Rainfall Prediction Using Deep Neural Network

Chitra Desai

Abstract A model when stated in simple terms is a mathematical equation, which is true when it implies to any model in machine learning including deep neural network. Every model will generate output for a given input, but important is to get output of desired accuracy. Machine learning models are trained on training data, and their best fit is judged on testing data. Before fitting the training data to the model to predict on unknown (test) data, pre-processing of data is essential to ensure model accuracy to acceptable level. This paper presents steps involved in pre-processing raw labelled dataset (Seattle weather) with 25,551 records (from year 1948 to 2017) to make it suitable for input to a deep neural network model. The data is split into 80% of training data and 20% of testing data. Scaling is performed on the data before it is passed to the deep neural network model. Deep neural network model that is multilayer perceptron model using sequential model API with dense layer is built and compiled using Adam optimizer resulting accuracy of 97.33% in predicting rainfall on a particular day.

Keywords Data pre-processing · Deep neural network · Adam optimizer · Sequential model

1 Introduction

Artificial neural network (ANN) models have been used for rainfall prediction [1, 2] and found suitable for handling complex large dataset, particularly of nonlinear nature. There are several methods [3] apart from artificial neural networks which have been used for forecasting rainfall; however, ANN has proved to be useful in identifying complex nonlinear relationship between input–output variables. As the number of hidden layers are increased for better performance [4], the concept of deep learning comes in, which are useful for rainfall prediction. The deep neural network models have mathematics behind them, the understanding of which enables one for the selection of architecture for fine tuning of deep learning models, setting of values

C. Desai (✉)
National Defence Academy, Pune, India

H. Sharma et al. (eds.), *Communication and Intelligent Systems*, Lecture Notes in Networks and Systems 204, https://doi.org/10.1007/978-981-16-1089-9_9

for hyperparameters and applying appropriate optimization. However, the success of a model for prediction or classification is directly impacted by the data used for training the model. The real-world data in its raw form may not be suitable to train the model, which signifies the importance of data pre-processing.

Pre-processing of data refers to improving the quality of data, which involves data cleaning, data reduction, data transformation and data encoding. In data cleaning, the missing values, duplicate values and outliers are dealt with. Data reduction refers to number of features being reduced, particularly adopted to reduce the effect of curse of dimensionality. Data transformation is applied to scale the data either using normalization or standardization. Data encoding ensures the categorical features in text format are encoded to numbers.

This paper presents steps involved in pre-processing raw labelled dataset (Seattle weather) with 25,551 records, to make it suitable for input to a deep neural network model. Insight into the data is further gained to identify the architecture suitable for chosen problem. Deep learning model, that is multilayer perceptron model using sequential model API with dense layer, is built and compiled using Adam optimizer for desired accuracy.

2 Methodology

Data obtained in raw form is suitably pre-processed to ensure all feature variables and target variable fit the DNN model. Using quantitative approach, the relationship between variables is identified. On pre-processing, the number of input neurons is identified. After pre-processing, data is divided into train–test data and scaling is applied. The DNN-sequential approach is adopted, and dense layers are added. At hidden layers, the activation function is chosen. As the problem belongs to the class of binary classification, sigmoid function is applied at the output layer. The epoch size is identified by observing the model loss and model accuracy curve. The test accuracy is further computed using the prediction capability of the model trained.

3 Data and Data Pre-processing

3.1 Data Set

The present study is made on Seattle, US weather dataset [5]. The dataset contains records of daily rainfall patterns from Jan 1st, 1948 to Dec 12th, 2017. The dataset consists of five columns, and the description of these columns is as follows as shown in Table 1.

There are total number of records which are 25,551. The memory usage is approximately 998 KB. Here DATE, PRCP, TMAX and TMIN are features (X),

Table 1 Data description for Seattle weather dataset

Column name	Data description
DATE	The date of the observation
PRCP	The amount of precipitation, in inches
TMAX	The maximum temperature for that day, in degrees Fahrenheit
TMIN	The minimum temperature for that day, in degrees Fahrenheit
RAIN	TRUE if rain was observed on that day, FALSE if it was not. [target]

Table 2 First five records of Seattle weather dataset

Date	PRCP	TMAX	TMIN	RAIN
1948-01-01	0.47	51	42	True
1948-01-02	0.59	45	36	True
1948-01-03	0.42	45	35	True
1948-01-04	0.31	45	34	True

