Activation Function

Deep Neural Networks
Session 03
Pramod Sharma
pramod.sharma@prasami.com

2

Agenda

- Sigmoid
- Softmax
- Tanh
- ReLU and its variations
- GELU (Gaussian Error Linear Unit)
- Swish and its variations

11/22/2024

pra-sâmi

3

Forward and Back Propagation

x1, w1, x2, w2, b → $z = x_1 * w_1 + x_2 * w_2 + b$ → $\hat{y} = a = \sigma(z)$ → $\ell(a, y)$

$\frac{\partial \ell}{\partial z} = \frac{\partial \ell}{\partial a} \cdot \frac{\partial a}{\partial z}$

$= \{-\frac{y}{a} + \frac{1-y}{1-a}\} * a * (1-a)$

$= a - y$

$\frac{\partial \sigma(z)}{\partial z} = \sigma(z) \cdot (1 - \sigma(z))$

$\frac{\partial a}{\partial z} = a.(1-a)$

$\frac{\partial \ell}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$

$z = X * W + b$

$\hat{y} = a = \sigma(z)$

$\sigma(z) = \frac{1}{1+e^{-z}}$

$\ell(a, y) = -[\, y * \log(a) + (1-y) * \log(1-a)\,]$

For binary classification:

$\ell(a, y) = -y * \log(a)$

$\Rightarrow$

$\frac{\partial \ell}{\partial w1} = x_1 \cdot \frac{\partial \ell}{\partial z} = x_1 \cdot (a-y)$

$\frac{\partial \ell}{\partial w2} = x_2 \cdot \frac{\partial \ell}{\partial z} = x_2 \cdot (a-y)$

$\frac{\partial \ell}{\partial b} = \frac{\partial \ell}{\partial z} = (a-y)$

$\Rightarrow$

$w_1 = w_1 - \alpha * \frac{\partial \ell}{\partial w_1} = w_1 - \alpha * x_1 * (a-y)$

$w_2 = w_2 - \alpha * \frac{\partial \ell}{\partial w_2} = w_2 - \alpha * x_2 * (a-y)$

$b = b - \alpha * \frac{\partial \ell}{\partial b} = b - \alpha * (a-y)$

Where α is learning rate. The cost function is

$J(W, b) = \frac{1}{m} * (\Sigma\, \ell(a, y))$

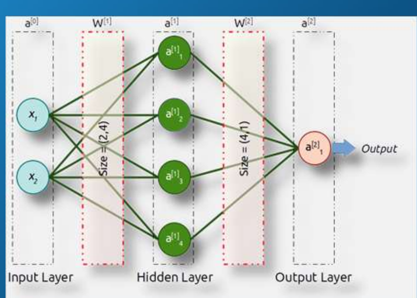
Hence $\frac{\partial J}{\partial w_1} = \frac{1}{m} * (\Sigma \frac{\partial \ell(a, y)}{\partial w_1})$

