Neural Networks

Recap: Story so far

$$\textbf{Predict } y \in \ \textbf{as} \begin{cases} y = \mathbf{x}^T\mathbf{w} & \text{(prediction function)} \\ p(y \mid \mathbf{x}, \mathbf{w}, \sigma^2) = \mathcal{N}\left(y \mid \mathbf{x}^T\mathbf{w}, \sigma^2\right) & \text{(probabilistic view)} \end{cases}$$

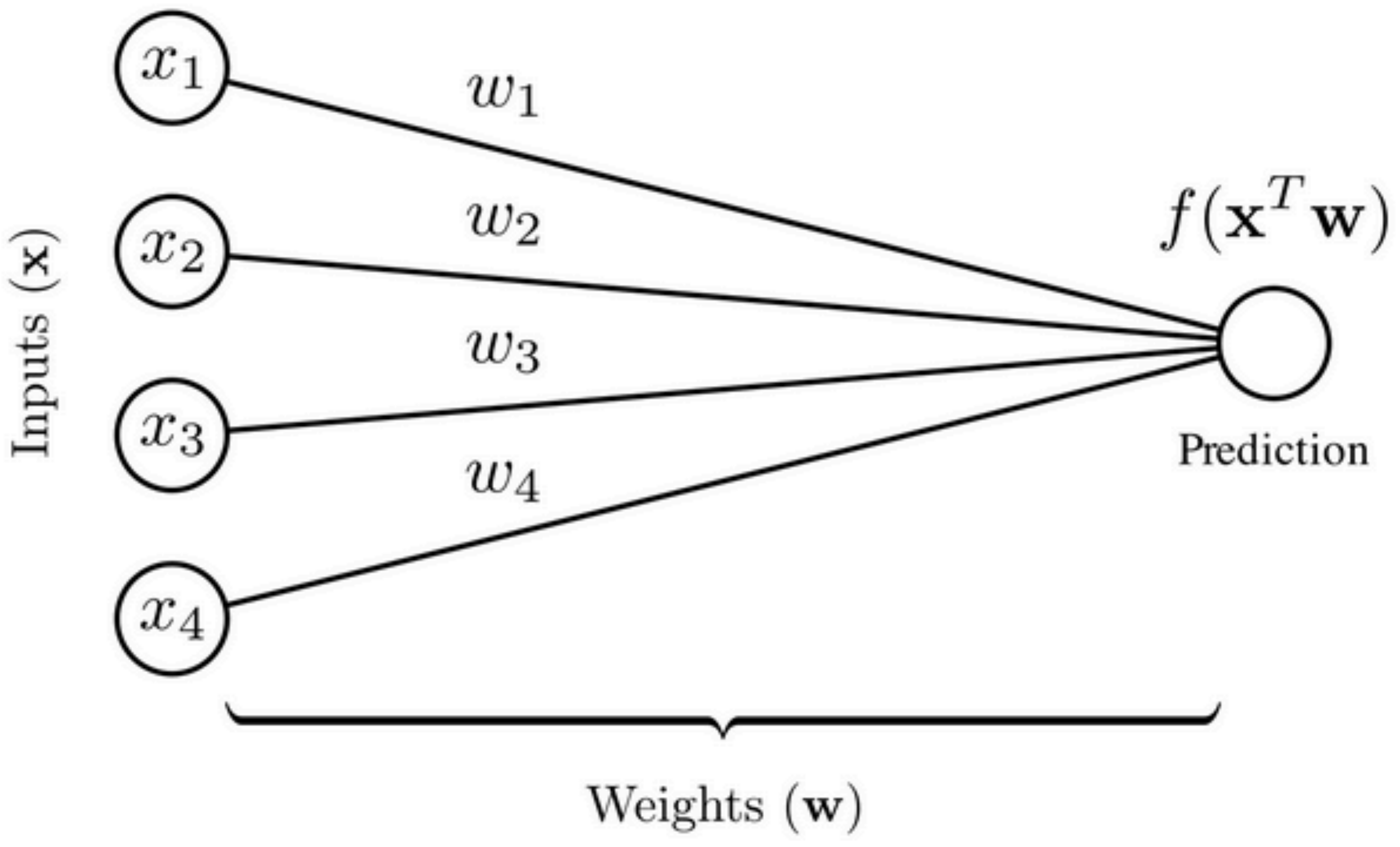
A reasonable model for *binary* outputs ($y \in \{0, 1\}$) is **logistic regression**:

$$\textbf{Predict } y \in \ \textbf{as} \begin{cases} y = \mathbb{I}(\mathbf{x}^T\mathbf{w} > 0) & \text{(prediction function)} \\ p(y = 1 \mid \mathbf{x}, \mathbf{w}) = \sigma(\mathbf{x}^T\mathbf{w}) & \text{(probabilistic view)} \end{cases}$$

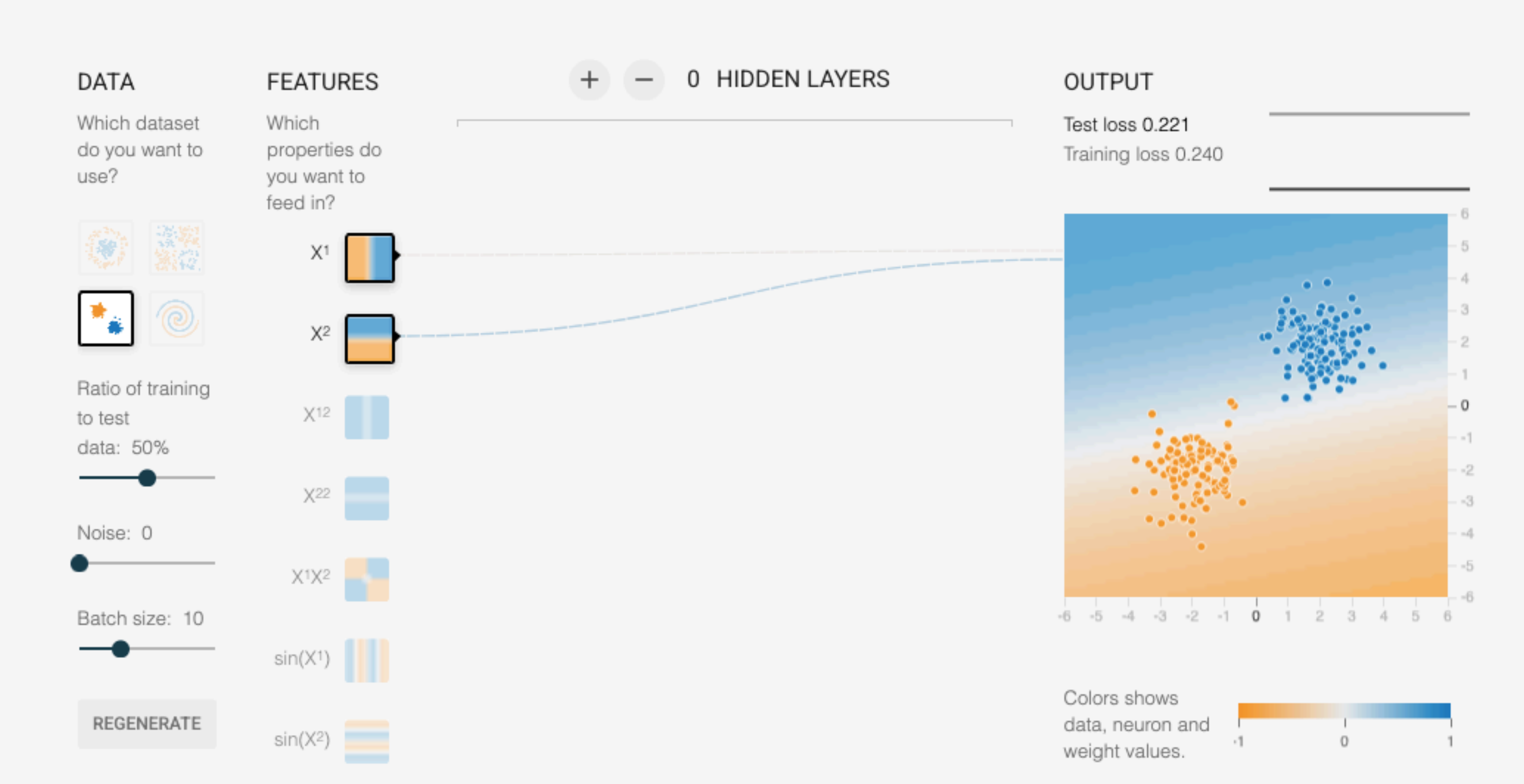
A reasonable model for *categorical* outputs ($y \in \{0, 1, \dots, C\}$) is **multinomial logistic regression**:

$$\textbf{Predict } y \in \ \textbf{as} \begin{cases} y = \underset{c}{\text{argmax}}\ \mathbf{x}^T\mathbf{w}_c & \text{(prediction function)} \\ p(y = c \mid \mathbf{x}, \mathbf{w}) = \text{softmax}(\mathbf{x}^T\mathbf{W})_c & \text{(probabilistic view)} \end{cases}$$

