CI-GAN : CO-CLUSTERING BY INFORMATION MAXIMIZING GENERATIVE ADVERSARIAL NETWORKS

Jaejun Lee[*], *Hyun Chul Lee*[†] *and Tomasz Palczewski*[†]

[*]David R. Cheriton School of Computer Science, University of Waterloo
[†]Samsung Research America

ABSTRACT

Simultaneously clustering rows and columns of a matrix, co-clustering can exploit the complex relationships between two different domains and identify groups of distinct nature. In this work, we introduce CI-GAN, a novel GAN-based approach for co-clustering. The model exploits two distinct GAN that cluster each domain independently and combines them intelligently by maximizing the mutual information between the input data and the generated co-clusters. From the experiments constructed with image, audio, and text datasets, it is found that such a systematic way of sharing information between the networks can improve co-clustering performance substantially; when CI-GAN is compared against five standard algorithms, it consistently reports the highest accuracy on both synthetic and real datasets.

Index Terms— Co-clustering, Generative modeling

1. INTRODUCTION

Capable of clustering both rows and columns of a data matrix simultaneously, co-clustering is an effective data analysis technique which exploits the duality of the entries. Circumventing the limitations of traditional clustering algorithms, various co-clustering algorithms have been leveraged to analyze multidimensional data in vastly different fields including information retrieval, computer vision as well as recommendation systems [1, 2, 3]. Realizing the wide range of use cases, mainly arose from the diversity in data format, many algorithms exploit the duality in different ways [4, 5, 6, 7]. However, co-clustering using representation learning is still veiled in mystery despite its effectiveness in extracting semantic features from unlabeled data. In this paper, we take the first step into the uncharted domain with *Co-Informatic Generative Adversarial Network* (CI-GAN), an information-theoretic extension of GAN for co-clustering.

The main components of CI-GAN are two InfoGAN [8] which are capable of distinguishing different types of rows and columns by maximizing the mutual information between the noise variables and the observations. To fully exploit the interplay between the two independent data representations, our model further combines the two networks into a single architecture intelligently by maximizing the mutual information between the input data and the generated co-clusters. It is motivated by the belief that a good generative modeling is effective not only to learn the disentangled representations of rows and columns independently, but also to learn the joint disentangled representation that can decipher the domain interplay structures in the correlation data space.

Applying different techniques to solve the maximization problem, we introduce two architectures under the name of CI-GAN. Our first version, *Combiner*, maximizes the mutual information objective in a relatively simplistic way; it captures the entries of similar behavior directly using an additional encoding vector. On the other hand, the second version, *AutoEncoder*, maximizes the lower bound of the mutual information objective by reconstructing the original data from paired domain-specific clustering labels.

To understand the effectiveness of CI-GAN, we evaluate the two architectures against five standard co-clustering algorithms on both synthetic and real datasets; synthetic datasets are designed to cover a wide range of general co-clustering cases, whereas real datasets focus on the most complicated case of co-clustering where the relationship between the domains are not sufficiently explicit. Throughout the experiments, we realize that the augmented objective of CI-GAN is particularly effective when the interrelation between the domains is fairly complex; our models consistently achieve the highest accuracy on various co-clustering cases.

2. GENERATIVE MODELING FOR CLUSTERING

Recently, neural networks have been a popular mechanism for generating clustering-friendly representations increasing the clustering accuracy in many domains [7]. One of the popular trends in this area is to use GAN as demonstrated by InfoGAN [8], BiGAN [9], and ClusterGAN [10].

Presented by Goodfellow et al., GAN consists of two neural networks, generator G and discriminator D. The goal of GAN is to generate realistic samples using G which is trained to fool D. The role of the adversarial network D is to distinguish real data from the output of G improving the quality of the generated data [11]. Overall, the objective can be written as follows, for a random variable z and a data sample x:

978-1-6654-3864-3/21/$31.00 ©2021 IEEE

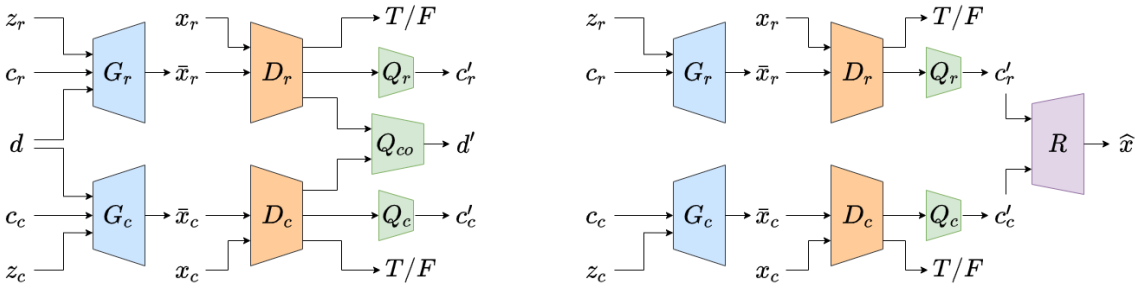

Fig. 1. Architecture of *Combiner* (left) and *AutoEncoder* (right). Both networks have G, D, and Q for clustering rows ($*_r$) and columns ($*_c$). However, training *Combiner* involves Q_{co} that reconstructs co-cluster encoding vector d, whereas *AutoEncoder* exploits a generator-like network R that reconstructs a paired original data $\hat{x}$ from a pair of generated cluster labels, c'_r and c'_c.

$$\min_G \max_D V(D,G)$$
$$= \mathbb{E}_{x\sim P_{data}}[logD(x)] + \mathbb{E}_{z\sim P_{noise}}[log(1-D(G(z)))]$$

When GAN is trained correctly, it can learn the mapping from a random distribution P_{noise} to the data distribution P_{data}. However, it has a clear limitation: the mapping is non-deterministic and unpredictable.

