

# Artificial Intelligence: Past, Present, and Future

Radford Neal

radfordneal@gmail.com

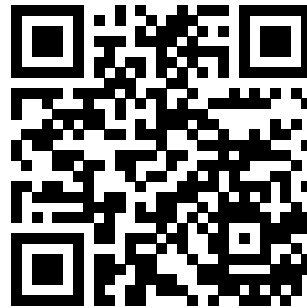
Lecture 1: The pace of AI progress

**Lecture 2: Computation and intelligence, artificial and natural**

Lecture 3: How modern AI works

Lecture 4: Can an AI think, feel, be conscious? How can we know?

Lecture 5: Benefits and dangers of AI, today and tomorrow



<https://glizen.com/radfordneal/ai-lectures/>

Originally presented at The Abelard School, Toronto, Spring 2025

# Questions About Brains and Computers

## **What is computation?**

In the 1930s, Alan Turing, Alonzo Church, and others tried to clarify what is meant by “mechanical computation”. Their ideas turned out to all be equivalent.

The abstract *Turing machine* is the best-known model of what can be computed, and coincides with what current computers can compute, given unlimited memory.

## **How does the brain work (so far as is known)?**

The brain contains a huge number of *neurons*, each of which has connections to many other neurons. These connections seem to play a big part in how the brain works.

But there are also other cells in the brain. And maybe complicated things go on within neurons. . .

## **How similar are brains and computers?**

Can brains compute things that computers can't?

Can we build a computer that behaves intelligently by mimicking how the brain works?

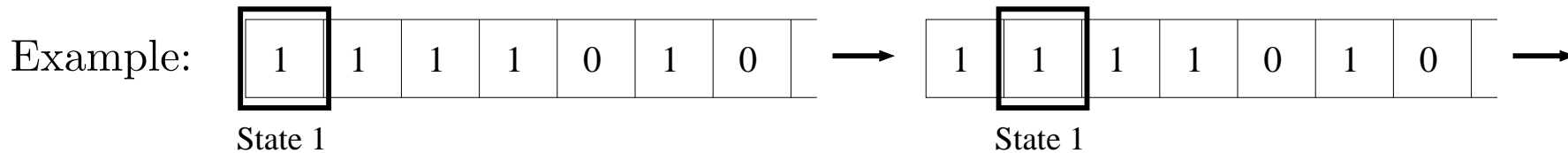
# Turing Machines

Turing envisioned a machine that operates in successive time steps using a *tape* that consists of *cells*, each containing some *symbol*. At each time, the machine is in some *state*.

The machine sees a single cell. Each step, based on the symbol it sees and the state, it will:

- 1) Write some symbol in the cell it sees.
- 2) Move to the cell to the left or right.
- 3) Enter a specified state.
- 4) Halt if the new state is “Halt”, otherwise go back to (1).

The number of symbols and number of states is fixed and finite. But the tape has an unlimited number of cells (so unlimited memory).



	State	Symbol	New Symbol	New State	Movement
Operation table:	1	1	1	1	Right
	1	0	0	2	Left
	2	1	0	Halt	—
	2	0	0	Halt	—

What does it do?

# What Can a Turing Machine Compute?

All actual computers can be simulated by a Turing machine (i.e., there's some set of states and a table of operations that do this). So Turing machines can compute everything actual computers can (also, vice versa, given unlimited memory).

But not everything can be computed by a Turing machine! (Or by a Python function.)

**The Halting Problem:** Suppose `will_halt(f, a)` returns `True` if function `f` halts when given argument `a`, and `False` if it doesn't (i.e., goes into endless loop).

Consider this function:

```
def puzzle(f):
    if will_halt(f, f):
        while True: pass # go into endless loop, never finishing
    else:
        return # halt
```

Will `puzzle(puzzle)` halt? Either answer is contradictory.

So no such `will_halt` function can exist. Whether a program will halt on given input is not (always) computable!

# How Efficiently Can a Turing Machine Compute Things?

Computer scientists often ask whether something can be computed in *polynomial time* — for all possible inputs, when the input is of length  $n$ , the time taken is less than  $cn^p$ , for some  $c$  and  $p$ .

