OPTIMAL TRANSPORT AND DOMAIN ADAPTATION

A comparison of subspace alignment algorithm and Sinkhorn algorithm

Ezukwoke K.I[12]

[1, 2] Department of Computer Science
Machine Learning and Data Mining
`{ifeanyi.ezukwoke}@etu.univ-st-etienne.fr`
University Jean Monnet, Saint-Etienne, France

Abstract

We analyse two algorithms for approximating the optimal transport distance for domain adaption problems. We compare precisely the result of subspace alignment algorithm and Sinkhorm's algorithm (from python optimal transport (POT) library). We experiment using the Office/Caltech dataset and perform One nearest neighbour classification (1-**NN**). Finally we compare the performance of a our version of Sinkhorm's algorithm to POT version.

Keywords
Transfer Learning, Domain adaptation, Optimal transport, Subspace alignment algorithm, Sinkhorm's algorithm.

1 INTRODUCTION

Transfer learning is an aspect of machine learning involving the transfer of knowledge acquired by a classifier to similar or different context by mapping the features of the orginal domain space to that of the target space. In classification problem, the objective is usually to learn a subset of the original data; using the knowledge of this subset, the algorithm can essentially predict the remaining subset (usually referred to as test data) from the same domain. In contrast to supervised learning, the objective in transfer learning is to learn on a source domain or probability distribution (S); using the knowledge of the source domain, the algorithm tries to learn a target domain (or probability distribution). The term **Domain adaptation** is used to refer to the optimization process of mapping a **source domain** (probability distribution) to a target domain (probability distribution). We refer to this different but related joint distributions as **domains** [1].

We represent the important notations for subsequence alignment in this paper using the hitherto symbols. The source domain data is denoted using S, the source domain labels L_s, target domain data T, target domain labels L_T. We represent notations for entropy regularized optimal transport, where a and b denote uniform input vectors, M $(d \times d)$ is cost matrix of mapping a to b using the joint probability P, d_M^λ is the optimal entropic transportation distance and λ is the regularization parameter.

2 OPTIMAL TRANSPORT AND DOMAIN ADAPTATION

Optimal transport is a linear programming problem where we hope to find the minimum distance between two different probability measures. Distances such as earth mover's distance [2, 3] and Monge-Kantorovich or Wasserstein distance [4] play an important role in trying to solve the optimal transport problem.
Our interest is not in the theoretical proofs of the models but infact in the experimental comparisons of the both algorithms (subspace alignment and entropic regularization).

3 SUBSPACE ALIGNEMENT FOR DOMAIN ADAPTATION

The objective of this methods is to generate a source and target subspace which consist of learning the transformation matrix (or alignment matrix) M. This matrix is responsible for mapping the source subspace to the target space [1]. M is then learned by minimizing the **Bregman matrix divergence** given by the equation

$$F(M) = \|X_s M - X_T\|_F^2 \tag{1}$$
$$M* = \arg\min_M (F(M)) \tag{2}$$

Where $\|\|_F^2$ is the Frobenius norm. Since the Frobenius norm is invariant to orthonormal operations, we can re-write equation 2 as

$$\|X_s' X_s M - X_s' X_T\|_F^2 = \|M - X_s' X_T\|_F^2 \tag{3}$$

From where we can conclude that the solution of the optimal M is

$$M^* = X_s' X_T \tag{4}$$

and the alignment coordinate is given by

$$X_a = X_s X_s' X_T \tag{5}$$

The algorithm for the subspace alignment domain adaptation follows

Algorithm 1: Subspace alignment DA algorithm in lower $d-$dimensional space

```
Input  : $S$ ($Source\ data$),
         $T$($Target\ data$),
         $L_s$($Source\ labels$)
Output : $L_T$($target\ labels$)
1 begin
2     $X_s \leftarrow PCA(S, d)$;
3     $X_T \leftarrow PCA(T, d)$;
4     $X_a \leftarrow X_s X_s^T X_T$;
5     $T_T \leftarrow T X_T$;
6     $T_T \leftarrow$
        $1-NN-Classifier(S_a, T_T, L_s)$;
7 end
```

