Machine Learning and Intelligent Systems

Neural Networks

Maria A. Zuluaga

Jan. 29, 2024

EURECOM - Data Science Department

Table of contents

History & Motivation

A Note on History

- The term neural network has its origins in attempts to find mathematical representations of information processing in biological systems
- **From Bishop:** it has been used very broadly to cover a wide range of different models, many of which have been the subject of exaggerated claims regarding their biological plausibility.

A Note on History

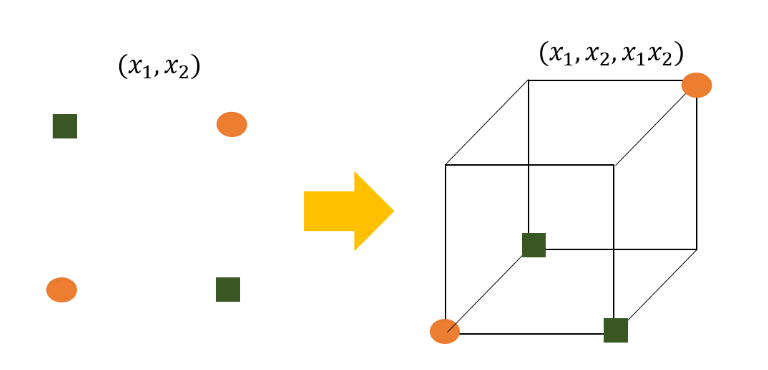
- The term neural network has its origins in attempts to find mathematical representations of information processing in biological systems
- **From Bishop:** it has been used very broadly to cover a wide range of different models, many of which have been the subject of exaggerated claims regarding their biological plausibility.
- They are nonlinear efficient models for statistical pattern recognition

Motivation: The XOR

- The work from Minsky and Papert on the limitations of the XOR had a devastating effect in AI, leading to what was called the AI Winter
- It is surprising that their work had such fatal consequences when the solution is quite simple

Motivation: The XOR

- The work from Minsky and Papert on the limitations of the XOR had a devastating effect in AI, leading to what was called the AI Winter
- It is surprising that their work had such fatal consequences when the solution is quite simple

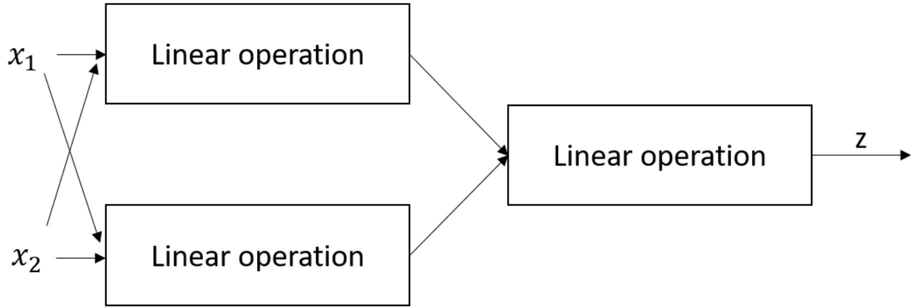


Motivation

A more powerful solution to it, could be the combination of linear classifiers, where the output of one, is the input to another classifier:

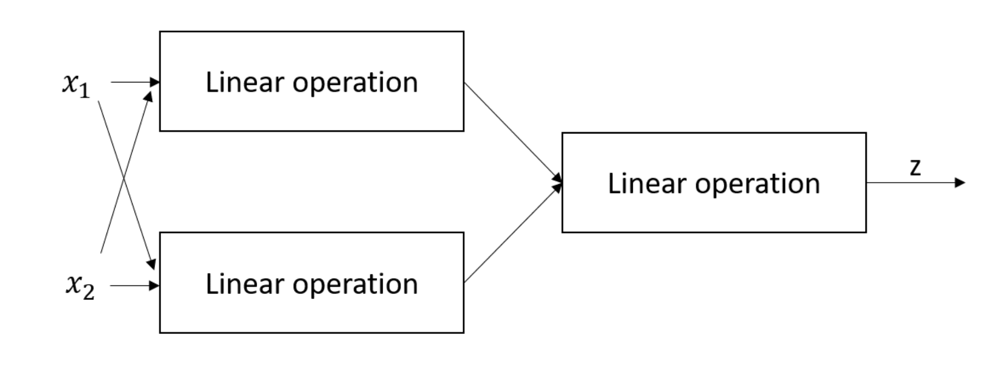
Motivation

A more powerful solution to it, could be the combination of linear classifiers, where the output of one, is the input to another classifier:



Motivation

A more powerful solution to it, could be the combination of linear classifiers, where the output of one, is the input to another classifier:



Question: Could you solve the XOR problem using 1's and 0's?

Motivation

- The combination of linear operators results in another linear operation.
- We need to add some non-linearities.

Motivation

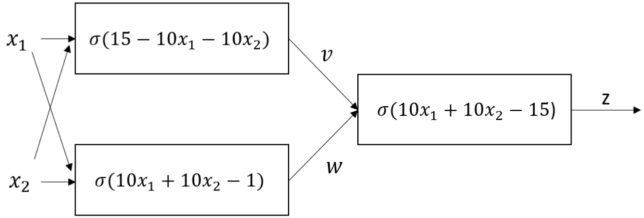
- The combination of linear operators results in another linear operation.
- We need to add some non-linearities.
- **Example:** The sigmoid function from lecture 3.

Motivation

- The combination of linear operators results in another linear operation.
- We need to add some non-linearities.
- **Example:** The sigmoid function from lecture 3.
- Using $\sigma(\cdot)$, we can build a two-layer perceptron that solves the XOR:

Motivation

- The combination of linear operators results in another linear operation.
- We need to add some non-linearities.
- **Example:** The sigmoid function from lecture 3.
- Using $\sigma(\cdot)$, we can build a two-layer perceptron that solves the XOR:



X1	X2	V	W	Z
0	0			
0	1			
1	0			
1	1			

Neural Nets

- Each of the boxes from the previous example can be denoted a **neuron**
- The neuron runs a linear operation, followed by a non-linear function on the linear operation's output.
- The sigmoid is the traditional non-linear function to use but, there are many others.

Neural Nets

- Each of the boxes from the previous example can be denoted a **neuron**
- The neuron runs a linear operation, followed by a non-linear function on the linear operation's output.
- The sigmoid is the traditional non-linear function to use but, there are many others.
- Interesting aspects of the sigmoid:
  - It is bounded so, it does not over-saturates other networks
  - Smooth
  - Well-defined gradients and Hessians

Let's formalize things...

Neural Nets: Structure

Definitions

For a 1 hidden layer network

- **Input layer:** $\mathbf{x}_1, \ldots, \mathbf{x}_D; \quad \mathbf{x}_0 = 1$
- **Hidden units:** $h_1, \ldots, h_M; \quad h_0 = 1$
- **Output layer:** $z_1, \ldots, z_k$

Definitions

For a 1 hidden layer network

- **Input layer:** $\mathbf{x}_1, \dots, \mathbf{x}_D; \quad \mathbf{x}_0 = 1$
- **Hidden units:** $h_1, \dots, h_M; \quad h_0 = 1$
- **Output layer:** $z_1, \dots, z_k$

The set of weights:

- **Layer 1 weights:** $M \times (D+1)$ matrix $\mathbf{V}$, with $\mathbf{V}_i^T$ the i^{th} row
- **Layer 2 weights:** $k \times (M+1)$ matrix $\mathbf{W}$, with $\mathbf{W}_i^T$ the i^{th} row

Definitions

For a 1 hidden layer network

- **Input layer:** $\mathbf{x}_1, \ldots, \mathbf{x}_D; \quad \mathbf{x}_0 = 1$
- **Hidden units:** $h_1, \ldots, h_M; \quad h_0 = 1$
- **Output layer:** $z_1, \ldots, z_k$

The set of weights:

- **Layer 1 weights:** $M \times (D+1)$ matrix $\mathbf{V}$, with $\mathbf{V}_i^T$ the i^{th} row
- **Layer 2 weights:** $k \times (M+1)$ matrix $\mathbf{W}$, with $\mathbf{W}_i^T$ the i^{th} row

The activation function:

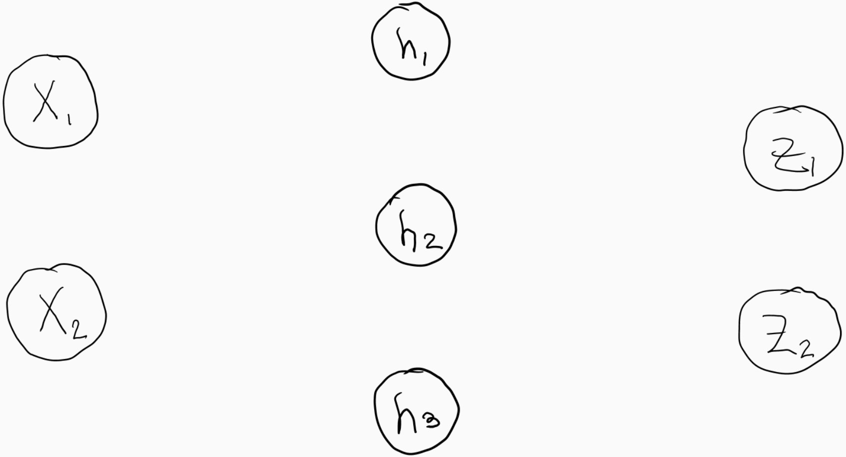
- **The sigmoid:** $\sigma(a) = \frac{1}{1+\exp(-a)}$
- For a vector $\mathbf{V}$, $\sigma(\mathbf{V}) = [\sigma(v_1), \sigma(v_2), \ldots]^T$, the sigmoid is applied component-wise

Example: 1 hidden layer network

Let's build a 1 hidden layer neural network with two inputs, three hidden units and two outputs

Example: 1 hidden layer network

Let's build a 1 hidden layer neural network with two inputs, three hidden units and two outputs

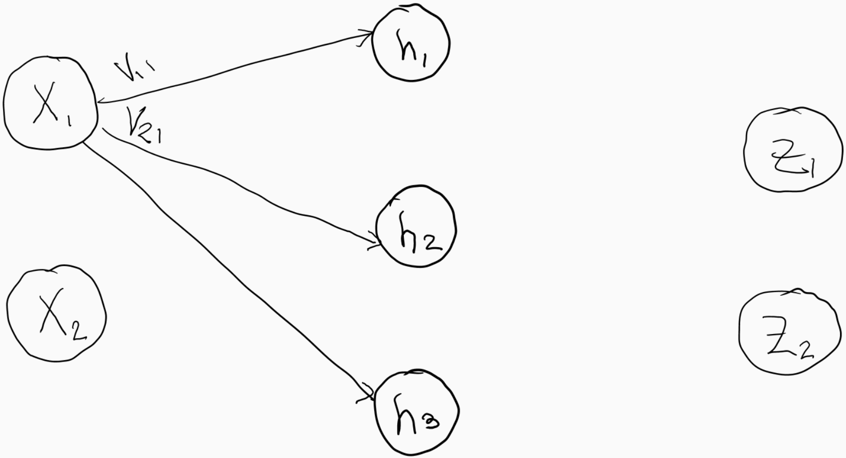


Example: Connecting the input to the hidden layer

We will use $\mathbf{V}$ as previously defined.

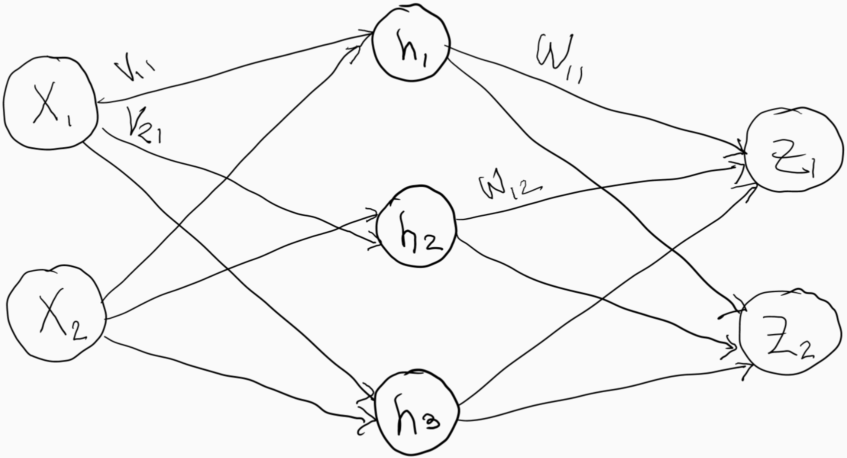
Example: Connecting the input to the hidden layer

We will use **V** as previously defined.

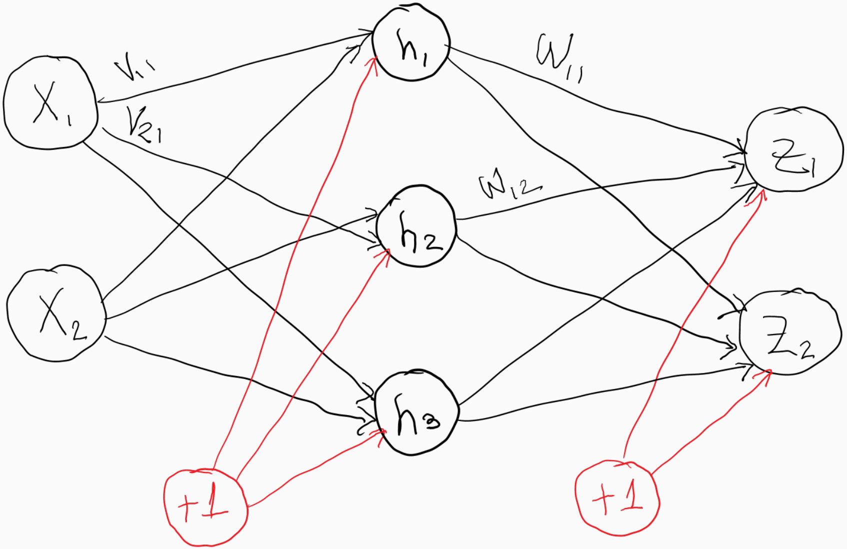


Example: All connections

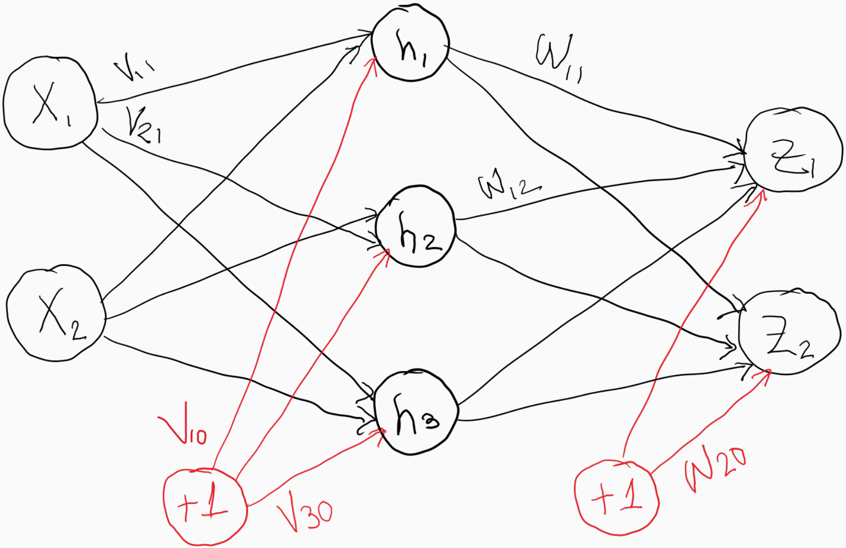
Let's also identify the elements of **W**



Example: The bias terms



Example: Complete network



Inputs and Outputs

The output of the hidden layer can be expressed as:

$$h = \sigma\left(\mathbf{V}\mathbf{x}\right) \tag{1}$$

Inputs and Outputs

The output of the hidden layer can be expressed as:

$$h = \sigma\left(\mathbf{V}\mathbf{x}\right) \tag{1}$$

which will then act as a weighted input of the output layer:

$$z = \sigma\left(\mathbf{W}h\right) = \sigma\left(\mathbf{W}\sigma(\mathbf{V}\mathbf{x})\right) \tag{2}$$