Recap: A new visualization

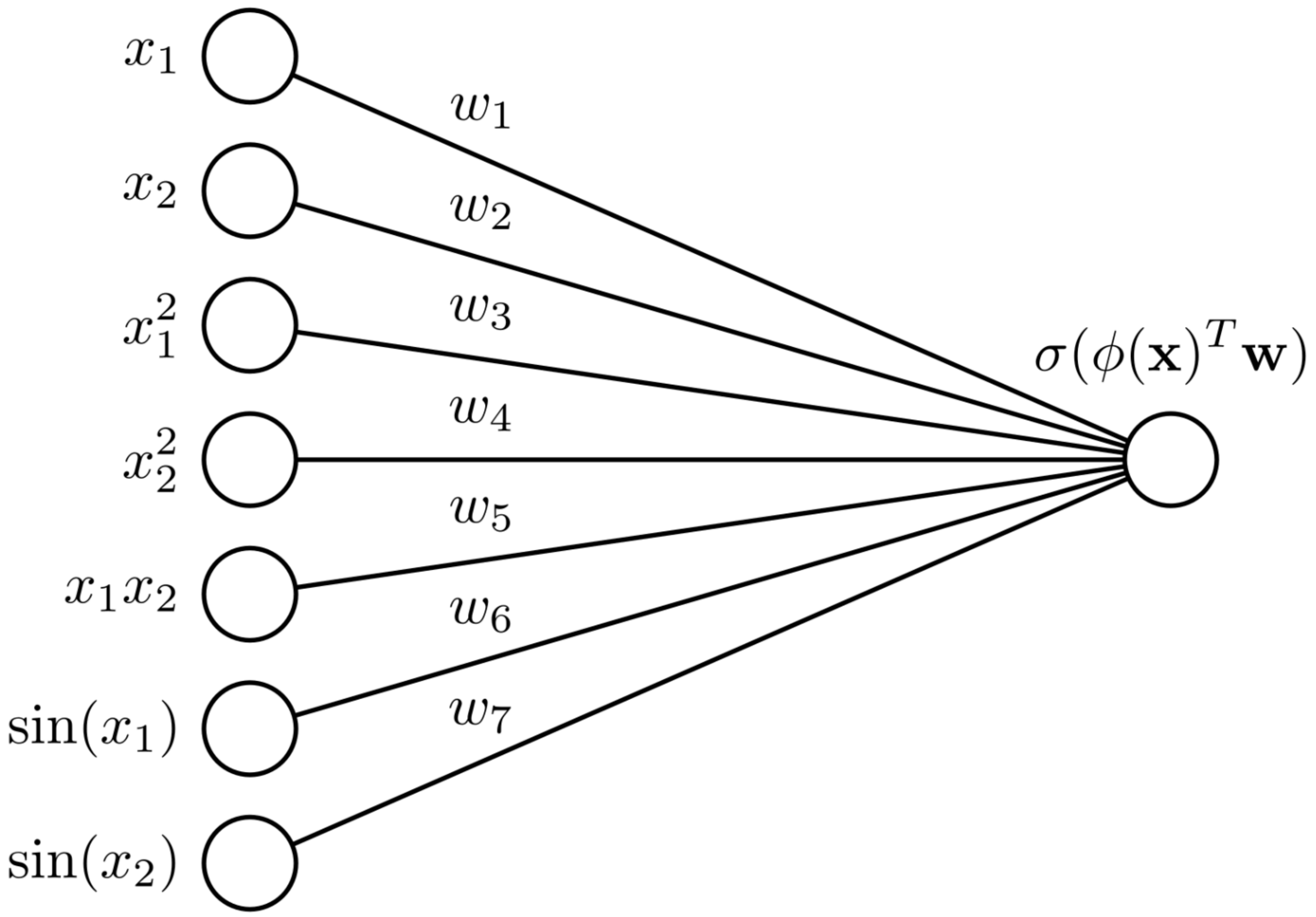


Recap: playground.tensorflow.org



Recap: feature transforms

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}, \quad \phi(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ x_1 x_2 \\ \sin(x_1) \\ \sin(x_2) \end{bmatrix}$$



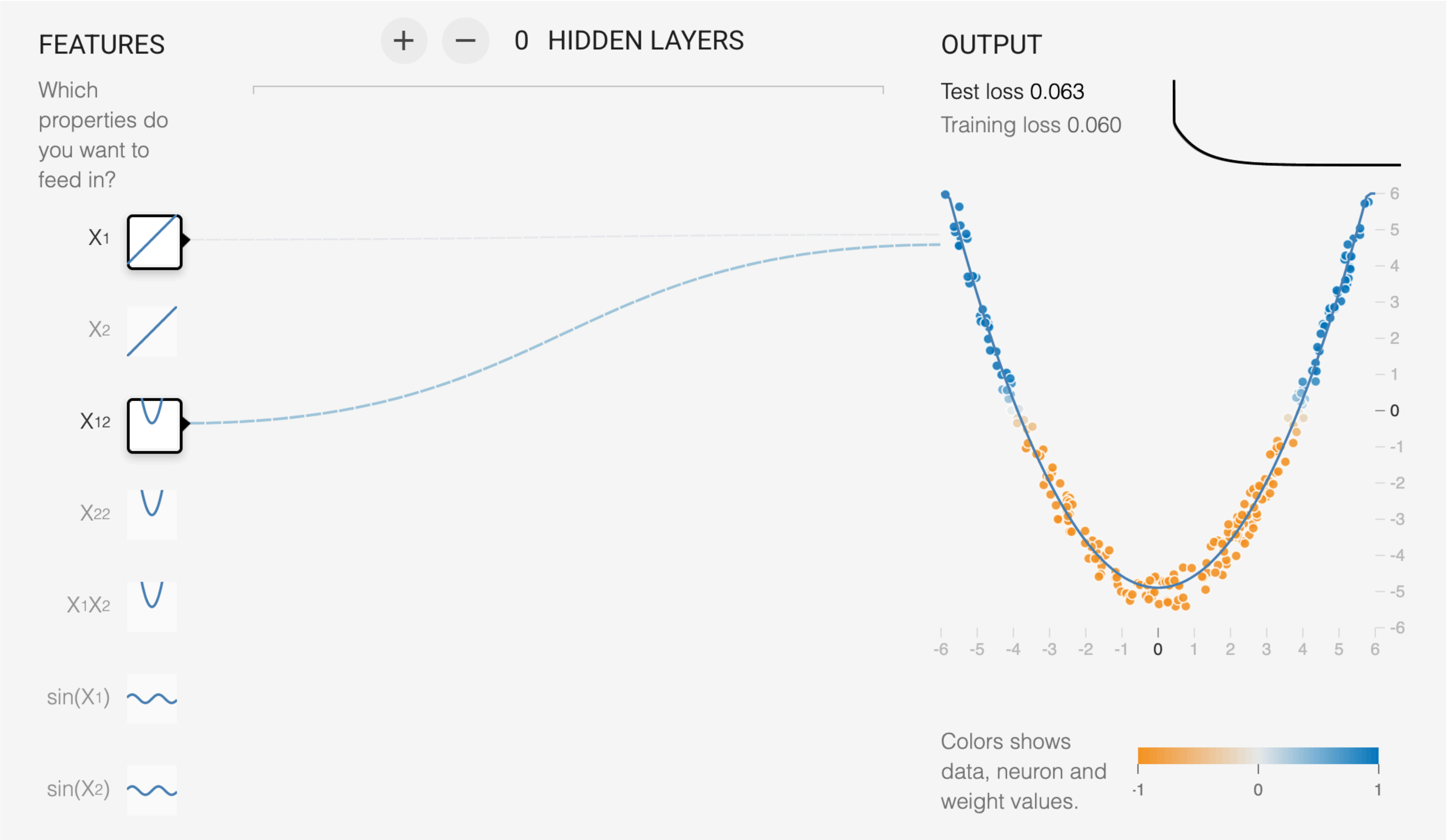
Recap: feature transforms for non-linear regression

Prediction function

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w} = w_2 x_1^2 + w_1 x_1 + b, \quad \phi(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_1^2 \\ 1 \end{bmatrix}$$

Probabilistic model

$$y_i \sim \mathcal{N}\left(\phi(\mathbf{x}_i)^T \mathbf{w}, \sigma^2\right)$$



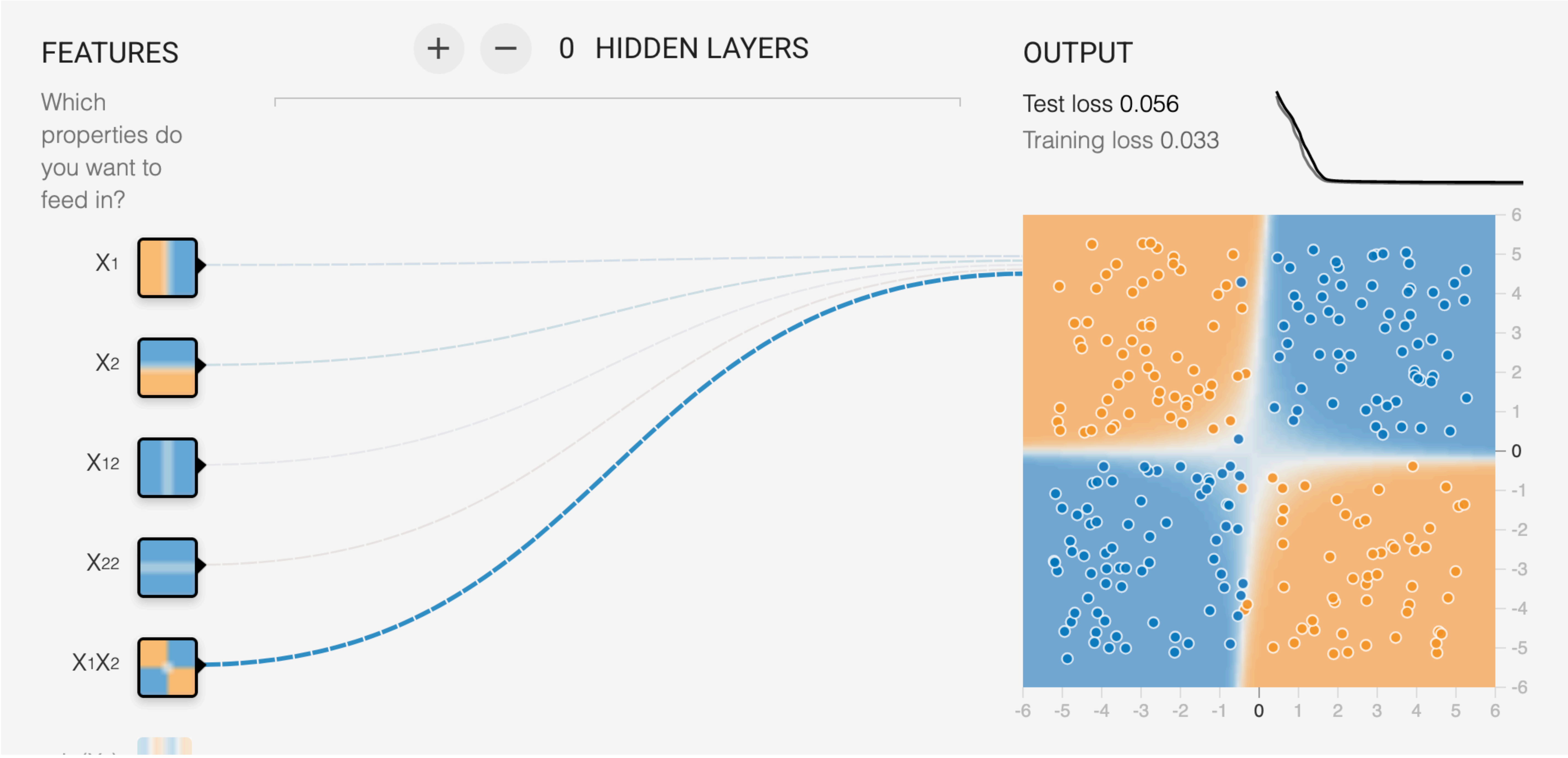
Recap: feature transforms for logistic regression

