Kernel Matrix Completion by Semidefinite Programming

Thore Graepel

Institute of Computational Science
ETH Zürich, Schweiz
graepel@inf.ethz.ch

Abstract We consider the problem of missing data in kernel-based learning algorithms. We explain how semidefinite programming can be used to perform an approximate weighted completion of the kernel matrix that ensures positive semidefiniteness and hence Mercer's condition. In numerical experiments we apply a support vector machine to the XOR classification task based on randomly sparsified kernel matrices from a polynomial kernel of degree 2. The approximate completion algorithm leads to better generalisation and to fewer support vectors as compared to a simple spectral truncation method at the cost of considerably longer runtime. We argue that semidefinite programming provides an interesting convex optimisation framework for machine learning in general and for kernel-machines in particular.

1 Introduction

Positive semidefinite symmetric kernels, also referred to as Mercer kernels, lie at the heart of a large number of kernel-based learning algorithms such as support vector machines, Gaussian processes, and kernel PCA [2]. Given an input sample $\boldsymbol{x} := \{x_1, \ldots, x_m\}$ of size m, the kernel matrix $\mathbf{K}$ is an $m \times m$ matrix with entries

$$k_{ij} := k\left(x_i, x_j\right) = \left\langle \boldsymbol{\phi}\left(x_i\right), \boldsymbol{\phi}\left(x_j\right) \right\rangle \ ,$$

where $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{F}$ is a mapping from input space $\mathcal{X}$ to some feature space $\mathcal{F}$, whose existence follows from the positive semidefiniteness of the kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as a consequence of Mercer's theorem. While many of the kernel-based algorithms have shown excellent performance in practice, they generally suffer from the $\mathcal{O}\left(m^2\right)$ scaling in memory requirements and computational costs induced by the $m \times m$ kernel matrix. It has been shown, that low-rank approximations of $\mathbf{K}$ retain most of the information of $\mathbf{K}$ while reducing both memory requirements and computational costs. On similar grounds it is expected that a certain number of missing components of $\mathbf{K}$ should not affect the performance of learning algorithms based on $\mathbf{K}$ [1].

Given a vectorial input representation, that is, $\mathcal{X} := \mathbb{R}^n$, it is natural to consider the case of missing vector components, e.g., $\mathbf{x} = (0.5, ?, ?, 1.0)$. However, kernel methods do not assume a vectorial input representation, but the kernel

function k can be defined over any set $\mathcal{X}^2$ of pairs of inputs. Hence, we consider the case of missing entries in the kernel matrix $\mathbf{K}$. It should be noted that the missing data could result not only from missing values in a given data sample, but also from deliberately avoiding the calculation of all the m^2 kernel values in the case of computationally expensive kernel functions such as those acting on strings, graphs, or generally more structured, complex representations [3].

2 Completing Kernel Matrices

In this work, we consider the problem of completing kernel matrices $\mathbf{K}^*$ with missing values as a positive semidefinite optimisation problem [4]. Let $\mathbf{W} = \mathbf{W}'$ be a symmetric $m \times m$ matrix of weights satisfying $\forall i \in \{1, \ldots, m\} : \mathbf{W}_{ii} > 0$. The weighted best approximate completion (AC) problem is given by

$$\begin{aligned} \min_{\mathbf{K}} \quad & \|\mathbf{W} \circ (\mathbf{K} - \mathbf{K}^*)\|^2 \\ \text{subject to} \quad & \mathcal{A}\mathbf{K} = \mathbf{b} \quad \text{and} \quad \mathbf{K} \succeq \mathbf{0}\,, \end{aligned} \tag{1}$$

