Knowledge Transfer with Low-Quality Data: a Feature Extraction Issue

Brian Quanz, Jun Huan, Meenakshi Mishra

Information and Telecommunication Technology Center,
Department of Electrical Engineering and Computer Science,
University of Kansas
Lawrence, Kansas, 66045 USA
{bquanz,jhuan,mmishra}@ku.edu

Abstract—Effectively utilizing readily available auxiliary data to improve predictive performance on new modeling tasks is a key problem in data mining. In this research the goal is to transfer knowledge between sources of data, particularly when ground truth information for the new modeling task is scarce or is expensive to collect where leveraging any auxiliary sources of data becomes a necessity.

Towards seamless knowledge transfer among tasks, effective representation of the data is a critical but yet not fully explored research area for the data engineer and data miner. Here we present a technique based on the idea of sparse coding, which essentially attempts to find an embedding for the data by assigning feature values based on subspace cluster membership. We modify the idea of sparse coding by focusing the identification of shared clusters between data when source and target data may have different distributions. In our paper, we point out cases where a direct application of sparse coding will lead to a failure of knowledge transfer. We then present the details of our extension to sparse coding, by incorporating distribution distance estimates for the embedded data, and show that the proposed algorithm can overcome the shortcomings of the sparse coding algorithm on synthetic data and achieve improved predictive performance on a real world chemical toxicity transfer learning task.

I. INTRODUCTION

Knowledge transfer, modeling data that are from related but not identically distributed sources, is a problem of fundamental importance in knowledge discovery and data engineering. It has been extensively demonstrated through experimental study that traditional modeling methods typically perform drastically worse when the identically distributed assumption no longer holds (e.g. [7], [6], [10], [20]). A recurring knowledge transfer scenario that arises naturally in many application domains is the task of using a set of often high-quality, labeled auxiliary data that is expensive to obtain, to help predict the labels of a set of new data believed to come from a different but similar distribution and having little or no label information.

Knowledge transfer (e.g. transfer learning, domain adaption, learning with out-of-domain data) has attracted significant research interest from the machine learning and data mining community [1], [10], [13], [21], [22], [23], [26]. Many learning and mining algorithms have been developed, including those based on exploring the clustering structure of data [10], [26], sampling strategies which select samples that are more likely coming from the same distribution [1], [13], [23], shared feature structure between the training data to testing data [21], [22], and latent variables for related tasks [15], [27], [28].

In this paper we investigate the problem of knowledge transfer in a totally different direction and focus on preprocessing techniques that are widely used in data engineering research. In particular, we notice that effective representation of the original data is a critical but yet not fully explored research area for knowledge transfer. Feature extraction methods have been widely utilized in data engineering for creating a suitable representation for subsequent modeling practices. One of the most commonly used feature extraction methods is Principle Component Analysis (PCA) [12], in which an ordered orthogonal basis is found for a set of data with the first vectors in the basis capturing most of the variance in the data, and the projection of the data instances on some top number of basis vectors is taken as the extracted feature representation. PCA based methods have also been applied to perform feature extraction for knowledge transfer tasks (e.g. directly in [27], in a kernel space in [20], and for comparison in [21], [2]). The direct application of PCA based methods for knowledge transfer, however, usually does not lead to optimal results due to various reasons. First different distributions of source and target data may mislead the direction of the principle components. Second, for high dimensional data where data are often clustered in subspaces rather than the full space, PCA may not reveal the best representation of the data.

Towards the end goal of effective data representation, we develop a general approach to feature extraction and data representation based on a technique called sparse coding. Sparse coding is widely used in high-dimensional data preprocessing for identifying a (small) group of higher-order features of data from the raw representations [19], [21]. Such higher-order features are suitable for subsequent analysis including subspace clustering [9] and missing value imputation [3]. The limitations of sparse coding are that sparse coding still does not explicitly consider distribution distance and can result in poor embeddings for knowledge transfer.

To address the limitations and enable effective feature extraction for data that may come from different distributions, we extend sparse coding to incorporate a regularization term that can in effect be used to control how identical the distributions for different data sets are under the learned embedding. In this

way we hope to obtain an underlying structure that allows easy knowledge transfer. We evaluate the proposed method with synthetic and real data experiments, including an application to drug toxicity prediction.

II. RELATED WORK

A. Feature Extraction with Sparse Coding

Sparse coding itself has been used for transfer learning [21] the idea being that it is able to capture higher level features of the data which can then be used to allow knowledge transfer (see discussion in Section III-C for details).

