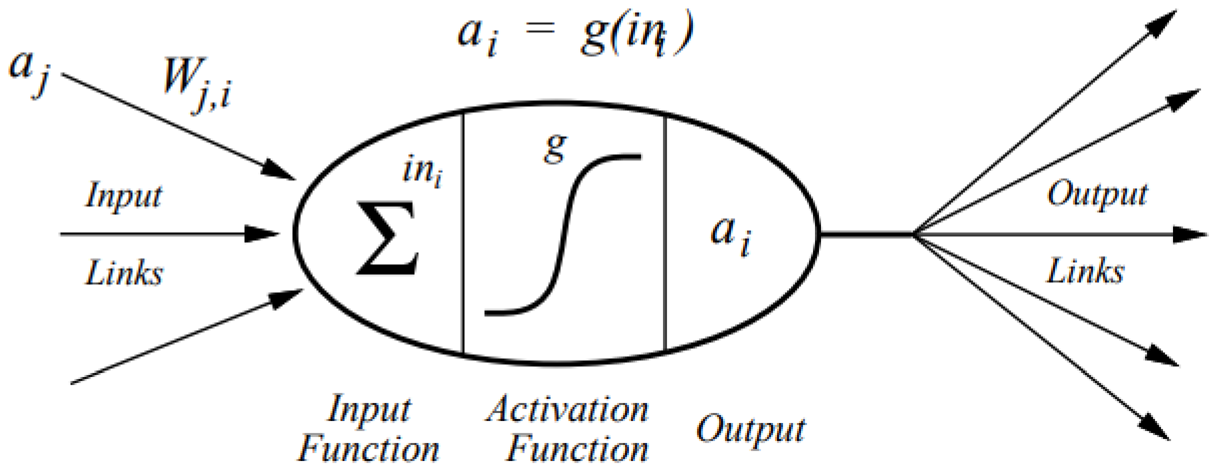
INTRODUCTION TO MACHINE LEARNING

NEURAL NETWORKS I

Mingon Kang, Ph.D.
Department of Computer Science @ UNLV

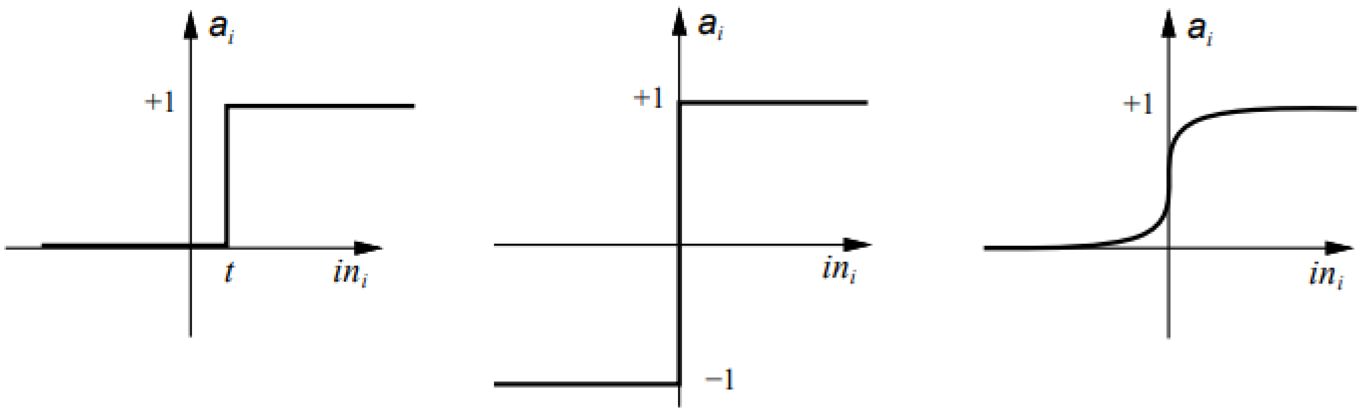
Begin with a single Perceptron



$$a_i = g(\sum_j W_{j,i} a_j)$$

Ref: Dr. Manuela Veloso, CMU

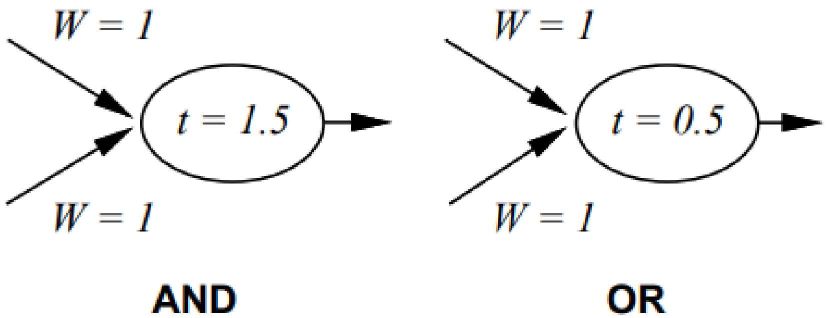
Threshold Functions

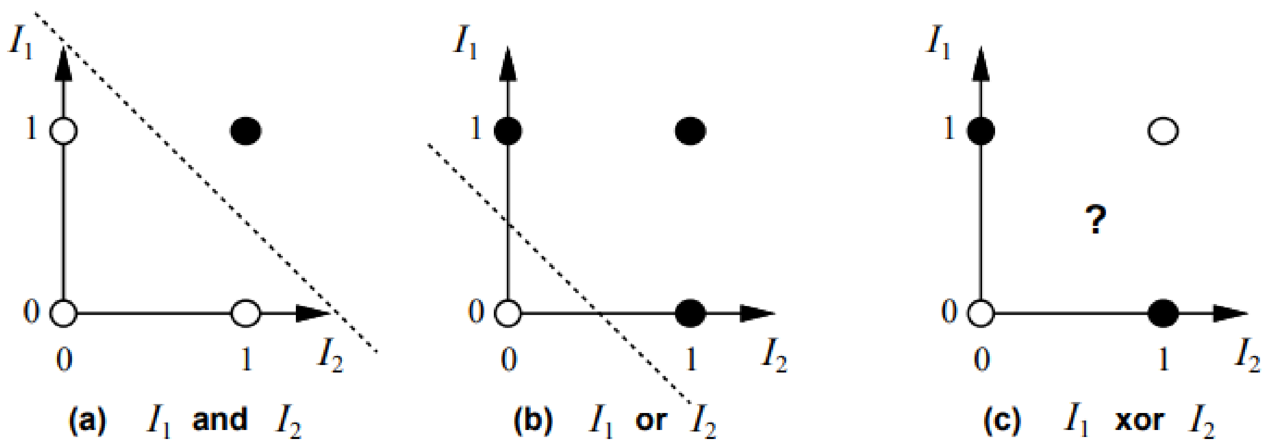


(a) Step function (b) Sign function (c) Sigmoid function

