Kernel Parameter Selection by Gap Maximization between Intra and Inter-Class Samples

John Yang, Hyeogjin Lee and Nojun Kwak
Graduate School of Convergence Science and Technology
Seoul National University
Seoul, Korea
Email: {yjohn, hjinlee, nojunk}@snu.ac.kr

***Abstract*—By maximizing the gap between classes in the reproducing kernel Hilbert space (RKHS), our method optimizes for the sigma values of radial basis function (RBF) or gaussian kernels. For each sample, we try to ensure the distance gap between intra-class and inter-class in RKHS to be large. Unlike previous methods of multiple kernel learning, our method does not need large amount of computations, which allows us to apply the proposed method to a larger set of data. Our method is compared with the method of kernel target alignment which is one of the most popular methods in multiple kernel learning to prove its efficiency of finding the optimal kernel parameter for the Face vs Non-face dataset.**

***Index Terms*—Multiple kernel learning, Kernel parameter selection, Gap maximization, Kernel methods.**

I. Introduction

Usually, for the data that is impossible to be linearly discriminated in the input space, kernel trick allows linear segregation in a more complex space. Selecting the kernel function and its parameters is an important issue in training process, but often requires heuristic trials to seek an opt kernel function and to tune the parameters for a given data.

A. Multiple Kernel Learning

As in Gönen and Alpaydın [1] [2], recent developments have suggested to use multiple kernels instead of selecting one specific kernel function and its corresponding parameters. One can approach this problem by considering convex combinations of p kernels, as in

$$k(x_i, x_j) = \sum_{m=1}^{p} \mu_m k^{(m)}(x_i, x_j), \tag{1}$$

with $\boldsymbol{\mu} \triangleq [\mu_1, \cdots, \mu_p] \geq 0$ (element-wise nonnegative) and $\sum_{m=1}^{p} \mu_m = 1$ for p different kernel $k^{(m)}$ functions and $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ is n input data. Learning the kernel parameters, then, consists of learning the weighting coefficients for each base kernel, rather than optimizing the kernel parameters of a single kernel. The optimized solution can then be used to understand which features of the examples are of importance for discrimination. An accurate classification can be obtained by a sparse weighting μ_m, and pragmatic experiments and deeper analysis can be studied more efficiently. One of the weaknesses of MKL is that not only the size of kernel matrices increase as the training instances increase, but also the hardware memory size must afford the complicated optimizing calculations. When an optimal training for a large data set is required, kernel method is occasionally not taken in the training process because of the large amount of calculations that correspond.

B. Kernel Target Alignment

The idea of Kernel alignment was firstly introduced by Cristianini et al. [3] with which our method is closely related. Their framework was introduced from maximizing the alignment between a target-kernel and a combination of multiple kernels. In their work, the similarity between two given kernels $k_1(\mathbf{x}_i, \mathbf{x}_j)$ and $k_2(\mathbf{x}_i, \mathbf{x}_j)$ is defined as:

$$A(\mathbf{K}_1, \mathbf{K}_2) = \frac{\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F}{\sqrt{\langle \mathbf{K}_1, \mathbf{K}_1 \rangle_F \langle \mathbf{K}_2, \mathbf{K}_2 \rangle_F}}, \tag{2}$$

where $\mathbf{K}_1$ and $\mathbf{K}_2$ are kernel matrices for the kernels, $k_1(\mathbf{x}_i, \mathbf{x}_j)$ and $k_2(\mathbf{x}_i, \mathbf{x}_j)$, respectively, Here, $\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F \triangleq \mathrm{tr}(\mathbf{K}_1^T \mathbf{K}_2)$ is the Frobenius inner product with $\mathrm{tr}(\cdot)$ being the trace of matrix. For binary classification problems, let the class label be $y_i \in \{1, -1\}$ for each training data $\mathbf{x}_i$ and the target matrix can be set as $\mathbf{K}_t = \mathbf{y}\mathbf{y}^T$. Given an sample $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, Frobenius inner product between Gram matrices $\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F$. The similarity alignment between the kernel matrix $\mathbf{K}$ and the target matrix is defined as

$$A(\mathbf{K}, \mathbf{K}_t) = \frac{\langle \mathbf{K}, \mathbf{K}_t \rangle_F}{\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F \langle \mathbf{K}_t, \mathbf{K}_t \rangle_F}} = \frac{\langle \mathbf{K}, \mathbf{y}\mathbf{y}^T \rangle_F}{n\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F}} \tag{3}$$

and the method of kernel alignment tries to maximize this similarity.