4 ENTROPY REGULARIZED OPTIMAL TRANSPORT

The method consists in adding an entropic term to the minimization problem, to make λ-convex one. But, instead of using a direct gradient descent (which can anyway only be easily implemented on the whole space we minimize on $U(a, b)$), Sinkhorn's algorithm finds a minimizer of the entropic regularized optimal transport (ϵ) iteratively.
We write the constraint minimization problem of optimal transport as

$$\begin{cases} \arg\min_{P \in \mathbb{R}_+^{n \times n}} & < P, M > \\ \text{subject to} & P1 = a, P^T 1 = b \end{cases} \tag{6}$$

By addin the entropic term we have that

$$\begin{cases} \arg\min_{P \in \mathbb{R}_+^{n \times n}} & < P, M > -\frac{1}{\lambda} E(P) \\ \text{subject to} & P1 = a, P^T 1 = b \end{cases} \tag{7}$$

Where the entropy, $E(P) = \sum_{i,j=1}^{N} -P_{ij}\log(P_{ij})$. By introducing Lagragian multiple and solving the dual problem we can derive the following.

$$u^{k+1} = a/Kv^k \tag{8}$$
$$v^{k+1} = b/K^T u^k \tag{9}$$

Where K is a guassian kernel given by $K = \exp(-\frac{M}{\lambda})$.

4.1 SINKHORN'S ALGORITHM

We solve the entropy optimization problem using the fixed point iteration method proposed by [5]. The algorithm is given below.

Algorithm 2: Computation of $d_M^\lambda(a,b)$ using Sinkhorn-Knopp's fixed point iteration

```
Input  : $\mathbf{M}, \lambda, a, b$
Output : $d_M^\lambda(a,b)$
1  begin
2      $I \leftarrow (a > 0)$;
3      $a \leftarrow a(I)$;
4      $M \leftarrow M(I,:)$;
5      $K \leftarrow exp(-\lambda * M)$;
6      $x \leftarrow ones(length(a), size(b,2))/length(r)$;
7      while $x$ changes do
8          $x \leftarrow diag(1./r) * K * (b * (K^T * (1/x)))$;
9          $u \leftarrow 1/x$;
10         $v \leftarrow b * (K^T * u)$;
11         $d_M^\lambda = \sum_{i=1}^{N}(u_i * (K * M) * v)$
12     end
13 end
```

5 EXPERIMENT

5.1 DATASET

We experiment using the Office/Caltech dataset used by computer vision and object recognition researchers. It contains 2533 data points, 800 Speeded-Up Robust Features (SURF), 4096 Deep Convolutional Activation Features (DeCAF), 4 subdomains (Caltech10 (**C**), Amazon (**A**), Webcam (**W**), dslr (**D**)). We normalize the dataset between $[0,1]$ using $MinMaxScaler$ from sklearn.

5.2 PERFORMANCE METRIC

Analysis the result of our algorithm requires the use of a classification algorithm. We use One Nearest Neighbour for testing the domain adaptation methods used in this paper. We compare the performance of this different adaptation models using Accuracy.
Accuracy is the measure of how often our classifier makes correct prediction. For a multi-classification problem, and it is given by

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{10}$$

Where **TP**: True Positives, **TN**: True Negatives, **FP**: False Positives **FP**: False Negatives.

5.3 Classification results

We compare the classification accuracy and running time for subspace alignment algorithm and entropy regularized optimal transport algorithm.

5.3.1 Dimension $d = 20$ and Regularization $\lambda = 5$

We observe from table 1 and 2 the accuracy and running time of both algorithms. We notice that **Subspace alignment domain adaptation (SADA)** has state-of-the-art performance, outperforming **Sinkhorn's algorithm** in terms of accuracy. In almost of the domain adaptation we perform, we find that although entropy regularized domain adaptation algorithm (Sinkhorn's algorithm) lags behind SADA, it still performs efficiently close to SADA in accuracy measurement.
In terms of running time, we observe that despite the state-of-the-art performance

of SADA, it takes too much time to compute the transportation distance between two domains. ERDA is the fastest in terms of time except in transfer of *caltech*10 to *Amazon* domain.

5.3.2 Dimension $d = 100$ and Regularization $\lambda = 10$

By increasing the dimension of the dataset, we only observe changes in SADA and by increasing the regularization value, we only observe changes in performance of ERDA. We observe that increasing the dimension of the data increases the accuracy for ($W \rightarrow D$ and $C \rightarrow A$ domain adaptations) while we observe a decrease in all domain adaptations of ERDA.

