

Deep Generative Models

Prepared by Dr. Sherif Mostafa

AI314

Master of Science in Artificial Intelligence

Lecture 2



Basics of Deep Learning

What is Deep Learning

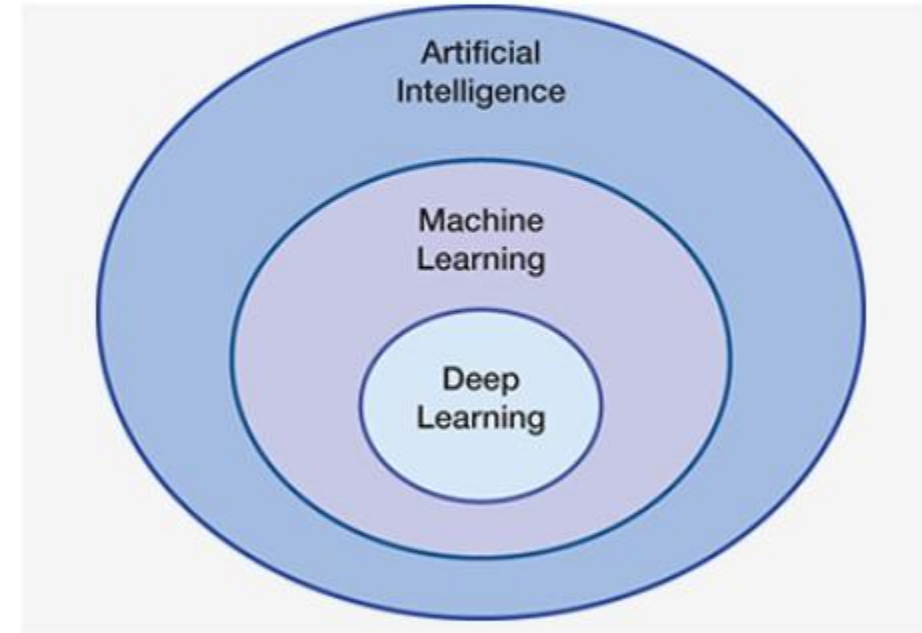
The perceptron

Multilayer Perceptron

Convolutional Neural Networks

What is deep Learning ?

Machine learning is a branch of artificial intelligence that gives computers the ability to learn without being programmed. Deep learning is a subfield of machine learning.



Common types of artificial neural networks

Neural network

is a system of artificial neurons-sometimes called perceptron- that are computational nodes used to classify and analyze data. The data is fed into the first layer of a neural network, with each perceptron making a decision, then passing that information onto multiple nodes in the next layer.

Training models with more than three layers are referred to as “deep neural networks” or “deep learning.” Some modern neural networks have hundreds or thousands of layers. The output of the final perceptron accomplish the task set to the neural network, such as classify an object or find patterns in data.

Common types of artificial neural networks

Feedforward neural networks (FF)

one of the oldest forms of neural networks, with data flowing one way through layers of artificial neurons until the output is achieved. In modern days, most feedforward neural networks are considered “deep feedforward” with several layers (and more than one “hidden” layer). Feedforward neural networks are typically paired with an error-correction algorithm called “backpropagation” that, in simple terms, starts with the result of the neural network and works back through to the beginning, finding errors to improve the accuracy of the neural network. Many simple but powerful neural networks are deep feedforward.

Common types of artificial neural networks

Recurrent neural networks (RNN)

differ from feedforward neural networks in that they typically use time series data or data that involves sequences. Unlike feedforward neural networks, which use weights in each node of the network, recurrent neural networks have “memory” of what happened in the previous layer as contingent to the output of the current layer. For instance, when performing natural language processing, RNNs can “keep in mind” other words used in a sentence. RNNs are often used for speech recognition, translation, and to caption images.

Common types of artificial neural networks

Long/short term memory (LSTM) is an advanced form of RNN that can use memory to “remember” what happened in previous layers. The difference between RNNs and LSTM is that LSTM can remember what happened several layers ago, through the use of “memory cells.” LSTM is often used in speech recognition and making predictions.

Common types of artificial neural networks

Generative adversarial networks (GAN) involve two neural networks competing against each other in a game that ultimately improves the accuracy of the output. One network (the generator) creates examples that the other network (the discriminator) attempts to prove true or false. GANs have been used to create realistic images and even make art.

What is deep Learning ?

Artificial intelligence is the capability of a machine to imitate intelligent human behavior. Machine learning (ML) is a branch of AI that gives computers the ability to “learn” from data without being explicitly programmed.

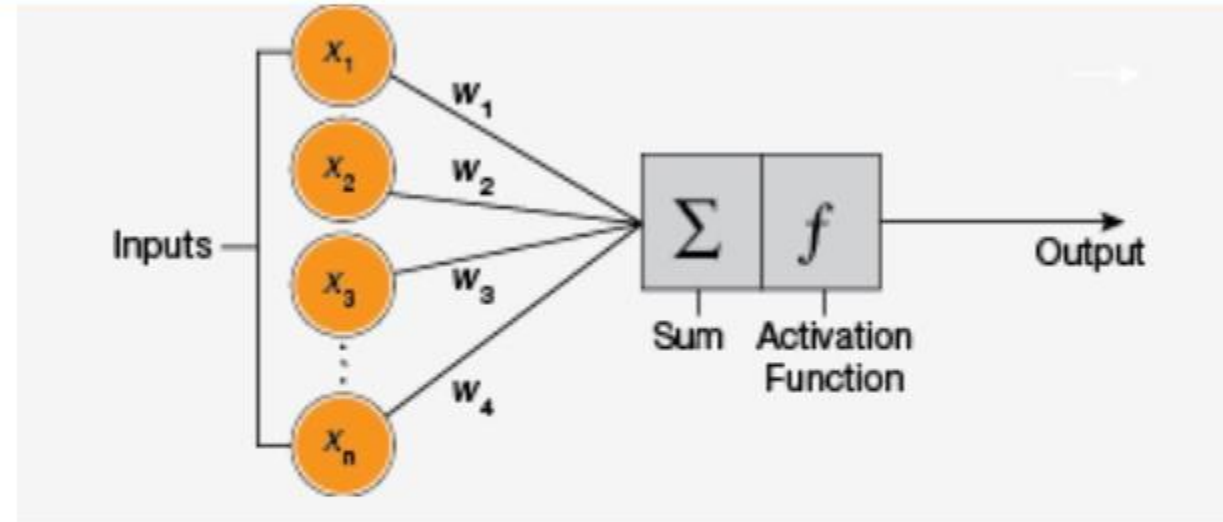
Deep learning is a subfield of ML that uses algorithms called artificial neural networks (ANNs), which are inspired by the structure and function of the brain and are capable of self-learning.

