Neural Networks II

Week 15

Applications

- Natural language processing (NLP)
- Speech recognition and synthesis
- Image recognition
- Self-driving cars
- Customer experience, healthcare, and robotics

Linear Regression

- Key foundation for deep learning
- A linear model
- Relationship between two or more variables
- Predict dependent variable based on independent variable
- Slope and intercept models the relationship

y	Dependent Variable
x	Independent Variable
a	Slope
b	Intercept

$$y = ax + b$$

$$y = a_1x_1 + a_2x_2 \ldots + a_nx_n + b$$

Building a linear regression model

- Find values for slope and intercept
- Use known values of x and y (multiple samples)
- Multiple independent variables make it complex

$5 = 2a + b, 9 = 4a + b$

$b = 5 - 2a$

$9 = 4a + 5 - 2a$

$a = (9-5)/2 = 2$

$b = 5 - 2*2 = 1$

Logistic regression

- A binary model
- Relationship between two or more variables
- Output is 0 or 1

y	Dependent Variable
x	Independent Variable
a	Slope
b	Intercept
f	Activation Function

$$y = f(ax + b)d$$

$$y = f(a_1x_1 + a_2x_2 \ldots + a_nx_n + b)$$

An analogy for Deep Learning

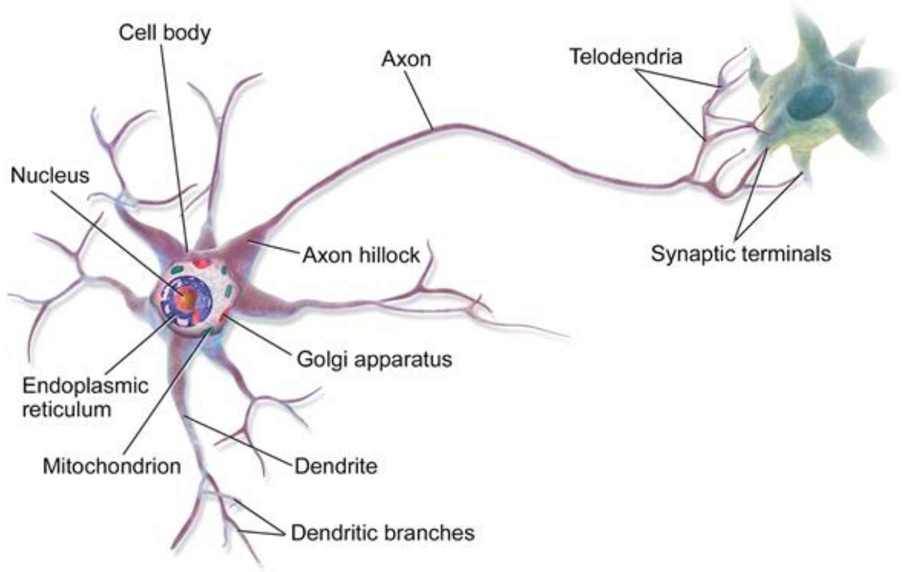
- Complex and iterative process
- Starts with random initialization and works towards the right values by trial and error

$$10 = 3a + b$$

Trial	a	b	3a + b	Error
1	1	1	4	6
2	4	3	15	-5
3	2	2	8	2
4	3	2	11	-1
5	2	3	9	1
6	2	4	10	0

The Perceptron

- An algorithm for supervised learning for binary classification
- Resembles a cell in the human brain
- A single cell neural network
- Based on logistic regression



