What/Why/How of Neural Networks

Sean Grate

November 18, 2020

1 Introduction

There are basically two different kinds of problems: (1) *supervised* and (2) *unsupervised*.

1. Given a labeled dataset $\{(x_i, y_i) \,:\, x_i \in \mathbb{R}^n, y_i \in S\}$, find a function $f\colon \mathbb{R}^n \to S$ s.t. $f(x_i) \approx y_i$.
2. Given a dataset $\{x_i \,:\, x_i \in \mathbb{R}^n\}$, find f s.t. $f(x_i)$ preserves certain structures or properties of f.

Definition 1. If $S = \mathbb{R}^m$, then this is a *regression* problem. If $S = \{G_1, \dots, G_m\}$, then this is a *classification* problem.

Definition 2. Let $\{(x_i, y_i) \,:\, x_i \in \mathbb{R}^n, y_i \in S\}$ be a labeled dataset. The x_i are called *feature vectors* and their components are called *features*. The y_i are called *labels* or *targets*.

We will only consider *parametric* models. Given a parametric family of functions

$$\mathcal{F} = \{f_\theta\colon \mathbb{R}^n \to S \,:\, \theta \in \mathbb{R}^p \text{ is a vector parameter}\},$$

our goal is to find parameters θ s.t. $f_\theta(x_i) \approx y_i$.

2 Loss Functions

How do you measure the a model's ability to approximate a desired function?

Definition 3. Let $\{(x_i, y_i) \,:\, x_i \in \mathbb{R}^n, y_i \in S\}$ be a labeled dataset. A *loss function* (or *cost function*) is a function $\mathcal{L}\colon S \times S \to \mathbb{R}$ that measures the difference between a prediction and a label.

Example 1. Suppose the label set S is given as $S = \mathbb{R}^m$ and $\{\hat{y}_i, y_i\}_{i=1}^k \subset S \times S$ is a collection of prediction/label pairs. The *mean squared error* (MSE) or l_2 loss is

$$\text{MSE} = \frac{1}{k} \sum_{i=1}^{k} \|\hat{y}_i - y_i\|_2^2,$$

where

$$\|\hat{y}_i - y_i\|_2 = \sqrt{\sum_{j=1}^{m} \left|\hat{y}_i^{(j)} - y_i^{(j)}\right|^2}$$

with (j) indicating the jth component of the vector.

Example 2. Suppose the label set S is given as $S = \mathbb{R}^m$ and $\{\hat{y}_i, y_i\}_{i=1}^{k} \subset S \times S$ is a collection of prediction/label pairs. The *mean absolute error* (MAE) or l_1 loss is

$$\text{MAE} = \frac{1}{k}\sum_{i=1}^{k} \|\hat{y}_i - y_i\|_1 ,$$

where

$$\|\hat{y}_i - y_i\|_1 = \sum_{j=1}^{m} \left|\hat{y}_i^{(j)} - y_i^{(j)}\right|$$

with (j) indicating the jth component of the vector.

Example 3. Suppose $S = \{G_1, \ldots, G_m\}$ and $(\hat{y}, y) \in S \times S$ is a prediction/label pair. Assume that $\hat{y}$ is a probability distribution. The *cross entropy loss* is

$$\mathcal{L} = -\sum_{j=1}^{m} y^{(j)} \log \hat{y}^{(j)}$$

with (j) indicating the jth component of the vector.

3 Linear Models

Definition 4. The process of *training* a model is the process of fitting the model to a dataset and updating the parameters of the model through some optimization procedure.

Example 4. Given a labeled dataset $\{(x_i, y_i) : x_i \in \mathbb{R}^n, y_i \in \mathbb{R}\}$, we can construct a *linear model*

$$y = w^\top x + b.$$

Here, $w \in \mathbb{R}^n$ is called the *weight vector* and $b \in \mathbb{R}$ is called the *bias*. With this model, define

$$\hat{y}_i = w^\top x_i + b.$$

The goal is to find w, b s.t. the loss function $\mathcal{L}$ is minimized. This is known as *linear regression* and can be extended to arbitrary finite dimensions.

Note. Let

$$\hat{y} = w^\top x + b = w^{(1)}x^{(1)} + \ldots + w^{(n)}x^{(n)} + b$$

be a linear model. There are limitations to linear models:

1. the model is a linear function in $x^{(i)}$;
2. if two features are linearly related, the components of w may be very large.

This can be accounted for through *regularization*, e.g. ridge regression or least absolute shrinkage and selection operator (LASSO).

4 Probabilistic Interpretation

Sometimes it is useful to interpret a model in a probabilistic manner. For example, if encountering a classification problem, this allows one to interpret a prediction as a confidence in choosing the correct class. To do this, we need to incorporate a function that rescale our outputs.

Example 5. The function

$$\sigma\colon \mathbb{R} \longrightarrow (0,1) \quad \text{where} \quad \sigma(x) = \frac{1}{1+e^{-x}}$$

is called the *logistic sigmoid.* This is shown in Figure 1

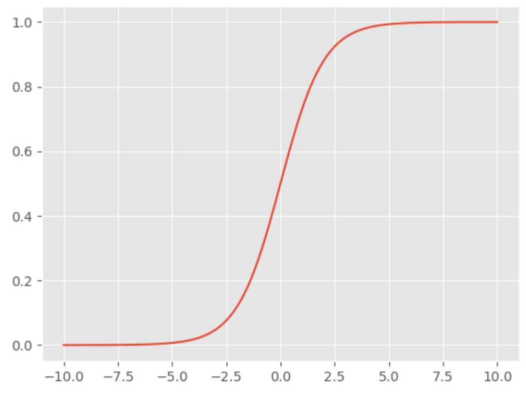

Figure 1: The logistic sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$.

