Sparseness and a Reduction from Totally Nonnegative Least Squares to SVM

Vamsi K. Potluru and Sergey M. Plis and Shuang Luan and Vince D. Calhoun and Thomas P. Hayes

***Abstract*— Nonnegative Least Squares (NNLS) is a general form for many important problems. We consider a special case of NNLS where the input is nonnegative. It is called Totally Nonnegative Least Squares (TNNLS) in the literature. We show a reduction of TNNLS to a single class Support Vector Machine (SVM), thus relating the sparsity of a TNNLS solution to the sparsity of supports in a SVM. This allows us to apply any SVM solver to the TNNLS problem. We get an order of magnitude improvement in running time by first obtaining a smaller version of our original problem with the same solution using a fast approximate SVM solver. Second, we use an exact NNLS solver to obtain the solution. We present experimental evidence that this approach improves the performance of state-of-the-art NNLS solvers by applying it to both randomly generated problems as well as to real datasets, calculating radiation therapy dosages for cancer patients.**

I. Introduction

Quite a few problems in machine learning and signal processing can be cast as Nonnegative Least Squares (NNLS) problems. The NNLS problem is fairly old and the algorithm by Lawson and Hanson [1974] appears to be one of the first to solve it. NNLS problems frequently come up in practice and there are quite a few algorithms to solve them Lawson and Hanson [1974], Kim et al. [2006], Bro and De Jong [1997], Van Benthem and Keenan [2004]. Note that the nonnegativity constraint is pretty natural in real problems, for instance, when we are modeling chemical concentrations, brain activations, and color intensities. Real world applications include target detection at subpixel level in remote sensing images [Chang and Heinz, 2000], and resolving tags into genes in the SAGE datasets [Zeng and Ogihara, 2009].

The special case "Totally Nonnegative Least Squares" (TNNLS) in which the input is all nonnegative was introduced by Merritt and Zhang [2005]. TNNLS has been applied to compressive sensing by O'Grady and Rickard [2008] who showed that nonnegativity is enough to recover a sufficiently sparse signal. Bruckstein et al. [2008] have explored the connection between uniqueness of nonnegative sparse solutions of underdetermined systems of equations while Donoho and Tanner [2006] explored thresholds for recovery of sparse solutions via l_1 minimization. But why is there this connection between nonnegative entries and sparsity?

We shed light on this question, by presenting a general reduction from TNNLS to a single class SVM problem. This shows that nonnegativity constraint gives us a sparse solution depending on the number of supports required to define a maximum-margin hyperplane of the suitably defined single class SVM problem. Thus, the sparsity of the solutions to TNNLS is directly tied to the size of the support of a related SVM problem.

A brief history of connections between SVM's and regression will place this work in perspective. Hochreiter and Mozer [2001] established an isomorphism between sparse separation and ϵ-SVM regression and used it to kernelize sparse separation. Similarly, a connection between Lasso and SVM's was established by Li et al. [2006] and further exploited for the kernel version of Lasso. Furthermore, the kernel adatron (KA) algorithm for solving SVM [Frießet al., 1998] resurfaced in the NNLS algorithm from Franc et al. [2005]. This allows us to predict, for instance, that the sequential minimal optimization (SMO) algorithm developed for SVM [Platt, 1998] will find application in solving NNLS problems with a sum constraint, for example the problem in Chang and Heinz [2000]. We note that Lasso, SVM, NNLS are all special cases of Nonnegative Quadratic Programming (NQP). So, it is not surprising that algorithms developed for one problem can be applied to another. However, it would be nice if we could do this automatically and with minimal effort.

In this paper, we show that for TNNLS, we can reduce it to a special case of SVM. So, quite a few solvers developed for SVM can be mechanically applied to solve TNNLS. In particular, we reduce TNNLS to the dual formulation of the single class SVM problem.

Moreover, our reduction can be used to solve TNNLS more efficiently. First, apply our reduction, and use an SVM solver to approximately solve the problem which enables us to find a large subset of the zeros of the solution vector. Then solve the smaller problem (with the zeros eliminated) by using an exact NNLS solver. We will describe some experiments comparing the speed of this approach with direct application of existing NNLS solvers.

II. Preliminaries

We give an introduction to the TNNLS and SVM problems.

A. Totally Nonnegative Least Squares

Let $\boldsymbol{W} \in R^{m \times n}$ be a matrix and $\boldsymbol{x} \in R^m$ a column vector. The nonnegative least squares problem (NNLS) is to find a column vector $\boldsymbol{h} \in R^n$ minimizing the following objective

Vamsi K. Potluru(email: ismav@cs.unm.edu), Shuang Luan(email: sluan@cs.unm.edu) and Thomas P. Hayes(email: hayes@cs.unm.edu) are with Department of Computer Science, University of New Mexico, USA. Vince D. Calhoun(email: vcalhoun@mrn.org) and Sergey M. Plis(email: pliz@cs.unm.edu) are with Mind Research Network, New Mexico, USA.

function:

$$G(\boldsymbol{h}) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{W}\boldsymbol{h}\|_2^2 \qquad \text{s.t. } \boldsymbol{h} \geq \boldsymbol{0} \tag{1}$$

If we additionally constrain all the elements of matrix $\boldsymbol{W}$ to be nonnegative and vector $\boldsymbol{x}$ to be positive, we get a special case of NNLS, which is referred to in the literature as totally nonnegative least squares or TNNLS.

