Today's Resources

Access the Notebook:

- Local Clone (**Recommended**)
  `git clone https://github.com/wtsi-hpag/xAIWorkshop`
- Online Hosting (**Slow**) Visit
  `https://githubtocolab.com/wtsi-hpag/xAIWorkshop`

Interactive Questions
(Keep open in a browser tab!)

Join `menti.com` with code `2834 2684`

wellcome sanger institute

First Principles of ML

A Peek Inside the 'Black Box' of Machine Learning

Jack Fraser-Govil
The Wellcome Sanger Institute, Hinxton, UK
15th October 2024

Seem Familiar?

Seem Familiar?

```
>> import ml_toolbox
```

Seem Familiar?

```
>> import ml_toolbox
>> data,labels = load_my_data()
```

Seem Familiar?

```
>> import ml_toolbox
>> data,labels = load_my_data()
>> model = ml_toolbox.make_model(layers=5,func=sigmoid)
```

Seem Familiar?

```
>> import ml_toolbox
>> data,labels = load_my_data()
>> model = ml_toolbox.make_model(layers=5,func=sigmoid)
>> model.Train(data,labels,epochs=1000)
```

Seem Familiar?

```
>> import ml_toolbox
>> data,labels = load_my_data()
>> model = ml_toolbox.make_model(layers=5,func=sigmoid)
>> model.Train(data,labels,epochs=1000)

Epoch 0: 0%|                | [0/1000 10it/s, loss = 10.4]
```

Seem Familiar?

```
>> import ml_toolbox
>> data,labels = load_my_data()
>> model = ml_toolbox.make_model(layers=5,func=sigmoid)
>> model.Train(data,labels,epochs=1000)

Epoch 100: 10%|xx              | [100/1000 10it/s, loss = 5.5]
```

Seem Familiar?

```
>> import ml_toolbox
>> data,labels = load_my_data()
>> model = ml_toolbox.make_model(layers=5,func=sigmoid)
>> model.Train(data,labels,epochs=1000)

Epoch 200: 20%|xxxx            | [200/1000 12it/s, loss = 3.4]
```

Seem Familiar?

```
>> import ml_toolbox
>> data,labels = load_my_data()
>> model = ml_toolbox.make_model(layers=5,func=sigmoid)
>> model.Train(data,labels,epochs=1000)

Epoch 300: 30%|xxxxxx          | [300/1000 9it/s, loss = 3.1]
```

Seem Familiar?

```
>> import ml_toolbox
>> data,labels = load_my_data()
>> model = ml_toolbox.make_model(layers=5,func=sigmoid)
>> model.Train(data,labels,epochs=1000)

Epoch 400: 40%|xxxxxx          | [400/1000 16it/s, loss = 2.9]
```

Seem Familiar?

```
>> import ml_toolbox
>> data,labels = load_my_data()
>> model = ml_toolbox.make_model(layers=5,func=sigmoid)
>> model.Train(data,labels,epochs=1000)

Epoch 500: 50%|xxxxxxxxx       | [500/1000 12it/s, loss = 1.4]
```

Seem Familiar?

```
>> import ml_toolbox
>> data,labels = load_my_data()
>> model = ml_toolbox.make_model(layers=5,func=sigmoid)
>> model.Train(data,labels,epochs=1000)

Epoch 700: 70%|xxxxxxxxxxxx    | [700/1000 5it/s, loss = 1.39]
```

Seem Familiar?

```
>> import ml_toolbox
>> data,labels = load_my_data()
>> model = ml_toolbox.make_model(layers=5,func=sigmoid)
>> model.Train(data,labels,epochs=1000)

Epoch 900: 90%|xxxxxxxxxxxxxxx | [900/1000 8it/s, loss = 1.385]
```

Seem Familiar?

```
>> import ml_toolbox
>> data,labels = load_my_data()
>> model = ml_toolbox.make_model(layers=5,func=sigmoid)
>> model.Train(data,labels,epochs=1000)

Epoch 1000: 100%|xxxxxxxxxxxxxxxx| [final loss = 1.382]
```

Seem Familiar?

```
>> import ml_toolbox
>> data,labels = load_my_data()
>> model = ml_toolbox.make_model(layers=5,func=sigmoid)
>> model.Train(data,labels,epochs=1000)

Epoch 1000: 100%|xxxxxxxxxxxxxxxx| [final loss = 1.382]

>> model.Test()
```

Today's Agenda

The aim for today is:

Today's Agenda

The aim for today is:

- Classic Perceptron

Today's Agenda

The aim for today is:

- Classic Perceptron
- Feedforward Networks

Today's Agenda

The aim for today is:

- Classic Perceptron
- Feedforward Networks
- Non-Linearity

Today's Agenda

The aim for today is:

- Classic Perceptron
- Feedforward Networks
- Non-Linearity
- Optimisation & Backpropagation

Today's Agenda

The aim for today is:

- Classic Perceptron
- Feedforward Networks
- Non-Linearity
- Optimisation & Backpropagation

As we progress you will slowly build up your own ML toolkit, built entirely from scratch!

A Warning

There will be equations.

A Warning

There will be equations.
You will need to know what they mean!

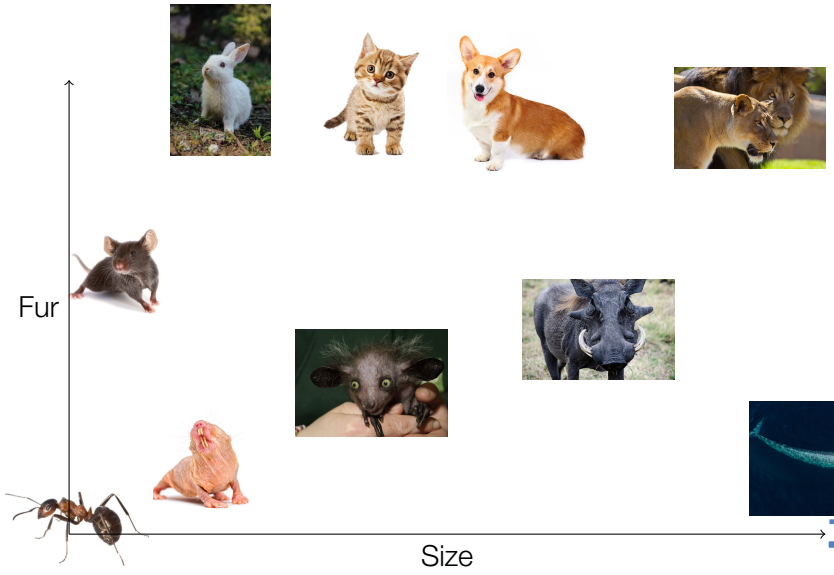
A Warning

There will be equations.
You will need to know what they mean!

Please, please, please, ask if you want clarification on the underlying mathematics and theory! That's why you're here today!

