Artificial Neural Networks

Feedforward Networks Part I

Outline

- Limitations of linear methods
- Biological Inspiration
- Artificial Neural Networks
  - Diagrammatic representation
  - Notation
  - Feedfoward
- ANN Representation
  - What are ANNs representing? What do these transformations accomplish?

Limitations of Linear Methods

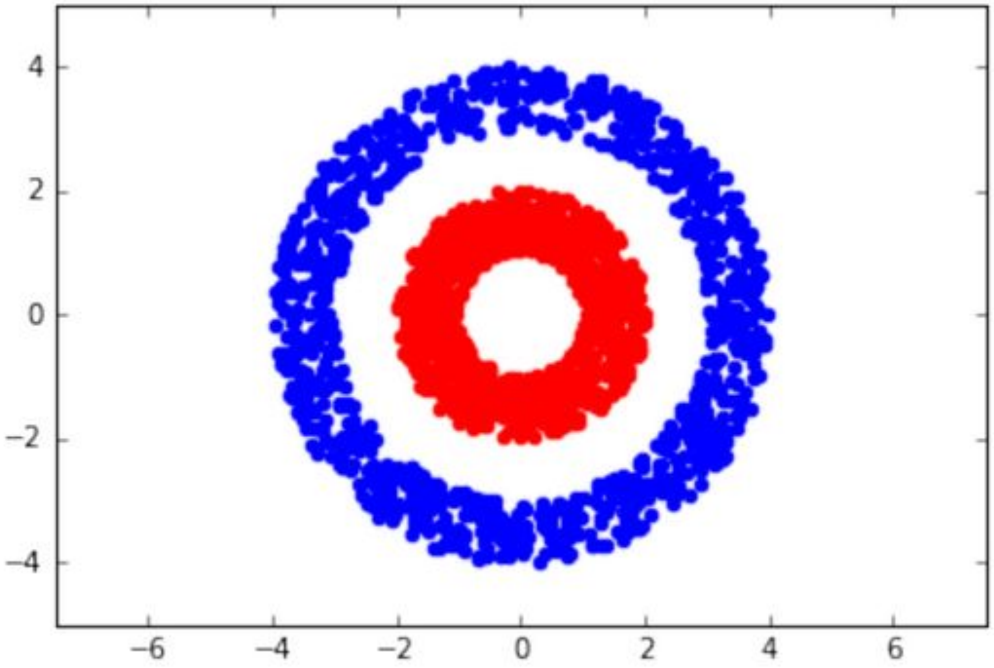
Limitations of Linear Methods

- Recall that Logistic Regression will find linear decision boundaries (hyperplanes)
- Thus, we don’t have to look far to find example problems where Linear and Logistic Regression will completely fail

Donut Dataset

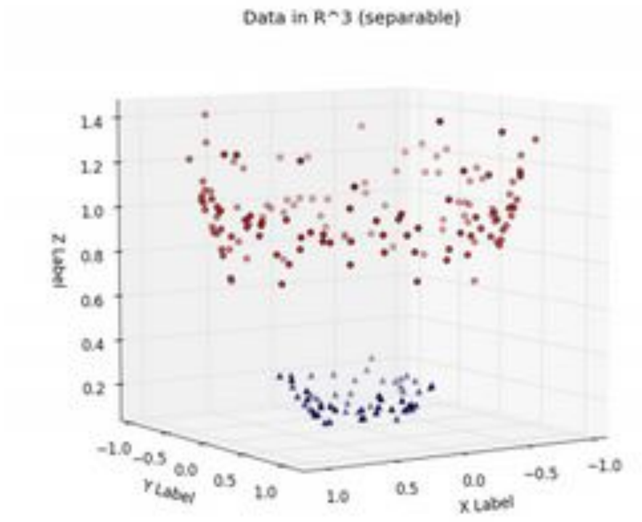
- A donut (or bullseye) dataset is where one class entirely encloses the other class
- There is no conceivable linear boundary that can help up solve this problem

- But with clever feature engineering, we can make progress with linear classifiers. Any guesses?



Donut Dataset

- If we project this dataset into a higher dimensional space, and transform the original features, then we can arrive at a linearly separable problem in the higher dimensional space.

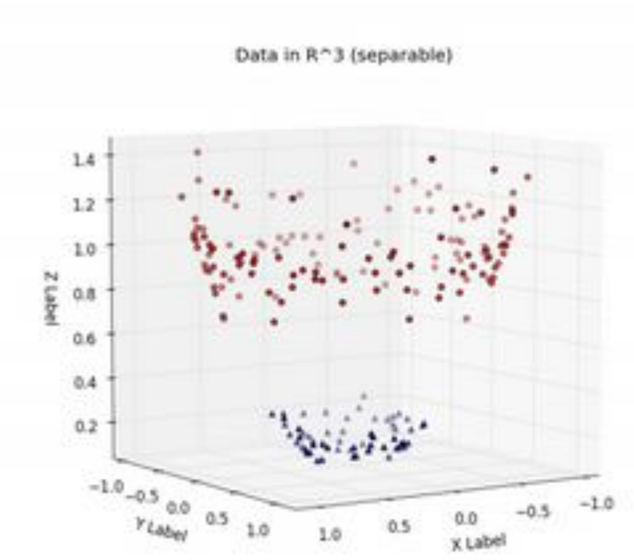


$$\phi : (x_1, x_2) \to (x_1^2, \sqrt{2x_1 x_2}, x_2^2)$$

Donut Dataset

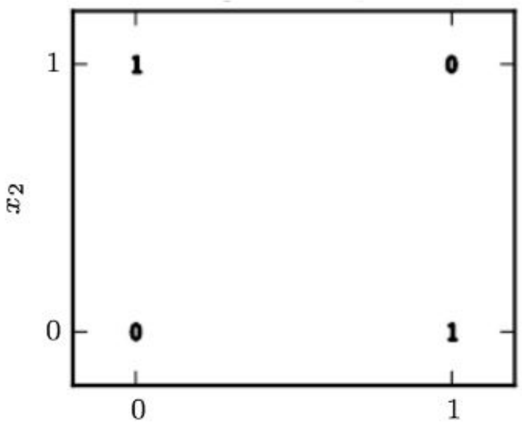
- The downside here is that we had to think of this coordinate transformation
- This kind of feature engineering is time consuming and not generally applicable.

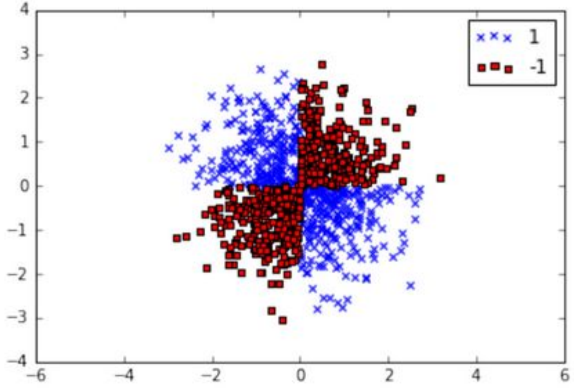
- Our goal will be to use models that are flexible enough to learn these kinds of feature transformations for us.



XOR Dataset

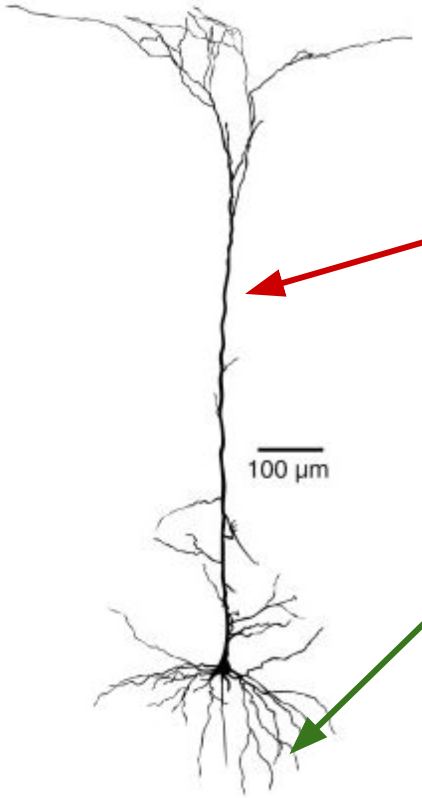
- The boolean XOR function, on 2D binary inputs.
  - {1,1} -> 0
  - {1, 0} -> 1
  - {0,1} -> 1
  - {0,0} -> 0
- Again, no possible way for a linear classifier to succeed here.





