OPTIMIZING 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 any given reproducing kernel in a classification context, without training any kernel machine. However, a poor position of the data in feature space can drastically reduce the value of the alignment. This implies that, in a kernel selection setting, the best kernel in a given collection may be associated with a low value of alignment. In this paper, we present a new 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 $\mathcal{F}$. The embedding is performed by substituting kernel values for the 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 (SVM) [1], which map data into a space where the classes of data are more readily separable, and maximize the margin – or distance – between a separating hyperplane and the closest points of each class.

Despite the success of kernel machines, the selection of an appropriate kernel is still critical for achieving 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 mainly 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, the effect of translation in feature space was not studied. While data translation in $\mathcal{F}$ does not, in theory, affect the resulting SVM, it can greatly modify the value of alignment. In a kernel selection setting, this translation *non*-invariance may lead to bad selection of 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. It was shown that the solution of this problem can be expressed as a linear combination of training vectors as soon as the initial solution is in the span of the training set. However, in our best knowledge, no experimental results were reported to show the effectiveness of this approach. 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 rather than in feature space. We also show the effectiveness of the two approaches for KTA.

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 the 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 $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_n$.

In [2], Cristianini *et al.* proposed to maximize the alignment with respect to $\boldsymbol{K}$ for the target $\boldsymbol{y}\boldsymbol{y}^\top$ in order to determine the most relevant kernel for a given classification task with target $y_i \in \{\pm 1\}$. In that case, the alignment can be written as:

$$\mathcal{A}(\boldsymbol{K}) = \frac{\langle \boldsymbol{K}, \boldsymbol{y}\boldsymbol{y}^\top \rangle_F}{\sqrt{\langle \boldsymbol{K}, \boldsymbol{K} \rangle_F \langle \boldsymbol{y}\boldsymbol{y}^\top, \boldsymbol{y}\boldsymbol{y}^\top \rangle_F}} = \frac{\boldsymbol{y}^\top \boldsymbol{K} \boldsymbol{y}}{n \left\| \boldsymbol{K} \right\|_F}. \qquad (2)$$

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. To see that, suppose the origin is far away from the convex hull of the data. Then, the elements of the kernel matrix $\boldsymbol{K}$ have all about the same value, say z. Using (2), it is easy to show:

$$\mathcal{A}(\boldsymbol{K}) \rightarrow \frac{\sum_{ij} y_j y_i z}{n\sqrt{\sum_{ij} z^2}} = \frac{(n^+ - n^-)^2}{n^2} = \frac{(\rho-1)^2}{(\rho+1)^2}, \quad (3)$$

where n^+ (resp. n^-) denotes the number of data points with $+1$ (resp. -1) labels, and $\rho = n^+/n^-$. From (3), it follows that KTA may depend on the ratio ρ only. In particular, for $\rho = 1$, one can found the worst alignment $\mathcal{A}(\boldsymbol{K}) = 0$ even if the data can be perfectly separated by a SVM. This strongly caution us the recentering of KTA before any kernel selection procedure based on this criterion.

3. DATA CENTERING IN FEATURE SPACE

In this Section, the goal is to optimize any criterion J which is a smooth function of elements of the Gram matrix $\boldsymbol{K}_a$ defined by:

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

where κ_a denotes the kernel representing the scalar product with the origin shifted in $\boldsymbol{a} \in \mathcal{F}$, i.e.:

$$\kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j) \triangleq \langle \phi(\boldsymbol{x}_i) - \boldsymbol{a}, \phi(\boldsymbol{x}_j) - \boldsymbol{a} \rangle. \quad (4)$$

In the rest of this paper, we assume that $\boldsymbol{a}$ is a linear combination of the training set, hence:

$$\boldsymbol{a} = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i). \quad (5)$$

The standard data centering method (e.g [8]) moves the origin at the center of gravity of the data by setting $\alpha_i = 1/n$ for all i. For a binary classification task, [9] suggested to set $\boldsymbol{a}$ between the centers of gravity of the two classes in feature space. Formally, this is performed for

$$\alpha_i = \begin{cases} 1/(2n^+) & \text{if } y_i = 1, \\ 1/(2n^-) & \text{if } y_i = -1. \end{cases} \quad (6)$$

