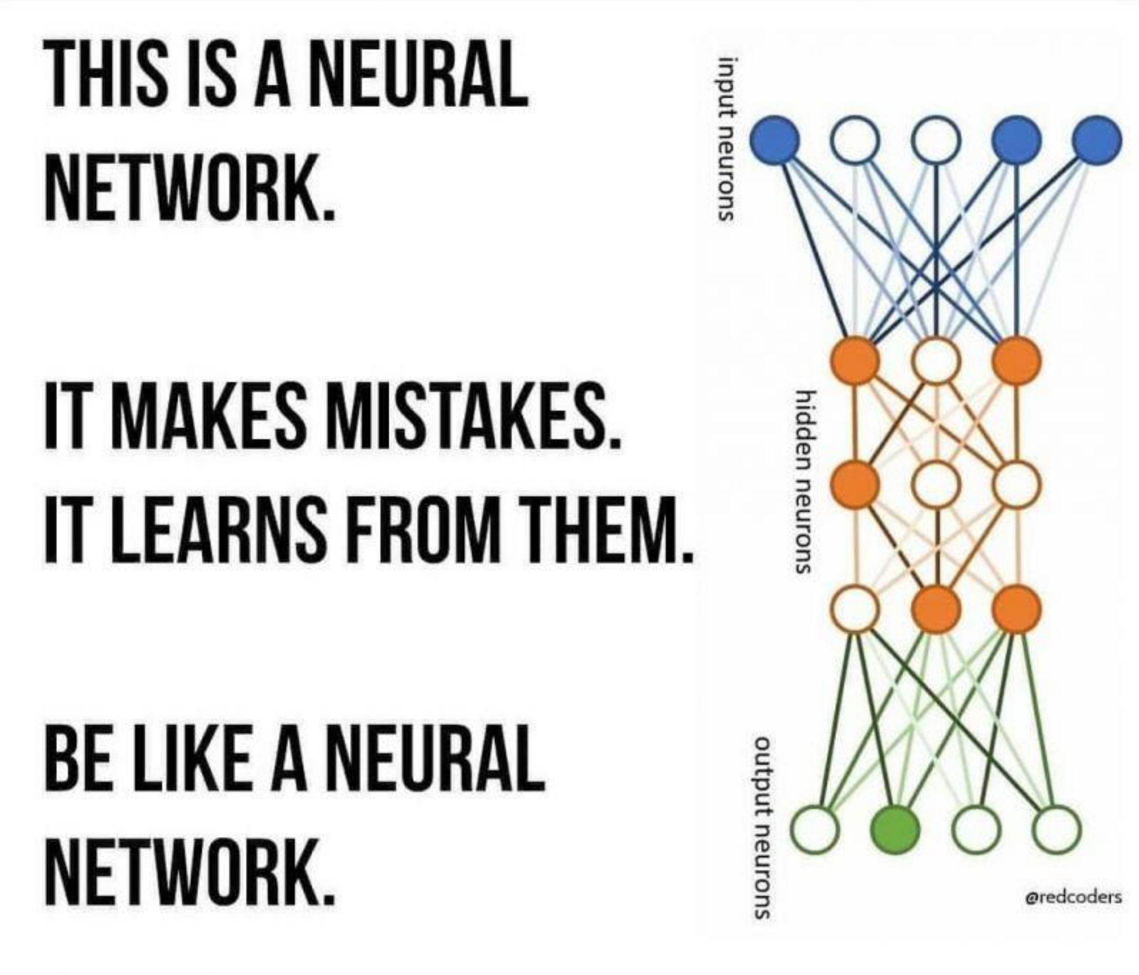


THIS IS A NEURAL NETWORK.

IT MAKES MISTAKES.
IT LEARNS FROM THEM.

BE LIKE A NEURAL NETWORK.

Welcome to Week 11 Lecture 1!

Data Science in Python & Machine Learning

Feature Engineering Review

What is one way you can engineer your features?

1. Extract month, week, day, hour, etc. from datetime
2.

Last Lecture Learning Goals

You are now able to:

1. Identify features for engineering
2. Select appropriate engineering strategies
3. Create non-linear feature combinations with PolynomialFeatures
4. Apply feature engineering to a dataset to improve model performance

Learning Goals

After this lesson you will be able to:

1. Summarize Forward Propagation

2. Explain how gradient descent relates to model learning

3. Visualize how a neural network learns to solve a problem.

4. Code a simple feed-forward neural network in Keras using densely connected layers.

Neural Networks are Like Brains

Neurons (nodes)
and connections (weights)

Image Source

Learn by Trial and Error

Neural Networks: Multi-Layered Perceptrons

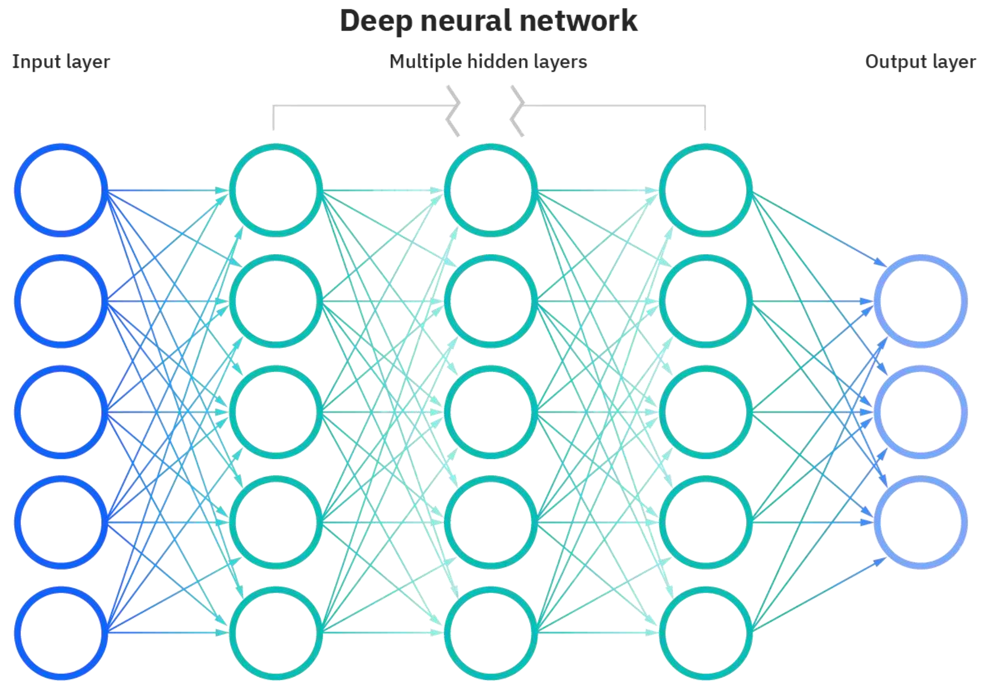


Image Source

Neural Networks Overview

- Possibly Many Layers
- Every layer discovers increasingly complex patterns
- Solve MANY different kinds of problems.

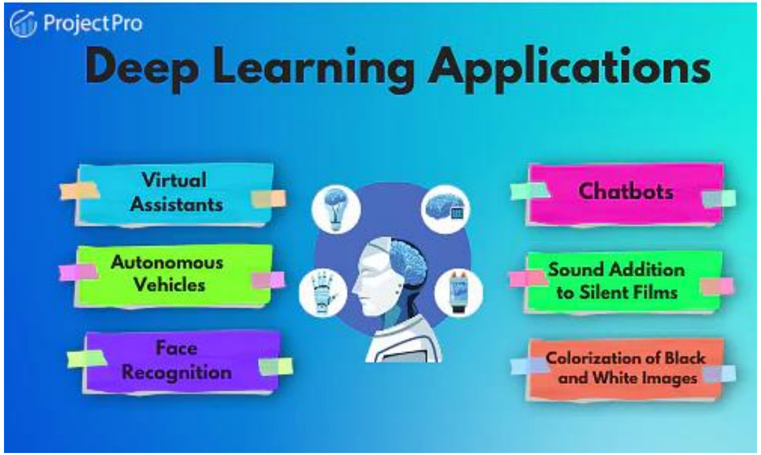
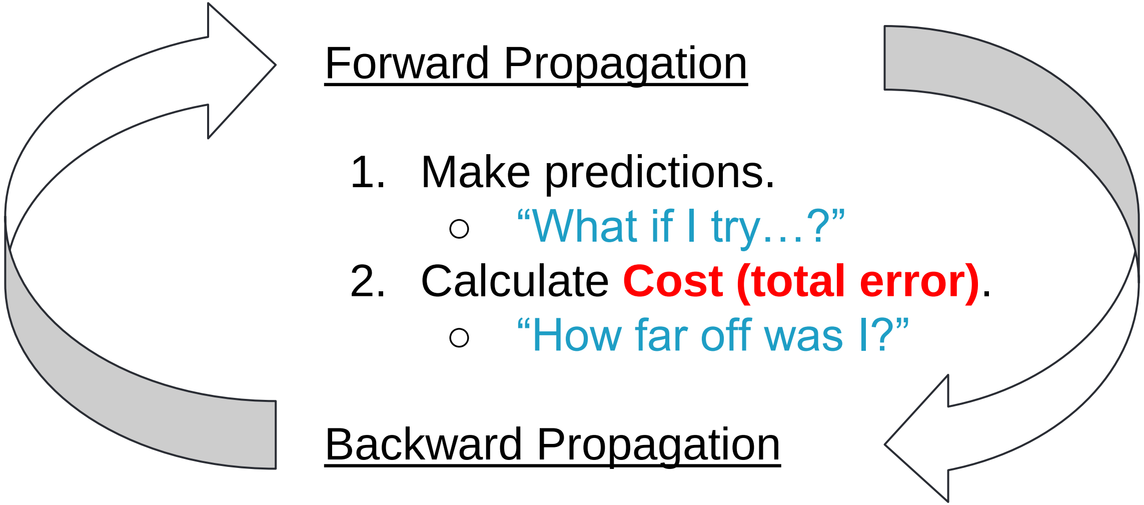

