

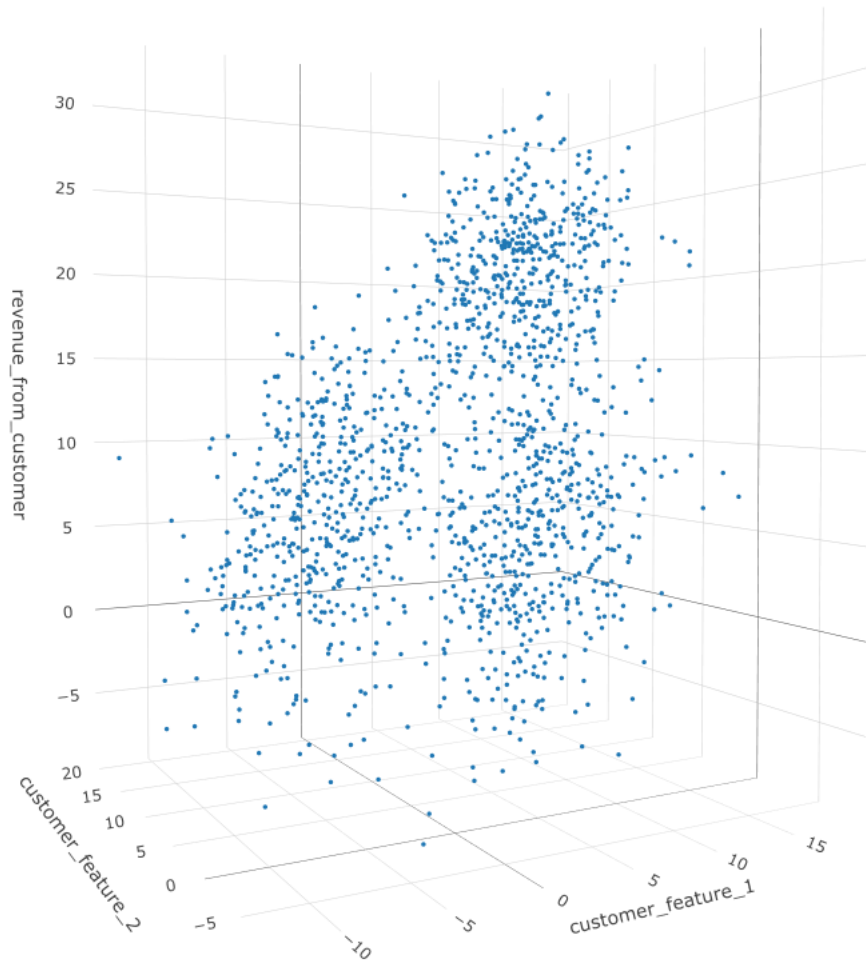
Machine Learning

Live Session #2

Review of Neural Network Basics

Supervised Learning

- Two types of supervised learning: regression and classification



Regression



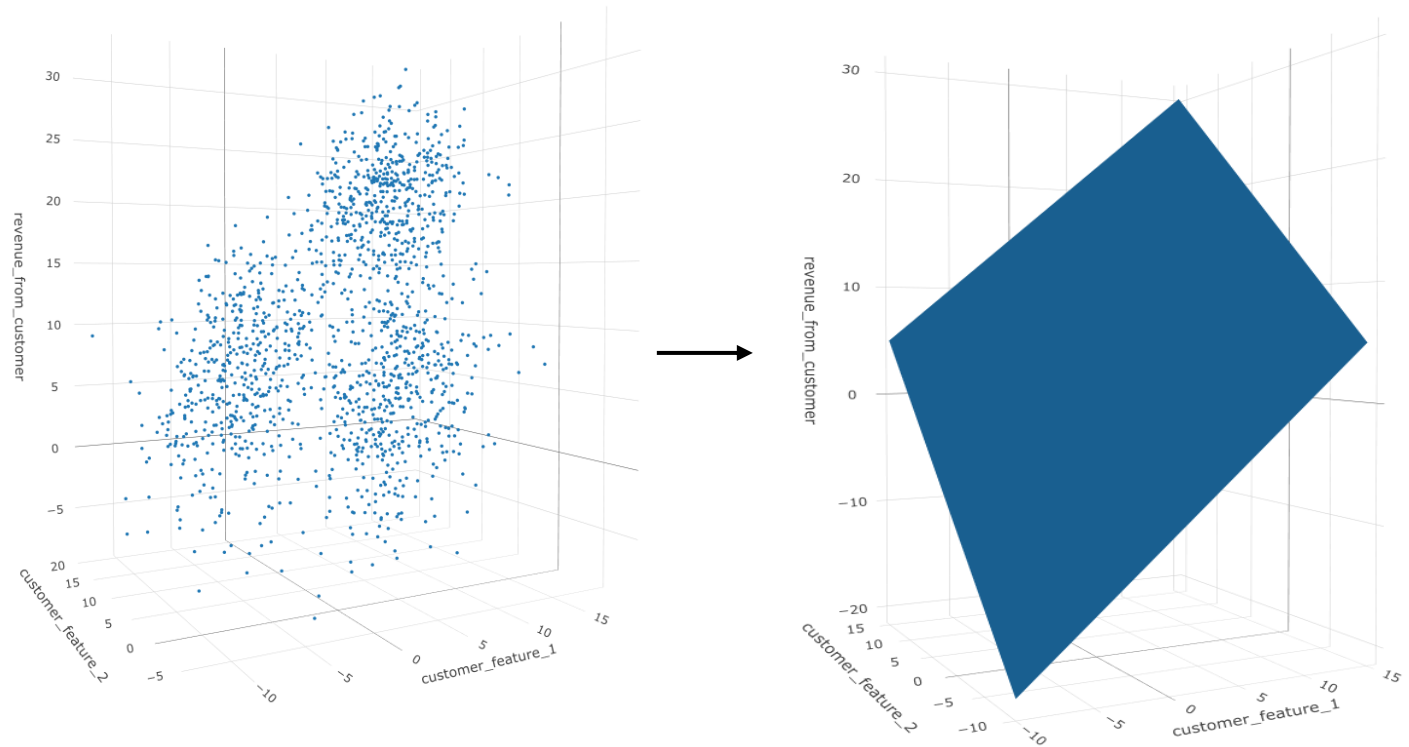
Classification

Regression Example

- The label is a numerical variable
- Want to predict the label for a new observation

Income (Customer Feature 1)	Age (Customer Feature 2)	Revenue from Customer
22003	45	14.03875
57230	54	23.31168
75137	28	24.05046
31208	54	18.5386
54078	23	18.50195
44413	44	20.63106
55237	46	22.32953

Training data



$$\text{Predicted Revenue} = \hat{y}(\text{Income}, \text{Age}) = w_1^* \times \text{Income} + w_2^* \times \text{Age} + w_0^*$$

