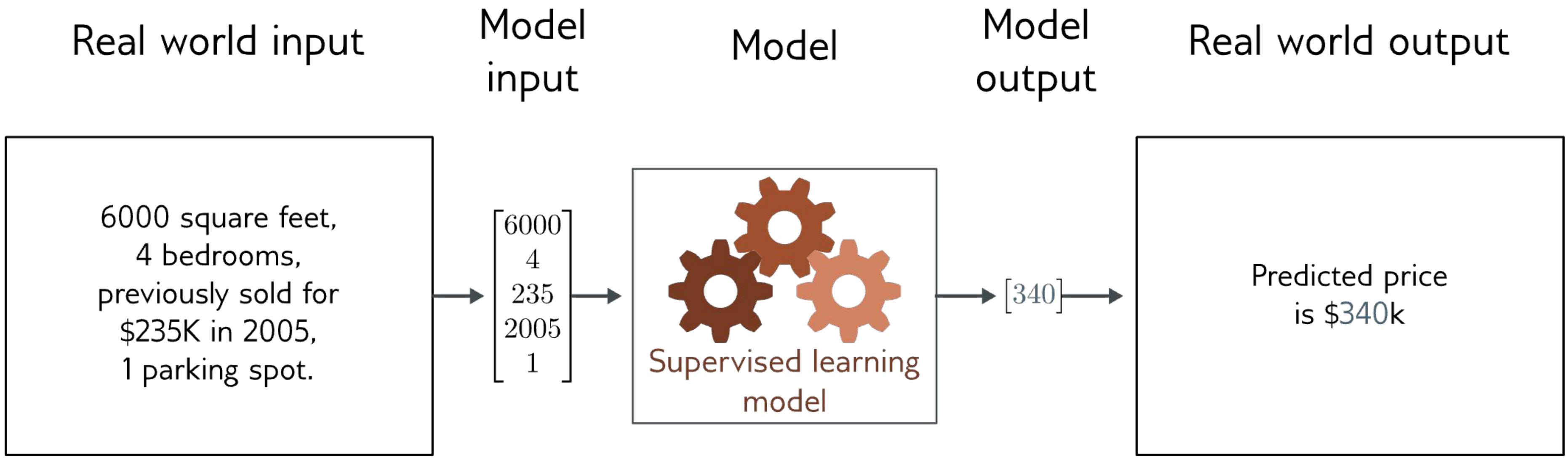
Deep Learning for Data Science
DS 542

Lecture 03
Shallow Neural Networks

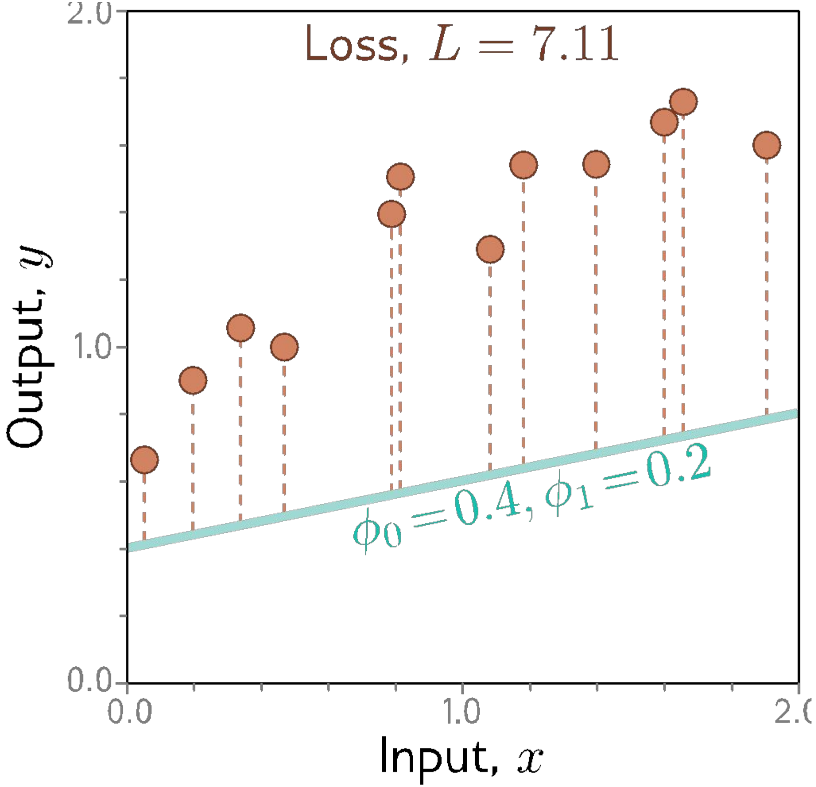
Slides originally by Thomas Gardos.

Recap: Regression



- Univariate regression problem (one output, real value)
- Fully connected network

Recap: 1D Linear regression loss function



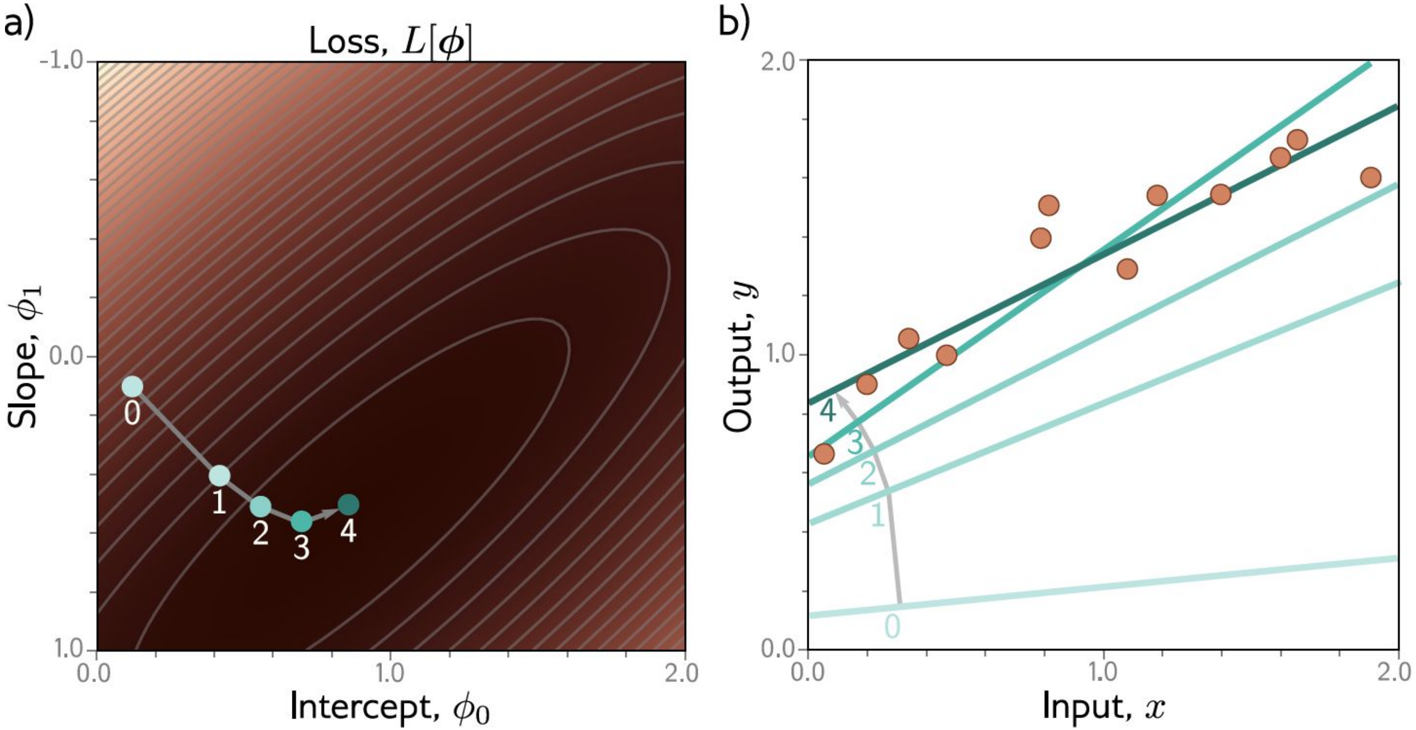
Loss function:

$$L[\phi] = \sum_{i=1}^{I}(\mathrm{f}[x_i, \phi] - y_i)^2$$

$$= \sum_{i=1}^{I}(\phi_0 + \phi_1 x_i - y_i)^2$$

"Least squares loss function"

Recap: 1D Linear regression training



This technique is known as gradient descent

Shallow neural networks

- 1D regression model is obviously limited
  - Limited to lines.
  - Only one input, and output
- General linear regression still limited
  - Limited to lines/planes/hyperplanes… only flat surfaces
  - Multiple inputs, only one output
  - At least it has an analytical solution?
- Shallow neural networks
  - Flexible enough to describe arbitrarily complex input/output mappings
  - Can have as many inputs as we want
  - Can have as many outputs as we want

This lecture we'll cover…

- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology
- Universal approximation HOWTO

1D Linear Regression vs 0 Hidden vs 1 Hidden

1D Linear Regression

- $y = f[x, \phi] = \phi_0 + \phi_1 x$

Shallow Neural Network with no hidden layers

- $y = f[x, \phi] = a[\phi_0 + \phi_1 x]$ a is an "activation function"

Shallow Neural Network with one hidden layer

- $y = f[x, \phi] = a_0[\phi_0 + \phi_1\ a_1[\theta_{10} + \theta_{11}x] + \phi_1\ a_2[\theta_{20} + \theta_{21}x] + \phi_1\ a_3[\theta_{30} + \theta_{31}x]]$

  usually skip a_0 (identity function) and use $a_1 = a_2 = a_3$

Example shallow network

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

Example shallow network

Activation function

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

Example shallow network

Activation function

$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$

$$= \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11}x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21}x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31}x]$$

If a is the identity function, $\mathrm{a}[x] = x$, then this simplifies to a linear function.

$$y = (\phi_0 + \phi_1\theta_{10} + \phi_1\theta_{20} + \phi_1\theta_{30}) + (\phi_1\theta_{11} + \phi_1\theta_{21} + \phi_1\theta_{31})\, x$$

So the activation functions are a critical part of neural network capability.

Example shallow network

Activation function

$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$

$$= \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11} x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21} x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31} x]$$

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

Rectified Linear Unit

(a very common activation function)

Example shallow network

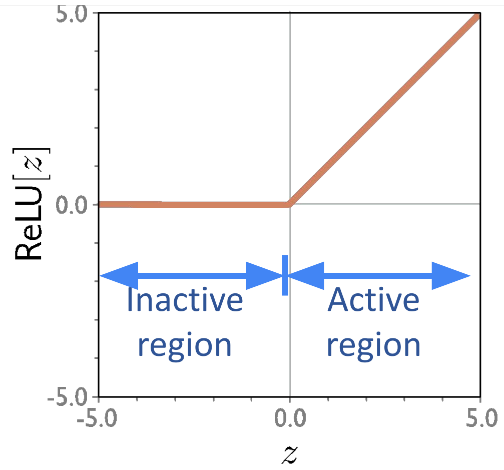
Activation function

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

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

Rectified Linear Unit

(a very common activation function)



Example shallow network

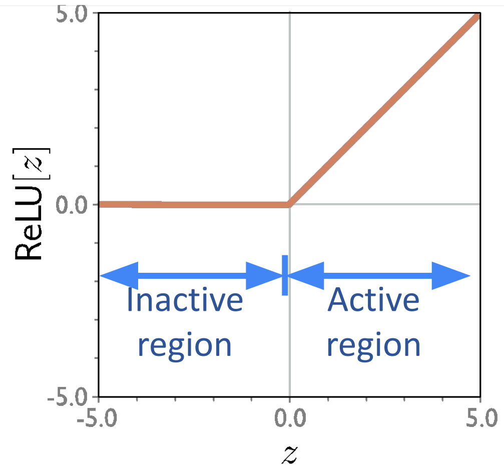
Activation function

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

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

Rectified Linear Unit

(a very common activation function)



Easy gradients - zero or one depending on activity.

Example shallow network

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

This model has 10 parameters:

$$\boldsymbol{\phi} = \{\phi_0, \phi_1, \phi_2, \phi_3, \theta_{10}, \theta_{11}, \theta_{20}, \theta_{21}, \theta_{30}, \theta_{31}\}$$

- Represents a family of functions
- Parameters determine a particular function
- Given the parameters, we can perform inference (evaluate the equation)

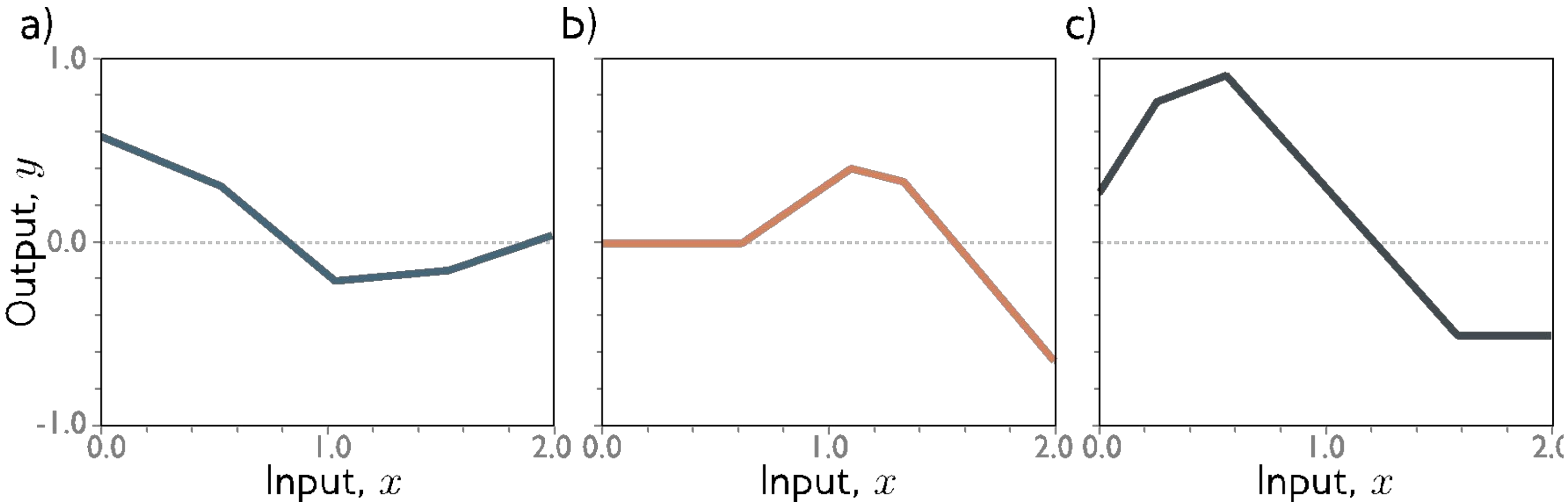
- Given training dataset $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}$
- Choose loss function $L[\boldsymbol{\phi}]$ (initially least squares)
- Change parameters to minimize loss function

Example shallow network

$$y = \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11}x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21}x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31}x].$$

Example shallow network

$$y = \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11}x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21}x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31}x].$$



Piecewise linear functions with three joints

Hidden units

$$y = \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11}x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21}x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31}x].$$

Break down into two parts:

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

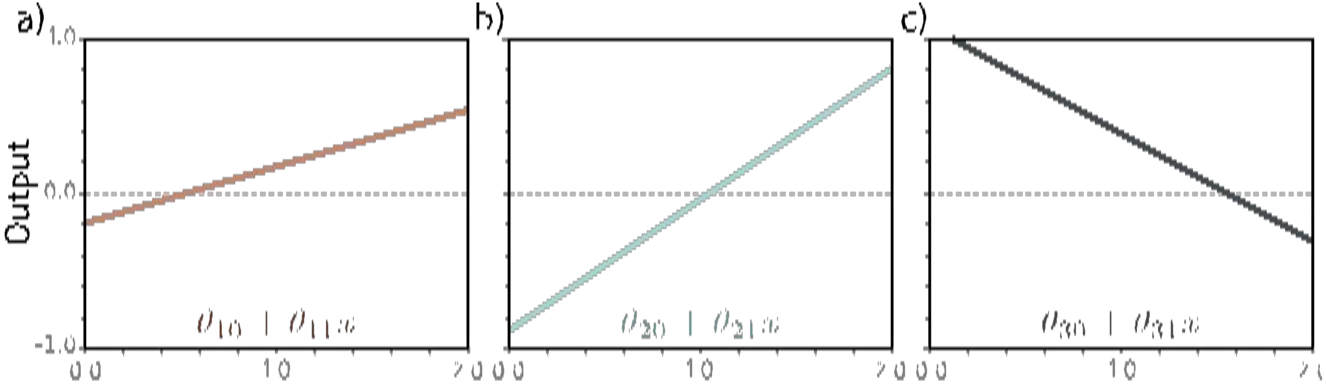
where:

