Explanatory Notes for 6.390

Shaunticlair Ruiz (Current TA)

Fall 2022

Basic Element

Now, we have idea of what neural networks **are**. But, we have yet to handle the **details**:

- What **is** a neuron?
- How do we "systematically" **combine** our neurons?
- How do we **train** this, like we would a **simple** model?

We'll handle all of these steps and more - the above description was just to give a **high-level** view of what we want to **accomplish**.

Now, we go down to the **bottom** level, and think about just **one neuron**: what does it look like, and how does it work?

First, some terminology:

> **Notation 1**
>
> **Neurons** are also sometimes called **units** or **nodes**.
>
> They are mostly **equivalent** names. They just reflect different **perspectives**.

What's in a neuron: The Linear Component

As we mentioned before, our goal is to combine **simple** units into a **bigger** one. So, we want a model that's **simple**.

Well, let's start with what we've done before: we've worked with the **linear** model

$$h(x) = \theta^T x + \theta_0 \tag{1}$$

This model has lots of nice properties:

- It limits itself to **addition** and **multiplication** (easy to compute)
- **Linearity** lets us prove some mathematical things, and use vector/**matrix** math
- The dot product between θ and x has a nice **geometric** interpretation.

This will make up the **first** part of our model.

> **Concept 2**
>
> Our **neuron** contains a **linear** function as its **first** component.

Weights and Biases

But, there's one minor **change**: before, we used θ because it represented our **hypothesis**.

But, every neuron is going to have its own **values** for its **linear** model:

$$\overbrace{f_1(x)}^{\text{Neuron 1}} = Ax + B \qquad\qquad \overbrace{f_1(x)}^{\text{Neuron 2}} = Cx + D \tag{2}$$

It wouldn't make much **sense** to call both A and C by the name θ.

We could use some clever **notation**, but why treat them as **hypotheses**? They are each only a **part** of our hypothesis Θ.

So, instead of thinking of each as a "hypothesis", let's switch perspectives.

Each value θ_k **scales** how much x_k affects the **output**: if we're doing

$$g(x) = 100x_1 + 2x_2 \tag{3}$$

Then, changing x_1 will have a much **bigger** effect on $g(x)$. Another way to say this is it **weighs** more heavily: it matters **more**.

Because of that, we call the number we scale x_1 by a **weight**.

Notation 3

A **weight** w_k tells you how heavily a **variable** x_k **weighs** into the output.

w_k is **equivalent** to θ_k: it's a **scalar** $w_k \in \mathbb{R}$.

$$\Big(\theta_1 x_1 + \theta_2 x_2\Big) \Longleftrightarrow \Big(w_1 x_1 + w_2 x_2\Big)$$

We can combine it into a vector $w \in \mathbb{R}^m$.

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} \qquad\qquad \theta^T x \Longleftrightarrow w^T x$$

Remember that $a \Longleftrightarrow b$ means a and b are equivalent!

What about our other term, θ_0? We call it an **offset**: it's the value we **shift** our linear model away from **origin**.

We'll use the same notation:

Notation 4

An **offset** w_0 tells you how far we **shift** $h(x)$ away from the origin.

w_0 is **equivalent** to θ_0: it's a **scalar** $w_0 \in \mathbb{R}$

$$\left((\theta^T x) + \theta_0\right) \Longleftrightarrow \left((w^T x) + w_0\right)$$

We also sometimes call this the **threshold** or the **bias**.

Alternate notation: we might call this variable b, for bias.

This gives us our linear model using our new notation:

Definition 5

The **linear component** for a neuron is given by

$$z(x) = w^T x + w_0$$

where $w \in \mathbb{R}^m$ and $w_0 \in \mathbb{R}$.

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

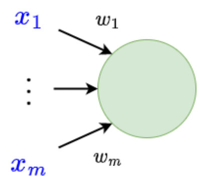
Linear Diagram

Now, we want to be able to depict our **linear** subunit. Let's do it piece-by-piece.

First, we have our vector $x = [x_1, x_2, ..., x_m]^T$:

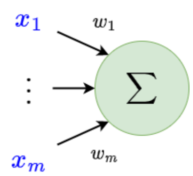
$$\begin{matrix} x_1 \\ \vdots \\ x_m \end{matrix}$$

Now, we want to **multiply** each term x_i by its corresponding **weight** w_i. We'll combine them into a **function**:



The circle represents our function.

