Deep Learning

1/ Neural Networks

Francesca Galassi, MCF, Esir

francesca.galassi@irisa.fr
Lab Empenn Irisa-Inria

Introduction

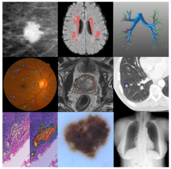
Course Overview

- Objectives :
  - ➟ Understand the basics of deep learning and neural networks.
  - ➟ Create and test your own neural network models.
  - ➟ Apply deep learning to practical tasks like image recognition.
  - ➟ Stay updated on the latest developments in the field.
- Structure :
  - ➟ Lectures (CM) : 7 sessions of 1.5 hours each.
  - ➟ Practicals (TP) : 4 sessions of 3 hours each using TensorFlow and Keras.
    Instructor : Jeremy Lefort-Besnard, *jeremy.lefort-besnard@inria.fr*
  - ➟ Evaluation (CC) : Exam scheduled for **May 13, 2024**.

Key Applications

➠ Medical Imaging

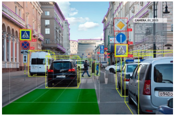
- Input (x) : Medical scans
- Output (y) : Disease diagnosis
- Model : Convolutional Neural Network



Litjens et al.(2019), *A survey on Deep Learning in MIA*

➠ Autonomous Driving

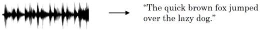
- Input (x) : Image
- Output (y) : Car position
- Model : Convolutional Neural Network



MIT : https://www.moralmachine.net/

➠ Speech Recognition

- Input (x) : Audio
- Output (y) : Text transcript
- Model : Recurrent Neural Network (RNN)



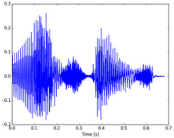
Mapping audio (x) to text transcripts (y)

Data Types

- Structured data : Data that follows a standardized format, has a well-defined structure, and is easily accessed by humans and computers. It is typically stored in a database.

Size	#bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
⋮	⋮	⋮
3000	4	540

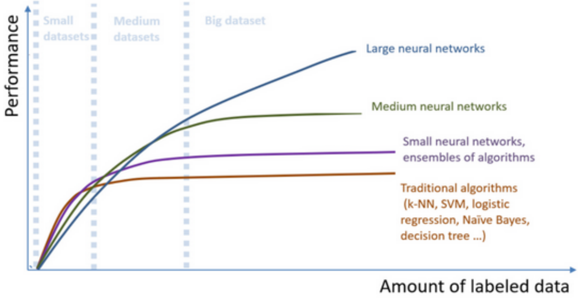
- Unstructured data : Data that is not stored in a structured database format, such as text and multimedia.



Dies ist ein Blindtext. An ihm lässt sich vieles über die Schrift ablesen, in der er gesetzt ist. Auf den ersten Blick wird der Grauwert der Schriftfläche sichtbar. Dann kann man prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt.

Scale Drives Deep Learning

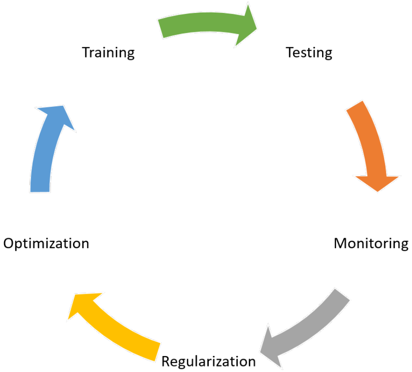
➡ The effectiveness of machine learning algorithms is influenced by both the algorithm itself and the volume of data used. [1]



1. Mukhamediev et al. From Classical Machine Learning to Deep Neural Networks. Appl. Sci. 2021, 11, 5541.

Scale Drives Deep Learning

- Large amount of training data.
- Fast computation using specialized hardware, e.g., GPUs.
- Algorithmic innovations, e.g., from Sigmoid to ReLU activation function.



