

Introduction to Deep Learning

Chapter 2: MLP – Matrix Notation

Bernd Bischl

Department of Statistics – LMU Munich

WS 2021/2022



LECTURE OUTLINE

This chapter introduces a compact way for representing feedforward neural networks: the matrix formalism from linear algebra.

- ➊ First, we explore networks with one hidden layer and one output unit.
- ➋ Next, we investigate networks with one hidden layer but multiple output units.
- ➌ Finally, we focus on multi-layer feedforward networks with an arbitrary number of hidden layers and output units.

Single Hidden Layer Networks for Regression and Binary Classification

SINGLE HIDDEN LAYER NETWORKS

- The input \mathbf{x} is a column vector with dimensions $p \times 1$.
- \mathbf{W} is a weight matrix with dimensions $p \times m$:

$$\mathbf{W} = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{p,1} & w_{p,2} & \cdots & w_{p,m} \end{pmatrix}$$

- For example, to obtain z_1 , we pick the first column of \mathbf{W} :

$$\mathbf{w}_1 = \begin{pmatrix} w_{1,1} \\ w_{2,1} \\ \vdots \\ w_{p,1} \end{pmatrix}$$

and compute $z_1 = \sigma(\mathbf{w}_1^\top \mathbf{x} + b_1)$, where b_1 is the bias of the first hidden neuron and $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is an activation function.

SINGLE HIDDEN LAYER NETWORKS: NOTATION

General notation:

- The network has m hidden neurons z_1, \dots, z_m with

$$z_j = \sigma(\mathbf{W}_j^\top \mathbf{x} + b_j)$$

- $z_{in,j} = \mathbf{W}_j^\top \mathbf{x} + b_j$
- $z_{out,j} = \sigma(z_{in,j}) = \sigma(\mathbf{W}_j^\top \mathbf{x} + b_j)$

for $j \in \{1, \dots, m\}$.

SINGLE HIDDEN LAYER NETWORKS: NOTATION

- Vectorized notation:

- $\mathbf{z}_{in} = (z_{in,1}, \dots, z_{in,m})^\top = \mathbf{W}^\top \mathbf{x} + \mathbf{b}$
(Note: $\mathbf{W}^\top \mathbf{x} = (\mathbf{x}^\top \mathbf{W})^\top$)

- $\mathbf{z} = \mathbf{z}_{out} = \sigma(\mathbf{z}_{in}) = \sigma(\mathbf{W}^\top \mathbf{x} + \mathbf{b})$, where the (hidden layer) activation function σ is applied element-wise to \mathbf{z}_{in} .

- Bias term:

- We sometimes omit the bias term by adding a constant feature to the input $\tilde{\mathbf{x}} = (1, x_1, \dots, x_p)$ and by adding the bias term to the weight matrix

$$\tilde{\mathbf{W}} = (\mathbf{b}, \mathbf{W}_1, \dots, \mathbf{W}_p).$$

- **Note:** For simplification purposes, we will not explicitly represent the bias term graphically in the following. However, the above “trick” makes it straightforward to represent it graphically.

SINGLE HIDDEN LAYER NETWORKS: NOTATION

General notation:

- For regression or binary classification: one output unit f where
 - $f_{in} = \mathbf{u}^\top \mathbf{z} + c$, i.e. a linear combination of derived features plus the bias term c of the output neuron, and
 - $f(\mathbf{x}) = f_{out} = \tau(f_{in}) = \tau(\mathbf{u}^\top \mathbf{z} + c)$, where τ is the output activation function.
- For regression τ is the identity function.
- For binary classification, τ is a sigmoid function.
- **Note:** The purpose of the hidden-layer activation function σ is to introduce non-linearities so that the network is able to learn complex functions whereas the purpose of τ is merely to get the final score on the same scale as the target.

SINGLE HIDDEN LAYER NETWORKS: NOTATION

General notation: Multiple inputs

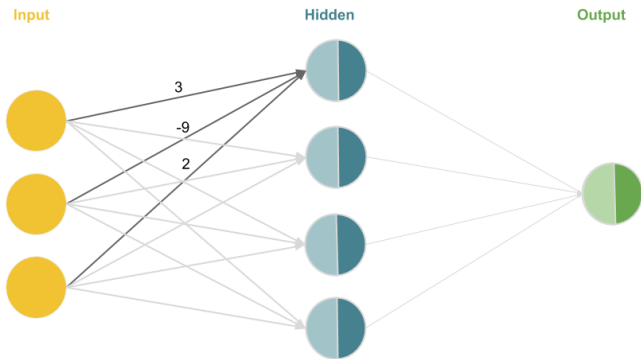
- It is possible to feed multiple inputs to a neural network simultaneously.
- The inputs $\mathbf{x}^{(i)}$, for $i \in \{1, \dots, n\}$, are arranged as rows in the **design matrix \mathbf{X}** .
 - \mathbf{X} is a $(n \times p)$ -matrix.
- The weighted sum in the hidden layer is now computed as $\mathbf{XW} + \mathbf{B}$, where,
 - \mathbf{W} , as usual, is a $(p \times m)$ matrix, and,
 - \mathbf{B} is a $(n \times m)$ matrix containing the bias vector \mathbf{b} (duplicated) as the rows of the matrix.
- The *matrix* of hidden activations $\mathbf{Z} = \sigma(\mathbf{XW} + \mathbf{B})$
 - \mathbf{Z} is a $(n \times m)$ matrix.

SINGLE HIDDEN LAYER NETWORKS: NOTATION

- The final output of the network, which contains a prediction for each input, is $\tau(\mathbf{Z}\mathbf{u} + \mathbf{C})$, where
 - \mathbf{u} is the vector of weights of the output neuron, and,
 - \mathbf{C} is a $(n \times 1)$ matrix whose elements are the (scalar) bias c of the output neuron.

SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

- Weights (and biases) of the network.



$$\begin{pmatrix} 3 & -9 & 2 \end{pmatrix} \begin{pmatrix} 5 \end{pmatrix}$$

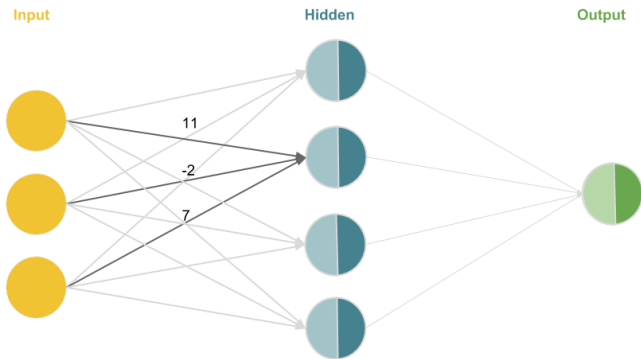
$W^T \quad \mathbf{b}$

$$\begin{pmatrix} \quad \end{pmatrix} \begin{pmatrix} \quad \end{pmatrix}$$

$\mathbf{u}^T \quad c$

SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

- Weights (and biases) of the network.



$$\begin{pmatrix} 3 & -9 & 2 \\ 11 & -2 & 7 \end{pmatrix} \begin{pmatrix} 5 \\ 2 \end{pmatrix}$$

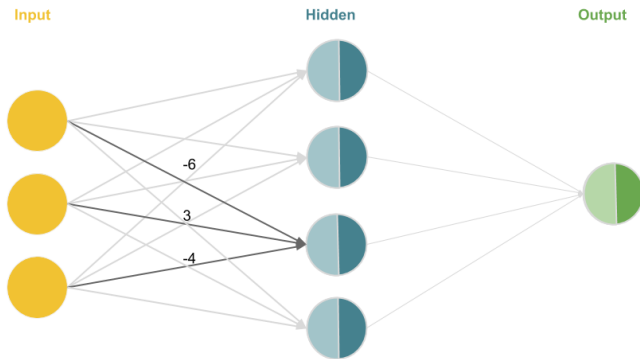
$$W^T \quad \mathbf{b}$$

$$\begin{pmatrix} & & \end{pmatrix} \begin{pmatrix} \end{pmatrix}$$

$$\mathbf{u}^T \quad c$$

SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

- Weights (and biases) of the network.



$$\begin{pmatrix} 3 & -9 & 2 \\ 11 & -2 & 7 \\ -6 & 3 & -4 \end{pmatrix} \begin{pmatrix} 5 \\ 2 \\ -1 \end{pmatrix}$$

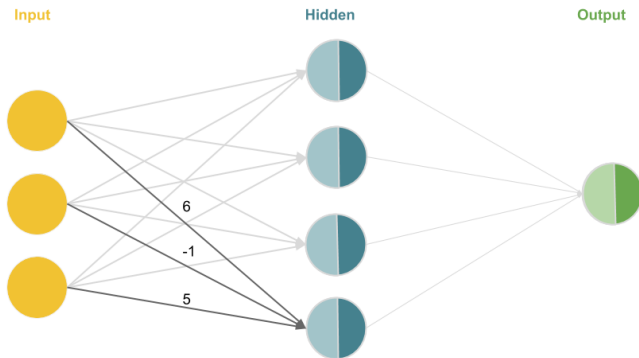
$W^T \quad \mathbf{b}$

$$\begin{pmatrix} \quad \end{pmatrix} \begin{pmatrix} \quad \end{pmatrix}$$

$\mathbf{u}^T \quad c$

SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

- Weights (and biases) of the network.



$$\begin{pmatrix} 3 & -9 & 2 \\ 11 & -2 & 7 \\ -6 & 3 & -4 \\ 6 & -1 & 5 \end{pmatrix} \begin{pmatrix} 5 \\ 2 \\ -1 \\ 1 \end{pmatrix}$$

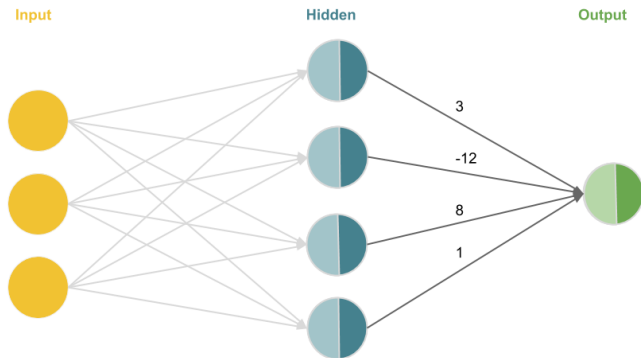
$$W^T \quad b$$

$$\begin{pmatrix} \quad \end{pmatrix} \begin{pmatrix} \quad \end{pmatrix}$$

$$u^T \quad c$$

SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

- Weights (and biases) of the network.



$$\begin{pmatrix} 3 & -9 & 2 \\ 11 & -2 & 7 \\ -6 & 3 & -4 \\ 6 & -1 & 5 \end{pmatrix} \begin{pmatrix} 5 \\ 2 \\ -1 \\ 1 \end{pmatrix}$$

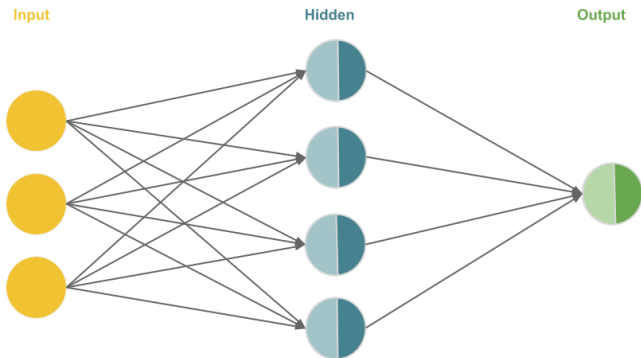
$$\mathbf{W}^T \quad \mathbf{b}$$

$$\begin{bmatrix} 3 & -12 & 8 & 1 \end{bmatrix} \begin{bmatrix} 6 \end{bmatrix}$$

$$\mathbf{u}^T \quad c$$

SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

- Weights (and biases) of the network.



$$\begin{pmatrix} 3 & -9 & 2 \\ 11 & -2 & 7 \\ -6 & 3 & -4 \\ 6 & -1 & 5 \end{pmatrix} \begin{pmatrix} 5 \\ 2 \\ -1 \\ 1 \end{pmatrix}$$

