TrIK-SVM : an alternative decomposition for kernel methods in Krein spaces

Gaëlle Loosli

TrIK-SVM : an alternative decomposition for kernel methods in Kreĭn spaces

Gaëlle Loosli[1,2]

1- PobRun
Brioude, France
2- UCA - LIMOS UMR 6158 CNRS
Clermont-Ferrand, France

Abstract. The proposed work aims at proposing a alternative kernel decomposition in the context of kernel machines with indefinite kernels. The original paper of KSVM (SVM in Kreĭn spaces) uses the eigen-decomposition, our proposition avoids this decompostion. We explain how it can help in designing an algorithm that won't require to compute the full kernel matrix. Finally we illustrate the good behavior of the proposed method compared to KSVM.

1 Introduction

Learning with indefinite kernels is a question that has been treated many times since the introduction of the hyperbolic tangent kernel (tanh) in the libsvm toolbox [1]. This particular kernel has never really shown its practical interest, but many other cases of indefinite kernels have been published, such as graph kernels or similarity based kernels [2]. In those cases, the need for an indefinite kernel is driven by the application field and the cost of avoiding the indefiniteness can be huge in terms of accuracy [3]. On the theoretical side, [4] have shown that learning with an indefinite kernel implies learning in reproducing kernel Kreĭn spaces (RKKS), in which basically the scalar product between a vector and itself can be negative. In [5], the KSVM algorithm is proven to produce a valid solution in an RKKS, even though the optmization in those spaces is not well defined. We anchor our work in the same framework, and we propose an upgraded algorithm for KSVM, named TrIK-SVM.

2 Essentials of RKKS

A Kreĭn space is a vector space in which the dot product of two identical vectors can provide a negative value. A well-known example of a Kreĭn space is the complex space, in which this property is denoted by $i^2 = -1$. In the learning context, it has been shown that trying to optimize a quadratic program (for instance an SVM) using an indefinite kernel matrix is actually a stabilization problem in a Reproducing Kernel Kreĭn space [4, 5].Those spaces are indefinite inner product spaces endowed with a Hilbertian topology without any requirement of positive-definiteness.

Definition 2.1. Kreĭn space [6] An inner product space $(\mathcal{K}, \langle .,. \rangle_{\mathcal{K}})$ is a Kreĭn space if there exists two Hilbert spaces $\mathcal{H}_+$, $\mathcal{H}_-$ spanning $\mathcal{K}$, with $f_+ \in \mathcal{H}_+$ and $f_- \in \mathcal{H}_-$, such that

$$\left| \begin{array}{ll} \forall f \in \mathcal{K} & f = f_+ \oplus f_- \\ \forall f, g \in \mathcal{K} & \langle f, g \rangle_{\mathcal{K}} = \langle f_+, g_+ \rangle_{\mathcal{H}_+} - \langle f_-, g_- \rangle_{\mathcal{H}_-} \end{array} \right. \tag{1}$$

In a nutshell, we provide here the essential facts that can be found in details in the above-cited literature:

- If $(\mathcal{H}_+, \langle ., . \rangle)$ and $(\mathcal{H}_-, -\langle ., . \rangle)$ are RKHS, $\mathcal{K} = \mathcal{H}_+ + \mathcal{H}_-$ is an RKKS.
- There is a representer thoerem in a RKKS, and the decomposition of the reproducing kernel $k = k_+ - k_-$ consists of reproducing kernels in the associated RKHS.
- Any symmetric indefinite kernel can be decomposed in k_+ and k_- and associated to an RKKS
- This decomposition is not unique

