CS224D Assignment2

Nathan Wan

May 2016

1 Tensorflow Softmax

(c) Placeholder variables are nodes in the computation graph that represent entry points for the computation.

Feed dictionaries use a data structure that identifies concrete values for placeholder variables to invoke the computation.

(e) TensorFlow's automatic differentiation means the graph defines not only the forward computation, but also the backward graph of gradients that can be evaluated and optimized. That is, the gradients are defined by the computation graph that's constructed.

2 Deep Networks for Named Entity Recognition

(a) Firstly, note that for $tanh(z) = 2\sigma(2z) - 1$, we can see that

$$\begin{aligned}
\frac{d}{dz}tanh(z) &= \frac{d}{dz}2\sigma(2z) - 1 \\
&= 2 \cdot (1 - \sigma(2z))\sigma(2z)2 \\
&= (1 - 2\sigma(2z) + 1)(2\sigma(2z)) \\
&= (1 - tanh(z))(2\sigma(2z) + 1 - 1) \\
&= (1 - tanh(z))(1 + tanh(z)) \\
&= 1 - tanh(z)^2
\end{aligned}$$

Let $\theta = hU + b_2$ s.t. $\hat{y} = softmax(\theta)$. And from the previous homework, $\frac{\partial J}{\partial \theta} = (\hat{y} - y)$. For the first layer,

$$\begin{aligned}
\frac{\partial J}{\partial b_2} &= \frac{\partial \theta}{\partial b_2}\frac{\partial J}{\partial \theta} = (\hat{y} - y) \\
\frac{\partial J}{\partial U} &= \frac{\partial \theta}{\partial U}\frac{\partial J}{\partial \theta} = h^T(\hat{y} - y)
\end{aligned}$$

Also since $\frac{\partial J}{\partial \theta} = (\hat{y} - y)U^T$,

$$\frac{\partial J}{\partial b_1} = \frac{\partial h}{\partial b_1}\frac{\partial J}{\partial h} = (1-h^2)\circ(\hat{y}-y)U^T$$

$$\frac{\partial J}{\partial W} = \frac{\partial h}{\partial W}\frac{\partial J}{\partial h} = (x^{(t)})^T\left((1-h^2)\circ(\hat{y}-y)U^T\right)$$

Note that $\frac{\partial J}{\partial L_i}$ depends on $x^{(t)}$.

$$\frac{\partial J}{\partial x^{(t)}} = \frac{\partial h}{\partial W}\frac{\partial J}{\partial h} = \left((1-h^2)\circ(\hat{y}-y)U^T\right)W^T$$

$\frac{\partial J}{\partial L_i} = 0$ for all values $i \notin [t-1, t+1]$. Otherwise, $\frac{\partial J}{\partial L_i}$ is the selection of $\frac{\partial J}{\partial x^{(t)}}$ vector corresponding to the concatenation the created $x^{(t)}$ from L_i.

(b) Since J_{reg} only depends on W and U, the gradients of J_{full} are the same as J from the previous section for b_2, b_1, and L_i.

$$\frac{\partial J_{reg}}{\partial W_{ij}} = \frac{\lambda}{2}(2W_{ij}) \Rightarrow \frac{\partial J_{reg}}{\partial W} = \lambda W$$

and similarly for U, $\frac{\partial J_{reg}}{\partial U} = \lambda U$. So,

$$\frac{\partial J_{full}}{\partial W} = \frac{\partial J}{\partial W} + \frac{\partial J_{reg}}{\partial W} = \frac{\partial h}{\partial W}\frac{\partial J}{\partial h} = (x^{(t)})^T\left((1-h^2)\circ(\hat{y}-y)U^T\right) + \lambda W$$

and

$$\frac{\partial J_{full}}{\partial U} = \frac{\partial J}{\partial U} + \frac{\partial J_{reg}}{\partial U} = \frac{\partial \theta}{\partial U}\frac{\partial J}{\partial \theta} = h^T(\hat{y}-y) + \lambda U$$