Prediction function

$$f(\mathbf{x}) = \mathbb{I}(\phi(\mathbf{x})^T \mathbf{w} \geq 0), \quad \phi(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \\ 1 \end{bmatrix}$$

Probabilistic model

$$y_i \sim \mathbf{Bernoulli}\big(\sigma(\phi(\mathbf{x_i})^{\mathbf{T}}\mathbf{w})\big), \quad p(y_i = 1 \mid \mathbf{x}_i, \mathbf{w}) = \sigma\big(\phi(\mathbf{x}_i)^T \mathbf{w}\big)$$



Neural networks setup

We've already seen that we can *learn* a function by defining our function in terms of a set of *parameters* $\mathbf{w}$:

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$$

and then minimizing a *loss* as a function of $\mathbf{w}$

$$\mathbf{w}^* = \underset{\mathbf{w}}{\text{argmin}}\ \mathbf{Loss}(\mathbf{w})$$

Which we can do with gradient descent:

$$\mathbf{w}^{(k+1)} \longleftarrow \mathbf{w}^{(k)} - \alpha \nabla_{\mathbf{w}} \mathbf{Loss}(\mathbf{w})$$

Can we *learn* the functions for our feature transforms?

A generic feature transform

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}, \quad \phi(\mathbf{x}) = \begin{bmatrix} g_1(x_1) \\ g_2(x_1) \\ g_3(x_1) \\ g_4(x_1) \end{bmatrix}$$

If we want to learn each *g,* what kind of function can we use?

A generic feature transform

$$f(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{w}, \quad \phi(\mathbf{x}) = \begin{bmatrix} g_1(x_1) \\ g_2(x_1) \\ g_3(x_1) \\ g_4(x_1) \end{bmatrix}$$

If we want to learn each *g,* what kind of function can we use?

Linear or logistic regression!

Logistic regression transform

$$g_i(\mathbf{x}) = \sigma(\mathbf{x}^T\mathbf{w}_i)$$

A *logistic regression* feature transform

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}, \quad \phi(\mathbf{x}) = \begin{bmatrix} g_1(x_1) \\ g_2(x_1) \\ g_3(x_1) \\ g_4(x_1) \end{bmatrix}$$

$$\Rightarrow$$

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}_0, \quad \phi(\mathbf{x}) = \begin{bmatrix} \sigma(\mathbf{x}^T \mathbf{w}_1) \\ \sigma(\mathbf{x}^T \mathbf{w}_2) \\ \sigma(\mathbf{x}^T \mathbf{w}_3) \\ \sigma(\mathbf{x}^T \mathbf{w}_4) \end{bmatrix}$$

A simple example

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{w}_0 = \begin{bmatrix} w_{01} \\ w_{02} \end{bmatrix}$$

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}_0, \quad \phi(\mathbf{x}) = \begin{bmatrix} \sigma(\mathbf{x}^T \mathbf{w}_1) \\ \sigma(\mathbf{x}^T \mathbf{w}_2) \\ \sigma(\mathbf{x}^T \mathbf{w}_3) \end{bmatrix} = \begin{bmatrix} \sigma(x_1 w_{11} + x_2 w_{12}) \\ \sigma(x_1 w_{21} + x_2 w_{22}) \\ \sigma(x_1 w_{31} + x_2 w_{32}) \end{bmatrix}$$

In this case, we can write out our prediction function explicitly as:

A simple example

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{w}_0 = \begin{bmatrix} w_{01} \\ w_{02} \end{bmatrix}$$

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}_0, \quad \phi(\mathbf{x}) = \begin{bmatrix} \sigma(\mathbf{x}^T \mathbf{w}_1) \\ \sigma(\mathbf{x}^T \mathbf{w}_2) \\ \sigma(\mathbf{x}^T \mathbf{w}_3) \end{bmatrix} = \begin{bmatrix} \sigma(x_1 w_{11} + x_2 w_{12}) \\ \sigma(x_1 w_{21} + x_2 w_{22}) \\ \sigma(x_1 w_{31} + x_2 w_{32}) \end{bmatrix}$$

In this case, we can write out our prediction function explicitly as:

$$f(\mathbf{x}) = w_{01} \cdot \sigma(x_1 w_{11} + x_2 w_{12}) + w_{02} \cdot \sigma(x_1 w_{21} + x_2 w_{22}) + w_{03} \cdot \sigma(x_1 w_{31} + x_2 w_{32})$$

As a node link-diagram

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{w}_0 = \begin{bmatrix} w_{01} \\ w_{02} \end{bmatrix}$$

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}_0, \quad \phi(\mathbf{x}) = \begin{bmatrix} \sigma(\mathbf{x}^T \mathbf{w}_1) \\ \sigma(\mathbf{x}^T \mathbf{w}_2) \\ \sigma(\mathbf{x}^T \mathbf{w}_3) \end{bmatrix} = \begin{bmatrix} \sigma(x_1 w_{11} + x_2 w_{12}) \\ \sigma(x_1 w_{21} + x_2 w_{22}) \\ \sigma(x_1 w_{31} + x_2 w_{32}) \end{bmatrix}$$

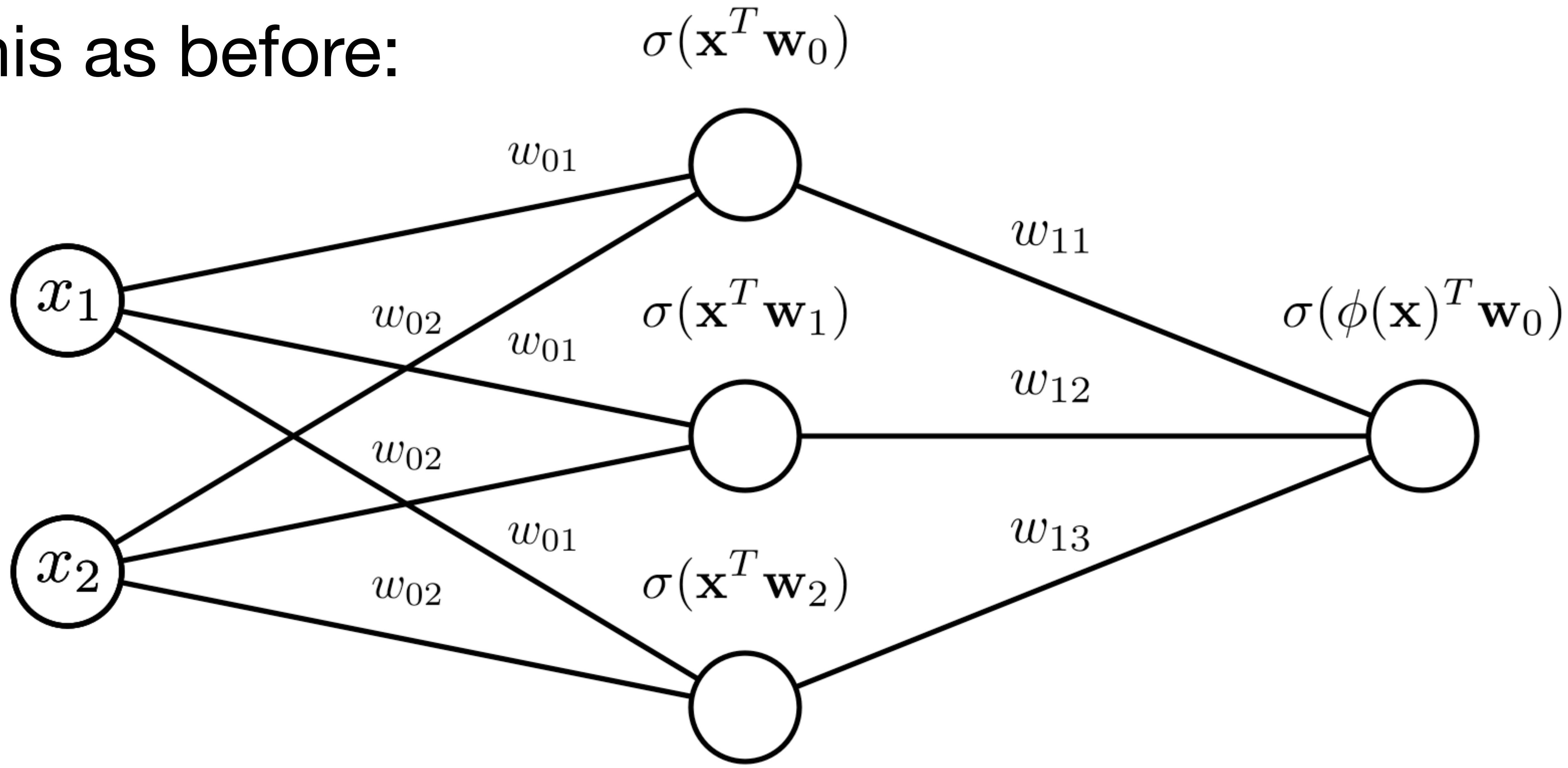
Visualizing this as before:

As a node link-diagram

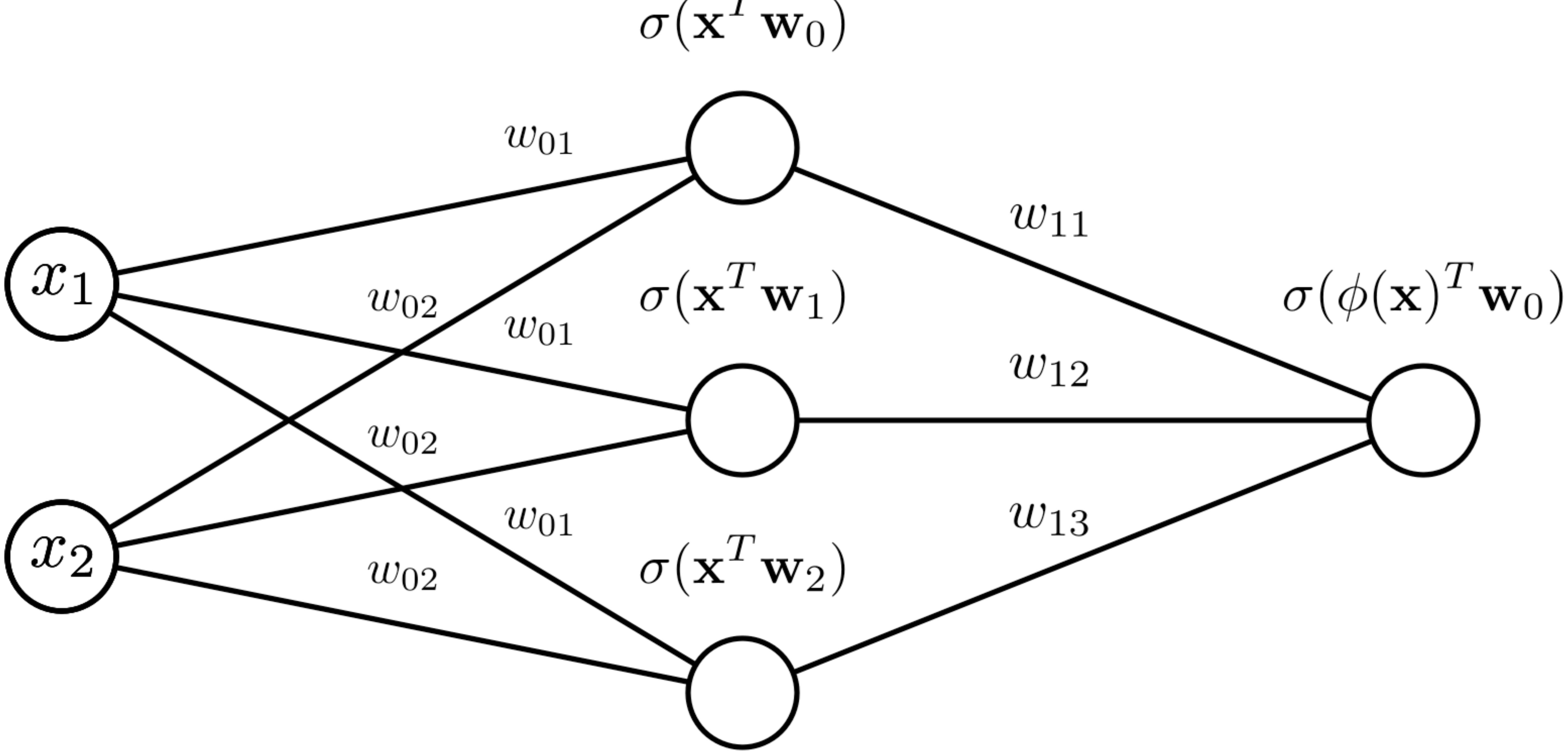
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{w}_0 = \begin{bmatrix} w_{01} \\ w_{02} \end{bmatrix}$$

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}_0, \quad \phi(\mathbf{x}) = \begin{bmatrix} \sigma(\mathbf{x}^T\mathbf{w}_1) \\ \sigma(\mathbf{x}^T\mathbf{w}_2) \\ \sigma(\mathbf{x}^T\mathbf{w}_3) \end{bmatrix} = \begin{bmatrix} \sigma(x_1 w_{11} + x_2 w_{12}) \\ \sigma(x_1 w_{21} + x_2 w_{22}) \\ \sigma(x_1 w_{31} + x_2 w_{32}) \end{bmatrix}$$

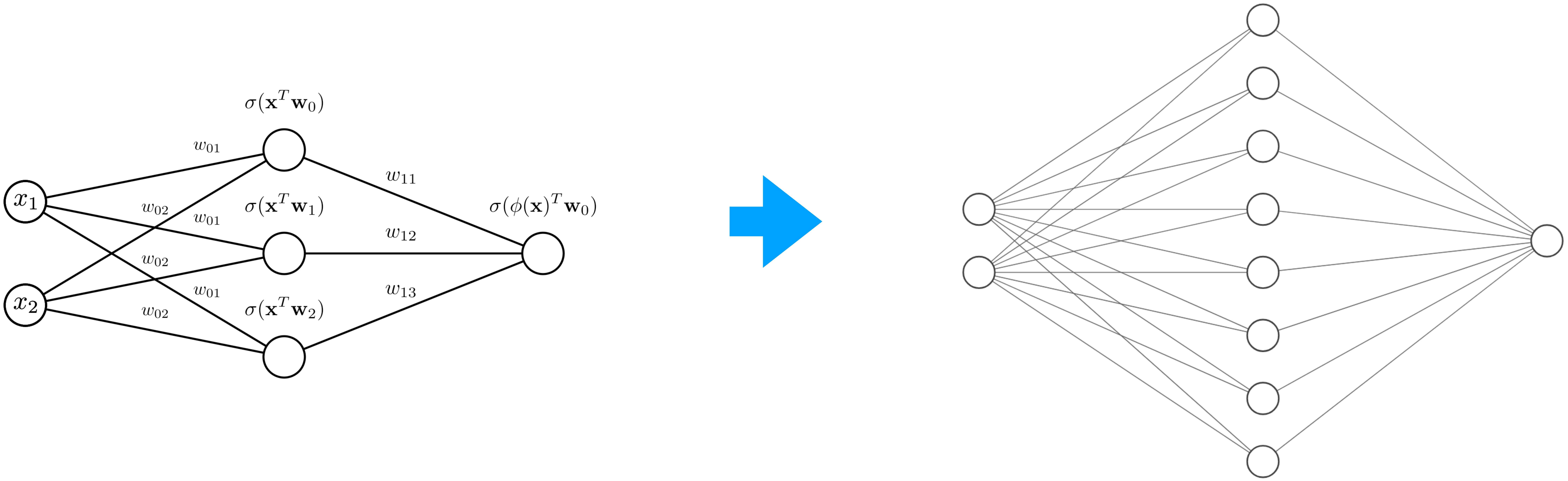
Visualizing this as before:



This is a Neural Network!



Omit labels for larger models



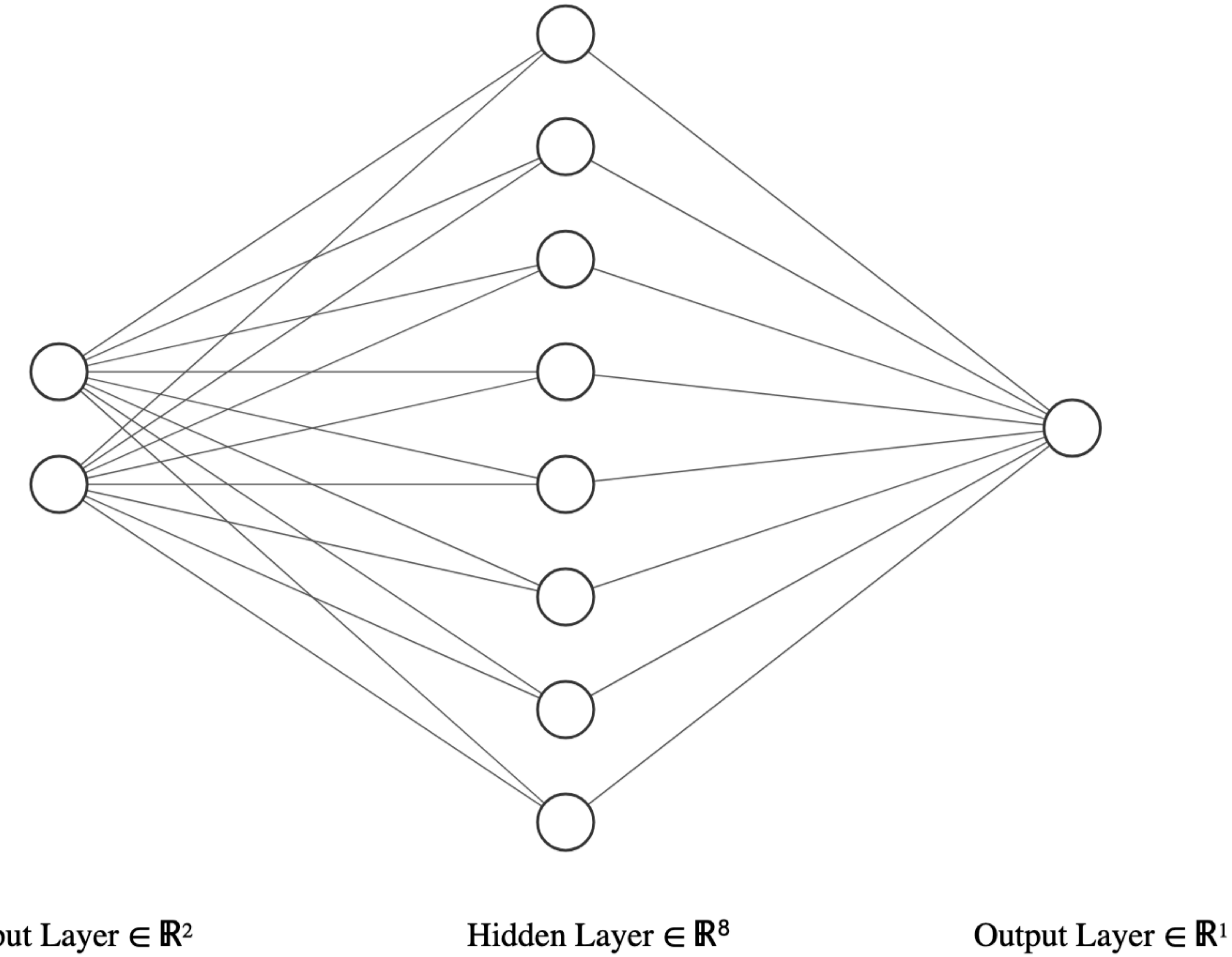
Neural network lingo

we call a single feature transform like $\sigma(x_1 w_{11} + x_2 w_{12})$ a **neuron**.

