Multi-class Image Classification: Learning Output Kernels with Block Coordinate Descent

Chia-Man Hung Wei Jiang Zhengying Liu Xiaoling Zhu

April 5, 2017

Abstract

Output Kernel Learning (OKL) is a kernel-based technology to deal with learning problems with multiple outputs, such as multi-class and multi-label classification or vectorial regression, while automatically learning a kernel which can exploit the relationship between different output components. OKL is based on a regularization problem over a reproducing kernel Hilbert space (RKHS), which can be solved by a block-wise coordinate descent method. When the dimensionality of the output space is high, constraints on the output kernel rank may be useful to make the algorithm computationally feasible. To demonstrate that OKL can improve the accuracy of prediction and reveal the output structure, we implement OKL together with low rank OKL and experiment them on several tasks, including digit recognition and image classification. We compare the training time and the accuracy of OKL and low rank OKL with different parameters.

Contents

1 Introduction

Nowadays, methods for learning vector-valued functions have become popular research topics, motivated by applications in multi-output regression, multi-label and multi-class classification. For these tasks, selecting a model that correctly exploits the relationships among different output components is crucial to ensure good learning performances. In this project, we introduce a method that simultaneously learns a vector-valued function and the kernel between the output components.

Within the framework of regularization in reproducing kernel Hilbert spaces (RKHS) of vector-valued functions [1] [6] [2], the matrix-valued kernel can be decomposed as the product of a scalar kernel (input kernel) and a positive semi-definite kernel that represents the similarity between the output components (output kernel). The output kernel can be used for visualization and revealing structures in output space, for example in multi-class classification task, output kernel can help to cluster classes into homogeneous group. The choice of the output kernel may significantly influence learning performance, especially when prior knowledge is not sufficient to fix the output kernel in advance. Therefore, it is important to select an efficient data-driven method to learn an output kernel.

The paper [5] first proposes an optimization problem of regularized risk functional, then shows that the objective function is invex so the stationary points are global minimizers. Finally the solution can be obtained by block coordinate descent.

However, this method directly operates on the full output kernel matrix which is full-rank. But when the dimensionality of the output space is very high, storing and manipulating the full matrix may not be efficient or feasible. Another paper [4] introduces a new OKL method that enforces a rank constraint on the output kernel and directly operates on a factor of the kernel matrix (low-rank OKL). The optimization problem can be seen as the kernelized version of nuclear norm regularization, which can also be solved by block coordinate descent algorithm.

This report is organized as follows. First, to make it self-contained, we recall some notions of reproducing kernel Hilbert space in the general case of operator-valued kernels and formulate the learning tasks as an optimization problem. Then, we introduce two methods - the OKL and the low rank OKL - that give an approximate solution. Last, we show the results of our experiments on different learning tasks and compare the two methods.

2 Problem

2.1 Reproducing Kernel Hilbert Spaces

First we recall several notations and facts of RKHS. We suppose that $\mathcal{Y}$ denote a Hilbert space endowed with inner product $\langle \cdot, \cdot \rangle_{\mathcal{Y}}$, and $\mathcal{L}(\mathcal{Y})$ the space of bounded linear operators from $\mathcal{Y}$ to itself. Every RKHS of $\mathcal{Y}$-valued functions (over $\mathcal{X}$) $\mathcal{H}$ can be associated with a unique positive semi-definite $\mathcal{Y}$-kernel H, called the reproducing kernel. Conversely, given a positive semi-definite $\mathcal{Y}$-kernel H on $\mathcal{X}$, there exists a unique

RKHS of $\mathcal{Y}$-valued functions defined over $\mathcal{X}$ whose reproducing kernel is H. For later use, we recall that the **reproducing property** in this operator-valued kernel version is that the following equality

$$\langle H(x,\cdot)y, g\rangle_{\mathcal{H}} = \langle y, g(x)\rangle_{\mathcal{Y}}$$

holds for any $x \in \mathcal{X}, y \in \mathcal{Y}, g \in \mathcal{H}$.

In our tasks of regression or classification, we assume $\mathcal{Y} = \mathbb{R}^m$, then $\mathcal{L}(\mathcal{Y})$ is the space of square matrices of order m, denoted by $\mathbb{M}^m$. By fixing a basis $\{b_i\}_{i\in\mathcal{T}}$ for the output space, where $\mathcal{T} = \{1, \cdots, m\}$, we can uniquely define an associated (scalar-valued) kernel R over $\mathcal{X} \times \mathcal{T}$ such that

$$\langle b_i, H(x_1, x_2)b_j\rangle_{\mathcal{Y}} = R((x_1, i), (x_2, j))$$

So that an $\mathcal{Y}$-kernel can be seen as a function that maps two inputs into $\mathbb{M}^m$. Similarly by fixing any function $g : \mathcal{X} \to \mathcal{Y}$, we can uniquely define an associated function $h : \mathcal{X} \times \mathcal{T} \to \mathbb{R}$ such that

$$g(x) = \sum_{i\in\mathcal{T}} h(x, i)b_i$$

Finally, by fixing a basis for the output space, the problem of learning a vector-valued function can be equivalently regarded as a problem of learning a scalar function defined over an enlarged input set.

