Your Name

October 5, 2025

What are Activation Functions?

- Activation functions introduce **non-linearity** into neural networks.

What are Activation Functions?

- Activation functions introduce **non-linearity** into neural networks.
- They determine whether a neuron should be activated or not based on the input.

What are Activation Functions?

- Activation functions introduce **non-linearity** into neural networks.
- They determine whether a neuron should be activated or not based on the input.
- Essential for learning complex patterns and enabling deep learning models to solve non-linear problems.

What are Activation Functions?

- Activation functions introduce **non-linearity** into neural networks.
- They determine whether a neuron should be activated or not based on the input.
- Essential for learning complex patterns and enabling deep learning models to solve non-linear problems.
- Mathematically, an activation function $f(x)$ transforms the input signal into an output signal.

Why Are They Important?

- Enable neural networks to model **complex, non-linear relationships**.

Why Are They Important?

- Enable neural networks to model **complex, non-linear relationships**.
- Without activation functions, neural networks would behave like linear regression models.

Why Are They Important?

- Enable neural networks to model **complex, non-linear relationships**.
- Without activation functions, neural networks would behave like linear regression models.
- Help in **gradient-based optimization** by providing differentiable outputs.

Why Are They Important?

- Enable neural networks to model **complex, non-linear relationships**.
- Without activation functions, neural networks would behave like linear regression models.
- Help in **gradient-based optimization** by providing differentiable outputs.
- Different activation functions suit different types of problems and architectures.

Common Types of Activation Functions

- Sigmoid

Common Types of Activation Functions

- Sigmoid
- ReLU (Rectified Linear Unit)

Common Types of Activation Functions

- Sigmoid
- ReLU (Rectified Linear Unit)
- Tanh (Hyperbolic Tangent)

Common Types of Activation Functions

- Sigmoid
- ReLU (Rectified Linear Unit)
- Tanh (Hyperbolic Tangent)
- Leaky ReLU

Common Types of Activation Functions

- Sigmoid
- ReLU (Rectified Linear Unit)
- Tanh (Hyperbolic Tangent)
- Leaky ReLU
- Softmax

Common Types of Activation Functions

- Sigmoid
- ReLU (Rectified Linear Unit)
- Tanh (Hyperbolic Tangent)
- Leaky ReLU
- Softmax

Note: Each has unique properties, making them suitable for specific tasks.

Sigmoid Activation Function

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

Sigmoid Activation Function

- Formula: $f(x) = \frac{1}{1+e^{-x}}$
- Output range: $(0, 1)$

Sigmoid Activation Function

- Formula: $f(x) = \frac{1}{1+e^{-x}}$
- Output range: $(0, 1)$
- Commonly used in binary classification tasks.

Sigmoid Activation Function

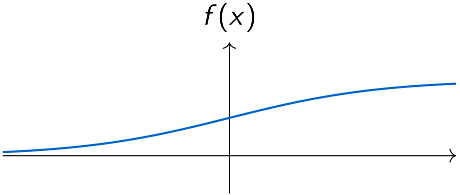
- Formula: $f(x) = \frac{1}{1+e^{-x}}$
- Output range: $(0, 1)$
- Commonly used in binary classification tasks.
- **Pros**: Smooth, interpretable output (probabilities).

Sigmoid Activation Function

- Formula: $f(x) = \frac{1}{1+e^{-x}}$
- Output range: $(0, 1)$
- Commonly used in binary classification tasks.
- **Pros**: Smooth, interpretable output (probabilities).
- **Cons**: Vanishing gradient problem, not zero-centered.

Sigmoid Activation Function

- Formula: $f(x) = \frac{1}{1+e^{-x}}$
- Output range: $(0, 1)$
- Commonly used in binary classification tasks.
- **Pros**: Smooth, interpretable output (probabilities).
- **Cons**: Vanishing gradient problem, not zero-centered.



ReLU (Rectified Linear Unit)

- Formula: $f(x) = \max(0, x)$

ReLU (Rectified Linear Unit)

- Formula: $f(x) = \max(0, x)$
- Output range: $[0, \infty)$

ReLU (Rectified Linear Unit)

- Formula: $f(x) = \max(0, x)$
- Output range: $[0, \infty)$
- Widely used in deep learning due to simplicity and effectiveness.

