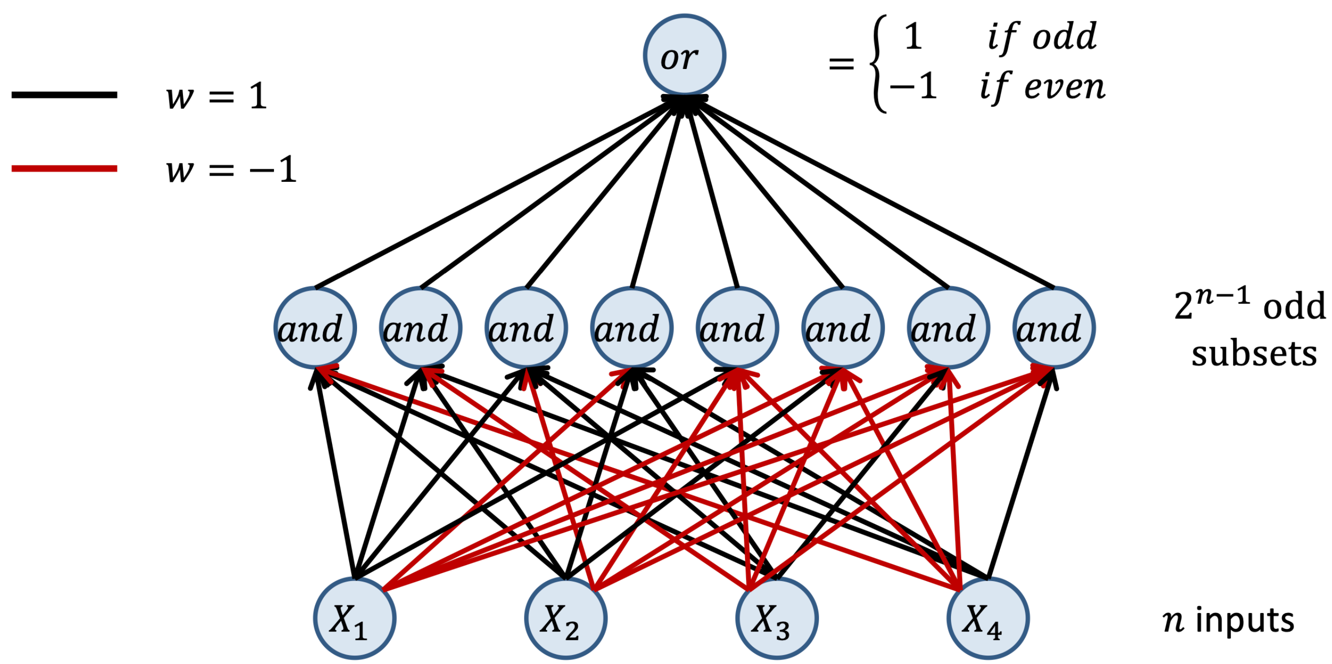
CS 4824/ECE 4424: Deep Neural Networks II

Acknowledgement:
Many of these slides are derived from Tom Mitchell, Pascal Poupart, Pieter Abbeel, Eric Eaton, Carlos Guestrin, William Cohen, and Andrew Moore.

