Iteratively Divide-and-Conquer Learning for Nonlinear Classification and Ranking

OU WU, Center for Applied Mathematics, Tianjin University
XUE MAO and WEIMING HU, NLPR, Institute of Automation, Chinese Academy of Sciences

Nonlinear classifiers (i.e., kernel support vector machines (SVMs)) are effective for nonlinear data classification. However, nonlinear classifiers are usually prohibitively expensive when dealing with large nonlinear data. Ensembles of linear classifiers have been proposed to address this inefficiency, which is called the ensemble linear classifiers for nonlinear data problem. In this article, a new iterative learning approach is introduced that involves two steps at each iteration: partitioning the data into clusters according to Gaussian mixture models with local consistency and then training basic classifiers (i.e., linear SVMs) for each cluster. The two divide-and-conquer steps are combined into a graphical model. Meanwhile, with training, each classifier is regarded as a task; clustered multitask learning is employed to capture the relatedness among different tasks and avoid overfitting in each task. In addition, two novel extensions are introduced based on the proposed approach. First, the approach is extended for quality-aware web data classification. In this problem, the types of web data vary in terms of information quality. The ignorance of the variations of information quality of web data leads to poor classification models. The proposed approach can effectively integrate quality-aware factors into web data classification. Second, the approach is extended for listwise learning to rank to construct an ensemble of linear ranking models, whereas most existing listwise ranking methods construct a solely linear ranking model. Experimental results on benchmark datasets show that our approach outperforms state-of-the-art algorithms. During prediction for nonlinear classification, it also obtains comparable classification performance to kernel SVMs, with much higher efficiency.

CCS Concepts: • **Theory of computation → Models of learning**; **Semi-supervised learning**;

Additional Key Words and Phrases: Divide-and-conquer, classification, listwise learning to rank, clustering, multi-task learning

ACM Reference format:
Ou Wu, Xue Mao, and Weiming Hu. 2017. Iteratively Divide-and-Conquer Learning for Nonlinear Classification and Ranking. *ACM Trans. Intell. Syst. Technol.* 9, 2, Article 18 (October 2017), 26 pages.
https://doi.org/10.1145/3122802

1 INTRODUCTION

Kernel support vector machines (SVMs) (Cortes and Vapnik 1995) are widely used for nonlinear data classification in the machine-learning community. Although kernel SVMs often produce satisfactory classification results, it can be computationally expensive when dealing with large

This work is supported by NSFC (61379098 and 61673377).
Authors' addresses: O. Wu (Corresponding author), 6-106, Center for Applied Mathematics, Tianjin University, China, 300072; email: wuou@tju.edu.cn; X. Mao and W. Hu, 95 Zhongguncun East, Beijing, China, 100190; emails: {xmao, wmhu}@nlpr.ia.ac.cn.

https://doi.org/10.1145/3122802

datasets. The complexity of kernel SVMs relies on the number of support vectors, which grows approximately linearly with the size of the training data. On the other hand, while a linear classifier is extremely efficient (Fan et al. 2008), it cannot handle nonlinear data with acceptable accuracy since it fails to consider the underlying structures of nonlinear data (e.g., clusters and manifolds). To address this issue, ensembles of linear models have been proposed. However, these methods either lack robustness, such as the CSVM model (Gu and Han 2013), or are time-consuming, such as the SVM-KNN model (Zhang et al. 2006). In our early work (Mao et al. 2014), a divide-and-conquer method was proposed in which the training data are divided into subsets according to Gaussian mixture model (GMM) clustering and linear SVM classifiers are constructed using a multitask learning strategy for each subset.

In this article, an iteratively divide-and-conquer classification approach is proposed to better capture the intrinsic property of nonlinear data. Each iteration contains a dividing step and a conquer step. Instead of being independent of each other, these two steps are combined into a generative model and alternatively performed in each iteration. Consequently, the two steps promote each other. In the dividing step, the involved training data are partitioned into a number of clusters (or training subsets) using the GMM. In the conquer step, a basic linear classifier (e.g., a linear SVM) is trained for each cluster. In order to exploit the local manifold structure to improve clustering performance, the locally consistent regularizer (Liu et al. 2010) is incorporated into the clustering process. In order to ensure that the data points in each cluster are linearly separable, some clusters may have relatively few points, which can result in overfitting. In this work, we consider training a linear SVM classifier for a cluster as a single task and the training of the classifier ensemble as a multitask learning problem. Clustered multitask learning (Zhou et al. 2011a) is used to exploit the intrinsic relatedness between tasks and avoid overfitting for each task. We use the EM algorithm (Dempster et al. 1977) to solve for model parameters based on the maximum likelihood estimation framework. During testing (or prediction), a test instance is first mapped into a cluster and the corresponding basic classifier of the cluster is used to classify the instance. Our approach can also be utilized for semisupervised learning, which enables our model to make full use of the available unlabeled data to detect the manifold structure.

Further, the proposed approach provides a new technical path to deal with two other distinct problems: quality-aware web data classification and listwise learning to rank. These two problems can also be solved by utilizing the proposed approach with a slight extension. In quality-aware web data classification, the types of web data vary in terms of information quantity or quality. For example, some pages contain numerous texts, whereas others contain few texts; some web videos are in high resolution, whereas other web videos are in low resolution. As a consequence, the quality of extracted features from different web data may also vary greatly. Existing learning algorithms on web data classification ignore the variations of data quality. Based on the proposed approach, the quality-aware factors of web data are utilized to partition web data into subsets. In each subset, the data quality of different samples varies slightly. An ensemble of classifiers is then trained.

Listwise learning to rank (LTR) has received great attention in recent years as it is useful in many applications, such as information retrieval, data mining, natural language processing, and speech recognition (Qin et al. 2008). In listwise LTR, the input space contains a ranked list (or permutations) for a set of instances, each of which is described by the preference features. A number of listwise LTR algorithms have been proposed in previous literature. The target ranking models in most existing listwise ranking studies are linear. In our early work (Wu et al. 2016), an ensemble of linear rankers has been investigated and initial promising results have been obtained. In this work, the proposed approach is used for listwise LTR in a more effective way.

The proposed approach is based on locally consistent clustering and multitask learning (LCC-MTL). Experimental results on benchmark datasets demonstrate that the proposed approach

outperforms state-of-the-art methods. In summary, this article makes the following three main contributions:

—We propose a new iteratively divide-and-conquer classification approach (LCC-MTL) in which a new generative model is used to combine locally consistent clustering and linear classifier learning. Compared with existing models, the parameters of the model are estimated more efficiently. Multitask learning is employed to train multiple linear SVMs on nonlinear datasets to avoid overfitting. Our approach can also be used for semisupervised learning, employing the unlabeled data to obtain more effective results in dividing.
—In addition to nonlinear data classification, two novel extensions of LCC-MTL are investigated for two other learning problems. The first problem is quality-aware web data classification; the second is listwise learning to rank. Based on the proposed approach, two new algorithms, LCC-MTL$_Q$ and LCC-MTL$_R$, are obtained for the two learning problems, respectively.
—In nonlinear data classification with linear SVMs, LCC-MTL achieves much higher efficiency than kernel SVM with comparable classification performances; in web data classification, LCC-MTL$_Q$ achieves better performance than existing quality-aware web data classification algorithms; in listwise ranking, LCC-MTL$_R$ also achieves better results than two classical ranking algorithms and our early proposed method.

The rest of the article is organized as follows. Section 2 brief reviews background to this study. Section 3 introduces the methodologies of the proposed approach and the algorithm for nonlinear data classification with linear SVMs. Section 4 describes two extensions for the proposed approach in web data classification and listwist LTR. Section 5 reports experimental results. Conclusions are given in Section 6.

2 RELATED WORK

This section introduces studies that are closely related to this work.

2.1 SVMs for Nonlinear Data

Kernel SVMs are highly time-consuming in both training and classification for nonlinear data when the data size is large. A number of recent studies have been proposed to deal with this problem. The following two categories of methods attract considerable attention.

2.1.1 Learning with Linear SVMs. This method divides feature space into regions and applies an ensemble of linear SVMs to approach nonlinear classification functions. A typical approach is lazy learning: given a testing sample, a classifier is trained in a subregion of the input space near the sample and then used to classify the sample. SVM-KNN (Zhang et al. 2006) and the adaptive SVM nearest-neighbor classifier (Blanzieri and Melgani 2006) belong to this category. Since the learning process is postponed until the testing phase, these methods are inefficient during testing. In contrast, some methods construct local classifiers during the training phase (eager learning), usually employing a divide-and-conquer strategy involving two steps: partitioning the data into clusters and training a classifier for each cluster. MLSVM (Fu et al. 2010), infinite SVM (Zhu et al. 2011) and CSVM (Gu and Han 2013) fall into this category. There also exist other eager learning methods based on local coordinate coding, such as LLSVM (Ladický and Torr 2011). These methods either lack robustness, such as CSVM, or are time-consuming, such as SVM-KNN, MLSVM, and LLSVM. Most of these methods suffer from disadvantages arising from nonconvexity optimization. Oiwa and Fujimaki (2014) proposed novel convex region-specific linear models—namely, partition-wise linear models—which are based on convexity optimization.

In our early work (Mao et al. 2014), a new generative model is proposed to partition the data using GMM clustering and to train linear SVMs using multitask learning.

2.1.2 Learning with Feature Mapping. A kernel can be viewed as an implicit nonlinear feature mapping from an original into a high-dimensional space. Some existing studies utilize an explicit feature-mapping technique to approximate kernel functions. This method first maps original data into a low-dimensional feature space in which the kernel of any two samples is well approximated by its inner product. Rahimi and Recht (2007) proposed a random projection-based method to approximate shift-invariant kernels. Vempati et al. (2010) extended the work conducted by Rahimi and Recht (2007) to approximate generalized radial-basic function (RBF) kernels. Pham and Pagh (2013) proposed an effective randomized tensor product technique, called Tensor Sketching, which can approximate any polynomial kernel. Pele et al. (2013) embedded an input vector into a high-dimensional but sparse vector and constructed a linear classifier based on piecewise linear functions in the individual features and pairwise features.

The present study on nonlinear data also belongs to the first technical path. Nevertheless, the proposed approach is iterative to partition training data into subsets and then learn linear SVMs.

