EE488B Special Topics in EE <Deep Learning and AlphaGo> October 11, 2017
EE, KAIST, Fall 2017

Solution Set #2

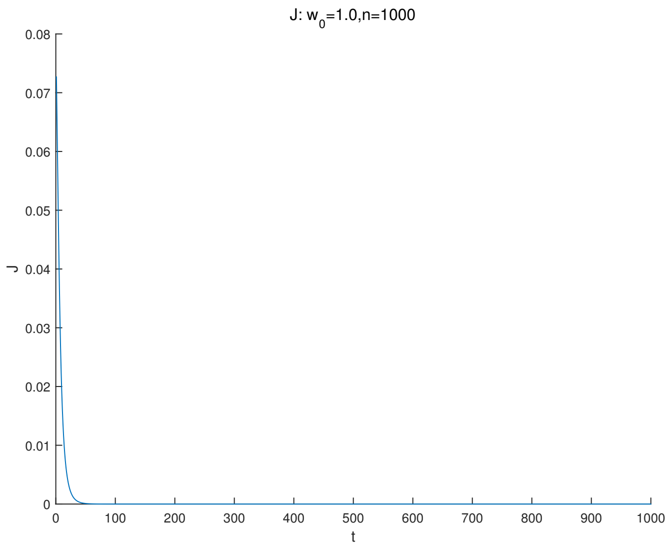
---

1. (Back-propagation for 100-layer network – 10 points) *Implement the training algorithm for Example 1 in page 42 in lecture notes #6. Assume $m = 2$ and $\mathbf{x} = \mathbf{y} = (-0.5, 0.5)^T$, i.e., we want to train a neural network to output 0.5 when the input is 0.5 and to output −0.5 when the input is −0.5. Assume $l = 100$, i.e., 100 layers. We want to see if such a deep neural network can be trained. Assume also $\alpha = 0.1$ (learning rate), MaxIter = 1000, and $\phi^{(k)}(x) = \tanh(x)$, $k = 1, 2, \ldots, l$. Note that $\tanh'(x) = \text{sech}^2(x) = \frac{4}{(e^x + e^{-x})^2}$, which can be used in your code. Include codes and plots in your homework. You may use any programming language.*

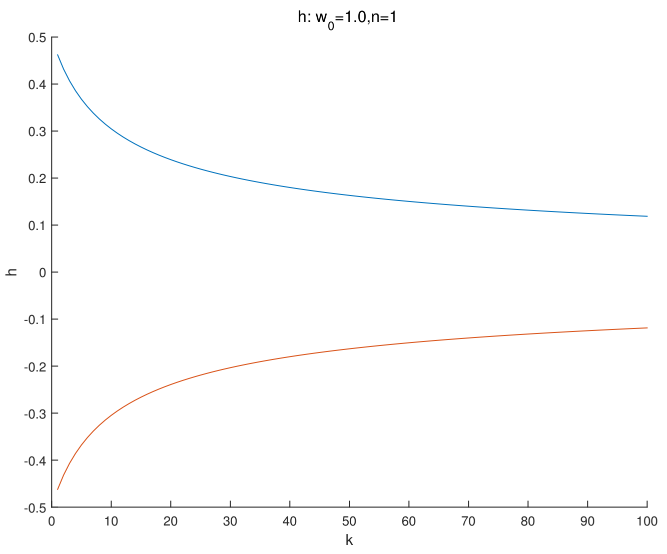
(a) *Run your code by initializing all weights as 1.0. Plot the cost J defined in page 40 of lecture notes #6 as a function of the iteration. Cost should converge to 0, i.e., training should be successful. Print out $h_{k,i}$ and $g_{k,i}$, $i = 1, 2$ obtained after just one iteration as functions of $k = 1, 2, \ldots, l$.*

Solution

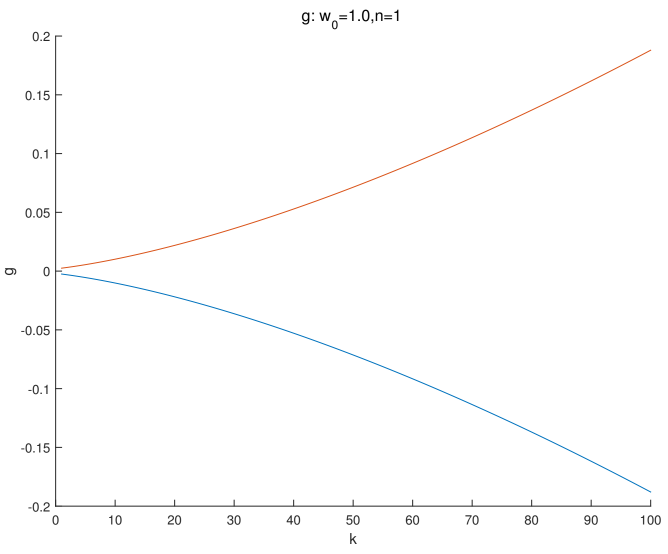
The following figure shows the cost over 1,000 iterations when all weights are initialized as 1. You can download the sample code from KLMS. The cost J starts from 0.0727 and goes down to 4.66×10^{-30} at the 1,000-th iteration, meaning that training is successful.



