Transformation Learning Via Kernel Alignment

Andrew Howard and Tony Jebara
Columbia University, New York, NY 10027 USA
{ahoward,jebara} @cs.columbia.edu

Abstract

This article proposes an algorithm to automatically learn useful transformations of data to improve accuracy in supervised classification tasks. These transformations take the form of a mixture of base transformations and are learned by maximizing the kernel alignment criterion. Because the proposed optimization is nonconvex, a semidefinite relaxation is derived to find an approximate global solution. This new convex algorithm learns kernels made up of a matrix mixture of transformations. This formulation yields a simpler optimization while achieving comparable or improved accuracies to previous transformation learning algorithms based on maximizing the margin. Remarkably, the new optimization problem does not slow down with the availability of additional data allowing it to scale to large datasets. One application of this method is learning monotonic transformations constructed from a base set of truncated ramp functions. These monotonic transformations permit a nonlinear filtering of the input to the classifier. The effectiveness of the method is demonstrated on synthetic data, text data and image data.

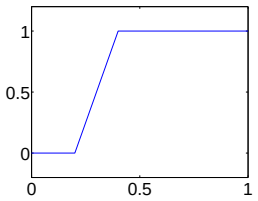
1 Introduction

In the past decade, large margin classifiers based on the support vector machine formulation have been successful in applied domains [4, 15]. A large part of this success can be attributed to the use of explicit nonlinear transformations or implicit mappings defined by reproducing kernels. These mappings are useful since finding a linear decision boundary in the new space is easy and mimics finding a nonlinear boundary in the original space. The mappings extend the applicability of linear classifiers which then can handle highly nonlinear datasets. An open question and the subject of much research is how to pick an appropriate transformation or kernel? Although there are various heuristics and rules of thumb, more principled approaches that learn the kernel from data are potentially more promising [1, 17, 14, 6, 13, 2, 5, 19, 20, 18, 12].

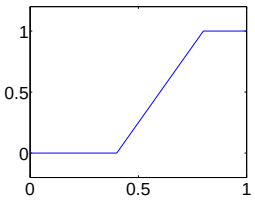
This paper investigates a data driven method to learn transformations that extends the kernel learning paradigm and offers an alternative to the previously proposed large margin transformation learning methods [8, 9]. Transformation learning was originally proposed to learn monotonic transformations [8]. Whereas kernel learning methods typically learn a kernel made up of a mixture of base kernels, $k(\vec{x},\vec{x}') = \sum_i m_i k_i(\vec{x},\vec{x}')$, transformation learning learns a mixture of transformations $\Phi(\vec{x}) = \sum_i m_i \phi_i(\vec{x})$ as depicted in Figure 1. This shows two truncated ramp functions in 1(a) and 1(b) being combined into a simple piecewise linear monotonic function in 1(c). Figure 2(a) shows a mixture of kernels based on the ramp functions and Figure 2(b) shows the kernel defined by the mixture of transformations. In this case, the mixture of kernels defines a kernel with a very clear flaw compared to the mixture of transformations and is a case where a mixture of transformations can more correctly encode the function of interest than the mixture of kernels.

In this paper, transformations will be learned by maximizing the kernel alignment score of the mixture of transformations kernel $k(\vec{x},\vec{x}') = \sum_{ij} m_i m_j \phi_i(\vec{x})^\top \phi_j(\vec{x}')$. This leads to a nonconvex optimization which is subsequently relaxed to yield a convex semidefinite program (SDP). This program learns a matrix mixture of transformations with kernels defined as $k(\vec{x},\vec{x}') = \sum_{ij} M_{ij} \phi_i(\vec{x})^\top \phi_j(\vec{x}')$. If the mixing matrix M is diagonal, this approach reduces to the usual mixture of kernels framework [1] and if M is rank one, this approach reduces to the mixture of transformations case. Otherwise, the matrix mixture is a strict generalization of the two cases.

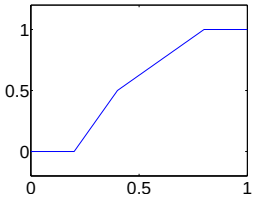
The paper is organized as follows. Section 2 starts with a review of the transformation learning framework and large margin transformation learning algorithms. Section 3 reviews kernel alignment and algorithms for maximizing the alignment to the ideal kernel made from the outer product of the labels. In Section 4, a framework for learning transformations via kernel alignment is explored. A straight forward generalization of kernel learning to transformation learning via alignment is first proposed and a convex relaxation to this nonconvex problem is derived. Because this



