Neural Networks & Representation Learning (CS-587)
Assignment #2

Due on April 30, 2023

Ass. Prof. N. Komontakis

University of Crete
Department of Computer Science

Nikolaos Kougioulis (ID 1285)

April 30, 2023

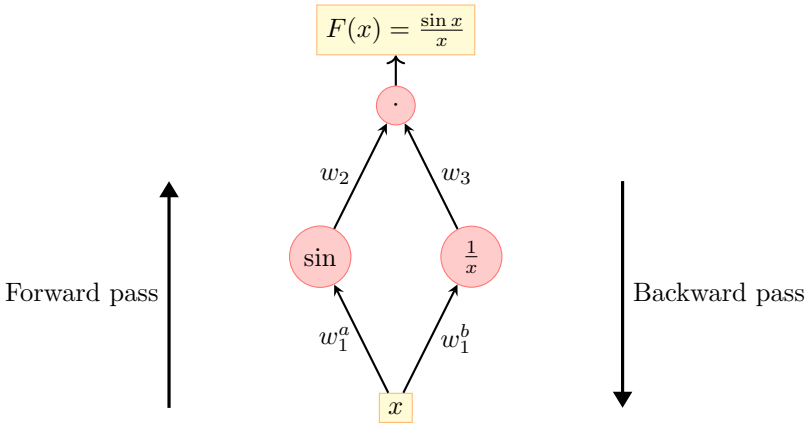


Figure 1: Computational graph of the function $F(x) = \dfrac{\sin x}{x}$. After decomposing the function during the forward pass, backpropagation computes the derivative values using the chain rule on the successors of each node in the DAG.

Assignment 2a

The task of part A of Assignment 2 is to get started with the TensorFlow library and its components (more precisely, version 1.15 of TensorFlow on Python 3.6) and compute the gradient using forward and backward propagation.

- **Question:** What is the functionality of upstream? Answer it in your report.

  The keyword *upstream* in the TensorFlow library (up+stream) refers to the flow of data and operations in the computational graph of the neural network, in the way that the output of a layer is the input of the next layer etc. This concept allows us to describe the flow of data in a neural network model without specifically referring to a particular TensorFlow module [1].

- **Question:** For the one-variable function $y(x) = \frac{\sin x}{x}$: i) design the computational graph ii) compute the derivative for the backpropagation process by hand

  i) The computational graph for the function $\frac{\sin x}{x}$ is shown in Figure 1.

  ii) During the forward pass, we have

$$F(x) \equiv \sin x \cdot \frac{1}{x} \equiv \sin w_1 \cdot \frac{1}{w_1} \equiv w_2 \cdot w_3 \equiv w_4$$

  During the backward pass, we obtain using the chain rule

$$\overline{F} = \overline{w_4} = 1 \text{ (seed)}$$

$$\overline{w}_3 = \frac{\partial F}{\partial w_3} = \frac{\partial F}{\partial w_4}\frac{\partial w_4}{\partial w_3} = \overline{w}_4 \cdot w_2$$

[1] https://www.tensorflow.org/api_docs/python/tf/custom_gradient

$$\overline{w}_2 = \frac{\partial F}{\partial w_2} = \frac{\partial F}{\partial w_4}\frac{\partial w_4}{\partial w_2} = \overline{w}_4 \cdot w_3$$

$$\overline{w_1}^a = \overline{w_2} \cdot \cos w_1$$

$$\overline{w_1}^b = \overline{w_3} \cdot \left(-\frac{1}{w_1^2}\right)$$

$$\overline{x} = \frac{\partial F}{\partial x} = \overline{w_2}\cos x + \overline{w_3} \cdot \left(-\frac{1}{x^2}\right) = \overline{w_4} w_3 \cos x - \overline{w_4} w_2 \frac{1}{x^2} = 1 \cdot \frac{1}{x}\cos x - \sin x \cdot \frac{1}{x^2}$$

That is,

$$\nabla_x F = \frac{\partial F}{\partial x} = \frac{1}{x}\cos x - \sin x \frac{1}{x^2}$$

which is the same as computing the derivative analytically.