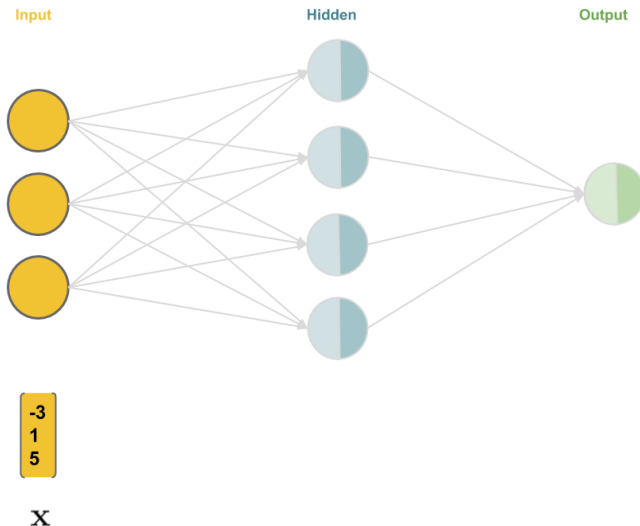
$$W^T \quad \mathbf{b}$$

$$\begin{pmatrix} 3 & -12 & 8 & 1 \end{pmatrix} \begin{pmatrix} 6 \end{pmatrix}$$

$$\mathbf{u}^T \quad c$$

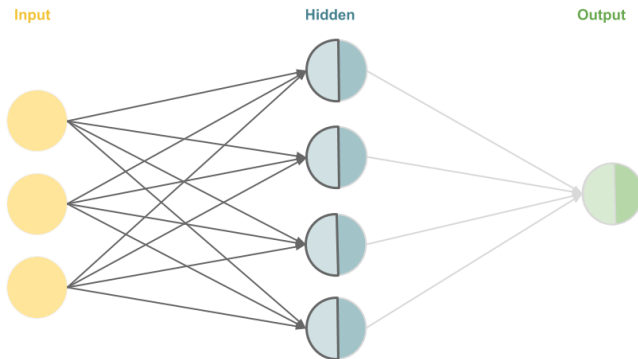
SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

Forward pass through the shallow neural network.



SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

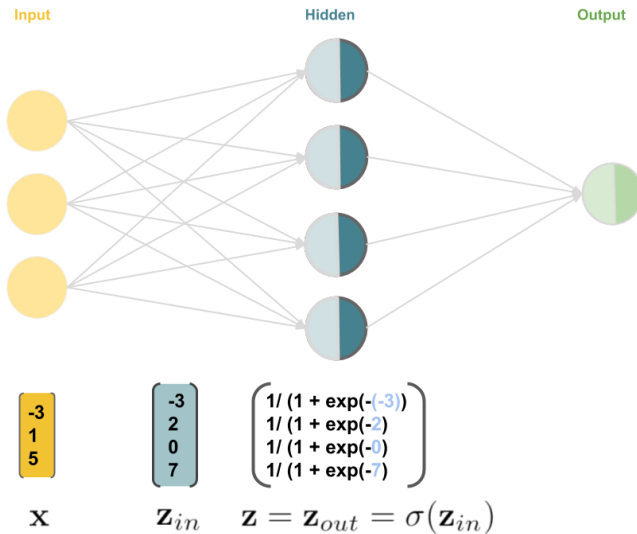
Forward pass through the shallow neural network.



$$\mathbf{x} = \begin{bmatrix} -3 \\ 1 \\ 5 \end{bmatrix} \quad \mathbf{z}_{in} = \mathbf{W}^T \mathbf{x} + \mathbf{b}$$
$$\begin{pmatrix} (-3)*3 + 1*(-9) + 5*2 + 5 \\ (-3)*11 + 1*(-2) + 5*7 + 2 \\ (-3)*(-6) + 1*3 + 5*(-4) + (-1) \\ (-3)*6 + 1*(-1) + 5*5 + 1 \end{pmatrix}$$

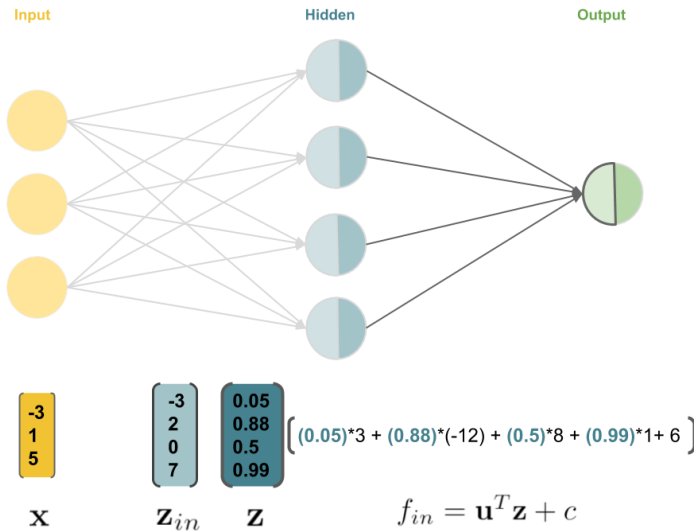
SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

Forward pass through the shallow neural network.



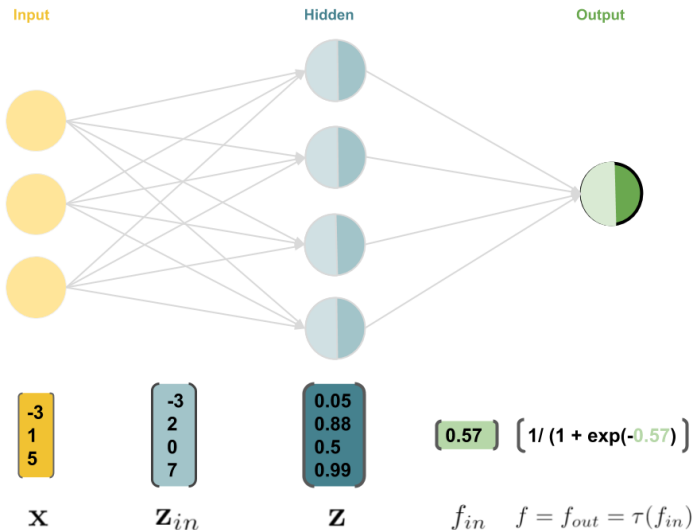
SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

Forward pass through the shallow neural network.



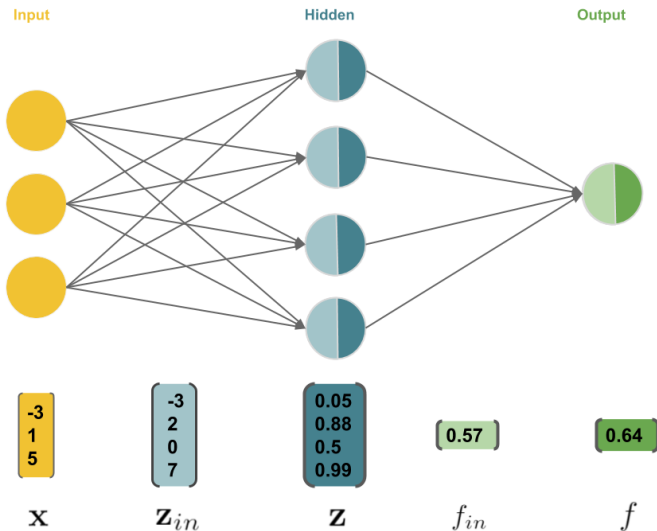
SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

Forward pass through the shallow neural network.



SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

Forward pass through the shallow neural network.



HIDDEN LAYER: ACTIVATION FUNCTION

- It is important to note that if the hidden layer does not have a non-linear activation, the network can only learn linear decision boundaries.
- For simplification purposes, we drop the bias terms in notation and let $\sigma = \text{id}$. Then:

$$\begin{aligned}f(\mathbf{x}) &= \tau(\mathbf{u}^\top \mathbf{z}) = \tau(\mathbf{u}^\top \sigma(\mathbf{W}^\top \mathbf{x})) \\&= \tau(\mathbf{u}^\top \sigma(\mathbf{W}^\top \mathbf{x})) \\&= \tau(\mathbf{u}^\top \mathbf{W}^\top \mathbf{x}) = \tau(\mathbf{v}^\top \mathbf{x})\end{aligned}$$

where $\mathbf{v} = \mathbf{W}\mathbf{u}$. It can be seen that $f(\mathbf{x})$ can only yield a linear decision boundary.

Single Hidden Layer Networks for Multi-Class Classification

MULTI-CLASS CLASSIFICATION

- We have only considered regression and binary classification problems so far.
- How can we get a neural network to perform multiclass classification?

MULTI-CLASS CLASSIFICATION

- The first step is to add additional neurons to the output layer.
- Each neuron in the layer will represent a specific class (number of neurons in the output layer = number of classes).

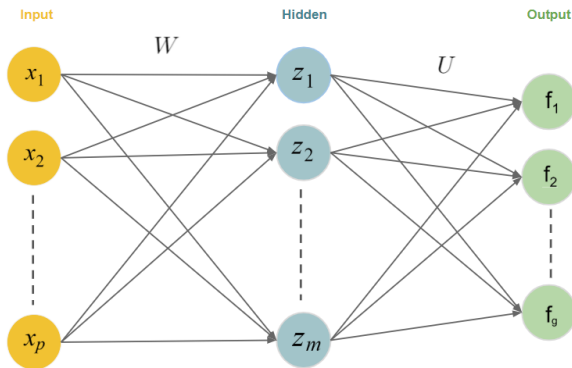


Figure: Structure of a single hidden layer, feed-forward neural network for g -class classification problems (bias term omitted).

MULTI-CLASS CLASSIFICATION

Notation:

- For g -class classification, g output units:

$$\mathbf{f} = (f_1, \dots, f_g)$$

- m hidden neurons z_1, \dots, z_m , with

$$z_j = \sigma(\mathbf{W}_j^\top \mathbf{x}), \quad j = 1, \dots, m.$$

- Compute linear combinations of derived features z :

$$f_{in,k} = \mathbf{U}_k^\top \mathbf{z}, \quad \mathbf{z} = (z_1, \dots, z_m)^\top, \quad k = 1, \dots, g$$

MULTI-CLASS CLASSIFICATION

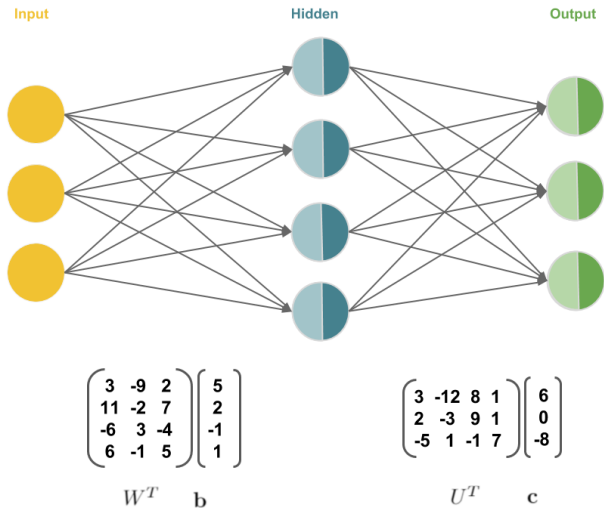
- The second step is to apply a softmax activation function to the output layer.
- This gives us a probability distribution over g different possible classes:

$$f_{out,k} = \tau_k(f_{in,k}) = \frac{\exp(f_{in,k})}{\sum_{k'=1}^g \exp(f_{in,k'})}$$

- This is the same transformation used in softmax regression!
- Derivative $\frac{\delta \tau(\mathbf{f}_{in})}{\delta \mathbf{f}_{in}} = \text{diag}(\tau(\mathbf{f}_{in})) - \tau(\mathbf{f}_{in})\tau(\mathbf{f}_{in})^\top$
- It is a “smooth” approximation of the argmax operation, so $\tau((1, 1000, 2)^\top) \approx (0, 1, 0)^\top$ (picks out 2nd element!).

