International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

# Scaling up Machine Learning Algorithms for Large Datasets

**Manju Joy**

Department of MCA, Federal Institute of Science and Technology (FISAT), Angamaly, Kerala, India

**Abstract:** ***Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed. There is a need to explore techniques for scaling up learning algorithms so that it can be applied to problems with millions of training examples, thousands of features, and hundreds of classes. Traditionally, the bottleneck preventing the development of more-intelligent systems via machine learning was limited data available. However, in many domains, the size of the datasets available now is so large and powerful learning algorithms are needed to learn from infinite data in finite time. This paper is a review of works in machine learning on methods for handling data sets containing large amounts of information. A method is proposed to handle this problem which is based on K-means clustering.***

**Keywords:** Relevant features, Feature selection, Decision tree, Clustering.

## 1. Introduction

Machine-learning research has been making great progress in many directions. Some of the current open problems in the area of Machine Learning are (1) the improvement of classification accuracy by learning ensembles of classifiers, (2) methods for scaling up machine learning algorithms, (3) reinforcement learning, and (4) the learning of complex stochastic models. The last five years have seen an explosion in machine-learning research. Two main reasons for this explosion are i) Separate research communities in symbolic machine learning, computational learning theory, neural networks, statistics, and pattern recognition have discovered one another and begun to work together. ii) Machine-learning techniques are being applied to new kind of problems, including knowledge discovery in databases, language processing, robot control, and combinatorial optimization, as well as to more traditional problems such as speech recognition, face recognition, handwriting recognition, medical data analysis, game playing etc.

## 2. Learning with Large Training sets

Decision tree algorithms have been extended to handle large data sets in three different ways. One approach is based on intelligently sampling subsets of the training data as the tree is grown. Decision trees are constructed by starting with the entire training set and an empty tree. A test is chosen for the root of the tree, and the training data are then partitioned into disjoint subsets depending on the outcome of the test. The algorithm is then applied recursively to each of these disjoint subsets. The algorithm terminates when all (or most) of the training examples within a subset of the data belong to the same class. At this point, a leaf node is created and labeled with the class. The process of choosing a test for the root of the tree (or for the root of each subtree) involves analyzing the training data and choosing the one feature that is the best predictor of the output class. In a large and redundant dataset, it might be possible to make this choice based on only a sample of the data. In 1993, an algorithm has been presented that dynamically chooses the sample based on how difficult the decision is at each node. The algorithm typically behaves by using a small sample near the root of the tree and then progressively enlarging the sample as the tree grows. This technique can reduce the time required to grow the tree without reducing the accuracy of the tree at all.

A second approach is based on developing clever data structures that avoid the need to store all training data in random-access memory. The hardest step for most decision tree algorithms is to find tests for real-valued features. These tests usually take the form $X_j <= \theta$ for some threshold value $\theta$. The standard approach is to sort the training examples at the current node according to the values of feature $X_j$ and then make a sequential pass over the sorted examples to choose the threshold $\theta$. SPRINT and SLIQ are two algorithms based on this approach. SPRINT has been applied to data sets containing 2.5 million examples. SLIQ that makes more use of main memory, scales to millions of training examples and is slightly faster than SPRINT.

A third approach to large data sets is to take advantage of ensembles of decision trees (Chan and Stolfo 1995). The training data can randomly be partitioned into *N* disjoint subsets. A separate decision tree can be grown from each subset in parallel. The trees can then vote to make classification decisions. Although the accuracy of each of the individual decision trees is less than the accuracy of a single tree grown with all the data, the accuracy of the ensemble is often better than the accuracy of a single tree.

A fourth approach to the problem of choosing splits for a real-valued input feature is to discretize the values of such features. For example, one feature of a training example might be students performance measured in percentage. This problem can be solved by grouping percentage into a small number of ranges (for example, 0%–25%; 25%– 50%; 50%–75%; 75%– 90%; and greater than 90%). If the ranges are chosen well, the resulting decision trees will still be accurate. Several simple and fast algorithms have been developed for choosing good discretization points.

## 3. Learning with many features

Two key issues in dealing with large datasets containing large amounts of information is

- the problem of selecting relevant features, and
- the problem of selecting relevant examples.

At a conceptual level, one can divide the task of concept learning into two subtasks: deciding which features to use in describing the concept and deciding how to combine those features. In this view, the selection of relevant features, and the elimination of irrelevant ones, is one of the central problems in machine learning,

The number of training examples needed to reach a desired level of accuracy, often called the ***sample complexity,*** grow slowly with the number of features present, if indeed not all these are needed to achieve good performance. In recent years, a growing amount of work in machine learning-both experimental and theoretical in nature-has focused on developing algorithms with such desirable properties.

