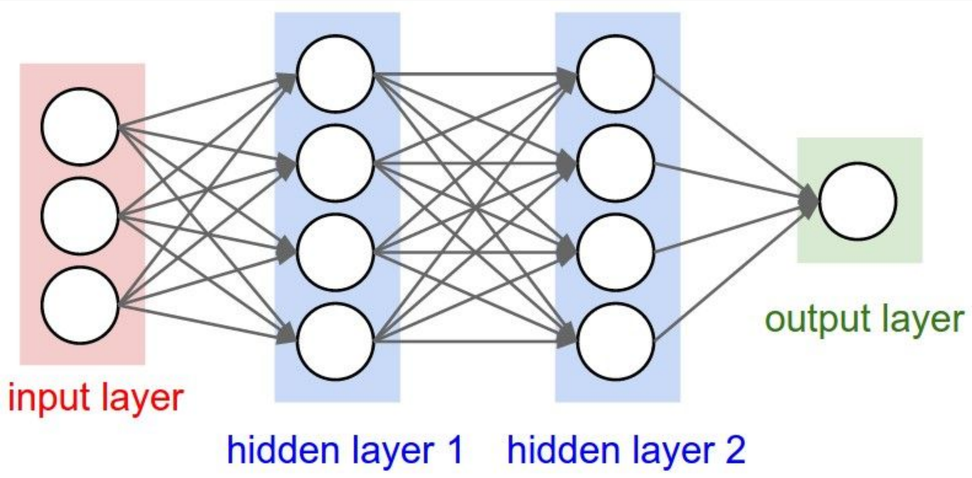
Deep Learning

Activation Functions

Activation Functions in Deep Learning

- Activation function: gets applied to input vector of values from previous layer
- Includes a weight (w) and bias (b)
- Summed over multiple inputs

$$f\left(\sum_i w_i x_i + b\right)$$



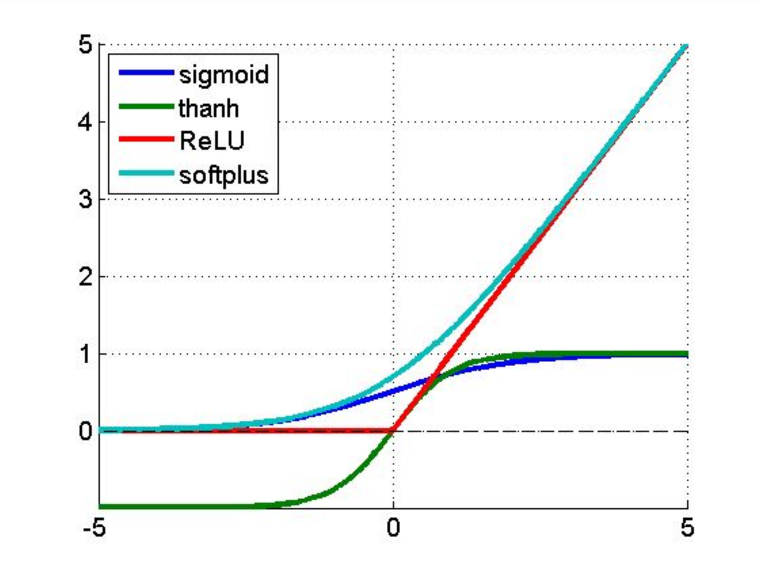
Common Activation Functions

Sigmoid/logistic/softmax functions

Hyperbolic tangent (tanh)

Rectified linear unit (ReLU)

