Rank Ordering Constraints Elimination with Application for Kernel Learning

Ying Xie,[a] **Chris H.Q. Ding,**[b,a] **Yihong Gong,**[c] **Zongze Wu**[d]
[a]Anhui University, Hefei, China; [b]University of Texas Arlington, Texas, USA;
[c]Xian Jiaotong University, Xian, China; [d]Guangdong University of Technology, Guangzhou, China
yingxie@126.com, chqding@uta.edu, ygong@xjtu.edu.cn, zzwu@scut.edu.cn

Abstract

A number of machine learning domains, such as information retrieval, recommender systems, kernel learning, neural network-biological systems etc, deal with importance scores. Very often, there exist some prior knowledge that could help improve the performance. In many cases, these prior knowledge manifest themselves in the rank ordering constraints. These inequality constraints are usually very difficult to deal with in optimization. In this paper, we provide a slack variable transformation methods, which effectively eliminates the rank ordering inequality constraints, and thus simplify the learning task significantly. We apply this transformation in kernel learning problem, and also provide an efficient algorithm to solved the transformed system. On seven datasets, our approach reduces the computational time by orders of magnitudes as compared to the current standard quadratically constrained quadratic programming(QCQP) optimization approach.

Introduction

A number of data mining and machine learning domains deal with importance scores relevant to rank ordering. Very often, there exist some prior knowledge that could help improve the performance.

In information retrieval (IR)(Manning, Schuetze, and Raghavan 2009), the relevance scores determine the ranking of retrieved documents. If some prior knowledge exist that says certain documents must be more relevant than certain other documents, than this partial rank ordering knowledge should be incorporated in the IR system, maybe in a form of constraints if the system uses an optimization framework to compute the relevance scores.

In recommender systems(Jannach et al. 2010), the final rating scores are used to rank the recommendation list to a user. If some prior knowledge are known that certain movies should be ranked higher than certain other movies, this prior knowledge should be incorporated into the system.

In kernel learning(Lanckriet et al. 2004), in the popular framework of learning the weights of a linear combination of input kernels, there are sometimes prior knowledge that should incorporated into the system. For example, spectral kernel learning optimizes the weights of eigenvector-based kernels so that to maximize the final kernel's alignment(Cristianini et al. 2001b; 2001a) with the class information in the data.

In neural network or biological systems, very often some parts play more important role than others. For example, in human eyes, the primary visual cortex (V1) is considered most important part that provides spatial features for human recognition (deep learning are motivated by simple-cell and complex cells populated in V1). This knowledge should be utilized in the learning system.

In many cases, these prior knowledge manifest themselves in the rank ordering constraints(Zhu et al. 2005; Hoi, Lyu, and Chang 2006). These are inequality constraints, and the generally very difficult to deal with in optimization. This is partly the reason that incorporating the prior knowledge in the form of rank ordering constraints are not widely used so far. Note also that rank ordering constraints also occur in other contexts, for example in cost-sensitive SVM (Lee and Scott 2010).

In this paper, we provide a slack variable transformation methods, that effectively eliminates the rank ordering inequality constraints, and simplify the learning task significantly. We apply this transformation in kernel learning problem, and also provide an efficient algorithm to solved the transformed system. On seven datasets, we demonstrate the slack variable transformation approach reduces the computational time by orders of magnitudes as compared to the current standard quadratically constrained quadratic programming(QCQP) optimization approach.

Rank Ordering Constraints

Let $\alpha_1, \cdots, \alpha_m$ be the relevance score of documents in information retrieval system, or rating score of items in recommender system, or the weights of kernels in linear combination of kernels for kernel learning, etc. Sometimes, we have some **prior knowledge** and wish to impose the rank ordering constraints(Zhu et al. 2005).

$$\alpha_i \geq \alpha_{i+1},\ i = 1, \cdots, m-1. \tag{1}$$

Typically, the learning system (IR system, recommender system, kernel learning system, etc) determines these importance score by optimizing a target function

$$\min_{\alpha} \quad J(\alpha) \tag{2}$$

$$s.t. \quad \alpha_i \geq 0, \quad i = 1, \cdots, m \tag{3}$$

$$\alpha_i \geq \alpha_{i+1}, \; i = 1, \cdots, m-1. \tag{4}$$

The nonnegative constraints $\alpha_i \geq 0$ in Eq.(2) enforce the fact that importance scores should naturally be non-negative. Nonnegative constraints can be efficiently handled partially due to the advancement of nonnegative matrix factorization methodologies. (Lee and Seung 2001; Ding, Li, and Jordan 2010).

