Robust Transduction for Unsupervised Adaptation

Abstract

Abstract.

1. Introduction

Intro

2. Related Work

Related work

3. Method

3.1. Problem Definition

We are trying to learn the relationships between two domains of data. In our setting, one of the domains is fully supervised and has both data points as well as class labels $\{\hat{\mathbf{x}}_\mathbf{i}, \hat{y}_i\}_{i\in[N^s]}$ such that $\mathbf{x_i}$ is the data point i and y_i is the corresponding label. Whereas, the other domain is unsupervised and only have data points as $\{\mathbf{x_i}\}_{i\in[N^u]}$. We further assume that there is a feature function $\Phi(\cdot)$ which is used in both of the domains.

We formulate our problem as a metric learning problem and consider an an asymmetric similarity metric;

$$s(\mathbf{x_i}, \hat{\mathbf{x}}_\mathbf{j}) = \Phi(\mathbf{x_i})^\intercal \mathbf{W} \Phi(\hat{\mathbf{x}}_\mathbf{j}) \tag{1}$$

such that it is high if two points from supervised and unsupervised domains are similar to each other.

We model our learning setting in a fully transductive setting; in other words, the main purpose of the method is recovering the labels y_i for each unsupervised example $\mathbf{x_i}$. We consider the following objective function in order to compute y_i as well as the similarity metric $\mathbf{W}$.

$$\begin{aligned} \min_{\mathbf{W}, y_i} \sum_{i\in[N^s]} \quad & [s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x}_{\mathbf{i}^-}) - s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x}_{\mathbf{i}^+}) + \alpha]_+ \\ s.t. \quad & i^+ = \arg\max_{j|y_j=\hat{y}_i} s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x_j}) \\ & i^- = \arg\max_{j|y_j\neq\hat{y}_i} s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x_j}) \end{aligned} \tag{2}$$

We solve this optimization problem via alternating minimization through iterating over solving for unsupervised labels y_i and the similarity metric $\mathbf{W}$. We explain these two steps the following sections.

3.2. Labeling Unsupervised Points

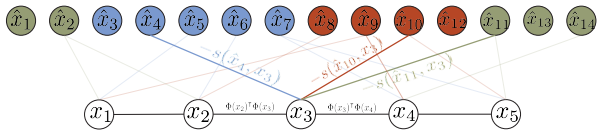

Figure 1. **Visualization of the Label Propogation.** Consider the unsupervised point x_3, the resulting label would be: $\arg\min_{y_3} - s(\hat{\mathbf{x}}_4, \mathbf{x}_3)\mathbb{1}(\mathbf{y}_3 = 1) - s(\hat{\mathbf{x}}_{10}, \mathbf{x}_3)\mathbb{1}(y_3 = 2) - s(\hat{\mathbf{x}}_{11}, \mathbf{x}_3)\mathbb{1}(y_3 = 0) + \Phi(\mathbf{x}_2)^\intercal\phi(\mathbf{x}_3)\mathbb{1}(y_3 \neq y_2) + \Phi(\mathbf{x}_3)^\intercal\phi(\mathbf{x}_4)\mathbb{1}(y_3 \neq y_4)$ assuming green is 0, blue is 1 and red is 2

In order to label the unsupervised data-points, we use nearest-neighbor(NN) rule as computing the NN supervised datapoint using the learned metric $s(\cdot, \cdot)$. Moreover, the NN rule will be accurate only if the metric is accurate enough. Since our algorithm is iterative alternating optimization, we still need rather accurate labeling even in the case of sub-optimal similarity metric. Hence, we introduce an additional robustness measure through label propogation. We first explain our NN-rule, then extend it to label propagation.

Given a similarity metric $s(\cdot, \cdot)$, the nearest neighbor rule is;

$$(y_i)^{pred} = \hat{y}_{\arg\max_j s(\mathbf{x_i}, \hat{\mathbf{x}}_\mathbf{j})} \tag{3}$$

The major issue for using the nearest neighbor rule is computationally efficiency. However, our implementation method and choice of parameters makes the NN rule tractable. We explain how we use stochastic gradient descent, select the batch size and efficiently implement the NN rule using OpenBLAS in Section**??**.

In order to make our transduction stage robust, we use label propagation. Main idea behind our label propagation approach is enforcing consistency of the labels over unsupervised data points. In order to enforce this consistency, we create a k-NN graph over the unsupervised data points such that neighbor set $\mathcal{N}(\mathbf{x_i})$ for $\mathbf{x}_i$ is the k-unsupervised data

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

point nearest to $\mathbf{x_i}$ using the l_2 distance in feature space. After the k-NN graph is created, we solve the following optimization problem for labeling unsupervised data points;

$$\begin{aligned}\arg\min_{y_i} \sum_{i\in N^u} -\max_{\hat{y}_j=y_i} s(\hat{\mathbf{x}}_\mathbf{j}, \mathbf{x_i}) \\ +\lambda \sum_{i\in N^u}\sum_{j\in\mathcal{N}(\mathbf{x_i})} \mathbf{x_i}^T\mathbf{x_j}\mathbb{1}(y_i\neq y_j)\end{aligned} \tag{4}$$

This problem is sub-modular and can easily be optimized through many methods like α-β swapping, quadratic pseudo-boolean optimization (QPBO), linear programming through roof-duality etc. We use α-β swapping algorithm from (**?**). In order to further explain the label propagation, we visualize a sample example with $k = 2$ and 3-class classification problem in Figure 1. Since it is rather out-of-scope of this paper, we explain the details of the α-β swapping algorithm to the appendix.

3.3. Learning Similarity Metric

Given the predicted labels y_i for unsupervised data points $\mathbf{x_i}$, we need to learn the asymmetric metric in order to optimize the loss function defined in (2). We extend the LMNN(Large Margin Nearest Neighbour)(**?**) construction to the multi-domain case in order to define our metric learning objective.