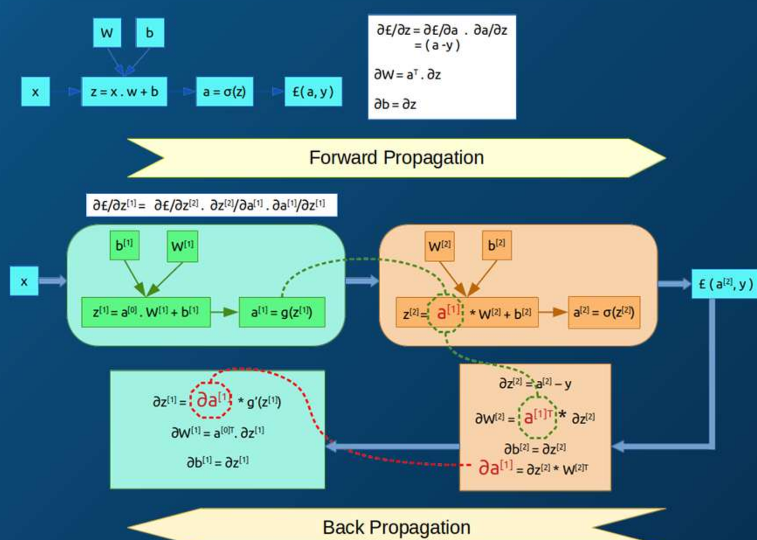
11/22/2024

pra-sâmi

4

Neural Network





11/22/2024

pra-sâmi

5

Activation Function

6

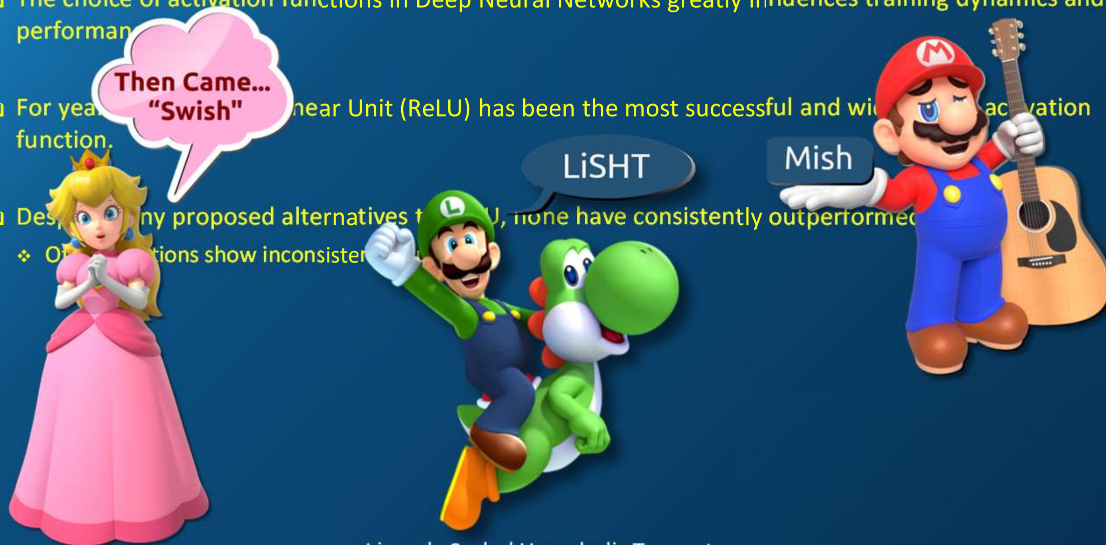
Overview

- The choice of activation functions in Deep Neural Networks greatly influences training dynamics and performance.
- For years, the Rectified Linear Unit (ReLU) has been the most successful and widely-used activation function.
- Despite many proposed alternatives to ReLU, none have consistently outperformed it:
  - Other functions show inconsistent results.

7

Overview

- The choice of activation functions in Deep Neural Networks greatly influences training dynamics and performan
- For yea near Unit (ReLU) has been the most successful and wi ac vation function.
- Des ny proposed alternatives t U, none have consistently outperformed
  - O tions show inconsisten



11/22/2024

pra-sâmi

8

Activation Functions

- An activation function is applied to each neuron in the network, determining whether it should be “fired” based on the relevance of its input for the model’s prediction.
- Activation functions help normalize neuron output to specific ranges:
  - Common ranges include [0, 1], [-1, 1], or other desired intervals.
- They must be computationally efficient as they are calculated for each neuron across all data instances.
- In essence, an activation function acts as a mathematical gate, turning a neuron “on” or “off.”

