Machine Learning Engineer Course
Day 17

- Deep Neural Network -

DIVE INTO CODE

Thursday July 1, 2021

DIOP Mouhamed

Agenda

1. Check-in
2. How to proceed
3. Quick Review
4. Deep Neural Network
5. Assignment
6. Scratch Deep Neural Network – Sample Code
7. Check-out

Check-in

3 minutes Please post the following point to Zoom chat.

Q. What did you learn in the previous week?
(Anything is fine.)

How to proceed - Objective

Understand the developmental content of neural networks through scratch

Let’s learn the basics of Deep Learning here

Quick Review (Neural Network)

Points for solving the previous assignment

Neural Networks

Weights, Biases, Nodes, Activation Functions

Back-propagation

Cross-entropy loss function (objective function)

Loss, Gradient

Full Batch, Mini Batch, Iterations, epochs, and

Batch size (data size of training)

Deep Neural Network

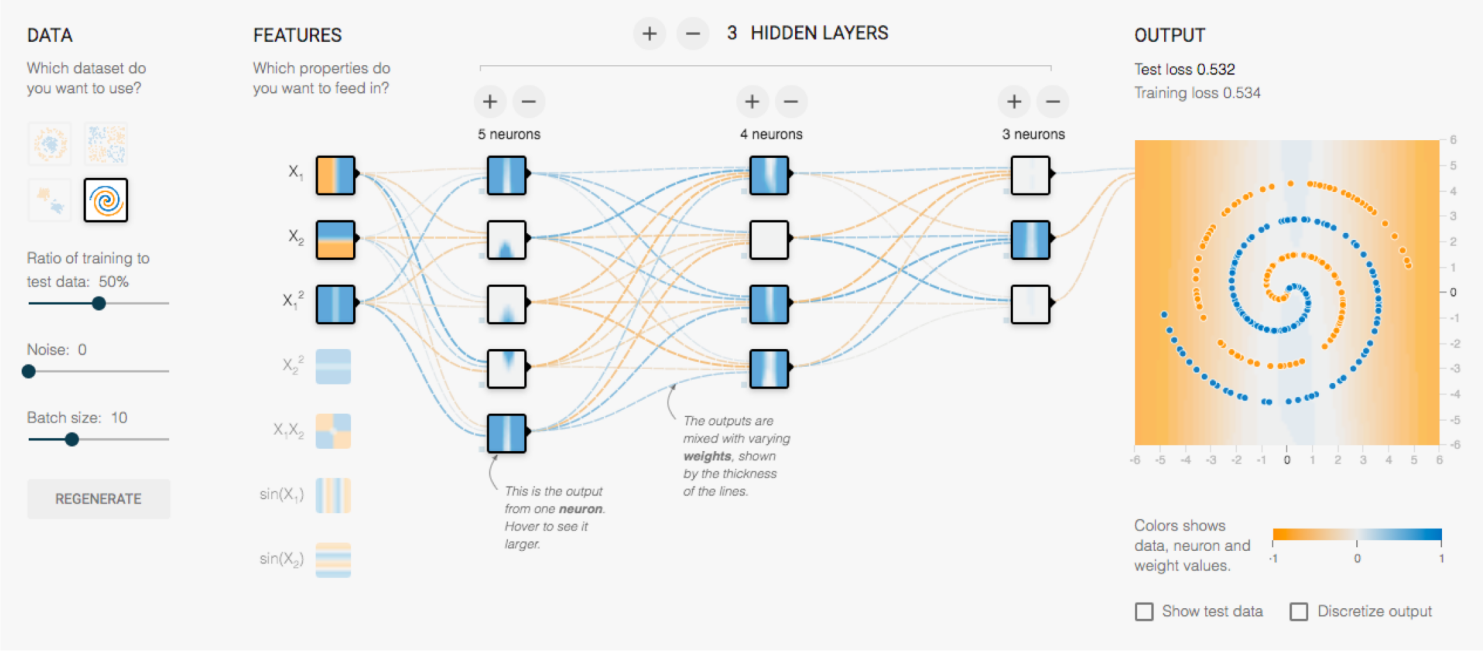
Deep Neural Network

Let's take a look at the whole network.

Try to increase or decrease the HiddenLayers or the number of nodes on this site.

You can also change the Activation function.

http://playground.tensorflow.org/#activation=relu&batchSize=10&dataset=spiral&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=5,4,3&seed=0.48429&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=true&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false



Learn about the issues in deep learning

① Ability to express oneself ← This time

Depends on the structure of the network

② Generalization capability

Depends on network structure & learning algorithm

③Optimization Capability

Network structure & Depends on learning algorithm

Learn about the issues in deep learning

In the late 1980s, there was a lively debate about how complex a neural network can learn a function, or in other words, its representation capability.