2.2 Learning Tasks

First we should learn a function taking values in $\mathbb{R}^m$. In multiple output regression, each component of the vector directly corresponds to an output. For multi-class classification, output data are modeled as binary vectors, with $+1$ in the position corresponding to the class. We regard g as confidence scores for the different classes, and the classification rule can be considered as

$$\hat{y}(x) = \arg\max_{i\in\mathcal{T}} g_i(x)$$

For binary multi-label classification, outputs are vectors with $+1$ or -1 at the different components. The classification rule is given by

$$\hat{y}_i(x) = \text{sign}(g_i(x))$$

2.3 Regularized Risk

We assume that $\mathbb{S}^m_{++}$ denotes the open cone of positive definite matrices, $\mathbb{S}^m_{+}$ denotes the closed cone of positive definite matrices. For any $\mathbf{A}$, $\mathbf{B} \in \mathbb{M}^m$, we denote the Frobenius inner product $\langle \mathbf{A}, \mathbf{B}\rangle_F := tr(\mathbf{A}^T\mathbf{B})$ and Frobenius norm $||\mathbf{A}||_F = \sqrt{\langle \mathbf{A}, \mathbf{A}\rangle_F}$.

Let $\mathcal{H}$ denote the RKHS of $\mathcal{Y}$-valued functions $g : \mathcal{X} \to \mathcal{Y}$ associated with the kernel H defined as

$$H = K\mathbf{L}$$

where K is a positive semi-definite scalar kernel on $\mathcal{X}$ that measures the similarity between input, and $\mathbf{L} \in \mathbb{S}_+^m$ is a symmetric positive semi-definite kernel that encodes the relationship between output components.

In order to simultaneously learn g and $\mathbf{L}$ from a training set of size l with data pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, we propose a regularized optimization problem on the risk function

$$\min_{\mathbf{L} \in \mathbb{S}_+^m} \min_{g \in \mathcal{H}} \sum_{i=1}^{l} \frac{||g(x_i) - y_i||_2^2}{2\lambda} + \frac{||g||_{\mathcal{H}}^2}{2} + \frac{||\mathbf{L}||_F^2}{2} \tag{1}$$

3 Methods

3.1 OKL with Block Coordinate Descent

According to the representer theorem, the solution of inner optimization problem has the form

$$g^*(x) = \sum_{i=1}^{l} H(x, x_i) c_i = \mathbf{L} \sum_{i=1}^{l} c_i K(x, x_i) \tag{2}$$

where $c_i \in \mathcal{Y} (i = 1, \cdots, l)$. Assume that $\mathbf{Y} = (y_1, \cdots, y_l)^T$, $\mathbf{C} = (c_1, \cdots, c_l)^T$, $\mathbf{K}_{ij} = K(x_i, x_j)$, then the problem (1) can be rewritten as

$$\min_{\mathbf{L} \in \mathbb{S}_+^m} \min_{\mathbf{C} \in \mathbb{R}^{l \times m}} \frac{||\mathbf{Y} - \mathbf{KCL}||_F^2}{2\lambda} + \frac{\langle \mathbf{C}^T \mathbf{KC}, \mathbf{L} \rangle_F}{2} + \frac{||\mathbf{L}||_F^2}{2} \tag{3}$$

To see this (the following computation is omitted in the paper [5]), we have for the first term

$$g(x_i) = \mathbf{L} \sum_{j=1}^{n} c_j K(x_j, x_i) = \mathbf{L}(\mathbf{C}^\top \mathbf{K})_i,$$

so

$$[g(x_1), ..., g(x_n)]^\top = (\mathbf{L}(\mathbf{C}^\top \mathbf{K}))^\top = \mathbf{KCL}$$

thus

$$\frac{1}{2\lambda} \sum_{i=1}^{l} \|y_i - g(x_i)\|_2^2 = \frac{1}{2\lambda} \|\mathbf{Y} - \mathbf{KCL}\|_F^2.$$

For the second term in (1), we can apply the reproducing property of the kernel $H = K\mathbf{L}$ and get

$$\begin{aligned} \|g\|_{\mathcal{H}}^2 &= \langle \sum_i H(x_i, \cdot) c_i, \sum_j H(x_j, \cdot) c_j \rangle_{\mathcal{H}} = \sum_{i,j} \langle H(x_i, \cdot) c_i, H(x_j, \cdot) c_j \rangle_{\mathcal{H}} \\ &= \sum_{i,j} \langle c_i, H(x_j, x_i) c_j \rangle_{\mathcal{Y}} = \sum_i \langle c_i, \mathbf{L} \sum_j K(x_j, x_i) c_j \rangle_{\mathcal{Y}} \\ &= \sum_i \langle c_i, \mathbf{L}(\mathbf{C}^\top \mathbf{K})_i \rangle_{\mathcal{Y}} = \mathrm{tr}(\mathbf{CLC}^\top \mathbf{K}) = \mathrm{tr}(\mathbf{LC}^\top \mathbf{KC}) \\ &= \langle \mathbf{C}^T \mathbf{KC}, \mathbf{L} \rangle_F. \end{aligned} \tag{4}$$

We let $Q(\mathbf{L}, \mathbf{C})$ denote the objective function, which is convex with respect to $\mathbf{C}$ and strongly convex with respect to $\mathbf{L}$. Moreover, Q is an invex function with respect to $(\mathbf{L}, \mathbf{C})$, which ensures that the stationary point is global minimizer. [10] In order to solve the problem (3), we apply the block-wise coordinate descent as in Algorithm 1, which minimizes $Q(\mathbf{L}, \mathbf{C})$ w.r.t $\mathbf{L}$ (respectively $\mathbf{C}$) by fixing $\mathbf{C}$ (respectively $\mathbf{L}$).

