Learning with Previously Unseen Features

Yuan Shi
Computer Science Department
University of Southern California
yuanshi@usc.edu

Craig A. Knoblock
Information Sciences Institute
University of Southern California
knoblock@isi.edu

Abstract

We study the problem of improving a machine learning model by identifying and using features that are not in the training set. This is applicable to machine learning systems deployed in an open environment. For example, a prediction model built on a set of sensors may be improved when it has access to new and relevant sensors at test time. To effectively use new features, we propose a novel approach that learns a model over both the original and new features, with the goal of making the joint distribution of features and predicted labels similar to that in the training set. Our approach can naturally leverage labels associated with these new features when they are accessible. We present an efficient optimization algorithm for learning the model parameters and empirically evaluate the approach on several regression and classification tasks. Experimental results show that our approach can achieve on average 11.2% improvement over baselines.

1 Introduction

We consider a setting where a machine learning system can access features that are previously unseen in the labeled training set. This often happens when machine learning systems are deployed in an open environment.

For example, consider a weather station equipped with a set of sensors measuring temperature, wind speed, pressure, etc. Suppose the station has a machine learning model that predicts humidity from the sensor readings. The prediction model is built on a training set containing historical sensor readings and the corresponding humidity. Now, the station plugs in a new sensor that measures dew point. Since dew point is strongly correlated with humidity, leveraging this new sensor could potentially improve the model.

As a second example, consider a model that predicts a job applicant's quality. It is built on a training set where each sample contains features from the resume of a previous applicant and the quality level labeled by HR. Now, for future applicants, the model is allowed to access applicants' social media such as Facebook and Twitter. Exploring new features extracted from social media may improve the prediction quality since social media data often reflect an applicant's personality and perspectives, which may not be revealed in resumes.

In the above examples, the new features are likely to provide complementary information over the original features in the labeled training set. Identifying and leveraging such features can potentially improve the underlying machine learning models. This is an important step towards developing intelligent systems that can interact and learn from an open environment. The challenge is that there is no labeled samples associated with those additional features. Collecting new labels can be costly and often requires human input. It is desirable to develop an approach that can automatically *exploit* previously unseen features.

This problem can be viewed in the framework of domain adaptation or transfer learning [Pan and Yang, 2010; Daume III and Marcu, 2006; Quionero-Candela *et al.*, 2009], which aims to adapt machine learning models trained on a source domain to a related but different target domain. Our problem can be viewed as a special case of *heterogeneous domain adaptation*, where the feature space of the source domain is different from that of the target domain. Although being a special case, our problem setting actually poses unique challenge for existing domain adaptation approaches.

Previous work on heterogeneous domain adaptation mainly consists of two types of approaches. One type learns a transformation to map the features from one domain to the other [Dai *et al.*, 2008; Socher *et al.*, 2013; Zhou *et al.*, 2014] by leveraging sample-level correspondences across domains (e.g., an image and its tag). This type of approach can be applied to our setting naturally since each target-domain sample provides a correspondence between the original and new features. However, learning a good transformation may not be possible when the dependency between the original and new features is weak.[1] In our context, the new features are expected to contain complementary information not available in the original features, making such transformation difficult to obtain. A second type of approach maps the source and target domains into a *domain-invariant* feature space in which they distribute similarly (in terms of the features in that space), and

[1] Weak dependency implies that there is no strong correspondence between original and new features. In other words, knowing original features is not very helpful when predicting new features.

then learns a model in that space using labeled data [Kulis *et al.*, 2011; Wang and Mahadevan, 2011; Argyriou *et al.*, 2008; Duan *et al.*, 2012; Shi *et al.*, 2010; Harel and Mannor, 2010; Wei and Pal, 2011; Yeh *et al.*, 2014]. These approaches are not applicable to our setting because the original features already provide a domain-invariant feature space, which would completely ignore new features in the target domain.

Our problem can also be viewed as a special case of semi-supervised learning [Zhu, 2005], with additional features associated with unlabeled data. To the best of our knowledge, such problem setting has not been explored before because existing semi-supervised learning algorithms focus on using unlabeled data from the same feature space of labeled data.

To address the challenge, we propose an approach named **LUF** (Learning with previously Unseen Features) that builds a machine learning model over both the original and new features. Intuitively, the desirable model should predict target-domain labels *consistent* with the labels in the source domain. We express the consistency in the notion of the joint distributions over the original features and labels, and we enforce the two domains to have similar joint distributions. **LUF** minimizes k-nearest neighbor distances across domains in the joint space and can be solved by an efficient optimization algorithm. **LUF** can be simply extended to the case where there exist labeled samples with new features. We evaluate **LUF** extensively on benchmark regression and classification datasets and a sensor adaptation application in the weather domain. In most of the evaluation cases, **LUF** outperforms other baseline methods, with an average improvement of 11.2%.

Contributions. To summarize, we study a new machine learning setting involving features that are not available in the labeled training set. We then propose a novel approach as well as an efficient optimization algorithm. Our approach is applicable to both regression and classification tasks and can naturally leverage labels associated with these new features. Our empirical study highlights the efficacy of the proposed approach.

