Machine Learning Engineer Course
Day 16

- Neural Network -

DIVE INTO CODE

Thursday June 24, 2021

DIOP Mouhamed

Agenda

1. Check-in
2. How to proceed
3. Quick Review
4. Neural Network
5. Assignment
6. Scratch 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 basics of **neural Network** from scratch and know how to handle image data in **MNIST**

Prerequisite knowledge

① Ability to write code to train and estimate using scikit-learn classification models.
② Know the calculation process of classification models (assumption function, objective function, steepest descent method) (1)

(1) Solving sprint4 logistic regression scratch.

Let’s learn the basics of Neural Networks

Quick Review (Ensemble Learning)

1. Blending

A method of evaluation using estimates from multiple models

2. Bagging

A method of dividing the training data and evaluating the estimates of multiple decision tree models

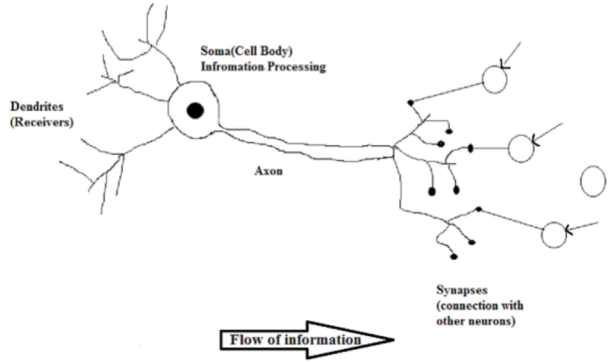
3. Stacking

A hybrid method of blending and bagging. A method that uses multiple models and divides the training data for evaluation

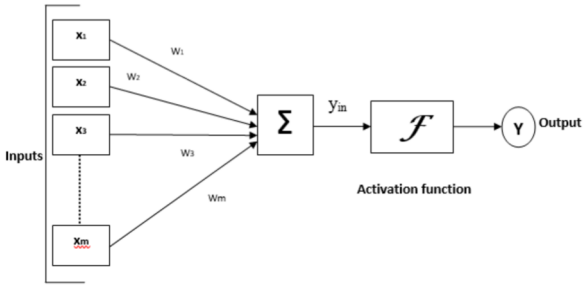
What is a neural network?

A mathematical model that mimics the transmission of information in the brain (nervous system).

When an input stimulus exceeds a threshold, a neuron fires (activates), generating an action potential and transmitting information to other cells. This chain of firings conveys information and achieves computation. The change in the strength of the connections between neurons is called learning.



On the other hand, in a neural network, the input vector is mapped to another vector space using weights (joint load) and activation functions. This is done in a chain of operations to extract information. The adjustment of the weights is called learning.
https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_basic_concepts.htm



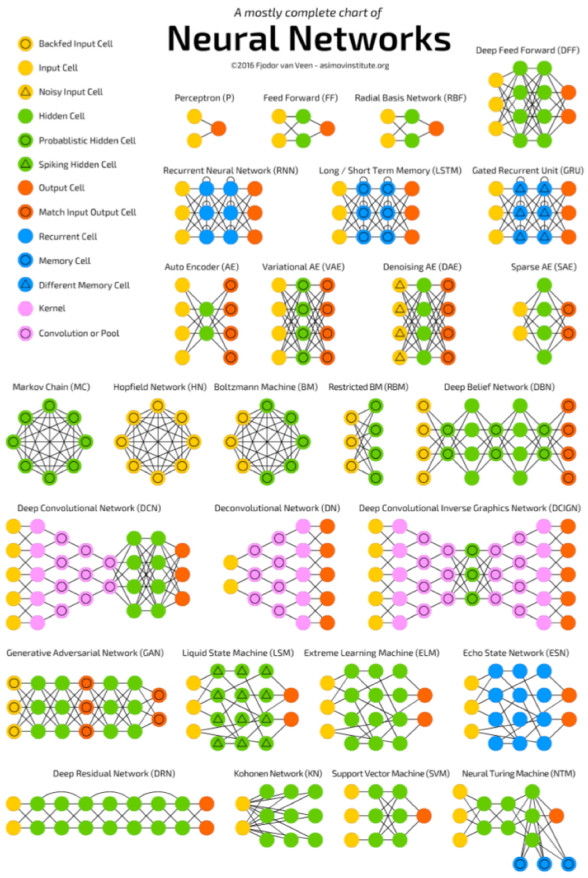
What is a neural network?