To overcome the limitation, InfoGAN introduces an adjustable generator; along with z, the generator of InfoGAN takes in an additional encoding vector c which captures the disentangled representation of the data distribution. By maximizing the mutual information I between c and the observation, the encoding vector can be used as a knob to generate different styles of data. Unfortunately, it is found that maximizing I directly is quite difficult. Therefore, the authors introduce an additional network Q which maximizes L_I, the lower bound of I, by reconstructing the original encoding vector c from the generated data [8].

$$\min_{G,Q} \max_D V(D,G) - L_I(G,Q)$$

3. CI-GAN

Although many variants of generative modeling have demonstrated their effectiveness in grouping data of similar pattern, we are not aware of any application for co-clustering problem. Therefore, we attempt to bridge this gap with CI-GAN, a novel GAN-based co-clustering framework.

The encoding vector c of InfoGAN can either be categorical or continuous. When c is categorical, each variation is often mapped to a different type of data as in the case of clustering. Given that co-clustering involves two distinct clustering processes, namely row and column clustering, our framework starts off with two distinct InfoGAN models that cluster rows ($*_r$) and columns ($*_c$) independently.

$$\text{rows: } \min_{G_r,Q_r} \max_{D_r} V(D_r,G_r) - L_I(G_r,Q_r)$$
$$\text{columns: } \min_{G_c,Q_c} \max_{D_c} V(D_c,G_c) - L_I(G_c,Q_c)$$

Unfortunately, the two independent models cannot capture the complex relationship between rows and columns. In this work, we combine them in a way that the two components work together to maximize the mutual information between the input data and the generated co-clusters. We introduce two approaches, *Combiner* and *AutoEncoder*, which differ in the techniques used for the maximization. Due to the limited space, we focus on describing how each variant is designed and relegate other details to Appendix 1 and 2.

In the following section, z refers to a random variable, c and d refer to categorical encoding vectors and x refers to the input data. Since CI-GAN involves reconstruction of the original data and the encoding vectors, variables without any additional style refer to the original form, $\bar{*}$ refer to samples generated by G, $*'$ refer to the reconstructed encoding vectors, and $\hat{*}$ refer to the reconstructed samples of $\bar{*}$. Lastly, subscripts are added to indicate components specific to row or column: $*_r$ and $*_c$, respectively.

Combiner

As a natural extension of InfoGAN, the first approach is to introduce another categorical vector d for every co-cluster (see the left architecture in Figure 1). Since distinct row and column cluster pair represents a co-cluster, d is designed to have a size of $c_r \times c_c$. Consequently, we introduce an auxiliary network Q_{co} and enable the two clustering tasks to share the information they find. Without loss of generality, the objective for *Combiner* can be defined as follows:

$$\min_{G,Q} \max_D V(D_r,G_r) + V(D_c,G_c)$$
$$- L_I(G_r,Q_r) - L_I(G_c,Q_c) - L_I(G,Q_{co})$$

AutoEncoder

The second variant is inspired by information-theoretic co-clustering of Dhillon et al. [1]: co-clustering problem is an optimization problem which maximizes I between the original data and the generated co-clusters. Applying the same concept to CI-GAN, we introduce the following idea.

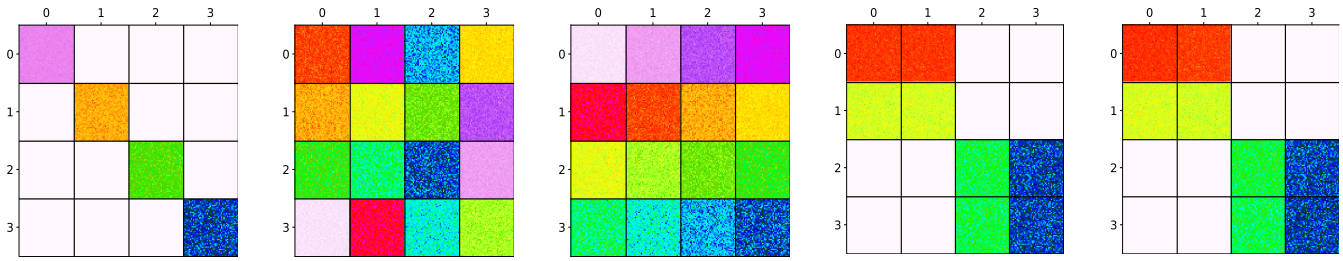

Fig. 2. Sample synthetic datasets illustrating different co-clustering cases. `gist_ncar_r` colormap from Matplotlib is used to color each entry. Starting from left to right: `Diagonal`, `Checker-Shuffled`, `Checker-Ordered`, `Mixed-Identical`, `Mixed-Similar`.

$$
\begin{aligned}
& \max I(x_r, x_c; c'_r, c'_c) \\
= \; & \max I(x_r, x_c; Q_r(\overline{x_r}), Q_c(\overline{x_c})) \\
= \; & \max I(x_r, x_c; Q_r(G_r(z_r, c_r)), Q_c(G_c(z_c, c_c)))
\end{aligned}
$$

Together, the objective can be formulated as follows:

$$
\begin{aligned}
& \min_{G,Q} \max_{D} V(D_r, G_r) + V(D_c, G_c) - I(x; Q(G(z, c))) \\
& I(x; Q(G(z, c))) \\
& \quad = I(x_r, x_c; Q_r(G_r(z_r, c_r)), Q_c(G_c(z_c, c_c)))
\end{aligned}
$$

Unfortunately, it is found that constructing the lower bound of $I(x; Q(G(z, c)))$, L_x, is quite difficult because the derivation involves nested posterior distributions along with the assumption that P_G has the same distribution as P_{data}. To overcome the limitation, *AutoEncoder* introduces another network R which reconstructs the original pairs of row and column samples from the two independent cluster labels c'_r and c'_c as shown in Figure 1.

$$
\begin{aligned}
& \min_{G,Q} \max_{D} V(D_r, G_r) + V(D_c, G_c) \\
& \qquad - L_I(G_r, Q_r) - L_I(G_c, Q_c) - L_x(G, R)
\end{aligned}
$$

4. EXPERIMENTAL SETUP

As we study existing literature on co-clustering, we have found that most works compare the algorithms only based on row clustering performance. When we review the implementation of Deep Co-Clustering [6], it is found that the column labels are not used to calculate the reported metrics. Similarly, Role et al. neglect the column labels in the report for CoClust [12]. From our investigation, it is found that such way of co-clustering evaluation is often inevitable because ground truth co-cluster labels are unknown.