2 Approach

We are given N labeled samples $\{(\mathbf{x}_s, y_s)\}_{s=1}^{\mathsf{N}}$, where $\mathbf{x}_s \in \mathcal{R}^D$ is the input features, and y_s is the corresponding label. Additionally we are given M unlabeled samples $\{(\mathbf{x}_t, \mathbf{z}_t)\}_{t=1}^{\mathsf{M}}$, where $\mathbf{x}_t \in \mathcal{R}^D$ follows the same distribution of $\mathbf{x}_s$. $\mathbf{z}_t \in R^W$ are the additional features associated with $\mathbf{x}_t$. In the following, we refer to the set of labeled samples as the source domain, and the set of unlabeled samples as the target domain.

We are interested in learning a model for the target domain that maps $(\mathbf{x}, \mathbf{z})$ to y. In the following, we denote the model as $f_\theta(\mathbf{x}, \mathbf{z})$ where θ represents the model parameters, and the predicted label as $\hat{y} = f_\theta(\mathbf{x}, \mathbf{z})$.

For each source-domain sample $(\mathbf{x}_s, y_s)$, if we could estimate its $\hat{\mathbf{z}}_s$ reliably, we can simply train $f_\theta(\mathbf{x}, \mathbf{z})$ on the source domain. However, estimating $\mathbf{z}$ from $\mathbf{x}$ can be challenging when their dependency is weak, as observed in our empirical study. On the other hand, training on the target domain is very challenging as there are no labels available.

We tackle the challenge based on the following intuition: if our model predicts target-domain labels $\{\hat{y}_t\}$ well, then $\{\hat{y}_t\}$ should be *consistent* with the training labels. Such consistency can be expressed through some joint patterns between $\mathbf{x}$ and y, for instance, the joint distribution of $(\mathbf{x}, y)$. Ideally, our model f_θ would make the joint distribution similar across domains, which motivates us to seek f_θ such that $\{(\mathbf{x}_s, y_s)\}$ and $\{(\mathbf{x}_t, \hat{y}_t)\}$ are *mixed* as much as possible. When this happens, each source-domain sample $(\mathbf{x}_s, y_s)$ becomes close to its k-nearest neighbors in the target domain, and vise versa. Therefore, we propose the following objective function to minimize the cross-domain k-nearest neighbor distances in the joint space of $(\mathbf{x}, y)$

$$\min_\theta \sum_s \sum_{t \in \mathcal{N}_{\mathcal{T}}^k(s)} \mathsf{dist}[(\mathbf{x}_s, y_s), (\mathbf{x}_t, \hat{y}_t)] + \sum_t \sum_{s \in \mathcal{N}_{\mathcal{S}}^k(t)} \mathsf{dist}[(\mathbf{x}_t, \hat{y}_t), (\mathbf{x}_s, y_s)] + \lambda \|\theta\|_2^2 \quad (1)$$

where dist is the distance function defined on $(\mathbf{x}, y)$. $\mathcal{N}_{\mathcal{T}}^k(s)$ denotes the set of indices corresponding to $(\mathbf{x}_s, y_s)$'s k-nearest neighbors in the target domain, and $\mathcal{N}_{\mathcal{S}}^k(t)$ denotes the set of indices corresponding to $(\mathbf{x}_t, y_t)$'s k-nearest neighbors in the source domain, where nearest neighbors are determined based on dist. $\|\theta\|_2^2$ is the regularization term on θ with $\lambda \geq 0$ as the regularization parameter.

For simplicity, we set dist to be the following weighted distance[2]

$$\mathsf{dist}[(\mathbf{x}_s, y_s), (\mathbf{x}_t, \hat{y}_t)] = \|\mathbf{x}_s - \mathbf{x}_t\|_2^2 + \gamma \Delta(y_s, \hat{y}_t) \quad (2)$$

where $\Delta(y_s, \hat{y}_t)$ measures the distance between y_s and $\hat{y}_t$, and $\gamma > 0$ is a weight balancing the scale of $\mathbf{x}$ and y. Let $v_{st}^2 = \|\mathbf{x}_s - \mathbf{x}_t\|_2^2$, we can write (1) into

$$\min_\theta \sum_s \sum_{t \in \mathcal{N}_{\mathcal{T}}^k(s)} \left[v_{st}^2 + \gamma \Delta\left(y_s, f_\theta(\mathbf{x}_t, \mathbf{z}_t)\right)\right] + \sum_t \sum_{s \in \mathcal{N}_{\mathcal{S}}^k(t)} \left[v_{ts}^2 + \gamma \Delta(y_s, f_\theta(\mathbf{x}_t, \mathbf{z}_t))\right] + \lambda \|\theta\|_2^2 \quad (3)$$

For regression tasks, we simply set $\Delta(y_s, \hat{y}_t) = \|y_s - \hat{y}_t\|^2$. For classification tasks with C classes, we use probabilistic classification models (e.g. logistic regression [Bishop, 2006]) that can predict class probability for a given sample. In this case, $\hat{y}_t$ is a C-dimensional vector representing the probability in each class, and y_s is a C-dimensional binary vector. We can set $\Delta(y_s, \hat{y}_t) = 1 - \sum_{c=1}^{\mathsf{C}} y_s(c)\hat{y}_t(c)$, so that a small Δ corresponds to similar class assignment.

Note that $\mathcal{N}_{\mathcal{T}}^k(s)$ and $\mathcal{N}_{\mathcal{S}}^k(t)$ are dependent on θ, hence (3) is non-smooth and non-convex in θ. In the following, we present an efficient optimization algorithm for finding a local minimum of Eq. (3).

