Journal of Machine Learning Research 16 (2015) 3849-3875 Submitted 9/14; Published 12/15

Marginalizing Stacked Linear Denoising Autoencoders

Minmin Chen M.CHEN@CRITEO.COM
Criteo
Palo Alto, CA 94301, USA

Kilian Q. Weinberger KILIAN@WUSTL.EDU
Zhixiang (Eddie) Xu XUZX@CSE.WUSTL.EDU
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130, USA

Fei Sha FEISHA@USC.EDU
Computer Science Department
University of Southern California
Los Angeles, CA 90089, USA

Editor: Leon Bottou

Abstract

Stacked denoising autoencoders (SDAs) have been successfully used to learn new representations for domain adaptation. They have attained record accuracy on standard benchmark tasks of sentiment analysis across different text domains. SDAs learn robust data representations by reconstruction, recovering original features from data that are artificially corrupted with noise. In this paper, we propose *marginalized Stacked Linear Denoising Autoencoder* (mSLDA) that addresses two crucial limitations of SDAs: high computational cost and lack of scalability to high-dimensional features. In contrast to SDAs, our approach of mSLDA marginalizes noise and thus does not require stochastic gradient descent or other optimization algorithms to learn parameters — in fact, the linear formulation gives rise to a closed-form solution. Consequently, mSLDA, which can be implemented in only 20 lines of MATLAB$^{\text{TM}}$, is about *two orders of magnitude* faster than a corresponding SDA. Furthermore, the representations learnt by mSLDA are as effective as the traditional SDAs, attaining almost identical accuracies in benchmark tasks.

Keywords: domain adaption, fast representation learning, noise marginalization, denoising autoencoders

1. Introduction

The goal of domain adaptation (Ben-David et al., 2010; Huang et al., 2006; Weinberger et al., 2009; Xue et al., 2008) is to generalize a classifier that is trained on a source domain, for which typically plenty of training data is available, to a target domain, for which data is scarce. Cross-domain generalization is important in many application areas of machine learning, where such an imbalance of training data may occur. Examples include computational biology (Liu et al., 2008), natural language processing (Daume III, 2007; McClosky et al., 2006), computer vision (Saenko et al., 2010) and web-search ranking (Chapelle et al., 2011).

Adaptation is challenging, because the data in the two domains is not identically distributed and a classifier trained on source can be expected to perform significantly worse on the target domain. Recent work has investigated several techniques to reduce this adaptation error:

- *instance re-weighting* (Huang et al., 2006; Mansour et al., 2009) is an approach to re-weight source inputs so that the distribution of the reweighed source data matches that of the target domain; instance weighting strategies assume that the source and target distribution share the same support and features. It tends to be less effective for tasks of high-dimensional, sparse features such as text documents and where source and target distributions differ more drastically.

- *joint feature mapping* (Blitzer et al., 2006; Gong et al., 2012; Xue et al., 2008; Glorot et al., 2011) is an approach to learn a new shared representation for the source and target domains, in which the two data distributions align. These algorithms are designed for highly divergent domains, which can contain different features, and are more closely related to our work.

- *parameter sharing* (Daume III, 2007; Chapelle et al., 2011; Weinberger et al., 2009) is an approach to adapt machine learning classifiers to incorporate shared weights across the two domains. This is arguably the most popular category of domain adaptation algorithms amongst practitioners, mostly due to their appealing simplicity (Daume III, 2007).

One of the most successful domain adaptation algorithms was introduced by Glorot et al. (2011), which falls into the second category. The authors use stacked denoising autoencoders (SDA) (Vincent et al., 2008) to learn a joint feature representation that can be shared across multiple domains. Denoising autoencoders are one-layer neural networks that are optimized to reconstruct input data from partial and random corruption. These denoisers can be stacked into deep learning architectures, which are then fine-tuned with back-propagation (Vincent et al., 2008). Glorot et al. (2011) use the internal representation of the intermediate layers of the SDA as input features for linear classifiers, an idea pioneered by Lee et al. (2009) and Vincent et al. (2010). The authors demonstrate in their work that such SDA-learned features are very effective for cross-domain generalization, even with straight-forward linear Support Vector Machines (SVM) (Cortes and Vapnik, 1995). For example, it yields record adaptation accuracies on the Amazon$^{\text{TM}}$ sentiment-analysis benchmark tasks of predicting review sentiment across product domains (Blitzer et al., 2006).

Although the capabilities of SDAs are remarkable, they are limited by their high computational cost. Compared with competing approaches (Blitzer et al., 2006; Xue et al., 2008; Chen et al., 2011b), SDAs are significantly slower to train. This is primarily the case because of the large number of model parameters in the denoising autoencoders, which are learned with iterative algorithms for numerical optimization. The challenge is further compounded by the dimensionality of the input data and the need for computationally intensive model selection procedures to tune hyperparameters. Consequently, even a highly optimized implementation (Bergstra et al., 2010) may require hours (even days) of training time on the larger Amazon$^{\text{TM}}$ benchmark data sets.

In this paper, we introduce a variation of SDAs that addresses these shortcomings. The proposed method, which we refer to as *marginalized Stacked Linear Denoising Autoencoder* (mSLDA), adopts the greedy layer-by-layer training of SDAs. Similarly, at each layer we learn a denoiser to recover input data from random corruption. However, a crucial difference is that we use *linear* denoisers as the basic building blocks. This restriction has two important advantages: 1. the random feature corruption can be marginalized out, which alleviates the need to iterate over many corrupted versions of the data; 2. the weights of the linear denoisers can be computed in closed form, in very little time (almost instantaneous). Conceptually, marginalizing the corruption is equivalent to training the model over an infinite number of corrupted versions of the input data.

Although the restriction to only linear denoisers makes mSLDA less expressive than SDA, we observe that for high dimensional data sets they are sufficient and mSLDA features match the original SDA features in quality. This is particularly impressive, as the training of the mSLDA features is several orders of magnitude faster (reducing training from up to 2 days for SDA to a few minutes with mSLDA).

Two earlier short paper on this work (Chen et al., 2012; Xu et al., 2012), already introduce this learning framework, but this longer version provides a significant amount of additional details. In particular, we provide extensions to different corruption models, further and deeper analysis of the mSLDA algorithm, additional experiments with different datasets (text documents and images), and new experimental results in semi-supervised settings. The remaining parts of the paper is organized as follows. In Section 2 we lay out the problem and review a couple of closely-related prior works. In Section 3 we introduce the mSLDA framework for learning representations. In Section 4 we discuss several input corruption models, which fit naturally into the mSLDA framework. In Section 5 we propose an extension to scale up our learning framework to inputs of high dimensions. In Section 7 we present an extensive set of results evaluating mSLDA on several text classification and object recognition tasks. In Section 8 we provide further analysis of the results and discuss strengths and limitations of mSLDA.

2. Background and Related Work

We assume that during training we are provided with labeled data from the source domain $L = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathcal{R}^d$ with corresponding labels $y_1, \ldots, y_m \subset \mathcal{Y}$. Here, $\mathcal{Y}$ can consist of real valued or categorical labels. We focus on the simple binary case with $\mathcal{Y} = \{+1, -1\}$ throughout this manuscript, however we would like to emphasize that our proposed feature learning algorithm is *unsupervised* and therefore agnostic to the label choice (which only affects the classifier trained on the learned features). If labeled *target* data is available, it can be included into L, although in our setting we do not assume this is the case. We are potentially also provided with *unlabeled* data $U = \{\mathbf{x}_{m+1} \ldots, \mathbf{x}_{m+u}\} \subset \mathcal{R}^d$, which may be sampled from source, target or other (related) source distributions. For notational simplicity we define $n = m + u$. Although we do assume that any two domains have some overlap in features, we *do not* assume that they have *identical* features. Instead, we pad all input vectors with zeros to make them of matching dimensionality d. Given this mix of labeled and unlabeled source and target data, our goal is to train a classifier that accurately predicts the labels of instances from the target domain T.

In the following, we briefly review work that is most similar to ours, including Structural Correspondence Learning (SCL) (Blitzer et al., 2006), Stacked Denoising Autoencoders (Glorot et al., 2011) and learning with marginalized corruption (van der Maaten et al., 2013).

2.1 Structural Correspondence Learning

The leaning of joint source / target representations explicitly for domain adaptation was pioneered by Blitzer et al. (2006) and their Structural Correspondence Learning (SCL) algorithm. SCL assumes a known set of pivot features, which appear frequently in both domains (source and target) and behave similarly. These are used to put domain specific words in correspondence. The low-rank representation learned with SCL essentially encodes the covariance between non-pivot features and the pivot features. As described in detail in Section 3, the single-layer mSLDA also learns the correlations between *all* the features. In this sense, the resulting feature space is similar to SCL, and the computation time of SCL and mSLDA are comparable. However, mSLDA introduces reconstruction from corruption and stacking of multiple denoising layers, which result in superior feature quality. Further, mSLDA does not require any side information about a pivot features set, which can be hard to identify (Blitzer et al., 2006).

2.2 Marginalized Corrupted Features

Recently, van der Maaten et al. (2013) proposed the Marginalized Corrupted Features (MCF) learning framework, which was inspired by our earlier publication of mSLDA (Chen et al., 2012).[1] MCF uses marginalized corruption to improve the generalization performance of linear classifiers, as an alternative to L_2 or L_1 norm regularization. MCF is equivalent to first generating infinitely many corrupted copies of the training data, with a pre-defined corruption distribution, and then training an unregularized classifier on this (infinite) data set. Training on additional corrupted inputs leads to substantially more robust classifiers, as has previously been shown by Burges and Schölkopf (1997). MCF borrows the idea from mSLDA to marginalize out this corruption, which leads to substantial improvements in speed and accuracy over explicitly corrupting only finitely many copies of the training data. In a similar spirit, Wang and Manning (2013) introduce marginalized dropout (Hinton et al., 2012) for logistic regression and show that the marginalized corruption can be interpreted as active regularization.