This is to ask how far the model can approximate the true continuous function (1) (function approximation capability).

(1) A function f(x) is continuous if it is continuous for all values of x in the domain of definition. f(x) is a continuous function

The case of shallow neural nets

"Even with a shallow model such as a three-layer neural net (with one intermediate layer), we can approximate any function with arbitrary accuracy by increasing the number of nodes in the intermediate layer infinitely.

From the Universal Approximation Theorem

https://pdfs.semanticscholar.org/05ce/b32839c26c8d2cb38d5529cf7720a68c3fab.pdf

Formulate the function approximation capability of a three-layer neural network: :
Approximate a true continuous functior $f : \mathbb{R}^m \to \mathbb{C}$ by g(x) such that

Parameters for output values

sigmoidal function

$$g(x) = \sum_{j=1}^{J} c_j \eta\left(a_j \cdot x - b_j\right)$$

Intermediate layer nodes

$$\left(a_j, b_j, c_j\right) \in \mathbb{R}^m \times \mathbb{R} \times \mathbb{C}$$

$\left(a_j, b_j\right) \in \mathbb{R}^m \times \mathbb{R}$ are the parameters of the middle layer

sigmoidal : $h(x) \to \begin{cases} 1 & (x \to \infty) \\ 0 & (x \to -\infty) \end{cases}$

table 2.1: $g(x) := \sum_{j=1}^{J} c_j \eta(a_j \cdot x - b_j)$ due to $f(x)$ Approximability

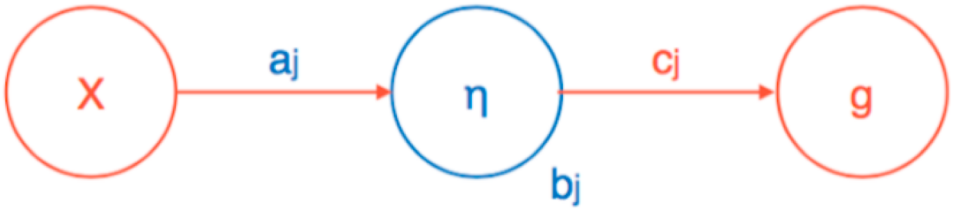
	f	η		
Irie and Miyake 1988	L^1	L^1	each point	Fourier inversion formula
Cybenko 1989	C	sigmoid	Broadly defined	Hahn-Banach
Hornik+ 1989	C	sigmoid	Broadly defined	Stone-Weierstrass
Funahashi 1989	C	Bounded continuous and monotonically increasing	Broadly defined	Fourier inversion formula
Carroll and Dickinson 1989	$\mathcal{D}$	tanh	L^2	Radon inversion formula
Mhaskar and Micchelli 1992	L^p	$\mathcal{S}_0'$	L^p	B-spline
Leshno+ 1993	C	$\mathcal{S}_0'$	Broadly defined	approximate unit element

