A Graph Autoencoder Approach to Crowdsourcing

Panagiotis A. Traganitis
Dept. of Electrical and Computer Engineering
Michigan State University
East Lansing, MI, USA
traganit@msu.edu

Charilaos I. Kanatsoulis
Dept. of Computer Science
Stanford University
Palo Alto, CA, USA
charilaos@cs.stanford.edu

2024 IEEE 13th Sensor Array and Multichannel Signal Processing Workshop (SAM) | 979-8-3503-4481-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/SAM60225.2024.10636521

***Abstract*—Crowdsourcing deals with combining and aggregating labels from crowds of annotators of unknown reliability. While most works on label aggregation operate under the assumption of independent and identically distributed data, the present work introduces an algorithm that operates under known data dependencies or correlations. To exploit these dependencies, a novel graph autoencoder-based algorithm is developed that fuses annotator labels for crowdsourced classification tasks. Numerical tests on real data showcase the potential of the proposed approach.**

***Index Terms*—Crowdsourcing, Weak supervision, Graph autoencoder, Graph neural networks, Classification**

I. Introduction

The proliferation of exceedingly large and deep neural networks and foundation models such as large language models, calls for massive labeled datasets for their training and fine-tuning. However, curating such datasets can be an expensive and labor-intensive task, especially if the data labeling process requires specialized knowledge. Crowdsourcing has recently emerged as a potential solution to this challenge. At its core, crowdsourcing utilizes crowds of human annotators to accomplish various tasks [1]. For dataset curation, crowdsourcing assigns data to various annotators and combines the labels they provide. Specifically, for classification tasks the annotators provide categorical responses for each datum. Nevertheless, human annotators may be unreliable and their labels erroneous or noisy. Thus, a key challenge in crowdsourcing is appropriately aggregating the potentially noisy labels provided by the annotators.

Arguably, the simplest method for aggregating labels is the majority vote rule. Under this rule, the label most annotators agree on is the one assigned to each datum. While simple, this method implicitly assumes that all annotators are of equal ability, and thus can yield suboptimal performance in various tasks. Seeking to improve the performance of majority voting, alternative methods use parametric models of annotator behavior, and typically assume that annotators are conditionally independent. The so-called Dawid and Skene model uses the expectation-maximization algorithm to jointly estimate annotator responses and aggregate noisy labels [2]. Building on the Dawid and Skene model and ideas from moment matching a plethora of algorithms that enjoy improved performance have been developed [3]–[7]. Methods based on deep neural networks have been proposed in [8] and [9], with the former using a restricted Boltzmann machine and the latter advocating an autoencoder structure to fuse annotator responses. Aiming to relax the conditional independence assumption, [10]–[13] propose models that take into account dependencies between annotators. Similar approaches are also advocated to identify spammer or adversarial annotators from their responses [14]–[17].

Most of the aforementioned algorithms typically operate under the assumption that the data are independent and identically distributed (i.i.d.). However, this assumption may not necessarily be true. For instance, time series data or text data, such as the ones arising in natural language processing, exhibit temporal or sequence dependencies. In addition, data arising from social or communication networks exhibit correlations or dependencies associated with the network structure. Extensions of the Dawid and Skene model for sequential and networked data have been proposed in [18]–[20]. When data features are available, recent works advocate so-called end-to-end (E2E) models. These models seek to simultaneously train a classifier and infer annotator reliability, using the available data features and noisy annotator responses. Such E2E models typically utilize Gaussian Processes [19], [21] or deep neural networks [22], [23] as classifiers.

In this work, we develop a novel approach to combining annotator responses under data dependencies, encoded by a graph. Such pairwise dependencies may be known a priori or can be estimated from auxiliary data related to the crowdsourcing task.