Actually, we solve a slightly more general version of the problem. The only constraint we have on $\boldsymbol{W}$ and $\boldsymbol{x}$ is that $\boldsymbol{W}^T\boldsymbol{x}$ is nonnegative. This includes the case of TNNLS. However, we will focus on TNNLS for this paper.

Many algorithms have been developed over the years to solve the NNLS problem. A brief history of these can be found in Kim et al. [2006]. For instance, the NNLS algorithm of Lawson and Hanson [1974] was modified by Bro and De Jong [1997] and was called FAST-NNLS(FNNLS). However, FNNLS requires the computation of matrix-matrix product (of the input matrix) and can be expensive for large-scale problems. This was ameliorated in the case of multiple right hand sides by Benthem and Keenan Van Benthem and Keenan [2004] and was called FCNNLS. We do not consider FCNNLS in this paper for we are solving NNLS problems with a single right hand side. Recently, advances in fast randomized projections have lead to the development of a randomized algorithm for NNLS which involves first employing a randomized Hadamard transform to construct a smaller NNLS problem. This is then solved by a standard NNLS solver. This is the main idea for the NNLS solver in Boutsidis and Drineas [2009] .

B. Support Vector Machines

Support vector machines (SVM) are now routinely used for many classification problems in machine learning as seen in Schölkopf and Smola [2001], Kecman [2001], Wang [2005] due to their ease of use and ability to generalize. In the basic case, the input data, corresponding to two groups, is mapped into a higher dimensional space, where a maximum-margin hyperplane is computed to separate them. The "kernel trick" is used to ensure that the mapping into higher dimensional space is never explicitly calculated. This can be formulated as a non-negative quadratic programming (NQP) problem and there are efficient algorithms to solve it, for instance Platt [1998].

Recently, the cost of training of kernel SVM's has shifted the focus of the SVM community back to linear SVM for large scale applications. This has lead to the formulation of very efficient linear SVM solvers which converge to a ϵ precision solution in linear(in the number of training points) time as seen in the papers by Franc and Sonnenburg [2008], Hsieh et al. [2008].

C. Linear SVMs

Given labeled training examples $(\boldsymbol{s}_i, y_i)_{i=1}^n$ where $y_i = \pm 1$ and a regularization constant $C > 0$, SVMs learn a linear classification rule. The primal formulation of the binary class linear SVM is:

$$\min_{\boldsymbol{w}} P(\boldsymbol{w}) := \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n} \max\{0, 1 - y_i(\langle \boldsymbol{w}, \boldsymbol{s}_i\rangle)\} \tag{2}$$

The dual formulation of the single class linear support vector machine (SVM) Schölkopf and Smola [2001] is to minimize the following objective function:

$$F(\boldsymbol{v}) = \frac{1}{2}\boldsymbol{v}^T\boldsymbol{A}\boldsymbol{v} - \boldsymbol{1}^T\boldsymbol{v} \qquad \text{s.t. } \boldsymbol{0} \leq \boldsymbol{v} \leq C\boldsymbol{1} \tag{3}$$

where $\boldsymbol{A}$ is the Gram matrix of data points$(\{\boldsymbol{s}_i\}_{i=1}^n)$ given by $A_{ij} = y_i y_j \boldsymbol{s}_i^T \boldsymbol{s}_j$. Our formulation assumes that the maximum-margin hyperplane passes through the origin. Note that, if we set C to ∞ we get the hard-margin maximum margin formulation for linear SVM.

III. Our Reduction from TNNLS to SVM

Next, we describe a general framework for reducing TNNLS to SVM. In particular, we show that the TNNLS problem can be reduced to solving a hard-margin single class dual SVM problem. We are now in a position to sketch the reduction from TNNLS to SVM. Let $\boldsymbol{D}$ denote the diagonal matrix whose diagonal elements are given by the vector $\frac{1}{\boldsymbol{W}^T\boldsymbol{x}}$. Also, let $\boldsymbol{h} = \boldsymbol{D}\boldsymbol{z}$. Then,

$$\begin{aligned} G(\boldsymbol{z}) &= \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{W}\boldsymbol{D}\boldsymbol{z}\|_2^2 \\ &= \frac{1}{2}\boldsymbol{z}^T(\boldsymbol{W}\boldsymbol{D})^T(\boldsymbol{W}\boldsymbol{D})\boldsymbol{z} - \boldsymbol{x}^T\boldsymbol{W}\boldsymbol{D}\boldsymbol{z} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{x} \\ &= \frac{1}{2}\boldsymbol{z}^T(\boldsymbol{W}\boldsymbol{D})^T(\boldsymbol{W}\boldsymbol{D})\boldsymbol{z} - \boldsymbol{1}^T\boldsymbol{z} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{x} \end{aligned}$$

