Active and dry spell Detection of Indian Monsoon

14.06.2018

—

V.Saicharan
NITK,Surathkal

Introduction

Indian monsoon is a complex non-linear phenomenon, and prediction of monsoon spells is a very challenging problem. A lot of research has been carried out in predicting the monsoon patterns over the Indian subcontinent. This work mainly focuses on predicting the phenomenon of active and dry spell patterns at a daily level.

Fluctuations in rainfall are mainly characterized by active and break spells of the Indian summer monsoon. This phenomenon is an important component of the rainfall variability and has a large impact on agricultural production and hence the economy of the country.

There have been several studies of breaks and also active spells in several cases identified on the basis of different criteria over regions differing in spatial scales. However, not much work has been carried out in predicting active and dry spells at a daily level.

On a daily level, a wet day was defined as when rainfall value is more than (mean + std) and a dry day is defined as when rainfall is lesser than (mean - std). 3 or more consecutive wet days are considered as an active spell and 3 or more consecutive dry days are considered as a dry spell.

We approach this problem by treating it as a multi-step time series classification problem. Our aim is to predict the occurence of an active/beak spell multiple time steps into the future given past data of rainfall and other variables.

Goals

1. To predict occurrence of active and dry spells at a daily level over Central Indian (21°-27°N and 72°-85° E) region at different lead times spanning from 1-5 days.

2. To compare and analyse different strategies used in prediction.

Problem definition

Time Series forecasting

The task of predicting active and dry spells at a lead time can be viewed as a time series classification problem. A time series is a series of data points indexed in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time.

Time series forecasting is the use of a model to predict future values based on previously observed values.

Multivariate time series, in particular uses multiple features to predict a value in future time-step.The factors affecting the rainfall on a particular day may not only comprise of previous rainfall values but can also include factors like temperature, sea level pressure,wind velocity etc.

The given problem of predicting active and dry spells can be viewed as a multi step forecasting problem.

Multi step forecasting can be described as follows:-

Given a time series $\{X_1, X_2, \ldots X_T\}$, T observations of the time series , we want to forecast the next L values of the given time series, $\{Y_{(T+1)}, Y_{(T+2)}, \ldots Y_{(T+L)}\}$. When X_i is a single real value it is a Univariate time series problem , and when X_i is a vector it is formulated as a multivariate time series problem.

Strategies involved

Multi step time series forecasting mainly uses two methods :-

1. Direct strategy
2. Recursive Strategy

Direct Strategy

The direct (or Independent) strategy consists of forecasting each time-step in the future independently from the others. The forecasts obtained by using the model is of the form :-

$$(Y_T \ldots Y_{T+L}) = f_h(X_{T-d}, \ldots, X_T)$$

This implies that the Direct strategy does not use any approximated values to compute the forecasts being then immune to the accumulation of errors.

Recursive Strategy

In this strategy, a model f is trained to perform a one-step ahead forecast.When forecasting H steps ahead, we first forecast the first step by applying the model.

Subsequently, we use the value just forecasted as part of the input variables for forecasting the next step (using the same one-step ahead model). We continue in this manner until we have forecasted the entire duration.

$$\hat{y}_{T-1+L} = f(X_{T-d}, \ldots, X_{T-1}) \text{ if } L = 1$$

$$\hat{y}_{T-1+L} = f(\hat{y}_{T+L-2}, \ldots, \hat{y}_T, X_{T-d}, \ldots, X_{T-1}) \text{ otherwise}$$

Depending on the noise present in the time series and the forecasting horizon, the recursive strategy. Depending on the time series, this strategy may suffer from low performance in multi-step ahead forecasting tasks due to accumulation of errors.

RNN and LSTM

A recurrent neural network (RNN) is a special case of neural network where the objective is to predict the next step in the sequence of observations with respect to the previous steps observed in the sequence. In fact, the idea behind RNNs is to make use of sequential observations and learn from the earlier stages to forecast future trends. As a result, the earlier stages data need to be remembered when guessing the next steps. In RNNs, the hidden layers act as internal storage for storing the information captured in earlier stages of reading sequential data

Recurrent neural networks(RNN) have been shown to perform well in time series modelling. A recurrent neural network deals with sequence problems because their connections form a directed cycle.However a RNN is known to suffer from the vanishing gradient problem and fails to capture long term dependencies.

LSTM is a special kind of RNN with additional features to memorize the sequence of data. The memorization of the earlier trend of the data is possible through some gates along with a memory line incorporated in a typical LSTM. The gates, which are based on sigmoidal neural network layer, enable the cells to optionally let data pass through or be forgotten.