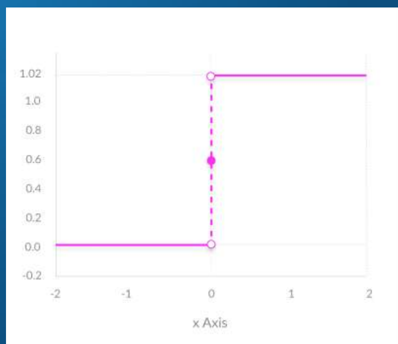
11/22/2024

pra-sâmi

9

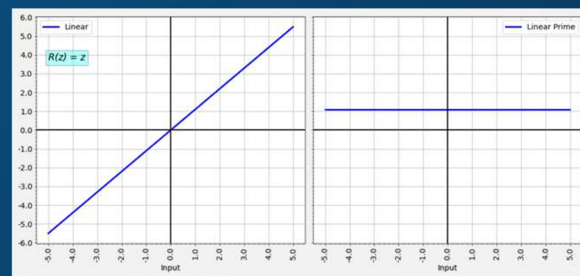
Activation Functions

- Binary Step function



- We already seen this in previous session!!!!

- Linear Activation Function



- That will be simple linear regression!
  - There may still be some use cases…

11/22/2024

pra-sâmi

10

Non-Linear Activation Functions

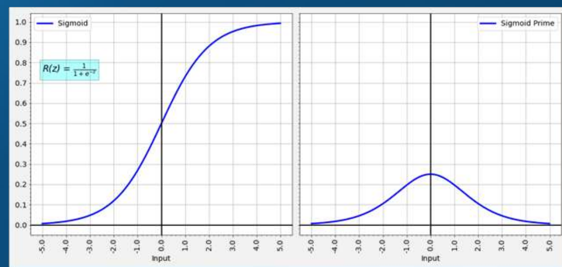
- There are many popular activation functions
  - Sigmoid / Logistic
  - Softmax
  - Tanh (Hyperbolic Tangent)
  - ReLU (Rectified Linear Unit)
  - Leaky ReLU
  - Parametric ReLU
  - ELU (Exponential Linear Unit)
  - GELU (Gaussian Error Linear Unit)
  - Swish
  - Lisht
  - Mish
- Stay tuned… it's an active research area…

11/22/2024

pra-sâmi

11

Sigmoid



- Takes a real-valued input and outputs a value between 0 and 1, i.e., [0,1][0,1].
- Easy to work with and widely used as an activation function.
- Non-linear, continuously differentiable, monotonic, with a fixed output range.
- Well-suited for binary classification tasks.

11/22/2024

pra-sâmi

12

Sigmoid – drawbacks

- Saturation at the ends:
  - The function becomes sluggish towards either end (close to 0 or 1), causing very small gradients and making updates slow.
- Vanishing Gradient Problem:
  - Sigmoid can cause gradients to vanish during backpropagation, especially in deep networks, making it hard for the network to learn effectively.
- Non-zero-centered output:
  - Since the output range is 0<output<10<output<1, it's not zero-centered. This can lead to uneven gradient updates, where weights are adjusted differently for positive and negative inputs, making optimization harder.
- Gradient Saturation:
  - When the sigmoid output saturates near 0 or 1, it "kills" the gradients, leading to stalled learning in certain layers of the network.
- Scaling Requirement:
  - Due to these issues, input values often need to be scaled before being fed into the sigmoid function to avoid saturation and improve learning speed.

11/22/2024

pra-sâmi

13

Softmax Function

- Originates from physics as the Boltzmann or Gibbs distribution; formulated by physicist Ludwig Boltzmann in 1868.
- Applied to reinforcement learning by Robert Duncan Luce in 1959 in his book “Individual Choice Behavior: A Theoretical Analysis.”
- Functionality:
  - Takes a vector of N real-valued inputs and converts it into a vector of N values that sum to 1, effectively creating a probability distribution.
  - Accepts positive or negative inputs and outputs values between 0 and 1.
- Application:
  - Commonly used in the output layer of classification models to represent probabilities for each class.