(a) $\phi_1(x)$

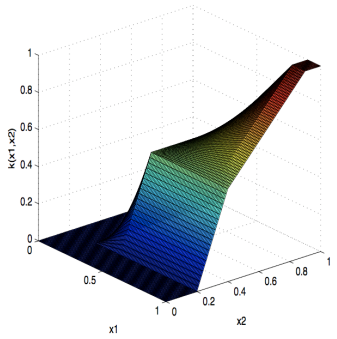


(b) $\phi_2(x)$

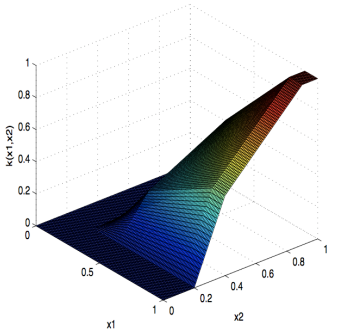


(c) $\Phi(x) = \frac{\phi_1(x)+\phi_2(x)}{2}$

Figure 1. Two truncated ramp functions (a) and (b) are combined to form a piecewise linear monotonic function (c).



(a) $\frac{\phi_1(x_1)^\top \phi_1(x_2)+\phi_2(x_1)^\top \phi_2(x_2)}{2}$



(b) $\frac{(\phi_1(x_1)+\phi_2(x_1))^\top (\phi_1(x_2)+\phi_2(x_2))}{4}$

Figure 2. Kernel defined by a mixture of kernels (a) and kernel defined by a mixture of transformations (b). Note the mixture of kernels has an unnecessary bump in the middle.

method does not perform well in practice, another more efficient and accurate method is proposed and an associated convex relaxation is derived. Section 5 demonstrates the usefulness of this new technique for learning monotonic transformations on a synthetic data set, on image histogram classification, and on text classification. Section 6 closes with a discussion.

2 Transformation Learning

Transformation learning was originally introduced in [8] in order to learn monotonic transformations to improve classification accuracy with support vector machines (SVMs). These transformations are similar to the nonlinear squashing functions in neural networks and are estimated in an adaptive way to improve the margin of the resulting classifier. The framework was developed further as a general transformation learning method in [9]. Algorithms were developed that simultaneously learn a mixture of transformations and a large margin hyperplane classifier. Transformations $\Phi(x) = \sum_i m_i \phi_i(x)$ were composed of a mixture of base transformations $\phi(x)$. The initial nonconvex optimization problem proposed was to augment a standard SVM with the mixture of transformations:

$$\begin{aligned} \min_{\vec{w},b,\vec{\xi},\vec{m}} \quad & \|\vec{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i\left(\left\langle \vec{w}, \sum_{j=1}^{J} m_j\phi_j(\vec{x}_i)\right\rangle + b\right) \geq 1-\xi_i \;\forall i \\ & \vec{\xi}\geq\vec{0}, \vec{m}\geq\vec{0}, \vec{m}^\top\vec{1}\leq 1 \end{aligned} \tag{1}$$

where $\vec{w}$ is the hyperplane normal, b is the bias term, $\vec{\xi}$ are the slack variables (as in the standard linear soft margin SVM). An alternating optimization scheme was used to find a local minimum.

Despite strong performance, the above algorithm was susceptible to local minima and often required restarts. This motivated the following convex relaxation to find global ap-

proximate solutions:

$$\begin{aligned} \min_{M,t,\lambda,\vec{\nu},\vec{\delta}} \quad & t \\ \text{s.t.} \quad & \begin{pmatrix} Y\sum_{ij} M_{ij}\phi_i^\top\phi_j Y & (\vec{1}+\vec{\nu}-\vec{\delta}+\lambda\vec{y}) \\ (\vec{1}+\vec{\nu}-\vec{\delta}+\lambda\vec{y})^\top & t-2C\vec{\delta}^\top\vec{1} \end{pmatrix} \succeq 0 \\ & \begin{pmatrix} 1 & \vec{m}^\top \\ \vec{m} & M \end{pmatrix} \succeq 0 \\ & \vec{m} \geq \vec{0}, \vec{m}^\top\vec{1} \leq 1, \sum_i M_{ii} \leq 1, \vec{\nu} \geq \vec{0}, \vec{\delta} \geq \vec{0}. \end{aligned} \tag{2}$$