In our work ,both direct and recursive strategies in multi-step forecasting have been implemented using RNN/LSTM models.

The direct strategy used a LSTM classification model. The recursive strategy uses a popular model called Seq2Seq which has found great uses in NLP tasks but it can also be used in multi-step time series forecasting.

Both the models have been explained in detail in the later sections.

Dataset

Data and features

The data for daily precipitation(rainfall) values was obtained from IMD(Indian Meteorological Department , Pune). Gridded rainfall data of the resolution (1x1) was available from 1948-2014 in netCDF format.The rainfall data was extracted for the period of June 1 to September 30.

Other climatic variables which are considered which affect the rainfall patterns include :

1. **Air temperature (AT)**
2. **Sea level pressure (SLP)**
3. **U Wind (UWND)**
4. **V Wind(VWND)**

All the variables considered above were at surface level.

These variables were obtained from NCEP/NCAR Reanalysis project.The NCEP/NCAR Reanalysis 1 project is using a state-of-the-art analysis/forecast system to perform data assimilation using past data from 1948 to the present.

Daily data was available in 2.5 x 2.5 gridded form and was extracted for the central Indian region for a period of June - September from 1948 - 2014. Data preprocessing

A considerable amount of time was spent into transforming the data into something useful. This included creating a MATLAB script to extract the data from NETCDF format to a useful CSV format.

The extracted data had missing values for some days. The missing values were filled with the values of the previous point in time series to maintain consistency.

Data smoothing

The obtained data for rainfall was plotted and observed.The daily anomalies contain high frequency fluctuations. To eliminate noise and high frequency fluctuations, a 1-4 day bandpass Lanczos filter with 25 weights is applied at every point of the time-series data. This results in smoothening of the data but at the same time it also preserves information about active and dry spells.

Normalising features

The extracted features which were extracted were normalised using the mean normalisation method.

After smoothing and normalising, the time series was plotted. As observed, the time series was very stochastic and had a lot of variation at daily level.
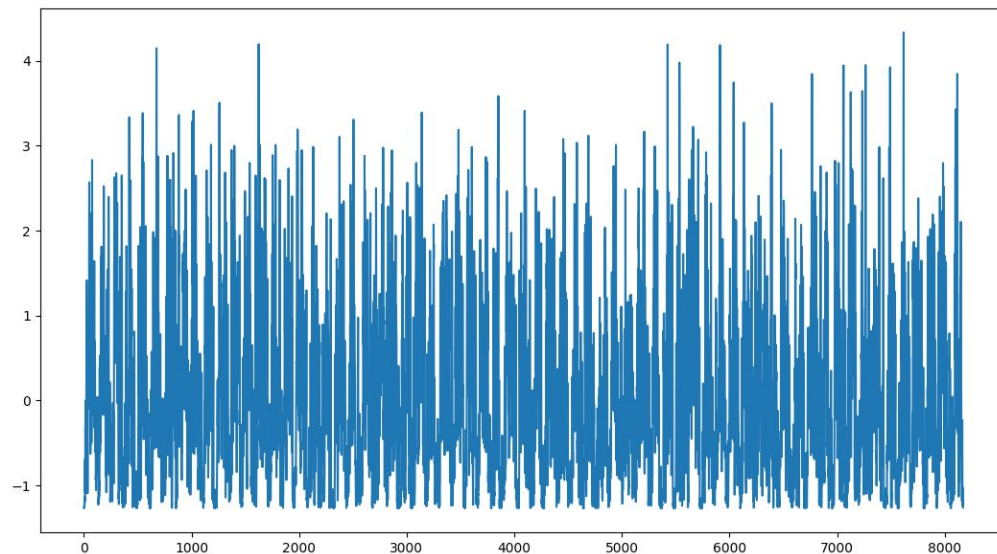
Sliding window method

Real time prediction of active/dry spells requires continuous feeding and learning. Over time, the total number of time-slots become too big resulting in high computational complexity.

To fix this problem, a sliding window method was used which is explained below. A sliding window of size TIME_STEPS is chosen which indicates a fixed number of previous time-slots to learn from in order to predict the next series of timesteps. The size of sliding window TIME_STEPS is a hyper-parameter to be tuned.

