Predicting Seattle Temperature using a Custom LSTM

Pedro Garcia
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Richardson, TX
pag170030@utdallas.edu

Sharvesh Suba Palaniraj
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Richardson, TX
sxs210399@utdallas.edu

Anthony Salto
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Richardson, TX
axs220582@utdallas.edu

Daisy Nguyen
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Richardson, TX
dxn230002@utdallas.edu

Abstract. We have built our own Long Short Term Memory model with NumPy to predict the next day's maximum temperature given seven days of Seattle weather data. We have 1461 daily observations from 2012 to 2015. We split the examples according to their date, we then fitted the scalers to the training data. Our results showed that the LSTM achieved a smaller RMSE of 2.60 °C compared to linear regression (2.75 °C) and persistence (2.93 °C). These values were obtained from one run with a random seed of 42.

I. Introduction

Often the maximum temperature of a particular day will be similar to the temperatures seen recently. However, there can be sudden changes so a simple guess may not be reliable. We wanted to explore if we could use a week's worth of weather data to predict the next days maximum. Using persistence and linear regression gave us simple baselines.

We built the LSTM model using NumPy. This allowed us to understand how the model works by implementing the gates and backwards pass manually. We trained the model on Seattle weather data and calculated how much error was made on the last part of the series.

II. Background

A. Time Series Prediction

Using persistence means the last maximum temperature observed is used as the prediction. Linear regression allows us to assign weights to the measurements. They provide useful baselines as they are less complex than an LSTM and use the same test time frame.

B. Recurrent Neural Networks

Recurrent Neural Networks process sequential observations. This can cause issues with the gradient becoming very small or large during training. The LSTM uses a cell state and gates to manage the flow of information [1].

C. Long Short Term Memory

The input gate decides how much new information to add to the cell. The forget gate decides how much of the previous cell state should be forgotten. The output gate helps to create the hidden state. This means that the model can use information from more than one day.

III. Dataset

This dataset contains 1461 daily observations from January 1st 2012 to December 31st 2015 [2]. Each observation contains:

- Precipitation(mm)
- Max Temp (°C)
- Min Temp (°C)
- Wind Speed (km/h)
- Weather Conditions (Categorical - Rain, Sun, Drizzle, Fog, Snow)

A. Target Variable

The target is the maximum temperature of the day after.

B. LSTM Model

The model consists of one LSTM layer containing 32 hidden units. Each observation contains 7 days of precipitation, max/min temperatures, wind and indicator variables for the weather. The model processes the days sequentially and the final hidden state is passed through a linear output layer.

At each time step t , the LSTM maintains a hidden state $h_t \in R^H$ and a cell state $c_t \in R^H$. The input vector x_t joins the previous hidden state h_{t-1} before it passes through four gates.

Forget gate:

$$f_t = \sigma(W_f[h_{t-1}, x_t]^T + b_f)$$

Input gate:

$$i_t = \sigma(W_i[h_{t-1}, x_t]^T + b_i)$$

Candidate cell state:

$$\bar{c}_t = tanh(W_c[h_{t-1}, x_t]^T + b_c)$$

Output gate:

$$o_t = \sigma(W_o[h_{t-1}, x_t]^T + b_o)$$

The model then calculates:

$$c_t = f_t \odot c_{t-1} + i_t \odot \bar{c}_t$$
$$h_t = o_t \odot tanh(c_t)$$

After the model reads all seven days, it sends the final hidden state h_T through a fully connected layer:

$$\hat{y} = W_{out} h_T + b_{out}$$

This gives the final output which is the estimated maximum temperature for the next day.

C. Training Strategy

Each observation contains 7 days of data and the next days maximum temperature. We checked that we had this correct when we adjusted the original script. Originally the target was 2 days after the end of the 7 day window.

Loss Function:

$$L = \frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2$$

We used mean squared error as our loss function. The gate outputs, hidden and cell states are stored during the forward pass. Gradients are calculated using backpropagation through time.

Stabilisation Techniques

- We limited the large gradients.
- Initialised weights to small random numbers.
- Initialised the forget gate bias to 1.

Hyperparameters

- Hidden units: 32
- Sequence Length: 7 Days
- Learning Rate: 0.001
- Batch Size: 32
- Epochs: 100

Data Splitting

We had 1454 examples after creating the 7 day windows. We split the data into training (first 1017), validation (next 218) and test (last 219) examples. The data is split based on date to ensure we do not have future examples in our training data.

For each epoch, we shuffled the training examples, we formed batches, and we recorded the training and validation losses. We then plotted both losses after training which generally fell over the run, although they did fluctuate between epochs.

IV. Method

A. Data Preprocessing

Sorted the data according to its date. Checked for any missing values or gaps. Converted the weather conditions into indicator variables. Created 7 day windows with the next days max temp as the target. Split the windows according to their dates. Fitted separate Min Max scalers to the training data. Applied these to the validation and test. For the persistence baseline we took the last maximum temperature observed and applied the target scaling.

B. Baselines

The persistence baseline takes the last temperature from the 7 day window and uses this as the prediction.

$$\hat{y}_{t+1} = y_t$$

No adjustments to the other weather observations are made.