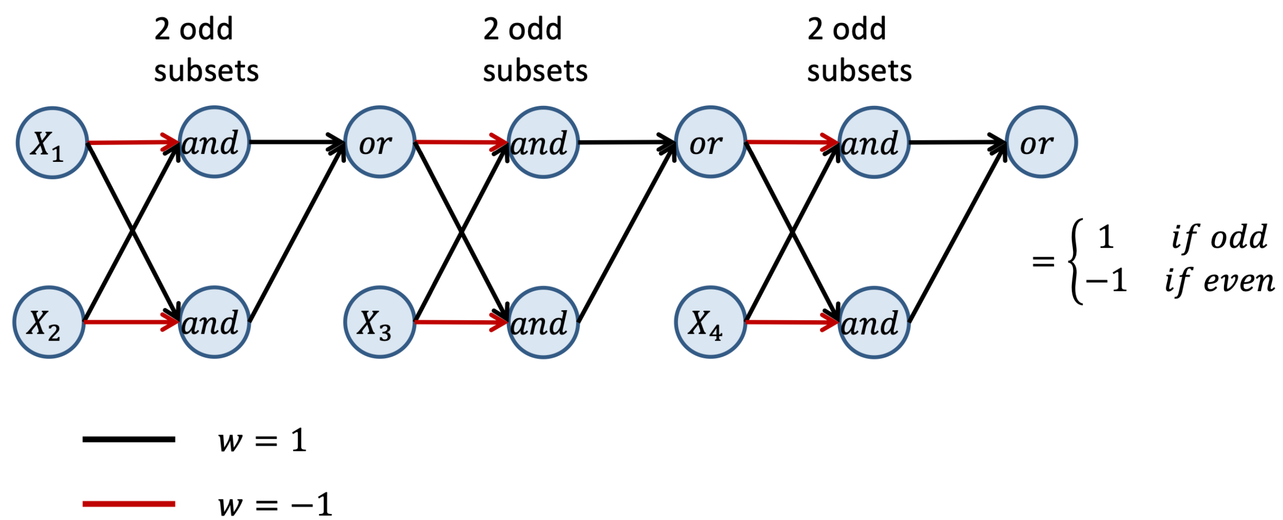
Parity Function — from shallow to deep neural network

◦ Deep neural networks of sigmoid and hyperbolic units often suffer from vanishing gradients



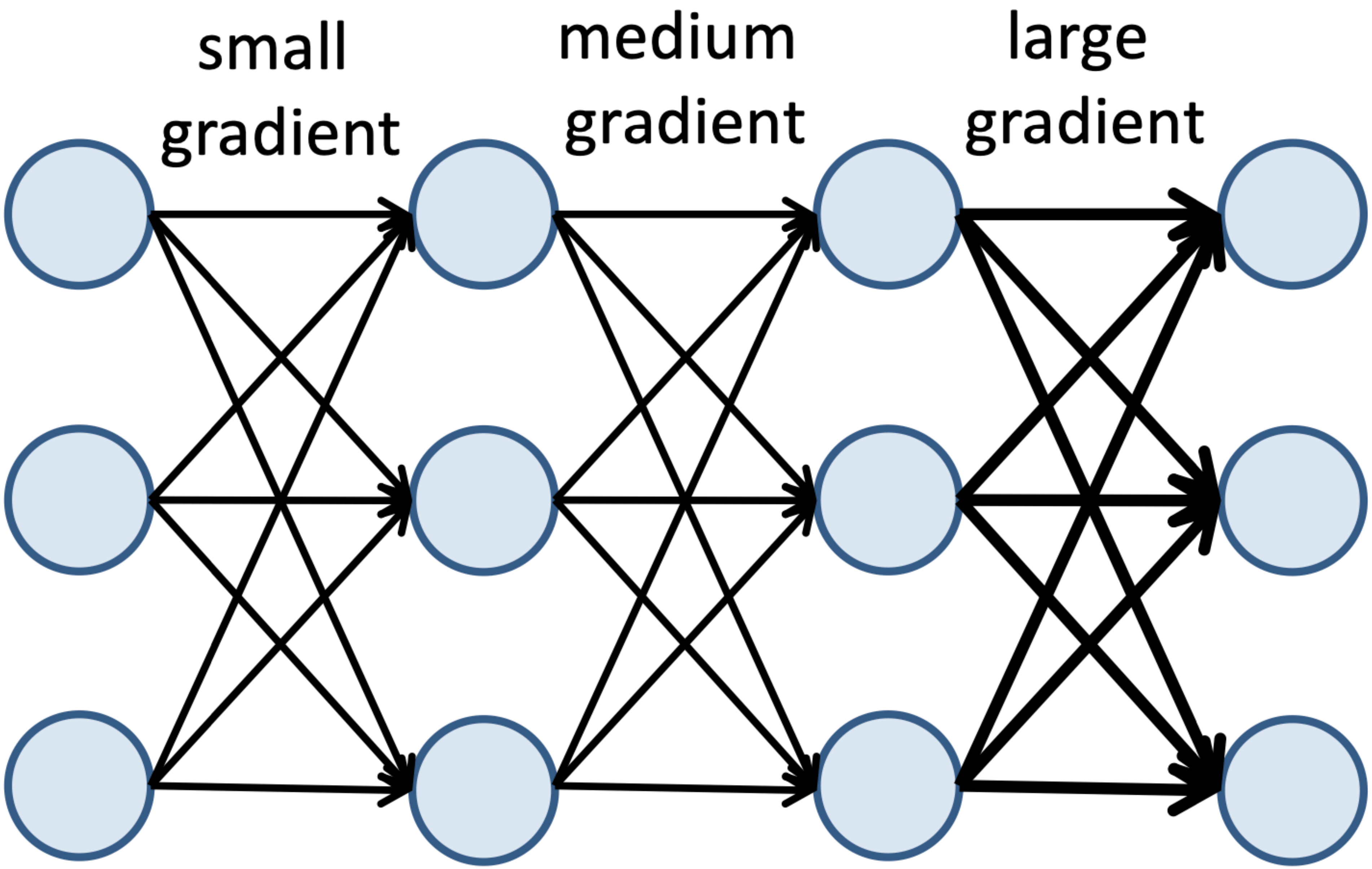
single layer of hidden nodes
(shallow architecture)