Sliding window example
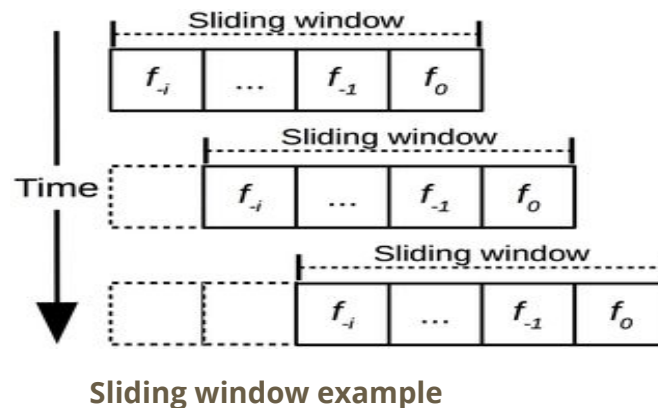
Time series flattening

For our datasets, each sample consisted of a TIME STEPS× NUM FEATURES matrix where each row represented the value of features on a particular day and each column represented the distinct features used in prediction.

As an example if were using RAINFALL,TEMPERATURE and SEA LEVEL PRESSURE of 4 previous days (DAY 1 to DAY 4) to predict the occurence of an active or dry spell on day 6 our data would be of the form 4 X 3 , where 4 indicates the number of TIME STEPS and 3 indicates the value of NUM FEATURES.

This data was processed and stored in a 3D matrix of the form whose dimensions were NUM EXAMPLES × NUM FEATURES × TIME STEPS.

For the algorithms that cannot take into account the temporal nature of the data, the data was flattened into a 2-D matrix of size NUM EXAMPLES × (NUM FEATURES × TIME STEPS).

Data split

After the flattening of time-series data, it was split three-fold into train, validation and test sets. The different models used were trained on the training data. Values for different hyper-parameters were chosen based on the performance on the validation set and finally the data was evaluated on the test-set.

The data-set consisted of 8174 time-steps out of which 80% of the data was used for training , 10% of the data was used for validation and the last 10% of data was used for testing the performance of the model.

Approximately, 53 years of the 67 were used for training, 7 years for validation and the last 7 years for testing.

Out of the total 8174 days in consideration:-

Total number of wet days - 1211

Total number of dry days - 1470

Total number of normal days - 5493

The number of active and dry spells was also calculated:-

Total number of active spells:- 191

Total number of dry spells :- 155

Models used

LSTM Classification model

The LSTM classification model follows the direct (independent) strategy for multi-step time series forecasting.

Model Architecture

The LSTM classification model used can be described as follows :-

- INPUT LAYER - The input to the first LSTM layer of the form (BATCH-SIZE X TIME-STEPS X NUM-FEATURES)

- HIDDEN LAYER - The LSTM model consists of 2 hidden layers with 80 neurons in each LSTM cell. The activation function used was tan-h.

- DENSE LAYER - A fully connected layer to connecting the last output of LSTM to values for different lead times.

- SOFTMAX-LAYER - Softmax activation function applied on the dense layer.

The model can be visualised as follows:-
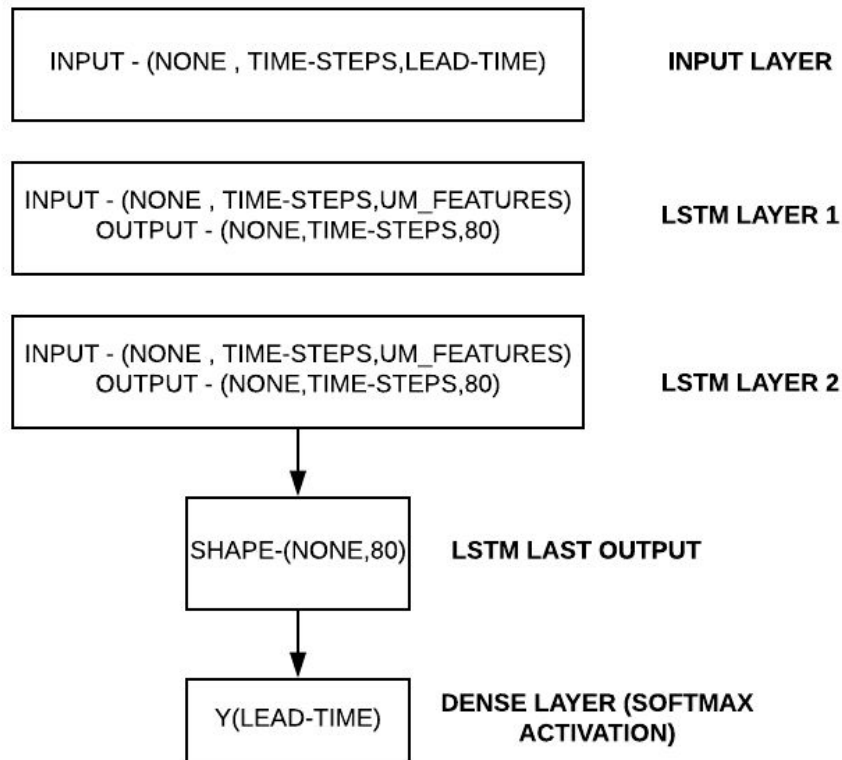
Architecture of the LSTM Classification model used

Training Procedure

The last LSTM cell in the model was used to predict the value at time T+LEAD TIME.

