Deep Learning Bootcamp

A hands-on intensive workshop covering neural fundamentals,
computer vision, and natural language processing using modern deep learning tools.

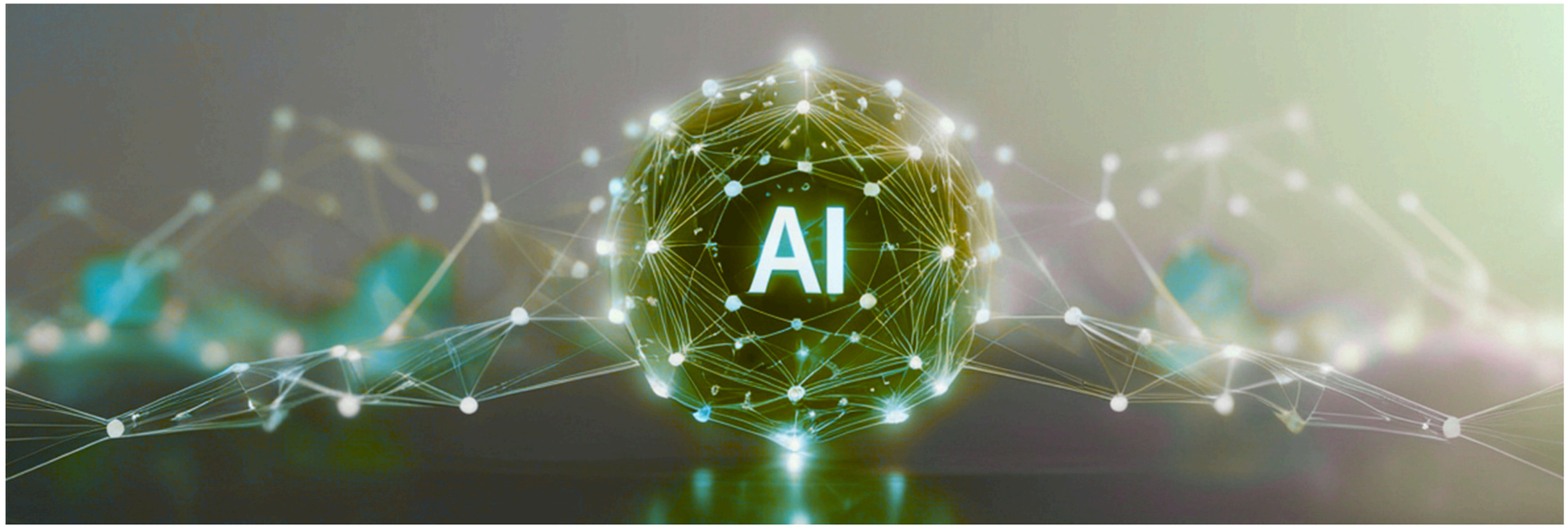
by: Lama Ayash

Day 1

Course Outlines

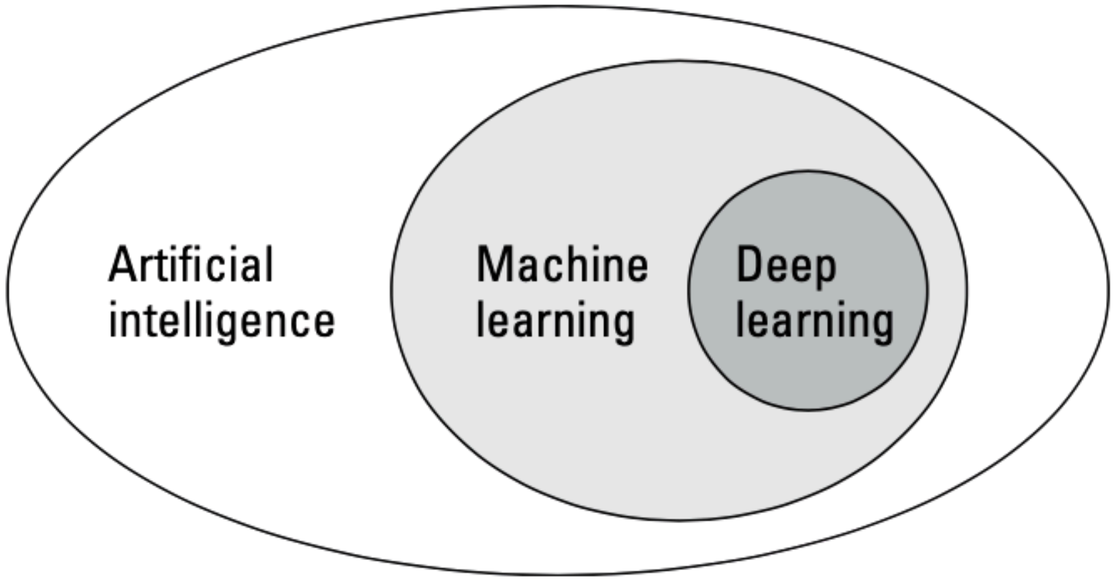
The Deep Learning Revolution

- Artificial Intelligence (AI), coined by John McCarthy in the 1950s, refers to machine-exhibited intelligence.
- AI spans from early expert systems to today’s advanced industry models.



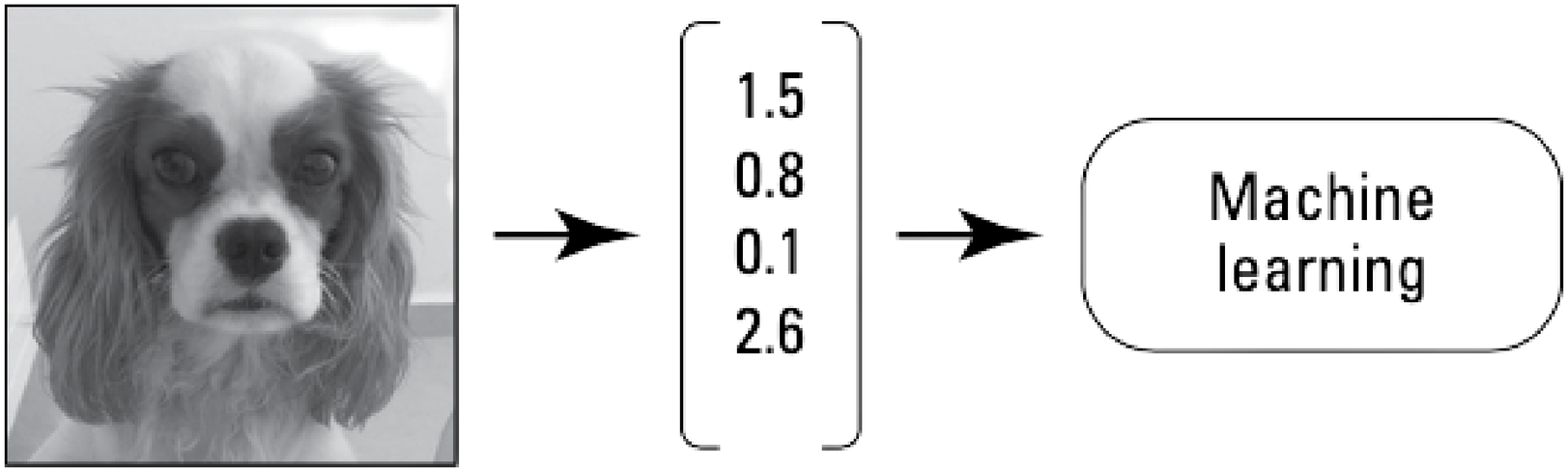
The Deep Learning Revolution

- Deep learning rapidly boosted AI performance (20–30%) in months instead of years.
- Progress was driven by improved algorithms and GPU computing.
- GPUs accelerated training by up to 100× over CPUs.
- NVIDIA GPUs became central to deep learning research.



Machine Learning Drawbacks

- Domain experts define features, such as **size, color, or texture,** based on problem knowledge.
- Each sample is represented as a numeric vector, comprising the selected features fed into the ML algorithm.



Machine Learning Drawbacks

Is this a cat or a dog?

Machine Learning Drawbacks

Is this a cat or a dog?

- **Drawback 1:** human-designed features capture only simple, linear patterns , not complex correlations.

Machine Learning Drawbacks

512 × 512 pixels

Feature extraction

1,5
0,8
0,1
2,6
...

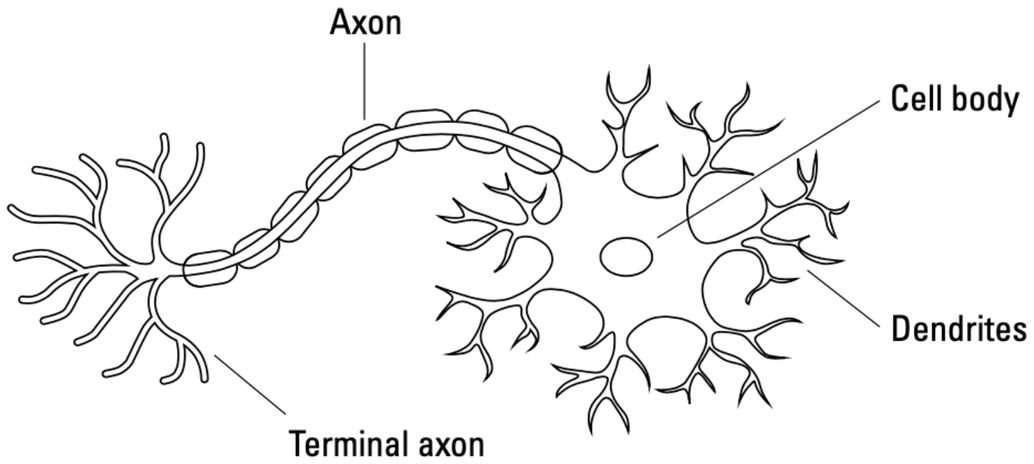
Feature vector

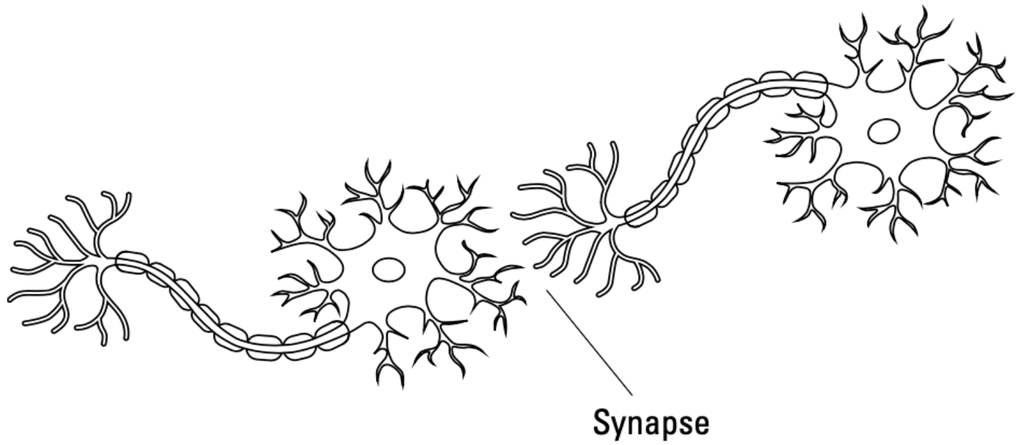
- **Drawback 1:** human-designed features capture only simple, linear patterns , not complex correlations.

- **Drawback 2:** most raw data information is lost when reduced to a few features.

The Inspiration Behind Neural Networks

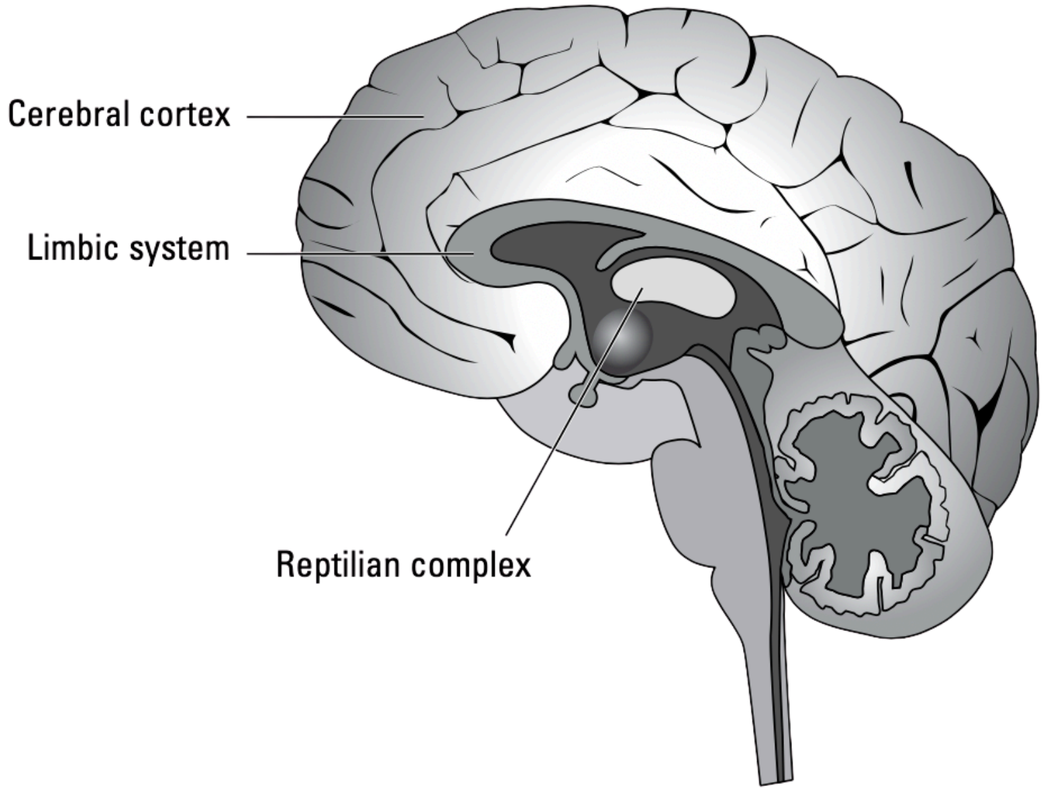
- The human brain comprises tens of billions of **neurons** connected through **synapses**.
- They don’t “know” what they process until trained through input connections.
- Learning happens by strengthening or weakening synaptic connections.





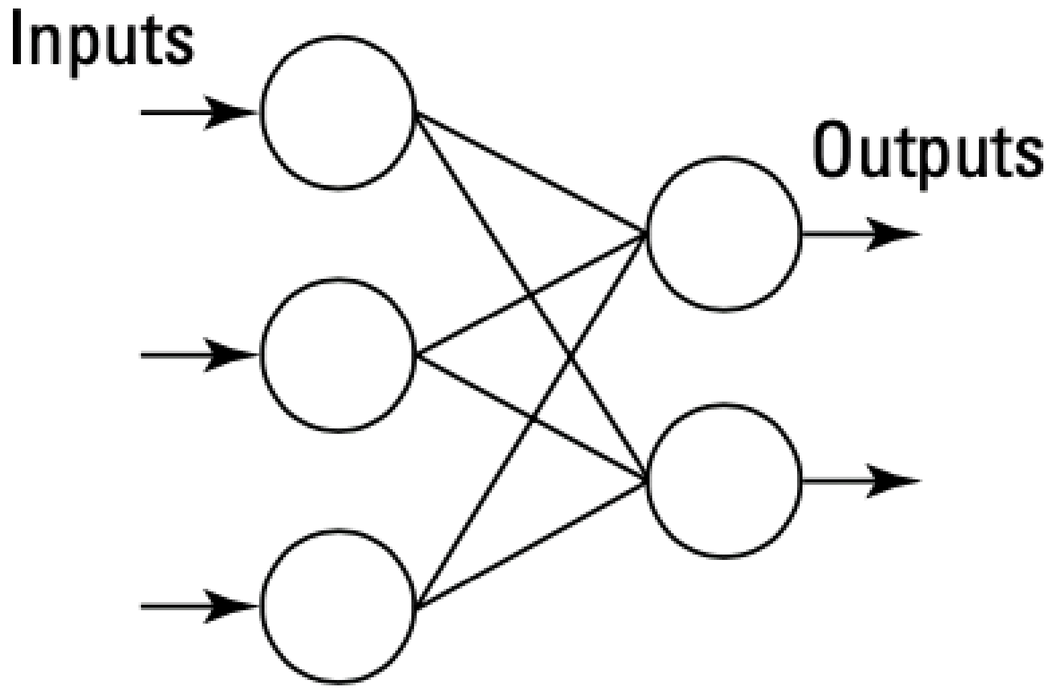
The Inspiration Behind Neural Networks

- **Suzana Herculano-Houzel:** shows that intelligence correlates with the **number of neurons** in the cerebral cortex, not total brain size.



