Hurricane trajectory prediction using LSTM

Sangeetha Viswanathan Sakthivel
A59013026
svsakthivel@ucsd.edu

Radhika Anish Mathuria
A59010223
rmathuria@ucsd.edu

Hari Prasad Sankar
A59003560
hsankar@ucsd.edu

Abstract

In this project, we predict the trajectory of an Atlantic hurricane by using a deep learning approach. As we are processing time series based geo-features, we used a Long Short Term Memory network to predict the next 5 positions of the hurricane, with each location 6 hours apart. We made use of the HURDAT2 database. We also modeled the problem as a Markov Transition Model along with using the conventional LSTM. We tested the model with storms from the database and used an MSE loss function with an Adam Optimizer to train the model. The github repository for the project is on link (click to follow).

1 Introduction

1.1 Background and Motivation

Natural disasters cause a lot of damage, both in terms of precious lives as well as property, mainly due to their uncertainty. Further, due to the rise in global warming, such disasters are even more frequent now. Timely prediction of these disasters is the need of the hour. As Artificial Intelligence techniques have found applications in all realms of science, they have also proven useful in predicting the occurrence and trajectories of natural disasters. As per the Office for Coastal Management, National Oceanic and Atmospheric Administration (United States of America): tropical cyclones (or hurricanes) have caused the most damage amongst all other natural disasters. They are also responsible for the highest number of deaths between 1980s and the current age. This is the main motivating factor for our project. If deep learning can be used for this physical application, it would be very beneficial.

1.2 Overview of the project

In this project, we try to predict the trajectory of the hurricane for a 30 hr interval in advance based on 5 already present data using a deep learning approach. We used the database on Atlantic hurricanes maintained by National Oceanic and Atmospheric Administration named as HURDAT. We develop a Markov Transition Model to additionally create probabilistic features based on a grid-based model of the earth where the Atlantic hurricanes have hit.

1.3 Contributions of the project

The key contributions of this project are as follows:

- Devised a deep learning approach to predict the trajectories of hurricanes.
- Creation of additional features from the available data based on geographical concepts of hurricanes.
- Used a different grid size for Markov Transition Model to extract the required transition probabilistic feature of hurricane from the data.
- Modeled a multi-input-multi-output Long Short Term Memory network and also an RNN to predict the next five locations of a hurricane, given it's first five locations

- Compared various combinations of scenarios by trying different optimizers (Adam, RM-SProp, SGD)

2 Related Works

Hurricane Trajectory prediction using deep learning has been a well studied field in the literature. Both [1] and [2] used LSTM modeling to predict the trajectory. In particular,the paper done by NIST [2] approaches the problem from a probabilistic point of view wherein the whole hurricane prone area is divided into grids and is given a probability score based on empirical data. These probability scores are passed along with the other geo-features for prediction purposes. We feel this feature would be very useful in the prediction. [2] although uses a similar grid based based RNN modeling does not use the probabilistic approach and uses the grid locks to improve the performance by reducing the truncation error in predicting the co-ordinates as minor error can propagate.

3 Problem Formulation

Consider a set of N-ordered locations of a given hurricane, recorded at fixed time-steps of h hours.

$$\text{Given: } \{L_{h_1}, L_{h_2}, L_{h_3}....L_{h_{N1}}\} \tag{1}$$

The problem in hand is to find the subsequent K locations which the hurricane will be in next T hours divided equally into T/N hours.

$$\text{Predict: } \{L^{'}_{h_{N1+1}}, L^{'}_{h_{N1+2}}, L^{'}_{h_{N1+3}}....L_{h_{N1+K}}\} \tag{2}$$

The cost function to be considered when solving the problem is,

$$\frac{1}{N}\min_{L_i}(g_i - L_i)^2 \tag{3}$$

where, g_i represents the ground truth latitude and longitude of the storm and L_i represents the estimated latitude and longitude at time $t+1$.

This problem can be reframed as a probabalistic problem too, where we intend to maximise the following function:

$$\text{argmax } P(X_{t+1} = L_i | X_t = g_t, X_{t-1} = g_{t-1}....X_1 = g_1) \tag{4}$$

But, the problem can be considered as a stochastic Markov process where the future state is only dependent on the current state and not on the previous time instances. Therefore, we can simplify the problem as follows:

$$\text{argmax } P(X_{t+1} = L_i | X_t = g_t) \tag{5}$$

The goal of the problem is to detect the incomplete points in the trajectory, given partial knowledge of some points. At each time step, we wish to learn a function that can learn from the past values of the sequence, and predict the next required points by minimizing the error.

