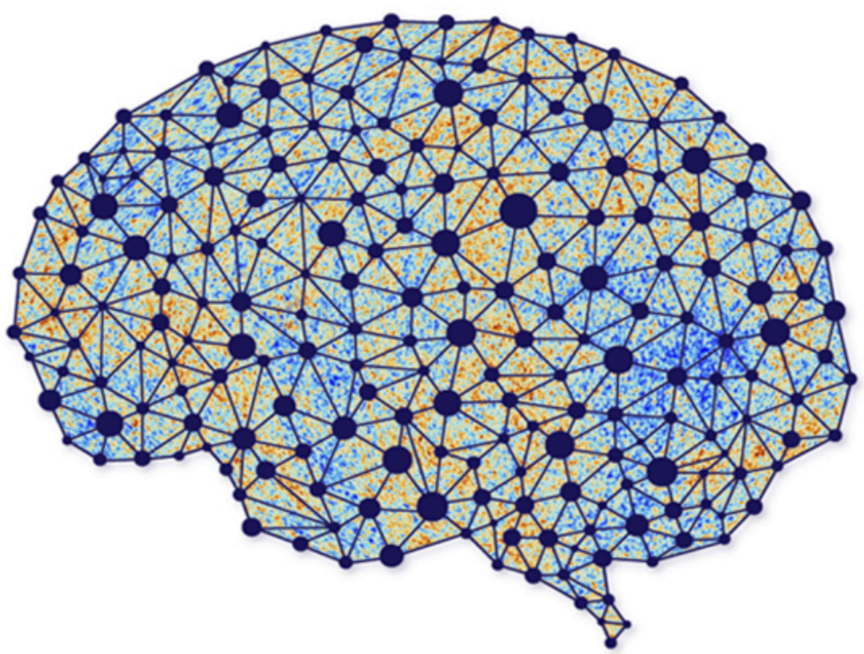
PHY 835: Machine Learning in Physics

Lecture 8: Shallow Neural Network

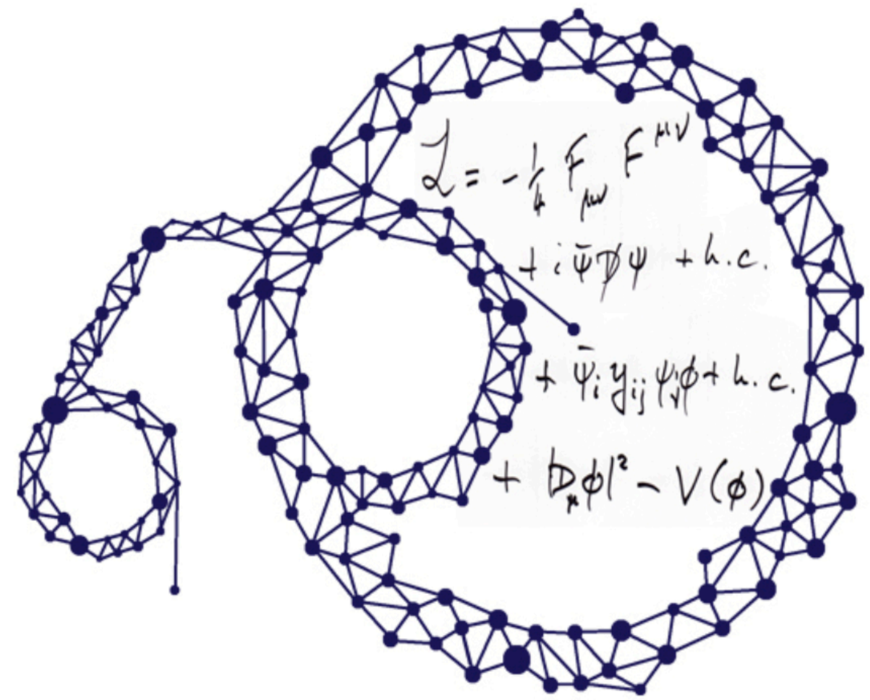
February 15, 2024



AI

∩

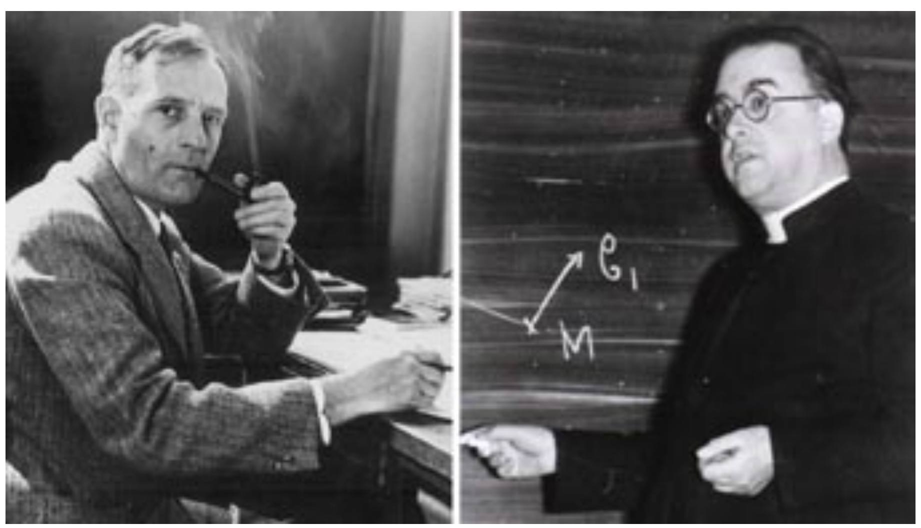
Universe

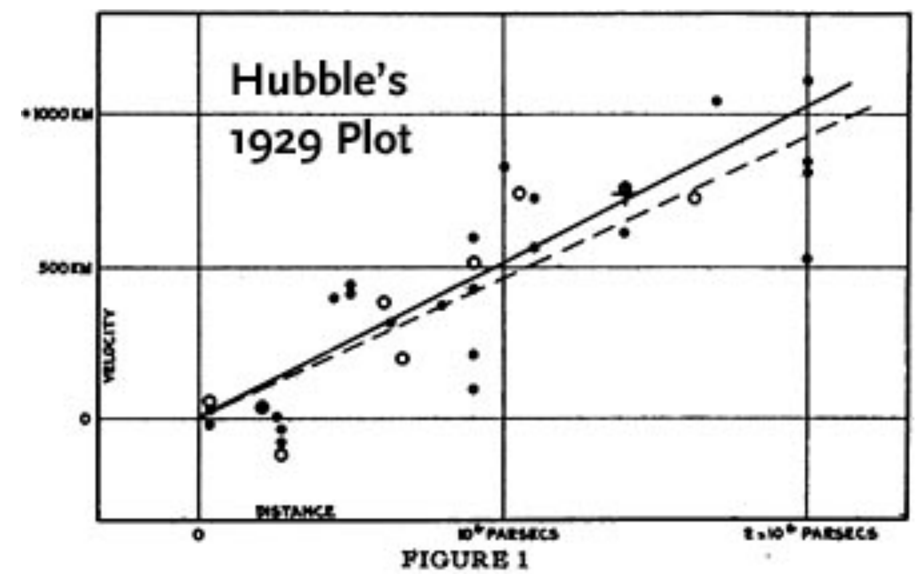


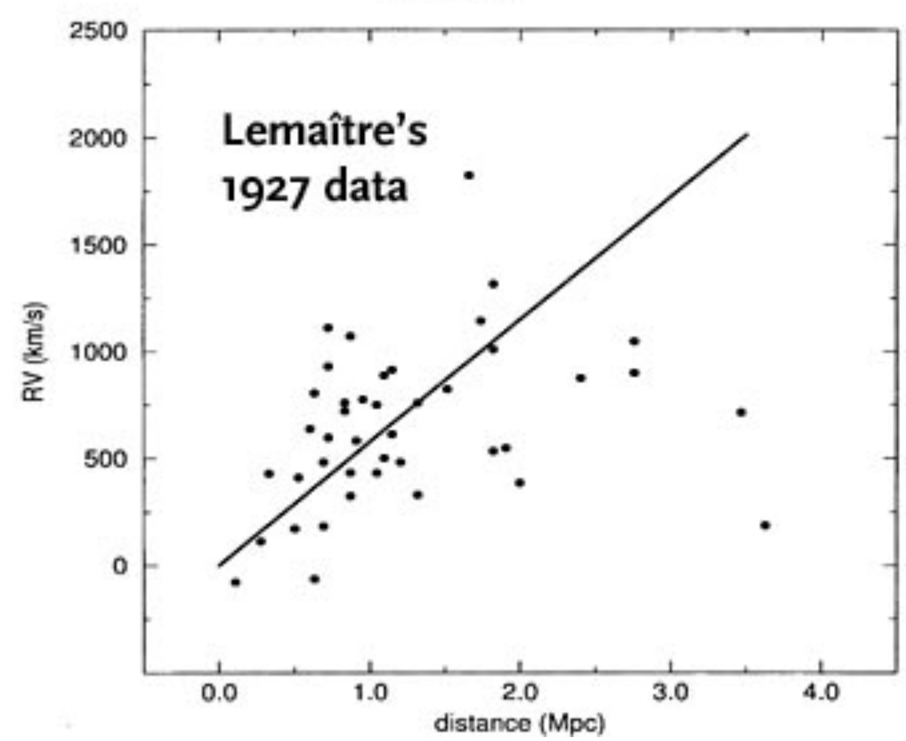
Gary Shiu

Linear Regression

- We have discussed linear regression in Lecture 4 which describes the input/output relationship as a line. A famous physics example is the **Hubble-Lemaitre Law**:







The linear relation between distance and recession velocity of galaxies, previously known as the Hubble law, has been renamed to the Hubble-Lemaitre Law:

https://www.iau.org/news/pressreleases/detail/iau1812/

Learning a function

- A NN is a parametrization of “big” (multivariate, non-linear) functions.

- Shallow NNs parametrize **piecewise linear functions** and are already expressive enough to approximate arbitrarily complex relationships between multi-dimensional inputs and outputs.

