Optimization Letters, , 1–7 (2010)

Privacy-Preserving Linear Programming

O. L. MANGASARIAN olvi@cs.wisc.edu
Computer Sciences Department
University of Wisconsin
Madison, WI 53706
Department of Mathematics
University of California at San Diego
La Jolla, CA 92093

Received March 16, 2010

Editor: Panos Pardalos

Abstract. We propose a privacy-preserving formulation of a linear program whose constraint matrix is partitioned into groups of columns where each group of columns and its corresponding cost coefficient vector are owned by a distinct entity. Each entity is unwilling to share or make public its column group or cost coefficient vector. By employing a random matrix transformation we construct a linear program based on the privately held data without revealing that data or making it public. The privacy-preserving transformed linear program has the same minimum value as the original linear program. Component groups of the solution of the transformed problem can be decoded and made public only by the original group that owns the corresponding columns of the constraint matrix and can be combined to give an exact solution vector of the original linear program.

Keywords: security, privacy-preserving, linear programming, vertically partitioned data

1. INTRODUCTION

Recently there has been substantial interest in privacy-preserving classification wherein the data to be classified is owned by different entities that are unwilling to reveal the data they hold or make it public. Various techniques were developed for generating classifiers without revealing the privately held data [11, 9, 10, 4, 1, 7, 5, 6]. Since underlying the classification problem is an optimization problem, often a linear program, we investigate here the problem of solving a general linear program where the m-by-n constraint matrix A is divided into p blocks of columns, each block of which together with the corresponding block of cost vector, is owned by a distinct entity not willing to make its data public. By using a linear transformation involving a different random matrix for each entity, as was done in [7] for the special case of classification problems, we are able to solve a privacy-preserving transformed linear program that generates an exact solution to the original linear program without revealing any of the privately held data.

A possible example illustrating this problem might be the classical diet problem of minimizing the cost of using n foods to generate m dietary supplements where the j-th food contains A_{ij} units of dietary supplement i. The privacy issue arises when the p column blocks of the m-by-n constraint matrix A, as well as the corresponding cost vector blocks, are owned by p entities unwilling to make their proprietary matrix column group and the corresponding cost vector public. The proposed approach here will allow us to solve this problem by a random linear transformation that will not reveal any of the privately held data but will give a publicly available exact minimum value to the original linear program. Component groups of the solution vector of the privacy-preserving transformed linear program can be decoded only by the group owners and can be made public by these entities to give an exact solution vector to the original linear program.

We briefly describe the contents of the paper. In Section 2 we give the theory and in Section 3 an implementation of our method for a privacy-preserving linear programming formulation using a random transformation. In Section 4 we give numerical examples demonstrating our approach. Section 5 concludes the paper with a summary and an open problem.

We describe our notation now. All vectors will be column vectors unless transposed to a row vector by a prime $'$. For a vector $x \in R^n$ the notation x_j will signify either the j-th component or j-th block of components. The scalar (inner) product of two vectors x and y in the n-dimensional real space R^n will be denoted by $x'y$. The notation $A \in R^{m\times n}$ will signify a real $m \times n$ matrix. For such a matrix, A' will denote the transpose of A, A_i will denote the i-th row or i-th block of rows of A and $A_{\cdot j}$ the j-th column or the j-th block of columns of A. A vector of zeros in a real space of arbitrary dimension will be denoted by 0.

2. Privacy-Preserving Linear Programming for Vertically Partitioned Data

We consider the linear program:

$$\min_{x\in X} c'x \text{ where } X = \{x \mid Ax \geq b\}, \tag{1}$$

and the matrix $A \in R^{m\times n}$ together with the cost vector $c \in R^n$, that is $\begin{bmatrix} c' \\ A \end{bmatrix}$, are divided into p vertical blocks of $n_1, n_2, \ldots\ldots$ and n_p, $(m+1)$-dimensional columns with $n_1 + n_2 + \ldots + n_p = n$. Each block of columns of A and corresponding block of the cost vector c' are "owned" by a distinct entity that is unwilling to make this block of data public or share it with the other entities. We wish to solve this linear program without revealing any privately held data. We shall achieve this by proceeding as follows.

