

10-301/601: Introduction to Machine Learning

Lecture 11 – Neural Networks

Matt Gormley & Henry Chai

2/17/25

Front Matter

- Announcements
 - Exam 1 on 2/17 (today!) from 7 PM - 9 PM
 - Make sure you check the seating chart (on [Piazza](#)) so that you know where you're going tonight!
 - Homework 4 released 2/17 (today!), due 2/26 at 11:59 PM

Motivation: Regularization

- **Occam's Razor:** prefer the simplest hypothesis
- What does it mean for a hypothesis (or model) to be **simple**?
 1. small number of features (**model selection**)
 2. small number of “important” features (**shrinkage**)

Regularization

- **Given** objective function: $J(\theta)$
- **Goal** is to find: $\hat{\theta} = \underset{\theta}{\operatorname{argmin}} J(\theta) + \lambda r(\theta)$
- **Key idea:** Define regularizer $r(\theta)$ s.t. we tradeoff between fitting the data and keeping the model simple
- **Choose form of $r(\theta)$:**
 - Example: q-norm (usually p-norm): $\|\theta\|_q = \left(\sum_{m=1}^M |\theta_m|^q \right)^{\frac{1}{q}}$

q	$r(\theta)$	yields parameters that are...	name	optimization notes
0	$\ \theta\ _0 = \sum \mathbb{1}(\theta_m \neq 0)$	zero values	L0 reg.	no good computational solutions
1	$\ \theta\ _1 = \sum \theta_m $	zero values	L1 reg.	subdifferentiable
2	$(\ \theta\ _2)^2 = \sum \theta_m^2$	small values	L2 reg.	differentiable

Regularization Examples

Add an **L2 regularizer** to Linear Regression (aka. Ridge Regression)

$$J_{\text{RR}}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_2^2$$

$$= \frac{1}{N} \sum_{i=1}^N \frac{1}{2} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 + \lambda \sum_{m=1}^M \theta_m^2$$

dimensions or features

Add an **L1 regularizer** to Linear Regression (aka. LASSO)

$$J_{\text{LASSO}}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1$$

$$= \frac{1}{N} \sum_{i=1}^N \frac{1}{2} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 + \lambda \sum_{m=1}^M |\theta_m|$$

Regularization Examples

Add an **L2 regularizer** to Logistic Regression

$$\begin{aligned} J'(\boldsymbol{\theta}) &= J(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_2^2 \\ &= \frac{1}{N} \sum_{i=1}^N -\log p(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}) + \lambda \sum_{m=1}^M \theta_m^2 \end{aligned}$$

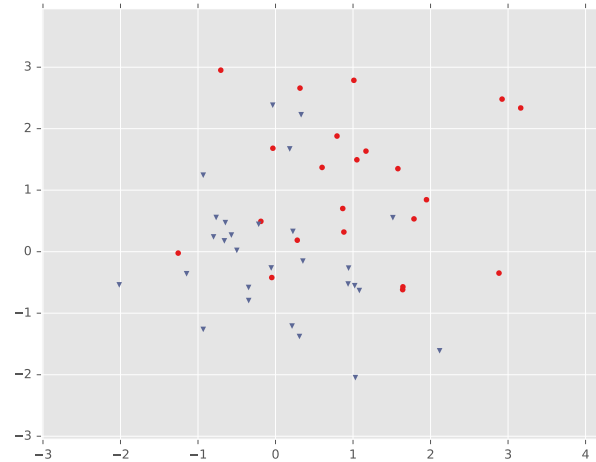
Add an **L1 regularizer** to Logistic Regression

$$\begin{aligned} J'(\boldsymbol{\theta}) &= J(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1 \\ &= \frac{1}{N} \sum_{i=1}^N -\log p(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}) + \lambda \sum_{m=1}^M |\theta_m| \end{aligned}$$

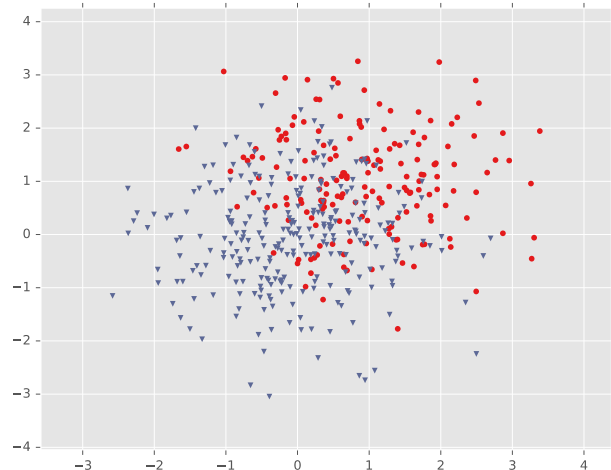
REGULARIZATION EXAMPLE: LOGISTIC REGRESSION

Example: Logistic Regression

Training
Data

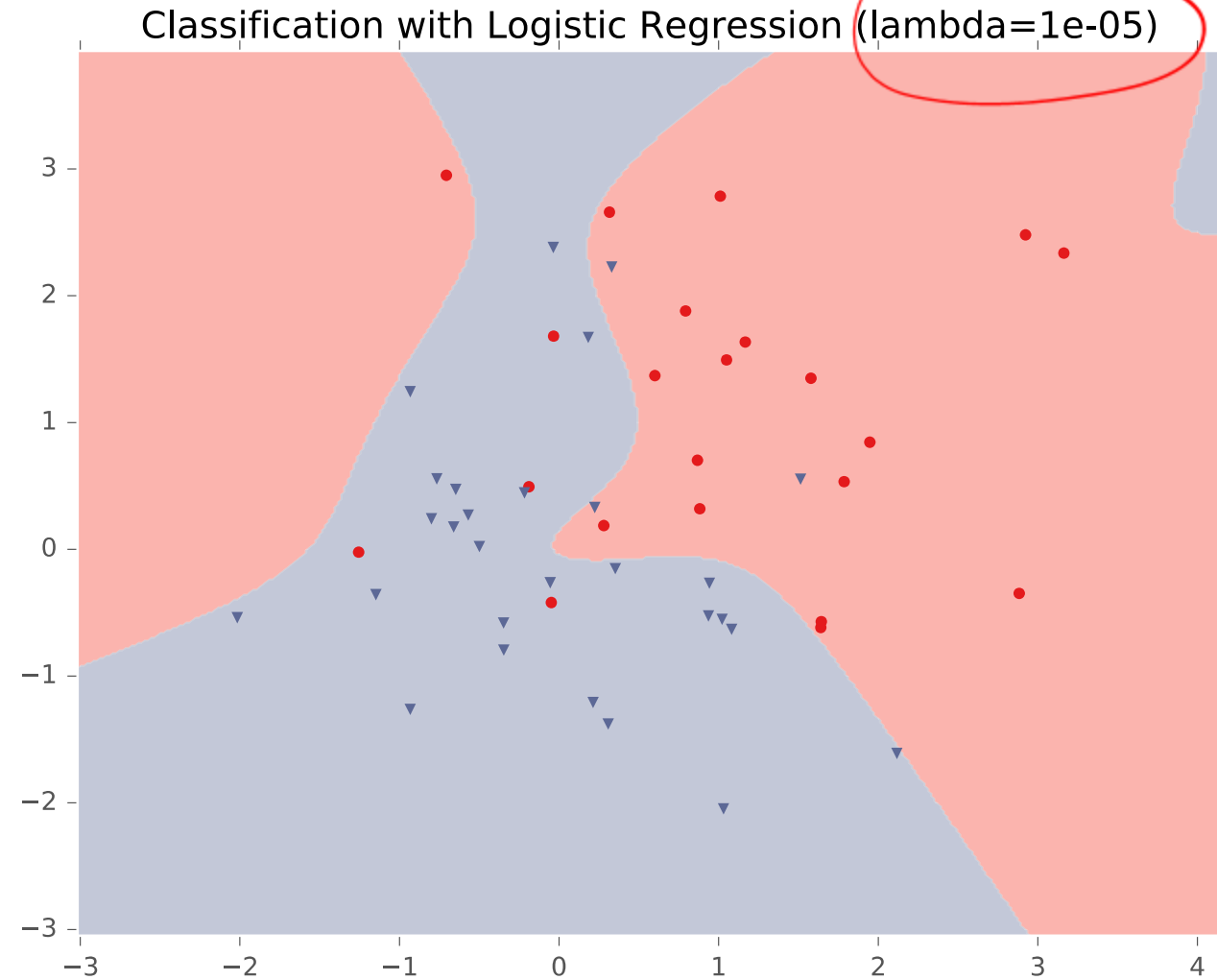


Test
Data

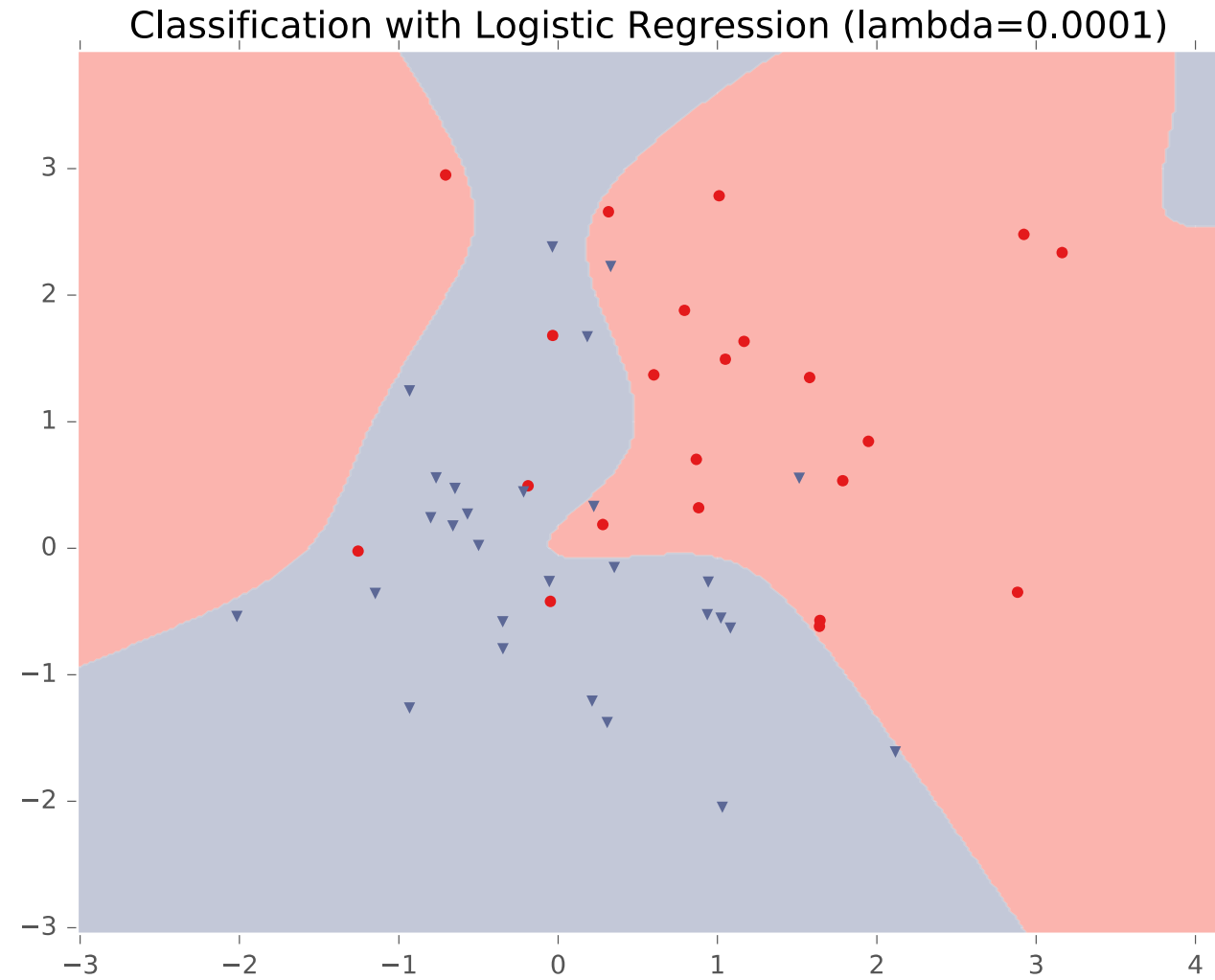


- For this example, we construct **nonlinear features** (i.e. feature engineering)
- Specifically, we add **polynomials up to order 9** of the two original features x_1 and x_2
- Thus our classifier is **linear** in the **high-dimensional feature space**, but the decision boundary is **nonlinear** when visualized in **low-dimensions** (i.e. the original two dimensions)