Biological Inspiration

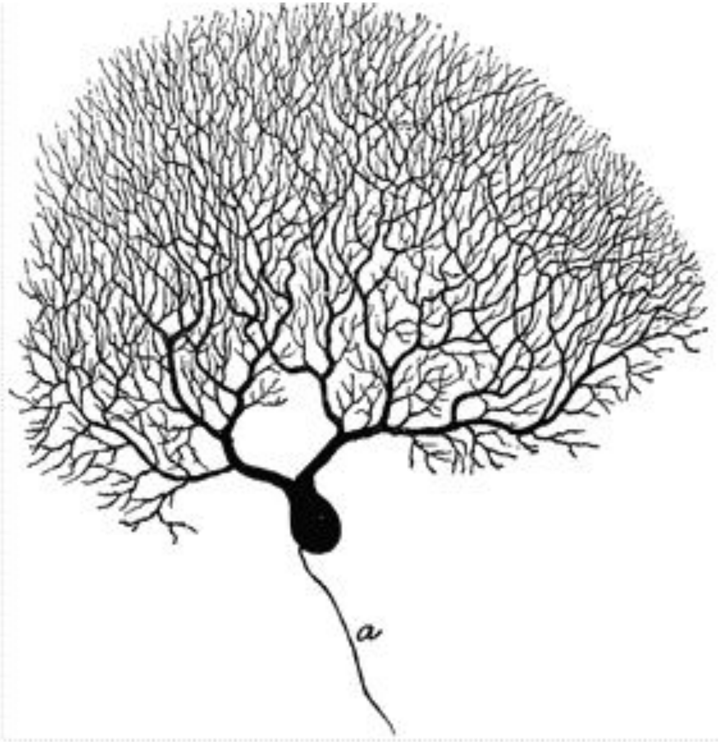
ANNs - Biological Inspiration



In this tracing of a **cortical pyramidal cell** we see:

- Cell body
- **Axon**
  - Output communication channel of the cell. When the cell “fires”, signal is sent down the axon to whichever cells this one communicates with
- **Dendrites**
  - Input communication channel of the cell. Other cells will synapse onto these areas in order to pass along signals.

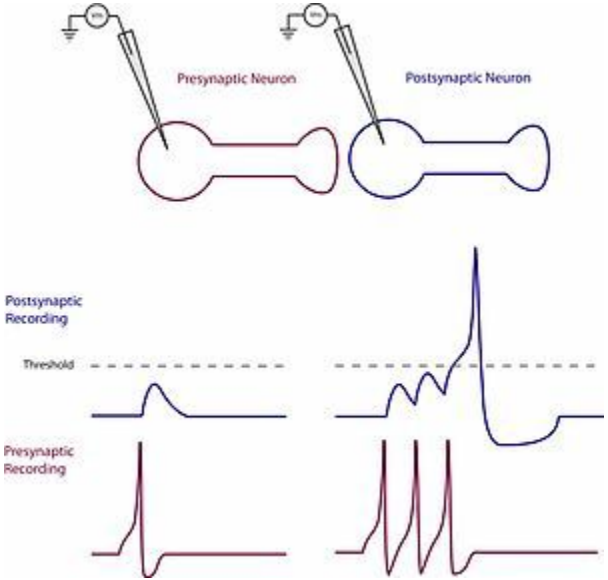
ANNs - Biological Inspiration



Biological neurons have a huge amount of variation in their shapes, but those basic input-output relationships hold.

This is a **cerebellar purkinje cell**. The most beautiful cell in the whole brain. These dendrites are incredible.

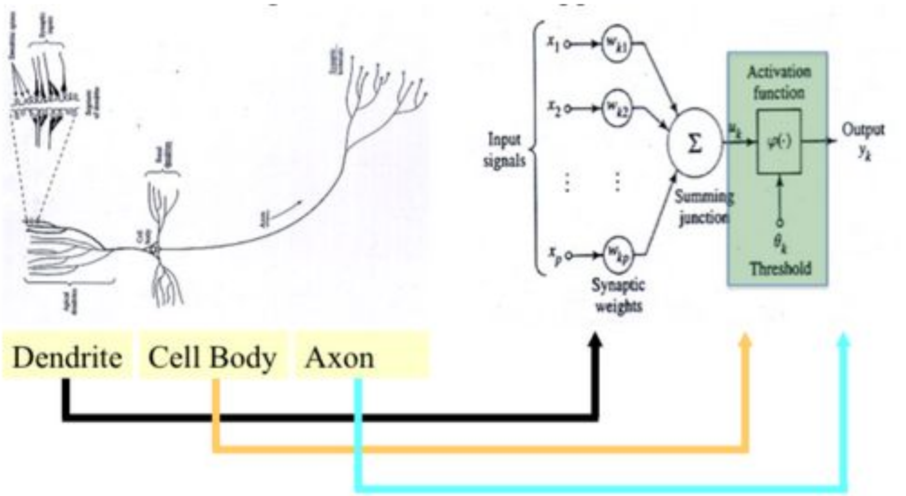
ANNs - Biological Inspiration



Action potential

- A cell receives lots of inputs into its dendrites
- These cause small fluctuations in membrane voltage
- When enough inputs occur at nearly the same time, and **threshold** is reached and the cell fires a “spike” or action potential
- This is how the cell sends a signal along to other downstream cells

ANNs - Biological Inspiration



- **Artificial Neural Networks**
    - ANNs are flexible mathematical functions
    - Composed of “hidden units” which are inspired by biological neurons
        - They have inputs
        - They compute a weighted sum of those inputs
        - They output any non-linear transformation of that sum
        - Their inputs and/or outputs can be other such hidden units
        - Stacking them in this way allows them to represent a rich set of functions

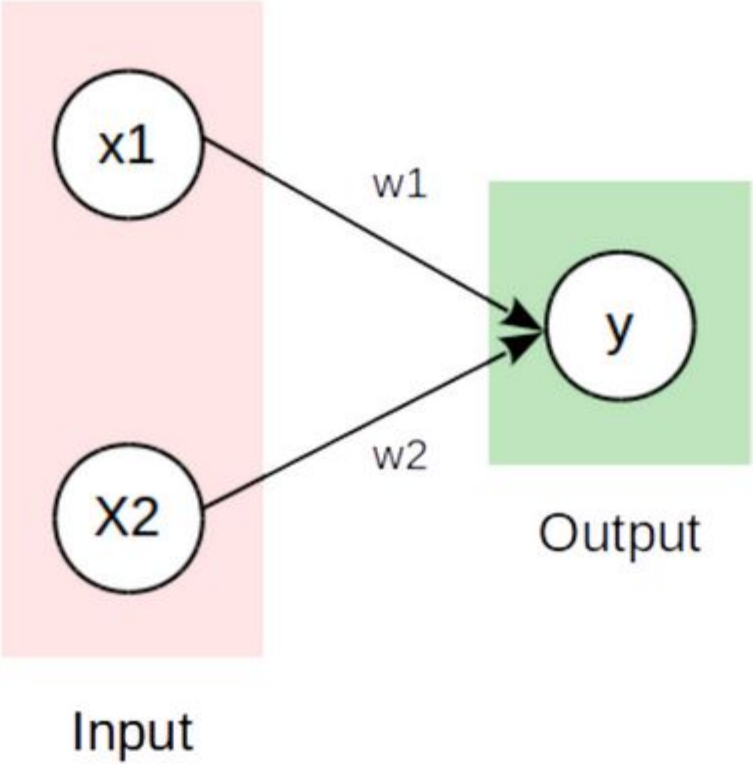
ANNs - Biological Inspiration - Caution

- This is approximately where the similarities between the biology and the machine learning end
- There is a lot of complexity in the biology that the ML community doesn’t try to get near
  - Dendrites are complex
  - Cells type variability
  - Connectivity in the brain much is more complex than feedforward or RNNs
  - Actual learning the brain (synaptic plasticity) is wildly different than backpropagation, and **much** more difficult
  - Convolutional neural networks **might** be performing similar computations as the primate visual system, but far too early to say
  - Modern reinforcement learning methods **might** be computationally similar to the dopamine system, too early to say

