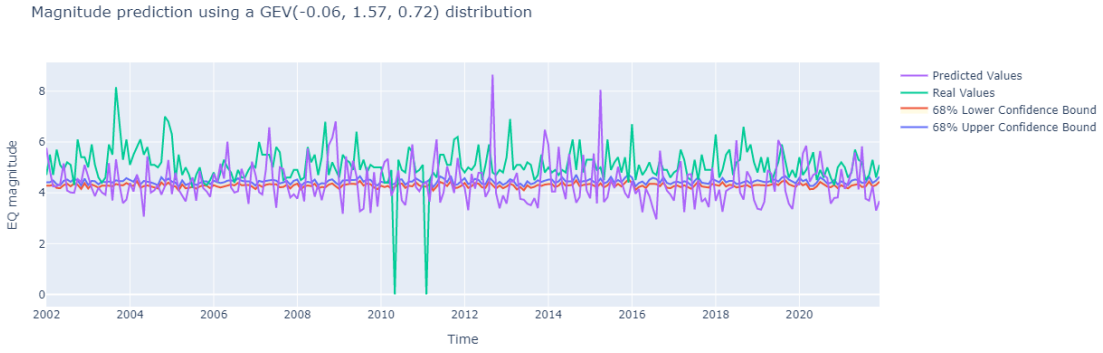


Figure 5.5.: Prediction of maximum magnitude by month in zone A for a $GEV(-0.06, 1.57, 0.72)$ distribution

5.4. Bayesian LSTM

By considering our frequency and intensity data sets as time series, we can use RNNs models to make our predictions. In the following parts, let's denote by $\{x_t\}_t$, our univariate time series, τ our lookback window, i.e., the number of observation used as input and F our prediction horizon with t, τ and F in months.

5.4.1. Frequency

To model frequency, we first trained a LSTM with a 7 year lookback window ($\tau = 84$) and a timestep output of $F = 1$ month. We used data from 1960 to 2014, worth of 660 months of data for training and the final 84 months as the test set. Our model is a many-to-many model, taking the seven features associated with the 7 different zones, and outputting the one month prediction for the seven different zones. The architecture used is two layers of LSTM cells with both 100 hidden units and *tanh* activation functions. The last layer is a dense layer with a *reLU* activation functions to account for positive counts.

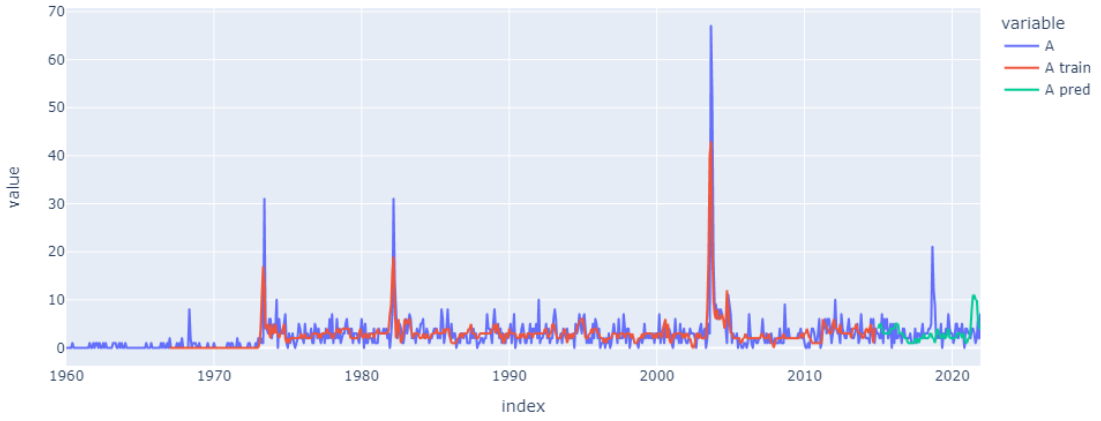


Figure 5.6.: LSTM Frequency Prediction for $\tau = 84$ and $F = 1$

This model is not really useful because 1 months predictions are of no use for the Cat Bond investor.

By adding MC Dropout to the model, with $p = 0.6$, we get the confidence interval in Figure 5.7.

We didn't try any additional models because frequency isn't really an important predictor of CAT Bond trigger in and of itself.