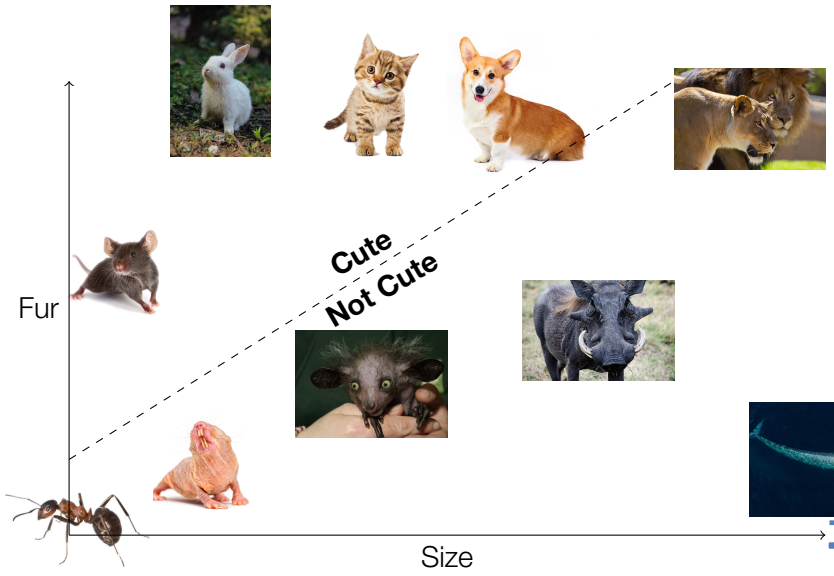
Part 1
The Perceptron

Basic Decision Making: Defining Cuteness



wellcome sanger institute

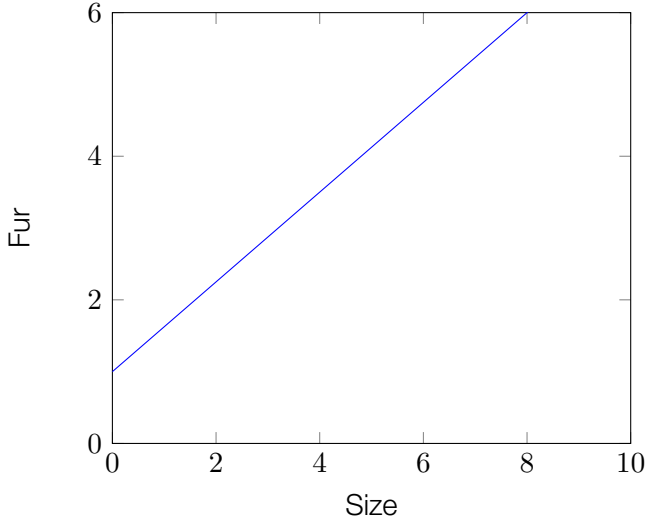
Basic Decision Making: Defining Cuteness



wellcome sanger institute

Splitting The Plane

In order to split the plane into two parts, we merely need to define a *line*.



Splitting the plane

Splitting the plane

Question: If we have N dimensions ($N = 2$), how many parameters do we need to define a line?

Splitting the plane

Question: If we have N dimensions ($N = 2$), how many parameters do we need to define a line?

Answer: The answer is N - in 2 dimensions, this is friendly $y = mx + c \rightarrow (m, c)$

The Perceptron

The Perceptron classifier algorithm is:

$$P(\mathbf{x}) = \begin{cases} 1 & \text{if } b + \mathbf{w} \cdot \mathbf{x}^1 > 0 \\ 0 & \text{else} \end{cases} \tag{1}$$

[1] The dot/inner product is covered in Section 3.1 in the notes

The Perceptron

The Perceptron classifier algorithm is:

$$P(\mathbf{x}) = \begin{cases} 1 & \text{if } \tilde{\mathbf{x}} \cdot \mathbf{w}^{1} > 0 \\ 0 & \text{else} \end{cases} \tag{1}$$

Where

$$\tilde{\mathbf{x}} = \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$$

[1] The dot/inner product is covered in Section 3.1 in the notes

The Perceptron

The Perceptron classifier algorithm is:

$$P(\mathbf{x}) = \begin{cases} 1 & \text{if } \tilde{\mathbf{x}} \cdot \mathbf{w}^1 > 0 \\ 0 & \text{else} \end{cases} \tag{1}$$

Where

$$\tilde{\mathbf{x}} = \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$$

$\mathbf{w}$ are the **weights**.

[1] The dot/inner product is covered in Section 3.1 in the notes

The Perceptron: Dimensions

Question: What is the dimensionality of the Perceptron? Does this conflict with our earlier statements?

The Perceptron: Dimensions

Question: What is the dimensionality of the Perceptron? Does this conflict with our earlier statements?

Hint: It has nothing to do with the 'bias'!

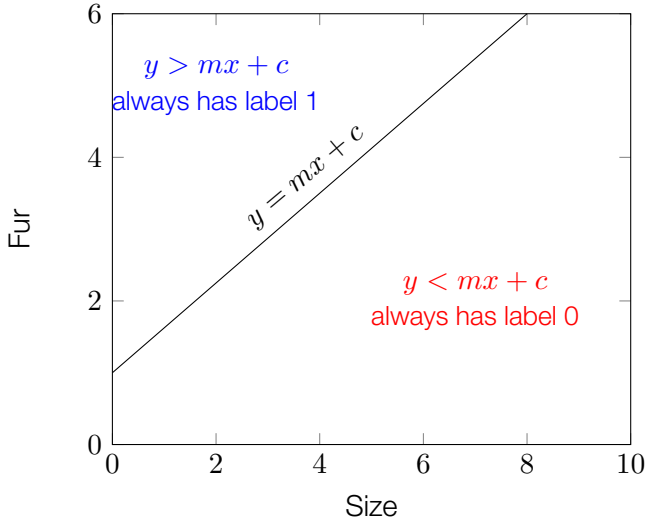
The Perceptron: Dimensions

Question: What is the dimensionality of the Perceptron? Does this conflict with our earlier statements?

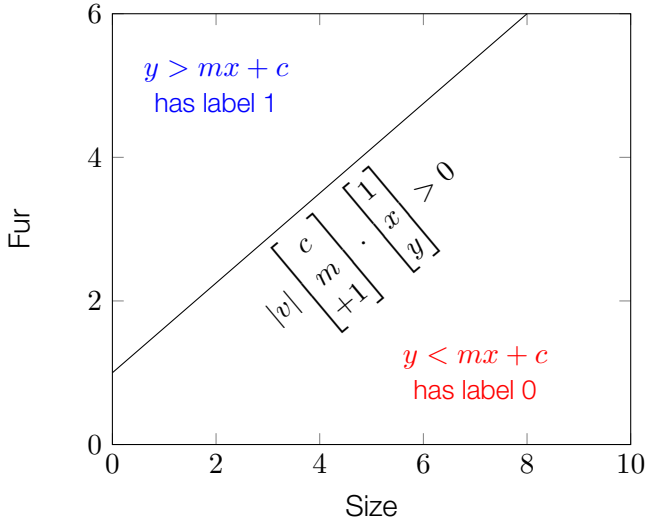
Hint: It has nothing to do with the 'bias'!