Table 3 Statistical description of Seattle weather dataset

	PRCP	TMAX	TMIN
Count	25,548.000000	25,551.000000	25,551.000000
Mean	0.106222	59.544206	44.514226
Std.	0.239031	12.772984	8.892836
Min.	0.000000	4.000000	0.000000
25%	0.000000	50.000000	38.000000
50%	0.000000	58.000000	45.000000
75%	0.100000	69.000000	52.000000
Max.	5.020000	103.000000	71.000000

and RAIN is our target variable *Y*. RAIN is categorical in nature which has two possible output—True (RAIN) or False (Not RAIN). Thus, the problem belongs to the class—binary classification. Features—PRCP, TMAX and TMIN—are numerical continuous values, and DATE is in format YYYY-MM-DD. Sample five records are displayed below from the dataset as shown in Tables 2 and 3 which shows the statistical description of the dataset:

3.2 Data Pre-processing

Deep neural network (DNN) models like any other machine learning model requires pre-processing of data, before the data is passed to input neuron. One of the important

step is to identify the missing values and if found treat them appropriately. Missing values can be dropped and can be substituted by either mean, median or any other relevant value like 0 or 1. It is also necessary to check for duplicate data to make our model more impactful. Often, the raw dataset may contain columns which are less important and can be ignored or avoided as input to the model which are identified during pre-processing. Also, certain new columns may be required to be generated from the existing one for extracting more feature value for the model. Deep learning model takes input to the input layer neurons in the form of real values, essentially it is to be ensured that text to number encoding is done prior to that.

It is observed that the dataset consists of three null values in PRCP and RAIN columns as shown in Table 4. Before we send our data to the model, it is required that the null values to be treated with appropriate action. In this case compared to the huge number of records dropping, the three records with null values in PRCP and RAIN are recommended, and accordingly, they are removed from the dataset leaving 25,548 records for further processing. Next, the dataset is checked for duplicate values and it is observed that there are no duplicate records in the dataset.

The DATE field is broken into 'YEAR', 'MON', 'DAY'. The value true is replaced by 1 and False by 0 in the field 'RAIN', that is the text (Boolean) data is converted to numeric. Thus, feature X will consist of 'PRCP', 'TMAX', 'TMIN', 'YEAR', 'MON', 'DAY' and 'RAIN' as target column Y. So now the data appears as shown in Table 5.

Table 4 NaN values in dataset

Date	PRCP	TMAX	TMIN
1998-06-02	NaN	72	52
1998-06-03	NaN	66	51
2005-09-05	NaN	70	52

Table 5 Sample data after splitting DATE column

PRCP	TMAX	TMIN	RAIN	YEAR	MON	DAY
0.47	51	42	1	1948	1	1
0.59	45	36	1	1948	1	2
0.42	45	35	1	1948	1	3
0.31	45	34	1	1948	1	4
0.17	45	32	1	1948	1	5

3.3 Data Insight

The dataset contains data from the year 1948 to 2017, and the month-wise distribution of data from the 1948 to 2017 is shown in Table 6.

The rainfall experience in a particular month for the period 1948 to 2017 is shown in Table 7. It is observed that lowest rainfall is observed in the month of July every year, and highest rainfall is observed in the month of December every year.

The histogram for the column rainfall is shown in Fig. 1. The number of records with no rainfall is 14,648, and number of records with rainfall is 10,900.

To get the scatter plot of precipitation with the temperature, additional column AVGTEMP is created from TMAX and TMIN. The obtained scatter plot is shown in Fig. 2. On referring to Table 3, it is observed that the maximum value of PRCP is 5.02, which as shown in Fig. 2 is an outlier. Except for few cases where the average temperature is in between 48 and 60°F, we observe PRCP value above 2.0.

The precipitation is the water released from clouds in the form of rainfall [6]. Refer to Table 8 [7] below and observe Fig. 3, it is seen that October to April every year, moderate rainfall is experienced and from May to September light rainfall is experienced in Seattle.

Table 6 Month-wise total records since 1948–2017

Month	1	2	3	4	5	6	7	8	9	10	11	12
Number of records	2170	1978	2170	2100	2170	2098	2170	2170	2099	2170	2100	2153

Table 7 Month-wise rainfall records since 1948–2017

Month	1	2	3	4	5	6	7	8	9	10	11	12
Number of records	1298	1103	1212	998	771	632	343	413	609	950	1264	1307