ANNs are trained to “learn” models and patterns rather than being explicitly told how to solve a problem. The building block of an ANN is called the perceptron

The perceptron

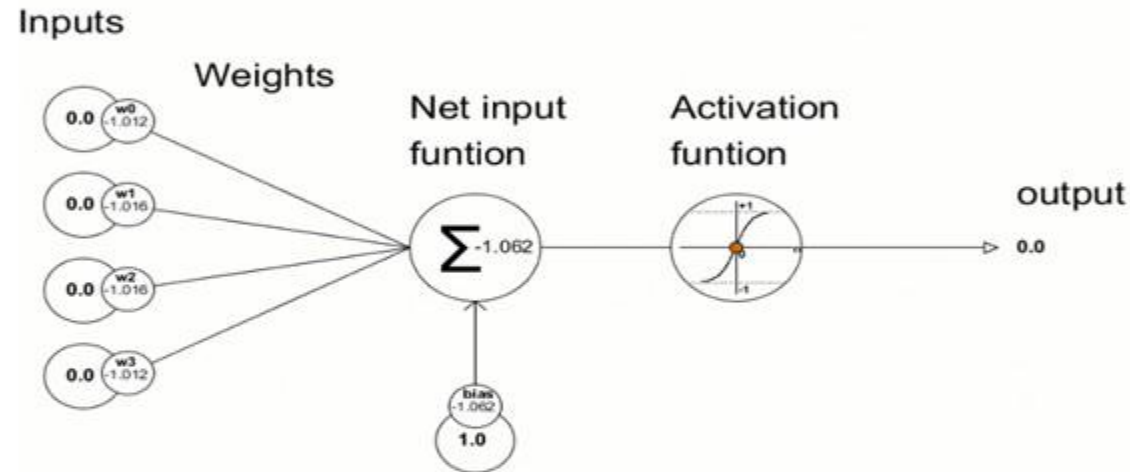
Perceptron is an algorithm that makes the neurons learn from the given information. It is of two types, Single-layer Perceptron does not contain hidden layers. Whereas Multi-layer Perceptron contains one or more hidden layers. Single-layer Perceptron is the simplest form of an Artificial neural network (ANN).

The perceptron



A perceptron is a mathematical model of a neuron. It receives weighted inputs, which are added together and passed to an activation function. The activation function decides whether it should produce an output.

The perceptron



A single neural network is a neural network that has only one layer of neurons. This type of network is also known as a perceptron. Perceptrons are the simplest type of neural network, and they can be used to solve simple problems.

The perceptron

A perceptron is a mathematical model of a biological neuron. It receives inputs and computes an output. Each input has an associated weight. All the inputs are individually multiplied by their weights, added together, and passed into an activation function that determines whether the neuron should fire and produce an output.

There are many different types of activation functions with different properties, but one of the simplest is the step function. A step function outputs a 1 if the input is higher than a certain threshold, otherwise it outputs a 0

The perceptron

For example, if a perceptron has two inputs (x_1 and x_2):

$$x_1 = 0.9$$

$$x_2 = 0.7$$

which have weightings (w_1 and w_2) of:

$$w_1 = 0.2$$

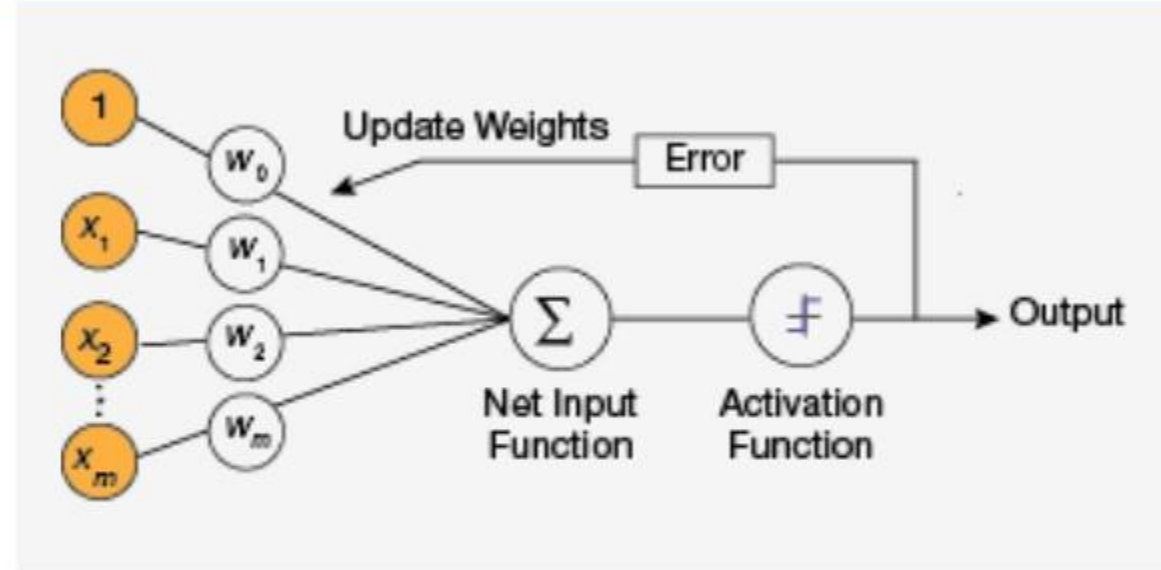
$$w_2 = 0.9$$

and the activation function threshold is equal to 0.75, then weighing the inputs and adding them together yields:

$$x_1w_1 + x_2w_2 = (0.9 \times 0.7) + (0.2 \times 0.9) = 0.81$$

Because the total input is higher than the threshold (0.75), the neuron will fire. Since we chose a simple step function, the output would be 1.