5.4.2. Intensity

To predict intensity, We decided to use two regressions LSTM models. One is using Bayes by Backprop as an inference scheme (Esposito, 2020) and the other one is using MC Dropout. They take as input the maximum earthquake magnitude on a monthly basis

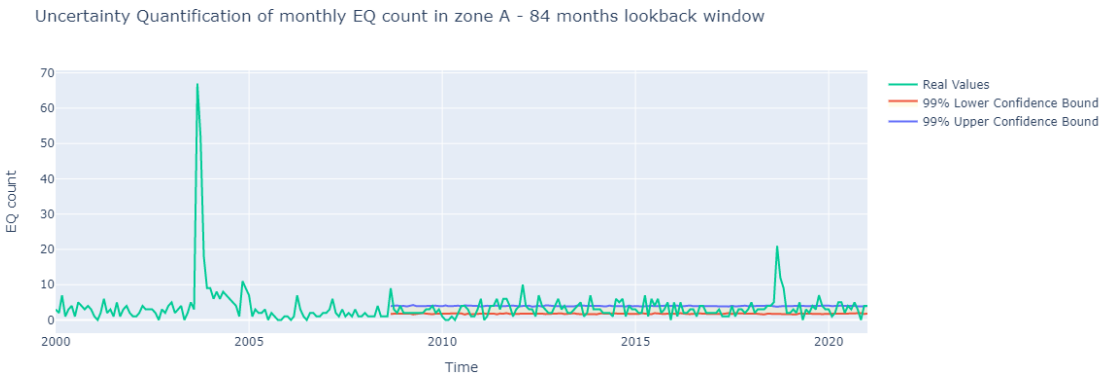
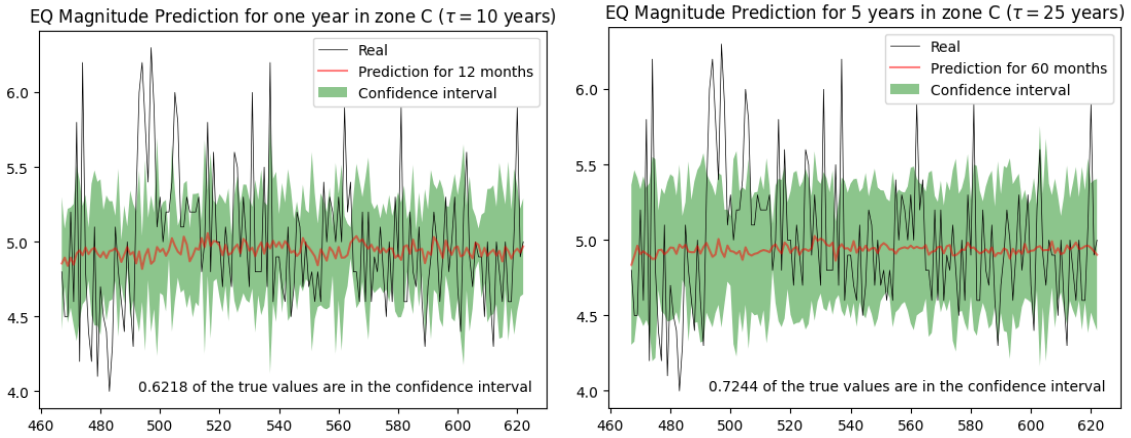


Figure 5.7.: BLSTM Frequency Prediction (MC Dropout) for $\tau = 84$ and $F = 1$



((a)) BLSTM Intensity Prediction for $\tau = 120$ and $F = 12$

((b)) BLSTM Intensity Prediction for $\tau = 300$ and $F = 60$

Figure 5.8.: Bayes by Backprop LSTM Predictions

and return the predicted maximum earthquake magnitude in the desired time period per months.

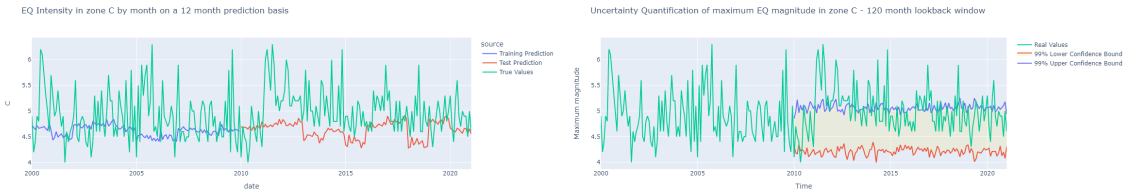
The first model uses BBB. Following (Blundell et al., 2015), we can use a scale mixture of two Gaussian densities as the prior. Each density is zero mean, but differing variances:

$$P(\boldsymbol{\theta}) = \prod_j \pi \mathcal{N}\left(\boldsymbol{\theta}_j \mid 0, \sigma_1^2\right) + (1-\pi)\mathcal{N}\left(\boldsymbol{\theta}_j \mid 0, \sigma_2^2\right) \tag{5.1}$$

where $\boldsymbol{\theta}_j$ is the jth weight of the network, $\mathcal{N}(x \mid \mu, \sigma^2)$ is the Gaussian density evaluated at x with mean μ and variance σ^2 and σ_1^2, σ_2^2 are the variances of the mixture components.

In our model, we used a single Gaussian prior ($\pi = 1$) of variance $\sigma^2 = 1$ and $\rho = -3$. We made predictions for a one year period and a 5 year period using lookback windows of 10 and 25 years respectively in zone C.

Our prediction (red) is the mean at each time step t of our model prediction and is



((a)) BLSTM Intensity Prediction for $\tau = 120$ and $F = 12$

((b)) BLSTM Intensity Uncertainty for $\tau = 120$ and $F = 12$

Figure 5.9.: MC Dropout LSTM Predictions

computed by a Monte Carlo simulation. The confidence interval is computed from the standard deviation associated with each Monte Carlo simulation. A measure of epistemic uncertainty $P(\boldsymbol{\theta} \mid D_x)$ is given by the ratio of true values outside our confidence interval. Our computations show that the epistemic uncertainty is lower for a lookback window of 25 years ($P(\boldsymbol{\theta} \mid D_x) = 0.7244$) than for a lookback window of 10 years ($P(\boldsymbol{\theta} \mid D_x) = 0.6218$). This results is coherent with the bayesian way of thinking. The more we have information at our disposal, the better we can update our beliefs accordingly.

The second model uses MC Dropout. We use $p = 0.5$ as our dropout probability and similarly to the previous model, we compute confidence intervals using Monte Carlo simulations at each time steps predicted.

We find an epistemic uncertainty of 0.446.

6. Discussion

In both frequency and intensity, the results given by the BLSTM models were rather disappointing. However, they are quite logical, given the nature of our input data, our prediction horizon and our problem at hand. Indeed, RNNs are particularly good at predicting time series and capturing patterns through time such as seasonality but earthquake's seasonality does not really exist. Furthermore, by enforcing a broader prediction time horizon, our models only feel confident predicting narrow confidence intervals close to the historical mean. Accordingly, the model whose predictions looked the most realistic were those produced by our first RNN model with a one month prediction horizon and a 7 year lookback window (see Figure 3.9).

The time window $\Delta_t = 1$ month used to create our count process could have also undermined our results. Indeed, recouping events by months, effectively reducing a time continuous process into somewhat large time windows inevitably hides any eventual time relationship.

7. Conclusion

Whereas density based models bring concrete numbers to the exceeding probabilities of earthquakes (see table 5.1), BNN models bring predictions and an easy metric evaluating its reliability. On the flip side, BNN lack the predictability power to extreme events (in our case study, high magnitude earthquakes) and instead give us a very safe prediction concentrated around the historical mean earthquake magnitude. As explained in 6, this behavior was somewhat expected given the nature of our problem.

In practice, earthquake prediction rely on widely different models such as elastic rebound (the study of the energy release during an earthquake), high energy electromagnetic pulses or the study of signals emitted from fault zones.

To improve on our models, we could have tried to better use spatial data by feeding our neural networks with the times series associated with each zones at once. We could've also tried to do more feature engineering. For instance, by adding features such as the amount of times elapsed between an earthquake of magnitude $M \geq m$ and the last one or the number of earthquake in zone Z between two occurrences of magnitude $M \geq m$.

We could also have improved the choice of parameters in our ANN models by better exploring our parameter space and exploring with different architectures (Bidirectional LSTM, peephole structure, different number of nodes and layers, etc.).

Bibliography

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural networks. https://doi.org/10.48550/ARXIV.1505.05424

Djekadom, O. (2016). *Etude du risque de base d'un cat bond basé sur un indice de marché institut des actuaires*. Institut des actuaires.