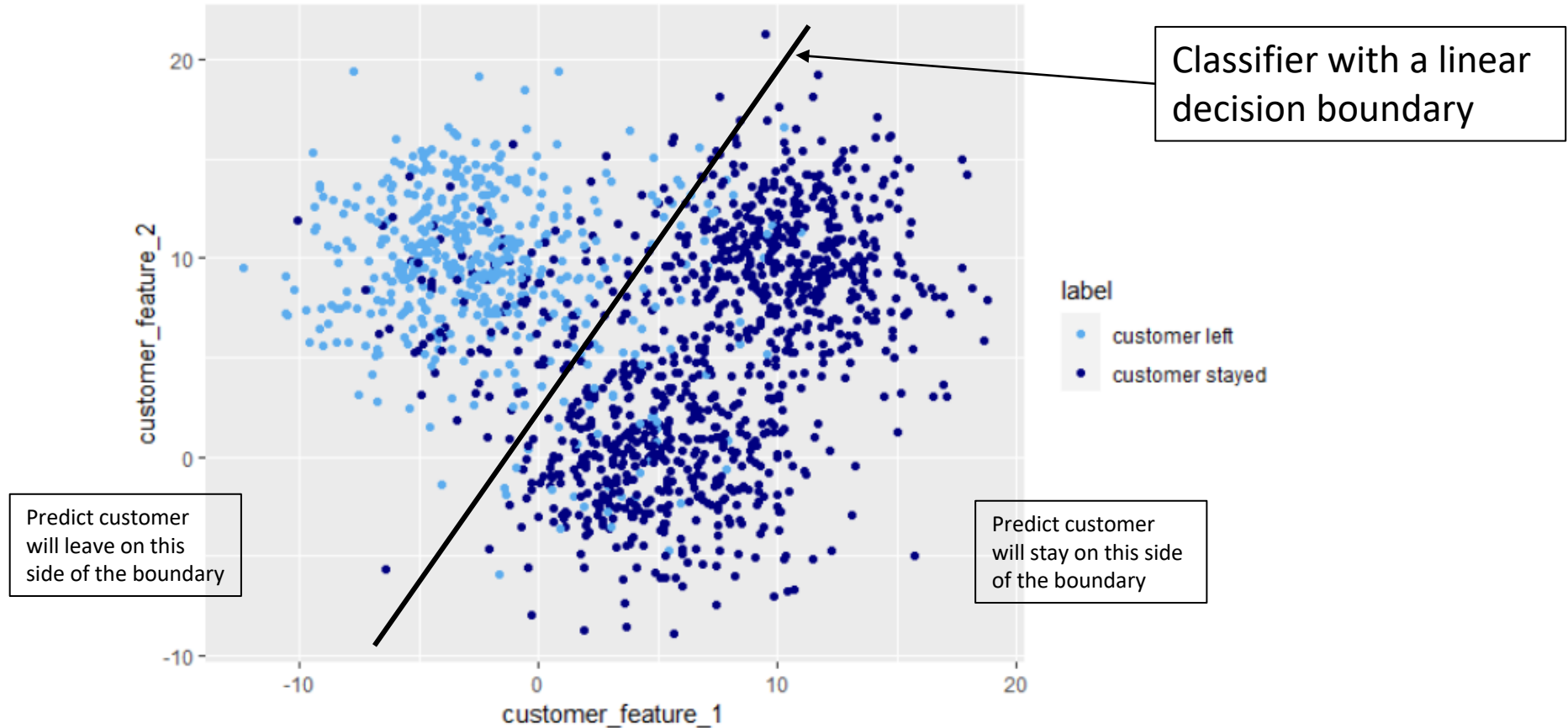
Linear Regression Equation

Classification Example

- The label is a categorical variable with some number of levels called classes
- Want to predict the class for a new observation

Income (Customer Feature 1)	Age (Customer Feature 2)	Customer Left Indicator
22003	45	1
57230	54	0
75137	28	0
31208	54	1
54078	23	0
44413	44	1
55237	46	0

Training data

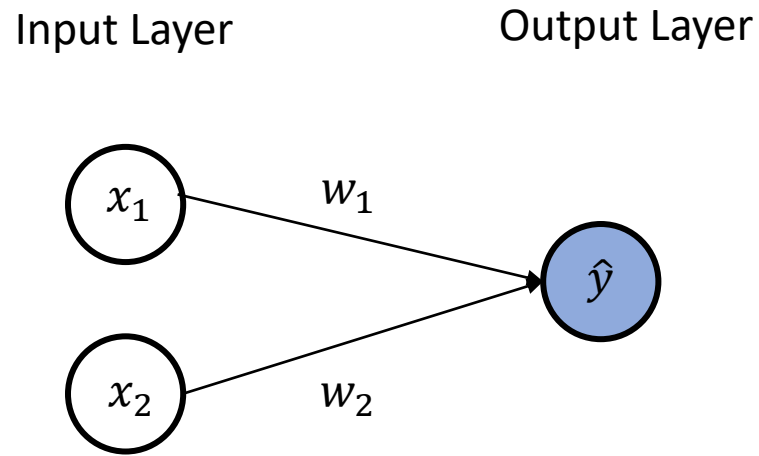


$$\text{Predicted Probability of Leaving} = \hat{y}(\text{Income}, \text{Age}) = \frac{1}{1 + \exp(-(w_1^* \times \text{Income} + w_2^* \times \text{Age} + w_0^*))}$$

Logistic Regression Equation

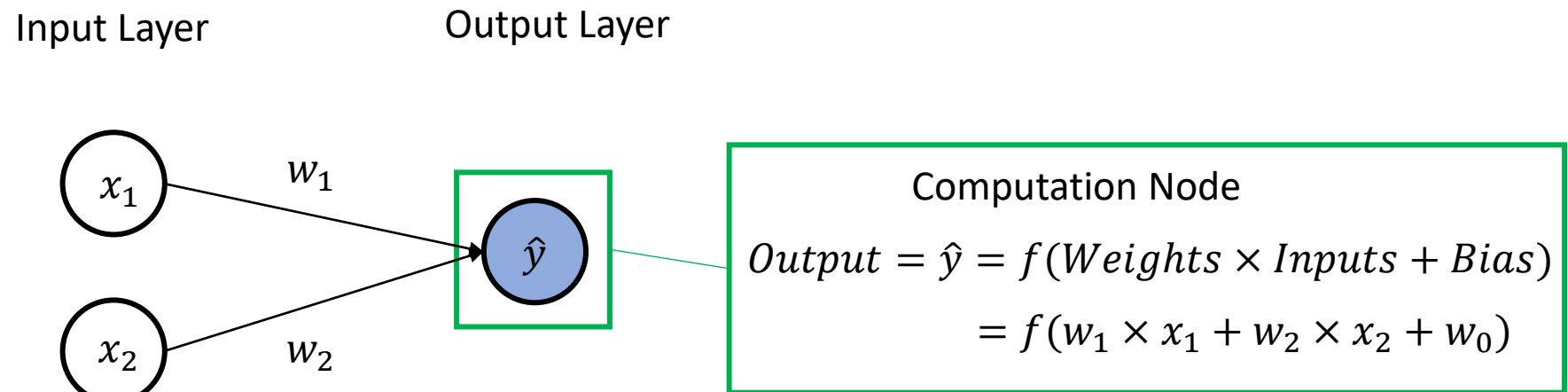
Neural Networks and Deep Learning

- Review: linear and logistic regression can be expressed as perceptrons: neural networks with only an input layer and output layer



