

# lecture14a-neural-networks

February 8, 2022

## 1 Lecture 13: Neural Networks

### 1.0.1 Applied Machine Learning

Volodymyr Kuleshov, Jin Sun Cornell Tech

## 2 Part 1: An Artificial Neuron

In this lecture, we will learn about a new class of machine learning algorithms inspired by the brain. We will start by defining a few building blocks for these algorithms, and draw connections to neuroscience.

## 3 Review: Components of A Supervised Machine Learning Problem

At a high level, a supervised machine learning problem has the following structure:

$$\underbrace{\text{Training Dataset}}_{\text{Attributes + Features}} + \underbrace{\text{Learning Algorithm}}_{\text{Model Class + Objective + Optimizer}} \rightarrow \text{Predictive Model}$$

Where does the dataset come from?

## 4 Review: Binary Classification

In supervised learning, we fit a model of the form

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

that maps inputs  $x \in \mathcal{X}$  to targets  $y \in \mathcal{Y}$ .

In classification, the space of targets  $\mathcal{Y}$  is *discrete*. Classification is binary if  $\mathcal{Y} = \{0, 1\}$

Each value of  $y$  value is a *class* and we are interested in finding a hyperplane that separates the different classes.

## 5 Review: Logistic Regression

Logistic regression fits a model of the form

$$f(x) = \sigma(\theta^\top x) = \frac{1}{1 + \exp(-\theta^\top x)},$$

where

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

is known as the *sigmoid* or *logistic* function.

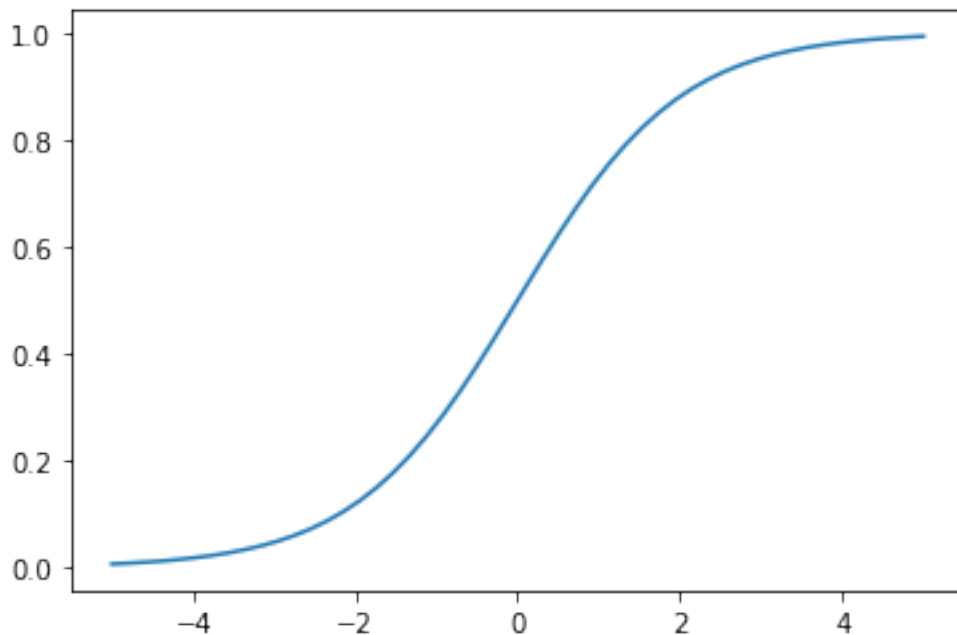
Here is how the logistic function looks like.

```
[1]: import numpy as np
      from matplotlib import pyplot as plt

      z = np.linspace(-5, 5)
      sigma = 1/(1+np.exp(-z))

      plt.plot(z, sigma)
```

```
[1]: [<matplotlib.lines.Line2D at 0x1172c9160>]
```



## 6 A Biological Neuron

In order to define an artificial neuron, let's look first a biological one.

