Neural networks through ice cream sales

Objectives:

- To ***fluently*** use the ***vocabulary*** of neural networks
- To connect ***familiar past algorithms*** to ***neural networks***, and therefore demystify the math
- To practice ***drawing*** and ***specifying*** the parameters of a neural net
- To list the ***parameters*** one can ***adjust*** when building a neural net

Scenario

You own a chain of ice cream stores.

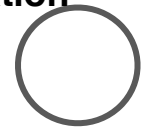
You want to build a model that will predict the sales numbers of a store, given the store's location, pricing of product, and perceived quality of the product.

Simpler models haven't produced great results, so you want to try a neural network. Plus, neural networks sound fancy. You like fancy.

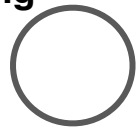
Problem summary

Variables

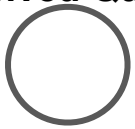
Location



Pricing

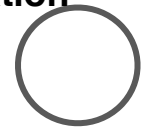


Perceived Quality

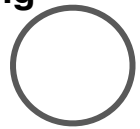


Variables

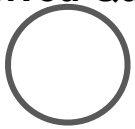
Location



Pricing

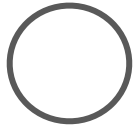


Perceived Quality



Target

Sales



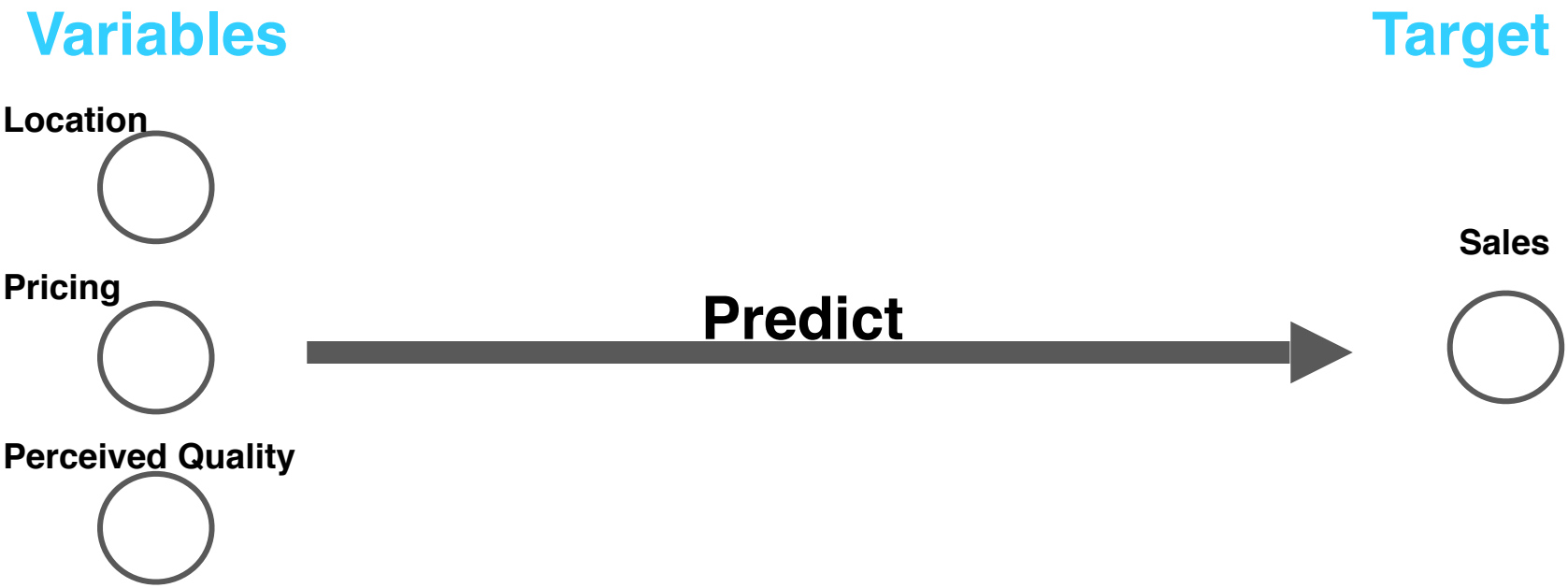
Variables

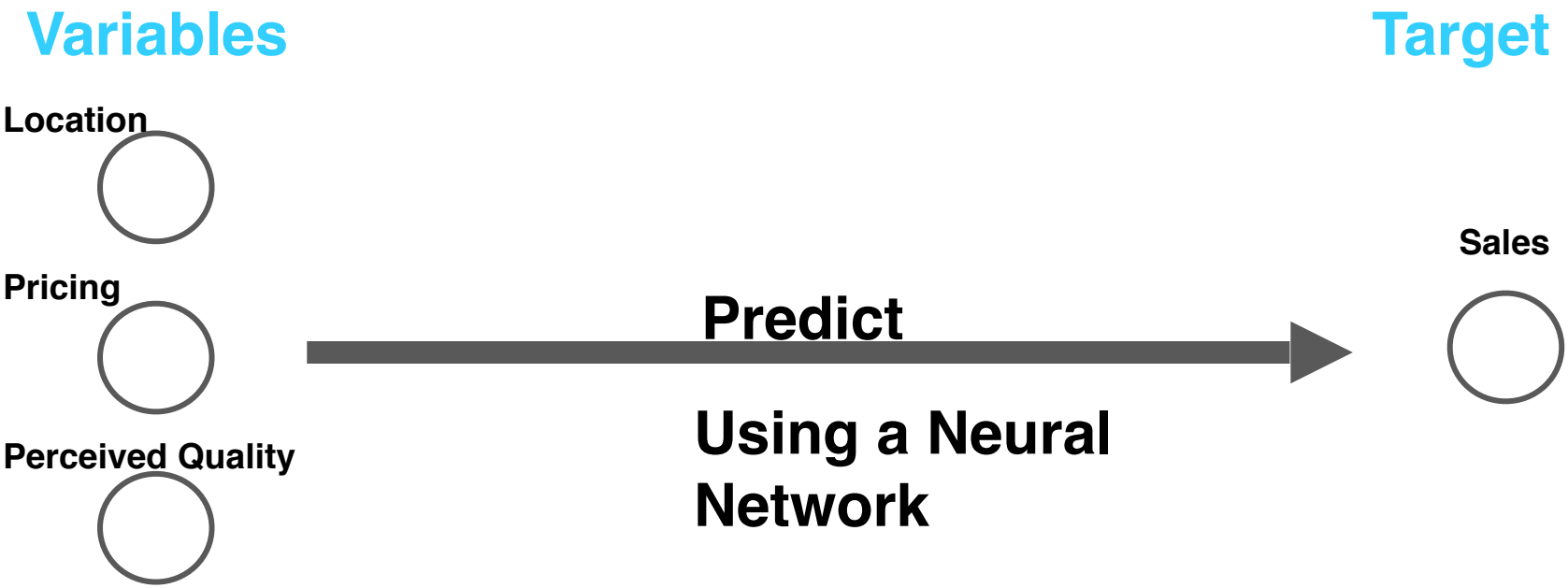
Target

Location

Pricing

Perceived Quality


Variables
Target
Location
Pricing
Perceived Quality
Predict
Sales

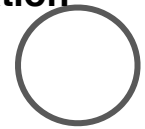

Variables
Target
Location
Pricing
Perceived Quality
Predict
Using a Neural Network
Sales

