An Approximation to Deep Learning Touristic-Related Time Series Forecasting

Daniel Trujillo Viedma(✉), Antonio Jesús Rivera Rivas, Francisco Charte Ojeda, and María José del Jesus Díaz

Andalusian Research Institute on Data Science and Computational Intelligence (DaSCI), Computer Science Department, University of Jaén, 23071 Jaén, Spain
dtviedma@ujaen.es
http://wwwdi.ujaen.es/

Abstract. Tourism is one of the biggest economic activities around the world. This means that an adequate planning of existing resources becomes crucial. Precise demand-related forecasting greatly improves this planning. *Deep Learning* models are showing an greatly improvement on time-series forecasting, particularly the LSTM, which is designed for this kind of tasks. This article introduces the touristic time-series forecasting using LSTM, and compares its accuracy against well known models RandomForest and ARIMA.

Our results shows that new LSTM models achieve the best accuracy.

Keywords: LSTM · ARIMA · Time series forecasting

1 Introduction

World tourism is considered one of the most important activities from an economic point of view, only exceeded by oil and derivative products [1]. In the case of Spain, 12.1% of the total employments in year 2013 were linked to touristic services, starting from a 9.8% in 2001 [3]. In 2016, it represents 13% of total employment and 11,2% of Spanish GDP, approximately 125.529 millions of euros [4]. Such a big economic activity requires an accurate demand forecasting, in order to adapt the resources (beds, services, etc.) to successfully attend a highly variable demand.

By its nature, time series are important to this kind of economic activity, given the high number of relevant variables around it that are modeled in a time-dependent way such as the number of customers, available beds, or average customer spending, among others. These variables can only be analyzed if their observations are taken into account in a time-ordered fashion, and this let's us to make predictions based on the same order. They are, therefore, time series.

Typically, times series have been modeled using the ARIMA (*AutoRegressive Integrated Moving Average*) [8] technique, which integrates a moving averages analysis with an autoregressive one, known as ARMA model, and extend it to work with differentiated time series. ARIMA, together with heuristic analysis in order to find the best values for its parameters, has a strong statistical foundation and is considered state of the art in time series forecasting.

On the other hand, new deep learning models have greatly improved previous machine learning algorithms, thanks to better available training methods and faster computers on which run those methods. These models have improved tasks such as handwriting recognition [9] or statistical machine translation [2].

One of these models, LSTM (*Long Short-Term Memory*) [6], is specially designed for time series forecasting. Its structure allows it to store relevant information to be consumed on the next input, providing an useful extended context when analyzing an input time step, greatly improving the accuracy.

The main purpose of this study is to introduce touristic time-series forecasting with LSTM, while comparing its accuracy against the well known models RandomForest and ARIMA.

This article is structured as follows: The next section, the second one, briefly describes the methods involved in this study. The third section details the experiments made to test al the models. Finally, the results of those experiments are shown and discussed with a brief, final conclusion.

2 Methods

This section gives an introductory description of the methods involved in this study, LSTM, RandomForest and ARIMA. As providing a complete description of these methods is not a goal of this article, most important references are cited.

2.1 LSTM

LSTM is an artificial recurrent neuron architecture. Its structure allows it to maintain an internal state, made from past observations of the neuron, that drives future outputs. This internal state constitutes, in fact, a memorized knowledge, stored in a component called CEC (*Constant Error Carrousel*).

LSTM is designed to solve the exploding and vanishing gradient problems which are extensively discussed in [7], being vanishing gradient the most frequently found in neural networks. This problem references the decreasing effect of a network parameter (like an actual weight) over the network output, as far as the number of layers between the parameter and the output increases, affecting very deep feedforward networks and recurrent networks with big time lags. This model boosted recurrent networks research, by providing practical advantages over traditional feedforward networks due to solving the vanishing gradient problem [7].

In it's original formulation, [10], this processing block, called *cell*, defines a data pipeline through which input values are transformed by means of multiplicative operations and activation functions. Additions to this structure were later added, setting the de facto standard configuration of the cell. These additions can be summed up in the following:

– *Forget gate*: Was introduced by [5]. Previously, the output of the CEC was simply copied to its input, and the input of the CEC was transformed by another activation function. With the addition of the product operation, this activation function no longer makes sense.
– *Peephole connections*: Bypass values from the CEC's output to the (input, forget and output) gates.