Perceptron formula

$$y = f(a_1x_1 + a_2x_2 \ldots + a_nx_n + b)$$

↓

$$y = f(w_1x_1 + w_2x_2 \ldots + w_nx_n + b)$$

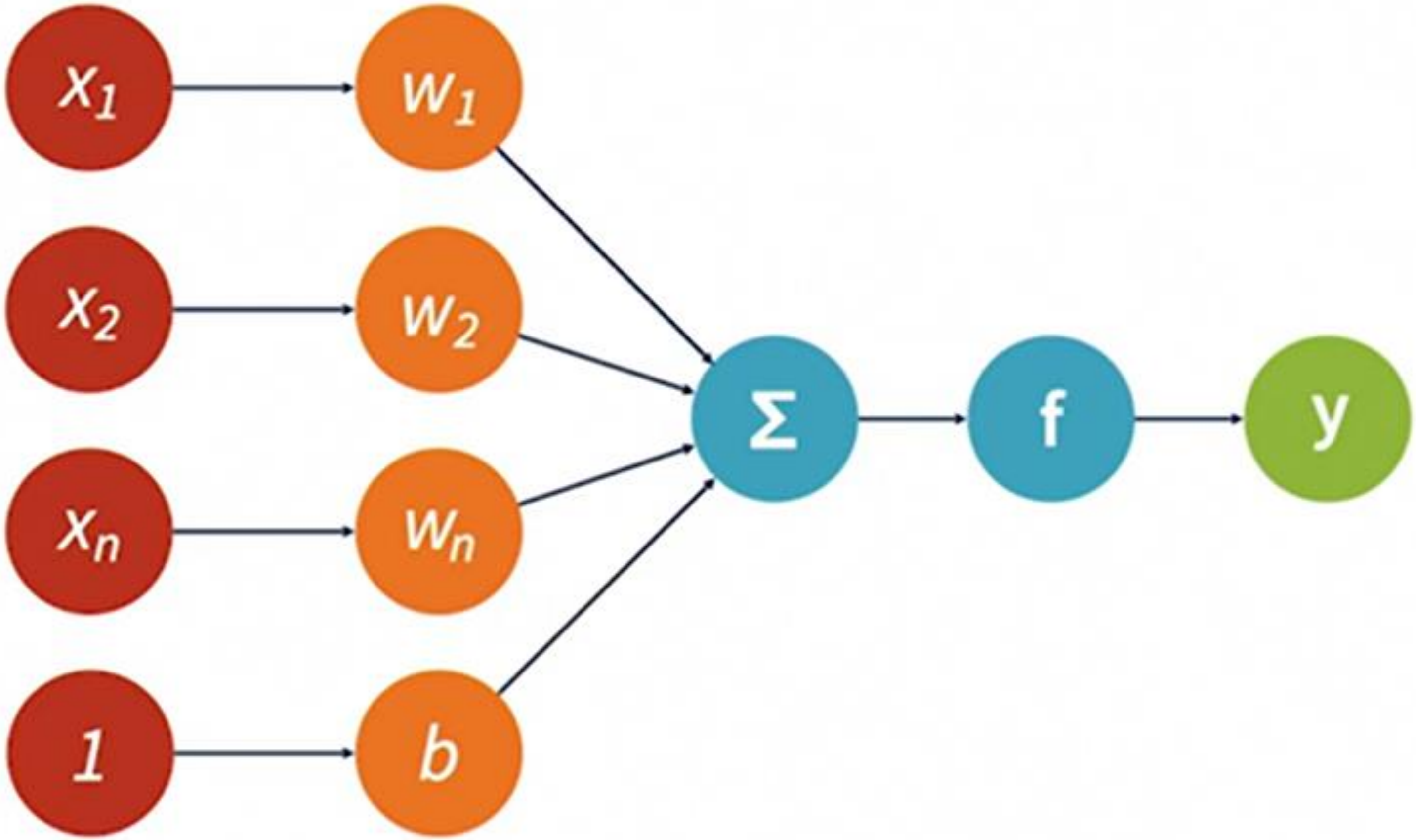
w	Weight
b	Bias
f	Activation Function

Activation Function Example

1 if value > 0

0 otherwise

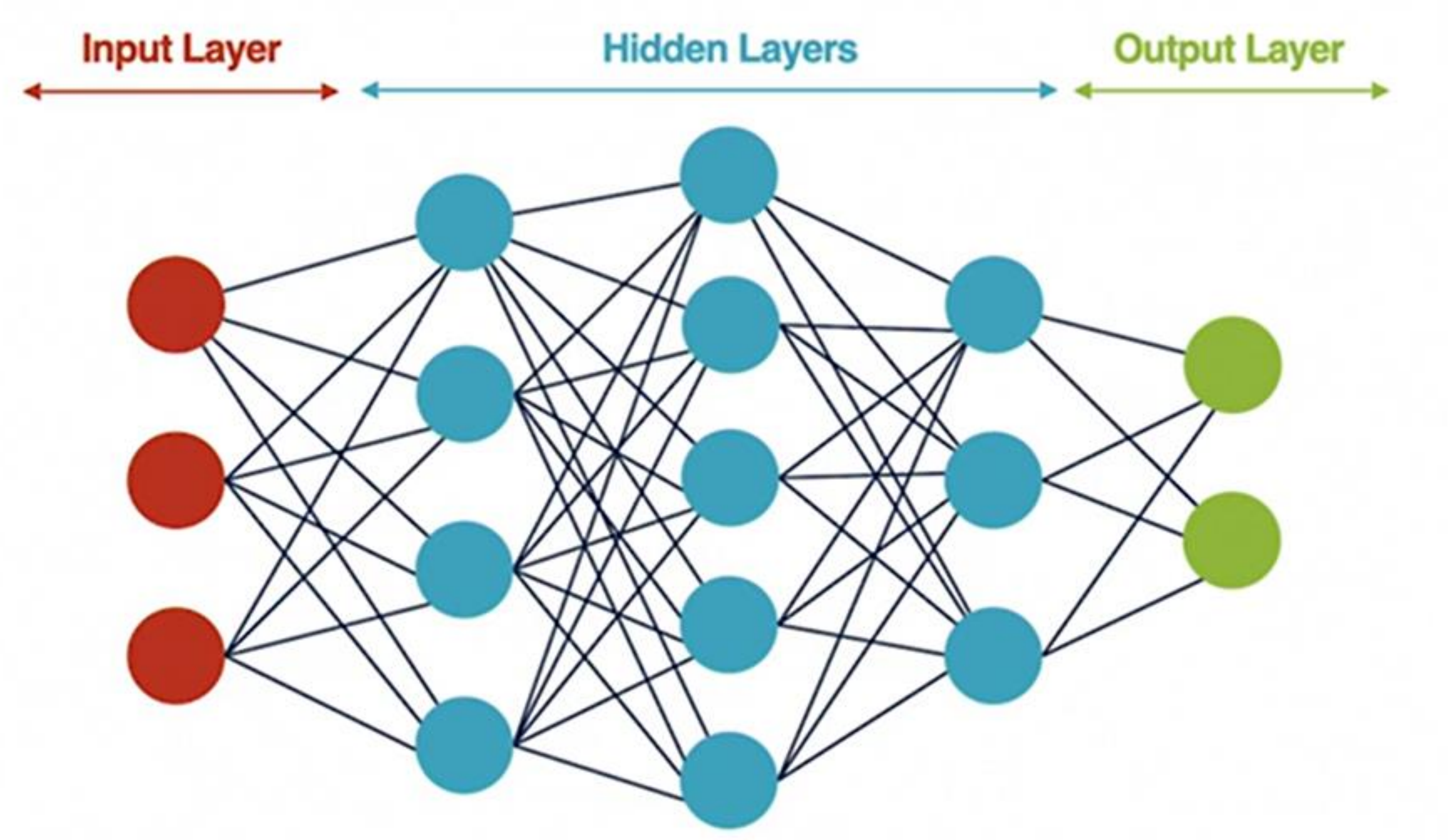
Perceptron



Artificial Neural Network

- A network of perceptron
- Perceptrons are called nodes in the neural network
- Nodes organized as layers
- Each node has weights, biases and activation functions
- Each node is connected to all nodes in the next layer

Artificial Neural Network



How ANN works for prediction?

- Inputs (independent variables) are sent from the input layer
- Inputs passed on to the nodes in the next hidden layer
- Each node computes its output based on its weights, biases, and activation functions
- Node output is then passed on as inputs to the next layer

Training an ANN

A model is represented by parameters and hyperparameters.

- Weights, biases, nodes, layers, layers, etc.

Training a model means determining the best possible values for the parameters and hyperparameters that maximize accuracy.

Inputs, weights, and biases might be n-dimensional arrays.

Recall the Linear Regression Analogy

Weight → a, Bias → b

Trial	a	b	3a + b	Error
1	1	1	4	6
2	4	3	15	-5
3	2	2	8	2
4	3	2	11	-1
5	2	3	9	1
6	2	4	10	0

Training Process

- Use training data (known values of inputs and outputs)