Ref: Dr. Manuela Veloso, CMU

Boolean Functions and Perceptron





Ref: Dr. Manuela Veloso, CMU

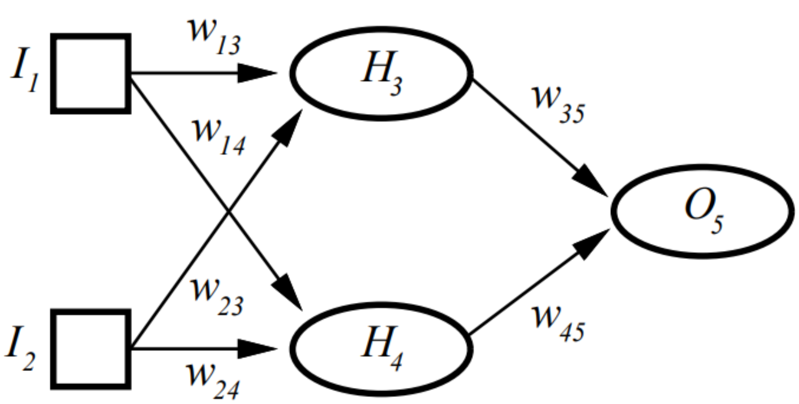
Discussion in Perceptrons

- A single perceptron works well to classify a linearly separable set of inputs
- Multi-layer perceptrons
  - found as a "solution" to represent nonlineaerly separable functions (1950s)
- Many local minima – non-convex
- Research in neural networks stopped until the 70s

Ref: Dr. Manuela Veloso, CMU

XOR?