Neural Networks and Deep Learning

- The computation in a neural network takes place in computation nodes
 - Each computation node has an input and an output
- In a perceptron, the only computation node is in the output layer
 - The output of the computation node is the output (\hat{y}) of our neural network model
 - The function f is called the activation function and it provides the expressive power of neural networks



Set $f(x) = x$ to get linear regression

Use training data to find optimal weight values w_1^* , w_2^* , w_0^*

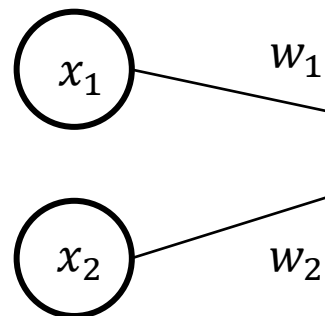
Neural Networks and Deep Learning

- The computation in a neural network takes place in computation nodes
 - Each computation node has an input and an output
- In a perceptron, the only computation node is in the output layer
 - The output of the computation node is the output (\hat{y}) of our neural network model
 - The function f is called the activation function and it provides the expressive power of neural networks

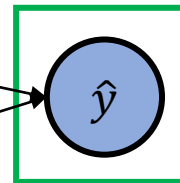
Income (Customer Feature 1)	Age (Customer Feature 2)	Revenue from Customer
22003	45	14.03875
57230	54	23.31168
75137	28	24.05046
31208	54	18.5386
54078	23	18.50195
44413	44	20.63106
55237	46	22.32953

Training data

Input Layer



Output Layer

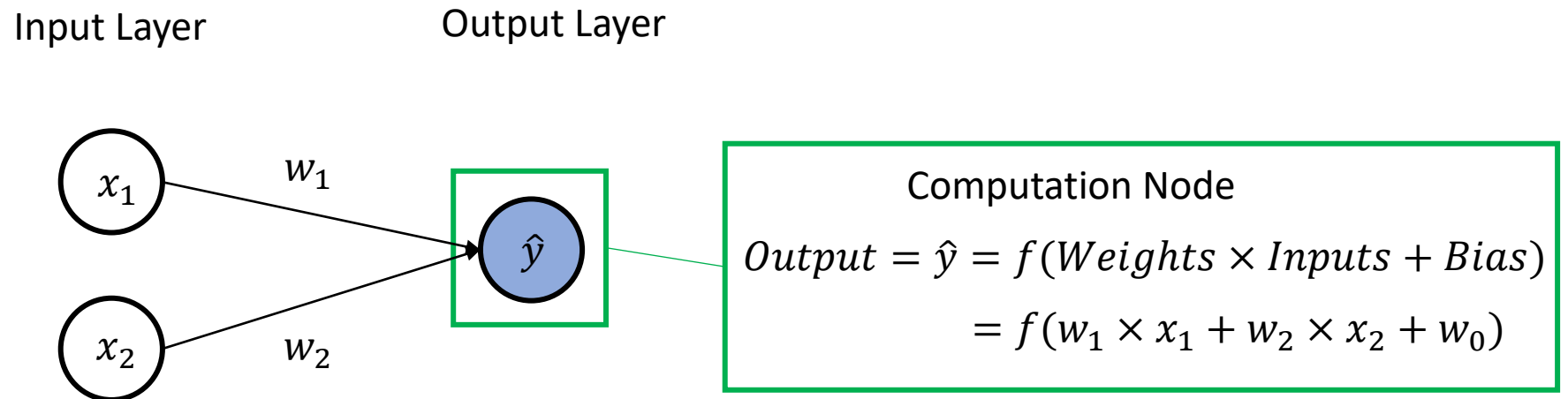


$$\hat{y} = \text{Predicted Revenue} = w_1 \times 22003 + w_2 \times 45 + w_0$$

Calculate predicted revenue for each observation and compare to the actual in the training data

Neural Networks and Deep Learning

- The computation in a neural network takes place in computation nodes
 - Each computation node has an input and an output
- In a perceptron, the only computation node is in the output layer
 - The output of the computation node is the output (\hat{y}) of our neural network model
 - The function f is called the activation function and it provides the expressive power of neural networks



Set $f(x) = \frac{1}{1+e^{-x}}$ to get logistic regression

Use training data to find optimal weight values w_1^* , w_2^* , w_0^*

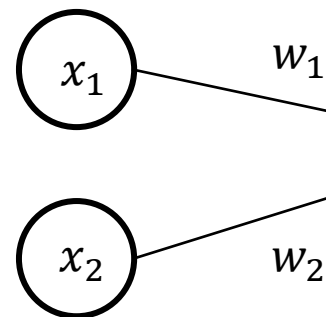
Neural Networks and Deep Learning

- The computation in a neural network takes place in computation nodes
 - Each computation node has an input and an output
- In a perceptron, the only computation node is in the output layer
 - The output of the computation node is the output (y) of our neural network model
 - The function f is called the activation function and it provides the expressive power of neural networks

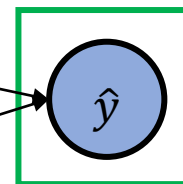
Income (Customer Feature 1)	Age (Customer Feature 2)	Customer Left (Label)
22003	45	1
57230	54	0
75137	28	0
31208	54	1
54078	23	0
44413	44	1
55237	46	0

