Grokking Artificial intelligence Algorithms
by Rishal Hurbans

Section 9 Neural Networks

What are they?

Artificial neural networks (ANNs) are powerful tools in the machine learning toolkit, used in a variety of ways to accomplish objectives such as image recognition, natural language processing, and game playing. They are best used with unstructured data where features between data may be hard to understand. ANNs fall under the deep learning category of Ai. The components of the Perceptron are described by variables that are useful in calculating the output. Weights modify the inputs; that value is processed by a hidden node; and finally, the result is provided as the output. The Perceptron is useful for solving simple problems, but as the dimensions of the data increases, it becomes less feasible. ANNs use the principles of the Perceptron and apply them to many hidden nodes as opposed to a single one. Data for ANNs must be prepared to be suitable. ANNs also have hidden layers that are not directly observed outside the network. Only the inputs and outputs are interacted with, which leads to the perception of ANNs as being black boxes. Each hidden node is similar to the Perceptron. A hidden node takes inputs and weights and then computes the sum and an activation function. Then the results of each hidden node are processed by a single output node.

Forward Propagation

A trained ANN is a network that has learned from examples and adjusted its weights to best predict the class of new examples. As with the Perceptron, the first step is calculating the weighted sum of the inputs and the weight of each hidden node. The next step is calculating the activation of each hidden node. We are using the sigmoid function, and the input for the function is the weighted sum of the inputs calculated for each hidden node. Now we have the activation results for each hidden node. When we mirror this result back to neurons, the activation results represent the activation intensity of each neuron. Because different hidden nodes may be concerned with different relationships in the data through the weights, the activations can be used in conjunction to determine an overall activation that represents the chance of a collision, given the inputs. One of the important functions for activation in our example is the sigmoid function. This method describes the mathematical function that represents the S curve.

Back propagation

computes the cost, the amount by which weights should be updated by using the Chain Rule, and adds the weight-update results to the existing weights. This process will compute the change for each weight given the cost. Remember that cost is calculated by using the example features, predicted output, and expected output.

Activation Functions
There are many options and all can have different effects on the neural network. Some Activation Functions are: step unit, sigmoid, hyperbolic tangent, and rectified linear unit.
Designing ANNs
Designing ANNs is experimental and dependent on the problem that is being solved. The architecture and configuration of an ANN usually change through trial and error as an attempt to improve the performance of the predictions. The inputs and outputs of an ANN are the fundamental parameters for use of the network. After an ANN model has been trained, the trained ANN model will potentially be used in different contexts and systems, and by different people. The inputs and outputs define the interface of the network. An ANN can consist of multiple hidden layers with varying numbers of nodes in each layer. Adding more hidden layers allows us to solve problems with higher dimensions and more complexity in the classification discrimination line. Adding more layers allows these complex classification functions to be found. The selection of the number of layers and nodes in an ANN usually comes down to experimentation and iterative improvement. Over time, we may gain intuition about suitable configurations, based on experiencing similar problems and solving them with similar configurations. Weight initialization is important because it establishes a starting point from which the weight will be adjusted slightly over many iterations. Weights that are initialized to be too small lead to the vanishing gradient problem described earlier, and weights that are initialized to be too large lead to another problem, the exploding gradient problem—in which weights move erratically around the desired result. Various weight-initialization schemes exist, each with its own pros and cons. A rule of thumb is to ensure that the mean of the activation results in a layer is 0—the mean of all results of the hidden nodes in a layer. Also, the variance of the activation results should be the same: the variability of the results from each hidden node should be consistent over several iterations. We can use bias in an ANN by adding a value to the weighted sum of the input nodes or other layers in the network. A bias can shift the activation value of the activation function. A bias provides flexibility in an ANN and shifts the activation function left or right. A key rule of thumb is to ensure that all nodes on the same layer use the same activation function. In multilayer ANNs, different layers may use different activation functions based on the problem to be solved. A network that determines whether loans should be granted, for example, might use the sigmoid function in the hidden layers to determine probabilities and a step function in the output to get a clear 0 or 1 decision.
Cost functions influence the ANN greatly, and using the correct function for the problem and dataset at hand is important because it describes the goal for the ANN. One of the most common cost functions is mean square error. Finally, the learning rate of the ANN describes how dramatically weights are adjusted during backpropagation. A slow learning rate may result in a long training process because weights are updated by tiny

amounts each time, and a high learning rate might result in dramatic changes in the weights, making for a chaotic training process.