For the crowdsourcing task we propose to employ a graph autoencoder (GAE) architecture [24] that does not require any data features. The encoder part of GAE is a cascade of message-passing Graph Neural Network (GNN) layers, while the decoder reconstructs the graph from the dot product of data pairs. Theoretical and empirical evidence suggest that GNN encoders learn powerful node embeddings [25] that capture the critical information from both the graph and the annotators [26], [27]. In fact the expressive power of GNNs along with the effectiveness of autoencoder architectures to learn informative and concise representation is a key to the success of our proposed method.

The rest of this paper is organized as follows: Section II introduces the problem statement and relevant preliminaries, while Section III presents our proposed algorithm. Numerical

The work of P. Traganitis was supported by NSF grant 2312546.

tests with real data are presented in Section IV and finally concluding remarks and future research directions are given in Section V.

Notation. Unless otherwise noted, lowercase bold letters, $\mathbf{x}$, denote column vectors, uppercase bold letters, $\mathbf{X}$, represent matrices, and calligraphic uppercase letters, $\mathcal{X}$, stand for sets. The (i,j)th entry of matrix $\mathbf{X}$ is denoted by $[\mathbf{X}]_{ij}$. Pr denotes probability, or the probability mass function; $\sim$ denotes "distributed as;" $^\top$ represents transpose; and $\mathbb{1}(\mathcal{A})$ is the indicator function for the event $\mathcal{A}$, that takes value 1 when $\mathcal{A}$ occurs, and 0 otherwise.

II. Problem statement and preliminaries

Consider a dataset consisting of N data $\mathcal{X} := \{\boldsymbol{x}_n\}_{n=1}^N$, where each datum can belong to one of K possible classes. The class each datum is assigned to is encoded via the labels $\{y_n\}_{n=1}^N$, e.g. $y_n = k$ if $\boldsymbol{x}_n$ belongs to class k. Class priors are collected in $\boldsymbol{\pi} := [\Pr(y_n = 1), \ldots, \Pr(y_n = K)]^\top$. M annotators or workers observe $\mathcal{X}$, or subsets of it, and provide estimates of labels, where $\check{y}_n^{(m)} \in \{1, \ldots, K\}$ denotes the label assigned to the n-th datum by annotator m. When an annotator does not provide a response for $\boldsymbol{x}_n$, we encode this by setting $\check{y}_n^{(m)} = 0$. Note that, in general, there is no guarantee that $\check{y}_n^{(m)} = y_n$. Given only the annotator responses, collected in the $N \times M$ matrix $\check{\mathbf{Y}}$, where $[\check{\mathbf{Y}}]_{n,m} = \check{y}_n^{(m)}$, the *crowdsourced classification* task involves fusing annotator responses to estimate the ground-truth labels, namely $\{\hat{y}_n\}_{n=1}^N$. Note that this is an *unsupervised* problem, as we do not have access to the ground-truth labels $\{y_n\}_{n=1}^N$.

A common assumption that most algorithms tackling the label aggregation task adhere to, is that data are independent and identically distributed according to some unknown distribution. However, this i.i.d. assumption is often violated. Suppose that the correlation or dependence between data in $\mathcal{X}$ is known, and is encoded in a *known* graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Here, $\mathcal{V}$ is the so-called vertex or node set, whose cardinality is the total number of data, $|\mathcal{V}| = N$, and $\mathcal{E}$ is the edge or link set. $\mathcal{G}$ may be known from the physics of the problem, or can be estimated from $\mathcal{X}$ using topology inference techniques [28]. Alternatively, such pairwise dependencies can be easily estimated if the annotators provide information regarding the similarity of different data, alongside their label estimates. Finally, a graph can be estimated via the similarity between annotator responses. A graph can be compactly represented by its $N \times N$ adjacency matrix $\mathbf{A}$, with entries $[\mathbf{A}]_{i,j} = 1$ if $(i,j) \in \mathcal{E}$, that is, if a link exists between nodes i and j, and $[\mathbf{A}] = 0$ otherwise. The next section will develop an algorithm that considers both annotator responses and the graph $\mathcal{G}$, to fuse the labels for each datum.

