Neural Networks

Overview

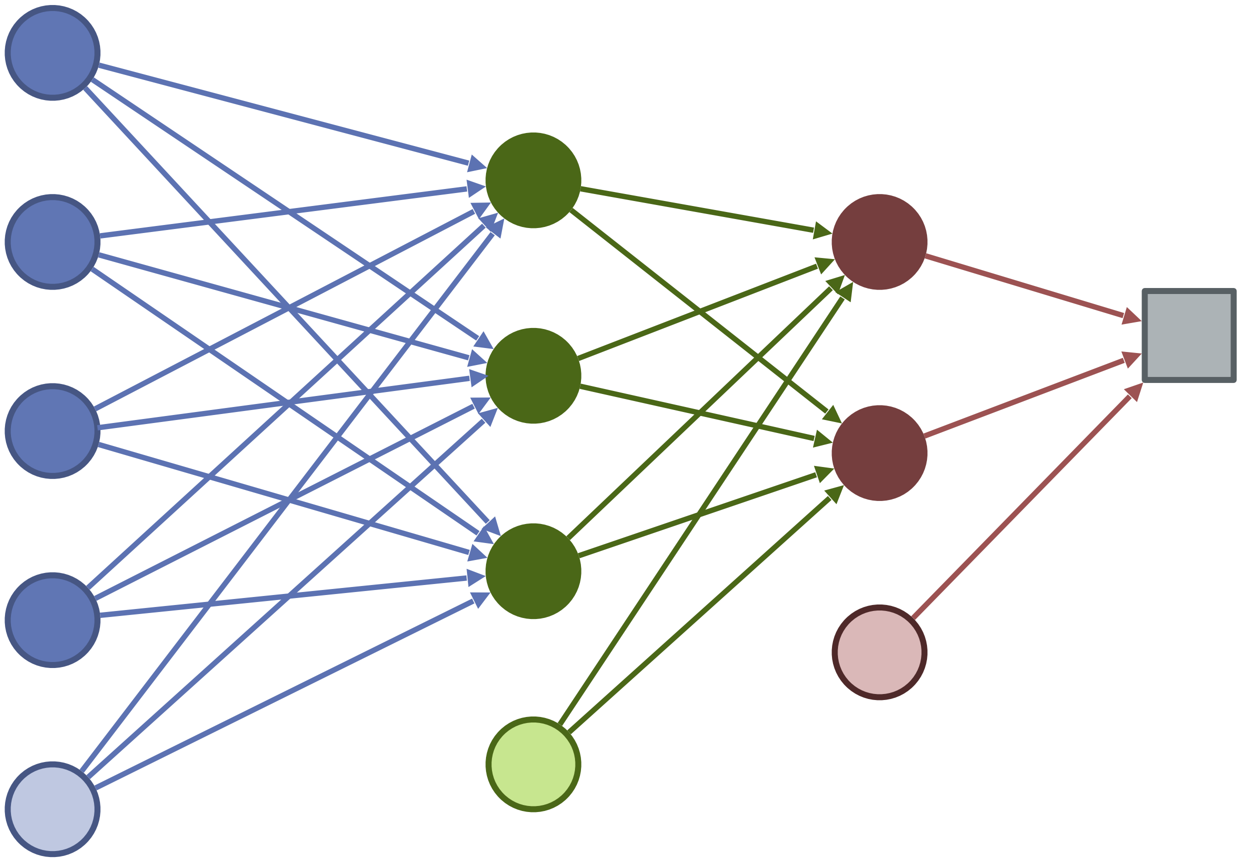
- Neural Networks are considered black-box models
- They are complex and do not provide any insight into variable relationships

- They have the potential to model very complicated patterns ("universal approximators")
- Can be used for both classification and continuous prediction tasks.

The History

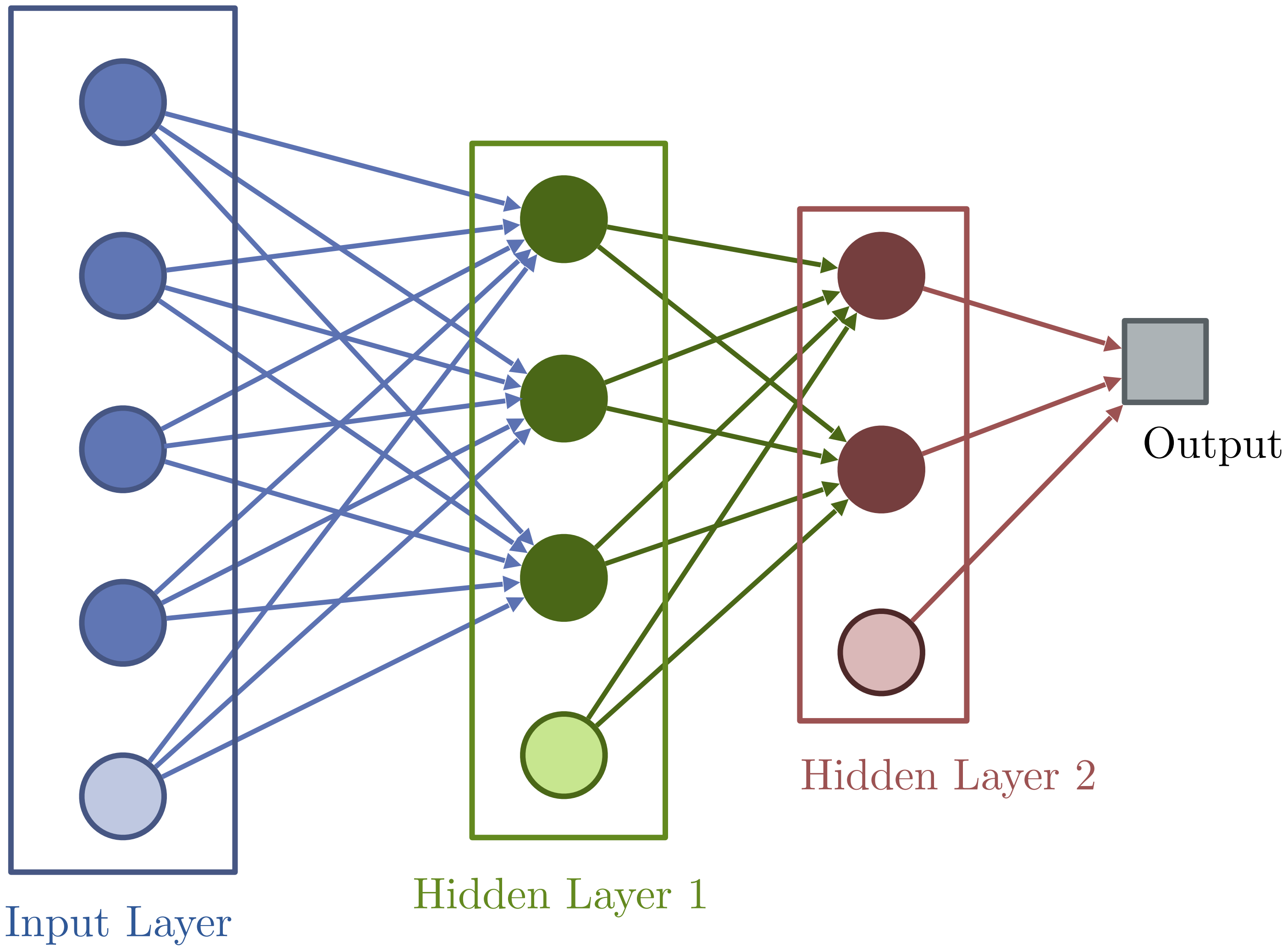
- Concept was welcomed with enthusiasm in 80's
- Didn't live up to expectations then
    - Too much hype, perhaps
- Overtaken by other black box techniques like Support Vector Machines with Kernels in 2000's
- Now in the age of image and visual recognition problems, neural networks have made comeback
    - Area of rapid development
    - Rebranded as "Deep Learning"
        - Recurrent Neural Networks
        - Convolutional Neural Networks
        - Feedforward Neural Networks