14

Softmax Function

- Multi-class Logistic Regression: Softmax is essentially an extension of logistic regression for multi-class classification.
- Range: Accepts both positive and negative values as input and outputs values between 0 and 1, i.e., $0 \leq$ output ≤ 1
  - Converts it into a vector of N values that sum to 1, effectively representing probabilities.
- Differentiable: Softmax is differentiable everywhere, making it suitable for backpropagation in neural networks.
- Relation to Sigmoid: The formula for Softmax is similar to Sigmoid;
  - In fact, Sigmoid is a special case of Softmax for binary classification.
- Role in Neural Networks: Softmax is typically used in the final layer of multi-layer neural networks, converting raw scores into probabilities.
- Non-linear Nature: As a non-linear function, it adds non-linearity to the network, which enhances the model’s learning capability.

15

Softmax vs. Sigmoid

- Sigmoid

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- Softmax

$$S(\vec{Z}) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}}$$

- For single class value will be [0, x], Softmax

$$S(\vec{Z}) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}}$$

$$S(\vec{Z}) = \frac{e^{z_1}}{e^{z_1} + e^{z_2}}$$

$$S(X) = \frac{e^{x}}{e^{0} + e^{x}}$$

$$S(X) = \frac{e^{x}}{1 + e^{x}}$$

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

Like Sigmoid Activation function, Vanishing Gradient is still a problem!

11/22/2024

pra-sâmi

16

Softmax vs. Argmax

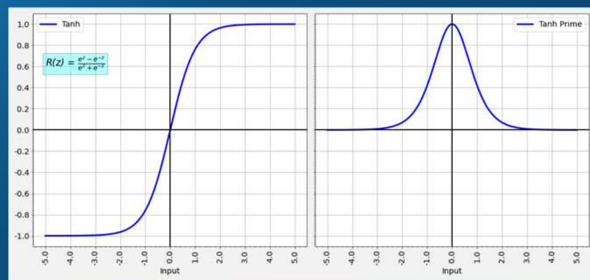
- Both work the same way, Softmax is expected to be a differentiable alternative to argmax
- Argmax returns index of highest value and no idea about other values.
- It is common to train using the Softmax

11/22/2024

pra-sâmi

18

Tanh



- A shifted version of the Sigmoid function, mapping inputs to a range of [−1,1]
- Non-linear and zero-centered: This zero-centered nature helps center data around zero, making it easier for the next layer to learn.
- Stronger Gradient: Tanh has a stronger gradient than Sigmoid, with steeper derivatives, making learning somewhat faster.
- Useful for Hidden Layers: Commonly used in hidden layers to help stabilize the network’s activations.

11/22/2024

19

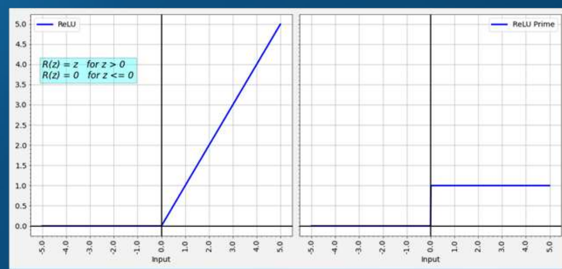
Tanh

- Advantages of the Tanh Function:
  - Handles Negative and Zero Inputs Well: Maps negative inputs to negative values and zero inputs near zero, aiding in better data representation.
  - Differentiable and Monotonic: The function is differentiable, and while it’s monotonic, its derivative is not, which can aid in diverse learning behaviors.
  - Faster Convergence:
    - Steeper gradients than the Sigmoid function enable faster learning.
    - Zero-centered output helps balance the gradients, making optimization easier.
- Disadvantages of the Tanh Function:
  - Vanishing Gradient Problem: Tanh still suffers from the vanishing gradient issue at extreme values, which can slow down learning, especially in deep networks.
  - Mixed Views in Research: Some studies suggest Tanh is better than Sigmoid, while others find limited or conditional advantages. The debate over its effectiveness continues.
- Practical Use: Tanh can be a good starting point in the early design stages for hidden layers in neural networks.