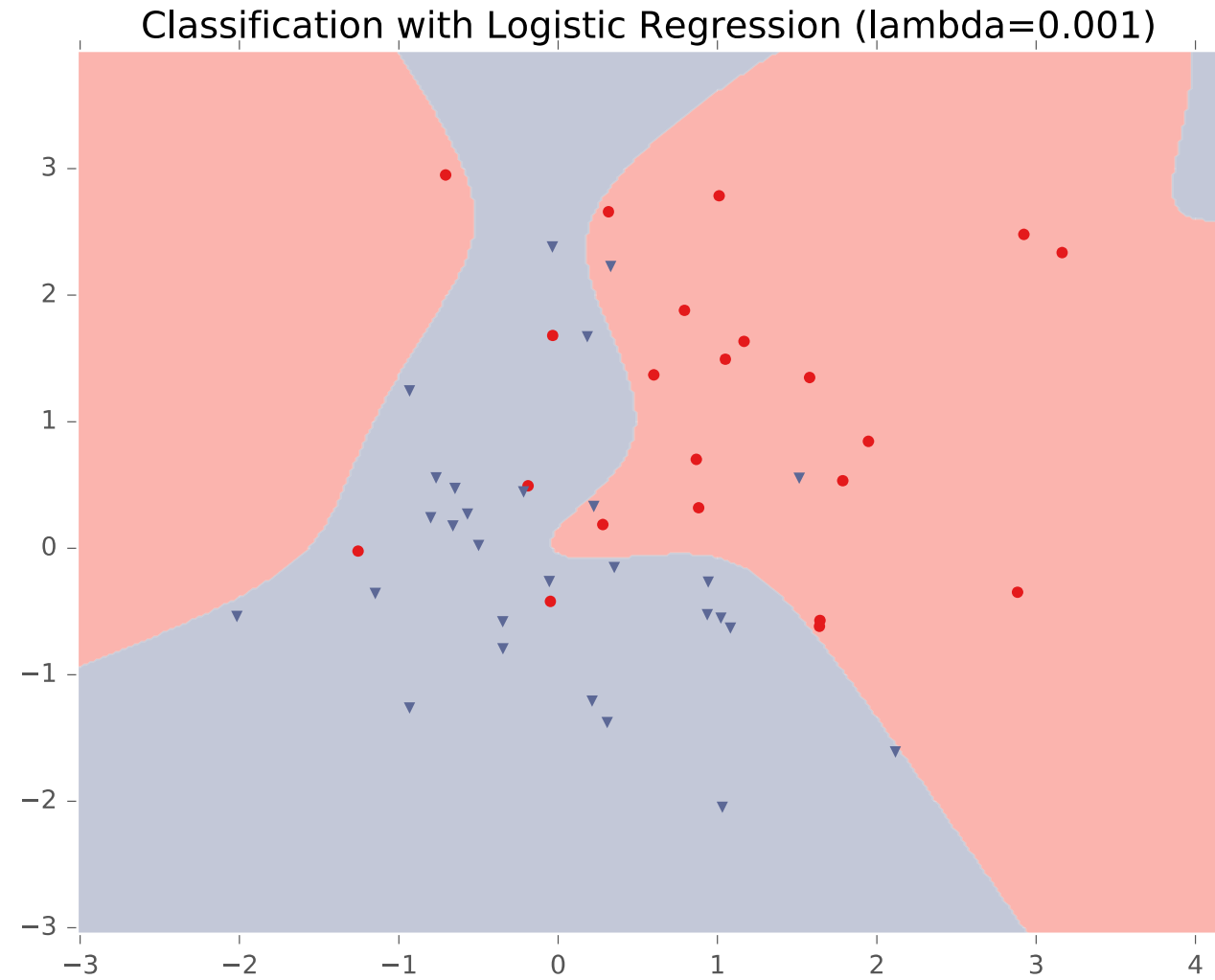
Example: Logistic Regression



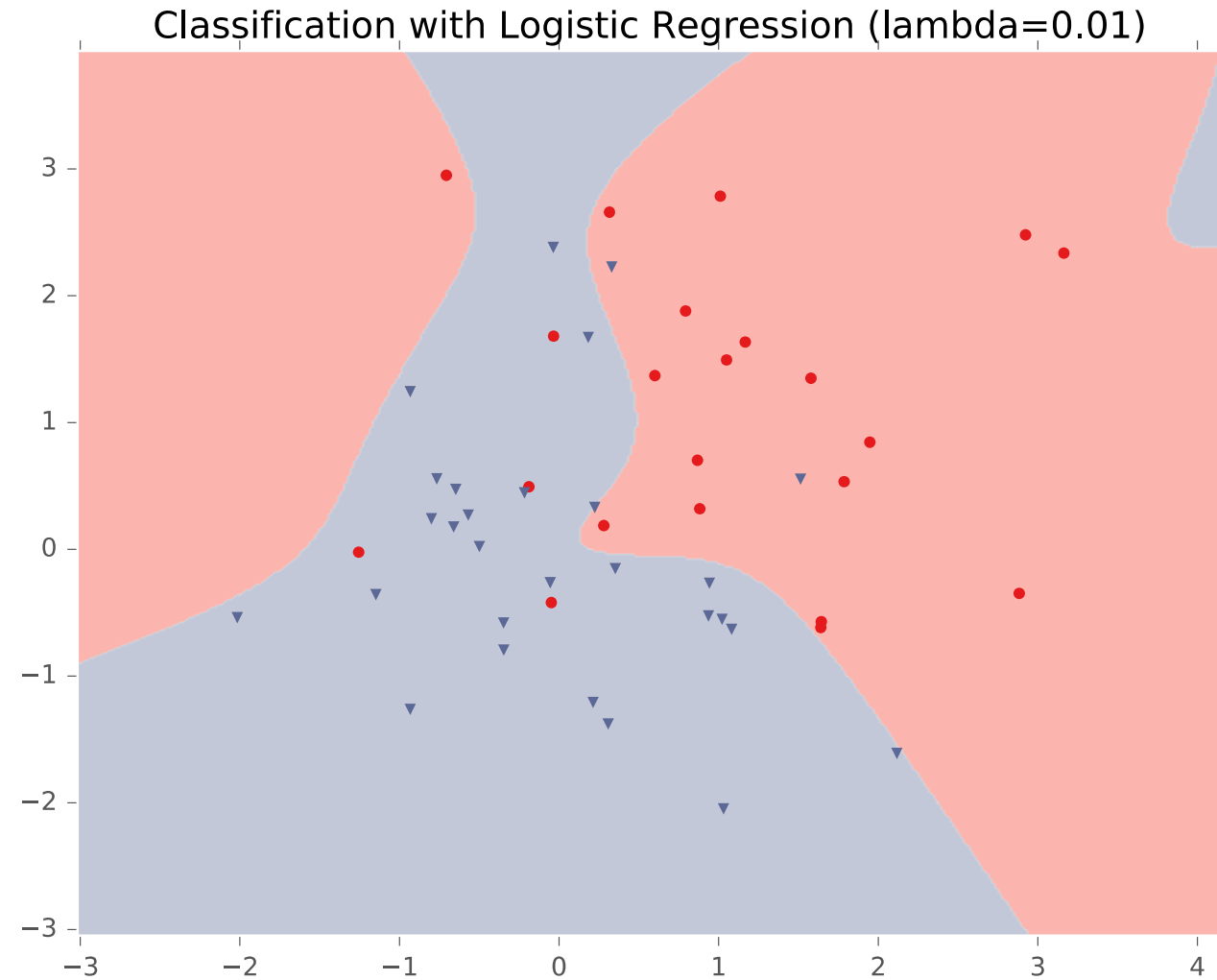
Example: Logistic Regression



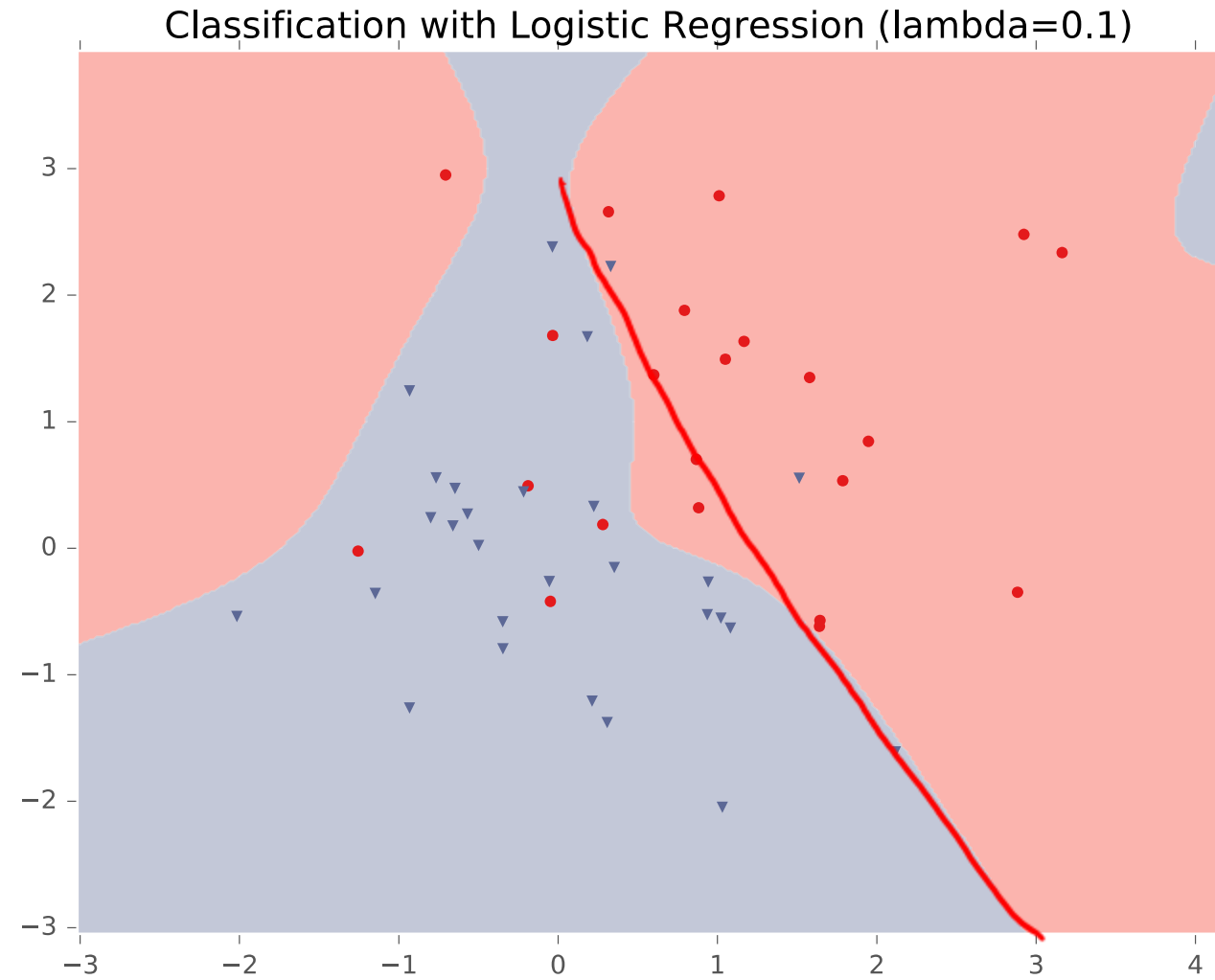
Example: Logistic Regression



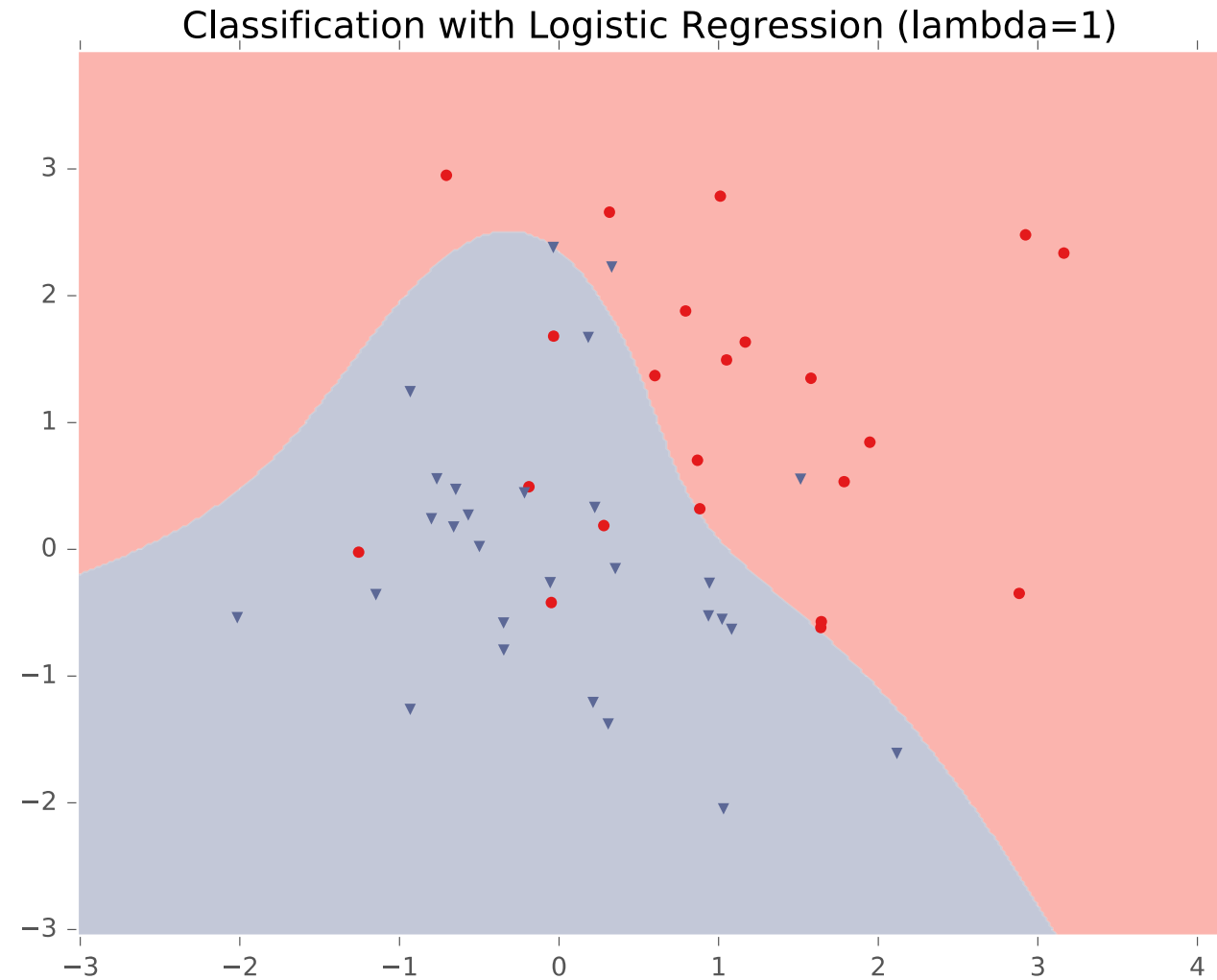
Example: Logistic Regression



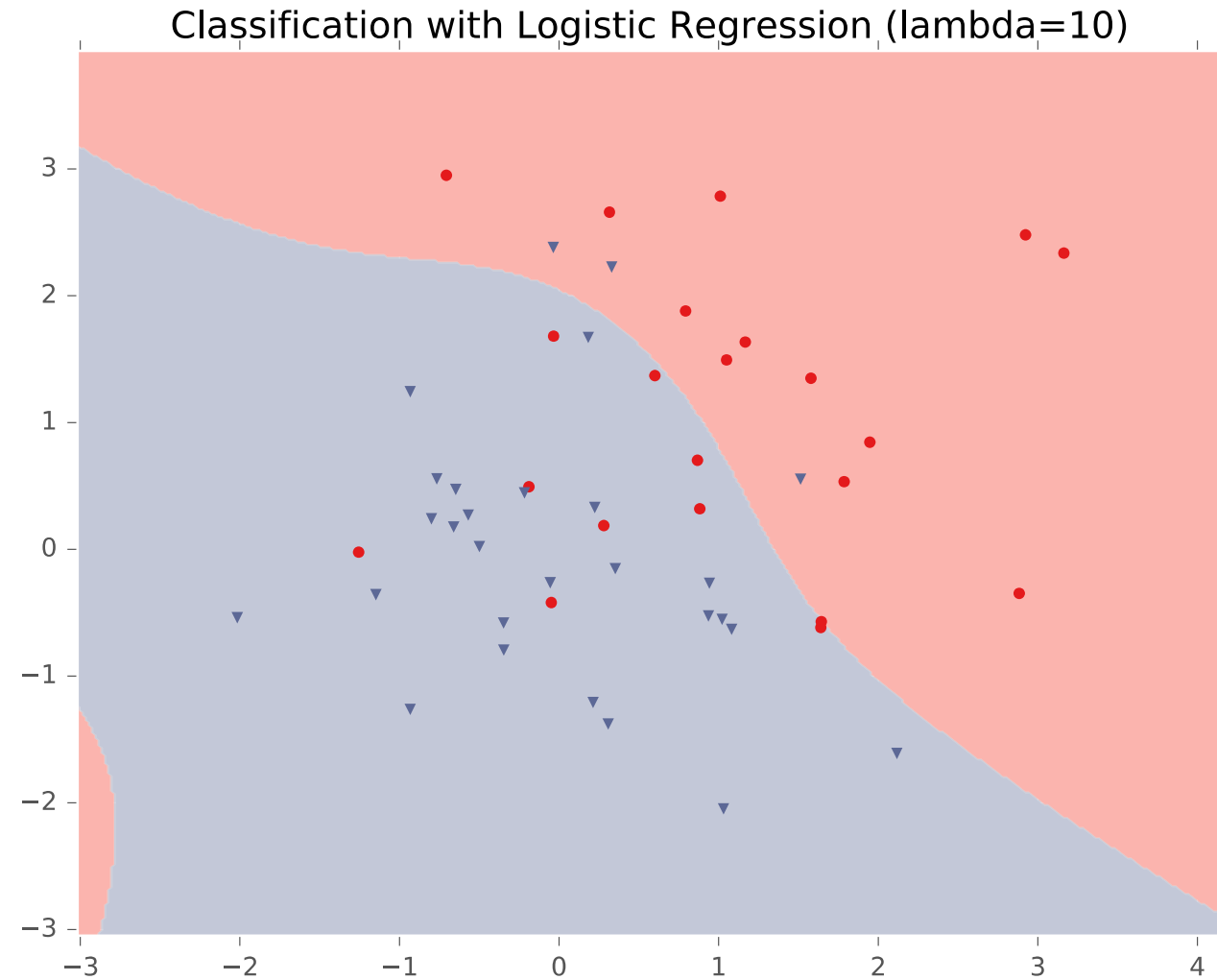
Example: Logistic Regression



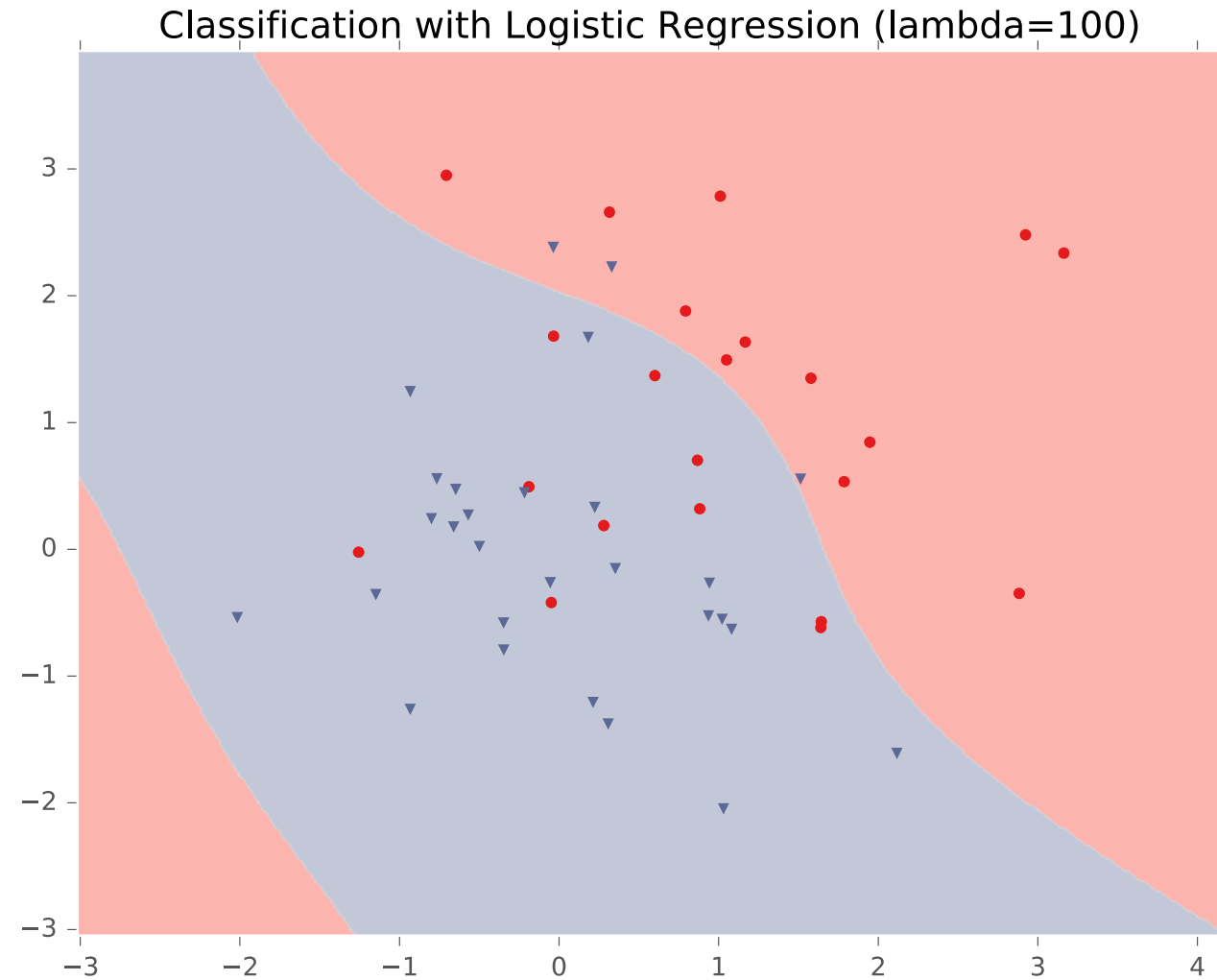
Example: Logistic Regression



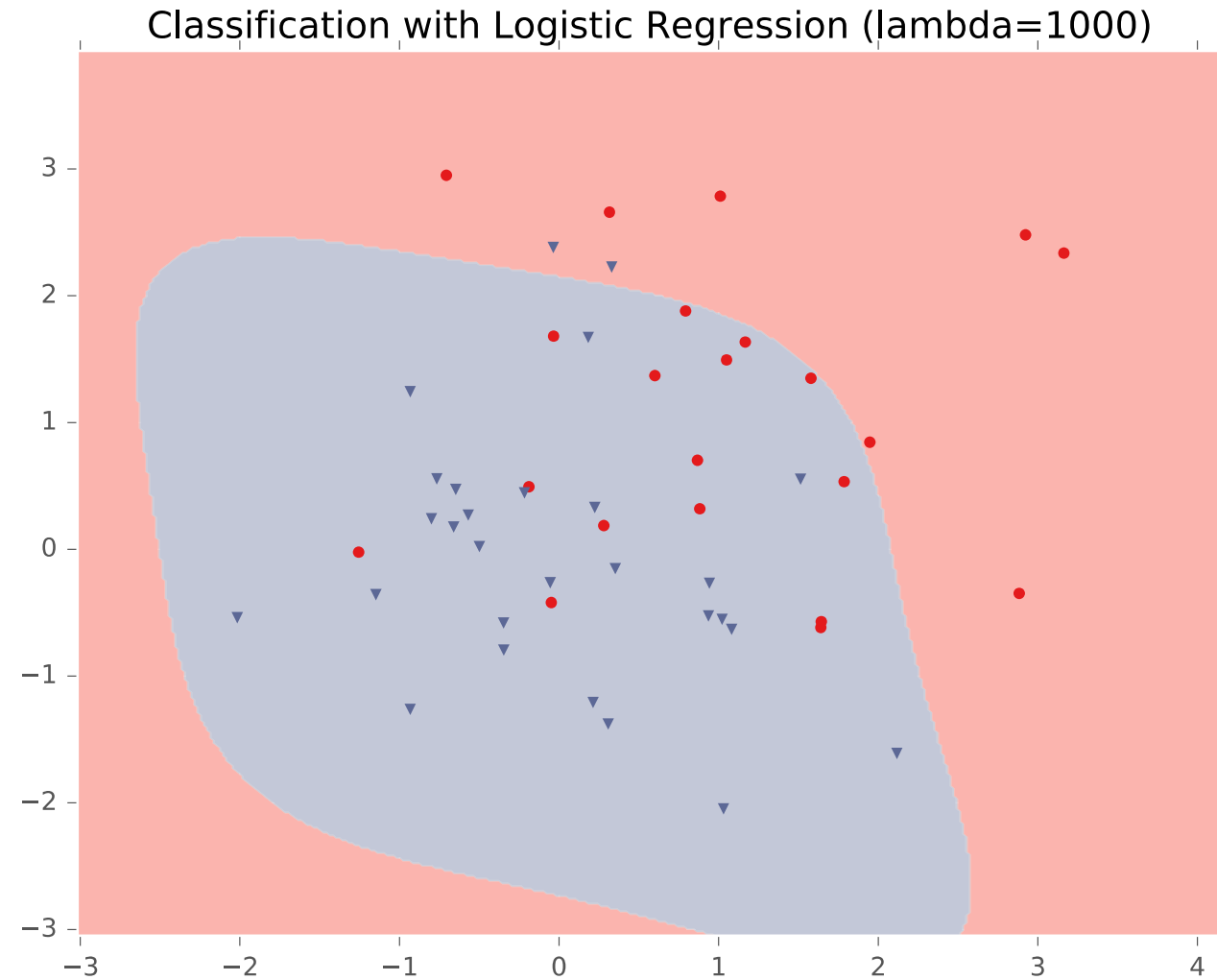
Example: Logistic Regression



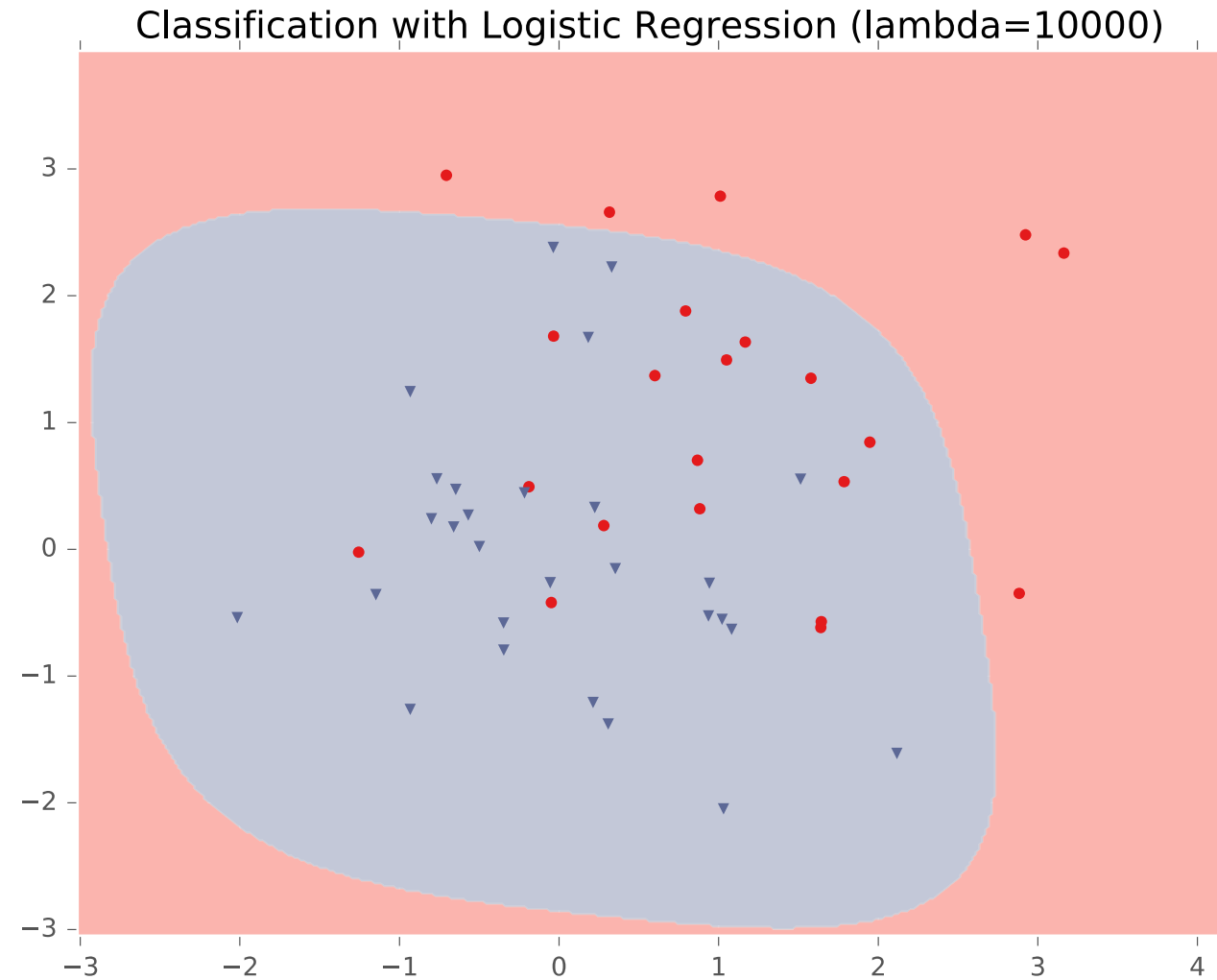
Example: Logistic Regression



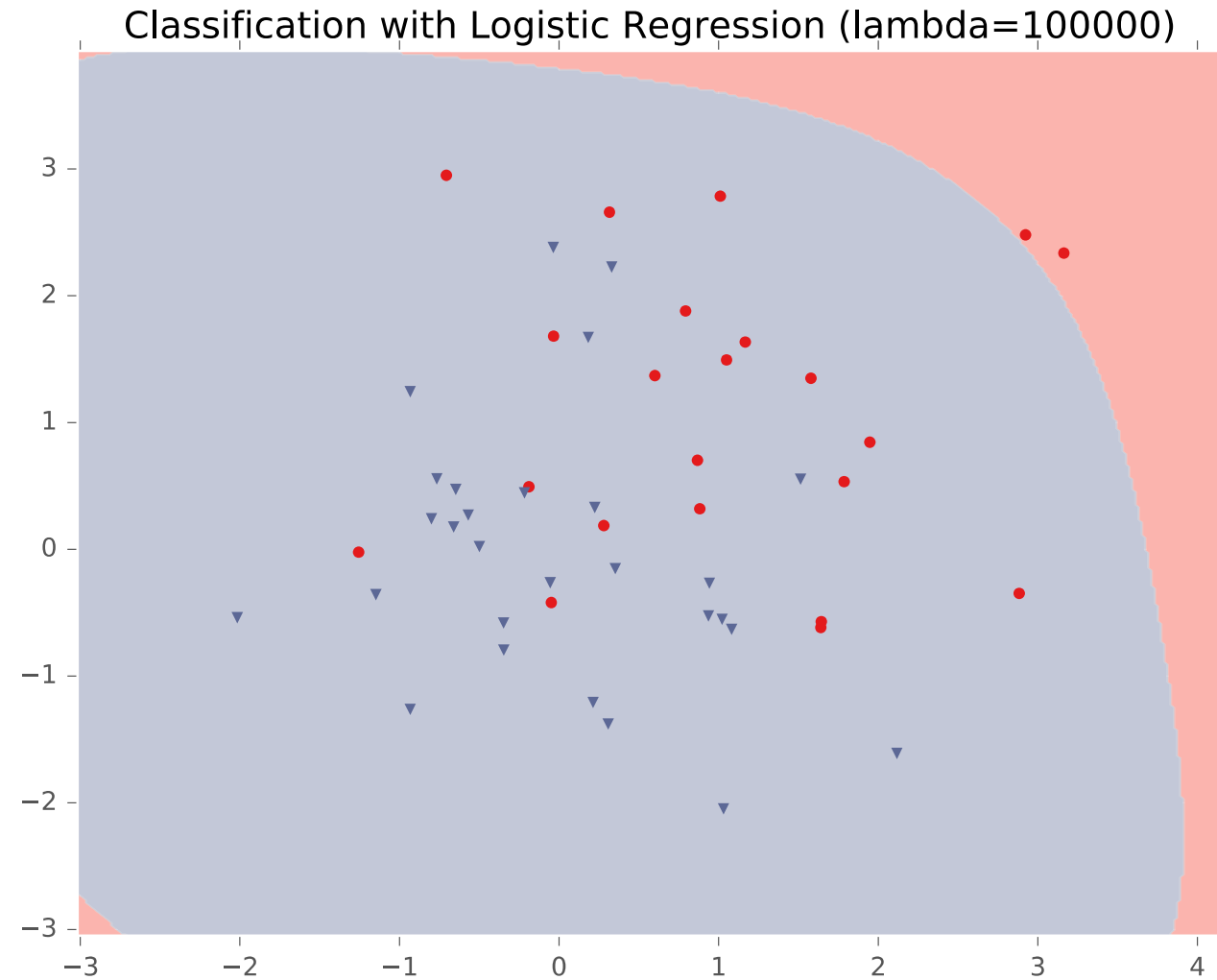
Example: Logistic Regression



Example: Logistic Regression

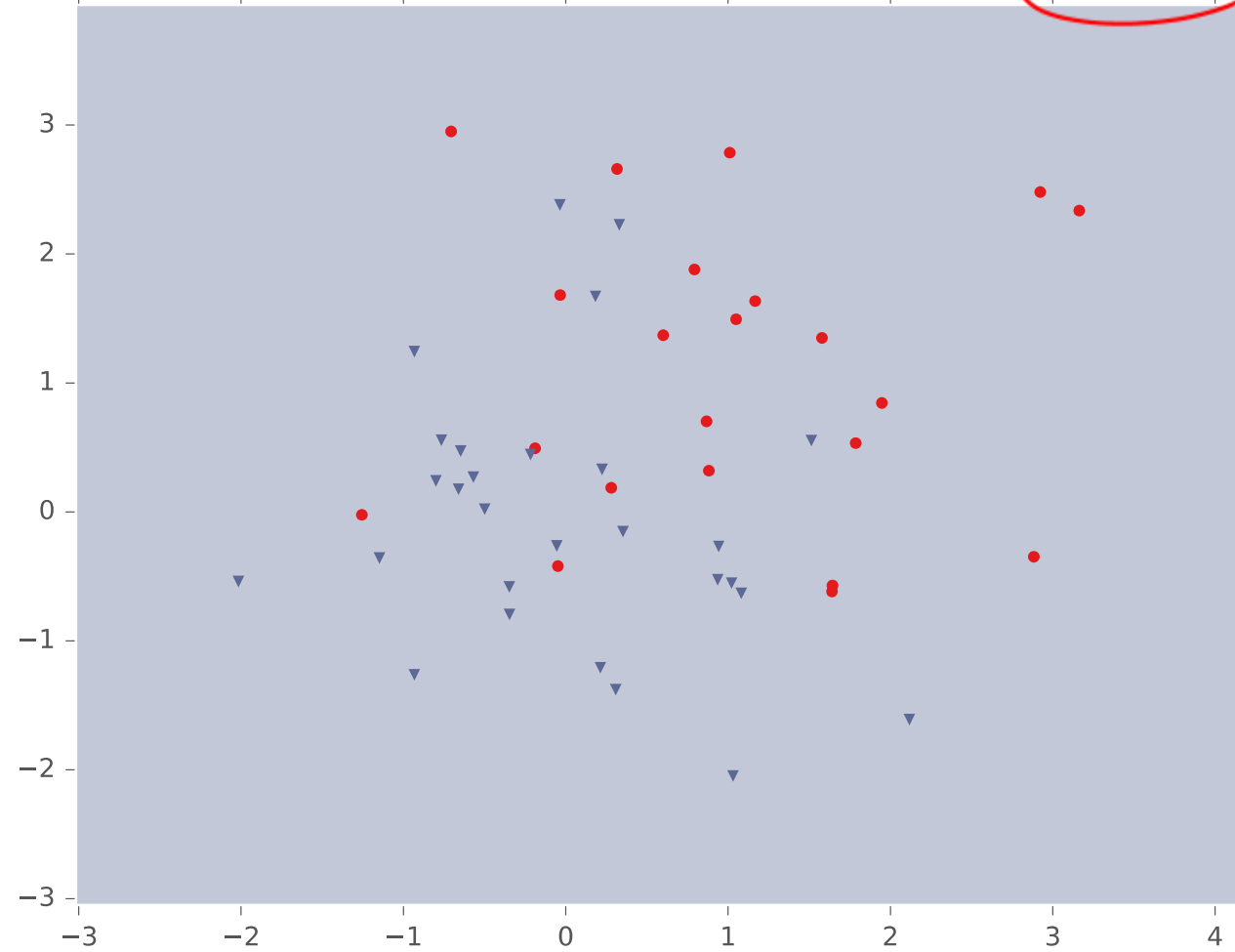


Example: Logistic Regression

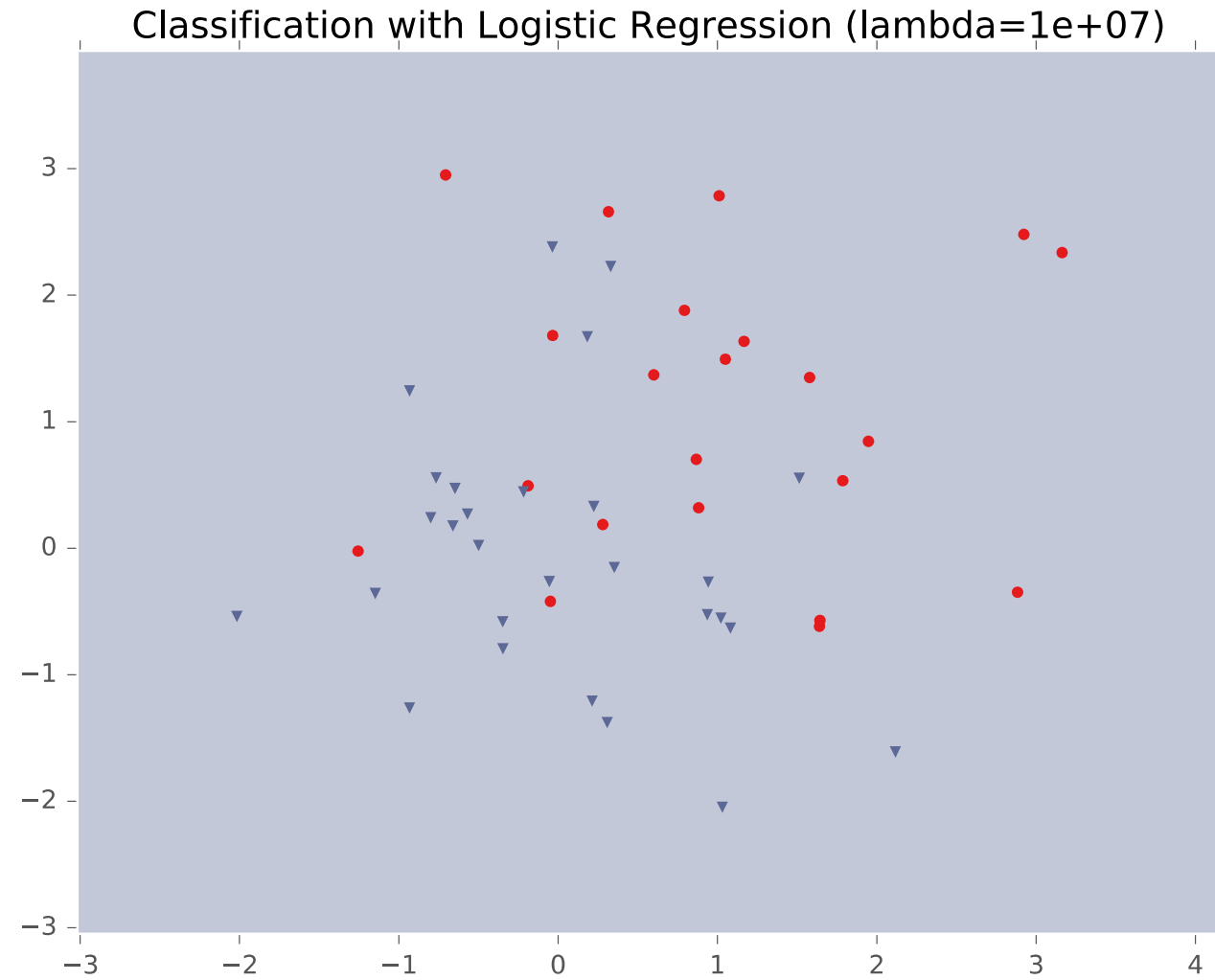


Example: Logistic Regression

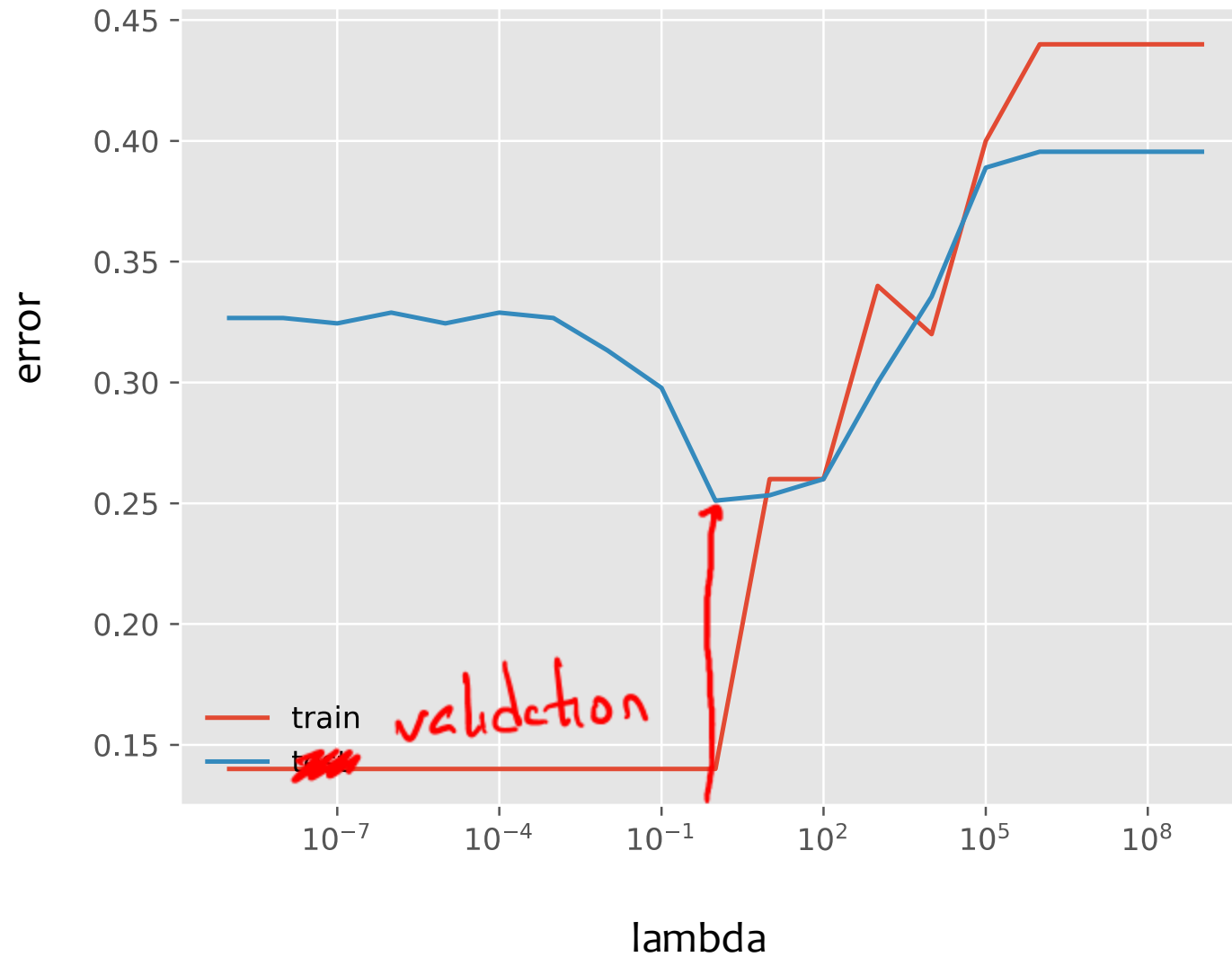
Classification with Logistic Regression ($\lambda=1e+06$)



Example: Logistic Regression



Example: Logistic Regression



Regularization

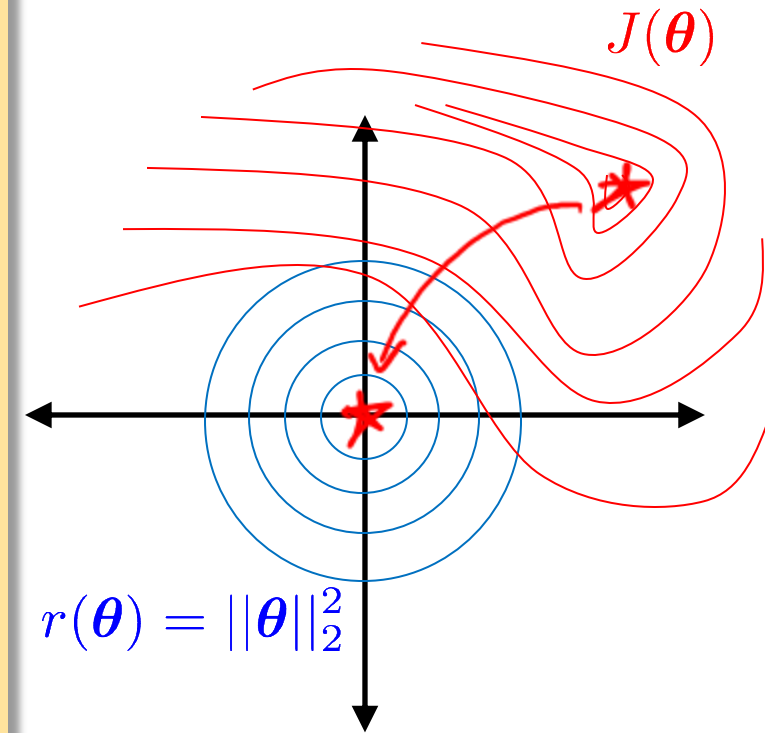
Poll Question 1:

Suppose we are minimizing $J'(\theta)$ where

$$J'(\theta) = J(\theta) + \lambda r(\theta)$$

As λ increases, the minimum of $J'(\theta)$ will...

- A. ...move towards the midpoint between $J(\theta)$ and $r(\theta)$
- B. ...move towards the minimum of $J(\theta)$
- ☒ C. ...move towards the minimum of $r(\theta)$
- D. ...move towards a theta vector of positive infinities
- E. ...move towards a theta vector of negative infinities
- F. ...stay the same



Regularization: Best Practices

Don't Regularize the Bias (Intercept) Parameter!

- In our models so far, the bias / intercept parameter is usually denoted by θ_0 -- that is, the parameter for which we fixed $x_0 = 1$
- Regularizers always avoid penalizing this bias / intercept parameter
- Why? Because otherwise the learning algorithms wouldn't be invariant to a shift in the y-values

Standardizing Data

- It's common to *standardize* each feature by subtracting its mean and dividing by its standard deviation