The Structure of a Neural Network

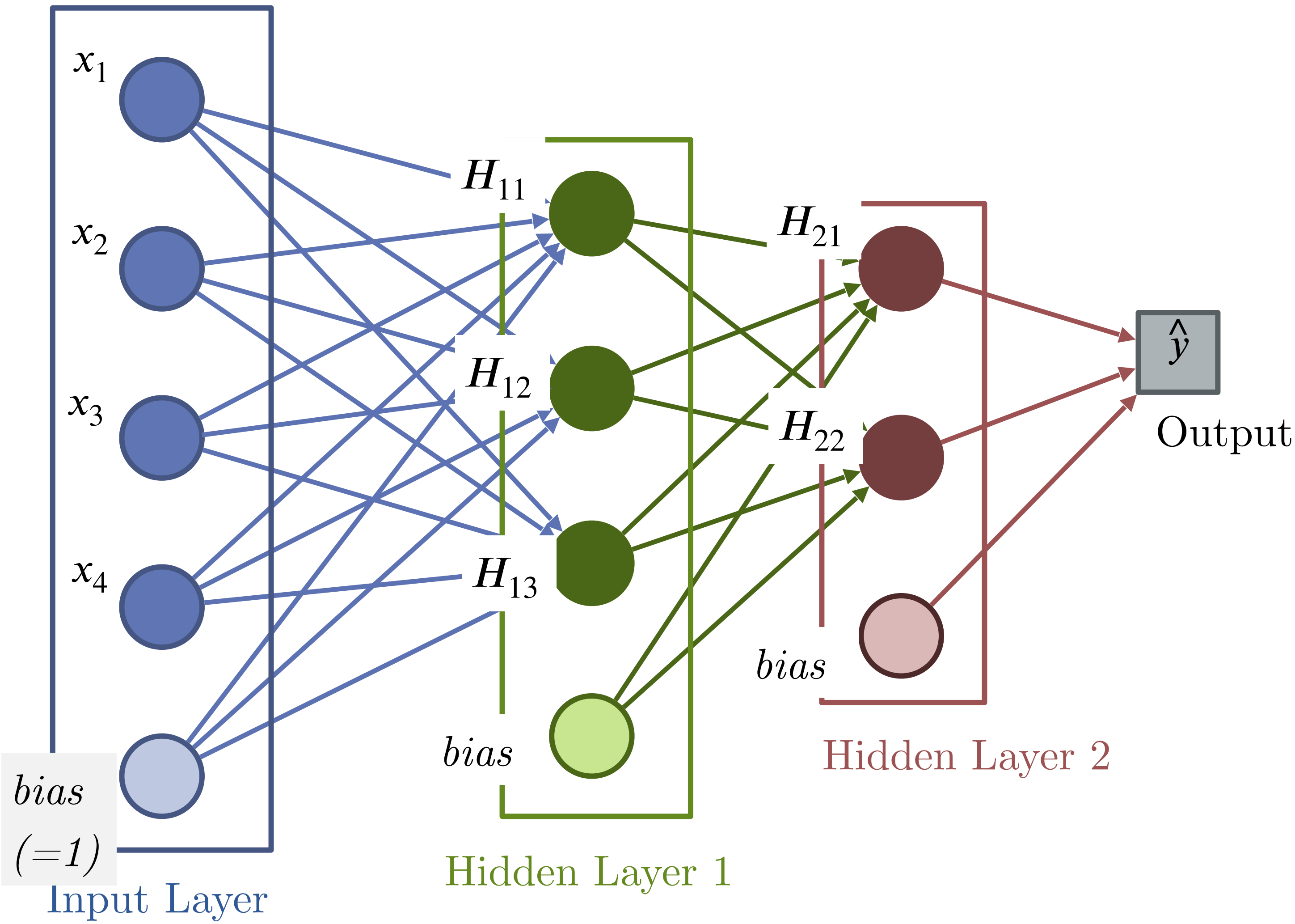


These Neural Networks are often called Multilayer Perceptrons (MLPs)