Ignoring the $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{x}$, which does not change the location of the minimum, we see that it is an instance of the SVM objective in equation 3, with $\boldsymbol{v}$ corresponding to $\boldsymbol{z}$ and $\boldsymbol{s}_i$ corresponding to $\boldsymbol{W}_i\boldsymbol{D}_{ii}$. We have a single class maximum margin classifier passing through origin where the datapoints given by $\{\boldsymbol{W}_i D_{ii}\}_i^n$ lie in the positive orthant. Since, the primal version of TNNLS corresponds to the dual of a single class SVM, the dual of TNNLS corresponds to the primal of the single class SVM. Geometrically, this corresponds to finding a maximum margin hyperplane which gives us the set of supports. Or in other words, it gives us the zero elements of the vector $\boldsymbol{z}$ or equivalently $\boldsymbol{h}$. In practice, we might find a superset of supports because we use a SVM solver to find an approximate maximum margin hyperplane. These issues are discussed in further detail in the next section.

A. Implementation issues

Algorithms for solving the SVM problem can be split into primal or dual depending on the version of the problem they solve. In this paper, we use a primal SVM solver to find an approximate maximum-margin hyperplane. This gives us a subset of nonsupport vectors. We solve for the remaining entries by invoking an exact NNLS solver. Note that, if we had a dual SVM solver, we could directly use it to solve the TNNLS problem. However, since we are using solvers whose

performance scales as $O(\log(1/\epsilon))$, it might be preferable to get an approximate solution for say $\epsilon = 0.001$ and get the exact solution by using some other exact solver. This depends on how much we care about the accuracy of the solution and is application dependent.

Recently, a lot of fast approximate solvers have been proposed to solve the linear SVM problem. OCAS by Franc and Sonnenburg [2008] is based on a cutting plane algorithm and is one of the state-of-the art solvers. It very quickly approximates a maximum-margin hyperplane and its running time is linear in the size of the input samples (see Franc and Sonnenburg [2008] or Hsieh et al. [2008] for details). For getting an approximate maximum-margin hyperplane, we use the OCAS solver of Franc and Sonnenburg [2008]. However, since we are using an approximate SVM solver, we find a subset of the zeros and have to solve for the smaller problem by using an exact NNLS solver.

B. Approximate solver

If we use a primal hard-margin SVM solver and find an approximate hyperplane, say $\boldsymbol{w}$ then $\boldsymbol{w}$ satisfies the condition: $1 - P(\boldsymbol{w}^*)/P(\boldsymbol{w}) \leq \epsilon$ where $\boldsymbol{w}^*$ denotes the optimal hyperplane and ϵ is tolerance to which we solve the problem. The way we set a coefficient to zero is if it's corresponding input s_i satisfies the condition $\langle \boldsymbol{w}^*, \boldsymbol{s}_i \rangle > 1$. Since we don't have the vector $\boldsymbol{w}^*$ and have access to only $\boldsymbol{w}, \epsilon$, we instead use the following test function: $\langle \boldsymbol{w}, \boldsymbol{s}_i \rangle > 1+\delta$ where δ is a function of ϵ and the data. By using the primal SVM solver, we are in fact solving the dual version of the TNNLS problem. Informally, this corresponds to finding the zeros of the solution vector. Once, we have all the zeros of the solution vector, the rest can be found by any least squares solver. However, if we end up with a subset of the zeros, as we do in this paper, we need to solve for the rest of the solution vector by using an NNLS solver. In practice, we find that ϵ set in the range $[10^{-4}, 10^{-6}]$ is a good compromise between speed and accuracy as seen in the experiment section IV. We don't actually give a formula for computing δ but found 10ϵ to be a good heuristic in practice.

In this paper, we use a soft-margin SVM solver. This results in the issue of selecting the soft-margin parameter C. If we had used a hard margin SVM solver, this would not an issue as $C = \infty$. However, in the case of soft-margin SVM we need to set it. Ideally, we want C to be as large as possible. We found that C in the range $[10, 100]$ is a good choice for a wide range of problem sizes as shown in experiment section IV.

We illustrate the issue of setting the parameters of C, ϵ and δ in Figure 1. If the data is not close to the maximum separating hyperplane then we need not solve the SVM to high accuracy and can get by with a rough solution. However, if the data is highly clustered as in the case of Figure 1 then, we need to set ϵ high and this requires higher computation time for the SVM solver. It might be that the reduction can take more time than if we solved it directly. At the moment, we don't have a nice way to resolve this question.

If we don't treat the SVM solver as a black box as we do now, we can do something smarter by checking "progress" at each iteration and can come up with heuristics as to when to switch to exact solver.