Each of the p entities chooses its own privately held random matrix $B_{\cdot j} \in R^{k\times n_j}$, $j = 1,\ldots,p$, where $k \geq n$. Define:

$$B = [B_{\cdot 1}\;\; B_{\cdot 2} \ldots\ldots B_{\cdot p}] \in R^{k\times n}. \tag{2}$$

We note immediately that the rank of the randomly generated matrix $B \in R^{k\times n}$ with $k \geq n$ is n [2], which is the reason for choosing $k \geq n$. Utilizing this fact we define the following invertible transformation:

$$x = B'u, \tag{3}$$

and its least 2-norm inverse:

$$u = B(B'B)^{-1}x. \tag{4}$$

We now transform our original linear program into the following "secure" linear program:

$$\min_{u\in U} c'B'u \text{ where } U = \{u \mid AB'u \geq b\}. \tag{5}$$

We use the term "secure" to describe the transformed linear program (5) because it does not reveal any of the privately held data $\begin{bmatrix} c'_j \\ A_{\cdot j} \end{bmatrix}$, $j = 1, \ldots, p$. This is so because for each entity different from entity j, it is impossible to compute either c_j from the revealed product $c_j{}'B_{\cdot j}{}'$, or $A_{\cdot j}$ from the revealed product $A_{\cdot j}B_{\cdot j}{}'$ *without* knowing the random matrix $B_{\cdot j}$ chosen by entity j and known to itself only. See also Remark 2 below.

We now relate our original linear program (1) to the transformed linear program (5) as follows.

Proposition 1 *Let $k \geq n$ for the random matrix $B \in R^{k \times n}$ of (2). The secure linear program (5) is solvable if and only if the linear program (1) is solvable in which case the extrema of both linear programs are equal.*

Proof: We begin by stating the duals of the two linear programs (1) and (5) respectively as follows:

$$\max_{v \in V} b'v \text{ where } V = \{v \mid A'v = c,\ v \geq 0\}, \tag{6}$$

$$\max_{w \in W} b'w \text{ where } W = \{w \mid BA'w = Bc,\ w \geq 0\}. \tag{7}$$

Let $\bar{x}$ and $\bar{v}$ solve the dual linear programs (1) and (6) respectively. Define $\bar{u}$ as in (3)-(4), that is $\bar{x} = B'\bar{u}$ and $\bar{u} = B(B'B)^{-1}\bar{x}$. Thus $\bar{u}$ satisfies the constraints of (5). Since $\bar{v}$ solves the dual problem (6) it follows that:

$$A'\bar{v} = c,\ \ \bar{v} \geq 0. \tag{8}$$

Consequently,

$$BA'\bar{v} = Bc,\ \ \bar{v} \geq 0. \tag{9}$$

Hence $\bar{v} \in W$, the dual feasible region of (7). Consequently the dual pair (5)-(7) are both feasible and hence by strong duality theory are both solvable with equal extrema. Consequently:

$$c'B'\bar{u} = c'\bar{x} = \min_{x \in X} c'x = \max_{v \in V} b'v = b'\bar{v} \leq \max_{w \in W} b'w = \min_{u \in U} c'B'u, \tag{10}$$

where the inequality above follows from the fact just established that $\bar{v} \in W$. Hence $\bar{u} = B(B'B)^{-1}\bar{x}$ solves (5).

Conversely now, let $\bar{u}$ and $\bar{w}$ solve the dual pair (5)-(7). Let $\bar{x} = B'\bar{u}$ and hence $\bar{x} \in X$. Since $\bar{w}$ solves the dual problem (7), it follows that

$$BA'\bar{w} = Bc,\ \bar{w} \geq 0. \tag{11}$$

Since the rank of the matrix B is n, it follows that

$$A'\bar{w} = c,\ \bar{w} \geq 0. \tag{12}$$

Hence $\bar{w} \in V$, that is it is feasible for the dual problem (6). Since $\bar{x} = B'\bar{u} \in X$ and $\bar{w} \in V$, it follows that the dual pair (1)-(6) are solvable.

We have thus established that the linear program (1) is solvable if and only if the secure linear program (5) is solvable. It remains to show that the extrema of these two linear programs are equal.

Since $\bar{w} \in W$ implies that $\bar{w} \in V$, it follows that

$$\max_{w\in W} b'w = b'\bar{w} \leq \max_{v\in V} b'v. \tag{13}$$

Hence:

$$\min_{u\in U} c'B'u = \max_{w\in W} b'w \geq \min_{x\in X} c'x = \max_{v\in V} b'v \geq \max_{w\in W} b'w = \min_{u\in U} c'B'u, \tag{14}$$

where the equalities above follow from the equality of optimal primal and dual objectives of linear programs, the first inequality follows from (10) and the second inequality from (13). Thus, $\min_{x\in X} c'x = \min_{u\in U} c'B'u$, and the extrema of (1) and (5) are equal. ∎

We turn now to an explicit implementation of the secure linear programming formulation (5).

3. Privacy-Preserving Linear Program (PPLP)

Starting with the linear program (1) that is partitioned among p entities as described in Section 2, we propose the following algorithm for generating a solution to the linear program without disclosing any of the privately held data.