Training data

Input Layer



Output Layer



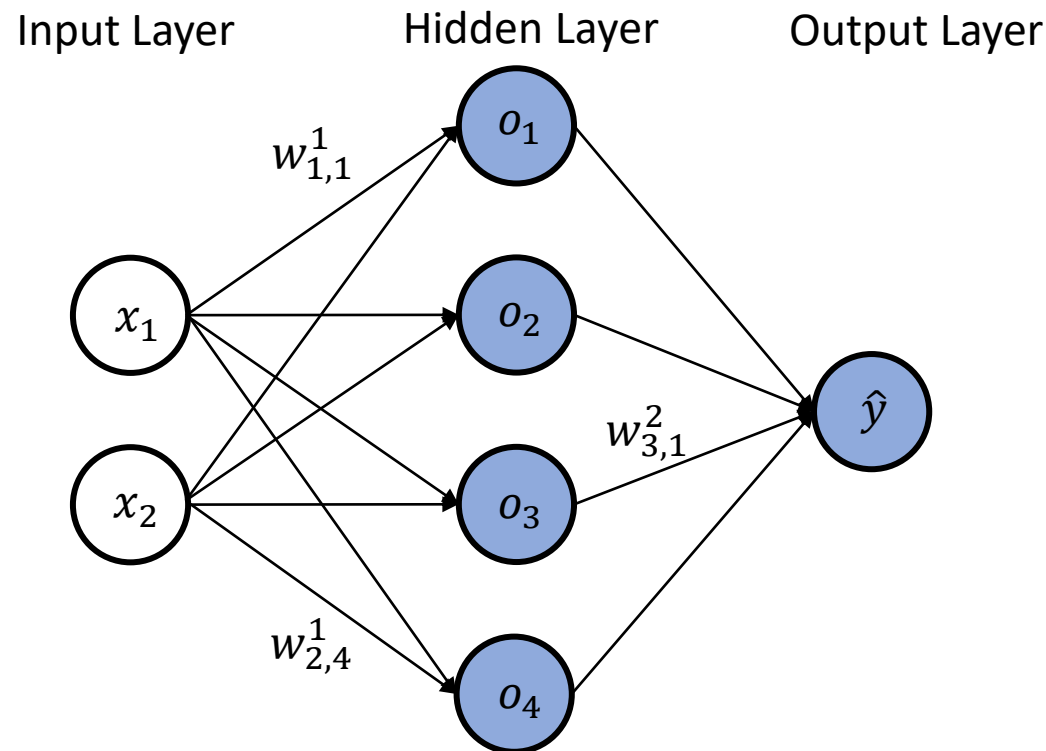
$$\hat{y} = \text{Predicted Probability of Leaving}$$
$$= \frac{1}{1 + e^{-(w_1 \times 22003 + w_2 \times 45 + w_0)}}$$

Calculate predicted probability for each observation and compare to the actual in the training data

If $\hat{y} \geq 0.5$, predict 1 (customer left)

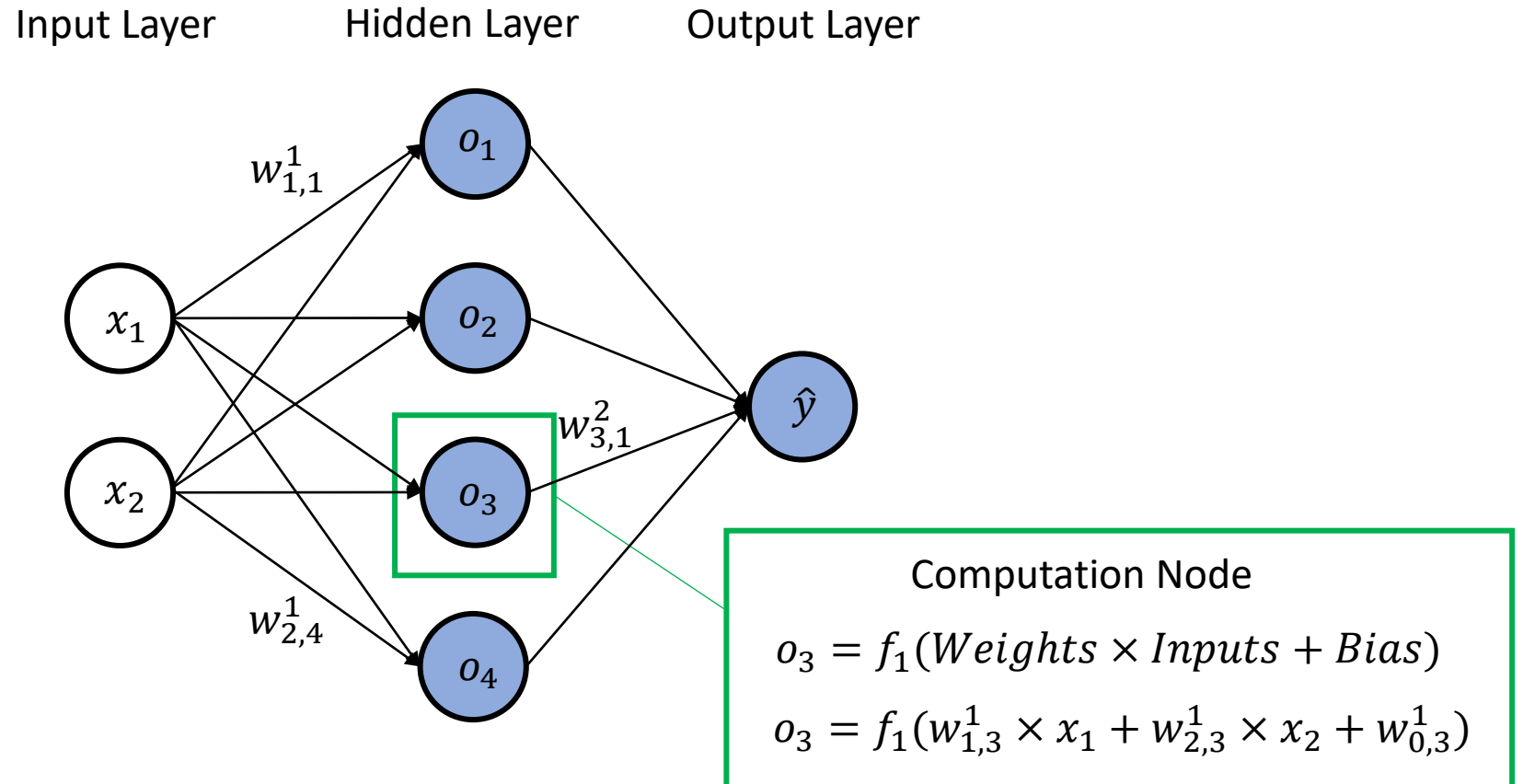
Neural Networks and Deep Learning

- Linear regression and logistic regression are both linear models
 - For realistic problems, the relationships between the dependent variable and independent variables are usually more complex
- Idea: use a composite function $y = g(h(x))$
 - This corresponds to adding hidden layers between the input and output layers



