CSCI 5525 Machine Learning Fall 2019

Lecture 9: Neural Networks (Part 1)

Feb 25th, 2020

Lecturer: Steven Wu *Scribe: Steven Wu*

We have just learned about kernel functions that allow us to *implicitly* lift the raw feature vector x to expanded feature $\phi(x)$ that may lie in $\mathbb{R}^\infty$. The kernel trick allows us to make linear predictions $\phi(x)^\intercal \mathbf{w}$ without explicitly writing down the weight vector $\mathbf{w}$. Note that the mapping ϕ is fixed after we choose the hyperparameters.

Now we will talk about neural networks that were originally invented by Frank Rosenblatt. When neural network was first invented, it was called *multi-layer perceptron*. Similar to kernel, the approach of neural networks also makes prediction of the form $\phi(x)^\intercal \mathbf{w}$, but it *explicitly learns* the feature expansion mapping $\phi(x)$.

So how do we make an expressive feature mapping ϕ? One natural idea is to take composition of linear functions.

Warmup: composition of linear functions.

- First linear transformation: $x \to W_1 x + b_1$
- Second linear transformation: $x \to W_2(W_1 x + b_1) + b_2$
- ......
- L-th linear transformation: $x \to W_L(\ldots(W_1 x + b_1)\ldots) + b_L$

Question: do we gain anything?
Well, not quite. Observe that

$$W_L(\ldots(W_1 x + b_1)\ldots) + b_L = Wx + b,$$

where $W = W_L \ldots W_1$ and $b = b_L + W_L b_{L-1} + \ldots + W_L \ldots W_2 b_1$.

1 Non-linear activation

To go beyond linear function, we will need to introduce "non-linearity" between the linear functions. Recall that in the lecture of logistic regression, we introduce a probability model

$$\mathbf{Pr}[Y = 1 \mid X = x] = \frac{1}{1 + \exp(-w^\intercal x)} \equiv \sigma(w^\intercal x)$$

where σ is the *logistic* or *sigmoid* function. See Figure 2. Now consider a vector-valued version that applies the logisitc function coordinate wise: $f_i(z) = \sigma(W_i z + b_i)$. This gives the most basic neural network.

$$x \to (f_L \circ \cdots \circ f_1)(x), \quad \text{where} \quad f_i(z) = \sigma(W_i z + b_i).$$

Here we call the $\{W_i\}_{i=1}^L$ the *weights*, and $\{b_i\}_{i=1}^L$ the *biases*.

More generally, given a collection of *activation* (or *nonlinearities, transfer, link*) functions $\{\sigma_i\}_{i=1}^L$, weights, and biases, we can write down a basic form of a neural network:

$$F(x, \theta) = \sigma_L \left(W_L(\ldots W_2 \sigma_1(W_1 x + b_1) + b_2 \ldots) + b_L \right)$$

where θ denotes the set of parameters $W_1, \ldots, W_L, b_1, \ldots, b_L$.

DAG view. We can view a neural network as a directed acyclic graph (DAG). The input layer basically have each node corresponding to a single x_i. See the illustration in Figure 1. In some applications, each x_i might be vector-valued. For example, if x_i corresponds to a pixel, it should contain 3 values. In this case, each W_{ij} will also be a vector. Any layer that is not the input layer or the output layer is called a *hidden layer*.

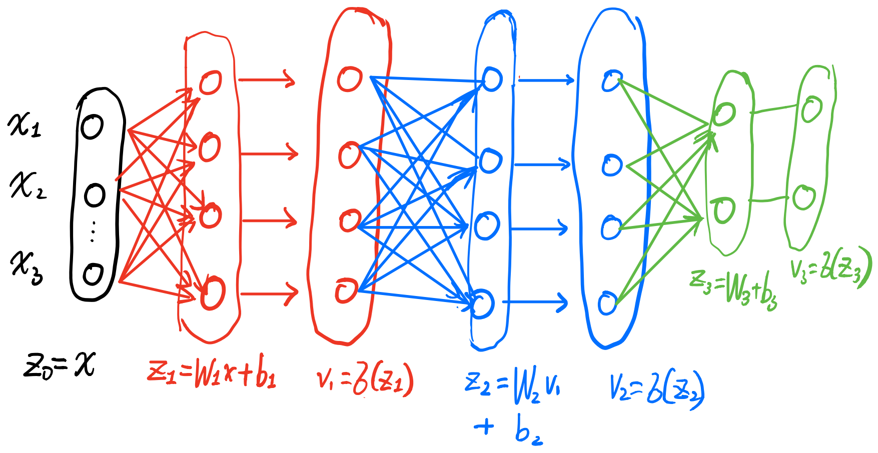

Figure 1: Graphical view of neural network.