In order to reduce the misalignment between the true co-clustering performance and the reported metrics in this work, we exploit datasets of which both rows and columns are clearly labeled; a co-cluster label is defined as a pair of row and column labels. Consequently, we are able to evaluate the full potential of different co-clustering algorithms.

For the experiments, CI-GAN models are implemented with PyTorch and compared against the five most standard algorithms: co-clustering by bipartite graph partitioning [13], spectral bi-clustering [3], information-theoretic co-clustering [1], co-clustering by spectral approximation [4], and CoClus [2, 5]. To improve readability, they are shortened to *Spec-Co*, *Spec-Bi*, *Info-CC*, *SA-CC*, and *CoClus*, respectively. For *Spec-Co* and *Spec-Bi*, we leverage scikit-learn package and for the remaining algorithms, we leverage the implementations provided by CoClust package [12].

The three metrics we use to compare the algorithms are accuracy (ACC), adjusted Rand index (ARI), and normalized mutual information (NMI). Since clustering is an unsupervised task, assigned labels can be substantially different from the original labels even though the underlying co-clusters are identical. This does not cause any issue for ARI and NMI because they are symmetric metrics. However, ACC is not symmetric and the mismatch can make the value fluctuate. Therefore, we apply Hungarian algorithm [14] to make sure that ACC is calculated from the right mapping. Since a strong correlation is observed among the three metrics, we mainly compare the algorithms based on ACC in this work.

5. EXPERIMENTS ON SYNTHETIC DATASETS

For the first experiment, we evaluate each algorithm on five synthetic datasets which cover vastly different co-clustering cases (see Figure 2). Each dataset has a size of $50,000 \times 50,000$ with values between 0 to 1. The number of clusters for each dimension is set to 4, resulting in 16 distinct co-clusters. Since each of the row and column samples is too huge to be directly fed into D, we simply select 196 indices at random but more sophisticated dimensionality reduction techniques such as PCA or AutoEncoder can be applied.

Prior to the experiments, we find two sets of hyperparameters in which the two models produce consistent results across the five datasets (details are described in Appendix 3). To minimize bias, ten different datasets are used; rows and columns are shuffled and augmented with random noise. In this work, we report the averaged metrics.

Table 1. Performance of different co-clustering algorithms on the five synthetic datasets.

Algorithm	Diagonal			Checker-Shuffled			Checker-Ordered			Mixed-Identical			Mixed-Similar		
	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC
Spec-Co	1.000	1.000	**1.000**	0.750	0.935	0.784	0.319	0.466	0.241	0.231	0.477	0.222	0.221	0.434	0.198
Spec-Bi	1.000	1.000	**1.000**	1.000	1.000	**1.000**	0.114	0.056	0.018	0.459	0.749	0.489	0.461	0.754	0.489
Info-CC	0.344	0.741	0.378	0.556	0.855	0.576	0.751	0.921	0.766	0.250	0.707	0.333	0.250	0.707	0.333
SA-CC	1.000	1.000	**1.000**	0.910	0.908	0.839	0.251	0.323	0.135	0.291	0.503	0.205	0.278	0.506	0.210
CoClus	0.588	0.847	0.585	0.344	0.741	0.378	0.250	0.707	0.333	0.250	0.707	0.333	0.250	0.707	0.333
Combiner	1.000	1.000	**1.000**	1.000	1.000	**1.000**	0.779	0.585	0.682	0.757	0.511	0.549	0.772	0.524	0.551
AutoEncoder	1.000	1.000	**1.000**	1.000	1.000	**1.000**	0.913	0.838	**0.895**	0.810	0.569	**0.568**	0.786	0.565	**0.616**

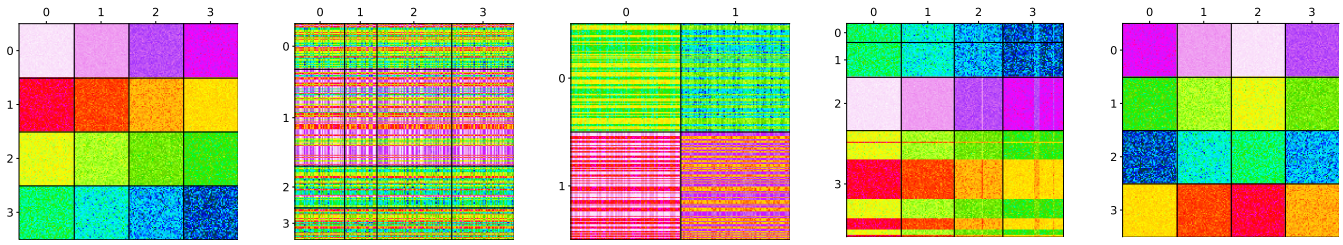

Fig. 3. Co-clustering results on `Checker-Ordered` of which the entries of the same labels are grouped. From left to right, with ACC in brackets: Original, *Spec-Bi* (0.037), *CoClus* (0.333), *Combiner* (0.640), *AutoEncoder* (1.000).

5.1. Datasets

`Diagonal`

In many cases, co-clusters are based on one-to-one mapping of row and column cluster. When rows and columns of such matrix are grouped, target co-clusters can be aligned along the diagonal axis and non-diagonal co-clusters exhibit similar behavior (see the first diagram in Figure 2). We categorize this type of matrix as `Diagonal`. To construct such matrix, we simply assign different values from $[0.1, 0.9]$ for each of the diagonal co-clusters.