Recently Xie et al. considered the related problem of transfer learning for data with overlapping feature sets, focusing on the application to web text data, where topics and term usage in web documents can change over time and between domains so that the leveraging pre-existing data can be modeled as a transfer learning task with data sets having differing but overlapping feature sets [27]. This is closely related work to the problem we consider here, and is a special case of transfer learning with missing values. They proposed to use the shared features to build regression models for predicting the missing values, then perform singular value decomposition to find a lower dimensional structure explaining the data and allowing the knowledge transfer. The approach has two key shortcomings. First, imputation and learning the embedding are performed separately, but the underlying structure is what explains the missing values so that the latent structure and imputation should be learned in tandem; from matrix completion theory we know finding the lowest rank matrix that matches the non-missing values allows perfect matrix completion under certain conditions [4], [3]. Secondly, traditional embedding techniques like SVD used in the previous approach can actually find poor embeddings for transfer learning since they are designed to approximate the data well and do not explicitly consider trying to make the data IID, in fact as we demonstrate with simple synthetic examples in a later section, the embeddings found can actually hinder transfer learning. Although not tested here we also describe how our algorithm can handle missing value imputation in tandem with the embedding process.

B. Transfer Learning and Domain Adaption

Many learning algorithms have been developed for knowledge transfer. A common approach is a model-based approach in which the different distributions are incorporated in a model, e.g. through domain specific priors [5] or through a model with general and domain-specific components [8]. Several approaches have also been developed for transductive transfer learning which consider the local structure of the unlabeled data, utilizing some unsupervised learning methods, such as clustering [10] or co-clustering [26]. There are methods based on model selection, selecting features that generalize well across distributions [18], [20], [22]. The difference between feature selection and feature generation is that we want to “discover” new features, based on the existing features, for knowledge transfer and we do so in a regularization framework, which aims to avoid over-fitting and minimize the generalization error.

III. METHODOLOGY

A. Notation

We use the following notations throughout the rest of the paper. We use lowercase letters to represent scalar values, lower-case letters with an arrow to represent vectors (e.g. $\vec{\omega}$), uppercase letters to represent matrices, and uppercase calligraphic letters to represent sets. Unless stated otherwise, all vectors are column vectors. We use $||A||_F$ to denote the Frobenius norm of a matrix $A$, $\sqrt{\text{Tr}(A^T A)}$, where $\text{Tr}$ denotes the trace; $||\vec{a}||_1$ denotes the $L_1$ norm of the k-dimensional vector $\vec{a}$, $\sum_{i=1}^{k} |a_i|$. Note, for convenience we use: $A_{ij}$ to denote the $i^{th}$ column vector of the matrix $A$, $A_{i:}$ to denote the $i^{th}$ row vector of the matrix $A$, and $A_{ij}$ to denote the $(i, j)^{th}$ entry of $A$, and similarly $a_i$ to denote the $i^{th}$ entry, or coefficient, of the vector $\vec{a}$. Additionally matrix powers are taken as entry-wise powers, for example, $A^2$ denotes the matrix obtained by squaring each entry in $A$.

B. Preliminary Background on Sparse Coding

Given a set of $n$ $p$-dimensional data points, $\{x_1, x_2, \ldots, x_n\}$ $\subset \mathbb{R}^p$, we form the $p \times n$ data matrix $X$ by taking $x_i$ as column $i$, $i = 1, \ldots, n$. The goal of sparse coding is to learn a set of $r$ $p$-dimensional basis vectors, $\{b_1, \ldots, b_r\}$ $\subset \mathbb{R}^p$ forming $p \times r$ basis matrix $B$ with column $i = b_i$, $i = 1, \ldots, r$, and a set of $n$ $r$-dimensional sparse (having few non-zero values) weight vectors, $\{\vec{w}_1, \ldots, \vec{w}_n\}$ $\subset \mathbb{R}^p$ forming weight matrix $W$ with column $i = \vec{w}_i$, $i = 1, \ldots, n$, that approximate the original patterns well, that is, $BW \approx X$. Assuming the reconstruction error for a data pattern $\vec{x} - BW$ follows a zero-mean Gaussian distribution with covariance $\sigma^2 I$, and taking a Laplace prior for the weight coefficients and assuming a uniform prior on the basis vectors, then the posterior probability of the data for a given $B$ and $W$ is proportional to Equation 1.

$$\prod_{i=1}^n e^{-||x_i - BW_i||_2^2/(2\sigma^2)} e^{-\alpha||w_i||_1}. \quad (1)$$

The maximum a posteriori estimate for the basis and vectors can then be found by maximizing the log of Equation 1 with the following optimization problem [16] :

$$\arg \min_{B, W} \frac{1}{2\sigma^2} ||X - BW||_F^2 + \alpha \sum_{i=1}^n ||w_i||_1$$

s.t. $||b_i||_2^2 \leq c \quad \forall i = 1, \ldots, n \quad (2)$

where the constraints on the norm of the basis vectors are introduced to prevent them from growing infinitely large, and can be viewed as regularization on the basis vectors as well. Typically $c$ is fixed, e.g. to 1, since allowing the basis norms to be bigger would allow the basis weights, the entries of $W$, to shrink (reducing the $L_1$ norm, and still produce the same reconstruction, so that the effect $\alpha$ has would change.
Here $\alpha$ acts as a tunable regularization parameter, trading off between sparsity of the weights and approximation of $X$. The resulting new data representation is then given by $W$. We label this sparse coding feature extraction method as SC in our experiments.