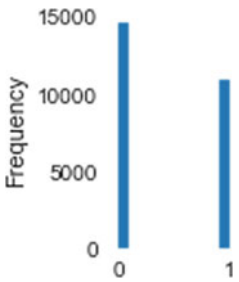

Fig. 1 Histogram for column RAIN

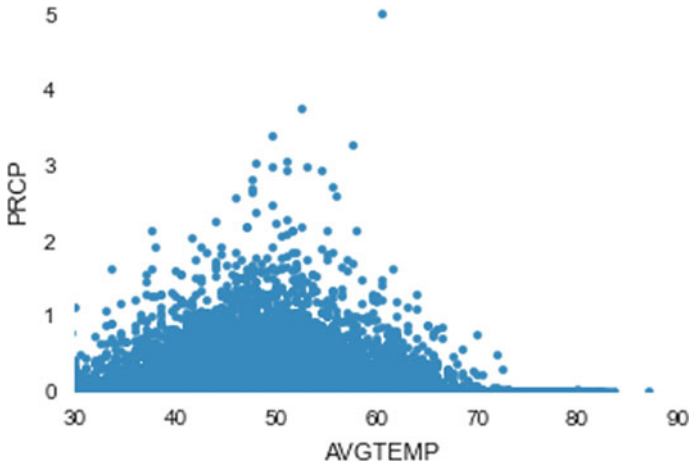

Fig. 2 Scatter plot

Table 8 Precipitation intensity

	Intensity in./h (cm/hour)	Median diameter (mm)	Velocity of fall ft/s (m/s)	Drops per second per square foot (m^2)
Fog	0.005 (0.013)	0.01	0.01 (0.003)	6,264,000 (67,425,000)
Mist	0.002 (0.005)	0.1	0.7 (0.21)	2510 (27,000)
Drizzle	0.01 (0.025)	0.96	13.5 (4.1)	14 (151)
Light rain	0.04 (0.10)	1.24	15.7 (4.8)	26 (280)
Moderate rain	0.15 (0.38)	1.60	18.7 (5.7)	46 (495)
Heavy rain	0.60 (1.52)	2.05	22.0 (6.7)	46 (495)
Excessive rain	1.60 (4.06)	2.40	24.0 (7.3)	76 (818)
Cloudburst	4.00 (10.2)	2.85	25.9 (7.9)	113 (1220)

Source Refer [6, 7]

3.4 Train–Test Split

To estimate the performance of machine learning algorithms, the data is split into training and testing data. Usually, the data is split into train–validate–test data. The train data is used to train the model; using validation data, the model is validated and tested using test data. In the present experiment, the data is split into only train and test data. The ratio chosen here is 80:20, that is 80% of training data and 20% of testing data.

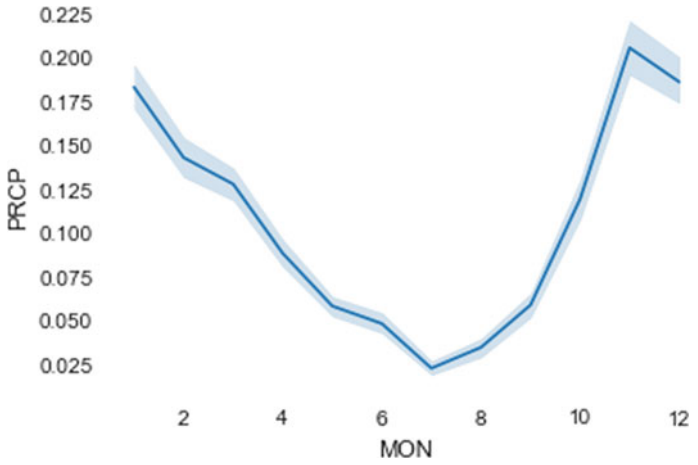

Fig. 3 Precipitation trend

3.5 Feature Scaling

Machine learning algorithms like linear regression, logistic regression and neural network that use gradient descent as an optimization technique require data to be scaled [8]. On observing the values across different columns in the Seattle weather dataset, we find there is varying range of values. By applying scaling, all values are brought on same scale before giving it to our deep neural network model. Scaling can be done using either normalization or standardization. In normalization, values are scaled in the range from 0 to 1, while in standardization values are centred around the mean. As we see outliers with respect to PRCP and AVGTEMP in the data, standardization on train and test data is applied.

4 Deep Neural Network Model

Neural network models in simple terms described as mathematical function which maps the input to generate the desired output. It comprises of input layer, output layer, arbitrary number of hidden layers, a set of weights, and bias between each layer and a choice of activation function and loss function.

