CMI LAB

UNIVERSITY OF WOLVERHAMPTON
KNOWLEDGE ▪ INNOVATION ▪ ENTERPRISE

6CS012 – Artificial Intelligence and Machine Learning

Ahsan Adeel

Theme lead, Conscious Multisensory Integration (CMI) Lab

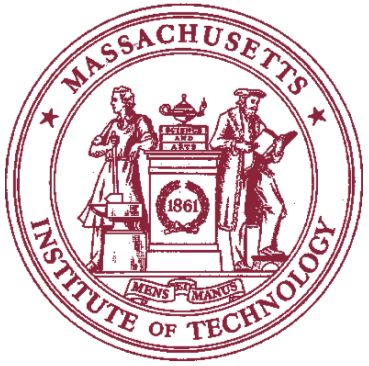
Fellow, MIT Synthetic Intelligence Lab, MIT

and Oxford Computational Neuroscience Lab, University of Oxford

Visiting EPSRC/MRC fellow, University of Stirling

http://www.cmilab.org/

http://www.cmilab.org/dr_ahsan_adeel.html#Home



NUFFIELD DEPARTMENT OF SURGICAL SCIENCES
Medical Sciences Division

Lecture 6 – Neural Network and Backpropagation

Lecture 5 Review
(Pay attention to the whiteboard)

WS5 – Task 1
(MNIST classification using Batch GD (BGD) driven softmax regression)

- **Download mnist folder from Canvas**
- **The provided code will extract and pre-process the dataset**
- **The provided code also does the post-processing including thresholding, accuracy etc.**
- **For your ease, following lines are left empty for you to fill.**

err =
gradients =
theta =
y_predict =

Note that batch gradient descent computes the gradient using the whole dataset

1.1. Use BGD and complete the model
1.2. Explain in detail (step by step) the whole process in your own words, including:
- **raw data**
- **pre-processing (both X and Y (onehot encoded))**
- **Number of iterations/epochs**
- **Calculation of above 4 lines**

1.3. Test the model on test data and comment on the results in detail

WS5 – Task 2
(MNIST classification using Stochastic GD (SGD) driven softmax regression)

- **Download mnist.pkl.gz from Canvas**
- **The provided code will extract and pre-process the dataset**
- **The provided code also does the post-processing including thresholding, accuracy etc.**
- **For your ease, following lines are left empty for you to fill.**

err =
gradients =
Updated theta =
y_predict =

Note that stochastic gradient descent computes the gradient using small batches

2.1. Use SGD and complete the model

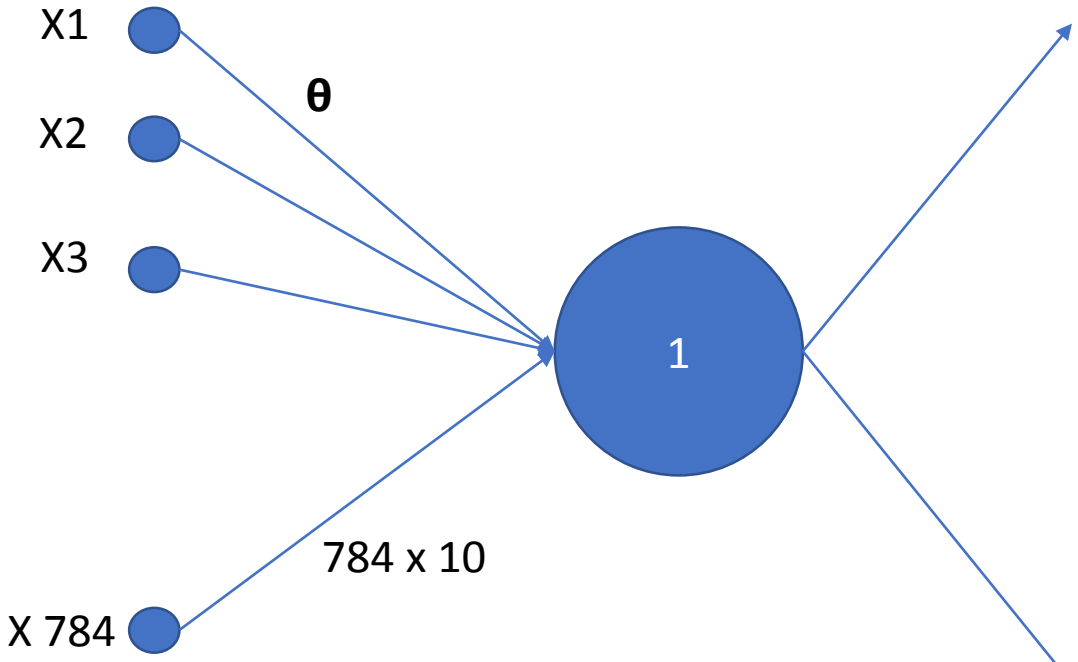
2.2. Explain in detail (step by step) the whole process in your own words, including:

- **raw data**
- **Number of iterations/epochs**
- **pre-processing (mini-batches)**
- **Calculation of above 4 lines**
- **difference between task 1 and 2, in terms of accuracy**

2.3. Test the model on test data and comment on the results in detail

Let's first use a Single Neuron to recognize handwritten digits

input layer
(784 neurons)



