Deep Learning

Vazgen Mikayelyan

October 20, 2020

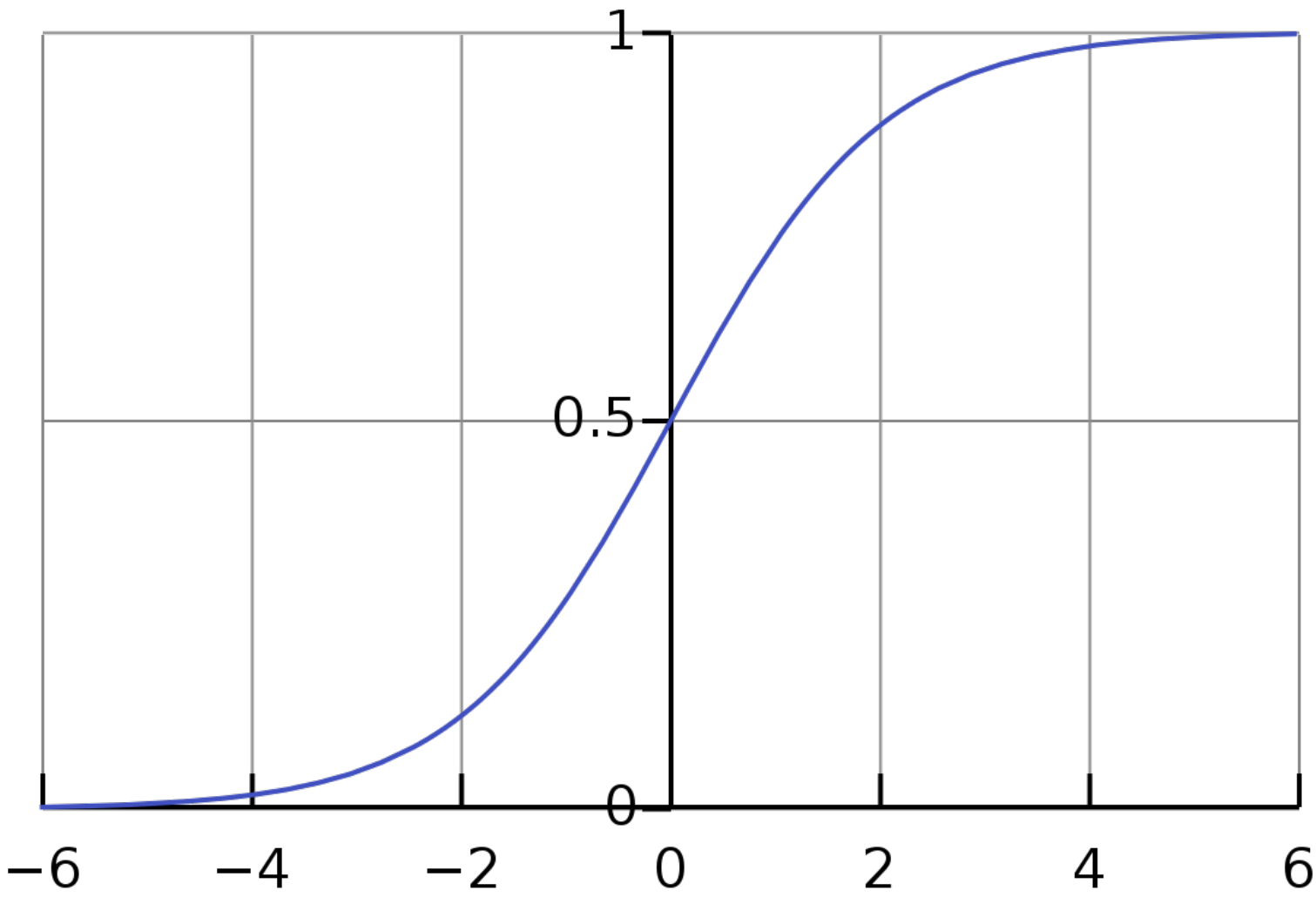
FAST DISCOVERING THE FUTURE

Activation functions

1. Sigmoid: $\sigma(x) = \dfrac{1}{1 + e^{-x}}$

Activation functions

1. Sigmoid: $\sigma(x) = \dfrac{1}{1 + e^{-x}}$

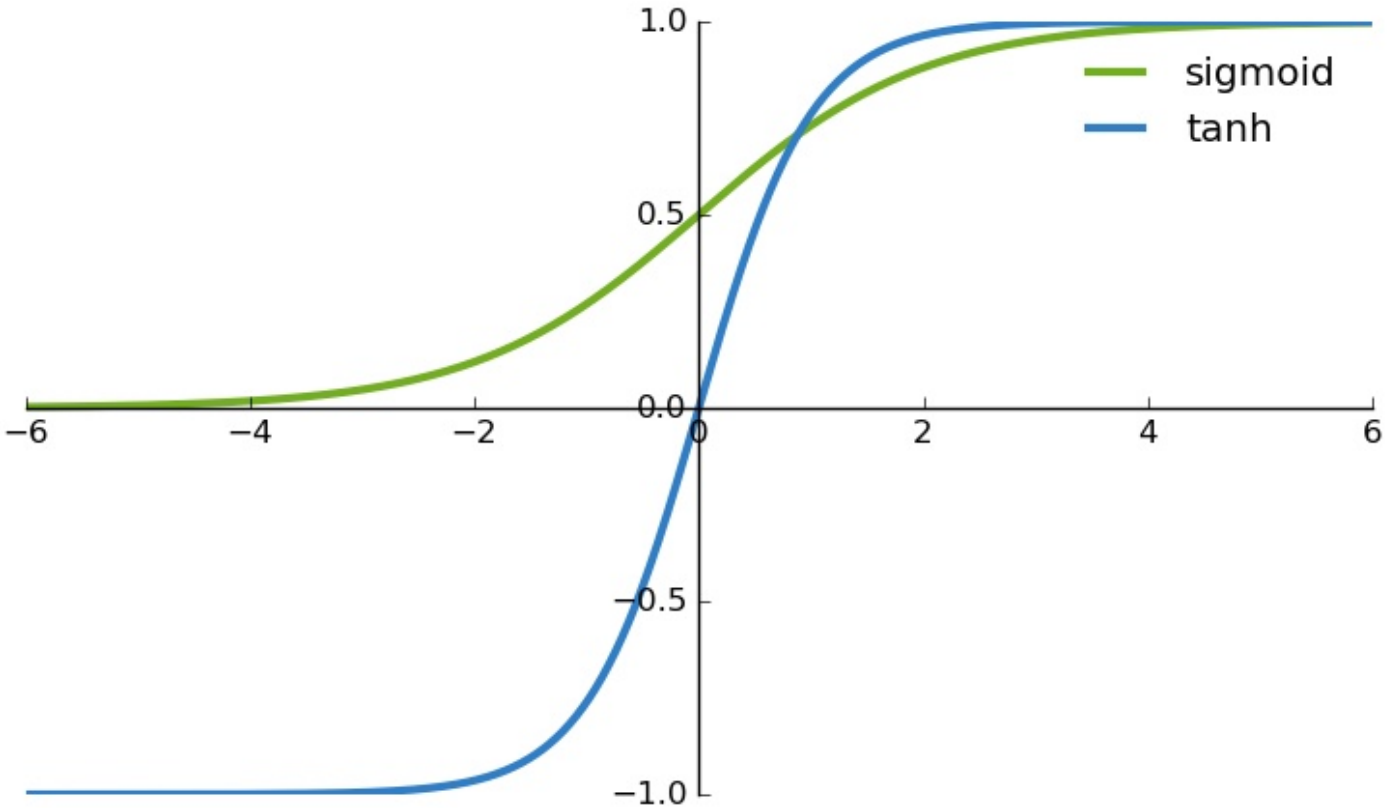


Activation functions

2. Tanh: $\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$

Activation functions

2. Tanh: $\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$