ANN types and use cases

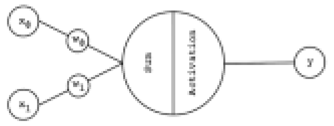
Convolutional neural networks (CNNs) are designed for image recognition. These networks can be used to find the relationships among different objects and unique areas within images. In image recognition, convolution operates on a single pixel and its neighbors within a certain radius. This technique is traditionally used for edge detection, image sharpening, and image blurring. CNNs use convolution and pooling to find relationships among pixels in an image. Convolution finds features in images, and pooling down samples the “patterns” by summarizing features, allowing unique signatures in images to be encoded concisely through learning from multiple images.

Recurrent neural networks (RNNs) accept a sequence of inputs with no predetermined length. These inputs are like spoken sentences. RNNs have a concept of memory consisting of hidden layers that represent time; this concept allows the network to retain information about the relationships among the sequences of inputs. When we are training a RNN, the weights in the hidden layers throughout time are also influenced by backpropagation; multiple weights represent the same weight at different points in time. RNNs are useful in applications pertaining to speech and text recognition and prediction. Related use cases include autocompletion of sentences in messaging applications, translation of spoken language to text, and translation between spoken languages.

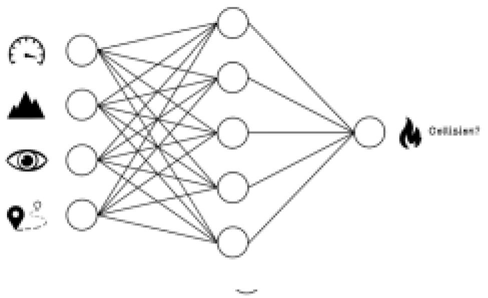
A generative adversarial network (GAN) consists of a generator network and a discriminator network. For example, the generator creates a potential solution such as an image or a landscape, and a discriminator uses real images of landscapes to determine the realism or correctness of the generated landscape. The error or cost is fed back into the network to further improve its ability to generate convincing landscapes and determine their correctness. The term adversarial is key, as we saw with game trees in chapter 3. These two components are competing to be better at what they do and, through that competition, generate incrementally better solutions. GANs are used to generate convincing fake videos (also known as deepfakes) of famous people, which raises concern about the authenticity of information in the media. GANs also have useful applications such as overlaying hairstyles on people’s faces. GANs have been used to generate 3D objects from 2D images, such as generating a 3D chair from a 2D picture. This use case may seem to be unimportant, but the network is accurately estimating and creating information from a source that is incomplete. It is a huge step in the advancement of AI and technology in general.

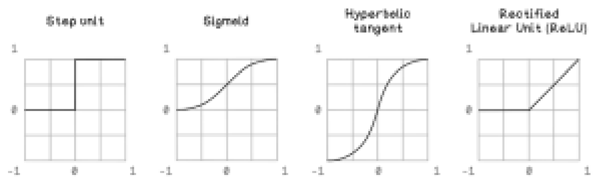
Important Figures

Artificial neural networks are inspired by the brain and can be seen as just another ML model.



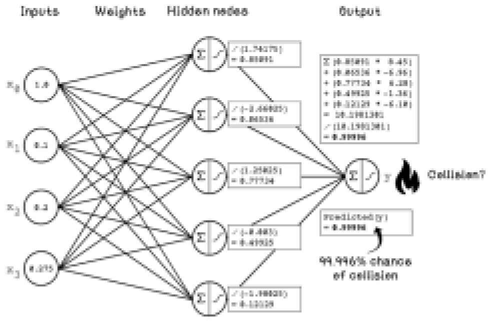
ANNs are based on the idea of the Perceptron.

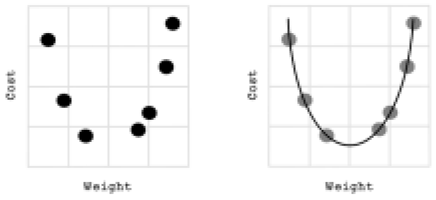




Activation functions help solve nonlinear problems.

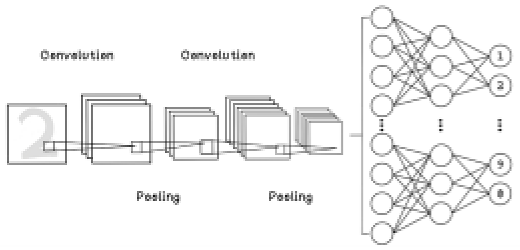
Forward propagation is used to use the ANN to make predictions and is also used in training.





Gradient descent optimization is one of many weight optimization options.

ANNs are flexible and can be adapted to solve many different problems.



Convolution
Convolution
Pooling
Pooling
1
2
9
0