[2] In our implementation, features are first normalized into similar scales.

2.1 Alternating Optimization

For the ease of optimization, we introduce a set of auxiliary variables to "decouple" the dependency of $\mathcal{N}_{\mathcal{T}}^k(s)$ and $\mathcal{N}_{\mathcal{S}}^k(t)$ on θ. Let $\mathcal{V}_{\mathcal{T}}^k(s)$ index $(\mathbf{x}_s, y_s)$'s any (not necessarily the nearest) k neighbors in the target domain, and $\mathcal{V}_{\mathcal{S}}^k(t)$ index $(\mathbf{x}_t, y_t)$'s any k neighbors in the source domain. It is easy to see

$$\begin{aligned}&\sum_{t\in\mathcal{N}_{\mathcal{T}}^k(s)}\left[v_{st}^2+\Delta(y_s,f_\theta(\mathbf{x}_t,\mathbf{z}_t))\right]\\&=\min_{\mathcal{V}_{\mathcal{T}}^k(s)}\sum_{t\in\mathcal{V}_{\mathcal{T}}^k(s)}\left[v_{st}^2+\Delta(y_s,f_\theta(\mathbf{x}_t,\mathbf{z}_t))\right]\end{aligned}\tag{4}$$

and the same relationship holds for $\mathcal{V}_{\mathcal{S}}^k(t)$ and $\mathcal{N}_{\mathcal{S}}^k(t)$. Thus (3) is equivalent to

$$\begin{aligned}&\min_{\theta,\{\mathcal{V}_{\mathcal{T}}^k(s)\},\{\mathcal{V}_{\mathcal{S}}^k(t)\}}\sum_s\sum_{t\in\mathcal{V}_{\mathcal{T}}^k(s)}\left[v_{st}^2+\Delta(y_s,f_\theta(\mathbf{x}_t,\mathbf{z}_t))\right]\\&+\sum_t\sum_{s\in\mathcal{V}_{\mathcal{S}}^k(t)}\left[v_{ts}^2+\Delta(y_s,f_\theta(\mathbf{x}_t,\mathbf{z}_t))\right]+\lambda\|\theta\|_2^2\end{aligned}\tag{5}$$

(5) can be efficiently optimized via an alternating procedure. When θ is fixed, we update $\{\mathcal{V}_{\mathcal{T}}^k(s)\}$ and $\{\mathcal{V}_{\mathcal{S}}^k(t)\}$ based on nearest neighbor search. When $\{\mathcal{V}_{\mathcal{T}}^k(s)\}$ and $\{\mathcal{V}_{\mathcal{S}}^k(t)\}$ are fixed, we optimize θ by solving

$$\begin{aligned}&\min_\theta\sum_s\sum_{t\in\mathcal{V}_{\mathcal{T}}^k(s)}\Delta(y_s,f_\theta(\mathbf{x}_t,\mathbf{z}_t))\\&+\sum_t\sum_{s\in\mathcal{V}_{\mathcal{S}}^k(t)}\Delta(y_s,f_\theta(\mathbf{x}_t,\mathbf{z}_t))+\lambda\|\theta\|_2^2\end{aligned}\tag{6}$$

which can be easier to optimize than (3) when f_θ is smooth in θ. For regression tasks, if linear regression is used, θ can be solved analytically. For classification tasks, if logistic regression is used, a global optimum of θ can be solved using gradient descent techniques.

The above alternating procedure decreases (5) at each alternating step, and converges to a local minimum of (3). Empirically, the procedure converges quickly (usually within 50 iterations). The outline of the algorithm is described in Algorithm 1.

Algorithm 1 Optimization algorithm for **LUF**

Input: source-domain samples $\{(\mathbf{x}_s, y_s)\}_{s=1}^{\mathsf{N}}$ and target-domain samples $\{(\mathbf{x}_t, \mathbf{z}_t)\}_{t=1}^{\mathsf{M}}$, neighbor size k, weight parameter γ, and regularization parameter λ.
Initialize θ by solving Eq. (8)
for $iter = 1, 2, \cdots, T$ **do**
  for $s = 1, 2, \cdots, N$ **do**
    Fix θ, update $\mathcal{V}_{\mathcal{T}}^k(s)$
  for $t = 1, 2, \cdots, M$ **do**
    Fix θ, update $\mathcal{V}_{\mathcal{S}}^k(t)$
  Fix $\{\mathcal{V}_{\mathcal{T}}^k(s)\}, \{\mathcal{V}_{\mathcal{S}}^k(t)\}$, optimize θ by solving Eq. (6)
Output: a local optimal solution θ^*.