- Create network architecture with intuition
- Start with random values for weights and biases
- Minimize error in predicting known outputs from inputs
- Adjust weights and biases until error is minimized
- Improve model by adjusting layers, node counts and other hyper parameters

The Input Layer

The input to deep learning is a vector of <u>numeric</u> values

A vector is a tuple of one or more values

- E.g., (1.0, 2.1, 3.5)

Defined usually as a NumPy array

Represents the feature variables for prediction

Samples and Features

Features

Employee ID	Age	Salary	Service
1001	28	40,000	5
1002	47	60,000	20
1003	35	55,000	10

Samples

Input Pre-Processing

- Features need to be converted to numeric representations

Input Type	Preprocessing Needed
Numeric	Centering and scaling
Categorical	Integer encoding, one-hot encoding
Text	TF-IDF, embeddings
Image	Pixels – RGB representation
Speech	Time series of numbers

Example

Raw Data

Age	Salary	Service
28	40,000	5
47	60,000	20
35	55,000	10

Centered and Scaled

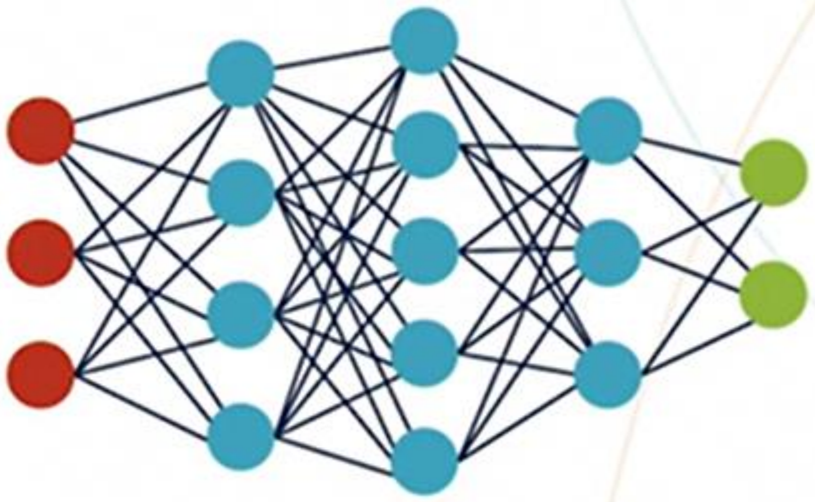
x1	x2	x3
-1.10	-1.37	-1.07
1.32	0.98	1.34
-0.21	0.39	-0.27

Transposed

x1	-1.1	1.32	-0.21
x2	-1.37	0.98	0.39
x3	-1.07	1.34	-0.27

Hidden Layers

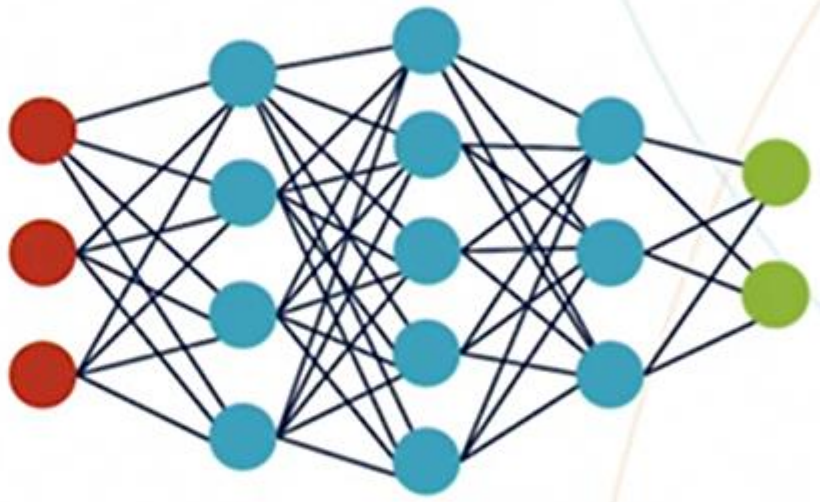
- An ANN can have one or more hidden layers
- Each hidden layer can have one or more nodes (or perceptron) (count in 2^n)
- A neural network architecture is defined by the number of layers and nodes



