

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



Why bother?

In a world with dozens of pre-built ML tools....why bother studying the fundamentals?

Why bother?

In a world with dozens of pre-built ML tools....why bother studying the fundamentals?
In a word...

Why bother?

In a world with dozens of pre-built ML tools....why bother studying the fundamentals?
In a word...

FOLKLORE

ML Folklore

ML Folklore

- ▶ ADAM vs AdaGrad?

ML Folklore

- ▶ ADAM vs AdaGrad?
- ▶ Softplus vs ReLu vs Leaky ReLu vs Sigmoid?

ML Folklore

- ▶ ADAM vs AdaGrad?
- ▶ Softplus vs ReLu vs Leaky ReLu vs Sigmoid?
- ▶ Cross-Entropy vs Least Squares?

ML Folklore

- ▶ ADAM vs AdaGrad?
- ▶ Softplus vs ReLu vs Leaky ReLu vs Sigmoid?
- ▶ Cross-Entropy vs Least Squares?
- ▶ Validation set magic numbers

ML Folklore

- ▶ ADAM vs AdaGrad?
- ▶ Softplus vs ReLu vs Leaky ReLu vs Sigmoid?
- ▶ Cross-Entropy vs Least Squares?
- ▶ Validation set magic numbers
- ▶ (Explainability!)

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
- ▶ Convolutional Networks*

(* Time dependent!)

Today's Agenda

The aim for today is:

- ▶ Classic Perceptron
- ▶ Feedforward Networks
- ▶ Non-Linearity
- ▶ Optimisation & Backpropagation
- ▶ Convolutional Networks*

(* Time dependent!)

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

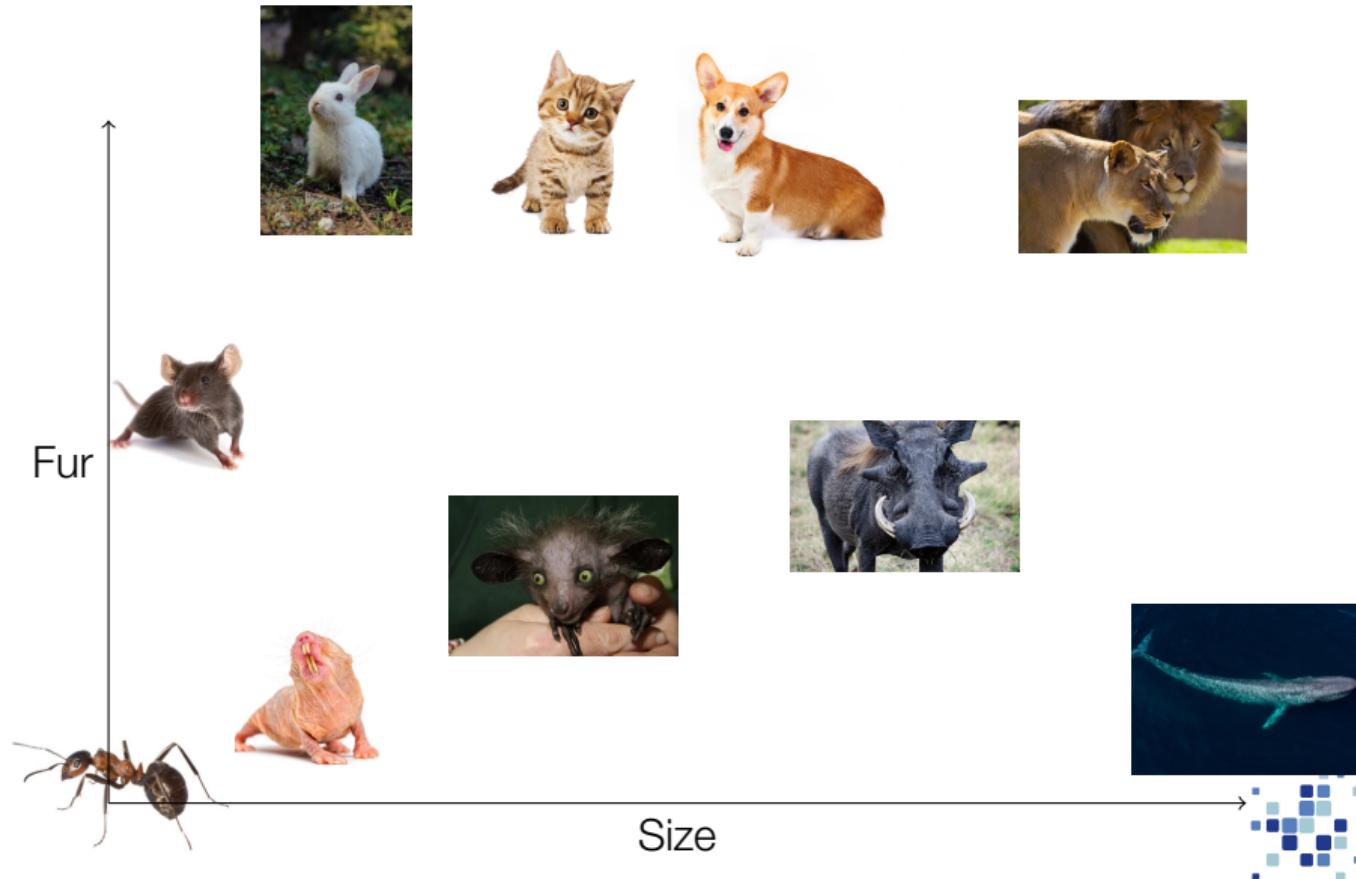
There will be equations.
You will need to know what they mean!

