AGGIE DATA SCIENCE CLUB

Spring 2024
Neural Networks

www.aggiedatascience.com

Aggie Data Science Club

aggiedatascience

Intro to Neural Networks

Word of Warning!

- This is **very dense** material presented **very quickly**
- **<u>Don’t feel bad if you don’t get it</u>**
- Please ask if you want a topic clarified
- Trying to fit multiple weeks of a grad class into an hour…

Overall Steps in a Neural Network Model:

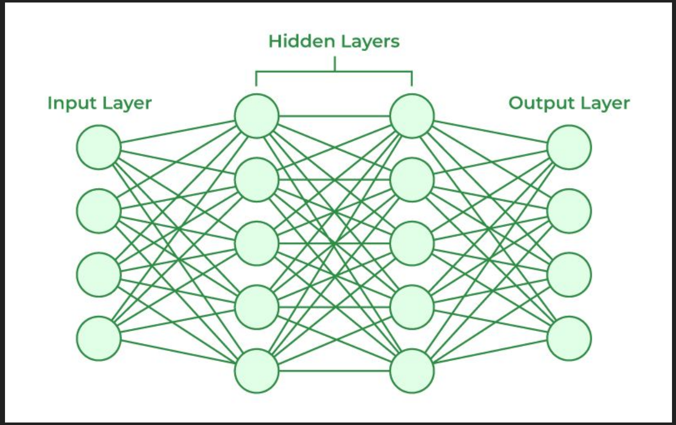
-Feed data

-Making Predictions and Forward Propagation: Computations are performed on each layer of the neural network to make predictions. This process is often called a “black box” because the process in finding these outputs is complex and not completely known. Activation functions like ReLU (Rectified Linear Unit) or sigmoid are used to understand more complex relationships in the data. The output of each layer, after applying the activation function, is then passed to the next layer.

-Back Propagation : Calculates gradients using chain rule, contains the partial derivatives of the loss function with respect to each weight and bias ($\partial L/\partial w$ and $\partial L/\partial b$). These partial derivatives indicate how much a small change in each parameter will affect the loss.

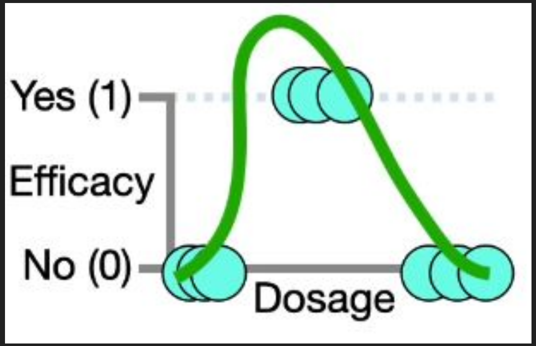
-Gradient Descent : Once you have these calculated gradients from back propagation, we use gradient descent in order to adjust the weight and bias values and minimize loss

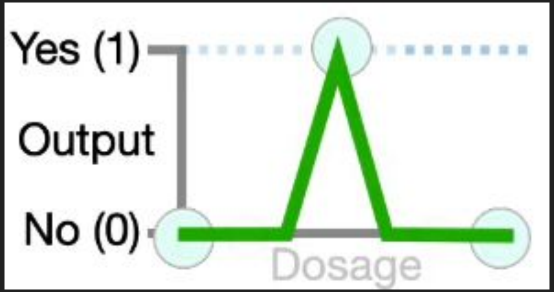
-Repeat



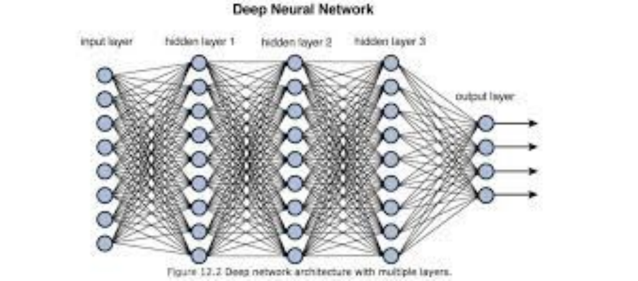
Why are NNs good?

- **Approximate ANY curve**
  - Logistic regression struggles with clusters of data
  - SVM only works with linearly separable data unless you use a kernel, which is not systematic
- **Learn hidden features**
  - Find underlying patterns in the data that are too complex for other models
- ***Very good* results**
  - Because of the ability to fit very well

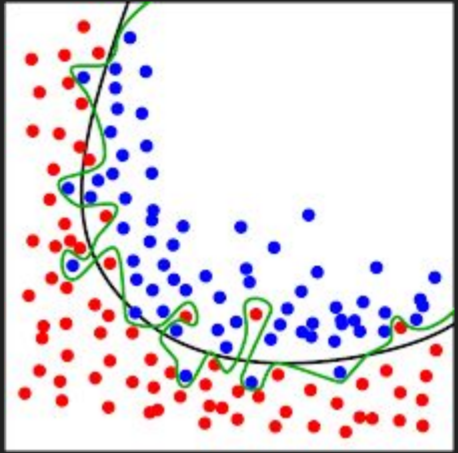




Risks of NNs



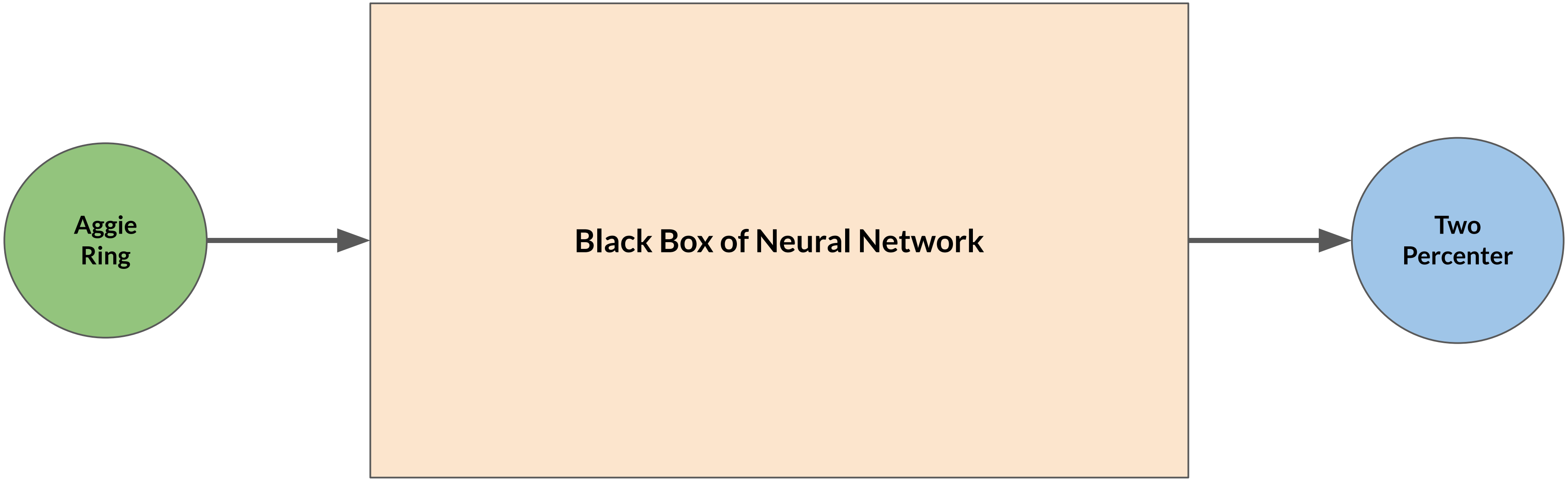
- **Need a LOT of data**
- **Lots of time/money/computing power to train**
  - Think about the GPU costs for something like ChatGPT → $700,000 daily!
  - More parameters = more computations = more time
- **Hyperparameter tuning is not easy**
  - Lots of granularity means many different options
- **Uninterpretable**
  - Bias and fairness implications in hidden features
  - Hard to explain the results to someone...
- **Prone to overfitting**
  - You can continue training all the way to the exact dataset
  - The squiggle can get very, very complicated...