Initialization: The quality of the solution depends on how we initialize θ. Intuitively, if we could get a good estimate of $\hat{\mathbf{z}}_s$ based on each source-domain sample $\mathbf{x}_s$, we can initialize θ by solving

$$\min_\theta\sum_s\Delta(y_s,f_\theta(\mathbf{x}_s,\hat{\mathbf{z}}_s))\tag{7}$$

However, estimating $\hat{\mathbf{z}}_s$ can be very challenging when the dependency between $\mathbf{x}$ and $\mathbf{z}$ is weak. Nevertheless, we can estimate a candidate set of $\hat{\mathbf{z}}_s$ based on target-domain data as follows: for each $\mathbf{x}_s$, we find a set of its nearest neighbors in $\{\mathbf{x}_t\}$, and use the corresponding $\mathbf{z}_t$ to form a candidate set $\mathcal{Z}_s$. We then minimize the model error by optimizing both θ and $\{\hat{\mathbf{z}}_s\}$:

$$\min_\theta\sum_s\min_{\hat{\mathbf{z}}_s\in\mathcal{Z}_s}\Delta(y_s,f_\theta(\mathbf{x}_s,\hat{\mathbf{z}}_s))\tag{8}$$

where $\hat{\mathbf{z}}_s$ is allowed to be *any* element of $\mathcal{Z}_s$. (8) essentially relaxes the dependency between $\mathbf{x}$ and $\mathbf{z}$, and uses the optimal θ for the relaxed setting as an initialization. By setting $\{\mathcal{Z}_s\}$ to different sizes, we can get different initial solutions for θ.

Complexity analysis: In Algorithm 1, each iteration $iter$ involves N updates on $\mathcal{V}_{\mathcal{T}}^k(s)$ and M updates on $\mathcal{V}_{\mathcal{S}}^k(t)$. Each update on $\mathcal{V}_{\mathcal{T}}^k(s)$ takes $O(MD^2)$ and each update on $\mathcal{V}_{\mathcal{S}}^k(t)$ takes $O(ND^2)$, where D is the dimensionality of original features. Therefore the complexity of updating $\{\mathcal{V}_{\mathcal{T}}^k(s)\}$ and $\{\mathcal{V}_{\mathcal{S}}^k(t)\}$ at each iteration is $O(NMD^2)$. Additionally, each iteration $iter$ involves leaning a linear regression or logistic regression function, whose complexity is $O((D+W)^2(N+M))$ where W is the dimensionality of new features and we assume $N > D + W$. Further assuming $W = O(D)$ and $M = O(N)$, we have the overall complexity as $O(D^2N^2)$.

Hyper-parameter tuning and overfitting prevention: When there are no labels in the target domain, tuning hyper-parameters is often very difficult. We tune the neighborhood size k by reducing the error of a k-NN classifier or regressor on the source domain. To tune the weight γ and regularization parameter λ, we apply a *leave-one-out* cross validation strategy on the source domain to simulate our problem setting. Specifically, we consider each feature in the source domain as $\mathbf{z}$, and the rest of the features as $\mathbf{x}$. We then split source-domain samples into two disjoint sets: one set has only $\mathbf{x}$, and the other set has both $\mathbf{x}$ and $\mathbf{z}$. We treat the first set as a synthesized source domain and the second set as a synthesized target domain, and apply the proposed approach to pick the optimal γ and λ. To prevent overfitting, we adopt an early stopping strategy: train a model on $\{(\mathbf{x}_t, \hat{y}_t)\}$ and apply it to source-domain data. If the prediction error on the source domain is larger than a certain threshold, we stop the learning process. We also terminate the optimization when the objective function decreases very slowly, which not only saves computational time but also reduces overfitting.

Leveraging labels in the target domain: **LUF** can naturally leverage labels in the target domain when they are available. Suppose a subset of target-domain samples are labeled,

and let $\mathcal{L}$ denote their sample indicies. We can directly enhance **LUF** by adding a *supervised* term into the objective function

$$\min_{\theta} \sum_{s} \sum_{t \in \mathcal{N}_T^k(s)} \left[v_{st}^2 + \gamma \Delta \left(y_s, f_\theta(\mathbf{x}_t, \mathbf{z}_t) \right) \right]$$

$$+ \sum_{t} \sum_{s \in \mathcal{N}_S^k(t)} \left[v_{ts}^2 + \gamma \Delta (y_s, f_\theta(\mathbf{x}_t, \mathbf{z}_t)) \right] + \lambda \|\theta\|_2^2 \quad (9)$$

$$+ \mu \sum_{t \in \mathcal{L}} \Delta(y_t, f_\theta(\mathbf{x}_t, \mathbf{z}_t)) \quad (10)$$

where $\mu \geq 0$ is a new weight parameter that can be tuned by cross validation on the target domain.

3 Experiments

We evaluate **LUF** on several regression and classification datasets and an application on sensor adaptation for weather stations.[3]

We develop the following methods for comparison:

- **R**: kernel regression [Bishop, 2006] trained on the source domain. Polynomial kernel[4] $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i{}^{\mathrm{T}} \mathbf{x}_j + c)^d$ is used where c and d are kernel parameters.
- **R-Z$^{\mathbf{KR}}$**, **R-Z$^{\mathbf{NN}}$**. These are domain adaptation methods that explore the correspondences between $\mathbf{x}$ and $\mathbf{z}$ in the target domain. They first estimate $\hat{\mathbf{z}}$ for each source-domain sample, and then train a kernel regression model over $\{(\mathbf{x}_s, \hat{\mathbf{z}}_s)\}$ on the source domain. Two strategies are explored for estimating $\hat{\mathbf{z}}$:
  - Z$^{\mathrm{KR}}$: train a kernel regression on the target domain to map $\mathbf{x}_t$ to $\mathbf{z}_t$, and use that model to predict $\hat{\mathbf{z}}_s$ for source-domain sample $\mathbf{x}_s$.
  - Z$^{\mathrm{NN}}$: similar to Z$^{\mathrm{KR}}$ except using a k-NN regression model .
- **R-Z***: kernel regression directly trained on the target domain, using the target-domain labels ("cheating"). This method, though unrealistic in practice, provides an upper bound on the best possible performance. We apply ten-fold cross validation on the target domain and report the average error.
- **C**: logistic regression [Bishop, 2006] trained on the source domain. Since the dimensionality of the classification datasets is relatively high, we directly use input features and do not apply polynomial kernel.
- **C-Z$^{\mathbf{KR}}$**, **C-Z$^{\mathbf{NN}}$**, **C-Z***. These methods are the same as **R-Z$^{\mathbf{KR}}$**, **R-Z$^{\mathbf{NN}}$**, **R-Z*** respectively, except that kernel regression is switched to logistic regression.
- **LUF**: the proposed approach for learning with previously unseen features. For regression tasks, it trains a linear model based on the explicit feature mappings derived from polynomial kernel. For classification tasks, it trains a logistic regression based on input features.

The hyper-parameters of the above methods, including c and d in polynomial kernel, k in k-NN regression, the regularization parameter λ, are tuned on the source domain. For **R-Z*** and **C-Z***, their regularization parameters are tuned on the target domain.

3.1 Results on Regression Datasets

We experiment with four regression datasets

- Abalone,[5] for predicting the age of abalone, contains 4,177 samples with 8 features.
- Bank,[6] which predicts the fraction of bank customers that are turned away due to queuing, contains 8,192 samples with 8 features.
- CPU,[7] for CPU running time prediction, contains 8,192 samples with 12 features.
- House,[8] for housing price prediction, contains 20,640 samples with 9 features.

In order to select a subset of features to be "unseen", we look at features with high predictive power, which enables a machine learning model to improve prediction performance by leveraging these features.[9] Specifically, we do the following check for each feature: if removing it increases the prediction error by more than 10 percent, then it is selected as an unseen feature. We then sort these unseen features based on their impact on the prediction errors in descending order. Table 1 shows the results when the top set of unseen features are explored.

For each dataset, we conduct experiments in 10 random trials. In each trial, we randomly split the dataset into the source/target domain, each with half the number of samples. We report the average root mean square error (RMSE) and standard error on the target domain.[10] As a preprocessing step, we normalize each feature to [0,1].

Table 1 summarizes the prediction error of different methods, where the improvement(%) is computed based on the best performing baseline. In most cases, **LUF** outperforms the other baselines, with an average improvement of 12.1%.

[3] Our algorithms and datasets can be accessed from https://github.com/yuanshi/UnseenFeatures

[4] We use polynomial kernel over other nonlinear kernels because it has explicit form for the nonlinear features. For most datasets, we find polynomial kernel performs comparably to Gaussian RBF kernel. Our focus is more on leveraging new features than choosing the best kernels.

[5] https://archive.ics.uci.edumldatasetsAbalone

[6] http://www.cs.toronto.edu/∼delve/data/bank/desc.html

[7] http://www.cs.toronto.edu/∼delve/data/comp-activ/desc.html

[8] http://lib.stat.cmu.edu/datasets/

[9] Our experimental policy selects unseen features useful for prediction, which is designed to evaluate **LUF** 's efficacy. We also conduct experiments on unseen features that are uninformative or noisy. We observe that **LUF** is quite robust to such features (performance drops less than 2%). **LUF** often stops using these features because either there is a minor decrease in the objective function or early stopping is triggered.

[10] All target-domain data are used in the learning process. We also tested our methods on hold-out evaluation data from the target domain and observed consistent performance.

Table 1: Prediction error (RMSE) on regression datasets

Dataset	Unseen feat. ID	**R**	**R-Z^{KR}**	**R-Z^{NN}**	**R-Z***	**LUF**	Improv.(%)
Abalone	1	2.42 ± 0.08	2.33 ± 0.076	2.31 ± 0.064	2.28 ± 0.08	$\mathbf{2.28 \pm 0.01}$	1.3
Bank	1	0.12 ± 0.00	$\mathbf{0.11 \pm 0.01}$	$\mathbf{0.11 \pm 0.00}$		0.12 ± 0.00	-3.7
	1,2	0.15 ± 0.00	0.16 ± 0.01	0.15 ± 0.01	0.034 ± 0.00	$\mathbf{0.13 \pm 0.00}$	17.8
	1,2,3	0.15 ± 0.00	0.16 ± 0.00	0.16 ± 0.01		$\mathbf{0.14 \pm 0.00}$	4.6
CPU	1	8.34 ± 0.20	9.17 ± 0.21	6.81 ± 0.13		$\mathbf{5.35 \pm 0.60}$	21.44
	1,2	8.37 ± 0.19	9.15 ± 0.26	6.23 ± 0.18	3.63 ± 0.089	$\mathbf{5.79 \pm 0.56}$	7.1
	1,2,3	8.59 ± 0.18	8.74 ± 0.24	5.75 ± 0.44		$\mathbf{5.39 \pm 0.48}$	6.3
House	1	10.17 ± 1.12	10.14 ± 1.11	9.55 ± 1.18		$\mathbf{6.82 \pm 0.055}$	28.6
	1,2	10.77 ± 1.29	10.49 ± 1.03	8.52 ± 0.86	6.66 ± 0.25	$\mathbf{6.90 \pm 0.054}$	19.1
	1,2,3	12.60 ± 1.48	12.22 ± 1.35	9.60 ± 1.15		$\mathbf{7.83 \pm 0.088}$	18.4

Note: the results of House are in the scale of 10^5.