The set of such problems is called  $P$ . Contrast this with problems that sometimes take exponential time, growing as  $ce^{an}$ .

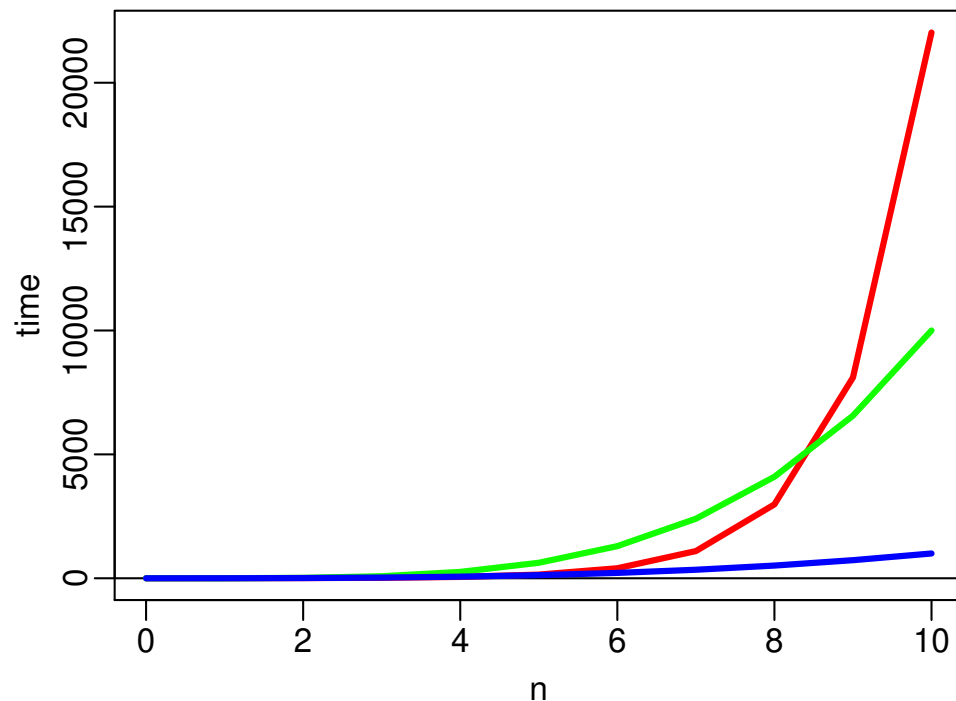
$P$  is the same for Turing machines and for all actual computers. But this hides some big differences —  $n^4$  is a lot slower than  $n^3$ . But still, a problem in  $P$  is often seen as being “efficiently computable”.

How time grows with the length of input as

$e^n$  (red)

$n^4$  (green)

$n^3$  (blue)



# The Church-Turing Thesis (and Extensions)

- The *Church-Turing thesis* states that Turing machines capture the idea of what's “mechanically computable”. Widely accepted; could be considered a definition.
- The *physical* Church-Turing thesis states that no physically-possible computer can compute things that a Turing machine can't. Widely but not universally accepted — some people think there could be uncomputable physical processes.

Note: Quantum computers don't violate this thesis.

- The *computational complexity* Church-Turing thesis states that Turing machines can compute everything more-or-less as fast as any physically-possible computer (up to a polynomial factor).

May or may not be true! It is not known whether quantum computers invalidate this.

# Digital Versus Analog Computers

Turing machines and the computers you know of are *digital* computers.

They operate on symbols, in precise ways. Numbers are represented symbolically — e.g., one-and-a-half by ‘1’, then ‘.’, then ‘5’ (or a similar binary scheme).

The same input always produces exactly the same output (with some exceptions).

The same program run on a different computer does exactly the same thing.

*Analog* computers have also been built. They represent numbers by physical quantities, and do numerical operations by some ‘analog’ of the operation.

**Silly example:** Represent a number by an amount of water in a glass. Add two numbers by pouring water from one glass into the other.

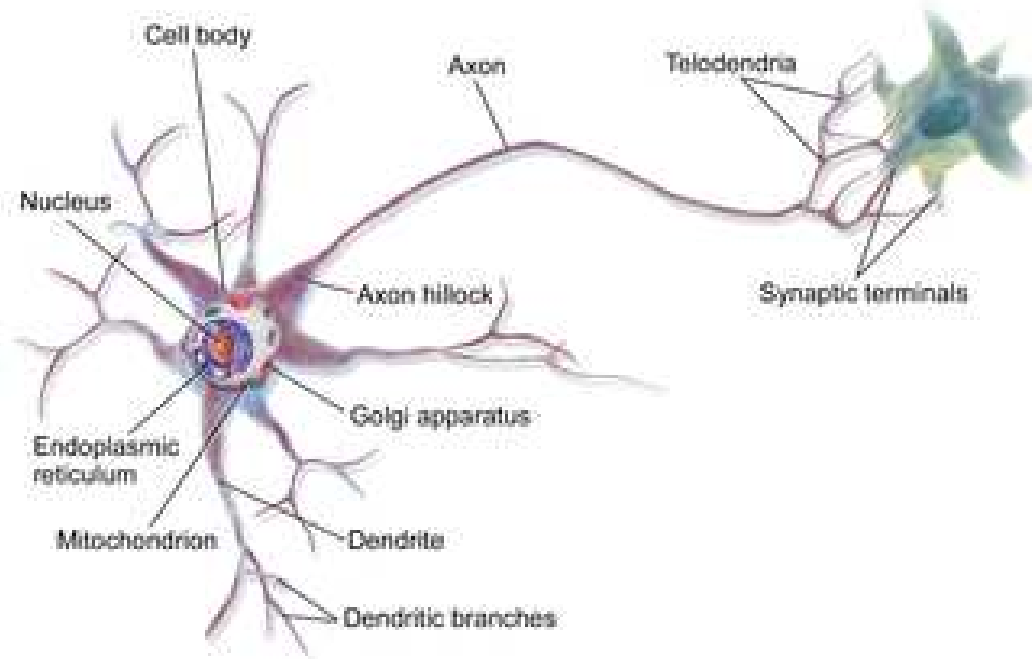
The output of an analog computer is not precisely determined by the input, and may be slightly different on a different analog computer (even of the same design).

Analog computers pretty much lost out to digital computers by the 1960s.

But perhaps an analog computer might be able to compute things that a digital computer can't? We're back to the physical Church-Turing thesis.

# Computation in the Brain — Neurons

Roughly half of the brain consists of *neurons*, which are believed to be the primary computational elements. There are about 100 billion ( $10^{11}$ ) neurons in the human brain.



Neurons are specialized cells, with

- a nucleus, mitochondria, etc. like other cells.
- a set of branching *dendrites*, that receive inputs.
- a long output *axon*, that branches at the end, forming
- *synapses* with dendrites on other neurons.

Input to a neuron via synapses on its dendrites affects its electrical polarization. When this crosses a threshold, an *action potential* (a “spike”) travels down the axon to the output synapses (at a speed of about 1 to 100 m/s, faster for “myelinated” axons).

Many variations exist, including sensory neurons, with no dendrites, and motor neurons, whose synapses are on muscle cells.



