Mathematical Intuition of Deep Learning

Wade Copeland

08 October, 2024

Why Learn the Mathematics of Deep Learning?

Learning the math behind a statistical method is like learning the secret to a magic trick. Once you know the math, you can never be fooled by the same trick again.

Applications to the Oversight Mission

- Each IG is responsible for the oversight of agencies that will implement deep learning to make more informed and better decisions, but come with risks such as bias, privacy, lack of transparency, and ethical complexities.
- Each IG will use deep leaning to provide better oversight.
  - Auditors will use deep learning to write better recommendations based on data about how clients responded to previous recommendations.
  - Evaluators will use deep learning to find evidence of systemic problems using large volumes of data.
  - Investigators will use deep learning to find leads based on patterns previously observed their case data.

The Deep in Deep Learning

- Deep Learning always refers to Deep Neural Networks.
- A Deep Neural Network is a Neural Network with more than two layers.
- ChatGPT Version 4 is reported to have 1.8 trillion parameters across 120 layers[1]!

Preliminaries

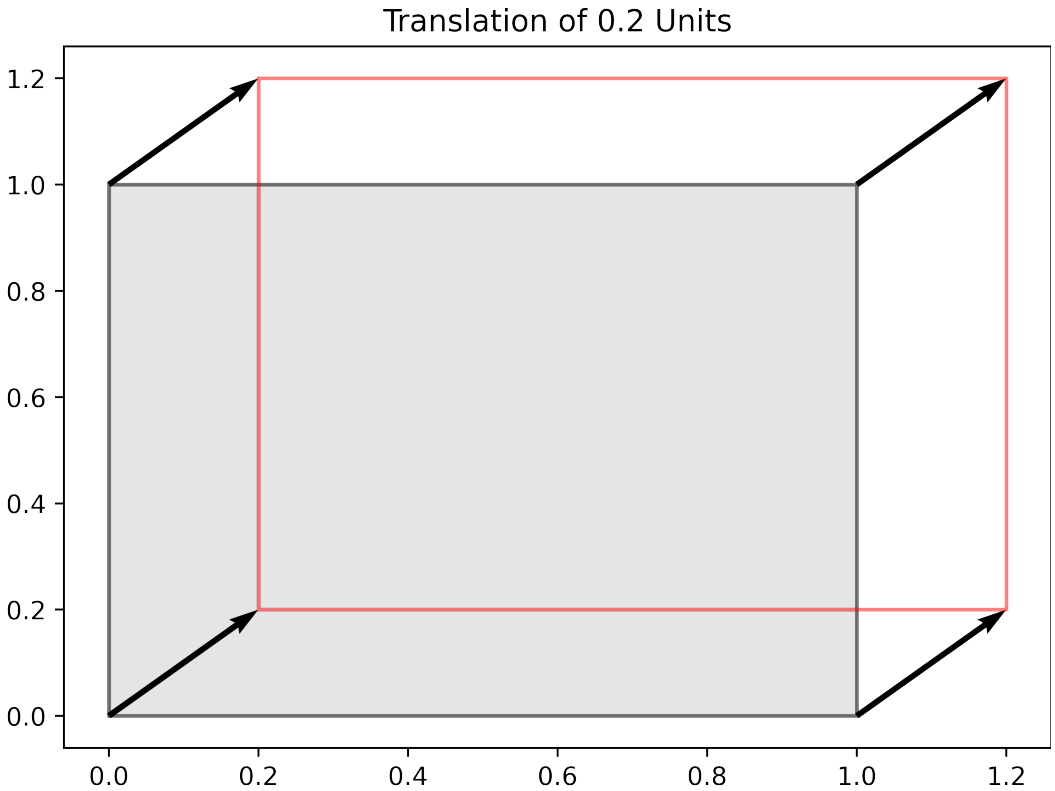
- This presentation is aimed at the someone who wants to learn about Deep Learning without having a background in sophisticated mathematics.
- To that end, we do need to briefly introduce some concepts from *Linear Algebra* and *Differential Calculus*, but will focus on an intuitive understanding.

Matricies

- A matrix is a grid of numbers with N rows and P columns denoted $X_{N \times P}$.
  - $X_{2 \times 2} = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix}$ is a 2×2 matrix.
  - $X_{1 \times 1} = \begin{bmatrix} x_1 \end{bmatrix}$ is a 1×1 matrix, also called a *scalar*.
  - $X_{1 \times 2} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$ is a 1×2 matrix, also called a *vector*.

Matrix Translation

- Matrix translation is the addition/subtraction of a constant to every element of a matrix.
  - The columns of $X_{2\times 6} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$ describe a square in 2-dimensional space. We can translate the square by 0.2 units by adding 0.2 to each element of $X_{2\times 6}$.

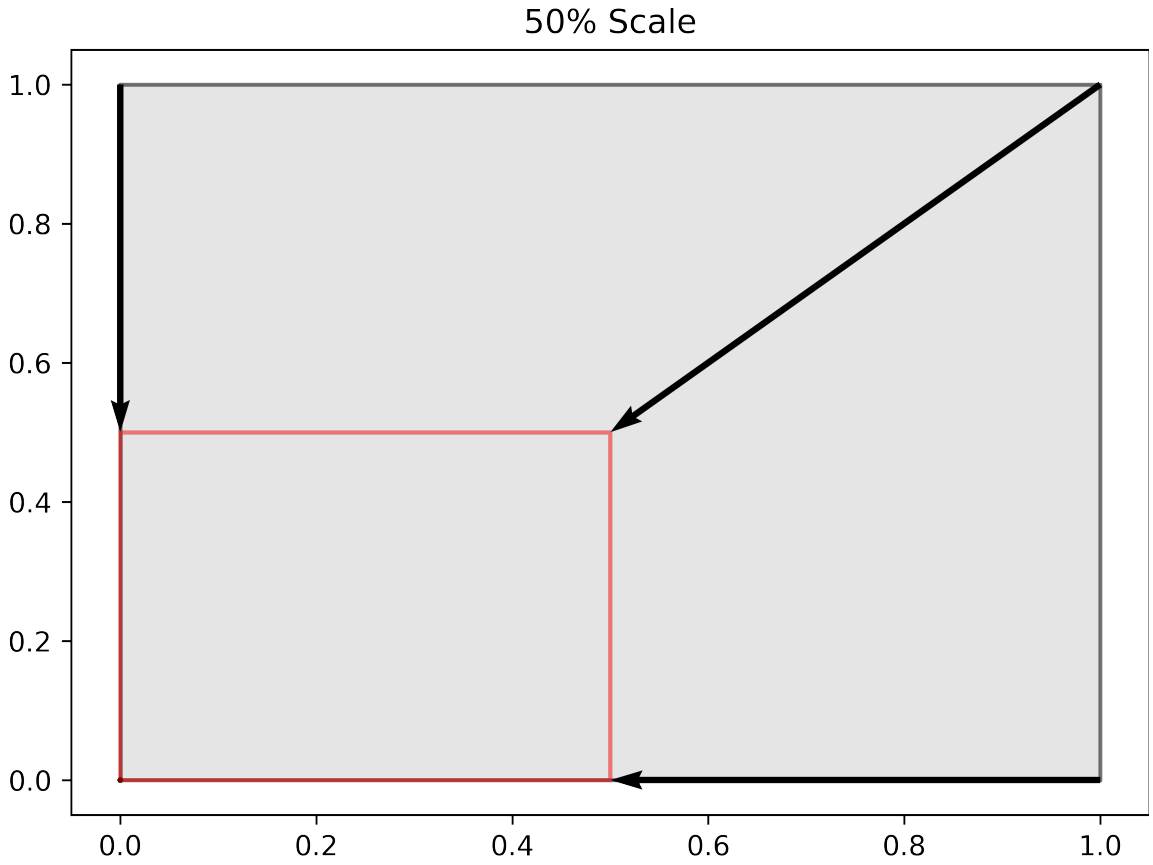


Matrix Transformation

- Matrix transformation is achieved through the matrix-specific operation called the *dot product* where the rows of X and the columns of Y are multiplied and summed.
  - The dot product of
    $$X_{1\times 2} \cdot Y_{2\times 1} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 y_1 + x_2 y_2 \end{bmatrix}$$
  - If the number of columns in X does not equal the number of rows in Y then the dot product operation is not defined.
  - The size of the resulting matrix is equal to the number of rows in X and the number of columns in Y.

