10-301/601: Introduction to Machine Learning Lecture 11 – Neural Networks

Matt Gormley & Henry Chai

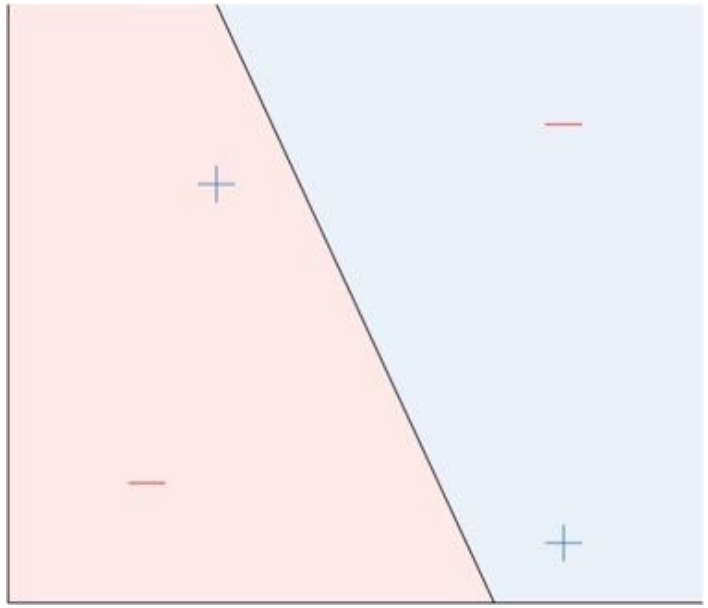
9/30/24

Front Matter

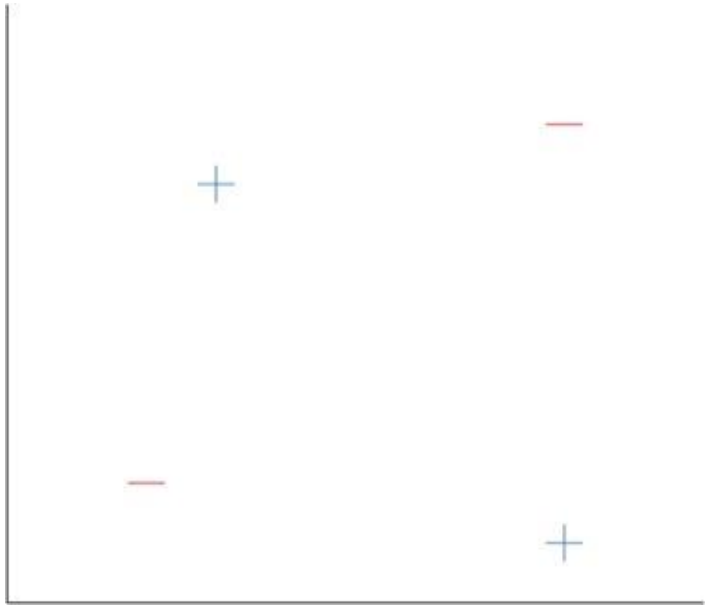
- Announcements
  - Exam 1 on 9/30 (today!) from 6:30 PM - 8:30 PM
    - Make sure you check the seating chart (on Piazza) so that you know where you're going tonight!
  - Homework 4 released 9/30 (today!), due 10/9 at 11:59 PM

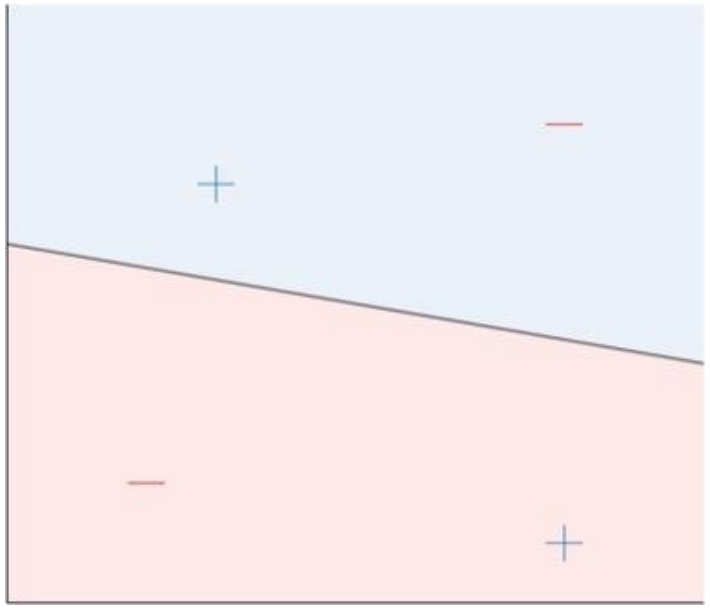
Biological Neural Network

Source: https://science-art.com/image/?id=2971&m=168&pagename=neural_network_3d#.W_wrzJNKhUM



h_1

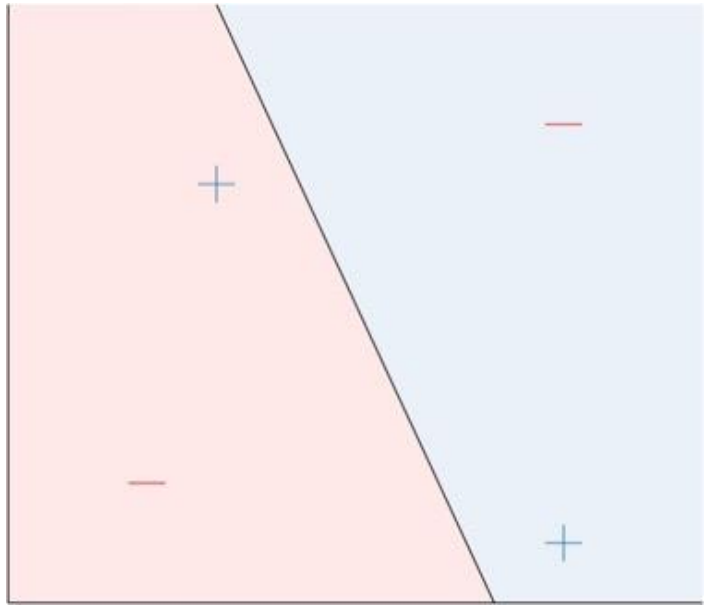




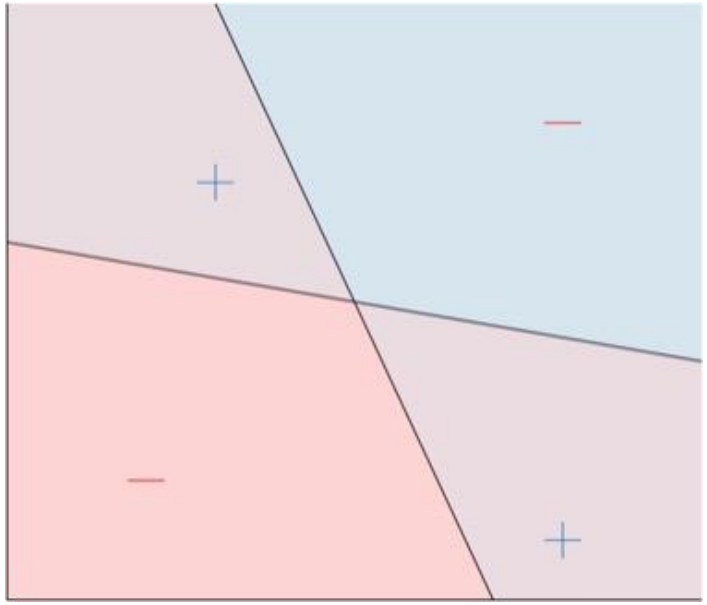
h_2

Perceptrons

- Linear model for classification
- $h(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^T \boldsymbol{x})$
- Predictions are $+1$ or -1

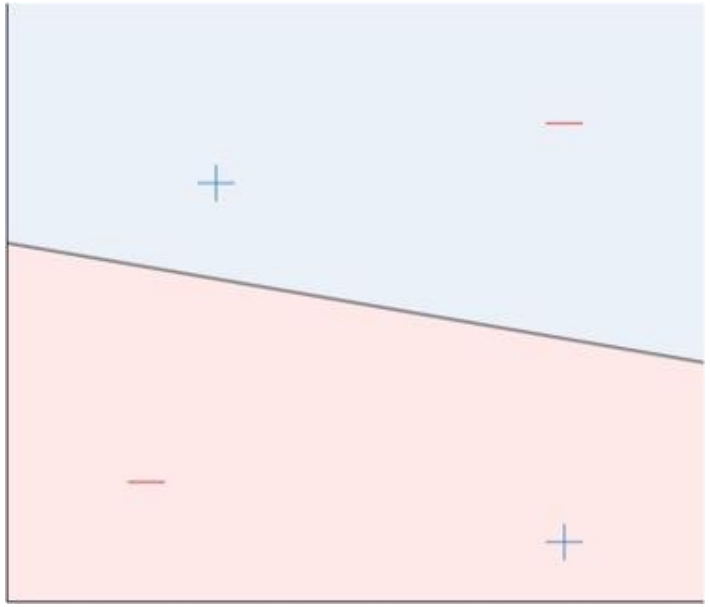


h_1



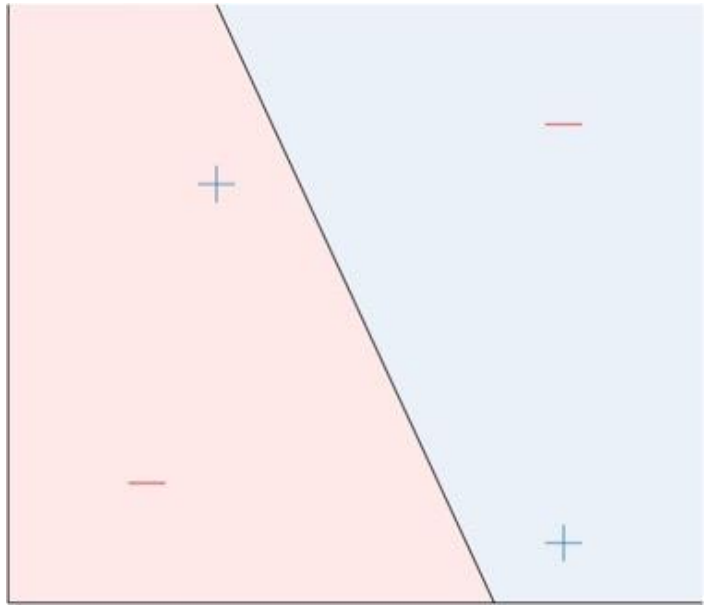
h_1

h_2

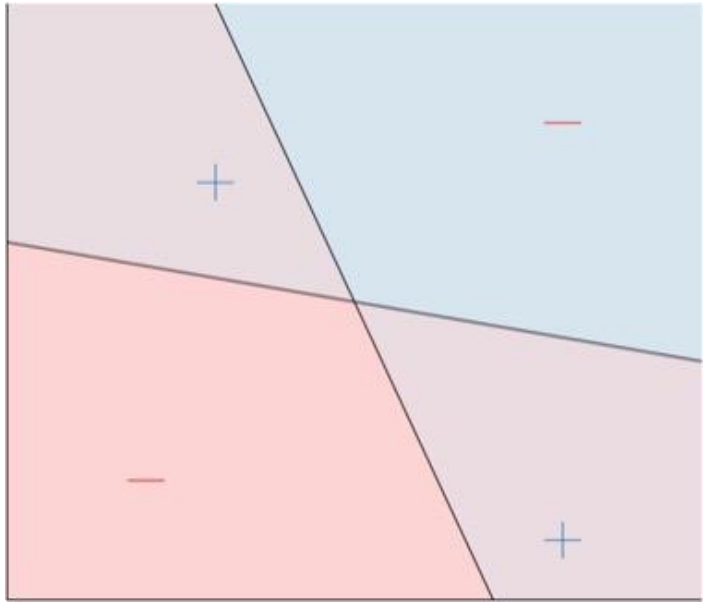


h_2

Combining Perceptrons

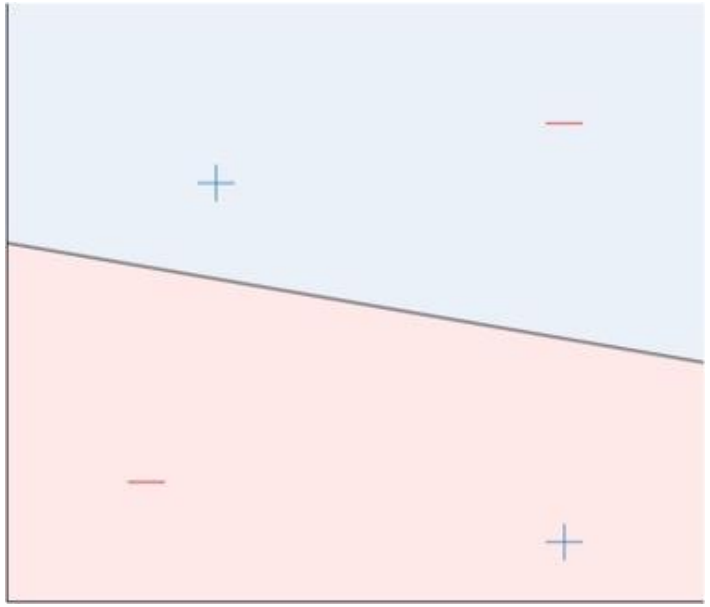


h_1



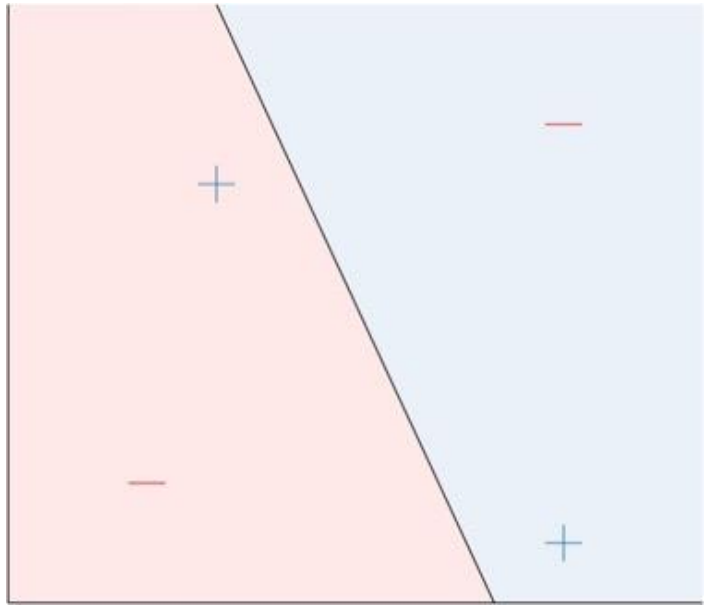
h_1

h_2

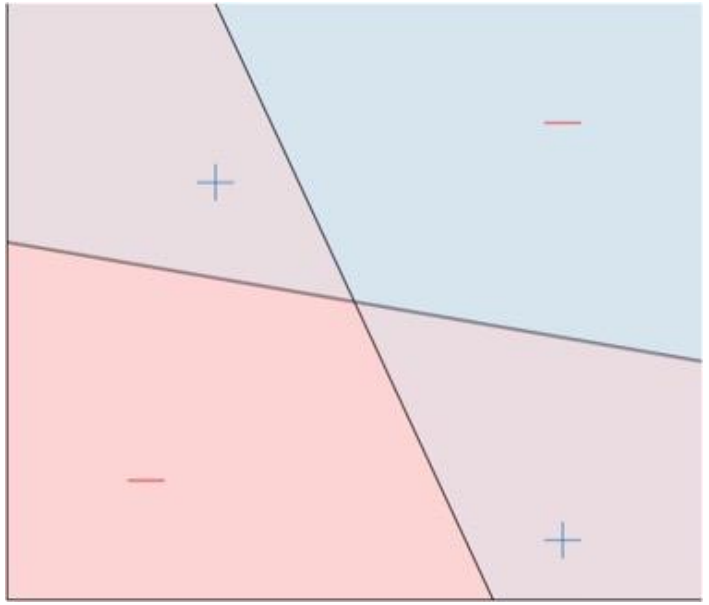


h_2

$$h(\boldsymbol{x}) = \begin{cases} +1 \text{ if } (h_1(\boldsymbol{x}) = +1 \text{ and } h_2(\boldsymbol{x}) = -1) \text{ or } (h_1(\boldsymbol{x}) = -1 \text{ and } h_2(\boldsymbol{x}) = +1) \\ -1 \text{ otherwise} \end{cases}$$

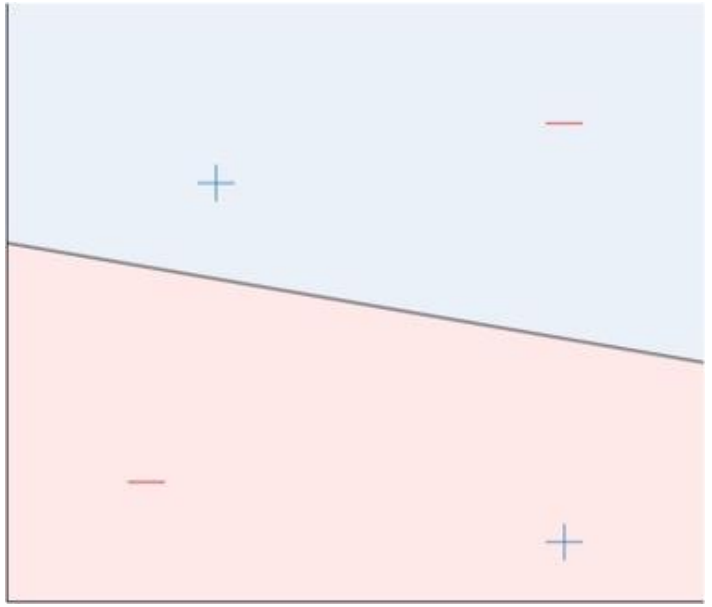


h_1



h_1

h_2



h_2

$$h(\boldsymbol{x}) = OR\left(AND\left(h_1(\boldsymbol{x}), \neg h_2(\boldsymbol{x})\right), AND(\neg h_1(\boldsymbol{x}), h_2(\boldsymbol{x}))\right)$$

Boolean Algebra

- Boolean variables are either $+1$ (“true”) or -1 (“false”)
- Basic Boolean operations:
  - Negation: $\neg z = -1 * z$
  - And: $AND(z_1, z_2) = \begin{cases} +1 \text{ if both } z_1 \text{ and } z_2 \text{ equal } +1 \\ -1 \text{ otherwise} \end{cases}$
  - Or: $OR(z_1, z_2) = \begin{cases} +1 \text{ if either } z_1 \text{ or } z_2 \text{ equals } +1 \\ -1 \text{ otherwise} \end{cases}$

Boolean Algebra

- Boolean variables are either $+1$ (”true”) or -1 (“false”)
- Basic Boolean operations
  - Negation: $\neg z = -1 * z$
  - And: $AND(z_1, z_2) = \text{sign}(z_1 + z_2 - 1.5)$
  - Or: $OR(z_1, z_2) = \text{sign}(z_1 + z_2 + 1.5)$