Accuarcy (%) (%)							
method	W→D	C→A	D→C	C→W	A→D	W→A	C→D
SADA	**78**	**43**	**33**	**26**	**36**	**34**	**34**
ERDA	59	35	24	23	34	25	32

Table 1: Accuracy comparison for Subspace alignment (SA) and Entropy regulerized(ER) DA. Dimension $d = 20$, Regularization parameter $\lambda = 5$

Time (secs) (%)							
method	W→D	C→A	D→C	C→W	A→D	W→A	C→D
SADA	0.70	**0.81**	0.92	0.70	0.72	0.72	0.72
ERDA	**0.12**	3.0	**0.35**	**0.58**	**0.26**	**0.55**	**0.31**

Table 2: Time comparison for Subspace alignment (SA) and Entropy regulerized(ER) DA. Dimension $d = 20$, Regularization parameter $\lambda = 5$

Accuarcy (%) (%)							
method	W→D	C→A	D→C	C→W	A→D	W→A	C→D
SADA	**87**	**45**	**32**	**25**	**32**	**32**	**38**
ERDA	55	33	23	19	32	27	28

Table 3: Accuracy comparison for Subspace alignment (SA) and Entropy regulerized(ER) DA. Dimension $d = 100$, Regularization parameter $\lambda = 10$

5.4 Performance Comparison between our Sinkhorn-Knopp and POT (Python Optimal Transport)

We implement the Entropy regularized optimal transport algorithm and call it **ERDA-O** using fixed point iteration method and report the comparison of its result with that of of the POT library (**ERDA-POT**). We observe that our version achieves similar result to the POT version as seen from table 5. The accuracy of both versions and exactly across all domains.

The different however, can be observed in the running time of each algorithm. Our version lags behind in running time when compared to the POT version. We attribute this behavior extra time required to check the halt criterion.

Time (secs) (%)							
method	W→D	C→A	D→C	C→W	A→D	W→A	C→D
SADA	0.72	**0.95**	0.84	0.82	0.75	0.76	0.76
ERDA	**0.12**	2.91	**0.36**	**0.58**	**0.26**	**0.58**	**0.31**

Table 4: Time comparison for Subspace alignment (SA) and Entropy regulerized(ER) DA. Dimension $d = 100$, Regularization parameter $\lambda = 10$

Accuarcy (%)							
method	W→D	C→A	D→C	C→W	A→D	W→A	C→D
ERDA-O	59	34	24	23	33	25	32
ERDA-POT	59	34	24	23	33	25	32

Table 5: Accuracy comparison for ERDA-O and ERDA-POT. Regularization parameter $\lambda = 5$

Time (secs)							
method	W→D	C→A	D→C	C→W	A→D	W→A	C→D
ERDA-O	0.16	3.68	**0.34**	0.57	**0.26**	0.56	0.31
ERDA-POT	**0.14**	**2.82**	0.35	**0.56**	0.31	**0.55**	0.31

Table 6: Time comparison for ERDA-O and ERDA-POT. Regularization parameter $\lambda = 5$

6 CONCLUSION

In this paper, we presented an experiment comparison between subspace alignment domain adaptation and entropy regularized domain adaptation (sinkhorn's algorithm). We conclude that although SADA lags behind ERDA in running time, it outperforms ERDA in terms of accuracy, achieving state-of-the-art result domain adaptation result for Office/Caltech dataset.
We also self-coded our version of Sinkorn's algorithm and compare its performance with that of POT (Python Optimal Transport library). We see that both algorithm behave alike in terms of performance and with a little difference in running time.

References

[1] Basuro F., Amaury H., Sebban M. Unsupervised Visual Domain Adaptation Using Subspace Alignment. *International Conference on Computer Vision, ICCV*, pp.2960-2967, 2013.

[2] Werman, M., Peleg, S., and Rosenfeld, A. A distance metric for multidimensional histograms. Computer Vision, Graphics, and Image Processing, 32(3):328 – 336, 1985.

[3] Rubner, Y., Tomasi, C., and Guibas, L. J. The earth mover's distance as a metric for image retrieval. International journal of computer vision, 40(2):99–121, 2000.

[4] Villani, C. Optimal transport: old and new, volume 338. *Springer Science & Business Media*, 2008.

[5] Marco Cuturi. Sinkhorn distances: lightspeed computation of optimal transport. *Proceedings of the 26th International Conference on Neural Information Processing Systems*, 26(2):2292-2300, 2013.