Global Scientific JOURNALS

GSJ: Volume 8, Issue 7, July 2020, Online: ISSN 2320-9186
www.globalscientificjournal.com

# Machine Learning

**Ameer M Shariff, PhD**

[1] Network Infrastructure Department, SSGA, Stamford, CT, United States

E-mail: ameer.shariff@gmail.com

Advisor: Dr Philip Bradford, Professor, Computer Science Department, University of Connecticut

## Abstract

Machine learning research is great progress in many directions. This article summarizes these four directions and discusses some of them Current open issues. There are four directions (1) Improvement of classification accuracy by Methods of Learning Classmates Samples, (2) Augmenting supervised learning algorithms, (3) reinforcement learning and (4) learning of complex random samples.

There has been an explosion of machine learning research over the past five years. There are many reasons for this explosion: first, different Symbolic machine learning, computational learning theory, neural networks, statistics, and research associations in pattern recognition have found and worked together. Second, machine learning techniques are being applied to a new type of problem, including database, language processing, robot control, and knowledge discovery.

For traditional problems such as glowing optimization, as well as speech recognition, facial recognition, handwriting recognition, medical data analysis, and game play. In this article, I have chosen four topics in machine learning, where there has been a lot of activity recently. The purpose of the article is to outline the findings of some of the wider AI audiences in these areas and some open research issues. Topic areas include (1) classification classifiers, (2) methods of scaling up supervised learning algorithms, (3) reinforcement learning, and (4) learning of complex random models. The reader should be warned that this article is not a comprehensive review of everything Instead of this aspect, my aim is to provide a representative sample of research in each of these four fields. In each field, there are many other documents describing the relevant work. I apologize to the authors whose work it is Unable to add to the article.

Keywords: Machine Learning, Computing

## 1. Ensembles the classifier

The first topic deals with methods for improving accuracy in supervised learning. I'll start with some marking. In supervised learning, a learning program y = f (x) is given a training example of the form {(X1, y1), ..., (xm, ym) for an unknown function. Zivulus is usually the vectors of the form <xi, 1, xi, 2, ..., xi, n> whose parts are discrete or true values such as height, weight, colour and age. They are also called the properties of xi. Uses notation xith to denote the jth attribute of xi. In some cases, I will drop the i subscription when indicated by reference. Y values are usually derived from discrete sets of {1, ..., K disc in terms of classification or regression. In this article, I mainly focus on classification. Some of the training examples may be contaminated by random noise.

Given a set of training examples S, a learning algorithm generates classification. Classification is a hypothesis about the true function. Depending on the new x values, it ts the corresponding y values. I H1, ..., denotes classmates by HL. Classmates are a group of classifications whose individual decisions are combined into some form (usually without weight or voting) to classify new ones. One of the most active areas of research in supervised teaching is the study of methods for building good actors of classification. The main finding is that the ensemble is often more accurate than the individual classifiers that make them. If individual classifiers disagree with one another, the ensemble may be more accurate than its component classification (Hansen and Salamia 1990). To see why, imagine that we have a set of three classifiers: {h1, h2, h3}, and consider the new case x. If the three classifiers are identical, then h1 (x) is false, while h2 (x) and h3 (x) are also false.

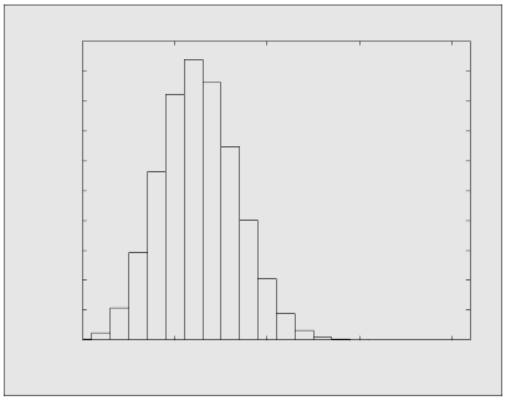

Figure 1 The Probability That Exactly I (of 21) Hypotheses Will Make an Error, Assuming Each Hypothesis Has an Error Rate of 0.3 and Makes Its Errors Independently of the Other Hypotheses.