where $\mathbf{A} \circ \mathbf{B}$ denotes the Hadamard or componentwise product between matrices $\mathbf{A}$ and $\mathbf{B}$, and the inner product $\langle \mathbf{A}, \mathbf{B} \rangle := \operatorname{trace}(\mathbf{B}'\mathbf{A})$ leads to the Frobenius norm $\|\mathbf{A}\| := \sqrt{\langle \mathbf{A}, \mathbf{A} \rangle}$. The weight matrix $\mathbf{W}$ allows to individually tune, how much cost is incurred by any component of $\mathbf{K}$ for deviating from given components in $\mathbf{K}^*$. Hence, for a missing value k^*_{ij} we assume that the corresponding weight $w_{ij} = 0$ and in order to deal with complete matrices, we set $k^*_{ij} = 0$, as well. The linear equality constraint $\mathcal{A}\mathbf{K} = \mathbf{b}$ can be used to enforce a hard constraint on the elements of the solution matrix $\mathbf{K}_{\mathrm{AC}}$, with $\mathcal{A} : S_m \to \mathbb{R}^l$ being a linear operator from the set S_m of symmetric $m \times m$ matrices to $\mathbb{R}^l$. Finally, $\mathbf{K} \succeq \mathbf{0}$ is referred to as a linear matrix inequality, with $\mathbf{A} \succeq \mathbf{B}$ denoting the statement that $\mathbf{B} - \mathbf{A}$ is positive semidefinite.

The AC problem (1) can be viewed as a way of projecting the incomplete matrix to the cone of positive semidefinite matrices. An alternative way of doing this is to perform a spectral truncation: zero out the missing elements $k^*_{ij} = 0$, effectively assuming orthogonality of $\boldsymbol{\phi}(x_i)$ and $\boldsymbol{\phi}(x_j)$ in $\mathcal{F}$, perform an eigenvalue decomposition $\mathbf{K}^* =: \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}'$, discard the negative eigenvalues $\lambda_i < 0$ and their corresponding eigenvectors $\mathbf{u}_i$ to obtain matrices $\boldsymbol{\Lambda}_+$ and $\mathbf{U}_+$, and reassemble the kernel matrix $\mathbf{K}_{\mathrm{ST}} := \mathbf{U}_+\boldsymbol{\Lambda}_+\mathbf{U}'_+$, which is positive semidefinite by construction. As will be seen in Section 4, however, this technique does not yield satisfactory results.

3 Semidefinite Programming

The weighted best AC problem belongs to the class of convex optimisation problems called *semidefinite programs,* which can be solved efficiently using interior

point methods, similarly to linear and quadratic programs. In general, semidefinite programs can be cast into the form [5]

$$\begin{aligned} &\min_{\mathbf{X}} \ \langle \mathbf{C}, \mathbf{X} \rangle \\ &\text{subject to } \ \mathcal{A}\mathbf{X} = \mathbf{b} \quad \text{and} \quad \mathbf{X} \succeq 0 \,. \end{aligned} \tag{2}$$

The condition $\mathbf{X} \succeq 0$ constrains the solution to lie in the cone of positive semidefinite matrices. This constraint turns out to be very powerful, because it effectively corresponds to infinitely many constraints, $\forall \mathbf{x} \in \mathbb{R}^m : \mathbf{x}'\mathbf{X}\mathbf{x} \geq 0$. In order to appreciate the generality of SDPs, note that we have

$$\mathbf{X}_1 \succeq \mathbf{0}, \ldots, \mathbf{X}_k \succeq \mathbf{0} \quad \Leftrightarrow \quad \operatorname{diag}(\mathbf{X}_1, \ldots \mathbf{X}_k) \succeq \mathbf{0} \,, \tag{3}$$

for diagonal block matrices, and the Schur complement

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}' & \mathbf{C} \end{pmatrix} \succeq \mathbf{0} \quad \Leftrightarrow \quad \mathbf{C} - \mathbf{B}'\mathbf{A}^{-1}\mathbf{B} \succeq \mathbf{0} \,, \tag{4}$$

for $\mathbf{A} \succ \mathbf{0}$ a $k \times k$ symmetric, $\mathbf{C}$ an $l \times l$ symmetric, and $\mathbf{B}$ a $k \times l$ matrix.