The Inspiration Behind Neural Networks

- **1943**: McCulloch and Pitts proposed the first mathematical model of an artificial neuron.
- **1958**: Frank Rosenblatt introduced the perceptron, a two-layer neural network
- Each connection between neurons carries a weight, and the output is the weighted sum of inputs.



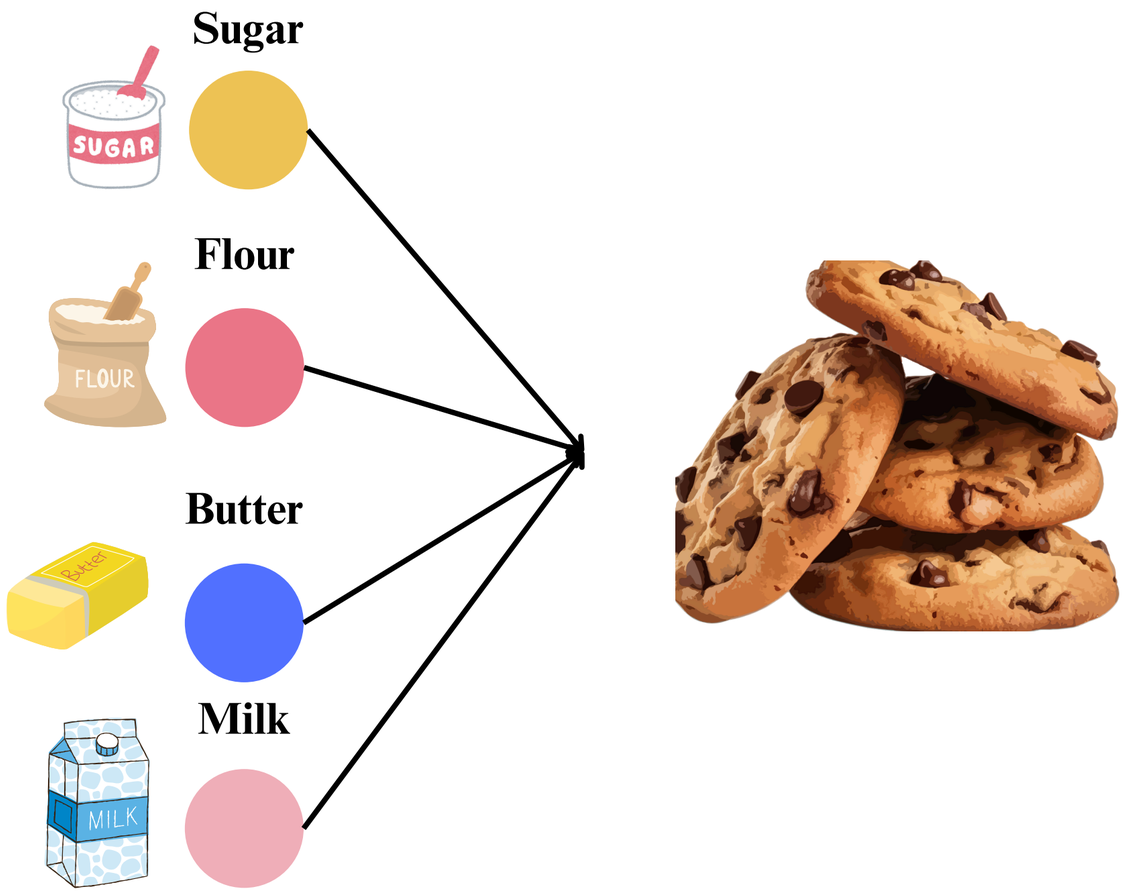
How Do Artificial Neural Networks Work?

Will the Cookie Taste Good?

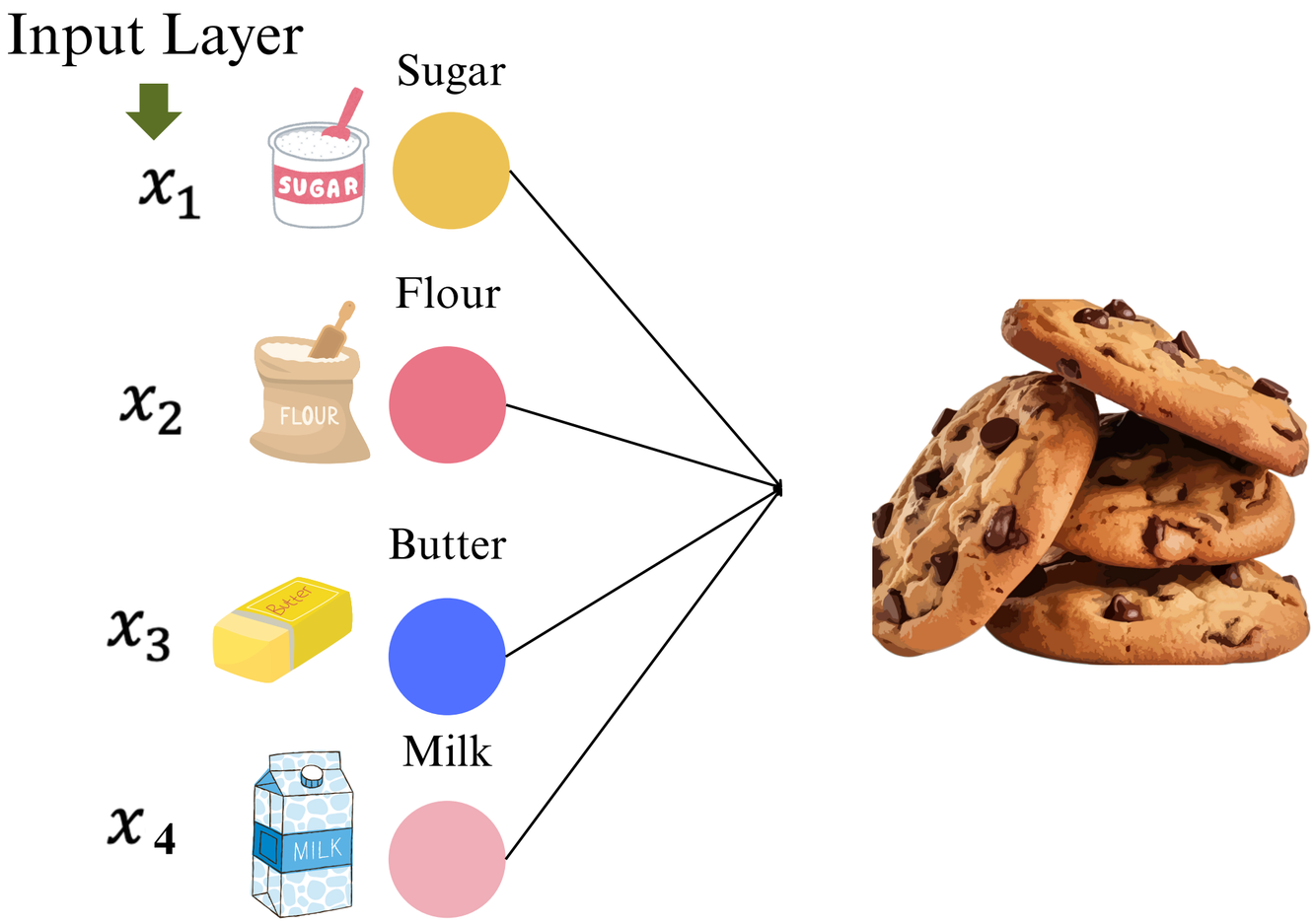
Artificial Neural Networks Example

Will the Cookie Taste Good?