ReLU (Rectified Linear Unit)

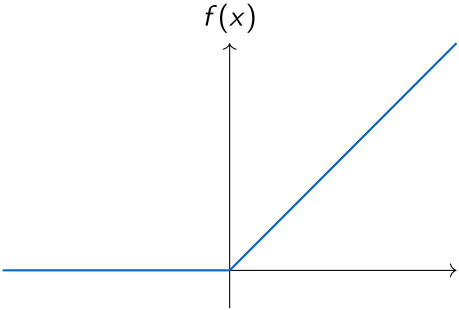
- Formula: $f(x) = \max(0, x)$
- Output range: $[0, \infty)$
- Widely used in deep learning due to simplicity and effectiveness.
- **Pros**: Avoids vanishing gradient, computationally efficient.

ReLU (Rectified Linear Unit)

- Formula: $f(x) = \max(0, x)$
- Output range: $[0, \infty)$
- Widely used in deep learning due to simplicity and effectiveness.
- **Pros**: Avoids vanishing gradient, computationally efficient.
- **Cons**: "Dying ReLU" problem (neurons outputting zero).

ReLU (Rectified Linear Unit)

- Formula: $f(x) = \max(0, x)$
- Output range: $[0, \infty)$
- Widely used in deep learning due to simplicity and effectiveness.
- **Pros**: Avoids vanishing gradient, computationally efficient.
- **Cons**: "Dying ReLU" problem (neurons outputting zero).



Tanh (Hyperbolic Tangent)

- Formula:
  $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Tanh (Hyperbolic Tangent)

- Formula:
  $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Output range: $(-1, 1)$

Tanh (Hyperbolic Tangent)

- Formula:
  $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Output range: $(-1, 1)$
- Used in hidden layers of neural networks.

Tanh (Hyperbolic Tangent)

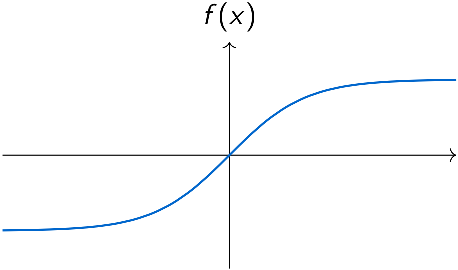
- Formula:
  $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Output range: $(-1, 1)$
- Used in hidden layers of neural networks.
- **Pros**: Zero-centered, stronger gradients than sigmoid.

Tanh (Hyperbolic Tangent)

- Formula:
  $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Output range: $(-1, 1)$
- Used in hidden layers of neural networks.
- **Pros**: Zero-centered, stronger gradients than sigmoid.
- **Cons**: Still suffers from vanishing gradient problem.

Tanh (Hyperbolic Tangent)

- Formula: $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Output range: $(-1, 1)$
- Used in hidden layers of neural networks.
- **Pros**: Zero-centered, stronger gradients than sigmoid.
- **Cons**: Still suffers from vanishing gradient problem.



Leaky ReLU

- Formula:
  $f(x) = \max(\alpha x, x)$, where α is a small constant (e.g., 0.01)

Leaky ReLU

- Formula: $f(x) = \max(\alpha x, x)$, where α is a small constant (e.g., 0.01)
- Output range: $(-\infty, \infty)$

Leaky ReLU

- Formula: $f(x) = \max(\alpha x, x)$, where α is a small constant (e.g., 0.01)
- Output range: $(-\infty, \infty)$
- Addresses the "dying ReLU" problem.

Leaky ReLU

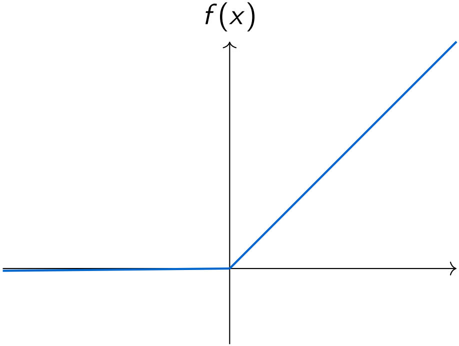
- Formula:
  $f(x) = \max(\alpha x, x)$, where α is a small constant (e.g., 0.01)
- Output range: $(-\infty, \infty)$
- Addresses the "dying ReLU" problem.
- **Pros**: Allows small gradients for negative inputs.

