STAT 153 & 248 - Time Series
Lecture Twenty Six

Fall 2025, UC Berkeley

Aditya Guntuboyina

Dec 04, 2025

1 AR to LSTM

We have an observed time series $\{y_t\}$. We want to predict future values of this time series. We convert this time series data to regression data (x_t, y_t) via: $x_t = (y_{t-1}, \ldots, y_{t-p})$ for some $p \geq 1$. We will re-index so that x_t and y_t are both defined for $t = 1, \ldots, n$. The models described below will apply to this regression setup. Each model will give a predicted value μ_t for y_t; μ_t will be a function of x_t as well as past x-values $x_{t-1}, x_{t-2}, \ldots, x_1$. Our loss function will be the squared error loss $\sum_{t=1}^{n}(y_t - \mu_t)^2$. μ_t will depend on various parameters defining the model which will be estimated by least squares (in practice, this least squares problem will solved by some variant of the gradient descent algorithm).

1.1 AR(p)

The AR(p) model is given by:

$$\mu_t = \beta_0 + \beta^T x_t.$$

1.2 Nonlinear AR(p)

This model is given by:

$$\begin{aligned} h_t &= \sigma(W x_t + b) \\ \mu_t &= \beta_0 + \beta^T h_t. \end{aligned} \tag{1}$$

Here h_t is $k \times 1$ (so that W is $k \times p$ and b is $k \times 1$; note that x_t is $p \times 1$). This model is called the single hidden-layer neural network. h_t is called the Hidden State. In this model, h_t only depends on x_t (x_t will be called the "input" at time t or the "current input"). Note that μ_t is a linear function of hidden state (this will be true in the subsequent models as well).

1.3 RNN

The Recurrent Neural Network is:

$$\begin{aligned} &h_t = \tanh(W_h h_{t-1} + W x_t + b) \quad \text{with initialization} \quad h_0 = 0 \\ &\mu_t = \beta_0 + \beta^T h_t \end{aligned} \tag{2}$$

So, in an RNN, the hidden state at time t also depends on the previous hidden state (in addition to depending on the current input). Also, the activation function is now the tanh function: $\tanh(u) := \frac{e^u - e^{-u}}{e^u + e^{-u}}$.

As we saw in the last lecture, RNNs either suffer from explosive behavior (if at least one of the eigenvalues of W_h exceeds 1 in modulus) or have a lack of long memory (if all eigenvalues of W_h have moduli strictly smaller than 1).

1.4 GRU

The Gated Recurrent Unit (GRU) is given by:

$$\begin{aligned} h_0 &= 0 \\ \tilde{h}_t &= \tanh\left(W_h(h_{t-1} \odot g_t) + W x_t + b\right) \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \\ z_t &= \sigma_{\text{sigmoid}}(W_h^z h_{t-1} + W^z x_t + b^z) \\ g_t &= \sigma_{\text{sigmoid}}(W_h^g h_{t-1} + W^g x_t + b^g) \\ \mu_t &= \beta_0 + \beta^T h_t \end{aligned} \tag{3}$$

Here the sigmoid activation function is $\sigma_{\text{sigmoid}}(u) := e^u/(1 + e^u)$. The symbol $\odot$ represents elementwise multiplication. If all components of z_t equal 0, and all components of g_t equal 1, then the GRU reduces to the RNN. z_t is called the update gate, while g_t is called the reset gate.

If all components of z_t are close to 1, then $h_t \approx h_{t-1}$. In this way, it is possible for the previous hidden state to be propagated to the next time point without much degradation. This allows long memory (unlike the case of RNN where the hidden state decays by a 'factor' of W_h for each time step).

The last equation in (3) ($\mu_t = \beta_0 + \beta^T h_t$) is not usually considered part of the GRU. The remaining equations form the GRU and the output of the GRU are the hidden states $h_1, h_2, \ldots, h_T$. These outputs are then converted into μ_t by a linear transformation, and μ_t is compared to y_t in the loss function.

1.5 LSTM

In RNN and GRU, the hidden state vector h_t is playing a dual role:

1. It contains information from the inputs $x_t, x_{t-1}, \ldots, x_1$ that are relevant for the prediction of the current output y_t
2. It is also supposed to store information from $x_t, x_{t-1}, \ldots, x_1$ that are needed for accurately predicting future output values $y_{t+1}, \ldots$.

In the LSTM, these two different roles are played by two separate hidden states h_t and c_t. h_t is still called the hidden state, but c_t is called the cell state. h_t (which plays the first role) represents the short-term memory (which captures the "current context" or information relevant for prediction of the current output). On the other hand, c_t (which plays the second role) represents the long-term memory which holds information relevant for future predictions (sometimes for predictions made well into the future).

Because the model uses both h_t (short term memory) and c_t (long term memory), it is called the Long Short Term Memory (LSTM) neural network.

