Efficient Parameter-free Clustering Using First Neighbor Relations Supplementary Material

M. Saquib Sarfraz[1,2], Vivek Sharma[1], Rainer Stiefelhagen[1]
[1]Karlsruhe Institute of Technology
[2]Daimler TSS, Germany
{firstname.lastname}@kit.edu

1. Appendix

1.1. Datasets Detail

We have used CNN features for STL-10, BBT/Buffy and MNIST datasets. Specifically, we use ResNet50 pretrained on ImageNet for STL-10, and pretrained VGG2-Face ResNet50 model features [1] for BBTs01, BFs05. On MNIST datasets, we show clustering results both on raw pixels and trained CNN features. Since the clustering algorithms performance varies with varying feature dimensions, we have used both 4096 and 256-dimesional features for MNIST to have this diversity of feature dimensions. We use [4] CNN features (4096-dim) for MNIST_10k and MNIST_70k, while for MNIST_8M we train a two layer MLP with 512 and 256 neurons with 60% data used for training. Here, the last 256-dimensional layer is used as the feature extractor, `MNIST_8M_CNN`. We also evaluate FINCH on MNIST raw pixel features with 784 (i.e., 28×28) dimensions, `MNIST_8M_PIXELS` and `MNIST_70k_PIXELS`.

1.2. Face Clustering- State-of-the-art Comparison on BBTs01, BFs05 and Accio Datasets

Here, first we report the pairwise F-measure performance on the ground-truth (true) number of clusters (i.e. 5 for BBTs01 (season 1, episodes 1 to 6) and 6 for BFs05 (season 1, episodes 1 to 6)) using the steps described in Algorithm 2. FINCH obtains a pairwise F-measure of $\mathbf{97.42}\%$ and $\mathbf{94.02}\%$ respectively. In comparison, the reported pairwise F-measure of the recent work by Jin *et al*. [5] for BBTs01 and BFs05 are 78.2% and 62.99% respectively.

Method	ACC		P	R	F
	BFs05e02	BBTs01e01	Accio, #C=40		
ULDML [2]	41.62	57.00	–	–	–
HMRF [10]	50.30	60.00	27.2	12.8	17.4
WBSLRR [11]	62.76	72.00	29.6	15.3	20.2
JFAC [13]	92.13	–	71.1	35.2	47.1
Imp-Triplet [12]	–	96.00	–	–	–
VDF [6]	87.46	89.62	–	–	–
TSiam [7]	92.46	98.58	76.3	36.2	49.1
SSiam [7]	90.87	99.04	77.7	37.1	50.2
FINCH	**92.73**	**99.16**	**73.30**	**71.11**	**72.19**

Table 1. SOTA comparison on face clustering.

In addition to BFs05 (season 5, episodes 1 to 6) and BBTs01 (season 1, episodes 1 to 6), we also report the individual performances on the popularly used episodes of BFs05 and BBTs01, these are BFs05e02 (season 5, episodes 2) and BBTs01e01 (season 1, episodes 1). BBTs01e01 has 41,220 frames, and BFs05e02 has 39,263 frames. In Table 1, we compare the performance of FINCH with the current state-of-the-art algorithms on BBTs01e01 and BFs05e02. FINCH estimates 7 clusters for BFs05e02, and is brought down to ground-truth number of 6 clusters using Algorithm 2, while for BBTs01e01 FINCH estimated exactly 5 clusters which is also the true number of clusters.

We also include FINCH results on Accio dataset [3] (Harry Potter movie series with a large number of dark scenes) with 36 named characters and 166885 faces/samples to cluster. The largest to smallest cluster ratios of Accio are very skewed: 30.65% and 0.06%. The performance on Accio is measured with B-Cube precision, recall and F-score on #clusters=40 as in the compared methods.

Note that, FINCH has simply clustered the extracted VGG2 feature vectors for the frames, without any data specific feature training/transfer and without exploiting any form of video level constraints as used in the compared method: video-level constraints [2], video editing style [8], dynamic clustering constraints in CNNs via MRF modeling [13], triplet loss based CNN training using must-link and must-not-link constraints obtained from video-level constraints [12], Siamese feature transfer/training [7] using track-level constraints (TSIAM) and a self-supervised transfer (SSIAM) on the extracted VGG2 feature vectors.

Steps/Partitions	MICE_PROTEIN	REUTERS	HAR	STL-10	MNIST_10k	MNIST_70K	BBTs01	BFs05	MNIST_8M_CNN	MNIST_8M_PIXELS
	1077	**10k**	**10299**	**13k**	**10k**	**70k**	**199346**	**206254**	**8.1M**	**8.1M**
1	351	1837	2465	2061	1699	11493	27294	29150	2015305	1845149
-	99.7214	94.36	96.7667	95.9462	99.25	99.7243	98.4053	95.8561	99.9999	99.9999
2	106	220	369	177	310	1891	6067	6830	694525	691696
-	96.9359	89.39	92.776	94.9846	99.18	99.68	98.3285	95.6137	99.997	99.9725
3	24	24	88	37	65	357	1406	1753	161371	175426
-	64.624	82.2	86.5521	94.7846	99.18	99.6729	98.2819	95.0396	99.9816	98.6474
4	**8**	**4**	18	**10**	17	75	251	355	33217	25609
-	**51.2535**	**66.14**	74.4441	**85.2846**	99.18	99.6729	98.0918	94.0888	99.951	94.3231
5	4	-	**6**	2	**10**	17	36	50	6945	4085
-	26.9266	-	**60.2389**	20	**99.18**	99.6729	97.9413	94.0748	99.9067	93.1139
6	3	-	2	-	-	**10**	**6**	**7**	1362	789
-	15.506	-	35.5957	-	-	**99.6729**	**97.9413**	**94.0748**	99.8833	92.2973
7	-	-	-	-	-	-	2	2	262	177
-	-	-	-	-	-	-	49.7778	54.6646	99.8762	90.3298
8	-	-	-	-	-	-	-	-	53	43
-	-	-	-	-	-	-	-	-	99.8762	83.359
9	-	-	-	-	-	-	-	-	**13**	**11**
-	-	-	-	-	-	-	-	-	**99.8762**	**66.6879**
10	-	-	-	-	-	-	-	-	4	-
-	-	-	-	-	-	-	-	-	41.1708	-

Table 2. FINCH steps run for all used datasets. Total number of clusters in each partition along with its respective accuracy as measured by Clustering Accuracy (ACC), each row is represented as ($^{\#clusters}_{ACC}$).