2n – 2 layers of hidden nodes
(deep architecture)



Vanishing Gradients

- Deep neural networks often suffer from vanishing gradients



Common activation functions

Sigmoid

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

$$\sigma'(a) = \sigma(a)(1 - (\sigma(a))$$

Tanh

$$h(a) = tanh(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$$

$$tanh'(a) = (1 - (tanh(a))^2$$

Softmax

$$h(\mathbf{a}) = \sigma(\mathbf{a})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Generalization of sigmoid/logistic f^n to multiple dimension

Simple Example

$$y = \sigma\left(w_4\, \sigma\left(w_3\, \sigma(w_2\, \sigma(w_1\, x))\right)\right)$$

$$x \xrightarrow{w_1} h_1 \xrightarrow{w_2} h_2 \xrightarrow{w_3} h_3 \xrightarrow{w_4} y$$

- Common weight initialization in (0,1) or in (-1, 1)
- Sigmoid function and its derivative always less than 1
- This leads to vanishing gradients:

$$\frac{\partial y}{\partial w_4} = \sigma'(a_4)\sigma(a_3)$$

$$\frac{\partial y}{\partial w_3} = \sigma'(a_4)w_4\sigma'(a_3)\sigma(a_2)$$

$$\frac{\partial y}{\partial w_2} = \sigma'(a_4)w_4\sigma'(a_3)w_3\sigma'(a_2)\sigma(a_1)$$

$$\frac{\partial y}{\partial w_1} = \sigma'(a_4)w_4\sigma'(a_3)w_3\sigma'(a_2)w_2\sigma'(a_1)x$$

