Fast Training on Large Genomics Data using Distributed Support Vector Machines

Nawanol Theera-Ampornpunt, Seong Gon Kim, Asish Ghoshal, Saurabh Bagchi, Ananth Grama, Somali Chaterji
Purdue University
West Lafayette, Indiana, USA
Email: {ntheeraa, kim1871, aghoshal, sbagchi, ayg, schaterj}@purdue.edu

***Abstract*—The field of genomics has seen a glorious explosion of high-quality data, with tremendous strides having been made in genomic sequencing instruments and computational genomics applications meant to make sense of the data. A common use case for genomics data is to answer the question if a specific genetic signature is correlated with some disease manifestations. Support Vector Machine (SVM) is a widely used classifier in computational literature. Previous studies have shown success in using these SVMs for the above use case of genomics data. However, SVMs suffer from a widely-recognized scalability problem in both memory use and computational time. It is as yet an unanswered question if training such classifiers can scale to the massive sizes that characterize many of the genomics data sets. We answer that question here for a specific dataset, in order to decipher whether some regulatory module of a particular combinatorial epigenetic "pattern" will regulate the expression of a gene. However, the specifics of the dataset is likely of less relevance to the claims of our work.**

We take a proposed theoretical technique for efficient training of SVM, namely Cascade SVM, create our classifier called EP-SVM, and empirically evaluate how it scales to the large genomics dataset. We implement Cascade SVM on the Apache Spark platform and open source this implementation[1]. Through our evaluation, we bring out the computational cost on each application process, the way of distributing the overall workload among multiple processes, which can potentially execute on different cores or different machines, and the cost of data transfer to different cores or different machines. We believe we are the first to shed light on the computational and network costs of training an SVM on a multi-dimensional genomics dataset. We also evaluate the accuracy of the classifier result as a function of the parameters of the SVM model.

Keywords—machine learning, classifier training, computational genomics, computational cost, network cost.

I. INTRODUCTION

The process of transcription, which is the process of copying DNA to RNA, in multicellular organisms is a complex and tightly-regulated process, with discrete modules taking on specific roles. In this decision-control gene-regulatory circuit, there are regions where the process of gene expression initiates (*aka*. transcription start sites, TSS) and those that result in altered gene expression, such as enhancers (enhance gene expression), or repressors of gene expression. Enhancers are classically defined as cis-acting DNA sequences that can increase gene expression. Historically, their identification has been challenging because they are interspersed in the 98% non-coding region of the human genome, resulting in a large search space of billions of DNA base pairs that constitute the genetic alphabet. Further, they may be located distant to the genes that they regulate, stemming from the three-dimensional DNA "origami" [2]. Higher-order three-dimensional organization of chromatin facilitates physical interactions between enhancers and their target promoters, wherein the enhancers may target genes or non-coding RNA on the same chromosome (in cis) or on different chromosomes (in trans) [3]. Rather than simple on-off switches, enhancers are thought to function as transcriptional rheostats, fine-tuning gene transcription levels [4]. In any given mammalian cell type, the number of hypothesized enhancer elements ranges from 50,000 to 100,000, as opposed to about 20,000 protein-coding genes, thus easily outnumbering the number of genes [5]. These enhancer variants go on to influence transcriptional output, thereby offering a mechanistic basis to explain their association with risk for many common diseases. Thus, genome-wide identification of functionally active enhancers is necessary to understand the expression of genes, as well as to map out developmental and disease-related, processes [4, 6].

Machine-learning methods have been used to predict the genome-wide locations of enhancers [7-10]. Essentially, the computational techniques predict enhancer locations from the DNA sequences, captured by the epigenetic modifications in the vicinity of the binding site, such as histone modifications. However, it has been found that training these models is computationally expensive and all previous work has made the simplification to train on only a small part of the experimentally available dataset. On the theoretical computational side, researchers have designed a distributed version of the common Support Vector Machine (SVM) classifier, called Cascade SVM [1]. It works by splitting the dataset into multiple partitions, working on each partition in parallel, and then in a binary tree-like manner, combining the Support Vectors (SVs) from each partition. In this paper, we apply the Cascade SVM classifier, as a

[1] Our open source implementation is available at: https://bitbucket.org/cellsandmachines/kernelsvmspark

distributed SVM implementation, to handle the enhancer prediction workload. We call our approach ***E***nhancer ***P***rediction through ***SVM***, or **EP-SVM**. The Cascading approach is guaranteed to converge to a global optimum, with multiple iterations through the binary tree or Cascading loop, but even a single iteration affords good generalization. In our work, we find a single iteration already gives good classifier performance, as demonstrated by the precision-recall curve.

Our study brings out that EP-SVM makes it feasible to train large-sized genomic datasets for an SVM classifier. A serial version of the classifier runs into long execution times, due to the high asymptotic complexity of training of an SVM ($O(n^3)$ for a commonly used algorithm) and the high memory demands. We also experimentally discover the relation between the number of partitions of the dataset in EP-SVM and the number of the available cores, for optimal performance. We bring out the fact that Cascade SVM does provide speedup even when a single core is used.

Our contributions in the paper can be summarized as follows:

1. We show how to build a machine learning (ML) classifier (an SVM classifier) for a biologically important use case, that is, prediction of enhancers—DNA elements that amplify gene expression and can be located at great distances from the target genes, making the enhancer prediction problem challenging. The ChIP-seq peak recognition patterns are non-deterministic and noisy and we find that SVM performs well for the workload, when we take care to decrease the diversity of the workload by training.
2. We show that a serial SVM is not able to handle training of even a fraction of the experimentally available dataset. We then go on to apply a previously theoretically proposed technique called Cascade SVM to the problem and adapt it to create our own computationally efficient classifier called EP-SVM.
3. We do a detailed empirical characterization of the effect of number of cores, communication cost, and number of partitions on the runtime of training of Cascade SVM.

The rest of the paper is organized as follows. In Section II, we give a brief description of the application and the dataset. Then, we give the necessary background on SVM and Cascade SVM. In Section IV, we describe our experimental strategy and experimental results. In Section V, we discuss the implications of our results. Then we conclude the paper.

II. Application and Dataset

A. Application context

In this work, we address the problem of genome-wide identification of enhancers, which are distal regulatory elements constituting a prominent class of non-coding drivers of gene expression. We call our protocol for predicting enhancers based on chromatin features in the human embryonic stem cell line H1, "EP-SVM", an acronym for "Enhancer Prediction using support vector machines (SVMs)" and further speed it up by implementing a distributed version of the algorithm. Enhancers are found to be frequently affected by GWAS (genome-wide association studies) variants. This indicates that the exploration of these enhancer elements can afford key insights into disease pathophysiology. The fact that enhancer sequences lack common sequence features and often reside far away from their target genes has made them difficult to identify through computational algorithms. Our efforts in this paper are driven by recent advances in epigenomic profiling methods that have uncovered enhancer-associated chromatin features, in different cell types and organisms. We use as our data sources—the NIH Roadmap Epigenomics Project, the ENCODE Project, and NCBI's Gene Expression Omnibus, all of which provide freely-available repositories of epigenomics maps.

Global prediction of enhancers will enable the rapid prediction of enhancers in new cell types, without the need for a separate training set for every new cell type. Once these putative enhancers in different cell types have been identified, it will be important to link them to the specific genic promoters that they regulate. This kind of interaction is complex with many-to-many associations, wherein one enhancer can regulate the expression of multiple genes, and multiple enhancers can affect the same gene, acting in synergy. Such predictions will lower the cost of generating enhancer and gene interaction data using experimental methods such as chromatin conformation capture-based protocols, which provide direct evidence for such physical interactions [11].

B. Dataset

The development of our distributed EP-SVM is motivated by the availability of data from large scale projects, such as the ENCODE project [12], which has annotated 400,000 putative human enhancers, with the current estimate of enhancer numbers being over a million [13]. Another extensive database is the NIH Roadmap Epigenomics Project [14] that also provides publicly-available epigenomics maps, complementary to ENCODE. In addition, the NCBI's Gene Expression Omnibus (GEO) repository [15], also hosts much previous work and data on enhancer prediction. The above three datasets were used as sources for our training and test datasets. The computationally minded reader may skip the rest of this dataset section. It is meant here to facilitate reproducibility of our results and hence gives exact details.

24 histone modifications─H2AK5ac, H2BK120ac, H2BK12ac, H2BK15ac, H2BK20ac, H2BK5ac, H3K14ac, H3K18ac, H3K23ac, H3K27ac, H3K27me3, H3K36me3, H3K4ac, H3K4me1, H3K4me2, H3K4me3, H3K56ac, H3K79me1, H3K79me2, H3K9ac, H3K9me3, H4K20me1, H4K5ac, and H4K91ac─in human embryonic stem cells (H1) were chosen from the NIH Roadmap Epigenomics repository as biologically relevant to the presence or absence of enhancers and used as input. The ChIP-seq reads of these histone modifications were binned into 100 bp intervals and normalized against its corresponding inputs by using an

RPKM (reads per kilobase per million) measure [15]. Multiple replicates of histone modifications were used to minimize batch-related differences, and the RPKM-levels of the replicates were averaged to produce a single RPKM measurement per histone modification.

Transcription factor binding sites (TFBSs): Transcription factors (TFs) play a central role in regulating gene expression. Binding of TFs to their target loci, such as enhancers, is a key step of activating (in the case of enhancers) or repressing (in the case of repressors) gene expression. Determination of TFBSs is an important, albeit challenging, problem because the DNA segments recognized by TFs are often short and dispersed in the genome, and are often dependent on the context or physiological state of the cell [16]. Chromatin immunoprecipitation (ChIP)-chip [17] and, more recently, ChIP-seq have become powerful tools to determine TFBSs at a genome-wide scale in different cell lines [18, 19]. Thus, in this study, we attempt to use the ChIP-seq signals obtained from abundantly encountered TFs in the embryonic stem cell line, H1, specifically the core TFs, Oct4, Sox2, and Nanog, found in embryonic stem cells. Genome-wide binding data for transcription factors Oct4, Sox2, and Nanog, in H1 were obtained using ChIP-seq and deposited under accession numbers GSE37858, GSE18292, and GSE17917 respectively, as described in [20]. MACS software [21] was used to call the TFBS peaks. ChIP-seq input files were used as background and parameters of mfold = 20 and default p-value cutoffs were used.

These TFBSs were chosen as enhancers for our method and labeled as positive, while transcription start sites (TSS) where chosen as non-enhancers and labeled as negative. Finally, we labeled each 100 bp bin site such that if the site lies within 2.5 kb of either DNase-I hypersensitivity site (DHS), TFBS, or CBP or p300 co-activator binding site, it was labeled positive. Otherwise, it was labeled negative.

Finally, the dataset was subsampled uniformly to generate a smaller dataset, which is then divided into disjoint training and testing sets. Both sets are 135 MB with 5,538 negative samples and 25,219 positive samples each.

III. Background: SVM and Cascade SVM

Here, we provide the relevant background for the classifier that we use in this work—Support Vector Machine (SVM) and the parallel version of SVM called Cascade SVM. The description of the basic SVM is derived in part from [1]. To the best of our knowledge, we are the first to use a distributed SVM classifier, specifically Cascade SVM, toward the computational prediction of regulatory DNA sequences. We have used Cascade SVM in our recent work [22] on a different biological dataset, specifically CLIP-seq datasets for the prediction of microRNA target sites on messenger RNA.. From our work, we find that the large size of genomics datasets makes it particularly attractive to use a cascading SVM.

A. Support Vector Machine

Support Vector Machines (SVMs) are a learning method used for binary classification. The basic idea is to find a hyperplane which separates the *d*-dimensional data perfectly into its two classes.

Mathematically, we are given l training examples [23]; $i = 1, \ldots, l$, where each example x_i has d inputs ($x_i \in \mathbf{R}^d$), and a class label with one of two values ($y_i \in [24]$). Now, all hyperplanes in $\mathbf{R}^d$ are parameterized by a vector ($\mathbf{w}$), and a constant (b), expressed in the equation

$$\mathbf{w}.\mathbf{x} + b = 0 \tag{1}$$

$\mathbf{w}$ is in fact the vector orthogonal to the hyperplane. Given such a hyperplane ($\mathbf{w}$, b) that separates the data, this gives the function

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}.\mathbf{x} + b) \tag{2}$$

That correctly classifies the training data and hopefully other testing data that it has not yet seen. However, a given hyperplane represented by ($\mathbf{w}$, b) is equally expressed by all pairs $\{\lambda\mathbf{w}, \lambda b\}$ for $\lambda \in \mathbf{R}^+$. So, we define the chosen hyperplane to be the one that separates the data from the hyperplane by a "distance" of at least 1. Such hyperplanes satisfy the following equation

$$y_i(\mathbf{x_i}.\mathbf{w} + b) \geq 1 \tag{3}$$

However, we find that a linear hyperplane does not seem to separate the two classes in our dataset well, such that, after extensive tuning, it gives a precision of 84% and recall of 57%. Contrast this with the fact that trivially, always predicting a positive would have given a precision of 80% and recall of 100%. Hence, to find a higher order hyperplane to separate the positive and the negative samples, we used a standard "kernel trick" attached to the standard SVM technique.

The kernel SVM transforms the data points to an arbitrary higher dimensional space via the following kernel function:

$$K(\mathbf{x_a}, \mathbf{x_b}) = \phi(\mathbf{x_a}).\phi(\mathbf{x_b}) \tag{4}$$

Where we are computing the dot product of the two transformed vectors, $\mathbf{x_a}$ and $\mathbf{x_b}$. We use the popular Radial Basis Function (RBF) kernel, which is defined by the equation

$$K(\mathbf{x_a}, \mathbf{x_b}) = \exp(-\gamma \left\| \mathbf{x_a} - \mathbf{x_b} \right\|^2) \tag{5}$$

The parameter γ defines how far the influence of a single support vector reaches, with low values meaning 'far' and high values meaning 'close'.

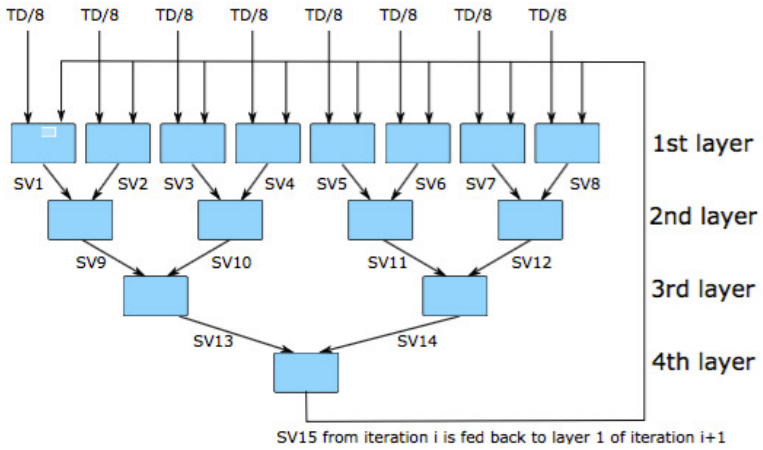

Fig. 1. Schematic showing Cascade SVM (adapted from [1]). TD: Training Data, which is split and fed into different model learners that can operate in parallel. The Support Vectors (SVs) from different blocks in each layer are combined to create a larger number of aggregated SVs.

We find that kernelization significantly expands the running time of our classifier, to the point where it becomes unusable for large sized datasets. For example, through evaluation of training time on smaller samples of the dataset and extrapolating from that, we find that training the kernel SVM serial version would take 45.4×10^3 years for the entire dataset![2]

B. *Distributed SVM*

We therefore use a previously proposed technique called Cascade SVM that distributes the process of training of an SVM [1]. The basic approach is schematically represented in Figure 1. The training dataset is first split into a number of segments. Then, an SVM is trained independently for each of the segments. Since the support vectors in each segment might not be global support vectors, the support vectors from two segments are combined by passing them through another SVM to filter out non-support vectors. This proceeds in a tree-like fashion until only one set of support vectors remains. The support vectors can then be fed back to the first layer and multiple iterations over the cascade of SVMs is guaranteed to take the solution to the global optimum, and often, only one iteration over the cascade is enough to produce a sufficiently good solution. We developed (and open sourced) a general-purpose, scalable, and memory efficient implementation of Cascade SVM on top of Apache Spark [25] that can be used to train kernel SVMs for large problems. We did not have to limit the number of SVs in our evaluation; the number of SVs turned out to be a reasonable 28.2% of the number of data points. We also got high prediction accuracy with a single iteration of execution of the cascade.

[2] The entire dataset is of size 300 GB and we use the standard SVM training algorithm that has complexity $O(n^3)$, where n is the number of training samples.

IV. Experiments

We design three experiments to evaluate the performance of EP-SVM training. The description of each experiment is shown in Table I.

Table I. Summary of the Experiments

Experiment	Description
1	Evaluates prediction accuracy in terms of precision and recall
2	Finds the relationship between number of partitions and training time with the numbers of machines and cores fixed
3	Shows the speedup gains when EP-SVM is run in a distributed manner

The SVM parameters used in the experiments are as follows: C (regularization parameter) = 1, radial basis function kernel parameter $\gamma = 1/480$. Class weight of positive data points relative to negative data points = 0.2196, for experiments 2 and 3 (for experiment 1, this is a control parameter that is varied). This means that the ratio of the penalty for mispredicting a positive sample to the penalty for mispredicting a negative sample is 0.2196.

A. *Experiment 1*

We first start by evaluating the prediction accuracy of the learned model. Note that Cascade SVM is guaranteed to give the same model as serial SVM, so prediction accuracy will not be affected by the number of partitions and machine configuration. One of the training parameters is the class weight parameter, which controls the relative importance of the two classes. By varying this parameter, we can adjust the tradeoff between precision and recall. Precision is defined as $\frac{TP}{TP+FP}$ and recall is defined as $\frac{TP}{TP+FN}$, where TP, FP, and FN denote the numbers of true positives, false positives, and false negatives, respectively. In plain words, precision is defined as, out of the samples that our classifier predicts as positive, how many are actually positive. Recall is, out of the total number of positive samples, how many are predicted by our classifier as positive samples. The resulting precision-recall curve is shown in Figure 2.

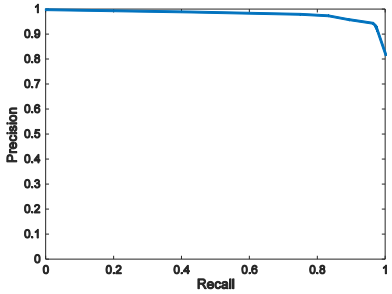

Fig. 2. Prediction accuracy of our classifier. The precision-recall curve is generated by varying the class weight parameter that gives the relative importance of predicting the positive and the negative samples.

With the two classes' weights being equal, precision is 0.9458 and recall is 0.9619. Incidentally, this is a higher accuracy than that reported in prior literature, in more biologically minded publications [7, 8]. For example, DEEP-EN and RFECS, the two latest approaches which use sophisticated statistical models, the highest recall is less than 93% and 88%. However, they try to predict a wider set of enhancer signatures – we use a subset, the TF-type enhancers – and this may also explain their lower accuracy. We can conclude however that the SVM is an appropriate classifier for this problem domain considering the creditable precision and recall scores.

B. *Experiment 2*

Cascade SVM works in multiple iterations, with each iteration containing multiple rounds. The number of rounds is $\lceil Log_2(P) \rceil + 1$, where P is the number of partitions. With smaller P, each partition is larger, and because SVM's training time complexity is $O(n^3)$, the first round can become the bottleneck. On the other hand, larger P moves the bottleneck away from the first round, to the last round where the number of support vectors limits the concurrency. After some point, increasing P further only increases the number of rounds without alleviating the bottleneck. Therefore, for a fixed configuration of machines, there is an optimal number of partitions.

For this experiment, we fix the number of machines as 2, and the number of cores per machines as 1. Both training and testing datasets are 135 MB in size. The number of partitions is varied from 2 to 48. The results are shown in Figure 3.

From the results, training time keeps decreasing as the number of partitions increases. This indicates that the inflection point, for when increasing the number of partitions actually hurts performance, is beyond 48 partitions, i.e., beyond 24X number of cores. Training time decreases more rapidly initially (at lower numbers of partitions) and the results start to show the diminishing value of dividing the data into more partitions as the number of partitions increases.

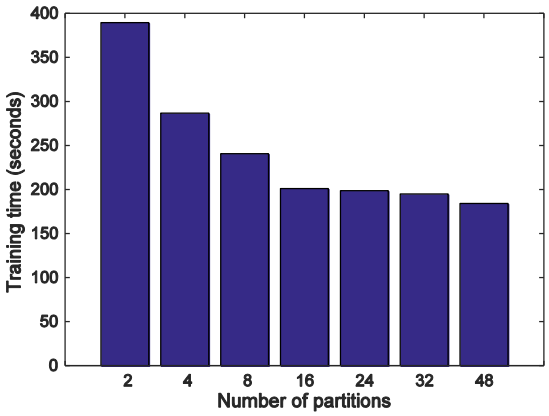

Fig. 3. Effect of number of partitions on training time. Training is done on 2 machines using 1 core on each machine in all cases.

Further, this experiment uses "only" 135 MB for the training set. There is a vastly larger amount of experimental data available, for this specific problem (with TFs and TSSs and histone modification signatures). Considering that training an SVM has time complexity of $O(n^3)$, it would be obvious that the advantage of EP-SVM over serial execution will continue to increase and become more prominent for the larger training data volumes.

C. *Experiment 3*

This experiment aims to measure speedup of EP-SVM compared to serial SVM for model training. To make results comparable, we run serial SVM by running Cascade SVM with the number of partitions and the number of cores set to 1. For the distributed configuration, the number of partitions is kept to be the same as the number of cores used. We divide the partitions equally among the machines initially and then wrap around. Thus, for our experimental cluster of 8 machines, when we have 8 partitions, we use 1 core on each machine; with 16 partitions, we use 2 cores on each machine. We report training time as well as speedup, defined as $T_{serial} / T_{dist(n)}$, where T_{serial} is the training time in the serial case, and $T_{dist(n)}$ is the training time for the distributed case, with n cores.

The results are shown in Figure 4. Training time keeps decreasing as we increase the number of cores. Going from 1 to 2 cores results in a super-linear speedup, due to partitioning. Beyond 2 cores, however, the speedup is sub-linear, as the training time in the final round starts to dominate. The final number of support vectors is 8,673, or 28.20% of the number of training data points, and this limits the maximum speedup that can be gained using EP-SVM.

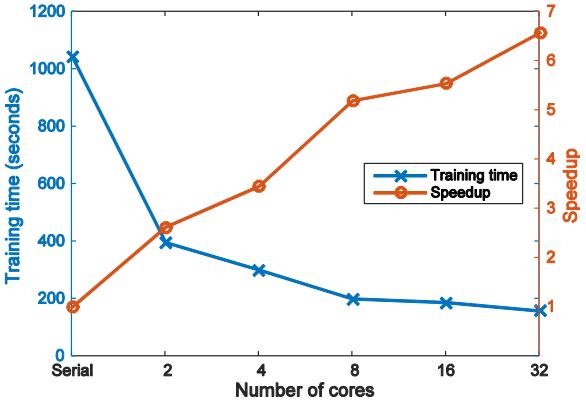

Fig. 4. Training time and speedup of EP-SVM compared to serial SVM. The number of partitions is the same as the number of cores used in all cases.

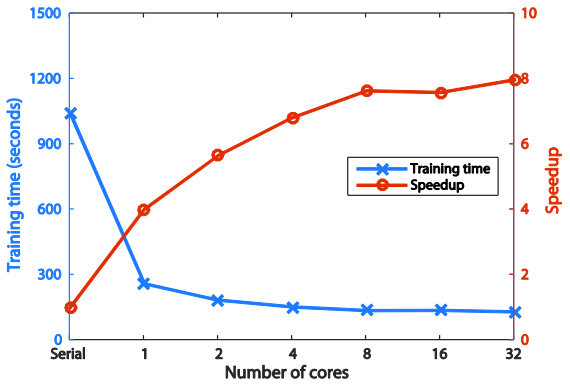

Fig. 5. Training time and speedup of EP-SVM compared to serial SVM. The number of partitions is kept at 96 in all cases.

We repeat this experiment but now with number of partitions = 96, which is multiple times larger than the number of cores (at least 3X, which occurs at the maximum core count of 32). The results are shown in Figure 5. The choice of a higher number of partitions is driven by our empirical insight from experiment 2 that keeping the number of partitions higher than the number of cores gives better performance. We do see better speed up here—e.g., using all 32 cores in our cluster, we see a speed up 7.98 compared to 6.58 in the earlier case.

We expected that the cost of distributing across the machines will bring down the speed up of EP-SVM. However, due to the nature of the experimental cluster that we use, a tightly coupled cluster of 8 machines connected by a 1 Gbps link, the network speeds are high enough and the number of Support Vectors is low enough that communication does not pose a bottleneck. Instead, the serial computation at the last level of the EP-SVM turns out to be the bottleneck that limits the speedup that can be achieved.

V. Discussion

We found that despite the speedups obtained through EP-SVM, it still takes too long to perform training on the entire experimentally generated dataset, which is approximately 300 GB. We believe that it is needed in this case to first cluster the dataset and then build a relatively simple model for each cluster. Such clustering can be done using a biological insight. For example, there are several different kinds of Transcription Factors, such as, NANOG, OCT4, and SOX2, and it may be possible to build an EP-SVM with a smaller number of Support Vectors if we looked at only one type of TF in one model.

Our results point to the fact that for EP-SVM, it is worthwhile to use a large number of partitions, larger than the number of cores that are available. At any one time, one core will execute one partition, but the larger number of partitions means that the different cores involved in the execution can load balance across the different partitions. A larger number of partitions means each partition is of smaller size, and therefore, a core that gets done with its current partition can go and pick up another partition from the queue of unprocessed partitions. This will reduce the amount of load imbalance among the different cores. Of course, taken to an extreme, the constant cost of processing a partition, such as,getting the data to the right core, starting execution, communicating the Support Vectors, will mean that the partition size should not be made smaller than a certain value. However, empirically, for our configuration, we did not encounter such situation.

It is an important question to gauge how generalizable our results are to other genomic datasets. The field of computational genomics is rapidly burgeoning and there have been a large number of models developed for diverse datasets. In this paper, we confronted the scalability of a widely used classifier for an important regulatory genomics problem. For different genomics problems, different models have been found to be accurate, such as, decision tree-based classifiers, Bayes net-based classifiers, etc. Our work here suggests that for the appropriate classifier for the specific problem, it would be important to adapt distributed versions of suitable classifiers. From our experience, it turned out that this adaptation is non-trivial. For one, it should consider the cost of distributing the data among multiple machines. Our experience shows that the theoretical speedups may not be realizable in practice, e.g., if the number of Support Vectors in EP-SVM does not come down to a small fraction of the total number of training samples. In such cases, in to order to bound the number of Support Vectors, biological insights should be used to cluster the training data, and then, create simpler models that can be combined, if necessary. The choice of the classifier may be guided, to some extent, by the availability of efficient distributed implementations. For example, in our earlier effort [26], we had obtained high accuracy for this problem using a Deep Neural Network (DNN). However, we did not find any usable or efficient distributed implementation of a DNN. This should spur the community to develop such efficient and scalable distributed machine learning implementations.

VI. Related Work

We have already referenced the foundational machine learning algorithms that we have adapted in our work. Here we refer to the two previous works on predicting enhancers that are closest in terms of using powerful statistical machine learning techniques.

Historically, computational identification of enhancers has proven to be challenging for several reasons [27]. First, the search space for enhancers is large—billions of DNA base pairs—scattered across 98% of the non-coding genome. Second, while enhancers regulate genes in *cis*, they do not display distinct locational or orientation-centric signals relative to the genes that they regulate—potentially located upstream, downstream, or even in introns of the genes that they modulate, often regulating multiple genes [28]. Enhancers function at a distance from their target genes via chromatin loops that bring the enhancers and target genes into proximity [29, 30], or via direct eRNA transcription from the enhancer DNA sequences [31]. Third, although a few attempts have been made to elucidate sequence-based signatures of enhancers [32, 33], they are very recent, and yet to be widely adopted.

Several computational methods that use histone modification signatures to identify enhancer regions have been developed. Won *et al*. proposed the usage of Hidden Markov Models (HMMs) to predict enhancers using three primary histone modifications [34]. Firpi *et al*. focused on the importance of recognizing the histone modification signals through data transformation and employed Time-Delayed Neural Networks (TDNNs) using a set of histone marks selected through simulated annealing [35]. Fernández *et al*. used Support Vector Machines (SVMs) on an optimized set of histone modifications found through Genetic Algorithms [9]. RFECS (Random Forest based Enhancer identification from Chromatin States) improved upon the limited number of training samples in previous approaches using Random Forests (RFs), in order to determine the optimal set of histone modifications to predict enhancers [20]. DEEP combined an ensemble of SVMs and an Artificial Neural Network to predict strong enhancers using multiple features derived from sequences and histone modifications [36].

VII. Conclusion

Genomics datasets tend to be large in size and their sizes are growing fast with the precipitous reduction in genomic sequencing costs. There is a huge push in the bioinformatics community to extract insights from the plethora of genomic data. Statistical machine learning techniques are being used in many problem contexts. In this paper, we show how to use a distributed variant of the popular classifier SVM, in a biologically important problem context, namely, whether a regulatory module called enhancer affects gene expression. We find that the distributed algorithm, called Cascade SVM, gives significant speedups over the serial version. However, the speedup saturates, stemming from the bottleneck of the computation that needs to be serially done at the last stage of the Cascade SVM. Our experimental results also show that it is useful to have a higher number of partitions of the data than the number of cores available to execute the workload, because it leads to better load balancing and utilization of compute resources. The communication bottleneck that has been seen in some applications of Cascade SVM does not materialize here since our cluster machines are well connected with fast network links. While we had set out to understand the computational and network costs of distribution, as a side benefit, we find that our classifier gives higher accuracy than previous machine learning schemes that were used for this biological problem.

VIII. References

1. Graf, H.P., Cosatto, E., Bottou, L., Dourdanovic, I., and Vapnik, V.: 'Parallel support vector machines: The cascade svm', in Editor (Ed.)^(Eds.): 'Book Parallel support vector machines: The cascade svm' (2004, edn.), pp. 521-528
2. Pennisi, E.: 'Inching toward the 3D genome', Science, 2015, 347, (6217), pp. 10-10
3. Teng, L., He, B., Wang, J., and Tan, K.: '4DGenome: a comprehensive database of chromatin interactions', Bioinformatics, 2015, 31, (15), pp. 2560-2564
4. Corradin, O., and Scacheri, P.C.: 'Enhancer variants: evaluating functions in common disease', Genome medicine, 2014, 6, (10), pp. 85
5. Kvon, E.Z., Kazmar, T., Stampfel, G., Yanez-Cuna, J.O., Pagani, M., Schernhuber, K., Dickson, B.J., and Stark, A.: 'Genome-scale functional characterization of Drosophila developmental enhancers in vivo', Nature, 2014, advance online publication
6. Farh, K.K.-H., Marson, A., Zhu, J., Kleinewietfeld, M., Housley, W.J., Beik, S., Shoresh, N., Whitton, H., Ryan, R.J.H., Shishkin, A.A., Hatan, M., Carrasco-Alfonso, M.J., Mayer, D., Luckey, C.J., Patsopoulos, N.A., De Jager, P.L., Kuchroo, V.K., Epstein, C.B., Daly, M.J., Hafler, D.A., and Bernstein, B.E.: 'Genetic and epigenetic fine mapping of causal autoimmune disease variants', Nature, 2015, 518, (7539), pp. 337-343
7. Rajagopal, N., Xie, W., Li, Y., Wagner, U., Wang, W., Stamatoyannopoulos, J., Ernst, J., Kellis, M., and Ren, B.: 'RFECS: a random-forest based algorithm for enhancer identification from chromatin state', PLoS Comput. Biol, 2013, 9, (3), pp. e1002968
8. Kleftogiannis, D., Kalnis, P., and Bajic, V.B.: 'DEEP: a general computational framework for predicting enhancers', Nucleic acids research, 2014, pp. gku1058
9. Fernández, M., and Miranda-Saavedra, D.: 'Genome-wide enhancer prediction from epigenetic signatures using genetic algorithm-optimized support vector machines', Nucleic acids research, 2012, 40, (10), pp. e77-e77
10. Erwin, G.D., Oksenberg, N., Truty, R.M., Kostka, D., Murphy, K.K., Ahituv, N., Pollard, K.S., and Capra, J.A.: 'Integrating diverse datasets improves developmental enhancer prediction', 2014
11. de Wit, E., and de Laat, W.: 'A decade of 3C technologies: insights into nuclear organization', Genes & development, 2012, 26, (1), pp. 11-24
12. Consortium, T.E.P.: 'The ENCODE (ENCyclopedia Of DNA Elements) Project', Science, 2004, 306, (5696), pp. 636-640
13. Pennacchio, L.A., Bickmore, W., Dean, A., Nobrega, M.A., and Bejerano, G.: 'Enhancers: five essential questions', Nature reviews. Genetics, 2013, 14, (4), pp. 288-295

14. Bernstein, B.E., Stamatoyannopoulos, J.A., Costello, J.F., Ren, B., Milosavljevic, A., Meissner, A., Kellis, M., Marra, M.A., Beaudet, A.L., Ecker, J.R., Farnham, P.J., Hirst, M., Lander, E.S., Mikkelsen, T.S., and Thomson, J.A.: 'The NIH Roadmap Epigenomics Mapping Consortium', Nature biotechnology, 2010, 28, (10), pp. 1045-1048
15. Barrett, T., Wilhite, S.E., Ledoux, P., Evangelista, C., Kim, I.F., Tomashevsky, M., Marshall, K.A., Phillippy, K.H., Sherman, P.M., and Holko, M.: 'NCBI GEO: archive for functional genomics data sets—update', Nucleic acids research, 2013, 41, (D1), pp. D991-D995
16. Won, K.-J., Ren, B., and Wang, W.: 'Genome-wide prediction of transcription factor binding sites using an integrated model', Genome biology, 2010, 11, (1), pp. R7-R7
17. Ren, B., Robert, F., Wyrick, J.J., Aparicio, O., Jennings, E.G., Simon, I., Zeitlinger, J., Schreiber, J., Hannett, N., Kanin, E., Volkert, T.L., Wilson, C.J., Bell, S.P., and Young, R.A.: 'Genome-Wide Location and Function of DNA Binding Proteins', Science, 2000, 290, (5500), pp. 2306-2309
18. Park, P.J.: 'ChIP–seq: advantages and challenges of a maturing technology', Nature Reviews Genetics, 2009, 10, (10), pp. 669-680
19. Pepke, S., Wold, B., and Mortazavi, A.: 'Computation for ChIP-seq and RNA-seq studies', Nature methods, 2009, 6, pp. S22-S32
20. Rajagopal, N., Xie, W., Li, Y., Wagner, U., Wang, W., Stamatoyannopoulos, J., Ernst, J., Kellis, M., and Ren, B.: 'RFECS: a random-forest based algorithm for enhancer identification from chromatin state', PLoS computational biology, 2013, 9, (3), pp. e1002968
21. Zhang, Y., Liu, T., Meyer, C.A., Eeckhoute, J., Johnson, D.S., Bernstein, B.E., Nusbaum, C., Myers, R.M., Brown, M., and Li, W.: 'Model-based analysis of ChIP-Seq (MACS)', Genome biology, 2008, 9, (9), pp. R137
22. Ghoshal, A., Grama, A., Bagchi, S., and Chaterji, S.: 'An Ensemble SVM Model for the Accurate Prediction of Non-Canonical MicroRNA Targets', in Editor (Ed.)^(Eds.): 'Book An Ensemble SVM Model for the Accurate Prediction of Non-Canonical MicroRNA Targets' (ACM, 2015, edn.), pp. pp. 403-412
23. Kunarso, G., Chia, N.-Y., Jeyakani, J., Hwang, C., Lu, X., Chan, Y.-S., Ng, H.-H., and Bourque, G.: 'Transposable elements have rewired the core regulatory network of human embryonic stem cells', Nature genetics, 2010, 42, (7), pp. 631-634
24. Arnold, C.D., Gerlach, D., Stelzer, C., Boryń, Ł.M., Rath, M., and Stark, A.: 'Genome-wide quantitative enhancer activity maps identified by STARR-seq', Science, 2013, 339, (6123), pp. 1074-1077
25. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., and Stoica, I.: 'Spark: cluster computing with working sets', in Editor (Ed.)^(Eds.): 'Book Spark: cluster computing with working sets' (2010, edn.), pp. 10
26. Kim, S., Ampornpunt, N., Grama, A., and Chaterji, S.: 'Interpretable Deep Neural Networks for Enhancer Prediction', in Editor (Ed.)^(Eds.): 'Book Interpretable Deep Neural Networks for Enhancer Prediction' (IEEE, 2015, edn.), pp. pp. 1-8
27. Pennacchio, L.A., Bickmore, W., Dean, A., Nobrega, M.A., and Bejerano, G.: 'Enhancers: five essential questions', Nat Rev Genet, 2013, 14, (4), pp. 288-295
28. Mohrs, M., Blankespoor, C.M., Wang, Z.-E., Loots, G.G., Afzal, V., Hadeiba, H., Shinkai, K., Rubin, E.M., and Locksley, R.M.: 'Deletion of a coordinate regulator of type 2 cytokine expression in mice', Nature immunology, 2001, 2, (9), pp. 842-847
29. Levine, M., Cattoglio, C., and Tjian, R.: 'Looping back to leap forward: transcription enters a new era', Cell, 2014, 157, (1), pp. 13-25
30. Krivega, I., and Dean, A.: 'Enhancer and promoter interactions—long distance calls', Current Opinion in Genetics & Development, 2012, 22, (2), pp. 79-85
31. Wang, D., Garcia-Bassets, I., Benner, C., Li, W., Su, X., Zhou, Y., Qiu, J., Liu, W., Kaikkonen, M.U., and Ohgi, K.A.: 'Reprogramming transcription by distinct classes of enhancers functionally defined by eRNA', Nature, 2011, 474, (7351), pp. 390-394
32. Yanez-Cuna, J.O., Arnold, C.D., Stampfel, G., Boryn, L.M., Gerlach, D., Rath, M., and Stark, A.: 'Dissection of thousands of cell type-specific enhancers identifies dinucleotide repeat motifs as general enhancer features', Genome research, 2014, pp. gr. 169243.169113
33. Rusk, N.: 'Genomics: Predicting enhancers by their sequence', Nature methods, 2014, 11, (6), pp. 606-607
34. Won, K.-J., Chepelev, I., Ren, B., and Wang, W.: 'Prediction of regulatory elements in mammalian genomes using chromatin signatures', BMC bioinformatics, 2008, 9, (1), pp. 547
35. Firpi, H.A., Ucar, D., and Tan, K.: 'Discover regulatory DNA elements using chromatin signatures and artificial neural network', Bioinformatics, 2010, 26, (13), pp. 1579-1586
36. Kleftogiannis, D., Kalnis, P., and Bajic, V.B.: 'DEEP: a general computational framework for predicting enhancers', Nucleic acids research, 2014