It is well-known that the rank ordering inequality constraints Eq.(4) are generally much harder to handle. Generally, the number of constraints is several hundreds up to thousands or much larger in today's big data environment. Thus there is urgent need to develop efficient computational methods to handle these rank ordering constraints.

More general rank ordering constraints

We solve a more general rank ordering constraints(Hoi, Lyu, and Chang 2006).

$$\alpha_i \geq \sigma \alpha_{i+1}, \; i = 1, \cdots, m-1. \tag{5}$$

where $\sigma \geq 1$. If $\sigma = 1$, this reduces to the standard rank ordering constraints Eq.(1). For larger σ, for example $\sigma = 1.2$, this gives a series of constraints which systematically **discourages/discounts** the lower ranked[1] variables: This is similar to **discounted cumulative gain** (DCG) (Manning, Schuetze, and Raghavan 2009) popularly used in rank ordering measures, where the lower ranked documents are systematically discounted.

Thus the constrained optimization problem Eqs.(2,3,4) is generalized to the following problem

$$\min_{\alpha} \quad J(\alpha) \tag{6}$$

$$s.t. \quad \alpha_i \geq 0, \quad i = 1, \cdots, m \tag{7}$$

$$\alpha_i \geq \sigma \alpha_{i+1}, \; i = 1, \cdots, m-1. \tag{8}$$

The main purpose of this paper is to develop an efficient method to deal with large number of constraints.

Slack variable transformation

The main contribution of this paper is to provide a variable transformation that **eliminate** the rank ordering constraints completely. This technique reduces the computational efforts by order of magnitudes, and is appropriate for big data analysis.

Our work is motivated by the success of support vector machines (SVM). of soft-margin SVM where different classes can not be completely separated in most practical applications. Slack variables are most prominently used in to handle the inequality constraints

$$y_i(w^T x_i + b) \geq 1, \; i = 1 \cdots n$$

where $y_i = \pm 1$ is the class of data sample x_i; and w, b are the weight vector and bias of the linear classifier $y = \text{sign}(w^T x + b)$. These inequality constraints are similar to the rank ordering constraints of Eqs.(1,5).

[1] α_1 is highest ranked and α_m is lowest ranked.

In SVM, these constraints are expressed as

$$y_i(w^T x_i + b) = 1 + s_i, \; s_i \geq 0, \; i = 1 \cdots n$$

where s_i is the **slack** variable.

Slack variable transformation

Our main results is that score variables $(\alpha_1, \cdots, \alpha_m)$ in Eqs.(6,7,8) can be transformed to the slack variables $(s_1, \cdots, s_m)$, leading to significant reduction of constraints.

Theorem 1 *(A. Slack variable transformation)*
Score variables $\alpha = (\alpha_1, \cdots, \alpha_m)^T$ can be transformed in to slack variables $s = (s_1, \cdots, s_m)^T$ through the relation

$$\alpha = Cs. \tag{9}$$

(B. Rank ordering constraints elimination)
The optimization problem Eqs.(6,7,8) is transformed to

$$\min_{s} \quad J(Cs) \tag{10}$$

$$s.t. \quad s_i \geq 0, \quad i = 1, \cdots, m \tag{11}$$

where $C \in \Re^{m \times m}$ is a upper-triangular matrix:

$$C = \begin{pmatrix} 1 & \sigma & \sigma^2 & \cdots & \sigma^{m-1} \\ 0 & 1 & \sigma & \cdots & \sigma^{m-2} \\ 0 & 0 & 1 & \cdots & \sigma^{m-3} \\ \cdots & & & \cdots & \cdots \\ 0 & \cdots & 0 & 1 & \sigma^1 \\ 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \tag{12}$$

Thus the rank ordering constraints Eq.(8) are completely eliminated.