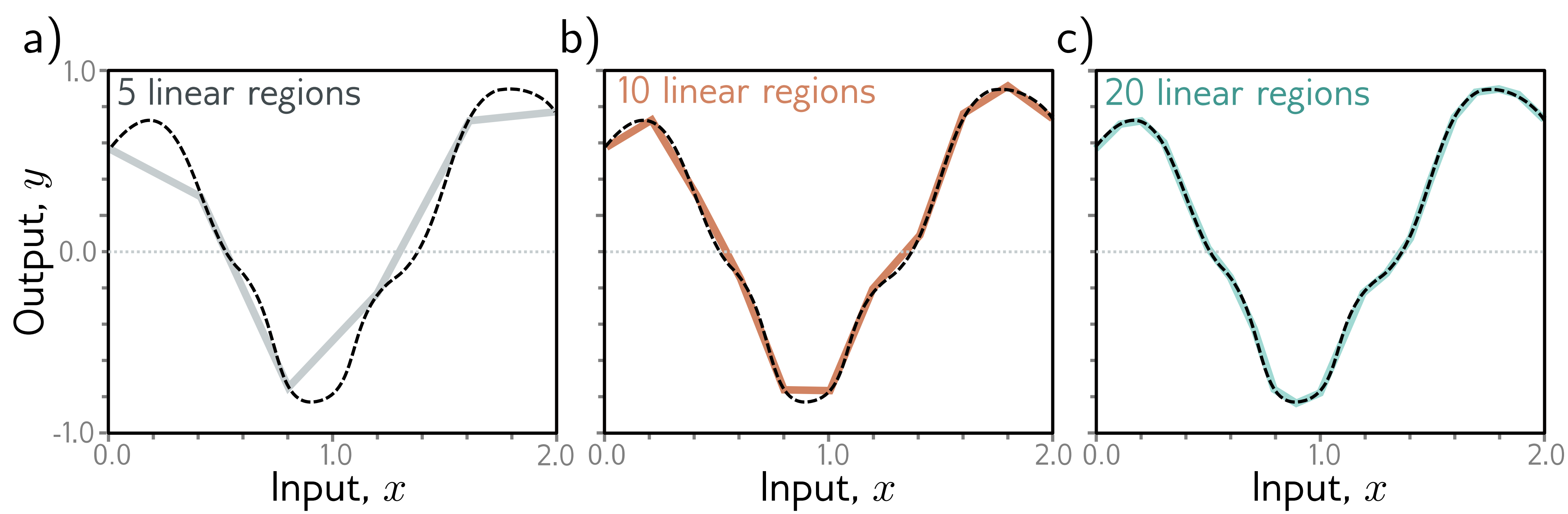


Figure from Simon Prince “Understanding Deep Learning"

Shallow Neural Networks

- Shallow neural networks are functions $\mathbf{y} = f(\mathbf{x}, \phi)$ with parameters ϕ that map multivariate inputs $\mathbf{x}$ to multivariate outputs $\mathbf{y}$.

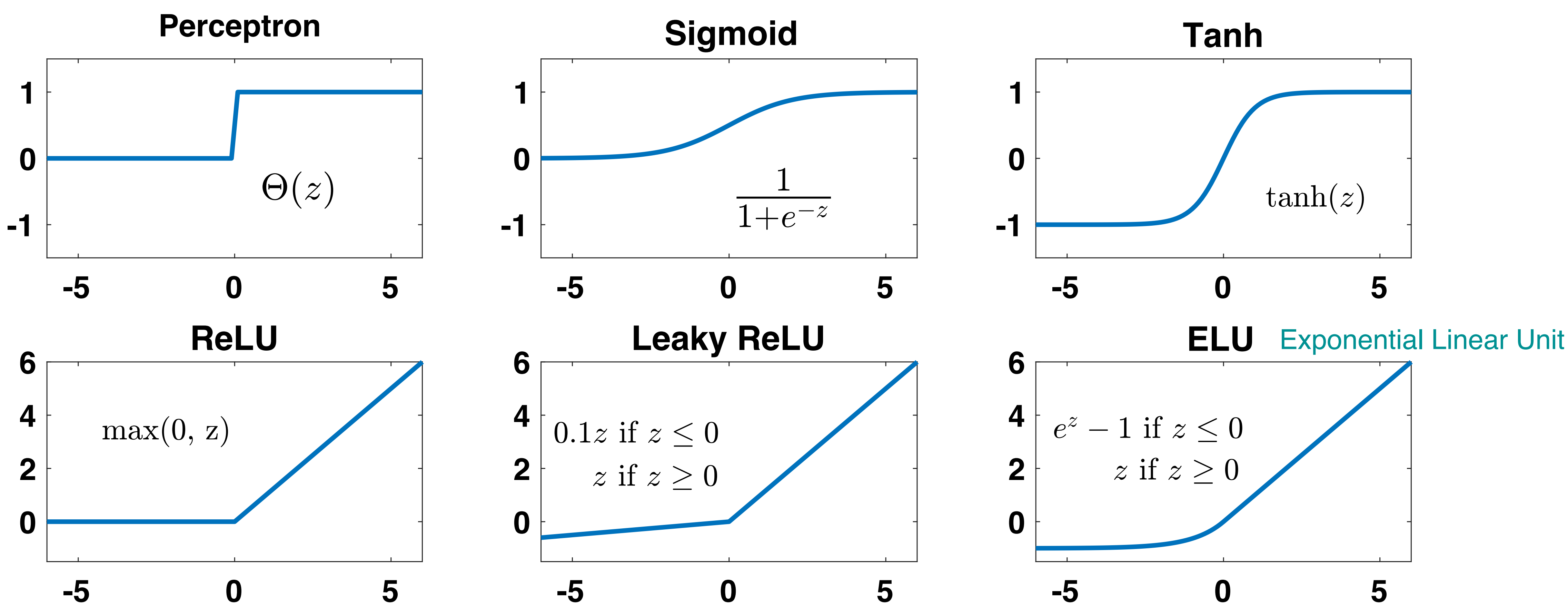
- As a warmup, consider $f(x, \phi)$ that maps a scalar input x to a scalar output y and has ten parameters $\phi = \{\phi_0, \phi_1, \phi_2, \phi_3, \theta_{10}, \theta_{11}, \theta_{20}, \theta_{21}, \theta_{30}, \theta_{31}\}$:

$$\begin{aligned} y &= \mathrm{f}[x, \boldsymbol{\phi}] \\ &= \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11} x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21} x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31} x]. \end{aligned}$$

- $a[\,\cdot\,]$ is known as the **activation function**. It cannot be a linear function in order for the NN to go beyond linear regression.

- Given a training dataset $\{x_i, y_i\}_{i=1}^{I}$, we can define a least squares loss function $L[\phi]$ to measure how effectively the model describes this dataset. To train the model, we find $\hat{\phi}$ that minimizes $L[\phi]$.

