Lecture G1 Functional Approximation 1

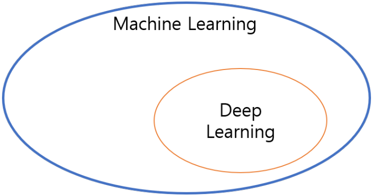
Sim, Min Kyu, Ph.D., mksim@seoultech.ac.kr

SEOULTECH 서울과학기술대학교 데이터사이언스학과

I. Introduction & Motivation

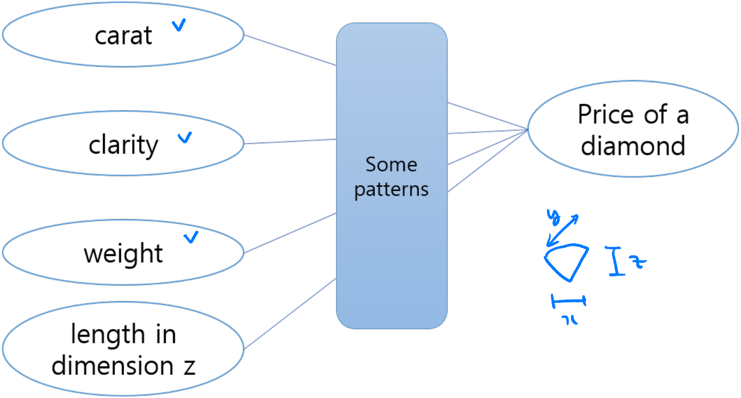
About

- Machine Learning (ML)
  - 'ML is a field of computer science that gives computers the ability to learn without explicitly programmed.' - Wikipedia
  - 'Mathematics is to find patterns.' – R. Feynman
  - My take is, ML is to let the machine to find mathematical patterns
    1. without explicitly programmed
    2. but within the boundary of possible patterns programmed by the researcher.
- Deep Learning (DL)
  - 'DL is part of a broader family of ML Methods.' - Wikipedia
  - DL uses a cascade of multiple layers of nonlinear processing units, i.e. 'deep' neural networks.

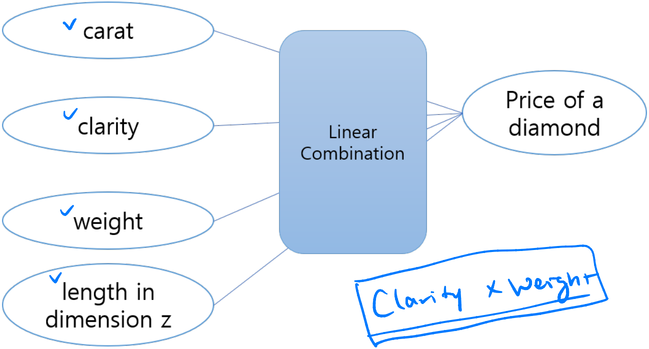


Illustration

"Mathematics is to find patterns." – R. Feynman



Linear regression



Clarity × Weight

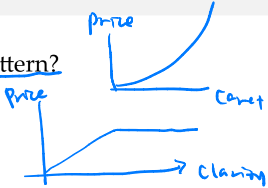
- $Price = \alpha + \beta_{carat} Carat + \beta_{clarity} Clarity + \beta_{weight} Weight + \beta_{len\_z} Len\_z$

↑10%

- Assumption
  1. Linearity (proportional output change given input change)
  2. Independence (no interaction among input variables)

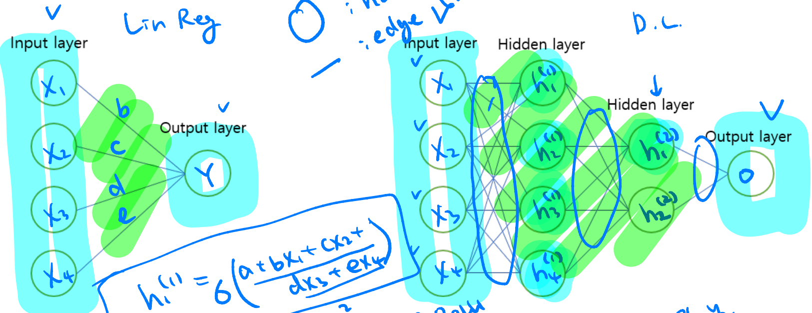
Sufficiency of linear regression?

- What if the assumptions are not sufficient to find the pattern?
  1. Nonlinearity
     - Price is indeed known to be proportional to square of carat.
     - After the certain level of clarity, it no longer matters.
  2. Interaction
     - Clarity matters more when carat is bigger.
     - Weight is negatively correlated to clarity.
     - Weight is positively correlated to carat.



