Faculty of Sciences of Tetouan

Introduction to Deep Learning

2- Shallow Versus Deep Neural Networks

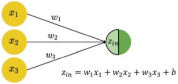
Prof. Monir EL ANNAS

Shallow Neural Network

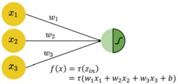
The Perceptron

- The basic computational unit for neural networks is the perceptron. It is a weighted sum of input values plus bias term, transformed by a non-linear activation function, resulting in an output value (single neuron).

❶ **Affine Transformation:** weighted sum of inputs plus bias.



❷ **Non-linear Activation:** a non-linear transformation applied to the weighted sum.



- A suitable choice of the activation function τ leads to known functions $f(x)$.
- The identity function gives us the simple linear regression:

$$y = \tau(\mathbf{w}^\top \mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$

- The logistic function (sigmoid function) gives us the logistic regression:

$$y = \tau(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$$

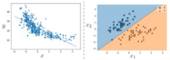
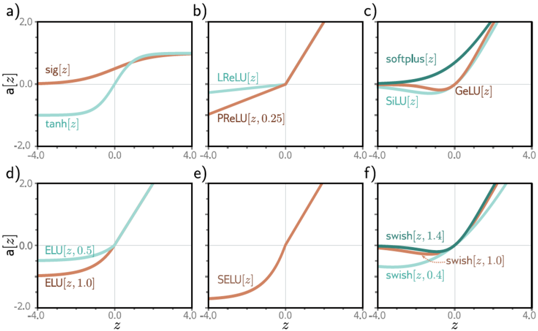


Figure: *Left*: A regression line learned by a single neuron. *Right*: A decision-boundary learned by a single neuron in a binary classification task.

Shallow Neural Network

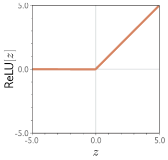
Activation Functions



$$a[z] = \mathrm{ReLU}[z] = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}.$$

Rectified Linear Unit

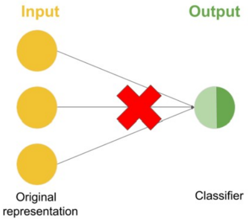
(particular kind of activation function)

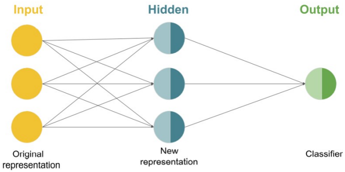


Shallow Neural Network

Shallow Neural Network

- Instead of a single neuron, we use more complex networks.
- Fully-connected neural networks describe the simplest neural network architecture where all neurons from one layer are connected to the neurons of the next layer.
- Fully-connected neural networks with a single layer are known as shallow neural networks.



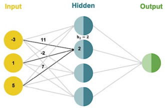


Shallow Neural Network

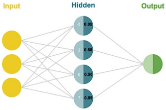
Forward Pass: Binary Classification Example

- Following the computation from left to right is called a **forward pass**. It is how Neural Networks make their predictions.

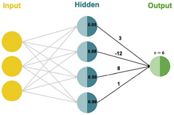
- Each neuron in the hidden layer performs an affine transformation on the inputs.
- Each hidden neuron performs a non-linear activation transformation on the weighted sum.
- The output neuron performs an affine transformation on its inputs.
- The output neuron performs a non-linear activation transformation on the weighted sum.



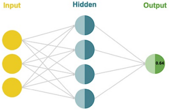
$$z_{\text{in}}^{(2)} = w_{12}x^{(1)} + w_{22}x^{(2)} + w_{32}x^{(3)} + b_2$$
$$z_{\text{in}}^{(2)} = 11 * (-3) + (-2) * 1 + 7 * 5 + 2 = 2$$



$$z_{\text{out}}^{(i)} = \sigma\left(z_{\text{in}}^{(i)}\right) = \frac{1}{1+e^{z_{\text{in}}^{(i)}}}$$



$$f_{\text{in}} = u_1 z_{\text{out}}^{(1)} + u_2 z_{\text{out}}^{(2)} + u_3 z_{\text{out}}^{(3)} + u_4 z_{\text{out}}^{(4)} + c$$



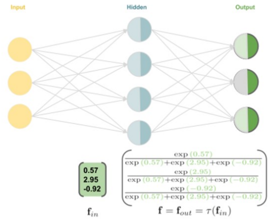
$$f_{\text{out}} = \sigma(f_{\text{in}}) = \frac{1}{1+e^{f_{\text{in}}}}$$
$$f_{\text{out}} = \frac{1}{1+e^{0.57}} = 0.64$$

- **Binary Classification**: one neuron in the output layer.
- **Activation function**: sigmoid activation function used in the output layer.