Inputs and Outputs

The output of the hidden layer can be expressed as:

$$h = \sigma\left(\mathbf{V}\mathbf{x}\right) \tag{1}$$

which will then act as a weighted input of the output layer:

$$z = \sigma\left(\mathbf{W}h\right) = \sigma\left(\mathbf{W}\sigma(\mathbf{V}\mathbf{x})\right) \tag{2}$$

Equivalently, these can be expressed element-wise. For Eq. 1:

$$h_i = \sigma\left(\sum_{j=0}^{D} V_{ij}x_j\right), \qquad \text{with } x_0 = 1$$

Inputs and Outputs

The output of the hidden layer can be expressed as:

$$h = \sigma\left(\mathbf{V}\mathbf{x}\right) \tag{1}$$

which will then act as a weighted input of the output layer:

$$z = \sigma\left(\mathbf{W}h\right) = \sigma\left(\mathbf{W}\sigma(\mathbf{V}\mathbf{x})\right) \tag{2}$$

Equivalently, these can be expressed element-wise. For Eq. 1:

$$h_i = \sigma\left(\sum_{j=0}^{D} V_{ij} x_j\right), \qquad \text{with } x_0 = 1$$

and Eq. 2

$$z_k = \sigma\left(\sum_{i=0}^{M} W_{ki} h_i\right), \qquad \text{with } h_0 = 1$$

Final Remarks

- This examples represents the simplest neural network possible as it contains only one hidden layer.
- It is commonly known as a **feed-forward network** or the **multilayer perceptron**
- In practical/real situations, it will be more common to have several hidden layers.
- In such case, it might be more convenient to drop the proposed notation for the weight matrices, $\mathbf{V}$, $\mathbf{W}$ and use super-indices:

$$
\begin{aligned}
\mathbf{V} &\rightarrow \mathbf{W}^{(1)} \\
\mathbf{W} &\rightarrow \mathbf{W}^{(2)} \\
\dots & \\
\dots &\rightarrow \mathbf{W}^{(L)}
\end{aligned}
$$

Final Remarks

The unification of notation for the weights allows to have a more compact expression for the network:

$$\hat{y}_k(\mathbf{x}, \mathbf{W}) = \sigma\left(\sum_{i=0}^{M} W_{ki}^{(2)} \sigma\left(\sum_{j=0}^{D} W_{ij}^{(1)} x_j\right)\right)$$

Final Remarks

The unification of notation for the weights allows to have a more compact expression for the network:

$$\hat{y}_k(\mathbf{x}, \mathbf{W}) = \sigma\left(\sum_{i=0}^{M} W_{ki}^{(2)} \sigma\left(\sum_{j=0}^{D} W_{ij}^{(1)} x_j\right)\right)$$

$$\hat{y}(\mathbf{x}, \mathbf{W}) = f\left(\sum_{i=0}^{M} \mathbf{W}_i \phi_i(\mathbf{x})\right)$$

Final Remarks

The unification of notation for the weights allows to have a more compact expression for the network:

$$\hat{y}_k(\mathbf{x}, \mathbf{W}) = \sigma\left(\sum_{i=0}^{M} W_{ki}^{(2)} \sigma\left(\sum_{j=0}^{D} W_{ij}^{(1)} x_j\right)\right)$$

$$\hat{y}(\mathbf{x}, \mathbf{W}) = f\left(\sum_{i=0}^{M} \mathbf{W}_i \phi_i(\mathbf{x})\right)$$

where we use f to denote any non-linear function.

Does this expression look familiar?

Features and Neural Nets

- A neuron can be seen as a feature map of the form

$$\phi_i(\mathbf{x}) = \left(\sum_{j=0}^{D} W_{ij} x_j \right)$$

- Therefore, each node in the network can be interpreted as a feature variable

Features and Neural Nets

- A neuron can be seen as a feature map of the form

$$\phi_i(\mathbf{x}) = \left(\sum_{j=0}^{D} W_{ij} x_j \right)$$

- Therefore, each node in the network can be interpreted as a feature variable
- By optimizing the weights $\{\mathbf{W}\}$ we are doing feature selection

Features and Neural Nets

- A neuron can be seen as a feature map of the form

$$\phi_i(\mathbf{x}) = \left(\sum_{j=0}^{D} W_{ij} x_j \right)$$

- Therefore, each node in the network can be interpreted as a feature variable
- By optimizing the weights $\{\mathbf{W}\}$ we are doing feature selection

- **Pre-trained networks:** Resulting features from optimization useful to many problems
- Need of a lot of data

Learning Process

The Learning process v1.0: The inefficient way

1. Pick a loss function:

$$L(z, y) = \frac{1}{2}\|z - y\|^2$$

The Learning process v1.0: The inefficient way

1. Pick a loss function:

$$L(z, y) = \frac{1}{2}\|z - y\|^2$$

2. Use batch or stochastic gradient descent to find the set of parameters $\hat{\mathbf{V}}, \hat{\mathbf{W}}$ that minimize the cost function J

$$J = \frac{1}{N}\sum_{i}^{N} L(\hat{y}_i, y_i)$$

The Learning process v1.0: The inefficient way

1. Pick a loss function:

$$L(z, y) = \frac{1}{2}\|z - y\|^2$$

2. Use batch or stochastic gradient descent to find the set of parameters $\hat{\mathbf{V}}, \hat{\mathbf{W}}$ that minimize the cost function J

$$J = \frac{1}{N}\sum_{i}^{N} L(\hat{y}_i, y_i)$$

3. With $\mathbf{w}$ denoting a vector containing all weights $\mathbf{V}, \mathbf{W}$, apply gradient descent:

$$\mathbf{w} \leftarrow \mathrm{RANDOM}()$$

repeat

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla J(\mathbf{w})$$

The Learning process v1.0: The inefficient way

Obtaining one derivative for each weight takes time linear in the number of neurons in the neural network to compute a derivative for one weight. Multiply that by the number of weights.

$$\mathcal{O}(nodes^2)$$

The Learning process v1.0: The inefficient way

Obtaining one derivative for each weight takes time linear in the number of neurons in the neural network to compute a derivative for one weight. Multiply that by the number of weights.

$$\mathcal{O}(nodes^2)$$

This is not scalable!

The Learning process v1.0: The inefficient way

Obtaining one derivative for each weight takes time linear in the number of neurons in the neural network to compute a derivative for one weight. Multiply that by the number of weights.

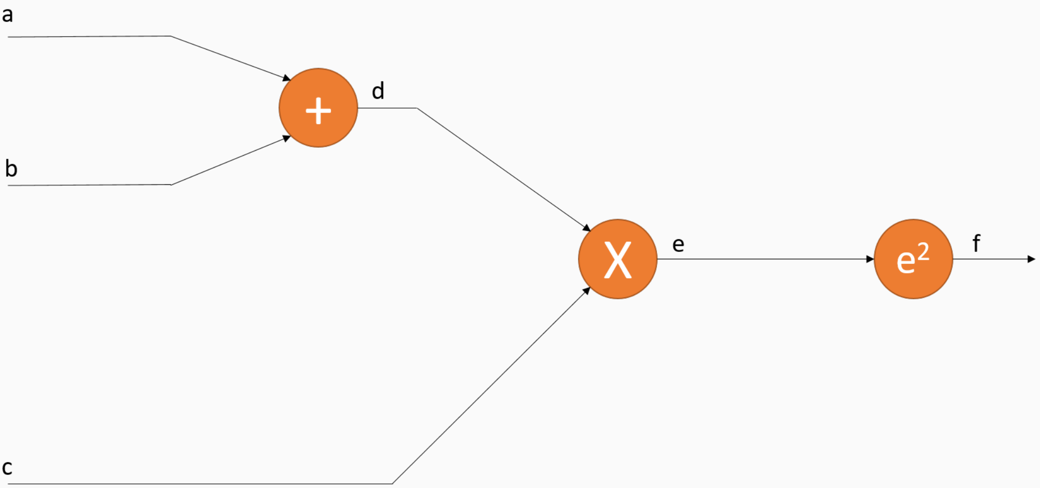
$$\mathcal{O}(nodes^2)$$

This is not scalable!

We will now focus on the backpropagation algorithm, which allows computation of the gradient in $\mathcal{O}(nodes)$ time.