Proof of Theorem 1

We define $\{\alpha_i\}$ using $\{s_i\}$ via the following recurrence relation. Starting at $i = m$, we run backwards:

$$\begin{aligned} &\alpha_m = s_m \\ &\quad \alpha_i = s_i + \sigma \alpha_{i+1}, \quad i = m-1, m-2, \cdots, 1 \end{aligned} \tag{13}$$

Since $s_i \geq 0$, constraints $\alpha_i \geq 0, \; \alpha_i \geq \sigma \alpha_{i+1}$ are clearly satisfied.

Now we need to solve the recurrence relation Eq.(13) to obtain $\{\alpha_i\}$ as a function of $\{s_i\}$.

We work backwards

$$\alpha_{m-1} = s_{m-1} + \sigma s_m,$$

$$\alpha_{m-2} = s_{m-2} + \sigma \alpha_{m-1} = s_{m-2} + \sigma s_{m-1} + \sigma^2 s_m,$$

$$\vdots$$

$$\alpha_i = \sum_{k=i}^{m} \sigma^{k-i} s_k, \; i = m, m-1, \cdots, 1 \tag{14}$$

Thus the solution to the recurrence relation Eq.(13) is given by Eq.(14). Eq.(14) can be written using matrix - vector multiplication, as Eq.(10) with C given in Eq.(13). □

Kernel Learning

From here on in the rest of this paper, we discuss the utilization of slack variable transform to kernel learning problem.

Kernel-based algorithms explore information about pairwise similarity between data points. Often, kernels are computed from feature vectors. In some applications, there exit several kernels $\{K_i\}$ on the data. One of the popular approach is linear combination of these kernels:

$$K(\alpha) = \sum_{i=1}^{m} \alpha_i K_i, \quad \alpha_i \geq 0, \tag{15}$$

The combination weights $\{\alpha_i\}$ are determined by some criteria. This is the most basic form of **kernel learning**.

A commonly used method to build kernels K_i is to use the eigenvectors $\{v_i\}$ of the Laplacian matrix[2] $(D - W)$, where W is either known pairwise similarity or a kernel computed from feature vectors. $D = \text{diag}(d_1, \cdots, d_n), d_i = \sum_j W_{ij}$. The kernel K_i is constructed as $K_i = v_i v_i^T$.

The criteria to determine the weights α is to incorporate the available class information into the kernel $K(\alpha)$. This is done through class label matrix $S = YY^T$, where Y is the class indicator matrix: $Y_{ik} = 1$ if data sample i belongs to class k. Otherwise, $Y_{ik} = 0$.

In **kernel alignment**, the alignment between two kernels K, S is defined as

$$A(K, S) = \frac{\sum_{ij} K_{ij} S_{ij}}{\sqrt{\sum_{ij} K_{ij}^2} \sqrt{\sum_{ij} S_{ij}^2}} \tag{16}$$

We optimize α so that $K(\alpha)$ is **best aligned** with class label kernel $S = YY^T$ through

$$\begin{aligned} &\max_{\{\alpha_i\}} A(K(\alpha), YY^T) \\ &s.t.\ \alpha_i \geq 0,\ i = 1, \cdots, m \end{aligned} \tag{17}$$

This framework is called **spectral kernel alignment**.

When using eigenvectors of the Laplacian matrix, v_i is more important than v_{i+1}. i.e., K_i is more important that K_{i+1}. Thus we may impose rank order constraints of Eq.(1) or Eq.(5). The above kernel learning problem becomes

$$\begin{aligned} &\max_{\{\alpha_i\}} A(K(\alpha), YY^T) \\ &s.t.\ \alpha_i \geq 0, \quad i = 1, \cdots, m \\ &\qquad \alpha_i \geq \alpha_{i+1}, i = 1, \cdots, m-1. \end{aligned} \tag{18}$$

Our goal here is to demonstrate the usefulness of the slack variable transform in kernel learning. Thus we will restrict to the above basic kernel learning framework, not getting into more advanced methods of kernel learning. Below, we gives more discussions of the kernel learning and related literatures.