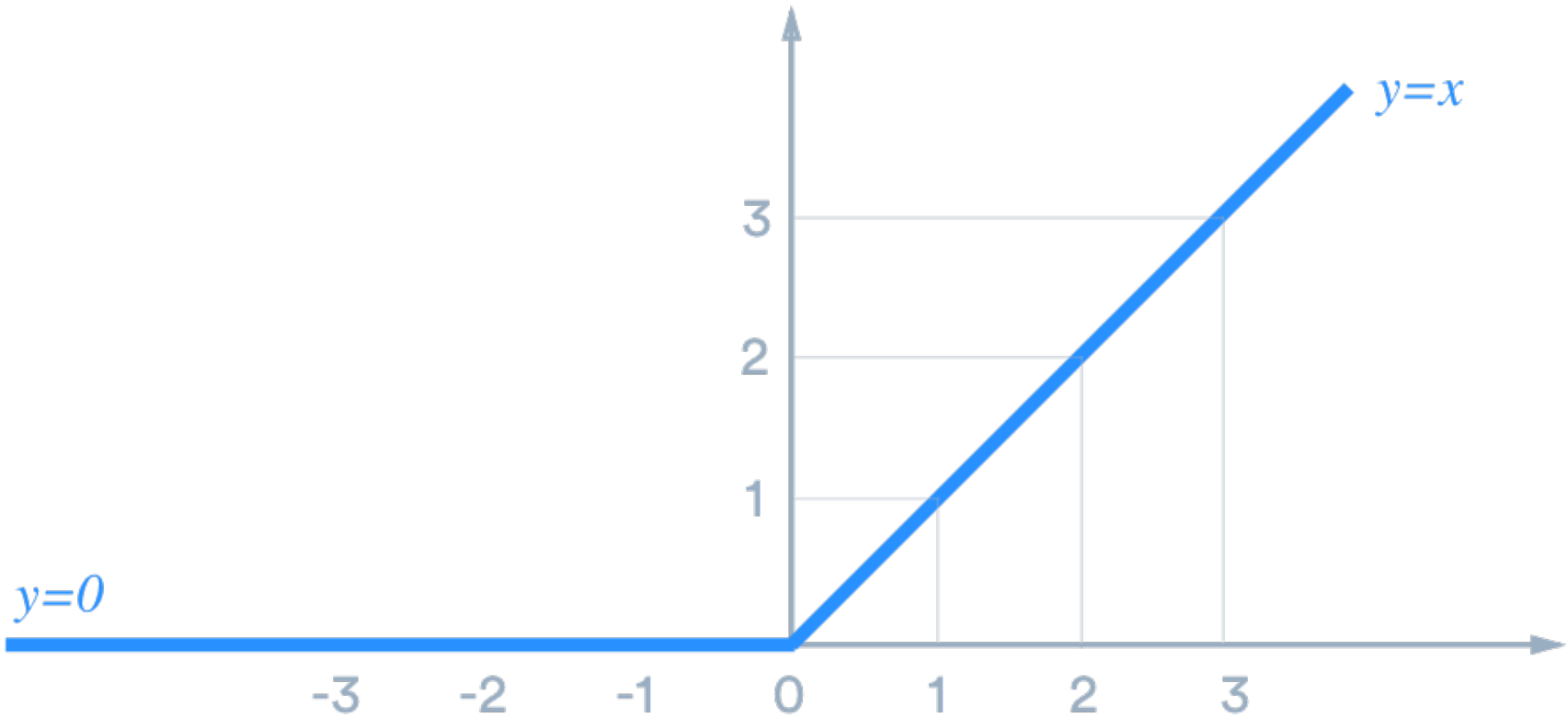


Activation functions

3. Rectified linear unit: $ReLU(x) = \max(0, x)$

Activation functions

3. Rectified linear unit: $ReLU(x) = \max(0, x)$

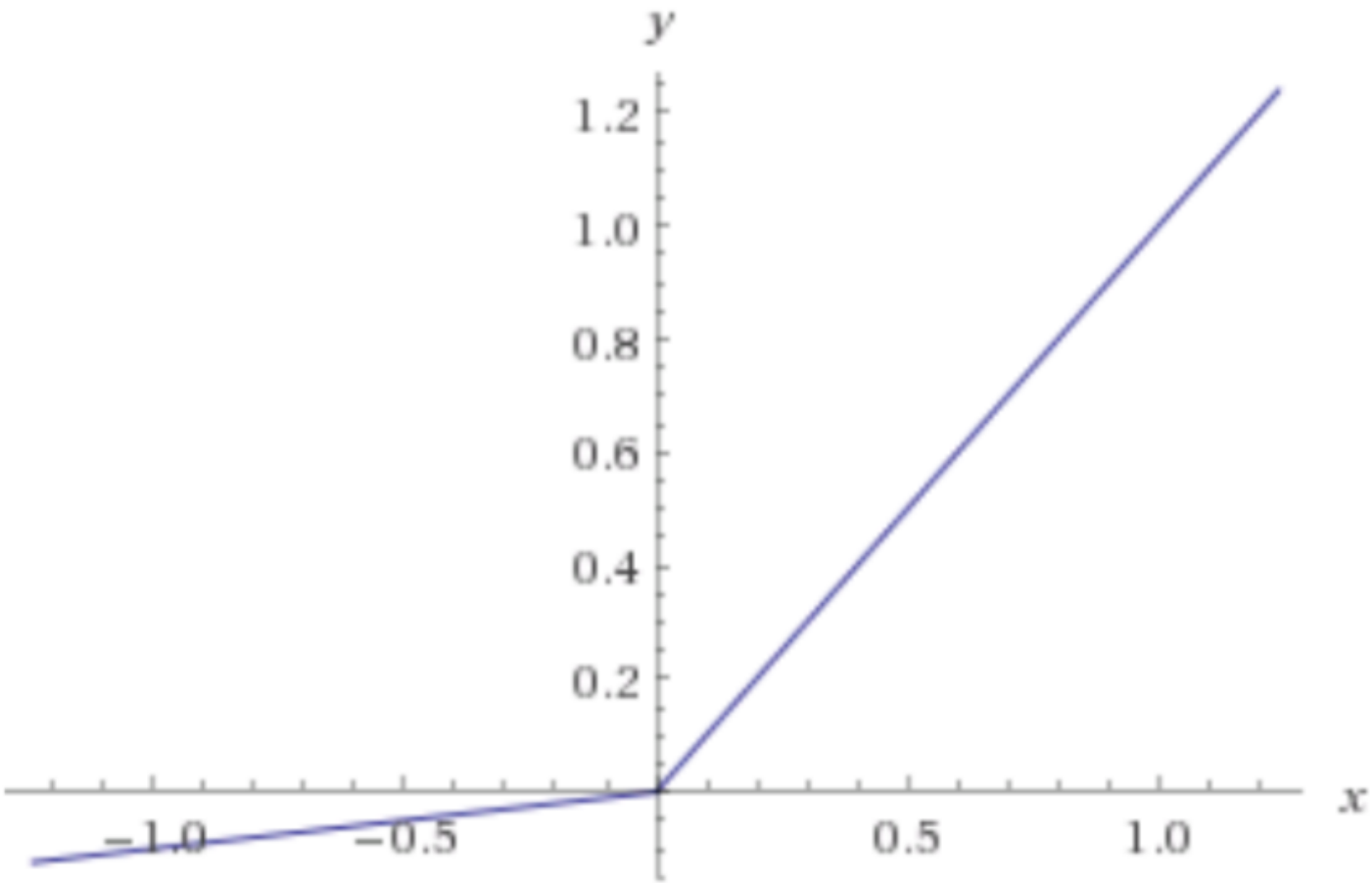


Activation functions

4. Leaky ReLU: $LR(x) = \begin{cases} 0.01x, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$

Activation functions

4. Leaky ReLU: $LR(x) = \begin{cases} 0.01x, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$

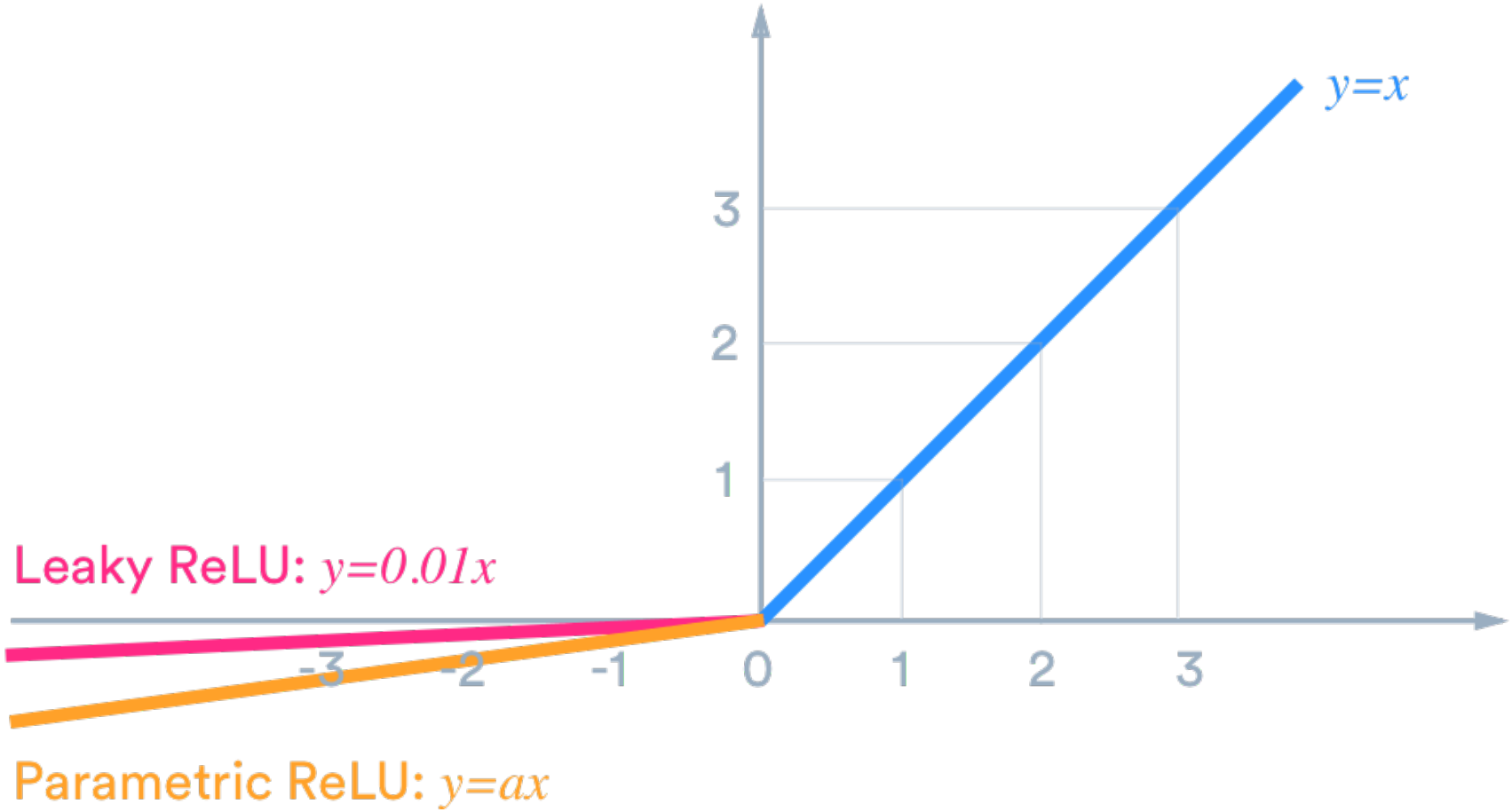


