ARTICLE

A beginner’s guide to artificial intelligence, machine learning, and cognitive computing

Explore some of the important aspects of AI and its subfields

By M. Tim Jones | Last updated October 12, 2018

Artificial Intelligence Machine Learning

For millennia, humans have pondered the idea of building intelligent machines. Ever since, artificial intelligence (AI) has had highs and lows, demonstrated successes and unfulfilled potential. Today, the news is filled with the application of machine learning algorithms to new problems. From cancer detection and prediction to image understanding and summarization and natural language processing, AI is empowering people and changing our world.

The history of modern AI has all the elements of a great drama. Beginning in the 1950s with a focus on thinking machines and interesting characters like Alan Turing and John von Neumann, AI began its first rise. Decades of booms and busts and impossibly high expectations followed, but AI and its pioneers pushed forward. AI is now exposing its true potential, focusing on applications and delivering technologies like deep learning and cognitive computing.

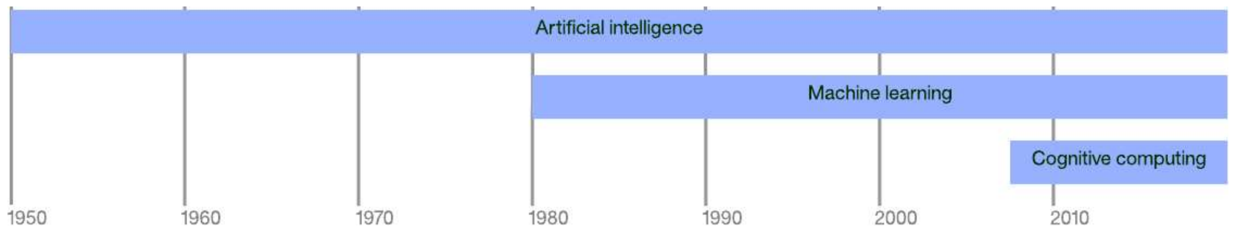
This article explores some of the important aspects of AI and its subfields. Let’s begin with a timeline of AI, and then dig into each of these elements.

Timeline of modern AI

Beginning in the 1950s, modern AI focused on what was called strong AI, which referred to AI that could generally perform any intellectual task that a human could.

The lack of progress in strong AI eventually led to what's called weak AI, or applying AI techniques to narrower problems. Until the 1980s, AI research was split between these two paradigms. But, around 1980, machine learning became a prominent area of research, its purpose to give computers the ability to learn and build models so that they could perform activities like prediction within specific domains.

Figure 1. A timeline of modern artificial intelligence



Building on research from both AI and machine learning, deep learning emerged around 2000. Computer scientists used neural networks in many layers with new topologies and learning methods. This evolution of neural networks has successfully solved complex problems in various domains.

In the past decade, cognitive computing has emerged, the goal of which is to build systems that can learn and naturally interact with humans. Cognitive computing was demonstrated by IBM Watson™ by successfully defeating world-class opponents at the *Jeopardy!* television game show.

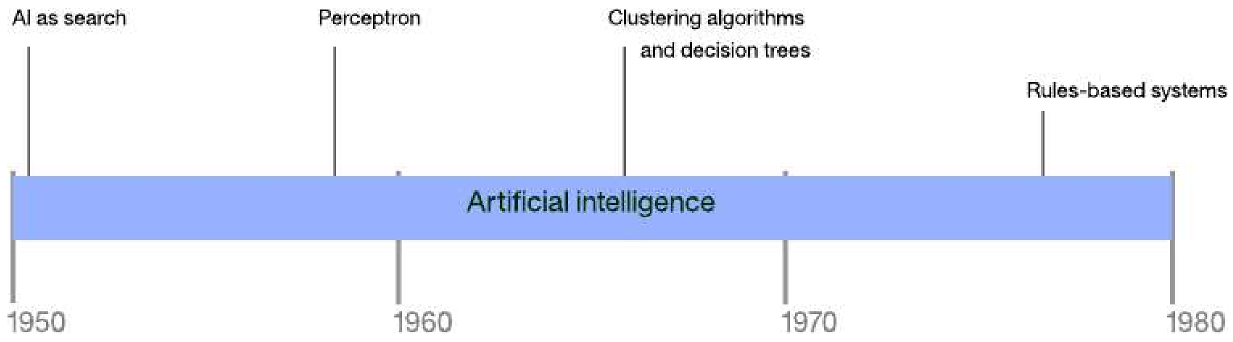
In this article, I'll explore each of these areas and explain some of the important algorithms that have driven their success.