Vocabulary

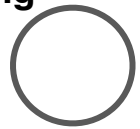
Using a Neural Network

Variables

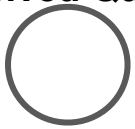
Location



Pricing

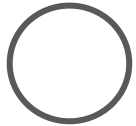


Perceived Quality



Target

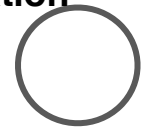
Sales



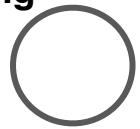
Using a Neural Network

Variables

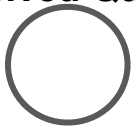
Location



Pricing



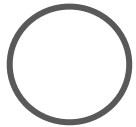
Perceived Quality



input layer

Target

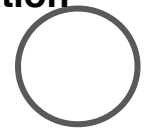
Sales



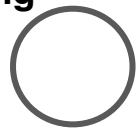
Using a Neural Network

Variables

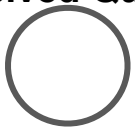
Location



Pricing



Perceived Quality

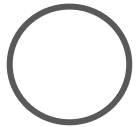


input layer

nodes: 3

Target

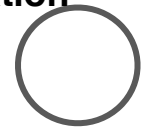
Sales



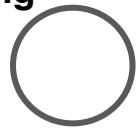
Using a Neural Network

Variables

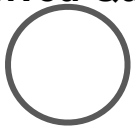
Location



Pricing

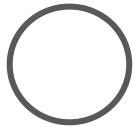


Perceived Quality



Target

Sales



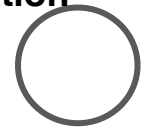
input layer
nodes: 3

output layer

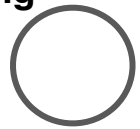
Using a Neural Network

Variables

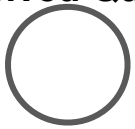
Location



Pricing

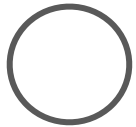


Perceived Quality



Target

Sales



input layer
nodes: 3

output layer
nodes: 1

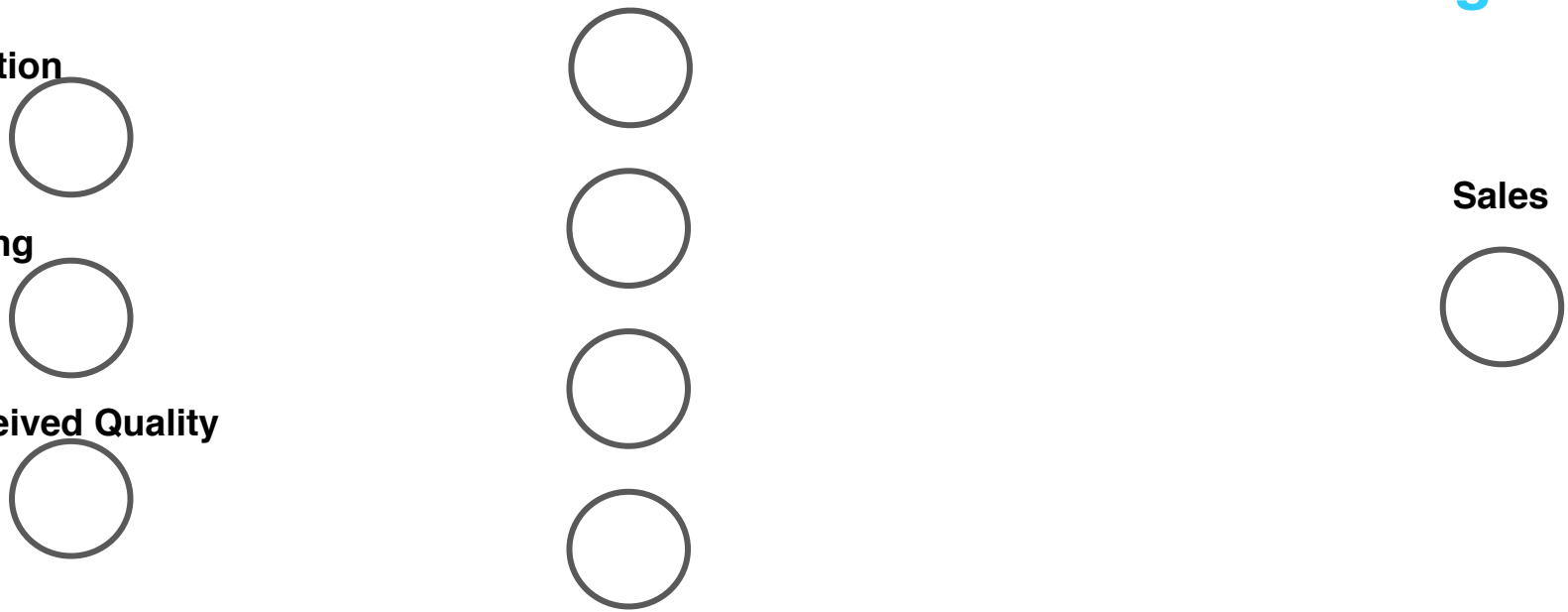
Using a Neural Network

Variables

Location

Pricing

Perceived Quality



Target

Sales

input layer
nodes: 3

output layer
nodes: 1

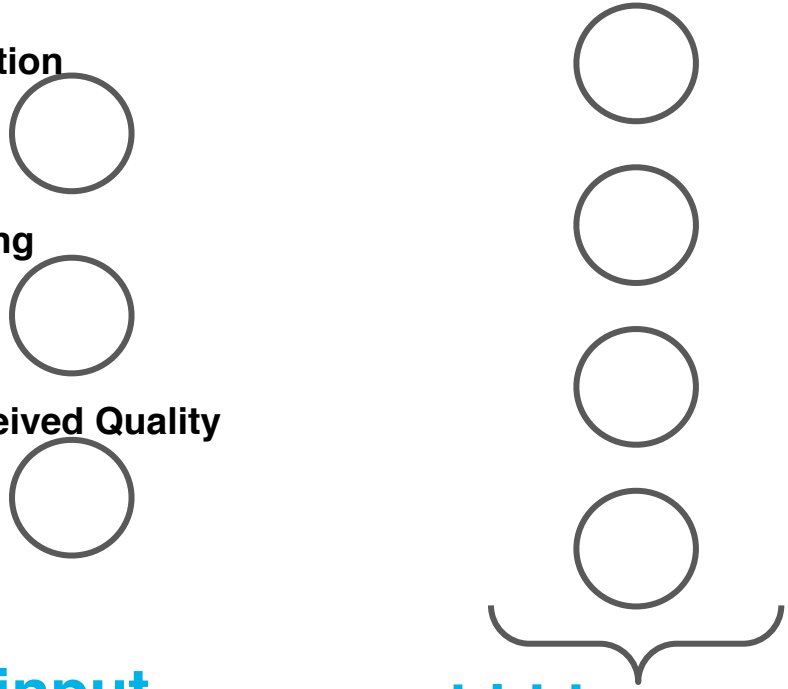
Using a Neural Network

Variables

Location

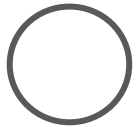
Pricing

Perceived Quality



Target

Sales



input layer
nodes: 3

hidden layer

output layer
nodes: 1

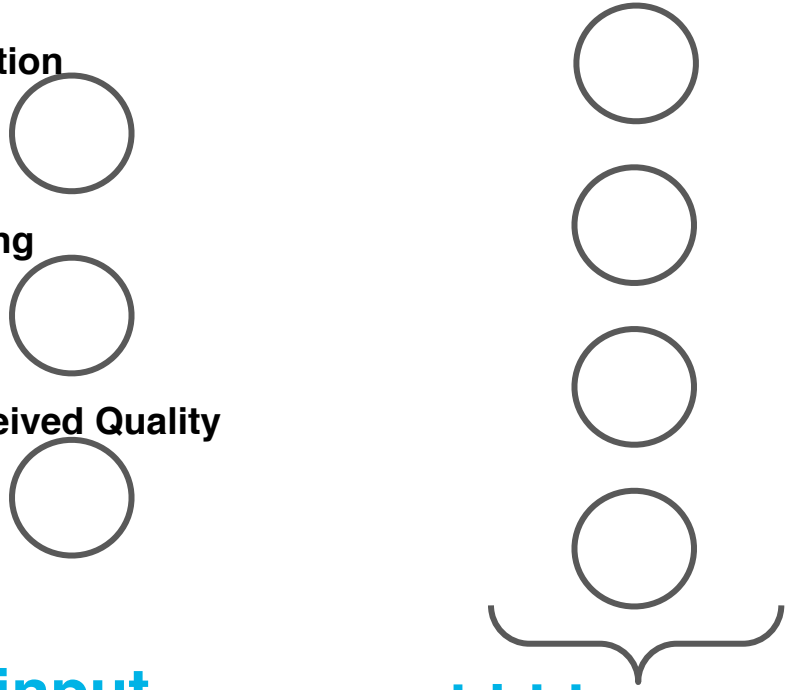
Using a Neural Network

Variables

Location

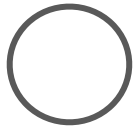
Pricing

Perceived Quality



Target

Sales



input layer
nodes: 3

hidden layer
nodes: 4

output layer
node: 1

How many layers in our neural network?