`Checker-Shuffled` & `Checker-Ordered`

Another common structure of co-clustering problem is `Checker-Shuffled` where each of the co-clusters shows different characteristics; a random value from $[0.1, 0.9]$ is assigned to each group. Throughout the experiments, we find that many algorithms fail to co-cluster the matrix when row and column clusters have the same pattern but differ by constant: `Checker-Ordered`. In this matrix, the values of the co-clusters are incremented by the same magnitude. The second matrix in Figure 2 illustrates `Checker-Shuffled` and the third illustrates `Checker-Ordered`.

`Mixed-Identical` & `Mixed-Similar`

Matrices of the aforementioned patterns may not require an explicit algorithm for co-clustering as two independent row and column clustering algorithms are sufficient. However, if multiple row or column clusters exhibit the same pattern, distinguishing them without considering both axes is impossible. The fourth matrix in Figure 2, `Mixed-Identical`, captures such scenario. Column clusters 0 and 1 cannot be distinguished without considering row clusters 0 and 1. Similarly, row clusters 2 and 3 cannot be distinguished without considering column clusters 2 and 3. In practice, such case is rare because each cluster often introduces different noise: `Mixed-Similar`, the fifth matrix.

5.2. Results

Table 1 summarizes the performance of each algorithm on the five synthetic datasets.[1] As mentioned in the previous section, co-clustering `Diagonal` and `Checker-Shuffled` are found to be relatively easy; *Spec-Bi*, *Combiner*, and *AutoEncoder* demonstrate superior performance than others finding every co-cluster on both datasets.

On the other hand, most of the standard algorithms generally report lower ACC for `Checker-Ordered`. The only exception is *Info-CC* which achieves higher ACC of 76.6%. It is found that *Combiner* and *AutoEncoder* report comparable ACC of 68.2% and 89.5%, respectively. To better understand the performance, sample co-clusters for `Checker-Ordered` datasets are shown in Figure 3. The visualization also aligns with our finding: CI-GAN models can learn the joint disentangled representation better.

On the two `Mixed-`$\star$ datasets, all the algorithms struggle to distinguish different co-clusters. Among the standard algorithms, `Spec-Bi` is found to be the most stable with the ACC of 48.9% on both datasets. *AutoEncoder* has shown the highest ACC, 56.8% on `Mixed-Identical` and 61.6% on `Mixed-Similar`. *Combiner* reports similar ACC to its variant, 54.9% and 55.1%, respectively.

[1] Appendix 4 illustrates co-clusters generated from the experiments

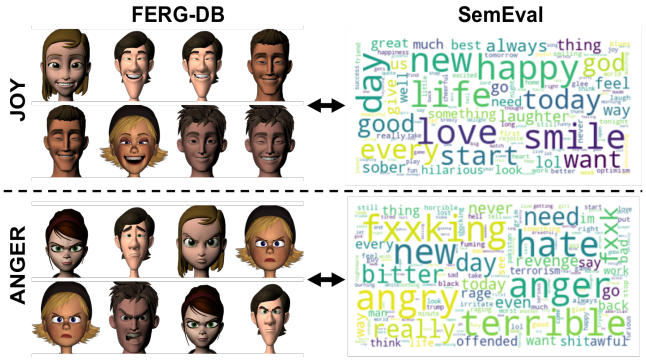

Fig. 4. Sample co-clusters between FERG-DB images and SemEval tweets (represented as word clouds).

Altogether, CI-GAN models demonstrate the most stable performance. Between the two variants, *AutoEncoder* is more robust and flexible; reconstructing paired original data $\hat{x}$ from generated cluster labels c' is more effective in maximizing the mutual information than generating co-cluster encoding vector d' from original data x.

6. EXPERIMENTS ON REAL DATASETS

Even though the experiments on the synthetic datasets cover a wide range of co-clustering cases, they have failed to mimic the most important scenario: the relationship between row and column clusters are strong but not sufficiently explicit. In the following sections, we further analyze the robustness of CI-GAN by evaluating each algorithm on datasets of multiple data formats; image and audio make up the first dataset while image and text make up the other dataset. Again, each of the experiments is repeated ten times and the averaged results are used to compare the algorithms.

6.1. Digit Co-clustering

The first task is to co-cluster image and audio samples that represent the same digit. For image, we exploit MNIST, the well-known image dataset of handwritten digits [15]. For audio, we use the Google Speech Commands (GSC) dataset which consists of one-second spoken utterances for the ten digits [16]. Since the GSC dataset contains about 3,000 audio clips per digit, we have randomly selected the same number of images constructing datasets of 30,000 samples for each domain. To reduce the feature size, we feed each sample into a network that generates a vector of size 196.

For the baseline algorithms, we need a matrix where rows represent one dimension and columns represent the other. To construct such matrix, we use the technique introduced by Loeff et al.: each index of the matrix is a dot product of row and column samples [17]. For the sake of consistency, the same set of reduced vectors is used to generate the matrix.

Table 2. Performance of different algorithms for digit and sentiment co-clustering.

Algorithm	Digit			Sentiment		
	NMI	ARI	ACC	NMI	ARI	ACC
Spec-Co	0.561	0.228	0.389	0.641	0.418	0.626
Spec-Bi	0.604	0.266	0.394	0.634	0.415	0.583
Info-CC	0.613	0.282	0.402	0.621	0.367	0.503
SA-CC	0.681	0.396	0.512	0.627	0.407	0.577
CoClus	0.427	0.077	0.103	0.513	0.247	0.311
Combiner	0.807	0.894	0.917	0.413	0.471	0.685
AutoEncoder	0.805	0.871	**0.922**	0.469	0.544	**0.688**

As described in the first column of Table 2, the highest ACC reported from the standard algorithms is only 51.2%. On the other hands, both variants of CI-GAN achieve much higher ACC demonstrating their robustness; *Combiner* reports 91.7% and *AutoEncoder* reports 92.2%. This indicates that CI-GAN is capable of identifying the image and audio of the same digit without any explicit information about which digit each sample represents.