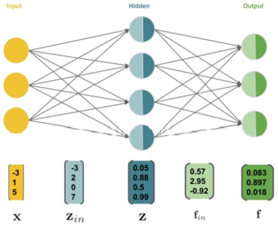
Shallow Neural Network

Forward Pass: Multi-Class Classification Example

Forward pass (Hidden: Sigmoid, Output: Softmax).



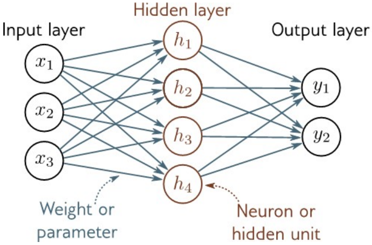
Forward pass (Hidden: Sigmoid, Output: Softmax).



- **Multi-Class Classification**: Add additional neurons to the output layer. Each neuron will represent a specific class.
- **Activation function**: softmax activation function used in the output layer $f_{\text{out},k} = \tau(f_{\text{in},k}) = \frac{\exp(f_{\text{in},k})}{\sum_{k'=1}^{G} \exp(f_{\text{in},k'})}$

Shallow Neural Network

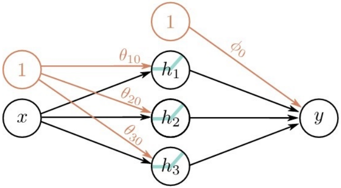
Formulation: Terminology



- Y-offsets = biases
- Slopes = weights
- Everything in one layer connected to everything in the next = fully connected network
- No loops = feedforward network
- Values after activation functions = activations
- Values before activation functions = pre-activations
- One hidden layer = shallow neural network
- More than one hidden layer = deep neural network
- Number of hidden units = capacity

Shallow Neural Network

Formulation: 1 input 1 output



- $y = f[\mathbf{x}, \phi] = \phi_0 + \phi_1 a[\theta_{10} + \theta_{11}\mathbf{x}] + \phi_2 a[\theta_{20} + \theta_{21}\mathbf{x}] + \phi_3 a[\theta_{30} + \theta_{31}\mathbf{x}]$
- Break down into two parts:

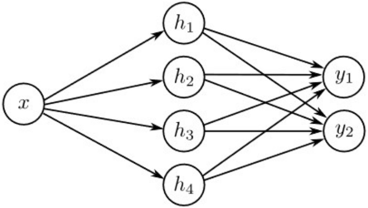
$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

- where we refer to h_1, h_2, and h_3 as hidden units :

$$\begin{aligned} h_1 &= \mathrm{a}\left[\theta_{10} + \theta_{11}x\right] \\ h_2 &= \mathrm{a}\left[\theta_{20} + \theta_{21}x\right] \\ h_3 &= \mathrm{a}\left[\theta_{30} + \theta_{31}x\right] \end{aligned}$$

Shallow Neural Network

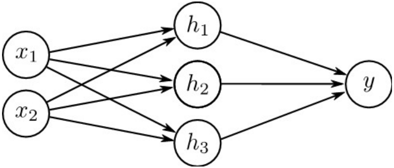
Formulation: More than one output



- Network with one input, four hidden units, and two outputs
- $$y_1 = \phi_{10} + \phi_{11} h_1 + \phi_{12} h_2 + \phi_{13} h_3 + \phi_{14} h_4$$
  $$y_2 = \phi_{20} + \phi_{21} h_1 + \phi_{22} h_2 + \phi_{23} h_3 + \phi_{24} h_4.$$
- where :
  $$\begin{aligned} h_1 &= \mathrm{a}\left[\theta_{10} + \theta_{11} x\right] \\ h_2 &= \mathrm{a}\left[\theta_{20} + \theta_{21} x\right] \\ h_3 &= \mathrm{a}\left[\theta_{30} + \theta_{31} x\right] \\ h_4 &= \mathrm{a}\left[\theta_{40} + \theta_{41} x\right], \end{aligned}$$