On completion of transformation of the data, DNN-sequential model is applied here on the pre-processed data as it allows layer by layer model building, which forms network of dense layer. As there are six input features (except 'RAIN' which is target variable) as shown in Table 5, the first dense input layer is set with six features. Activation function rectified linear activation function 'ReLU' is used at hidden layers. Another dense layer with four neurons is added with 'ReLU' activation. As the problem belongs to the class of binary classification, 'sigmoid' function is used

Table 9 Model summary

Layer (type)	Output shape	Param #
dense_1 (Dense)	(None, 6)	42
dense_2 (Dense)	(None, 4)	28
dense_3 (Dense)	(None, 1)	5
Total Params: 75 Trainable params: 75 Non-trainable params: 0		

at the output layer, which will generate output either 1 or 0, for rainfall or no rainfall, respectively. Thus, the model has 6 input, two hidden layers with 6 and 4 neurons and output layer with one output. The model is implemented using keras.io [9]. The model summary is shown in Table 9.

To train deep neural network models, adaptive optimization algorithms are used. The examples include Adam [10], Adagrad [11], RMSprop [12]. Adaptive here refers that it computes individual learning rate for different parameters. The model is compiled using Adam optimizer which is seen as a combination of RMSprop and stochastic gradient descent [13] with momentum with few distinctions. The nature of problem being binary classification and as the target variables are {0,1}, the loss function is computed using cross-entropy [14]. The model is fitted to training data using 10 epochs and batch size of 64. One epoch is the complete pass through the training data. Epoch is a hyperparameter, and there is no thumb rule for that. Batch size is the number of sample processed in the single mini-batch.

5 Result and Conclusion

In this section, the results generated at various stages are discussed.

5.1 Weights and Bias

The model weights for first layer dense_1, with the output shape (None, 6) generates 42 parameters, that is 36 weights and 6 bias, similarly 24 weights and 4 bias for dense_2 and 4 weights and 1 bias for dense_3, total 75 parameters. The values generated across each weight and bias is as shown in Table 10.

Table 10 Weight and bias at hidden layers

Layer (type)	Weights	Bias
dense_1 (Dense)	[1.141973, 1.97442, −1.734192, 1.275294, −2.042584, −2.0059183], [−0.29253215, 0.12715897, 0.83936656, −0.2561121, 0.21438691, −0.00820558], [0.3164518, −0.1136589, −0.19223101, 0.4917892, −0.0732223, −0.07245291], [−0.12871715, 0.03039517, −0.18574753, −0.15315911, −0.12539569, 0.00333604], [−0.14801866, 0.07896663, 0.93043214, 0.5128609, −0.35926303, 0.29292443], [0.16915724, 0.00389614, −0.38254952, −0.28463966, 0.22222997, 0.00662909]	[1.0052627, 1.0992466, 0.28027502, 0.6150749, 0.31144813, −0.14159845]
dense_2 (Dense)	[−0.51795757, −0.85841197, −0.616253, 1.39788], [−0.87039506, −1.31375, −0.62691236, 2.1033716], [0.5722295, 0.95757383, −0.6314308, 0.10791895], [−0.78557706, −0.5372498, 0.2611877, 0.39101806], [−0.44372734, 1.4242774, 0.0454119, −0.62419903], [−0.6753952, 0.69929206, −0.34458962, −0.95543414]	[−0.19212776, 0.28099677, 0.08030465, 0.6100016]
dense_3 (Dense)	[0.5681466], [−1.3521026], [−0.12679887], [2.3315494]	[−0.08006777]

5.2 Training and Validation: Loss and Accuracy

After the model is complied, the training data is fit using 10 epoch and 64 batch size. As the number of epochs increases, more information is learned. Both training and validation accuracy increase as the number of epochs increases. The resultant training and validation loss and accuracy in each epoch are shown in Table 11. The model took 2.48 s to train.

Figure 4 shows at each increasing epoch the model loss and decreases and the model accuracy increases. From 8th epoch onwards, the curve starts flattening, and by 10th epoch, it becomes stagnant. Here, the batch size is 64. Thus, further training is stopped and model used for testing data.

Table 11 Training and validation loss and accuracy

Epoch	Training loss	Training accuracy	Validation loss	Validation accuracy
1	0.5548	0.7310	0.4685	0.7852
2	0.4041	0.8349	0.3324	0.8774
3	0.2866	0.8939	0.2363	0.9132
4	0.2210	0.9145	0.1939	0.9271
5	0.1839	0.9308	0.1654	0.9376
6	0.1547	0.9442	0.1415	0.9503
7	0.1326	0.9527	0.1224	0.9557
8	0.1143	0.9582	0.1067	0.9606
9	0.0990	0.9642	0.0938	0.9655
10	0.0859	0.9670	0.0819	0.9667