Linear regression vs Deep neural network

○ : node
— : edge (weight)



$$h_1^{(1)} = \sigma(a + bx_1 + cx_2 + dx_3 + ex_4)$$

$$\sigma(x) = \begin{cases} x^2 \\ \max(x, 0) \text{ ReLU} \end{cases}$$

$x.\ y.$

$ax+by$

$ax+by+c$

Linear Reg.	Deep Learning
Linear	Nonlinear
Independence	Allows dependence
Intuitive ✓	Less intuitive ✓

II. Mechanism of deep learning

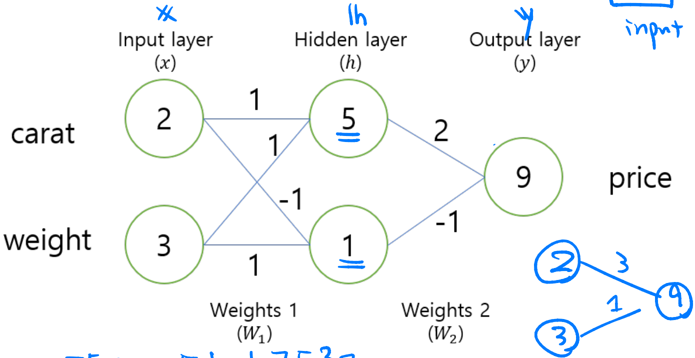
Mechanism anatomy

1. Forward propagation - *the backbone of the network*
2. Activation function - *the essential elements for non-linearity*
3. Loss function - *the performance criteria of your estimate*
4. Gradient descent - *feedback for improvement*
5. Back propagation - *how feedback is propagated*

1. Forward propagation - *the backbone of the network*

$\boxed{2} \rightarrow 2$

input



- $\mathrm{h} = W_1 \mathrm{x}$ $\quad \begin{bmatrix}5\\1\end{bmatrix} = \begin{bmatrix}1 & 1\\-1 & 1\end{bmatrix}\begin{bmatrix}2\\3\end{bmatrix}$
- $\mathrm{y} = W_2 \mathrm{h}$ $\quad 9 = \begin{bmatrix}2 & -1\end{bmatrix}\begin{bmatrix}5\\1\end{bmatrix}$
- The forward propagation process can be summarized as as follows:

$$\mathrm{x} \xrightarrow{\times W_1} \mathrm{h} \xrightarrow{\times W_2} \mathrm{y}$$

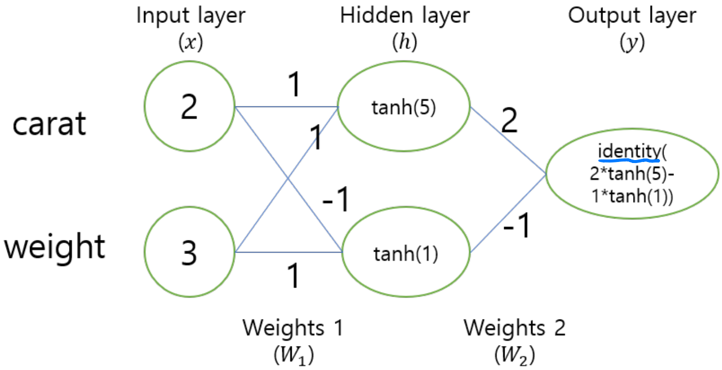
2. Activation function - *the essential elements for non-linearity*

- The previous page's output $9 = \underbrace{\begin{bmatrix}2 & -1\end{bmatrix}\begin{bmatrix}1 & 1\\ -1 & 1\end{bmatrix}}_{\begin{bmatrix}3 & 1\end{bmatrix}}\begin{bmatrix}2\\ 3\end{bmatrix}$
- Why need activation function?
  - Linear transformation of linear transformation is linear. We need some nonlinearity for trivial forward propagation.
  - Output layer needs to be structured for desirable output. For example, we may want it to be in range of $(-\infty, \infty)$, $[0, \infty)$, $[0, 1]$, or so on.
- Activation function is a nonlinear function applied to hidden layers or an output layer.