Answer: We need $N + 1$ dimensions. N dimensions define a line - but need an additional dimension to add *directionality*

The Perceptron: Dimensions

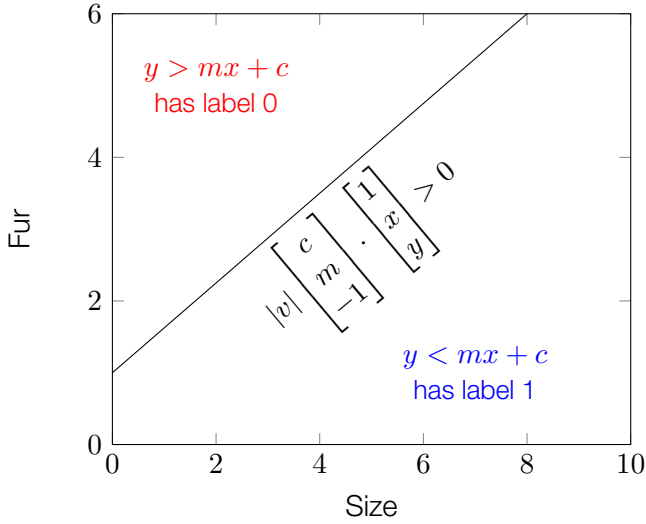


The Perceptron: Dimensions



wellcome sanger institute

The Perceptron: Dimensions



wellcome sanger institute

Exercise 1: Perceptron Classifier

Task 1: Write a perceptron classifier for the data in `cuteness.dat`

Guidance: Class structure & predetermined weights are provided in the notebook. Ignore `Train()` for now

Question: The provided weights classify only one animal correctly. Which is it?

Training A Perceptron

Training a perceptron is easy. Loop over the data, make a prediction (P), compare it to the truth (T) and if it is wrong:

Training A Perceptron

Training a perceptron is easy. Loop over the data, make a prediction (P), compare it to the truth (T) and if it is wrong:

$$\mathbf{w} \rightarrow \mathbf{w} + r \times (T - P)\, \tilde{\mathbf{x}} \tag{2}$$

Training A Perceptron

Training a perceptron is easy. Loop over the data, make a prediction (P), compare it to the truth (T) and if it is wrong:

$$\mathbf{w} \rightarrow \mathbf{w} + r \times (T - P)\, \tilde{\mathbf{x}} \tag{2}$$

This works because it always makes $\mathbf{w} \cdot \tilde{\mathbf{x}}$ move closer towards zero (and hence to the tipping point of altering its decision).

Exercise 2: Train Your Perceptron

Task 2: Write the `Train()` method for your Perceptron, then apply to `cuteness` data.

Exercise 2: Train Your Perceptron

Task 2: Write the `Train()` method for your Perceptron, then apply to `cuteness` data.

Question: How long does it take to train to 100% accuracy?

Exercise 2: Train Your Perceptron

Task 2: Write the `Train()` method for your Perceptron, then apply to `cuteness` data.

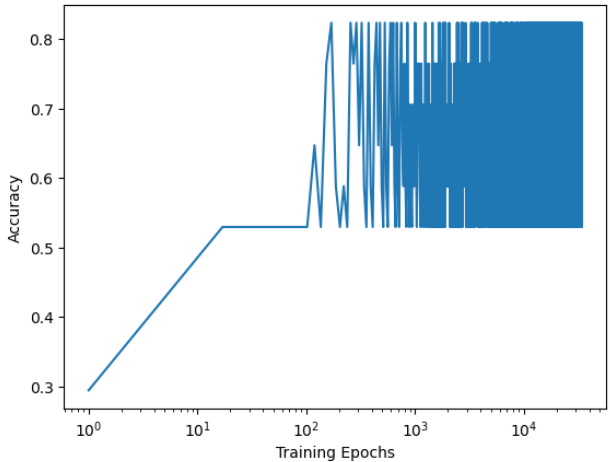
Question: How long does it take to train to 100% accuracy?
Give your answers on `menti.com` with code `2834 2684`

Exercise 3: The Limits of Linearity

Task 3: Apply your method to the data in `cuteness_augmented.dat`, which adds the Axolotl, Otter, Capybara, Quokka, Slug, Tasmanian Devil and Giant Panda.
Question: What has gone wrong? Who are the troublemakers?

Exercise 3: The Limits of Linearity

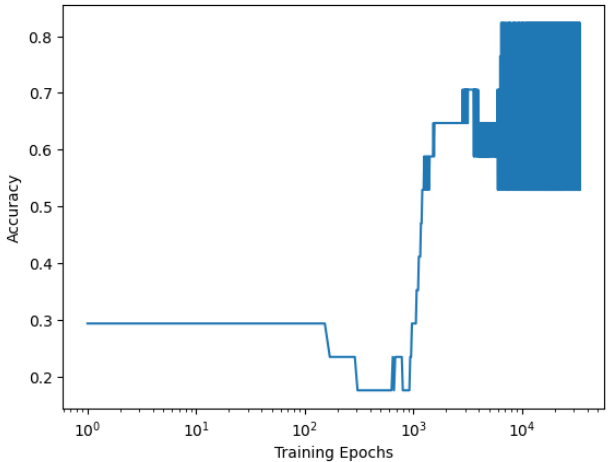
Task 3: Apply your method to the data in `cuteness_augmented.dat`, which adds the Axolotl, Otter, Capybara, Quokka, Slug, Tasmanian Devil and Giant Panda.
Question: What has gone wrong? Who are the troublemakers?



wellcome sanger institute

Exercise 3: The Limits of Linearity

Task 3: Apply your method to the data in `cuteness_augmented.dat`, which adds the Axolotl, Otter, Capybara, Quokka, Slug, Tasmanian Devil and Giant Panda.
Question: What has gone wrong? Who are the troublemakers?



wellcome sanger institute

The Nonlinear Perceptron

Eminently possible to have a non-linear perceptron.

The Nonlinear Perceptron

Eminently possible to have a non-linear perceptron.
Pass $\mathbf{x}$ through a non-linear transform to make it:

$$\mathbf{x}' = \begin{pmatrix} 1 \\ x \\ y \\ x^2 \\ \sin(x) \\ x^2 \exp(y) \\ \cos(\sin(\exp(\sin(\log(x^2 + 9xy))))) \\ \vdots \end{pmatrix} \tag{3}$$

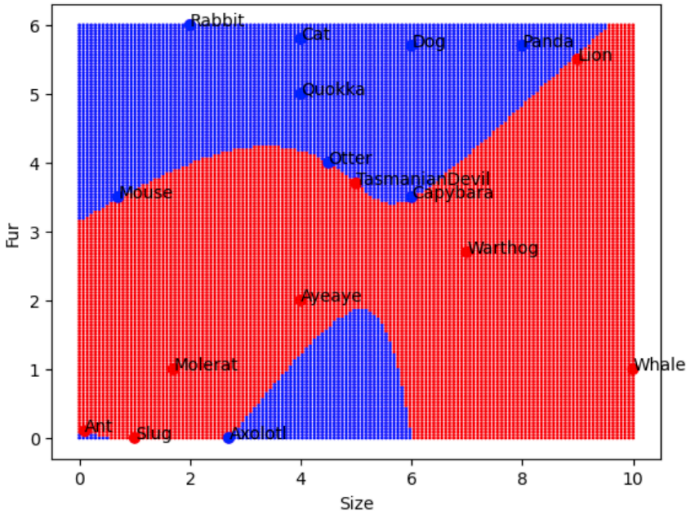
The Nonlinear Perceptron

Here is the results of a perceptron where:

$$\mathbf{x}'(x,y) = \begin{pmatrix} 1 & x & y & x^2 & xy & y^2 & x^3 & x^2y & xy^2 & y^3 & x^4 & x^3y & x^2y^2 & xy^3 & y^4 \end{pmatrix}^\intercal \tag{4}$$

Bonus Question: Why did I choose this?

