

# Nonlinear Classification via Linear SVMs and Multi-Task Learning

Xue Mao, Ou Wu, Weiming Hu  
NLPR, Institute of Automation  
Chinese Academy of Sciences, China  
{xue.mao, wuou, wmu}@nlpr.ia.ac.cn

Peter O'Donovan  
Department of Computer Science  
University of Toronto, Canada  
odonovan@dgp.toronto.edu

## ABSTRACT

Kernel SVM is prohibitively expensive when dealing with large nonlinear data. While ensembles of linear classifiers have been proposed to address this inefficiency, these methods are time-consuming or lack robustness. We propose an efficient classifier for nonlinear data using a new iterative learning algorithm, which partitions the data into clusters, and then trains a linear SVM for each cluster. These two steps are combined into a graphical model, with the parameters estimated efficiently using the EM algorithm. During training, clustered multi-task learning is used to capture the relatedness among the multiple linear SVMs and avoid overfitting. Experimental results on benchmark datasets show that our method outperforms state-of-the-art methods. During prediction, it also obtains comparable classification performance to kernel SVM, with much higher efficiency.

## Categories and Subject Descriptors

G.3 [Probability and Statistics]: Statistical computing;  
I.2.6 [Artificial Intelligence]: Learning

## General Terms

Algorithms, Performance, Experimentation

## Keywords

Nonlinear Classification; Linear SVMs; Multi-Task Learning

## 1. INTRODUCTION

Kernel SVM often produces satisfactory classification results on nonlinear data. Unfortunately, the complexity of kernel SVM relies on the number of support vectors. Alternatively, while linear SVM is extremely efficient, it performs poorly on nonlinear data. Ensembles of linear SVMs can improve performance, though these methods either lack robustness, such as CSVM [5] which aligns the linear SVM weight vectors with a global weight vector, or are time-consuming, such as SVM-KNN [8] which uses a lazy learning strategy.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.  
CIKM'14, November 3–7, 2014, Shanghai, China.  
Copyright 2014 ACM 978-1-4503-2598-1/14/11 ...\$15.00.  
<http://dx.doi.org/10.1145/2661829.2662068>.

In this paper, an efficient classifier for nonlinear data is constructed by using Linear SVMs and Multi-Task Learning (LSVM-MTL). The method uses a divide-and-conquer strategy which partitions the data into clusters using a Gaussian mixture model (GMM), and then trains a linear SVM for each cluster. Instead of being treated independently, the two steps are combined into a generative model and alternatively performed in each iteration. To ensure the data points in each cluster are linearly separable, some clusters may have relatively few points, and can overfit. In this work, we consider training a linear SVM for a cluster as a single task, and the training of the classifier ensemble as a multi-task learning problem. Clustered multi-task learning is used to exploit the relatedness between tasks, and avoid overfitting. To our knowledge, multi-task learning has not previously been used to train multiple linear SVMs on nonlinear datasets. Experimental results on benchmark datasets demonstrate that the model outperforms state-of-the-art methods. For prediction, LSVM-MTL achieves much higher efficiency than kernel SVM, with comparable classification performance.

## 2. RELATED WORK

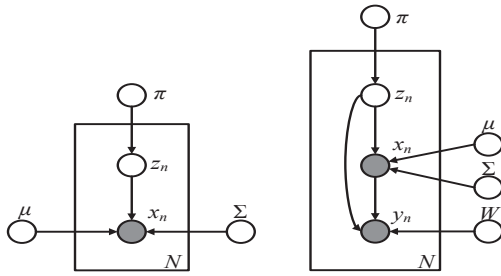
Methods for learning multiple linear SVMs for nonlinear data can be roughly divided into two categories. In lazy learning methods, such as SVM-KNN [8], the learning process is postponed until the testing phase, which is therefore expensive. By contrast, eager learning methods construct local classifiers during the training phase, usually employing a divide-and-conquer strategy. MLSVM [4] and CSVM [5] fall into this category. Other eager learning methods, such as LLSVM [6], use local coordinate coding. These methods either lack robustness, such as CSVM, or are time-consuming, such as SVM-KNN, MLSVM and LLSVM.