Linear regression uses all the observations in the 7 day window. Each window was flattened into a single vector. The model was trained on the training data and tested on the same examples as the LSTM.

C. Training Custom LSTM

The LSTM reads each seven day input, one day at a time. During the training phase, we calculated the gradients across the 7 steps, then we clipped them, and lastly updated the weights with Adam [3].

D. Evaluation

The model was evaluated using mean squared error on the scaled targets. The root mean squared error was also calculated in °C. The graphs below show the loss curves, predictions for the first 100 examples and RMSE comparison.

The learning rate was set to 0.001 and the random seed to 42. There was no early stopping implemented within the code. Both the training and validation losses had an overall downward trend with some fluctuation between the epochs. However the validation loss had some variation between epochs.

V. Results

The table below shows the errors for the test data using the corrected code. The MSE is calculated using the scaled temperatures. The RMSE converts the errors back to °C making it easier to compare the models.

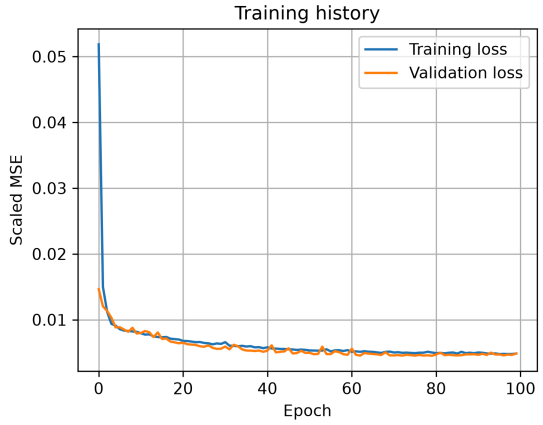
TABLE I. Test Set Performance

Model	Test MSE	Test RMSE (°C)
Persistence	0.0062	2.93
Linear Regression	0.0055	2.75
Custom LSTM	0.0049	2.60

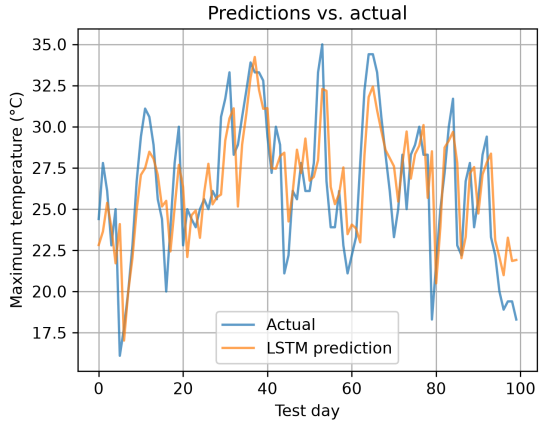
A. Analysis

The LSTM had the smallest test RMSE with 2.60 °C. The linear regression model had an RMSE of 2.75 °C, this is 0.15 °C higher than the LSTM's 2.60 °C. Compared to linear regression, the LSTM reduced RMSE by about 5.5%. The persistence model had an error of 2.93 °C. From the prediction graph we can see that the LSTM followed a lot of the pattern but did not capture some of the sudden changes. These values were obtained from one seeded run. The slight improvement over the linear regression model may not happen if the model was initialised differently or with different test data. The original code gave a RMSE of 3.28 °C with the same seed. This was predicting 2 days ahead and the scalers were fitted to the whole dataset. The corrected value is much smaller numerically (20.7% smaller). However, since we are forecasting one day ahead rather than two, these values cannot be compared directly.

B. Training

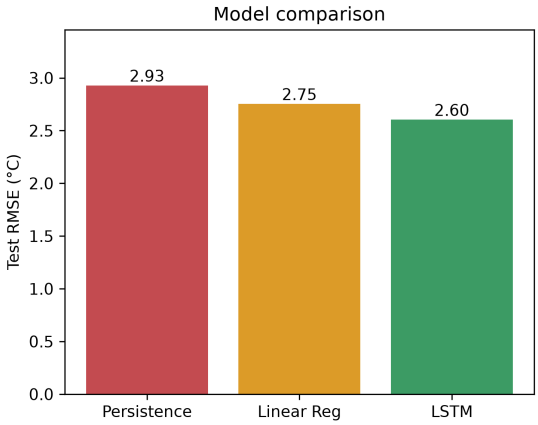


C. Prediction vs Actual



The line follows the general trend for the first 100 examples. However there are still a few peaks and troughs not hitting the target.

D. Model Comparison



VI. Conclusion

Overall the custom LSTM improved the RMSE by 0.15 °C compared to linear regression. This was a minor improvement. However, in the seeded run the model did better than both baselines. The corrected code now matches each 7 day window to the next day. The model is now learning the scaling values from the training data only. If the experiment was repeated with different seeds and later weather data we would see if the LSTM still had the slight advantage.

References

[1] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] Kaggle, "Weather Prediction Dataset." [Online]. Available: https://www.kaggle.com/datasets/ananthr1/weather-prediction.

[3] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization." arXiv:1412.6980, 2014