2.2 Quality-Aware Fusion

Biometrics refers to the automatic identification of users based on their physiologic and behavioral characteristics. Information fusion has received considerable attention in biometrics and has proved to be effective by extensive studies (Nandakumar et al. 2007). Recent studies on multimodal biometrics focus on the quality-aware fusion method because the quality of biometric data is usually negatively affected by factors such as environment, noise, devices, and physiologic or behavioral change by the user (Kittler et al. 2007). However, the factors affecting one biometric modality often doe not affect other biometric modalities. For example, if illumination variation is regarded as a degrading factor for face biometrics, it is completely irrelevant to the fingerprint modality. Therefore, evaluating the qualities of the multiple modalities of biometric data and dynamically fusing the multimodal information with quality-aware factors is investigated. A number of quality-aware fusion algorithms are developed. Poh and Kittler (2012) proposed a unified framework for quality-aware fusion of multimodal biometrics. Quality-aware fusion assumes that the classifiers in each modality are given. As a result, quality-aware fusion pursues only dynamic fusion strategies while quality-aware learning pursues both dynamic fusion and classifier parameters.

Obvious differences exist between quality-aware web data classification and quality-aware fusion in biometrics: (1) quality-aware web data classification focuses on learning classifiers, whereas quality-aware fusion focuses on fusion and assumes that classifiers are given; (2) quality-aware fusion is designed only for multimodal data, whereas the data in quality-aware classification can be single model. Our early work (Wu et al. 2014) proposed the first quality-aware learning algorithm in which information quantity and quality are considered in web data classification. The training data are partitioned into training subsets according to GMM clustering on quality-aware factors. A multitask feature-learning approach is utilized to select features on each subset. This algorithm is not iterative.

2.3 Multitask Learning

Multitask learning (MTL) is a method in which multiple related tasks are learned simultaneously to improve generalization performance. This approach has drawn widespread attention in recent years. Some methods are formulated under the regularization framework (Evgeniou and Pontil 2004; Zhou et al. 2011a), and others are based on the Bayesian model (Yu et al. 2005; Zhang and

Yeung 2010). Recently, Luo et al. (2013, 2015) introduced manifold regularization into MTL and obtained promising results in multilabel image classification.

2.4 Listwise Learning to Rank (LTR)

Listwise LTR is a crucial technique for information retrieval. Existing listwise LTR algorithms first define a loss function over a ground-truth ordering and a predicted ordering generated by the ranking function. An optimal ranking function is then trained according to the minimization of training loss. Two classical methods are ListMLE (Xia et al. 2008) and ListNet (Cao et al. 2007). The former defines a likelihood loss function based on the Packett-Luce Model (Cao et al. 2007), while the latter defines a loss function based on KL-divergence between two permutation probability distributions. The target ranking models in most listwise ranking studies are linear. There are a limited number of studies that construct nonlinear listwise ranking models that are mainly based on a decision tree. Pavlov et al. (2010) combined a sequence of boosted tree as a ranking model based on the bag strategy. Moon et al. (2010) also constructed the ranking model based on the combination of decision tree. These two algorithms still require relevance labels during training, although the labels are claimed to be unavailable for the conventional listwise setting. Our early work (Wu et al. 2016) proposed the first learning algorithm (RPC-MTL) for piecewise linear ranking models in listwise LTR. Nevertheless, the clustering step in RPC-MTL is required to preserve the rankings of instances in different objects. Therefore, the clustering procedure is quite heuristic. In this study, the learning for piecewise linear ranking model is investigated in a unified view with the other learning problems, that is, nonlinear data classification and quality-aware web data classification. The manifold structure of training data in LTR can be better captured.

3 THE PROPOSED APPROACH

In order to facilitate analysis, we introduce the following notations. Assume that there are N i.i.d. samples whose features are $X = \{x_i\}_{i=1,\ldots,N}$, where x_i represents features. The corresponding labels are $Y = \{y_i\}_{i=1,\ldots,N}$. The latent variables $Z = \{z_i\}_{i=1,\ldots,N}$ denote the assignments of samples to the K mixtures. The parameters $\mu = \{\mu_k\}_{k=1,\ldots,K}$ and $\Sigma = \{\Sigma_k\}_{k=1,\ldots,K}$ denote the centroids and covariance matrixes of Gaussian components, respectively. $W = \{w_k\}_{k=1,\ldots,K}$ represents the parameters of the K basic models (e.g., linear SVMs in nonlinear classification) for the K clusters. We denote by $\pi = \{\pi_k\}_{k=1,\ldots,K}$ the mixing coefficients of the GMM. Let $\Theta = \{\pi, \mu, \Sigma, W\}$ be the total set of parameters of our model. In quality-aware classification, let d_i be the additional quality-aware factors for sample x_i. In listwise LTR, x_i is further represented by $(x_i^1, \ldots, x_i^{n_i})$, where n_i denotes the number of objects in x_i, and the (ordering) label y_i on x_i is represented by $(y_i^1, \ldots, y_i^{n_i})$, where y_i^j is the rank assigned to the object x_i^j.

3.1 Overview of the Proposed Approach

We first use nonlinear data classification to illustrate the proposed approach. The extensions to web data classification and listwise LTR are introduced in the next section. An overview of the proposed approach LCC-MTL is presented in Figure 1. In a single divide-and-conquer learning procedure, a clustering step is utilized to partition the raw training set (X, Y) into a number of training subsets (clusters). A number of basic classifiers are subsequently trained for each training subset. This divide-and-conquer process relies heavily on the clustering results in the dividing stage. To this end, an iteratively divide-and-conquer approach is adopted. Once the learning is finished, in the prediction stage, the features (x_{test}) of a test sample are extracted. The clustering model is then used to assign x_{test} into a cluster based on cluster parameters. Finally, the cluster's corresponding classifier is applied to classify the test sample according to x_{test}.

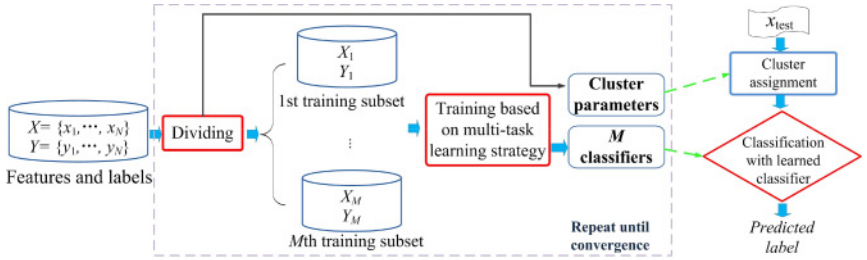


Fig. 1. Overview of the proposed approach LCC-MTL for nonlinear classification.

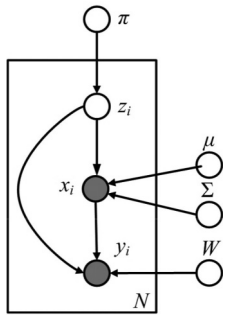


Fig. 2. The generative graphical model of LCC-MTL.

Specifically, the proposed LCC-MTL approach is iterated between two steps: partitioning the data into several clusters with a locally consistent GMM and training a basic classifier (a linear SVM in this work) in each of these clusters. We assume that the samples in each sufficiently small cluster are linearly separable. Instead of being independent of each other, the two steps promote each other: the clustering results of the GMM can improve the classification performance of the basic classifier in each cluster and vice versa. This idea has been integrated into a generative graphical model (also called LCC-MTL) as shown in Figure 2. The upper part of the generative model corresponds to the GMM, which is responsible for partitioning the input space into clusters. Given a prior probability (π) of picking a Gaussian component, a sample is first generated based on the GMM with parameters μ and Σ. The lower part is the generative process for the label y_i, which is generated based on the sample x_i and the parameter W of the function to be learned.

3.2 A New Generative Graphical Model

From Figure 1, the joint distribution over X and Y can be written as follows:

$$
\begin{aligned}
P(X,Y|\Theta) &= \prod_{i=1}^{N} P(x_i,y_i|\Theta) \\
&= \prod_{i=1}^{N}\sum_{z_i=1}^{K} \pi_{z_i} P(x_i|z_i,\mu,\Sigma) P(y_i|x_i,z_i,W) \\
&= \prod_{i=1}^{N}\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|z_i=k,\mu_k,\Sigma_k) P(y_i|x_i,w_k), \quad (1)
\end{aligned}
$$

which is obtained by summing the joint distribution of observed variables X and Y over all possible latent states of Z, with z_i taking values in $\{1, \ldots, K\}$. The mixing coefficient π_k is the prior probability of picking the kth Gaussian component. $P(x_i|z_i = k, \mu, \Sigma) = \mathcal{N}(x_i|z_i = k, \mu_k, \Sigma_k)$ is a Gaussian component of the mixture, specifying the probability of x_i conditioned on the kth component. $P(y_i|x_i, w_k)$ is the posterior probability of the ith sample output by the kth classifier. We estimate the parameters of our model by maximum likelihood estimation. The regularized log-likelihood function can be formulated as

$$\begin{aligned}\mathcal{L}(\Theta) &= \sum_{i=1}^{N} \log P(x_i, y_i|\Theta) + \Omega(W) \\ &= \sum_{i=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|z_i = k, \mu_k, \Sigma_k) P(y_i|x_i, w_k) + \Omega(W) \\ &= \sum_{i=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|z_i = k, \mu_k, \Sigma_k) P(y_i|x_i, w_k) + \Omega(W), \quad (2)\end{aligned}$$

where $\Omega(W)$ denotes a regularization term on the weight vectors of the classifiers. This term encodes prior knowledge about the K classifiers. In the following two sections, the locally consistent regularizer and multitask learning will be incorporated into the above maximum likelihood estimation framework.

3.3 Incorporating the Locally Consistent Regularizer

The first step of our model is to partition the data with a GMM. However, the standard GMM fits the data in the Euclidean space. Previous studies have shown that naturally occurring data may live on or near an underlying submanifold and the clustering performance can be greatly enhanced if the local manifold structure is exploited. Liu et al. (2010) proposed the Locally Consistent Gaussian Mixture Model (LCGMM), which smoothes the conditional probability distribution along the geodesics of the data manifold. A nearest neighbor graph is first constructed on the training data to model the nonlinear manifold structure. The edge weight matrix of the graph is defined as follows:

$$A_{ij} = \begin{cases} 1 & \text{if } x_i \in N_p(x_j) \text{ or } x_j \in N_p(x_i). \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

where $N_p(d_i)$ denotes the p nearest neighbors of d_i. Let $P_i = P(z_i|x_i, \mu, \Sigma)$ denote the distribution on z_i given x_i, μ, and Σ based on the GMM models. Let $P_i(k) = P(z_i = k|x_i, \mu, \Sigma)$. The smoothness of P_i on the graph can be measured by the following locally consistent regularizer:

$$\mathcal{R} = \frac{1}{2} \sum_{i,j=1}^{N} (D(P_i||P_j) + D(P_j||P_i))A_{ij}, \quad (4)$$

where $D(P_i||P_j) = \sum_{k=1}^{K} P_i(k) \log \frac{P_i(k)}{P_j(k)}$ is the KL-divergence between the conditional distribution P_i and P_j. The smaller the $\mathcal{R}$, the smoother the P_i over the nearest neighbor graph. In other words, if two samples are close on the manifold, their distributions over different Gaussian components are likely to be similar. The regularizer can be directly incorporated into our maximum likelihood estimation framework, which is now formulated as

$$\mathcal{L}(\Theta) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|z_i = k, \mu_k, \Sigma_k) P(y_i|x_i, w_k) + \Omega(W) - \lambda \mathcal{R}. \quad (5)$$

Since the performance is quite insensitive to the regularization parameter λ and the number of nearest neighbors p, we set them to 0.1 and 20, respectively, as in Liu et al. (2010). The locally consistent regularizer significantly enhances the clustering performance of GMM in the first step of our model.