11/22/2024

20

Rectified Linear Units (ReLU)



- Non-linear function (almost)
- Better performance than Sigmoid or Tan in almost all models
- It avoids and rectifies vanishing gradient problem.
- ReLU is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations.
- Suitable for Hidden layers only.

pra-sâmi

21

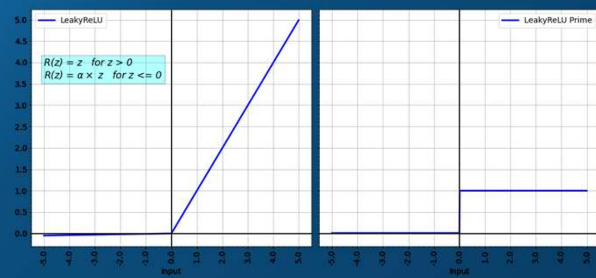
Rectified Linear Units (ReLU)

- Some gradients can be fragile during training and can die.
- For activations in the region (x<0) of ReLu , gradient will be zero
  - Weights will not get adjusted during descent
  - Neurons which go into that state will stop responding to variations in error/ input
  - Dying ReLu problem
- The range of ReLu is [0, ∞]
  - Can blow up the activation

pra-sâmi

22

Leaky ReLU



- ❑ a variant of the ReLU (Rectified Linear Unit) activation function
  - ❖ Attempt to fix the “dying ReLU” problem

- ❑ Leaky ReLU allows a small, non-zero slope (usually a small constant like 0.01) for negative values.
  - ❖ $R(z_i) = \begin{vmatrix} z_i & if\, z_i \geq 0 \\ a_i \cdot z_i & if\ z_i < 0 \end{vmatrix}$

23

Leaky ReLU

- ❑ Prevents Dying Neurons

- ❑ Improves Training Efficiency

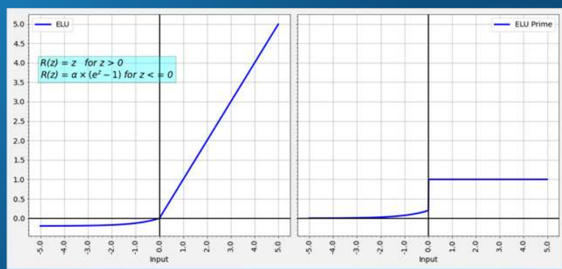
- ❑ Simplicity and Effectiveness

- ❑ Disadvantages:
  - ❖ Small Negative Gradient
  - ❖ Hyperparameter Tuning
  - ❖ Not Zero-Centered
  - ❖ Potential for Exploding Gradients
  - ❖ Limited Improvement over ReLU

- ❑ These limitations are generally minor and can often be mitigated with proper tuning or alternative activation functions, depending on the specific neural network architecture

24

Exponential Linear Unit (ELU)



$$F(z_i) = \begin{vmatrix} z_i \quad if z_i \geq 0 \\ \alpha * (e^{z_i} - 1) \;\; if \; z_i < 0 \end{vmatrix}$$

α is generally 1.0 but can be tuned.

- Converges faster ; Has alpha constant which should be positive number
- ELU is a strong alternative to ReLU.
  - Provides a smooth curve for negative values
- Unlike to ReLU, ELU is zero-centered, as it outputs negative values for negative inputs and positive values for positive inputs, helping improve gradient flow
- For x > 0, it can blow up the activation with the output range of [0, ∞].

11/22/2024

pra-sâmi

25

Exponential Linear Unit (ELU)

Advantages

- Prevents Dying Neurons
  - Allows a small negative output (instead of zero)
- Zero-Centered
  - Can lead to faster convergence
- Smoothness
  - ELU is continuous and differentiable everywhere
- Better Representation for Negative Values
  - Network can learn better representations for data with negative correlations.