- For regularization, this helps all the features be penalized in the same units
(e.g. convert both centimeters and kilometers to z-scores)

Takeaways

1. **Nonlinear basis functions** allow **linear models** (e.g. Linear Regression, Logistic Regression) to capture **nonlinear** aspects of the original input
2. Nonlinear features **require no changes to the model** (i.e. just preprocessing)
3. **Regularization** helps to avoid **overfitting**

Feature Engineering / Regularization Objectives

You should be able to...

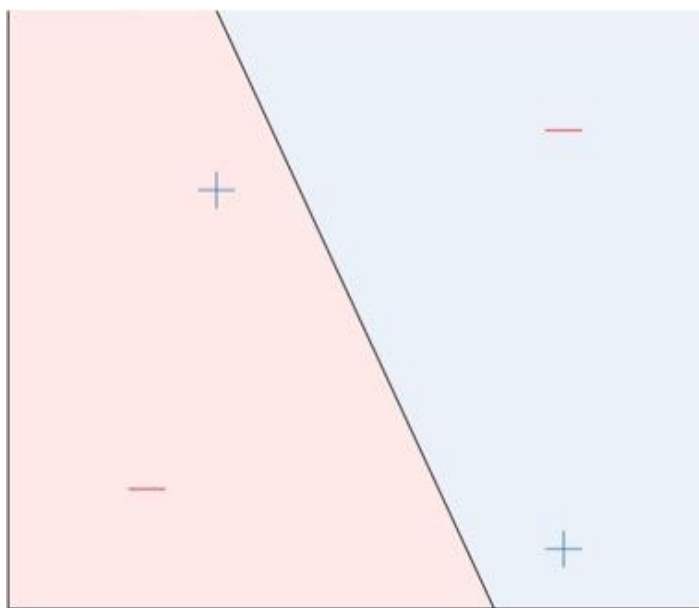
- Engineer appropriate features for a new task
- Use feature selection techniques to identify and remove irrelevant features
- Identify when a model is overfitting
- Add a regularizer to an existing objective in order to combat overfitting
- Explain why we should **not** regularize the bias term
- Convert linearly inseparable dataset to a linearly separable dataset in higher dimensions
- Describe feature engineering in common application areas

Biological Neural Network

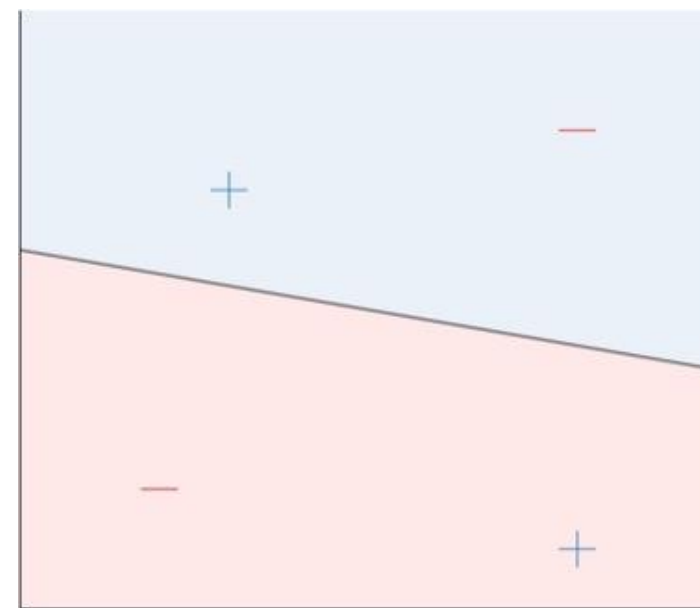
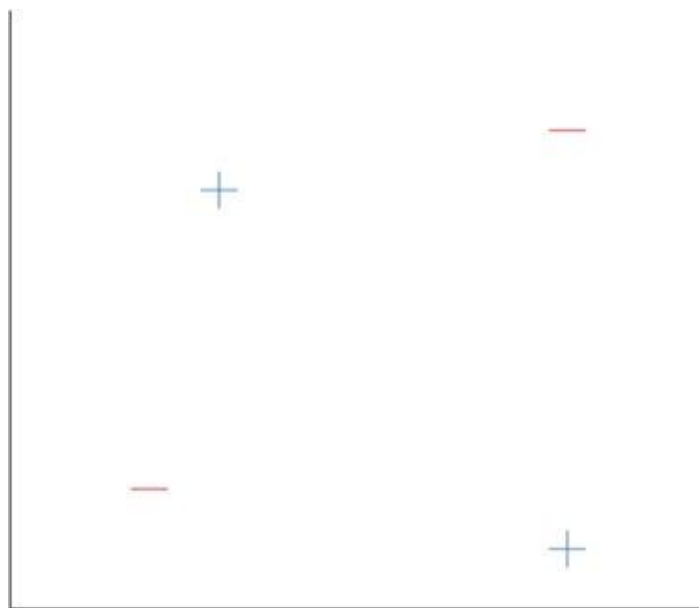


Biological Neural Network(s)





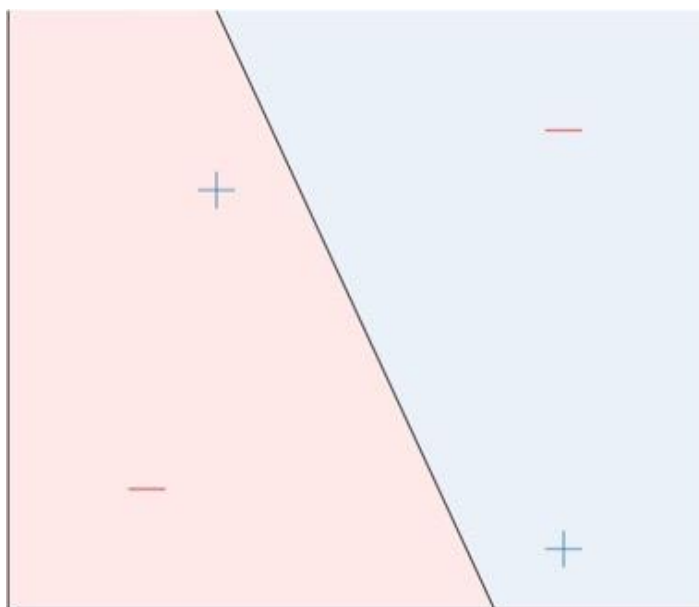
h_1



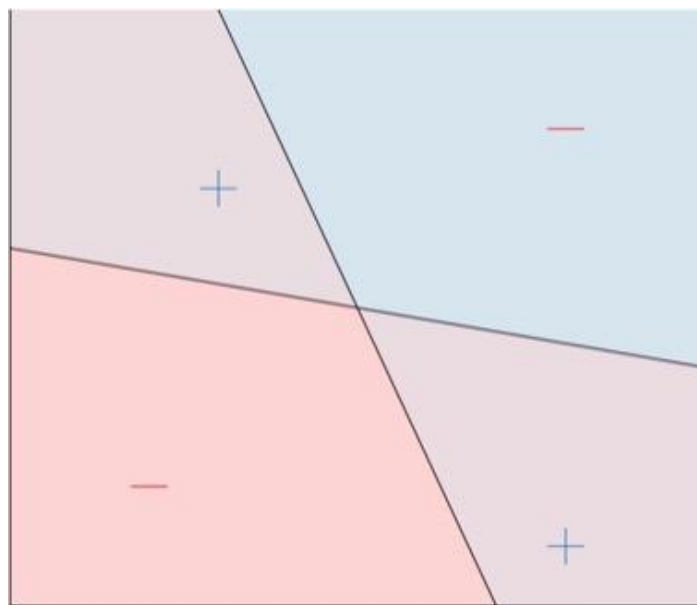
h_2

Perceptrons

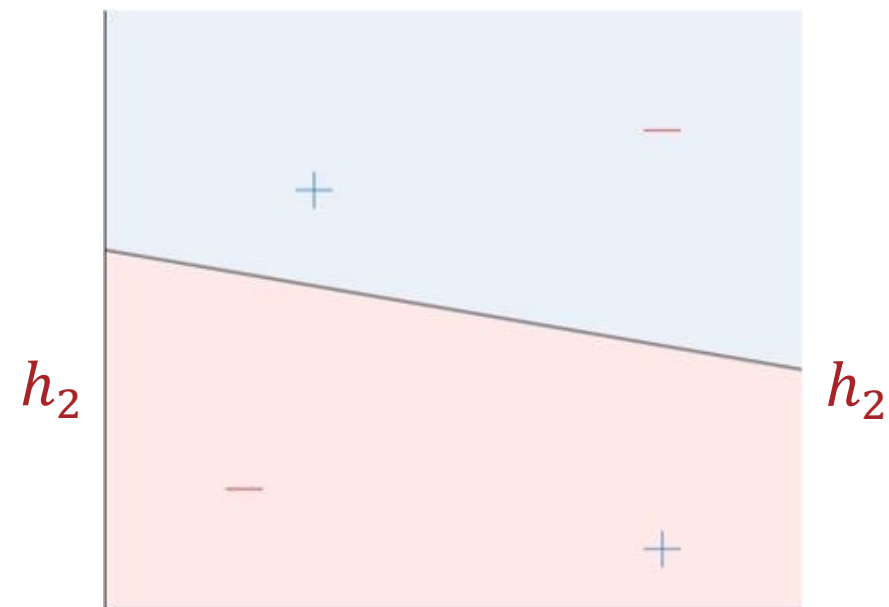
- Linear model for classification
- $h(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x})$
- Predictions are $+1$ or -1



h_1

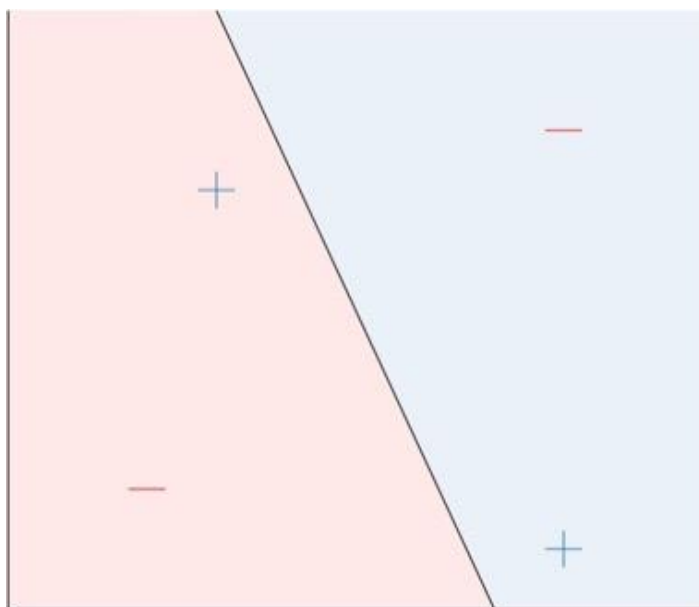


h_1

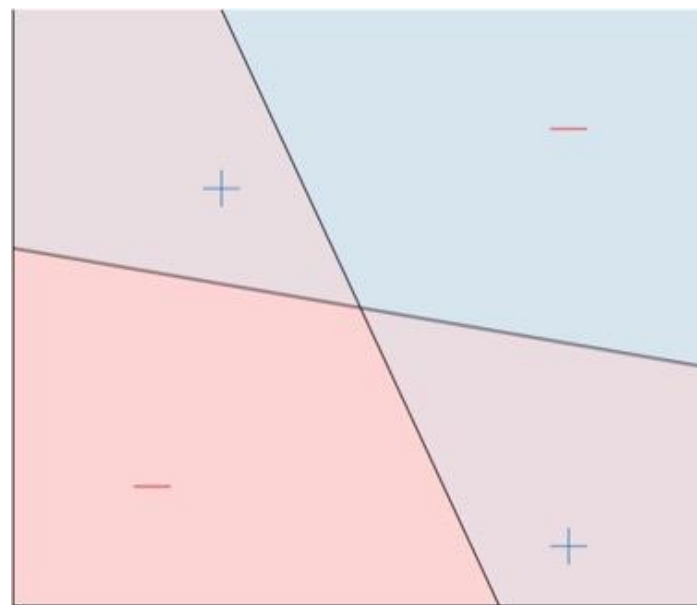


h_2

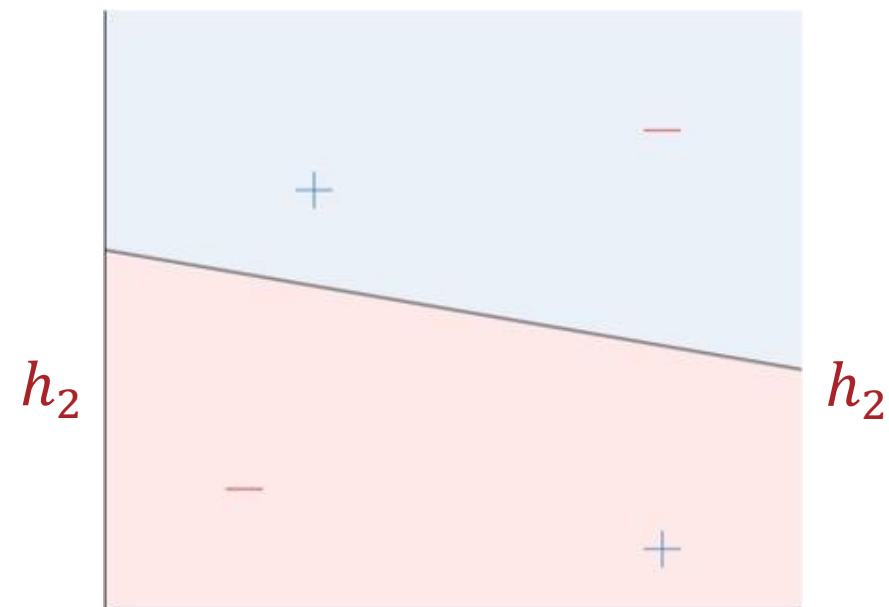
Combining Perceptrons



h_1

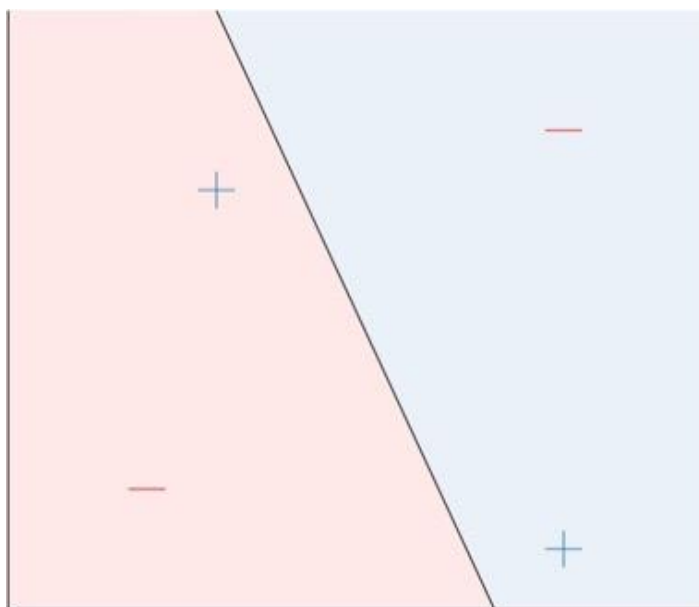


h_1

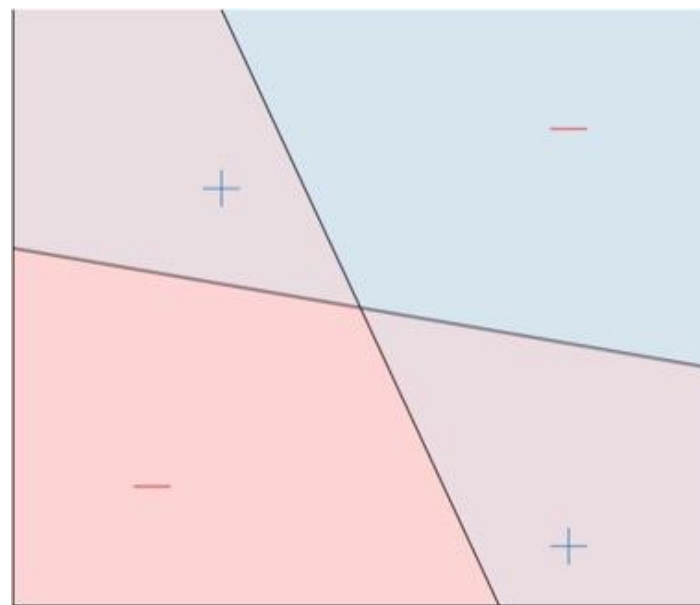


h_2

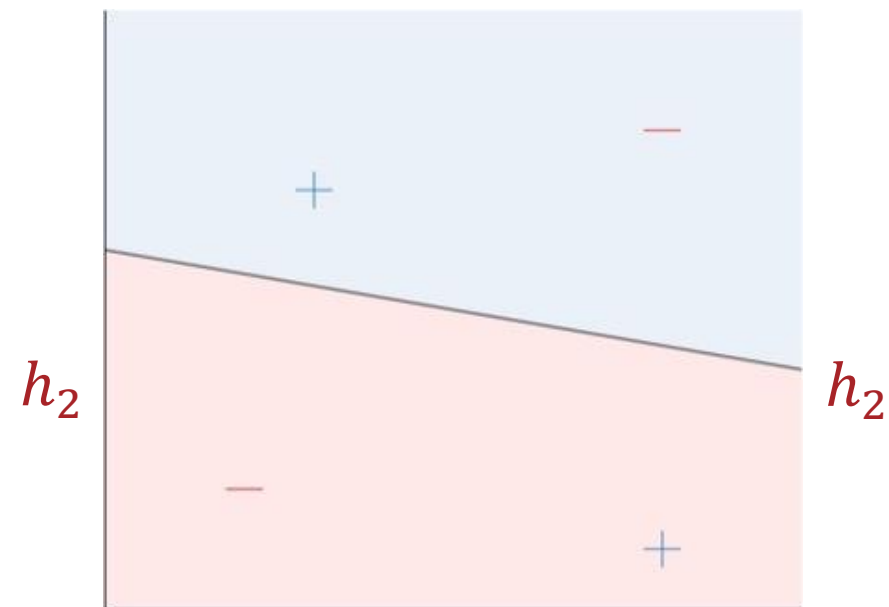
$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } (h_1(\mathbf{x}) = +1 \text{ and } h_2(\mathbf{x}) = -1) \text{ or } (h_1(\mathbf{x}) = -1 \text{ and } h_2(\mathbf{x}) = +1) \\ -1 & \text{otherwise} \end{cases}$$