2.3 Stacked Denoising Autoencoder

Our work is mostly inspired by Autoencoders. Various forms of autoencoders have been developed in the machine learning community (Rumelhart et al., 1986; Baldi and Hornik, 1989; Kavukcuoglu et al., 2009; Lee et al., 2009; Vincent et al., 2008; Rifai et al., 2011). In its simplest form, an autoencoder has two components, an encoder $h(\cdot)$ maps an input $\mathbf{x} \in \mathcal{R}^d$ to some hidden representation $h(\mathbf{x}) \in \mathcal{R}^{d_h}$, and a decoder $g(\cdot)$ maps this hidden representation back to a reconstructed version of $\mathbf{x}$, such that $g(h(\mathbf{x})) \approx \mathbf{x}$. The parameters of the autoencoders are learned to minimize the reconstruction error, measured by some loss $\ell(\mathbf{x}, g(h(\mathbf{x})))$. Choices for the loss include squared error or Kullback-Leibler divergence (when the feature values are in $[0, 1]$.)

Denoising Autoencoders (DAs) incorporate a slight modification to this setup and corrupt the inputs before mapping them into the hidden representation. They are trained to reconstruct (or *denoise*) the original input $\mathbf{x}$ from its corrupted version $\tilde{\mathbf{x}}$ by minimizing $\ell(\mathbf{x}, g(h(\tilde{\mathbf{x}})))$. Typical choices of corruption include additive isotropic Gaussian noise or binary masking noise. As in Vincent et al. (2008), we primarily use the latter and set a fraction of the features of each input to *zero*. This is a natural choice for bag-of-word representations of text documents, where author specific word preferences can influence the existence or absence of words in the source and target domains.

The stacked denoising autoencoder (SDA) of Vincent et al. (2008) stacks several DAs together to create higher-level representations, by feeding the hidden representation of the t^{th} DA as input into the $(t+1)^{th}$ DA. The training is performed greedily, layer by layer.

[1] In this earlier work we refer to mSLDA as simply marginalized Stacked Denoising Autoencoder (mSDA). Since then we added the term “Linear” to avoid confusion.

Feature Generation. Recently, Lee et al. (2009) and Glorot et al. (2011) have identified autoencoders as a powerful tool for automatic discovery and extraction of nonlinear features. For example, Lee et al. (2009) demonstrate that the hidden representations computed by all or partial layers of a convolutional deep belief network (CDBN) make excellent features for classification with SVMs. The pre-processing with a CDBN improves the generalization by increasing robustness against noise and label-invariant transformations.

Glorot et al. (2011) successfully apply SDAs to extract features for domain adaptation in document sentiment analysis. The authors train an SDA to reconstruct the unlabeled input vectors on the union of the source and target data. A classifier (*e.g.* a linear SVM) trained on the resulting feature representation $h(\mathbf{x})$ transfers significantly better from source to target than one trained on $\mathbf{x}$ directly. Similar to CDBNs, SDAs also combine correlated input dimensions, as they reconstruct removed feature values from the remaining uncorrupted ones. In fact, Glorot et al. (2011) show that SDAs are able to disentangle hidden factors, which explain the variations in the input data, and automatically group features in accordance with their relatedness to these factors. This helps transfer across domains as these generic concepts are invariant to domain-specific vocabularies.

As an intuitive example, imagine that we classify product reviews according to their sentiments. The source data consists of *book* reviews, the target of *kitchen appliances*. A classifier trained on the original bag-of-words source never encounters the bigram *energy efficient* during training and therefore assigns zero weight to it. In the learned SDA representation, the bigram *energy efficient* would tend to reconstruct, and be reconstructed by, co-occurring features, typically of similar sentiment (*e.g. good* or *love*). The SDA will preform the same reconstruction also on the source data, in other words, it will "reconstruct" bigrams like *energy efficient* in book reviews that contain words with positive sentiment. Thus, the source-trained classifier can assign weights even to features that never occur in its original domain representation.

Although SDAs generate excellent features for domain adaptation, they have several drawbacks: 1) Training with (stochastic) gradient descent is slow and hard to parallelize, and SDAs take relatively long to train—even with efficient GPU implementations (Bergstra et al., 2010) and reconstruction sampling for sparse data (Dauphin et al., 2011); 2) There are several hyper-parameters (learning rate, number of epochs, noise ratio, mini-batch size and network structure), which need to be set by cross validation—this is particularly expensive as each individual run can take several hours; 3) The optimization is inherently non-convex and dependent on its initialization.

3. Marginalized Stacked Linear Denoising Autoencoders

In this section we introduce a modified version of SDA, which we refer to as *marginalized* Stacked Linear Denoising Autoencoder (mSLDA). In practice if a SDA is trained to learn features (rather than predict a target label directly), linear autoencoders are typically sufficient. Our proposed algorithm consists of stacked linear denoising autoencoders where the corruption is marginalized out in closed form — effectively yielding orders of magnitude speedups during training time. In addition, mSLDA has fewer hyper-parameters, allowing for much faster model-selection, and is layer-wise convex.

3.1 Noise Model

Similar to SDA, mSLDA learns to reconstruct the original input from its corrupted version. Therefore, we start by defining a corrupting distribution that specifies how training observations $\mathbf{x}$ are transformed into corrupted versions $\tilde{\mathbf{x}}$. Throughout the paper, we assume a corrupting distribution of the form:

$$p(\tilde{\mathbf{x}}|\mathbf{x}) = \prod_{\alpha=1}^{d} p_E(\tilde{x}_\alpha|x_\alpha;\eta_\alpha). \tag{1}$$

where η_d is the list of user-defined hyper-parameters for the corrupting distribution. That is, we assume that 1) each dimension of the input $\mathbf{x}$ is corrupted independently; 2) the individual corrupting distributions have well-defined (finite) mean and variance, such as the Bernoulli, Poisson and Gaussian distribution. As we are going to explain later, these two assumptions leads to very efficient optimizations of our models.

For now, we are going to focus on the blank-out noise model (also often referred to as "mask-out"), which randomly sets each feature to *zero* with probability $p_\alpha \geq 0$. More precisely (with $\eta_\alpha = p_\alpha$),

$$p_E(\tilde{x}_\alpha|x_\alpha;\eta_\alpha) = \begin{cases} 0 & \text{with probability} \quad p_\alpha \\ x_\alpha & \text{with probability} \quad 1-p_\alpha \end{cases}. \tag{2}$$

Although our model is more general, for simplification we will assume that the corruption probability is identical for all features, *i.e.* $p_\alpha = p$ for all dimensions α. In Section 4, we will extend this model to different corrupting distributions.

3.2 Single-layer Denoiser

The basic building block of mSLDA is a one-layer linear denoising autoencoder. We take the unlabeled inputs $\mathbf{x}_1,\ldots,\mathbf{x}_n$ from $L \cup U$ and corrupt them with the blank-out noise, which sets each feature to 0 with probability $p \geq 0$. Let us denote the corrupted version of $\mathbf{x}_i$ as $\tilde{\mathbf{x}}_i$. As opposed to the two-level *encoder* and *decoder* in SDA, we reconstruct the corrupted inputs with a single linear mapping $\mathbf{W}: \mathcal{R}^d \rightarrow \mathcal{R}^d$, that minimizes the squared reconstruction loss

$$\frac{1}{2n}\sum_{i=1}^{n}\|\mathbf{x}_i - \mathbf{W}\tilde{\mathbf{x}}_i\|^2. \tag{3}$$

To simplify notation, we assume that a constant feature is added to the input, $\mathbf{x}_i = [\mathbf{x}_i; 1]$, and an appropriate bias is incorporated within the mapping $\mathbf{W} = [\mathbf{W}, \mathbf{b}]$. The constant feature is *never* corrupted.

The solution to (3) depends on which features of each input are randomly corrupted. To lower the variance, we perform t passes of corruption and reconstruction over the training set, each time with new randomly chosen corruptions for each input. We solve for the matrix $\mathbf{W}$ that minimizes the overall squared loss

$$\mathcal{L}_{sq}^{t}(\mathbf{W}) = \frac{1}{2nt}\sum_{i=1}^{n}\sum_{j=1}^{t}\|\mathbf{x}_i - \mathbf{W}\tilde{\mathbf{x}}_{i,j}\|^2, \tag{4}$$

where $\tilde{\mathbf{x}}_{i,j}$ represents the j^{th} corrupted version of the original input $\mathbf{x}_i$.

Let us define the design matrix $\mathbf{X} = [\mathbf{x}_1,\ldots,\mathbf{x}_n] \in \mathcal{R}^{d\times n}$ and its t-times repeated version as $\overline{\mathbf{X}} = [\mathbf{X},\ldots,\mathbf{X}]$. Further, we denote the corrupted version of $\overline{\mathbf{X}}$ as $\tilde{\mathbf{X}}$. With this notation, the loss in eq. (4) can be expressed in matrix form as

$$\mathcal{L}_{sq}^{t}(\mathbf{W}) = \frac{1}{2nt}\text{tr}\left[\left(\overline{\mathbf{X}} - \mathbf{W}\widetilde{\mathbf{X}}\right)^\top\left(\overline{\mathbf{X}} - \mathbf{W}\widetilde{\mathbf{X}}\right)\right]. \tag{5}$$

Algorithm 1 mLDA (for blankout corruption) in MATLAB$^{\text{TM}}$.

```
function [W,h]=mLDA(X,p);
 X=[X;ones(1,size(X,2))];
 d=size(X,1);
 q=[ones(d-1,1).*(1-p); 1];
 S=X*X';
 Q=S.*(q*q');
 Q(1:d+1:end)=q.*diag(S);
 P=S.*repmat(q',d,1);
 W=P(1:end-1,:)/(Q+1e-5*eye(d));
 h=tanh(W*X);
```

Similar to ordinary least squares (Bishop, 2006), it is straight-forward to derive a closed-form solution to (5):

$$\mathbf{W} = \mathbf{P}\mathbf{Q}^{-1} \text{ with } \mathbf{Q} = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^\top \text{ and } \mathbf{P} = \overline{\mathbf{X}}\widetilde{\mathbf{X}}^\top. \tag{6}$$

In practice (6) can be computed as a system of linear equations, without the costly matrix inversion. (The worst-case complexity is still $O(n^3)$, but the average runtime is much accelerated.)