# Synapses Between Neurons

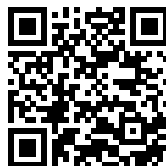
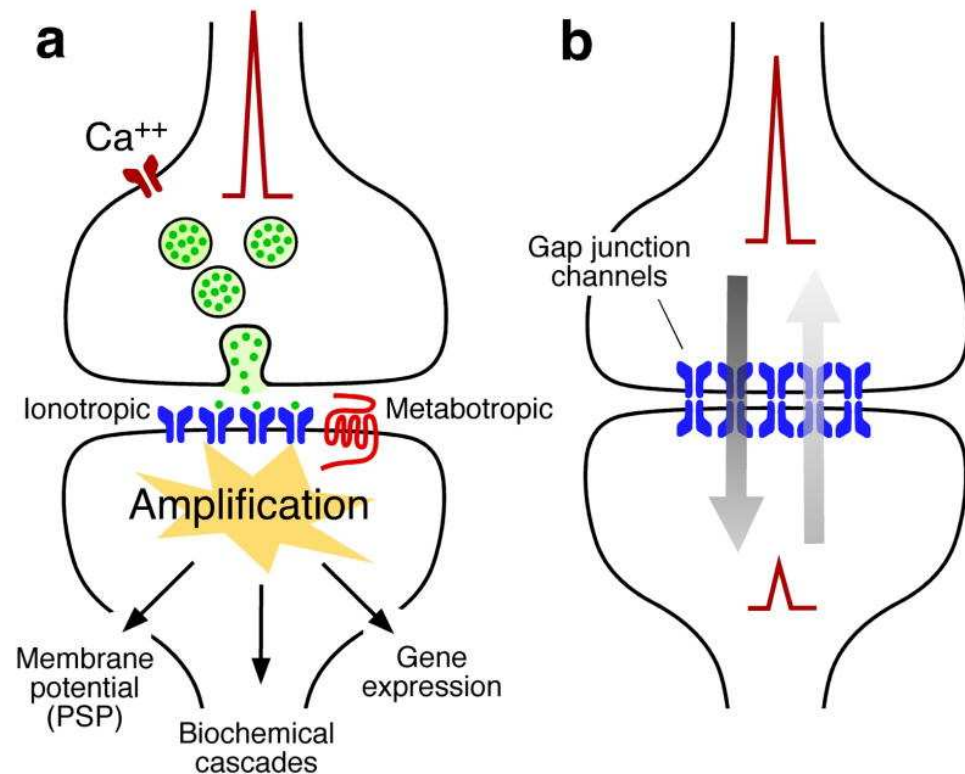
A synapse from an axon branch onto a dendrite branch can be *chemical* ((a) at the right), or *electrical* ((b) at the right).

Electrical synapses respond faster, and are bidirectional — passing attenuated signals back and forth.

Chemical synapses are more complex, and predominate in humans.

A single neuron may have up to several thousand synapses. Humans have about 1 quadrillion ( $10^{15}$ ) synapses.

When an action potential arrives at a chemical synapse, vesicles containing a *neurotransmitter* are released, diffuse across the gap to the dendrite side, and are sensed by receptors. Some receptors excite the neuron (an action potential becomes more likely), others inhibit the neuron.



# Brain Development and Adaptation

The  $3 \times 10^9$  bases in the human genome cannot possibly specify the  $10^{15}$  synaptic connections in the brain. But the overall organization may be genetically specified — e.g, how the cerebral cortex consists of millions of repeated structures called *cortical columns*.

The detailed pattern of connections must arise from some program of development to adulthood, combined with learning from experience.

**Generation and selection** is one mechanism:

- Neurons are created in the embryo, but some are then eliminated during fetal development — those that don't develop correctly, or fail to connect to others.
- The number of synapses grows through early childhood, but then is cut about in half by adulthood — again, based on which seem useful.

New neurons are created throughout life, but at very much lower levels than during development.

**Neural plasticity** occurs throughout life, both with synapses being created and destroyed, and with the strengths of synapses changing, based on experience.

# Unknowns and Speculations About the Brain

The human brain is far from being understood.

Indeed, biologists are still working on understanding the 302 neurons in the 1mm long nematode worm *Caenorhabditis elegans* (<https://github.com/Jessie940611/BAAIWorm>).

Some unknowns and speculations:

- How do connections between neurons encode information? Is the rate at which a neuron is spiking all that matters? Or is the exact timing of each spike important?
- Are connections between neurons all that's going on? Or might important information or capabilities be in the body of a neuron, or elsewhere?  
If so, the brain may be even more complex than it seems to us now.
- Could the brain possibly be a quantum computer?  
Unlikely, but if so, it could be much more computationally powerful than it seems.
- Could the brain be computing something that isn't computable by a Turing machine?  
Seems very unlikely, but believed by some people (e.g., Roger Penrose).