Hidden units

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

1. compute three linear functions

Linear Functions

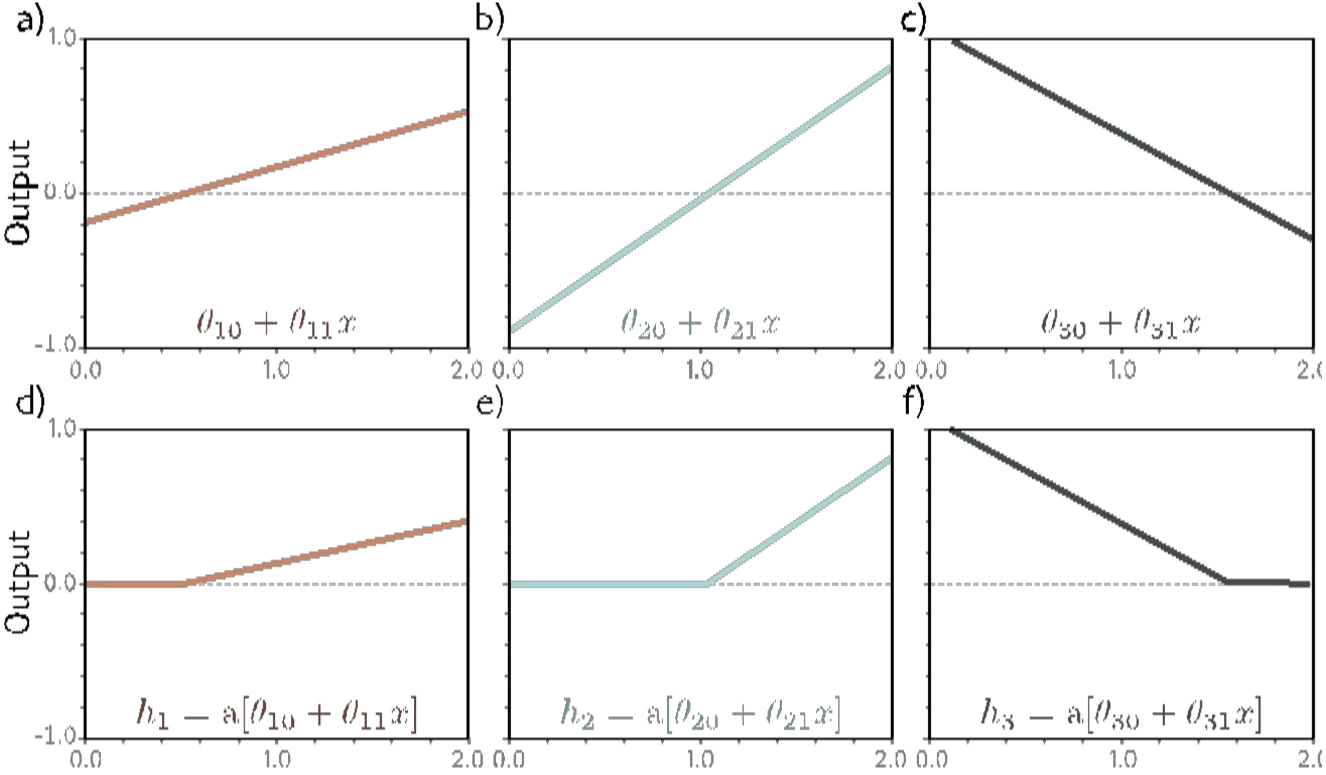


2. Pass through ReLU functions (creates hidden units)

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

Linear Functions

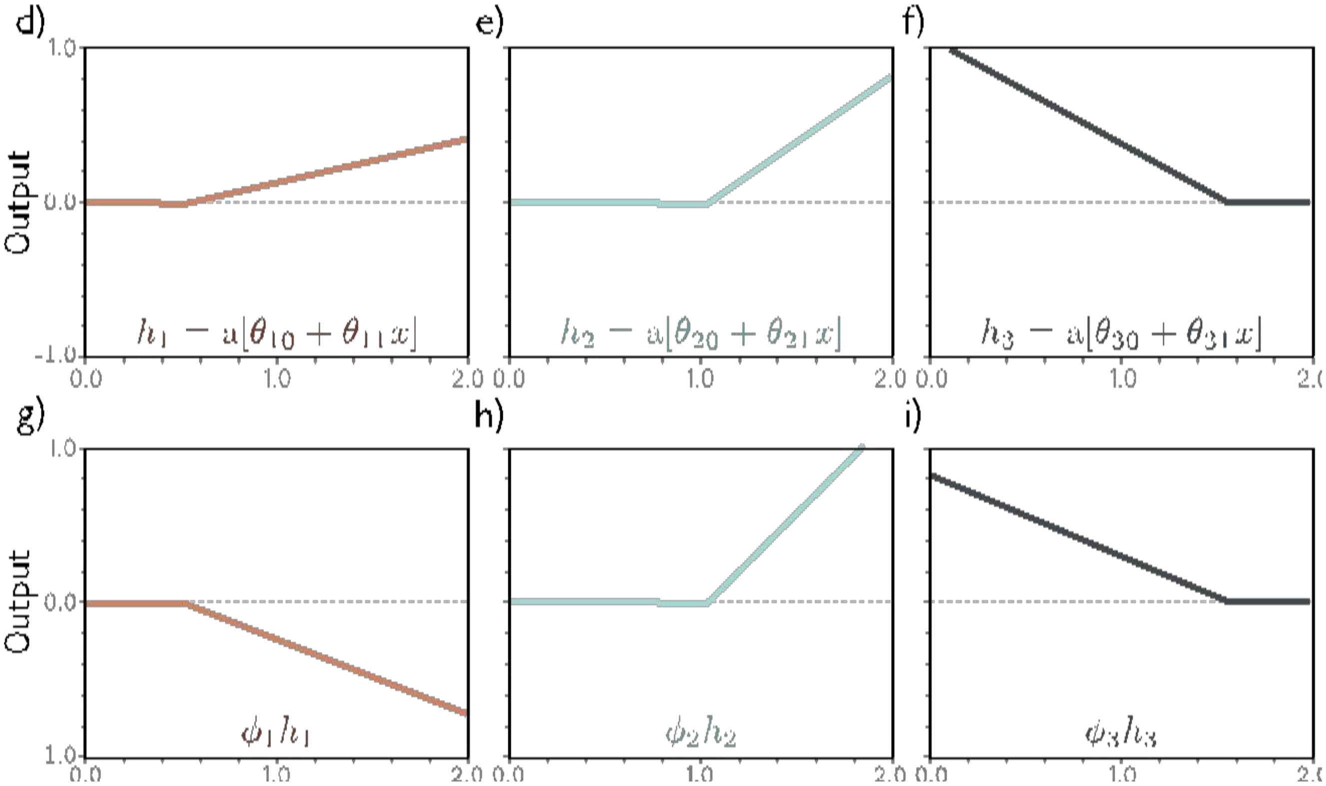
After Activation



3. Weight the hidden units

After Activation

Weight the Hidden units

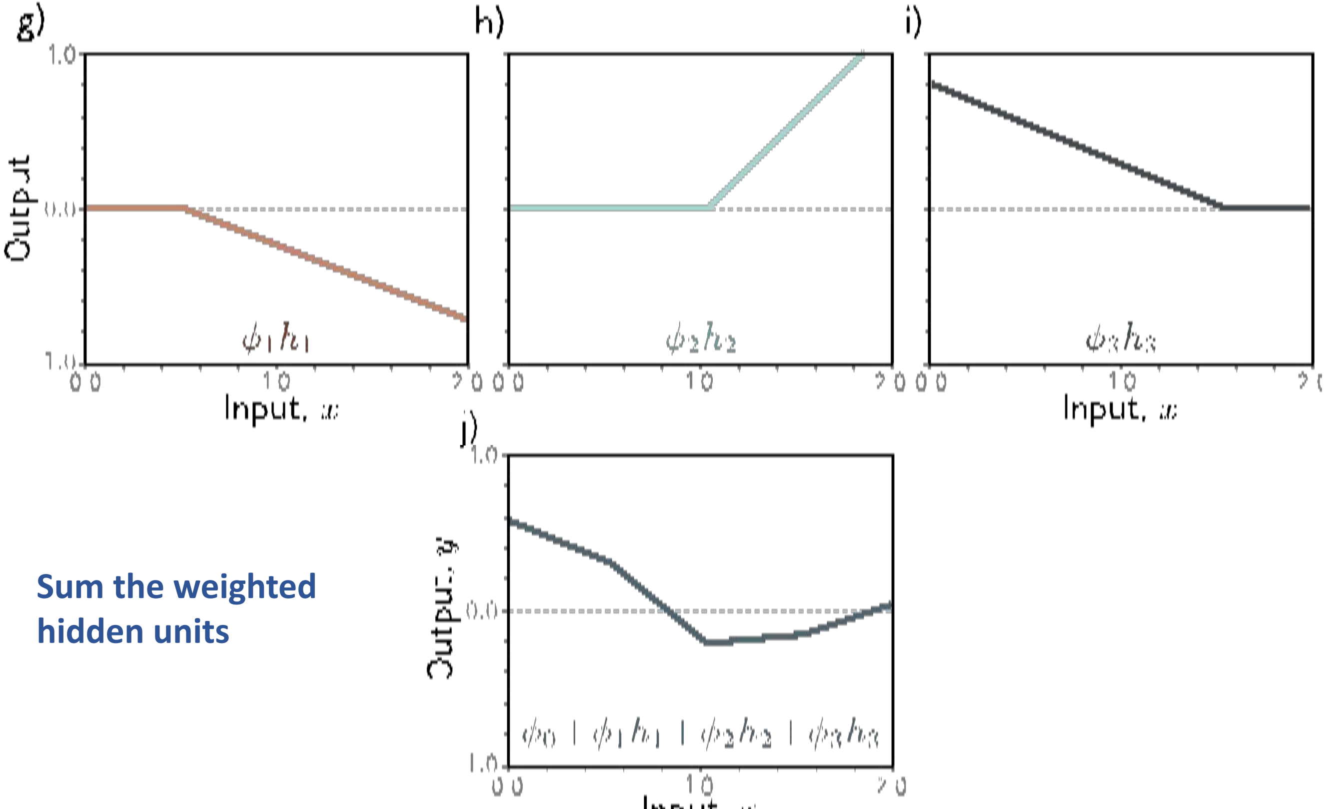


4. Sum the weighted hidden units to create output

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

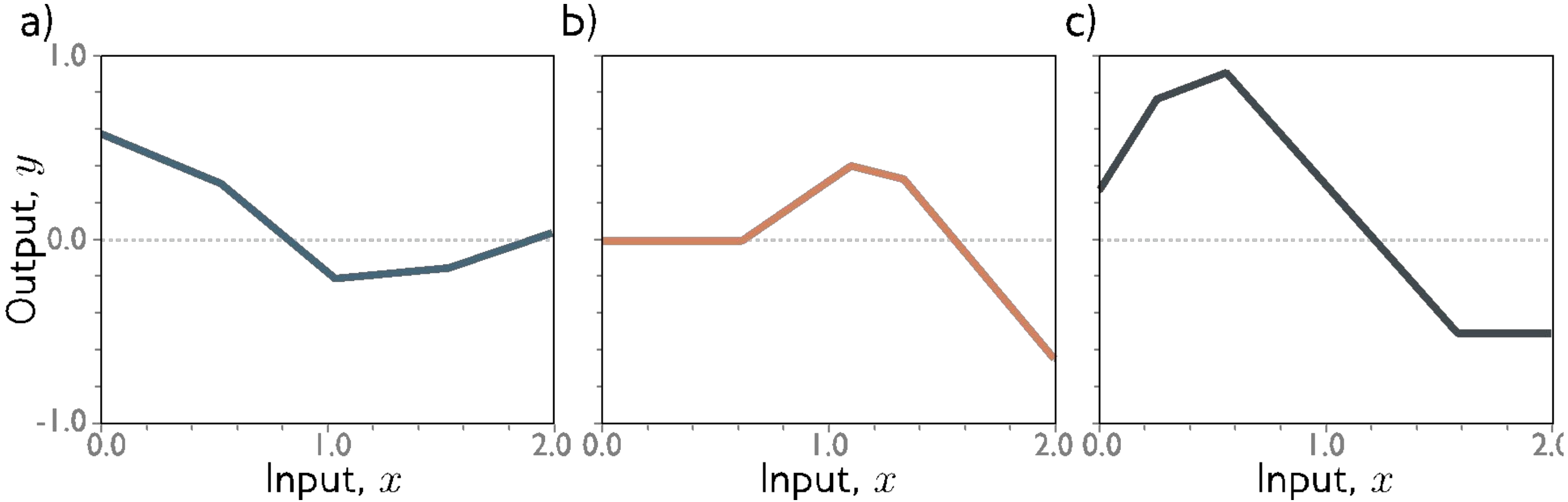
Weight the hidden units

Sum the weighted hidden units



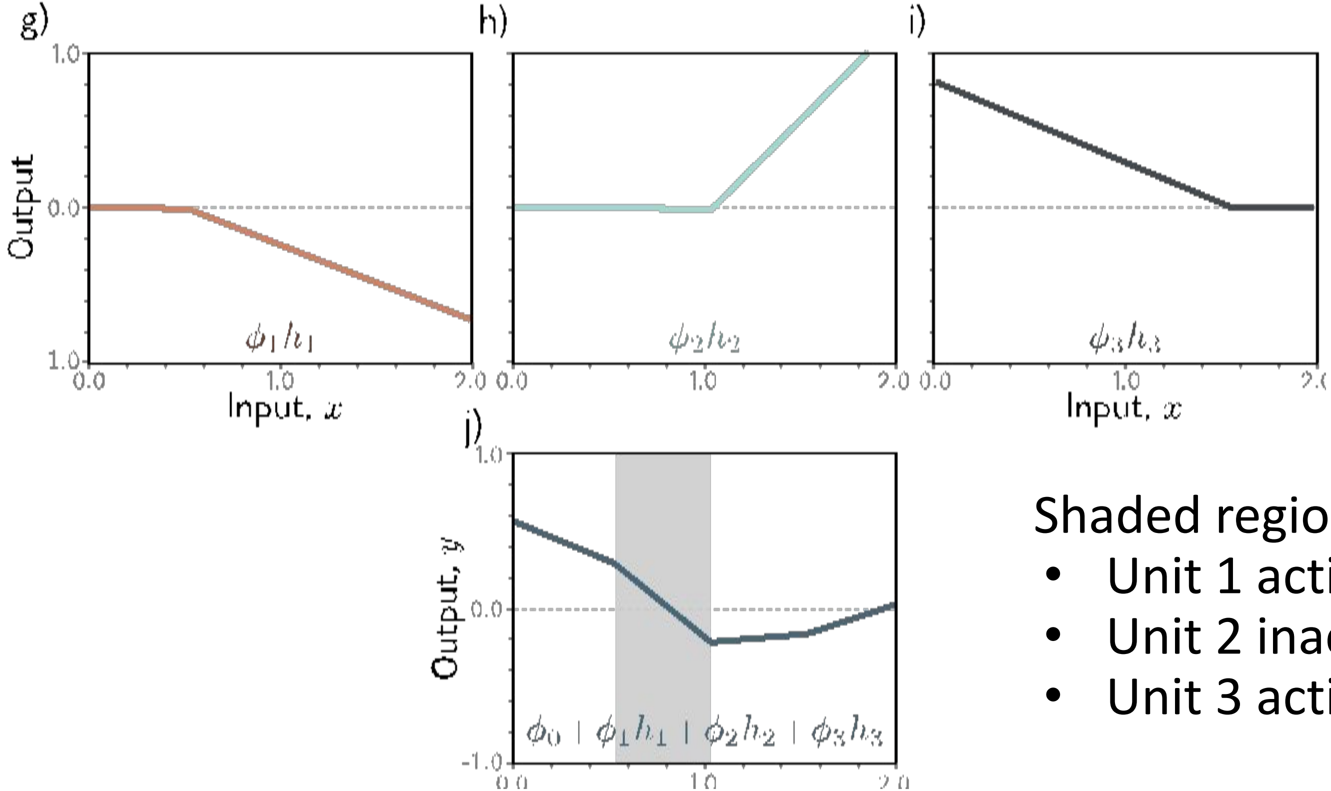
Example: 3 different shallow networks

$$y = \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11}x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21}x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31}x].$$