3.3 Marginalized Linear Denoising Autoencoder

The larger t is, the more corruptions we average over. Ideally we would like $t \to \infty$, effectively using infinitely many copies of noisy data to compute the denoising transformation $\mathbf{W}$. In this scenario, as $t \to \infty$, the loss $\mathcal{L}_{sq}$ in (4) becomes the expected reconstruction loss under $p(\tilde{\mathbf{x}}_i|\mathbf{x})$

$$\mathcal{L}_{sq}^{\infty}(\mathbf{W}) = \frac{1}{2n}\sum_{i=1}^{n} \mathbb{E}_{p(\tilde{\mathbf{x}}_i|\mathbf{x})}\left[\|\mathbf{x}_i - \mathbf{W}\tilde{\mathbf{x}}_i\|^2\right]. \tag{7}$$

We can expand this equation to obtain

$$\mathcal{L}_{sq}^{\infty}(\mathbf{W}) = \frac{1}{2n}\sum_{i=1}^{n}\left(\mathbf{x}_i\mathbf{x}_i^\top - 2\mathbf{x}_i\mathbb{E}[\tilde{\mathbf{x}}_i]^\top\mathbf{W}^\top + \mathbf{W}\mathbb{E}[\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_i^\top]\mathbf{W}^\top\right), \tag{8}$$

and, by solving for $\mathbf{W}$, the solution to (7) can then be expressed as

$$\mathbf{W} = \mathbb{E}[\mathbf{P}]\mathbb{E}[\mathbf{Q}]^{-1} \text{ with } \mathbb{E}[\mathbf{Q}] = \sum_{i=1}^{n}\mathbb{E}\left[\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_i^\top\right] \text{ and } \mathbb{E}[\mathbf{P}] = \sum_{i=1}^{n}\mathbf{x}_i\mathbb{E}[\tilde{\mathbf{x}}_i]^\top. \tag{9}$$

We refer to this algorithm as marginalized Linear Denoising Autoencoder (mLDA).

3.3.1 BLANKOUT CORRUPTION

As an example, let us consider the blankout corruption,

$$p_E(\tilde{x}_\alpha|x_\alpha;\eta_\alpha) = \begin{cases} 0 & \text{with probability} \quad p_\alpha \\ x_\alpha & \text{with probability} \quad 1-p_\alpha \end{cases}. \tag{10}$$

For notational convenience, we define a vector $\mathbf{q} = [1-p, \ldots, 1-p, 1]^\top \in \mathcal{R}^{d+1}$, where $\mathbf{q}_\alpha$ represents the probability of a feature α "surviving" the corruption. (As the constant feature is never corrupted, we have $\mathbf{q}_{d+1} = 1$.) According to the blank-out noise model defined in (10), the expected value of the corruption $\mathbb{E}[\tilde{\mathbf{x}}_i]$ can be computed as $\mathbf{x}_i \cdot \mathbf{q}$.[2] We further define the scatter matrix of the original uncorrupted input as $\mathbf{S} = \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top$, and express the expectation $\mathbb{E}[\mathbf{P}]$ as

$$\mathbb{E}[\mathbf{P}] = \sum_{i=1}^{n} \mathbf{x}_i (\mathbf{x}_i \cdot \mathbf{q})^\top \text{ with } \mathbb{E}[\mathbf{P}]_{\alpha\beta} = \mathbf{S}_{\alpha\beta} \mathbf{q}_\beta. \tag{11}$$

Similarly, we can compute the expectation

$$\mathbb{E}[\mathbf{Q}] = \sum_{i=1}^{n} \mathbb{E}\left[\tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^\top\right].$$

An off-diagonal entry in the matrix $\tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^\top$ with index (α, β) is uncorrupted if the two features α and β both "survived" the corruption. This happens with probability $(1-p)^2$. For the diagonal entries, this holds with probability $1-p$ (because it only requires the one corresponding feature to "survived" the corruption). Thus, we can express the expectation of the matrix $\mathbf{Q}$ as

$$\mathbb{E}[\mathbf{Q}]_{\alpha,\beta} = \begin{cases} \mathbf{S}_{\alpha\beta} \mathbf{q}_\alpha \mathbf{q}_\beta & \text{if} \quad \alpha \neq \beta \\ \mathbf{S}_{\alpha\beta} \mathbf{q}_\alpha & \text{if} \quad \alpha = \beta \end{cases}. \tag{12}$$

With the help of these matrix expectations, we can compute the reconstructive mapping $\mathbf{W}$ directly in closed-form without ever explicitly constructing a single corrupted input $\tilde{\mathbf{x}}_i$. Algorithm 1 shows a 10-line MATLAB$^{\text{TM}}$ implementation of mLDA with blankout corruption. The mLDA has several advantages over traditional denoisers: 1) It requires only a single sweep through the data to compute the matrices $E[\mathbf{Q}], E[\mathbf{P}]$; 2) Training is convex and a globally optimal solution is guaranteed; 3) The optimization is performed in non-iterative closed-form.

3.4 Nonlinear feature generation and stacking

Arguably two of the key contributors to the success of the SDA are its *nonlinearity* and the *stacking* of multiple layers of denoising autoencoders to create a "deep" learning architecture. Our framework has the same capabilities.

In SDAs, the nonlinearity is injected through the nonlinear *encoder* function $h(\cdot)$, which is learned together with the reconstruction weights $\mathbf{W}$. Such an approach makes the training procedure highly non-convex and requires iterative procedures to learn the model parameters. To preserve the closed-form solution from the linear mapping in equation (5) we insert nonlinearity into our learned representation *after* the weights $\mathbf{W}$ are computed. A nonlinear squashing-function is applied on the output of each mLDA. Several choices are possible, including sigmoid, hyperbolic tangent, or the rectifier function (Nair and Hinton, 2010). Throughout this work, we use the hyperbolic tangent $\tanh()$ function and provide a detailed comparison of various squashing function in Figure 4 in Section 7.1.2.

Inspired by the layer-wise stacking of SDA, we stack several mLDA layers by feeding the output of the $(t–1)^{th}$ mLDA (after the squashing function) as the input into the t^{th} mLDA. Let us denote the

[2] Here, $\mathbf{y} = \mathbf{x} \cdot \mathbf{z}$ denotes element-wise vector multiplication, *i.e.* $y_i = x_i z_i$.

Algorithm 2 mSLDA in MATLAB$^{\text{TM}}$.

```
function [Ws,hs]=mSLDA(X,p,L);
 [d,n]=size(X);
 Ws=zeros(d,d+1,L);
 hs=zeros(d,n,L+1);
 hs(:,:,1)=X;
 for t=1:L
  [Ws(:,:,t), hs(:,:,t+1)]=mLDA(hs(:,:,t),p);
 end;
```

output of the t^{th} mLDA as $\mathbf{h}^t$ and the original input as $\mathbf{h}^0 = \mathbf{x}$. The training is performed greedily layer by layer: each map $\mathbf{W}^t$ is learned (in closed-form) to reconstruct the previous mLDA output $\mathbf{h}^{t-1}$ from all possible corruptions and the output of the t^{th} layer becomes $\mathbf{h}^t = \tanh(\mathbf{W}^t\mathbf{h}^{t-1})$. In our experiments, as detailed in in Section 7.1.2, we found that even without the nonlinear squashing function, stacking still improves the performance. However, the nonlinearity improves over the linear stacking significantly. We refer to the stacked denoising algorithm as marginalized Stacked Linear Denoising Autoencoders (mSLDA). Algorithm 2 shows a 8-lines MATLAB$^{\text{TM}}$ implementation of mSLDA.

3.5 mSLDA for Domain Adaptation

We apply mSLDA to domain adaptation by first learning features in an unsupervised fashion on the union of the source and target data sets. One observation reported in (Glorot et al., 2011) is that if multiple domains are available, sharing the unsupervised pre-training of SDA across all domains is beneficial compared to pre-training on the source and target only. We observe a similar trend with our approach. The results reported in Section 7 are based on features learned on data from all available domains. Once a mSLDA is trained, the output of all layers, after squashing $(\tanh(\mathbf{W}^t\mathbf{h}^{t-1}))$ combined with the original features $\mathbf{h}^0$, are concatenated and form the new representation. All inputs are transformed into the new feature space. A linear Support Vector Machine (SVM) (Chang and Lin, 2011) is then trained on the transformed source inputs and tested on the target domain. There are two sets of meta-parameters in mSLDA: the corruption parameters (*e.g.* p in the case of blankout corruption) and the number of layers L. In our experiments, both are set with 5-fold cross validation on the labeled data from the *source* domain. As the mSLDA training is almost instantaneous, this grid search is almost entirely dominated by the SVM training time.