The following figures and tables show $h_{k,i}$ and $g_{k,i}$, $i = 1, 2$ obtained after just one iteration.



k	1	2	3	4	5	...	96	97	98	99	100
$h_{1:100,1}$	0.4621	0.4318	0.4068	0.3858	0.3677		0.1210	0.1204	0.1198	0.1193	0.1187
$h_{1:100,2}$	-0.4621	-0.4318	-0.4068	-0.3858	-0.3677		-0.1210	-0.1204	-0.1198	-0.1193	-0.1187

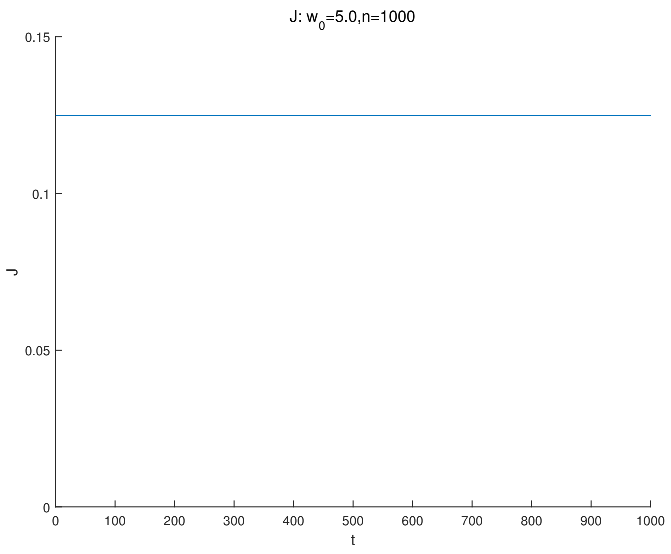


k	1	2	3	4	5	...	96	97	98	99	100
$g_{1:100,1}$	-0.0025	-0.0032	-0.0039	-0.0047	-0.0055		-0.1733	-0.1800	-0.1826	-0.1853	-0.1880
$g_{1:100,2}$	0.0025	0.0032	0.0039	0.0047	0.0055		0.1733	0.1800	0.1826	0.1853	0.1880

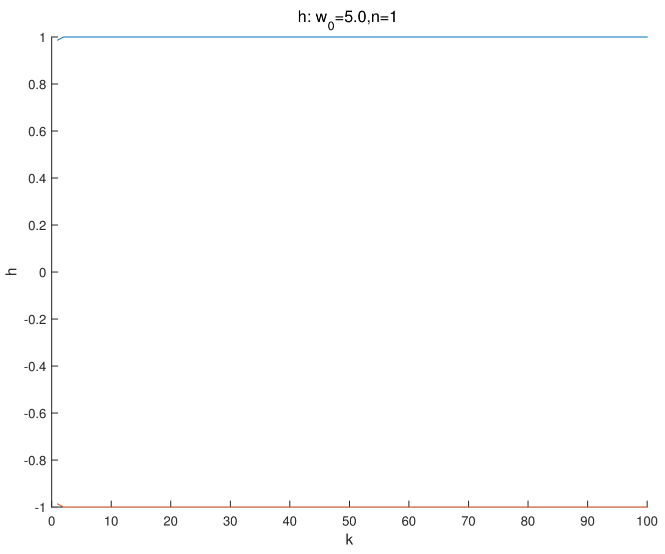
(b) *Run your code by initializing all weights as 5. This time, the cost will be stuck at about 0.125, i.e., training will fail. Print out $h_{k,i}$ and $g_{k,i}$, $i = 1, 2$ obtained after just one iteration as functions of $k = 1, 2, \ldots, l$. How are they different from $h_{k,i}$'s and $g_{k,i}$'s you obtained in (a)?*