Algorithm 1 OKL with Block Coordinate Descent

1: **Input K**
2: **Initialization L**, **C**, **E**, **Z** $\leftarrow 0$
3: **while** $||\mathbf{Z} + \lambda\mathbf{C} - \mathbf{Y}||_F \geq \delta$ **do**
4: $\quad \mathbf{C} \leftarrow$ solution to $\mathbf{KCL} + \lambda\mathbf{C} = \mathbf{Y}$
5: $\quad \mathbf{E} \leftarrow \mathbf{KC}$
6: $\quad \mathbf{P} \leftarrow \frac{1}{2}\mathbf{E}^T\mathbf{C} - \mathbf{L}$
7: $\quad \mathbf{Q} \leftarrow$ solution to $(\mathbf{E}^T\mathbf{E} + \lambda\mathbf{IQ}) = \mathbf{P}$
8: $\quad \mathbf{L} \leftarrow \mathbf{L} + \lambda\mathbf{Q}$
9: $\quad \mathbf{Z} \leftarrow \mathbf{EL}$
10: **end while**
11: **Output C, L**

As the subproblem w.r.t $\mathbf{L}$ is well explained in [5], we will only give more details on how we implement the subproblem w.r.t $\mathbf{C}$ in line 4. The subproblem aims to find $\mathbf{C}$ such that

$$\mathbf{KCL} + \lambda\mathbf{C} = \mathbf{Y}.$$

As both $\mathbf{K}$ and $\mathbf{L}$ are real symmetric matrix (thus diagonalizable), we can find two orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$ such that

$$\mathbf{KU} = \mathbf{UD_K}$$

and

$$\mathbf{LV} = \mathbf{VD_L}$$

where $\mathbf{D_K} = \text{diag}\{\lambda_1, ..., \lambda_\ell\}$ and $\mathbf{D_L} = \text{diag}\{\mu_1, ..., \mu_m\}$ are two diagonal matrices. Then by multiplying $\mathbf{U}^\top$ on the left and $\mathbf{V}$ on the right, we get

$$\mathbf{U}^\top\mathbf{KCLV} + \lambda\mathbf{U}^\top\mathbf{CV} = \mathbf{D_K}\mathbf{U}^\top\mathbf{CVD_L} + \lambda\mathbf{U}^\top\mathbf{CV} = \mathbf{U}^\top\mathbf{YV}$$

so we have the simple relation

$$(\mathbf{U}^\top\mathbf{CV})_{ij} = \frac{1}{\lambda + \lambda_i\mu_j}(\mathbf{U}^\top\mathbf{YV})_{ij}$$

and we can reconstruct $\mathbf{C}$ from $\mathbf{U}^\top\mathbf{CV}$ easily, again from $\mathbf{U}$ and $\mathbf{V}$. So all we need is to find the eigen-decomposition for $\mathbf{K}$ (once) and for $\mathbf{L}$ (several times).

3.2 Low Rank OKL with Block Coordinate Descent

However, the method above directly operates on the full output kernel. But when the dimensionality of the output space is very high, storing and manipulating the full matrix may not be efficient or feasible. Now we apply a new OKL method that enforces a rank constraint on the output kernel and directly operates on a factor of the kernel matrix (low-rank OKL).

Now we rewrite the optimization (1) with constraint of output rank.

$$\min_{\mathbf{L}\in\mathbb{S}_+^{m,p}} \min_{g\in\mathcal{H}} \sum_{i=1}^{l} \frac{||g(x_i)-y_i||_2^2}{2\lambda} + \frac{||g||_{\mathcal{H}}^2}{2} + \frac{||\mathbf{L}||_F^2}{2} \tag{5}$$

with the notation

$$\mathbb{S}_+^{m,p} = \{\mathbf{A}\in\mathbb{S}_+^m :\ \text{rank}(\mathbf{A}) \le p\}$$

Remark that although the objective function is invex, due to the rank constraint, the stationary points are not guaranteed to be feasible.

Now we consider a map $g : \mathcal{X} \to \mathcal{Y}$ of the form

$$g(x) = (g_2 \circ g_1)(x)$$

where g_1 belongs to an RKHS $\mathcal{H}_1$ of vector-valued functions whose kernel is decomposable as $H = K\mathbf{I}$, and g_2 belongs to $\mathcal{H}_2$ of linear operators of the type $g_2(z) = \mathbf{B}z$. We can interpret that g_1 performs a non-linear feature extraction and g_2 combines the extracted features to produce output vector.

