MACHINE LEARNING: THE FUTURE

Anupam Mahajan*

Madhur Chanana*

Dishant Sharma*

KunalSachdeva*

ABSTRACT

Much of current machine learning (ML) research has lost its connection to problems of import to the larger world of science and society. From this perspective, there exist glaring limitations in the data sets we investigate, the metrics we employ for evaluation, and the degree to which results are communicated back to their originating domains.What changes are needed to how we conduct research to increase the impact that ML has? We present six Impact Challenges to explicitly focus the field's energy and attention,and we discuss existing obstacles that must be addressed. I will first discuss current work in machine learning in particular,feedforeword Artificial Neural Nets (ANN), Boolean Belief Nets (BBN).

Among techniques employing recursion, Recurrent Neural Nets, Context Free Grammar Discovery, Genetic Algorithms, and Genetic Programming have been prominent.

Key Terms : AI,ANN,ML,SA,CFG,OOPs

* CSE Department , Dronacharya College Of Engineering, Gurgaon, Haryana, India

1 Introduction

The Machine Learning field evolved from the broad field of *Artificial Intelligence*, which aims to mimic intelligent abilities of humans by machines. In the field of *Machine Learning* one considers the important question of how to make machines able to "learn". Learning in this context is understood as *inductive inference*, where one observes *examples* that represent incomplete information about some "statistical phenomenon".

In *unsupervised* learning one typically tries to uncover hidden regularities (e.g. clusters) or to detect anomalies in the data (for instance some unusual machine function or a network intrusion). In *supervised learning*, there is a *label* associated with each example. It is supposed to be the answer to a question about the example. Ifthe label is discrete, then the task is called *classification problem* – otherwise, for real valuedlabels we speak of a *regression problem*. Based on these examples (including, the labels), one is particularly interested to *predict* the answer for other cases before they are explicitly observed. Hence, learning is not only a question of remembering but also of *generalization to unseen cases*.

Many machine learning problems are phrased in terms of an objective function to be optimized. It is time for us to ask a question of larger scope: what is the field's

objective function? Do we seek to maximize performance on isolated data sets? Or can we characterize progress in a more meaningful way that measures the concrete impact of machine learning innovations?

Much of machine learning (ML) research is inspired by weighty problems from biology, medicine, finance, astronomy, etc. The growing area of computational sustainability (Gomes, 2009) seeks to connect ML advances to real-world challenges in the environment, economy, and society. The CALO (Cognitive Assistant that Learns and Organizes) project aimed to integrate learning and reasoning into a desktop assistant, potentially impacting everyone who uses a computer (SRI International, 2003–2009). Machine learning has effectively solved spam email detection (Zdziarski, 2005) and machine translation (Koehn et al., 2003), two problems of global import. And so on. And yet we still observe a proliferation of published ML papers that evaluate new algorithms on a handful of isolated benchmark data sets. Their "real world" experiments may operate on data that originated in the real world, but the results are rarely communicated back to the origin. Quantitative improvements in performance are rarely accompanied by an assessment of whether those gains matter to the world outside of machine learning research.

This phenomenon occurs because there is no widespread emphasis, in the training of graduate student researchers or in the review process for submitted papers, on connecting ML advances back to the larger world. Even the rich assortment of applications-driven ML research often fails to take the final step to translate results into impact.

2 Supervised Classification

An important task in Machine Learning is *classification*, also referred to as pattern recognition, where one attempts to build algorithms capable of automatically constructing methods for distinguishing between different exemplars, based on their differentiating patterns.
Watanabe [1985] described a pattern as ”the opposite of chaos; it is an entity,vaguely defined, that could be given a name.” Examples of patterns are human faces, text documents, handwritten letters or digits, EEG signals, and the DNA sequences that may cause a certain disease. More formally, the goal of a (supervised) classification task is to find a functional mapping between the input data X, describing the input pattern, to a class label Y (e.g. −1 or +1), such that Y = f(X). The construction of the mapping is based on so-called *training data* supplied to the classification algorithm. The aim is to accurately predict the correct label on unseen data.
A pattern (also: “example”) is described by its *features*. These are the characteristics of the examples for a given problem. For instance, in a face recognition task some features could be the color of the eyes or the distance between the eyes. Thus, the input 1 to a pattern recognition task can be viewed as a two-dimensional matrix, whose axes are the examples and the features.

