

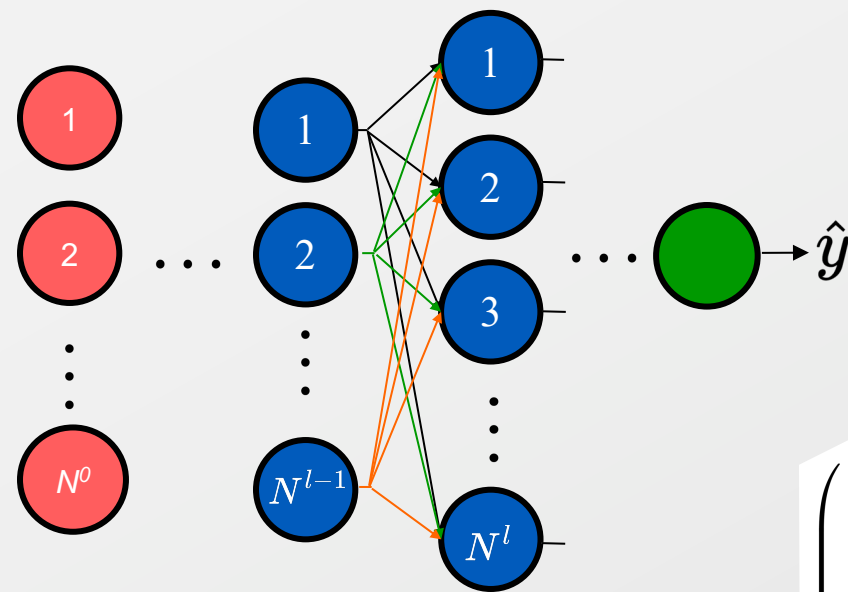
# Lecture 22

## Neural Networks 2

Backpropagation; case studies

*Prof. David A. Kofke  
CE 500 – Modeling Potential-Energy Surfaces  
Department of Chemical & Biological Engineering  
University at Buffalo*

# Here is a summary of the neural-network structure and notation

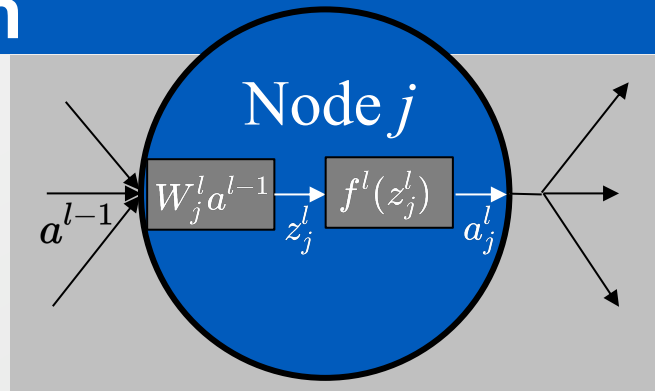


Input;  
layer 0  
 $\mathbf{x} \equiv a^0$

layer  $l-1$       layer  $l$

Output;  
layer  $\mathcal{L}$

Hidden layers



$$\begin{pmatrix} a_1^l \\ a_2^l \\ \vdots \\ a_{N^l}^l \\ 1 \end{pmatrix} = \begin{pmatrix} f^l(z_1^l) \\ f^l(z_2^l) \\ \vdots \\ f^l(z_{N^l}^l) \\ 1 \end{pmatrix}$$

activation  
function

$$a^l = f^l(z^l) \\ = f^l \circ z^l$$

$$\begin{pmatrix} z_1^l \\ z_2^l \\ \vdots \\ z_{N^l}^l \end{pmatrix} = \begin{pmatrix} w_{11}^l & w_{12}^l & \cdots & w_{1(N^{l-1}+1)}^l \\ w_{21}^l & w_{22}^l & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ w_{N^l 1}^l & \cdots & \cdots & w_{N^l(N^{l-1}+1)}^l \end{pmatrix} \begin{pmatrix} a_1^{l-1} \\ a_2^{l-1} \\ \vdots \\ a_{N^{l-1}}^{l-1} \\ 1 \end{pmatrix}$$