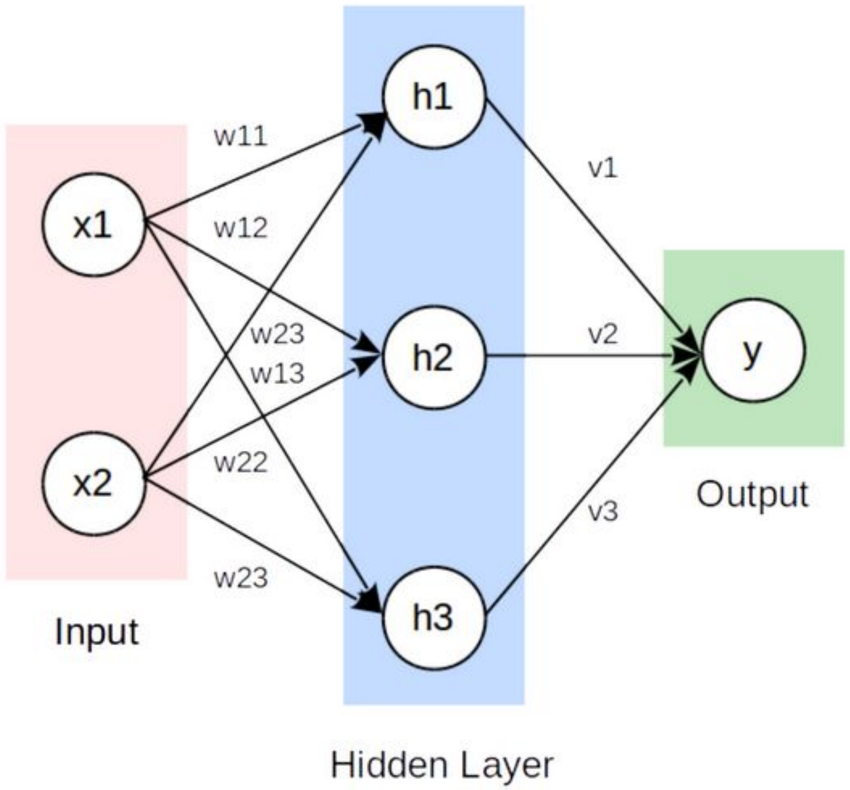
- And that is all fine. ANNs are a useful mathematical toy that can achieve impressive things. There is no denying that, just be careful with the brain talk.

ANN Diagrammatic Representation and Notation

Linear & Logistic Regression

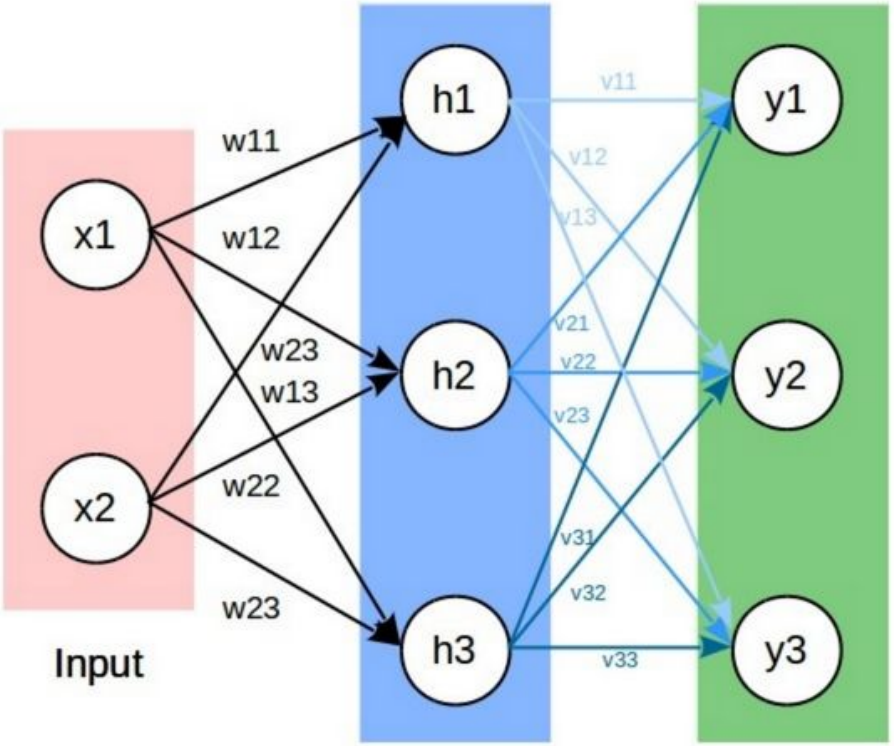


$$\begin{cases} a = \mathbf{w} \cdot \mathbf{x} + b & \text{Linear Activation Function} \\ y = \sigma(a) = \frac{1}{1+exp(-a)} & \text{Sigmoid transform of } a \end{cases}$$



