OPTIMIZATION OF KERNEL ALIGNMENT BY DATA TRANSLATION IN FEATURE SPACE

Jean-Baptiste Pothin, Cédric Richard

Institut Charles Delaunay (ICD-M2S, FRE CNRS 2848)
Université de Technologie de Troyes, 12 rue Marie Curie, BP 2060, 10010 Troyes cedex - France
jean_baptiste.pothin@utt.fr cedric.richard@utt.fr

ABSTRACT

Kernel-target alignment is commonly used to predict the behavior of reproducing kernels in a classification context, without training any kernel machine. In this paper, we show that a poor position of training data in feature space can drastically reduce the value of alignment. This implies that, in a kernel selection setting, the best kernel of a given collection may be characterized by a low alignment. To overcome this situation, we present a gradient ascent algorithm for maximizing the alignment by data translation in feature space. The aim is to reduce the biais introduced by the translation *non*-invariance of this criterion. Experimental results on multi-dimensional benchmarks show the effectiveness of our approach.

Index Terms— kernel alignment, data translation, SVM

1. INTRODUCTION

Kernel-based methods map a set of data $\boldsymbol{x}$ from the input space $\mathcal{X}$ into some other (possibly infinite) feature space $\mathcal{F}$ via a nonlinear map ϕ, and then apply a linear procedure in this space. Embedding is performed by substituting kernel values for inner products, i.e., $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{F}}$. This provides an elegant way of dealing with nonlinear algorithms by reducing them to linear ones in $\mathcal{F}$. A typical example is Support Vector Machines (SVMs) [1], which map data into a space where the classes of data are more readily separable, and maximize the margin – or distance – between the separating hyperplane and the closest points of each class.

Despite the success of kernel machines, the selection of an appropriate kernel is still critical to achieve good generalization performance. Recently, an interesting solution has been developed through the concept of kernel-target alignment (KTA). The latter measures the degree of agreement between a reproducing kernel and a learning task [2]. Previous works on KTA focused on its optimization by linear combination of kernels in a transductive or inductive settings [3, 4, 5, 6]. More recently, we proposed in [7] a gradient ascent algorithm for maximizing KTA over a linear transform in input space. In all these references, effects of translation in feature space were not studied. While data translation in $\mathcal{F}$ does not affect the resulting SVM in theory, it can greatly modify the value of alignment. In a kernel selection setting, this translation *non*-invariance may lead to an unappropriate kernel. To solve this problem, the standard data centering method should be used [8]. While this method is very simple, it is not optimal in a KTA sense. In [9], Meilà formulated the data centering problem in the form of a criterion J to be maximized by data translation in feature space. The proposed algorithm is based on gradient ascent in feature space. While this approach can be theoretically justified, the author did not take into account the particular form of the solution, namely, a linear combination of the training vectors. In this paper, we explicitly model the translation in the form of a linear combination of training vectors and propose a new algorithm based on gradient ascent in parameter space.

The rest of this paper is organized as follows. In Section 2, kernel-target alignment is introduced. Our gradient ascent algorithm is presented in Section 3. In Section 4, we apply this method in the KTA case, and show its effectiveness through simulations in Section 5. Finally, concluding remarks and suggestions follow.

2. KERNEL-TARGET ALIGNMENT

The alignment criterion is a measure of similarity between two kernels, or between a kernel and a target function [2]. Given a n-sample data set $\mathcal{D}_n$, the alignment of kernels κ_1 and κ_2 is defined as follows

$$\mathcal{A}(\boldsymbol{K}_1, \boldsymbol{K}_2) = \frac{\langle \boldsymbol{K}_1, \boldsymbol{K}_2 \rangle_F}{\sqrt{\langle \boldsymbol{K}_1, \boldsymbol{K}_1 \rangle_F \langle \boldsymbol{K}_2, \boldsymbol{K}_2 \rangle_F}}, \qquad (1)$$

where $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius inner product, and $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ are the Gram matrices with respective entries $\kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\kappa_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$, for all $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_n$.