The problem in 2 is non-convex, but fixing either $W$ or $B$ the problem becomes convex in the other (i.e. fix $W$ and the problem is convex in $B$ and vice versa). This was exploited in [16] along with a Lagrange dual solution for learning the basis to derive an efficient algorithm for solving this problem, by alternatively fixing $W$ or $B$ and solving for the optimal value of the other. We thus take a similar alternating optimization approach for our algorithms, as described in subsequent sections.

C. Advantages and Limitations of Sparse Coding for Feature Extraction in Knowledge Transfer

One benefit of sparse coding for knowledge transfer comes from the viewpoint of sparse coding as a way of learning higher-order more general representations of data from the given low level representations [19], [21]. By forcing the representations to be sparse combinations of the basis vectors it helps to ensure that the basis found is efficient at representing the set of patterns and generally captures the main patterns of interest in the low level input representations. The idea then is that while the low-level details for different data sets may be different, they will have some commonalities, or overlap, in the higher-level representation that allows general principles to be inferred in this higher order representation that are applicable to the different data sets. Such an approach has been applied to learning higher order representations for knowledge transfer using auxiliary data sources [21], [2], [17]. However the fundamental assumption here then must be that the data sets are identically distributed in this higher order representation - if they are not, then the higher order representation will still have the same issue as before - of non-identically distributed data, and will still not enable knowledge transfer. As it is, sparse coding provides no such guarantee.

Another way of viewing sparse coding which potentially offers more insight is from a geometric perspective; sparse coding can be viewed as a way of performing subspace clustering. By forcing the new data representations to be sparse the algorithm tries to find a set of representative vectors or directions with the representations only being active among a few of the basis vectors - the set of vectors for which a datum representation is non-zero could be seen as its subspace membership. It can be shown that if the data points lie in a set of independent subspaces, then sparse coding can be used to fully identify the subspace clusters [9]. In this sense sparse coding could be seen as being useful for knowledge transfer in the same sense as other cluster-based transfer learning methods: by identifying the shared cluster structure of the auxiliary data with the target data, it can in effect select only those auxiliary data belonging to the same clusters as the target data for extracting knowledge, or learning patterns, since only those data will have the same sets of features active as the target data. The active features in the new representations can then be viewed as the coordinates in the shared subspaces for the found basis. This ability to handle multimodal data is a major advantage of the sparse coding algorithm over other embedding algorithms such as principle component analysis [12] which only looks at directions of greatest variance completely missing any internal structure and further restricting all basis vectors found to be perpendicular. However a fundamental issue here with sparse coding comes from the case of target data and auxiliary data lying mostly in different subspaces. In the case of an auxiliary data set and a target data set lying in different subspaces, sparse coding will generally result in representations for which no active features are shared between the two data sets, since each will only have non-zero weights for those basis vectors belonging to its own subspace (see Section IV-A for an illustration of this case). In this case no knowledge transfer is possible because the only non-zero features in the target data will always be zero in the auxiliary data, so the auxiliary data cannot be used to help determine patterns for those features and thus the target data. Nevertheless, just because the shared cluster assumption used by sparse coding and many other knowledge transfer methods no longer holds does not mean we should abandon our hope of utilizing available high-quality auxiliary data. In the next few sections we propose some modifications to sparse coding to allow knowledge transfer in such cases, and more generally for whenever the embedding found still does not result in identically distributed data.

Another issue with sparse coding comes from selecting the size of the basis. In an unsupervised setting where we learn a basis and weights that explain all of the data best, as we allow the basis to grow beyond a certain size, the possible generalization shrinks. It is easy to see that if we allow the basis dimension to equal the number of points, that a basis that minimizes the objective function is given by one basis vector in the direction of each input data point. First all basis vector norms will be maximized in order to allow minimum weights. Because the L1 penalty is used the additional penalty is the same for larger weight values, so the smallest weight possible always comes from a direct path to a data point. In this case sense each point would be assigned to its own coordinate, no patterns could be found from the data. As we allow the basis to grow, sparse coding basically becomes similar to a weighted k-nearest-neighbor algorithm [29].

D. Improving Sparse Coding with Regularization

A fundamental limitation as described in the last section is that sparse coding may actually find an embedding that hinders knowledge transfer - there is nothing forcing the data sets in the new feature representations to be identically distributed. Since our goal is to transfer knowledge when data distributions are not identical in order to utilize auxiliary data, it therefore makes sense to address this problem by trying to enforce the the embedded data sets to be identically distributed. To do this we propose to incorporate a distribution distance estimation between the embedded data sets. Following the
regularized regression framework in Equation 2, to incorporate distribution distance, we add a tunable regularization term on the embedding weights for the two data sets that penalizes the estimated distribution distance between these sets of weights. This type of regularization could be viewed as a soft constraint that enforces the estimated distributions of the different data sets to be identical. The new optimization problem is given in Equation 3, where $U$ and $V$ are used to denote the weights for the training (source) and test (target) sets respectively, for convenience, $p$ and $q$ represent the probability density functions (pdfs) for each set respectively, and $d(\cdot)$ some distribution distance function.