As products of factors less than 1 gets longer, gradient vanishes

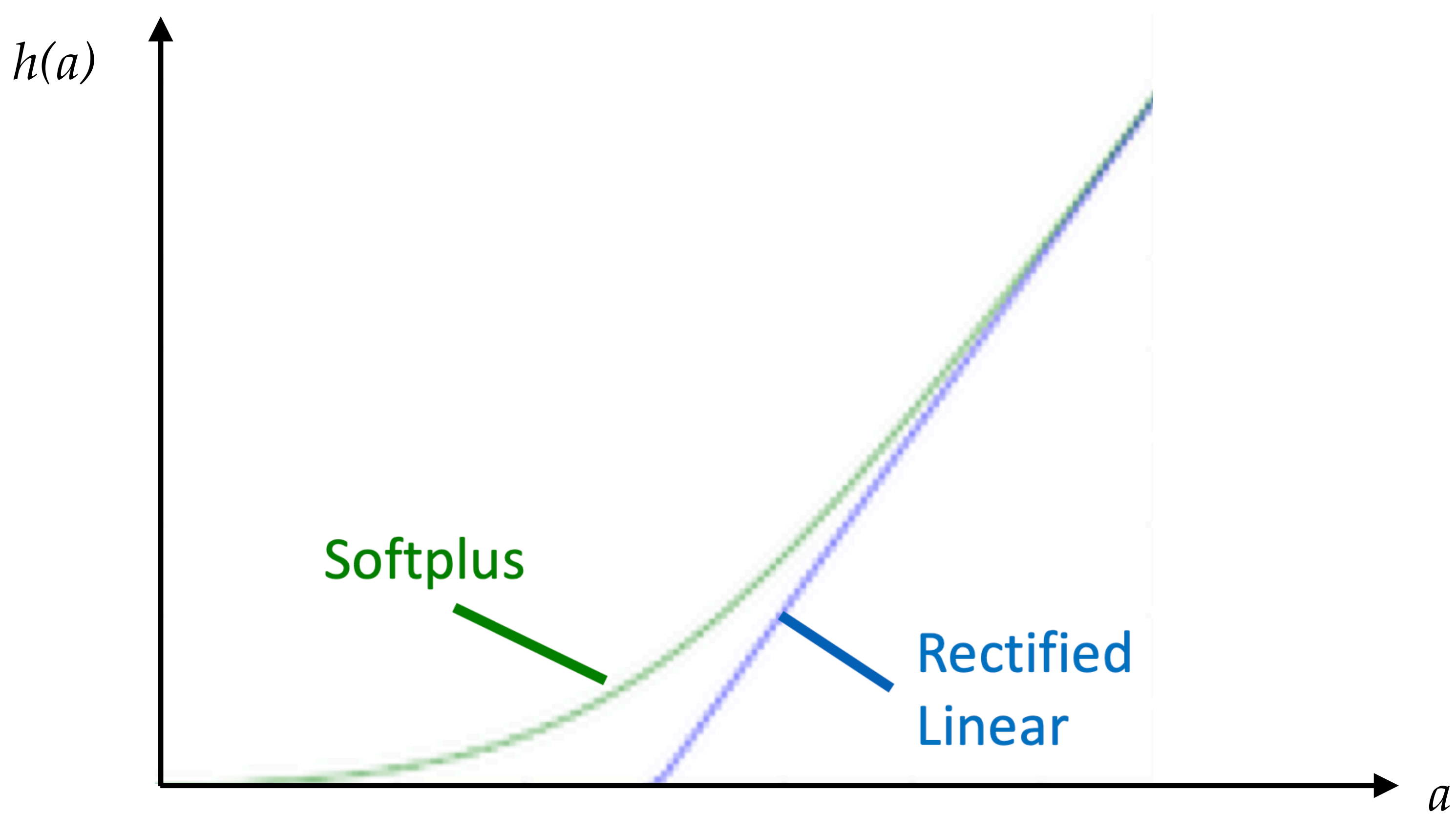
Avoiding Vanishing Gradients

- Several popular solutions:
  - Pre-training
  - **Rectified linear units and maxout units**
  - Skip connections
  - Batch normalization

Rectified Linear Units (ReLU)

- **Rectified linear**: $h(a) = \max(0,a)$
  - Gradient is 0 or 1 w.r.t. a
  - Sparse computation