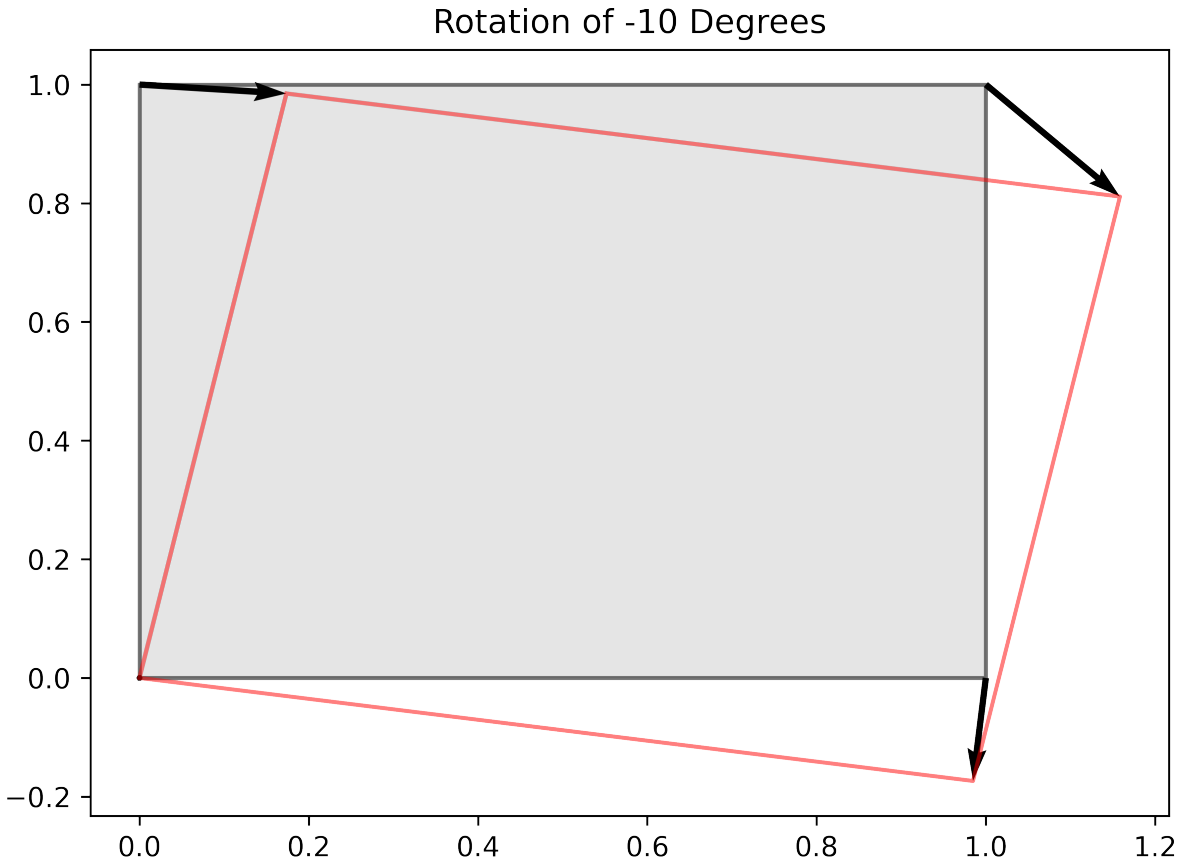
Matrix Transformation: Scale

- $X_{2\times 6}$ can be scaled by 50% taking the dot product of $\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$ and $X_{2\times 6}$.



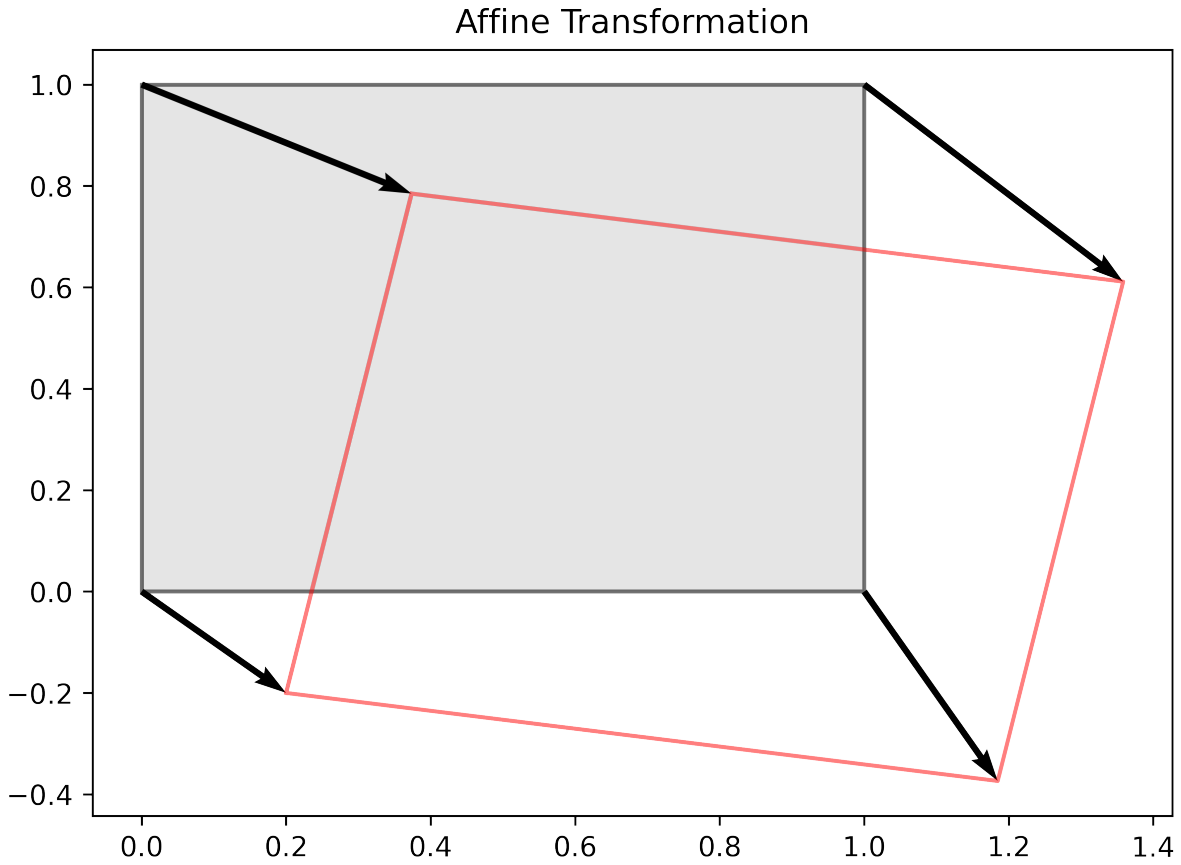
Matrix Transformation: Rotation

- $X_{2\times 6}$ can be rotated by negative 10 degrees by taking the dot product of $\begin{bmatrix} cos(0.175) & -sin(0.175) \\ sin(0.175) & cos(0.175) \end{bmatrix}$ and $X_{2\times 6}$.



