Notes on LSTM's and GRU's

Yashar Ahmadian

February 17, 2018

1 Traditional recurrent rate networks

In recurrent rate networks (written in the "rate form") the state of the network is the vector of neuronal firing rates, $\mathbf{r}(t) \in \mathbb{R}^N$, which evolves according to the system of (continuous-time) equations:

$$\boldsymbol{\tau} \odot \frac{d\mathbf{r}}{dt} = -\mathbf{r} + \varphi(W\mathbf{r} + \mathbf{I}) \tag{1}$$

Here $\odot$ denotes the element-wise product of two vectors, $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_N)^{\mathrm{T}}$ is the vector of single-neuronal relaxation time constants, φ is the neuronal input-output nonlinearity, W is the matrix of recurrent synaptic weights, and $\mathbf{I}(t)$ is the vector of external inputs. The latter can be written as $\mathbf{I}(t) = P\mathbf{x}(t) + \mathbf{b}$, where $\mathbf{x}(t) \in \mathbb{R}^M$ is the vector of activations across M input channels (or M external neurons), the $N \times M$ matrix P can be thought of as a matrix of feedforward weights, and $\mathbf{b}$ is a constant vector of "biases".

If we discretize time *a la* Euler (i.e. replacing $\frac{d\mathbf{r}}{dt}$ with $\frac{\mathbf{r}_t - \mathbf{r}_{t-1}}{\Delta t}$, with the identification $\mathbf{r}_t = \mathbf{r}(t\Delta t)$), we obtain the update rule:[1]

$$\mathbf{r}_t = (1 - \mathbf{z}) \odot \mathbf{r}_{t-1} + \mathbf{z} \odot \varphi(W\mathbf{r}_{t-1} + P\mathbf{x}_t + \mathbf{b}) \tag{2}$$

where I defined $\mathbf{z}$ to be the vector with *constant* components

$$z_i = \frac{\Delta t}{\tau_i}, \qquad\qquad i = 1, \ldots, N. \tag{3}$$

In machine learning and artificial neural nets, $\varphi(u) = \tanh(u)$ is a common choice. If biological realism is desired, however, $\varphi(u)$ must have a **non-negative output**, and the matrix W and P must satisfy **Dale's principle**,[2] *i.e.*, matrix elements in a given column must have the same sign. For example, a biologically motivated $\varphi(u)$ is the rectified supralinear power-law function used in the Stabilized Supralinear Network: $\varphi(u) = \max(0, u)^n$ with $n > 1$.

[1] Note that in the right hand side of Eq. (2), we would have more uniform if we had $P\mathbf{x}_{t-1}$ instead of $P\mathbf{x}_t$; but the difference between the two choices amounts to a single time-shift of the external sources and a corresponding change of convention for relative timing of external and network variable timing. I will keep the current notation for consistency with machine learning custom.

[2] Biologically, Dale's principle says that neurons only express one neurotransmitter. Since the main (ionotropically acting) neurotransmitters in the brain are glutamate (excitatory) and GABA (inhibitory) this means that the synaptic projections of a given neuron are either all excitatory/positive or all inhibitory/negative, which leads to the mathematical statement in terms of matrix columns.

Rate equations are often also written in the "voltage notation", with state-vector $\mathbf{v}(t)$ (for the relationship between the rate and voltage notations see Miller and Fumarola (2012)). In this case the continuous-time version of the equations of motion read

$$\boldsymbol{\tau} \odot \frac{d\mathbf{v}}{dt} = -\mathbf{v} + W\varphi(\mathbf{r}) + \mathbf{I}, \tag{4}$$

which leads to the discrete-time update rule:

$$\mathbf{v}_t = (1 - \mathbf{z}) \odot \mathbf{v}_{t-1} + \mathbf{z} \odot [W\varphi(\mathbf{v}_{t-1}) + P\mathbf{x}_t + \mathbf{b}] \,. \tag{5}$$

2 Long Short-term Memory (LSTM) Networks

From here on, time is discrete.