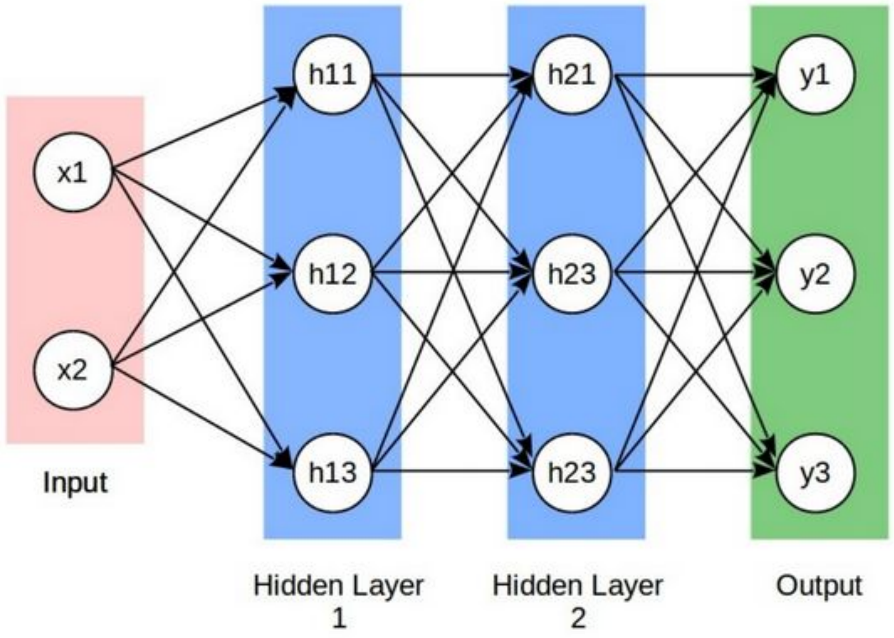
This ANN has

- 2-D inputs
- 1-D outputs
- A single hidden layer with 3 hidden nodes



This ANN has

- 2-D inputs
- 3-D outputs
- A single hidden layer with 3 units

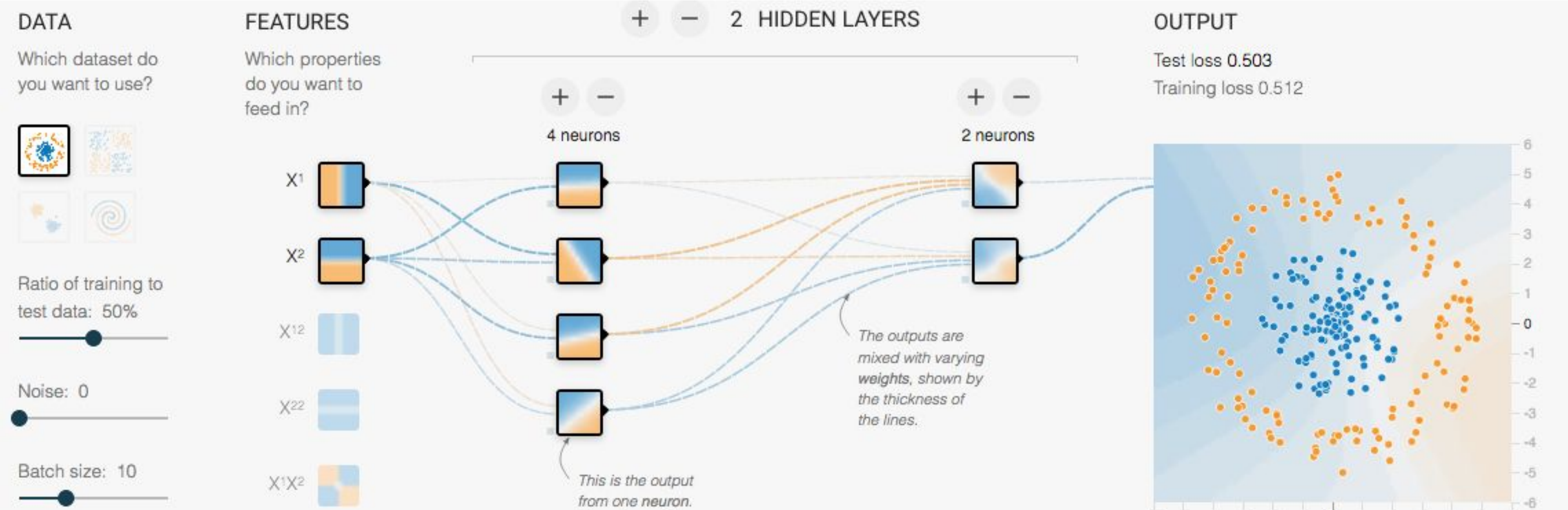


This 4-layer network has

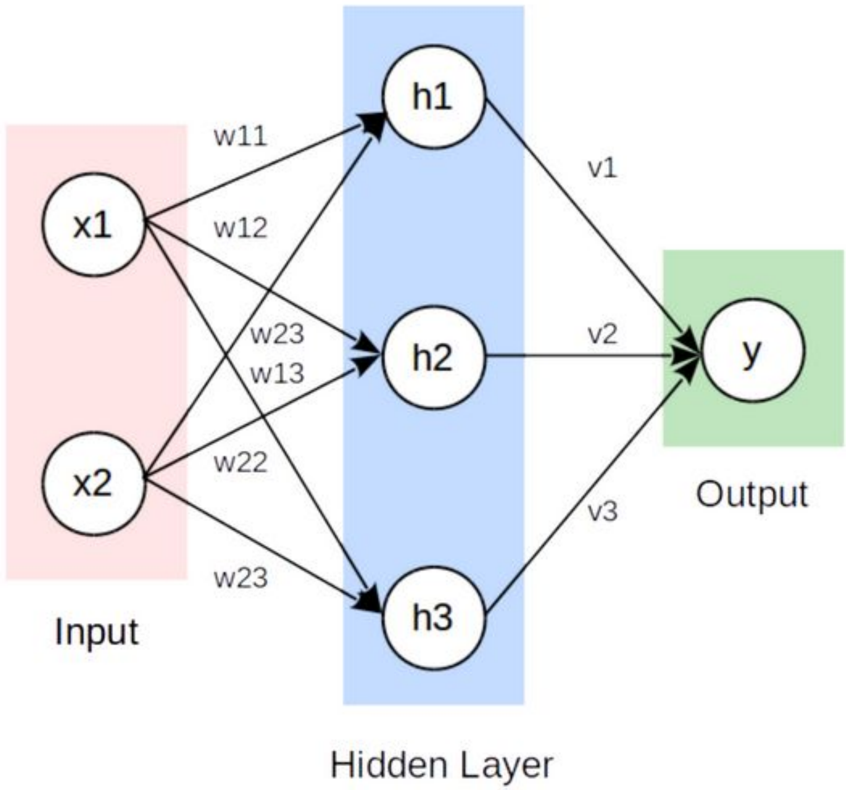
- 2-D inputs
- 3-D outputs
- Multiple hidden layers
  - 3 units in 1st hidden layer
  - 3 units in 2nd hidden layer

Exercise

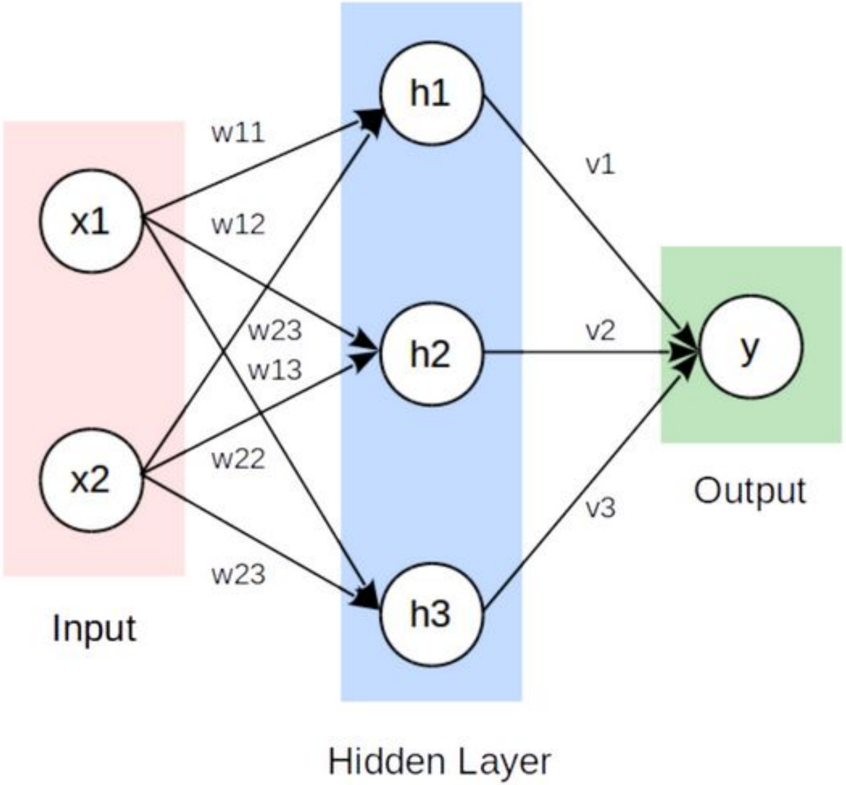
As we will begin to see, these hidden layers allow ANNs to have a high capacity to represent (and learn) complex relationships. It will be a little while before we get to the topic of ***learning***, but we can gain intuition about ANN capacity using the Tensorflow Playground



Notation and Computation

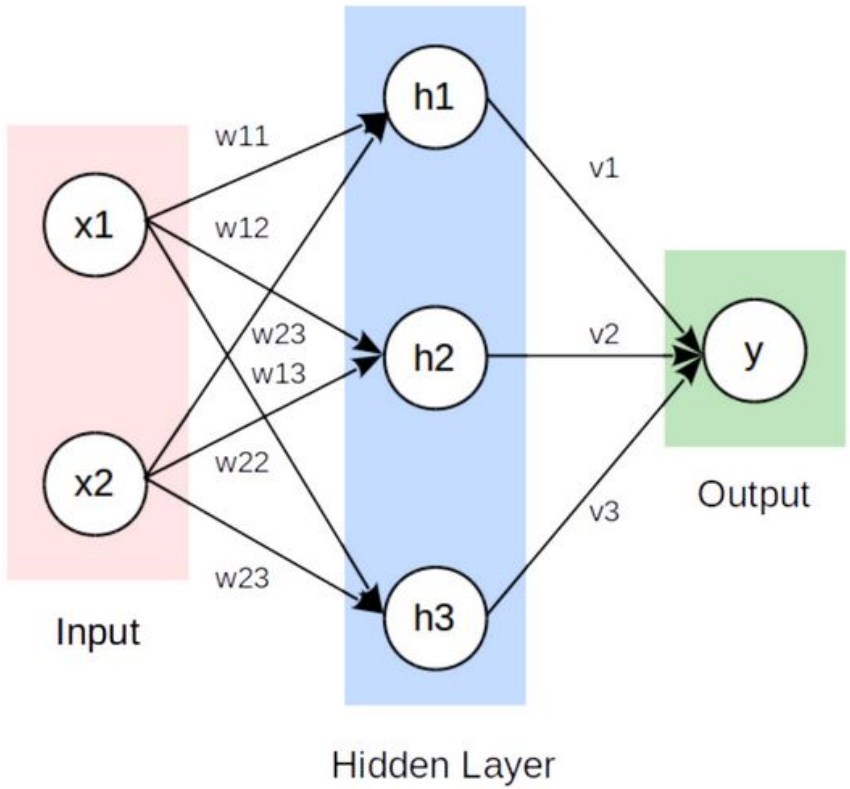


Notation and concepts are quite similar to the linear case, but we will introduce nonlinear functions as we transform variables from input (left) to output (right).



$$\vec{x}^T = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \vec{b}^T = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

