5 - Deep Learning Overview

5.3 - Activation Function

Marciel Barros

Setembro, 2020

1. Deep Learning Summary
2. Training Rules
3. **Activation Function**
4. Regularization
5. Optimizer
6. Types of Neural Networks
7. Common Problems

Activation Function

- Activation functions are important for the neural network model to learn and understand complex non-linear functions. They allow introduction of non-linear features to the network.
- Without activation functions, output signals are only simple **linear functions**. The complexity of linear functions is limited, and the capability of learning complex function mappings from data is low.

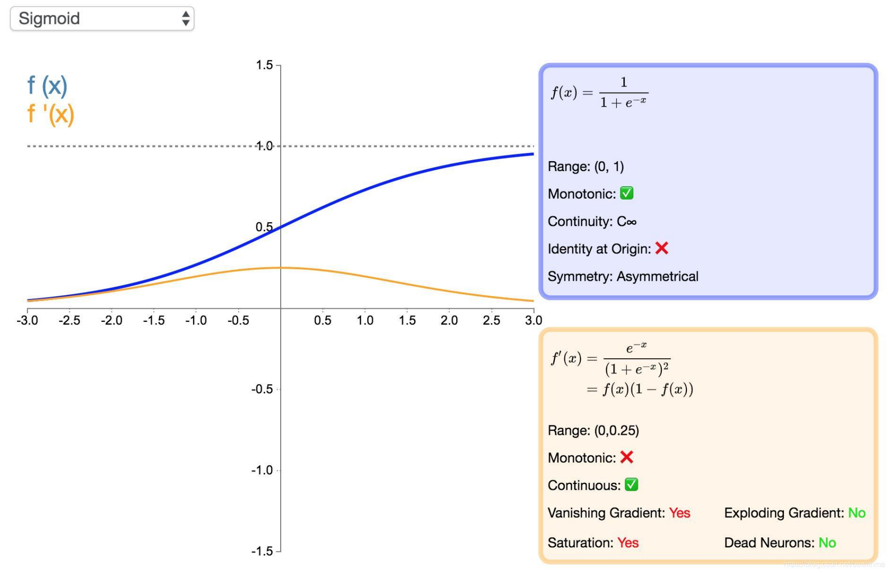
Activation Function

$$output = f(w_1x_1 + w_2x_2 + w_3x_3 \ldots) = f(W^t \bullet X)$$

Sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$

- The sigmoid function is monotonic, continuous, and easy to derive.
- The output is bounded, and the network is easy to converge.
- When the network is very deep, more and more backpropagation gradients fall into the saturation area so that the gradient module becomes smaller.
- Generally, if the sigmoid network has five or fewer layers, the gradient is degraded to 0, which is difficult to train. This phenomenon is a **vanishing gradient**. In addition, the output of the sigmoid is not zero-centered.



$$f(x) = \frac{1}{1 + e^{-x}}$$

Range: (0, 1)

Monotonic: ✅

Continuity: C∞

Identity at Origin: ❌

Symmetry: Asymmetrical

$$f'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = f(x)(1 - f(x))$$

Range: (0,0.25)