The equations underlying the LSTM are the following. These describe how (h_t, c_t) are calculated from (h_{t-1}, c_{t-1}) and the current input x_t.

$$\begin{aligned}
&h_0 = 0 \quad \text{and} \quad c_0 = 0 \\
&\tilde{c}_t = \tanh\left(W_{hc} h_{t-1} + W_{ic} x_t + b_c\right) \\
&c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
&h_t = o_t \odot \tanh(c_t) \\
&f_t = \sigma_{\text{sigmoid}}\left(W_{hf} h_{t-1} + W_{if} x_t + b_f\right) \\
&i_t = \sigma_{\text{sigmoid}}\left(W_{hi} h_{t-1} + W_{ii} x_t + b_i\right) \\
&o_t = \sigma_{\text{sigmoid}}\left(W_{ho} h_{t-1} + W_{io} x_t + b_o\right) \\
&\mu_t = \beta_0 + \beta^T h_t
\end{aligned} \tag{4}$$

As in the case of (3), the last equation in (4) ($\mu_t = \beta_0 + \beta^T h_t$) is not considered part of the LSTM. The remaining equations form the LSTM and the output of the LSTM are $(c_1, h_1), h_2, \ldots, (c_T, h_T)$. These outputs are then converted into μ_t by a linear transformation (note that μ_t only depends on h_t and not on c_t), and μ_t is compared to y_t in the loss function.

The LSTM has three gates: f_t (forget gate), i_t (input gate) and o_t (output gate). Here is some high-level intuition into the LSTM equations:

1. **Candidate Cell State** $\tilde{c}_t$: $\tilde{c}_t = \tanh\left(W_{hc} h_{t-1} + W_{ic} x_t + b_c\right)$. This is the proposed new content that could be added to the long-term memory. It plays the same conceptual role as $\tilde{h}_t$ in the GRU

2. **Forget Gate** f_t: $f_t = \sigma_{\text{sigmoid}}\left(W_{hf} h_{t-1} + W_{if} x_t + b_f\right)$. The forget gate decides how much of the previous long-term memory c_{t-1} should be retained. If (each component of) $f_t \approx 1$, then we keep most of c_{t-1} but if $f_t \approx 0$, then most of c_{t-1} is discarded. This ability to selectively preserve information is what enables LSTMs to maintain memory over hundreds of time steps solving the lack of long memory problem of RNNs.

3. **Input Gate** i_t: $i_t = \sigma_{\text{sigmoid}}\left(W_{hi} h_{t-1} + W_{ii} x_t + b_i\right)$. This gate decides how much of the proposed new content $\tilde{c}_t$ actually enters the long-term memory. If (each component of) $i_t \approx 1$, then most of the new information is accepted into the long-term memory, and if $i_t \approx 0$ then the new information is blocked almost entirely. Together, f_t and i_t determine how the long-term memory evolves.

4. **Output Gate** o_t: $o_t = \sigma_{\text{sigmoid}}\left(W_{ho} h_{t-1} + W_{io} x_t + b_o\right)$. This gate decides how much of the cell state becomes visible through the short-term memory h_t: $h_t = o_t \odot \tanh(c_t)$. Even if c_t stores a great deal of information, the model can choose to reveal only the portion that is immediately relevant for predicting the current output.

Note that the gate equations (as well as the equation for $\tilde{c}_t$) only involve h_{t-1} and x_t (and not c_{t-1}).

We will use the LSTM with $p = 1$ and $x_t = y_{t-1}$. We will train the model (using PyTorch) on the data $(x_1, y_1), \ldots, (x_T, y_T)$. After fitting the model, we will obtain predictions for $y_{T+1}, y_{T+2}, \ldots$ in the following way:

1. Run the LSTM equations over the observed $x_1, \ldots, x_T$ to compute the final hidden state (h_T, c_T).

2. Use the last observed value x_T as the first input for prediction

3. Recursively feed the previous prediction into the LSTM as the next input

To give more details, let (h_T, c_T) be the final hidden and cell states after the training data. Then for the first forecast step, we do:

$$
\begin{aligned}
&x_{T+1} = y_T \\
&\text{Compute } (i_{T+1}, f_{T+1}, o_{T+1}, \tilde{c}_{T+1}) \text{ using the LSTM equations,} \\
&c_{T+1} = f_{T+1} \odot c_T + i_{T+1} \odot \tilde{c}_{T+1} \\
&h_{T+1} = o_{T+1} \odot \tanh(c_{T+1}) \\
&\hat{y}_{T+1} = \beta_0 + \beta^T h_{T+1}.
\end{aligned}
$$

Then we set $x_{T+2} = \hat{y}_{T+1}$, and then use this value (along with (h_{T+1}, c_{T+1}) calculated above to get h_{T+2}, c_{T+2} and from there $\hat{y}_{T+2} = \beta_0 + \beta^T h_{T+2}$). Then we set $x_{T+3} = \hat{y}_{T+2}$ and proceed similarly.