Training a perceptron

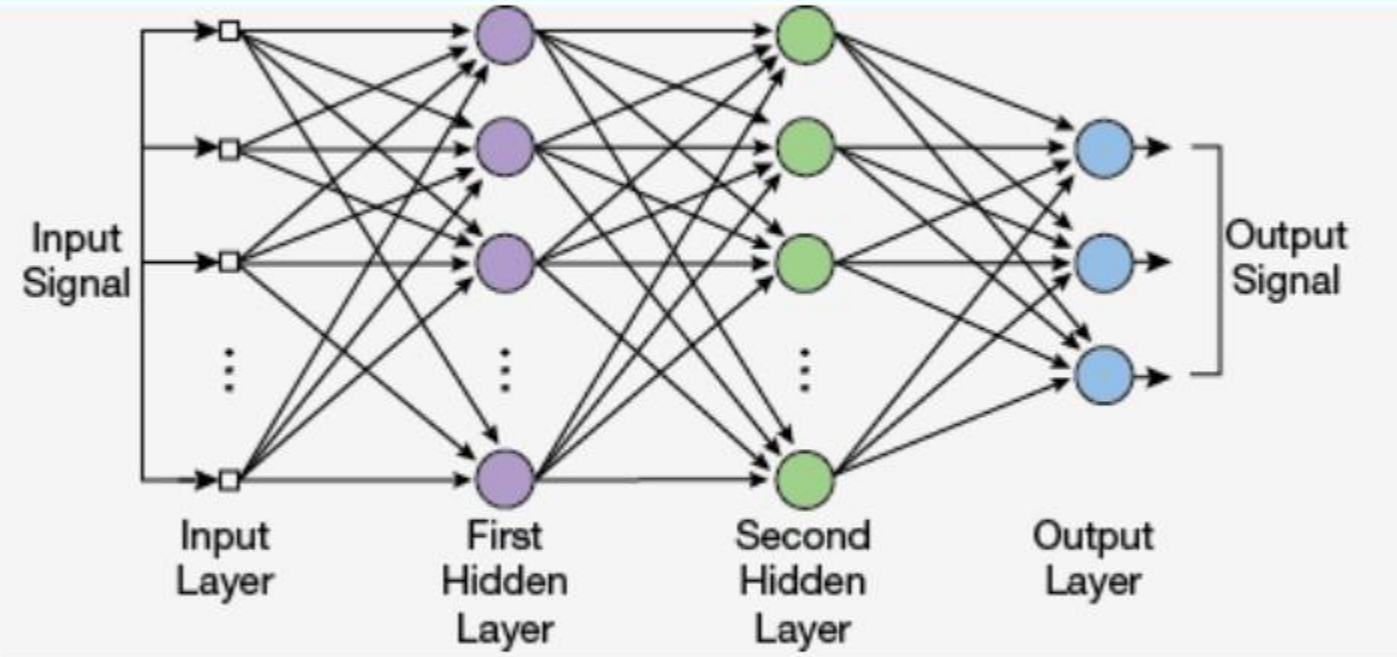


To train a perceptron, the weights are adjusted to minimize the output error. Output error is defined as the difference between the desired output and the actual output.

Training a perceptron

Training a perceptron involves feeding it multiple training samples and calculating the output for each of them. After each sample, the weights are adjusted to minimize the output error, usually defined as the difference between the desired (target) and the actual outputs