3.4 Incorporating Multitask Learning

Our model learns the multiple predictors simultaneously rather than independently. More specifically, if training a predictor (e.g., a linear SVM) in a cluster is regarded as a task, training predictors for all clusters corresponds to a multitask learning problem[1]. This idea is inspired by the fact that in order to ensure that the samples in each cluster are linearly separable, the samples available for each cluster may be limited, which may lead to overfitting in each cluster. More important, since all the clusters are partitioned from the same dataset, they should be latently related in some way. Multitask learning can be employed to capture the intrinsic relatedness between tasks and avoid overfitting in each task.

In this article, clustered multitask learning (Jacob et al. 2008; Zhou et al. 2011a) is utilized, which assumes that tasks can be clustered into different groups and tasks from the same group have similar weight vectors. When training a classifier, we often desire a decision boundary that is smooth and has constrained curvature, since a decision boundary with arbitrary curvature would be likely to overfit the data (Ladický and Torr 2011). Clustered multitask learning is helpful as tasks in adjacent regions on the decision boundary should have similar weight vectors and should be clustered into one group. Clustered multitask learning can be formalized into the following regularizer:

$$\Omega_{MT}(W) = \sum_{r=1}^{R} \sum_{k \in \mathcal{I}_r} \|w_k - \bar{w}_r\|_2^2. \tag{6}$$

Equation (6) assumes that the total K tasks are clustered into R clusters, with the index set of the rth cluster defined as $\mathcal{I}_r = \{k | k \in \text{cluster}\, r\}$. The average weight vector of the rth cluster is denoted by $\bar{w}_r = \frac{1}{m_r} \sum_{k \in \mathcal{I}_r} w_k$, where there are m_r tasks in the rth cluster. Equation (6) measures the within-cluster variance, which requires tasks from the same cluster to have similar weight vectors.

3.5 The EM Algorithm and Implementation

A typical technique for finding maximum likelihood estimates of parameters in latent variable models is the EM algorithm. We now apply the EM algorithm to the above LCC-MTL model. Let $\Theta^{(t)} = \{\pi^{(t)}, \mu^{(t)}, \Sigma^{(t)}, W^{(t)}\} = \{\pi_k^{(t)}, \mu_k^{(t)}, \Sigma_k^{(t)}, w_k^{(t)}\}_{k=1,\ldots,K}$ denote the collection of parameters at the tth iteration.

In the tth E step, the posterior probability of assigning the ith sample to the kth linear SVM is evaluated as

$$P_i(k)^{(t)} = \frac{\pi_k^{(t)} \mathcal{N}(x_i | z_i = k, \mu_k^{(t)}, \Sigma_k^{(t)}) P(y_i | x_i, w_k^{(t)})}{\sum_{k=1}^{K} \pi_k^{(t)} \mathcal{N}(x_i | z_i = k, \mu_k^{(t)}, \Sigma_k^{(t)}) P(y_i | x_i, w_k^{(t)})} \tag{7}$$

[1]The learning tasks for different clusters are different yet related.

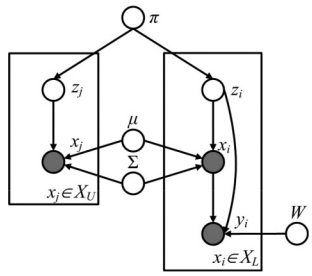


Fig. 3. The generative graphical model of semisupervised LCC-MTL.

This posterior probability is then utilized to derive the following lower bound on the log-likelihood function:

$$Q(\Theta^{(t+1)};\Theta^{(t)}) = \sum_{i=1}^{N}\sum_{k=1}^{K} P_i(k)^{(t)} \log\Big[\pi_k^{(t+1)} \cdot \mathcal{N}(x_i|z_i=k,\mu_k^{(t+1)},\Sigma_k^{(t+1)})P(y_i|x_i,w_k^{(t+1)})\Big] + \Omega(W^{(t+1)}) - \lambda\mathcal{R}, \quad (8)$$

where $\Omega(W^{(t+1)})$ is the regularization term for $W^{(t+1)}$.

In the tth M step, the parameter is updated to $\Theta^{(t+1)}$ by maximizing Equation (8). Considering the incorporated locally consistent regularizer, the GMM-related parameters $\{\pi, \mu, \Sigma\}$ are updated as follows:

$$\pi_k^{(t+1)} = \frac{N_k}{N} \quad (9)$$

$$\mu_k^{(t+1)} = \frac{1}{N_k}\sum\nolimits_{i=1}^{N} P_i(k)^{(t)} x_i - \frac{\lambda}{2N_k}\sum\nolimits_{i,j=1}^{N}\Big(\big(P_i(k)^{(t)} - P_j(k)^{(t)}\big)(x_i - x_j)\Big)A_{ij} \quad (10)$$

$$\Sigma_k^{(t+1)} = \frac{1}{N_k}\sum\nolimits_{i=1}^{N} P_i(k)^{(t)} S_{i,k} - \frac{\lambda}{2N_k}\sum\nolimits_{i,j=1}^{N}\Big(\big(P_i(k)^{(t)} - P_j(k)^{(t)}\big)(S_{i,k} - S_{j,k})\Big)A_{ij}, \quad (11)$$

where

$$N_k = \sum_{i=1}^{N} P_i(k)^{(t)} \quad \text{and} \quad S_{i,k} = \Big(x_i - \mu_k^{(t+1)}\Big)\Big(x_i - \mu_k^{(t+1)}\Big)^{\mathrm{T}}. \quad (12)$$

The update of W depends on the concrete forms of $P(y_i|x_i, w_k^{(t)})$ and $\Omega(W^{(t)})$.

3.6 Semisupervised LCC-MTL

In many applications, labeled data are expensive to collect. Semisupervised learning addresses this problem by using the abundant unlabeled data. Our model can be naturally extended to semisupervised learning as shown in Figure 3. In the left part of Figure 3, an unlabeled sample x_j is generated according to the variable z_j and the associated parameters π, μ, and Σ; in the right part, a sample x_i and its label y_i are generated with the same process as shown in Figure 2. The left and the right parts share the same parameters π, μ, and Σ. Semisupervised LCC-MTL utilizes sufficient unlabeled data X_U to capture the inherent data structures. The unlabeled dataset X_U, together with the labeled dataset X_L, Y_L, gives the information about the parameters $\{\pi, \mu, \Sigma\}$. The log-likelihood

function in Equation (5) now can be formulated as

$$\mathcal{L}(\Theta) = \sum_{x_i \in X_L} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i | z_i = k, \mu_k, \Sigma_k) P(y_i | x_i, w_k) + \sum_{x_j \in X_U} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(x_j | z_j = k, \mu_k, \Sigma_k) + \Omega(W) - \lambda \mathcal{R}, \quad (13)$$

where the first two terms are the "supervised" term based on X_L and "unsupervised" term based on X_U, respectively. Here, the locally consistent regularizer $\mathcal{R}$ is computed over both labeled and unlabeled data. The abundant unlabeled data make the locally consistent regularizer detect the local manifold structure more accurately, which leads to the improvement of classification performance shown in the experimental section. It is also possible to use a weight to balance the contributions of labeled and unlabeled data in maximum likelihood estimation.

3.7 LCC-MTL for Nonlinear Data Classification with Linear SVMs

As presented earlier, LCC-MTL is motivated by nonlinear data classification. This section introduces the concrete algorithm of LCC-MTL in nonlinear data classification with linear SVMs. Our previous work (Mao et al. 2014) has shown that it is generally a reasonable choice to cluster the tasks into groups when applying multiple linear SVMs to nonlinear datasets. We recall that training a linear SVM usually leads to the following quadratic optimization problem:

$$\min_{w} \frac{\|w\|_2^2}{2} + C \sum_{i} \ell(w; x_i, y_i), \quad (14)$$

where the first term, inversely proportional to the classifier margin, is the regularization term on the weight vector of the linear SVM and the second term is the total loss incurred. To incorporate linear SVM, we define the probability of $P(y_i|x_i, w_k)$[2] as follows:

$$P(y_i | x_i, w_k) = \exp(-\ell(w_k; x_i, y_i)), \quad (15)$$

where $\ell(w_k; x_i, y_i) = max(0, 1 - y_i \cdot w_k^T x_i)$. The value of $P(y_i|x_i, w_k)$ will be equal to 1 if the loss $\ell(w_k; x_i, y_i)$ is zero; otherwise, $P(y_i|x_i, w_k)$ will be less than 1.

Further, we define $\Omega(W) = \Omega_{MT} + ||w||_2^2$. To update the weight vectors W of the linear SVMs, we then solve the following optimization problem:

$$\max_{W^{(t+1)}} \sum_{i=1}^{N} \sum_{k=1}^{K} P(c_k | d_i)^{(t)} \log P(y_i | x_i, w_k^{(t+1)}) - \alpha \sum_{r=1}^{R} \sum_{k \in \mathcal{I}_r} \|w_k^{(t+1)} - \bar{w}_r^{(t+1)}\|_2^2 - \beta \sum_{k=1}^{K} \|w_k^{(t+1)}\|_2^2. \quad (16)$$

With the first term regarded as a weighted loss function, Equation (16) is equivalent to the clustered multitask learning problem, which can be solved using the MALSAR package (Zhou et al. 2011b). Since the linear SVMs do not change abruptly across iterations, we initialize their weight vectors with the results of the last iteration, which greatly accelerates the training.

A sketch of the algorithm is presented in Algorithm 1. K-means is utilized to initialize the mixing coefficients π, centroids μ, and covariance matrixes Σ. A linear SVM is then trained for each cluster, resulting in the initial weight vectors W.