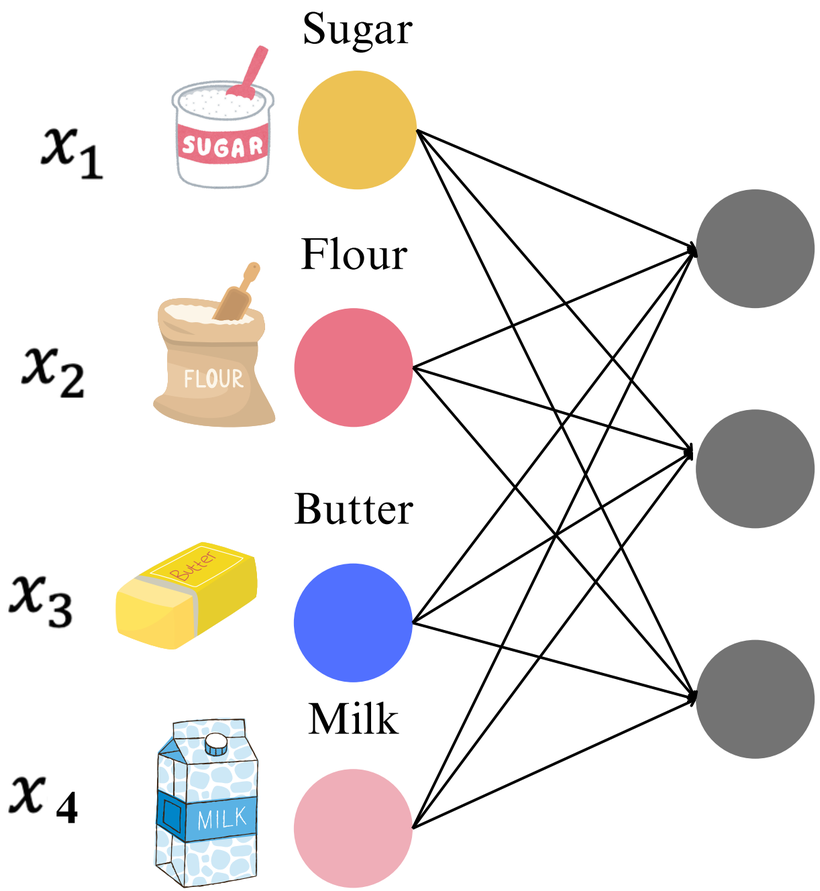
Artificial Neural Networks Example



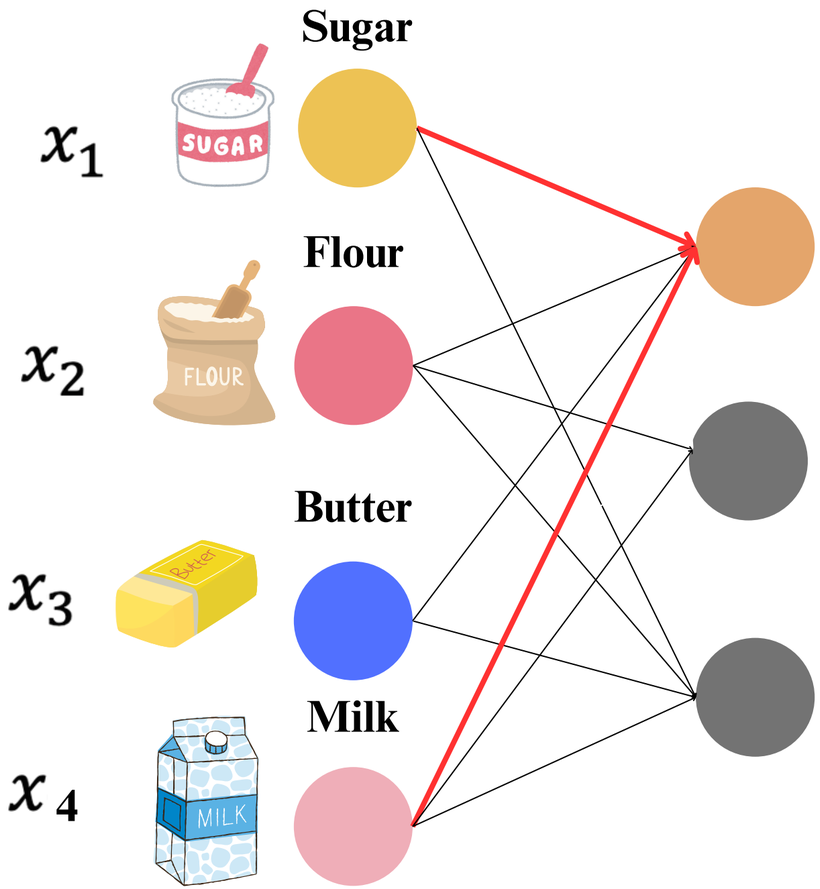
Artificial Neural Networks Example



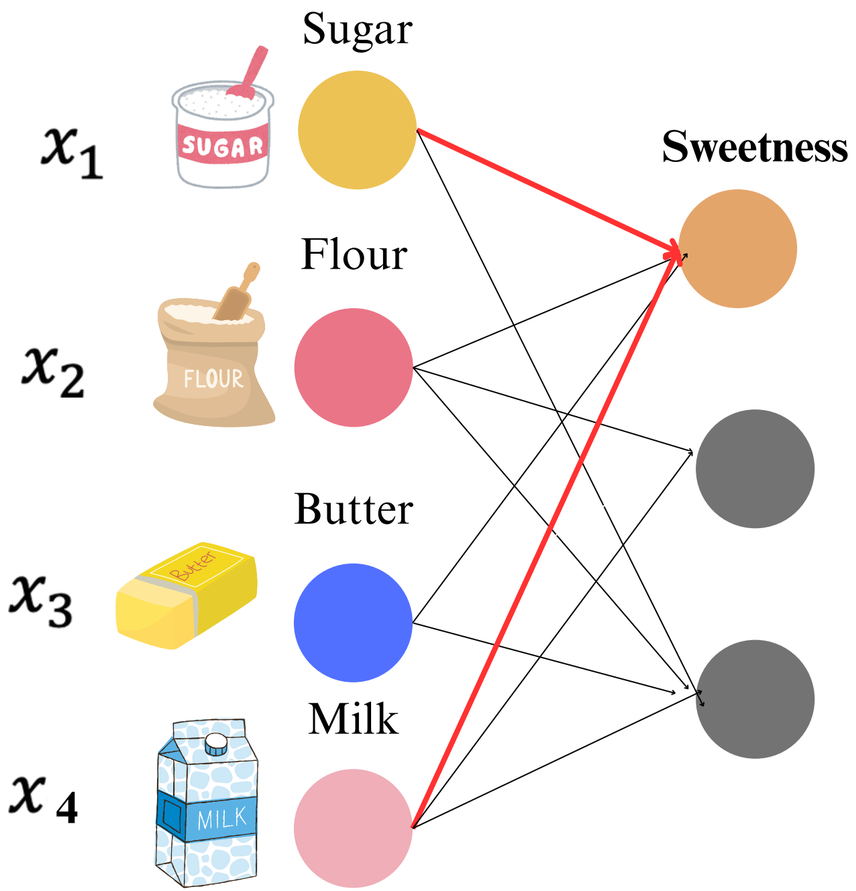
Artificial Neural Networks Example



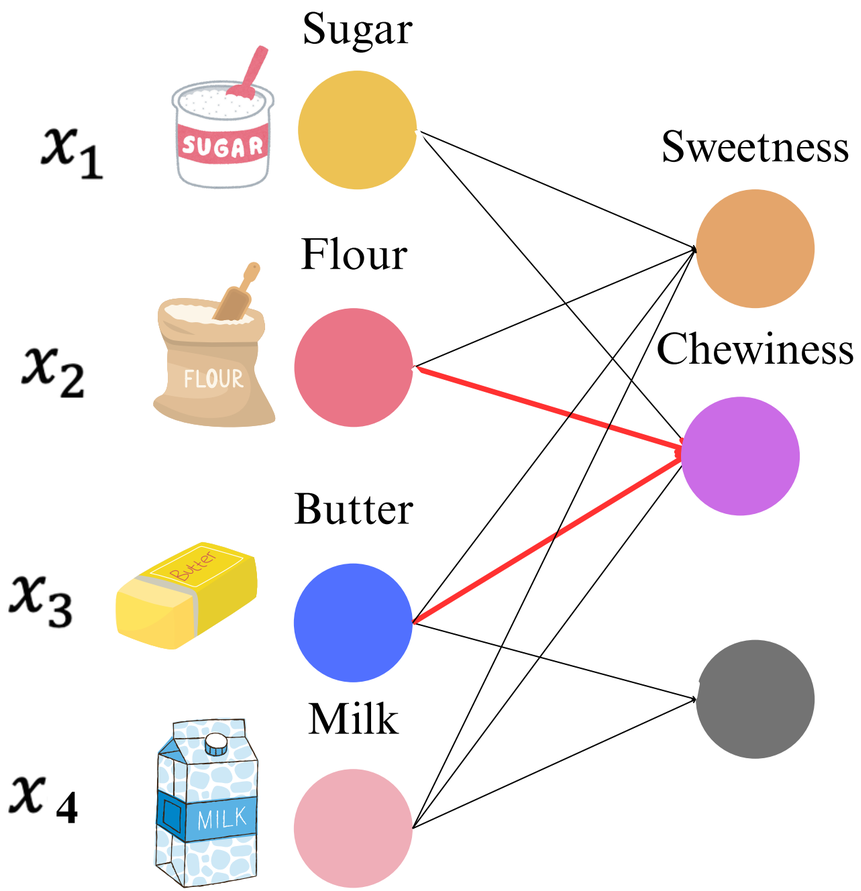
Artificial Neural Networks Example



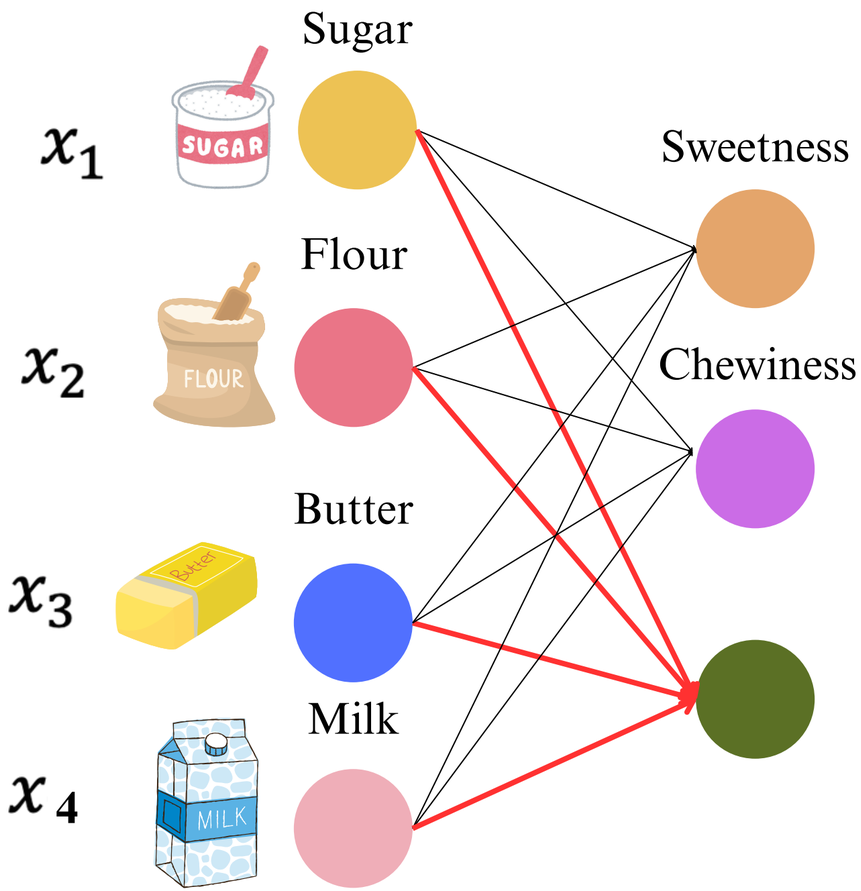
Artificial Neural Networks Example



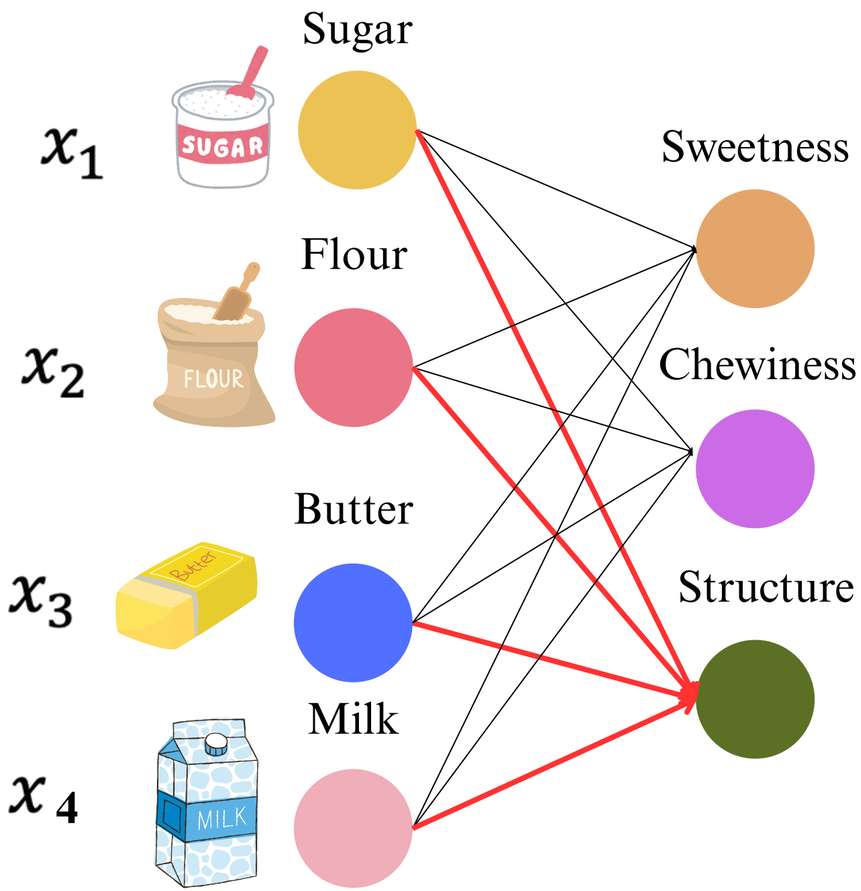
Artificial Neural Networks Example



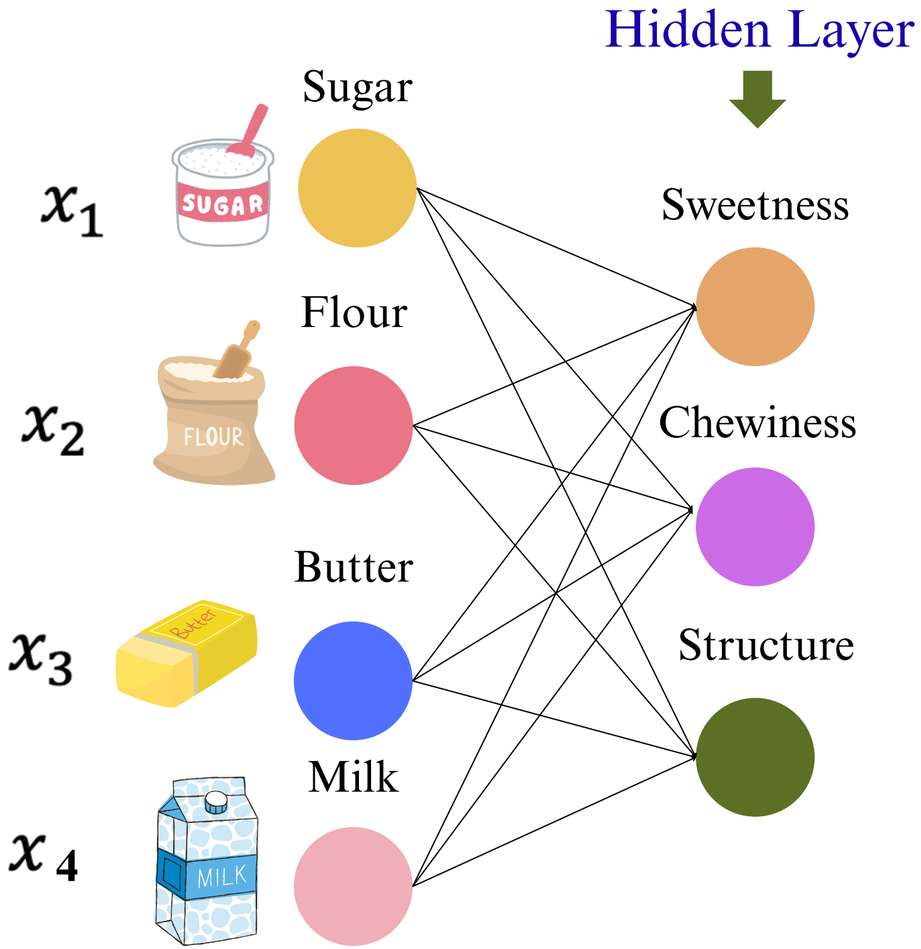
Artificial Neural Networks Example



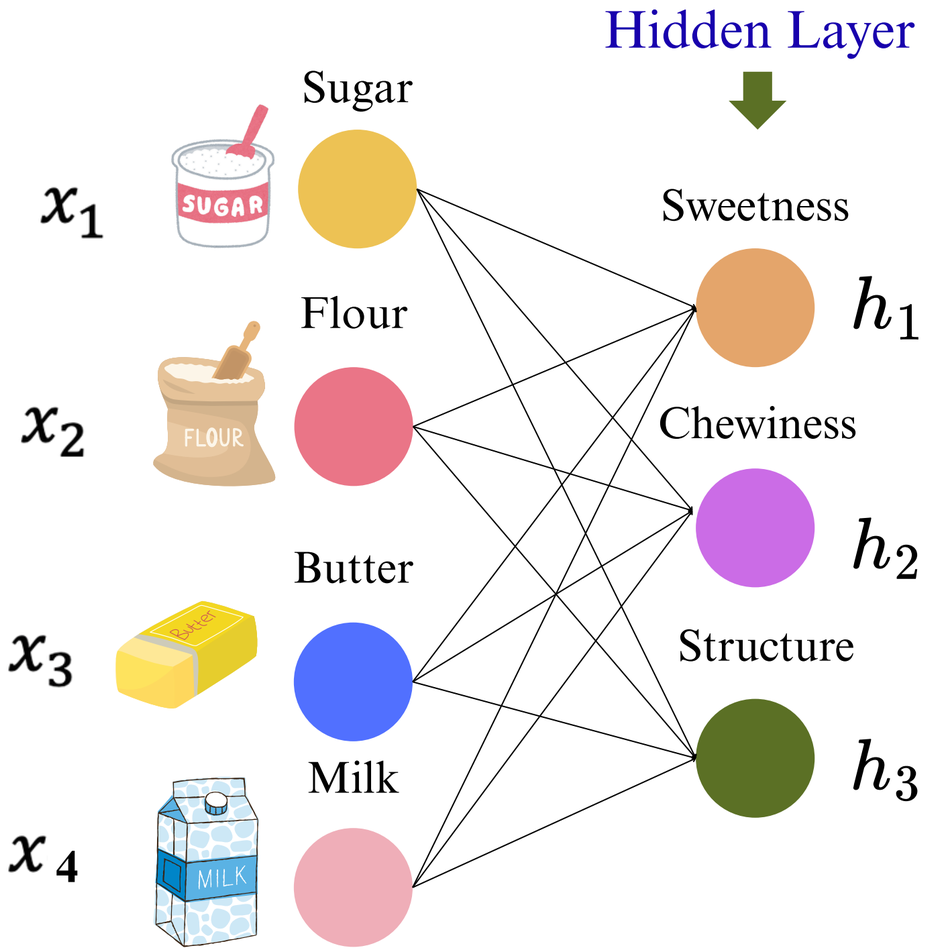
Artificial Neural Networks Example



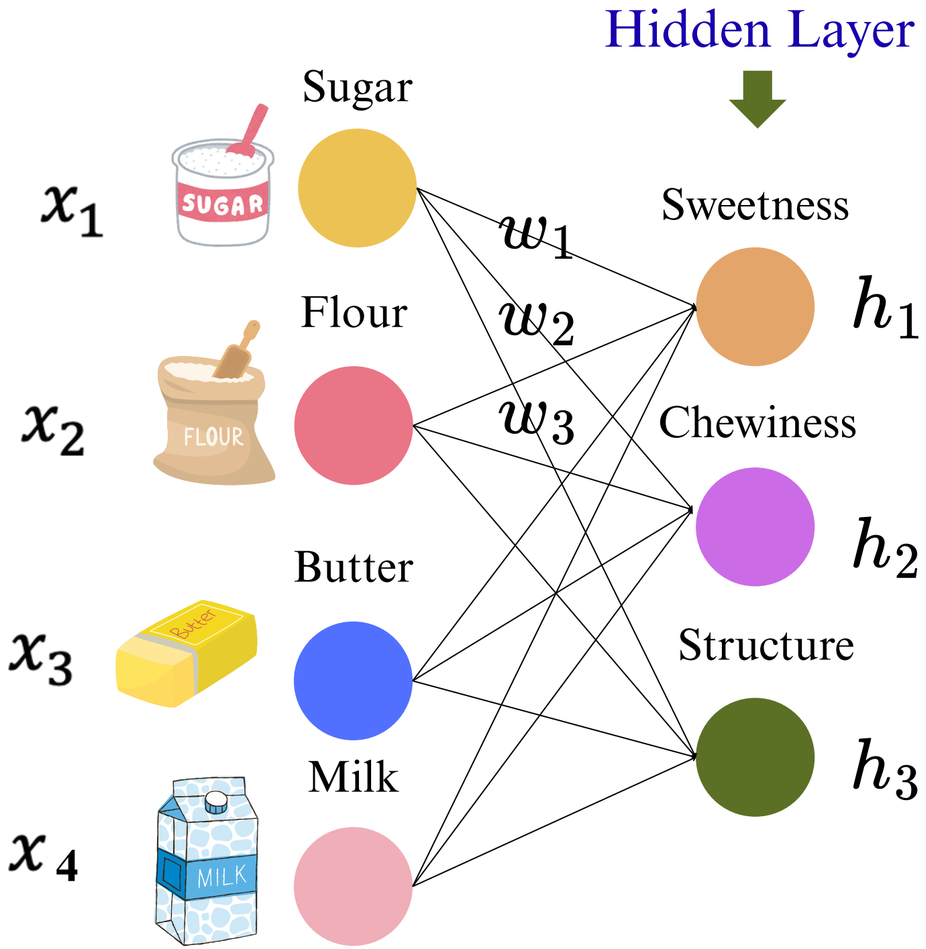
Artificial Neural Networks Example



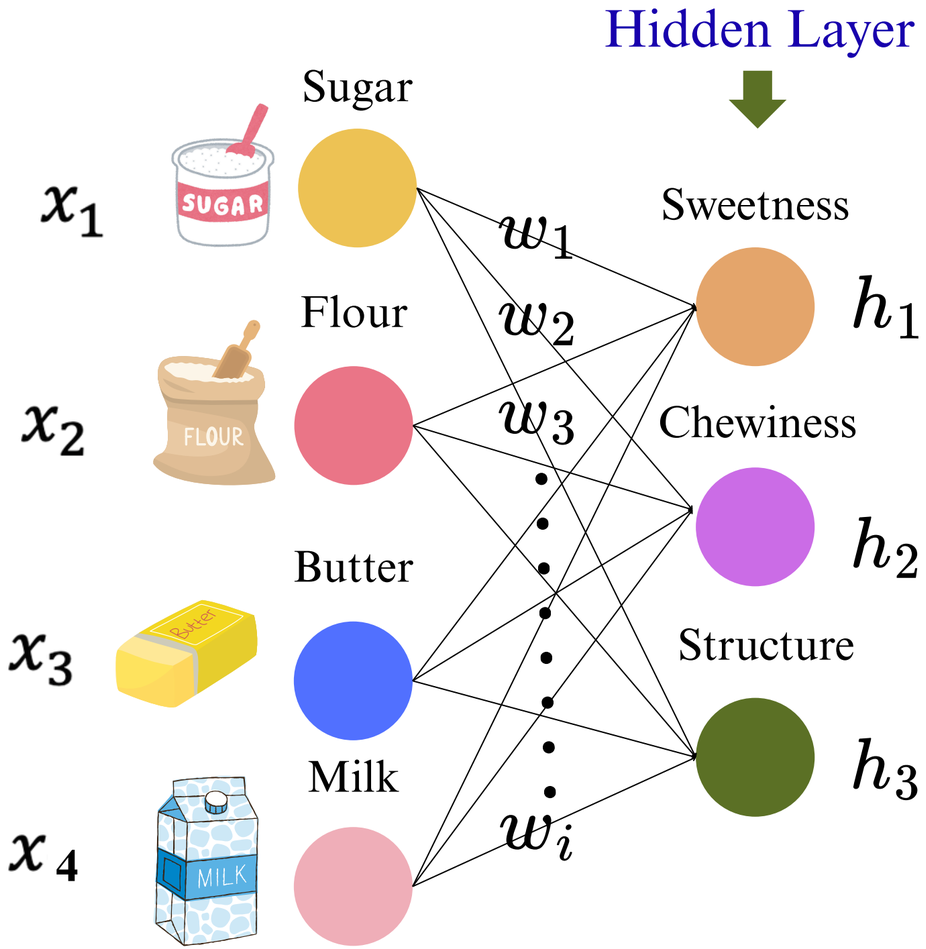
Artificial Neural Networks Example