where X is a matrix of feature vectors, $\vec{y}$ is a vector of the labels, Y is a matrix with $\vec{y}$ on the diagonal and zero elsewhere, t is a dummy variable to optimize, $\vec{\nu}$ and $\vec{\delta}$ and λ are dual variables and the mixing matrix M combines transformations into a kernel. We sometimes denote $\phi_i = \phi_i(X)$ for brevity when the meaning is clear from the context. The resulting kernel is defined as the matrix mixture of transformations: $k(x,\tilde{x}) = \sum_{ij} M_{ij}\phi_i(x)^\top\phi_j(\tilde{x})$. This representation generalizes both the mixture of kernels which is recovered when M is diagonal and the mixture of transformations when M is rank 1. In practice this convex relaxation performs better than the nonconvex formulation and always gives the same unique global optimum. One key shortcoming of the method is its ability to scale to large datasets since the large margin criterion is computationally cumbersome. Instead, it is possible to consider a simpler criterion such as kernel target alignment which may provide *dramatic computational improvements* without any significant reduction in generalization performance. This permits transformation learning methods to apply to large scale datasets.

3 Kernel Alignment

Kernel alignment was originally proposed in [6] as a measure of similarity between kernel matrices. The alignment between two kernels is defined as:

$$A(K_1, K_2) = \frac{\langle K_1, K_2 \rangle_F}{\|K_1\|_F \|K_2\|_F}. \tag{3}$$

This can be seen as a similarity score based on the cosine of the angle between K_1 and K_2 in an appropriate space. For arbitrary matrices, this score ranges between -1 and 1. However, since kernel alignment is only measured using positive semidefinite Gram matrices, the score is lower bounded by 0.

The alignment can give a measure of the quality of a kernel by comparing it to an idealized kernel made up of the outer product of observed labels $\vec{y}\vec{y}^\top$:

$$A(K, \vec{y}\vec{y}^\top) = \frac{\langle K, \vec{y}\vec{y}^\top \rangle_F}{\|K\|_F \|\vec{y}\vec{y}^\top\|_F} = \frac{\langle K, \vec{y}\vec{y}^\top \rangle_F}{N\|K\|_F}. \tag{4}$$

This score defines a useful quantity to optimize when searching for an optimal kernel:

$$\begin{aligned} \max_K \quad & A(K, \vec{y}\vec{y}^\top) \\ \text{s.t.} \quad & K \in \mathcal{K}. \end{aligned} \tag{5}$$

Instead of maximizing the normalized score in (5), an alternative equivalent optimization [11] can be derived that can be handled by standard solvers depending on the class of kernel being considered.

$$\begin{aligned} \max_K \quad & \langle K, \vec{y}\vec{y}^\top \rangle_F \\ \text{s.t.} \quad & \text{trace}(A) \leq 1 \\ & \begin{pmatrix} A & K^\top \\ K & I_N \end{pmatrix} \succeq 0 \\ & K \in \mathcal{K}. \end{aligned} \tag{6}$$

If $\mathcal{K}$ is the set of all positive definite matrices, this optimization yields a trivial solution of $K = \frac{c}{n}\vec{y}\vec{y}^\top$. This indicates that it is important to constrain K to avoid such a degenerate solution. The multiple kernel learning framework uses the constraint $K = \sum_i m_i K_i$ forcing the solution to involve a linear combination of base kernels. This problem then becomes the following semidefinite program:

$$\begin{aligned} \max_{\vec{m}} \quad & \left\langle \sum_i m_i K_i, \vec{y}\vec{y}^\top \right\rangle_F \\ \text{s.t.} \quad & \text{trace}(A) \leq 1 \\ & \begin{pmatrix} A & K^\top \\ K & I_N \end{pmatrix} \succeq 0 \\ & \sum_i m_i K_i \succeq 0. \end{aligned} \tag{7}$$

If the weights $\vec{m}$ are constrained to be positive, the SDP simplifies further into a quadratically constrained quadratic program (QCQP):

$$\begin{aligned} \max_{\vec{m}} \quad & \vec{m}^\top q \\ \text{s.t.} \quad & \vec{m}^\top S\vec{m} \leq 1 \\ & \vec{m} \geq \vec{0} \end{aligned} \tag{8}$$

where $q_i = \langle K_i, \vec{y}\vec{y}^\top \rangle_F$ and $S_{ij} = \langle K_i, K_j \rangle_F$. We next consider the use of the kernel target alignment criterion for learning transformations instead of simply learning linear combinations of kernels.

4 Transformation Learning Via Alignment

In this section, the method of kernel alignment is generalized to estimate an optimal set of transformations. A

direct formulation, however, leads to a nonconvex optimization problem which may be plagued by local minima. Therefore, a convex relaxation is introduced. We first use the standard approach to derive an alignment optimization for learning transformations by constraining the norm. Subsequently, we will derive a transformation alignment problem with constraints on the mixing components rather than on the norm of the learned kernel. The latter method will be demonstrated to scale better and perform more accurately in practice.