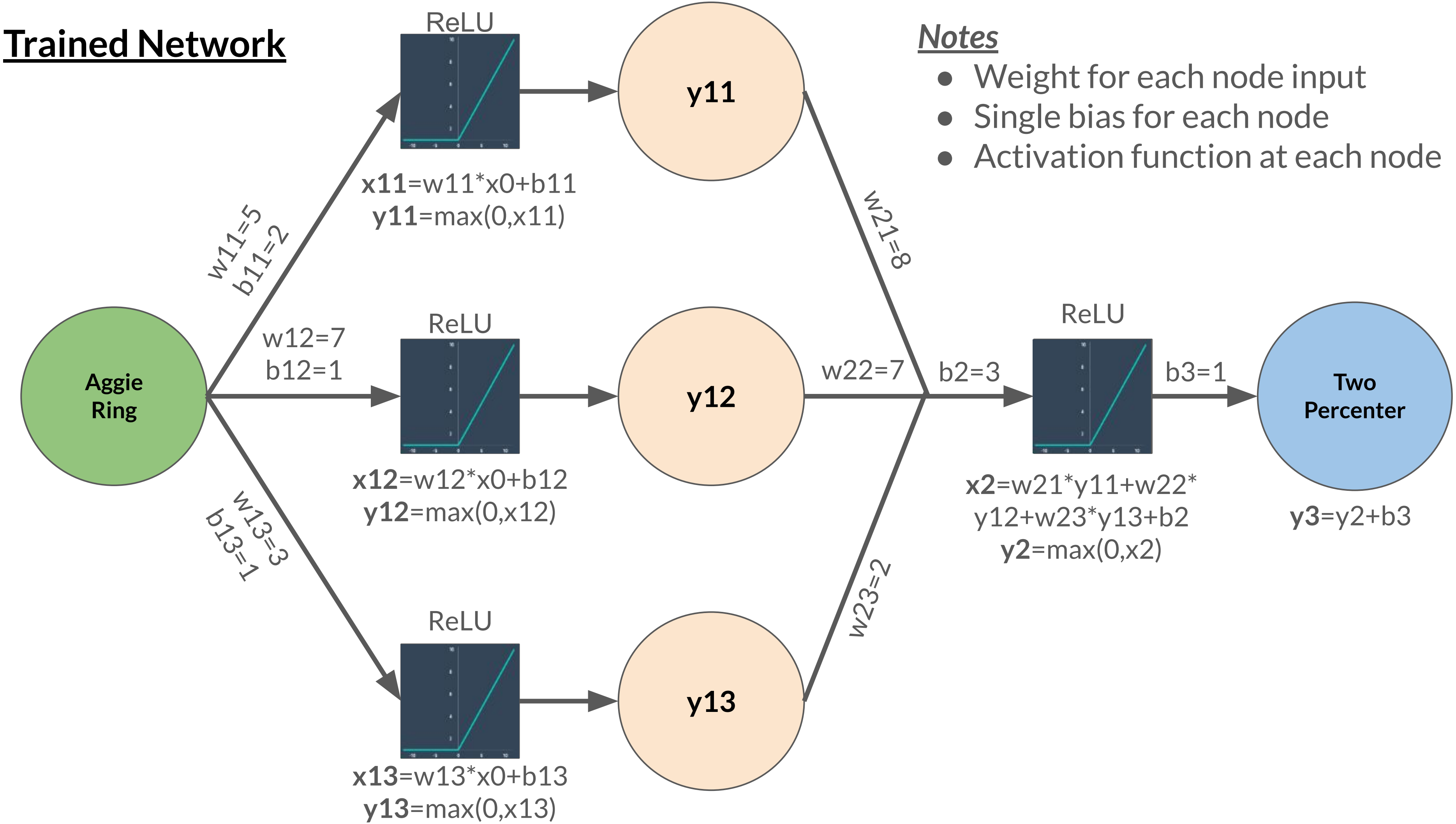
Code Demo!

How do NNs work?

Goal



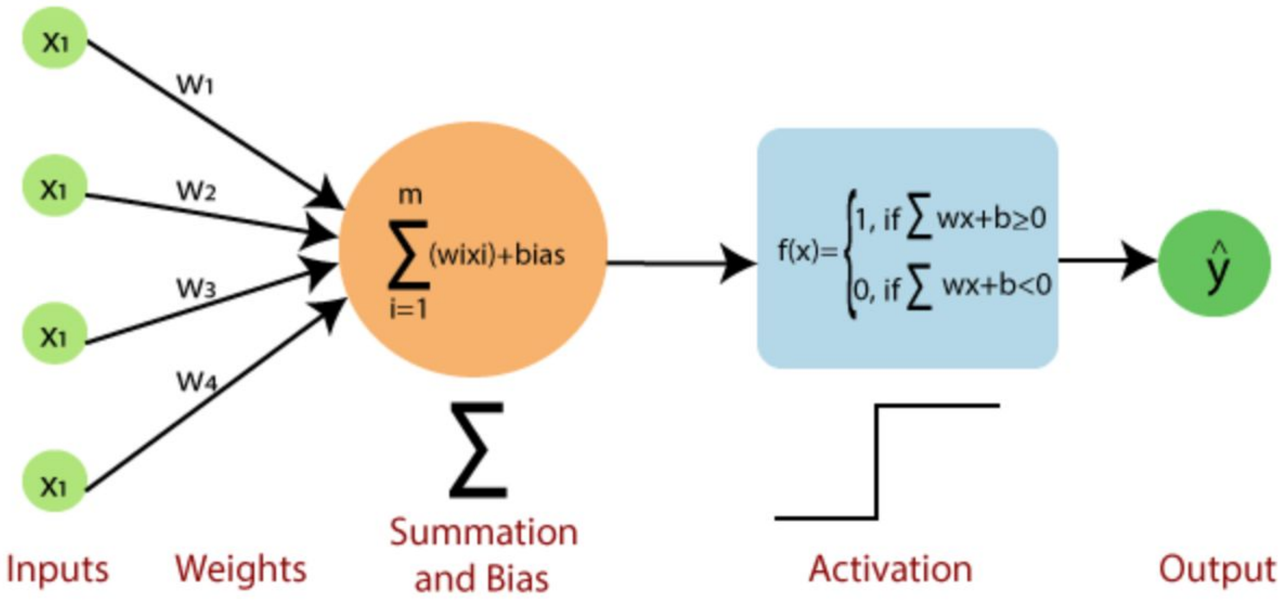
Trained Network



Notes

- Weight for each node input
- Single bias for each node
- Activation function at each node

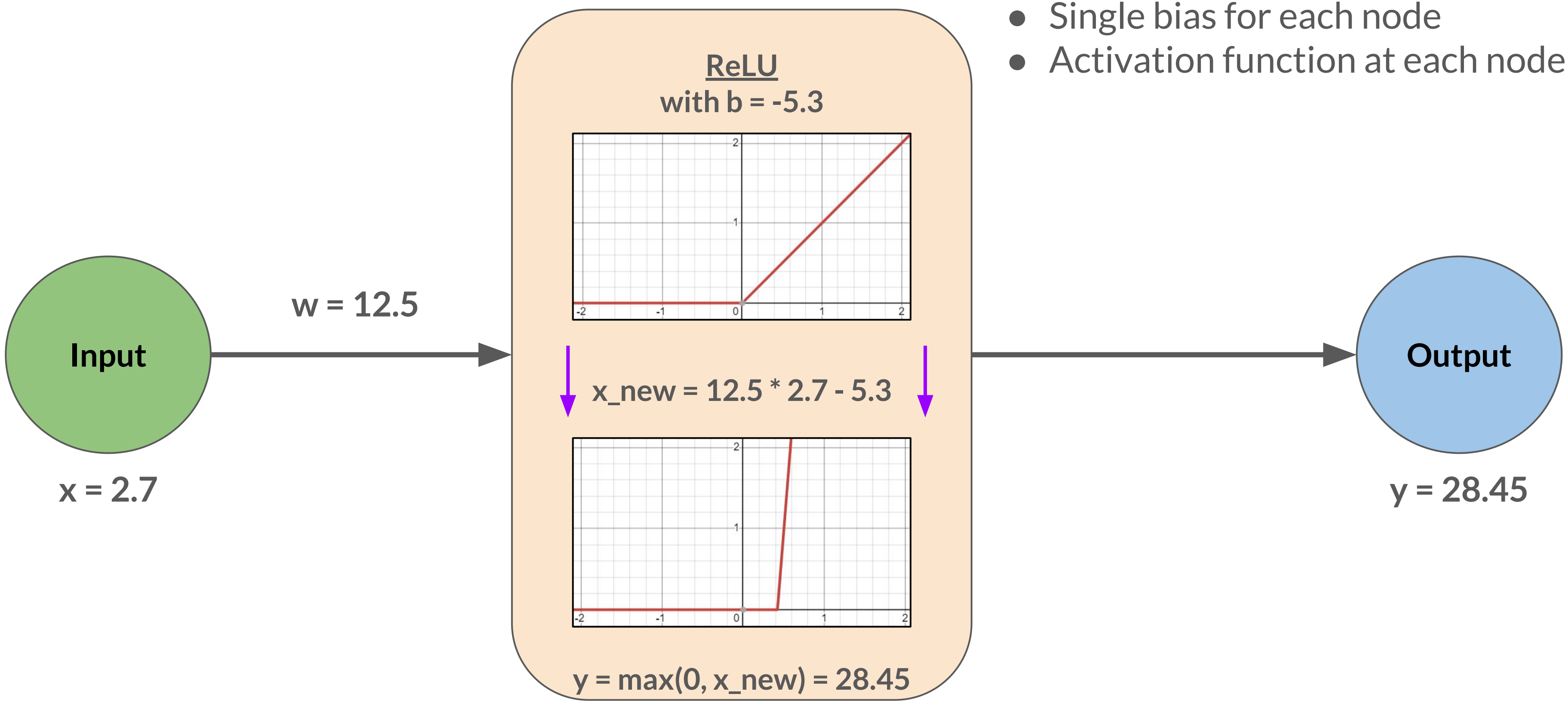
General Form of Perceptron



Single-Layer Perceptron (Single Input)