3.1 How to predict the trajectory of a sequence using Machine Learning?

Recurrent Neural Networks (RNNs), especially LSTMs are designed to solve this problem. These networks take input variables like ordinary MLPs, but additionally use internal variables, which contain an internal state. These are responsible for accumulating the previous values of the sequence.

The sequence prediction problem can be modeled in multiple ways such as:
1) One to one (O2O)

This is not a very strong choice for sequence prediction as the model is not allowed to learn over multiple past values which could be beneficial for it.
2) One to many (O2M)
This is popularly used in giving captions for images, where an image is taken as input, and a series of words (the caption) is generated.
3) Many to one (M2O)
This takes multiple inputs, and predicts the next single time step of the sequence.
4) Many to many (M2M)
In this case, multiple time steps are taken as input, and the model produces multiple outputs. It is especially useful in time series forecasting. In our problem statement, we are creating an M2M LSTM.

3.2 Recurrent Neural Networks

The nature of Recurrent Neural Networks allows it to be trained for sequential data.

The forward pass of an RNN works in the same way as a multi-layer perceptron. At the hidden layer, information from previous output units is also taken as input. Backpropagation uses both the current and prior inputs as input, which can be seen in the below figure (xi is input, bh is from previous outputs). The equations for this are given as below:

$$out_h^t = \sum_{i=1}^{l} w_{ih} x_i^t + \sum_{h'=1}^{H} w_{h'h} b_{h'}^{t-1} \tag{6}$$

$$b_h^t = \theta_h(a_h^t) \tag{7}$$

RNNs face vanishing gradient problem. LSTM is a special kind of RNN which solves this and can handle long term dependencies. LSTM too has a repeating structure like RNN, but the structure is different with the presence of 3 gates to control the flow of the information.

4 Method/Approach

We are using a Long Short Term Memory network to solve this trajectory prediction problem. LSTMs are known to outperform RNN.

4.1 Lambert's conformal conic Projection

We flatten the surface using Lambert's conformal conic projection shown in Fig. 1.Since we want to preserve the metric space, we choose Lambert's conic projection. Hence, even though distance is distorted a little bit, the shapes of the landmass and the angles are not misrepresented. We choose 33°N and 45°N as our reference parallels, as the region between these covers the Atlantic ocean.

$$\begin{aligned} x &= \rho \sin\left[n\left(\lambda - \lambda_0\right)\right] \\ y &= \rho_0 - \rho \cos\left[n\left(\lambda - \lambda_0\right)\right] \end{aligned} \tag{8}$$

$$\begin{aligned} F &= \frac{\cos\phi_1 \tan^n\left(\frac{1}{4}\pi + \frac{1}{2}\phi_1\right)}{n} \\ n &= \frac{\ln\left(\cos\phi_1 \sec\phi_2\right)}{\ln\left[\tan\left(\frac{1}{4}\pi + \frac{1}{2}\phi_2\right)\cot\left(\frac{1}{4}\pi + \frac{1}{2}\phi_1\right)\right]} \\ \rho &= F \cot^n\left(\frac{1}{4}\pi + \frac{1}{2}\phi\right) \\ \rho_0 &= F \cot^n\left(\frac{1}{4}\pi + \frac{1}{2}\phi_0\right). \end{aligned} \tag{9}$$

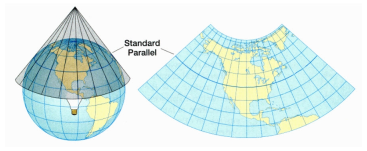


Figure 1: LCC projection is used to convert the field of operation from 3D to 2D. Source: commons.wikimedia.org/wiki/File:Lambertconformalconic.svg

After this, we extract some additional features to further strengthen the model.

4.2 Feature Extraction

With reference to [2],At the core, there are two important features that we extracted, Translational velocity (V) Translational Direction (θ). These parameters are computed as follows :

$$\theta_l = \tan^{-1} \frac{\phi_l - \phi_{l-1}}{\theta_l - \theta_{l-1}} \tag{10}$$

$$V_l = \frac{d(\phi_{l-1}, \phi_l, \lambda_{l-1}, \lambda_l)}{6hrs} \tag{11}$$

where, the distance d is computed using the haversine formula, which lets us compute the great-circle distance between 2 points on earth. Apart from this, the HURDAT2 database also contains wind speed,ocean temperature.