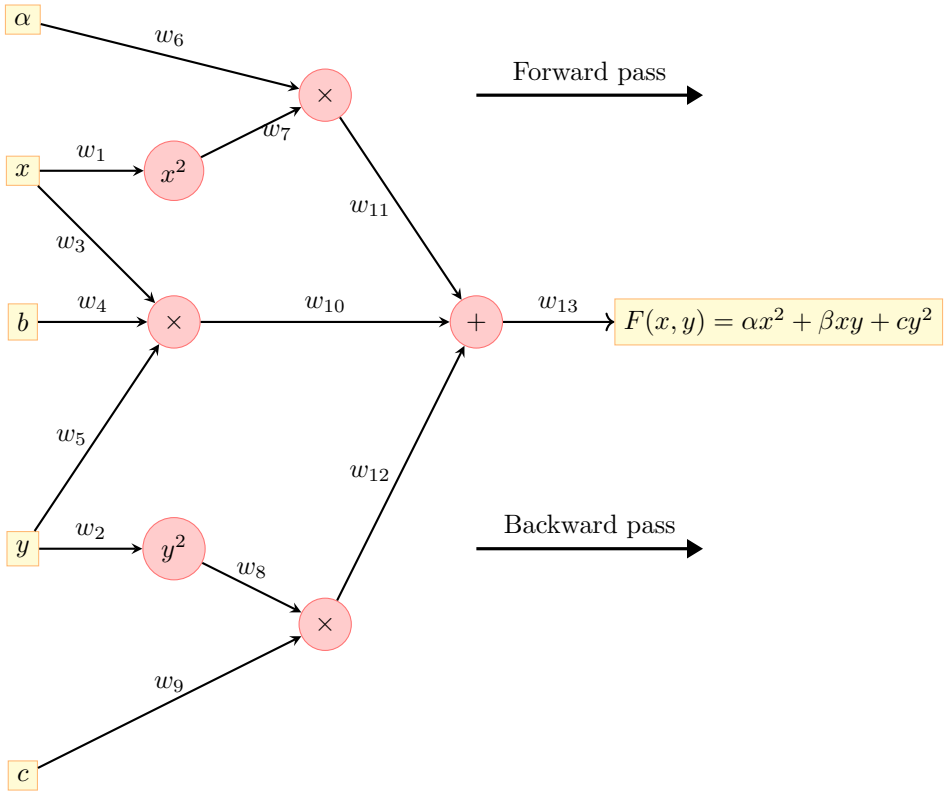


Figure 2: Computational graph of the function $f(x, y) = ax^2 + bxy + cy^2$.

- **Question:** For the two-variable function $f(x, y) = ax^2 + bxy + cy^2$: i) design the computational graph by hand - also include the partial derivatives $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$.

  The computational graph is shown in Figure 2. For the partial derivatives $\frac{\partial F}{\partial x}$, $\frac{\partial F}{\partial y}$, using the chain rule, we need to compute (no computation is asked by the exercise - just including the formulas)

$$\frac{\partial F}{\partial x} = \frac{\partial F}{\partial w_{13}} \frac{\partial w_{13}}{\partial w_{11}} \frac{\partial w_{11}}{\partial w_7} \frac{\partial w_7}{\partial w_1} \frac{\partial w_1}{\partial x} + \frac{\partial F}{\partial w_{13}} \frac{\partial w_{13}}{\partial w_{10}} \frac{\partial w_{10}}{\partial w_3} \frac{\partial w_3}{\partial x}$$

$$\frac{\partial F}{\partial y} = \frac{\partial F}{\partial w_{13}} \frac{\partial w_{13}}{\partial w_{10}} \frac{\partial w_{10}}{\partial w_5} \frac{\partial w_5}{\partial w_y} + \frac{\partial F}{\partial w_{12}} \frac{\partial w_{12}}{\partial w_8} \frac{\partial w_8}{\partial w_2} \frac{\partial w_2}{\partial y}$$