Logistic Regression

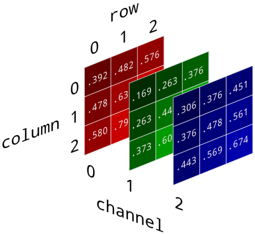
Binary Classification

- Classification of an image :

Sample Image (ImageNet) 128x128

$\longrightarrow$ Output label y : 1 (fox) or 0 (non-fox)

- Image representation on a computer : 3 matrices (i.e., Red, Green, Blue channels)



$$\text{Input } x = \begin{bmatrix} 255 \\ 231 \\ \vdots \\ 134 \\ \vdots \end{bmatrix} \quad ; \; n = 128x128x3 = 49152$$

Notation

- Training set :

  a training example is a pair (x, y) where $x \in \mathbb{R}^n, y \in \{0, 1\}$

  m training examples : $\left\{\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \ldots, \left(x^{(m)}, y^{(m)}\right)\right\}$

  Compact notation :

- Input features :

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

- Output labels :

$$Y = \left[y^{(1)} y^{(2)} \ldots y^{(m)}\right]$$

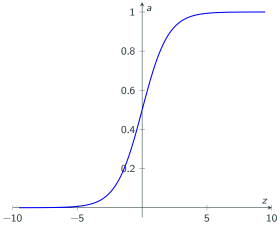
$$Y \in \mathbb{R}^{1 \times m}$$

$$Y.\text{shape} = (1, m)$$

Logistic Regression Model

- Given $x \in \mathbb{R}^n$, find $\hat{y} = P(y = 1 \mid x)$
- Parameters : $w \in \mathbb{R}^n, b \in \mathbb{R}$

➡ Output of Logistic Regression : $\hat{y} = \sigma\left(w^T x + b\right)$, $0 \leqslant \hat{y} \leqslant 1$



Sigmoid Function : $\sigma(z) = \frac{1}{1+e^{-z}}$

- If z is large positive, $\sigma(z) \approx \frac{1}{1+0} = 1$
- If z is large negative, $\sigma(z) \approx \frac{1}{1+big} \approx 0$

Logistic Regression Model : Cost Function

- Given a training set $\left\{\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(m)}, y^{(m)}\right)\right\}$, find w and b $\mid$ $\hat{y}^{(i)} \approx y^{(i)}$
  - $\hat{y}^{(i)}$ is the prediction and $y^{(i)}$ is the ground truth for the i-th sample
- For each training example i :

$$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b), \text{ where } \sigma\left(z^{(i)}\right) = \frac{1}{1 + e^{-z^{(i)}}}, \text{ and } z^{(i)} = w^T x^{(i)} + b$$

- Loss function, i.e., the error for a single training example :

$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

$$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

- If $y = 1$: $\mathcal{L}(\hat{y}, y) = -\log \hat{y}$ $\leftarrow$ want $\log \hat{y}$ large, want $\hat{y}$ large
- If $y = 0$: $\mathcal{L}(\hat{y}, y) = -\log(1 - \hat{y})$ $\leftarrow$ want $\log(1 - \hat{y})$ large, want $\hat{y}$ small
- Remark : $0 \leqslant \hat{y} \leqslant 1$

Logistic Regression Model : Cost Function

- Loss Function, i.e., the error for a single training example :

$$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

$$0 \leqslant \hat{y} \leqslant 1$$

- Cost Function, i.e., the average of the loss functions on the **entire** training set :

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)\right]$$

- The training objective is to find w and b that minimize the cost function $J(w, b)$.

Computation Graph

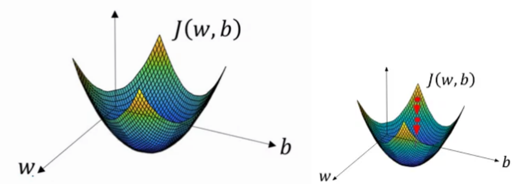
Gradient Descent

➟ Cost Function, i.e., the average of the loss functions on the **entire** training set :

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\hat{y}^{(i)} + \left(1-y^{(i)}\right)\log\left(1-\hat{y}^{(i)}\right)\right]$$

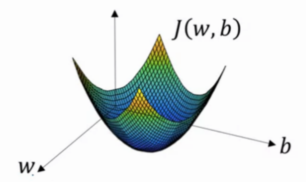
➟ The training objective is to find w and b that minimize the cost function $J(w,b)$.