Multi-task learning (MTL) is a method where multiple related tasks are learned simultaneously to improve generalization [9]. It has been used in various areas, such as web mining [7].

## 3. LSVM-MTL MODEL

### 3.1 Model Formulation

At a high level, LSVM-MTL is an iterative divide-and-conquer approach which alternates between two steps: partitioning the data into clusters with a GMM and training a linear SVM in each cluster. Instead of being independent, the two steps promote each other: GMM clustering improves the SVM classification performance for each cluster, and vice versa. This idea is integrated into our LSVM-MTL model



**Figure 1: (a) GMM (b) LSVM-MTL**

as shown in Figure 1(b). The upper part of the model corresponds to the GMM in Figure 1(a), which is responsible for partitioning the data into clusters. The lower part of the model trains linear SVMs for each cluster.

We introduce the following notation: in Figure 1(b), there are  $N$  i.i.d. samples  $X = \{x_n\}_{n=1, \dots, N}$  and their corresponding labels  $Y = \{y_n\}_{n=1, \dots, N}$ . The latent variables  $Z = \{z_n\}_{n=1, \dots, N}$  denote the assignments of samples to the  $K$  mixtures. The parameters  $\mu = \{\mu_j\}_{j=1, \dots, K}$  and  $\Sigma = \{\Sigma_j\}_{j=1, \dots, K}$  denote the centroids and covariance matrices of Gaussian components respectively.  $W = \{w_j\}_{j=1, \dots, K}$  represents the weight vectors of the linear SVMs for the  $K$  clusters.  $\pi = \{\pi_j\}_{j=1, \dots, K}$  are the mixing coefficients of the GMM. Let  $\Theta = \{\pi, \mu, \Sigma, W\}$  be the total set of model parameters. The joint distribution over  $X$  and  $Y$  is:

$$\begin{aligned} P(X, Y | \Theta) &= \prod_{n=1}^N P(x_n, y_n | \Theta) \\ &= \prod_{n=1}^N \sum_{z_n=1}^K \pi_{z_n} P(x_n | z_n, \mu, \Sigma) P(y_n | x_n, z_n, W) \\ &= \prod_{n=1}^N \sum_{j=1}^K \pi_j \mathcal{N}(x_n | z_n = j, \mu_j, \Sigma_j) P(y_n | x_n, w_j) \quad (1) \end{aligned}$$

which is obtained by summing the joint distribution of observed variables  $X, Y$  and latent variables  $Z$  over all possible states of  $Z$ , with  $z_n$  taking values in  $\{1, \dots, K\}$ . The mixing coefficient  $\pi_j$  is the prior probability of picking the  $j$ th Gaussian component.  $P(x_n | z_n = j, \mu, \Sigma) = \mathcal{N}(x_n | z_n = j, \mu_j, \Sigma_j)$  is a Gaussian component of the mixture, specifying the probability of  $x_n$  conditioned on the  $j$ th component.  $P(y_n | x_n, w_j)$  is the posterior probability of the  $n$ th sample output by the  $j$ th linear SVM. We estimate the parameters by maximum likelihood. The regularized log-likelihood is:

$$\begin{aligned} \mathcal{L}(\Theta) &= \sum_{n=1}^N \log P(x_n, y_n | \Theta) + \Omega(W) \\ &= \sum_{n=1}^N \log \sum_{j=1}^K \pi_j \mathcal{N}(x_n | z_n = j, \mu_j, \Sigma_j) P(y_n | x_n, w_j) + \Omega(W) \quad (2) \end{aligned}$$

where  $\Omega(W)$  is a regularization term on the linear SVM weight vectors. This term encodes prior knowledge about the  $K$  classifiers, and is defined shortly.

Unlike previous work, our model learns the multiple linear classifiers simultaneously rather than independently. More specifically, if training a linear SVM in a cluster is regarded as a task, training linear SVMs for all clusters corresponds to a multi-task learning problem. To ensure that the samples in each cluster are linearly separable, the samples available for each cluster may be limited, possibly leading to over-fitting. More importantly, since all the clusters are partitioned from the same dataset, they should be latently related. Multi-task learning can be employed to capture the intrinsic relatedness between tasks and avoid over-fitting in each task.