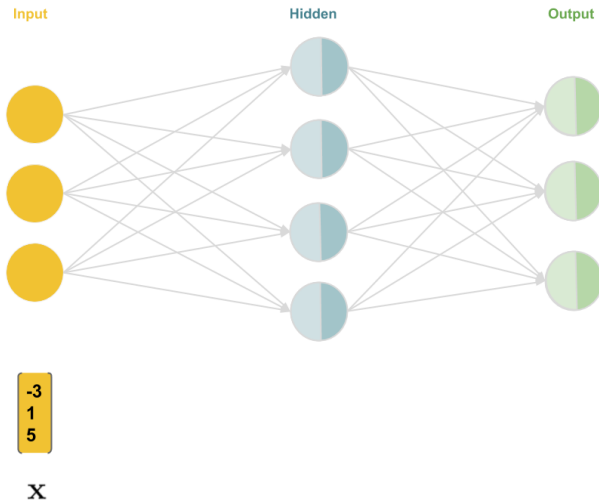
MULTI-CLASS CLASSIFICATION: EXAMPLE

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



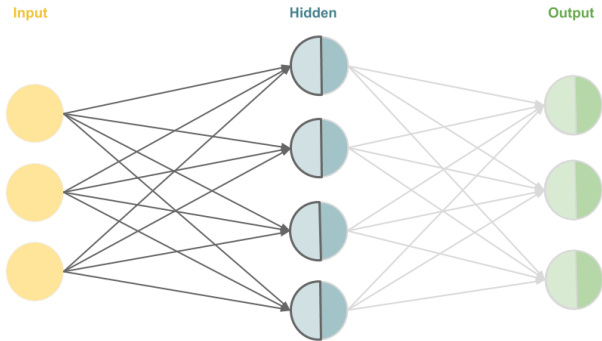
MULTI-CLASS CLASSIFICATION: EXAMPLE

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



MULTI-CLASS CLASSIFICATION: EXAMPLE

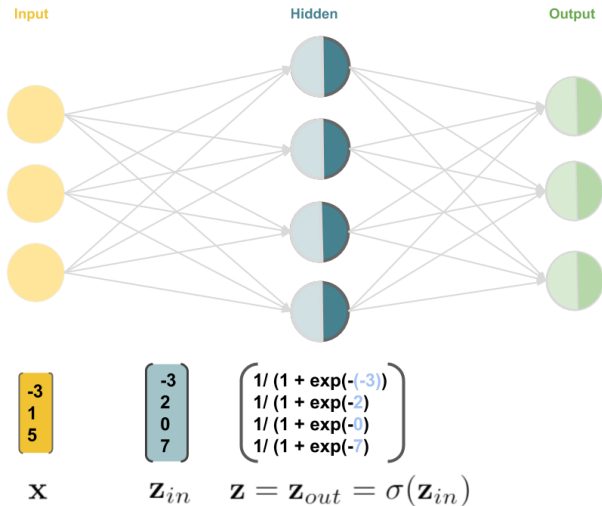
Forward pass (Hidden: Sigmoid, Output: Softmax).



$$\begin{bmatrix} -3 \\ 1 \\ 5 \end{bmatrix} \quad \begin{pmatrix} (-3)*3 + 1*(-9) + 5*2 + 5 \\ (-3)*11 + 1*(-2) + 5*7 + 2 \\ (-3)*(-6) + 1*3 + 5*(-4) + (-1) \\ (-3)*6 + 1*(-1) + 5*5 + 1 \end{pmatrix}$$
$$\mathbf{x} \quad \mathbf{z}_{in} = \mathbf{w}^T \mathbf{x} + \mathbf{b}$$

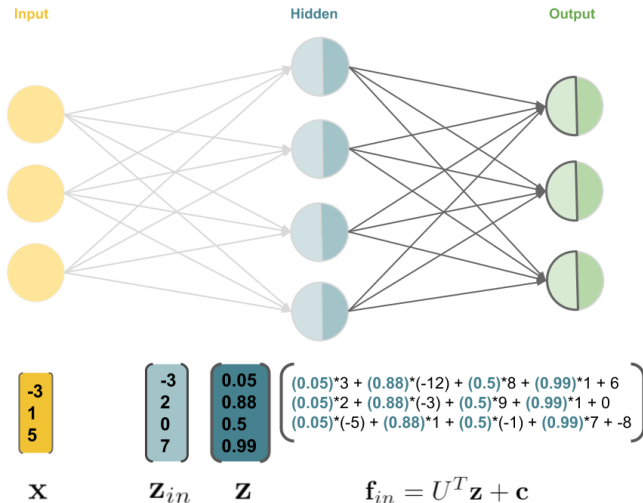
MULTI-CLASS CLASSIFICATION: EXAMPLE

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



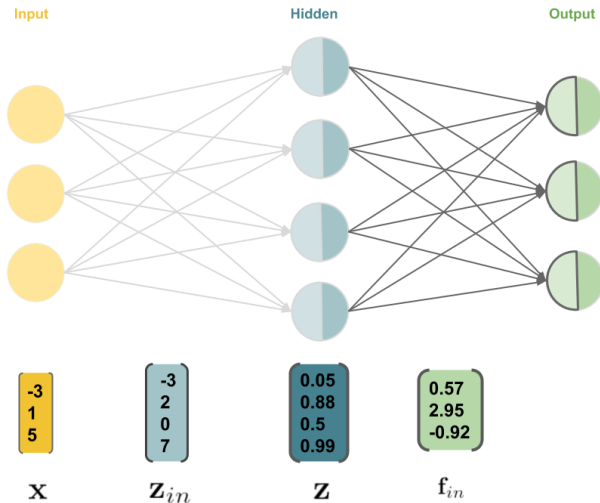
MULTI-CLASS CLASSIFICATION: EXAMPLE

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



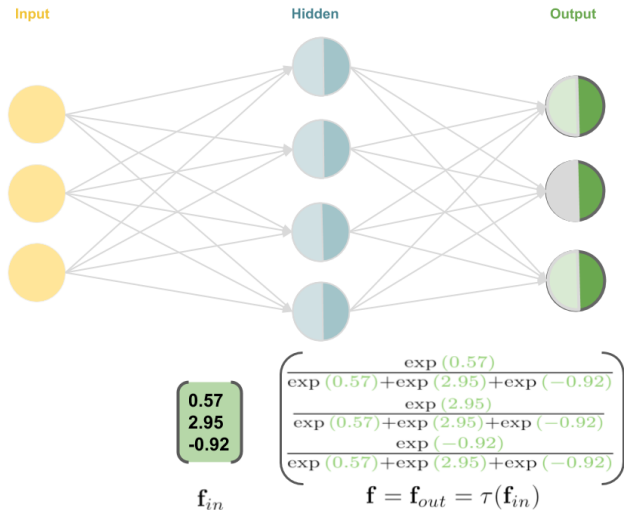
MULTI-CLASS CLASSIFICATION: EXAMPLE

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



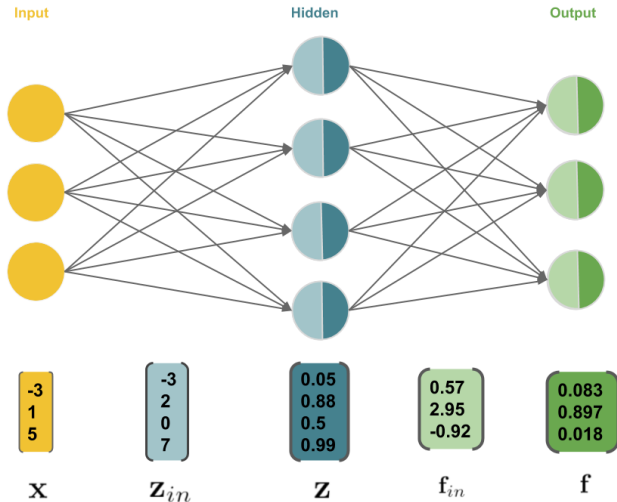
MULTI-CLASS CLASSIFICATION: EXAMPLE

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



MULTI-CLASS CLASSIFICATION: EXAMPLE

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



SOFTMAX LOSS

- The loss function for a softmax classifier is

$$L(y, f(\mathbf{x})) = - \sum_{k=1}^g [y = k] \log \left(\frac{\exp(f_{in,k})}{\sum_{k'=1}^g \exp(f_{in,k'})} \right)$$

$$\text{where } [y = k] = \begin{cases} 1 & \text{if } y = k \\ 0 & \text{otherwise} \end{cases}.$$

- This is equivalent to the cross-entropy loss when the label vector \mathbf{y} is one-hot coded (e.g. $\mathbf{y} = (0, 0, 1, 0)^\top$).
- Optimization: Again, there is no analytic solution.