However, if the classification errors are unrelated, then when h1 (x) is false, h2 (x) and h3 (x) may be correct, so that the majority vote correctly classifies x. More precisely, if the error rate of L hypothesis hl is equal to all p <1/2 If the errors are independent, then the probability that the majority vote is incorrect is the area under the binomial distribution, where hypotheses greater than L / 2 are incorrect. Figure 1 shows this region for the simulated ensemble of 21 Hypothesis, each error rate is 0.3. For 11 or more hypotheses, the area under the curve is simultaneously 0.026, which is much less than the error rate of the individual hypotheses.

## 2. Methods for constructing Ensembles

Several ways have been developed to create the ensemble. Some methods are common, and they can be applied to any learning algorithm. Other methods are specific to specific algorithms. I will start by reviewing the general methods

The first approach turns training examples into making multiple hypotheses. Learning algorithms can be implemented multiple times, each time with a different subset of training examples. This technique works especially for volatile learning algorithms - output classifiers undergo large changes in response to small changes in training data. Decision trees, neural networks and rule-learning algorithms are all inconsistent. Linear-regression, near-neighbour and nonlinear-threshold algorithms are generally stable. Bagging is the most direct way to change a training set. In each run, the bagging learning algorithm is combined with a set of M training sets, with a sample of M training examples randomly drawn from the original training set of M objects. This type of training set is called bootstrap.

A replica of the original training set, and the technique is called bootstrap aggregation (Breiman 1996a). Each bootstrap replicate averages 63.2 percent of the original training set, with multiple training scenarios appearing multiple times. Another training-set model method is to construct training sets by ignoring unrelated subsets of the training data. For example, the training set can be randomly divided into 10 dismount subsets. Then, 10 overlapping training sets can be constructed by separating one of these 10 subsets. The same approach can be used to create training sets for ten-fold cross validation; Therefore, groups created in this way are sometimes called cross-validated committees (Permanento, Munro, and Doyle 1996).

The third method for transforming the training set is described by the ADABOOST algorithm, developed by Freund and Shepaire (1996, 1995) and shown in Figure 2. ADABOOST puts probability distribution pl (x) on training examples. In each iteration l, it draws a training set of size m by sampling with the substitution according to the probability distribution pl (x). The learning algorithm is implemented to generate the classification HL. The error rate (weighted according to pl (x)) of this classification on training examples is calculated and used to adjust the probability distribution in the training context. (Note that, in Fig. 2)

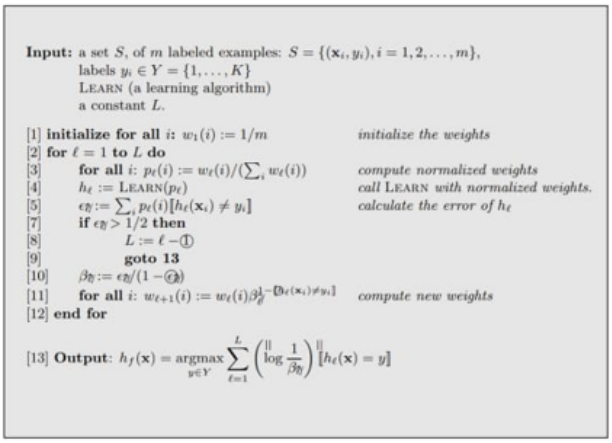


Figure 2 The ADABOOST.M1 Algorithm. The formula [[E]] is 1 if E is true and 0 otherwise

The probability distribution is obtained by generalizing the set of weights (W) over the training examples.)

A change in weight can have an impact Put more weight on training examples that are misconfigured by HL and less on properly classified examples. In subsequent iterations, ADABOOST makes learning progressively more difficult Problem.

(I). Critical training scenario errors cause larger gradient-descending steps than trivial (underweight) example errors. However, if the algorithm cannot use the probability distribution pl, the training model can be constructed by constructing a random sample proportional to the probability pl. This process makes ADABOOST more random, but experiments have shown that it is still effective. (Figure 3) compares the performance of C4.5 was compared with AD4OOST.M1 (using random samples). One point is made for each of the 27 test domains taken from the Irwin repository of machine learning databases (Merz and Murphy 1996). We

can see that most points are above the y = x line, which indicates that the error rate of ADABOOST is less than the error rate of C4.5. (Figure 4) compares the performance of bagging (with C4.5) to C4.5 only. Then, we see that bagging leads to large size reductions in the error rate of C4.5 for many problems. Finally, Figure 5 compares with Boosting (both use C4.5 as the underlying algorithm). The results suggest that the two methods are comparable, although the boosting is still visible there is an advantage over bagging.

