Explanatory Notes for 6.390

Shaunticlair Ruiz (Current TA)

Fall 2022

Networks

Now, we have fully developed the individual **neuron**.

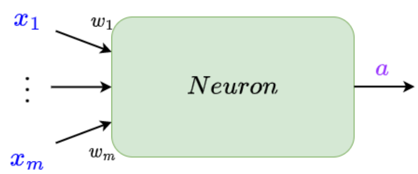
We can even do **gradient descent** on it: just like when we were doing LLCs, we can use the **chain rule**.

We'll get into this more, later in the chapter.

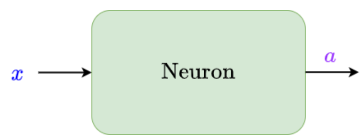
So, we return to the idea from the beginning of this chapter: combining multiple neurons into a **network**.

Abstraction

For this next section, we'll **simplify** the above diagram to this:



In fact, for more **simplicity**, we'll draw **one** arrow to represent the whole vector x. However, nothing about the **actual** math has changed.



This is also called **abstraction** - we need it a lot in this chapter.

Definition 1

Abstraction is a way to view your system more **broadly**: removing excess details, to make it **easier** to work with.

Abstraction takes a **complicated** system, and focuses on only the **important** details. Everything else is **excluded** from the model.

Often, this **simplified** view boils a system down to its the **inputs** and **outputs**: the "interface".

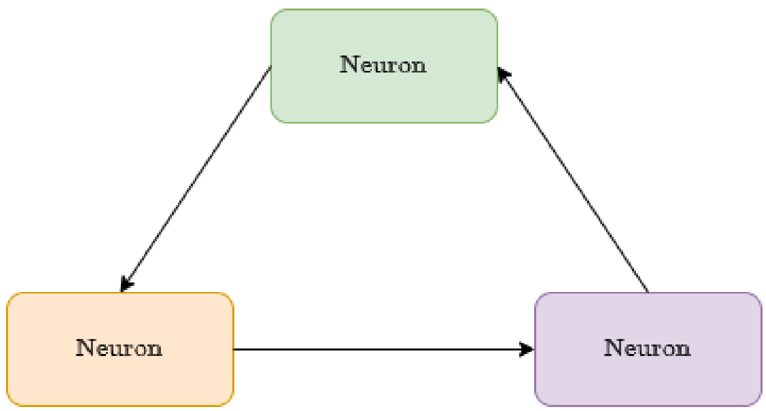
Example: Rather than thinking about all of the **mechanics** of how a car works, you might **abstract** it down to the pedals, the steering wheel, and how that causes the car to move.

Some limitations: acyclic networks

We won't allow for just **any** kind of network: we can create ones that might be unhelpful, or just very **difficult** to **analyze**.

For now, we can get interesting and **useful** behavior while keeping it **systematic**. We'll define this "system" later.

We'll assume our networks are **acyclic**: they do not create closed **loops**, where something can affects it own input.



This is a cyclic network: this is messy and we won't worry about this for now.

This means information only **flows** in one direction, "forward": it never flows "backwards".

Concept 2

For simple **neural networks**, we assume that they are **acyclic**: there are no **cycles**, or loops.

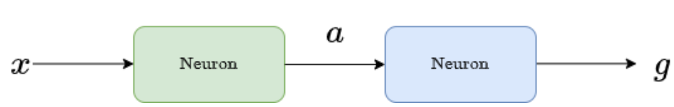
This means that **no neuron** has an output that affects its **input**, directly or indirectly.

We call these **feed-forward** networks.

We'll show how to build up the rest of what we need.

How to build networks

Suppose we have two neuron in **series**, our **simplest** arrangement:

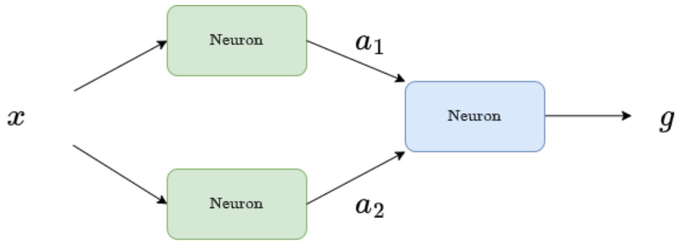


Our first neuron takes in a whole **vector** of values, $x = [x_1, x_2, ..., x_m]^T$. But, it only **outputs** a single value, a.

Remember that while we only see one arrow from x, each data point x_i is included.

That means the second neuron only receives **one** value, but it's capable of handling a full **vector**. We can add more values!

