Online Methods for Multi-Domain Learning and Adaptation

Mark Dredze and **Koby Crammer**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104 USA
{mdredze,crammer}@cis.upenn.edu

Abstract

NLP tasks are often domain specific, yet systems can learn behaviors across multiple domains. We develop a new multi-domain online learning framework based on parameter combination from multiple classifiers. Our algorithms draw from multi-task learning and domain adaptation to adapt multiple source domain classifiers to a new target domain, learn across multiple similar domains, and learn across a large number of disparate domains. We evaluate our algorithms on two popular NLP domain adaptation tasks: sentiment classification and spam filtering.

1 Introduction

Statistical classifiers routinely process millions of websites, emails, blogs and other text every day. Variability across different data sources means that training a single classifier obscures differences and separate classifiers ignore similarities. Similarly, adding new domains to existing systems requires adapting existing classifiers.

We present new online algorithms for three *multi-domain learning* scenarios: adapting existing classifiers to new domains, learning across multiple similar domains and scaling systems to many disparate domains. Multi-domain learning combines characteristics of both multi-task learning and domain adaptation and drawing from both areas, we develop a multi-classifier parameter combination technique for confidence-weighted (CW) linear classifiers (Dredze et al., 2008). We focus on online algorithms that scale to large amounts of data.

Next, we describe multi-domain learning and review the CW algorithm. We then consider our three settings using multi-classifier parameter combination. We conclude with related work.

2 Multi-Domain Learning

In online multi-domain learning, each instance $\boldsymbol{x}$ is drawn from a domain d specific distribution $\boldsymbol{x} \sim \mathcal{D}_d$ over a vectors space $\mathbb{R}^N$ and labeled with a domain specific function f_d with label $y \in \{-1, +1\}$ (for binary classification.) On round i the classifier receives instance $\boldsymbol{x}_i$ and domain identifier d_i and predicts label $\hat{y}_i \in \{-1, +1\}$. It then receives the true label $y_i \in \{-1, +1\}$ and updates its prediction rule.

As an example, consider a multi-user spam filter, which must give high quality predictions for new users (without new user data), learn on multiple users simultaneously and scale to thousands of accounts. While a single classifier trained on all users would generalize across users and extend to new users, it would fail to learn user-specific preferences. Alternatively, separate classifiers would capture user-specific behaviors but would not generalize across users. The approach we take to solving multi-domain problems is to combine domain-specific classifiers. In the adaptation setting, we combine source domain classifiers for a new target domain. For learning across domains, we combine domain-specific classifiers and a shared classifier learned across all domains. For learning across disparate domains we learn which domain-specific and shared classifiers to combine.

Multi-domain learning combines properties of both multi-task learning and domain adaptation. As

Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 689–697,
Honolulu, October 2008. ©2008 Association for Computational Linguistics

in multi-task learning, we consider domains that are labeled with different classification functions. For example, one user may enjoy some emails that another user considers spam: differing in their classification function. The goal of multi-task learning is to generalize across tasks/domains (Dekel et al., 2006; Evgeniou and Pontil, 2004). Furthermore, as in domain adaptation, some examples are draw from different distributions. For example, one user may receive emails about engineering while another about art, differing in their distribution over features. Domain adaptation deals with these feature distribution changes (Blitzer et al., 2007; Jiang and Zhai, 2007). Our work combines these two areas by learning both across distributions and behaviors or functions.

3 Confidence-Weighted Linear Classifiers

Confidence-weighted (CW) linear classification (Dredze et al., 2008), a new online algorithm, maintains a probabilistic measure of parameter confidence, which may be useful in combining parameters from different domain distributions. We summarize CW learning to familiarize the reader.

Parameter confidence is formalized by a Gaussian distribution over weight vectors with mean $\boldsymbol{\mu} \in \mathbb{R}^N$ and diagonal covariance $\Sigma \in \mathbb{R}^{N\times N}$. The values μ_j and $\Sigma_{j,j}$ represent knowledge of and confidence in the parameter for feature j. The smaller $\Sigma_{j,j}$, the more confidence we have in the mean parameter value μ_j. In this work we consider diagonal covariance matrices to scale to NLP data.