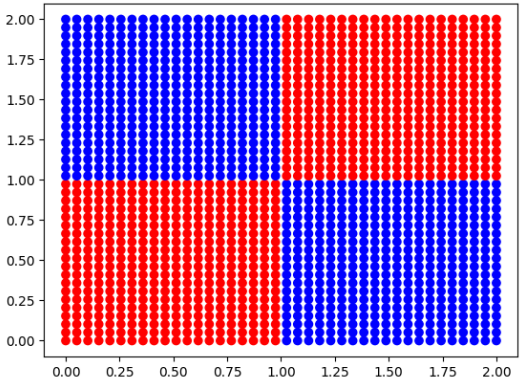
The Nonlinear Perceptron



wellcome sanger institute

The Limits of NonLinearity

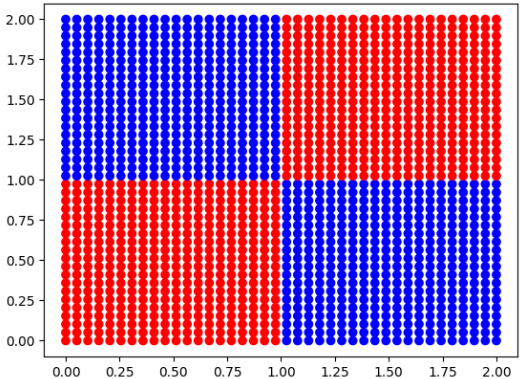
Target



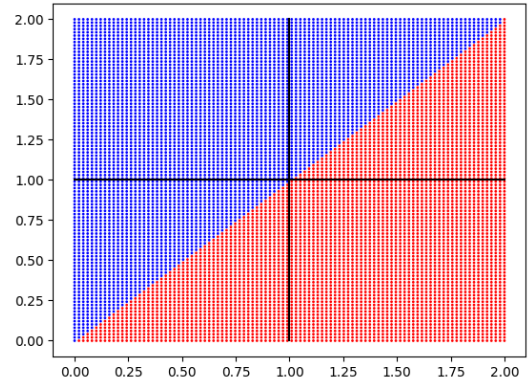
wellcome sanger institute

The Limits of NonLinearity

Target



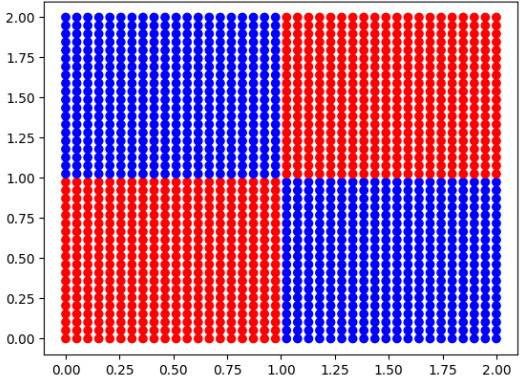
Reality (1st Order - 3 parameters)



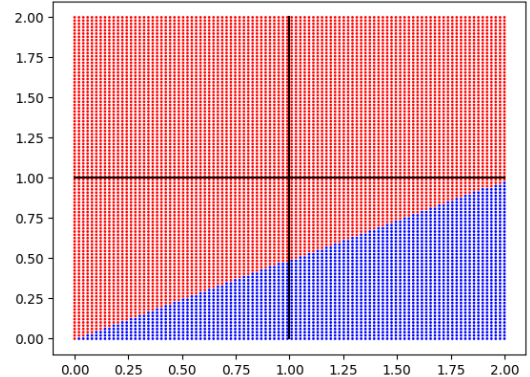
wellcome sanger institute

The Limits of NonLinearity

Target



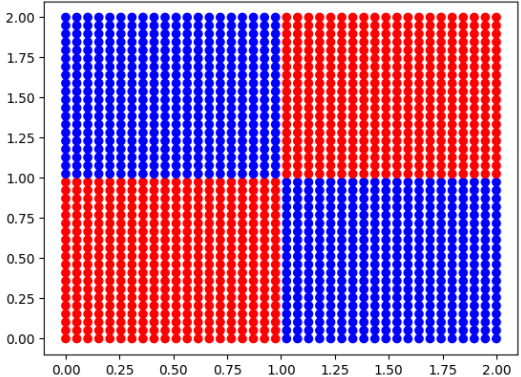
Reality (2nd Order - 6 parameters)



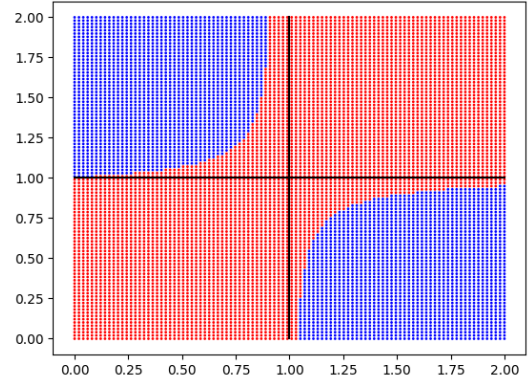
wellcome sanger institute

The Limits of NonLinearity

Target



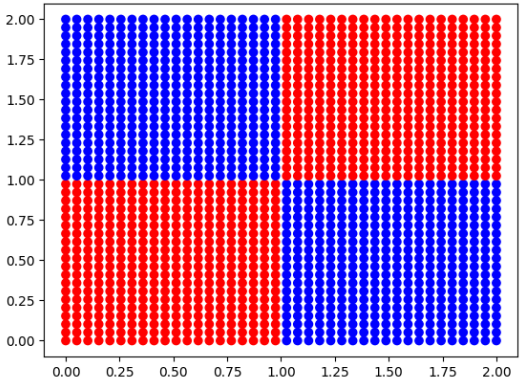
Reality (3rd Order - 10 parameters)



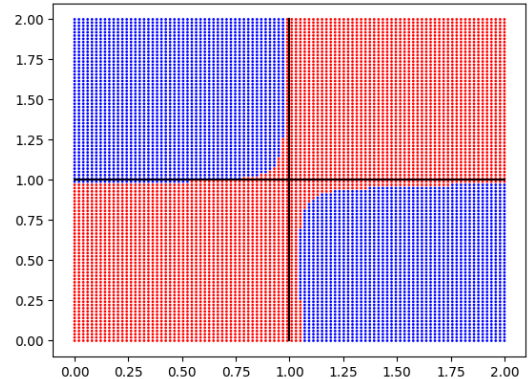
wellcome sanger institute

The Limits of NonLinearity

Target



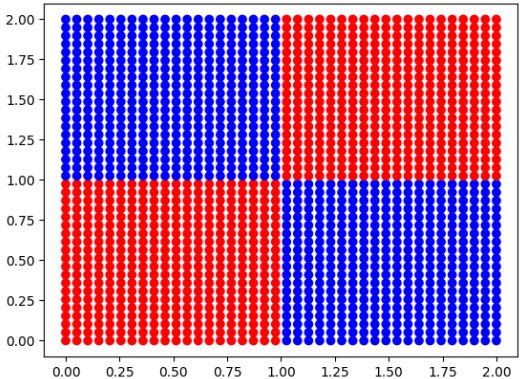
Reality (4th Order - 15 parameters)



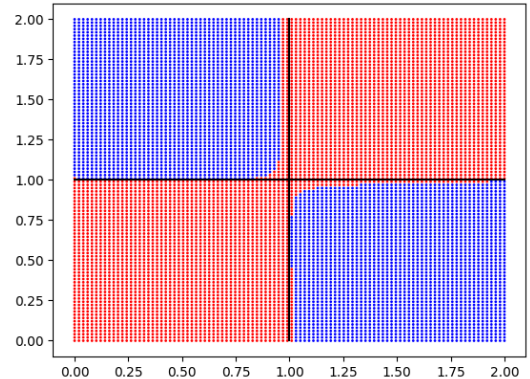
wellcome sanger institute

The Limits of NonLinearity

Target



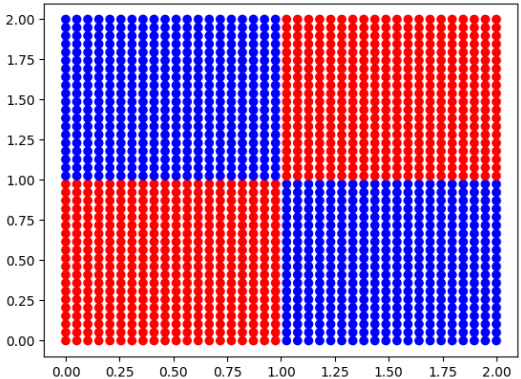
Reality (5th Order - 21 parameters)



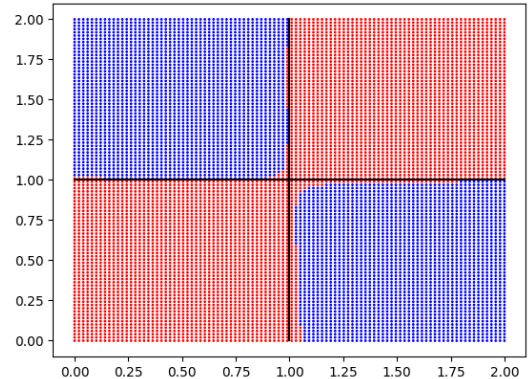
wellcome sanger institute

The Limits of NonLinearity

Target



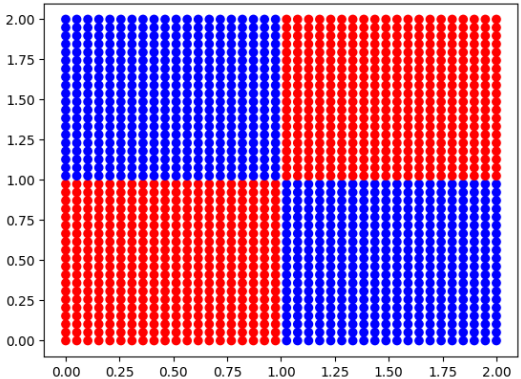
Reality (5th Order - 21 parameters)



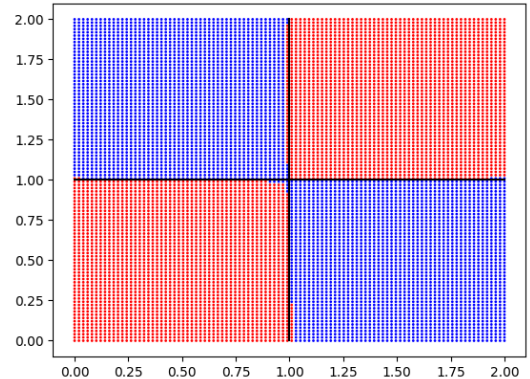
wellcome sanger institute

The Limits of NonLinearity

Target



Reality (20th Order - 231 parameters)



wellcome sanger institute

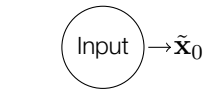
What do to?

What do to?

MORE!