- $\mathbf{X} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, g(x) = \max(x, 0)$
- $\mathbf{W}_1 = [1, 1; 1,1], \mathbf{W}_2 = [1, -2], \mathrm{c} = [0, -1, 0]$



Neural Network

- Feedforward Networks or Multilayer Perceptrons (MLPs)
  - E.g., for a classifier

  $y = f^*(\boldsymbol{x}; \boldsymbol{\theta})$ maps an input $\boldsymbol{x}$ to a category y

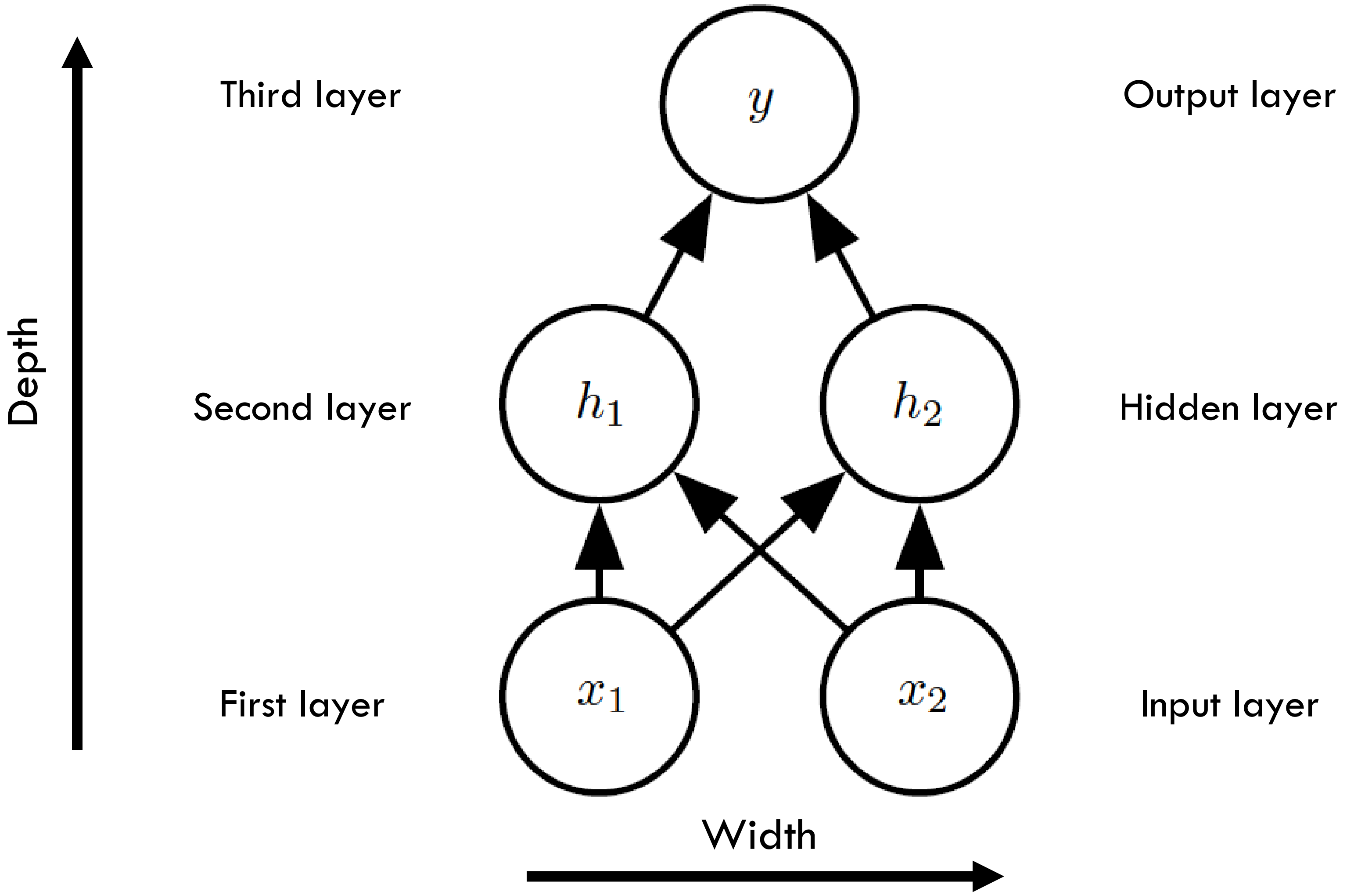
  A feedforward network defines a mapping $y = f(\boldsymbol{x}; \boldsymbol{\theta})$ and learns the optimal parameters $\boldsymbol{\theta}$.