A model predicts the highest probability label,

$$\arg\max_{y\in\{\pm 1\}} \Pr_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{\mu},\Sigma)} \left[y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i)\geq 0\right] \ .$$

The Gaussian distribution over parameter vectors $\boldsymbol{w}$ induces a univariate Gaussian distribution over the score $S_i = \boldsymbol{w}\cdot\boldsymbol{x}_i$ parameterized by $\boldsymbol{\mu}$, Σ and the instance $\boldsymbol{x}_i$: $S_i \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right)$, with mean $\mu_i = \boldsymbol{\mu}\cdot\boldsymbol{x}_i$ and variance $\sigma_i^2 = \boldsymbol{x}_i^\top \Sigma \boldsymbol{x}_i$.

The CW algorithm is inspired by the Passive Aggressive (PA) update (Crammer et al., 2006) — which ensures a positive margin while minimizing parameter change. CW replaces the Euclidean distance used in the PA update with the Kullback-Leibler (KL) divergence over Gaussian distributions. It also replaces the minimal margin constraint with a minimal probability constraint: with some given probability $\eta \in (0.5, 1]$ a drawn classifier will assign the correct label. This strategy yields the following objective solved on each round of learning:

$$\begin{aligned} &\min \mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{\mu},\Sigma\right) \parallel \mathcal{N}\left(\boldsymbol{\mu}_i,\Sigma_i\right)\right) \\ &\text{s.t.} \Pr\left[y_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i\right)\geq 0\right]\geq\eta \ , \end{aligned}$$

where $(\boldsymbol{\mu}_i, \Sigma_i)$ are the parameters on round i and $\boldsymbol{w} \sim \mathcal{N}\left(\boldsymbol{\mu},\Sigma\right)$. The constraint ensures that the resulting parameters $\left(\boldsymbol{\mu}_{i+1},\Sigma_{i+1}\right)$ will correctly classify $\boldsymbol{x}_i$ with probability at least η. For convenience we write $\phi = \Phi^{-1}(\eta)$, where Φ is the cumulative function of the normal distribution. The optimization problem above is not convex, but a closed form approximation of its solution has the following additive form: $\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \alpha_i y_i \Sigma_i \boldsymbol{x}_i$ and $\Sigma_{i+1}^{-1} = \Sigma_i^{-1} + 2\alpha_i\phi\boldsymbol{x}_i\boldsymbol{x}_i^\top$ for,

$$\alpha_i = \frac{-(1+2\phi\mu_i)+\sqrt{(1+2\phi\mu_i)^2-8\phi\left(\mu_i-\phi\sigma_i^2\right)}}{4\phi\sigma_i^2} \ .$$

Each update changes the feature weights $\boldsymbol{\mu}$, and increases confidence (variance Σ always decreases). We employ CW classifiers since they provide confidence estimates, which are useful for classifier combination. Additionally, since we require per-parameter confidence estimates, other confidence based classifiers are not suitable for this setting.

4 Multi-Classifier Parameter Combination

The basis of our approach to multi-domain learning is to combine the parameters of CW classifiers from separate domains while respecting parameter confidence. A combination method takes M CW classifiers each parameterized by its own mean and variance parameters $\{(\boldsymbol{\mu}^m, \Sigma^m)\}_{m=1}^M$ and produces a single combined classifier $(\boldsymbol{\mu}^c, \Sigma^c)$. A simple technique would be to average the parameters of classifiers into a new classifier. However, this ignores the difference in feature distributions. Consider for example that the weight associated with some word in a source classifier has a value of 0. This could either mean that the word is very rare or that it is neutral for prediction (like the work "the"). The information captured by the variance parameter allow us to distinguish between the two cases: an high-variance indicates a lack of confidence in the value of the weight vectors because of small number of examples (first case), and vise-versa, small-variance indicates that the value of the weight is based on plenty of evidence. We favor combinations sensitive to this distinction.

Since CW classifiers are Gaussian distributions, we formalize classifier parameter combination as finding a new distribution that minimizes the weighted-divergence to a set of given distributions:

$$(\boldsymbol{\mu}^c, \Sigma^c) = \arg\min \sum_m^M \mathbb{D}((\boldsymbol{\mu}^c, \Sigma^c)||(\boldsymbol{\mu}^m, \Sigma^m)\ ;\ \mathbf{b}^m) ,$$

where (since Σ is diagonal),

$$\mathbb{D}((\boldsymbol{\mu}^c, \Sigma^c)||(\boldsymbol{\mu}, \Sigma)\ ;\ \mathbf{b}) = \\ \sum_f^N b_f D((\boldsymbol{\mu}_f^c, \Sigma_{f,f}^c)||(\boldsymbol{\mu}_f, \Sigma_{f,f}))\ .$$