Main intuition behind our formulation is searching for a metric which will label the supervised data points correctly using the unsupervised data points and their predicted labels. Since at this stage we already have a predicted label for each unsupervised data points, we can estimate a label for the supervised data points using these predicted labels. We also have ground truth labels for the supervised data points and we can look for a metric which will maximize the accuracy. In other words, the over all metric learning is combination of;

- predicting $\hat{y}_j^{pred}$ using $\mathbf{x_i}, \hat{\mathbf{x}}_\mathbf{j}, \hat{y}_j$
- learning $s(\cdot,\cdot)$ by penalizing $(\hat{y}_j)^{pred} \neq \hat{y}_j$

Fortunately, this can be jointly solved by minimizing the triplet loss as we define through nearest same-class and different-class examples of each supervised datapoint from unsupervised data points. In other words, we find the nearest positive and negative examples through;

$$\begin{aligned} i^+ &= \arg\max_{j|y_j=\hat{y}_i} s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x_j}) \\ i^- &= \arg\max_{j|y_j\neq\hat{y}_i} s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x_j}) \end{aligned} \tag{5}$$

Then, we construct the triplet loss function with regularizer as;

$$\min_{\mathbf{W},y_i} \sum_{i\in[N^s]} [s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x}_{\mathbf{i}^-}) - s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x}_{\mathbf{i}^+}) + \alpha]_+ + r(\mathbf{W}) \tag{6}$$

Algorithm 1 Robust Transduction with Metric Learning

Input: unsupervised $\mathbf{x}_i$, supervised $\hat{\mathbf{x}}_i$, y_i, batch size B
repeat
  Sample $(\mathbf{x}_1^b, \ldots \mathbf{x^b}_B)$, $(\hat{\mathbf{x}}_1^b, \ldots \hat{\mathbf{x}}_B^b)$, $(\hat{y}_1^b, \ldots \hat{x}_B^b)$
  Solve (4) using $(\mathbf{x}_1^b, \ldots \mathbf{x^b}_B)$ and $(\hat{\mathbf{x}}_1, \ldots \hat{\mathbf{x}}_{N^s})$
  for $i = 1$ **to** B **do**
    if $\hat{y}_i$ **in** $y_1 \ldots y_B$ **then**
      Compute (i^+, i^-) via $(\mathbf{x}_1^b, \ldots \mathbf{x^b}_B)$, $(\hat{\mathbf{x}}_1^b, \ldots \hat{\mathbf{x}}_B^b)$
      Update $\frac{\partial loss(y_i,\mathbf{W})}{\partial \mathbf{\Theta}}$ and $\frac{\partial loss(y_i,\mathbf{W})}{\partial \mathbf{W}}$ using (7,8)
    end if
  end for
  $\mathbf{W} \leftarrow \mathbf{W} + \alpha\frac{\partial loss(y_i,\mathbf{W})}{\partial \mathbf{W}}$
  $\mathbf{\Theta} \leftarrow \mathbf{\Theta} + \alpha\frac{\partial loss(y_i,\mathbf{W})}{\partial \mathbf{\Theta}}$
until CONVERGENCE or MAX_ITER

which is convex in terms of the $\mathbf{W}$ if the regularizer is convex; and we optimize it by using Stochastic gradient descent through the subgradient $\frac{\partial loss(y_i,\mathbf{W})}{\partial \mathbf{W}} =$

$$\sum_{i\in[N^s]} \mathbb{1}(s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x}_{\mathbf{i}^-}) - s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x}_{\mathbf{i}^+}) > \alpha)\,(\hat{\mathbf{x}}_\mathbf{i}\mathbf{x}_{\mathbf{i}^-}{}^\intercal - \hat{\mathbf{x}}_\mathbf{i}\mathbf{x}_{\mathbf{i}^+}{}^\intercal) + \frac{\partial r(\mathbf{W})}{\partial \mathbf{W}} \tag{7}$$

As a regularizer we are using the Frobenius norm of the similarity matrix as $r(\mathbf{W}) = \frac{1}{2}\|\mathbf{W}\|_F^2$. We explain the details of this optimization routine and how we implement in the Section **??**.

3.4. Learning Features

In Section 3.2 and 3.3, we developed our transductive labeling method with propogation and the way to learn the metric $s(\cdot,\cdot)$ through learning $\mathbf{W}$. During this formulation, we used a pre-defined feature function Φ. However, the current trends in machine learning suggests that learning this feature function Φ from the data directly is a promising direction especially for visual data points. Hence, we consider the case $\Phi_{\mathbf{\Theta}}$ is a parametrized feature function with parameter set $\mathbf{\Theta}$. A typical example is CNNs(convolutional neural networks) with $\mathbf{\Theta}$ as concatenation of weights and biases. We update the feature weights as part of the metric learning. Hence, the gradient update for $\mathbf{\Theta}$ becomes; $\frac{\partial loss(y_i,\mathbf{W})}{\partial \mathbf{\Theta}} =$

$$\sum_{i\in[N^s]} \mathbb{1}(s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x}_{\mathbf{i}^-}) - s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x}_{\mathbf{i}^+}))\left(\frac{\partial s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x}_{\mathbf{i}^-})}{\partial \mathbf{\Theta}} - \frac{\partial s(\hat{\mathbf{x}}_\mathbf{i}, \mathbf{x}_{\mathbf{i}^+}))}{\partial \mathbf{\Theta}}\right) \tag{8}$$

3.5. Implementation Details

3.6. Weakly-Supervised Case

4. Convergence Analysis

It is monotonically decreasing after the each iteration

5. Experimental Results

6. Conclusion

Acknowledgements

a