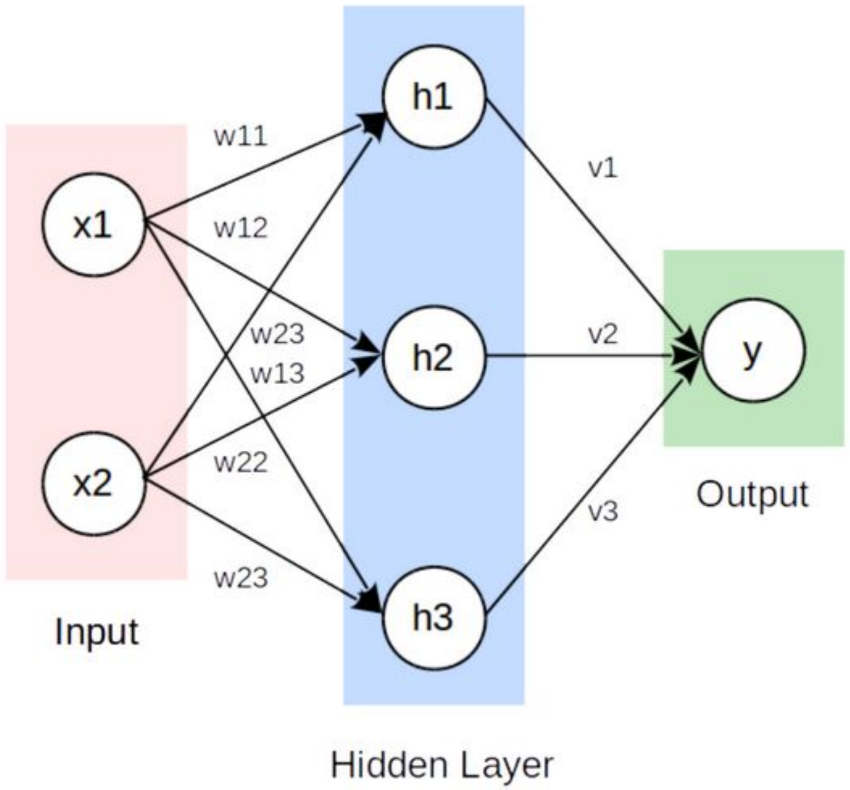
$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}$$



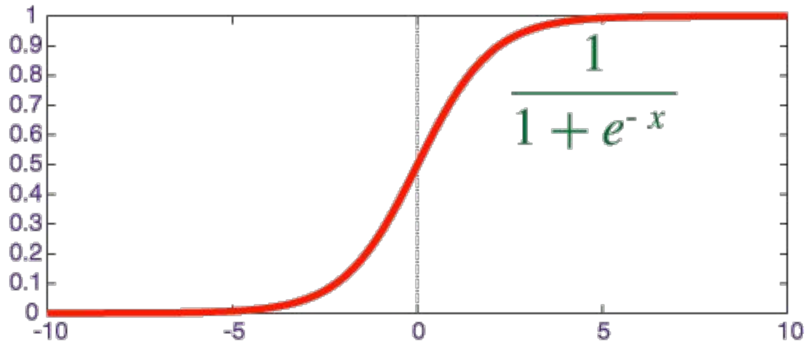
$$\vec{x}^T = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \quad \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

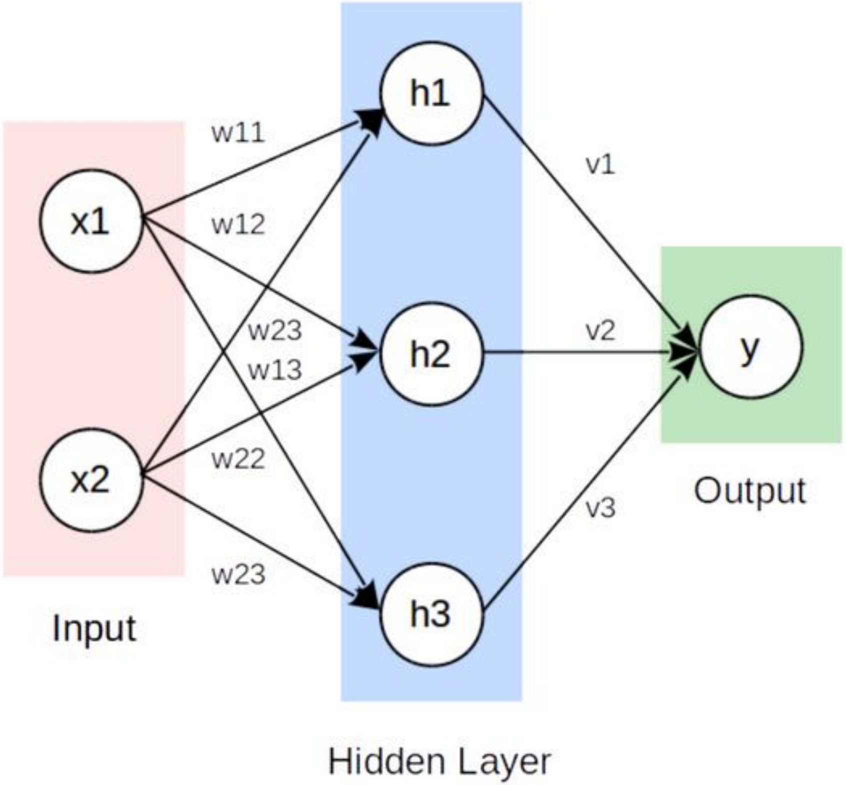
Each unit can be computed as two parts:

- Linear part: weighted sum of inputs (plus bias)
- Non-linear part: transformation of that sum by a nonlinearity of our choosing



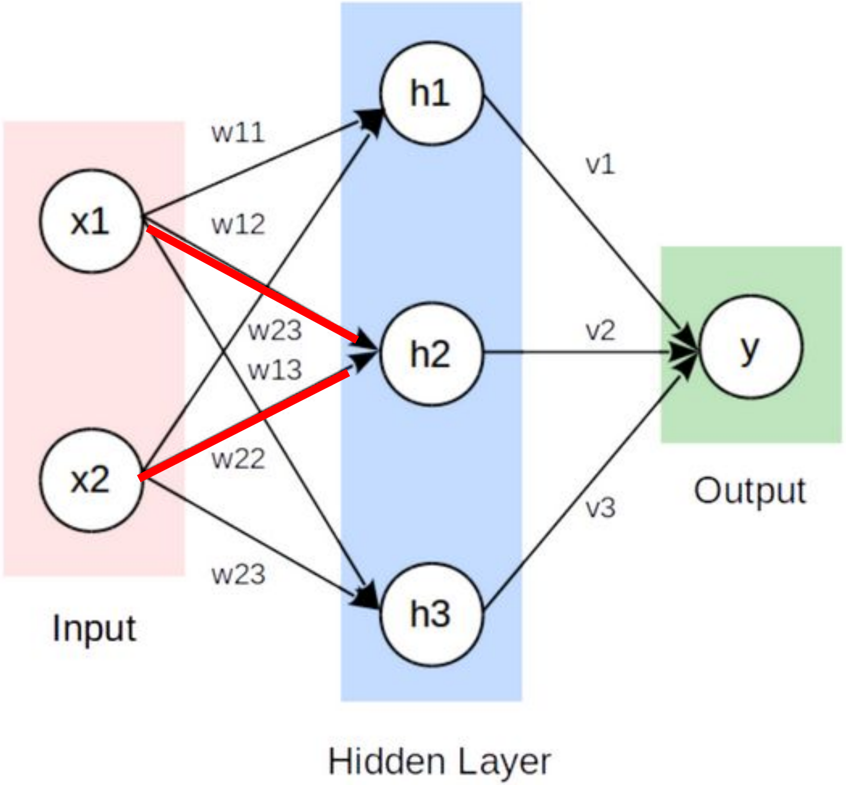
Later, we will discuss many possible non-linear functions. For now, we will refer to unnamed functions such as f(z) and g(z). For simplicity, you can keep in mind the familiar sigmoid function.





$$\vec{x}^T = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \quad \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$a_i = w_{1i}x_1 + w_{2i}x_2 + b_i$$
$$h_i = f(a_i)$$