6.2. Sentiment Co-clustering

Next, we co-cluster images and texts of the same sentiment (Figure 4 illustrates sample co-clusters). The two datasets are Facial Expression Research Group Database (FERG-DB) [18] and 2018 SemEval emotion classification dataset [19]. FERG-DB consists of stylized characters with annotated facial expressions and SemEval dataset contains tweets of different emotion classes. Since SemEval dataset is unbalanced and small in size compared to FERG-DB, we have selected five classes that have the most number of samples: "anger", "disgust", "fear", "joy", and "sadness". We then randomly select 500 tweets and images for each class to construct a dataset of 2,500 samples for each domain. The size of each feature is again reduced to 196 and the same technique is used to generate the matrix of two domains.

The second column of Table 2 summarizes our finding. *Spec-Co* reports the best performance among the standard algorithms with the ACC of 62.6%. CI-GAN models again demonstrate their superior flexibility but the differences are not as huge as in the former experiment: 68.5% and 68.8% from *Combiner* and *AutoEncoder*, respectively.

6.3. Domain-specific Clustering Performance

In this section, we attempt to understand how CI-GAN is different from the naïve clustering approach: two independent InfoGAN. Based on Table 3, it is clear that the co-clustering performance are closely related to the domain-specific clustering performance; the former scores are high when the latter scores are high. This is expected since we construct the final co-clustering labels from per-domain clustering labels.

Table 3. Domain-specific clustering accuracy of CI-GAN for digit co-clustering and sentiment co-clustering.

Architecture	Domain	ACC	Domain	ACC
Two Independent InfoGAN	MNIST	0.947	FERG-DB	0.804
	GSC	0.884	SemEval	0.404
	co-clustering	**0.941**	co-clustering	0.556
Combiner	MNIST	0.924	FERG-DB	0.808
	GSC	0.799	SemEval	0.542
	co-clustering	0.917	co-clustering	0.685
AutoEncoder	MNIST	0.938	FERG-DB	0.947
	GSC	0.868	SemEval	0.519
	co-clustering	0.922	co-clustering	**0.688**

For the digit co-clustering, the naïve approach reports higher ACC than CI-GAN. On the other hands, the opposite behavior is observed from the sentiment co-clustering. This indicates that the information sharing of CI-GAN can affect the co-clustering performance in a negative way when the boundaries among the clusters are clear.

Analyzing sentimental information is known to be much harder than distinguishing digits as the boundaries are not as explicit. Consequently, CI-GAN outperforms the naïve approach reinforcing our finding from the two `Mixed-`$\star$ experiments; CI-GAN models are better at deciphering the domain interplay structures in the correlation data space.

7. CONCLUSION

In this work, we introduce CI-GAN, co-clustering based on generative modeling. From our thorough experiments, we demonstrate that generative modeling with augmented mutual information can be a powerful tool for co-clustering data of different types. Future research can explore other variations of mutual information on CI-GAN architecture.

8. REFERENCES

[1] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic co-clustering," in *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 2003.

[2] M. Ailem, F. Role, and M. Nadif, "Co-clustering document-term matrices by direct maximization of graph modularity," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, 2015.

[3] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein, "Spectral biclustering of microarray data: Coclustering genes and conditions," *Genome research*, vol. 13, 2003.

[4] L. Labiod and M. Nadif, "Co-clustering for binary and categorical data with maximum modularity," in *IEEE International Conference on Data Mining*, 2011.

[5] M. Ailem, F. Role, and M. Nadif, "Graph modularity maximization as an effective method for co-clustering text data," *Knowledge-Based Systems*, vol. 109, 2016.

[6] D. Xu, C. Wei, B. Zong, J. Ni, D. Song, W. Yu, Y. Chen, H. Chen, and X. Zhang, "Deep co-clustering," in *Proceedings of the 2019 SIAM International Conference on Data Mining*, 2019.

[7] M. Erxue, G. Xifeng, Z. Gen L. Qiang, C. Jianjing, and L. Jun, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, 2018.

[8] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in neural information processing systems*, 2016.

[9] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *arXiv:1605.09782*, 2016.

[10] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, "ClusterGAN: Latent space clustering in generative adversarial networks," in *The Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014.

[12] F. Role, S. Morbieu, and M. Nadif, "Coclust: A python package for co-clustering," 2018.

[13] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001.

[14] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, 1955.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, 1998.

[16] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv:1804.03209*, 2018.

[17] N. Loeff, C. O. Alm, and D. A. Forsyth, "Discriminating image senses by clustering with multimodal features," in *Proceedings of the COLING/ACL 2006 main conference poster sessions*, 2006.

[18] D. Aneja, A. Colburn, G. Faigin, L. Shapiro, and B. Mones, "Modeling stylized character expressions via deep learning," in *Asian conference on computer vision*, 2016.

[19] S. Mohammad, F. Bravo-Marquez, M. Salameh, and S. Kiritchenko, "SemEval-2018 task 1: Affect in tweets," in *Proceedings of the 12th international workshop on semantic evaluation*, 2018.