This demonstrates that **LUF** is able to exploit useful information in the additional features. In a few cases, **LUF** performs slightly worse than the best baseline, but the gap is much smaller than that in successful cases. This reveals that although **LUF** may overfit due to its unsupervised nature, it is quite robust.

It is interesting to study the improvement of **LUF** over baselines when the number of unseen features increases. With more unseen features, baselines tend to become worse but **LUF** has more useful features to leverage, which may lead to greater improvement, e.g. on Bank [1,2]. On the other hand, more unseen features makes the remaining ones less useful. **LUF** can suffer from this since the sample distances become less meaningful, e.g. on Bank [1,2,3] and CPU [1,2,3]. In practice, this may not be a severe problem because the features in the source domain are often reasonably good to begin with.

3.2 Results on Classification Datasets

We experiment with three classification datasets

- USPS, which recognizes handwriting digits from images, contains 9,298 samples from 10 classes [Hull, 1994].
- Books, which performs sentiment analysis on book reviews from Amazon, contains 4,000 samples from 2 classes [Blitzer *et al.*, 2006].
- Webcam, which recognizes objects in low-resolution images taken by web cameras, contains 795 samples from 10 classes [Kulis *et al.*, 2011].

For Books and Webcam, we split the data into the source/target domain, each with half the number of samples. For USPS, the source/target domain uses 1,500 samples each. For all datasets, we scale each feature to [0,1], and then use principal component analysis (PCA) to reduce the dimensionality to 100, which reduces computational cost and feature noise.

Similar to regression tasks, we check each feature as follows: if removing it increases the classification error by more than 10 percent, then it is selected as an unseen feature. Table 2 shows the results when the top 1 to 3 unseen features are explored. For each dataset, 10 random splits are used and averaged classification errors are reported. **C-Z^{KR}** and **C-Z^{NN}** perform better than **C** in all cases. This suggests that $\hat{\mathbf{z}}$ can be estimated fairly well for these datasets. **LUF** outperforms **C-Z^{KR}** and **C-Z^{NN}** in most cases, with average improvement 3.6%. When the number of unseen features increases, **LUF** shows greater improvement over baselines as estimating $\mathbf{z}$ becomes harder.

3.3 Sensor Adaptation for Weather Stations

We apply our proposed approach to an application for sensor adaptation. In particular, we investigate how sensors at weather stations can be adapted when sensor failure happens. In a real-world environment, sensor failure often occurs at weather stations [Dereszynski and Dietterich, 2011] and can cause problems for system modules relying on the failed sensor. We aim to develop a robust system that automatically reconstructs the missing signal from the working ones.

Missing signal reconstruction can be cast as a regression task, and a regression model can be built based on historical sensor readings. The reconstruction quality depends a lot on the correlations between the missing signal and the remaining ones. For example, reconstructing temperature from dew point and humidity can be fairly accurate but reconstructing wind speed from dew point and humidity is very hard. Our system handles this challenge by allowing a weather station to access the sensors at a nearby station when a sensor failure is detected.

We conduct experiments with data from Weather Underground,[11] which contains sensor data from a large number of personal weather stations worldwide. Each station consists of 5-10 sensors and produces a sample (combined readings form all sensors) at a fixed time interval (e.g. every 10 minutes). We examine three stations in San Francisco, San Jose, and New York, respectively, as well as their nearby stations. For a given station, we use 3,000 samples randomly drawn from Jan. 2015 to Aug. 2015 as the source domain, and 3,000 random samples from Jan. 2016 to Aug. 2016 as the target domain.

Table 3 summarizes the results. The reported cases correspond to signals that are difficult to reconstruct using a weather station's own sensors.[12] **LUF** achieves positive im-

[11] http://www.wunderground.com

[12] Note that precipitation data are not available for SF-A and SF-B, and wind speed and wind gust data are not available for NY-A and NY-B.

Table 2: Classification error rate (%) on classification datasets

Dataset	Unseen feat. ID	**C**	**C-Z^{KR}**	**C-Z^{NN}**	**C-Z***	**LUF**	Improv.(%)
USPS	1	10.9 ± 0.08	9.2 ± 0.2	8.1 ± 0.1		$\mathbf{8.0 \pm 0.2}$	1.1
	1,2	14.9 ± 0.1	13.4 ± 0.2	8.5 ± 0.2	7.8 ± 0.08	$\mathbf{8.2 \pm 0.2}$	3.5
	1,2,3	19.3 ± 0.1	16.3 ± 0.2	9.3 ± 0.2		$\mathbf{8.7 \pm 0.2}$	7.1
Books	1	24.9 ± 0.4	24.2 ± 0.3	$\mathbf{23.4 \pm 0.3}$		23.8 ± 0.3	-1.7
	1,2	26.5 ± 0.3	25.6 ± 0.4	24.9 ± 0.3	22.6 ± 0.2	$\mathbf{24.1 \pm 0.3}$	3.6
	1,2,3	28.7 ± 0.3	27.8 ± 0.3	25.9 ± 0.3		$\mathbf{24.6 \pm 0.2}$	5.0
Webcam	1	36.7 ± 0.2	35.4 ± 0.3	35.4 ± 0.3		$\mathbf{34.8 \pm 0.4}$	1.7
	1,2	38.5 ± 0.3	37.5 ± 0.3	37.4 ± 0.3	33.2 ± 0.2	$\mathbf{35.3 \pm 0.3}$	6.3
	1,2,3	40.5 ± 0.4	39.1 ± 0.4	38.3 ± 0.3		$\mathbf{36.2 \pm 0.3}$	5.4