Configuration of a neural network

① A processing unit is called a node (neuron), and a set of nodes creates a hierarchical structure.

② Make the output of one layer the input of another layer

③ Have an activation function that changes the distribution of the output



What is a neural network?

History of Neural Networks

1943 -The first mathematical model of neural networks Walter Pitts and Warren McCulloch
1950 - Predictions of machine learning Alan Turing
1952 -Father of machine learning (named "machine learning") Arthur Samuel
1957 -proposed the basis for deep neural networks (created the perceptron in hardware) Frank Rosenblatt
1959 -Discovery of simple and complex cells David H. Hubel and Torsten Wiesel
1960 -Control theory Henry J. Kelley
1965 -Created the first deep learning network Alexey Ivakhnenko and V.G. Lapa
1979-80 - How ANNs Recognize Visual Patterns Kunihiko Fukushima
1982 -Creating the Hopfield Network John Hopfield
1985 -Programs Learn Pronunciation of English Words Terry Sejnowski
1986 - Improving Shape Recognition and Word Prediction David Rumelhart, Geoffrey Hinton, and Ronald J. Williams
1989 - Machine reads handwritten numbers Yann LeCun
1989 - Q-learning Christopher Watkins
1993 - Achieving "very deep learning" tasks Jürgen Schmidhuber
1995 - Support Vector Machines Corinna Cortes and Vladimir Vapnik
1997 - Long-term short-term memory Jürgen Schmidhuber and Sepp H
1998 - Gradient-based learning Yann LeCun
2009 - ImageNet Fei-Fei Li
https://www.import.io/post/history-of-deep-learning/

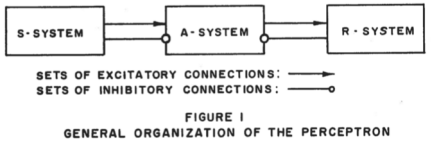


FIGURE I
GENERAL ORGANIZATION OF THE PERCEPTRON

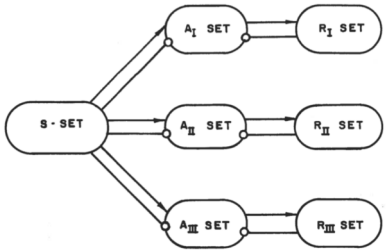
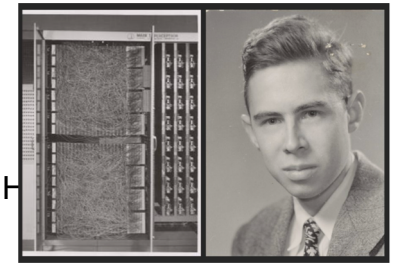


FIGURE 2
ORGANIZATION OF A PERCEPTRON WITH THREE INDEPENDENT OUTPUT-SETS



"The Perceptron: A Perceiving and Recognizing Automaton"
https://www.import.io/wp-content/uploads/2017/06/rosenblatt-1957.pdf

Implementation Flow

Know the problem setup of neural networks.

① Determine the initial value of the weight (combined load).
② Create a fully coupled layer forward propagation (assumed function + activation function)
③ Create an objective function (cross-entropy error)
④ Create a full coupling layer backpropagation
⑤ Estimate.

Dataset

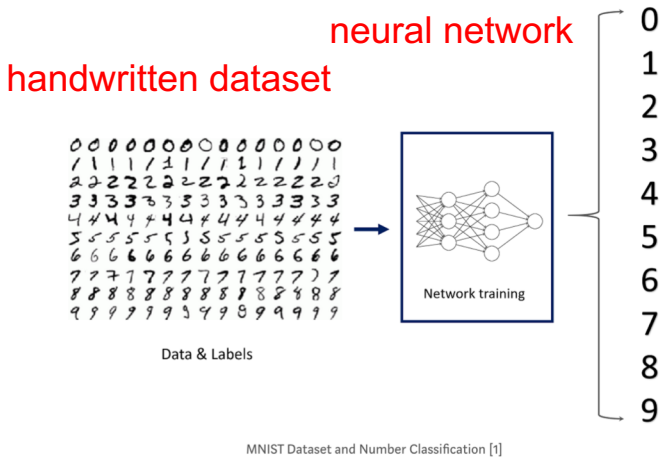
A handwritten character data set (MNIST data set) is input to a neural network, and multi-class classification of 0-9 (numbers) is performed.