Keep this in mind as we build neural networks

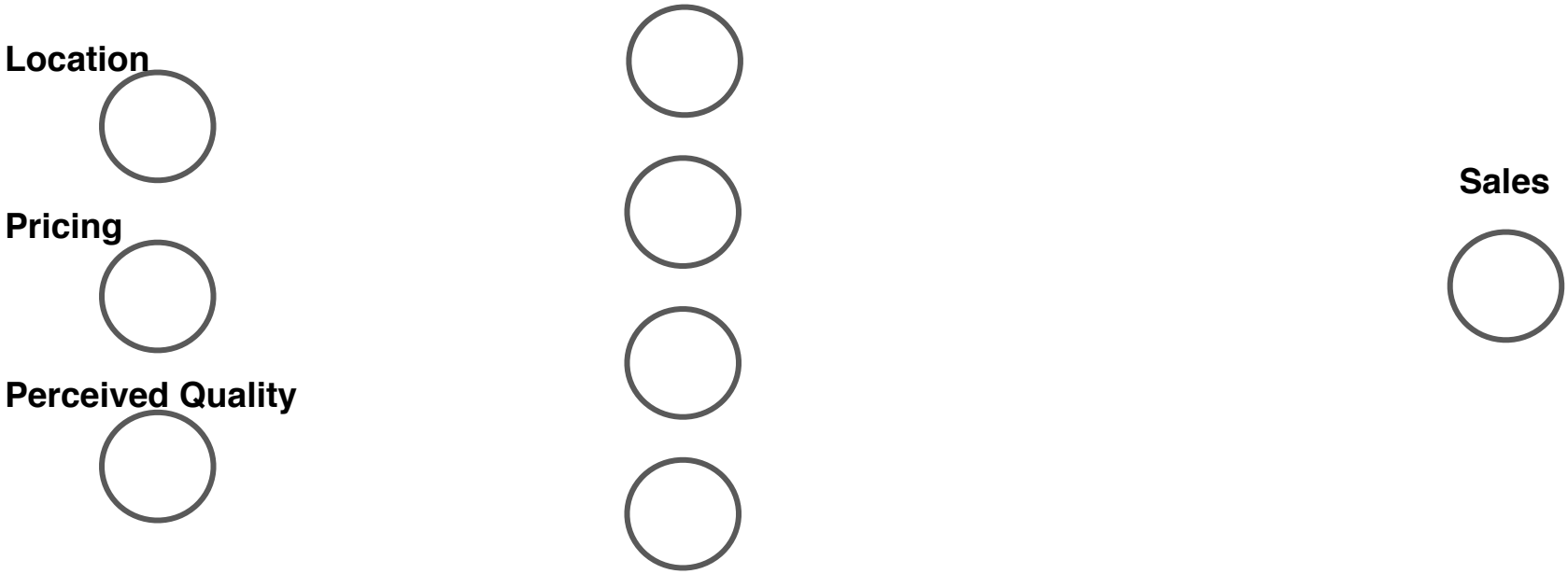
Draw this out:

We want to build a neural network using gender(assume binary), years of education, marital status(single vs wed), and years of employment to predict income.

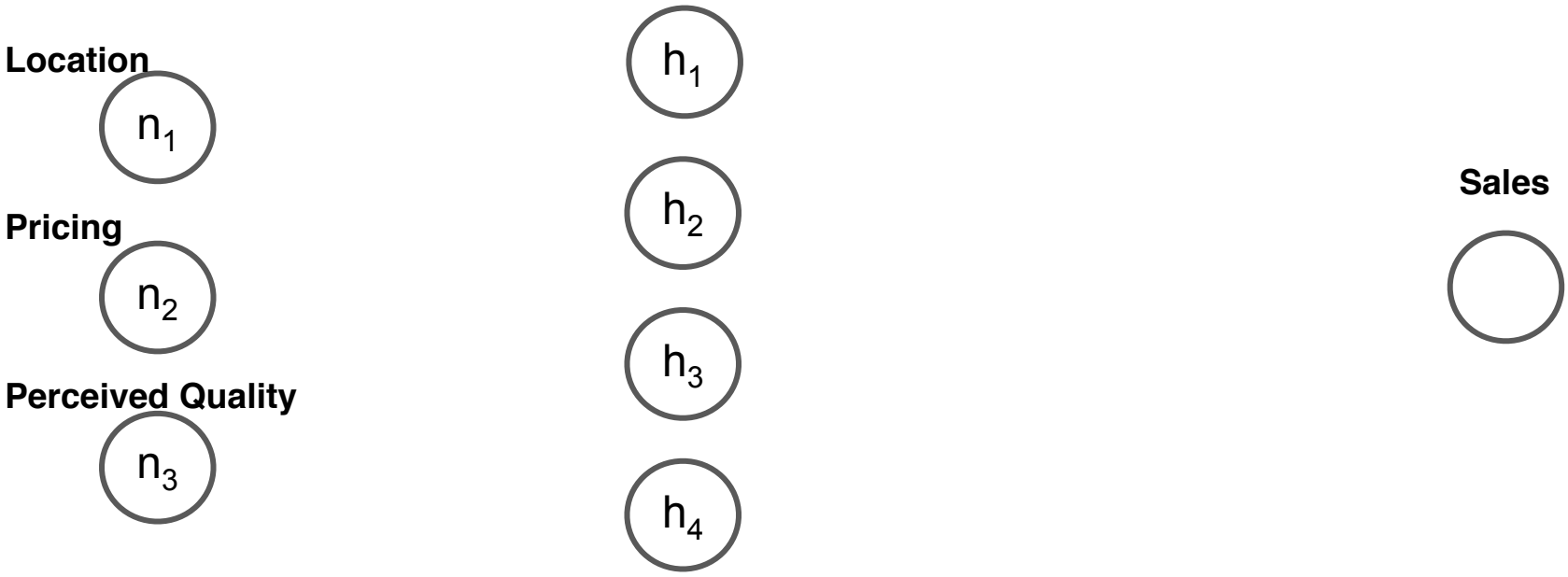
We are going to use two hidden layers. The first one will have three nodes and the second will have 5.

Draw and compare w neighbors.