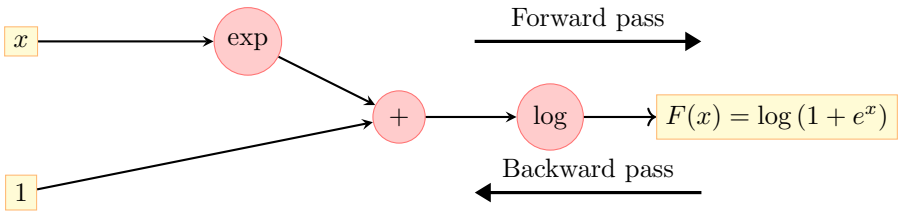


Figure 3: Computational graph of the function $F(x) = \log(1 + e^x)$

- **Question:** For the function $F(x) = \log(1 + e^x)$, design the computational graph (include it in your report).

  The computational graph is shown in Figure 3.

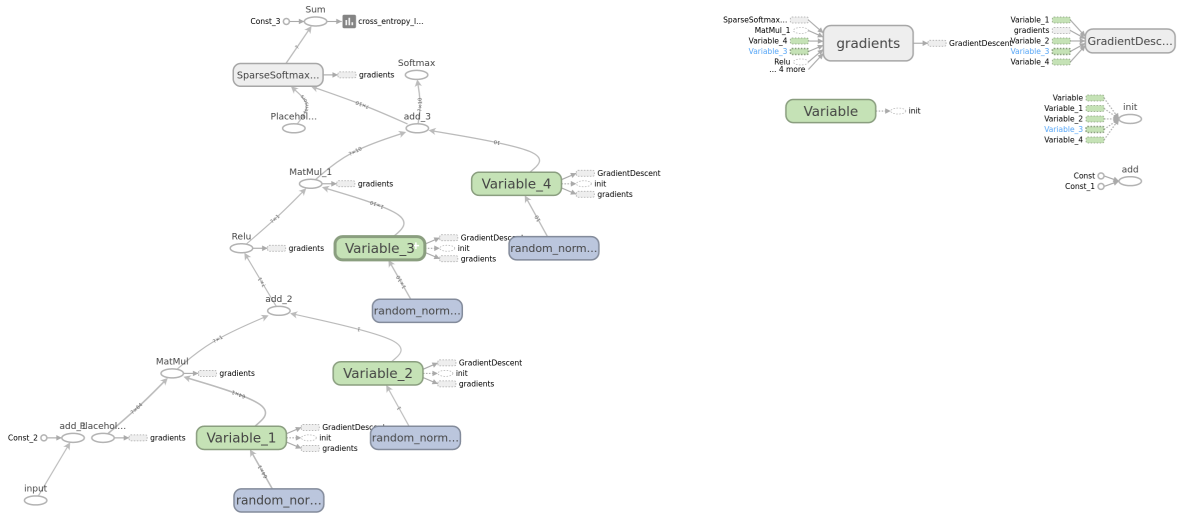


Figure 4: Computational Graph of NN.

Assignment 2b

The task in this part of the assignment is to perform supervised learning using neural networks (on image recognition) using the `load_digits` dataset, in TensorFlow and Tensorboard.

The dataset contains 1,797 samples of 8x8 images of handwritten digits, where and each pixel in the image is represented by a grayscale value ranging from 0 to 16. With a total of output variables (target) being 10 (numbers 0,1,...,9).

We begin as asked by creating a one hidden layer model, with default parameters `batch_size = 32`, `hid_size = 1`, `learning_rate = 0.01`, `num_epochs = 10` and the Relu activation function, with no regularization term. The resulting computational graph, generated using Tensorboard, is shown in Figure 4. By default, the TensorBoard library shows the op-level graph, with data flowing from bottom to top, so it is upside down compared to the code. It also allows us to view not only the main graph but the auxiliary nodes as well (right part of the figure).