Monotonic: ❌

Continuous: ✅

Vanishing Gradient: Yes   Exploding Gradient: No

Saturation: Yes   Dead Neurons: No

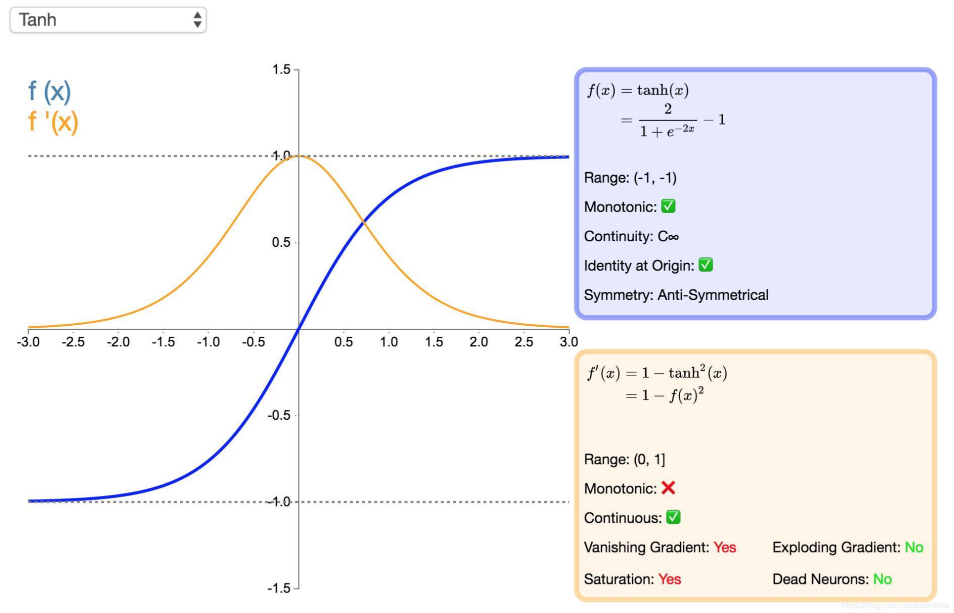
https://blog.csdn.net/bewithme

Tanh

•

- Tanh function and sigmoid function have similar shortcomings.
- The derivative of the tanh function is nearly 0 at its extremes.
- However, because the tanh function is symmetric with respect to the origin, the average of the outputs is closer to 0 than that of the sigmoid function.
- Therefore, SGD can reduce the required number of iterations because it is closer to the natural gradient descent.

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



$$f(x) = \tanh(x) \\ = \frac{2}{1+e^{-2x}} - 1$$

Range: (-1, -1)

Monotonic: ✓

Continuity: C∞

Identity at Origin: ✓

Symmetry: Anti-Symmetrical

$$f'(x) = 1 - \tanh^2(x) \\ = 1 - f(x)^2$$

Range: (0, 1]

Monotonic: ✗

Continuous: ✓

Vanishing Gradient: Yes

Exploding Gradient: No

Saturation: Yes

Dead Neurons: No

https://blog.csdn.net/bewithme

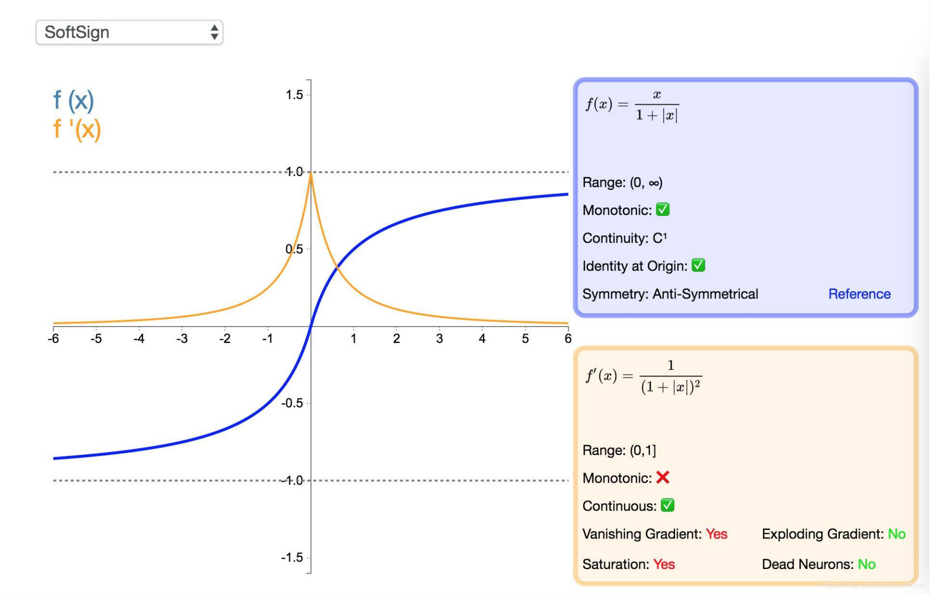
Softsign

$$f(x) = \frac{x}{|x| + 1}$$

This function saturates more slowly than the tanh function.

When the sigmoid, tanh, and softsign functions are used to train a deep neural network, the vanishing gradient problem is inevitable. The derivative of the functions approaches 0 at its extremes. When the network is very deep, more and more backpropagation gradients fall into the saturation area so that the gradient module becomes smaller and finally close to 0, and the weight cannot be updated.

Generally, if the neural network has more than five layers, the gradient is degraded to 0, which is difficult to train.