Solution

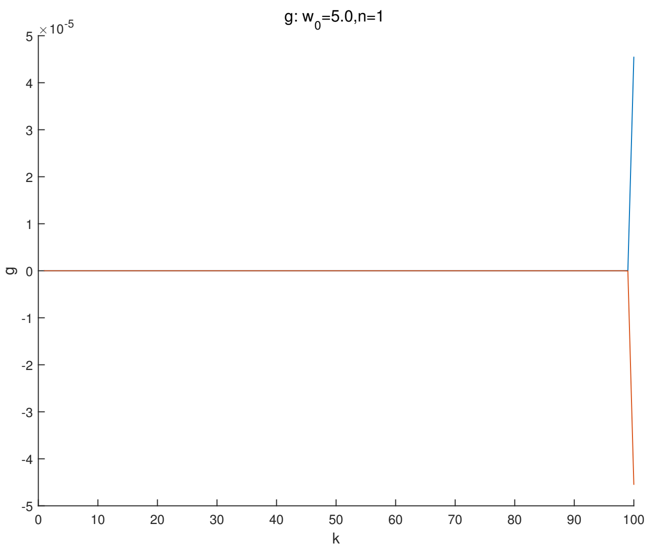
The following figure shows the cost over 1,000 iterations when all weights are initialized as 5. You can see the cost J is stuck at 0.125.



The following figures and tables show $h_{k,i}$ and $g_{k,i}$, $i = 1, 2$ obtained after just one iteration. As you can see, $h_{k,i}$'s are almost the same for all k's and $g_{k,i}$'s vanish quickly as k decreases.



k	1	2	3	4	5	...	96	97	98	99	100
$h_{1:100,1}$	0.9866	0.9999	0.9999	0.9999	0.9999		0.9999	0.9999	0.9999	0.9999	0.9999
$h_{1:100,2}$	-0.9866	-0.9999	-0.9999	-0.9999	-0.9999		-0.9999	-0.9999	-0.9999	-0.9999	-0.9999



k	1	2	3	4	5	...
$g_{1:100,1}$	5.83×10^{-304}	4.39×10^{-303}	4.23×10^{-300}	4.65×10^{-297}	5.12×10^{-294}	
$g_{1:100,2}$	-5.83×10^{-304}	-4.39×10^{-303}	-4.23×10^{-300}	-4.65×10^{-297}	-5.12×10^{-294}	

k	...	96	97	98	99	100
$g_{1:100,1}$		3.10×10^{-17}	3.41×10^{-14}	3.75×10^{-11}	4.13×10^{-8}	4.54×10^{-5}
$g_{1:100,2}$		-3.10×10^{-17}	-3.41×10^{-14}	-3.75×10^{-11}	-4.13×10^{-8}	-4.54×10^{-5}

(c) *Explain why training fails in (b). Try to see how $h_{k,i}$'s behave as k increases (forward propagation) and how the gradients $g_{k,i}$'s behave as k decreases (backward propagation). You will be able to see $g_{k,i}$'s vanish as k decreases, which is known as the vanishing gradient problem. Why does the gradient vanish as k decreases? How does the actual gradient $\nabla_{\mathbf{w}} J$ behave?*

Solution

To calculate g for deep layered network, $k = 1, \ldots, l$, we can use a recursive equation for g derived in page 41 of lecture notes #6, i.e.,

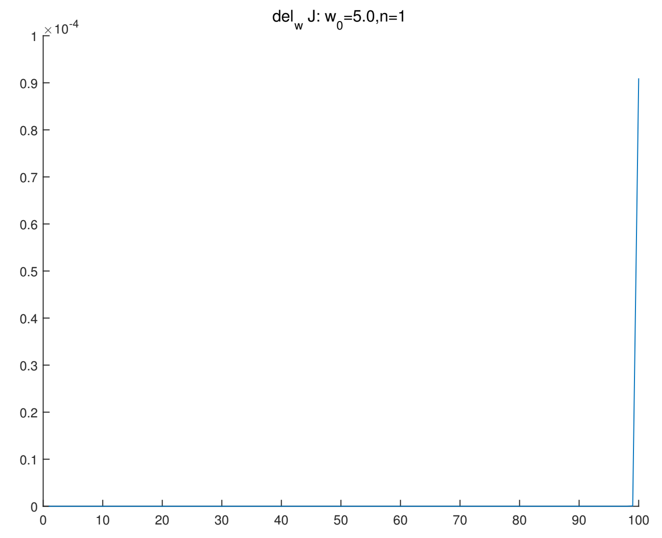
$$\mathbf{g}_{k-1} = \phi^{(k-1)\prime}(\mathbf{u}_{k-1}) \odot \mathbf{g}_k w_k,$$

Note that $\tanh'(x) = \operatorname{sech}^2(x) = \frac{4}{(e^x+e^{-x})^2} \leq 1$ for all x and is exponentially decreasing for large x. Since $u_{k,i} = h_{k-1,i} w_k$ is close to ± 5 because w's are initialized as 5 and h's are close to ± 1, we have $\phi'(u_{k,i}) = \operatorname{sech}^2(u_{k,i}) \sim 1.8 \times 10^{-4}$. Therefore, $g_{k-1,i} = \phi'(u_{k-1,i}) g_{k,i} w_k \sim 1.8 \times 10^{-4} \times 5 \times g_{k,i} \sim 9 \times 10^{-4} g_{k,i}$, which is why $\mathbf{g}_k$'s vanish so quickly as k is decreased, which is known as the vanishing gradient problem.

