# Domain Adaptation of Conditional Probability Models via Feature Subsetting

Sandeepkumar Satpal and Sunita Sarawagi*

IIT Bombay

**Abstract.** The goal in domain adaptation is to train a model using labeled data sampled from a domain different from the target domain on which the model will be deployed. We exploit unlabeled data from the target domain to train a model that maximizes likelihood over the training sample while minimizing the distance between the training and target distribution. Our focus is conditional probability models used for predicting a label structure $\mathbf{y}$ given input $\mathbf{x}$ based on features defined jointly over $\mathbf{x}$ and $\mathbf{y}$. We propose practical measures of divergence between the two domains based on which we penalize features with large divergence, while improving the effectiveness of other less deviant correlated features. Empirical evaluation on several real-life information extraction tasks using Conditional Random Fields (CRFs) show that our method of domain adaptation leads to significant reduction in error.

## 1 Introduction

Most statistical learning techniques are based on the assumption that the training data is representative of the distribution on which the trained model is deployed. This assumption gets routinely broken in applications like information extraction, speech recognition, text classification, and opinion mining that are being increasingly used at large scales. In most such applications, an offline phase is used to collect carefully labeled data for training. However, the settings during deployment could be highly varied with little or no labeled data for that setting. For example, it is easy to find plenty of labeled data for named entity recognition in news articles but our goal might be to recognize person names from blogs. It is not easy to find labeled data for blogs but there is no dearth of unlabeled data.

Our goal in domain adaptation is to use labeled data from some domain to train a model that maximizes accuracy in a target domain for which we only have unlabeled data available. We concentrate on adapting structured learning tasks that model the conditional probability of a predicted structure $\mathbf{y}$ given input $\mathbf{x}$ as a linear exponential function of features defined over $\mathbf{x}$ and $\mathbf{y}$. A logistic classifier is a special case of such models where the predicted structure is a single discrete class label. Such conditional models allow users the flexibility of defining features without bothering about whether they are correlated or not.

* Contact author, sunita@iitb.ac.in

Therefore, most real-life applications of these models involve a large number of features, contributing in varying strengths to the prediction task. With overfitting avoided using a suitable regularizer, these models provide state-of-the-art accuracy values in settings where features behave the same way in the training and target domain [1,2,3]. However, we observed that such models are rather brittle in that they perform very poorly on target data with even a small subset of features distorted in spite of other highly correlated features remaining intact.

We show how to detect features with large divergence in the two domains and penalize the more distorted features so that other less deviant correlated features start exerting a larger influence. A challenge is designing a reliable measure of divergence given only unlabeled data from the target domain whereas our features are defined over function of both labels $\mathbf{y}$ and input $\mathbf{x}$. We propose a measure of distortion as a function of the difference in expectation over the target samples and the trained conditional model. We formulate this as an optimization problem and present efficient algorithms for solving it. On seven real-life datasets, we show that our domain adapted classifier provides much higher accuracy than an unadapted model.

The rest of the paper is organized as follows. We discuss related work in Section 2. We describe our basic learning model in Section 3 and present our approach to domain adaptation in Section 4. We report results of an empirical evaluation of our model in Section 5.

## 2 Related work

*Transfer learning:* In transfer learning [4,5,6,7] the goal is to use available training data from a related domain, along with training data from the target domain, to train the target classifier. A popular technique is to use the classifier in the related domain to define a prior [4,6,7] for the classifier trained using the in-domain data. For example, [7] proposes to first create a classifier using training data from the related domain. The output parameters are used as the mean of a Gaussian prior for the second classifier trained using labeled data of the target domain. A different type of prior is defined in [8] where the prior is used to give more importance to features that are useful across domains. Another interesting approach is based on replicating features so that shared features exploit labeled data from both domains whereas domain-specific features are trained only using in-domain data [5]. Our goal is different in that we do not have any labeled data from the target domain. Transfer learning is supervised domain adaptation whereas we are interested in unsupervised domain adaptation.

*Structural correspondence learning:* A recent proposal [9,10] for unsupervised domain adaptation is to define new features that capture the correspondence between features in the two domains. The new features are weights of "mini" classifiers that predict value of user-chosen anchor features that remain invariant across the domains. Successful domain adaptation will require both addition and deletion of features. Deletion is required for features that are missing or severely

distorted, whereas when features are substituted, for example, the inter-author separator is changed from "comma" to a "new line", addition of features that capture their correspondence is more useful. Given that most structured learning tasks involve many correlated features, careful feature subsetting could lead to significant accuracy gains, as we show in this paper.