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

$$\begin{array}{rl} \kappa_a(\boldsymbol{x}, \boldsymbol{x}') = & \kappa(\boldsymbol{x}, \boldsymbol{x}') + \sum_i \sum_j \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ & - \sum_i \alpha_i [\kappa(\boldsymbol{x}_i, \boldsymbol{x}) + \kappa(\boldsymbol{x}_i, \boldsymbol{x}')]. \end{array} \quad (7)$$

Thus, the "centered" Gram matrix is given by

$$\boldsymbol{K}_a = \boldsymbol{K} - \boldsymbol{\Gamma}_\alpha^\top \boldsymbol{K} - \boldsymbol{K}\boldsymbol{\Gamma}_\alpha + \boldsymbol{\Gamma}_\alpha^\top \boldsymbol{K} \boldsymbol{\Gamma}_\alpha, \quad (8)$$

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

3.1. Data centering by gradient step in α-space

Let the centering criterion be

$$\max_{\boldsymbol{\alpha}} J(\boldsymbol{K}_a). \quad (9)$$

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

$$\nabla_{\boldsymbol{\alpha}} J(\boldsymbol{K}_a) = \sum_{i=1}^{n} \sum_{j=1}^{n} \underbrace{\frac{\partial J}{\partial \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)}}_{g_{ij}} \times \nabla_{\boldsymbol{\alpha}} \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad (10)$$

where

$$\nabla_{\boldsymbol{\alpha}} \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{pmatrix} \partial \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)/\partial \alpha_1 \\ \cdots \\ \partial \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)/\partial \alpha_n \end{pmatrix}. \quad (11)$$

From equation (7)

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

Since $\kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is symmetric, $g_{ij} = g_{ji}$, and therefore:

$$\begin{array}{rcl} \sum_i \sum_j g_{ij}\, \kappa(\boldsymbol{x}_l, \boldsymbol{x}_i) = & \sum_i \kappa(\boldsymbol{x}_l, \boldsymbol{x}_i) \sum_j g_{ji} & \\ = & \sum_i \kappa(\boldsymbol{x}_l, \boldsymbol{x}_i)\, \boldsymbol{g}_i^\top \boldsymbol{e} & \\ = & \boldsymbol{k}_l^\top \boldsymbol{G} \boldsymbol{e}, & \end{array} \quad (12)$$

where $\boldsymbol{e}$ is the column vector of 1's, $\boldsymbol{g}_i = (g_{1i}, \cdots, g_{ni})^\top$, $\boldsymbol{G} = (\boldsymbol{g}_1, \cdots, \boldsymbol{g}_n)^\top$, $\boldsymbol{k}_l = (\kappa(\boldsymbol{x}_1, \boldsymbol{x}_l), \cdots, \kappa(\boldsymbol{x}_n, \boldsymbol{x}_l))^\top$. Similarly, one can show $\sum_i \sum_j g_{ij}\, \kappa(\boldsymbol{x}_l, \boldsymbol{x}_j) = \boldsymbol{k}_l^\top \boldsymbol{G} \boldsymbol{e}$. Thus, the gradient vector (10) can be computed as:

$$\nabla_{\boldsymbol{\alpha}} J = 2\boldsymbol{K}(\lambda \boldsymbol{\alpha} - \boldsymbol{G}\boldsymbol{e}), \quad (13)$$

where $\lambda = \sum_i \sum_j g_{ij}$. Taking a step in the direction $\nabla_{\boldsymbol{\alpha}} J$ with step size η means:

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + 2\eta \boldsymbol{K}(\lambda \boldsymbol{\alpha} - \boldsymbol{G}\boldsymbol{e}) \triangleq \boldsymbol{\alpha} + \boldsymbol{\Delta\alpha}. \quad (14)$$

With abuse of notations, denote by $\boldsymbol{K}_{a+\Delta\alpha}$ the updated Gram matrix associated with the parameter vector (14). From (8), it follows:

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

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

$$\boldsymbol{K}_a \leftarrow \boldsymbol{K}_a - \boldsymbol{v}\boldsymbol{e}^\top - \boldsymbol{e}\boldsymbol{v}^\top + (2\boldsymbol{\alpha} + \boldsymbol{\Delta\alpha})^\top \boldsymbol{v} \times \boldsymbol{E}, \quad (15)$$

where $\boldsymbol{v} = \boldsymbol{K}\boldsymbol{\Delta\alpha}$ and $\boldsymbol{E} = \boldsymbol{e}\boldsymbol{e}^\top$. Finally, our gradient ascent algorithm can be summarize 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 $\boldsymbol{K}_a$ by (15) and $\boldsymbol{\alpha}$ by (14);
5. Go to step 3 until convergence.