C. Bounds

The soft margin parameter C can be set in a precise manner if we solve the primal SVM problem exactly. Notice that the soft-margin SVM formulation (2) has a dual formulation (3) where the parameter C only appears as an upper-bound constraint. Let $L = \max(\frac{\boldsymbol{W}^T\boldsymbol{x}}{diag(\boldsymbol{W}^T\boldsymbol{W})})$ where the function *diag* outputs the diagonal of a given input matrix. Setting C to be any value greater than L would make the single-class soft-margin SVM problem with nonnegative inputs equivalent to its hard-margin formulation. However, since we solve the soft-margin to only ϵ precision, it becomes tricky as to what the optimal value for parameter C should be.

We have the following bound on the primal solver objective:

$$(1 - \epsilon)P(\boldsymbol{w}) \leq P(\boldsymbol{w^*}) \leq P(\boldsymbol{w})$$

We can find a nonnegative vector $\boldsymbol{f}$ from the above inequality such that if $\langle \boldsymbol{w} - \boldsymbol{f}, \boldsymbol{s}_i \rangle > 1$ implies that $\langle \boldsymbol{w}^*, \boldsymbol{s}_i \rangle > 1$ for all inputs s_i. In practice, we found this approach for estimating the supports conservative.

IV. Experiments

In this section, we are going to present the results of applying various NNLS algorithms to different datasets. For the PQN-NNLS solver, we use the code supplied by the authors of Kim et al. [2006]. The default settings for the solver were used. The randomized NNLS solver code is based on Avron et al. [2010] and Kim et al. [2006].

We ran all the experiments on a machine with 2.2Ghz cpu power and 32GB of physical memory with 8 cores. The number of threads was set to 1 to ensure that we are using a single core.

Besides random data sets, we have also applied our TNNLS solver to data sets that arise from Gamma Knife radiosurgery [Chin and Regine, 2008] and particle radiation therapy [De Laney and Kooy, 2007].

Gamma knife radiosurgergy has been a well-known treatment modality for many brain tumors and functional disorders. It uese γ-rays emitted from radioactive ^{60}Co sources to eradicate tumors and eliminate them. These sources are placed in a hemispherical, circular or linear array and their γ-ray beams are focused on a single point, creating a spherical high dose volume. Generally speaking, the goal of Gamma Knife is to use these spherical high dose volume to create a radiation dose distribution where the high dose regions are conformed to the targeted tumors. The problem is a typical TNNLS problem, where each column of the matrix W is a spherical high dose volume, and vector x is the ideal dose distribution, and the vector h is the weighting (i.e., "beam-on" time) of each high dose volume. Natually, everything is nonnegative in the problem. Our experimental results on Gamma Knife radiosurgery are in Section IV-B.

Another type of medical problem that we have experimented is the particle radiation therapy, where charged particles such

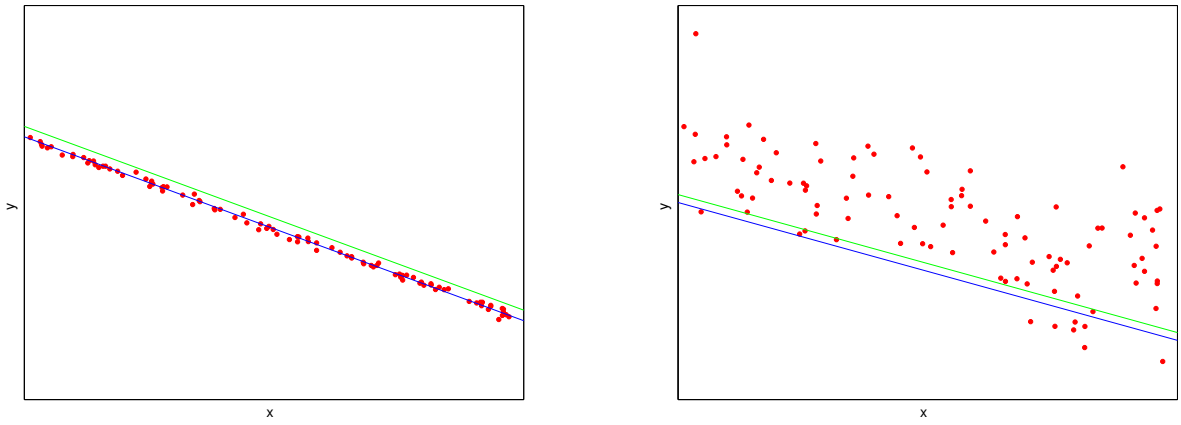

Fig. 1. We illustrate two cases where SVM gives us no speed up(left) and the other case where it does(right). The approximate hyperplane which is output by the SVM solver is shown in blue and the threshold hyperplane whether we accept a point as a non-support is given by the line in green. Parameters C, ϵ have to be fed in to the SVM solver. A priori, we have to trade-off high values of C and low values of ϵ with computation time. The best values of these parameters with respect to computation time and correctness of solution depend on the distribution of the data. This is illustrated in the figure. We plot two dimensional points on a plane in each figure. If the data is too clustered along the maximum-margin hyperplane(left) and the values of C, ϵ and δ are not set properly, we might end up declaring that all points are potentially supports after running the SVM solver. A better distribution of data would result in the case(right) where we prune the number of potential supports thereby reducing the size of the original problem.