*Robust learning:* A different approach to handling features that are distorted in the test data is to learn classifiers that are robust to limited amounts of distortion. For example, [11] shows how to create SVM classifiers that provide good worst case performance with the deletion of any subset of features of size no more than $k$. In robust learning a model is trained once unlike in the case of domain adaptation where the model is *retrained* to adapt to any systematic difference between the two domains.

*Correcting sample selection bias:* In some cases, the training distribution fails to be representative of the test distribution because of a selection bias in the training instances, for example due to active learning. A popular strategy to correct for the bias [12,13] is to weight training examples differentially. Such methods are not likely to be useful for domain adaptation because all instances from the train domain could have very small probability in the target domain and the real issue is that of choosing the right representation through feature reweighting rather than instance reweighting.

In summary, the problem of unsupervised domain adaptation is related to, but distinct, from many problems in machine learning. To the best of our knowledge, domain adaptation via feature subsetting has not been addressed before in the literature.

## 3 Background

### 3.1 The basic learning model

We consider conditional models of structure learning where the goal is to predict a label $\mathbf{y}$ from a structured space $\mathcal{Y}$ given an input $\mathbf{x}$. We assume a feature vector representation $\mathcal{F} : (\mathbf{x}, \mathbf{y}) \mapsto \mathcal{R}^K$ that maps any $(\mathbf{x}, \mathbf{y})$ pair to a vector of $K$ reals. The conditional probability model is a log-linear function over these features. Thus, $\Pr(\mathbf{y}|\mathbf{x})$ is this Gibbs distribution

$$\Pr(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{1}{z_{\mathbf{w}}(\mathbf{x})} \exp \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) \tag{1}$$

where $\mathbf{w}$ is the parameter vector of the model where the $k^{th}$ component $w_k$ is called the weight of feature $f_k$. The term $z_{\mathbf{w}}(\mathbf{x}) = \sum_{\mathbf{y}'} \exp \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}')$ is a normalizing constant.

In practice, each feature $f_k(\mathbf{x}, \mathbf{y})$ is defined as a sum of local features that apply over smaller subsets of variables. When the features decompose over cliques of an undirected graph on labels $\mathbf{y}$, we get Conditional Random Fields [1]. This

decomposition is exploited for efficient inference over the space of variables $\mathbf{y}$. For example, in information extraction, the underlying graph is a linear chain where features decompose over pairs of adjacent variables.

During training the goal is to maximize log-likelihood over a given training set $D = \{(\mathbf{x}_\ell, \mathbf{y}_\ell)\}_{\ell=1}^{N}$ expressed as

$$L(\mathbf{w}) = \sum_{\ell} \log \Pr(\mathbf{y}_\ell | \mathbf{x}_\ell, \mathbf{w}) = \sum_{\ell} (\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_\ell, \mathbf{y}_\ell) - \log z_{\mathbf{w}}(\mathbf{x}_\ell)) \tag{2}$$

We wish to find a $\mathbf{w}$ that maximizes $L(\mathbf{w})$. In practice, the norm of $\mathbf{w}$ is not allowed to grow too large to avoid overfitting. This is achieved by subtracting a regularization term $R(\mathbf{w}) = \frac{||\mathbf{w}||^\gamma}{\sigma^2}$ with $\gamma = 1$ or 2 and a user-provided variance $\sigma^2$. The resultant objective is convex, and can thus be maximized by gradient ascent, or one of many related methods.

During deployment, given an input $\mathbf{x}$, we predict a $\mathbf{y}$ for which $\Pr(\mathbf{y}|\mathbf{x})$ is maximum. The justification for this step is that the test data follows the same distribution as the training data, using which we learnt a $\mathbf{w}$ so as to maximize the probability of the correct prediction.

### 3.2 Train and target data distributions

In domain adaptation we need to deploy a model in a domain where the distribution of $(\mathbf{x}, \mathbf{y})$ is different from the distribution from which the training data was obtained. Let $\mathcal{D}$ denote the distribution from which the training sample $D$ was taken. Let $\mathcal{D}'$ denote the target distribution on which we wish to deploy the model. We do not have any labeled data from $\mathcal{D}'$, instead we have lots of unlabeled data $D'$. Let $D' = \{(\mathbf{x}_\ell)\}_{\ell=1}^{N'}$.