Alternatively,the last cell could be just used to predict the values from time step T to T+LEAD TIME but it was observed that this method does not perform as well as predicting the single value at time-step T+LEAD-TIME.
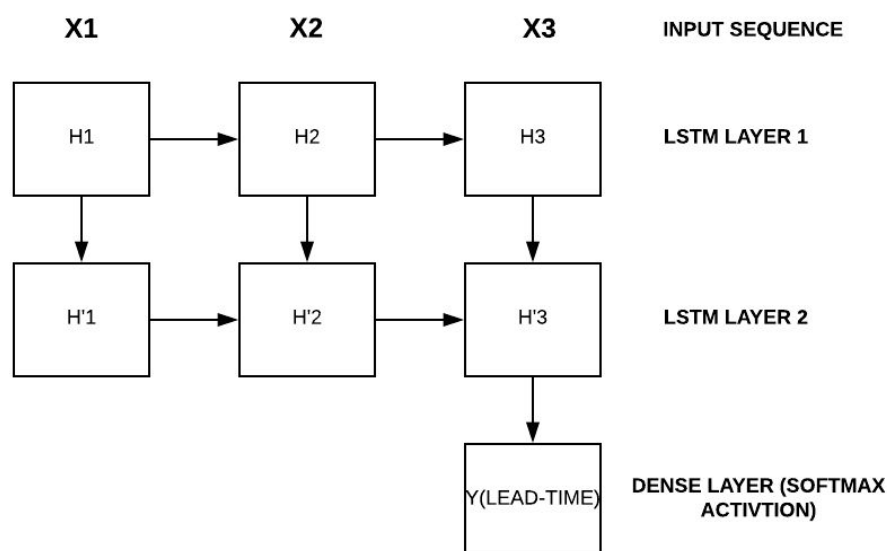
An example illustrating using 3 time steps to predict LEAD-TIME days ahead

Seq2Seq Model

Sequence-to-sequence (seq2seq) model ([Sutskever et al., 2014, Cho et al., 2014](#)) has enjoyed wide success in the field of NLP, particularly in Neural Machine Translation.

The Sequence to Sequence model (seq2seq) consists of two RNNs - an encoder and a decoder. The encoder reads the input sequence, and emits a context (a function of final hidden state of encoder), which would ideally capture the essence (semantic summary) of the input sequence. Based on this context, the decoder generates the output sequence, one word at a time while looking at the context and the previous output during each timestep.
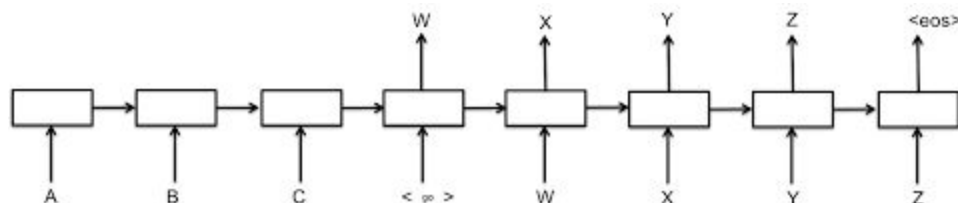
Attention mechanism

