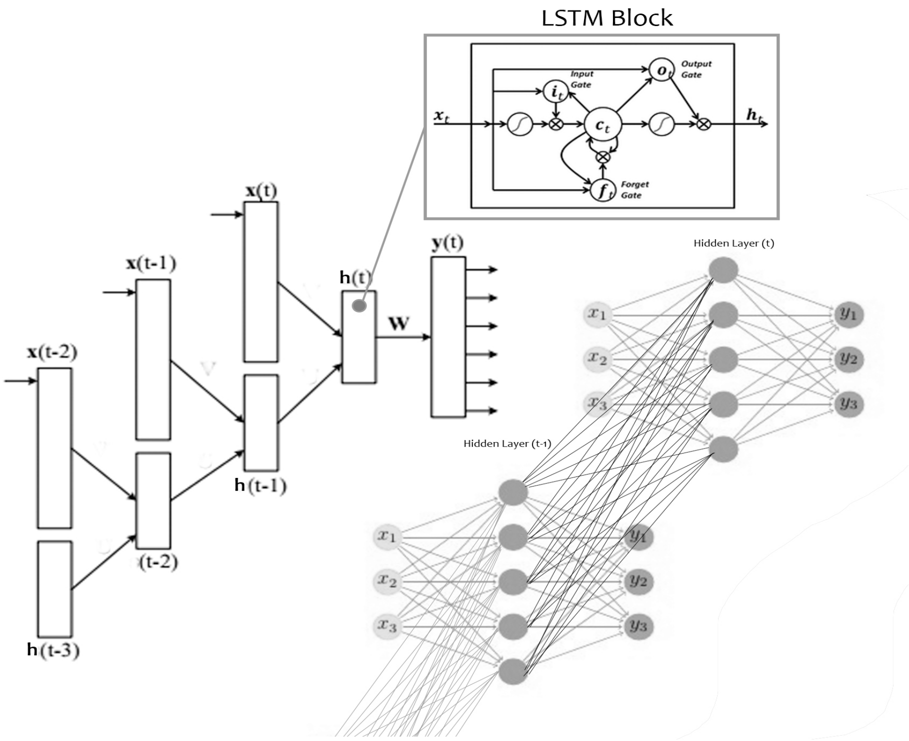
LSTM Recurrent Neural Network Implementation

Andreas Theophilou

September 7, 2016

1 Introduction



2 FeedForward

Input and Context Layers to LSTMBlock (hidden) outputs

InputGate

$$iGate_j^t = \sigma(wi_j^{(x)} xl_t + wi_j^{(h)} hl_{t-1} + wi_j^{(c)} cell_{t-1} + b_j^{(i)})$$

iG Weight Connections

$$wi_j^{(x)} xl_t = \begin{pmatrix} who_{11} & \cdots & who_{1i} \\ \vdots & \ddots & \vdots \\ who_{j1} & \cdots & who_{ji} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ .. \\ .. \\ h_j \end{pmatrix} : wi_j^{(h)} hl_{t-1} = \begin{pmatrix} who_{11} & \cdots & who_{1i} \\ \vdots & \ddots & \vdots \\ who_{j1} & \cdots & who_{ji} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ .. \\ .. \\ h_j \end{pmatrix}$$

$$wi_j^{(c)} cell_{t-1} = \begin{pmatrix} who_{11} & \cdots & who_{1i} \\ \vdots & \ddots & \vdots \\ who_{j1} & \cdots & who_{ji} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ .. \\ .. \\ h_j \end{pmatrix}$$

CellGate

$$g_j^{(t)} = \phi(wc_j^{(x)} xl_t + wc_j^{(h)} hl_j^{(t-1)} + b_j^{(c)})$$
$$cell_j^t = fGate_j^{(t)} * cell_j^{(t-1)} + i_j^{(t)} g_j^{(t)}$$

ForgetGate

$$fGate_j^t = \sigma(wf_j^{(x)} xl_t + wf_j^{(h)} hl_{t-1} + wf_j^{(c)} cell_{t-1} + b_j^{(f)})$$