Shallow Neural Network

Formulation: More than one input



- Neural network with 2D multivariate input $\mathrm{x} = [x_1, x_2]^T$ and scalar output y.
- $y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$.
- where : $$\begin{aligned} h_1 &= \mathrm{a}[\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2] \\ h_2 &= \mathrm{a}[\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2] \\ h_3 &= \mathrm{a}[\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2], \end{aligned}$$

Shallow Neural Network

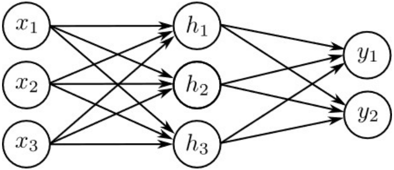
Formulation: General case arbitrary inputs hidden units outputs

- D_o Outputs, D hidden units, and D_i inputs

$$h_d = a\left(\theta_{d0} + \sum_{i=1}^{D_i} \theta_{di} x_i\right)$$

$$y_j = \phi_{j0} + \sum_{d=1}^{D} \phi_{jd} h_d$$

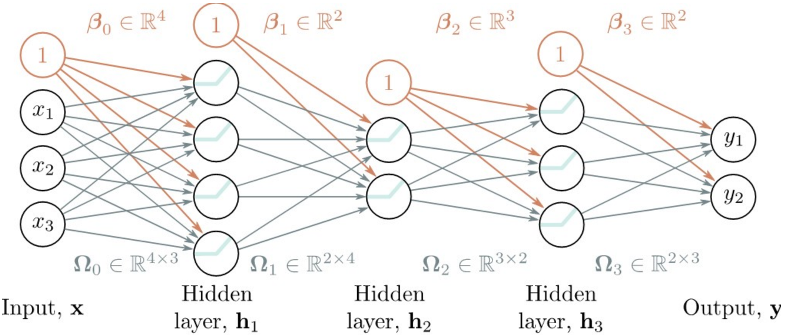
- e.g., Three inputs, three hidden units, two outputs



Deep neural networks

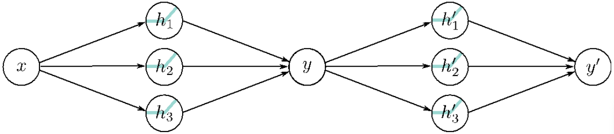
Deep neural networks

- Networks with more than one hidden layer
- Intuition becomes more difficult



Deep neural networks

Formulation: Deep network with two hidden layers



- Networks 1:

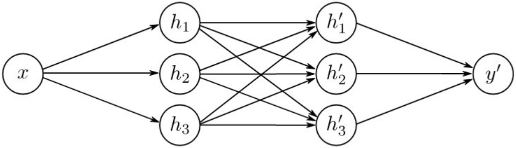
$$
\begin{aligned}
h_1 &= a(\theta_{10} + \theta_{11}x) \\
h_2 &= a(\theta_{20} + \theta_{21}x) \\
h_3 &= a(\theta_{30} + \theta_{31}x) \\
y &= \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3
\end{aligned}
$$

- Networks 2:

$$
\begin{aligned}
h'_1 &= a(\theta'_{10} + \theta'_{11}y) \\
h'_2 &= a(\theta'_{20} + \theta'_{21}y) \\
h'_3 &= a(\theta'_{30} + \theta'_{31}y) \\
y' &= \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3
\end{aligned}
$$