ProjectPro
Deep Learning Applications
Virtual Assistants
Autonomous Vehicles
Face Recognition
Chatbots
Sound Addition to Silent Films
Colorization of Black and White Images

Image Source

Iterative Learning



Forward Propagation

1. Make predictions.
   - ○ "What if I try…?"
2. Calculate **Cost (total error)**.
   - ○ "How far off was I?"

Backward Propagation

1. Change weights with **Gradient Descent**.
   a. "How can I do better?

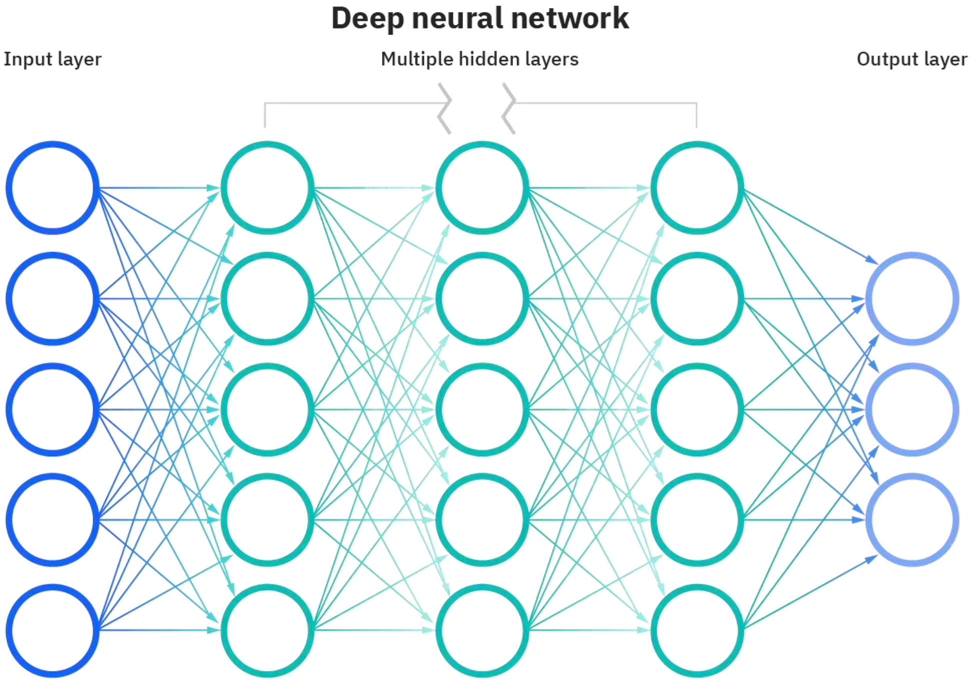
Repeat!

Forward Propagation

Features go in…

Data moves **forward** through the layers and is processed at each step.

The final output is the model prediction



…Prediction comes out

Image Source

Forward Propagation

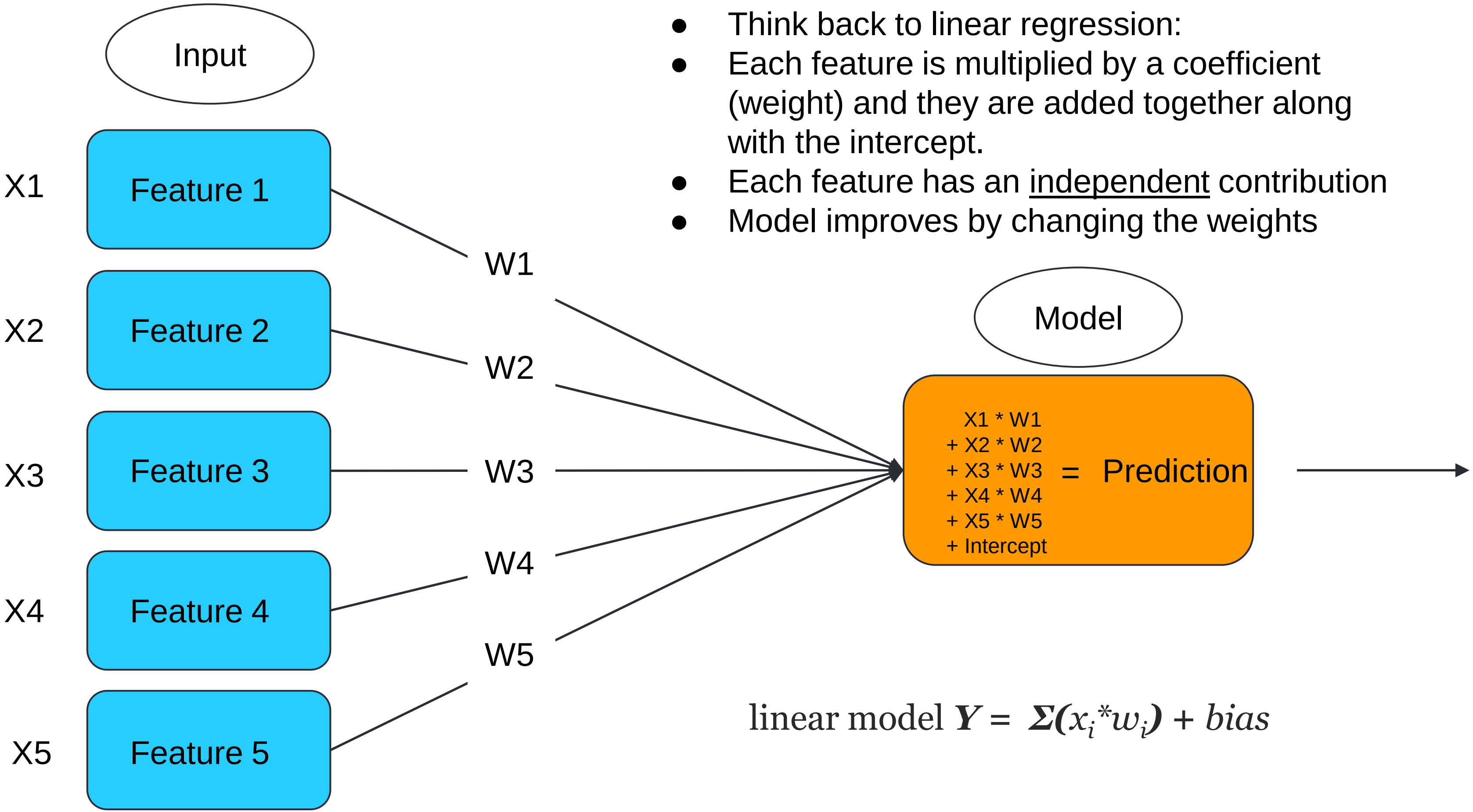
Does this formula look familiar?

$$\mathbf{y\_pred} = \mathbf{\Sigma(}x_i{*}w_i\mathbf{)} + \text{bias}$$

translation:

prediction = sum of weights times features, plus intercept (bias term)