The math behind networks is not that scary



We need some notation to make this work

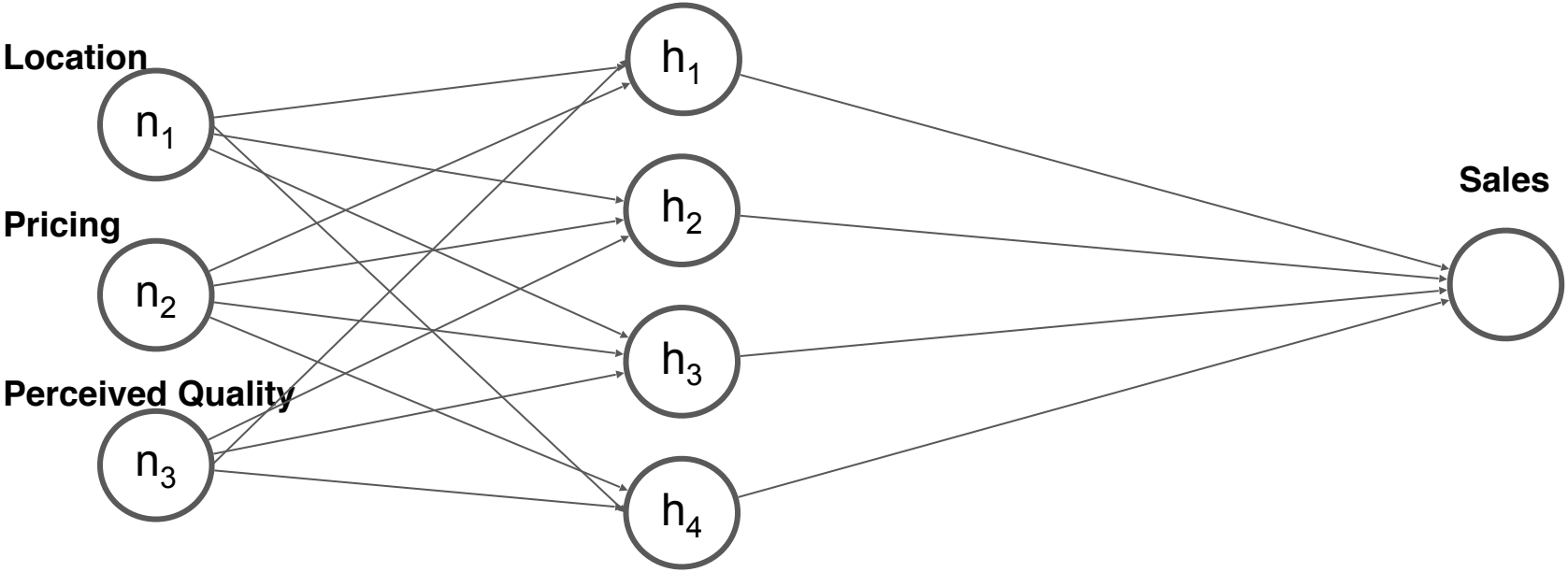


input layer
nodes: 3

hidden layer
nodes: 4

output layer
one node

May have seen diagrams like this

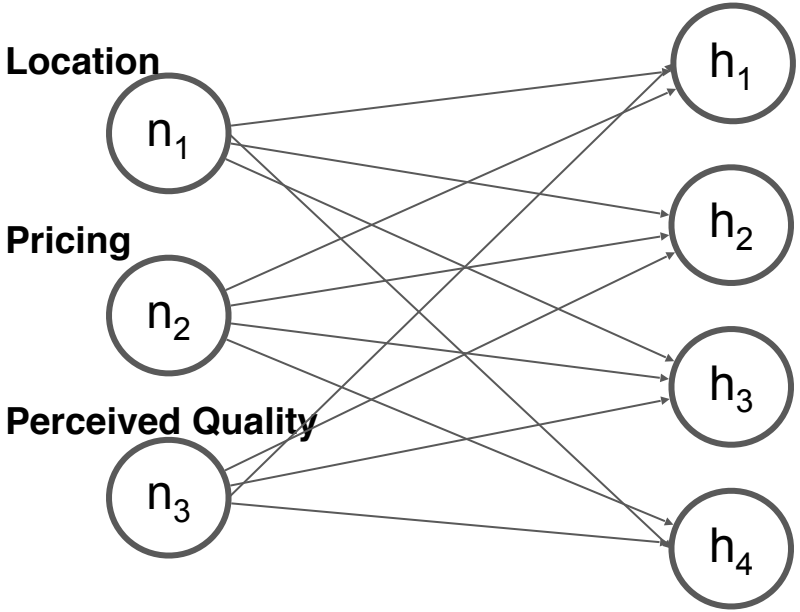


input layer
nodes: 3

hidden layer

output layer
one node

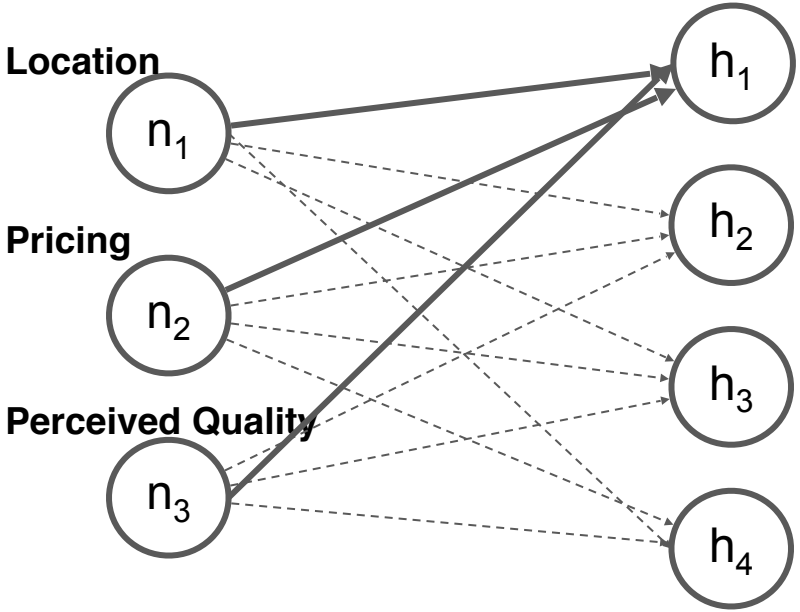
For simplicity we are only going to focus on one layer



input layer
nodes: 3

hidden layer

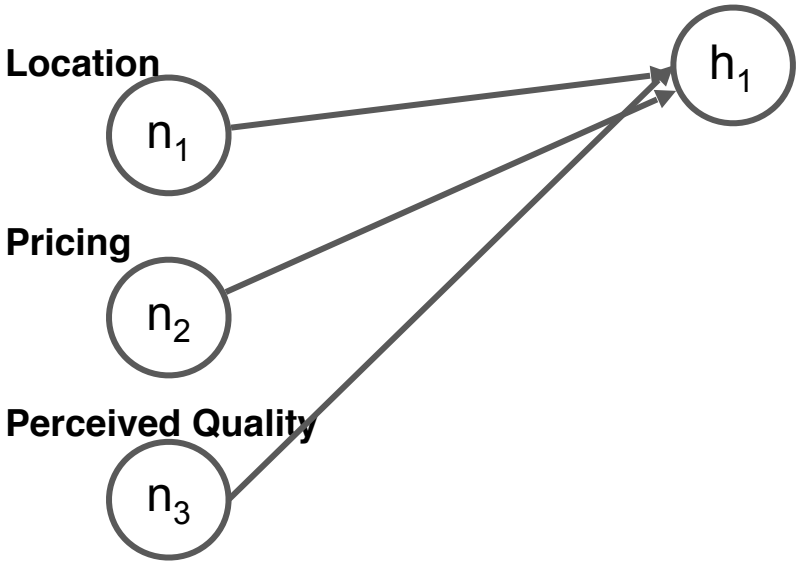
And specifically the first node in the hidden layer



input layer
nodes: 3

hidden layer

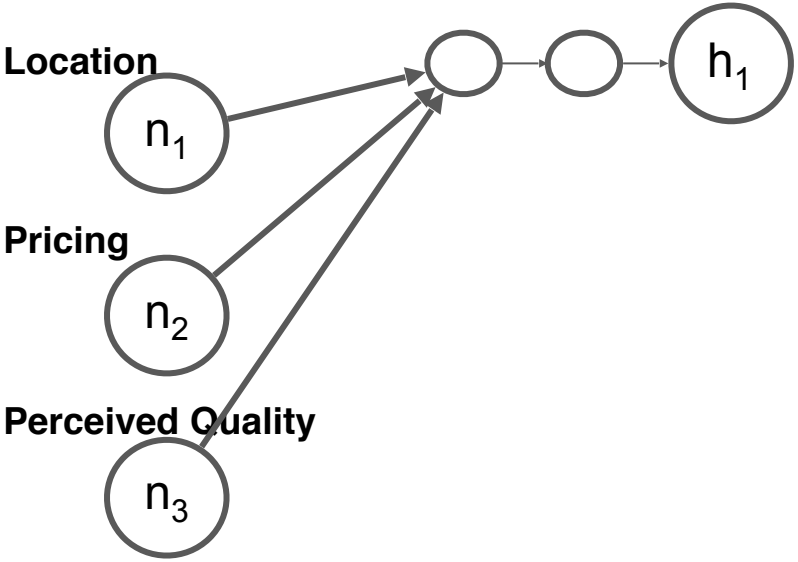
Problem: this common diagram isn't representative