In domain adaptation our goal is to use both the labeled samples $D$ from $\mathcal{D}$ and the unlabeled samples $D'$ from distribution $\mathcal{D}'$ to train a model that maximizes accuracy on $\mathcal{D}'$. The accuracy in the $\mathcal{D}$ distribution is of no interest to us. Therefore the normal goal during CRF training of maximizing likelihood of $D$ is not justified anymore because $D$ is not representative of the distribution on which the model will be deployed. This is also what makes the problem different from semi-supervised learning where the labeled and unlabeled data come from the same distribution.

## 4 Domain adaptation

Our approach to domain adaptation is to choose a representation where the training and test distributions are close, and once that is achieved we can justify training a model to maximize accuracy on the labeled training domain. Our starting representation is the user provided feature vector $\mathbf{f}(\mathbf{x}, \mathbf{y})$. During domain adaptation we select the subset $S$ of features such that the distance between the train and target distributions is small in the projected space while maximizing

likelihood on the training data. Our ideal objective of maximizing likelihood of the target distribution $\mathcal{D}$ for which we have no labeled samples

$$\text{argmax}_{\mathbf{w}} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}'} \sum_{k} w_k f_k(\mathbf{x},\mathbf{y}) - \log z_{\mathbf{w}}(\mathbf{x}) \tag{3}$$

is replaced with the achievable objective

$$\begin{aligned} \text{argmax}_{\mathbf{w},S} \sum_{(\mathbf{x},\mathbf{y})\in D} \sum_{k\in S} w_k f_k(\mathbf{x},\mathbf{y}) - \log z_{\mathbf{w}}(\mathbf{x}) \\ \text{such that } \ \text{dist}(\mathcal{D},\mathcal{D}'|S,D,D') \leq \epsilon. \end{aligned} \tag{4}$$

where $\text{dist}(\mathcal{D},\mathcal{D}'|S,D,D')$ is a suitable measure of distance between the two domains in a representation corresponding to the features in set $S$ and as estimated from the labeled samples $D$ from $\mathcal{D}$ and unlabeled samples $D'$ from $\mathcal{D}'$.

### 4.1 Distance function

We next discuss how to measure the distance between the two distributions. A direct approach is to first estimate their full $(\mathbf{x},\mathbf{y})$ distributions using sample data and then measure the distance between the two distributions using some function like KL distance. This is often difficult and requires a lot of training data. One of the main reasons for the success of the conditional approach for structured learning tasks is that they do not require the modeling of the distribution over $\mathbf{x}$.

Recently, [13] proposed to correct for sample selection bias in the training data by reducing the difference in the mean of the $\mathbf{x}$ features in the training and target distribution. There are several reasons why this method will not work well in our setting. First, in structured learning settings, the feature representation is in terms of both $\mathbf{x}$ and $\mathbf{y}$. Even if, we consider the scalar classification problem where we simplify the feature representation to be a cross product of features defined over $\mathbf{x}$ and labels $\mathbf{y}$, we can obtain more accurate distance measures by comparing the $\mathbf{x}$ means of each $\mathbf{y}$ separately rather than collapsing them on single means. Also, the method proposed in [13] assumes that $\Pr(\mathbf{y}|\mathbf{x})$ is the same in the training and test distribution. In our case, we assume that there exist some representation under which the two distributions are the same, but this is not true for all representations. In particular, this is not true for the starting representation used during normal training.