Neural Network

- Called Networks
  - Involve many various functions
  - Associated with a directed acyclic graph that represent how the functions are composed together
  - Mostly, chain structures
    - $f(x) = f^{(3)}\left(f^{(2)}\left(f^{(1)}(x)\right)\right)$

    $f^{(i)}$: the i-th layer in the network

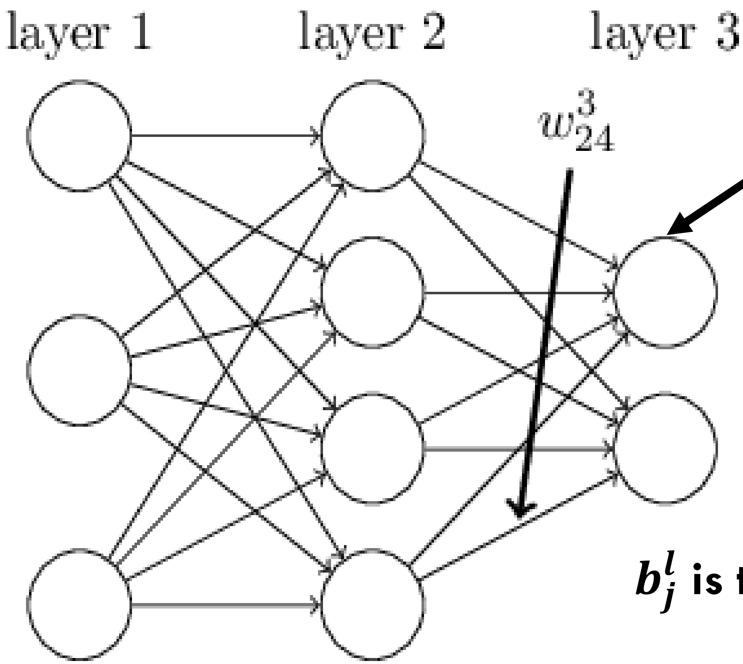
Neural Network



Design of a NN model

- Architecture of the network
  - How many layers
  - How these layers are connected to each other
  - How many units are in each layer
- Activation function: to compute the hidden layer values
- Cost function: to optimize the model
- Optimizer: how to optimize the model

Weights and Bias



b^3_1

Labeling the Weights of a Neural Network

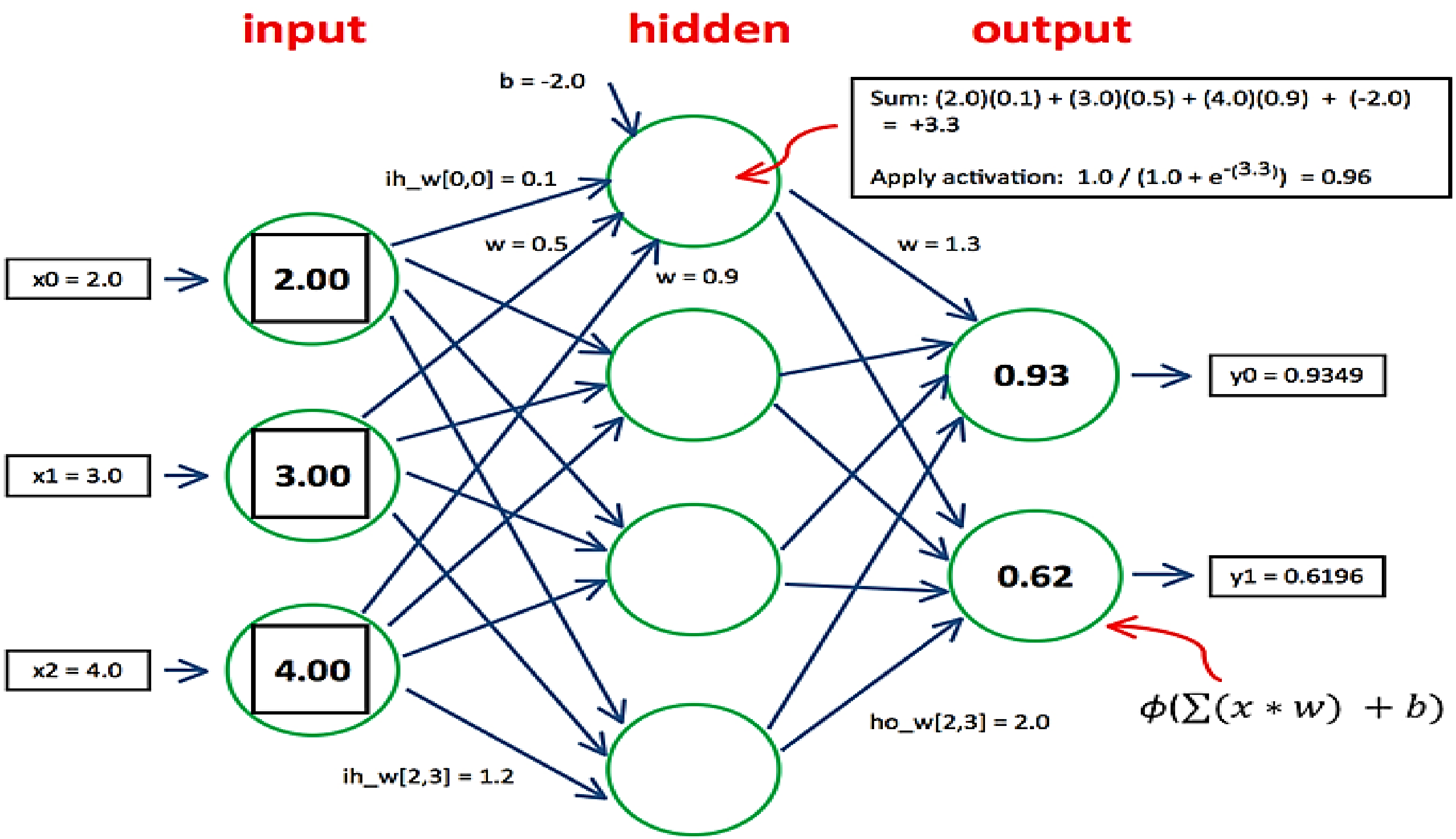
w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