Table 3: Prediction error (RMSE) on weather data

Stations	Missing Signal	**R**	**R-Z^{KR}**	**R-Z^{NN}**	**R-Z***	**LUF**	Imp.(%)
SF-A : SF-B	wind speed	5.80 ± 0.024	5.94 ± 0.051	$\mathbf{5.76 \pm 0.030}$	5.13 ± 0.019	5.93 ± 0.032	-2.9
	wind gust	10.52 ± 0.059	10.76 ± 0.20	10.45 ± 0.18	7.94 ± 0.045	$\mathbf{9.70 \pm 0.068}$	7.2
	pressure	4.53 ± 0.23	4.93 ± 0.25	4.60 ± 0.35	0.32 ± 0.029	$\mathbf{1.52 \pm 0.25}$	66.4
SJ-A : SJ-B	wind speed	3.76 ± 0.082	3.94 ± 0.15	3.78 ± 0.066	1.11 ± 0.018	$\mathbf{3.74 \pm 0.053}$	0.51
	wind gust	4.41 ± 0.083	4.38 ± 0.092	4.37 ± 0.10	1.10 ± 0.013	$\mathbf{4.16 \pm 0.045}$	4.8
	pressure	4.01 ± 0.021	4.03 ± 0.10	3.86 ± 0.079	0.15 ± 0.012	$\mathbf{1.95 \pm 0.068}$	49.5
	precipitation	0.57 ± 0.034	0.56 ± 0.082	0.61 ± 0.12	0.07 ± 0.014	$\mathbf{0.46 \pm 0.062}$	17.9
NY-A : NY-B	pressure	11.40 ± 0.14	11.43 ± 1.17	10.34 ± 0.019	0.43 ± 0.022	$\mathbf{9.74 \pm 0.21}$	5.8
	precipitation	3.17 ± 0.092	3.96 ± 0.14	4.19 ± 0.26	0.68 ± 0.032	$\mathbf{2.82 \pm 0.15}$	11.5

Note: the results are reported in the following units: mph (wind speed), mph (wind gust), in (pressure), in (precipitation). The listed weather stations correspond to the following station IDs in Weather Underground: SF-A (KCASANFR142), SF-B (KCASANFR114), SJ-A (KCASANJO121), SJ-B (KCASANJO139), NY-A (KNYNEWYO139), NY-B (KNYNEWYO132).

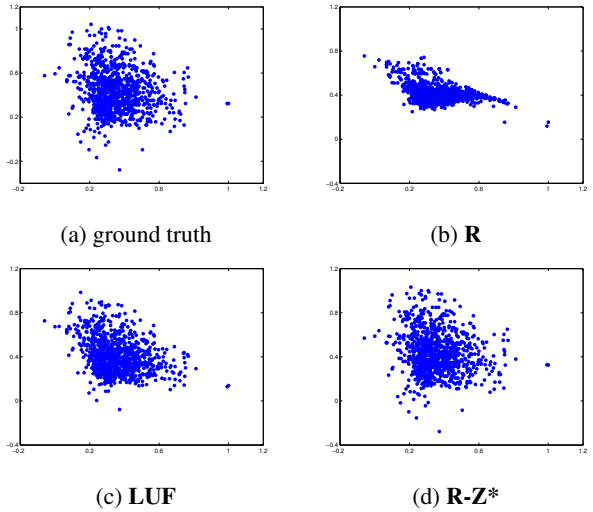
provement in 8 out of 9 cases, with most significant improvement (66.7%) on reconstructing pressure for SF-A. Note that the amount of improvement is consistent with the gap between **R-Z*** and other baselines. When the gap is large, **LUF** leverages better features and outperforms other baselines with a large margin. When the gap is small (wind speed at SF-A), **LUF** performs worse than **R** due to overfitting, but with a fairly small performance drop.

Figure 1 visualizes the joint distributions over original features and predicted labels, where we choose wind speed as a representative feature and pressure as the label, on station SF-A. As can be observed, **R** generates a significantly different joint distribution compared to the actual one, while **LUF** produces a much closer distribution. **R-Z*** performs even better but relies on the labels from the target domain.

4 Related Work

Our learning setting can be viewed as a special case of heterogeneous domain adaptation [Pan and Yang, 2010], which adapts a learning model across different feature spaces. The work of [Zhao and Hoi, 2010] and [Hou and Zhou, 2016] also consider new features in the target domain. However, their approaches require labels from the target domain, which does not work for our setting. A similar nearest-neighbor-based objective function is used in [Kulis *et al.*, 2011]. However, their neighbors are fixed but ours are dynamically updated. More importantly, our approach computes distances over both features and labels.