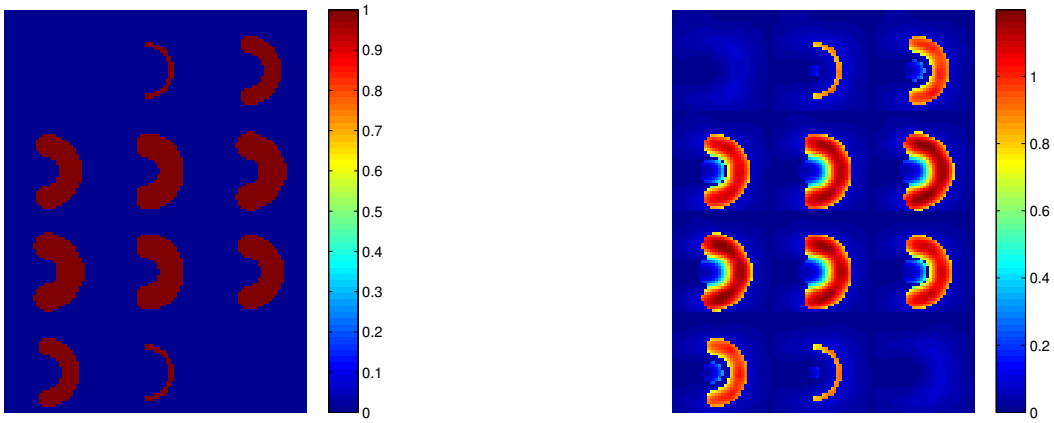

Fig. 2. C-shaped tumor in the Phantom tumor dataset. We plot 12 of the 35 slices. (Left) After we apply TNNLS solvers, the dosage we end up with for each of the corresponding target slices is shown(Right)

as protons and carbon ions are used to irradiate tumors. This problem is similar to Gamma Knife radiosurgery, because the goal is to use particles beams to cover a targeted tumor to achieve an ideal dose distribution. Figure 3 shows the profiles of proton and carbon ions in comparison to X-rays. As can be seen, the dose profiles of protons and carbon ions display a distinct localized peak, called a Bragg Peak. The Bragg Peak makes TNNLS modeling particularly suitable for planning particle therapy, where the goal is to find the weighting for each particle beam to created a distribution as close to the target distribution as possible. Our experimental results on carbon therapy are shown in Section IV-C.

A. *Random problems*

We evaluate the performance of the algorithm on randomly generated problems by varying size, aspect ratio and sparsity of the input data.

1) Size: We have applied the algorithm to a suite of 200 randomly generated problems of varying size. This is done by sampling the entries of $\boldsymbol{W}, \boldsymbol{x}$ uniformly from $[0, 1]$. We set the size of the matrix $\boldsymbol{W}$ to be $300i \times 200i$, where i ranges from 1 to 40. And, for each size, we create 5 randomly generated problems.

First, we applied the NNLS algorithms FNNLS [Bro and De Jong, 1997], RAND-PQN-NNLS [Boutsidis and Drineas, 2009] and PQN-NNLS [Kim et al., 2006] to solve this set of

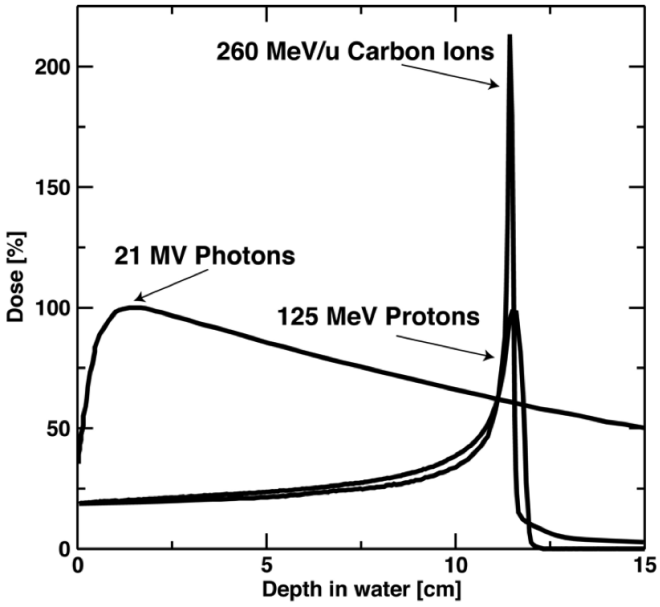

Fig. 3. Dose profiles of proton and antiproton beams

problems. The running times for these algorithms are shown in Figure 4 with solid markers.