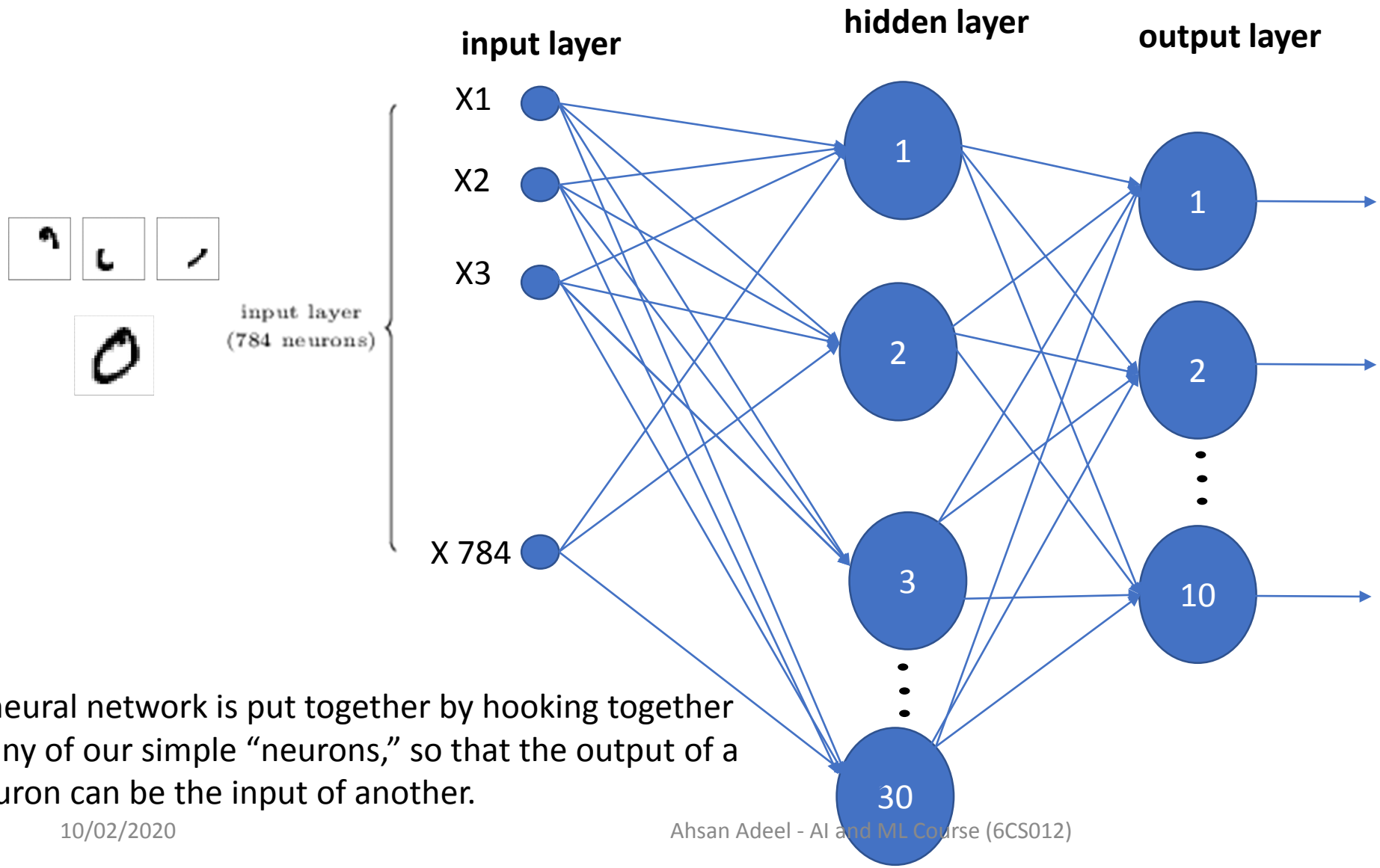
$$\boldsymbol{\theta}^{(\text{next step})} = \boldsymbol{\theta} - \eta \nabla_{\theta} J(\theta)$$

784 x 10

784 x 10

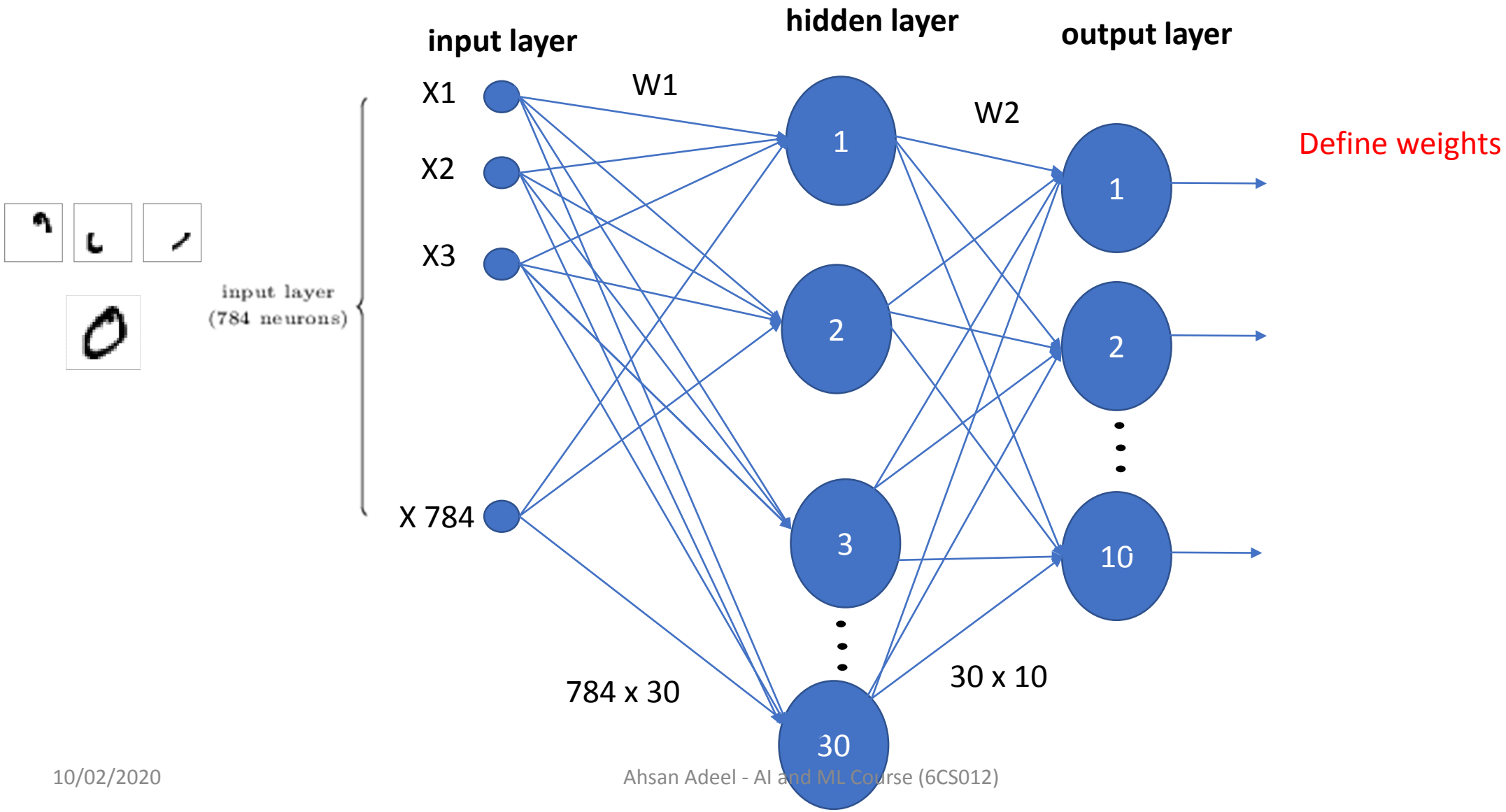
$$\nabla_{\boldsymbol{\theta}} \text{MSE}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \text{MSE}(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_1} \text{MSE}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \text{MSE}(\boldsymbol{\theta}) \end{pmatrix} = \frac{2}{m} \mathbf{X}^T (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