LSTM networks have state-vectors $\mathbf{c}_t$ (one variable per unit/neuron). The unit/neuron outputs are however distinct and are denoted by $\mathbf{h}_t$. Each unit also has three "gate" variables $\mathbf{i}_t, \mathbf{o}_t, \mathbf{f}_t$, respectively called input, output and "forget" gates. The latter take values in the interval $[0, 1]$ and gate the inputs, outputs, and "memory" of the LSTM units.

The updates in the commonly used notation are given by

$$\begin{aligned} \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \varphi_1(W\,\mathbf{h}_{t-1} + P\,\mathbf{x}_t + \mathbf{b}) && (6)\\ \mathbf{h}_t &= \mathbf{o}_t \odot \varphi_2(\mathbf{c}_t) && (7)\\ \boldsymbol{y}_t &= \sigma(W_y\,\mathbf{h}_{t-1} + P_y\,\mathbf{x}_t + \mathbf{b}_y) \qquad \boldsymbol{y} \in \{\mathbf{i}, \mathbf{o}, \mathbf{f}\}. && (8) \end{aligned}$$

The standard choice of nonlinearities is

$$\begin{aligned} \sigma(u) &= \frac{1}{1 + e^{-u}} && (9)\\ \varphi_1(u) = \varphi_2(u) &= \tanh(u) && (10) \end{aligned}$$

Since we want the gating variables to be in $[0, 1]$ the choice of logistic/sigmoid function for $\sigma(u)$ is rather canonical; but in principle other (possibly unidentical) choices of φ_1 and φ_2 would/could be useful too).

The learnable/trainable parameters of the network are $(W, P, \mathbf{b}, W_{\mathbf{i}}, P_{\mathbf{i}}, \mathbf{b}_{\mathbf{i}}, W_{\mathbf{o}}, P_{\mathbf{o}}, \mathbf{b}_{\mathbf{o}}, W_{\mathbf{f}}, P_{\mathbf{f}}, \mathbf{b}_{\mathbf{f}})$, that given some training set of input $((vx_t)_{t=1}^T)$ and target sequences are usually trained using some variant of (stochastic) gradient descent via back-propagation through time.

By substituting Eq. (7) into Eq. (6) the update rules can be more succinctly written as

$$\begin{aligned} \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \varphi_1(W\,\mathbf{o}_{t-1} \odot \varphi_2(\mathbf{c}_{t-1}) + P\,\mathbf{x}_t + \mathbf{b}) && (11)\\ \boldsymbol{y}_t &= \sigma(W_y\,\mathbf{o}_{t-1} \odot \varphi_2(\mathbf{c}_{t-1}) + P_y\,\mathbf{x}_t + \mathbf{b}_y) \qquad \boldsymbol{y} \in \{\mathbf{i}, \mathbf{o}, \mathbf{f}\}. && (12) \end{aligned}$$

But the representation Eqs. (6)–(8) is nice in explicitly revealing that different LSTM units only communicate through their $\mathbf{h}_t$ outputs. It also makes the Markovian state-structure explicit.

Now imagine that all $\mathbf{o}_t$ was fixed to 1, $\mathbf{i}_t$ was fixed to a constant vector $\mathbf{z}$, and $\mathbf{f}_t$ was fixed to $1 - \mathbf{z}$. Then we would have

$$\mathbf{c}_t = (1 - \mathbf{z}) \odot \mathbf{c}_{t-1} + \mathbf{z} \odot \varphi_1(W\,\varphi_2(\mathbf{c}_t) + P\,\mathbf{x}_t + \mathbf{b}) \tag{13}$$

If we take φ_2 (respectively, φ_1) to be linear/identity, then these become identical to the usual rate equations written in the "rate format" with $\mathbf{r}_t \equiv \mathbf{c}_t$ ("voltage format" with $\mathbf{v}_t \equiv \mathbf{c}_t$), discussed in Sec. 1. On the other hand, with both φ_2 and φ_1 nonlinear but positive, we can think of one of these nonlinearities as a synaptic nonlinearity for example resulting from (very fast) short-term synaptic depression.

3 Gated recurrent units (GRU)

GRU's are simpler versions of gated recurrent nets that in complexity stand somewhere between LSTM's and traditional (non-gated) recurrent nets. They differ from LSTM's in that they have only two gating variables instead of three, and also the nonlinearity φ_2 is discarded (*i.e.*, it's taken to be identity).