The (classifier specific) importance-weights $\mathbf{b}^m \in \mathbb{R}_+^N$ are used to weigh certain parameters of some domains differently in the combination. When $\mathbb{D}$ is the Euclidean distance (L2), we have,

$$D((\boldsymbol{\mu}_f^c, \Sigma_{f,f}^c)||(\boldsymbol{\mu}_f, \Sigma_{f,f})) = \\ (\boldsymbol{\mu}_f^c - \boldsymbol{\mu}_f)^2 + (\Sigma_{f,f}^c - \Sigma_{f,f})^2\ .$$

and we obtain:

$$\boldsymbol{\mu}_f^c = \frac{1}{\sum_m^M b_f^m} \sum_m^M b_f^m \boldsymbol{\mu}_f^m,$$

$$\Sigma_{f,f}^c = \frac{1}{\sum_{m\in M} b_f^m} \sum_m^M b_f^m \Sigma_{f,f}^m\ . \quad (1)$$

Note that this is a (weighted) average of parameters. The other case we consider is when $\mathbb{D}$ is a weighted KL divergence we obtain a weighting of $\boldsymbol{\mu}$ by Σ^{-1}:

$$\boldsymbol{\mu}_f^c = \left(\sum_m^M (\Sigma_{f,f}^m)^{-1} b_f^m\right)^{-1} \sum_m^M (\Sigma_{f,f}^m)^{-1} \boldsymbol{\mu}_f^m b_f^m$$

$$(\Sigma^c)^{-1} = \left(M \sum_m^M b_f^m\right)^{-1} \sum_m^M (\Sigma_f^m)^{-1} {b_f}^m\ . \quad (2)$$

While each parameter is weighed by its variance in the KL, we can also explicitly encode this behavior as $b_f^m = a - \Sigma_{f,f}^m \geq 0$, where a is the initialization value for $\Sigma_{f,f}^m$. We call this weighting "variance" as opposed to a uniform weighting of parameters ($b_f^m = 1$). We therefore have two combination methods (L2 and KL) and two weighting methods (uniform and variance).

5 Datasets

For evaluation we selected two domain adaptation datasets: spam (Jiang and Zhai, 2007) and sentiment (Blitzer et al., 2007). The spam data contains two tasks, one with three users (task A) and one with 15 (task B). The goal is to classify an email (bag-of-words) as either spam or ham (not-spam) and each user may have slightly different preferences and features. We used 700 and 100 training messages for each user for task A and B respectively and 300 test emails for each user.

The sentiment data contains product reviews from Amazon for four product types: books, dvds, electronics and kitchen appliances and we extended this with three additional domains: apparel, music and videos. We follow Blitzer *et. al.* for feature extraction. We created different datasets by modifying the decision boundary using the ordinal rating of each instance (1-5 stars) and excluding boundary instances. We use four versions of this data:

- **All** - 7 domains, one per product type
- **Books** - 3 domains of books with the binary decision boundary set to 2, 3 and 4 stars
- **DVDs** - Same as *Books* but with DVD reviews
- **Books+DVDs** - Combined *Books* and *DVDs*

The *All* dataset captures the typical domain adaptation scenario, where each domain has the same decision function but different features. *Books* and *DVDs* have the opposite problem: the same features but different classification boundaries. *Books+DVDs* combines both issues. Experiments use 1500 training and 100 test instances per domain.

6 Multi-Domain Adaptation

We begin by examining the typical domain adaptation scenario, but from an online perspective since learning systems often must adapt to new users or domains quickly and with no training data. For example, a spam filter with separate classifiers trained on each user must also classify mail for a new user. Since other user's training data may have been deleted or be private, the existing classifiers must be combined for the new user.

