50.039 – Theory and Practice of Deep learning

Alex

Week 08

1 Task 1:

The following are the defining equations for a LSTM cell with a forget gate f_t,

$$
\begin{aligned}
i_t &= \sigma(W^i x_t + U^i h_{t-1}) \\
f_t &= \sigma(W^f x_t + U^f h_{t-1}) \\
o_t &= \sigma(W^o x_t + U^o h_{t-1}) \\
u_t &= \tanh(W^c x_t + U^c h_{t-1}) \\
c_t &= f_t \circ c_{t-1} + i_t \circ u_t \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
$$

The symbol $\circ$ denotes element-wise multiplication and $\sigma(x) = \frac{1}{1+e^{-x}}$

- is c_{t-1} as function of h_{t-1}? Give 3 sentences at most as answer why or why not.

  if c_{t-1} were a function of h_{t-1}, then c_t is a function of h_t, too. This is not the case according to the equations. c_t is computed from f_t,c_{t-1} i_t u_t, all of them depend only on x_t, h_{t-1}. So: nope!

- Write down the derivative $\frac{\partial h_t}{\partial h_{t-1}}$ (which is the equivalent of the term $\frac{\partial s_t}{\partial s_{t-1}}$ in the lecture) expressed as a function of $\frac{\partial i_t}{\partial h_{t-1}}$, $\frac{\partial f_t}{\partial h_{t-1}}$, $\frac{\partial o_t}{\partial h_{t-1}}$, $\frac{\partial u_t}{\partial h_{t-1}}$. You do not need to resolve the terms $\frac{\partial i_t}{\partial h_{t-1}}$, $\frac{\partial f_t}{\partial h_{t-1}}$, $\frac{\partial o_t}{\partial h_{t-1}}$, $\frac{\partial u_t}{\partial h_{t-1}}$.

  Note that for element-wise multiplications the product rule holds, even if you multiply two vectors.

  You do not need to answer this question: Do you understand why I do **not** want you to explicitly compute by hand $\frac{\partial h_t}{\partial h_{t-3}}$?

Solution:

$$
\begin{aligned}
h_t &= o_t \circ \tanh(c_t) \\
\frac{\partial h_t}{\partial h_{t-1}} &= \frac{\partial o_t}{\partial h_{t-1}} \circ \tanh(c_t) + o_t \circ (\tanh'(c_t) \frac{\partial c_t}{\partial h_{t-1}}) \\
c_t &= f_t \circ c_{t-1} + i_t \circ u_t \\
\frac{\partial c_t}{\partial h_{t-1}} &= \frac{\partial f_t}{\partial h_{t-1}} \circ c_{t-1} + f_t \circ \frac{\partial c_{t-1}}{\partial h_{t-1}} + \frac{\partial i_t}{\partial h_{t-1}} \circ u_t + i_t \circ \frac{\partial u_t}{\partial h_{t-1}} \\
&= \frac{\partial f_t}{\partial h_{t-1}} \circ c_{t-1} + 0 + \frac{\partial i_t}{\partial h_{t-1}} \circ u_t + i_t \circ \frac{\partial u_t}{\partial h_{t-1}}
\end{aligned}
$$

- Calculate $\sigma'(z)$, the derivative of the sigmoid (not all tasks can be hard, right?)

$$
\begin{aligned}
\sigma(z) &= \frac{1}{1+e^{-z}} \\
\sigma'(z) &= -1 \frac{1}{(1+e^{-z})^2}(-e^{-z}) = \frac{e^{-z}}{(1+e^{-z})^2}
\end{aligned}
$$

- Write down the derivative $\frac{\partial f_t}{\partial h_{t-1}}$ expressed as a function of σ' and other terms as required. Lets look at a single dimension – d_1 for the output f_t, and d_1 for the input h_{t-1}

$$
\begin{aligned}
f_t &= \sigma(W^f x_t + U^f h_{t-1}) \\
f_t[d_1] &= \sigma(W^f[d_1,:] x_t + U^f[d_1,:] h_{t-1})
\end{aligned}
$$