Deep neural networks

Formulation: Deep network with two hidden layers



- Substituting the expression for y gives:

$$h'_1 = \mathrm{a}\left[\theta'_{10} + \theta'_{11} y\right] = \mathrm{a}\left[\theta'_{10} + \theta'_{11}\phi_0 + \theta'_{11}\phi_1 h_1 + \theta'_{11}\phi_2 h_2 + \theta'_{11}\phi_3 h_3\right]$$

$$h'_2 = \mathrm{a}\left[\theta'_{20} + \theta'_{21} y\right] = \mathrm{a}\left[\theta'_{20} + \theta'_{21}\phi_0 + \theta'_{21}\phi_1 h_1 + \theta'_{21}\phi_2 h_2 + \theta'_{21}\phi_3 h_3\right]$$

$$h'_3 = \mathrm{a}\left[\theta'_{30} + \theta'_{31} y\right] = \mathrm{a}\left[\theta'_{30} + \theta'_{31}\phi_0 + \theta'_{31}\phi_1 h_1 + \theta'_{31}\phi_2 h_2 + \theta'_{31}\phi_3 h_3\right]$$

- Which we can rewrite as:

$$h'_1 = \mathrm{a}\left[\psi_{10} + \psi_{11} h_1 + \psi_{12} h_2 + \psi_{13} h_3\right]$$

$$h'_2 = \mathrm{a}\left[\psi_{20} + \psi_{21} h_1 + \psi_{22} h_2 + \psi_{23} h_3\right]$$

$$h'_3 = \mathrm{a}\left[\psi_{30} + \psi_{31} h_1 + \psi_{32} h_2 + \psi_{33} h_3\right]$$

- We can combine the equations to get one expression:

$$\begin{aligned} y' = \phi'_0 &+ \phi'_1 \mathrm{a}\left[\psi_{10} + \psi_{11}\mathrm{a}\left[\theta_{10} + \theta_{11}x\right] + \psi_{12}\mathrm{a}\left[\theta_{20} + \theta_{21}x\right] + \psi_{13}\mathrm{a}\left[\theta_{30} + \theta_{31}x\right]\right] \\ &+ \phi'_2 \mathrm{a}\left[\psi_{20} + \psi_{21}\mathrm{a}\left[\theta_{10} + \theta_{11}x\right] + \psi_{22}\mathrm{a}\left[\theta_{20} + \theta_{21}x\right] + \psi_{23}\mathrm{a}\left[\theta_{30} + \theta_{31}x\right]\right] \\ &+ \phi'_3 \mathrm{a}\left[\psi_{30} + \psi_{31}\mathrm{a}\left[\theta_{10} + \theta_{11}x\right] + \psi_{32}\mathrm{a}\left[\theta_{20} + \theta_{21}x\right] + \psi_{33}\mathrm{a}\left[\theta_{30} + \theta_{31}x\right]\right] \end{aligned}$$

Deep neural networks

Formulation: Hyperparameters

- K layers $=$ depth of network
- D_k hidden units per layer $=$ width of network
- These are called hyperparameters – chosen before training the network.
- Can try retraining with different hyperparameters – hyperparameter optimization or hyperparameter search.

Deep neural networks

Formulation: The compact form

- **Notation:**

$h_1 = a(\theta_{10} + \theta_{11} x)$

$h_2 = a(\theta_{20} + \theta_{21} x)$

$h_3 = a(\theta_{30} + \theta_{31} x)$

$h'_1 = a(\psi_{10} + \psi_{11} h_1 + \psi_{12} h_2 + \psi_{13} h_3)$

$h'_2 = a(\psi_{20} + \psi_{21} h_1 + \psi_{22} h_2 + \psi_{23} h_3)$

$h'_3 = a(\psi_{30} + \psi_{31} h_1 + \psi_{32} h_2 + \psi_{33} h_3)$

$y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$

- **The compact form:**

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \mathbf{a} \left[\begin{bmatrix} \theta_{10} \\ \theta_{20} \\ \theta_{30} \end{bmatrix} + \begin{bmatrix} \theta_{11} \\ \theta_{21} \\ \theta_{31} \end{bmatrix} x \right]$$

$$\begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix} = a \left(\begin{bmatrix} \psi_{10} \\ \psi_{20} \\ \psi_{30} \end{bmatrix} + \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \\ \psi_{31} & \psi_{32} & \psi_{33} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} \right)$$

$$y' = \phi'_0 + \begin{bmatrix} \phi'_1 & \phi'_2 & \phi'_3 \end{bmatrix} \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix}$$

