Artificial Neural Networks

Prof. Eric A. Suess

April 20, 2020

Introduction

Today we will introduce **Artificial Neural Networks** (ANN).

Get to know the **terms** involved in thinking about ANNs.

Introduction

The author begins the introduction with **magic**, discussion of the idea of a **black box**, and ends with "there is no need to be intimidated!"

black box

Neural Networks are considered a **black box** process.

ANNs are based on complex mathematical systems.

But not a **zero node NN** is an alternative representation of the simple linear regression model.

$y = mx + b$

$y(x) = w_1 x + w_2 1$

$y(x) = f(w_1 x + w_2 1)$

artificial neurons

- ANNs are **versatile learners** that can be applied to nearly any learning task: **classification**, **numeric prediction**, and even **unsupervised pattern recognition**.
- ANNs are best applied to problems where the **input data** and the **output data** are **well-understood** or at least fairly simple, yet the process that relates the input to the output is **extremely complex**.

artificial neurons

ANNs are designed as **conceptual models** of **human brain activity**.

- incoming signals received by a cell's **dendrites**
- signal transmitted through the **axon**
- **synapse**
- see page 207/221 and 208/222.
- **activation function**

artificial neurons

An artificial neuron with n input **dendrites**, with **weights** w on the inputs x, the **activation function** f, and the resulting signal y is the output **axon**.

$y(x) = f\left(\sum_{i=1}^{n} w_i x_i\right)$

Activation functions

In a biological sense, the **activation function** could be imagined as a process that involves summing the total input sinal and determining whether it meets the firing threshold.

If so, the neuron passes the signal on. Otherwise, it does nothing.

Activation functions

- **threshold** activation function
- **unit step** activation function
- **sigmoid** activation function - **differentiable**
- **linear** activation function
- **Gaussian** activation function - **Radial Basis Function** (RBF) network
- **relu** activation function

Activation functions

For many of the activation functions, the range of input values that affect the output signal is relatively **narrow**.

The **compression** of the sinal results in a saturated signal at the high and low ends of very dynamic inputs.

When this occurs, the activation function is called a **squashing function**.

The solution to this is to use **standardization/normalization** of the features.

Network topology

The capacity of a neural network to learn is rooted in its **topology**, or the paterns and structures of interconnected neurons.

- number of **layers** (see page 212/224)
- can the **network travel backward**? (see page 213/227)
- **number of nodes** (see page 214/228)

Number of layers

A set of neurons called **input nodes** receive unprocessed signals directly from the input data. Each input node is responsible for processing a single feature in the dataset.

The feature's value is transformed by the node's activation function. The signals resulting from the input nodes are received by the **output node**, which uses its own activation function to generate a final prediction.

- **single-layer** network
- **multilayer** network
- **hidden layers** / **deep learning**

Direction of infomation travel

- **feedforeward** networks - commonly used
- feedback networks - theortical - not used

When people talk about applying ANNs they are most likely talking about using the **multilayer preceptron** (MLP) topology.

Number of nodes in each layer

The number of **input nodes** is *predetermined* by the number of features in the input data.

The number of **output nodes** is *predetermined* by the number of outcomes to be modeled or the number of class level in the outcome.

The number of **hidden nodes** is *left to the user to decide* prior to training the model.

More complex network topologies with a **greater number** of network connections allow the learning of more complex problems.

But run the risk of **overfitting**.

Number of nodes in each layer

A **best practice** is to use the **fewest nodes** that result in **adequate performance** on a validation dataset.

It has been proven that a neural network with **at least one hidden layer** of suffiently many neurons is a **universal function approximator**.

Training ANNs

Learning by experience.

The network's connection weights reflect the patterns observed over time.

Training ANNs by adjusting connection weights is very computationally intensive.

An efficent method of traning an ANN was discovered, called **backpropagation**.

weights

How does the algorithm determine how much (or whether) a weight should be changed?

gradient descent

derivative of each activation function.

Modeling the strength of concrete

The author gives as an example of the use of ANNs.

The analysis of the **concrete** dataset.

Software: R neuralNet package

- A nice way to get started learning about ANNs in R is to read the paper in the The R Journal neuralnet: Training of Neural Networks
- I have made an Rnotebooks of the code presented in the paper.
- neuralnet.Rmd

software: R h2o package

- A modern machine learning software company is h2o.ai
- There is an R package to install and use the software.

Software: R tensorflow and keras packages

- Google's Deep Learning software is now available from within R by installing the tensorflow package.
- A very commonly used frontend for tensorflow is keras. There is an R package for keras also.
- R Tensorflow
- R Keras