Foundational AI

Research prior to 1950 introduced the idea that the brain consisted of an electrical network of pulses that fired and somehow orchestrated thought and consciousness. Alan Turing showed that any computation could be implemented digitally. The idea, then, that building a machine that could mimic the human brain couldn't be far off.

Much early research focused on this strong aspect of AI, but this period also introduced the foundational concepts that all machine learning and deep learning are built on today.

Figure 2. Timeline of artificial intelligence approaches to 1980



AI as search

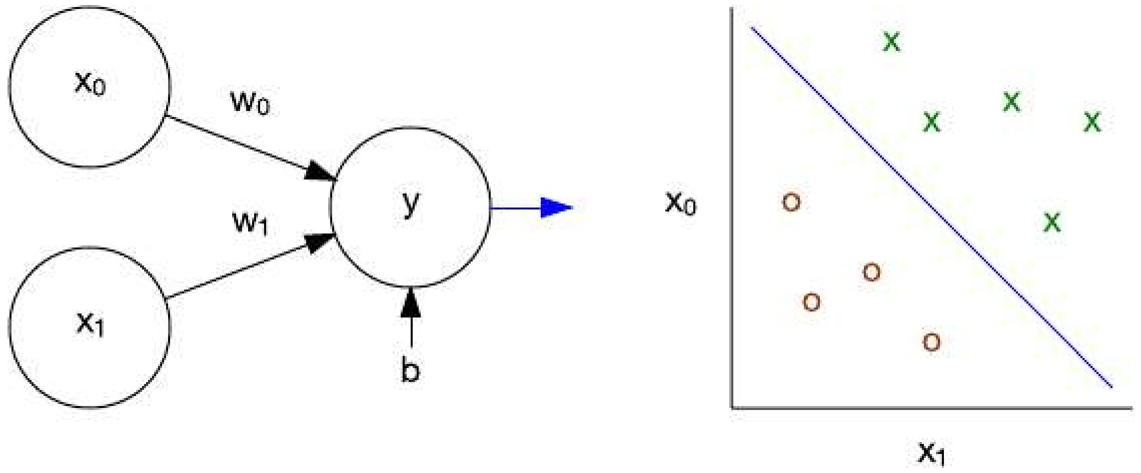
Many problems in AI can be solved through brute-force search (such as depth or breadth first search). However, considering the search space for moderate problems, basic search quickly suffers. One of the earliest examples of AI as search was the development of a checkers-playing program. Arthur Samuel built the first such program on the IBM 701 Electronic Data Processing Machine, implementing an optimization to search trees called *alpha-beta pruning*. His program also recorded the reward for a specific move, allowing the application to learn with each game played (making it the first self-learning program). To increase the rate at which the program learned, Samuel programmed it to play itself, increasing its ability to play and learn.

Although you can successfully apply search to many simple problems, the approach quickly fails as the number of choices increases. Take the simple game of tic-tac-toe as an example. At the start of a game, there are nine possible moves. Each move results in eight possible countermoves, and so on. The full tree of moves for tic-tac-toe contains (unoptimized for rotation to remove duplicates) is 362,880 nodes. If you then extend this same thought experiment to chess or Go, you quickly see the downside of search.

Perceptrons

The perceptron was an early supervised learning algorithm for single-layer neural networks. Given an input feature vector, the perceptron algorithm could learn to classify inputs as belonging to a specific class. Using a training set, the network's weights and bias could be updated for linear classification. The perceptron was first implemented for the IBM 704, and then on custom hardware for image recognition.