We use clustered multi-task learning [9], which clusters tasks into groups, with tasks in each group having similar weight vectors. This approach is motivated by the desire for the decision boundary to be smooth and have constrained curvature, since a decision boundary with arbitrary curvature will likely overfit the data [6]. Hence, tasks in adjacent regions on the decision boundary should have similar weight vectors and should be clustered into one group. We illustrate this issue with a synthetic dataset in the experimental section. In practice, we find this clustering is beneficial for using multiple linear SVMs to nonlinear datasets. Note that LSVM-MTL clusters based on task similarities and the cluster structure is unknown beforehand. Furthermore, note that this clustering of SVM weight vectors is distinct from the GMM data clustering described previously.

In order to incorporate multi-task learning and linear SVMs into the above maximum likelihood estimation framework (2), we recall that training a linear SVM usually leads to the following quadratic optimization problem:

$$\min_w \frac{\|w\|^2}{2} + C \sum_n \ell(w; x_n, y_n) \quad (3)$$

where the first term regularizes the SVM weight vector, and the second term measures the total loss. We next describe correspondence between minimization problem in (3) and maximization problem of the log-likelihood function in (2). The first term in (3) corresponds to the second term in (2), since they both regularize the weight vector. Finally, to incorporate multi-task learning,  $\Omega(W)$  is formulated as:

$$\Omega(W) = -\alpha \sum_{c=1}^r \sum_{j \in \mathcal{I}_c} \|w_j - \bar{w}_c\|_2^2 - \beta \sum_{j=1}^K \|w_j\|_2^2 \quad (4)$$

where the first term on the right-hand side assumes that the total  $K$  tasks are clustered into  $r$  clusters, with the index set of the  $c$ th cluster defined as  $\mathcal{I}_c = \{j | j \in \text{cluster } c\}$ . The average weight vector of the  $c$ th cluster is  $\bar{w}_c = \frac{1}{m_c} \sum_{j \in \mathcal{I}_c} w_j$ , where there are  $m_c$  tasks in the  $c$ th cluster. The first term measures the within-cluster variance, which requires tasks from the same cluster to have similar weight vectors. The second term improves the generalization performance.

We next establish the relationship between the first term in (2) and the second term in (3) by defining:

$$P(y_n | x_n, w_j) = \exp(-\ell(w_j; x_n, y_n)) \quad (5)$$

The posterior probability  $P(y_n | x_n, w_j)$  will be equal to 1 if the loss  $\ell(w_j; x_n, y_n)$  is zero, otherwise  $P(y_n | x_n, w_j)$  will be less than 1. To facilitate computation,  $P(y_n | x_n, w_j)$  is not normalized. When we employ the EM algorithm to maximize (2) in the following section, the value of the log-likelihood function will increase in each iteration, regardless of whether  $P(y_n | x_n, w_j)$  is a proper probability measure.

### 3.2 The EM Algorithm and Implementation

We now apply the EM algorithm to the above LSVM-MTL model. Let  $\Theta^{(t)} = \{\pi_j^{(t)}, \mu_j^{(t)}, \Sigma_j^{(t)}, w_j^{(t)} | j = 1, \dots, K\}$  denote the collection of parameters at the  $t$ th iteration.

In each E step, the posterior probability of assigning the  $n$ th sample to the  $j$ th linear SVM is evaluated as:

$$q_{n,j}^{(t)} = \frac{\pi_j^{(t)} \mathcal{N}(x_n | z_n = j, \mu_j^{(t)}, \Sigma_j^{(t)}) P(y_n | x_n, w_j^{(t)})}{\sum_{j=1}^K \pi_j^{(t)} \mathcal{N}(x_n | z_n = j, \mu_j^{(t)}, \Sigma_j^{(t)}) P(y_n | x_n, w_j^{(t)})} \quad (6)$$

---

**Algorithm 1** LSVM-MTL

---

**Input:** Training data  $\{(x_n, y_n) | n = 1, \dots, N\} \subset \mathbb{R}^d \times \{-1, 1\}$  and the number of clusters  $K$

**Output:** Parameter  $\Theta = \{\pi_j, \mu_j, \Sigma_j, w_j | j = 1, \dots, K\}$   
Initialise  $\Theta$  by K-means.

**repeat**

E step: Evaluate  $q_{n,j}$  using Equation (6).