Let us consider the actual gradient $\nabla_{\mathbf{w}} J$, which is given as

$$\frac{\partial J}{\partial w_k} = \mathbf{h}_{k-1}^T \mathbf{g}_k, \quad k = 1, \ldots, l,$$

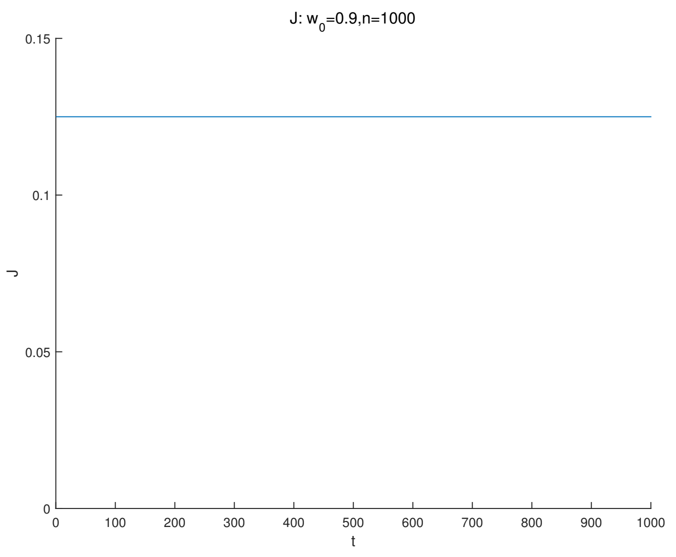
Since $h_{k,i}$ is almost constant (± 1) over k, and $g_{k,i}$ vanishes quickly as k decreases, we can expect $\nabla_{\mathbf{w}} J$ also vanishes as k decreases, which is confirmed in the following figure.



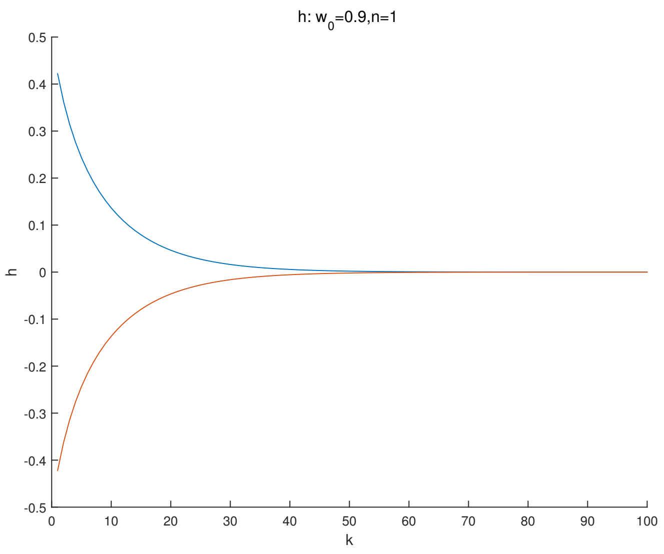
(d) *Run your code by initializing all weights as 0.9. This time, the cost will be stuck at about 0.125, i.e., training will fail. Print out $h_{k,i}$ and $g_{k,i}$, $i = 1, 2$ obtained after just one iteration as functions of $k = 1, 2, \ldots, l$. How are they different from $h_{k,i}$'s and $g_{k,i}$'s you obtained in (a) and (c)?*