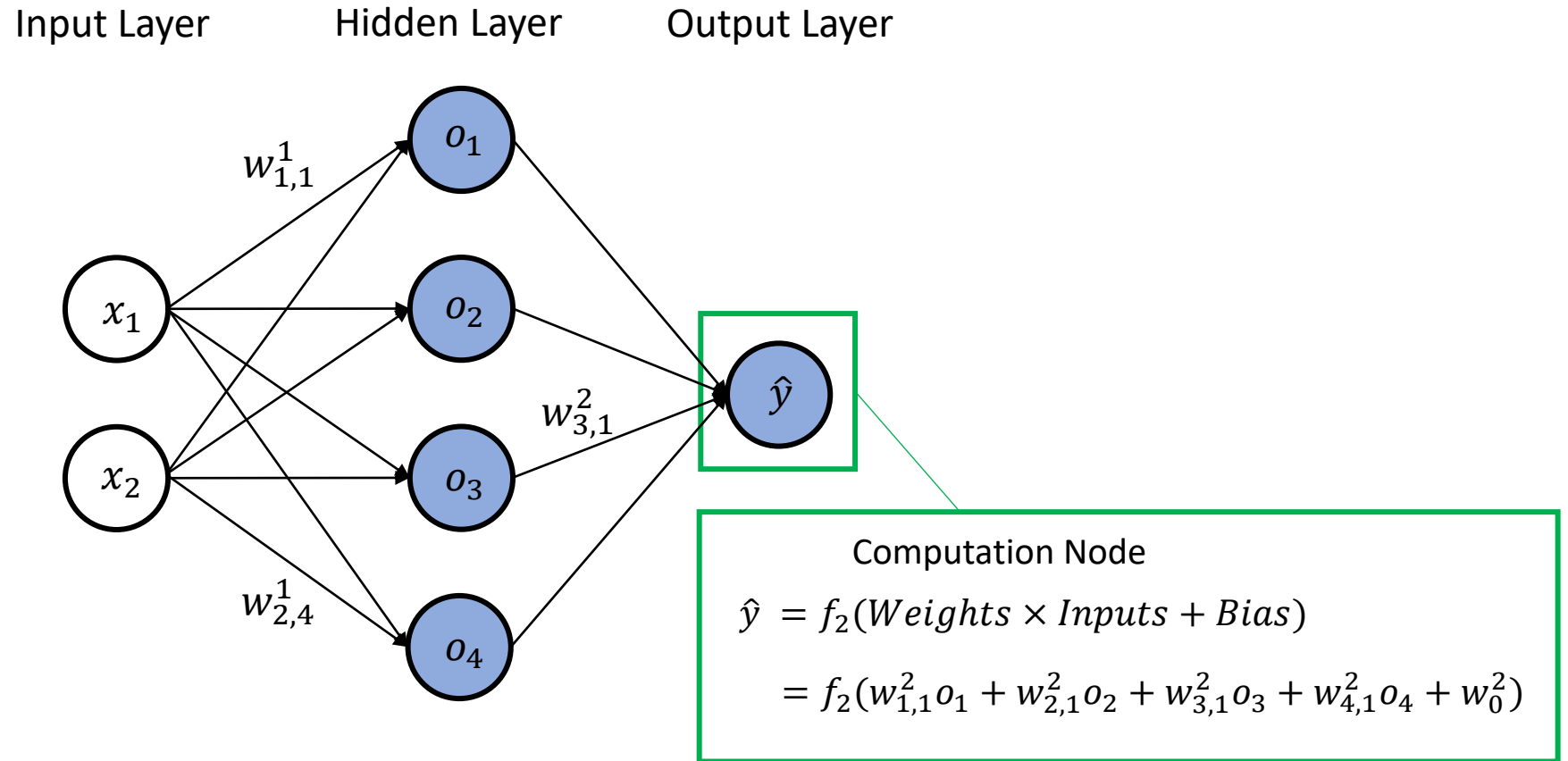
Neural Networks and Deep Learning

- Now, a layer of computation nodes are added, which can significantly increase the expressive power of the neural network
 - I.e., it can capture more complex relationships in the data



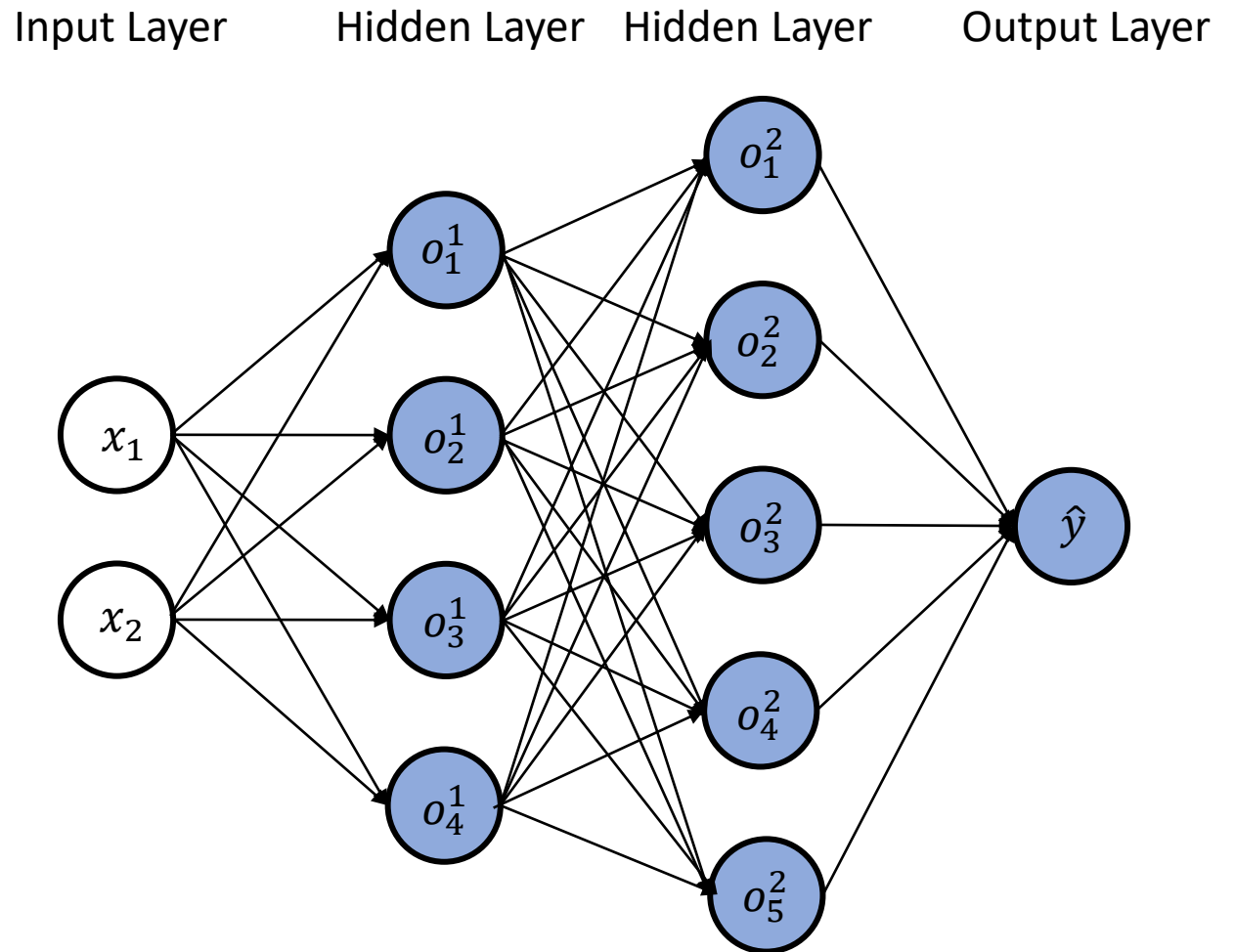
Neural Networks and Deep Learning

- The output of the neural network model can now be highly non-linear
 - Each layer can use a different activation function (e.g., f_1 versus f_2)
 - The non-linearity of the activation functions is critical
 - If only linear activation functions are used, the result is only a linear model!