Example shallow network = piecewise linear functions
1 “joint” per ReLU function

Activation pattern = which hidden units are activated



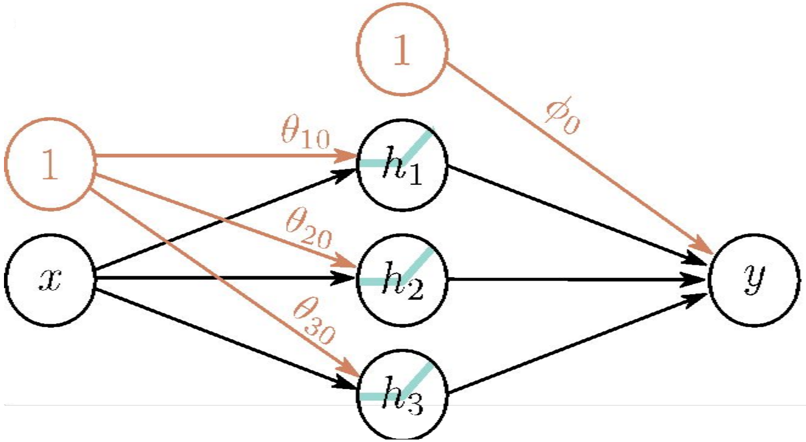
Shaded region:

- Unit 1 active
- Unit 2 inactive
- Unit 3 active

Depicting neural networks

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

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



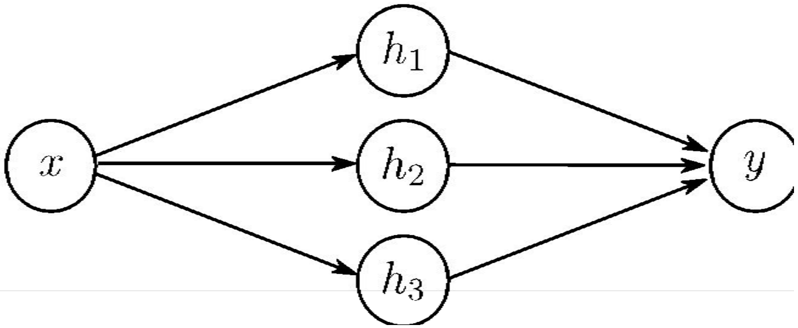
Each parameter multiplies its source and adds to its target

Depicting neural networks

Usually don't show the bias terms

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

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



Shallow neural networks

- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology
- Universal approximation HOWTO

With 3 hidden units:

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

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

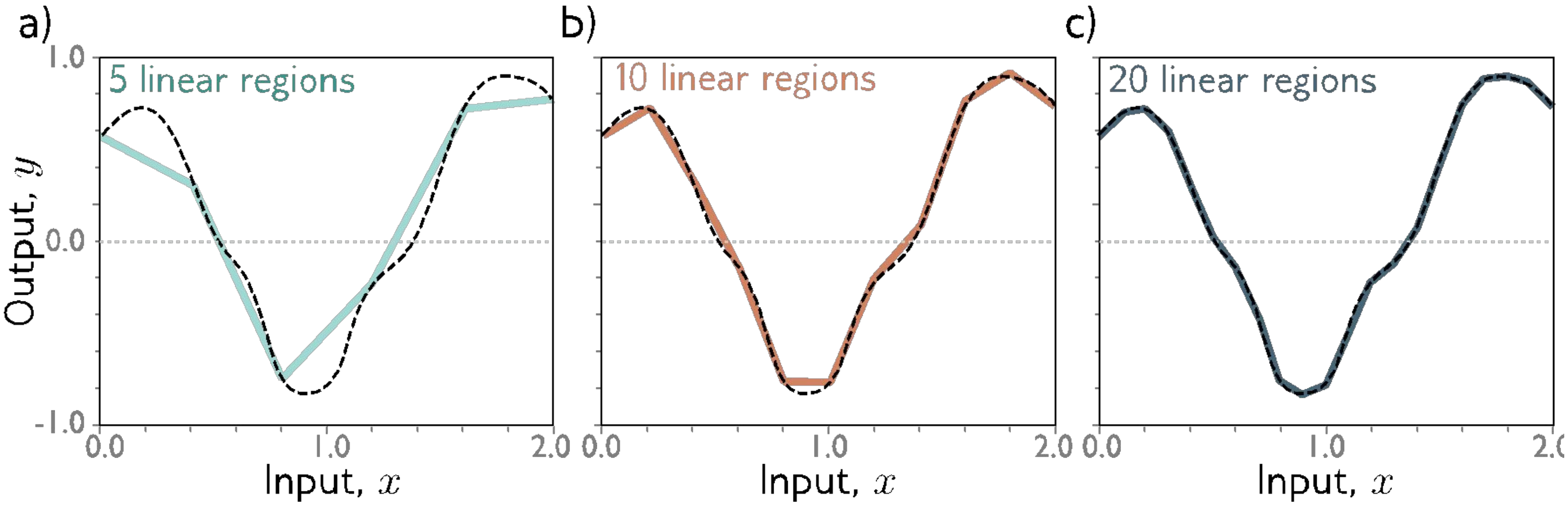
With D hidden units:

$$h_d = \mathrm{a}[\theta_{d0} + \theta_{d1}x]$$

$$y = \phi_0 + \sum_{d=1}^{D} \phi_d h_d$$

With enough hidden units…

… we can describe any 1D function to arbitrary accuracy



Universal approximation theorem

"a formal proof that, with enough hidden units, a shallow neural network can describe any continuous function in $\mathbb{R}^D$ to arbitrary precision"

Will circle back to this at the end to show how this works.

Shallow neural networks

- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology
- Universal approximation HOWTO

Two outputs

- 1 input, 4 hidden units, 2 outputs

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

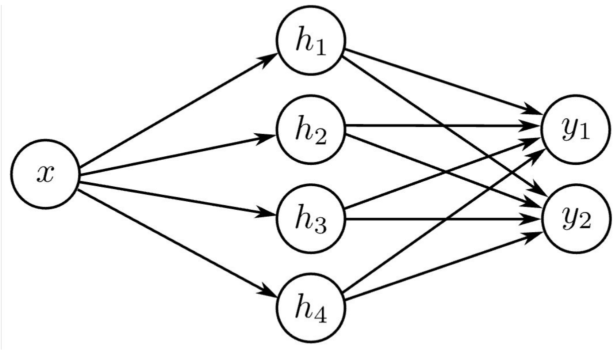
$$y_1 = \phi_{10} + \phi_{11}h_1 + \phi_{12}h_2 + \phi_{13}h_3 + \phi_{14}h_4$$
$$y_2 = \phi_{20} + \phi_{21}h_1 + \phi_{22}h_2 + \phi_{23}h_3 + \phi_{24}h_4$$

Two outputs

- 1 input, 4 hidden units, 2 outputs

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

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

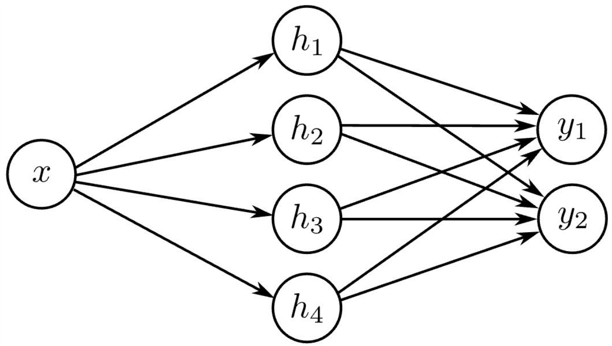


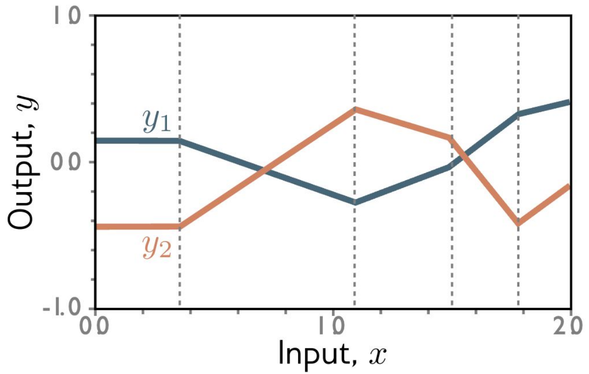
Two outputs

- 1 input, 4 hidden units, 2 outputs

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

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





Shallow neural networks

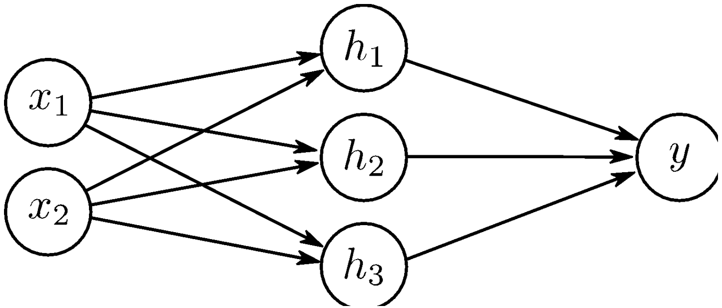
- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology
- Universal approximation HOWTO