$z^l = W^l a^{l-1}$

Weight  
indexes  $w_{jk}^l$

Layer number  $l$

Number of node  
receiving input  
in layer  $l$

Number of node  
providing output  
from layer  $l-1$

# Derivatives of loss function with respect to parameters are needed for learning

- Loss function

$$\theta(\text{new}) = \theta(\text{old}) - \gamma \frac{\partial L}{\partial \theta}$$

$$L(W^1, \dots, W^L) \equiv \sum_{\text{samples}, i}^{n_{\text{batch}}} L_i(\hat{y}_i) \equiv \sum_i L_i \circ \hat{y}_i \equiv \sum_i (y_i - \hat{y}_i)^2 \equiv \sum_i (y_i - \hat{y}(\mathbf{x}_i; W^1, \dots, W^L))^2$$

$\theta$  is a generic symbol for a NN parameter. In our case, it is the  $W$  matrix index,  $i$

- Derivative matrix represents all derivatives for layer  $l$

$$\frac{\partial L}{\partial W^l} \equiv \begin{pmatrix} \frac{\partial L}{\partial w_{11}^l} & \cdots & \frac{\partial L}{\partial w_{1(N^{l-1}+1)}^l} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial w_{N^l 1}^l} & \cdots & \frac{\partial L}{\partial w_{N^l(N^{l-1}+1)}^l} \end{pmatrix} = \sum_i \frac{\partial L_i}{\partial W^l} = \sum_i \frac{\partial L_i}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial W^l}$$

$$\frac{\partial L_i}{\partial \hat{y}_i} \equiv L'_i \circ \hat{y}_i = -2(y_i - \hat{y}_i)$$

# $\hat{\mathbf{y}}$ is given as a series of nested operations

$$\hat{\mathbf{y}} = W^{\mathcal{L}} f^{\mathcal{L}-1} (W^{\mathcal{L}-1} \dots f^2 (W^2 f^1 (W^1 \mathbf{x})))$$

(define notation)

$$\equiv W^{\mathcal{L}} f^{\mathcal{L}-1} \circ W^{\mathcal{L}-1} \dots f^2 \circ W^2 f^1 \circ W^1 \mathbf{x}$$

– Expression can end with any  $a$  or  $z$

$$\hat{y}_i = \hat{y}(\mathbf{x}_i) = z^{\mathcal{L}}(\mathbf{x}_i) \quad \left[ \begin{array}{l} \text{No activation function} \\ \text{in output layer: } f^{\mathcal{L}}(z) = z \end{array} \right]$$

$$= W^{\mathcal{L}} a^{\mathcal{L}-1}(\mathbf{x}_i)$$

$$= W^{\mathcal{L}} f^{\mathcal{L}-1} \circ z^{\mathcal{L}-1}(\mathbf{x}_i)$$

$$= W^{\mathcal{L}} f^{\mathcal{L}-1} \circ W^{\mathcal{L}-1} a^{\mathcal{L}-2}(\mathbf{x}_i)$$