Let's use multiple neurons to recognize handwritten digits

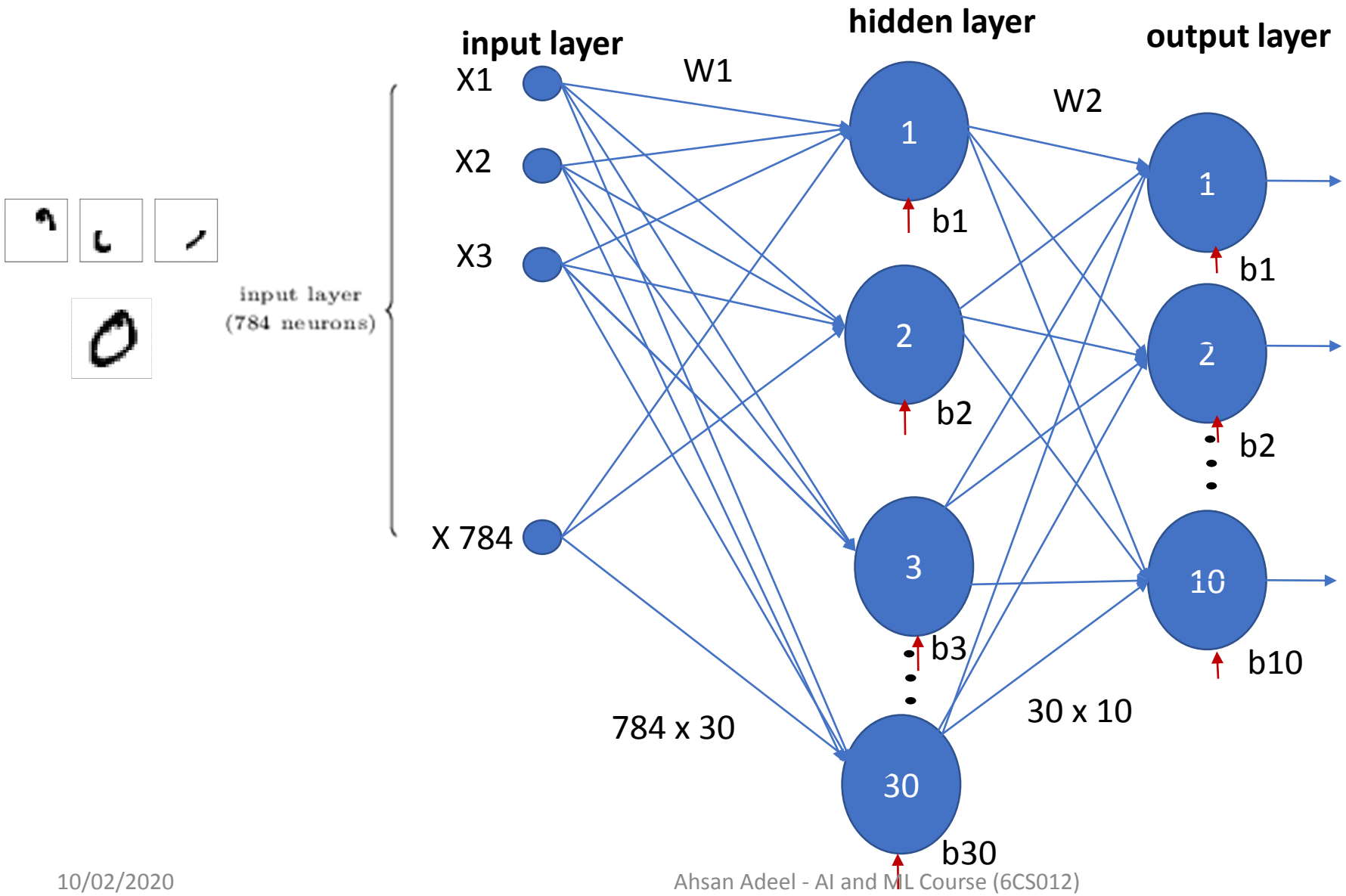


A neural network is put together by hooking together many of our simple "neurons," so that the output of a neuron can be the input of another.