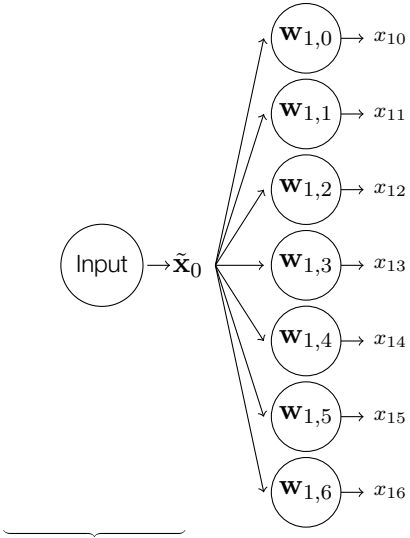
The Multilayered Perceptron

The Multilayered Perceptron

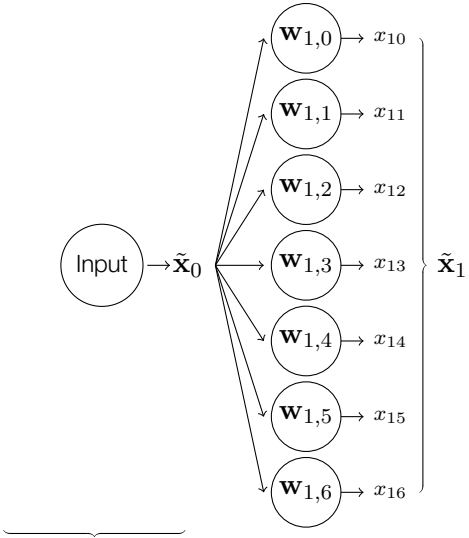


Input layer

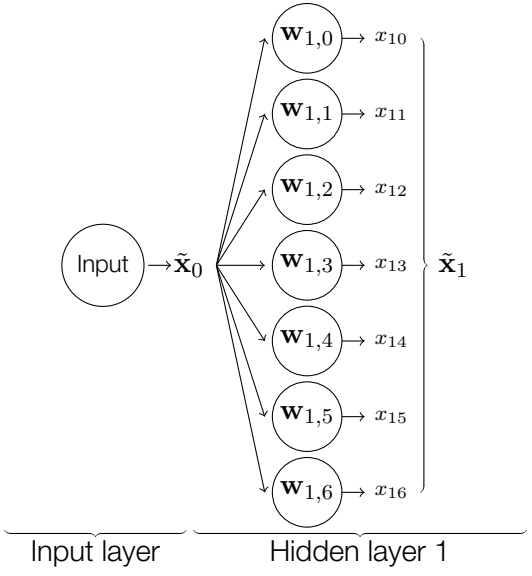
The Multilayered Perceptron



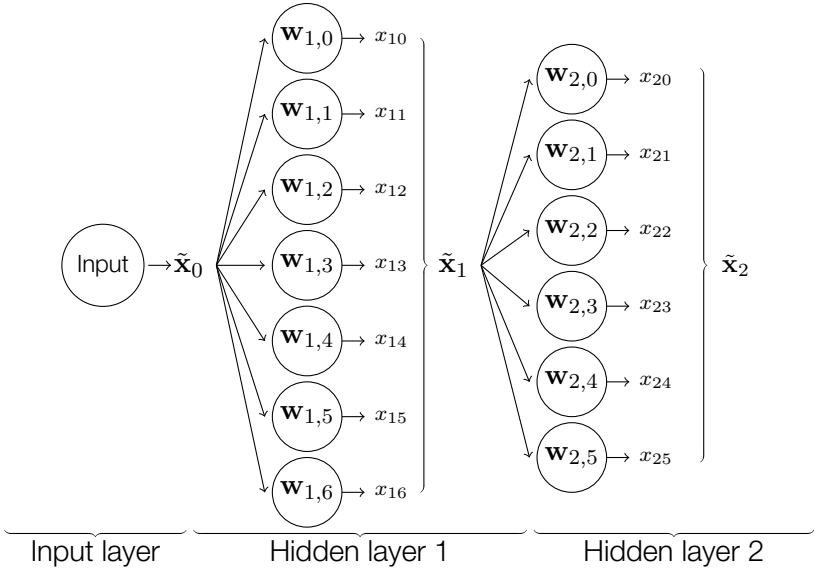
The Multilayered Perceptron



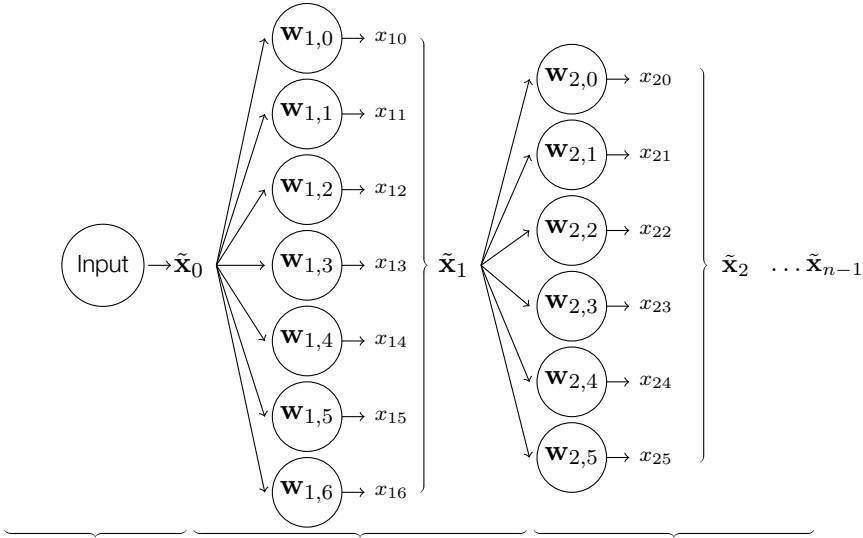
The Multilayered Perceptron



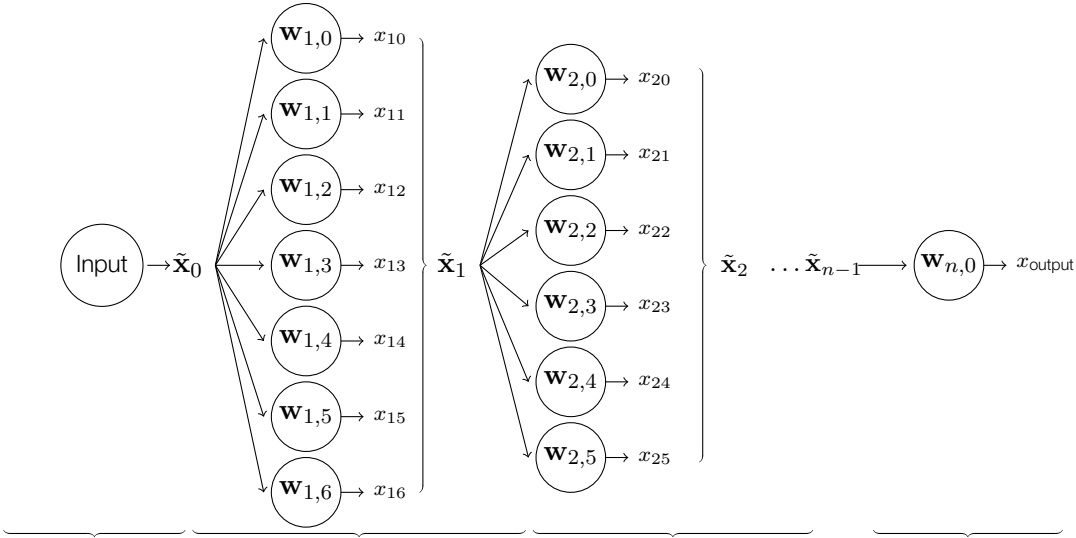
The Multilayered Perceptron



The Multilayered Perceptron



The Multilayered Perceptron



Exercise 4: The Multilayered Perceptron

Task 4: Write a Feedforward Network.

Exercise 4: The Multilayered Perceptron

Task 4: Write a Feedforward Network.

Guidance: I have provided the skeleton of a Node, Layer and Network class. Randomly initialise your weights between -1 and 1.

Exercise 4: The Multilayered Perceptron

Task 4: Write a Feedforward Network.

Guidance: I have provided the skeleton of a Node, Layer and Network class. Randomly initialise your weights between -1 and 1.

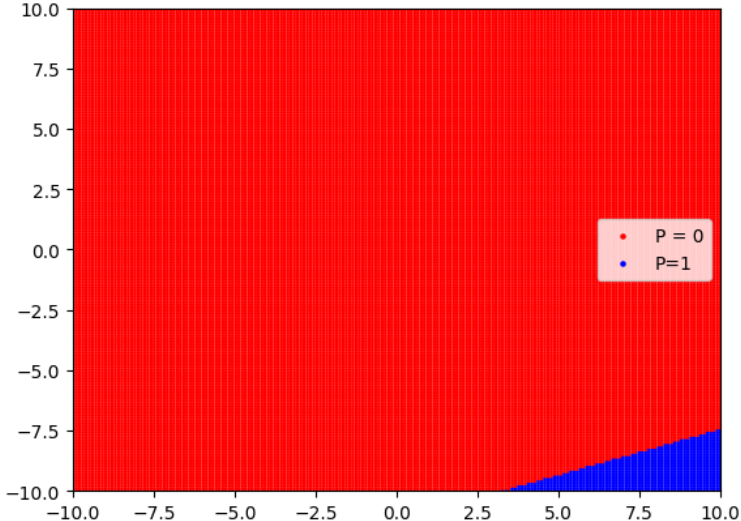
Question: Do you notice anything odd about how this predictor behaves?

All that for nothing?

If everything went well, you should see something like this:

All that for nothing?

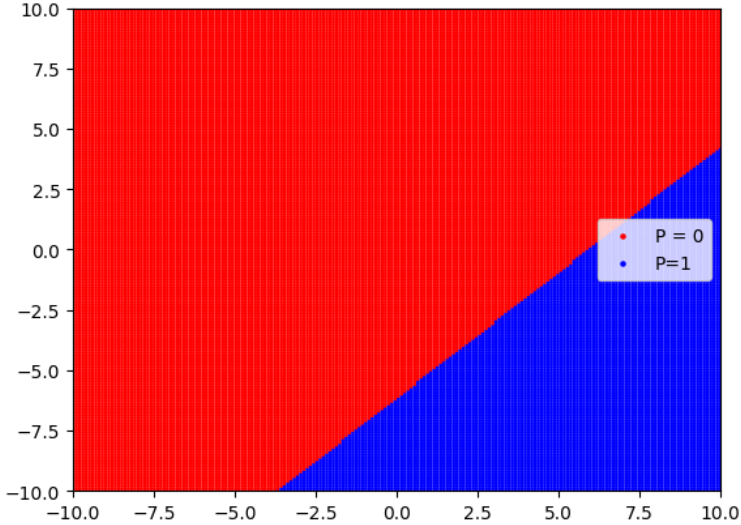
If everything went well, you should see something like this:



wellcome sanger institute

All that for nothing?

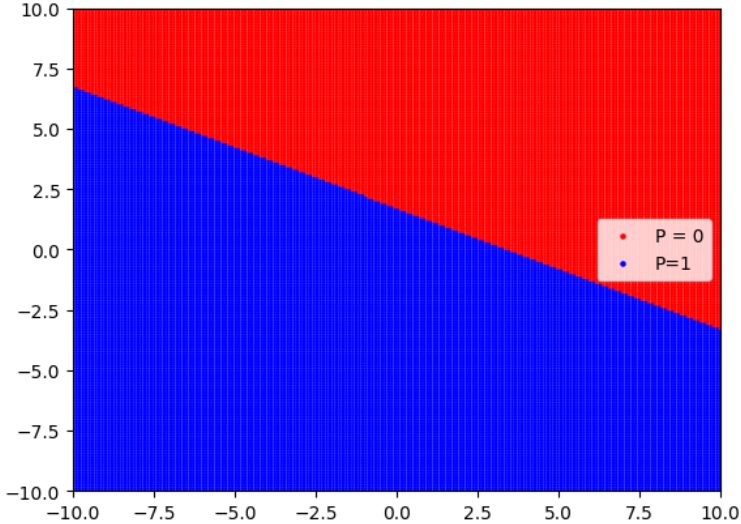
If everything went well, you should see something like this:



wellcome sanger institute

All that for nothing?

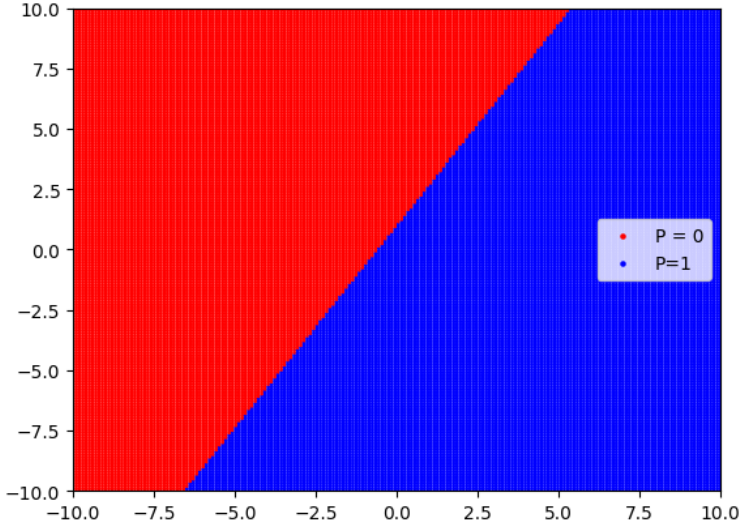
If everything went well, you should see something like this:



wellcome sanger institute

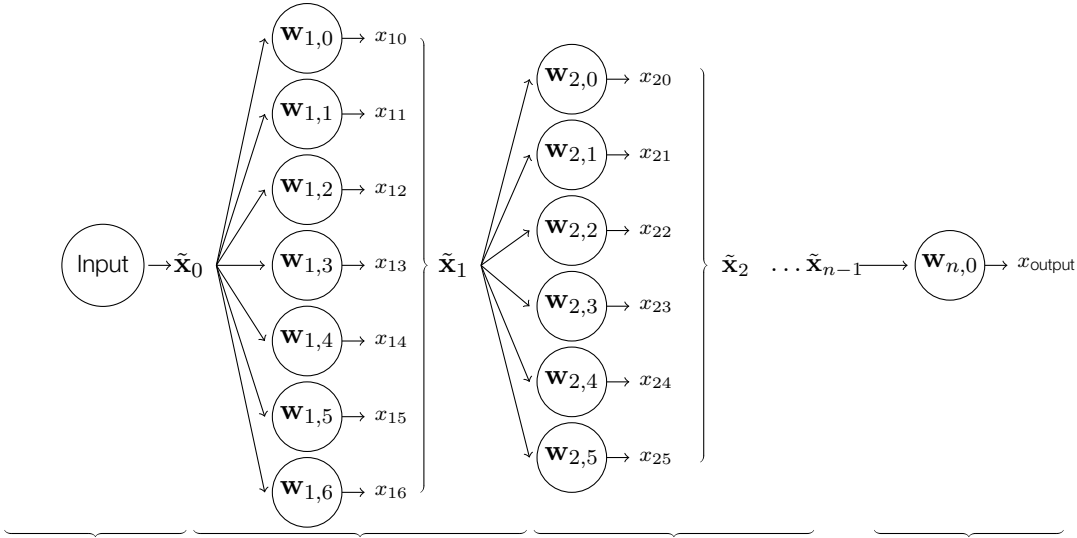
All that for nothing?

If everything went well, you should see something like this:

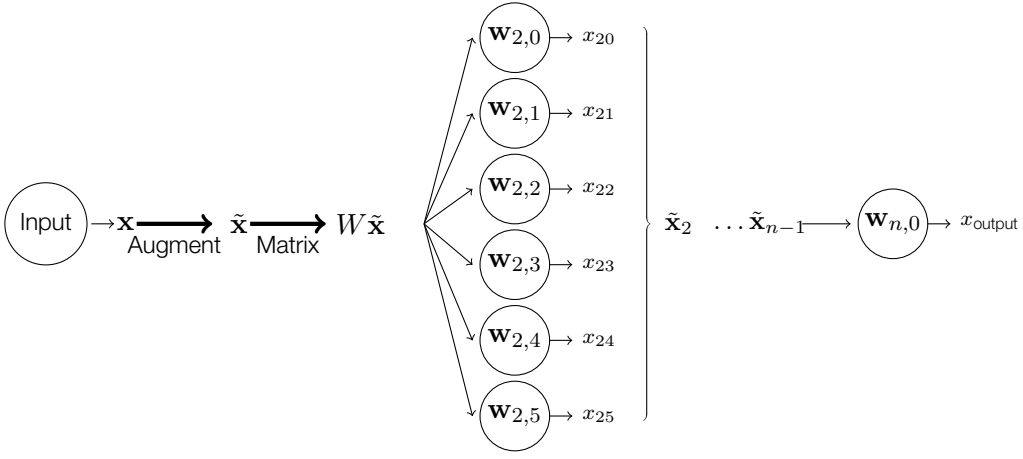


wellcome sanger institute

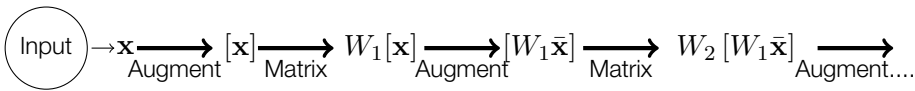
Affine Mess We Find Ourselves in



Affine Mess We Find Ourselves in



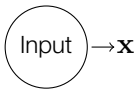
Affine Mess We Find Ourselves in



Affine Mess We Find Ourselves in

Or, in terms of **Affine Operators** (Chapter 3.4 in the notes):

