Geometry of Machine Learning

C. Arnett & J. Cox, & S. Pothagoni
under Dr. Matt Holzer and Jessica Masterson

Mason Experimental Geometry Lab

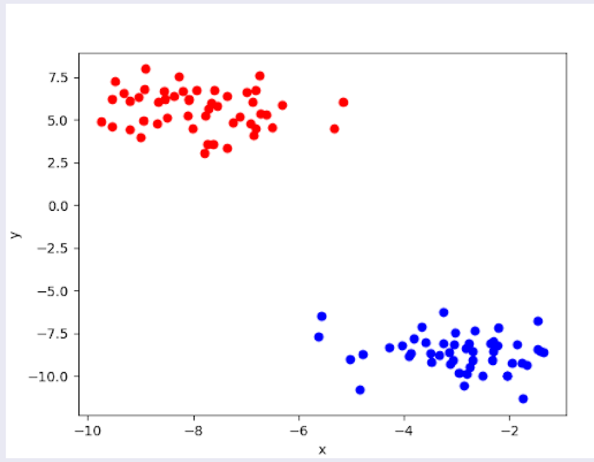
GEORGE MASON UNIVERSITY

December 4, 2020

Linear Classification

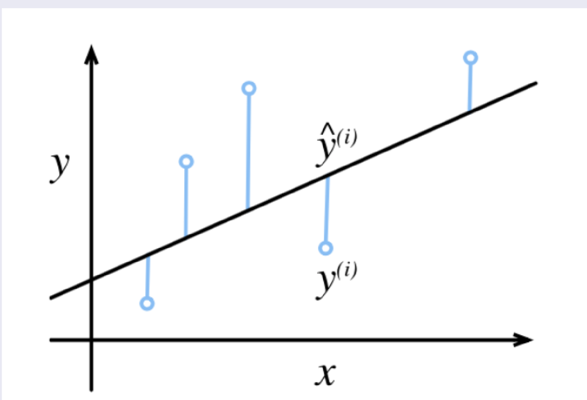
Here is a set of data with features x and y and label z. On the right is the visualization of the data.

x	y	z
-9.0	5.0	1
-8.0	6.0	1
-3.0	-8.0	0
-2	-10.0	0
-1	-8.0	0
...	...	...

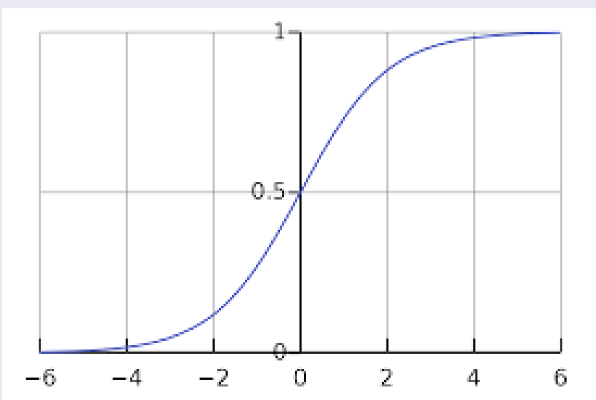


Visually, it is clear that the data is linearly separable. However, if we wanted to automate this process how should we program a computer to do so?

MSE and The Sigmoid Function



Mean Squared Error [1]



Sigmoid Function

$$MSE = \frac{1}{n}\sum_{i=1}^{n}|Z_i - \hat{Z}_i|^2 \qquad S(x) = \frac{1}{1+e^{-x}}$$

