ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0. Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

Learning geometric combinations of Gaussian kernels with alternating Quasi-Newton algorithm

David Picard[1], Nicolas Thome[2], Matthieu Cord[2], Alain Rakotomamonjy[3]

1- ETIS - ENSEA/CNRS/Université de Cergy Pontoise
6 avenue du Ponceau, F-95000 Cergy Pontoise

2- LIP6 - UPMC Univ Paris 6
4 place Jussieu, 75005 Paris, France

3- LITIS EA4108, INSA-Université de Rouen
Avenue de l'université, 76800 Saint Etienne du Rouvray

Abstract. We propose a novel algorithm for learning a geometric combination of Gaussian kernel jointly with a SVM classifier. This problem is the product counterpart of MKL, with restriction to Gaussian kernels. Our algorithm finds a local solution by alternating a Quasi-Newton gradient descent over the kernels and a classical SVM solver over the instances. We show promising results on well known data sets which suggest the soundness of the approach.

1 Introduction

Multiple Kernel Learning(MKL) [1] is now widely used in machine learning applications as an alternative method for combining multiple features, thanks to the availability of efficient algorithms [2, 3]. It is also a very hot topic in Computer Vision community where the visual feature combination problem is very challenging for classification tasks (see for instance the workshop on Kernels and Distances for Computer Vision at ICCV 2011). Different features are obtained by projection of local descriptors on a visual codebook, and MKL strategies used to optimize their combination [4]. More general combinations have been proposed by Varma et al. in [5] along with a gradient descent based learning algorithm named GMKL, which when considered with a sum combination boils down to the MKL problem. Varma also investigate the use of a weighted product combination of Gaussian kernels and found it to be more powerful than the sum combination.

In this paper, we further analyze the product combination of Gaussian kernels. Our contributions are three fold: (1) For our specific kernel combination, called PKL, we propose a novel optimization scheme, different from the Varma's algorithm. It is based on alternating a quasi-Newton gradient descent for learning the kernel weights and a classical SVM solver for the sample weights; (2) We discuss about convexity for these classes of problems and show that the associated optimization problem is non-convex; (3) we show promising results on well known data sets.

2 Product kernel learning

We first define the notations used in this paper. Let $\{\mathbf{x}_i \in \mathbb{R}^M\}_{1\leq i\leq N}$ be a training set of samples $\mathbf{x}_i$, and f a decision function in a RKHS $\mathcal{H}$. $\mathcal{H}$ is the combination of M RKHS $\mathcal{H}_m$ with minor kernels $K_m(\cdot,\cdot) = e^{-\gamma_m d_m(\cdot,\cdot)}$ such that the major kernel $K(\cdot,\cdot) = \prod_m e^{-\gamma_m d_m(\cdot,\cdot)}$. We denote $d_m(\cdot,\cdot)$ the distance on the subspace $\mathcal{H}_m$. We propose to obtain the decision function f by solving the following optimization problem:

$$\begin{array}{rl} \min\limits_{\gamma,f,b,\xi} & \frac{1}{2}||f||^2_{\mathcal{H}} + C\sum_{i=1}^N \xi_i \\ s.t. & y_i(f(\mathbf{x}_i)+b) \geq 1-\xi_i,\ \xi_i \geq 0, \forall i \quad \gamma_m \geq 0, \forall m \end{array} \tag{1}$$

subscript m always denotes the minor kernels and ranges from 1 to M, and subscript i and j always denote the training samples and range from 1 to N. For the sake of clarity, we now omit these ranges in the summation/product symbols.

We name this learning problem *Product Kernel Learning* (PKL). In [5], Varma et al. proposed an algorithm for learning any general combination of kernels named GMKL. Our problem is thus a special case of GMKL restricted to a product of Gaussian kernels. However, it can be applied to any product of Gaussian kernels, regardless of the distance used. Following the methodology of [2], we propose to solve this optimization problem using an alternate strategy :

$$\min_\gamma J(\gamma) \quad \text{such that} \quad \gamma_m \geq 0, \forall m \tag{2}$$

with:

$$J(\gamma) = \left\{ \begin{array}{cc} \min_{f,b,\xi} & \frac{1}{2}||f||^2_{\mathcal{H}} + C\sum_i \xi_i \\ s.t. & y_i(f(\mathbf{x}_i)+b) \geq 1-\xi_i, \forall i \\ & \xi_i \geq 0, \forall i \end{array} \right. \tag{3}$$