3 Recurrent Neural Networks: Language Modeling

(a) Since $y^{(t)}$ are one hot vectors, if k is the index of the "hot" value,

$$J^{(t)} = -log(\hat{y}_k^{(t)})$$

and

$$PP^{(t)} = \frac{1}{\hat{y}_k^{(t)}}$$

From this we see

$$\begin{aligned}
PP^{(t)} = \frac{1}{\hat{y}_k^{(t)}} &= exp(log(\frac{1}{\hat{y}_k^{(t)}})) \\
&= exp(log(1) - log(\hat{y}_k^{(t))})) \\
&= exp(-log(\hat{y}_k^{(t))})) \\
&= exp(J^{(t)})
\end{aligned}$$

Because $PP^{(t)} \propto J^{(t)}$, optimizing one optimizes the other proportionally. So optimizing the arithmetic mean of the cross entropy loss, $J^{(t)}$ would mean optimizing for

$$\frac{1}{T}\sum_t J^{(t)} = \frac{1}{T}\sum_t log(PP^{(t)}) = \frac{1}{T}log(\prod_t PP^{(t)}) = log\left(\left(\prod_t PP^{(t)}\right)^{1/T}\right)$$

When predictions are completely random, $\bar{P}(x) = \frac{1}{|V|}$. So $PP^{(t)} = \frac{1}{\frac{1}{|V|}} = |V|$, and the cross entropy loss $J^{(t)} = log(|V|)$.

$$J^{(t)}\Big|_{|V|=2000} = log(2000) = 3.30$$

and

$$J^{(t)}\Big|_{|V|=2000} = log(10000) = 4$$

(b)

$$\frac{\partial J^{(t)}}{\partial U} = (h^{(t)})^T(\hat{y}^{(t)} - y^{(t)})$$

$$\frac{\partial J^{(t)}}{\partial b_2} = (\hat{y}^{(t)} - y^{(t)})$$

Since $\frac{\partial J^{(t)}}{\partial h^{(t)}} = (\hat{y} - y)U^T$,

$$\frac{\partial J^{(t)}}{\partial L_{x^{(t)}}} = \frac{\partial J^{(t)}}{\partial e^{(t)}} = \frac{\partial h^{(t)}}{\partial e^{(t)}}\frac{\partial J^{(t)}}{\partial h^{(t)}} = \left((1 - h^{(t)}) \circ h^{(t)} \circ (\hat{y} - y)U^T\right) I^T$$

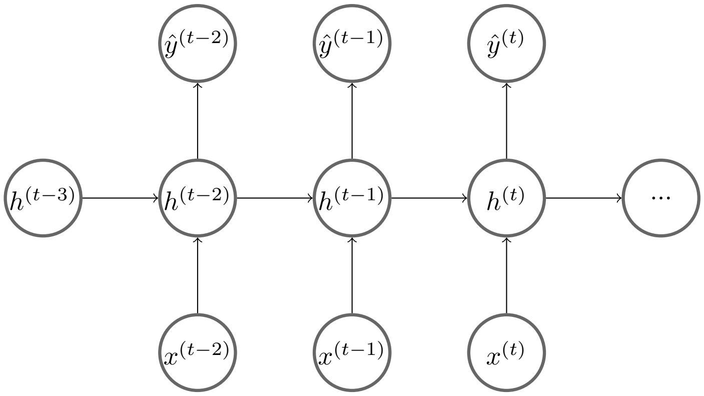
where $e^{(t)} \in \mathbb{R}^d$.

