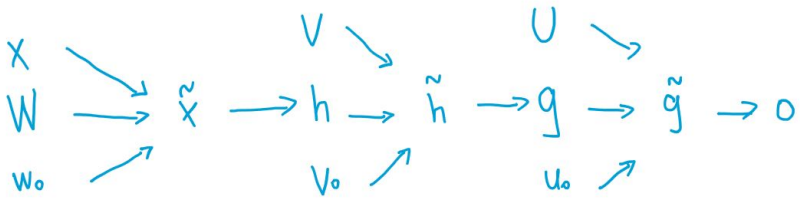
CSC 311 - Assignment 2

Hongyu Chen

November 15, 2020

1 4. Neural Networks: theory

1.1 4a)



1.2 4b)

We first prove that:

$\frac{\partial(c(t,o))}{\partial \tilde{g}_j} = \frac{\partial(-tlogo-(1-t)log(1-o))}{\partial \tilde{g}_j}$ (by equation (9))

$= -t\frac{logo}{\partial \tilde{g}_j} - (1-t)\frac{log(1-o)}{\partial \tilde{g}_j}$ (Since t is a binary value which is a constant)

$= -\frac{t}{o}\frac{\partial o}{\partial \tilde{g}_j} - \frac{1-t}{1-o}\frac{\partial(1-o)}{\partial \tilde{g}_j}$ (By chain rule)

$= -\frac{t_j}{o_j}(o_j(1-o_j)) + \frac{1-t_j}{1-o_j}(o_j(1-o_j))$ (by equation (8) $o_j = \sigma(\tilde{g_j})$)

$= -t_j(1-o_j) + (1-t_j)o_j$

$= -t_j + t_j o_j + o_j - o_j t_j$

$= o_j - t_j$

Then:

$[\frac{\partial C}{\partial \tilde{G}}]_{nj} = \frac{\partial C}{\partial \tilde{G}_{nj}} = \frac{\partial C}{\partial \tilde{g}_j^{(n)}}$ (by definition of gradient)

$= \frac{\partial(c(t^{(n)}, o^{(n)}))}{\partial \tilde{g}_j^{(n)}}$ (by equation (9) and we only care about o^n here since it's depends on $\tilde{g}_j^{(n)}$)

$= o_j^{(n)} - t_j^{(n)}$ (by the prove above)

$= O_{nj} - T_{nj} = [O - T]_{nj}$

Which means all of their components are equal to each other. So $\frac{\partial C}{\partial \tilde{G}} = O - T$

1.3 4c)

$[\frac{\partial C}{\partial \tilde{H}}]_{nk} = \frac{\partial C}{\partial \tilde{H}_{nk}}$ (by definition of gradient)

$= \frac{\partial C}{\partial \tilde{h}_k^{(n)}}$

$= \frac{\partial C}{\partial g_k^{(n)}} \frac{\partial g_k^{(n)}}{\partial \tilde{h}_k^{(n)}}$ (by chain rule)

$= \frac{\partial C}{\partial g_k^{(n)}}(1 - (g_k^{(n)})^2)$ (By equation (7) $g = tanh(\tilde{h})$ (equation 5))

$= \frac{\partial C}{\partial G_{nk}}(1 - (G_{nk})^2)$

$= (1 - G_{nk}^2)[\frac{\partial C}{\partial G}]_{nk}$

1.4 4d)

First we need to know:

$[\frac{\partial C}{\partial \tilde{H}}]_{nj} = \frac{\partial C}{\partial \tilde{H}_{nj}} = \frac{\partial C}{\partial \tilde{h}_j^{(n)}}$ (by definition of gradient)

$= \frac{\partial \sum_m c(t^m, o^m)}{\partial \tilde{h}_j^{(n)}}$ (by equation (9))

$= \frac{\partial(c(t^{(n)}, o^{(n)}))}{\partial \tilde{h}_j^{(n)}}$ (we only care about o^n here since it's depends on $\tilde{h}_j^{(n)}$)

Then:

$[\frac{\partial C}{\partial \tilde{V}}]_{kj} = \frac{\partial C}{\partial V_{kj}}$

$= \frac{\partial \sum_n c(t^{(n)}, o^{(n)})}{\partial V_{kj}}$ (by equation (9))

$= \sum_n \frac{\partial c(t^{(n)}, o^{(n)})}{\partial V_{kj}}$

$= \sum_n \frac{\partial c(t^{(n)}, o^{(n)})}{\partial \tilde{h}_j^{(n)}} \frac{\partial \tilde{h}_j^{(n)}}{\partial V_{kj}}$ (by chain rule)