From (3) it is easy to see that a linear program, $\min_{\mathbf{x} \geq \mathbf{0}} \langle \mathbf{c}, \mathbf{x} \rangle$, s.t. $\mathbf{A}\mathbf{x} = \mathbf{b}$, can be cast as a semidefinite program interpreting each single component x_i of $\mathbf{x}$ as a positive semidefinite matrix in (3). Similarly, a quadratically constrained convex quadratic program (QCQP), $\min_{\mathbf{x} \geq \mathbf{0}} \mathbf{x}'\mathbf{Q}_0\mathbf{x} - \mathbf{q}_0'\mathbf{x} - c_0$, s.t. $\forall i > 0 \, \mathbf{x}'\mathbf{Q}_i\mathbf{x} - \mathbf{q}_i'\mathbf{x} - c_i \leq \mathbf{0}$ can be formulated as a semidefinite program. Using (4) a convex quadratic constraint $\mathbf{x}'\mathbf{Q}_i\mathbf{x} - \mathbf{q}_i'\mathbf{x} - c_i \leq \mathbf{0}$ is equivalent to the linear matrix constraint

$$\begin{pmatrix} \mathbf{I} & \mathbf{C}_i\mathbf{x} \\ \mathbf{x}'\mathbf{C}_i' & \mathbf{q}_i'\mathbf{x} + c_i \end{pmatrix} \succeq \mathbf{0} \quad \Leftrightarrow \quad \mathbf{x}'\mathbf{Q}_i\mathbf{x} - \mathbf{q}_i'\mathbf{x} - c_i \leq \mathbf{0} \,,$$

where $\mathbf{Q}_i =: \mathbf{C}_i'\mathbf{C}_i$. Thus solving the semidefinite program

$$\begin{aligned} &\min_{t, \mathbf{x}} \ t \\ &\text{subject to } \begin{pmatrix} \mathbf{I} & \mathbf{C}_0\mathbf{x} \\ \mathbf{x}'\mathbf{C}_0' & \mathbf{q}_0'\mathbf{x} + c_0 + t \end{pmatrix} \succeq \mathbf{0} \quad \text{and} \quad \begin{pmatrix} \mathbf{I} & \mathbf{C}_1\mathbf{x} \\ \mathbf{x}'\mathbf{C}_1' & \mathbf{q}_1'\mathbf{x} + c_1 + t \end{pmatrix} \succeq \mathbf{0} \end{aligned}$$

is equivalent to solving the QCQP. It should be noted, however, that these formulations are often not efficient in practice, where the particular structure of the problem should be considered.

The two special cases discussed are of particular interest in machine learning: They show that the linear and quadratic programs that have become so popular in the context of support vector machines and kernel machines in general [2], can be cast into the more general framework of semidefinite programming (SDP), an insight that might pave the way for yet more sofisticated learning algorithms based on convex programming.

Returning to the AC problem (1) the objective of (1) has the same solution as $\min_{\mathbf{K}} (\operatorname{vec}(\mathbf{W})' \circ \operatorname{vec}'(\mathbf{K}) \, \mathbf{I}_{m^2} \operatorname{vec}(\mathbf{W}) \circ \operatorname{vec}(\mathbf{K}) - 2 \langle \mathbf{W} \circ \mathbf{W} \circ \mathbf{K}^*, \mathbf{K} \rangle)$. Using

again the Schur complement (4) we can write the AC problem as the following semidefinite program:

$$\min_{\mathbf{K},t} \; (t - 2\langle \mathbf{W} \circ \mathbf{W} \circ \mathbf{K}^*, \mathbf{K} \rangle)$$

$$\text{subject to } \mathbf{K} \succeq \mathbf{0} \quad \text{and} \quad \begin{pmatrix} \mathbf{I}_{m^2} & \operatorname{vec}(\mathbf{W}) \circ \operatorname{vec}(\mathbf{K}) \\ \operatorname{vec}(\mathbf{W})' \circ \operatorname{vec}'(\mathbf{K}) & t \end{pmatrix} \succeq \mathbf{0}\,.$$