Notes

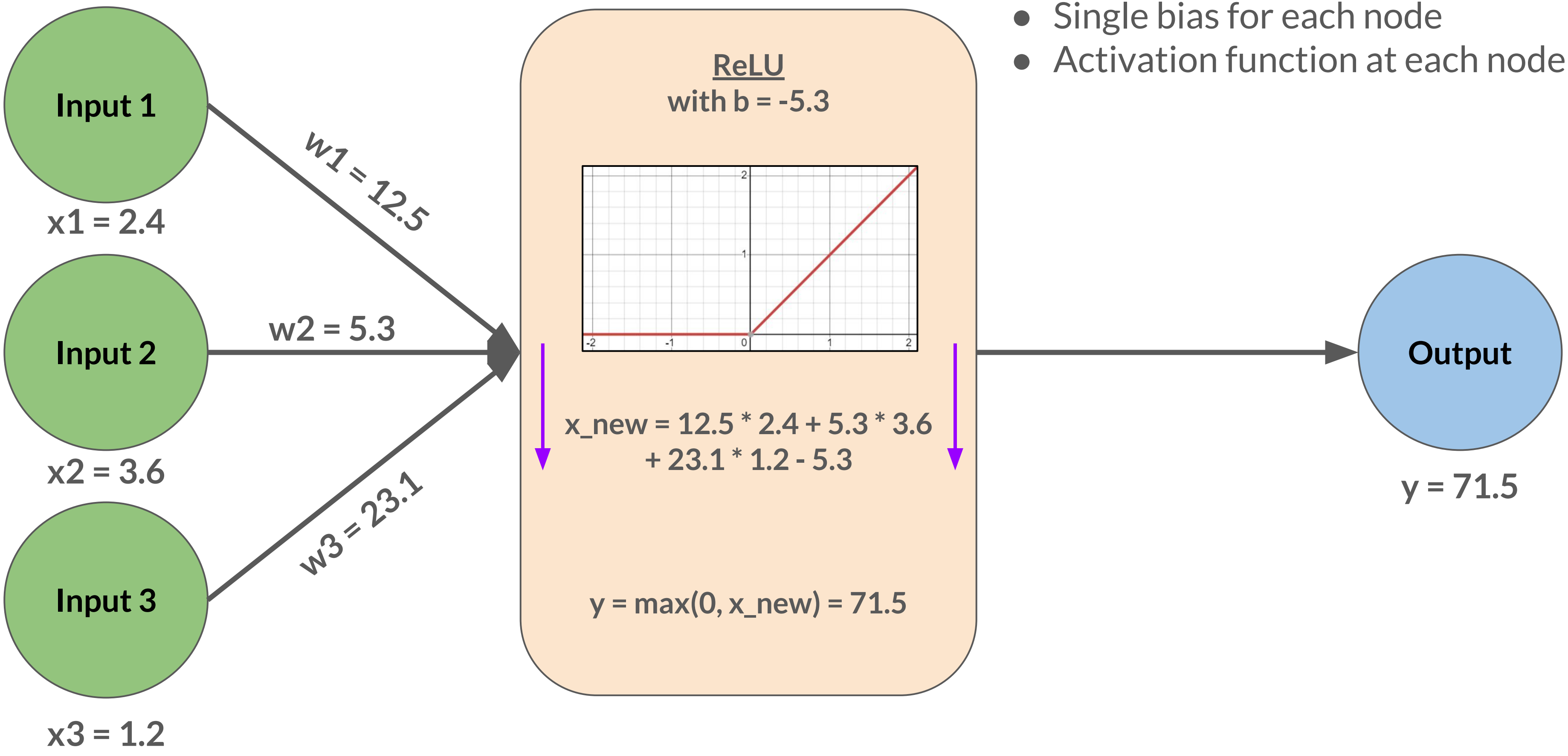
- Weight for each node input
- Single bias for each node
- Activation function at each node



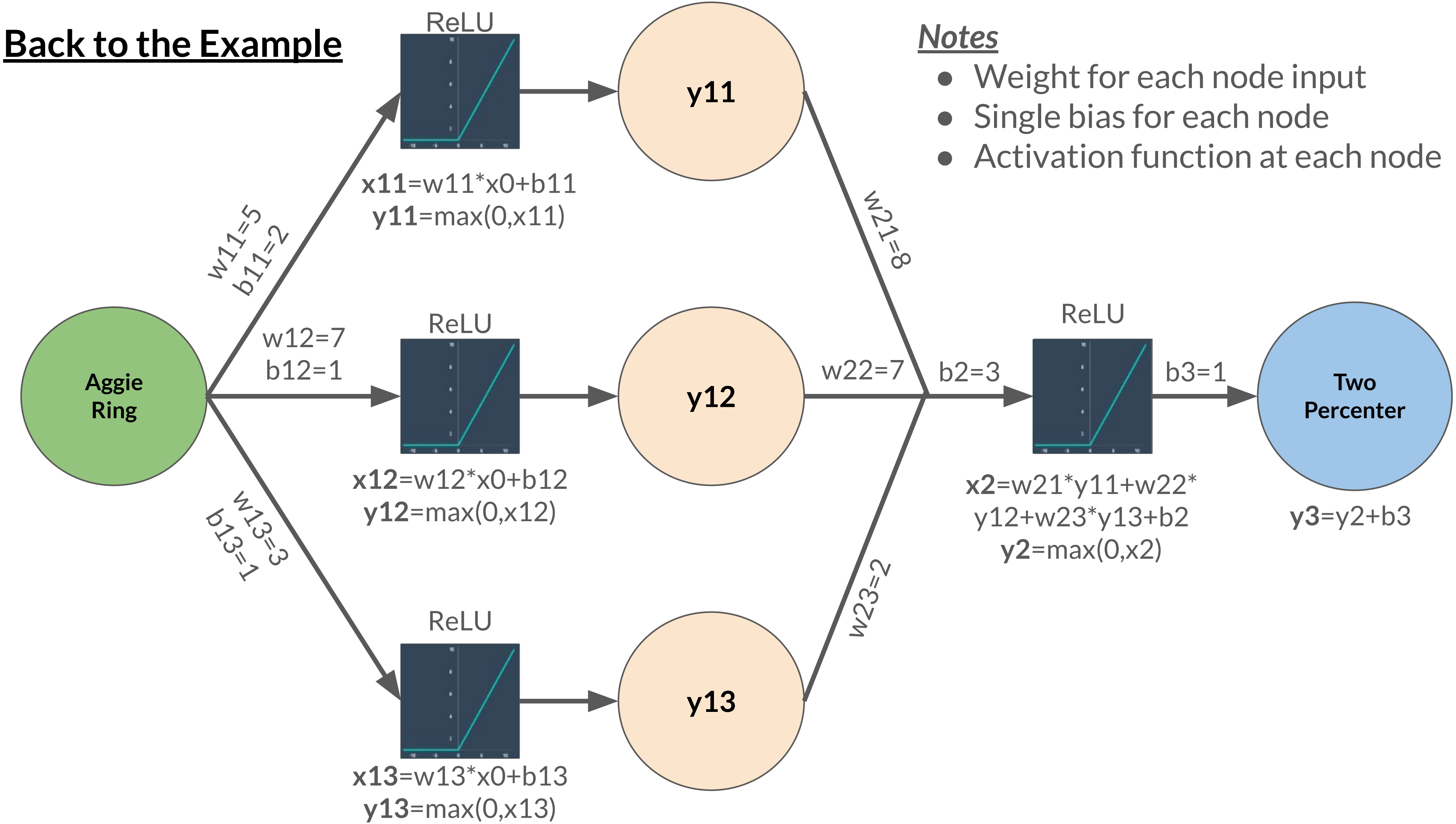
Single-Layer Perceptron (Multi Input)