The paper [5] shows that the problem (5) is equivalent to

$$\min_{g_2\in\mathcal{H}_2} \min_{g_1\in\mathcal{H}_\infty} \sum_{i=1}^{l} \frac{||y_i-(g_2\circ g_1)(x_i)||_2^2}{2\lambda} + \frac{||g_1||_{\mathcal{H}_1}^2}{2} + \frac{||g_2||_{\mathcal{H}_2}^2}{2} \tag{6}$$

According the the representer theorem, the inner problem of (6) admits a solution of the form $g_1 = \sum_{i=1}^{l} a_i K_{x_i}$ and thus $g = \mathbf{B}(\sum_{i=1}^{l} a_i K_{x_i})$. With the notation $\mathbf{A} = (a_1, \cdots, a_l)^T$, the problem 6 can be rewritten as

$$\min_{\mathbf{B}\in\mathbb{R}^{m\times p}} \min_{\mathbf{A}\in\mathbb{R}^{l\times p}} \frac{||\mathbf{Y}-\mathbf{K}\mathbf{A}\mathbf{B}^T||_F^2}{2\lambda} + \frac{\langle\mathbf{A},\mathbf{K}\mathbf{A}\rangle_F}{2} + \frac{||\mathbf{B}||_F^2}{2} \tag{7}$$

In order to solve the problem (7), we apply the block-wise coordinate descent.

4 Experiments

4.1 Implementation

Our code[1] is written in python. We mainly use the `opencv` library for image feature extraction, `sklearn` for data splitting and SVM, and `numpy` for basic computation.

In a nutshell, we implement OKL and low rank OKL for all tasks by adapting from the Matlab code provided on Dinuzzo's website[2]. For multi-class image classification,

[1] `https://github.com/zhengying-liu/Structured-Data-Project`

[2] `http://people.tuebingen.mpg.de/fdinuzzo/okl.html`

Algorithm 2 Low rank OKL with Block Coordinate Descent

Input K
Initialization $\mathbf{B} \leftarrow \mathbf{I}_{m \times p}$
eigendecomposition $\mathbf{K} = U_X \text{diag}(\lambda_X) U_X^T$
$\tilde{\mathbf{Y}} \leftarrow U_X^T \mathbf{Y}$
do
  eigendecomposition $\mathbf{B}^T\mathbf{B} = U_Y \text{diag}(\lambda_Y) \mathbf{U}_Y^T$
  $\mathbf{Q} \leftarrow \tilde{\mathbf{Y}}\mathbf{B}\mathbf{U}_Y$
  $\mathbf{V} \leftarrow \mathbf{Q} \oslash (\lambda_X \lambda_Y^T + \lambda e e^T)$
  $\mathbf{B}_p \leftarrow \mathbf{B}$
  $\mathbf{E} \leftarrow \text{diag}(\lambda_X)\mathbf{V}$
  $\mathbf{B} \leftarrow \tilde{\mathbf{Y}}^T \mathbf{E}(\mathbf{E}^T\mathbf{E} + \lambda \mathbf{I})^{-1} \mathbf{U}_Y^T$
until $||\mathbf{B} - \mathbf{B}_p||_F \geq \delta$
$\mathbf{A} \leftarrow U_X \mathbf{V} \mathbf{U}_Y^T$
Output A,B

We also implement Histograms of Oriented Gradients (HOG) [3] on our own, a simple classical feature extraction method, but it turns out that our implementation is slower than that provided by the `opencv` library. Thus, we rather adopt their implementation.

4.2 Multi-Output Regression

In the first experiments, we use the synthetic multiple time series reconstruction datasets provided by Dinuzzo. We choose the number of processes (i.e. the output dimension) $m = 200$ and randomly extract $l = 100$ samples to be used for training, another 100 as validation set for tuning the regularization parameter λ and rank p, and finally the remaining 100 as test set are used for examining the prediction accuracy with tuned λ. The input kernels $\mathbf{K}$ for the three sets are available in the dataset.

We first implement OKL and low rank OKL with block coordinate descent and then compare the results of two methods with the baseline obtained by fixing the output kernel to the identity. For this multi output regression task, we adopt Mean Square Error (MSE) as evaluation metric. We also take the execution time as a factor of comparison.

As for the model selection, for OKL method, the solution is computed for 30 logarithmically spaced values of the regularization parameter λ in the range

$$\lambda \in [10^{-5}\alpha, \alpha], \qquad \alpha = \sqrt{||\mathbf{Y}^T\mathbf{K}\mathbf{Y}||_2} \tag{8}$$

For low rank OKL, the solution is computed for each pair of value of the rank parameter $p = 1, \cdots, m$ and the same value of λ above.

The results of parameters tuning show that: For OKL, when $\lambda = 13.093$, MSE for validation set reaches the minimum = 18.982; for low rank OKL, when $p = 12$, $\lambda = 28.966$, MSE for validation set reaches the minimum = 18.788.