Multilayer perceptron



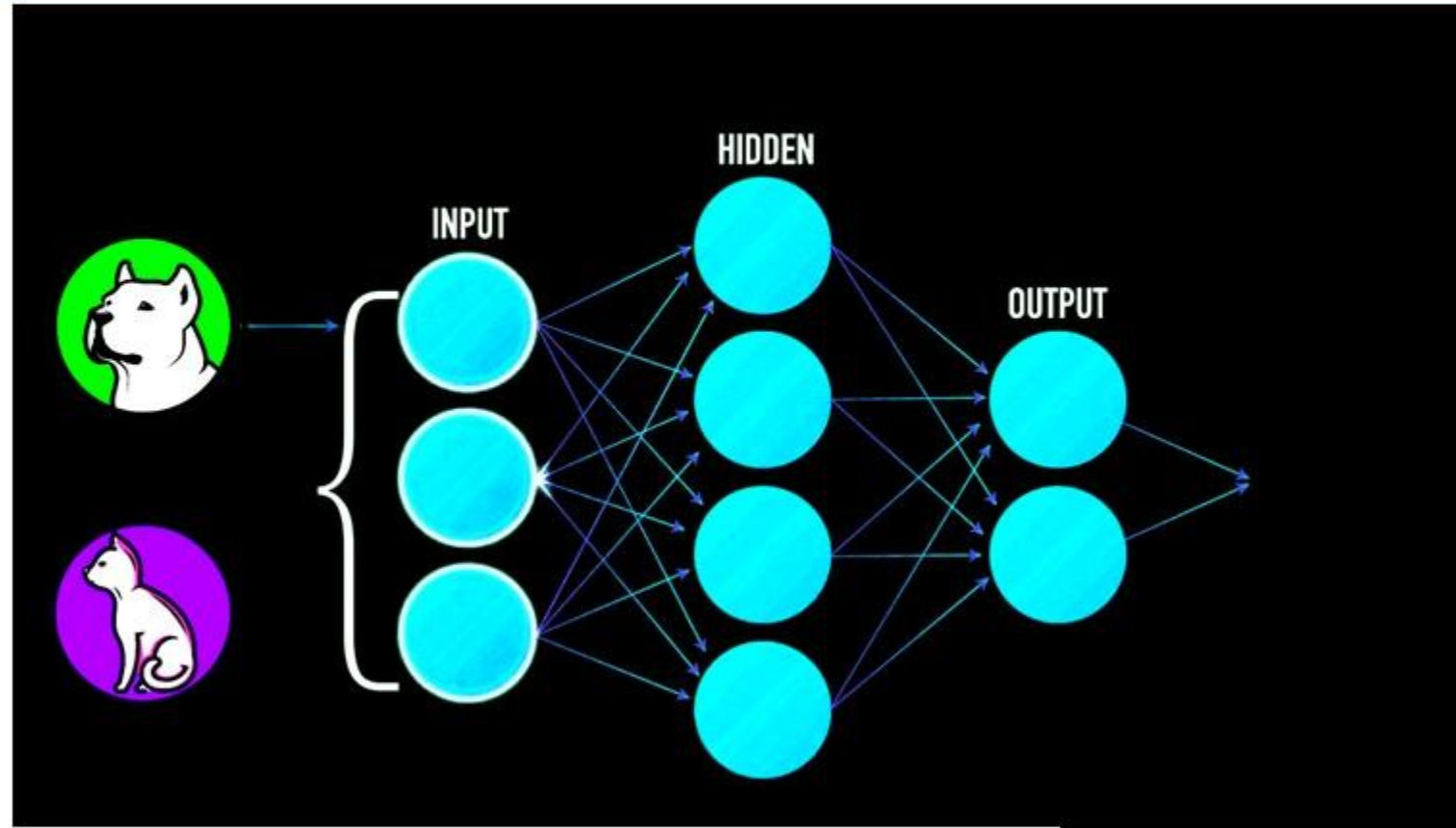
A multilayer perceptron has multiple layers of neurons between the input and output. Each neuron's output is connected to every neuron in the next layer.

Multilayer perceptron

A multilayer perceptron (MLP) is an artificial neural network with multiple layers of neurons between input and output. MLPs are also called feedforward neural networks. Feedforward means that data flow in one direction from the input to the output layer. Typically, every neuron's output is connected to every neuron in the next layer. Layers that come between the input and output layers are referred to as hidden layers.

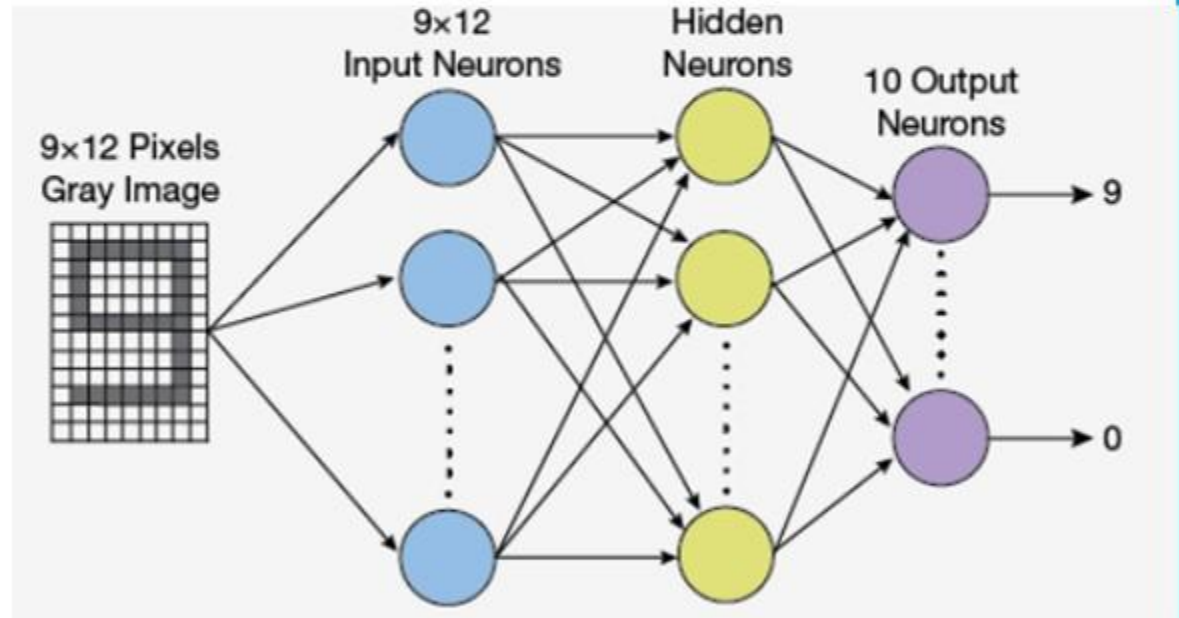
MLPs are widely used for pattern classification, recognition, prediction, and approximation, and can learn complicated patterns that are not separable using linear or other easily articulated curves. The capacity of an MLP network to learn complicated patterns increases with the number of neurons and layers

Multilayer perceptron



Multilayer perceptron

A multilayer perceptron for recognizing digits printed on a check would have a grid of inputs to read individual pixels of digits, followed by one or more layers of hidden neurons, and 10 output neurons to indicate which number was recognized. Initially, the MLP's output would be random, but as it is trained, it will adjust weights between the neurons and start classifying inputs correctly.