Idea: (y_train: about the correct answer label)

For two-class classification.
For two-class classification: use the correct label for one sample as a scalar (int type), such as 0 or 1, for training.

In the case of multi-class classification
For multi-class classification: Given the correct label for a single sample, convert it to a one-hot vector with the length of the class and use it for training.

Let's print DIVER's y_train_one_hot [0] and check it out!



MNIST Dataset and Number Classification [1]

Dataset

Reshape the image data and convert it from (3D shape) to (2D shape).

Idea: (X_train: About image data)

A handwritten character data set is downloaded in the form of a 3D SHAPE (1,28,28) (1).

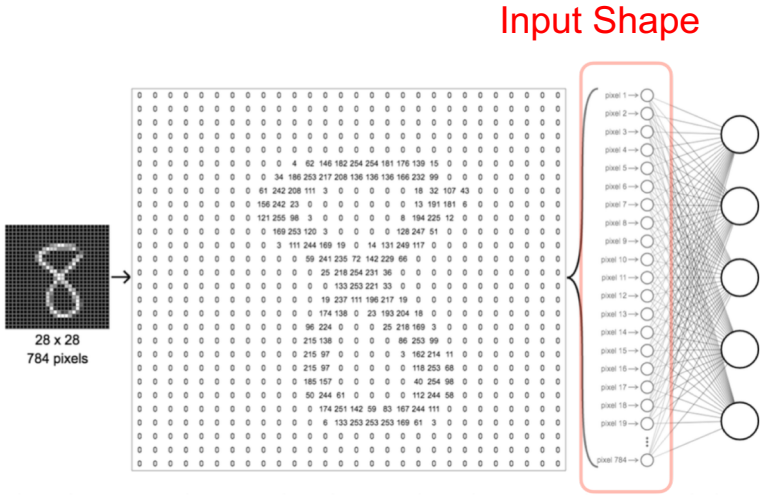
This is converted to 2D shape (1,784) data for input to the neural network.

In addition, the following normalization process is performed on the feature values.

X_train /= 255.0

X_test /= 255.0

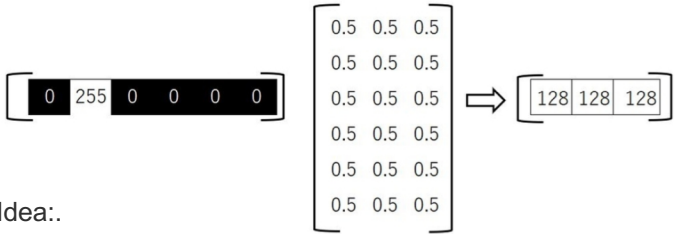
(1) The number of samples here is considered to be 1.



Linear transformation by weight (linear load)

Have an intermediate layer between the input and the output. The more outputs from the input to the intermediate layer, the more expressive the model becomes.