$$\arg\min_{B,W} \|X - BW\|_F^2 + \alpha \sum_{i=1}^{n} \|\vec{u}_i\|_1 + \beta d(p_U,q_V)$$

s.t. \[\|\vec{b}_i\|_2^2 \leq c \quad \forall i = 1, \ldots, n\]

(3)

Since the penalty only includes the weight terms, we can still perform the alternating optimization. Here $\beta$ is another tunable regularization parameter which controls the importance given to enforcing small distribution distance. In this case most distribution distance measures will result in a non-convex problem for fitting $W$. Thus we can only find a local solution. Avoiding this non-convexity is an open problem since accurate distribution distance measures as functions of the finite-dimensional embedding can have multiple local minima (as illustrated in Section IV-A) unless simpler but also less accurate distribution distance measures are used. Note that since the distribution distance only depends on $W$ the problem remains unchanged and is still convex when $W$ is fixed.

In general, most probability distribution distance measures require the pdfs of the two distributions in question. One commonly used measure that is an exception is the maximum mean discrepancy (MMD) estimate [11], [20], that is useful in some kernel spaces, but in the original input space (i.e. with a linear kernel) provides only a weak measurement, for example not being able to distinguish between two different distributions with the same mean. To use a more accurate distribution distance measure, we therefore need to estimate the pdfs of the two distributions. In order to do this we propose to use a nonparametric density estimation technique, kernel density estimation; this can be thought of as providing a smoothed histogram.

In general estimation tasks, the usefulness of kernel density estimation is somewhat limited due to the curse of dimensionality, with the risk of the estimator growing with the dimensionality of the data [25]. However in our case there are several benefits to using kernel density estimation. First, since we need to restrict the dimensionality of the data to some degree to allow generalization between data sources, this should alleviate to some extent the curse of dimensionality. Secondly, we are not actually concerned with estimating the densities, just determining a difference in the densities of two distributions and how this changes as the data changes, so as long as this difference and change is captured it doesn’t matter how accurate the density estimation is. Finally, using a differentiable kernel function in the estimation enables straightforward computation of derivatives which allows easy incorporation in standard optimization techniques like gradient descent. Since the specific kernel function chosen is not very important for kernel density estimation [25] we use the differentiable Gaussian kernel $k(x,y) \propto \exp(-(1/2h)||\vec{x} - \vec{y}||^2_2)$ where $h$ is the kernel width, in our implementations.

With this approach we can then use a wide variety of distribution distance measures that use the pdfs, including f-divergences such as $\chi^2$-divergence and Kullback-Leibler divergence and $L_p$-norm distance measures. Here we use the symmetric version of the common KL-divergence measure, the Jensen-Shannon divergence. The KL-divergence is given by $d_{KL}(P||Q) = E_P[\log(p/q)]$ and JS-divergence is $d_{JS} = 0.5(d_{KL}(P||Q) + d_{KL}(Q||P))$. In general computing the KL-divergence for multivariate data with continuous variables is still an open problem, but by estimating the density we can then use the sample mean approximation to expected value given our data sample to predict the KL-divergence as the expected value of the log-odds of the pdfs. Below we derive expressions for the distance measure, and the gradient of the distance measure.

We use $K$, $G$, and $S$ to denote the kernel matrices for $U$ with itself, $U$ with $V$ and with itself, e.g. $G$ is an $n \times m$ matrix with entries $G(i,j) = \exp(-(1/(2h)||\vec{u}_i - \vec{v}_j||^2_2))$, where $h$ is the kernel width. Then to calculate the probability vectors for each data set under each distribution, we have the following:

$$\begin{align*}
\vec{p}_u &= (1/(n(2\pi h)^{d/2}))K^T, \\
\vec{q}_u &= (1/(m(2\pi h)^{d/2}))G^T, \\
\vec{p}_v &= (1/(n(2\pi h)^{d/2}))G^T, \\
\vec{q}_v &= (1/(m(2\pi h)^{d/2}))S^T,
\end{align*}\quad (4)$$

where e.g. $\vec{p}_v$ represents the pdf for the first data set ($U$) evaluated at each point in the second data set $V$ and $\vec{1}$ denotes a vector of all ones of the appropriate length. Then the JS divergence estimate is given with Equation 5.

$$d_{JS} = \frac{1}{2}((\vec{p}_u - \log(\vec{p}_u))/n + \vec{1}^T(\log(\vec{q}_u) - \log(\vec{p}_u))/m)$$

(5)

Then the gradient for the $l_{th}$ column of $U$ and $V$ is given in Equation 6.

$$\begin{align*}
\nabla_{\vec{u}_l}d_{JS} &= \frac{1}{2nh}(U - \vec{u}_l)(K_{l:}/\vec{p}_u + K_{l:}/p(\vec{u}_l)) \\
&- \frac{1}{2mh}G^T_{l:}/q(\vec{u}_l) + G^T_{l:}/\vec{p}_v \\
\nabla_{\vec{v}_l}d_{JS} &= \frac{1}{2nh}(U - \vec{v}_l)(S_{l:}/\vec{q}_v + S_{l:}/q(\vec{v}_l)) \\
&- \frac{1}{2mh}(U - \vec{v}_l)(G_{l:}/p(\vec{u}_l) + G_{l:}/\vec{q}_u)
\end{align*}\quad (6)$$