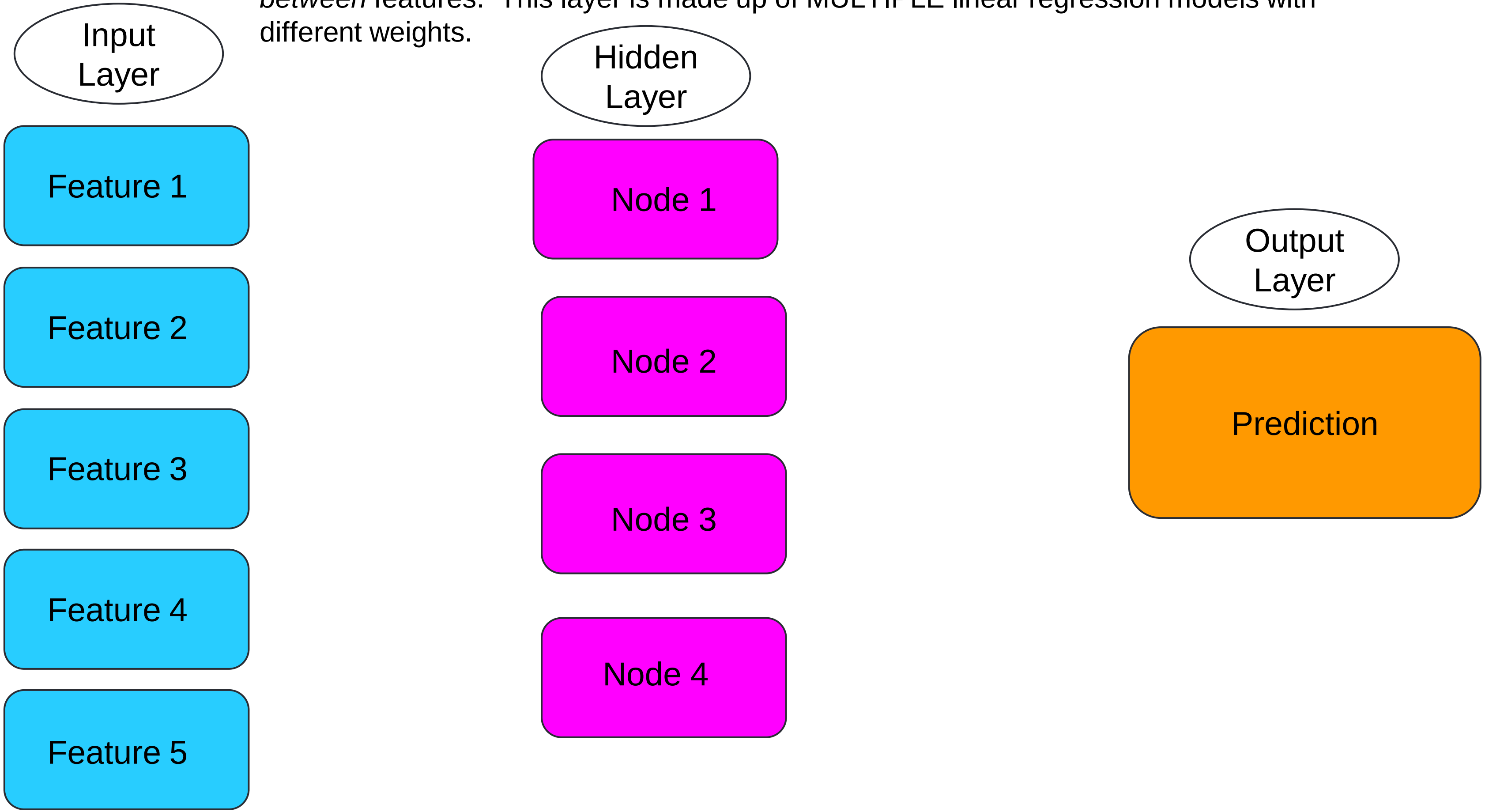
Image Source



- Think back to linear regression:
- Each feature is multiplied by a coefficient (weight) and they are added together along with the intercept.
- Each feature has an independent contribution
- Model improves by changing the weights

linear model $\boldsymbol{Y} = \boldsymbol{\Sigma(}x_i{*}w_i\boldsymbol{)} + \textit{bias}$

A neural network has (at least one) additional layer(s) that take into account *interactions between* features. This layer is made up of MULTIPLE linear regression models with different weights.



Input Layer

Feature 1

Feature 2

Feature 3

Feature 4

Feature 5

Hidden Layer

Node 1

Node 2

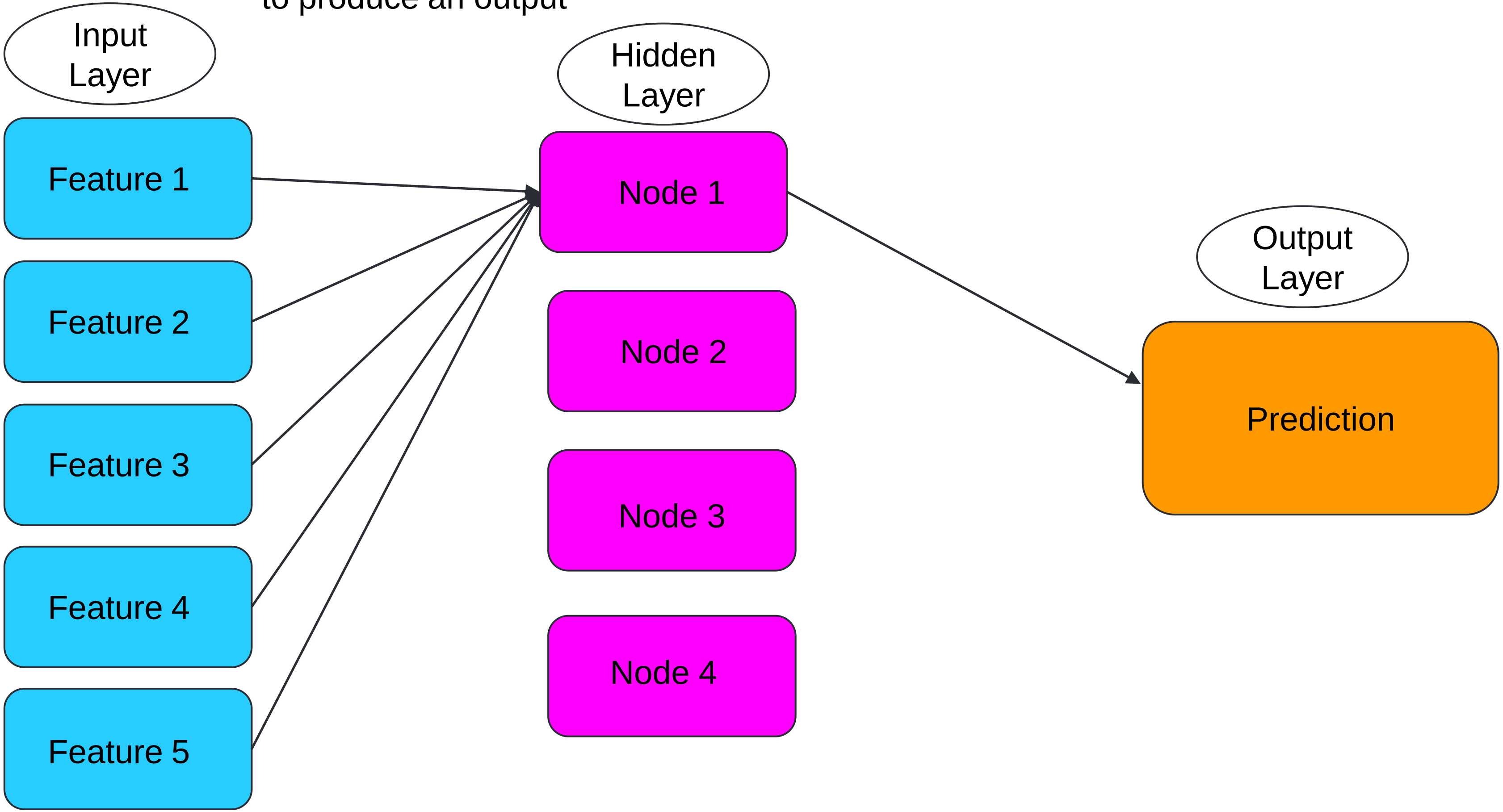
Node 3

Node 4

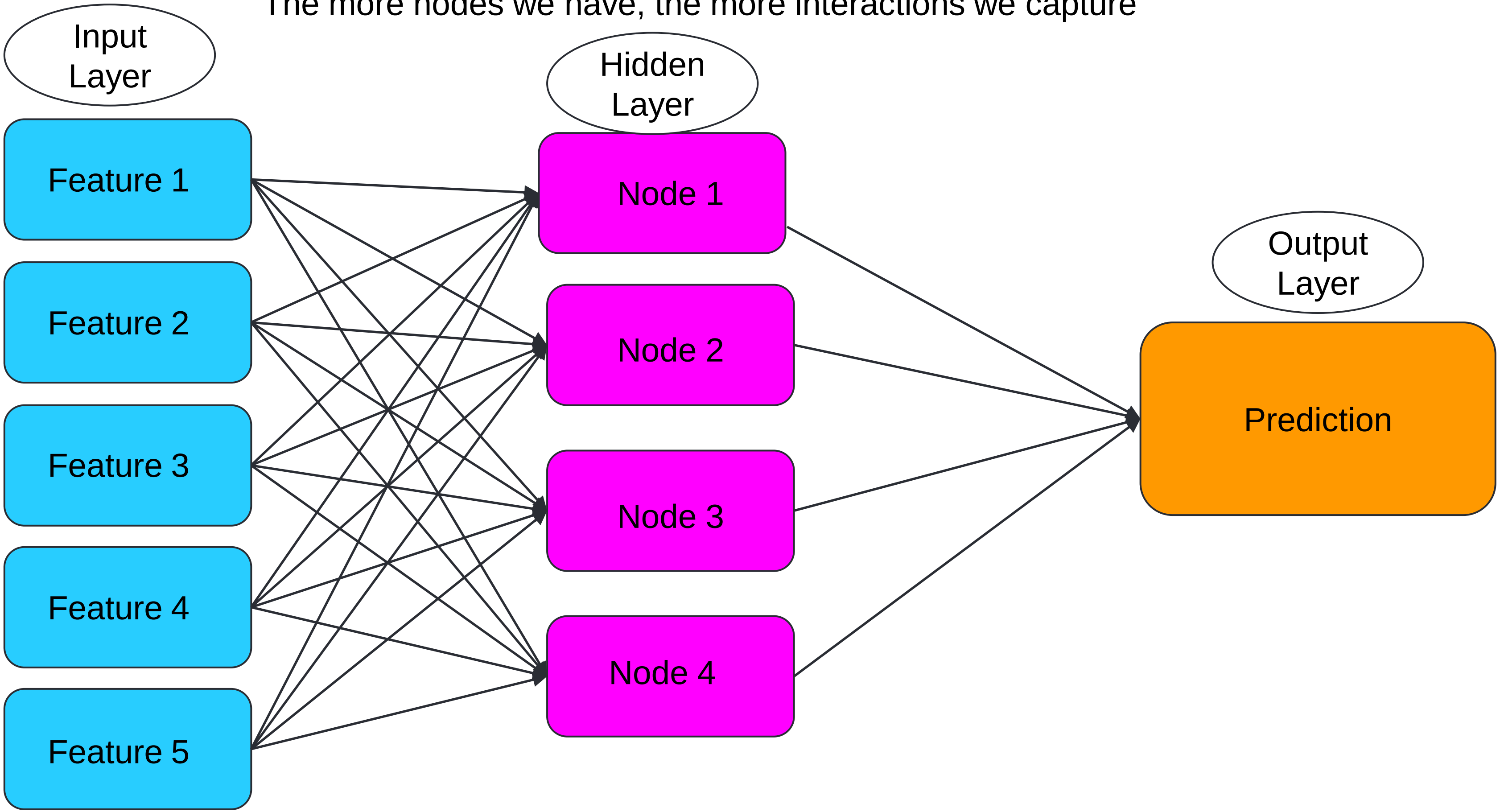
Output Layer

Prediction

This shows how Node 1 takes information from each feature to produce an output



This happens with each of the nodes in the hidden layer
The more nodes we have, the more interactions we capture