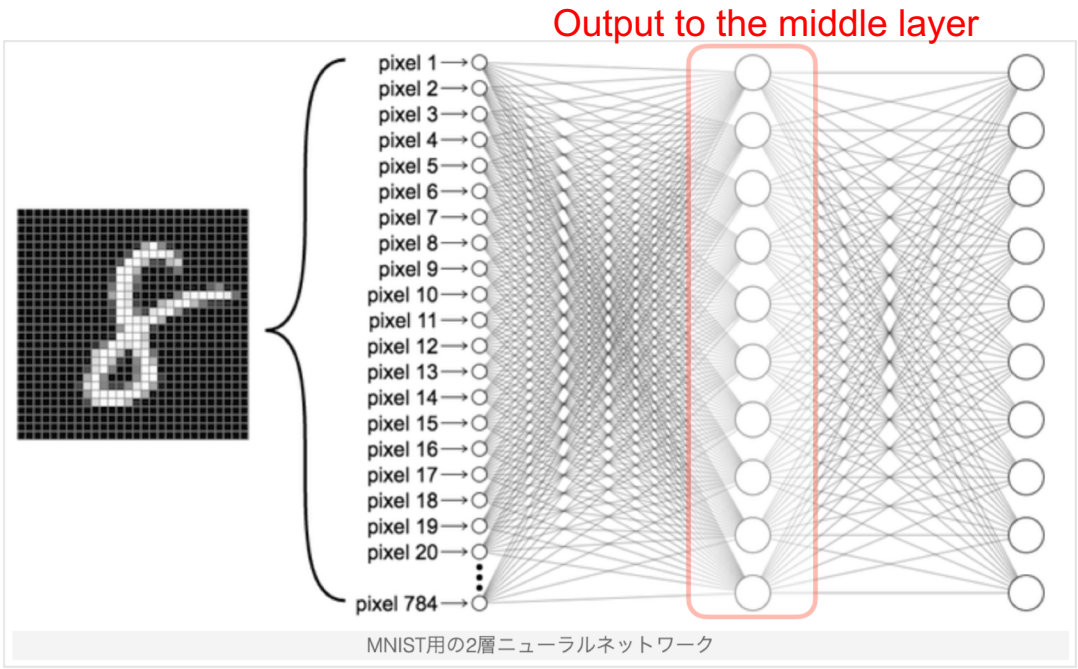
Recall that the number of features in the output to the next stage of stacking (Out of Fold) in **Sprint 7** corresponded to the number of model types (M) that we prepared. The expressiveness will be dealt with in **Sprint 9.**



Idea:.
To increase the number of outputs to the intermediate layer to two or more, multiply the input by a weight matrix (not a weight vector). Also, add up the biases for the number of outputs in the intermediate layer. This operation is called linear transformation and translation (in the context of machine learning, both together are often called linear transformation).

Examples:.
https://arakan-pgm-ai.hatenablog.com/entry/2018/11/05/090000

Assumption: (To begin with)
The linear combination in linear regression was the product of the weight vector and the translation by bias with respect to the input.



Output of the intermediate layer

If we focus on just one output to the middle layer, it is the same as a linear combination (weighted sum).

Idea:.

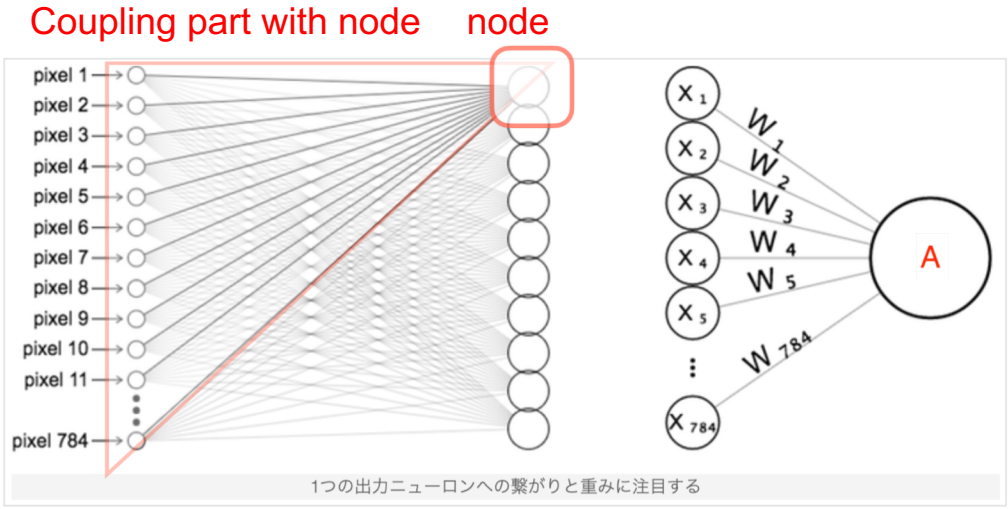
One output to the middle layer is expressed in the form of an equation, which is similar to a linear combination (with bias).

Vector W : W0 ,..... , W784

A = W0X0 + W1X1 + W2X2 + W3X3 + ... + W784X784

Assumption.