OutputGate

$$oGate_j^t = \sigma(wo_j^{(x)} xl_t + wo_j^{(h)} hl_{t-1} + wo_j^{(c)} c_{t-1} + b_j^{(i)})$$

LSTMBlock output (hout)

$$z_j^{(t)} = \phi(cell_j^{(t)})$$
$$hl_j^t = oGate_j^{(t)} z_j^{(t)}$$

where σ is the transfer function sigmoid and ϕ is the transfer function tanh

$$\begin{pmatrix} who_{11} & \cdots & who_{1i} \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ who_{j1} & \cdots & who_{ji} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ .. \\ .. \\ h_j \end{pmatrix} = \begin{pmatrix} o_1 \\ o_2 \\ .. \\ o_i \end{pmatrix}$$

Softmax transfer function

$$y_i = \varphi(ol)_i = \frac{e^{ol_i}}{\sum_{k=1}^{K} e^{ol_k}} = \frac{e^{who_i^T hl}}{\sum_{k=1}^{K} e^{who_k^T hl}}$$

Given that most computative operations are of matrix * vector and with mini batching matrix * matrix form, vectorizing loops can greatly improve the networks performance. For calculations performed on the cpu making use of intel's SIMD instruction set extension can allow speeds to be multiple times faster at the loss of floating point precision.
Hidden Layer to Output layer

3 Back Propagation

3.1 via Classification

Multi-Class Cross Entropy

$$L(t,y) = -[\sum_{i=1}^{M}\sum_{c=1}^{C} 1_{(t_i=c)} * ln(y_{ic})] = -[\sum_{i=1}^{M}\sum_{c=1}^{C} 1_{(t_i=c)} * ln(\frac{e^{who_i^T hl}}{\sum_{k=1}^{K} e^{who_k^T hl}})]$$

Back Propagation

$$\frac{\partial L}{\partial ol} = -\sum_{i=1}^{M}[\sum_{c=1}^{C} \frac{t_c}{y_c}\frac{\partial y_c}{\partial ol_i}]$$

Derivative of the softmax activation function

$$\text{if } c = i : \frac{\partial y_c}{\partial ol_i} = \frac{\partial y_i}{\partial ol_i} = \frac{\partial \frac{e^{ol_i}}{\sum_k e^{ol_k}}}{\partial ol_i} = \frac{(\frac{\partial}{\partial ol_i} e^{ol_i})(\sum_k e^{ol_k}) - e^{ol_i}(\frac{\partial}{\partial ol_i}(\sum_k e^{ol_k}))}{(\sum_k e^{ol_k})^2}$$

$$= \frac{e^{ol_i}(\sum_k e^{ol_k}) - e^{ol_i}e^{ol_i}}{(\sum_k e^{ol_k})^2} = (\frac{e^{ol_i}}{\sum_k e^{ol_k}}) - (\frac{e^{ol_i}}{\sum_k e^{ol_k}})^2 = \varphi(ol)_i - \varphi(ol)_i^2$$

$$= y_i(1-y_i)$$

$$\text{if } c \neq i : \frac{\partial y_c}{\partial ol_i} = \frac{(\frac{\partial}{\partial ol_i} e^{ol_c})(\sum_k e^{ol_k}) - e^{ol_c}(\frac{\partial}{\partial ol_i}(\sum_k e^{ol_k}))}{(\sum_k e^{ol_k})^2}$$

$$= \frac{(0) - e^{ol_c}(e^{ol_i})}{(\sum_k e^{ol_k})^2} = -(\frac{e^{ol_i}}{\sum_k e^{ol_k}})(\frac{e^{ol_c}}{\sum_k e^{ol_k}}) = -\varphi(ol)_i\varphi(ol)_c$$

$$= -y_i y_c$$

therefore

$$\sum_{c=1}^{C}\frac{t_c}{y_c}\frac{\partial y_c}{\partial ol_i} = \frac{t_i}{y_i}\frac{\partial y_i}{\partial ol_i} + \sum_{c\neq i}^{C}\frac{t_c}{y_c}\frac{\partial y_c}{\partial ol_i} = \frac{t_i}{y_i}(y_i(1-y_i)) + \sum_{c\neq i}^{C}\frac{t_c}{y_c}(-y_i y_c)$$