Disadvantages

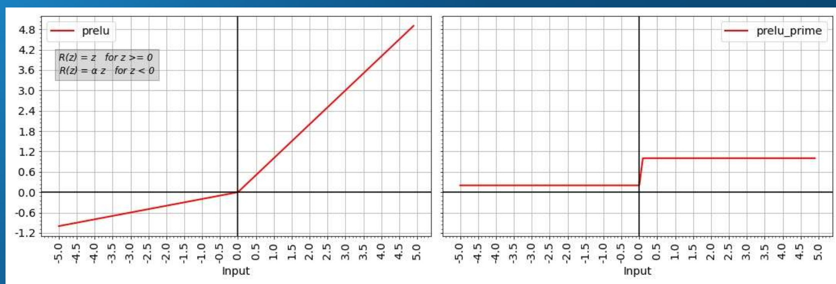
- Computational Cost
  - Requires computing an exponential function
- Potential for Exploding Gradients
  - Uses an exponential function for negative inputs. For very large negative inputs, the function can produce large gradients
- Hyperparameter Tuning
- Slower Convergence
  - It may not always converge as quickly as other activation functions, like Swish or SELU
- Limited Performance in Certain Architectures

11/22/2024

pra-sâmi

26

Parameterized ReLU



$$F(z_i) = \begin{vmatrix} z_i & if\, z_i \geq 0 \\ \alpha * z_i & if\ z_i < 0 \end{vmatrix}$$

α is learned during training.

- A Parametric Rectified Linear Unit, or PReLU, is an activation function that generalizes the traditional rectified unit with a slope for negative values.
- The intuition is that different layers may require different types of nonlinearity.
- α is initialized and then updated during training as part of learning process
- For complex networks this may lead to overfitting.

11/22/2024

27

Parameterized ReLU

- Pick your own parameter (α). Layers learn the parameter.
- In experiments with convolutional neural networks, PReLus for the initial layer have more positive slopes, i.e. closer to linear.
  - Since the filters of the upper layers are edge or texture detectors,
  - This shows a circumstance where positive and negative responses of filters are respected.
- In contrast, deeper layers have smaller coefficients
  - Model becomes more discriminative at later layers
  - While it wants to retain more information at earlier layers.

11/22/2024

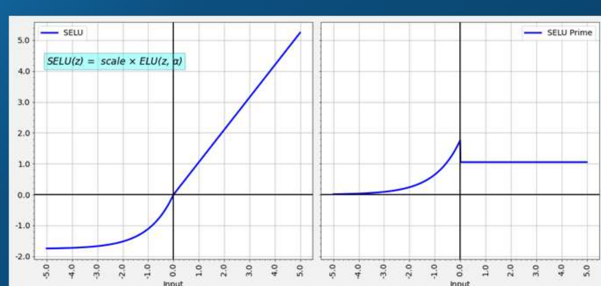
28

Challenges with ReLU

- The consistent problem is that its derivative is 0 for half of the values of the input x in the Function, i.e. f(x)=max(0,x)
- As parameter update algorithm, could used Stochastic Gradient Descent and other optimizers
  - If the parameter itself is 0, then that parameter will never be updated as it just assigns the parameter back to itself
  - Leading close to 40% Dead Neurons in the Neural network environment where z is negative
  - Various substitutes like Leaky ReLU, Prameterized ReLU, have unsuccessfully tried to devoid it of this issue.