Figure 3. Perceptron and linear classification

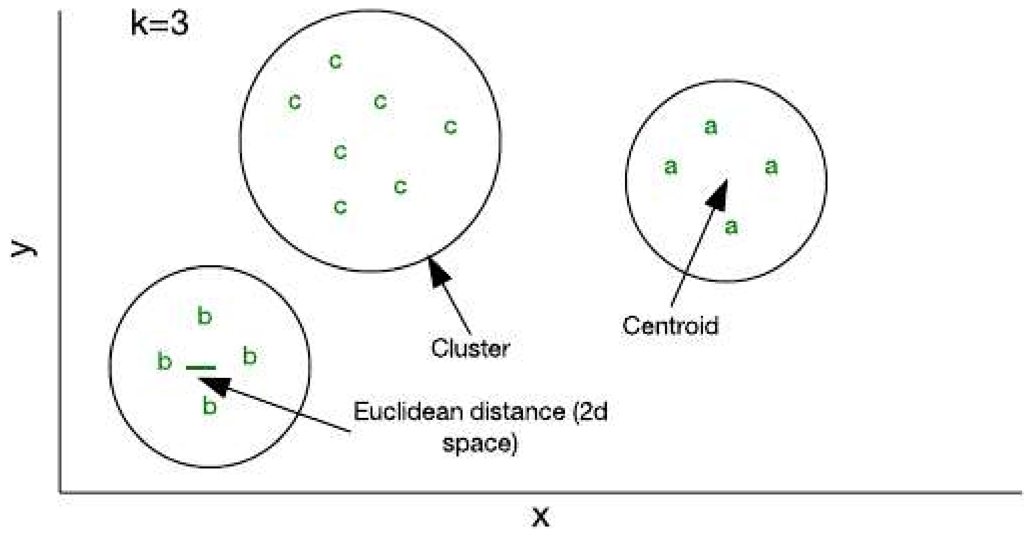


As a linear classifier, the perceptron was capable of linear separable problems. The key example of the limitations of the perceptron was its inability to learn an exclusive OR (XOR) function. Multilayer perceptrons solved this problem and paved the way for more complex algorithms, network topologies, and deep learning.

Clustering algorithms

With perceptrons, the approach was supervised. Users provided data to train the network, and then test the network against new data. Clustering algorithms take a different approach called unsupervised learning. In this model, the algorithm organizes a set of feature vectors into clusters based on one or more attributes of the data.

Figure 4. Clustering in a two-dimensional feature space



One of the simplest algorithms that you can implement in a small amount of code is called k-means. In this algorithm, *k* indicates the number of clusters in which you can assign samples. You can initialize a cluster with a random feature vector, and then add all other samples to their closest cluster (given that each sample represents a feature vector and a Euclidean distance used to identify "distance"). As you add samples to a cluster, its centroid—that is, the center of the cluster—is recalculated.

The algorithm then checks the samples again to ensure that they exist in the closest cluster and ends when no samples change cluster membership.

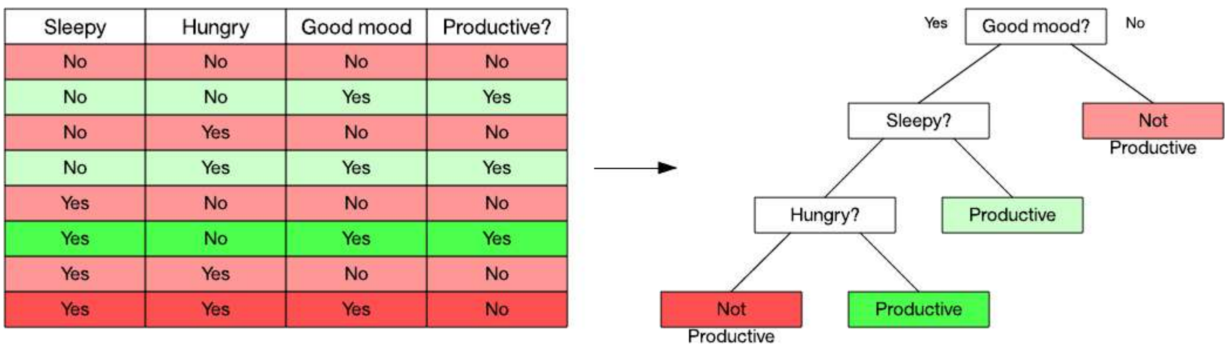
Although k-means is relatively efficient, you must specify *k* in advance. Depending on the data, other approaches might be more efficient, such as hierarchical or distribution-based clustering.

Decision trees

Closely related to clustering is the decision tree. A decision tree is a predictive model about observations that lead to some conclusion. Conclusions are represented as leaves on the tree, while nodes are decision points where an observation diverges. Decision trees are built from decision tree learning algorithms, where the data set is split into subsets based on attribute value tests (through a process called recursive partitioning).

Consider the example in the following figure. In this data set, I can observe when someone was productive based on three factors. Using a decision tree learning algorithm, I can identify attributes by using a metric (one example is information gain). In this example, mood is a primary factor in productivity, so I split the data set according to whether "good mood" is Yes or No. The No side is simple: It's always nonproductive. But, the Yes side requires me to split the data set again based on the other two attributes. I've colorized the data set to illustrate where observations led to my leaf nodes.

Figure 5. A simple data set and resulting decision tree



Sleepy	Hungry	Good mood	Productive?
No	No	No	No
No	No	Yes	Yes
No	Yes	No	No
No	Yes	Yes	Yes
Yes	No	No	No
Yes	No	Yes	Yes
Yes	Yes	No	No
Yes	Yes	Yes	No

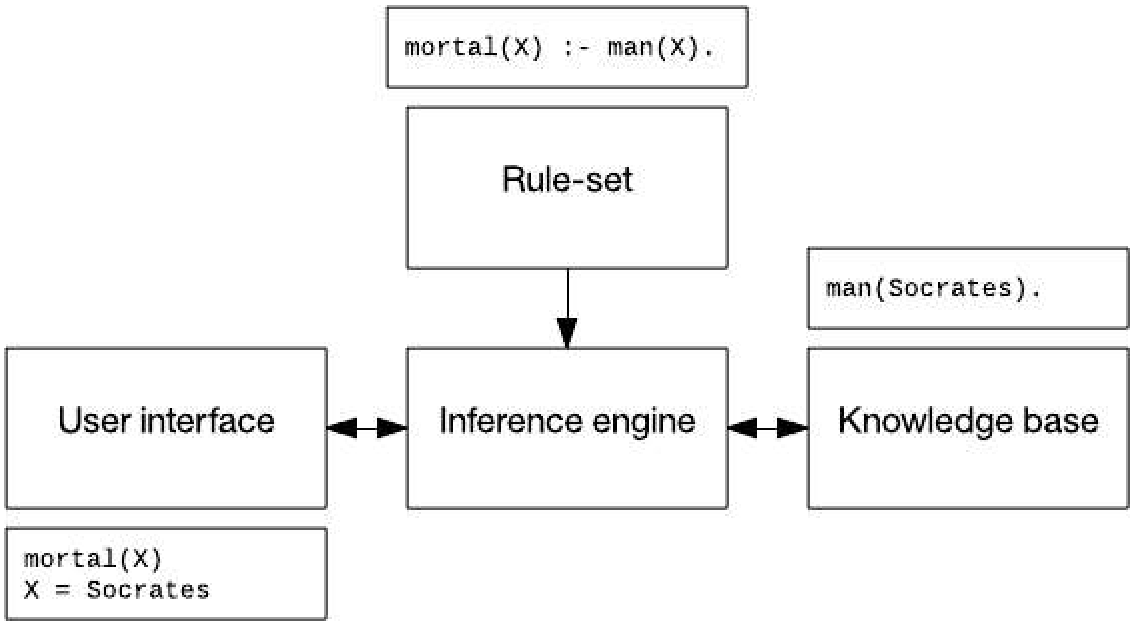
A useful aspect of decision trees is their inherent organization, which gives you the ability to easily (and graphically) explain how you classified an item. Popular decision tree learning algorithms include C4.5 and the Classification and Regression Tree.

Rules-based systems

The first system built on rules and inference, called Dendral, was developed in 1965, but it wasn't until the 1970s that these so-called "expert systems" hit their stride. A rules-based system is one that stores both knowledge and rules and uses a reasoning system to draw conclusions.

A rules-based system typically consists of a rule set, a knowledge base, an inference engine (using forward or backward rule chaining), and a user interface. In the following figure, I use a piece of knowledge ("Socrates was a man"), a rule ("if man, then mortal"), and an interaction on who is mortal.