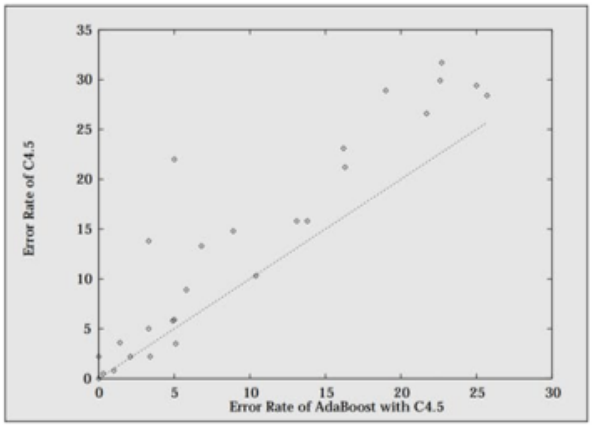


Figure 3 Comparison of ADABOOST.M1 (applied to C4.5) with C4.5 by Itself.

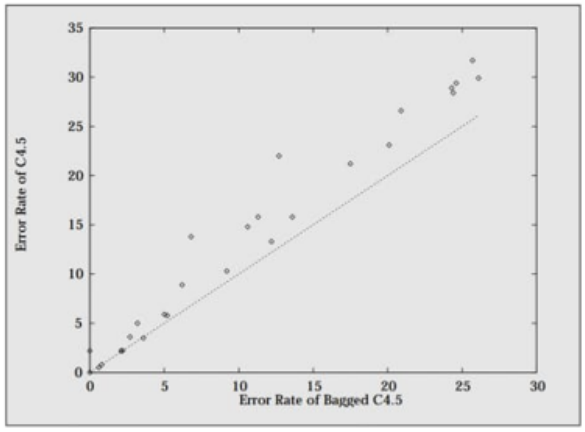


Figure 4 Comparison of Bagging (applied to C4)

## 3. Scaling up in Machine Learning Algorithm

The second major research area has explored millions of training examples, thousands of features, and methods of scaling learning algorithms to apply them to hundreds of classroom problems. Database-mining applications cause major machine learning problems, where millions of transactions occur every day and It is advisable to have a machine learning algorithm that can set such large data within a few hours of a computer. Another area where major learning problems arise is the retrieval of information from the full-text database and the World Wide Web. In information retrieval, every word of a document can be considered an input attribute; Therefore, every training example can be explained by thousands of features. Finally, the application of speech recognition, object recognition, and letter recognition to Chinese and Japanese current conditions that require discrimination in hundreds or thousands of classes.

## 5. Scaling up learning Algorithm

Scale learning algorithms applicable to very large problems. With the methods described here, problems with a million training examples can be solved in a reasonable amount of time in a computer. However, it is unclear whether the current stack of ideas permits solving problems with billions of training examples. Gathering more practical experience with very large problems is an important open-ended problem so that we can understand their characteristics and determine where these algorithms fail.

A recurring theme is the use of a subgroup of training data to make important intermediate decisions (such as the selection of relevant attributes). Another aspect is the development of efficient online algorithms such as WinNOW. These are always algorithms that can provide a useful answer no matter how long they last. The longer they last, the better the results. The problem of managing thousands of output classes is an important public factor. Class enablers of classification have already described two methods that are well suited for this case: error-correction output coding and ADABOOST.OC. These two methods should be well measured with the number of classes. Error-correction output coding has been tested on 126 classes of problems, but tests on very large problems with thousands of classes have not yet been performed.

## 6. Reinforcement Learning

The previous two sections discussed problems in supervised learning from examples. This section addresses problems of sequential deciding and control that come under the heading of reinforcement learning.

Work in reinforcement learning dates back to earliest days of AI when Arthur Samuel (1959) developed his famous checkers program.

More recently, there are several important advances within the practice and theory of reinforcement learning. Perhaps the foremost famous work is Gerry Tesauro's (1992) TD-GAMMON program, which has learned to play backgammon better than the other computer virus and almost also because the best human players. Two

other interesting applications are the work of Zhang and Dietterich (1995) on job-shop scheduling and Crites and Barto (1995) on real-time scheduling of passenger elevators. Kaelbling, Littman, and Moore (1996) published a superb survey of reinforcement learning, and Mahadevan and Kaelbling (1996) report on a recent National Science

Foundation–sponsored workshop on the topic. Two new books (Barto and Sutton 1997; Bertsekas and Tsitskilis 1996) describe the newly developed reinforcement learning algorithms and therefore the theory behind them. I summarize these developments here.