Comparing Training Time

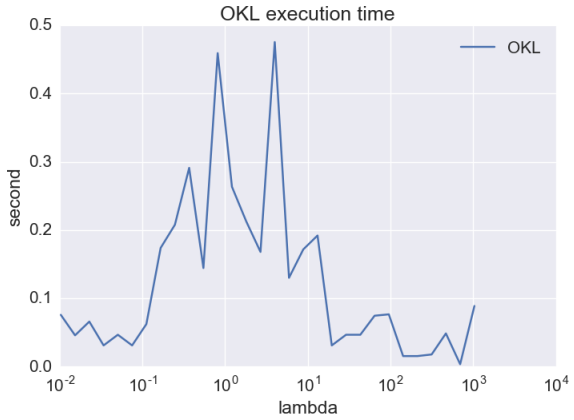

Figure 1: OKL training time as a function of λ

We can observe from Figure 1 that for OKL with block coordinate descent, most of the computation time is spent in correspondence with intermediate values of λ. Later we will see that these values of λ also correspond to the best reconstruction performances.

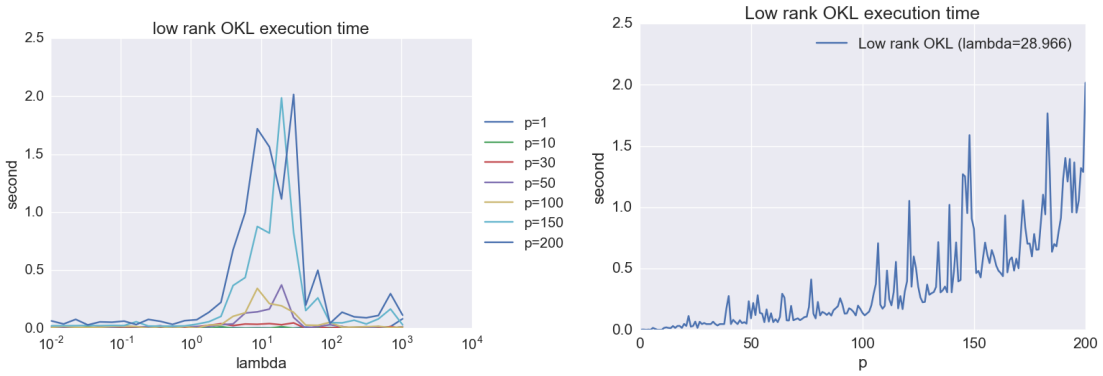

Figure 2: Low rank OKL training time as a function of λ and p

As seen from the left panel of Figure 2, we choose several values of rank $p = 1, 10, 30, 50, 100, 150, 200$ and plot the training time in function of λ. For each fixed p, we have the same pattern of the influence of λ. Moreover, the "peak pattern" is more obvious when p is larger. From the right panel of Figure 2 where the λ is the optimal choice, we can conclude that the execution time has a tendency of augmentation with the increase of p.

Comparing Figure 2 with Figure 1, we find that training time in correspondence with the best low rank model ($p = 12$, $\lambda = 28.966$) is reduced by more than an order of magnitude with respect to the full rank model ($\lambda = 13.093$).

Comparing MSE

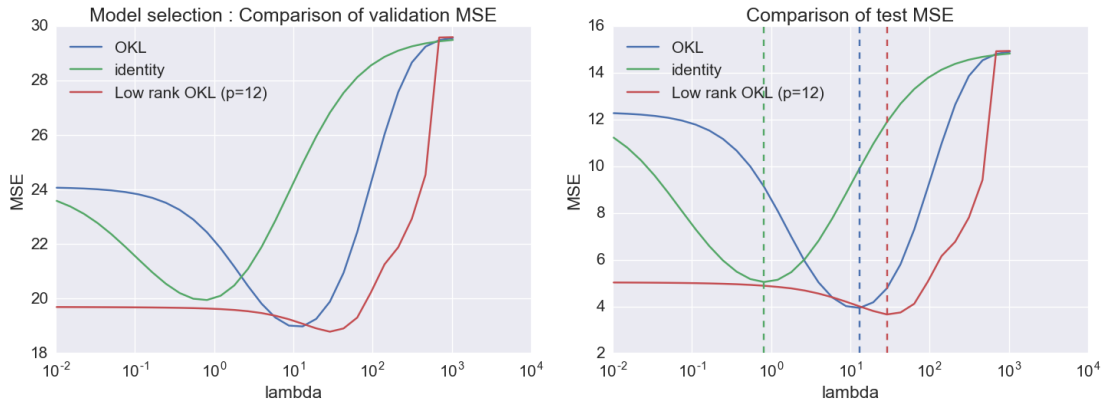

Figure 3: MSE as a function of λ for validation set (left panel) and test set (right panel, where dotted line shows the position of best-tuned λ). Green line: baseline, blue line: OKL, red line: low-rank OKL

The left panel of Figure 3 shows that the OKL with block coordinate descent yields better results than the baseline, when the value of λ is wisely selected. And the low rank OKL outperforms the other two methods in general case. The right panel shows that the best-tuned parameter λ from validation set still performs well in the test set.