Activation Function



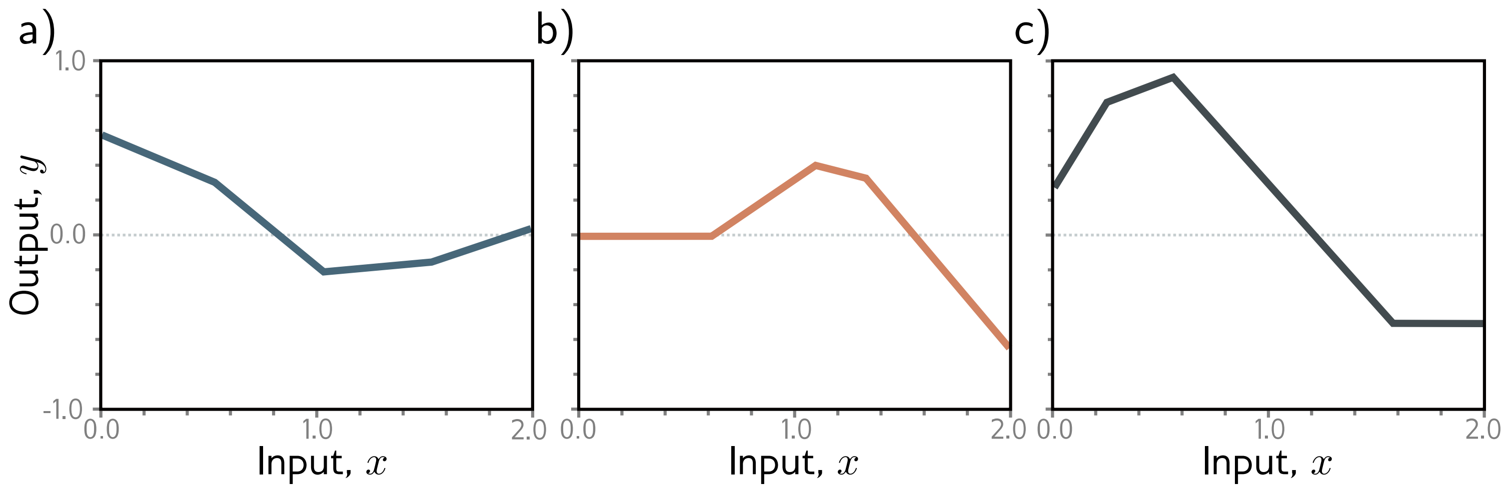
- For illustrative purpose, we consider the most common choice known as the **rectified linear unit** or ReLU:

$$
\mathrm{a}[z] = \mathrm{ReLU}[z] = \begin{cases} 0 & z < 0 \\ z & z \ge 0 \end{cases}.
$$

- Perceptron: derivative is either vanishing or infinite. Sigmoid & tanh: differentiable but have vanishing derivatives away from the origin.

NN Intuition

- In the ten-parameter example, we model the dataset with a family of continuous piecewise linear functions with up to 4 linear regions.



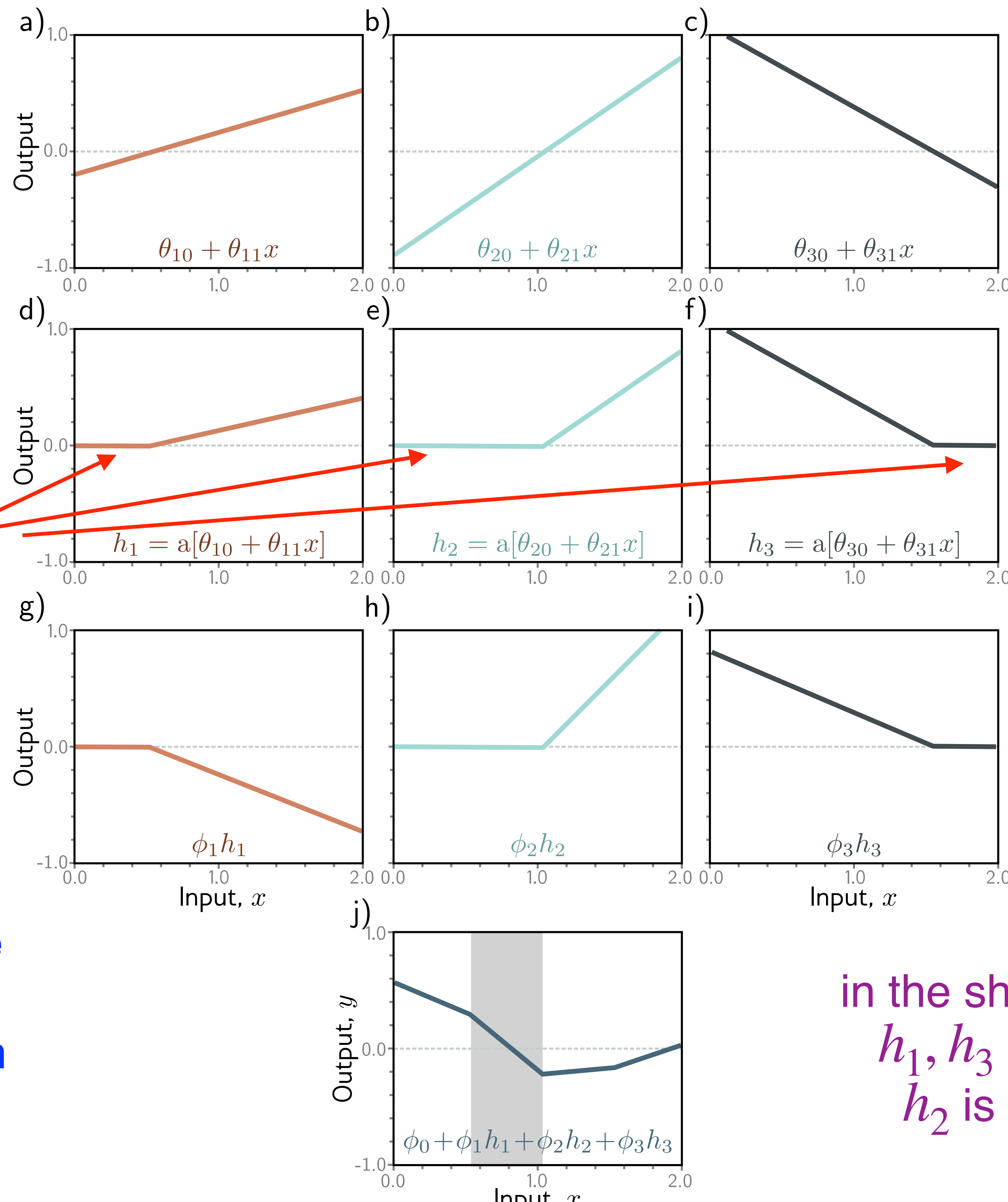
- To see why, we define the intermediate quantities as hidden units:

$$
\begin{aligned}
h_1 &= \mathrm{a}[\theta_{10} + \theta_{11}x] \\
h_2 &= \mathrm{a}[\theta_{20} + \theta_{21}x] \\
h_3 &= \mathrm{a}[\theta_{30} + \theta_{31}x],
\end{aligned}
$$