Figure 6. A rules-based system

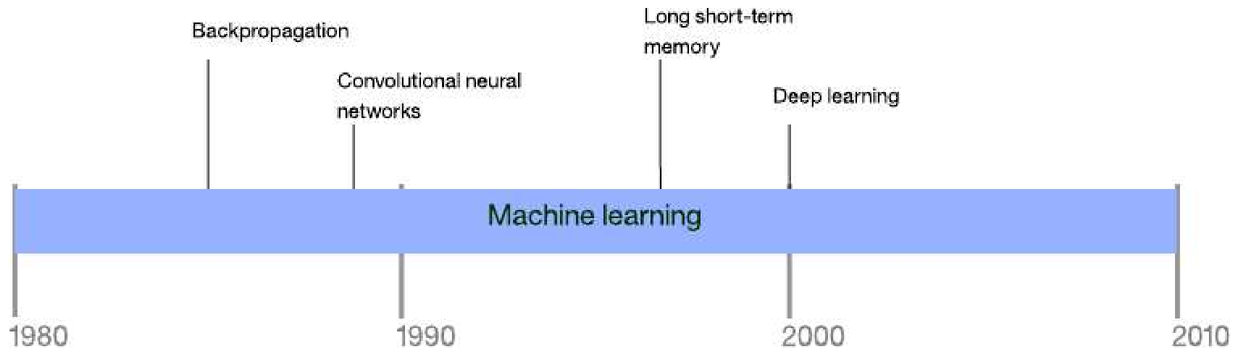


Rules-based systems have been applied to speech recognition, planning and control, and disease identification. One system developed in the 1990s for monitoring and diagnosing dam stability, called Kaleidos, is still in operation today.

Machine learning

Machine learning is a subfield of AI and computer science that has its roots in statistics and mathematical optimization. Machine learning covers techniques in supervised and unsupervised learning for applications in prediction, analytics, and data mining. It is not restricted to deep learning, and in this section, I explore some of the algorithms that have led to this surprisingly efficient approach.