We propose to compare the two distributions by comparing component-wise the means of the features in their $(\mathbf{x},\mathbf{y})$ space. Let $E^k_D$ and $E^k_{D'}$ denote the expected value of the $k^{th}$ feature under distributions $\mathcal{D}$ and $\mathcal{D}'$ respectively. For the training distribution, we estimate it empirically from the sample $D$ as $E^k_D = \sum_{(\mathbf{x}_\ell,\mathbf{y}_\ell)\in D} \frac{f_k(\mathbf{x}_\ell,\mathbf{y}_\ell)}{N}$. For the target distribution $\mathcal{D}'$ since in the sample $D'$ we have only $\mathbf{x}$ values, we use the expected value of the feature as calculated under the $\Pr(\mathbf{y}|\mathbf{x},\mathbf{w})$ distribution. Thus,

$$E^k_{D'} = \frac{1}{N'} \sum_{\mathbf{x}_\ell\in D'} \sum_{\mathbf{y}} f_k(\mathbf{x}_\ell,\mathbf{y}) \Pr(\mathbf{y}|\mathbf{x}_\ell,\mathbf{w}) \tag{5}$$

Using $\mathbf{E}_D$ and $\mathbf{E}_{D'}$, we replace $\text{dist}(\mathcal{D}, \mathcal{D}'|S, D, D')$ with the distance between the above sample means as $\sum_{k \in S} d(E^k_D, E^k_{D'})$. The precise form of the distance function will depend on the nature of the specific features. For example, for sparse binary features, it is useful to interpret the mean values as probability of occurrence of a binomial distribution. In such cases, distance measures like cross-entropy and the log-odds ratio seem meaningful [14]. When the features are arbitrary real values, a $L1$ or square distance would be more appropriate.

### 4.2 Overall objective

In terms of the new distance function, we can rewrite the objective as

$$\begin{aligned} \text{argmax}_{\mathbf{w},S} \sum_{(\mathbf{x},\mathbf{y}) \in D} \sum_{k \in S} w_k f_k(\mathbf{x}, \mathbf{y}) - \log z_{\mathbf{w}}(\mathbf{x}) \\ \text{such that } \sum_{k \in S} d(E^k_D, E^k_{D'}) \leq \epsilon. \end{aligned} \tag{6}$$

The above objective presents a difficult combinatorial optimization problem over the exponentially many subsets of features. We convert the discrete feature selection problem to a soft selection problem by rewriting the constraint $\sum_{k \in S} d(E^k_D, E^k_{D'}) \leq \epsilon$ as $\sum_{k=1}^{K} |w_k|^\gamma d(E^k_D, E^k_{D'}) \leq \epsilon'$. Also, using the Lagrange dual formulation, we push the constraints into the objective and get the equivalent objective for an appropriate value of $\lambda$ as

$$\text{argmax}_{\mathbf{w}} \sum_{(\mathbf{x},\mathbf{y}) \in D} \sum_{k} w_k f_k(\mathbf{x}, \mathbf{y}) - \log z_{\mathbf{w}}(\mathbf{x}) - \lambda \sum_{k} |w_k|^\gamma d(E^k_D, E^k_{D'}) \tag{7}$$

The above formulation has several intuitive interpretations. We can treat this as a standard accuracy-regularized training method with the only difference that the $w_k$ are weighted in proportional to the distance between the training and target distribution along the $k$-th feature component. A feature with a large distance should get a smaller weight. Another interpretation is in terms of prior distributions over the parameters where the variance is not constant over all features, as is normally the case, but is inversely proportional to the divergence of the feature over the two distributions. When $\gamma$ is 1 the prior is a Laplace distribution and when $\gamma = 2$ the prior is a Gaussian distribution with variance of the $k$th parameter as $\frac{1}{d(E^k_D, E^k_{D'})}$. So when the distance is large, the parameter is likely to stay close to its mean value of 0.

### 4.3 Training algorithm

We now discuss how we solve the optimization problem in Equation 7. For concreteness, we assume that $\gamma = 2$ and the distance function is the square distance defined as $d(E^k_D, E^k_{D'}) = (E^k_D - E^k_{D'})^2$. The final objective then becomes.

$$L(\mathbf{w}) = \text{argmax}_{\mathbf{w}} \sum_{(\mathbf{x},\mathbf{y}) \in D} (\sum_{k} w_k f_k(\mathbf{x}, \mathbf{y}) - \log z_{\mathbf{w}}(\mathbf{x})) - \lambda \sum_{k} w_k^2 (E^k_D - E^k_{D',\mathbf{w}})^2$$

where $E^k_{D',\mathbf{w}} = \frac{1}{|D'|}\sum_{\mathbf{x}_i \in D'}\sum_{\mathbf{y}} f_k(\mathbf{x}_i,\mathbf{y})\frac{\exp \mathbf{wf}(\mathbf{x}_i,\mathbf{y})}{z_{\mathbf{w}}(\mathbf{x}_i)}$. The above is a smooth differentiable function of $\mathbf{w}$. We can use standard gradient descent approaches to solve it. The gradient with respect to the $k^{th}$ parameter is