Two inputs

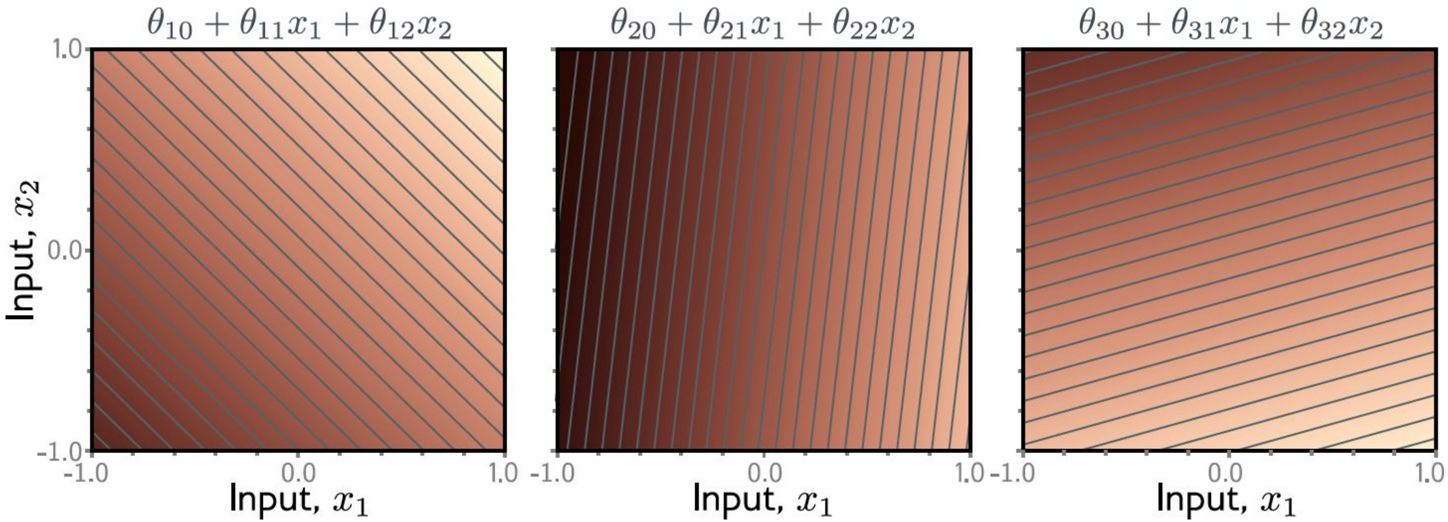
- 2 inputs, 3 hidden units, 1 output

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2]$$

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

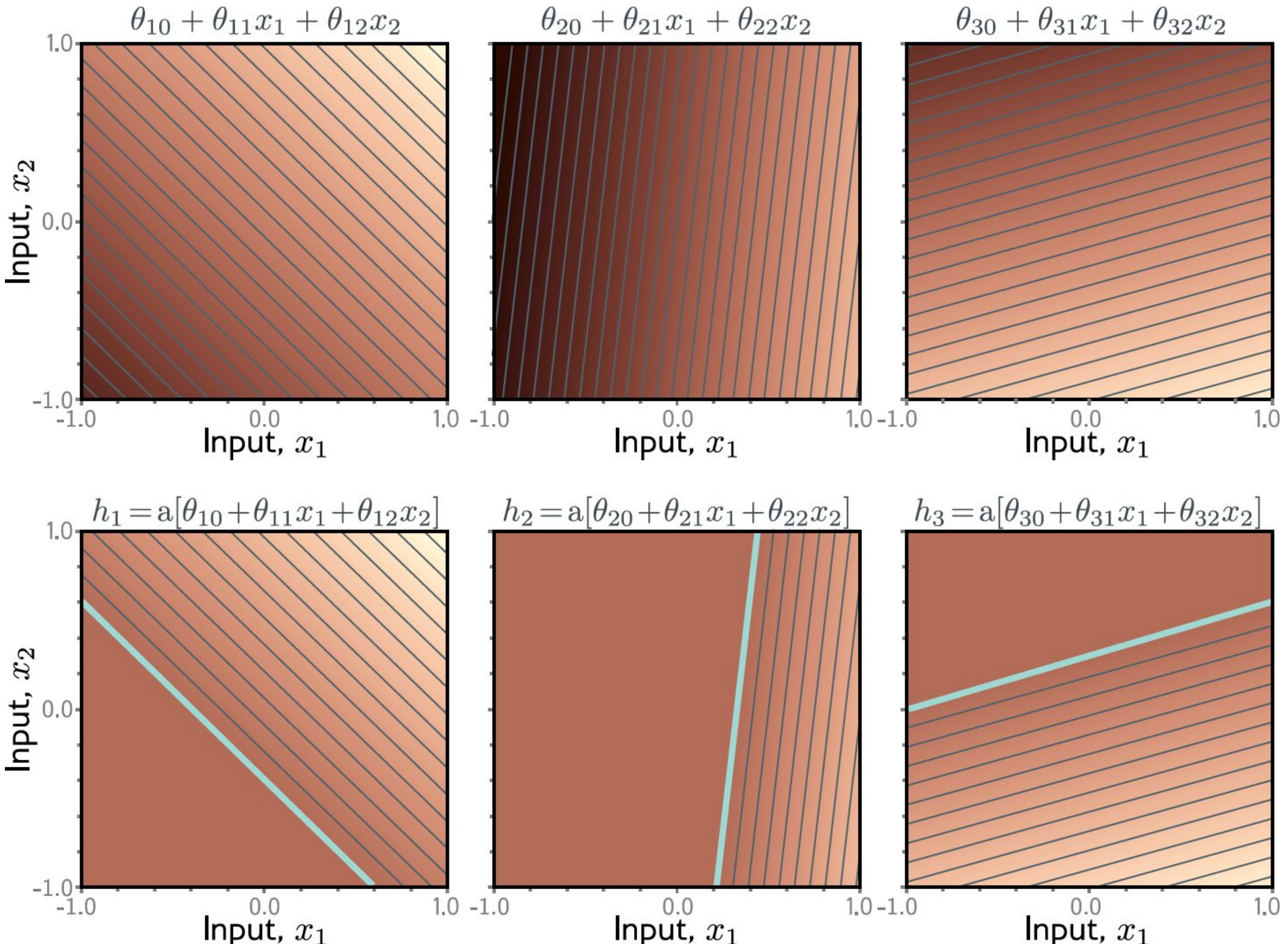


Linear Functions

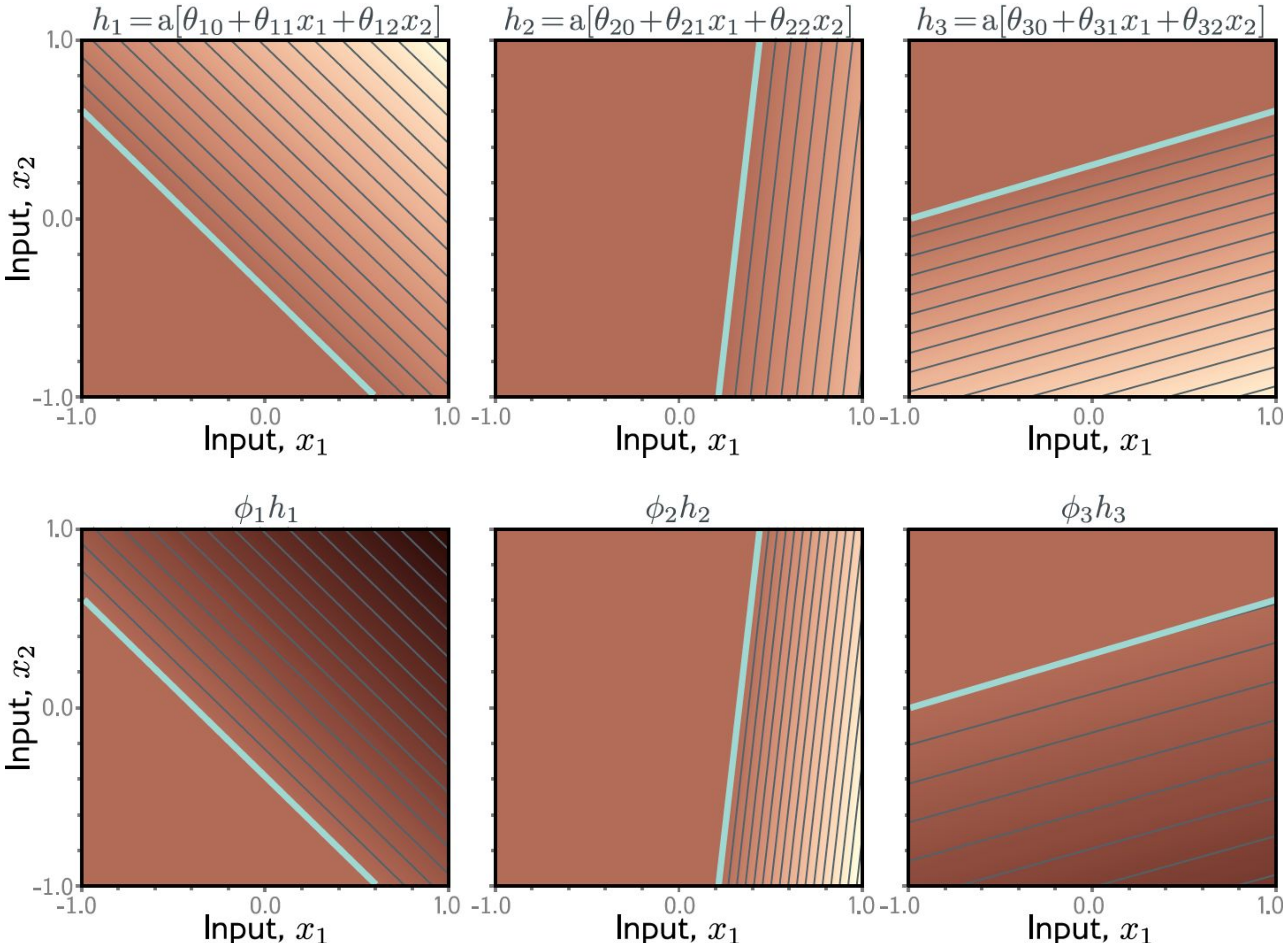


Linear Functions

After Activation



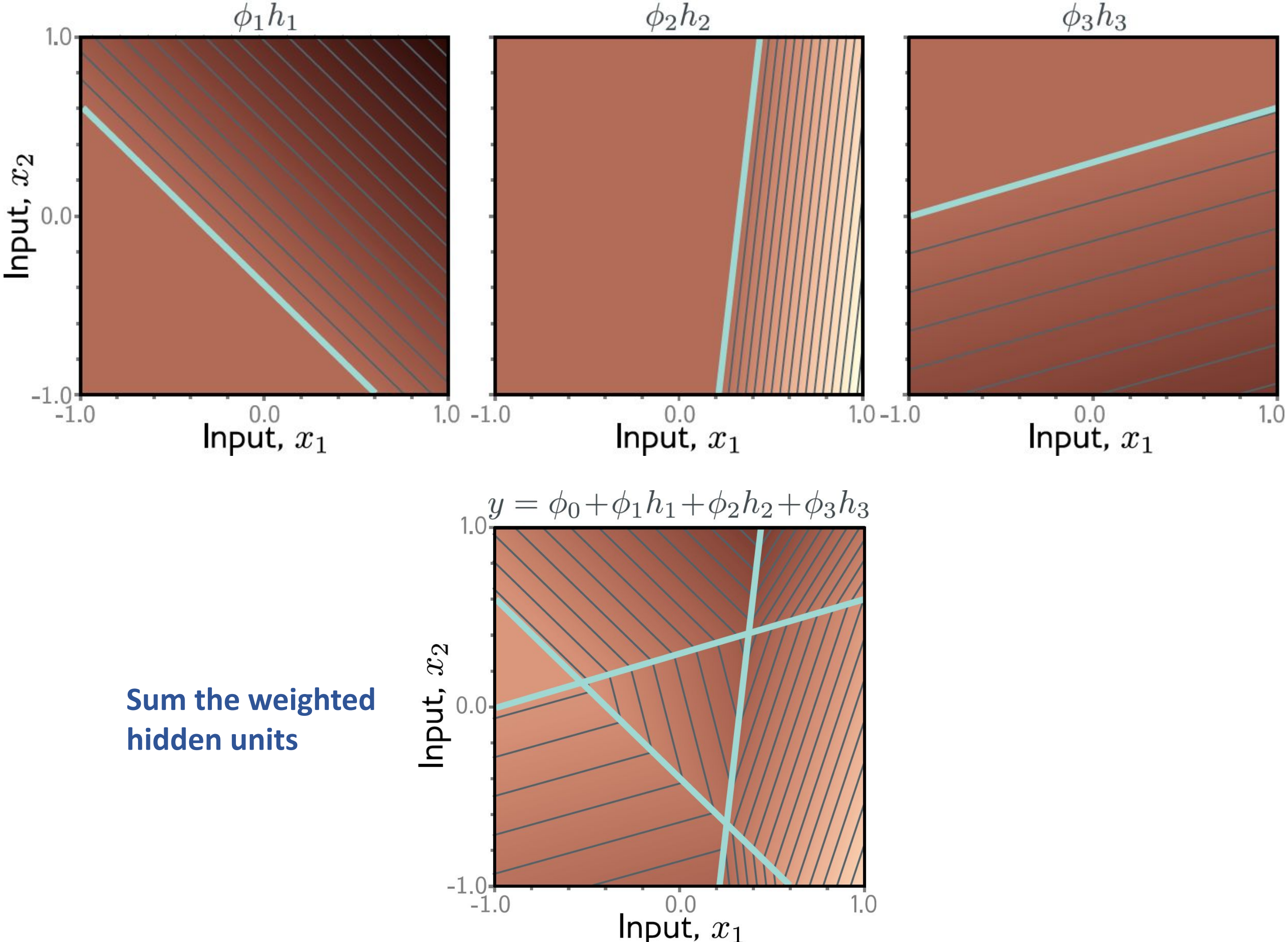
After Activatio n

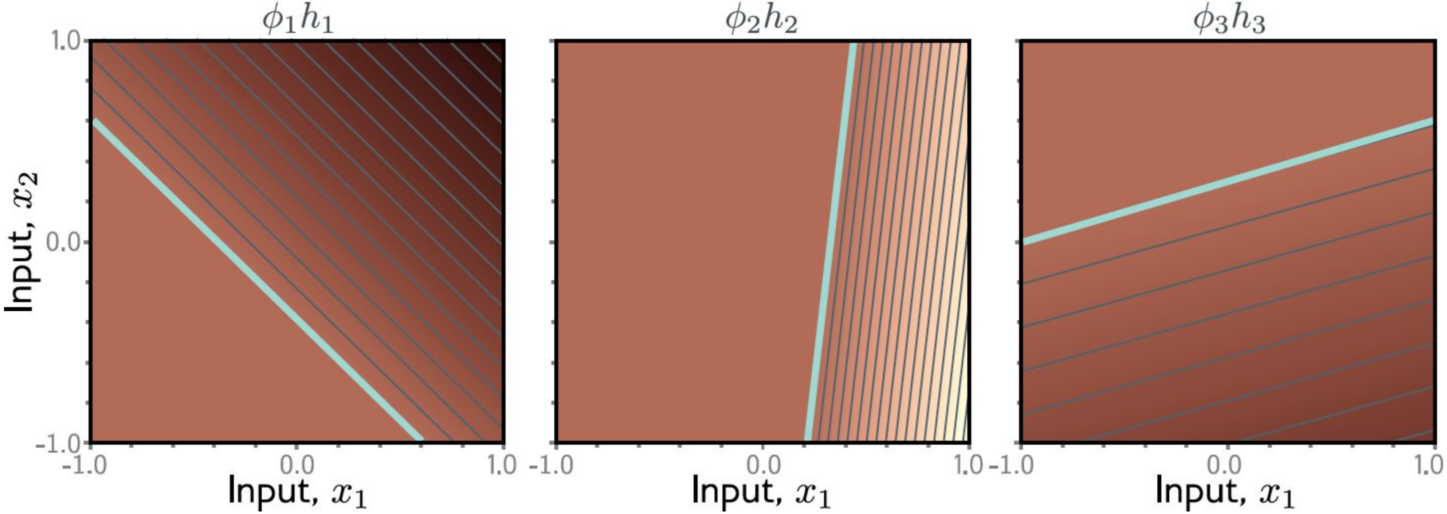


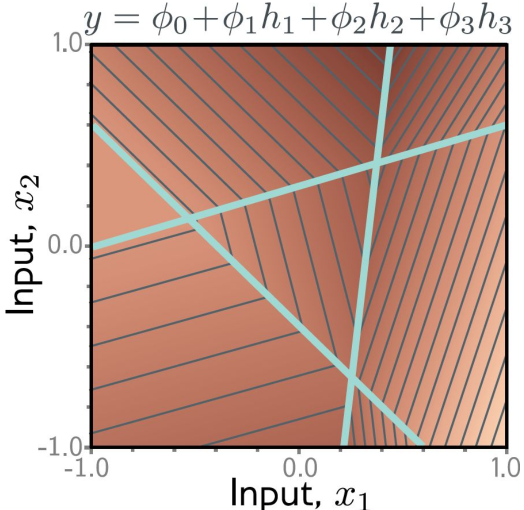
Weight the Hidden units

Weight the hidden units

Sum the weighted hidden units



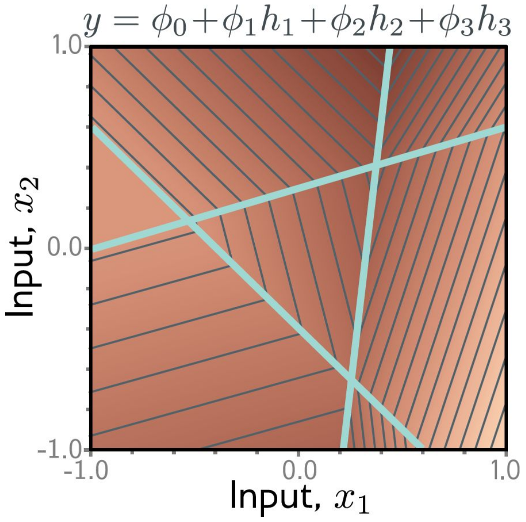


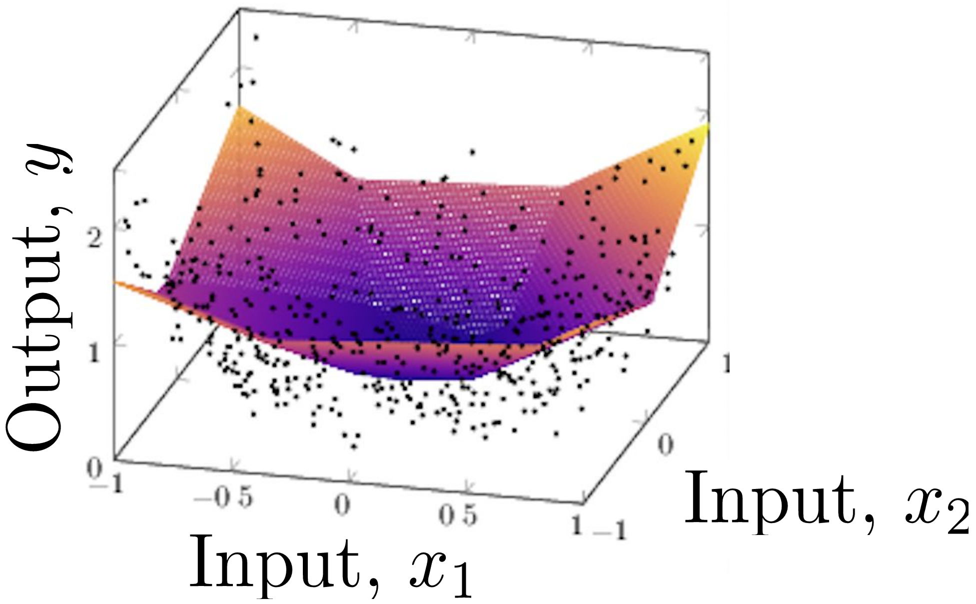


Convex polygonal regions

A region of $\mathbb{R}^D$ is convex if we can draw a straight line between any two points on the boundary of the region without intersecting the boundary in another place.

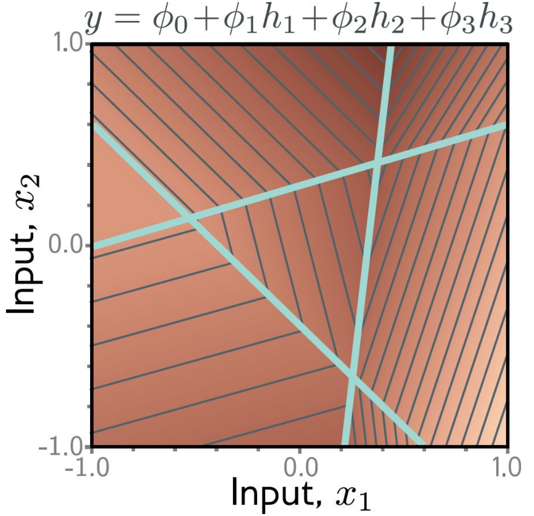
Fitting a dataset where:
each sample has 2 inputs and 1 output





Question:

- For the 2D case, what if there were two outputs?
- If this is one of the outputs, what would the other one look like?