In a neural network, a single output is called a node (or neuron, or unit) as the basic unit of computation.



non-linear transformation

At each node, the linear transformation is further transformed into a nonlinear transformation by the activation function.

Idea:.

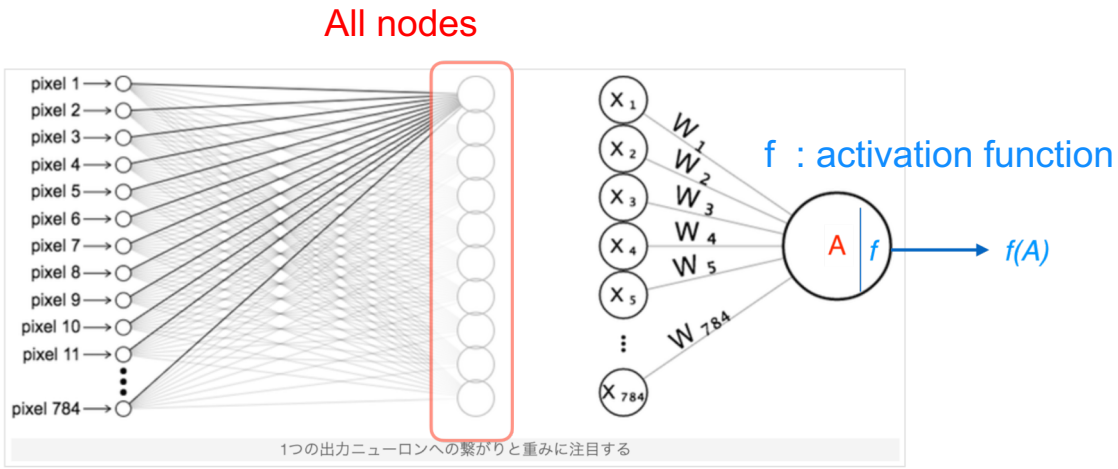
Nonlinear transformation is the process of passing an input that has been linearly transformed by weights through a nonlinear function (f). The nonlinear function in a neural network is called the activation function.

Assumption: (To begin with)

The computation at a single node is similar to the output in logistic regression.

If we consider a neural network with only one layer of architecture (no intermediate layer and the number of output nodes is 1) and the activation function is the sigmoid function, we can consider the whole network as a logistic regression model.

Y = sigmoid(W0X0 + W1X1 + W2X2 + ... + W784X784)

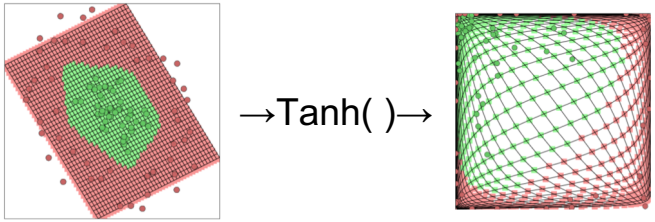


Types of activation functions

Many activation functions monotonically increase the output with respect to the input (2). By using functions with discriminable and sigmoidal properties, we can approximate any (measurable) function.

(2) Monotonically increasing means that if the input is larger, the output will also be larger, so the ranking of the resulting values will not change.

Idea: How do we use the tanh and softmax functions in Sprint?
In this Sprint, we will use tanh function and softmax function.
In multiclass classification, softmax is used as the activation function of the output node. This function returns a value between 0.0 and 1.0, which can be regarded as a probability. Since we are going to classify 10 classes, 10 nodes are placed in the output layer to correspond to each class. Adding up the probabilities between 0.0 and 1.0 from each of these nodes, we get an overall probability of 1. The node with the highest probability corresponds to the predicted class.