Algorithm 1 PPLP Algorithm

(I) All p entities agree on a $k \geq n$, the number of rows of the random matrix $B \in R^{k\times n}$ as defined in (2).

(II) Each entity generates its own privately held random matrix $B_{\cdot j} \in R^{k\times n_j}$, $j = 1,\ldots\ldots,p$, where n_j is the number of features held by entity j which results in:

$$B = [B_{\cdot 1}\ \ B_{\cdot 2}\ldots\ldots B_{\cdot p}] \in R^{k\times n}. \tag{15}$$

(III) Each entity j makes public only its matrix product $A_{\cdot j}B_{\cdot j}{}'$ as well as its cost coefficient product $B_{\cdot j}c_j$. These products do not reveal either $A_{\cdot j}$ or c_j but allow the public computation of the full constraint matrix needed for the secure linear program (5):

$$AB' = A_{\cdot 1}B_{\cdot 1}{}' + A_{\cdot 2}B_{\cdot 2}{}' + \ldots\ldots + A_{\cdot p}B_{\cdot p}{}', \tag{16}$$

as well as the cost coefficient for (5):

$$c'B' = c_1'B_{\cdot 1}{}' + c_2'B_{\cdot 2}{}' + \ldots\ldots + c_p'B_{\cdot p}{}'. \tag{17}$$

(IV) A public optimal solution vector u to the secure linear program (5) and a public optimal objective function value $c'B'u$ are computed. By Proposition 1 this optimal value equals the optimal objective function of the original linear program (1).

(V) Each entity computes its optimal x_j component group from (3) as follows:

$$x_j = B_{\cdot j}{}'u, \quad j = 1,\ldots,p. \tag{18}$$

(VI) *The solution component vectors* x_j, $j = 1,\dots,p$, *are revealed by its owners if a public solution vector to the original linear program is agreed upon. Else the component vectors may be kept private if only the minimum value of (1) is needed, in which case that minimum value equals the the publicly available minimum value* $\min_{u\in U} c'B'u$ *of the secure linear program (5).*

Remark 2 *Note that in the above algorithm no entity* j *reveals its data matrix* $A_{\cdot j}$ *or its cost group vector* c_j. *Components of the solution vector* x_j *are revealed if agreed upon. Note further that it is impossible to compute the* mn_j *numbers constituting* $A_{\cdot j} \in R^{m\times n_j}$ *given the* mk *numbers constituting* $(A_{\cdot j}B_{\cdot j}{}') \in R^{m\times k}$ *and not knowing* $B_{\cdot j} \in R^{k\times n_j}$. *Similarly it is impossible to compute* $c_j \in R^{n_j}$ *from* $c_j{}'B_{\cdot j}{}' \in R^k$, *or* $x_j \in R^{n_j}$ *from* $B_{\cdot j}{}'u \in R^{n_j}$ *without knowing* $B_{\cdot j}$. *Hence, all entities share the publicly computed optimal value of their common linear program but each entity shares or keeps private their optimal components values unless agreed upon to make them public.*

We turn now to some computational results.

4. Computational Results

We demonstrate our results by solving two examples as follows on a 4 Gigabyte machine running $i386\_rhe15$ Linux. We utilize the CPLEX linear programming code [3] within MATLAB [8] to solve our linear programs. The first example has 100 constraints and 1000 variables, while the second example has 100 variables and 1000 constraints. For both examples, the components of the matrix A were uniformly distributed in the interval $[-50,50]$, and the components of the primal solution $\bar{x}$ of (1) were uniformly distributed in the interval $[-5,5]$, with about half of the primal constraints of (1) being active with corresponding dual variables uniformly distributed in the interval $[0,10]$. We used $k = n$ in both examples. Similar results were obtained for $k > n$.

Example 1 *For our first example we generated a random solvable linear program (1) with* $m = 100$ *and* $n = 1000$. *We partitioned the columns of* A *as well as the cost vector* c *into three groups with* $n_1 = 500$, $n_2 = 300$ *and* $n_3 = 200$. *We generated three random matrices, with coefficients uniformly distributed in the interval [0,1] with* $B_{\cdot 1} \in R^{n\times n_1}$, $B_{\cdot 2} \in R^{n\times n_2}$ *and* $B_{\cdot 3} \in R^{n\times n_3}$. *We solved the secure linear program (5) and compared its optimal objective value with that of (1). The two optimal objectives agreed to 14 significant figures attained at two distinct optimal solution points. Computation time was 0.163 seconds.*

Example 2 *For our second example we generated a random solvable linear program (1) with* $m = 1000$ *and* $n = 100$. *We partitioned the columns of* A *as well as the cost vector* c *into three groups with* $n_1 = 50$, $n_2 = 30$ *and* $n_3 = 20$. *We generated three random matrices, with coefficients uniformly distributed in the interval [0,1] with* $B_{\cdot 1} \in R^{n\times n_1}$, $B_{\cdot 2} \in R^{n\times n_2}$ *and* $B_{\cdot 3} \in R^{n\times n_3}$. *We solved the secure linear program (5) and compared its optimal objective value with that of (1). The two optimal objectives agreed to 13 significant figures attained at points that were essentially the same, that is the* ∞*-norm of their difference was less than* $2.3e-13$. *Computation time was 0.177 seconds.*

We note that the solution vector values for the linear programs (1) and (5) in the above two examples may or may not be the same because either problem may not have a unique solution. Thus attaining the same optimal value was our main comparative measure.