Next, we applied our reduction to obtain the SVM problem, and used the OCAS algorithm to find most of the zeros of the solution vector, and finally solved for the nonzeros by giving it to each of the 3 NNLS solvers mentioned above. The objective value obtained for the exact solver and the corresponding OCAS initialized solver match up to 6 significant digits. The parameters in OCAS initialized solvers are set as $(C, \epsilon) = (10, 10^{-4})$ for all problems. We plot the running time for the 3 OCAS initialized algorithms in Figure 4 using hollow markers. For each size, we plot the mean of running times. Note that these running times are for the entire procedure of reduction, running the SVM solver, and exact solution of the smaller NNLS problem. Except for the smallest cases, the OCAS initialized solver beats the corresponding exact solver. For larger matrices, the figure shows at least an order of magnitude improvement in the running times.

2) Aspect ratio: We did a similar analysis on a suite of 30 randomly generated problems of varying aspect ratio. The number of rows is set to 12000 and the number of columns is $1200 \times i$ where i goes from 1 to 6. For each i, we generate 5 random instances. The parameters in OCAS initialized solvers are set as $(C, \epsilon) = (10, 10^{-4})$. The running times for all the six solvers are shown in Figure 4.

3) Sparsity: We also compared the running times for two of the solvers by varying the sparsity of the input matrix $\boldsymbol{W}$. We choose a fixed sized problem of size 1200×800 and varied the sparsity from 0.1 to 1.0 in increments of 0.1. Note that 1.0 corresponds to all the elements being nonzero. The parameters in OCAS initialized solvers are set as $(C, \epsilon) = (100, 10^{-6})$. The plots of running times for the two solvers FNNLS and PQN-NNLS and their OCAS initalized solvers are shown in

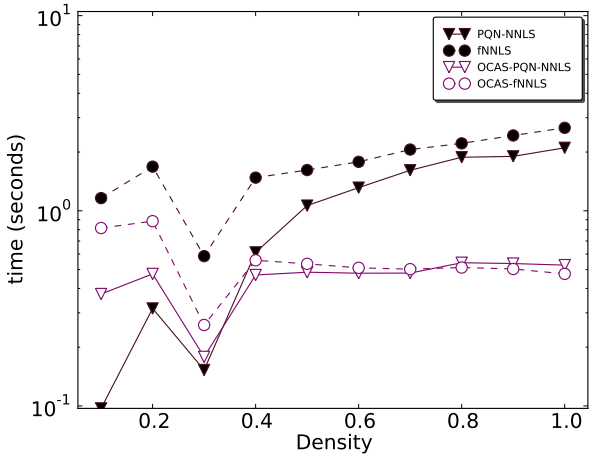

Fig. 5. We plot running times for FNNLS and PQN-NNLS and their corresponding OCAS initialized solvers. The problem size is fixed at 1200×800 and we vary the density from 0.1 to 1.0. These are the mean times for 10 runs at each density level.

Figure IV-A.3.

B. Phantom tumor dataset

We have also applied our TNNLS solver to a data set from a phantom commonly used for benchmarking radiosurgery treatment planning systems [Luan et al., 2009]. The phantom contains a C-shaped tumor surrounding a spherical critical structure and simulates a spine tumor case. In this data set, the size of the input matrix W is 42875×20268 and the input vector x is of size 42875. Clinically, each column of the matrix W represents the radiation energy distribution deposited by a "shot" of radiation in Gamma Knife radiosurgery. The matrix x represents the ideal radiation energy deposition as prescribed