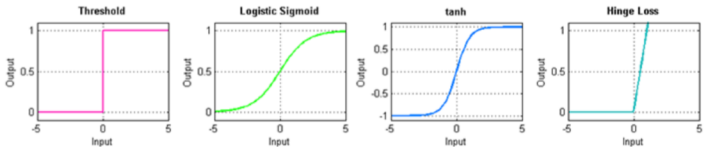


Case study

https://cs.stanford.edu/people/karpathy/convnetjs//demo/classify2d.html

Assumption

The hidden layer performs linear and non-linear transformations and finally outputs a representation such that the data is linearly separable.
Visualize the softmax function. →
https://qiita.com/rtok/items/b1affc619d826eea61fd



Learning with errors

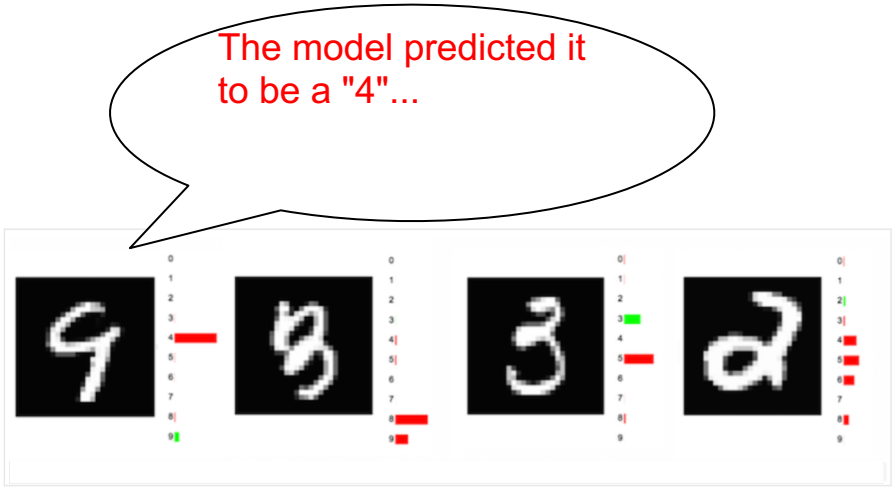
It estimates the error between the true value of the input image and the prediction result, and learns to reduce it.

Idea.

Sum the errors between the predicted value y and the true value y (the correct label) for the class, average them over the batch size (the size of the data to be trained at one time), and evaluate the errors.

Assumptions

In deep learning, as in conventional machine learning, we solve an optimization problem to minimize the objective function. Since this is a classification problem, the objective function for the above process is the cross entropy loss function used before.



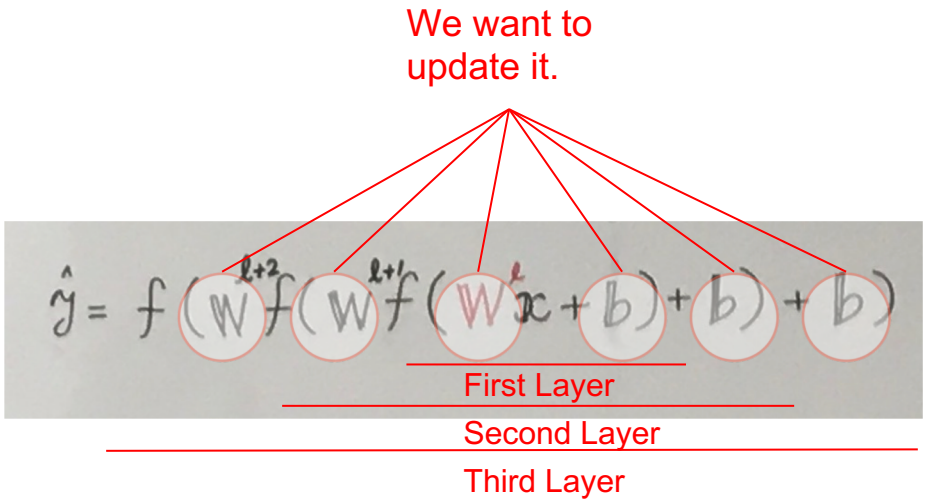
delta rule

In neural networks, updating the weights (joint load) is called learning.

The rule for updating weights during learning is called a learning rule. Although the learning rule in the brain is still unclear, the conditions under which synapses are strengthened are known.
Two famous learning rules in neural networks are the Hebb rule, which mimics the fact that synaptic connections between neurons that fire at the same time are strengthened, and the delta rule, which updates the weights sequentially according to the incremental formula $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \Delta \boldsymbol{w}_t$

Idea:.
For a two-layer neural network with no hidden layers, the gradient could be obtained from the error between the output and the true value using stochastic gradient descent (SGD), and the weights could be updated by the gradient (1960); SGD was then called the Widrow-Hoff method (or delta rule).