Let's use multiple neurons to recognize handwritten digits



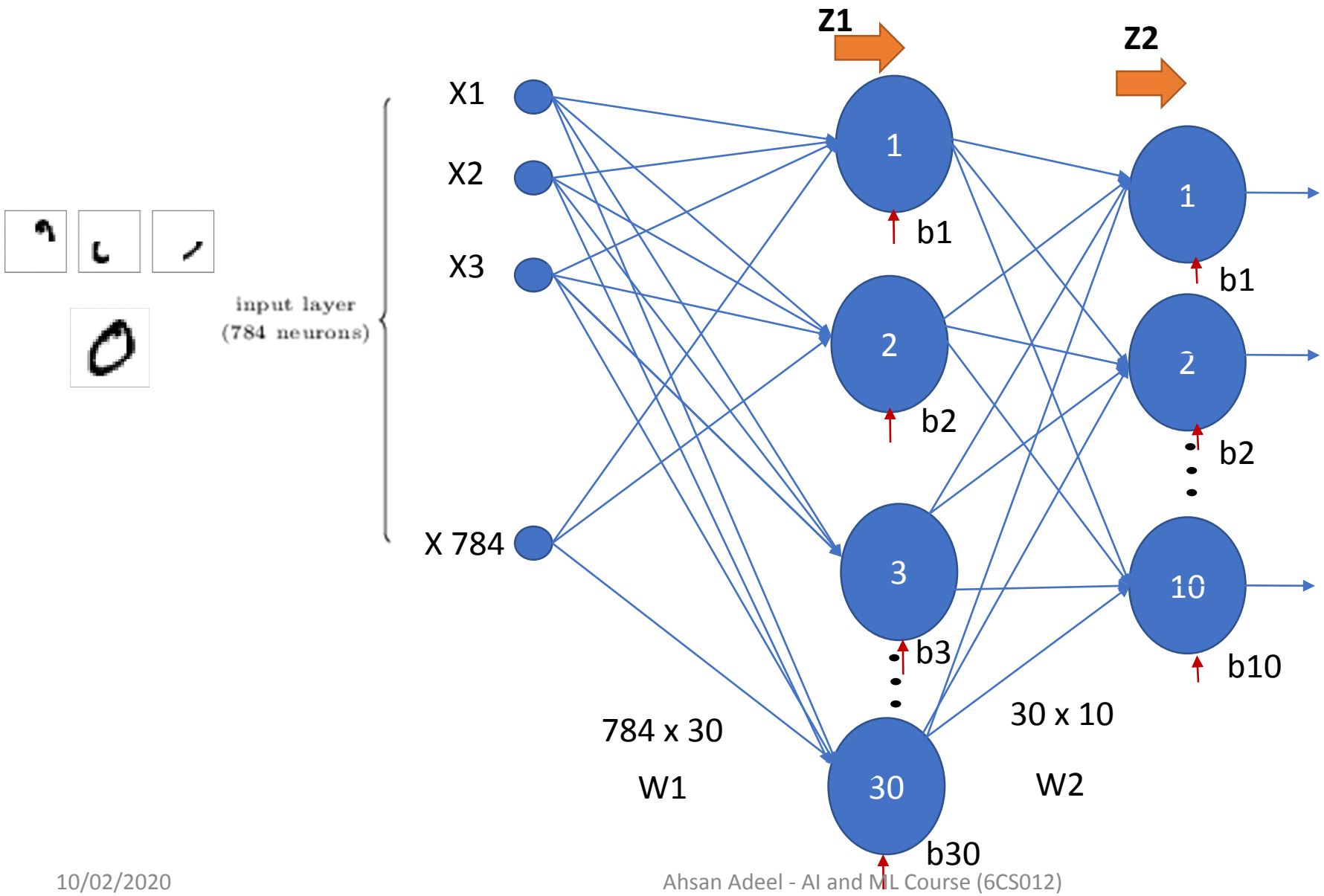
Let's use multiple neurons to recognize handwritten digits



Define biases

Can you recall the **bias** term from Lecture 1 or 2?