4. Corruption beyond blank-out

In the previous section, we introduced mSLDA under the blank-out corruption model and derived the layer-wise closed form solution $\mathbf{W} = \mathbb{E}[\mathbf{P}]\mathbb{E}[\mathbf{Q}]^{-1}$. The derivation up to eq. (9) makes no explicit assumption on the corruption distribution and holds for any member of the exponential family with finite mean $\mathbb{E}[\tilde{\mathbf{x}}_i]$, and variance $\mathbb{V}[\tilde{\mathbf{x}}_i]$. This can be made explicit by expanding the terms $\mathbb{E}[\mathbf{P}], \mathbb{E}[\mathbf{Q}]$ as

$$\mathbb{E}[\mathbf{P}] = \sum_{i=1}^{n} \mathbf{x}_i \mathbb{E}[\tilde{\mathbf{x}}_i]^\top \text{ and } \mathbb{E}[\mathbf{Q}] = \sum_{i=1}^{n} \mathbb{E}\left[\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_i^\top\right] = \sum_{i=1}^{n}\Big(\mathbb{E}[\tilde{\mathbf{x}}_i]\mathbb{E}[\tilde{\mathbf{x}}_i]^\top + \mathbb{V}[\tilde{\mathbf{x}}_i]\Big). \tag{13}$$

4.1 Poisson corruption

For discrete feature values (*e.g.* word counts in a document), one interesting example of a corruption distribution is the Poisson distribution. Here, the corruption is defined as,

$$p_E(\tilde{x}_\alpha|x_\alpha;\eta_\alpha) = \frac{x_\alpha^{\tilde{x}_\alpha} e^{-x_\alpha}}{\tilde{x}_\alpha!}, \alpha = 1, \cdots, d \tag{14}$$

where the arrival rate η_α is set to x_α. In this case, we have $\mathbb{E}[\mathbf{x}] = \mathbf{x}$, and $\mathbb{V}[\mathbf{x}] = \Delta(\mathbf{x})$.[3] Note that the off-diagonal entries of the variance matrix is zero since we assume that each dimension of the input is corrupted independently. Plugging the definition (14) into eq. (13) results in

$$\mathbb{E}_{Poi}[\mathbf{P}] = \sum_{i=1}^{n} \mathbf{x}_i\mathbf{x}_i^\top = \mathbf{S} \text{ and } \mathbb{E}_{Poi}[\mathbf{Q}] = \sum_{i=1}^{n} \mathbf{x}_i\mathbf{x}_i^\top + \sum_{i=1}^{n} \Delta(\mathbf{x}_i) = \mathbf{S} + \Delta(\sum_{i=1}^{n} \mathbf{x}_i).$$

Comparing with the blank-out noise, where the corruption simulates the existence or completely absence of words due to authors' word preference, the Poisson corruption imitates different appearing frequencies for each word. Since we set the arrival rate of the distribution to be x_α, words with higher frequency in the original input will have less chance to be complete removed. In other words, we would expect the Poisson corruption to bring in less drastic change to the corrupted input $\tilde{\mathbf{x}}$ than the blank-out noise. As we can see in the experiments, the representations learned with Poisson corruption is not as robust as those with blank-out noise for domain adaptation where we would expect some words in the source domain to be completely removed from the target domain and vice versa.

4.2 Feature dependent blank-out

In Section 3.2, we introduced mLDA under the blank-out corruption model with uniform corruption rate for individual dimensions of the input. The definition of the corruption models in (1), however, allows different features to have arbitrarily different corruption rate. This enables us to incorporate prior knowledge of the corrupting distribution into our model flexibly and randomly blank-out features of different dimensions at different rate. The derivation of the two expectations $\mathbb{E}[\mathbf{P}]$ and $\mathbb{E}[\mathbf{Q}]$ is the same as in equation (11) and (12), except that a different corrupting vector $\mathbf{q}$ will be used, where each entry $\mathbf{q}_\alpha$ can take a different value.

5. Extension to High Dimensional Data

Many data sets (*e.g.* bag-of-words text documents) are naturally high dimensional and sparse. As the dimensionality increases, hill-climbing approaches used in SDAs can become prohibitively expensive. In practice, a work-around is to truncate the input data to the $r \ll d$ most common features (Glorot et al., 2011). Unfortunately, this prevents SDAs from utilizing important information found in rarer features. (As we show in Section 7, including these rarer features leads to significantly better results.) High dimensionality also poses a challenge to mSLDA, as the system of linear equations in (9) of complexity $O(d^3)$ becomes too costly. In this section we describe how to approximate this calculation with a simple division into $\frac{d}{r}$ sub-problems of $O(r^3)$.

[3] Here, $\Delta(\mathbf{x})$ denotes a diagonal square matrix with $\mathbf{x}$ along its diagonal.

We combine the concept of "pivot features" from Blitzer et al. (2006) and the use of most-frequent features from Glorot et al. (2011). Instead of learning a single mapping $\mathbf{W} \in \mathcal{R}^{d\times(d+1)}$ to reconstruct all corrupted features, we learn *multiple mappings* but only reconstruct the $r \ll d$ most frequent features (here, $r = 5000$). For an input $\mathbf{x}_i$ we denote the shortened r-dimensional vector consisting of the r most-frequent features as $\mathbf{z}_i \in \mathcal{R}^r$. We divide the input features randomly into S mutually exclusive sub-sets of (roughly) equal size and learn a mapping from each one of these subsets to $\mathbf{z}_i$. Intuitively, this corresponds to "translating" rare features into common features (this is particularly successful with text documents, where the meaning of infrequent terms can often be approximated by a more frequent term.) Without loss of generality, we assume that the feature-dimensions in the input space are in random order and divide up the input vectors as $\mathbf{x}_i = \left[\mathbf{x}_i^{1\top}, \ldots, \mathbf{x}_i^{S\top}\right]^\top$. For each one of these sub-spaces we learn an independent mapping $\mathbf{W}^s$ which minimizes

$$\mathcal{L}_s(\mathbf{W}^s) = \frac{1}{2n}\sum_{i=1}^{n}\sum_{s=1}^{S}\|\mathbf{z}_i - \mathbf{W}^s\tilde{\mathbf{x}}_i^s\|^2. \tag{15}$$

Each mapping $\mathbf{W}^s$ can be solved in closed-form as in eq. (9), following the method described in section 1. We define the output of the first layer in the resulting mSLDA as the average of all reconstructions,

$$\mathbf{h}^1 = \tanh\left(\frac{1}{S}\sum_{s=1}^{S}\mathbf{W}^s\mathbf{x}^s\right). \tag{16}$$

Once the first layer of dimension $r \ll d$ is learned no further dimensionality reduction is required and we can stack subsequent layers using the regular mSLDA as described in Section 3.4 and Algorithm 2. It is worth pointing out that, although features might be separated in different sub-sets within the first layer, they can still be combined in subsequent layers of the mSLDA.

6. Alternative Formulation

In this section we want to provide the reader briefly with an alternative interpretation of mLDA with **unbiased** blank-out noise. Slightly different from the blank-out noise we introduced in Section 3.1, the unbiased version rescales the uncorrupted features to $\frac{1}{1-p}$ of their original values. More precisely (with $\eta_d = p$),

$$p_E(\tilde{x}_\alpha|x_\alpha;\eta_\alpha) = \begin{cases} 0 & \text{with probability} \quad p \\ \frac{1}{1-p}x_\alpha & \text{with probability} \quad 1-p \end{cases}. \tag{17}$$

Under this specific corruption model, and in the case where all the features are normalized to have norm 1 *across inputs*, *i.e.*, $\forall\alpha \in \{1,\cdots,d\}$, $\sum_{i=1}^{n} x_{i\alpha}^2 = 1$, we can then re-interpret mLDA reconstruction in eq. (7) as auto-Ridge Regression,

$$\min_{\mathbf{W}} \frac{1}{2n}\sum_{i=1}^{n}\|\mathbf{x}_i - \mathbf{W}\mathbf{x}_i\|^2 + \lambda\|\mathbf{W}\|_2^2. \tag{18}$$

with $\lambda = \frac{p}{2n(1-p)}$. In the extreme case of $p = 0$ and consequently $\lambda = 0$, the solution to (18) is trivially $\mathbf{W} = \mathbf{I}$. However, as λ increases, the $l2$ regularization encourages weights within $\mathbf{W}$ to be of comparable magnitude and reduces large diagonal entries. As p approaches 1 the regularization trade-off λ becomes ill-defined—corresponding to the pathological case where all the features are removed in all examples, making it impossible to learn. This alternative interpretation illustrates the effect of reconstruction from blank-out corruption. Features are reconstructed from themselves and other co-occurring features and the hyper-parameter p regulates this trade-off. The effect of applying the learned reconstruction matrix to the original input, i.e., $\mathbf{Wx}$, is a smoothing of related feature values to increase robustness of the representation. In the case of domain adaptation, this facilitates some immunity over distribution drift between training and testing.

7. Experimental Results

In this section, we evaluate mSLDA on two real-world domain adaption tasks, as well as a semi-supervised learning task, and compare it with competing algorithms.

7.1 Domain adaption on text data

First, we consider a domain adaptation task for sentiment analysis. We evaluate mSLDA on the *Amazon reviews* benchmark data sets (Blitzer et al., 2006) together with several other algorithms for representation learning and domain adaptation.

Dataset. The dataset contains more than $340,000$ reviews from 25 different types of products from Amazon.com. For simplicity (and comparability), we follow the convention of (Chen et al., 2011a; Glorot et al., 2011) and only consider the binary classification problem whether a review is positive (higher than 3 stars) or negative (3 stars or lower). As mSLDA and SDA focus on feature learning, we use the raw bag-of-words (bow) unigram/bigram features as their input. To be fair to other algorithms that we compare to, we also pre-process with tf-idf (Salton and Buckley, 1988) and use the transformed feature vectors as their input if that leads to better results. Finally, we remove five domains which contain less than $1,000$ reviews.

Different domains in the complete set vary substantially in terms of number of instances and class distribution. Some domains (books and music) have hundreds of thousands of reviews, while others (food and outdoor) have only a few hundred. The proportion of negative examples in different domains also differs greatly. There are a total of 380 possible transfer tasks (*e.g.* $Apparel \rightarrow Baby$). To counter the effect of class- and size-imbalance, a more controlled smaller dataset was created by Blitzer et al. (2007), which contains reviews of four types of products: books, DVDs, electronics, and kitchen appliances. Here, each domain consists of $2,000$ labeled inputs and approximately $4,000$ unlabeled ones (varying slightly between domains) and the two classes are exactly balanced. Table 1 contains the statistics on the complete set as well as the control set. Almost all prior work provides results only on this smaller set with its more manageable *twelve* transfer tasks. We focus most of our comparative analysis on this smaller set but also provide results on the entire data for completeness.