- The output is given by combining the hidden units w/ a linear function:

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3.$$

Activation Pattern



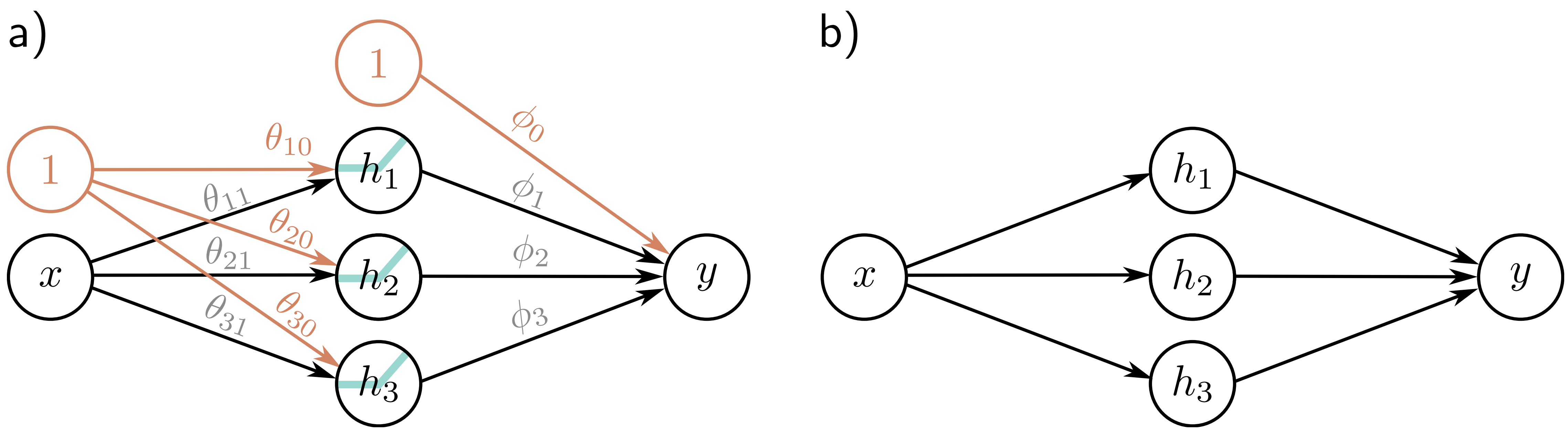
clipped by ReLU

define locations of the 3 joints
$\Rightarrow$ 4 linear regions

[Only 3 of the slopes are independent; the 4-th is either zero or sum of slopes from the other regions.]

in the shaded region
h_1, h_3 are **active**
h_2 is **inactive**.

Depicting Neural Networks



inputs hidden units outputs inputs hidden units outputs

- The intercepts (known as **biases**) are usually not shown in the NN architecture, the NN is simplified to the picture on the right.

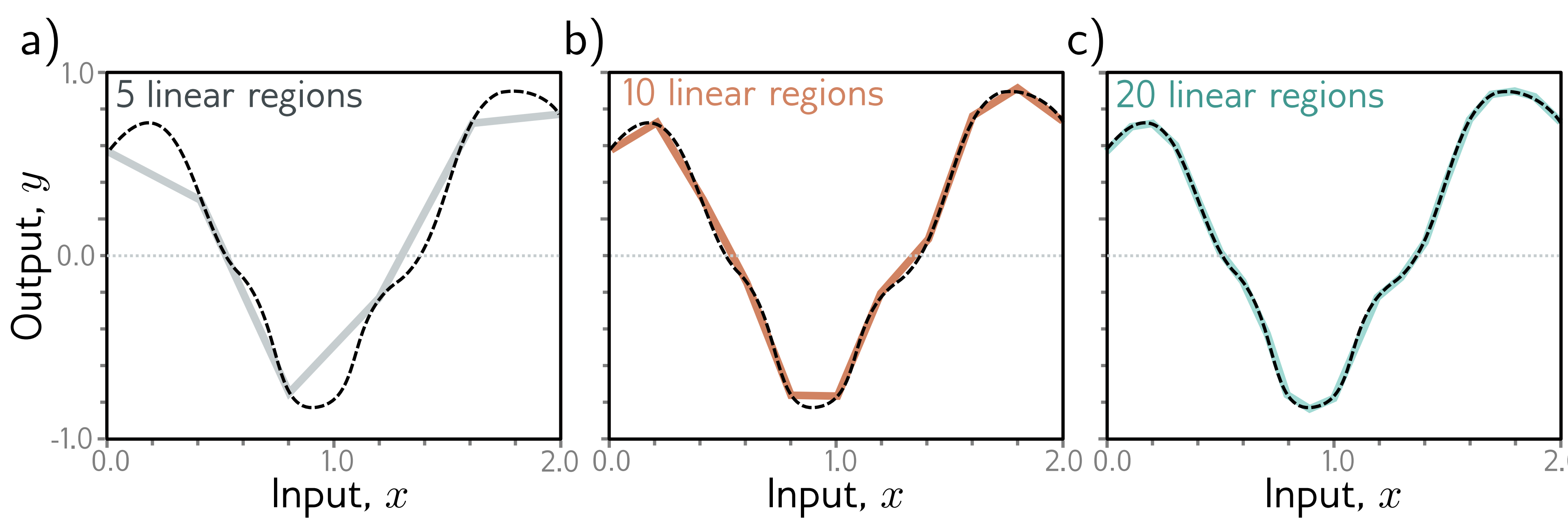
Universal Approximation Theorem

- Generalizing to D hidden units:

$$h_d = \text{a}[\theta_{d0} + \theta_{d1}x], \qquad y = \phi_0 + \sum_{d=1}^{D} \phi_d h_d.$$