- For hidden layers
  1. ReLU
     - $(-\infty, \infty) \to [0, \infty)$
     - $f(x) = x^+ = max(x, 0)$
     - Known to be most effective
  2. tanh
     - $(-\infty, \infty) \to (-1, 1)$ time series
     - $f(x) = tanh(x)$

- For output layer
  3. sigmoid
     - $(-\infty, \infty) \to (0, 1]$ binary classification
     - $f(x) = \frac{1}{1+e^{-x}}$
     - Maps to probability space
  4. softmax
     - $(-\infty, \infty)^n \to (0, 1]^n$ multiclass classification
     - $f(x_k) = \frac{exp(x_k)}{\sum_{i=1}^{n} exp(x_i)}$
     - Used for classification problems
  5. linear (identity)
     - $(-\infty, \infty) \to (-\infty, \infty)$
     - $f(x) = x$
     - Used for regression problems ✓

- The previous forward propagation should be modified, for example, as below:



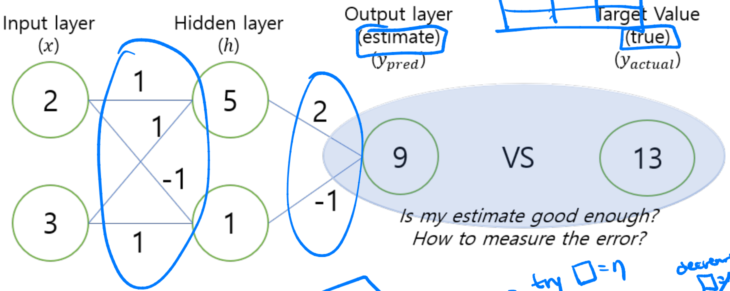
- $\mathrm{h} = tanh(W_1\mathrm{x})$
- $\mathrm{y} = identity(W_2\mathrm{h})$
- The forward propagation process should be revised as follows:

$$\mathrm{x} \xrightarrow{\quad \times W_1 \quad \text{then apply an act. fn.} \quad} \mathrm{h} \xrightarrow{\quad \times W_2 \quad \text{then apply an act. fn.} \quad} \mathrm{y}$$

3. Loss function - *the performance evaluation of your estimates*

- Assuming, for simplicity of our discussion, that activation functions are all identity (linear) functions.

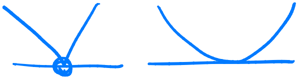
x		y
2	3	13



3 × □ = 19
try □ = 5
19 − 3 × 5 = 4
increase □ = 5
try □ = 7
19 − 3 × 7 = −2
decrease □ = 7

	Estimate $\hat{y}$	True value (target) y	Error $\hat{y}-y$	Squared error (SE) $\hat{y}-y$
Sample 1	10	20	-10	100
Sample 2	8	3	5	25
Sample 3	6	1	5	25

- Loss function measures aggregated error.
- MAD (Mean Average Deviation) $|x|$
  - $loss = \frac{|-10|+|5|+|5|}{3}$ v
  - Though intuitive, not widely used because not differentiable.

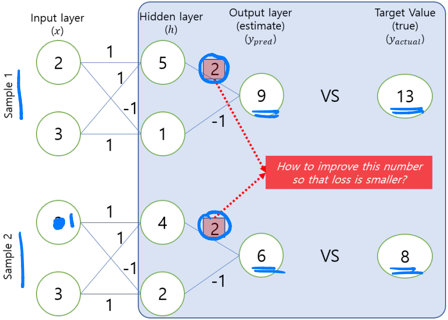


min x^2-4x+3

- MSE (Mean Squared Error)
  - $loss = \frac{100+25+25}{3}$
  - Most common for parametric target value.
  - Penalizes large errors severely.
- Cross Entropy
  - Loss: $E = \sum_k t_k log(y_k)$, where t_k is k-th target and y_k is an k-th estimate.
  - Commonly used for classification problem.

4. Gradient descent - *feedback for improvement*

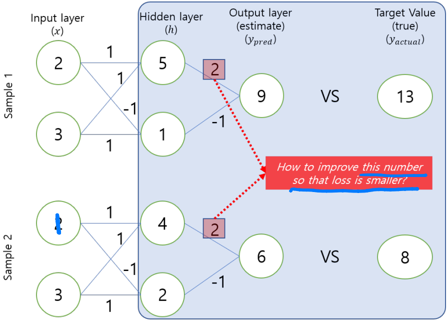
- After measuring loss function, the weights must be improved so that loss becomes smaller.
- Let us confine our attention to W_2, currently $[2\ 1]$, a weight vector for the hidden layer (h). Let us confine further to its first element, currently $\boxed{W_2(1) = 2.}$
- How can we find better value for $W_2(1)$?