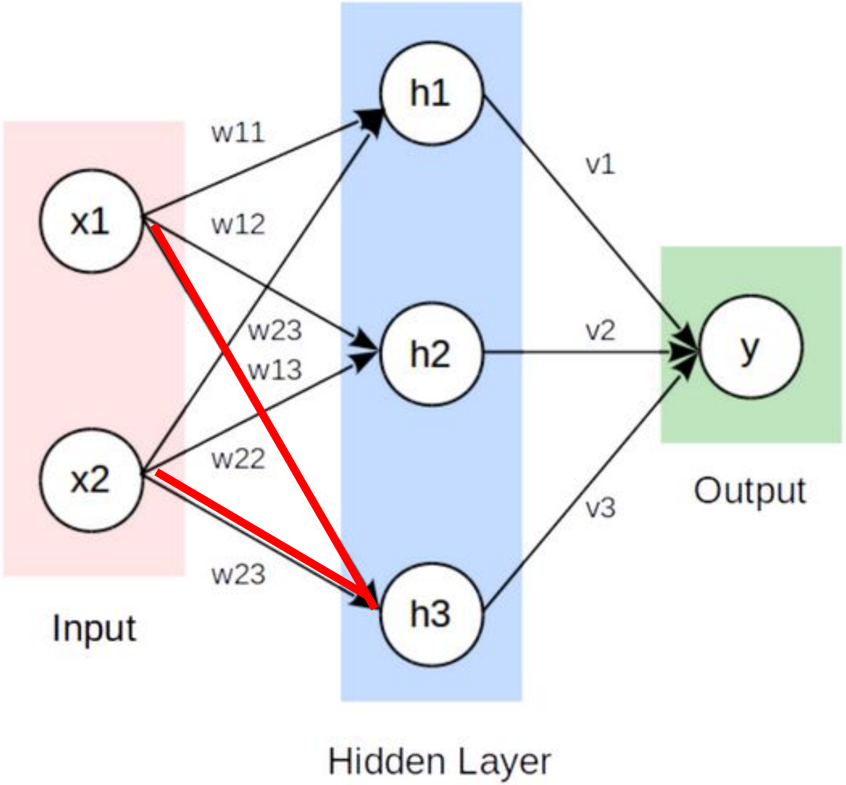


$$\vec{x}^T = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \quad \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$a_2 = w_{12}x_1 + w_{22}x_2 + b_2$$

$$h_2 = f(a_2)$$

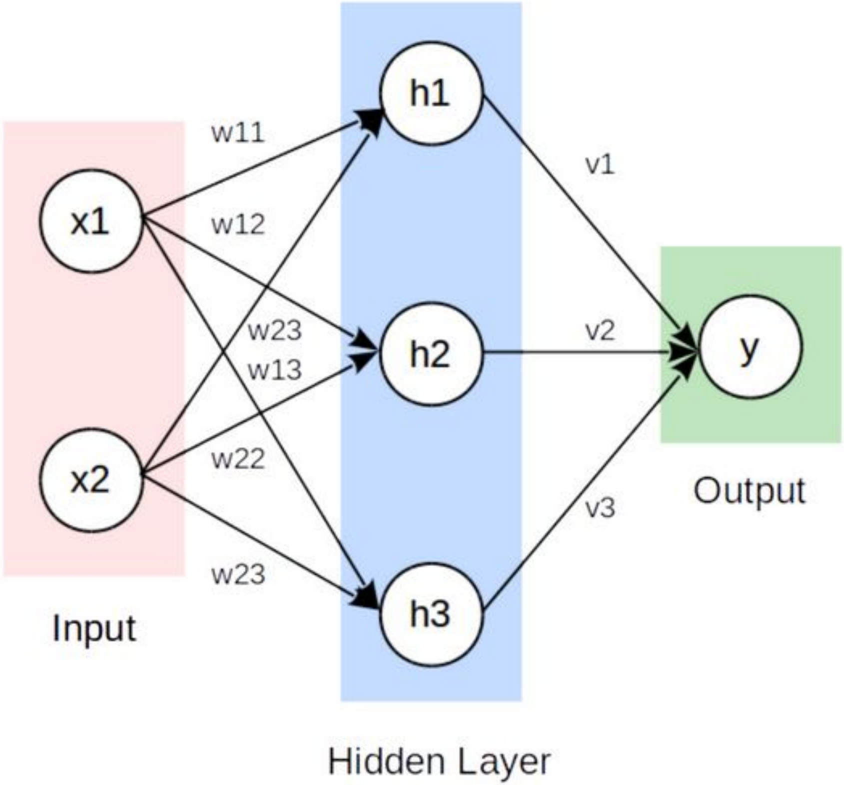
$$h_2 = f(w_{12}x_1 + w_{22}x_2 + b_2)$$



$$\vec{x}^T = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \quad \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$a_3 = w_{13}x_1 + w_{23}x_2 + b_3$$

$$h_3 = f(a_3)$$

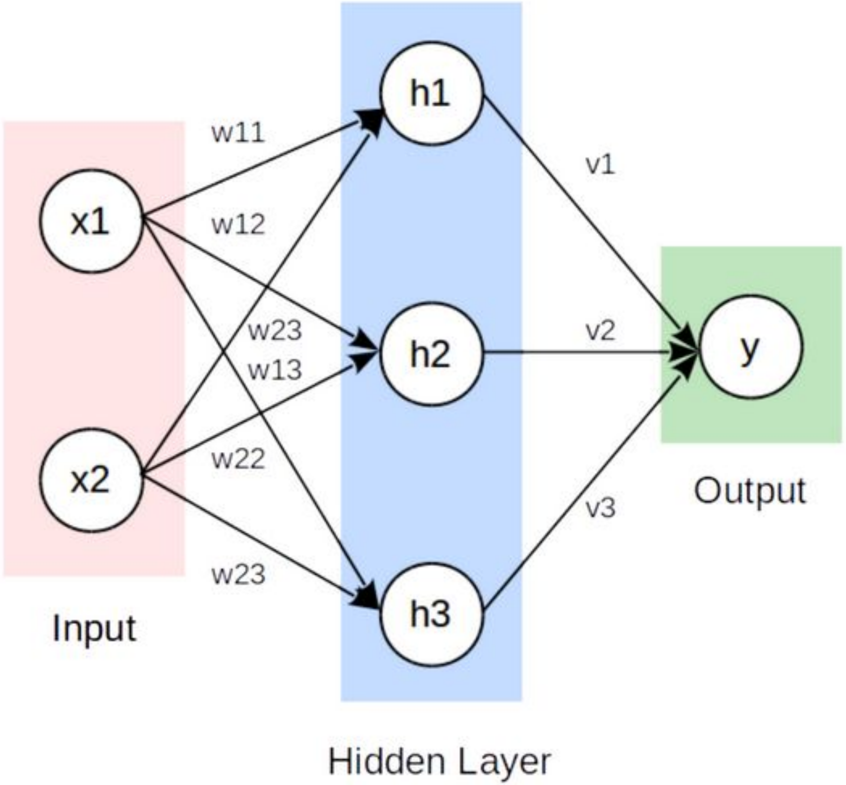


$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \qquad \vec{x}^T = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$a_i = w_{1i}x_1 + w_{2i}x_2 + b_i$$

$$a_i = \vec{x} \cdot W_i + b_i$$

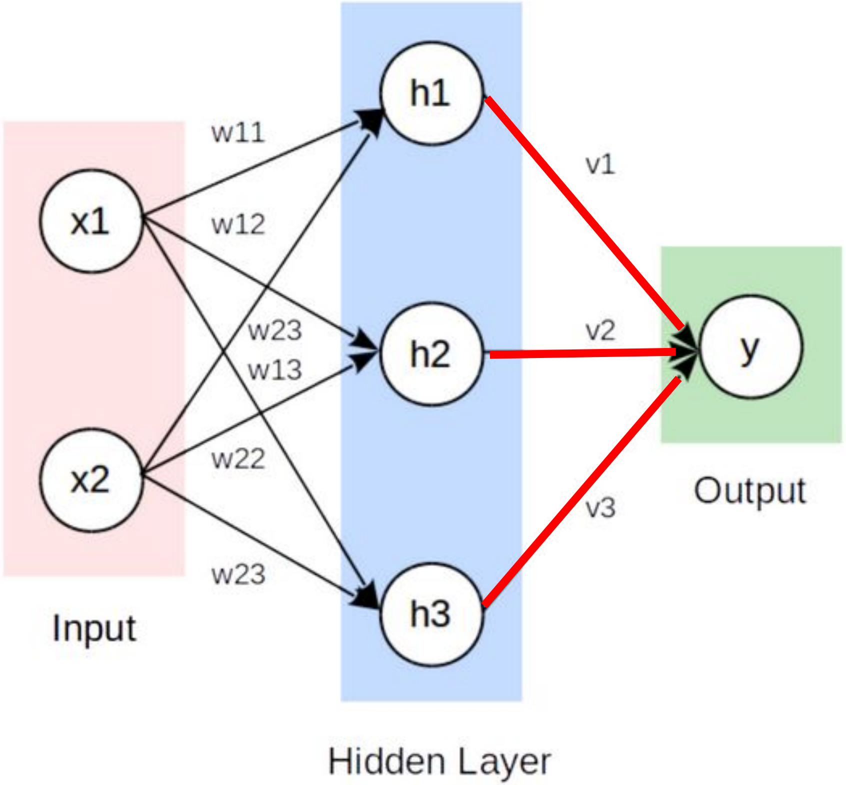
Generic Form



$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \qquad \vec{x}^T = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