$$\frac{\partial L}{\partial w_k} = \sum_{(\mathbf{x},\mathbf{y})\in D} f_k(\mathbf{x},\mathbf{y}) - NE^k_{D,\mathbf{w}} - 2\lambda(w_k(E^k_D - E^k_{D',\mathbf{w}})^2 - \sum_j w_j^2(E^j_D - E^j_{D',\mathbf{w}})\frac{\partial E^j_{D',\mathbf{w}}}{\partial w_k})$$

where

$$\begin{aligned}\frac{\partial E^j_{D',\mathbf{w}}}{\partial w_k} &= \frac{1}{N'}\sum_{\mathbf{x}_i\in D'}\sum_{\mathbf{y}} f_j(\mathbf{x}_i,\mathbf{y})\frac{\exp \mathbf{wf}(\mathbf{x}_i,\mathbf{y})}{z_{\mathbf{w}}(\mathbf{x}_i)}(f_k(\mathbf{x}_i,\mathbf{y}) - \sum_{\mathbf{y}'} f_k(\mathbf{x}_i,\mathbf{y}')\frac{\exp \mathbf{wf}(\mathbf{x}_i,\mathbf{y}')}{z_{\mathbf{w}}(\mathbf{x}_i)})\\ &= \frac{1}{N'}\sum_{\mathbf{x}_i\in D'}\sum_{\mathbf{y}} f_j(\mathbf{x}_i,\mathbf{y})\Pr(\mathbf{y}|\mathbf{x}_i)(f_k(\mathbf{x}_i,\mathbf{y}) - \sum_{\mathbf{y}'} f_k(\mathbf{x}_i,\mathbf{y}')\Pr(\mathbf{y}'|\mathbf{x}_i))\\ &= (E^{jk}_{D',\mathbf{w}} - E^j_{D',\mathbf{w}}E^k_{D',\mathbf{w}})\end{aligned}$$

where $E^{jk}_{D'}$ is the expectation of the product of features $j$ and $k$ with respect to the empirical $\mathbf{x}$ distribution from $D'$ and $\Pr(\mathbf{y}|\mathbf{w},\mathbf{x})$. With respect to these distributions, the term $(E^{jk}_{D',\mathbf{w}} - E^j_{D',\mathbf{w}}E^k_{D',\mathbf{w}})$ represents the covariance between features $j$ and $k$. As in normal CRF training [1], we have to exploit the decomposability of the label space to evaluate these terms tractably.

There are two problem with the above objective.

1. The function is not convex, unlike the normal CRF objective with constant weighting of the regularizers.
2. The gradient is expensive to compute since the covariance terms are quadratic in the number of features. In typical structured learning tasks, for example in information extraction, the number of features tend to be very large.

We address both these issues by following a nested iterative approach to training. In each iteration, we fix feature distances with respect to the current values of the parameters and find the optimum value of the parameters treating the distance values as constant. This makes the inner optimization problem convex and linear in the number of features. We found that in practice with two or three iterations we get most of the benefit of complete training at significantly reduced cost.

## 5 Experiments

We evaluate the effectiveness of our proposed method on seven domain adaptation tasks constructed from the following four entity extraction benchmarks

*CoNLL 2003 dataset* The ConLL 2003 dataset [1] is a well-known benchmark for Named Entity Recognition where the goal is to extract entities like persons, organizations, and locations from news articles.

[1] `http://cnts.uia.ac.be/conll2003/ner/`

*Cora citations* Cora citations [3] consists of citations collected from the reference section of several academic papers. The extraction task is to find author names, titles, venue, and year.

*Cora headers* Cora headers [3] consists of headers of research papers covering fields like the title, author names, affiliations, and abstract of a paper. Even though headers and citations come from the same repository, the way authors and titles appear in paper headers is very different from the way they appear in paper citations, making it interesting for domain adaptation.