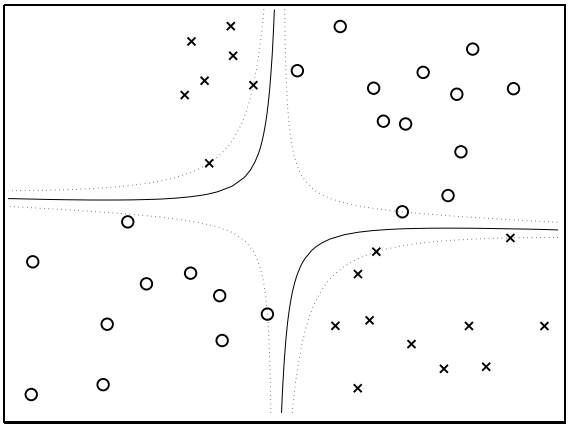

Figure 1. XOR classification task with support vector solution. Shown are 40 points drawn uniformly at random from $[-1,1] \times [-1,1]$ with their associated class labels $\circ$ and $\times$. The polynomial kernel of degree 2 is sufficient to separate the data correctly, based on 6 support vectors.

4 Numerical Experiments

To test the performance of the AC (1) we consider the simple yet non-trivial XOR classification task as shown in Figure 1. For the training sample we generated $m = 40$ points $\mathbf{x} := (x_1, x_2)$ uniformly at random in $[-1,1] \times [-1,1]$ and assigned them to two classes according to $y = \operatorname{sign}(x_1 x_2)$. The test sample consisted of 1 000 points generated according to the same scheme. Training and test sample were held fixed for the simulations. We calculated the kernel matrix $\mathbf{K}$ using the second order polynomial kernel $k(\mathbf{x}, \mathbf{z}) := (\mathbf{x}'\mathbf{z})^2$, which is the simplest kernel that solves the XOR problem. The training sample as well as the solution of a "hard margin" support vector machine based on the original kernel matrix $\mathbf{K}$ are shown in Figure 1.

We generated the matrix $\mathbf{K}^*$ by randomly selecting pairs of symmetric off-diagonal elements of $\mathbf{K}$ with probability $1 - d$ for deletion, thereby maintaining $\mathbf{K}^* = (\mathbf{K}^*)'$. We solved the AC problem (1) for $w_{ij} = 0$ for missing values and $w_{ij} = 1$ for non-missing values, thus not discriminating between different

elements of the kernel matrix $\mathbf{K}$. For the solution we used a dual-step-first interior point method[1][4]. For comparison we also performed the spectral truncation method as described at the end of Section 2. Based on the completed matrices $\mathbf{K}_{\mathrm{AC}}$ and $\mathbf{K}_{\mathrm{ST}}$ we trained a support vector machine and evaluated test error and number of support vectors.

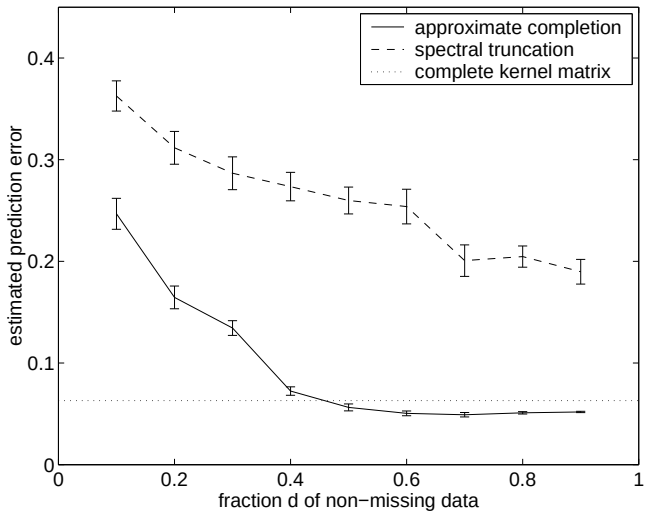

Figure 2. Estimated prediction error for XOR classification (see Figure 1) based on AC and spectral truncation as a function of the fraction d of non-missing data. The curves show an average over 30 instantiations of $\mathbf{K}^*$, and the errorbars indicate one standard deviation of the estimated mean.

