

LING530F: Deep Learning for Natural Language Processing (DL-NLP)

Muhammad Abdul-Mageed

`muhammad.mageed@ubc.ca`

Natural Language Processing Lab

The University of British Columbia

Table of Contents

1 Backpropagation I

- Computing With Gates
- Pre-Activation
- Notation Hell

2 Backpropagation II

Computing With Gates

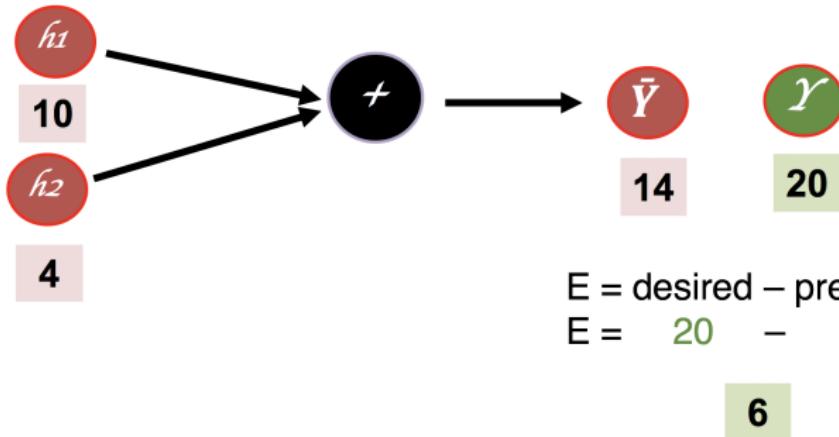


Figure: An add gate operating over units from a hidden layer **I**. The error **E** tells us how far the predicted outcome \hat{y} is from the gold target y . In practice, we use cross-entropy and so **E** here is an oversimplification.

A Change in Input, Causes a Change in Output

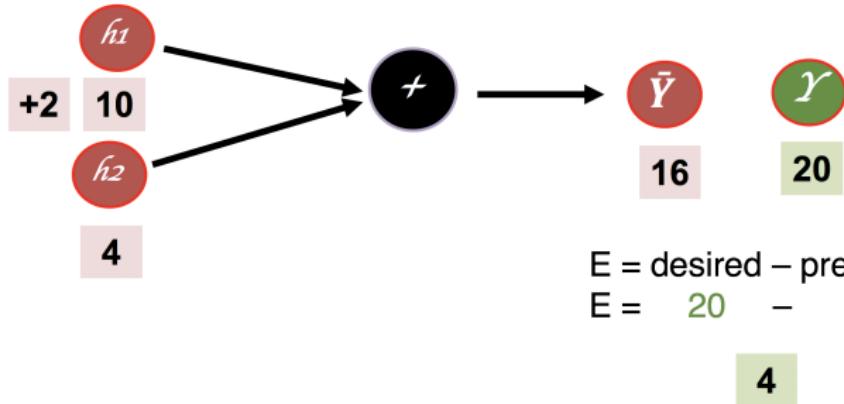
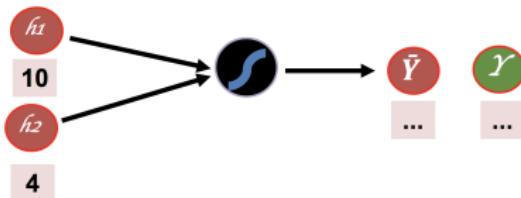


Figure: By increasing the value of h_1 , we can decrease the error E .

Generalizing to Other Functions, Across Layers



Changing Weights, Across Layers

- We can use other functions, like **sigmoid**.
- We change the weights by **identifying the derivative**.
- Since we want to minimize our cost function, we **move in the direction opposite to the derivative**.
- We can **change weights and biases** (i.e., our θ , which constitutes the parameters of the model) **in any layer except input**. (**why?**)

Pre-Activation in One Unit

$\mathbf{W}^{(1)}$ is weight matrix for input-hidden connections.

$$\mathbf{W}^{(1)} = \begin{matrix} w_{11} & w_{12} \end{matrix}$$

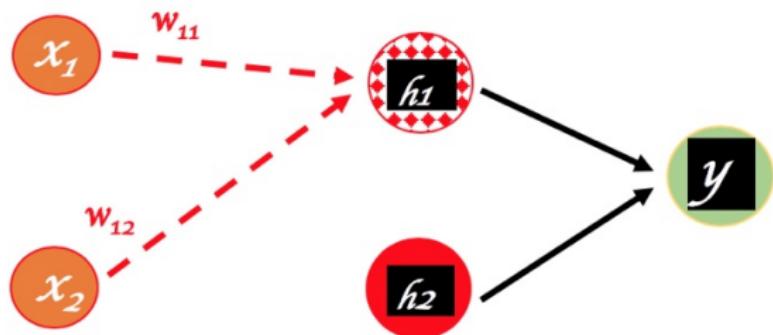


Figure: Connection weights from input to hidden in the 2×2 matrix \mathbf{W} are indexed with i (hidden unit index) and j (input unit index). We refer to the matrix as $\mathbf{W}^{(1)}$ to denote it is the first layer matrix (input-to-hidden). For simplification, bias is dropped.

