AssignmentReport-Group187

February 20, 2024

1 Assignment 1 Report

This is an outline for your report to ease the amount of work required to create your report. Jupyter notebook supports markdown, and I recommend you to check out this cheat sheet. If you are not familiar with markdown.

Before delivery, **remember to convert this file to PDF**. You can do it in two ways: 1. Print the webpage (ctrl+P or cmd+P) 2. Export with latex. This is somewhat more difficult, but you'll get somehwat of a "prettier" PDF. Go to File -> Download as -> PDF via LaTeX. You might have to install nbconvert and pandoc through conda; `conda install nbconvert pandoc`.

2 Task 1

2.1 task 1a)

Fill in task 1a image of hand-written notes which are easy to read, or latex equations here

Task 1a)

Input → Hidden layer → eq 2: $w_{j'i'} := w_{j'i'} - \alpha \boxed{\frac{\partial C}{\partial w_{j'i'}}}$

Chain-rule

$$\frac{\partial C}{\partial w_{j'i'}} = \frac{\partial C}{\partial a_{j'}} \underbrace{\frac{\partial a_{j'}}{\partial z_{j'}}}_{\text{with } a_{j'} = f(z_{j'})} \underbrace{\frac{\partial z_{j'}}{\partial w_{j'i'}}}_{\text{using } z_{j'} = \sum_i w_{j'i'} x_i :}$$

Sigmoid function: $\frac{\partial z_{j'}}{\partial w_{j'i'}} = x_i$

$$\frac{\partial a_{j'}}{\partial z_{j'}} = f'(z_{j'}) = f(z_{j'}) \cdot (1 - f(z_{j'}))$$

$$\frac{d}{dz}(f(z)) = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right) = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1+e^{-z}-1}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}} - \frac{1}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right) = f(z)\cdot(1-f(z))$$

$$\frac{\partial C}{\partial w_{j'i'}} = \underbrace{\frac{\partial C}{\partial a_{j'}}}_{\text{backward-prop! (Output→hidden)}} \cdot f'(z_{j'}) \cdot x_i$$

$$\frac{\partial C}{\partial a_j} = \sum_k \underbrace{\frac{\partial C}{\partial z_k}}_{= \delta_k = -(y_k - \hat{y}_k)} \underbrace{\frac{\partial z_k}{\partial a_j}}_{= w_{kj'} \text{ since } z_k = \sum_j w_{kj'} a_{j'}} = \sum_k \delta_k w_{kj'}$$

$$\frac{\partial C}{\partial w_{j'i'}} = \underbrace{\sum_k w_{kj'} \delta_k \cdot f'(z_{j'})}_{= \delta_{j'}} \cdot x_i$$

$= \delta_{j'}$, q.e.d.

$$\frac{\partial C}{\partial w_{j'i'}} = \delta_{j'} x_{i'}$$

Into eq. 2:

$$w_{j'i'} := w_{j'i'} - \alpha \frac{\partial C}{\partial w_{j'i'}} = \underbrace{w_{j'i'} - \alpha \delta_{j'} x_i}_{= \text{eq. 3}}$$

$=$ eq. 3, q.e.d.

2.2 task 1b)

Task 1b)

Output → Hidden layer:

$$w_{kj} := w_{kj} - \alpha \underbrace{\frac{\partial C}{\partial w_j}}_{\delta_k a_j = \frac{\partial C}{\partial z_k} a_j = -(y_k - \hat{y}_k)\, a_j}$$

$$\rightarrow w_{kj} := w_{kj} - \alpha\, (\hat{y}_k - y_k)\, a_j^T$$

→ Vectorized:

$$\underbrace{W^{(2)}}_{(K\times J)} := W^{(2)} - \alpha\, \overbrace{\overbrace{\underbrace{(\hat{Y}-Y)}_{(K\times \underset{N}{1})}}^{\delta^{(2)}}\cdot \underbrace{A^T}_{(J\times \underset{N}{1})}}^{(K\times J)}$$

+ averaged over all samples / batch (N)

Hidden layer → Input layer:

$$w_{ji} := w_{ji} - \alpha\, \delta_j x_i \quad , \quad \delta_j = f'(z_j)\cdot \sum_k w_{kj}\cdot \overbrace{\delta_k}^{-(y_k - \hat{y}_k)}$$

$$\underbrace{w_{ji}}_{(J\times I)} := w_{ji} - \alpha\, \underbrace{f'(z_j)}_{(J\times 1)} \sum_k \underbrace{w_{kj}}_{(K\times J)}\cdot \underbrace{(\hat{y}_k - y_k)}_{(K\times 1)}\cdot \underbrace{x_i^T}_{(1\times I)}$$

$$w_{kj}^T\cdot(\hat{y}_k - y_k) = (J\times 1)$$

$$f'(z_j) \odot (w_{kj}^T\cdot(\hat{y}_k - y_k)) = (J\times 1)$$

sigmoid to every element

element wise Multiply

$$=(J\times I)$$

$$\rightarrow \underbrace{W^{(1)}}_{(J\times I)} := W^{(1)} - \alpha\, \overbrace{\underbrace{f'(z)}_{(J\times \underset{N}{1})} \odot \left(\underbrace{W^{(2)}}_{(J\times K)}\cdot \underbrace{(\hat{Y}-Y)}_{(K\times \underset{N}{1})}\right)}^{=\delta^{(1)} = (J\times \overset{N}{1})}\cdot \underbrace{X^T}_{(\underset{N}{1}\times I)}$$