Input Layer

Feature 1

Feature 2

Feature 3

Feature 4

Feature 5

Hidden Layer

Node 1

Node 2

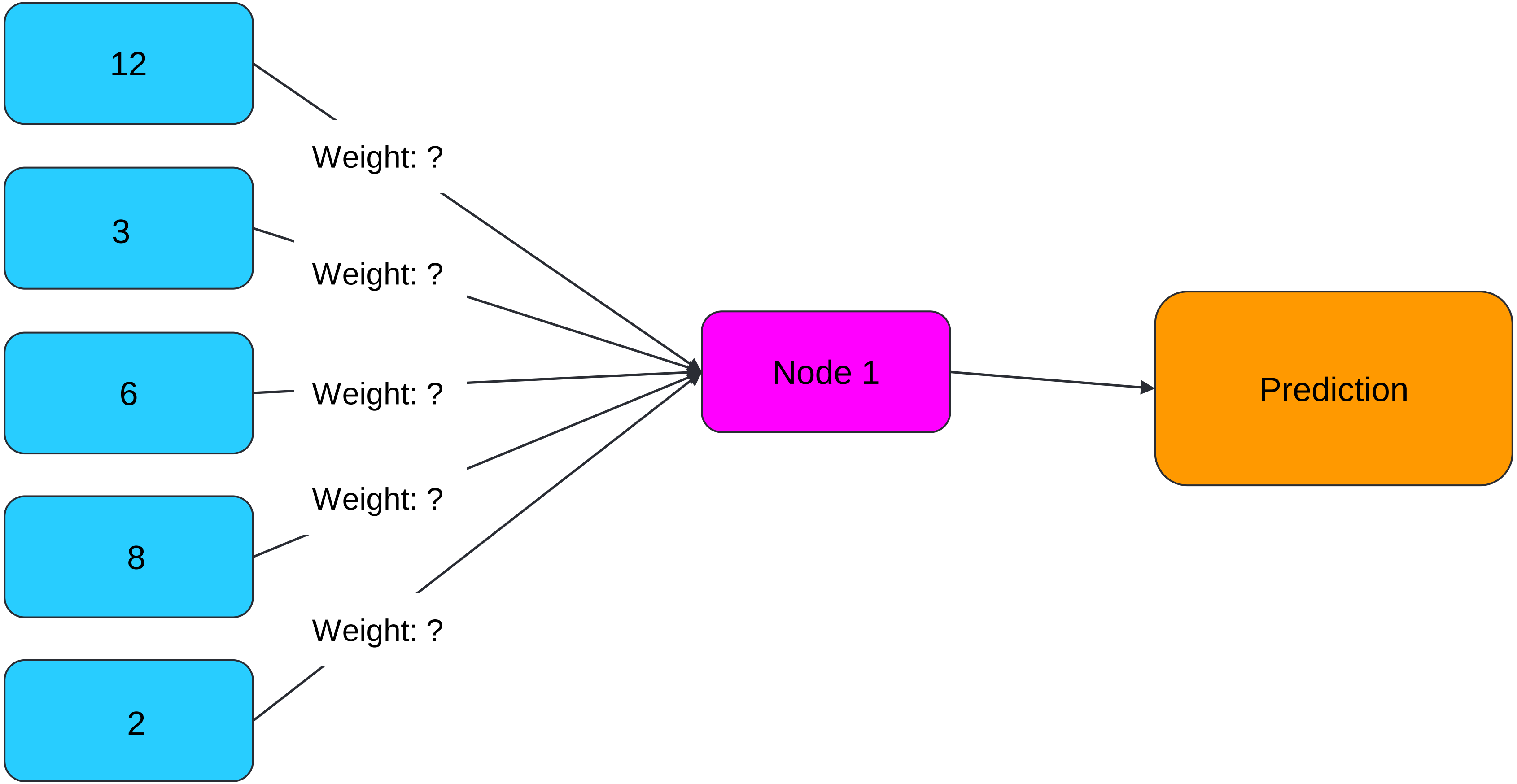
Node 3

Node 4

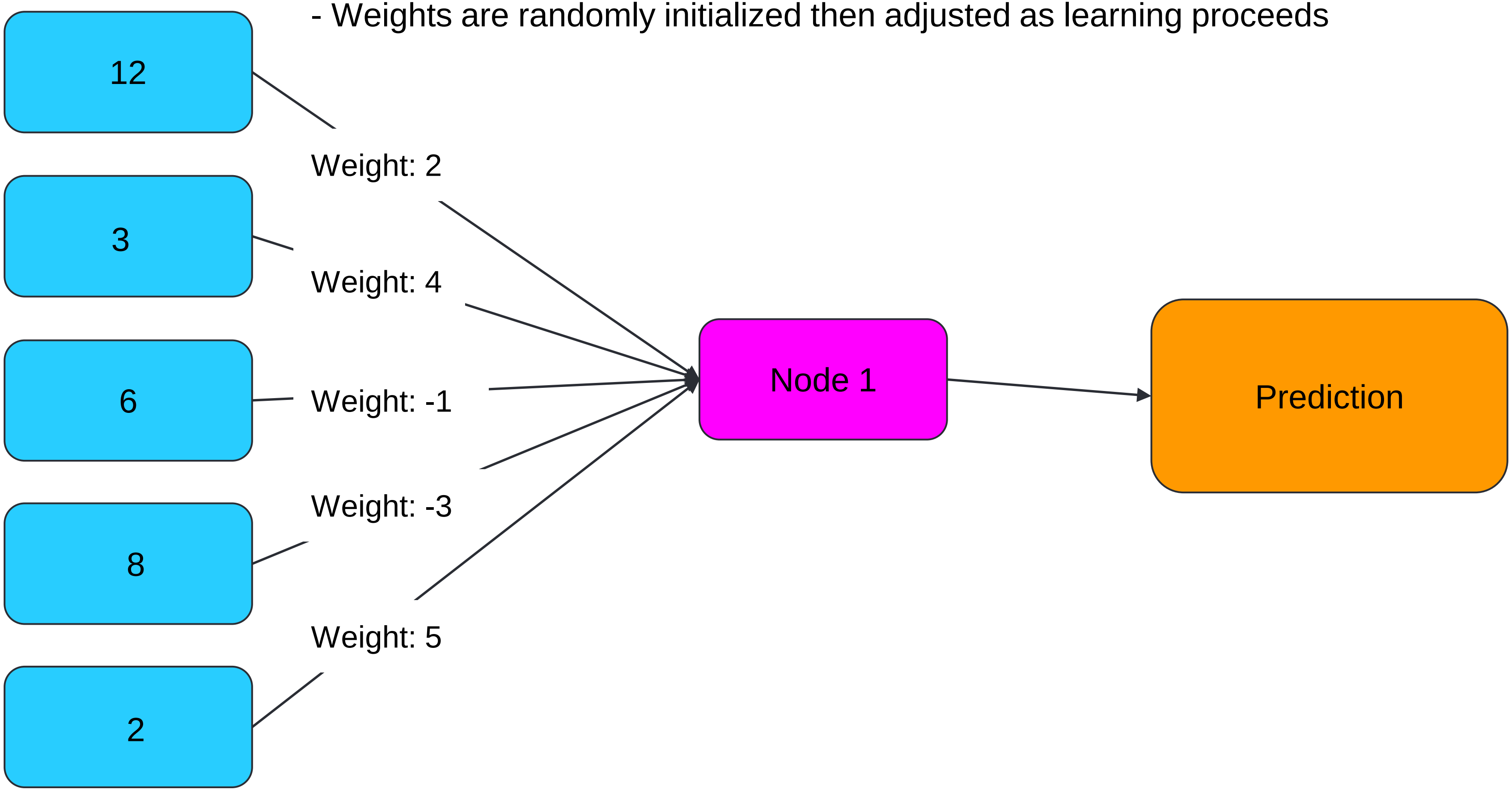
Output Layer

Prediction

- For each node, a weight is applied to each feature. Weights can be positive or negative



- For each node, a weight is applied to each feature. Weights can be positive or negative
- Weights are randomly initialized then adjusted as learning proceeds