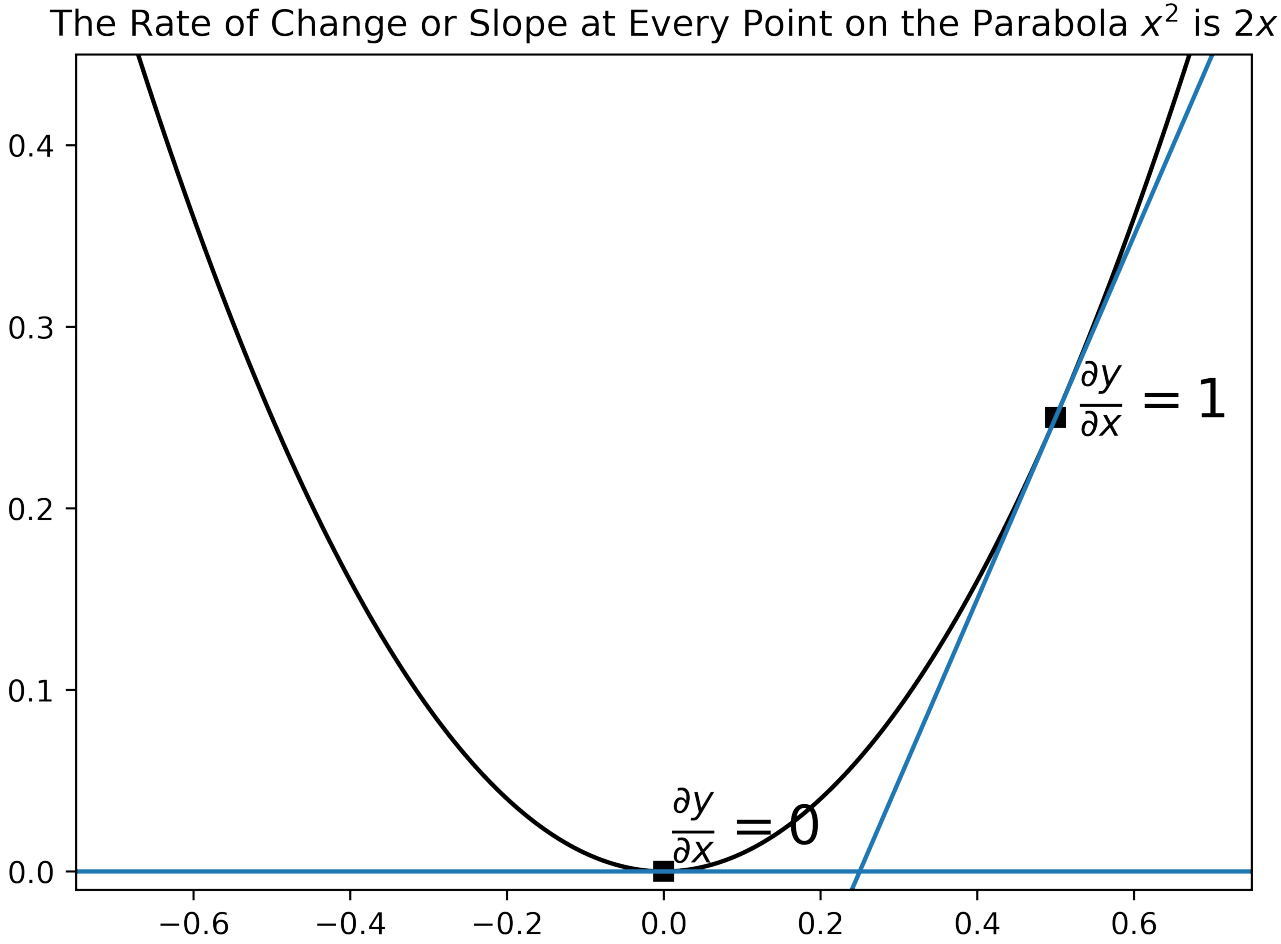
Matris Transformation: Affine

- An affine transformation of $X_{2\times 6}$ combines transformation and translation. For example, combining the rotation transformation of -10 degrees and translation of 0.2 units.



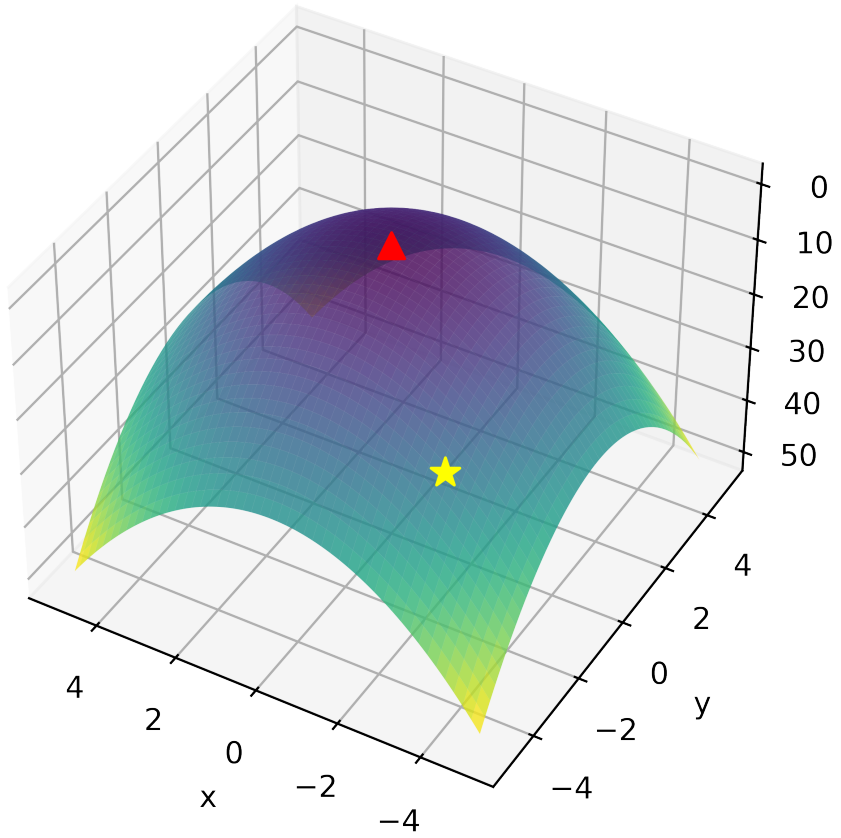
Rate of Change: The Derivative

- A derivative measures the rate of change or slope at any point denoted by $\frac{\partial y}{\partial x}$, which is read "the change in y with respect to the change in x."



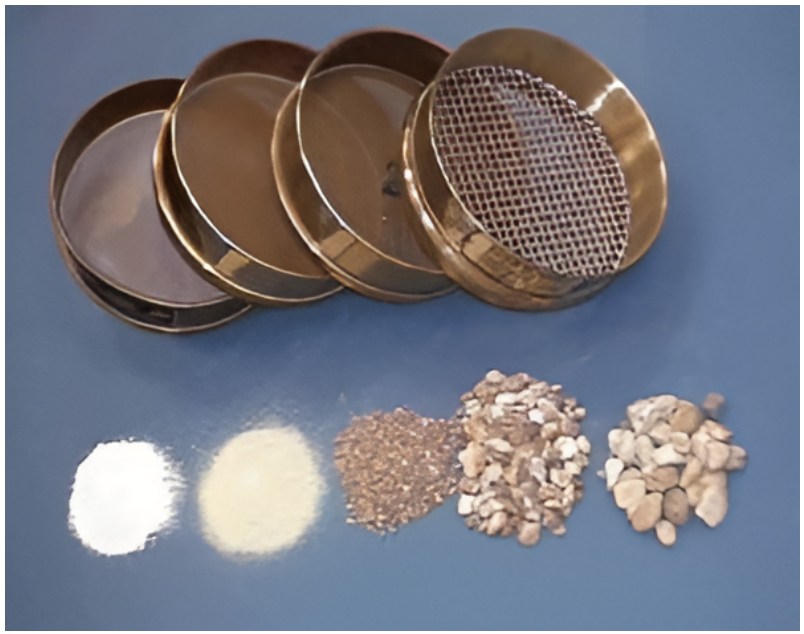
Rate of Change: The Gradient

- The gradient is a generalization of the derivative that measures the rate of change or slope in every direction. For example, if we have the surface $z = x^2 + y^2$ the slope in the x direction is $\frac{\partial z}{\partial x} = 2x$ and the slope in the y direction is $\frac{\partial z}{\partial y} = 2y$.



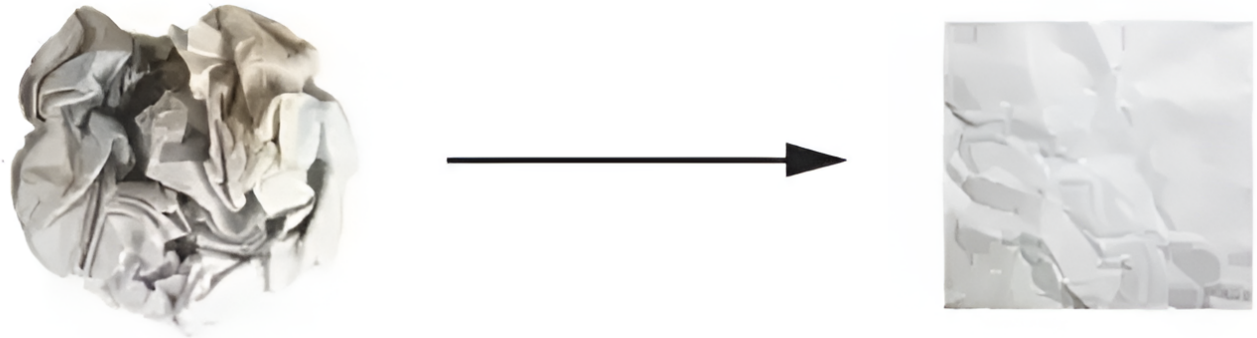
Deep Learning Intuition I

- A Neural Network is an information sieve that takes as input the outcomes, and the features we want to use to predict the outcomes[2].
- At the top of the sieve is the most granular representation of the data, and for each successive layer down, the data becomes more and more refined, that is, better at predicting the outcome[2].