Kernel Alignment method was later developed into more efficient optimization setting by Lankriet et al. [4]. They developed Cristianini's idea into a quadratically constrained quadratic program by implementing multiple kernel combination into alignment maximization problem. They precalculate the numerator of Equ. 3 as $q_i = \langle \mathbf{K}_i, \mathbf{y}\mathbf{y}^T \rangle_F$, and the denominator $\mathbf{S}_{ij} = \langle \mathbf{K}_i, \mathbf{K}_j \rangle_F$ to reduce their number of constraints. Thus, the alignment maximization with multiple kernel combination becomes :

$$\begin{aligned} \max_{\boldsymbol{\mu}} \quad & \boldsymbol{\mu}^T \mathbf{q} \\ \text{subject to} \quad & \boldsymbol{\mu}^T \mathbf{S} \boldsymbol{\mu} \leq 1 \end{aligned} \tag{4}$$

In order to set up this optimization setting, computational complexity required is $\mathcal{O}(pdn^2 + n^2 + pn^2 + p^2n^2) : \mathcal{O}(pdn^2)$

for computing all p kernels (n^2 times d-vector operations), $\mathcal{O}(n^2)$ for the target kernel matrix, $\mathbf{y}\mathbf{y}^T$, $\mathcal{O}(pn^2)$ for $\mathbf{q}$ and $\mathcal{O}(p^2n^2)$ for $\mathbf{S}$. Later in this paper, our method is compared against Lankriet's alignment maximization.

II. Kernel parameter selection by Gap-Maximization

Our method initiates from considering the similarity among feature vectors for each sample. Considering that the elements of a kernel matrix measure the similarity between samples, it is natural to find kernel parameters $\boldsymbol{\mu} = [\mu_1, \cdots, \mu_p]$ that maximizes the minimum gap between within-class kernel element and the between-class kernel element. This can be formulated as

$$\begin{aligned} \underset{\mu}{\text{minimize}} \quad & -b \\ \text{subject to} \quad & k_{ji} - k_{li} \geq b, \forall i \in \{1, \cdots, n\}, \\ & k_{gh} = \sum_{m=1}^{p} \mu_m k_{gh}^{(m)}, \\ & j \in \{j | y_j = y_i, j \neq i\}, \\ & l \in \{l | y_l \neq y_i\}, \\ & \sum_{m=1}^{p} \mu_m = 1, \boldsymbol{\mu} \geq 0. \end{aligned} \tag{5}$$

This is a simple linear programming but it has $n_1 n_2 (n-2)$ constraints where n_1 and n_2 are the number of samples for class 1 and -1 respectively and satisfies $n = n_1 + n_2$. Because of the huge number of constraints, in our work, we try to reduce this number significantly by considering only the kernels that preserve relative distance information in the input space and RKHS.

For a given kernel matrix which is based on relative distances of input feature vectors such as Gaussian kernels, the information of the distances within the original space must be also conserved in the kernel space. In case of the Gaussian kernel

$$k_{ij} = k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|_2^2}{\sigma^2}\right), \tag{6}$$

k_{ij} depends on $\|x_i - x_j\|_2$ with σ^2 influencing only as an exponential scaling factor. Therefore the relative distance information between samples in RKHS does not change with parameter σ^2.

Consider binary classification problems with training data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ and its class labels $y_i \in \{-1, 1\}$. For this dataset, we first calculate the Euclidean distance matrix:

$$\mathbf{D} = \left[(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)\right]_{\forall i,j} = [d_{ij}]_{\forall i,j}, \tag{7}$$

where d_{ij} is the squared distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. For each column i of distance matrix $\mathbf{D}$ which represents distances in the original space from $\mathbf{x}_i$ to others, we find the index of a sample $\mathbf{x}_j$ with maximum distance from $\mathbf{x}_i$ among the ones that share the same class labels with $\mathbf{x}_i$ and the index of a sample $\mathbf{x}_l$ whose distance is minimum among the ones with different class labels than $\mathbf{x}_i$, as in Equ. 5.