We call the whole set of transformed features the **hidden layer**:

$$\begin{bmatrix} \sigma(\mathbf{x}^T \mathbf{w}_1) \\ \sigma(\mathbf{x}^T \mathbf{w}_2) \\ \sigma(\mathbf{x}^T \mathbf{w}_3) \\ \sigma(\mathbf{x}^T \mathbf{w}_4) \end{bmatrix}$$

We call $\mathbf{x}$ the **input** and $f(\mathbf{x})$ the **output**.



Optimizing neural networks

We can still define a **loss function** for a neural network in the same way we did with our simpler linear models. The only difference is that now we have more parameters to choose:

$$\mathbf{Loss}(\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2, \dots)$$

Let's look at the logistic regression negative log-likelihood loss for the simple neural network we saw above:

$$p(y=1 \mid \mathbf{x}, \mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) = \sigma(\phi(\mathbf{x})^T \mathbf{w}_0), \quad \phi(\mathbf{x}) = \begin{bmatrix} \sigma(\mathbf{x}^T \mathbf{w}_1) \\ \sigma(\mathbf{x}^T \mathbf{w}_2) \\ \sigma(\mathbf{x}^T \mathbf{w}_3) \end{bmatrix}$$

$$= \sigma\big(w_{01} \cdot \sigma(x_1 w_{11} + x_2 w_{12}) + w_{02} \cdot \sigma(x_1 w_{21} + x_2 w_{22}) + w_{03} \cdot \sigma(x_1 w_{31} + x_2 w_{32})\big)$$

$$\mathbf{NLL}(\mathbf{w}_0, \dots, \mathbf{X}, \mathbf{y}) = -\sum_{i=1}^{N} \left[y_i \log p(y=1 \mid \mathbf{x}, \mathbf{w}_0, \dots) + (1 - y_i) \log p(y=0 \mid \mathbf{x}, \mathbf{w}_0, \dots) \right]$$

Optimizing neural networks

$$p(y = 1 \mid \mathbf{x}, \mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) = \sigma(\phi(\mathbf{x})^T \mathbf{w}_0), \quad \phi(\mathbf{x}) = \begin{bmatrix} \sigma(\mathbf{x}^T \mathbf{w}_1) \\ \sigma(\mathbf{x}^T \mathbf{w}_2) \\ \sigma(\mathbf{x}^T \mathbf{w}_3) \end{bmatrix}$$

$$= \sigma\big(w_{01} \cdot \sigma(x_1 w_{11} + x_2 w_{12}) + w_{02} \cdot \sigma(x_1 w_{21} + x_2 w_{22}) + w_{03} \cdot \sigma(x_1 w_{31} + x_2 w_{32})\big)$$

$$\mathbf{NLL}(\mathbf{w}_0, \dots, \mathbf{X}, \mathbf{y}) = -\sum_{i=1}^{N} \Big[y_i \log p(y = 1 \mid \mathbf{x}, \mathbf{w}_0, \dots) + (1 - y_i) \log p(y = 0 \mid \mathbf{x}, \mathbf{w}_0, \dots) \Big]$$

We can (in principle) take the gradient with respect to *all* of the parameters!

$$\nabla_{\mathbf{w}_0 \dots} = \begin{bmatrix} \frac{\partial \mathbf{NLL}}{\partial w_{01}} \\ \frac{\partial \mathbf{NLL}}{\partial w_{02}} \\ \frac{\partial \mathbf{NLL}}{\partial w_{03}} \\ \vdots \end{bmatrix}$$

Maybe tedious in practice, more on this next time…

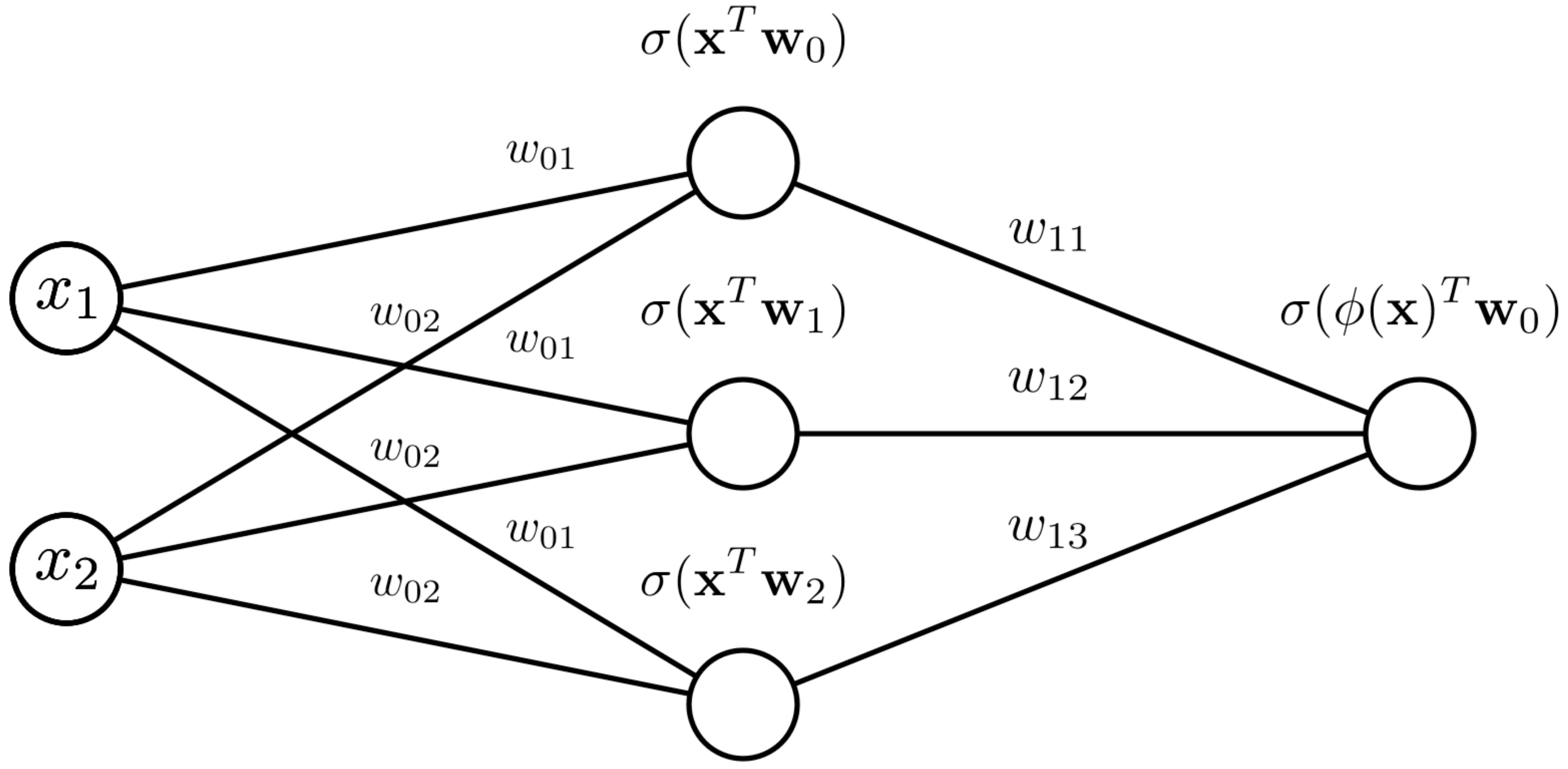
Try it out!

Neural networks with matrix notation

Our neural network

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{w}_0 = \begin{bmatrix} w_{01} \\ w_{02} \end{bmatrix}$$

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}_0, \quad \phi(\mathbf{x}) = \begin{bmatrix} \sigma(\mathbf{x}^T \mathbf{w}_1) \\ \sigma(\mathbf{x}^T \mathbf{w}_2) \\ \sigma(\mathbf{x}^T \mathbf{w}_3) \end{bmatrix} = \begin{bmatrix} \sigma(x_1 w_{11} + x_2 w_{12}) \\ \sigma(x_1 w_{21} + x_2 w_{22}) \\ \sigma(x_1 w_{31} + x_2 w_{32}) \end{bmatrix}$$



Write the linear function for each neuron as a matrix vector product

$$\begin{bmatrix} \mathbf{x}^T \mathbf{w}_1 \\ \mathbf{x}^T \mathbf{w}_2 \\ \mathbf{x}^T \mathbf{w}_3 \end{bmatrix} = \begin{bmatrix} x_1 w_{11} + x_2 w_{12} \\ x_1 w_{21} + x_2 w_{22} \\ x_1 w_{31} + x_2 w_{32} \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{x}^T \mathbf{w}_1 \\ \mathbf{x}^T \mathbf{w}_2 \\ \mathbf{x}^T \mathbf{w}_3 \end{bmatrix} = \mathbf{W}\mathbf{x} = (\mathbf{x}^T \mathbf{W}^T)^T$$

Neural networks with matrix notation

Check the dimensions!

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{x}^T \mathbf{w}_1 \\ \mathbf{x}^T \mathbf{w}_2 \\ \mathbf{x}^T \mathbf{w}_3 \end{bmatrix} = \mathbf{W}\mathbf{x} = (\mathbf{x}^T \mathbf{W}^T)^T$$

Neural networks with matrix notation

Check the dimensions!