4.3 Markov Transition Model

A plain geo-feature based statistical model doesn't give that much information about the future storm location because instantaneous storm features are not strong enough heuristics to predict an uncertain phenomenon like a hurricane. As a result, some form of additional features that bias the prediction based on previous storms need to be fed to the model. We do this by using a Markov transition model. For the purpose of features, we planned to pass K^2 possible locations each with a probability of P_k as features. The model is trained with these probability features for each prediction output in addition to the normal features. A 5x5 grid is chosen as shown in Fig. 2 with the current location as the centre. This grid will divide the whole Latitude and Longitude area into equally spread areas with each cell bounded by 4 latitude longitude pairs.

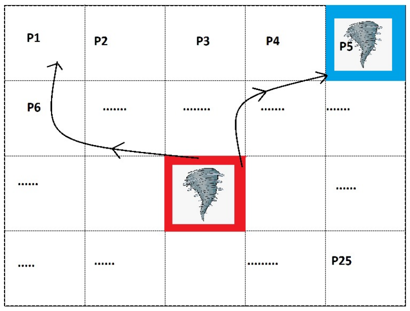


Figure 2: Calculation of probabilities from the grid model

The probability that a storm will go to a particular latitude and Longitude in that 5x5 grid is given by the probability extracted from number of storms in the database having been to that particular cell.

$$\begin{array}{ll} P_{(i,j)k} = n_k/p & \text{for } p \neq 0 \\ P_{(i,j)}(k) = 1/m & \text{for } p = 0 \end{array} \tag{12}$$

Where i,j represent the current storm location in the matrix cell,'k' represents 24 subcells surrounding the cell (i,j)

$$\sum_{k=1}^{k=m} n_k = p \tag{13}$$

(13) just reaffirms the 3^{rd} axiom of probability that the storm in cell (i,j) can only go to one of the 24 cells in the grid in consideration.

Algorithm: Markov state transition Matrix

Input : LCC projection Matrix containing co-ordinates of Hurricanes
output : Markov state transition Matrix

```
1. Remove the outliers in the dataset i.e. values that are far off from the majority of values.
2. starting point (s1,s2) $\leftarrow$ min(LCC X co-ordinates, LCC Y co-ordinates)
3. step size $\leftarrow 10^5$
4. create a window ABCD
       A $\leftarrow$ (s1,s2)
       B $\leftarrow$ (s1,s1+step size)
       C $\leftarrow$ (s1+step size, s2+ step size)
       D $\leftarrow$ (s1+step size,s2).
5. Slide the window in x-axis to find the number of storms 'N' in the current window interval.
6. Markov storm matrix $\leftarrow$ N.
7. if window reaches the maximum LCC X co-ordinates bounds, do,
       if the window is not bounded by maximum LCC Y co-ordinates,
           move the window in y-axis by one step size. Perform step 4 $\rightarrow$ 6 again.
       else, Markov state transition matrix is complete.
```

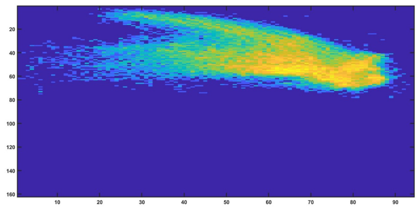


Figure 3: Heatmap of the Markov Transition Probability matrix

4.4 Loss function

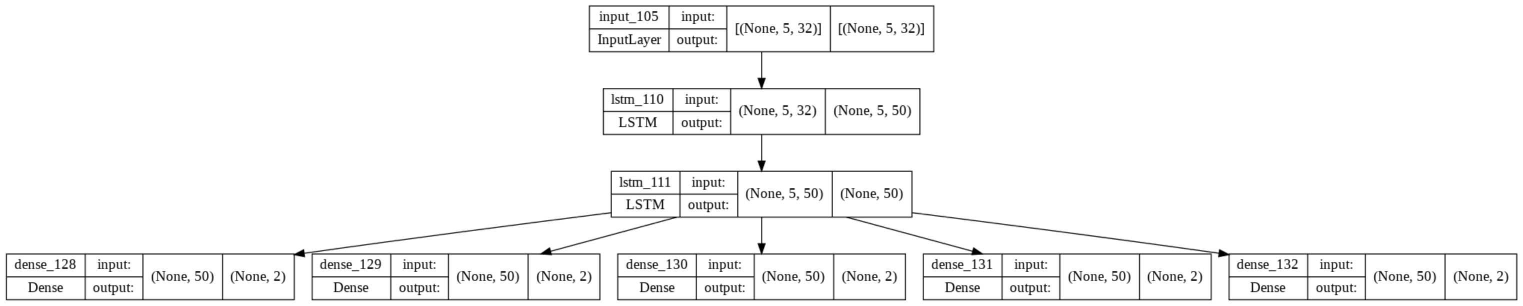
In many/one-to-one models, one can measure the loss by comparing the prediction value and actual value(yy). But in many-to-many models, each output node can measure the loss. There are many losses, so common ways to use it for training is by averaging whole losses. This kind of loss is called sequence loss. After that, specific optimizer (SGD or Adam) tries to minimize the loss with backpropagation.