We remark that $J(\gamma)$ is the optimal value of a standard SVM problem and thus it can be expressed using a standard SVM dual formulation:

$$\begin{array}{rrl} J(\gamma) & = & \max\limits_\alpha \lambda(\alpha,\gamma) \\ & s.t. & \lambda(\alpha,\gamma) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \prod_m e^{-\gamma_m d_m(\mathbf{x}_i,\mathbf{x}_j)} \\ & & \sum_i \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad \forall i \end{array} \tag{4}$$

thanks to the strong duality of the SVM problem, we can express the PKL problem using this dual formulation:

$$\begin{array}{rl} \min\limits_\gamma \max\limits_\alpha & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \prod_m e^{-\gamma_m d_m(\mathbf{x}_i,\mathbf{x}_j)} \\ s.t. & \sum_i \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad \forall i \end{array} \tag{5}$$

We now want to stress two points about the optimization problem. First of all, one should note that our PKL framework does not implicitly impose any sparsity constraints in the optimization problem, making it rather simple. However, as we

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0. Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

shall see in the experimental section, the algorithm still tends to make weights related to irrelevant minor kernels to 0, inducing then sparsity in that sense. Another point that we make clear in this paper is the non-convexity of the problem. Although, there exists a general intuitive agreement that the problem (5) is not convex [5], we provide in the sequel a clear proof of this statement.

Finally, similarities can be seen with model selection problems for ARD Kernel like in [6]. However, algorithms in this category usually optimize their criteria on a validation set, whereas our optimization problem is defined on a single training set.

3 QN-PKL optimization scheme

In [5], the authors propose an alternating procedure between α and γ subproblems, where the γ step is performed using a simple gradient descent. We propose to improve this step by using a Quasi-Newton descent. The overall optimization scheme is shown in algorithm 1, where Γ is the vector of γ_m parameters and A the vector of $\alpha_i y_i$ parameters.

In the inner loop, we obtain the $\alpha^\star$ solution of equation (4) by the use of a very fast and efficient SVM solver (in our case LaSVM [7]). We approximate the inverse Hessian matrix by a diagonal matrix $\mathbf{B}$ inspired by successful Quasi-Newton procedure used in large scale linear SVM [8]. Consequently, we named the algorithm QN-PKL. At each iteration t, we search for an improvement of the objective value in the descent direction $\mathbf{p}_{t-1}$ (opposite of gradient direction), weighted by $\mathbf{B}_{t-1}$ and a step size δ. The step size δ is decimate exponentially until an improvement is found or $\delta < \varepsilon$. After the possible objective function improvement, the descent direction $\mathbf{p}_t$ is updated according to the following equation:

$$\mathbf{p}_t \leftarrow -\frac{1}{2}\left[A^\top (D_m \circ K) A\right]_m \tag{6}$$

With $\circ$ being the Hadamard product between matrices. $\mathbf{B}$ is updated according to the following diagonal discrete approximation:

$$\mathbf{B}_t \leftarrow \left[\frac{\Gamma_t[m] - \Gamma_{t-1}[m]}{\mathbf{p}_t[m] - \mathbf{p}_{t-1}[m]}\right]_m \tag{7}$$

4 Convexity

In the hereby section, we prove the PKL problem to be non convex. The proof is based on showing that $\lambda(\alpha, \gamma)$ is non convex with respects to γ, making thus the full problem (5) non-convex. Let us compute the Hessian H_γ of λ over γ :

$$\begin{aligned} H_\gamma = \frac{\partial^2 \lambda}{\partial \gamma_k \partial \gamma_l} &= -\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j d_k(\mathbf{x}_i \mathbf{x}_j) d_l(\mathbf{x}_i, \mathbf{x}_j) \prod_m e^{-\gamma_m d_m(\mathbf{x}_i, \mathbf{x}_j)} && (8) \\ &= -\frac{1}{2} A^\top (D_k \circ D_l \circ K) A && (9) \end{aligned}$$