Notes

- Weight for each node input
- Single bias for each node
- Activation function at each node



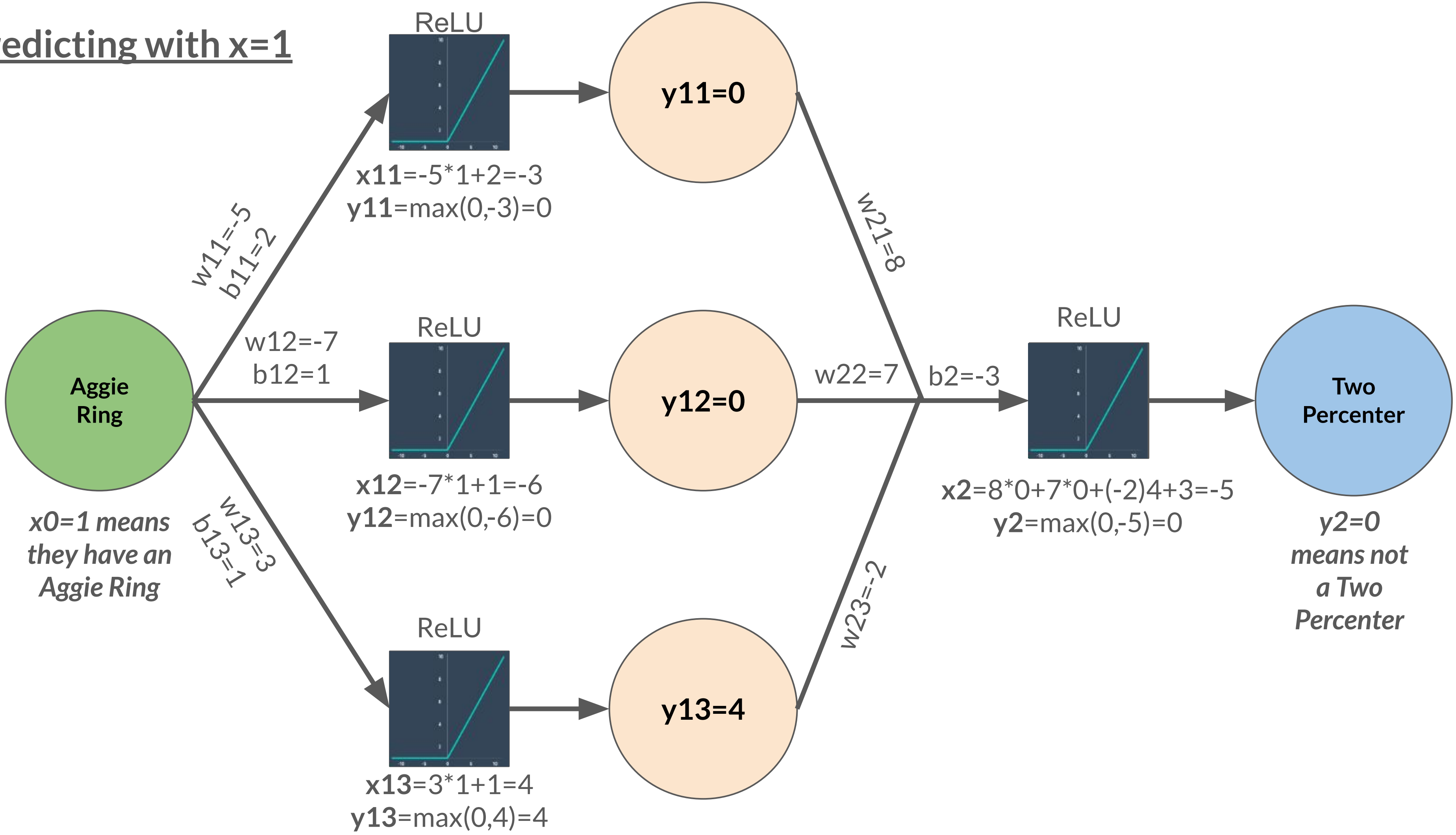
Back to the Example



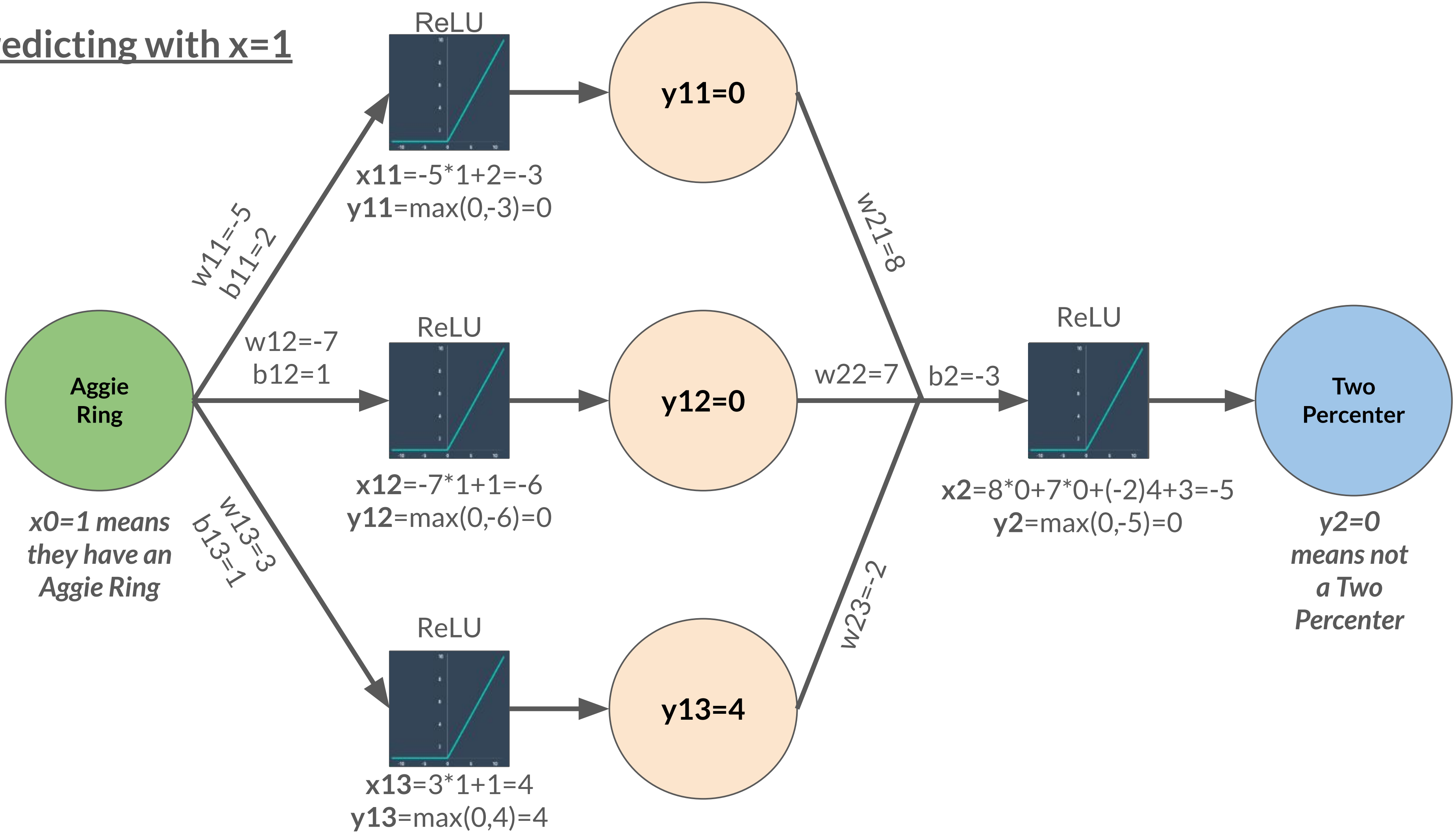
Notes

- Weight for each node input
- Single bias for each node
- Activation function at each node