TODO: PUT NEURON IMAGE FROM HERE: <https://cs231n.github.io/neural-networks-1/>

- Each neuron receives input signals from its dendrites
- It produces output signals along its axon, which connects to the dendrites of other neurons.

## 7 An Artificial Neuron: Example

We can imitate this machinery using an idealized artificial neuron. \* The neuron receives signals  $x_j$  at dendrites, which are modulated multiplicatively:  $w_j \cdot x_j$ . \* The body of the neuron sums the modulated inputs:  $\sum_{j=1}^d w_j \cdot x_j$ . \* These go into the activation function that produces an output.

TODO: PUT ARTIFICIAL NEURON IMAGE FROM HERE: <https://cs231n.github.io/neural-networks-1/>

## 8 An Artificial Neuron: Notation

More formally, we say that a neuron is a model  $f : \mathbb{R}^d \rightarrow [0, 1]$ , with the following components: \* Inputs  $x_1, x_2, \dots, x_d$ , denoted by a vector  $x$ . \* Weight vector  $w \in \mathbb{R}^d$  that modulates input  $x$  as  $w^\top x$ . \* An activation function  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$  that computes the output  $\sigma(w^\top x)$  of the neuron based on the sum of modulated features  $w^\top x$ .

## 9 Logistic Regression as an Artificial Neuron

Logistic regression is a model of the form

$$f(x) = \sigma(\theta^\top x) = \frac{1}{1 + \exp(-\theta^\top x)},$$

that can be interpreted as a neuron that uses the *sigmoid* as the activation function.

## 10 Perceptron

Another model of a neuron.

## 11 Example

Need to implement a small example. Can probably copy-paste implementation of LR from the LR slides.

## 12 Activation Functions

Let's list a few.

# Part 2: Artificial Neural Networks

Let's now see how we can connect neurons into networks that form complex models that further mimic the brain.

## 13 Review: Artificial Neuron

We say that a neuron is a model  $f : \mathbb{R}^d \rightarrow [0, 1]$ , with the following components:

- \* Inputs  $x_1, x_2, \dots, x_d$ , denoted by a vector  $x$ .
- \* Weight vector  $w \in \mathbb{R}^d$  that modulates input  $x$  as  $w^\top x$ .
- \* An activation function  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$  that computes the output  $\sigma(w^\top x)$  of the neuron based on the sum of modulated features  $w^\top x$ .

## 14 Review: Logistic Regression as Neuron

Logistic regression is a model of the form

$$f(x) = \sigma(\theta^\top x) = \frac{1}{1 + \exp(-\theta^\top x)},$$

that can be interpreted as a neuron that uses the *sigmoid* as the activation function.

## 15 Neural Networks: Intuition

A neural network is a directed graph in which a node is a neuron that takes as input the outputs of the neurons that are connected to it.

TODO: Add an image here. Maybe layer image from here: <https://cs231n.github.io/neural-networks-1/> (It probably needs some annotations)

Networks are typically organized in layers.

## 16 Neural Networks: Layers

A neural network layer is a model  $f : \mathbb{R}^d \rightarrow \mathbb{R}^p$  that applies  $p$  neurons in parallel to an input  $x$ .

$$f(x) = \begin{bmatrix} \sigma(w_1^\top x) \\ \sigma(w_2^\top x) \\ \vdots \\ \sigma(w_p^\top x) \end{bmatrix}.$$

where each  $w_k$  is the vector of weights for the  $k$ -th neuron. We refer to  $p$  as the *size* of the layer.

By combining the  $w_k$  into one matrix  $W$ , we can write in a more succinct vectorized form:

$$f(x) = \sigma(W \cdot x) = \begin{bmatrix} \sigma(w_1^\top x) \\ \sigma(w_2^\top x) \\ \vdots \\ \sigma(w_p^\top x) \end{bmatrix},$$

where  $\sigma(W \cdot x)_k = \sigma(w_k^\top x)$  and  $W_{kj} = (w_k)_j$ .