Solution

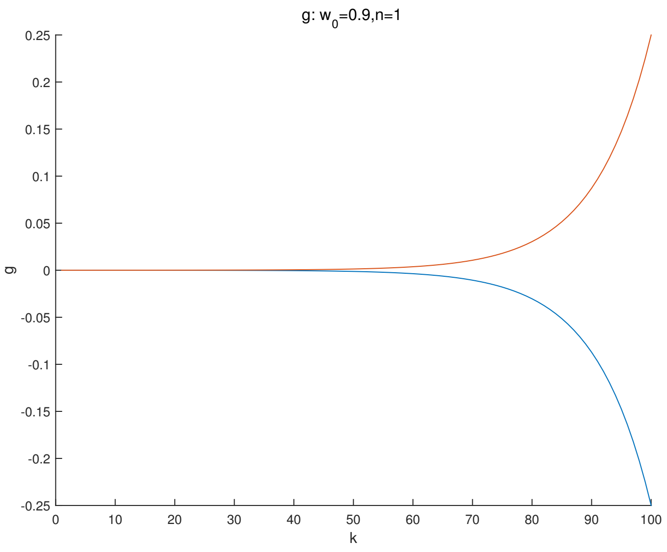
The following figure shows the cost over 1,000 iterations when all weights are initialized as 0.9. You can see the cost J is stuck at 0.125.



The following figures and tables show $h_{k,i}$ and $g_{k,i}$, $i = 1, 2$ obtained after just one iteration.



k	1	2	3	4	5	...	96	97	98	99	100
$h_{1:100,1}$	0.4219	0.3625	0.3151	0.2762	0.2436		1.54×10^{-5}	1.39×10^{-5}	1.25×10^{-5}	1.13×10^{-5}	1.01×10^{-5}
$h_{1:100,2}$	-0.4219	-0.3625	-0.3151	-0.2762	-0.2436		-1.54×10^{-5}	-1.39×10^{-5}	-1.25×10^{-5}	-1.13×10^{-5}	-1.01×10^{-5}

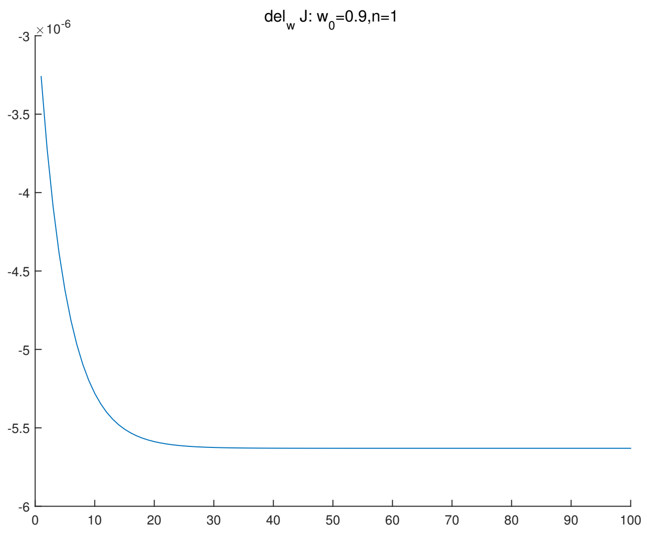


k	1	2	3	4	5	...	96	97	98	99	100
$g_{1:100,1}$	-3.26×10^{-6}	-4.41×10^{-6}	-5.64×10^{-6}	-6.95×10^{-6}	-8.37×10^{-6}		-0.1640	-0.1822	-0.2025	-0.2250	-0.2500
$g_{1:100,2}$	3.26×10^{-6}	4.41×10^{-6}	5.64×10^{-6}	6.95×10^{-6}	8.37×10^{-6}		0.1640	0.1822	0.2025	0.2250	0.2500

(e) *Explain why training fails in (d). Try to see how $h_{k,i}$'s behave as k increases (forward propagation) and how the gradients $g_{k,i}$'s behave as k decreases (backward propagation). You will be able to see $g_{k,i}$'s vanish as k decreases. Is this the only reason why training fails? How does the actual gradient $\nabla_{\mathbf{w}} J$ behave?*

Solution

The following figure shows the actual gradient $\nabla_{\mathbf{w}} J$.



Not only the values of $g_{k,i}$'s vanish as k decreases, but the values of $h_{k,i}$'s vanish as k increases. As a result the actual gradient $\frac{\partial J}{\partial w_k} = \mathbf{h}_{k-1}^T \mathbf{g}_k$ is very small for all values of k and training becomes very slow and even after 1,000 iterations the cost is still about 0.125.

When the number of layers is small, say 3, then training will be successful even when w's are initialized as 0.9. If the number of layers is increased to 100, then training fails due to vanishing $g_{k,i}$'s and $h_{k,i}$'s.