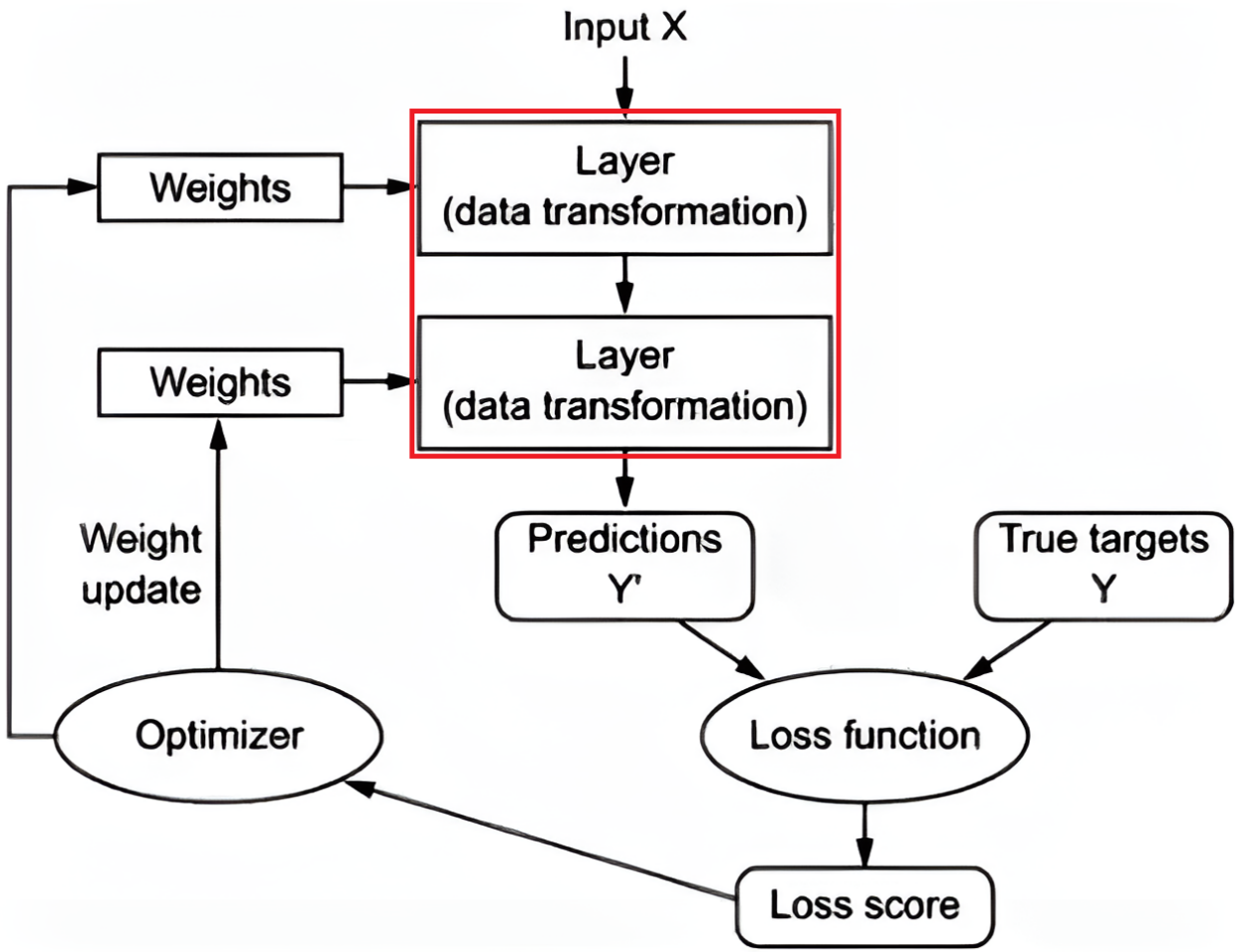
Deep Learning Intuition II

- Consider a blue and red piece of paper crumpled together and your goal is to uncrumple them. Each movement of your fingers to uncrumple the papers is like a layer of a Neural Network, until finally at the last layer the colors are fully separated[2].



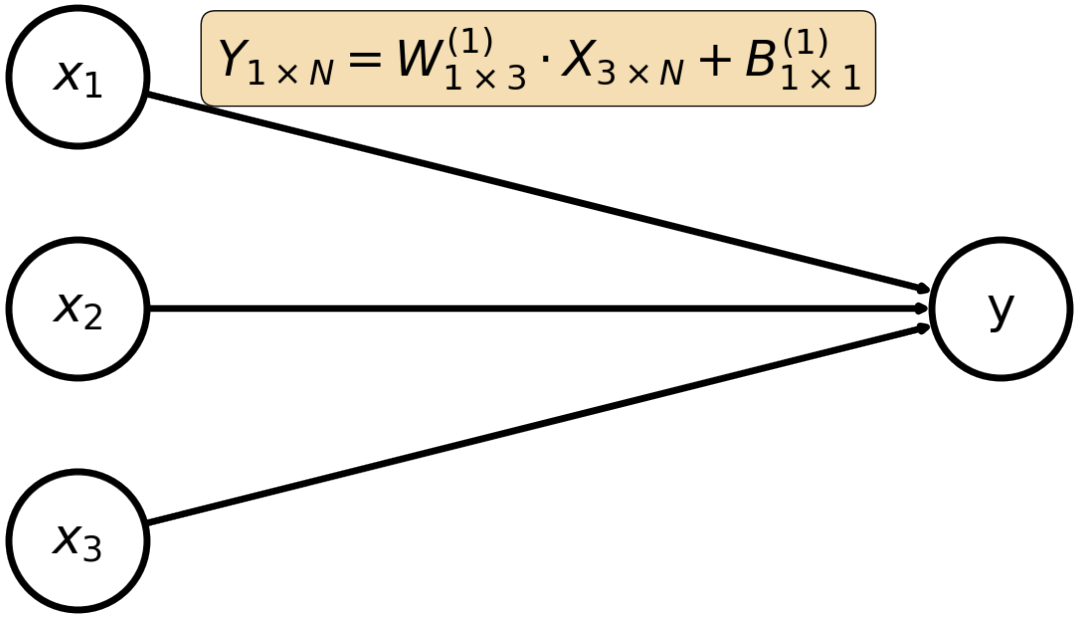
Deep Learning Roadmap: Layers

- The full process of fitting a Neural Network is shown below. As a first step, we will explain the mathematical representation of the layers[2].



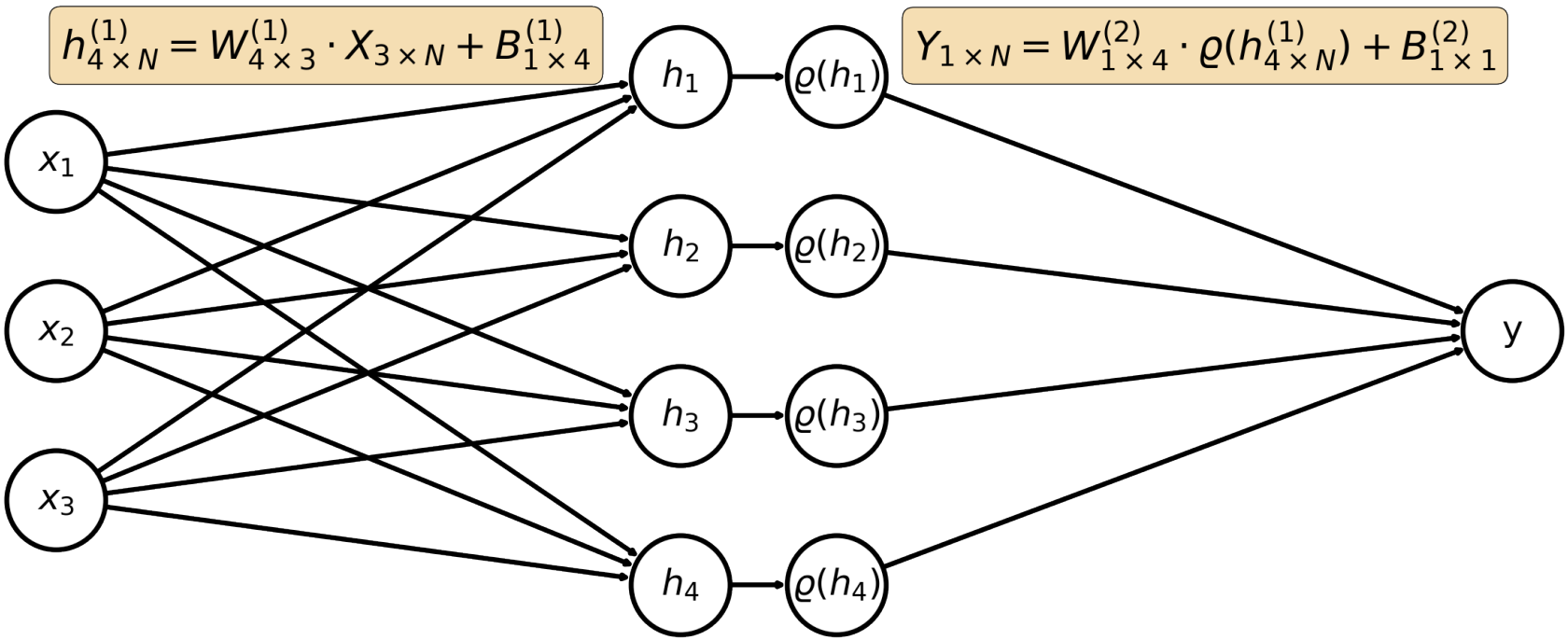
Representation of a Single-Layer Neural Network