From Equation 6 we see that moving in the direction of the negative computed gradient makes sense intuitively as a rule to bring two distributions closer together. The distribution distance gradient component for a given embedded point $x$ corresponds to summing the vectors from $x$ to each of the
classes, \( \mathcal{C} \) data well, generally by estimating the probability density function (pdf) evaluated at \( x \) and the strength of the kernel value in the total density estimate for that value. In other words, with gradient descent \( x \) will tend to move toward the points of the other data set, and away from the points in its own data set, in a weighted manner. However, by also including the term causing the embedding to represent the input matrix well this should help counter the diffusion effect for each data set. We refer to this method as sparse coding with distribution distance regularization (SCDD).

Here we considered one source data set, which could actually be a combination of several source data sets, and one target data set. It is straightforward to extend the above approach to multiple data sets, e.g. one way is to simply add additional pairwise terms as above for the additional data sets.

\[ \text{E. Incorporating Target Data Label Information} \]

A common data mining or knowledge discovery task, which is the focus of our experiments in this work, is classification, that is learning a predictive model from the data capable of determining which class a data instance belongs to from its feature representation. Specifically we have a set \( \mathcal{C} \) of \( k \) classes, \( \mathcal{C} = \{1, 2, \ldots, k\} \) and each data instance \( \vec{x}_i \) has an (known or unknown) associated class label \( y_i \in \mathcal{C} \). The final goal of classification is then to predict the labels of the target data well, generally by estimating \( P(y|\vec{x}) \) from the labeled data. Even for data where ground truth label information is expensive and time consuming to obtain, usually a small amount of label information can still be obtained. Thus we should be able to leverage this information for knowledge discovery when available.

Furthermore, distribution distance regularization may not always be enough for knowledge discovery. Enforcing small distribution distance for the distribution of the data instances for the two data sets does not guarantee the conditional distributions resulting from the embeddings will be identical. In fact since sparse coding with distribution distance will try to approximate the data well while decreasing the distribution distance, it can end up finding a local non-ideal minimum to the optimization problem 7 that misaligns the conditional distributions (e.g. compare synthetic experiments 1 and 2 in Section IV-A). In general unless it is certain the distributions of the source and target data sets are closely similar, some ground truth information for the target data is necessary to determine the correct embedding for the data.

We explored several options for incorporating conditional distribution information in the sparse coding formulation including estimating conditional and joint distributions with kernel density estimation. We found a class-based distribution distance estimation approach to work best, where we use the same distribution distance estimate as in the previous section, only calculated between the instances of the same class between the two data sets, for each class. The new objective is given by Equation 7.

\[
\arg \min_{B,W} \|X - f(BW)\|_F^2 + \alpha \sum_{i=1}^n ||\vec{a}_i||_1 + \beta d_{JS}(p_U, q_V) \\
+ \beta_2(d_{JS}(p_U1, q_{V1}) + d_{JS}(p_U2, q_{V2}))
\] (7)

Here \( U1 \) denotes those embedded data instances in \( U \) that have label 1 and \( U2 \) those that have label 2 and similarly for \( V1 \) and \( V2 \). For simplicity we just described the case of only two classes, but our approach extends easily to multiple classes, simply by using a distribution distance term for each class. Then computing the divergence and gradient for the new component is the same as in the previous section, simply restricted to each class, specifically bringing together the distributions \( P(\vec{x}|y = i) \) for each \( i \) in \( \mathcal{C} \). We refer to this method as sparse coding with distribution distance and class-based distribution distance regularization (SCDDCD).

Importantly, in our implementation we only compute the gradient component for the auxiliary data, and not for the target data, since there are typically very few target data labels. If we updated the labeled target instances as well the few labeled instances would tend to quickly move toward the other data set without influencing the embedding found for the remainder of the target data set - failing to reduce the distribution distance of the true conditional distributions since the unlabeled points would be unaffected.

F. Solving the Optimization Problems

The general approach we take to solving the optimization problems presented in the last few sections is one of block coordinate descent, or alternating optimization. We generate a random basis \( B \) of input size \( r \) then continually update the weights \( W \) to minimize the objective value while holding the basis fixed, followed by updating the basis to minimize the objective value while holding the weights fixed, until convergence.