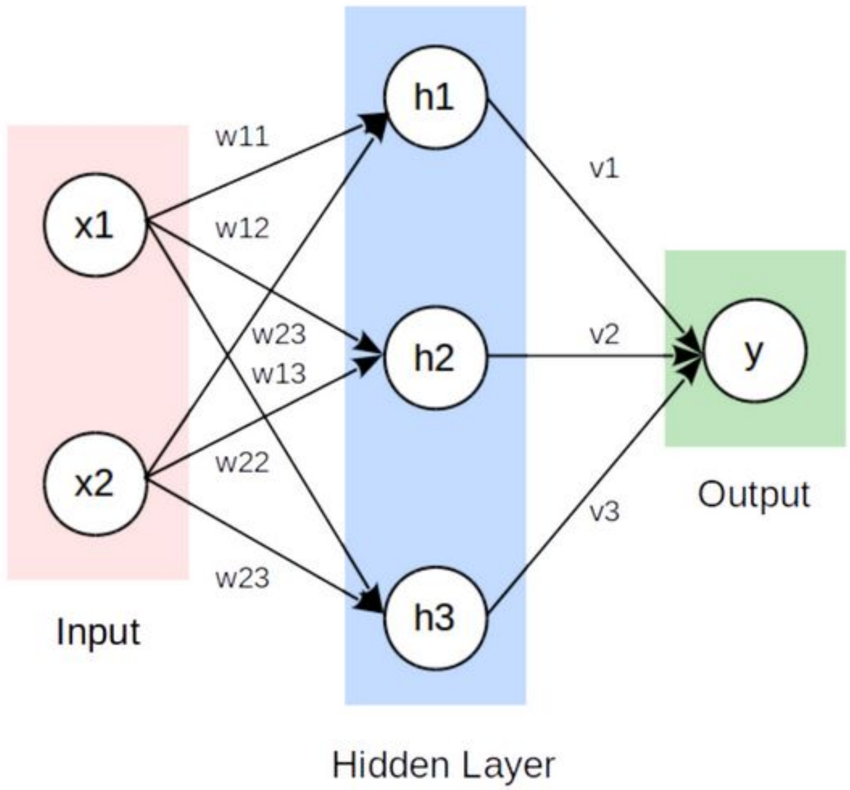
$$\vec{a} = \vec{x}W + \vec{b}$$

$$\boxed{\vec{h} = f(\vec{x}W + \vec{b})}$$



$$y = g(v_1 h_1 + v_2 h_2 + v_3 h_3 + c)$$

$$y = g(\vec{v}^T \cdot \vec{h} + c)$$

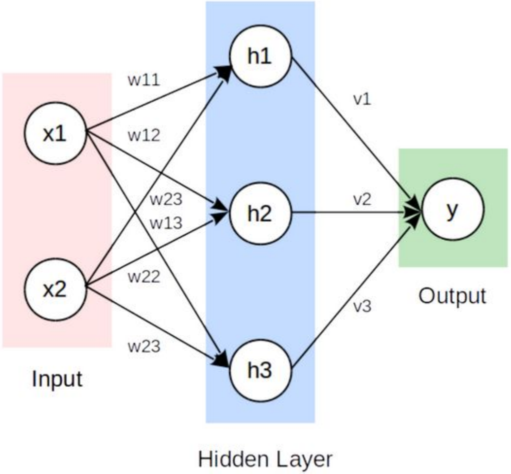


Note, y is a recursive and composite function that depends on every other variable in this graph.

$$
\begin{aligned}
y &= g(v_1 h_1 + v_2 h_2 + v_3 h_3 + c) \\
&= g(v_1 f(w_{11} x_1 + w_{12} x_2 + b_1) \\
&+ v_2 f(w_{21} x_1 + w_{22} x_2 + b_2) \\
&+ v_3 f(w_{31} x_1 + w_{32} x_2 + b_3) \\
&+ c)
\end{aligned}
$$

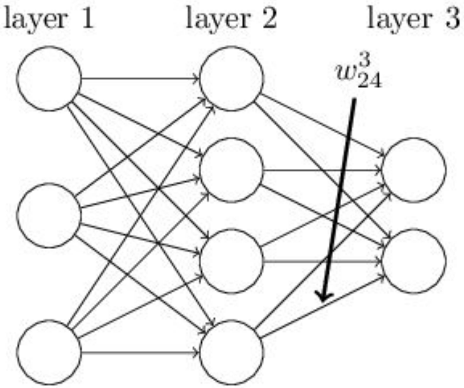
We will almost always omit these dependencies and use simplified notation. But we need to keep this in mind later, when we compute gradients.

Caution - Abuse of Notation



$$\vec{x}^T = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}$$

$$h = \sigma(XW + b)$$



w^l_{jk} is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}$$

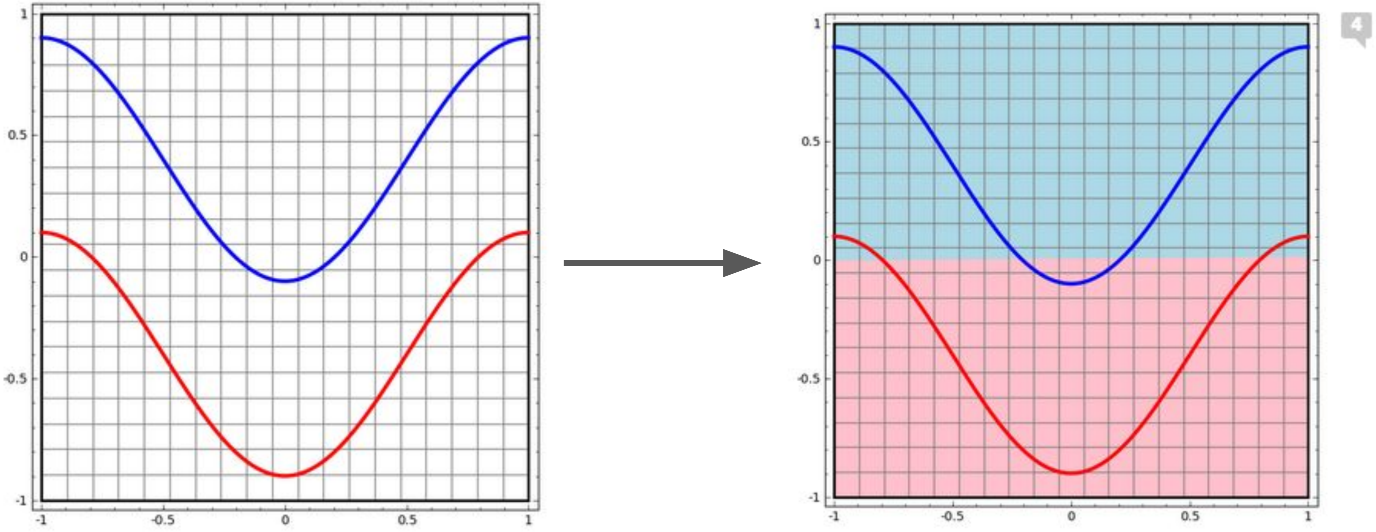
$$a^l = \sigma(w^l a^{l-1} + b^l).$$

Any of these conventions are fine, just try to remain self-consistent. When implementing, think through dimensionality of linear algebra operations and you should be fine.

ANN Representation

What are ANNs representing?

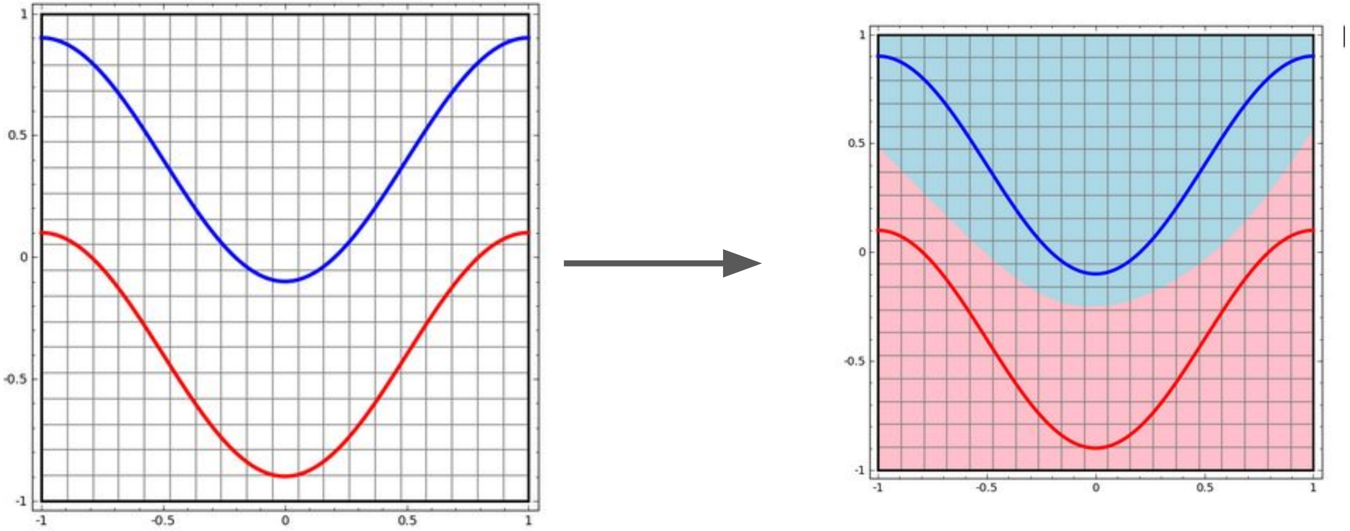
To separate the blue space from the red space, there just isn't a linear boundary that can do it.



http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

What are ANNs representing?