Boolean Algebra

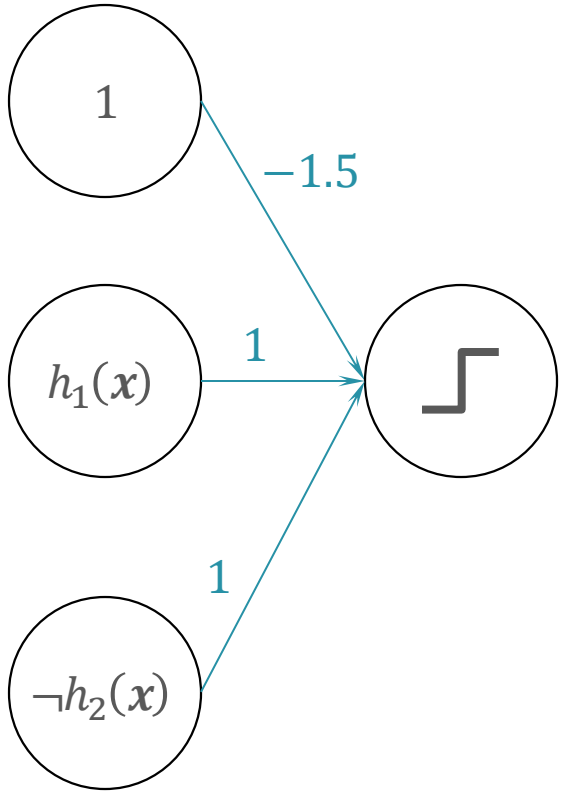
- Boolean variables are either $+1$ (”true”) or -1 (“false”)
- Basic Boolean operations
  - Negation: $\neg z = -1 * z$
  - And: $AND(z_1, z_2) = \text{sign}\left([-1.5, 1, 1]\begin{bmatrix}1\\z_1\\z_2\end{bmatrix}\right)$
  - Or: $OR(z_1, z_2) = \text{sign}\left([1.5, 1, 1]\begin{bmatrix}1\\z_1\\z_2\end{bmatrix}\right)$

Building a Network

$$h(\boldsymbol{x}) = OR\left(AND\left(h_1(\boldsymbol{x}), \neg h_2(\boldsymbol{x})\right), AND\left(\neg h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\right)\right)$$

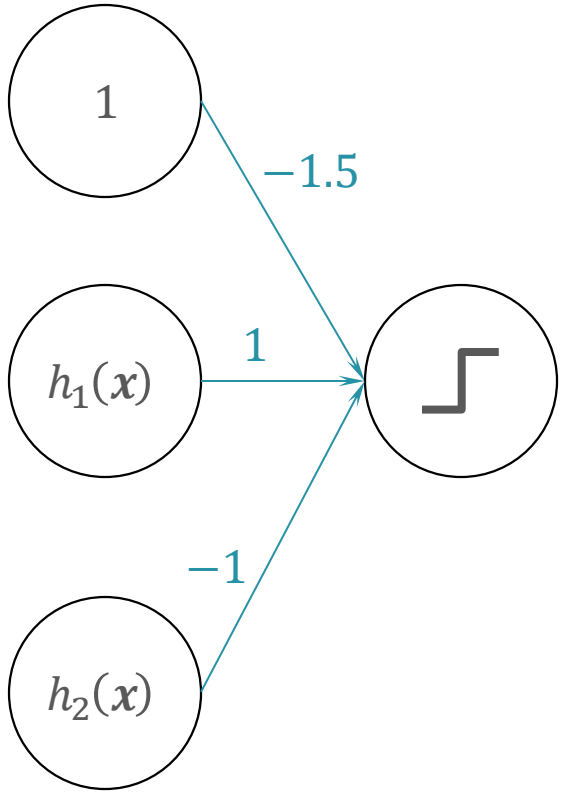
Building a Network

$$h(\boldsymbol{x}) = OR\Big(AND\big(h_1(\boldsymbol{x}), \neg h_2(\boldsymbol{x})\big), AND\big(\neg h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\big)\Big)$$



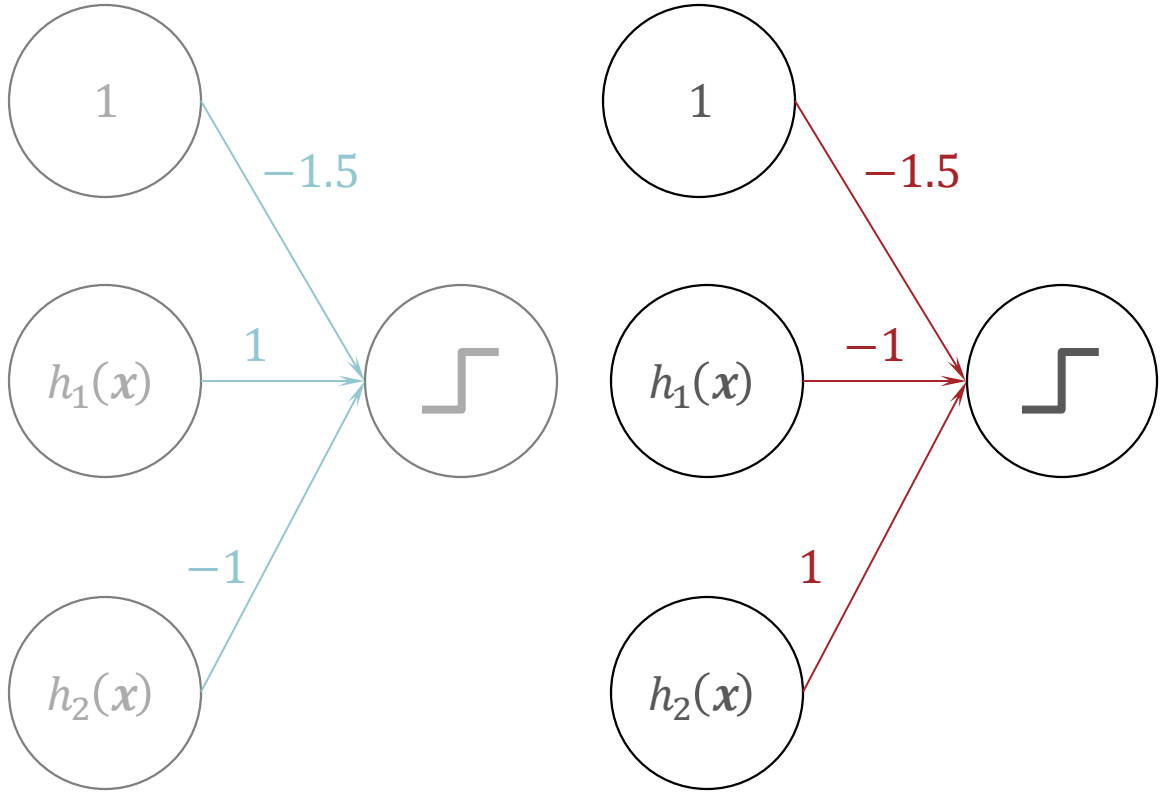
Building a Network

$$h(\boldsymbol{x}) = OR\Big(AND\big(h_1(\boldsymbol{x}), \neg h_2(\boldsymbol{x})\big), AND\big(\neg h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\big)\Big)$$



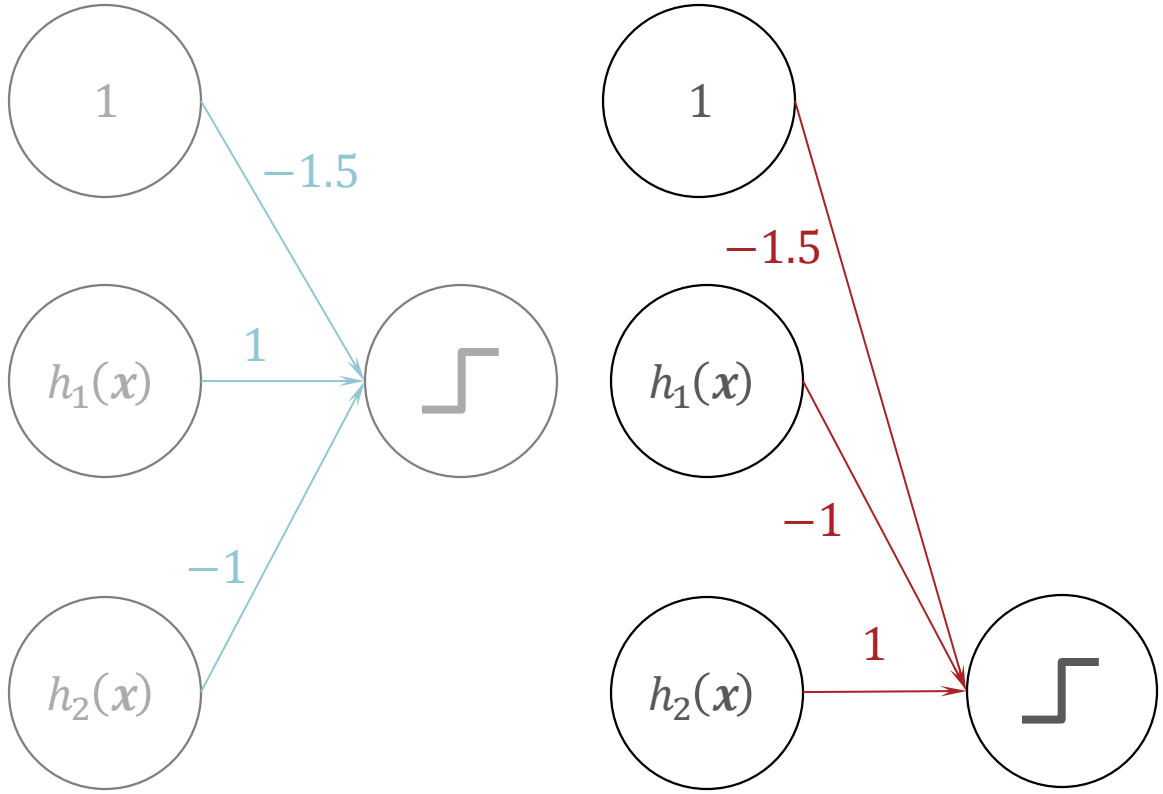
Building a Network

$$h(\boldsymbol{x}) = OR\left(AND\left(h_1(\boldsymbol{x}), \neg h_2(\boldsymbol{x})\right), AND\left(\neg h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\right)\right)$$



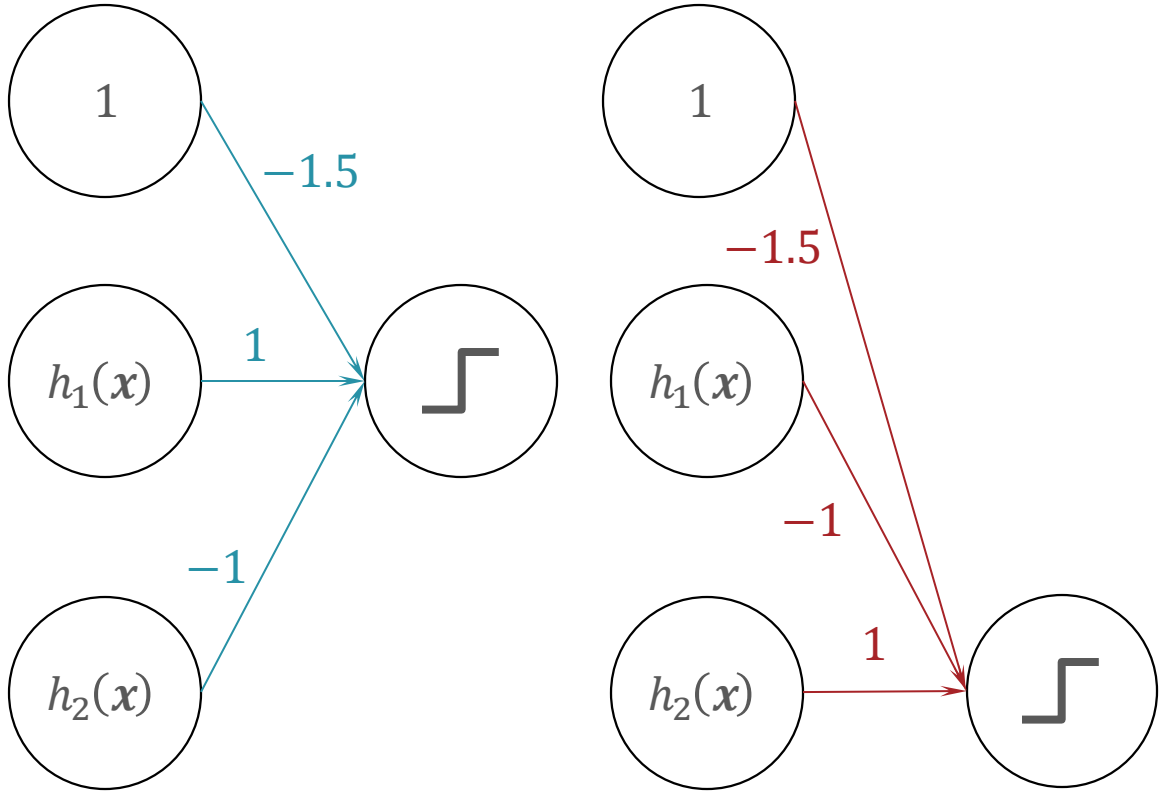
Building a Network

$$h(\boldsymbol{x}) = OR\left(AND\left(h_1(\boldsymbol{x}), \neg h_2(\boldsymbol{x})\right), AND\left(\neg h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\right)\right)$$



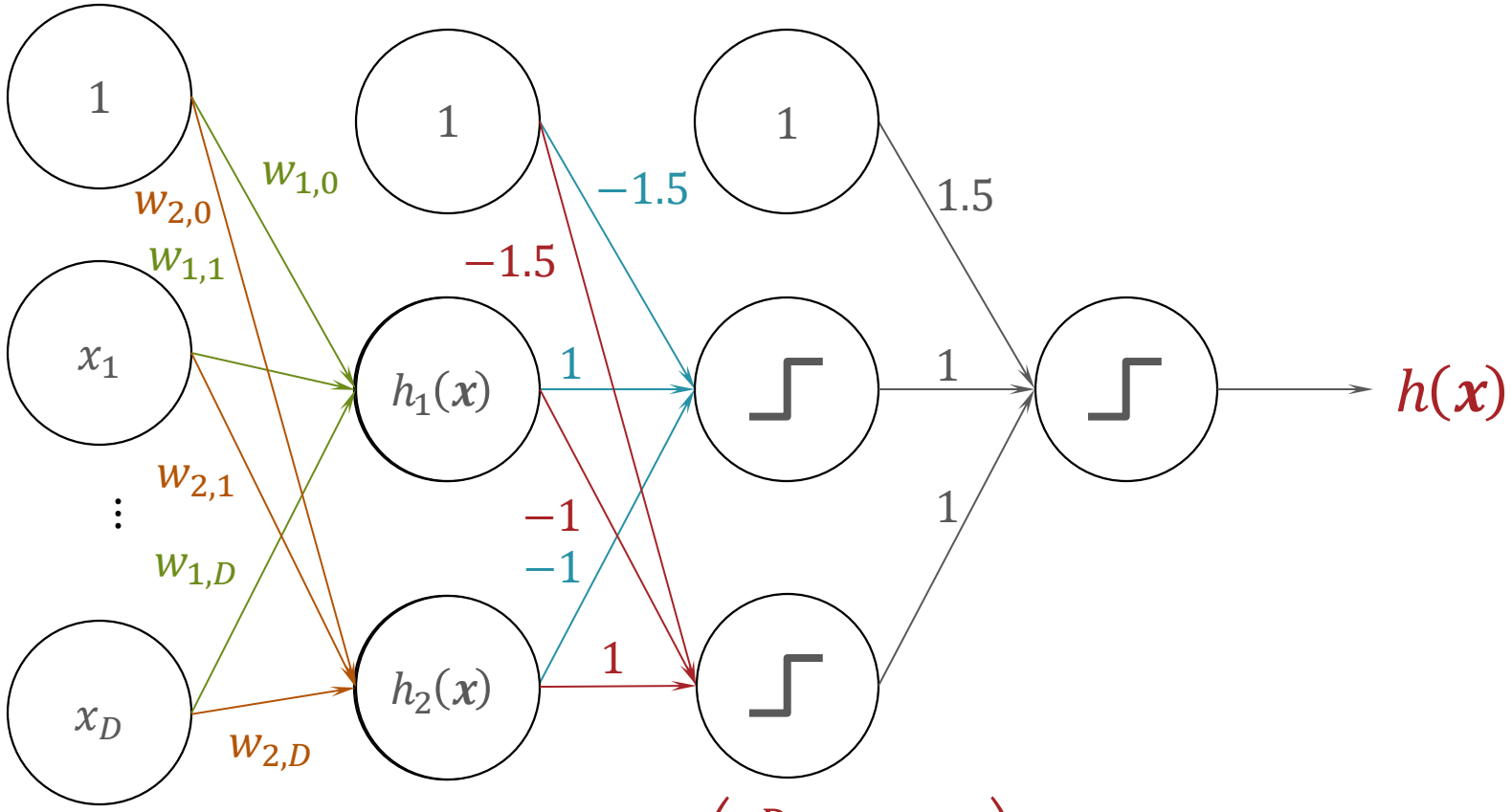
Building a Network

$$h(\boldsymbol{x}) = OR\Big(AND\big(h_1(\boldsymbol{x}), \neg h_2(\boldsymbol{x})\big), AND\big(\neg h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\big)\Big)$$