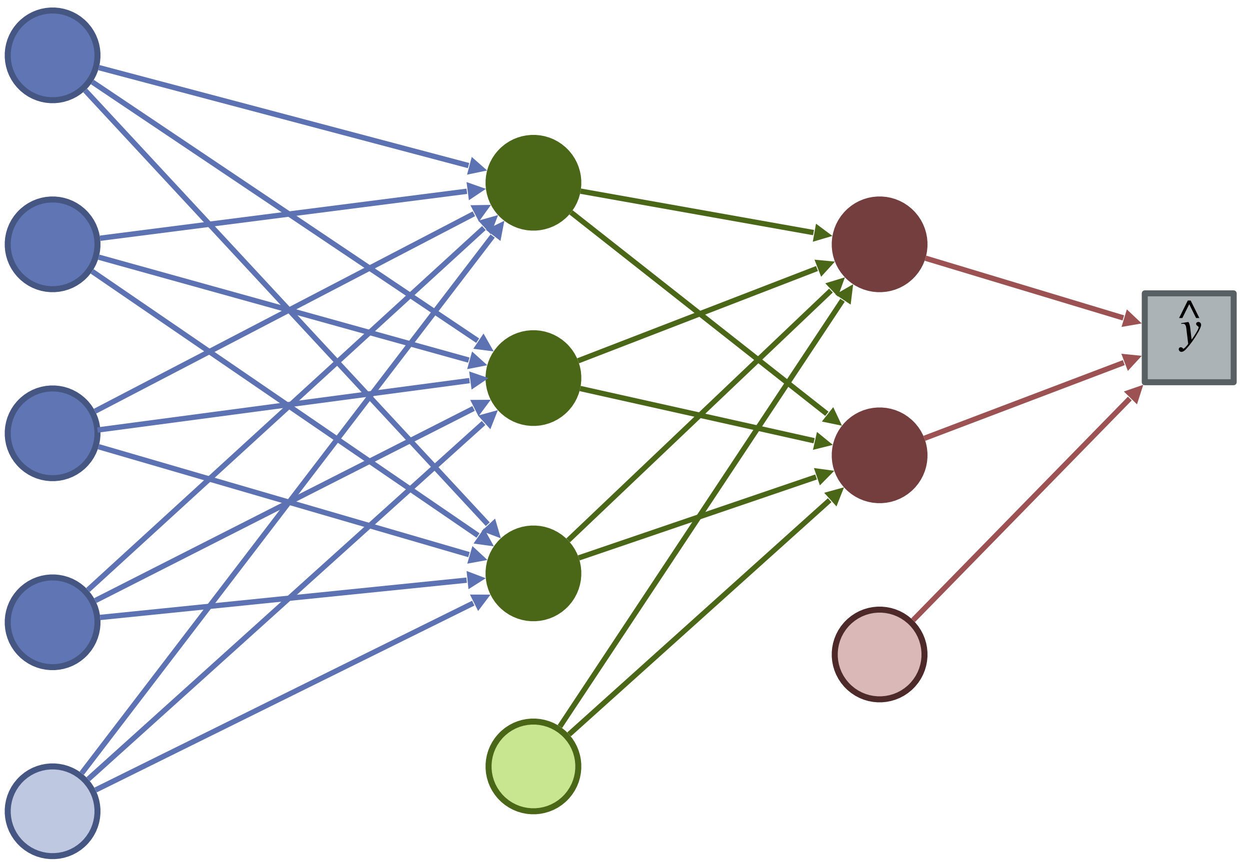
The Structure of a Neural Network



The Structure of a Neural Network

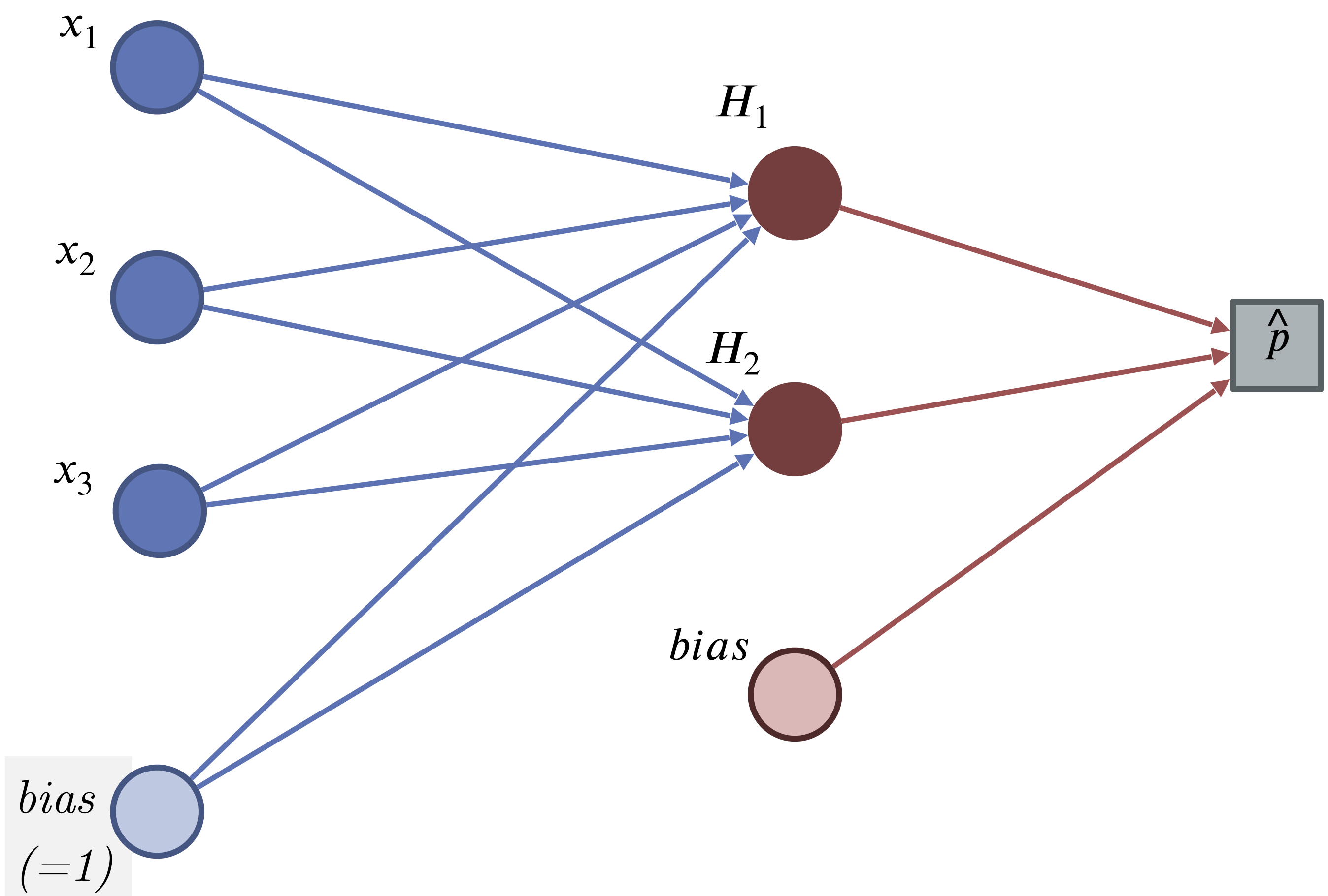


The Structure of a Neural Network



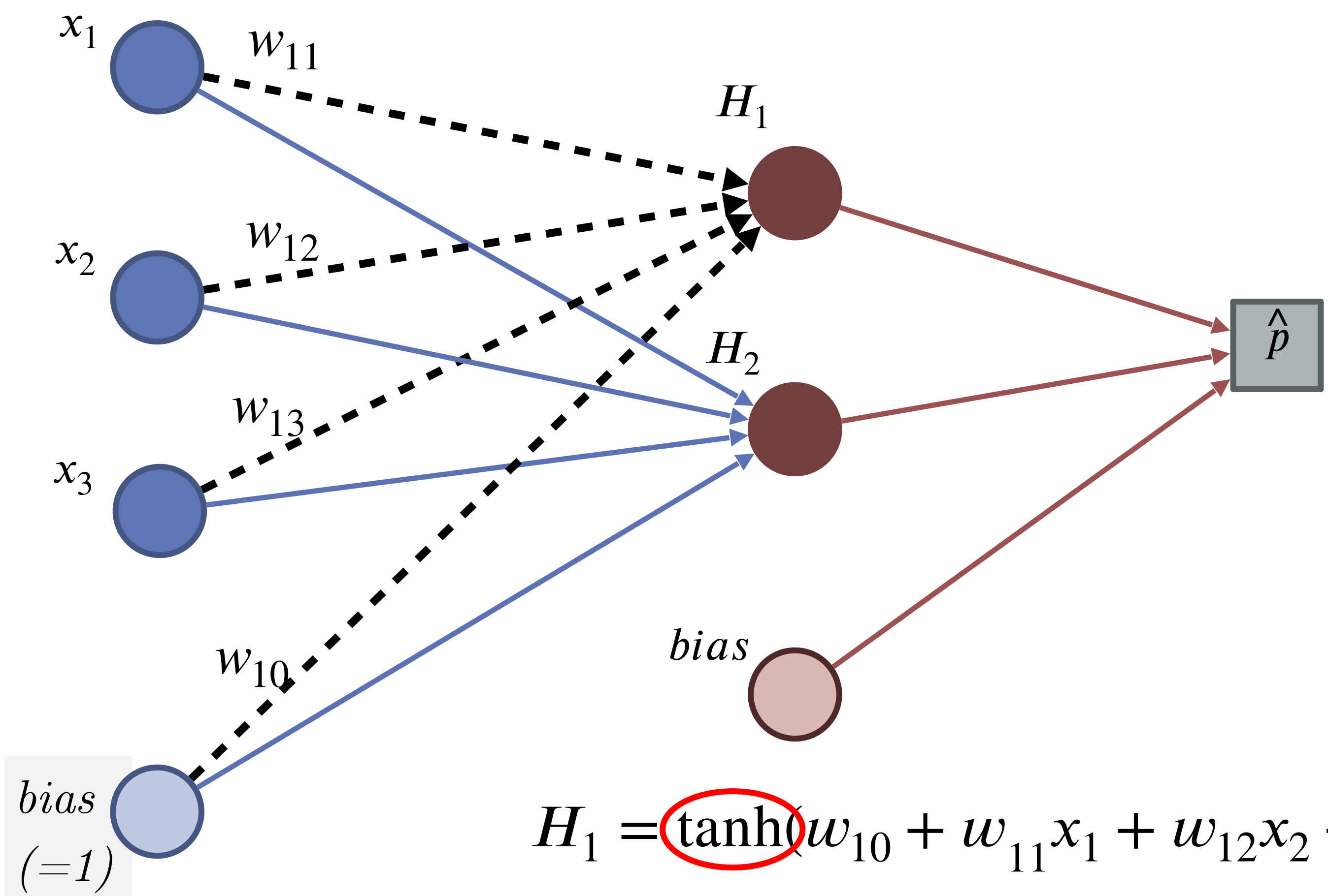
Associated with each line in this diagram is a parameter to be solved for!

A Simpler Neural Network



To avoid triple subscripts, let's simplify our network to 1 hidden layer and just 3 input variables. We'll assume a binary target