2. (Back-propagation with bias terms – 10 points)

(a) *Let's introduce a bias term in each layer in Example 1 in pages 40–42 in lecture notes #6, i.e.,* $\mathbf{h}_k = \phi^{(k)}(\mathbf{h}_{k-1} w_k + \mathbf{1} b_k)$, $k = 1, 2, \ldots, l$, *where* $\mathbf{1}$ *is an all-one vector of length* m *and* $b_k \in \mathbb{R}$ *is the bias term at the* k*-th layer. Note that you need* $\mathbf{1}$ *since there are* m *training examples. Define* $\mathbf{u}_k = \mathbf{h}_{k-1} w_k + \mathbf{1} b_k$ *and* $\mathbf{g}_k = \nabla_{\mathbf{u}_k} J$, $k = 1, 2, \ldots, l$. *Express* $\mathbf{g}_l$ *in terms of other variables such as* $\mathbf{u}_l$, $\mathbf{h}_l$, *and* $\mathbf{y}$. *Express* $\mathbf{g}_{k-1}$ *in terms of other variables including* $\mathbf{g}_k$, *i.e., backward propagation. Express* $\frac{\partial J}{\partial w_k}$, $\frac{\partial J}{\partial b_k}$, $k = 1, 2., \ldots, l$ *using* $\mathbf{h}_k$ *and* $\mathbf{g}_k$. *Show the whole training algorithm including forward and backward propagations and gradient descent as a pseudo code.*

Solution

First, we express $\mathbf{g}_l$ in terms of other variables such as $\mathbf{u}_l$, $\mathbf{h}_l$, and $\mathbf{y}$. It can be shown that $\mathbf{g}_l = \frac{1}{m}\phi^{(l)'}(\mathbf{u}_l) \odot (\mathbf{h}_l - \mathbf{y})$ and $\mathbf{g}_{k-1} = \phi^{(k-1)'}(\mathbf{u}_{k-1}) \odot \mathbf{g}_k w_k$, $k = 2, \ldots, l$ since

$$g_{l,i} = \frac{\partial J}{\partial u_{l,i}} = \frac{\partial}{\partial u_{l,i}} \frac{1}{2m} \|\mathbf{y} - \phi^{(l)}(\mathbf{u}_l)\|^2 = \frac{\partial}{\partial u_{l,i}} \frac{1}{2m} (y_i - \phi^{(l)}(u_{l,i}))^2 = \frac{1}{m}\phi^{(l)'}(u_{l,i})(\phi^{(l)}(u_{l,i}) - y_i),$$

$$g_{l-1,i} = \frac{\partial J}{\partial u_{l-1,i}} = \sum_{i'} \frac{\partial u_{l,i'}}{\partial u_{l-1,i}} \frac{\partial J}{\partial u_{l,i'}} = \phi^{(l-1)'}(u_{l-1,i}) w_l g_{l,i},$$

$$\vdots$$

$$g_{1,i} = \phi^{(1)'}(u_{1,i}) w_2 g_{2,i}.$$

Then, we get $\frac{\partial J}{\partial w_k} = \mathbf{h}_{k-1}^T \mathbf{g}_k$ and $\frac{\partial J}{\partial b_k} = \mathbf{1}^T \mathbf{g}_k$ for $k = 1, \ldots, l$ since

$$\frac{\partial J}{\partial w_k} = \sum_{i'} \frac{\partial u_{k,i'}}{\partial w_k} \frac{\partial J}{\partial u_{k,i'}} = \sum_{i'} h_{k-1,i'} g_{k,i'} = \mathbf{h}_{k-1}^T \mathbf{g}_k,$$
$$\frac{\partial J}{\partial b_k} = \sum_{i'} \frac{\partial u_{k,i'}}{\partial b_k} \frac{\partial J}{\partial u_{k,i'}} = \sum_{i'} g_{k,i'} = \mathbf{1}^T \mathbf{g}_k.$$

The whole training algorithm is given as follows.

Algorithm 1 Training with forward and backward propagations

```
 1: function TrainingAlgorithm($l$, $\alpha$, $\mathbf{x}$, $\mathbf{y}$, $MaxIter$)
 2:     initialize $w[1], w[2], \ldots, w[l], b[1], b[2], \ldots, b[l]$ randomly
 3:     $\mathbf{h}[0] \leftarrow \mathbf{x}$
 4:     for $iter = 1, 2, \ldots, MaxIter$ do
 5:         for $k = 1, 2, \ldots, l$ do                              ▷ Forward propagation
 6:             $\mathbf{u}[k] \leftarrow \mathbf{h}[k-1] * w[k] + \mathbf{1} * b[k]$
 7:             $\mathbf{h}[k] \leftarrow \phi^{(k)}(\mathbf{u}[k])$
 8:         $\mathbf{g}[l] \leftarrow \phi^{(l)\prime}(\mathbf{u}[l]) * (\mathbf{h}[l] - \mathbf{y})/m$
 9:         for $k = l-1, l-2, \ldots, 1$ do                           ▷ Backward propagation
10:             $\mathbf{g}[k] \leftarrow \phi^{(k)\prime}(\mathbf{u}[k]) * \mathbf{g}[k+1] * w[k+1]$
11:         for $k = 1, 2, \ldots, l$ do                              ▷ Gradient descent
12:             $w[k] \leftarrow w[k] - \alpha \mathbf{h}[k-1]^\mathsf{T} * \mathbf{g}[k]$
13:             $b[k] \leftarrow b[k] - \alpha \mathbf{1}^\mathsf{T} * \mathbf{g}[k]$
14:     return $w[1], w[2], \ldots, w[l], b[1], b[2], \ldots, b[l]$
```

