Homework #2 – Neural Network Overview – Activation Functions and Back Propagation
CAP 5619, Deep & Reinforcement Learning (Spring 2022), Department of Computer Science, Florida State University

---

Points: 80
Due: Beginning of the class (6:35pm) on Monday, February 14, 2022

Submission: You need to submit electronically via Canvas by uploading a) a pdf file (named "**hw2-Firstname-Lastname.pdf**") for your answers to the questions, and b) the program(s) you have created (for Problems 2, 3, 4, and other ones) (named as "**hw2-prog-Firstname-Lastname.???**"); if there are multiple program files, please zip them as a single archive. Here replace "Firstname" using your first name and replace "Lastname" using your last name in the file names.

The main purpose of this assignment is to let you be familiar with neural network architecture elements including activation functions, and the basic back propagation algorithm.

Problem 1 (20 points) As neural networks are typically trained using (stochastic) gradient descent optimization algorithms, properties of the activation functions affect the learning. Here we divide the domain of an activation function into: 1) fast learning region if the magnitude of the gradient is larger than 0.99, 2) active learning region if the magnitude of the gradient is between 0.01 and 0.99 (inclusive), 3) slow learning region if the magnitude of the gradient is larger than 0 but smaller than 0.01, and 4) inactive learning region if the magnitude of the gradient is 0. For each of the following activation functions, **plot** its gradient in the range from -5 to 5 of the input and then **list** the four types of regions. If the gradient is not well defined for an input value, indicate so and then use any reasonable value.

(1) Rectified linear unit $f(z) = \begin{cases} z & if\ z \geq 0 \\ 0 & otherwise \end{cases}$.

(2) Logistic sigmoid activation function $f(z) = \sigma(z) = \frac{1}{1+e^{-z}}$.

(3) Piece-wise linear unit $f(z) = \begin{cases} 0.05z + 0.95 & if\ z > 1 \\ z & if\ 1 \geq z \geq -1 \\ 0.05z - 0.95 & Otherwise \end{cases}$.

(4) Swish $f(z) = z\sigma(3z)$, where $\sigma(z) = \frac{1}{1+e^{-z}}$.

(5) Exponential Linear Unit (ELU) $f(z) = \begin{cases} z & if\ z \geq 0 \\ 0.1(e^z - 1) & otherwise \end{cases}$. (Note here is a special case of the general ELU function with α=0.1.)

Problem 2 (20 points) Using the framework you have established, design and train a fully connected ReLU neural network to learn how to approximate

$$g(x) = \sin\left(\frac{\pi}{4}x\right).$$

Here all the activation functions in the neural network are ReLU except the output layer, where the linear function is used. Note that your neural network should have as few as possible neurons in the hidden layer(s) and the largest error (i.e., the absolute difference between g(x) and the output of your neural network) should be no more than 0.075 in the range of inputs from -2 to 2. You can generate and use no more than 200 training samples.

(a) Plot the training error with respect to the epoch along with the average and maximal error of the network at the end of the epoch for the inputs from -2 to 2.

(b) Note that nonlinearity is essential for (deep) neural networks as one with no nonlinearity is reduced to a one-layer neural network. The nonlinearity of every ReLU neural network can be characterized using activation patterns, along with the activation regions. For any valid input, the activation pattern of a ReLU network is a binary string for all its ReLU neurons, where 1 is assigned to such a neuron whose gradient is 1 and 0 is assigned to such a neuron whose gradient is 0. An activation region consists of all the inputs with the same binary activation string. Compute the activation regions for your trained network and color them for the inputs from -2 to 2. For each activation region, give its activation pattern.

(c) One way to characterize the dynamics of a neural network during training is to quantify the activation pattern changes. At the beginning and end of each epoch, compute the activation patterns for all the training examples and then compute the sum of the Hamming distance between the binary activation

patterns at the beginning and end of the epoch over all the training samples. Then plot the total Hamming distance with respect to the training epoch. Is the overall trend consistent with your expectation? Provide a justification.

Problem 3 (20 points) Answer the following questions regarding the back-propagation algorithms for the fully connected neural networks.

(1) (**5 points**) Describe in your own words Algorithms 6.3 and 6.4 in the Deep Learning textbook (on pages 205 and 206). For each algorithm, you need to describe what each line is doing and why the algorithm would work correctly.

(2) (**5 points**) Explain the complexity of Algorithm 6.3 and Algorithm 6.4 in terms of the dimension of the input (x), the depth of the network (l), and the number of neurons in each layer.

(3) (**10 points**) Using the program you have developed for Problem 2, give the values for all the variables in Algorithms 6.3 and 6.4 for x = 1 and y=sin(π/4). For the weights, use them of your trained network.

Problem 4 (**20 points**) The weights and biases of a softmax layer (implemented using equations (6.28) and (6.29) in the Deep textbook) can be downloaded from http://www.cs.fsu.edu/~liux/courses/deepRL/assignments/hw2_softmax_weights.m. (Note that they are given as a Matlab program and you should be able to extract the numbers easily from the file.)

(1) (**2 points**) What is the architecture of the softmax layer? In other words, how many inputs, how many neurons, and how many connections are there in the layer?

(2) **(4 points)** Classify the following example, which can be downloaded from http://www.cs.fsu.edu/~liux/courses/deepRL/assignments/hw2_softmax_sample.txt .

(3) **(8 points)** Suppose that we use cross entropy loss, the learning rate is 0.05, and the correct label for the sample is the second class (i.e., class 1 if you labels the classes as 0, 1, 2, …, 19), after we apply one step gradient decent optimization using the given sample in (2), how many of the weights and biases will be increased, decreased, or remain the same respectively? For numerical stability, we consider that any change smaller than 0.0001 is the same.

(4) **(6 points)** Compute a small vector so that the new sample created by adding the vector to the given sample will be classified as from the third class (i.e., class 2 if you label the classes as 0, 1, …, 19). The length of the vector (in L_2 norm) should be as small as possible. Explain how you have obtained your solution and confirm that the resulting sample is indeed classified as from the third class. (Note: in the literature, such examples are known as adversarial examples and see https://www.tensorflow.org/tutorials/generative/adversarial_fgsm using TensorFlow. Hint: any reasonable way to compute the change vector will be accepted; to compute the optimal solution, you need to formulate it as a quadratic optimization problem, that is, to minimize the square of the length of the vector under the constraints the output from the third class is the largest.)

(5) **(extra credit option 3 points)** As in (4), could you make a small change to the given sample so that it will be classified as from the third class but with the maximal changed value to be minimized? In other words, we like the vector to be as small as possible according to the L_∞ norm. Explain how you have obtained your solution and confirm that the resulting sample is indeed classified as from the third class. (Hint: to compute the optimal solution, you need to formulate it as a linear programming problem by introducing an auxiliary variable.)

Extra Credit Problem

Problem 5 (5 points) In the paper "Distributed Representations of Words and Phrases and their Compositionality" (which can be downloaded from https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf), an objective function based on negative sampling is defined (see Equation (4) in the paper).

(1) (**2 points**) Suppose that we use k negative samples to approximate the expectation, write down the objective function for one sample. Make sure that you define/explain all the variables in your function.

(2) (**2 points**) Derive the learning rule using gradient descent for the objective function given in (1).

(3) (**1 point**) Explain the key differences between the learning rule for part (2) and the one we derived in class for the skip gram model; if you think that the two learning rules are equivalent, justify your answer.