Assignment

Homework 05

Andrew Park

Applied Machine Learning Homework Assignment

CORNELL TECH | HOME OF THE JACOBS TECHNION-CORNELL INSTITUTE

November 20, 2023

1. General Summary

The coding assignment was about understanding applying Multilayer Perceptrons (MLP) and Convolutional Neural Networks (CNN) for image recognition. The assignement helped me to understand and experiment practical application of CNN in diverse variations, including using Stochastic Gradient Descent (SGD) as the optimizer, utilizing dropout techniques, adding additional convolution layer, and finally using high and low learning rates, iterating over numerous epochs. Since there was a lot of iterations and epochs, it took a long time to finish the assignment, but definitely worthwhile. By visualizing the model performance with training accuracy and validation accuracy as the assignment guide required, I was able to learn essential points of neural network architecture, how to prevent overfitting, and how to improve the performance itself.

The written assignemnt was about exploring several critical concepts in neural networks and machine learning. I had to answer how binary classification works in neural networks, why neural networks generalize so effectively to unseen data, how machine learning concept in hardware (GPU) computing, and finally questions about the correlations with MLP and ReLU units. This theoretical portion was best for enhancing my comprehension of the foundational principles and applications of MLP, ReLU and neural networks.

WRITTEN EXERCISES

Problem 1

Binary Classification Following Lectures 18 and 19, we would like to train a 2-hidden layer neural network with threshold active function to represent the following decision boundaries. If an input point lies in the yellow region, then the neural network should return 1, and if the input point lies within the grey region, then the neural network should return 0. What is the minimum number of neurons in each hidden layer? Explain your reasoning. How many neurons in total do you need in this neural network?

Solution.

10 for the 1st hidden layer + 18 for the 2nd hidden layer + 1 for the output neuron = 29

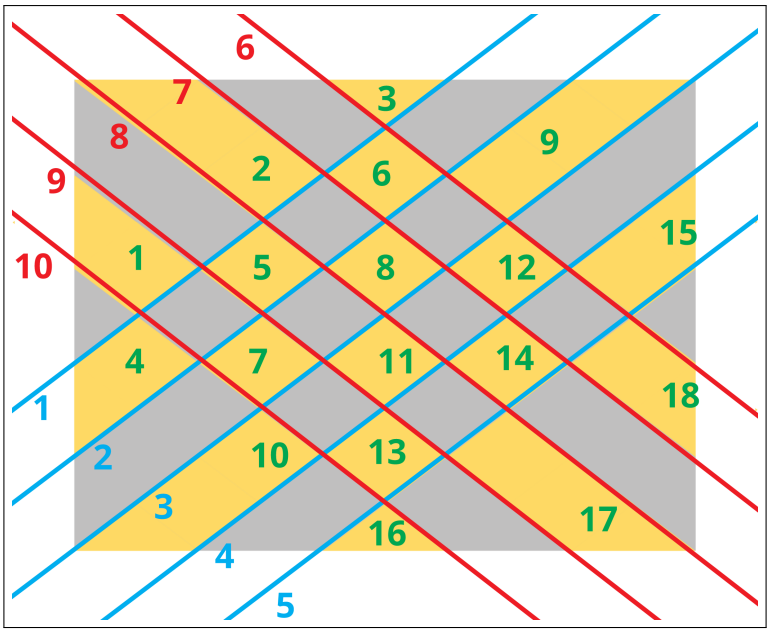

Figure 1: Counting Neurons for the 2-Hidden Layer Neural Network

First, the network starts with two inputs, X_1 and X_2.

Next, the first hidden layer can act as a set of linear classifiers, where each neuron in the layer learns to separate input data based on a linear decision boundary. This can be calculated by measuring the number of diagonal lines that separates checkered pattern, which is 10 neurons (cyan+red).

The second hidden layer can combine the outputs of the first layer's neurons to form the

decision boundaries. In a geometric sense, the first layer's linear classifiers define lines in the input space, and the second layer combines these lines to form more complex shapes. In the provided question, the second layer can combine them to form rhombus or trapezoid shaped regions (which are colored yellow and grey). This can be calculated by measuring the number of yellow shaped regions, which is 18 neurons (green).

Finally, the single output neuron fires the result with 1 or 0. It fires 1 if the input point lies within the yellow region, and 0 if the input point lies within the grey region.

Problem 2

According to the best of our current understanding, why do modern deep neural networks generalize so well to unseen data? Choose all correct answers.

(a) MLPs with threshold activation function are universal function approximators.

(b) A Relu activation function carries more information about the input data than a sigmoid activation function.

(c) MLPs are shift invariant.