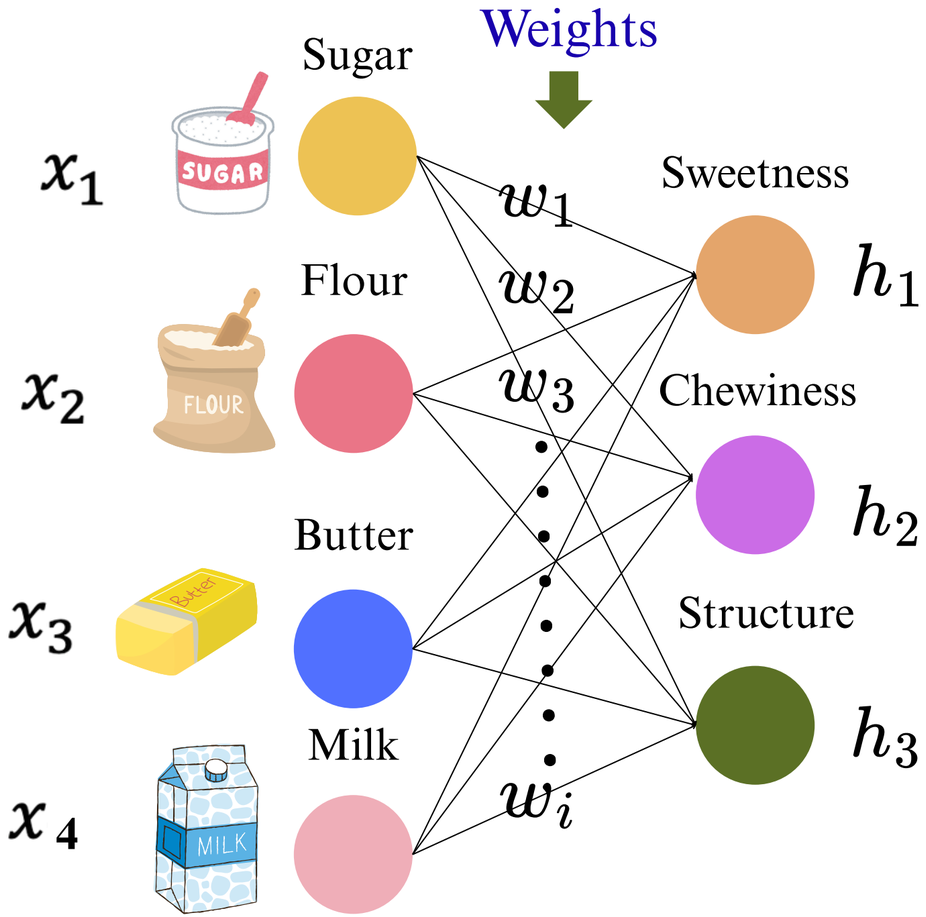
Artificial Neural Networks Example



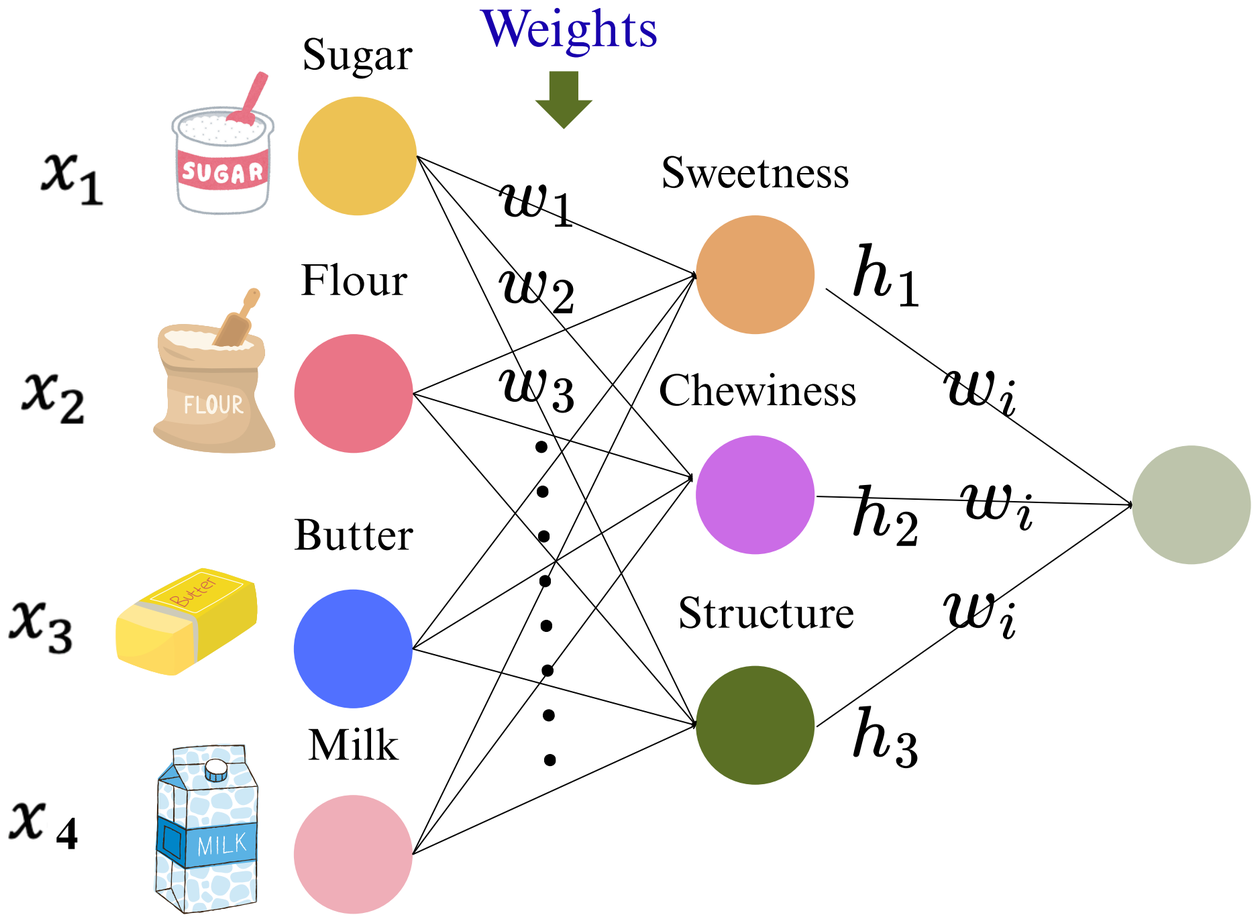
Artificial Neural Networks Example



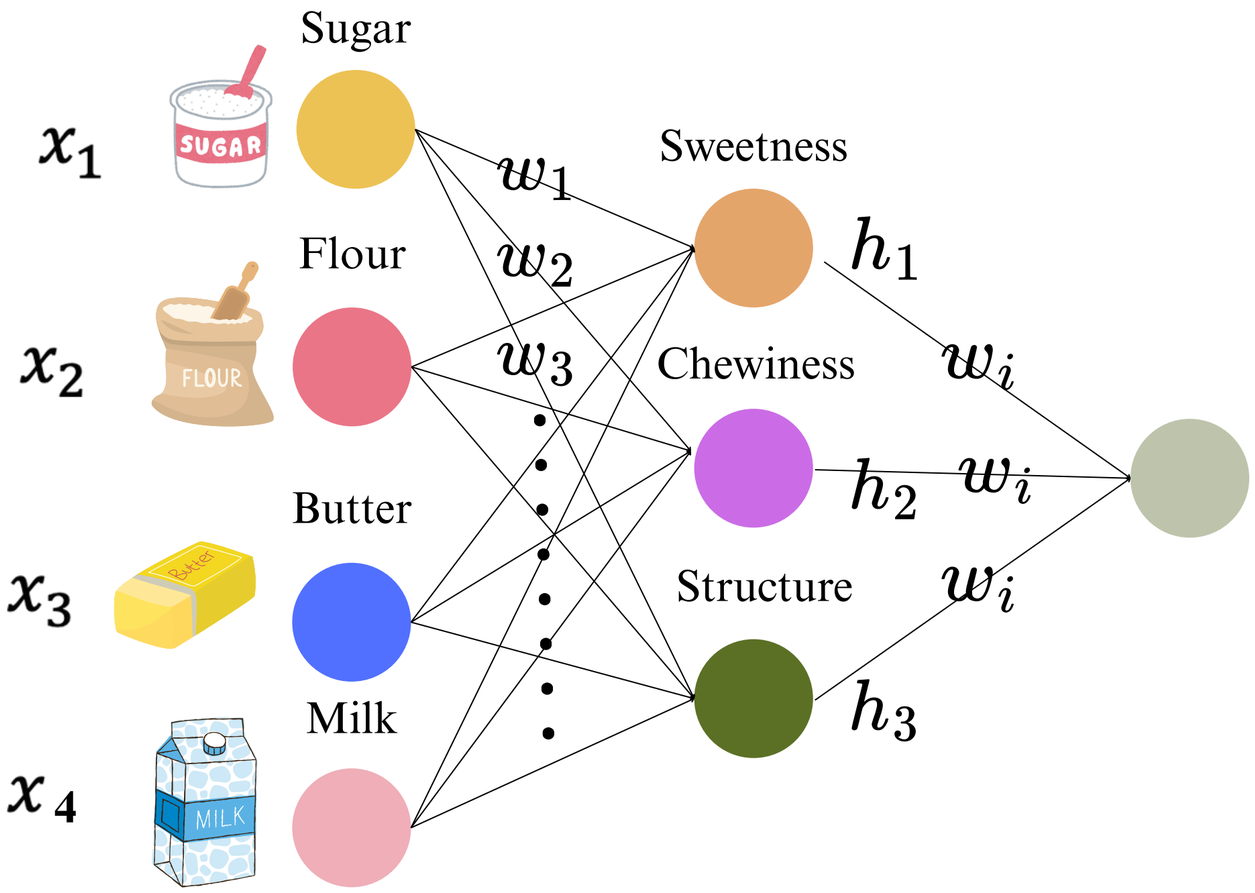
Artificial Neural Networks Example



Artificial Neural Networks Example



Artificial Neural Networks Example



Step 1: Combine ingredients with weights

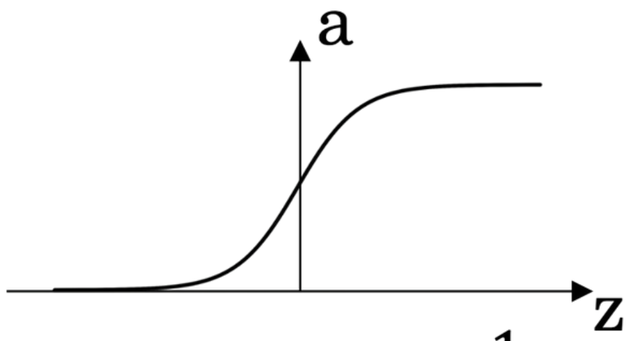
$$z_i = \sum_{j=1}^{4} w_{ij} x_j + b_i$$

Step 2: Pass through an activation function

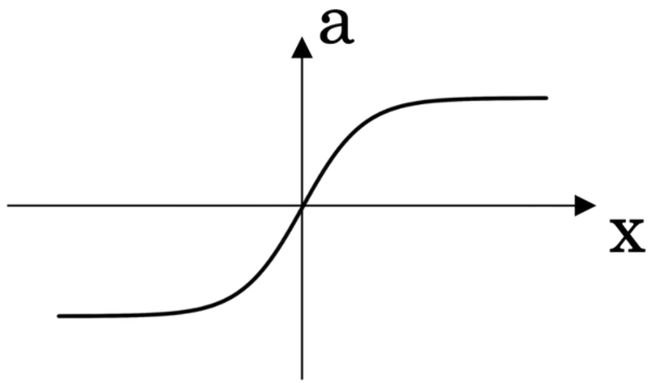
What's activation functions?

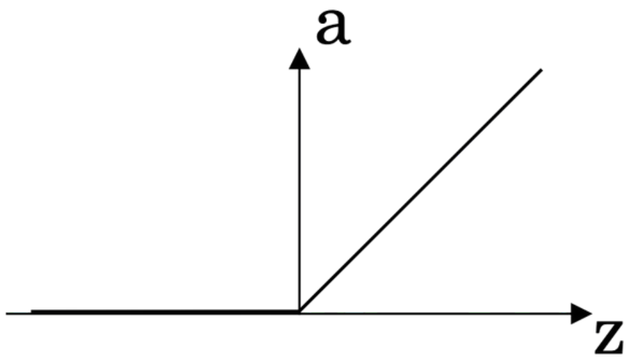
Activation Functions

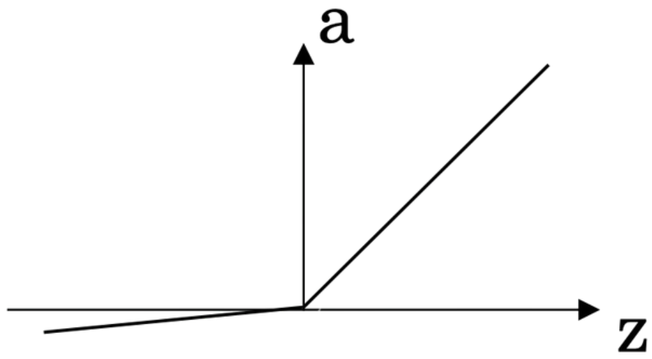
- Introduce nonlinearity
- Allow the model to learn complex patterns
- Without them, the network becomes just linear regression