Shallow neural networks

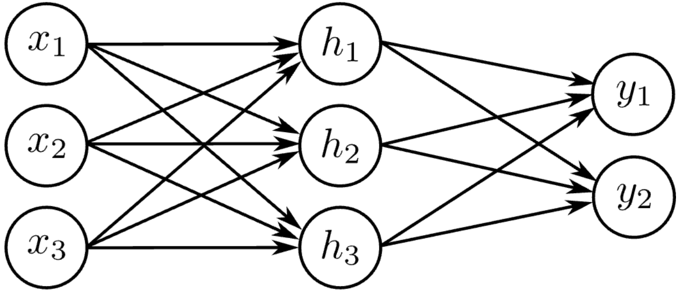
- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology
- Universal approximation HOWTO

Arbitrary inputs, hidden units, outputs

- D_i inputs, D hidden units, and D_o Outputs

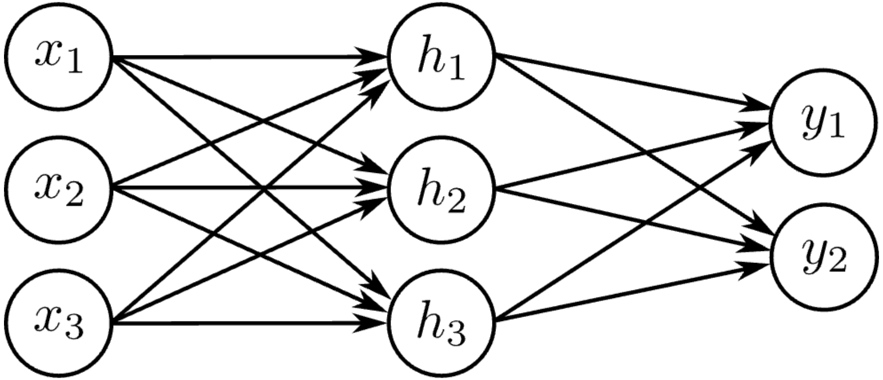
$$h_d = \mathrm{a}\left[\theta_{d0} + \sum_{i=1}^{D_i} \theta_{di} x_i\right] \qquad y_j = \phi_{j0} + \sum_{d=1}^{D} \phi_{jd} h_d$$

- e.g., Three inputs, three hidden units, two outputs

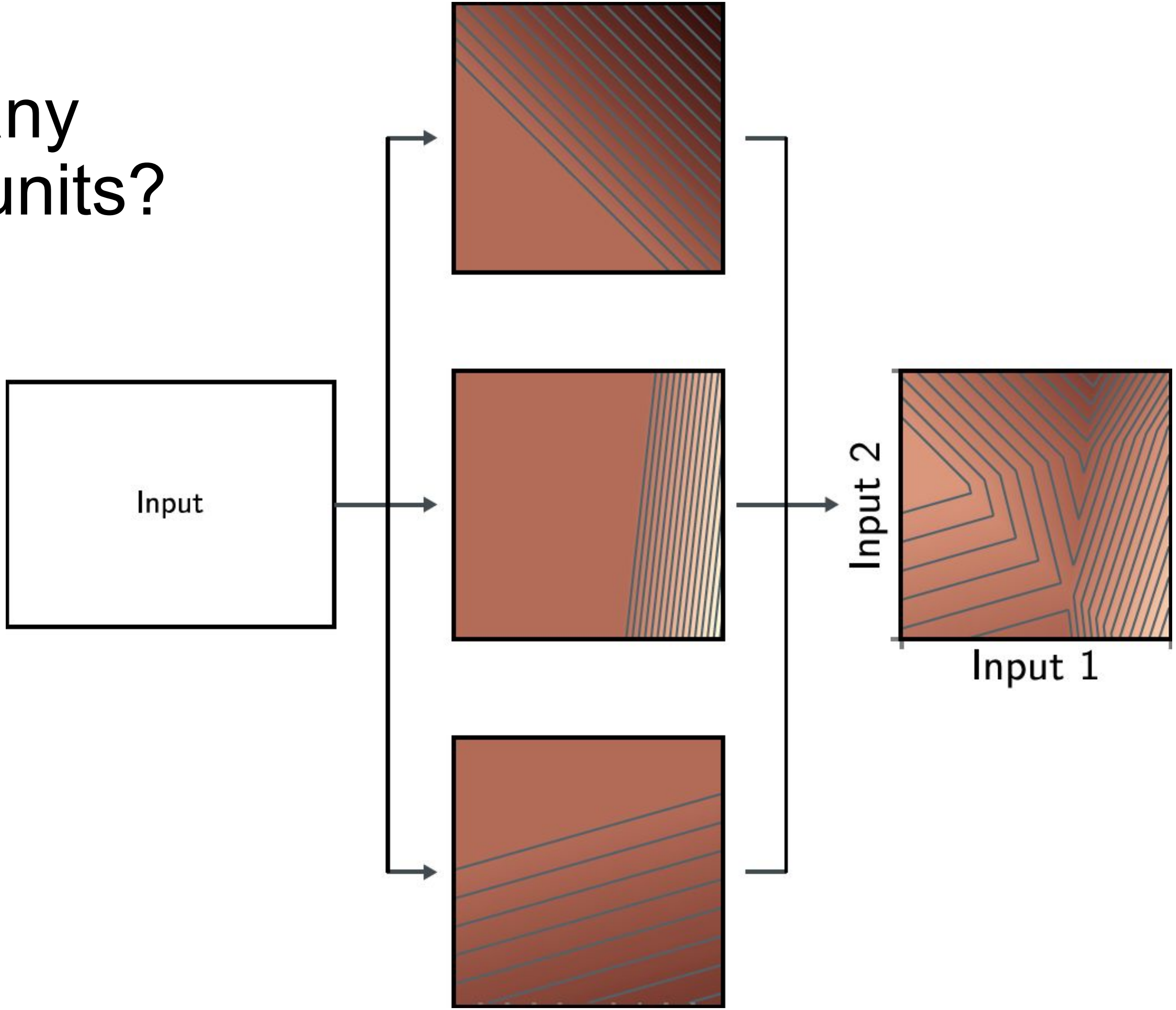


Question:

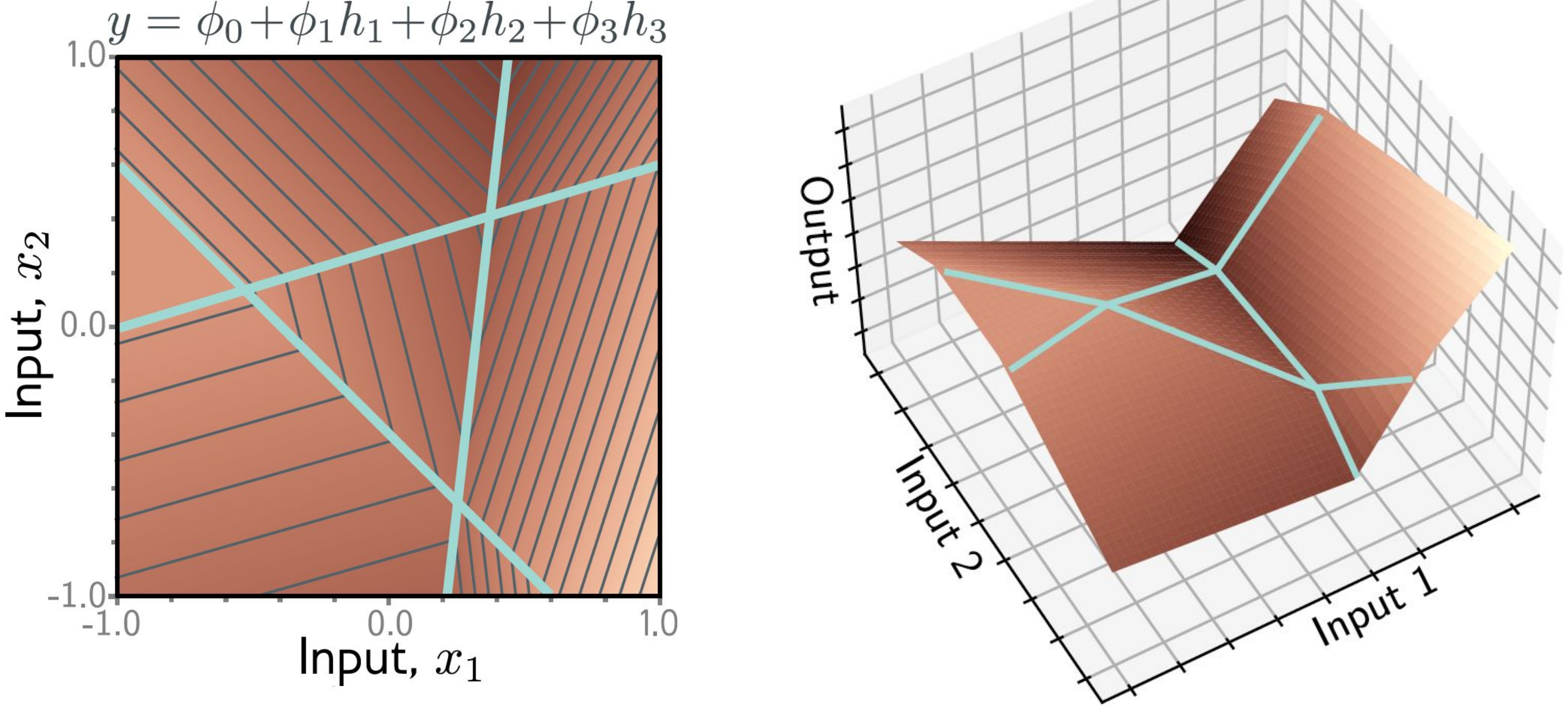
- How many parameters does this model have?



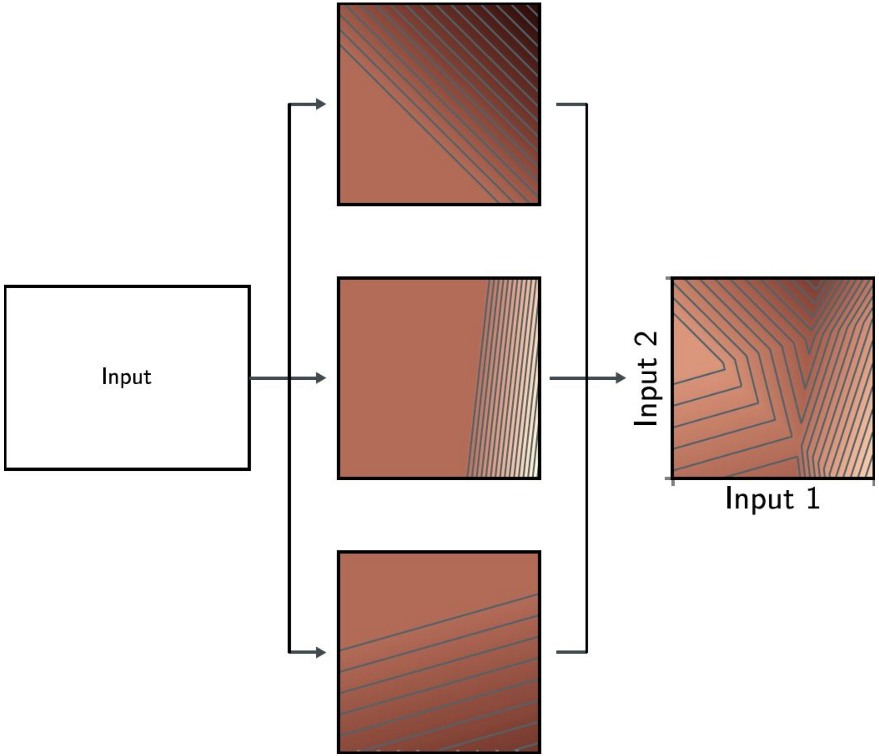
How many hidden units?



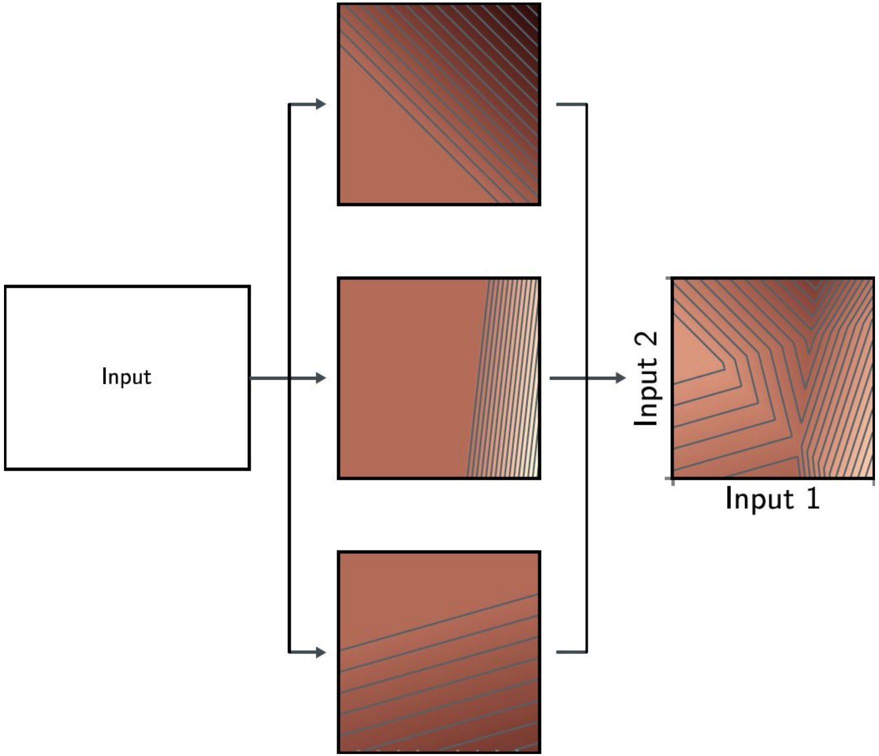
Output with boundaries and in 3D

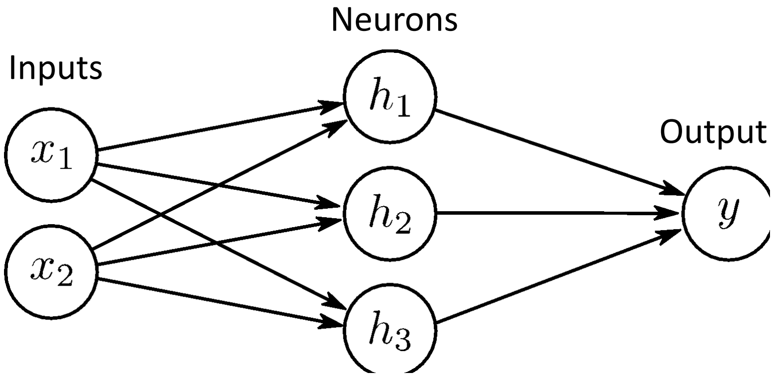


How would you draw and write this neural network?



How would you draw and write this neural network?