[2]In practice, we use the normalized Laplacian matrix $D^{-1/2}(D - Q)D^{-1/2}$.

Brief Discussion on Kernel Learning

Since the introduction of support vector machine, kernels play central role in machine learning. Standard methods on constructing kernels are built on pairwise distances between feature vectors x_i, x_j, such as the RBF kernel $K_{ij} = \exp(-\|x_i - x_j\|^2/\sigma)$. However, this approach does not take into account the class label information available in some applications, such as semi-supervised learning.

Kernel learning(Lanckriet et al. 2004) initially constructs a linear combination of input kernels. The weights α are optimized according to some procedures. A popular approach is to use kernel alignment(Cristianini et al. 2001b; 2001a) to maximally align the kernel with the available class labels.

In spectral kernel construction(Chapelle, Weston, and Schälkopf 2002; Kondor and Lafferty 2002; Smola and Kondor 2003), eigenvectors of the Laplacian matrix are used to construct the kernel. Here α is set to a pre-defined function from for example $\alpha_i = e^{-\mu \lambda_i}$ (Kondor and Lafferty 2002) or $\alpha_i = 1/(\lambda_i + \epsilon)$ (Zhu et al. 2005) where λ_i are the eigenvalues.

In these approaches, there are sometimes prior knowledge that should incorporated into the system. These show up as the rank ordering constraints of Eqs.(1,5).

This rank order contrained approach achieved success in several studies (Zhu et al. 2005; Hoi, Lyu, and Chang 2006; Cortes, Mohri, and Rostamizadeh 2010; Howard and Jebara 2009; Ye, Ji, and Chen 2008; Cuturi 2011; Liu et al. 2009).

However, in solving the alignment problem in semi-supervised learning case, a quadratically constrained quadratic programming(QCQP) problem is required to address an ordered constraint that generally improve the learned kernel(Zhu et al. 2005).

In practice, these methods are time consuming. QCQP method in general scales with m^3 where m is the number of constraint; Usually m is the same order of n, n is the number of data samples. This is prohibitively slow for large problems.

The slack variable transform introduced above effectively eliminates the rank order constraints, and greatly speeds up computation. We explain this in the rest of the paper.

Semi-supervised kernel learning

The most useful application of kernel learning is in semi-supervised learning area. In semi-supervised learning, the input data set contains L labeled and U unlabeled instances. Our task is to learn the class labels for the unlabeled data. Here we follow the transductive learning where both labeled data and unlabeled data are used to build the final optimal kernel. Let $n = L + U$ represents the total number of samples; typically $L \ll U$. The data are feature vectors $X_{p\times n} = \{x_1, x_2, \cdots, x_n\}$ in some input space. The label information $Y_{n\times c} = \{y_1, y_2, \cdots, y_c\}$ is defined as$\{0, 1\}$, c is the number of classes.

For presentation convenience, we decompose the class label matrix and base kernel K as:

$$Y = \begin{pmatrix} Y^L \\ Y^U \end{pmatrix}, \quad K = \begin{pmatrix} K^{LL} & K^{LU} \\ K^{UL} & K^{UU} \end{pmatrix} \tag{19}$$

Since we do not know Y^U, we can not align K with Y. Instead, we align the labeled part K^{LL}. We maximize the correlation/alignment using

$$\begin{aligned}&\max_{\{\alpha_i\}} A(K^{LL}, Y^L {Y^L}^T), \\ &\quad s.t.\ \alpha_i \geq 0,\ i = 1, \cdots, m\end{aligned} \tag{20}$$

The following can be easily derived:

Lemma 1 *For the semi-supervised kernel alignment,*

$$A(K^{LL}, Y^L {Y^L}^T) = \frac{b^T \alpha}{\sqrt{\alpha^T F \alpha}}, \tag{21}$$

where F, b are given by

$$F_{ij} = ({v_i^L}^T v_j^L)^2, \quad b_i = ({Y^L}^T v_i)^2, \tag{22}$$

and v_i is an eigenvector of length $n = L + U$, decomposed as [3]

$$v_i = \begin{pmatrix} v_i^L \\ \\ v_i^U \end{pmatrix}, \quad K^{LL} = \sum_{i=1}^m \alpha_i v_i^L (v_i^L)^T, \tag{23}$$