Solving a quadratic program in an RKKS ($\mathcal{K}$) is a stabilization problem that cannot be directly solved. However, it can be reformulated as an equivalent minimization problem [5] lying in an associated RKHS ($\mathcal{H}$) defined as the sum of the positive ($\mathcal{H}_+$) and negative ($\mathcal{H}_-$) parts of the RKKS. The solution of the minimization problem lies in $\mathcal{H}$ and thus needs to be projected back to $\mathcal{K}$. While the original paper shows the interest of KSVM considering the accuracy of the method on a large variety of problems, KSVM is not used in practice. Indeed, it suffers from several major practical drawbacks: it requires to compute the eigen-decompostion of the matrix (computation time and stability issues) and the final solution is full (the sparsity is lost during the projection) back to the RKKS. Recently, several papers dealing with indefinite matrices have noted those drawbacks and proposed alternative methods. Essentially, some paper are proposing minimization methods that ignore the stabilization problem [7] and others rely on using kernel as features, which is a way to embed the RKKS in an RKHS [8, 9]. The first category of methods are likely to provide an admissible but not optimal solution in the RKKS. The second category usually works pretty well but might hide some information from the negative part. One can note a major improvement over KSVM in terms of computational complexity, brought in [10] in the form of a carefully approximated version, based on kernel approximation, partial eigen-decomposition and CVM algorithm. The final solution is also approximated in order to be sparser [11].

The stabilization problem We recall here the stabilization problems in both RKKS and RKHS to exhibit the place of the kernel decomposition that will be discussed later. Let $\{x_i \in \mathcal{X}, y_i \in [-1, 1], \forall i \in [1, \dots, \ell]\}$ be a binary training set, and τ an hyper-parameter for error penalization, the following problem aims at finding coefficients $\alpha_i (\forall i \in [1, \dots, \ell])$ and a bias b such that the decision function $D(x) = sign(f(x) + b)$ with $f(x) = \sum_{i=1}^{\ell} \alpha_i y_i k(x_i, x)$. The equivalent minimization problem is formulated by replacing f by $\tilde{f}$, such that $\langle \tilde{f}, \tilde{f} \rangle = \langle \tilde{f_+}, \tilde{f_+} \rangle_{\mathcal{H}_+} + \langle \tilde{f_-}, \tilde{f_-} \rangle_{\mathcal{H}_-}$ and $\tilde{f} = \tilde{f_+} + \tilde{f_-}$. Then the equivalent minimization problem to eq.2 becomes:

$$\begin{cases} \underset{f \in \mathcal{K}, b \in \mathbf{R}}{\text{stab}} & \frac{1}{2} \langle f, f \rangle_{\mathcal{K}} \\ \text{s.t.} & \sum_{i=1}^{\ell} \max(0, \\ & (1 - y_i(f(x_i) + b))) \leq \tau \end{cases} \qquad \begin{cases} \underset{\tilde{f} \in \mathcal{H}, b \in \mathbf{R}}{\min} & \frac{1}{2} \langle \tilde{f}, \tilde{f} \rangle_{\mathcal{H}} \\ \text{s.t.} & \sum_{i=1}^{\ell} \max(0, \\ & (1 - y_i(\tilde{f}(x_i) + b))) \leq \tau \end{cases} \tag{2}$$

The dual can be calculated as a usual SVM and it lets the modified kernel matrix $\tilde{K} = K_+ + K_-$ appear :

$$\begin{cases} \max\limits_{\tilde{\alpha}} & -\frac{1}{2}\tilde{\alpha}^\top Y(K_+ + K_-)Y\tilde{\alpha} + \tilde{\alpha}^\top \mathbf{1} \\ \text{with} & \tilde{\alpha}^\top y = 0 \\ \text{and} & 0 \leq \tilde{\alpha}_i \leq C \quad \forall i \in [1 \dots \ell] \end{cases} \tag{3}$$

Y is the diagonal matrix of labels y. This problem provides a solution that needs to be projected in the Kreĭn space and all the previous computations are *independent of the choice of the decomposition.*

3 TrIK-SVM

Our new algorithm aims at solving eq.2 in a more efficient way than KSVM. First let's detail the KSVM decomposition [5], which is naturally based on the eigen-decomposition of the kernel matrix. Then we give the new decomposition.

KSVM "flip" Decomposition Let U and D be respectively the eigenvectors and the diagonal matrix containing the eigenvalues ok the kernel matrix K, such that $K = U^\top DU$. The modified semi-definite positive (SDP) kernel associated to K is defined as $\tilde{K} = U|D|U^\top$ where $|U|$ contains the absolute values of U. In this case, $K_+ = U \max(D, 0)U^\top$ and $K_- = -U \min(D, 0)U^\top$.