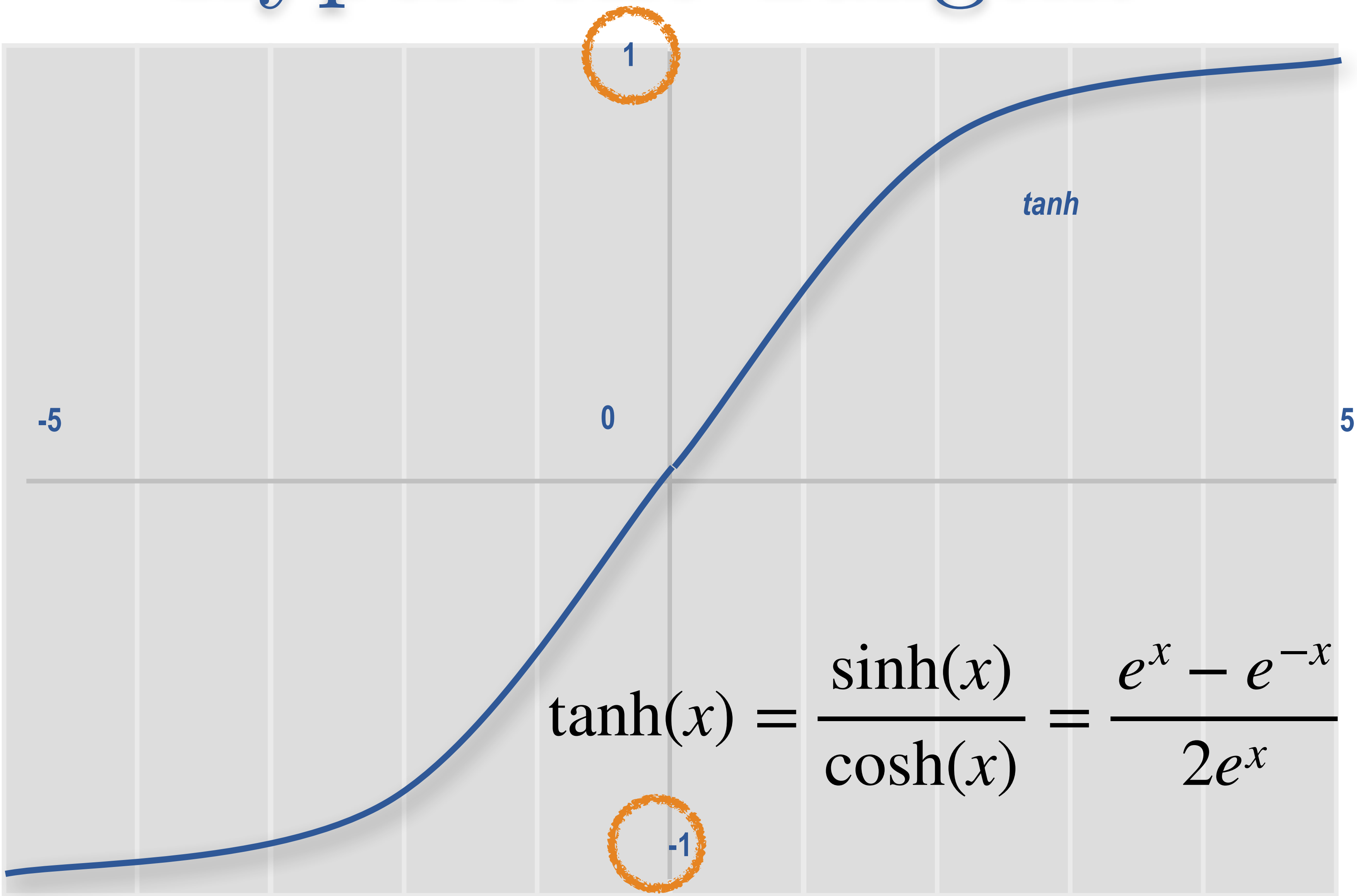
Math Structure of a Neural Network



$$H_1 = \tanh(w_{10} + w_{11}x_1 + w_{12}x_2 + w_{13}x_3)$$

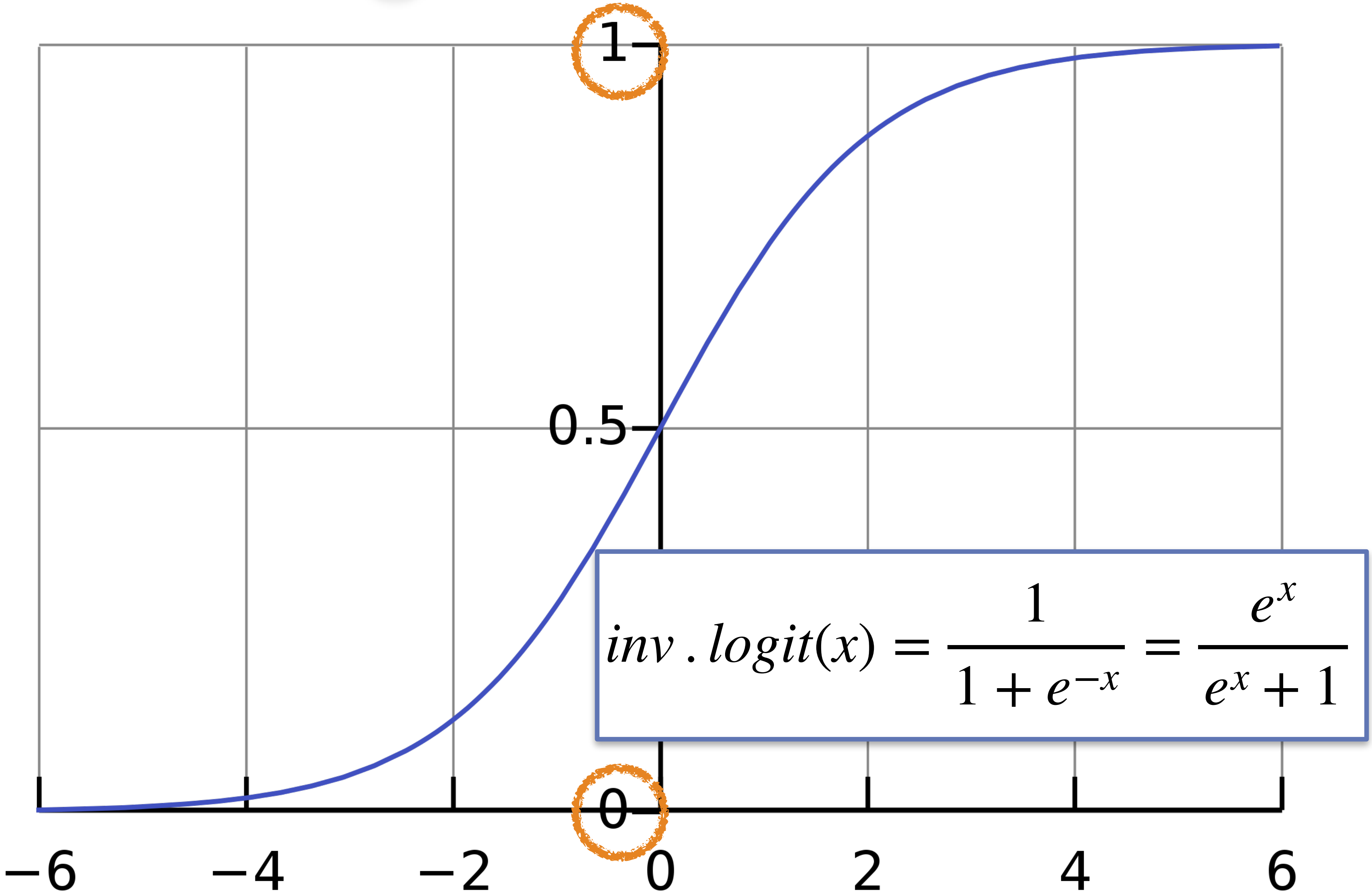
Hyperbolic tangent. One of many possible "sigmoid" functions. Range is -1 to 1. Related to logistic function.

Hyperbolic Tangent



$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{2e^{x}}$$

Logistic Function



$$inv.logit(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

Relationship between Hyperbolic Tangent and Logistic

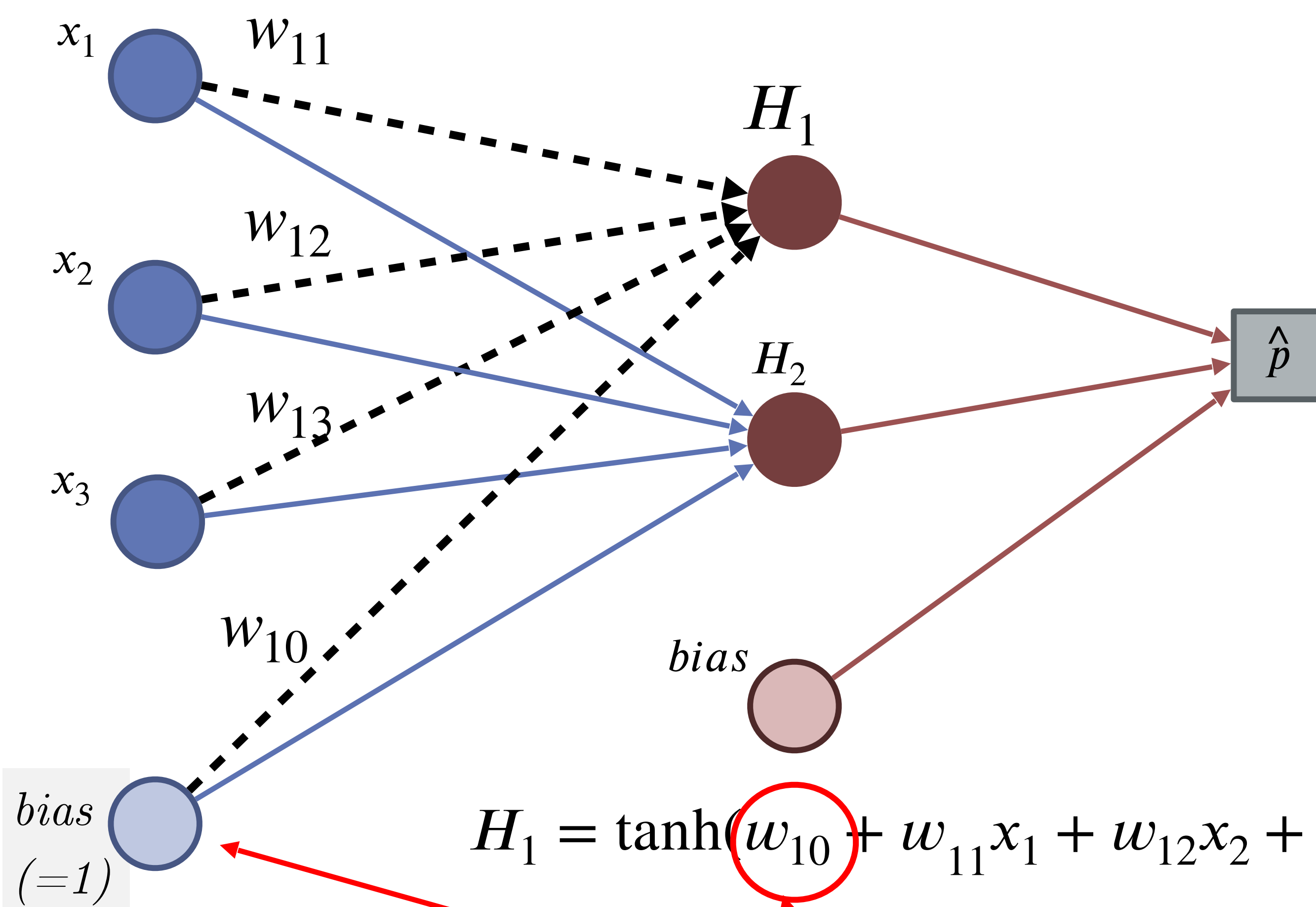
$$\tanh(x) = \frac{e^x - e^{-x}}{2e^x} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

$$inv.logit(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

$$\tanh(x) = 2inv.logit(2x) - 1$$

Takeaway: Neural networks can be thought of as compositions of logistic regressions

