ESANN'2007 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2007, d-side publi., ISBN 2-930307-07-2.

Deploying SDP for machine learning

Tijl De Bie*

Katholieke Universiteit Leuven, OKP Research Group,
Tiensestraat 102, 3000 Leuven, Belgium.
University Of Bristol, Dept. of Engineering Mathematics,
Queen's Building, University Walk, Bristol, BS8 1TR, UK.

Abstract. We discuss the use in machine learning of a general type of convex optimisation problem known as semi-definite programming (SDP) [1]. We intend to argue that SDP's arise quite naturally in a variety of situations, accounting for their omnipresence in modern machine learning approaches, and we provide examples in support.

1 Introduction

The recent flourishing of convex optimisation in machine learning has probably been sparked by the invention of the Support Vector Machine (SVM) [2]. It was quickly recognised that this classification method matches Neural Networks classifiers (NN) in flexibility, while it outclasses them in at least two ways. First, the SVM objective is convex and can be optimised on polynomial time, whereas the NN objective is non-convex and hard to optimise. And second, empirically SVM classifiers are often observed to generalise better than NN classifiers.

Another research domain in which this paper is rooted is combinatorial optimisation. One approach to combinatorial optimisation is to loosen, i.e. to *relax* the difficult combinatorial constraints, so as to arrive at a convex optimisation problem [3, 4]. Then, an approximate solution of the unrelaxed original problem can be derived from the exact global solution of the relaxed problem by 'projecting' the relaxed solution on the unrelaxed constraint set. This approach often comes with theoretical guarantees in terms of computation time and approximation quality. Also empirically the results are often strikingly good.

Both these successes have inspired machine learning researchers, in particular those active in kernel methods, to focus on algorithms that can be phrased as or relaxed to convex optimisation problems. In this paper, we will focus on the class of semi-definite programming (SDP) problems as a rich subclass of convex optimisation. SDP's have recently received considerable attention both from within machine learning (mainly interested in *using* SDP as a black box tool), as well as from the convex optimisation community in *designing efficient SDP solvers* (hence designing the internal wheels that drive this black box). As we will see in this paper, this unique combination of technology push and market pull has led to interesting scientific results with practical applicability.

This paper intends to provide a non-exhaustive overview of the use of SDP in machine learning. Besides that, one small section is devoted to the technology used to solve SDP's efficiently, and contains pointers to freely available solvers.

*This work is supported by CoE EF/05/007 SymBioSys, and GOA/2005/04, both from the Research Council K.U.Leuven. We are indebted to Nello Cristianini for insightful discussions.

2 Semi-Definite Programming as a template problem

The distinguishing ingredient of a SDP's is a matrix inequality constraint. For $\mathbf{F}$ a square symmetric matrix, with the matrix inequality $\mathbf{F} \succeq \mathbf{0}$ we mean that all eigenvalues of $\mathbf{F}$ are nonnegative, and we say that the matrix $\mathbf{F}$ is constrained to be positive semi-definite (PSD), or that $\mathbf{F}$ belongs to the cone of positive semi-definite matrices. Then, a general SDP problem can be written as:

$$\boxed{\begin{array}{rl} \min_{\mathbf{x}} & \mathbf{a}'\mathbf{x}, \\ \text{s.t.} & \mathbf{F}_0 + \sum_{i=1}^{n} x_i \mathbf{F}_i \succeq \mathbf{0}, \\ & \mathbf{B}'\mathbf{x} = \mathbf{0}. \end{array}}$$

We should stress that, despite its apparent simplicity, this problem encompasses large classes of convex optimisation problems, such as linear programs (LP), convex quadratic programs (QP), second order cone programs (SOCP), and more. See [1] for an excellent overview of the flexibility of SDP's.

In particular, if all $\mathbf{F}_i$ have the same block-diagonal structure, the PSD constraint is equivalent to a set of PSD constraints on each of the blocks of $\mathbf{F}_0 + \sum_{i=1}^{n} x_i \mathbf{F}_i$. If such a block is of size 1×1, the PSD constraint for that block reduces to a linear inequality constraint. Furthermore, if such a block has size 2×2, the PSD constraint reduces to a quadratic constraint. For example: $\begin{pmatrix} 1 & x_2 \\ x_2 & x_1 \end{pmatrix} \succeq \mathbf{0} \Leftrightarrow x_1 \geq x_2^2$. This example is in fact a special case of an extremely useful lemma, known as the Schur complement lemma.