h_1



h_1



h_2

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

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

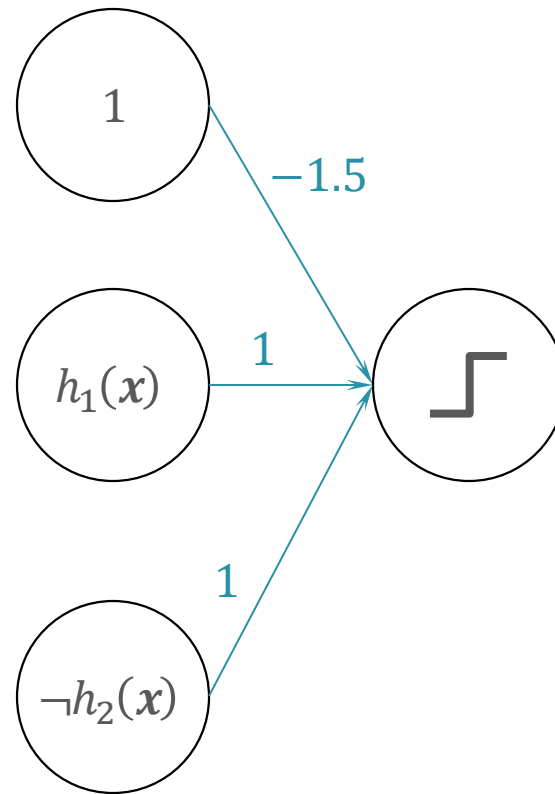
- Boolean variables are either $+1$ ("true") or -1 ("false")
- Basic Boolean operations
 - Negation: $\neg z = -1 * z$
 - And: $AND(z_1, z_2) = \text{sign} \left([-1.5, 1, 1] \begin{bmatrix} 1 \\ z_1 \\ z_2 \end{bmatrix} \right)$
 - Or: $OR(z_1, z_2) = \text{sign} \left([1.5, 1, 1] \begin{bmatrix} 1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

Building a Network

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

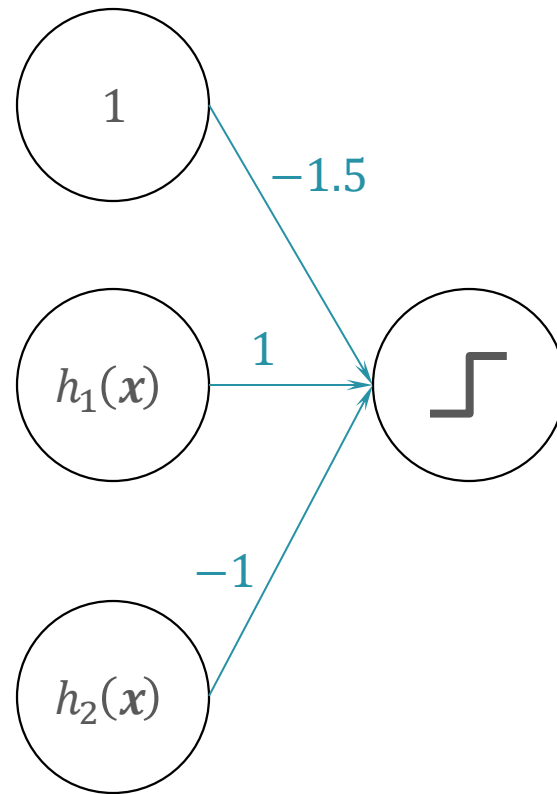
Building a Network

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



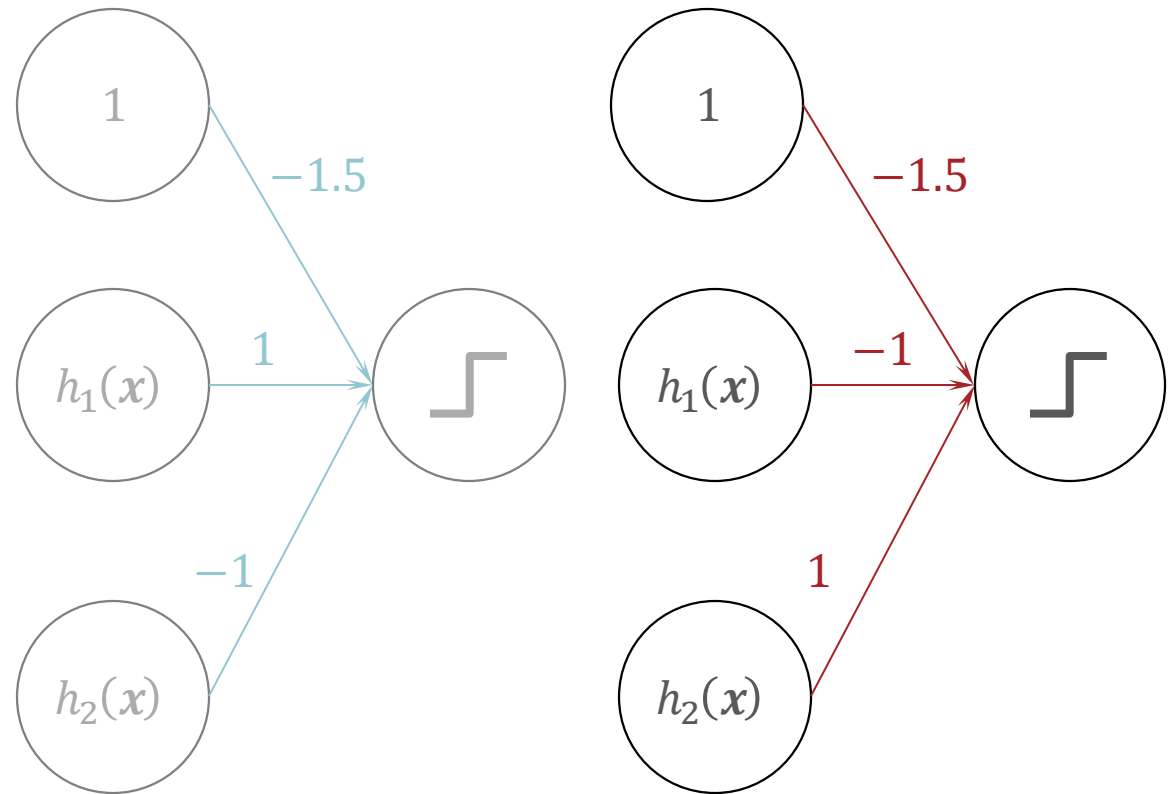
Building a Network

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



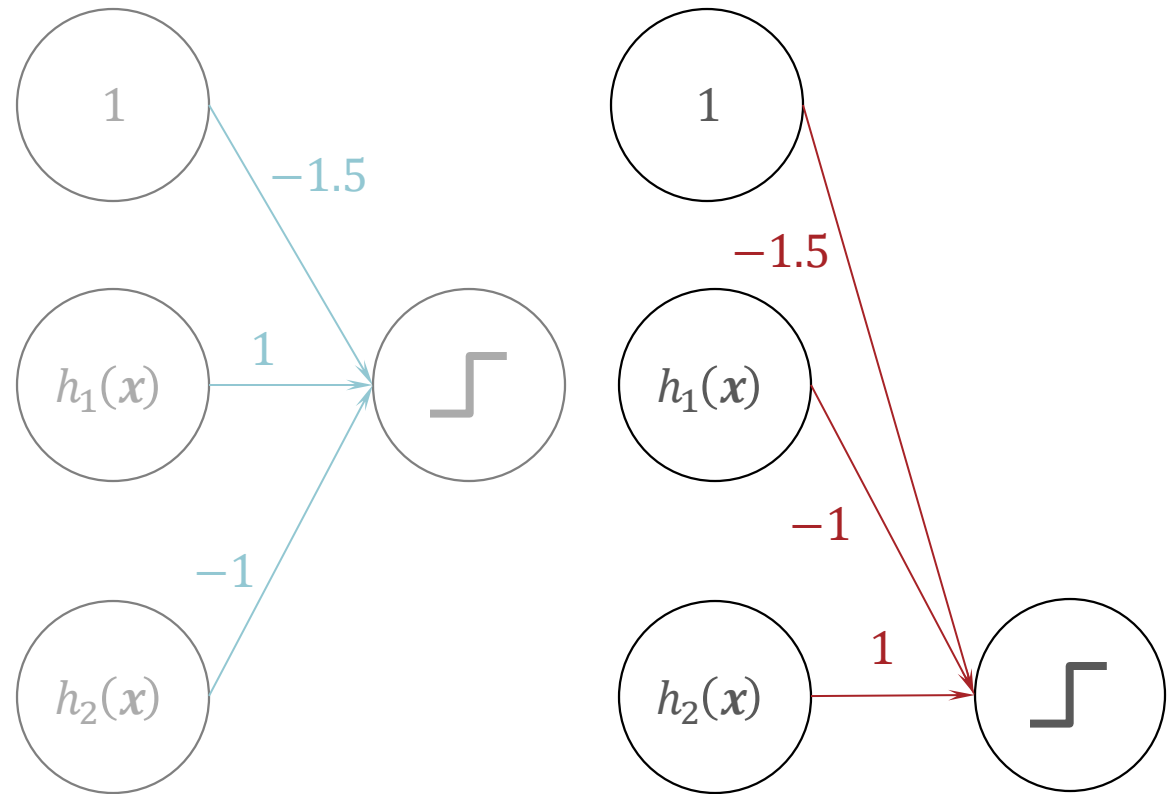
Building a Network

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



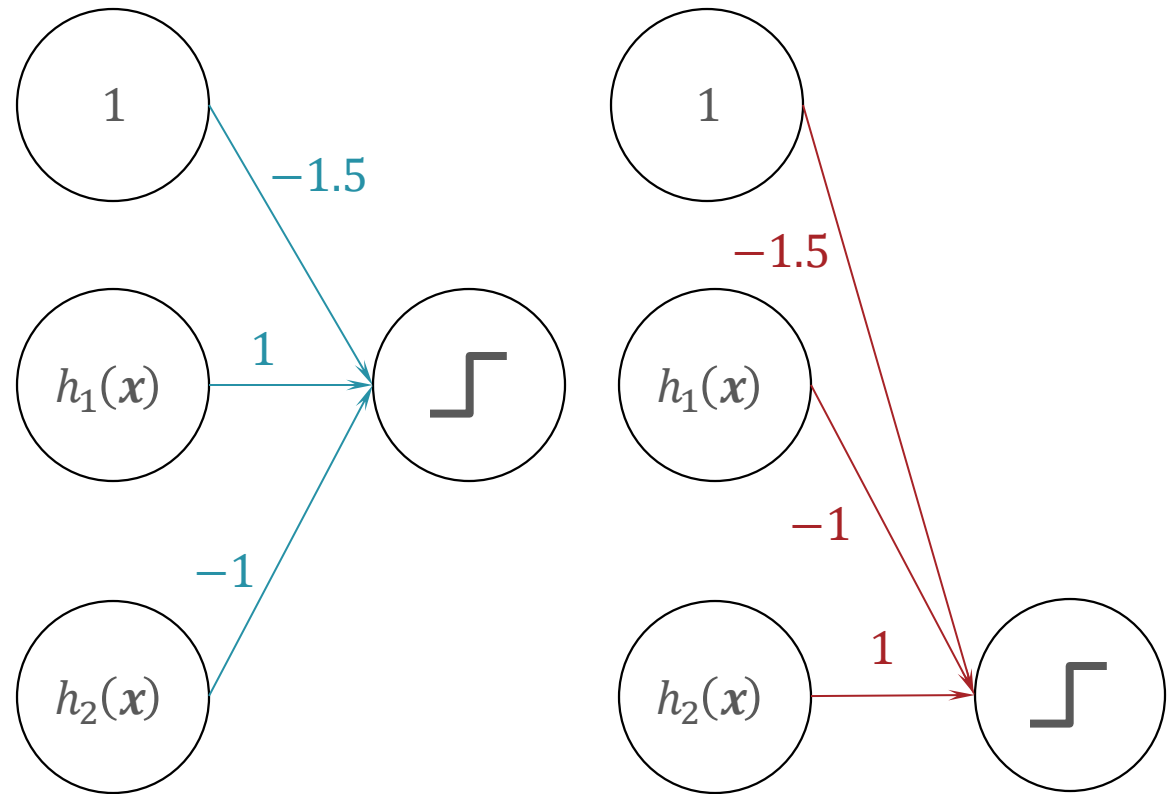
Building a Network

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



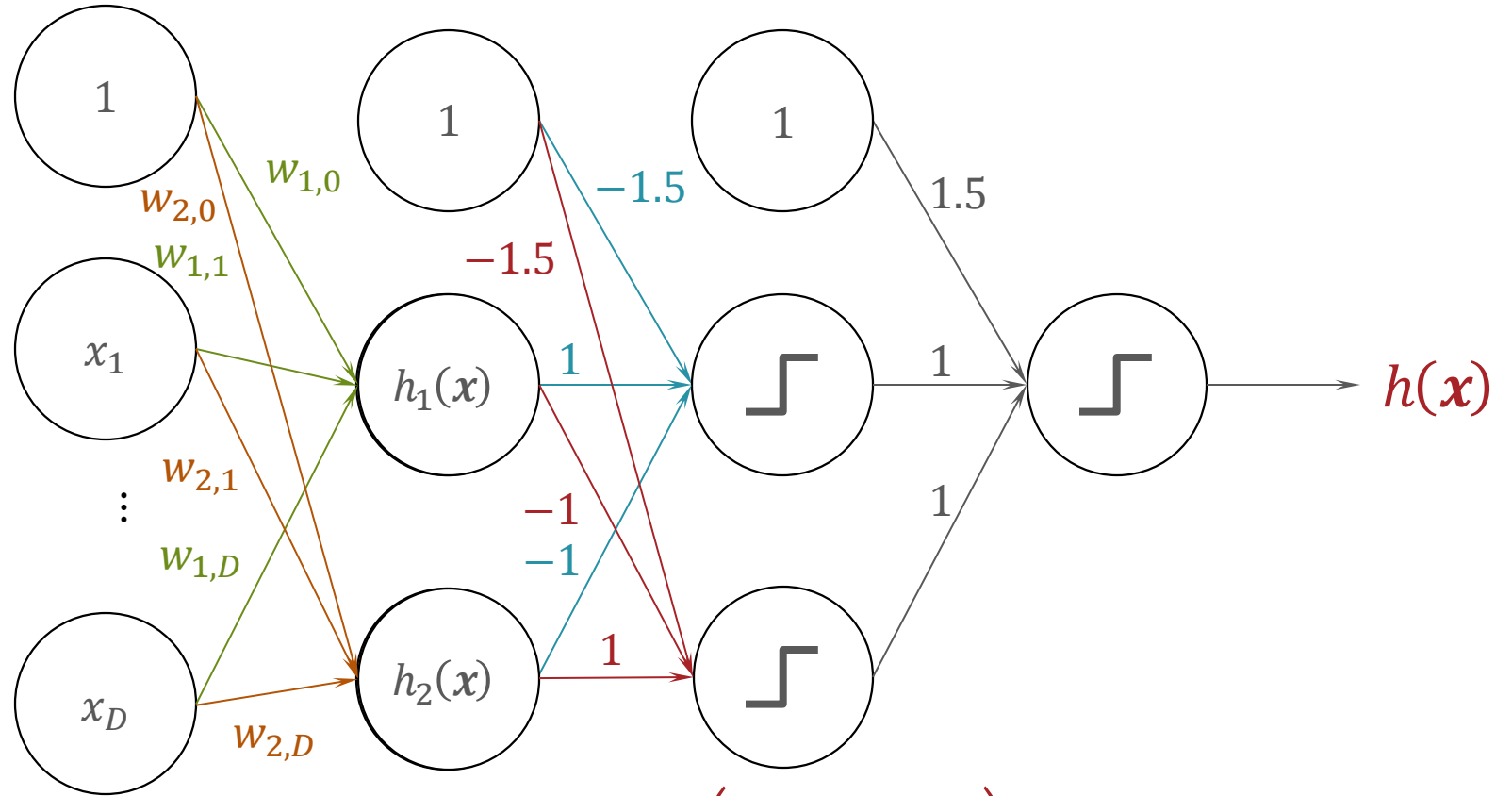
Building a Network

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



Building a Network

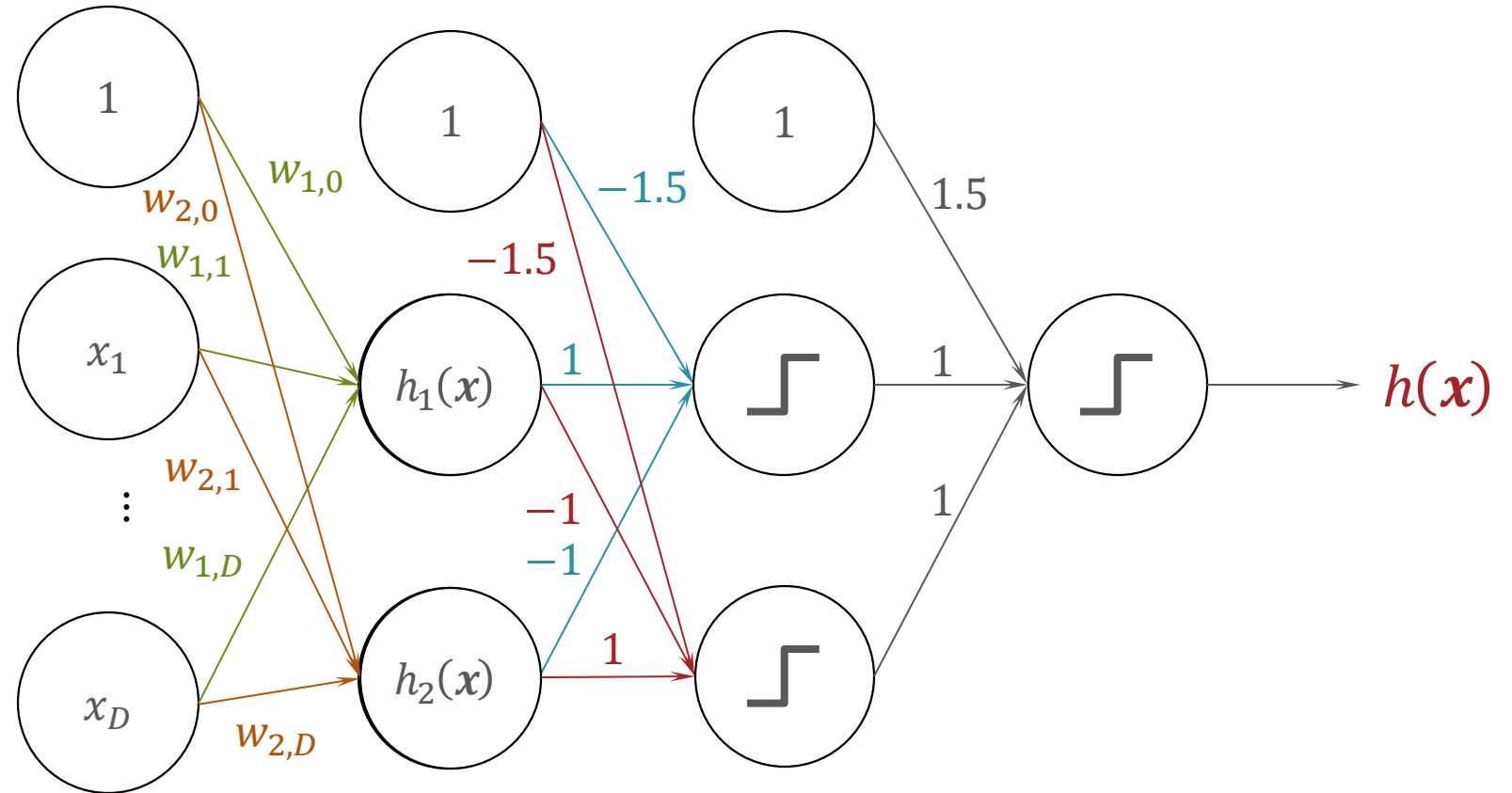
$$h(\mathbf{x}) = \text{OR} \left(\text{AND}(h_1(\mathbf{x}), \neg h_2(\mathbf{x})), \text{AND}(\neg h_1(\mathbf{x}), h_2(\mathbf{x})) \right)$$



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

Building a Network

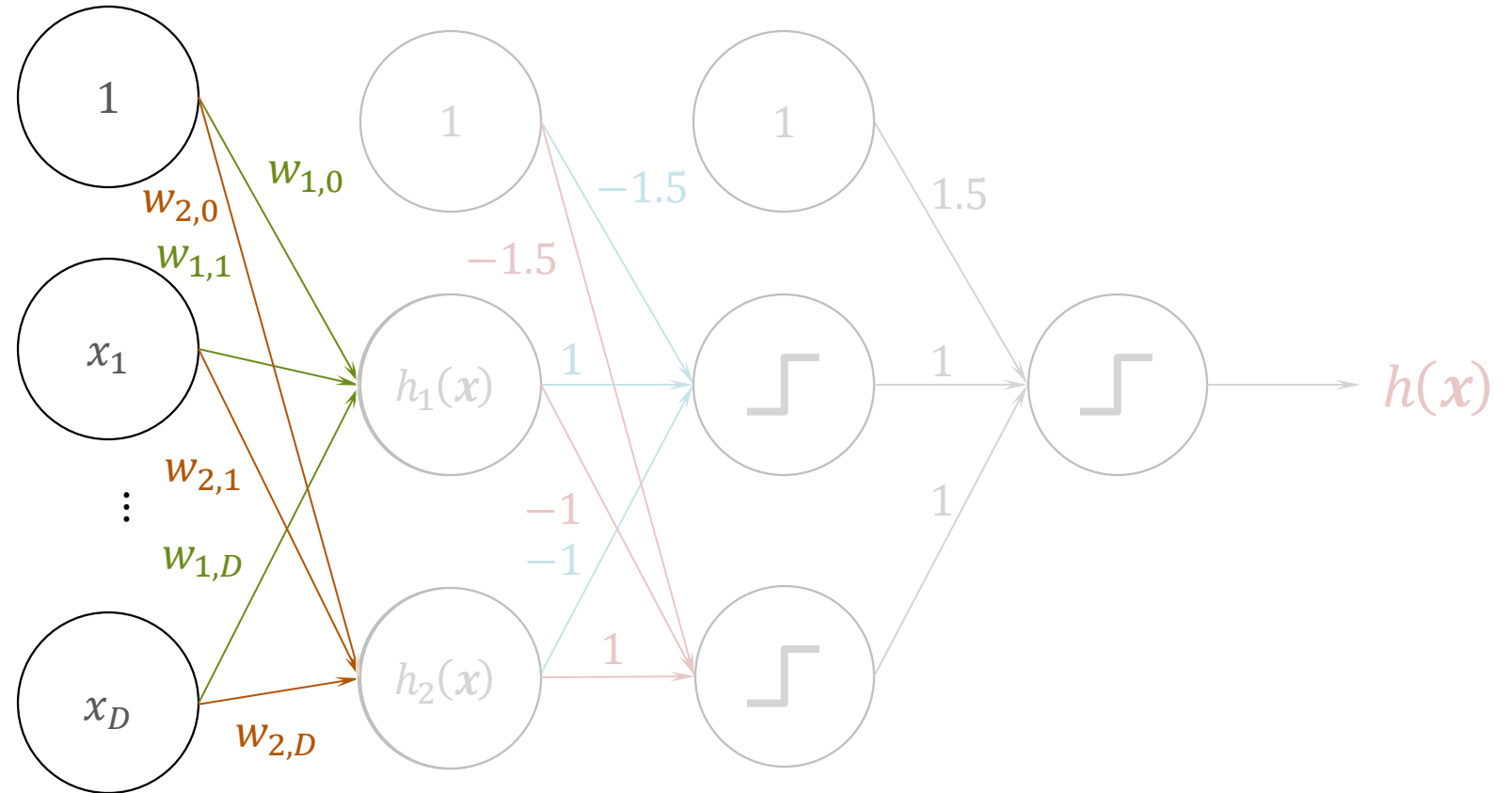
$$h(\mathbf{x}) = OR \left(AND(h_1(\mathbf{x}), \neg h_2(\mathbf{x})), AND(\neg h_1(\mathbf{x}), h_2(\mathbf{x})) \right)$$