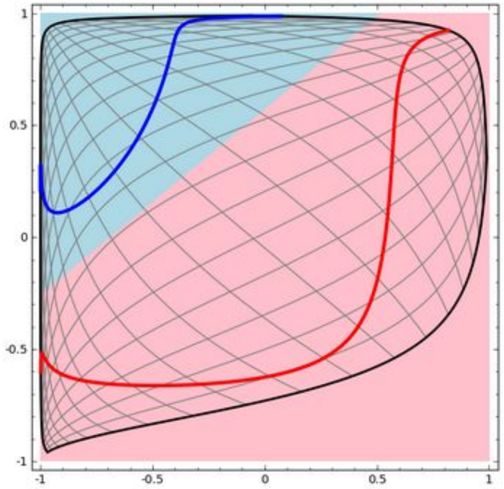
We would need to come up with a non-linear boundary to get it right.



http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

What are ANNs representing?

Equivalently, we can non-linearly transform the input space into come new representation/coordinate system. Then a linear bounday might be fine.

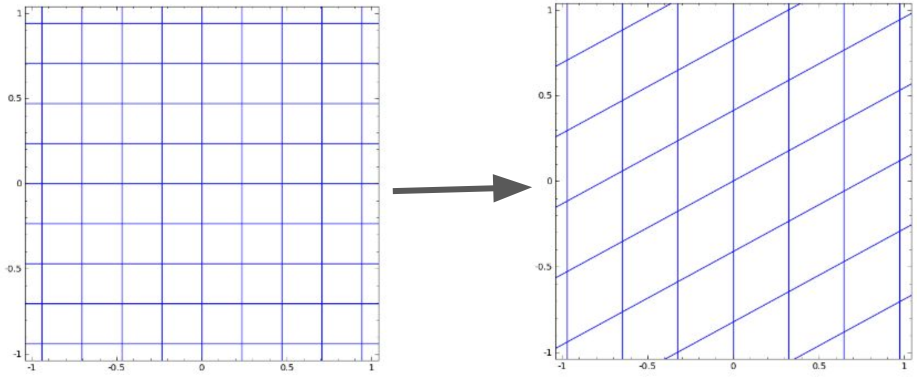


- Note the original coordinate system (the gray grid) is highly warped.
- But in this new representation, there is a linear boundary between blue and red.

http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

What are ANNs representing?

Breaking this down more slowly, into the components of what a single hidden layers does.

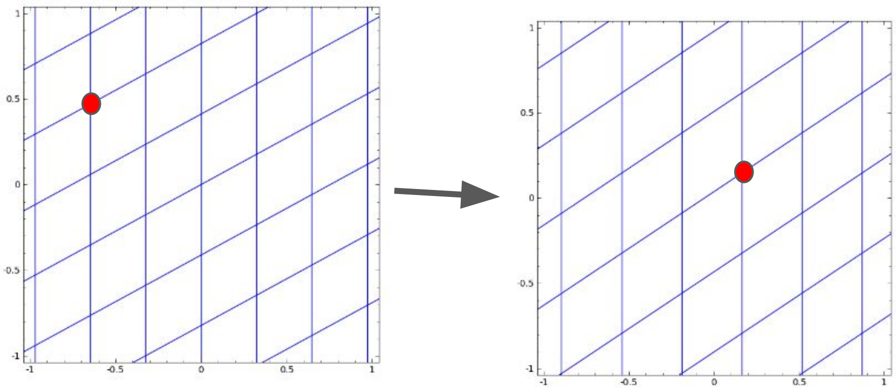


- Rotation
  - Matrix multiplication that preserves the same dimensionality can be thought of as rotating and/or shearing the input coordinates

http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

What are ANNs representing?

Breaking this down more slowly, into the components of what a single hidden layers does.

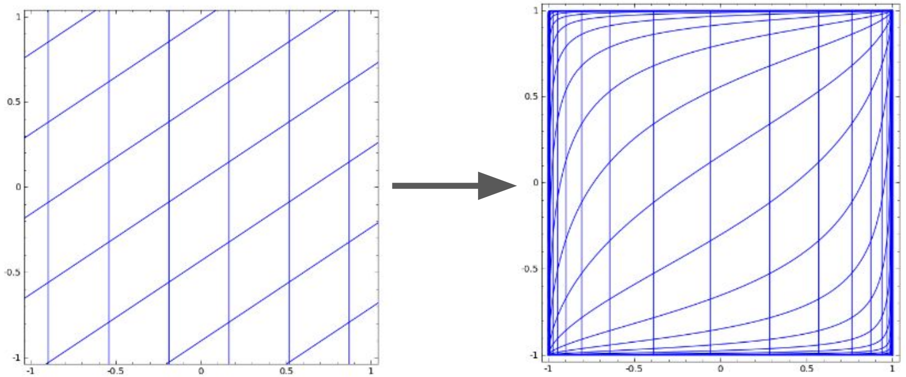


- Translation
  - Adding the bias vector simply translates our coordinate system

http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

What are ANNs representing?

Breaking this down more slowly, into the components of what a single hidden layers does.

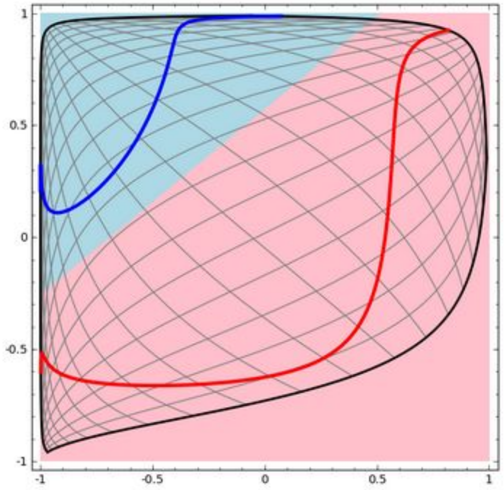


- Non-linear warping
  - Our non-linearity now warps the coordinate space
  - Most deformation is at large values of the coordinate space
  - Areas near the origin are relatively unchanged

http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

What are ANNs representing?

Breaking this down more slowly, into the components of what a single hidden layers does.

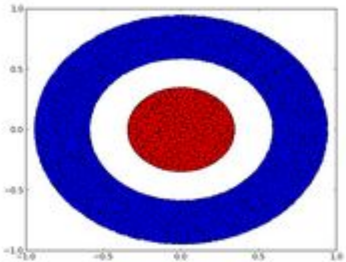


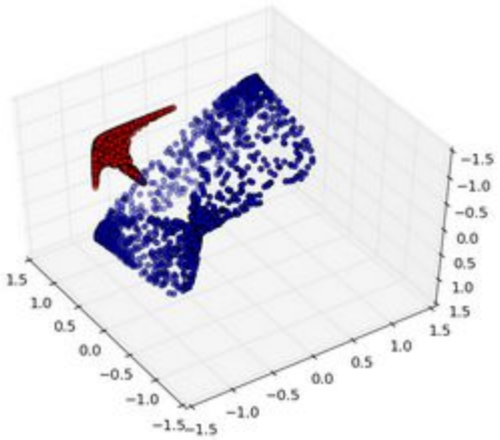
- In aggregate, this provides a flexible mechanism for learning many kinds of relationships

http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

What are ANNs representing?

Increased flexibility by increasing dimensionality of hidden layers.





- Our donut problem can only be solved by having more units in the hidden layer than there are dimensions in the input.
- But unlike before, we didn’t have to hand-design these design. We just has to pick any sufficiently flexible ANN and it will work out.

http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/