Note that W^f is a matrix, and $W^f[d_1,:]$ is a vector. Same holds for U^f and $U^f[d_1,:]$

$$
\begin{aligned}
f_t[d_1] &= \sigma(W^f[d_1,:] \cdot x_t + U^f[d_1,:] \cdot h_{t-1}) \\
\frac{\partial f_t[d_1]}{\partial h_{t-1}[d_2]} &= \sigma'(W^f[d_1,:] \cdot x_t + U^f[d_1,:] \cdot h_{t-1}) \frac{\partial (U^f[d_1,:] \cdot h_{t-1})}{\partial h_{t-1}[d_2]}
\end{aligned}
$$

$U^f[d_1,:] \cdot h_{t-1} \in \mathbb{R}^1$ is just an inner product:

$$
U^f[d_1,:] \cdot h_{t-1} = \sum_d U^f(d_1,d) \cdot h_{t-1}(d)
$$

which is a sum of components $U^f[d_1,d] \cdot h_{t-1}[d]$ for the d-th dimension. Therefore

$$
\begin{aligned}
\frac{\partial (U^f[d_1,:] \cdot h_{t-1})}{\partial h_{t-1}[d_2]} &= \frac{\partial (U^f[d_1,d_2] \cdot h_{t-1}[d_2]}{\partial h_{t-1}[d_2]} \\
&= U^f[d_1,d_2]
\end{aligned}
$$

$U^f[d_1,d_2]$ is the entry $U^f[d_1,d_2]$ of the matrix U^f in terms of numpy / pytorch indexing. Therefore:

$$
\begin{aligned}
f_t[d_1] &= \sigma(W^f[d_1,:] x_t + U^f[d_1,:] h_{t-1}) \\
\frac{\partial f_t[d_1]}{\partial h_{t-1}[d_2]} &= \sigma'(W^f[d_1,:] \cdot x_t + U^f[d_1,:] \cdot h_{t-1}) U^f[d_1,d_2]
\end{aligned}
$$

- how to activate a gate ? Why is this important?

  If an LSTM has a forget gate, then it is good practice to initialize it such that f_t is a vector of values close to 1 at the start of the training. This is done usually by adjusting the bias terms (, which we dropped here, though).

  Why one wants $f_t \approx 1$ at init? If $f_t \approx 1$, then at the beginning we have

  $$c_t = f_t \circ c_{t-1} + i_t \circ u_t \approx 1 \circ c_{t-1} + i_t \circ u_t = c_{t-1} + i_t \circ u_t,$$

  and thus $\frac{\partial c_t}{\partial c_{t-1}} \approx 1$, that is the gradient flows back through time unchallenged through the memory cell vectors c_t .

  Back to the question: how to activate a gate ... not using a bias ?

  Consider

  $$f_t = \sigma(W^f x_t + U^f h_{t-1})$$

  The output is a vector because

  $x_t \in \mathbb{R}^d$ is a vector but $W_f \in \mathbb{R}^{k \times d}$ is a matrix, $h_{t-1} \in \mathbb{R}^k$ is a vector but $W_f \in \mathbb{R}^{k \times k}$ is a matrix,

  so the output will be a vector $\in \mathbb{R}^k$. Lets look at one component of the vector:

  $$f_t(d) = \sigma(W^f(d) \cdot x_t + U^f(d) \cdot h_{t-1})$$

  where $W^f(d)$ is a $\mathbb{R}^{1 \times d}$ row or column vector and $U^f(d)$ is a $\mathbb{R}^{1 \times k}$ row or column vector. Thus $W^f(d) \cdot x_t$ is just an inner product of two vectors.

  Question: if $x_t = 0$, $U^f(d) \neq 0$, which vector h_{t-1} among all the vectors of euclidean length 1 maximized the values of $f_t(d)$?

  Speaking in math, find

  $$\begin{aligned} &\text{argmax}_{\{h_{t-1}:\|h_{t-1}\|_2=1\}} f_t(d) \\ =\ &\text{argmax}_{\{h_{t-1}:\|h_{t-1}\|_2=1\}} \sigma(W^f(d) \cdot x_t + U^f(d) \cdot h_{t-1}) \end{aligned}$$

  Think geometrically (its an inner product, and the euclidean norm is $v \cdot v = \|v\|_2^2$!!!) to find the solution, then it is easy.