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

Building a Network

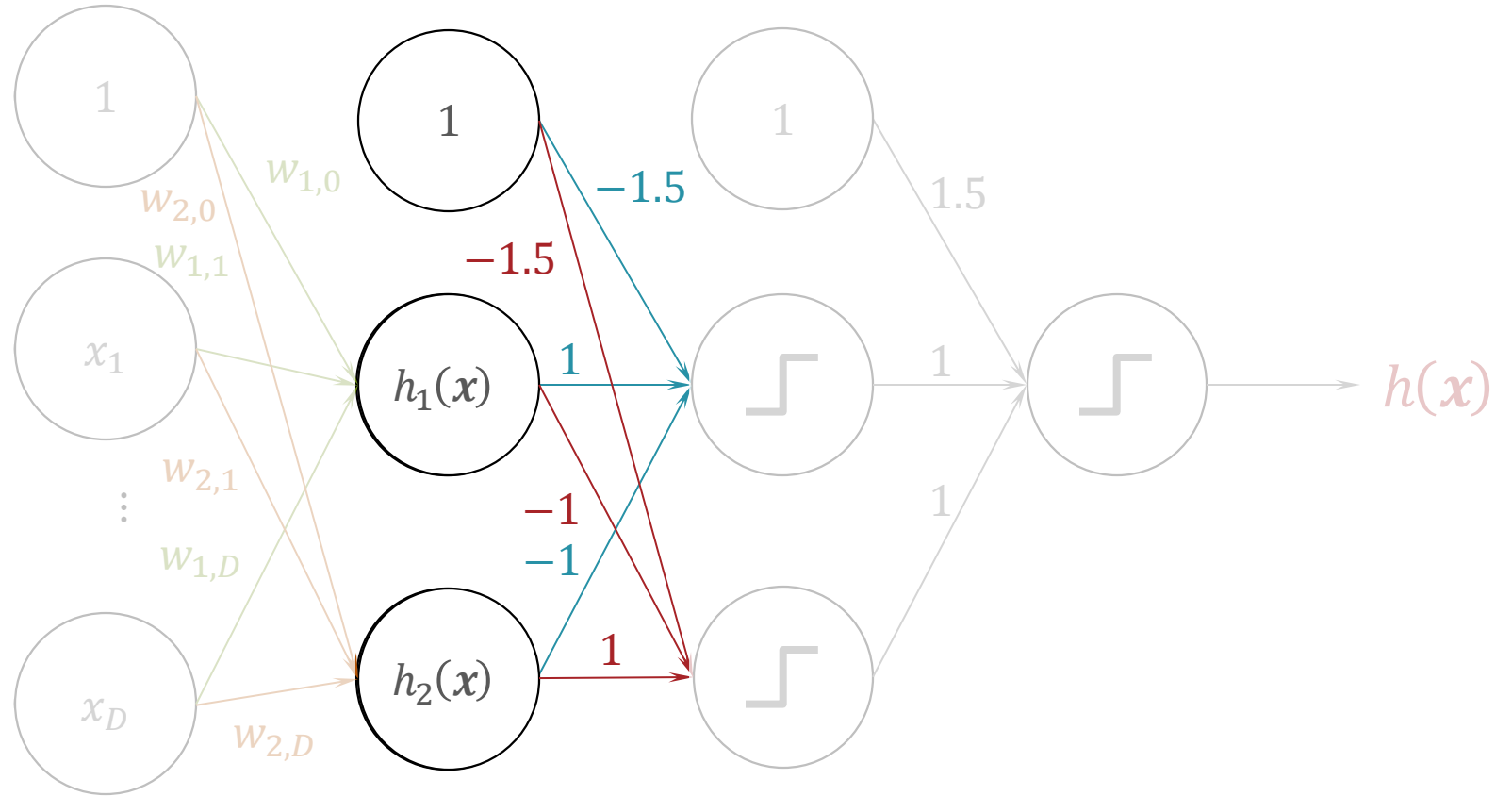
$$h(\mathbf{x}) = OR \left(AND(h_1(\mathbf{x}), \neg h_2(\mathbf{x})), AND(\neg h_1(\mathbf{x}), h_2(\mathbf{x})) \right)$$



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

Building a Network

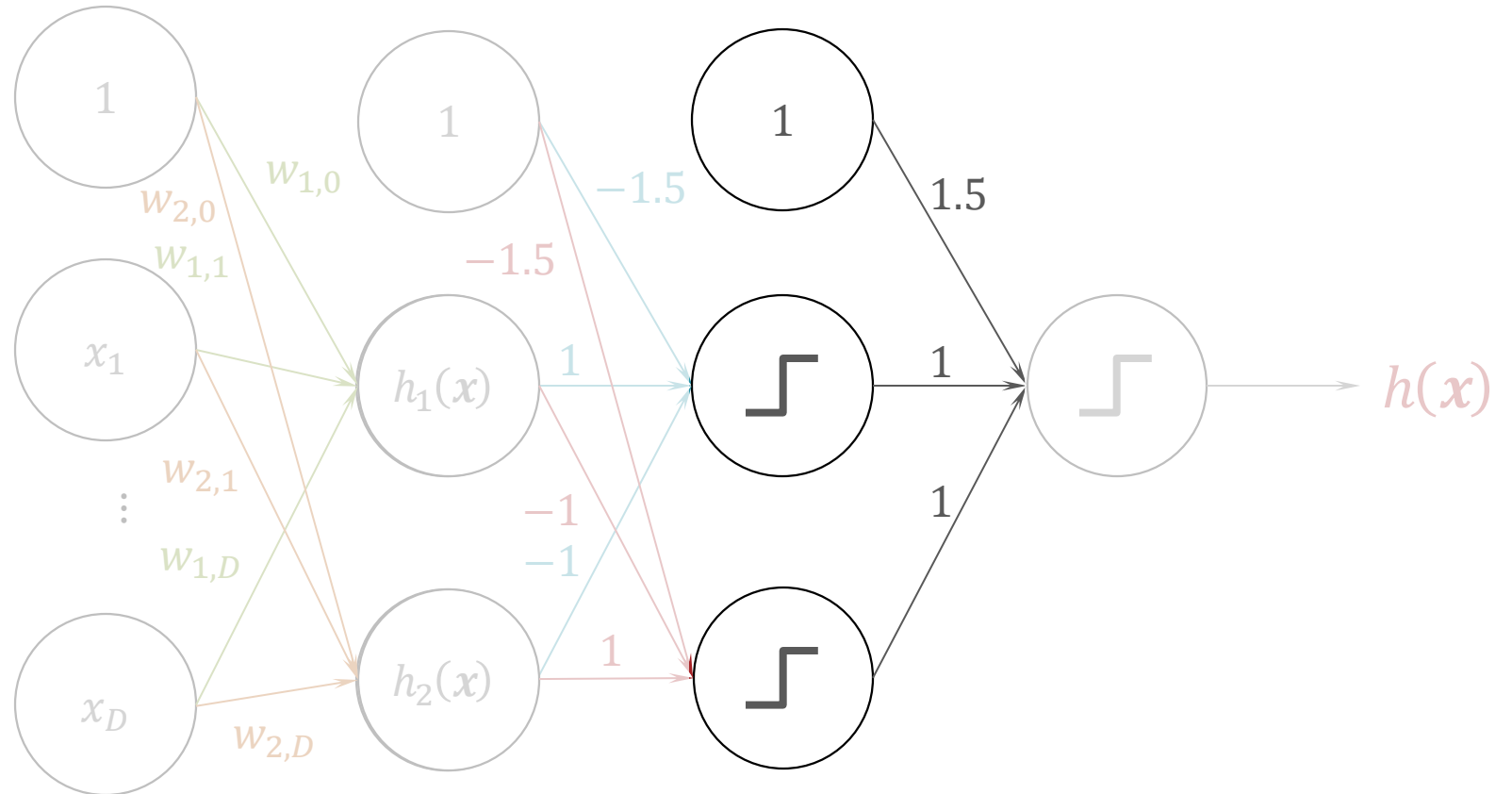
$$h(\mathbf{x}) = OR \left(AND(h_1(\mathbf{x}), \neg h_2(\mathbf{x})), AND(\neg h_1(\mathbf{x}), h_2(\mathbf{x})) \right)$$



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

Building a Network

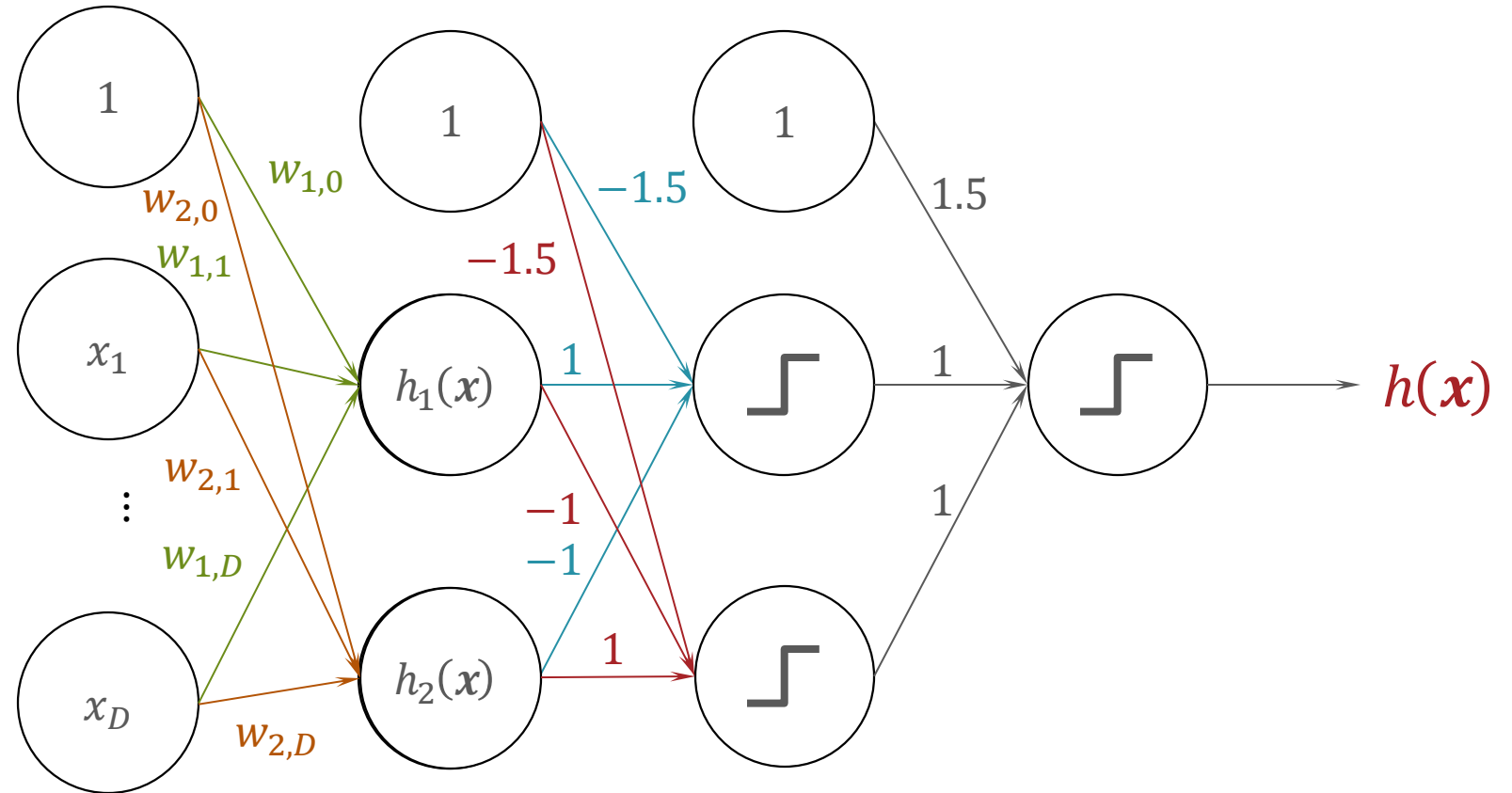
$$h(\mathbf{x}) = \text{OR} \left(\text{AND}(h_1(\mathbf{x}), \neg h_2(\mathbf{x})), \text{AND}(\neg h_1(\mathbf{x}), h_2(\mathbf{x})) \right)$$



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

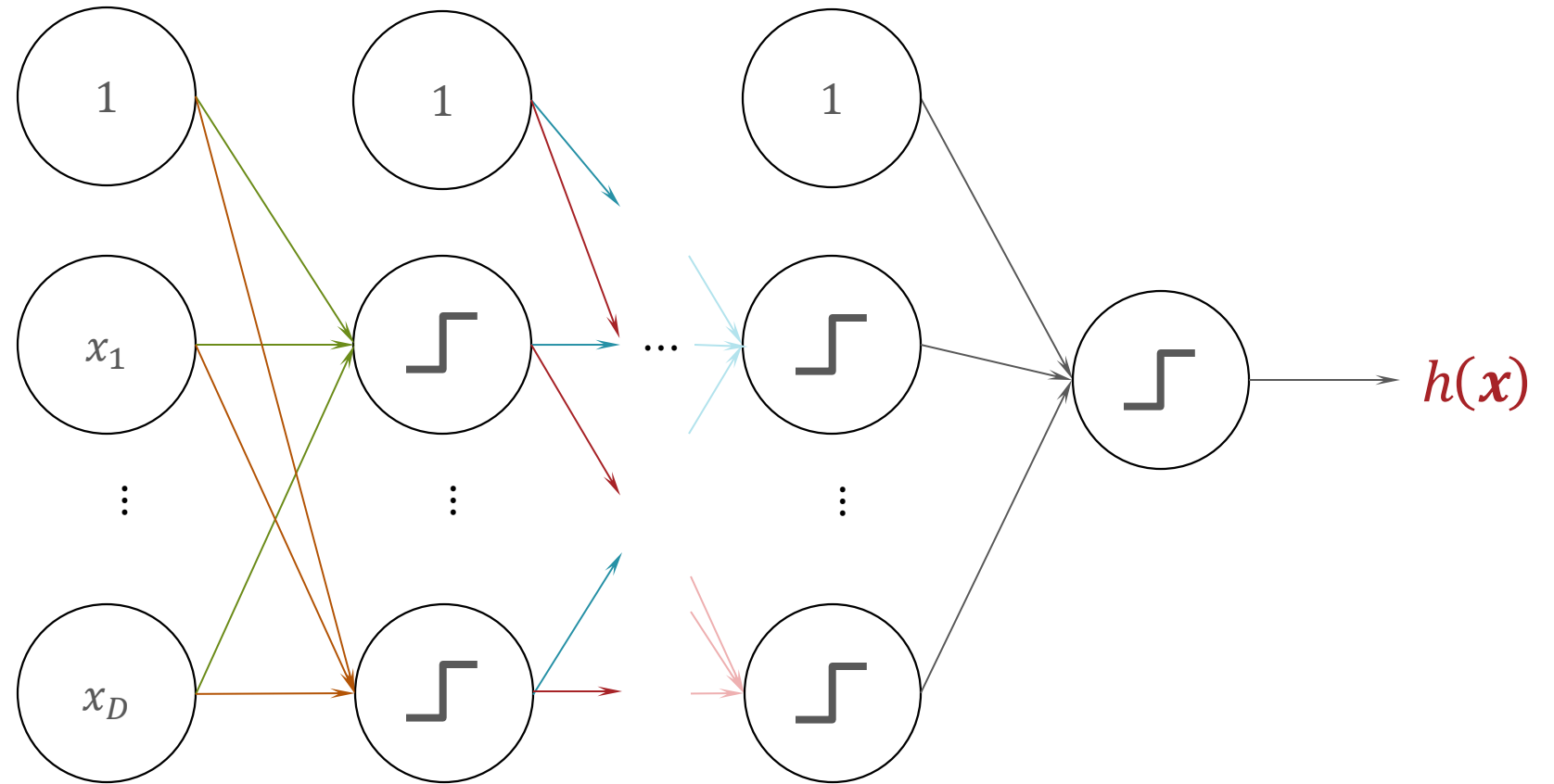
Building a Network

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

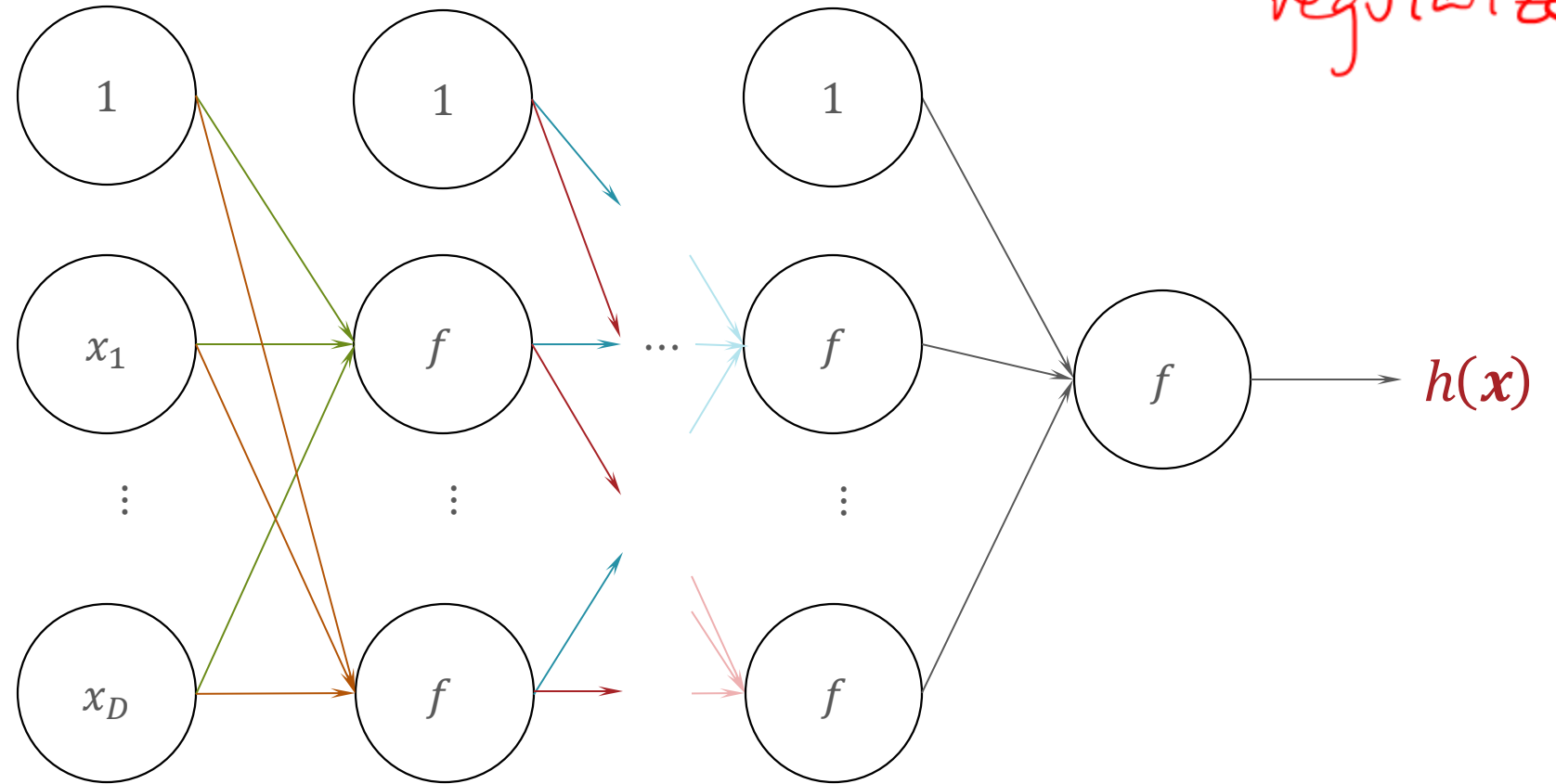


$$h(\mathbf{x}) = \text{sign}(\text{sign}(\text{sign}(\mathbf{w}_1^T \mathbf{x}) - \text{sign}(\mathbf{w}_2^T \mathbf{x}) - 1.5) + \text{sign}(-\text{sign}(\mathbf{w}_1^T \mathbf{x}) + \text{sign}(\mathbf{w}_2^T \mathbf{x}) - 1.5) + 1.5)$$

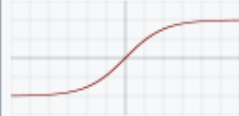
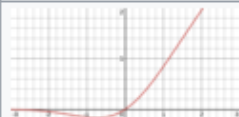
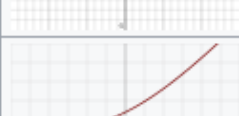
Multi-Layer Perceptron (MLP)



(Fully-Connected) Feed Forward Neural Network



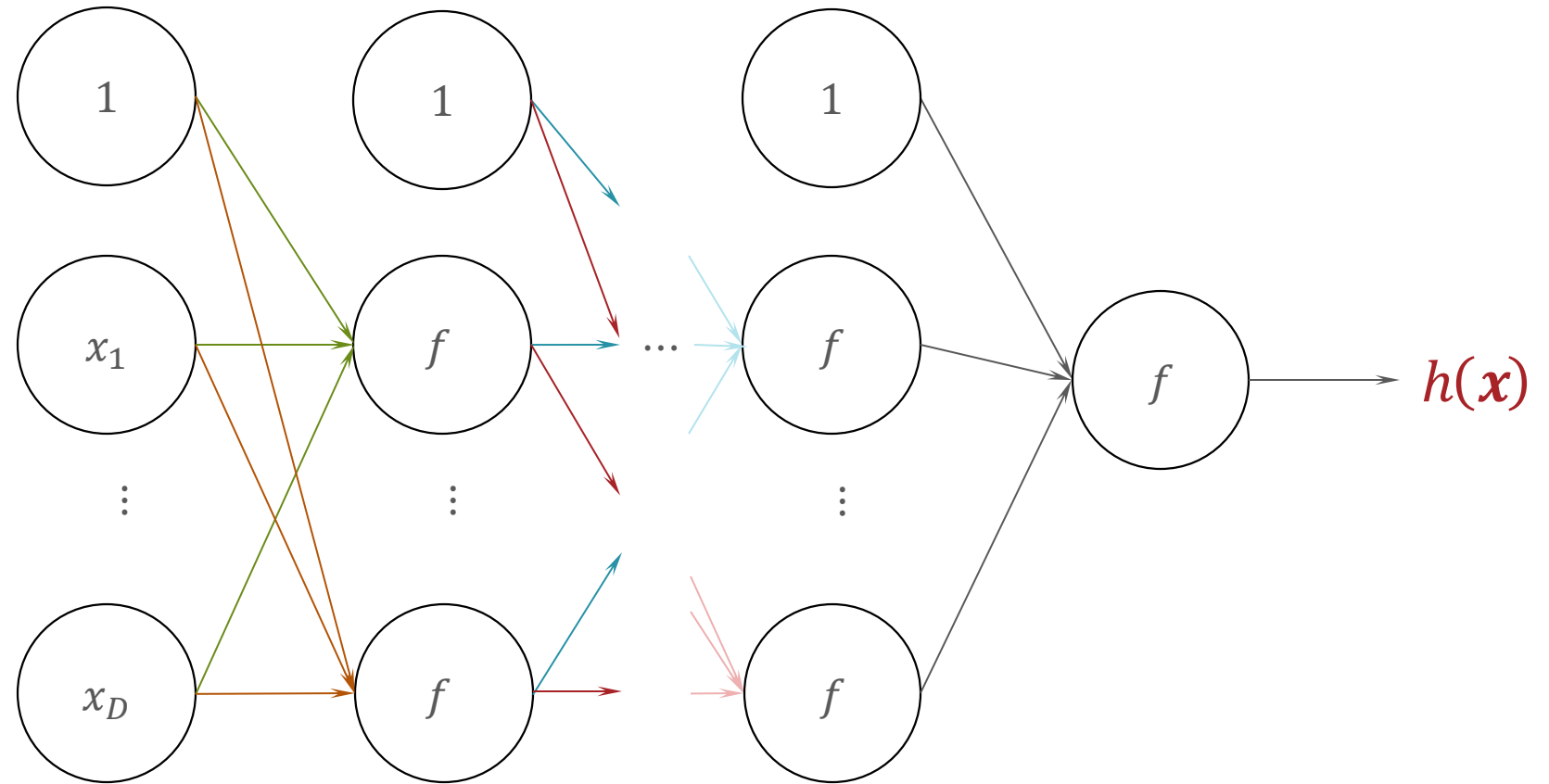
Activation Functions

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]		$\begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ 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

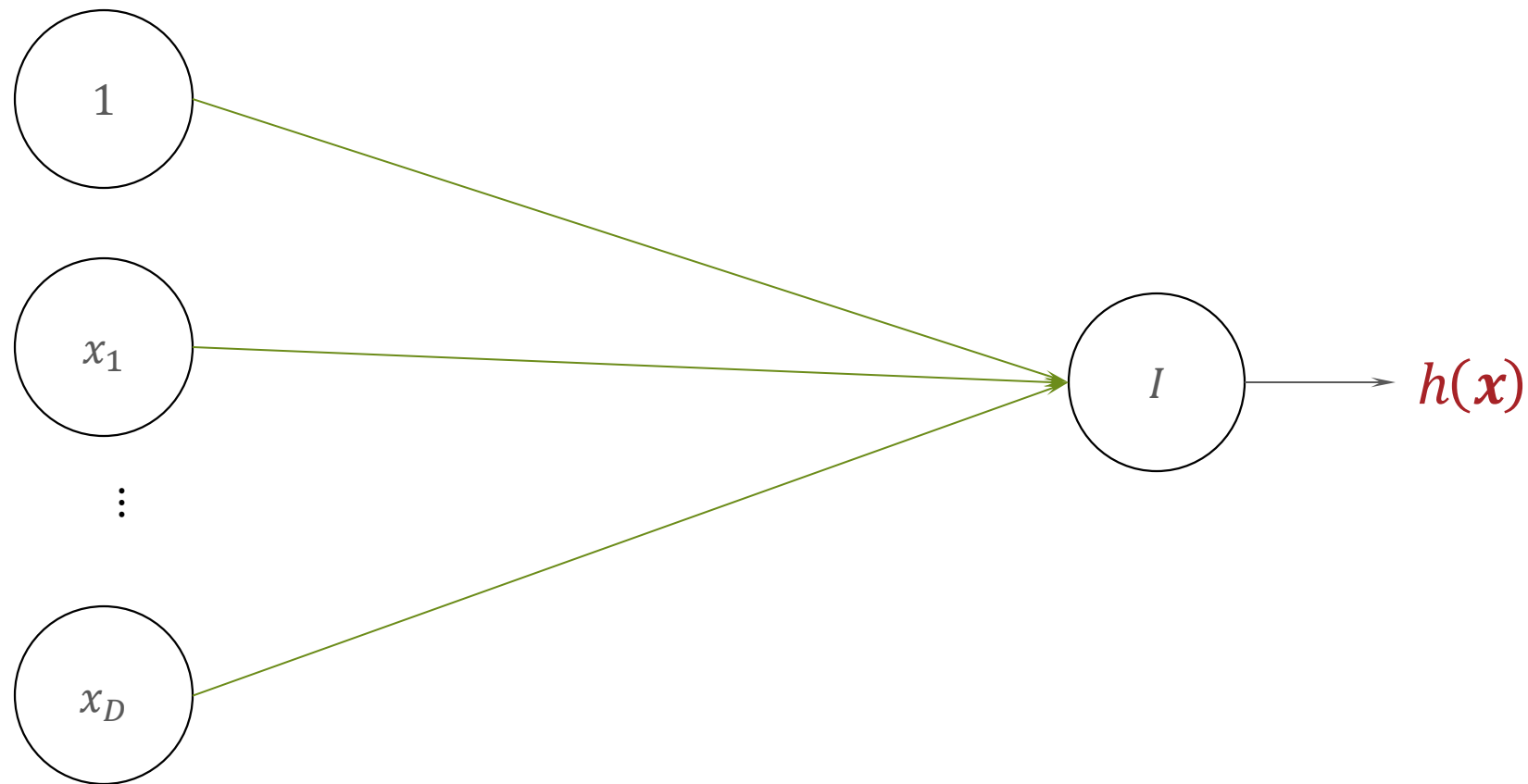
Poll Question 2

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

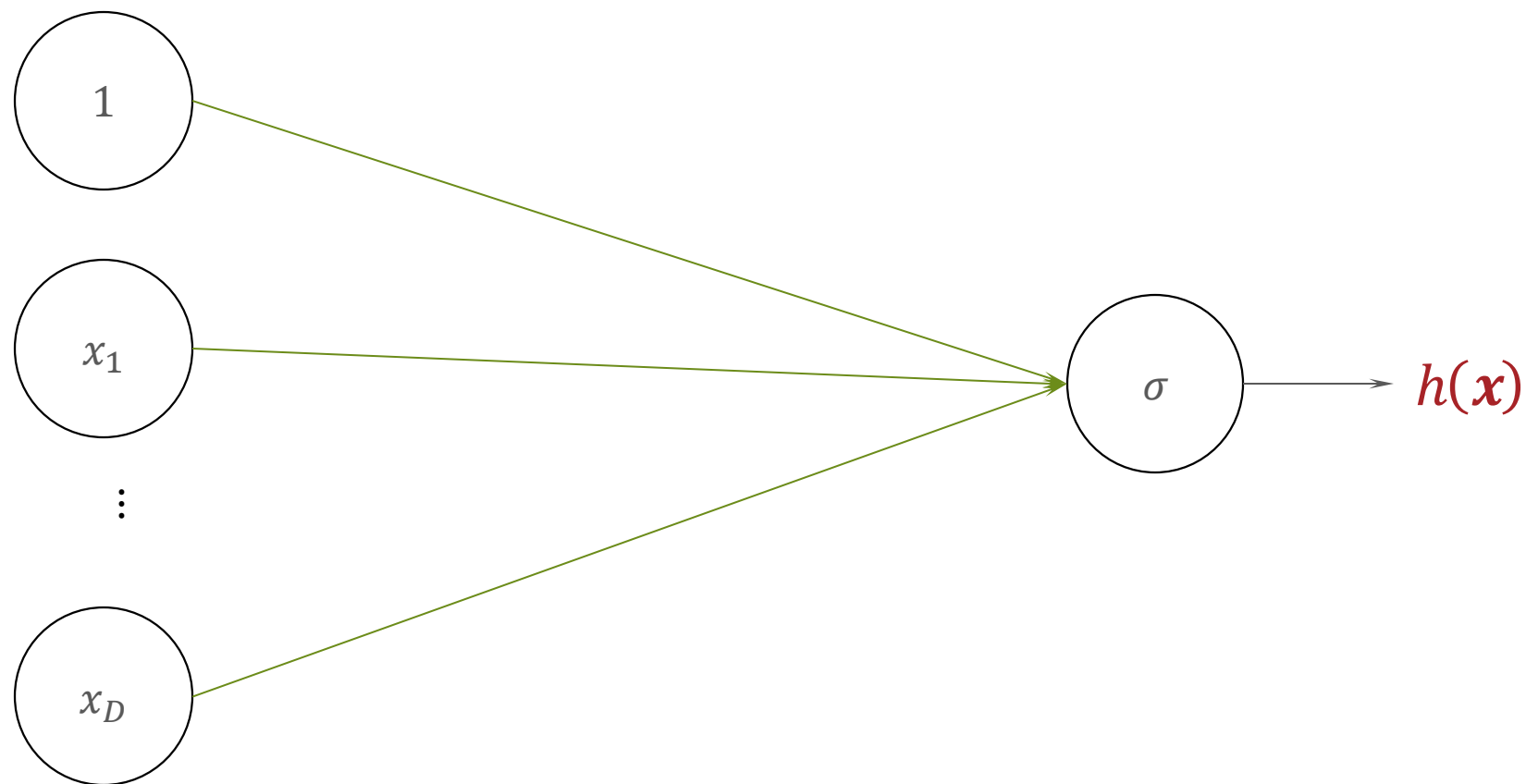
- A. Only true for linear regression
- B. Only true for logistic regression
- C. TOXIC
- ☒ D. True for both
- E. False for both



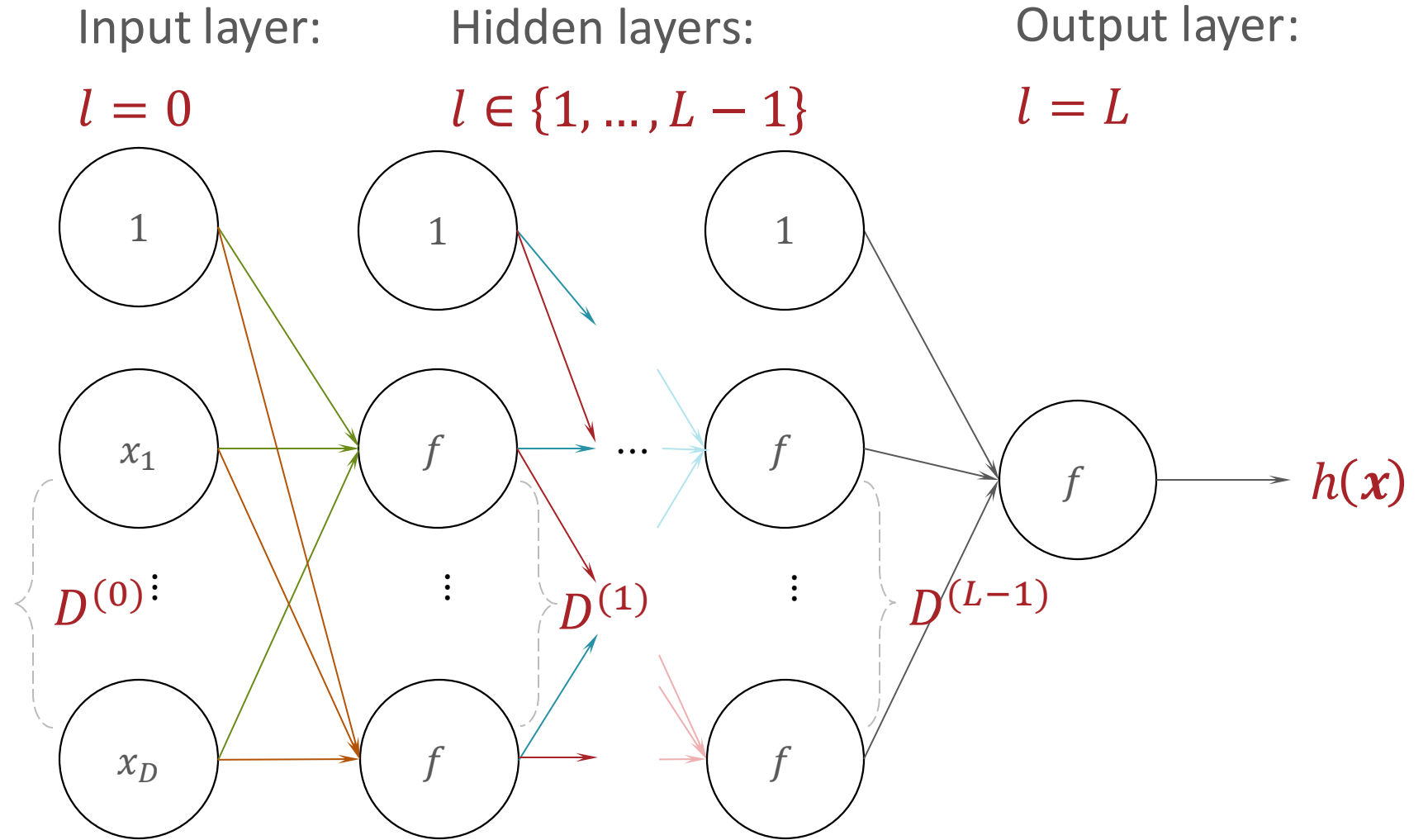
Linear Regression as a Neural Network



Logistic Regression as a Neural Network



(Fully-Connected) Feed Forward Neural Network

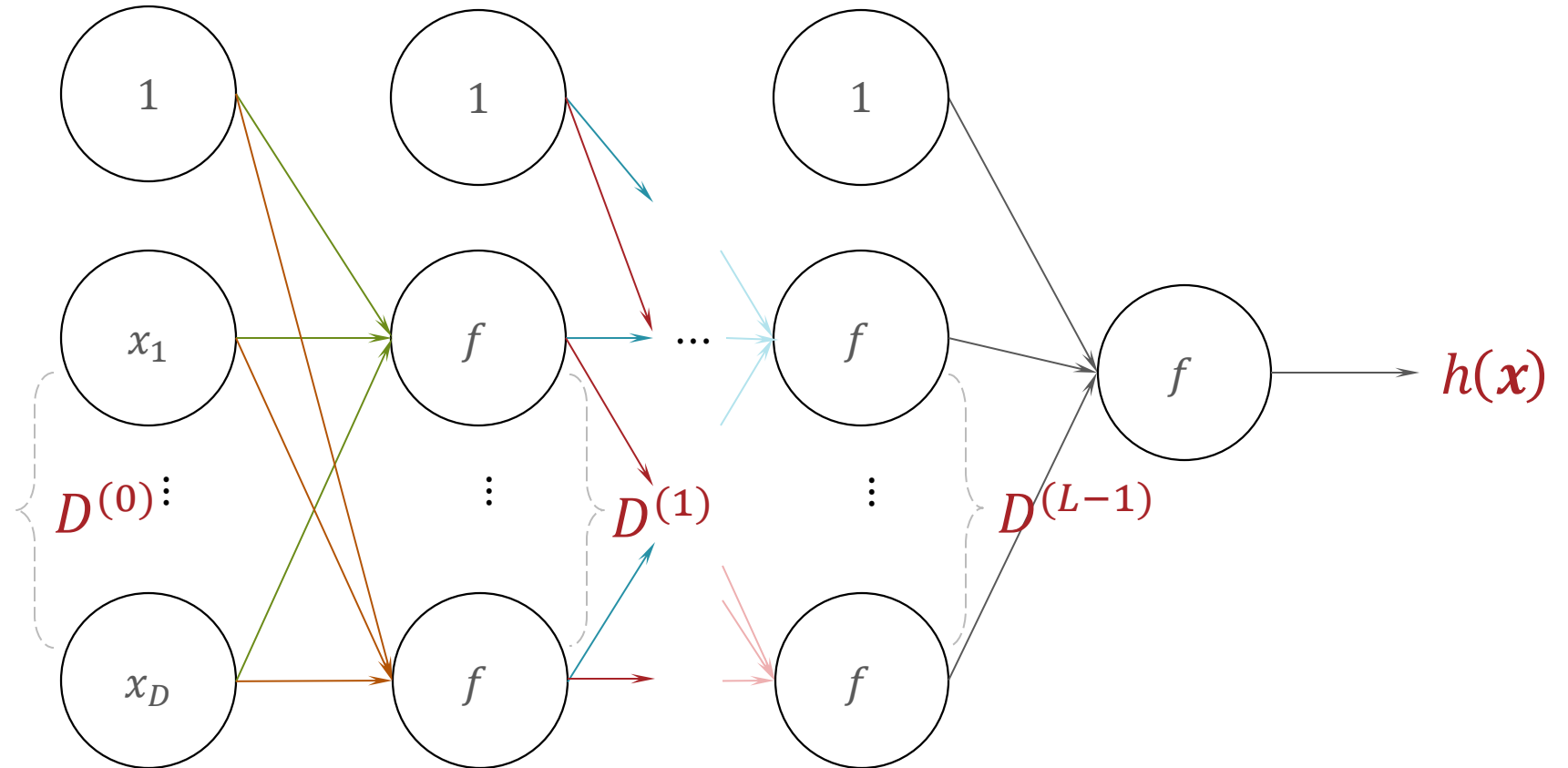


Layer l has dimension $D^{(l)}$ $\rightarrow$ Layer l has $D^{(l)} + 1$ nodes, counting the bias node

(Fully-Connected) Feed Forward Neural Network

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

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

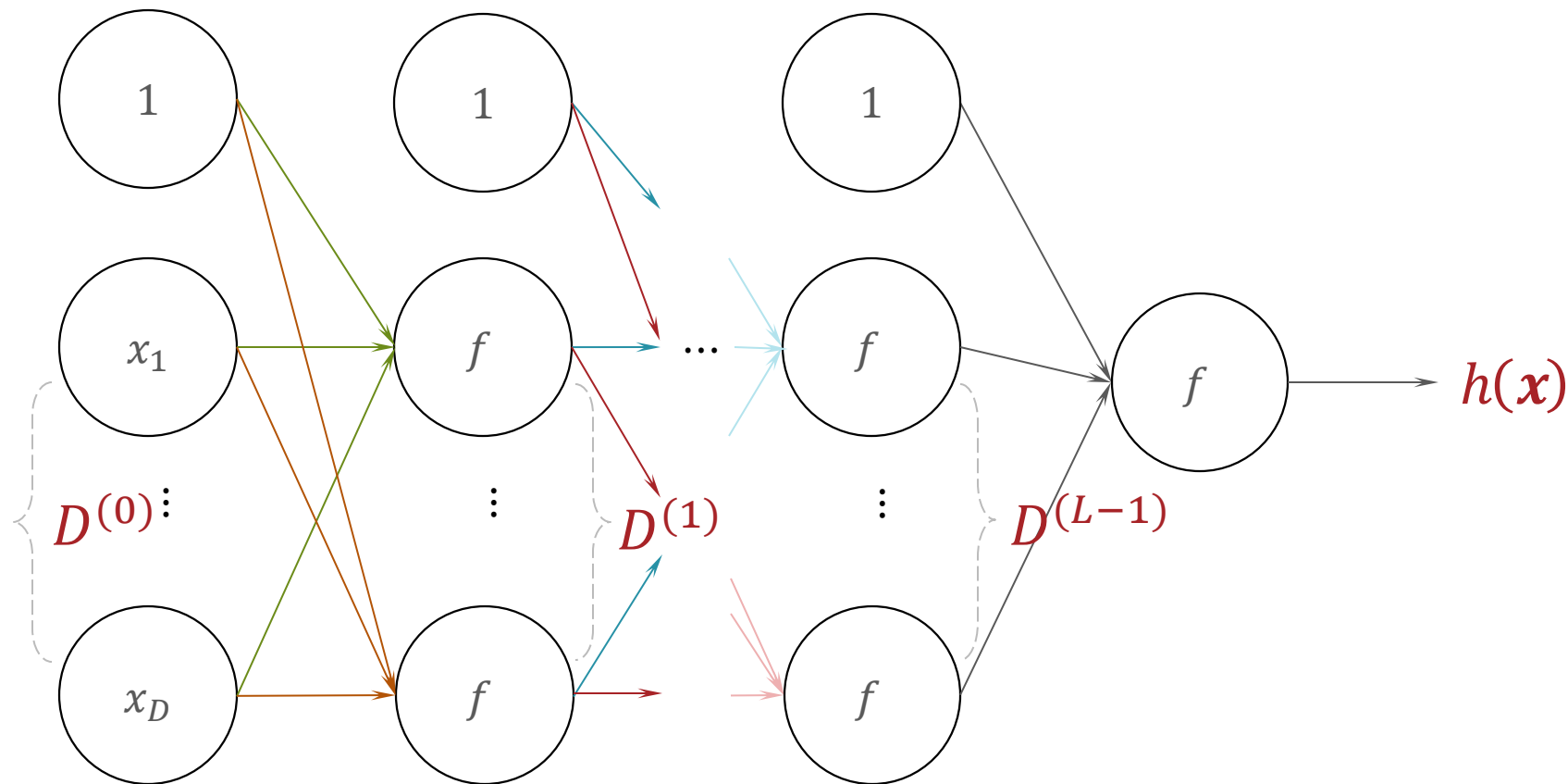


$w_{j,i}^{(l)}$ is the weight between node i in layer $l - 1$ and node j in layer l

So what are all these layers doing for us anyway?

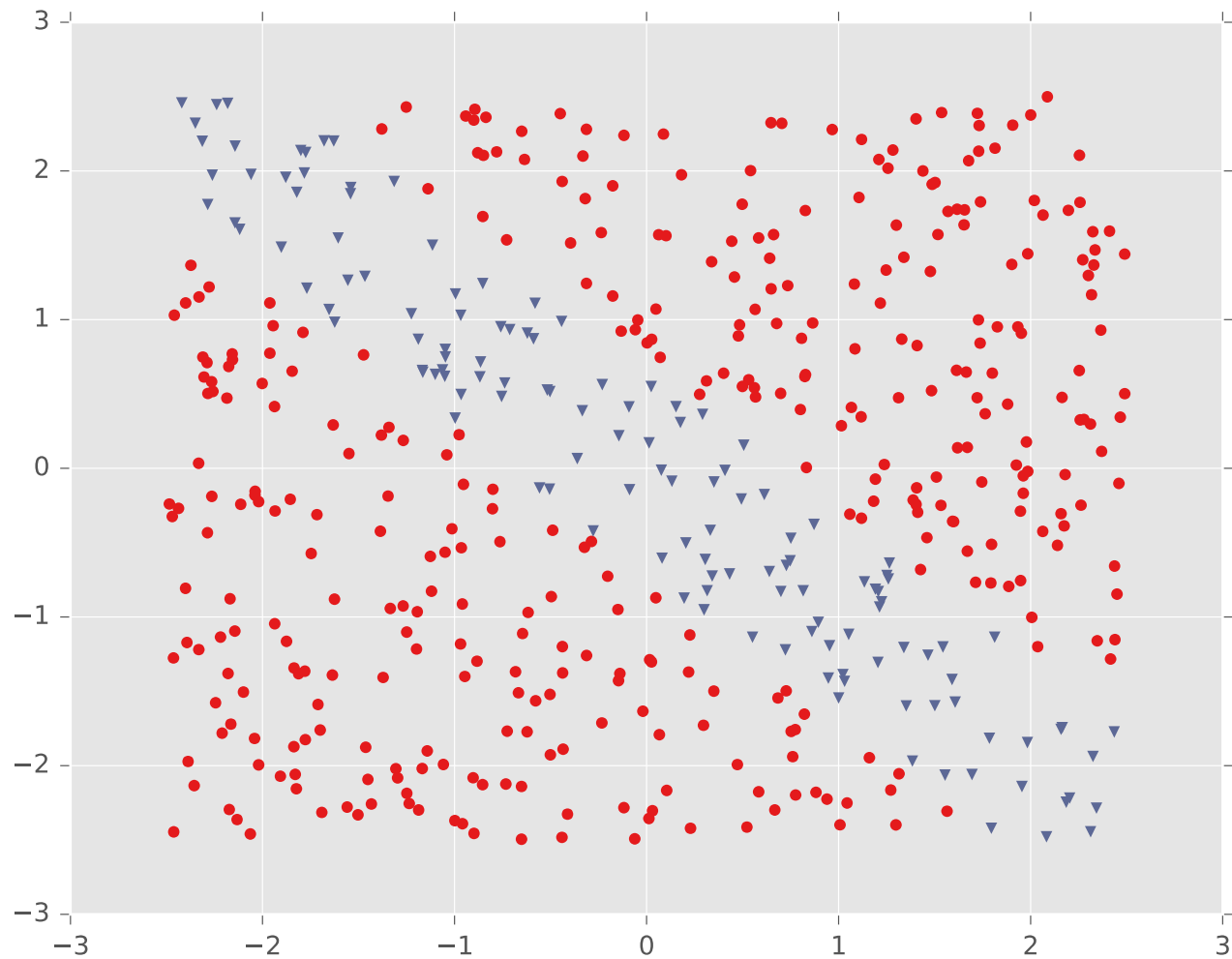
The weights between layer $l - 1$ and layer l are a matrix:

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



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

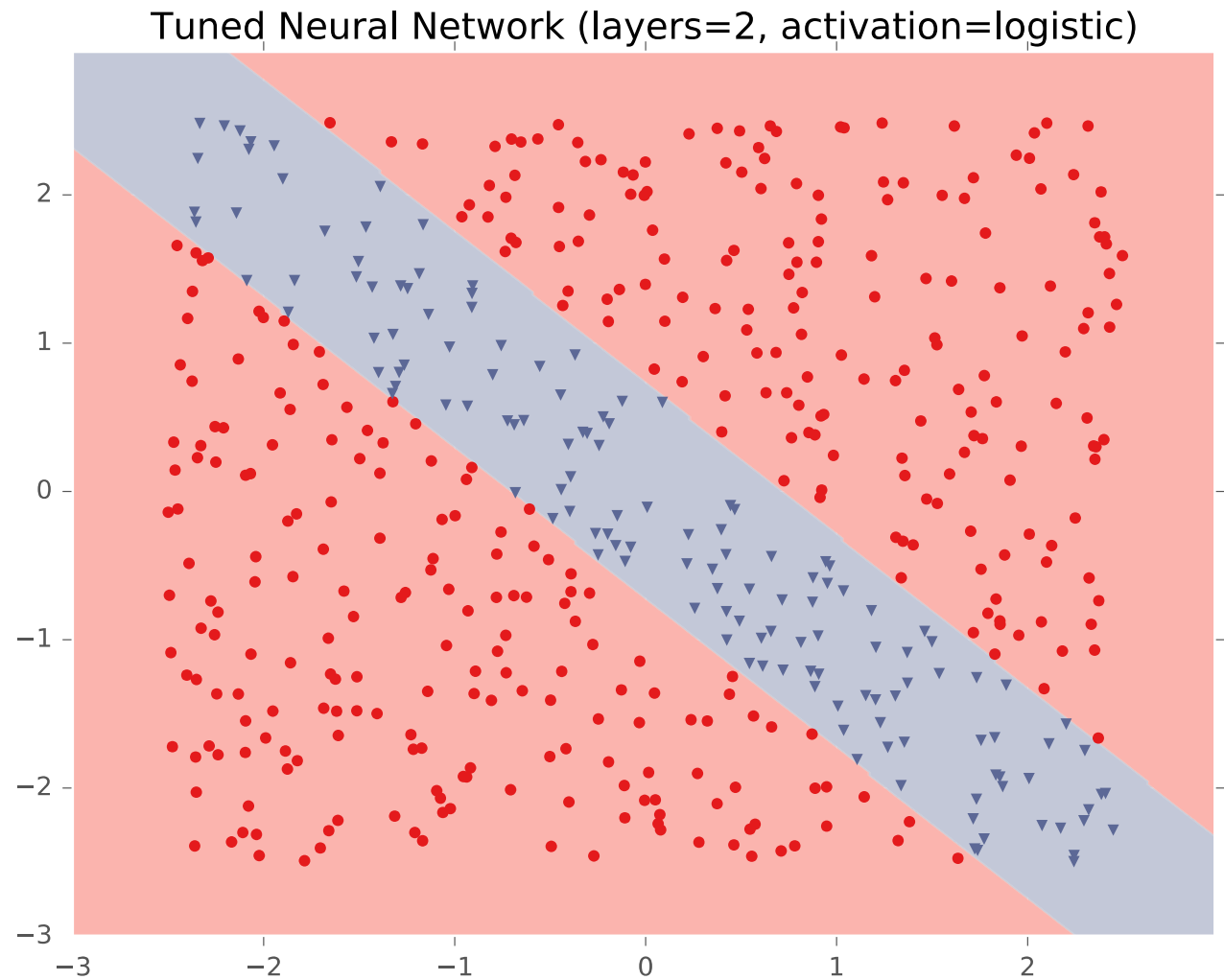
Neural Network Decision Boundaries: Example 1



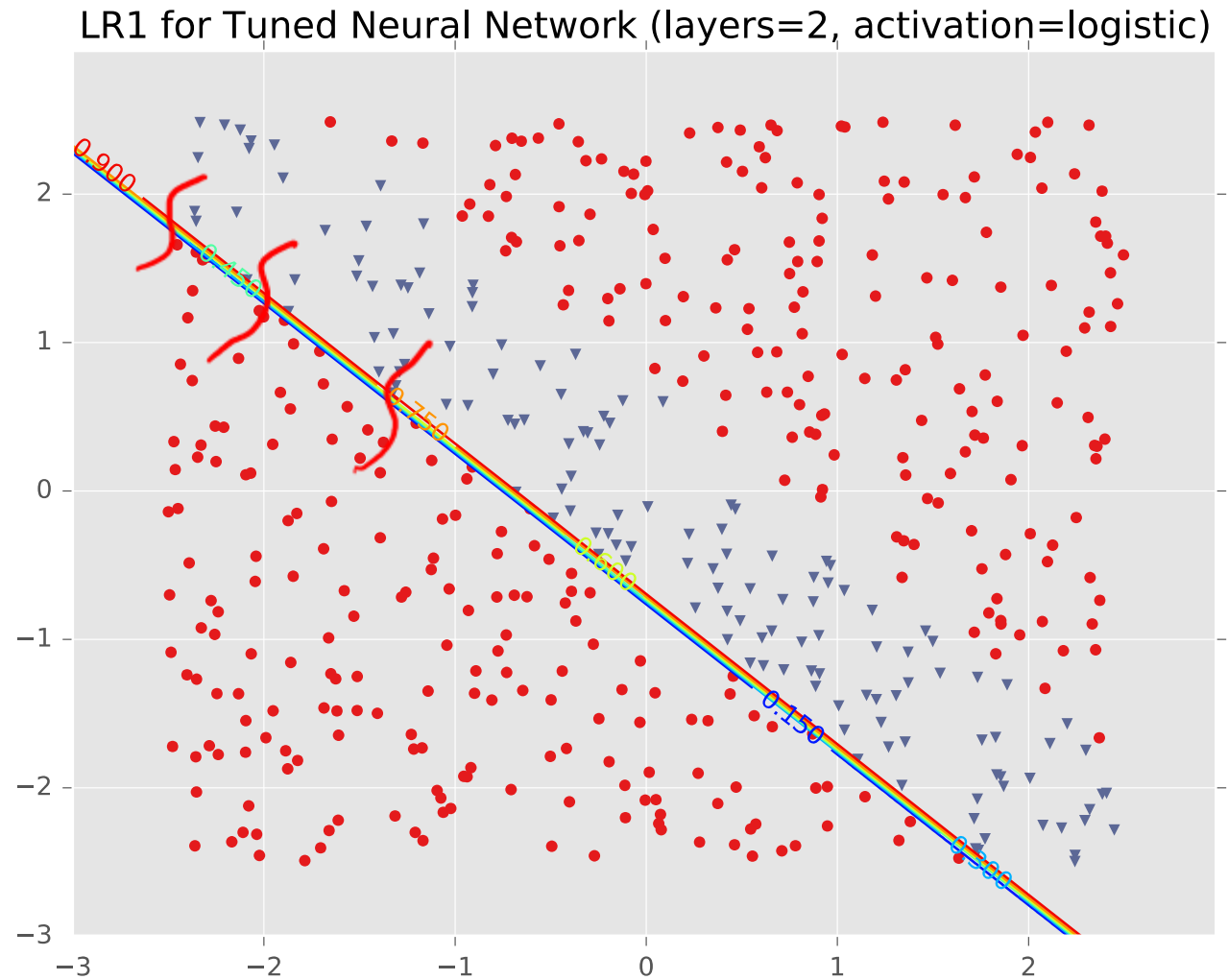
Neural Network Decision Boundaries: Example 1



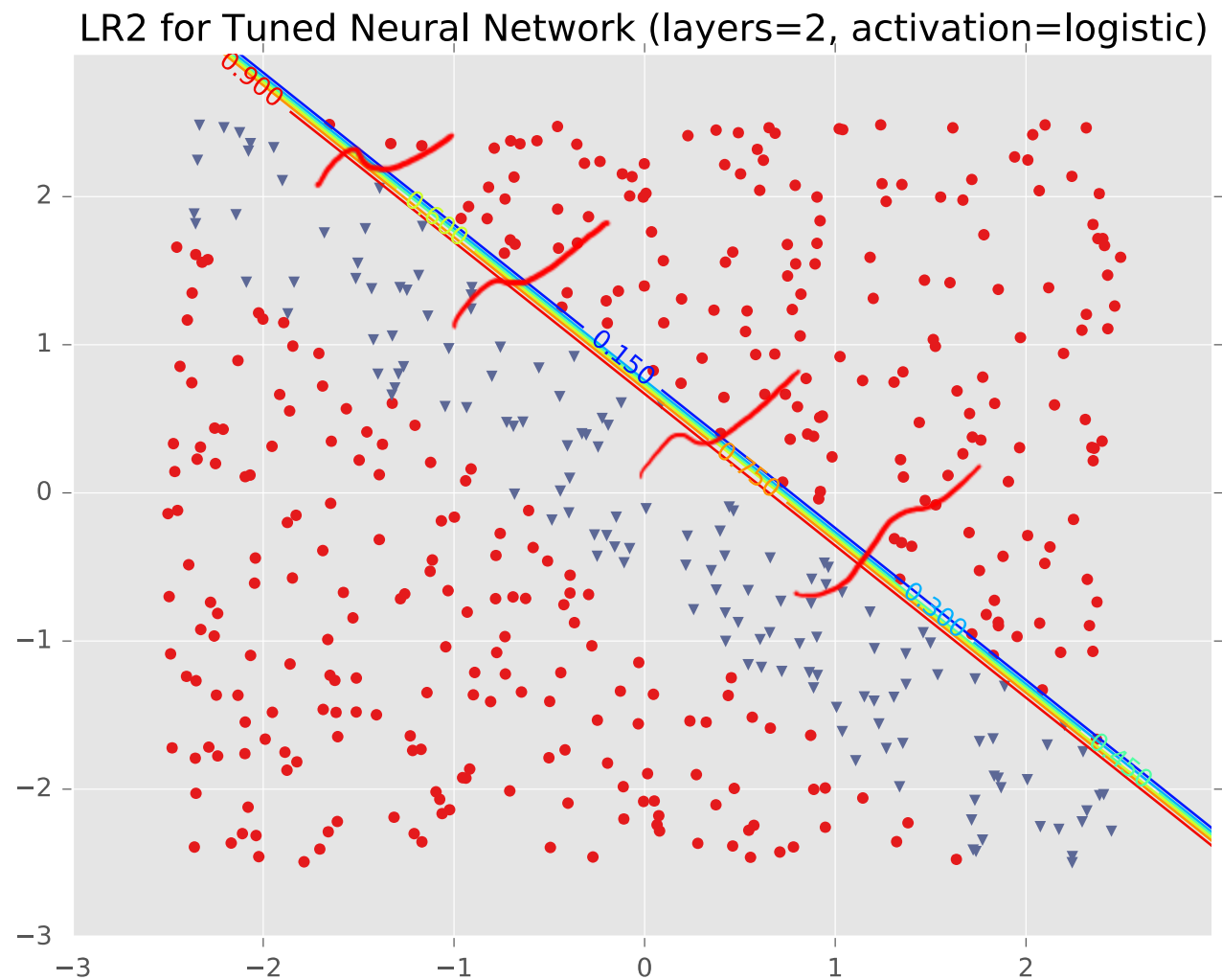
Neural Network Decision Boundaries: Example 1



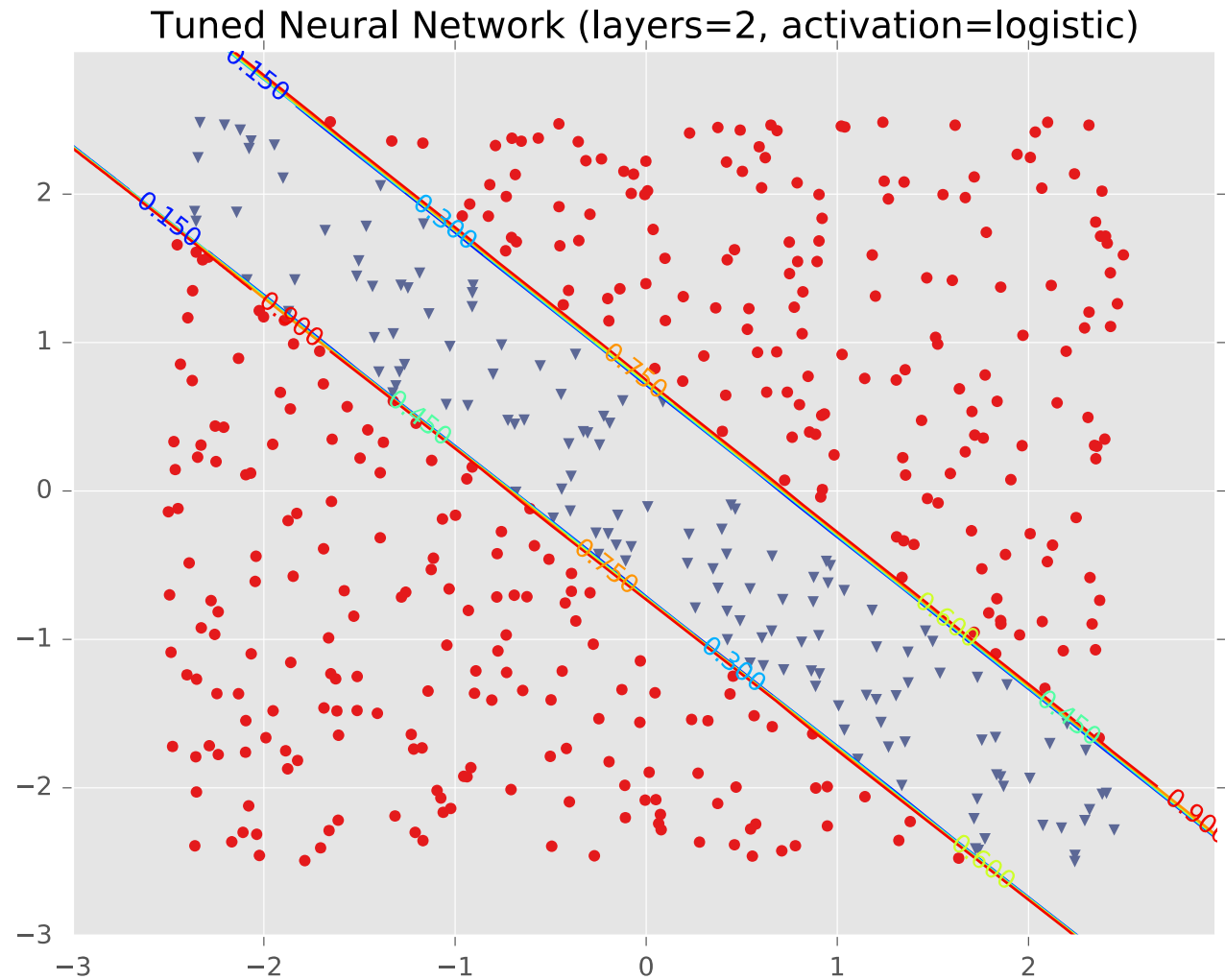
Neural Network Decision Boundaries: Example 1



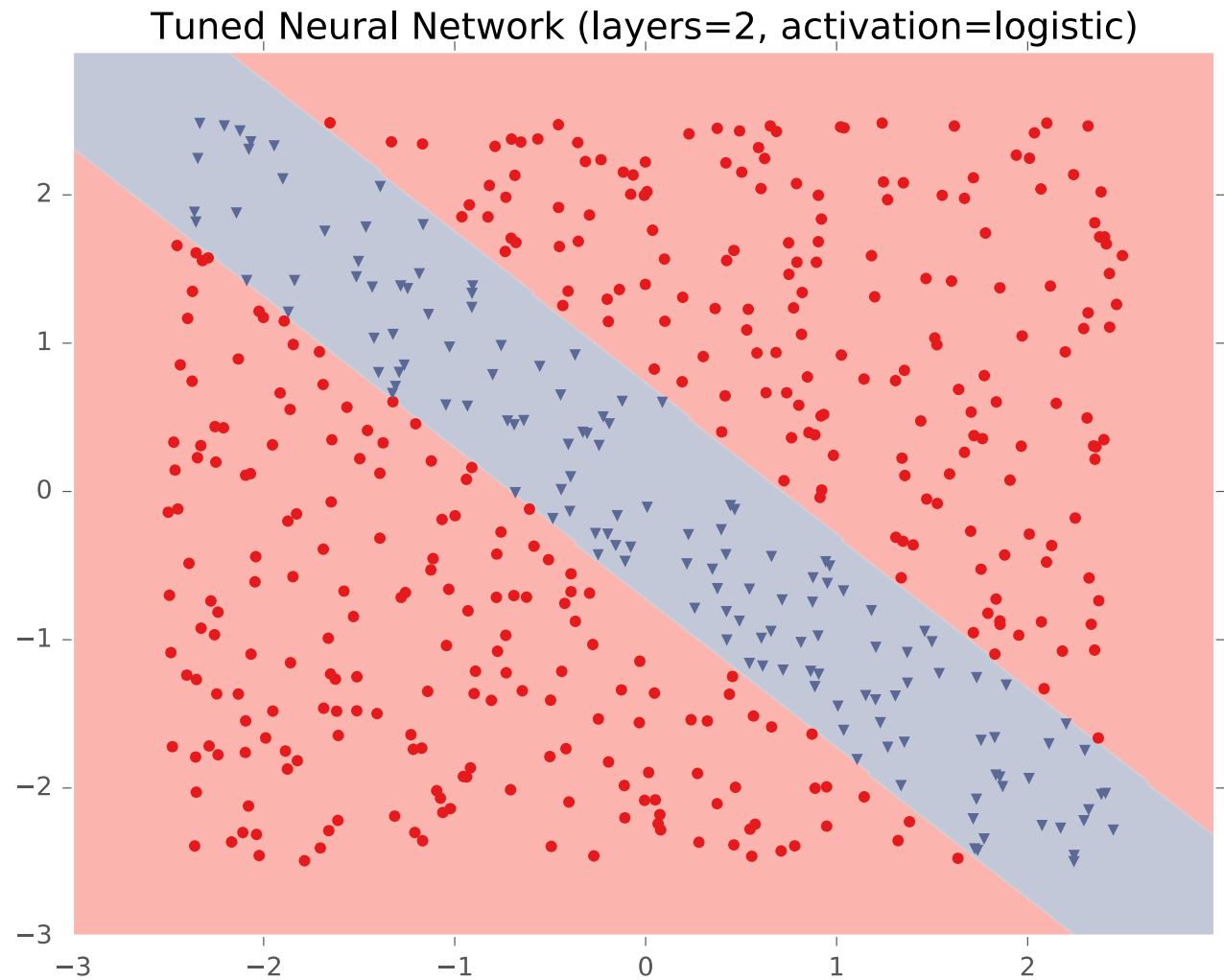
Neural Network Decision Boundaries: Example 1



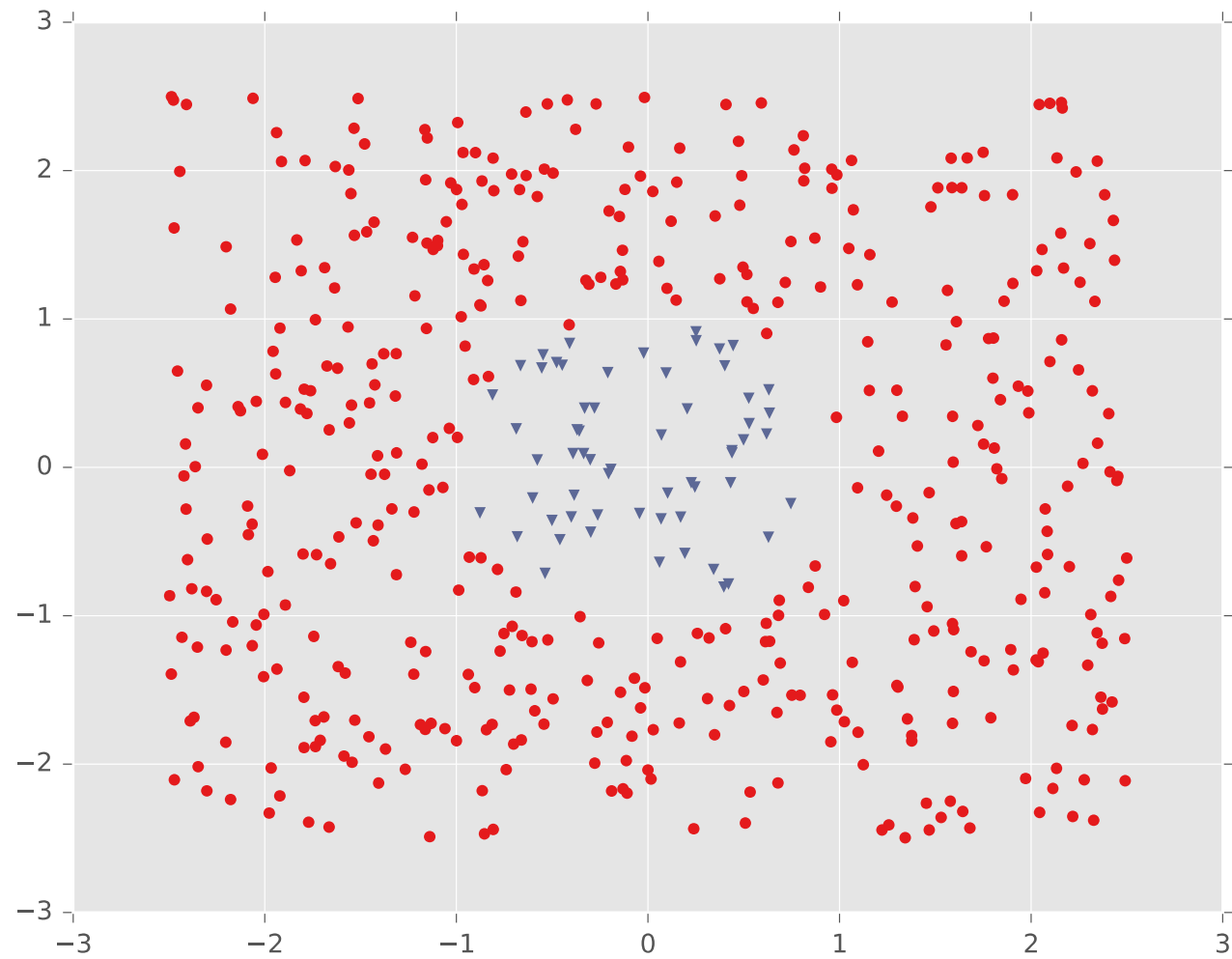
Neural Network Decision Boundaries: Example 1



Neural Network Decision Boundaries: Example 1



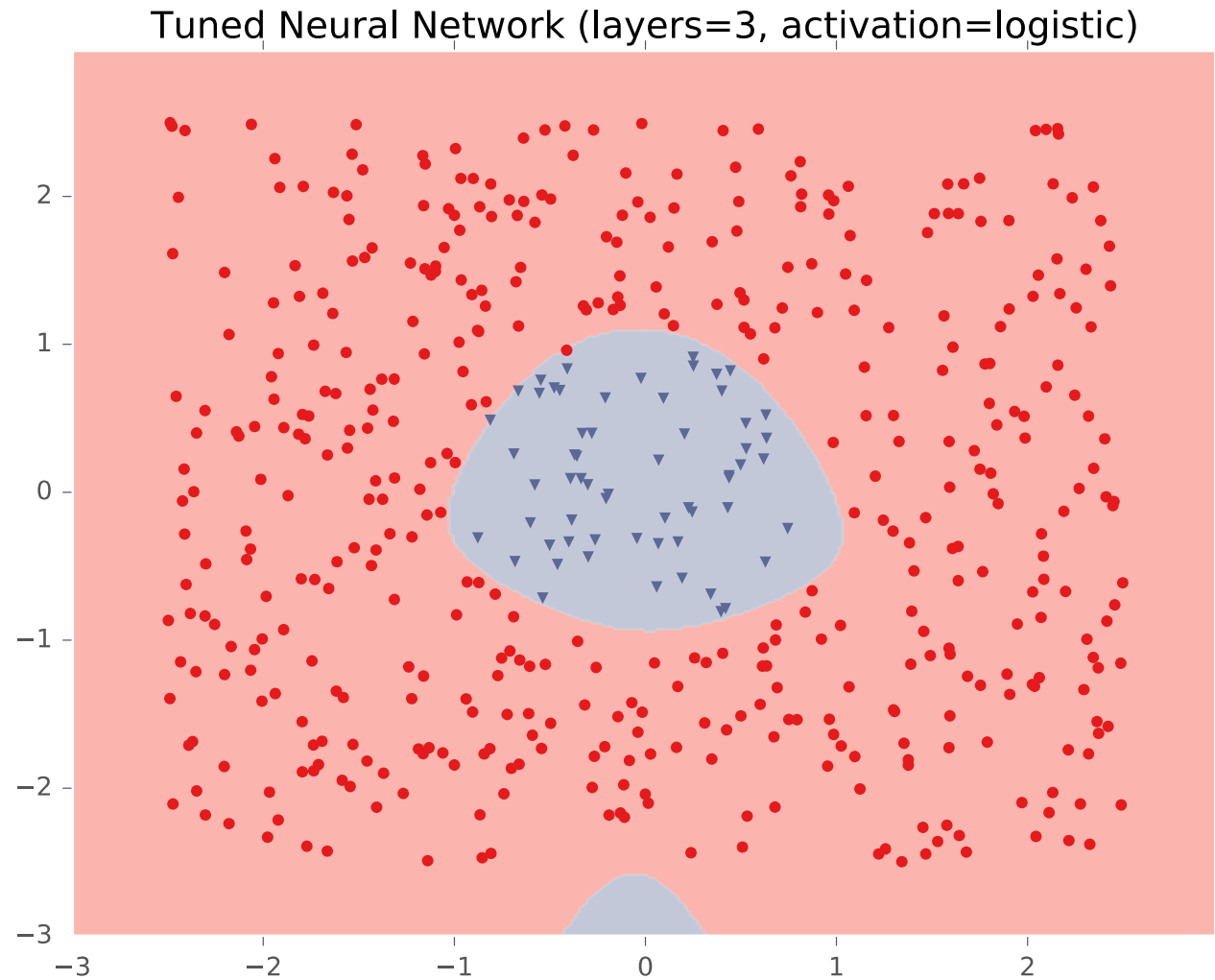
Neural Network Decision Boundaries: Example 2