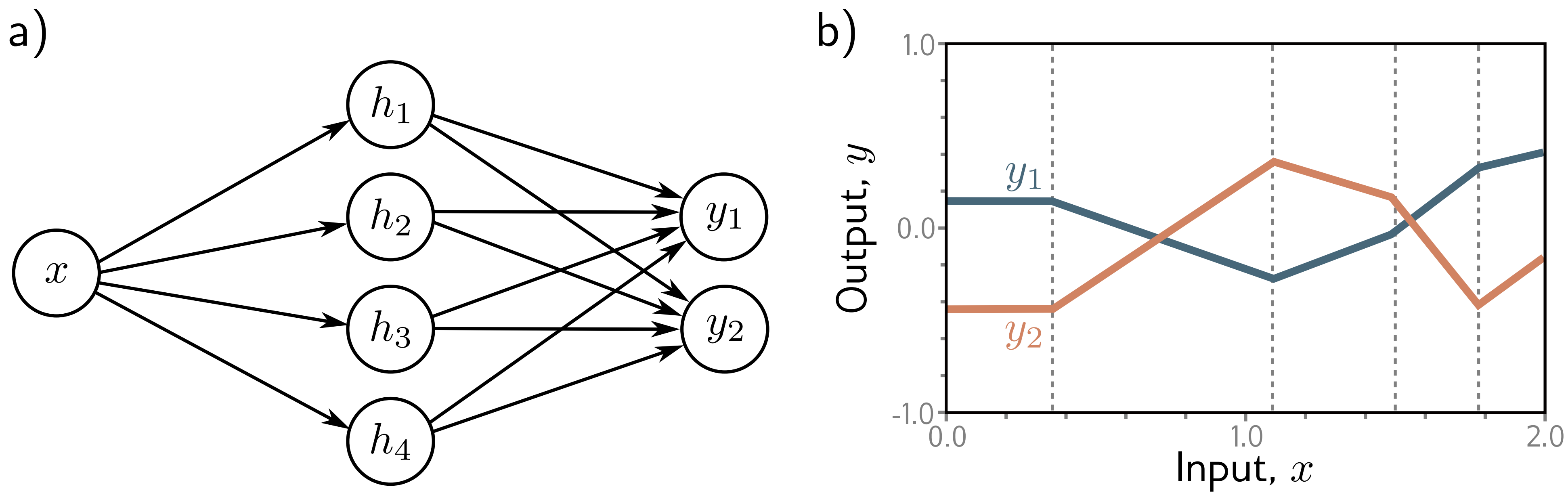
- D = **network capacity**; there are D joints and $D + 1$ linear regions.

- **Universal approximation theorem:** ∀ continuous function, ∃ a shallow network that can approximate it to any specified precision; holds for networks that map multivariate inputs to multivariate outputs.



Multivariate Outputs

- For example, $\mathbf{y} = [y_1, y_2]^T$:



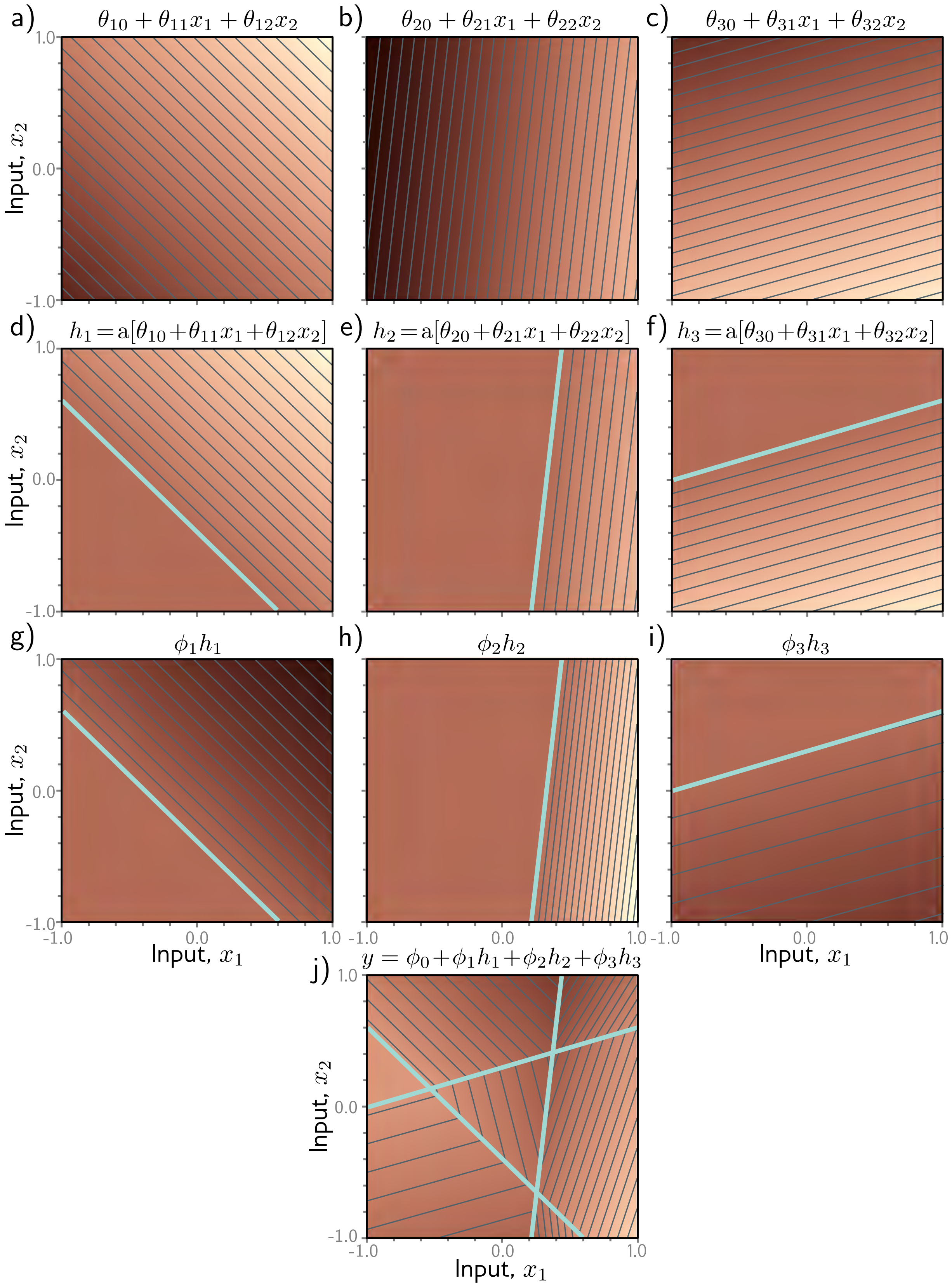
- The hidden units for both outputs are the same:

$$
\begin{aligned}
h_1 &= \mathrm{a}[\theta_{10} + \theta_{11}x] \\
h_2 &= \mathrm{a}[\theta_{20} + \theta_{21}x] \\
h_3 &= \mathrm{a}[\theta_{30} + \theta_{31}x] \\
h_4 &= \mathrm{a}[\theta_{40} + \theta_{41}x],
\end{aligned}
$$

- The joints are the same but the slopes of the linear regions and the vertical offsets can differ:

$$
\begin{aligned}
y_1 &= \phi_{10} + \phi_{11}h_1 + \phi_{12}h_2 + \phi_{13}h_3 + \phi_{14}h_4 \\
y_2 &= \phi_{20} + \phi_{21}h_1 + \phi_{22}h_2 + \phi_{23}h_3 + \phi_{24}h_4.
\end{aligned}
$$

Multivariate Inputs



The hidden units depend on both inputs

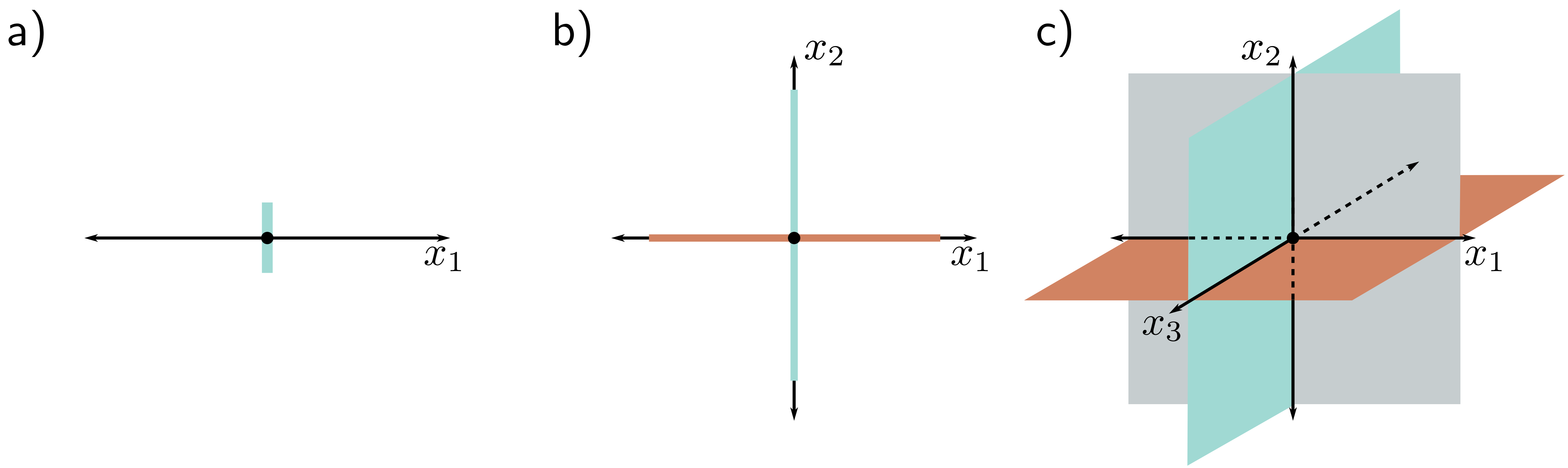
$$
\begin{aligned}
h_1 &= \text{a}[\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2] \\
h_2 &= \text{a}[\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2] \\
h_3 &= \text{a}[\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2],
\end{aligned}
$$

They create a continuous piecewise linear surface consisting of convex polygonal regions, each with a different activation patten.

Generalizable to more than 2 inputs but difficult to visualize such cases.

More linear regions

- If $D = D_i = \#$ input dimensions, can align the hyperplanes with the coordinate axes and show that there are 2^{D_i} orthants:



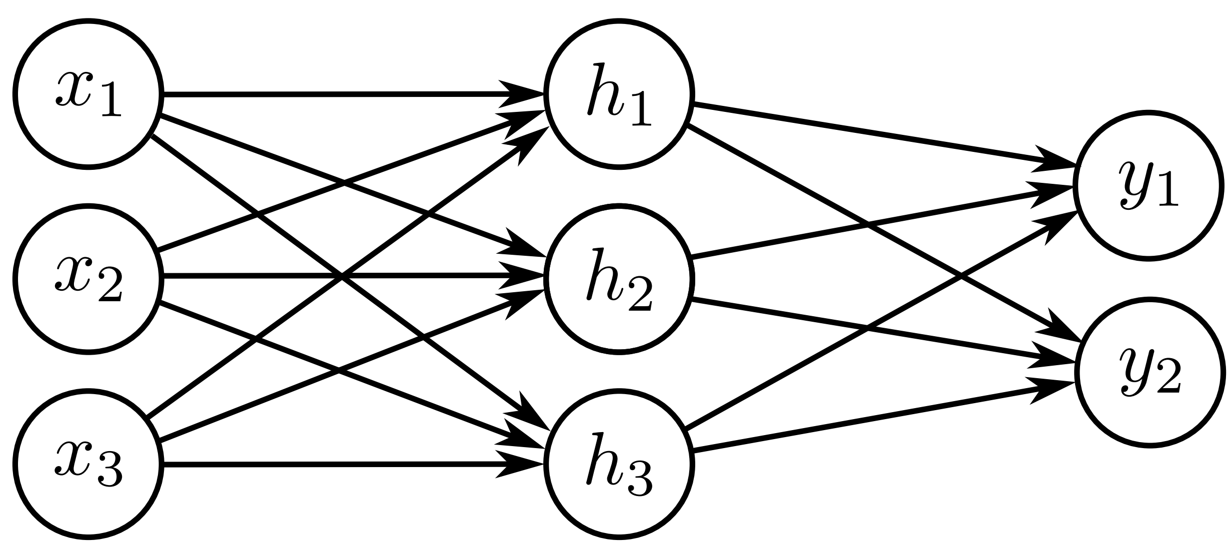
- Shallow neural networks usually have more hidden units than input dimensions, so they typically create more than 2^{D_i} linear regions.