Figure 2 shows the estimated prediction error for the XOR task as a function of the fraction d of non-missing data averaged over 30 random instances of $\mathbf{K}^*$. It is observed, that the AC yields by far lower error values than the spectral truncation method. The average error based on $\mathbf{K}_{\mathrm{AC}}$ approaches the error based on the original $\mathbf{K}$ for values of $d \geq 0.4$. In Figure 3 (left) we plotted the average number of support vectors for the same set of simulations. Again we see that $\mathbf{K}_{\mathrm{AC}}$ performs much better than $\mathbf{K}_{\mathrm{ST}}$, leading to fewer support vectors for values of $d > 0.2$. However, the superior performance has its price in terms of computational complexity. In Figure 3 (right) we plotted the average CPU time in seconds required for the matrix completion. While the spectral truncation methods is very fast independently of d, the AC takes orders of magnitude more time increasing with d. This behaviour is due to the dual step first strategy and would be reversed for the primal step first algorithm, indicating that in practice

[1] MATLAB code for the approximate completion problem can be found at `http://orion.math.uwaterloo.ca:80/~hwolkowi/`

the algorithm should be chosen depending on the value of d. Also it appears that the MATLAB implementation used could be accelerated. A more detailed discussion of the time complexity of the algorithm used can be found in [4].

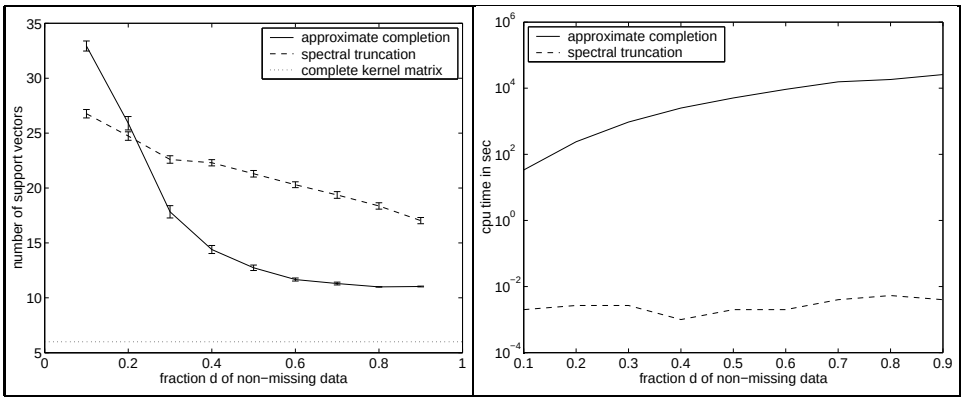

Figure 3. (left) Average number of support vectors as a function of the fraction d of non-missing data for the XOR task (for details see Figures 1 and 2). **(right)** CPU time required for a single kernel matrix completion.

5 Discussion and Conclusion

We demonstrated that semidefinite programming can be used to complete kernel matrices with missing values. It appears, however, that semidefinite programming may have many more applications in data analysis and machine learning. In particular, many problems in machine learning are related to finding appropriate distance measures and Euclidean embeddings that are closely related to semidefinite matrix completion as discussed here.

References

1. D. Achlioptas, F. McSherry, and B. Schölkopf. Sampling techniques for kernel methods. In S. B. Thomas G. Dietterich and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*.
2. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.
3. D. Haussler. Convolutional kernels on discrete structures. Technical Report UCSC-CRL-99-10, Computer Science Department, University of California at Santa Cruz, 1999.
4. C. Johnson, B. Kroschel, and H. Wolkowicz. An interior-point method for approximate positive semidefinite completions. *Comput. Optim. Appl.*, 9(2):175–190, 1998.
5. L. Vanderberghe and S. Boyd. Semidefinite programming. *SIAM Rev.*, 38(1):49–95, 1996.