Methods. As *baseline*, we train a linear SVM on the raw bag-of-words representation of the labeled *source* and test it on *target*. We also include the results of the same setup with dense features obtained by projecting the entire data set (labeled and unlabeled *source+target*) onto a low-dimensional sub-space with PCA (we refer to this setting as *PCA*). Besides these two baselines, we evaluate the efficacy of a linear SVM trained on features learned by mSLDA and two alternative feature learning algorithms, Structural Correspondence Learning (*SCL*) (Blitzer et al., 2006) and

DOMAIN	LABELED	UNLABELED (TEST)	NEG. INPUTS
COMPLETE (LARGE) SET			
APPAREL	4470	4470	14.52%
BABY	2046	2045	21.46%
BEAUTY	1314	1314	15.94%
BOOKS	27169	27168	12.09%
CAMERA	2652	2652	16.35%
DVDS	23044	23044	14.16%
ELECTRONICS	10197	10196	21.94%
FOOD	692	691	13.02%
GROCERY	1238	1238	13.57%
HEALTH	3254	3253	21.25%
JEWELRY	982	982	14.82%
KITCHEN	9233	9233	20.96%
MAGAZINES	1195	1195	22.64%
MUSIC	62181	62181	8.33%
OUTDOOR	729	729	20.71%
SOFTWARE	1033	1032	37.72%
SPORTS	2679	2679	18.78%
TOYS	6318	6318	19.67%
VIDEO	8695	8694	13.64%
VIDEOGAME	720	720	17.15%
CONTROLLED (SMALL) SET			
BOOKS	2000	4465	50%
DVDS	2000	3586	50%
ELECTRONICS	2000	5681	50%
KITCHEN	2000	5945	50%

Table 1: Statistics of the large and small set of the Amazon review dataset (Blitzer et al., 2007).

.

1-layer[4] *SDA* (Glorot et al., 2011). We also compare against *CODA* (Chen et al., 2011a), a state-of-the-art domain adaptation algorithm which is based on sample- and feature-selection, applied to tf-idf features. Finally, we also include a comparison with learning with Marginalized Corrupted Features (MCF) (van der Maaten et al., 2013), which also uses data corruption as a tool to improve generalization. For CODA, SDA, SCL and MCF, we use implementations provided by the authors. All hyper-parameters are set by 5-fold cross validation on the source training set[5].

Metrics. Following Glorot et al. (2011), we evaluate our results with the *transfer error* $e(S,T)$ and the *in-domain error* $e(T,T)$. The *transfer error* $e(S,T)$ denotes the classification error of a classifier trained on the labeled *source* data and tested on the unlabeled *target* data. The *in-domain*

[4] We were only able to obtain the 1-layer implementation from the authors. Anecdotally, multiple-layer *SDA* implementations only lead to small improvements on this benchmark set but increase the training time drastically. The code we obtained from the authors implements the reconstruction sampling technique that was used to speed up the training of SDA for sparse inputs. While the original raw bow inputs are sparse, the output of one-layer SDA is no longer sparse, therefore, it becomes much more expensive to train.

[5] We keep the default values of some of the parameters in SCL, *e.g.* the number of stop-words removed and stemming parameters — as they were already tuned for this benchmark set by the authors.

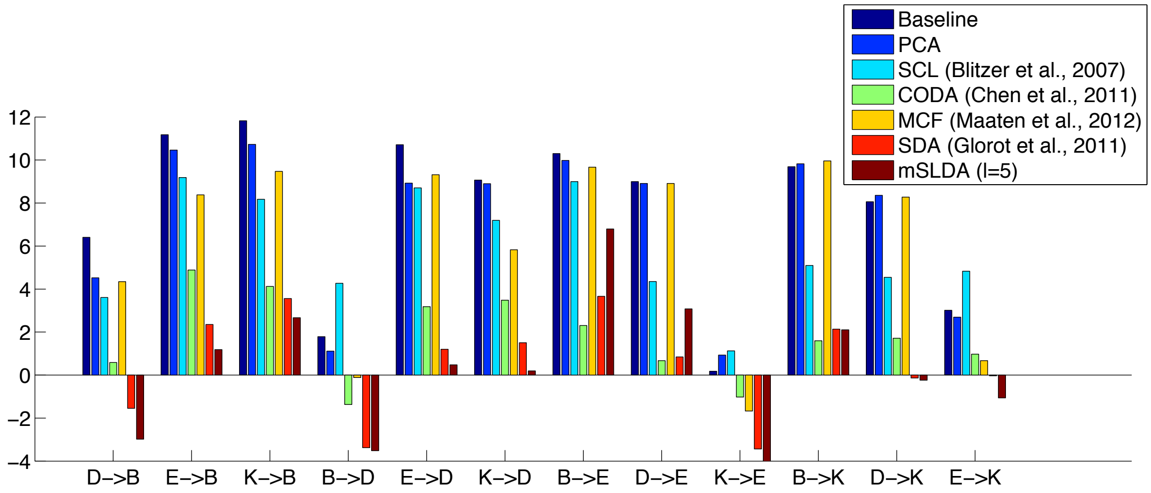

Figure 1: Comparison of mSLDA and existing works across all twelve domain adaptation task in the small Amazon review dataset.

error $e(T,T)$ denotes the classification error of a classifier that is trained on the labeled *target* data and tested on the unlabeled *target* data. Similar to Glorot et al. (2011) we measure the performance of a domain adaptation algorithm in terms of the *transfer loss*, defined as $e(S,T)-e_b(T,T)$, where $e_b(T,T)$ defines the in-domain error of the baseline (trained on the raw bow inputs). In other words, the transfer loss measures how much higher the error of an *adapted* classifier is in comparison to a linear SVM that is trained on actual *labeled target* bow data.

The various domain-adaptation tasks vary substantially in difficulty, which is why we do not average the transfer losses (which would be dominated by a few most difficult tasks). Instead, we average the *transfer ratio*, $e(S,T)/e_b(T,T)$, the ratio of the *transfer error* over the *in-domain error*. As with the *transfer loss*, a lower *transfer ratio* implies better domain adaptation.

Timing. For timing purposes, we ignore the time of the SVM training and only report the mSLDA or SDA training time.[6] As both algorithms are unsupervised, we do not re-train for different transfer tasks within a benchmark set — instead we learn one representation on the union of all domains. CODA (Chen et al., 2011b) on the other hand does not take advantage of data besides source and target. We report the average training time per transfer task.[7] All experiments were conducted on an off-the-shelf desktop with dual 6-core Intel i7 CPUs clocked at 2.66Ghz.

7.1.1 COMPARISON WITH RELATED WORK

In the first set of experiments, we use the setting from (Glorot et al., 2011) on the small Amazon benchmark set. The input data is reduced to only the $5,000$ most frequent terms of unigrams and bigrams as features.

[6] As the SVM classifier is linear, we can use the extremely efficient LIBLINEAR (Fan et al., 2008) classifier, and the training time is usually in the order of seconds.

[7] In CODA, the feature splitting and classifier training are inseparable and we necessarily include both in our timing.

Comparison per task. Figure 1 presents a detailed comparison of the transfer loss across the twelve domain adaptation tasks using the various methods mentioned. The reviews are from the domains *Books, Kitchen appliances, Electronics, DVDs*. Linear SVMs trained on the features generated by SDA and mSLDA clearly outperform all the other methods. Although MCF has been shown to be an effective approach for countering overfitting using noise corruption, it does not perform as well under the domain adaptation setting. As it only makes use of the training data from the source domain, it can not generalize to unseen words (terms) from the target domain. mSLDA and SDA have the advantage over CODA and MCF algorithm that they can make use of the unlabeled data from multiple source domains. For several tasks, the transfer loss becomes negative — in other words, a SVM trained on the transformed *source* data has higher accuracy than one trained on the original *target* data. (This is possible because there is more source data available. In particular, mSLDA or SDA make use of the abundant unlabeled data from multiple source domains to learn a more robust representation.) This is a strong indication that the learned new representation bridges the gap between domains. It is worth pointing out that in ten out of the twelve tasks mSLDA quickly achieves a lower transfer-loss than one-layer SDA.

Timing. Figure 2 (left) depicts the transfer ratio as a function of training time required for different algorithms, averaged over 12 tasks. It compares the results of mSLDA with the baseline, PCA, SCL, CODA and SDA. The time is plotted in log scale. We can make three observations: 1) SDA outperforms all other related work in terms of transfer-ratio, but is also the slowest to train. Note that the code we used for training SDA already implements the reconstruction sampling technique (Dauphin et al., 2011) that is specially designed to speed up the training of SDA on sparse inputs. However, as shown in the figure, it still takes more than 5 hours of training time. 2) SCL and PCA are relatively fast, but their features cannot compete in terms of transfer performance. 3) The training time of mSLDA is two orders of magnitude faster than that of SDA (180× speedup), with comparable transfer ratio. Training one layer of mLDA on all $27,677$ documents from the small set requires less than 25 seconds. A 5-layer mSLDA requires less than 2 minutes to train, and the resulting feature transformation achieves slightly better transfer ratio than a one-layer SDA.

Large scale results. To demonstrate the capabilities of mSLDA to scale to large data sets, we also evaluate it on the complete set with $n = 340,000$ reviews from 20 domains and a total of 380 domain adaptation tasks (see right plot in Figure 2). We compare mSLDA to SDA (1-layer). The large set is more heterogeneous in terms of the number of domains, domain size and class distribution than the small set. Nonetheless, a similar trend can be observed. Both the transfer error and transfer ratio are averaged across 380 tasks. The transfer ratio reported in Figure 2 (right) corresponds to averaged transfer errors of (*baseline)* 13.93%, (*one-layer SDA*) 10.50%, (*mSLDA, $l = 1$*) 11.50%, (*mSLDA, $l = 3$*) 10.47%, (*mSLDA, $l = 5$*) 10.33%. With only one layer, mSLDA performs a little worse than *SDA* but reduces the training time from over two days to about five minutes (700× speedup). With three layers, mSLDA matches the transfer-error and transfer-ratio of one-layer SDA and still only requires 14 minutes of training time (230× speedup).

7.1.2 FURTHER ANALYSIS

In addition to comparison with prior work, we also analyze various other aspects of mSLDA.