12

3

6

8

2

Weight: 2

Weight: 4

Weight: -1

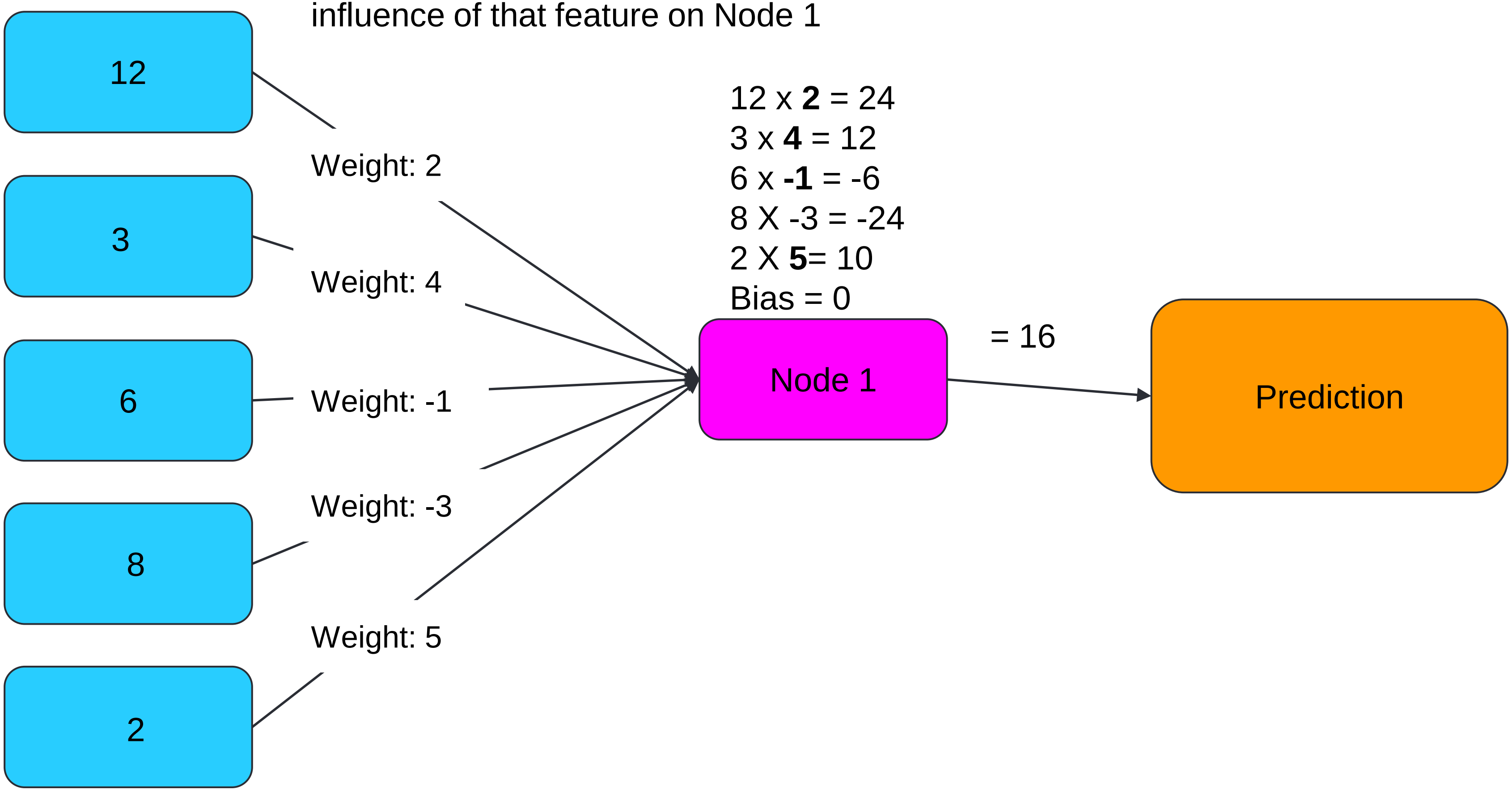
Weight: -3

Weight: 5

Node 1

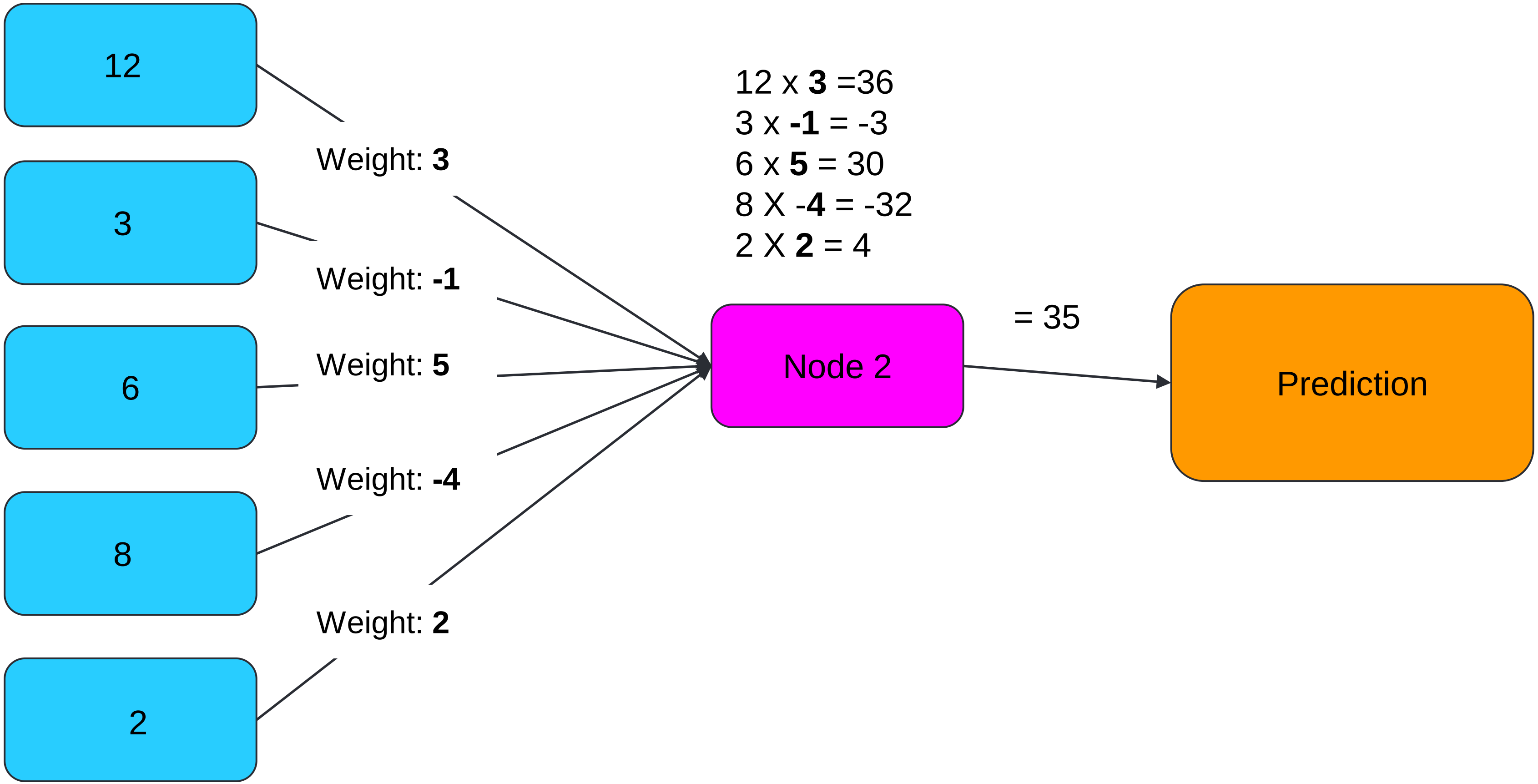
Prediction

For example, let’s apply the weights shown
The greater the weight, the greater the influence of that feature on Node 1



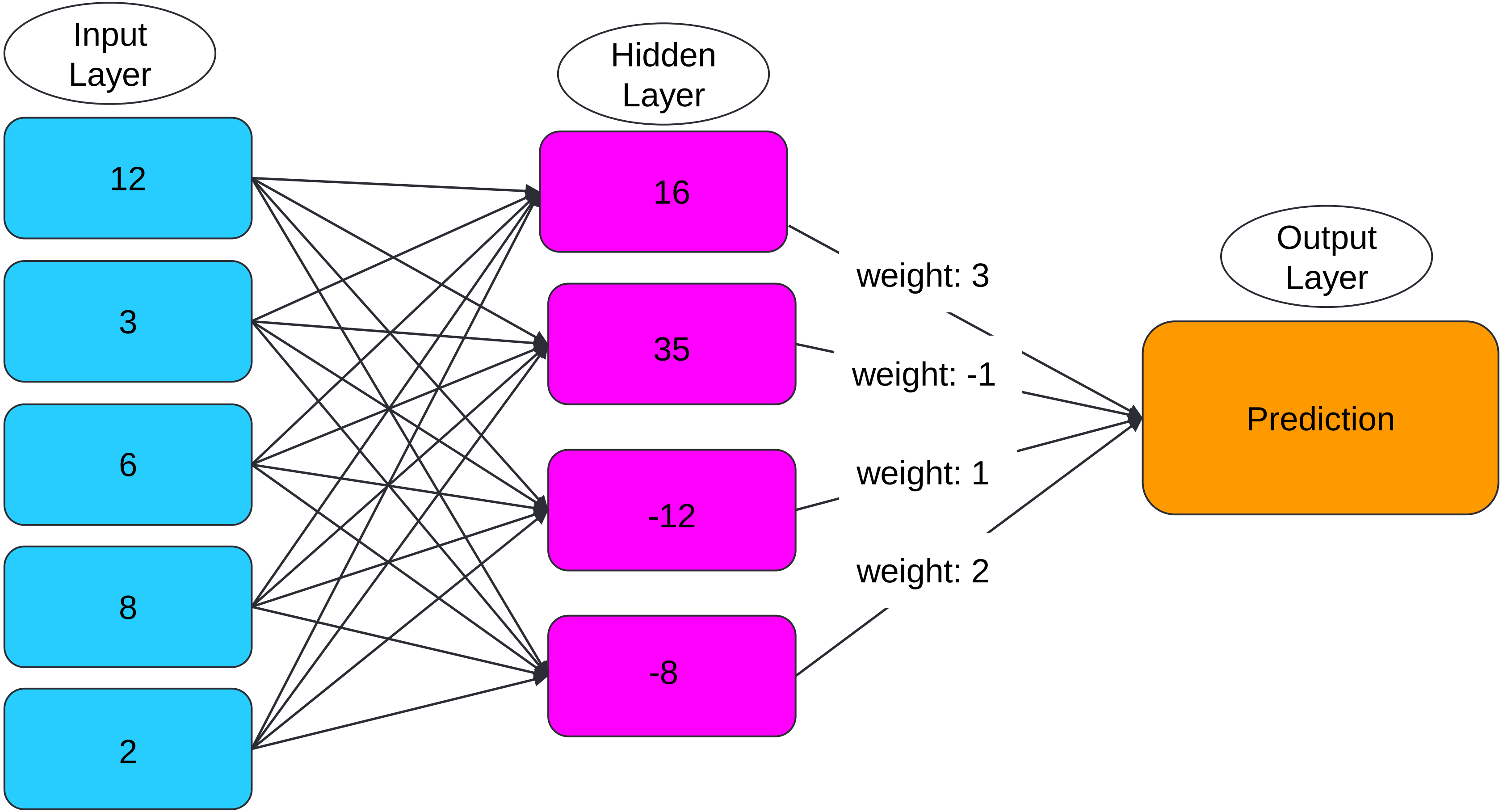
12 x **2** = 24
3 x **4** = 12
6 x **-1** = -6
8 X -3 = -24
2 X **5**= 10
Bias = 0