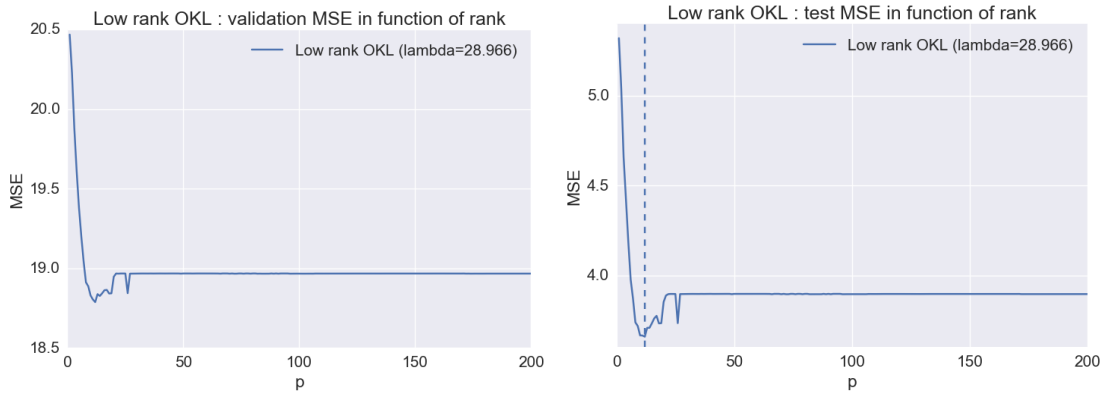

Figure 4: MSE for low rank OKL in function of p for validation set (left panel) and test set (right panel, where dotted line shows the position of best-tuned p)

The left panel of Figure 4 shows that the prediction error descends sharply when p increases form 1 to 10, but then remains stable until the largest value $p = m$. If we take the training time into consideration, it is wise to choose a smaller rank value around 10. The right panel shows that the best-tuned parameter p from validation set still performs well in the test set.

In conclusion, in terms of prediction error, the OKL and low rank OKL have similar accuracy. But if training time is considered, the low rank OKL performs better than OKL.

4.3 Digit Recognition

USPS digits dataset[3]refers to numeric data obtained from the scanning of handwritten digits from 0 to 9. The features are pixel representation of size 16×16. We turn the labels of each digit to a vector of size 10, where each position represent a component. Then this task is turned into a multi-class classification with output dimension $m = 10$.

We use the standard training/test split of USPS digits, with the rate $4 : 1$, and then apply a Gaussian kernel on the raw pixel representation. After implementing the OKL method, we evaluate the prediction effect with the metric of accuracy. As for the model selection, the solution is computed for 30 logarithmically spaced values of the regularization parameter λ in the same range as (8). Finally, according to the output kernel $\mathbf{L}$, visualization of similarities of output components can be shown.

For this task, we do not use low-rank OKL. Since the output space is already low dimensional ($m = 10$), there is no need to reduce the rank to improve training efficiency. In fact, experiments show that the low-rank OKL in this case will massively reduce the accuracy of prediction.

For OKL, when $\lambda = 0.065$, test accuracy reaches the maximum $= 0.981$.

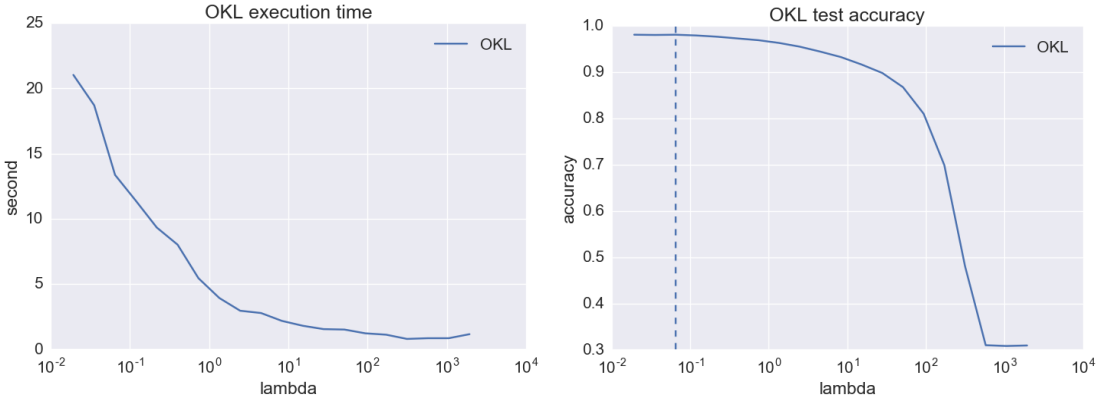

Figure 5: OKL training time (left panel) and accuracy (right panel) as a function of λ, where dotted line shows the position of best-tuned λ)

We observe from Figure 5 that when λ becomes smaller, the accuracy improves but the training time also sharply increases.

As seen from the right panel of Figure 5, when λ is small enough to reach a threshold, the accuracy will remain at a high value. To explain this observation, We consider that When λ is small enough, the regularization is very weak. Learning results will perfectly

[3]`http://riemenschneider.hayko.at/vision/dataset/task.php?did=96`

perform on training set but may not be suitable for test set, which leads to the overfitting issue in general case. But in our task, we assume that the test dataset is very similar to training set, because the shape of digits is simplex enough. As a result, the overfitting issue does not exist in this task.

Now we can illustrate the relationship between output components according to the value of output kernel matrix **L**.