Pre-Activation in One Unit: More Connection Weights

$\mathbf{W}^{(1)}$ is weight matrix for input-hidden connections.

$$\mathbf{W}^{(1)} = \begin{matrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{matrix}$$

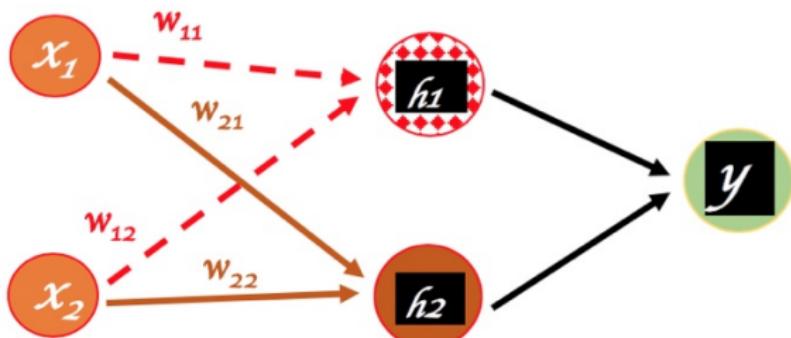


Figure: Connection weights from input to hidden in the 2×2 matrix \mathbf{W} are indexed with i (hidden unit index) and j (input unit index). We refer to the matrix as $\mathbf{W}^{(1)}$ to denote it is the first layer matrix (input-to-hidden).

Two Hidden Layers Network, With One Output

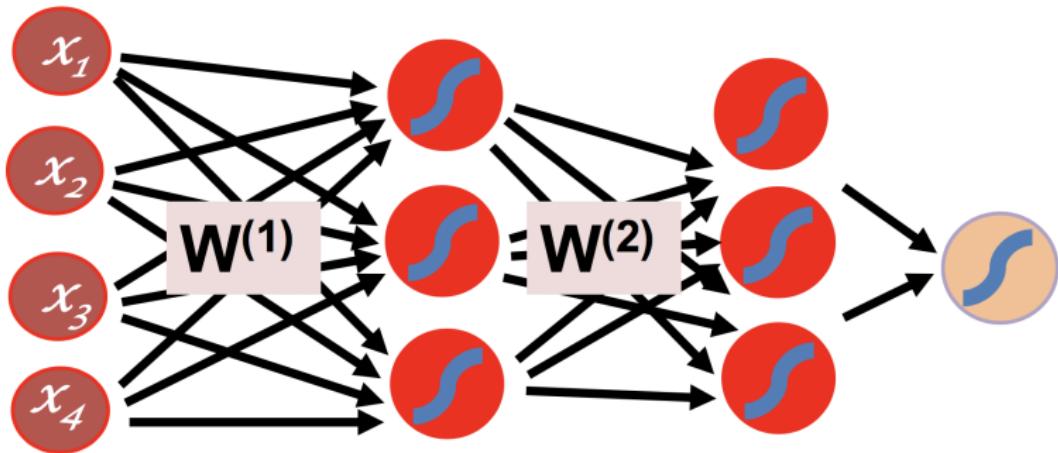


Figure: A network with **two** hidden layers. Now we have **two connection weight matrixes** $W^{(1)}$ and $W^{(2)}$. The network has only one output unit (e.g., performing binary classification).