Training a multilayer perceptron

Training a single perceptron is easy — all weights are adjusted repeatedly until the output matches the expected value for all training data. For a single perceptron, weights can be adjusted using the formulas:

$$\Delta W_i = \eta(t - o)x_i$$

$$W_i + \Delta W_i \rightarrow W_i$$

where w_i is the weight, Δw_i is the weight adjustment, t is the target output, o is the actual output, and η is the learning rate — usually a small value used to moderate the rate of change of weights

Training a multilayer perceptron

However, this approach of tweaking each weight independently does not work for an MLP because each neuron's output is an input for all neurons in the next layer. Tweaking the weight on one connection impacts not only the neuron it propagates to directly, but also all of the neurons in the following layers as well, and thus affects all the outputs. Therefore, you cannot obtain the best set of weights by optimizing one weight at a time. Instead, the entire space of possible weight combinations must be searched simultaneously. The primary method for doing this relies on a technique called gradient descent.

Training a multilayer perceptron

That is how gradient descent works: If you can define a function over all weights that reflects the difference between the desired output and calculated output, then the function will be lowest (*i.e.*, the bottom of the hill) when the MLP's output matches the desired output. Moving toward this lowest value will become a matter of calculating the gradient (or derivative of the function) and taking a small step in the direction of the gradient.

Training a multilayer perceptron

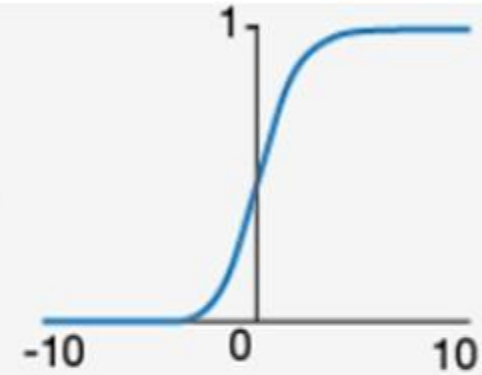
Backpropagation, short for “backward propagation of errors,” is the most commonly used algorithm for training MLPs using gradient descent. The backward part of the name stems from the fact that calculation of the gradient proceeds backward through the network. The gradient of the final layer of weights is calculated first and the gradient of the first layer of weights is calculated last.

Training a multilayer perceptron

Sigmoid and tanh functions are nonlinear activation functions. The output of the sigmoid function is a value between 0 and 1. The output of the sigmoid function can be used to represent a probability, often the probability that the input belongs to a category

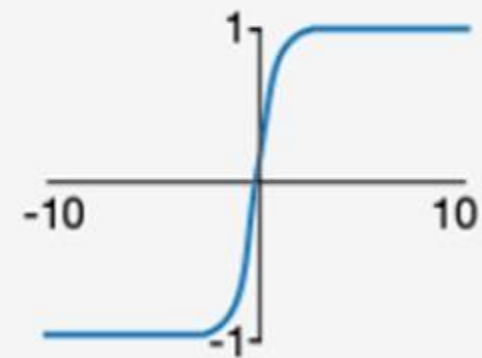
Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



tanh

$\tanh(x)$



Training a multilayer perceptron

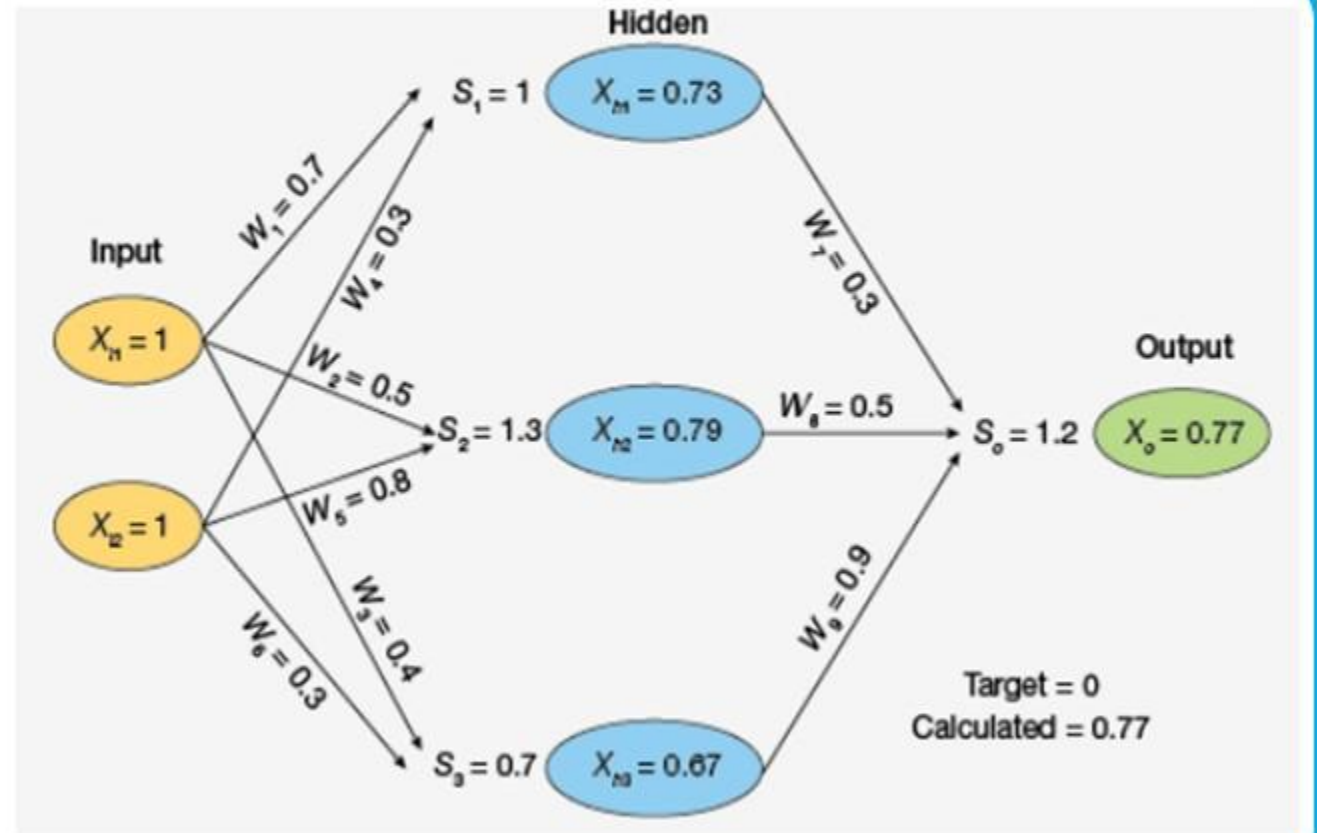
Before looking at how backpropagation works, recall that a perceptron calculates a weighted sum of its input and then decides whether it should fire. The decision about whether or not to fire is made by the activation function. In the perceptron example, we used a step function that outputted a 1 if the input was higher than a certain threshold, otherwise it outputted a 0. In practice, ANNs use nonlinear activation functions like the sigmoid or tanh functions, at least in part because a simple step function does not lend itself to calculating gradients — its derivative is 0.