After computing $\mathbf{D}$, only the two values in each column indexed in the previous step are going to feed into the p Gaussian kernel function:

$$\mathbf{K}^{(m)} = \exp\left(\frac{\mathbf{D}}{\sigma_m^2}\right) \tag{8}$$

where the $\exp(\cdot)$ is Hadamard (element-wise) exponential.

Then we try to find the kernel mixture parameters $\boldsymbol{\mu}$ in Equ. 1 that maximizes the difference between the elements of the combined kernel located in the indices of maximum and minimum value in each column that we figured from $\mathbf{D}$. The reason we primarily computed $\mathbf{D}$ is because when Gaussian kernel is used, the order of the samples by the distance does not change in RKHS because the relative distance is conserved by the distance measure within the kernel function.

This approach reduces a significant amount of computational burdens of MKL methods. To build a kernel matrix, it usually requires calculations in the order of n^2 along with d-vector operations during the training process. The computations increase even more up to pdn^2 for multiple kernel learning with p kernels. However, because it does not need to compute for all kernels, our method requires less computional expense. The construction of $\mathbf{D}$ needs dn^2 operations for d-vectors. After $\mathcal{O}(n^2)$ for findinig minimum and maximum values for all n columns is calculated, the last computation is computing kernel values for p kernels with $2 \times n$ values selected from $\mathbf{D}$ in the order of $2pn$. The whole computation for our method then requires $\mathcal{O}(dn^2 + n^2 + pn)$ before the optimization process, which is significantly less expensive than kernel alignment maximization in [4].

Let our final kernel be:

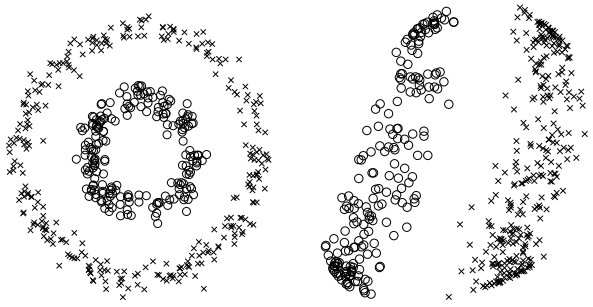
$$\mathbf{K} = \sum_{m=1}^{p} \mu_m \mathbf{K}^{(m)} = \left[\sum_{m=1}^{p} \mu_m k_{ij}^{(m)}\right]_{\forall i,j} = [k_{ij}]_{\forall i,j}. \tag{9}$$

Then, our primal problem becomes:

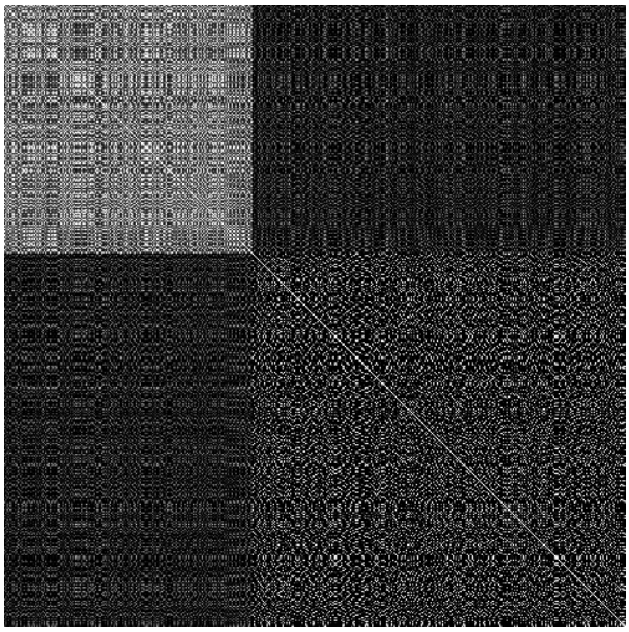
$$\begin{aligned} \underset{\mu}{\text{minimize}} \quad & -b \\ \text{subject to} \quad & k_{ji} - k_{li} \geq b, \forall i \in \{1, \cdots, n\}, \\ & k_{gh} = \sum_{m=1}^{p} \mu_m k_{gh}^{(m)}, \\ & j = \underset{f \in \{1,\ldots,i-1,i+1,\ldots,n\}}{\arg\max} k_{fi} \ \text{ s.t. } y_f = y_i, \\ & l = \underset{f \in \{1,\ldots,n\}}{\arg\min} k_{fi} \ \text{ s.t. } y_f \neq y_i, \\ & \sum_{m=1}^{p} \mu_m = 1, \boldsymbol{\mu} \geq 0. \end{aligned} \tag{10}$$

When computing for the maximally distant feature vectors for intra-class and the minimally distant feature vectors for inter-class, use of Gaussian kernels is computationally helpful.