- A single layer Neural Network is represented by the affine transformation $Y = W \cdot X + B^3$.



Representation of a Two-Layer Neural Network

- A two layer Neural Network includes a layer of *neurons* between the features and the outcome. These are called *hidden neurons* because they are not observed in the input data[3].



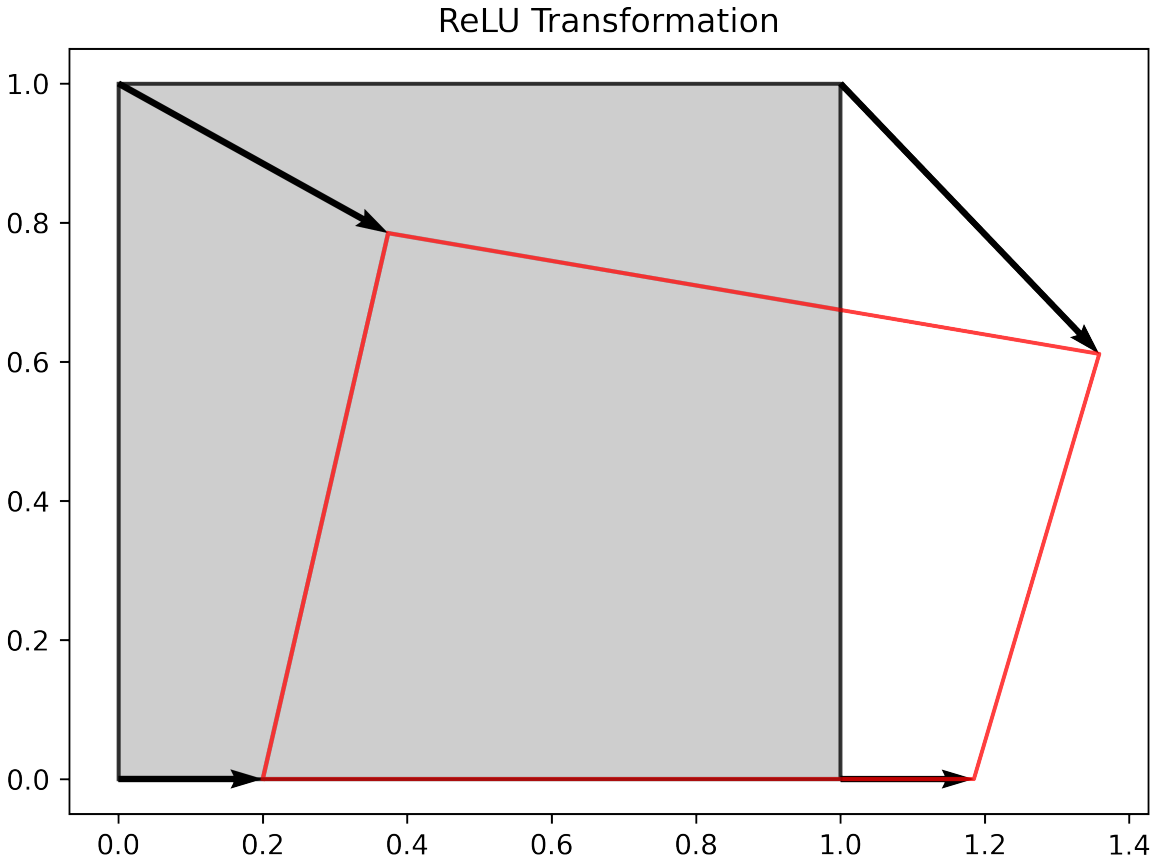
What Is $\varrho(h)$ and Why Do We Need It?

- The function $\varrho(h)$ is a non-linear transformation applied to the otherwise linear affine transformation[3].
- A two-layer Neural Network without a non-linear transformation between layers is a single-layer neural network in disguise[2]!

$$
\begin{aligned}
Y_{1\times N} = \\
W^{(2)}_{1\times 4} \cdot h^{(1)}_{4\times N} + B^{(2)}_{1\times 1} = \\
W^{(2)}_{1\times 4} \cdot (W^{(1)}_{4\times 3} \cdot X_{3\times N} + B^{(1)}_{1\times 4}) + B^{(2)}_{1\times 1} = \\
(W^{(2)}_{1\times 4} \cdot W^{(1)}_{4\times 3}) \cdot X_{3\times N} + (W^{(2)}_{1\times 4} B^{(1)}_{1\times 4} + B^{(2)}_{1\times 1})
\end{aligned}
$$

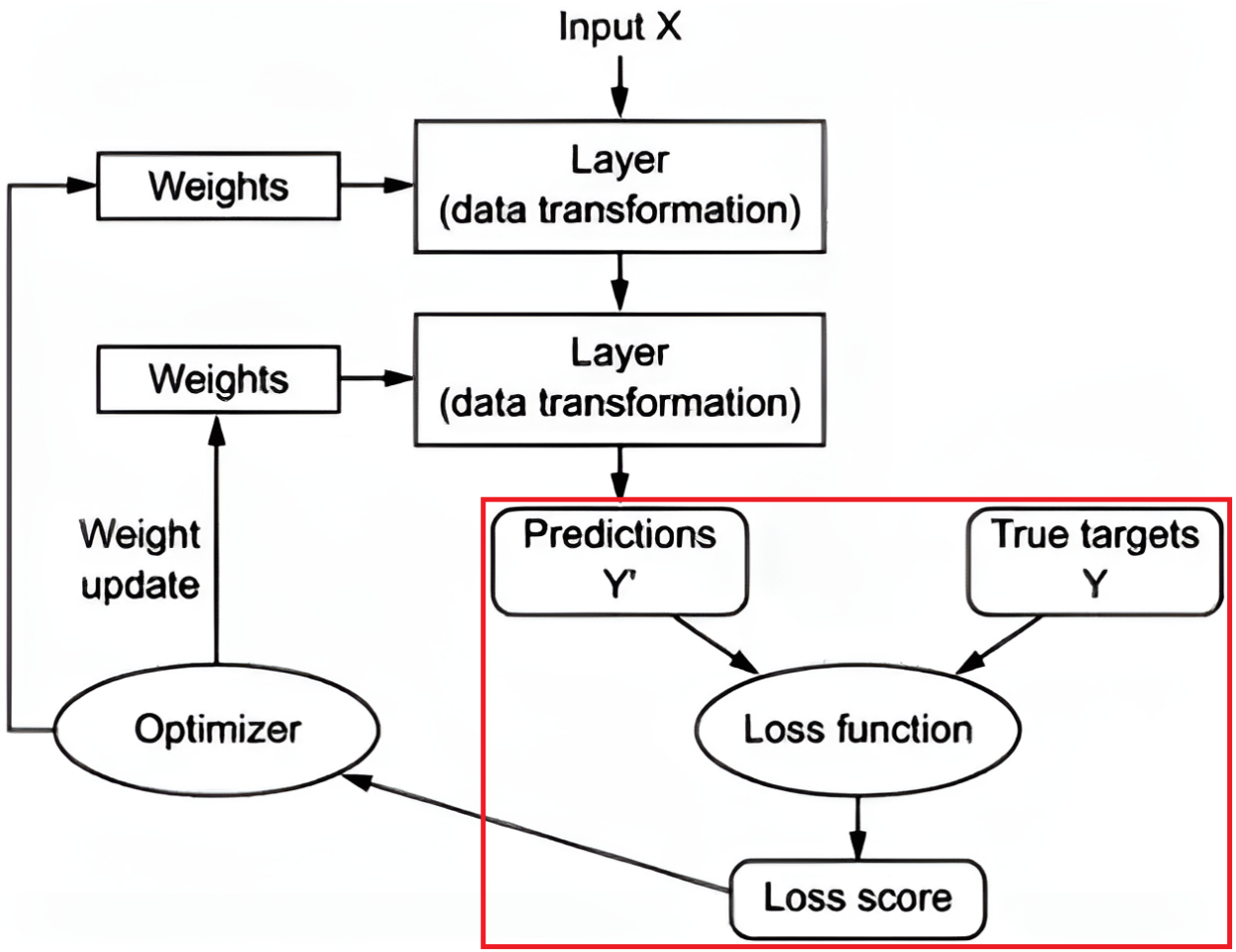
Rectified Linear Unit (ReLU)