1.3. Partitions for each dataset

Table 2 shows the total FINCH steps run for all used datasets. Each step produces a partition of data with shown number of clusters in each. The clustering accuracy (ACC) at each step is reported that demonstrate the quality of the merges and the clusters obtained. As can be seen, despite the varying nature/distribution and dimensionality of data, FINCH is able to recover the ground-truth or a very close partition in all cases.

1.4. Details for baselines

We have tried different parameters for the baselines, and where applicable used the recommended parameters (by respective authors), to report their best NMI scores. As an example, Table 3 shows the impact of changing the preference parameter of Affinity prop (AP) on two datasets. As shown, on preference=-100 AP estimates ground truth 8 clusters on mice-protein data, but the performance is worst, while the same preference value on MNIST-10k data produces far more clusters than groundtruth while also lower NMI score. The hyper-parameters do not generalize on different data. This also motivates why a parameter-free clustering algorithm is needed.

For AP, SC, BR and kmeans++ we use the implementation available in the scikit-learn package. For RO, we use the implementation available in OpenBR framework

Dataset	Affinity Propagation (AP): Preference				FINCH	True #C
	Reported	-10	-50	-100		
Mice_Protein	**59.10**	57.63	44.09	37.93	**51.64**	
Estim. #C	67	48	12	**8**	**8**	8
MNIST_10k	69.97	58.02	63.97	67.13	**97.55**	
Estim. #C	116	1260	304	178	**10**	10

Table 3. Example: Impact of varying preference parameter of AP.

`http://openbiometrics.org/`. For HAC we use the implementation provided by Matlab. For JP, we use the implementation available from [9]. For RCC, we use the python implementation recommended by the authors. Similarly for SSC and MV-LRSSC we have used the Matlab implementations provided by the respective authors. Table 4 summarizes the used parameters for the baselines.

References

[1] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *FG*, 2018.

[2] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Unsupervised Metric Learning for Face Identification in TV Video. In *ICCV*, 2011.

[3] Esam Ghaleb, Makarand Tapaswi, Ziad Al-Halah, Hazim Kemal Ekenel, and Rainer Stiefelhagen. Accio: A data set for face track retrieval in movies across age. In *Proceedings*

Algorithm	Parameter settings
AP	Preference = median of similarities, damping factor = 0.5, max_iter=200, convergence_iter=15
JP	# neighbors=10, min_similarity_to_cluster =0.2, min_similarity_as_neighbor=0
RO	distance threshold t=14, number of top neighbors K=20
RCC	maximum total iteration $maxiter$=100, maximum inner iteration $inner\_iter$=4
BR	threshold=0.5, branching_factor=50
Kmeans	init="k-means++", n_init=10, max_iter=300
SC	eigen_solver=*"arpack"*, affinity=*"nearest_neighbors"*, n_neighbors=10
SSC	r=0, affine=false, alpha=20, outlier=true, rho=1
MV-LRSSC	mu=0.0001, lambda1=0.9, lambda2=0.1, lambda3=0.7, noisy=true

Table 4. Parameter settings for baselines

of the 5th ACM on International Conference on Multimedia Retrieval, pages 455–458. ACM, 2015.

[4] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv:1406.2227*, 2014.

[5] SouYoung Jin, Hang Su, Chris Stauffer, and Erik Learned-Miller. End-to-end Face Detection and Cast Grouping in Movies using ErdsRnyi Clustering. In *ICCV*, 2017.

[6] Vivek Sharma, M Saquib Sarfraz, and Rainer Stiefelhagen. A simple and effective technique for face clustering in tv series. In *CVPR Workshop on Brave New Motion Representations*, 2017.

[7] Vivek Sharma, Makarand Tapaswi, M Saquib Sarfraz, and Rainer Stiefelhagen. Self-supervised learning of face representations for video face clustering. In *International Conference on Automatic Face and Gesture Recognition.*, 2019.

[8] Makarand Tapaswi, Omkar M Parkhi, Esa Rahtu, Eric Sommerlade, Rainer Stiefelhagen, and Andrew Zisserman. Total Cluster: A Person Agnostic Clustering Method for Broadcast Videos. In *ICVGIP*, 2014.

[9] Ágnes Vathy-Fogarassy and János Abonyi. *Graph-based clustering and data visualization algorithms*. Springer, 2013.

[10] Baoyuan Wu, Yifan Zhang, Bao-Gang Hu, and Qiang Ji. Constrained Clustering and its Application to Face Clustering in Videos. In *CVPR*, 2013.

[11] Shijie Xiao, Mingkui Tan, and Dong Xu. Weighted Block-sparse Low Rank Representation for Face Clustering in Videos. In *ECCV*, 2014.

[12] Shun Zhang, Yihong Gong, and Jinjun Wang. Deep Metric Learning with Improved Triplet Loss for Face Clustering in Videos. In *Pacific Rim Conference on Multimedia*, 2016.

[13] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Joint face representation adaptation and clustering in videos. In *ECCV*, 2016.