	Target Domain	Train				L2		KL	
		All Src	*Target*	*Best Src*	*Avg Src*	*Uniform*	*Variance*	*Uniform*	*Variance*
Spam	user0	3.85	1.80	4.80	8.26	5.25	4.63	4.53	**4.32**
	user1	3.57	3.17	4.28	6.91	4.53	**3.80**	4.23	3.83
	user2	3.30	2.40	3.77	5.75	4.75	**4.60**	4.93	4.67
Sentiment	apparel	12.32	12.02	14.12	21.15	14.03	**13.18**	13.50	13.48
	books	16.85	18.95	22.95	25.76	19.58	**18.63**	19.53	19.05
	dvd	13.65	17.40	17.30	21.89	15.53	**13.73**	14.48	14.15
	kitchen	13.65	14.40	15.52	22.88	16.68	15.10	14.78	**14.02**
	electronics	15.00	14.93	15.52	23.84	18.75	17.37	17.45	**16.82**
	music	18.20	18.30	20.75	24.19	18.38	**17.83**	18.10	18.22
	video	17.00	19.27	19.43	25.78	17.13	**16.25**	16.33	16.42

Table 1: Test error for multi-source adaptation on sentiment and spam data. Combining classifiers improves over selecting a single classifier a priori (*Avg Src*).

We combine the existing user-specific classifiers into a single new classifier for a new user. Since nothing is known about the new user (their decision function), each source classifier may be useful. However, feature similarity – possibly measured using unlabeled data – could be used to weigh source domains. Specifically, we combine the parameters of each classifier according to their confidence using the combination methods described above.

We evaluated the four combination strategies – L2 vs. KL, uniform vs. variance – on spam and sentiment data. For each evaluation, a single domain was held out for testing while separate classifiers were trained on each source domain, i.e. no target training. Source classifiers are then combined and the combined classifier is evaluated on the test data (400 instances) of the target domain. Each classifier was trained for 5 iterations over the training data (to ensure convergence) and each experiment was repeated using 10-fold cross validation. The CW parameter ϕ was tuned on a single randomized run for each experiment. We include several baselines: training on target data to obtain an upper bound on performance (*Target*), training on all source domains together, a useful strategy if all source data is maintained (*All Src*), selecting (with omniscience) the best performing source classifier on target data (*Best Src*), and the expected real world performance of randomly selecting a source classifier (*Avg Src*).

While at least one source classifier achieved high performance on the target domain (*Best Src*), the correct source classifier cannot be selected without target data and selecting a random source classifier yields high error. In contrast, a combined classifier almost always improved over the best source domain classifier (table 1). That some of our results improve over the best training scenario is likely caused by increased training data from using multiple domains. Increases over all available training data are very interesting and may be due to a regularization effect of training separate models.

The L2 methods performed best and KL improved 7 out of 10 combinations. Classifier parameter combination can clearly yield good classifiers without prior knowledge of the target domain.

7 Learning Across Domains

In addition to adapting to new domains, multi-domain systems should learn common behaviors across domains. Naively, we can assume that the domains are either sufficiently similar to warrant one classifier or different enough for separate classifiers. The reality is often more complex. Instead, we maintain shared and domain-specific parameters and combine them for learning and prediction.

Multi-task learning aims to learn common behaviors across related problems, a similar goal to multi-domain learning. The primary difference is the nature of the domains/tasks: in our setting each domain is the same task but differs in the types of features in addition to the decision function. A multi-task approach can be adapted to our setting by using our classifier combination techniques.

	Spam		*Sentiment*			
Method	*Task A*	*Task B*	*Books*	*DVD*	*Books+DVD*	*All*
Single	3.88	8.75	23.7	25.11	23.26	16.57
Separate	5.46	14.53	22.22	21.64	21.23	21.89
Feature Splitting	4.16	8.93	15.65	16.20	14.60	17.45
MDR	4.09	9.18	15.65	15.12	13.76	17.45
MDR+L2	4.27	8.61	**12.70**	14.95	12.73	**17.16**
MDR+L2-Var	**3.75**	**7.52**	12.90	14.21	**12.52**	17.37
MDR+KL	4.32	9.22	13.51	**13.81**	13.32	17.20
MDR+KL-Var	4.02	8.70	14.93	14.03	14.22	18.40

Table 2: Online training error for learning across domains.

	Spam		*Sentiment*			
Method	*Task A*	*Task B*	*Books*	*DVD*	*Books+DVD*	*All*
Single	2.11	5.60	18.43	18.67	19.08	14.09
Separate	2.43	8.5	18.87	15.97	16.45	17.23
Feature Splitting	1.94	5.51	9.97	9.70	9.05	14.73
MDR	1.94	5.69	9.97	8.33	8.20	14.73
MDR+L2	**1.87**	5.16	6.63	7.97	7.62	**14.20**
MDR+L2-Var	1.90	**4.78**	**6.40**	7.83	**7.30**	14.33
MDR+KL	1.94	5.61	8.37	**7.07**	8.43	14.60
MDR+KL-Var	1.97	5.46	9.40	7.50	8.05	15.50

Table 3: Test data error: learning across domains (MDR) improves over the baselines and Daumé (2007).