Algorithm 1 QN-PKL

```
Input: training set $\mathcal{A} = \{x_i, y_i\}$, parameters $\Gamma_1 = \{\gamma_m\}$ inited to $\frac{1}{M}$, threshold $\varepsilon$,
initial learning rate $b_1$
$(A, o) \leftarrow$SVMSolve($\mathcal{A}$, $\Gamma$) // initial alpha coefficients, initial cost $o$
$\mathbf{p}_1 \leftarrow -\frac{1}{2}\left[A^\top (D_m \circ K) A\right]_m$, $\mathbf{B}_1 \leftarrow [b_1]_m$ // init descent direction and scale
$g \leftarrow \infty$ // set objective gap to infinity
$t \leftarrow 1$ // first iteration
while $g > \varepsilon$ do
    $\delta \leftarrow \delta_0$ // init $\delta$ value
    repeat
        $\Gamma' \leftarrow \Gamma_t + \delta \mathbf{B}_t \mathbf{p}_t$ // update $\Gamma$
        $(A, r) \leftarrow$ SVMSolve($\mathcal{A}$, $\Gamma'$) // update $\alpha$ coefficients, new cost $r$
        $\delta \leftarrow \frac{\delta}{10}$ // decrease update step for next loop
    until $r < o$ or $\delta < \varepsilon$
    $t \leftarrow t + 1$ // increase iterations
    $\Gamma_t \leftarrow \Gamma'$ // store new $\Gamma$
    $\mathbf{p}_t \leftarrow -\frac{1}{2}\left[A^\top (D_m \circ K) A\right]_m$// update descent direction
    $\mathbf{B}_t \leftarrow \left[\frac{\Gamma_t[m]-\Gamma_{t-1}[m]}{\mathbf{p}_t[m]-\mathbf{p}_{t-1}[m]}\right]_m$ // update direction scale
    $g \leftarrow \frac{o-r}{o}$, $o \leftarrow r$ // compute objective gap and store old value
end while
```

with A being the vector of αy, D_k the distance matrix relative to component k, and K the Gram matrix of kernel $K(\cdot,\cdot)$.

Suppose H_γ is positive semi-definite, then all its diagonal values are positive or null for any A coefficients:

$$\forall A,\ \forall k, \quad -\frac{1}{2} A^\top (D_k^{\circ 2} \circ K) A \geq 0 \tag{10}$$

which means that the matrix $C = -\frac{1}{2} D_k^{\circ 2} \circ K$ is itself positive semi-definite. Let us denote Λ the vector of eigenvalues of C. We remark that:

$$tr(C) = \Lambda^\top \mathbf{1} = 0 \tag{11}$$

However, all eigenvalues of C are positive or null since C is positive semi-definite, thus $\Lambda = 0$ and consequently $C = 0$. This result is absurd since neither D_k nor K are null (outside the trivial case were all samples $\mathbf{x}_i$ are the same vector). This means that C is not positive semi-definite, and consequently H_γ is not positive semi-definite.

The same reciprocal reasoning can be done with negative semi-definite property, which in turns means that H_γ is indefinite. The direct consequence is that the complete Hessian H with respects to all primal variables is indefinite and thus PKL optimization is not convex.

Table 1: Classification accuracies for SimpleMKL and QN-PKL on UCI data sets. M is the number of features.

Data	M	MKL (%)	time	QNPKL (%)	time	GMKL[5]	SVM
Sonar	60	77.9 ±5.5	6.2s	**86.6** ±3.4	1.0s	82.3 ±4.8	81.3 ±3.9
Iono	34	91.3 ±2.3	7.1s	**94.1** ±1.5	0.6s	93.0 ±2.1	90.1 ±2.3
Pima	8	71.4 ±4.1	4.4s	76.4 ±1.9	2.8s	**77.2** ±**2.1**	75.4 ±2.3

5 Experiments

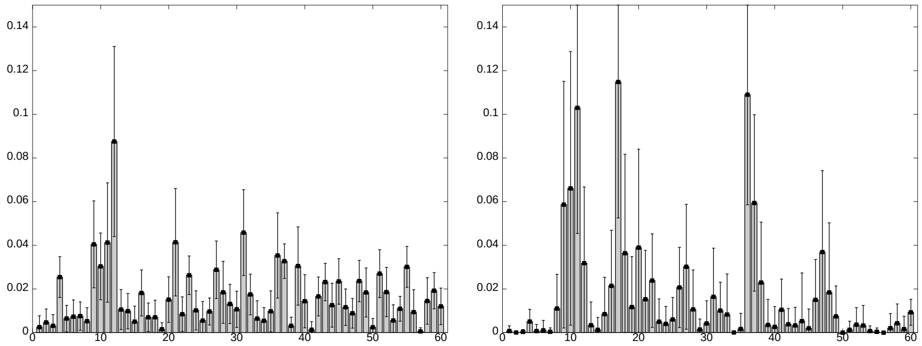
We carried out experiments comparing our QN-PKL with MKL and SVM on 3 standard UCI data sets. We used the same setup as in [5] (*i.e.* one Gaussian kernel per component as minor kernels, SVM using a product combination with uniform weights). For fairness of the MKL time evaluation, we use our own Java implementation of SimpleMKL [2] with the same internal SVM solver[1].