Attention mechanism is an extension to the Seq2Seq model proposed by

If only the last encoder is passed between the encoder and decoder, that single vector carries the burden of encoding the entire input sequence.Attention allows the decoder network to "focus" on a different part of the encoder outputs for every step of the decoder's own outputs. First a set of *attention weights* is calculated . These will be multiplied by the encoder output vectors to create a weighted combination. The result should contain information about that specific part of the input sequence, and thus help the decoder choose the right output words.
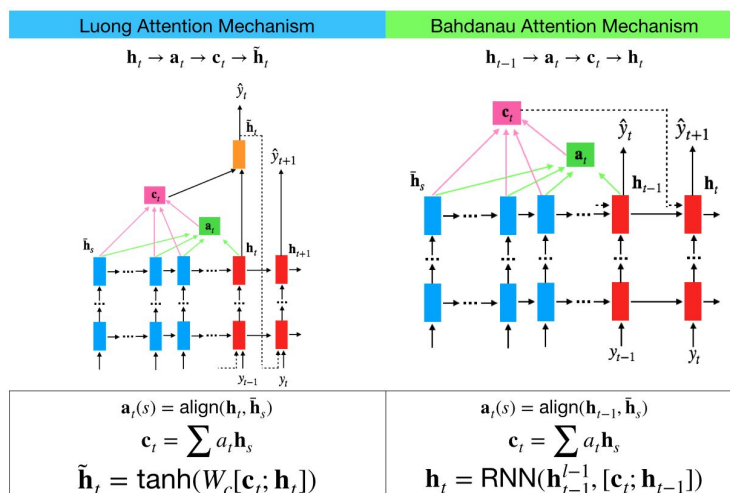
Model Architecture

The Seq2Seq model used can be described as follows :-

Implementing Seq2Seq with attention mechanism improves the performance of the model. Thus in our case, it was implemented in the same way.

- INPUT LAYER - The input to the first Encoder LSTM layer of the form (BATCH-SIZE X TIME-STEPS X NUM-FEATURES)

- HIDDEN LAYER - The model consists of 2 hidden layers with 100 neurons in each LSTM cell. The activation function used was tan-h.

● DENSE SOFTMAX-LAYER - Softmax activation function applied on the dense layer

The model can be visualised as follows :-

```
┌──────────────────────────────────┐
│  INPUT SHAPE - (NONE , TIME-STEPS , │        INPUT LAYER
│          NUM_FEATURES)             │
└──────────────────────────────────┘
              │
              ▼
┌──────────────────────────────────┐
│  INPUT SHAPE - (NONE , TIME-STEPS , │      ENCODER LAYER 1
│          NUM_FEATURES)             │
│  OUTPUT SHAPE-(NONE , TIME-STEPS,100)│
└──────────────────────────────────┘

┌──────────────────────────────────┐
│  INPUT SHAPE - (NONE , TIME-STEPS , 100)│   ENCODER LAYER 2
│  OUTPUT SHAPE-(NONE ,TIME-STEPS,100)│
└──────────────────────────────────┘

        ┌──────────────┐
        │ LAST ENCODER │
        │   STATE -    │
        │  (NONE,100)  │
        └──────────────┘

DEOCDER LSTM LAYER 1 ┌────────────────────────────────┐        ┌────────────────────────┐
                     │ INPUT SHAPE - (NONE , LEAD-TIME , 3)│◄─── │ GO TOKEN  - SHAPE(NONE,3)│
                     │ OUTPUT SHAPE-(NONE ,LEAD-TIME,100)│       └────────────────────────┘
                     └────────────────────────────────┘

DEOCDER LSTM LAYER 2 ┌────────────────────────────────┐
                     │ INPUT SHAPE - (NONE , LEAD-TIME , 100)│
                     │ OUTPUT SHAPE-(NONE ,LEAD-TIME,100)│
                     └────────────────────────────────┘

DENSE LAYER (SOFTMAX  ┌────────────────────────────────┐
ACTIVATION)           │ INPUT SHAPE - (NONE , LEAD-TIME , 100)│
                     │ OUTPUT SHAPE-(NONE ,LEAD-TIME,3)│
                     └────────────────────────────────┘
```

Architecture of the Seq2Seq Classification model used

Training Procedure

The input sequence is fed to the first encoder layer. The last cell of the encoder stores the contextual information related to the input sequence. It is fed to the decoder during the decode stage.

During decoding steps, a **'GO'** token is fed as initial input to decoder **(**this can be a zeros vector), and subsequently, provide the output of the decoder at the previous time-step T as