We seek to learn domain specific parameters guided by shared parameters. Dekel et al. (2006) followed this approach for an online multi-task algorithm, although they did not have shared parameters and assumed that a training round comprised an example from each task. Evgeniou and Pontil (2004) achieved a similar goal by using shared parameters for multi-task regularization. Specifically, they assumed that the weight vector for problem d could be represented as $w^c = w^d + w^s$, where w^d are task specific parameters and w^s are shared across all tasks. In this framework, all tasks are close to some underlying mean w^s and each one deviates from this mean by w^d. Their SVM style multi-task objective minimizes the loss of w^c and the norm of w^d and w^s, with a tradeoff parameter allowing for domain deviance from the mean. The simple domain adaptation algorithm of feature splitting used by Daumé (2007) is a special case of this model where the norms are equally weighted. An analogous CW objective is:

$$\min \frac{1}{\lambda_1}\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{\mu}^d, \Sigma^d\right) \parallel \mathcal{N}\left(\boldsymbol{\mu}_i^d, \Sigma_i^d\right)\right) + \frac{1}{\lambda_2}\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{\mu}^s, \Sigma^s\right) \parallel \mathcal{N}\left(\boldsymbol{\mu}_i^s, \Sigma_i^s\right)\right)$$
$$\text{s.t. } \Pr_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{\mu}^c, \Sigma^c)}\left[y_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i\right) \geq 0\right] \geq \eta\,. \quad (3)$$

$\left(\boldsymbol{\mu}^d, \Sigma^d\right)$ are the parameters for domain d, $(\boldsymbol{\mu}^s, \Sigma^s)$ for the shared classifier and $(\boldsymbol{\mu}^c, \Sigma^c)$ for the combination of the domain and shared classifiers. The parameters are combined via (2) with only two elements summed - one for the shared parameters s and the other for the domain parameters d. This captures the intuition of Evgeniou and Pontil: updates enforce the learning condition on the combined parameters and minimize parameter change. For convenience, we rewrite $\lambda_2 = 2 - 2\lambda_1$, where $\lambda_1 \in [0, 1]$. If classifiers are combined using the sum of the individual weight vectors and $\lambda_1 = 0.5$, this is identical to feature splitting (Daumé) for CW classifiers.

The domain specific and shared classifiers can be updated using the closed form solution to (3) as:

$$\begin{aligned} \boldsymbol{\mu}^s &= \boldsymbol{\mu}_i^s + \lambda_2 \alpha y_i \Sigma^c \boldsymbol{x}_i \\ (\Sigma^s)^{-1} &= (\Sigma_i^s)^{-1} + 2\lambda_2 \alpha \phi \boldsymbol{x}_i \boldsymbol{x}_i^T \\ \boldsymbol{\mu}^d &= \boldsymbol{\mu}_i^d + \lambda_1 \alpha y_i \Sigma_i^c \boldsymbol{x}_i \\ (\Sigma^d)^{-1} &= (\Sigma_i^d)^{-1} + 2\lambda_1 \alpha \phi \boldsymbol{x}_i \boldsymbol{x}_i^T \end{aligned} \tag{4}$$

We call this objective Multi-Domain Regularization (MDR). As before, the combined parameters are produced by one of the combination methods. On each round, the algorithm receives instance $\boldsymbol{x}_i$ and domain d_i for which it creates a combined classifier $(\boldsymbol{\mu}^c, \Sigma^c)$ using the shared $(\boldsymbol{\mu}^s, \Sigma^s)$ and domain specific parameters $\left(\boldsymbol{\mu}^d, \Sigma^d\right)$. A prediction is issued using the standard linear classifier prediction rule $\text{sign}(\boldsymbol{\mu}^c \cdot \boldsymbol{x})$ and updates follow (4). The effect is that features similar across domains quickly converge in the shared classifier, sharing information across domains. The combined classifier reflects shared and domain specific parameter confidences: weights with low variance (i.e. greater confidence) will contribute more.