b^l_j **is the bias of the** j^{th} **neuron of the** l^{th} **layer**

Weighted Input and Activation of a Neuron

- Weighted Input of the j^{th} neuron of the l^{th} layer is z_j^l:
  - $z_j^l = \sum_k \left(\left(w_{jk}^l \ a_k^{l-1} \right) + \ b_j^l \right)$
- Activation from the j^{th} neuron of the l^{th} layer is a_j^l:
  - $a_j^l = f\left(z_j^l \right)$, where f is the activation function
  - $a_j^l = f\left(\sum_k \left(\left(w_{jk}^l \ a_k^{l-1} \right) + \ b_j^l \right) \right)$
- In matrix notation, the activation becomes:
  - $a^l = f\left(w^l \ a^{l-1} + \ b^l \right)$

Example



Ref: https://visualstudiomagazine.com/articles/2013/05/01/neural-network-feed-forward.aspx

Activation Functions

- Sigmoid Activation: $a_j^l = \sigma(z_j^l) = \frac{1}{1+e^{-z_j^l}}$

- Softmax Activation: $a_j^l = \frac{e^{z_j^l}}{\sum_k e^{z_k^l}}$

- Tanh Activation: $tanh(z_j^l) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

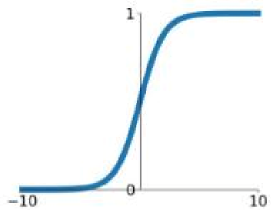
- Rectified Linear Activation: $max(0, z_j^l)$, maximum of $0\ or\ z_j^l$

Activation Functions

Activation Functions

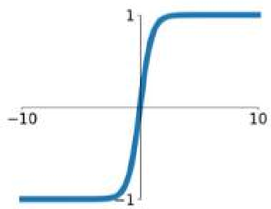
Sigmoid

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



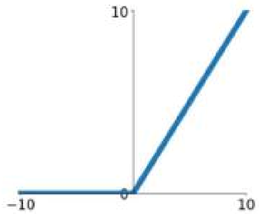
tanh

$\tanh(x)$



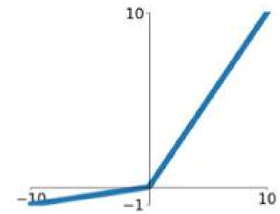
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

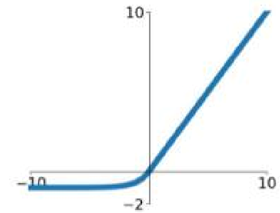


Maxout

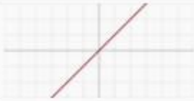
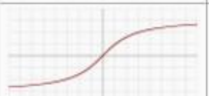
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