TrIK-SVM "shift" Decomposition Admitting we know the most negative eigenvalue of the kernel matrix, denoted λ:

$$K = (K - \lambda I) + \lambda I \tag{4}$$

with I the identity matrix. Checking that $K_+ = K - \lambda I$ and $K_- = -\lambda I$ are both SDP matrices is straight forward. Then the modified SDP kernel becomes $\tilde{K} = K - 2\lambda I$.

Transition matrices Transition matrices are used to project either the kernel matrix or the solution back and forth between the RKKS and the RKHS.

$$\begin{array}{l|l} KSVM & TrIK-SVM \\ \quad P_+ = U_+U_+^\top \quad P_- = U_-U_-^\top & \quad P_+ = (K - \lambda I)K^{-1} = I - \lambda K^{-1} \\ \quad P = P_+ - P_- & \quad P_- = \lambda K^{-1} \\ \quad\quad = USU^\top \quad \text{with } S = sign(D) & \quad P = P_+ - P_- = I - 2\lambda K^{-1} \end{array} \tag{5}$$

Algorithm Based on the transition matrix, the algorithm of KSVM is very simple, it consists in projecting the indefinite kernel matrix K into the RKHS ($\tilde{K}$), solve a regular SVM and project the obtained solution $\tilde{\alpha}$ back to the RKKS (α). The most straight forward way to use the proposed decomposition is to apply it directly, similarely to the KSVM algorithm. However doing so suffers from severe numerical unstabilities. Instead we observe that the proposed decomposition can be applied in a rank-one fashion. This features makes it possible to design a solver that can compute the kernel elements on the fly and produce directly a solution in the RKKS.

3.1 TrIK-SVM algorithm

We propose here a dedicated solver that computes at each step both the solution in a RKHS and in the RKKS, so that gradient steps are performed in the minimization setting and optimality conditions are checked in the RKKS. The algorithm is based on a active set method [12]. The algorithm iteratively checks dual or primal constraints and updates the solution after each modification of the support vector set. The dual checking deals with bounds on $\tilde{\alpha}$: in case of a bound violation, the corresponding point is removed from the set of support vectors. The primal checking deals with the good classification constraint: in case of a violation, it means that there remains some misclassified training examples and one is picked to be added to the set of support vectors. Algorithm 3.1 gives the global scheme in which we insert the indefiniteness treatment. Steps **[1.]** and **[3.]** are modified while step **[2.]** will not. In the following, we present in parallel the hard-margin case (HM) and the soft-margin case (SM). Bounded support vectors are subscripted *(bsv)* and free support vectors are subscripted *(sv)*. Matrices $\tilde{G}$ and G are the labeled versions of $\tilde{K}$ and K ($G = YKY$).

Algorithm 1 TrIK-SVM

Require: y, C, K, λ, SV

$converged \leftarrow false$
while $not(converged)$ **do**
  [1.] $[\tilde{\alpha}, \alpha, b] \leftarrow linearSystem(K, C, y, \lambda, SV)$
  [2.] $[\tilde{\alpha}, SV, adm] \leftarrow dualAdmissibility(\tilde{\alpha}, SV)$
  if adm **then**
    [3.] $[SV, opt] \leftarrow primalOptimality(\tilde{\alpha}, b, K, C, y, \lambda, SV)$
    if opt **then**
      $converged \leftarrow true$
    end if
  end if
end while
return α, b

Step [1.] The idea is to compute at each step both $\tilde{\alpha}$ (already done) and α. Originally, this step, restricted to examples that are in the set of current support vectors, consists in computing b (see [12] for details) and $\tilde{\alpha}$. Introducing our kernel decomposition, and noting that only diagonal terms are different between G and $\tilde{G}$, we obtain that $\alpha_{sv} = (I - 2\lambda G^{-1}_{sv,sv})\tilde{\alpha}_{sv}$.