[2]Our definition follows the definition in Sollich (2000). Because the loss function in our definition is the standard hinge loss, the value of the probability is definitely in the range of (0, 1]. Therefore, the normalization used in Sollich (2000) is not required.

ALGORITHM 1: LCC-MTL

Input: Training data $\{(x_i, y_i)|i = 1, \ldots, N\}$ and the number of clusters K, λ, $t = 1$
Output: Parameter $\Theta = \{\pi_k, \mu_k, \Sigma_k, w_k|k = 1, \ldots, K\}$
Initialize Θ by K-means and linear SVMs.
repeat
E step: Evaluate $P_i(k)^{(t)}$ using Equation (7).
M step: Reestimate $\pi_k^{(t+1)}$, $\mu_k^{(t+1)}$, and $\Sigma_k^{(t+1)}$ using Equations (9), (10), and (11), respectively.
Reestimate $w_k^{(t+1)}$ for all k simultaneously as a multitask learning problem using Equation (16).
$t = t + 1$.
until convergence
For quality-aware web data classification, train kernel SVMs for each training subset using multitask kernel SVMs (Cai and Cherkassky 2012).

We now show that each iteration of EM is guaranteed to increase the log-likelihood in Equation (5). The difference between the log-likelihood function values in two successive iterations is formulated as

$$\mathcal{L}(\Theta^{(t+1)}) - \mathcal{L}(\Theta^{(t)}) = |Q(\Theta^{(t+1)}; \Theta^{(t)}) - Q(\Theta^{(t)}; \Theta^{(t)})| - [H(\Theta^{(t+1)}; \Theta^{(t)}) - H(\Theta^{(t)}; \Theta^{(t)})], \quad (17)$$

where $H(\Theta; \Theta^{(t)}) = \sum_Z logP(Z|X, Y, \Theta)P(Z|X, Y, \Theta^{(t)})$. The first term on the right-hand side of Equation (17) is nonnegative, which is derived by the M step. The second term is less than or equal to 0 by Jensen's inequality (Hastie et al. 2001). Therefore, since the log-likelihood is nondecreasing, our algorithm is guaranteed to converge. For the real datasets in the experiments, our algorithm generally converges within 10 iterations.

During testing, a new sample (featured by x) is classified by the weighted average of the linear classifiers:

$$\sum_{k=1}^{K} \pi_k \mathcal{N}(x|z = k, \mu_k, \Sigma_k)(P(1|x, w_k) - P(-1|x, w_k)). \quad (18)$$

The sample is classified as positive if the weighted average is greater than 0, and negative otherwise. Obviously, the prediction complexity is linear in the number of tasks K. Prediction efficiency is particularly critical for large-scale or online applications. The prediction complexity of the proposed method is $O(K)$. The prediction complexity of kernel SVM is $O(N_k)$, where N_k is the number of support vectors. Because N_k is usually much larger than K, the prediction complexity of the proposed method is much lower than that of SVM. For example, in the IJCNN1 set in our experiments, K is about ten, where N_k is about 7924. The prediction complexity of SpSVM (an improved algorithm of SVM) (Keerthi et al. 2006) is also larger than that of the proposed method because the value of N_k in SpSVM is also larger than K. The prediction complexity of SVM-KNN is also $O(N_k)$. The prediction complexity of LLSVM is $O(N_a)$, where N_a is the number of anchor points. In practice, N_a is usually much larger than K. The prediction complexities of CSVM and LSVM-MTL are the same as that of the proposed method.

The training complexity of the proposed method mainly depends on the two iterative steps, that is, GMM clustering and accelerating projected gradient-based optimizing. The accelerating projected gradient-based optimizing equals the optimizing for a clustered multitask learning problem, which is borrowed from MALSAR (Zhou et al. 2011b). The accelerating projected gradient-based optimizing is efficient according to the experiments conducted in Zhou et al. (2011b). Because our

Fig. 4. Three web pages with different proportions of images and texts.

Fig. 5. Three images with different lengths of text descriptions.

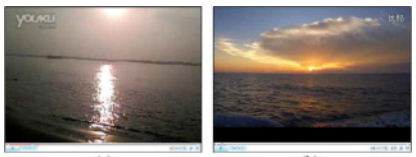


Fig. 6. Two web videos with different visual quality.

algorithm generally converges within 10 iterations on the involved datasets in the experiments, the consumption time of the training stage for the proposed method is close to that of kernel SVM.

4 TWO EXTENSIONS OF LCC-MTL

4.1 Web Data Classification with Quality-Aware Factors

We first explain why the quality should be considered in web data classification in detail. The types of web data vary in two aspects:

— *Information quantity is usually distinct.* Take web pages as an example. Some pages contain many images, whereas other pages contain few images. Some pages contain numerous texts, whereas other pages contain few texts. This phoneme still exists for images. Some web images have many text descriptions, whereas other web images have limited text descriptions. Figure 4 shows three web pages with different proportions of texts and images. In Figure 4(a), the page contains a number of images and a small amount of text; in Figure 4(c), the page contains few images but a lot of text. Figure 5 shows three examples of web images with different lengths of text description.

— *Information quality is usually distinct.* The quality of web images and videos is greatly affected by factors such as the performance of capture devices and the environment. As many web images and videos are produced by low-quality devices, they have low resolutions or distorted colors. Figure 6 illustrates how videos with similar contents differ in quality (e.g.,

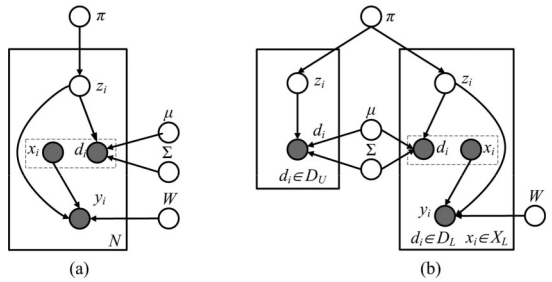


Fig. 7. The generative graphical model of LCC-MTL in quality-aware Web data classification (a) and the semi-supervised version (b).

resolution and color distortion). It is very likely that the Figure 6(a) video is obtained by a low-quality camera.

In Figure 4, image features (or text features) should make distinct contributions in the classification of Figure 4(a) and Figure 4(c) pages. Likewise, text features should make distinct contributions when classifying the three images in Figure 5. Considering that information quantity can also be viewed as a quality measure for web information, the factors related to both information quantity and quality are called *quality-aware factors*. Some typical quality-aware factors are the text length of a web document, the image count in a web page, and visual quality of a web image or video. Quality-aware factors should ideally be considered during classifier training and testing.

The proposed approach can still be applied to quality-aware web data classification[3] with a slight modification that the data partition relies solely on quality-aware factors instead of the whole features, and the SVM models rely solely on features (x_i). The modified approach and the semisupervised version are shown in Figure 7. Let D be the set of quality factors and $d_i \in D$ be the quality factors of the ith sample x_i. Equation (1) becomes

$$\begin{aligned} P(X, D, Y|\Theta) &= \prod_{i=1}^{N} P(x_i, d_i, y_i|\Theta) \\ &= \prod_{i=1}^{N} \sum_{z_i=1}^{K} \pi_{z_i} P(d_i|z_i, \mu, \Sigma) P(y_i|x_i, z_i, W) \\ &= \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(d_i|z_i = k, \mu_k, \Sigma_k) P(y_i|x_i, w_k). \end{aligned} \quad (19)$$

Theoretically, in web data classification, any type of classifiers (e.g., random forest) can be used. Because SVM has been proven effective in our early work, SVM is still used. The entire algorithmic steps are similar to those shown in Algorithm 1 by replacing x_i with d_i in Equations (7) through (11) in the E steps and M steps. The algorithm is called LCC-MTL$_Q$. Ideally, multitask kernel SVMs (Cai and Cherkassky 2012) should be used in each iteration. However, the training complexity is quite high. To accelerate the training speed, the linear SVMs are used in iteration and kernel SVMs are used in the last iteration for web data classification.

[3]Theoretically, this problem also belongs to nonlinear data classification. Nevertheless, there are two distinct differences. First, this problem contains quality-aware factors; second, the "classification accuracy" instead of the "prediction complexity" is the primary consideration.

4.2 Listwise Learning to Rank with Linear Rankers

As introduced previously, almost all existing LTR algorithms utilize linear ranking functions (rankers) to model data. Nevertheless, real-world data are usually nonlinear and our early work (Wu et al. 2016) shows that a piecewise linear ranker is more effective. In our early work, a divide-and-conquer learning process is used and a ranking-preserve clustering approach is leveraged to divide training data into training subsets. In this work, the proposed approach is used and an iterative divide-and-conquer learning algorithm is obtained for listwise LTR.

To apply the proposed approach, the joint distribution over X and Y is written as follows:

$$P(X,Y|\Theta) = \prod_{i=1}^{N} P(x_i,y_i|\Theta) \quad = \prod_{i=1}^{N} \sum_{z_i \in \{0,1,\ldots,K\}^{n_i}} P(z_i)P(x_i|z_i,\mu,\Sigma)P(y_i|x_i,z_i,W), \tag{20}$$

where $z_i^j = k$ means that x_i^j is generated according to the kth component, $P(z_i) = \prod_{j=1}^{n_i} \pi_{z_i^j}$, and

$$P(x_i|z_i,\mu,\Sigma) = \prod_{j=1}^{n_i} p(x_i^j|\mu_{z_i^j},\Sigma_{z_i^j}) \tag{21}$$

$P(y_i|x_i,z_i,W)$ is calculated according to the Packett-Luce model (Cao et al. 2007) shown as follows:

$$P(y_i|x_i,z_i,W) = \prod_{j=1}^{n_i} \frac{exp\left(W^T_{z_i^{y_i^{-1}(j)}} x_i^{y_i^{-1}(j)}\right)}{\sum_{l=j}^{n_i} exp\left(W^T_{z_i^{y_i^{-1}(l)}} x_i^{y_i^{-1}(l)}\right)}, \tag{22}$$

where $y_i^{-1}(j)$ is the instance index of the jth rank in x_i.

