Transfer Learning and Optimal Transport

Oluchi Ibeneme[a] and *__Shamprikta Mehreen__
oluchi.ibeneme@etu.univ-st-etienne.fr
shamprikta.mehreen@etu.univ-st-etienne.fr
University Jean Monnet,Saint-Etienne, France

Abstract

In this report, two different domain adaptation methods Subspace Alignment algorithm and Sinkhorn-Knopp algorithm (from python optimal transport library (POT)[3]) are analyzed. Then we compared their results by testing them on a real-world computer vision Office[10]/Caltech[8] [7] data set, performing One-nearest neighbour classification (1-NN). Finally, we compare the result of our version of Sinhorn-Knopp Algorithm with the POT version.

Keywords: Domain Adaptation, Transfer Learning, Optimal Transport, Subspace Alignment algorithm, Sinkhorn-Knopp algorithm.

1 Introduction

Domain Adaptation is the ability to apply an algorithm trained in one or more "source domains" to a different (but related) "target domain" [2]. **Transfer Learning** is a machine learning method where we can reuse a classifier built for a task as the starting point for a classifier on another task. It requires the transfer of acquired knowledge of a model to a different or similar context through mapping the features of domain space to target space. Unlike supervised learning, transfer learning does not involve learning from a subset of the original data and then to predict on remaining subset. Rather it learns on the source domain (S) and using the acquired knowledge from the source domain, it attempts to learn the target domain (T).

In this report, the source domain data is denoted by S, the target domain data is denoted by T, the source domain labels L_s and the target domain labels L_T.

2 Subspace Alignment

The idea behind **Subspace Alignment** is to directly reduce the discrepancy between the two domains by moving the source and target subspaces closer [6]. First, we compute the d principal components having the highest variance. We represent the source domain by source subspace $X_S \in R^{n_s \times d}$ and the target domain is represented by the target subspace $X_T \in R^{n_t \times d}$.

The objective of subspace alignment is to generate X_s and X_t which consists of learning the alignment or transformation matrix M, which is used for mapping the X_s to X_t. M is learned by minimizing the given **Bregman matrix divergence**[1]:

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

$$M* = argmin_M(F(M)) \tag{2}$$

Where $||_F^2$ is the Frobenius norm. As the Frobenius norm is invariant to orthonormal oprations, we can rewrite equation (2) as follows:

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

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

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

Then using the alignment matrix M, we compute $X_a = M.X_s$ such that the aligned source subspace(X_a) and the target subspace remains as close as possible. Then we project the source data to the target aligned source subspace $S_a = SX_a$ and target data to the target subspace $T_a = TX_T$. Finally, we fit a 1-NN classifer on S_a and make predicitons on T_a.

The algorithm for the subspace alignment is given below

Algorithm 1 Subspace Alignment Algorithm

Input: S (Source Data)
T(Target Data)
$L_s(SourceLabels)$
Output: $L_T(TargetLabels)$
begin
$X_s \leftarrow PCA(S,d)$;
$X_T \leftarrow PCA(T,d)$;
$X_a \leftarrow X_s X_s^T X_T$;
$T_T \leftarrow TX_T$;
$T_T \leftarrow 1-NN-Classifier(S_a,T_T,L_s)$;
end

3 Entropic Regularized Optimal Transport

The entropy regularized optimal transport between two empirical distributions of data matrices $S \in R^{n_s \times d}$ and $T \in R^{n_t \times d}$. It requires adding an regularization term λ to the minimization problem. We used the fixed point iteration method proposed in [5] for solving the entropy optimization problem .

Algorithm 2 Sinkhorn-Knopp fixed point iteration for computing $d_M^{\lambda}(a,b)$

Input: M, λ, a, b
Output: $d_M^{\lambda}(a,b)$
begin
$I \leftarrow a(a>0)$;
$a \leftarrow a(I)$;
$M \leftarrow M(I,:)$;
$K \leftarrow exp(-\lambda * M)$;
$x \leftarrow ones(length(a), size(b,2))$
$/length(r)$;
while x changes **do**
$x \leftarrow diag(1./r) * K *$
$(b * (K^T * (1/x)))$;
$u \leftarrow 1/x$;
$v \leftarrow b * (K^T * u)$;
$d_M^{\lambda} = \sum_{i=1}^{N} (u_i * (K * M) * v)$
end while
end

4 Experiment

4.1 Datasets

We experimented using the computer vision Office/Caltech data where the classification task is to assign an image to a class based on its content. All the three datasets (Surf, Googlenet, Caffenet) have 4 domains Amazon (A), Caltech (C),Webcam (W) and DSLR (D). We normalize dataset using *StandardScaler* from Sklearn.