“neural network”

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

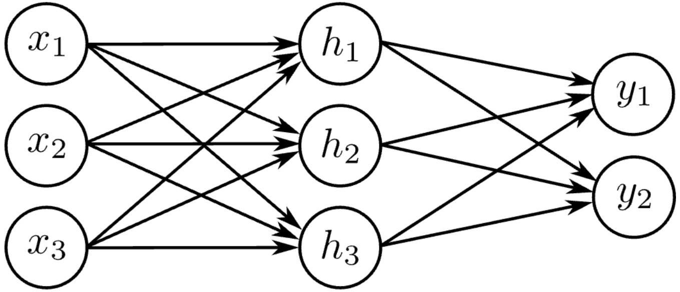
$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11} x_1 + \theta_{12} x_2]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21} x_1 + \theta_{22} x_2]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31} x_1 + \theta_{32} x_2]$$

Arbitrary inputs, hidden units, outputs

- D_o Outputs, *D* hidden units, and D_i inputs

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

- e.g., Three inputs, three hidden units, two outputs

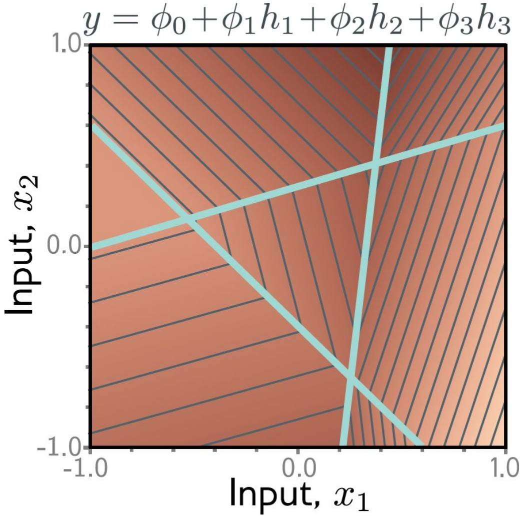


Shallow neural networks

- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology
- Universal approximation HOWTO

Number of output regions

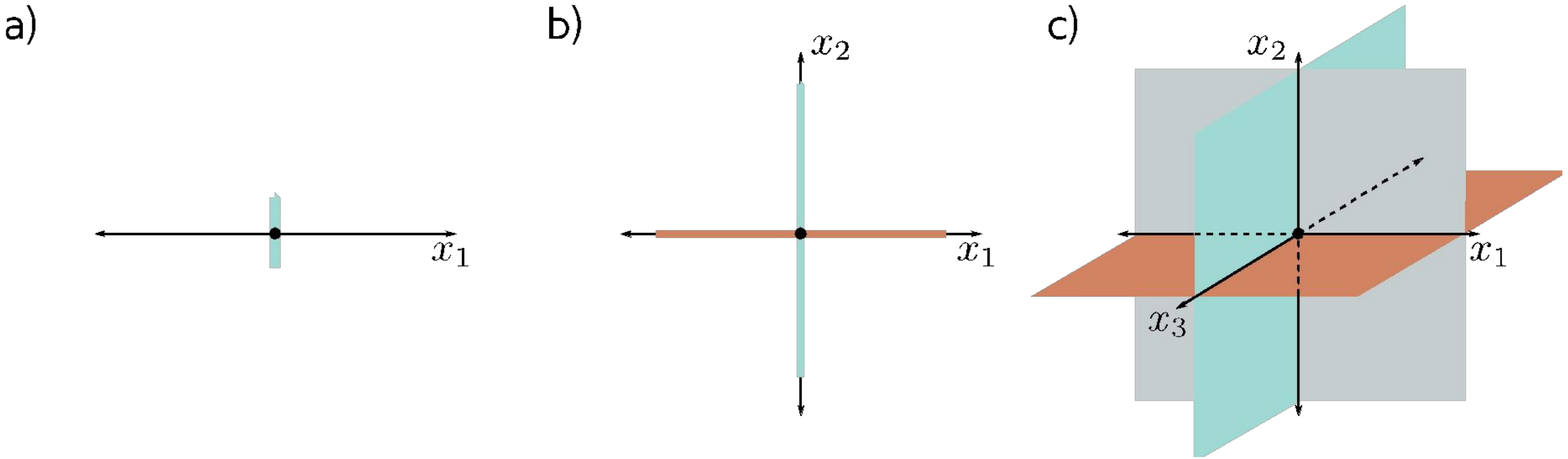
- In general, each output consists of multi-dimensional convex polytopes
- With two inputs, and three hidden units, we saw there were seven polygons for each output:



Polytope -- Wikipedia
In elementary geometry, a polytope is a geometric object with flat sides (faces). Polytopes are the generalization of three-dimensional polyhedra to any number of dimensions. Polytopes may exist in any general number of dimensions n as an n-dimensional polytope or n-polytope.

Example with $D = D_i \rightarrow 2^{D_i}$ regions

D_i : # of inputs
D : # of hidden units
D_o : # of outputs



1 input (1-dimension) with 1 hidden unit creates two regions (one joint)

2 input (2-dimensions) with 2 hidden units creates four regions (two lines)

3 inputs (3-dimensions) with 3 hidden units creates eight regions (three planes)

Number of regions:

D_i : # of inputs
D : # of hidden units
D_o : # of outputs

- Number of regions created by $D > D_i$ hyper-planes in D_i dimensions was proved by Zaslavsky (1975) to be:

$$\sum_{j=0}^{D_i} \binom{D}{j} = \frac{D!}{j!(D-j)!}$$

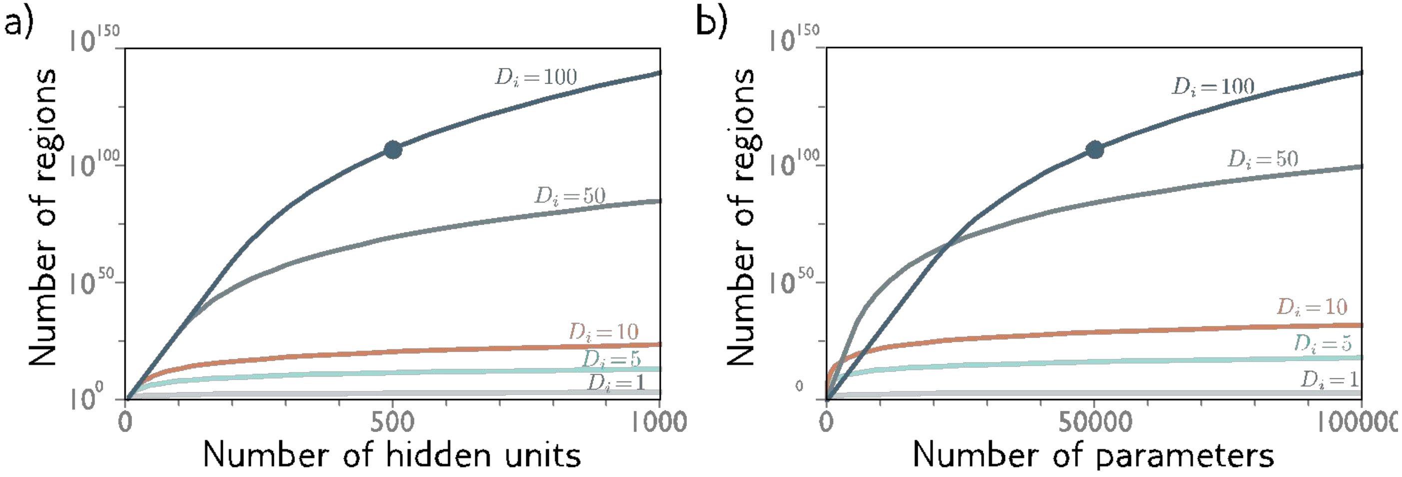
← Binomial coefficients!

- How big is this? It's greater than 2^{D_i} but less than 2^D.

Number of output regions

D_i : # of inputs
D : # of hidden units
D_o : # of outputs

- In general, each output consists of D dimensional convex polytopes
- How many?



Highlighted point = 500 hidden units or 51,001 parameters

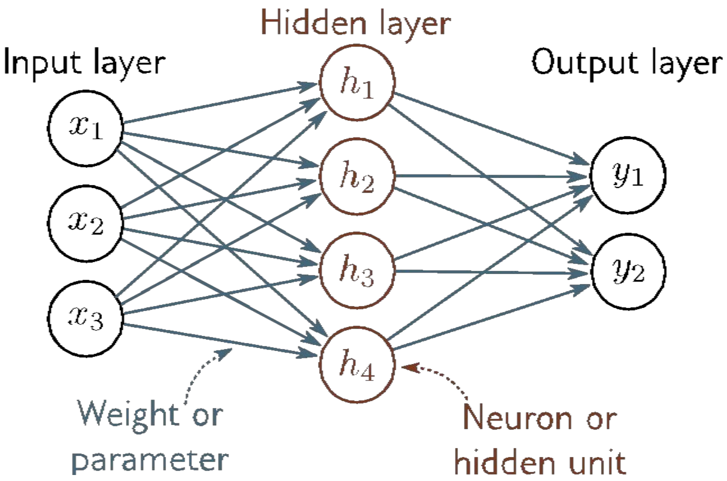
Is More Output Regions Good?

- More output regions ~ more flexibility.
- More output regions ~ more freedom to overfit.
  - But not completely arbitrary freedom to overfit.
  - Still linear within each region.
  - Still consistency between adjacent regions.
- Training details will matter

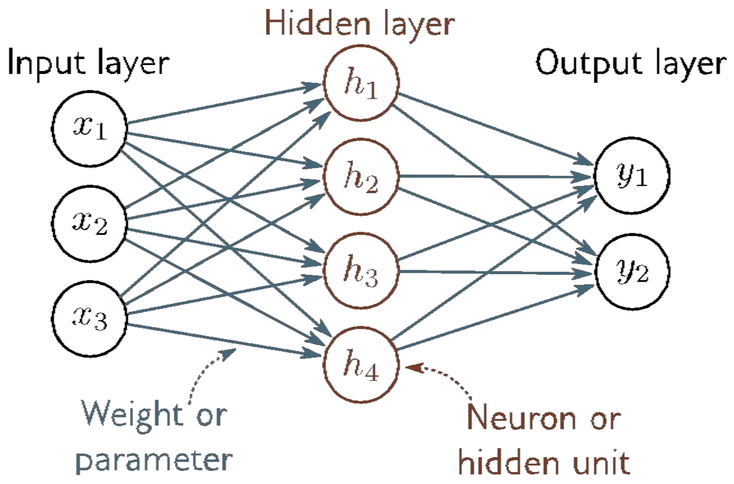
Shallow neural networks

- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology
- Universal approximation HOWTO

Nomenclature

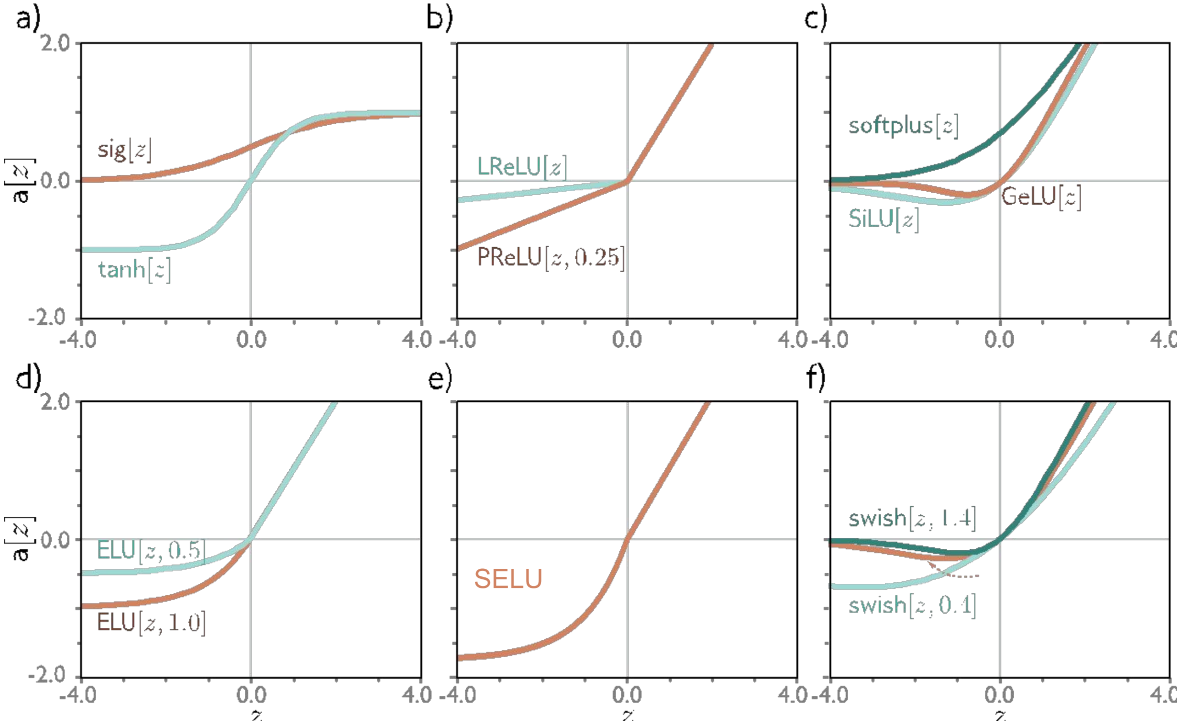


Nomenclature



- Y-offsets = biases
- Slopes = weights
- Everything in one layer connected to everything in the next = fully connected network (multi-layer perceptron)
- No loops = feedforward network
- Values after ReLU (activation functions) = activations
- Values before ReLU = pre-activations
- One hidden layer = shallow neural network
- More than one hidden layer = deep neural network
- Number of hidden units ≈ capacity

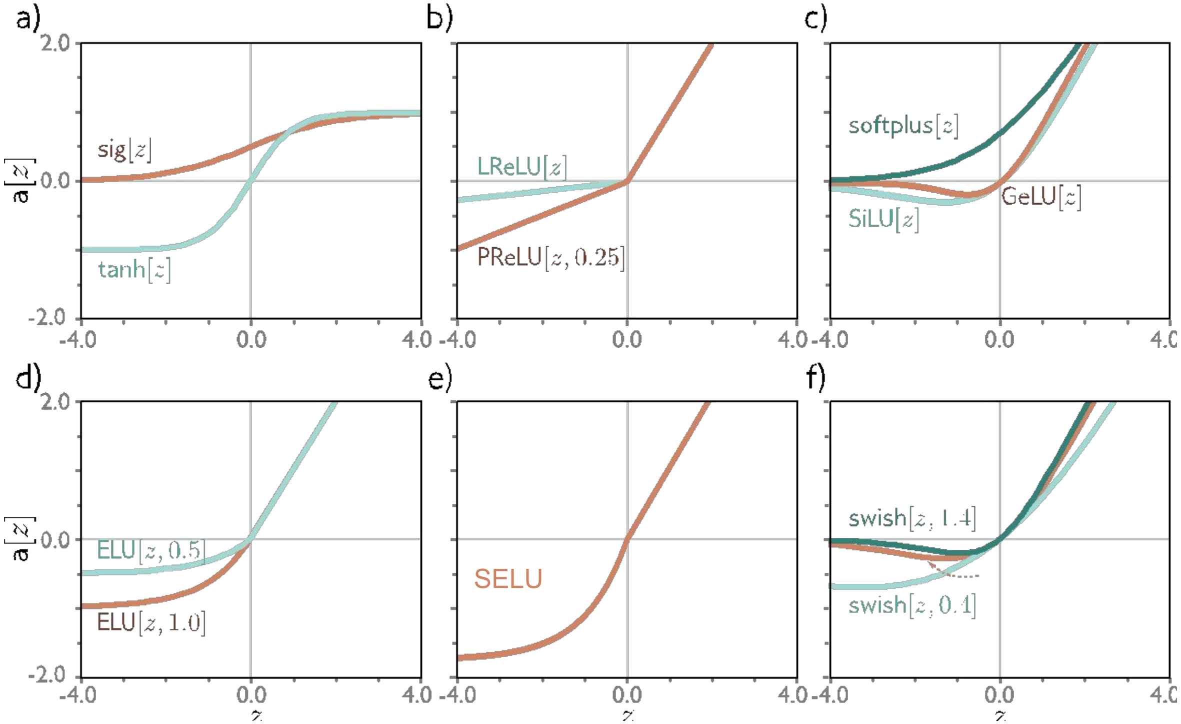
Other activation functions



Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv:1710.05941*.