Leaky ReLU

- Formula:
  $f(x) = \max(\alpha x, x)$, where α is a small constant (e.g., 0.01)
- Output range: $(-\infty, \infty)$
- Addresses the "dying ReLU" problem.
- **Pros**: Allows small gradients for negative inputs.
- **Cons**: Requires tuning of α, less common than ReLU.

Leaky ReLU

- Formula:
  $f(x) = \max(\alpha x, x)$, where α is a small constant (e.g., 0.01)
- Output range: $(-\infty, \infty)$
- Addresses the "dying ReLU" problem.
- **Pros**: Allows small gradients for negative inputs.
- **Cons**: Requires tuning of α, less common than ReLU.



Softmax

- Formula: $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$

Softmax

- Formula: $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$
- Output range: $(0, 1)$, with sum of outputs $= 1$

Softmax

- Formula: $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$
- Output range: $(0, 1)$, with sum of outputs $= 1$
- Used in multi-class classification (output layer).

Softmax

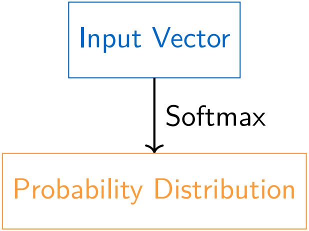
- Formula: $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$
- Output range: $(0, 1)$, with sum of outputs $= 1$
- Used in multi-class classification (output layer).
- **Pros**: Interpretable as probabilities, good for classification.

Softmax

- Formula: $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$
- Output range: $(0, 1)$, with sum of outputs $= 1$
- Used in multi-class classification (output layer).
- **Pros**: Interpretable as probabilities, good for classification.
- **Cons**: Computationally expensive for large classes.

Softmax

- Formula: $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$
- Output range: $(0, 1)$, with sum of outputs $= 1$
- Used in multi-class classification (output layer).
- **Pros**: Interpretable as probabilities, good for classification.
- **Cons**: Computationally expensive for large classes.



Comparison of Activation Functions

Function	**Output Range**	**Pros**	
Sigmoid	(0, 1)	Probabilistic output	V
ReLU	$[0, \infty)$	Fast, avoids vanishing gradient	
Tanh	(-1, 1)	Zero-centered	V
Leaky ReLU	$(-\infty, \infty)$	Prevents dying ReLU	
Softmax	(0, 1)	Probabilistic, multi-class	Comp

Conclusion

- Activation functions are critical for enabling neural networks to solve complex problems.

Conclusion

- Activation functions are critical for enabling neural networks to solve complex problems.
- Choose the activation function based on the task:

Conclusion

- Activation functions are critical for enabling neural networks to solve complex problems.
- Choose the activation function based on the task:
  - Binary classification: Sigmoid

Conclusion

- Activation functions are critical for enabling neural networks to solve complex problems.
- Choose the activation function based on the task:
  - Binary classification: Sigmoid
  - Deep networks: ReLU or Leaky ReLU

Conclusion

- Activation functions are critical for enabling neural networks to solve complex problems.
- Choose the activation function based on the task:
  - Binary classification: Sigmoid
  - Deep networks: ReLU or Leaky ReLU
  - Multi-class classification: Softmax

Conclusion

- Activation functions are critical for enabling neural networks to solve complex problems.
- Choose the activation function based on the task:
  - Binary classification: Sigmoid
  - Deep networks: ReLU or Leaky ReLU
  - Multi-class classification: Softmax
  - Hidden layers: Tanh or ReLU variants

Conclusion

- Activation functions are critical for enabling neural networks to solve complex problems.
- Choose the activation function based on the task:
  - Binary classification: Sigmoid
  - Deep networks: ReLU or Leaky ReLU
  - Multi-class classification: Softmax
  - Hidden layers: Tanh or ReLU variants
- Experimentation is key to finding the best activation function for your model!

Thank You!

Questions?

Contact: Your Name (your.email@example.com)