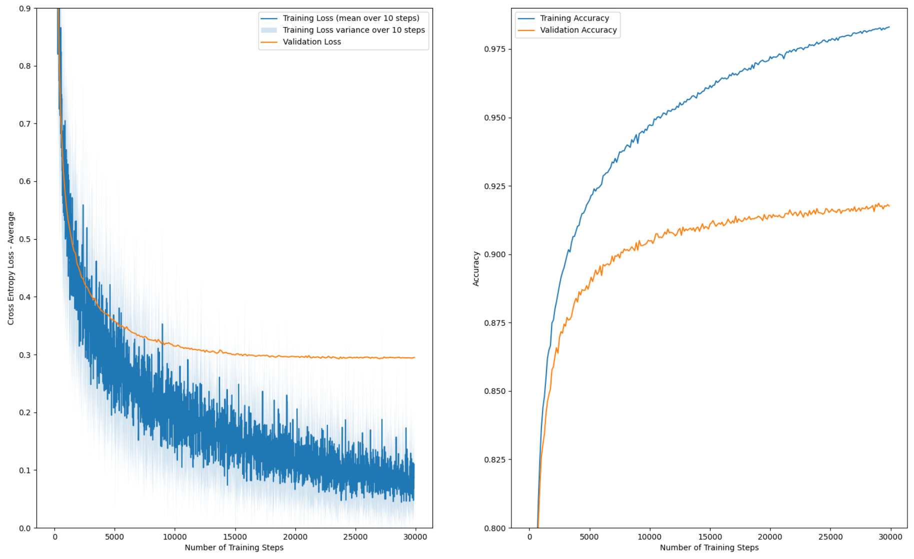
$$\text{Short:}\begin{cases} \underbrace{W^{(2)}}_{(K\times J)} := W^{(2)} - \alpha\, \underbrace{\delta^{(2)}}_{(K\times \underset{N}{1})} \underbrace{A^T}_{(\underset{N}{1}\times J)} & , \delta^{(2)} = -(\underbrace{Y}_{(K\times \underset{N}{1})} - \underbrace{\hat{Y}}_{(K\times \underset{N}{1})}) \\ \underbrace{W^{(1)}}_{(J\times I)} := W^{(1)} - \alpha\, \underbrace{\delta^{(1)}}_{(J\times \underset{N}{1})} \underbrace{X^T}_{(\underset{N}{1}\times I)} & , \delta^{(1)} = \underbrace{f'(z)}_{(J\times \underset{N}{1})} \odot (\underbrace{W^{(2)T}}_{(J\times K)}\cdot \underbrace{\delta^{(2)}}_{(K\times \underset{N}{1})}) \end{cases}$$

3 Task 2

4 Task 2a)

4.0.1 mean: 33.55, std: 78.88 (rounded values)

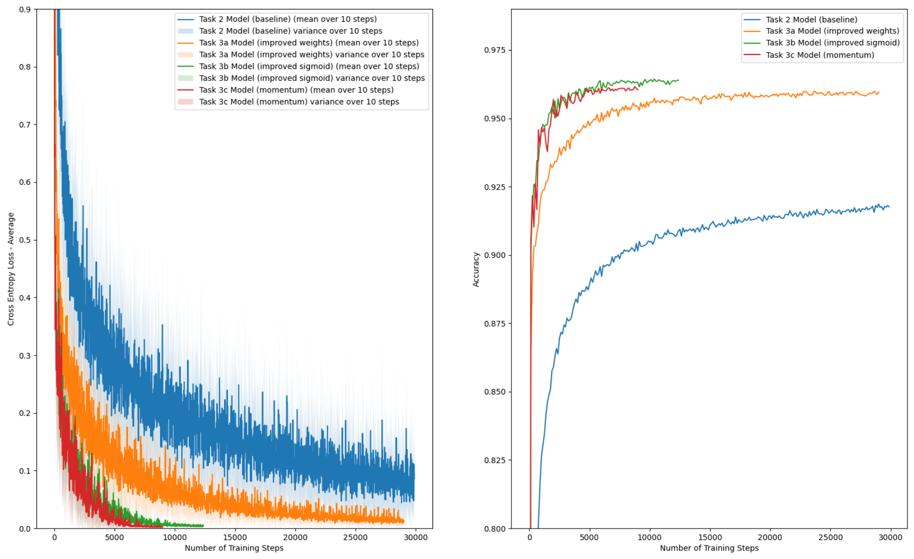
4.1 Task 2c)



4.2 Task 2d)

Total number of parameters (weights + biases) = 784 * 64 + 64 + 64 * 10 = 50880

5 Task 3



Each addition led to faster convergence speed and also generally a lower loss value. The only addition that stood out in an opposite way was momentum: it improved the convergence speed and loss, albeit on the cost of accuracy. On three models the accuracy is pretty high, this could indicate overfitting and hence lower generalization performance.