input to the decoder at the next time-step T+1. This method is known as Greedy sampling and makes the model more robust to learn from previous mistakes.
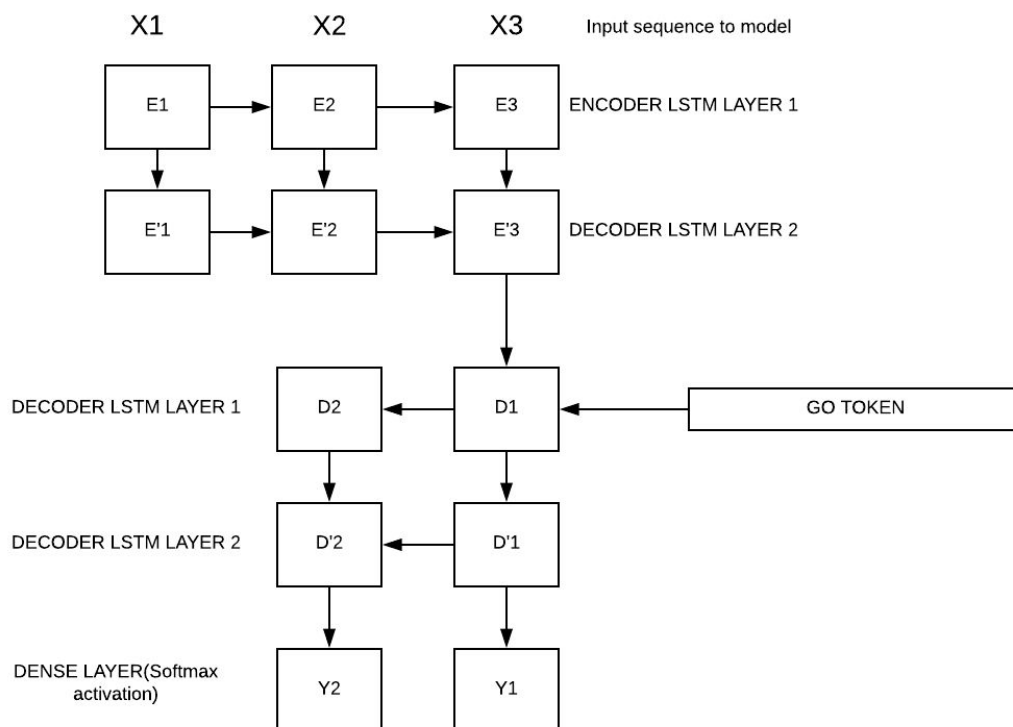
An example illustrating using Seq2Seq model 3 time steps to predict 2 days ahead

Experiments

Hyperparameters

With all the model several experiments were conducted to arrive at the optimal values for various hyper-parameters. The values were chosen based on the performance on the validation set.

Hyper-parameters for the model included the number of layers, the number of neurons in each LSTM cell and the activations functions used.

For training , the learning rate value was set to 1e-3 and Mini-batch gradient descent was used with a batch size of 16. AS an optimizer for the network Adam Optimizer was used.

Since overfitting was a major concern, dropout was implemented on the hidden LSTM to reduce overfitting.

Loss function

Since this particular problem could be modelled as a multi-class classification , a softmax cross entropy loss was used.

However, there was a problem of class imbalance. Out of the total days considered in the data , 15% were found to be active spells , 18% to be break spells and 67% to be regular spells.

To solve the issue of class imbalance , weighted softmax cross entropy loss was considered. Weighted softmax loss allows to use different scores for different classes, and hence a misclassification of the less common classes can be penalised to improve prediction accuracy. The weights for penalising class scores were considered as hyper-parameters.

Another aspect regarding the loss function in case of Se2Seq model was that it was a combination of two losses

1) average loss of each time step prediction - Loss 1

2) loss of the prediction calculated at the last time step. - Loss 2

Beta in the combined loss function is a hyper-parameter.

Total loss = Beta * (Loss 1) + (1-Beta) * Loss 2 + L2 regularization loss.

Performance metric

Since the problem is a multi-class classification problem, to assess the performance of the models, F-Score was used.

It considers both the precision P and the recall R of the test to compute the score. The F score is the harmonic average of the precision and recall, where an F score reaches its best value at 1 (perfect precision and recall) and worst at 0.

P is the number of correct positive results divided by the number of all positive results returned by the classifier, and R is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

The F score for multi-class classification problem can be calculated as the weighted average of the F score for each class.

Results and Analysis

The performances of the LSTM classification model and the Seq2Seq model were analysed by comparing it with other models which included SVM and KNN classifiers.

For the SVM Model various kernels were implemented but the linear kernel was found to give the best results.

For the KNN model the number of neighbours considered was considered as a hyper-parameter and varied with LEAD-TIME.

The comparison was carried out for lead times ranging from 1-5 days.

While the KNN algorithm performed poorly, the SVM and LSTM classification model achieved almost similar results. The Seq2Seq model with attention seemed to perform the best and gave good results upto 5 days ahead.