\textit{a) Updating the Basis:} We originally tried several different approaches for fitting the basis \( B \) given fixed weight matrix \( W \), including a Lagrange dual approach, and the popular Nesterov’s method. We found that as the basis size \( r \) grew beyond only a very small size a simple projected gradient descent with a line search worked best in terms of efficiency and the embedding found. The gradient of the any of the objective functions we use from Equations 2, 3, and 7 with respect to the basis \( B \) is given by Equation 8.

\[
\nabla_B \text{obj.} = -XW^T + BWW^T
\] (8)

To update the basis we first compute the negative gradient as the step direction. After computing the new basis by adding the negative gradient, we project it onto the L2 ball constraint for each basis vector, which amounts to scaling each vector to be of max length \( c \). Then a line search is performed where the step size is decreased if the objective value does not decrease. The process is repeated until convergence.
b) \textit{Updating the Weights:} The same approach for updating the basis is used for updating the weights, except that we use the sub-gradient to incorporate the non-differentiable L1 norm regularization term, and add in the gradient terms for the appropriate distribution distance regularization terms depending on the methods used as described in Sections III-B and III-E and Equation 6. Additionally no projection is necessary since there are no constraints on the weights. The sub-gradient of the objective functions for the weight matrix $W$ excluding the distribution distance regularization terms is given in Equation 9.

$$\nabla_w \text{obj.} = -B^T X + B^T BW + \alpha \text{sign}(W)$$ \hspace{1cm} (9)

Here sign() is the sign function which returns 1 if its input is greater than 0, 0 if equal to 0, and -1 if less than 0.

G. \textit{Weighted Loss Sparse Coding}

A typical issue that arises in knowledge transfer between different sources of data is that the data have different feature sets, so that only some overlapping set of features is shared in common for different pairs of data sets, and additionally missing values are common. Our approach can easily be adapted to handle such cases by introducing a non-negative weighting matrix, consisting of zeros and ones for the case of missing values or features. For this situation we introduce a non-negative, $p \times n$ weight matrix $P$ (not to be confused with the basis coefficient matrix, or weight matrix, $W$ which represents the encoding of the data patterns for the basis $B$). This weight matrix is used to weight the reconstruction error described above, so that in the optimization problems more importance is placed on those more heavily weighted entries. This formulation can also be used to perform sparse coding for data with missing values, by simply placing a zero in $P$ at each missing entry, and ones elsewhere. The resulting optimization problem, the weighted loss sparse coding problem is given in 10, and the extensions for incorporating distribution distance regularization are the same as described previously for unweighted sparse coding.

$$\arg \min_{B,W} \|P \circ (X - BW)\|_F^2 + \alpha \sum_{i=1}^{n} \|w_i\|_1$$ \hspace{1cm} (10)

s.t. $\|\tilde{b_i}\|^2 \leq c \quad \forall i = 1, ..., n$

Here $\circ$ is the Hadamard product, the entry-wise product between two matrices. The gradient computation is similar, e.g. the gradient with respect to $B$ is given in Equation 11 and the one for $W$ takes a similar form.

$$\nabla_B \text{obj.} = -(P^2 \circ X)W^T + (P^2 \circ (BW))W^T$$ \hspace{1cm} (11)

In this work in our experiments we did not test this aspect of our algorithms. Full exploration of cases of feature generation for knowledge transfer for data with different feature sets is an area of future work.

IV. \textit{Experimental Study}

We have implemented our methods in Matlab. All experiments are run on a 178-node cluster where each node contains two Intel Xeon EM64T 3.2 Ghz processors and 4G memory. In order to evaluate the performance of the different feature extraction methods for knowledge transfer, we have created synthetic data sets and collected real-world data sets for chemical toxicity prediction for environmental protection. Below we show our experimental study results.

A. \textit{Synthetic Data Experiments}

For the synthetic data, we demonstrate the case where the target data set lies mostly in a different cluster than a source data set from which we want to enable knowledge transfer. To simulate this scenario, we generate two data sets, a source, or training, data set, and a target, or testing, data set. To generate data we randomly sample 25 points each from two simple 2D Gaussian distributions, one for each class. The first with mean $(0.6, 0)$, the second with mean $(-3, 0)$ and both with covariance matrix $\{\{1, 0\}, \{0, 5\}\}$. We then rotate the source data by some number of degrees $\theta$ and the target distribution by the same amount in the opposite direction $-\theta$, using the rotation matrix $R = \{\{\cos(\theta), -\sin(\theta)\}, \{\sin(\theta), \cos(\theta)\}\}$.

a) Synthetic Experiment 1: For the first experiment, we sample 50 points for each data set as described above and rotate the training data by $\theta = 25$ degrees and the testing by $\theta = -25$ degrees. No labeled test instances are provided for learning the embeddings.

b) Synthetic Experiment 2: We generate 50 points for each data set using the same set up as described above, except this time rotate the training data by $+55$ degrees around the origin, and the testing data by $-55$ degrees, increasing their dissimilarity and hence the difficulty of the knowledge transfer. We then randomly provide only a single label from each class for the testing data to be used in learning the embedding and final classifier.

c) \textit{Experiment Protocol:} In our experimental study, we did not do an extensive parameter search but simply picked a default value of 1 for the kernel width, the Lasso penalty weight of $\alpha_1 = 0.2$ (a larger value just tends to compress the points more along the basis directions found), and a heavy weighting of 2000 for the each distribution distance component when included. In the plots showing the results we also plot the support vector machine (SVM) decision boundary found from training on all labeled embedded data points (including the two labeled points of the test data), with default linear SVM parameter $C = 1$.

d) \textit{Experiment Results:} The results for various embedding approaches are shown in Figure 1 and Figure 2, with all figures plotted on square plots. For the first experiment, sparse coding identifies the two major subspace clusters, and actual hurts the performance since it essentially assigns each data set to one dimension. The data sets are similar enough however, that just incorporating the distribution distance regularization allows for a very good embedding to be found (Figure 1(d)).
In the second experiment, as before, sparse coding (Figure 2(d)) identifies the two major subspaces or clusters the data belong to, which does not help transfer knowledge in this case since as before each cluster corresponds to a specific data set, so each is assigned its own dimension.