Lemma 1 (Schur complement lemma) *For symmetric* $\mathbf{A} \succ \mathbf{0}$ *and* $\mathbf{C} \succeq \mathbf{0}$*:*

$$\mathbf{C} \succeq \mathbf{B}^T \mathbf{A}^{-1} \mathbf{B} \quad \Leftrightarrow \quad \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{pmatrix} \succeq \mathbf{0}.$$

In practice, it is convenient and more efficient to specify separate linear, quadratic, and PSD constraints, rather than subsuming them into one block-structured PSD constraint. We will do this in the sequel of this paper.

3 SDP in machine learning

3.1 Finding the right kernel matrix

Let us consider a data set $X = \{x_1, x_2, \cdots, x_n\}$. Kernel methods are designed to find patterns in such data without making explicit use of the data representations x_i. Instead they rely solely on the use of kernel function evaluations between pairs of data items x_i and x_j, a kernel function being defined as a symmetric and positive semi-definite function $k : \mathcal{X}^2 \to \mathbb{R} : (x_i, x_j) \to k(x_i, x_j)$.

For a given data set $X = \{x_1, x_2, \cdots, x_n\}$, all kernel evaluations between all pairs of data items are usually summarised in a single $n \times n$ kernel matrix, denoted by $\mathbf{K}$, with $\mathbf{K}(i, j) = k(x_i, x_j)$. By the positive semi-definiteness of k, we know that $\mathbf{K} \succeq \mathbf{0}$ for any data set X. In practice, it often suffices to know that $\mathbf{K} \succeq \mathbf{0}$ on the training data, or on the training data and test data together (when these test data are available).

3.1.1 Data fusion

A major issue in kernel methods is the choice of the kernel function. Depending on the data type, various options exist, and model selection techniques are required to make a choice. However, an alternative approach is less drastic, and aims at combining several plausible choices rather than selecting just one. Assume that several kernel matrices $\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_m$ are given, computed with different kernel functions $k_1, k_2, \ldots, k_m$. Then, a new kernel matrix that integrates information from each of these individual matrices can be computed as a linear combination:

$$\mathbf{K} = \sum_{i=1}^{m} \mu_i \mathbf{K}_i.$$

For this kernel to be valid, it needs to be positive semi-definite. Thus one sees that the constraint $\sum_{i=1}^{m} \mu_i \mathbf{K}_i \succeq \mathbf{0}$ naturally arises here, and subject to this constraint the weights μ_i can be chosen so as to optimise a performance criterium, such as a statistical bound.

This strategy has been developed first in [5] and applied in [6] in a classification setting. Let $M(\mathbf{K}, \mathbf{y})$ be the size of the margin achieved by an SVM classifier for a kernel $\mathbf{K}$ and label vector $\mathbf{y}$ (see left box below for a formal definition in terms of the dual SVM formulation). Then a data fusion method that maximises this margin is given by the right box below:

$$\boxed{\begin{array}{rl} \frac{1}{M(\mathbf{K},\mathbf{y})^2} = \max_{\boldsymbol{\alpha}} & 2 \cdot \mathbf{1}'\boldsymbol{\alpha} - \boldsymbol{\alpha}'(\mathbf{K} \odot \mathbf{y}\mathbf{y}')\boldsymbol{\alpha} \\ \text{s.t.} & \mathbf{y}'\boldsymbol{\alpha} = 0 \\ & \boldsymbol{\alpha} \geq \mathbf{0} \end{array}} \quad \boxed{\begin{array}{ll} \min_{\boldsymbol{\mu}} & \frac{1}{M(\mathbf{K},\mathbf{y})^2} \\ & \mathbf{K} = \sum \mu_j \mathbf{K}_j \\ & \text{trace}\,(\mathbf{K}) = 1 \\ & \mathbf{K} \succeq \mathbf{0} \end{array}}$$

Clearly, this formulation is not yet in the standard SDP form shown in Section 2. However, using duality theory and the Schur complement lemma it may be reformulated as such [5]. Space requirements force us to omit the details here.