Let's use multiple neurons to recognize handwritten digits

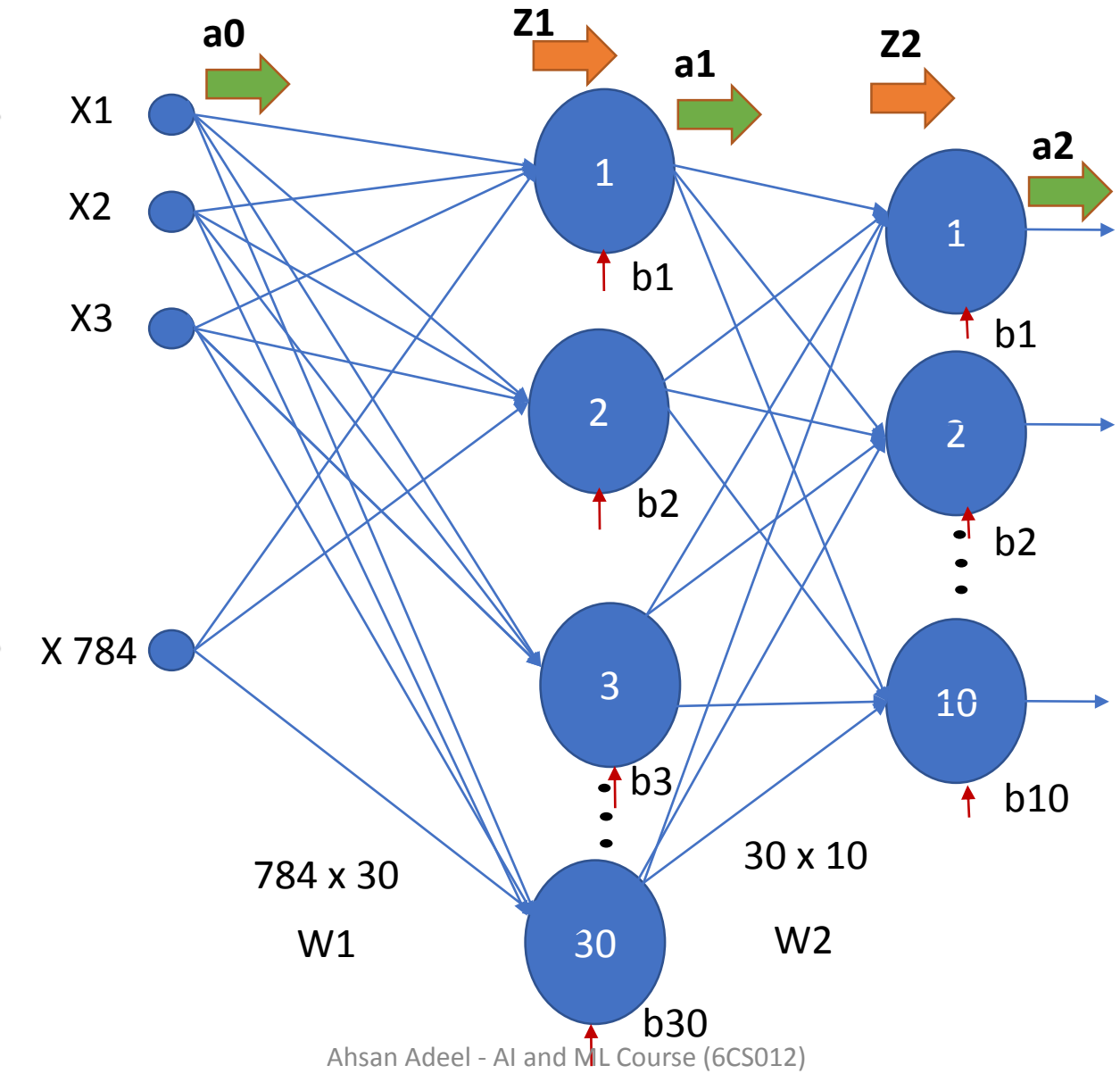


Define weighted input

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l,$$

$$Z^l = W^l a^{l-1} + b^l$$

Let's use multiple neurons to recognize handwritten digits



Define activations

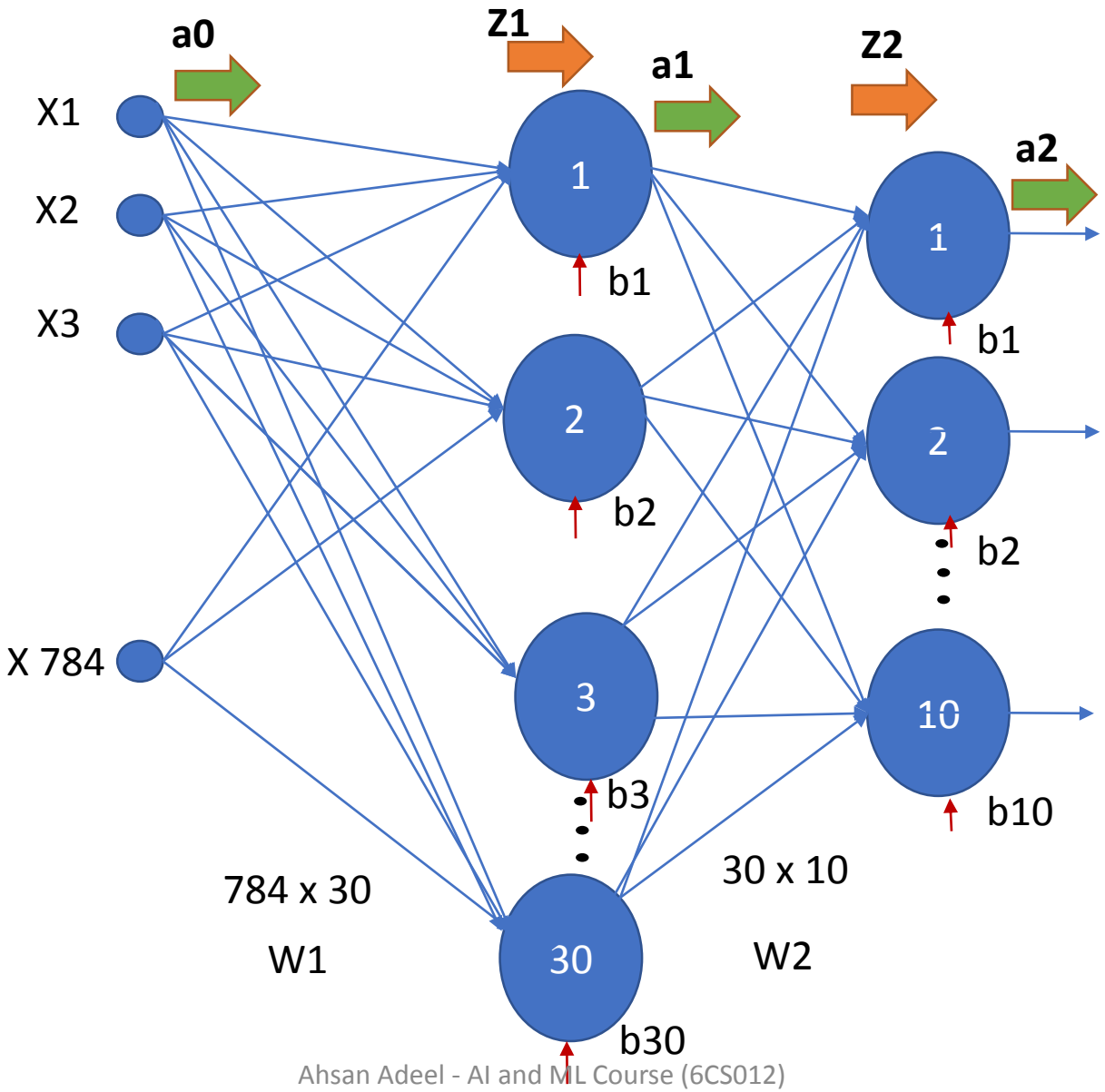
$$Z^l = W^l a^{l-1} + b^l$$

$$a^l = \sigma(Z^l)$$

Let's use multiple neurons to recognize handwritten digits

Forward propagation

- A_0 =?
- Z_1 =?
- A_1 =?
- Z_2 =?
- A_2 =?
- error =



$$Z^l = W^l a^{l-1} + b^l$$