Lemma 2 *In Lemma 1, F is a semi-positive definite.*

Reformulation of semi-supervised kernel learning

We need to solve Eq.(20). Note that the scale of $\{\alpha_i\}$ is undetermined up to an over-all proportional constant. We can fix this constant by imposing $b^T \alpha = 1$, and minimize $\alpha^T F \alpha$ (this is desirable because F is s.d.p). Thus optimization problem of Eq.(20) is equivalent to

$$\begin{aligned}&\min_{\{\alpha_i\}} \alpha^T F \alpha, \\ &\quad s.t.\ b^T \alpha = 1 \\ &\qquad \alpha_i \geq 0,\ i = 1, \cdots, m\end{aligned} \tag{24}$$

Since the objective function is convex, and the constraints are convex, the optimization is convex and has a unique solution. We will device an efficient algorithm to solve it.

Incorporating Prior Knowledge

When using eigenvectors of the Laplacian matrix, v_i is more important than v_{i+1}. i.e., K_i is more important that K_{i+1}. Thus we may impose rank order constraints of Eqs.(1,5). The above kernel learning problem becomes

$$\begin{aligned}&\min_{\{\alpha_i\}} \alpha^T F \alpha, \\ &\quad s.t.\ b^T \alpha = 1 \\ &\qquad \alpha_i \geq 0,\ i = 1, \cdots, m \\ &\qquad \alpha_i \geq \sigma \alpha_{i+1},\ i = 1, \cdots, m-1.\end{aligned} \tag{25}$$

This problem is usually solved using QCQP approach. Our contribution is to use slack variable transform of Theorem 1, and transform the above optimization problem to

[3]Although eigenvectors $v_1, \cdots, v_m$ are mutually orthonormal, however, the labeled part $v_1^L, \cdots, v_m^L$ are not orthonormal. This seems to be ignored or ambiguous in previous work.

Theorem 2 *The optimization problem of Eq.(25) is equivalent to solving the following*

$$\begin{aligned}&\min_s s^T \tilde{F} s, \\ &\quad s.t.\ \tilde{b}^T s = 1 \\ &\qquad s_i \geq 0,\ i = 1, \cdots, m\end{aligned} \tag{26}$$

where

$$\tilde{F} = C^T F C,\ \tilde{b} = C^T b. \tag{27}$$

Once the optimal solution s^* is obtained from Eq.(23), the optimal $\alpha^* = C s^*$.

With this theorem, the difficult rank order constraints are eliminated. Thus the problem of Eq.(25) can be solved more efficiently than the usual QCQP approach.

Solution Algorithm

Here we present an efficient algorithm to solve the optimization problem Eq.(24) and Eq.(26).

The algorithm for solving Eq.(29) is the following:

```
(1) Initialize $\alpha = (1 \cdots 1)^T$; set $\alpha = \alpha/(\alpha^T b)$ .
(2) For every $i$, update $\alpha_i$ using
        $\alpha_i \leftarrow \alpha_i b_i (\alpha^T F \alpha)/(F\alpha)_i$        (28)
(3) Set  $\alpha \leftarrow \alpha/(\alpha^T b)$.
Repeat (2,3) until $\alpha$ converges.
```

This algorithm scales as $O(n^2)$ (assuming F is a dense matrix). However, because the operations involved are matrix-vector multiplications which are typically very efficiently implemented using Level-3 BLAS on multi-core processors and frequently used data are stored in Cache. Thus, this algorithm is suitable for large datasets. When storage for large matrix F becomes a bottleneck, Nyström type of approximation scheme can be implemented to save significantly amount of memory. This algorithm can be efficiently implemented on multi-core Cache-based processor, and suitable for large datasets.

Analysis of the Algorithm

We study the following issues: (1) KKT condition, (2) Correctness, (3) Feasibility, and (4) Convergence .