As we expected, just incorporating distribution distance (Figure 2(e)) may not help tremendously in this case, since the nearest alignment of the distributions happens by misaligning the two classes between the two data sets. Incorporating the very few available test labels with distribution distance regularization between the data points of the same classes as described in Section III-E allows for a very good embedding to be found for transfer learning - the points of each class are grouped together.

In addition we plot the results for PCA in Figure 2(b). We see that PCA does not move the two distributions close and hence bears poor classification results. To show that the distribution distance minimizing alone is not enough and to demonstrate the utility of sparse coding, we show what happens if just the evenly weighted sum of distribution distance and class distribution distances are minimized with the same gradient procedure, without any sparse coding component, in Figure 2(c). This results in a poor embedding.

Furthermore we note that this example also illustrates how even restricting the basis size for PCA can easily fail: the principal component found is in the direction $(0.040, -0.999)$ which is nearly perpendicular to the best single projection direction for knowledge transfer in this case.

Finally in Figure 3 we show a more extreme case, where the same data generation process was used, but the rotation for each data set was increased by 10 degrees. In this example the basic embedding approaches completely fail whereas in-
B. Knowledge Transfer for Chemical Toxicity Prediction

We evaluated the performance of the aforementioned feature extraction approaches on an environment protection application. The overarching goal of the study is to identify efficient and accurate computational approaches to evaluate toxicity of chemicals and their effects on the environment. Collecting high-quality data for chemical toxicity study is an expensive and time-consuming process. For example, for the ToxCast data set described below, the study to obtain the animal toxicity endpoints for about 320 chemicals cost nearly 2 million dollars and took over a year to perform. In reality, there are millions of chemicals that need to be evaluated. There is no feasible experimental approach that we could imagine for collecting such data; modeling and computing are indispensable components in the battle for a clean and healthy environment.

The data engineering challenge here is to leverage high-quality data collected from the EPA and to build models for chemicals that may deviate from the source distribution. Towards that end, we collected our data sets and designed our experiments as detailed below.

a) Source data set: TOXCAST: Environmental Protection Agency (EPA) has initiated a program called TOXCAST [14] (http://www.epa.gov/nctc/toxcast/) in which they have performed a series of in vitro tests to collect features for predicting toxicity of chemicals. The TOXCAST data set included results of 309 unique chemicals from pesticides, a serious concern for environmental prediction. A total of 624 different assays, which can be classified into 9 different technologies, were used to predict toxicity of these chemicals.

In vivo toxicity responses of most of these chemicals have been compiled in another project by EPA called Toxrefdb (http://epa.gov/nctc/toxrefdb/). This study includes a complete toxicity profile of 474 different chemicals. To construct data set 1, test results from the TOXCAST data set and the chemical descriptors of the chemicals from the software Dragon were used as the feature space. The class labels of these chemicals were the toxicity of these chemicals as recorded in the Toxrefdb data set. The endpoint considered was Tumors on mouse liver. After removing duplicates and compounds with missing or inconclusive endpoint results, the data set consists of 235 chemical compounds.

b) Testing data set: CPDB: The Carcinogenic Potency Database (CPDB) (ref: http://potency.berkeley.edu/) is a widely used data resource which contains the results for carcinogenic tests on 1547 chemicals. The results in the dataset are reported on rats, mice, hamsters, dogs and nonhuman primates. All the chemicals that proved carcinogenic on mouse liver in the CPDB dataset were selected. These were around 50 in number. Thus, around 50 drugs were randomly picked from FDA approved drugs list and these constituted the non-carcinogenic class. The carcinogenic chemicals selected from the CPDB dataset and the non-carcinogenic chemicals selected from the list of FDA approved drugs together formed the second dataset (Dataset 2) with a total of 112 compounds.

c) Experiment Protocol: We convert the chemical structures to vector-format data using chemical descriptors calculated using software called DRAGON (version 5) [24]. The descriptors that we used are a total of 120 atom centered fragments descriptors calculated for each chemical. In our experience (unpublished data), such descriptors are good candidates for chemical activity prediction.