Backpropagation Algorithm

Gradient over a Graph of Arithmetic Operations

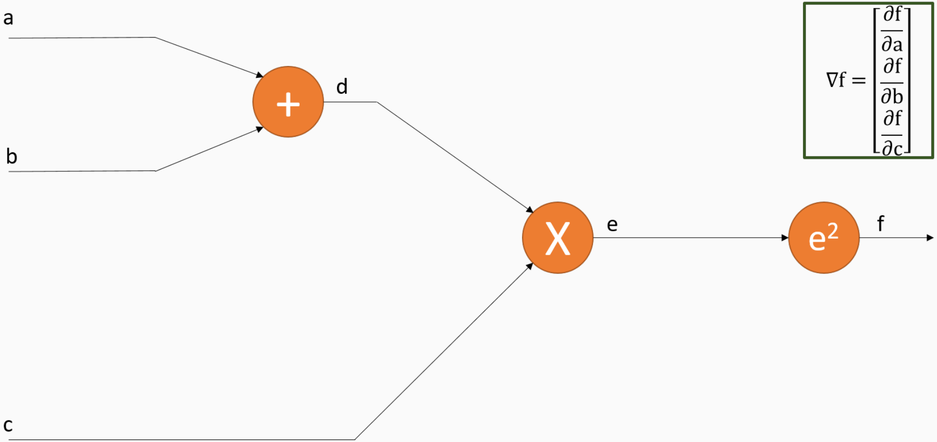


Can we estimate f?

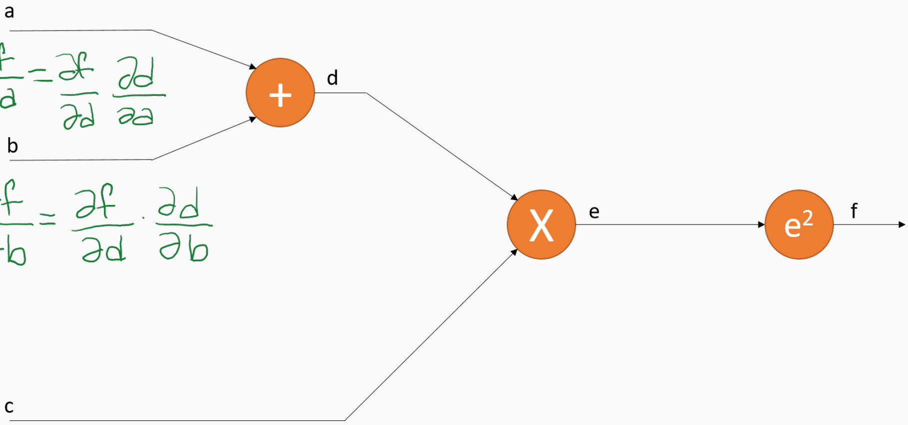
Gradient over a Graph of Arithmetic Operations

Can we estimate ∇f? $\rightarrow$ **Yes, we will use the chain rule**

$$\boxed{\nabla f = \begin{bmatrix} \frac{\partial f}{\partial a} \\ \frac{\partial f}{\partial b} \\ \frac{\partial f}{\partial c} \end{bmatrix}}$$



Step 1: Chain rule

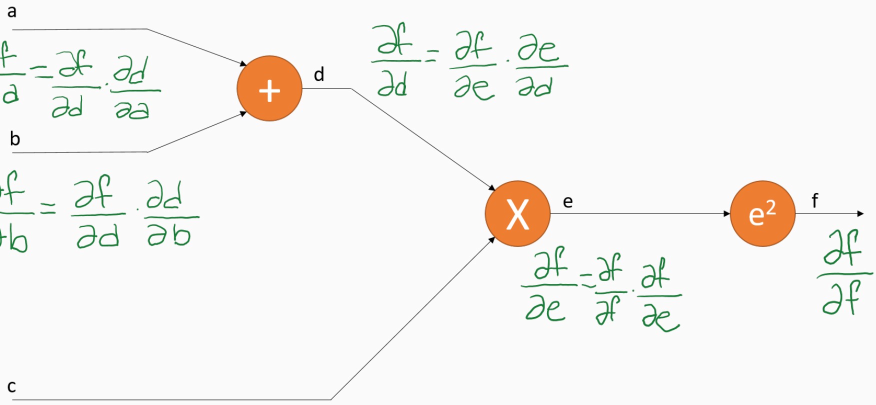


$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial d} \frac{\partial d}{\partial a}$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial b}$$

$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial c}$$

Step 2: Forward pass chain rule along all nodes



$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial a}$$

$$\frac{\partial f}{\partial d} = \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial d}$$

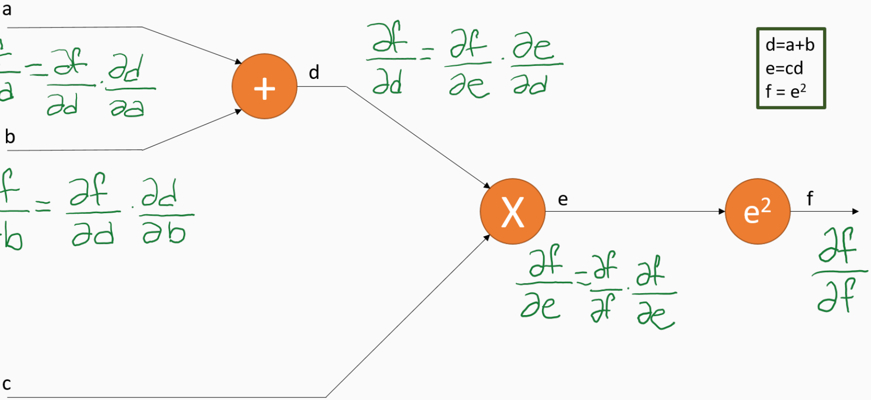
$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial b}$$

$$\frac{\partial f}{\partial e} = \frac{\partial f}{\partial f} \cdot \frac{\partial f}{\partial e}$$

$$\frac{\partial f}{\partial f} = 1$$

$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial c}$$

Step 3: Forward pass - identify known expressions



$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial a}$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial b}$$

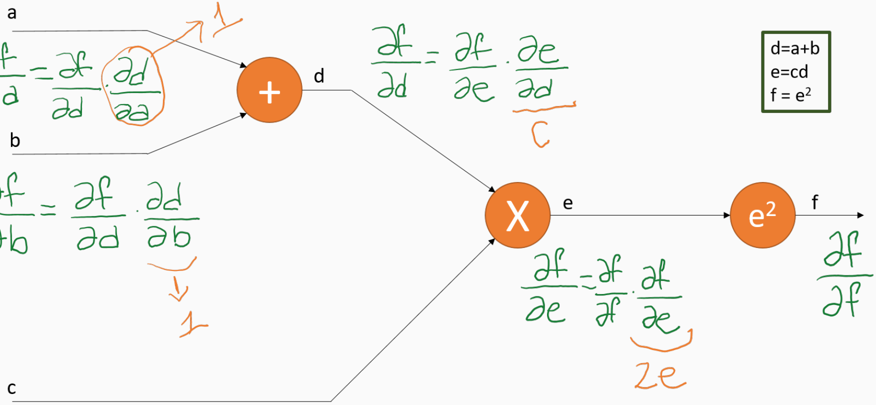
$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial c}$$

$$\frac{\partial f}{\partial d} = \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial d}$$

$$\frac{\partial f}{\partial e} = \frac{\partial f}{\partial f} \cdot \frac{\partial f}{\partial e}$$

$$\frac{\partial f}{\partial f} = 1$$

Step 4: Forward pass - replace known expressions



$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial a} \rightarrow 1$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial d} \cdot \underbrace{\frac{\partial d}{\partial b}}_{1}$$