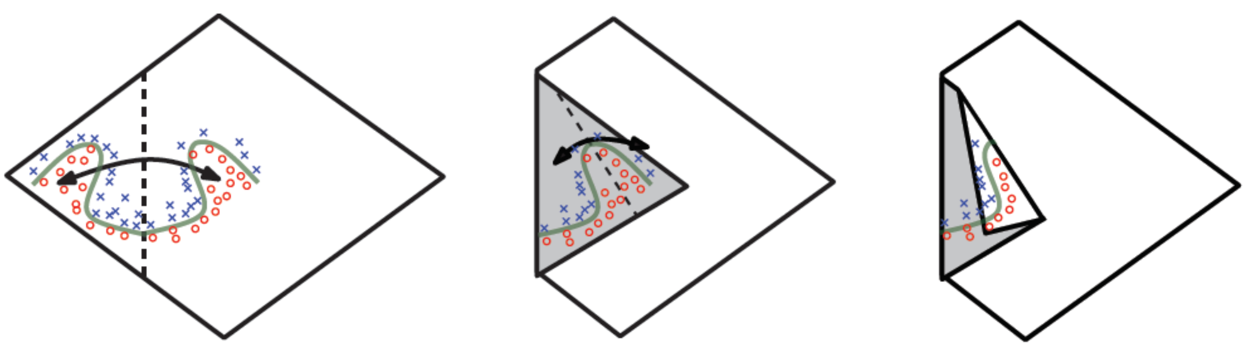


Activation Functions

Name	Plot	Equation	Derivative
Identity		$f(x)=x$	$f'(x)=1$
Binary step		$f(x)=\begin{cases}0 & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}0 & \text{for } x\neq 0\\ ? & \text{for } x=0\end{cases}$
Logistic (a.k.a Soft step)		$f(x)=\frac{1}{1+e^{-x}}$	$f'(x)=f(x)(1-f(x))$
TanH		$f(x)=\tanh(x)=\frac{2}{1+e^{-2x}}-1$	$f'(x)=1-f(x)^2$
ArcTan		$f(x)=\tan^{-1}(x)$	$f'(x)=\frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x)=\begin{cases}0 & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}0 & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x)=\begin{cases}\alpha x & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}\alpha & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x)=\begin{cases}\alpha(e^x-1) & \text{for } x<0\\ x & \text{for } x\geq 0\end{cases}$	$f'(x)=\begin{cases}f(x)+\alpha & \text{for } x<0\\ 1 & \text{for } x\geq 0\end{cases}$
SoftPlus		$f(x)=\log_e(1+e^x)$	$f'(x)=\frac{1}{1+e^{-x}}$

Universal Approximator Theorem

- One hidden layer may be enough to represent (not learn) an approximation of any function to an arbitrary degree of accuracy
- So why deeper?
  - Shallow net may need (exponentially) more width
  - Shallow net may overfit more



Cost Functions

- Quadratic Cost: $C = \frac{1}{2n} \sum_x ||y(x) - a^L(x)||^2$
- Binary Cross-Entropy Cost: $C = -\frac{1}{n} \sum_x \left(y(x) \times \log_e\left(a^L(x)\right) + (1 - y) \times \log_e\left(1 - a^L(x)\right)\right)$
- Negative Log-likelihood Cost: $C = -\log_e\left(a_y^L\right)$
- A Cost function must satisfy the following two conditions (for backpropagation):
  - The Cost function C should be calculated as an average over the cost functions C_x for individual training examples.
  - The cost functions for the individual training examples C_x and consequently the Cost C function must be a function of the outputs of the neural network.

Examples of Neg Log-likelihood Cost

- Given the posterior probability and the ground truth:
  - A set of output probabilities: e.g. [0.1, 0.3, 0.5, 0.1]
  - Ground truth: e.g., [0, 0, 0, 1]
- Likelihood
  - 0*0.1 + 0*0.3 + 0*0.5 + 1*0.1 = 0.1
  - NLL: -ln(0.1) = 2.3
- If ground truth is [0, 0, 1, 0]
  - 0*0.1 + 0*0.3 + 1*0.5 + 0*0.1 = 0.5
  - NLL: -ln(0.5) = 0.69

Output Types

Output Type	Output Distribution	Output Layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary cross-entropy
Discrete	Multinoulli	Softmax	Discrete cross-entropy
Continuous	Gaussian	Linear	Gaussian cross-entropy (MSE)
Continuous	Mixture of Gaussian	Mixture Density	Cross-entropy
Continuous	Arbitrary	See part III: GAN, VAE, FVBN	Various

Discussion

- Construct a neural network for MNIST data
  - 28*28 pixel images and 10 labels
- Construct a neural network for a binary classification problem
  - 10 features and 2 labels
- Construct a neural network for predicting tomorrow’s temperature in degree
  - 10 variables