Esposito, P. (2020). Blitz - bayesian layers in torch zoo (a bayesian deep learing library for torch).

Fortunato, M., Blundell, C., & Vinyals, O. (2017). Bayesian recurrent neural networks. https://doi.org/10.48550/ARXIV.1704.02798

Gal, Y., & Ghahramani, Z. (2015). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. https://doi.org/10.48550/ARXIV.1506.02142

GlobeNewswire. (2021). *Palomar holdings, inc. sponsors torrey pines re catastrophe bond*. GlobeNewswire. https://www.globenewswire.com/news-release/2021/03/31/2202863/0/en/Palomar-Holdings-Inc-Sponsors-Torrey-Pines-Re-Catastrophe-Bond.html

Graves, A. (2011). Practical variational inference for neural networks. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, & K. Weinberger (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2011/file/7eb3c8be3d411e8ebfab08eba5f49632-Paper.pdf

Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, *37*(1), 388–427. https://doi.org/10.1016/j.ijforecast.2020.06.008

Jospin, L. V., Buntine, W. L., Boussaıd, F., Laga, H., & Bennamoun, M. (2020). Hands-on bayesian neural networks - a tutorial for deep learning users. *CoRR*, *abs/2007.06823*. https://arxiv.org/abs/2007.06823

Lillicrap, T. P., & Santoro, A. (2019). Backpropagation through time and the brain [Machine Learning, Big Data, and Neuroscience]. *Current Opinion in Neurobiology*, *55*, 82–89. https://doi.org/https://doi.org/10.1016/j.conb.2019.01.011

McCulloch, 2.), W.S. (1, & Pitts, 2.), W. (1. (1990). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, *52*(1-2), 99–115. https://docelec.univ-lyon1.fr/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-0025112578&lang=fr&site=eds-live

Minsky, M., & Papert, S. A. (2017). *Perceptrons: An introduction to computational geometry*. The MIT Press.

Nguyen, A., Yosinski, J., & Clune, J. (2014). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. https://doi.org/10.48550/ARXIV.1412.1897

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *International conference on machine learning*, 1310–1318.

Recurrent neural networks [Accessed: 2022-04-20]. (n.d.).

Shao, J., & Papaioannou, A. (2015). Catastrophe risk bonds with applications to earthquakes. *European Actuarial Journal*, *5*. https://doi.org/10.1007/s13385-015-0104-9

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

Telesca, L. (2018). 4 - fractal, informational and topological methods for the analysis of discrete and continuous seismic time series: An overview. In T. Chelidze, F. Vallianatos, & L. Telesca (Eds.), *Complexity of seismic time series* (pp. 95–139). Elsevier. https://doi.org/https://doi.org/10.1016/B978-0-12-813138-1.00004-3

Toulemonde, G. (2017). *Théorie des valeurs extrêmes et gestion des risques extrêmes*. https://docplayer.fr/179133972-Theorie-des-valeurs-extremes-et-gestion-des-risques-extremes.html2017

Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: Lstm cells and network architectures. *Neural Computation*, *31*(7), 1235–1270. https://doi.org/10.1162/neco_a_01199

Appendix

A. Project management

For this project, we used both R and Python. Python was used for the neural networks models whereas R was used for statistical analysis and modelling. We hosted our code on Github which allowed us to better track our progress and run our models on our own machines. The repository can be found here.

For sharing our progress, we used on the one hand, Google drive for the documents and on the other hand Github for the codes. The sharing of information was fluid as we met almost every day during class and were in constant contact via messenger. Moreover, on an almost weekly basis, we held meetings online using teams, zoom or messenger. We also held online meetings with our supervising teacher Mr. Couloumy, to present our work and receive his guidelines.

All three of us wanted to develop our skills in machine learning, finance and Cat Nat modelling, so we found in this subject the perfect fulfillment of our aspirations. We spent more than half of the allotted time documenting, defining and structuring our project. So, with the deadline approaching, we decided to refocus our work on forecasting the intensity and frequency of earthquakes, which plays a central role in the pricing of cat bounds in an incomplete market.

B. Code

Here are some executable part of our code, hosted on Colab:

- Python Script - Data acquisition
- Python Script - BLSTM with MC Dropout
- Python Script - BLSTM with BBB
- Python Script - Multioutput LSTM, one month prediction
- R Script - Magnitude Density Approach
- Python Script - Magnitude Density Prediction
- Frequency Statistical Modeling