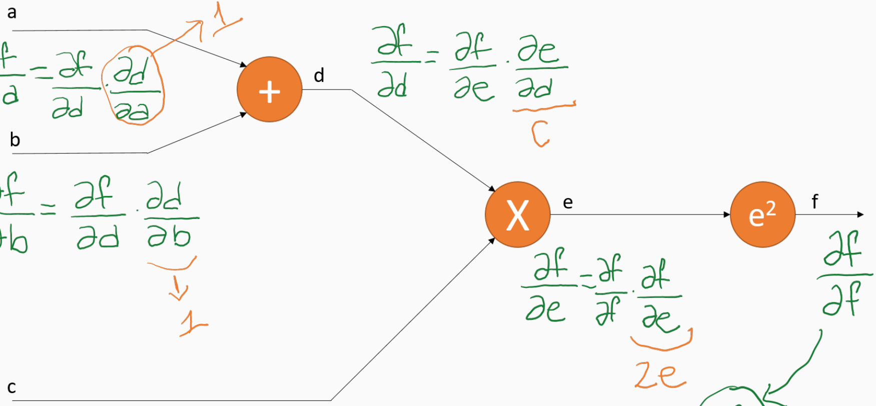
$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial c} \rightarrow d$$

$$\frac{\partial f}{\partial d} = \frac{\partial f}{\partial e} \cdot \underbrace{\frac{\partial e}{\partial d}}_{c}$$

$$\frac{\partial f}{\partial e} = \frac{\partial f}{\partial f} \cdot \underbrace{\frac{\partial f}{\partial e}}_{2e}$$

$$\frac{\partial f}{\partial f} = 1$$

Step 5: Backpropagation



$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial a} \rightarrow 1$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial d} \cdot \underbrace{\frac{\partial d}{\partial b}}_{1}$$

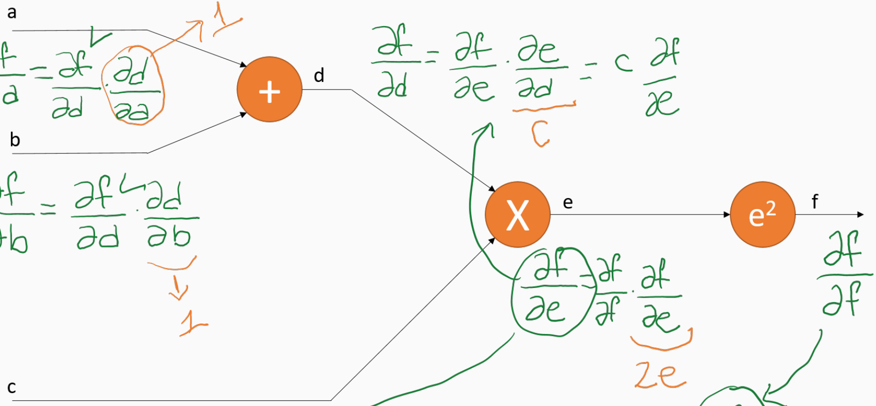
$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial c} \rightarrow d$$

$$\frac{\partial f}{\partial d} = \frac{\partial f}{\partial e} \cdot \underbrace{\frac{\partial e}{\partial d}}_{c}$$

$$\frac{\partial f}{\partial e} = \frac{\partial f}{\partial f} \cdot \underbrace{\frac{\partial f}{\partial e}}_{2e} = 2e \frac{\partial f}{\partial f}$$

$$\frac{\partial f}{\partial f} = 1$$

Step 6: Backpropagation to next layer



$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial a} \rightarrow 1$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial d} \cdot \underbrace{\frac{\partial d}{\partial b}}_{1}$$

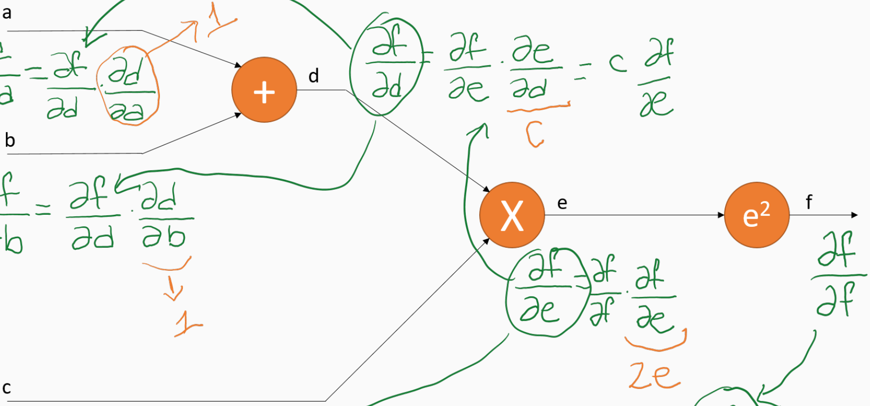
$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial c} \rightarrow d \checkmark$$

$$\frac{\partial f}{\partial d} = \frac{\partial f}{\partial e} \cdot \underbrace{\frac{\partial e}{\partial d}}_{c} = c\,\frac{\partial f}{\partial e}$$

$$\frac{\partial f}{\partial e} = \frac{\partial f}{\partial f} \cdot \underbrace{\frac{\partial f}{\partial e}}_{2e} = 2e\,\frac{\partial f}{\partial f}$$

$$\frac{\partial f}{\partial f} = 1$$

Process completed



$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial a} \rightarrow 1$$

$$\frac{\partial f}{\partial d} = \frac{\partial f}{\partial e} \cdot \underbrace{\frac{\partial e}{\partial d}}_{c} = c \, \frac{\partial f}{\partial e}$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial d} \cdot \underbrace{\frac{\partial d}{\partial b}}_{1}$$

$$\frac{\partial f}{\partial e} = \frac{\partial f}{\partial f} \cdot \underbrace{\frac{\partial f}{\partial e}}_{2e} = 2e \, \frac{\partial f}{\partial f}$$

$$\frac{\partial f}{\partial f} = 1$$

$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial c} \rightarrow d \quad ✓$$

The pattern

Each z gives a partial derivative of the form:

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial s} \frac{\partial s}{\partial z}$$

where z is an input to s

The pattern

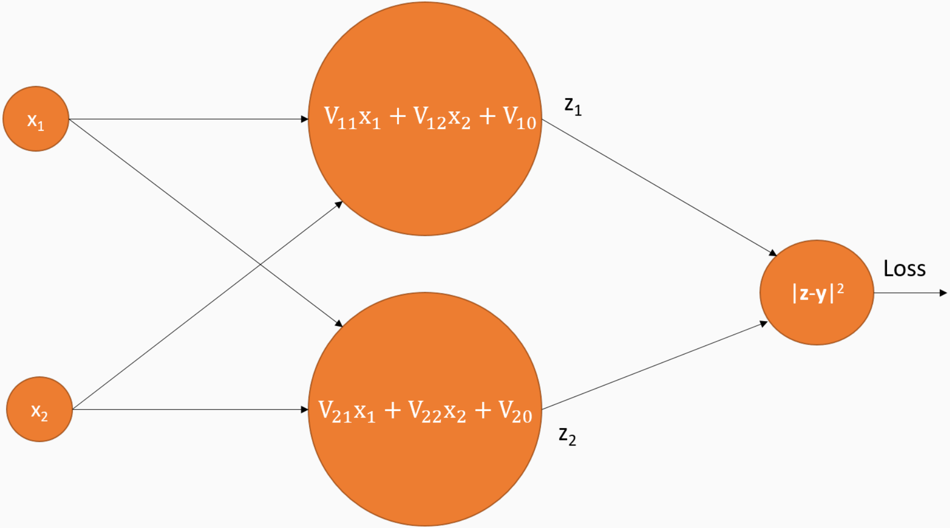
Each z gives a partial derivative of the form:

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial s} \frac{\partial s}{\partial z}$$

where z is an input to s

- Green term: Computed in the backward pass
- Orange term: Computed in the forward pass