The Problem
In a three-layer neural network, the delta rule may be able to calculate the error for the last output, but it cannot calculate the error for the output of the middle layer. Therefore, the gradient of the weights in the middle layer cannot be calculated, and the weights in the middle layer cannot be updated.



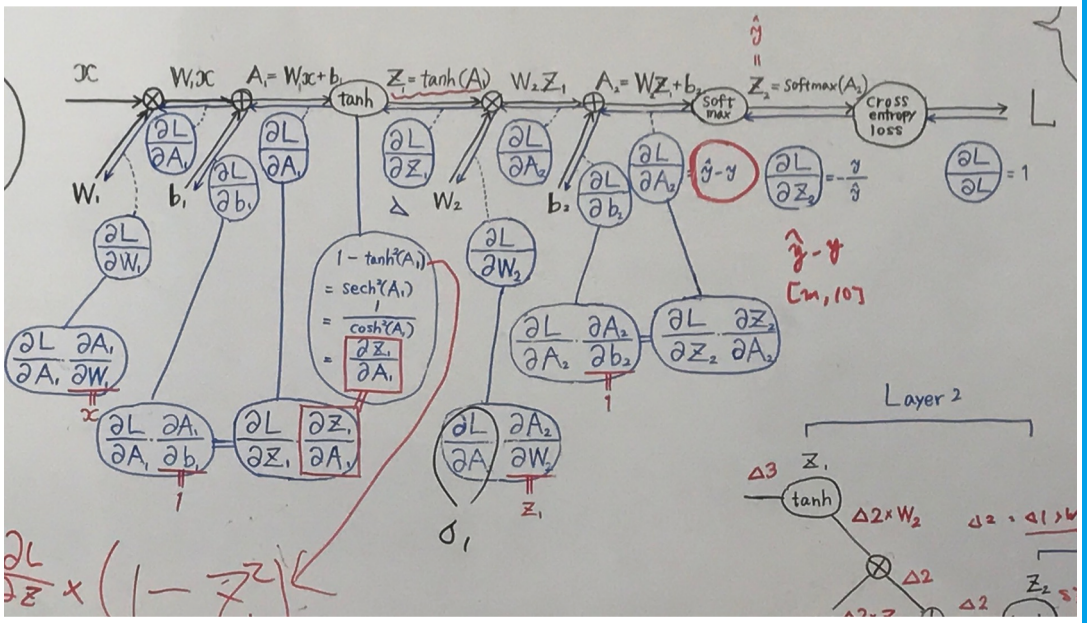
The output of a neural network with three or more layers will be a composite of many functions.

Use the error back propagation method (generalized delta law)

In the error back propagation method, the gradient can be obtained algebraically while letting the error flow in the opposite direction of forward propagation.

1. create weights and bias values with random initial values and find the output of all units (forward propagation up to here)
2. find the error of all output units (back propagation from here)
3. Find the error of all the hidden units.
4. Evaluate the required derivative

If the final output layer is the canonically connected function (3), then the value of the error function differentiated by this final output layer unit will be in the form (y - y) *z1, where z1 is the difference between the final output and the true value.
Therefore, the error at the start of the propagation of error back propagation should be y - y and the propagation should start.
(3) A kind of connected function (inverse function of activation function).
https://www.slideshare.net/satoshikawamoto77/prml-436



Differentiation of composite functions

The derivative of the composite function is calculated using the chain rule.

The chain rule is an equation that determines the derivative of a composite function by the product of the partial derivatives of each function. The chain rule of differentiation allows us to find the error (portion) of each layer. The obtained error is shed in the reverse direction of forward propagation.

