SAND2015-4428C

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2015) 1–8

Procedia Computer Science

www.elsevier.com/locate/procedia

MapReduce SVM Game

Craig M. Vineyard, Stephen J. Verzi, Conrad D. James, and James B. Aimone[a], Gregory L. Heileman[b]

[a]*Sandia National Laboratories, Albuquerque, New Mexico 87185-1327 USA*
Email: cmviney@sandia.gov
[b]*Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM, 87131 USA*

Abstract

Despite technological advances making computing devices faster, smaller, and more prevalent in today's age, data generation and collection has outpaced data processing capabilities. Simply having more compute platforms does not provide a means of addressing challenging problems in the big data era. Rather, alternative processing approaches are needed and the application of machine learning to big data is hugely important. The MapReduce programming paradigm is an alternative to conventional supercomputing approaches, and requires less stringent data passing constrained problem decompositions. Rather, MapReduce relies upon defining a means of partitioning the desired problem so that subsets may be computed independently and recombined to yield the net desired result. However, not all machine learning algorithms are amenable to such an approach. Game-theoretic algorithms are often innately distributed, consisting of local interactions between players without requiring a central authority and are iterative by nature rather than requiring extensive retraining. Effectively, a game-theoretic approach to machine learning is well suited for the MapReduce paradigm and provides a novel, alternative new perspective to addressing the big data problem. In this paper we present a variant of our Support Vector Machine (SVM) Game classifier which may be used in a distributed manner, and show an illustrative example of applying this algorithm.

Keywords: Support vector machine, Game theory, Machine learning, MapReduce

1. Introduction

As computational costs have decreased over the past decades, the prevalence of computing devices in everyday life has increased immensely. Rather than simply using a few of the increasingly more efficient computational devices, instead more and more computing devices are used in ever expanding ways. Considering computing as a resource, this phenomenon is described by Jevons paradox [1]. Coupled with this prolific increase in computing is the ability to collect and record all sorts of data. Computing devices are no longer constrained to existing as large bulky items, but rather are increasingly more mobile, and as such may be stowed in pockets or strapped on wrists. Effectively, data may be generated perpetually throughout the day such as by fitness trackers logging steps, tracking vital signs, and even generating statistics when users are asleep.

Amidst the emergence of the big data era, computing advances have not kept up with the ability to process the big data explosion. Parallel computing is a natural solution, with the goal of trying to process

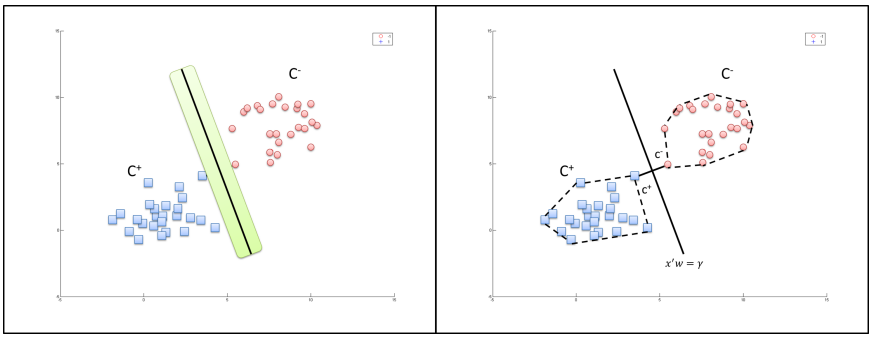

Fig. 1. Support Vector Machine Maximum Margin Principle and Equivalent Geometric SVM

greater amounts of data at once. However, conventional supercompuing approaches have remained expensive, and additionally, not all problems are well suited for high performance computing implementations. For instance, while neural networks in the brain are fundamentally parallel and distributed, artificial neural networks models are limited in their parallelizability due to constraints imposed the learning algorithms employed to train them. Consequently, algorithmic approaches for utilizing advances in computing technology and data availability are needed.