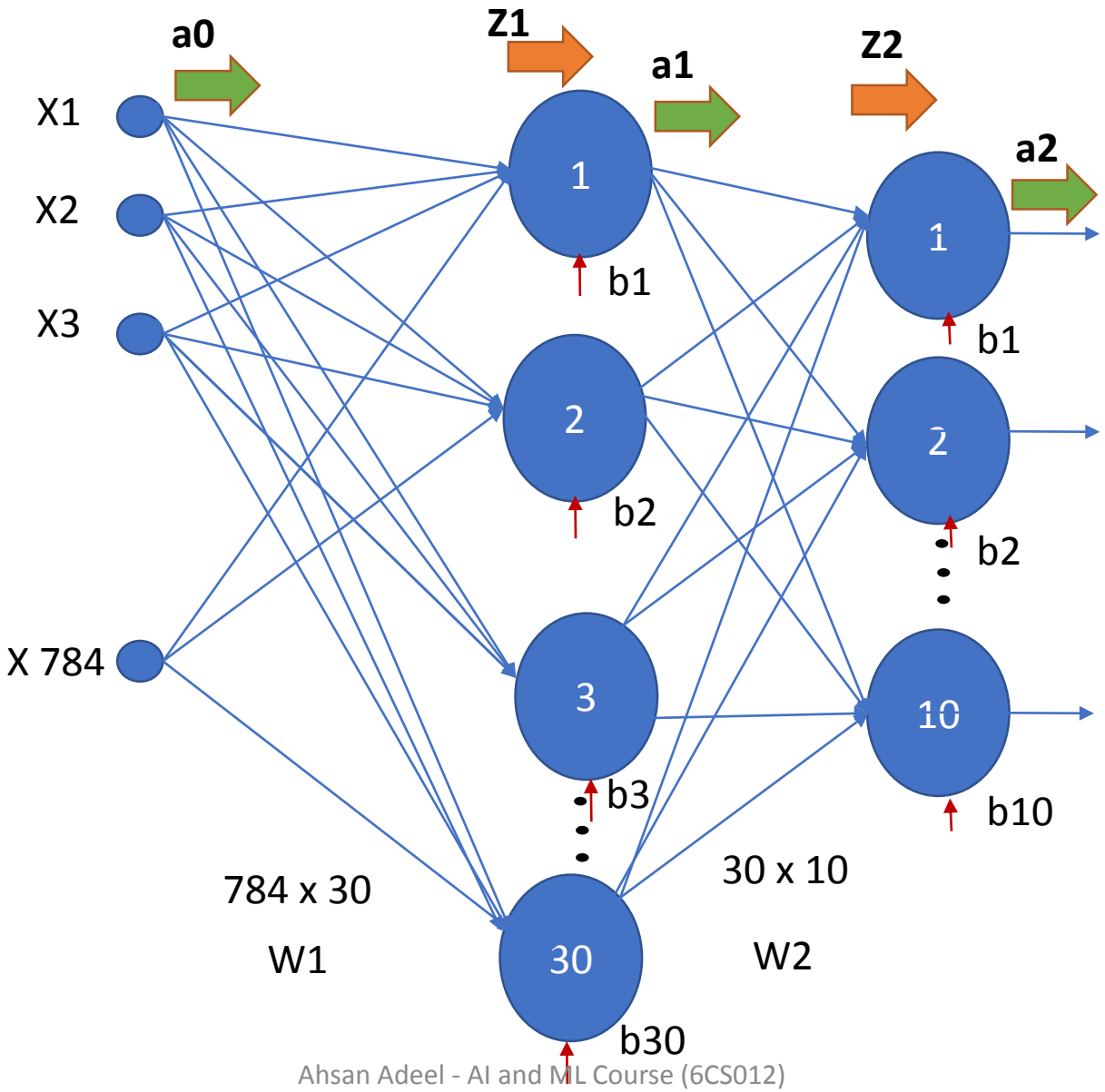
$$a^l = \sigma(Z^l)$$

Let's use multiple neurons to recognize handwritten digits

Backpropagation

Update:

- W1?
- b1?
- W2?
- b2



$$Z^l = W^l a^{l-1} + b^l$$

$$a^l = \sigma(Z^l)$$

Backpropagation

Update:

- W1?
- b1?
- W2?
- b2

W1= W1 - $\boldsymbol{\eta \nabla} \mathrm{J}(\mathrm{W1})$
b1= b1 - $\boldsymbol{\eta \nabla} \mathrm{J}(\mathrm{b1})$
W2= W2 - $\boldsymbol{\eta \nabla} \mathrm{J}(\mathrm{W2})$
b2= b2 - $\boldsymbol{\eta \nabla} \mathrm{J}(\mathrm{b2})$

Backpropagation

Update:

- W1?
- b1?
- W2?
- b2

W1= W1 - $\eta \nabla J(W1)$

b1= b1 - $\eta \nabla J(b1)$

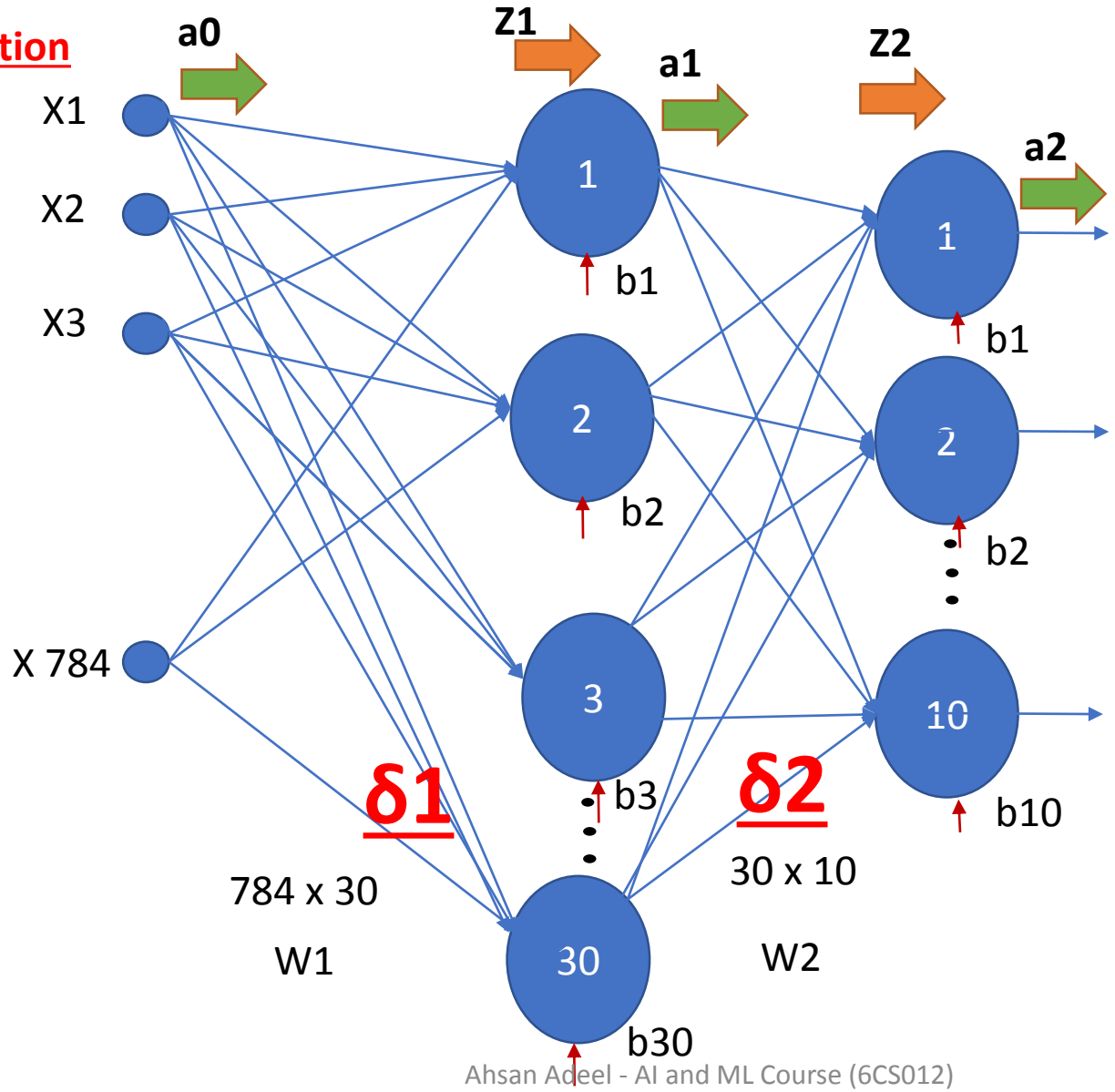
W2= W2 -$\eta \nabla J(W2)$

b2= b2 - $\eta \nabla J(b2)$

How to find these gradients now?

Let's use multiple neurons to recognize handwritten digits

Introduce an error function

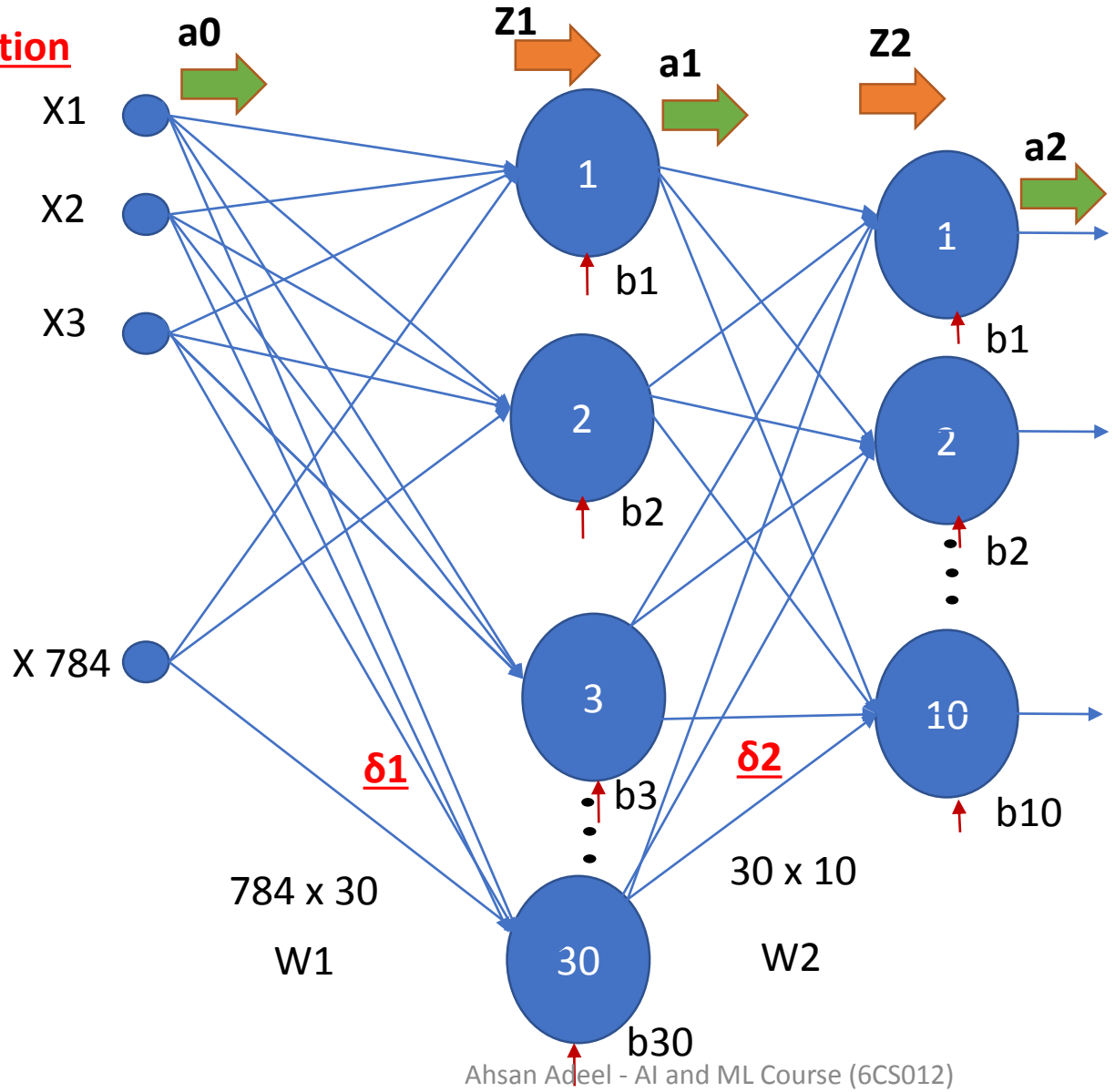


$$Z^l = W^l a^{l-1} + b^l$$

$$a^l = \sigma(Z^l)$$

Let's use multiple neurons to recognize handwritten digits

Introduce an error function



$\delta 3 = a2 - y$

$$Z^l = W^l a^{l-1} + b^l$$

$$a^l = \sigma(Z^l)$$

Four BP equations

Introduce an error function

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{BP1}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{BP2}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{BP3}$$

$$\frac{\partial C}{\partial w^l_{j\,k}} = a^{l-1}_k \delta^l_j \tag{BP4}$$

Let's call last as "layer L" and all others as "layers l"

σ(z) = 1/1+exp(−z)
σ′(z) = σ(z) * (1- σ(z))