4.1 Norm Constrained Alignment

First consider learning a mixture of transformations by maximizing the alignment score to the labels where the kernel is defined as $K = \sum_{ij} m_i m_j \phi_i(x)^\top \phi_j(x)$ and optionally $m_i \geq 0$. This can be phrased as a similar optimization to (5) but with a mixture of transformations rather than a mixture of kernels as follows:

$$\begin{aligned} \max_{m} \quad & A\left(\sum_{ij} m_i m_j \phi_i(x)^\top \phi_j(x), \vec{y}\vec{y}^\top\right) \\ \text{s.t.} \quad & \vec{m} \geq \vec{0}. \end{aligned} \tag{9}$$

We can now apply the general alignment optimization derived in (6) to the task of learning a mixture of transformations:

$$\begin{aligned} \max_{m} \quad & \sum_{ij} m_i m_j \langle \phi_i(X)^\top \phi_j(X), yy^t \rangle \\ \text{s.t.} \quad & \text{trace}(A) \leq 1 \\ & \begin{pmatrix} A & \sum_{ij} m_i m_j \phi_j^\top \phi_i \\ \sum_{ij} m_i m_j \phi_i^\top \phi_j & I_N \end{pmatrix} \succeq 0 \\ & \vec{m} \geq \vec{0} \end{aligned} \tag{10}$$

Unfortunately, it is clear that this optimization is nonconvex. Consider convexifying it using a standard semidefinite relaxation [16]:

$$\begin{aligned} \max_{M} \quad & \sum_{ij} M_{ij} \langle \phi_i^\top \phi_j, yy^t \rangle \\ \text{s.t.} \quad & \text{trace}(A) \leq 1 \\ & \begin{pmatrix} A & \sum_{ij} M_{ji} \phi_j^\top \phi_i \\ \sum_{ij} M_{ij} \phi_i^\top \phi_j & I_N \end{pmatrix} \succeq 0 \\ & \begin{pmatrix} 1 & \vec{m}^\top \\ \vec{m} & M \end{pmatrix} \succeq 0, \vec{m} \geq 0. \end{aligned} \tag{11}$$

This relaxation is a semidefinite program which can be handled by standard interior point solvers. The learned kernel is $K = \sum_{ij} M_{ij} \phi_i(x) \phi_j(x)$ and defines a matrix mixture of transformations. Detailed derivations can be found in Appendix A.1. Unfortunately, the above semidefinite program scales with the number of kernels as well as the number of datapoints in the classification problem which prevents it from handling large datasets. We next consider another regularization scheme for recovering the kernel K which will lead to a slightly different semidefinite program which does not scale with the number of datapoints and therefore can handle large datasets.

4.2 Mixing Weight Constrained Alignment

Optimizing the kernel alignment $A(K, \vec{y}\vec{y}^\top)$ in (5) can be seen as maximizing a similarity score $\langle K, \vec{y}\vec{y}^\top \rangle_F$ subject to a complexity constraint based on the norm $\|K\|_F = 1$. Other types of complexity control could be used in this setting. We now investigate simple constraints on the mixing weights of the kernel $\vec{m} \geq \vec{0}$ and $\vec{m}^\top \vec{1} \leq 1$:

$$\begin{aligned} \max_{m} \quad & \left\langle \sum_{ij} m_i m_j \phi_i^\top \phi_j, \vec{y}\vec{y}^\top \right\rangle_F \\ \text{s.t.} \quad & \vec{m} \geq \vec{0}, \vec{m}^\top \vec{1} \leq 1. \end{aligned} \tag{12}$$

This nonconvex problem can be convexified via standard techniques into the following semidefinite relaxation:

$$\begin{aligned} \max_{m} \quad & \left\langle \sum_{ij} M_{ij} \phi_i^\top \phi_j, \vec{y}\vec{y}^\top \right\rangle_F \\ \text{s.t.} \quad & \begin{pmatrix} 1 & \vec{m}^\top \\ \vec{m} & M \end{pmatrix} \succeq 0 \\ & \vec{m} \geq \vec{0}, \vec{m}^\top \vec{1} \leq 1, \sum_i M_{ii} \leq 1. \end{aligned} \tag{13}$$

The derivation can be found in Appendix A.2. The method has better space and time complexity than either norm constrained transformation learning or large margin transformation learning. While the previous two methods have semidefinite blocks that scale with the number of data points and require storing all $N \times N$ kernel matrices in memory, this formulation requires neither and can precompute the most burdensome quantity with only matrix-vector and vector-vector multiplication:

$$\left\langle \sum_{ij} M_{ij} \phi_i^\top \phi_j, \vec{y}\vec{y}^\top \right\rangle_F = \sum_{ij} M_{ij} \text{trace}\left((\vec{y}^\top \phi_i^\top)(\phi_j \vec{y})\right).$$

Thus, a non-trivial semidefinite program for transformation learning is derived which remains extremely fast on large scale datasets.