$$\begin{array}{ll}
(HM) & (SM) \\
\tilde{G}_{sv,sv}\tilde{\alpha}_{sv} + by_{sv} = \mathbf{1}_{sv} & \tilde{G}_{sv,sv}\tilde{\alpha}_{sv} + C\tilde{G}_{sv,bsv}\mathbf{1}_{bsv} + by_{sv} = \mathbf{1}_{sv} \\
\tilde{\alpha} = \tilde{G}^{-1}(\mathbf{1}_{sv} - by_{sv}) & \tilde{\alpha} = \tilde{G}^{-1}(\mathbf{1}_{sv} - C\tilde{G}_{sv,bsv} - by_{sv}) \\
G_{sv,sv}P_{sv,sv}\tilde{\alpha}_{sv} + by_{sv} = \mathbf{1}_{sv} & G_{sv,sv}P_{sv,sv}\tilde{\alpha}_{sv} + CG_{sv,bsv}\mathbf{1}_{bsv} + by_{sv} = \mathbf{1}_{sv} \\
\alpha_{sv} = P_{sv,sv}\tilde{\alpha}_{sv} & \alpha_{sv} = P_{sv,sv}\tilde{\alpha}_{sv}
\end{array} \tag{6}$$

Step [2.] This step aims at checking the dual admissibility, ie. checking that all $\tilde{\alpha}_i \geq 0$ and projecting the solution in the admissible set if it is not. Since in the RKKS α is not constrained [5], this step can only be performed in the RKHS.

Step [3.] This last step is performed once the current set of support vector gives an admissible solution in the dual. Its goal is to check the optimality in

the primal by verifying that all non support vectors are well classified, which has to be true in the RKKS.
Using the modified steps in the original algorithm, we can compute values for $\tilde{K}$ and K on the fly, or use a cache strategy for more efficiency. Since both solutions are computed on the SV set at each step, the output α can be sparse.

3.2 Experimental evaluation of TrIK-SVM

KSVM has already proven its interest, so the goal of the presented experiments is to check that TrIK-SVM behaves at least as well as KSVM. We took randomly generated dataset (based on `make_blobs` in scikit-learn) in dimension 5 with 4 centers (2 are assigned to class -1, 2 are assigned to class 1, at random). Resulting problems may be almost separable and others very overlapping.

Sensitivity to λ The first point to check about our decomposition is linked to the fact that we claim that any value under the least eigenvalue of the spectrum will produce a valid decomposition. If it's true on the paper, one might wonder on the numerical effect. Figures 1 illustrate the stability of the solution in the RKKS independently of λ. We run TrIK-SVM on one randomly picked dataset, with different values of λ, starting from the true least eigenvalue up to 20 times this value. We report the α and $\tilde{\alpha}$ values for each λ.

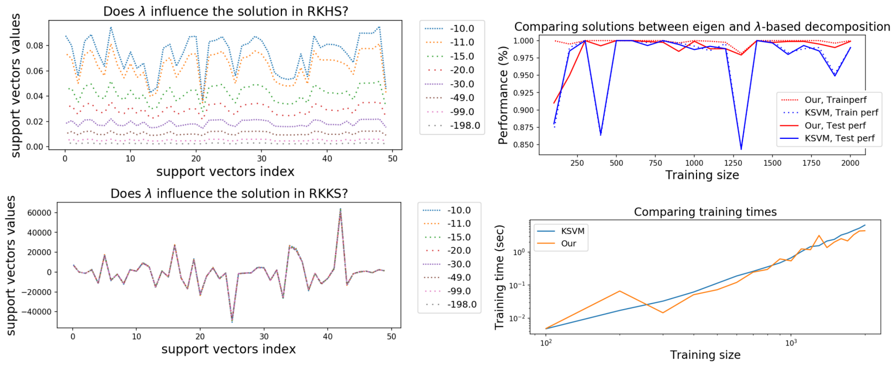

Figure 1: **Left:** solutions in the RKHS (top) and in RKKS (bottom) for decreasing λ values, starting from the least computed eigenvalue. We observe that solutions are similar with a scale factor in RKHS but quasi-identical in the original RKKS. **Right top:** train and test accuracy on 30 toy datasets generated randomly, and of increasing sizes (from 100 to 3000 training examples). On this task, TrIK-SVM performs at least as well as KSVM. **Right bottom:** training time in loglog scale.