As described earlier, if Equ. 5 is used, there will be $n_1 n_2 (n-1)$ constraints. However, if the combination of Gaussian kernels that relative distance does not depend on parameter values are used, $\mathbf{D}$ matrix allows the pre-understanding



(a) (b)



(c)

Fig. 1: Toy example of our method. (a) The graph shows input data that is non-linearly distributed around the graph. (b) By kernel principal component analysis (KPCA), the coordinates in the kernel space are represented by two major eigenvectors for the ease of structural understanding. (c) This is the kernel matrix with the values from 0 to 1 that maps the input data in (a) to coordinates in (b). As you can see, the upper-left and the lower-right are sparse in many of the kernel components.

of indices of where maximum and minimum are present in each column. This allows the reduction of constraints required to only 1 for each instance. The optimization requires n constraints after all.

Our optimal solution finds the right value of σ for the Gaussian kernels that maps data to the kernel space where data can be clustered within its neighbors the most. In doing so, our method does not try to make all intra-class kernel components to be 1, but only ones selected from $\mathbf{D}$. Considering the actual coordinates of feature vectors after they are mapped into kernel space, our optimal kernel space is where the vectors can be clustered with the ones of their own class. This suggests that intra-class part in the kernel matrix optimization can be partially sparse as long as it is fullly ranked matrix. If intra-class sections are full rank, it means that they can be clustered as together. Restricting the result from having the obvious solution which is to have all data becomes 1, each inter-class feature vector is taken to the furthest in the optimized kernel space, and minimally valued in the kernel matrix $\mathbf{K}$.

Algorithm 1 The Gap-Maximization to solve for the kernel space that is optimized by a convex combination of p Gausian kernels by maximizing the gap between the classes.

Input: Training data $\mathbf{X}$, training class c and kernel parameters $\sigma^2 = \left[\sigma_1^2, \ldots, \sigma_p^2\right]^T$.
Output: Weights of kernel parameter $\boldsymbol{\mu}$.

```
Construct D as (7).
for i = 1, ..., n do
    Find d_ji such that d_ji ≥ d_fi ∀f ∈ {f|y_f = y_i}.
    Find d_li such that d_li ≤ d_fi ∀f ∈ {f|y_f ≠ y_i}.
end for
for m = 1, ..., p do
    for i = 1, ..., n do
        k_ji^(m) = exp(-d_ji / σ_m^2).
        k_li^(m) = exp(-d_li / σ_m^2).
    end for
end for
Find μ = [μ_1, ··· , μ_p] solving (10) or (11).
```

III. Generalization

Instead of taking their indices for minimum and maximum values in account for each column, we take indices where nth minimum and maximum values are because of possible outliers. Considering furthest or closest feature vector in original space, if the distance is too large relative to other ones, the outliers can distract optimizing toward the right solution. Also, the more number of training instances, more probability of outliers in a dataset would be present. Thus, from $\mathbf{D}$, the indices for ϵth minimum and maximum values for each column are taken in order to optimize. Then our problem becomes:

$$
\begin{aligned}
\underset{\mu}{\text{minimize}} \quad & -b \\
\text{subject to} \quad & k_{ji} - k_{li} \geq b, \forall i \in \{1, \cdots, n\}, \\
& k_{gh} = \sum_{m=1}^{p} \mu_m k_{gh}^{(m)}, \\
& j = \underset{f \in \{1,\ldots,i-1,i+1,\ldots,n\}}{\epsilon^{\text{th}}\arg\max} k_{fi} \quad \text{s.t. } y_f = y_i, \\
& l = \underset{f \in \{1,\ldots,n\}}{\epsilon^{\text{th}}\arg\min} k_{fi} \quad \text{s.t. } y_f \neq y_i, \\
& \sum_{m=1}^{p} \mu_m = 1, \boldsymbol{\mu} \geq 0.
\end{aligned}
\tag{11}
$$

Fig. 2: Examples of face and non-face images.