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \qquad \begin{bmatrix} \mathbf{x}^T\mathbf{w}_1 \\ \mathbf{x}^T\mathbf{w}_2 \\ \mathbf{x}^T\mathbf{w}_3 \end{bmatrix} = \mathbf{W}\mathbf{x} = (\mathbf{x}^T\mathbf{W}^T)^T$$

Substitute into our prediction function:

$$\phi(\mathbf{x}) = \sigma(\mathbf{x}^T\mathbf{W}^T)^T, \quad f(\mathbf{x}) = \sigma(\mathbf{x}^T\mathbf{W}^T)\mathbf{w}_0$$

Neural networks with matrix notation

Data and weights as matrices

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \\ \vdots \end{bmatrix} \quad \mathbf{W}_1 = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \mathbf{w}_3^T \\ \vdots \end{bmatrix} \qquad \phi(\mathbf{X}) = \sigma(\mathbf{X}\mathbf{W}^T)^T = \begin{bmatrix} \sigma(\mathbf{x}_1^T\mathbf{w}_1) & \sigma(\mathbf{x}_1^T\mathbf{w}_2) & \dots & \sigma(\mathbf{x}_1^T\mathbf{w}_h \\ \sigma(\mathbf{x}_2^T\mathbf{w}_1) & \sigma(\mathbf{x}_2^T\mathbf{w}_2) & \dots & \sigma(\mathbf{x}_2^T\mathbf{w}_h) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma(\mathbf{x}_N^T\mathbf{w}_1) & \sigma(\mathbf{x}_N^T\mathbf{w}_2) & \dots & \sigma(\mathbf{x}_N^T\mathbf{w}_h) \end{bmatrix}$$

Prediction function for dataset

$$f(\mathbf{x}) = \sigma(\mathbf{X}\mathbf{W}^T)\mathbf{w}_0$$

Neural networks with matrix notation

$$f(\mathbf{x}) = \sigma(\mathbf{X}\mathbf{W}^T)\mathbf{w}_0$$

To summarize:

- $\mathbf{X}$: $N \times d$ matrix of observations
- $\mathbf{W}$: $h \times d$ matrix of network weights
- $\mathbf{w}_0$: h $(\times 1)$ vector of linear regression weights

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \\ \vdots \end{bmatrix} \quad \mathbf{W}_1 = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \mathbf{w}_3^T \\ \vdots \end{bmatrix}$$

If we check that our dimensions work for matrix multiplication we see that we get the $N \times 1$ vector of predictions we are looking for!

Neural networks with matrix notation

$$f(\mathbf{x}) = \sigma(\mathbf{X}\mathbf{W}^T)\mathbf{w}_0$$

To summarize:

- $\mathbf{X}$: $N \times d$ matrix of observations
- $\mathbf{W}$: $h \times d$ matrix of network weights
- $\mathbf{w}_0$: $h\ (\times 1)$ vector of linear regression weights

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \\ \vdots \end{bmatrix} \qquad \mathbf{W}_1 = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \mathbf{w}_3^T \\ \vdots \end{bmatrix}$$

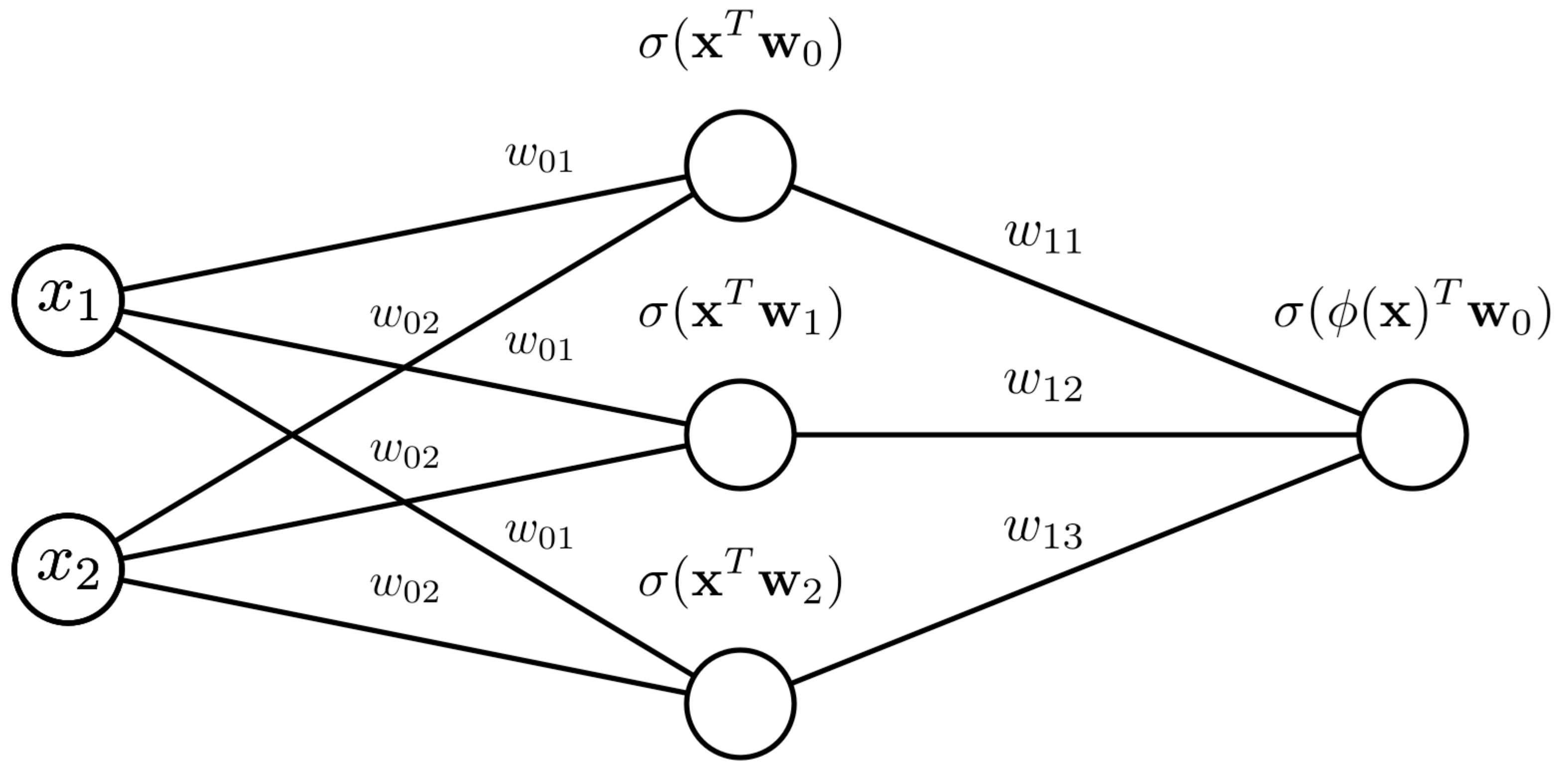
If we check that our dimensions work for matrix multiplication we see that we get the $N \times 1$ vector of predictions we are looking for!

$$(N \times d)(h \times d)^T(h \times 1) \rightarrow (N \times d)(d \times h)(h \times 1) \rightarrow (N \times h)(h \times 1)$$

$$\longrightarrow (N \times 1)$$

Neural networks with matrix notation

Write all the weights for a layer in a big matrix



$$\mathbf{W}_1 = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \mathbf{w}_3^T \\ \vdots \end{bmatrix}$$

$$f(\mathbf{x}) = \sigma(\mathbf{W}_1\mathbf{x})^T\mathbf{w_0}$$

Compact prediction function!

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \\ \vdots \end{bmatrix}$$

Can do the same for the full dataset

$$f(\mathbf{x}) = \sigma(\mathbf{X}\mathbf{W}_1)\mathbf{w_0}$$

Why *logistic regression?*

Try a linear transform

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{w}_0 = \begin{bmatrix} w_{01} \\ w_{02} \end{bmatrix}$$

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}_0, \quad \phi(\mathbf{x}) = \begin{bmatrix} \mathbf{x}^T \mathbf{w}_1 \\ \mathbf{x}^T \mathbf{w}_2 \\ \mathbf{x}^T \mathbf{w}_3 \end{bmatrix} = \begin{bmatrix} x_1 w_{11} + x_2 w_{12} \\ x_1 w_{21} + x_2 w_{22} \\ x_1 w_{31} + x_2 w_{32} \end{bmatrix}$$

In this case, we can write out our prediction function explicitly as:

$$f(\mathbf{x}) = w_{01} \cdot (x_1 w_{11} + x_2 w_{12}) + w_{02} \cdot (x_1 w_{21} + x_2 w_{22}) + w_{03} \cdot (x_1 w_{31} + x_2 w_{32})$$

Why *logistic regression?*

Try a linear transform

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{w}_0 = \begin{bmatrix} w_{01} \\ w_{02} \end{bmatrix}$$

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}_0, \quad \phi(\mathbf{x}) = \begin{bmatrix} \mathbf{x}^T \mathbf{w}_1 \\ \mathbf{x}^T \mathbf{w}_2 \\ \mathbf{x}^T \mathbf{w}_3 \end{bmatrix} = \begin{bmatrix} x_1 w_{11} + x_2 w_{12} \\ x_1 w_{21} + x_2 w_{22} \\ x_1 w_{31} + x_2 w_{32} \end{bmatrix}$$

In this case, we can write out our prediction function explicitly as:

$$f(\mathbf{x}) = w_{01} \cdot (x_1 w_{11} + x_2 w_{12}) + w_{02} \cdot (x_1 w_{21} + x_2 w_{22}) + w_{03} \cdot (x_1 w_{31} + x_2 w_{32})$$

$$= (w_{11} w_{01}) x_1 + (w_{12} w_{01}) x_2 + (w_{21} w_{02}) x_1 + (w_{22} w_{02}) x_2 + (w_{31} w_{03}) x_1 + (w_{32} w_{03}) x_2$$

$$= (w_{11} w_{01} + w_{21} w_{02} + w_{31} w_{03}) x_1 + (w_{12} w_{01} + w_{22} w_{02} + w_{32} w_{03}) x_2$$

Still linear!