I will first introduce a model that I call a **simple-GRU**, by starting from traditional rate equations Eq. (2) but replacing the constants $\mathbf{z}$ (related to neuronal time-constant as in Eq. (3)) by dynamical gating variables updated as in LSTM:

$$\begin{aligned} \mathbf{r}_t &= (1 - \mathbf{z}_t) \odot \mathbf{r}_{t-1} + \mathbf{z}_t \odot \varphi(W\mathbf{r}_{t-1} + P\mathbf{x}_t + \mathbf{b}) && (14) \\ \mathbf{z}_t &= \sigma(W_z\mathbf{r}_{t-1} + P_z\mathbf{x}_t + \mathbf{b}_z) && (15) \end{aligned}$$

The case of long time-constants that corresponds to input integration (and long memory time-scale for a unit) corresponds to $\mathbf{z} \to 0$, while $\mathbf{z} \to 1$ corresponds to no memory. *So here the network can be trained so that each unit can independently and dynamically interpolate between these two extremes depending on the current state of activity across the network.* Note that if we identify $\mathbf{c}_t$ of LSTM's with $\mathbf{r}_t$ we see that this is exactly like the LSTM, except the nonliearity φ_2 and the output gates are discarded, while the forget and input gates are taken to be precisely anti-correlated and related via $\mathbf{i}_t = 1 - \mathbf{f}_t = \mathbf{z}_t$.

The standard GRU's used in deep learning introduce another gating variable which is almost, but not quite, like the output gate of LSTM's. It's not quite like the LSTM output gate, because it only enters in Eq. (14) and not in Eq. (15). It also differs in the temporal alignment of its two factors compared to those in Eq. (7) (which is important in allowing the dynamical system form, Eq. (**??**) below, for state variables h_t, which is not possible to achieve for LSTM's in terms of state-variables c_t, but would have been if Eq. (7) used $\mathbf{o}_{t+1}$ instead of $\mathbf{o}_t$, and in the current formulation is achievable in terms of the $\mathbf{h}_t$ which are indeed state-variables for LSTM). Thus a GRU update rule is given by

$$\begin{aligned} \mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \varphi(W\mathbf{o}_t \odot \mathbf{h}_{t-1} + P\mathbf{x}_t + \mathbf{b}) && (16) \\ \mathbf{z}_t &= \sigma(W_z\mathbf{h}_{t-1} + P_z\mathbf{x}_t + \mathbf{b}_z) && (17) \\ \mathbf{o}_t &= \sigma(W_o\mathbf{h}_{t-1} + P_o\mathbf{x}_t + \mathbf{b}_o) && (18) \end{aligned}$$

where I changed notation by replacing $\mathbf{r}_t$ with $\mathbf{h}_t$. The above equations are still written in non-standard notation. In the more common notation, $\mathbf{o}_t$ is denoted by $\mathbf{r}_t$ and is called the **reset gate**, while $\mathbf{z}_t$ is redefined via $\mathbf{z}_t \equiv 1 - \mathbf{z}_t$ and is referred to as the **update gate**. See:
`https://en.wikipedia.org/wiki/Gated_recurrent_unit`

The GRU update can be written in the form

$$\begin{aligned} \mathbf{h}_t &= \mathcal{F}(\mathbf{h}_{t-1}, \mathbf{x}_t) && (19) \\ &= \sigma(-W_z\mathbf{h}_{t-1} - P_z\mathbf{x}_t - \mathbf{b}_z) \odot \mathbf{h}_{t-1} \\ &\quad + \sigma(W_z\mathbf{h}_{t-1} + P_z\mathbf{x}_t + \mathbf{b}_z) \odot \varphi[W(\sigma(W_o\mathbf{h}_{t-1} + P_o\mathbf{x}_t + \mathbf{b}_o) \odot \mathbf{h}_{t-1}) + P\mathbf{x}_t + \mathbf{b}] && (20) \end{aligned}$$

References

Miller, K. D. and Fumarola, F. (2012). Mathematical equivalence of two common forms of firing rate models of neural networks. *Neural Comput.*, **24**(1), 25–31.