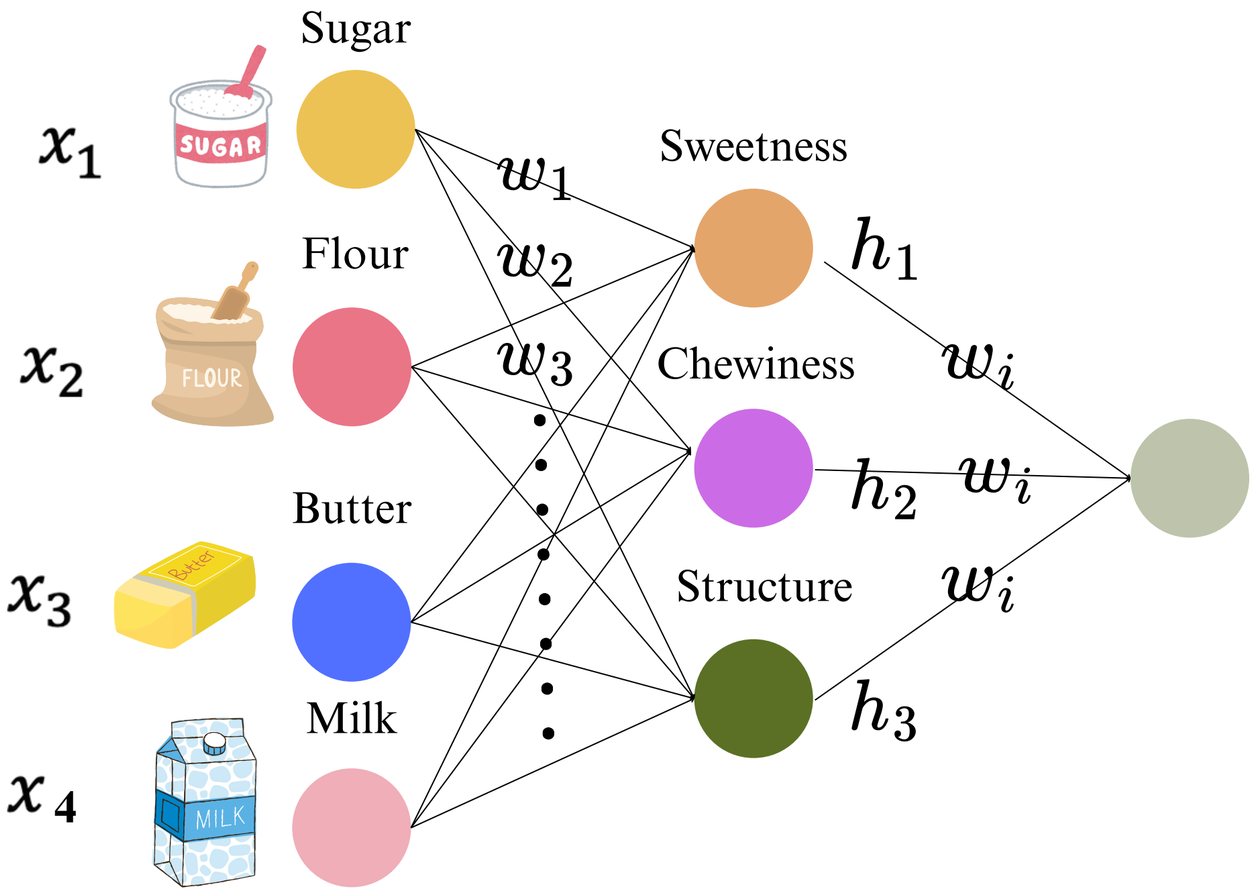
sigmoid: $a = \dfrac{1}{1 + e^{-z}}$







Artificial Neural Networks Example



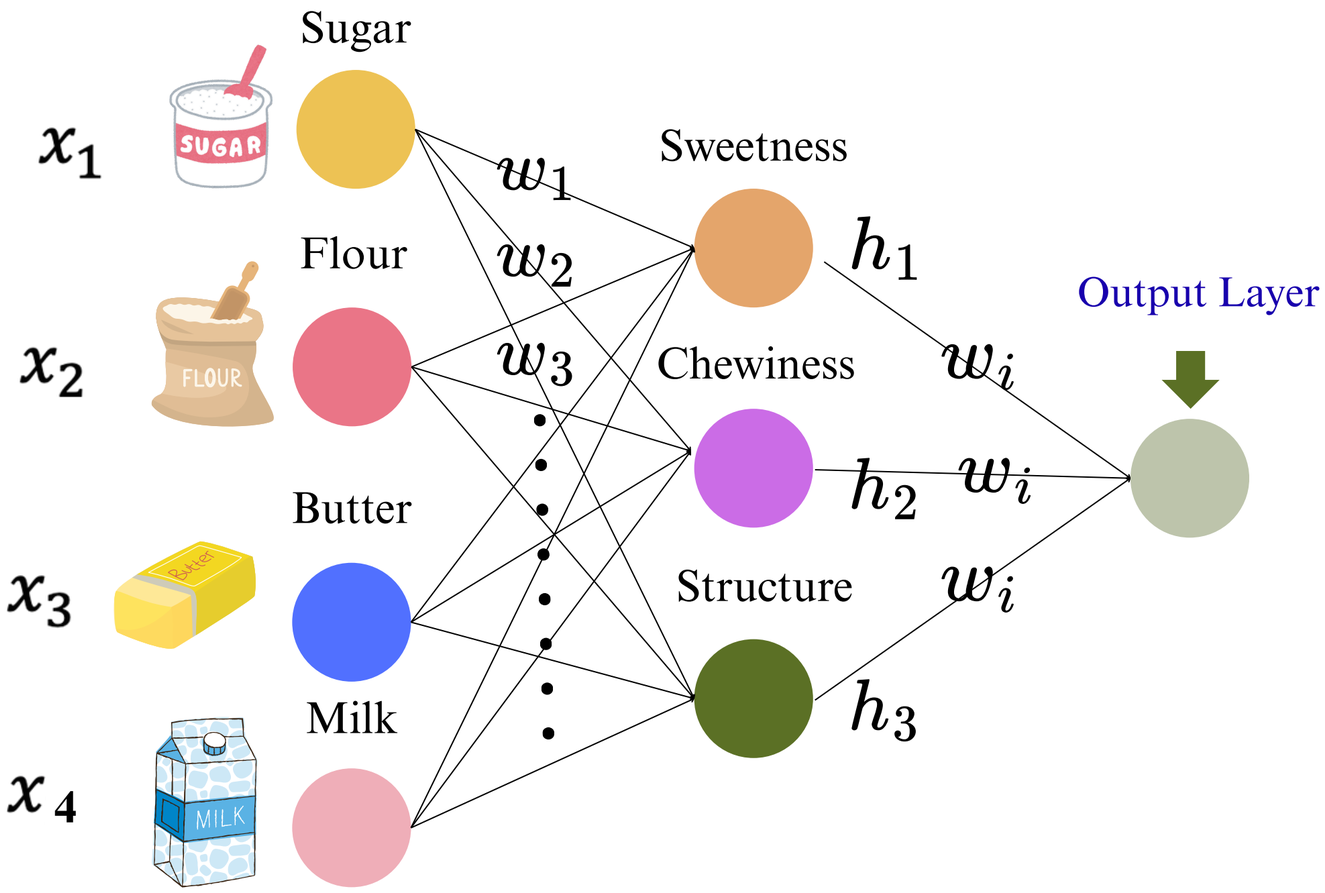
Step 1: Combine ingredients with weights

$$z_i = \sum_{j=1}^{4} w_{ij} x_j + b_i$$

Step 2: Pass through an activation function

$$h_i = f(z_i)$$

Artificial Neural Networks Example



Step 1: Combine ingredients with weights

$$z_i = \sum_{j=1}^{4} w_{ij} x_j + b_i$$

Step 2: Pass through an activation function

$$h_i = f(z_i)$$

Step 3: Hidden layer calculation

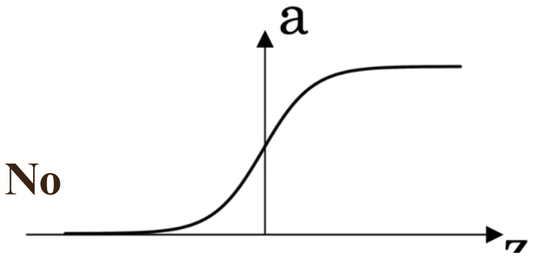
$$z_i = \sum_{j=1}^{4} w_{ij} x_j + b_i$$

$$h_i = f(z_i)$$

Step 4: Output layer calculation

$$z_{\text{out}} = w_{o1} h_1 + w_{o2} h_2 + w_{o3} h_3 + b_o$$

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



Artificial Neural Networks Example

Input Layer

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \text{Sugar} \\ \text{Flour} \\ \text{Butter} \\ \text{Milk} \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 2 \\ 4 \end{bmatrix}$$

Artificial Neural Networks Example

Input Layer

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \text{Sugar} \\ \text{Flour} \\ \text{Butter} \\ \text{Milk} \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 2 \\ 4 \end{bmatrix} \qquad W^{(1)} = \begin{bmatrix} 0.6 & 0.1 & 0.0 & 0.5 \\ 0.2 & 0.4 & 0.7 & 0.1 \\ 0.1 & 0.8 & 0.3 & 0.2 \end{bmatrix} \qquad b^{(1)} = \begin{bmatrix} 0.3 \\ 0.1 \\ 0.2 \end{bmatrix}$$

Artificial Neural Networks Example

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \text{Sugar} \\ \text{Flour} \\ \text{Butter} \\ \text{Milk} \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 2 \\ 4 \end{bmatrix} \quad W^{(1)} = \begin{bmatrix} 0.6 & 0.1 & 0.0 & 0.5 \\ 0.2 & 0.4 & 0.7 & 0.1 \\ 0.1 & 0.8 & 0.3 & 0.2 \end{bmatrix} \quad b^{(1)} = \begin{bmatrix} 0.3 \\ 0.1 \\ 0.2 \end{bmatrix}$$

$$z^{(1)} = \begin{bmatrix} ? \\ ? \\ ? \end{bmatrix}$$

Artificial Neural Networks Example

$$z_1 = 0.6(3) + 0.1(5) + 0(2) + 0.5(4) + 0.3$$
$$= 1.8 + 0.5 + 0 + 2 + 0.3 =$$

$$z^{(1)} = \begin{bmatrix} 4.6 \\ ? \\ ? \end{bmatrix}$$

Artificial Neural Networks Example

$$z_2 = 0.2(3) + 0.4(5) + 0.7(2) + 0.1(4) + 0.1$$
$$= 0.6 + 2 + 1.4 + 0.4 + 0.1 =$$

$$z^{(1)} = \begin{bmatrix} 4.6 \\ 4.5 \\ ? \end{bmatrix}$$

Artificial Neural Networks Example

$$z_3 = 0.1(3) + 0.8(5) + 0.3(2) + 0.2(4) + 0.2$$
$$= 0.3 + 4 + 0.6 + 0.8 + 0.2 =$$

$$z^{(1)} = \begin{bmatrix} 4.6 \\ 4.5 \\ 5.9 \end{bmatrix}$$

Artificial Neural Networks Example

$$z^{(1)} = \begin{bmatrix} 4.6 \\ 4.5 \\ 5.9 \end{bmatrix} \quad \sigma(z) = \frac{1}{1+e^{-z}} \quad \sigma(z) = \begin{bmatrix} 0.99 \\ 0.989 \\ 0.997 \end{bmatrix}$$

Artificial Neural Networks Example

$$\sigma(z) = \begin{bmatrix} 0.99 \\ 0.989 \\ 0.997 \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} 0.7 & 0.5 & 0.9 \end{bmatrix}$$

$$b^{(2)} = 0.2$$

$$z^{(2)} = W^{(2)}h + b^{(2)}$$

$$z^{(2)} = 0.7(0.99) + 0.5(0.989) + 0.9(0.997) + 0.2$$

$$= 0.693 + 0.4945 + 0.8973 + 0.2 = 2.2848$$

Artificial Neural Networks Example

$$z^{(2)} = W^{(2)}h + b^{(2)}$$

$$z^{(2)} = 0.7(0.99) + 0.5(0.989) + 0.9(0.997) + 0.2$$

$$= 0.693 + 0.4945 + 0.8973 + 0.2 = 2.2848$$

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

$$\hat{y} = 0.907$$

The model thinks the cookie is good

Artificial Neural Networks Example

$$z^{(2)} = W^{(2)}h + b^{(2)}$$

$$z^{(2)} = 0.7(0.99) + 0.5(0.989) + 0.9(0.997) + 0.2$$

$$= 0.693 + 0.4945 + 0.8973 + 0.2 = 2.2848$$

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

$$\hat{y} = 0.907$$

The model thinks the cookie is good

Is that True?

Artificial Neural Networks Example

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

$$\hat{y} = 0.907$$

The model thinks the cookie is good

Is that True?

$$\hat{y} \geq 0.5 \Rightarrow \text{class} = 1 \text{ (Good cookie)}$$

$$\hat{y} < 0.5 \Rightarrow \text{class} = 0 \text{ (Bad cookie)}$$

Is that True??

$\hat{y}$

y

We need a way to measure how wrong the prediction is.

Artificial Neural Networks Example

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

$$\hat{y} = 0.907$$

The model thinks the cookie is good

Is that True?

Is that True??

$\hat{y}$

y

We need a way to measure how wrong the prediction is.

Loss function

Loss Functions

- A loss function measures how far the model's prediction (y^) is from the true value (y).
- It quantifies the error so the model knows how much it needs to improve.

Loss Function	Applicability to Classification	Applicability to Regression
Mean Square Error (MSE) / L2 Loss	✗	✓
Mean Absolute Error (MAE) / L1 Loss	✗	✓
Binary Cross-Entropy Loss / Log Loss	✓	✗
Categorical Cross-Entropy Loss	✓	✗
Hinge Loss	✓	✗
Huber Loss / Smooth Mean Absolute Error	✗	✓

Artificial Neural Networks Example

Binary Cross-Entropy Loss

$$L = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

$$y = 0, \quad \hat{y} = 0.907$$

$$L = -\log(1 - 0.907) = -\log(0.093) = 2.375$$

The loss is HIGH ↑

Artificial Neural Networks Example

Binary Cross-Entropy Loss

$$L = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

$$y = 0, \quad \hat{y} = 0.907$$

$$L = -\log(1 - 0.907) = -\log(0.093) = 2.375$$

The loss is HIGH ↑

We're done… right?

Backpropagation

Backpropagation

- Backpropagation is the algorithm used to compute how much each weight in a neural network contributed to the overall error, and then update those weights to minimize the loss.