input layer
nodes: 3

hidden layer

What is shown as one is really three

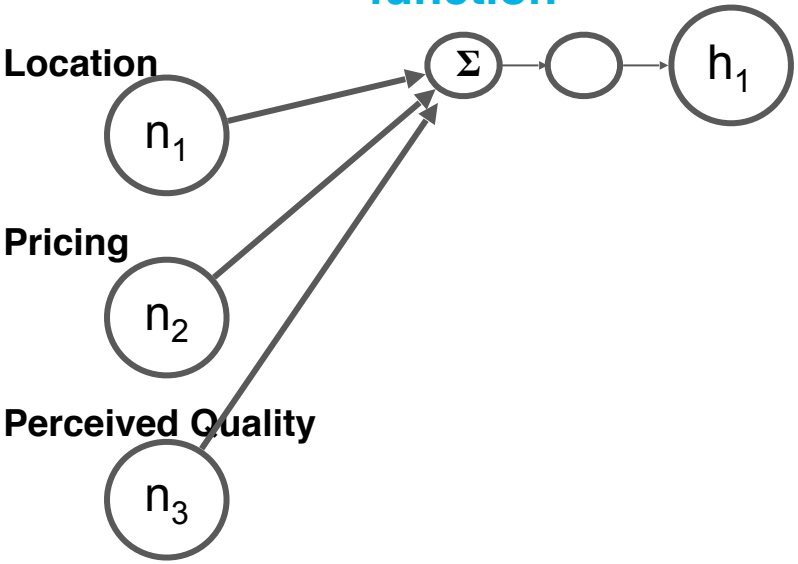


Each node that receives input from previous nodes actually has **three** parts

input layer
nodes: 3

hidden layer

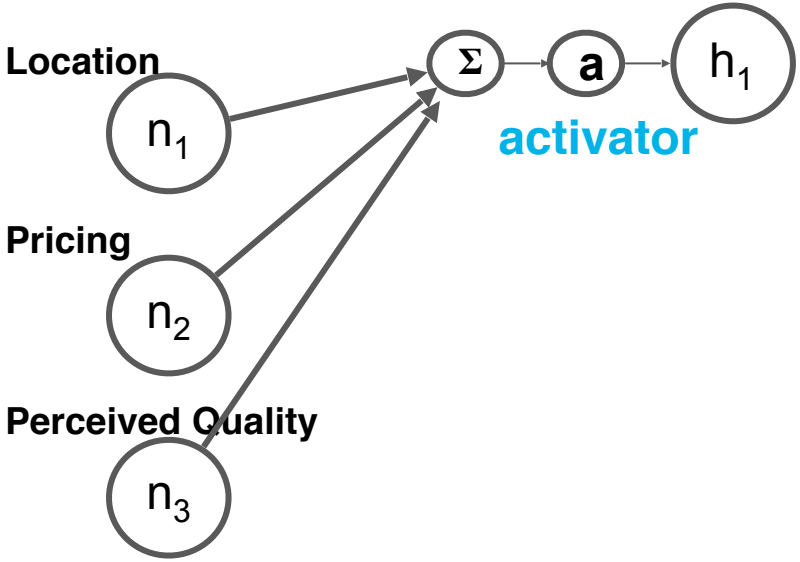
collector/
summation
function



Each node that receives input from previous nodes actually has **three** parts

input layer
nodes: 3

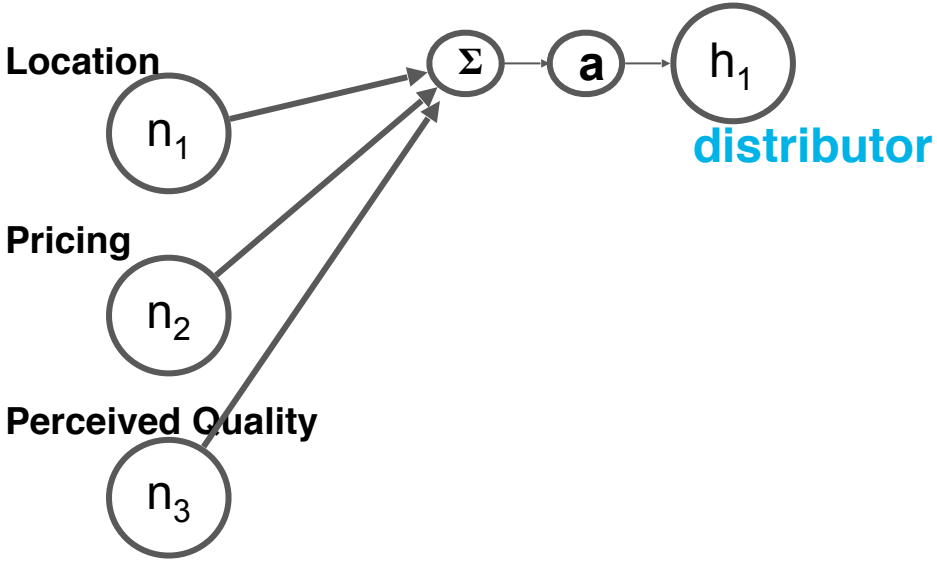
hidden layer



Each node that receives input from previous nodes actually has **three** parts

input layer
nodes: 3

hidden layer

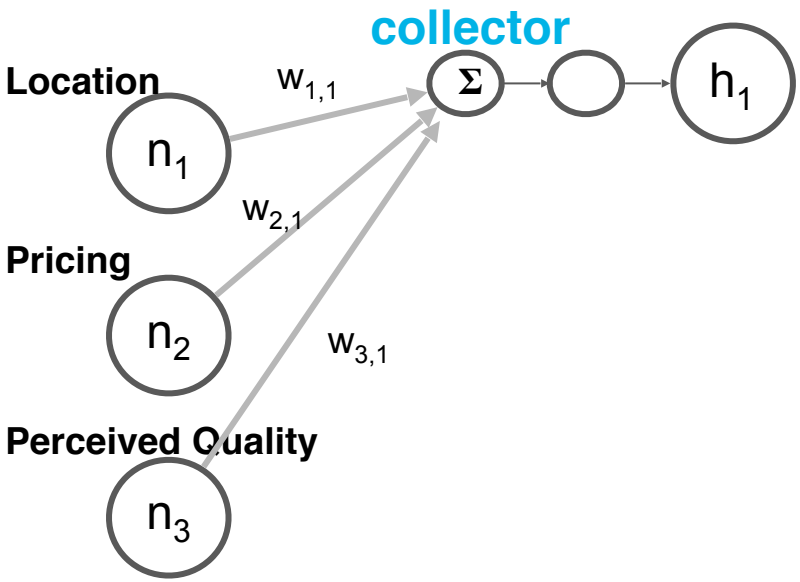


distributor

Each node that receives input from previous nodes actually has **three** parts

input layer
nodes: 3

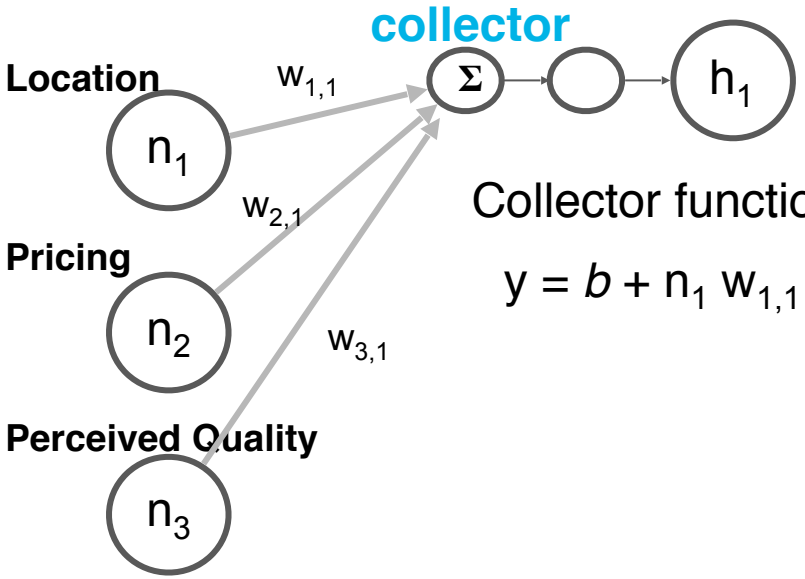
hidden layer



input layer
nodes: 3

hidden layer

This function should look familiar

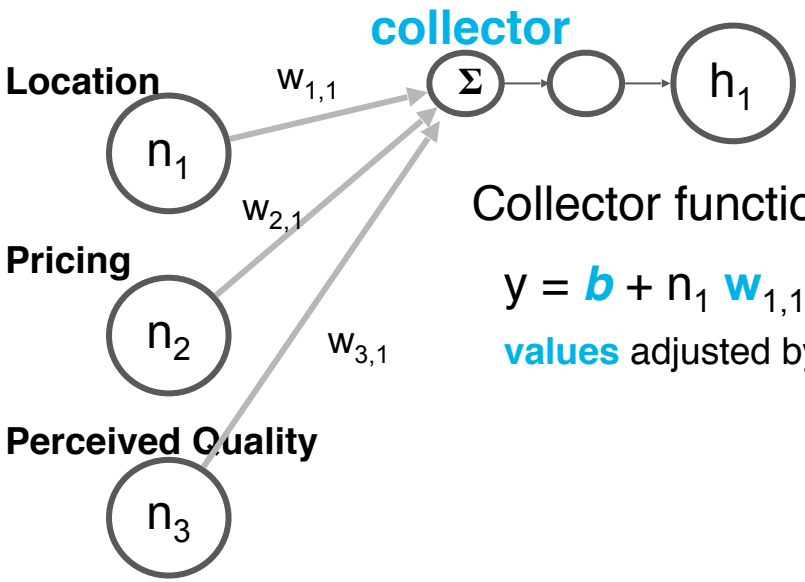


Collector function:

$$y = b + n_1 w_{1,1} + n_2 w_{2,1} + n_3 w_{3,1}$$

input layer
nodes: 3

hidden layer



Collector function:

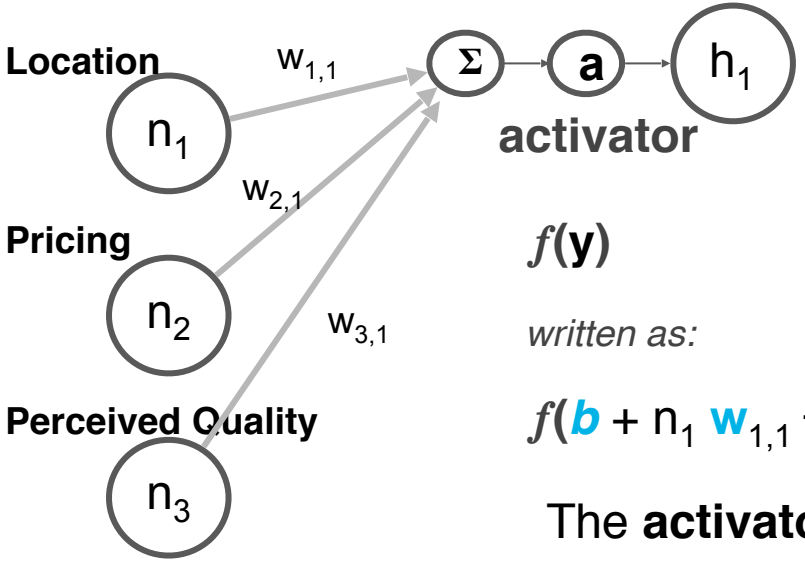
$$y = b + n_1 w_{1,1} + n_2 w_{2,1} + n_3 w_{3,1}$$

values adjusted by the algorithm

input layer

nodes: 3

hidden layer



The **activator** is a function chosen by **you** that takes the output of the collector as input.

$f(\mathbf{y})$

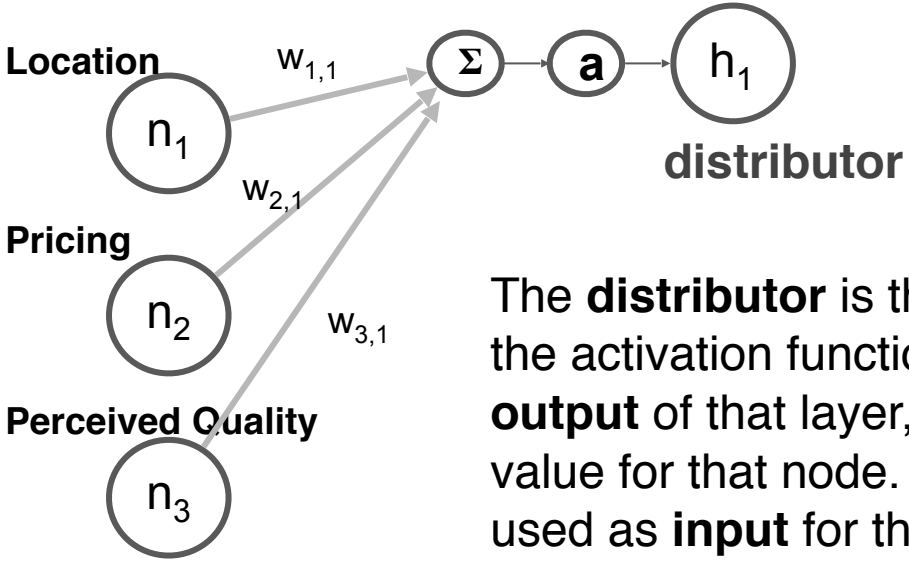
written as:

$$f(b + n_1\, w_{1,1} + n_2\, w_{2,1} + n_3\, w_{3,1})$$

The **activator** is specified for **each** layer. Nodes within a layer all use the same activation function

input layer
nodes: 3

hidden layer

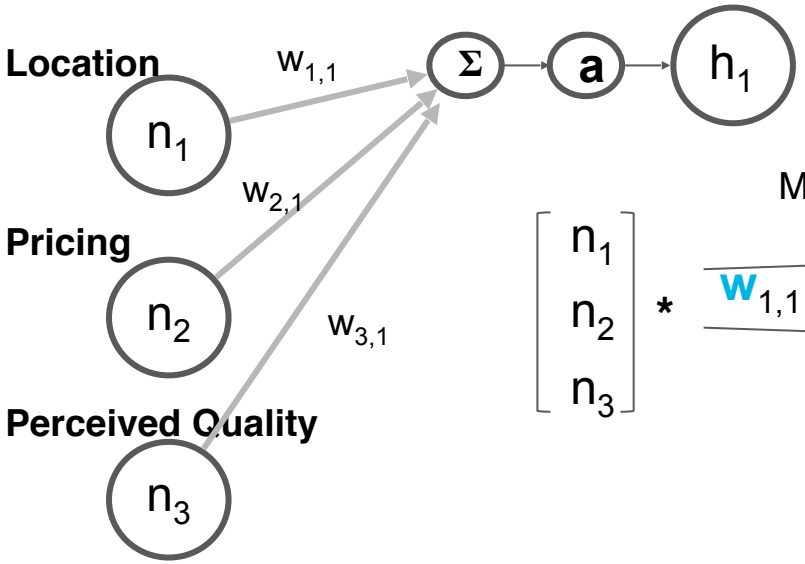


The **distributor** is the **output** of the activation function. It is the **output** of that layer, the final value for that node. It is then used as **input** for the next layer.

input layer
nodes: 3

hidden layer

For the math lovers in the room:



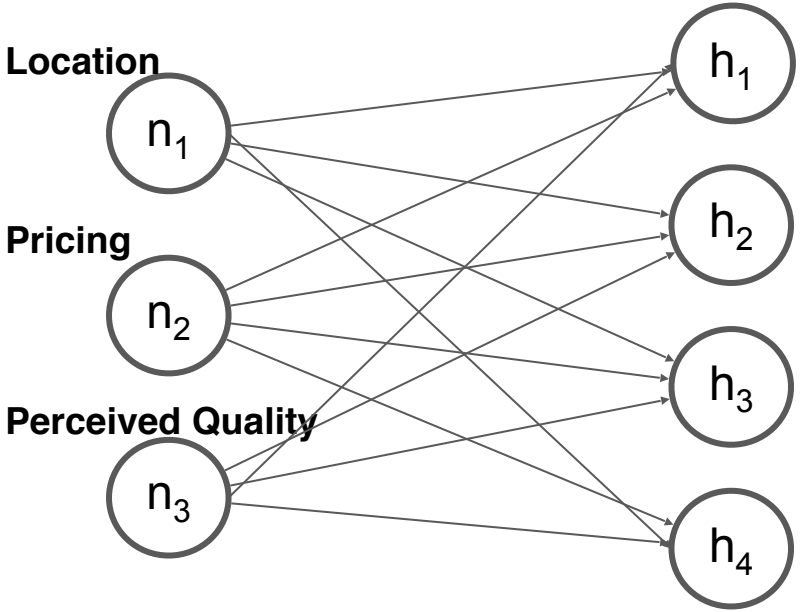
Matrix notation for one hidden node

$$\begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} * \overline{w_{1,1}\, w_{2,1}\, w_{3,1}} + b_1 = z_1 \rightarrow f_{a1}(y_1) \rightarrow h_1$$

input layer
nodes: 3

hidden layer

The full math for a layer:



input layer
nodes: 3

hidden layer

Matrix notation for one hidden layer

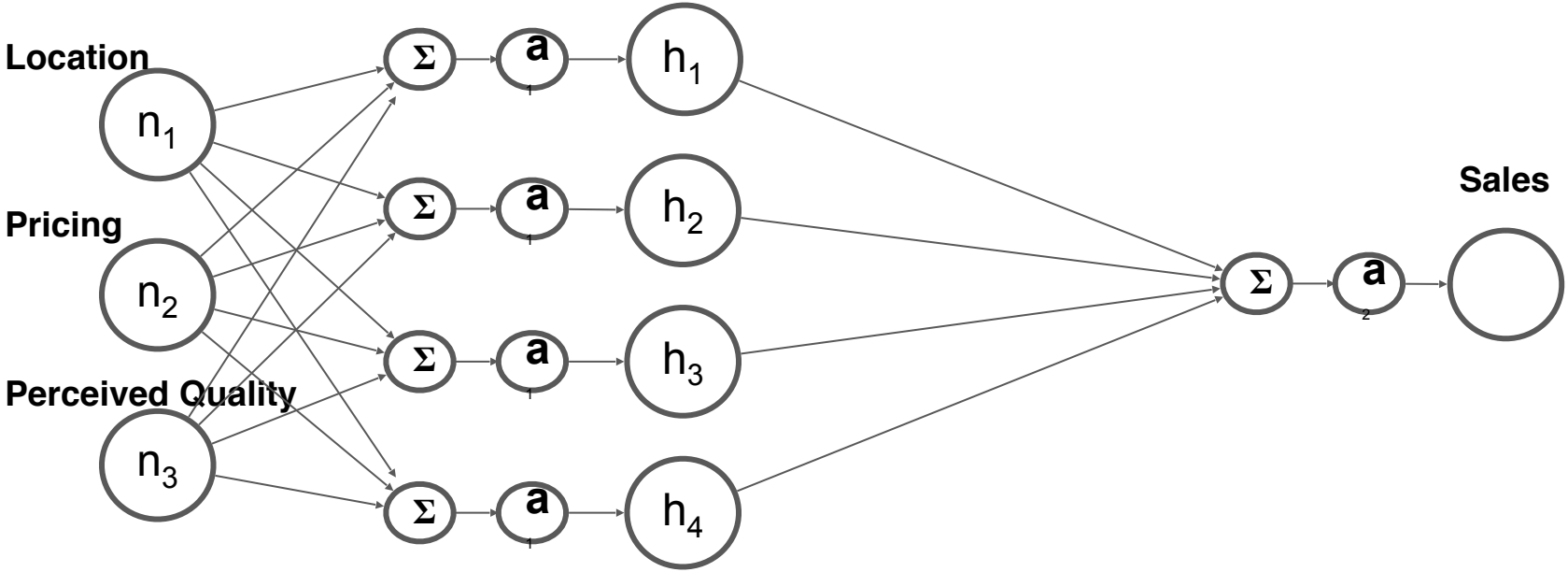
$$\begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} * \begin{bmatrix} w_{1,1} & w_{2,1} & w_{3,1} \\ w_{1,2} & w_{2,2} & w_{3,2} \\ w_{1,3} & w_{2,3} & w_{3,3} \\ w_{1,4} & w_{2,4} & w_{3,4} \end{bmatrix}^T + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

$$\begin{bmatrix} f(y_1) \\ f(y_2) \\ f(y_3) \\ f(y_4) \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix}$$

When creating a layer, what are the specs that **you** choose?

Summary

Hidden layer variables	**You define**	**Computer figures out**
weights		✔
activation function	✔	
bias		✔
number of nodes	✔	



input layer
nodes: 3

hidden layer

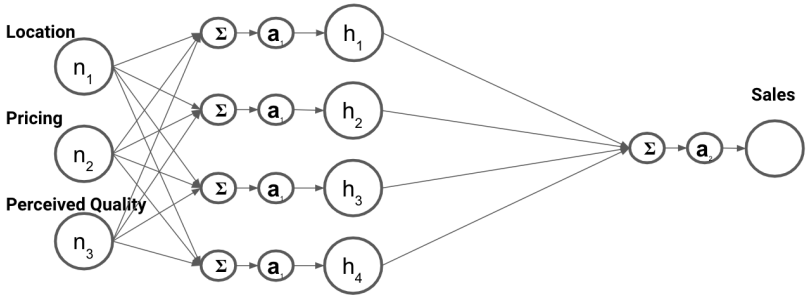
output layer
one node

Summary so far:

What you choose when building a neural network:

At the ***network*** level:
- Number of input variables
- Number of hidden layers
- If it is a classification or regression problem

At the ***layer*** level:
- The number of nodes
- The activation function



What else can we adjust?

What you choose when building a neural network:

At the ***network*** level:

- Number of input variables
- Number of hidden layers
- If it is a classification or regression problem
- **Batch size**
- **Number of epochs**
- **Learning rate & optimizer**
- **Regularization type and lambda**

At the ***layer*** level:

- The number of nodes
- The activation function

Batches and **Epochs** are about data processing

That's a lot of math and a lot of data. The dataset is split into chunks and passed through the network one chunk at a time.

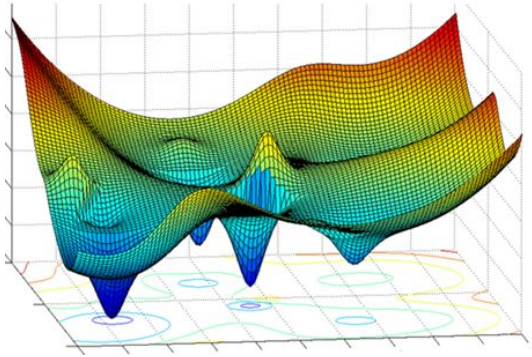
Batch defines the number of observations in each "chunk".

Epochs are how many times you want the whole dataset to go through the network.

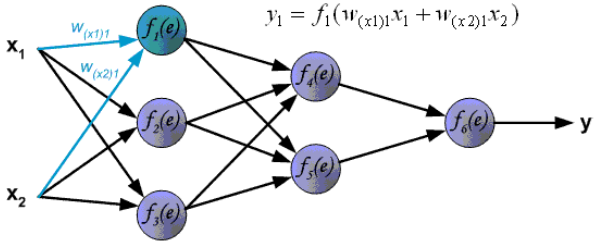
All the **batches** = one **epoch**.

Learning rate is from gradient descent

Scenario	**Cost/Loss Function**
Regression	MSE
Binary classification	Cross-Entropy (Logarithmic loss)
Multi-class classification	Softmax of Cross-Entropy

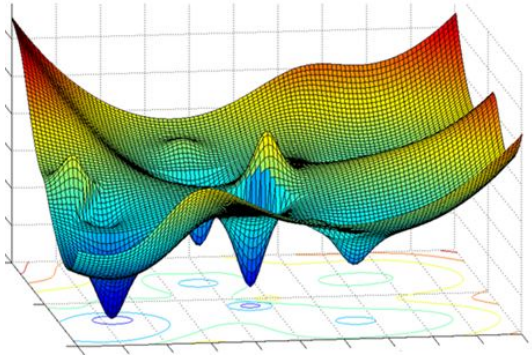


FP

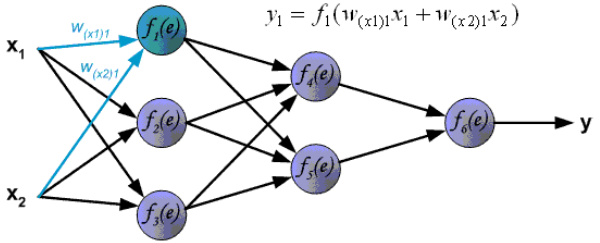


Optimizer is how the gradient is calculated

Options
sgd
rmsprop
Adagrad
Adadelta
Adam
And more!