Predicting with x=1

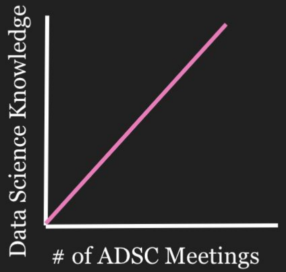


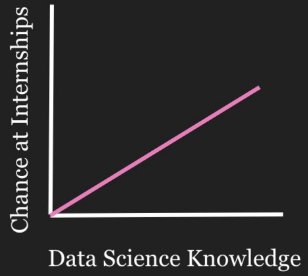
Predicting with x=1



Chain Rule and Gradient Descent

Refresher on Chain Rule





-used for backpropagation in neural networks

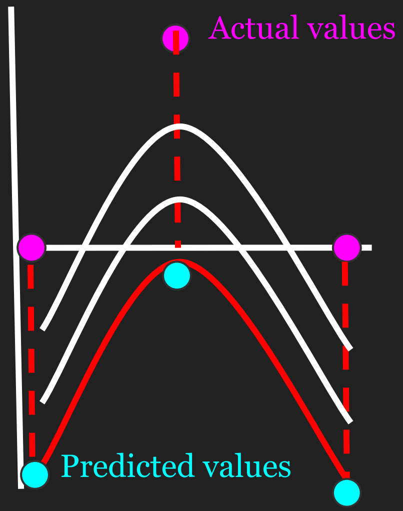
Derivative of f(g(h)): (f'g(h)) * (g'(h))

Ex: Derivative of sin(2x) = 2cos(2x)

Let's compare the relationship between the number of ADSC meetings and the amount of data science knowledge you have. Then, we'll compare how the amount of data science knowledge you have relates to your chances of getting an internship. We want to know the relationship between the number of ADSC meetings you attend and the chance of getting internships. (Notice it's all positively correlated :))

$$\frac{d\text{ knowledge}}{d\ \#\text{ of meetings}} = 2 \qquad \frac{d\text{ internships}}{d\text{ knowledge}} = \frac{1}{2} \qquad \frac{d\text{ internships}}{d\ \#\text{ of meetings}} = \frac{d\text{ knowledge}}{d\ \#\text{of meetings}} * \frac{d\text{ internships}}{d\text{ knowledge}} = 2 * \frac{1}{2} = 1$$

Gradient Descent



Gradient Descent is a more effective way of minimizing loss functions to optimize values in the neural network such as weights and biases. The ultimate goal is to adjust the parameters so that the difference between target values and predicted values is smaller and therefore makes the model more accurate.

It works for any loss function you, for example, MSE or SSR.

Residual = Observed - Predicted Values

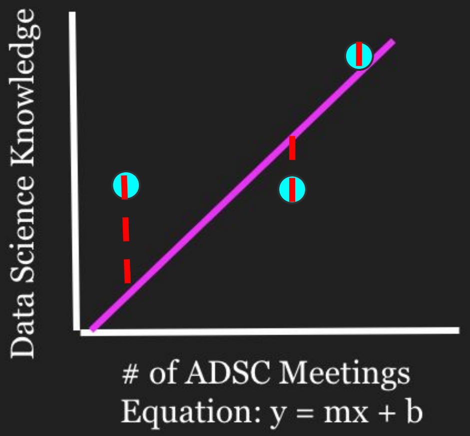
0 - -2.5 =2.5   1 - -1.2 = 2.2   0 - -2.51 = 2.51

Example of a Common Loss Function:

SSR = ∑(Observed - Predicted) ^2

SSR = (2.5)^2 + (2.2)^2 + (2.51)^2

-In neural networks, we adjust the weights and biases in order to find where the minimum loss occurs. If we look at a linear regression model, we would want to adjust the slope and/or the intercept (bias) in the equation y = mx + b.



If we're looking at adjusting the intercept alone:

Loss Function: SSR = ∑(Observed - Predicted) ^2

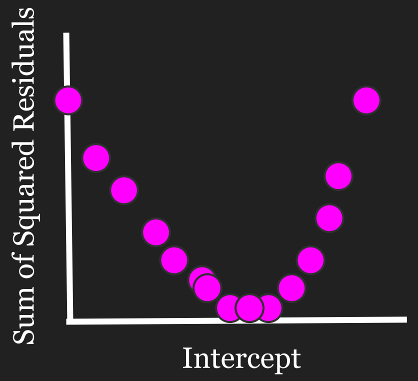
How it changes in respect to the intercept:

∂SSR / ∂b = (2) * ∑(Observed - Predicted) * (-1)

∂SSR / ∂b = (2) * ∑(Observed - (mx+b)) * (-1)

-Plug in different values for the intercept (b) into the equation above and plot the values. We do this to see how the loss function is changing over time and see where the loss is the lowest.

Note: We would need to plug in many values which takes time so we use gradient descent to this more efficiently.



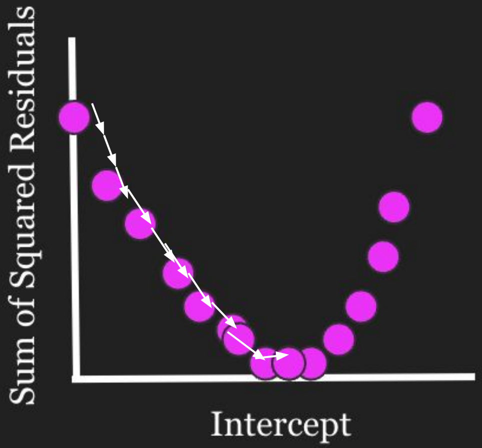
α = Learning Rate

α * (∂SSR / ∂b) = Step Size

New intercept = Old intercept - ***α*** * (∂SSR / ∂b)

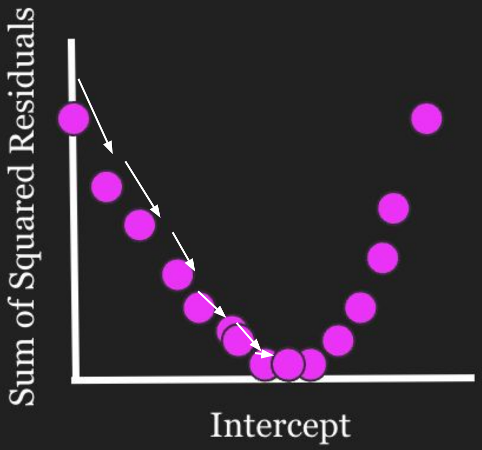
****The goal is to find where the loss is at a minimum****

Slow Learning Rate and Small Step Sizes



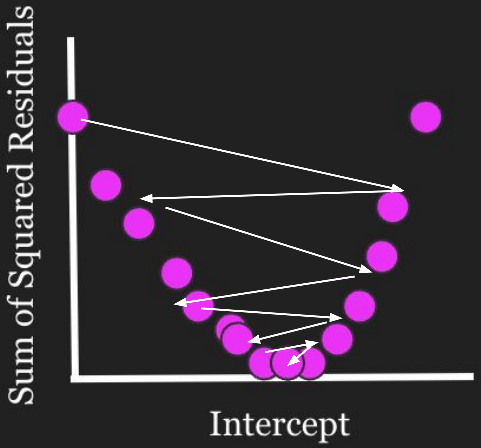
- ➔ Slow

Gradient Descent



- ➔ Efficient
- ➔ Most Accurate

Fast Learning Rate and Large Step Sizes

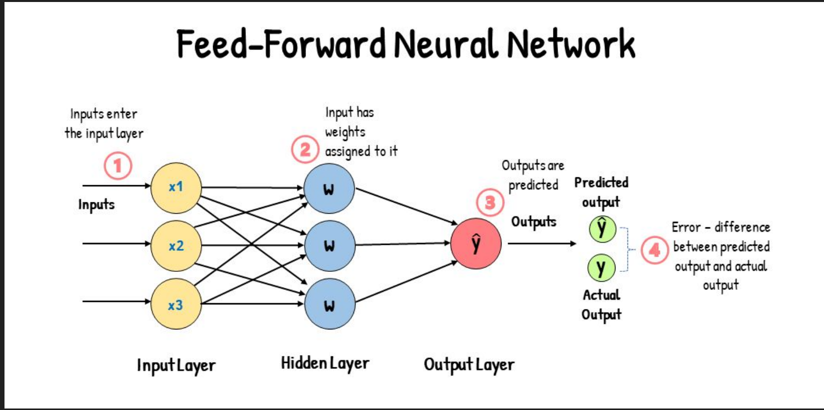


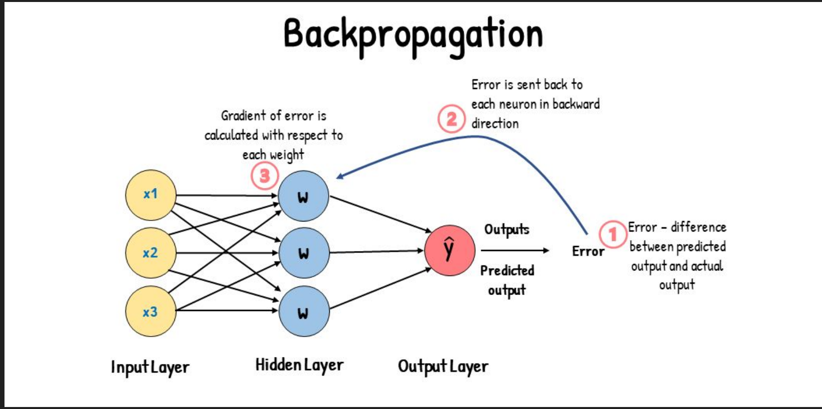
- ➔ Could be fast or slow

Backpropagation

Backpropagation

- "Backward propagation of errors"
- **<u>Iteratively updates weights across all layers</u>** using gradient descent (and chain rule)
- With a given ***error function*** and ***initial values of weights/biases***, the method calculates the **gradient of the error function** with respect to all the neural network weights
- Consists of **<u>continuous forward prediction and backward error propagation to predict values and update weights</u>**
- Important: partial computations of the gradient from one layer are ***reused*** in the computation of the gradient for the previous layer
  - ***Efficient*** computation that ensures ***continuous ongoing*** updates in the right direction