Deep neural networks

Formulation: The compact form

- **Notation:**

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \theta_{10} \\ \theta_{20} \\ \theta_{30} \end{bmatrix} + \begin{bmatrix} \theta_{11} \\ \theta_{21} \\ \theta_{31} \end{bmatrix} x\right]$$

$$\begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix} = a\left(\begin{bmatrix} \psi_{10} \\ \psi_{20} \\ \psi_{30} \end{bmatrix} + \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \\ \psi_{31} & \psi_{32} & \psi_{33} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}\right)$$

$$y' = \phi'_0 + \begin{bmatrix} \phi'_1 & \phi'_2 & \phi'_3 \end{bmatrix} \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix}$$

- **The compact form:**

$$h = a(\boldsymbol{\theta}_0 + \boldsymbol{\theta}\mathbf{x})$$

$$h' = a(\boldsymbol{\psi}_0 + \boldsymbol{\Psi}\mathbf{h})$$

$$y = \phi'_0 + \boldsymbol{\phi}'\mathbf{h}'$$

Deep neural networks

Formulation: The compact form

- **Notation:**

$$h = a(\boldsymbol{\theta}_0 + \boldsymbol{\theta}\mathbf{x})$$

$$h' = a(\psi_0 + \boldsymbol{\Psi}\mathbf{h})$$

$$y = \phi'_0 + \boldsymbol{\phi}'\mathbf{h}'$$

- **The compact form:**

$$h_1 = a(\beta_0 + \Omega_0 x)$$

$$h_2 = a(\beta_1 + \Omega_1 h_1)$$

$$y = \beta_2 + \Omega_2 h_2$$

- A general deep network $\mathbf{y} = \mathrm{f}[\mathrm{x}, \phi]$ with K layers can now be written as:

$$\begin{aligned}
\mathbf{h}_1 &= \mathbf{a}\left[\boldsymbol{\beta}_0 + \Omega_0\mathbf{x}\right] \\
\mathbf{h}_2 &= \mathbf{a}\left[\boldsymbol{\beta}_1 + \Omega_1\mathbf{h}_1\right] \\
\mathbf{h}_3 &= \mathbf{a}\left[\boldsymbol{\beta}_2 + \Omega_2\mathbf{h}_2\right] \\
&\vdots \\
\mathbf{h}_K &= \mathbf{a}\left[\boldsymbol{\beta}_{K-1} + \Omega_{K-1}\mathbf{h}_{K-1}\right] \\
\mathbf{y} &= \boldsymbol{\beta}_K + \Omega_K\mathbf{h}_K.
\end{aligned}$$

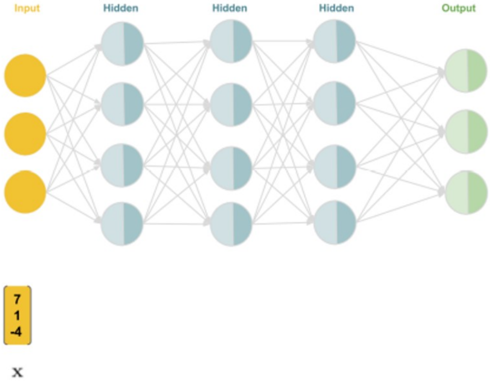
- We can equivalently write the network as a single function:

$$\mathbf{y} = \boldsymbol{\beta}_K + \Omega_K\mathbf{a}\left[\boldsymbol{\beta}_{K-1} + \Omega_{K-1}\mathbf{a}\left[\ldots\boldsymbol{\beta}_2 + \Omega_2\mathbf{a}\left[\boldsymbol{\beta}_1 + \Omega_1\mathbf{a}\left[\boldsymbol{\beta}_0 + \Omega_0\mathbf{x}\right]\right]\ldots\right]\right].$$