Induction algorithms differ considerably in their emphasis on focusing on relevant features. At one extreme lies the simple nearest-neighbor method, which classifies test instances by retrieving the nearest stored training example, using all available attributes in its distance computations. Although this approach has excellent asymptotic accuracy, a little thought reveals that the presence of irrelevant attributes should considerably slow the rate of learning. In fact, average-case analysis of simple nearest-neighbor indicates that number of training examples needed to reach a given accuracy grows exponentially with the number of irrelevant attributes, even for conjunctive target concepts. Experimental studies of nearest-neighbor are consistent with this discouraging conclusion.

At the other extreme lie induction methods that explicitly attempt to select relevant features and reject irrelevant ones. Techniques for learning logical descriptions constitute the simplest example of this approach, and there are more sophisticated methods for identifying relevant attributes that can augment and improve any induction method, including nearest-neighbor. Theoretical and experimental results for these methods are much more encouraging. For instance, theoretical results show that if, by focusing on only a small subset of features, an algorithm can significantly reduce the number of hypotheses under consideration, then there is a corresponding reduction in the sample size sufficient to guarantee good generalization. Somewhat in the middle of the above two extremes are feature-weighting methods that do not explicitly select subsets of features, but still aim to achieve good scaling behavior.

In many learning problems, there are hundreds or thousands of potential features describing each input object **x**. Popular learning algorithms such as C4.5 and back propagation do not scale well when there are many features. Irrelevant and noisy input features provide little information. It is easy for learning algorithms to be confused by the noisy features and construct poor classifiers. Hence, in practical applications, it is wise to carefully choose which features to provide to the learning algorithm. Research in machine learning has sought to automate the selection and weighting of features, and many different algorithms have been developed for this purpose.

For feature selection, three main approaches have been pursued. The first approach is to perform some initial analysis of the training data and select a subset of the features to feed to the learning algorithm. A second approach is to try different subsets of the features on the learning algorithm, estimate the performance of the algorithm with these features, and keep the subsets that perform best. The third approach is to integrate the selection and weighting of features directly into the learning algorithm.

### 3.1 Selecting and Weighting Features by Preprocessing

A simple preprocessing technique is to compute the mutual information (also called the information gain) between each input feature and the class. The *mutual information* between two random variables is the average reduction in uncertainty about the second variable given a value of the first. Good results have been obtained with nearest-neighbor algorithms using mutual information weighting. A problem with mutual information weighting is that it treats each feature independently. For features whose predictive power is only apparent in combination with other features, mutual information assigns a weight of zero. One of the most successful preprocessing algorithms to date is the RELIEF-F algorithm (Kononenko 1994). The basic idea of these algorithms is to draw examples at random, compute their nearest neighbors, and adjust a set of feature weights to give more weight to features that discriminate the example from neighbors of different classes.

### 3.2 Selecting and Weighting Features by Testing with the Learning Algorithm

A computationally expensive method called the *wrapper method is available for* selecting input features. The idea used in wrapper method is to generate sets of features, run the learning algorithm using only these features, and evaluate the resulting classifiers using 10-fold cross-validation (or a single holdout set). In 10-fold cross-validation, the training data are subdivided randomly into 10 disjoint equal-sized sets. The learning algorithm is applied 10 times, each time on a training set containing all but one of these subsets. The resulting classifier is tested on the one-tenth of the data that was held out. The performance of the 10 classifiers (on their 10 respective holdout sets) is averaged to provide an estimate of the overall performance of the learning algorithm when trained with the given features.

Step-wise selection algorithms have been explored that start with a set of features (for example, the empty set) and considered adding or deleting a single feature. The possible changes to the feature set are evaluated (using 10-fold cross-validation), and the best change is made. Then, a new set of changes is considered. This method is only practical for data sets with relatively small numbers of features and fast learning algorithms, but it gave excellent results.

A much more efficient approach to feature selection is leave-one-out cross-validation (LOOCV) with the nearest-neighbor algorithm. In *leave-one-out cross-validation,* each training example is temporarily deleted from the training data, and the nearest-neighbor learning algorithm is applied to predict the class of the example. The total number of classification errors is the leave-one-out cross-validated estimate of the error rate of the learning algorithm. The LOOCV error is used to compare different sets of features with the goal of finding the set of relevant features that minimizes the LOOCV error.

### 3.3 Integrating Feature Weighting into the Learning Algorithm

Two methods that integrate feature selection directly into the learning algorithm are VSM method and WINNOW method. Both of them have been shown to work well experimentally, and the second method, called WINNOW, works extremely well in problems with thousands of potentially relevant input features.