3.2. Data centering 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 update rule was:

$$\boldsymbol{a} \leftarrow \boldsymbol{a} + \eta \nabla_{\boldsymbol{a}} J(\boldsymbol{K}_a), \tag{16}$$

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

$$\boldsymbol{\delta}_a = \gamma \boldsymbol{a} + \sum_{i=1}^{n} \gamma_i \phi(\boldsymbol{x}_i) \tag{17}$$

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 (16) requires to know $\boldsymbol{a}$ by its coordinates in $\mathcal{F}$. Noting 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 solution in order to stay in the span of the training vectors at each step. We now show links between this approach and our own algorithm.

Let $\boldsymbol{a}$ be the linear combination (5). Note that initializing $\boldsymbol{a} = \boldsymbol{a}_0 = \boldsymbol{0}$ can be done easily by setting $\alpha_i = 0$ for all i. Combining (16) and (17), we found

$$\begin{array}{ll} \boldsymbol{a} + \eta \nabla_{\boldsymbol{a}} J(\boldsymbol{K}_a) & = (1+\gamma)\boldsymbol{a} + \sum_{i=1}^{n} \gamma_i \phi(\boldsymbol{x}_i), \\ & = \sum_{i=1}^{n} [\alpha_i + \gamma \alpha_i + \gamma_i] \phi(\boldsymbol{x}_i). \end{array} \tag{18}$$

It immediately follows that the update rule (16) does not need to be computed explicitly. More formally, since $\gamma_i = -2\eta \boldsymbol{g}_i^\top \boldsymbol{e}$ and $\gamma = 2\eta\lambda$, one can update $\boldsymbol{\alpha}$ by

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + 2\eta(\lambda \boldsymbol{\alpha} - \boldsymbol{G}\boldsymbol{e}). \tag{19}$$

Comparing (14) and (19), one can see that our algorithm requires the Gram matrix $\boldsymbol{K}$ at each step while Meila's algorithm 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 CENTERING

In this Section, we consider data centering with the criterion $J(\boldsymbol{K}_a) = \mathcal{A}(\boldsymbol{K}_a)$ and the algorithm presented in Section 3. For this purpose, we need to compute the terms $g_{ij} = \partial \mathcal{A} / \partial \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Using the standard derivative for a quotient, we get:

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

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

$$\boldsymbol{g}_i^\top \boldsymbol{e} = \frac{y_i (n^+ - n^-)}{n\|\boldsymbol{K}_a\|_F} - \frac{\mathcal{A}(\boldsymbol{K}_a)}{\|\boldsymbol{K}_a\|_F^2} \boldsymbol{k}_{a_i}^\top \boldsymbol{e}, \tag{20}$$

and

$$\boldsymbol{G}\boldsymbol{e} = \frac{n^+ - n^-}{n\|\boldsymbol{K}_a\|_F} \boldsymbol{y} - \frac{\mathcal{A}(\boldsymbol{K}_a)}{\|\boldsymbol{K}_a\|_F^2} \boldsymbol{K}_a \boldsymbol{e}. \tag{21}$$

Therefore, the update rule (14) for $\boldsymbol{\alpha}$ can be expressed as:

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + 2\eta \boldsymbol{K} \left(\lambda \boldsymbol{\alpha} - \frac{n^+ - n^-}{n\|\boldsymbol{K}_a\|_F} \boldsymbol{y} + \frac{\mathcal{A}(\boldsymbol{K}_a)}{\|\boldsymbol{K}_a\|_F^2} \boldsymbol{K}_a \boldsymbol{e} \right), \tag{22}$$

where

$$\lambda = \frac{(n^+ - n^-)^2}{n\|\boldsymbol{K}_a\|_F} - \frac{\mathcal{A}(\boldsymbol{K}_a)}{\|\boldsymbol{K}_a\|_F^2} \sum_i \sum_j \kappa_a(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{23}$$