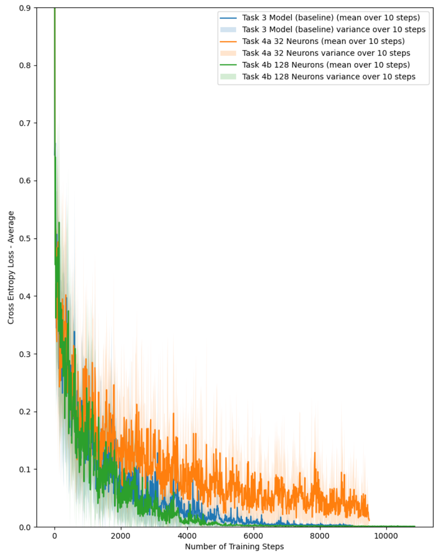
6 Task 4

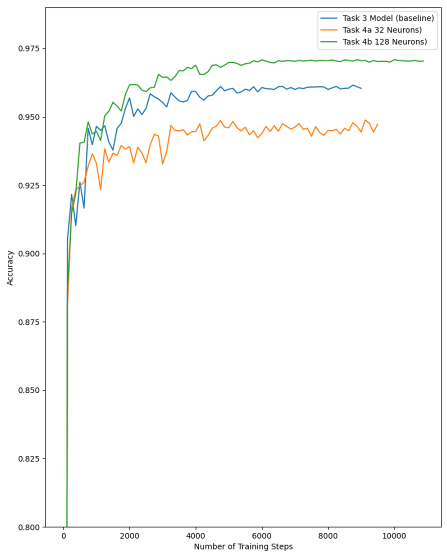
6.1 Task 4a)

Setting the number of hidden units to 32 leads to a lower accuracy, slower convergence speed and a higher loss value.

6.2 Task 4b)

Setting the number of hidden units to 128 on the other hand leads to a very high accuracy, which could even intigate overfitting. However, it improves the loss value at the start of the training, but then performs worse than the baseline with 64 hidden units.





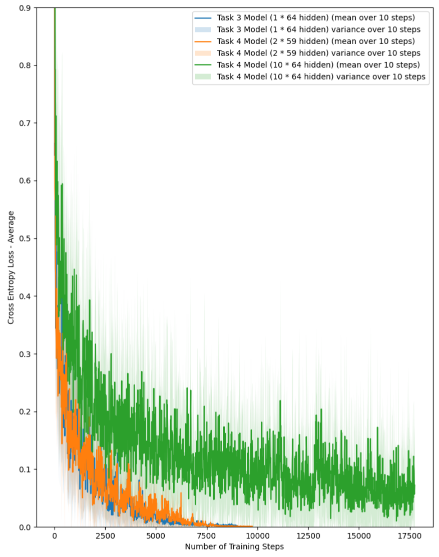
6.3 Task 4d

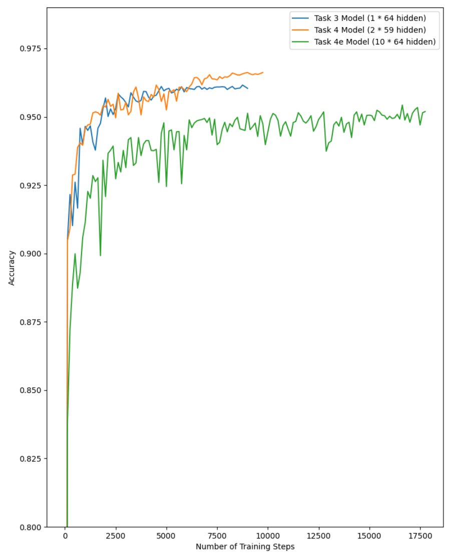
Task 3 model:

- Inputs: 785 neurons (including bias)
- Hidden layer 1: 64 neurons
- Output layer: 10 neurons
- total number of parameters: 785 * 64 (number of connections from input to hidden layer, including bias) + 64*10 (number of connections from hidden layer to output layer) = **50880**

New model used:

- Inputs: 785 neurons
- Hidden layer 1: 59 neurons
- Hidden layer 2: 59 neurons
- Output layer: 10 neurons
- total number of parameters: 785 * 59 (number of connections from input to hidden layer 1, including bias) + 59 * 59 (number of connections from hidden layer 1 to hidden layer 2) + 59*10 (number of connections from hidden layer to output layer) = **50386**





The network with more layers has a similar convergence speed as the baseline from task 3. However, it has a better accuracy.

6.4 Task 4e)

The model has a worse convergence speed, loss and accuracy. This effect is called the problem of vanishing gradients, because the network has too many layers ("is too deep") the gradients get decrease over time and vanish which worsens the performance.