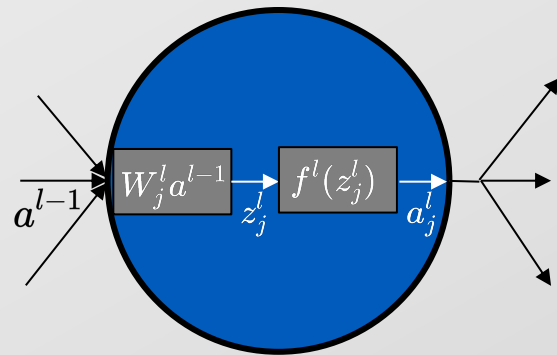
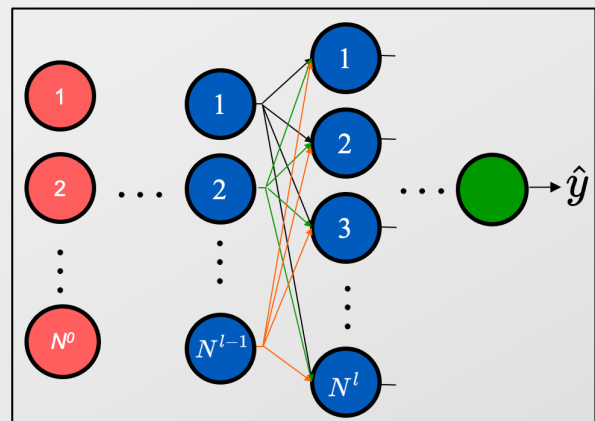
$$= W^{\mathcal{L}} f^{\mathcal{L}-1} \circ W^{\mathcal{L}-1} f^{\mathcal{L}-2} \circ \dots f^l \circ z^l(\mathbf{x}_i)$$

$$= W^{\mathcal{L}} f^{\mathcal{L}-1} \circ W^{\mathcal{L}-1} f^{\mathcal{L}-2} \circ \dots f^l \circ W^l a^{l-1}(\mathbf{x}_i)$$

$$= W^{\mathcal{L}} f^{\mathcal{L}-1} \circ W^{\mathcal{L}-1} f^{\mathcal{L}-2} \circ \dots f^l \circ W^l f^{l-1} \circ z^{l-1}(\mathbf{x}_i)$$

$$z^l = W^l a^{l-1}$$

$$\begin{aligned} a^l &= f^l(z^l) \\ &= f^l \circ z^l \end{aligned}$$



# The layer- $l$ $z$ derivative ( $\delta^l$ ) can be used to get the desired $W^l$ derivatives

$$\begin{pmatrix} \frac{\partial L}{\partial w_{11}^l} & \cdots & \frac{\partial L}{\partial w_{1(N^{l-1}+1)}^l} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial w_{N^l 1}^l} & \cdots & \frac{\partial L}{\partial w_{N^l(N^{l-1}+1)}^l} \end{pmatrix} \leftarrow \frac{\partial L_i}{\partial W^l} = \sum_{j=1}^{N^l} \boxed{\frac{\partial L_i}{\partial z_j^l}} \frac{\partial z_j^l}{\partial W^l} \equiv \sum_{j=1}^{N^l} \boxed{\delta_j^l} \frac{\partial z_j^l}{\partial W^l}$$

define  $\delta^l \equiv \begin{pmatrix} \delta_1^l \\ \vdots \\ \delta_{N^l}^l \end{pmatrix}$

$z^l = W^l a^{l-1}$

$\begin{pmatrix} z_1^l \\ z_2^l \\ \vdots \\ z_{N^l}^l \end{pmatrix} = \begin{pmatrix} w_{11}^l & w_{12}^l & \cdots & w_{1(N^{l-1}+1)}^l \\ w_{21}^l & w_{22}^l & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ w_{N^l 1}^l & \cdots & \cdots & w_{N^l(N^{l-1}+1)}^l \end{pmatrix} \begin{pmatrix} a_1^{l-1} \\ a_2^{l-1} \\ \vdots \\ a_{N^{l-1}}^{l-1} \\ 1 \end{pmatrix}$

Each  $z_j$  depends only on row  $j$  of  $W$

$$\sum_j \delta_j^l \frac{\partial z_j^l}{\partial W^l} = \delta_1^l \times \begin{pmatrix} a_1^{l-1} & a_2^{l-1} & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \xrightarrow{\text{same}} \begin{pmatrix} 0 & 0 & \cdots & 0 \\ a_1^{l-1} & a_2^{l-1} & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} + \cdots + \delta_{N^l}^l \times \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_1^{l-1} & a_2^{l-1} & \cdots & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \delta_1^l a_1^{l-1} & \delta_1^l a_2^{l-1} & \cdots & \delta_1^l \\ \delta_2^l a_1^{l-1} & \delta_2^l a_2^{l-1} & \cdots & \delta_2^l \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{N^l}^l a_1^{l-1} & \delta_{N^l}^l a_2^{l-1} & \cdots & \delta_{N^l}^l \end{pmatrix} = \begin{pmatrix} \delta_1^l \\ \delta_2^l \\ \vdots \\ \delta_{N^l}^l \end{pmatrix} (a_1^{l-1} \ a_2^{l-1} \ \cdots \ 1) = \delta^l (a^{l-1})^T \rightarrow \boxed{\frac{\partial L_i}{\partial W^l} = \delta^l (a^{l-1})^T}$$