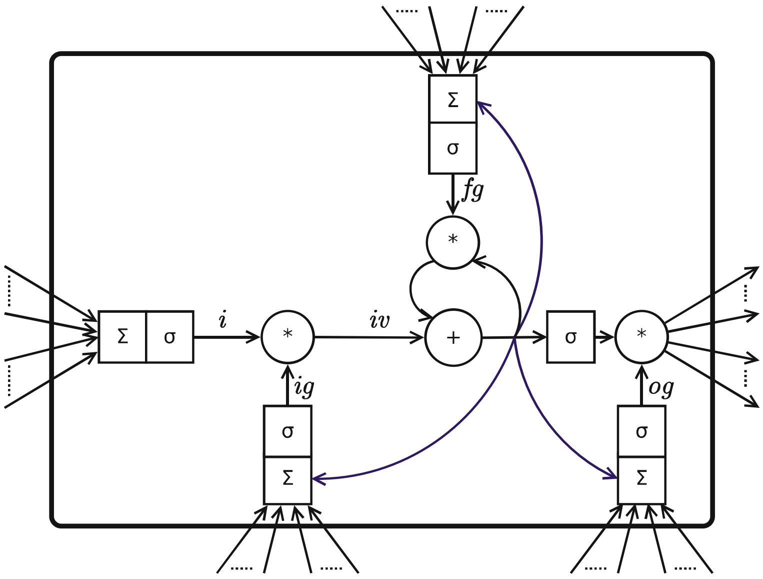


Fig. 1. Schematic of an LSTM cell.

Figure 1 details the LSTM cell components. The data pipeline can be quickly seen, starting from the left side, where the input values are provided to the cell, until the right side, where the output values are returned from the cell; while the data is transformed by each gated unit. Also, the CEC can be appreciated.

This data pipeline can be described as follows:

1. Input values are provided to the *input*, *input gate* y *forget gate*, computing values *i*, *ig* y *fg*, respectively.
2. *i* and *ig* are combined by a pointwise product operation, obtaining *iv*.
3. The pointwise product between *fg* and the value from the last CEC evaluation is computed. This value will feed the CEC the next time step, added up in a pointwise fashion with *iv*. If there are peephole connections present, this CEC's output value is copied to the destination of these connections.
4. The CEC's output value feed an activation function, which is multiplied, also in a pointwise fashion, with the *output gate* value, named *og*, which is computed given the recurrent input to the cell and the previously computed CEC's output value.

This internal structure was designed to work with long input time dependencies. Along with an adequate training algorithm, such as backpropagation, a constant error flow through the CEC is assured, while the *input gate* protects CEC from irrelevant inputs, while the *output gate* protects another cells from irrelevant CEC values.

This *modern* LSTM is described in detail in [6].

The most commonly used training method used with this kind of neural networks is BPTT (*Backpropagation Through Time*), which is a generalization of the well known Backpropagation algorithm, adapted to work with recurrent networks. It starts by unrolling the LSTM layer so that it becomes a feedforward network, given a recursion limit. Then, applies a standard Backpropagation to obtain changes in weights. Lastly, aggregates (typically, with an statistical mean) the changes corresponding to the same recursive weight. This algorithm is detailed in [11].

2.2 RandomForest

RandomForest is just an ensemble of decision trees in which every tree is specialized on predicting over a number of randomly chosen variables from the whole dataset. A new instance is evaluated by retrieving the output of each tree separately and then deciding the final output of the model. Usually, the most predicted class, or the mean of predicted values is given as a final output.

The number of variables each tree sees is set as a parameter, and then the optimal number of trees is derived by means of minimal out-of-bag error.

In this scenario, a RandomForest model can be trained to predict the next value of a sequence given the past ones, and also including another variables to improve accuracy.

2.3 ARIMA

ARIMA [8] is a widely used method for time series forecasting. It works by finding a function that approximates the input time series and that can be evaluated for further time steps. That means that ARIMA is powerful for predictive tasks, but also useful in descriptive tasks.

ARIMA builds on top of the ARMA model, extending it to work with differentiated time series (in essence, a data sequence composed by the differences between an element and the following one from the original sequence). ARMA stands for *AutoRegressive and Moving Averages* and consists on finding an autoregressive adjust over the differences of the input values, and then model the error with a moving averages model.

3 Experiments

In order to establish a comparison between the distinct models considered, a real world touristic dataset has been used to train several models of each kind (LSTM, RandomForest and ARIMA), validating them by means of two error metrics.