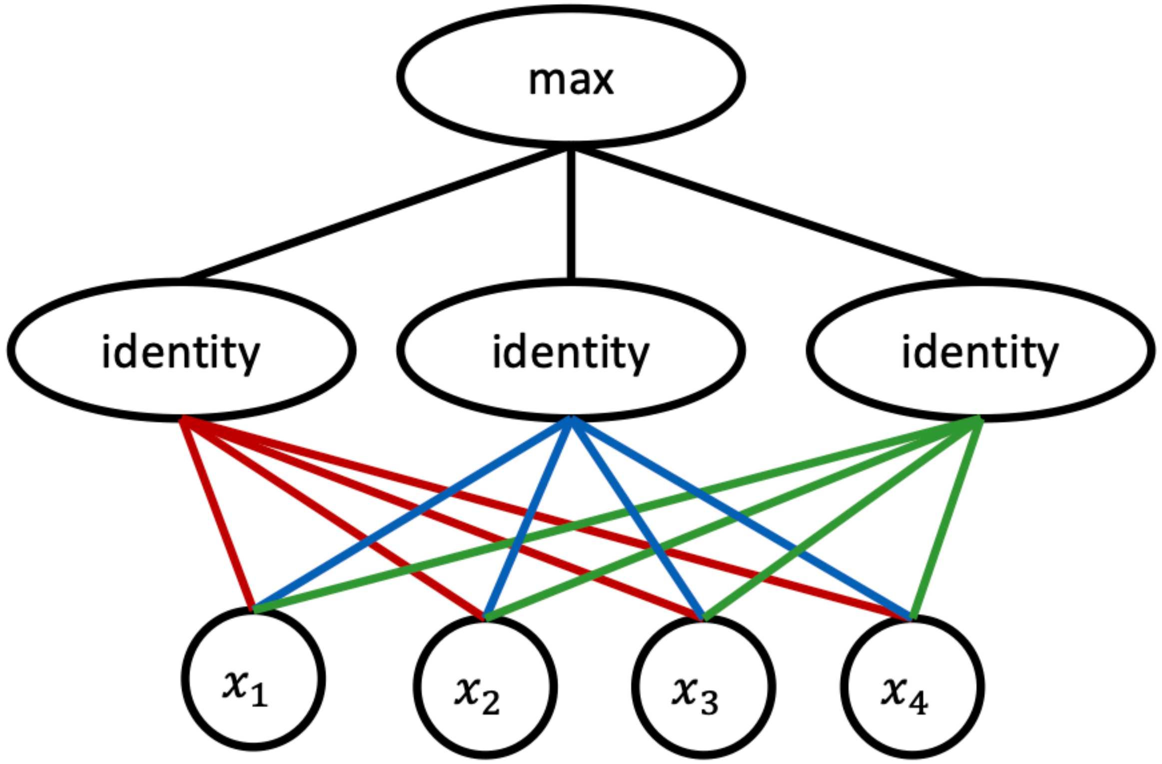
- **Soft version ("softplus")**: $h(a) = \log(1 + e^{a})$
  - But softplus does not prevent gradient vanishing (gradient < 1)
  - Making rectified linear unit smooth does not help!



Maxout Units

- Generalization of rectified linear units

$$\boldsymbol{max}\left\{\sum_i \boldsymbol{w_i^{(1)}} \boldsymbol{x_i}, \sum_i \boldsymbol{w_i^{(2)}} \boldsymbol{x_i}, \sum_i \boldsymbol{w_i^{(3)}} \boldsymbol{x_i}, \dots\right\}$$



Overfitting

- High expressivity increases the risk of overfitting
  - # of parameters is often larger than the amount of data

- Some solutions:
  - Regularization
  - **Dropout**
  - Data augmentation

Dropout — Training

- **Idea**: randomly "drop"some units from the network when training

- Training: at each iteration of gradient descent
  - Each input unit is dropped with probability p_1 (e.g., 0.2)
  - Each hidden unit is dropped with probability p_2 (e.g., 0.5)

Dropout — Prediction

- **Idea**: during prediction, probabilistically account for the effect of randomly "dropped" units from the network during training

- Prediction(testing):
  - Multiply each input unit by $1 - p_1$
  - Multiply each hidden unit $1 - p_2$

Dropout — Intuition

- Dropout can be viewed as an approximate form of ensemble learning
- In each training iteration, a different subnetwork is trained
- At test time, these subnetworks are “merged” by averaging their weights

Speech

- **2006** (Hinton and coworkers): first effective algorithm for deep NN
  - layer-wise training of Stacked Restricted Boltzmann Machines (SRBM)s
- **2009**: Breakthrough in acoustic modeling
  - replace Gaussian Mixture Models by SRBMs
  - Improved speech recognition at Google, Microsoft, IBM
- **2013**: recurrent neural nets (LSTM)
  - Google error rate: **23**% (2013) to **8**% (2015)
  - Microsoft error rate: **5.9**% (Oct 17, 2016) same as human performance
- …

Image Classification

- ImageNet Large Scale Visual Recognition Challenge