5 Experiments

In this section we investigate the empirical effectiveness of the proposed alignment based transformation learning

algorithms. The newly proposed norm constrained (NC-TLA) (11) and mixing weight constrained (MWC-TLA) (13) transformation learning algorithms are compared to standard kernel alignment (KA) (8) and to large margin transformation learning (LMTL) (2). In [9], large margin transformation learning performed best overall when compared to many other kernel and transformation learning methods. We investigate the same problems and use the large margin transformation learning method as the state of the art baseline.

The experiments use truncated ramp functions as base transformations in order to learn piecewise linear monotonic transformations of the input data. These ramp functions are defined as:

$$\phi_j(x) = \begin{cases} 0 & x \leq z_j \\ \frac{x - z_j}{z_{j+1} - z_j} & z_j < x < z_{j+1} \\ 1 & z_{j+1} \leq x \end{cases} \quad (14)$$

where z_j is the j'th knot defined a priori at the quantiles. These truncated ramp functions are depicted in Figure 1. Positivity constraints on the mixing weights enforce monotonicity of the final learned transformations. The kernels defined for the kernel alignment algorithm are simply inner products of each transform with itself $k_j(x, \tilde{x}) = \phi_j(x)^\top \phi_j(\tilde{x})$. All experiments were repeated 10 times using cross-validation and test results were averaged. The optimization problems were solved in Matlab with Yalmip and SeDuMi.

5.1 Synthetic Experiment

This experiment demonstrates the methods' abilities to recover a simple monotonic transformation of the data. Six hundred two dimensional data points were sampled near a hyperplane with margin. These data points were transformed with three different monotonic transformations: the linear, logarithmic and quadratic functions. Clearly, if the inverse transformation is recovered, a linear hyperplane will separate the data perfectly. The data was split evenly into sets of 200 training, 200 cross validation, and 200 testing points. The inverse transformation to be recovered are the linear, exponential and square root functions.

Table 1 summarizes the results using the four algorithms. The kernel alignment and the norm constrained transformation learning via alignment performed poorly on this task. This is mainly because both methods penalize transformations with large norm and end up favoring some truncated ramp functions more than others. The large margin transformation learning and mixing weight constrained transformation learning via alignment performed similarly. MWC-TLA had a small edge on LMTL and performed better than or equivalently on all three experiments. Remarkably, the performance of MWC-TLA required dramatically less compute time and was orders of magnitude faster.

	x	$\exp(x)$	$\sqrt{x}$	Avg
LMTL	0.20	0.20	**0.35**	0.25
KA	12.55	11.95	12.20	12.23
NC-TLA	6.75	6.30	8.65	7.23
MWC-TLA	**0.15**	**0.05**	**0.35**	**0.18**

Table 1. Percent testing error rates for the synthetic experiments. Data was sampled near a linear decision surface and then transformed with a monotonic function. The inverse of the transformation is then learned in order to achieve accurate linear classification. The Norm Constrained Transformation Learning via Kernel Alignment (NC-TLA) and Mixing Weight Constrained Transformation Learning via Kernel Alignment (MWC-TLA) were compared to Large Margin Transformation Learning (LMTL) and Kernel Alignment (KA). Results for each dataset (the three target monotonic functions, linear, exponential and square root) are reported in their respective columns as well as the average in the Avg column. The results are averaged over 10 runs and the best error rate for each experiment is in bold.

5.2 Image Histogram Classification

A real-world experiment involving images from the Corel image dataset was then investigated. Four categories of animals were chosen from the images: (1) eagles, (2) elephants, (3) horses, and (4) tigers. Each category consists of 100 color images. These images are represented as RGB histograms following the standard binning strategy suggested in [3, 7]. In previous work, it was shown that learned monotonic transformations were superior to kernels such as the RBF and polynomial kernel in [3]. All six pairwise classification tasks were solved with a split of 160 examples in the training set, 20 examples in the cross validation and 20 in the testing set. These experiments were repeated 10 times and the results averaged and reported in Table 2.