## 7. Open problems in Reinforcement Learning

Many important problems remain unsolved in reinforcement learning, which reflects the relative youth of the sector. I discuss a couple of of those problems here. First, the utilization of multilayer sigmoidal neural networks for value-function approximation has worked, but there's no reason to believe that such networks are compatible to reinforcement learning. First, they have a tendency to forget episodes (both good and bad) unless they're retrained on the episodes frequently. Second, the necessity to form small gradient-descent steps makes learning slow, particularly within the early stages. a crucial open problem is to clarify what properties a perfect value-function approximator would possess and develop function approximators with these properties. Initial research suggests that value-function approximators should be local averagers that compute the worth of a replacement state by interpolating among the values of previously visited states (Gordon 1995).

A second key problem is to develop reinforcement methods for hierarchical problem solving. For very large search spaces, where the space to the goal and therefore the branching factor are big, no search method can work well. Often such large search spaces have a hierarchical (or approximately hierarchical) structure which will be exploited to scale back the value of search. There are several studies of ideas for hierarchical reinforcement learning (for example, Dayan and Hinton [1993], Kaelbling [1993], and Singh [1992]).

The third key problem is to develop intelligent exploration methods. Weak exploration methods that believe random or biased random choice of actions can't be expected to scale well to large, complex spaces. A property of the successful applications shown previously (particularly, backgammon and job-shop scheduling) is that even random search reaches a goal state and receives a gift. In domains where success is contingent an extended sequence of successful choices, random search features a low probability of receiving any reward. More intelligent search methods, like means-ends analysis, got to be integrated into reinforcement learning systems as they need been integrated into other learning architectures like SOAR (Laird, Newell, and Rosenbloom 1987) and PRODIGY (Minton et al. 1989).

A fourth problem is that optimizing cumulative discounted reward isn't always appropriate. In problems where the system must operate continuously, a far better goal is to maximise the typical reward per unit time. However, algorithms for this criterion are more complex and not also behaved. Several new methods are suggests recently (Mahadevan 1996; Ok and Tadepalli 1996; Schwartz 1993).

The fifth, and maybe most difficult, problem is that existing reinforcement learning algorithms assume that the whole state of the environment is visible at whenever step. This assumption isn't true in many applications, such as robot navigation or factory control, where the available sensors provide only partial information about the environment. a couple of algorithms for the answer of hidden-state reinforcement learning problems are developed (Littman, Cassandra, and Kaelbling 1995; McCallum 1995; Parr and Russell 1995; Cassandra, Kaelbling, and Littman 1994). Exact solution appears to be difficult. The challenge is to seek out approximate methods that scale well to large hidden-state applications. Despite these substantial open problems, reinforcement learning methods are already being applied to a good range of commercial problems where traditional dynamic programming methods are infeasible. Researchers within the area are optimistic that reinforcement learning algorithms can solve many problems that have resisted solution by machine-learning methods within the past. Indeed, the overall problem of selecting actions to optimize expected utility is exactly the matter faced by general intelligent agents. Reinforcement learning provides one approach to attacking these problems.

## 8. Learning Stochastic Model

The final topic that I discuss is that the area of learning stochastic models. Traditionally, researchers in machine learning have sought general-purpose learning algorithms—such because the decision tree, rule, neural network, and nearest-neighbor algorithms—that could efficiently search an outsized and versatile space of classifiers for an honest fit training data. Although these algorithms are general, they need a serious drawback, during a practical problem where there's extensive prior knowledge, it is often difficult to include this prior knowledge into these general algorithms. A secondary problem is that the classifiers constructed by these general learning algorithms are often difficult to interpret—their internal structure won't have any correspondence to the real-world process that's generating the training data over the past five years approximately , there has been tremendous interest during a more knowledge based approach supported stochastic modeling.