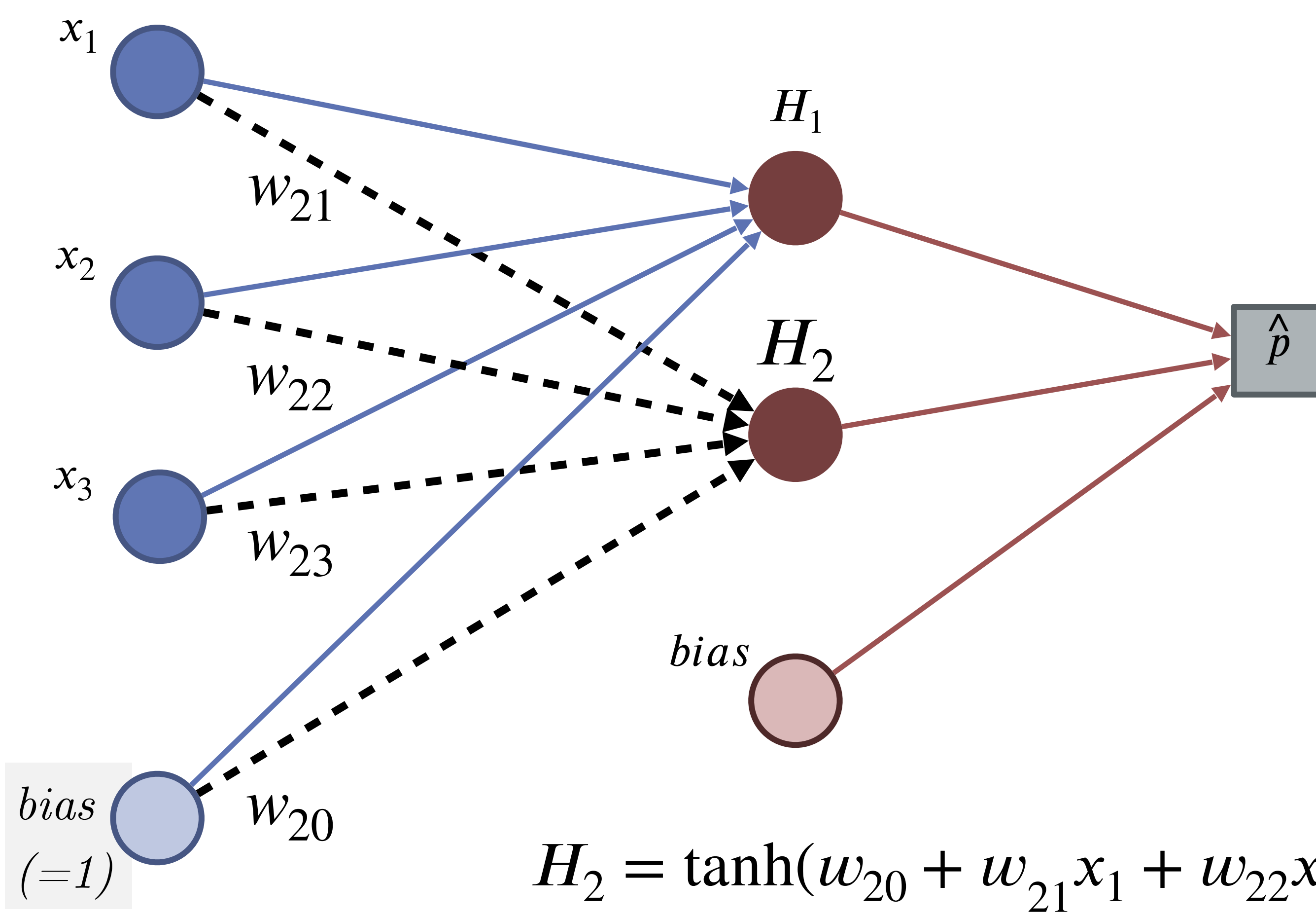
Math Structure of a Neural Network



$$H_1 = \tanh(w_{10} + w_{11}x_1 + w_{12}x_2 + w_{13}x_3)$$

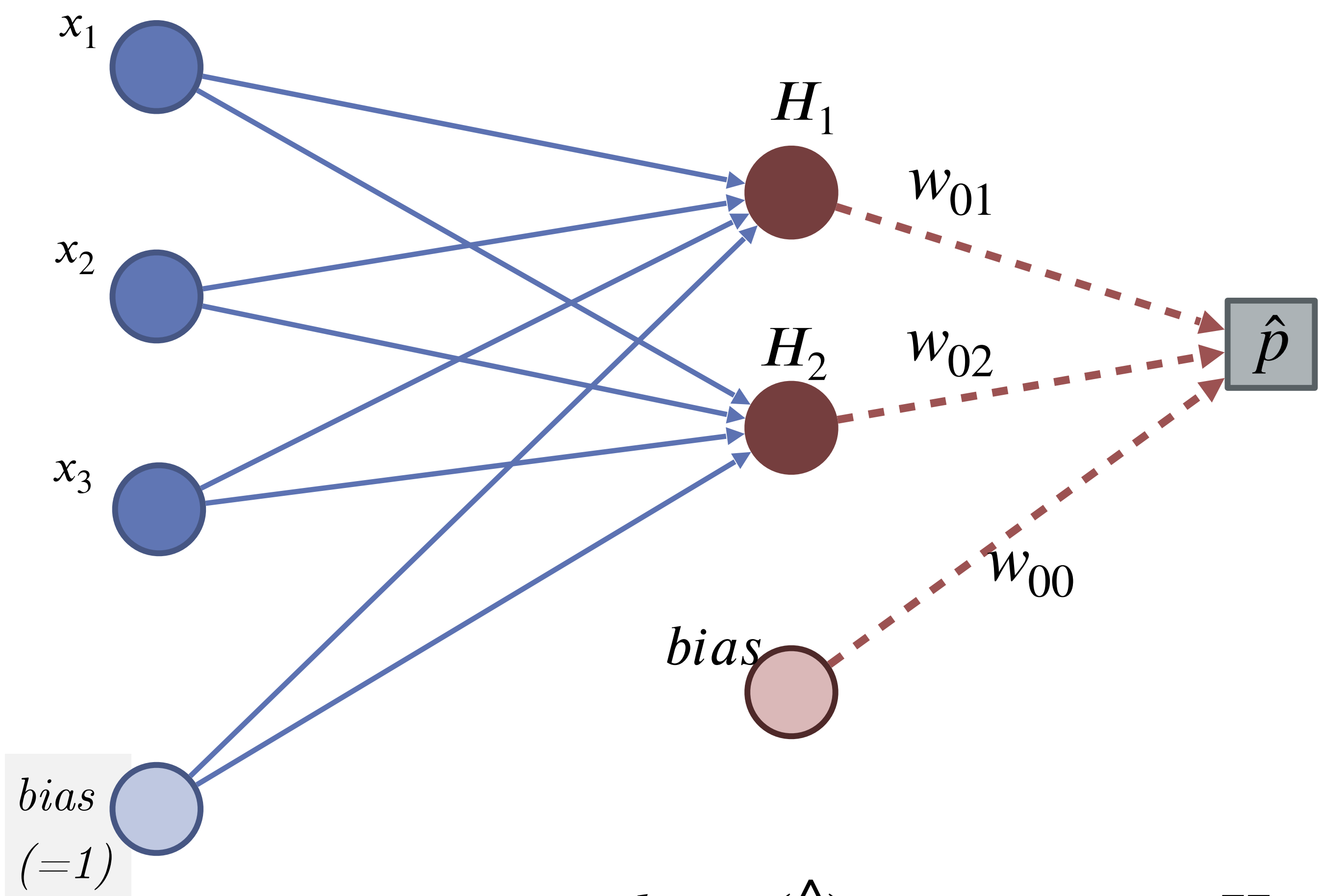
The intercept of each equation called "bias" term