For binary classification, the decision statistic should satisfy $\phi(\boldsymbol{x}_i) = y_i$, where y_i is the class label of $\boldsymbol{x}_i$. By setting $y_i = \pm 1$, the ideal Gram matrix would be given by

$$\boldsymbol{K}^*(i,j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle = \begin{cases} 1 & \text{if } y_i = y_j \\ -1 & \text{if } y_i \neq y_j. \end{cases} \qquad (2)$$

In [2], Cristianini *et al.* proposed to maximize the alignment, with respect to $\boldsymbol{K}$, between $\boldsymbol{K}$ and target $\boldsymbol{K}^* = \boldsymbol{y}\boldsymbol{y}^t$ in order to determine the most relevant kernel for a classification task.

$$\mathcal{A}(\boldsymbol{K}, \boldsymbol{K}^*) = \frac{\langle \boldsymbol{K}, \boldsymbol{y}\boldsymbol{y}^t \rangle_F}{\sqrt{\langle \boldsymbol{K}, \boldsymbol{K} \rangle_F \langle \boldsymbol{y}\boldsymbol{y}^t, \boldsymbol{y}\boldsymbol{y}^t \rangle_F}} = \frac{\boldsymbol{y}^t \boldsymbol{K} \boldsymbol{y}}{n \left\| \boldsymbol{K} \right\|_F}. \qquad (3)$$

The ease with which KTA can be estimated using only training data makes it an interesting tool for kernel selection. However, a poor position of the data in the feature space can drastically affect this criterion. For instance, suppose the origin is far away from the convex hull of training data. In that case, the elements of the kernel matrix have almost the same value, say z. Using (3), it is easy to show:

$$\mathcal{A}(\boldsymbol{K}) \rightarrow \frac{\sum_{ij} y_j y_i z}{n\sqrt{\sum_{ij} z^2}} = \frac{(n^+ - n^-)^2}{n^2} = \frac{(k-1)^2}{(k+1)^2}, \quad (4)$$

where $n^+(n^-)$ denote the number of data points with $+1(-1)$ labels and $k = n^+/n^-$. From (4), it follows that KTA may depend on the ratio k only. This strongly recommend recentering of data prior to any kernel selection based on KTA.

3. DATA TRANSLATION IN FEATURE SPACE

The aim of this section is to optimize any criterion J depending on the elements of the Gram matrix $\boldsymbol{K}_a$ defined as

$$[\boldsymbol{K}_a]_{i,j=1,\cdots,n} = \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j),$$

where κ_a denotes the inner product of shifted data in feature space, namely,

$$\kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j) \triangleq \langle \phi(\boldsymbol{x}_i) - \boldsymbol{a}, \phi(\boldsymbol{x}_j) - \boldsymbol{a} \rangle \quad (5)$$

with $\boldsymbol{a} \in \mathcal{F}$. In the rest of this paper, we assume that $\boldsymbol{a}$ is a linear combination of training data, that is,

$$\boldsymbol{a} = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i). \quad (6)$$

The standard data centering method (see, e.g, [8]) moves the origin to the center of gravity of the data by setting $\alpha_i = 1/n$ for all i. In a binary classification context, it is suggested in [9] to set $\boldsymbol{a}$ as follows:

$$\alpha_i = \begin{cases} 1/(2n^+) & \text{if } y_i = 1, \\ 1/(2n^-) & \text{if } y_i = -1. \end{cases} \quad (7)$$

By applying the so-called kernel trick on the above definitions for $\kappa_a(\boldsymbol{x}, \boldsymbol{x}')$ and $\boldsymbol{a}$, we obtain

$$\begin{array}{rl} \kappa_a(\boldsymbol{x}, \boldsymbol{x}') = & \kappa(\boldsymbol{x}, \boldsymbol{x}') + \sum_i \sum_j \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ & - \sum_i \alpha_i [\kappa(\boldsymbol{x}_i, \boldsymbol{x}) + \kappa(\boldsymbol{x}_i, \boldsymbol{x}')]. \end{array} \quad (8)$$

Thus, the resulting Gram matrix is given by

$$\boldsymbol{K}_a = \boldsymbol{K} - \boldsymbol{\Gamma}_\alpha^t \boldsymbol{K} - \boldsymbol{K}\boldsymbol{\Gamma}_\alpha + \boldsymbol{\Gamma}_\alpha^t \boldsymbol{K} \boldsymbol{\Gamma}_\alpha, \quad (9)$$

where $\boldsymbol{\Gamma}_\alpha = (\boldsymbol{\alpha}, \cdots, \boldsymbol{\alpha})$ and $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)^t$.