Figure 7. Timeline of machine learning approaches

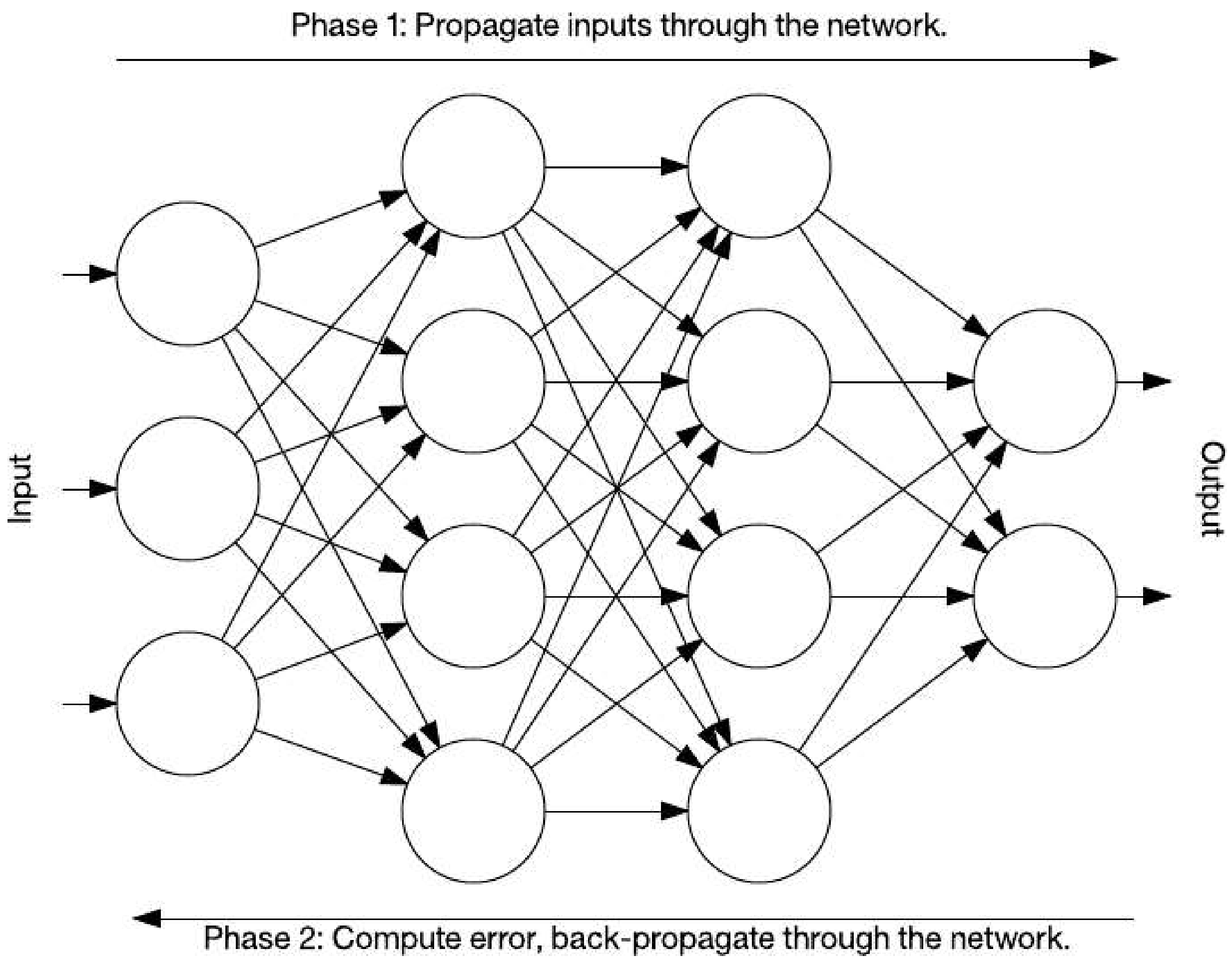


Backpropagation

The true power of neural networks is their multilayer variant. Training single-layer perceptrons is straightforward, but the resulting network is not very powerful. The question became, How can we train networks that have multiple layers? This is where backpropagation came in.

Backpropagation is an algorithm for training neural networks that have many layers. It works in two phases. The first phase is the propagation of inputs through a neural network to the final layer (called feedforward). In the second phase, the algorithm computes an error, and then backpropagates this error (adjusting the weights) from the final layer to the first.

Figure 8. Backpropagation in a nutshell


Phase 1: Propagate inputs through the network.
Input
Output
Phase 2: Compute error, back-propagate through the network.

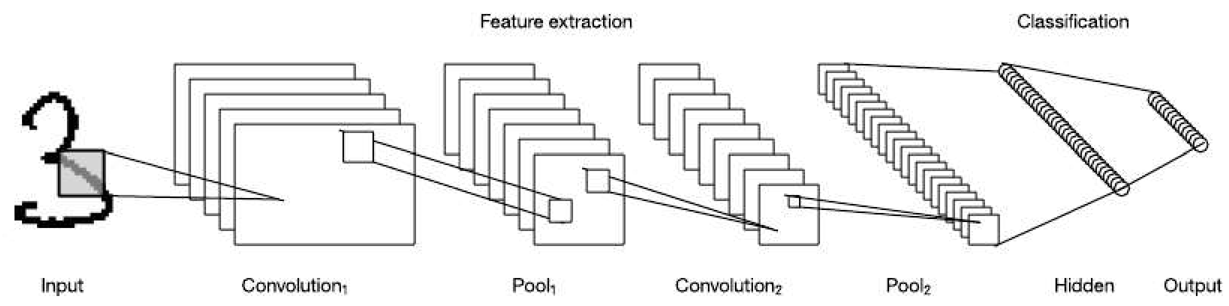
During training, intermediate layers of the network organize themselves to map portions of the input space to the output space. Backpropagation, through supervised learning, identifies an error in the input-to-output mapping, and then adjusts the weights accordingly (with a learning rate) to correct this error. Backpropagation continues to be an important aspect of neural network learning. With faster and cheaper computing resources, it continues to be applied to larger and denser networks.

Convolutional neural networks

Convolutional neural networks (CNNs) are multilayer neural networks that take their inspiration from the animal visual cortex. The architecture is useful in various applications, including image processing. The first CNN was created by Yann LeCun, and at the time, the architecture focused on handwritten character-recognition tasks like reading postal codes.