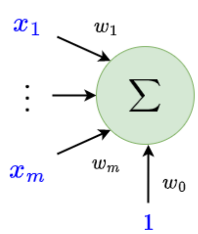
How are we combining them? Well, we're adding them together.



Note that we use the $\sum$ symbol, because we're **adding** after we **multiply**. In fact, we can write this as

$$w^T x = \sum_{i=1}^{m} w_i x_i \tag{4}$$

We'll include the bias term as well: remember that we can represent w_0 as $1 * w_0$ to match with the other terms.

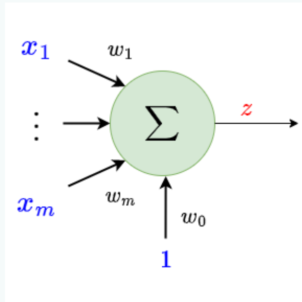


The blue "1" term is **multiplied** by w_0, just like how x_k gets multiplied by w_k.

We have our full function! All we need to do is include our output, z:

Notation 6

We can depict our linear function $z = w^T x + w_0$ as



Thus, z is a function of x:

$$z(x) = w^T x + w_0 \tag{5}$$

Which, in $\sum$ notation, we could write as

$$z(x) = \left(\sum_{i=1}^{m} w_i x_i \right) + w_0 \tag{6}$$

Adding nonlinearity

We'll continue building our neuron based on what we've done **before**. When doing linear regression, that linear unit was all we had.

But, once we do classification, we found that it was helpful to have a second, **non-linear** component: we used **sigmoid** $\sigma(u)$.

We might not necessarily want the **same** nonlinear function, so instead, we'll just generalize: we have *some* second component, which is allowed to be **nonlinear**.

We call this component our **activation** function. Why do we call it that? It comes from the historical **inspiration** of neurons in the brain.

Biological neurons only "fire" (give an output) above a certain threshold of **input**: that's when they **activate**.

Some activation functions reflect this, but they don't have to.

Definition 7

Our **neuron** contains a potentially **nonlinear** function f called an **activation function** as its **second** component.

We notate this as

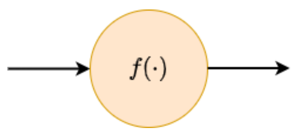
$$a = f(z) \tag{7}$$

Where z is the **output** of the **linear** component, and a is the **output** of the **activation** component.

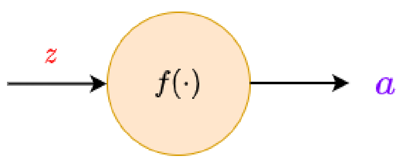
Note that z and a are **real numbers**: we have $f : \mathbb{R} \to \mathbb{R}$

Nonlinear Diagram

We'll depict a function f.



It takes in our **linear** output, z, and outputs our **neuron** output, a.



Note some vocabulary used for z:

Notation 8

z, the **output** of our **linear** function, is called the **pre-activation**.

This is because it is the result **before** we run the **activation** function.

And for a:

Notation 9

a, the **output** of our **activation** function, is called the **activation**.

Putting it together

So now, our neuron is complete.

Definition 10

Our **neuron** is made of

- A **linear** component that takes the neuron's input x, and applies a linear function

$$z = w^T x + w_0$$

  - The **pre-activation** z is the **output** of the **linear** function.
  - It is also the **input** of the **activation function** f.

- A (potentially nonlinear) **activation** component that takes the pre-activation z and applies an **activation function** f:

$$a = f(z)$$

  - The **activation** a is the **output** of this **activation function**.