Deep neural networks

Forward Pass: Multi-Class Classification Example

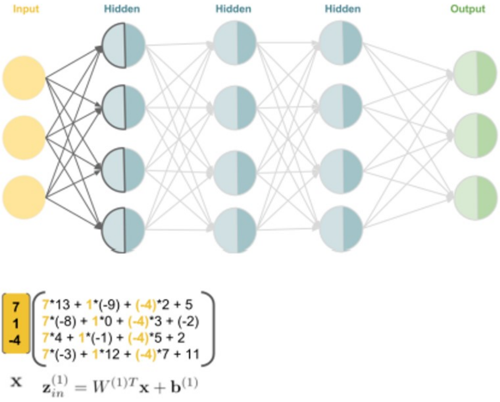
- Forward pass (Hidden: ReLU, Output: Softmax)



Deep neural networks

Forward Pass: Multi-Class Classification Example

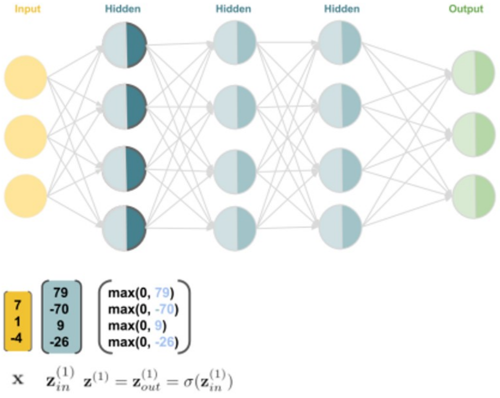
- Forward pass (Hidden: ReLU, Output: Softmax)



Deep neural networks

Forward Pass: Multi-Class Classification Example

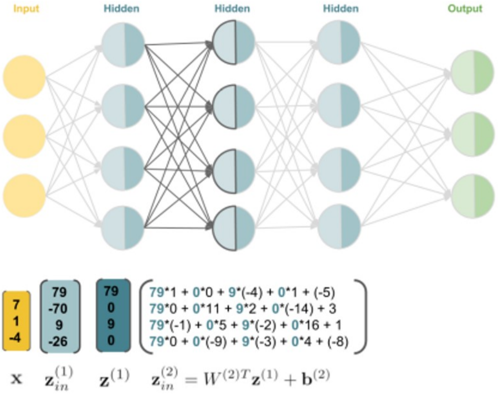
- Forward pass (Hidden: ReLU, Output: Softmax)



Deep neural networks

Forward Pass: Multi-Class Classification Example

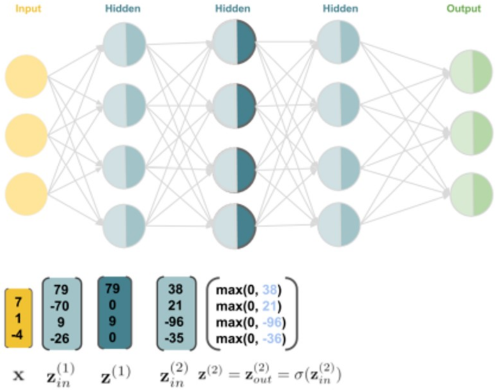
- Forward pass (Hidden: ReLU, Output: Softmax)



Deep neural networks

Forward Pass: Multi-Class Classification Example

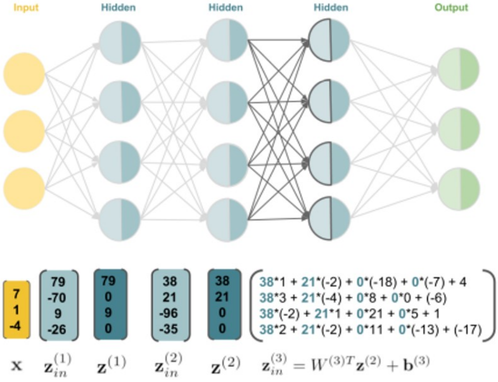
- Forward pass (Hidden: ReLU, Output: Softmax)



Deep neural networks

Forward Pass: Multi-Class Classification Example

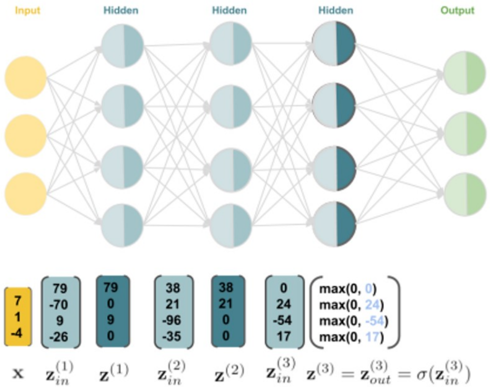
- Forward pass (Hidden: ReLU, Output: Softmax)