# Computational Power of Brains Versus Computers

High-end personal computer with GPUs (1500 Watts power):

- Memory (Gigabyte =  $10^9$  bytes, Terabyte =  $10^{12}$  bytes):
  - 100 Terabytes of hard disk space (0.1 Gigabytes/second bandwidth)
  - 1 Terabyte of system RAM (100 Gigabytes/second bandwidth)
  - 0.1 Terabytes of RAM in the GPUs (1000 Gigabytes/second bandwidth)
- Speed of computation:
  - 100 trillion operations per second (in GPUs)

And the brain (20 Watts)? Hard to say, since it has both analog and digital aspects.

Here are some guessed equivalences (might be much higher or lower):

- Memory:
  - Long-term memory in  $10^{15}$  synapses, each one byte? 1000 Terabytes
  - Short-term memory in activations of  $10^{11}$  neurons, each one byte? 0.1 Terabytes
- Speed of computation:
  - Action potentials produced by  $10^{11}$  neurons, on average 10 times per second?  
based on summing about 1000 connections? 1000 trillion operations per second

# Implications of Compute Power

## **Compute overhang?**

Is present-day computing power sufficient for human level (or even super-human level) intelligence? Is all that is needed a more clever algorithm, or more data to train on?

Or does human level intelligence require faster computation / more memory / a different organization (maybe computation and memory merged)?

## **Brain emulation?**

How feasible is it to emulate the operation of a human brain on a computer?

Requires sufficient memory and compute power.

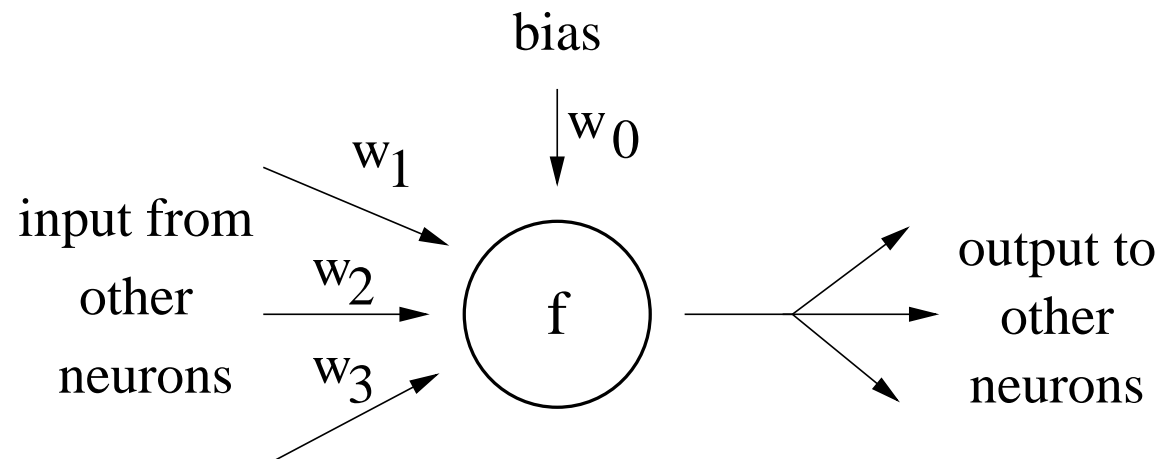
Also needs some way of “scanning” the state of a brain, which seems extremely difficult. But some people think it’s possible (e.g., Robin Hanson).

# Artificial Neural Networks

AI researchers back to the 1950s have created artificial neurons, inspired by simplified models of real neurons.

Typically, the “neuron” receives a weighted sum of input from other neurons (plus a “bias”), applies some *activation function*,  $f$ , to this sum, and distributes this value to other neurons.

Sometimes, the activation function gives a probability for the value of the neuron to be 1.



$$\text{output value} = f\left(w_0 + \sum_i w_i (\textit{i}'\text{th input value})\right)$$

Analogy to real neurons: Values are average spiking rates, the  $w_i$  are synaptic strengths.

# Learning in Natural and Artificial Neural Networks

How does the brain learn?