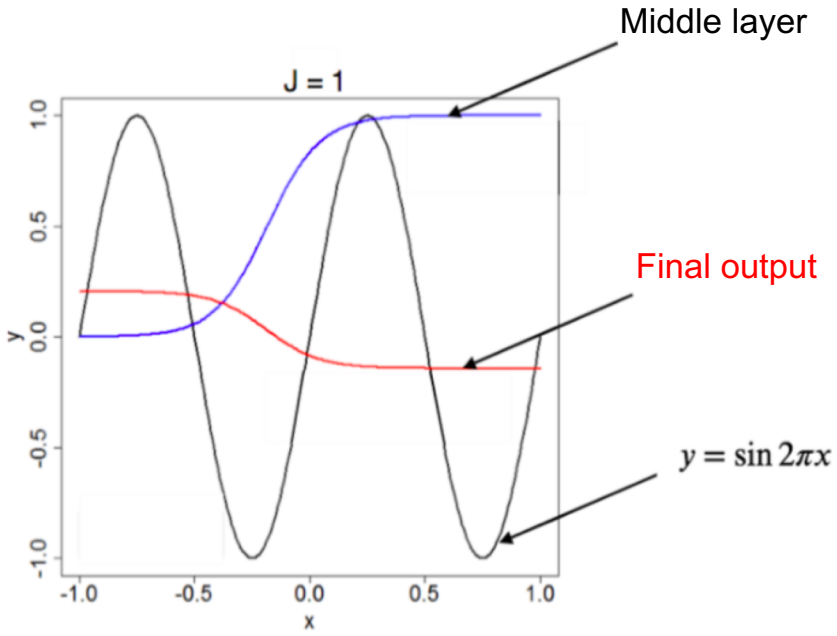
The case of shallow neural nets

Function to be approximated : $y = \sin 2\pi x$

NN with 1 node : $g(x) = \sum_{j=1}^{1} c_j \eta\left(a_j \cdot x - b_j\right)$

Intermediate layer nodes



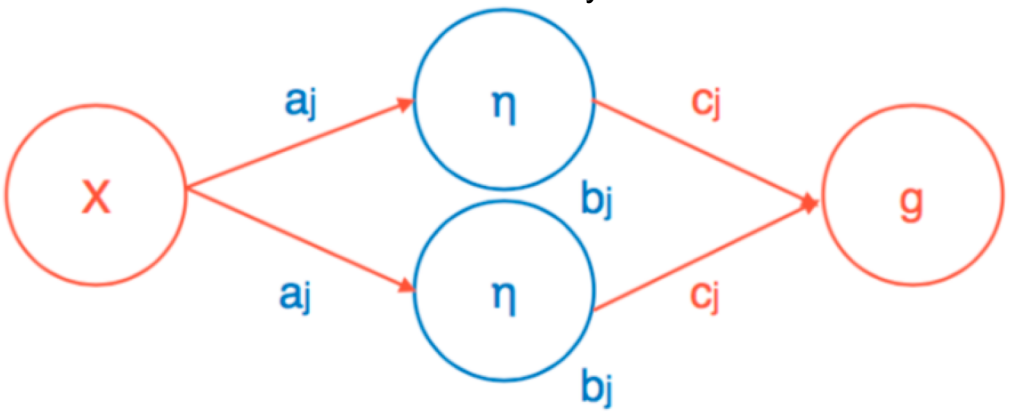


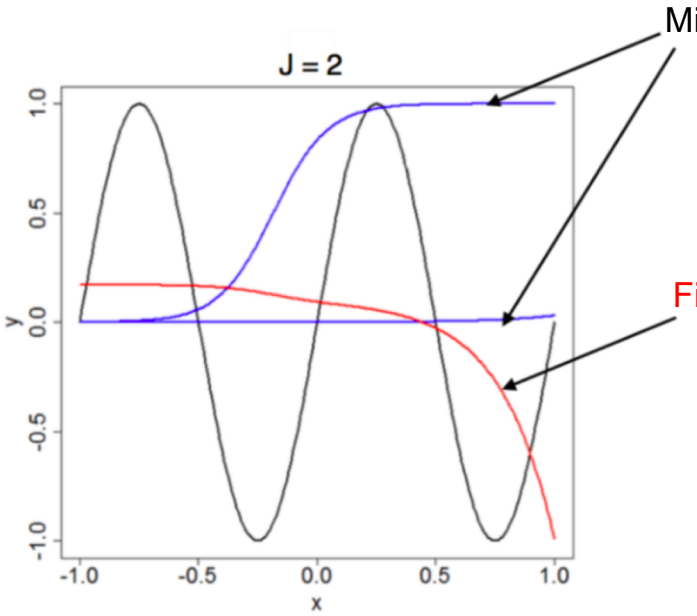
The case of shallow neural nets

Function to be approximated : $y = \sin 2\pi x$

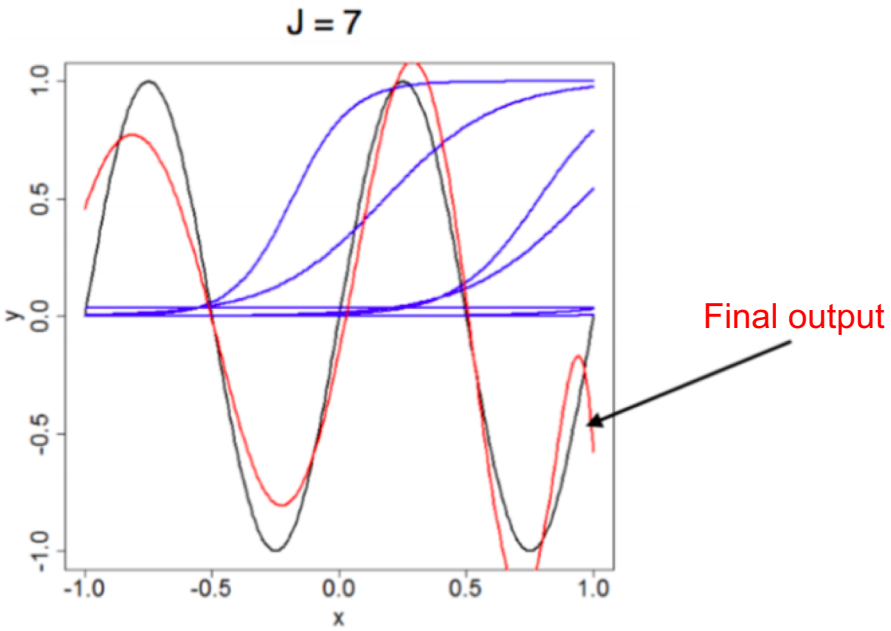
NN with 2 nodes : $g(x) = \sum_{j=1}^{2} c_j \eta\left(a_j \cdot x - b_j\right)$