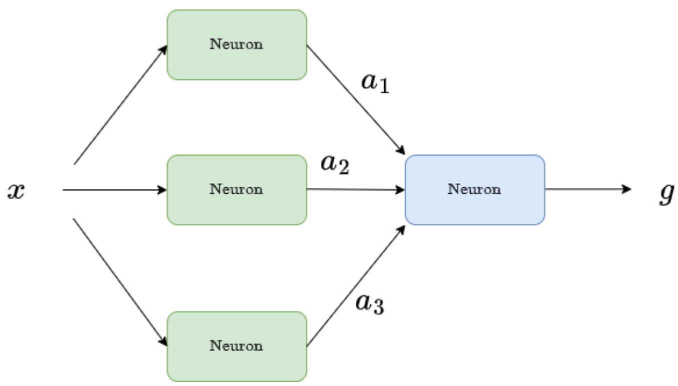
Let's add **another** neuron.



Our rightmost neuron now has **2 inputs**, which can be stored in a vector,

$$A = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \tag{1}$$

We could increase the **length** of this vector by adding more **neurons**.



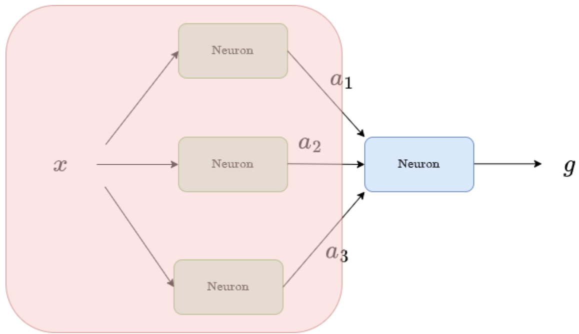
$$A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{2}$$

For our **rightmost** neuron, this is effectively the **same** as x: an **input vector**.

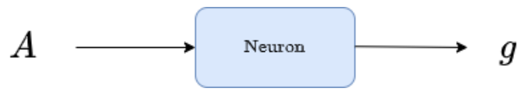
Layers

This gives us an idea for how to **build** our network: using multiple neurons in **parallel**, we can output a new vector A!

This is useful, because it means we can **simplify**: from the rightmost neuron's perspective, it just sees that **vector** as an input.



We can take this entire layer...



And just reduce it down to the vector A.

Because it's so useful, we'll give this set of neurons a name: a **layer**.

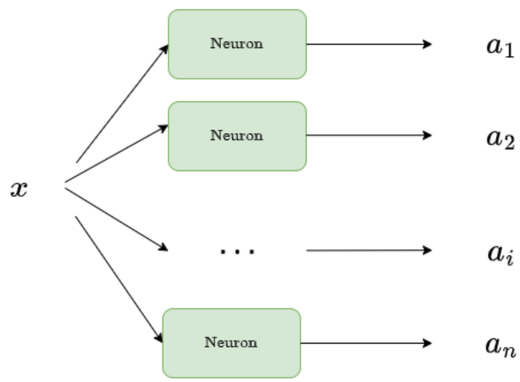
Definition 3

A **layer** is a set of **neurons** that are in "parallel":

- They all have **inputs** from the same **previous layer**
  - This **previous layer** could also be the **original input** x.
- They all have **outputs** to the same **next layer**
  - This **next layer** could also be the **final output** of the neural network.
- And none of these neurons are directly **connected** to each other.

This **layering** structure allows us to simplify our **analysis**: anything that comes after the layer only has to work with a **single vector**.

A layer in general might looks like this:



A general layer in a neural network.

The Basic Structure of a Neural Network

We could pick many structures for neural networks, but for simplicity, this will define our **template** for this chapter.

Definition 4

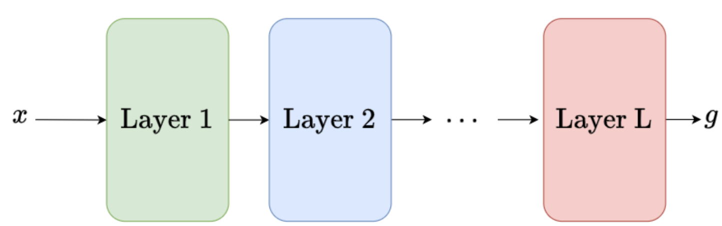
We structure our **neural networks** as a series of **layers**, where each layer is the **input** to the next layer.

This means that **layers** are a basic unit of a neural network, one level above a **neuron**.

In short, we have:

- A **neuron**, made of a linear and an activation component
- A **layer**, made of many neurons in parallel
- A **neural** network, made of many layers in series

Our goal is some kind of structure that looks something like this:



A neural network.

We now have a high-level view of our entire neural network, so now we dig into the details of a single layer.