Note. The *softmax* function will also allow one to interpret a model's output as a probability distribution and is computed as

$$\text{softmax}\colon \mathbb{R}^m \longrightarrow [0,1]^m \quad \text{where} \quad \text{softmax}(x)^{(i)} = \frac{e^{x^{(i)}}}{\sum_{j=1}^{m} e^{x^{(j)}}}.$$

In this sense, each component of the softmax output is a probability proportional to the exponentials of the input.

5 Nonlinear Functions

The key to neural networks is the incorporation of nonlinear functions. These allow the neural network more generalizability and the ability to adapt to any kind of dataset. Nonlinear functions help give neural networks the property that they are *universal approximators*—they can approximate any continuous function.

Example 6. The function

$$\tanh\colon \mathbb{R} \longrightarrow (-1,1) \quad \text{where} \quad \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x}-1}{e^{2x}+1}$$

is called the *hyperbolic tangent*. This is shown in Figure 2.

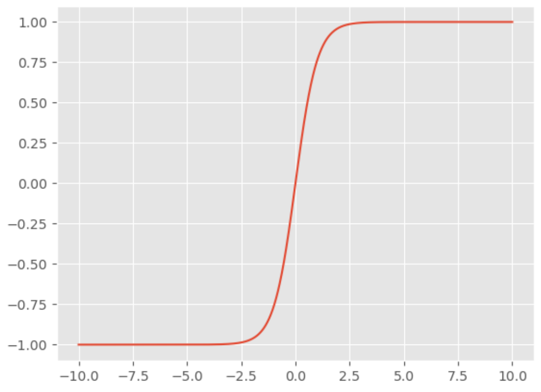

Figure 2: The hyperbolic tangent function $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

Example 7. The function

$$\text{ReLU}\colon \mathbb{R} \longrightarrow \mathbb{R}_{\geq 0} \quad \text{where} \quad \text{ReLU}(x) = \max(0,x).$$

is called the *rectified linear unit* (ReLU). This is shown in Figure 3.

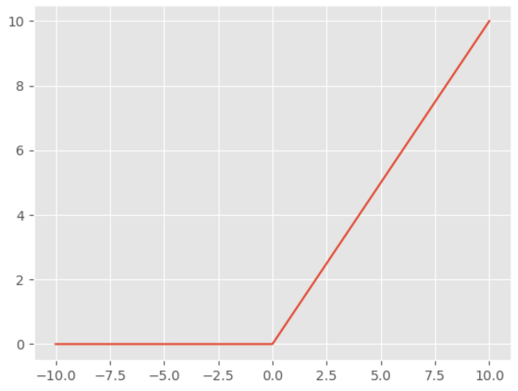

Figure 3: The ReLU function $\text{ReLU}(x) = \max(0,x)$.

Example 8. The function

$$\text{softplus}\colon \mathbb{R} \longrightarrow \mathbb{R}_{>0} \quad \text{where} \quad \text{softplus}(x) = \log(1+e^x).$$

is called the *softplus*. This is shown in Figure 4.

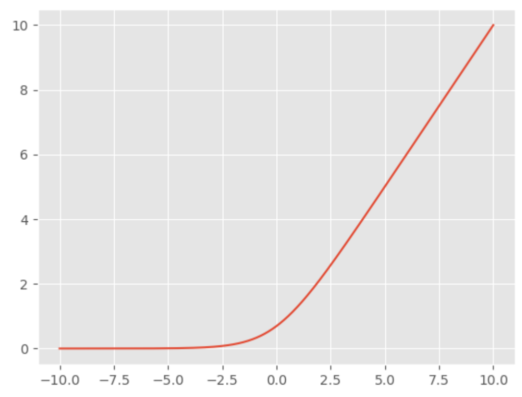

Figure 4: The softplus function $\mathrm{softplus}(x) = \log(1 + e^x)$.

6 Neural Networks

We are now ready to define a neural network.

Definition 5. Given nonlinear functions g_l and setting $h^{(0)} := x \in \mathbb{R}^n$, the function $y = f(x, \theta)$ defined by

$$\begin{aligned}
h^{(1)} &= g_{n_1}(W^{(1)}x + b^{(1)}) = g_{n_1}(w^{(1)}h^{(0)} + b^{(1)}) \in \mathbb{R}^{n_1} \\
h^{(2)} &= g_{n_2}(W^{(2)}h^{(1)} + b^{(2)}) \in \mathbb{R}_{n_2} \\
&\vdots \\
h^{(L)} &= g_{n_L}(W^{(L)}h^{(L-1)} + b^{(L)}) \in \mathbb{R}_{n_L} \\
y &= Wh^{(L)} + b \in \mathbb{R}^m
\end{aligned}$$

is called a *feedforward neural network* (FNN) or *multilayer perceptron* (MLP) with L layers. Here the $h^{(l)} \in \mathbb{R}^{n_l}$ are called *hidden variables* (or *hidden units*), $W^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$, and $b^{(l)} \in \mathbb{R}^{n_l}$. The *depth* of the network is L, the number of layers. We call the set of parameters

$$\theta = \left\{W^{(l)}, b^{(l)}\right\}_{l=1}^{L} \cup \{W, b\}$$

defining the network *trainable parameters*. We also call L and n_l *hyperparameters*. The nonlinear functions g_l are also called *activation functions*.