CS 224N: Assignment #2

Saran Ahluwalia

2019-01-06

Question 1(c) Placeholder variables in Tensorflow represent trainable data, which is usually initialized according to some rules (e.g. with zeros or some random distribution) and then gradually modified with back-propagated errors. Feed dictionaries affords flexibility in associating various sets of labels, hyperparameters and data with the same graph while conducting batch training or model tuning.

Question 1(e) Tensorflow removes the need to define gradients explicitly by "symbolically" computing gradients for back propagation by using gradient functions pre-defined (registered) with each operation in the graph.

Question 2(a) The sentence *"I parsed this sentence correctly"* can be parsed through the following sequence:

stack	buffer	new dependency	transition
[ROOT]	[I, parsed, this, sentence, correctly]		Initial Config.
[ROOT, I]	[parsed, this, sentence, correctly]		`SHIFT`
[ROOT, I, parsed]	[this, sentence, correctly]		`SHIFT`
[ROOT, parsed]	[this, sentence, correctly]	parsed $\rightarrow$ I	`LEFT-ARC`
[ROOT, parsed, this]	[sentence, correctly]		`SHIFT`
[ROOT, parsed, this, sentence]	[correctly]		`SHIFT`
[ROOT, parsed, sentence]	[correctly]	sentence $\rightarrow$ this	`LEFT-ARC`
[ROOT, parsed]	[correctly]	parsed $\rightarrow$ sentence	`RIGHT-ARC`
[ROOT, parsed, correctly]	[]		`SHIFT`
[ROOT, parsed]	[]	parsed $\rightarrow$ correctly	`RIGHT-ARC`
[ROOT]	[]	ROOT $\rightarrow$ parsed	`RIGHT-ARC`

Question 2(b) A sentence containing n words will be parsed in $2n$ steps since it takes one step to place a word onto the stack (`SHIFT` from the buffer) and another step (`LEF-ARC` or `RIGHT-ARC`) to remove it from the stack.

Question 2(f) In order to maintain overall magnitude of the layer's impact, it's expectation should not be affected by "dropped" parameters. For this reason:

$$\forall i : h_i = \mathbb{E}_{pdrop}[h_{drop}]_i = \gamma(0p_{drop} + 1(1 - p_{drop}))h_i = \gamma(1 - p_{drop})h_i$$

Therefore:

$$1 = \gamma(1 - p_{drop}) \text{ or}$$
$$\gamma = \frac{1}{1 - p_{drop}}$$

Question 2(g) (i) Momentum-based optimization algorithms are similar to regular gradient descent but instead of precise gradient evaluated at any given iteration they use running average of gradient from several previous iterations. Regular gradient descent leads the search into the steepest direction, which may be quite different from the optimal direction to the minimum if cost function gradients are very asymmetrical in different dimensions. This may lead to wild oscilations across the optimal search curve and momentum helps to smooth them out. (ii) Adaptive learning rate methods like Adam effectively select per-dimension learning rates that allows slowly-changing weights to use relatively large step sufficient to escape or rapidly traverse "plateaus" while quickly-changing ones are updated with smaller steps thus avoiding overshootings and oscillations around minimum.

Question 2(h) The submitted model has achieved 88.61% UAS on dev set and 89.01% on test set respectively.

Question 3(a) Assuming $\boldsymbol{y}^{(t)}$ is a one-hot vector, generic cross-entropy loss can be expressed as

$$CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -log(\hat{\boldsymbol{y}}^{(t)}_{k(t)}) \tag{1}$$

where $k(t)$ is the non-zero idex of $\boldsymbol{y}^{(t)}_{k(t)}$, i.e. $0 \neq \boldsymbol{y}^{(t)}_{k(t)}$. Similarly, the perplexity can also be expressed as

$$PP(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = \frac{1}{\hat{\boldsymbol{y}}^{(t)}_{k(t)}} \tag{2}$$

These expressions make it easier to see that there is a direct relationships between the two measures, specifically:

$$PP(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = e^{CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)})} \text{ or} \tag{3}$$

$$CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = log(PP(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)})) \tag{4}$$

For a "language model" that gives uniformly random "prediction" out of dictionary of $|V|$ words the perplexity of such prediction will be the size of the dictionary, i.e. $|V|$. Consequently, the corresponding cross-entropy will be $log(|V|)$ or for the 10000-word dictionary approximately 9.21034.