# Recursion can be used to compute $\delta^l$

$$\hat{y}_i = W^{\mathcal{L}} f^{\mathcal{L}-1} \circ W^{\mathcal{L}-1} f^{\mathcal{L}-2} \circ \dots f^l \circ z^l(\mathbf{x}_i)$$

$$= W^{\mathcal{L}} f^{\mathcal{L}-1} \circ W^{\mathcal{L}-1} f^{\mathcal{L}-2} \circ \dots f^l \circ W^l f^{l-1} \circ z^{l-1}(\mathbf{x}_i)$$

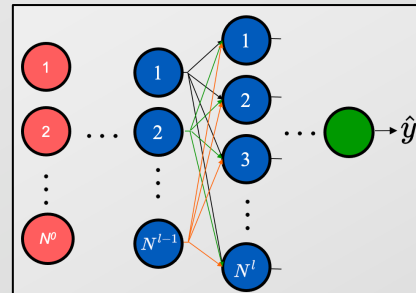
$$\delta^l \equiv \frac{\partial L_i}{\partial z^l} = (L'_i \circ \hat{y}_i) \frac{\partial \hat{y}_i}{\partial z^l}$$

$$(\delta^{l-1})^T = (L'_i \circ \hat{y}_i) \left( \frac{\partial \hat{y}_i}{\partial z^l} \right)^T \frac{\partial z^l}{\partial z^{l-1}} = (\delta^l)^T W^l \text{diag}[(f^{l-1})' \circ z^{l-1}]$$

$a_j^{l-1}$  depends only on  $z_j^{l-1}$

$z^l = W^l f^{l-1} \circ z^{l-1}$

This is the basis for *backpropagation*. Compute derivatives for output layer, and sweep back to get derivatives for each lower layer in succession ( $l \rightarrow l-1$ ), to the first one



# Feed-forward is just one of many types of neural networks that have been developed

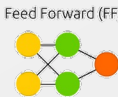
## A mostly complete chart of Neural Networks

©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org

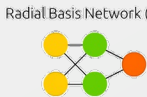
- Input Cell
- Backfed Input Cell
- Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- Spiking Hidden Cell
- Capsule Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Gated Memory Cell
- Kernel
- Convolution or Pool



Perceptron (P)

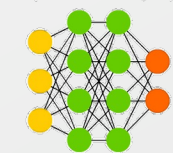


Feed Forward (FF)

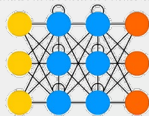


Radial Basis Network (RBF)

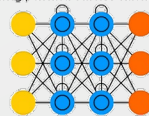
Deep Feed Forward (DFF)



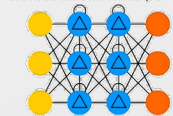
Recurrent Neural Network (RNN)



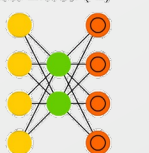
Long / Short Term Memory (LSTM)



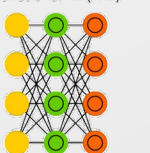
Gated Recurrent Unit (GRU)



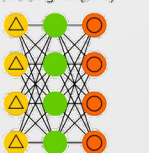
Auto Encoder (AE)



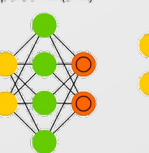
Variational AE (VAE)