A similar process happens for EACH node
Each node has different weights. Note that this shows Node 2

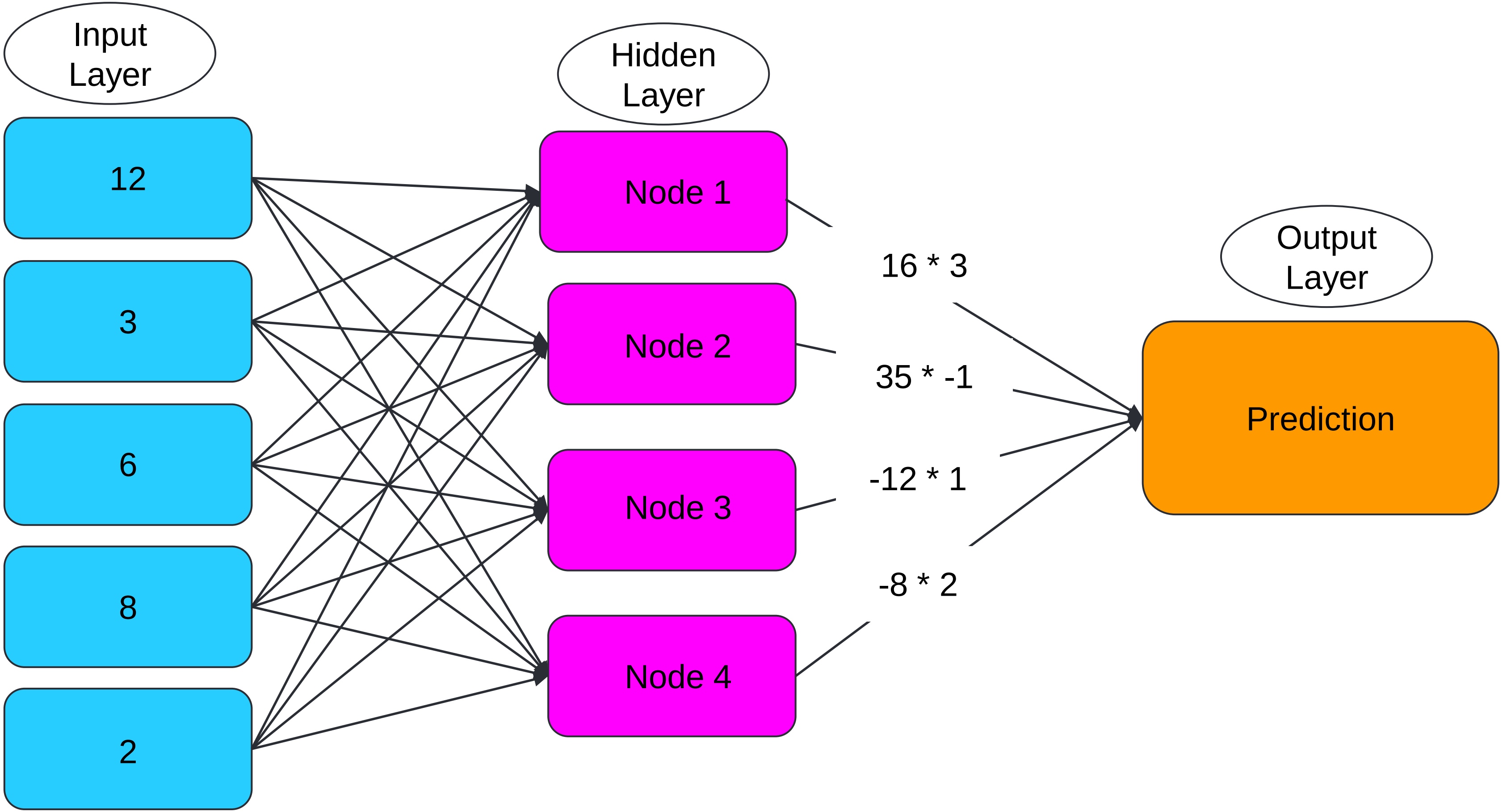


12 x **3** =36
3 x **-1** = -3
6 x **5** = 30
8 X **-4** = -32
2 X **2** = 4

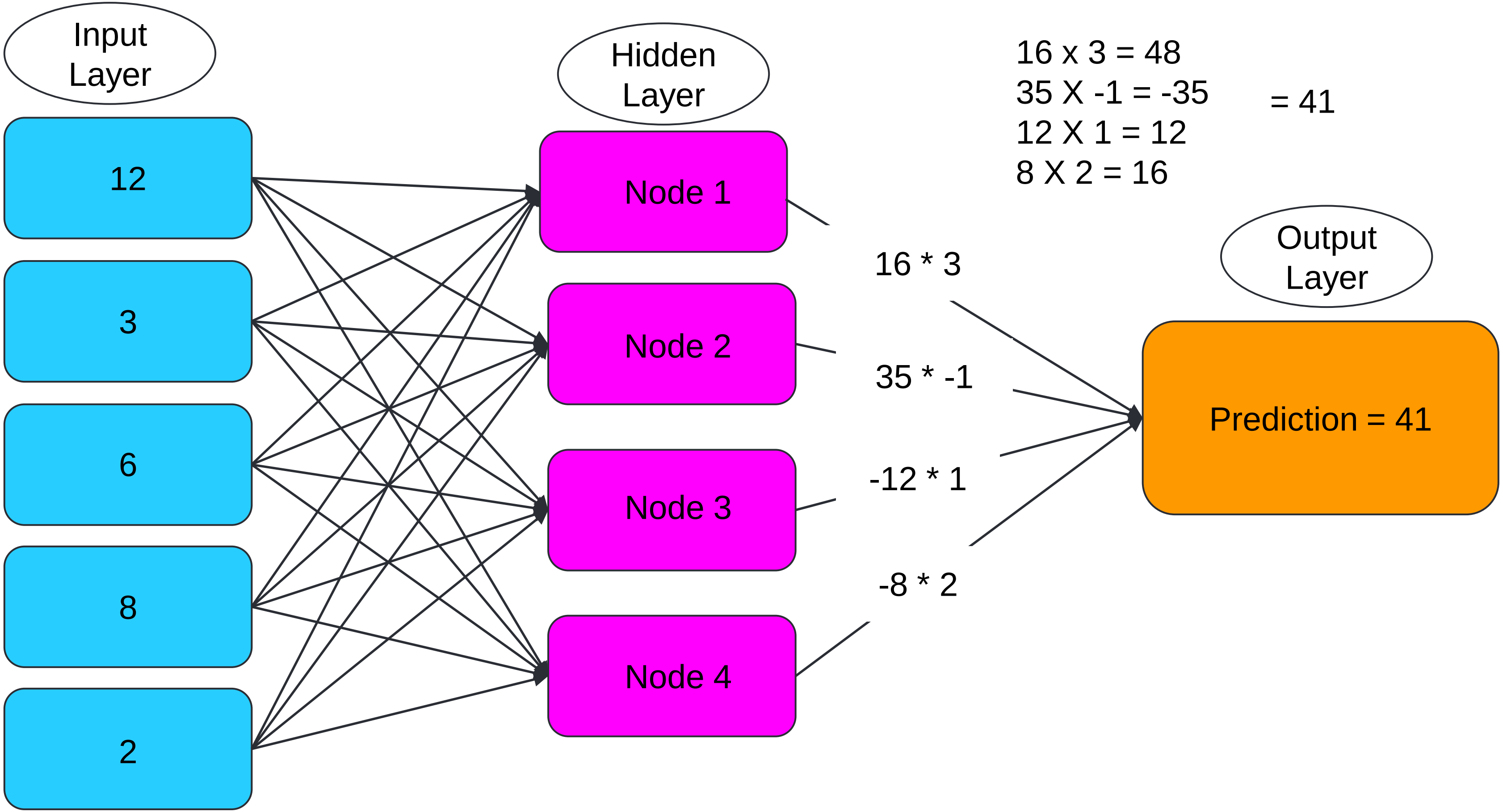
Let’s demonstrate the effect of node weights