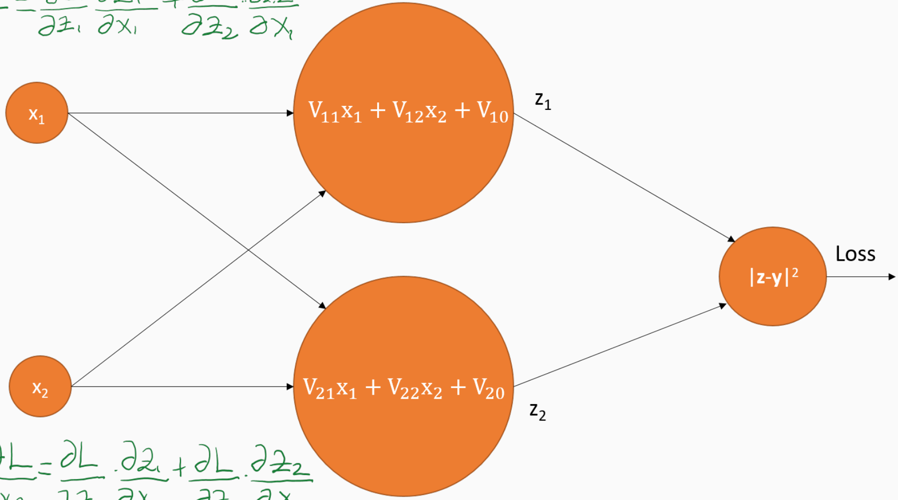
A more complex example



Step 1: Chain rule

We need to make use of multi-variate calculus

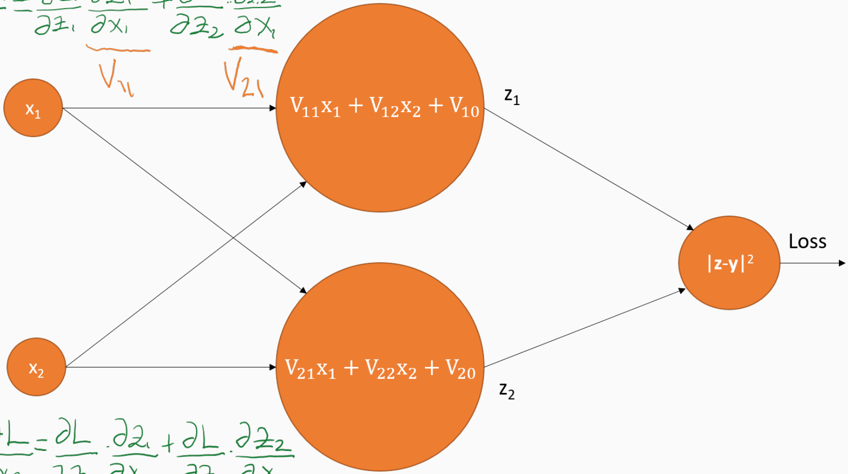
$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial x_1} + \frac{\partial L}{\partial z_2}\cdot\frac{\partial z_2}{\partial x_1}$$



$$\frac{\partial L}{\partial x_2} = \frac{\partial L}{\partial z_1}\cdot\frac{\partial z_1}{\partial x_2} + \frac{\partial L}{\partial z_2}\cdot\frac{\partial z_2}{\partial x_2}$$

Step 2: Forward pass estimations

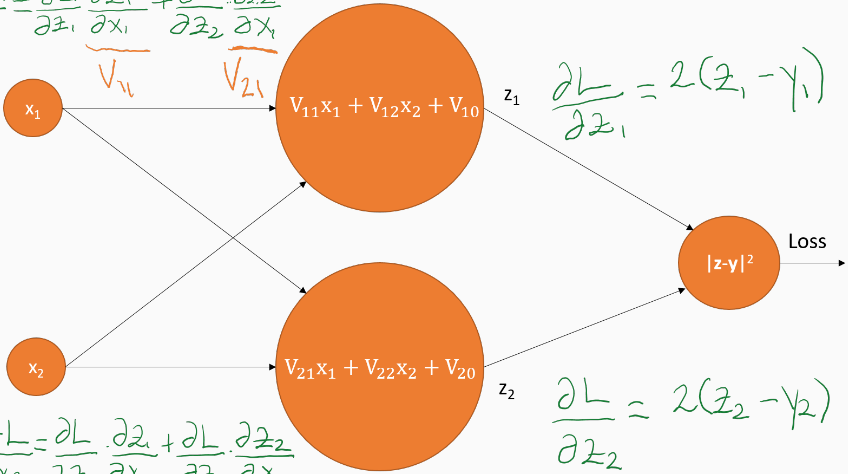
$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial z_1} \underbrace{\frac{\partial z_1}{\partial x_1}}_{V_{11}} + \frac{\partial L}{\partial z_2} \cdot \underbrace{\frac{\partial z_2}{\partial x_1}}_{V_{21}}$$



$$\frac{\partial L}{\partial x_2} = \frac{\partial L}{\partial z_1} \cdot \underbrace{\frac{\partial z_1}{\partial x_2}}_{V_{12}} + \frac{\partial L}{\partial z_2} \cdot \underbrace{\frac{\partial z_2}{\partial x_2}}_{V_{22}}$$

Step 3: Last layer

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial z_1}\underbrace{\frac{\partial z_1}{\partial x_1}}_{V_{11}} + \frac{\partial L}{\partial z_2}\underbrace{\frac{\partial z_2}{\partial x_1}}_{V_{21}}$$



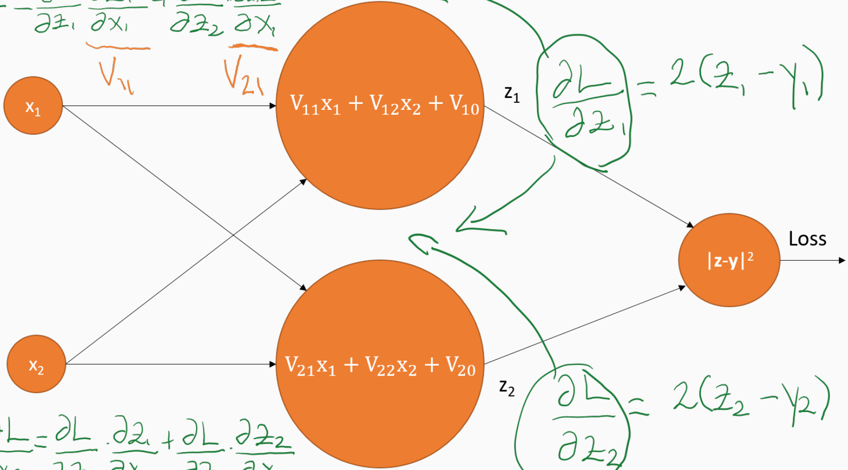
$$\frac{\partial L}{\partial z_1} = 2(z_1 - y_1)$$

$$\frac{\partial L}{\partial z_2} = 2(z_2 - y_2)$$

$$\frac{\partial L}{\partial x_2} = \frac{\partial L}{\partial z_1}\underbrace{\frac{\partial z_1}{\partial x_2}}_{V_{12}} + \frac{\partial L}{\partial z_2}\underbrace{\frac{\partial z_2}{\partial x_2}}_{V_{22}}$$

Step 4: Backpropagation

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial z_1}\underbrace{\frac{\partial z_1}{\partial x_1}}_{V_{11}} + \frac{\partial L}{\partial z_2}\cdot\underbrace{\frac{\partial z_2}{\partial x_1}}_{V_{21}}$$



$$\frac{\partial L}{\partial z_1} = 2(z_1 - y_1)$$

$$\frac{\partial L}{\partial z_2} = 2(z_2 - y_2)$$

$$\frac{\partial L}{\partial x_2} = \frac{\partial L}{\partial z_1}\cdot\underbrace{\frac{\partial z_1}{\partial x_2}}_{V_{12}} + \frac{\partial L}{\partial z_2}\cdot\underbrace{\frac{\partial z_2}{\partial x_2}}_{V_{22}}$$

The Backpropagation Algorithm

- The backpropagation algorithm is a dynamic programming algorithm for computing the gradients. Using stochastic gradient descent it allows to do this computation in linear time w.r.t the number of weights.
- It was key to the re-birth of neural networks.

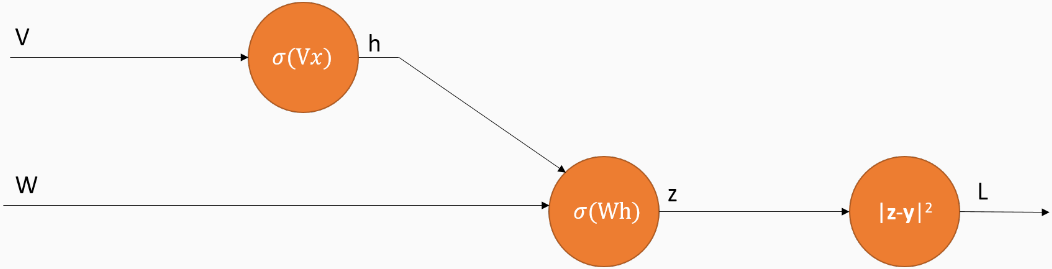
The Backpropagation Algorithm

- The backpropagation algorithm is a dynamic programming algorithm for computing the gradients. Using stochastic gradient descent it allows to do this computation in linear time w.r.t the number of weights.
- It was key to the re-birth of neural networks.
- Let's use it to find the terms for the first neural network we used.