Intermediate layer nodes



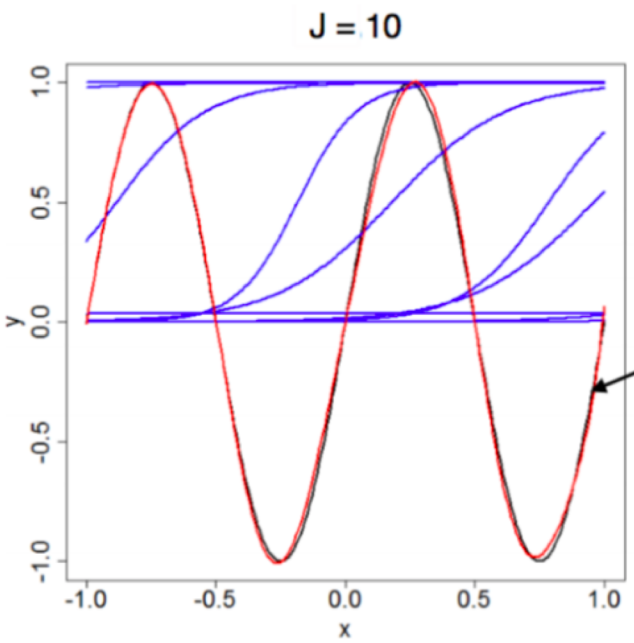


The case of shallow neural nets



Final output

$$g(\mathrm{x}) = \sum_{j=1}^{7} c_j \eta\left(\mathrm{a}_j \cdot \mathrm{x} - b_j\right)$$



Final output

$$g(\mathrm{x}) = \sum_{j=1}^{10} c_j \eta\left(\mathrm{a}_j \cdot \mathrm{x} - b_j\right)$$

The case of deep neural nets

The case of deep neural nets

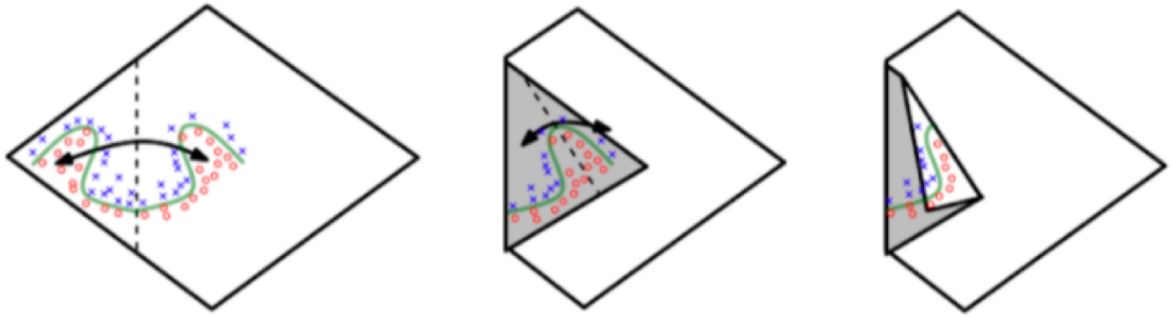
In a shallow neural net, if It has been proven that if there are an infinite number of nodes in the middle layer,it is possible to approximate any function.

$$g(\mathrm{x}) = \sum_{j=1}^{\infty} c_j \eta\left(\mathrm{a}_j \cdot \mathrm{x} - b_j\right)$$

One intermediate layer is sufficient to approximate an arbitrary function, but that one layer can be unrealistically large.