*Citeseer citations* This dataset consists of journal articles we collected from Citeseer and therefore formatted slightly differently from the Cora dataset. Also, unlike Cora it consists only of journal entries. The dataset is available at http://www.it.iitb.ac.in/∼sunita/data/personalBib.tar.gz.

| Task | Train domain | Target domain | Label | Train | | Target | |
|---|---|---|---|---|---|---|---|
| | | | | #train | #test | #train | #test |
| Cite_Cora | Citeseer citations | Cora citations | Author | 35 | 62 | 205 | 294 |
| Cora_Cite | Cora citations | Citeseer citations | Author | 155 | 294 | 39 | 62 |
| Title_Caps | Citeseer citations | All-Caps | Title | 35 | 62 | 39 | 62 |
| Author_Caps | Citeseer citations | All-Caps | Author | 35 | 62 | 39 | 62 |
| Cite_Conll | Citeseer citations | CoNLL | Person | 35 | 62 | 808 | 1191 |
| Conll_Cite | CoNLL | Citeseer citations | Person | 304 | 1191 | 39 | 62 |
| Hdr_Cite | Cora headers | Citeseer citations | Title | 45 | 87 | 39 | 62 |

**Table 1.** Description of domain adaptation tasks used in our experiments

In Table 1 we provide details of seven domain adaptation tasks created using various combination of these four datasets as the train and target domains and the extracted label. In tasks Title_Caps and Author_Caps the target domain differs from the train domain only in one respect: all words are fully capitalized in the target domain whereas in the train domain they are normal text records with a mix and capital and small letters. The last four columns specify for each of the two domains, the number of records used during training and testing respectively. For the target domain, the training documents are unlabeled.

We used a sequential CRF [1,2] with L2 regularization as our baseline model for information extraction. The package that we used is downloadable from [15]. We used the BCEU encoding of the entities where an entity like person name is decomposed into four labels: Begin-person, Continue-person, End-person, and Unique-person. Each token contributed two types of features: (1) the token itself if it was encountered in the training set and, (2) the set of regular expressions like digit or not, capitalized or not that the token matches. For each label $i$, these features where fired for the $i$th word and two words to the left and right of the word.

We evaluated our methods using F1 accuracy[2] at the level of individual tokens. We do not report span-level accuracy because the lack of standardization in what defines the boundaries of an entity, makes it difficult to get useful cross-domain comparison at the span-level. For example, in Citeseer the last punctuation (".") is outside the title entity whereas in Cora it is inside. In each experiment performance was averaged over four runs obtained by varying the subset of instances used for training and testing. Unless otherwise stated, our default method of domain adaptation uses $\gamma = 1$, $\lambda = 1$ and the square log-odd distance function $(\log E_D^k - \log E_{D'}^k)^2$. This distance function has been shown to work well [14] for sparse indicator features commonly found in information extraction tasks. We used the $\epsilon$-approximation trick proposed in [16] for handling the discontinuity of the objective when $\gamma = 1$.

### 5.1 Overall improvement with domain adaptation

In Table 2 we show the accuracy of the original unadapted model and the adapted model trained using our method respectively called "Original" and "Adapted". Along with the accuracy on the target domain, for comparison we also show accuracy on the train domain. In all cases, we find that the accuracy of the

| Dataset-Name | Train domain | | Target domain | |
|---|---|---|---|---|
| | Original | Adapted | Original | Adapted |
| Cite_Cora | 97.4 | 95.9 | 30.7 | 62.7 |
| Cora_Cite | 98.2 | 97.6 | 26.0 | 68.6 |
| Title_Caps | 94.4 | 93.2 | 41.8 | 90.1 |
| Author_Caps | 97.4 | 94.3 | 85.8 | 94.0 |
| Cite_Conll | 97.4 | 95.8 | 40.1 | 45.0 |
| Conll_Cite | 90.5 | 85.8 | 40.9 | 41.9 |
| Hdr_Cite | 85.3 | 76.0 | 12.0 | 27.8 |

**Table 2.** F1 Accuracy before and after domain adaptation.

target domain improves with domain adaptation. In some cases, the accuracy improvement is very dramatic, for example increasing from 26% to 69% on the second task.