Word reconstruction As explained in Section 6, applying the learned reconstruction matrix to the original input amounts to smooth-out related feature values, which in turn helps alleviate the shift

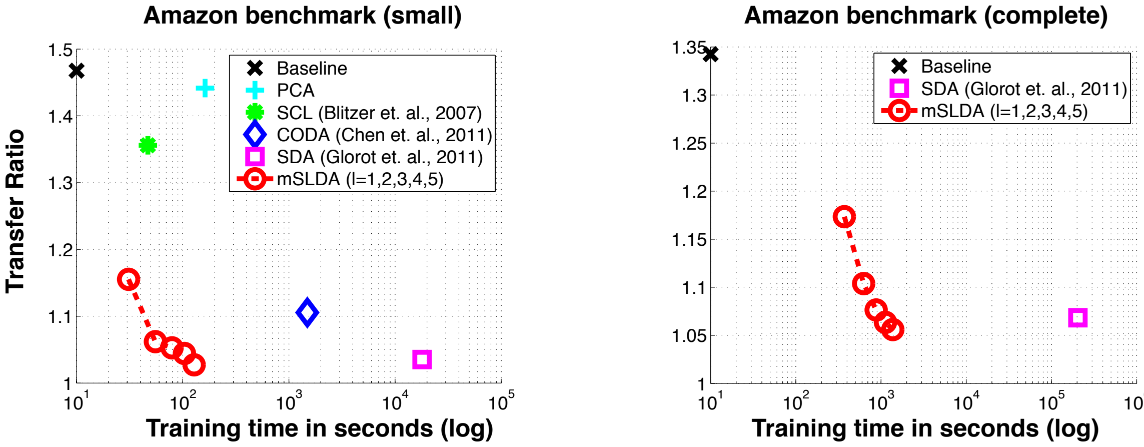

Figure 2: Transfer ratio and training times on the small (*left*) and full (*right*) Amazon Benchmark data. Results are averaged across the twelve and 380 domain adaptation tasks in the respective data sets $(5,000$ features).

between training and testing distributions. In this experiment, we apply the matrix $\mathbf{W}$ learned on the Amazon review dataset, to new input documents of a single word $\mathbf{x}$, and list the terms of the largest feature values after the smoothing $\mathbf{Wx}$. Each row of Table 2 shows an input document with a single term, and the reconstructed terms in decreasing order of feature value. As an example (*row 1*), mLDA smooths out a feature vector with a single entry at the term "great" to a denser version with values at "great for", "works great", "excellent", etc. In other words, mLDA captures word-level synonymy. The number in parentheses indicates the frequency of each word in the dataset. We can see that less frequent terms of similar meaning are reconstructed from the more frequent ones, and vice versa. As a result, a classifier trained on the smooth version of the feature vectors will be more robust, especially on rarer terms, comparing to one trained on the original sparse input.

Low-frequency features. Prior work often limits the input data to the most frequent features (Glorot et al., 2011). However there may be valuable signal in the less frequent features. We use the modification from section 5 to scale mSLDA (5-layers) up to high dimensions and include less-frequent uni-grams and bi-grams in the input (small Amazon set). In the case of SDA we make the first layer a dimensionality reducing transformation from d dimensions to 5000. The left plot in Figure 3 shows the performance of mSLDA and SDA as the input dimensionality increases (words are picked in decreasing order of their frequency). The transfer ratio is computed relative to the baseline with $d = 5000$ feature. Clearly, both algorithms benefit from having more features up to $30,000$. mSLDA matches the transfer-ratio of one-layer SDA consistently and, as the dimensionality increases, gains even higher speed-up. With $30,000$ input features, SDA requires over one day and mSLDA only 3 minutes $(458\times$ speedup).

Effect of different squashing functions. Figure 4 shows the transfer ratio of mSLDA when different squashing functions are used after applying the mapping $\mathbf{W}$. We explored four different options, linear (*i.e.*, without applying any squashing function), rectifier squashing (*i.e.*, $x \to \max(0,x)$), upper bounded rectifier units (*i.e.*, $x \to \min(1,\max(0,x))$) and the hyperbolic tangent function

great(7233)	great for(484), works great(421), excellent(1697), awesome(457), easy to(1560), love it(517), great product(318), great price(183), perfect(1252), fantastic(467)
bad(2347)	horrible(511), worst(820), a bad(383), stupid(348), awful(353), terrible(593), acting(610), movie is(654), waste(1189), lame(149)
poor(1144)	poor quality(184), poorly(385), very disappointed(284), your money(637), terrible(593), very difficult(111), save your(251), disappointing(444), returned(552), waste(1189)
return(800)	returned(552), defective(263), refund(258), arrived(356), ordered(651), shipping(463), to amazon(117), i returned(221), the item(185), received(855)
fantastic(467)	love it(517), i love(1265), excellent(1697), great(7233), amazing(650), a must(364), highly recommend(560), awesome(457), i highly(379), wonderful(975), love(3066)
is amazing(141)	amazing(650), awesome(457), love it(517), a must(364), fantastic(467), great(7233), incredible(264), a wonderful(294), well worth(234), excellent(1697)
well made(136)	sturdy(314), handles(261), kitchen(784), easy to(1560), knife(314), looks great(128), pleased(503), to clean(607), stainless(320), very nice(247)
informative(133)	covers(252), an excellent(440), information(758), sections(126), helpful(368), valuable(145), guide(268), provides(372), knowledge(288), book(5523)
awkward(119)	awkward(119), to hold(309), is too(367), too small(129), disappointing(444), difficult to(489), useless(398), desk(187), impossible to(238), way too(237)

Table 2: Term reconstruction from the Amazon review dataset. Each row shows a different input term, along with terms reconstructed from this particular input in decreasing order of feature value (from left to right). The number in the parentheses indicates the frequency of each word in the dataset.

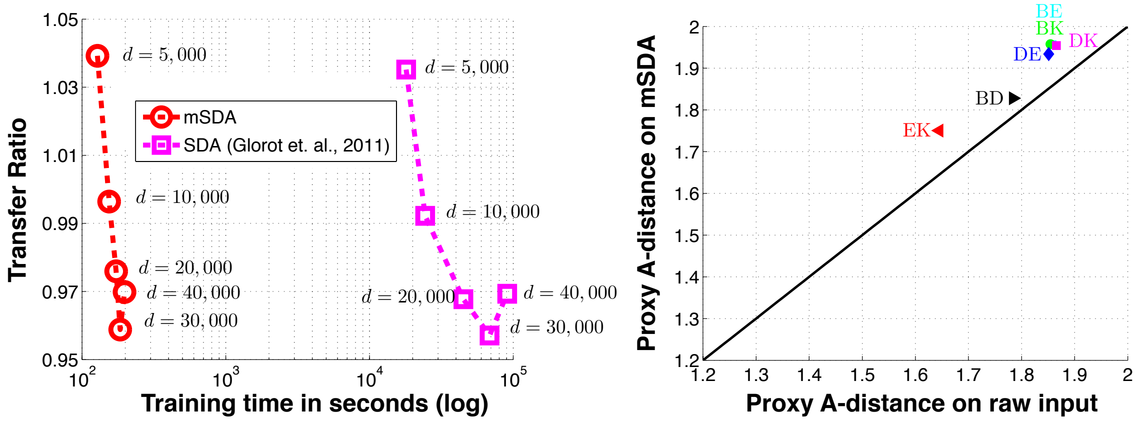

Figure 3: *Left:* Transfer ratio as a function of the input dimensionality (terms are picked in decreasing order of their frequency). *Right:* Besides domain adaptation, mSLDA *also* helps in domain *recognition* tasks.

($x \rightarrow \tanh(x)$), which we have been using in all other experiments. The blank-out noise is used in this experiment, with the corruption level cross-validated within $[0.1, 0.9]$ of 0.1 interval. As shown in the figure, the upper bounded rectifier performs similarly as the $\tanh()$ function. For the

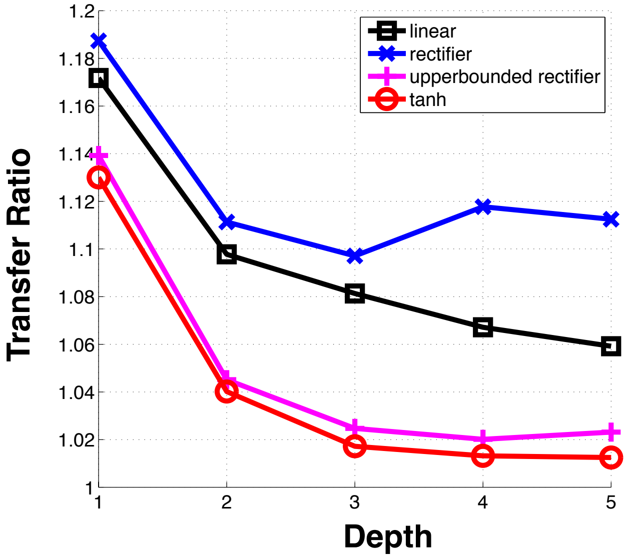

Figure 4: Transfer ratio with different squashing functions.

unbounded rectifier squashing, the performance first improves as we stack more layers, but deteriorates after three layers. The reason is that the function has no effect on large values after applying the mapping. The loss at the deeper layers is dominated by a few more frequent features while ignoring other features. One interesting observation is that even without any nonlinear squashing, the performance of mSLDA still improves as we increase the depth, as shown by the black curve in the figure.

Effect of the number of the "pivot" features. In this experiment, we investigate how the number of the "pivot" features, r, in the high-dimensional extension from Section 5 affects the performance of the algorithm. As we increase r, we would expect the transfer accuracy to be improved as well. On the other hand, since the algorithm scale cubic in term of r, the time required to solve for the mapping $\mathbf{W}$ will also increases. In the extreme case, when $r = d$, the extension reduces to our original algorithm. Figure 5 shows the transfer ratio as a function of the training time as the size of the "pivot" features increases. We do observe a reduction in transfer ratio as r increases, however, the improvement becomes marginal when r is sufficiently large (*i.e.*, 5,000). In this case, we can still finish the training of the model relatively fast (*i.e.*, within a couple of minutes).