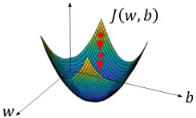
➟ Gradient Descent Algorithm : Learning the parameters w and b.



Gradient Descent

➡ Gradient Descent Algorithm : Moving towards the global optimum by taking steps in the steepest downhill direction.





i. Initialize parameters w and b (e.g., to zeros or random values) ;

ii. Repeat until convergence :

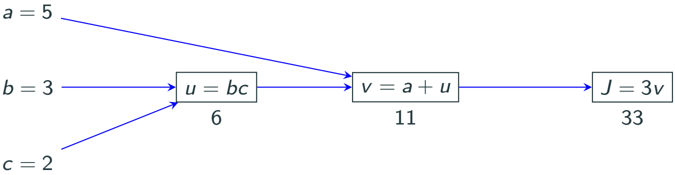
$$w := w - \alpha \frac{dJ(w, b)}{dw}$$

$$b := b - \alpha \frac{dJ(w, b)}{db} \qquad , \alpha \text{ is the learning rate}$$

iii. Return parameters.

Computation Graph

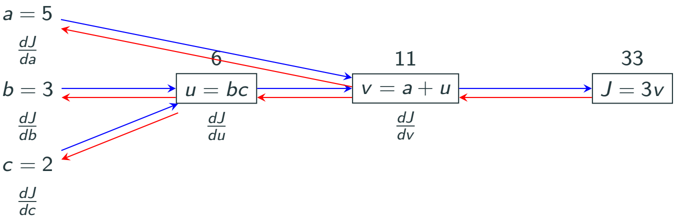
- Neural Networks Computations : a forward pass for output computation, a backward pass for gradient computation.
- The computation graph organizes function computation left-to-right, i.e., forward pass.
- Example : Calculate the value of the output variable $J(a, b, c) = 3(a + bc)$.



Derivatives with a Computation Graph

➟ The computation graph organizes the computation of derivatives right-to-left, i.e., backward pass.

- Example : compute the partial derivatives of $J(a, b, c) = 3(a + bc)$.



Using the chain rule :

$$\frac{dJ}{du} = \underbrace{\frac{dJ}{dv}}_{3} \cdot \underbrace{\frac{dv}{du}}_{1} = 3 \qquad \frac{dJ}{dc} = \underbrace{\frac{dJ}{du}}_{3} \cdot \underbrace{\frac{du}{dc}}_{3} = 9$$

$$\frac{dJ}{da} = \underbrace{\frac{dJ}{dv}}_{3} \cdot \underbrace{\frac{dv}{da}}_{1} = 3 \qquad \frac{dJ}{db} = \underbrace{\frac{dJ}{du}}_{3} \cdot \underbrace{\frac{du}{db}}_{1} = 3$$

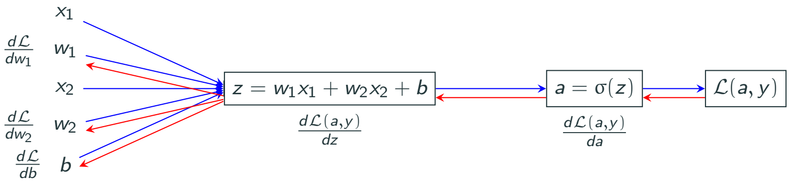
Computation Graph for Logistic Regression

$$z = w^T x + b$$

$$\hat{y} = a = \sigma(z)$$

$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$

➟ Computation graph for one training example and two features :



➟ One step backward :

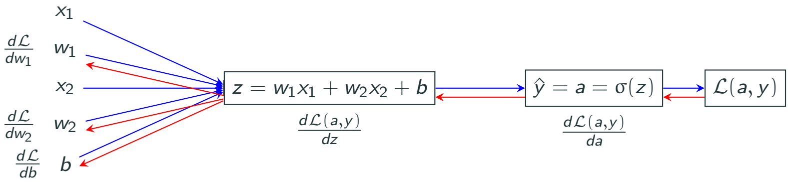
$$\frac{d\mathcal{L}(a, y)}{da} = -\frac{y}{a} + \frac{1 - y}{1 - a}$$