III. Graph autoencoder for crowdsourcing

For each datum n, the annotator responses $\{\check{y}_n^{(m)}\}_{m=1}^M$, despite potential errors, contain information regarding the true label y_n. Thus, the crowdsourcing problem can be considered as the task of condensing or "compressing" the per datum responses into a single variable, that should be close, in some sense, to the true label y_n. Recent works have shown that this "compression" is possible using an autoencoder structure [9]. An autoencoder is an algorithm that consists of two neural networks: an encoder network, that compresses its' input into latent variables, and a decoder network that attempts to reconstruct the input from the latent variables. In the setup of this paper, and taking into account the available graph $\mathcal{G}$, that captures relationships between data, the "compression" of annotator responses can be realized using a GAE. While conceptually the same as an ordinary autoencoder, the encoder and decoder networks in a GAE, are tailored to the graph $\mathcal{G}$. The encoder network constructs latent representations of the inputs by taking into account $\mathcal{G}$, and the decoder seeks to reconstruct $\mathcal{G}$, i.e. the adjacency $\mathbf{A}$, from the aforementioned latent representations.

Prior to using a GAE-based algorithm, all scalar annotator responses in $\check{\mathbf{Y}}$ are converted into so-called one-hot vectors, i.e. $\check{y}_n^{(m)} = k$ is converted to a canonical $K \times 1$ vector $\check{\mathbf{y}}_n^{(m)}$ that has a 1 in it's k-th entry, and zeroes elsewhere. Then per datum n, all annotator responses can be stacked into a $MK \times 1$ vector $\check{\mathbf{y}}_n$, i.e.

$$\check{\mathbf{y}}_n = [\check{\mathbf{y}}_n^{(1)\top}, \check{\mathbf{y}}_n^{(2)\top}, \ldots, \check{\mathbf{y}}_n^{(M)\top}]^\top. \tag{1}$$

The encoder has to "compress" the per datum vector $\check{\mathbf{y}}_n$ into a $K \times 1$ vector $\mathbf{z}_n$ that encodes the label y_n. The decoder network then has to reconstruct the graph $\mathcal{G}$ from $\mathbf{z}_n$. The ensuing subsections will describe in detail the structure of the proposed graph autoencoder for crowdsourcing.

A. Message-passing encoder

To jointly process the information of each annotator along with the relational structure of the data the encoder layer is designed using graph message-passing operations. In particular, a graph neural network (GNN) encoder is employed, which consists of a cascade of layers defined by the following recursive equation:

$$\mathbf{x}_n^{(l)} = g^{(l-1)}\left(f^{(l-1)}\left(\left\{\mathbf{x}_u^{(l-1)} : u \in \mathcal{N}(n)\right\}\right)\right), \tag{2}$$

where $\mathcal{N}(n)$ is the neighborhood of vertex n, i.e., $u \in \mathcal{N}(n)$ if and only if $(u,n) \in \mathcal{E}$. The function $f^{(l)}$ aggregates information from the multiset of neighboring signals, whereas $g^{(l)}$ performs a transformation of the aggregated signals. The proposed encoder uses an architecture similar to the graph convolutional network (GCN) [29]. In particular, for each layer l, $f^{(l)}$ is the elementwise mean pooling and $g^{(l)}$ is an affine transformation followed by a sigmoid function. In other words $g^{(l)}$ is a multi-layer perceptron (MLP) with sigmoid activation. Overall (2) takes the form:

$$\mathbf{X}^{(l)} = \sigma\left(\mathbf{H}^{(l)}\mathbf{X}^{(l-1)}\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right), \tag{3}$$

where $\mathbf{X}^{(l)}$ is a matrix whose n-th column is $\mathbf{x}_n^{(l)}$, $\mathbf{D}$ is a diagonal matrix with elements the degrees of each node (datum), $\mathbf{H}^{(l)}$ is a $F_l \times F_{l-1}$ matrix containing the trainable parameters of layer l, and σ is the sigmoid function, applied