Training a multilayer perceptron

The sigmoid function maps its input to the range 0 to 1. You might recall that probabilities, too, are represented by values between 0 and 1. Hence, the output of the sigmoid function can be used to represent a probability — often the probability that the input belongs to a category (*e.g.*, cat or dog). For this reason, it is one of the most widely used activation functions for artificial neural networks.

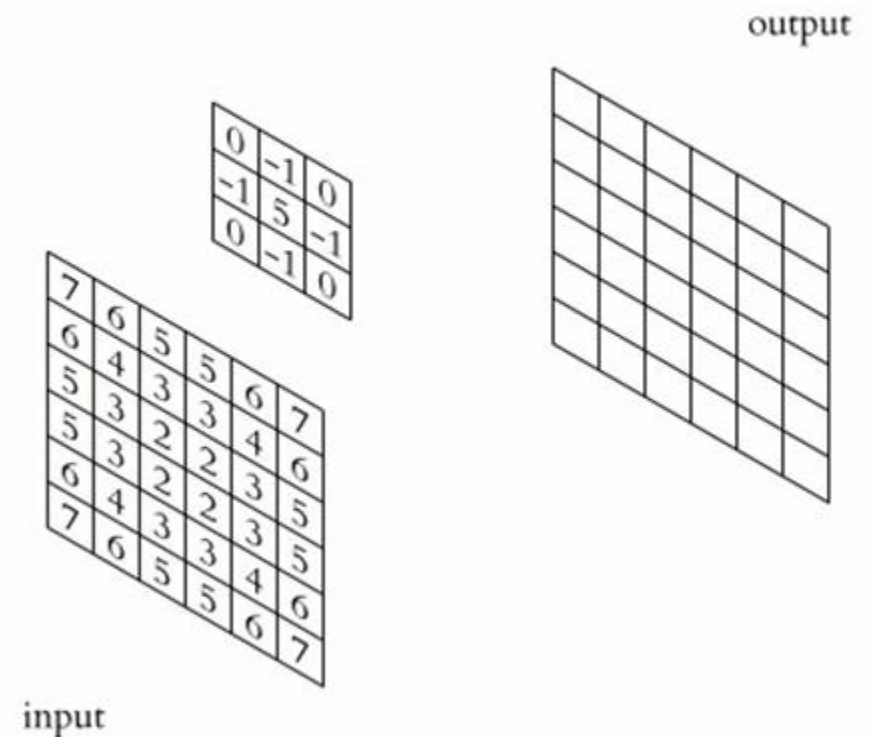
Training an MLP with backpropagation

An example MLP with three layers accepts an input of [1, 1] and computes an output of 0.77



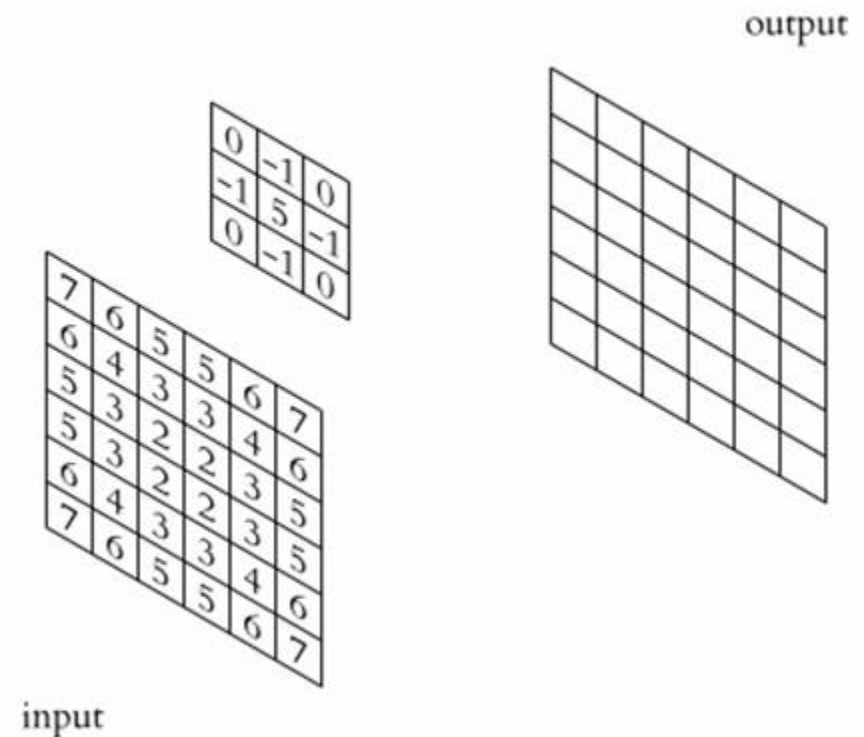
Convolutional Neural Networks (CNNs)

ConvNets or Convolution Networks are neural networks that use convolution of data matrices as one of the major functions. To be precise it uses cross-correlation instead of convolution. Cross-correlation is basically convolution without the kernel being flipped. Kernel is the smaller moving matrix in the below picture, helps in performing the convolution operation.