Why *logistic regression?*

Try a linear transform (vector notation)

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}_0, \quad \phi(\mathbf{x}) = \begin{bmatrix} \mathbf{x}^T \mathbf{w}_1 \\ \mathbf{x}^T \mathbf{w}_2 \\ \mathbf{x}^T \mathbf{w}_3 \\ \mathbf{x}^T \mathbf{w}_4 \end{bmatrix}$$

$$f(\mathbf{x}) = w_{01}(\mathbf{x}^T \mathbf{w}_1) + w_{02}(\mathbf{x}^T \mathbf{w}_2) + \ldots$$

$$= \mathbf{x}^T (w_{01} \mathbf{w}_1) + \mathbf{x}^T (w_{02} \mathbf{w}_2) + \ldots$$

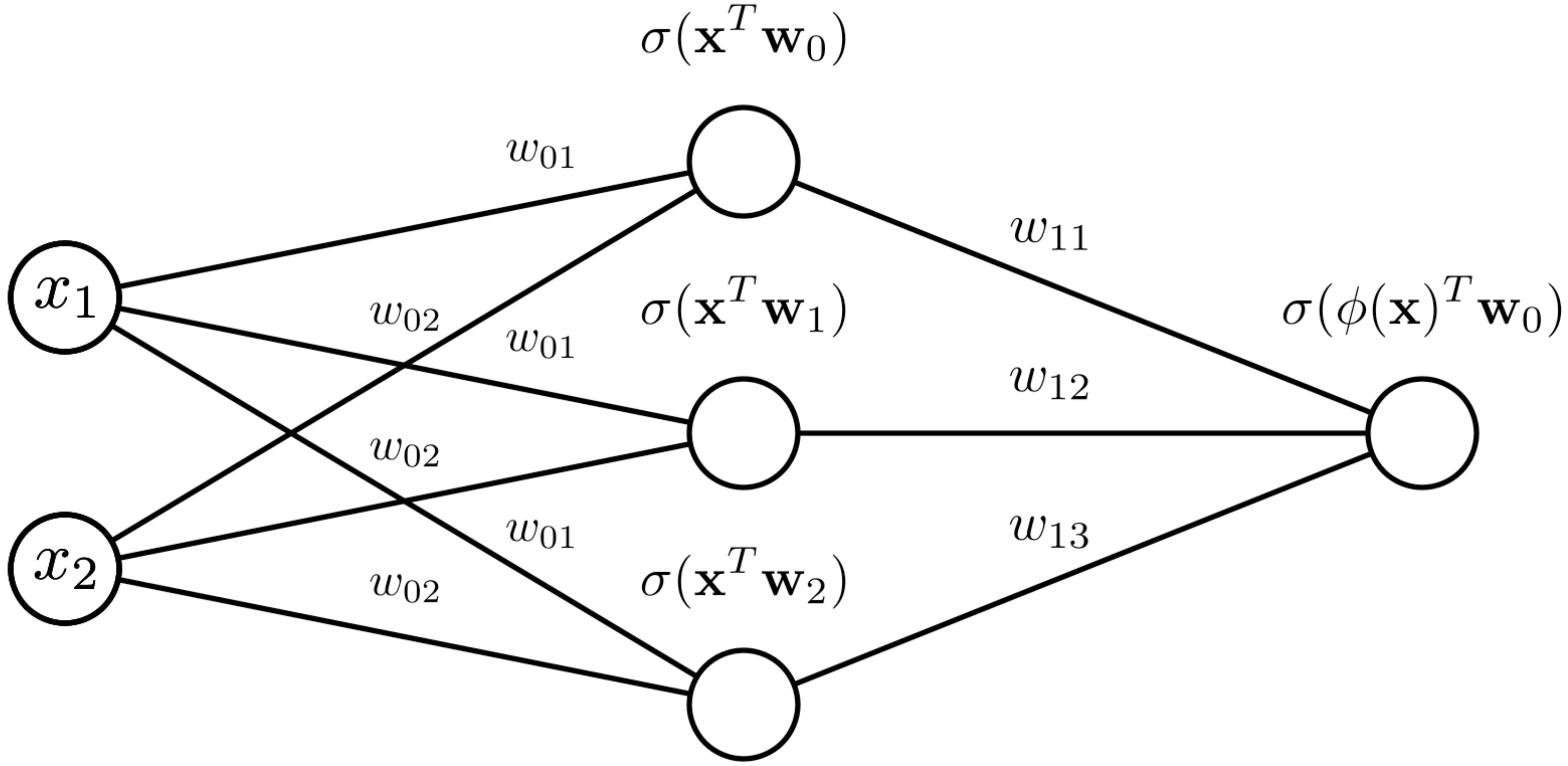
Try it out!

Why *logistic regression?*

We can't use a linear transform, but do we need to use the ***sigmoid function?***

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{w}_0 = \begin{bmatrix} w_{01} \\ w_{02} \end{bmatrix}$$

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}_0, \quad \phi(\mathbf{x}) = \begin{bmatrix} \sigma(\mathbf{x}^T\mathbf{w}_1) \\ \sigma(\mathbf{x}^T\mathbf{w}_2) \\ \sigma(\mathbf{x}^T\mathbf{w}_3) \end{bmatrix} = \begin{bmatrix} \sigma(x_1 w_{11} + x_2 w_{12}) \\ \sigma(x_1 w_{21} + x_2 w_{22}) \\ \sigma(x_1 w_{31} + x_2 w_{32}) \end{bmatrix}$$



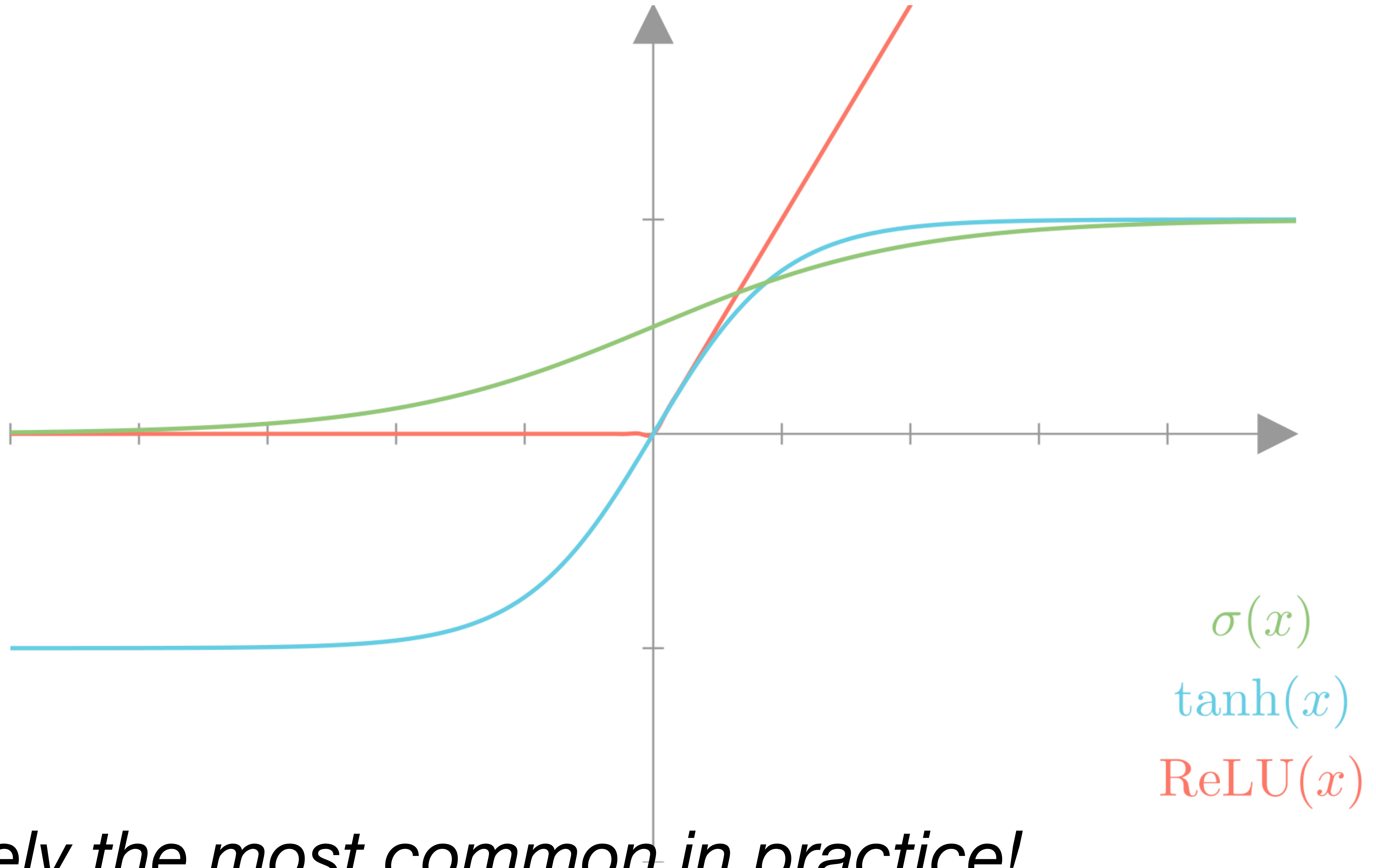
Activation functions!

Plenty of replacements for the sigmoid function are used in practice!