Multi-Layer Feedforward Neural Networks

FEEDFORWARD NEURAL NETWORKS

- We will now extend the model class once again, such that we allow an arbitrary amount of / (hidden) layers.
- The general term for this model class is (multi-layer) **feedforward networks** (inputs are passed through the network from left to right, no feedback-loops are allowed)

FEEDFORWARD NEURAL NETWORKS

- We can characterize those models by the following chain structure:

$$f(\mathbf{x}) = \tau \circ \phi \circ \sigma^{(l)} \circ \phi^{(l)} \circ \sigma^{(l-1)} \circ \phi^{(l-1)} \circ \dots \circ \sigma^{(1)} \circ \phi^{(1)}$$

where $\sigma^{(i)}$ and $\phi^{(i)}$ are the activation function and the weighted sum of hidden layer i , respectively. τ and ϕ are the corresponding components of the output layer.

- Each hidden layer has:
 - an associated weight matrix $\mathbf{W}^{(i)}$, bias $\mathbf{b}^{(i)}$ and activations $\mathbf{z}^{(i)}$ for $i \in \{1 \dots l\}$
 - $\mathbf{z}^{(i)} = \sigma^{(i)}(\phi^{(i)}) = \sigma^{(i)}(\mathbf{W}^{(i)T} \mathbf{z}^{(i-1)} + \mathbf{b}^{(i)})$, where $\mathbf{z}^{(0)} = \mathbf{x}$.
- Again, without non-linear activations in the hidden layers, the network can only learn linear decision boundaries.

FEEDFORWARD NEURAL NETWORKS

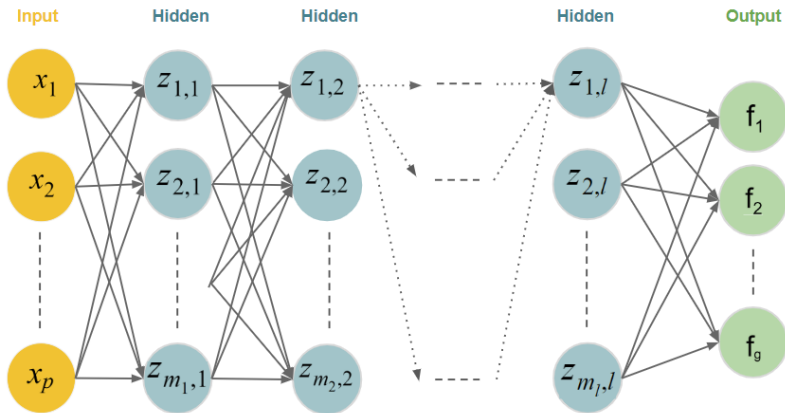
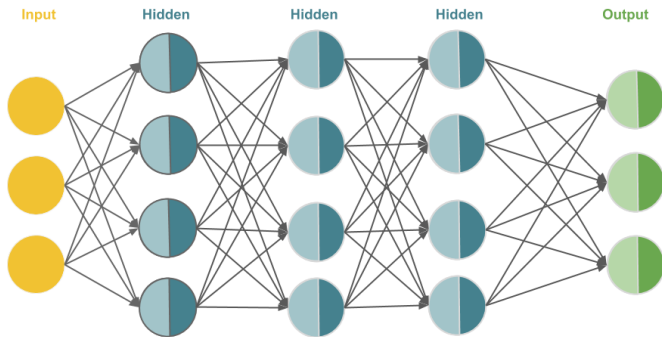


Figure: Structure of a deep neural network with l hidden layers (bias terms omitted).

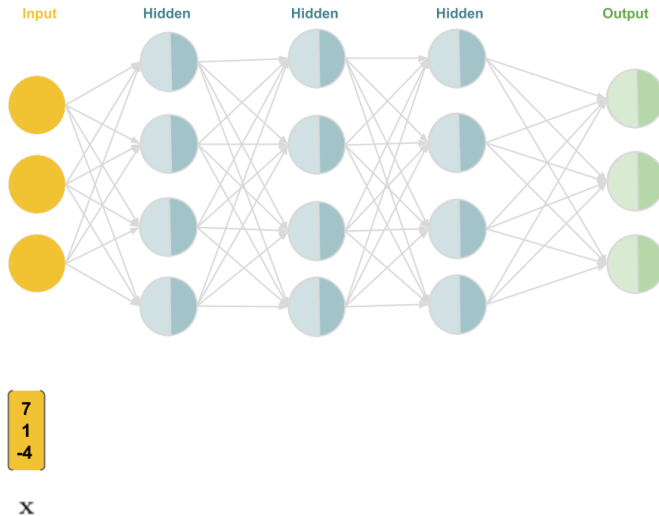
FEEDFORWARD NEURAL NETWORKS: EXAMPLE



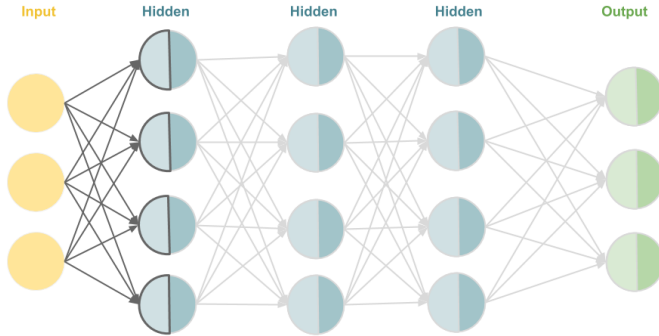
$$\begin{pmatrix} 13 & -9 & 2 \\ -8 & 0 & 3 \\ 4 & -1 & 5 \\ -3 & 12 & 7 \end{pmatrix} \begin{pmatrix} 5 \\ -2 \\ 2 \\ 11 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & -4 & 1 \\ 0 & 11 & 2 & -14 \\ -1 & 5 & -2 & 16 \\ 0 & -9 & -3 & 4 \end{pmatrix} \begin{pmatrix} -5 \\ 3 \\ 1 \\ -8 \end{pmatrix} \quad \begin{pmatrix} 1 & -2 & -18 & -7 \\ 3 & -4 & 8 & 0 \\ -2 & 1 & 21 & 5 \\ 2 & -2 & 11 & -13 \end{pmatrix} \begin{pmatrix} 4 \\ -6 \\ 1 \\ -17 \end{pmatrix} \quad \begin{pmatrix} 9 & 3 & -1 & -4 \\ -8 & -2 & 14 & 3 \\ 13 & 2 & -9 & -1 \end{pmatrix} \begin{pmatrix} -1 \\ -4 \\ -30 \end{pmatrix}$$