Building a Network

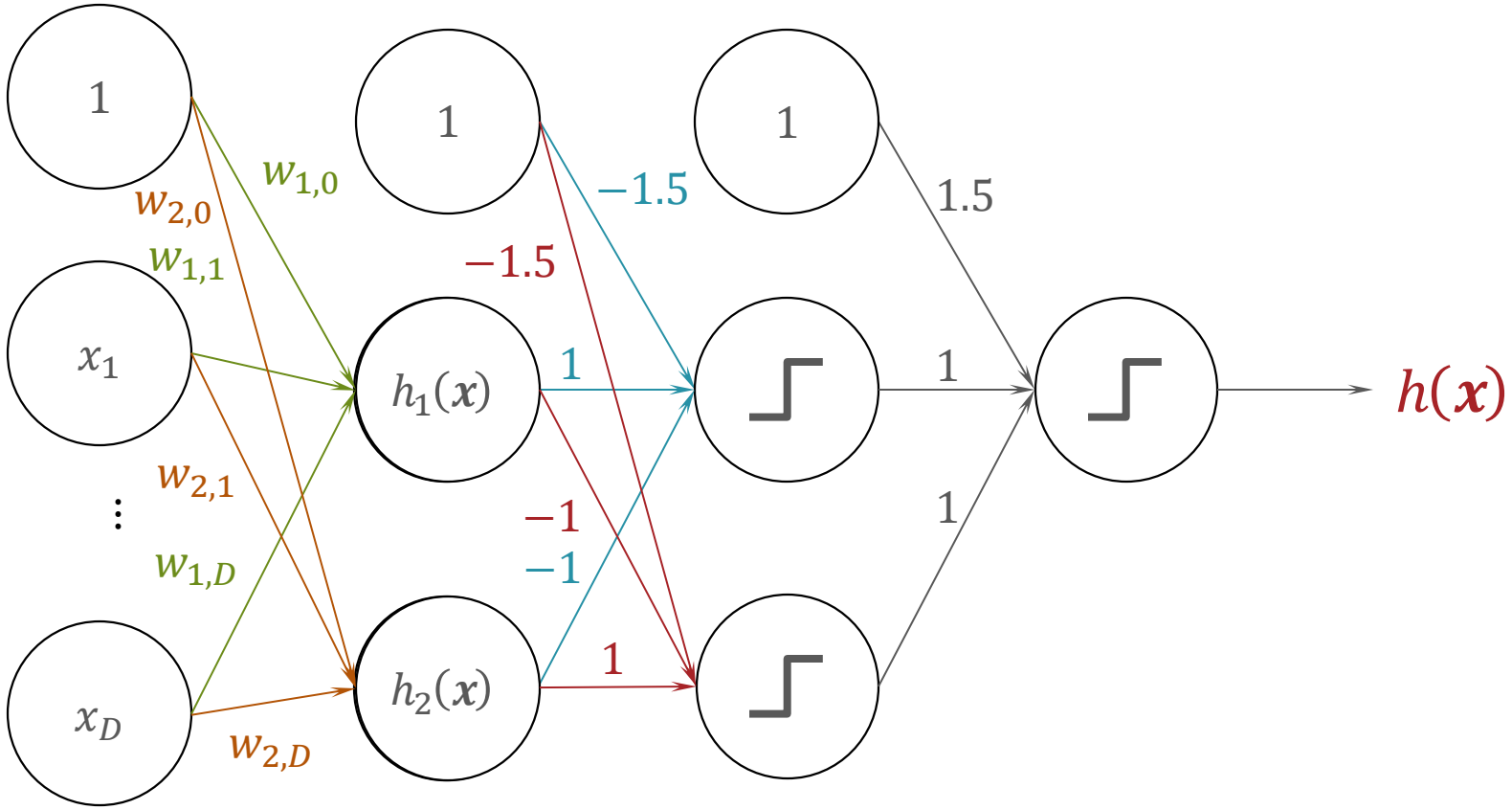
$$h(\boldsymbol{x}) = OR\Big(AND\big(h_1(\boldsymbol{x}), \neg h_2(\boldsymbol{x})\big), AND\big(\neg h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\big)\Big)$$



$$h_i(\boldsymbol{x}) = \text{sign}\big(\boldsymbol{w}_i^T \boldsymbol{x}\big) = \text{sign}\left(\sum_{d=0}^{D} w_{i,d}\, x_d\right)$$

Building a Network

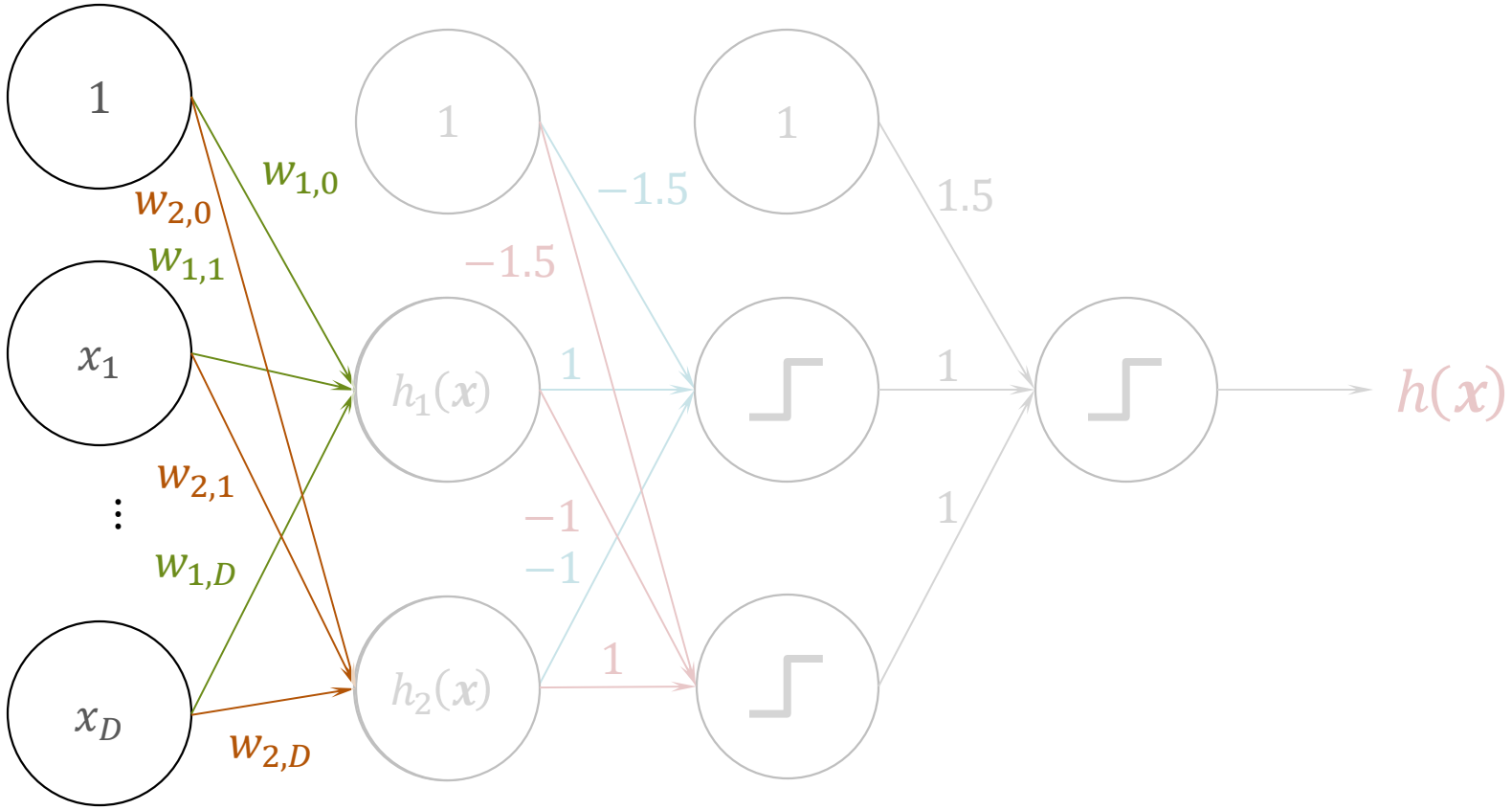
$$h(\boldsymbol{x}) = OR\Big(AND\big(h_1(\boldsymbol{x}), \neg h_2(\boldsymbol{x})\big), AND\big(\neg h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\big)\Big)$$



$$h(\boldsymbol{x}) = \text{sign}(\text{sign}(\text{sign}(\boldsymbol{w}_1^T\boldsymbol{x}) - \text{sign}(\boldsymbol{w}_2^T\boldsymbol{x}) - 1.5) + \text{sign}(-\text{sign}(\boldsymbol{w}_1^T\boldsymbol{x}) + \text{sign}(\boldsymbol{w}_2^T\boldsymbol{x}) - 1.5) + 1.5)$$

Building a Network

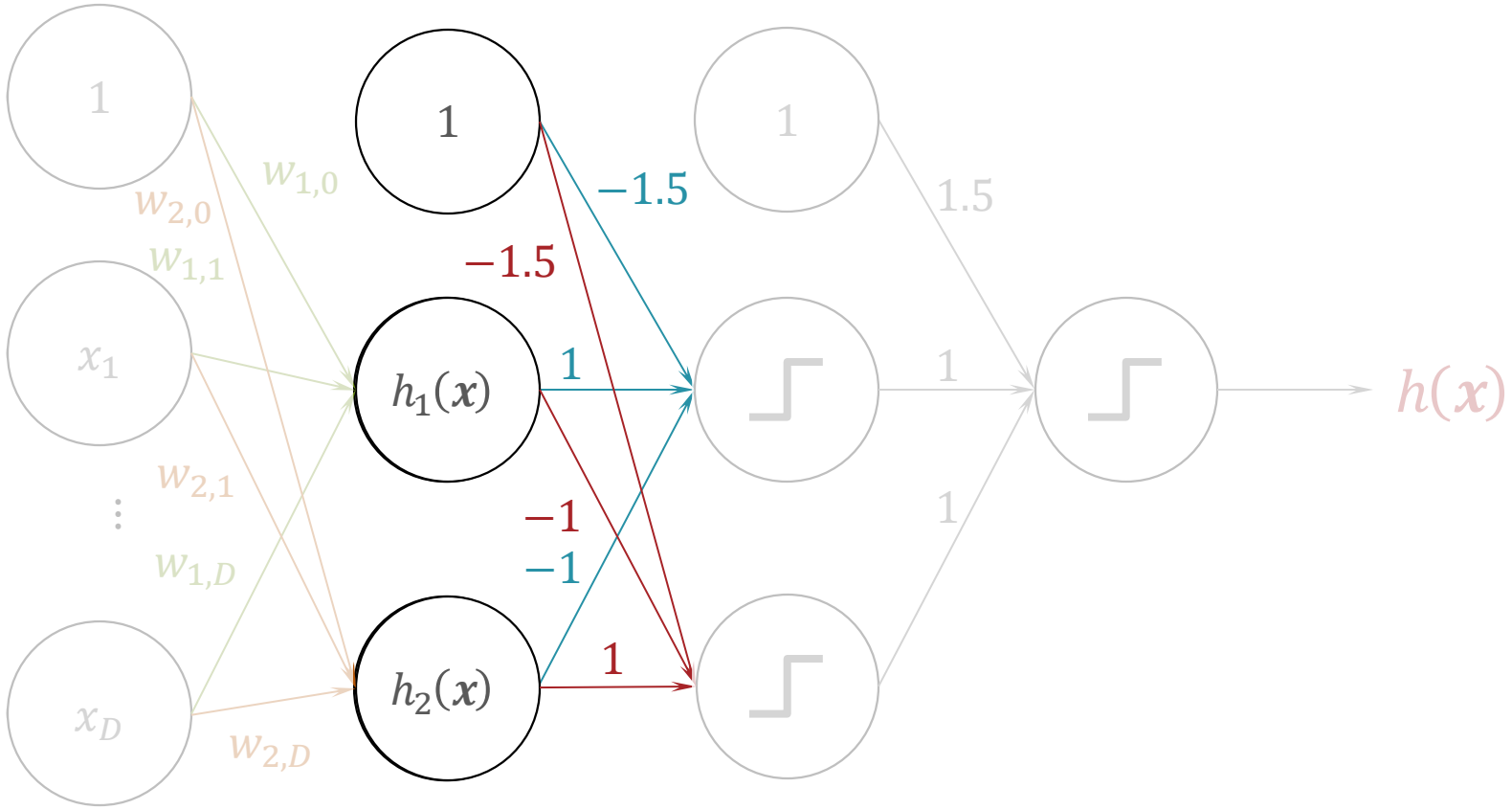
$$h(\boldsymbol{x}) = OR\left(AND\left(h_1(\boldsymbol{x}), \neg h_2(\boldsymbol{x})\right), AND\left(\neg h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\right)\right)$$



$$h(\boldsymbol{x}) = \text{sign}(\text{sign}(\text{sign}(\boldsymbol{w}_1^T\boldsymbol{x}) - \text{sign}(\boldsymbol{w}_2^T\boldsymbol{x}) - 1.5) + \text{sign}(-\text{sign}(\boldsymbol{w}_1^T\boldsymbol{x}) + \text{sign}(\boldsymbol{w}_2^T\boldsymbol{x}) - 1.5) + 1.5)$$

Building a Network

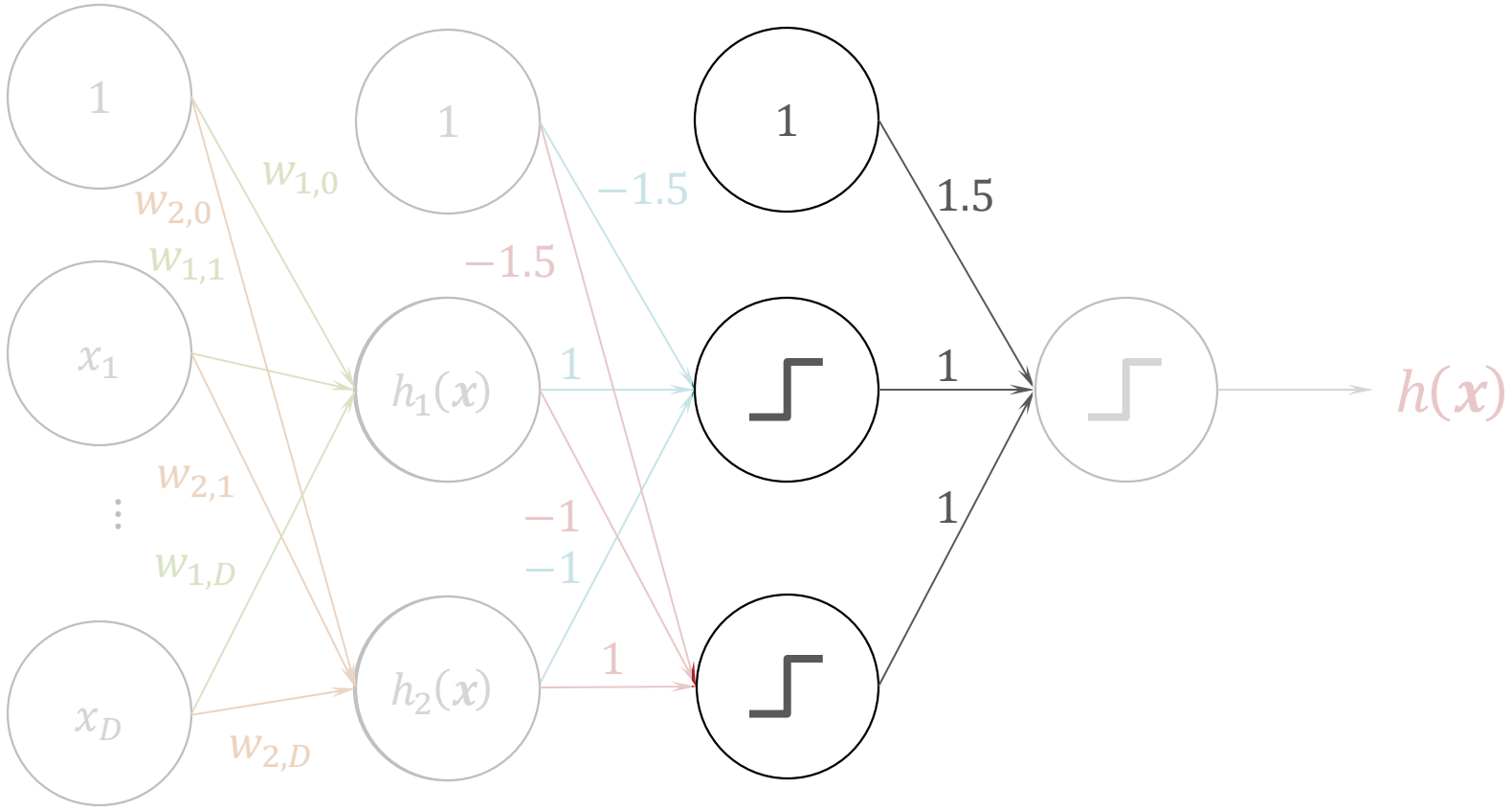
$$h(\boldsymbol{x}) = OR\left(AND\left(h_1(\boldsymbol{x}), \neg h_2(\boldsymbol{x})\right), AND\left(\neg h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\right)\right)$$



$$h(\boldsymbol{x}) = \text{sign}(\text{sign}(\text{sign}(\boldsymbol{w}_1^T\boldsymbol{x}) - \text{sign}(\boldsymbol{w}_2^T\boldsymbol{x}) - 1.5) + \text{sign}(-\text{sign}(\boldsymbol{w}_1^T\boldsymbol{x}) + \text{sign}(\boldsymbol{w}_2^T\boldsymbol{x}) - 1.5) + 1.5)$$

Building a Network

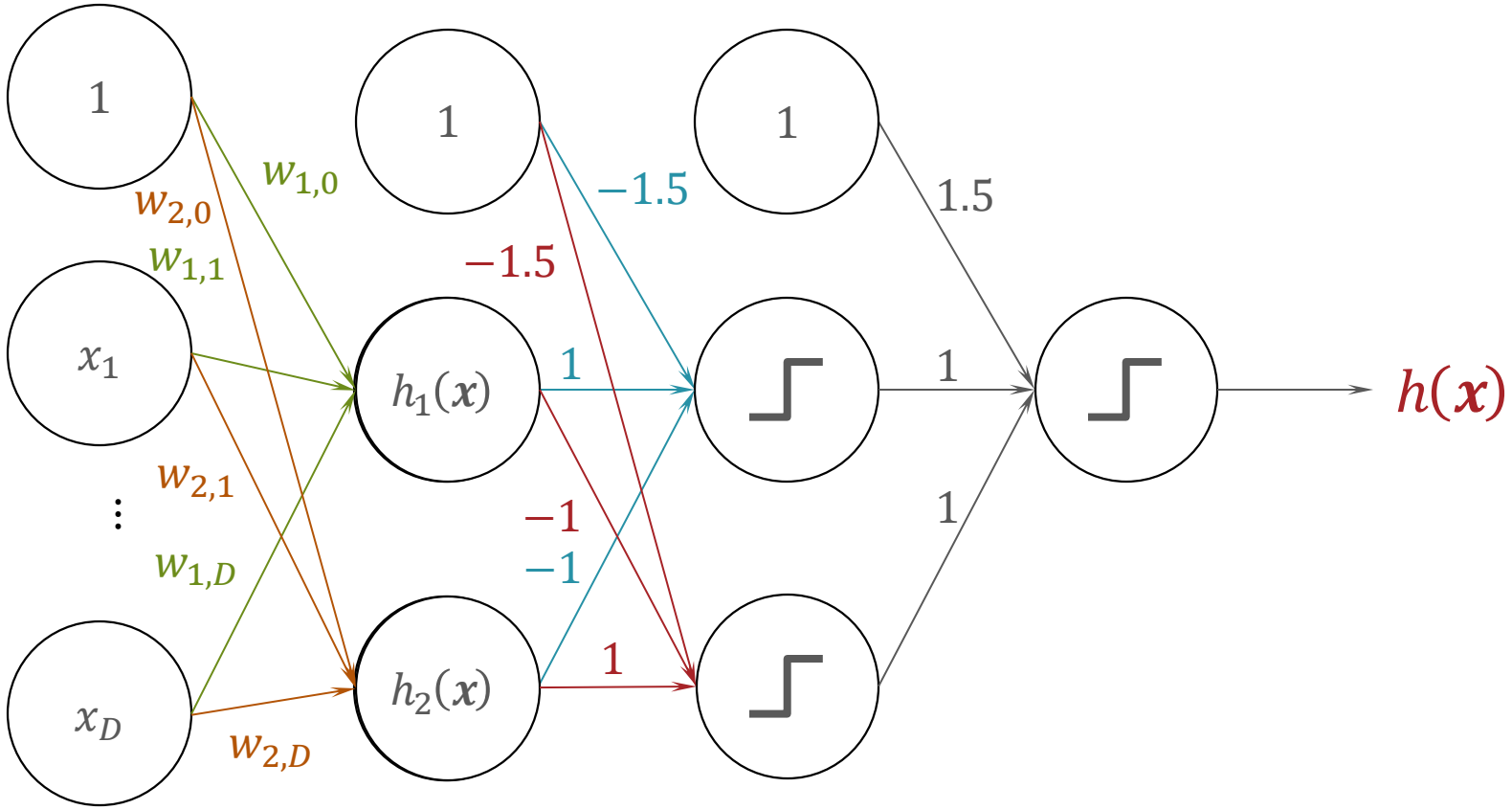
$$h(\boldsymbol{x}) = OR\left(AND\left(h_1(\boldsymbol{x}), \neg h_2(\boldsymbol{x})\right), AND\left(\neg h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\right)\right)$$



$$h(\boldsymbol{x}) = \text{sign}(\text{sign}(\text{sign}(\boldsymbol{w}_1^T\boldsymbol{x}) - \text{sign}(\boldsymbol{w}_2^T\boldsymbol{x}) - 1.5) + \text{sign}(-\text{sign}(\boldsymbol{w}_1^T\boldsymbol{x}) + \text{sign}(\boldsymbol{w}_2^T\boldsymbol{x}) - 1.5) + 1.5)$$