Inputs and Outputs

- The output of each node in the previous layer becomes the input for each node in the current layer (fully connected)

- Each node produces one output that is forwarded to the next layer



How to determine hidden layer architecture?

Each node *learns* something about the feature-target relationship

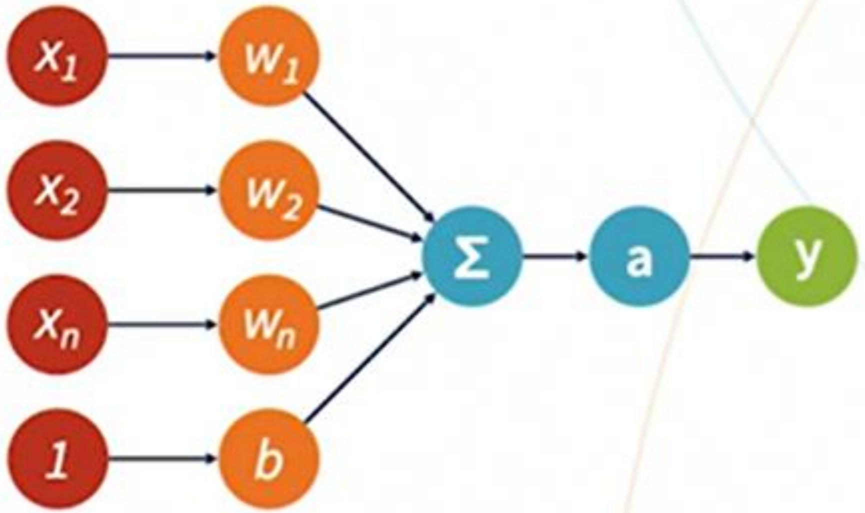
More nodes and layers mean

- Web applications on AWS
- Analytics on GCP

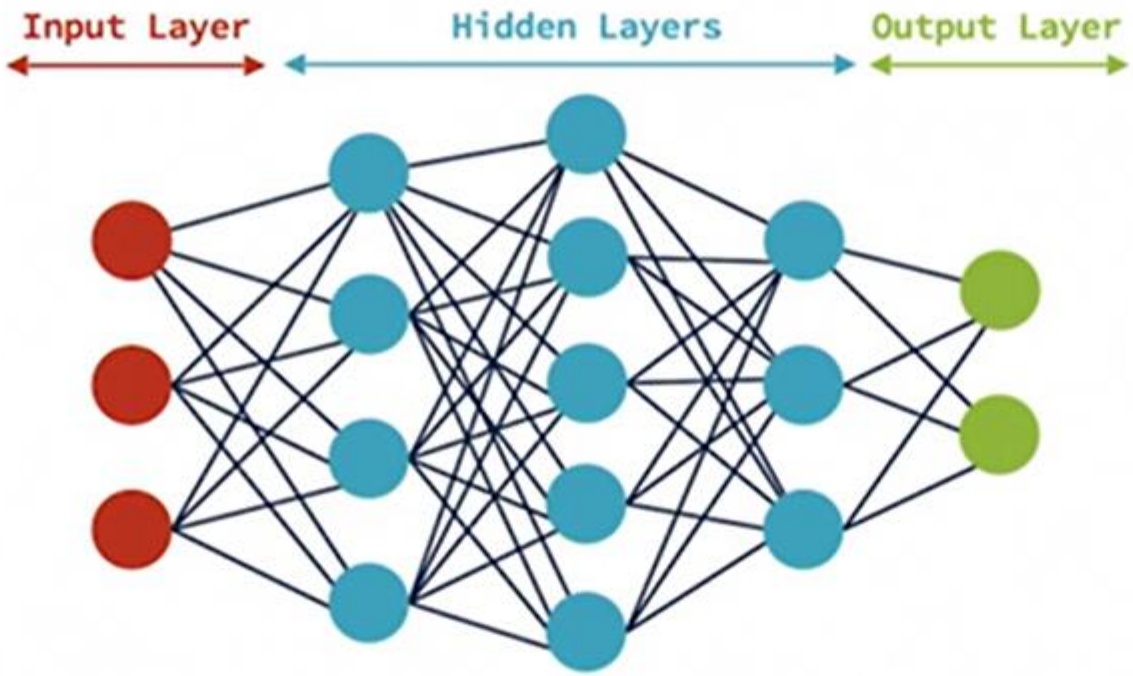
Architecture decided by experimentation

Weights and biases

- Weights and biases represent the trainable parameters in an ANN
- Numeric values
- Each input for each node has a weight associated with it
- Each node has a bias associated with it