Other activation functions

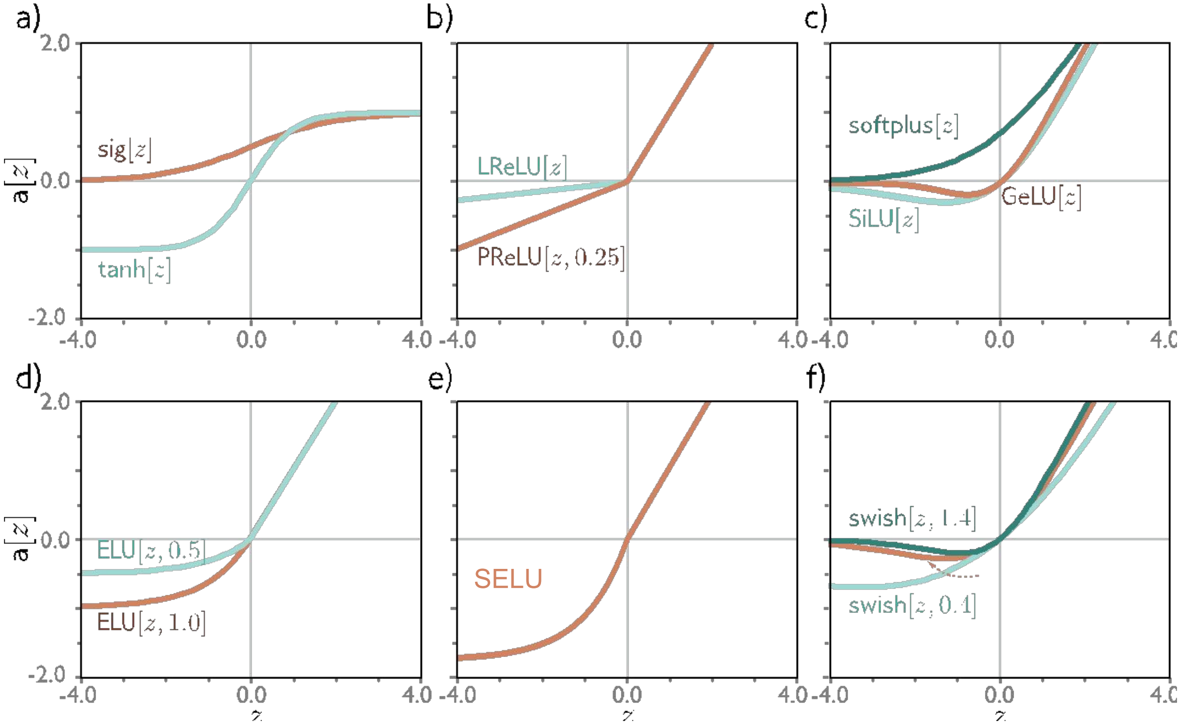
Sigmoid function $\mathrm{sig}[z]$ is handy for limiting output range since its range is [0,1].



Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv:1710.05941*.

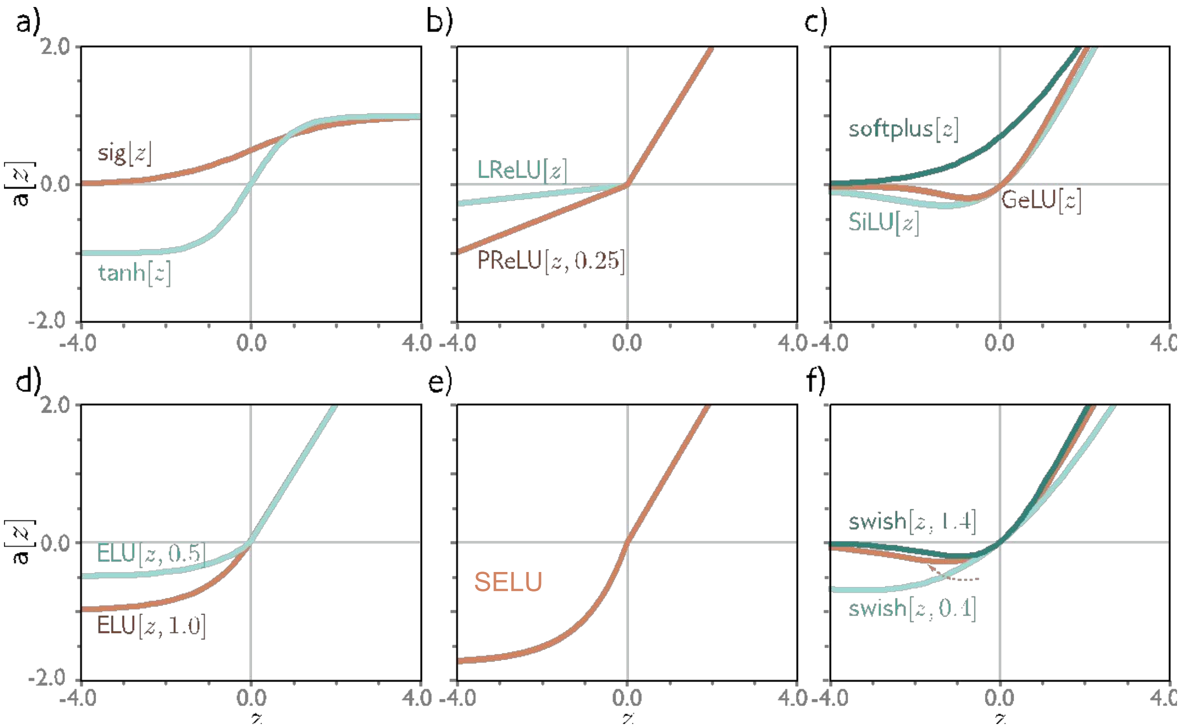
Other activation functions

But both sig[z] and tanh[z] suffer from low gradients ~0 for large inputs. They were the defaults before RELU.



Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv:1710.05941*.

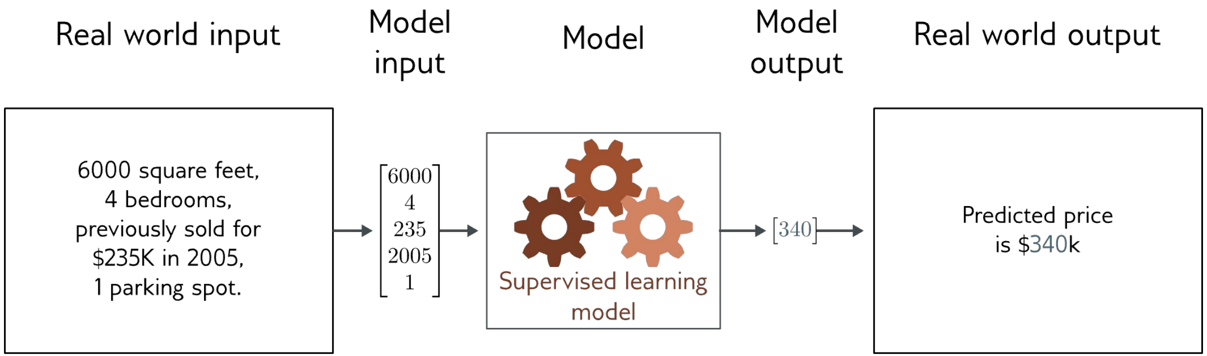
Other activation functions



Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv:1710.05941*.

None of these are polynomials. Polynomial activations restrict output to be polynomials.

Regression



We have built a model that can:

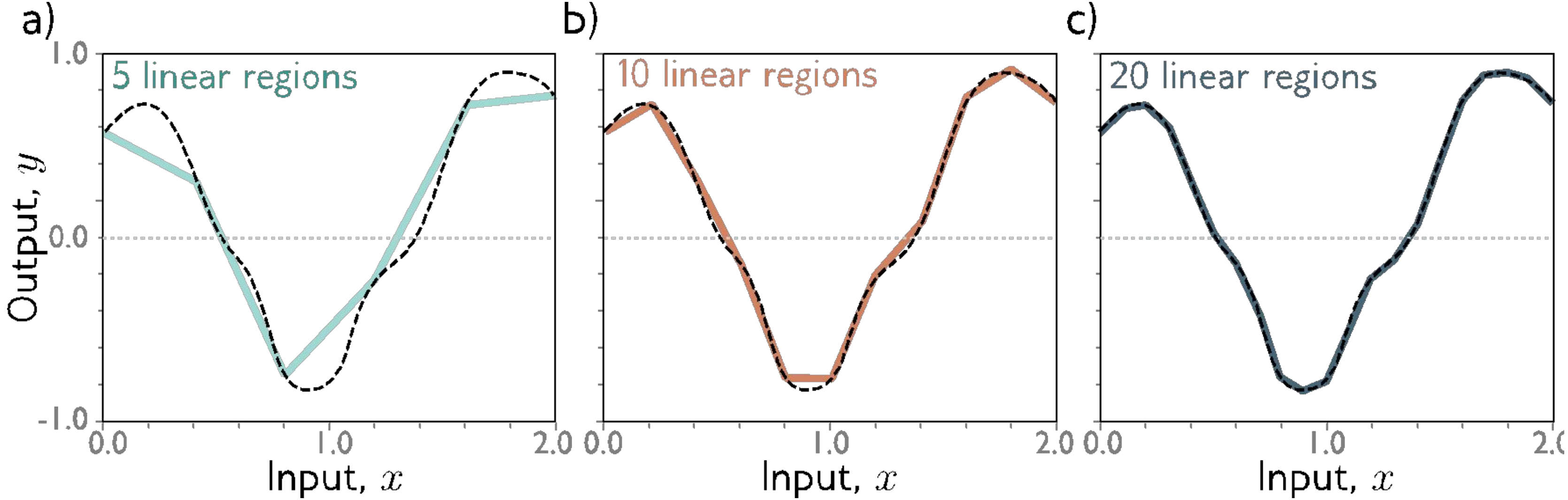
- take an arbitrary number of inputs
- output an arbitrary number of outputs
- model a function of arbitrary complexity between the two

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

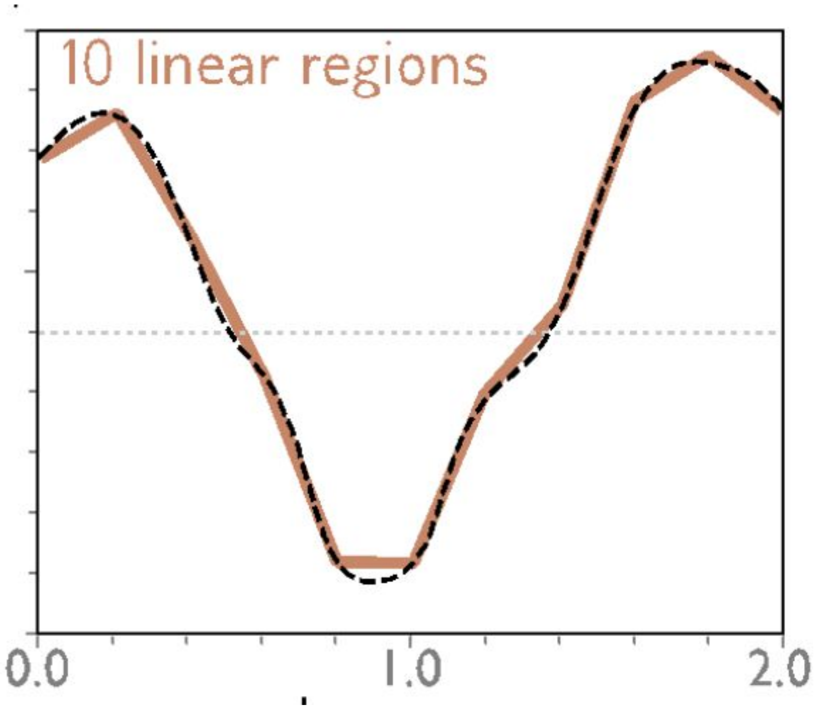
Shallow neural networks

- Example network, 1 input, 1 output
- Universal approximation theorem
- More than one output
- More than one input
- General case
- Number of regions
- Terminology
- Universal approximation HOWTO

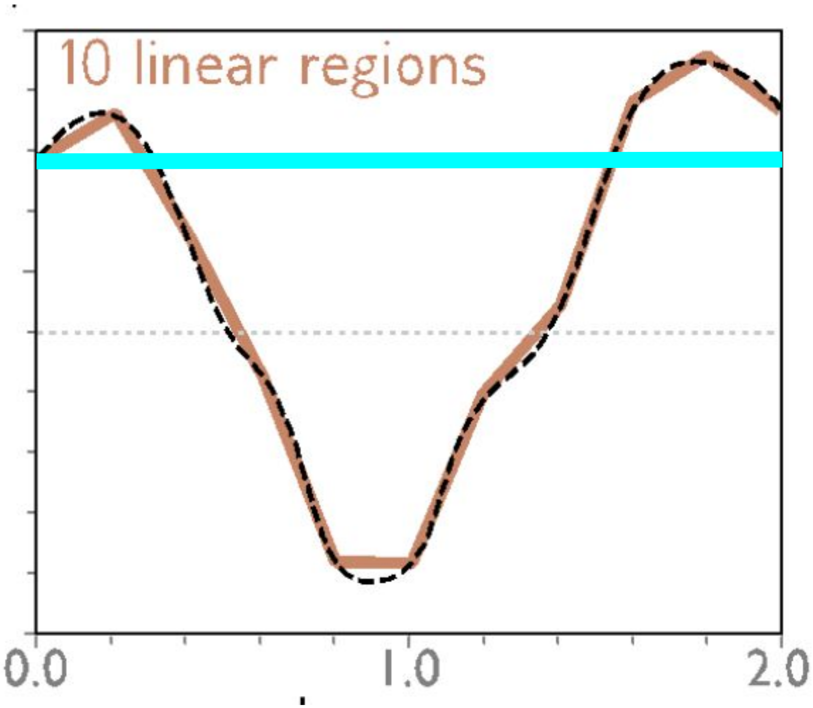
Universal Approximation HOWTO



Universal Approximation HOWTO

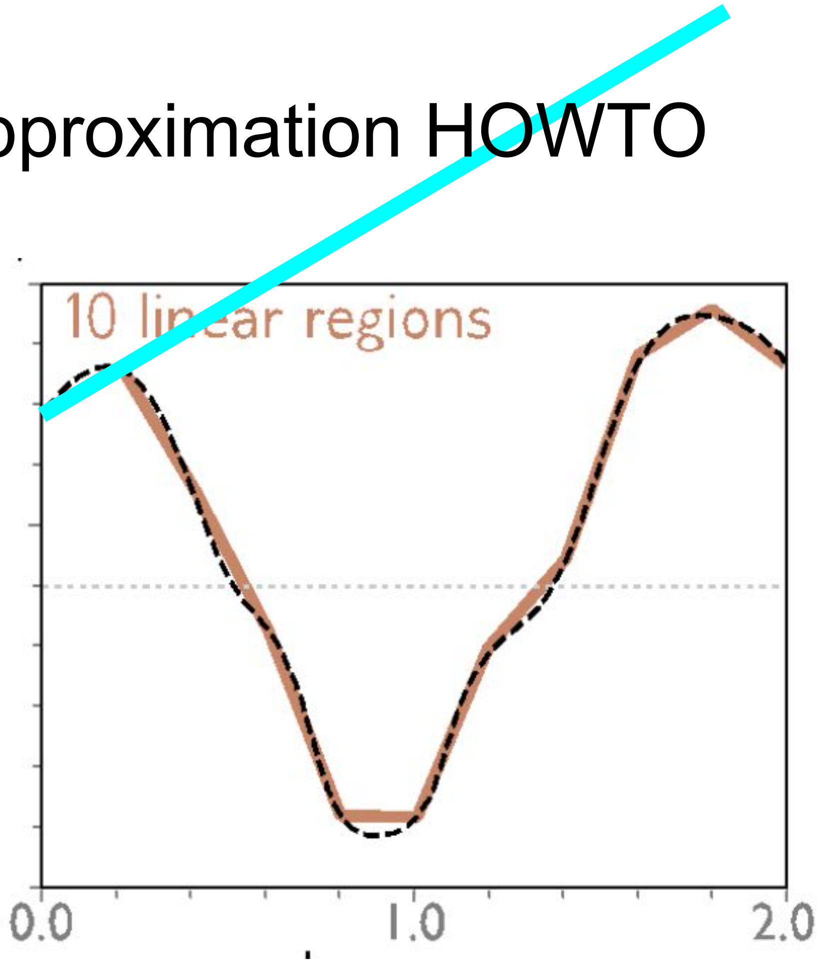


Universal Approximation HOWTO



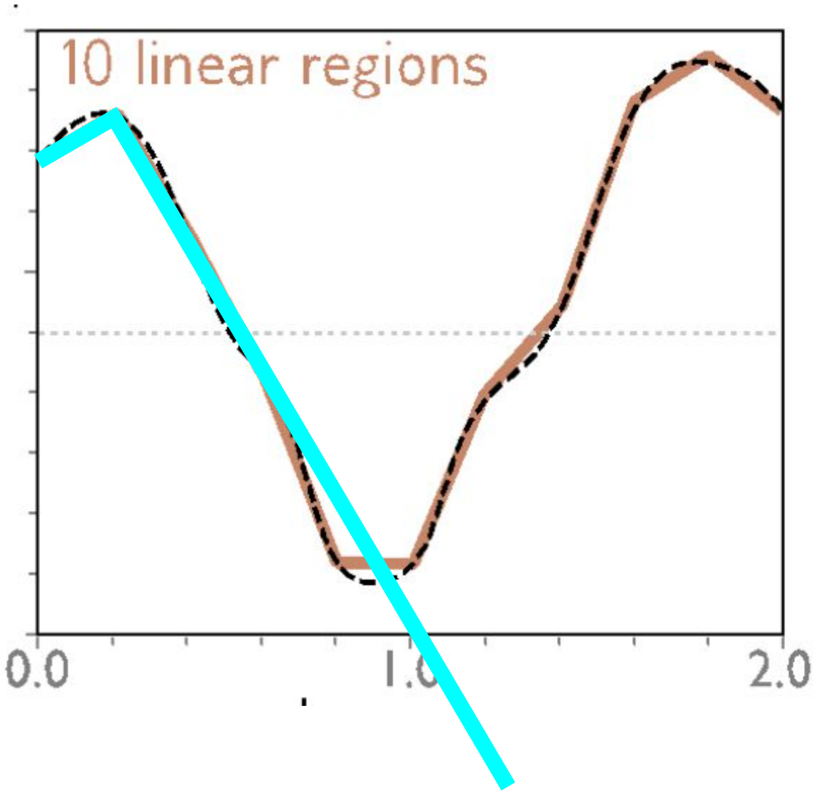
Just bias.