Math Structure of a Neural Network



$$H_2 = \tanh(w_{20} + w_{21}x_1 + w_{22}x_2 + w_{23}x_3)$$

Math Structure of a Neural Network



$$logit(\hat{p}) = w_{00} + w_{01}H_1 + w_{02}H_2$$

Estimating Parameters of a Neural Network

- With just 3 input variables and 1 hidden layer containing 2 hidden units, we have to estimate **11** parameters!

$$H_1 = \tanh(w_{10} + w_{11}x_1 + w_{12}x_2 + w_{13}x_3)$$

$$H_2 = \tanh(w_{20} + w_{21}x_1 + w_{22}x_2 + w_{23}x_3)$$

$$logit\ (\hat{p}) = w_{00} + w_{01}H_1 + w_{02}H_2$$

- Weight estimates found by maximizing the log-likelihood function for a class target
- The process involves an algorithm called backpropagation to descend the gradient toward a solution.
  - Complete Example of backpropogation: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Estimating Parameters of a Neural Network

- With just 3 input variables and 1 hidden layer containing 2 hidden units, we have to estimate **11** parameters!

$$H_1 = \tanh(w_{10} + w_{11}x_1 + w_{12}x_2 + w_{13}x_3)$$

$$H_2 = \tanh(w_{20} + w_{21}x_1 + w_{22}x_2 + w_{23}x_3)$$

$$logit\,(\hat{p}) = w_{00} + w_{01}H_1 + w_{02}H_2$$

- Probability estimates are obtained by solving the logit equation for $\hat{p}$ for each observation:

$$\hat{p} = \frac{1}{1 + e^{-logit(\hat{p})}}$$

Backpropagation Algorithm

- **Forward phase:** Starting with some initial weights (often random), the calculations are passed through the network to the output layer where a predicted value is computed.
- **Backward phase:** The predicted value is compared to the actual value and the error is propagated backwards in the network to modify the connection weights.
- Repeat until something like convergence.

Standardization

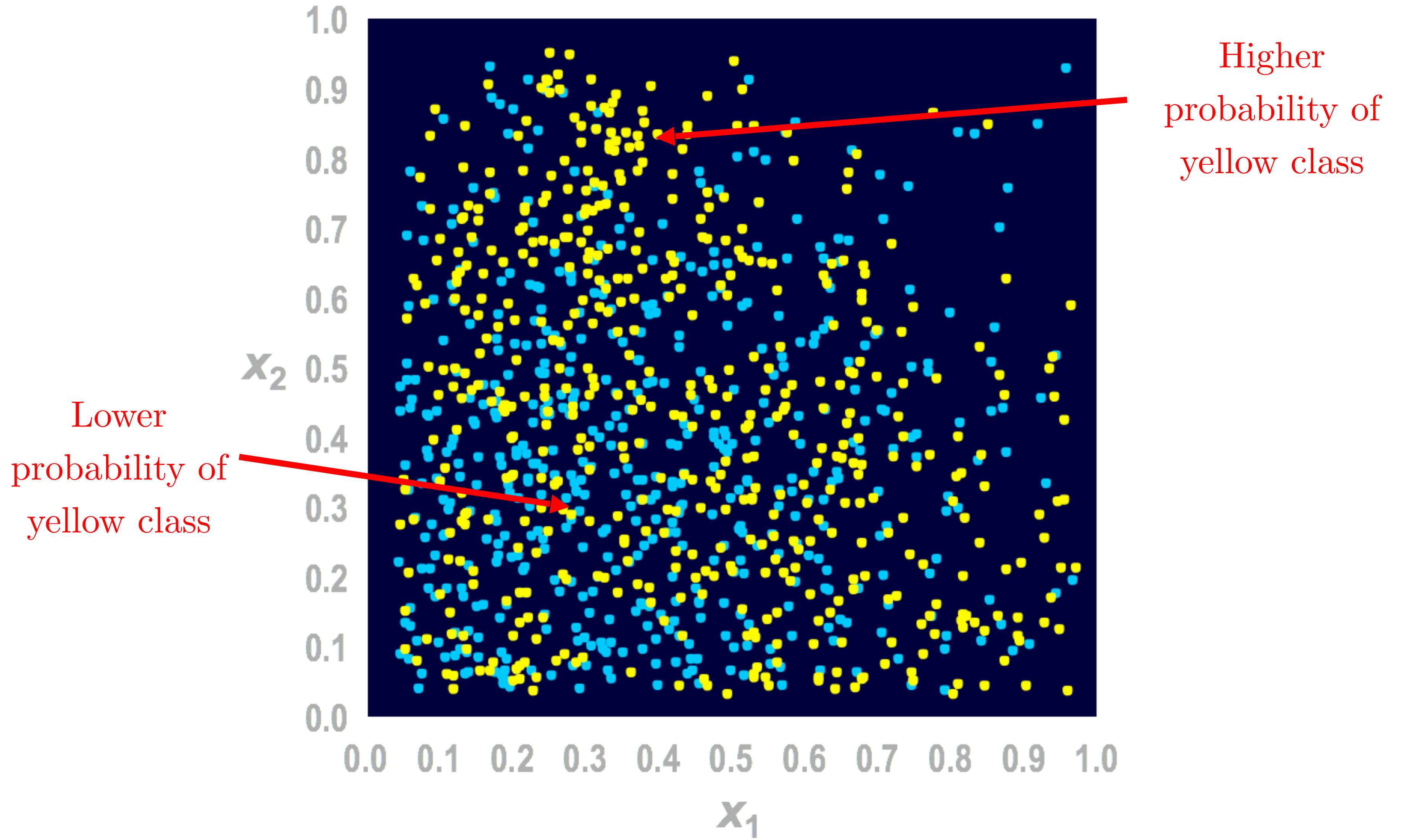
- Neural Networks **work best when input data are scaled** to a narrow range around 0
- For bell shaped data, statistical z-score standardization appropriate:

$$\frac{x - \bar{x}}{s_x}$$

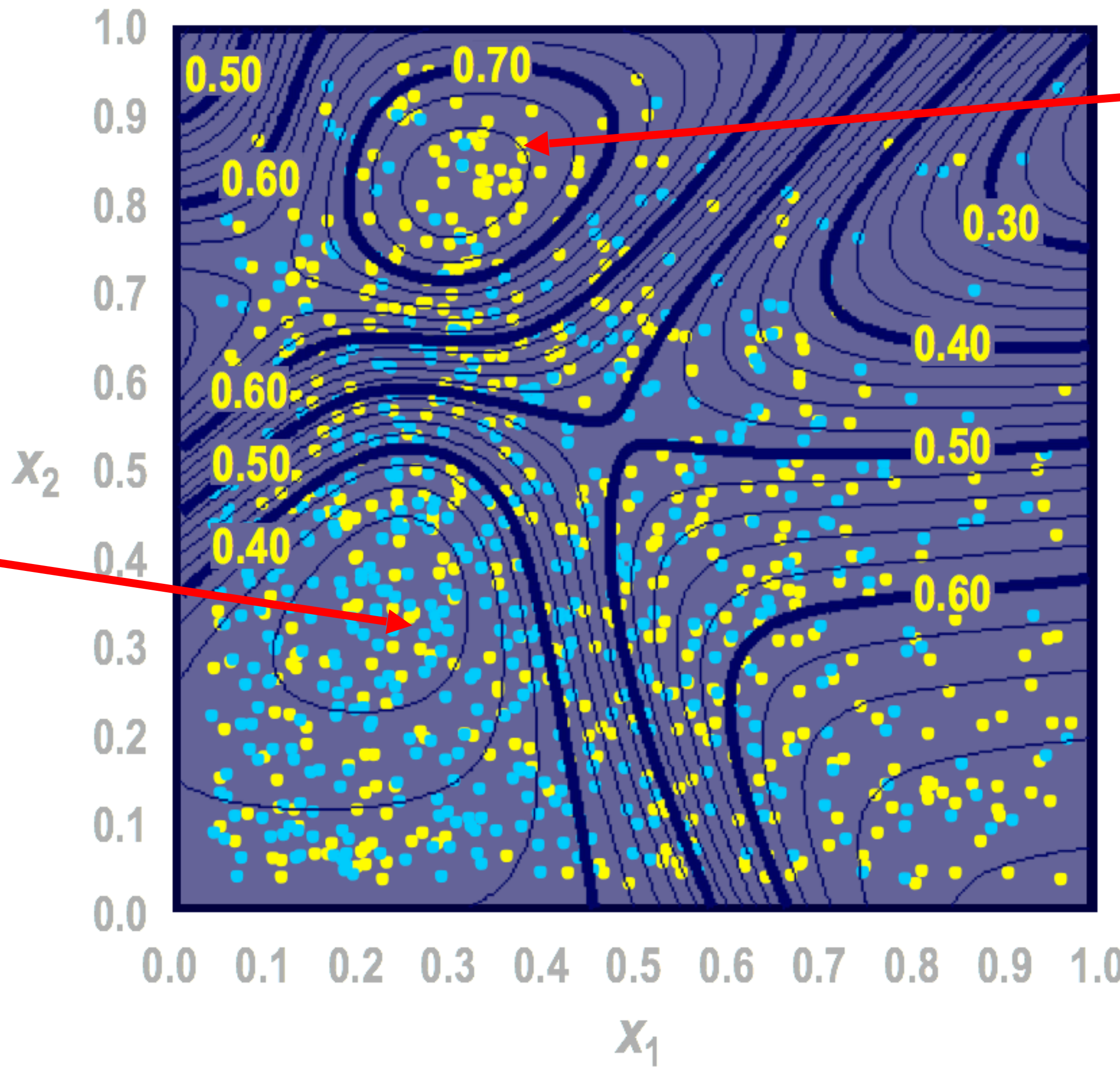
- For severely non-normal data (i.e. asymmetric), midrange standardization good alternative

$$\frac{x - \text{midrange}(x)}{0.5 \cdot \text{range}(x)} = \frac{x - \frac{max(x) + min(x)}{2}}{\frac{max(x) - min(x)}{2}}$$