$(W^{(1)})^T \quad \mathbf{b}^{(1)} \quad (W^{(2)})^T \quad \mathbf{b}^{(2)} \quad (W^{(3)})^T \quad \mathbf{b}^{(3)} \quad U^T \quad \mathbf{c}$

FEEDFORWARD NEURAL NETWORKS: EXAMPLE

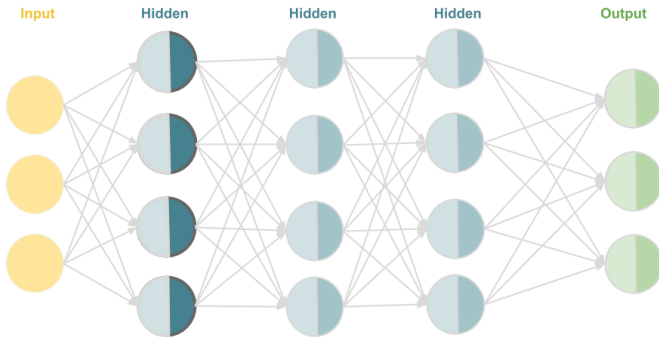


FEEDFORWARD NEURAL NETWORKS: EXAMPLE



$$\begin{bmatrix} 7 \\ 1 \\ -4 \end{bmatrix} \begin{pmatrix} 7*13 + 1*(-9) + (-4)*2 + 5 \\ 7*(-8) + 1*0 + (-4)*3 + (-2) \\ 7*4 + 1*(-1) + (-4)*5 + 2 \\ 7*(-3) + 1*12 + (-4)*7 + 11 \end{pmatrix}$$
$$\mathbf{x} \quad \mathbf{z}_{in}^{(1)} = \mathbf{W}^{(1)T} \mathbf{x} + \mathbf{b}^{(1)}$$

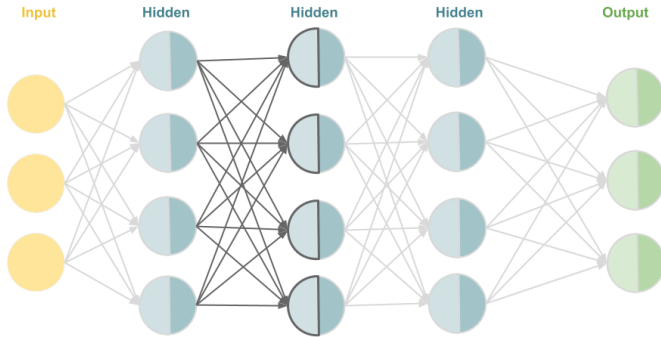
FEEDFORWARD NEURAL NETWORKS: EXAMPLE



$$\begin{bmatrix} 7 \\ 1 \\ -4 \end{bmatrix} \quad \begin{bmatrix} 79 \\ -70 \\ 9 \\ -26 \end{bmatrix} \quad \begin{bmatrix} \max(0, 79) \\ \max(0, -70) \\ \max(0, 9) \\ \max(0, -26) \end{bmatrix}$$

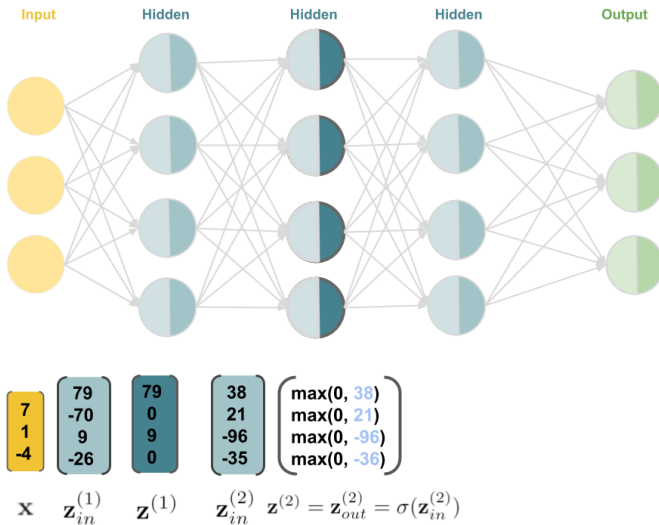
$$\mathbf{x} \quad \mathbf{z}_{in}^{(1)} \quad \mathbf{z}^{(1)} = \mathbf{z}_{out}^{(1)} = \sigma(\mathbf{z}_{in}^{(1)})$$

FEEDFORWARD NEURAL NETWORKS: EXAMPLE

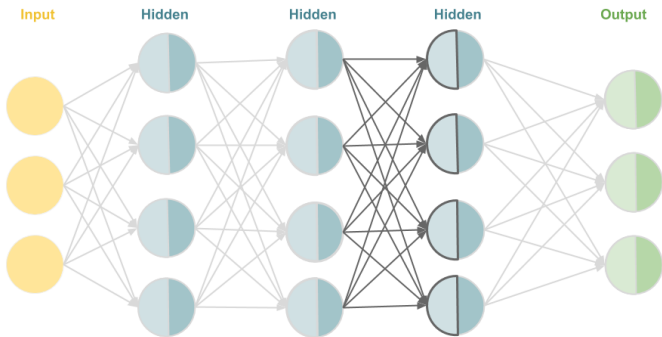


$$\begin{array}{c}
 \begin{bmatrix} 7 \\ 1 \\ -4 \end{bmatrix} \quad \begin{bmatrix} 79 \\ -70 \\ 9 \\ -26 \end{bmatrix} \quad \begin{bmatrix} 79 \\ 0 \\ 9 \\ 0 \end{bmatrix} \quad \left(\begin{array}{l} 79*1 + 0*0 + 9*(-4) + 0*1 + (-5) \\ 79*0 + 0*11 + 9*2 + 0*(-14) + 3 \\ 79*(-1) + 0*5 + 9*(-2) + 0*16 + 1 \\ 79*0 + 0*(-9) + 9*(-3) + 0*4 + (-8) \end{array} \right) \\
 \mathbf{x} \quad \mathbf{z}_{in}^{(1)} \quad \mathbf{z}^{(1)} \quad \mathbf{z}_{in}^{(2)} = W^{(2)T} \mathbf{z}^{(1)} + \mathbf{b}^{(2)}
 \end{array}$$