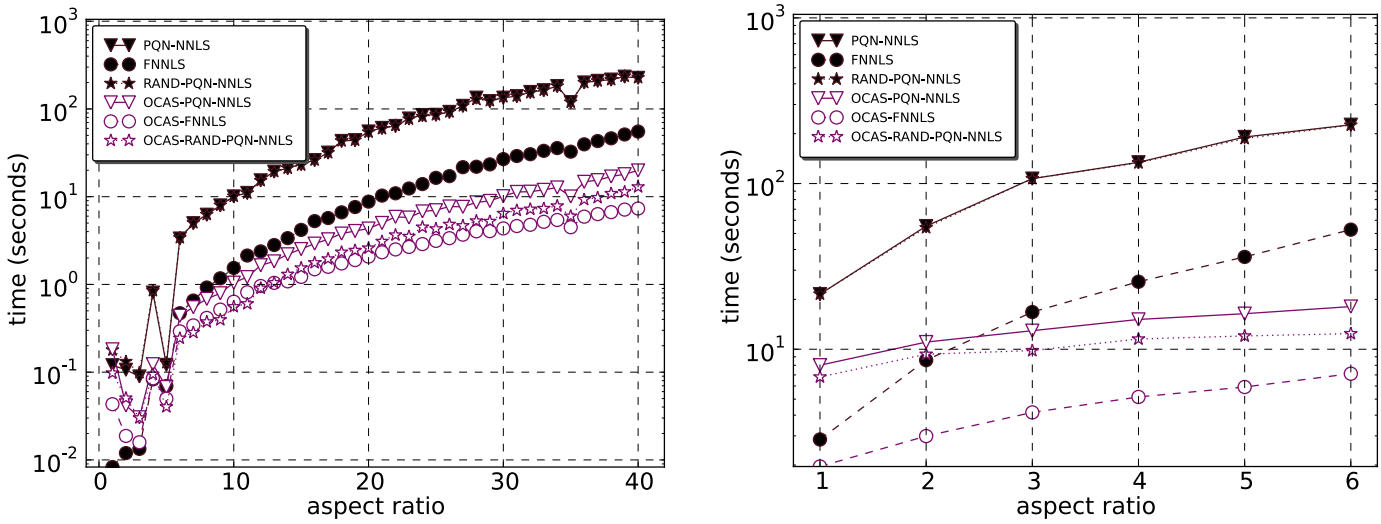

Fig. 4. We plot the running time for all 6 approaches. Lines with filled markers correspond to NNLS solvers and hollow markers correspond to NNLS solvers initialized by the OCAS solver and our reduction technique. The x-axis is indexed by i, which controls the size of the input matrix, which is $300i \times 200i$.

Dataset	FNNLS	PQN-NNLS	OCAS-FNNLS	OCAS-PQN-NNLS	Scaling factor
Phantom tumor	18.69	78.66	**1.0**	5.48	186s
Skull-base tumor	2.21	43.16	**1.0**	9.01	906s
Prostate tumor	1.46	2.04	**1.0**	1.35	134s

Fig. 6. Running times on the different datasets using the solvers FNNLS, PQN-NNLS and RAND-PQN-NNLS and their corresponding OCAS initialized counterparts can be obtained by multiplying the corresponding entry with the scaling factor. Comparison between running times across different NNLS solvers should be taken with a grain of salt for the stopping criterion for each solver is potentially different. However, stopping criterion between a solver and its OCAS initialized solver are the same and thus can be compared.

by the physician. The sought variable h denotes the beam-on time each shot (i.e., a column of W) to create a radiation dose distribution that is as close to the ideal as possible. All solvers have the same objective value up to 6 significant digits. The parameters in OCAS initialized solvers are set as $(C, \epsilon) = (10, 10^{-4})$. The running times are shown in Figure 6.

C. Real tumor datasets

Besides randomly generated data, we also applied our TNNLS solver to two real radiation therapy data sets, both obtained from the German Cancer Research Center (DKFZ), of Heidelberg, Germany. The first of these is a skull base tumor case that was treated with carbon ion therapy. In this data set, the size of input matrix W is 227920×6505 and the input vector x is 227920. Just like the dataset in Section IV-B, the columns of W represent the radiation energy distribution of an ion beam, while the vector x represents the prescription. The goal of the optimization is to calculate the beam-on time for each individual beam. Note that the default setting of the soft margin to 10 didn't converge to the exact solution, so we used 100 for this dataset. The objective values match upto 6 significant digits as before. The parameters in OCAS initialized solvers are set as $(C, \epsilon) = (100, 10^{-4})$. The running times are shown in Table 6.

The second real data set is a prostate carcinoma case that was treated using two opposing beams. In this data set, the input matrix is 8284×7388. The parameters in the OCAS initialized solvers are set as $(C, \epsilon) = (100, 10^{-5})$.

D. Discussion

The speed up in the case of Real Tumor dataset is only around 2 times compared to the magnitude improvement we get in the case of Random and Phantom tumor datasets for the FNNLS solver. This is to be expected because the input matrix is "tall" and our algorithm does better when we are dealing with "fat" matrices. We have used the exact solver from Kim et al. [2006] in combination with the randomized algorithm of Boutsidis and Drineas [2009]. Other exact solvers can also be used.

Note, the speedup is not uniform across the various problems. As, we noted in section "approximate solvers", this depends on the spread of the data. If the datapoints are not clustered along the maximum margin hyperplane, we can solve it pretty quickly using the approximate SVM solver. However, for cases, where this is not true, the running time for our solver is increased.

In the case where the matrix $\boldsymbol{W}$ is sparse, we found that a more aggressive setting for the parameters C, ϵ was required.

V. Conclusions and Future Work

We have shown a reduction from TNNLS to a single-class SVM. This gave us insight into the connection between nonnegativity and sparsity and further enabled us to propose an efficient algorithm to solve the TNNLS problem. The new algorithm is simple to implement and involves combining an SVM solver (such as OCAS) with an exact NNLS solver. We have shown its application to random problems, as well as to two real examples of dose calculation in radiation therapy. Also, we show that nonnegativity corresponds to sparsity depending on how many elements lie on the maximum-margin hyperplane. This explains the connection between nonnegativity and sparsity posed in the work on compressive sensing by O'Grady and Rickard [2008]. Also, the running time depended on the spread of data and we would like explore when it make sense to use the reduction and how to set the parameters in the SVM to best trade-off between the approximate SVM solver and an exact NNLS solver. This approach can potentially be also used for solving Nonnegative Matrix Factorization (NMF), which is a subject for future work. Our approach seems to be more suitable for "fat" matrices, where the number of rows and columns are similar. Also, if we have an additional L_1 regularizer in the objective function, our framework can be extended to handle it.

The NNLS solver used in Luan et al. [2009] used advanced techniques such as multi-threading and vector commands, and has only floating point precision, while our TNNLS solver has double precision. A similar speed up is conceivable if solvers used in this paper were implemented in a similar manner.