A quick recap on the terms

- **Input layer:** $\mathbf{x}_1, \ldots, \mathbf{x}_D; \quad \mathbf{x}_0 = 1$
- **Layer 1 weights:** $M \times (D+1)$ matrix $\mathbf{V}$, with $\mathbf{V}_j^T$ the j^{th} row
- **Layer 2 weights:** $k \times (M+1)$ matrix $\mathbf{W}$, with $\mathbf{W}_i^T$ the i^{th} row
- **The sigmoid:** $\sigma(a) = \frac{1}{1+\exp(-a)}$
- **Hidden units:** $h_j = \sigma(\mathbf{V}_j \mathbf{x})$
- **Output layer:** $z_i = \sigma(\mathbf{W}_i \mathbf{h})$
- **Sigmoid derivative:** $\sigma'(a) = \sigma(a)\,(1 - \sigma(a))$

The network revisited

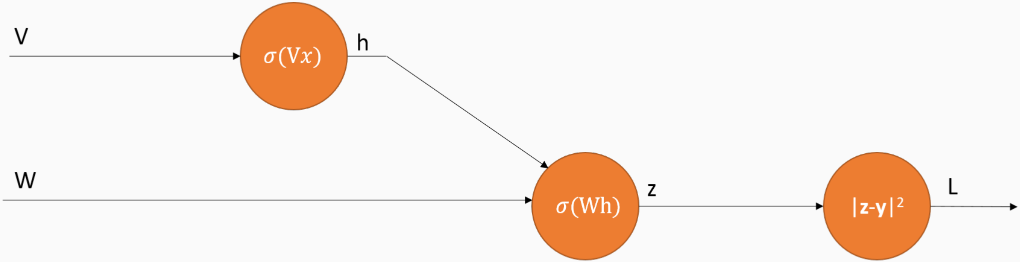


This is the same network as the one in slide 12

It has been reformulated using only W and V as inputs

If you are not convinced, try expanding the terms

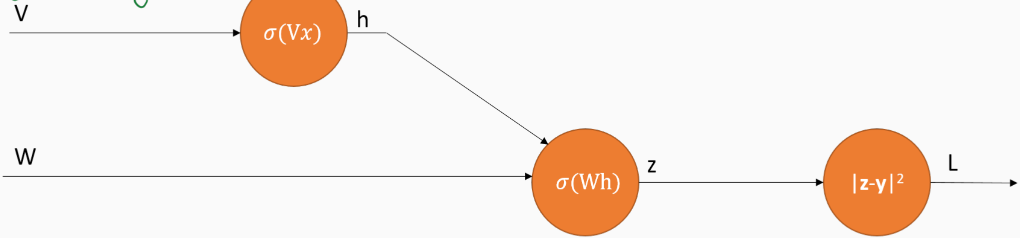
Step 1: The chain rule in the forward pass



$$\nabla_{w_i} L = \frac{\partial L}{\partial z_i} \cdot \nabla_{w_i} z_i$$

Step 1: The chain rule in the forward pass

$$\nabla_{v_j} L = \frac{\partial L}{\partial h_j} \cdot \nabla_{v_j} h_j$$



$$\nabla_{w_i} L = \frac{\partial L}{\partial z_i} \cdot \nabla_{w_i} z_i$$

Step 2: Identifying the derivatives

We need to replace the terms that we can estimate during the forward pass. We can rely on the terms that we listed in slide 35:

Step 2: Identifying the derivatives

We need to replace the terms that we can estimate during the forward pass. We can rely on the terms that we listed in slide 35:

$$\nabla_{\mathbf{V}_j} h_j = \sigma'(\mathbf{V}_j \mathbf{x}) = h_j(1 - h_j)\mathbf{x}$$

Step 2: Identifying the derivatives

We need to replace the terms that we can estimate during the forward pass. We can rely on the terms that we listed in slide 35:

$$\nabla_{\mathbf{V}_j} h_j = \sigma'(\mathbf{V}_j \mathbf{x}) = h_j(1 - h_j)\mathbf{x}$$
$$\nabla_{\mathbf{W}_i} z_i = \sigma'(\mathbf{W}_i \mathbf{h}) = z_i(1 - z_i)\mathbf{h}$$

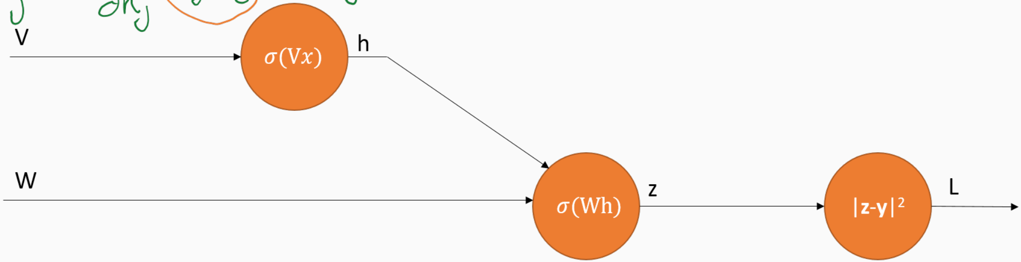
Step 2: Identifying the derivatives

We need to replace the terms that we can estimate during the forward pass. We can rely on the terms that we listed in slide 35:

$$\nabla_{\mathbf{V}_j} h_j = \sigma'(\mathbf{V}_j \mathbf{x}) = h_j(1 - h_j)\mathbf{x}$$
$$\nabla_{\mathbf{W}_i} z_i = \sigma'(\mathbf{W}_i \mathbf{h}) = z_i(1 - z_i)\mathbf{h}$$
$$\nabla_h z_i = \sigma'(\mathbf{W}_i z_i) = z_i(1 - z_i)\mathbf{W}_i$$

Step 3: Forward pass continuation

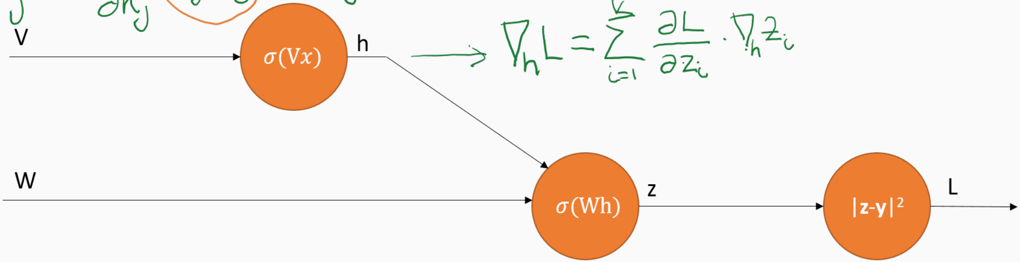
$$\nabla_{V_j} L = \frac{\partial L}{\partial h_j} \nabla_{V_j} h_j = \frac{\partial L}{\partial h_j} \cdot h_j(1-h_j)x$$



$$\nabla_{W_i} L = \frac{\partial L}{\partial z_i} \nabla_{W_i} z_i = \frac{\partial L}{\partial z_i} \cdot z_i(1-z_i)h$$

Step 4: Next layer of forward pass

$$\nabla_{v_j} L = \frac{\partial L}{\partial h_j} \nabla_{v_j} h_j = \frac{\partial L}{\partial h_j} \cdot h_j(1-h_j)x$$