The cross entropy loss over the number of iterations is shown in Figure 5, tracked during training using Tensorboard. Notice how initially there is a sudden drop in the loss and after a number of iterations it begins oscillating around a fixed point, where no increment of iterations and number of epochs can further lower the loss of the model.

For Part 3 of the assignment, we attempt to fine-tune the hyperparameters of our model in order to maximize the test accuracy, while also adding a regularization term. Due to the high number of combinations between each hyperparameter (batch size, hidden layer size, learning rate, number of epochs and regularization strength) we have opted to fix the activation function instead of inserting it as a hyperparameter as well.

We perform hyperparameter tuning using random walk search, that is, out of 160 randomly selected parameter configurations select the one that maximizes our test accuracy. Of the RELu, tanh and sigmoid activation functions, the first two performed slightly worse, hence we opted to only use the sigmoid activation function.

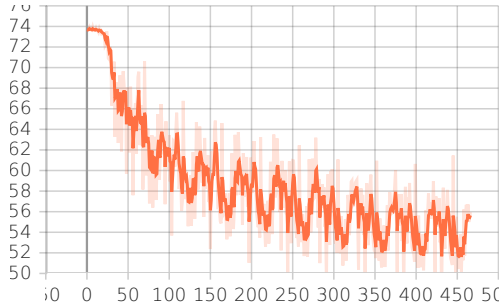
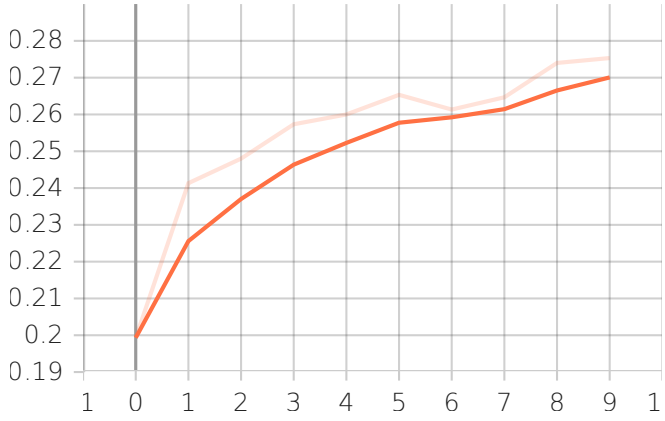
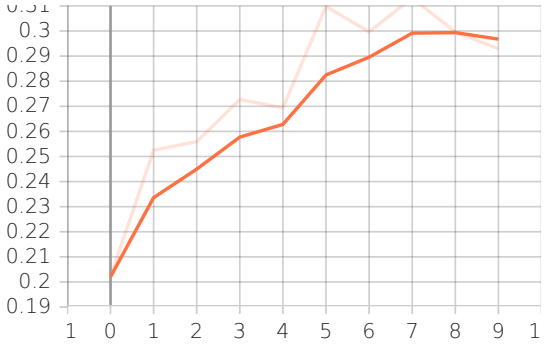


Figure 5: Plot of the cross-entropy loss of the model in Part 2. Bold colored plot is shown with a smoothing value of 0.6.



(a) Train accuracy, exported from Tensorboard of the model in Part 2. Bold colored plot is shown with a smoothing value of 0.6.



(b) Test accuracy, exported from Tensorboard of the model in Part 2. Bold colored plot is shown with a smoothing value of 0.6.

Figure 6: Figures

Figure 7: Plot of the cross-entropy loss of the fine-tuned model, exported from Tensorboard. Bold colored plot is shown with a smoothing value of 0.6.

After selecting and evaluating the random configurations, our optimal yields a test set accuracy of 0.9259 using the following parameters:

- activation function: sigmoid
- batch size: 64
- hidden size: 30
- best learning rate: 0.05
- best regularization strength: 0.001
- best num epochs: 20

While training, we keep monitoring the cross-entropy loss of the fine-tuned model in Tensorboard. Notice the spike at the first iterations and then a decrement close to zero during the training process. This is because at the beginning of training, the parameters of the model are randomly initialized and it is more likely to make random predictions.

References

[1] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.