The Norm Constrained Transformation Learning via Kernel Alignment and standard Kernel Alignment algorithms again perform poorly. The Mixing Weight Constrained Transformation Learning via Alignment algorithm had equivalent error rates on five out of six datasets as the Large Margin Transformation Learning and outperformed LMTL on the sixth one. This again demonstrates an effec-

	1/2	1/3	1/4	2/3	2/4	3/4	Avg
LMTL	**2.5**	**1.0**	**0.5**	**1.0**	2.0	**1.0**	1.34
KA	3.5	1.0	3.5	3.5	8.5	6.5	4.42
NC-TLA	3.0	2.0	2.0	4.0	5.0	7.5	3.92
MWC-TLA	**2.5**	**1.0**	**0.5**	**1.0**	**1.5**	**1.0**	**1.25**

Table 2. Percent testing error rates on Corel image histogram dataset. Four classes of animals, (1) eagles, (2) elephants, (3) horses, and (4) tigers were used in 6 pair-wise classification tasks. The average error rate over all experiments is reported in the Avg column. The Norm Constrained Transformation Learning via Kernel Alignment (NC-TLA) and Mixing Weight Constrained Transformation Learning via Kernel Alignment (MWC-TLA) were compared to Large Margin Transformation Learning (LMTL) and Kernel Alignment (KA). Results for each pair-wise classification are reported in their respective columns as well as the average in the Avg column. The results are averaged over 10 runs and the best error rate for each experiment is in bold.

tive technique that performs comparably to the previously proposed LMTL. However, the performance is achieved with remarkably better running times which have very little dependence on the training set size.

5.3 Document Classification

In another real-world experiment, the WebKB dataset was considered. Therein, webpages from four universities are split into multiple categories. We considered classification of the four largest categories which consists of 1641 student webpages, 1124 faculty webpages, 930 course webpages, and 540 project webpages. Each page was processed to remove HTML tags and converted into a bag of words representation (i.e. each document is simply a vector of word counts). In the experiments, 80 percent of the data was used for training, 10 percent for cross validation and 10 percent for testing.

In [10, 7] it was shown that preprocessing the word counts with a square root transformation improved classification accuracy. In [9], Large Margin Transformation Learning was shown to perform as well as the square root transformation and better than other transformation and kernel learning techniques. We compare Mixing Weight Constrained Transformation Learning via Alignment to Large Margin Transformation Learning. Norm Constrained Transformation Learning via Alignment could not be run on this dataset because the $N \times N$ kernel matrices were too large to store in memory. LMTL also suffers from this problem so an extragradient based algorithm was used which was proposed in [9] and computes kernels as they are needed as opposed to storing them in memory. Results are reported in Table 3. MWC-TLA performs better than LMTL on 3 out of 6 experiments and a little worse overall. The new alignment based transformation learning can solve this medium sized problem without having to resort to a specialized solver and achieved comparable results to LMTL.

	1/2	1/3	2/3	1/4	2/4	3/4	Avg
LMTL	**1.32**	2.03	4.51	1.13	**4.35**	2.80	**2.69**
MWC-TLA	1.80	**1.82**	4.44	**1.09**	5.51	**2.71**	2.90

Table 3. Percent testing error rates for the WebKB bag of words webpage classification. Four classes of webpages, (1) student, (2) faculty, (3) course, and (4) projects were used in all 6 pair-wise classification tasks. The average error rate is reported in the Avg column. The Mixing Weight Constrained Transformation Learning via Kernel Alignment (MWC-TLA) was compared to Large Margin Transformation Learning (LMTL). Results for each pair-wise classification are reported in their respective columns as well as the average in the Avg column. The results are averaged over 10 runs and the best error rate for each experiment is in bold.

5.4 Empirical Running Time Comparisons

We now compare the empirical running time of Large Margin Transformation Learning, Norm Constrained Transformation Learning via Alignment, and Mixing Weight Constrained Transformation Learning via Alignment. The synthetic experiment from Section 5 was varied over several training sizes for learning the square root transformation. At 800 training points LMTL runs out of memory using a 32 bit version of Matlab. Results for the experiments are found in Figure 3. The running time for the alignment-based algorithms include the running time for the alignment SDP as well as the running time for a simple dual SVM implemented with quadratic programming and Mosek.

Previous sections show that MWC-TLA is a viable alternative to LMTL in terms of accuracy. The empirical running time results show that it could be considered as a replacement based on its favorable scaling. The alignment SDP only scales with the number of transformations and not the data points as opposed to LMTL and NC-TLA which scale with both the data points and the number of transformations.