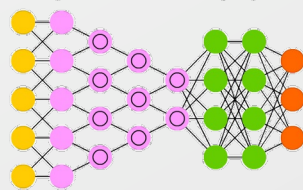
Denoising AE (DAE)



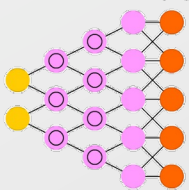
Sparse AE (SAE)



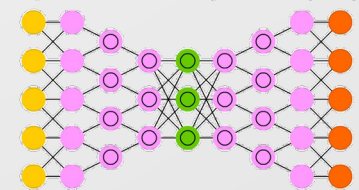
Deep Convolutional Network (DCN)



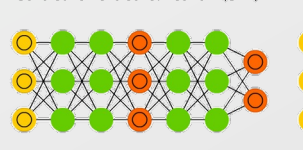
Deconvolutional Network (DN)



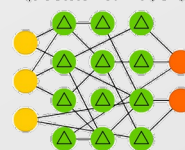
Deep Convolutional Inverse Graphics Network (DCIGN)



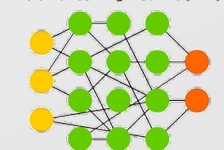
Generative Adversarial Network (GAN)



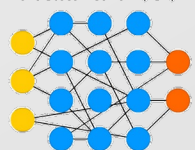
Liquid State Machine (LSM)



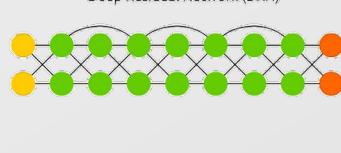
Extreme Learning Machine (ELM)



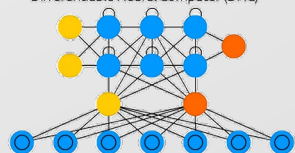
Echo State Network (ESN)



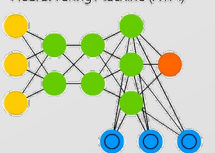
Deep Residual Network (DRN)



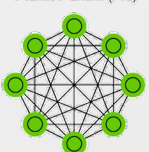
Differentiable Neural Computer (DNC)



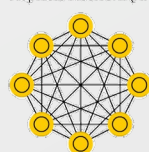
Neural Turing Machine (NTM)



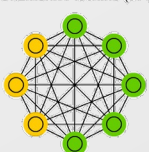
Markov Chain (MC)



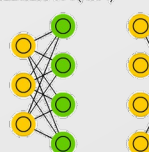
Hopfield Network (HN)



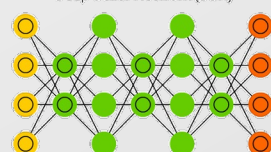
Boltzmann Machine (BM)



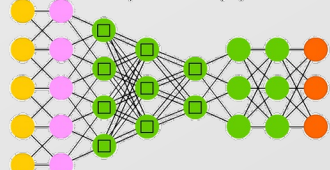
Restricted BM (RBM)



Deep Belief Network (DBN)



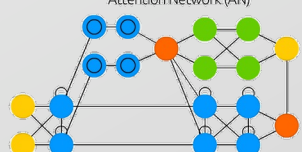
Capsule Network (CN)



Kohonen Network (KN)



Attention Network (AN)



<https://www.asimovinstitute.org/neural-network-zoo/>

# Suggested Reading/Viewing

- Dral, Pavlo O; Kananenka, Alexei A; Ge, Fuchun; Xue, Bao-Xin, Chapter 8, Neural Networks. In *Quantum Chemistry in the Age of Machine Learning*.
  - <https://doi.org/10.1016/B978-0-323-90049-2.00011-1>
  - Posted on UBLearn
- Jinzhe Zeng, Liqun Cao, and Tong Zhu, Chapter 12, Neural Network Potentials. In *Quantum Chemistry in the Age of Machine Learning*.
  - Posted on UBLearn
- <https://en.wikipedia.org/wiki/Backpropagation>
- Using TensorFlow to solve regression problems, including the MPG example
  - <https://www.youtube.com/watch?v=-vHQub0NXI4>