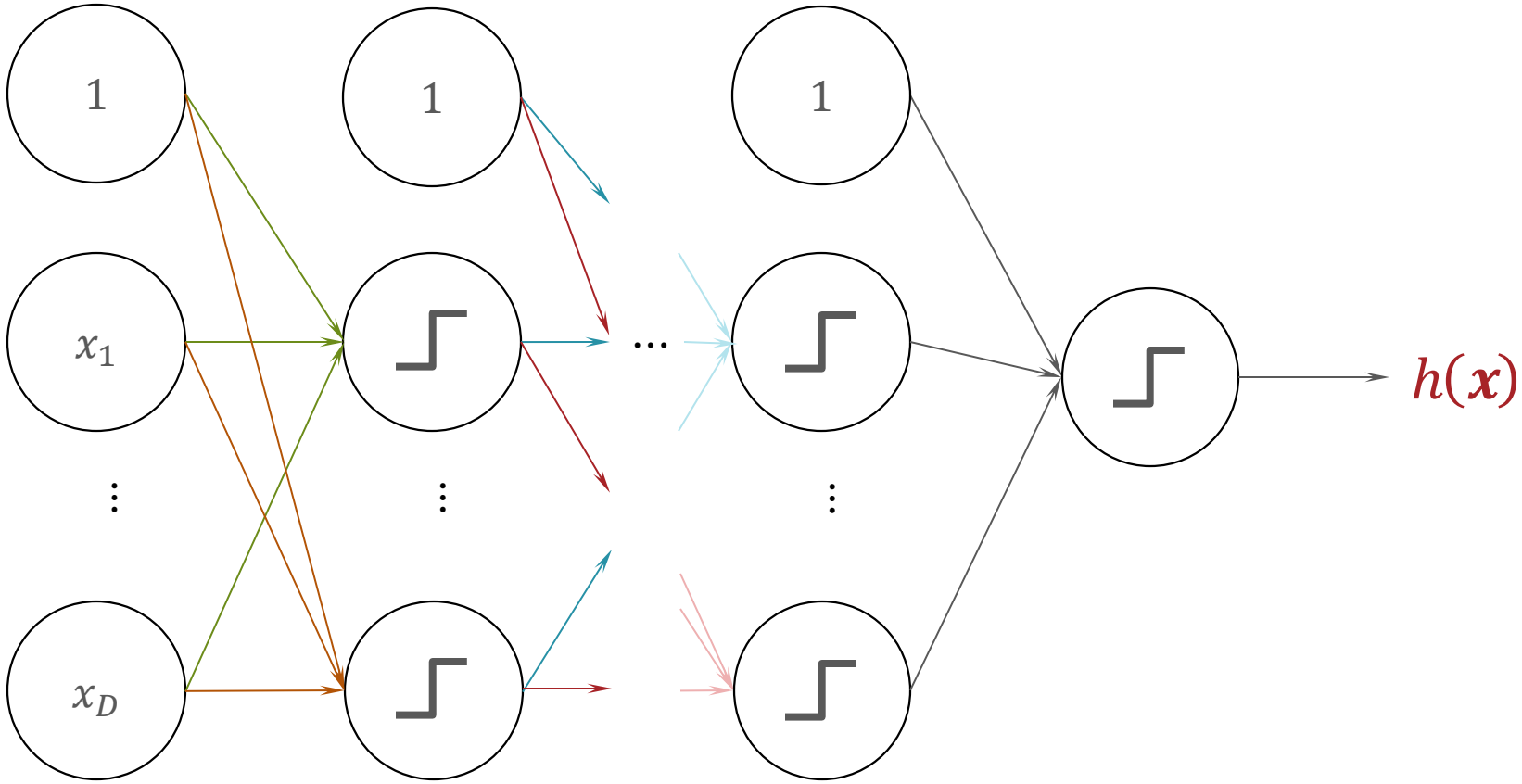
Building a Network

$$h(\boldsymbol{x}) = OR\Big(AND\big(h_1(\boldsymbol{x}), \neg h_2(\boldsymbol{x})\big), AND\big(\neg h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\big)\Big)$$

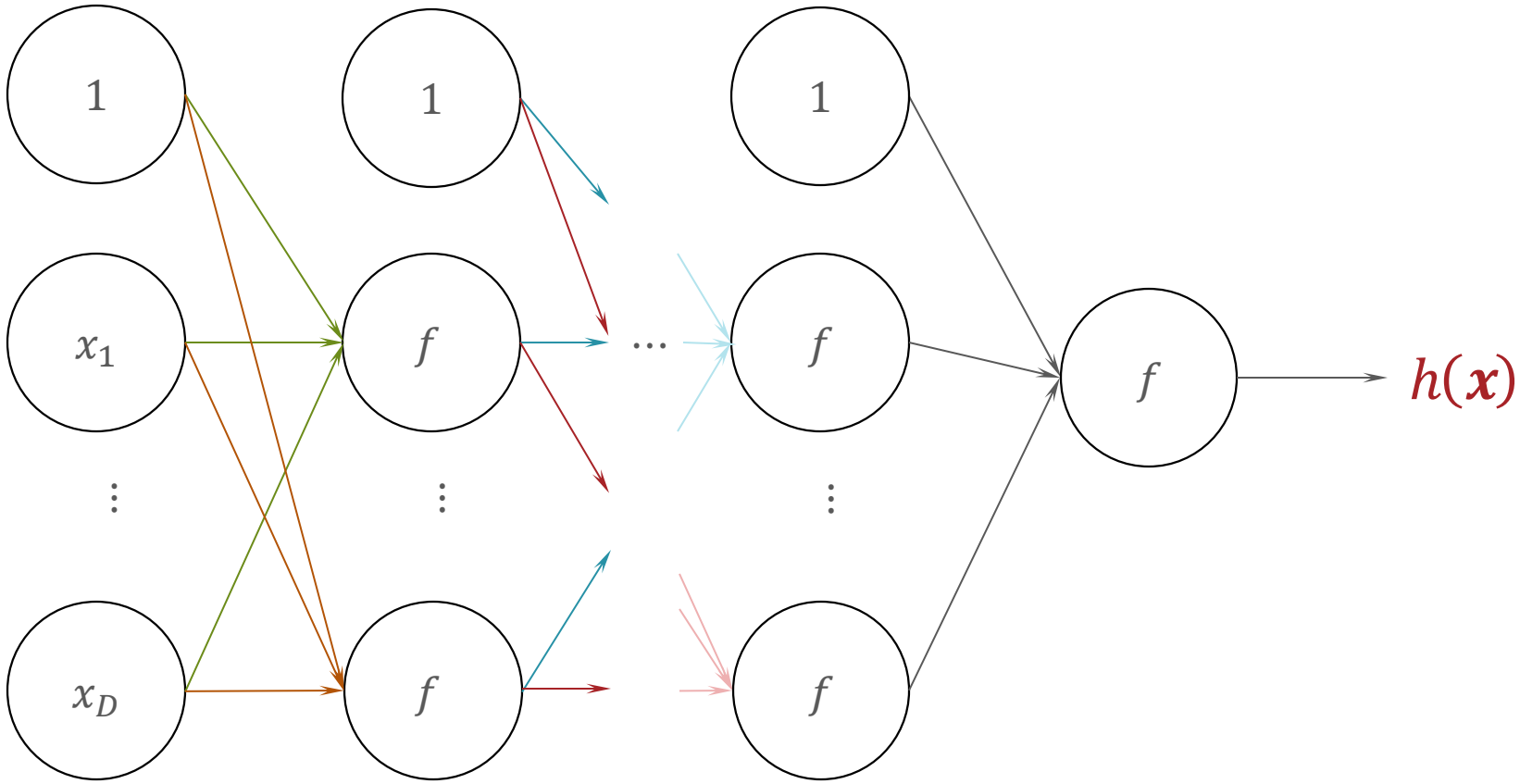


$$h(\boldsymbol{x}) = \text{sign}(\text{sign}(\text{sign}(\boldsymbol{w}_1^T\boldsymbol{x}) - \text{sign}(\boldsymbol{w}_2^T\boldsymbol{x}) - 1.5) + \text{sign}(-\text{sign}(\boldsymbol{w}_1^T\boldsymbol{x}) + \text{sign}(\boldsymbol{w}_2^T\boldsymbol{x}) - 1.5) + 1.5)$$

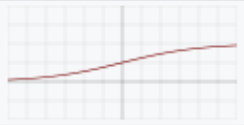
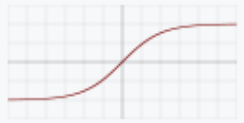
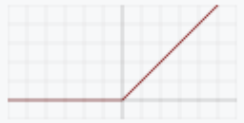
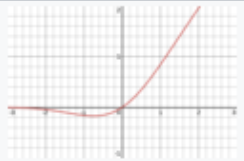
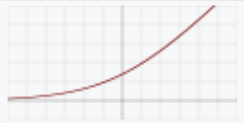
Multi-Layer Perceptron (MLP)



(Fully-Connected) Feed Forward Neural Network



Activation Functions

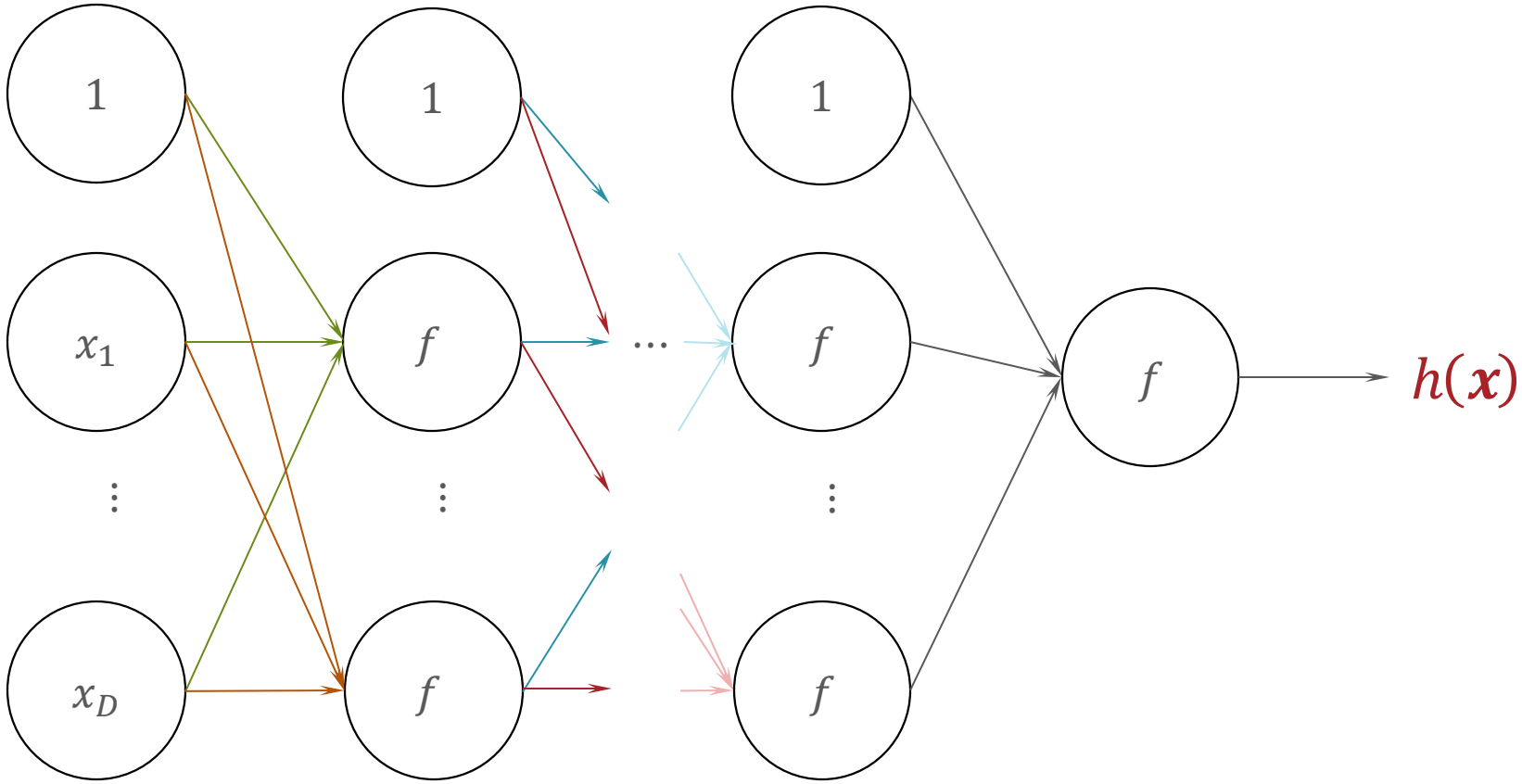
Logistic, sigmoid, or soft step		$\sigma(x) = \frac{1}{1+e^{-x}}$
Hyperbolic tangent (tanh)		$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Rectified linear unit (ReLU)[7]		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ $= \max\{0, x\} = x\mathbf{1}_{x>0}$
Gaussian Error Linear Unit (GELU)[4]		$\frac{1}{2}x\left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$ $= x\Phi(x)$
Softplus[8]		$\ln(1 + e^x)$
Exponential linear unit (ELU)[9]		$\begin{cases} \alpha\left(e^x - 1\right) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ with parameter α
Leaky rectified linear unit (Leaky ReLU)[11]		$\begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$
Parametric rectified linear unit (PReLU)[12]		$\begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ with parameter α

Source: https://en.wikipedia.org/wiki/Activation_function

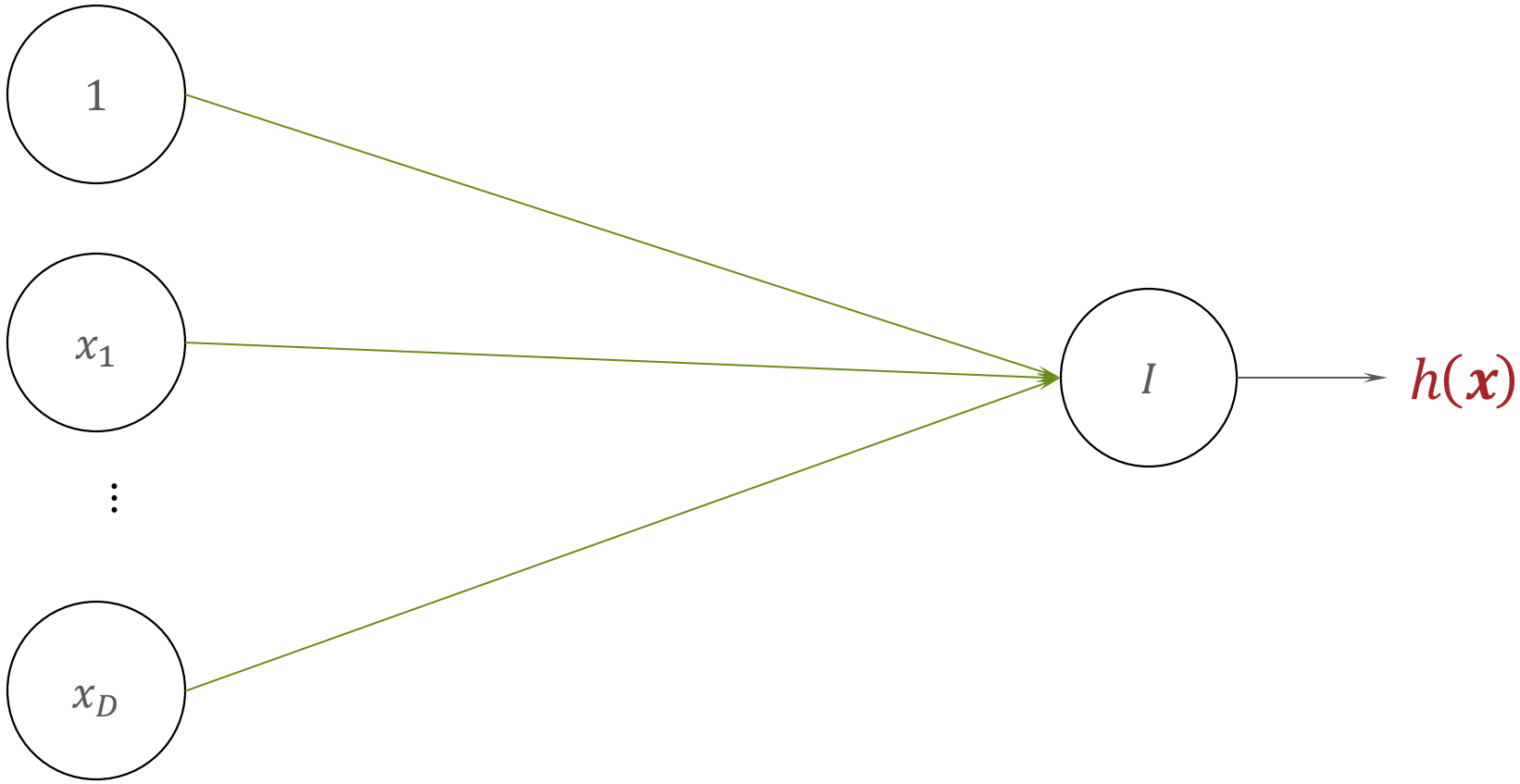
Poll Question 1

True or False: Linear and logistic regression models can be expressed as neural networks.