Cost Function and Loss Function

Predicted Value:

$$\hat{Z}_i = S(x_i w_x + y_i w_y + b)$$

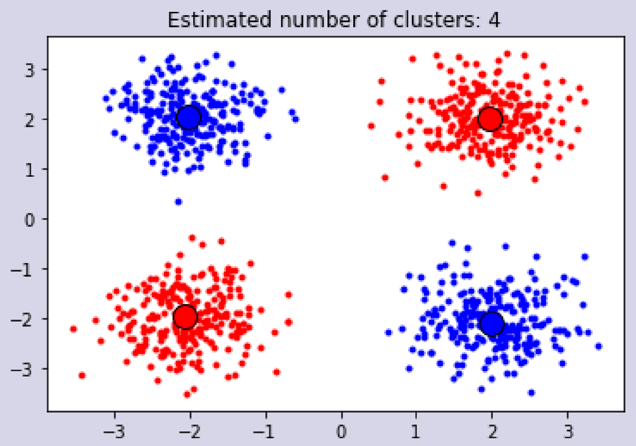
An accurate weight and bias is required for an accurate prediction. Rather than trying to guess the right weight and bias we instead generate random ones. These weights and biases are then updated by

$$\mu = \textbf{Learning Rate}$$
$$W_t = W_{t-1} - \mu MSE \nabla_{W_{t-1}}$$
$$b_t = b_{t-1} - \mu MSE \nabla_{b_{t-1}}$$

Understanding and Visualizing Neural Networks

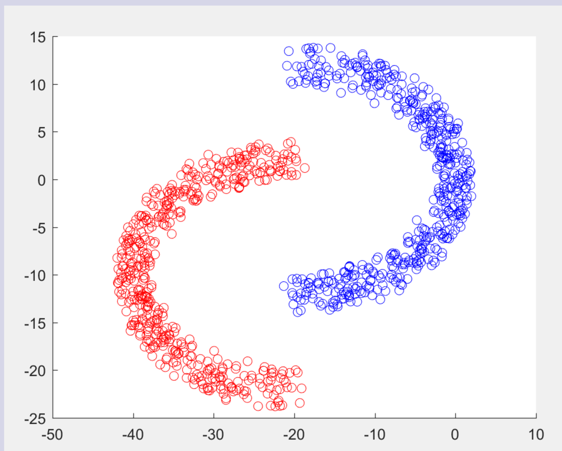
Nonlinear Data

However, most datasets aren't as simple as the shown before. So, what happens when our data isn't linearly separable?
Example of a nonlinear data set:

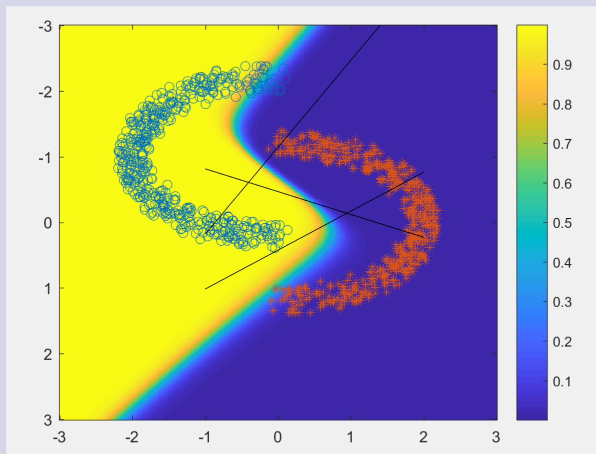


Visual Interpretation of Multilayered NN

Below is an example of a training set that needs to be classified using the NN.

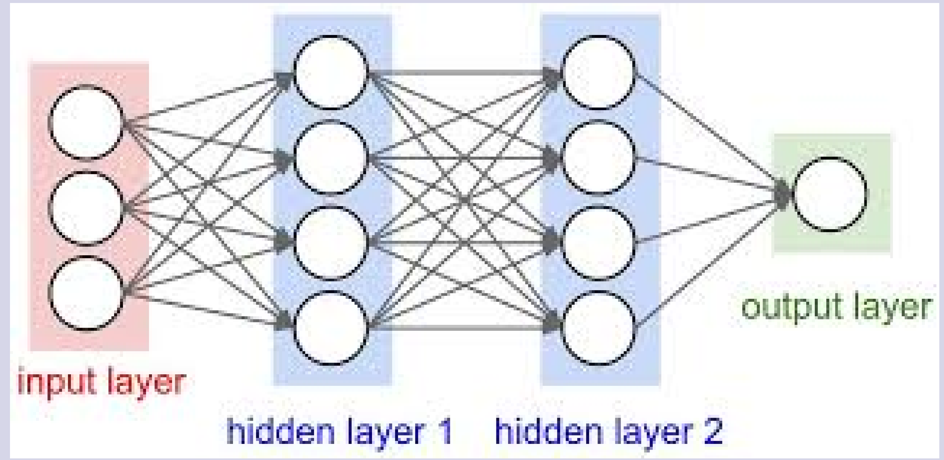


After implementing Machine Learning, the Neural Network is able to accurately separate the data and classify it. This can be seen with the lines drawn and the colored decision boundary.