When we **compose** them together, we get

$$a = f(z) = f\left(w^T x + w_0\right)$$

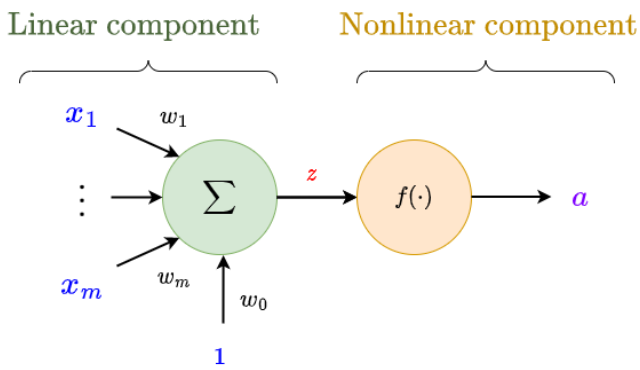
When we say "compose", we mean **function composition**: combining $f(x)$ and $g(x)$ into $f(g(x))$.

We can also use $\sum$ notation to get:

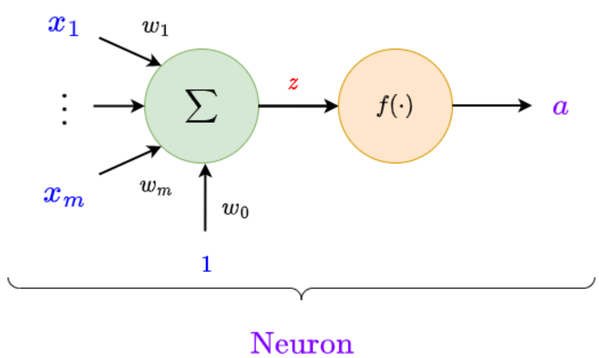
$$a = f(z) = f\left(\left(\sum_{i=1}^{m} w_i x_i\right) + w_0\right)$$

Neuron Diagram

Finally, we can **compose** our neuron into one big **diagram**:

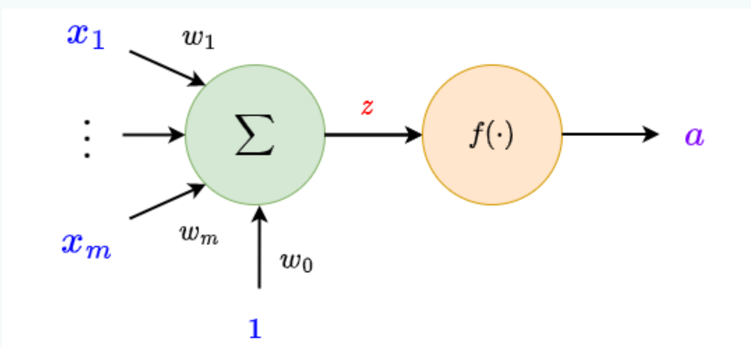


From here on out, we'll treat this as a **single** object:



Notation 11

We can depict our **neuron** $f(w^T x + w_0)$ as



- x is our **input** (neuron input, linear input)
- z is our **pre-activation** (linear output, activation input)
- a is our **activation** (neuron output, activation output)

This neuron will be the **basic unit** we work with for the rest of this **chapter** - it's one of the most **important** objects in all of machine learning.

Our Loss Function

One more detail: we will want to **train** these neurons. In order to be able to **measure** their performance, we'll need a **loss** function.

This **isn't** any different from usual: we just need a **function** of the form

$$\mathcal{L}(g, y) \tag{8}$$

In **regression**, we wrote our loss as

$$\mathcal{L}\Big(\; h(x;\Theta), \;\; y \;\Big)$$

The right term, $y^{(i)}$, is unchanged: we still need to compare against the **correct** answer.

The main change is we aren't using Θ notation: we'll **replace** it with (w, w_0)

$$\mathcal{L}\left(\; h\Big(x;(w, w_0)\Big), \;\; y \;\right)$$

And finally, we get the loss for multiple data points:

We skip doing $1/n$ averaging because we often use this for SGD: we plan to take small steps as we go, rather than adding up our steps all at once.

$$\sum_i \mathcal{L}\left(\; h\Big(x^{(i)};(w, w_0)\Big), \;\; y^{(i)} \;\right)$$

And with this, not only is our neuron **complete**, but we have everything we need to **work** with it.

> **Concept 12**
> For a **complete neuron**, we need to specify
>
> - Our **weights** and **offset**
> - Our **activation** function
> - Our **loss** function

From here, we could do **stochastic gradient descent** as we usually do, to **optimize** this neuron's **performance**.