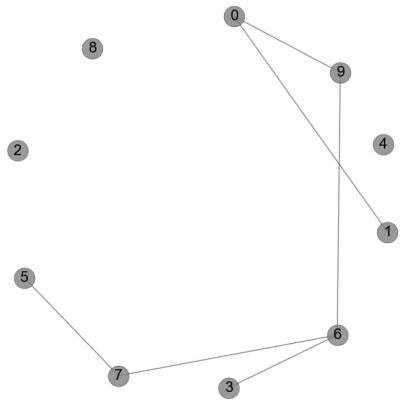

Figure 6: Interpreting the output kernel, by visualizing edges associated with off diagonal entries of **L** with largest value

Note that, since we use the Gaussian kernel on the pixels, the local information in the image is lost. It follows that the learned the class similarities are 'blind' to shape information. Classes that have correlated 'digit' regions are deemed similar.

In conclusion, the accuracy achieved (98.1%) is comparable to published results and the visualization can reveal the similarity between components.

4.4 Image Classification

Caltech 101[4] [7] and Caltech 250[5] [8] are typically used to perform multi-class image classification. In Caltech 101, pictures of objects belong to 101 categories. There are about 40 to 800 images per category. Most categories have about 50 images. The size of each image is roughly 300 x 200 pixels. In Caltech 256, pictures of objects belong to 256 categories, which increases the difficulty of the task. The number of images of the smallest category is increased to about 80. In the following, experiments are done with images from Caltech 101.

[4] `http://www.vision.caltech.edu/Image_Datasets/Caltech101/`
[5] `http://www.vision.caltech.edu/Image_Datasets/Caltech256/`

Rather than using raw pixels, we use HOG and SIFT to extract local features from images.

Histograms of Oriented Gradients (HOG) We follow the method introduced in [3], which consists of dividing the image into regions, for each region accumulating a local 1-D histogram of local gradient directions over the pixels of the region. A histogram corresponds to a local descriptor.

Scale-Invariant Feature Transform (SIFT) As described in [9], this method is composed of the following steps. First we build a pyramid of images convolved with Gaussian filters at different scales, and then the differences of successive Gaussian-blurred images. Potential keypoints are taken from the local extrema of the difference-of-Gaussians pyramid. Second, we adjust the keypoints to more accurate positions by interpolation using the quadratic Taylor expansion of the Difference-of-Gaussians function taking the candidate keypoints of the previous step as origins. At the same time, we discard keypoints that are either unstable, low-contrast or on edges. Third, we assign an orientation to each keypoint by computing a histogram of oriented gradients in the neighboring pixels. The orientation of the highest peak is assigned. If other peaks are within 80% of the highest peak, new keypoints at the same position with those orientations will be created. Last, we build a local feature descriptor for each keypoint by creating 16 histograms, taking the orientation of the previous step into account; each histogram is built upon the Gaussian-weighted oriented gradients of neighboring pixels. By concatenating 16 histograms, we obtain a local feature descriptor.

Once we have the features, we class the images by using a classifier. In this case, we try OKL and low rank OKL.

Comparing Training Time

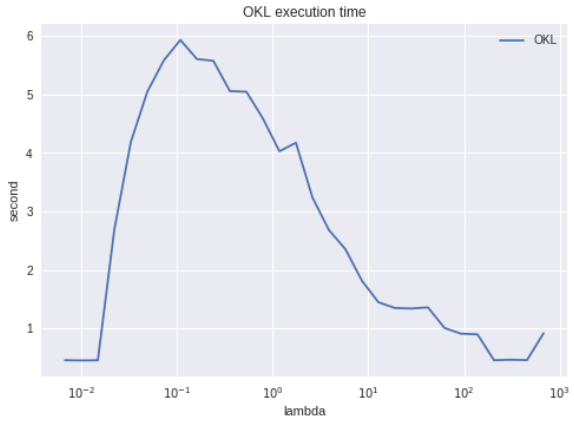

Figure 7: OKL training time as a function of λ

We can observe from Figure 7 that for OKL with block coordinate descent, the maximum of execution time appears when the best accuracy score is achieved for $\lambda = 0.109$. As the value of λ progressively goes far from 0.109, execution time decreases.