Game-theoretic algorithms are often innately distributed, consisting of local interactions between players without requiring a central authority and are iterative by nature rather than requiring extensive retraining. As such, game theory applied to machine learning provides a novel, alternative perspective to addressing the big data problem. In this paper we will describe the classic pattern classification Support Vector Machine (SVM) algorithm, describe means by which it can be applied to parallel computing platforms, and then show how our game-theoretic variant of SVM can likewise be applied. We then provide illustrative results highlighting this approach.

2. SVM

The Support Vector Machine (SVM) approach to learning seeks to find the separating hyperplane that maximizes the margin between the patterns in the classes it is separating, and these patterns serve as the support vectors [2][3]. Conceptually, this is similar to taking into account the long term classification goal as opposed to settling for the first discriminant which yields no training error. Furthermore, Bennett et al. proved there is a geometric interpretation which is equivalent to the dual of the canonical quadratic optimization approach to SVM [4]. This approach first constructs the convex hulls, the smallest convex set of points which fully encompass the set, around each of the data classes. Next it finds the closest points to each-other on each respective convex hull. The resulting discriminant is the perpendicular bisector of the line segment formed by these points. Figure 1 illustrates both the fundamental SVM principle as well as the geometric SVM approach. In the left half of the figure, the green rectangle represents the margin between the classes and the resulting discriminant is the black line central to this region. The right half of the figure depicts the identical discriminant resulting from the geometric approach.

Fundamentally, both the canonical variant of SVM and the geometric interpretation focus upon identifying the support vectors from the rest of the training data. As the size of the training data increases, this becomes intractable and reduction or parallelization techniques become necessary. One such approach is the Cascade SVM model by Graf et al. in which the overall problem is broken up into phases of smaller optimization problems. By dividing the overall problem into phases, non-support vectors are eliminated in early stages of processing and only prospective data points are passed forward to later stages which are effectively able to be operated upon a smaller optimization problem to ultimately identify the final support vectors [5]. Doing so additionally allows for parallelization of the SVM algorithm as the smaller optimizations may be solved independently and spread across multiple processors. Next we will describe a game theoretic

Algorithm 1 Basic SVM Game

procedure SVM_GAME$_B(S)$ ▷ Sets $C^+, C^- \in S$
  while iterations≠desired iterations **do**
    for each class C^+ and C^- **do**
      $p_x \leftarrow random(C^1)$
      $p_y \leftarrow random(C^1)$
      $p_r \leftarrow random(C^2)$
      if $\mathrm{d}(p_x, p_r) \leq \mathrm{d}(p_y, p_r)$ **then**
        $p_x \leftarrow \alpha$ from p_y
      else
        $p_y \leftarrow \alpha$ from p_x
      end if
    end for
  end while
  return (p_i, p_j)
    s.t. $d(p_i, p_j) < d(p_m, p_n)\ \forall\ m, n \neq i, j$ and $p_{i,m} \in C^+, p_{j,n} \in C^-$
end procedure

Fig. 2. Basic SVM Game Algorithm

approach to SVM and subsequently show how its innate parallelism allows for it to be implemented in a distributed manner analogous to the Cascade SVM approach.

3. SVM Game

With a desired outcome or a goal in mind, game theoretic mechanism design develops a framework defining player actions and the effect of these actions in efforts to attain the desired goal [6]. Using the geometric SVM learning paradigm as a desired goal, we have developed an iterated game to identify which data patterns are closest to the opposing class and thus define the position and shape of the resulting discriminant [7]. Our SVM Game is a two player iterated game where the data patterns are the players. As a Condorcet method, our game evaluates pairwise interactions between data points [8]. Each iteration of the game randomly selects two players from the same class and one data pattern from the opposing class. The pattern from the opposing class is not a player in the game, but rather provides a reference to determine which player is closer to the opposing class [9]. In canonical SVM, an alpha value (α) is a scalar multiplier of the support vectors. All data points initially start with the same finite amount of α, and through optimization the α is redistributed to the support vectors in amounts corresponding to their influence on the discriminant. Likewise, in our game, each player (data point) starts with an initial (equal) quantity of α. For each iteration of the game, competing players pass or hold a percentage of their α. Individual players do not choose which action to take (pass or hold), but rather their actions are dictated by their proximity to the reference point from the opposing class. In this sense, rather than players choosing a strategy, their actions correspond to innate properties of the players [10]. Fig. 2 shows the basic SVM Game algorithm just described.