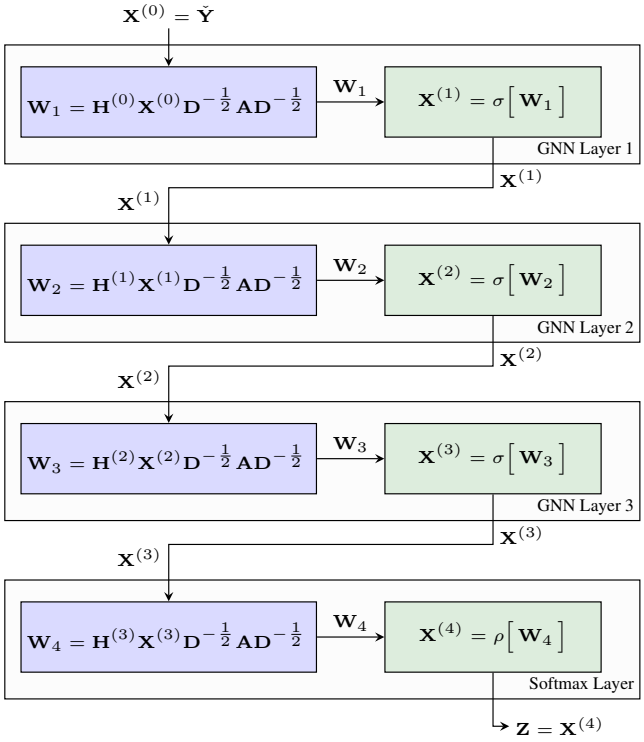

Fig. 1: Proposed Encoder Architecture

elementwise. The per datum input to the GNN is the vector of annotator responses, i.e., $\mathbf{x}_n^{(0)} = \check{\mathbf{y}}_n$. The encoder used in this paper consists of three layers, as described in (2), with $(F_0, F_1, F_2, F_3) = (MK, 2K, 2K, K)$. The embedding at the output of the encoder is refined by a softmax layer to ensure that the output $\mathbf{z}$ lies in a probability simplex. As a result, the per datum output is a $K \times 1$ vector $\mathbf{z}_n$, where $\mathbf{z}_n[k]$ represents the probability of datum n to belong to class k. Specifically, the i-th entry of $\mathbf{z}$ is

$$[\mathbf{z}]_i = \frac{e^{[\mathbf{x}^{(3)}]_i}}{\sum_{j=1}^{K} e^{[\mathbf{x}^{(3)}]_j}}. \tag{4}$$

The proposed encoder architecture is illustrated in Fig. 1.

B. Decoder network

The autoencoder architecture considered here reconstructs the graph instead of the annotator responses. The intuition is that in a graph with community structure, the most informative node feature would be the node label. In other words, given a categorical variable for each node, the categorical variable that would reconstruct the graph with the lowest possible error would be the label of the node (datum).

Similar to a standard graph autoencoder, the decoder network here consists of an "inner product decoder", whose goal is to reconstruct the adjacency matrix $\mathbf{A}$ of the graph $\mathcal{G}$. Given two latent variables $\mathbf{z}_i, \mathbf{z}_j$ corresponding to data (nodes) i and j, the probability of (i, j)-th edge of the graph is estimated as

$$\Pr([\hat{\mathbf{A}}]_{i,j} = 1) = \sigma(\mathbf{z}_i \mathbf{z}_j^\top). \tag{5}$$

Algorithm 1 Crowd Graph Autoencoder (CrowdGAE)

1: **Input:** Annotator responses $\{\check{y}_n^{(m)}\}_{n=1,m=1}^{N,M}$, K, graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, regularization constant $\lambda > 0$
2: **Output:** Estimated labels $\{\hat{y}_n\}_{n=1}^{N}$
3: Convert annotator responses per datum n into vector format as (1).
4: Train autoencoder via (7)
5: Obtain latent representations $\mathbf{z}_n$ for $n = 1, \ldots, N$.
6: Estimate label $\hat{y}_n$ via (10) for $n = 1, \ldots, N$