Two Hidden Layers Network, With Multiple Outputs

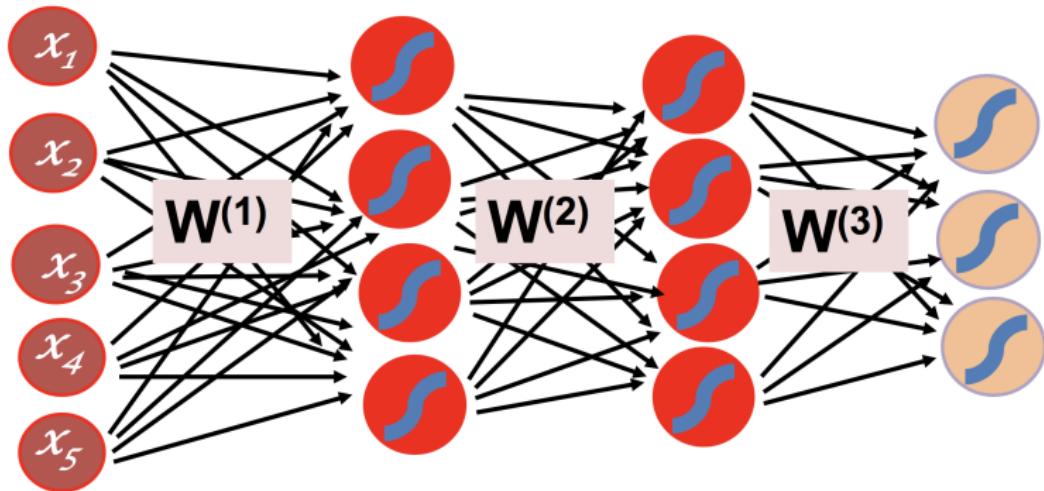


Figure: A network with **two** hidden layers. Now we have **three connection weight matrixes** $W^{(1)}, W^{(2)}$, and $W^{(3)}$. The network has multiple output units (e.g., performing multi-class classification).

Indexing Hidden Layers

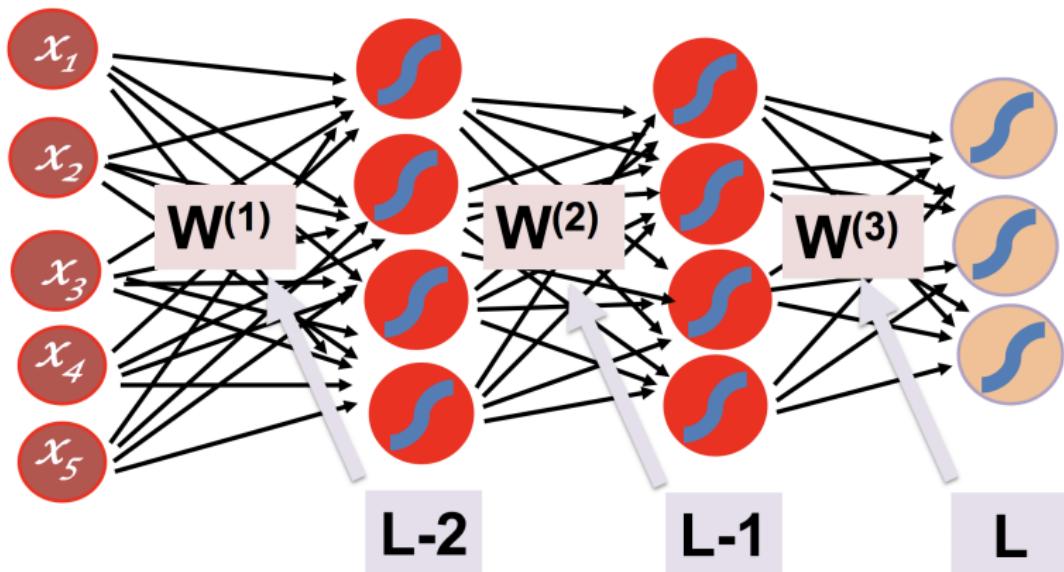


Figure: We also index our **hidden layers** as $\ell = 1, 2, L-1, \dots, L$.