The lower bound on the log-likelihood function defined in Equation (8) becomes

$$\begin{aligned} Q(\Theta^{(t+1)};\Theta^{(t)}) &= E(lnP(X,Y,Z|\Theta^{(t+1)})|X,Y,\Theta^{(t)}) \\ &= \sum_{i=1}^{N}\sum_{z_i}(lnP(z_i) + lnP(x_i|z_i,\mu,\Sigma) + lnP(y_i|x_i,z_i,W^{(t+1)}))P(z_i|x_i,y_i,\Theta^{(t)}) \\ &\quad + \Omega(W^{(t+1)}) - \lambda\mathcal{R}, \end{aligned} \tag{23}$$

where $\Omega(W^{(t+1)})$ is the regularization term for $W^{(t+1)}$, and

$$P(z_i|x_i,y_i,\Theta^{(t)}) = \frac{P(x_i,z_i,y_i|\Theta^{(t)})}{P(x_i,y_i|\Theta^{(t)})}. \tag{24}$$

Let X_{inst} be the set of all instances in X. For an instance v, let $I(v)$ and $J(v)$ be the index of the sample that contains v and the index of v in the sample, respectively. By maximizing Equation (23), we obtain

$$\pi_k^{(t+1)} = N_k/N \tag{25}$$

$$\mu_k^{(t+1)} = \sum_{i=1}^{N}\sum_{j=1}^{n_i} x_i^j \tau_{i,j,k}^{(t)}/N + \frac{\lambda}{2N_k} \sum_{v,\nu \in X_{inst}} \left(\left(\tau^{(t)}_{I(v),J(v),k} - \tau^{(t)}_{I(\nu),J(\nu),k}\right)(v-\nu)\right) A_{I(v),J(v),I(\nu),J(\nu)} \tag{26}$$

$$\Sigma_k^{(t+1)} = \frac{1}{N_k}\sum_{i=1}^{N}\sum_{j=1}^{n_i}\tau_{i,j,k}^{(t)}S_{i,j,k} - \frac{\lambda}{2N_k}\sum_{v,v\in X_{inst}}\left(\left(\tau_{I(v),J(v),k}^{(t)} - \tau_{I(v),J(v),k}^{(t)}\right)\left(S_{I(v),J(v),k} - S_{I(v),J(v),k}\right)\right)A_{I(v),J(v),I(v),J(v)}, \quad (27)$$

where

$$\tau_{i,j,k}^{(t)} = P(z_i^j = k|x_i, y_i, \Theta^{(t)}) \quad (28)$$

$$N_k = \sum_{i=1}^{N}\tau_{i,j,k}^{(t)} \quad (29)$$

$$S_{i,j,k} = (x_i^j - \mu_k^{(t+1)})(x_i^j - \mu_k^{(t+1)})^{\mathrm{T}} \quad (30)$$

The parameter $W^{(t+1)}$ is obtained by maximizing

$$Q_W(\Theta^{(t+1)};\Theta^{(t)}) = \sum_{i=1}^{N}\sum_{z_i}(lnP(y_i|x_i,z_i,W^{(t+1)}))P(z_i|x_i,y_i,\Theta^{(t)}) + \lambda\Omega(W^{(t+1)}) \quad (31)$$

where $\Omega(W^{(t+1)})$ is defined by Equation (6).

The computational complexity of $P(z_i|x_i,y_i,\Theta^{(t)})$ is $O(K^{n_i})$. Therefore, when either K or n_i is large, it is impractical to calculate the exact value of $P(z_i|x_i,y_i,\Theta^{(t)})$. In this work, the Metropolis sampling method is adopted to approximately calculate the value of $P(z_i|x_i,y_i,\Theta^{(t)})$. During sampling, the proportion of the conditional probability of a new candidate sample (z'') to the previous sampled sample (z') must be calculated. Based on Equation (24), the following equation is obtained:

$$\frac{P(z''|x_i,y_i,\Theta^{(t)})}{P(z'|x_i,y_i,\Theta^{(t)})} = \frac{P(z'')P(x_i|z'',\mu^{(t)},\Sigma^{(t)})P(y_i|x_i,z'',W^{(t)})}{P(z')P(x_i|z',\mu^{(t)},\Sigma^{(t)})P(y_i|x_i,z',W^{(t)})}. \quad (32)$$

z'' can be generated according to the following process. Given z', a random number $j \in \{1,\ldots,|z'|\}$ is selected. We change the value of z'^j randomly and z'' is then obtained. Whether z'' is accepted depends on the value of $\frac{P(z''|x_i,y_i,\Theta^{(t)})}{P(z'|x_i,y_i,\Theta^{(t)})}$. Assuming that for each training sample x_i, we obtain a set of sampled $MS_i^{(t)} = \{z_{(i)}(1),\ldots,z_{(i)}(M)\}$ in the tth iteration. We can then calculate the approximate value of $P(z_i|x_i,y_i,\Theta^{(t)})$ according to MS_i and Equation (31) can be approximately maximized. The value of W can then be obtained. However, this work adopts a heuristic yet more efficient method. $P(z_i|x_i,y_i,\Theta^{(t)})$ is assumed to attain its maximum value at $z^* \in MS_i^{(t)}$. Then Equation (31) is reduced to the following equation:

$$Q_W(\Theta^{(t+1)};\Theta^{(t)}) = \sum_{i=1}^{N} lnP(y_i|x_i,z_i^*,W^{(t+1)}) + \lambda_1\Omega_{MT}(W^{(t+1)}). \quad (33)$$

The above maximization can be solved via stochastic gradient descent. The algorithmic steps are shown in Algorithm 2.

5 EXPERIMENTS

This section evaluates the proposed approach LCC-MTL on the aforementioned three learning problems: nonlinear data classification with linear SVMs, quality-aware web data classification, and listwise LTR. The two extensions of LCC-MTL on the latter two problems are called LCC-MTL$_Q$ and LCC-MTL$_R$, respectively.

Table 1. Summary of the Real Datasets in Our Experiments

Datasets	# training	# test	# features	# classes
IJCNN1	49,990	91,701	22	2
SVMGUIDE1	3,089	4000	4	2
SKIN	125,000	120,057	3	2
LETTER	3,093	1,546	16	2
Pendigits	7,494	3,498	16	2
Landsat Satellite	4,435	2000	36	2

ALGORITHM 2: LCC-MTL for Listwise Ranking (LCC-MTL$_R$)

Input: Training data $\{(x_i, y_i)|i = 1, \ldots, N\}$, the number of clusters K, λ, $t = 1$
Output: Parameter $\Theta = \{\pi_k, \mu_k, \Sigma_k, w_k|k = 1, \ldots, K\}$
Initialize Θ by K-means and ListMLE.
repeat
Sampling: Sample $MS_i^{(t)}$ for each object x_i according to Equation (32).
E step: Evaluate $\tau_{i,j,k}^{(t)}$ using Equation (24) based on MS_i.
M step: Reestimate $\pi_k^{(t+1)}$, $\mu_k^{(t+1)}$, and $\Sigma_k^{(t+1)}$ for all k using Equations (25), (26), and (27), respectively.
Reestimate $w_k^{(t+1)}$ for all k simultaneously using Equation (33).
$t = t + 1$.
until convergence

5.1 Nonlinear Data Classification with Linear SVMs

5.1.1 Datasets. We use six benchmark datasets: IJCNN1, SVMGUIDE1, SKIN segmentation, LETTER recognition, Pendigits, and Landsat Satellite. The first two are taken from the LibSVM website (Chang and Lin 2011); the others are available at the UCI machine-learning repository (Bache and Lichman 2013). All the datasets have been divided into training and testing sets except the SKIN and LETTER datasets. For the SKIN dataset, the first half of positive and negative samples are used for training. For the LETTER dataset, letters A, B, C and D, E, F are grouped into the positive and negative classes, respectively, with two-thirds of each class used for training. In order to use the datasets for binary classification, for the Pendigits dataset, digits 0 to 4 are labelled as the positive class and the remaining digits are labelled as the negative class. For the Landsat Satellite dataset, classes 1 to 3 are labelled as positive, with the remaining labelled as negative. Each sample vector in each dataset is l_2-normalized to unit length. Table 1 gives a brief summary of these datasets.

We compare LCC-MTL with 12 previously mentioned methods: Linear SVM, Kernel SVM, SVM-KNN, SpSVM, HME, K-means+SVM, MLSVM, LLSVM, CSVM, partition-wise linear models (PLMs)[4] (Oiwa and Fujimaki 2014), tensor sketching (TS)[5] (Pham and Pagh 2013), and our early method LSVM-MTL. For kernel SVM, we use the RBF kernel. For SpSVM, the number of basis functions is set to 70. For HME, the number of experts is set to 16 to construct a balanced hierarchy. The parameters of all the other methods are set as in Gu and Han (2013), with most parameters set by cross-validation. For those methods involving K-means clustering or other random factors, we calculate the average accuracy and the standard deviation on the test set over

[4]The authors did not provide the code. Therefore, we implemented it according to the paper.
[5]The codes are available at https://bitbucket.org/johanvts/fastkernel/.

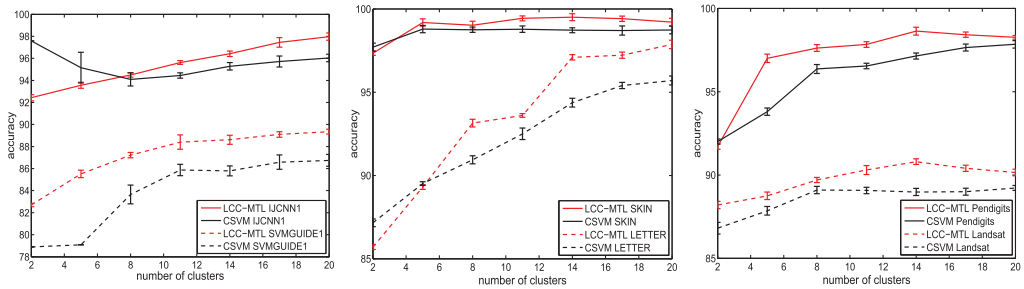
Table 2. Comparison of Different Classifiers in Terms of Classification Accuracy (%)

Datasets	IJCNN1	SVMGUIDE1	SKIN	LETTER	Pendigits	Landsat Satellite
Linear SVM	91.01	79.13	97.43	84.60	80.84	86.01
Kernel SVM	98.72	87.95	99.60	99.35	98.91	91.20
SVM-KNN	92.45	85.78	98.88	95.05	97.43	86.93
SpSVM	95.13±0.43	87.67±0.10	99.47±0.12	93.79±0.38	95.93±0.63	88.84±0.27
HME	93.92±0.27	88.43±0.32	97.05±0.18	93.63±0.21	95.25±0.14	87.32±0.19
K-means+SVM	93.87±0.53	83.25±0.72	97.82±0.28	93.66±0.35	96.89±0.19	87.55±0.23
MLSVM	93.41±0.19	83.27±0.64	98.12±0.37	93.89±0.42	97.21±0.26	87.63±0.28
LLSVM	94.07±0.45	87.64±0.30	98.36±0.21	95.68±0.17	98.11±0.38	87.42±0.11
CSVM	95.41±0.34	86.32±0.47	98.72±0.15	94.37±0.26	97.14±0.18	88.98±0.21
LSVM-MTL	96.32±0.27	87.88±0.43	98.70±0.19	96.12±0.14	98.28±0.23	89.70±0.15
PLM	95.51±0.48	87.65±0.57	99.18±0.13	97.37±0.28	98.24±0.22	89.36±0.23
TS	94.79±0.33	85.42±0.56	96.16±0.17	95.81±0.22	97.63±0.13	88.60±0.25
LCC-MTL	96.42±0.25	88.62±0.41	99.50±0.21	97.09±0.16	98.63±0.24	90.80±0.17

10 random repetitions. The results are presented in Table 2. Here, we set the number of clusters K to 14 for K-means+SVM, CSVM, and LCC-MTL. The parameter settings for PLM and TS are the same as the settings in Oiwa and Fujimaki (2014) and Pham and Pagh (2013), respectively.