4.2 Performance Metric

We used One Nearest Neighbour (1-NN) for testing the different domain adaption methods Subspace Alignment, Entropy regularization and Sinkhorn-knopp algorithm. We then compared the performance of these methods using Accuracy metric. **Accuracy** is of how often the classifier makes correct prediction.

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \tag{5}$$

Where **TP** is True Positives, **TN** is True Negatives, **FP** is False Positives and **FN** is False Negatives.

5 Results

We compared the classification results in terms of accuracy and time using Surf[9], Googlenet and Caffenet datasets. Four domains Webcam, Dslr, Amazon and Caltech has been donated by W, D, A and C respectively.

We performed knn [4] using different domain adaptation methods 1) Subspace Alignment, 2) Only using Knn on raw data, 3) Sinkhorn-knopp algorithm (using Python Optimal Transport Library), and 4) Sinkhorn-knopp algorithm (our implementation) on all possible combination of Source domain and Target domain. All the comparisons are given below.

Accuracy(%)						
						Surf Dataset
Source Domain	W			D		
Target Domain	D	A	C	W	A	C
Subspace Alignment	0.90	0.29	0.29	0.81	0.29	0.29
knn on raw data	0.31	0.18	0.29	0.81	0.14	0.14
Sinkhorn OT	0.73	0.33	0.28	0.60	0.29	0.28
Sinkhorn	0.55	0.27	0.18	0.47	0.28	0.22

Table 1: Accuracy(%) comparison on the Surf Dataset using $d = 105$ for Subspace Alignment, Applying knn on raw data (without subspace Alignment), regularization $\lambda = 10$ for both Sinkhorn-POT version and our implemented version.

Accuracy(%)						
						Surf Dataset
Source Domain	**A**			**C**		
Target Domain	**W**	**D**	**C**	**W**	**D**	**A**
Subspace Alignment	0.38	0.36	0.39	0.33	0.39	0.42
knn on raw data	0.18	0.11	0.20	0.12	0.09	0.18
Sinkhorn OT	0.28	0.30	0.24	0.23	0.33	0.29
Sinkhorn	0.33	0.32	0.27	0.19	0.28	0.33

Table 2: Accuracy(%) comparison comparison on the Surf Dataset using $d = 105$ for Subspace Alignment, Applying knn on raw data (without subspace Alignment), regularization $\lambda = 10$ for both Sinkhorn-POT version and our implemented version.

Time (secs)						
						Surf Dataset
Source Domain	**W**			**D**		
Target Domain	**D**	**A**	**C**	**W**	**A**	**C**
Subspace Alignment	0.02	0.12	0.19	0.02	0.08	0.09
knn on raw data	0.12	0.61	0.69	0.11	0.31	0.37
Sinkhorn OT	0.10	0.46	0.53	0.10	0.27	0.31
Sinkhorn	0.10	0.43	0.47	0.10	0.25	0.29

Table 3: Time (secs) comparison on the Googlenet Dataset using $d = 105$ for Subspace Alignment, Applying knn on raw data (without subspace Alignment), regularization $\lambda = 10$ for both Sinkhorn-POT version and our implemented version.

Time (secs)						
						Surf Dataset
Source Domain	**A**			**C**		
Target Domain	**W**	**D**	**C**	**W**	**D**	**A**
Subspace Alignment	0.09	0.04	0.26	0.11	0.06	0.29
knn on raw data	0.56	0.33	1.90	0.49	0.48	2.22
Sinkhorn OT	0.39	0.26	1.37	0.49	0.34	1.34
Sinkhorn	0.37	0.25	1.22	0.46	0.32	1.31

Table 4: Time (secs) comparison on the Googlenet Dataset using $d = 105$ for Subspace Alignment, Applying knn on raw data (without subspace Alignment), regularization $\lambda = 10$ for both Sinkhorn-POT version and our implemented version.