Forward Pass

- Compute predictions using current weights

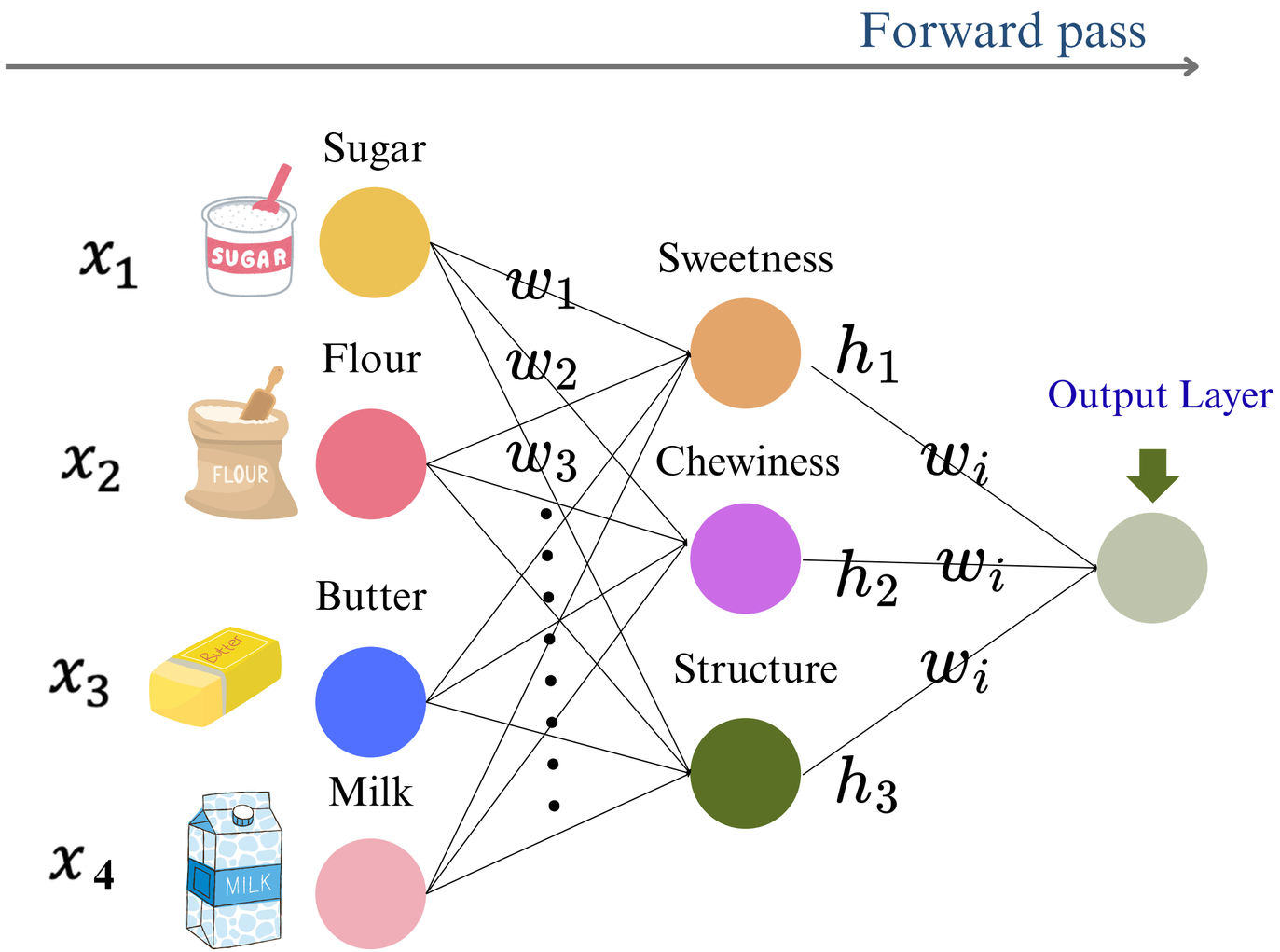
Compute Loss

- Measure how wrong the prediction is

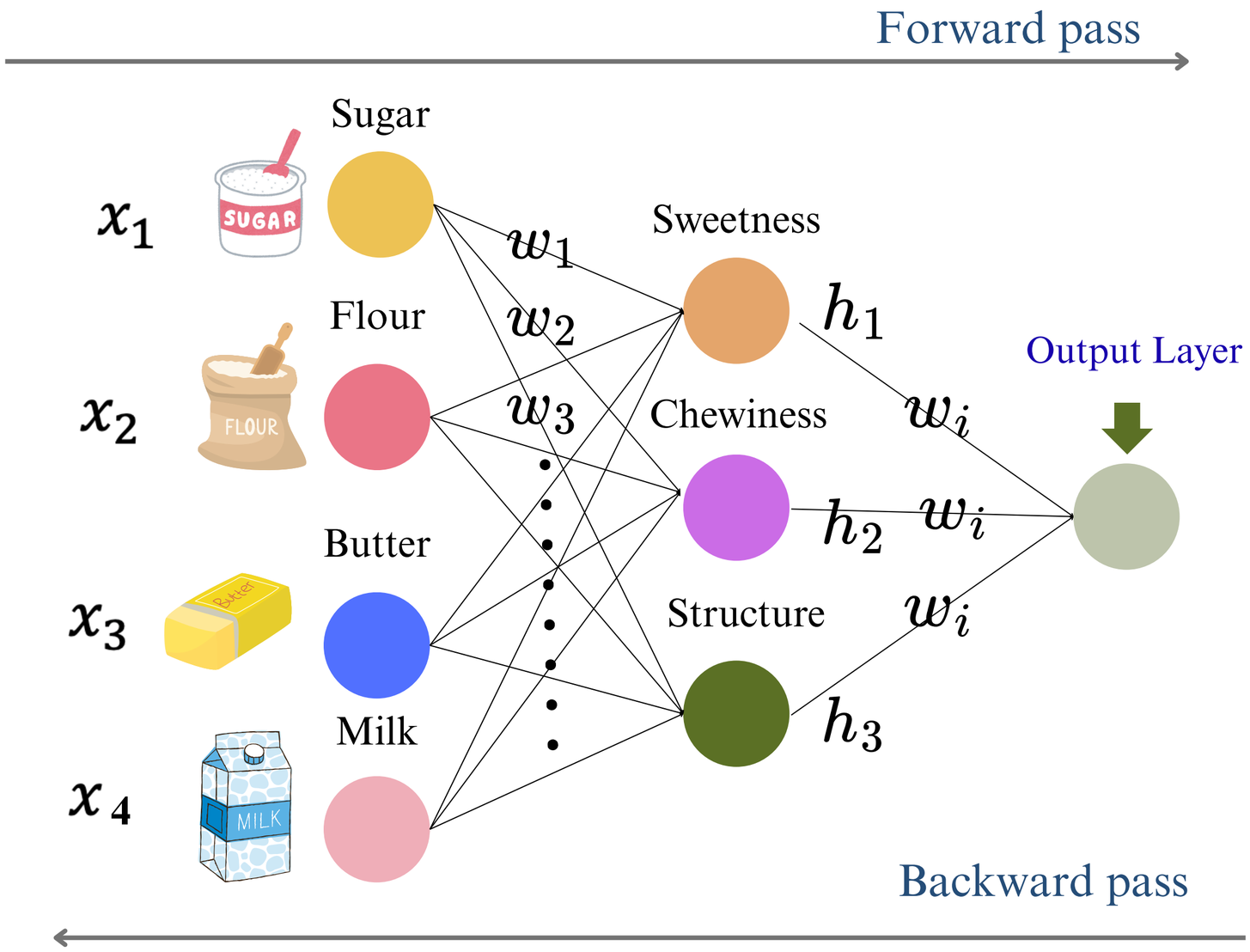
Backward Pass (Backprop)

- Compute gradients
- Update Weights (Gradient Descent)
- Repeat until the loss is small

Backpropagation



Backpropagation



Cost Function

- A cost function is an objective function that evaluates the performance of a model by computing the total error over the training dataset.

$$L(f(x^{(i)}), y^{(i)}) \sum_{i=1}^{m} \frac{1}{m} = J(w, b)$$

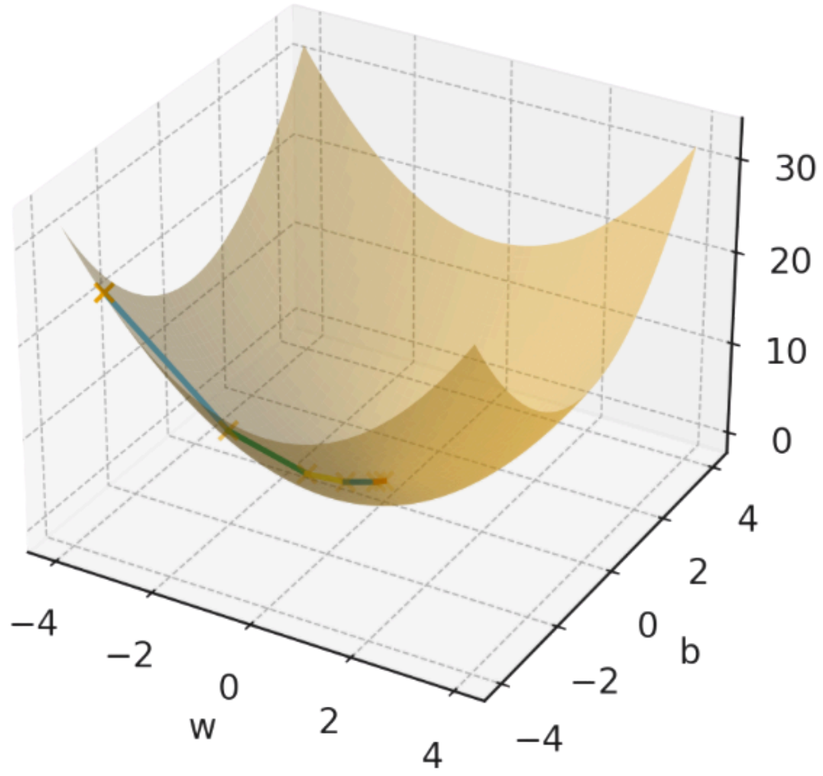
- Update these parameters step by step, gradually reducing the error.
- This process is called ***Gradient Descent***.

Cost Function

- The model starts with random parameter values (w, b)
- Computes the gradient (slope) of the cost function

$$w = w - \alpha \frac{\partial}{\partial w} J(w, b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w, b)$$



Cost Function

- If changing w by a small amount ε causes the cost J(w) to increase by k × ε, then the derivative is:

K= $\frac{d}{dw}J(w)$

function $J(w) = w^2$

varible $w = 3$:

function $J(w) = 9$

lets increase w by ϵ=0.001

w=*3.001*

J(w)=*9.006001*

Cost Function

function $J(w) = w^2$

varible $w = 3$:

function $J(w) = 9$

lets increase w by ϵ=0.001:

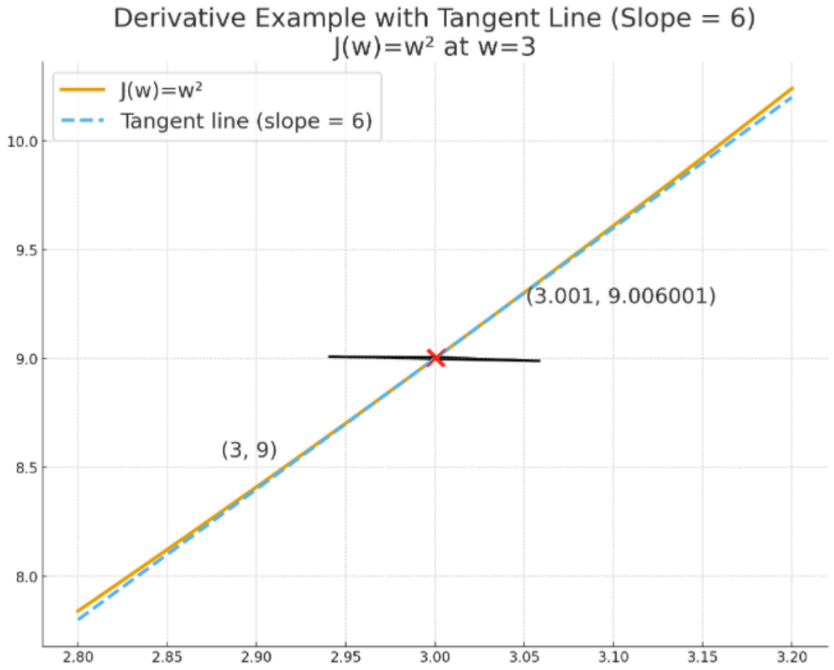
w=3.001

J(w)=9.006001

ΔJ≈0.006001

$$6 \times 0.001 = 0.006$$

$$\frac{d}{dw} J(w) = 2w = 6$$



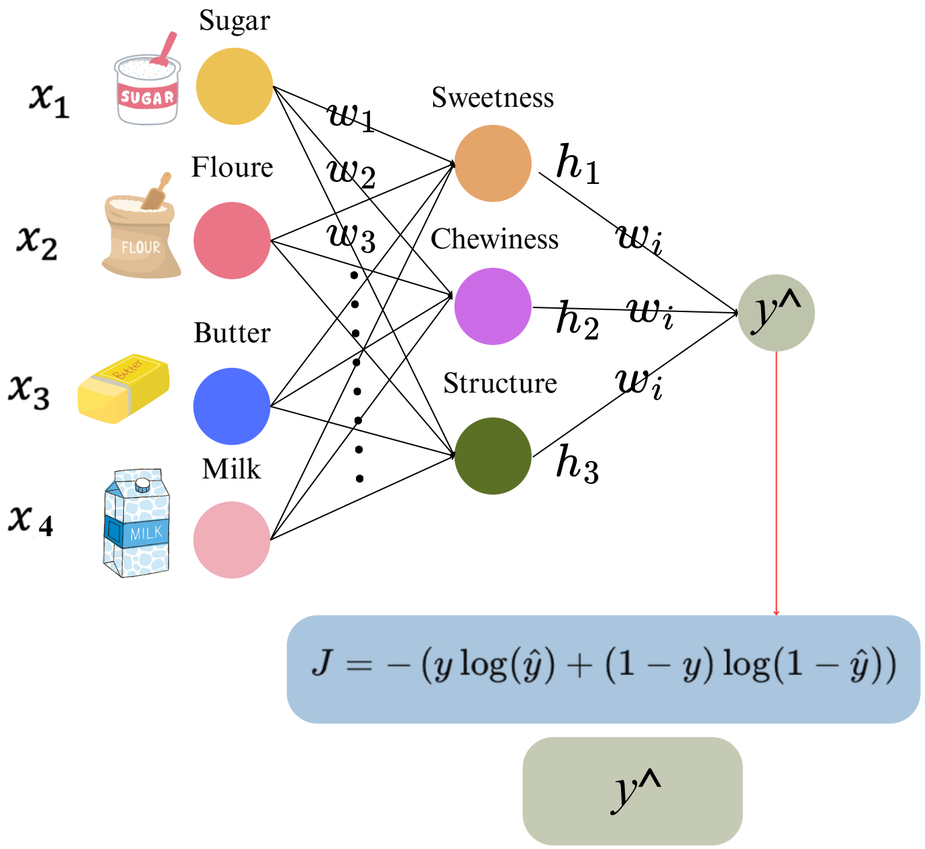
Cost Function

1-Derivative of Loss w.r.t. Prediction

$$\frac{\partial J}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

$$\frac{\partial J}{\partial \hat{y}} = -\frac{0}{0.907} + \frac{1}{1-0.907} = \frac{1}{0.093} \approx 10.75$$

If the prediction changes a bit, loss will change ~10.75× that bit."



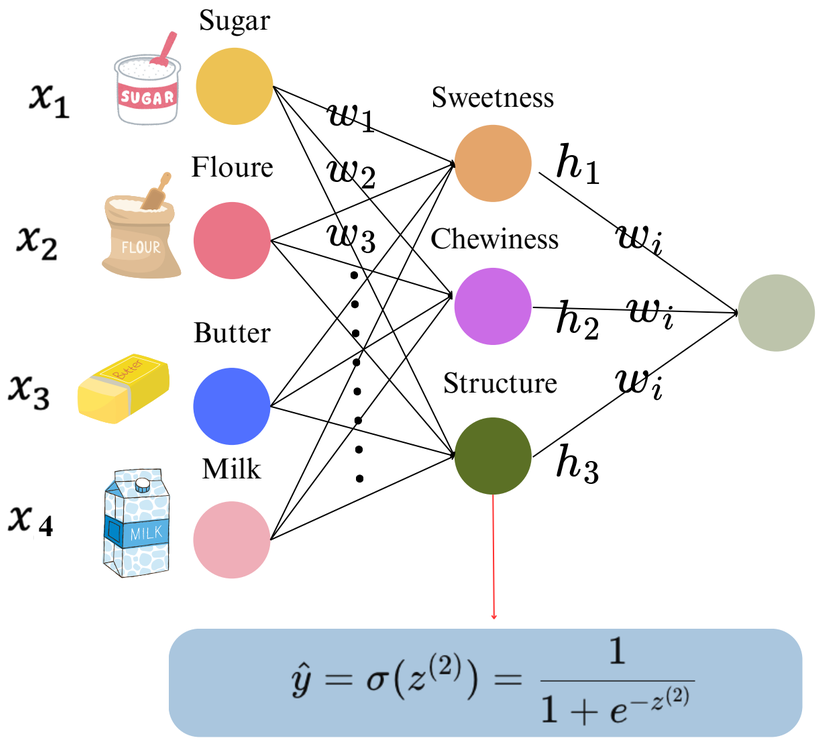
Cost Function

2 – Derivative of Sigmoid

$$\hat{y} = \sigma(z^{(2)}) = \frac{1}{1 + e^{-z^{(2)}}}$$

$$\frac{\partial \hat{y}}{\partial z^{(2)}} = \hat{y}(1 - \hat{y})$$

$$\frac{\partial \hat{y}}{\partial z^{(2)}} = 0.907(1 - 0.907) \approx 0.08435$$



Cost Function

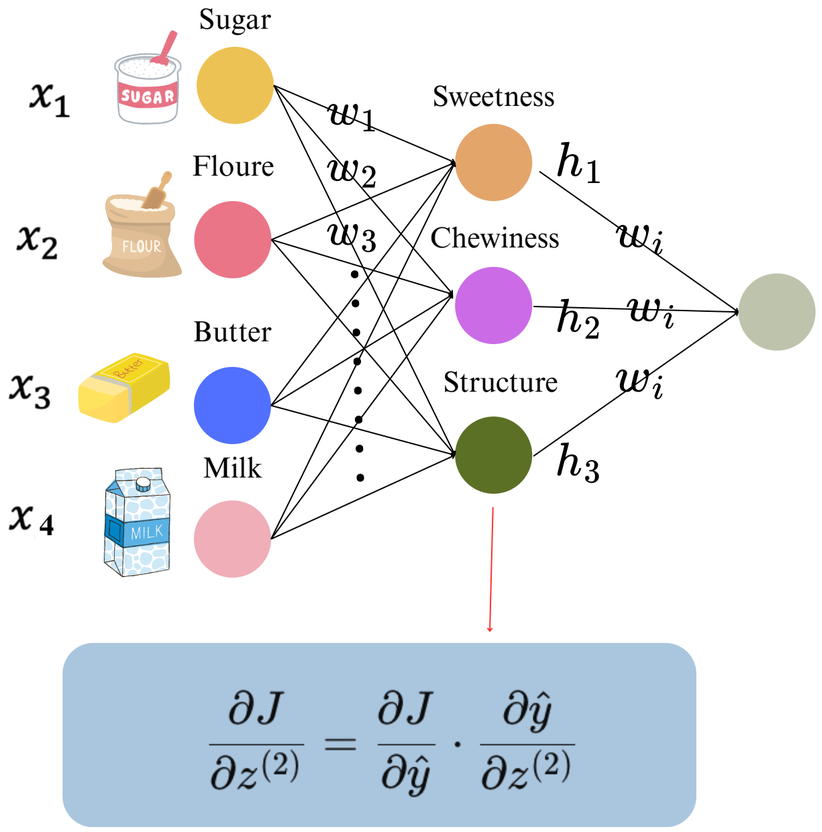
3- Derivative of Loss w.r.t z^2 (Chain Rule):