Additionally, as an extension to the basic SVM Game, a coalitional SVM Game provides stability as well as a means of addressing non-linear problems. In this game variant, each player (data point) has a coalition partner which is an affiliation of a data point with a single member of the opposite class believed to be the closest member of the opposing class based upon data seen so far. Coalition partners are one-way pairings which may be many to one. Effectively, this builds coalitions within a given class of the grouping of like-minded players who all agree upon the preferred (closest) player of the opposing class. Every iteration of the coalitional SVM game allows each interacting player to consider the relative distance to both the reference point from the opposing class as well as their coalition partner. When all players in a given class form the same coalition, they are in agreement as to which player is the closest point amongst the opposing class and this unanimous Condorcet winner allows a linear discriminant to be constructed if both classes form single coalitions. If a unanimous decision cannot be reached this illustrates that a Condorcet winner does not exist and rather a non-linear solution is needed. In lieu of an unanimous Condorcet winner, rather

Algorithm 2 Coalitional SVM Game

1: **procedure** SVM_GAME$_C(S)$ ▷ Sets $C^+, C^- \in S$
2: **for all** $p_i \in S$ **do** ▷ Initialization
3: $p_i \leftarrow$ Initial Coalition Parter from Opposite Class
4: **end for**
5: **while** iterations≠desired iterations **do**
6: **for each** class C^+ and C^- **do**
7: $p_x \leftarrow random(C^1)$
8: $p_y \leftarrow random(C^1)$
9: $p_r \leftarrow random(C^2)$
10: Coalition$(p_x) \leftarrow \min($d(p_x, p_r),d$(p_x, Coalitiona(p_x))$,d$(p_x, Coalition(p_y)))$
11: Coalition$(p_y) \leftarrow \min($d(p_y, p_r),d$(p_y, Coalitiona(p_y))$,d$(p_y, Coalition(p_x)))$
12: **if** d$(p_x$,Coalition$(p_x)) \leq$ d$(p_y$,Coalition$(p_y))$ **then**
13: $p_x \leftarrow \alpha$ from p_y
14: **else**
15: $p_y \leftarrow \alpha$ from p_x
16: **end if**
17: **end for**
18: **end while**
19: **return** (p_i, p_j)
20: s.t. $d(p_i, p_j) < d(p_m, p_n)\ \forall\ m, n \neq i, j$ and $p_{i,m} \in C^+, p_{j,n} \in C^-$
21: **end procedure**

Fig. 3. Coalitional SVM Game Algorithm

the irreducible coalitions constitute Smith Sets which are a partitioning of the global problem such that within each of the Smith sets there is a local Condorcet winner [11]. Since each Smith Set consists of a local unanimous Condorcet winner, a global non-linear solution may be constructed by the composition of these local solutions. The coalitional SVM Game algorithm is shown in Fig. 3, and for more details regarding the SVM Game see [7].

4. MapReduce SVM Game

Since the SVM Game is innately distributed, it is a natural extension to apply the SVM Game to a parallel and distributed computational environment. MapReduce is a programming model for processing parallelizable problems across huge datasets using a large number of nodes [12]. First, in the "Map" step the master node takes the input and divides it into smaller sub-problems which are then distributed to the worker nodes. The worker nodes then process the smaller problems, and pass their individual answers back to the master node. In the "Reduce" step the master node collects the answers to all the sub-problems and combines them in some way to form the answer to the original problem. The MapReduce programming model is illustrated in Fig. 4.

The SVM Game is directly amenable to implementation in a MapReduce programming model paradigm. Game iterations are independent of one another in the sense that only the players involved need to communicate and they need not send the result of an interaction to a centralized authority. By partitioning the overall problem into a desired number of Mapped smaller sub-problems, each of these partitions may play the SVM Game to yield local winners. In the Reduce step, the local winners from the Map partitions are combined, and then the game is played on these points to yield the global solution (analogous to the Cascade SVM approach for canonical SVM). Fig. 5 depicts the general MapReduce SVM Game algorithm. The Map step occurs in lines 2 through 5 in which a partition function generates the sub-problems which may then played in parallel by the SVM Game. The Reduce step occurs on lines 6 and 7 in which the results of the Map partitions are combined and serve as the input to a subsequent call to the SVM Game algorithm. In lines 4 and 6 of the algorithm shown, the desired variant of the SVM Game class of algorithms may be evoked.