$$= t_i - t_i y_i + \sum_{c\neq i}^{C}(-t_c y_i) = t_i + \sum_{c=1}^{C}(-t_c y_i) = t_i - y_i\sum_{c=1}^{C}(t_c) = t_i - y_i$$

complete derivative with respect to the loss function

$$\frac{\partial L}{\partial ol} = -\sum_{i=1}^{M}[t_i - y_i] = \sum_{i=1}^{M}[y_i - t_i] = \sum_{i=1}^{M}[\delta_i]$$

$$\therefore \frac{\partial L}{\partial ol_i} = y_i - t_i = \delta_i$$

3.2 Weight Updates

$$\frac{\partial L}{\partial who_{ji}} = \frac{\partial L}{\partial ol_i}\frac{\partial ol_i}{\partial who_{ji}} = \delta_i \frac{\partial ol_i}{\partial who_{ji}} = \delta_i h_j = \delta_i h_j$$

Adagrad:

$$\Delta whoCache_{ji} \leftarrow whoCache_{ji} + (\delta_i h_j)^2$$

$$\Delta who_{ji} \leftarrow who_{ji} - \eta * \frac{(\delta_i h_j)}{\sqrt{\Delta whoCache_{ji} + 10^{-8}}}$$

Hidden Layer Gradients

$$\frac{\partial L}{\partial hl_j} = \frac{\partial L}{\partial ol}\frac{\partial ol}{\partial hl_j} = \sum_{i=1}^{M}[y_i - t_i]\frac{\partial ol}{\partial hl_j} = \sum_{i=1}^{M}\delta_i who_{ji}$$

LSTM Derivatives

LSTM internal Backward Pass

$$\frac{\partial hl}{\partial cell_j} = (\sum_{i=1}^{M}\delta_i who_{ji}) * oGate_j * (1 - z_j z_j) * iGate_j * (1 - g_j g_j)$$

$$\frac{\partial hl}{\partial iGate_j} = (\sum_{i=1}^{M}\delta_i who_{ji}) * oGate_j * (1 - z_j z_j) * g_j * (\sigma(iGate_j)(1 - \sigma(iGate_j))$$

$$\frac{\partial hl}{\partial oGate_j} = (\sum_{i=1}^{M}\delta_i who_{ji}) * \sigma(oGate_j) * z$$

$$\frac{\partial hl}{\partial fGate_j} = (\sum_{i=1}^{M}\delta_i who_{ji}) * oGate_j * (1 - z_j z_j) * (\sigma(fGate_j)(1 - \sigma(fGate_j)) * cell_j^{(t-1)}$$

Internal Cell Bridge Weight Updates:
Adagrad:

$$\Delta wiCache_j^{(c)} \leftarrow wiCache_j^{(c)} + (\nabla iGate_j * cell_j^{(t-1)})^2 : wi_j^{(c)} \leftarrow wi_j^{(c)} - \eta * \frac{(\nabla iGate_j * cell_j^{(t-1)})}{\sqrt{\Delta wiCache_j^{(c)} + 10^{-8}}}$$

$$\Delta wfCache_j^{(c)} \leftarrow wfCache_j^{(c)} + (\nabla fGate_j * cell_j^{(t-1)})^2 : wf_j^{(c)} \leftarrow wf_j^{(c)} - \eta * \frac{(\nabla fGate_j * cell_j^{(t-1)})}{\sqrt{\Delta wfCache_j^{(c)} + 10^{-8}}}$$

$$\Delta woCache_j^{(c)} \leftarrow woCache_j^{(c)} + (\nabla oGate_j * cell_j^{(t-1)})^2 : wo_j^{(c)} \leftarrow wo_j^{(c)} - \eta * \frac{(\nabla oGate_j * cell_j^{(t-1)})}{\sqrt{\Delta woCache_j^{(c)} + 10^{-8}}}$$

Updating input and context weights
...