(d) Modern deep neural networks do not have enough capacity (number of nodes and number of layers) to overfit the training data.

(e) New tricks like batch normalization and residual connections make it highly unlikely to overfit.

(f) Large-scale datasets with 50,000+ examples are practically impossible for any model to overfit.

(g) While some theoretical progress is being made on toy models, the research community still does not know why practical deep nonlinear networks generalize so well.

Solution.

(a) (b), (e), (g)

(a) Multilayer Perceptrons (MLPs) with threshold activation functions can approximate any continuous function. This universal approximation capability is a fundamental reason why deep networks can generalize well.

(b) This is true because ReLU functions help alleviate the vanishing gradient problem in deep learning models, allowing better learning and generalization. However, saying it carries "more information" than sigmoid is not accurate. ReLU functions are more about the efficiency of training and the capability of the network to learn complex patterns due to

the non-saturating nature.

(e) Techniques like batch normalization and residual connections have been shown to improve the generalization of deep neural networks by making the optimization landscape smoother, which in turn helps in training deep networks more effectively.

(g) This statement reflects the current state of research in deep learning. While there is some understanding of practical deep nonlinear networks, a comprehensive theoretical explanation for the strong generalization performance is still lacking, especially in complex, real-world scenarios.

Problem 3

In the guest lecture about hardware for machine learning, you learned about systolic arrays for carrying out efficient matrix multiplication.

Assume you have a 32×32 systolic array that can multiply a 32×32 input matrix by a 32×32 weight matrix. Assume the input matrix streams horizontally from left to right, and the weight matrix streams vertically from top to bottom (same as the example in class). How many cycles do you need to perform this matrix multiplication?

Now, assume you have a M ×N systolic array that can multiply a M ×K input matrix by a K ×N weight matrix, as shown below. How many cycles do you need to perform such a matrix multiplication? You should express the cycle count using M,N,K.

Solution.

For a systolic array architecture, the number of cycles needed for matrix multiplication depends on the propagation of data through the array and the size of the matrices involved.

For a 32 × 32 systolic array multiplying two 32 × 32 matrices:

1. The first element of the input matrix will take 32 cycles to fill the array horizontally.
2. Next, 32 cycles for the last element of the input matrix to pass through the array horizontally.
3. Lastly, 32 more cycles for the results to propagate through the array.

However, there are two overlaps. One is when the first element of the input matrix reaches it's end and the last element of the input matrix starts to fill the array. The other is when the first output begins to leave while the last inputs are still entering. So, the total number of cycles C is $C = 32 + 32 + 32 - 2 =$ **94 cycles**

The generalized formula for an $M \times N$ systolic array multiplying an $M \times K$ input matrix with a $K \times N$ weight matrix is $C = M + K - 1 + N - 1$. This accounts for the time to fill the array with the first row of the input matrix (M cycles), the time to pass the last row of the input matrix through the array ($K - 1$ cycles, since the array is already filled with the first $M - 1$ rows), and the time for the results to propagate out of the array ($N - 1$ cycles, again because the results start to come out as the last elements are still entering).

$C = M + K + N - 2$.

Problem 4

(a) Can a multilayer perceptron with one hidden layer of ReLU units represent any function that multilayer perceptron with one hidden layer of linear units (no activation function) can represent? If so, would you need more, less, or the same number of hidden nodes?

(b) Can aMLPwith one hidden layer of linear units (no activation function) represent the mapping that an MLP with ReLU units can represent? If so, would you need more, less, or the same number of hidden nodes?

Solution.

(a) Yes, an MLP with one hidden layer of ReLU units can represent any function that an MLP with one hidden layer of linear units can. The capability of ReLU to represent linear functions means that we can use the same number of ReLU units as linear units to represent linear functions since ReLU acts linearly for positive inputs and simply zeros out negative inputs. The complexity comes when we need to simulate both positive and negative responses of linear units using ReLUs, which might lead to requiring more units for achieving the same output as linear units. However, in practice, because of the zeroing out of negative inputs, we may actually end up with a sparser and potentially more efficient representation.

(b) No, an MLP with one hidden layer of linear units cannot represent all the mappings that an MLP with ReLU units can. This is because linear units are fundamentally limited to linear transformations, and stacking multiple layers of linear transformations still results in a linear transformation overall. On contrary, ReLU introduces non-linearity into the model, allowing the network to capture more complex, non-linear relationships in the data. As a result, no matter how many linear nodes we have, they cannot capture non-linear patterns that ReLU nodes can . In summary, while ReLU units can represent linear unit functions (potentially with more nodes), linear units cannot represent the full range of functions that ReLU units can, due to the inherent limitation of linear units in capturing non-linear relationships.