Convolutional Neural Networks (CNNs)

The kernel overlaps the first matrix or the input. Then the dot product of each overlapping term is calculated and added. This operation results in a convoluted image also called feature map (sharpened image in this case). Convolution is performed to extract different features from an image.



Stages involved in developing a common CNN

A. Convolution Layer

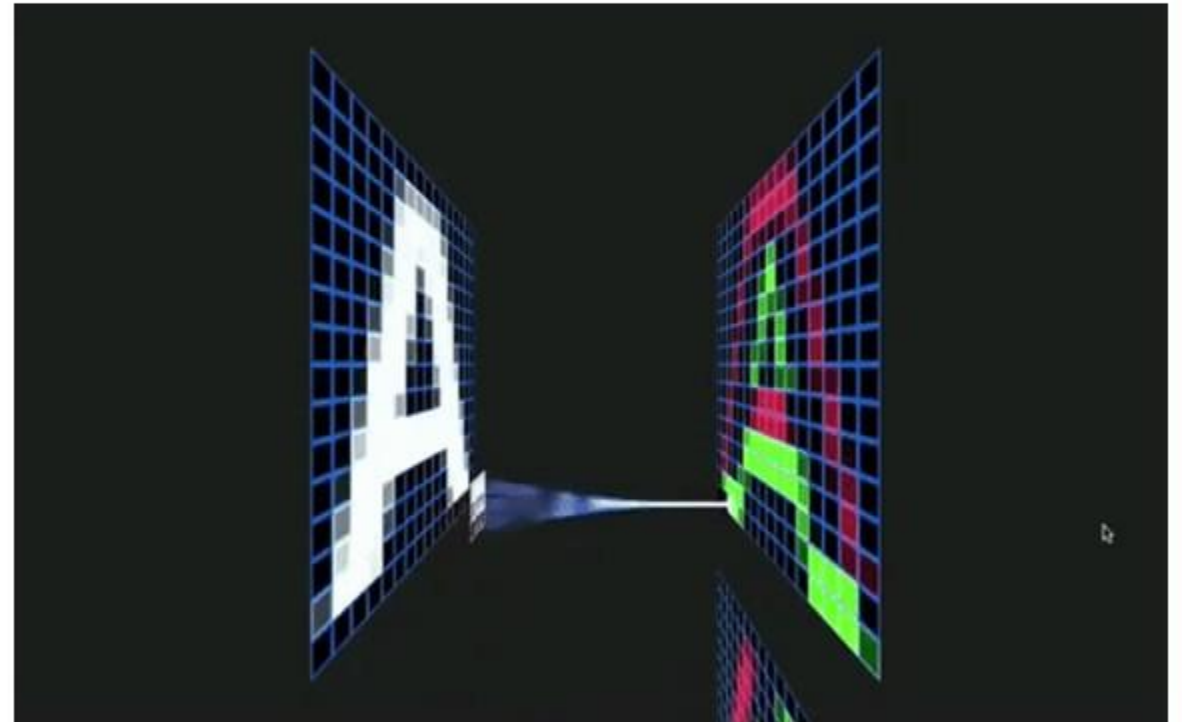
A convolutional layer is the main building block of a CNN. It contains a set of filters (or kernels), parameters of which are to be learned throughout the training. Filters size are usually smaller than the actual image. Each filter convolves with the image and creates an activation map. For convolution the filter slid across the height and width of the image and the dot product between every element of the filter and the input is calculated at every spatial position.



Stages involved in developing a common CNN

B. ReLu Layer

The feature map is then passed through the ReLu layer which uses the ReLu function ($=1$ if greater, $=0$ if less than threshold) to remove the negative samples (in red) from the above feature maps and creates a map that's just got positive values



Stages involved in developing a common CNN

C. Maxpool Layer

Maxpool Layer performs the pooling operation on the output from the detector stage. This results in a smaller sized image having lesser noise and better-defined features such as edges.