The key to the effectiveness of VSM(***variable-kernel similarity metric)*** method is that it learns a weighted distance metric for measuring the distance between the new data point X and each training point $X_i$. VSM also adjusts the size s of the Gaussian distribution depending on the local density of training examples in the neighborhood of $\mathbf{X}_i$. In detail, VSM is controlled by a set of learned feature weights $w_1, \ldots, w_n$, a kernel radius parameter $r$, and a number of neighbors $R$.

WINNOW algorithm developed by Littlestone is a linear threshold algorithm for two-class problems with binary (that is, 0/1 valued) input features. It classifies a new example **x** into class 2 if $\sum_j W_j X_j > \theta$ and into class 1 otherwise. WINNOW is an online algorithm; it accepts examples one at a time and updates the weights $W_j$ as necessary. WINNOW initializes its weights W$j$ to 1. It then accepts a new example (**x**, $y$) and applies the threshold rule to compute the predicted class $y$'. If the predicted class is correct ($y' = y$), WINNOW does nothing. However, if the predicted class is wrong, WINNOW updates its weights as follows. If $y' = 0$ and $y = 1$, then the weights are too low; so, for each feature such that $xj = 1$, W$j$ := W$j$ · a, where a is a number greater than 1 called the *promotion parameter.* If $y' = 1$ and $y = 0$, then the weights were too high; so, for each feature X$j$ = 1, it decreases the corresponding weight by setting W$j$ := W$j$ · b, where b is a number less than 1 called the *demotion parameter.*

WINNOW is an example of an *exponential update algorithm*. The weights of the relevant features grow exponentially, but the weights of the irrelevant features shrink exponentially. A property of this algorithm is, it excel when the number of relevant features is small compared to the total number of features. WINNOW has been applied to several experimental learning problems. An important advantage of WINNOW, in addition to its speed and ability to operate online, is that it can adapt rapidly to changes in the target function.

K-means clustering can be used as a fast alternative training method. The main advantage of this approach is that it is very fast and easily implemented at large scale. K-means is one of the simplest unsupervised learning algorithms for solving the clustering problem. Clustering means organizing data into classes such that there is high intra-class similarity and low inter-class similarity. It finds natural groupings among objects. The procedure is simple and easy to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed. At this point re-calculate k new centroids of the clusters resulting from the previous step. After obtaining these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more. *K-means* clustering has been used as a feature learning, in either (semi) supervised or unsupervised learning. The basic approach is first to train a *k*-means clustering representation, using the input training data (which need not be labeled). Then, to project any input datum into the new feature space. K-means can give best results when data set are distinct or well separated from each other.

A human expert is likely to be guided by imprecise impressions or perhaps makes an examination of only a relatively small number of examples. Machine learning algorithms are data driven, and are able to examine large amounts of data. The results of using machine learning are often more accurate than what can be created through direct programming. So machine learning skills are in high demand. The work in machine learning has importance for expert system development, problem solving, computer vision and intelligent tutoring systems etc. The development of powerful learning systems may ultimately open an unprecedented range of new applications.

## References

[1] Thomas G. Dietterich,"Machine-Learning Research-Four Current Directions", AI Magazine Volume 18 Number 4(1997) (©AAAI) , pp105-113,1997.(journal style)

[2] Yaser Abu Mostafa, "Learning from Data", Introductory Machine learning Course, Division of Engineering and Applied Science, Caltech. .Available:https://work.***caltech***.*edu/ telecourse. html [released on April* 2012] [Online]. (General Internet site)

[3] Avrim L. Blum , Pat Langley, "Selection of relevant features and examples in machine Learning ", Artificial Intelligence, Volume 97, Issues 1-2, December 1997, pp 245-271. (journal style)

[4] Rob Schapire , Lecture notes on "Theoretical Machine Learning ", February 4, 2008.(technical report style)

[5] Pedro Domingos, Geoff Hulten, "A General Method for Scaling Up Machine Learning Algorithms and its Application to Clustering", In Proceedings of the Eighteenth International Conference on Machine Learning, pp106-113, .(conference style)
[6] Chris Williams, "Introduction -Machine Learning and Pattern Recognition", Lecture Slides, School of Informatics, University of Edinburgh, August 2014.(journal style)

## Author Profile

**Manju Joy** received MCA degree from MES College of Engineering, Kuttipuram in 2002. Presently she is working as Assistant professor in the Department of MCA at Federal Institute of Science and Technology(FISAT), Angamaly, Kerala.