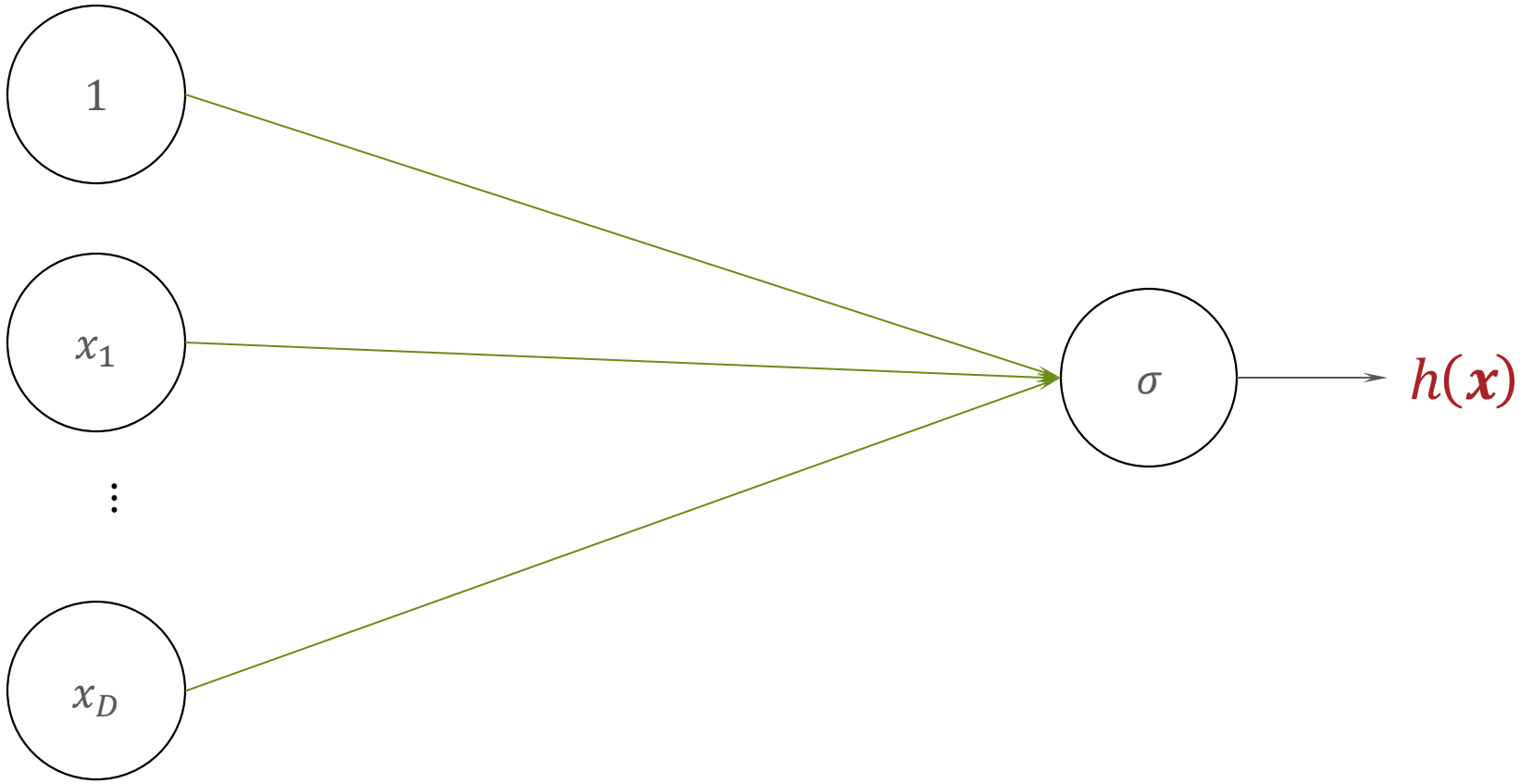
A. Only true for linear regression
B. Only true for logistic regression
C. TOXIC
D. True for both
E. False for both



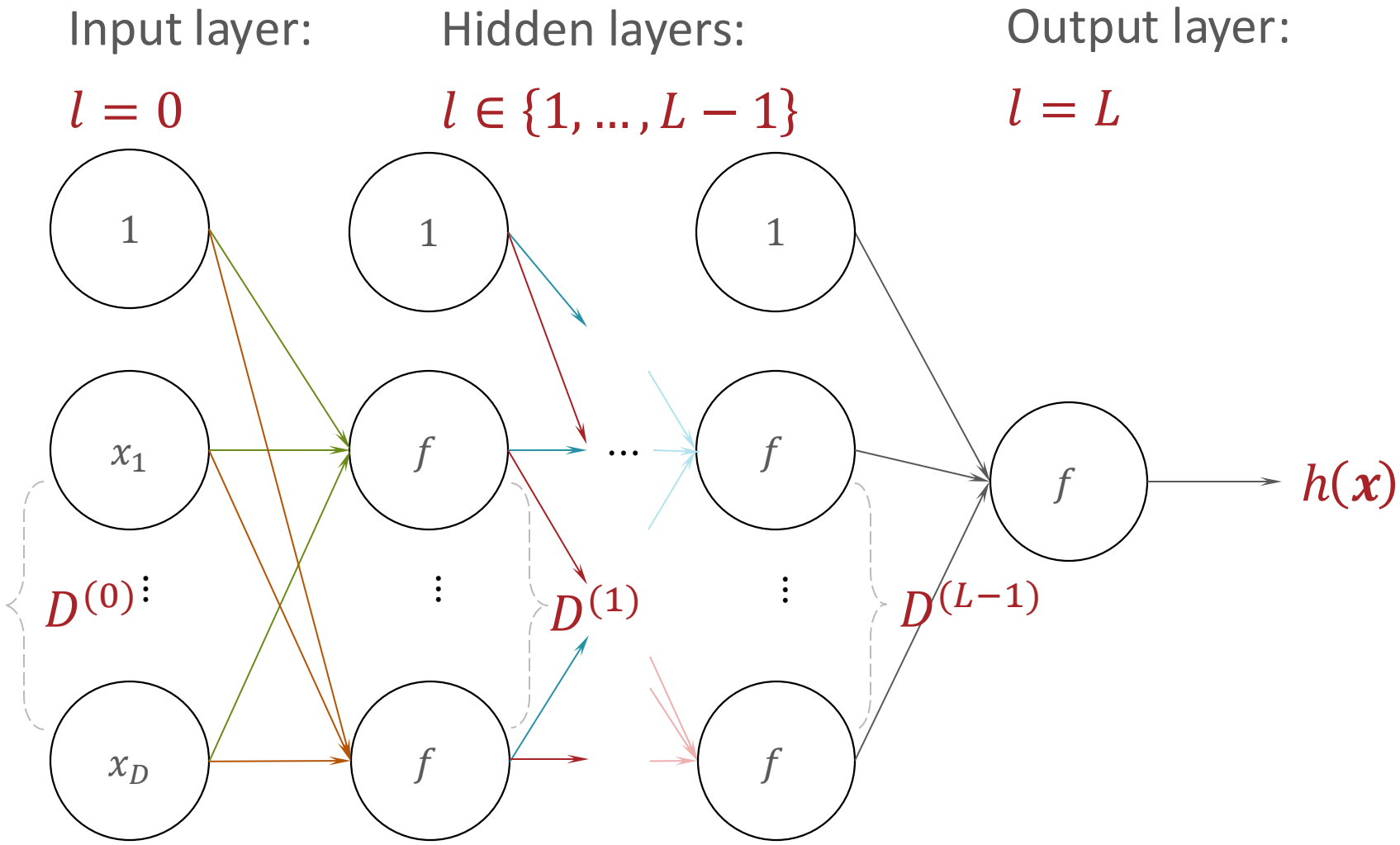
Linear Regression as a Neural Network



Logistic Regression as a Neural Network



(Fully-Connected) Feed Forward Neural Network

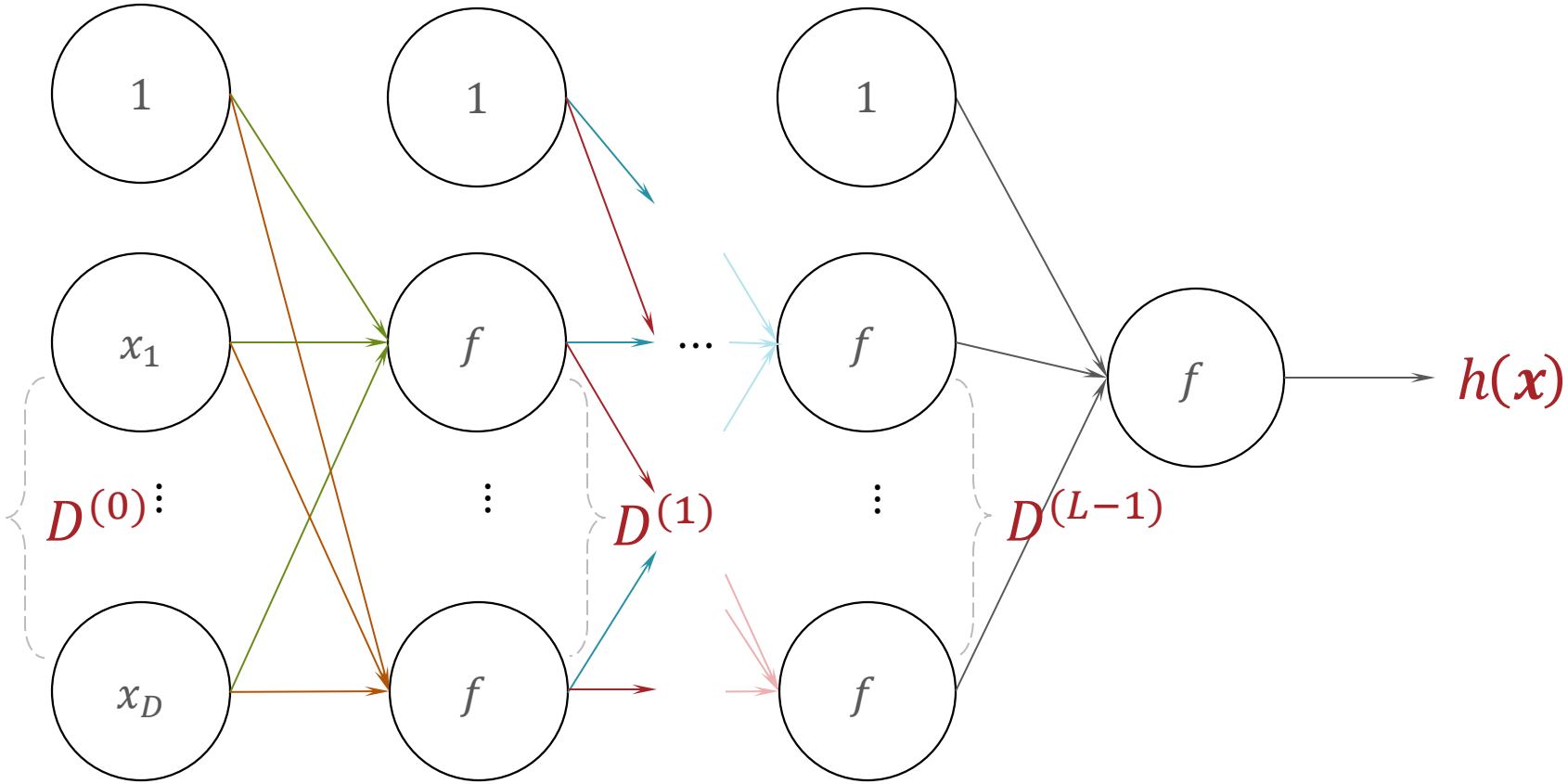


Layer l has dimension $D^{(l)}$ → Layer l has $D^{(l)} + 1$ nodes, counting the bias node

(Fully-Connected) Feed Forward Neural Network

The weights between layer $l-1$ and layer l are a matrix:

$$W^{(l)} \in \mathbb{R}^{D^{(l)} \times \left(D^{(l-1)}+1\right)}$$

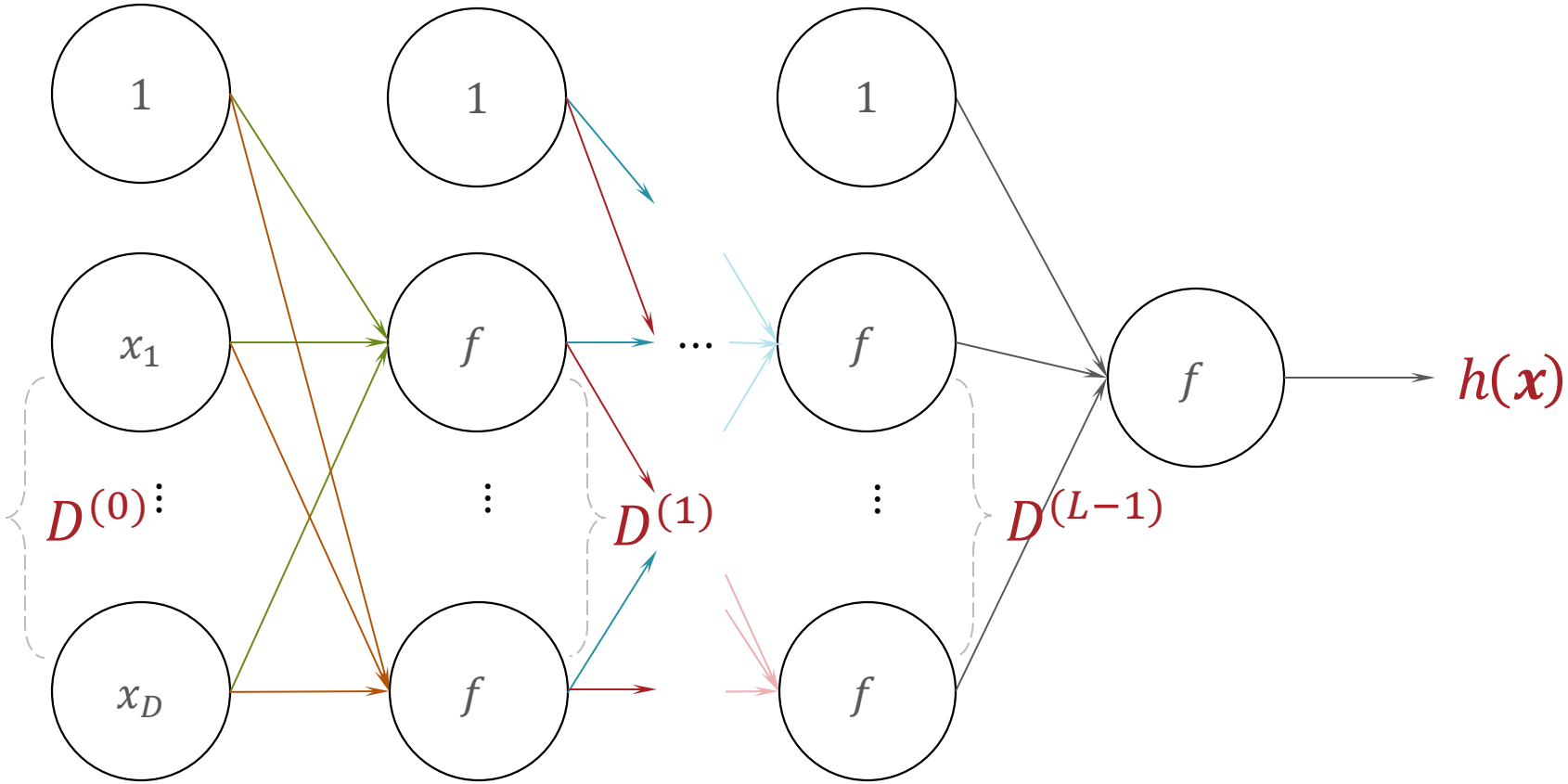


$w_{j,i}^{(l)}$ is the weight between node i in layer $l-1$ and node j in layer l

So what are all these layers doing for us anyway?

The weights between layer $l-1$ and layer l are a matrix:

$$W^{(l)} \in \mathbb{R}^{D^{(l)} \times \left(D^{(l-1)}+1\right)}$$



$w_{j,i}^{(l)}$ is the weight between node i in layer $l-1$ and node j in layer l