Results

Lead 1

	SVM(Linear Kernel)	KNN	LSTM Classification model	Seq2Seq Model
Dry Day Precision	0.931	0.81	0.95	0.95
Dry day Recall	0.955	0.942	0.95	0.98

Wet day Precision	0.841	0.74	0.87	0.856
Wet day recall	0.835	0.75	0.884	0.884
Normal day precision	0.944	0.9236	0.96	0.967
Normal day recall	0.947	0.877	0.96	0.953
Overall F-Score	0.935	0.875	0.954	0.953

Confusion matrix of LSTM model

	Dry day	Normal day	Wet day
Dry day	150	7	0
Normal day	7	516	15
Wet day	0	14	107

Confusion matrix of Seq2Seq model

	Dry day	Normal day	Wet day
Dry day	154	3	0
Normal day	8	513	18
Wet day	0	14	107

Lead 2

	SVM(Linear Kernel)	KNN	LSTM Classification model	Seq2Seq Model
Dry day Precision	0.869	0.756	0.885	0.8988
Dry day Recall	0.936	0.770	0.936	0.9617
Wet day Precision	0.807	0.651	0.803	0.85
Wet day recall	0.752	0.462	0.842	0.801
Normal day precision	0.925	0.82	0.944	0.943
Normal day recall	0.921	0.87	0.92	0.936
F-Score	0.903	0.795	0.910	0.926

Confusion matrix of LSTM model

	Dry day	Normal day	Wet day
Dry day	147	10	0
Normal day	19	494	25
Wet day	0	19	102

Confusion matrix of Seq2Seq model

	Dry day	Normal day	Wet day
Dry day	151	6	0
Normal day	17	504	17
Wet day	0	24	97

Lead 3

	SVM(Linear Kernel)	KNN	LSTM Classification model	Seq2Seq Model
Dry day Precision	0.793	0.633	0.809	0.909
Dry day Recall	0.808	0.596	0.840	0.834
Wet day Precision	0.784	0.319	0.782	0.796
Wet day recall	0.67	0.25	0.652	0.744
Normal day precision	0.871	0.74	0.88	0.90
Normal day recall	0.897	0.78	0.901	0.93
F-Score	0.849	0.667	0.859	0.890

Confusion matrix of LSTM model

	Dry day	Normal day	Wet day

Dry day	132	25	0
Normal day	31	485	22
Wet day	0	42	79

Confusion matrix of Seq2Seq model

	Dry day	Normal day	Wet day
Dry day	131	26	0
Normal day	13	502	23
Wet day	0	31	90

Lead 4

	SVM(Linear Kernel)	KNN	LSTM Classification model	Seq2Seq Model
Dry day Precision	0.696	0.54	0.670	0.751
Dry day Recall	0.701	0.509	0.670	0.808
Wet day	0.536	0.219	0.614	0.686

Precision				
Wet day recall	0.363	0.190	0.421	0.578
Normal day precision	0.784	0.70	0.8	0.836
Normal day recall	0.83	0.74	0.855	0.862
F-Score	0.745	0.6157	0.769	0.814

Confusion matrix of LSTM model

	Dry day	Normal day	Wet day
Dry day	112	45	0
Normal day	45	460	32
Wet day	0	70	51

Confusion matrix of Seq2Seq model

	Dry day	Normal day	Wet day
Dry day	127	30	0
Normal day	42	464	32
Wet day	0	51	70

Lead 5

	SVM(Linear Kernel)	KNN	LSTM Classification model	Seq2Seq Model
Dry day Precision	0.641	0.434	0.681	0.744
Dry day Recall	0.605	0.318	0.60	0.707
Wet day Precision	0.333	0.301	0.525	0.531
Wet day recall	0.272	0.231	0.34	0.413
Normal day precision	0.74	0.68	0.761	0.8
Normal day recall	0.797	0.765	0.845	0.84
F-Score	0.684	0.603	0.717	0.759

Confusion matrix of LSTM model

	Dry day	Normal day	Wet day
Dry day	94	63	0
Normal day	44	456	37
Wet day	0	80	41

Confusion matrix of Seq2Seq model

	Dry day	Normal day	Wet day
Dry day	109	46	0
Normal day	38	456	44
Wet day	0	71	50

Extension

An example of predicting active and dry spells from daily rainfall classification. Active spells were defined as 3 or more consecutive days with rainfall above (std + mean). Dry spells were defined as 3 or more consecutive days with rainfall above (std - mean).

The number of active and dry spells during the test period were noted and found to be 21 active spells and 16 dry spells. The duration of the activee and dry spells along with the prediction of the proposed LSTM and Seq2Seq model is summarised in the table below.