$f(x) = \frac{x}{1 + |x|}$

Range: (0, ∞)

Monotonic: ✓

Continuity: C^1

Identity at Origin: ✓

Symmetry: Anti-Symmetrical

Reference

$f'(x) = \frac{1}{(1 + |x|)^2}$

Range: (0,1]

Monotonic: ✗

Continuous: ✓

Vanishing Gradient: Yes

Exploding Gradient: No

Saturation: Yes

Dead Neurons: No

Rectified Linear Unit (ReLU)

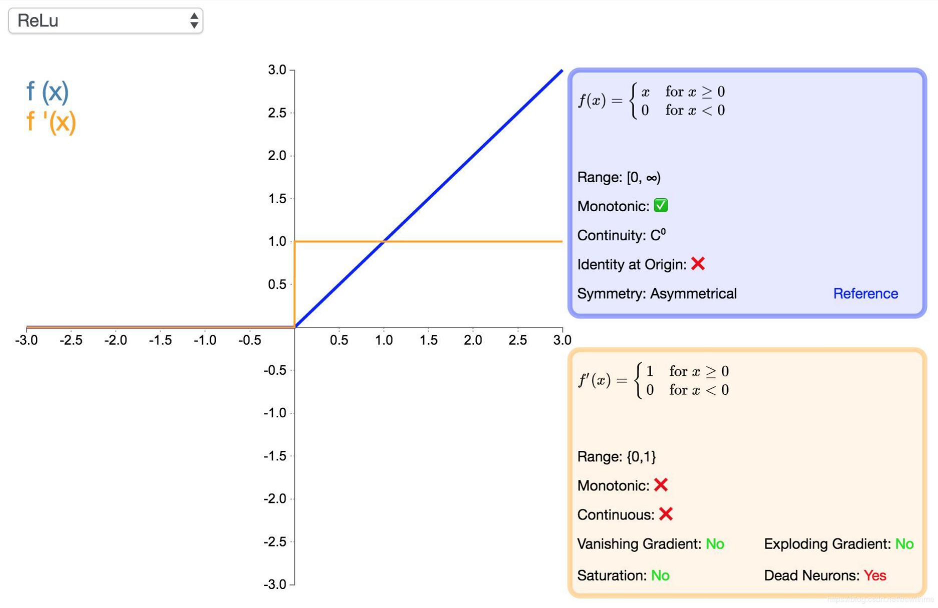
$$y = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Advantages:

- Compared with sigmoid and tanh, ReLU supports fast convergence in SGD.
- Compared with the sigmoid and tanh functions involving exponentiation, the ReLU can be implemented more easily.
- The vanishing gradient problem can be effectively alleviated.
- The ReLU has a good performance during unsupervised pre-training.

Disadvantages:

- There is no upper bound, so that the training is relatively easy to diverge.
- The ReLU is not differentiable at x = 0 and a derivative is forcibly defined at this point.
- The surface defined at the zero point is not smooth enough in some regression problems.



$f(x) = \begin{cases} x & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}$

Range: [0, ∞)