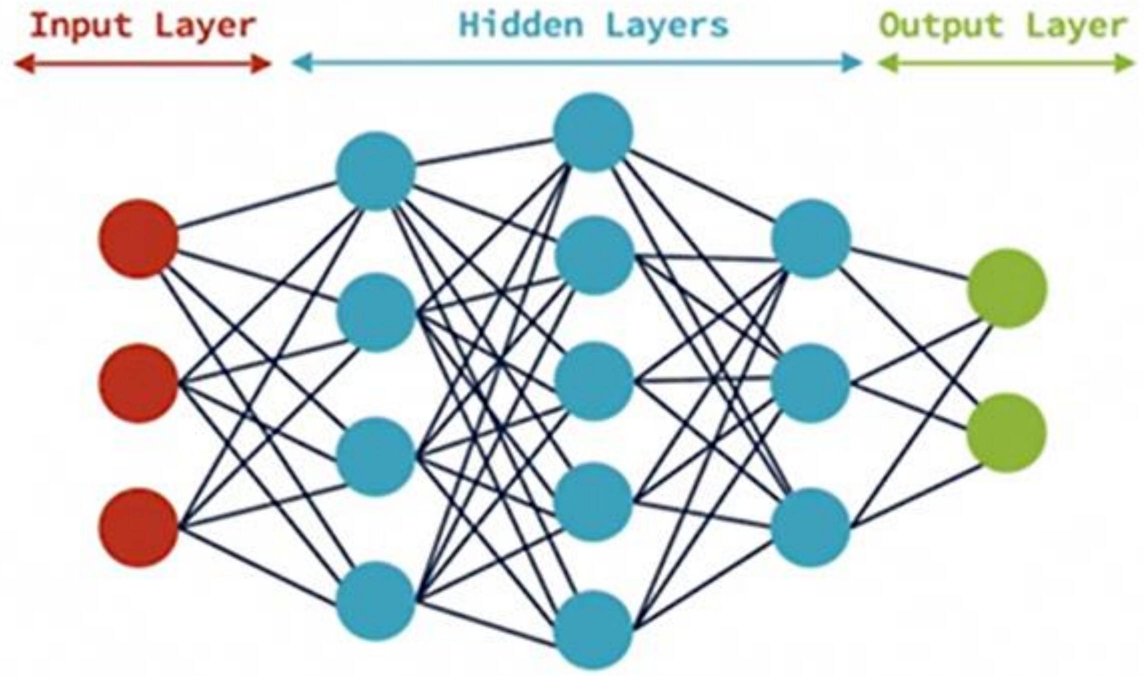
Weights and biases for a layer



Layer	Inputs	Nodes	Weights	Biases
HL 1	3	4	12	4
HL 2	4	5	20	5
HL 3	5	3	15	3
Output	3	2	6	2
Total			**53**	**14**

Computing Outputs

- This example shows the computation for hidden layer two which has four inputs and five nodes.

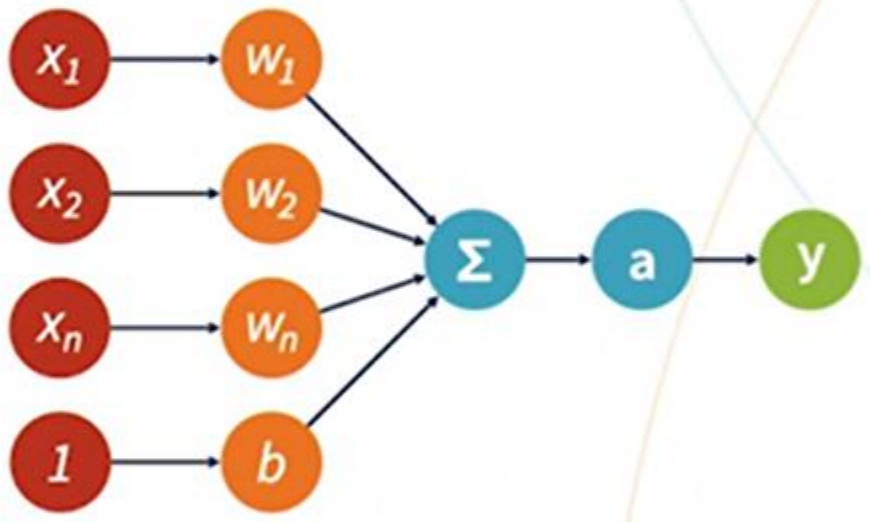


$$\begin{bmatrix} w11 & w21 & w31 & w41 \\ w12 & w22 & w32 & w42 \\ w13 & w23 & w34 & w43 \\ w14 & w24 & w34 & w44 \\ w15 & w25 & w35 & w45 \end{bmatrix} * \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix} + \begin{bmatrix} b1 \\ b2 \\ b3 \\ b4 \\ b5 \end{bmatrix} = \begin{bmatrix} y1 \\ y2 \\ y3 \\ y4 \\ y5 \end{bmatrix}$$

$$W^T X + B = Y$$

Activation functions

- Determines which nodes propagate information to the next layer
- Filters and normalizes
- Converts output to nonlinear
- Critical in learning patterns



Popular Activation Functions

Activation Function	Output
Sigmoid	0 to 1
Tanh	-1 to +1
Rectified Linear Unit (ReLU)	0 if x < 0; x otherwise
Softmax	Vector of probabilities, with sum=1

Choice depends on problem and experimentation.