Neural Network Decision Boundaries: Example 1

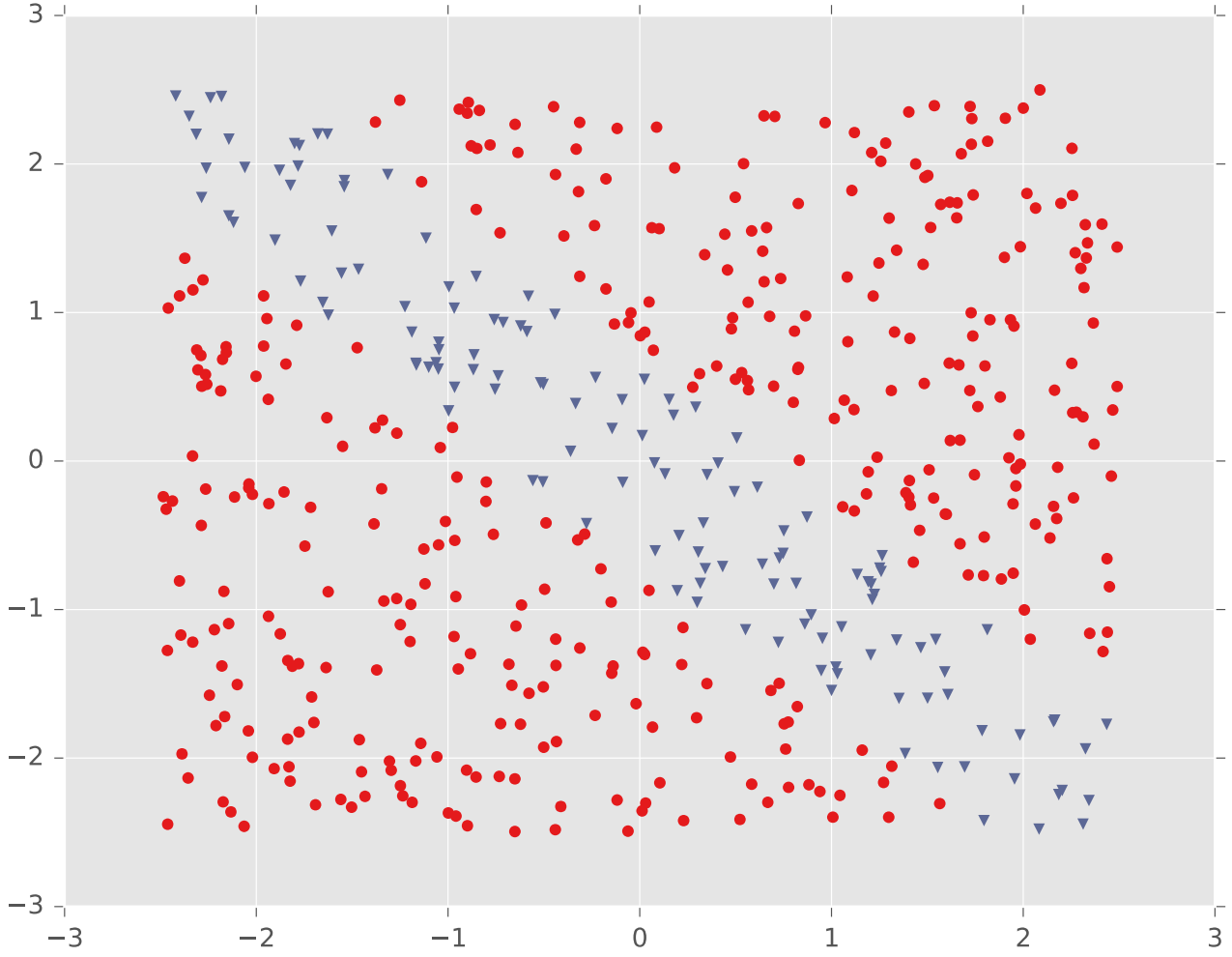


Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 1

Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 1

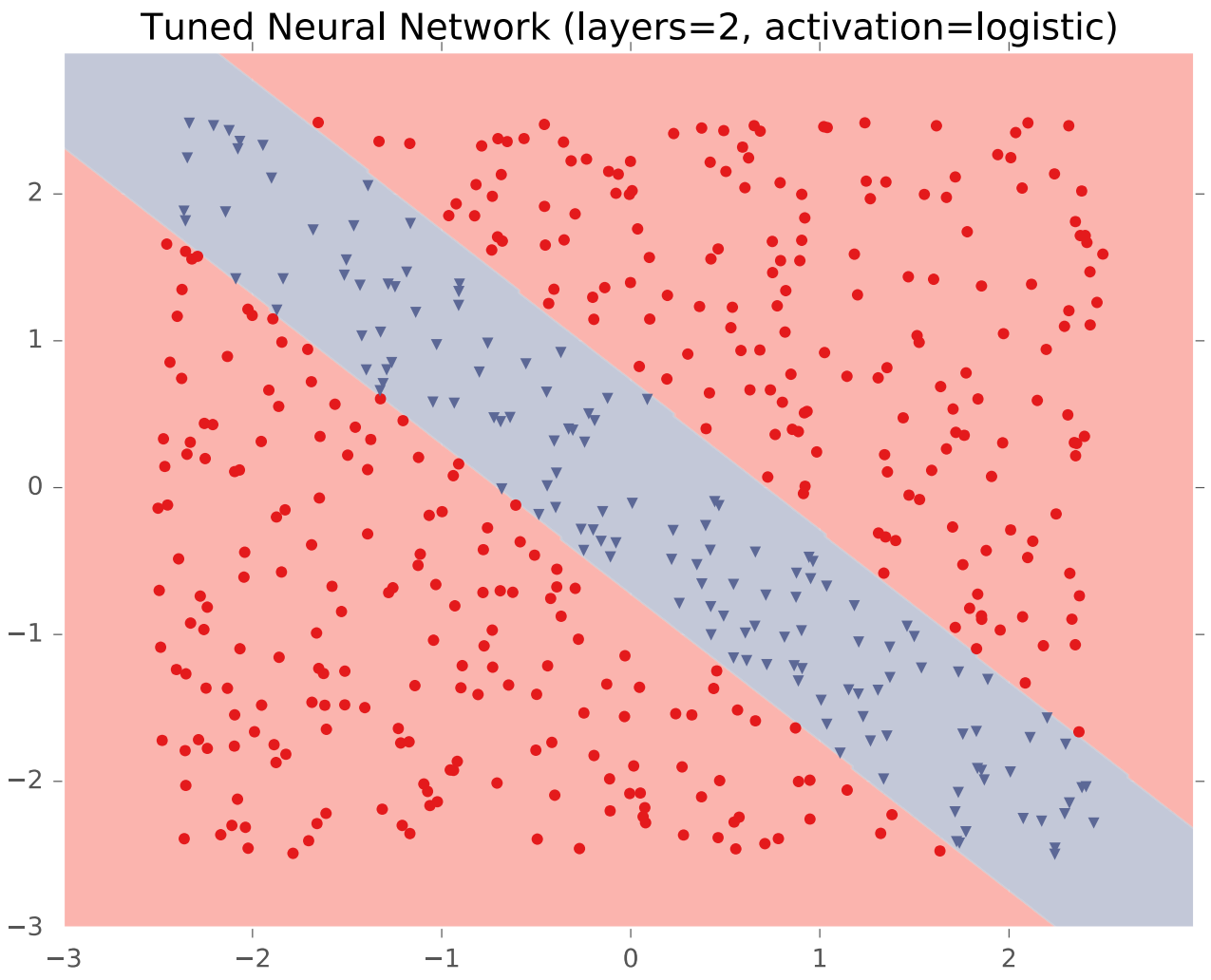


Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 1

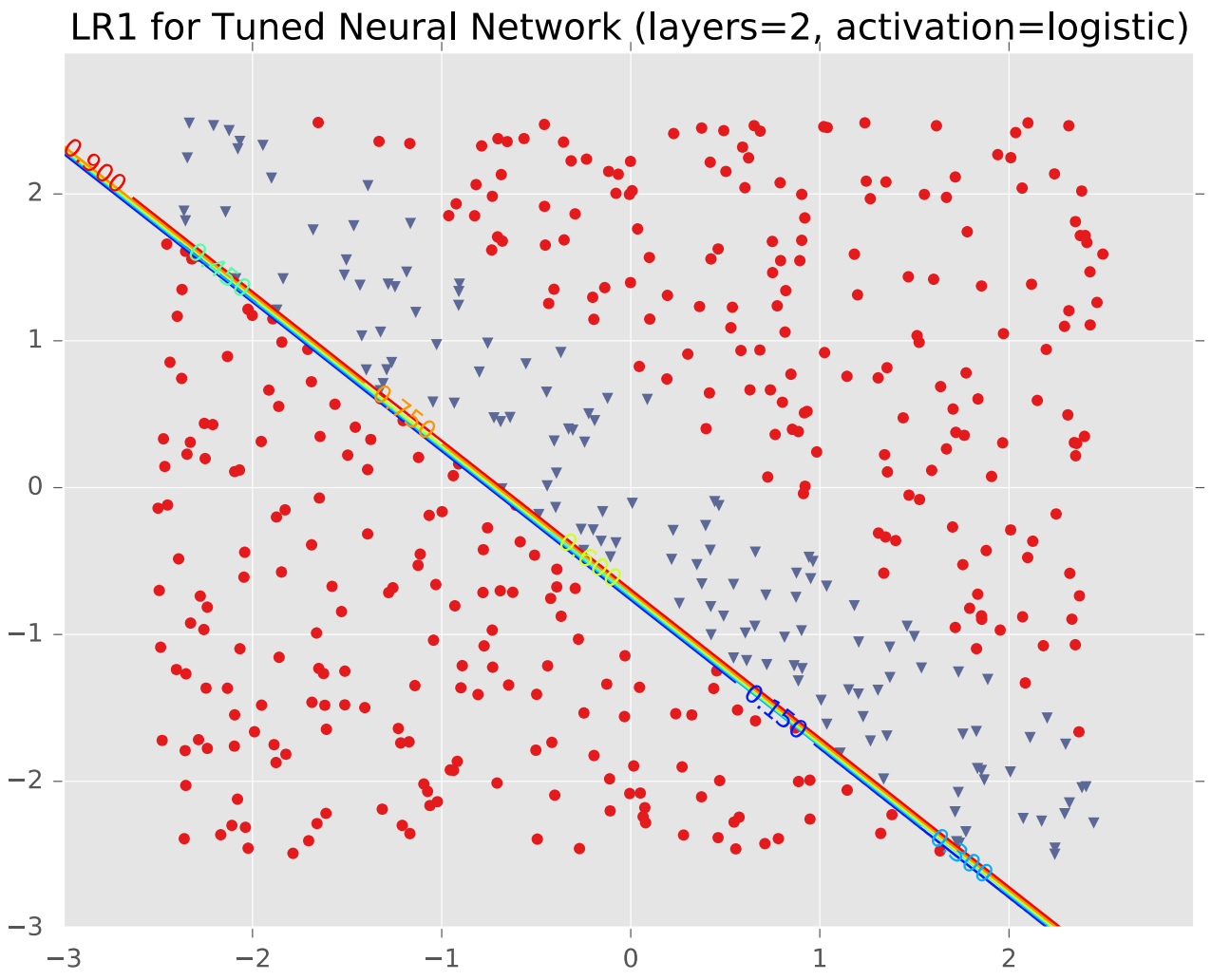


Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 1

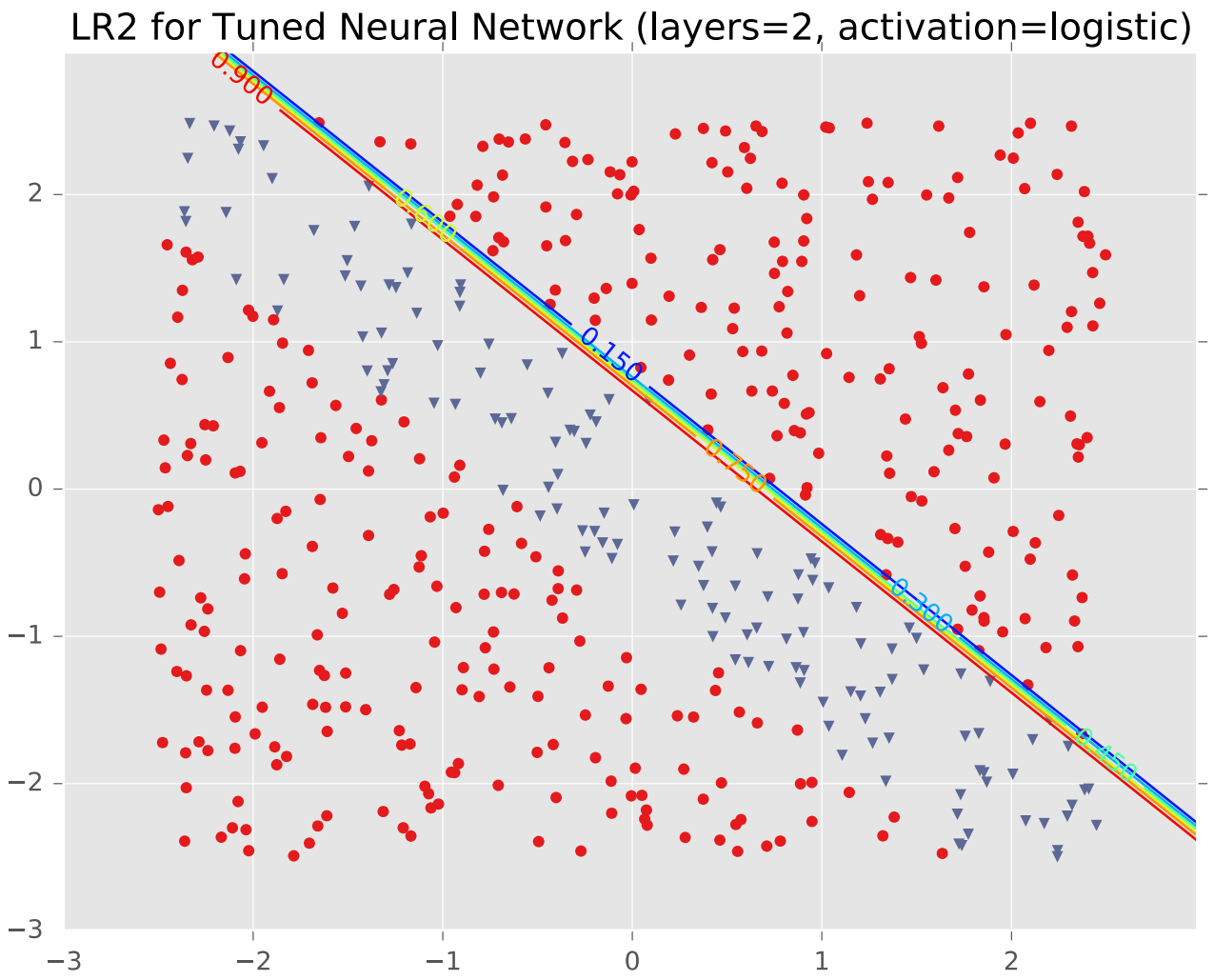


Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 1

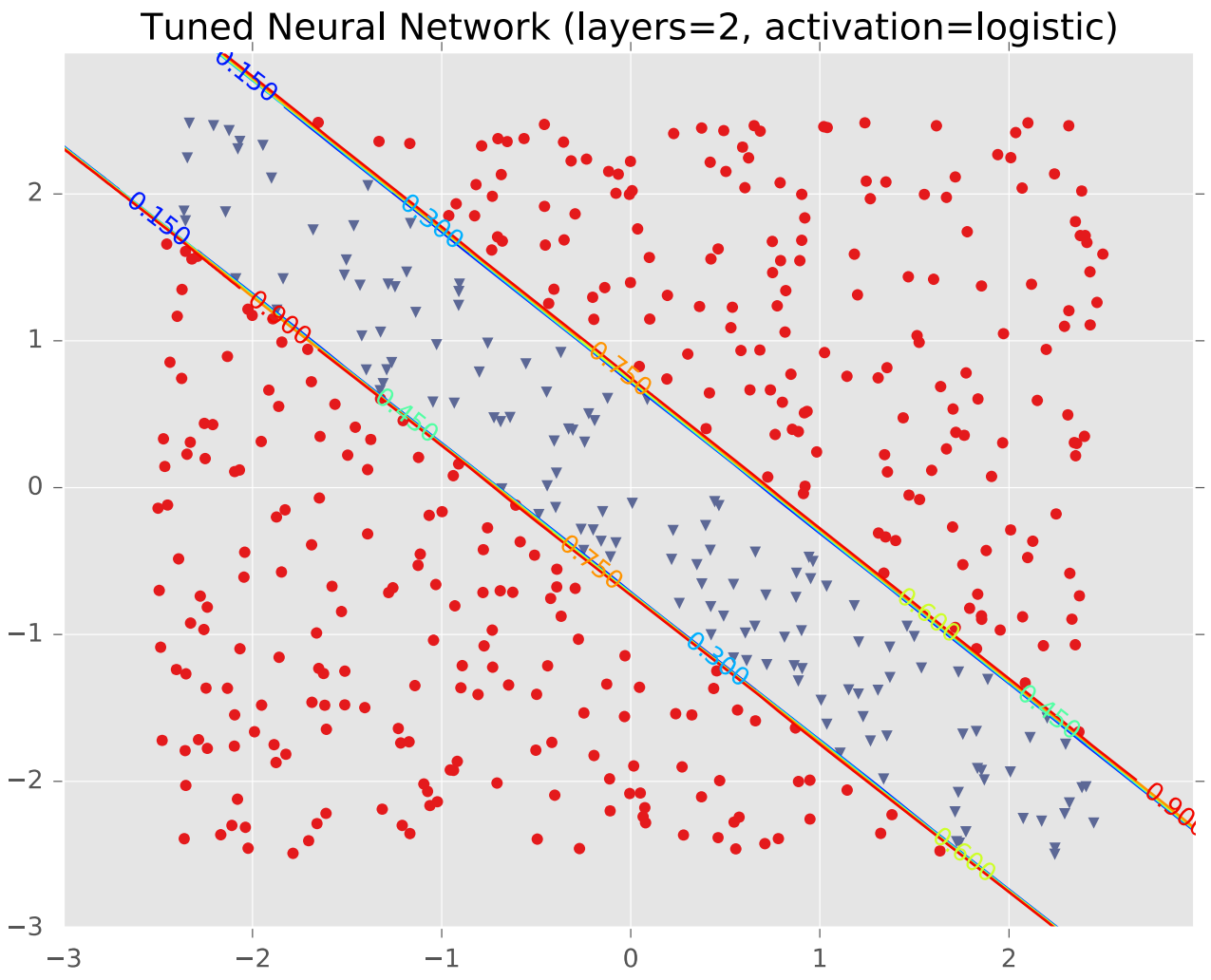


Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 1

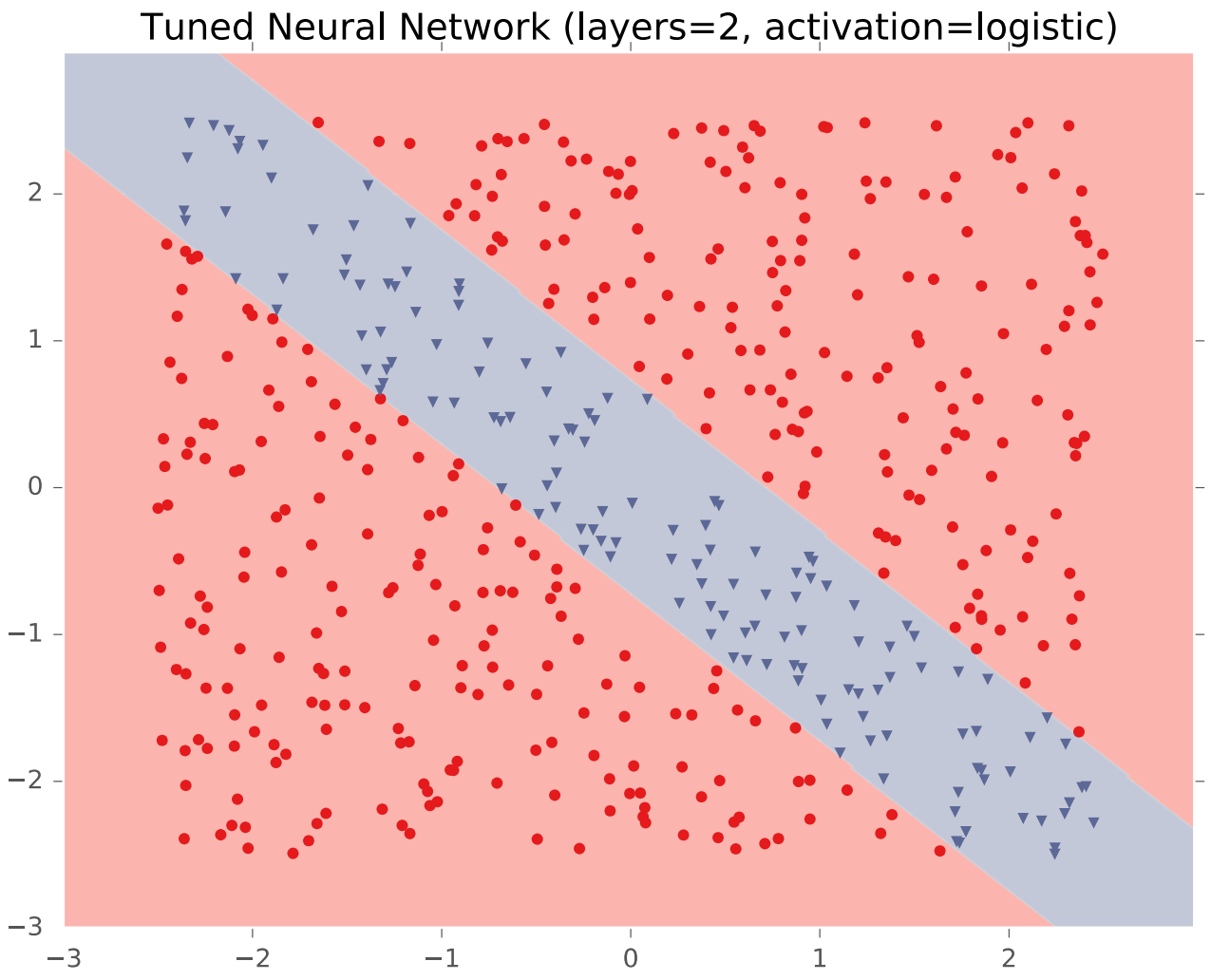


Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 2

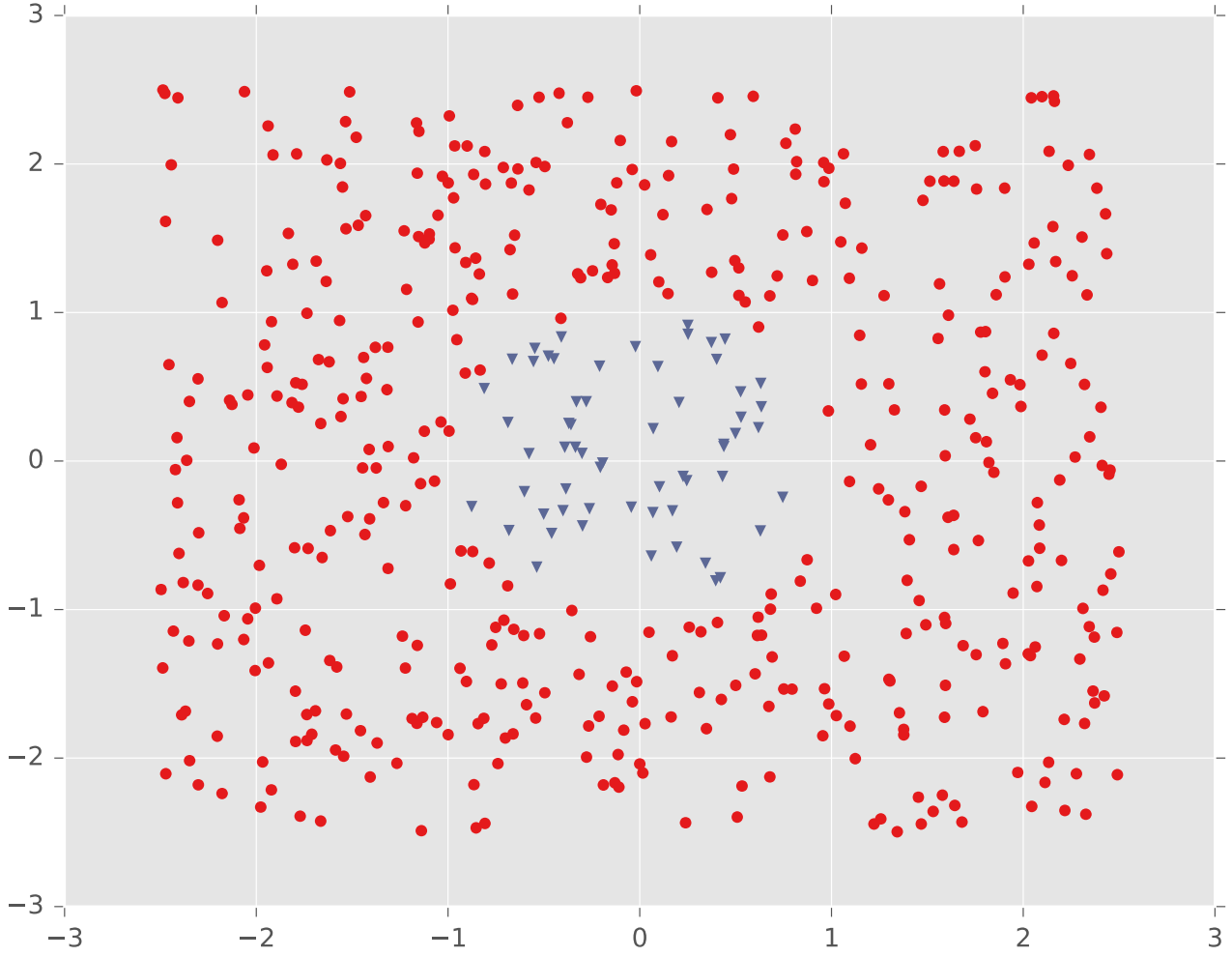


Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 2

Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 2

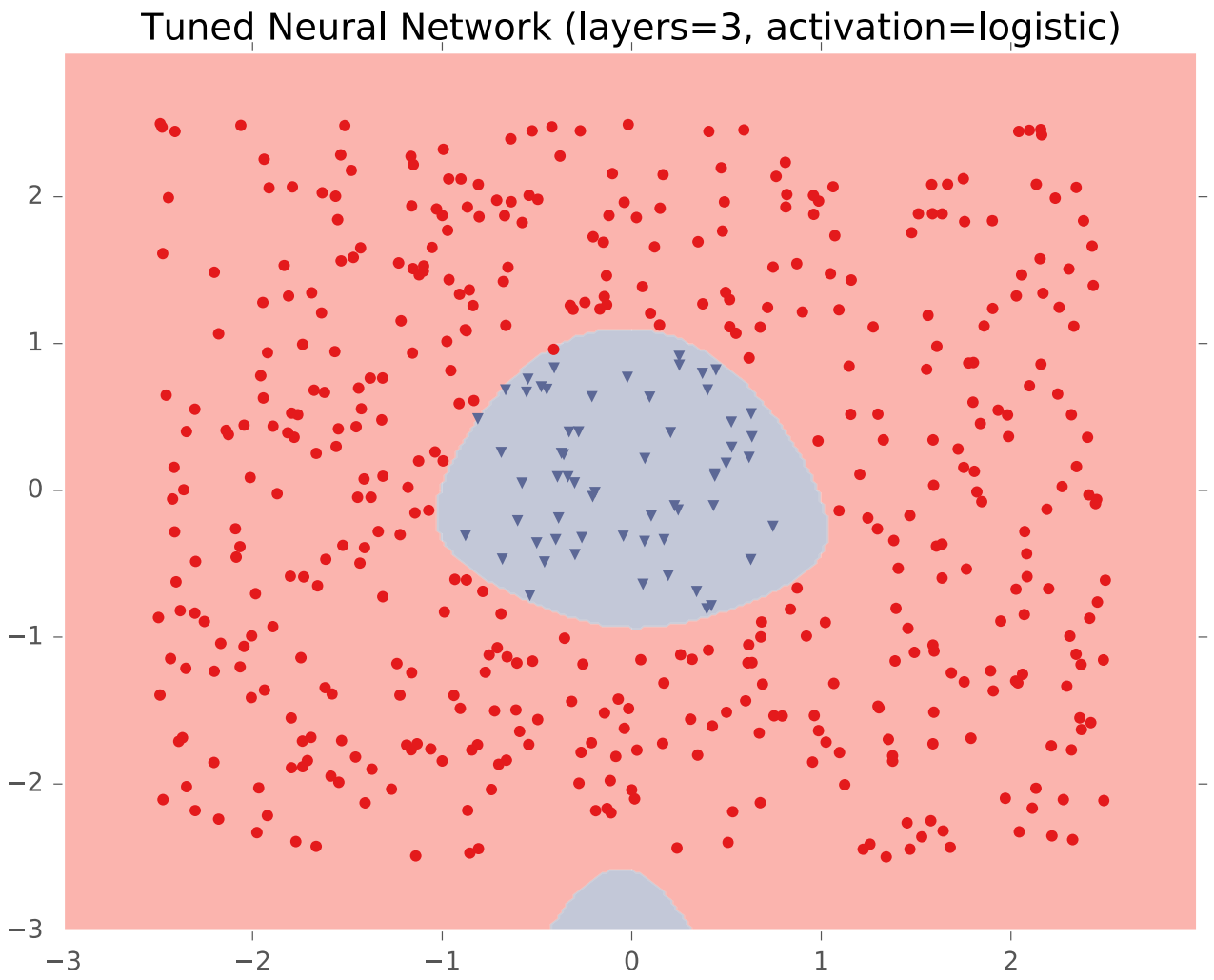


Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 2

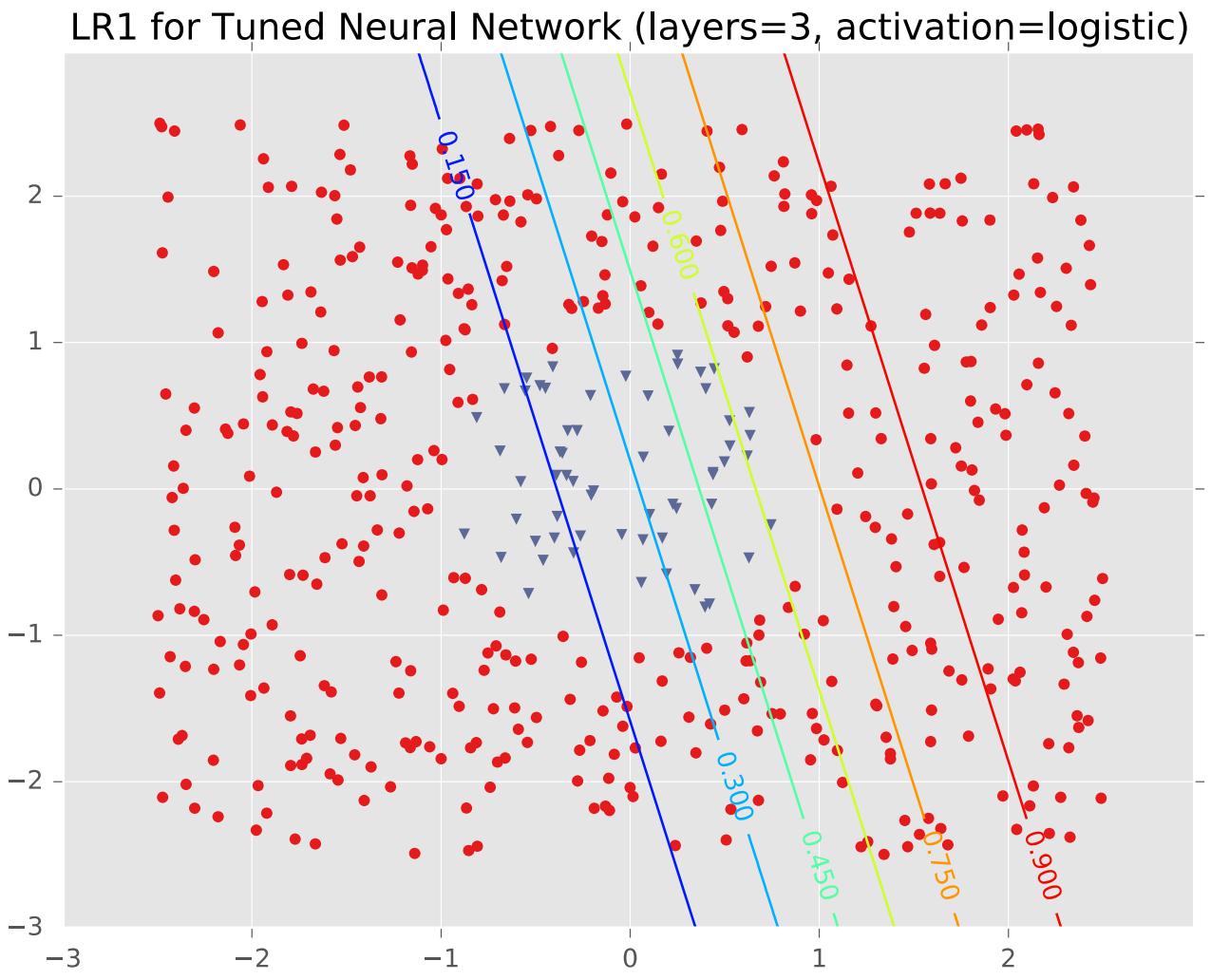


Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 2

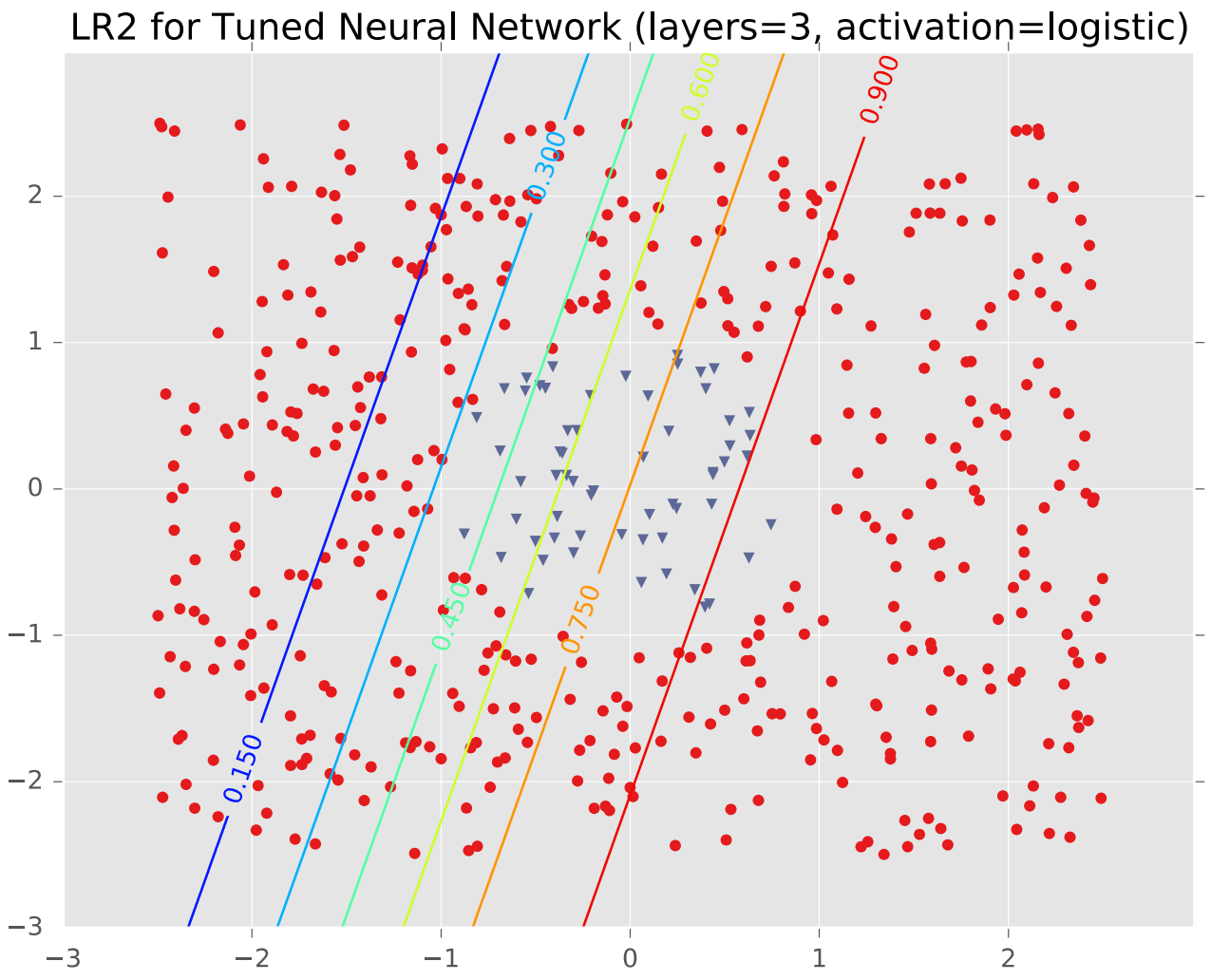


Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 2

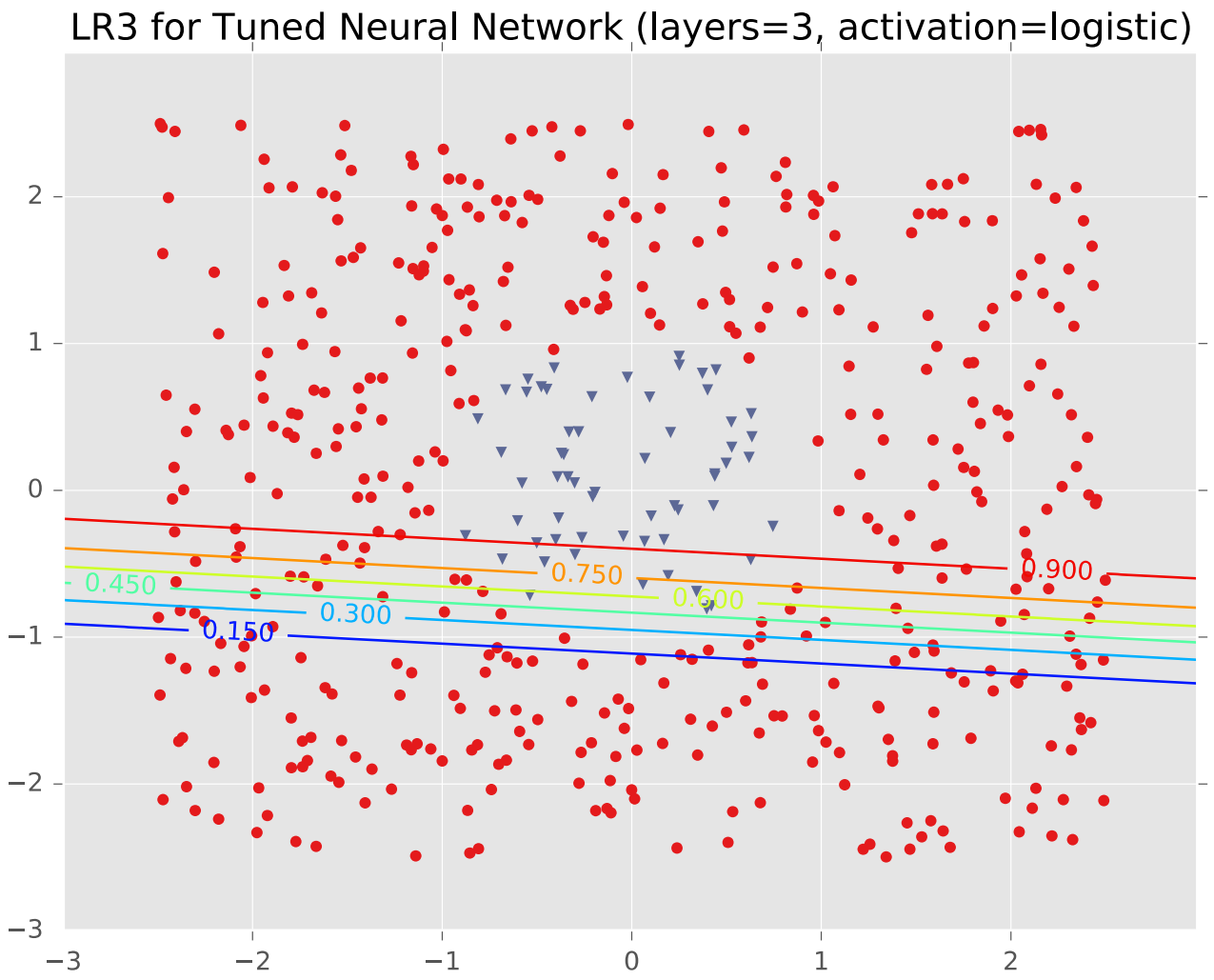


Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 2

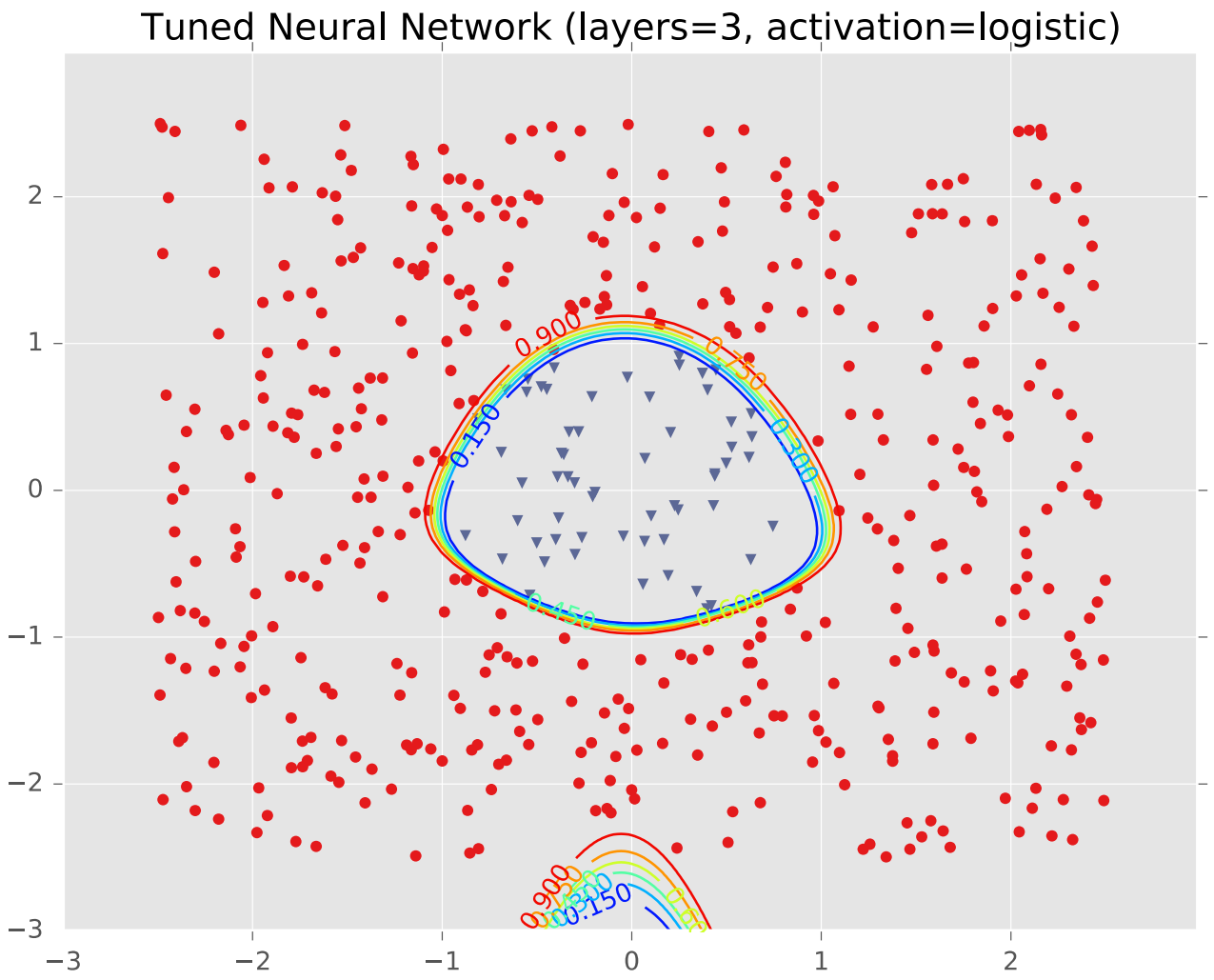