From all the models trained, a selection of the best of each kind was made by training accuracy criteria. Later, these models were validated with new data, which doesn't got involved in training at all. This way, we can simulate a realistic scenario where the future values are not know at the present, but will be known later.

3.1 Error Metrics

We have considered 2 distinct error functions: RMSE (*Root Mean Squared Error*) and MAPE (*Mean Absolute Percentage Error*).

These error metrics quantifies the deviation of the predictions made with the models from the real data.

Given E the expected values and P the predicted values (both of them, with the same number of observations), these metrics are defined as follows:

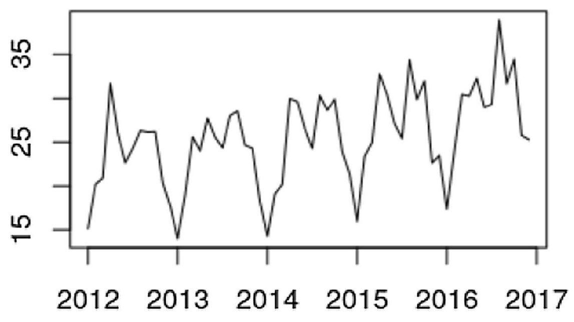
$$RMSE(E,P) = \sqrt{MSE(E,P)} \tag{1}$$

$$MSE(E,P) = \frac{\sum_{i=1}^{n}(E_i - P_i)^2}{n} \tag{2}$$

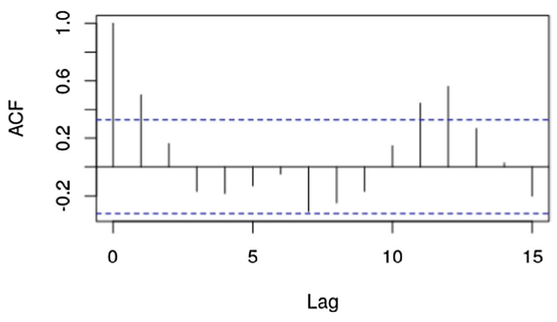
$$MAPE(E,P) = \frac{100}{n}\sum_{i=1}^{n}\frac{E_i - P_i}{E_i} \tag{3}$$

3.2 Dataset

The dataset used in this study was retrieved from the Spain's Instituto Nacional de Estadística. More precisely, the touristic occupation survey, occupation grade per place series in the province of Jaén. The data has been plotted in Fig. 2a. Data is offered in a monthly resolution, and only values between January 2012 and December 2016 are considered.



(a) Visualization of the data set.



(b) Autocorrelation function.

Fig. 2. Raw data and ACF graphs.

At first appearance, some patterns can be easily recognised, as well as occupation peaks which corresponds to the summer holidays and the Easter holidays. As expected, there's a clear correlation between years.

From this dataset, values of year were extracted to be used only in the final validation, being the rest (January 2012 - December 2015) used for training.

In order to improve accuracy, additional variables whose values are known a priori have been merged into the dataset, namely:

- Year.
- Month of the year.
- Easter: Boolean stating if the instance belongs to a month where a Easter occurs.
- Sort weeks: Number of holidays in Tuesday or Thursday.
- Long weekends: Number of holidays in Monday or Friday.

The time lags included in LSTM and RandomForest instances were chosen following a autocorrelation function criteria. This function applied over the training input data is shown in Fig. 2b, in where the most relevant time lags can be seen easily.

Though this is the general structure of the dataset, some model's training algorithms implementations forces us to adapt this structure:

LSTM and RandomForest: Each instance has the following form: $(V_n, V_{n-1}, V_{n-11}, V_{n-12}, HW, SW, LW)$.
ARIMA w/xregs: Each instance has the following form: $(V_n, V_{n-12}, HW, SW, LW)$.

Being:

n The month the instance data refers to.
V_i The value of the time series at month i.
HW If the month contains an Easter (or part of it).
SW Number of short weeks within the month.
LW Number of long weekends within the month.

3.3 Methods

Experiments with LSTM neural networks were made with a *Python* script, using *Keras* library with *TensorFlow*. Also, *Pandas* library was used for data structures, *Numpy* for numerical computing, *Matplotlib* for data visualization and *Scikit-learn* for error metric computing and data normalization.

The parameter settings taken into account have been: 5 and 10 LSTM cells per network; 800 and 1000 iterations; and 1, 2, and 4 batch sizes.

Given the probabilistic nature of the training algorithm (Backpropagation Through Time), which randomly initializes the networks weights, 20 repetitions of each parameter set have been made, in order to reduce the impact of the initial weights on the computed errors.