$x+y=4$

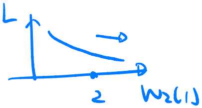
$+\ \underline{-x+y=2}$

$2y=6$



h_1	W_2	$\hat{y}$	y	Error	SE	MSE (L)
	$[0 \;\; -1]$	$-1, 2$		$-14, -10$	$(-14)^2, (-10)^2$	148
	$[1 \;\; -1]$	$4, 2$		$-9, -6$	$(-9)^2, (-6)^2$	58.5
$\begin{bmatrix} 5 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \end{bmatrix}$	$[2 \;\; -1]$	$9, 6$	$13, 8$	$-4, -2$	$(-4)^2, (-2)^2$	10
	$[3 \;\; -1]$	$14, 10$		$1, 2$	$1^2, 2^2$	2.5
	$[4 \;\; -1]$	$19, 14$		$6, 6$	$6^2, 6^2$	36

- The current weight $W_2(1)$ should be increased so that L is decreased. The sign of $\frac{\partial L}{\partial W_2(1)}$ is negative. In other words, if $W_2(1)$ is increased, L is decreased.
- This tells us that, the weight is updated in the following way, where α is often called 'learning rate'.



$$W_2^{new}(1) \leftarrow W_2^{old}(1) - \alpha \cdot \frac{\partial L}{\partial W_2(1)}$$

- This updating is to occur simultaneously for all weights.
- If written in a general sense,

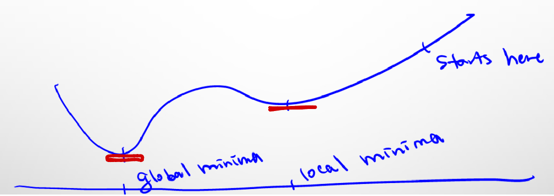
$\nabla_w L$

$$\mathrm{w} \leftarrow \mathrm{w} - \alpha \cdot \nabla_{\mathrm{w}} L,$$

where $\nabla_{\mathrm{w}} L = \left(\frac{\partial L}{\partial \mathrm{w}_1}, \frac{\partial L}{\partial \mathrm{w}_2}, \cdots, \frac{\partial L}{\partial \mathrm{w}_n}\right)$ is first-order derivatives of L with respect to all elements of w. $\nabla_{\mathrm{w}} L$ is geometrically slope. ∇ reads 'nabla'.
- This tells us that both loss function and activation functions must be differentiable. They need to be 1) monotone, 2) continuous, and 3) differentiable.

almost every where

- Optimizer
  - This process of gradient descent is to optimize the weight of neural network. And the algorithm concerning finding learning rate or step size is called 'optimizer'.
  - Widely used optimizer includes adam, RMSprop, SGD, and so on.
- Weight initialization
  - It is not always guaranteed to find global optimum given a poor choice of initial weight.
  - Thus, the initial weights must be randomized within a wide range to sufficiently cover the possible optimum.

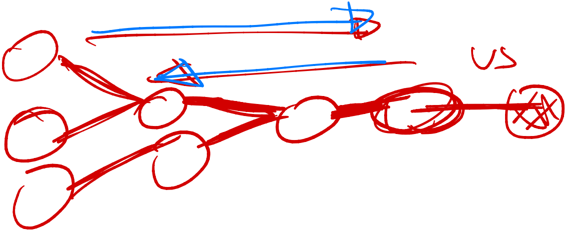


- Batch and Epoch
  - Conducting differentiation involves numerically cost operation, so it is desirable to collect some number of sample and then to do weight updating.
  - The size of this subset of the data for an update once is called **batch**.
  - One time to go through the whole dataset once is called **epoch**.

5. Back propagation - *how feedback is propagated*

- The weight updating, of course, does not only take place on the last hidden layer, but all weights in front.
- The information is propagated based on ‘Chain rule of calculus.’

$$\frac{d}{dx}\boxed{f(g(x))} = \underline{f'(g(x))}\,\underline{g'(x)}$$



Summary of deep learning mechanism

1. Forward propagation
    - Choose the number of layer
    - Choose the number of nodes in each layer
2. Activation function
    - Choose the activation function for hidden layers
    - Choose the activation function for the last layer based on the characteristics of target output.
3. Loss function
    - Choose the loss function b~~e~~ased on the characteristics of target output
4. Gradient Descent
    - Choose the size of batch
    - Choose the the number of epochs
    - Choose the optimizer
    - (Choose default learning rate)
    - (Choose initial weights)
5. Back propagation

Implementation

- The above mechanism describes supervised learning, because we assume the possesion of correct target values.
- The input data needs to be scaled between [-1,1] range, so that weights effectively cover the variation of data.
- The design factors of neural network summarized in the previous page are called hyperparameter.
- To find hyperparameter, split train-validataion-test sets appropriately.

```
"Exceptional people, I have found, either start out being optimistic or learn to be optimistic
because they realize that they can't get what they want in life without being optimistic.
- B. Rotella in How Champions Think: In Sports and in Life"
```