Figure courtesy of Matt Gormley

Neural Network Decision Boundaries: Example 2

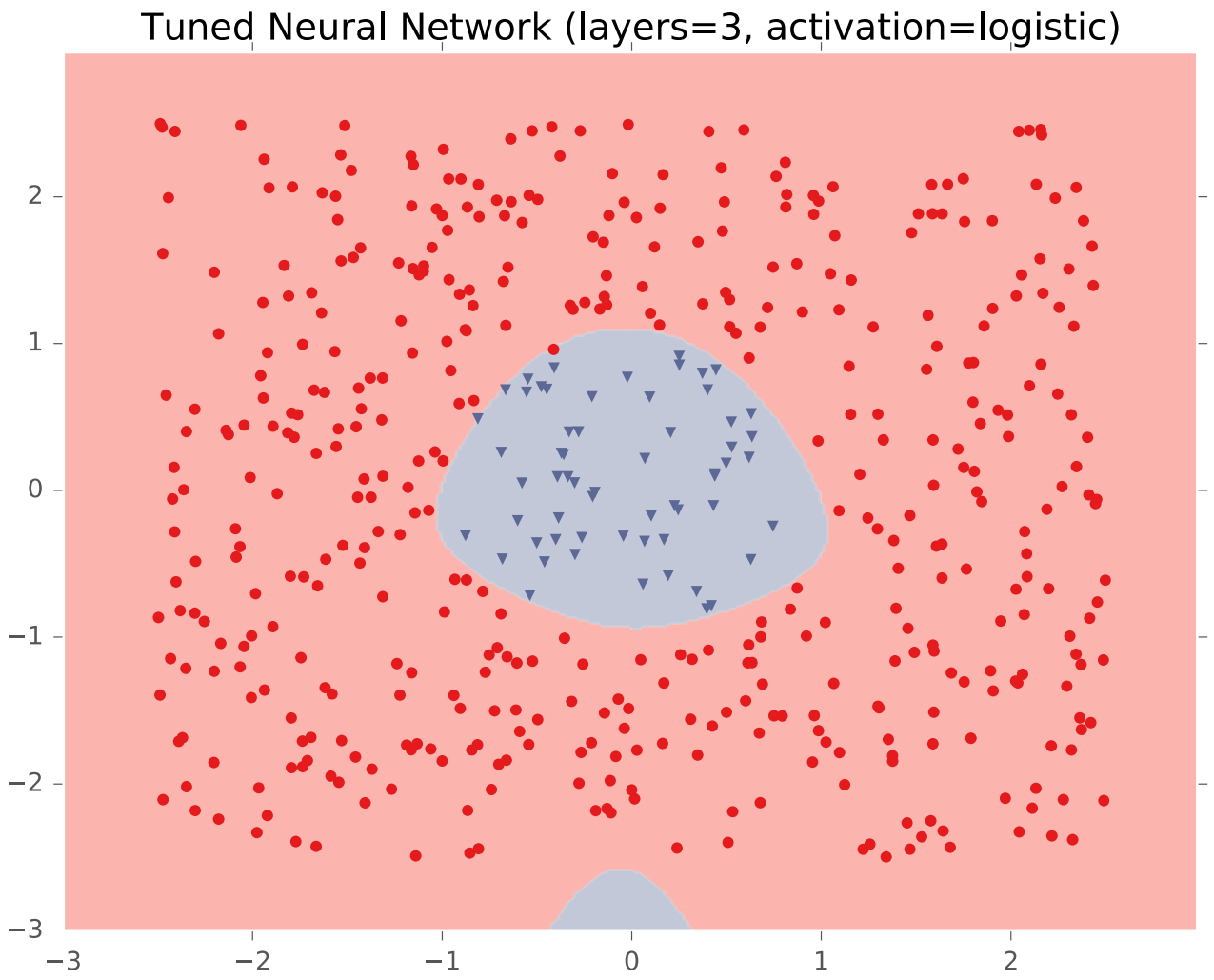
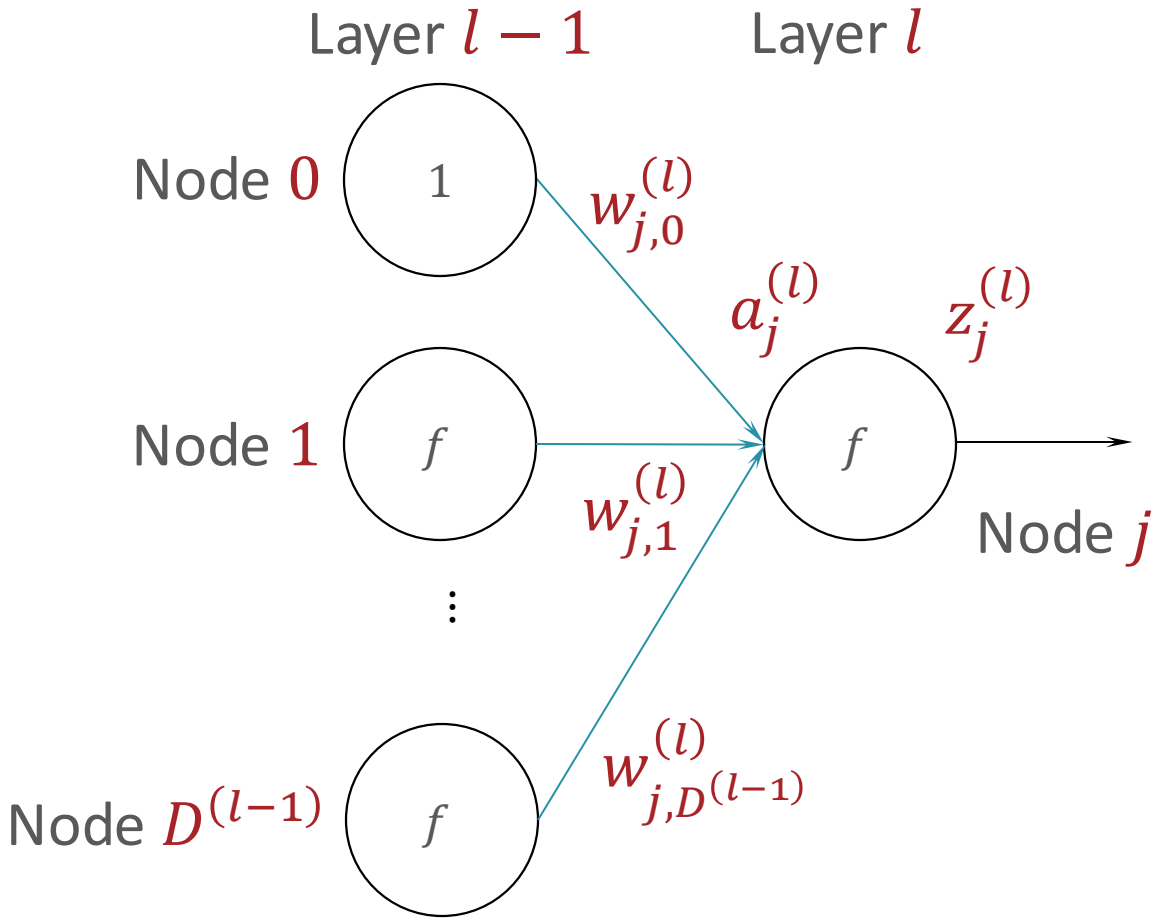


Figure courtesy of Matt Gormley

Signal and Outputs

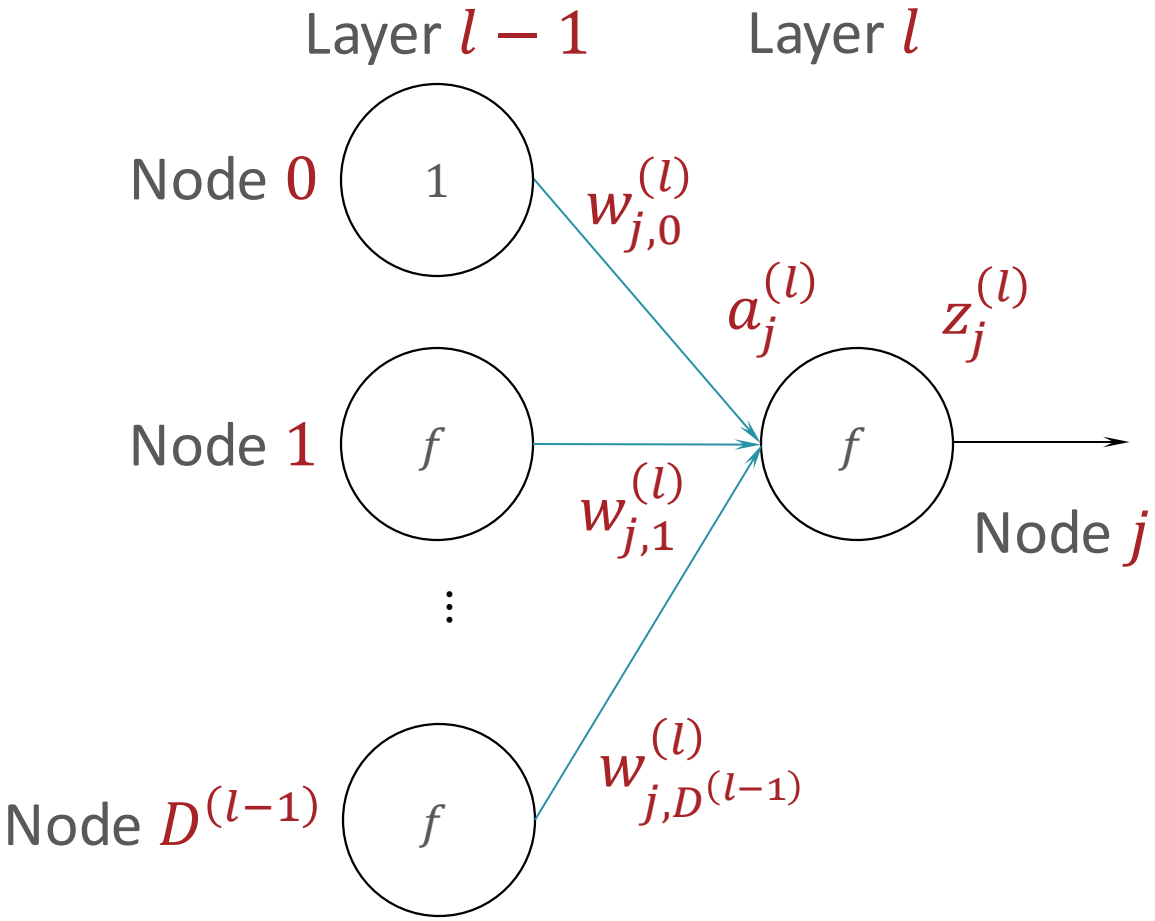
Every node has an incoming *signal* and outgoing *output*



$$a_j^{(l)} = \sum_{i=0}^{D^{(l-1)}} w_{j,i}^{(l)} z_i^{(l-1)} \text{ and } z_j^{(l)} = f\left(a_j^{(l)}\right)$$

Signal and Outputs

Every node has an incoming *signal* and outgoing *output*



$$\boldsymbol{a}^{(l)} = W^{(l)}\boldsymbol{z}^{(l-1)} \text{ and } \boldsymbol{z}^{(l)} = \left[1, f\left(\boldsymbol{z}^{(l)}\right)\right]^T$$

Forward Propagation for Making Predictions

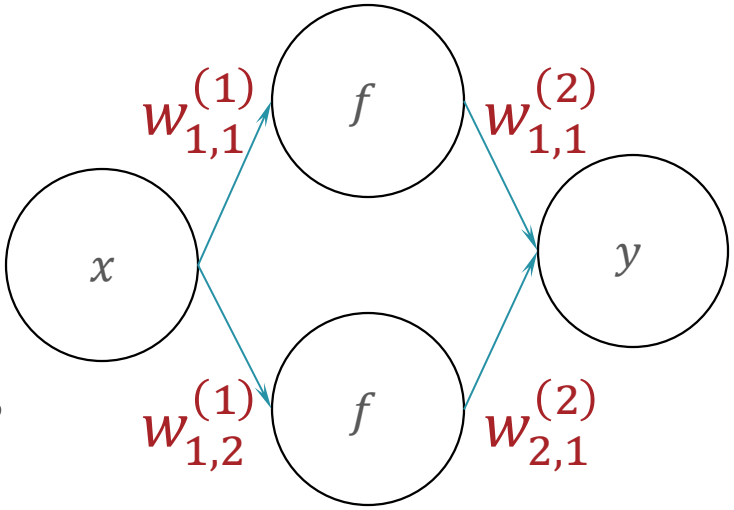
- Input: weights $W^{(1)}, \dots, W^{(L)}$ and a query data point $\boldsymbol{x}$
- Initialize $\boldsymbol{z}^{(0)} = [1, \boldsymbol{x}]^T$
- For $l = 1, \dots, L$
  - $\boldsymbol{a}^{(l)} = W^{(l)} \boldsymbol{z}^{(l-1)}$
  - $\boldsymbol{z}^{(l)} = \left[1, f\left(\boldsymbol{a}^{(l)}\right)\right]^T$
- Output: $h_{W^{(1)},\dots,W^{(L)}}(\boldsymbol{x}) = \boldsymbol{z}^{(L)}$

Gradient Descent for Learning

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}, \eta^{(0)}$