Re-Indexing Connection Weights With Hidden Layer Labels

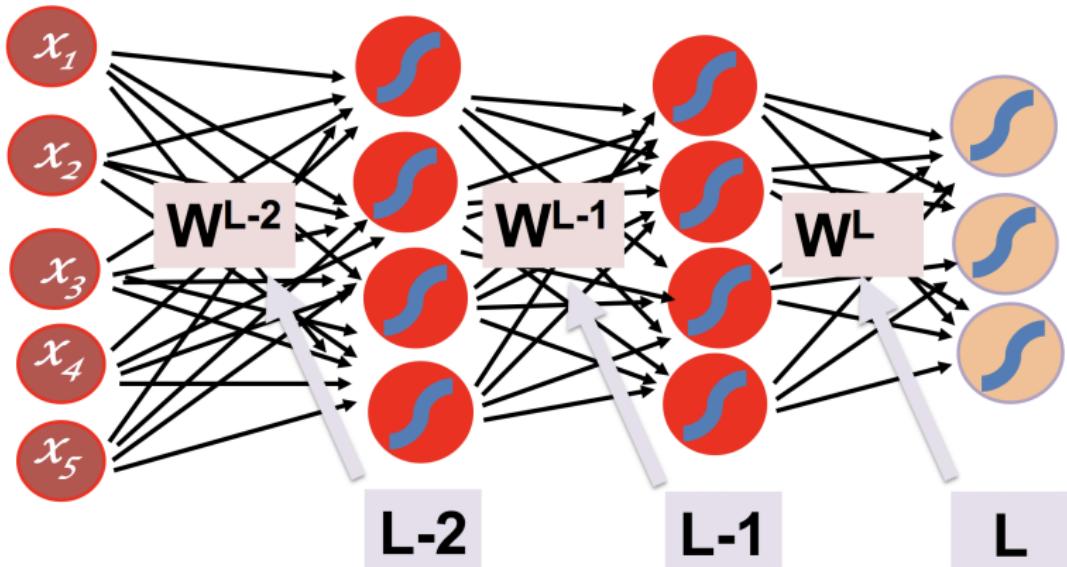


Figure: We re-index our **connection weights with hidden layer labels**, to simplify.

Recall ji Indexing

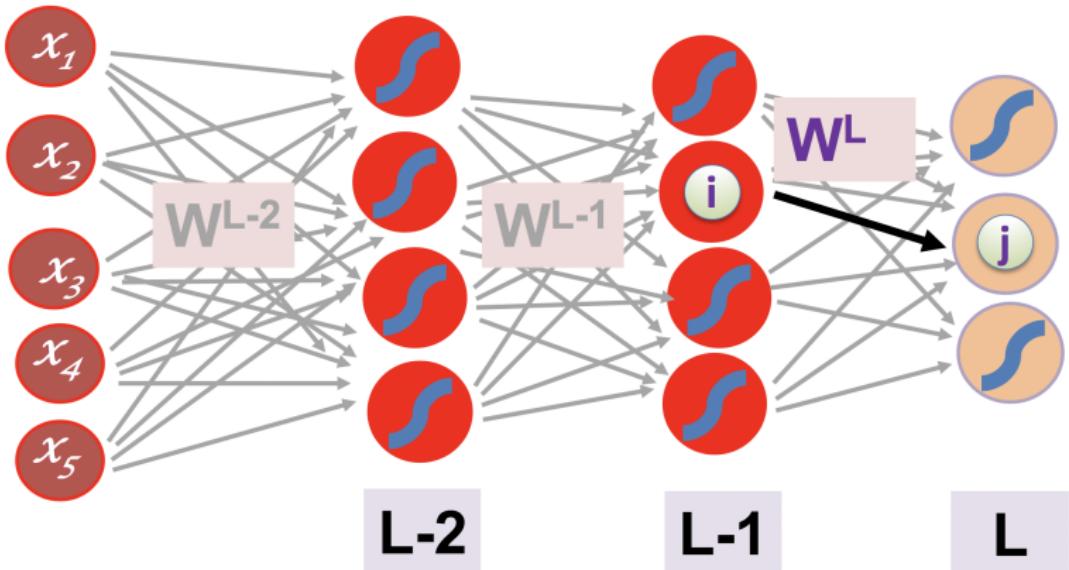


Figure: We re-index our **connection weights with hidden layer labels**, to simplify.

Node Connections (From Connection Weights Matrix W)