LEAD 3

	Length of active spell	LSTM predictions intersections	Seq2Seq predictions intersections
Active spell 1	3	2	3
Active spell 2	3	2	2
Active spell 3	3	3	3
Active spell 4	4	4	4
Active	3	3	3

spell 5			
Active spell 6	4	1	2
Active spell 7	4	1	3
Active spell 8	8	5	6
Active spell 9	6	3	3
Active spell 10	3	0	2
Active spell 11	6	2	5
Active spell 12	4	2	3
Active spell 13	4	3	4
Active spell 14	4	4	4
Active spell 15	3	2	2
Active spell 16	11	7	9
Active spell 17	5	2	3
Active spell 18	6	5	5
Active spell 19	3	1	2
Active spell 20	4	3	4
Active spell 21	4	2	2

LEAD 3

|	Length of active spell	LSTM predictions intersections	Seq2Seq predictions intersections
Dry spell 1	30	28	28
Dry spell 2	10	6	8
Dry spell 3	13	13	13
Dry spell 4	7	7	6
Active spell 5	5	3	4
Dry spell 6	14	13	13
Dry spell 7	16	15	15
Dry spell 8	6	5	5
Dry spell 9	9	7	7
Dry spell 10	9	6	4
Dry spell 11	15	11	12
Dry spell 12	4	2	3
Dry spell	3	0	0

13			
Dry spell 14	5	3	4
Dry spell 15	3	0	2
Dry spell 16	6	4	5

It was noted that the the Seq2Seq model manages to predict the active and dry spells more accurately.

Although the LSTM model does come close, it does predict many false occurences of active and break spells which can be observed from the confusion matrix. The Seq2Seq model is also better at predicting 2 consecutive active and wet days although it has not been considered in the tables above.

Analysis

Accuracy of models
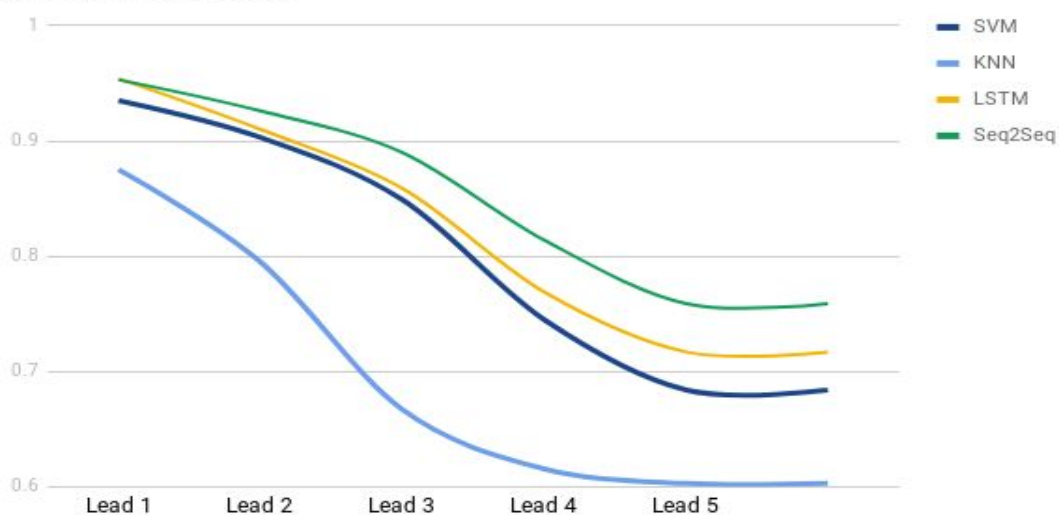
One of the important observations was that detecting active spells was harder than detecting dry spells. The possible explanation could be that an active spell usually lasted only a lesser duration of time compared to a dry spell, thus making it more random and harder to detect.

Another major problem faced during the classification process was that the data-set was imbalanced. Weighted softmax loss was implemented but the weights add to the number of hyper-parameters thus making the model harder to tune.

Conclusion

The prediction of active and dry spells of Indian summer monsoon was performed during RNN's and LSTM using two different models and compared with other simple classifiers.The Seq2Seq model with attention achieved a good accuracy upto 5 days ahead.

Although the experiment was carried out multiple times, a more extensive hyperparameter search could yield better results and could be extended for predictions more than 5-days.

The performance of the LSTM classification model and the Seq2Seq model can also be greatly improved by acquiring more data.

Future work

The major problem faced during the experiments was the imbalance of data. A possible solution could be to under-sample or over-sample data but this would result in loss of temporal meaning in the data. Any method which could handle the imbalanced classes could be implemented to improve the performance of the models.

Another possible area of extension would be to use CNN's for multivariate time series data. Although CNN's have extensively used as feature extractors in images, they could also be used to extract features from time-series. CNN's and RNN/LSTM's could also be combined in a single model.