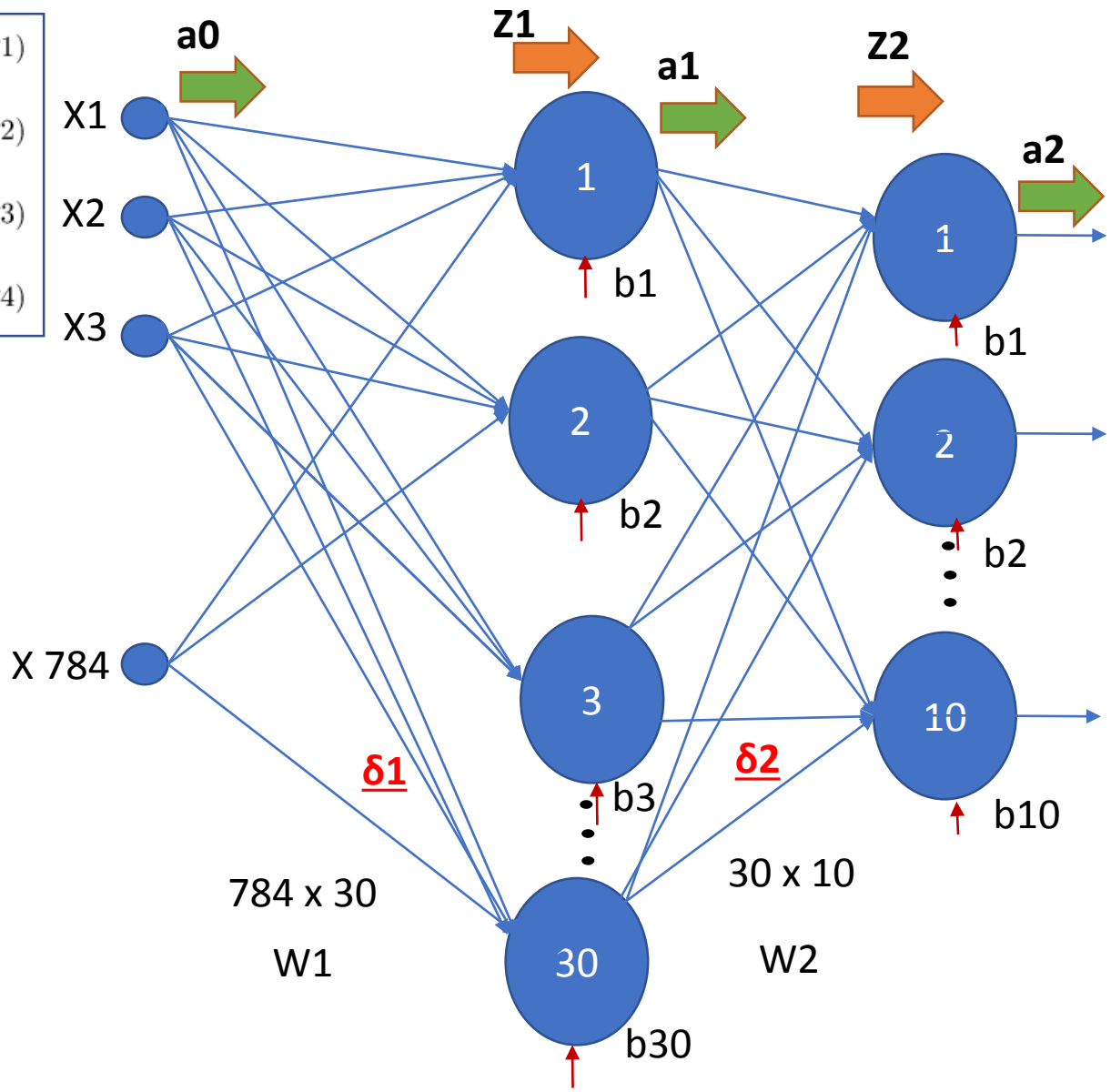
Let's use multiple neurons to recognize handwritten digits

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \quad \text{(BP1)}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \quad \text{(BP2)}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \quad \text{(BP3)}$$

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \quad \text{(BP4)}$$



δ1 = ?
δ2 = ?
δ3 = ?

$\nabla J(W1) = \delta 1 \cdot a0^T$
$\nabla J(W2) = \delta 2 \cdot a1^T$
$\nabla J(b1) = \delta 1$
$\nabla J(b2) = \delta 2$

$W1 = W1 - \eta \nabla J(W1)$
$W2 = W3 - \eta \nabla J(W2)$
$b1 = b1 - \eta \nabla J(b1)$
$b2 = b2 - \eta \nabla J(b2)$

$\delta 3 = a2 - y$

$Z^l = W^l a^{l-1} + b^l$

$a^l = \sigma(Z^l)$

WS6 – Task 1
(MNIST classification using SGD driven ANN

- **Download WS6 folder from Canvas**
- **Use the "MNIST_Data" from WS5 and store it in the same folder**
- **The provided code will extract and pre-process the dataset**
- **The provided code also does the post-processing including thresholding, accuracy etc.**
- **For your ease, following lines are left empty for you to fill.**

Use Python 2.7

```
# ForwardProp

Z_0 = x
A_0 = Z_0
Z_1 =
A_1 =
Z_2 =
A_2 =
error =
```

```
# Backprop

delta3 =
delta_2 =

d_W_2 = # derivative of 'J' w.r.t 'W2'
d_b_2 = # derivative of 'J' w.r.t 'b2'

delta_1 =
d_W_1 = # derivative of 'J' w.r.t 'W1'
d_b_1 = # derivative of 'J' w.r.t 'b1'

W_1 = W_1 – eta *d_W_1
W_2 = W_2 – eta * d_W_2

b_1 = b_1 – eta *d_b_1
b_2 = b_2 – eta *d_b_2
```

```
# ForwardProp – Testing on a test data

A_0 = x_t
A_0 = np.reshape(Z_0, (784, 500))

Z_1 =
A_1 =
Z_2 =
A_2 =
```

WS6 – Task 1
(MNIST classification using SGD driven ANN

1.1. Explain in detail (step by step) the forward and backpropagation algorithm with equations
1.2. Complete the equations and run the code
1.3. Test the model on test data
1.4. Compare results with WS5 Task 2
1.5. Comment on results in detail

References:

- http://deeplearning.stanford.edu/tutorial
- Huff, Trevor, and Scott C. Dulebohn. "Neuroanatomy, Visual Cortex." (2017).
- Ian Goodfellow and Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016, url: http://www.deeplearningbook.org
- https://www.khanacademy.org/math/statistics-probabilit
- 3blue1brown: https://www.3blue1brown.com/
- http://neuralnetworksanddeeplearning.com/index.html