Transfer distance. Ben-David et al. (2007) suggest the Proxy-A-distance (PAD) as a measure of how different two domains are from each other. The metric is defined as $2(1-2\epsilon)$, where ϵ is the generalization error of a classifier (a linear SVM in our case) trained on the binary classification problem to distinguish inputs *between* the two domains. The right plot in Figure 3 shows the PAD before and after mSLDA is applied. Surprisingly, the distance *increases* in the new representation — *i.e.* distinguishing between two domains becomes *easier* with the mSLDA features. We explain this effect through the fact that mSLDA is unsupervised and learns a generally better representation for the input data. This helps both tasks, distinguishing between domains and sentiment analysis (*e.g.* in the electronic-domain mSLDA might interpolate the feature "dvd player" from "blue ray",

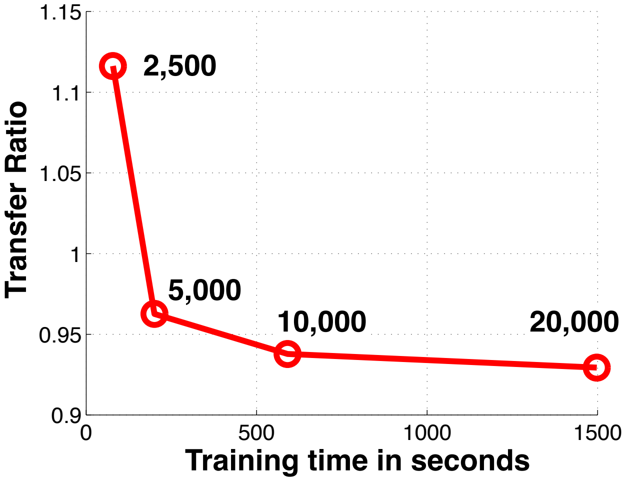

Figure 5: High dimensional extension with difference "pivot" feature size.

both are not particularly relevant for sentiment analysis but might help distinguish the review from the *book* domain.). Glorot et al. (2011) observe a similar effect with the representations learned with SDA.

Different noise model. We also apply mSLDA with the Poisson corruption model, and compare it with the blank-out noise model on the Amazon reviews data set. As shown in Figure 6, the representation learned using mSLDA with Poisson corruption also improves over the raw bag-of-word representation. Intuitively, the Poisson corruption changes the word counts and simulates the case where the same document was written with a slightly increased or decreased number occurrences of a particular word. A nice property of this corruption model is that it introduces no additional hyper-parameters. In comparison with blank-out corruption, the improvement of Poisson corruption is not as pronounced. While the Poisson corruption allows for small perturbation on the count of different words employed in the review, the blank-out noise model enables more drastic change, *i.e.*, directly removing some words. The latter scenario may reflect more closely how documents vary across domains, which results in a more robust representation. The experiment suggests that we could explore our prior knowledge on the data to properly choose the corrupting distribution used in mSLDA for better performance.

7.1.3 GENERAL TRENDS

In summary, we observe a few general trends across all experiments: 1) With one layer, mSLDA is up to three orders of magnitudes faster but slightly less expressive than the original SDA. This can be attributed to the fact that mSLDA has no hidden layer. 2) There is a clear trend that additional "stacked" layers improve the results significantly (here, up to five layers). With additional layers the mSLDA features reach (and surpass) the accuracy of 1-layer SDA and still obtain a several hundred-fold speedup. 3) The mSLDA features help diverse classification tasks, domain classification and sentiment analysis, and can be trained very efficiently on high-dimensional data.

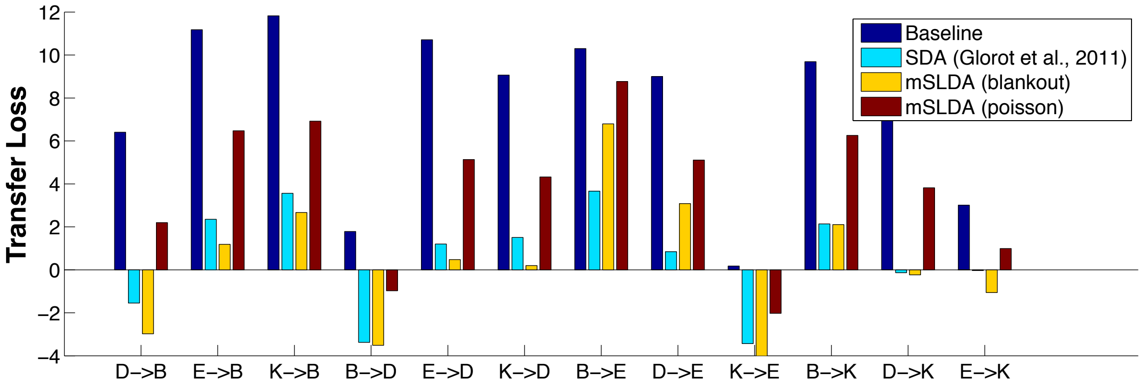

Figure 6: Comparison of mSLDA with different corruption noise in the small Amazon review dataset.

7.2 Domain adaptation on images

In this section, we evaluate mSLDA on a dataset collected by Saenko et al. (2010) for studying domain shifts in visual category recognition tasks, together with several other algorithms designed for this dataset.

Dataset. The dataset contains a total of 4,652 images of 31 categories from three domains: images from the web, images from a digital SLR camera, and images from a webcam. As shown in Figure 7, images from these domains are quite different visually. Images in the first domain are product shots downloaded from Amazon.com. The images are of medium resolution typically taken in an environment with studio lighting conditions and from a canonical viewpoint. Each category has around 90 images, capturing large intra-class variation of these categories. Images from the second domain are captured using a digital SLR camera in realistic environment with natural lighting condition. Each category has 5 different objects, and on average 3 images are captured for each object at different viewpoint. Images from the third domain are taken using a webcam. These images are of low resolution, noisy and suffer from white balance artifacts. Similar as in the second domain, 5 objects for each category are captured from different viewpoints.

Several interesting domain shifts were captured in the datasets. First, it allows us to investigate the possibility of adapting models learned on web images, which are much easier to obtain, to images captured with expensive dSLR cameras or webcams (*e.g.* mounted on robotic platforms). Second, since the same set of objects are recorded using both high-quality dSLR and the simple webcam, it allows a controlled examination of the effect of visual shift caused by different sensors.

We used the same image representation as in Saenko et al. (2010). Local scale-invariant interest points are extracted using SURF detector (Bay et al., 2006). Each image is then represented as a bag-of-visual-word with a codebook of size $d = 800$.

Our evaluation also follows the same setup as in Saenko et al. (2010). For the source domain, 8 labels per category for webcam/dSLR and 20 for amazon are available, meanwhile only three labels from the target domain are used in training as well. Five runs of experiments, each one with a set of randomly selected labels, are carried out and we report the averaged accuracies.

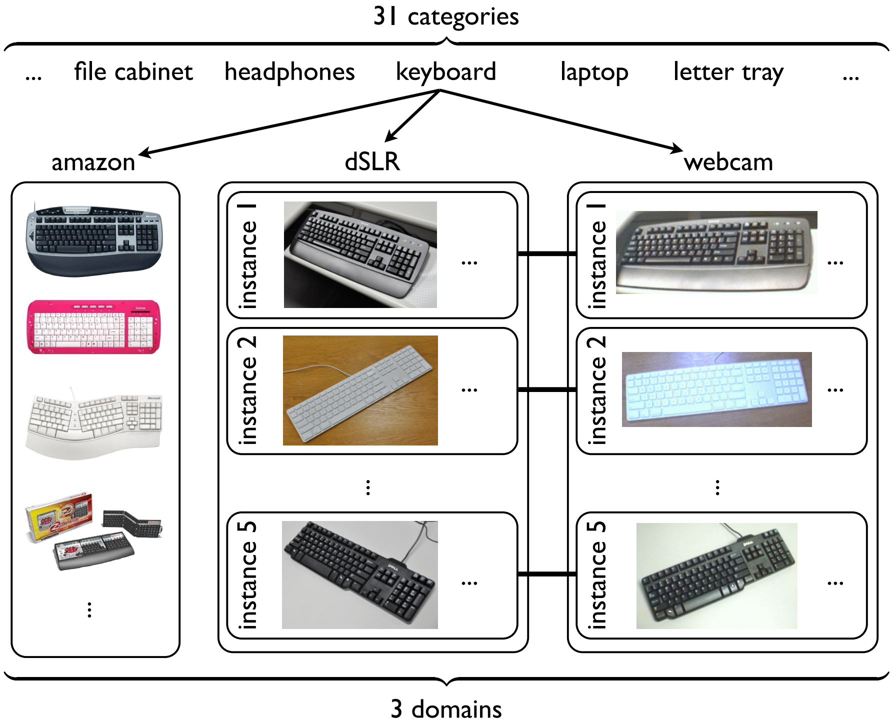

Figure 7: Sample images from the visual shift dataset. Images of objects from 31 categories are downloaded from the web as well as captured by a high definition and a low definition camera. (Saenko et al., 2010)

Methods. As *baseline*, we train a kNN model (with $k = 1$) on the raw bag-of-word representation using the *source* labeled data and test it on *target* domain (knn(A)). The same model is also trained on the combination of labeled examples from both *source* and *target* domains (knn(A+B)). We also include the results of a metric learning algorithm using information-theoretic metric learning (Davis et al., 2007). A kNN model is then trained in the projected feature space, either on all the labeled data from both domains (ITML(A+B)), or only on B labels (ITML(B)). Besides these two baselines, we also include the metric learning methods developed in Saenko et al. (2010) and its asymmetric variant by Kulis et al. (2011). For mSLDA, we present results from training both a kNN model and a linear SVM model after learning the new representation.

Table 3 summarizes the performance of these algorithms on the three domain adaptation tasks, *i.e.,* $Webcam \rightarrow dSLR$, $dSLR \rightarrow Webcam$ and $Amazon \rightarrow Webcam$. The table shows the classification test-accuracies in the target domain using various domain adaptation techniques. As we can see from comparing the two baseline algorithms, the shift between the two domains $Dslr$ and $Webcam$ is moderate since the images display the same objects and the two domains only vary in the camera resolution and lightning conditions. The adaptation between the $Amazon$ domain and $Dslr/Webcam$ involves a more drastic change, and is more challenging. mSLDA performs on par with the adapted knn methods which were especially designed on this dataset.

		Baseline		ITML		constrained ML		mSLDA	
source	**target**	knn(A)	knn(A+B)	ITML(A+B)	ITML(B)	Asymm	Symm	knn	linearSVM
webcam	dslr	.10	0.19	0.13	0.24	0.23	0.25	0.20	0.23
dslr	webcam	.26	0.28	0.20	0.27	0.28	0.29	0.31	0.38
amazon	webcam	0.08	0.22	0.10	0.28	0.27	0.23	0.28	0.27

Table 3: Domain adaptation results (accuracy) for categories seen during training in the target domain.