The process repeats for each node



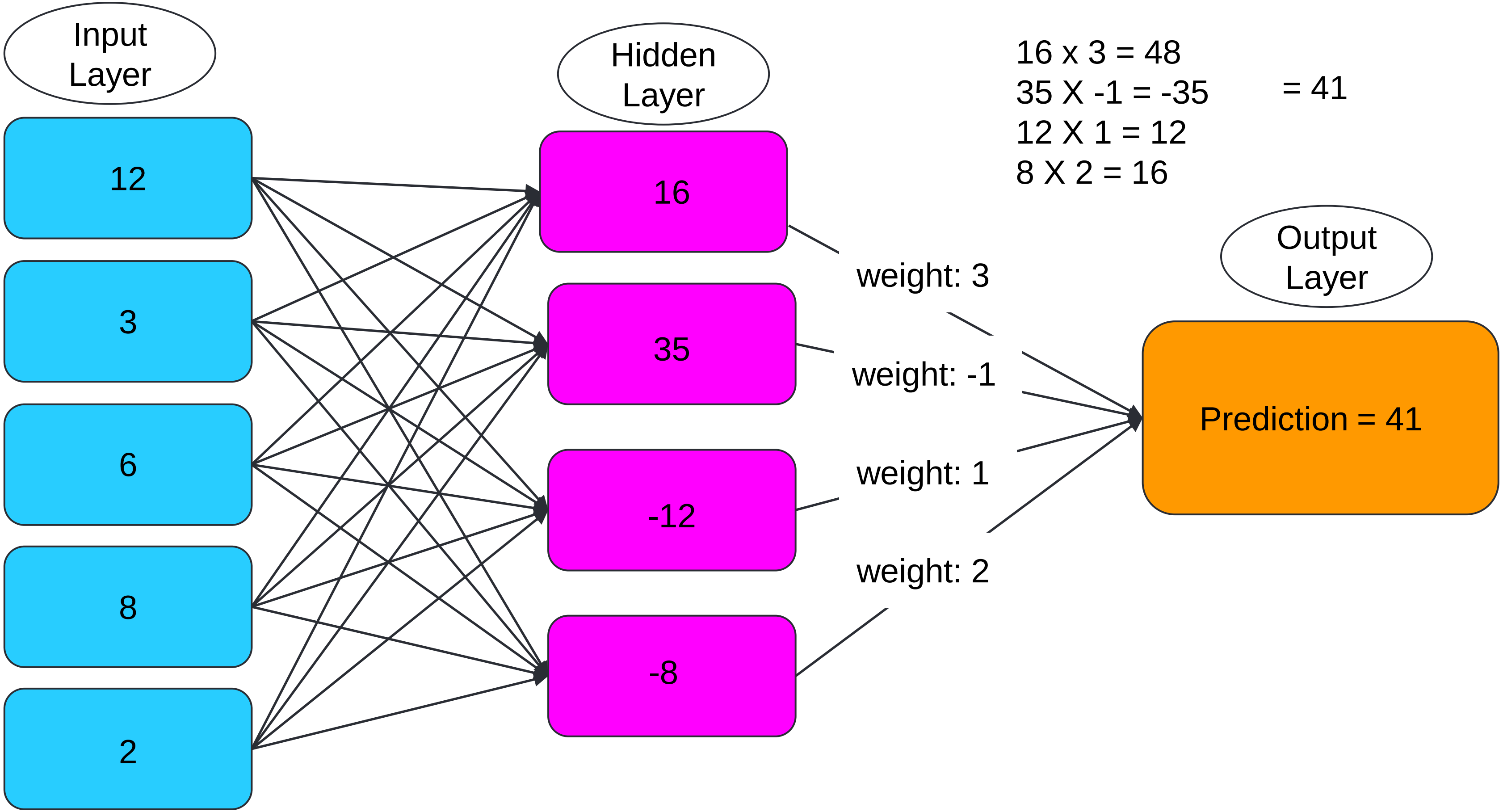
The process repeats for each node



16 x 3 = 48
35 X -1 = -35
12 X 1 = 12
8 X 2 = 16

= 41

This is the model prediction for these features



Input Layer

12

3

6

8

2

Hidden Layer

16

35

-12

-8

weight: 3

weight: -1

weight: 1

weight: 2

Output Layer

Prediction = 41

16 x 3 = 48

35 X -1 = -35

12 X 1 = 12

8 X 2 = 16

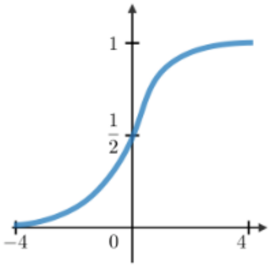
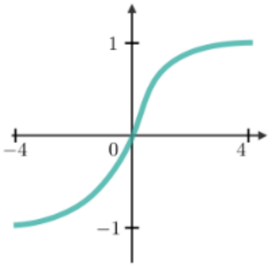
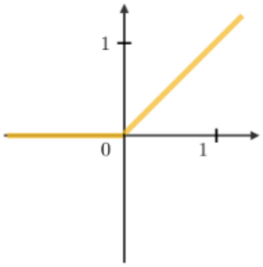
= 41

Activation Function

- Activation functions are applied to the output of each node before it is passed to the next layer.
- These allow a neural network to find non-linear relationships between features and targets
- The activation function can also limit the range of final outputs to match the problem type.

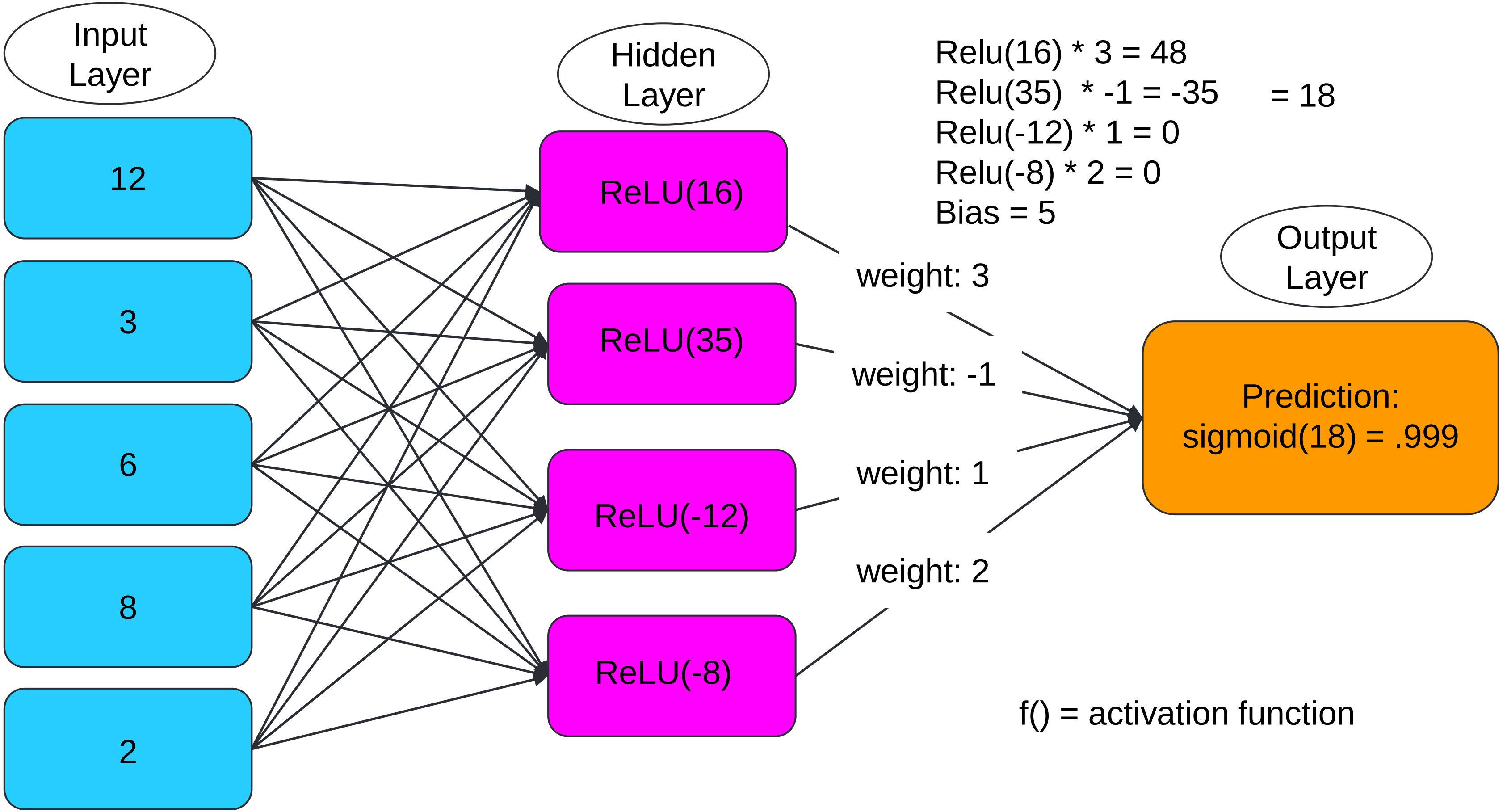
More info in Activation function

Activation Function

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	$g(z) = \max(0, z)$
		

- Hyperbolic tangent (Tanh) sometimes used instead of ReLU for hidden layers
- Sigmoid is used for output layers for classification models
- ReLU (Rectified Linear Activation) is very common for hidden layers
  - If the sum is positive, output is the input number
  - But, if the sum is negative, a value of 0 is applied instead

This is the model prediction for these features



Relu(16) * 3 = 48
Relu(35) * -1 = -35
Relu(-12) * 1 = 0
Relu(-8) * 2 = 0
Bias = 5

= 18

f() = activation function

Cost (AKA Loss)

- Overall error on all training samples:
- It’s a single number generated by a **Cost function**
  Ex. Mean Squared Error, Mean Absolute Error
- **Lower must be better (no accuracy or R^2)**

Prediction	True Value	Error	Squared Error
5	4	1	1
7	8	-1	1
-5	-8	3	9
-8	-5	-3	9

MSE = 4.47

Summarize Forward Propagation

In your breakout groups:

1. Quickly choose :
    a. **a reporter** who will take notes and share your summary to the whole class.
    b. **a facilitator** to keep the conversation moving.
    c. The rest are **researchers** who will study the slides and/or LP to help.