Monotonic: ✓

Continuity: C^0

Identity at Origin: ✗

Symmetry: Asymmetrical

Reference

$f'(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}$

Range: {0,1}

Monotonic: ✗

Continuous: ✗

Vanishing Gradient: No

Exploding Gradient: No

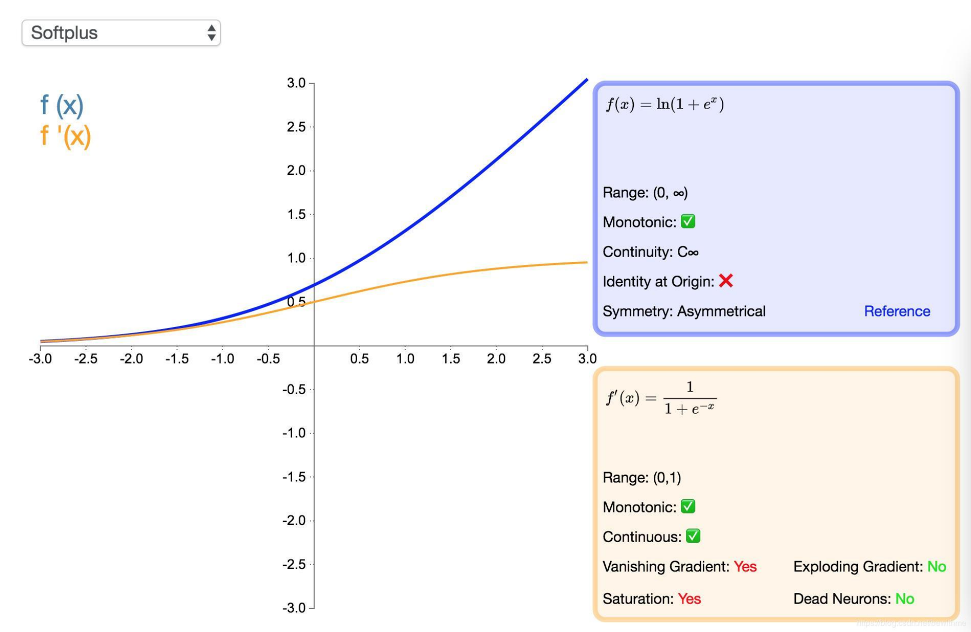
Saturation: No

Dead Neurons: Yes

Softplus

$$f(x) = \ln(e^x + 1)$$

- Compared with ReLU, this function has more complex computation. However, it has a continuous derivative and defines a smooth curved surface.



$f(x) = \ln(1 + e^x)$

Range: (0, ∞)

Monotonic: ✓

Continuity: C∞

Identity at Origin: ✗

Symmetry: Asymmetrical

Reference

$f'(x) = \frac{1}{1 + e^{-x}}$

Range: (0,1)

Monotonic: ✓

Continuous: ✓

Vanishing Gradient: Yes

Exploding Gradient: No

Saturation: Yes

Dead Neurons: No

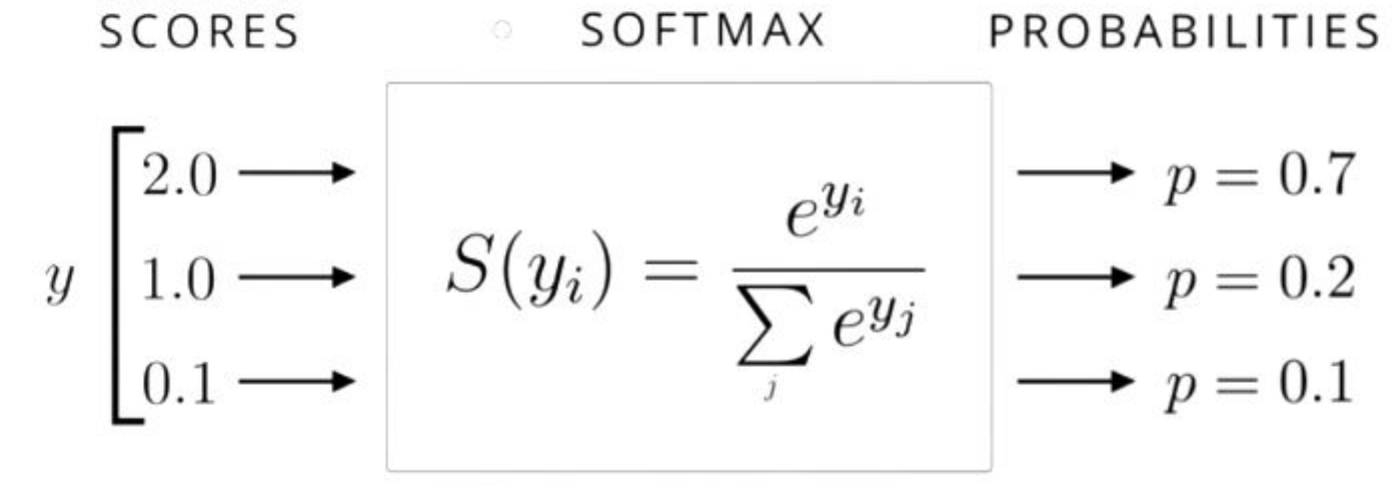
Softmax

- Softmax function:

$$\sigma(\mathrm{z})_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

- The Softmax function is used to map a K-dimensional vector of arbitrary real values to another K-dimensional vector of real values, where each vector element is in the interval (0, 1). All the elements add up to 1.
- The Softmax function is often used as the output layer of a multiclass classification task.

Softmax



$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d

Softmax - Example

If the input of a softmax function is [1,2,4,2,1], which of the following option may be the output?

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

y	e^y	S(y)
1	2.71	0.04
2	7.39	0.10
4	54.6	0.72
2	7.39	0.10
1	2.71	0.04
sum(e^y)	74.8	

Summary

Softmax can **stabilize** the values of overflow and underflow;

Sigmoid is used as the gate function in LSTM;

Tanh **cannot** effectively solve the vanishing gradient problem;

5 - Deep Learning Overview

Next: 5.4 - Regularization

Marciel Barros

Setembro, 2020