We evaluate MDR on a single pass over a stream of instances from multiple domains, simulating a real world setting. Parameters λ_1 and ϕ are iteratively optimized on a single randomized run for each dataset. All experiments use 10-fold CV. In addition to evaluating the four combination methods with MDR, we evaluate the performance of a single classifier trained on all domains (*Single*), a separate classifier trained on each domain (*Separate*), *Feature Splitting* (Daumé) and feature splitting with optimized λ_1 (*MDR*). Table 3 shows results on test data and table 2 shows online training error.

In this setting, L2 combinations prove best on 5 of 6 datasets, with the variance weighted combination doing the best. MDR (optimizing λ_1) slightly improves over feature splitting, and the combination methods improve in every case. Our best result is statistically significant compared to *Feature Splitting* using McNemar's test ($p = .001$) for *Task B*, *Books*, *DVD*, *Books+DVD*. While a single or separate classifiers have a different effect on each dataset, MDR gives the best performance overall.

8 Learning in Many Domains

So far we have considered settings with a small number of similar domains. While this is typical of multi-task problems, real world settings present many domains which do not all share the same behaviors. Online algorithms scale to numerous examples and we desire the same behavior for numerous domains. Consider a spam filter used by a large email provider, which filters billions of emails for millions of users. Suppose that spammers control many accounts and maliciously label spam as legitimate. Alternatively, subsets of users may share preferences. Since behaviors are not consistent across domains, shared parameters cannot be learned. We seek algorithms robust to this behavior.

Since subsets of users share behaviors, these can be learned using our MDR framework. For example, discovering spammer and legitimate mail accounts would enable intra-group learning. The challenge is the online discovery of these subsets while learning model parameters. We augment the MDR framework to additionally learn this mapping.

We begin by generalizing MDR to include k shared classifiers instead of a single set of shared parameters. Each set of shared parameters represents a different subset of domains. If the corresponding shared parameters are known for a domain, we could use the same objective (3) and update (4) as before. If there are many fewer shared parameters than domains ($k \ll D$), we can benefit from multi-domain learning. Next, we augment the learning algorithm to learn a mapping between the domains and shared classifiers. Intuitively, a domain should be mapped to shared parameters that correctly classify that domain. A common technique for learning such experts in the Weighted Majority algorithm (Littlestone and Warmuth, 1994), which weighs a mixture of experts (classifiers). However, since we require a hard assignment — pick a single shared parameter set s — rather than a mixture, the algorithm reduces to picking the classifier s with the fewest mistakes in predicting domain d. This requires tracking the number of mistakes made by each shared classifier on each domain once a label is revealed. For learning, the shared classifier with the fewest mistakes for a domain is selected for an MDR update. Classifier ties are broken randomly. While we experi-

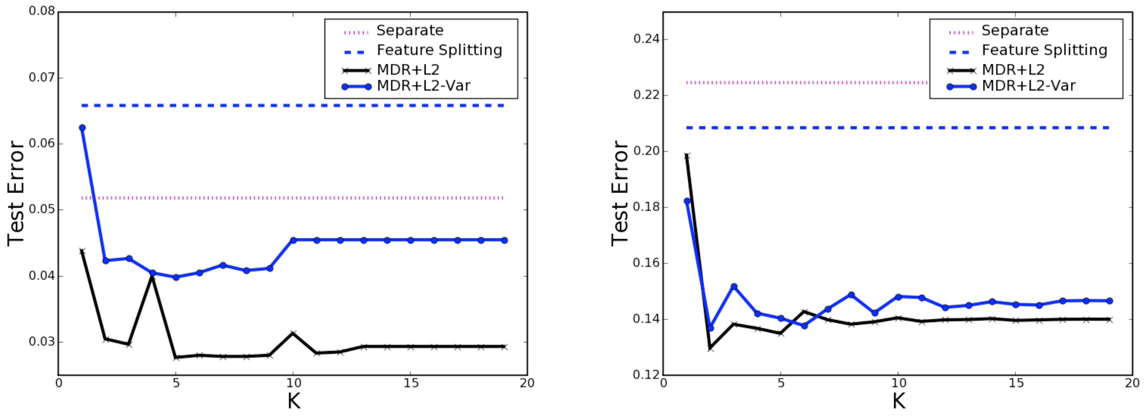

Figure 1: Learning across many domains - spam (left) and sentiment (right) - with MDR using k shared classifiers.