Accuracy(%)						
						Googlenet Dataset
Source Domain	**W**			**D**		
Target Domain	**D**	**A**	**C**	**W**	**A**	**C**
Subspace Alignment	0.99	0.90	0.84	0.99	0.87	0.83
knn on raw data	0.99	0.87	0.79	0.98	0.82	0.79
Sinkhorn OT	0.97	0.93	0.89	0.96	0.93	0.87
Sinkhorn	0.99	0.95	0.91	0.97	0.91	0.91

Table 5: Accuracy(%) comparison on the Googlenet Dataset using $d = 105$ for Subspace Alignment, Applying knn on raw data (without subspace Alignment), regularization $\lambda = 10$ for both Sinkhorn-POT version and our implemented version.

Accuracy(%)						
						Googlenet Dataset
Source Domain	**A**			**C**		
Target Domain	**W**	**D**	**C**	**W**	**D**	**A**
Subspace Alignment	0.83	0.89	0.86	0.89	0.92	0.92
knn on raw data	0.80	0.90	0.87	0.89	0.90	0.92
Sinkhorn OT	0.88	0.90	0.88	0.94	0.90	0.95
Sinkhorn	0.90	0.96	0.90	0.95	0.92	0.94

Table 6: Accuracy(%) comparison on the Googlenet Dataset using $d = 105$ for Subspace Alignment, Applying knn on raw data (without subspace Alignment), regularization $\lambda = 10$ for both Sinkhorn-POT version and our implemented version.

Time (secs)						
						Googlenet Dataset
Source Domain	**W**			**D**		
Target Domain	**D**	**A**	**C**	**W**	**A**	**C**
Subspace Alignment	0.01	0.10	0.12	0.02	0.07	0.08
knn on raw data	0.14	0.69	0.83	0.15	0.38	0.45
Sinkhorn OT	0.10	0.36	0.46	0.12	0.27	0.31
Sinkhorn	0.09	0.37	0.21	0.11	0.28	0.36

Table 7: Time (secs) comparison on the Googlenet Dataset using $d = 105$ for Subspace Alignment, Applying knn on raw data (without subspace Alignment), regularization $\lambda = 10$ for both Sinkhorn-POT version and our implemented version.

Time (secs)						
						Googlenet Dataset
Source Domain	**A**			**C**		
Target Domain	**W**	**D**	**C**	**W**	**D**	**A**
Subspace Alignment	0.72	0.03	0.22	0.09	0.41	0.21
knn on raw data	0.71	0.44	2.43	0.97	0.58	2.27
Sinkhorn OT	0.32	0.23	1.41	0.38	0.25	0.97
Sinkhorn	0.31	0.21	1.14	0.33	0.23	0.78

Table 8: Time (secs) comparison on the Googlenet Dataset using $d = 105$ for Subspace Alignment, Applying knn on raw data (without subspace Alignment), regularization $\lambda = 10$ for both Sinkhorn-POT version and our implemented version.

Accuracy(%)						
						Caffenet Dataset
Source Domain	**W**			**D**		
Target Domain	**D**	**A**	**C**	**W**	**A**	**C**
Subspace Alignment	1.00	0.81	0.72	0.99	0.83	0.74
knn on raw data	0.96	0.70	0.61	0.95	0.71	0.67
Sinkhorn OT	0.99	0.87	0.80	0.93	0.86	0.79
Sinkhorn	1.0	0.92	0.87	0.98	0.93	0.86

Table 9: Accuracy(%) comparison on the Caffenet Dataset using $d = 105$ for Subspace Alignment, Applying knn on raw data (without subspace Alignment), regularization $\lambda = 10$ for both Sinkhorn-POT version and our implemented version.

Accuracy(%)						
						Caffenet Dataset
Source Domain	**A**			**C**		
Target Domain	**W**	**D**	**C**	**W**	**D**	**A**
Subspace Alignment	0.81	0.81	0.79	0.80	0.85	0.88
knn on raw data	0.74	0.82	0.76	0.85	0.86	0.88
Sinkhorn OT	0.88	0.87	0.84	0.78	0.85	0.89
Sinkhorn	0.90	0.92	0.86	0.90	0.91	0.91

Table 10: Accuracy(%) comparison on the Caffenet Dataset using $d = 105$ for Subspace Alignment, Applying knn on raw data (without subspace Alignment), regularization $\lambda = 10$ for both Sinkhorn-POT version and our implemented version.