In table 1, we present a comparison of mean accuracies on *sonar*, *ionosphere*, and *pima* data sets. We can see that the PKL performs better on all three data-sets. This can be explained by the fact that the generalized Gaussian kernel is more powerful than a sum of axis-wise Gaussian kernels, even with PKL converging to a local optimum.

We also report results from [5] on these datasets. We obtain better performances on larger dimensional data-sets, and similar performances on lower dimensional data-sets. It is worth noting that GMKL uses a ℓ_2 regularization penalty on low dimensional data-set (*e.g. pima*) which leads to less sparse combination and better results, whereas QN-PKL doesn't use any regularization constraint at all.

We plot kernel weights (normalized to 1) of both algorithms for the *Sonar* data set on Fig. 1. We can see that QN-PKL outputs less uniform combinations than MKL, even though there is no sparse regulization constraint in our algo-

[1] software is available at `http://perso-etis.ensea.fr/~picard`



(a) Weights of MKL.

(b) Weights of QN-PKL.

Fig. 1: Comparison of obtained weights for the Sonar data set.

Table 2: Classification accuracies for SimpleMKL and QN-PKL on Noisy UCI data sets.

Data set	MKL (%)	R	QN-PKL (%)	R
Sonar	72.2 ± 3.5	84.9	**80.8** ± 6.3	97.7
Inonosphere	91.2 ± 2.6	67.2	**93.3** ± 2.4	99.9
Pima	71.5 ± 3.4	61.1	**74.8** ± 3.0	89.9

rithm. We explain this natural sparse behavior by the product of kernels which acts as an 'AND' gate.

Regarding computational time, QN-PKL seems between 5 and 10 times faster than SimpleMKL on a 3GHz Intel Core2 Duo laptop.

To investigate the sparsity of QN-PKL, we then added 25 random features sampled from the Normal distribution at the end of each input vectors. In table 2, we present accuracies for these noisy data sets. We also compute the ratio $R = \sum_{m=1}^{M} \gamma_m / \sum_{m=1}^{M+25} \gamma_m$ of the relevant weights over the total weights. Using this measure, we can see that QN-PKL is much more efficient at zeroing noisy kernels than MKL, even without any sparsity constraint.

6 Conclusion

This paper presents a fast Quasi-Newton algorithm for learning a product of Gaussian kernels, named QN-PKL. We prove the related problem to be non-convex. QN-PKL finds a local solution by alternating a standard SVM optimization loop for learning sample weights, and a approximate second order descent for the kernel weights. Experiments show our algorithm to be as effective as SimpleMKL while being significantly faster. Moreover, QN-PKL seems to be naturally more efficient at discarding irrelevant features than MKL.

References

[1] Francis R. Bach and Gert R. G. Lanckriet. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the 21st ICML*, 2004.

[2] Alain Rakotomamonjy, Francis R. Bach, Stéphane Canu, and Yves Grandvalet. SimpleMKL. *JMLR*, 9:2491–2521, November 2008.

[3] Olivier Chapelle and Alain Rakotomamonjy. Second order optimization of kernel parameters. In *In NIPS Workshop on Kernel Learning*, 2008.

[4] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *2009 IEEE 12th ICCV*, pages 606–613. IEEE, 2009.

[5] Manik Varma and Bodla Rakesh Babu. More generality in efficient multiple kernel learning. In *Proceedings of the 26th ICML*, ICML '09, pages 1065–1072, New York, NY, USA, 2009.

[6] T. Glasmachers and C. Igel. Maximum likelihood model selection for 1-norm soft margin svms with multiple parameters. *IEEE Transactions on PAMI*, 32(8):1522–1528, 2010.

[7] Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *JMLR*, 6:1579–1619, September 2005.

[8] Antoine Bordes, Léon Bottou, and Patrick Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *JMLR*, 10:1737–1754, July 2009.