By increasing the number of nodes and expanding the layers horizontally, the expressive power increases polynomially, while by increasing the number of layers, the expressive power increases exponentially.

Do we still need to add more layers?



Number of units in the middle layer

$$\left(\prod_{i=1}^{L-1} \left\lfloor \frac{n_i}{n_0} \right\rfloor^{n_0}\right) \sum_{j=0}^{n_0} \binom{n_L}{j}.$$

L : Number of layers
n : Width of the middle layer
n_0 : Dimension of input

$$n_i \geq n_0 \text{ for all } i \in [L]$$

Expressive ability: How many polyhedra can you divide the domain into?

https://papers.nips.cc/paper/5422-on-the-number-of-linear-regions-of-deep-neural-networks.pdf

Learn about the issues in deep learning

① Performance

Depends on the structure of the network

② Generalization ability ← A little introduction

Network structure & Depends on learning algorithm

③ Optimization Capability

Network structure & Depends on learning algorithm

Generalization capability

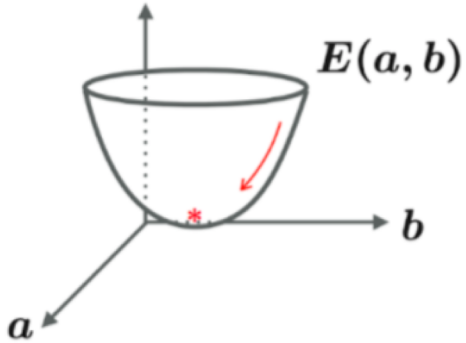
We want to get a model with high generalizability.

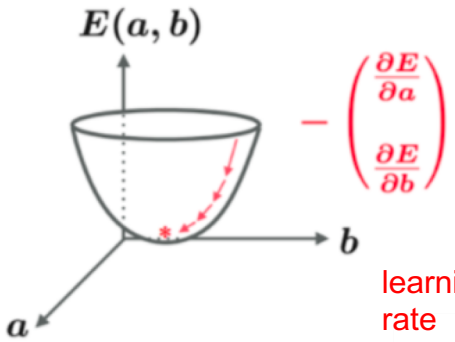
To acquire rules and knowledge from a finite amount of training data, and to acquire models that are not included in the training data (sampled from the same distribution), but that can be successfully inferred from unknown data with similar properties to the training data.

Gradient Descent Method:

Solving optimization problems (learning data) numerically → Find the minimum value

$$a^*, b^* = \mathrm{argmin}\, E(a, b)$$



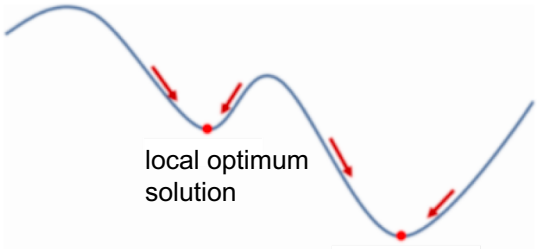


$$-\begin{pmatrix} \frac{\partial E}{\partial a} \\ \frac{\partial E}{\partial b} \end{pmatrix}$$

learning rate

$$a^{t+1} = a^t - 0.01\frac{\partial E(a^t, b^t)}{\partial a}$$

$$b^{t+1} = b^t - 0.01\frac{\partial E(a^t, b^t)}{\partial b}$$



local optimum solution

The most suitable solution for large domains

Generalization capability

Some minima are good for generalizability and some are bad for generalizability

Good: flatness

Bad: sharpness

Stochastic Gradient Descent (SGD)

Batch size is small.