5. Conclusion and Outlook

We have shown how to securely solve a linear program when its constraint matrix and objective function data are partitioned among entities unwilling to share their data or make it public. Another interesting problem in this realm occurs when the constraint matrix rows and the corresponding right hand side data are partitioned among entities similarly unwilling to make their data public. We refer to the first problem, already addressed in the present paper, as a vertically partitioned linear program and the latter problem as horizontally partitioned linear program. Just as in the case for horizontally partitioned classification problems [5], all entities use the *same* random matrix $B \in R^{k\times n}$ in the secure linear program (5) to compute privately their matrix product A_iB', $i = 1,\ldots,p$. These matrix products are combined to generate a public AB' as follows:

$$AB' = \begin{bmatrix} A_1B' \\ \vdots \\ A_pB' \end{bmatrix} \tag{19}$$

Because B is publicly known, unlike the case treated in this paper, k must be less than n again as in [5] in order to prevent the computation of A_i from the publicly available A_iB'. This latter condition that $k < n$ prevents the secure linear program (5) from giving an exact solution to our original linear program (1). However, this issue does not hinder the generation of an accurate privacy-preserving classifier in [5], as measured by testing set correctness, but does prevent here a secure linear program (5) from generating an accurate solution to the original linear program (1). Circumventing this issue for a general linear program is an interesting problem for possible future research that we do not have an immediate answer for now.

Acknowledgments The research described in this Data Mining Institute Report 10-01, March 2010, was supported by the Microsoft Corporation and ExxonMobil. I am indebted to Alice Bednarz of the University of Adelaide, Australia, who pointed out a difficulty associated with possibly including a nonnegativity constraint in the linear program (1).

References

1. K. Chen and L. Liu. Privacy preserving data classification with rotation perturbation. In *Proceedings of the Fifth International Conference of Data Mining (ICDM'05)*, pages 589–592. IEEE, 2005.
2. X. Feng and Z. Zhang. The rank of a random matrix. *Applied Mathematics and Computation*, 185:689–694, 2007.
3. ILOG, Incline Village, Nevada. *ILOG CPLEX 9.0 User's Manual*, 2003. http://www.ilog.com/products/cplex/.
4. Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. Cryptographically private support vector machines. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 618–624, New York, NY, USA, 2006. ACM.

5. O. L. Mangasarian and E. W. Wild. Privacy-preserving classification of horizontally partitioned data via random kernels. Technical Report 07-03, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, November 2007. Proceedings of the 2008 International Conference on Data Mining, DMIN08, Las Vegas July 2008, Volume II, 473-479, R. Stahlbock, S.V. Crone and S. Lessman, Editors.
6. O. L. Mangasarian and E. W. Wild. Privacy-preserving random kernel classification of checkerboard partitioned data. Technical Report 08-02, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, September 2008. Annals of Information Systems XIII, 2010, 375-387.
7. O. L. Mangasarian, E. W. Wild, and G. M. Fung. Privacy-preserving classification of vertically partitioned data via random kernels. Technical Report 07-02, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, September 2007. ACM Transactions on Knowledge Discovery from Data (TKDD) Volume 2, Issue 3, October 2008.
8. MATLAB. *User's Guide*. The MathWorks, Inc., Natick, MA 01760, 1994-2006. http://www.mathworks.com.
9. M.-J. Xiao, L.-S. Huang, H. Shen, and Y.-L. Luo. Privacy preserving id3 algorithm over horizontally partitioned data. In *Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05)*, pages 239–243. IEEE Computer Society, 2005.
10. H. Yu, X. Jiang, and J. Vaidya. Privacy-preserving SVM using nonlinear kernels on horizontally partitioned data. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 603–610, New York, NY, USA, 2006. ACM Press.
11. H. Yu, J. Vaidya, and X. Jiang. Privacy-preserving svm classification on vertically partitioned data. In *Proceedings of PAKDD '06*, volume 3918 of *LNCS: Lecture Notes in Computer Science*, pages 647 – 656. Springer-Verlag, January 2006.