For Title_Caps and Author_Caps where the target domain is just a fully capitalized version of the train domain, we find that the unadapted model performs very poorly whereas with adaptation we get accuracy comparable to the accuracy on the train domain. This illustrates the importance of adaptation even in domains that differ only slightly from the training domain. The top few features of the original model whose weight reduces almost to zero in the adapted model are: `IsInitCapital`, `IsInitCapital.left-2`, `IsInitCapital.right+2`, `W_Extract`, `IsAllSmallCase`, `IsAllSmallCase.left-2`, `IsAllSmallCase.right+2`. Most

[2] F1 is defined as 2*precision*recall/(precision+recall.)

of these are case related features which have no importance in the target domain. In contrast, the top few features whose weight increases significantly are `Punctuation, Punctuation.left-1, Punctuation.right+1, W_ACM.right+2`. These features remain invariant in the two domains since they are related to punctuation or fully capitalized words.

Another interesting observation from these tables is that on the train domain while the accuracy does drop after adapting to a different domain, the drop is only slight. This shows that in most cases, the model has other redundant features that start playing a role when some subset of its features are penalized.

### 5.2 Comparison with other methods

In Table 3 we compare our default method of domain adaptation to a number of other alternatives.

We compare with the recently proposed structural correspondence learning (SCL) [9] (described in Section 2). We find that SCL also shows significant accuracy improvements beyond the original unadapted model but the gain is lower than our method in all except the last dataset. Since our method of feature deletion is orthogonal to the SCL approach of feature addition, we also report results with both methods combined in the "SCL+Our" column of Table 3. In most cases, the combined method is better than either of the two.

We also compare our method to semi-supervised learning (SSL) proposed in [17] which adds to the training objective an additional goal of minimizing entropy labels for the unlabeled documents. In column SSL of Table 3 we show the results for the weight settings for which we obtained highest accuracy. Quite predictably, SSL is not competitive as a method of domain adaptation. We show

| Task | Original | Adapted | SCL | SCL+Our | SSL | $\mathbf{x}$-dist | $\gamma = 2$ | Square-dist |
|---|---|---|---|---|---|---|---|---|
| Cite_Cora | 30.7 | 62.7 | 47.3 | 63.3 | 31.5 | 27.6 | 63.6 | 32.8 |
| Cora_Cite | 26.0 | 68.6 | 68.6 | 67.8 | 26.0 | 76.2 | 75.9 | 46.0 |
| Title_Caps | 41.8 | 90.1 | 80.1 | 90.9 | 46.8 | 90.3 | 77.3 | 46.0 |
| Author_Caps | 85.8 | 94.0 | 87.1 | 94.7 | 86.4 | 94.3 | 94.2 | 86.4 |
| Cite_Conll | 40.1 | 45.0 | 52.1 | 45.1 | 40.4 | 40.9 | 45.7 | 32.2 |
| Conll_Cite | 40.9 | 41.9 | 43.9 | 41.1 | 43.0 | 36.8 | 43.6 | 44.1 |
| Hdr_Cite | 12.0 | 27.8 | 57.9 | 38.9 | 19.7 | 24.3 | 23.7 | 18.5 |

**Table 3.** Comparison of our method of domain adaptation with alternatives

the importance of comparing the mean of features in the joint $(\mathbf{x}, \mathbf{y})$ space instead of means along the projected $\mathbf{x}$ space as proposed in [13]. The latter is simpler to optimize because the distance function is independent of $\mathbf{w}$ and we get a simple convex objective. The results shown in column $\mathbf{x}$-dist of Table 3 indicate that in almost all cases the performance of the $\mathbf{x}$-only distance function is significantly worse than our method.

We vary our choice of $\gamma$ from 1 to 2, that is using weighted L2 regularizer instead of L1 in column $\gamma = 2$ of Table 3. We find that our default of L1 distance performs much better than L2. This observation agrees with earlier reports on the efficacy of feature selection using L1 instead of L2 regularizers. Next, we vary our default choice of the distance function. We chose log-odds ratio because it has been found to perform better on sparse Bernoulli features. Instead, if we use a regular square distance between the expected values of features, we find that the accuracy is much worse as shown in the column marked Square-dist.

### 5.3 Effect of training data

Another interesting aspect of domain adaptation is the performance of the adapted model with increasing training data. In Figure 1 we show the accuracy of the adapted model on the target domain and the unadapted model on the train domain with increasing labeled training data. The $y$ axis is the change in error compared to the error with 10% training data. As expected with statistical learners, with increasing training data, the error within the domain decreases. In contrast, the error of the adapted model either stays almost the same or increases slightly with more out-of-domain training data.