M step: Re-estimate the GMM-related parameters  $\pi_j$ ,  $\mu_j$  and  $\Sigma_j$  by following the steps in [2].

Re-estimate  $w_j$  for all  $j$  simultaneously as a multi-task learning problem using Equation (8).

**until** convergence

---

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

$$Q(\Theta^{(t+1)}; \Theta^{(t)}) = \sum_{n=1}^N \sum_{j=1}^K q_{n,j}^{(t)} \log [\pi_j^{(t+1)}] \quad (7)$$

$$\mathcal{N}(x_n | z_n = j, \mu_j^{(t+1)}, \Sigma_j^{(t+1)}) P(y_n | x_n, w_j^{(t+1)}) + \Omega(W^{(t+1)})$$

where  $\Omega(W^{(t+1)})$  is the regularization term given by (4).  $\Theta^{(t)}$  is the parameter for the current iteration, which directly determines the posterior probability  $q_{n,j}^{(t)}$  on the right-hand side through Equation (6).

In the M step, the parameter is updated to  $\Theta^{(t+1)}$  by maximizing (7). Fortunately, updating the parameters of the GMM and the linear SVMs is decoupled in (7). For updating the GMM-related parameters  $\{\pi, \mu, \Sigma\}$ , we follow the steps in [2]. To update the weight vectors  $W$  of the linear SVMs, we solve the following optimization problem:

$$\max_W \sum_{n=1}^N \sum_{j=1}^K q_{n,j}^{(t)} \log P(y_n | x_n, w_j^{(t+1)})$$

$$- \alpha \sum_{c=1}^r \sum_{j \in \mathcal{I}_c} \|w_j^{(t+1)} - \bar{w}_c^{(t+1)}\|_2^2 - \beta \sum_{j=1}^K \|w_j^{(t+1)}\|_2^2 \quad (8)$$

where  $\log P(y_n | x_n, w_j^{(t+1)}) = -\ell(w_j^{(t+1)}; x_n, y_n)$  using Equation (5). With the first term regarded as a weighted loss function, Equation (8) is equivalent to the clustered multi-task learning and is solved by the method of [9].

A sketch of our algorithm is presented in Algorithm 1. K-means is utilized to initialize the mixing coefficients  $\pi$ , centroids  $\mu$  and covariance matrixes  $\Sigma$ . A linear SVM is then trained for each cluster, resulting in the initial weight vectors  $W$ . With the log-likelihood function in Equation (2) being increased in each iteration of EM, our algorithm is guaranteed to converge.

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

$$\sum_{j=1}^K \pi_j \mathcal{N}(x | z = j, \mu_j, \Sigma_j) (P(1 | x, w_j) - P(-1 | x, w_j)) \quad (9)$$

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.

## 4. EXPERIMENTS

### 4.1 Synthetic Dataset

The synthetic dataset in Figure 2 consists of two shifted sine signals, with 1000 points each, and each signal considered a separate class. K-means is used to partition the data into five clusters, denoted with different colors. Linear SVM is then applied to each cluster. This simple baseline is denoted as K-means+SVM, and is also the initial stage of LSVM-MTL. Figure 2(a) shows that the linear SVM does not accurately classify the data points in each cluster. Applying CSVM [5] gives similar results. The lack of improvement for CSVM here is because the method is not iterative, so its performance is directly determined by the initial K-means clustering. Additionally, CSVM aligns each SVM weight vector with a global weight vector, which is inappropriate here. In contrast to CSVM, LSVM-MTL is iterative; Figure 2(b) shows the result at the fifth iteration. The linear SVMs correctly classify the data in each cluster, which empirically shows that the two steps in LSVM-MTL mutually reinforce each other. When the number of clusters (tasks) is increased to ten, Figure 2(c) illustrates that the ten tasks are clustered into five groups, with each group containing two tasks in adjacent regions. This result occurs because the decision boundary is smooth and tasks in adjacent regions on the decision boundary are similar. Note that the decision boundaries in Figure 2(c) correctly classify all the data points, since each linear SVM is only applied to its corresponding cluster.