CI-GAN : CO-CLUSTERING BY INFORMATION MAXIMIZING GENERATIVE ADVERSARIAL NETWORKS — APPENDIX

1. VARIATIONAL MUTUAL INFORMATION MAXIMIZATION FOR *AUTOENCODER*

1.1. InfoGAN

To understand the objective of *AutoEncoder*, we need to understand how InfoGAN exploits variational mutual information maximization [1] to overcome the difficulties in maximizing I of the following objective (H represents entropy).

$$\min_G \max_D V(D,G) - I(c; G(z,c))$$
$$I(c; G(z,c)) = H(c) - H(c|G(z,c))$$

Directly maximizing I is found to be difficult as the derivation involves a posterior distribution $P(c|\overline{x})$ where $\overline{x}$ represent the samples generated by $G(z,c)$. Therefore, Chen et al. exploit variational mutual information maximization and optimize the lower bound of $I(c; G(z,c))$, L_I, instead. The key idea is to use an auxiliary distribution $Q(c|\overline{x})$ to approximate $P(c|\overline{x})$ [2].

$$L_I(G,Q) = \mathbb{E}_{c\sim P(c),\overline{x}\sim G(z,c)}[\log Q(c|\overline{x})] + H(c)$$

Since the lower bound becomes tight as $Q(c|\overline{x})$ approaches the true distribution $P(c|\overline{x})$, the objective becomes the following.

$$\min_{G,Q} \max_D V(D,G) - L_I(G,Q)$$

1.2. *AutoEncoder*

As previously described in Section 3, we start from the following objective.

$$\min_{G,Q} \max_D V(D_r,G_r) + V(D_c,G_c) - I(x; Q(G(z,c)))$$
$$I(x; Q(G(z,c))) = I(x_r, x_c; Q_r(G_r(z_r,c_r)), Q_c(G_c(z_c,c_c)))$$

Though $I(x; Q(G(z,c)))$ look similar to $I(c; G(z,c))$ of InfoGAN, directly applying variational mutual information maximization to derive the lower bound is not possible due to the nested posterior distributions and the assumption that P_G has the same distribution as P_{data}. Therefore, we first derive $L_{\overline{x}}$, the lower bound for $I(\overline{x}; Q(G(z,c)))$, and exploit the similarity between $I(\overline{x}; Q(G(z,c)))$ and $I(x; Q(G(z,c)))$ to derive L_x, the lower bound for $I(x; Q(G(z,c)))$.

Before we dive into the detailed derivation, we accentuate *Lemma 5.1* which is originally introduced and proven by Chen et al. [2]. In this section, we replace x' with $\widehat{x}$ to be consistent with the notations used in this work.

Lemma 5.1 *For random variables X,Y and function $f(x,y)$ under suitable regularity conditions:* $\mathbb{E}_{x\sim X, y\sim Y|x}[f(x,y)] = \mathbb{E}_{x\sim X, y\sim Y|x, \widehat{x}\sim X|y}[f(\widehat{x},y)]$

$L_{\overline{x}}$: Lower bound for $I(\overline{x}; Q(G(z,c)))$

Along with $P(c|\overline{x})$, the maximization of $I(\overline{x}; Q(G(z,c)))$ involves another posterior distribution $P(\overline{x}|c')$. Similar to how $Q(c|x)$ is used to approximate $P(c|x)$, we introduce another distribution $R(\overline{x}|c')$ for approximating $P(\overline{x}|c')$.

$$\begin{aligned}
I(\overline{x}; Q(G(z,c))) &= H(\overline{x}) - H(\overline{x}|Q(G(z,c))) && \text{(by the definition of } I\text{)} \\
&= H(\overline{x}) + \mathbb{E}_{c'\sim Q(G(z,c))}[\mathbb{E}_{\widehat{x}\sim P(\overline{x}|c')}[\log P(\widehat{x}|c')]] \\
&= H(\overline{x}) + \mathbb{E}_{c'\sim Q(\overline{x})}[\mathbb{E}_{\widehat{x}\sim P(\overline{x}|c')}[\log P(\widehat{x}|c')]] && (\overline{x} = G(z,c)) \\
&\geq H(\overline{x}) + \mathbb{E}_{c'\sim Q(\overline{x})}[\underbrace{D_{KL}(P(\cdot|c') \parallel R(\cdot|c'))}_{\geq 0} + \mathbb{E}_{\widehat{x}\sim P(\overline{x}|c')}[\log R(\widehat{x}|c')]]
\end{aligned}$$

$$
\begin{aligned}
&= H(\overline{x}) + \mathbb{E}_{c' \sim Q(\overline{x})}[\mathbb{E}_{\widehat{x} \sim P(\overline{x}|c')}[\log R(\widehat{x}|c')]] \\
&\qquad\qquad (R(\overline{x}|c') \text{ is an approximation for } P(\overline{x}|c')) \\
&= H(\overline{x}) + \mathbb{E}_{c' \sim Q(\overline{x}), \widehat{x} \sim P(\overline{x}|c')}[\log R(\widehat{x}|c')] \\
&= H(\overline{x}) + \mathbb{E}_{c' \sim P_Q(\overline{x}), \widehat{x} \sim P(\overline{x}|c')}[\log R(\widehat{x}|c')] \\
&= H(\overline{x}) + \mathbb{E}_{\overline{x} \sim P(\overline{x}), c' \sim P_Q(\overline{x}), \widehat{x} \sim P(\overline{x}|c')}[\log R(\widehat{x}|c')] \\
&= H(\overline{x}) + \mathbb{E}_{\overline{x} \sim P(\overline{x}), c' \sim P_Q(\overline{x})}[\log R(\overline{x}|c')] && \text{(by } \textit{Lemma 5.1}\text{)} \\
&= H(\overline{x}) + \mathbb{E}_{\overline{x} \sim P(\overline{x}), c' \sim Q(\overline{x})}[\log R(\overline{x}|c')] \\
&= H(\overline{x}) + \mathbb{E}_{\overline{x} \sim P(\overline{x}), c' \sim Q(G(z,c))}[\log R(\overline{x}|c')] && (\overline{x} = G(z,c)) \\
&= L_{\overline{x}}(G, Q, R)
\end{aligned}
$$

Overall, as the distributions Q and R approach the true posterior distribution, the lower bound becomes tight increasing the mutual information $I(\overline{x}; Q(G(z,c)))$.