Pattern classification tasks are often divided into several sub-tasks:

1. Data collection and representation.
2. Feature selection and/or feature reduction.
3. Classification.

Data collection and representation are mostly problem-specific. Therefore it is difficult to give general statements about this step of the process. In broad terms, one should try to find invariant features, that describe the differences in classes as best as possible. Feature selection and feature reduction attempt to reduce the dimensionality (i.e. the number of features) for the remaining steps of the task. Finally, the classification phase of the process finds the actual mapping between patterns and labels (or targets). In many applications the second step is not essential or is implicitly performed in the third step.

3 Classification Algorithms

Although Machine Learning is a relatively young field of research, there exist more learning algorithms than I can mention in this introduction. I chose to describe six methods that I am frequently using when solving data analysis tasks (usually classification). The first four methods are traditional techniques that have been widely used in the past and work reasonably well when analyzing low dimensional data sets with not too few labeled training examples. In the second part I will briefly outline two methods (Support Vector Machines & Boosting) that have received a lot of attention in the Machine Learning community recently. They are able to solve high-dimensional problems with very few examples (e.g. fifty) quite accurately and also work efficiently when examples are abundant (for instance several hundred thousands of examples).

3.1 Traditional Techniques

k-Nearest Neighbor Classification Arguably the simplest method is the *k-Nearest Neighbor*classifier [Cover and Hart, 1967]. Here the k points of the training data closest to the test point are found, and a label is given to the test point by a majority vote between the k points. This method is highly intuitive and attains – given its simplicity – remarkably low classification errors, but it is computationally expensive and requires a large memory to store the training data.

Linear Discriminant Analysis computes a hyperplane in the input space that minimizes the within-class variance and maximizes the between class distance [Fisher, 1936]. It can be efficiently computed in the linear case even with large data sets. However, often a linear separation is not sufficient. Nonlinear extensions by using kernels exist [Mika et al., 2003], however, making it difficult to apply it to problems with large training sets.

Decision Trees Another intuitive class of classification algorithms are *decision trees*. These algorithms solve the classification problem by repeatedly partitioning the in-
2 put space, so as to build a tree whose nodes are as pure as possible (that is, they contain points of a single class). Classification of a new test point is achieved by moving from top to bottom along the branches of the tree, starting from the root node, until a terminal node is reached. Decision trees are simple yet effective classification schemes for small datasets. The computational complexity scales unfavourably with the number of dimensions of the data. Large datasets tend to result in complicated trees, which in turn require a large memory for storage. The C4.5 implementation by Quinlan [1992] is frequently used and can be downloaded at http://www.rulequest.com/Personal.

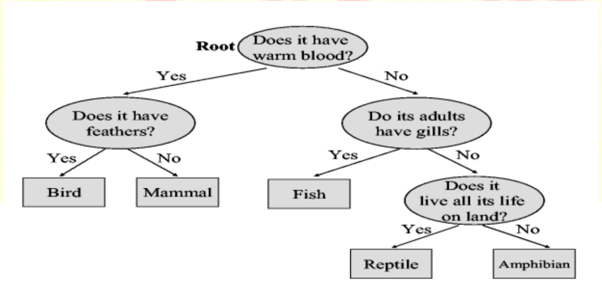

Figure 1: An example a decision tree (Figure taken from Yom-Tov [2004]).

Neural Networks are perhaps one of the most commonly used approaches to classification. Neural networks (suggested first by Turing [1992]) are a computational model inspired by the connectivity of neurons in animate nervous systems. A further boost to their popularity came with theproof that they can approximate any function mappingvia the Universal Approximation Theorem [Haykin,1999]. A simple scheme for a neural network isshown in Figure 2. Each circle denotes a computationalelement referred to as a *neuron*, which computesa weighted sum of its inputs, and possibly performs anonlinear function on this sum. If certain classes ofnonlinear functions are used, the function computedby the network can approximate any function (specifically a mapping from the training patterns to the training targets), provided enoughneurons exist in the network and enough training examples are provided.