7.3 Semi-supervised Learning on Text

Although mSLDA was first introduced particularly for domain adaptation, it also applies to semi-supervised learning tasks. In other words, we can use mSLDA to learn more robust representations on unlabeled data, and then train a classifier on this learned representation using labeled data only.

Dataset. We use the Reuters RCV1/RCV2 multilingual, multiview text categorization test collection (Amini et al., 2010) for evaluation. The set contains documents written in five different languages (English, French, German, Spanish and Italian) which share the same set of categories (C15, CCAT, E21, ECAT, GCAT, M11). In our experiments, we only use the subset of document that are written in English, which has 18,758 documents of vocabulary size 21,531.

Methods. As baselines, we train a linear SVM on the raw bag-of-words (*BOW*) and *TF-IDF* representations of the labeled data (Sparck Jones, 1972). In addition, we also compare against Latent semantic indexing (*LSI*) (Deerwester et al., 1990). The number of retained eigenvectors was chosen by cross-validation. For both LSI and mSLDA, we learn a new representation using the full training set (without labels), and then train a linear SVM classifier on a small subset of labeled examples using that new representation.

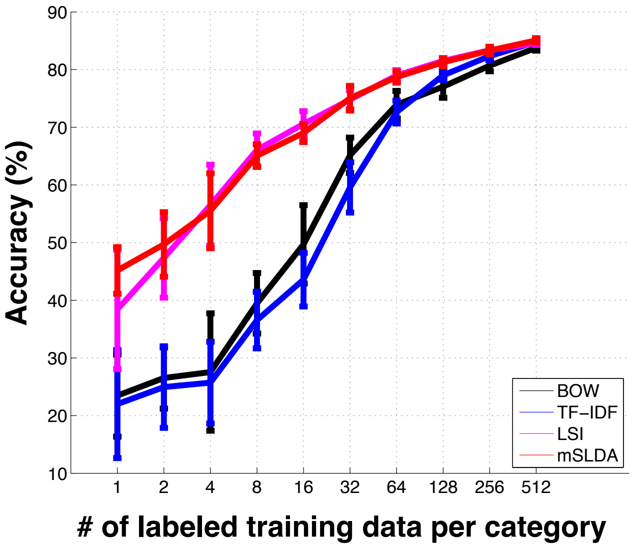

Figure 8: Semi-supervised learning results on the Reuters RCV1/RCV2 dataset.

As shown in Figure 8, we gradually increase the size of the labeled subset. For each setting, we average over 10 runs of each algorithm and report the mean accuracy as well as the variance. mSLDA performs similarly to LSI, and significantly outperform the baseline methods that were trained without unlabeled data. In summary, mSLDA learns a better representation for sparse BOW text data — however the improvement is not as pronounced as for domain adaptation. Since learning mSLDA features is cheap, it can be used as an alternative feature representation for text.

8. Discussion

In this paper we presented, mSLDA, an algorithm that marginalizes out corruption in SDA training. A key step to making this marginalization tractable, is to limit all layers within the SDA to be linear. One interesting question is to what degree this limits the expressiveness of mSLDA. As we show in our empirical results, Section 7, if mSLDA is used for feature learning, this seems to hardly matter (although more layers are necessary — something that is not really a problem as mSLDA training is so much faster.) However, the original SDA can also be used for supervised training (with fine-tuning), which is not possible with the mSLDA formulation. Maybe the fact that mSLDA works so well for bag-of-words data tells us something about the features learned by SDA. Instead of uncovering hidden concepts, as pointed out by Vincent et al. (2010), it may be more important (or simply sufficient) to learn a *common feature representation* across domains. This representation translates features from both domains into a joint space and because bag-of-words data is high dimensional, a linear mapping may just be powerful enough. Recent studies by Chen et al. (2014) seem to suggest that on more difficult image data sets the non-linear hidden representations are more important and mSLDA cannot match the performance of the original SDA.

It is an interesting observation that stacking multiple mLDA layers helps to improve these representations. One interpretation of mSLDA is to view it as a directed graph algorithm. The weight matrix $\mathbf{W}$ represents the weights of the directed edges, *i.e.* the edge from feature d to feature b has weight W_{bd}. The non-zero entries in the binary document vector $\mathbf{x}$ correspond to nodes in this graph. The transformation $\mathbf{Wx}$ takes one step in this graph, starting from the terms in $\mathbf{x}$, and accumulates the edge weights for every other term/node that is reached with this step. Stacking multiple mLDA layers is then equivalent to taking multiple consecutive steps in this fashion. Why is this helpful? Imagine two words have similar meanings but rarely co-occur. For example the terms *Obama* and *Reagan* both refer to presidents of the United States but probably rarely appear in the same sentence. If we want to perform domain adaptation from articles written in the 1980s to the 2010s, it would be good to learn that these two words refer to related entities. A single layer mLDA would learn to reconstruct co-occurring words from the term *Reagan*, such as *White House*, *President*, *United States* and it would "reconstruct" these words, but it would not reconstruct the term *Obama*. It would however also learn, from the unlabeled target data, that these very same words co-occur with the term *Obama* in more recent documents. So in the second layer it will reconstruct the word *Obama* from the terms it added in the first layer. In the graph view of mSLDA this means that *Reagan* and *Obama* are not connected through heavily weighted direct edges, but they are connected through heavily weighted two-step paths.

One interesting aspect of mSLDA is that there are only very few hyper parameters. Because training is so fast, these can be set very efficiently with cross-validation. In contrast, setting the hyper-parameters of SDA in an optimal fashion is much more time consuming. In our experiments we did not take this into account, but it is another important factor, as it may make it significantly easier to actually find the optimal hyper-parameters for mSLDA in practice—something that will improve testing accuracy and training time alike.

Finally, although in this manuscript we primarily focused on blank-out and Poisson corruption, our proposed framework is decisively general. Different corruption distributions can be chosen for different applications, in particular when side information is available. For example, if data consists of unreliable sensor readings, then blank-out corruption could be used where the probability of blank-out is fine-tuned for each specific feature —mimicking the actual drop out rate of that particular sensor. As future work, it is also conceivable that the distribution could be learned with a generative model from the data directly.

9. Acknowledgements

We would like to thank Laurens van der Maaten for pointing out the alternative Ridge Regression formulation of mSLDA under blank-out corruption. KQW, ZX, MC were supported by NSF grants 1149882 and 1137211. FS is supported by NSF IIS-0957742, DARPA CSSG N10AP20019 and D11AP00278. The authors would also like to thank Yoshua Bengio for helpful discussions.

References

Massih R Amini, Nicolas Usunier, and Cyril Goutte. Learning from multiple partially observed views-an application to multilingual text categorization. In *Advances in neural information processing systems*, volume 1, pages 28–36, 2010.

Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *Advances in Neural Information Processing Systems*, 19:137, 2007.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, volume 4, page 3. Austin, TX, 2010.

Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia, 2006.

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

Christopher JC Burges and Bernhard Schölkopf. Improving the accuracy and speed of support vector machines. In *Advances in Neural Information Processing Systems*, pages 375–381, 1997.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Boosted multi-task learning. *Machine learning*, 85(1-2):149–173, 2011.

Minmin Chen, Yixin Chen, and Kilian Q Weinberger. Automatic feature decomposition for single view co-training. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 953–960, 2011a.

Minmin Chen, Kilian Q Weinberger, and John Blitzer. Co-training for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 2456–2464, 2011b.

Minmin Chen, Zhixiang Xu, Fei Sha, and Kilian Q Weinberger. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 767–774, 2012.

Minmin Chen, Kilian Q Weinberger, Fei Sha, and Yoshua Bengio. Marginalized denoising autoencoders for nonlinear representations. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1476–1484, 2014.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Hal Daume III. Frustratingly easy domain adaptation. In *Association for Computational Linguistics*, page 256, 2007.

Yann N Dauphin, Xavier Glorot, and Yoshua Bengio. Large-scale learning of embeddings with reconstruction sampling. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 945–952, 2011.

Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.

Scott C Deerwester, Susan T Dumais, Thomas K Landauer, George W Furnas, and Richard A Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

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

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520, 2011.

Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2066–2073. IEEE, 2012.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*, pages 601–608, 2006.

Koray Kavukcuoglu, Marc Aurelio Ranzato, Rob Fergus, and Yann Le-Cun. Learning invariant features through topographic filter maps. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1605–1612. IEEE, 2009.

Brian Kulis, Kate Saenko, and Trevor Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1785–1792. IEEE, 2011.

Honglak Lee, Yan Largman, Peter Pham, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in neural information processing systems*, 22:1096–1104, 2009.

Qian Liu, Aaron Mackey, David Roos, and Fernando Pereira. Evigan: a hidden variable model for integrating gene evidence for eukaryotic gene prediction. *Bioinformatics*, 2008.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In *Advances in Neural Information Processing Systems*, pages 1041–1048, 2009.

David McClosky, Eugene Charniak, and Mark Johnson. Reranking and self-training for parser adaptation. In *Proceedings of the 44th Association for Computational Linguistics*, pages 337–344. Association for Computational Linguistics, 2006.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 833–840, 2011.

David E Rumelhart, Geoffery E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Computer Vision–ECCV 2010*, pages 213–226. Springer, 2010.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

Laurens van der Maaten, Minmin Chen, Stephen Tyree, and Kilian Weinberger. Learning with marginalized corrupted features. In *Proceedings of the International Conference on Machine Learning*, 2013.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 9999:3371–3408, 2010.

Sida Wang and Christopher Manning. Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 118–126, 2013.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.

Zhixiang Eddie Xu, Minmin Chen, Kilian Weinberger, and Fei Sha. From sbow to dcot marginalized encoders for text representation. In *CIKM*, pages 1879–1884, 2012.

Gui-Rong Xue, Wenyuan Dai, Qiang Yang, and Yong Yu. Topic-bridged plsa for cross-domain text classification. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 627–634. ACM, 2008.