Universal Approximation HOWTO



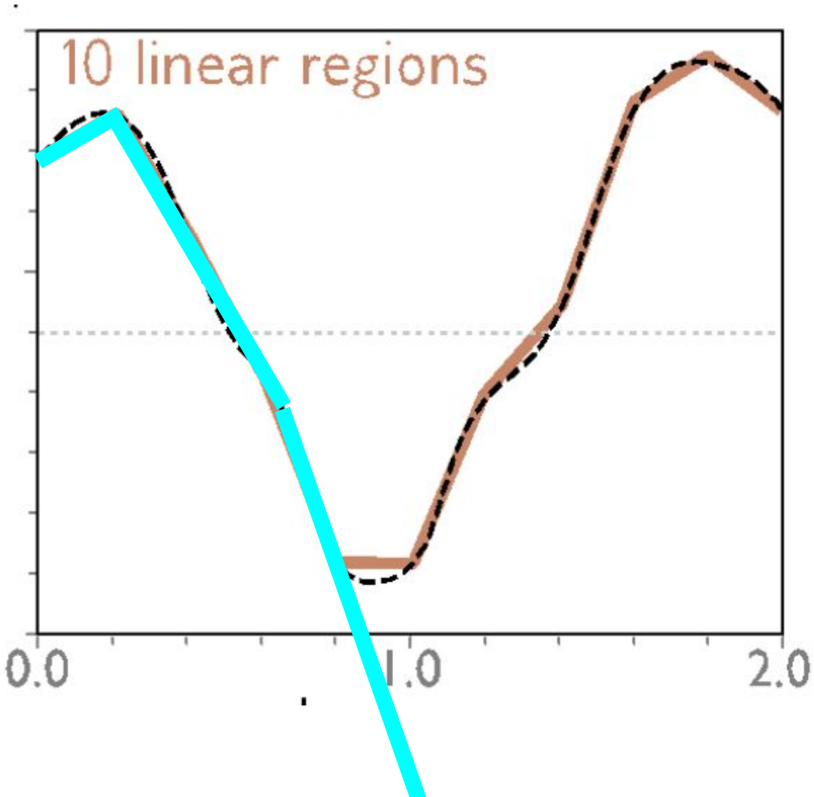
Add a hidden node that activates at x=0, setting slope to match first region.

Universal Approximation HOWTO



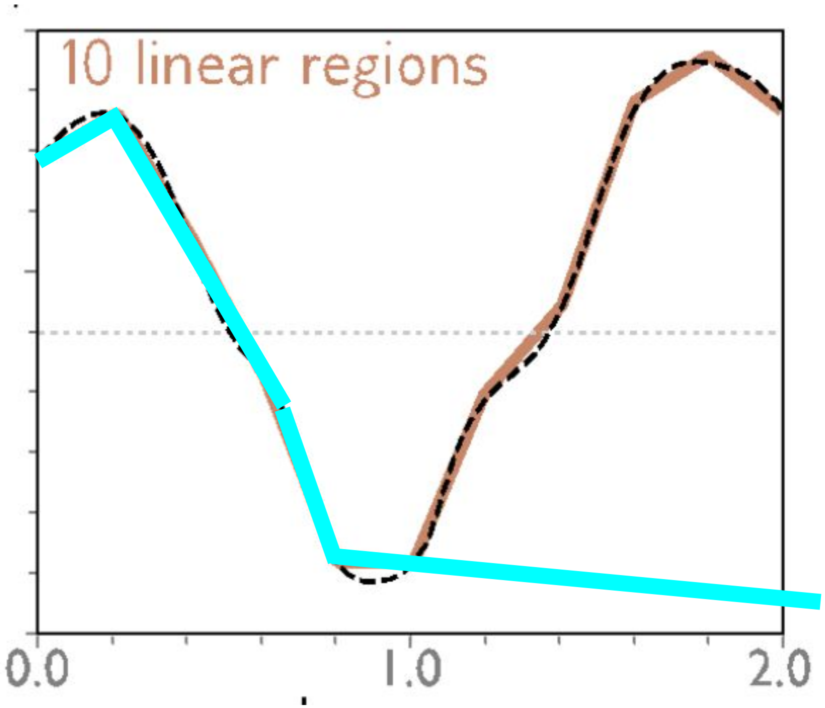
Each new hidden node activates at the next region boundary.

Universal Approximation HOWTO



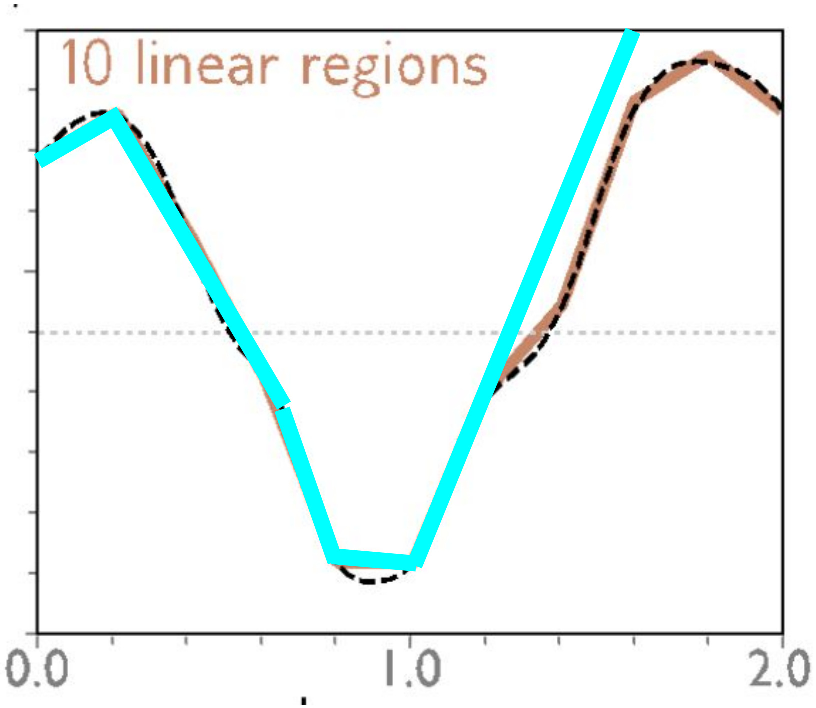
This process works by extending the number of regions perfectly matched one at a time.

Universal Approximation HOWTO



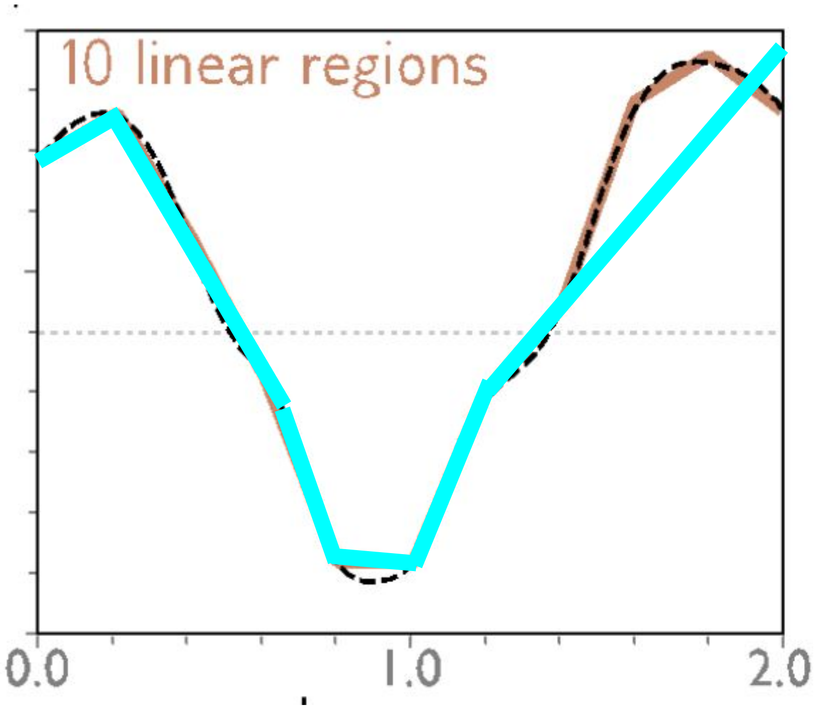
Regions not matched yet are completely ignored.

Universal Approximation HOWTO



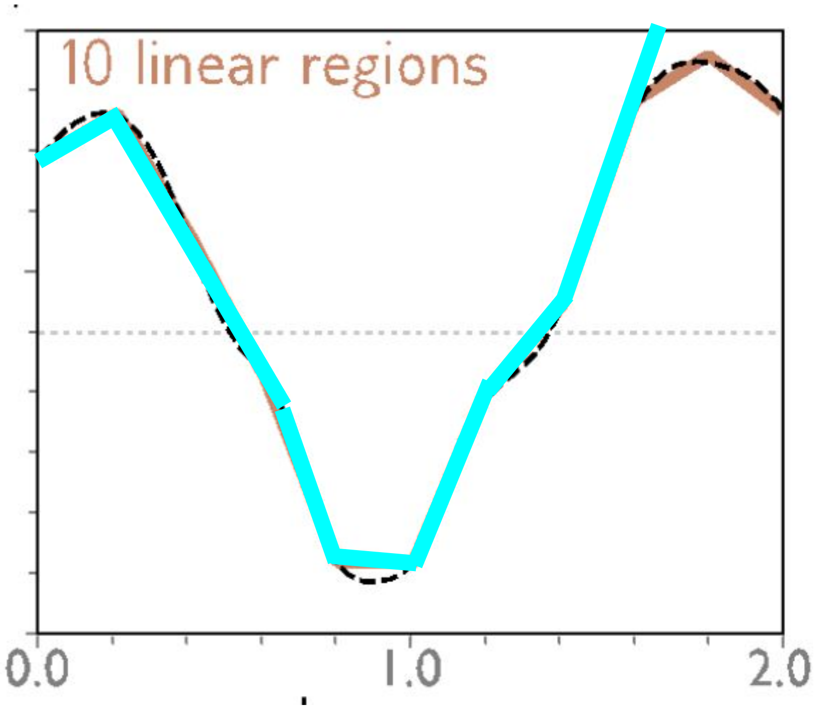
With 1 input variable, it is easy to order the regions, and avoid impacting the previously matched regions.

Universal Approximation HOWTO



With 1 input variable, it is easy to order the regions, and avoid impacting the previously matched regions.

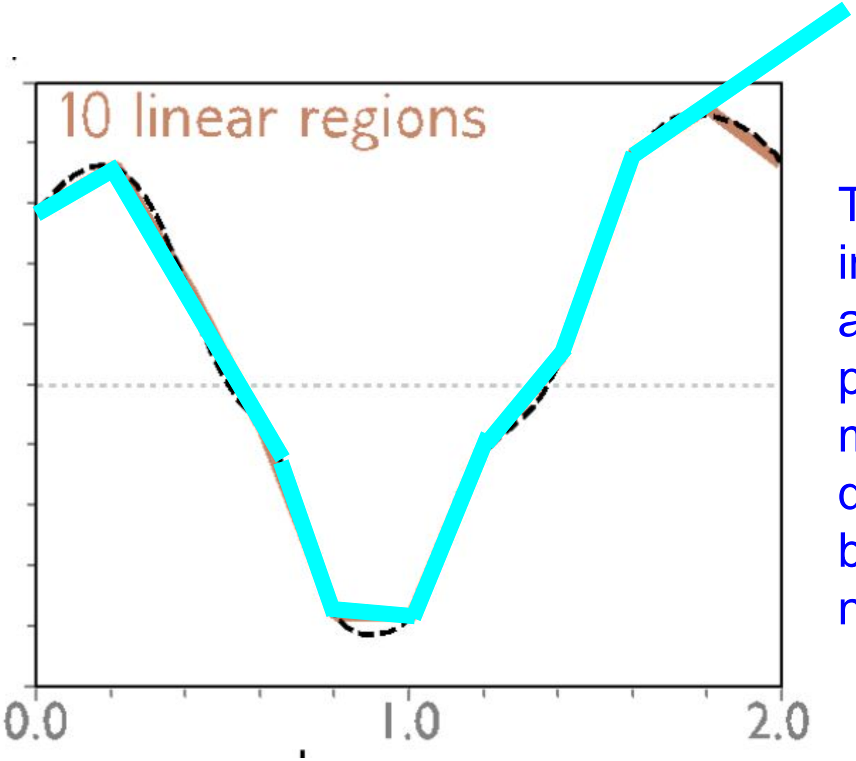
Universal Approximation HOWTO



$+ \phi_i \, a[x - b_i]$

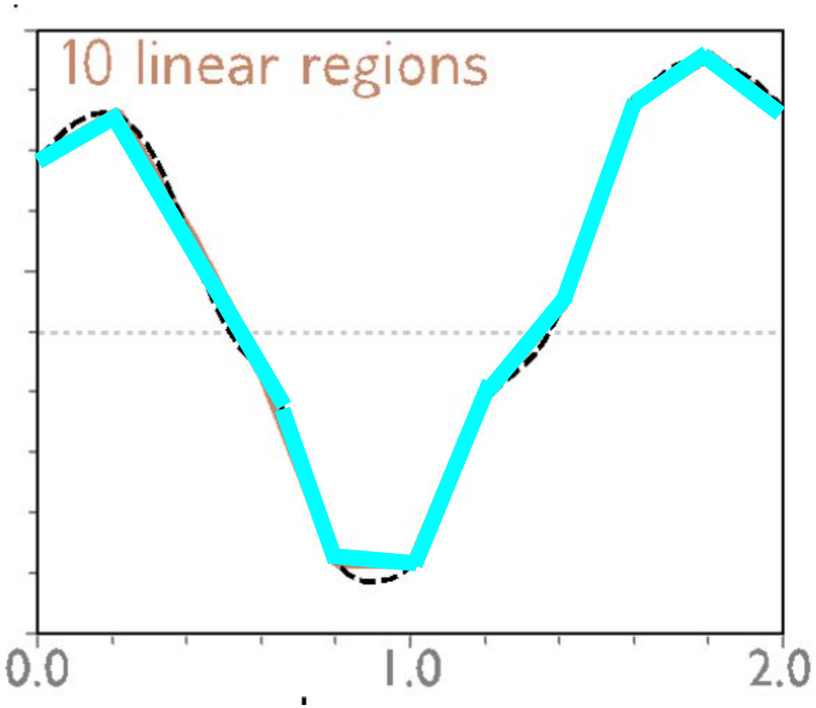
where b_i is the *i*th region boundary

Universal Approximation HOWTO



This process of incrementally matching a region while avoiding previous matches gets much harder with more dimensions. But it will be easier with deep networks.

Universal Approximation HOWTO



BTW we do not construct universal approximations like this in practice. We are training from points, not fitting curves. And training data may conflict.

Next Week

- Deep Neural Networks
  - More of the same?
  - But different?
  - And better?

Feedback?