5. Results

Next we provide results showing the ability for the SVM Game to be partitioned in a MapReduce like manner. The upper left plot in Fig. 6 illustrates the full data used for this example. The two classes are

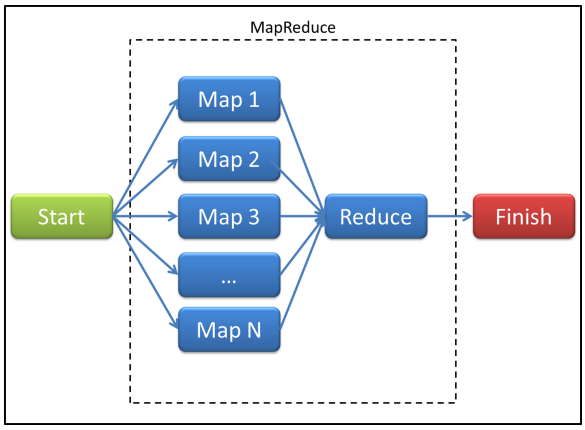

Fig. 4. MapReduce Distributed Computing Paradigm

Algorithm 3 MapReduce SVM Game

procedure SVM_GAME$_{MR}(S)$ ▷ Sets $C^+, C^- \in S$
  M $\leftarrow$ Partition(S)
  for each $m_i \in$ M **do** ▷ Executed in parallel
    MAP$_i \leftarrow$ SVM_GAME(m_i)
  end for
  Reduce $\leftarrow$ SVM_GAME(MAP)
  return Sets $A^+, A^- \in$ Reduce
end procedure

Fig. 5. MapReduce SVM Game Algorithm

linearly separable Gaussian distributions, each comprised of 50 data points. The MapReduce paradigm is not necessary for a problem of this size; however, for demonstrative purposes it allows the technique to be clearly demonstrated. For simplicity, we have opted to partition the overall class into five smaller problems, each consisting of ten data points per class. For this example, the partitioning is done randomly yielding the sub-problems shown by the remaining plots of Fig. 6. With each sub-problem having played the SVM Game independently, the red squares highlight the player from each class with the most alpha as a result of the game play. Alternatively, remaining coalitions may be returned as the result of the sub-problem game play. These winning points are returned in the Reduce step, at which point a subsequent round of the SVM Game is played only on these winning points to yield the data points from each class that produce the resulting discriminant. Fig. 7 depicts the result of the Reduce step. The upper half of the figure shows only the five local winners from the sub games and the resulting discriminant, while the lower half of the figure includes the full data set.

For this illustration we have run the Basic SVM Game with a fixed number of iterations. Doing so has the benefit of each sub-problem requiring the same amount of computational time and yielding one resulting answer that is returned for the Reduce step. Alternatively, the Coalitional SVM Game could also be used while allowing each sub-problem to run to convergence. Such an approach may result in both unequal processing time per node running the sub-problem as well as a variable number of coalitions returned from each sub-problem. As before, doing so confers the stability property as well as the ability to address non-linear data. Just as the irreducible coalitions comprising the Smith Set in a canonical implementation provide a means of generating a piecewise linear discriminant, likewise any irreducible coalitions of the sub-problems may be included in the final game play of the Reduce step and all resulting irreducible coalitions may then shape the piecewise linear solution to the global problem. The same pre-processing approaches previously