Example of partial differentiation of a composite function

$$\frac{d((f \circ g)(x))}{dx} = \frac{d(f(g(x)))}{d(g(x))}\frac{d(g(x))}{dx}$$

$z = f(y), y = g(x)$ If we assume that

$$z = f(y), y = g(x)$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

local gradient

Gradient for updating

Upstream gradient delta

$$y = f(u), u = g(x), y = f(g(x)) \ldots (1)$$

$$\frac{dy}{dx} = \frac{dy}{du} \bullet \frac{du}{dx} \ldots (2)$$

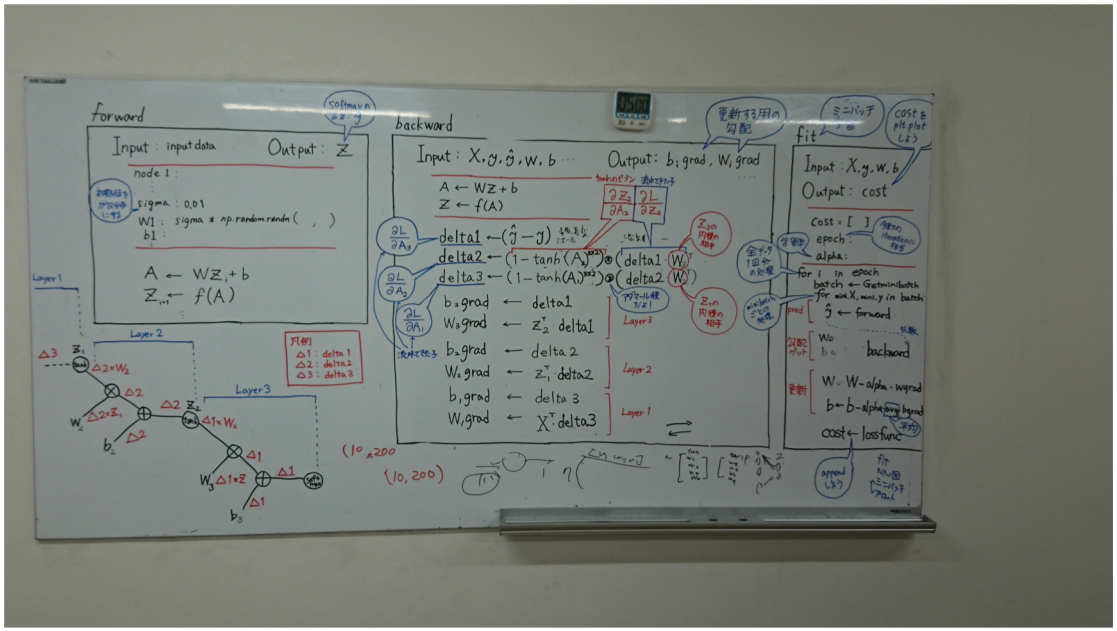
$$z = f(y1, y2, \cdots, ym), yi = gi(x1, x2, \cdots xn) \ldots (3)$$

$$\frac{\partial z}{\partial xi} = \frac{\partial f}{\partial y1} \bullet \frac{\partial y1}{\partial xi} + \frac{\partial f}{\partial y2} \bullet \frac{\partial y2}{\partial xi} + \cdots + \frac{\partial f}{\partial ym} \bullet \frac{\partial ym}{\partial xi} \ldots (4)$$

Tracking deltas with computational graphs

Considering the computational graph, the delta delta (upstream gradient) flows in the opposite direction of the forward propagation.

In the quasi-propagation phase, in the dot product phase, the dot product operation is performed on the incoming delta delta with the value that was originally the counterpart of the dot product calculation to obtain the gradient for updating. On the other hand, in the phase that was an addition in the forward, the delta delta that flowed in is directly used as the gradient for updating. Furthermore, in the phase of the activation function in the forward, the new delta delta is obtained by performing the adamantine product of the derivative of the activation function with the flowing delta delta.



What is a neural network?

Points for solving the 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)

Sample code

How to solve problems

“Scratch Neural Network”

[Problem 1] Create a code to determine the initial value of the weights
[Problem 2] Implementing Forward propagation
[Problem 3] Implementation of Cross-Entropy Error
[Problem 4] Implementation of backpropagation
[Problem 5] Estimation
[Problem 6] Learning and Estimation
[Problem 7] Plotting the Learning Curve
[Problem 8] (Advance Problem) Checking for misclassification

Sprint 9 – Neural Network

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

Sprint 9 – Neural Network

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

Sprint 9 – Neural Network

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

Sprint 9 – 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 1, 2021 19:30~20:30

🚨ToDo: Deep Neural Network
https://diver.diveintocode.jp/curriculums/1877

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