Probability Surface of a Neural Network



Probability Surface of a Neural Network



Higher predicted probability of yellow class

Lower predicted probability of yellow class

Neural Networks Summary

Advantages

- Can be adapted to **nominal or continuous target variables**
- Capable of modeling **complex nonlinear patterns**
- Make **no assumptions** about the data's distributions.

Disadvantages

- Neural Networks have **no mechanism for variable selection**. You provide inputs. All inputs are used.
- **No insights into the variable importance**
  - Signs of weights can cancel each other out through the networks
  - Each input gets weight for each hidden unit which then get combined
- Extremely **computationally intensive** (slowwww to train)
- **Many hyperparameters** (# levels, # neurons per level, activations)
- **Prone to overfitting** training data

Predicting the Strength of Concrete

• • •

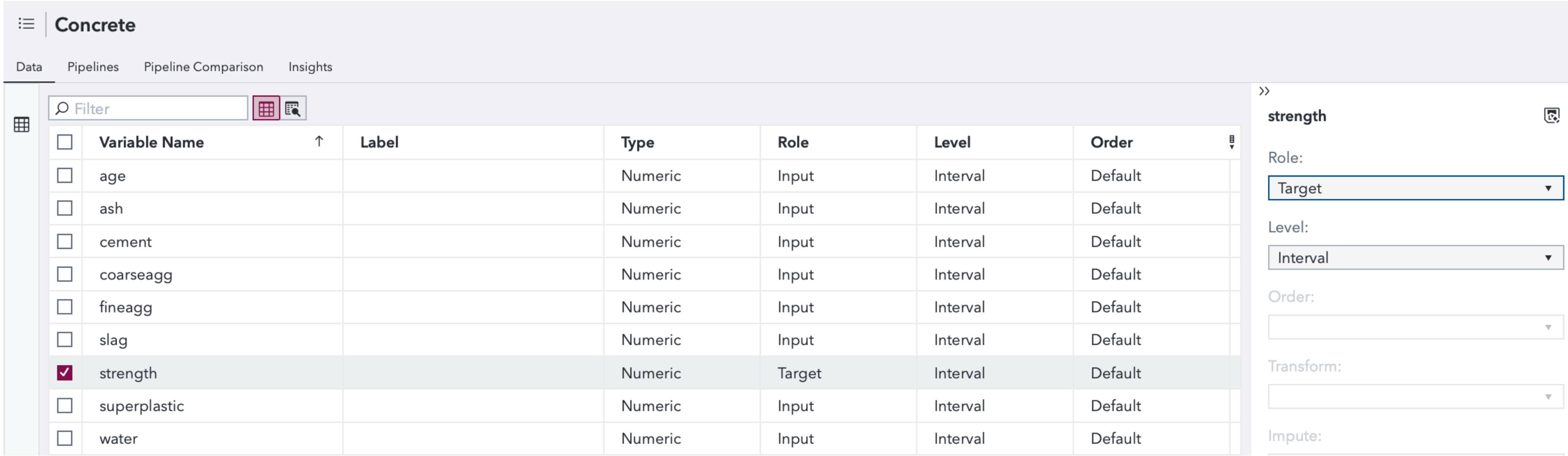
Nonlinearity abounds

Concrete Data

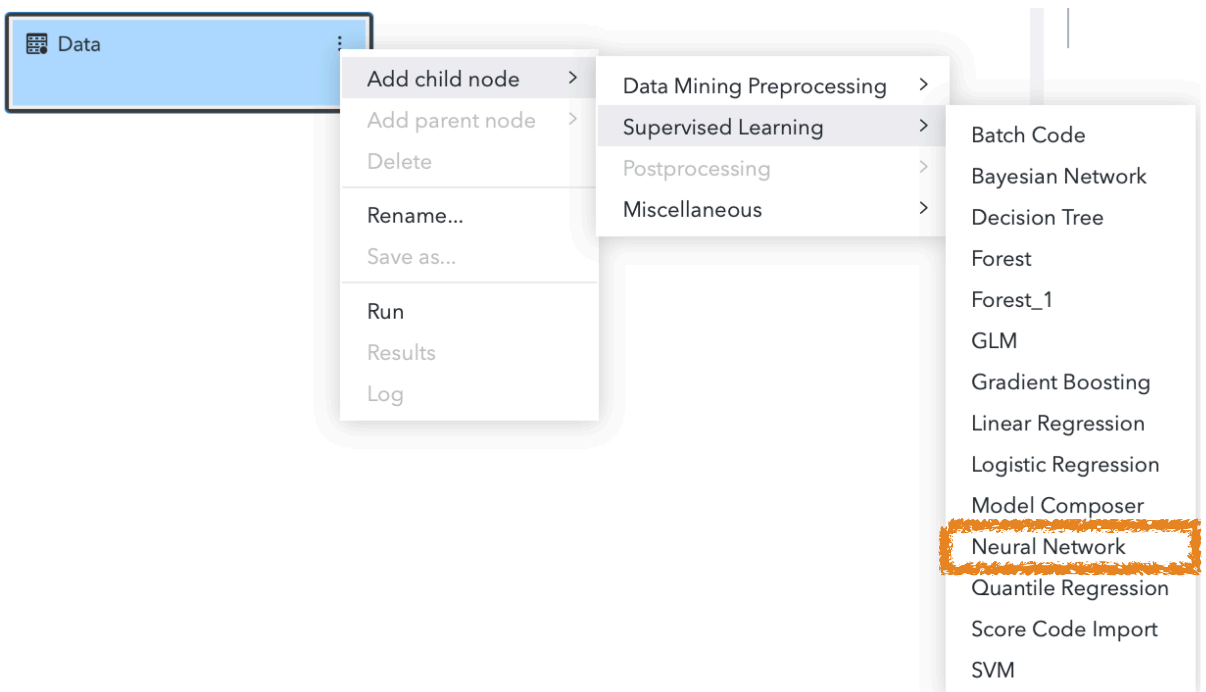
1. Cement
2. Slag
3. Ash
4. Water
5. Superplastic
6. Course Aggregate
7. Fine Aggregate
8. Age