Question 3(b) Assuming the following variables:

$$e^{(t)} = x^{(t)}\boldsymbol{L} \tag{5}$$

$$z^{(t)} = h^{(t-1)}\boldsymbol{H} + e^{(t)}\boldsymbol{I} + \boldsymbol{b_1} \tag{6}$$

$$h^{(t)} = \sigma(z^{(t)}) \tag{7}$$

$$\theta^{(t)} = h^{(t)}\boldsymbol{U} + \boldsymbol{b_2} \tag{8}$$

$$\hat{y}^{(t)} = softmax(\theta^{(t)}) \tag{9}$$

$$J^{(t)} = CE(y^{(t)}, \hat{y}^{(t)}) \tag{10}$$

$$\delta^{(t)} = \frac{\partial J^{(t)}}{\partial h^{(t)}} = (\hat{y}^{(t)} - y^{(t)})\boldsymbol{U}^\top \tag{11}$$

$$\sigma'\delta^{(t)} = \frac{\partial J^{(t)}}{\partial z^{(t)}} = \delta^{(t)} \circ z^{(t)} \circ (1 - z^{(t)}) \tag{12}$$

$$\tag{13}$$

we can derive the following gradients

$$\frac{\partial J^{(t)}}{\partial \boldsymbol{b_2}} = \frac{\partial J^{(t)}}{\partial \theta^{(t)}} \frac{\partial \theta^{(t)}}{\partial \boldsymbol{b_2}} = (\hat{y}^{(t)} - y^{(t)}) \times 1 \qquad = \hat{y}^{(t)} - y^{(t)} \tag{14}$$

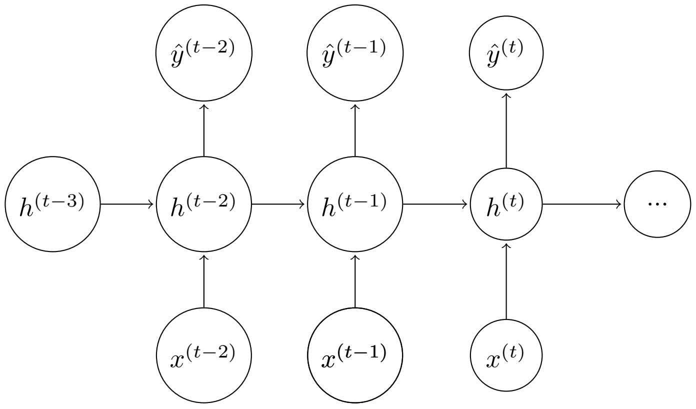
$$\frac{\partial J^{(t)}}{\partial \boldsymbol{L}_{x^{(t)}}} = \frac{\partial J^{(t)}}{\partial z^{(t)}} \frac{\partial z^{(t)}}{\partial e^{(t)}} \frac{\partial e^{(t)}}{\partial \boldsymbol{L}_{x^{(t)}}} = x^{(t)\top}(\sigma' \delta^{(t)} \boldsymbol{I}^\top) \qquad = x^{(t)\top}\sigma'\delta^{(t)}\boldsymbol{I}^\top \tag{15}$$

$$\left.\frac{\partial J^{(t)}}{\partial \boldsymbol{I}}\right|_{(t)} = \frac{\partial J^{(t)}}{\partial z^{(t)}} \left.\frac{\partial z^{(t)}}{\partial \boldsymbol{I}}\right|_{(t)} \qquad = e^{(t)\top}\sigma'\delta^{(t)} \tag{16}$$

$$\left.\frac{\partial J^{(t)}}{\partial \boldsymbol{H}}\right|_{(t)} = \frac{\partial J^{(t)}}{\partial z^{(t)}} \left.\frac{\partial z^{(t)}}{\partial \boldsymbol{H}}\right|_{(t)} \qquad = h^{(t-1)\top}\sigma'\delta^{(t)} \tag{17}$$

$$\frac{\partial J^{(t)}}{\partial h_{(t-1)}} = \frac{\partial J^{(t)}}{\partial z^{(t)}} \frac{\partial z^{(t)}}{\partial h^{(t-1)}} = \delta^{(t-1)} \qquad = \sigma'\delta^{(t)}\boldsymbol{H}^\top \tag{18}$$