Neural Networks and Deep Learning

- What if we add another hidden layer?
 - Each hidden layer implements a transformation of the data
 - Hopefully, each transformation is a better representation of the data
 - Each hidden layer can have a different number of nodes
 - Each layer can use a different activation function



Training Neural Networks

Neural Networks and Deep Learning

- Weights W are initialized randomly
 - Unless another initialization is specified (e.g., from a previous model or experience)
- Then, compare the predictions of the model with the actual labels in the training data
- Based on the comparison, update the weights to (hopefully) get the predictions closer to the actual labels
- Repeat the previous two steps until stopping criterion is met
- How do we make the comparison and update the weights to find the optimal values W^* ?

Neural Networks and Deep Learning

- How do we make the comparison?
 - This is the role of the loss function, which is *only* a function of the weights W , given the training data and architecture of the NN

$$Loss(W; \text{training set, architecture of NN})$$

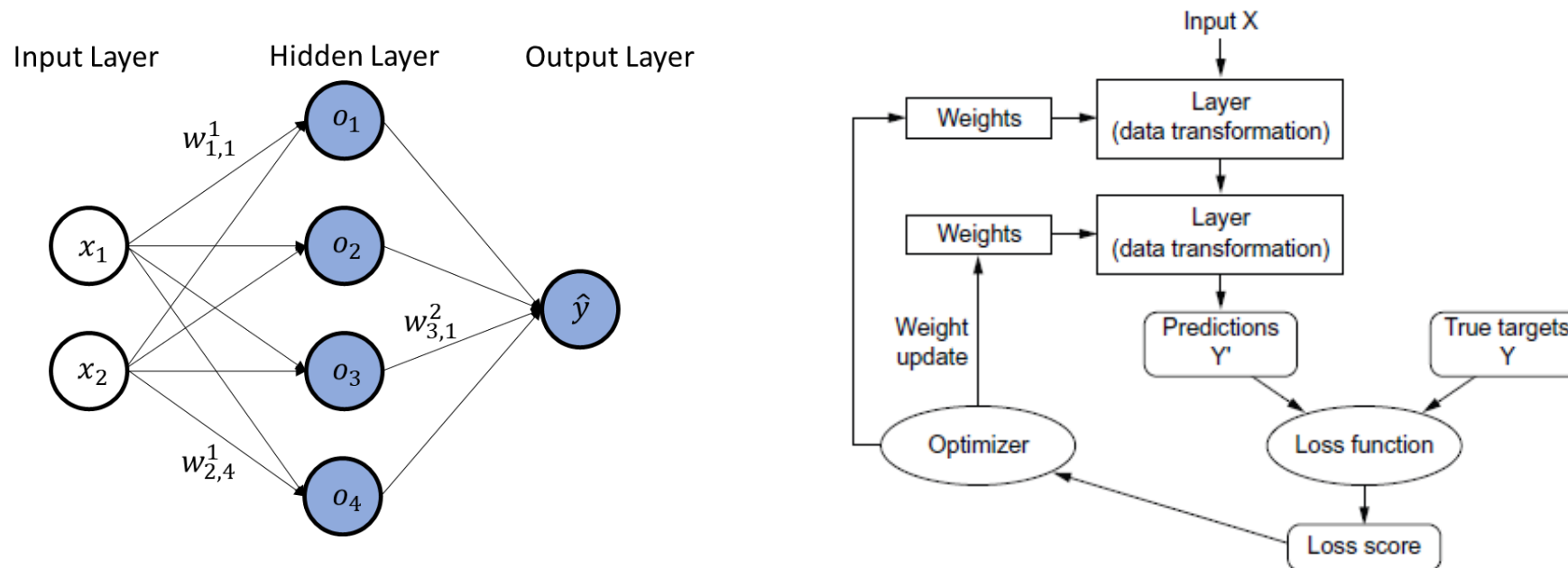
- Larger value of the loss means the compared values were further away
- Smaller value of the loss means the compared values were closer

Details of Training Process

- Important: for a particular training set and architecture, the loss is only a function of the weights W

$$\text{Loss}(W; \text{training set, architecture of NN})$$

- The loss function is used along with an optimization algorithm to determine how to change the weights to improve the loss
- The goal of the optimization is to find the optimal weight values W^* that minimize the loss



Neural Networks and Deep Learning

- To calculate the loss for a given set of weight values W
 1. Each x in training set is put through the network to obtain a prediction $\hat{y}(x ; W)$
 2. Calculate the loss function (example below using mean squared loss function)

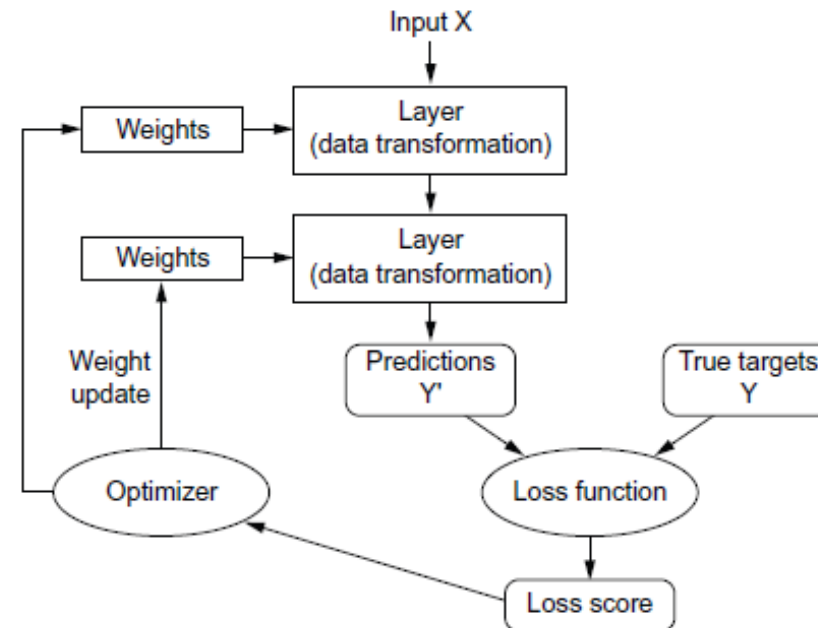
$$Loss(W; \text{training set, architecture of NN}) = \frac{1}{n} \sum_{i=1}^n (\hat{y}(x_i ; W) - y_i)^2$$

x_1, x_2, \dots, x_n are training observations and y_1, y_2, \dots, y_n are actual labels

Neural Networks and Deep Learning

- When the training set is too large, it is not feasible to calculate the loss based on the entire training set each time
 - Solution: use batches consisting of a subset of rows of training data
 - Take for example a batch size of 2
 - Iterate through the training data taking two rows at a time
 - On each iteration, define the loss function
 $Loss(W | \text{selected two rows of training data, architecture of NN})$
 - After iterating through the entire training set, we have completed an epoch