29

Scaled ELU (SELU)

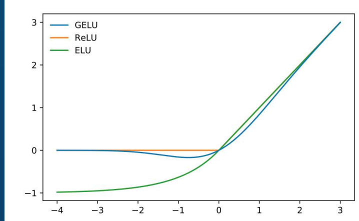


- Activation was introduced in a 2017 paper by Klambauer et al
- Properly initialization, the networks will self-normalize
  - Each layer's output will roughly be zero-centered with standard deviation equal to one
- Helps prevent the vanishing or exploding gradients problems

30

Gaussian Error Linear Unit (GELU)

- Aa smooth, continuous activation function
- Popular in deep learning, especially in transformer-based models like BERT and GPT.
- Combines the benefits of both the ReLU function and probabilistic interpretations based on the Gaussian distribution,
  - Providing a smoother approach to activation.



The GELU ($\mu = 0, \sigma = 1$), ReLU, and ELU($\alpha = 1$)

- Contrary to the ReLU, GELU weights its inputs by their value instead of thresholding them by their sign
- The GELU activation function is x∗ Φ(x),
  - where Φ(x) : the standard Gaussian cumulative distribution function refer scipy's norm.cdf(x)
  - GELU(x) = x P (X <= x) = x Φ(x) $\approx$ 0.5 x [1 +tanh{$\sqrt{2/\pi}$ (X + 0.04475 x^3) }]
  - Or x σ (1.702 x)

31

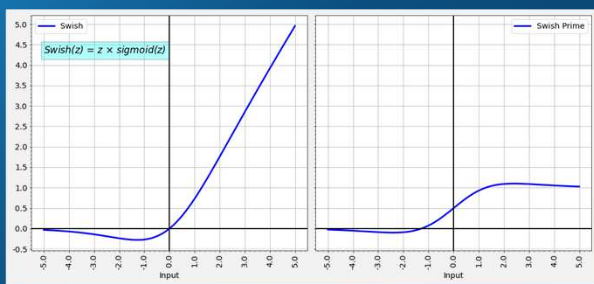
Gaussian Error Linear Unit (GELU)

- Smoother nonlinearity
  - Combination of the gaussian distribution and the error function provides a smoother curve for both positive and negative values
- Probabilistic interpretation
  - Behaves like a soft version of relu
- Zero-centered
- Smooth handling of negative values
  - Probabilistic nature means that negative values are not entirely squashed to zero (like in relu) but instead undergo a smooth transformation

- Advantages:
  - Improved gradient flow
  - Zero-centered output
  - Better performance in complex architectures
- Disadvantages
  - Computational complexity
  - Slower training : due to its smooth, non-linear nature and additional computational overhead, GELU may lead to slower training times compared to relu and its variants
  - Difficult hyperparameter tuning
  - Limited adoption
  - Possible exploding gradients

32

Swish



- Google Brain Team proposed a new activation function:
  - f(x) = x · sigmoid(x)
- Experiments show that Swish tends to work better than ReLU on deeper models across a number of challenging data sets
  - Simply replacing ReLUs with Swish units improves top-1 classification accuracy on ImageNet by 0.9% for Mobile NASNetA and 0.6% for Inception-ResNet-v2
- The simplicity of Swish and its similarity to ReLU make it easy for practitioners to replace ReLUs with Swish units in any neural network.
- Swish is a smooth, non-monotonic function that consistently matches or outperforms ReLU on deep networks

11/22/2024

pra-sâmi

33

Swish

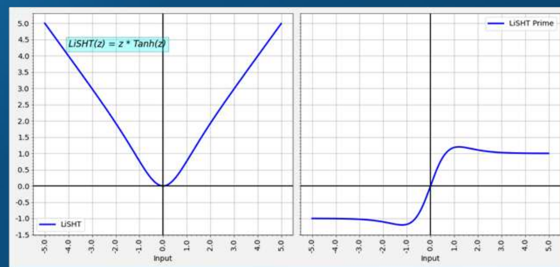
- Unbounded above and bounded below
  - Non-monotonic attribute that actually creates the difference
- We can train deeper Swish networks than ReLU networks when using BatchNorm (Ioffe & Szegedy, 2015) despite having gradient squishing property
- With MNIST data set, when Swish and ReLU are compared, both activation functions achieve similar performances up to 40 layers.
- Swish outperforms ReLU by a large margin in the range between 40 and 50 layers
  - For less than 40 layers, performance is comparable
- In very deep networks, Swish achieves higher test accuracy than ReLU.
- Swish outperforms ReLU on every batch size, suggesting that the performance difference between the two activation functions remains even when varying the batch size.
- Gradient descent problem was still there may be to a lesser degree!