➟ Applying the chain rule :

$$\frac{d\mathcal{L}(a, y)}{dz} = \frac{d\mathcal{L}(a, y)}{da}\frac{da}{dz} = (-\frac{y}{a} + \frac{1 - y}{1 - a}) \cdot a(1 - a) = a - y$$

Computation Graph for Logistic Regression

➟ Computation graph for one training example and two features :



➟ Applying the chain rule :

$$\frac{d\mathcal{L}(a,y)}{dz} = \frac{d\mathcal{L}(a,y)}{da} \cdot \frac{da}{dz} = (-\frac{y}{a} + \frac{1-y}{1-a}) \cdot a(1-a) = a - y$$

➟ Weights and bias :

$$\frac{d\mathcal{L}(a,y)}{dw_1} = \frac{d\mathcal{L}(a,y)}{dz} \cdot \frac{dz}{dw_1} = x_1 \cdot (a-y)$$

$$\frac{d\mathcal{L}(a,y)}{dw_2} = x_2 \cdot (a-y) \quad ; \quad \frac{d\mathcal{L}(a,y)}{db} = a - y$$

Computation Graph for Logistic Regression

- Applying the chain rule :

$$\frac{d\mathcal{L}(a,y)}{dz} = \frac{d\mathcal{L}(a,y)}{da} \cdot \frac{da}{dz} = (-\frac{y}{a} + \frac{1-y}{1-a}) \cdot a(1-a) = a - y$$

$$\frac{d\mathcal{L}(a,y)}{dw_1} = \frac{d\mathcal{L}(a,y)}{dz} \cdot \frac{dz}{dw_1} = x_1 \cdot (a-y)$$

$$\frac{d\mathcal{L}(a,y)}{dw_2} = x_2 \cdot (a-y) \quad ; \quad \frac{d\mathcal{L}(a,y)}{db} = a - y$$

- Update the parameters (one step of gradient descent) :

$$w_1 := w_1 - \alpha \frac{d\mathcal{L}}{dw_1}$$
$$w_2 := w_2 - \alpha \frac{d\mathcal{L}}{dw_2}$$
$$b := b - \alpha \frac{d\mathcal{L}}{db}$$

Gradient Descent on m Examples

➟ Cost function for m training examples :

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(a^{(i)}, y^{(i)}\right)$$

$$a^{(i)} = \hat{y}^{(i)} = \sigma\left(z^{(i)}\right) = \sigma\left(w^{\top} x^{(i)} + b\right)$$

➟ For instance, the derivative for m training example wrt to w_1 :

$$\frac{dJ(w, b)}{dw_1} = \frac{1}{m} \sum_{i=1}^{m} \frac{d\mathcal{L}\left(a^{(i)}, y^{(i)}\right)}{dw_1}$$

➟ Compute the derivative on each training example and average them.

Gradient Descent on m Examples

- To optimize the parameters, we compute the derivative on m training examples and average them (assuming 2 features).

(i.) Initialization :

$$J = 0\,;\; dw_1 = \frac{dJ}{dw_1} = 0\,;\; dw_2 = \frac{dJ}{dw_2} = 0\,;\; db = \frac{dJ}{db} = 0$$

(ii.)

For $i = 1$ to m

$$z^{(i)} = w^T x^{(i)} + b$$

$$J \mathrel{+}= -\left[y^{(i)} \log a^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - a^{(i)}\right)\right]$$

$$dz^{(i)} = a^{(i)} - y^{(i)}$$

$$dw_1 \mathrel{+}= x_1^{(i)} \cdot dz^{(i)}$$

$$dw_2 \mathrel{+}= x_2^{(i)} \cdot dz^{(i)}$$

$$db \mathrel{+}= dz^{(i)}$$

Gradient Descent on m Examples

(iii). Compute the derivatives for m training examples and average them ($n = 2$ features) :

$$J \mathrel{/}= m\,;\; dw_1 \mathrel{/}= m\,;\; dw_2 \mathrel{/}= m\,;\; db \mathrel{/}= m$$

(vi). Update the parameters (one step of gradient descent) :

$$w_1 := w_1 - \alpha \cdot dw_1$$

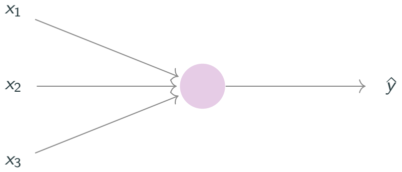
$$w_2 := w_2 - \alpha \cdot dw_2$$

$$b := b - \alpha \cdot db$$

Neural Networks

What is a Neural Network ?