3.1. Data translation by gradient step in α-space

Consider the following centering problem

$$\max_{\boldsymbol{\alpha}} J(\boldsymbol{K}_a). \quad (10)$$

Assuming that the gradient of J with respect to $\boldsymbol{\alpha}$ exists, we have:

$$\nabla_{\boldsymbol{\alpha}} J(\boldsymbol{K}_a) = \sum_{i=1}^{n} \sum_{j=1}^{n} \underbrace{\frac{\partial J}{\partial \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)}}_{g_{ij}} \times \nabla_{\boldsymbol{\alpha}} \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad (11)$$

where

$$\nabla_{\boldsymbol{\alpha}} \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{pmatrix} \partial \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)/\partial \alpha_1 \\ \cdots \\ \partial \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)/\partial \alpha_n \end{pmatrix}. \quad (12)$$

From equation (8), we obtain

$$\frac{\partial \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \alpha_l} = -[\kappa(\boldsymbol{x}_l, \boldsymbol{x}_i) + \kappa(\boldsymbol{x}_l, \boldsymbol{x}_j)] + 2 \sum_{i'} \alpha_{i'} \kappa(\boldsymbol{x}_l, \boldsymbol{x}_{i'}).$$

Since $\kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is symmetric, $g_{ij} = g_{ji}$. This yields

$$\begin{array}{rcl} \sum_i \sum_j g_{ij}\, \kappa(\boldsymbol{x}_l, \boldsymbol{x}_i) = & \sum_i \kappa(\boldsymbol{x}_l, \boldsymbol{x}_i) \sum_j g_{ji} & \\ = & \sum_i \kappa(\boldsymbol{x}_l, \boldsymbol{x}_i)\, \boldsymbol{g}_i^t \boldsymbol{e} & \\ = & \boldsymbol{k}_l^t \boldsymbol{G} \boldsymbol{e}, & \end{array} \quad (13)$$

where $\boldsymbol{e}$ is the column vector of 1's, $\boldsymbol{g}_i = (g_{1i}, \cdots, g_{ni})^t$, $\boldsymbol{G} = (\boldsymbol{g}_1, \cdots, \boldsymbol{g}_n)^t$, $\boldsymbol{k}_l = (\kappa(\boldsymbol{x}_1, \boldsymbol{x}_l), \cdots, \kappa(\boldsymbol{x}_n, \boldsymbol{x}_l)^t$. Similarly, we show that $\sum_i \sum_j g_{ij}\, \kappa(\boldsymbol{x}_l, \boldsymbol{x}_j) = \boldsymbol{k}_l^t \boldsymbol{G} \boldsymbol{e}$. Thus, the gradient vector (11) can be computed as

$$\nabla_{\boldsymbol{\alpha}} J = 2\boldsymbol{K}(\lambda \boldsymbol{\alpha} - \boldsymbol{G}\boldsymbol{e}), \quad (14)$$

where $\lambda = \sum_i \sum_j g_{ij}$. Using the steepest direction $\nabla_{\boldsymbol{\alpha}} J$ with step size parameter η yields

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + 2\eta \boldsymbol{K}(\lambda \boldsymbol{\alpha} - \boldsymbol{G}\boldsymbol{e}) \triangleq \boldsymbol{\alpha} + \boldsymbol{\Delta}\boldsymbol{\alpha}. \quad (15)$$