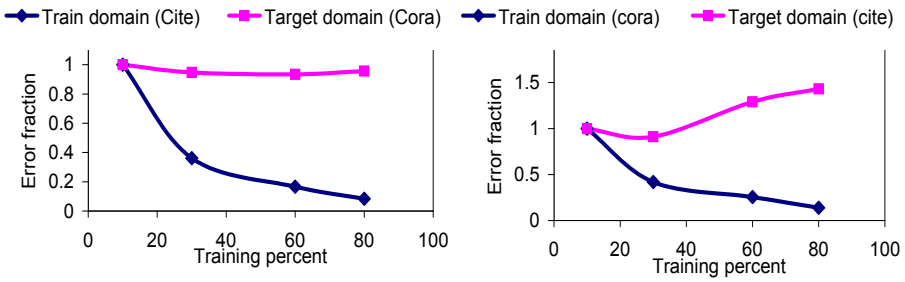


**Fig. 1.** Effect of increasing labeled training data on train and target domains for tasks Cite_Cora (left) and Cora_Cite (right)

## 6 Conclusion

In this paper we proposed a new method of unsupervised domain adaptation that selects a subset of features for which the distance between the train and target distribution is minimized while maximizing likelihood of the labeled data. The main challenge in this task is estimating distribution distance in the $(\mathbf{x}, \mathbf{y})$ space in which the model features are defined given only unlabeled samples from the target domain. We defined a distance measure and a method for solving the combined optimization problem that is both efficient and leads to significant accuracy improvements. In future, we would like to develop a theoretical analysis of this algorithm.

*Acknowledgments* The work reported here was supported by grants from Microsoft Research and an IBM Faculty award.

## References

1. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the International Conference on Machine Learning (ICML-2001), Williams, MA (2001)
2. Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: In Proceedings of HLT-NAACL. (2003)
3. Peng, F., McCallum, A.: Accurate information extraction from research papers using conditional random fields. In: HLT-NAACL. (2004) 329–336
4. Li, X., Bilmes, J.: A Bayesian Divergence Prior for Classifier Adaptation. Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS-2007) (2007)
5. Daumé III, H.: Frustratingly easy domain adaptation. In: Conference of the Association for Computational Linguistics (ACL), Prague, Czech Republic (2007)
6. Ando, R., Zhang, T.: A framework for learning predictive structures from multiple tasks and unlabeled data. Journal of Machine Learning Research **6** (2005) 1817–1853
7. Chelba, Acero: Adaptation of maximum entropy capitalizer: Little data can help a lot. In: EMNLP. (2004)
8. Jiang, J., Zhai, C.: Exploiting domain structure for named entity recognition. In: HLT-NAACL. (2006) 74–81
9. Blitzer, J., McDonald, R., Pereira, F.: Domain Adaptation with Structural Correspondence Learning. In: Proceedings of the Empirical Methods in Natural Language Processing (EMNLP). (2006)
10. Ben-David, S., Blitzer, J., Crammer, K., Pereira, F.: Analysis of representations for domain adaptation. In: Advances in Neural Information Processing Systems 20, Cambridge, MA, MIT Press (2007)
11. Globerson, A., Rowels, S.: Nightmare at test time: robust learning by feature deletion. In: ICML. (2006) 353–360
12. Zadrozny, B.: Learning and evaluating classifiers under sample selection bias. In: ACM International Conference Proceeding Series, ACM Press New York, NY, USA (2004)
13. Huang, J., Smola, A., Gretton, A., Borgwardt, K., Schölkopf, B.: Correcting Sample Selection Bias by Unlabeled Data. In: Advances in Neural Information Processing Systems 20, Cambridge, MA, MIT Press (2007)
14. Mladenic, D., Grobelnik, M.: Feature selection for unbalanced class distribution and naive bayes. In: ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning. (1999) 258–267
15. Sarawagi, S.: The crf project: a java implementation. `http://crf.sourceforge.net` (2004)
16. Lee, S.I., Lee, H., Abbeel, P., Ng, A.Y.: Efficient l1 regularized logistic regression. In: AAAI. (2006)
17. Jiao, F., Wang, S., Lee, C.H., Greiner, R., Schuurmans, D.: Semi-supervised conditional random fields for improved sequence segmentation and labeling. In: ACL. (2006)