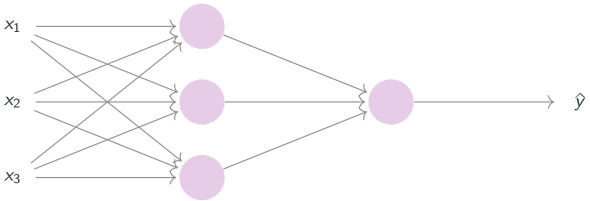
➟ Logistic Regression as :



Computations within the node :

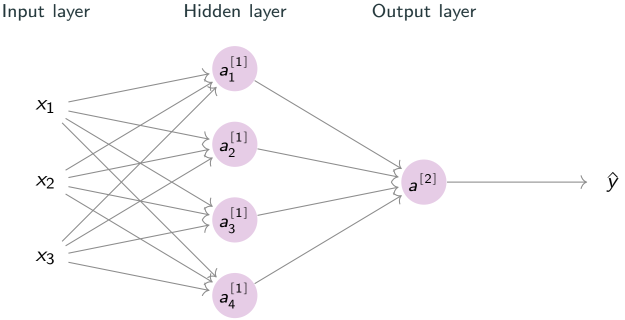
1. $z = w^T x + b$
2. $a = \sigma(z)$

➟ By stacking multiple nodes together :



Neural Network Representation

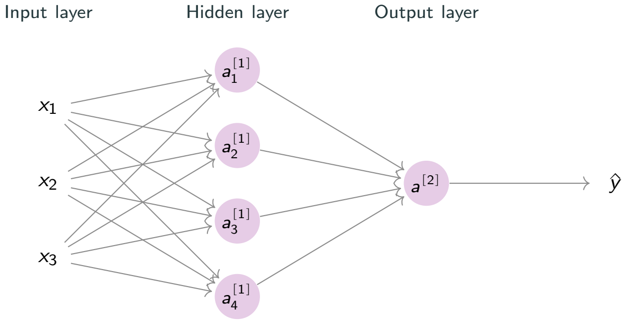
➟ 2-layer Neural Network (NN) :



- The input to a NN is a set of features x_i.
- The output is the predicted value $\hat{y}$, generated by the output layer.
- The intermediate variables $a_i^{[l]}$ are the hidden units or hidden neurons :
  - ➟ Layer l, unit i ;
  - ➟ Hidden because their *true* values are not observed in the training dataset.

Neural Network Representation

➟ 2-layer Neural Network (NN) :

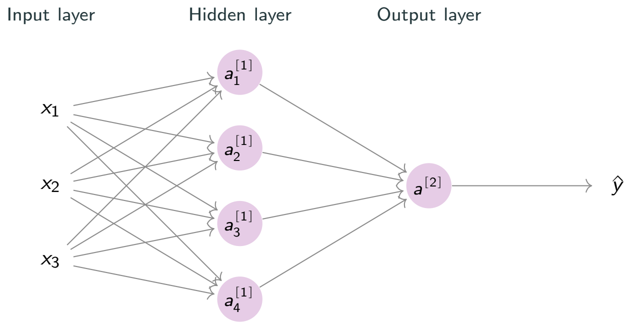


- $a^{[1]}$ is a 4-dimensional vector : $a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix}.$

- The output $\hat{y}$ is equal to $a^{[2]}$.