  The inner product $u \cdot v$ is maximized among all vectors of same length for that vector v which is parallel to u.

  $$\text{argmax}_{v:\|v\|_2=1} u \cdot v = \frac{u}{\|u\|_2}$$

  Therefore: $h_{t-1} = \frac{U^f(d)}{\|U^f(d)\|_2}$
  Side note: The solution depends on the output index d, that is for every d you have a different solution.

- in the task above, does the argmax depend on the value of $W^f(d) \cdot x_t$? Does the max depend on it? Give at most 3 sentences justification The argmax - the vector h_{t-1} which maximizes the objective does not depend on it. The value of the objective does depend on $W^f(d) \cdot x_t$. One can compare $x_t = 0$ – then we have $\sigma(U^f(d) \cdot \frac{U^f(d)}{\|U^f(d)\|_2}) = \sigma(1)$ versus $x_t = 10000\frac{W^f(d)}{\|W^f(d)\|_2}$b – then we would have $\sigma(10000 + 1)$ which is larger than $\sigma(1)$.

2 Task 2:

- You are given a 2-dimensional convolution with spatial size $(78, 84)$. When using a kernel of size $(5, 5)$ and stride 3 with padding of 2 , what will be the spatial size of the feature map which is the output of the convolution? Note that spatial size does not depend on the number of input or output channels.

$$floor(\frac{input + 2 * pad - kernelsize}{stride}) + 1$$
$$floor(\frac{78 + 2 * 2 - 5}{3}) + 1 = 26$$
$$floor(\frac{84 + 2 * 2 - 5}{3}) + 1 = 28$$

- You are given a 2-dimensional convolution with spatial size $(64, 64)$. When using a kernel of size $(3, 5)$ and stride 2 with padding of 0 , what will be the spatial size of the feature map which is the output of the convolution? Note same that spatial size does not depend on the number of input or output channels.

$$floor(\frac{input + 2 * pad - kernelsize}{stride}) + 1$$
$$floor(\frac{64 + 0 - 3}{2}) + 1 = 31$$
$$floor(\frac{64 + 0 - 5}{2}) + 1 = 30$$

- You are given a 1-dimensional convolution, when using a kernel of size 9 and stride 3 with padding 1. What spatial input size do you need to have,

so that you have a spatial output size of 16?

$$
\begin{aligned}
& floor(\frac{x+2*1-9}{3})+1=16 \\
\Rightarrow & floor(\frac{x+2*1-9}{3})=15 \\
\Rightarrow & (\frac{x+2*1-9}{3} \geq 15) \text{ and } (\frac{x+2*1-9}{3}<16) \\
\Rightarrow & (x-7 \geq 45) \text{ and } (x-7<48) \\
\Rightarrow & (x \geq 52) \text{ and } (x<55) \\
\Rightarrow & 52 \leq x<55
\end{aligned}
$$

How many trainable parameters are

- in a 2-D convolutional layer with input $(32, 19, 19)$, kernel size $(7, 7)$, stride 3, 64 kernel channels, no padding, no bias term?

  $32 * 7 * 7 * 64$

- how many multiplications and how many additions are performed in this case above?

  $floor(\frac{19+0-7}{3})+1=5$ outputs in one dimension. Alternative to see it: for an input of length 19 (an array with indices $0:18$), a kernel of size 7, without padding, can be placed at : $0:6, 3:9, 6:12, 9:15, 12:18$ - so 5 times.

  For a 2d-convolution that are $5*5=25$ inner products to be computed. Each inner product is with a kernel of size $(7, 7)$ - thus 49 multiplications and 48 addition ops (adding 2 elements is 1 add, adding 3 elements is 2 add ... etc). Therefore we have $49*n_{in}*25*n_{out}$ mul and $(49*n_{in}-1)*25*n_{out}$ add ops, where $n_{in}=32$ and $n_{out}=64$

- in a 2-D convolutional layer with input $(512, 25, 25)$, kernel size $(1, 1)$, stride 1, 128 kernel channels, padding 2, no bias term?

  $512 * 128$