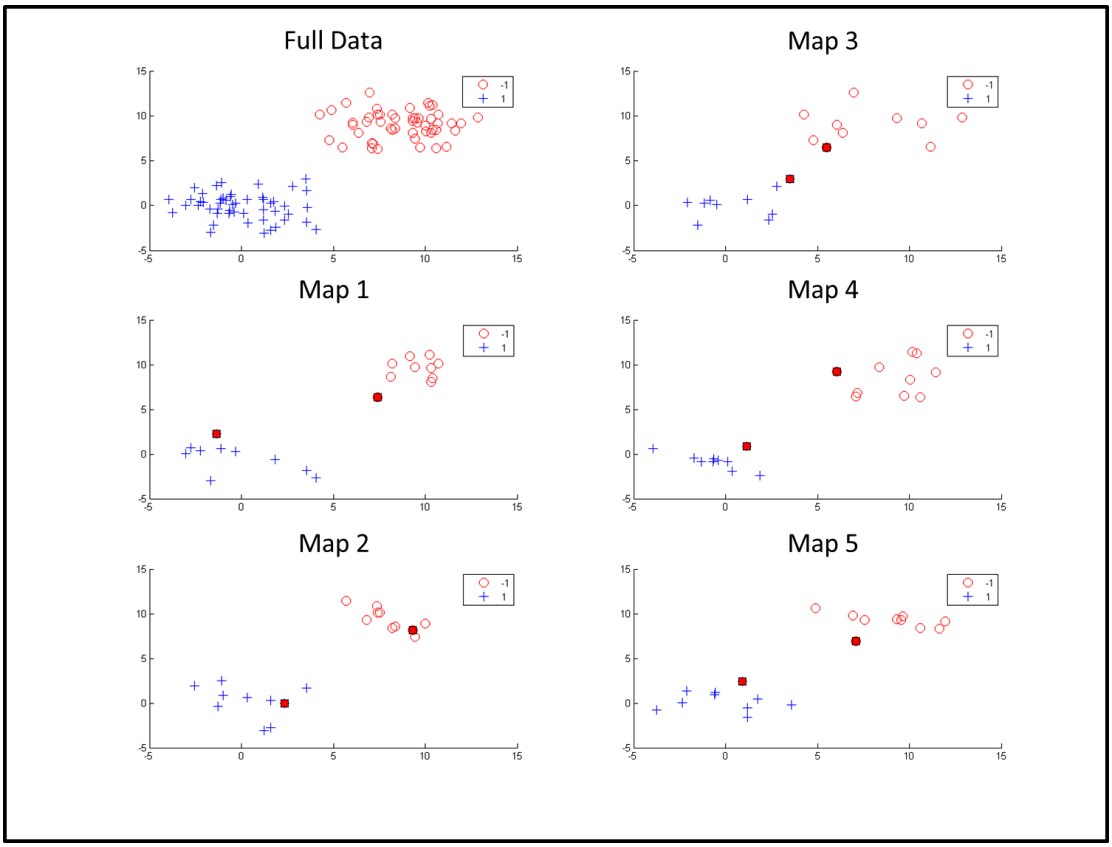

Fig. 6. Distributed MapReduce Paradigm Proof of Concept Example

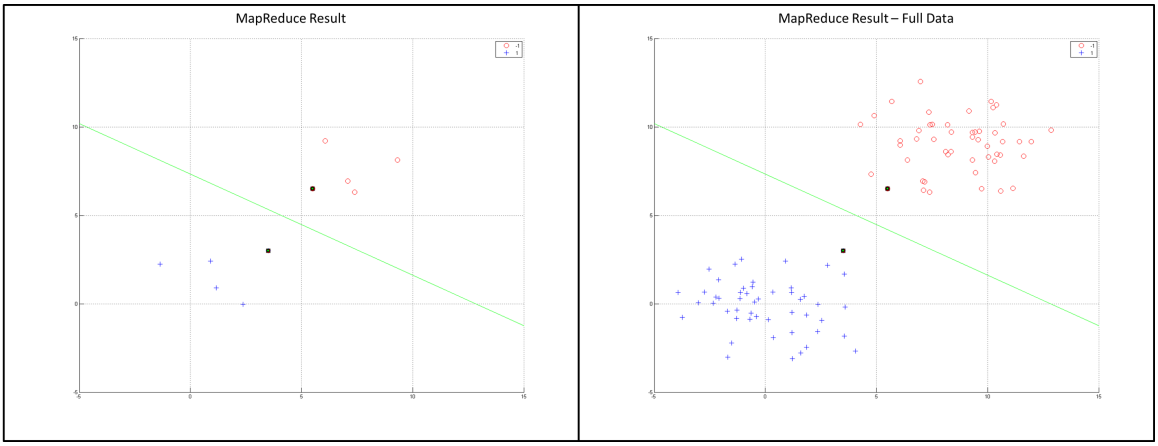

Fig. 7. Resulting Discriminants from Playing SVM Game on Results of Map Partitions

described may be applied prior to the Map partitioning step to address overlapping, noisy, or inconsistent data.