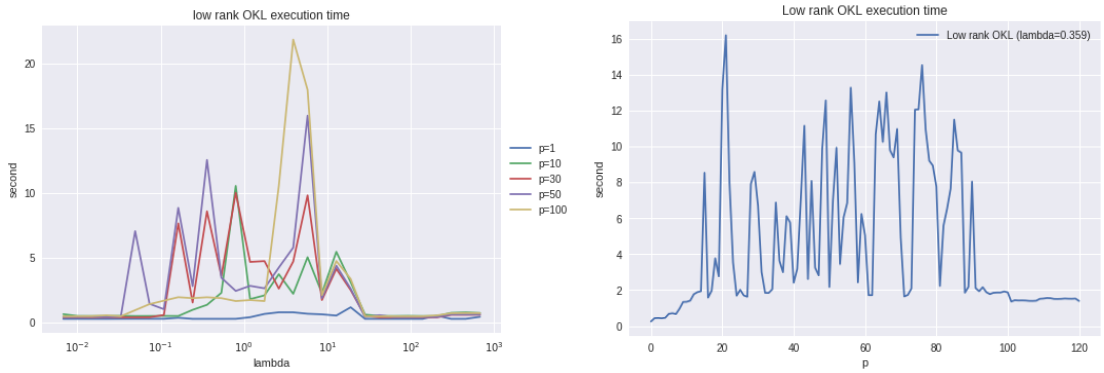

Figure 8: Low rank OKL training time as a function of λ and p

As seen from the left panel of Figure 8, we choose several values of rank $p = 1, 10, 30, 50, 100$ and plot the training time in function of λ. For each fixed p, we have nearly the same pattern of the influence of λ. Moreover, the "peak pattern" is more obvious when p is larger. From the right panel of Figure 2 where the λ is the optimal choice, we can observe that the peaks of the execution time appear irregularly when $p < 100$ and then maintain at a low level when then value of p exceeds the output dimension (101 in this case).

Comparing Figure 8 with Figure 7, we find that training time in correspondence with the best low rank model (p = 100, lambda = 0.359) is shorter than that of the best rank model ($\lambda = 0.109$) even for the same accuracy score (0.45).

Comparing accuracy score

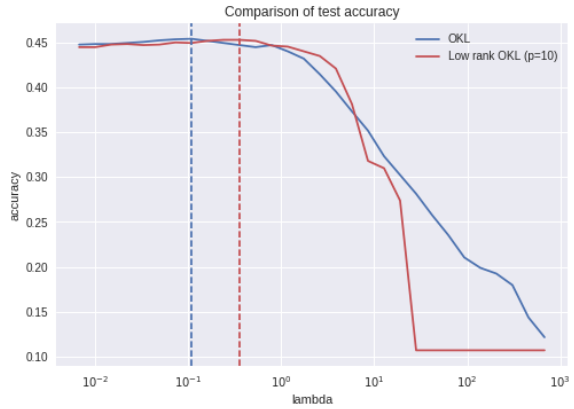

Figure 9: Accuracy score as a function of λ for test set (where dotted line shows the position of best-tuned λ).

Figure 9 shows that, OKL and LROKL with block coordinate descent yield almost the same results (accuracy = 0.45) if the value of λ is wisely selected. When the value of λ is small enough, both the results given by OKL and LROKL are comparable and satisfying. As for LROKL, if the value of λ is large enough the accuracy score sharply decreases and then remains unchanged at a low level.

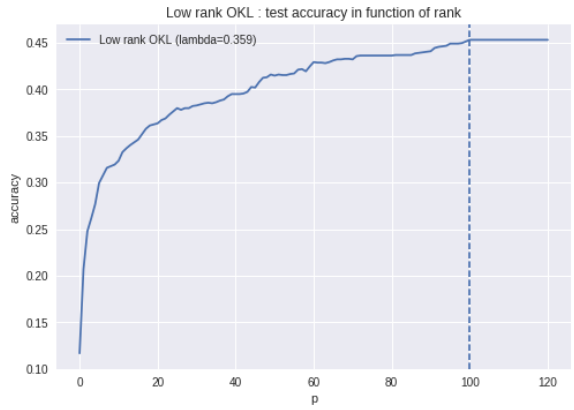

Figure 10: Accuracy score for low rank OKL as a function of p for test set (where dotted line shows the position of best-tuned p)

Figure 10 shows that, as the value of p grows, accuracy score first increases at a large rate and then slowly reaches its maximum when p gets close to 101 (the output dimension). Nevertheless, the accuracy score remains unchanged if p exceeds 101.

To conclude, LROKL and OKL manifest similar performance in the classification of Caltech101 except for the execution time, since the output dimension is not large enough

to benefit from the amelioration of LROKL.

5 Conclusion

In the report, we gave an overview of the output kernel learning problems and two methods to learn simultaneously a vector-valued function and a kernel between the output's components. The first method is formulated as an optimization problem and can be solved by an efficient block-wise coordinate descent algorithm. The second one adds a constraint on the output kernel rank and may be useful when the dimensionality is high. By comparing the two methods on several learning tasks - multi-output regression, digit recognition, multi-class image classification - we showed that both of them are satisfying and yield similar performance in general, but if the execution time is considered, low rank OKL performs better than OKL in high dimensional cases. In the task of digit recognition, we visualized that the learned output kernel encodes the relationship between the output components as claimed.

References

[1] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[2] A. Caponnetto, C. A. Micchelli, M. Pontil, and Y. Ying. Universal multi-task kernels. *Journal of Machine Learning Research*, 9(Jul):1615–1646, 2008.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[4] F. Dinuzzo and K. Fukumizu. Learning low-rank output kernels. In *ACML*, pages 181–196, 2011.

[5] F. Dinuzzo, C. S. Ong, G. Pillonetto, and P. V. Gehler. Learning output kernels with block coordinate descent. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 49–56, 2011.

[6] T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6(Apr):615–637, 2005.

[7] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1):59–70, 2007.

[8] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[10] P. Pardalos, J. R. Birge, D.-Z. Du, C. Floudas, J. Mockus, H. Sherali, and G. Stavroulakis. *Nonconvex Optimization and Its Applications*, volume 1. Springer, 1994.