The ARIMA and RandomForest experimentation was made in R, justified by the quality of the implementations of these two kind of models in *caret*, *randomForest* and *forecast* packages. More precisely, we have used *auto.arima* function to train the ARIMA model, and caret's *train* function, with *method* parameter set to *"rf"*, and *mtry* parameter varying from 1 to 6. In the ARIMA case, we trained 2 kind of models: with and without external regressors, given the important accuracy disparity both models exhibit.

3.4 Results

Table 1 shows the error made by the best accurated LSTM models trained, in ascending error order. For each one of the parameter configuration, mean and standard deviation of RMSE from 20 repetitions are listed. The first (less erratic) model is selected to compete with ARIMA and RandomForest. In a similar way, Table 2 shows the result for RandomForest experiments. Also, like in the LSTM case, 20 repetitions were made for each specific parameter setting.

Table 1. LSTM - Training dataset - RMSE

Num LSTM	Epochs	batch size	RMSE	
			Average	Std. Dev.
5	1000	4	2.1291	0.0260
5	1000	1	2.1323	0.0283
5	800	1	2.1331	0.0332
5	800	2	2.1361	0.0334
10	800	1	2.1389	0.0314
5	800	4	2.1389	0.0320
10	800	4	2.1419	0.0216
5	1000	2	2.1446	0.0331
10	800	2	2.1455	0.0292
10	1000	4	2.1481	0.0271
10	1000	1	2.1496	0.0501
10	1000	2	2.1698	0.0195

Table 2. RandomForest - Training dataset - RMSE

mtry	RMSE	
	Average	Std. Dev.
6	2.4680	0.0636
5	2.5092	0.0463
4	2.5300	0.0608
3	2.5370	0.0352
2	2.6612	0.0898
1	3.4334	0.0639

Table 3 shows the result of ARIMA experiment. Given the fact that ARIMA is not probabilistic, it doesn't make sense to run it several times, since all of it will have exactly the same error. It's easy to see that, when external regressors are added into the ARIMA model training, it greatly improves the accuracy of the resulting model.

Table 3. ARIMA

xregs	Training	Testing	
	RMSE	RMSE	MAPE
No	2.4056	1.8901	5.0111
Yes	3.6616	6.7481	18.7578

Table 4. Testing dataset accuracy

	RMSE	MAPE
LSTM	1.7216	4.8653
ARIMA w/xregs	1.8901	5.0111
ARIMA	6.7481	18.7578
RandomForest	5.7435	16.2852

Lastly, Table 4 shows the accuracy, this time on the testing dataset, of the best parameter setting for each kind of model (LSTM, ARIMA and Random-Forest). This data provides evidence that support our thesis, that LSTM models have best accuracy on testing data than classical (non deep learning), state of the art, ones.

4 Conclusion

In this work, we established a comparison between 3 different kind of models, including a novel, deep learning one, and another coming from Statistics field, considered the state of the art in time series forecasting, to improve a real world touristic occupation forecasting, which justifies by the economic importance of tourism activities around the world.

The results of this comparison, which can be seen in Table 4, shows that the deep learning model, LSTM, can achieve best accuracy than the other models taken into account.

Acknowledgements. This work is partially supported by the Spanish Ministry of Science and Technology under project TIN2015-68454-R.

References

1. Altés, C.: Marketing y turismo. Editorial Síntesis, Madrid (1993)
2. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
3. Cuadrado Roura, J.R., López Morales, J.M., et al.: El turismo, motor del crecimiento y de la recuperación de la economía española (2015)
4. de Estadística, I.N.: Aportación del turismo a la economía española (2016). http://www.ine.es/
5. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: continual prediction with LSTM (1999)
6. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. **18**(5–6), 602–610 (2005)
7. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al.: Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (2001)
8. Hyndman, R.J., Khandakar, Y., et al.: Automatic time series for forecasting: the forecast package for R. No. 6/07. Monash University, Department of Econometrics and Business Statistics (2007)
9. Romanjuk, V.V.: Training data expansion and boosting of convolutional neural networks for reducing the mnist dataset error rate. Naukovi Visti NTUU KPI **6**, 29–34 (2016)
10. Schmidhuber, J., Hochreiter, S.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
11. Werbos, P.J.: Generalization of backpropagation with application to a recurrent gas market model. Neural Netw. **1**(4), 339–356 (1988)