### 4.2 Real Datasets

We use six benchmark datasets: IJCNN1, SVMGUIDE1, SKIN segmentation, LETTER recognition, Pendigits and Landsat Satellite. The first two are available at the LibSVM website [3], and the others are taken from the UCI machine learning repository [1].

We compare LSVM-MTL with seven previously-mentioned methods: Linear SVM, Kernel SVM, SVM-KNN, K-means+SVM, MLSVM, LLSVM, and CSVM. The parameters of all the methods are set as in [5], with most parameters set by cross validation. For methods using K-means clustering, we calculate the average accuracy and the standard deviation on the test set over 10 random repetitions. The results are presented in Table 1. Here, we set the number of clusters  $K$  to 14 for K-means+SVM, CSVM and LSVM-MTL. As expected, linear and kernel SVM achieve the worst and best performance, respectively, over all datasets. Nevertheless, kernel SVM can be prohibitively expensive for large-scale datasets. Our proposed LSVM-MTL achieves not only comparable performance to kernel SVM, but also much higher efficiency for prediction. The reason is that the prediction complexity of LSVM-MTL is linear in the number of tasks  $K$ , while the complexity of kernel SVM scales with the number of support vectors. For example, with  $K = 14$ , the prediction time of LSVM-MTL on the IJCNN1 dataset is 0.62 seconds, whereas the time of kernel SVM is 34.71 seconds, with 7924 support vectors learned. Even though SVM-KNN and LLSVM also perform well in most cases, they are slow due to the nature of lazy learning and local coordinate coding respectively. LLSVM is sometimes slower than kernel SVM [5]. The relatively poor performance of K-means+SVM is likely due to its ignorance of the relatedness

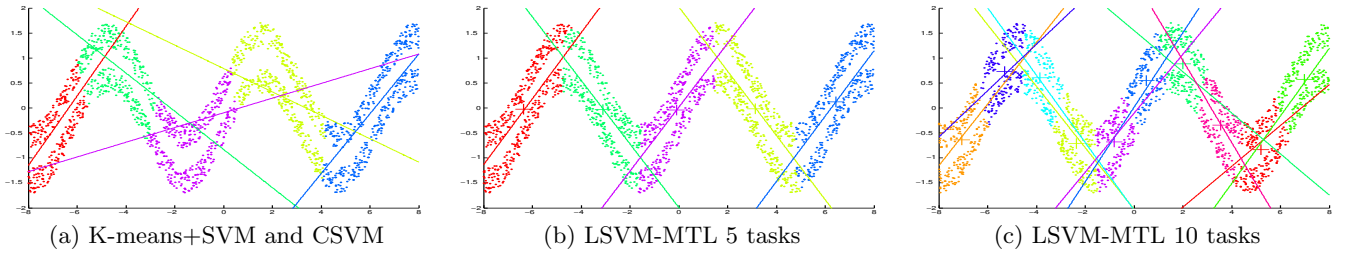


Figure 2: Learned classifiers on the synthetic sine dataset

Table 1: 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
K-means+SVM	93.87 $\pm$ 0.53	83.25 $\pm$ 0.72	97.82 $\pm$ 0.28	93.66 $\pm$ 0.35	96.89 $\pm$ 0.19	87.55 $\pm$ 0.23
MLSVM	93.41 $\pm$ 0.19	83.27 $\pm$ 0.64	98.12 $\pm$ 0.37	93.89 $\pm$ 0.42	97.21 $\pm$ 0.26	87.63 $\pm$ 0.28
LLSVM	94.07 $\pm$ 0.45	87.64 $\pm$ 0.30	98.36 $\pm$ 0.21	95.68 $\pm$ 0.17	98.11 $\pm$ 0.38	87.42 $\pm$ 0.11
CSVM	95.41 $\pm$ 0.34	86.32 $\pm$ 0.47	98.72 $\pm$ 0.15	94.37 $\pm$ 0.26	97.14 $\pm$ 0.18	88.98 $\pm$ 0.21
LSVM-MTL	96.32 $\pm$ 0.27	87.88 $\pm$ 0.43	98.70 $\pm$ 0.19	96.12 $\pm$ 0.14	98.28 $\pm$ 0.23	89.70 $\pm$ 0.15