Time (secs)						
						Caffenet Dataset
Source Domain	**W**			**D**		
Target Domain	**D**	**A**	**C**	**W**	**A**	**C**
Subspace Alignment	0.01	0.11	0.12	0.02	0.77	0.09
knn on raw data	0.58	0.93	3.56	0.51	1.55	1.88
Sinkhorn OT	0.32	1.71	2.10	0.35	1.03	1.20
Sinkhorn	0.3	1.50	1.90	0.36	0.96	1.22

Table 11: Time (secs) comparison on the Caffenet Dataset using $d = 105$ for Subspace Alignment, Applying knn on raw data (without subspace Alignment), regularization $\lambda = 10$ for both Sinkhorn-POT version and our implemented version.

Time (secs)						
						Caffenet Dataset
Source Domain	**A**			**C**		
Target Domain	**W**	**D**	**C**	**W**	**D**	**A**
Subspace Alignment	0.08	0.04	0.24	8.11	0.05	0.25
knn on raw data	3.01	1.75	10.54	4.40	2.46	11.81
Sinkhorn OT	1.49	0.89	5.44	1.71	1.05	4.26
Sinkhorn	1.29	0.71	5.58	1.29	.80	3.37

Table 12: Time (secs) comparison on the Caffenet Dataset using $d = 105$ for Subspace Alignment, Applying knn on raw data (without subspace Alignment), regularization $\lambda = 10$ for both Sinkhorn-POT version and our implemented version.

6 Discussion

From the above result, we can observe that for the Surf dataset, Subspace Alignment(SA) outperforms the result using raw data (without using SA) in terms of accuracy, and SA is faster in terms of time. SA outperforms Sinkhorn (POT library) as well in terms of accuracy and again, it's faster than Sinkhorn(POT). Sinkhorn (POT version) performs slightly better than our version of Sinkhorn when it comes to accuracy. But our version of Sinkhorn is faster than the Sinkhorn POT version.

For the Googlenet dataset, we can notice that the SA gives slightly better accuracy than the one using raw data (without using SA), and SA is faster in terms of time. SA and Sinkhorn (POT library) gives similar accuracy and, it's faster than Sinkhorn(POT). Sinkhorn (POT version) and our version gives very similar result when it comes to accuracy. And our version of Sinkhorn is faster than the Sinkhorn POT version.

For the Caffenet dataset, we can observe that the SA gives better accuracy than the one using raw data (without using SA), and SA is faster as well in terms of time.

SA and Sinkhorn (POT library) gives very similar accuracy but, SA is faster than Sinkhorn(POT). Our version of Sinkhorn gives better accuracy than Sinkhorn (POT version). And our version of Sinkhorn is faster than the Sinkhorn POT version.

7 Conclusions

In this report, we analyzed Subspace Alignment domain adaptation and entropy regularized domain adaptation (Sinkhorn-knopp Algorithm). We come to the conclusion that Subspace Alignment method is faster than Entropy Regularization method, and it outperforms Entropy Regularization when it comes to accuracy. We also implemented and compared the performance of the implemented Sinkhorn-knopp algorithm with the POT (Python Optimal Transport Library) version and noticed both algorithms gives very similar accuracy, and our version is faster than the POT version of Sinkhorn-knopp Algorithm.

References

[1]

[2] Domain adaptation.
URL `https://en.wikipedia.org/wiki/Domain_adaptation`

[3] Pot: Python optimal transport.
URL `https://pythonot.github.io/`

[4] sklearn.neighbors.kneighborsclassifier.
URL `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`

[5] M. Cuturi, Sinkhorn distances: Lightspeed computation of optimal transport, in: Advances in neural information processing systems, 2013, pp. 2292–2300.

[6] B. Fernando, A. Habrard, M. Sebban, T. Tuytelaars, Subspace alignment for domain adaptation, arXiv preprint arXiv:1409.5241.

[7] B. Gong, Y. Shi, F. Sha, K. Grauman, Geodesic flow kernel for unsupervised domain adaptation, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 2066–2073.

[8] R. Gopalan, R. Li, R. Chellappa, Domain adaptation for object recognition: An unsupervised approach, in: 2011 international conference on computer vision, IEEE, 2011, pp. 999–1006.

[9] T. T. Herbert Bay, L. Gool, Surf: Speeded up robust features, in: 9th European conference on computer vision, Vol. 3951, 2006, pp. 404–417.

[10] K. Saenko, B. Kulis, M. Fritz, T. Darrell, Adapting visual category models to new domains, in: European conference on computer vision, Springer, 2010, pp. 213–226.