Softplus



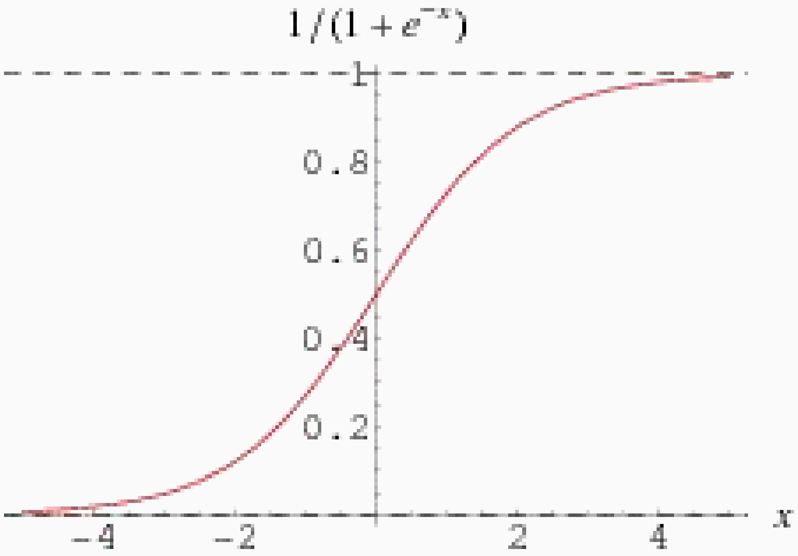
Sigmoid/logistic/softmax

Logistic sigmoid $S(t) = \frac{1}{1+e^{-t}}.$

- Range: (0,1)
- Interpretable as a probability

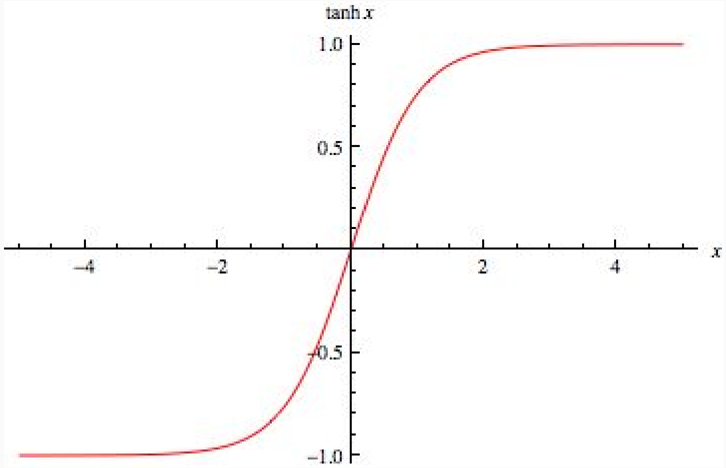
Softmax $\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$

- Generalization for multidimensional vector
- Values sum to 1



Tanh

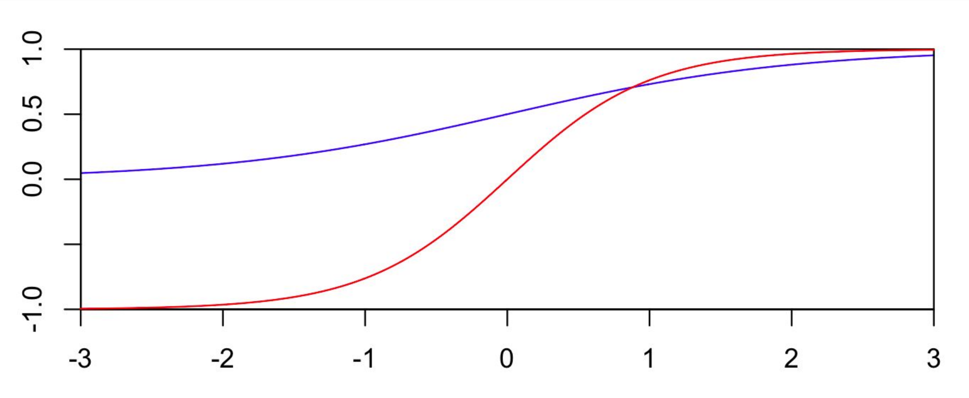
- Range of (-1, 1)
- Simply a rescaled version of logistic sigmoid



$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} =$$
$$= \frac{e^{2x} - 1}{e^{2x} + 1} = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

Comparing sigmoid and tanh

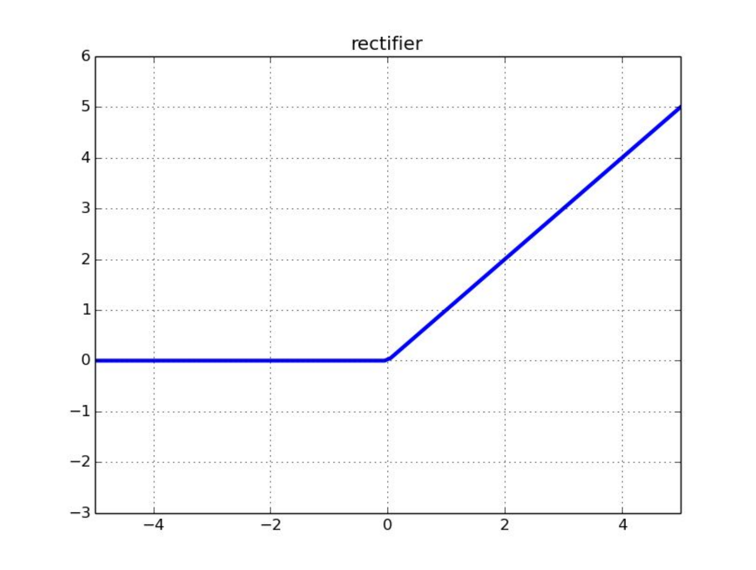
- Centers
  - Sigmoid centered at 0.5
  - Tanh centered at 0
- Gradients
  - Sigmoid gradient range is (0,0.25)
  - Tanh gradient range is (0,1)



ReLU

- Range of (0,∞)
- Biological justification - approximates neuron firing rates
- Mathematical justification - $\sum_{i=1}^{inf} \sigma(x - i + 0.5)$ approximates a “stepped sigmoid”
- Deep learning justification - more efficient and better results

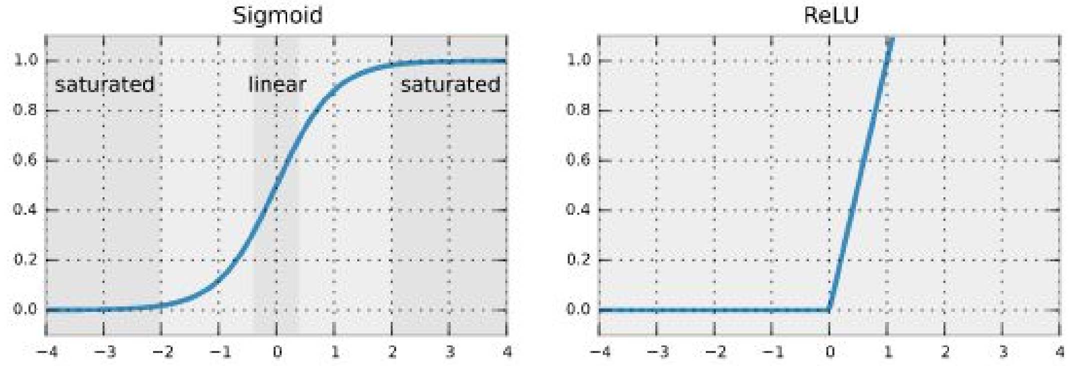
$$f(x) = \max(0, x),$$



Comparing Sigmoid/Tanh and ReLU

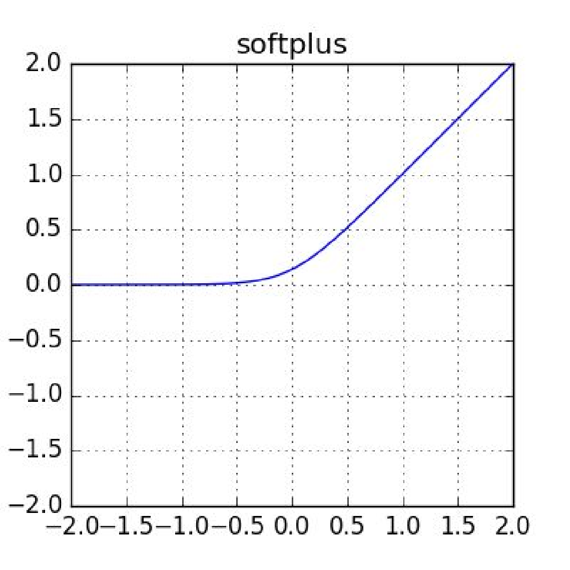
- Easier to calculate activation and gradient
  - See activation functions
  - Gradient: 0 if x < 0, 1 if x > 0
- Sparse neuron activations
  - Simplifies network
  - Can "kill" neurons
- Non-linear near 0

- Saturated neurons
  - In sigmoid, gradient "vanishes" far away from 0
  - Large absolute values cause "saturation," where the gradient is too small for learning



Softplus

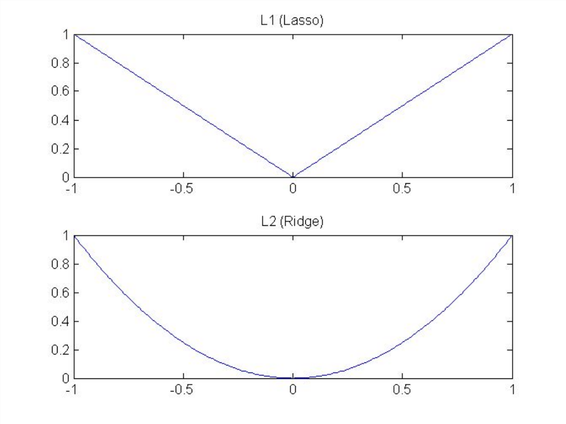
- Range of (0,∞)
- Approximation of "stepped sigmoid"
- Smooth approximation of ReLU

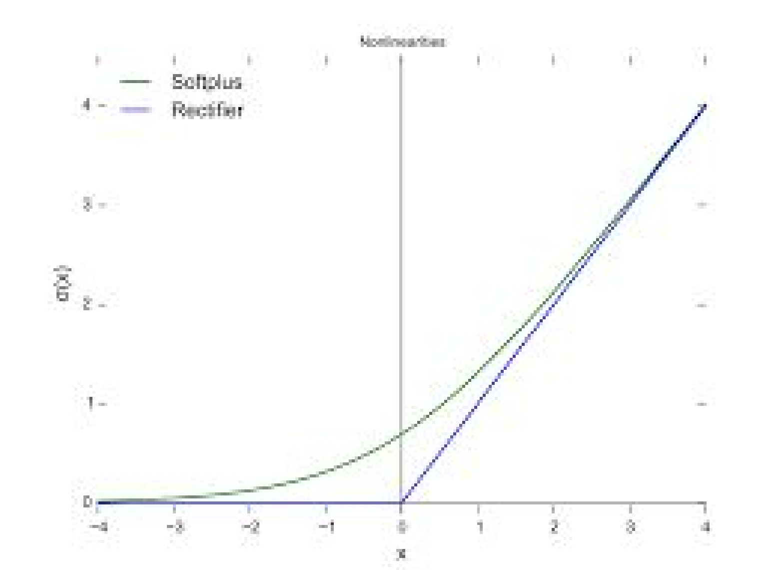


$$f(x) = \ln(1 + e^x),$$

Comparing ReLU and Softplus

- Computational cost
- Sparsity - similar to L1 vs L2 regularization





Layers for Activation Functions

- Hidden layers
  - Considerations: computationally cheap, reliable gradient, sparsity
  - Typical choice: ReLU
- Output layers
  - Considerations: output range, interpretability
  - Typical choice: softmax/sigmoid (or linear for continuous outputs)
- Gating layers
  - Considerations: definite output range
  - Typical choice: sigmoid

Other Activation Functions

- Leaky ReLU - small slope where $x < 0$ to avoid dead neurons
- Exponential linear unit (ELU) - exponential where $x < 0$
- Maxout
    - Max of multiple (linear?) activation functions
    - Generalization of ReLU functions

References

CS231n Convolutional Neural Networks for Visual Recognition
Efficient BackProp - Yann LeCun
Rectified Linear Units Improve Restricted Boltzmann Machines
Deep Sparse Rectifier Neural Networks
Rectifier Nonlinearities Improve Neural Network Acoustic Models