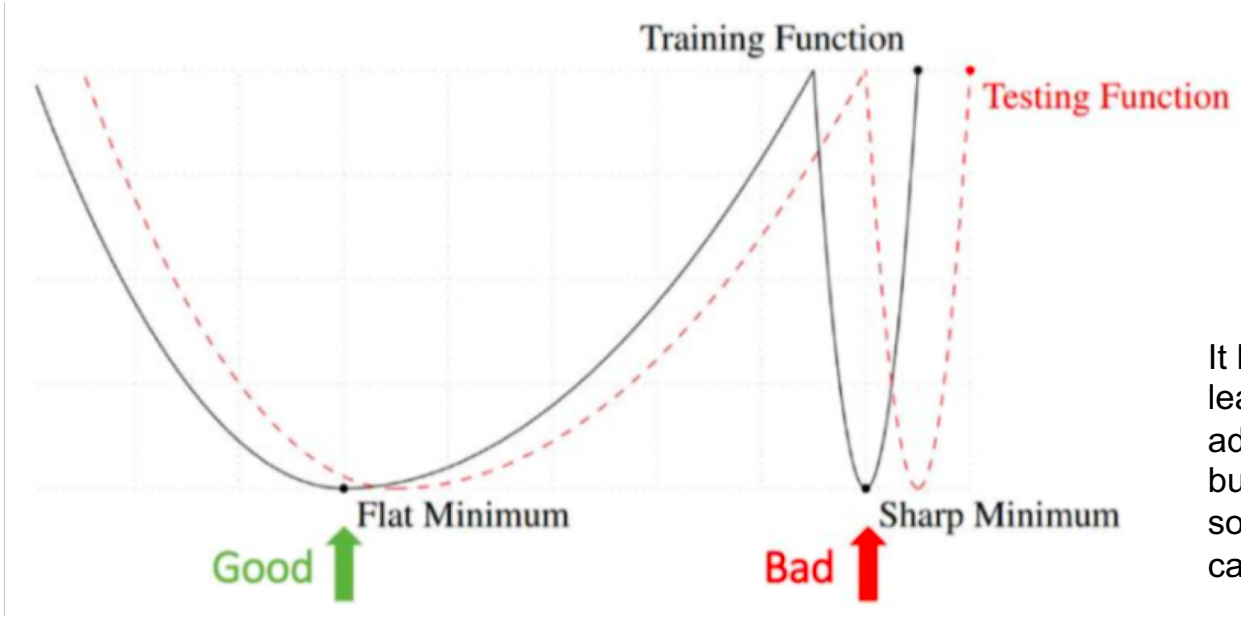
Increases gradient noise, converges to flat global optimal solution, and has good generalization performance.

Batch size is large.

Converges to a sharp global optimum solution, with low generalization performance

Are you implicitly regularizing?

https://arxiv.org/pdf/1710.06451.pdf

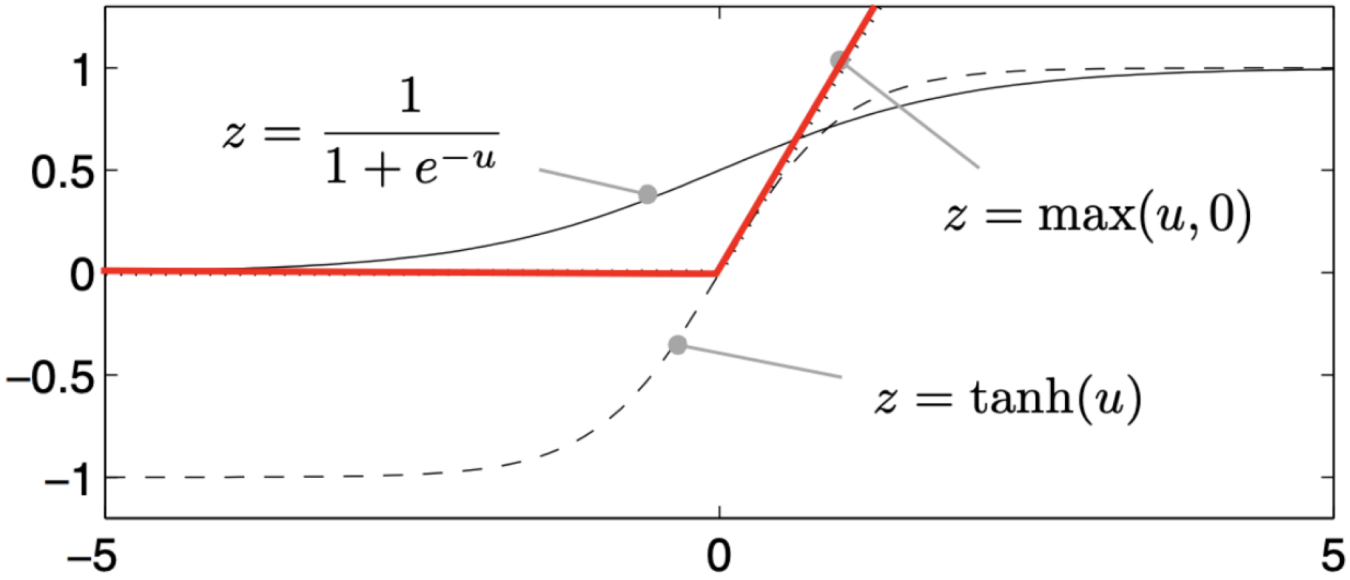


It has been argued that AdaGrad learning, which automatically adjusts the learning rate, is fast but converges to the optimal solution with low generalization capability.