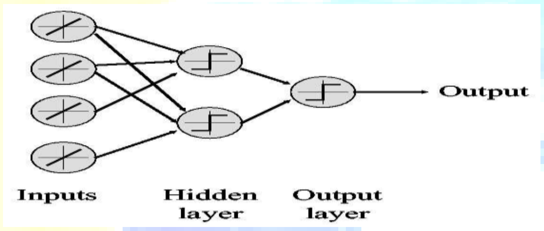

Figure 2: A schematic diagram of a neural network. Each circle in the hidden and output layer is a computational element known as a neuron. (Figure taken from Yom-Tov [2004])

4 Models for Machine Learning: Recursive and Nonrecursive.

In 1957 Frank Rosenblatt (Ros 57) wrote a report on Perceptrons that initiated a burst of enthusiasm in an area of machine learning. About 12 years later, Minsky and Papert (Min 69) wrote an analysis of Perceptrons | showing that they couldn't learn the logical \exclusive or" function. This resulted in a cutoff of federal funds for neural nets for a while, but eventually it was found that

artificial neurons that were only slightly different from Perceptrons*could* do exclusive or" - and that they were *universal* in the sense of being able to approximate any continuous function.

There was a new burst of enthusiasm, (and even funding) for neural nets | marked to some extent in 1986 by Rumelhart and McClelland's \Parallel Distributed Processing" (Rum 86). A second edition of Minsky and Papert's book (Min 88) appeared in 1988| pointing out that while neural

nets might be able to distinguish between sets that had an even (versus odd) number of elements, they needed a proportionally larger number of neurons and larger data set to do so. They had the same
difficulty in dealing with the \greater than" function.It is clear that in general, neural nets do very poorly in discriminations that are best described recursively. While the nets can (through their particular kind
of universality) make such discriminations, they need a large number of neurons and a very large data set to get much precision.It is characteristic of smart induction models, that they get good predictions
with small amounts of data. Humans occasionally do what is called \One shot learning."

Another kind of problem in which neural nets do poorly: we have a data set that consists of a few cycles of a sine wave plus a little noise. Neural nets can extrapolate such data into the *near* future. They will not, however, recognize that the data is, indeed, a sine wave plus noise | which would enable extrapolation into the *more distant* future. If the data set were continued with a noisy sine wave of another frequency and amplitude, the neural net would need lots of data points and many more neurons to extrapolate it. A much more sophisticated learner would realize that two sine waves only need six parameters to characterize them | so it could extrapolate the sine waves with much higher
precision using fewer data points.In general, to do economical prediction, we need (at least) facilities to make recursive definitions.

It should be noted that it is not only feed forward neural nets that lack these facilities. A large number of techniques used in current machine learning are no better: i.e. Radial Basis Functions, Boolean Belief Nets, Support Vector Machines, and Prediction by Partial Matching are a few: for certain areas of prediction these techniques are fine. They work very well for the problems thatare normally given in machine learning contests.With enough cubical Lego blocks one can make very beautiful buildings-
but with the addition of wheels, axles, motors, sensors and computers, one cando much more!
To work really difficult problems, it is necessary to have all possible facilitiesavailable. Recursion is one of these facilities. Later, I will discuss more advancedtools.

Some machine learning techniques that get past the “Minsky-Papert recursion barrier": Recurrent Neural Nets, various Evolutionary Techniques, Minimum Description Length/Minimum Message Length, Algorithmic Probability/Universal Distribution, Stochastic Grammars, Inductive Logic Programming

5 SA, The Scientist's Assistant

We will describe recent developments in a system for machine learning that we've been working on for some time (Sol 86, 89, 03a). It is meant to be a \Scientist's Assistant" of great power and versatility in many areas of science and mathematics. It differs from other ambitious work in this area in that we
are not so much interested in knowledge itself, as we are in how it is acquired - how machines may learn.

We will begin with the description of a simple kind of inductive inferencesystem. We are given a sequence of *Q;A*pairs (questions and correct answers).Then, given a new *Q*, the system must give an appropriate answer. At first,the problems will be mathematical questions in which there is only one correct answer. The system tries to find an appropriate function *F* so that for allexamples, *Qi;Ai*; *F*(*Qi*) = *Ai*. We look for *F* functions that have highest a priori probabilities - that have \short descriptions". In generating such functions,we use compositions of primitive functions built into the system. The overall
language used is very close to Lisp, but there is a difference in how recursionis represented. I am not yet certain as to whether this language will be a realimprovement over Lisp in the present system.