3.1.2 Reducing diagonal dominance of a kernel matrix

Besides for data fusion, SDP is handy in adapting the kernel matrix slightly in order to satisfy some criterium. For example, certain kernel matrices are known to be strongly diagonal dominant, which may hamper learning in certain cases. In that case, SDP provides a means to adapt the kernel matrix by subtracting a diagonal matrix from it, which is as large as possible while leaving the resulting matrix PSD. A suitable reformulation of this problem in optimisation terms leads to the following SDP [7]:

$$\boxed{\begin{array}{rl} \min_{\mathbf{d}} & -\mathbf{1}'\mathbf{d}, \\ \text{s.t.} & \mathbf{K} - \text{diag}\,(\mathbf{d}) \succeq \mathbf{0}. \end{array}}$$

3.2 Relaxations of labelling problems

Let us now discuss a totally different use of SDP in machine learning.

3.2.1 *Clustering based on the normalised graph cut*

Consider an undirected weighted graph over a set $\mathcal{S}$ of n vertices $\mathbf{x}_i$, with weights $a_{ij} \geq 0$ between $\mathbf{x}_i$ and $\mathbf{x}_j$ summarised in a symmetric affinity matrix $\mathbf{A}$. Vertices connected with a larger edge weight are assumed to be more similar in some sense, and conversely, dissimilar points are connected by an edge with small edge weight. Our goal is to arrive at a bipartitioning of the vertices so that similar vertices are grouped together, and dissimilar vertices are separated.

Satisfying the former condition (to group similar points) is easily achieved by searching for a bipartitioning of $\mathcal{S}$ into two disjoint sets $\mathcal{N}$ and $\mathcal{P}$ that minimises the sum of the weights of edges connecting a vertex in one cluster ($\mathcal{N}$) with one in the other ($\mathcal{P}$). Clearly, simply minimising this so-called *cut cost* is not sufficient, as the minimum is trivially achieved by assigning all data points to one single cluster. However, combining this objective with the latter condition solves this problem, i.e. by simultaneously forcing the bipartitioning to separate dissimilar vertices. In practice this can be achieved by enforcing that neither of the clusters becomes too small. Both these aspects can be jointly taken into account by minimising the so-called *normalised cut cost*, defined as:

$$\frac{\mathrm{cut}(\mathcal{P},\mathcal{N})}{\mathrm{assoc}(\mathcal{P},\mathcal{S})}+\frac{\mathrm{cut}(\mathcal{N},\mathcal{P})}{\mathrm{assoc}(\mathcal{N},\mathcal{S})}=\left(\frac{1}{\mathrm{assoc}(\mathcal{P},\mathcal{S})}+\frac{1}{\mathrm{assoc}(\mathcal{N},\mathcal{S})}\right)\cdot\mathrm{cut}(\mathcal{P},\mathcal{N}),$$

where $\mathrm{cut}(\mathcal{P},\mathcal{N})=\mathrm{cut}(\mathcal{N},\mathcal{P})=\sum_{i:\mathbf{x}_i\in\mathcal{P},j:\mathbf{x}_j\in\mathcal{N}} a_{ij}$ is the cut between sets $\mathcal{P}$ and $\mathcal{N}$, and $\mathrm{assoc}(\mathcal{P},\mathcal{S})=\sum_{i:\mathbf{x}_i\in\mathcal{P},j:\mathbf{x}_j\in\mathcal{S}} a_{ij}$ the association between sets $\mathcal{P}$ and the full sample $\mathcal{S}$.

To get a handle on this cost function, let us reformulate it into algebraic terms using a label vector $\mathbf{y}\in\{-1,1\}^n$ (indicating cluster membership), the affinity matrix $\mathbf{A}$, the degree vector $\mathbf{d}=\mathbf{A1}$ and associated matrix $\mathbf{D}=\mathrm{diag}(\mathbf{d})$, and shorthand notations $s_+=\mathrm{assoc}(\mathcal{P},\mathcal{S})$, $s_-=\mathrm{assoc}(\mathcal{N},\mathcal{S})$, and $s=s_++s_-$. Note that $\mathrm{cut}(\mathcal{P},\mathcal{N})=\frac{(\mathbf{1}+\mathbf{y})'}{2}\mathbf{A}\frac{(\mathbf{1}-\mathbf{y})'}{2}=\frac{1}{4}\left(-\mathbf{y}'\mathbf{A}\mathbf{y}+\mathbf{1}'\mathbf{A}\mathbf{1}\right)=\frac{1}{4}\mathbf{y}'(\mathbf{D}-\mathbf{A})\mathbf{y}$. Furthermore, $s_+=\mathrm{assoc}(\mathcal{P},\mathcal{S})=\frac{1}{2}\mathbf{1}'\mathbf{A}(\mathbf{1}+\mathbf{y})=\frac{1}{2}\mathbf{d}'(\mathbf{1}+\mathbf{y})$ and $s_-=\frac{1}{2}\mathbf{d}'(\mathbf{1}-\mathbf{y})$. Then we can write the minimisation of the normalised cut cost as:

$$\begin{array}{rl}\min_{\mathbf{y},s_+,s_-} & \frac{s}{4s_+s_-}\cdot\mathbf{y}'(\mathbf{D}-\mathbf{A})\mathbf{y}\\ \text{s.t.} & \mathbf{y}\in\{-1,1\}^n, \qquad \mathbf{d}'\mathbf{y}=s_+-s_-, \qquad s_++s_-=s.\end{array}$$

Unfortunately, this is a provably hard optimisation problem. However, it can be approximated by means of relaxation techniques. The first relaxation described for this problem is a spectral relaxation [8]. More recently, we have developed a tighter relaxation that is based on SDP [9]. We briefly summarise the derivation.

Using $\mathbf{\Gamma}=\mathbf{y}\mathbf{y}'$ we can write an equivalent optimisation problem:

$$\begin{array}{rl}\min_{\mathbf{\Gamma},s_+,s_-} & \frac{s}{4s_+s_-}\langle\mathbf{\Gamma},\mathbf{D}-\mathbf{A}\rangle \qquad\qquad \left[\text{where } \langle\mathbf{X},\mathbf{Y}\rangle\triangleq\mathrm{trace}\left(\mathbf{X}'\cdot\mathbf{Y}\right)\right]\\ \text{s.t.} & \mathbf{\Gamma}=\mathbf{y}\mathbf{y}', \quad \mathbf{y}\in\{-1,1\}^n,\\ & \langle\mathbf{\Gamma},\mathbf{d}\mathbf{d}'\rangle=(s_+-s_-)^2=(s_++s_-)^2-4s_+s_-,\\ & s_++s_-=s, \quad s_+>0, \quad s_->0.\end{array}$$

Note that these constraints imply that $\mathbf{\Gamma}'=\mathbf{\Gamma}\succeq\mathbf{0}$ and $\mathrm{diag}(\mathbf{\Gamma})=\mathbf{1}$. Hence we can relax the constraint set by adding these two redundant constraints, and dropping $\mathbf{\Gamma} = \mathbf{y}\mathbf{y}'$ and $\mathbf{y} \in \{-1, 1\}^n$. If we further substitute $q = \frac{s^2}{4s_+ s_-}$ and $\widehat{\mathbf{\Gamma}} = q\mathbf{\Gamma}$, we get the following SDP relaxation of the normalised cut minimisation:

$$\boxed{\begin{array}{rl} \min_{\widehat{\mathbf{\Gamma}},q} & \langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{D}-\mathbf{A}}{s} \rangle \\ \text{s.t.} & \widehat{\mathbf{\Gamma}}' = \widehat{\mathbf{\Gamma}} \succeq \mathbf{0}, \quad \operatorname{diag}(\widehat{\mathbf{\Gamma}}) = q\mathbf{1}, \quad \langle \widehat{\mathbf{\Gamma}}, \frac{\mathbf{d}\mathbf{d}'}{s^2} \rangle = q - 1. \end{array}}$$

In order to derive a valid label vector $\mathbf{y}$ based on the relaxed label matrix $\widehat{\mathbf{\Gamma}}$ various techniques are available, as discussed in e.g. [4, 9].

3.2.2 Transduction based on the normalised cut

Interestingly, this optimisation problem can be adapted in order to satisfy a given set of constraints on the labels (i.e., part of the labels are given to be 1 or -1). This leads to an algorithm to cluster subject to constraints, also known as transduction in the machine learning terminology. Enforcing label constraints can be done explicitly on the label matrix $\widehat{\mathbf{\Gamma}}$, or more efficiently by ensuring the constraints will hold constructively by reparameterising the label matrix [9].