- A common $\varrho(h)$ is the *ReLU* transformation which replaces all negative values with 0[3].



Deep Learning Roadmap: Loss

- After a Neural Network is fit, we need a way to measure how good it is at predicting the target or *outcome* of interest. This is called the *loss function*[2].



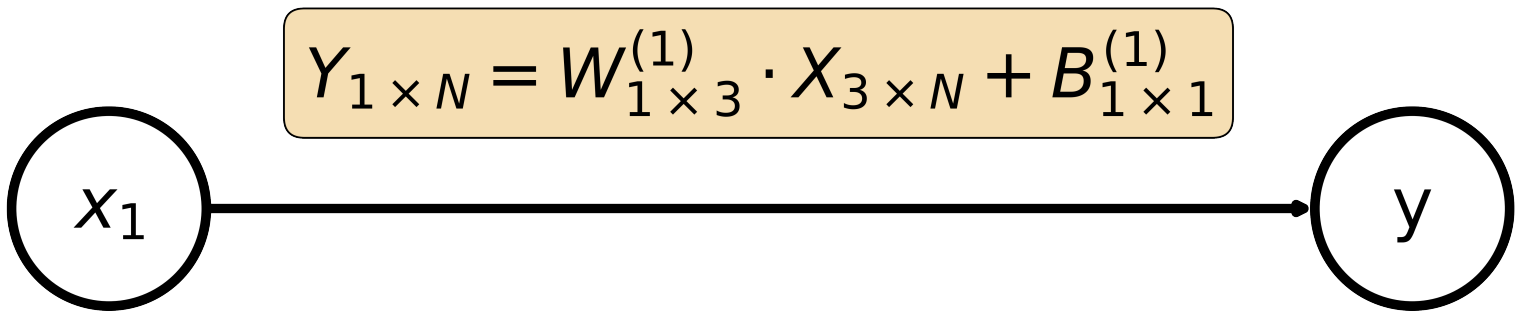
Mean Squared Error

- The most common measure of loss is the *Mean Squared Error*, which measures the average of the squared deviations from the observed outcomes and our prediction of them[2].

$$e = \frac{1}{n}\sum_{i=1}^{n}(y - y')^2$$

Forward Pass I

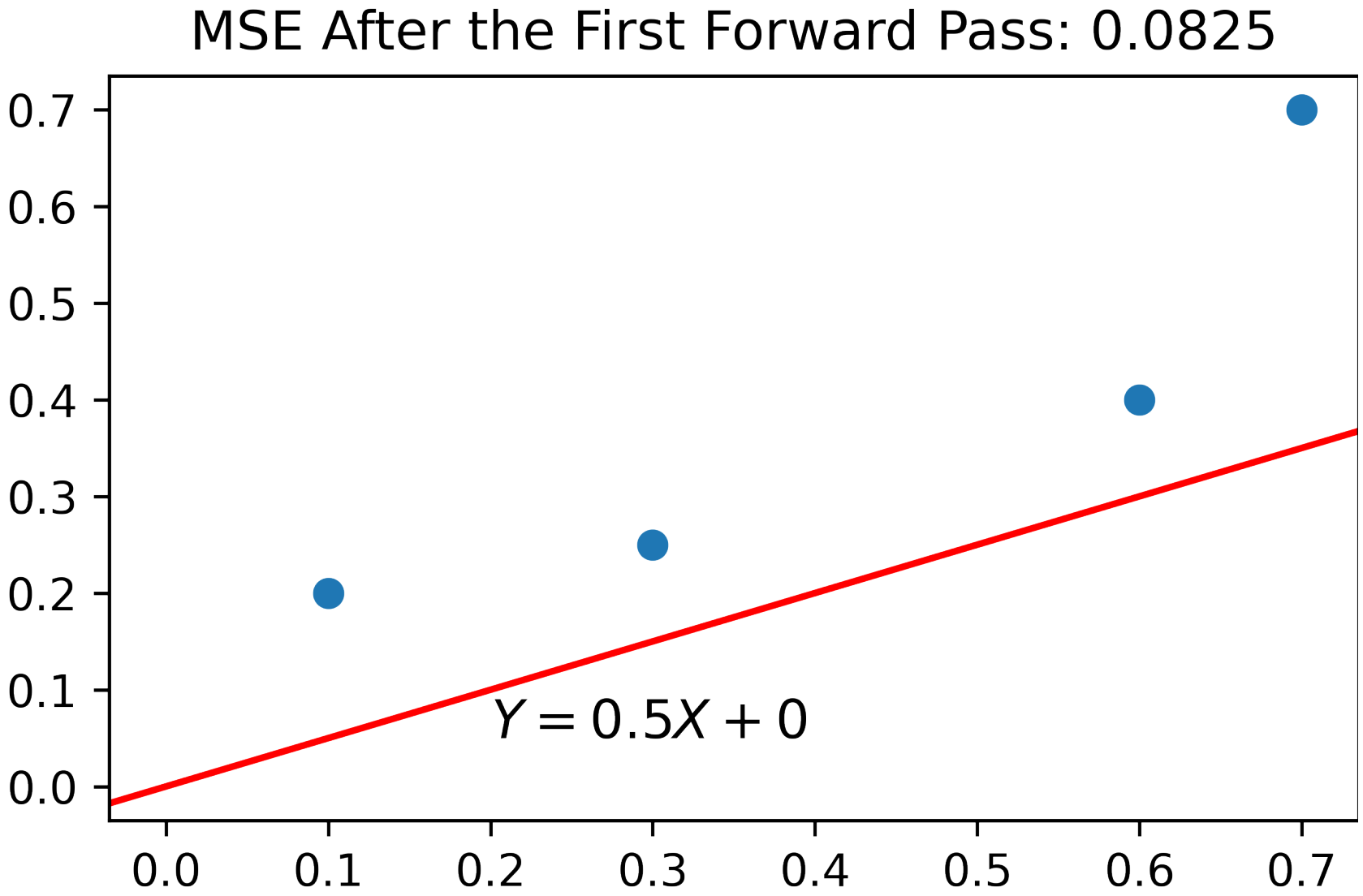
- Consider the most basic of Neural Networks, a single feature used to predict a single outcome.
- The *forward pass* calculates the predictions of the outcome from the input data using the values of W and B^2.