Unsurprisingly, linear SVM achieves the lowest performance on all the datasets. Kernel SVM achieves the best performance on all the datasets except the SVMGUIDE1 dataset. Nevertheless, kernel SVM can be prohibitively expensive when dealing with large datasets. Our proposed LCC-MTL achieves not only comparable performance to kernel SVM, but also much higher efficiency in prediction. The reason is that the prediction complexity of LCC-MTL is linear in the number of tasks K, while the prediction complexity of kernel SVM scales with the number of support vectors. For example, with $K = 14$, the prediction time of LCC-MTL on the IJCNN1 dataset is 0.24 seconds, whereas the time of kernel SVM is 34.71 seconds, with 7,924 support vectors learned. SpSVM is a kind of kernel SVM fast evaluation by reducing the number of basis functions (support vectors). Although we set the number of basis functions to 70, five times the number of tasks, its performance is not comparable to our method. HME is a classical mixture of expert methods. Its slightly inferior performance may be due to the fact that its gating function is not flexible enough in partitioning the feature space and the adopted expert function is the generalized linear model rather than the SVM. Even though SVM-KNN and LLSVM perform well on some datasets, they are slow due to the nature of lazy learning and local coordinate coding, respectively. LLSVM is sometimes slower than kernel SVM (Gu and Han 2013). The poor performance of K-means+SVM is likely a result of its ignorance of the relatedness among the multiple tasks. MLSVM only yields slightly better results than K-means+SVM with a considerable increase in computational complexity. PLM also achieves performance close to that of LCC-MTL. The performance of TS is not stable due partially to the fact that TS assumes that the polynomial kernel is suitable for the data.

CSVM is highly similar to the proposed algorithm; hence, we compare it with LCC-MTL in more detail. Figure 8 shows the classification accuracies of CSVM and LCC-MTL with the number of clusters K ranging from 2 to 20. LCC-MTL outperforms CSVM and LSVM-MTL on all the datasets. The performance of LCC-MTL generally improves with the number of clusters. Two factors may account for this improvement. First, when the number of clusters increases, the samples in each cluster become linearly separable, and the corresponding linear SVM can classify them well. Second, with the increasing number of clusters (tasks), multitask learning is better utilized to transfer knowledge among tasks and avoid overfitting. The performance of LCC-MTL generally stabilizes as K exceeds a certain threshold. Therefore, LCC-MTL is quite robust to the choice of K. In practice, K can be set slightly larger, as efficiency is only slightly affected.



(a) IJCNN1 and SVMGUIDE1 (b) SKIN and LETTER (c) Pendigits and Landsat Satellite

Fig. 8. Comparison of classification accuracies for CSVM and LCC-MTL with respect to the number of clusters K.

We then remove three-quarters of labels of training data in IJCNN1 and Landsat Satellite. Supervised LCC-MTL using only labeled data achieves accuracies of 0.84 in IJCNN1 and 0.79 in Landsat Satellite, while semisupervised LCC-MTL achieves accuracies of 0.89 in IJCNN1 and 0.86 in Landsat Satellite, respectively. This improvement demonstrates that semisupervised LCC-MTL can exploit the abundant unlabeled data to capture the manifold structure of data.

5.2 Web Data Classification

5.2.1 Experimental Setup. Two common usedly classification algorithms, SVM and random forest (RF) (Breiman 2001), are used as the baseline competing methods. The two algorithms, LQHC and LQSC, presented in our early work (Wu et al. 2014) are also used as the competing methods. Another intuitive algorithm, which directly takes quality-aware factors as additional features, is also compared. This algorithm directly combines the conventional and quality-aware factors as a new feature vector for each sample, which is called **direct concatenation**. The radial basis kernel is chosen for (kernel) SVM. The parameters C and g are searched via five-cross validation in $\{0.1, 1, 10, 50, 100\}$ and $\{0.001, 0.01, 0.1, 1, 10\}$, respectively. For the SVM used in LQHC and LQSC, the parameters are searched with the same settings. For RF, only the number of trees in $\{10, 50, 100, 200, 300\}$ is changed, and other parameters are defaults. Specifically, the parameter γ in LQHC and LQSC is searched in $\{0.0001, 0.001, 0.01, 0.1, 1\}$. For the direct concatenation algorithm, the SVM is used. The maximum number of iterations used in LQSC is set to 20. Three measures—precision, recall, and $F1$—are used.

5.2.2 Results on Cannabis Web Page Recognition. Illicit cannabis web pages have a negative influence on users, especially teenagers (Wang et al. 2011). The dataset consisting of 4,427 normal and cannabis web pages in Wang et al. (2011) is used. Throughout the experiments, all the web pages are randomly split into two equal parts. One part is used for training and the other is used for testing. The random splitting is repeated 10 times and the average classification results are recorded. Given a web page, let I_c be its image count and W_c be its word count. They are normalized as follows: $NI_c = min(I_c/80, 1)$ and $NW_c = min(W_c/8000, 1)$.

Some pages contain more than 2,000 words, whereas some pages contain no more than 10 words. Some pages contain more than 50 images, whereas some pages contain no image. Three typical pages are also shown in Figure 9. The parameters NI_c and NW_c are taken as the quality-aware factors[6] of each page.

[6]It should be noted that some other factors, such as the number of hyperlinks and the image sizes, can also be taken as quality-aware factors. These factors will be considered in our future work.

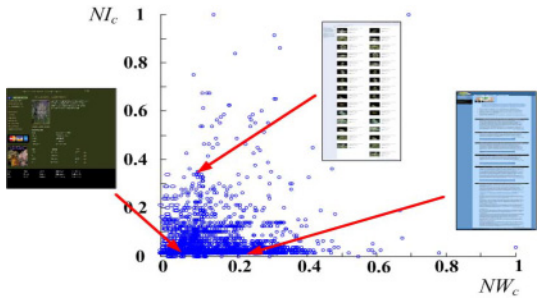


Fig. 9. The distribution of NI_C and NW_C on the cannabis web page dataset and three typical web pages.

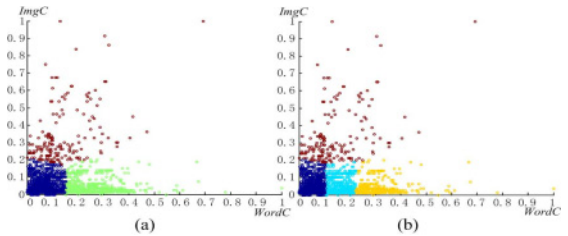


Fig. 10. The clustering of the quality-aware factors of the cannabis web page set.

The document frequency method is used for text features. A total of 100 words are used. Therefore, the text features for each page are a 100-dimensional vector. A page usually contains more than one image. The image features are extracted as follows. First, the standard scale-invariant feature transform (Lowe 2004) is used for local patch description, and the bag-of-words model (Csurka et al. 2004) is used to construct the histogram for each image. Second, all histograms are clustered into m subsets. All the images of each page are allocated into m clusters, and the normalized histogram of the numbers of images in all the m clusters is taken as the feature vector. In the experiments, m is set to 50. Therefore, the image features of each page consist of a 50-dimensional vector. The text and image features of each page are concatenated, and a 150-dimensional feature vector is obtained.

The clustering results with K-means for *ImgC* and *WordC* are shown in Figure 10. In Figure 10(a), the pages are divided into three clusters: image dominant (the top cluster), text dominant (the right cluster), and mixture of images and texts. In Figure 10(b), the pages are divided into four clusters. The cluster of mixture of images and texts and the cluster of text dominant in Figure 10(a) are further divided into three parts in Figure 10(b). The right part contains more texts than the middle part, while the middle part contains more texts than the left part. We have also observed that the clusters do not have clear margins. Therefore, using a soft clustering strategy is more reasonable than that using a hard strategy.

To explore the effects of *ImgC* and *WordC* on classification, the data are split according to the two factors. The left image of Figure 11 shows the data clustering by using *WordC*. The corresponding data subset of each quality cluster is randomly split into two equal parts. One part is used for training and the other is used for test. The random split is repeated 10 times and the average classification results are recorded. The $F1$ values on the three clusters' corresponding datasets by SVM are shown in the right image of Figure 11. The text length is not positively correlated with the classification results. Given that the number of images of the collected pages is mainly in [0, 5],

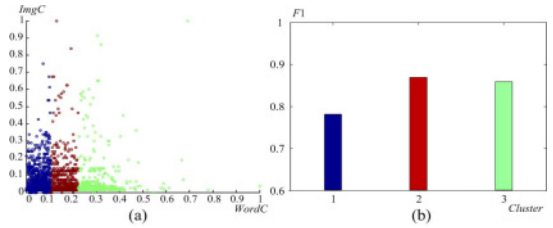


Fig. 11. The clustering of quality-based factors of the cannabis web page set according to word counts (a), and the $F1$ values of the corresponding data subsets (b).