Neural Network Representation

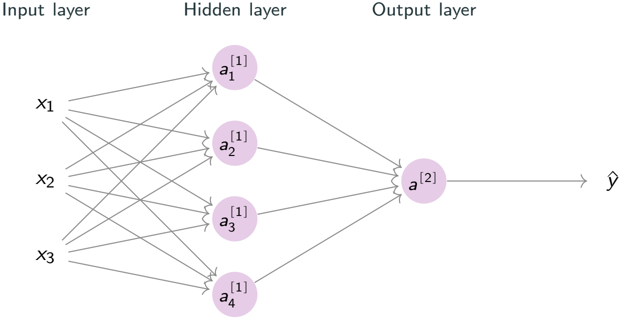
➟ 2-layer Neural Network (NN) :



- Parameters are associated with hidden and output layers :
  - Hidden layer (layer 1) : $w^{[1]}$ matrix of size $[4, 3]$, $b^{[1]}$ vector of size $[4, 1]$;
  - Output layer (layer 2) : $w^{[2]}$ vector of size $[1, 4]$, $b^{[2]}$ vector of size $[1, 1]$;
  - The size of $w^{[l]}$ is [*number of units, number of input features/activations*].

Neural Network Computations

➟ 2-layer Neural Network (NN) :

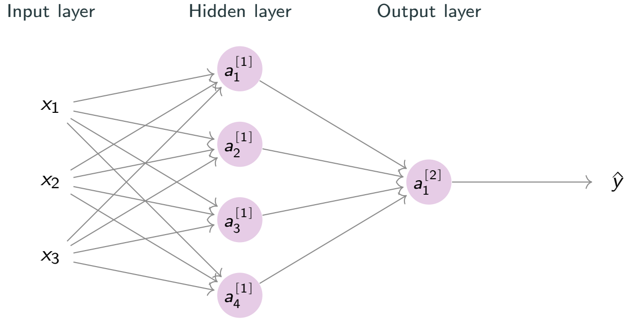


- Within each unit, a two-step computation :

1. $z_i^{[l]} = w_i^{[l]T} x + b_i^{[l]}$
2. $a_i^{[l]} = \sigma\left(z_i^{[l]}\right)$

➟ **Remark** : Logistic Regression is performed at each unit.

Neural Network Computations



- Computations in the hidden layer (or layer 1) at each unit :

$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]} \quad ; \quad a_1^{[1]} = \sigma\left(z_1^{[1]}\right)$$
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]} \quad ; \quad a_2^{[1]} = \sigma\left(z_2^{[1]}\right)$$
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]} \quad ; \quad a_3^{[1]} = \sigma\left(z_3^{[1]}\right)$$
$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]} \quad ; \quad a_4^{[1]} = \sigma\left(z_4^{[1]}\right)$$

Neural Network Computations

- Computations in the hidden layer (or layer 1) at each $a_i^{[1]}$ unit :

$$
\begin{aligned}
z_1^{[1]} &= w_1^{[1]T} x + b_1^{[1]} \quad ; \quad a_1^{[1]} = \sigma\left(z_1^{[1]}\right) \\
z_2^{[1]} &= w_2^{[1]T} x + b_2^{[1]} \quad ; \quad a_2^{[1]} = \sigma\left(z_2^{[1]}\right) \\
z_3^{[1]} &= w_3^{[1]T} x + b_3^{[1]} \quad ; \quad a_3^{[1]} = \sigma\left(z_3^{[1]}\right) \\
z_4^{[1]} &= w_4^{[1]T} x + b_4^{[1]} \quad ; \quad a_4^{[1]} = \sigma\left(z_4^{[1]}\right)
\end{aligned}
$$

➟ Vectorization : to make neural network computations quicker - than using a *for* loop.

➟ Vectorized notation, i.e., by stacking units vertically :

$$
\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_2^{[1]} \\ z_4^{[1]} \end{bmatrix} = \underbrace{\begin{bmatrix} - & w_1^{[1]T} & - \\ - & w_2^{[1]T} & - \\ - & w_3^{[1]T} & - \\ - & w_4^{[1]T} & - \end{bmatrix}}_{\text{matrix } (4,3)} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \underbrace{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}}_{\text{vector } (4,1)}
$$