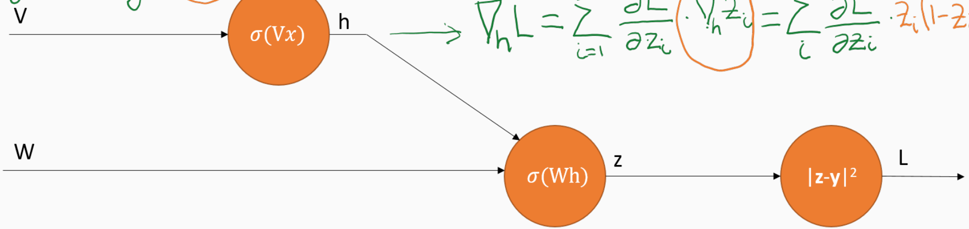
$$\longrightarrow \nabla_h L = \sum_{i=1}^{k} \frac{\partial L}{\partial z_i} \cdot \nabla_h z_i$$

$$\nabla_{w_i} L = \frac{\partial L}{\partial z_i} \nabla_{w_i} z_i = \frac{\partial L}{\partial z_i} \cdot z_i(1-z_i)h$$

Step 5: Next layer of forward pass

$$\nabla_{V_j} L = \frac{\partial L}{\partial h_j} \cdot \nabla_{V_j} h_j = \frac{\partial L}{\partial h_j} \cdot h_j(1-h_j)x$$

$$\longrightarrow \nabla_h L = \sum_{i=1}^{k} \frac{\partial L}{\partial z_i} \cdot \nabla_h z_i = \sum_i \frac{\partial L}{\partial z_i} \cdot z_i(1-z_i) W_i$$

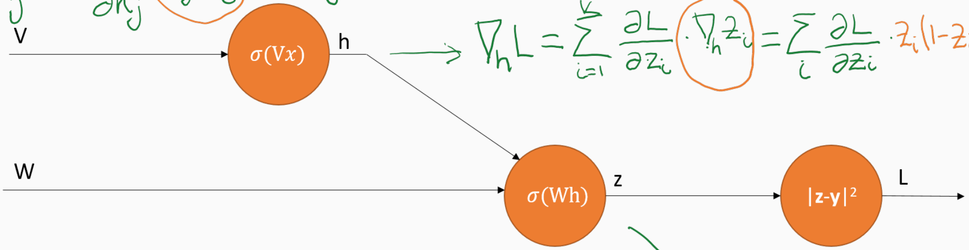


$$\nabla_{W_i} L = \frac{\partial L}{\partial z_i} \cdot \nabla_{W_i} z_i = \frac{\partial L}{\partial z_i} \cdot z_i(1-z_i) h$$

Step 6: Final layer

$$\nabla_{v_j} L = \frac{\partial L}{\partial h_j} \cdot \nabla_{v_j} h_j = \frac{\partial L}{\partial h_j} \cdot h_j(1-h_j)x$$

$$\nabla_h L = \sum_{i=1}^{k} \frac{\partial L}{\partial z_i} \cdot \nabla_h z_i = \sum_i \frac{\partial L}{\partial z_i} \cdot z_i(1-z_i) w_i$$



$$\nabla_z L = 2(z-y)$$

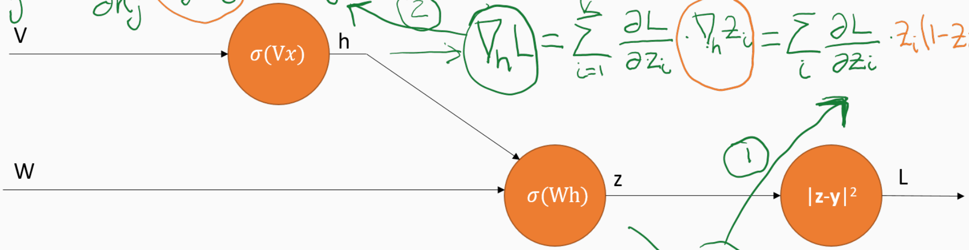
$$\nabla_{w_i} L = \frac{\partial L}{\partial z_i} \cdot \nabla_{w_i} z_i = \frac{\partial L}{\partial z_i} \cdot z_i(1-z_i) h$$

Step 7: Backpropagation

$$\nabla_{v_j} L = \frac{\partial L}{\partial h_j} \cdot \nabla_{v_j} h_j = \frac{\partial L}{\partial h_j} \cdot h_j(1-h_j)x$$

(2)

$$\nabla_h L = \sum_{i=1}^{k} \frac{\partial L}{\partial z_i} \cdot \nabla_h z_i = \sum_i \frac{\partial L}{\partial z_i} \cdot z_i(1-z_i)W_i$$



(1)

$$\nabla_z L = 2(z-y)$$

$$\nabla_{W_i} L = \frac{\partial L}{\partial z_i} \cdot \nabla_{W_i} z_i = \frac{\partial L}{\partial z_i} \cdot z_i(1-z_i)h$$

(1)

How to use this?

1. For an input vector $\mathbf{x}_i$ to the network, do a forward pass over the full network. The required computations can be generalized as:

$$\phi_i = a_i = \sum_j W_{ij} z_j, \qquad z_i = h(a_i)$$

   where we denote a_i the activation.

How to use this?

1. For an input vector $\mathbf{x}_i$ to the network, do a forward pass over the full network. The required computations can be generalized as:

$$\phi_i = a_i = \sum_j W_{ij} z_j, \qquad z_i = h(a_i)$$

   where we denote a_i the activation.

2. Evaluate the error in the output units. We will formalize it as:

$$\delta_k = \nabla L(\hat{y}_k, y_k) \cdot h'(a_k^{(L)})$$

How to use this?

1. For an input vector $\mathbf{x}_i$ to the network, do a forward pass over the full network. The required computations can be generalized as:

$$\phi_i = a_i = \sum_j W_{ij} z_j, \qquad z_i = h(a_i)$$

   where we denote a_i the activation.

2. Evaluate the error in the output units. We will formalize it as:

$$\delta_k = \nabla L(\hat{y}_k, y_k) \cdot h'(a_k^{(L)})$$

3. Backward pass the δs to the previous hidden units. This can be generalized as:

$$\delta_j^{(l)} = h'\left(a_j^{(l)}\right) \sum_i W_{ij}^{(l+1)} \delta_i^{(l+1)}$$

How to use this?

What we did in the previous steps:

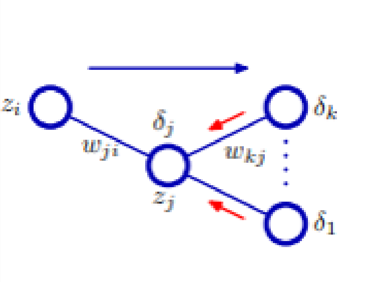


Figure 5.7 Bishop - PRML

$$\phi_i = a_i = \sum_j W_{ij} z_j, \qquad z_i = h(a_i)$$

$$\delta_j^{(l)} = h'\left(a_j^{(l)}\right) \sum_i W_{ij}^{(l+1)} \delta_i^{(l+1)}$$

How to use this?

4. Estimate all the required derivatives $\nabla_{\mathbf{w}} L$

How to use this?

4. Estimate all the required derivatives $\nabla_{\mathbf{w}} L$
5. Change the weights based on estimated gradients by $-\alpha \nabla_{\mathbf{w}} L$, where α is the learning rate.

How to use this?

4. Estimate all the required derivatives $\nabla_{\mathbf{w}} L$
5. Change the weights based on estimated gradients by $-\alpha \nabla_{\mathbf{w}} L$, where α is the learning rate.
6. Go back to step 1 and repeat until a number of iterations or a desired minimum is reached

The MLP

The Multi Layer Perceptron

Advantages

- Very general, can be applied in many situations
- Powerful according to theory
- Efficient according to practice

The Multi Layer Perceptron

Advantages

- Very general, can be applied in many situations
- Powerful according to theory
- Efficient according to practice

Drawbacks

- Training is often slow
- Choice of optimal number of layers & neurons difficult
- Little understanding of real model

Recap

Recap

In this lecture...

- We introduced feedforward networks, aka the multilayer perceptron
- We introduced the backpropagation algorithm which is the mechanism to train feedforward networks
- We saw the strengths but also the limitations of MLPs
- Deep Learning course (spring term) if you want to learn about more powerful neural network architectures

Disclaimer: Part of the examples in this lecture have been adapted from J. Shewchuk course

Key Concepts

- Backpropagation
- Multilayered perceptron (MLP)
- Feedforward networks
- Chain rule

References

Further Reading and Useful Material

Source	Notes
Pattern Recognition and Machine Learning	Ch 5
The Elements of Statistical Learning	Ch 11