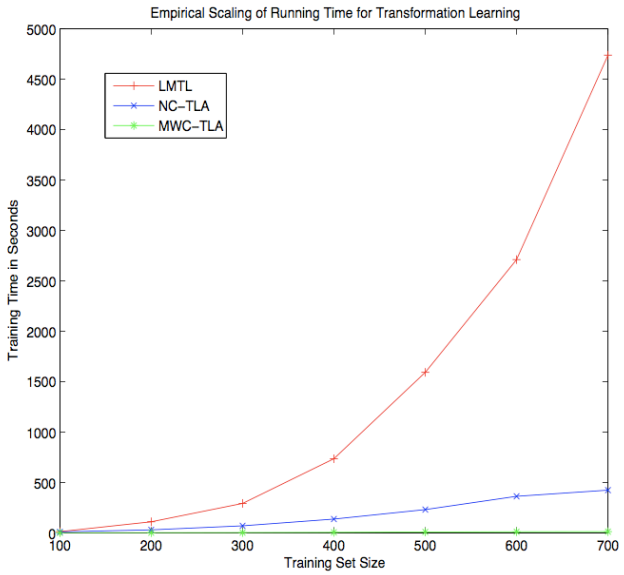


Figure 3. Empirical scaling results for Large Margin Transformation Learning (LMTL), Norm Constrained Transformation Learning via Alignment (NC-TLA) and Mixing Weight Constrained Transformation Learning via Alignment (MWC-TLA).

Thus, extremely large datasets could be used since, clearly, the SDP has no dependence on the number of training samples in the dataset.

6 Discussion

In this paper we presented two algorithms for learning transformations by optimizing the kernel alignment. The initial optimizations were nonconvex so appropriate convex relaxations were derived to find globally optimal solutions. It was shown empirically that the direct generalization of kernel alignment to transformation learning, Norm Constrained Transformation Learning via Alignment, did not perform well and the Mixing Weight Constrained Transformation Learning via Alignment was far superior. The MWC-TLA algorithm was shown to have comparable accuracy to LMTL but has far better scaling properties. These results suggest that MWC-TLA can be used in place of LMTL and can be readily applied to large scale datasets. An extragradient algorithm has been developed for LMTL in order to overcome the poor scaling in terms of number of data points, it however does not help much in terms of scaling with the number of transformations. Even with the extragradient algorithm, kernels need to be cached or computed on the fly. MWC-TLA has the advantage of only needing to precompute kernels and does not needed to keep them in memory.

A Derivation of Transformation Learning via Alignment Optimizations

A.1 Norm Constrained Transformation Learning via Alignment

In order to derive an optimization to maximize the alignment of a mixture of transformations with unit norm we first recall a general optimization result from [11]. Maximizing the alignment of a kernel to the label kernel as follows

$$\begin{aligned} \max_{K} \quad & A(K, \vec{y}\vec{y}^\top) \\ \text{s.t.} \quad & K \in \mathcal{K}, \end{aligned} \tag{15}$$

can be solved via the semidefinite program:

$$\begin{aligned} \max_{K} \quad & \left\langle K, \vec{y}\vec{y}^\top \right\rangle_F \\ \text{s.t.} \quad & \text{trace}(A) \leq 1 \\ & \begin{pmatrix} A & K^\top \\ K & I_N \end{pmatrix} \succeq 0 \\ & K \in \mathcal{K}. \end{aligned} \tag{16}$$

We can now apply this general alignment optimization to the task of learning a mixture of transformations:

$$\begin{aligned} \max_{m} \quad & \sum_{ij} m_i m_j \langle \phi_i^\top \phi_j, yy^t \rangle \\ \text{s.t.} \quad & \text{trace}(A) \leq 1 \\ & \begin{pmatrix} A & \sum_{ij} m_i m_j \phi_j^\top \phi_i \\ \sum_{ij} m_i m_j \phi_i^\top \phi_j & I_N \end{pmatrix} \succeq 0 \\ & \vec{m} \geq \vec{0}. \end{aligned}$$

Unfortunately, this optimization is nonconvex due to the quadratic interaction of the mixing weights but can be convexified using standard semidefinite relaxations. First, a matrix variable M replaces quadratic terms involving m and is defined element-wise as $M_{ij} = m_i m_j$. This yields:

$$\begin{aligned} \max_{M} \quad & \sum_{ij} M_{ij} \langle \phi_i^\top \phi_j, yy^t \rangle \\ \text{s.t.} \quad & \text{trace}(A) \leq 1 \\ & \begin{pmatrix} A & \sum_{ij} M_{ij} \phi_j^\top \phi_i \\ \sum_{ij} M_{ij} \phi_i^\top \phi_j & I_N \end{pmatrix} \succeq 0 \\ & M - \vec{m}\vec{m}^\top = 0 \\ & \vec{m} \geq \vec{0}. \end{aligned}$$