As a caveat, to guarantee each local Map sub-problem does not yield an errant solution analogous to a local minima, each partition must play against the complete opposing class. However, if representative samplings may be generated from the opposing class, a smaller sub-sample of the opposing class may be used to reduce the memory requirements for the Map partitions. It is also possible that some local errors may be corrected for in the subsequent Reduce step combining the results of the individual sub problems. For example, by examining the individual results of the Map sub-problems shown in Fig. 6, one can see that each Map sub-problem did not all yield optimal results. In particular, consider Map 2 in the lower left. If more game iterations are run, the positive (blue plus) class will select the data point to the right and up from the currently selected point marked by the red box. This point is clearly closer to the opposing class, but this discrepancy ultimately did not degrade the final result because regardless of which of these points is selected, either is further from the negative (red circle) class than the winning positive point of Map 3 which is the closest point on the convex hull of the entire class and is identified as such in the Reduce step.

6. Conclusion & Future Work

We are not the only ones to apply SVM to the MapReduce distributed programming paradigm. Sun et al. and Catak et al. have applied SVM to Twitter and Hadoop MapReduce programming models respectively [13] [14]. These approaches apply the Cascade SVM approach to the MapReduce paradigm explicitly, whereas the original work by Graf et al. describes how to partition SVM problems such that they may be solved either in sequential segments on a single processor (with the benefit of making each sub-problem computationally tractable), or distributed across multiple processors in a canonical computing paradigm.

Fundamentally, playing a MapReduce SVM Game is simply a means of constraining the game iterations played with the benefit of allowing for the game iterations to be played in parallel. The game iterations played on the Map partitions are simply individual game iterations whose possible reference points and opposing players are selected from a sub-population rather than from the overall distribution. Likewise, in the Reduce step, playing further game iterations only on the results attained from the individual Map games is analogous to selectively choosing players rather than doing so uniformly. Thus, conceptually the MapReduce programming paradigm can be employed such that it does not impact the SVM Game play mechanisms but rather only constrains the player selection process, and does so in a manner that allows for the game to be implemented on parallel compute clusters.

As future work, we plan to investigate the trade-offs associated with how the distributed problem is constructed. As presented here (and depicted in Fig. 4) we have used a single layer hierarchy where all the Maps are played in parallel and combined in a single Reduce step. Alternatively, a multi-layer hierarchy could also be employed such that intermediate Reduce steps could be employed at intermediate layers. A potential benefit of such a hierarchical approach would be smaller Map sub-problems requiring fewer game iterations be played. Taken to the extreme, Map partitions would consist of only two data points, and the subsequent game play would eliminate a data point from consideration with each Map. The resulting hierarchy would require a logarithmic number of layers to combine the hierarchy of all two point sub-problems. Operating upon two data points at a time to refine the overall solution would be similar to the Sequential Minimal Optimization (SMO) approach of chunking a quadratic programming problem into pairs of data points which sequentially work towards the global solution [15]. However, this approach would natively allow all the pairwise evaluations at a given step to occur concurrently rather than sequentially. In conjunction with exploring the trade-offs of how to partition problems, we plan to benchmark the MapReduce SVM Game on larger problems than the demonstrative proof of concept illustration we have shown here.

Additionally, we would also like to investigate sampling strategies for how Maps are partitioned. The simplest approach is to do so randomly, however domain knowledge or more sophisticated sampling techniques which can guarantee properties such as how representative a sample is with respect to the overall distribution may prove beneficial.