TABLE I: Experiment result on Face vs. Non-face Dataset

	Alignment	Gap Maximization (ours)
Recognition Rate	97.25%	97.92%
Activated Sigma Value(s)	$10^{2.4}, 10^{2.5}$	$10^{3.1}$
Weighting Coefficient(s)	0.1748, 0.8252	1.0000

Since kernel tricks are usually used for non-linearly distributed data in the input space, we set up a binary class toy example data that was designed to have non-linear distribution. Figure 1 visualizes toy example of our method. Figure 1(b) clearly shows the data that is mapped to the kernel space by a kernel matrix constructed in Fig. 1(c) is easily separable by an optimized linear plane of support vector machine (SVM). As it also can be seen in Fig. 1(c), there exist many sparse components in kernel matrix while doing a good job in discriminating the features in the kernel space. This is resulted by our method that optimizes physical distance in the kernel space with its neighboring features to be close, not the values.

IV. Experiment

We evaluated our method on Face vs. Non-face data from the Color FERET face database [5], which consists 8,000 face and non-face instances in binary classes. The face images were cropped based on the centers of the right and left pupils and they were resized to 24x24 pixels. The non-face images were randomly cropped and resized to have the same size. The 400 test data instances are selected from the whole dataset, and the rest was used as training data. We have experimented kernel target alignment method of Lankriet's alignment maximization [4] and gap maximization method of ours to compare the performance of each. During the experiments, we used CVX toolbox [6] as an optimization solver. Since support vector machine (SVM) algorithm [7] is widely used, our experiments were also proceeded with SVM for the classifier.

We tried both algorithms with sigma values that increments by $10^{0.1}$ from 10^{-3} to 10^{6}. Each optimization resulted in different combination of sigma values. As the result, kernel target alignment is optimized to have combination of two dominating sigma values, $10^{2.4}$ and $10^{2.5}$ with weighting coefficients, $\mu_{10^{2.4}} = 0.1748$ and $\mu_{10^{2.5}} = 0.8252$, respectively. Our method is optimized to have one dominant sigma value activated, $10^{3.1}$ with a weighting coefficient $\mu_{10^{3.1}} = 1$.

The difference in recognition performance is not large, but our method is obtained by far lower amount of computations than kernel target alignment is. While our method only computes the ones that are ϵth indices for minimum and maximum values from the matrix $\mathbf{D}$, alignment method takes all the datasets into accounts in order to achieve its optimized solution.

V. Conclusion

We propose the method called, Gap Maximization between intra and inter-class samples, an efficient way of selecting opt kernel parameter, sigma value, for a given dataset by multiple kernel learning. The proposed method necessitates less compuational complexity than previously proposed MKL method, the kernel alignment maximization. Since computing the right parameters has been usually achieved by heuristic approaches, our method would make experiments more practical in terms of both performance and efficiency.

Further study of this method would be the applications of this method's approach to learning the kernel with multiple features. Datasets available these days are represented in different features for better recognition performances. Differentiating features that actually useful in the process of recognition from the negligible features are important process of feature extraction. By feeding multiple features and combining them in the optimization, one can identify features that are significant for recognition more efficiently.

References

[1] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *J. Machine Learning Research, JMLR*, vol. 12, pp. 2211–2268, 2011.

[2] S. S. Bucak, R. Jin, and A. K. Jain, "Multiple kernel learning for visual object recognition: A review," *IEEE Trans. Pattern Analysis and Machine Intelligence, TPAMI*, vol. 36, no. 7, pp. 1354–1369, 2014.

[3] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola, "On kernel-target alignment," in *Advances in Neural Information Processing Systems, NIPS*, 2001, pp. 367–373.

[4] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Machine Learning Research, JMLR*, vol. 5, pp. 27–72, 2004.

[5] P. J. Phillips, H. Moon, S. Rizvi, P. J. Rauss *et al.*, "The feret evaluation methodology for face-recognition algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence, TPAMI*, vol. 22, no. 10, pp. 1090–1104, 2000.

[6] M. Grant and S. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, 2008, pp. 95–110.

[7] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.