Question 3(c) The following is a sketch for the 3-timestep-unrolled network:


$$\begin{tikzcd}
 & \hat{y}^{(t-2)} & \hat{y}^{(t-1)} & \hat{y}^{(t)} & \\
h^{(t-3)} \arrow[r] & h^{(t-2)} \arrow[r] \arrow[u] & h^{(t-1)} \arrow[r] \arrow[u] & h^{(t)} \arrow[r] \arrow[u] & \cdots \\
 & x^{(t-2)} \arrow[u] & x^{(t-1)} \arrow[u] & x^{(t)} \arrow[u] &
\end{tikzcd}$$

$$\frac{\partial J^{(t)}}{\partial \boldsymbol{L}_{x^{(t-1)}}} = \frac{\partial J^{(t)}}{\partial z^{(t-1)}} \frac{\partial z^{(t-1)}}{\partial e^{(t-1)}} \frac{\partial e^{(t-1)}}{\partial \boldsymbol{L}_{x^{(t-1)}}} \qquad = x^{(t-1)\top}\sigma'\delta^{(t-1)}\boldsymbol{I}^\top \tag{19}$$

$$\left.\frac{\partial J^{(t)}}{\partial \boldsymbol{I}}\right|_{(t-1)} = \frac{\partial J^{(t)}}{\partial z^{(t-1)}} \left.\frac{\partial z^{(t-1)}}{\partial \boldsymbol{I}}\right|_{(t-1)} \qquad = e^{(t-1)\top}\sigma'\delta^{(t-1)} \tag{20}$$

$$\left.\frac{\partial J^{(t)}}{\partial \boldsymbol{H}}\right|_{(t-1)} = \frac{\partial J^{(t)}}{\partial z^{(t-1)}} \left.\frac{\partial z^{(t-1)}}{\partial \boldsymbol{H}}\right|_{(t-1)} \qquad = h^{(t-2)\top}\sigma'\delta^{(t-1)} \tag{21}$$

Question 3(d) Taking into account that $x^{(t)}$ is a one-hot vector, which allows simple indexation optimization, computational costs of (5) through (10) are $O(1)$, $O(D_h \times (D_h + d + 1))$, $O(D_h)$, $O(D_h \times |V|)$ and O(1) respectively. Hence, computing $J^{(t)}$ involves $O(D_h \times (D_h + d + |V|))$ operations. Since $|V|$ is two-three orders of magnitude greater than either d or D_h, it is evident that the cost is dominated by matrix multiplication by $\boldsymbol{U}$, i.e. is $O(|V| \times D_h)$.

Similarly one can demonstrate that backpropagation step is domitated by computing $\delta^{(t)}$ and $\frac{\partial J^{(t)}}{\partial \boldsymbol{U}}$, which involves $O(|V| \times D_h)$ operations. Consequently back-propagation for τ steps in time is $O(|V| \times D_h \times \tau)$-costly.

Note, that even though it may look like (15) involves $O(|V| \times D_h \times d)$ operations, similar indexing optimization reduces it to $O(D_h \times d)$.

In order to optimize this model one could suggest using negative sampling instead of softmax to compute the loss. This way instead of full $\boldsymbol{U}$ multiplication in (8) we would use only K randomly selected columns along with the positive one selected by non-zero index of $y^{(t)}$ thus doing only $O(K \times D_h)$ operations. During back-propagation, we would only need to update corresponding rows of U thus also incuring similar cost per time-step.