The LeNet CNN architecture is made up of several layers that implement feature extraction, and then classification. The image is divided into receptive fields that feed into a convolutional layer that extracts features from the input image. The next step is pooling, which reduces the dimensionality of the extracted features (through down-sampling) while retaining the most important information (typically through max pooling). The algorithm then performs another convolution and pooling step that feeds into a fully connected, multilayer perceptron. The final output layer of this network is a set of nodes that identify features of the image (in this case, a node per identified number). Users can train the network through backpropagation.

Figure 9. The LeNet convolutional neural network architecture



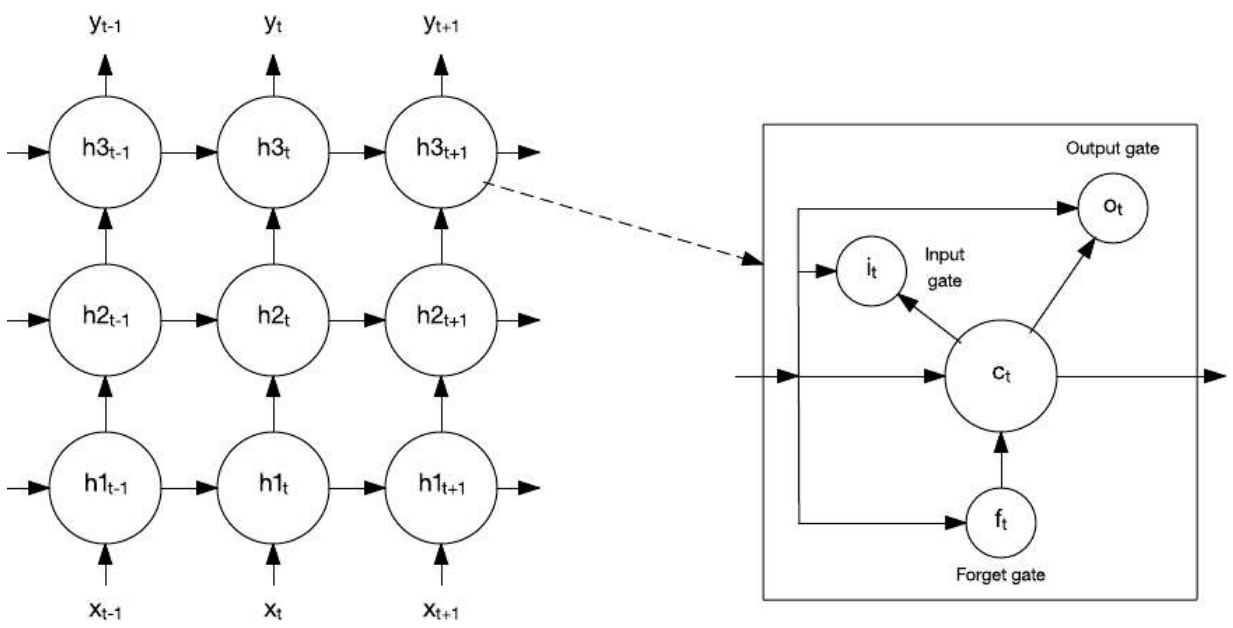
The use of deep layers of processing, convolutions, pooling, and a fully connected classification layer opened the door to various new applications of neural networks. In addition to image processing, the CNN has been successfully applied to video recognition and many tasks within natural language processing. CNNs have also been efficiently implemented within GPUs, greatly improving their performance.

Long short-term memory

Recall in the discussion of backpropagation that the network being trained was feedforward. In this architecture, users feed inputs into the network and propagate them forward through the hidden layers to the output layer. But, many other neural network topologies exist. One, which I investigate here, allows connections between nodes to form a directed cycle. These networks are called recurrent neural networks, and they can feed backwards to prior layers or to subsequent nodes within their layer. This property makes these networks ideal for time series data.

In 1997, a special kind of recurrent network was created called the long short-term memory (LSTM). The LSTM consists of memory cells that within a network remember values for a short or long time.

Figure 10. A long short-term memory network and memory cell



A memory cell contains gates that control how information flows into or out of the cell. The input gate controls when new information can flow into the memory. The forget gate controls how long an existing piece of information is retained. Finally, the output gate controls when the information contained in the cell is used in the output from the cell. The cell also contains weights that control each gate. The training algorithm, commonly backpropagation-through-time (a variant of backpropagation), optimizes these weights based on the resulting error.