Activation functions

5. Parametric ReLU: $PR(x) = \begin{cases} ax, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$

Activation functions

5. Parametric ReLU: $PR(x) = \begin{cases} ax, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$

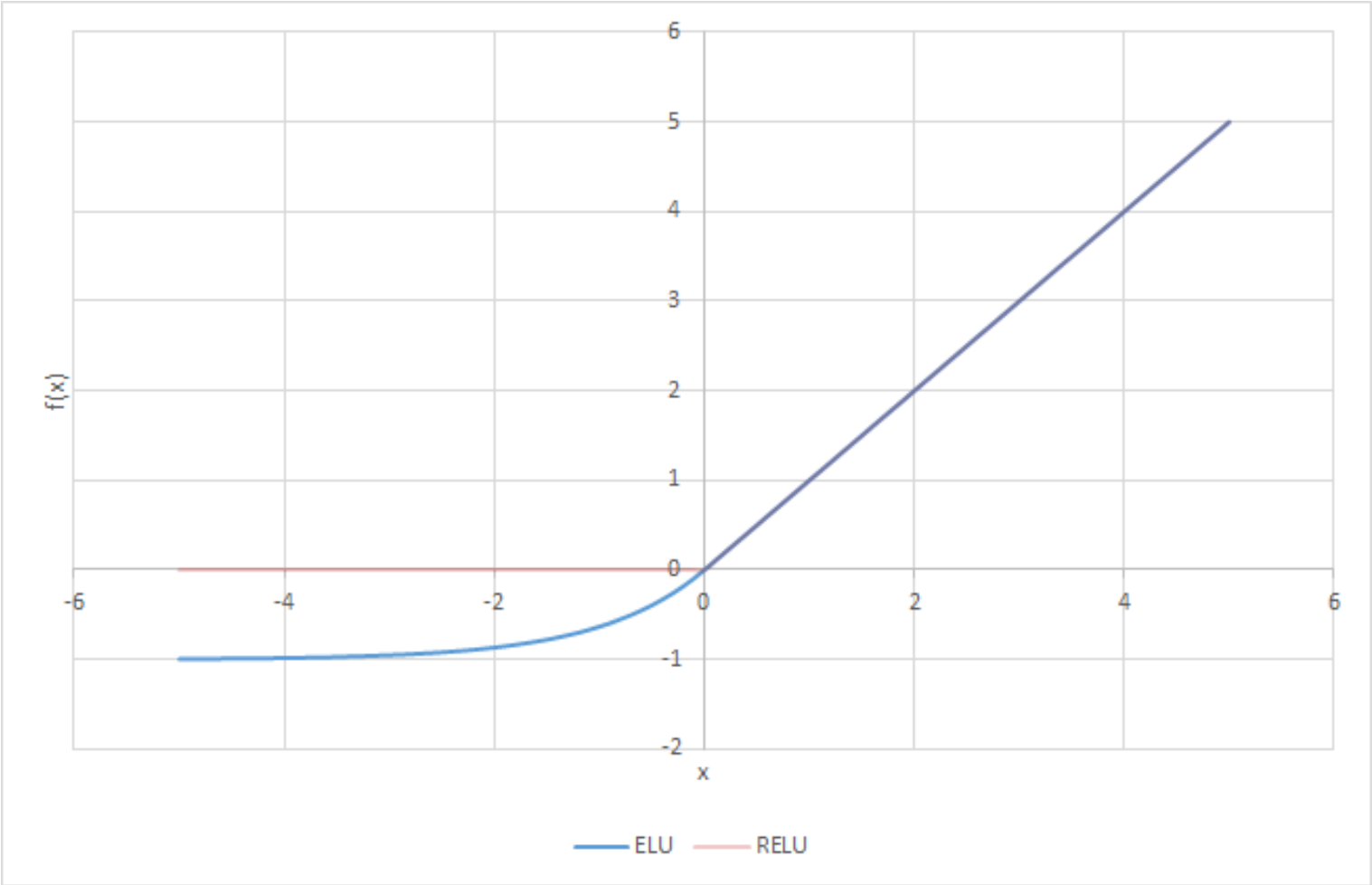


Activation functions

6. Exponential linear unit: $ELU(x) = \begin{cases} a(e^x - 1), & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$

Activation functions