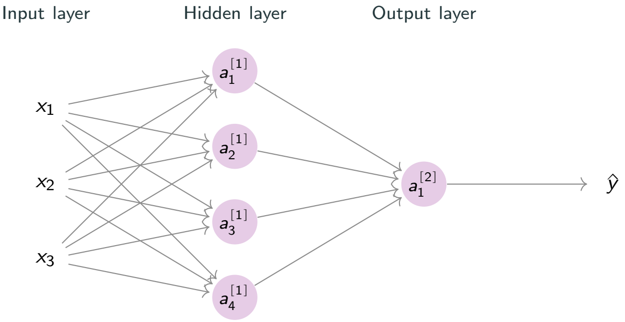
Neural Network Computations

➟ Vectorized notation 1st computation :

$$z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_2^{[1]} \\ z_4^{[1]} \end{bmatrix} = \underbrace{\overbrace{\begin{bmatrix} - & w_1^{[1]T} & - \\ - & w_2^{[1]T} & - \\ - & w_3^{[1]T} & - \\ - & w_4^{[1]T} & - \end{bmatrix}}^{W^{[1]}}}_{\text{matrix } (4,3)} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \underbrace{\overbrace{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}}^{b^{[1]}}}_{\text{vector } (4,1)}$$

➟ Vectorized notation 2nd computation : $a^{[1]} = \underbrace{\begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix}}_{\text{vector } (4,1)} = \sigma\left(z^{[1]}\right)$

Neural Network computations



➟ Given an input x, vectorized representation of the computations :

$$\left.\begin{aligned} \underset{(4,1)}{z}^{[1]} &= \underset{(4,3)}{W}^{[1]} \underset{(3,1)}{x} + \underset{(4,1)}{b}^{[1]} \\ \underset{(4,1)}{a}^{[1]} &= \sigma\left(z^{[1]}\right) \end{aligned}\right\} \text{layer 1}$$

$$\left.\begin{aligned} \underset{(1,1)}{z}^{[2]} &= \underset{(1,4)}{W}^{[2]} \underset{(4,1)}{a}^{[1]} + \underset{(1,1)}{b}^{[2]} \\ \underset{(1,1)}{a}^{[2]} &= \sigma\left(z^{[2]}\right) \end{aligned}\right\} \text{layer 2}$$

Neural Network Computations

➟ Vectorizing across multiple examples

➟ For a single input training example x :

$$x \xrightarrow{\text{compute}} a^{[2]} = \hat{y}$$

➟ When dealing with m training examples :

$$x^{(1)} \longrightarrow a^{[2](1)} = \hat{y}^{(1)}$$

$$x^{(2)} \longrightarrow a^{[2](2)} = \hat{y}^{(2)}$$

...

$$x^{(m)} \longrightarrow a^{[2](m)} = \hat{y}^{(m)}$$

➟ For m training examples :

for $i = 1$ to m

$$z^{[1](i)} = W^{[1]} x^{(i)} + b^{[1]}$$

$$a^{[1](i)} = \sigma\left(z^{[1](i)}\right)$$

$$z^{[2](i)} = W^{[2]} a^{[1](i)} + b^{[2]}$$

$$a^{[2](i)} = \sigma\left(z^{[2](i)}\right)$$

Neural Network Computations

- For m training examples :

$$
\begin{aligned}
&\text{for } i = 1 \text{ to } m\\
&\quad z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}\\
&\quad a^{[1](i)} = \sigma\left(z^{[1](i)}\right)\\
&\quad z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}\\
&\quad a^{[2](i)} = \sigma\left(z^{[2](i)}\right)
\end{aligned}
$$

- Vectorizing across the m examples :

$$
X = \underbrace{\begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}}_{(n,m)}
$$

$$
Z = \underbrace{\begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}}_{(\text{number of layer units}, m)}
$$

$$
A = \underbrace{\begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}}_{(\text{number of layer units}, m)}
$$

Neural Network Computations

- For m training examples :

  for $i = 1$ to m

$$z^{[1](i)} = W^{[1]} x^{(i)} + b^{[1]}$$

$$a^{[1](i)} = \sigma\left(z^{[1](i)}\right)$$

$$z^{[2](i)} = W^{[2]} a^{[1](i)} + b^{[2]}$$

$$a^{[2](i)} = \sigma\left(z^{[2](i)}\right)$$

- Vectorizing across m training examples :

$$Z^{[1]} = W^{[1]} X + b^{[1]}$$

$$A^{[1]} = \sigma\left(Z^{[1]}\right)$$

$$Z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$$

$$A^{[2]} = \sigma\left(Z^{[2]}\right)$$