FEEDFORWARD NEURAL NETWORKS: EXAMPLE



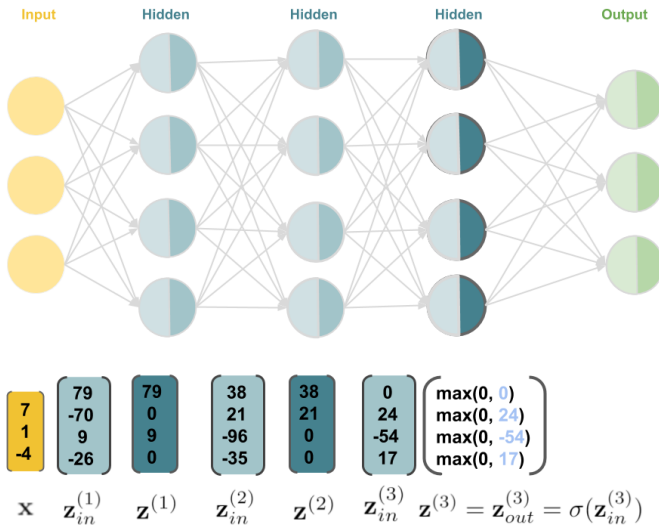
FEEDFORWARD NEURAL NETWORKS: EXAMPLE



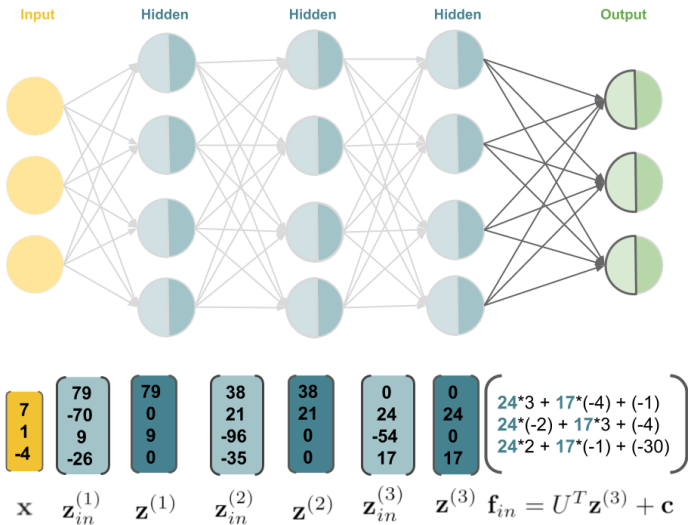
$$\begin{array}{c}
 \boxed{\begin{matrix} 7 \\ 1 \\ -4 \end{matrix}} \quad \boxed{\begin{matrix} 79 \\ -70 \\ 9 \\ -26 \end{matrix}} \quad \boxed{\begin{matrix} 79 \\ 0 \\ 9 \\ 0 \end{matrix}} \quad \boxed{\begin{matrix} 38 \\ 21 \\ -96 \\ -35 \end{matrix}} \quad \boxed{\begin{matrix} 38 \\ 21 \\ 0 \\ 0 \end{matrix}} \quad \left(\begin{array}{l} 38*1 + 21*(-2) + 0*(-18) + 0*(-7) + 4 \\ 38*3 + 21*(-4) + 0*8 + 0*0 + (-6) \\ 38*(-2) + 21*1 + 0*21 + 0*5 + 1 \\ 38*2 + 21*(-2) + 0*11 + 0*(-13) + (-17) \end{array} \right)
 \end{array}$$

$$\mathbf{x} \quad \mathbf{z}_{in}^{(1)} \quad \mathbf{z}^{(1)} \quad \mathbf{z}_{in}^{(2)} \quad \mathbf{z}^{(2)} \quad \mathbf{z}_{in}^{(3)} = \mathbf{W}^{(3)T} \mathbf{z}^{(2)} + \mathbf{b}^{(3)}$$

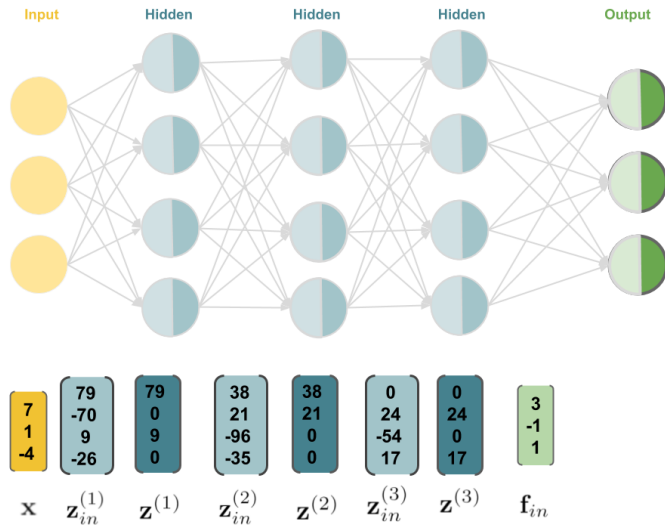
FEEDFORWARD NEURAL NETWORKS: EXAMPLE



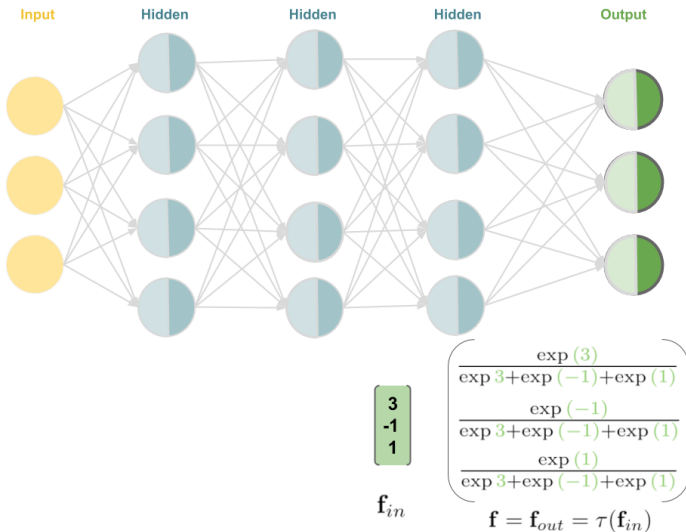
FEEDFORWARD NEURAL NETWORKS: EXAMPLE



FEEDFORWARD NEURAL NETWORKS: EXAMPLE



FEEDFORWARD NEURAL NETWORKS: EXAMPLE



FEEDFORWARD NEURAL NETWORKS: EXAMPLE