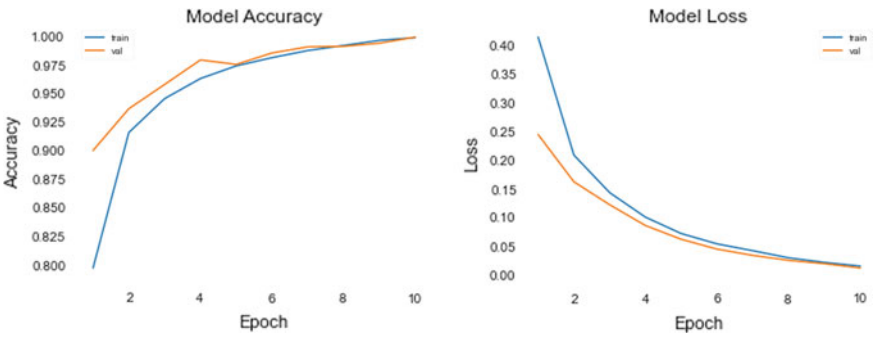

Fig. 4 Model accuracy and model loss

Table 12 Test loss and test accuracy

Test loss	0.07455659657716751
Test accuracy	0.9733855128288269

5.3 Test Loss and Test Accuracy

The model is applied on testing data, and the resultant test loss and test accuracy is shown in Table 12.

5.4 Comparative Analysis

The target variable in the present problem statement is binary, that is 1 for it will rain and 0 for will not rain. Thus, it is a classification problem and several machine learning classification algorithms can be fitted to this dataset on appropriate pre-processing. Logistic regression model [15] is one such model which best fits for binary classification. The training data is fitted to logistic regression model, and it is observed that test accuracy obtained is 0.9330724070450098.

6 Conclusion

The paper presents the steps involved in pre-processing the raw data before passing it to a deep neural network model. The architecture of the deep neural network model is influenced by the feature vectors that go as input and the target variable which is the expected output. The activation function and optimizer used impacts the loss function. Here, the Seattle weather data is used for rainfall prediction which is available for a period from 1948 to 2017. The prediction of rainfall on a particular day which belongs to the class of binary classification is trained using deep neural network model. The sequential model with dense layer, ReLU activation function at hidden layer, sigmoid function at output layer, Adam optimizer, 10 epochs with batch size 64 is implemented on the dataset to achieve the test accuracy of 97.33%. The training data when was fitted to logistic regression model also, and it gave accuracy of 93.30%. Thus, it is recommended to use DNN model, as in logistic regression the classification is linear, whereas the DNN model will be useful for more complex and nonlinear data.

References

1. Nayak DR, Mahapatra A, Mishra P (2013) A survey on rainfall prediction using artificial neural network. Int J Comput Appl
2. Lee S, Cho S, Wong PM (1998) Rainfall prediction using artificial neural networks. J Geogr Inf Decis Anal 2(2):233–242
3. Lee J, Kim C-G, Lee JE, Kim NW, Kim H (2018) Application of artificial neural networks to rainfall forecasting in the Geum River Basin, Korea. Water 10:1448. https://doi.org/10.3390/w10101448
4. Aswin S, Srikanth G, Vinayakumar R (2018) Deep learning models for the prediction of rainfall. In: Conference: 2018 international conference on communication and signal processing (ICCSP). https://doi.org/10.1109/ICCSP.2018.8523829
5. https://www.kaggle.com/rtatman/did-it-rain-in-seattle-19482017
6. Rain: A Water Resource. USGS General Interest Publication. https://www.usgs.gov/special-topic/water-science-school/science/precipitation-and-water-cycle?qt-science_center_objects=0#qt-science_center_objects. Last accessed 12/101/2020
7. Lull HW (1959) Soil compaction on forest and range lands. U.S. Dept. of Agriculture, Forestry Service, Misc. Publication No. 768
8. Bhandari A (2020) Feature scaling for machine learning: understanding the difference between normalization versus standardization, https://www.analyicsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/
9. Keras Homepage. https://keras.io/. Last accessed 14/10/2020
10. Kingma DP, Ba JL (2014) Adam: a method for stochastic optimization. arXiv:1412.6980v9
11. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 12:2121–2159
12. Tieleman T, Hinton G (2012) Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. COURSERA: neural networks for machine learning 4(2):26–31
13. Ashia C, Wilson RR, Stern M, Srebro N, Recht B (2017) The marginal value of adaptive gradient methods in machine learning. arXiv:1705.08292v2
14. Mannor S, Peleg D, Reuven R (2005) The cross entropy method for classification, ICML. In: '05: proceedings of the 22nd international conference on machine learning, pp 561–568. https://doi.org/10.1145/1102351.1102422
15. Jakaitiene A (2019) Nonlinear regression models. In: Encyclopedia of bioinformatics and computational biology