Many of the most advanced and charismatic Big Data applications (like the recent IBM silicon-and-software Jeopardy quiz show contestant, "Watson") involve the mining of heterogeneous data sets for otherwise-obscured knowledge, patterns, and relationships. Unlike traditional data processing, Big Data discovery- and insight-oriented pursuits stress computational capabilities that are strengths of human cognition, such as classification and pattern recognition in noisy, imprecise environments, but are serious weaknesses of conventional digital computers and programming, which otherwise excel at exact numeric precision and binary logic.

This article describes different approaches by innovative researchers inspired by and intending to match or exceed human cognitive capabilities, applying advanced machine learning algorithms and techniques from the field of artificial intelligence, and even directly simulating the organization and functioning of the most complex and powerful associative and predictive Big Data engine in the world (so far!): the human brain’s neocortex.

Introduction

For IT management, the most interesting thing about Big Data isn’t that it’s big (though it is big, and growing exponentially; IBM estimates that 90% of the data in the world has been generated in the last 2 years⁴). Knowledge is what we seek, hidden in the complexity of the unprecedented volumes of unstructured data collected from every conceivable source, and it’s ours if we can only discover and expose the veins of authentic association, patterns, and meaning therein. IT organizations need to not only understand and facilitate the storage and analysis of business-relevant Big Data by their constituents, but also recognize that they are the collectors, stewards, and analyzers of IT-oriented Big Data, and become expert at exposing and exploiting the knowledge inherent in the Big Data they collect about IT infrastructure, users, services, transactions, data, and more to improve the management of IT.

Recognizing and extracting meaning from seemingly chaotic swarms of raw data is something at which the human brain excels. Digital computers excel at precise calculations of numbers and the manipulation of symbols. Like the growth of Big Data, computer performance is growing at an exponential rate – as it has for over 45 years, since a young Gordon Moore first noted his famous “circuit density doubling every two years” silicon scaling “law” in 1965². As of May, 2012³, the current fastest computer in the world, Japan's K computer, performs over 10.5 petaFLOPS (peta = 1,000 trillion; FLOPS = floating point operations – an add or multiply of two numbers, for instance – per second). Yet, when confronted with the chaos-taming mental tasks any five-year-old child effortlessly manages in her daily life (real-time processing of huge volumes of incredibly noisy sensory information, seemingly-instantaneous learning from myriad unstructured presentations of new experiences in arbitrary domains, and associative organization and recall of infinitely-variable memories), the most powerful computers in the world, running the best programs devised by
man, are almost embarrassingly slow and limited. The performance difference
is even more stark when one contrasts the K machine’s imposing physical
statistics (over 700,000 SPARC64 cores and 1.4 petabytes of RAM housed in a
multi-acre sprawl of 864 cabinets consuming over 12 megawatts of power) with
the human brain, occupying less than 2 liters of volume and burning a mere 20
watts.

Unfortunately, Big Data is, almost by definition, beyond the capacity or patience
of any person to manually peruse and analyze using conventional tools and
queries. If we want to decode Big Data and keep up with its exponential growth,
we need to teach computers to think more like humans. Researchers have been
studying and implementing “machine learning” algorithms for decades that can
achieve human-like learning from data to perform prediction, classification, and
association in narrowly-defined circumstances, and recent advances in available
computational power as well as the algorithms themselves are making these
techniques invaluable for extracting precious knowledge from mountains of raw
data. As a result, the field of machine learning, a branch of artificial intelligence
(AI), is becoming tightly associated with Big Data.

This paper will review major classes of machine learning strategies and try to
present approachable examples and explanations with a minimum of
mathematics. The first section will explore the definition of machine learning,
discuss the human-cognition-like capabilities it can produce, and list some of
the more notable recent application examples in the news. The next section
will describe some learning algorithms and the kinds of learning problems for
which they are well suited. The last section will focus on two exciting efforts to
directly model the organization and functioning of the human brain in software
and silicon, with the goal of dramatically broadening flexibility and capabilities
of artificial brains on the kinds of cognition tasks previously only performed with
any significant ease by humans, potentially powerful additions to the Big Data
toolkit.

**What is Machine Learning?**

As humans, we are intuitively familiar with learning. We learn continuously,
informally or formally, as we go about our lives. Learning involves the
acquisition and integration of new knowledge. Learning implies that one’s
performance is improved in some way by new understanding gained through
learning, enabling superior decision making or the exercise of new skills.
Learning may be accomplished by instruction, where an external expert or
source imparts knowledge, or by study, where one gathers knowledge through
observation, experimentation, or experience. Learning underlies our ability to
adapt to new or changing circumstances.

"Machine learning" generally adheres to the same principles as human learning,
but usually and explicitly within a particular problem domain and with
foundational mathematical rigor rarely present in the mental picture we have of
human learning. However, the end result is the same: a program constructed
applying machine learning principles improves the quality of its output
(according to metrics of importance to the application) with training, exposure
to past data, or through trial-and-error.

Machine learning is useful when we have a set of data that is characterized by
some unknown function (if there is no underlying order, however obscure, or if
the data is random, then there is nothing to discover and learn). Machine
learning can be used to discover and parameterize that function, creating a
means of predicting correct function outputs when presented with new, previously unseen data.

Aside from experiments specifically designed to model human learning to better understand our own brains and biological mechanisms, machine learning might be employed any time we want self-customizing programs that adapt to data sets or over time. Machine learning can also accomplish tasks that would be difficult to program explicitly due to data complexity and difficult-to-enumerate processing procedures. Big Data can come from sensors and sources of almost any and all types, from text to video, raw or organized, and many challenges associated with mining such complex aggregations of data are well-suited to machine learning. The kinds of undirected association and classification tasks we want to perform to extract knowledge and meaning from the flood of Big Data are strengths of machine learning algorithms.

**Notable Applications of Machine Learning**

Machine learning already has a positive impact on your life, and some futuristic applications examples have appeared in recent years based on new advances. Here are a few notable examples of today’s and tomorrow’s best:

- **Recommendation engines**: Machine learning algorithms are at the heart of recommendation engines for companies like Netflix. Netflix uses ratings and history of viewed videos by a subscriber to learn their tastes in entertainment and make recommendations. By offering new movies (new = not previously viewed by this user) that were highly rated by other users with similar tastes, Netflix improves the likelihood a recommended video will please the user. Part of the recommendation is a prediction of the rating the user will give the movie, should they choose to watch it. According to Netflix, 75% of the videos watched by an average user are chosen from the recommendations produced and presented by their engine, a strong indicator of its effectiveness. As a measure of the value of accurate recommendations, in 2011, Netflix awarded a million dollar prize to the winner of a competition to come up with a learning engine that could improve movie recommendations by 10%.4

- **Recognition systems**: Recognition and processing of free form and indistinct patterns in real-world circumstances is being accomplished by machine learning algorithms in applications like handwriting (e.g., United States Post Office address finalization), music (e.g., SoundHound music search), voice (e.g., Google voice search, Apple Siri), and vision (e.g., face recognition in digital cameras). These are applications that would be difficult to program in traditional ways given the variation and noisiness of real-world samples (background noise, lighting, orientation and alignment, etc.) and among the subjects themselves (no two faces, voices, or handwriting styles are identical, for example).

- **Control systems**: Machine learning is at the heart of recent dramatic examples of autonomous machines like Google’s self-driving cars or Boston Dynamics’ robots. These systems have an array of sensors that provide information about the orientation of the machine and position of its various components (like wheels, knees, feet, etc.), a set of motors and control actuators they can use to influence their position and orientation, and must learn the correct sequences of actions to successfully interact with and traverse their environment in widely varied and complex conditions. Boston Dynamics’ “Big Dog” quadruped can balance heavy loads and pack them across wild terrain unassisted, responding to changing traction, obstacles, and upsetting events along the way. Google’s autonomous cars have amassed
over 180,000 miles on public highways with minimal human assistance. A learning system inputs positional and spatial information from a laser range-finder and mapper, multiple radar units and cameras, and GPS to determine its position and to detect and properly react to pedestrians, other cars, and traffic control systems.5

• Informatics and data mining: Discovering patterns and correlations in unstructured data, finding relationships and dependencies, organizing information into associated structures, predicting future states or events, and detecting anomalies are all applications of machine learning in the field that is rapidly becoming generically known as “big data”. The most dramatic recent example of such a system is IBM’s “Watson”, constructed with the main goal of competing with humans answering quiz questions (stated as a question) on the television show “Jeopardy!” Like several systems listed in categories above, Watson does not use a single machine learning algorithm, but instead has several subsystems that use different machine learning strategies to interpret and process natural human language, analyze the questions (particularly difficult for Jeopardy!, as the writers take particular pride in cleverly wording clues to have subtle or multiple meanings), and arrive at the most likely answer. The contest takes place in real time and, like the human players, Watson must calculate the probability that it knows the correct answer before competitively pushing the buzzer button to be allowed to state it first, as there is a penalty for incorrect responses. While a quiz show contestant system may seem frivolous, Watson drove a set of developments that have legitimate real-world value, and IBM is now actively employing Watson-derivative systems in medicine, government, power grid management, and other application areas as part of its “Smart Planet” initiative.6

Slicing and Dicing Machine Learning

There are many machine learning algorithms and they vary greatly depending on the benefit or capability we are attempting to produce and the data or input we want the algorithm to learn. There is no “one-size-fits-all”. Andrew Ng, director of Stanford’s Artificial Intelligence Lab, states that the most important skill one needs to successfully apply machine learning is to be able to choose the appropriate algorithm for each particular application under consideration.7

The underlying purpose of many machine learning algorithms is prediction. Given some prior experience, training, or data set, the algorithm is able to infer an appropriate response to new situations or data. We can differentiate among machine learning algorithms by the nature of the predictions they make. The prediction may be an estimate, for example, producing the likely price a new item will garner in a market given a history of prices of other items sold in the past. In this case, the price estimate represents an answer chosen from a continuous range of possibilities. For historical reasons, this type of problem – trying to find a function that relates a set of input variables to an output variable – is called regression.

Alternately, the task may be one of classification, where the algorithm is asked to choose a discrete label or category that a particular set of data inputs represents, for example, whether a particular set of medical characteristics surrounding an unknown tissue lump likely indicate a malignant or a benign tumor.

Another way to sort machine learning algorithms is by whether the training phase is separated from the operational phase. A learning algorithm is said to be an off-line learning algorithm if the system is first trained with a set of
training data that establishes its decision parameters, and then applies those parameters to a second set of “real” data. Off-line systems only learn during the training session. On-line algorithms learn continuously from the data that is presented to them, even as they are asked to produce an output derived from it and evolve their decision parameters in response to changes in the data stream.

Machine learning algorithms also generally fall into two classes depending on whether the training data set includes the “correct” or desired answer (commonly called supervised learning) or not (unsupervised learning). Each record in the training data set for a supervised learning classifier algorithm might contain not only the data attributes for each training instance, but also the correct class for the instance. Unsupervised machine learning takes in unlabeled data inputs and is expected to discover associations or relationships among those data instances.

Somewhere between supervised and unsupervised learning is reinforcement learning. Reinforcement learning is primarily used in control systems that might take a sequence of steps over a period of time to arrive at a desired outcome. We may know what that outcome should be, but we may not know whether any particular step is a good one (advances us toward the goal) or a bad one. Like unsupervised learning, reinforcement learning systems are trained with unlabeled data (the “right answer” is not given), but are then “rewarded” for making positive progress or achieving a goal, or “punished” for poor progress or an undesirable outcome. “Positive progress” may be counter-intuitive if it is sometimes necessary to proceed in ways that temporarily seem wrong in order to avoid an obstacle, and reinforcement learning has a general “credit assignment problem” across all the steps in a sequence when evaluating how to apportion blame for an undesirable outcome to the intermediate steps that achieved it. It would be incorrect to assume, for example, that applying the brakes caused crashes in autonomous cars simply because every failed self-driving attempt terminated with the brakes applied as the control system tried to avoid the crash. Instead, blame (negative credit) should be levied to the actions that led, perhaps much earlier, to the crash.

Yet another interesting way to distinguish types of machine learning algorithms (and this should not be considered an exhaustive list) might be to look at their origins. Many useful algorithms have been developed with mathematical rigor and provable theoretical underpinnings from well-understood fields like probability and statistics. Others, acknowledging that organic brains are very efficient and proficient at pattern detection and recognition, classification and association, and many other complex and varied learning tasks, try to emulate the functioning and organization of components or even subsystems of the brain. The simple perceptron and artificial neural networks are two related examples of biology-inspired approaches, but there is also significant progress on understanding and replicating the brain’s hierarchical working, which this paper will discuss further later.

**Machine Learning Strategies for Big Data**

**Massaging the Data**

Before we look at a few commonly-used types of machine learning algorithms in a bit more depth to better understand how they work, when they might be used, and some strengths and limitations, let’s talk about key considerations when approaching a machine learning problem: input data feature selection and preconditioning.
As an example, consider a hypothetical text classification problem analogous to Google News, where we wish to automatically classify all discoverable on-line news stories as they appear on the Internet into the typical categories the New York Times (NYT) uses to organize its stories (e.g., “World”, “US”, “Politics”, “Business”, “Sports”, “Technology”, “Entertainment”, etc.) by finding and associating keywords in the articles with the output categories. We might be able to use the on-line archives of the NYT as a supervised training set, since it represents a massive store of articles that have already been classified by the editors. Our training set would be a large database of instances, each consisting of a news article (a text file, the input to the learning machine), and the category they were classified into (the correct answer). Each story is variable length and may contain thousands of words, where each word is a potential feature that the machine learning algorithm must consider when trying to identify correlations between the input data set and the desired output.

In this example, the raw input data (each instance article) contains a high fraction of “noise”, or words that have a very low probability of being keywords (for example, common words that are frequently used in every article, like “the”). There are several reasons why it would be a good idea to precondition the input data to remove as much noise as possible from the data without losing meaning. The more features we consider, the larger and more unwieldy many machine learning algorithms may become, multiplying computation requirements and threatening feasibility. Machine learning systems can learn to ignore the meaningless words but, if possible, it is better to reduce the input stream to those features that have a reasonable probability of contributing to the solution.

Feature selection and preconditioning of the input data can go far beyond just filtering. Sometimes, it may be a good idea to transform the data into a different form to permit use of simpler or less costly algorithms without running into their inherent limitations. An example of this is discussed in more detail later. The main point here is that great performance and tractability gains can be made by the machine learning practitioner’s savvy choices and treatment of input features.

**Linear Regression**

This class of supervised machine learning algorithms works on labeled training set data (the correct answer is supplied with each instance’s inputs). The goal of the algorithm is to learn the best parameters of a linear function (the hypothesis) that maps new inputs to a continuously-valued output (a real number). The estimate is possible because the hypothesis function is a line (if the data has a single feature), a plane (if the data has two features), or a hyper-plane (if the data has more than two features) fitted to the training data that minimizes the total error across the data. The input data may have any number of features, but it is easiest to illustrate 1 or 2 features without having to try diagramming higher-dimension hyper-planes. After training, outputs for new sets of inputs are calculated by simply indexing into the line/plane/hyper-plane using the instance feature values.

A simple example (see Figure 1) is predicting the selling price of a particular model of used automobile. Imagine we have a historical data set with one feature, the odometer mileage of the vehicle, and the actual selling price it garnered on the market. Linear regression allows us to use the historical data to discover the parameters of the hypothesis function that generates a line that best approximates the data. We can predict future selling prices of this model...
of car to be sold by inputting the odometer mileage and reading out the projected selling price. This example can be refined and made more sophisticated by adding additional features (e.g., age of the vehicle, size of the engine, presence of various options, etc.) to our training set.

Training involves finding the parameters of the fitted linear approximation that produce the smallest error between the training set and the approximation. A cost function computes the summed error of the training instances as the parameters of the approximation function are varied. In linear regression, a least mean squares cost function is often used that measures the sum of the distances from the training instances to the fitted hypothesis. If we could plot the shape of the cost function across the range of all possible parameters of the hypothesis (potentially an infinitely large task), we would paint a picture of a solution landscape with as many dimensions as there are parameters. The optimal (least cost) solution would be found at the lowest point in that landscape. Our learning challenge is to locate it (or a reasonable approximation) without having to do infinite calculations.

Linear regression can learn in different ways. One way is to initialize the parameters of the hypothesis model with random values, utilize an iterative gradient descent algorithm that determines the cost (error) at the current step values and the direction of slope toward the nearest least-cost point, and then adjust the parameter values to step “down” the cost function curve in that direction by an amount proportional to the steepness of the slope. In this way, the iterative algorithm slows its learning rate as it converges on the minimum until it reaches a stable point at zero slope (see Figure 2).

If we run through all the training instances to calculate the error for each step of the gradient descent algorithm, this is called batch gradient descent. An alternative is to adjust the parameters according to the gradient of the error for each training set instance in turn, a variation called stochastic or incremental gradient descent. When the number of training set instances is very large, stochastic gradient descent can make immediate progress toward a solution and get “close” to the minimum faster than batch gradient descent, though it may never exactly converge at the minimum (instead, oscillating back and forth across the minimum as individual instances continue to be examined). In practice, however, the “close” solutions may be close enough and the speed of arriving at that approximate solution more valuable than achieving the exact minimum.
Linear regression also offers an efficient learning “cheat”, however, as it is often possible to use matrix calculus to explicitly calculate the minimized total error and the resulting hypothesis function parameters in a single step.

The quality of the estimates delivered by linear regression depends on several factors, but one issue is the bias of the linear model: fitting a linear function of a particular order to the data when the data may fundamentally require a more complex model (e.g., a higher-order polynomial curve) to achieve a reasonable total error. When the data isn’t very well represented by the model, it’s called underfitting; it is equally possible to overfit a high-order polynomial to a small training set and also do poorly on real-world data.

The Perceptron

The perceptron is our first biologically-inspired machine learning strategy. In the 1960s, it was advanced as an approximate model of how an individual neuron in the brain functions. As it became better understood, it’s value as a learning solution, as well as its approximation of a biological neuron, were shown to be much more limited than its initial proponents believed; however, it has a very simple learning algorithm, remains good for binary classification, and can learn in an online mode even as it makes predictions.

Visually, the perceptron can be diagrammed as a set of input nodes, one for each feature (see Figure 3). Each input has an associated weight value that is multiplied by the input. The results of the weight vector multiplication with the input vector are summed together in the “body” of the perceptron and subjected to a threshold function that outputs a 1 if the sum is greater than or equal to zero or a -1 if the sum is less than zero. (You can perhaps see the resemblance to a biological neuron, with weighted input axons summed together in the neuron body to decide whether the neuron will fire or not.)

Perceptrons require supervised learning. Learning is accomplished by adjusting the weights of the inputs according to the direction of the error at the output, by multiplying each input value by the incorrect output (-1, 1) and adding it to the weight. Training (adjusting the weights) only occurs when the perceptron output does not agree with the training set answer (1 or -1). When learning a training set offline, we may iterate through the training set many times, hopefully to converge on an acceptable set of weights.

Perceptrons can classify linearly separable data in any number of dimensions, that is, it can draw a line/plane/hyper-plane between two classes of data instances, and the algorithm will converge as long as the data is linearly separable (see Figure 4). Even if the data is not entirely separable (see Figure 5), a modification of the learning algorithm called the “pocket” algorithm, which calculates total error as training iterations progress and retains the weight settings resulting in the least error over the entire set, can be used to classify noisy data sets with some degree of error (mis-classified instances).
Earlier, I stated that input data transformation could provide ways to work around the limitations of certain machine learning mechanisms. With the linearly separable data limitation of the perceptron in mind, consider the classification problem illustrated in Figure 6a. Note that the two classes cannot be linearly separated, but this does not mean that we cannot perform a non-linear preconditioning processing step on the data by finding the centroid of the data (Figure 6b) and using each instance’s distance in each feature dimension to the centroid (Figure 6c) as the features we will classify. By performing the non-linear transformation on the data set prior to training (and presentation of new data), we are able to use the simple and computationally inexpensive perceptron to learn to classify a problem set that, in raw form, would be beyond its capability.
K-Means

Clustering algorithms are a way of discovering underlying organizing principles in unlabeled data, so they represent an unsupervised learning mechanism. K-means is a core algorithm for many clustering strategies that operates by calculating the Euclidean distance (in multi-dimensional “feature space”) from every instance to each of k cluster centroids and associating the instances with the centroid to which they are nearest. Initially, the number of clusters to be sought (k) is chosen by the user (or sometimes algorithmically, when k-means is part of a more complex organizing algorithm) and the initial placement of the k centroids in feature space is chosen at random. Each iteration, after associating all instances to the closest centroid, k-means calculates a new position for the centroid based on the mean “location” of the associated instances in feature space. Eventually, cluster membership and the locations of the centroids stabilize, and the iteration is complete.

Note that clustering algorithms start with unlabeled instance data (the “right” answers – in the case of clustering, the natural groupings of data -- are not given) with any number of features. Figure 7a illustrates such a data set plotted using only two feature dimensions. Figure 7b illustrates k=2 after running k-means, and Figure 7c shows k=3. Since the number (or even genuine existence) of clusters of meaning to the data is unknown, the choosing of k is exploratory, and the efficacy of the clustering degree is only revealed by examination of the resulting organization structure.

Sophisticated clustering algorithms have many variations. Some permit data set instances to be members of more than one cluster, sometimes with probabilistic membership. There are algorithms to build cluster hierarchies (trees) by recursively subdividing clusters, and also by building from the leaf nodes of trees toward larger and larger class generalizations.

Once a valid cluster hierarchy has been created, it is straightforward to walk the tree, extracting the emergent rules about the instance features that governed assignment of cluster membership at the nodes. These rules can then be used to reason about the data and perform assignments of new instances to their proper place in the discovered taxonomy.
K-means is very sensitive to initial choices of “seed” locations of the centroids and can easily settle into local minima. Finding globally optimal clusters may be very difficult. To increase the odds of success, it is common to run the algorithm multiple times with different seeds and choose the results with the least total squared distance of the instances to the centroids. Used skillfully, k-means is a powerful mechanism for exposing order in unlabeled data sets.

**Artificial Neural Networks**

The perceptron was first invented in 1957 by Frank Rosenblatt and was actively and enthusiastically studied until its limitations were formally proven in 1969 by Seymour Papert and prestigious artificial intelligence pioneer Marvin Minski. Though it was shown shortly thereafter that networks of perceptrons could transcend those limitations (e.g., the inability to learn the Boolean XOR – exclusive OR – function), it took a decade or more for the pall Minski and Papert’s work had cast on perceptrons to lift and artificial neural networks composed of multiple layers of multiple perceptron-like constructs to be explored and characterized. Today, they are commonly applied in a variety of applications, including control of robots, financial applications (e.g., trading, review/approval of credit), pattern recognition tasks, medical diagnosis, and knowledge discovery (data mining).

Recall the non-linearly-separable data set of Figure 6a, which required a non-linear transformation to make the problem tractable for a perceptron. Artificial neural networks are capable of directly learning such a classification problem set without massaging the supervised learning data set, thanks to their more complex architecture and the development of a learning algorithm that could propagate output error back through the network to train so-called “hidden layers” (layers not connected to inputs, nor generating the final network output) of perceptrons.

It’s worth noting that it is the connections among neurons (represented by the network graph and the weights) that accomplish storage of the learned patterns and are the distributed memory of the system. This attribute confers inherent resiliency and scalability (depending on the implementation) on neural networks and is such a defining property that many neural network researchers refer to the general field as “connectionism”.

The perceptron used for neural networks is somewhat modified. Instead of a simple post-summation threshold, we require a continuous (differentiable) function in order to compute the deltas between the actual output and the desired output, and to calculate an apportionment of that delta to be used to adjust the weights of intermediate layers of perceptrons. The so-called “backpropagation” learning algorithm has two phases, the first a feed-forward propagation of the training set inputs flowing through the layers of perceptrons to calculate the activation of the final output(s), and the backward propagation of error and resulting weight adjustments. While the artificial neural network is biologically-inspired and provides analogous recognition and classification capabilities, there is no biological equivalent of the backpropagation learning algorithm. Artificial neural networks only approximate a subset of the functions of real biological networks of neurons, and their operation is not equivalent.

The backpropagation learning algorithm is somewhat computationally intense, in part due to relatively slow convergence requiring large iteration counts, and in part due to the multiplication of complexity represented by the increased scale of an artificial neural network over a single perceptron. Artificial neural networks are frequently applied to problems with very large numbers of
features, such as the pixels in a set of images, multiplying the dimensions of the network required to accomplish the task and exacerbating the amount of time spent in the learning cycle.

Organizationally, the network of perceptrons typically connects all inputs to all perceptrons in the input layer. Subsequent layers may have more, the same, or fewer perceptrons than the input layer. The final output layer may have one or more perceptrons, as required to encode the desired output activation pattern(s) (see Figure 8).

![Diagram of a neural network](image)

**Simulating Human Cognition**

**Emulation vs. Simulation**

There is a spectrum of philosophies about how closely AI should resemble and copy natural intelligence. At one end, we have AI researchers that care little about the biological underpinnings of cognition, who only wish to understand its fundamental logical and mathematical foundation and are happy to have the structure and operation of their programs bear no resemblance to nervous systems or brains. At the core, they believe that properly written programs running on conventional digital computers can efficiently produce the most important capabilities of biological systems without slavishly copying them down to the cellular level. At the other end are cross-disciplinary scientists that are straddling computer science and neurobiology, who believe that biological cognition has much to teach us about how traits like learning, memory, inference, and prediction can be emergent properties of extreme numbers of simple, massively interconnected neurons, and that studying and simulating biological models is, in fact, the best path to understanding and creating effective models (and even non-von Neumann hardware) for artificial systems too.

We have discussed a few examples of machine learning mechanisms designed to learn to perform recognition, association, classification, and prediction tasks directly from the data (with or without supervisory help). The organization and functioning of two of them -- perceptrons and artificial neural networks -- were inspired by biology, but they by no means represent a serious effort to reverse-engineer the actual functioning of a brain. No claims of biological verisimilitude are made. Rather, these devices are intended to only approximate certain limited aspects of biological subsystems, like the perceptron’s summing of weighted axons and the artificial neural network’s connection-oriented distributed representation of knowledge in the strengths of connections between multiple layers.

As has been noted, some of these techniques date back more than 50 years.
Much has since been learned about the functioning and organization of biological cognitive systems. Concurrently, available computational power has steadily increased (if one can describe exponential growth as “steady”), making more refined simulation of the functioning and organization of significant portions of biological brains an enticing possibility.

Two organizations at opposite ends of the scale spectrum are good examples of multidisciplinary research applying advancing neurological understanding to the simulation of biological cognition. The first is a small start-up, Numenta, and the second is enterprise computing giant IBM.

Numenta’s Hierarchical Temporal Memory

Numenta co-founder Jeff Hawkins (previous co-founder of Palm and Handspring) and his team have recently entered a limited beta period for their cloud-based service called “Grok”, which learns the structure and organization of a provided data stream, makes predictions about future events, and detects anomalies. They call the underlying theory and architecture of their software Hierarchical Temporal Memory (HTM), which acknowledges two notable biologically-inspired aspects of their design (inspired from studies of the neocortex, the highest-order thinking layer of mammalian brains), that the operation and organization of the cortex is hierarchical, and that it mostly operates on time-varying sequences of signals13,14.

![Figure 9](image)

Neurons (cells, in Numenta’s lexicon) in HTM are much more accurately modeled than in traditional neural networks (see Figure 9), possessing synapses (the connections between tips of the branches of one neuron’s output axons and the inputs of the next) that bridge to a single proximal and multiple distal dendrites (heavily branching input integrators that connect to the output axons of other cells). The action potential of the output is directed along the axon in two directions, to make connections to many local cells, and to make connections with distant cells.

Emulating the architecture found in mammalian cortices, the cells are organized into a sheet of neuron columns, with multiple cells per column, that participate in joint activations when they recognize a pattern on their inputs. Regions of the artificial cortex (see Figure 10) contain large numbers of columns (and even larger numbers of cells) and, while multiple columns are “wired” to each bit from the input data stream, the strongest activating columns of cells in the region inhibit their neighbors, helping to establish a sparse pattern of
activations (and a corresponding sparse distributed representation) for any input pattern or sequence that is “memorized”.

Cells can be in one of three states: predictive, active from feed-forward, and inactive. The predictive state is primarily a result of lateral connections to cells in other columns and training reinforces a connection between two neurons if the “pre-synaptic” (upstream) neuron activated just before the current time step. This is a temporally-oriented version of the “Hebbian learning rule”, which roughly states that “neurons that fire together, wire together” - an observation derived from neurology. At a future time, upon seeing a similar pattern in the neighboring neurons, the trained neuron would transition into a predictive state, indicating it has seen this pattern before. If the prediction is correct, those cells then become activated.

Learning is accomplished by adjusting “permanence” weights at each synapse, modifying the durability of the connection rather than the strength. Two processes, called the temporal pooler and the spatial pooler, are responsible for reinforcing synapse connections among cells in a region, or between feed-forward inputs and the cells in the columns. Learning is online and unsupervised, so the HTM model could be used for classification, prediction, or discovery and organization of meaning from seemingly random data.

Numenta’s HTM architecture and cortical learning algorithm go far beyond neural networks in biological veracity, mirroring the latest discoveries and information about the organization and apparent functioning of the layers of the brain responsible for making sense of the world, but are still an abstraction of the signaling, responses, and learning adaptation of the biological model. Numenta says they want to use their technology to challenge the usual offline mode of data mining (first, accumulate data in a data warehouse, then later run analytical algorithms on the stored data to discover and extract knowledge). Noting that the value of much data is ephemeral (e.g., market data that might indicate a buy or sell), they propose that their online learning capability to work with live streams of data can not only obviate some of the need for data warehouse storage but also provide a real-time competitive advantage by being able to immediately deliver insight and predictions derived from streams of data almost as fast as they are delivered.
IBM’s Cognitive Computing Initiative

Dharmendra Modha, manager of “cognitive computing” at IBM Research-Almaden in San Jose, California, has described their biologically-inspired cognition research and simulation efforts as being particularly applicable to the analysis and understanding of the deluge of data from myriad sources that constitutes Big Data.  

IBM’s cognitive computing program (sponsored by DARPA as part of its Systems of Neuromorphic Adaptive Plastic Scalable Electronics – SyNAPSE -- program) is taking a biological modeling abstraction to new levels of detail: the spiking electric impulse communications behavior of real neuron activation potentials as they are generated and propagate in the brain. At the same time, they are pushing cognitive simulations to extreme scales and even designing and producing neuromorphic silicon (mimicking the design and functioning of the biological systems in the brain).

Using (and sometimes performing) some of the latest physiological mappings of real mammalian brains, IBM researchers have created detailed models of neurons (how they communicate and learn, an architecture and strategy for organizing them in patterns and layers reflecting those found in biology, and so on) and embodied them in a very large scale simulator called C2. Like Numenta, IBM is focused on the columnar organization of neurons and their connectivity in the half-dozen layers of the cortex, but IBM also includes a couple layers of the thalamus, a “gray matter” sub-cortical region in the brain that distributes sensory signals from the body into the cortex and between regions in the cortex.

IBM’s simulated neurons are even more detailed and exacting in their reproduction of the spiking nature of the signaling among biological neurons than Numenta’s. IBM uses a Hebbian-variant learning algorithm called spike-timing dependent plasticity (STDP) to reinforce or dissolve synapse connections between neurons. STDP observes that the synapse connection between two neurons is particularly strengthened when the pre-synaptic (sending) neuron spike happens just prior to the post-synaptic (receiving) neuron activation spike. In C2 (and in their 256-neuron silicon chip), synaptic weights are increased or reduced in response to the timing and propagation of spikes down realistically-delayed axon and dendrite branches and integrators.

In 2009, IBM teamed up with supercomputing scientists at the Department of Energy’s Lawrence Livermore National Laboratory (LLNL) to run an extreme scale configuration of the C2 simulator on LLNL’s IBM-produced Dawn Blue Gene/P supercomputer. Using over 3 million core-hours of capacity, C2 saturated 147,456 CPUs and 144 terabytes of memory to simulate 1.6 billion neurons connected by almost 9 trillion synapses. For scale, this is more than the estimated number of neurons and synapses in the cerebral cortex of a cat and approximately 4.5% of the population of neurons and synapses in the average human cortex. It must be noted that the simulation could not run in real time; time steps proceeded at a rate over 640 times slower than the normal functioning of a real biological brain.

Working with silicon chip designers, the IBM researchers have also designed and fabricated neuromorphic integrated circuits with 256 neurons that exactly replicate the functioning of the C2 simulator, only in real time (potentially faster). While the large-scale C2 software simulations on LLNL’s supercomputers were at a truly impressive scale, the neuromorphic chip is remarkable for its extreme low power (several orders of magnitude lower than the C2 simulator per neuron) at real-time speed. The architecture of the
neuromorphic chip distributes all memory in a very fine-grain way among the neurons and does not resemble a conventional processor in almost any way (there isn’t even a clock; the circuits are event-driven), thus eliminating the bottlenecks imposed by conventional von Neumann architecture between processors and memory, and achieving a 3 orders-of-magnitude speed-up over the very fast supercomputer processors on the same C2 simulation tasks. For very large-scale simulations like this to have practical value, neuromorphic integrated circuits will be required. All the C2 runs at scale on the LLNL supercomputer produced less than 2 minutes of simulated “cat brain time”.

Conclusions

Big Data is all about deriving insight, understanding, and knowledge from the vast amounts of data now being routinely collected from myriad customer, business, and public sources. Often, the understanding of these complex and immense streams of data defies traditional database-oriented queries and analysis, requiring that we learn the pertinent characteristics and infer underlying order beneath the data from the data itself.

Machine learning and cognition is a powerful and essential class of tools for accomplishing many of the mental gymnastics associated with understanding Big Data: discovery of relationships and patterns in the data, predicting new outcomes based on dynamic analysis of time-series records, classifying information and inferring when a set of patterns constitutes a particular opportunity, situation, or diagnosis, adaptively learning from a “warehouse-free” real-time stream of data to almost instantly forecast future trends and detect anomalies, and more. Machine learning techniques range from top-down algorithms firmly backed by statistics/probability theory and mathematical rigor to amazingly detailed simulations of the neurological processing elements and physiological connectivity of our own brains.

It is an exciting and daunting time for Big Data, machine learning, and cognition research, as both the problems and the solutions are mind-bogglingly complex and evolving at exponential rates. Though the field of machine learning is over half a century old, the pace of change and capability is clearly linked to Moore’s law. It’s certainly safe to predict that machine learning and cognition, and the understanding of Big Data they enable, are going to profoundly alter our lives in many ways over the next 50 years.

References

Magazine, 31 (2010)