- Initialize all weights $W_{(0)}^{(1)}, \ldots, W_{(0)}^{(L)}$ to small, random numbers and set $t = 0$ (???)
- While TERMINATION CRITERION is not satisfied (???)
  - For $l = 1, \ldots, L$
    - Compute $G^{(l)} = \nabla_{W^{(l)}} \ell_{\mathcal{D}} \left(W_{(t)}^{(1)}, \ldots, W_{(t)}^{(L)}\right)$ (???)
    - Update $W^{(l)}$: $W_{(t+1)}^{(l)} = W_{(t)}^{(l)} - \eta_0 G^{(l)}$
  - Increment t: $t = t + 1$
- Output: $W_{(t)}^{(1)}, \ldots, W_{(t)}^{(L)}$

Poll Question 2

- Suppose you are training a two-layer (one-hidden layer) neural network with sigmoid activations for binary classification.



- True or False: There is a unique set of parameters that maximizes the likelihood of the dataset above.

A. TOXIC B. True C. False

Neural Network Learning Objectives

You should be able to…

1. Explain the biological motivations for a neural network
2. Combine simpler models (e.g. linear regression, binary logistic regression, multinomial logistic regression) as components to build up feed-forward neural network architectures
3. Explain the reasons why a neural network can model nonlinear decision boundaries for classification
4. Compare and contrast feature engineering with learning features
5. Identify (some of) the options available when designing the architecture of a neural network
6. Implement a feed-forward neural network