(b) *In Example 2 in pages 43–46 in lecture notes #6, show that* $\mathbf{G}^{(2)} = -\frac{1}{m}\mathbf{Y}^T$*, where* $\mathbf{Y}_{j,i} = 1$ *if* $j = y_i$ *and 0 otherwise, which was defined in page 44 of lecture notes #6.*

Solution

First, we prove that

$$\nabla_A \mathrm{tr}(AB) = B^T.$$

$\mathrm{tr}(AB)$ is given by

$$\mathrm{tr}(AB) = \sum_i \sum_j a_{ij} b_{ji}. \tag{1}$$

If we take the partial derivative of $\mathrm{tr}(AB)$ with respect to a_{ij}, we get b_{ji}. Thus, $\nabla_A \mathrm{tr}(AB) =$

B^T. Next, we prove $\mathbf{G}^{(2)} = -\frac{1}{m}\mathbf{Y}^T$. Note that J_{MLE} is given by

$$\begin{aligned}
J_{\text{MLE}} &= -\frac{1}{m}\sum_{i=1}^{m} \log p_{\text{model}}(y_i|\mathbf{X}_{i,\cdot}; \mathbf{W}^{(1)}, \mathbf{W}^{(2)}) \\
&= -\frac{1}{m}\sum_{i=1}^{m} \log \exp(\phi(\mathbf{X}_{i,\cdot}\mathbf{W}^{(1)})\mathbf{W}^{(2)}_{\cdot,y_i}) \\
&= -\frac{1}{m}\sum_{i=1}^{m} \phi(\mathbf{X}_{i,\cdot}\mathbf{W}^{(1)})\mathbf{W}^{(2)}_{\cdot,y_i} \\
&= -\frac{1}{m}\sum_{i=1}^{m} \mathbf{H}_{i,\cdot}\mathbf{W}^{(2)}_{\cdot,y_i} \\
&= -\frac{1}{m}\sum_{i=1}^{m} \mathbf{H}_{i,\cdot}\mathbf{W}^{(2)}\mathbf{Y}_{\cdot,i} \\
&= -\frac{1}{m}\text{tr}(\mathbf{H}\mathbf{W}^{(2)}\mathbf{Y}) \\
&= -\frac{1}{m}\text{tr}(\mathbf{U}^{(2)}\mathbf{Y}), \qquad (2)
\end{aligned}$$

where $\mathbf{Y}_{j,i} = 1$ if $y_i = j$ and 0 otherwise, $\mathbf{U}^{(2)}$ is defined as $\mathbf{H}\mathbf{W}^{(2)}$. Combining (1), (2) and the derivation of $\mathbf{G}^{(2)} = \nabla_{U^{(2)}} J_{\text{MLE}}$ in lecture notes #6, we get $\mathbf{G}^{(2)} = -\frac{1}{m}\mathbf{Y}^T$.

(c) *Let's introduce a bias term in the first layer in Example 2 in pages 43–46 in lecture notes #6, i.e.,* $\mathbf{H} = \phi(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{1}\mathbf{b}^T)$*, where* $\mathbf{1}$ *is an all-one vector of length* m *and* $\mathbf{b}$ *is the bias vector of length* n_1*. Note that you need* $\mathbf{1}$ *since there are* m *training examples. Define* $\mathbf{U}^{(1)} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{1}\mathbf{b}^T$ *and* $\mathbf{G}^{(1)} = \nabla_{\mathbf{U}^{(1)}} J_{\text{MLE}}$*. Assume the other variables such as* $\mathbf{U}^{(2)}$ *and* $\mathbf{G}^{(2)}$ *are defined the same way as done in Example 2. Express* $\mathbf{G}^{(1)}$ *in terms of other variables such as* $\mathbf{U}^{(1)}$*,* $\mathbf{G}^{(2)}$*, and* $\mathbf{W}^{(2)}$*. Express* $\nabla_{\mathbf{W}^{(1)}} J_{\text{MLE}}$ *and* $\nabla_{\mathbf{b}} J_{\text{MLE}}$ *using other variables such as* $\mathbf{X}$ *and* $\mathbf{G}^{(1)}$*.*