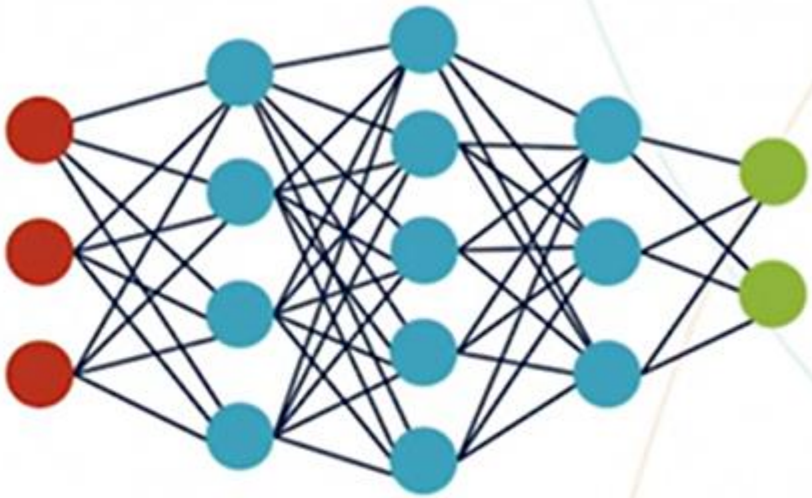
The Output Layer

- One layer of output, produces desired Y
- Has its own weights and biases
- Softmax activation used for classification problems
- May need postprocessing to convert to business values

Output Layer size

Size depends on the problem

- 1 for binary classification
- n for a n-class classification
- 1 for regression problems
- Vary based on other problem domains



Training the network – data pre-processing

	Sample 1	Sample 2	Sample 3	Sample 4
Feature 1	x11	x21	x31	x41
Feature 2	x12	x22	x32	x42
Feature 3	x13	x23	x34	x43
Feature 4	x14	x24	x34	x44
Feature 5	x15	x25	x35	x45
Target	y1	y2	y3	y4

Split input

- Training set: Used to fit the parameters
- Validation set: Used for model selection/tuning
- Test set: Used to measure the final model performance
- Usual split: 80:10:10

Select values for the model

Select values for the model

- Layers and nodes in the layer, activation functions
- Hyper parameters

Selection criteria

- Initial selection based on intuition/reference
- Adjustment based on results

Initialize weights

- All weights and bias parameters need to be initialized to some value before we start training

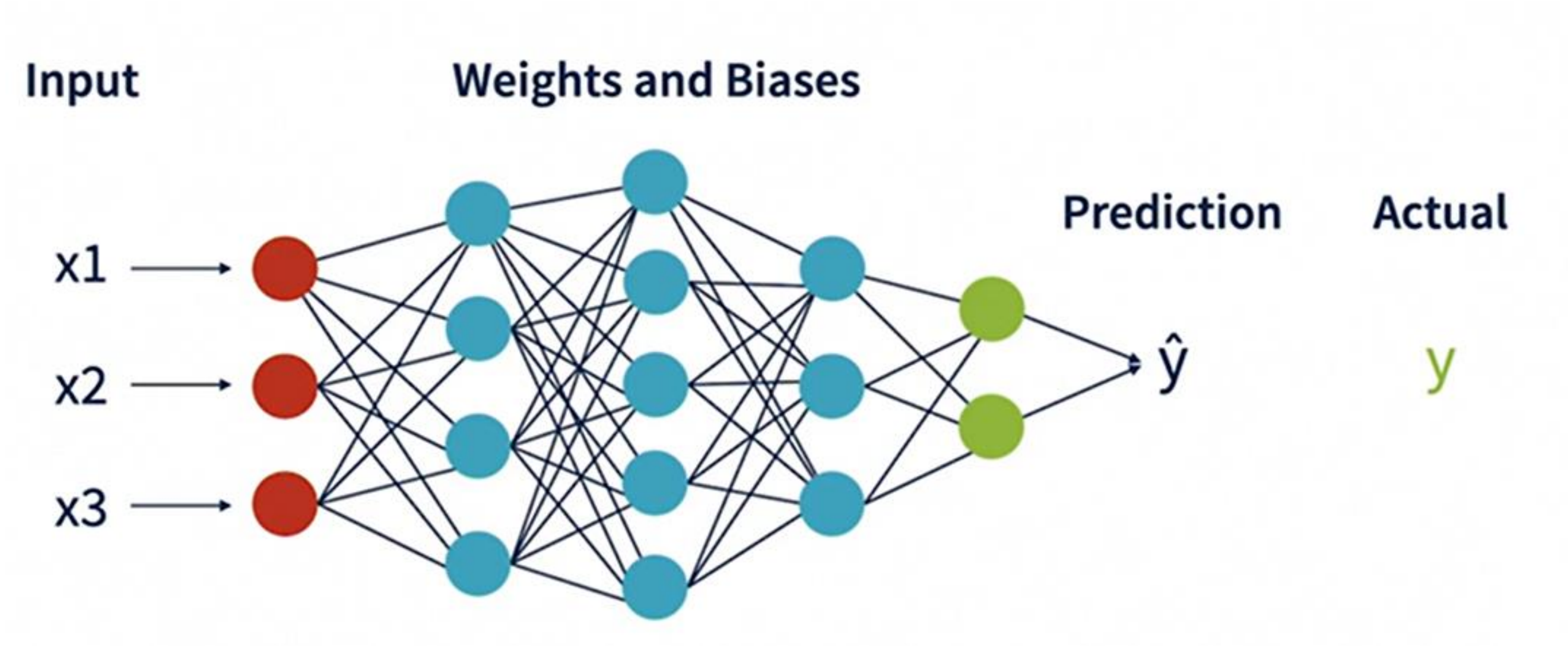
- Zero initialization: Initialize to zeros, not recommended