A stochastic model describes the real-world process by which the observed data are generated. Sometimes, the terms generative stochastic model and causal model are wont to emphasize this attitude. The stochastic model is usually represented as a probabilistic network—a graph structure that captures the probabilistic dependencies (and independencies) among a group of random variables. Each node within the graph has an associated probability distribution, and from these individual distributions, the joint distribution of the

observed data is often computed. to unravel a learning problem, the programmer designs the structure of the graphand chooses the sorts of the probability distributions, yielding a stochastic model with many free parameters (that is, the parameters of the node-probability distributions). Given a training sample, learning algorithms are often applied to work out the values of the free parameters, thereby fitting the model to the info. Once a stochastic model has been learned, probabilistic inference are often administered to support tasks like classification, diagnosis, and prediction. More details on probabilistic networks are given in two recent textbooks: Jensen (1996) and Castillo, Gutierrez, and Hadi (1997).

## 9. Conclusion

Any survey should select specific areas and leave it to others. I should briefly mention some other active areas. The central theme in the machine learning control learning. There have been many developments in this area. Researcher's explore different punishment functions and redesign methods (including crossvalidation) to prevent overuse. Be aware of the over-fitting process obtained by the statistical concepts of bias and discrimination, and most authors have developed a bias-based critique for classification problems. Another active case study was conducted algorithms for learned relationships the Horn-Clause Program. This area is also known as like inductive logic programming, and many algorithms and theoretical results have been developed in this area.

Finally, many applications address practical problems that arise in such applications. Methods of learned knowledge and methods algorithms for extracting semantic rules and noise detection from neural network and data to learn outliers in easy-to-understand classifiers and algorithms.There have been many exciting developments in the last five years, and they are relevant. The literature on machine learning is growing rapidly. Apply AI and science learning techniques to more areas of computer science I hope the flow, to attack their Interesting problems and practical solutions continues. This is an exciting time to work in machine learning

## References

Abu-Mostafa, Y. 1990. Learning from Hints in Neural Networks. Journal of Complexity 6:192–198.

Ali, K. M., and Pazzani, M. J. 1996. Error Reduction through Learning Multiple Descriptions. Machine Learning 24(3): 173–202.

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1995. Learning to Act Using Real-Time Dynamic Programming. Artificial Intelligence 72:81–138.

Barto, A. G., and Sutton, R. 1997. Introduction to Reinforcement Learning. Cambridge, Mass.: MIT Press. Bellman, R. E. 1957.

Dynamic Programming. Princeton, N.J.: Princeton University Press. Bertsekas, D. P., and Tsitsiklis, J. N. 1996.

Neuro-Dynamic Programming. Belmont, Mass.: Athena Scientific. Blum, A., and Rivest, R. L. 1988.

Training a 3-Node Neural Network Is NP-Complete (Extended Abstract).

In Proceedings of the 1988 Workshop on Computational Learning Theory, 9–18. San Francisco, Calif.: Morgan Kaufmann. Blum, A. 1997.

Empirical Support for WINNOW and Weighted-Majority Algorithms: Results on a Calendar Scheduling Domain. Machine Learning 26(1): 5–24.

Breiman, L. 1996a. Bagging Predictors. Machine Learning 24(2): 123–140. Breiman, L. 1996b. Stacked Regressions. Machine Learning 24:49–64.

Buntine, W. 1996. A Guide to the Literature on Learning Probabilistic Networks from Data. IEEE Transactions on Knowledge and Data Engineering 8:195–210. Buntine, W. 1994.

Operations for Learning with Graphical Models. Journal of Artificial Intelligence Research 2:159–225. Buntine, W. L. 1990.

A Theory of Learning Classification Rules. Ph.D. thesis, School of Computing Science, University of Technology.

## Acknowledgements

I am extremely thankful to Dr. Philip Bradford, Professor in University of Connecticut for supporting in creating this internal refernce architecture article. His guidelines have been extremely helpful and helped in formulating this formal article on security concerns within cloud infrastructure.

## Authors

**Ameer M Shariff** achieved his Bachelor of Engineering from University of Mumbai, Masters in Systems Engineering from Madurai Kamaraj University and Doctorate in Computer Engineering from University of Southampton, UK. He has been working as Sr. Infrastructure Architect for General Electric/State Street for past 15 years and has total experience of 20 years in the field of Network and Security Infrastructure. Currently he is woking as Sr. Consultant for Atos-Syntel. He has been using his expertise in establishing strong network and security controls in our internal infrastructure.