1.1 Choices of activation functions.

- Indicator or threshold:

$$z \to \mathbf{1}[z \geq 0]$$

- Sigmoid or logistic (Figure 2):

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

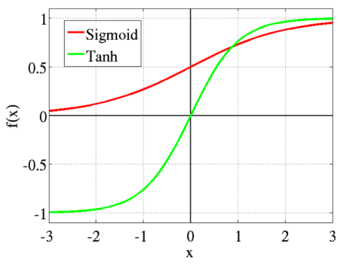

Figure 2: Logistic/sigmoid and hyperbolic functions

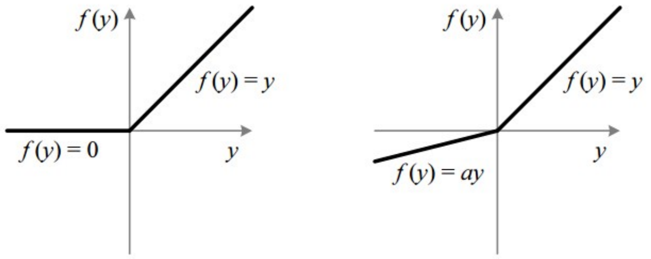

Figure 3: ReLU and Leaky ReLU functions

- Hyperbolic tangent:
$$z \to \tanh(z)$$
- Rectificed linear unit (ReLU) (Figure 3):
$$z \to \max\{0, z\}$$
Variants include Leaky ReLU and ELU. These are the most popular choices now since the AlexNet paper [1], which kicked off the Deep Learning revolution.
- Identity: $z \to z$. This is often used in the last layer when we evaluate the loss.

2 Expressiveness

It turns out this one hidden layer can already enable tremendously more representation power than a simple linear function. In fact, you can use such networks to approximate any "reasonable" functions according to the universal approximation theorem below.

Theorem 2.1 (Universal approximation theorem)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be any continuous function. For any approximation error $\epsilon > 0$, there exists a set of parameters $\theta = (W_1, b_1, W_2, b_2)$ such that for any $x \in [0, 1]^d$*
$$|f(x) - (W_2\sigma(W_1x + b_1) + b_2)| \leq \epsilon$$
where σ is a nonconstant, bounded, and continuous function (e.g. ReLU and logistic).

In other words, a single hidden layer neural network can approximate any continuous function to any degree of precision. However, such a neural network can be very wide, and even though it exists, we may not easily find it. More recently, there has been analogous universal approximation theorem with deep neural network with bounded widths that are essentially the dimension of the data [2].

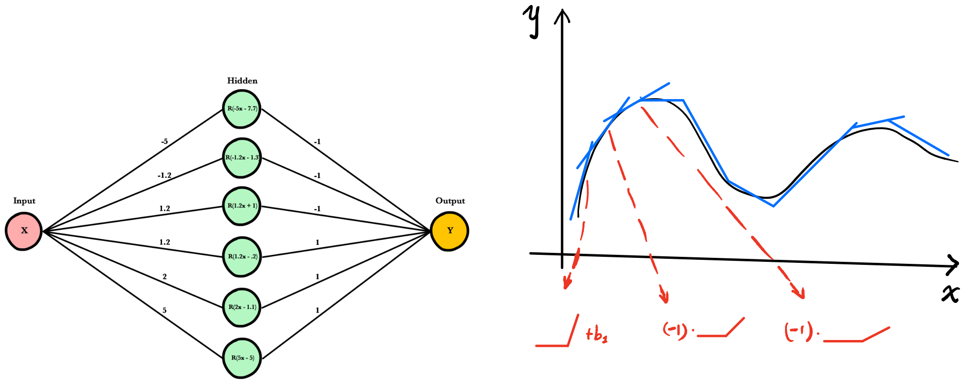

Figure 4: Universal approximation theorem in the special case where $x, f(x) \in \mathbb{R}$. On the left: the neural network graph in this case. On the right: intuition about the theorem. The continuous function f (in black) can be approximated a piecewise linear functions such that each piece is given by a weighted ReLU function (with an additive bias term).

3 Learning pipeline.

- Split the data into traning and validation datasets.
- **Hyperparameters:** pick a class of functions for the networks (or architecture), the function $F(\cdot, \cdot)$.
- **ERM:** on training data set $\{(x_i, y_i)\}$, we pick a loss function ℓ (e.g. cross entropy loss function, square loss) and perform *empirical risk minimization*:

$$\arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, F(x_i, \theta))$$

- Choose the architecture with the lowest validation error.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors,

Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012.

[2] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: A view from the width. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6231–6239. Curran Associates, Inc., 2017.