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

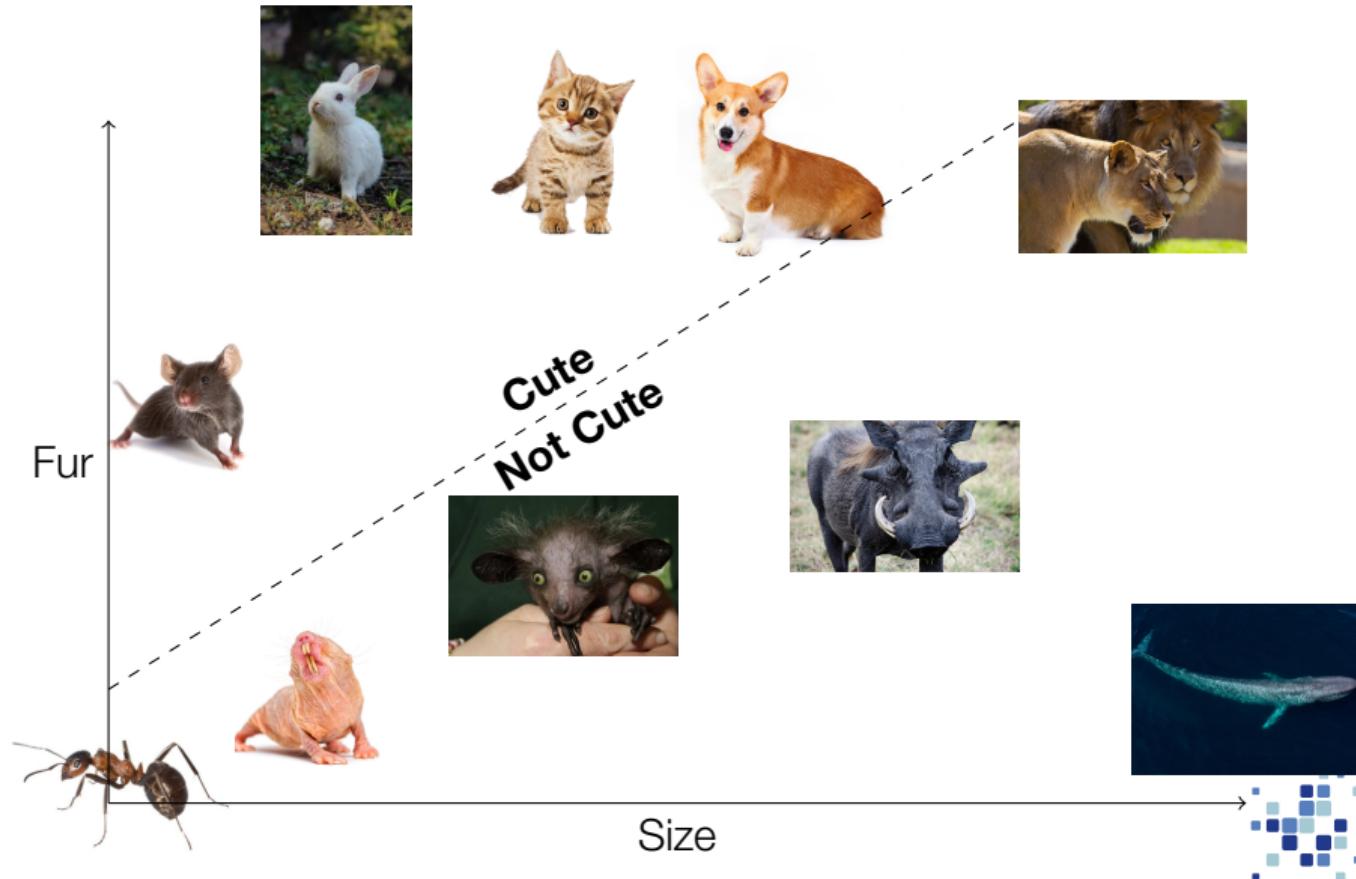
Part 1

The Perceptron

Basic Decision Making: Defining Cuteness



Basic Decision Making: Defining Cuteness



Splitting The Plane

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

Splitting The Plane

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

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

Splitting The Plane

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

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 $y = mx + c \rightarrow (m, c)$

Splitting The Plane

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

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 $y = mx + c \rightarrow (m, c)$

Question: Why then do we need $N + 1$ dimensions?

The Perceptron

We need 3 parameters to define our bi-directional line. The Perceptron classifier algorithm is:

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

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

The Perceptron

We need 3 parameters to define our bi-directional line. The Perceptron classifier algorithm is:

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

Where

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

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

The Perceptron

We need 3 parameters to define our bi-directional line. The Perceptron classifier algorithm is:

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

Where

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

\mathbf{w} our the **weights**

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