*Hebbian* learning is an idea from over 75 years ago — make a synapse stronger when *both* the pre-synaptic neuron and the post-synaptic neuron fire. This is a “local” rule, involving just the two connected neurons.

Currently, learning in artificial neural networks is often non-local, with *backpropagation* the dominant method, even though this is implausible as a mechanism in the brain.

Here’s the idea of backpropagation learning in a network of “neurons”:

- Define what the network should do — e.g., predict the class of an image.
- Define an “error” measure for how much it failed to do this, for a particular case.
- For every episode of learning (e.g., every training case):
  - For each “weight” in the network (synaptic strength), find out how much a small change in this weight would increase or decrease the error.
  - Increase each weight by a small amount proportional to how much such an increase would decrease the error (or decrease the weight if this is negative).

# A Simple Neural Network (Logistic Regression)

Suppose we want to predict a binary (0/1) outcome, based on some inputs. A simple neural network can do this, one equivalent to the statistical method of *logistic regression*.

If there are three inputs,  $x_1, x_2, x_3$ , we can predict the outcome,  $y$ , using the conditional probability that  $y$  is 1,  $P(y = 1 | x_1, x_2, x_3)$ . The probability that  $y$  is 0 is one minus this.

We model this probability using a weighted sum of inputs (plus a “bias”):

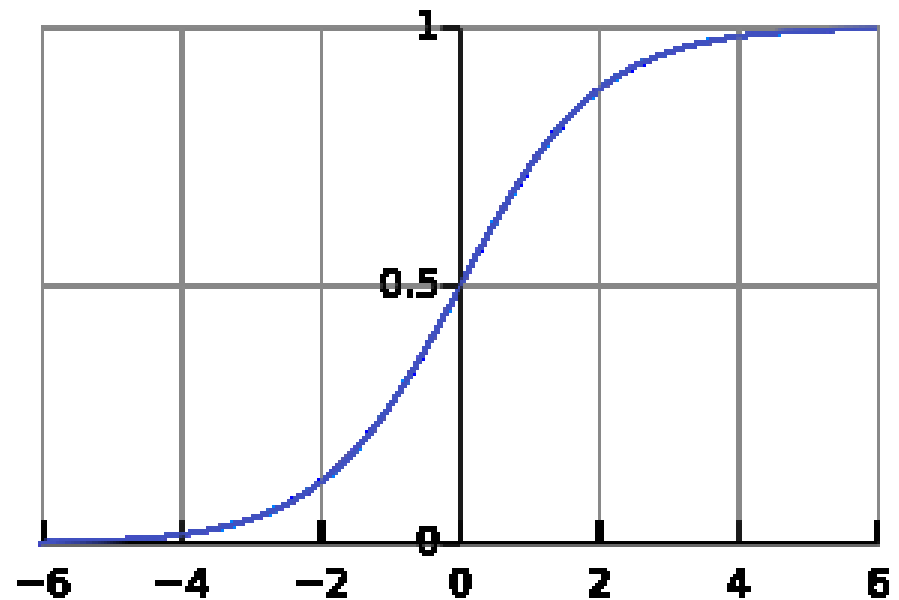
$$P(y = 1 | x_1, x_2, x_3) = \text{Logistic}(w_0 + w_1x_1 + w_2x_2 + w_3x_3)$$

Here, the logistic function is defined as

$$\text{Logistic}(z) = \frac{1}{1 + e^{-z}}$$

It's shown to the right.

For big positive values of  $z$ , the probability of  $y = 1$  is close to one. Big negative values make the probability of  $y = 1$  close to zero.



# Logistic Regression for the Titanic Data

As an example, we can try to predict who on the Titanic survived its sinking.

I have data for 2201 people on board. For each person, four binary variables represent

- Whether they were a crew member (rather than a passenger).
- Whether they were an adult (rather than a child).
- Whether they were female (rather than male).
- Whether they survived (rather than died).

The logistic regression model with parameters  $w_0, w_1, w_2, w_3$  is

$$\text{Probability of survival} = \text{logistic}\left(w_0 + w_1 \text{Crew} + w_2 \text{Adult} + w_3 \text{Female}\right)$$

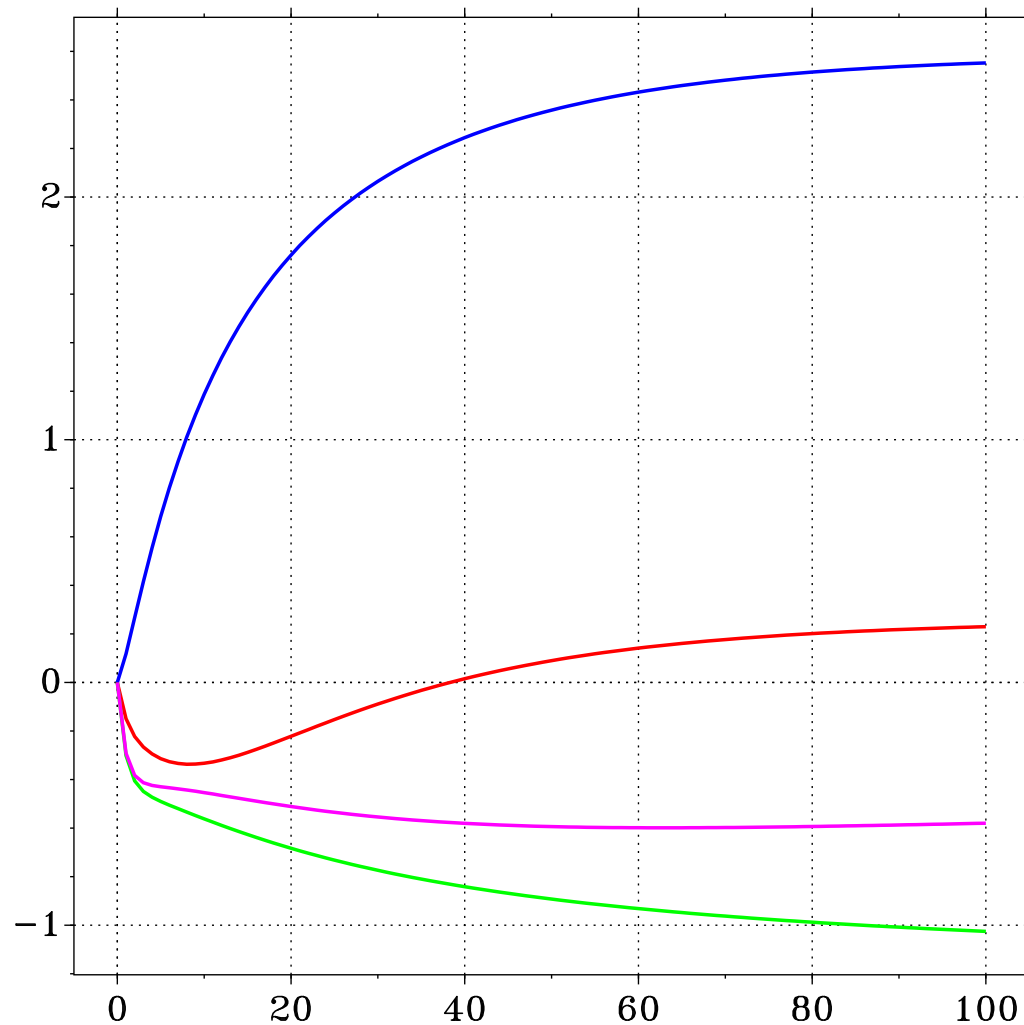
Here,  $\text{Crew} = 1$  if the person was a crew member, and  $\text{Crew} = 0$  if they were a passenger, and similarly for  $\text{Adult}$  and  $\text{Female}$ .

I randomly selected 1101 people to use for learning these parameters (the training set), leaving 1100 people to use for testing how well the model predicts.