Acknowledgement

The first author would like to acknowledge the support from NIBIB grants 1 R01 EB 000840 and 1 R01 EB 005846. The second author was supported by NIMH grant 1 R01 MH076282-01. The latter two grants were funded as part of the NSF/NIH Collaborative Research in Computational Neuroscience Program. Also, we would like to acknowledge the software support from the Austin group and the RPI group. The third author would like to acknowledge the support from grants NCI R01CA117997 and NSF CBET-0755054.

References

H. Avron, P. Maymounkov, and S. Toledo. Blendenpik: Supercharging lapack's least-squares solver. SIAM Journal on Scientific Computing, 32(3):1217–1236, 2010. ISSN 1064-8275.

C. Boutsidis and P. Drineas. Random projections for the nonnegative least-squares problem. Linear Algebra and its Applications, 431(5-7):760–771, 2009. ISSN 0024-3795.

R. Bro and S. De Jong. A fast non-negativity-constrained least squares algorithm. Journal of Chemometrics, 11(5): 393–401, 1997. ISSN 1099-128X.

A.M. Bruckstein, M. Elad, and M. Zibulevsky. A non-negative and sparse enough solution of an underdetermined linear system of equations is unique. Submitted to IEEE Transactions on Information Theory, 2008.

C.I. Chang and D.C. Heinz. Constrained subpixel target detection for remotely sensed imagery. Geoscience and Remote Sensing, IEEE Transactions on, 38(3):1144–1159, 2000. ISSN 0196-2892.

L. Chin and W. Regine. Principles and practice of stereotactic radiosurgery. Springer Verlag, 2008. ISBN 0387710698.

T.F. De Laney and H.M. Kooy. Proton and charged particle radiotherapy. Lippincott Williams & Wilkins, 2007. ISBN 0781765528.

D.L. Donoho and J. Tanner. Thresholds for the recovery of sparse solutions via l1 minimization. In Information Sciences and Systems, 2006 40th Annual Conference on, pages 202–206. IEEE, 2006. ISBN 1424403499.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. In Proceedings of the 25th international conference on Machine learning, pages 320–327. ACM, 2008.

V. Franc, V. Hlavac, and M. Navara. Sequential coordinate-wise algorithm for the non-negative least squares problem. In Computer Analysis of Images and Patterns, page 407, 2005. URL http://dx.doi.org/10.1007/11556121_50.

Thilo-Thomas Frieß, Nello Cristianini, and Colin Campbell. The Kernel-Adatron algorithm: a fast and simple learning procedure for support vector machines. In Proc. 15th International Conf. on Machine Learning, pages 188–196. Morgan Kaufmann, San Francisco, CA, 1998.

S. Hochreiter and M.C. Mozer. Monaural separation and classification of mixed signals: A support-vector regression perspective. In 3rd International Conference on Independent Component Analysis and Blind Signal Separation, San Diego, CA. Citeseer, 2001.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In Proceedings of the 25th international conference on Machine learning, ICML '08, pages 408–415, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: http://doi.acm.org/10.1145/1390156.1390208. URL http://doi.acm.org/10.1145/1390156.1390208.

V. Kecman. Learning and soft computing: Support vector machines, neural networks, and fuzzy logic models. The MIT press, 2001. ISBN 0262112558.

D. Kim, S. Sra, and I.S. Dhillon. A new projected quasi-newton approach for the nonnegative least squares

problem. Citeseer, 2006.

CL Lawson and RJ Hanson. Solving least squares problems, 340 pp, 1974.

F. Li, Y. Yang, and E. Xing. From lasso regression to feature vector machine. Advances in Neural Information Processing Systems, 18:779, 2006. ISSN 1049-5258.

S. Luan, N. Swanson, Z. Chen, and L. Ma. Dynamic gamma knife radiosurgery. Physics in Medicine and Biology, 54: 1579, 2009.

M. Merritt and Y. Zhang. Interior-point gradient method for large-scale totally nonnegative least squares problems. Journal of optimization theory and applications, 126(1): 191–202, 2005. ISSN 0022-3239.

P.D. O'Grady and S.T. Rickard. Compressive sampling of non-negative signals. In Machine Learning for Signal Processing, 2008. MLSP 2008. IEEE Workshop on, pages 133–138. IEEE, 2008.

J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines, 1998. URL citeseer.ist.psu.edu/platt98sequential.html.

Bernhard Schölkopf and Alexander J. Smola. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning). The MIT Press, 2001. ISBN 0262194759. URL http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/0262194759.

M.H. Van Benthem and M.R. Keenan. Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. Journal of chemometrics, 18(10):441–450, 2004. ISSN 1099-128X.

L. Wang. Support Vector Machines: theory and applications. Springer Verlag, 2005. ISBN 3540243887.

E. Zeng and M. Ogihara. NONNEGATIVE LEAST SQUARE–A NEW LOOK INTO SAGE DATA. In Proceedings of CSB, volume 9, 2009.