We removed any descriptors with variance 0 across both data sets, resulting in a total of 95 features. We then normalized each feature across all data to have mean 0 and variance 1. We use the source data TOXCAST and various numbers of samples from the testing data set CPDB to build a model. We then evaluate the accuracy of the model using the remainder of CPDB. For each run, we randomly sample the given number of test instances from testing data CPDB to be used in the training, and use internal cross-validation with the training data (with the cross-validation evaluation using only the labeled testing data selected to be included in the training) to select any model parameters. To simplify the model selection for the SCDD and SCDDCD methods, we fixed the kernel width $h$ to be equal to the basis size of the embedding and for SCDDCD fixed the regularization parameters for the class distribution distance and data distribution distance to be equal.

For model comparison, we collect the sensitivity (TP/(TP+FN)), specificity (TN/(TN+FP)) and accuracy.
((TP+TN)/S) for the constructed models, where TP stands for the number of true positives, FP stands for the number of false positives, TN stands for the number of true negatives, FN for the number of false negative, and S stands for the total number of samples.

All the values reported are collected from the testing data set only and are averaged across 100 experiments with mean and standard deviation reported. The results are shown in Tables I, II, and III.

Since the focus here is on feature extraction, and to have a fair comparison of the different feature extraction methods, we use a fixed classifier (SVM with fixed C and linear kernel) for all methods (including the baseline of no embedding, the original feature space). For each embedding approach, a default linear SVM classifier with parameter $C = 1$ is used on the embedded data to obtain the final predictions. The abbreviation of the methods shown are: SVM - SVM classifier is trained in the original feature space using all the auxiliary (source) data and the labeled test instances; SVMTG - no auxiliary data is used to train the SVM classifier in the original feature space, only the labeled test instances; PCA - principal component analysis is used on the combined auxiliary (training) and testing data to find the embedding; SC - Sparse coding (Section III-B, Equation 2); SCDD - sparse coding with just distribution distance regularization (Section III-D, Equation 3); SCDDCD - sparse coding with both distribution distance regularization and class-based distribution distance regularization (Section III-E, Equation 7).

d) Experiment Results: Table I shows the accuracy results for the experiments, with each row corresponding to a method and each column corresponding to a number of labeled test instance used in training. In increasing order. Table II and Table III similarly show results for the specificity and sensitivity, respectively, which provide a measure of the bias of a method toward either reducing Type I errors (false positives) or Type II errors (false negatives).

From the results we see that sparse coding incorporating both distribution distance and the class-based distribution distance components (SCDDCD) in all cases obtains the best accuracy out of all the methods. With only 4 labeled test data instances, the SVM classifier trained using no auxiliary data (SVMTG) does little better than random guessing on average, but the SCDDCD embedding method is able to raise the mean accuracy by an addition of 10 percent. As expected with very little labeled target data, utilizing the available auxiliary data becomes a necessity. As the amount of labeled test data given increases, the performance of SVMTG increases correspondingly, but the SCDDCD method still consistency out-performs the SVMTG method. Even with as many as 40 labeled test instances, utilizing the auxiliary data with the SCDDCD method still offers significant improvement over using only target data (SVMTG). For 4 labeled test instances sparse coding (SC) achieves similar performance to SCDDCD; in this case the benefit of including the test instances could be masked by noise. However, sparse coding improves more slowly with increasing labeled test data and is quickly out-performed by SVMTG. Also just incorporating distribution distance with sparse coding (SCDD) slightly hurts performance for the smaller amounts of labeled test instances, and generally performs about the same as SC. In this case it is clearly not enough to just consider the distribution distance between the data sets. Except for the first set of experiments with the number of labeled test instances equal to 4 for which PCA performed worse than SC, PCA has similar performance to the SC method and is thus also not able to most effectively utilize the auxiliary data in these experiments.

From the specificity and sensitivity results (Tables II and III) we see that all of the embedding methods that utilize the auxiliary data have a bias toward increased specificity at a cost of decreased sensitivity. However the opposite is true for the method using only the target data, SVMTG. The SCDDCD method however is somewhat more balanced.

V. CONCLUSIONS AND FUTURE WORK

For low quality data, data with little to no ground truth information coming from a different distribution than available high quality data sets, it is necessary to leverage the available auxiliary data sources to aid in knowledge discovery. In order to enable knowledge transfer for mis-matching distribution data sets we explore a feature extraction perspective, starting with the popular sparse coding approach which learns a set of higher order features for the data. After discussing the advantages and limitations of sparse coding for knowledge transfer we proposed new feature generation algorithms to address those limitations and enable knowledge transfer, and verified the effectiveness of our approach on real and synthetic data.

The next step is to apply our proposed algorithms to multi-modal mixture data, that is data with different but overlapping
sets of features. This task includes integrating sources of different types of data, for instance images and text for webpage knowledge discovery and data mining tasks. Another example comes from health data, where data on patients comes from a variety of mediums, including quantitative measurements, written descriptions and images. Such applications could use the weighted-loss sparse coding approach discussed in Section III-G. We believe our proposed approach could provide a good starting point for addressing the complicated task of knowledge transfer from varied, multimodal data sources.

Acknowledgments

This work has been supported by the National Science Foundation under Grant No. 0845951 and a Graduate Research Fellowship award for B.Q.

REFERENCES