L_x : Lower bound for $I(x; Q(G(z,c)))$

To derive L_x, we first assume that P_G has the same distribution as P_{data} ($x = G(z,c)$). Also we leverage an auxiliary distribution $R(x|c')$ to approximate $P(x|c')$. Then, L_x can be derived in almost the same way.

$$
\begin{aligned}
I(x; Q(G(z,c))) &= H(x) - H(x|Q(G(z,c))) && \text{(by the definition of } I\text{)} \\
&= H(x) + \mathbb{E}_{c' \sim Q(G(z,c))}[\mathbb{E}_{\widehat{x} \sim P(x|c')}[\log P(\widehat{x}|c')]] \\
&= H(x) + \mathbb{E}_{c' \sim Q(x)}[\mathbb{E}_{\widehat{x} \sim P(x|c')}[\log P(\widehat{x}|c')]] && (x = G(z,c)) \\
&\geq H(x) + \mathbb{E}_{c' \sim Q(x)}[\underbrace{D_{KL}(P(\cdot|c') \parallel R(\cdot|c'))}_{\geq 0} + \mathbb{E}_{\widehat{x} \sim P(x|c')}[\log R(\widehat{x}|c')]] \\
&= H(x) + \mathbb{E}_{c' \sim Q(x)}[\mathbb{E}_{\widehat{x} \sim P(x|c')}[\log R(\widehat{x}|c')]] \\
&\qquad\qquad (R(x|c') \text{ is an approximation for } P(x|c')) \\
&= H(x) + \mathbb{E}_{c' \sim Q(x), \widehat{x} \sim P(x|c')}[\log R(\widehat{x}|c')] \\
&= H(x) + \mathbb{E}_{c' \sim P_Q(x), \widehat{x} \sim P(x|c')}[\log R(\widehat{x}|c')] \\
&= H(x) + \mathbb{E}_{x \sim P(x), c' \sim P_Q(x), \widehat{x} \sim P(x|c')}[\log R(\widehat{x}|c')] \\
&= H(x) + \mathbb{E}_{x \sim P(x), c' \sim P_Q(x)}[\log R(x|c')] && \text{(by } \textit{Lemma 5.1}\text{)} \\
&= H(x) + \mathbb{E}_{x \sim P(x), c' \sim Q(x)}[\log R(x|c')] \\
&= H(x) + \mathbb{E}_{x \sim P(x), c' \sim Q(G(z,c))}[\log R(x|c')] && (x = G(z,c)) \\
&= L_x(G, Q, R)
\end{aligned}
$$

Given the two lower bounds L_x and $L_{\overline{x}}$, the differences can be summarized as follow:

1. L_x has a constant term $H(x)$ while $L_{\overline{x}}$ has $H(\overline{x})$
2. Generator for L_x has a distribution of P_{data} while the one for $L_{\overline{x}}$ has $P_{G(z,c)}$
3. The distribution R approximates $P(x|c')$ in the case of L_x while it approximates $P(\overline{x}|c')$ in the case of $L_{\overline{x}}$

Interestingly, the first difference can be ignored as it simply yields that the gap between the two lower bounds is bounded by a constant value. Furthermore, the second difference is minimized indirectly by GAN—throughout the training process, generator learns to produce realistic data; the distribution $P_{G(z,c)}$ becomes the distribution P_{data}. To minimize the difference between $P(x|c')$ and $P(\overline{x}|c')$, *AutoEncoder* exploits the network R which reconstructs the original pair of row and column samples from the two independent cluster labels c'_r and c'_c.

Altogether, the objective of *AutoEncoder* can be written as follows with the architecture in Figure 1.

$$
\min_{G,Q} \max_{D} V(D_r, G_r) + V(D_c, G_c) - L_I(G_r, Q_r) - L_I(G_c, Q_c) - L_x(G, R)
$$

2. MODEL ARCHITECTURE

As illustrated in Figure 1, *Combiner* and *AutoEncoder* share the same architectures for G, D and Q in order to ensure coherence. First of all, the size of z is set to 50 and the input data is assumed to have 196 features (14×14). The size of c depends on the dataset as it must be equal to the number of clusters.

Given a random variable z and a random encoding vector c, G first applies a fully connected (FC) layer and a batch normalization (BN) layer to generate a tensor of size $128 \times 7 \times 7$. We then upscale the tensor by factor of 2 and apply a 2D convolutional (CONV) layer reducing the number of channels to 64. With a stride of 1 for both dimensions, a padding of 1 on all sides, and a kernel of size 3×3, we then obtain a tensor of size $64 \times 14 \times 14$. Next, the tensor is fed into another BN layer with a rectified linear unit (ReLU) activation. The final layer of G is a 2D CONV layer with `tanh` activation.

D and Q share a single network for feature representation; we repeatably apply a group of layers which consists of a 2D CONV layer with ReLU activation, a dropout layer, and a BN layer. The stride of each CONV layer is set to 2 and the number of channels is reduced by factor of 2 starting from 16. Therefore, the final tensor has a size of $64 \times 2 \times 2$. D and Q then apply different FC layers to evaluate authenticity and to reconstruct the initial encoding vector c, respectively.

While *Combiner* has an additional FC layer for Q_{co} that reconstructs d, *AutoEncoder* has R which reconstructs the original pairs from encoding pairs. The network architecture of R is same as G except that their input and output tensor have different sizes; the input for R is the concatenation of c'_r and c'_c and the output is the concatenation of row and column samples.

3. HYPERPARAMETER TUNING FOR SYNTHETIC DATASETS

Since CI-GAN consists of multiple networks, we have applied grid search to find the best hyperparameter setting for each model. For *Combiner*, every component is trained with the learning rate of 0.00005. Q_{co} is trained once for every three epochs of row and column training. The quality of the generated co-clusters is found to be better when G is updated along with Q_{co}. On the other hand, *AutoEncoder* produces the best result with the learning rate of 0.0001. Training G along with R is found to be unnecessary when R is trained once for every five epochs of row and column training.