- Random initialization: Initialize to random values from a standard normal distribution (mean = 0, SD = 1)

Forward Propagation

	Sample 1	Sample 2	Sample 3	Sample 4
Feature 1	x11	x21	x31	x41
Feature 2	x12	x22	x32	x42
Feature 3	x13	x23	x34	x43
Target	y1	y2	y3	y4
Prediction	ŷ1	ŷ2	ŷ3	ŷ4

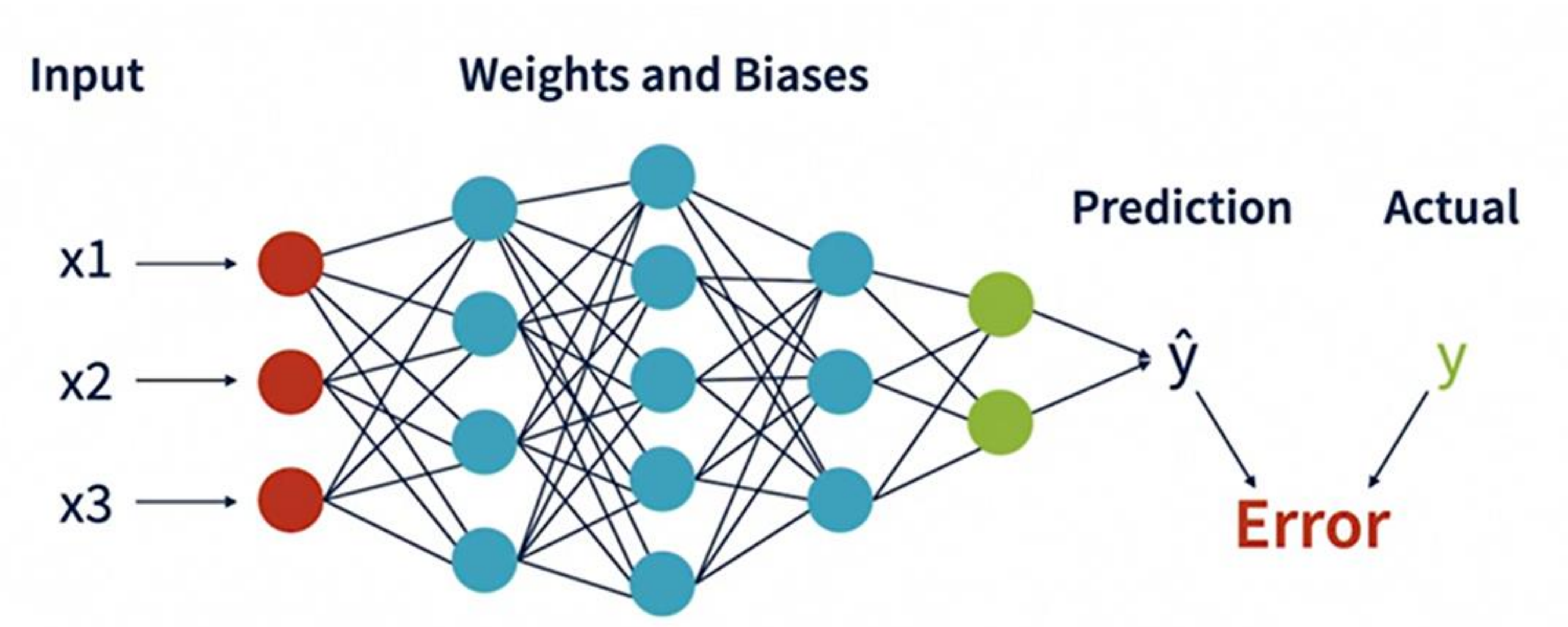
Forward propagation: 1 Sample



Forward propagation: All Samples

- Send each sample through the neutral network and obtain the value of ŷ
- Repeat for all samples and collect a set of ŷ
- Compare the values of ŷ to y to obtain error rates

Measuring Accuracy and Error



Loss and Cost function

- A loss function measures the prediction error for a single sample
- A cost function measures the error across a set of samples
- Popular Cost Functions

Cost Functions	Applications
Mean Square Error (MSE)	Regression
Root Mean Square Error (RMSE)	Regression
Binary Cross Entropy	Binary classification
Categorical Cross Entropy	Multi-class classification