Table 3. Results of Cannabis Web Page Recognition

	Precision	Recall	$F1$
SVM (on conventional features x_i)	0.9323	0.8563	0.8926
RF (on conventional features x_i)	0.9291	0.8580	0.8921
(Wang et al. 2011) (on conventional features x_i)	0.9211	0.8933	0.9070
Direct concatenation	0.9195	0.9001	0.9097
LQHC ($K = 3$)	0.9676	0.8887	0.9265
LQSC ($K = 3$)	0.9781	0.8983	0.9365
LCC-MTL$_Q$ ($K = 3$)	0.9753	0.9148	**0.9441**

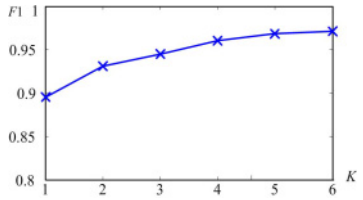


Fig. 12. The variations of the $F1$ values of LCC-MTL$_Q$ with different numbers of clusters (K) on the cannabis page set.

the pages are directly divided into three subsets if $I_c \leq 2$, or $2 < I_c \leq 5$, or $I_c > 5$. The $F1$ values of the three subsets are 0.5498, 0.6635, and 0.8333, respectively.

Table 3 shows the classification results of the seven competing algorithms. In LQHC, LQSC, and LCC-MTL$_Q$, the number of clusters (K) is set as 3. All four learning algorithms using quality-aware factors (direct concatenation, LQHC, LQSC, LCC-MTL$_Q$) achieve better results compared with the other three algorithms which are based on conventional features alone. The $F1$ value of LCC-MTL$_Q$ is about 5.15% higher than that of the SVM, which does not utilize dividing features.

To test the robustness of LCC-MTL$_Q$, we perform LCC-MTL$_Q$ under different numbers of clusters (K). Figure 12 shows the recognition results of LCC-MTL$_Q$ with the increasing of K in terms of the $F1$ values. When $K = 1$, the $F1$ value of LCC-MTL$_Q$ is equal to kernel SVM. The reason is that when $K = 1$, LCC-MTL$_Q$ is reduced to kernel SVM. When $K \geq 1$, LCC-MTL$_Q$ achieves increasing $F1$ values. When K equals 6, the $F1$ value is 0.9719. The partial reason for the performance improvement is that with the increase of K, the quality-aware factors in each training subset vary slightly and become more similar with each other. Furthermore, although the numbers of samples in each training subset become smaller, indicating that the corresponding classifiers may be insufficiently learned, the multitask learning used here alleviates this problem by transferring knowledge among training subsets.

Table 4. $F1$ Values Based on Partial Labeled Cannabis Web Pages

	1/4 labeled data	1/2 labeled data	3/4 labeled data
(Supervised) LCC-MTL$_Q$ ($K = 3$)	0.8817	0.9047	0.9304
(Supervised) LCC-MTL$_Q$ ($K = 5$)	0.8967	0.9323	0.9576
Semi-supervised LCC-MTL$_Q$ ($K = 3$)	0.9109	0.9287	0.9317
Semi-supervised LCC-MTL$_Q$ ($K = 5$)	0.9316	0.9570	0.9634

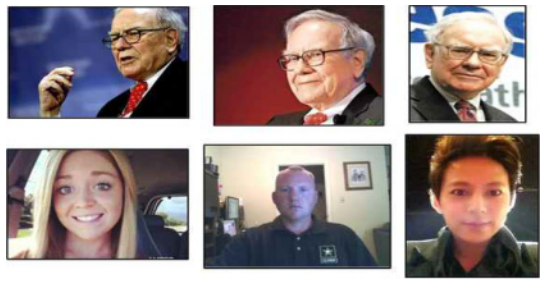


Fig. 13. Six images from the Internet.

To evaluate the performance of semisupervised LCC-MTL$_Q$, one-quarter, one-half, and three-quarters of the labels of the training data are removed to construct three partial labeled training sets, respectively. We then compare semisupervised LCC-MTL$_Q$ and (supervised) LCC-MTL$_Q$. Semisupervised LCC-MTL$_Q$ is run on the whole partial labeled training data, whereas LCC-MTL$_Q$ is *only* run on the labeled data. The results are shown in Table 4. Semisupervised LCC-MTL$_Q$ consistently outperforms (supervised) LCC-MTL$_Q$ when a number of unlabeled samples are available during the dividing step.

5.2.3 Results on Pornographic Image Recognition. Recently, pornographic image recognition has attracted much attention in both academic research and industrial application. Most existing algorithms rely on the skin features of images. Therefore, skin detection is a key step and serves as the basis in many previous algorithms. However, the illumination of web images is very complex. Figure 13 shows normal images from the Internet. The top three images feature the same person. However, the skin colors change under different illumination conditions. The bottom three images are captured by phone or PC cameras and have low-quality illumination conditions. Considering that skin detection plays a crucial role in existing studies, we evaluate the quality of detected skin pixels and then apply the quality to succeeding model training and classification.

Assessing directly the quality of extracted skin pixels for pornographic image classification is difficult. Note that the quality of extracted skin pixels is most affected by illumination (Hu et al. 2007). Therefore, we adopt an alternative strategy. First, we estimate the illumination of each image. We then cluster the illumination and sort images with similar illumination conditions into the same cluster. Consequently, the quality levels of detected skin pixels of the images in the same training subset may be similar. The algorithm proposed by Weijer et al. (2007) is applied to esti-

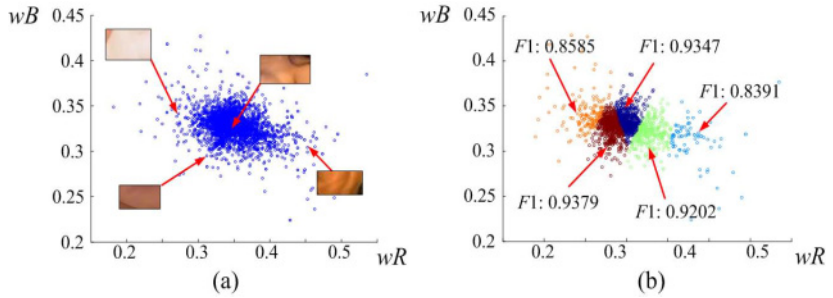


Fig. 14. (a) The distribution of the quality-aware factors of the pornographic image set and some skin patches. (b) The clusters of the quality-aware factors and the $F1$ values.

Table 5. Results of Pornographic Image Recognition

	Precision	Recall	$F1$
SVM (only on features x_i)	0.9097	0.8920	0.9008
RF (only on features x_i)	0.9196	0.9018	0.9106
LQHC ($K = 3$)	0.9325	0.9144	0.9234
LQSC ($K = 3$)	0.9524	0.9339	0.9430
LCC-MTL$_Q$ ($K = 3$)	0.9672	0.9408	**0.9538**

mate the illumination of an input image. The algorithm outputs the illumination color with two quantities (wR, wB) that are taken as the quality-aware factors for an image.

The image data introduced in Zuo et al. (2010) is applied. The distribution of the estimated illumination is shown in Figure 14. The images in some areas have bad illumination conditions. Figure 14(a) also shows the skin patches of some sample images. The colors of skin with different illumination conditions vary significantly.

To explore the relationship between the classification performance and illumination, we divide the dataset according to the estimated illumination. The corresponding data subset for each cluster is randomly split into two equal parts. One part is used for training and the other is used for testing. The random split is repeated 10 times. A RBF kernel SVM classifier is used and the average classification results are recorded. Finally, the $F1$ values of the different clusters' corresponding data subsets are obtained. Figure 14(b) shows the clustering of quality-aware factors and the $F1$ results. The clusters with worse illumination have lower $F1$ values.

Skin detection and feature extraction adapt the methods used by Zuo et al. (2010). Table 5 shows the classification results of the five competing methods. For LQHC, LQSC, and LCC-MTL$_Q$, the number of clusters is set to 3. All learning algorithms using quality-aware factors—direct concatenation, LQHC, LQSC, and LCC-MTL$_Q$—still achieve better results than the others do. The $F1$ value of the LCC-MTL method is about 5.30% higher than that of the SVM without considering information quality.

We then perform LCC-MTL$_Q$ under different numbers of clusters. Figure 15 shows the $F1$ values with the increasing of K. Similar observations to those from Figure 12 are obtained.

To evaluate the performance of semisupervised LCC-MTL$_Q$ on this classification task, one-quarter, one-half, and three-quarters of labels of the training data are also removed to construct three partial labeled training sets. Semisupervised LCC-MTL$_Q$ is run on total partial labeled training data, whereas (supervised) LCC-MTL$_Q$ is *only* run on the labeled data. The results are shown in

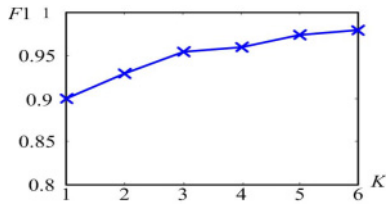


Fig. 15. The variations of the $F1$ values of LCC-MTL$_Q$ with different numbers of clusters (K) on the porno image set.

Table 6. $F1$ Values Based on Partial Labeled Porno Images

	1/4 labeled data	1/2 labeled data	3/4 labeled data
(Supervised) LCC-MTL$_Q$ ($K = 3$)	0.9038	0.9233	0.9307
(Supervised) LCC-MTL$_Q$ ($K = 5$)	0.9082	0.9361	0.9556
Semisupervised LCC-MTL$_Q$ ($K = 3$)	0.9285	0.9406	0.9499
Semisupervised LCC-MTL$_Q$ ($K = 5$)	0.9377	0.9621	0.9692

Table 6. Semisupervised LCC-MTL$_Q$ outperforms (supervised) LCC-MTL$_Q$ especially when only 1/4 training data are labeled.

5.3 Results on Listwise LTR

This section compares the proposed algorithm LCC-MTL$_R$ against two classical linear LTR algorithms (ListMLE and ListNet) and our early proposed algorithm RPC-MTL (Wu et al. 2016). Two benchmark LTR datasets, MQ2007-list and MQ2008-list, are used. These two datasets are compiled by the LETOR package (Liu 2009) and used in various listwise LTR studies. They are constructed based on Gov2 web page collection and two query sets from the Million Query track of TREC 2007 and TREC 2008. MQ2007-list contains about 1700 queries with ranked documents and MQ2008-list contains about 800 queries with ranked documents. Each query–document pair features a 46-dimensional vector. LETOR provides a 5-fold partition for these two datasets to facilitate a 5-fold cross-validation strategy. In each fold, there are three subsets for learning: training, validation, and testing. Both datasets do not provide relevance scores. We follow the score setting in Wu et al. (2016): The score of the top-1 document is defined as 1 and the score of the document in the end of the ranking list is defined as 0. The scores of the middle documents are linearly calculated based on their ranking positions. Both numbers of clusters in RPC-MTL and LCC-MTL$_R$ are set as five on the two datasets. In LCC-MTL$_R$, the sampling size is set as 1,000 for each object.