Deep neural networks

Forward Pass: Multi-Class Classification Example

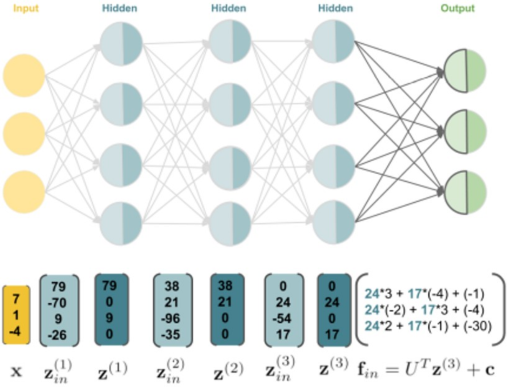
- Forward pass (Hidden: ReLU, Output: Softmax)



Deep neural networks

Forward Pass: Multi-Class Classification Example

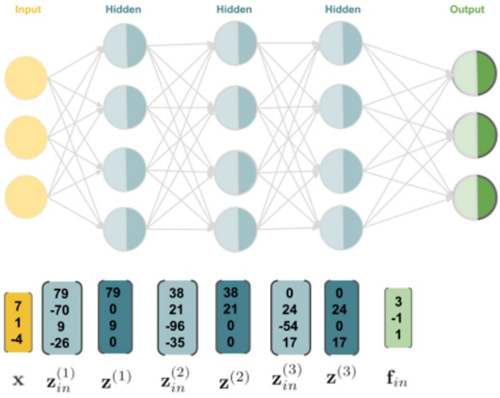
- Forward pass (Hidden: ReLU, Output: Softmax)



Deep neural networks

Forward Pass: Multi-Class Classification Example

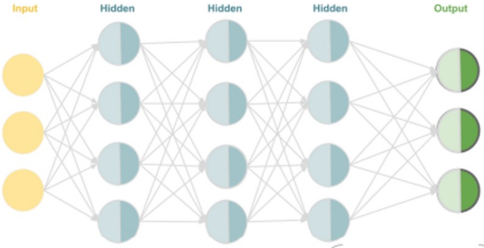
- Forward pass (Hidden: ReLU, Output: Softmax)



Deep neural networks

Forward Pass: Multi-Class Classification Example

- Forward pass (Hidden: ReLU, Output: Softmax)



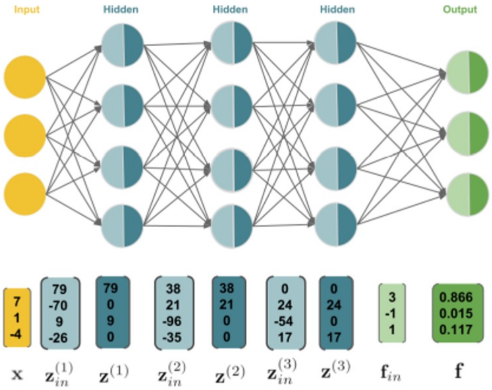
$$\mathbf{f}_{in} = \begin{bmatrix} 3 \\ -1 \\ 1 \end{bmatrix}$$

$$\mathbf{f} = \mathbf{f}_{out} = \tau(\mathbf{f}_{in}) = \begin{pmatrix} \frac{\exp(3)}{\exp 3 + \exp(-1) + \exp(1)} \\ \frac{\exp(-1)}{\exp 3 + \exp(-1) + \exp(1)} \\ \frac{\exp(1)}{\exp 3 + \exp(-1) + \exp(1)} \end{pmatrix}$$

Deep neural networks

Forward Pass: Multi-Class Classification Example

- Forward pass (Hidden: ReLU, Output: Softmax)



Where are we going?

- We have defined families of very flexible networks that map multiple inputs to multiple outputs
- Now we need to train them:
  - How to choose loss functions
  - How to find minima of the loss function
  - How to do this in particular for deep networks
- Then we need to test them