Training large neural network models such as deep learning networks is a slow, computationally intensive process. If the learning phase can be parallelized that is one possible means of improvement, but not all

learning algorithms are well suited for parallelization. Various research efforts have employed this approach such as parallelized stochastic gradient descent by Zinkevich et al. which allows gradient descent based learning algorithms to be parallelized [16]. An additional possible benefit of the research we have presented here is to provide an alternative approach to train up a large neural network. Scholkopf et al. have shown how the canonical SVM algorihtm may be used to train a Radial Basis Function (RBF) network [17], so likewise another extension of this work would be to use our MapReduce SVM Game to train a large RBF or other deep network.

Many game-theoretic interactions are not single instance occurrences, but rather consist of repeated game play allowing players to adjust strategies as well as compete against multiple opponents. Effectively, a game-theoretic approach is also amenable for addressing the evolving nature of big data problems. Rather than requiring extensive retraining, additional game play iterations can be run to update the model as addition data is received. This is analogous to the ability to partition and recombine SVM Game interactions, as we have shown here in the MapReduce paradigm, but extended such that through repeated game iterations the problem decomposition occurs over time as additional data is received. For a description of how repeated iterations of the SVM Game allow the algorithm to address non-stationary data such as an evolving data stream see [18].

Leveraging the innate distributed nature of game-theoretic interactions, we have presented an alternative approach to SVM which is directly parallelizable in a MapReduce programming paradigm and furthermore provides an alternative strategic perspective to the problem.

Acknowledgment

This research was possible in part by LDRD program support from Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energys National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] B. Alcott, Jevons' paradox, Ecological economics 54 (1) (2005) 9–21.
[2] V. N. Vapnik, V. Vapnik, Statistical learning theory, Vol. 2, Wiley New York, 1998.
[3] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification (2nd Edition), 2nd Edition, Wiley-Interscience, 2001.
[4] K. P. Bennett, E. J. Bredensteiner, Duality and geometry in svm classifiers, in: Proceedings of the Seventeenth International Conference on Machine Learning, ICML '00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000, pp. 57–64. URL http://dl.acm.org/citation.cfm?id=645529.657972
[5] H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, V. Vapnik, Parallel support vector machines: The cascade svm, in: Advances in neural information processing systems, 2004, pp. 521–528.
[6] N. Nisan, Introduction to mechanism design (for computer scientists), Algorithmic game theory 209 (2007) 242.
[7] C. M. Vineyard, G. L. Heileman, S. J. Verzi, A Game Theoretic Coalitional Support Vector Machine Classifier, under Review (2015).
[8] J. Kleinberg, Networks, Crowds, and Markets, Cambridge University Press, 2010.
[9] C. M. Vineyard, G. L. Heileman, S. J. Verzi, R. Jordan, Game theoretic mechanism design applied to machine learning classification, in: Cognitive Information Processing (CIP), 2012 3rd International Workshop on, IEEE, 2012, pp. 1–5.
[10] K. Mitchell, J. Ryan, Game theory models of animal behavior.
[11] J. H. Smith, Aggregation of preferences with variable electorate, Econometrica 41 (6) (1973) 1027–41.
[12] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.
[13] Z. Sun, G. Fox, Study on parallel svm based on mapreduce, in: International Conference on Parallel and Distributed Processing Techniques and Applications, Citeseer, 2012, pp. 16–19.
[14] F. Ö. Çatak, M. E. Balaban, A mapreduce based distributed svm algorithm for binary classification, Turkish Journal of Electrical Engineering & Computer Science.
[15] J. Platt, et al., Sequential minimal optimization: A fast algorithm for training support vector machines.
[16] M. Zinkevich, M. Weimer, L. Li, A. J. Smola, Parallelized stochastic gradient descent, in: Advances in Neural Information Processing Systems, 2010, pp. 2595–2603.
[17] B. Scholkopf, K.-K. Sung, C. J. Burges, F. Girosi, P. Niyogi, T. Poggio, V. Vapnik, Comparing support vector machines with gaussian kernels to radial basis function classifiers, Signal Processing, IEEE Transactions on 45 (11) (1997) 2758–2765.
[18] C. M. Vineyard, S. J. Verzi, C. D. James, J. B. Aimone, G. L. Heileman, Repeated play of the svm game as a means of adaptive classification, in: IJCNN 2015, IEEE, 2015.