Good combination of activation function and initial value

```
Xavier ---Sigmoid/Tanh
He ---ReLU
```

Xavier
He
ReLU
Tanh

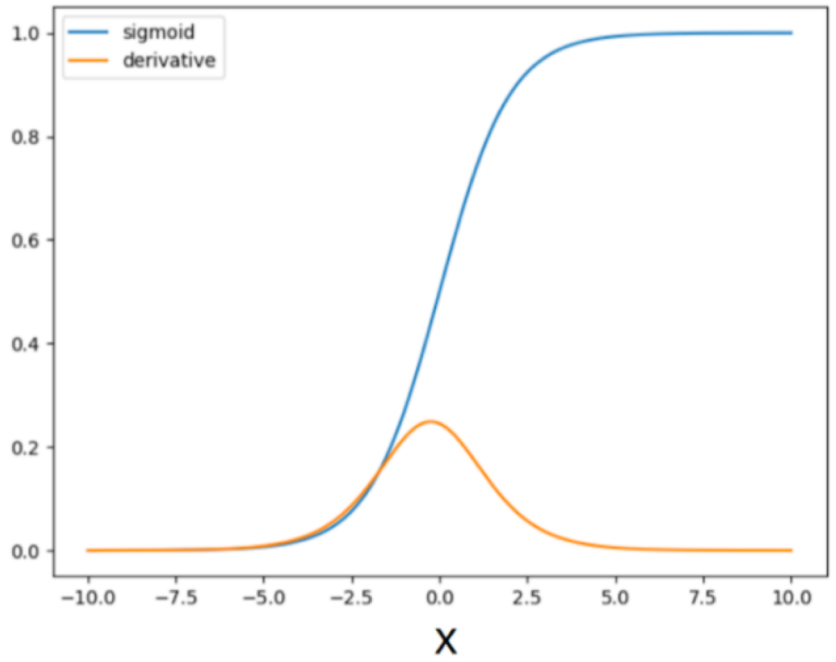


Vanishing of a gradient

About the activation function

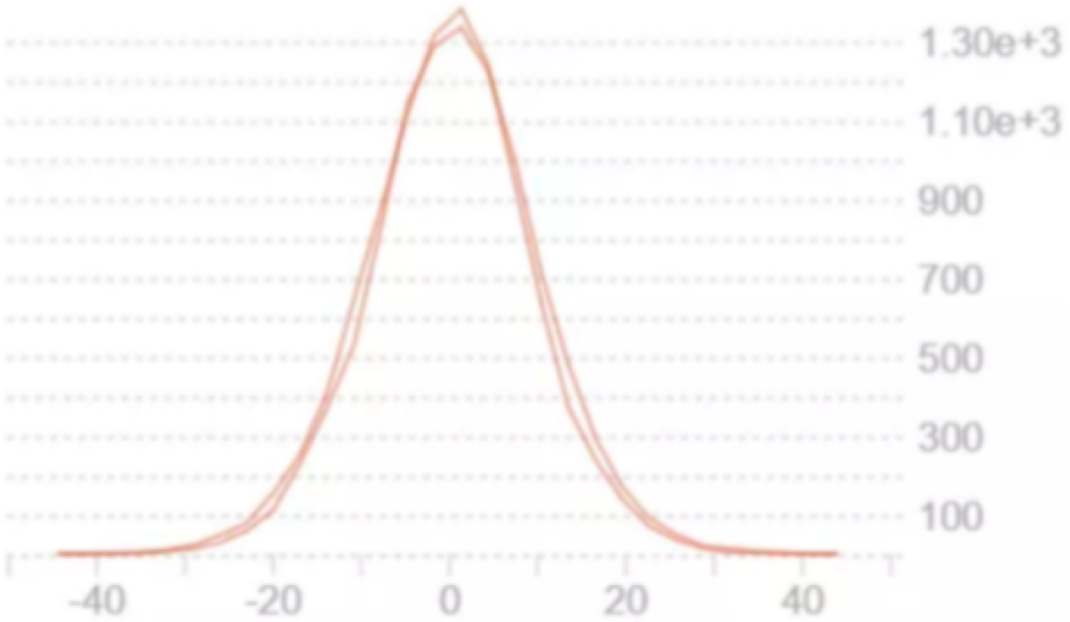
We've tried various network architectures, but for some reason, learning sometimes doesn't progress.