11/22/2024

pra-sâmi

34

LiSHT Activation Function



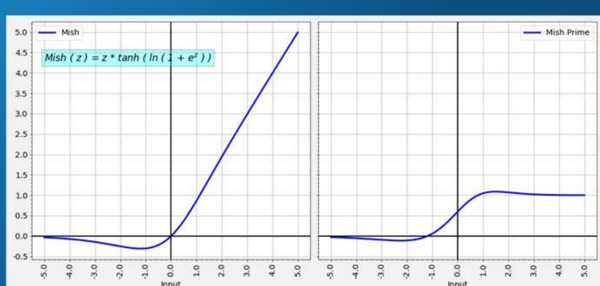
- The function scale the non-linear Hyperbolic Tangent (Tanh) function by a linear function
  - Help tackle the dying gradient problem
- According to paper it has outperformed Swish on a number of problems

11/22/2024

pra-sâmi

35

Mish



f(z) = z * tanh (softplus (z))
= z * tanh (ln (1 + e^z))

- Inspired by Swish and has been shown to outperform it in a variety of computer vision tasks
- Mish was “found by systematic analysis and experimentation over the characteristics that made Swish so effective”.
- Mish seems to be the best activation in stock,
  - But jury is still out

11/22/2024

pra-sâmi

36

Reflect…

- Which of the following is a common activation function used in last layer of deep neural networks?
  - A) Linear activation
  - B) Step function
  - C) Sigmoid function
  - D) Exponential function
- Answer: C) Sigmoid function
- What is the vanishing gradient problem in deep neural networks?
  - A) The problem of too many layers in the network
  - B) The problem of exploding gradients during training
  - C) The problem of slow convergence during training
  - D) The problem of very negligible gradients in early layers
- Answer: D) The problem of very negligible gradients in early layers
- What is the purpose of the softmax activation function in the output layer of a classification neural network?
  - A) To introduce non-linearity
  - B) To convert logits into probabilities
  - C) To prevent overfitting
  - D) To reduce the dimensionality of the output
- Answer: B) To convert logits into probabilities
- In deep learning, what does the term "epoch" refer to during training?
  - A) A complete pass through the training dataset
  - B) The number of layers in the neural network
  - C) The learning rate of the optimizer
  - D) The size of the mini-batch used for training
- Answer: A) A complete pass through the training dataset

11/22/2024

pra-sâmi

37

Reflect…

- What is the role of activation functions in deep neural networks?
  - A) To normalize the input data
  - B) To compute the loss function
  - C) To introduce non-linearity into the network
  - D) To reduce overfitting
- Answer: C) To introduce non-linearity into the network
- What does the term "backpropagation" refer to in the context of neural networks?
  - A) The process of adjusting the weights of the network based on the prediction error
  - B) The process of training the network using labeled data
  - C) The process of selecting the optimal hyperparameters for the network
  - D) The process of initializing the weights of the network
- Answer: A) The process of adjusting the weights of the network based on the prediction error
- Which of the following is NOT a commonly used activation function in deep neural networks?
  - A) Sigmoid
  - B) ReLU (Rectified Linear Unit)
  - C) Tanh (Hyperbolic Tangent)
  - D) Linear
- Answer: D) Linear

11/22/2024

pra-sâmi

38

Next Session…

- Loss / Cost Optimization
- Stochastic Gradient Descent (SGD) and others
- Momentum Learning Rates
- Adaptive Learning Rates

11/22/2024

39

11/22/2024