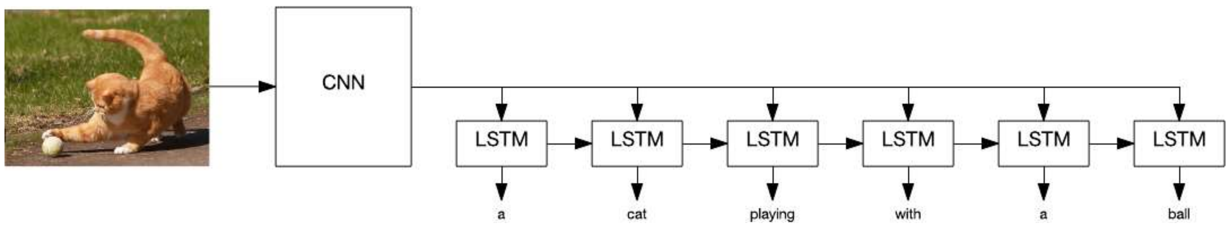
The LSTM has been applied to speech recognition, handwriting recognition, text-to-speech synthesis, image captioning, and various other tasks. I'll revisit the LSTM shortly.

Deep learning

Deep learning is a relatively new set of methods that’s changing machine learning in fundamental ways. Deep learning isn’t an algorithm, per se, but rather a family of algorithms that implement deep networks with unsupervised learning. These networks are so deep that new methods of computation, such as GPUs, are required to build them (in addition to clusters of compute nodes).

This article has explored two deep learning algorithms so far: CNNs and LSTMs. These algorithms have been combined to achieve several surprisingly intelligent tasks. As shown in the following figure, CNNs and LSTMs have been used to identify, and then describe in natural language a picture or video.

Figure 11. Combining convolutional neural networks and long short-term memory networks for image description



Deep learning algorithms have also been applied to facial recognition, identifying tuberculosis with 96 percent accuracy, self-driving vehicles, and many other complex problems.

However, despite the results of applying deep learning algorithms, problems exist that we have yet to solve. A recent application of deep learning to skin cancer detection found that the algorithm was more accurate than a board-certified dermatologist. But, where dermatologists could enumerate the factors that led to their diagnosis, there’s no way to identify which factors a deep learning program used in its classification. This is called deep learning’s black box problem.

Another application, called Deep Patient, was able to successfully predict disease given a patient’s medical records. The application proved to be considerably better at forecasting disease than physicians—even for schizophrenia, which is notoriously difficult to predict. So, even though the models work well, no one can reach into the massive neural networks to identify why.

Cognitive computing

AI and machine learning are filled with examples of biological inspiration. And, while early AI focused on the grand goals of building machines that mimicked the human brain, cognitive computing is working toward this goal.

Cognitive computing, building on neural networks and deep learning, is applying knowledge from cognitive science to build systems that simulate human thought processes. However, rather than focus on a singular set of technologies, cognitive computing covers several disciplines, including machine learning, natural language processing, vision, and human-computer interaction.

An example of cognitive computing is IBM Watson, which demonstrated state-of-the-art question-and-answer interactions on *Jeopardy!* but that IBM has since extended through a set of web services. These services expose application programming interfaces for visual recognition, speech-to-text, and text-to-speech function; language understanding and translation; and conversational engines to build powerful virtual agents.

Going further

This article covered just a fraction of AI's history and the latest in neural network and deep learning approaches. Although AI and machine learning have had their ups and downs, new approaches like deep learning and cognitive computing have significantly raised the bar in these disciplines. A conscious machine might still be out of reach, but systems that help improve people's lives are here today.

For more information on developing cognitive IoT solutions for anomaly detection by using deep learning, see Introducing deep learning and long-short term memory networks.

SOCIAL

CONTENTS

RESOURCES

IBM Watson services

IBM Bluemix

Related content

BLOG | OCT 17, 2018

Create an AI feedback loop with Continuous Relevancy Training in Watson Discovery

Artificial Intelligence | Knowledge Discovery | +

CONFERENCE

Girl Geek Conference

December 1, 2018

London

Artificial Intelligence | Data Science | +

CONFERENCE

OpenTech AI Summit Munich, Germany

September 26, 2018

Berlin

Analytics | API Management | +