$$\frac{\partial J}{\partial z^{(2)}} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^{(2)}}$$

$$\frac{\partial J}{\partial z^{(2)}} \approx 10.75 \times 0.08435 \approx 0.907$$

sigmoid + binary cross-entropy

$$\frac{\partial J}{\partial z^{(2)}} = \hat{y} - y$$



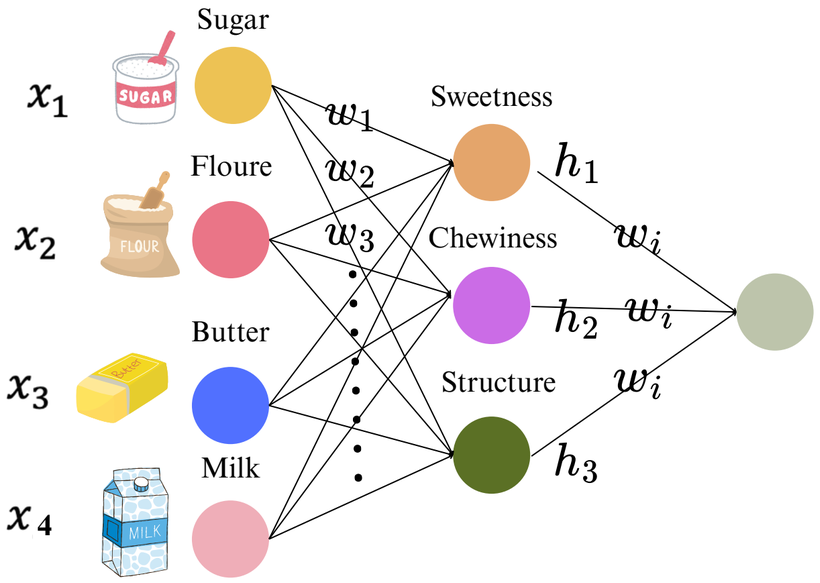
Cost Function

4-Gradients of Output Layer

$$z^{(2)} = W^{(2)}h + b^{(2)} = w_1^{(2)}h_1 + w_2^{(2)}h_2 + w_3^{(2)}h_3 + b^{(2)}$$

$$\frac{\partial z^{(2)}}{\partial w_j^{(2)}} = h_j, \quad \frac{\partial z^{(2)}}{\partial b^{(2)}} = 1$$

$$\frac{\partial J}{\partial w_j^{(2)}} = \frac{\partial J}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_j^{(2)}} = (\hat{y} - y)\, h_j$$



$$\frac{\partial J}{\partial w_j^{(2)}} = \frac{\partial J}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_j^{(2)}} = (\hat{y} - y)\, h_j$$

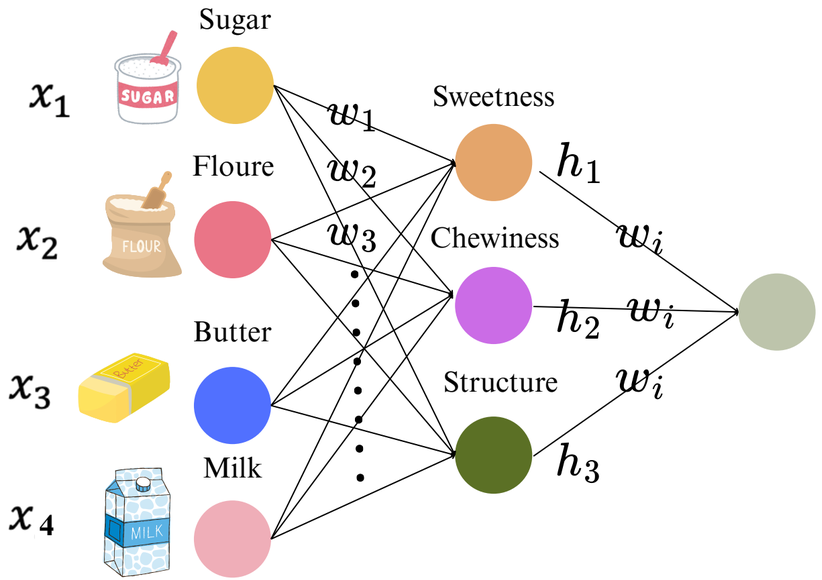
Cost Function

4-Gradients of Output Layer

$h \approx [0.99,\ 0.989,\ 0.997]^\top$:

- $\frac{\partial J}{\partial w_1^{(2)}} \approx 0.907 \times 0.99 = 0.898$
- $\frac{\partial J}{\partial w_2^{(2)}} \approx 0.907 \times 0.989 = 0.897$
- $\frac{\partial J}{\partial w_3^{(2)}} \approx 0.907 \times 0.997 = 0.904$

$$\frac{\partial J}{\partial b^{(2)}} = \frac{\partial J}{\partial z^{(2)}} \cdot 1 = 0.907$$



$$\frac{\partial J}{\partial w_j^{(2)}} = \frac{\partial J}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_j^{(2)}} = (\hat{y} - y)\, h_j$$

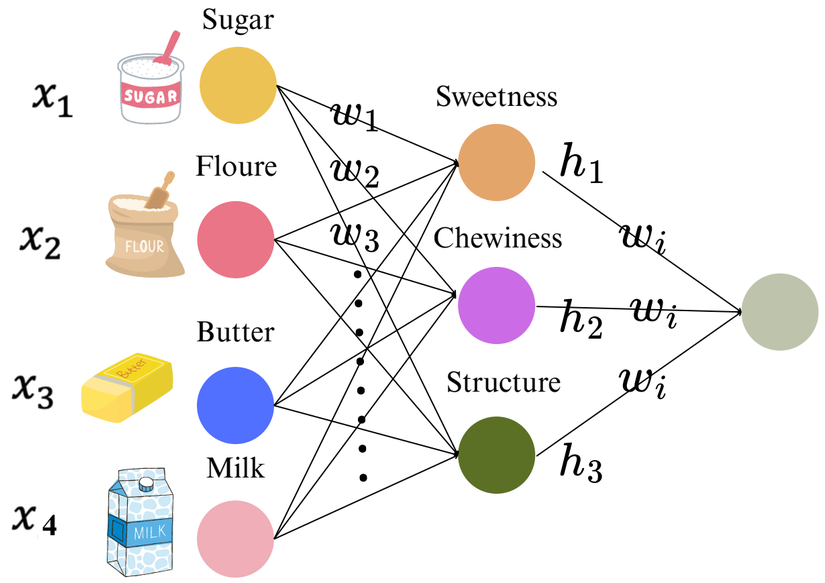
Cost Function

5 – Gradients w.r.t Hidden Activations h

$$z^{(2)} = w_1^{(2)} h_1 + w_2^{(2)} h_2 + w_3^{(2)} h_3 + b^{(2)}$$

$$\frac{\partial z^{(2)}}{\partial h_j} = w_j^{(2)}$$

$$\frac{\partial J}{\partial h_j} = \frac{\partial J}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial h_j} = (\hat{y} - y)\, w_j^{(2)}$$



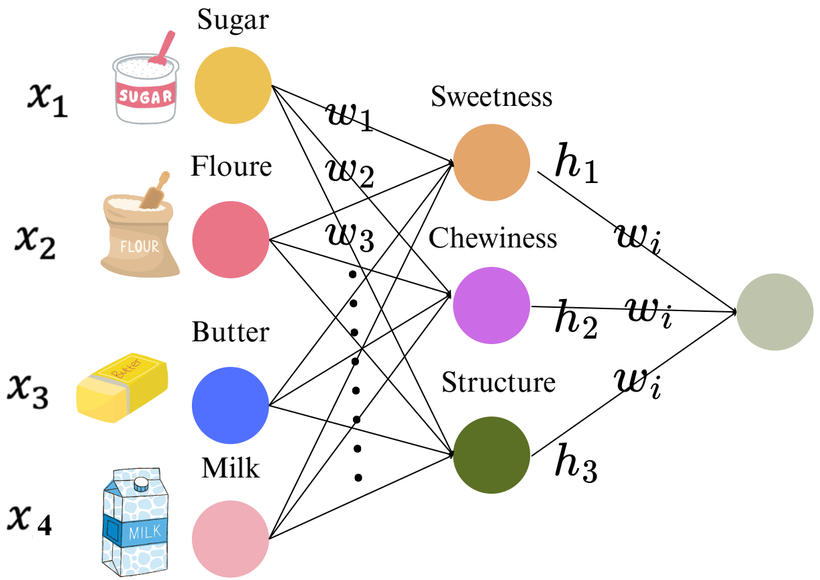
$$\frac{\partial J}{\partial w_j^{(2)}} = \frac{\partial J}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_j^{(2)}} =$$

Cost Function

5 – Gradients w.r.t Hidden Activations h

Using $\hat{y} - y = 0.907$ and $W^{(2)} = [0.7, 0.5, 0.9]$:

- $\frac{\partial J}{\partial h_1} \approx 0.907 \times 0.7 = 0.635$
- $\frac{\partial J}{\partial h_2} \approx 0.907 \times 0.5 = 0.454$
- $\frac{\partial J}{\partial h_3} \approx 0.907 \times 0.9 = 0.816$



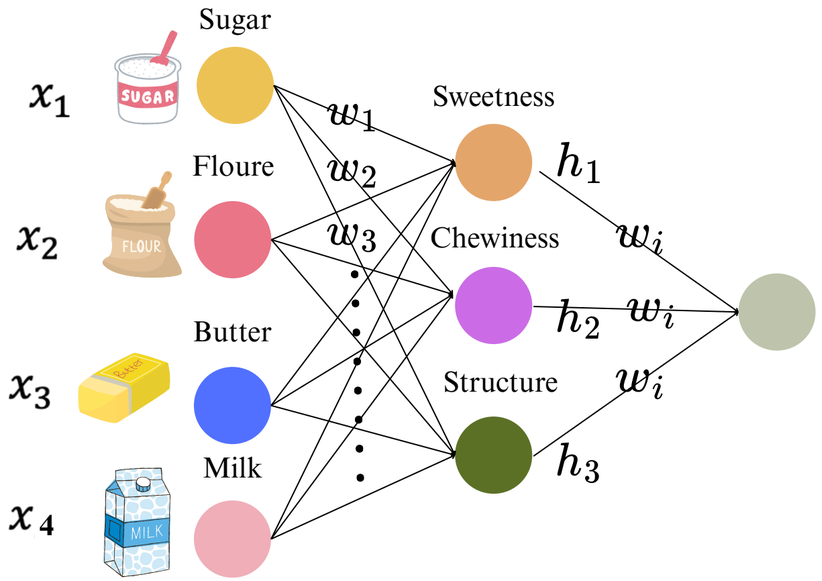
$$\frac{\partial J}{\partial w_j^{(2)}} = \frac{\partial J}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_j^{(2)}} :$$

Cost Function

Step 6 : Derivative of Hidden Sigmoid”

$$h_j = \sigma(z_j^{(1)}) \Rightarrow \frac{\partial h_j}{\partial z_j^{(1)}} = h_j(1 - h_j)$$

- $\frac{\partial J}{\partial z_1^{(1)}} \approx 0.635 \times 0.0099 = 0.00629$
- $\frac{\partial J}{\partial z_2^{(1)}} \approx 0.454 \times 0.01088 = 0.00493$
- $\frac{\partial J}{\partial z_3^{(1)}} \approx 0.816 \times 0.00299 = 0.00244$



$$\frac{\partial J}{\partial w_j^{(2)}} = \frac{\partial J}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_j^{(2)}}$$

Cost Function

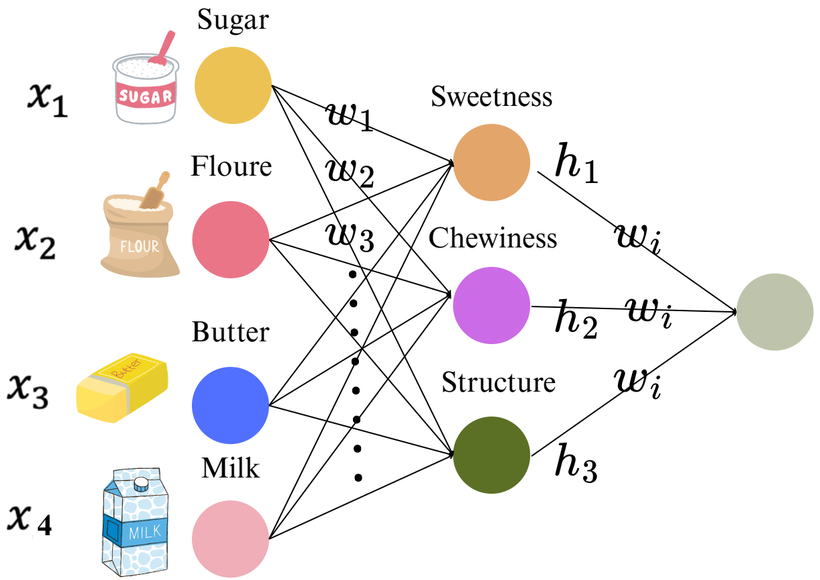
Step 7: Gradients w.r.t Hidden Weights

$$0.00244 = \frac{J\partial}{{}^{(1)}_{3}z\partial} \quad ,0.00493 = \frac{J\partial}{{}^{(1)}_{2}z\partial} \quad ,0.00629 = \frac{J\partial}{{}^{(1)}_{1}z\partial}$$

$$\begin{bmatrix} 3 \\ 5 \\ 2 \\ 4 \end{bmatrix}$$

$$\boxed{{}_{i}x \cdot \frac{J\partial}{{}^{(1)}_{j}z\partial} = \frac{J\partial}{{}^{(1)}_{ij}w\partial}}$$

$$\boxed{\frac{J\partial}{{}^{(1)}_{j}z\partial} = \frac{J\partial}{{}^{(1)}_{j}b\partial}}$$



$$\frac{\partial J}{\partial w_j^{(2)}} = \frac{\partial J}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_j^{(2)}} :$$

Cost Function

Step 7: Gradients w.r.t Hidden Weights

$$0.00244 = \frac{J\partial}{{}^{(1)}_{3}z\partial} \quad ,0.00493 = \frac{J\partial}{{}^{(1)}_{2}z\partial} \quad ,0.00629 = \frac{J\partial}{{}^{(1)}_{1}z\partial}$$

$$\begin{bmatrix} 3 \\ 5 \\ 2 \\ 4 \end{bmatrix}$$

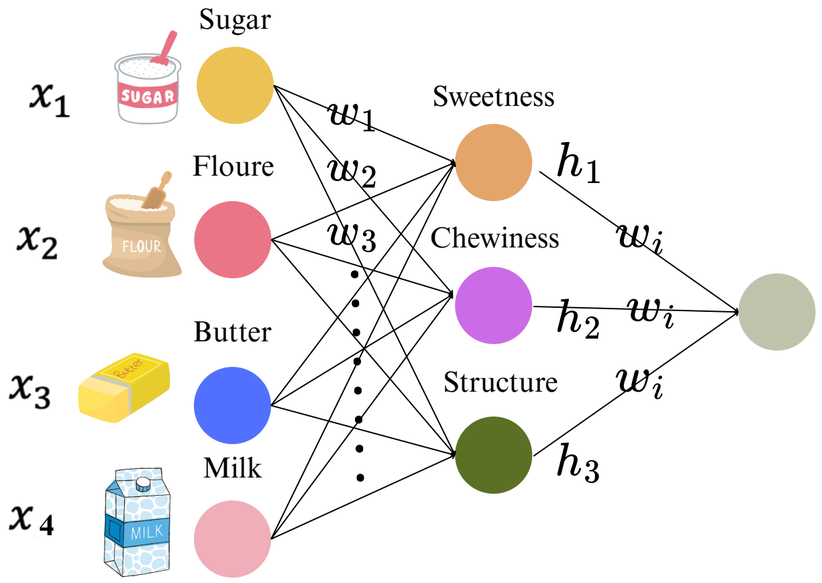
$$0.01887 = 3 \times 0.00629 = \frac{J\partial}{{}^{(1)}_{11}w\partial}$$

$$0.03145 = 5 \times 0.00629 = \frac{J\partial}{{}^{(1)}_{21}w\partial}$$

$$0.01258 = 2 \times 0.00629 = \frac{J\partial}{{}^{(1)}_{31}w\partial}$$

$$0.02516 = 4 \times 0.00629 = \frac{J\partial}{{}^{(1)}_{41}w\partial}$$

$$0.00629 = \frac{J\partial}{{}^{(1)}_{1}b\partial}$$



$$\frac{\partial J}{\partial w_j^{(2)}} = \frac{\partial J}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_j^{(2)}} :$$