One of the reasons why back propagation does not work well is the saturation of the hidden layer. The blue function below is the Sigmoid function, and the orange function is its first derivative. We can see that when abs(x)>6, the derivative approaches 0 (saturation). Since the back propagation method to update the weights depends on the derivatives of the activation function, when the output of a node reaches the saturation region, learning slows down or does not take place at all.



Vanishing of a gradient

Let's pass the Sigmoid function with the initial variance of the weights as 1.0.
Let's assume the input distribution is as follows.



Vanishing of a gradient

Output distribution

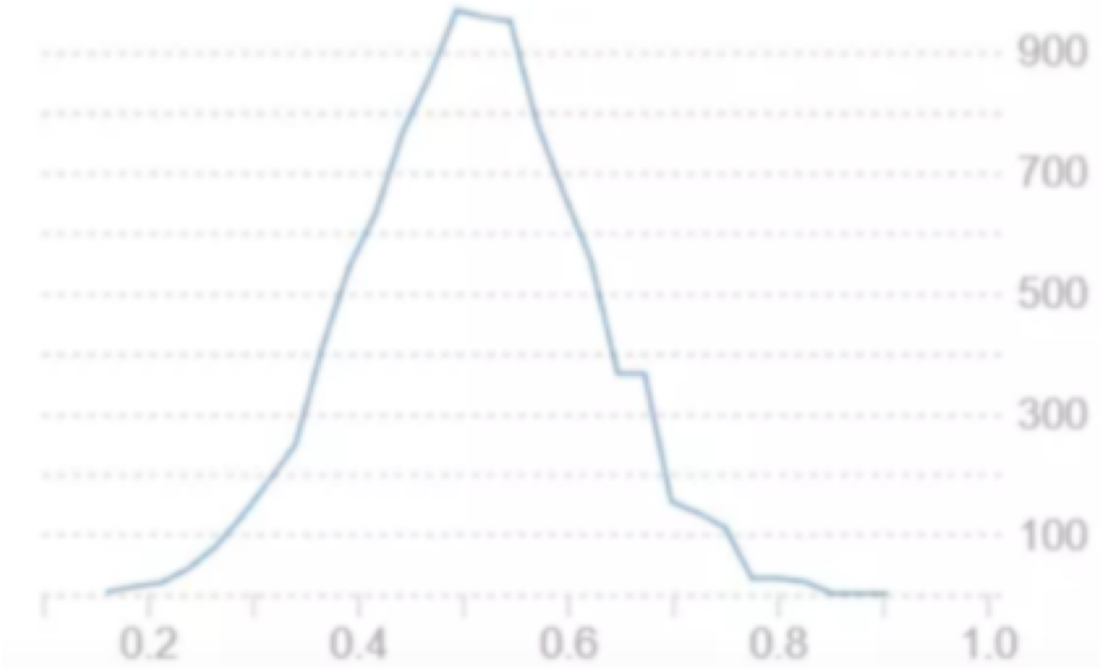
After passing through Sigmoid, the output was narrowed down to the saturation region of Sigmoid curve, i.e. close to 0 and 1.

In back propagation, the derivatives of the Sigmoid function are adamantly product of the flowing delta, so the gradient may vanish when the derivatives are almost zero.

Gradient loss (avoidance)

If the initial value of the weight is Xavier

The output was distributed around the linear region of the Sigmoid function (0.5), and no saturation occurred.



Sample code

How to solve problems

“Scratch Deep Neural Network”

[Problem 1] Fully Connected Layers
[Problem 2] Initialization method
[Problem 3] Optimization methods
[Problem 4] Activation Functions
[Problem 5] ReLU Class creation
[Problem 6] Initial value of weight
[Problem 7] Optimization method
[Problem 8] Class completion
[Problem 9] Learning and estimation

Sprint 10 – Deep Neural Network

Explanation about this Sprint is given but please try it on your own first.

Sprint 10 – Deep Neural Network

Please work on your own after class and submit your assignments on DIVER.

Sprint 10 – Deep Neural Network

A Sample Code of this Sprint is given but please try it on your own.

Sprint 10 – Deep Neural Network

Please work on your own after class and submit your assignments on DIVER.

ToDo by next class

Next class will be Zoom : Thursday July 8, 2021 19:30~20:30

🚨ToDo: Convolutional Neural Network 1 (SimpleConv1d)
https://diver.diveintocode.jp/curriculums/1899

Check-out

3 minutes Please post the following point to Zoom chat.

Q. Current feelings and reflections
(joy, anger, sorrow, anticipation, nervousness, etc.)

Thank You For Your Attention