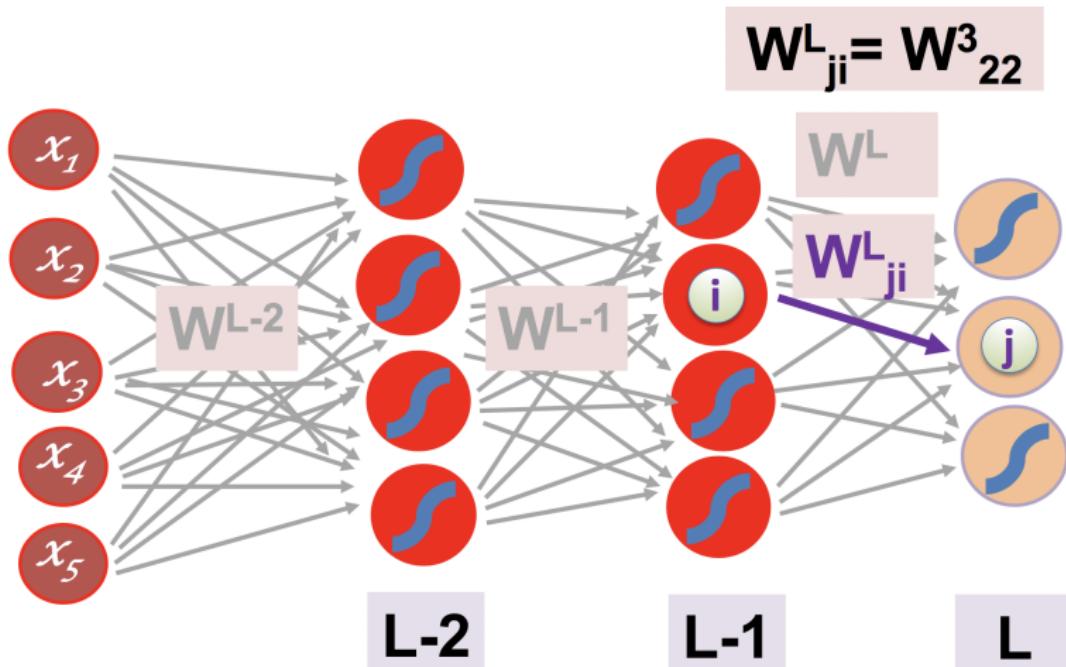


Figure: Weight of the connection from **node i** (in layer L-1) to **node j** (in layer L).

Node Connections Contd.

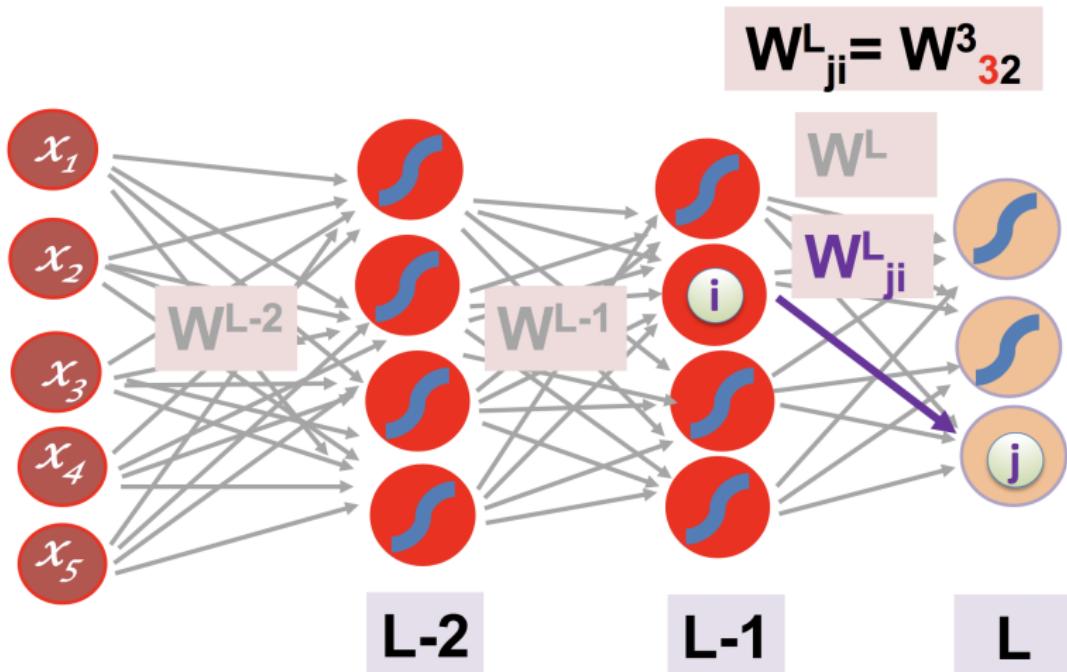


Figure: Weight of the connection from **node $i=2$** (in layer L-1 → **not indexed**) to **node $j=3$** (in layer L → **indexed in superscript**).