$= \sum_n [\frac{\partial C}{\partial \tilde{H}}]_{nj} \frac{\partial \tilde{h}_j^{(n)}}{\partial V_{kj}}$ (by the prove about)

(By equation (5) we know that $\tilde{h} = hV + v_0$ so :

$\frac{\partial \tilde{h}_j^{(n)}}{\partial V_{kj}} = h_k^{(n)}$)

So we continue the calculation from above:

$= \sum_n [\frac{\partial C}{\partial \tilde{H}}]_{nj} h_k^{(n)}$
$= \sum_n [\frac{\partial C}{\partial \tilde{H}}]_{nj} H_{nk}$

$= \sum_n [\frac{\partial C}{\partial \tilde{H}}]_{nj} H_{kn}^T$

$= \sum_n [H^T]_{kn} [\frac{\partial C}{\partial \tilde{H}}]_{nj}$ (Matrix multiplication)

$= [H^T \frac{\partial C}{\partial \tilde{H}}]_{kj}$

Then $\frac{\partial C}{\partial V} = H^T \frac{\partial C}{\partial \tilde{H}}$ Since every components in these two are equal.

1.5 4e)

According to the proof from part 4(d) we know that :

$[\frac{\partial C}{v_0}]_j = \sum_n [\frac{\partial C}{\partial \tilde{H}}]_{nj} \frac{\partial \tilde{h}_j^{(n)}}{\partial v_{0_j}}$

$= \sum_n [\frac{\partial C}{\partial \tilde{H}}]_{nj} \vec{1}$ (By equation 5 ,v_0 is a constant)

$= \sum_n [\vec{1}]_n [\frac{\partial C}{\partial \tilde{H}}]_{nj}$

$= [\vec{1} \frac{\partial C}{\partial \tilde{H}}]_j$ (By matrix multiplication)

Then we know that $\frac{\partial C}{\partial v_0} = \vec{1} \frac{\partial C}{\partial \tilde{H}}$ Since every components in these two are equal.

1.6 4f)

$[\frac{\partial C}{\partial H}]_{nk} = \frac{\partial C}{\partial H_{nk}} = \frac{\partial C}{\partial h_k^{(n)}}$ (by definition of gradient)

$= \frac{\partial \sum_m c(t^m, o^m)}{\partial h_k^{(n)}}$ (by equation (9))

$= \frac{\partial (c(t^{(n)}, o^{(n)}))}{\partial h_k^{(n)}}$ (we only care about o^n here since it's depends on $h_k^{(n)}$)

$= \sum_j \frac{\partial (c(t^{(n)}, o^{(n)}))}{\partial \tilde{h}_j^n} \frac{\partial \tilde{h}_j^n}{\partial h_k^{(n)}}$ (By equation (10))

$= \sum_j [\frac{\partial C}{\partial \tilde{H}}]_{nj} \frac{\partial \tilde{h}_j^n}{\partial h_k^{(n)}}$ (By the first part proof of 4d))

$= \sum_j [\frac{\partial C}{\partial \tilde{H}}]_{nj} V_{kj}$ (By equation (5))

$= \sum_j [\frac{\partial C}{\partial \tilde{H}}]_{nj} V_{jk}^T$

$= [\frac{\partial C}{\partial \tilde{H}} V^T]_{nk}$ (By matrix multiplication)

Then we know that $\frac{\partial C}{\partial H} = \frac{\partial C}{\partial \tilde{H}} V^T$ Since every components in these two are equal.

2 5. Neural Networks: implementation

2.1 5e)

In this question we set the maximum number of iteration as 1. The reason for accuracy decreases with batch size, and cross entropy increases is for small batch size, the learning process will converges very quickly, but for the large batch size, the learning process will converge slowly. At the same iteration smaller batch size will converge faster. Here is the run time for each iteration of my program:

batch size (2**0) runtime: 23.266368627548218
batch size (2**1) runtime: 11.932782173156738
batch size (2**2) runtime: 5.774262189865112
batch size (2**3) runtime: 3.154838800430298
batch size (2**4) runtime: 1.7582886219024658
batch size (2**5) runtime: 1.2529397010803223
batch size (2**6) runtime: 0.8675491809844971
batch size (2**7) runtime: 0.6975538730621338
batch size (2**8) runtime: 0.5969803333282471
batch size (2**9) runtime: 0.5671384334564209
batch size (2**10) runtime: 0.5290675163269043
batch size (2**11) runtime: 0.5251491069793701
batch size (2**12) runtime: 0.5324108600616455
batch size (2**13) runtime: 0.5390825271606445

Using batch size of 1 is inefficient. The run time of using batch size 1 is very high. If we want to execute the program on a massively parallel machine, such as a gpu with batch size of 1, it will take a really long run time. GPUs are not inefficient for small batch sizes