This is equivalent and still nonconvex due to the equality constraint $M - \vec{m}\vec{m}^\top = 0$. This can be relaxed to a linear matrix inequality $M - \vec{m}\vec{m}^\top \succeq 0$ which can be converted into a semidefinite constraint via the Schur complement lemma:

$$M - \vec{m}\vec{m}^\top \succeq 0 \iff \begin{pmatrix} 1 & \vec{m}^\top \\ \vec{m} & M \end{pmatrix} \succeq 0.$$

Substituting this result produces the following SDP:

$$\begin{aligned} \max_{M} \quad & \sum_{ij} M_{ij} \langle \phi_i^\top \phi_j, yy^t \rangle \\ \text{s.t.} \quad & \text{trace}(A) \leq 1 \\ & \begin{pmatrix} A & \sum_{ij} M_{ji} \phi_j^\top \phi_i \\ \sum_{ij} M_{ij} \phi_i^\top \phi_j & I_N \end{pmatrix} \succeq 0 \\ & \begin{pmatrix} 1 & \vec{m}^\top \\ \vec{m} & M \end{pmatrix} \succeq 0, \vec{m} \geq \vec{0}. \end{aligned}$$

A.2 Mixing Weight Constrained Transformation Learning via Alignment

The mixing weight constrained alignment problem can be derived in a similar way. The original nonconvex optimization can be set up as:

$$\begin{aligned} \max_{\vec{m}} \quad & \left\langle \sum_{ij} m_i m_j \phi_i^\top \phi_j, \vec{y}\vec{y}^\top \right\rangle_F \\ \text{s.t.} \quad & \vec{m} \geq \vec{0}, \vec{m}^\top \vec{1} \leq 1, \sum_i m_i^2 \leq 1, \forall i. \end{aligned}$$

This includes an additional redundant constraint $\sum_i m_i^2 \leq 1$ that will is useful in the relaxation. Introducing the positive semidefinite matrix M as in the previous subsection yields the desired convex relaxation:

$$\begin{aligned} \max_{\vec{m}} \quad & \left\langle \sum_{ij} M_{ij} \phi_i^\top \phi_j, \vec{y}\vec{y}^\top \right\rangle_F \\ \text{s.t.} \quad & \begin{pmatrix} 1 & \vec{m}^\top \\ \vec{m} & M \end{pmatrix} \succeq 0 \\ & \vec{m} \geq \vec{0}, \vec{m}^\top \vec{1} \leq 1, \sum_i M_{ii} \leq 1. \end{aligned}$$

B Acknowledgments

This work was supported in part by the National Science Foundation under Grant IIS-0347499.

References

[1] F.R. Bach, G.R.G. Lanckriet, and M.I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[2] O. Bousquet and D. Herrmann. On the complexity of learning the kernel matrix. In *Advances in Neural Information Processing 15*, 2003.

[3] O. Chapelle, P. Hafner, and V.N. Vapnik. Support vector machines for histogram-based classication. *Neural Networks, IEEE Transactions on*, 10:1055–1064, 1999.

[4] C. Cortes and V.N. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[5] K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. In *Advances in Neural Information Processing 14*, 2002.

[6] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. On kernel target alignment. Technical report, NeuroColt 2001-099, Royal Holloway University London, 2001.

[7] M. Hein and O. Bousquet. Hilbertian metrics and positive definite kernels on probability measures. In *Artificial Intelligence and Statistics (AISTATs)*, 2005.

[8] A. Howard and T. Jebara. Learning monotonic transformations for classification. In *Advances in Neural Information Processing Systems 20*, 2008.

[9] A. Howard and T. Jebara. Large margin transformation learning. *Journal of Machine Learning Research*, To Appear, 2009.

[10] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.

[11] G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[12] D.P. Lewis, T. Jebara, and W.S. Noble. Nonstationary kernel combination. In *International Conference on Machine Learning*, 2006.

[13] C.S. Ong, A.J. Smola, and R.C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 2005.

[14] A. Rakatomamonjy, F.R. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, 2007.

[15] B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, 2002.

[16] N.Z. Shor. Quadratic optimization problems. *Tekhnicheskaya Kibernetika*, 1:128–139, 1987.

[17] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[18] K. Tsuda, S. Akaho, and K. Asai. The EM algorithm for kernel matrix completion with auxiliary data. *Journal of Machine Learning Research*, 4:67–81, 2003.

[19] C.K.I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

[20] C.K.I. Williams and C.E. Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing*, 1995.