In essence, each neuron is being used to plot out a line to accurately separate the data. This then gets sent to through the Sigmoid function which to create a decision boundary. We repeat this process for every layer

Multilayered NN



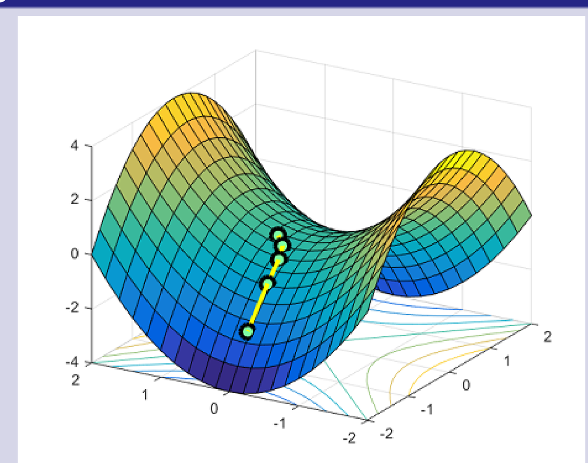
A. Rosebrock (2016)

Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a common learning strategy that involves randomly sampling from the input data for each epoch rather than passing the entire dataset. This increases the speed of learning but it can also help the network get off saddle points when trying to find a local minimum.

```
import numpy as np
# X = [...]
# y = [...]
epochs = 100
index = np.random.choice(len(X), epochs, replace=True
)
for i in range(epochs):
    X_ = np.expand_dims(X[i], axis=0)
    y_ = np.expand_dims(y[i], axis=0)
```

Saddle Points



Escaping Saddle Points (2016)

By randomly sampling through the data, each sample has a strong opinion on the direction of the gradient. So, this prevents the gradient from stalling at a local minimum.

Results

"Stalling" Gradient Descent

As it turns out, the NN can sometimes get "stuck" during the learning process for a number of epochs (iterations).

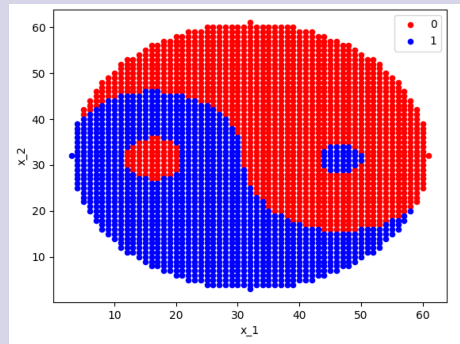
Evidence and associated literature suggests that the increased frequency of saddle points at higher dimensions can sometimes interfere with the convergence of Gradient Descent - this is may be due to getting caught in some saddle point. [3]

Learning the Surface of the Torus

Given a random point on the surface of the torus versus a random point in $\mathbb{R}^3$, our NN was is able to predict accurately whether or not it is on the surface of the torus or not.

However, if we create a noisy torus and ask the NN the same question the accuracy substantially decreases.

Learning the Yin-Yang Symbol



Yin-Yang dataset

References

- A. Zhang and Z. C. Lipton and M. Li and A. J. Smola (2020), Dive into Deep Learning 2020, https://d2l.ai
- Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli and Y. Bengio (2014) *Identifying and attacking the saddle point problem in high-dimensional non-convex optimization* arXiv:1406.2572v1 [Computer Science]
- *Off the Convex Path - Escaping Saddle Points (2016)*, [Link]
- N. Manzini (2019), Single hidden layer neural network, [Link]
- A. Rosebrock (2016), A simple neural network with Python and Keras, [Link]