Machines That Learn from Hints

Machine learning improves significantly by taking advantage of information available from intelligent hints

by Yaser S. Abu-Mostafa

"Why is an elephant big, dark and strangely shaped?" the question goes. "Because if it was small, white and round, it would be an aspirin." This answer may ring funny to human ears, but it could well prove informative to a computer trying to identify such objects as elephants or aspirin. Knowledge we commonly take for granted is not available to machines unless carefully spelled out. For machines, learning is not at all simple.

Despite the challenges, machine learning is one of the fastest-growing technologies today. The past few years have witnessed an explosion of applications, ranging from automated reading of handwritten zip codes at the post office to predicting seat demand in the airline industry. Indeed, the last time you received a credit card from a bank, chances are it was approved by a machine that learned on its own how to evaluate credit risk. And the future of machine learning is on the rise.

Designing a computer program to handle a particular job almost invariably demands a thorough understanding of that task and its solution. Machine learning therefore has a fundamental appeal. Instead of devising a specialized program, one could merely provide training examples to a versatile machine that would learn on its own.

A self-learning credit-card approval system would, for instance, use historical data about "good" and "bad" customers to judge applicants. The machine does not care about the details of the problem. All it does is take matched pairs of inputs (in this case, personal information) and outputs (credit behavior) and absorb whatever information their relation contains. The trained machine then serves to evaluate new applicants. This kind of procedure takes automation one step further than normally envisaged. It not only applies a computer to a repetitive task, it automates the very problem of designing a system to perform that task.

One can, in principle, apply the methodology of machine learning to a wide array of problems. If, however, the input-output examples available lack vital information, the machine may fail to acquire proficiency. Fortunately, one can often append the needed information in the form of an intelligent hint. The hints used in machine learning range from simple observations to sophisticated knowledge.

In computer-vision applications, for instance, in which the goal is to recognize objects, there are many invariance hints. These assert that an object remains the same object when it shifts position in the range of view or changes in size. In financial-market applications, there are many monotonicity hints, which state that if an input consistently shifts in one sense or direction, the output must also consistently move just one way. Each particular application has its own hints that can aid the learning process.

If one knows enough about a given application to offer hints, why bother with machine learning in the first place? Why not employ this knowledge to design a specialized machine for the job? In some instances one can do so, but the fact of the matter is that usually too little is known about a problem to specify a method for its solution according to a well-defined set of rules.

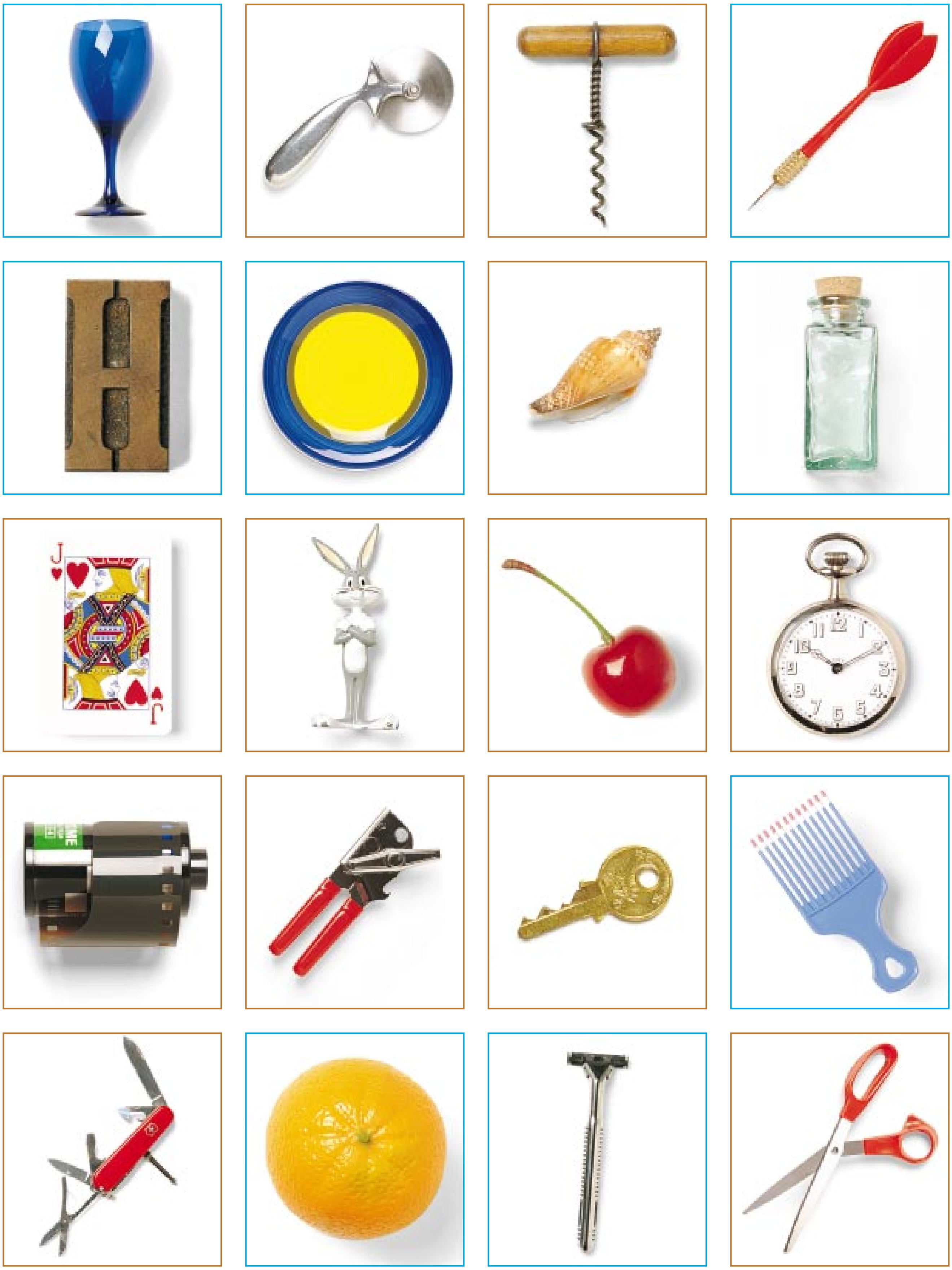
Applications range between two extremes: structured problems that are totally defined and require no examples, and random problems that are completely undefined and depend entirely on training examples for their solution. Machine learning using intelligent hints is the way to handle the vast middle ground.

Machine-Learning Paradigm

How do machines learn? Many different models for machine learning have been devised. Typically the implementation used will have a general structure that is broadly tailored to the problem, but it will also have many free parameters—these might be thought of as the knobs and dials for tuning the machine. The values given to these adjustments determine how the machine will ultimately act; different settings will produce completely different results.

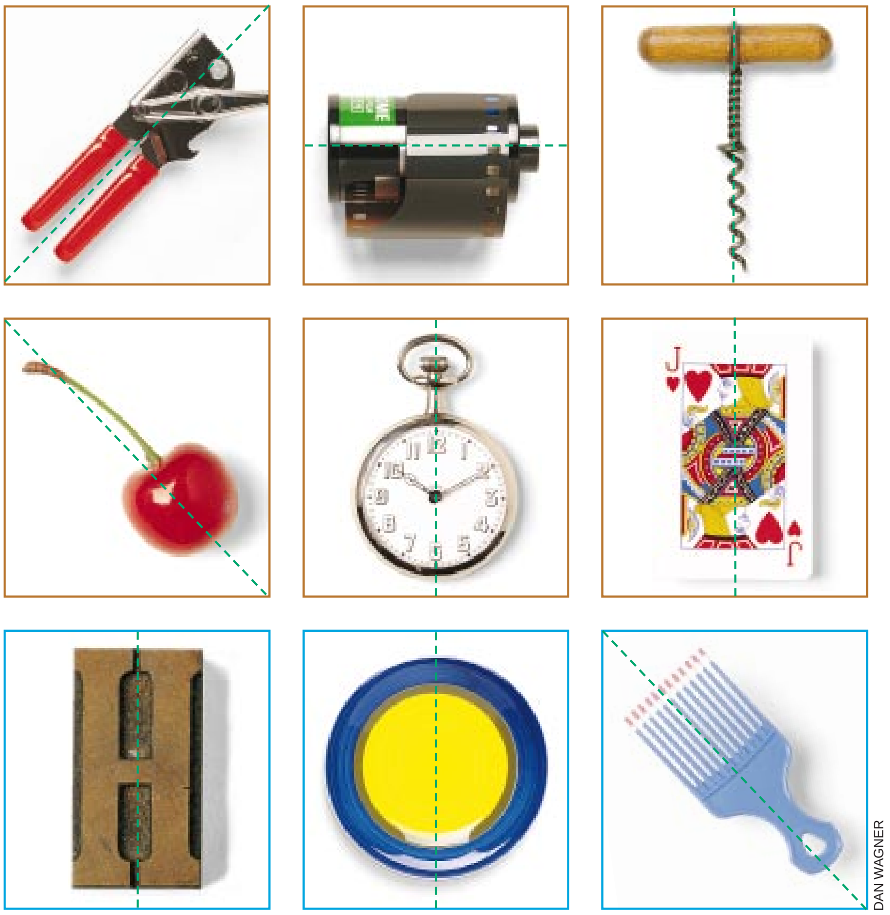
The behavior of a machine can be viewed mathematically as a function that associates input values (the specifics of a problem to be solved) with corresponding output values (the decision or action to be made). The goal in machine learning is to make the machine emulate the target function, the desired mapping of inputs to outputs. We can use training examples from the target function to guide the selection of values for the machine's free parameters. With each example, the machine refines its internal settings so that it matches the

YASER S. ABU-MOSTAFA is professor of electrical engineering and computer science at the California Institute of Technology and chairman of NeuroDollars, a California-based corporation. He received a B.Sc. from Cairo University in 1979, an M.S.E.E. from the Georgia Institute of Technology in 1981 and a Ph.D. from Caltech in 1983, where he now heads the Learning Systems Group and serves as one of the principal investigators for the National Science Foundation Center for Neuromorphic Systems Engineering. Abu-Mostafa has been a technical consultant for Citibank since 1988. He has published widely in the areas of learning theory, neural networks, pattern recognition, information theory and computational complexity.

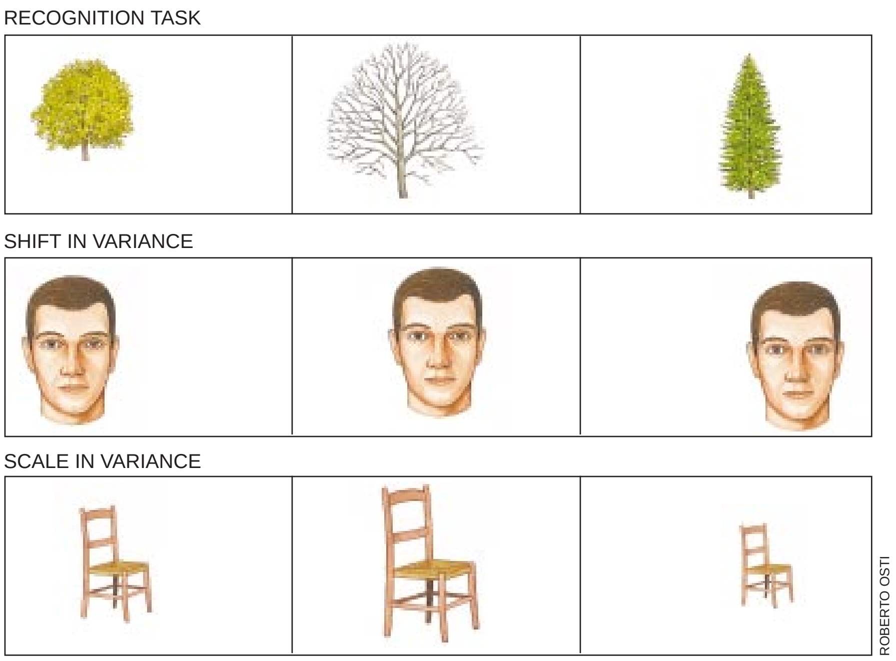


DAN WAGNER

CAN YOU SOLVE IT? These objects have been sorted into two classes, indicated by either a blue or brown border. Which characteristic distinguishes them? Computers programmed to learn from examples often face similar puzzles. Providing the machine with hints can make learning faster and easier. For a hint to help with this puzzle, turn the page.



VISUAL HINT aids both machines and people in solving the puzzle. Drawing the axis makes it clear that the top six objects lack the mirror-image symmetry exhibited by the bottom three. This characteristic distinguishes brown and blue categories.



INVARIANCE HINTS can help machines recognize that objects do not lose their identity when viewed in a new way. A machine attempting to identify trees, for instance, would not inherently know that size and position did not matter (*top*). Training on "virtual" examples of quite different subjects—such as a face or a chair—could prompt the machine to grasp these principles.

inputs and outputs appropriately. When the machine reaches a setting that corresponds as closely as possible to the target function, it will have in effect "learned" it. Machine learning is simply the search for the right positions for the knobs. Because the search is guided by the training examples, this paradigm is called, naturally enough, learning from examples.

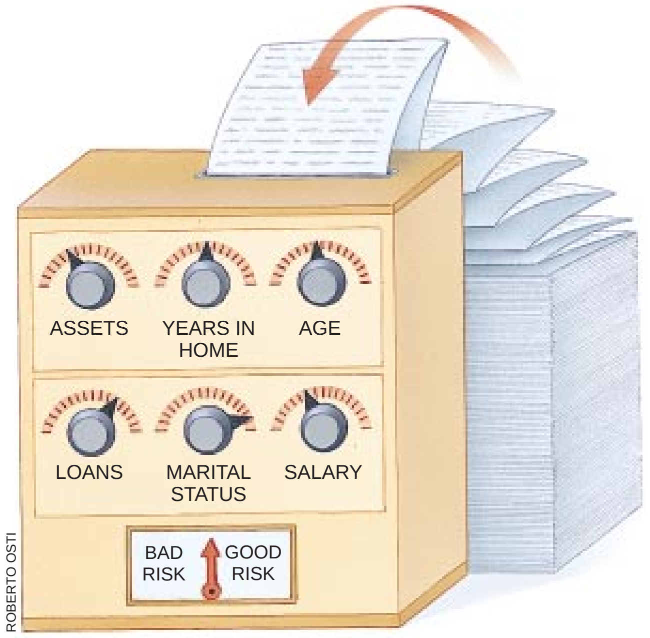
The most widely applied form of such machine learning is the neural network [see "How Neural Networks Learn from Experience," by Geoffrey E. Hinton; SCIENTIFIC AMERICAN, September 1992]. Neural networks were inspired by the power of real neurobiological systems. They consist of many computational elements interconnected in such a way that each element's output reflects inputs from a number of other elements. The adjustable parameters of a neural network are called synaptic weights after their biological counterparts, the synapses that connect nerve cells in the brain. The flexibility of neural networks and the simplicity of their training have made them the machine-learning model of choice for the past 10 years; neural networks now find uses in a broad range of machine-learning applications. Although specialized electronic and even optical networks have been built [see "Optical Neural Computers," by Yaser S. Abu-Mostafa and Demetri Psaltis; SCIENTIFIC AMERICAN, March 1987], in most cases, one implements a neural network simply as a program running on a personal computer or workstation.

With all the training required, we might imagine the need for tedious late-night sessions at the computer, supervising the machine as it learns. Fortunately, responsibility for finding the optimal adjustments usually falls on a learning algorithm, a method that reduces the process to a series of simple, repetitive steps that the computer can perform independently. One of the most common learning systems in use today is the back-propagation algorithm for training neural networks. This technique was popularized primarily by David E. Rumelhart while at the University of California at San Diego.

Back-propagation uses simple calculus to decide how to change the parameters of the neural network. It takes a training example—an input and its corresponding output—and makes small modifications to the network parameters to minimize the difference between the current response of the network and the target response. This step is repeated over and over, each time nudging the network a bit closer to the desired effect. After going through all the examples several times, the network can replicate the target function reasonably well.

Let us see how this might work for credit-card approval. An input-output example in this case would be the data supplied by an applicant (such as age, salary and marital status) and her eventual credit performance (whether the bank profited or lost in extending credit). A neural network that approves credit cards would ideally predict a person's later behavior just by looking at the data provided by the individual. To learn this function, the network takes thousands of cases of real people's applications and credit behavior, and it keeps modifying its internal parameters in an attempt to match its output with these historical records.

Some of the variable parameters may be concerned with the salary and would move the network toward approving people who earn more. Others may be concerned with a combination of age or marital status, favoring particular combinations of these attributes. Eventually, a final setting is reached that makes the network produce the right response with as many training examples as possible. Now the network can be used to grade a fresh application by extrapolating from its "experience" to predict how the new customer will behave with credit.



MACHINE LEARNING involves adjusting a system's internal parameters such that it makes the proper associations between data inputs and desired outputs. A credit-approval system, for example, would be trained to link applicants' personal data with their known credit behavior. In effect, the learning process "tunes the dials" until the machine can duplicate the input-output relations in the training examples.

Learning with Intelligent Hints

Whether implemented as a neural network or in some other way, all machine-learning methods share this same fundamental premise of learning from examples. For the machine to learn successfully, it must be able to generalize from the limited input-output samples on which it was trained. Do the training examples convey enough information for the machine to respond properly to novel inputs? Perhaps not. Because the machine does well on the training examples does not necessarily imply it will do equally well on something it has never before encountered.

Remember that a machine knows nothing about the function it is trying to learn except what it sees in the training data. If the data are deficient—there may be too few training examples or too much irrelevant information contained in them—the machine will not generalize properly. Or the examples may not encompass all the important information. Suppose, for instance, that I want to train a machine-vision system to recognize trees. I may not be able to specify how to identify a tree in exact mathematical terms, so I cannot structure the problem and give the machine rigid rules to apply. If I simply show the machine pictures of trees and objects that are not trees, I am giving it information, but I am still not telling it everything I know. For instance, I know that a tree remains a tree if it is shifted a little or rescaled. People intuitively realize that much, but the machine does not—unless specifically told so. Without hints, the machine might take a very long time, if ever, to reach that form of "understanding."

Even the simplest hint can boost learning. As in a game of 20 questions, in which the answers to some elementary questions can narrow the search significantly, a few hints may make the difference between learning a function and not learning it at all. To take advantage of this situation, I introduced a formalism called learning from hints some six years ago, and it has since become a feature of many learning systems. The most notable achievements of this approach are in automated trading systems for financial markets and in systems for handwritten character recognition.

The credit-card-application problem can also benefit from intelligent hints. Admittedly, it is difficult to define exactly what makes a person a good credit risk, but one hint is obvious: if two people are identical except that one earns less money than the other, and the machine approves credit for the lower-paid person, it must also approve credit for the higher-paid one. This is one of many possible monotonicity hints. While the machine is learning, it should set its free parameters in such a way that it matches inputs and outputs according to the target function but simultaneously satisfies such hints.

One application of learning from hints that my colleagues and I conducted in our Learning Systems Group at the California Institute of Technology is in the area of foreign-exchange trading. We ran a machine-learning experiment to forecast the exchange rates for the U.S. dollar against four major foreign currencies. We wanted to test whether the resulting trading system would be more profitable when we injected a commonsense hint into the learning process. The hint we used reflected a symmetry that was obvious to us: if a given pattern in the price history implies a certain movement (up or down) in U.S. dollars compared with a foreign currency, the foreign currency should move the same way if that pattern emerges in its own price history. The results of the experiment were quite successful. In all four markets, the symmetry hint brought the system a consistent increase in profit.

To assure that the improvement in our neural-network program came from information contained in the symmetry hint, we tried to fool the machine with two alternatives. The first was an uninformative hint, giving the machine random pieces of information. To our satisfaction, the machine did not benefit. Performance was about the same as when there was no hint at all. Next we fed the machine a hint that provided deliberately erroneous information. Performance then deteriorated rapidly, as would be expected. The intelligent hint had truly helped.

Implementing the Hints

The main challenge to using hints in machine learning is in automating the process. Hints come in various guises; they range in character from subtle to glaringly obvious. How can one algorithm orchestrate learning from all the varied pieces of information encompassed by such hints?

First, all the representations of hints must be standardized to enable the learning algorithm to deal with them on equal footing. The clue for the proper way to accomplish this end comes from the representation of the target function itself: as input-output examples. What one is telling the machine is, "When we input so and so, you should output such and such." The identity of the input-output examples completely distinguishes one target function from another. Similarly, if we could represent each of the desired hints by a set of examples, it would not matter what type of hint we wanted to introduce.

To represent the monotonicity hint in credit-card approval, an example might take the form of two persons with identical data except for salary. When the applications of both are presented to the machine, its responses may agree with the hint (by approving or denying credit to both or by approving only the higher-salaried applicant), or it may disagree (by approving only the lower-salaried person). The learning algorithm can adjust the machine's parameters to satisfy the hint, exactly as though it were incorporating one more example of the target function. Learning from hints in this way can therefore piggyback other learning mechanisms.

Remarkably, the examples representing the hint need not be real. The two applicants with different salaries could be hypothetical, or "virtual," cases. We can use virtual examples because we are not requiring the machine to make the correct decision about a real person but rather to act in a way that is consistent with the hint. This principle can also apply to the symmetry hint in foreign exchange; virtual examples can be constructed from price patterns that never occurred in history. For the task of computer vision, we can represent invariance hints using pictures of objects that are completely unrelated to the real target function. Our training for this hint does not require that the machine's output be right or wrong, only that it remain consistent as the input pattern shifts or changes in size.

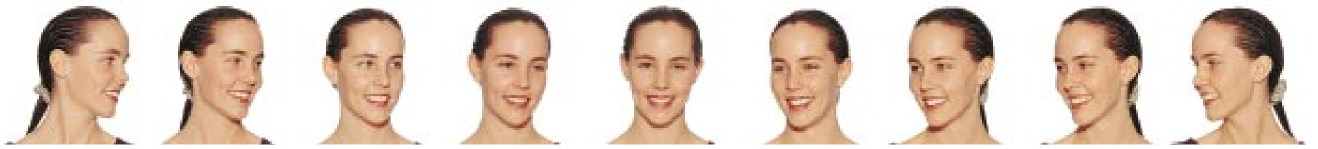
Virtual examples are extremely important in many applications because they can add substantial information to what may be a meager set of training examples. In foreign-exchange forecasting, for instance, the real data are limited to a small set of recent price patterns. The ability to supplement this scant historical data with virtual examples is thus of great value. The remaining challenge is to find a learning algorithm that can achieve the proper balance between the hints and the real input-output examples.

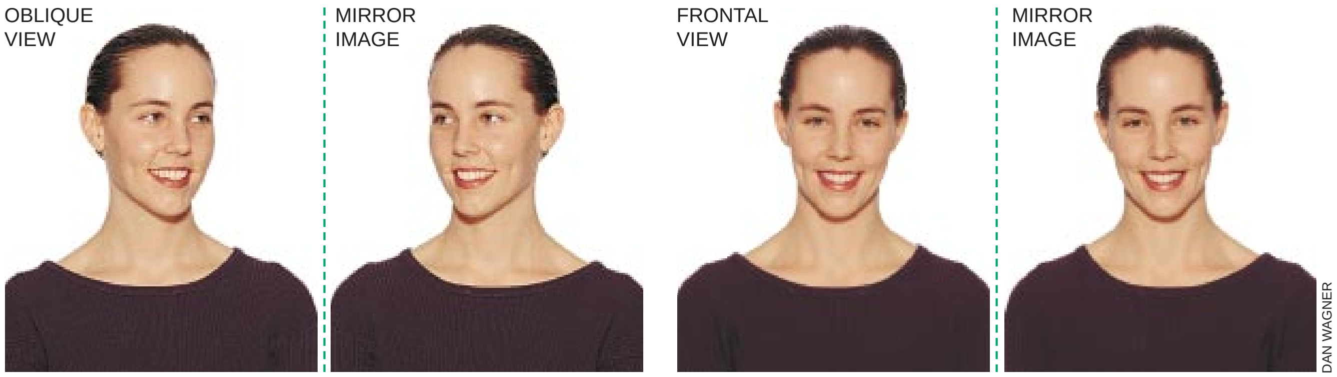
JEAN MIELE *The Stock Market*

FOREIGN-EXCHANGE trading systems improve with symmetry hints. For instance, if the value of the dollar rises relative to the deutsche mark when a certain pattern appears in its price history, the mark should rise in the same way when its history shows that pattern. A neural-network program for foreign exchange that was provided with this hint performed more profitably than did machines supplied with either an uninformative hint, an erroneous hint or no hint.

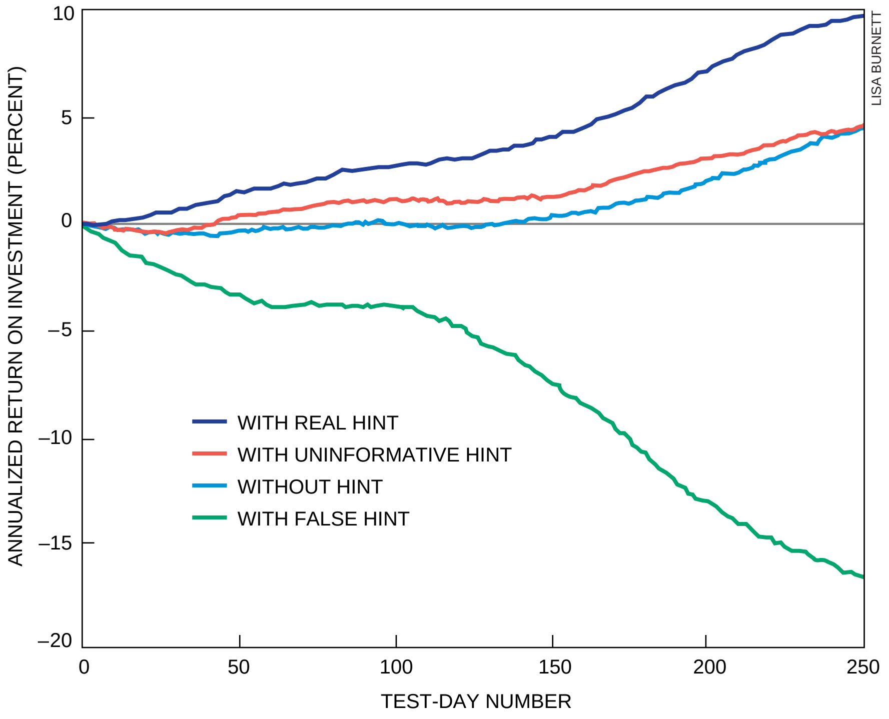
The Balancing Act

A learning algorithm will strive to adjust the parameters of the machine to agree simultaneously with the entire training set and with all the examples of each hint. But a perfect solution is normally impossible, so some compro-





RECOGNIZING A FACE from any angle becomes easier with a symmetry hint. A naive machine, knowing nothing about human faces, would need many views to learn how to master this task (*top*). A hint that faces are symmetrical (*bottom*) allows generalization from a smaller number of angles. Animals may rely on an innate understanding of such a hint: monkeys can be taught to recognize a face more reliably if they memorize it first from an oblique angle rather than a frontal view.



mise must be found. To achieve this goal, the machine needs to assess how much agreement exists at each step. During the training process, some hints may be learned better than others. But if the computer can determine which hint is least well learned, it can pay more attention to it in the next iteration. This is the concept behind Adaptive Minimization, an algorithm I developed that schedules different hints for learning in a way that achieves a balance among them and with the training set.

The Adaptive Minimization algorithm is "adaptive" in the sense that it constantly evaluates how well the machine is satisfying both the hints and training examples, and it continually modifies the adjustable parameters. The term "minimization" reflects that the algorithm is trying to minimize a quantitative measure of the error between the current actions of the machine and the behavior ultimately desired for it.

Although I began work in this area with many of the basic ideas in mind, I have to admit that my first breakthrough came from necessity rather than spontaneous insight. I had been invited to present the hints framework at a scientific meeting, and only the day before did I find out that the seminar was scheduled to be 10 minutes longer than I had anticipated. The thought of facing my audience with an embarrassingly short lecture kept me up all night trying to see if I could expand on the algorithmic part of my talk. That evening the main idea of the Adaptive Minimization algorithm came to me; the next day I delivered it in a well-received lecture. I doubt my audience appreciated just how recent those results were!

Learning Impediments

There are many challenges still facing the technology of machine learning. Perhaps the most severe faults stem from the tendency of machines to "overlearn" from training examples—something that can compromise a machine's ability to function correctly. Overlearning takes place when the machine memorizes the training examples at the expense of generalization. You might encounter a similarly ironic situation if you travel to Egypt and take a tour of the Pyramids. Some local guides provide an elaborate narrative for the tour, in English, and can answer common questions about the pharaohs perfectly. If you are encouraged by this performance and further query them, you will be surprised to find out that they do not speak English! They have memorized the necessary English sentences for a tour, but they have not "generalized" at all. Machine learning can suffer the same fate.

Another common pitfall in more complex machine-learning problems is a requirement for excessive computation time. As the learning algorithm searches for the optimal settings of free parameters (called the global optimum), it sometimes gets trapped in a poorer configuration (called a local optimum) that is better than similar solutions but still not the best that is theoretically possible. There is no efficient way for avoiding local optima in general. Some learning tasks have been shown to be NP-complete, a technical term that characterizes a class of computational problems believed to require excessive amounts of computer time to find the global optimum. In practice, however, the problem has not been debilitating. Satisfactory performance usually requires only that the machine reach a good local optimum.

Despite the existence of such difficulties, machine learning has proved itself worthy in solving a wide array of real-world problems. It is a classical subject rooted in research carried out many decades ago, but it has been rejuvenated and expanded in recent years. With the addition of procedures for learning from hints and other technical advances yet to come, machine learning will undoubtedly continue to make its way into our daily lives.

FURTHER READING

LEARNING FROM HINTS IN NEURAL NETWORKS. Y. S. Abu-Mostafa in *Journal of Complexity,* Vol. 6, No. 2, pages 192–198; June 1990.

ARTIFICIAL INTELLIGENCE. Special issue of *Communications of the ACM,* Vol. 37, No. 3; March 1994.

LEARNING FROM HINTS. Y. S. Abu-Mostafa in *Journal of Complexity,* Vol. 10, No. 1, pages 165–178; March 1994.

NEURAL NETWORKS IN THE CAPITAL MARKETS. Edited by Paul Refenes. John Wiley, 1995.