General Case

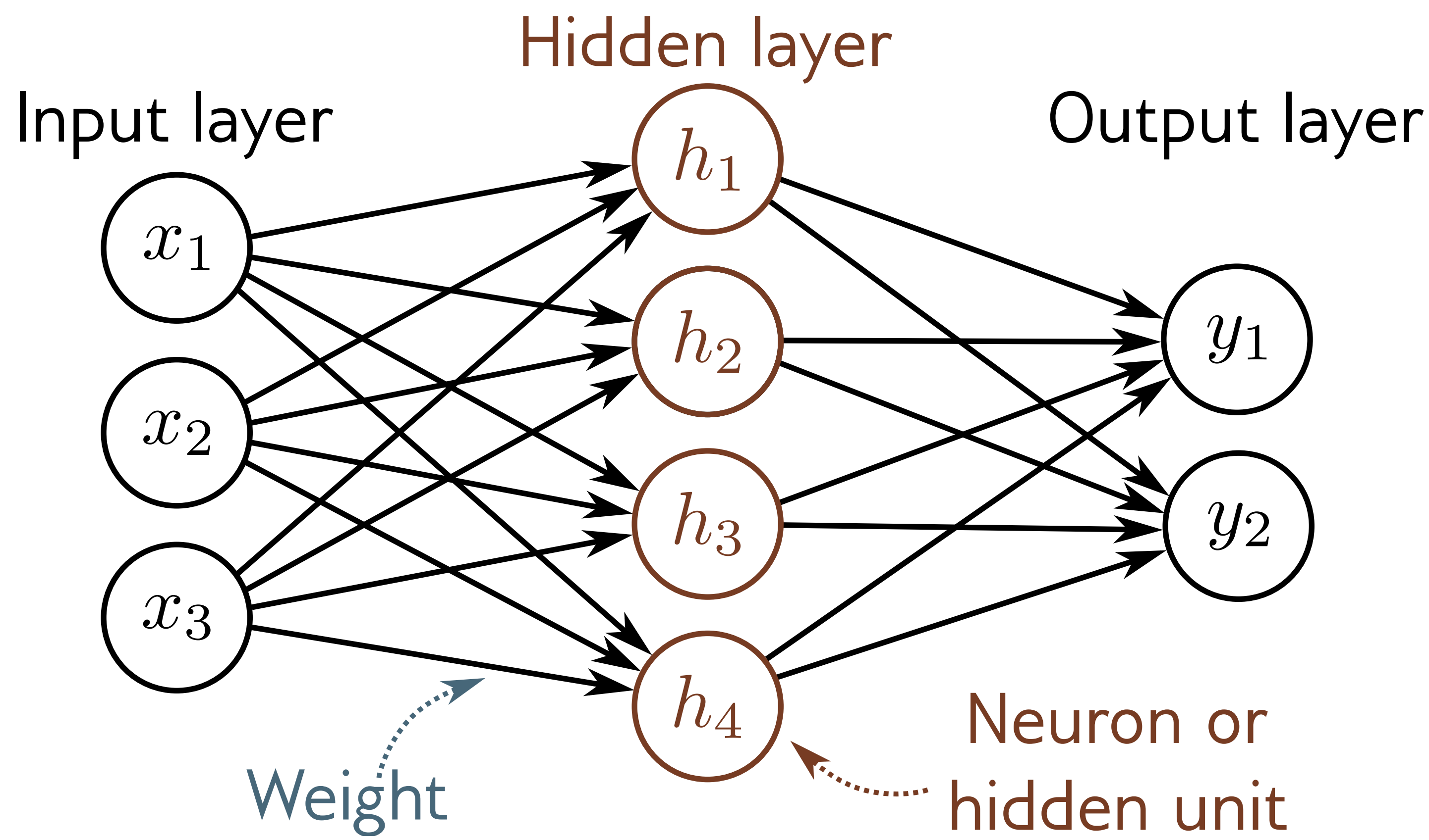
- In general, a shallow NN is a function $\mathbf{y} = f(\mathbf{x}, \boldsymbol{\phi})$ that maps a multi-dimensional input $\mathbf{x} \in \mathbb{R}^{D_i}$ to a multi-dimensional output $\mathbf{y} \in \mathbb{R}^{D_0}$ using $\mathbf{h} \in \mathbb{R}^{D}$ hidden units:

$$h_d = \mathrm{a}\left[\theta_{d0} + \sum_{i=1}^{D_i} \theta_{di} x_i\right], \qquad y_j = \phi_{j0} + \sum_{d=1}^{D} \phi_{jd} h_d,$$

- Graphically, a shallow NN is depicted as e.g.



Terminology



- Any NN with at least one hidden layer is called a multi-layer perceptron, or MLP.
- NNs with one hidden layer are called shallow NNs. NNs with multiple hidden layers are called deep NNs.
- NNs with connections form an acyclic graph (a graph w/0 loops) are feedforward NNs.
- Every element in one layer connects to every element in the next: fully connected NNs.