The *Q;A*formalism for problems is fairly general. We can express many kinds of information in that form. This makes it easy to put information into the system.Atfirst, the problems will all be deterministic: only one correct solution for each problem. Later we will allow several possible solutions to each problem and the system must find probabilities for each of them.The system starts with equal probability for all primitives, and finds solutions to simple problems by combining them | roughly in order of probability of each string of primitives. Because some trials can take a very long time |

occasionally not converging at all | we have to and some way to truncate trials. We use a technique called Levin Search (Lsearch) in which a testing time limit is assigned to a trial proportional to the a priori probability of that trial- which is roughly negative exponential in the number of symbols in the trial description. Short descriptions get lots of time, long descriptions, very little time. Symbolically:

$Ti = Pi ¢ T$

Pi is the a priori probability assigned to a trial
Ti is the maximum time allowed for the trial.
T is a constant for each run. we start with *T* set to the time for about five
instructions to be executed.

For a single run, we do all possible trials, using $Ti = Pi ¢ T$: Since$\sum Pi \le 1, \sum Ti \le T$: the total time for a run is $\le T$. If we don't find an acceptable trial in that run, we double *T* and do a new run. These runs continue by doubling *T* until we find an acceptable solution. It is easy to show that if *Pj*is the
probability of an acceptable solution, and it takes *Tj*time to generate and test this solution, then the entire search time will be $<2Tj/Pj$.

After we have solved a fair number of simple problems this way, we no longer assign equal probability to all primitives. We take the entire set of problem solutions as a corpus for any of the (initially) non-recursive prediction methods of Section 1: These methods can be used to assign probabilities to
various possible continuations of that corpus of problem solutions| creating candidates for problem solutions. We end up with a much higher probability being assigned to the correct solution, so *Ti/Pi* for that solution is much smaller,and we take much less time to find it.
At first, we use non-recursive prediction methods, because they are usually faster. Eventually, we will use the recursive methods mentioned in Section 1. Probabilistic Grammar Discovery

appears to be very promising. For a particular problem, prediction techniques will be tried in expected *Tj*=*Pj*order| small values first. The evaluation of this expected value is a kind of *Metaprediction* problem.

I have not yet programmed much of SA. Schmidhuber, however, has programmed OOPS (Sch 02), which is similar to SA in many ways. OOPS has been able to find a generally recursive solution for the -Tower of Hanoi" problem, after having solved a simpler problem with a recursive solution.

6 The Future of A.I.

How far are we from serious A.I.? It is my impression that we are not very far.Koza's system is very good, and though it is quite slow, there are several techniques for speeding it up and augmenting its functionality.Another promising system is Schmidhuber's OOPS (Sch 02). It uses Levin search over a Turing complete set of instructions to find solutions to problems, and has been able to find recursive solutions for them. Though it suffers from various deficiencies, most of them can be corrected with techniques that have been already developed in the machine learning community.In a more general context, we have just about all the needed tools and parts.It remains only to put them together.

7 Conclusions

Machine learning offers a cornucopia of useful ways to approach problems that otherwise defy manual solution.However, much current ML research suffers from a growing detachment from those real problems. Many investigators withdraw into their private studies with a copy of the data set and work in isolation to perfectalgorithmic performance. Publishing results to the MLcommunity is the end of the process. Successes usually are not communicated back to the original problem

setting, or not in a form that can be used.Yet these opportunities for real impact are widespread.

The worlds of law, finance, politics, medicine, education,and more stand to benefit from systems thatcan analyze, adapt, and take (or at least recommend) action. This paper identifies six examples of Impact Challenges and several real obstacles in the hope of inspiring a lively discussion of how ML can best make a difference. Aiming for real impact does not just increase our job satisfaction (though it may well dothat); it is the only way to get the rest of the world tonotice, recognize, value, and adopt ML solutions.

Machine Learning research has been extremely active the last few years. The result is a large number of very accurate and efficient algorithms that are quite easy to use for a practitioner. It seems rewarding and almost mandatory for (computer) scientist and engineers to learn how and where Machine Learning can help to automate tasks or provide predictions where humans have difficulties to comprehend large amounts of data. The long list of examples where Machine Learning techniques were successfully applied includes: Text classification and categorization [e.g. Joachims, 2001] (for instance spam filtering), network intrusion detection [e.g. Laskov et al., 2004], Bioinformatics (e.g. cancer tissue classification, gene finding; e.g. Furey et al. [2000], Zien et al. [2000], Sonnenburg et al. [2002]), brain computer interfacing [e.g. Blankertz et al., 2003], monitoring of electric appliances [e.g. Onoda et al., 2000], optimization of hard disk caching strategies [e.g. Gramacy et al., 2003] and disk spin-down prediction [e.g. Helmbold et al., 2000]), drug discovery [e.g. Warmuth et al., 2003]), high-energy physics particle classification, recognition of hand writing, natural scene analysis etc.