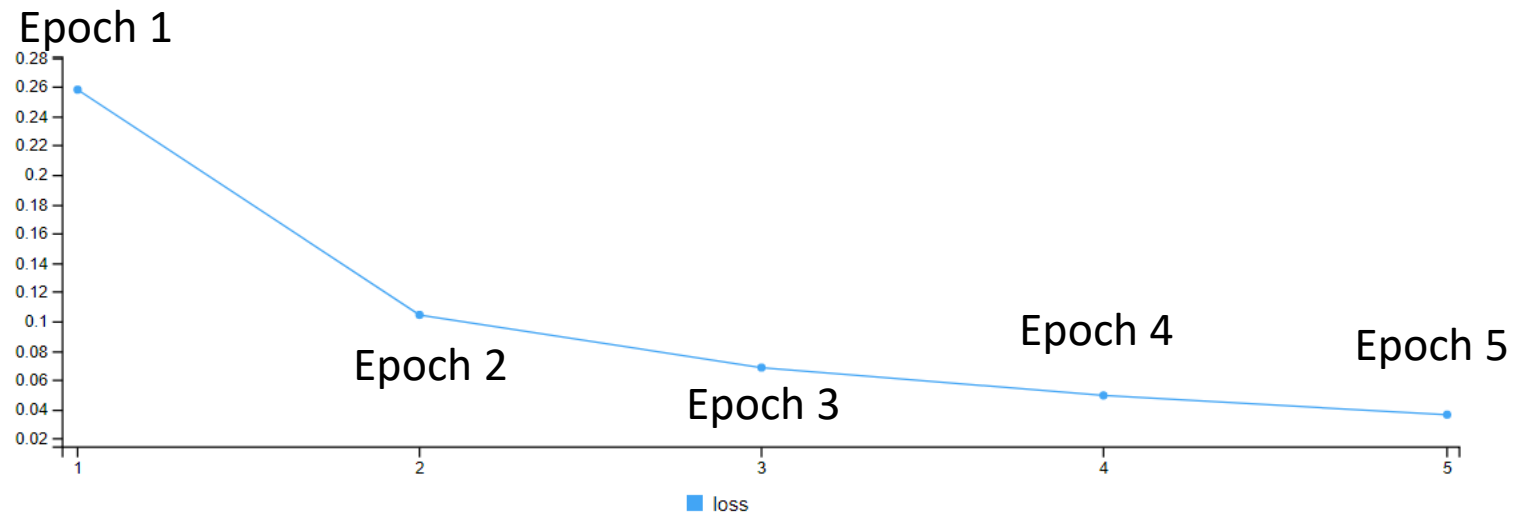
Income (Customer Feature 1)	Age (Customer Feature 2)	Revenue from Customer	Predicted Revenue Given Set of Weights W
22003	45	14.03875	$\hat{y}(22003, 45 W)$
57230	54	23.31168	$\hat{y}(57230, 54 W)$
75137	28	24.05046	$\hat{y}(75137, 28 W)$
31208	54	18.5386	$\hat{y}(31208, 54 W)$
54078	23	18.50195	$\hat{y}(54078, 23 W)$
44413	44	20.63106	$\hat{y}(44413, 44 W)$
55237	46	22.32953	$\hat{y}(55237, 46 W)$



Neural Networks and Deep Learning

- For large training sets, it is not feasible to calculate the loss based on the entire training set each time
 - Solution: use batches consisting of a subset of rows of training data
 - Take for example a batch size of 2
 - Iterate through the training data taking two rows at a time
 - On each iteration, define the loss function $Loss(W | \text{selected two rows of training data, architecture of NN})$
 - Use this loss function in conjunction with optimization algorithm to decide how to update weights
- After iterating through the entire training set, we have completed one epoch
 - We specify the number of epochs in advance

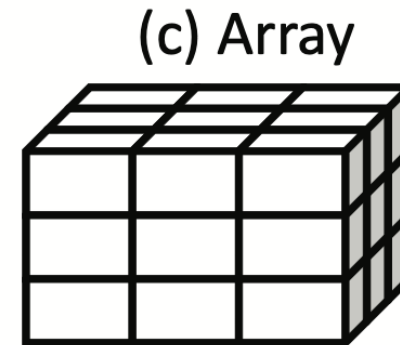
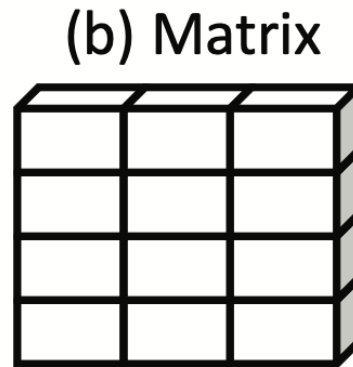
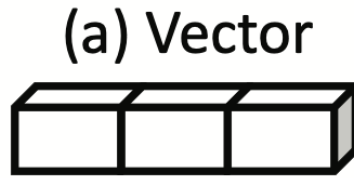
Income (Customer Feature 1)	Age (Customer Feature 2)	Revenue from Customer	Predicted Revenue Given Set of Weights W
22003	45	14.03875	$\hat{y}(22003, 45 W)$
57230	54	23.31168	$\hat{y}(57230, 54 W)$
75137	28	24.05046	$\hat{y}(75137, 28 W)$
31208	54	18.5386	$\hat{y}(31208, 54 W)$
54078	23	18.50195	$\hat{y}(54078, 23 W)$
44413	44	20.63106	$\hat{y}(44413, 44 W)$
55237	46	22.32953	$\hat{y}(55237, 46 W)$



Data Structures for TensorFlow

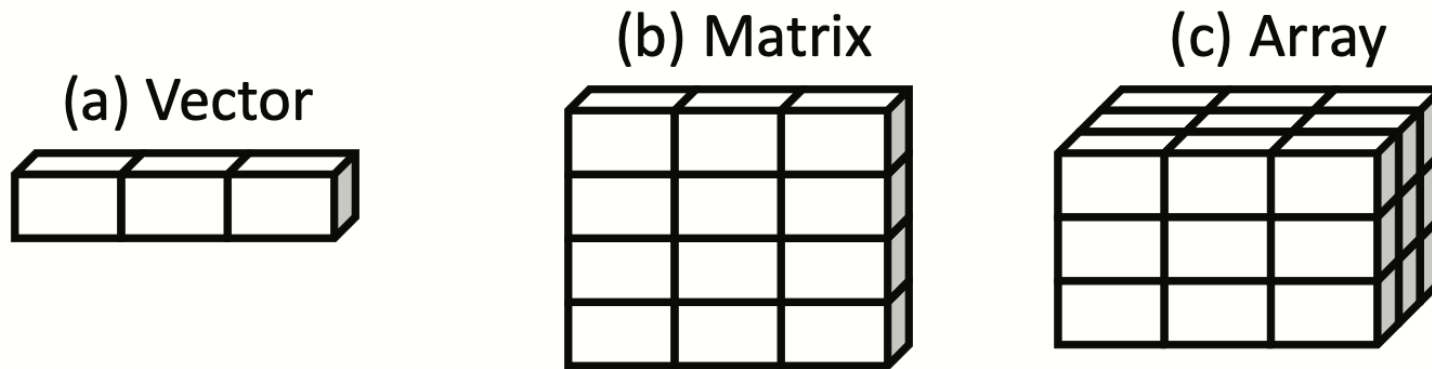
Tensors

- Tensors are data structures that hold the data that we input into TensorFlow and Keras functions
- An array in R is a tensor