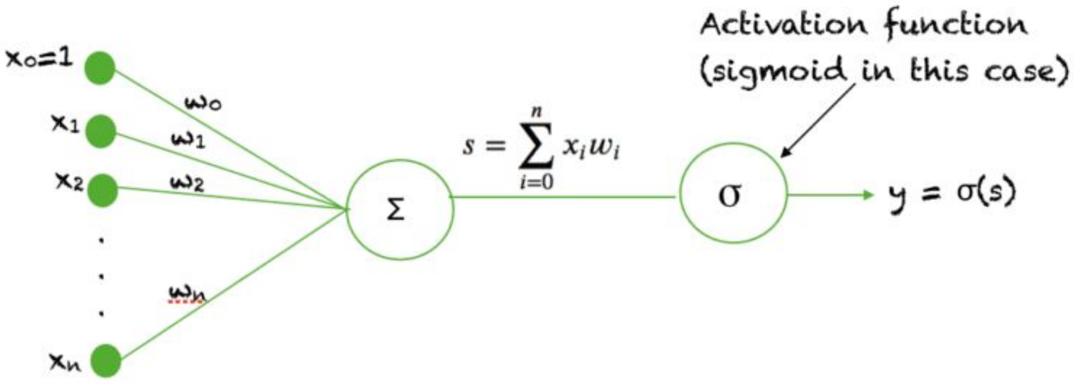




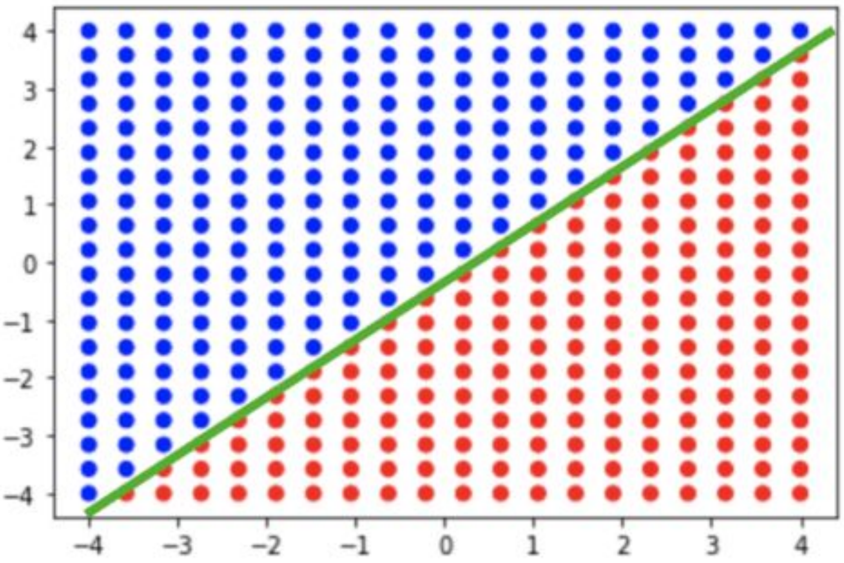
Activation Functions

How activation functions are used

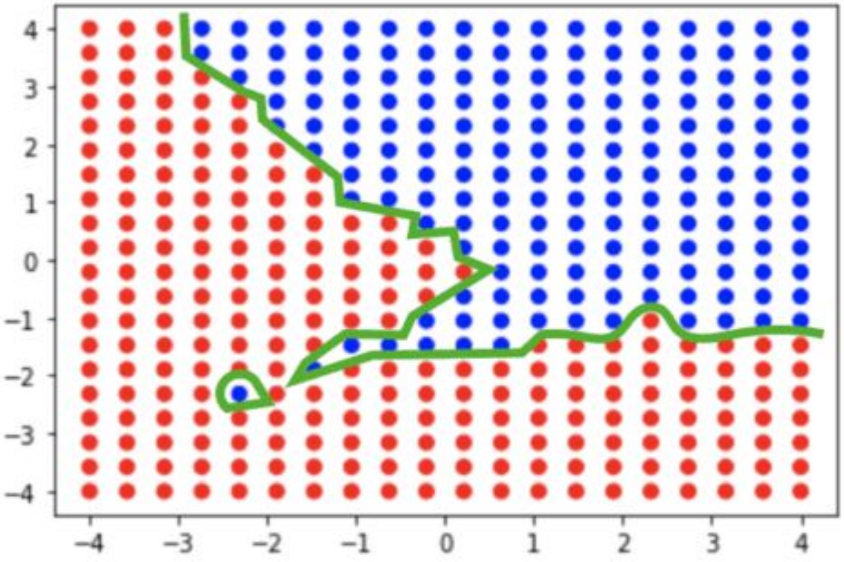
1. Computer wants to make choices
2. It has many options to choose from
3. Activation functions give the computer an idea of which options are the "strongest"
   a. Activation levels



Linearity vs Non-linearity



Linear boundary



Non-linear boundary

Sigmoid f(x) = 1 / (1+e^(-x))

What is it?

- Applies nonlinearity to functions allowing model to learn complex relationships

Use cases

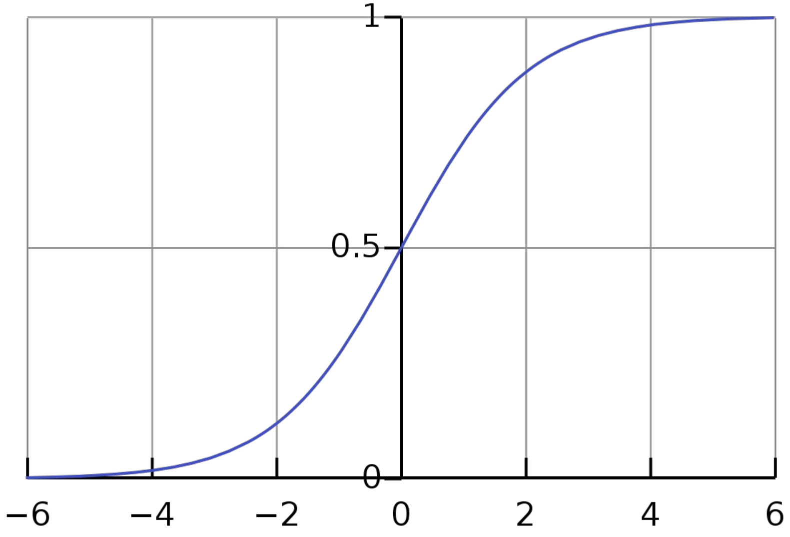
- Binary classification
    - Final output layer
- Probabilities

Vanishing Gradients

Sigmoid / Logistic

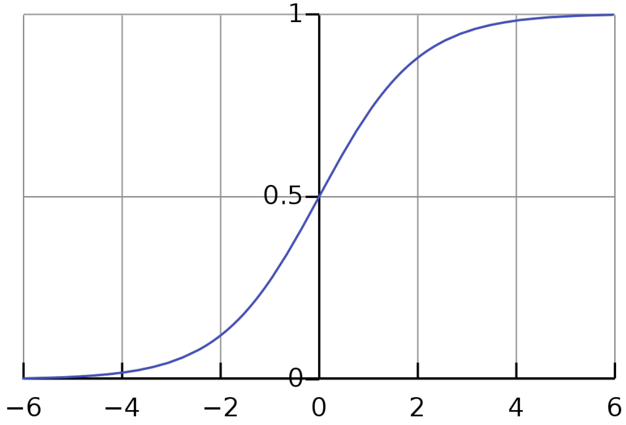
$$f(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid - Graphical Representation

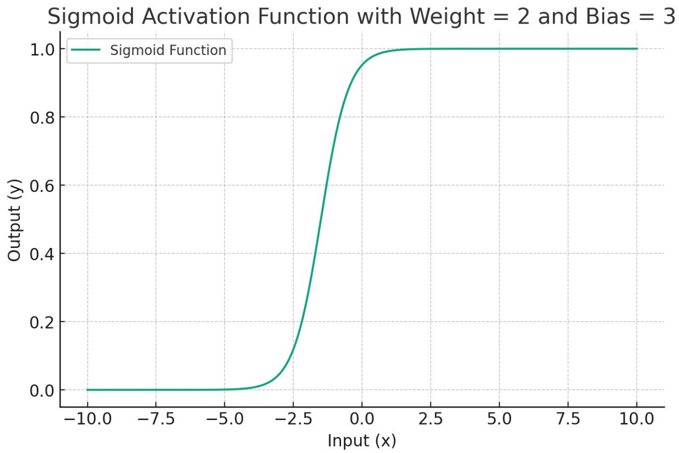


Input	Output
Most negative values	A value Close to 0
Most positive values	A value close to 1
Values close to 0	A value between 0 and 1

Old vs new based on weights



W = 2, b = 3



ReLU - f(x) = max(0,x)

What is it?