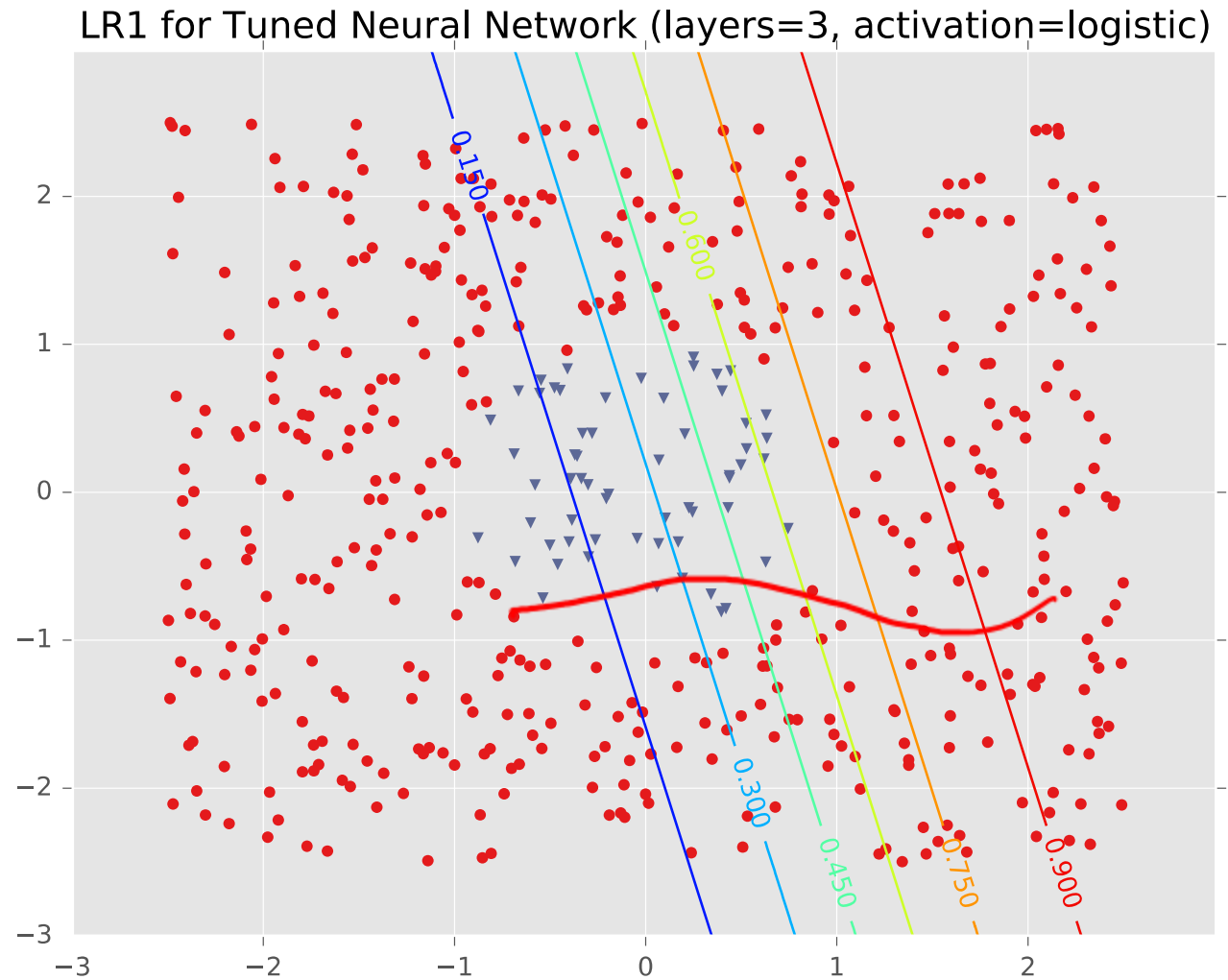
Neural Network Decision Boundaries: Example 2



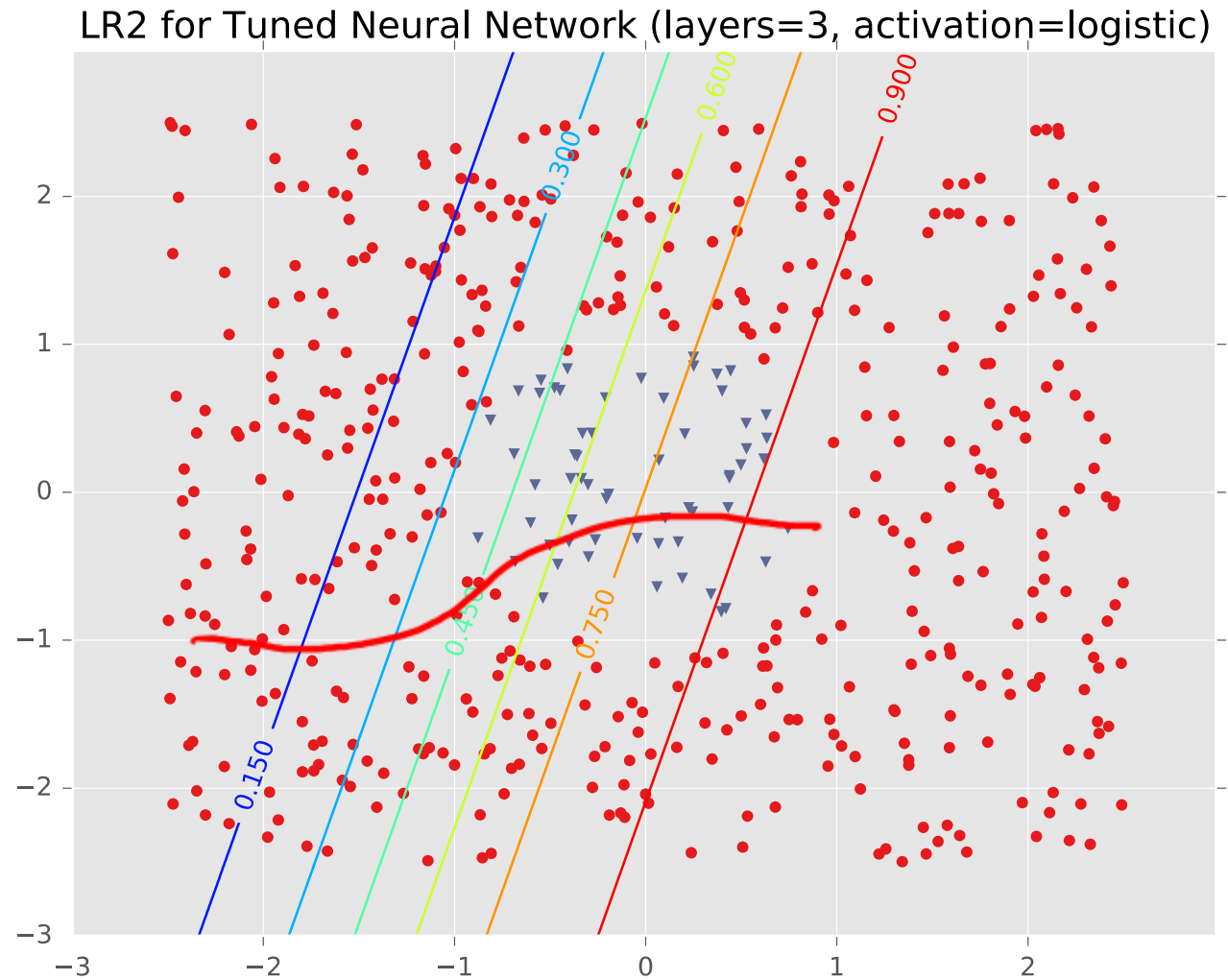
Neural Network Decision Boundaries: Example 2



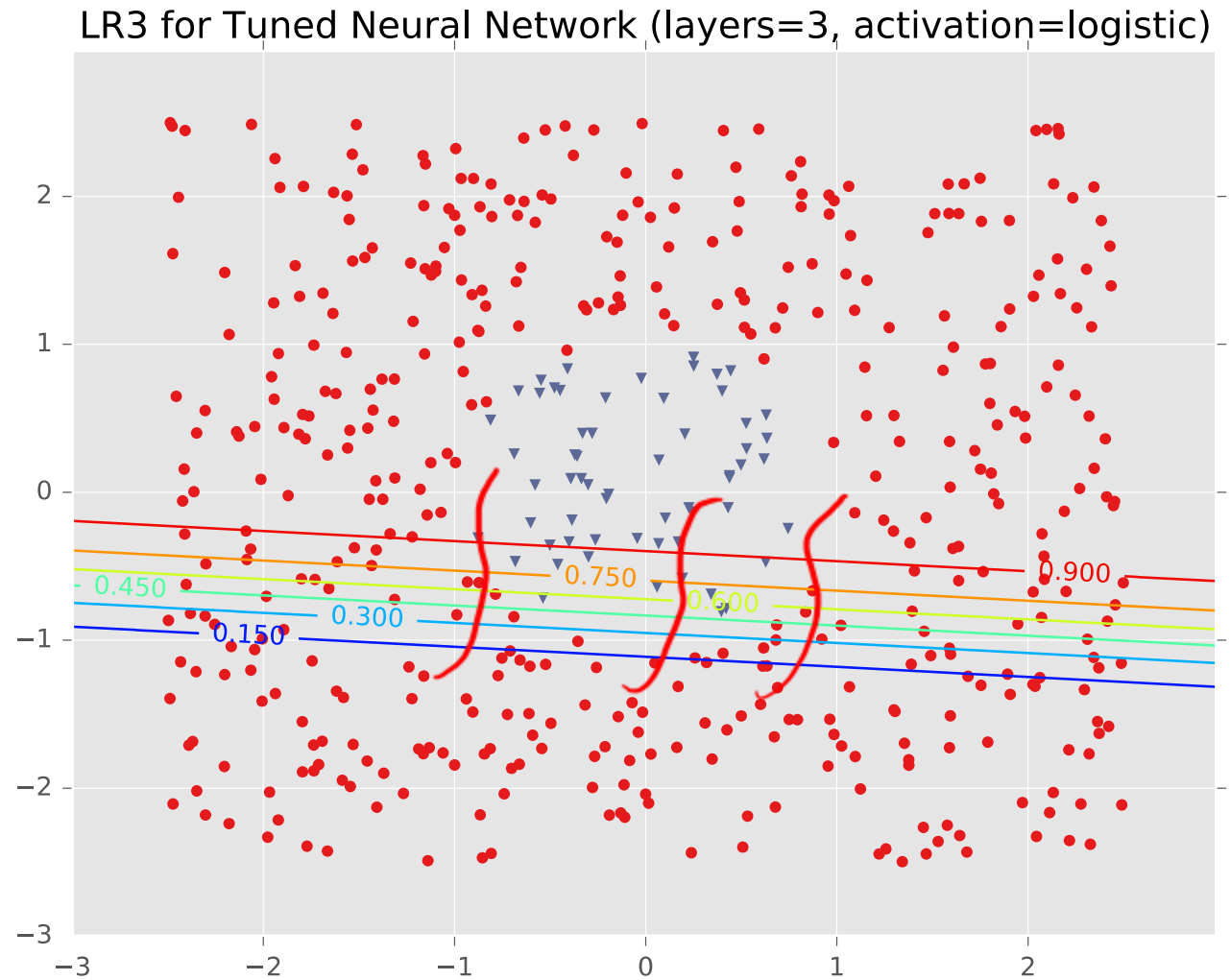
Neural Network Decision Boundaries: Example 2



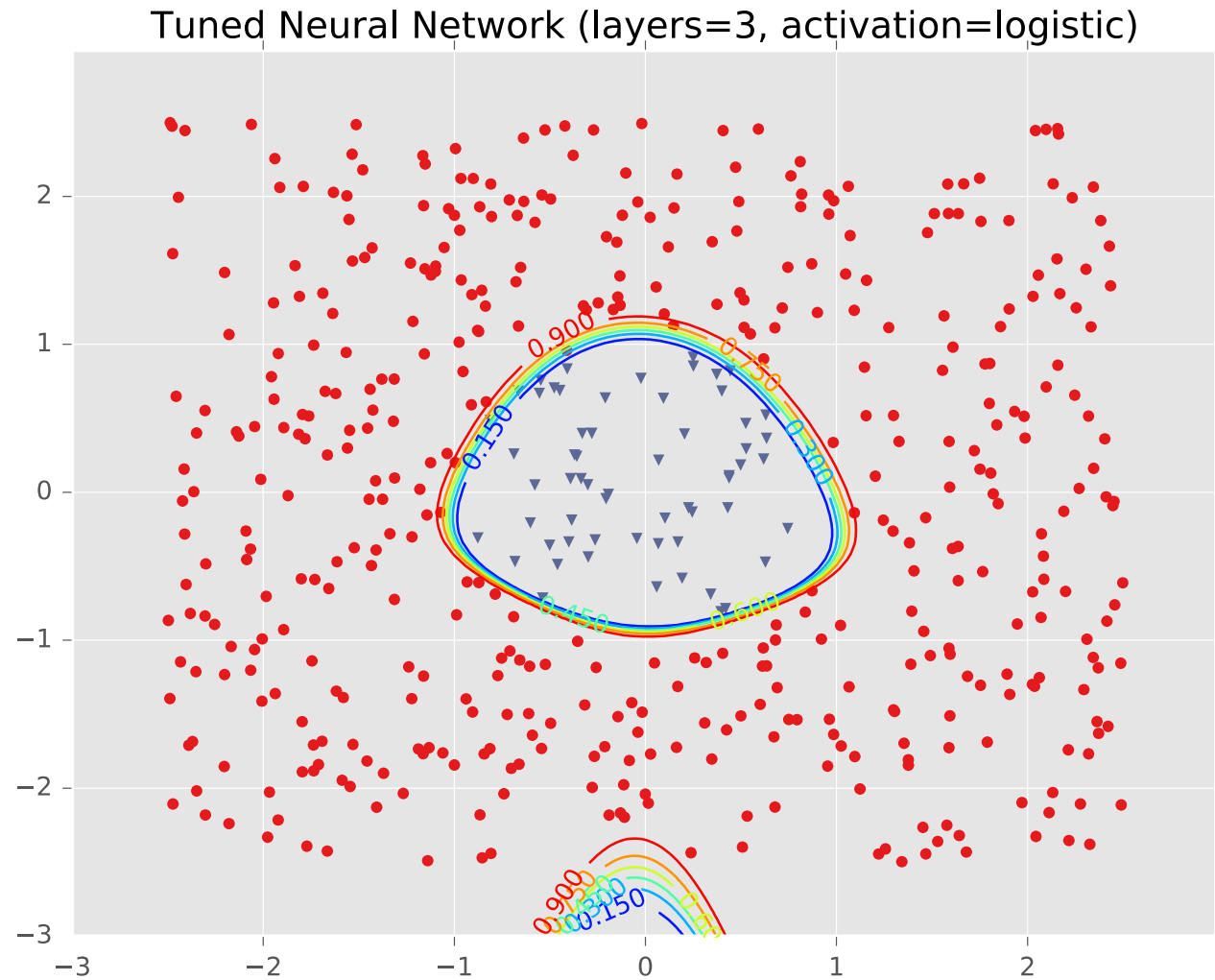
Neural Network Decision Boundaries: Example 2



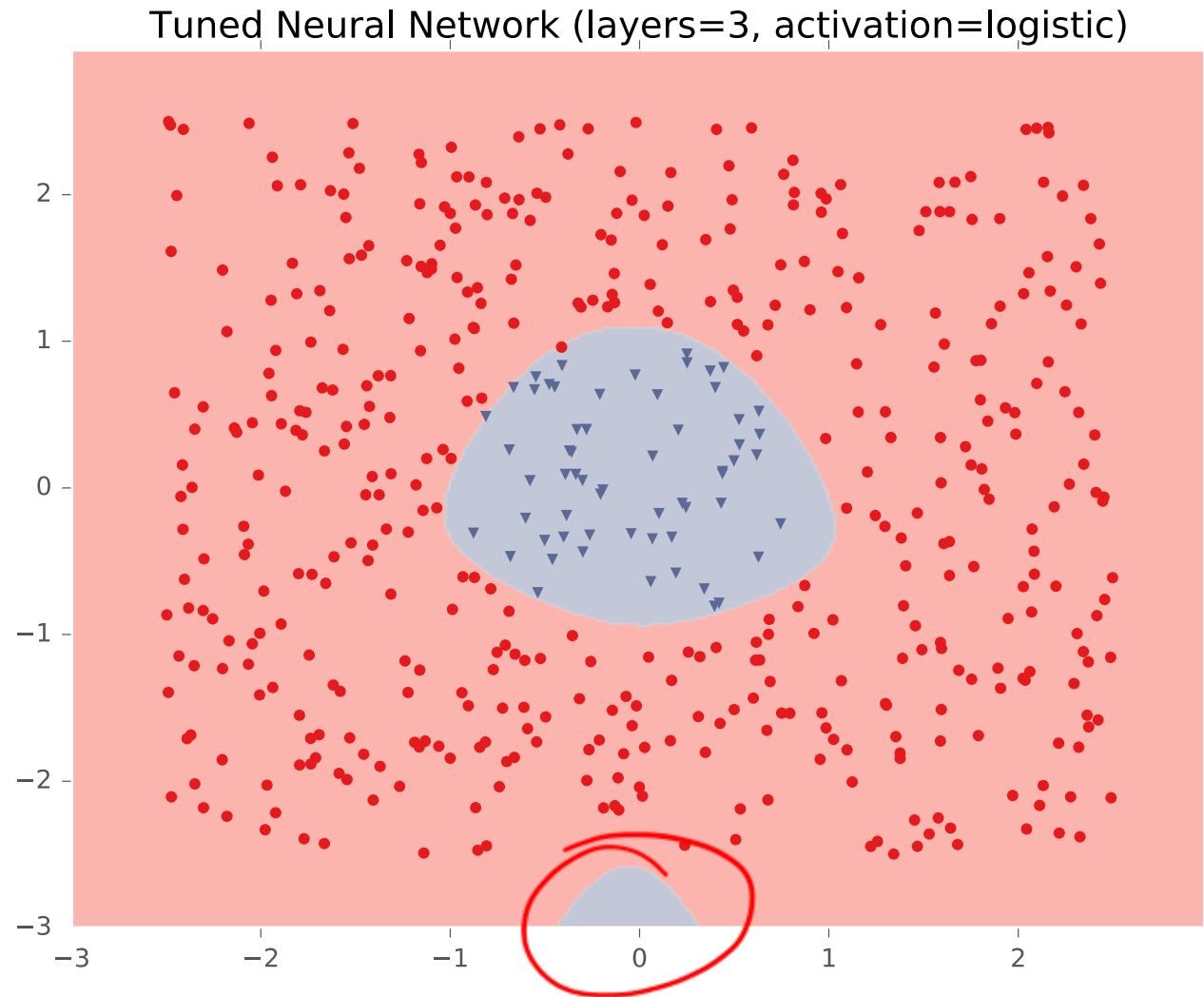
Neural Network Decision Boundaries: Example 2



Neural Network Decision Boundaries: Example 2

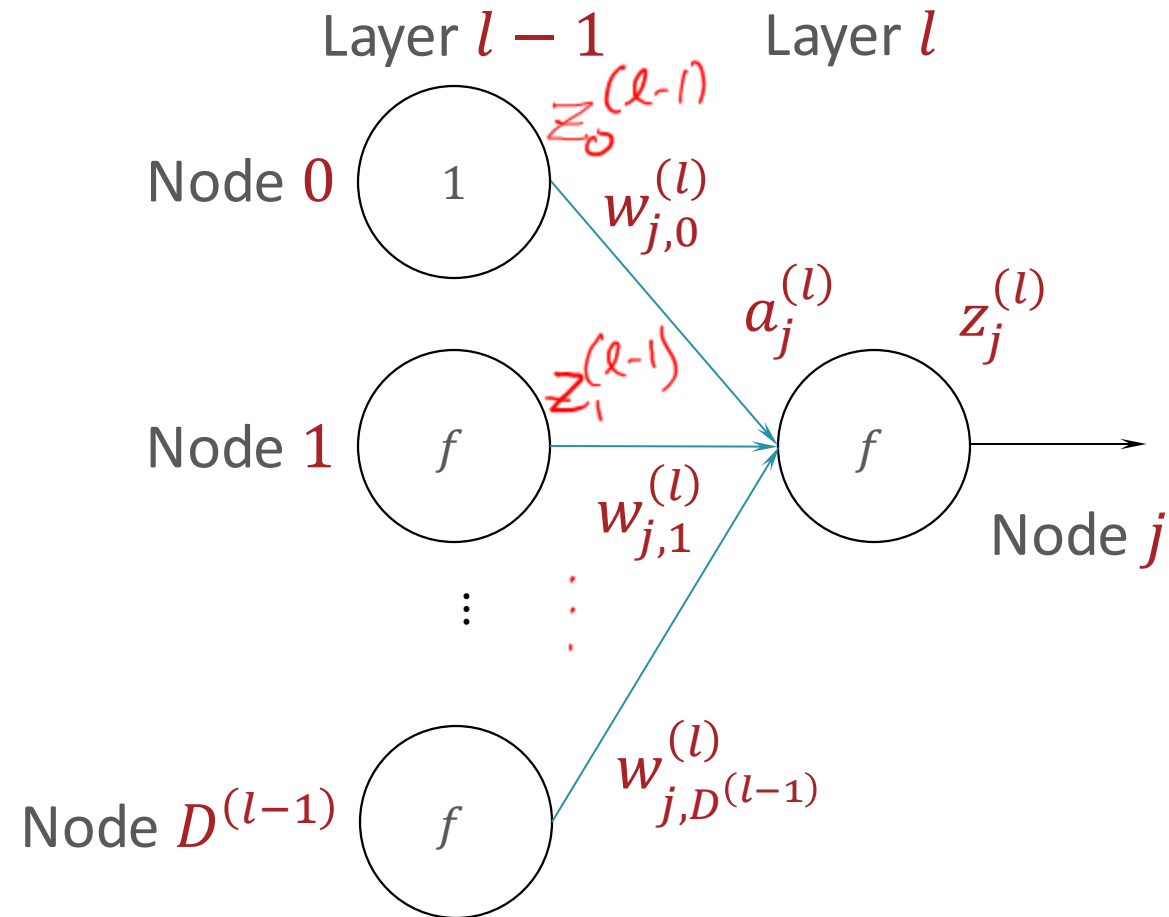


Neural Network Decision Boundaries: Example 2



Signal and Outputs

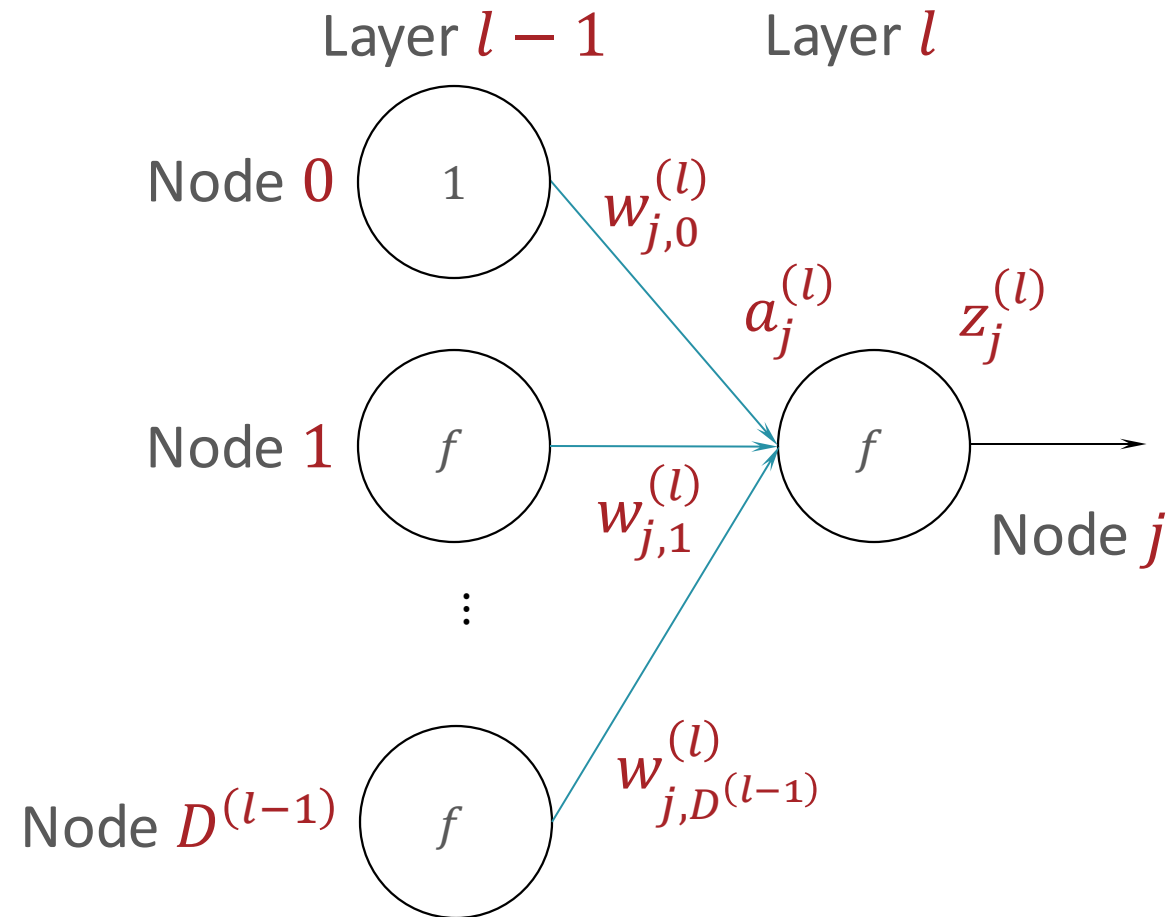
Every node has an incoming *signal* and outgoing *output*



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

Signal and Outputs

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



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

Forward Propagation for Making Predictions

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

Okay, but
where do
these weights
 $W^{(1)}, \dots, W^{(L)}$
come from?

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

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