The Vector of Weights $W_{j,:}^L$

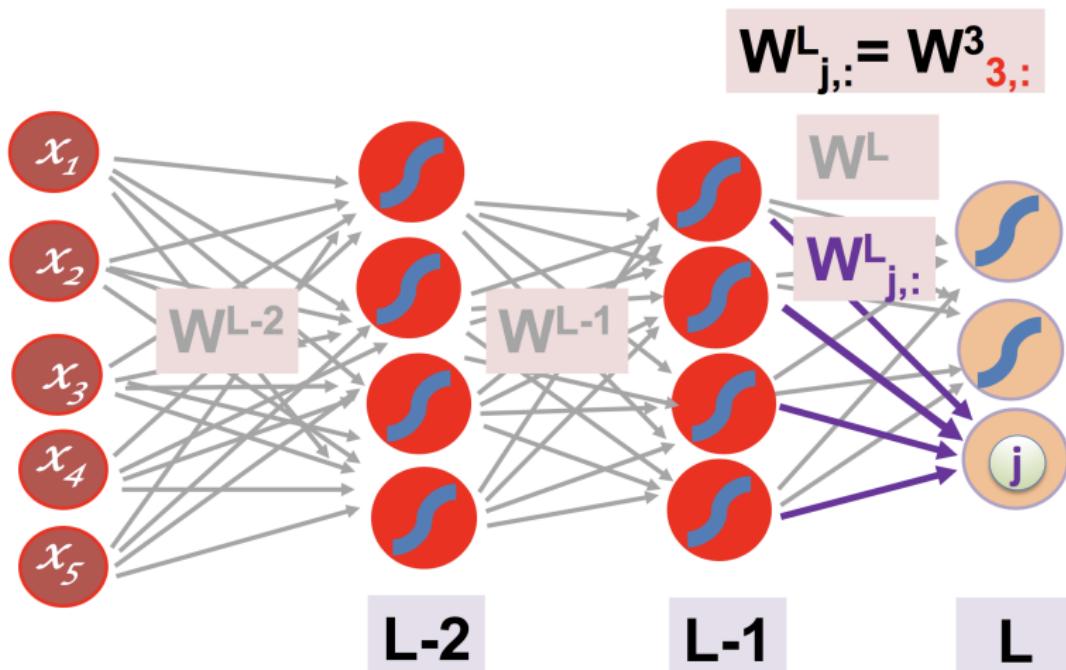


Figure: **Vector of weights** from **each node in layer L-1** to **node j in Layer L**.

The Vector of Inputs $Z_{j,:}^L$

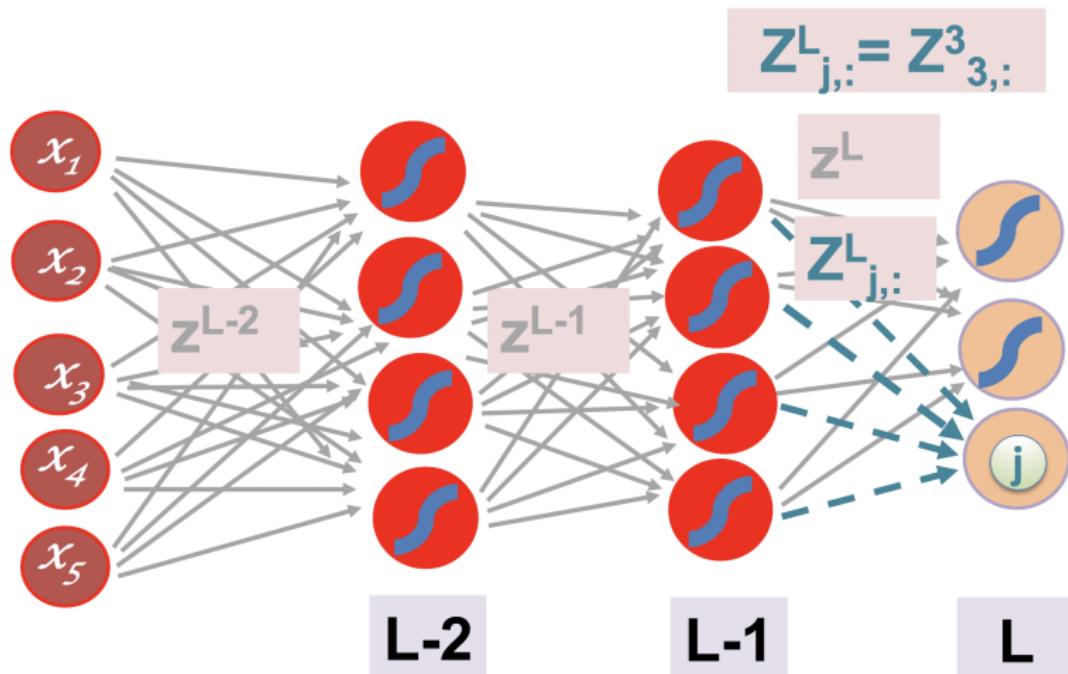


Figure: Vector of inputs from each node in layer L-1 to node j in Layer L.