Solution

Using $\mathbf{U}^{(2)} = \phi(\mathbf{U}^{(1)})\mathbf{W}^{(2)}$, we get $\mathbf{G}^{(1)} = \phi'(\mathbf{U}^{(1)}) \odot (\mathbf{G}^{(2)}(\mathbf{W}^{(2)})^T)$ since

$$\begin{aligned}
G^{(1)}_{i,j} &= (\nabla_{\mathbf{U}^{(1)}} J_{\text{MLE}})|_{i,j} \\
&= \frac{\partial J_{\text{MLE}}}{\partial U^{(1)}_{i,j}} \\
&= \sum_{i'}\sum_{j'} \frac{\partial U^{(2)}_{i',j'}}{\partial U^{(1)}_{i,j}} \frac{\partial J_{\text{MLE}}}{\partial U^{(2)}_{i',j'}} \\
&= \sum_{i'}\sum_{j'} \frac{\partial \sum_k \phi(U^{(1)}_{i',k})W^{(2)}_{k,j'}}{\partial U^{(1)}_{i,j}} \frac{\partial J_{\text{MLE}}}{\partial U^{(2)}_{i',j'}} \\
&= \sum_{i'}\sum_{j'}\sum_{k} \delta_{i,i'}\delta_{j,k}\phi'(U^{(1)}_{i',k})W^{(2)}_{k,j'} \frac{\partial J_{\text{MLE}}}{\partial U^{(2)}_{i',j'}} \\
&= \sum_{j'} \phi'(U^{(1)}_{i,j}) \frac{\partial J_{\text{MLE}}}{\partial U^{(2)}_{i,j'}} W^{(2)}_{j,j'} \\
&= \phi'(\mathbf{U}^{(1)}) \odot (\mathbf{G}^{(2)}(\mathbf{W}^{(2)})^T)|_{i,j}.
\end{aligned}$$

$\nabla_{\mathbf{W}^{(1)}} J_{\text{MLE}} = \mathbf{X}^T \mathbf{G}^{(1)}$ since

$$\begin{aligned}
\nabla_{\mathbf{W}^{(1)}} J_{\text{MLE}}|_{i,j} &= \frac{\partial J_{\text{MLE}}}{\partial W^{(1)}_{i,j}} \\
&= \sum_{i'} \sum_{j'} \frac{\partial U^{(1)}_{i',j'}}{\partial W^{(1)}_{i,j}} \frac{\partial J_{\text{MLE}}}{\partial U^{(1)}_{i',j'}} \\
&= \sum_{i'} \sum_{j'} \frac{\partial \sum_k X_{i',k} W^{(1)}_{k,j'} + b_{j'}}{\partial W^{(1)}_{i,j}} \frac{\partial J_{\text{MLE}}}{\partial U^{(1)}_{i',j'}} \\
&= \sum_{i'} \sum_{j'} \sum_{k} X_{i',k} \delta_{i,k} \delta_{j,j'} \frac{\partial J_{\text{MLE}}}{\partial U^{(1)}_{i',j'}} \\
&= \sum_{i'} X_{i',i} \frac{\partial J_{\text{MLE}}}{\partial U^{(1)}_{i',j}} \\
&= \mathbf{X}^T \mathbf{G}^{(1)}|_{i,j}.
\end{aligned}$$

Also, $\nabla_{\mathbf{b}} J_{\text{MLE}} = \mathbf{1}^T \mathbf{G}^{(1)}$ since

$$\begin{aligned}
\nabla_{\mathbf{b}} J_{\text{MLE}}|_{j} &= \frac{\partial J_{\text{MLE}}}{\partial b_j} \\
&= \sum_{i'} \sum_{j'} \frac{\partial U^{(1)}_{i',j'}}{\partial b_j} \frac{\partial J_{\text{MLE}}}{\partial U^{(1)}_{i',j'}} \\
&= \sum_{i'} \sum_{j'} \frac{\partial \sum_k X_{i',k} W^{(1)}_{k,j'} + b_{j'}}{\partial b_j} \frac{\partial J_{\text{MLE}}}{\partial U^{(1)}_{i',j'}} \\
&= \sum_{i'} \sum_{j'} \delta_{j,j'} \frac{\partial J_{\text{MLE}}}{\partial U^{(1)}_{i',j'}} \\
&= \sum_{i'} \frac{\partial J_{\text{MLE}}}{\partial U^{(1)}_{i',j}} \\
&= \mathbf{1}^T \mathbf{G}^{(1)}|_{j}.
\end{aligned}$$