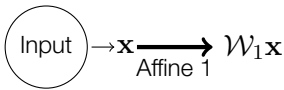
$$\left(\text{Affine = Matrix multiplicaiton + translation} \implies \mathcal{W}\mathbf{v} = W\mathbf{v} + \mathbf{w}\right)$$



Affine Mess We Find Ourselves in

Or, in terms of **Affine Operators** (Chapter 3.4 in the notes):

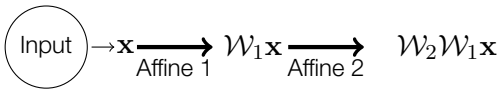
$$\left(\text{Affine = Matrix multiplicaiton + translation} \implies \mathcal{W}\mathbf{v} = W\mathbf{v} + \mathbf{w}\right)$$



Affine Mess We Find Ourselves in

Or, in terms of **Affine Operators** (Chapter 3.4 in the notes):

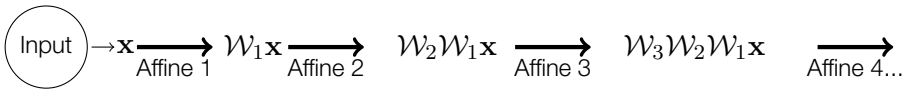
$$\left(\text{Affine = Matrix multiplicaiton + translation} \implies \mathcal{W}\mathbf{v} = W\mathbf{v} + \mathbf{w}\right)$$



Affine Mess We Find Ourselves in

Or, in terms of **Affine Operators** (Chapter 3.4 in the notes):

$$\left(\text{Affine = Matrix multiplicaiton + translation} \implies \mathcal{W}\mathbf{v} = W\mathbf{v} + \mathbf{w}\right)$$



Affine Mess We Find Ourselves in

But...affine operators obey:

$$\hat{\mathcal{W}}_2\hat{\mathcal{W}}_1 = \hat{\mathcal{V}} \tag{5}$$

(Affine + Affine = Affine)
Which means:

$$\hat{\mathcal{W}}_N\hat{\mathcal{W}}_{N-1}\hat{\mathcal{W}}_{N-2}\ldots\hat{\mathcal{W}}_1\mathbf{x}_{\text{input}} = \hat{\mathcal{V}}\mathbf{x}_{\text{input}} \tag{6}$$

Affine Mess We Find Ourselves in

But...affine operators obey:

$$\hat{\mathcal{W}}_2\hat{\mathcal{W}}_1 = \hat{\mathcal{V}} \tag{5}$$

(Affine + Affine = Affine)
Which means:

$$\hat{\mathcal{W}}_N\hat{\mathcal{W}}_{N-1}\hat{\mathcal{W}}_{N-2}\ldots\hat{\mathcal{W}}_1\mathbf{x}_{\text{input}} = \hat{\mathcal{V}}\mathbf{x}_{\text{input}} \tag{6}$$

If you do the maths:

$$\hat{\mathcal{V}}\mathbf{x} = \mathbf{v}\cdot\mathbf{x} + \mathbf{b} \tag{7}$$

wellcome sanger institute

Affine Mess We Find Ourselves in

But...affine operators obey:

$$\hat{\mathcal{W}}_2\hat{\mathcal{W}}_1 = \hat{\mathcal{V}} \tag{5}$$

(Affine + Affine = Affine)
Which means:

$$\hat{\mathcal{W}}_N\hat{\mathcal{W}}_{N-1}\hat{\mathcal{W}}_{N-2}\ldots\hat{\mathcal{W}}_1\mathbf{x}_{\text{input}} = \hat{\mathcal{V}}\mathbf{x}_{\text{input}} \tag{6}$$

If you do the maths:

$$\hat{\mathcal{V}}\mathbf{x} = \mathbf{v}\cdot\mathbf{x} + \mathbf{b} \tag{7}$$

Which is just....a single perceptron layer

Affine Mess We Find Ourselves in

If all you do is multiply by matrices & add vectors, you will *never* be doing anything other than an (extremely wasteful) perceptron algorithm

NonLinearity to the Rescue!

Need to break this affine relationship: enter **activation functions**.

NonLinearity to the Rescue!

Let:

$$\mathbf{x}_n = \text{activate}\left(W\mathbf{x}_{n-1} + \mathbf{b}\right) \tag{8}$$

NonLinearity to the Rescue!

Let:

$$\mathbf{x}_n = \text{activate}\left(W\mathbf{x}_{n-1} + \mathbf{b}\right) \tag{8}$$

One of the simples ways to break linearity is to add a ***elementwise function***:

$$\begin{aligned} \mathbf{v} &= W\mathbf{x}_{n-1} + \mathbf{b} \\ [\mathbf{x}_n]_i &= \sigma(v_i) \end{aligned} \tag{9}$$

Elementwise Activation Functions

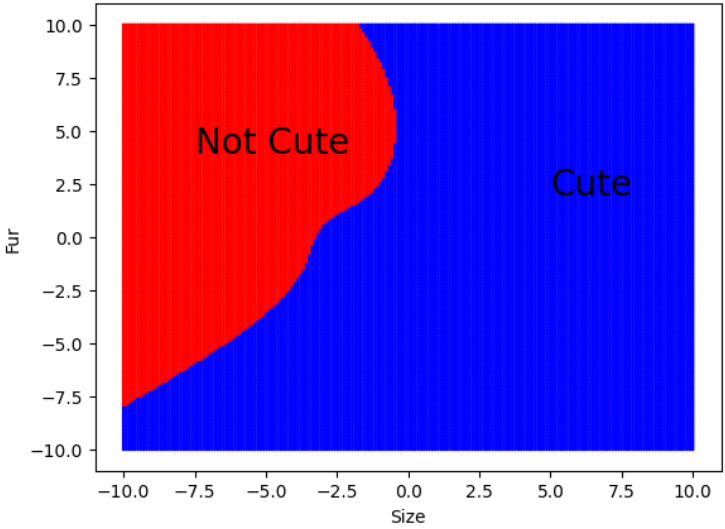
You can probably all name dozens of these functions in common use.

Sigmoid /Logit	$\sigma(x)$	$= \frac{1}{1+\exp(-x)}$
Rectified Linear Unit (ReLU)	$\text{Relu}(x)$	$= \begin{cases} x & x > 0 \\ 0 & \text{else} \end{cases}$
Leaky ReLU	$\text{Lelu}(x)$	$= \begin{cases} x & x > 0 \\ 0.01x & \text{else} \end{cases}$
tanh	$\tanh(x)$	$= \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$
Softplus	$\text{sft}(x)$	$= \log\left(1+\exp(-x)\right)$

*(Bonus Question: Can you name any commonly used functions which **don't** follow this pattern?)*

NonLinearity Ahoy!

By replacing my (randomly initialised) hidden layers with Sigmoid nodes, I get:



wellcome sanger institute