The Vector of Activations a_L^{L-1}

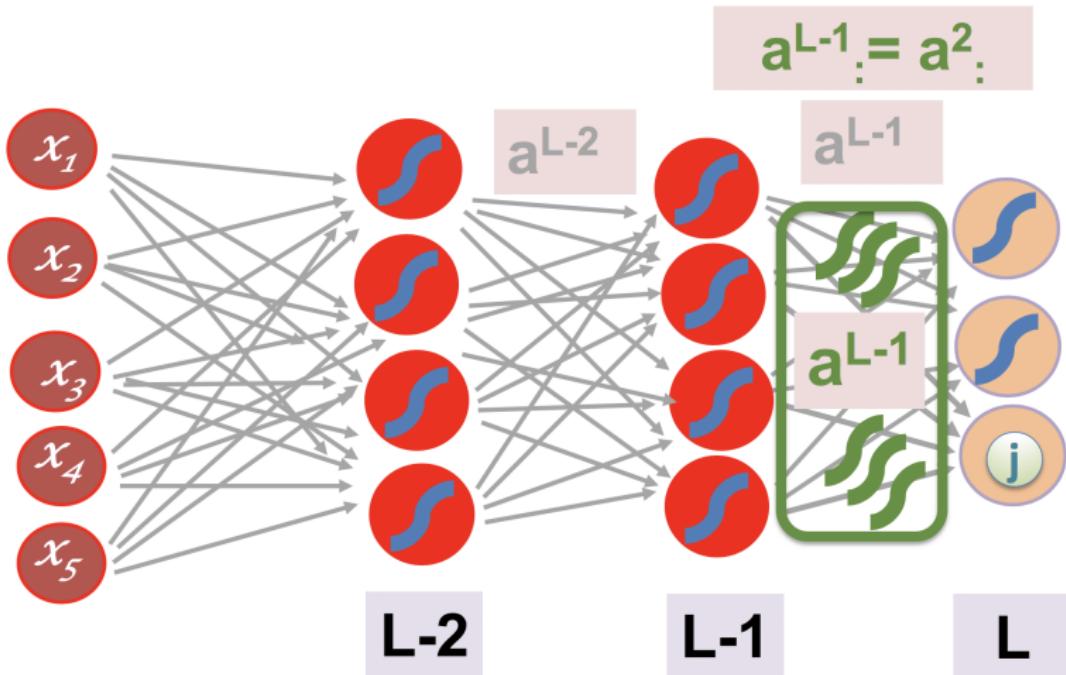


Figure: **Vector of activations** from **each node** in layer L-1 to **node j** in Layer L.

What's in a Cell?

$$z_j^I = \sum_{i=0}^{n-1} w_{ji}^I a_i^{I-1}$$

$$a_j^I = g^I(z_j^I)$$

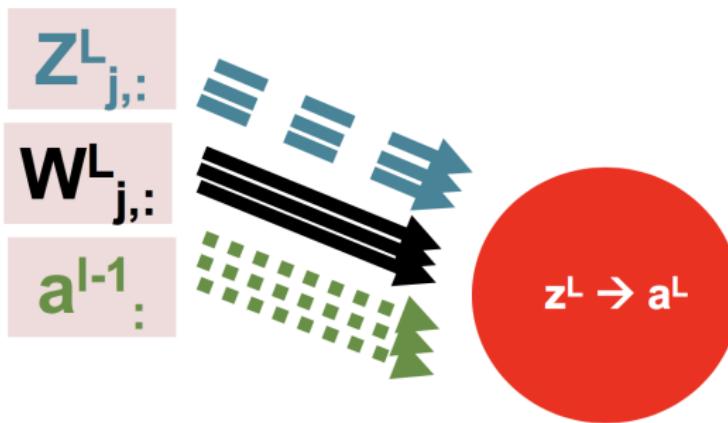


Figure: Processing in a cell.

The Loss C

- For a single data point, a cost C can be expressed as squared difference, summing over each node j in output layer L

1: Cost in each activation node j at Layer L

$$C = \sum_{j=0}^{n-1} (a_j^L - y_j)^2$$

Cross-Entropy

- Note:** In deep learning, we usually use cross-entropy as the loss function (see here), but for simplicity we use the squared difference here. This does not change the way backpropagation works.

Input For Node j in layer l

- Input for node j in layer l is the weighted sum of the activation outputs from the previous layer $\ell - 1$.
- An individual term is as follows:

2: Individual term of sum of input for node j in layer $\ell - 1$

$$w_{ji}^l a_i^{l-1}$$

- Input sum:

3: Input for node j in layer $\ell - 1$ summed

$$z_j^l = \sum_{i=0}^{n-1} w_{ji}^l a_i^{l-1}$$

Activation Output

- Activation output of a given node j in layer l is the result of passing z_j^l to an activation function g (e.g., sigmoid).

4: Individual term of sum of input for node j in layer $\ell - 1$

$$a_j^\ell = g^\ell(z_j^\ell)$$

Cost C as Composition of Functions

- We can express cost as a function of activation output a .

5: Cost as a Function of a

$$a_j^l = g^l(z_j^l)$$

$$C_j = (a_j^l - y_j)^2$$

- Total loss for a single input is a sum over all losses of all output nodes:

6: Cost as a Function of a

$$C = \sum_{j=0}^{n-1} C_j$$

Changing Weights, Across Layers

- Loss is a composition of functions and will use the chain rule of calculus to differentiate this function of functions:
- This means we will take the derivative of the loss with respect to θ (the weights and biases).