With some abuse of notations, we denote by $\boldsymbol{K}_{a+\Delta\alpha}$ the updated Gram matrix associated with the parameter vector (15). From (9), it follows:

$$\begin{array}{l} \boldsymbol{K}_{a+\boldsymbol{\Delta}\boldsymbol{\alpha}} \\ = \boldsymbol{K} - \boldsymbol{\Gamma}_{\alpha+\Delta\alpha}^t \boldsymbol{K} - \boldsymbol{K}\boldsymbol{\Gamma}_{\alpha+\Delta\alpha} + \boldsymbol{\Gamma}_{\alpha+\Delta\alpha}^t \boldsymbol{K} \boldsymbol{\Gamma}_{\alpha+\Delta\alpha}, \\ = \boldsymbol{K}_a - \boldsymbol{\Gamma}_{\Delta\alpha}^t \boldsymbol{K} - \boldsymbol{K}\boldsymbol{\Gamma}_{\Delta\alpha} + \boldsymbol{\Gamma}_{\Delta\alpha}^t \boldsymbol{K} \boldsymbol{\Gamma}_{\Delta\alpha} + 2\boldsymbol{\Gamma}_\alpha^t \boldsymbol{K} \boldsymbol{\Gamma}_{\Delta\alpha}. \end{array}$$

Hence, the recursive update for $\boldsymbol{K}_a$ can be written as:

$$\boldsymbol{K}_a \leftarrow \boldsymbol{K}_a - \boldsymbol{v}\boldsymbol{e}^t - \boldsymbol{e}\boldsymbol{v}^t + (2\boldsymbol{\alpha} + \boldsymbol{\Delta}\boldsymbol{\alpha})^t \boldsymbol{v} \times \boldsymbol{E}, \quad (16)$$

where $\boldsymbol{v} = \boldsymbol{K}\boldsymbol{\Delta}\boldsymbol{\alpha}$ and $\boldsymbol{E} = \boldsymbol{e}\boldsymbol{e}^t$. Finally, our gradient ascent algorithm can be summarized as follow:

1. Choose the initial solution $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}_0$;
2. Compute the Gram matrix $\boldsymbol{K}$;
3. Compute the matrix $[\boldsymbol{G}]_{ij} = \partial J/\partial \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)$;
4. Update $\boldsymbol{K}_a$ with (16), and $\boldsymbol{\alpha}$ with (15);
5. Go to step 3 until convergence.

3.2. Data translation by gradient step in $\mathcal{F}$-space

In [9], Meilà considered the general problem $\max_{\boldsymbol{a}\in\mathcal{F}} J(\boldsymbol{K}_a)$ and suggested to solve it by gradient ascent. The proposed updating rule was

$$\boldsymbol{a} \leftarrow \boldsymbol{a} + \eta \nabla_{\boldsymbol{a}} J(\boldsymbol{K}_a), \tag{17}$$

where the step $\boldsymbol{\delta}_a = \eta \nabla_{\boldsymbol{a}} J(\boldsymbol{K}_a)$ was shown to be

$$\boldsymbol{\delta}_a = \gamma \boldsymbol{a} + \sum_{i=1}^{n} \gamma_i \phi(\boldsymbol{x}_i) \tag{18}$$

with $\gamma = -\sum_i \gamma_i$ and $\gamma_i = -2\eta \sum_{j=1}^{n} \partial J / \partial \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Although all the coefficients γ above can be computed using only kernel evaluation, it is clear that, in the general case, the computation of (17) requires to know the coordinates of $\boldsymbol{a}$ in feature space $\mathcal{F}$. Observing that $\boldsymbol{\delta}_a$ is expressed as a linear combination of $\boldsymbol{a}$ and the training vectors, Meilà suggested to choose $\boldsymbol{a}_0 = \boldsymbol{0}$ as the initial condition in order to stay in the span of the training vectors at each step. Now we study the connection of this approach with our own algorithm.

