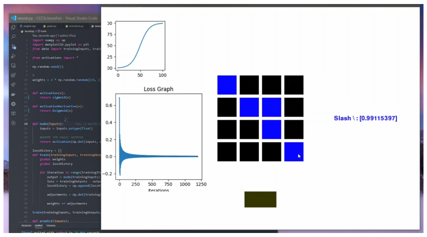
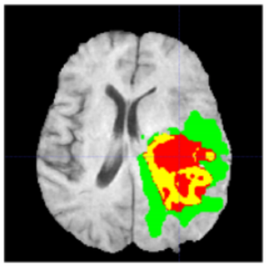
How Various Activation Functions Affect the Loss and Output of a Neural Network

2021

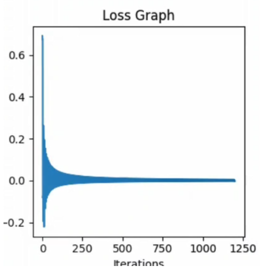


```
import numpy as np
import matplotlib.pyplot as plt
from data import trainingInputs, trai

from activations import *

np.random.seed(1)

#
weights = 2 * np.random.random((16, 1


def activation(x):
    return sigmoid(x)

def activationDerivative(x):
    return Dsigmoid(x)


def node(inputs):
    inputs = inputs.astype(float)

    #WHERE THE MAGIC HAPPENS
    return activation(np.dot(inputs, w

lossHistory = []
def train(trainingInputs, trainingOutp
    global weights
    global lossHistory

    for iteration in range(trainingIte
        output = node(trainingInputs)
        loss = trainingOutputs - outpu
        lossHistory = np.append(lossHi

        adjustments = np.dot(training

        weights += adjustments

train(trainingInputs, trainingOutputs,

def predict(inputs):
```

Abstract — In machine learning, artificial neural networks take advantage of "activation functions." These functions determine each node/neuron's final output in the network (determining whether or not to fire/synapses) to output the correct final prediction (similarly to a real brain). Determining what activation function to use is no easy task, but researchers have found that some activation functions are better suited than others for various tasks through trial and error. Oftentimes, developers build classification neural networks using the sigmoid activation function since it has been proven to work. Its effectiveness comes from having a limited range [0,1] (useful for boolean classification) and an excellent track record, but it is important to challenge this industry standard by searching for more optimal alternatives. Optimizing classification neural networks could result in more performant models for utilities that rely on machine learning. This experiment's model was trained to differentiate basic patterns such as slashes and the letter O within a four-by-four matrix of pixels. While this model serves a rather niche test case, classification neural networks like these impact fields such as radiology,



defense system development, biometrics authentication, and more when scaled up.



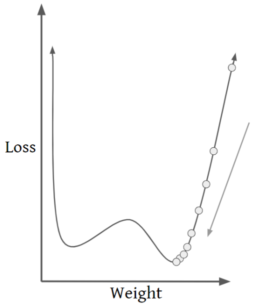
This experiment will apply the scientific method to determine which activation function is most suitable for simple pattern recognition tasks (of two possible predictions). The accuracy is measured by rendering a loss graph for each activation function. The loss graph visualizes the difference between the output and the preferred output over a period of training iterations. Note that the training data will be kept constant across all tests in order to abide by the scientific method. For this experiment, the following activations will be tested: sigmoid, binary step, johnStep (custom), & softplus. The results alluded that the binary step was the most accurate and had a 0% loss for each test.

1. Background/Introduction

To better understand this experiment's technicalities, it is crucial to understand how artificial neural networks (ANNs) work in general. ANNs imitate the brain; they consist of "nodes" which are equivalent to neurons in the brain. Each node is assigned a random value called weights, similarly to how neurons have strengths between connections. The weights are randomly initialized; therefore, generate an initial random output when asked to classify a given input. In order to make the output less random, the ANN must be "trained." The iterative training process tweaks the weights through a process called gradient descent and backpropagation.