7: Cost as a Function of a

$$C_j = (a_j^l(g^l(z_j^l)) - y_j)^2$$

P.S. y_j is a constant parameter that we only use to define the cost function, and so C can be thought of as just a function of output a .

Derivative of Loss C With Respect to Weight W_{32}^2

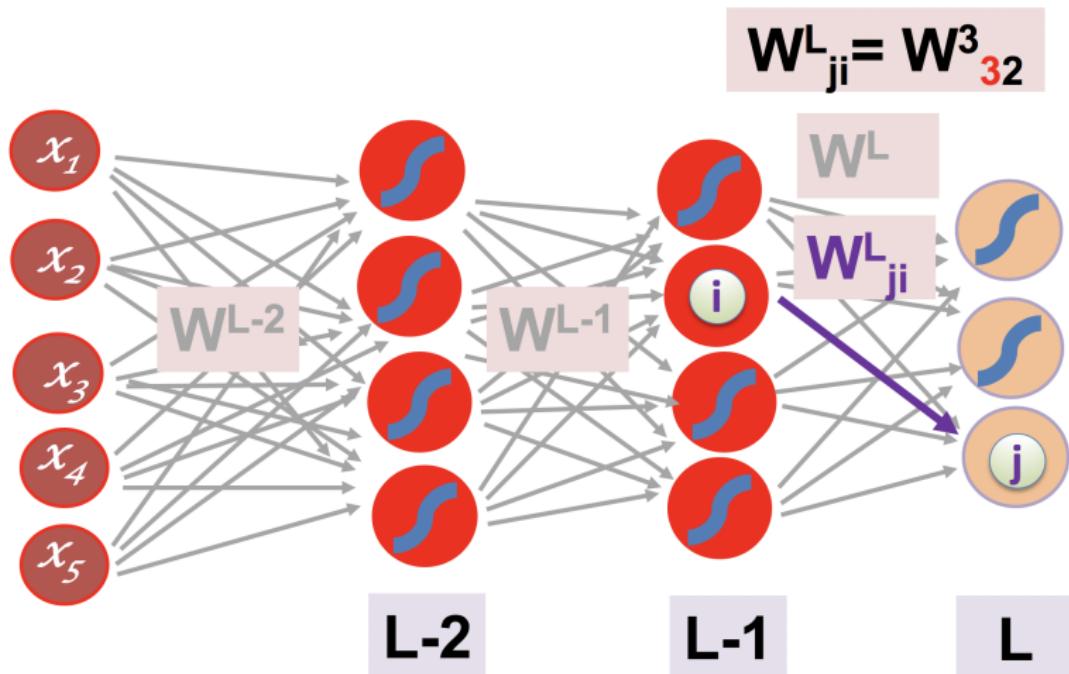
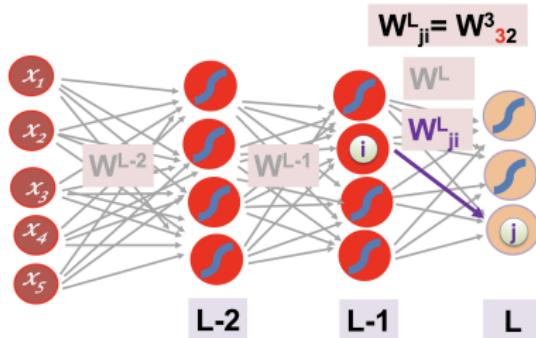


Figure: Derivative of C with respect to W_{32}^3 is denoted as $\frac{\partial C}{\partial W_{32}^3}$.

Derivative of Loss C With Respect to Weight W_{32}^3 Contd



- Derivative of C with respect to W_{32}^3 is denoted as $\frac{\partial C}{\partial W_{32}^3}$
- Loss C is a function of output activation a ,** which in turn is a function of input z , which itself is a function of weights w (all indexes dropped for simplification).

Putting it All Together

8: The C, a, and z functions

z is a function of **w**:

$$z_j^l = \sum_{j=0}^{n-1} w_{ji}^l a_i^{l-1}$$

a is a function of **z**:

$$a_j^l = g^l(z_j^l)$$

C is a function of **a**:

$$C_j = a_j^l(g^l(z_j^l))$$

We take the product of the derivatives of compositional function C:

$$\frac{\partial C}{\partial W_{32}^3} = \left(\frac{\partial C}{\partial a_3^3} \right) \left(\frac{\partial a_3^3}{\partial z_3^3} \right) \left(\frac{\partial z_3^3}{\partial W_{32}^3} \right)$$