Tensors

- Tensors are data structures that hold the data that we input into TensorFlow and Keras functions
- An array in R is a tensor

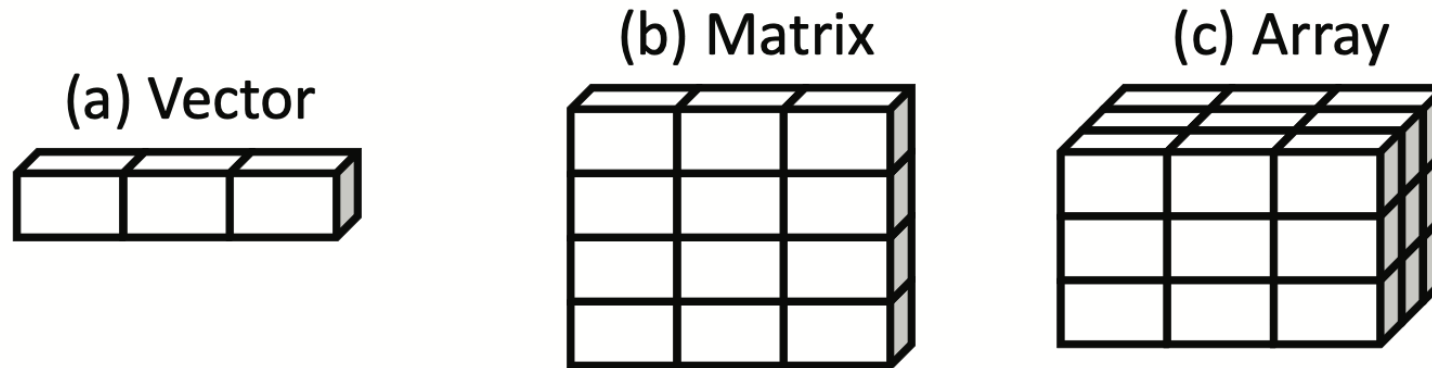


2.2.2 Vectors (rank 1 tensors)

An array of numbers is called a *vector*, or rank 1 tensor, or 1D tensor. A rank 1 tensor is said to have exactly one axis.

Tensors

- Tensors are data structures that hold the data that we input into TensorFlow and Keras functions
- An array in R is a tensor



2.2.3 Matrices (rank 2 tensors)

An array of vectors is a *matrix*, or rank 2 tensor, or 2D tensor. A matrix has two axes (often referred to as *rows* and *columns*). You can visually interpret a matrix as a rectangular grid of numbers:

When each vector is an observation, it is called ***vector data***

Income	Age
22003	45
57230	54
75137	28
31208	54
54078	23
44413	44
55237	46

Tensors

- Tensors are data structures that hold the data that we input into TensorFlow and Keras functions
- An array in R is a tensor

2.2.5 Key attributes

A tensor is defined by the following three key attributes:

- *Number of axes (rank)*—For instance, a rank 3 tensor has three axes, and a matrix has two axes. This is available from `length(dim(x))`.
- *Shape*—This is an integer vector that describes how many dimensions the tensor has along each axis. For instance, the previous matrix example has shape `(3, 5)`, and the rank 3 tensor example has shape `(2, 3, 4)`. A vector has a shape with a single element, such as `(5)`. R arrays don't distinguish between 1D vectors and scalar tensors, but conceptually, tensors can also be scalar with shape `()`.
- *Data type*—This is the type of the data contained in the tensor. R arrays have support for R's built-in data types like `double` and `integer`. Conceptually, however, tensors can support any type of homogeneous data type, and other tensor implementations also provide support for types like `float16`, `float32`, `float64` (corresponding to R's `double`), `int32` (R's `integer` type), and so on. In TensorFlow, you are also likely to come across string tensors.

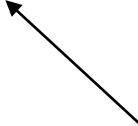
Tensors

- Tensors are data structures that hold the data that we input into TensorFlow and Keras functions
- An array in R is a tensor

2.2.8 Real-world examples of data tensors

Let's make data tensors more concrete with a few examples similar to what you'll encounter later. The data you'll manipulate will almost always fall into one of the following categories:

- *Vector data*—Rank 2 tensors of shape (samples, features), where each sample is a vector of numerical attributes (“features”)
- *Times-series data or sequence data*—Rank 3 tensors of shape (samples, timesteps, features), where each sample is a sequence (of length timesteps) of feature vectors
- *Images*—Rank 4 tensors of shape (samples, height, width, channels), where each sample is a 2D grid of pixels, and each pixel is represented by a vector of values (“channels”)
- *Video*—Rank 5 tensors of shape (samples, frames, height, width, channels), where each sample is a sequence (of length frames) of images



Income	Age
22003	45
57230	54
75137	28
31208	54
54078	23
44413	44
55237	46

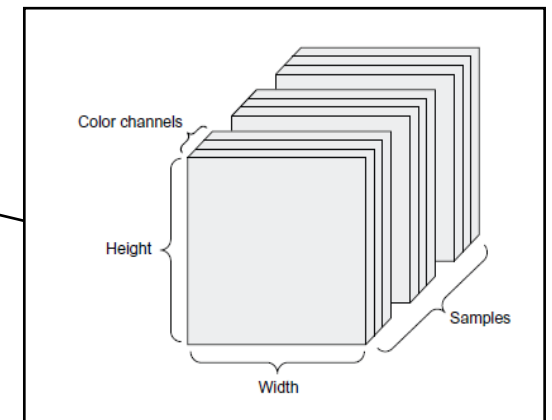
Tensors

- Tensors are data structures that hold the data that we input into TensorFlow and Keras functions
- An array in R is a tensor

2.2.8 Real-world examples of data tensors

Let's make data tensors more concrete with a few examples similar to what you'll encounter later. The data you'll manipulate will almost always fall into one of the following categories:

- *Vector data*—Rank 2 tensors of shape (samples, features), where each sample is a vector of numerical attributes (“features”)
- *Times-series data or sequence data*—Rank 3 tensors of shape (samples, timesteps, features), where each sample is a sequence (of length timesteps) of feature vectors
- *Images*—Rank 4 tensors of shape (samples, height, width, channels), where each sample is a 2D grid of pixels, and each pixel is represented by a vector of values (“channels”)
- *Video*—Rank 5 tensors of shape (samples, frames, height, width, channels), where each sample is a sequence (of length frames) of images



Visual for In-Class R Lab