Let $\boldsymbol{a}$ be the linear combination (6). Note that initializing $\boldsymbol{a} = \boldsymbol{a}_0 = \boldsymbol{0}$ can be done easily with $\alpha_i = 0$ for all i. Combining (17) and (18), we find

$$\begin{array}{ll} \boldsymbol{a} + \eta \nabla_{\boldsymbol{a}} J(\boldsymbol{K}_a) & = (1+\gamma)\boldsymbol{a} + \sum_{i=1}^{n} \gamma_i \phi(\boldsymbol{x}_i), \\ & = \sum_{i=1}^{n} [\alpha_i + \gamma \alpha_i + \gamma_i] \phi(\boldsymbol{x}_i). \end{array} \tag{19}$$

It immediately follows that rule (17) does not need to be computed explicitly. More formally, since $\gamma_i = -2\eta \boldsymbol{g}_i^t \boldsymbol{e}$ and $\gamma = 2\eta\lambda$, one can update $\boldsymbol{\alpha}$ as follows

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + 2\eta(\lambda \boldsymbol{\alpha} - \boldsymbol{G}\boldsymbol{e}). \tag{20}$$

Comparing (15) and (20), we see that our algorithm requires the Gram matrix $\boldsymbol{K}$ at each step whereas Meila's algorithm does not. Intuitively, this matrix can be viewed as a coordinate matrix projecting the step $\boldsymbol{\delta}_a$ onto the span of the training set. As shown in Section 5, this improves the convergence rate of the method.

4. OPTIMIZATION OF KTA BY DATA TRANSLATION

We shall now consider data centering problem with criterion $J(\boldsymbol{K}_a) = \mathcal{A}(\boldsymbol{K}_a)$ and the algorithm presented in Section 3. We first need to compute the terms $g_{ij} = \partial \mathcal{A} / \partial \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)$. We directly obtain

$$\begin{aligned} g_{ij} &= \frac{1}{n} \times \frac{\partial \Big(\langle \boldsymbol{K}_a, \boldsymbol{y}\boldsymbol{y}^t \rangle_F / \|\boldsymbol{K}_a\|_F \Big)}{\partial \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)} \\ &= \frac{y_i y_j \|\boldsymbol{K}_a\|_F - \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j) \|\boldsymbol{K}_a\|_F^{-1} \langle \boldsymbol{K}_a, \boldsymbol{y}\boldsymbol{y}^t \rangle_F}{n \|\boldsymbol{K}_a\|_F^2} \\ &= \frac{y_i y_j}{n\|\boldsymbol{K}_a\|_F} - \frac{\mathcal{A}(\boldsymbol{K}_a)}{\|\boldsymbol{K}_a\|_F^2} \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j). \end{aligned} \tag{21}$$

From the above definitions for $\boldsymbol{g}_i$ and $\boldsymbol{G}$, we have

$$\boldsymbol{g}_i^t \boldsymbol{e} = \frac{y_i (n^+ - n^-)}{n\|\boldsymbol{K}_a\|_F} - \frac{\mathcal{A}(\boldsymbol{K}_a)}{\|\boldsymbol{K}_a\|_F^2} \boldsymbol{k}_{a_i}^t \boldsymbol{e}, \tag{22}$$

and

$$\boldsymbol{G}\boldsymbol{e} = \frac{n^+ - n^-}{n\|\boldsymbol{K}_a\|_F} \boldsymbol{y} - \frac{\mathcal{A}(\boldsymbol{K}_a)}{\|\boldsymbol{K}_a\|_F^2} \boldsymbol{K}_a \boldsymbol{e}. \tag{23}$$

Therefore, rule (15) can be expressed as:

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + 2\eta \boldsymbol{K} \left(\lambda \boldsymbol{\alpha} - \frac{n^+ - n^-}{n\|\boldsymbol{K}_a\|_F} \boldsymbol{y} + \frac{\mathcal{A}(\boldsymbol{K}_a)}{\|\boldsymbol{K}_a\|_F^2} \boldsymbol{K}_a \boldsymbol{e} \right), \tag{24}$$

where

$$\lambda = \frac{(n^+ - n^-)^2}{n\|\boldsymbol{K}_a\|_F} - \frac{\mathcal{A}(\boldsymbol{K}_a)}{\|\boldsymbol{K}_a\|_F^2} \sum_i \sum_j \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{25}$$

5. EXPERIMENTS

To validate our algorithm, we used the Waveform benchmark. This set, available at http://www.ics.uci.edu/~mlearn/, contains 400 training samples of 21 variables each, and 4600 validation samples.