- ReLU stands for Rectified Linear Unit. ReLU activation function is one
  used activation functions in the deep learning models.

Advantages

- Easy to implement and very fast
- The calculation speed is very fast
- Does not “squash” weights

Disadvantages

- On negative inputs, it loses function
    - Gradients might be returned as zero
- Can only be used on hidden layers
- Dying ReLU

$$f(x) = max(0, x)$$

Where,

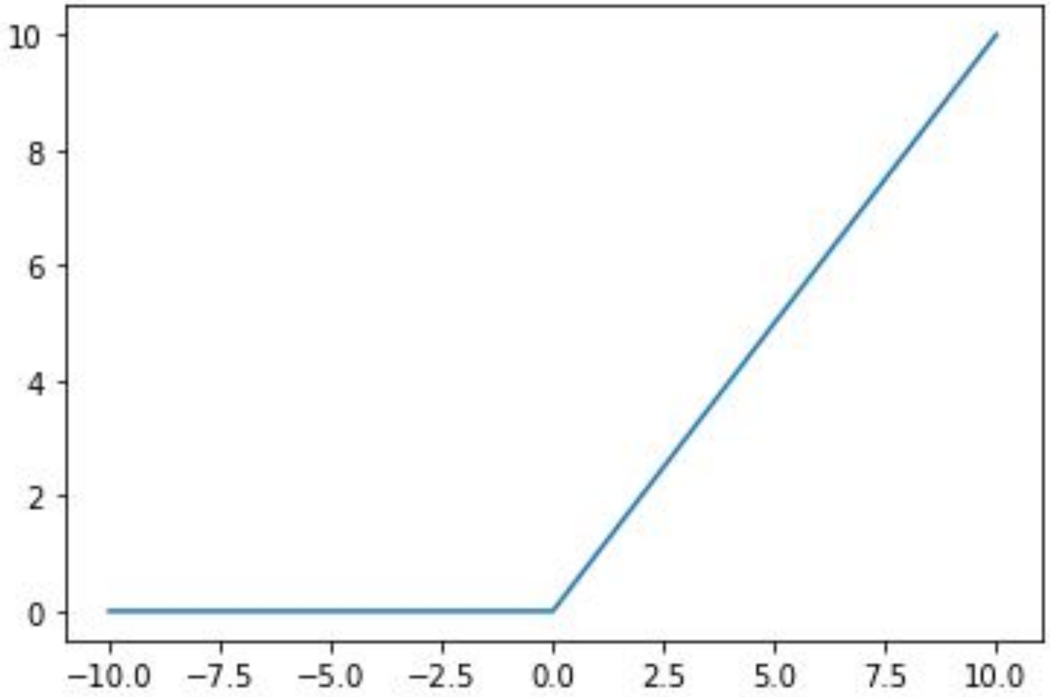
x – input to neuron

- This function also Ramp function

- Analogous to half – wave rectifier

```
if input > 0:
        return input
else:
        return 0
```

ReLU - Graphical Representation



Softmax

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for j = 1,...,K}$$

- Similar to sigmoid, except in the denominator we sum together all of the things in our raw output
- While sigmoid is useful for binary classification, softmax can be used for multi classification

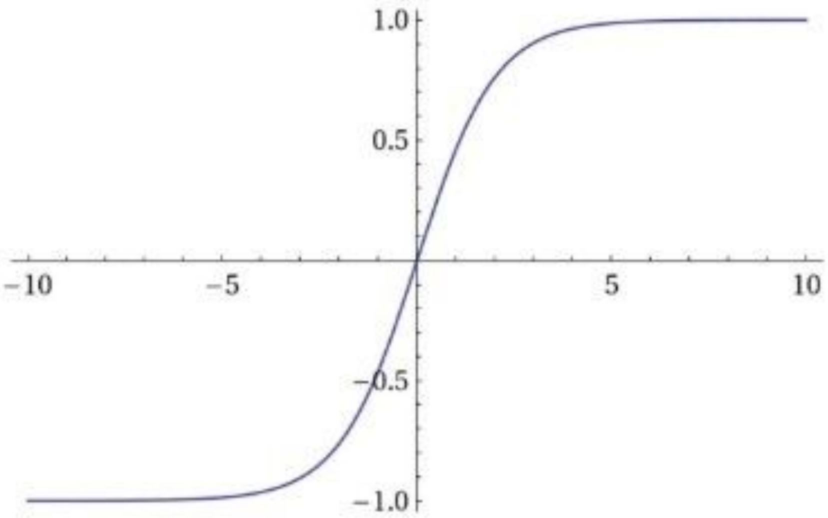
Raw output values	[3.2, -5.7, 0.6]
Applying **sigmoid** to raw output values	sigmoid calculation for the first raw output value: $\sigma(3.2) = \frac{e^{3.2}}{1+e^{3.2}} = 0.96$ result of sigmoid calculation for all three output values: [0.96, 0.0033, 0.65] Sum: 0.96 + 0.0033 + 0.65 = 1.61 ≠ 1
Applying **softmax** to raw output values	softmax calculation for the first raw output value: $\text{softmax}(3.2) = \frac{e^{3.2}}{e^{3.2}+e^{-5.7}+e^{0.6}} = 0.93$ result of softmax calculation for all three output values: [0.93, 0.00013, 0.069] Sum: 0.93 + 0.069 + 0.00013 = 1

Softmax

Softmax Function	**Sigmoid Function**
Used for multi-classification in logistic regression model.	Used for binary classification in logistic regression model.
The probabilities sum will be 1	The probabilities sum need not be 1
Used in the different layers of neural networks	Used as activation function while building neural networks
The high value will have the higher probability than other values .	The high value will have the high probability but not the higher probability.

Graphical Representation

Softmax Activation Function



Example :

$$2.33 \longrightarrow P\ (\text{Class 1}) = \frac{\exp\ (2.33)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.83827314$$

$$-1.46 \longrightarrow P\ (\text{Class 2}) = \frac{\exp\ (-1.46)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.01894129$$

$$0.56 \longrightarrow P\ (\text{Class 3}) = \frac{\exp\ (0.56)}{\exp(2.33) + \exp(-1.46) + \exp(0.56)} = 0.14278557$$

Activity: Build a Network for Two-Percenters

The Scenario: TAMU Merch Needs Help!

- TAMU Merch team needs help selling Travis Scott clothes
- They know that the *filthy* two-percenters won't buy their clothes
- **How do we target people that are *not* two-percenters?**
  - Neural networks!

Your Job: Design a Neural Net!