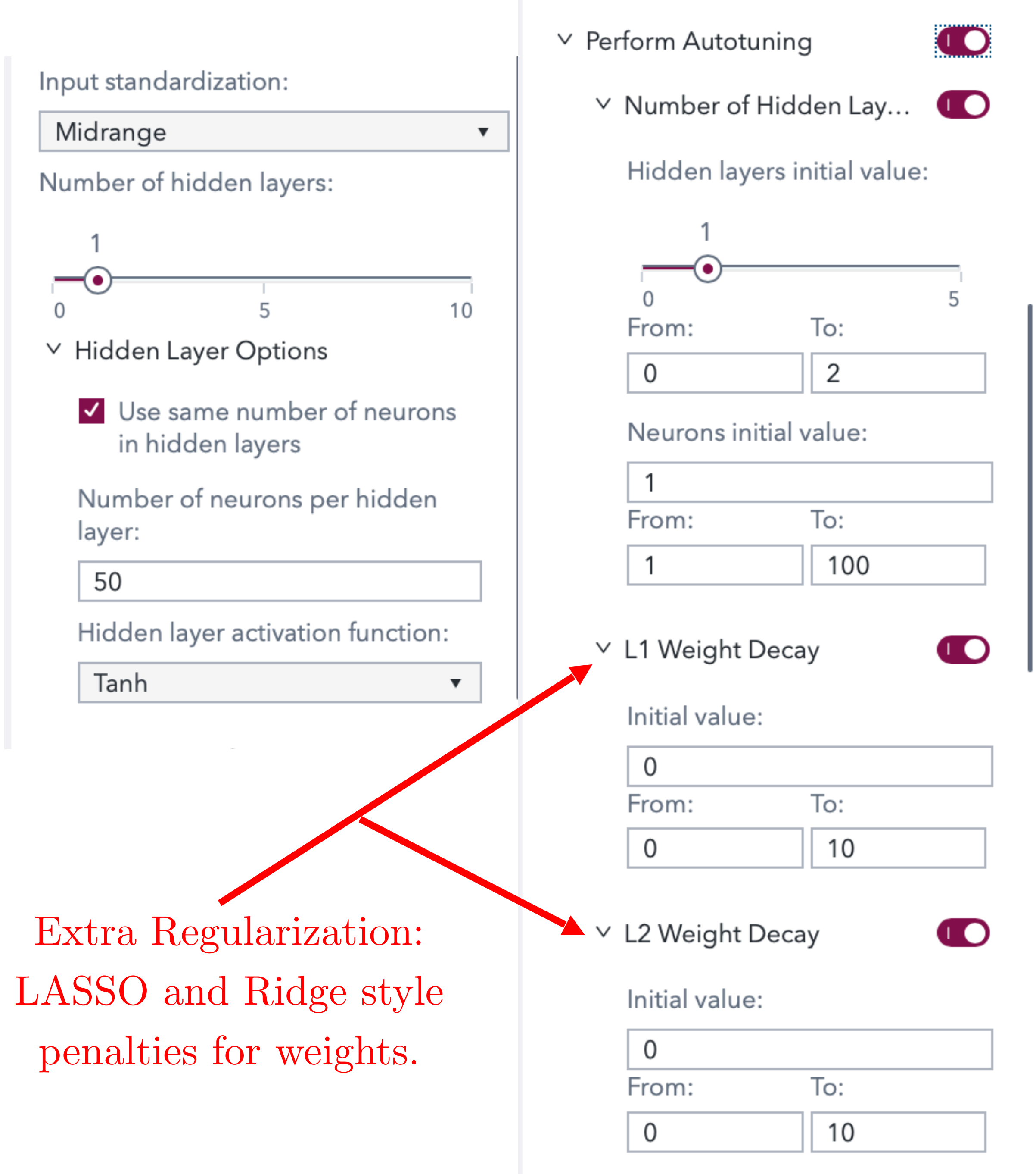
Strength

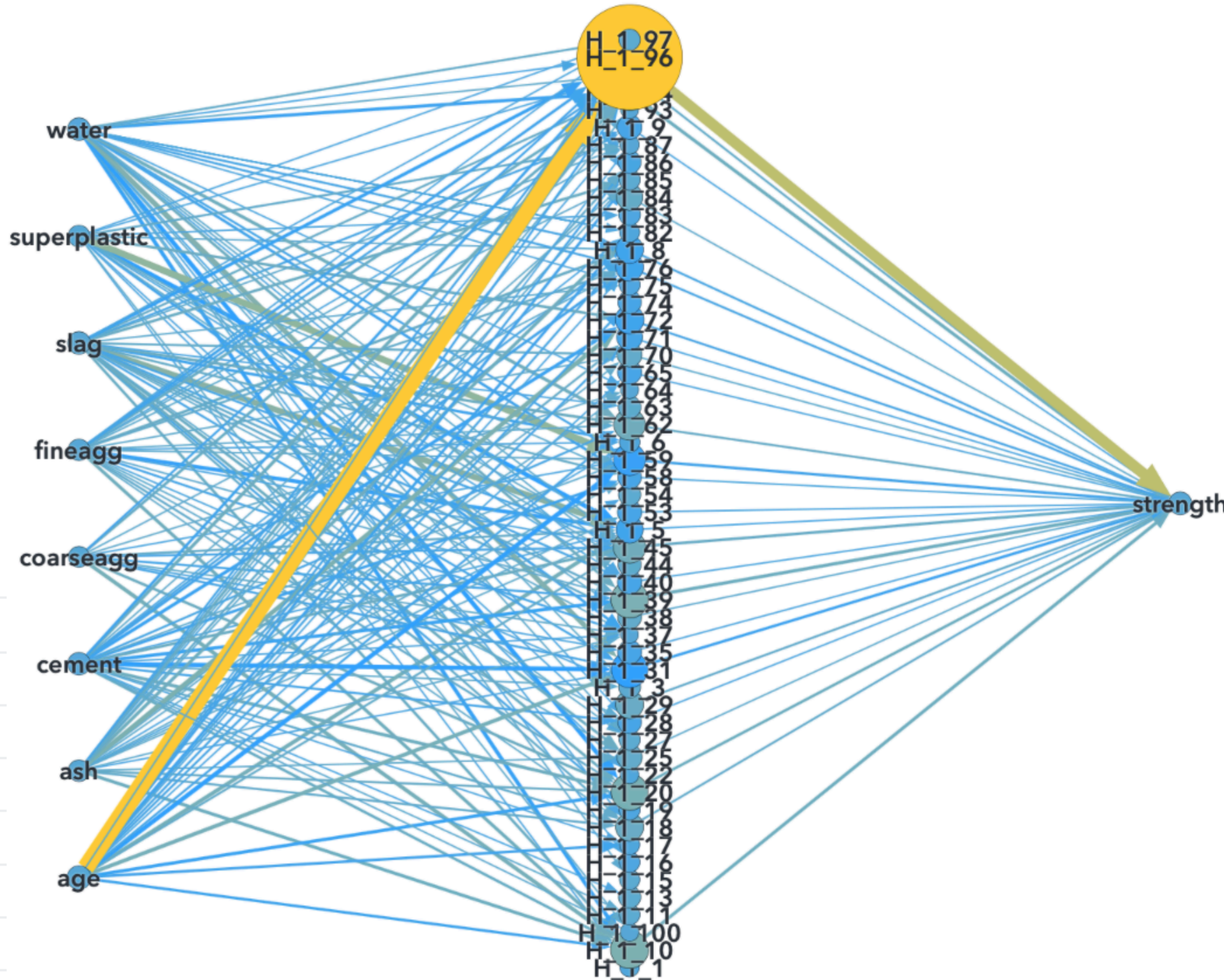
The predictor variables, all measured in kg per cubic meter of mix (aside from age which is measured in days), are used to model the compressive strength of concrete as measured in MPa.



	Variable Name	Label	Type	Role	Level	Order
☐	age		Numeric	Input	Interval	Default
☐	ash		Numeric	Input	Interval	Default
☐	cement		Numeric	Input	Interval	Default
☐	coarseagg		Numeric	Input	Interval	Default
☐	fineagg		Numeric	Input	Interval	Default
☐	slag		Numeric	Input	Interval	Default
☑	strength		Numeric	Target	Interval	Default
☐	superplastic		Numeric	Input	Interval	Default
☐	water		Numeric	Input	Interval	Default

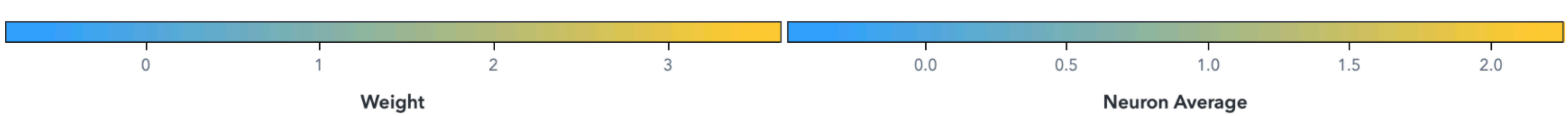



Input standardization:
Midrange
Number of hidden layers:
1
0
5
10
Hidden Layer Options
Use same number of neurons in hidden layers
Number of neurons per hidden layer:
50
Hidden layer activation function:
Tanh
Perform Autotuning
Number of Hidden Lay...
Hidden layers initial value:
1
0
5
From:
To:
0
2
Neurons initial value:
1
From:
To:
1
100
L1 Weight Decay
Initial value:
0
From:
To:
0
10
L2 Weight Decay
Initial value:
0
From:
To:
0
10
Extra Regularization:
LASSO and Ridge style
penalties for weights.



Autotune Best Configuration

Parameter	Value
Evaluation	52
Hidden Layers	1
Neurons in Hidden Layer 1	100
Neurons in Hidden Layer 2	0
L1 Regularization	0
L2 Regularization	0
Average Square Error	42.458



```r
set.seed(11117)
nnet33 = neuralnet(strength~cement+slag+ash+water+superplastic+coarseagg+fineagg+age,
                   data=train_norm, rep=2, stepmax=10^6, hidden=c(3,3))
plot(nnet33)
```