In the first experiment, we considered polynomial kernels $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + 1)^d$ with $d = 1, \cdots, 4$ and Radial Basis Functions (RBF) $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma^2)$, with $\sigma \in \{.1, .5, 1, 5\}$. For each kernel, we trained a l_1-SVM with the best $C \in \{1, 5, 10, 50, 100, 500, 1000, 5000\}$ found by hold-out testing. Table 1 reports the generalization error for each SVM. We see that the minimum 10.37% was reached with the Gaussian kernel parameterized by $\sigma = 5$. Next, we considered KTA and data centering. Both Meila's and our algorithm stopped when the improvement of the alignment was less than 10^{-6}. The step size η in (15) and (20) was obtained with trail-and-error search for each kernel and algorithm. Table 2 reports the alignment for each candidate and different centering methods. First note that moving the origin to the center of gravity in feature space degraded the value of the alignment, except for $\sigma = 5$. In comparaison, setting the origin halfway between the centers of gravity of the two classes was found to be a better heuristic. We also observe that Meilà's and our algorithm were very similar. They both gave a great improvement in the KTA score, in particular for the Gaussian kernel with $\sigma = 1$. From Table 3, note that the alignment measured on the validation set also greatly increased after data centering. Using KTA for kernel selection without data centering, we would have selected the polynomial kernel with degree 1, see Table 2. As shown in Table 1, we would have obtained a generalization error of 13.61%, which is not the best performance. The latter was reached by centering data and applying KTA. Figure 1 shows the evolution of alignment for RBF kernel with $\sigma = 1$. We note that

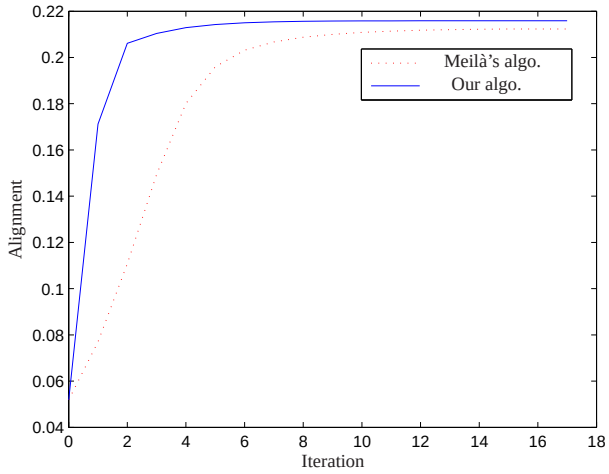

Fig. 1. Evolution of the alignment for $\sigma = 1$ (RBF kernel).

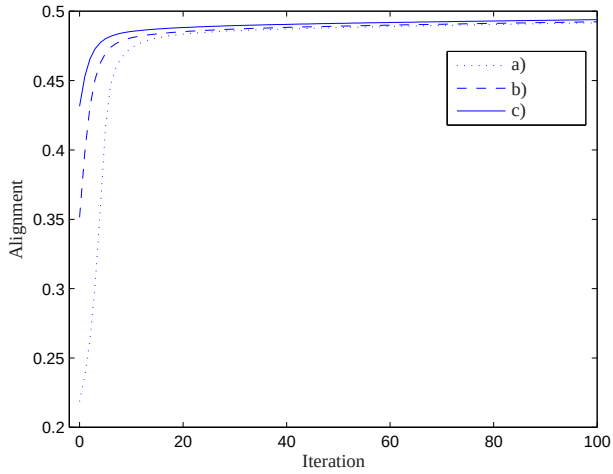

Fig. 2. Evolution of the alignment for different initial solution, a) null vector, b) $\alpha_i = 1/n$ for all i, c) see eqn. (7)

having explicitly define the gradient ascent in $\boldsymbol{\alpha}$-space improved the convergence rate of the general centering method of Meilà. Similar results were founded with all the other kernels. In Figure 2, we show the effect of the initial condition $\boldsymbol{\alpha}_0$ for RBF kernel with $\sigma = 5$. As noted previously, starting from (7) seems to be a good heuristic since the convergence rate is the best in that case.