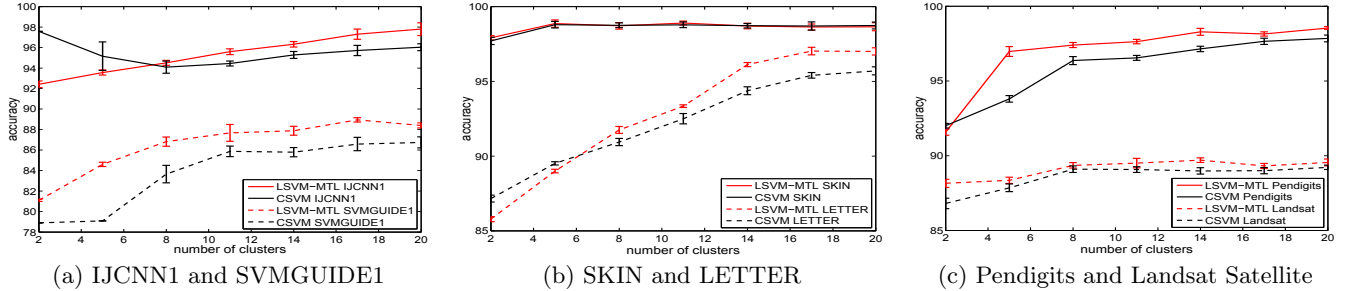


Figure 3: Classification accuracy of CSVM and LSVM-MTL with respect to the number of clusters  $K$

among the multiple tasks. MLSVM only yields slightly better results than K-means+SVM, with increased complexity.

CSVM is state-of-the-art for this field, so we compare it with LSVM-MTL in detail. Figure 3 shows the classification accuracy of CSVM and LSVM-MTL with the number of clusters ranging from 2 to 20. LSVM-MTL outperforms CSVM on all the datasets except the SKIN dataset. However, the SKIN dataset is simple, so even linear SVM produces satisfactory results. The performance of LSVM-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 SVM can classify them well. Second, with more clusters (tasks), multi-task learning is better utilized to transfer knowledge between tasks and avoid over-fitting. The performance of LSVM-MTL generally stabilizes as the number of clusters exceeds a certain threshold.

## 5. CONCLUSIONS

In this paper, we have proposed the LSVM-MTL model, which clusters the data with a GMM and trains a linear SVM for each cluster. These two steps are combined into a generative model and implemented with an EM algorithm. Furthermore, we consider the training of each linear SVM as a single task and use clustered multi-task learning to capture the relatedness between tasks. Experimental results on benchmark datasets demonstrate that LSVM-MTL outperforms state-of-the-art methods. In the prediction phase, it also achieves much higher efficiency than kernel SVM with comparable classification performance.

## 6. ACKNOWLEDGMENTS

This work is partly supported by NSFC (Grant No. 61379098, 61003115, 61103056) and Baidu research fund.

## 7. REFERENCES

- [1] K. Bache and M. Lichman. UCI machine learning repository, 2013.
- [2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [3] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. *ACM TIST*, 2(3):27, 2011.
- [4] Z. Fu, A. Robles-Kelly, and J. Zhou. Mixing linear SVMs for nonlinear classification. *IEEE TNN*, 21(12):1963–1975, 2010.
- [5] Q. Gu and J. Han. Clustered support vector machines. In *Proc. of AISTATS*, pages 307–315, 2013.
- [6] L. Ladicky and P. Torr. Locally linear support vector machines. In *Proc. of ICML*, pages 985–992, 2011.
- [7] O. Wu, R. Hu, X. Mao, and W. Hu. Quality-based learning for web data classification. In *Proc. of AAAI*, pages 194–200, 2014.
- [8] H. Zhang, A. C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Proc. of CVPR*, pages 2126–2136, 2006.
- [9] J. Zhou, J. Chen, and J. Ye. Clustered multi-task learning via alternating structure optimization. In *Proc. of NIPS*, pages 702–710, 2011.