Another way to address our setting is to treat it as a missing data problem: the additional features available in the target domain are completely missing in the source domain. The baselines in our empirical study are also approaches for missing value imputation [Grzymala-Busse and Hu, 2000; Lakshminarayan *et al.*, 1996; Batista *et al.*, 2002]. However, recovering these features in the source domain is very chal-



(a) ground truth (b) **R**

(c) **LUF** (d) **R-Z***

x-axis represents wind speed (feature), and y-axis represents pressure (label) given by different approaches. Ground truth corresponds to the actual pressure. Values are in normalized scales.

Figure 1: Visualization of wind speed and predicted pressure on weather station SF-A

lenging in nature.

Our setting is related to zero-shot learning [Larochelle *et al.*, 2008; Farhadi *et al.*, 2009] which explores unseen classes at test time, as well as to privileged information-based learning [Vapnik and Vashist, 2009] which leverages additional information only available in training. Although related, our setting is very different from theirs.

5 Conclusion

We presented a novel machine learning approach that leverages previously unseen features in the training set. The approach is applicable to both classification and regression tasks. Supported by our empirical results, the approach can be used to improve a learning model when new features are accessible. Our future work includes theoretical analysis and more real-world applications. We also plan to develop algorithms that can automatically determine when to explore new features and which features to select from a large pool of features in an open environment.

Acknowledgements

This material is based upon work supported by the United States Air Force and the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-16-C-0045. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force and DARPA.

References

[Argyriou *et al.*, 2008] Andreas Argyriou, Andreas Maurer, and Massimiliano Pontil. An algorithm for transfer learning in a heterogeneous environment. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 71–85. Springer, 2008.

[Batista *et al.*, 2002] Gustavo EAPA Batista, Maria Carolina Monard, et al. A study of k-nearest neighbour as an imputation method. *HIS*, 87(251-260):48, 2002.

[Bishop, 2006] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

[Blitzer *et al.*, 2006] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.

[Dai *et al.*, 2008] Wenyuan Dai, Yuqiang Chen, Gui-Rong Xue, Qiang Yang, and Yong Yu. Translated learning: Transfer learning across different feature spaces. In *Advances in neural information processing systems*, pages 353–360, 2008.

[Daume III and Marcu, 2006] Hal Daume III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, 2006.

[Dereszynski and Dietterich, 2011] Ethan W Dereszynski and Thomas G Dietterich. Spatiotemporal models for data-anomaly detection in dynamic environmental monitoring campaigns. *ACM Transactions on Sensor Networks (TOSN)*, 8(1):3, 2011.

[Duan *et al.*, 2012] Lixin Duan, Dong Xu, and Ivor W Tsang. Learning with augmented features for heterogeneous domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 711–718, 2012.

[Farhadi *et al.*, 2009] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1778–1785. IEEE, 2009.

[Grzymala-Busse and Hu, 2000] Jerzy W Grzymala-Busse and Ming Hu. A comparison of several approaches to missing attribute values in data mining. In *International Conference on Rough Sets and Current Trends in Computing*, pages 378–385. Springer, 2000.

[Harel and Mannor, 2010] Maayan Harel and Shie Mannor. Learning from multiple outlooks. 2010.

[Hou and Zhou, 2016] Chenping Hou and Zhi-Hua Zhou. One-pass learning with incremental and decremental features. *arXiv preprint arXiv:1605.09082*, 2016.

[Hull, 1994] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

[Kulis *et al.*, 2011] 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.

[Lakshminarayan *et al.*, 1996] Kamakshi Lakshminarayan, Steven A Harp, Robert P Goldman, Tariq Samad, et al. Imputation of missing data using machine learning techniques. In *KDD*, pages 140–145, 1996.

[Larochelle *et al.*, 2008] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *AAAI*, volume 1, page 3, 2008.

[Pan and Yang, 2010] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[Quionero-Candela *et al.*, 2009] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

[Shi *et al.*, 2010] Xiaoxiao Shi, Qi Liu, Wei Fan, S Yu Philip, and Ruixin Zhu. Transfer learning on heterogenous feature spaces via spectral transformation. In *2010 IEEE international conference on data mining*, pages 1049–1054. IEEE, 2010.

[Socher *et al.*, 2013] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot

learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013.

[Vapnik and Vashist, 2009] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5):544–557, 2009.

[Wang and Mahadevan, 2011] Chang Wang and Sridhar Mahadevan. Heterogeneous domain adaptation using manifold alignment. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1541, 2011.

[Wei and Pal, 2011] Bin Wei and Christopher J Pal. Heterogeneous transfer learning with rbms. In *AAAI*, 2011.

[Yeh *et al.*, 2014] Yi-Ren Yeh, Chun-Hao Huang, and Yu-Chiang Frank Wang. Heterogeneous domain adaptation and classification by exploiting the correlation subspace. *IEEE Transactions on Image Processing*, 23(5):2009–2018, 2014.

[Zhao and Hoi, 2010] Peilin Zhao and Steven C Hoi. Otl: A framework of online transfer learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1231–1238, 2010.

[Zhou *et al.*, 2014] Joey Tianyi Zhou, Sinno Jialin Pan, Ivor W Tsang, and Yan Yan. Hybrid heterogeneous transfer learning through deep learning. In *AAAI*, pages 2213–2220, 2014.

[Zhu, 2005] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.