KKT condition. We introduce a Lagrangian multiplier λ to enforce $b^T \alpha = 1$. and another Lagrangian multiplier $\xi_i \geq 0$ to enforce $\alpha_i \geq 0$. The Lagrangian function is

$$\mathcal{L} = \alpha^\intercal F \alpha - 2\lambda \left(b^T \alpha - 1\right) - \sum_{i=1}^n \alpha_i \xi_i, \tag{29}$$

The corresponding KKT conditions are

$$\begin{aligned}\frac{\partial \mathcal{L}(\alpha)}{\partial \alpha_i} &= 2\,(F\alpha)_i - 2\lambda b_i - \xi_i = 0,\ \forall i \\ \frac{\partial \mathcal{L}(\alpha)}{\partial \lambda} &= -2\left(b^T \alpha - 1\right) = 0, \\ \xi_i \alpha_i &= 0,\ \forall i\end{aligned} \tag{30}$$

From first equation of Eq.(30), we have $\xi_i = 2(A\alpha)_i - 2\lambda b_i$. Substituting into second equation, we obtain the condition

$$[(F\alpha)_i - \lambda b_i]\,\alpha_i = 0 \tag{31}$$

From this condition, summing over i, we obtain

$$\lambda = \alpha^T F\alpha > 0, \tag{32}$$

because F is semi-positive definite. With this λ value, the computational algorithm of Eq.(28) is in fact the updating rule

$$\alpha_i \leftarrow \alpha_i \frac{\lambda b_i}{(F\alpha)_i} \tag{33}$$

Correctness. We now show the algorithm is exact, i.e., when the algorithm converges, the converged solution satisfies the KKT condition. By definition, this is the correct solution.

Theorem 3 *Using updating algorithm of Eq.(33), the converged solution satisfies the KKT condition of Eq.(30).*

Proof. At convergence, from Eq.(33), we have

$$\alpha_i^* = \alpha_i^* \lambda b_i/(F\alpha^*)_i, \tag{34}$$

This is exactly the KKT condition of Eq.(31)

Feasibility. We show that at convergence, the solution satisfies the $\sum_i \alpha_i b_i = 1$ constraint. From Eq.(34), we have $\alpha_i^* \lambda b_i = \alpha_i^* (F\alpha^*)_i$. Summing over i, we have

$$\lambda(\sum_i \alpha_i^* b_i) = \alpha^{*T} F\alpha^*. \tag{35}$$

Since $\lambda = \alpha^{*T} F\alpha^* > 0$ in most cases due to F is s.p.d., we have $\sum_i \alpha_i^* b_i = 1$, i.e., α^* is a feasible solution.

Convergence. Below, Theorem 4 shows that the Lagrangian function $\mathcal{L}(\alpha)$ monotonically decrease. Since $\mathcal{L}(\alpha)$ is bonded below due to $F \succeq 0$, this monotonicity implies convergence.

Theorem 4 *The following Lagrangian function*

$$\mathcal{L}(\alpha) = \alpha^T F\alpha - 2\lambda(\alpha^T b - 1) \tag{36}$$

monotonically decreases under the updating rule Eq.(33).

Proof is omitted due to space limitation.

Experiments

We run the above rank order constrained Kernel Alignment on several data sets(some of them are used in (Zhu et al. 2005)). These datasets are shown in Table 1. In our experiments, $KA0$ means unordered constrained kernel alignment model, $KA1$ means ordered constrained kernel alignment model. Once the kernel is learned through the semi-supervised learning, we use support vector machine (SVM) and the harmonic function (Zhu, Ghahramani, and Lafferty 2003) as the base classification methods.

Computational time comparison

Figure 1 show the average computational time, based on a PC with a 2.3GHz Intel Core Processor, 6GB memory running Windows 7, with the Matlab implementation. Clearly, the time on our proposed algorithm are order of magnitude smaller compared to the QCQP algorithm. For example on pc-mac dataset, QCQP used 12.5 seconds but our ordered alignment model(KA1) used only 0.125 seconds.