2. Work together to come up with a summary of the forward propagation process. In your group's own words:
    a. **How does forward propagation work and what does it do?**

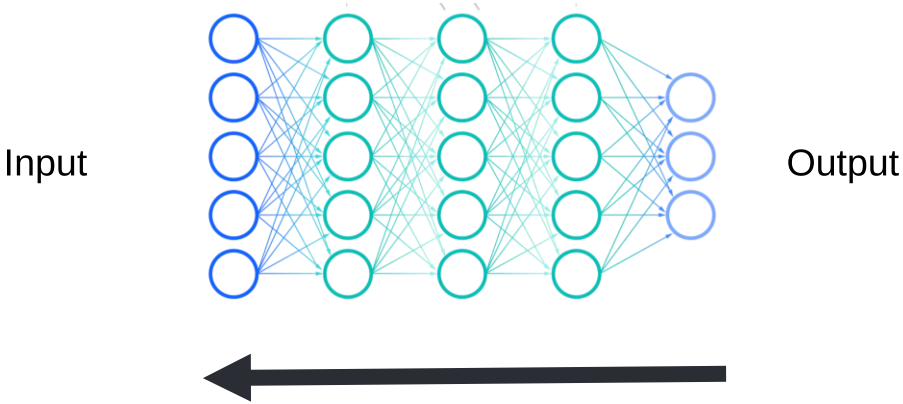
3. Add your summary to the Padlet under your room number:

4. You have 5 minutes!

Backward Propagation

Backward propagation is the process **changing the weights** of the nodes to **reduce the *Cost*** (total error) of the model on the data

Changing the weights is the process of learning.



Backward Propagation: Gradient Descent

Imagine:

A ball rolling down hill.

The **Weights** of the neural network is the ball.

The **Cost** is the altitude

Gradient Descent is gravity

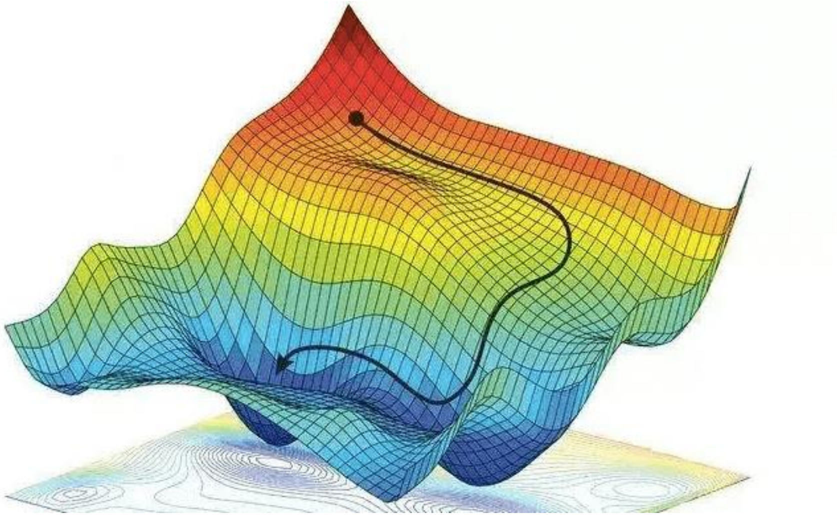
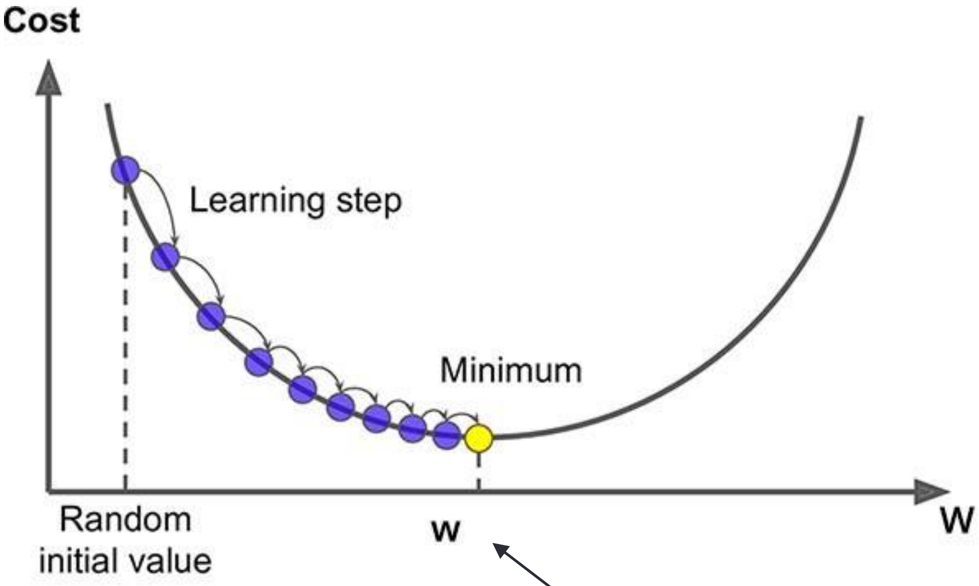


Image Source

How does the model know how to adjust the weights?

Gradient Descent



Value of one specific weight

Back Propagation Poll

Backpropagation is the process of…

Epochs:

An **Epoch** is when:

Forward Propagation

1. Make predictions.
2. Calculate **Cost**.

Backward Propagation

Use **Gradient Descent** to change **Weights.**

1. **Learn to do better next time!**

Repeat!

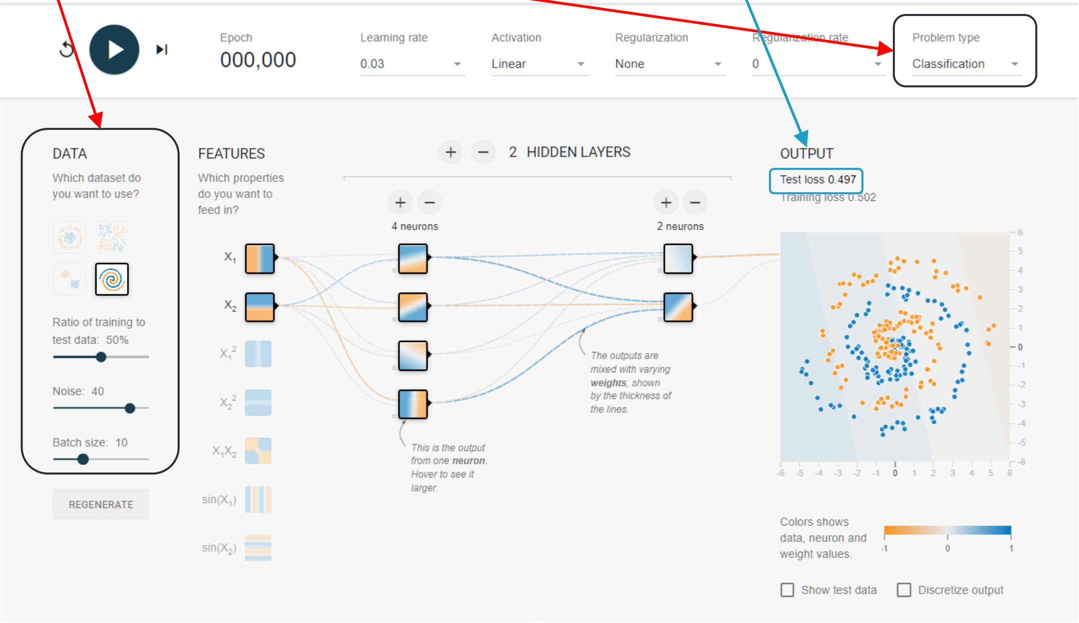
Visualizing the Neural Network in Action

In your breakout group (7 minutes):

1. Follow this link to Tensorflow Playground
2. Make sure your initial settings to match this image:
3. Adjust the following to **minimize your testing loss**:
   a. Features
   b. Layers
   c. Neurons
   d. Learning rate
   e. Activation Function
   f. Regularization
4. Be ready to share your best network and your best (lowest) testing loss.

Do Not Change These

Minimize This



Epoch 000,000

Learning rate 0.03

Activation Linear

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 40

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

2 HIDDEN LAYERS

4 neurons

2 neurons

OUTPUT

Test loss 0.497

Training loss 0.502

The outputs are mixed with varying **weights**, shown by the thickness of the lines.

This is the output from one **neuron**. Hover to see it larger.

Colors shows data, neuron and weight values.

□ Show test data □ Discretize output

Code-along

Next, we will go together to build a neural network in Keras.

Here is the notebook

Review:

1. Neural Networks are made up of **layers** of **nodes** and the **weights** between them.

2. Using **forward propagation**, they make guesses about how to solve a problem and determine how far off they were.

3. Using **backward propagation**, they change their weights to do better next time.

4. This process is called an **epoch** and it repeats many times.

Announcements

New Code Review Slots Open Tomorrow!!!

Sign up now!

Announcements

- Assignments Due this Week by Friday at 9am PST

- Belt Exam
  a. This weekend!
     December 2nd - 4th
     Set aside 8-12 hours to complete
  b. Must have attended 80% of classes
  c. Must have submitted:

All assignments from weeks 1 & 2
and all resubmits from week 1.

Assignments Due:

1. Neural Network Exercise - Optional Kaggle Competition
    a. Explain your Model Changes
    b. Recommend you submit an entry!
        Upload a screenshot of your score and rank on Discord!
2. Project 2 - Part 5 - Presentation slides
    a. Remember: NON-DATA SCIENCE AUDIENCE!!!

Daily Schedule

Next Lecture: **Tuning Neural Networks**

Read:

- Bias and Variance in Deep Learning
- Dropout
- Early Stopping
- Regression Models in Keras
- Binary Classification Models in Keras
- Multiclass Classification Models in Keras

Daily Schedule