6. Exponential linear unit: $ELU(x) = \begin{cases} a(e^x - 1), & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$

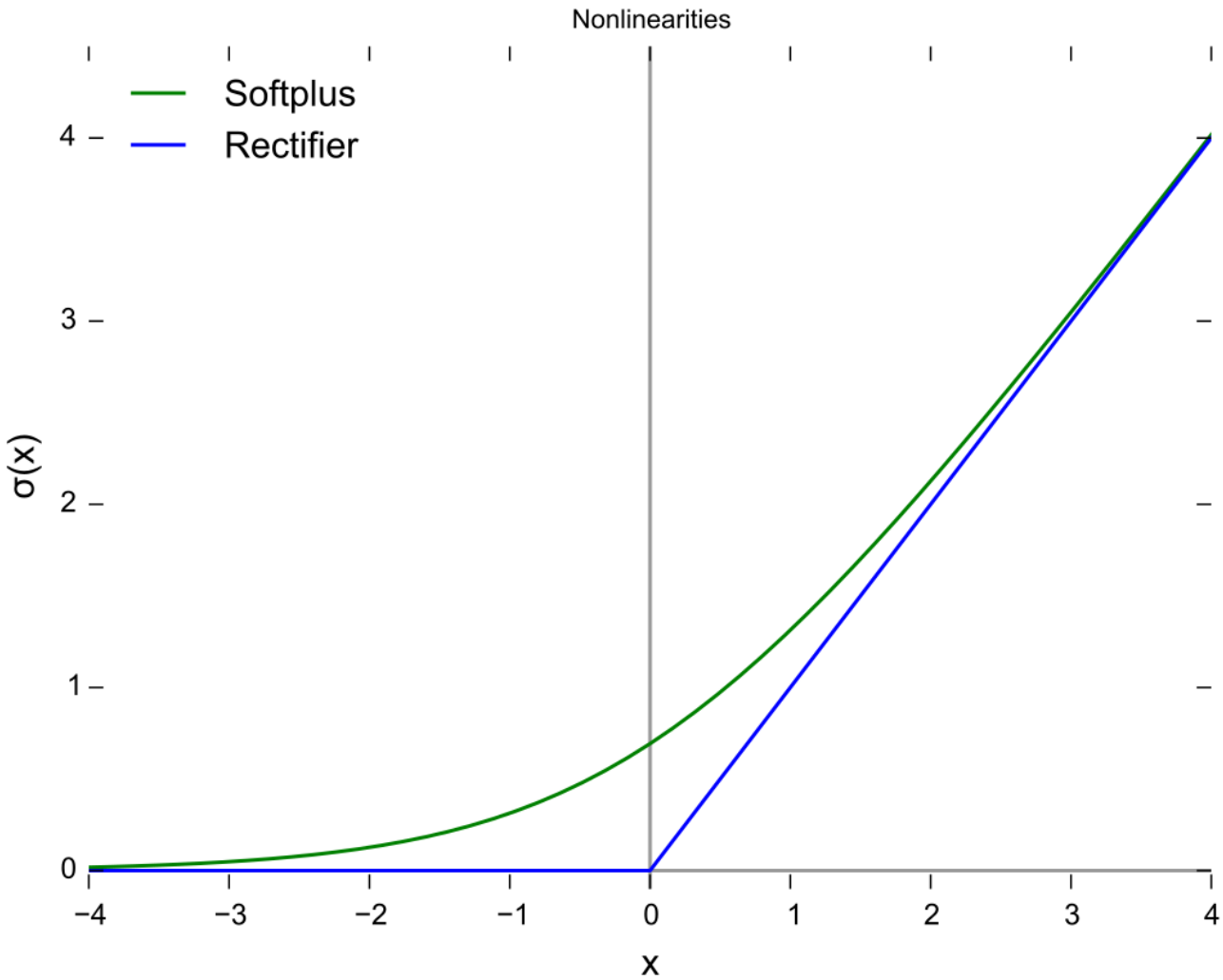


Activation functions

7. SoftPlus: $SP(x) = \log(1 + e^x)$

Activation functions

7. SoftPlus: $SP(x) = \log(1 + e^x)$



Activation functions

8. Softmax: $S\left(x_1, x_2, \ldots, x_n\right) = \left(\frac{e^{x_1}}{\sum_{i=1}^{n} e^{x_i}}, \frac{e^{x_2}}{\sum_{i=1}^{n} e^{x_i}}, \ldots, \frac{e^{x_n}}{\sum_{i=1}^{n} e^{x_i}} \right)$

Questions

1. Why do we need activation functions?

Questions