5. EXPERIMENTS

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

In our experiments, we considered Radial Basis Functions (RBF) $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma^2)$, with $\sigma \in \{.1, .2, .5, 1, 2, 5\}$. 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 can see that the minimum 10.37% was achieved for 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 η in eqn. (14) and (19) was optimized empirically for each kernel in both algorithms. Table 2 reports the alignment for each candidate and different centering methods. Note first that for this real data, moving the origin at 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 remark 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 $\sigma = 1$. Note that from Table 3, the alignment measured on the validation set is also greatly improved after data centering. Suppose now we used KTA

σ	.1	.2	.5	1	2	5
error	14.72	15.41	15.09	14.59	10.91	**10.37**

Table 1. Generalization performance (in %) for a l_1-SVM with RBF and hyperparameter σ.

Centering \ σ	.1	.2	.5	1	2	5
none	.050	.050	.050	.052	.221	.219
#1	.044	.044	.044	.046	.204	.351
#2	.057	.056	.057	.059	.256	.431
Meila's algo.	.212	.213	.213	.215	.387	.498
our's algo.	.213	.213	.213	.216	.388	.499

Table 2. Alignment for the training set and 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 \ σ	.1	.2	.5	1	2	5
none	.015	.015	.147	.021	.240	.200
#1	.034	.034	.034	.039	.229	.298
#2	.036	.036	.036	.043	.288	.379
Meilà's algo.	.117	.117	.117	.120	.375	.458
our algo.	.117	.117	.117	.120	.374	.459

Table 3. Alignment for the validation set. See Table 2 for a brief description of the different centering methods compared.

without centering for kernel selection. In that case, the best alignment is given by the polynomial kernel of degree 1, see Table 2. From Table 1, we would obtain a generalization error of 13.61%, which is not the minimum. However, if we now recenter the data with our algorithm for instance, then the best model is found by KTA. Figure 1 shows the evolution of the alignment for the RBF kernel with $\sigma = 1$. We note that having explicitly define the gradient ascent in $\boldsymbol{\alpha}$-space has improved the convergence rate comparing to the general centering method of Meilà. Similar results were founded for other kernels, but more analysis are needed to explain this behavior. In Figure 2, we show the effect of the initial solution $\boldsymbol{\alpha}_0$ for the RBF kernel with $\sigma = 5$. Despite the local nature of the search, the algorithm gave the maximum alignment 0.499 for the simple solutions considered. As noted previously, starting from (6) seems to be a good heuristic in practice.

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 the method 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 of the step of the gradient ascent in a data-dependent way. We also plan to study the problem consisting in optimizing a linear combination of centered kernels.

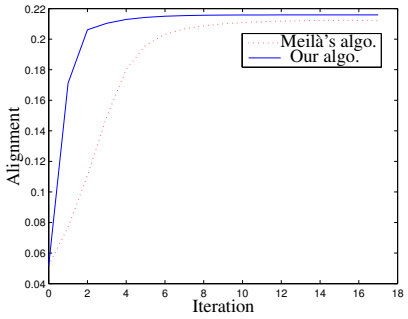

Fig. 1. Evolution of the alignment for $\sigma = 1$ (RBF kernel).

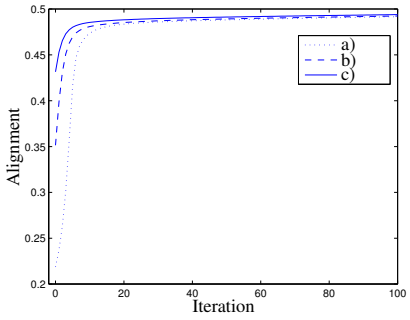

Fig. 2. Evolution of the alignment for different initial solution, a) null vector, b) $\alpha_i = 1/n$ for all i, c) eqn. (6).

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 kernel-target alignment criterion," *ICASSP'07*, vol. 3, pp. 1065–1068, 2007 - *(Best Student Paper Finalist)*.

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

[9] M. Meilà, "Data centering in feature space," University of Washington, Tech. Rep., 2003.