We choose the Mean Squared Error (MSE) loss as it is known to be well-suited for regression problems like this, and it penalizes large errors significantly more than smaller ones due to the quadratic nature. Also, when we did our data analysis, we found most of the data was normally distributed, and hence MSE is a good choice since it inherently assumes that data is gaussian.

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} (y - \hat{y}_i)^2 \tag{14}$$

4.5 Architecture

The LSTM model architecture as shown in Fig. 4 has a stacked layer structure. It has 2 LSTM layers with the 1st layer returning a sequence. It is then followed by 5 dense layers corresponding to output for the next 5 timesteps. The activation function used for the hidden layers in the tanh function. LSTM with 2 layers is generally good to detect complex features and our data doesn't has a very complex structure, so 2 layers is optimum.



input_105 | InputLayer | input: [(None, 5, 32)] | output: [(None, 5, 32)]
lstm_110 | LSTM | input: (None, 5, 32) | output: (None, 5, 50)
lstm_111 | LSTM | input: (None, 5, 50) | output: (None, 50)
dense_128 | Dense | input: (None, 50) | output: (None, 2)
dense_129 | Dense | input: (None, 50) | output: (None, 2)
dense_130 | Dense | input: (None, 50) | output: (None, 2)
dense_131 | Dense | input: (None, 50) | output: (None, 2)
dense_132 | Dense | input: (None, 50) | output: (None, 2)

Figure 4: The LSTM model architecture developed in Keras

5 Results

5.1 Dataset

About the dataset The HURDAT2 dataset contains comma delimited information on location, pressure, winds and other features collected over intervals of 6 hours. There are 1814 storms in HURDAT2, with a total of around 49,0000 entries. The database contains different attributes related to the storm, some of which are date, time, latitude, longitude, maximum wind, minimum pressure and information related to the winds in various cardinal directions. A scatter plot of the latitude and longitude points in the dataset in shown in Fig. 5. The dataset can be found here (click to follow).

Pre-processing Some storms had very limited number of datapoints, due to which we eliminate some storms in our data pre-processing.

In machine learning problems, the range of features passed may vary greatly. In order to create uniformity while training, and speed up the training and convergence of the network, the data is often scaled. For the data we are giving as input to the Markov-LSTM model, we are using min-max normalisation. This is given as:

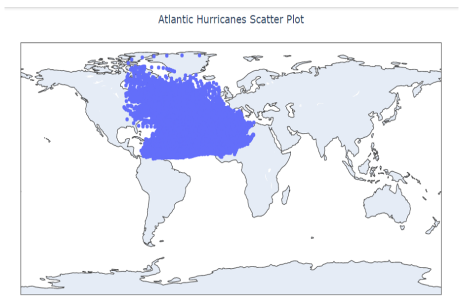


Figure 5: Scatter plot of the storm points

$$f_{normalised} = \frac{f - f_{min}}{f_{max} - f_{min}} \tag{15}$$

Train-validation-test split The split we used was 80-10-10, which is generally used in ML problems.

5.2 Hyperparameter tuning and Model performance

Choice of optimizer: We compared the performance of various optimizers, Stochastic Gradient Descent (SGD), RMSProp and Adam optimizer. Adam optimizer is one of the most chosen optimizers as it makes use of the momentum concept from SGD with momentum and on top of that, has an adaptive learning rate like RMSProp. It is also computationally efficient.

The results we get are also in line with this intuition, i.e. the Adam Optimizer outperforms RMSProp and SGD. This is shown in the following fig. 6,7,8:

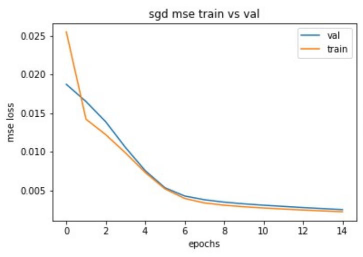


Figure 6: SGD

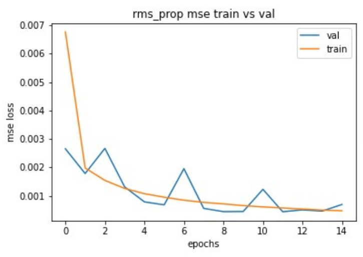


Figure 7: RMSProp

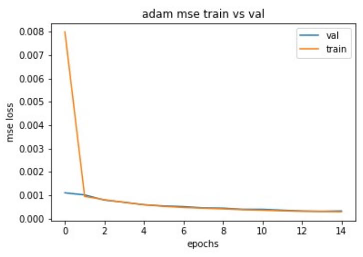


Figure 8: Adam

Learning rate and epochs For these 2 hyperparameters, we ran grid search in Keras. The learning rate used is 0.001 and the number of epochs are 15. We can see from the loss figures that this is an optimal choice as the loss almost stabilizes after 10 epochs.

Results and Discussion We tested the model with selected storms from the database and Table 1 reports the average error for each predicted storm. Depending on the number of entries for each storm, we predicted the next 7-20 time steps, using the current 5 time steps. The average error is in the range of 200-500 kilometers. Fig. 9 and 10 give the error v/s time step, and we can see that initially the error is smaller, and with each next time step, it increases since the error propagates.

Sr. No.	**Storm Name**	
1.	Unnamed 1	295.17700
2.	Unnamed 2	493.35302
3.	Unnamed 3	523.5334
4.	Kendra	444.70133
5.	Laurie	210.20983
6.	Martha	459.30075
7.	Twenty Two	283.45815
8.	Wilma	297.09255
9.	Zeta	260.7159

Table 1: Average prediction Error(kms) for storms in HURDAT2

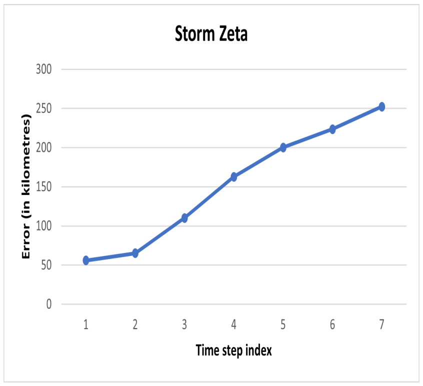


Figure 9: Storm Zeta

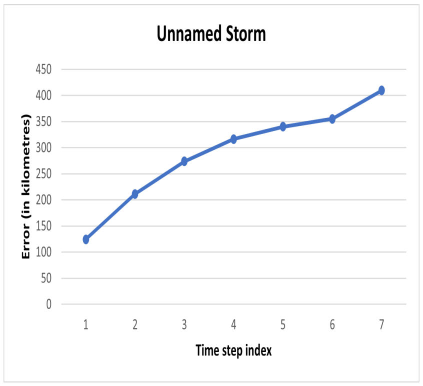


Figure 10: Unnamed Storm

We have also visualised the predicted trajectory v/s the actual trajectory of a hurricane using geoscatter in MATLAB as shown in Fig. 11.

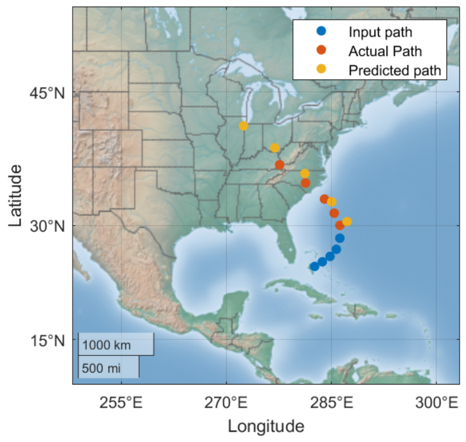


Figure 11: Visualisation of the predicted trajectory of a hurricane

6 Future Scope of works

The project had followed a probabilistic approach in finding the trajectory of Hurricane. The model was implemented on a vanilla version of LSTM/RNN. This can be further improved by trying with Gated recurrent Neural network or Multiplicative LSTM to see how the performance varies with different architecture. Also, a more comprehensive dataset consisting of storms from pacific can also be used to predict the hurricane trajectory in both side of the coasts.

7 References

[1] Alemany, S, Beltran, J, Perez, A, & Ganzfried, S, Predicting Hurricane Trajectories Using a Recurrent Neural Network, Proceedings of the AAAI Conference on Artificial Intelligence, no. 33, pp. 468-475, 2019.

[2] Bose, R , Pintar, A and Simiu, E, Forecasting the Evolution of North Atlantic Hurricanes: A Deep Learning Approach, Technical Note (NIST TN), National Institute of Standards and Technology, Gaithersburg, MD, 2021.