TrIKSVM vs KSVM The instabilities due to the eigen-decomposition in KSVM are more easily observed for datasets with many similar training points. In our experiments, it happens more often than for real datasets, since all training point are picked in a restricted space. Figure 1 present an illustrative result of the accuracy performance of TrIK-SVM and KSVM. We observe that TrIK-SVM is competitive over KSVM, which suffers from numerical instabilities. We also plot on figure 1 the corresponding training time for each, on loglog axis. At first sight the advantage of TrIK- SVM is not obvious, but we need to mention that here, KSVM is internally using libsvm which is fully optimized, while the current TrIK-SVM implementation is fully implemented in Python without any trick and pre-computes the complete kernel, so there's space for improvement.

4 Conclusion

This paper introduces a novel algorithm called TrIK-SVM that solves most of the KSVM limitations: it provides a sparse solution, that can be computed without pre-computing the full kernel matrix and thus that can benefit from advanced cache strategies. Since the hyper-parameter λ does not influence the quality of the solution neither the computation time, given that it is sufficiently negative, TrIK-SVM comes without additional parameter to tune compared to a classical SVM. Future work includes application to problems others than binary classification, including Multiple Kernel Setting, and a more efficient implementation of the proposed algorithm.

References

[1] Lin Hsuan-Tien and Lin Chih-Jen. A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan., 2003.

[2] Yihua Chen and Maya R Gupta. Fusing similarities and kernels for classification. In *Information Fusion, 2009. FUSION'09. 12th International Conference on*, pages 474–481. IEEE, 2009.

[3] Gaëlle Loosli. Study on the loss of information caused by the "positivation" of graph kernels for 3d shapes. In *24th European Symposium on Artificial Neural Networks Bruges, Belgium, April 27-28-29*, 2016.

[4] Cheng Soon Ong, Xavier Mary, Stéphane Canu, and Alexander J. Smola. Learning with non-positive kernels. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 81, New York, NY, USA, 2004. ACM.

[5] Gaëlle Loosli, Stéphane Canu, and Cheng Soon Ong. Learning SVM in Kreĭn Spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(6):1204–1216, 2016.

[6] T. Ya. Azizov and I. S. Iokhvidov. *Linear Operators in Spaces with an Indefinite Metric*. Wiley, 1989.

[7] Hai-Ming Xu, Hui Xue, Xiaohong Chen, and Yunyun Wang. Solving Indefinite Kernel Support Vector Machine with Difference of Convex Functions Programming. In *AAAI*, pages 2782–2788, 2017.

[8] Xiaolin Huang, Andreas Maier, Joachim Hornegger, and Johan AK Suykens. Indefinite kernels in least squares support vector machines and principal component analysis. *Applied and Computational Harmonic Analysis*, 43(1):162–172, 2017.

[9] Siamak Mehrkanoon, Xiaolin Huang, and Johan A. K. Suykens. Indefinite kernel spectral learning. *Pattern Recognition*, 78:144 – 153, 2018.

[10] Frank-Michael Schleif and Peter Tino. Indefinite Core Vector Machine. *Pattern Recognition*, 71:187–195, 2017.

[11] Frank-Michael Schleif, Christoph Raab, and Peter Tino. Sparsification of Indefinite Learning Models. In Xiao Bai, Edwin R. Hancock, Tin Kam Ho, Richard C. Wilson, Battista Biggio, and Antonio Robles-Kelly, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, Lecture Notes in Computer Science, pages 173–183. Springer International Publishing, 2018.

[12] S. V. N. Vishwanathan, Alexander J. Smola, and M. Narasimha Murty. SimpleSVM. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 760–767, Washington, DC, USA, 2003. AAAI Press.