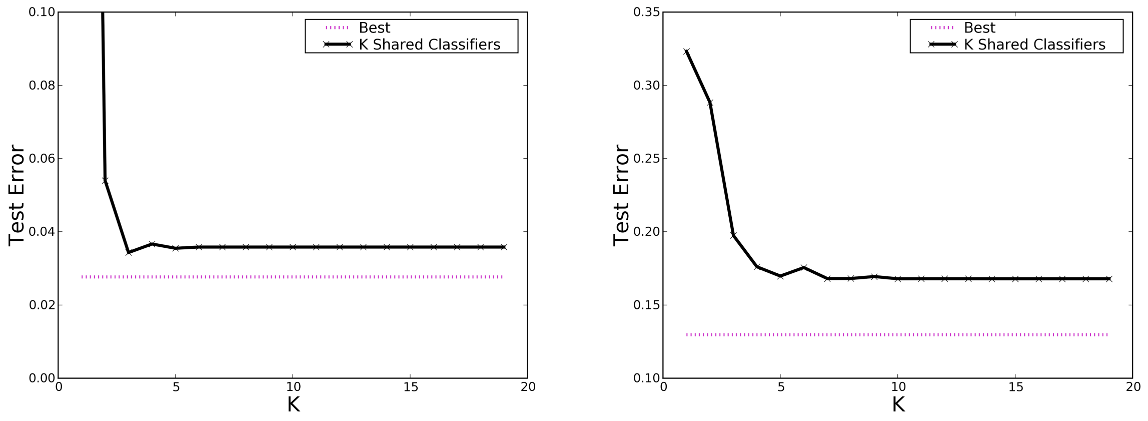

Figure 2: Learning across many domains - spam (left) and sentiment (right) - with no domain specific parameters.

mented with more complex techniques, this simple method worked well in practice. When a new domain is added to the system, it takes fewer examples to learn which shared classifier to use instead of learning a new model from scratch.

While this approach adds another free parameter (k) that can be set using development data, we observe that k can instead be fixed to a large constant. Since only a single shared classifier is updated each round, the algorithm will favor selecting a previously used classifier as opposed to a new one, using as many classifiers as needed but not scaling up to k. This may not be optimal, but it is a simple.

To evaluate a larger number of domains, we created many varying domains using spam and sentiment data. For spam, 6 email users were created by splitting the 3 task A users into 2 users, and flipping the label of one of these users (a malicious user), yielding 400 train and 100 test emails per user. For sentiment, the book domain was split into 3 groups with binary boundaries at a rating of 2, 3 or 4. Each of these groups was split into 8 groups of which half had their labels flipped, creating 24 domains. The same procedure was repeated for DVD reviews but for a decision boundary of 3, 6 groups were created, and for a boundary of 2 and 4, 3 groups were created with 1 and 2 domains flipped respectively, resulting in 12 DVD domains and 36 total domains with various decision boundaries, features, and inverted decision functions. Each domain used 300 train and 100 test instances. 10-fold cross validation with one training iteration was used to train models on these two datasets. Parameters were optimized as before. Experiments were repeated for various settings of k. Since L2 performed well before, we evaluated MDR+L2 and MDR+L2-Var.

The results are shown in figure 1. For both spam and sentiment adding additional shared parameters beyond the single shared classifier significantly reduces error, with further reductions as k increases. This yields a 45% error reduction for spam and a 38% reduction for sentiment over the best baseline. While each task has an optimal k (about 5 for spam, 2 for sentiment), larger values still achieve low error, indicating the flexibility of using large k values.

While adding parameters clearly helps for many domains, it may be impractical to keep domain-specific classifiers for thousands or millions of domains. In this case, we could eliminate the domain-specific classifiers and rely on the k shared classifiers only, learning the domain to classifier mapping. We compare this approach using the best result from MDR above, again varying k. Figure 2 shows that losing domain-specific parameters hurts performance, but is still an improvement over baseline methods. Additionally, we can expect better performance as the number of similar domains increases. This may be an attractive alternative to keeping a very large number of parameters.