Cost Function

Step 7: Gradients w.r.t Hidden Weights

$$0.00244 = \frac{J\partial}{{}^{(1)}_{3}z\partial} \quad ,0.00493 = \frac{J\partial}{{}^{(1)}_{2}z\partial} \quad ,0.00629 = \frac{J\partial}{{}^{(1)}_{1}z\partial}$$

$$\begin{bmatrix} 3 \\ 5 \\ 2 \\ 4 \end{bmatrix}$$

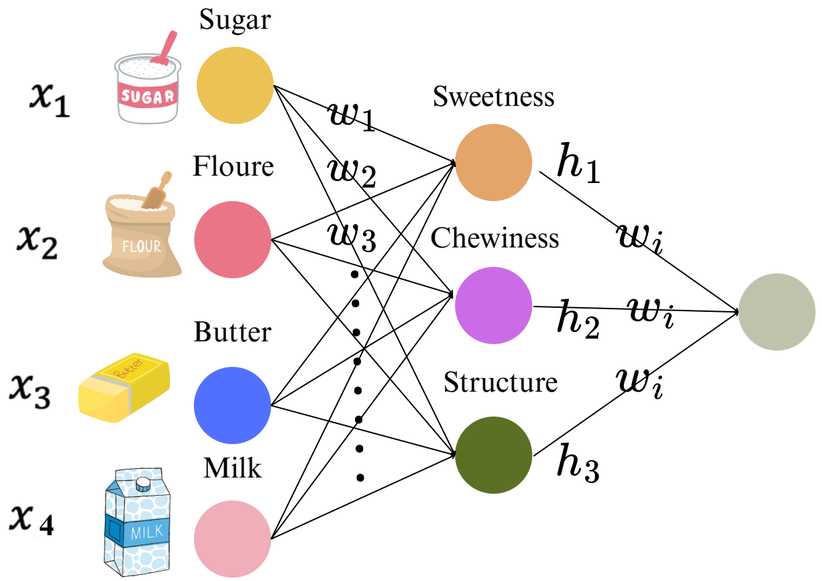
$$0.01479 = 3 \times 0.00493 = \frac{J\partial}{{}^{(1)}_{12}w\partial}$$

$$0.02465 = 5 \times 0.00493 = \frac{J\partial}{{}^{(1)}_{22}w\partial}$$

$$0.00986 = 2 \times 0.00493 = \frac{J\partial}{{}^{(1)}_{32}w\partial}$$

$$0.01972 = 4 \times 0.00493 = \frac{J\partial}{{}^{(1)}_{42}w\partial}$$

$$0.00493 = \frac{J\partial}{{}^{(1)}_{2}b\partial}$$



$$\frac{\partial J}{\partial w_j^{(2)}} = \frac{\partial J}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_j^{(2)}} :$$

Cost Function

Step 7: Gradients w.r.t Hidden Weights

$$0.00244 = \frac{J\partial}{{}^{(1)}_{3}z\partial} \quad ,0.00493 = \frac{J\partial}{{}^{(1)}_{2}z\partial} \quad ,0.00629 = \frac{J\partial}{{}^{(1)}_{1}z\partial}$$

$$\begin{bmatrix} 3 \\ 5 \\ 2 \\ 4 \end{bmatrix}$$

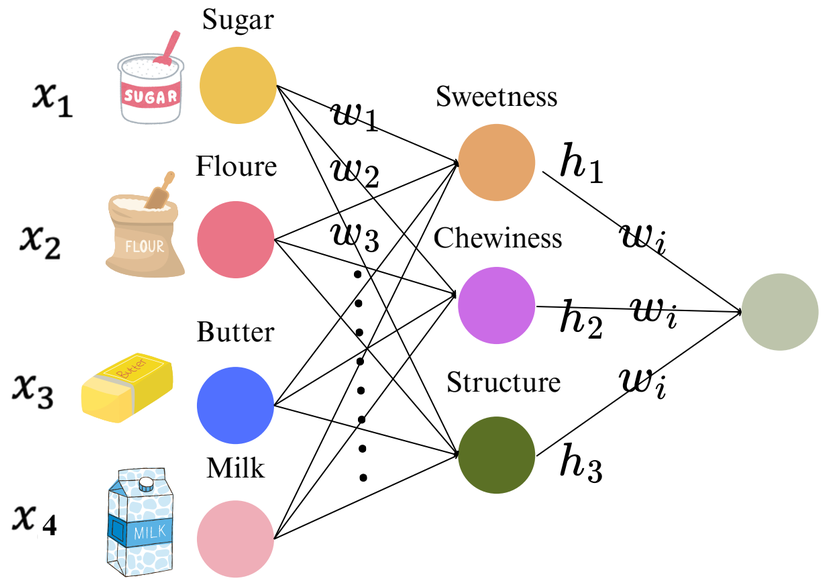
$$0.00732 = 3 \times 0.00244 = \frac{J\partial}{{}^{(1)}_{13}w\partial}$$

$$0.01220 = 5 \times 0.00244 = \frac{J\partial}{{}^{(1)}_{23}w\partial}$$

$$0.00488 = 2 \times 0.00244 = \frac{J\partial}{{}^{(1)}_{33}w\partial}$$

$$0.00976 = 4 \times 0.00244 = \frac{J\partial}{{}^{(1)}_{43}w\partial}$$

$$0.00244 = \frac{J\partial}{{}^{(1)}_{3}b\partial}$$



$$\frac{\partial J}{\partial w_j^{(2)}} = \frac{\partial J}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_j^{(2)}} :$$

Cost Function

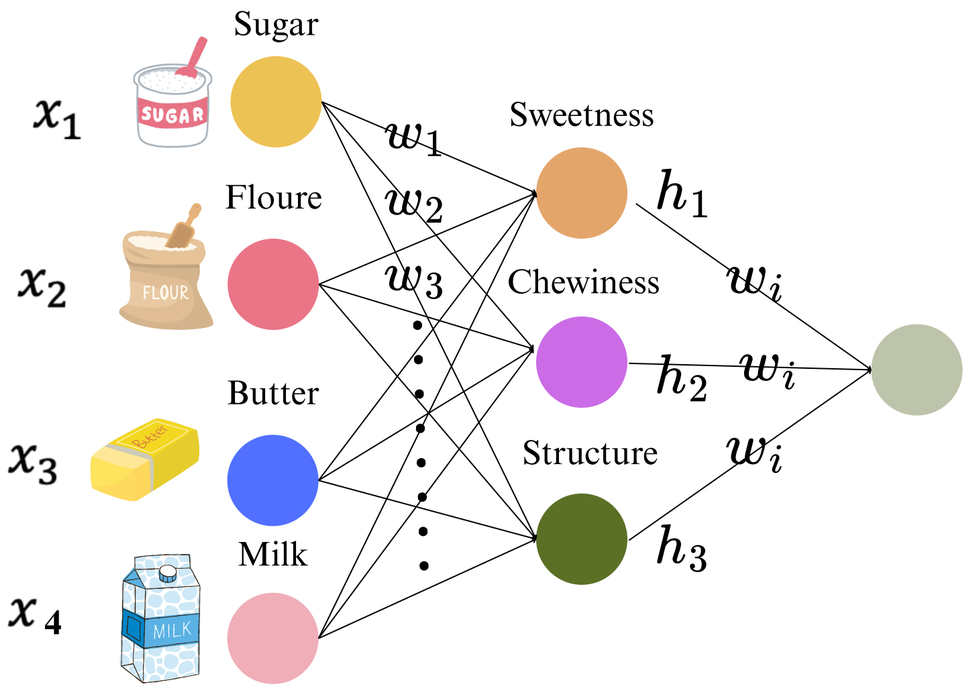
Final Update Rule (for all weights)

$$w_{\text{new}} = w_{\text{old}} - \alpha \frac{\partial J}{\partial w}$$

$$0.50 = {}^{(2)}_{1}w \qquad 0.898 = \frac{J\partial}{{}^{(2)}_{1}w\partial}$$

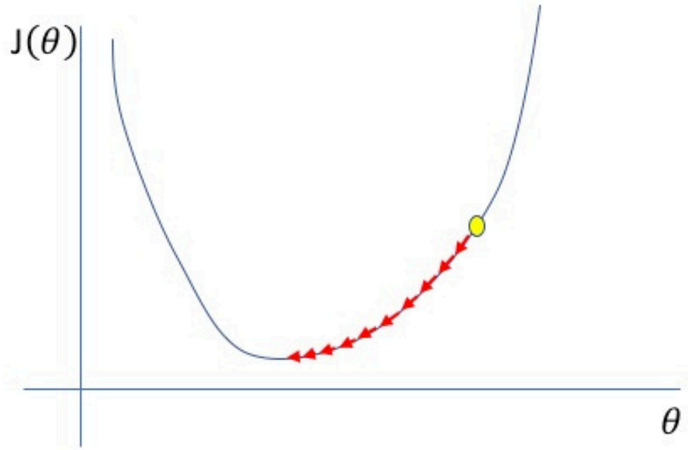
$$0.898 \times 0.1 - 0.50 = {}^{(2)}_{\text{new},1}w$$

$$\boxed{0.4102} = 0.0898 - 0.50 =$$



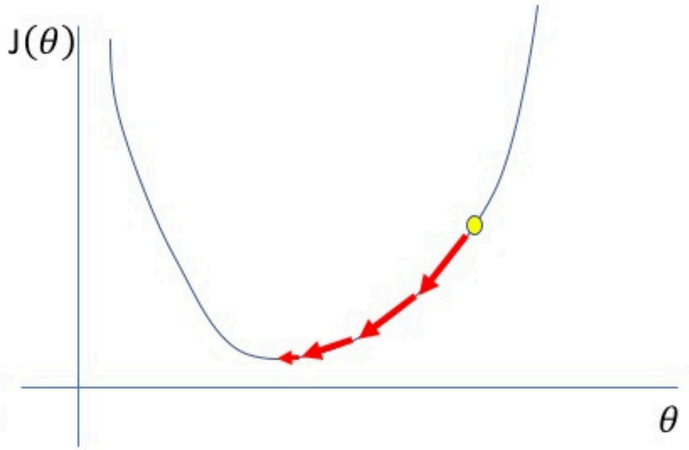
Learning Rate

Too low



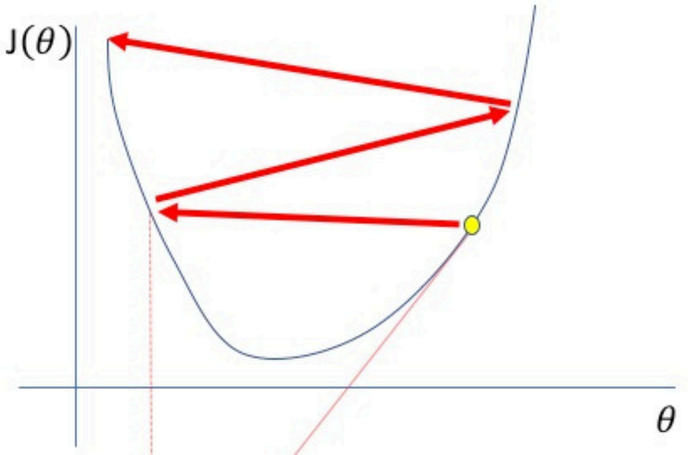
A small learning rate requires many updates before reaching the minimum point

Just right



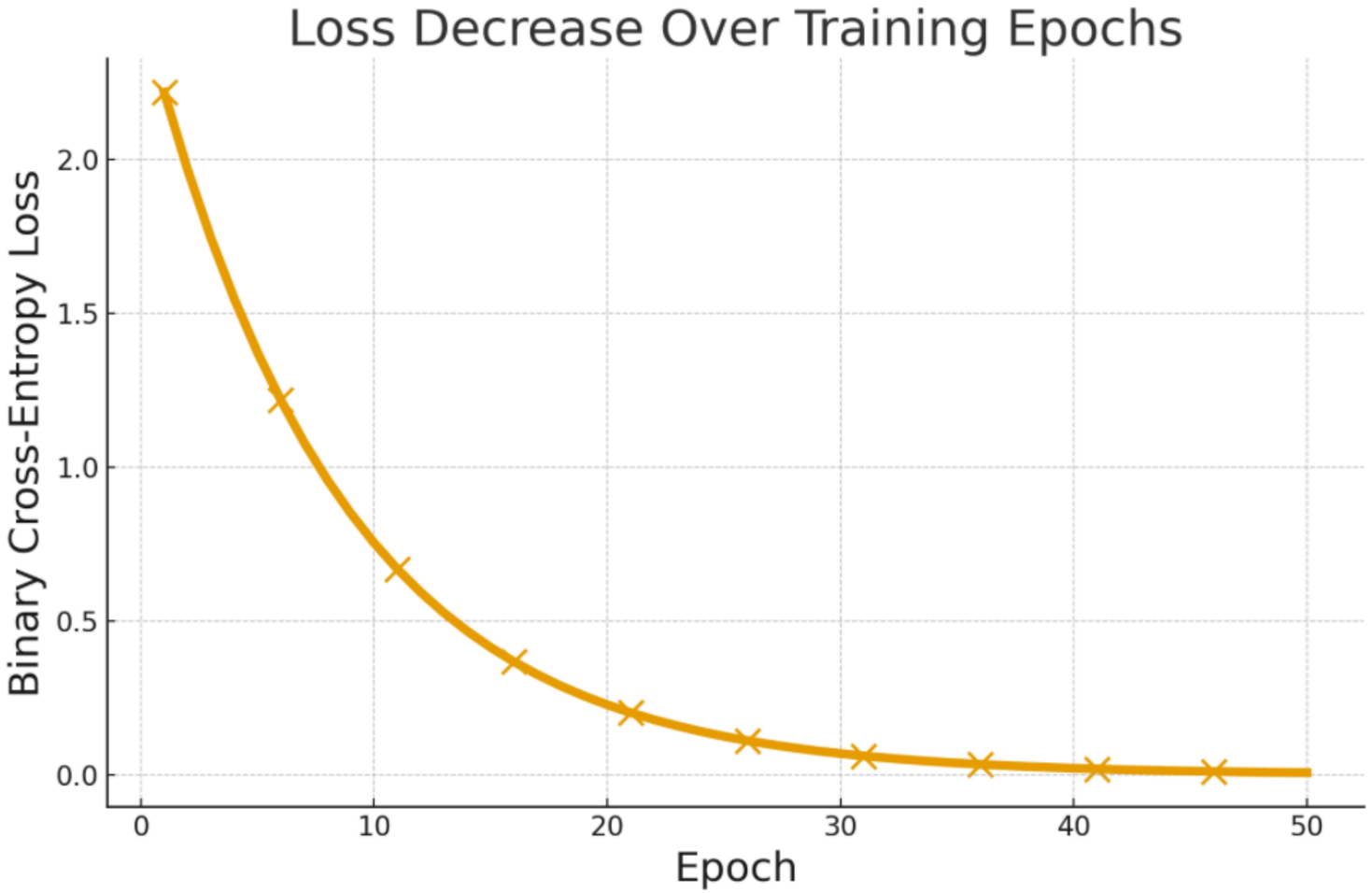
The optimal learning rate swiftly reaches the minimum point

Too high



Too large of a learning rate causes drastic updates which lead to divergent behaviors

Cost Function




Q
A

Thank You