Forward Pass II

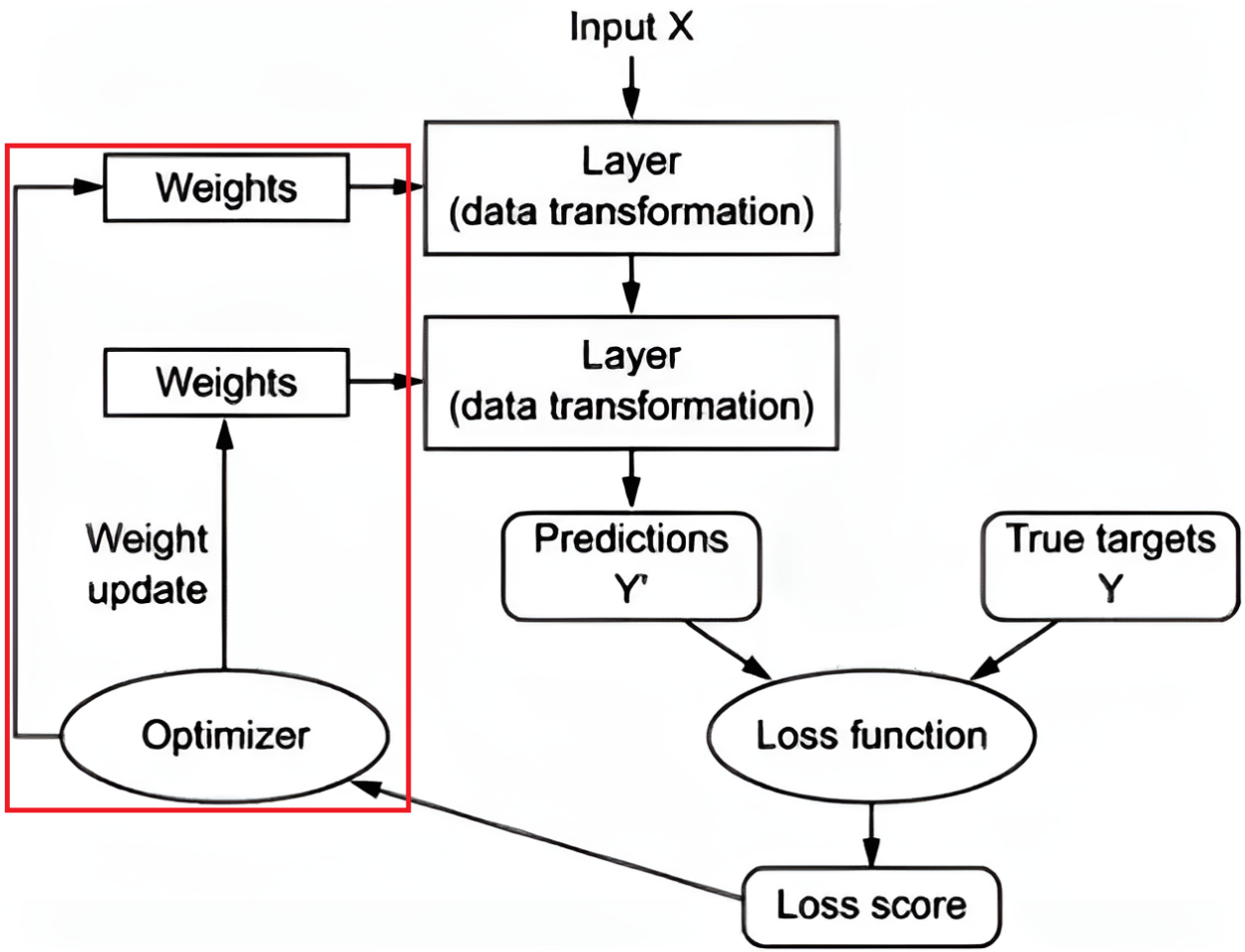
- Let $Y = \begin{bmatrix} 0.2 & 0.25 & 0.4 & 0.7 \end{bmatrix}$, $X = \begin{bmatrix} 0.1 & 0.3 & 0.6 & 0.7 \end{bmatrix}$, $W = \begin{bmatrix} 0.5 \end{bmatrix}$, $B = \begin{bmatrix} 0 \end{bmatrix}$.
- The initial predictions from the first *forward pass* are $Y' = \begin{bmatrix} 0.5 \end{bmatrix} \cdot \begin{bmatrix} 0.2 & 0.25 & 0.4 & 0.7 \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} = \begin{bmatrix} 0.05 & 0.15 & 0.3 & 0.35 \end{bmatrix}$.
- The calculated loss is $e(w, b) = \frac{1}{4}((0.2 - 0.05)^2 + (0.25 - 0.15)^2 + (0.4 - 0.3)^2 + (0.7 - 0.35)^2) = 0.0825$.

Forward Pass III



Deep Learning Roadmap: Optimization

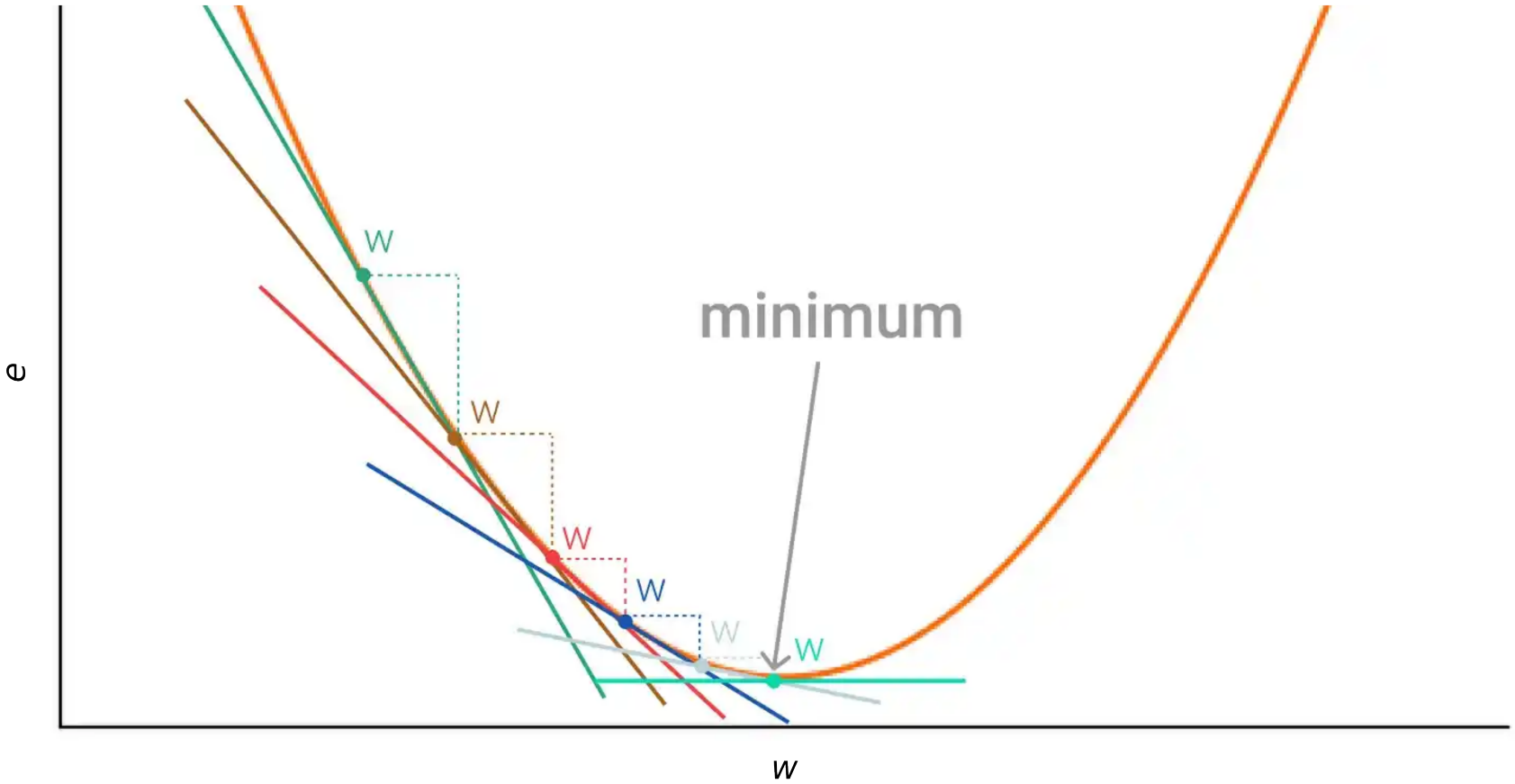
- The optimization step has the goal of *learning* the values of W and B that minimized the MSE^2.