1. Why do we need activation functions?
2. How should we define activation functions, for a layer or for a neuron?

Outline

Gradient Descent

Let $f : \mathbb{R}^k \to \mathbb{R}$ be a convex function and we want to find its global minimum.

Gradient Descent

Let $f : \mathbb{R}^k \to \mathbb{R}$ be a convex function and we want to find its global minimum. This optimization algorithm is based on the fact that the fastest decreasing direction of the function is the opposite direction of gradient:

$$x_{n+1} = x_n - \alpha \nabla f(x_n)$$

and $x_0 \in \mathbb{R}^k$ is a arbitrary point.

Outline

Linear Regression

Let $(x_i, y_i)_{i=1}^n$, $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}$ be our training data.

Linear Regression

Let $(x_i, y_i)_{i=1}^n$, $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}$ be our training data. Consider the function

$$f(x) = f\left(x^1, x^2, \ldots, x^k\right) = w^1 x^1 + w^2 x^2 + \ldots + w^k x^k + b = w^T x + b.$$

Linear Regression

Let $(x_i, y_i)_{i=1}^n$, $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}$ be our training data. Consider the function

$$f(x) = f\left(x^1, x^2, \ldots, x^k\right) = w^1 x^1 + w^2 x^2 + \ldots + w^k x^k + b = w^T x + b.$$

Our aim is to find parameters $b, w^1, w^2, \ldots, w^k$ such that

$$f(x_i) \approx y_i, i = 1, \ldots, n.$$

Linear Regression

Let $(x_i, y_i)_{i=1}^n$, $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}$ be our training data. Consider the function

$$f(x) = f\left(x^1, x^2, \ldots, x^k\right) = w^1 x^1 + w^2 x^2 + \ldots + w^k x^k + b = w^T x + b.$$

Our aim is to find parameters $b, w^1, w^2, \ldots, w^k$ such that

$$f(x_i) \approx y_i, i = 1, \ldots, n.$$

We choose L^2 distance as our loss function:

$$\frac{1}{n} \sum_{l=1}^{n} \left(f(x_l) - y_l\right)^2.$$

Questions

1. Should we minimize the loss function using gradient descent?

Questions

1. Should we minimize the loss function using gradient descent?
2. Can you represent this model as a neural network?