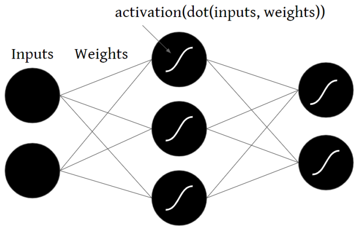
Gradient descent is the process of calculating how wrong the model is (%

error) to construct what is called a "loss function." The following steps include finding the minimum of the loss function to generate adjustment values which tweak the weights (making the model less wrong over time). Gradient descent does this by taking the derivative at a point on the loss function to find the slope/gradient. It then moves one unit (the learning rate) in the downward direction of said gradient/slope until it reaches the local minimum. This process is executed repeatedly until the model is "trained." Once fully trained, given any input (even an input it has never seen before) a reasonable prediction based on the input is returned.



As mentioned, the purpose of the ANN in this experiment is to classify slashes and Os. One can appreciate the fact that instead of hard-programming each possible combination of pixel orders that could represent a slash or letter O, only a select few examples were shown to the model (3 of each) and the rest was learned (think of this as an advanced regression).

2. Importance of Activation Functions



One key component of an ANN is called an "activation function," as mentioned earlier, the activation function's output determines each node/neuron's final output (the argument to the activation function is computed by taking the dot product of the weight and input vectors). Not only that, but the activation function and its derivative, play a crucial role in the training process. Through the use of activation functions, neural networks can better fit a curve to data compared to traditional regression algorithms.

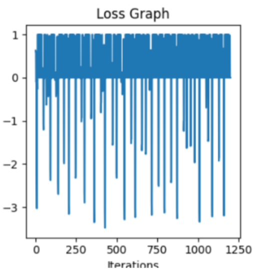
Engineers and scientists choose activation functions according to the task they seek to accomplish. Some activation functions have limits in the infinity positive or negative direction (such as the sigmoid). These limits can be useful if the range of possible outputs is better off being restricted. There is a downside to this, however (in some cases), a problem called the "vanishing gradient" can interfere with the training process. It happens when the slope/gradient is too small for gradient descent to be effective (the derivative of the sigmoid gets infinitesimal with large input scalers). Another deciding factor in choosing an activation function could be how fast it is; some functions such as ReLU and the binary step take less operations to compute which results in a brisk training and predicting process.

Interestingly enough, there is an infinite number of possible activation functions, but only the most popular ones plus the "johnStep activation" have been tested in this experiment (the johnStep is a horizontally compressed sigmoidal function which makes it more step-like). Observing the sigmoid and binary step's effectiveness led to the creation of the johnStep.

The four activations function’s losses will vary, identifying the most optimal activation function can be achieved by observing which had the lowest loss among the four.

3. Results

The average loss (how wrong the model was) is as follows: sigmoid — 0.01213811, binary step — 0.0. softplus — 0.57875671, johnStep — 6.84870763E-41. The average loss does not tell the whole story, however. It is also important to interpret the data for both the slash and the letter O independently. When making said observation, one can not overlook the vast difference between the softplus' recognition of the slash and letter O. Since the numerical representation of a slash was 1, and the softplus does not have a limited range in the positive direction, the loss would often spike while still classifying slashes correctly. These spikes are due to large losses in the **positive direction.** The model's prediction is still correct, though, since the final output (regardless of loss) is closer to 1 than it is to 0 and thus returning "slash." Despite these nuances, the overall most effective activation functions were the binary step and the johnStep. This could be due to having a restricted range [0,1] and limits in the infinitely positive/negative direction that approach 1 and 0 much quicker than the sigmoid.



References

Sanderson, Grant. "Neural Networks." *3Blue1Brown*. 19 January 2021 https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

Enyinna Nwankpa, Chigozie. "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning." *Arxiv*. 19 January 2021 https://arxiv.org/pdf/1811.03378.pdf

Image Citation

“Automatically Segmenting Brain Tumors with AI.” *Nvidia*. 27 Nov. 2018, news.developer.nvidia.com/automatically-segmenting-brain-tumors-with-ai