9 Related Work

Multi-domain learning intersects two areas of research: domain adaptation and multi-task learning. In domain adaptation, a classifier trained for a source domain is transfered to a target domain using either unlabeled or a small amount of labeled target data. Blitzer et al. (2007) used structural correspondence learning to train a classifier on source data with new features induced from target unlabeled data. In a complimentary approach, Jiang and Zhai (2007) weighed training instances based on their similarity to unlabeled target domain data. Several approaches utilize source data for training on a limited number of target labels, including feature splitting (Daumé, 2007) and adding the source classifier's prediction as a feature (Chelba and Acero, 2004). Others have considered transfer learning, in which an existing domain is used to improve learning in a new domain, such as constructing priors (Raina et al., 2006; Marx et al., 2008) and learning parameter functions for text classification from related data (Do and Ng, 2006). These methods largely require batch learning, unlabeled target data, or available source data at adaptation. In contrast, our algorithms operate purely online and can be applied when no target data is available.

Multi-task algorithms, also known as inductive transfer, learn a set of related problems simultaneously (Caruana, 1997). The most relevant approach is that of Regularized Multi-Task Learning (Evgeniou and Pontil, 2004), which we use to motivate our online algorithm. Dekel et al. (2006) gave a similar online approach but did not use shared parameters and assumed multiple instances for each round. We generalize this work to both include an arbitrary classifier combination and many shared classifiers. Some multi-task work has also considered the grouping of tasks similar to our learning of domain subgroups (Thrun and O'Sullivan, 1998; Bakker and Heskes, 2003).

There are many techniques for combining the output of multiple classifiers for ensemble learning or mixture of experts. Kittler et al. (Mar 1998) provide a theoretical framework for combining classifiers. Some empirical work has considered adding versus multiplying classifier output (Tax et al., 2000), using local accuracy estimates for combination (Woods et al., 1997), and applications to NLP tasks (Florian et al., 2003). However, these papers consider combining classifier output for prediction. In contrast, we consider parameter combination for both prediction and learning.

10 Conclusion

We have explored several multi-domain learning settings using CW classifiers and a combination method. Our approach creates a better classifier for a new target domain than selecting a random source classifier a prior, reduces learning error on multiple domains compared to baseline approaches, can handle many disparate domains by using many shared classifiers, and scales to a very large number of domains with a small performance reduction. These scenarios are realistic for NLP systems in the wild. This work also raises some questions about learning on large numbers of disparate domains: can a hierarchical online clustering yield a better representation than just selecting between k shared parameters? Additionally, how can prior knowledge about domain similarity be included into the combination methods? We plan to explore these questions in future work.

Acknowledgements This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185.

References

B. Bakker and T. Heskes. 2003. Task clustering and gating for bayesian multi–task learning. *Journal of Machine Learning Research*, 4:83–99.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics (ACL)*.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28:41–75.

Ciprian Chelba and Alex Acero. 2004. Adaptation of max- imum entropy classifier: Little data can help a lot. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Hal Daumé. 2007. Frustratingly easy domain adaptation. In *Association for Computational Linguistics (ACL)*.

Ofer Dekel, Philip M. Long, and Yoram Singer. 2006. Online multitask learning. In *Conference on Learning Theory (COLT)*.

Chuong B. Do and Andrew Ng. 2006. Transfer learning for text classification. In *Advances in Neural Information Processing Systems (NIPS)*.

Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. In *International Conference on Machine Learning (ICML)*.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi-task learning. In *Conference on Knowledge Discovery and Data Mining (KDD)*.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Conference on Computational Natural Language Learning (CONLL)*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Association for Computational Linguistics (ACL)*.

J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. Mar 1998. On combining classifiers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(3):226–239.

N. Littlestone and M. K. Warmuth. 1994. The weighted majority algorithm. *Information and Computation*, 108:212–261.

Zvika Marx, Michael T. Rosenstein, Thomas G. Dietterich, and Leslie Pack Kaelbling. 2008. Two algorithms for transfer learning. In *Inductive Transfer: 10 years later*.

Rajat Raina, Andrew Ng, and Daphne Koller. 2006. Constructing informative priors using transfer learning. In *International Conference on Machine Learning (ICML)*.

David M. J. Tax, Martijn van Breukelen, Robert P. W. Duina, and Josef Kittler. 2000. Combining multiple classifiers by averaging or by multiplying? *Pattern Recognition*, 33(9):1475–1485, September.

S. Thrun and J. O'Sullivan. 1998. Clustering learning tasks and the selective cross–task transfer of knowledge. In S. Thrun and L.Y. Pratt, editors, *Learning To Learn*. Kluwer Academic Publishers.

Kevin Woods, W. Philip Kegelmeyer Jr., and Kevin Bowyer. 1997. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410.