Sigmoid: $\sigma(x) = \dfrac{1}{1+e^{-x}}$

Hyperbolic tangent: $\tanh(x) = \dfrac{e^{2x}-1}{e^{2x}+1}$

Rectifed linear: $\mathrm{ReLU}(x) = \max(x,\ 0)$

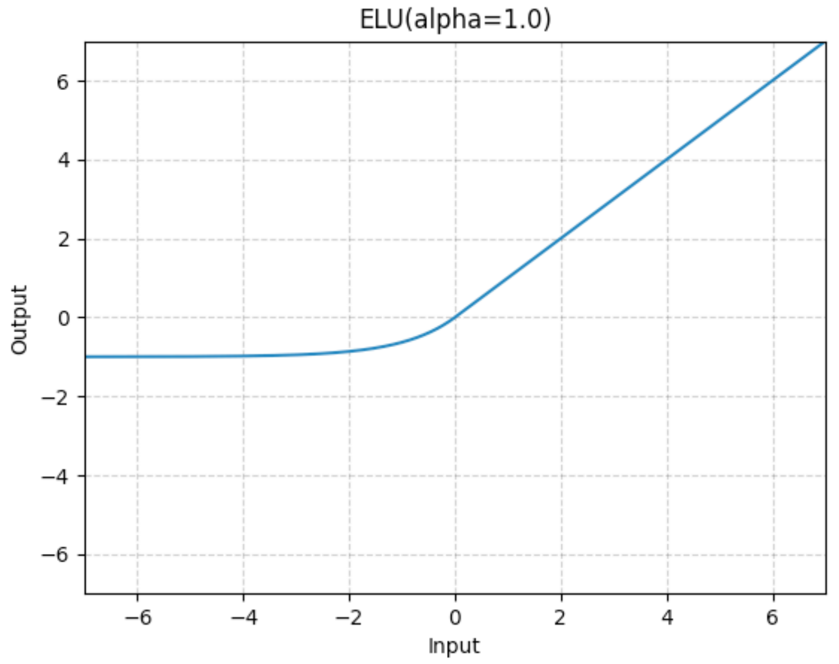


Rectified linear *is likely the most common in practice!*

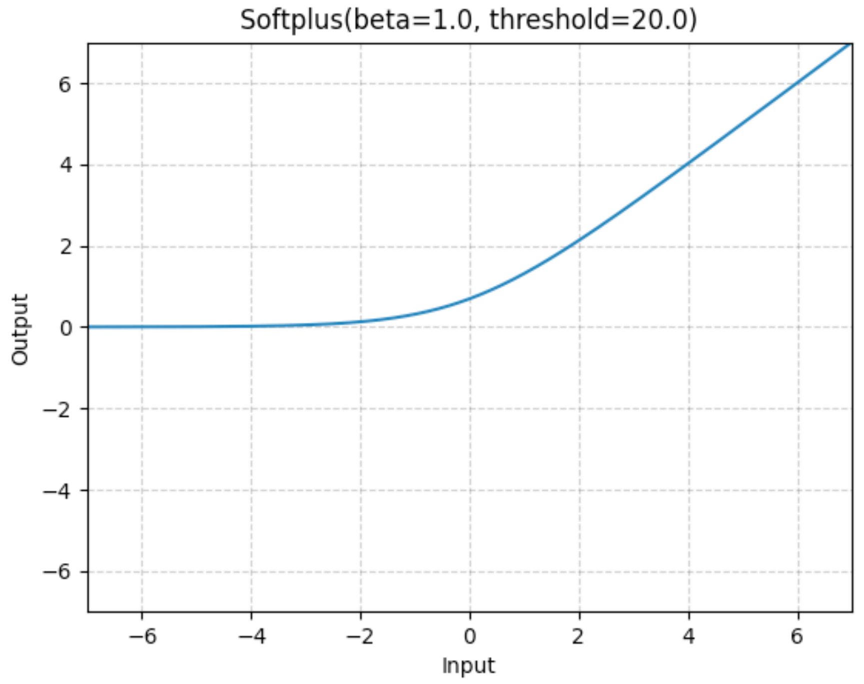
Can you guess why? (Try it out)

Common activations

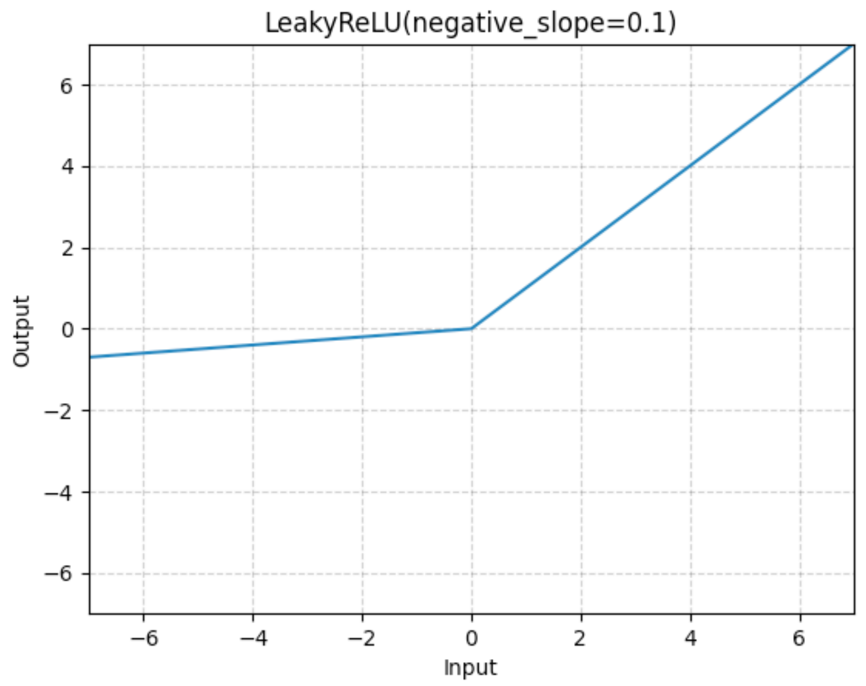
$$\mathrm{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha * (\exp(x) - 1), & \text{if } x \leq 0 \end{cases}$$



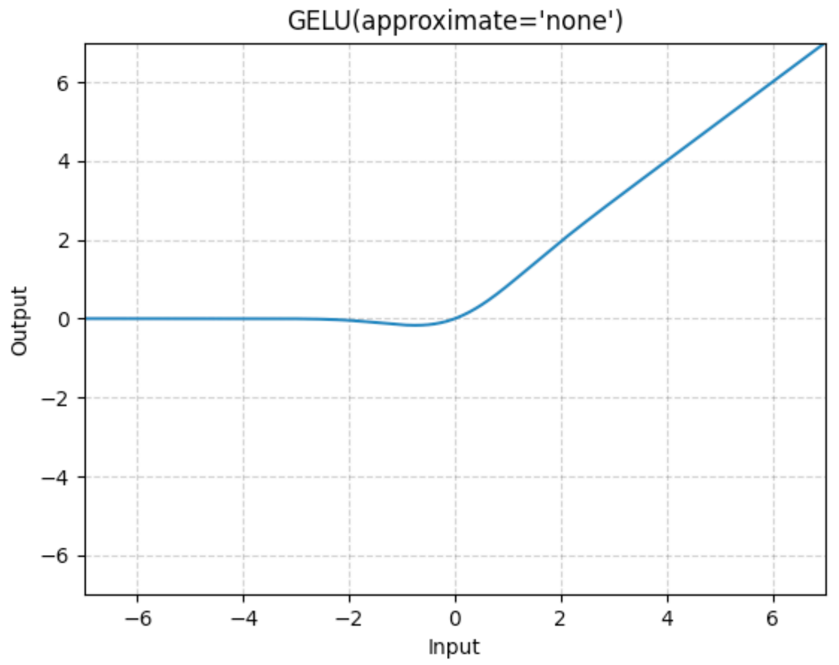
$$\mathrm{Softplus}(x) = \frac{1}{\beta} * \log(1 + \exp(\beta * x))$$



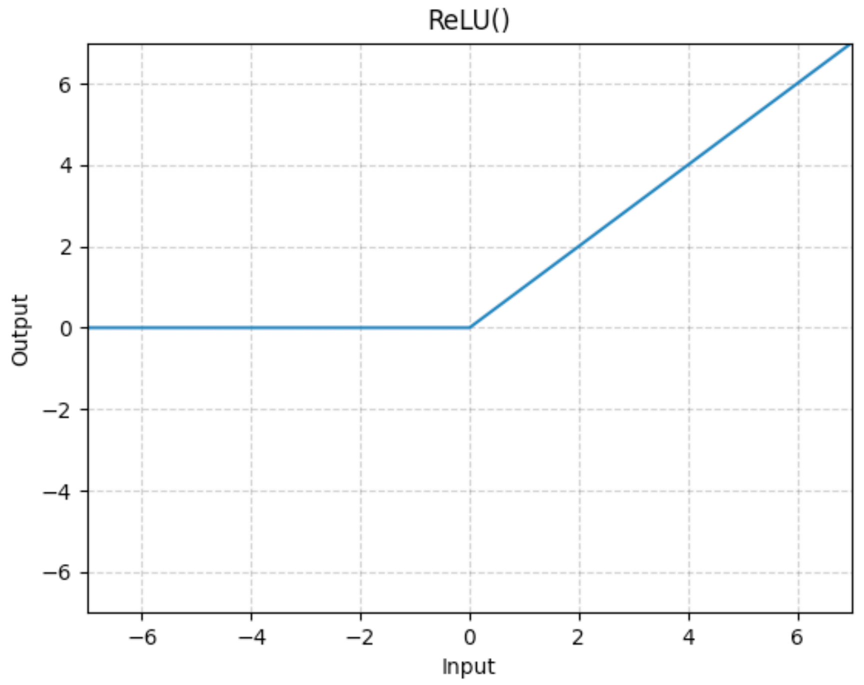
$$\mathrm{LeakyReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \text{negative\_slope} \times x, & \text{otherwise} \end{cases}$$



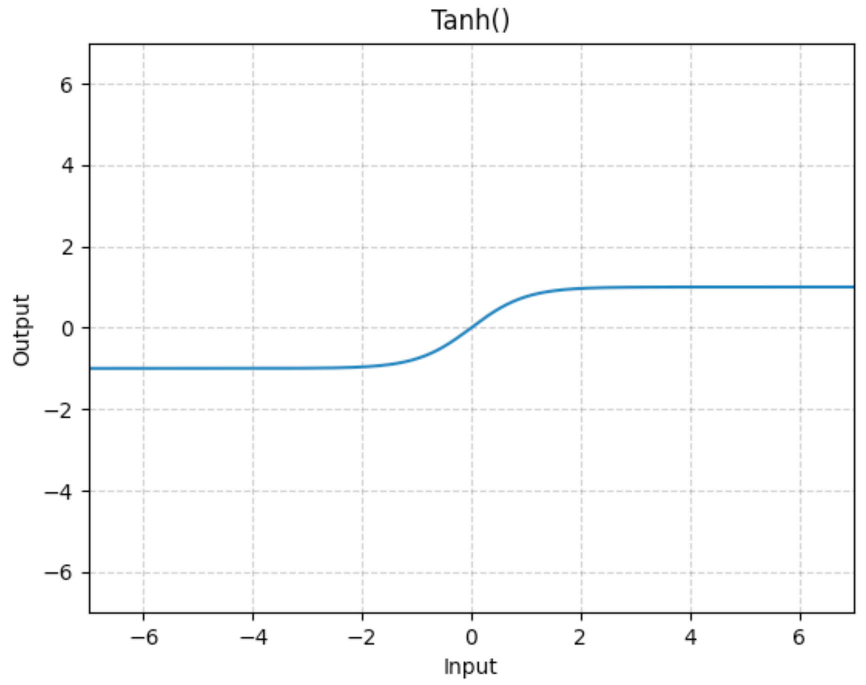
$$\mathrm{GELU}(x) = x * \Phi(x)$$



$$\mathrm{ReLU}(x) = (x)^+ = \max(0, x)$$



$$\mathrm{Tanh}(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$



We'll talk more about the merits of these options later on!