6. CONCLUSION

In this paper, we considered data centering in feature space. We defined the translation as a linear combination of training vectors and proposed to optimize a given criterion by a gradient ascent in the parameter space. We used this principle to improve the kernel-target alignment (KTA) criterion. Experimental results with a multi-dimensional benchmark showed the effectiveness of our approach for this criterion. Further works includes the optimization in a data-dependent way of the step of the gradient ascent. We also plan to study the problem consisting in optimizing a linear combination of centered kernels.

7. REFERENCES

[1] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY: Springer, 1995.

[2] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola, "On kernel-target alignment," *Advances in Neural Information Processing Systems*, vol. 14, pp. 367–373, 2002.

[3] J. Kandola, J. Shawe-Taylor, and N. Cristianini, "On the extensions of kernel alignment," Department of Computer Science, University of London, Tech. Rep. 120, 2002.

[4] ——, "Optimizing kernel alignment over combinations of kernels," Department of Computer Science, University of London, Tech. Rep. 121, 2002.

[5] G. R. G. Lanckriet, N. Cristianini, P. L. Bartlett, L. El Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semi-definite programming," *Proc. of the nineteenth International Conference on Machine Learning*, pp. 323–330, 2002.

[6] J.-B. Pothin and C. Richard, "A greedy algorithm for optimizing the kernel alignment and the performance of kernel machines." *EUSIPCO'06*, 4-8 September 2006.

[7] ——, "Optimal feature representation for kernel machines using the kernel-target alignment criterion." *Accepted in ICASSP'07*, 15-20 avril 2007.

[8] N. Cristianini, "Support vector and kernel machines," *Tutorial at ICML*, 2001.

[9] M. Meilà, "Data centering in feature space," in *Technical report*, University of Washington, 2003.

[10] H. Xiong and M. O. Ahmad, "Optimizing the kernel in the empirical feature space," *IEEE Transactions on Neural Networks*, vol. 16, no. 2, pp. 460–474, March 2005.

	$d=1$	$d=2$	$d=3$	$d=4$	$\sigma=.1$	$\sigma=.2$	$\sigma=.5$	$\sigma=1$	$\sigma=2$	$\sigma=5$	$\sigma=10$
error (%)	13.61	14.33	12.11	18.07	14.72	15.41	15.09	14.59	10.91	**10.37**	10.52

Table 1. Generalization performance for l_1-SVM based on the polynomial kernel of degree d or RBF with hyperparameter σ.

Centering method	$d=1$	$d=2$	$d=3$	$d=4$	$\sigma=.1$	$\sigma=.2$	$\sigma=.5$	$\sigma=1$	$\sigma=2$	$\sigma=5$	$\sigma=10$
none	.385	.112	.239	.089	.050	.050	.050	.052	.221	.219	.148
#1	.354	.079	.204	.059	.044	.044	.044	.046	.204	.351	.356
#2	.433	.100	.255	.075	.057	.056	.057	.059	.256	.431	.436
Meila's algorithm	.487	.249	.346	.214	.212	.213	.213	.215	.387	.498	.493
our's algorithm	.482	.250	.346	.215	.213	.213	.213	.216	.388	.499	.494

Table 2. Alignment for the training set and the polynomial kernels of degree d or the RBF with hyperparameter σ. The origin in feature space was not changed (*none*), moved to the center of gravity of the data (#1), moved halfway between the centers of gravity of the two classes (#2), optimized using Meilà's or our algorithm.

Centering method	$d=1$	$d=2$	$d=3$	$d=4$	$\sigma=.1$	$\sigma=.2$	$\sigma=.5$	$\sigma=1$	$\sigma=2$	$\sigma=5$	$\sigma=10$
none	.316	.104	.175	.068	.015	.015	.147	.021	.240	.200	.142
#1	.283	.060	.136	.035	.034	.034	.034	.039	.229	.298	.289
#2	.365	.078	.185	.049	.036	.036	.036	.043	.288	.379	.372
Meilà's algorithm	.439	.223	.291	.179	.117	.117	.117	.120	.375	.458	.446
our algorithm	.440	.225	.290	.178	.117	.117	.117	.120	.374	.459	.446

Table 3. Alignment for the validation set.