Table 1: Datasets

Dataset	# of instances	classes	dimensions
pc-mac	1943	2	600
baseball-hockey	1993	2	600
one-two	2200	2	320
odd-even	4000	2	320
Scence	2407	6	294
USPS	400	10	256
BinNumbers	1014	26	320

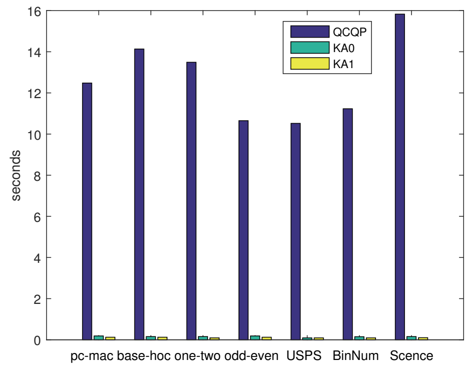

Figure 1: Time cost(seconds) on QCQP, KA0, KA1 on seven datasets (r=200, labeled size=100).

Performance on Semi-Supervised Learning

We use different number of data points as labeled data and the rest as unlabeled data, and learn the kernel. The learned kernels are used in SVM classification and harmonic function based semi-supervised learning. All results are the average of 30 runs each time using a different random set as labeled data points. The average accuracy are shown in Figure 2. We see that ordered constrained kernels consistently outperform kernels without ordering.

Summary

We propose slack variable transformation that effectively eliminates the rank order constraints and thus significantly speed the optimization of learning systems. We design an efficient algorithm to solve the transformed optimization in kernel learning/alignment problem. Experiments show that the proposed transformation combined the new efficient algorithm speeds up the computation by orders of magnitudes.

Acknowledgements. This work is supported by National Basic Research Program of China (973 Program) Grant 2015CB351705, National Natural Science Foundation of China (NSFC) Grants 61572030, 61602005, and U.S. National Science Foundation Grant NSF-DBI 1356628.

References

Chapelle, O.; Weston, J.; and Schälkopf, B. 2002. Cluster kernels for semi-supervised learning. *Advances in neural information processing systems.* 15:585–592.