# How the Learning Progresses

Here's how the network weights change as learning proceeds through 100 "epochs" (in which each training case is looked at once):

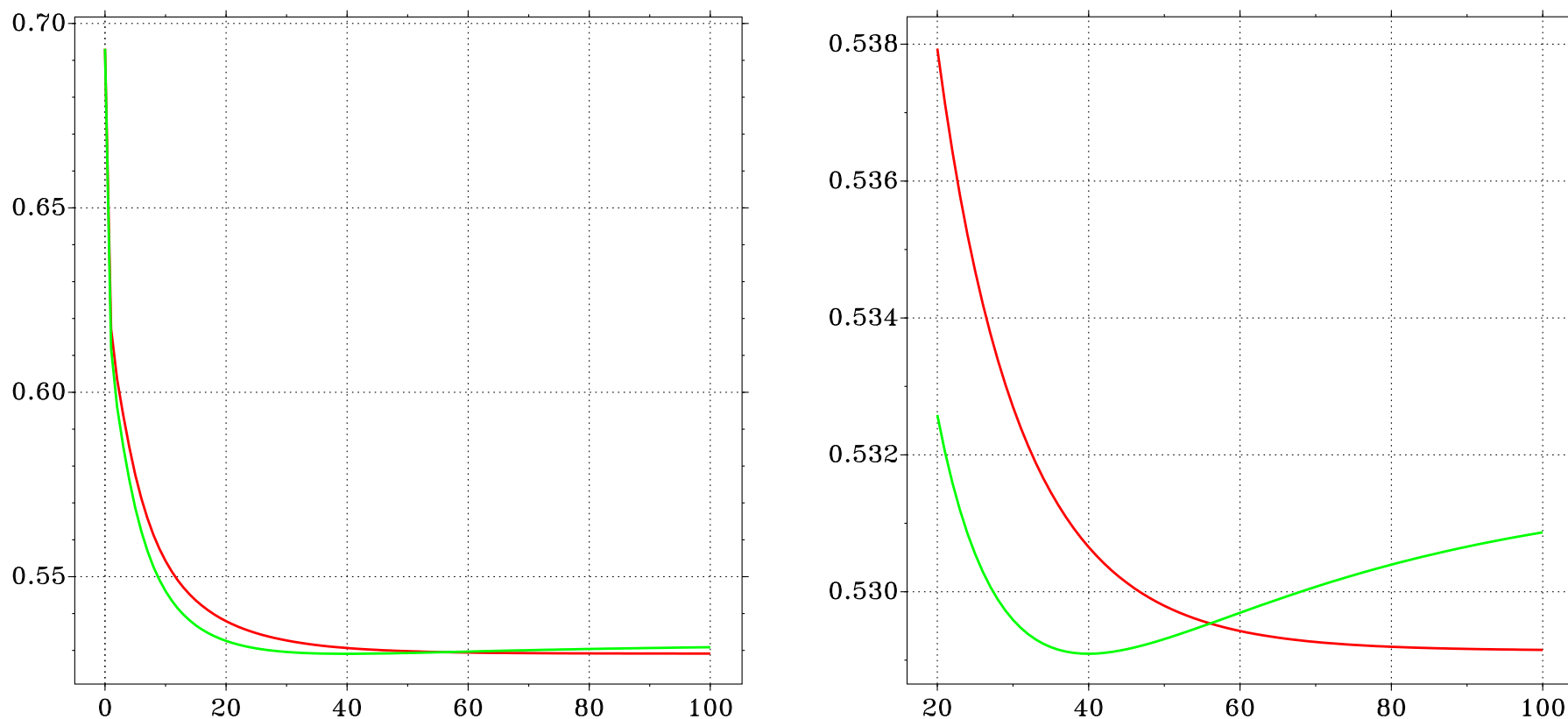


**red:**  $w_1$  (crew?)  
**green:**  $w_2$  (adult?)  
**blue:**  $w_3$  (female?)  
**magenta:**  $w_0$  (output bias)

# Training and Test Performance

With the network weights found after 100 epochs of training, the network has an error rate of 22.4% when classifying test cases. (Versus 31.8% error always guessing they died.)

Here's the average negative log probability for training cases (red) and test cases (green), as training progresses for 100 epochs. Right plot shows only from epoch 20 on.



There is some “overfitting” — better on test cases after only 40 epochs than after all 100.