In the transduction case, minimising the normalised cut cost can be regarded as a form of capacity control (the normalised cut cost being a regulariser). Searching for a label vector compatible with the training labels and with a small cost must ensure generalisation towards the unlabelled points, as shown in [10].

3.2.3 SVM transduction

A more common approach to transduction is based on SVM's, and is a hard combinatorial problem as well. Interestingly, it can be addressed using a similar relaxation approach, as shown in [11, 12].

4 On the theoretical and practical complexity of SDP's

We have argued that SDP's form a broad class of optimisation problems, including LP's, convex QP's, and SOCP's. Obviously, such a generality comes at a price. While the worst case complexity of interior point solvers for SDP's is provably polynomial, it is so with a high exponent: the complexity depends quadratically on the number of variables, and polynomially with an exponent of roughly 2.5 in the size of the PSD matrix.

Nevertheless, excellent interior point solvers for SDP's are available, such as SeDuMi [13] and SDPT3 [14]. Furthermore, in practice dedicated methods may exploit the problem structures to yield significantly faster methods. Additionally, tight approximation techniques may allow for dramatic speed-ups, as we have shown in [9]. Consequently, problems such as normalised cut cost clustering and transduction (3.2.1 and 3.2.2) can be solved for many thousands of vertices.

Additionally, recently a general purpose SDP solver has been developed that seems considerably faster and more memory efficient than standard interior point solvers, see e.g. [15]. One may hope that such breakthroughs may one day render SDP's as generally applicable as LP's are today.

5 Conclusions

We have provided a non-exhaustive list of SDP applications in machine learning. We had to omit discussions of other uses, such as for approximate inference in graphical models [16], distance metric learning [17], sparse PCA [18], kernel matrix completion [19], nonlinear dimensionality reduction [20], and much more.

References

[1] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[2] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, Cambridge, U.K., 2000.

[3] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[4] C. Helmberg. Semidefinite programming for combinatorial optimization. Habilitationsschrift ZIB-Report ZR-00-34, TU Berlin, Konrad-Zuse-Zentrum Berlin, 2000.

[5] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72, 2004.

[6] G. Lanckriet, T. De Bie, N. Cristianini, M. Jordan, and W. Stafford Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.

[7] J. S. Kandola, T. Graepel, and J. Shawe-Taylor. Reducing kernel matrix diagonal dominance using semi-definite programming. In *COLT'03*, pages 288–302, 2003.

[8] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[9] T. De Bie and N. Cristianini. Fast SDP relaxations of graph cut clustering, transduction, and other combinatorial problems. *JMLR*, 7:1409–1436, 2006.

[10] P. Derbeko, R. El-Yaniv, and R. Meir. Explicit learning curves for transduction and application to clustering and compression algorithms. *JAIR*, 22:117–142, 2004.

[11] T. De Bie and N. Cristianini. Convex methods for transduction. In *NIPS'03*, pages 73–80, 2004.

[12] T. De Bie and N. Cristianini. Semi-supervised learning using semi-definite programming. In O. Chapelle, B. Schoëlkopf, and A. Zien, editors, *Semi-supervised learning.* MIT Press, Cambridge-Massachussets, 2006.

[13] J.F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software, Special issue on Interior Point Methods (CD supplement with software)*, 11-12:625–653, 1999.

[14] R. H. Tütüncü K. C. Toh, M. J. Todd. Sdpt3 — a matlab software package for semidefinite programming. *Optimization Methods and Software*, 11/12:545–581, 1999.

[15] S. Burer and R.D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (series B)*, 95(2):329–357, 2003.

[16] M. J. Wainwright and M. I. Jordan. Log-determinant relaxation for approximate inference in discrete markov random fields. *IEEE Transactions on Signal Processing*, 54(6):2099–2109, June 2006.

[17] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *NIPS'02*, pages 505–512, 2003.

[18] A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. In *NIPS'04*, 2004.

[19] T. Graepel. Kernel matrix completion by semidefinite programming. In *Proc. of ICANN'02*, volume 2415/2002. Springer Berlin / Heidelberg, 2002.

[20] W. Weinberger, B. Packer, and L. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *Proc. of AISTATS'05*, 2005.