Gradient Descent I

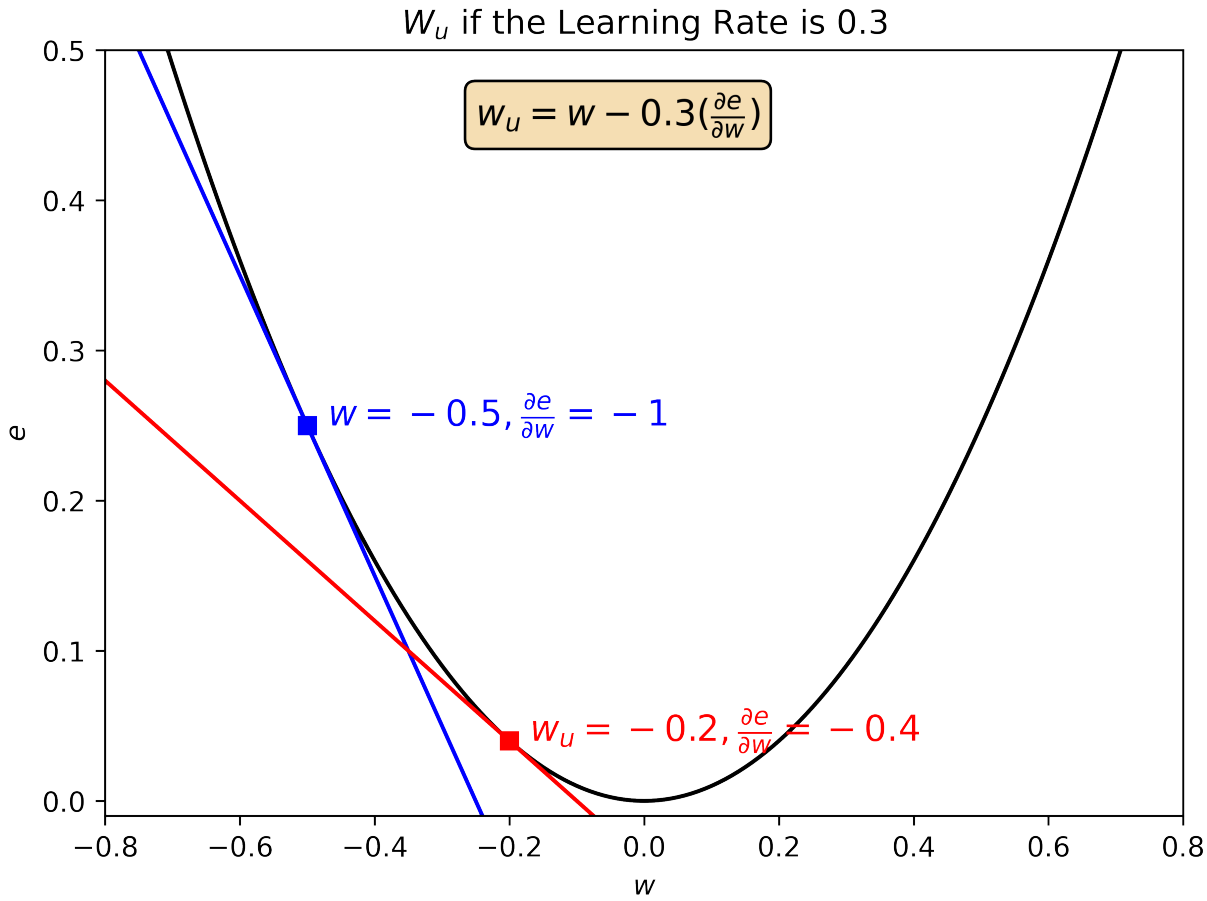
- $e = \frac{1}{n}\sum_{i=1}^{n}(y - y')^2$ is a parabola like x^2 that we saw previously. Therefore it has a single minimum where the gradient is zero[4].

The Gradient Descent Algorithm to Find the Minimum of a Parabola



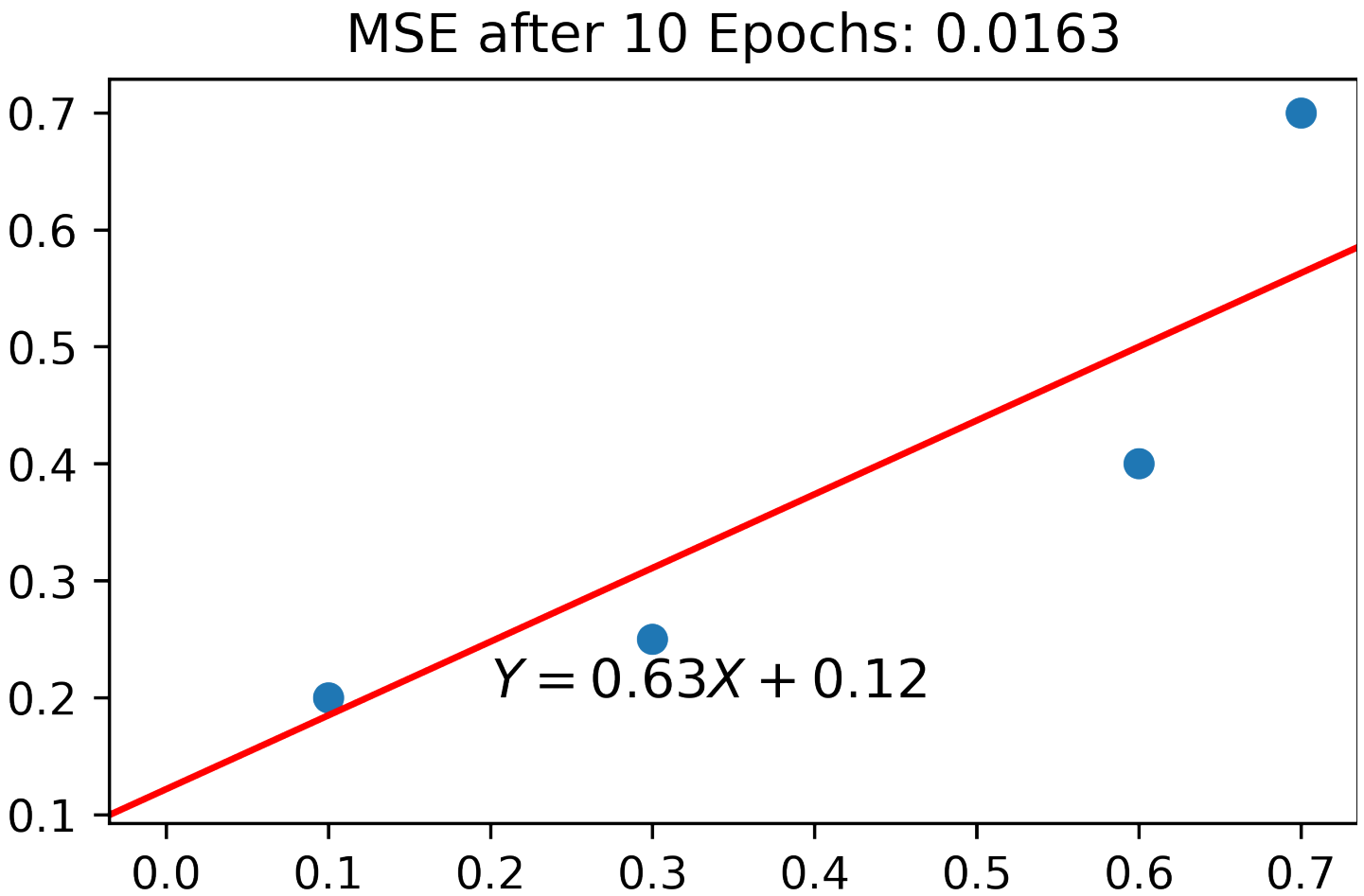
Gradient Descent II

- In general, if we move w and b in the opposite direction of the gradient $\frac{\partial e}{\partial w}$ and $\frac{\partial e}{\partial b}$, then the MSE will decrease[2].
  - The updated value of w is $w_u = w - \alpha(\frac{\partial e}{\partial w})$.
  - The updated value of b is $b_u = b - \alpha(\frac{\partial e}{\partial b})$.



Backward Pass

- Adjusting W and B using gradient descent and updating the model predictions is called a *backward pass*.
- Each cycle of a *forward pass* followed by a *backward pass* is called an *epoch*.



References I

1. Arya, N. (2023, July 19). *GPT-4 details have been leaked!* https://www.kdnuggets.com/2023/07/gpt4-details-leaked.html
2. Chollet, F. (2021). *Deep learning with python*. Simon; Schuster.
3. Berner, J., Grohs, P., Kutyniok, G., & Petersen, P. (2021). The modern mathematics of deep learning. *arXiv Preprint arXiv:2105.04026*, *78*.
4. Bento, C. (2021, June 2). *Stochastic gradient descent explained in real life*. https://towardsdatascience.com/stochastic-gradient-descent-explained-in-real-life-predicting-your-pizzas-cooking-time-b7639d5e6a32