## 17 Neural Networks: Notation

A neural network is a model  $f : \mathbb{R} \rightarrow \mathbb{R}$  that consists of a composition of  $L$  neural network layers:

$$f(x) = f_L \circ f_{L-1} \circ \dots \circ f_1(x).$$

The final layer  $f_L$  has size one (assuming the neural net has one output); intermediary layers  $f_l$  can have any number of neurons.

The notation  $f \circ g(x)$  denotes the composition  $f(g(x))$  of functions

## 18 Example of a Neural Network

- Let's implement a small neural net in the same that we implemented logistic regression
- Then we just run it

## 19 Types of Neural Network Layers

There are many types of neural network layers that can exist. Here are a few: \* Output layer: normally has one neuron and special activation function that depends on the problem \* Input layer: normally, this is just the input vector  $x$ . \* Hidden layer: Any layer between input and output. \* Dense layer: A layer in which every input is connected to every neuron. \* Convolutional layer: A layer in which the operation  $w^\top x$  implements a mathematical [convolution](#). \* Anything else?

## 20 Neuroscience Angle

Anything we should say here?

# Part 3: Backpropagation

We have defined what is an artificial neural network.

Let's not see how we can train it.

## 21 Review: Neural Network Layers

A neural network layer is a model  $f : \mathbb{R}^d \rightarrow \mathbb{R}^p$  that applies  $p$  neurons in parallel to an input  $x$ .

$$f(x) = \sigma(W \cdot x) = \begin{bmatrix} \sigma(w_1^\top x) \\ \sigma(w_2^\top x) \\ \vdots \\ \sigma(w_p^\top x) \end{bmatrix},$$

where each  $w_k$  is the vector of weights for the  $k$ -th neuron and  $W_{kj} = (w_k)_j$ . We refer to  $p$  as the *size* of the layer.

## 22 Review: Neural Networks

A neural network is a model  $f : \mathbb{R} \rightarrow \mathbb{R}$  that consists of a composition of  $L$  neural network layers:

$$f(x) = f_L \circ f_{L-1} \circ \dots \circ f_1(x).$$

The final layer  $f_L$  has size one (assuming the neural net has one output); intermediary layers  $f_l$  can have any number of neurons.

The notation  $f \circ g(x)$  denotes the composition  $f(g(x))$  of functions

TODO: Add some kind of image from the previous part of the lecture

## 23 Review: The Gradient

The gradient  $\nabla_{\theta} f$  further extends the derivative to multivariate functions  $f : \mathbb{R}^d \rightarrow \mathbb{R}$ , and is defined at a point  $\theta_0$  as

$$\nabla_{\theta} f(\theta_0) = \begin{bmatrix} \frac{\partial f(\theta_0)}{\partial \theta_1} \\ \frac{\partial f(\theta_0)}{\partial \theta_2} \\ \vdots \\ \frac{\partial f(\theta_0)}{\partial \theta_d} \end{bmatrix}.$$

In other words, the  $j$ -th entry of the vector  $\nabla_{\theta} f(\theta_0)$  is the partial derivative  $\frac{\partial f(\theta_0)}{\partial \theta_j}$  of  $f$  with respect to the  $j$ -th component of  $\theta$ .

## 24 Review: Gradient Descent

If we want to optimize an objective  $J(\theta)$ , we start with an initial guess  $\theta_0$  for the parameters and repeat the following update until the function is no longer decreasing:

$$\theta_i := \theta_{i-1} - \alpha \cdot \nabla_{\theta} J(\theta_{i-1}).$$

As code, this method may look as follows:

```
theta, theta_prev = random_initialization()
while abs(J(theta) - J(theta_prev)) > conv_threshold:
    theta_prev = theta
    theta = theta_prev - step_size * gradient(theta_prev)
```

## 25 Backpropagation

How do we apply gradient descent to a neural network?

Explain backpropagation

## 26 Review: Chain Rule of Calculus

Probably will need to review this at some point.

## 27 Example

Let's implement backprop with the simple NN model we had earlier.

# Part 4: Stochastic Gradient Descent

In practice, neural networks are often trained on very large datasets.

This requires a modification to the gradient descent algorithm that we have seen earlier.

Volodymyr will create this section

[ ]: