ParitoSVR: Parallel Iterated Optimizer for Support Vector Regression in the Primal

Kamalika Das* Kanishka Bhaduri† Nikunj Oza‡

Abstract

Regression problems on massive data sets are ubiquitous in many application domains including the Internet, earth and space sciences, and aviation. Support vector regression (SVR) is a popular technique for modeling the input-output relations of a set of variables under the added constraint of maximizing the margin, thereby leading to a very generalizable and regularized model. However, for a dataset with m training points, it is challenging to build SVR models due to the $O(m^3)$ cost involved in building them. In this paper we propose ParitoSVR — a parallel iterated optimizer for Support Vector Regression in the primal that can be deployed over a network of machines, where each machine iteratively solves a small (sub-)problem based only on the data observed locally and these solutions are then combined to form the solution to the global problem. Experiments on real datasets demonstrate the accuracy and scalability of our algorithm. As a real application, we use ParitoSVR to detect flights having abnormal fuel consumption from a fleet-wide commercial aviation database.

1 Introduction

In many application domains, it is important to predict the value of one feature based on certain other measured features. For example, in commercial aviation, it is very important to model the fuel consumption based on input parameters such as aircraft speed, wind speed, control surfaces, engine power, pitch, roll, yaw etc. This is because according to the Air Transportation Association (ATA), fuel is an airline's largest expense at a staggering 17.5 billion gallons per year[1]. Identifying flights with abnormal fuel consumption may help the airlines to do proper maintenance of these aircrafts and save operating costs. For such problems, a regression model can be learned that predicts the fuel flow based on these input parameters. One such popular regression method is Support vector machines (SVM) [1] which is a class of maximum margin classifiers, that demonstrates good generalization performance. SVM's can also exploit the kernel trick, thereby making them suitable for non-linear model learning as well. SVMs however are computationally expensive for large datasets.

In this paper we propose Parallel Iterated Optimizer for Support Vector Regression in the Primal (ParitoSVR), a new support vector regression algorithm that can be deployed over a network of machines, where each machine solves a small (sub-)problem based only on the data observed locally and these solutions are then combined to form the solution to the global problem. Our proposed method is based on the Alternating Direction Method of Multipliers (ADMM) optimization technique [2][3], which is parallelizable for separable convex problems, and converges to the exact solution as the centralized version with theoretical guarantees.

2 Background

Our ParitoSVR algorithm uses as a building block two components: (1) Alternating Direction Method of Multipliers (ADMM), and (2) SVR. In this section, we discuss these two topics.

ADMM: ADMM [3] is a decomposition algorithm for solving separable convex optimization problems of the form:

$$\begin{aligned} (2.1) \qquad & \min_{\mathbf{x},\mathbf{y}} \quad G_1(\mathbf{x}) + G_2(\mathbf{y}) \\ & \text{subject to} \quad A\mathbf{x} - \mathbf{y} = 0, \quad \mathbf{x} \in \mathbb{R}^n, \quad \mathbf{y} \in \mathbb{R}^m \end{aligned}$$

where $A \in \mathbb{R}^{m \times n}$ and G_1 and G_2 are convex functions. ADMM is an iterative technique and the update equations are:

$$\begin{aligned} \mathbf{x}^{t+1} &= \min_{\mathbf{x}} \left\{ G_1(\mathbf{x}) + \rho/2 \left\| A\mathbf{x} - \mathbf{y}^t + \mathbf{p}^t \right\|_2^2 \right\} \\ \mathbf{y}^{t+1} &= \min_{\mathbf{y}} \left\{ G_2(\mathbf{y}) + \rho/2 \left\| A\mathbf{x}^{t+1} - \mathbf{y} + \mathbf{p}^t \right\|_2^2 \right\} \\ \mathbf{p}^{t+1} &= \mathbf{p}^t + A\mathbf{x}^{t+1} - \mathbf{y}^{t+1} \end{aligned}$$

where $\mathbf{p} = (1/\rho)\mathbf{z}$. ADMM effectively decouples the $\mathbf{x}$ and $\mathbf{y}$ updates such that parallel execution becomes possible. In a distributed computing framework, this becomes even more interesting since each computing node can now solve a (smaller) subproblem in $\mathbf{x}$ inde-

*UARC, NASA Ames Research Center. kamalika.das@nasa.gov

†Netflix Inc. kanishka.bh@gmail.com

‡NASA Ames Research Center. nikunj.c.oza@nasa.gov

[1] http://www.airlines.org/Energy/Fuels101/Pages/AirlineEnergyQA.aspx

pendently, and then, these solutions can be efficiently gathered to compute the consensus variable $\mathbf{y}$ and the dual variable $\mathbf{p}$. ADMM converges within a few iterations when moderate precision is required. This can be particularly useful for many large scale problems, similar to what we consider here.

SVR: Give m data tuples (training set) $\mathcal{D} = (\mathbf{x}_i, y_i)_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^n$ is the input and $y_i \in \mathbb{R}$ is the corresponding output or target, SVR solves the following optimization problem:

$$\min_{\mathbf{w},b} \left[\lambda ||\mathbf{w}||^2 + \sum_{i=1}^{m} \ell_\varepsilon(\mathbf{w} \cdot \mathbf{x}_i + b - y_i) \right] \tag{2.2}$$

where λ is a constant and ℓ_ε is the *ε-insensitive loss function* defined as, $\ell_\varepsilon(r) = \max(|r| - \varepsilon, 0)$. This is a convex optimization problem which can be solved using convex optimization solvers such as CVX[2].

In the next section we show how to build SVR models for very large datasets using distributed computing via the ADMM technique.

3 ParitoSVR formulation

For the linear ParitoSVR algorithm setup, we assume that the training data is distributed among N client processors (nodes) $P_1, \ldots, P_N$ with a central machine P_0 acting as the server or collector. The dataset at machine P_j, denoted by D_j, consists of m_j data points *i.e.* $D_j = \left\{ \mathbf{x}_i^{(j)}, y_i^{(j)} \right\}_{i=1}^{m_j}$. It is assumed that the datasets are disjoint: $D_i \bigcap D_j = \emptyset$ and $\bigcup_{j=1}^{N} D_j = D$, where D is the total (global) data set. The goal is to learn a linear support vector regression model on D without exchanging all of the data among all the nodes.

Given Eqn. 2.2, the optimization problem is now:

$$\min_{\mathbf{w}} \left[\sum_{i=1}^{m} \ell_\varepsilon(\mathbf{w} \cdot \mathbf{x}_i - y_i) + \lambda ||\mathbf{w}||^2 \right]$$

$$\Leftrightarrow \min_{\mathbf{w}} \left[\sum_{j=1}^{N} \sum_{i=1}^{m_j} \ell_\varepsilon \left(\mathbf{w} \cdot \mathbf{x}_i^{(j)} - y_i^{(j)} \right) + \lambda \left\| \mathbf{w} \right\|^2 \right]$$

The inner sum can be computed by each node independently (assuming that $\mathbf{w}$ is known). We next write it in a form such that it is decoupled across the nodes:

$$\min_{\mathbf{w}_1, \ldots, \mathbf{w}_N, \mathbf{z}} \left[\sum_{j=1}^{N} \sum_{i=1}^{m_j} \ell_\varepsilon \left(\mathbf{w}_j \cdot \mathbf{x}_i^{(j)} - y_i^{(j)} \right) + \lambda \left\| \mathbf{z} \right\|^2 \right] \tag{3.3}$$

$$\text{subject to} \quad \mathbf{w}_j = \mathbf{z}$$

In the ADMM decomposition, each node can solve its local problem using its own data and optimization variable and then coordinate the results across the nodes to drive them into consensus. The nodes update the consensus variable $\mathbf{z}$ iteratively, based on their local data and scatter-gather operations on $\mathbf{z}$ until they converge to the same result.

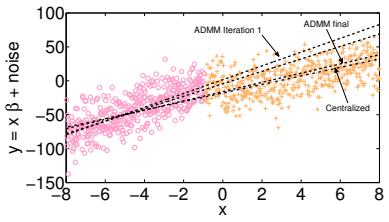


Figure 1: Models formed by node 1 on synthetic dataset as the algorithm progresses.

THEOREM 3.1. *The ADMM update rules for the linear support vector regression primal optimization are:*

$$\mathbf{w}_j^{t+1} = \min_{\mathbf{w}_j} \left\{ \sum_{i=1}^{m_j} \ell_\varepsilon \left(\mathbf{w}_j \cdot \mathbf{x}_i^{(j)} - y_i^{(j)} \right) + \frac{\rho}{2} \left\| \mathbf{w}_j - \mathbf{z}^t - \mathbf{u}_j^t \right\|_2^2 \right\}$$

$$\mathbf{z}^{t+1} = \min_{\mathbf{z}} \left\{ \lambda \left\| \mathbf{z} \right\|_2^2 + \frac{N\rho}{2} \left\| \mathbf{z} - \overline{\mathbf{w}}^{t+1} - \overline{\mathbf{u}}^t \right\|_2^2 \right\}$$

$$\mathbf{u}_j^{t+1} = \mathbf{u}_j^t + \mathbf{w}_j^{t+1} - \mathbf{z}^{t+1}$$

where $\mathbf{u} \in \mathbb{R}^n$ is the (scaled) dual variable and $\overline{\mathbf{w}}^{t+1}$ and $\overline{\mathbf{u}}^{t+1}$ are the averages of the variables over all the nodes.

Proof. We omit the proof here due to shortage of space.

The $\mathbf{w}$ update can be executed in parallel for each machine. It involves solving a convex optimization problem in $n+1$ variables at each node. This solution depends only on the data available at that partition. The $\mathbf{z}$ update step involves computing the average of the $\mathbf{w}$ and $\mathbf{u}$ vectors in order to combine the results from the different partitions. Critical to the working of ADMM is the convergence criteria. The primal and dual residuals can be written as: $r_p^t = \left\| \mathbf{w}^t - \mathbf{z}^t \right\|_2^2, r_d^t = \left\| \rho(\mathbf{z}^t - \mathbf{z}^{t-1}) \right\|$ Also, given the thresholds ϵ_{pri} and ϵ_{dual}, the primal and dual thresholds can be written as, $\epsilon_{pri} = \epsilon_{abs}\sqrt{m} + \epsilon_{rel} \max(\left\| \mathbf{w} \right\|, \left\| \mathbf{z} \right\|)$ and and $\epsilon_{dual} = \epsilon_{abs}\sqrt{m} + \rho\epsilon_{rel} \left\| \mathbf{u} \right\|$. The iterations terminate when $r_p^t < \epsilon_{pri}$ and $r_d^t < \epsilon_{dual}$.

4 Experiments

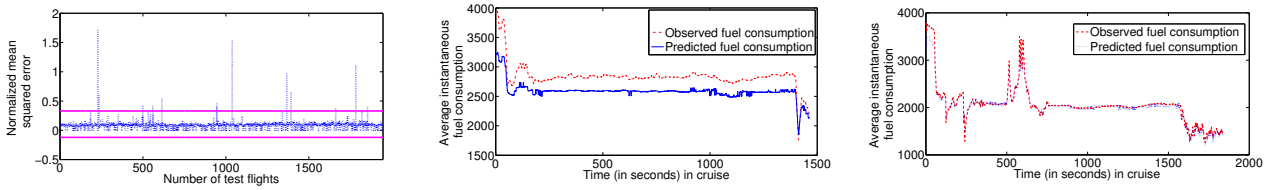
In this section we demonstrate the performance of the ParitoSVR algorithm.

ParitoSVR has been implemented in MATLAB 2011b. The experiments have been executed in NASA Pleiades supercomputer facility[3]. For solving the convex problems at each iteration, we have used the convex optimization toolbox CVX for Matlab[4].

[2] `http://cvxr.com/cvx/`

[3] `http://www.nas.nasa.gov/hecc/resources/pleiades.html`

[4] `http://cvxr.com/cvx/`



(a) Squared error for all flights (b) Outlier flight fuel consumption (c) Normal flight fuel consumption

Figure 2: Fuel flow study on CarrierX dataset. Fig. (a) shows squared error for all test flights, the 3-σ bound and flights which cross the threshold. Fig. (b) shows the observed and predicted fuel flow of top ranked anomalous flight. Fig. (c) shows the same for a normal flight.

Fig. 1 shows the sample dataset generated from a linear model following $y = \mathbf{w} \times \mathbf{x} + noise$, where $\mathbf{w}$ is the weight of the regression model. We have used 2 nodes in this experiment and, for each node, chosen a different $\mathbf{w}$ vector so that each node sees a different data distribution. The data of the two nodes are shown in two different colors (circle and plus markers). Also shown in the figure are the models (straight lines) formed by node 1 at different iterations of linear ParitoSVR algorithm.

4.1 Anomaly detection on CarrierX dataset We use the linear ParitoSVR algorithm to detect anomalous fuel consumption in a commercial aircraft. We model the average fuel flow as a function of 29 different parameters that measure system parameters such as lateral and longitudinal acceleration, roll and pitch angle, air pressure, and velocity, as well as external parameters such as wind speed and direction. We have used all 1500 flights (≈ 4.5 million training instances) for a specific tail number for a particular year for training, and tested subsequent years' flights for predicting fuel consumption. Flights for which the mean squared errors of the predicted instantaneous fuel consumption fall outside the 3-σ boundary of the average mean squared error, are tagged anomalous (σ is the standard deviation of the mean predictions). Out of approximately 1800 flights for a test year, 14 flights were determined to be anomalous. Figure 2(a) shows the mean squared errors for each of the flights in blue and the 3-σ bounds in green. The instantaneous fuel flow for the top ranked anomalous flight among these 14 flights is shown in Figure 2(b). The red graph depicting observed fuel flow is significantly higher than the predicted fuel consumption, shown in blue.

5 Conclusion

In this paper we have proposed ParitoSVR — a parallel iterated optimizer which solves support vector regression in the primal. Our formulation is parallelizable among a number of computing nodes connected to a central computing node. Empirical study show that our algorithm is accurate and scalable, ideal for large scale deployment. As future work, we plan to develop asynchronous version of this problem for peer-to-peer architectures.

Acknowledgements

The material is based upon work supported by the ARMD seedling fund from the National Aeronautics and Space Administration under Prime Contract Number NAS2-03144 awarded to the University of California, Santa Cruz, University Affiliated Research Center.

References

[1] V. V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., 1995.

[2] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Found. and Trends in Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.