FP



Regularization - adjusts the weights by layer

Adding a set penalization term at each collector node.

You can choose no regularization, lasso, or ridge.

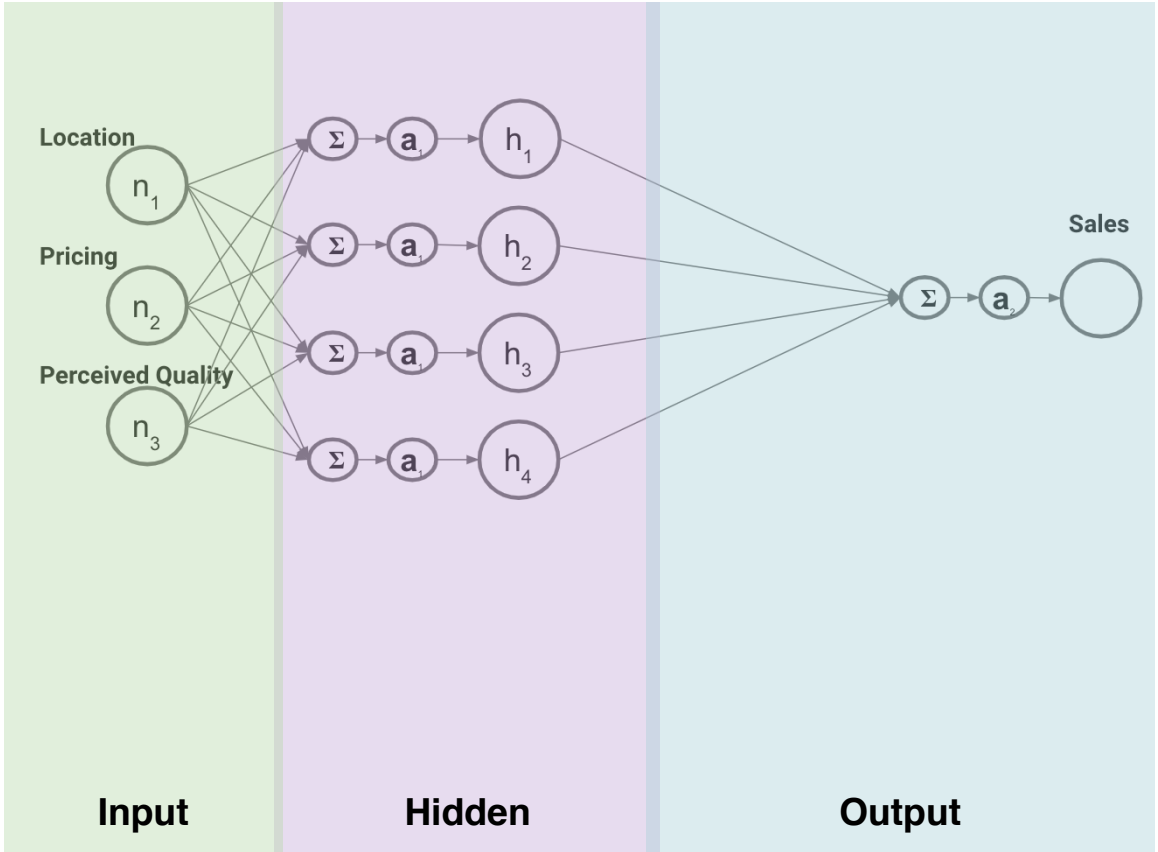
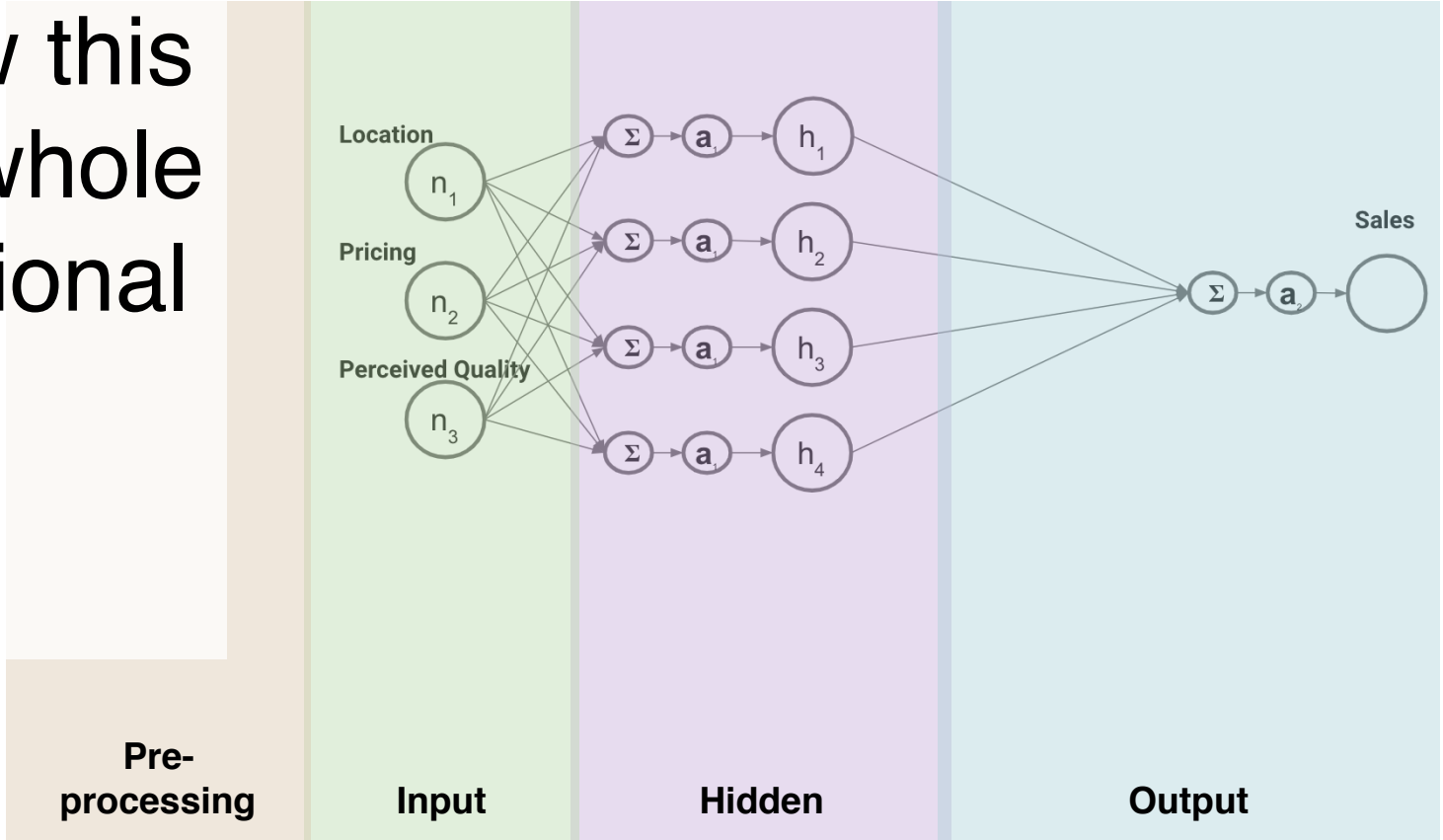
Write in sentences what this code does

```
model = Sequential()
model.add(Dense(32, activation='relu', input_dim=100))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Generate dummy data
import numpy as np
data = np.random.random((1000, 100))
labels = np.random.randint(2, size=(1000, 1))

# Train the model, iterating on the data in batches of 32 samples
model.fit(data, labels, epochs=10, batch_size=32)
```

Even without learning Keras explicitly, you should be able to recognize keywords and concepts based on this review.

Dense = fully connected to all previous nodes

While we have covered this:



Know this is a whole additional area.