Stages involved in developing a common CNN

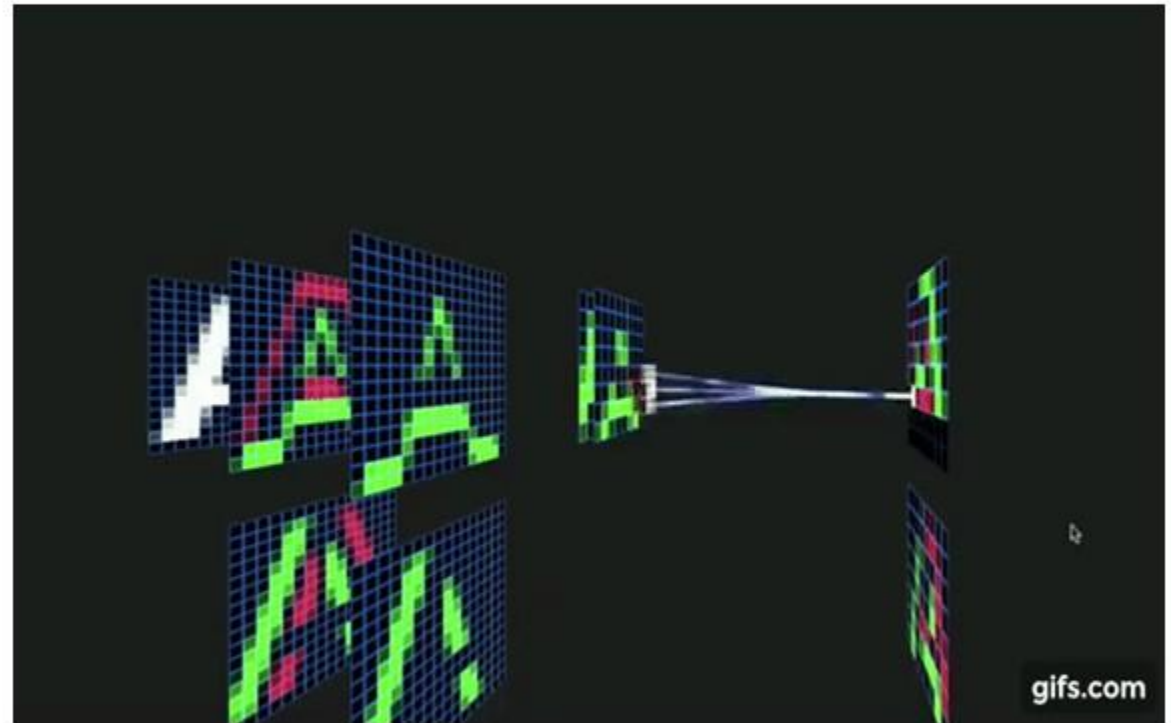
Pooling

It is required of a CNN to be stable in the sense that it's indifferent to the little changes in the input. To be able to achieve that CNNs tend to move a kernel over the feature map after the detector stage (convolution + ReLu) and take the max of all the pixels corresponding to the kernel within the rectangular neighborhood. This means for example, if we're detecting an eye in the top left corner of the image, a slight change in the image would not affect the decision to be made about the eye in the top left corner.

Stages involved in developing a common CNN

D. Add another combination of above layers to make a complex CNN

The last layer used for classification operation is a fully connected layer which is a linear layer performing the log of softmax algorithm to find the percentages of similarities between the final feature map and pattern to be classified.



Points to remember when implementing a CNN

It's better to apply pooling to some channels than all the channels, since we want the invariability to input in finding only some features but for the rest, we'd like to retain their every little detail.

Unsupervised learning in CNNs can provide for a better generalization and prevents overfitting. In such a case sometimes, you can get the weights of the rest of the model and train the last layer because it reduces the whole problem to a convex optimization problem, which can be solved using the Linear regression, but SVM is better at it in some cases.

Supervised learning is a machine learning model that maps a specific input to an output using labeled training data (structured data). In simple terms, to train the algorithm to recognize pictures of cats, feed it pictures labeled as cats.

Unsupervised learning is a machine learning model that learns patterns based on unlabeled data (unstructured data). Unlike supervised learning, the end result is not known ahead of time. Rather, the algorithm learns from the data, categorizing it into groups based on attributes. For instance, unsupervised learning is good at pattern matching and descriptive modeling.

In addition to supervised and unsupervised learning, a mixed approach called semi-supervised learning is often employed, where only some of the data is labeled. In semi-supervised learning, an end result is known, but the algorithm must figure out how to organize and structure the data to achieve the desired results.

Reinforcement learning is a machine learning model that can be broadly described as “learn by doing.” An “agent” learns to perform a defined task by trial and error (a feedback loop) until its performance is within a desirable range. The agent receives positive reinforcement when it performs the task well and negative reinforcement when it performs poorly. An example of reinforcement learning would be teaching a robotic hand to pick up a ball.

How does unsupervised learning work

unsupervised learning uses self-learning algorithms—they learn without any labels or prior training. Instead, the model is given raw, unlabeled data and has to infer its own rules and structure the information based on similarities, differences, and patterns without explicit instructions on how to work with each piece of data.

Unsupervised learning algorithms are better suited for more complex processing tasks, such as organizing large datasets into clusters. They are useful for identifying previously undetected patterns in data and can help identify features useful for categorizing data.

Unsupervised machine learning methods

In general, there are three types of unsupervised learning tasks: clustering, association rules, and dimensionality reduction.

Clustering

Clustering is a technique for exploring raw, unlabeled data and breaking it down into groups (or clusters) based on similarities or differences. It is used in a variety of applications, including customer segmentation, fraud detection, and image analysis. Clustering algorithms split data into natural groups by finding similar structures or patterns in uncategorized data.

Unsupervised machine learning methods

Association

Association rule mining is a rule-based approach to reveal interesting relationships between data points in large datasets. Unsupervised learning algorithms search for frequent if-then associations—also called rules—to discover correlations and co-occurrences within the data and the different connections between data objects.

Unsupervised machine learning methods

Dimensionality reduction

Dimensionality reduction is an unsupervised learning technique that reduces the number of features, or dimensions, in a dataset. More data is generally better for machine learning, but it can also make it more challenging to visualize the data. Dimensionality reduction extracts important features from the dataset, reducing the number of irrelevant or random features present

Some real-world unsupervised learning examples

- **Anomaly detection:** Unsupervised clustering can process large datasets and discover data points that are atypical in a dataset.
- **Recommendation engines:** Using association rules, unsupervised machine learning can help explore transactional data to discover patterns or trends that can be used to drive personalized recommendations for online retailers.
- **Customer segmentation:** Unsupervised learning is also commonly used to generate buyer persona profiles by clustering customers' common traits or purchasing behaviors. These profiles can then be used to guide marketing and other business strategies.

Some real-world unsupervised learning examples

- **Fraud detection:** Unsupervised learning is useful for anomaly detection, revealing unusual data points in datasets. These insights can help uncover events or behaviors that deviate from normal patterns in the data, revealing fraudulent transactions or unusual behavior like bot activity.
- **Natural language processing (NLP):** Unsupervised learning is commonly used for various NLP applications, such as categorizing articles in news sections, text translation and classification, or speech recognition in conversational interfaces.

Supervised learning vs. unsupervised learning

The main difference between supervised learning and unsupervised learning is the type of input data that you use. Unlike unsupervised machine learning algorithms, supervised learning relies on labeled training data to determine whether pattern recognition within a dataset is accurate.

The goals of supervised learning models are also predetermined, meaning that the type of output of a model is already known before the algorithms are applied. In other words, the input is mapped to the output based on the training data.



Thank You