The results of NDCG@n ($n = 1, 2, \ldots, 10$) on the two datasets are displayed in Figures 16 and 17, respectively. By conducting the t-test, LCC-MTL$_R$ outperforms the other competing methods when $n < 5$. When $n \geq 5$, LCC-MTL is comparable to RPC-MTL. In addition, the results of LCC-MTL$_R$ are still better than those of ListMLE and ListNet when $n \geq 5$. To evaluate the robustness of LCC-MTL$_R$ in terms of the number of clusters (i.e., K), we compare the values of NDCG@1 and 5 under different K values. The results are shown in Figure 18. When K increases, the performance of LCC-MTL$_R$ increases and becomes stable when $K \geq 6$.

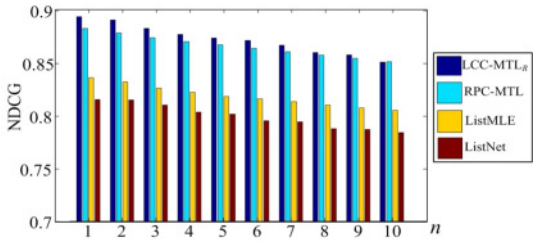


Fig. 16. The NDCG results on MQ2007-list.

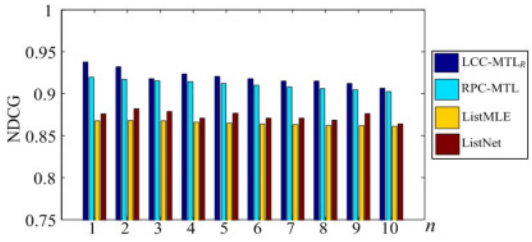


Fig. 17. The NDCG results on MQ2008-list.

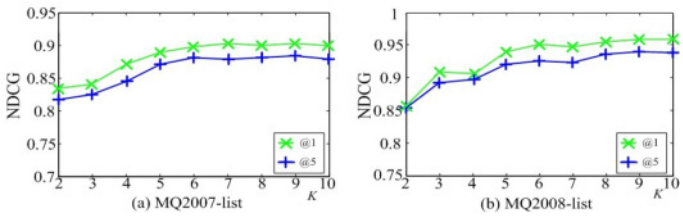


Fig. 18. The variations of the *NDCG* values of LCC-MTL$_R$ with different numbers (K) of clusters.

6 CONCLUSION

In this article, we have proposed a new divide-and-conquer approach, called LCC-MTL, to deal with learning for nonlinear data classification with linear SVMs. LCC-MTL consists of two iterative steps: clustering the data using a GMM with local consistency and training a classifier for each cluster. These two steps are combined into a generative model and implemented with an EM algorithm. Furthermore, we have considered the training of each classifier as a single task and used clustered multitask learning to capture the relatedness among tasks. The proposed approach has also been extended to two distinct learning problems: quality-aware web data classification and listwise learning to rank. Two new algorithms are obtained for these two problems. Experimental results on benchmark datasets demonstrate that the LCC-MTL and the two extensions (i.e., LCC-MTL$_Q$ and LCC-MTL$_R$) outperform state-of-the-art methods in nonlinear classification, quality-aware web data classification, and listwise LTR, respectively. In the prediction phase, it also achieves much higher efficiency than kernel SVMs with comparable classification performance in nonlinear data classification.

REFERENCES

K. Bache and M. Lichman. 2013. UCI Machine Learning Repository. Retrieved September 8, 2017 from http://archive.ics.uci.edu/ml.

Enrico Blanzieri and Farid Melgani. 2006. An adaptive SVM nearest neighbor classifier for remotely sensed imagery. In *IEEE International Conference on Geoscience and Remote Sensing Symposium*. 3931–3934.

Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1, 5–32.

F. Cai and V. Cherkassky. 2012. Generalized SMO algorithm for SVM-based multi-task learning. *IEEE Transactions on Neural Networks and Learning Systems* 23, 6, 997–1003.

Zhe Cao, Tao Qin, Tie Yan Liu, Ming Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. In *International Conference on Machine Learning*. 129–136.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2, 3, 27:1–27:27.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning* 20, 3, 273–297.

Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cdric Bray. 2004. Visual categorization with bags of keypoints. *ECCV Workshops on Statistical Learning in Computer Vision* 1–22.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B (Methodological)* 1–38.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi-task learning. In *Proceedings of ACM KDD*. 109–117.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research* 9, 1871–1874.

Zhouyu Fu, Antonio Robles-Kelly, and Jun Zhou. 2010. Mixing linear SVMs for nonlinear classification. *IEEE Transactions on Neural Networks* 21, 12, 1963–1975.

Quanquan Gu and Jiawei Han. 2013. Clustered support vector machines. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS'13)*. 307–315.

T. Hastie, R. Tibshirani, and J. Friedman. 2001. *The Elements of Statistical Learning*. Springer, New York.

Weiming Hu, Ou Wu, Zhouyao Chen, Zhouyu Fu, and Steve Maybank. 2007. Recognition of pornographic web pages by classifying texts and images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 6, 1019–34.

Laurent Jacob, Francis Bach, and Jean-Philippe Vert. 2008. Clustered multi-task learning: A convex formulation. In *Advances in Neural Information Processing Systems (NIPS)*. 745–752.

S. Sathiya Keerthi, Olivier Chapelle, and Dennis DeCoste. 2006. Building support vector machines with reduced classifier complexity. *The Journal of Machine Learning Research* 7, 1493–1515.

J. Kittler, N. Poh, and K. Kryszczuk. 2007. Quality dependent fusion of intramodal and multimodal biometric experts. *Proceedings of SPIE - The International Society for Optical Engineering* 6539, 653903–653903–14.

Lubor Ladický and Philip Torr. 2011. Locally linear support vector machines. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*. 985–992.

Jialu Liu, Deng Cai, and Xiaofei He. 2010. Gaussian mixture model with local consistency. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI'10)*. 512–517.

Tie-Yan Liu. 2009. *Learning to Rank for Information Retrieval*. Now Publishers. 423–434 pages.

David G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60, 60, 91–110.

Yong Luo, Dacheng Tao, Bo Geng, and Chao Xu. 2013. Manifold regularized multitask learning for semi-supervised multi-label image classification. *IEEE Transactions on Image Processing* 22, 2, 523–536.

Yong Luo, Yonggang Wen, Dacheng Tao, and Jie Gui. 2015. Large margin multi-modal multi-task feature extraction for image classification. *IEEE Transactions on Image Processing* 25, 1, 414–427.

Xue Mao, Ou Wu, Weiming Hu, and Peter O'Donovan. 2014. Nonlinear classification via linear SVMs and multi-task learning. In *Proceedings of ACM CIKM*. 1955–1958.

Taesup Moon, Alex Smola, Yi Chang, and Zhaohui Zheng. 2010. IntervalRank: Isotonic regression with listwise and pairwise constraints. In *ACM International Conference on Web Search and Data Mining*. 151–160.

Karthik Nandakumar, Yi Chen, Sarat C. Dass, and Anil Jain. 2007. Likelihood ratio-based biometric score fusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 2, 342–347.

Hidekazu Oiwa and Ryohei Fujimaki. 2014. Partition-wise linear models. In *Advances in Neural Information Processing Systems*. 3527–3535.

Dmitry Yurievich Pavlov, Alexey Gorodilov, and Cliff A. Brunk. 2010. BagBoo: A scalable hybrid bagging-the-boosting model. In *ACM International Conference on Information and Knowledge Management*. 1897–1900.

O. Pele, B. Taskar, A. Globerson, and M. Werman. 2013. The pairwise piecewise-linear embedding for efficient non-linear classification. In *International Conference on Machine Learning*. 205–213.

Ninh Pham and Rasmus Pagh. 2013. Fast and scalable polynomial kernels via explicit feature maps. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 239–247.

N. Poh and J. Kittler. 2012. A unified framework for biometric expert fusion incorporating quality measures. *IEEE Transactions on Software Engineering* 34, 1, 3–18.

Tao Qin, Tie Yan Liu, Xu Dong Zhang, De Sheng Wang, and Hang Li. 2008. Global ranking using continuous conditional random fields. In *Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December*. 1281–1288.

Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems* 20, 1177–1184.

Peter Sollich. 2000. Probabilistic methods for support vector machines. In *Advances in Neural Information Processing Systems (NIPS)*. 349–355.

S. Vempati, A. Vedaldi, A. Zisserman, and C. V. Jawahar. 2010. Generalized RBF feature maps for efficient detection. In *Proceedings of British Machine Vision Conference*. 1–11.

Yinjuan Wang, Nianhua Xie, Weiming Hu, and Jinfeng Yang. 2011. Multi-modal multiple-instance learning with the application to the cannabis webpage recognition. In *Pattern Recognition*. 105–109.

Joost van de Weijer, Theo Gevers, and Arjan Gijsenij. 2007. Edge-based color constancy. *IEEE Transactions on Image Processing* 16, 9, 2207–2214.

Ou Wu, Ruiguang Hu, Xue Mao, and Weiming Hu. 2014. Quality-based learning for web data classification. In *AAAI Conference on Artificial Intelligence*. 194–200.

Ou Wu, Qiang You, Xue Mao, Fen Xia, Fei Yuan, and Weiming Hu. 2016. Listwise learning to rank by exploring structure of objects. *IEEE Transactions on Knowledge and Data Engineering* 28, 7, 1934–1939.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank:theory and algorithm. In *International Conference on Machine Learning*. 1192–1199.

Kai Yu, Volker Tresp, and Anton Schwaighofer. 2005. Learning gaussian processes from multiple tasks. In *Proceedings of ICML*. 1012–1019.

Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. 2006. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2126–2136.

Yu Zhang and Dit-Yan Yeung. 2010. Multi-task learning using generalized *t* process. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS'10)*. 964–971.

Jiayu Zhou, Jianhui Chen, and Jieping Ye. 2011a. Clustered multi-task learning via alternating structure optimization. In *Advances in Neural Information Processing Systems (NIPS)*. 702–710.

J. Zhou, J. Chen, and J. Ye. 2011b. *MALSAR: Multi-tAsk Learning via StructurAl Regularization*. Arizona State University. Retrieved September 8, 2017 from http://www.public.asu.edu/ jye02/Software/MALSAR.

Jun Zhu, Ning Chen, and Eric P. Xing. 2011. Infinite SVM: A Dirichlet process mixture of large-margin kernel machines. In *ICML*. 617–624.

Haiqiangn Zuo, Weiming Hu, and Ou Wu. 2010. Patch-based skin color detection and its application to pornography image filtering. In *International Conference on World Wide Web (WWW)*. 1227–1228.

Received April 2017; revised June 2017; accepted July 2017