Evaluating at time step t, we hold $h^{(t-1)}$ constant:

$$\left.\frac{\partial J^{(t)}}{\partial I}\right|_{(t)} = (e^{(t)})^T \Big((1 - h^{(t)}) \circ h^{(t)} \circ (\hat{y} - y)U^T\Big)$$
$$\left.\frac{\partial J^{(t)}}{\partial H}\right|_{(t)} = (h^{(t-1)})^T \Big((1 - h^{(t)}) \circ h^{(t)} \circ (\hat{y} - y)U^T\Big)$$
$$\left.\frac{\partial J^{(t)}}{\partial b_1}\right|_{(t)} = \Big((1 - h^{(t)}) \circ h^{(t)} \circ (\hat{y} - y)U^T\Big)$$

and finally,

$$\frac{\partial J^{(t)}}{\partial h^{(t-1)}} = \Big((1 - h^{(t)}) \circ h^{(t)} \circ (\hat{y} - y)U^T\Big) H^T$$



(c)
$$\frac{\partial J^{(t)}}{\partial L_{x^{(t-1)}}} = \frac{\partial h^{(t-1)}}{\partial L_{x^{(t-1)}}} \frac{\partial J^{(t)}}{\partial h^{(t-1)}} = \Big((1 - h^{(t-1)}) \circ h^{(t-1)} \circ \delta^{(t-1)}\Big) I^T$$

and

$$\left.\frac{\partial J^{(t)}}{\partial I}\right|_{(t-1)} = (e^{(t-1)})^T \Big((1 - h^{(t-1)}) \circ h^{(t-1)} \circ \delta^{(t-1)}\Big)$$
$$\left.\frac{\partial J^{(t)}}{\partial H}\right|_{(t-1)} = (h^{(t-2)})^T \Big((1 - h^{(t-1)}) \circ h^{(t-1)} \circ \delta^{(t-1)}\Big)$$
$$\left.\frac{\partial J^{(t)}}{\partial b_1}\right|_{(t-1)} = \Big((1 - h^{(t-1)}) \circ h^{(t-1)} \circ \delta^{(t-1)}\Big)$$

(d) For a matrix multiplication of two matrices of dimensions $A \times B$ and $B \times C$, there are $A \cdot C \cdot (B + B - 1)$ operations because the output matrix has size

$A \times C$, and each element has B multiplications and $B-1$ additions, in other words $O(ABC)$.

For a given $h^{(t-1)}$, we need

$$sigmoid\left(O(D_h^2) + O(d^2 D_h) + O(D_h)\right) \equiv O(D_h^2)$$

operations to obtain $h^{(t)}$. For a given $h^{(t)}$, we need

$$-log\left(softmax\left(O(D_h|V|) + O(|V|)\right)\right) \equiv O(D_h|V|)$$

operations to obtain $J^{(t)}$. Since $d << D_h << |V|$, $O(D_h^2) + O(D_h|V|) \equiv O(D_h|V|)$.

One step of backpropagation would update all parameters in L, I, H, U, b_1, and b_2. The expensive step comes from calculating the δ's which costs:

$$(D_h \circ O(|V|D_h)) + O(D_h^2)$$

and $O(D_h^2)$ matrix multiplication at the most to get the individual variable gradients. For each additional step of back propagation, a $\circ D_h$ and $O(D_h^2)$ multiplication is required. So, backpropagation of τ steps would cost $O(|V|D_h) + \tau O(D_h^2)$. The slow steps are always the ones involving large multiplications of the vocabulary dimension $|V|$ because it is so large.

(f)

```
> <eos> funny slice regardless
<eos> funny <unk> regardless of millions of rockefeller route the
    negotiation <eos>

> <eos> lehman brothers
<eos> lehman brothers picking director of london corp. agreed to fend
     off its tender offer but some appreciation people minority data
    is designed to be able to arrange last year 's fourth quarter of
    N shares this by the exchange 's auction advanced attributable to
     ford spending theory in a joint venture valued at \$ N billion
    nyse said speculation on stateswest sought <unk> <eos>

> <eos> my great test
<eos> my great test legislation at odds out the mid-1990s <eos>

> <eos> my great test
<eos> my great test away but instead of these <unk> into japanese
    artists work and have a research collection that it occurs what
    but the shield has conducted the single spate of vermont 's
    existing board including new york <eos>
```