Obviously, in this brief summary I have to be far from being complete. I did not mention regression algorithms (e.g. ridge regression, regression trees), unsupervised learning algorithms (such as clustering, principle component analysis), reinforcement learning, online learning algorithms or model-selection issues. Some of these techniques extend the applicability of Machine Learning algorithms drastically.

References

[1] (Cil 05) Cilibrasi, R. and Vit¶anyi, P. 2005. Clustering by Compression. *IEEE Transactions on Information Theory* ,Vol 51, No. 4.

[2] (Cra 85) Cramer, N.L. 1985. A Representation for the Adaptive Generation of Simple Sequential Programs. In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Carnegie-Mellon University, July 24{26, 1985, J.J. Grefenstette, ed., Lawrence Erlbaum Associates, Hillsdale, N.J., pp. 183-187.
http://www.sover.net/~nichael/nlc-publications/icga85/index.html

[3] (Hut 02) Hutter, M. 2002. Optimality of Universal Bayesian Sequence Prediction for General Loss and Alphabet.
http://www.idsia.ch/~marcus/ai/

[4] (Gac 97) G¶acs, P. 1997, Theorem 5.2.1. in*An Introduction to Kolmogorov Complexity and Its Applications*, Li, M. and Vit¶anyi, P. , Springer-Verlag, N.Y. pp. 328-331.

[5] (Koz 99) Koza, J.R., Bennett III, F.H., Andre, D., Keane, M. 1999. *Genetic Programming III*, Mogan Kaufmann,

[6] (Min 69) Minsky, M.L., and Papert, S. 1969. *Perceptrons: An Introduction to Computational Geometry*, (First Edition), MIT Press, Cambridge, Mass.

[7] (Min 88) Minsky, M.L., and Papert, S. 1969. *Perceptrons: An Introduction to Computational Geometry*, (Expanded Edition), MIT Press, Cambridge, Mass.
[8] (New 57) Newell, A, Shaw, J., and Simon, H. 1963. Empirical Explorations with the Logic Theory Machine. *Proc of the Western Joint Computer Conference*, 15, 218-239. Reprinted in Feigenbaum and Feldman.

[9] Haykin. *Neural Networks: A comprehensive foundation, 2nd Ed.* Prentice-Hall, 1999.

[10] Meir and G. R¨atsch. An introduction to boosting and leveraging. In S. Mendelson .

[11] Langley, Pat. The changing science of machine learning.
Machine Learning, 82:275–279, 2011.

[12] (Sha 03) Shan, Y., McKay, R., Baxter, R, Abbass, H., Essam, D, Nguyen,
H. 2003. Grammar Model-Based Program Evolution. Canberra, Australia.

[13] (Sch 02) Schmidhuber, J. 2002. Optimal Ordered Problem Solver. TR
IDSIA-12-02, 31 July. http://www.idsia.ch/~juergen/oops.html

[14] (Sol 60) Solomonoff, R.J. 1960. A Preliminary Report on a General Theory
of Inductive Inference. (Revision of Report V{131), Contract AF 49(639){
376, Report ZTB{138, Zator Co., Cambridge, Mass.
http://world.std.com/~rjs/pubs.html

[15] (Sol 64a) Solomonoff, R.J. 1964. A Formal Theory of Inductive Inference.
Information and Control, Part I: Vol 7, No. 1, pp. 1{22.
http://world.std.com/~rjs/pubs.html

[16] (Sol 78) Solomonoff, R.J. 1978. Complexity{Based Induction Systems:
Comparisons and Convergence Theorems.*IEEE Trans. on Information
Theory*, VolIT{24, No. 4, pp. 422{432.
http://world.std.com/~rjs/pubs.html

[17] (Sol 86) Solomonoff, R.J. 1986. The Application of Algorithmic Probability to Problems in Artificial Intelligence.