- **Given a set of demographic survey data, you need to create a model that can determine whether someone is a two-percenter based on certain attributes**
- Implement the input layer, hidden layer(s), and activation function(s)
- Set a learning rate to determine how your model performs gradient descent
- Select a loss function to evaluate your model
- *Note that it is up to **you** to select what input parameters are important, the number of hidden layers, and the number of nodes in each layer. These will differ amongst groups.*

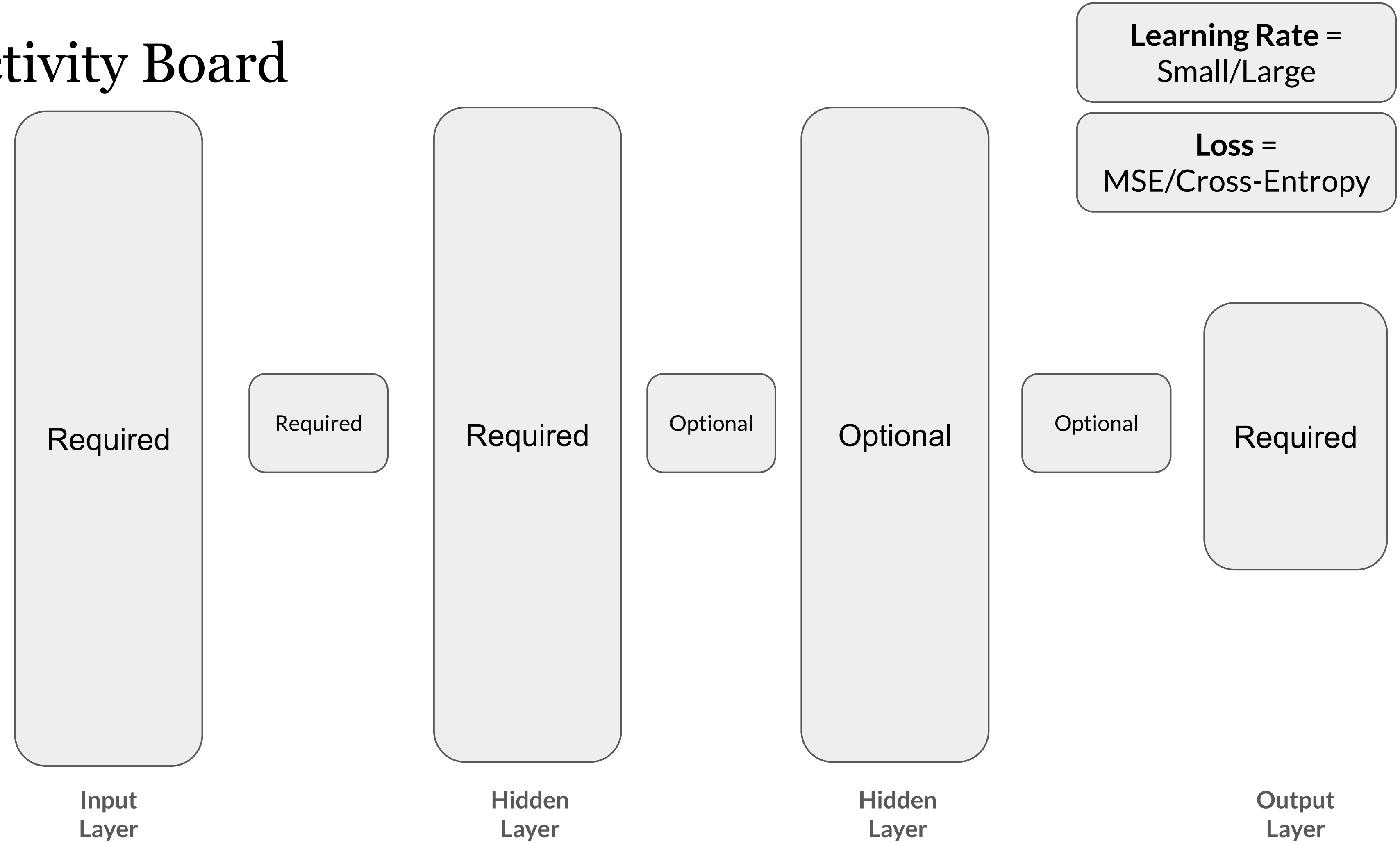
Tunable Hyperparameters

1. Relevant input parameters
2. Number of hidden layers
3. Number of nodes in each layer
4. Activation functions
5. Learning rate
6. Loss function

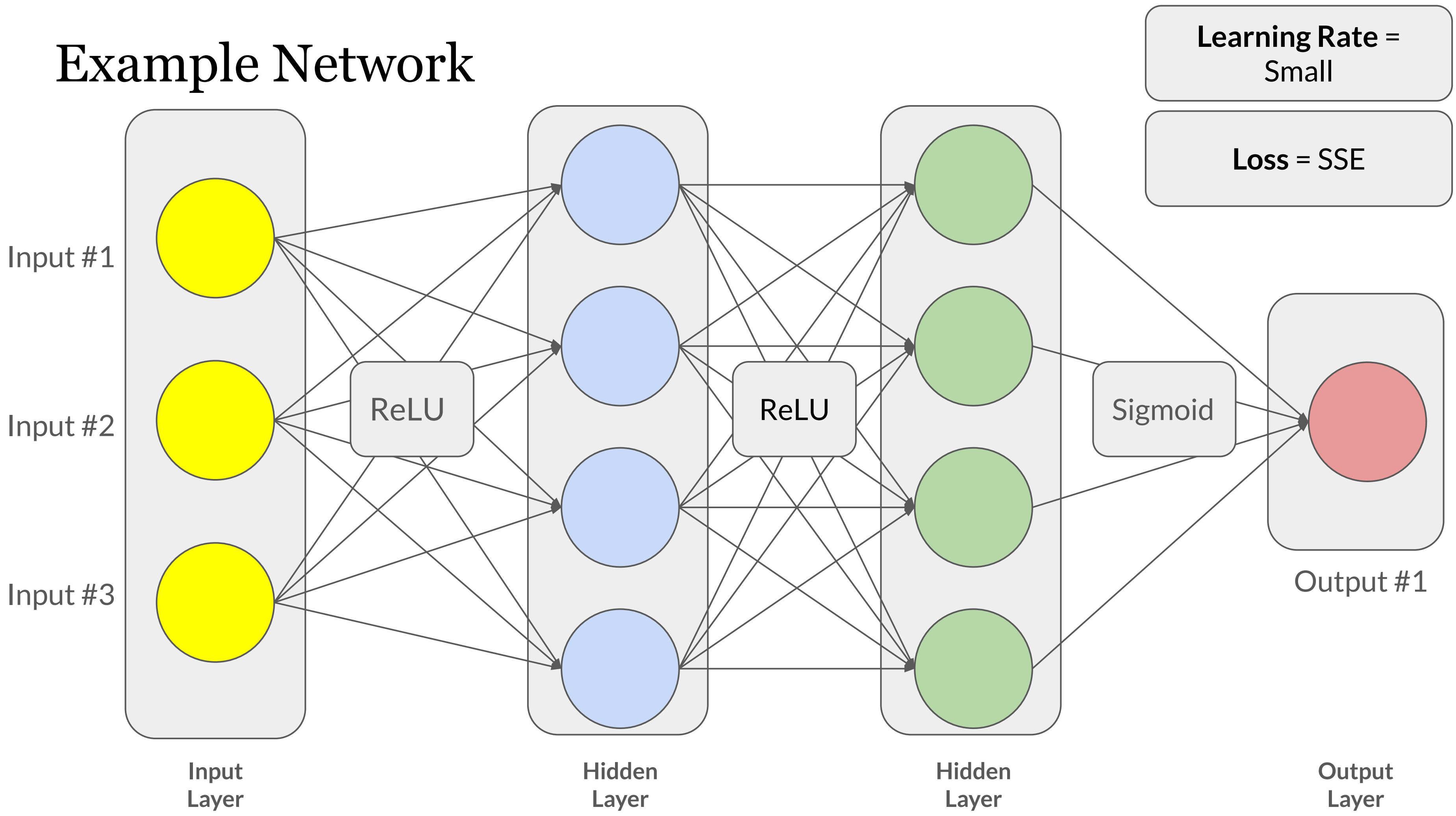
Your Job: Design a Neural Net!

- **Given a set of demographic survey data, you need to create a model that can determine whether someone is a two-percenter based on certain attributes**
- Implement the input layer, hidden layer(s), and activation function(s)
- Set a learning rate to determine how your model performs gradient descent
- Select a loss function to evaluate your model
- *Note that it is up to **you** to select what input parameters are important, the number of hidden layers, and the number of nodes in each layer. These will differ amongst groups.*

Activity Board



Example Network



Logistics

- Design the neural net in 10-15 minutes
- Present your design for 2 minutes
  - How did you choose the hyperparameters?
  - How are you satisfying the business needs?
  - What tradeoffs does your model address?
- **Important**: there is no *right* answer, so we're simply interested in the *why*

Rubric

Category	Max Points	Score
Presentation (flow, timing of 5 minute)	10	
Communication (public speaking)	10	
Catering to business (understandable, how it would help the business)	10	
Formatting	5	
Explanation of hyperparemeters: Why you chose the inputs you chose	15	
Explanation of hyperparemeters: Why you chose certain activation functions and the tradeoffs	15	
Explanation of hyperparemeters: How you chose the learning rate/reasoning	15	
Explanation of hyperparemeters: Explanation for your loss function	15	
Explanation of hyperparemeters: Explanation for the number of neurons	5	
Total	100	