4. SAMPLE CO-CLUSTERS ON SYNTHETIC DATASETS

Figure 5∼9 are co-clusters generated from one of the experiments on synthetic datasets. Each entries are colored based on `gist_ncar_r` colormap of Matplotlib and the two dimensions are grouped based on their labels for better interpretation.

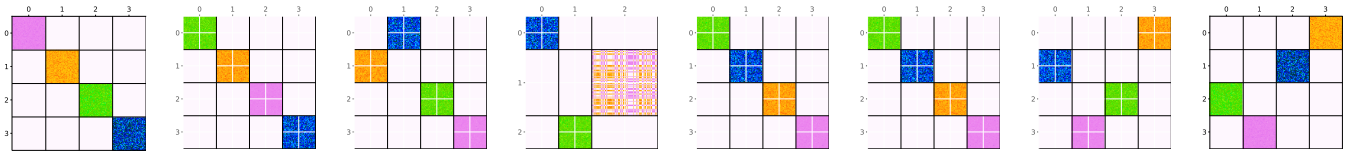

Fig. 5. Co-clusters and accuracy on `Diagonal` dataset. From left to right: Original, *Spec-Co* (1.000), *Spec-Bi* (1.000), *Info-CC* (0.579), *SA-CC* (1.000), *CoClus* (0.563), *Combiner* (1.000), *AutoEncoder* (1.000).

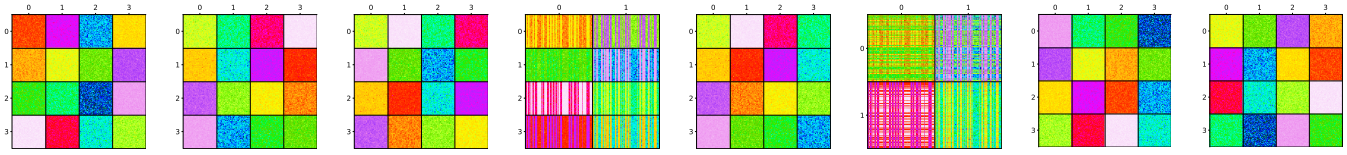

Fig. 6. Co-clusters and accuracy on `Checker-Shuffled` dataset. From left to right: Original, *Spec-Co* (1.000), *Spec-Bi* (1.000), *Info-CC* (0.636), *SA-CC* (1.000), *CoClus* (0.333), *Combiner* (1.000), *AutoEncoder* (1.000).

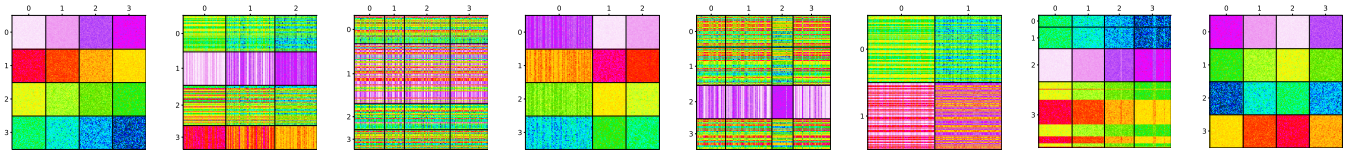

Fig. 7. Co-clusters and accuracy on `Checker-Ordered` dataset. From left to right: Original, *Spec-Co* (0.255), *Spec-Bi* (0.037), *Info-CC* (0.784), *SA-CC* (0.132), *CoClus* (0.333), *Combiner* (0.640), *AutoEncoder* (1.000).

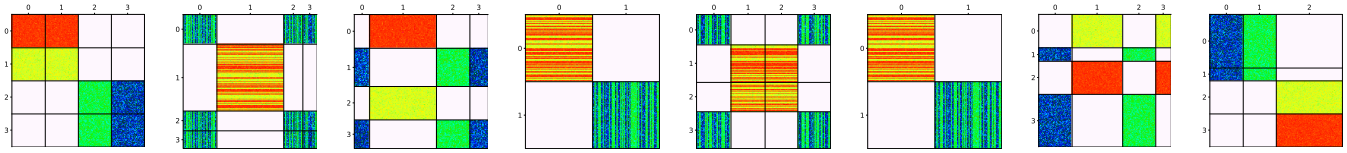

Fig. 8. Co-clusters and accuracy on `Mixed-Identical` dataset. From left to right: Original, *Spec-Co* (0.266), *Spec-Bi* (0.519), *Info-CC* (0.333), *SA-CC* (0.206), *CoClus* (0.333), *Combiner* (0.604), *AutoEncoder* (0.574).

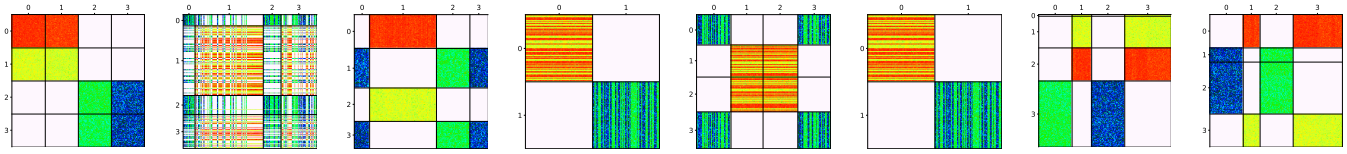

Fig. 9. Co-clusters and accuracy on `Mixed-Similar` dataset. From left to right: Original, *Spec-Co* (0.057), *Spec-Bi* (0.519), *Info-CC* (0.333), *SA-CC* (0.203), *CoClus* (0.333), *Combiner* (0.601), *AutoEncoder* (0.650).

5. REFERENCES

[1] D. Barber and F. Agakov, "The IM algorithm : A variational approach to information maximization," *Advances in neural information processing systems*, vol. 16, 2004.

[2] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in neural information processing systems*, 2016.