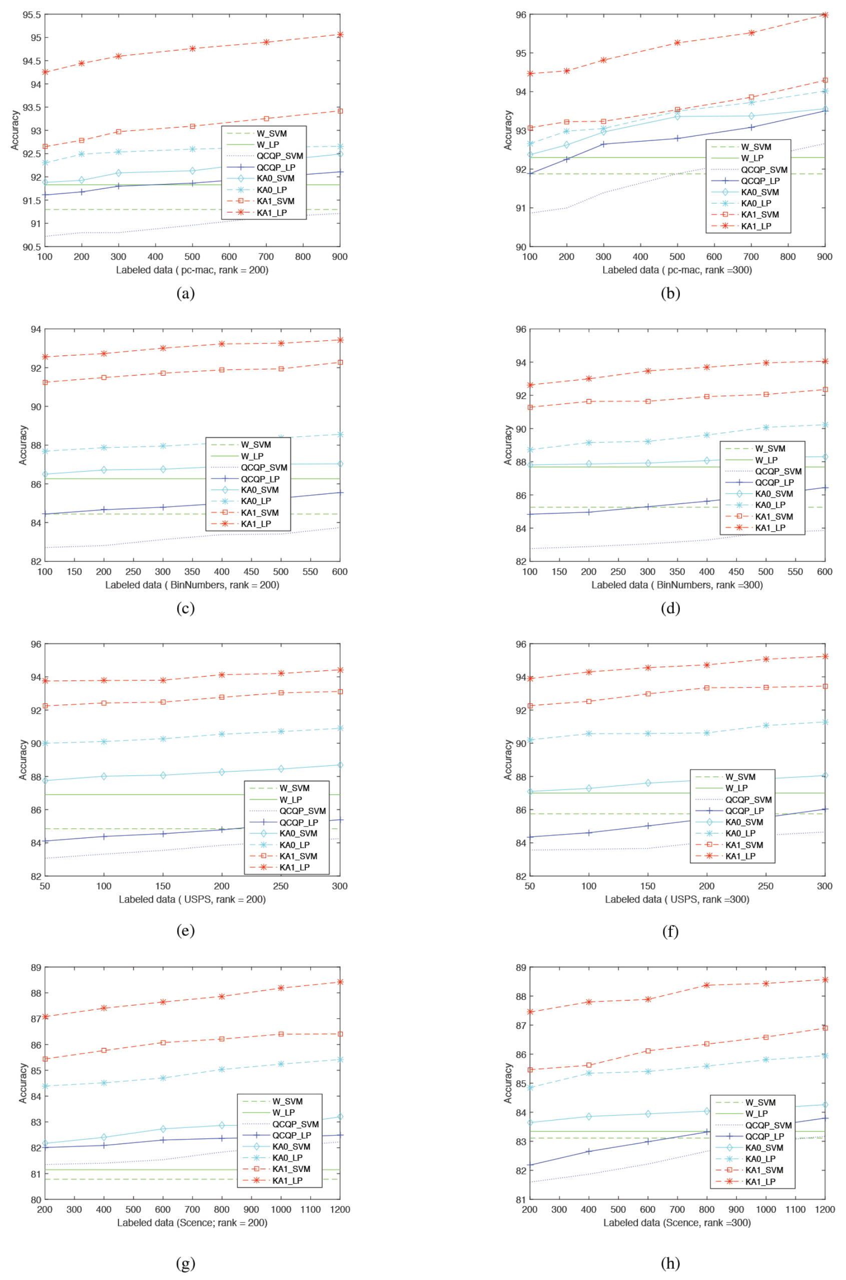

Figure 2: Classification accuracy(%) using SVM and label propagation(LP) on the learned kernels through different methods. The number of training data are shown. The rest of the data are used for testing.

Cortes, C.; Mohri, M.; and Rostamizadeh, A. 2010. Two-stage learning kernel algorithms. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 239–246.

Cristianini, N.; Kandola, J.; Elisseeff, A.; and Shawe Taylor, J. 2001a. On optimizing kernel alignment. *Royal Holloway Univeristy of London, London, UK, Technical Report NC-TR-01-087*.

Cristianini, N.; Shawe-Taylor, J.; Elisseeff, A.; and Kandola, J. S. 2001b. On kernel-target alignment. In *NIPS*, 367–373.

Cuturi, M. 2011. Fast global alignment kernels. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 929–936.

Ding, C.; Li, T.; and Jordan, M. 2010. Convex and semi-nonnegative matrix factorizations. *IEEE Trans. Pattern Analysis and Machine Intelligence*. (LBNL Tech Report 60428, 2006).

Hoi, S. C.; Lyu, M. R.; and Chang, E. Y. 2006. Learning the unified kernel machines for classification. In *Proceedings of the 12th ACM SIGKDD international conference on*

Knowledge discovery and data mining., 187–196. ACM.

Howard, A., and Jebara, T. 2009. Transformation learning via kernel alignment. In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, 301–308. IEEE.

Jannach, D.; Zanker, M.; Felfernig, A.; and Friedrich, G. 2010. *Recommender Systems: An Introduction*. Cambridge University Press.

Kondor, R., and Lafferty, J. 2002. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, volume 2, 315–322.

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

Lee, G., and Scott, C. 2010. Nested support vector machines. *IEEE Transaction on Signal Processing* 58:1648–1660.

Lee, D., and Seung, H. S. 2001. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*. Cambridge, MA: MIT Press.

Liu, W.; Qian, B.; Cui, J.; and Liu, J. 2009. Spectral kernel learning for semi-supervised classification. In *Proceedings of the International Joint Conference on Artificial Intelligence.*

Manning, C.; Schuetze, H.; and Raghavan, P. 2009. *Introduction to Information Retrieval*. Cambridge University Press.

Smola, A., and Kondor, R. 2003. Kernels and regularization on graphs. *Learning theory and kernel machines.* 144–158.

Ye, J.; Ji, S.; and Chen, J. 2008. Multi-class discriminant kernel learning via convex programming. *Journal of Machine Learning Research* 9:719–758.

Zhu, X.; Kandola, J.; Ghahramani, Z.; and Lafferty, J. 2005. Nonparametric transforms of graph kernels for semi-supervised learning. *Advances in Neural Information Processing Systems.*

Zhu, X.; Ghahramani, Z.; and Lafferty, J. 2003. Semi-supervised learning using gaussian fields and harmonic functions. *Proc. Int'l Conf. Machine Learning*.