Measuring Accuracy

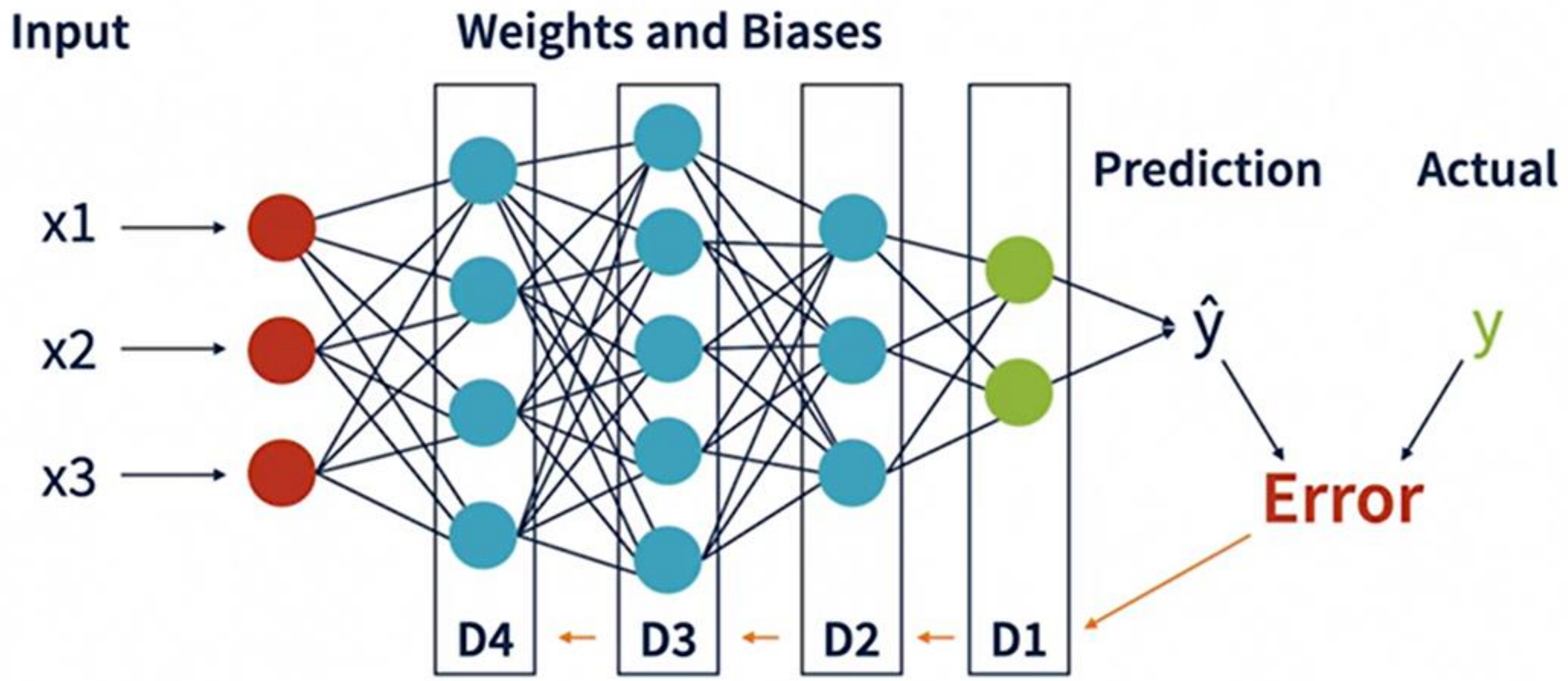
- Send a set of samples through the ANN and predict outcome
- Estimate the prediction error between the predicted outcome and expected outcome using a cost function
- Use back propagation to adjust weights based on the error value

Back Propagation

- Each node in a neutral network contributes to the overall error in prediction (differing contributions)
- A node's contribution is driven by its weights and bias
- Weights and biases need to be adjusted to lower the error contribution by each node

How does it work?

- It works in reverse of the forward propagation.
- Start from the output layer
- Compute a delta value based on the error found
- Apply the delta to adjust the weights and biases in the layer
- Derive a new error value
- Back propagate the new error to the previous layer and repeat


Input
Weights and Biases
x1
x2
x3
Prediction
Actual
ŷ
y
Error
D4
D3
D2
D1

Gradient Descent

Repeat the learning process.

- Forward propagation
- Estimate error
- Backward propagate
- Adjust weights and biases

Batch

- A set of samples sent through ANN in a single pass
- The training data set can be divided into one or more batches
- Training data is sent to the ANN one batch at a time
- Cost estimated and parameters updated one batch at a time
- Batch gradient descent
  - Batch size = training set size
- Mini-batch gradient descent
  - Batch size < training set size
- Typical batch sizes are 32, 64, 128, etc.

Epoch

- The number of times the entire training set is sent through the ANN
- An epoch has one or more batches
- The training process completes when all epoch is complete
- Epoch sizes can be higher to achieve better accuracy

Epoch and Batch Example

- Training set size = 1000, batch size = 128, epoch = 50
- Batches per epoch = ceil (1000 / 128) = 8
- Total iterations (passes) through ANN = 8 * 50 = 400
- Batch size and epoch are hyperparameters that can be tuned to improve model accuracy

Validation and Testing

- Validation
  - During learning, the predictions are obtained for the same data that is used to train the parameters (weights and biases)
  - After each epoch and corresponding parameter updates, the model can be used to predict for the validation data set
  - Accuracy and/or loss can be measured and investigated
  - Model can be fine-tuned and learning process repeated based on results.
- Evaluation
  - After all fine-tuning is completed and final model obtained, the test data set can be used to evaluate the model
  - Results obtained with test data is used to measure the performance of the model

Summary: An ANN Model

Parameters

- Weights
- Biases

Hyperparameters

- Number of layers, nodes in each layer, activation function
- Cost functions, learning rate, optimizers
- Batch size, epoch

Summary: Prediction Process

Preprocess and prepare inputs

Pass inputs to the first layer

- Compute Y using weights, biases, activation
- Pass to the next layer

Repeat process until output layer

Postprocess output for predictions