Dataset	N	M	K
Citeseer	3,312	10	7
Cora	2,708	10	10
Pubmed	19,717	10	3
Music Genre	700	44	10
Sentence Polarity	5,000	203	2
Bluebird	108	39	2

TABLE I: Dataset properties

Letting $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N]$ the overall probability of the reconstructed adjacency matrix is given as

$$\Pr(\hat{\mathbf{A}}) = \sigma(\mathbf{Z}\mathbf{Z}^\top). \tag{6}$$

C. Training

The graph autoencoder treats the process of finding low-dimensional embeddings in the context of link prediction, or binary classification. The unknown parameters of the model, collected in $\mathbf{\Theta} = \{\mathbf{H}^{(0)}, \mathbf{H}^{(1)}, \mathbf{H}^{(2)}\}$, are estimated by solving the following optimization problem

$$\min_{\mathbf{\Theta}} H(\mathbf{A}, \Pr(\hat{\mathbf{A}})) + \lambda R(\check{\mathbf{Y}}, \mathbf{Z}) \tag{7}$$

where $H(\mathbf{A}, \Pr(\hat{\mathbf{A}}))$ is the binary cross-entropy [30] between the edges in $\mathbf{A}$ and the estimated edges in $\hat{\mathbf{A}}$, $R(\check{\mathbf{Y}}, \mathbf{Z})$ is a regularization term, and $\lambda > 0$. The cross-entropy term promotes accurate reconstruction of the graph, while the regularization term $R(\check{\mathbf{Y}}, \mathbf{Z})$ is used to ensure that indices of the latent variables $\mathbf{z}$ maintain the proper ordering. Specifically the regularization term is

$$R(\check{\mathbf{Y}}, \mathbf{Z}) = \sum_{n=1}^{N} D_{\mathrm{KL}}(\mathbf{z}_n || \hat{\boldsymbol{\pi}}_n), \tag{8}$$

where $D_{\mathrm{KL}}(p||q)$ is the Kullback-Leibler divergence between two discrete distributions p and q [30]. Thus the regularization term forces the latent variables $\mathbf{z}_n$ to be "close" to some prior distribution $\hat{\boldsymbol{\pi}}_n$. Here, the distribution $\hat{\boldsymbol{\pi}}_n$ per datum can be directly obtained by averaging annotator responses, i.e.

$$\hat{\boldsymbol{\pi}}_n = \frac{1}{M_n} \sum_{m=1}^{M} \check{\mathbf{y}}_n^{(m)} \tag{9}$$

where M_n is the number of annotators that have provided a response for datum n. It is important to note that, the

Algorithm / Dataset	Citeseer	Cora	Pubmed	Music Genre	Sentence Polarity	Bluebird
MV	46.23	42.98	70.73	71.14	88.95	75.92
DS	55.40	58.60	75.12	76	91.48	87.96
LAA	48.422	45.34	72.52	77.57	88.177	90.7
CrowdGAE	62	63.51	77.69	82	95.69	93.51

TABLE II: Accuracy (higher is better) for real data experiments

training process of the graph autoencoder only requires the noisy annotator labels and the graph.

D. Label Inference

After training the graph autoencoder and with the latent variables for all n data $\{\mathbf{z}_n\}_{n=1}^N$ at hand, the final aggregated labels $\{\hat{y}_n\}_{n=1}^N$ can be readily estimated. For datum n, the estimated label $\hat{y}_n$ is given as the index of $\mathbf{z}_n$ that corresponds to the entry of largest magnitude, i.e.

$$\hat{y}_n = \underset{k}{\text{argmax}}[\mathbf{z}_n]_k. \tag{10}$$

The entire label aggregation procedure is tabulated in Alg. 1

IV. Numerical Tests

The performance of the proposed method is evaluated in this section using six real datasets. The proposed algorithm, denoted as *CrowdGAE* is compared with the EM algorithm of Dawid and Skene initialized via majority voting, denoted as *DS*, the majority voting rule, denoted as *MV*, and the label-aware autoencoder algorithm of [9], denoted as *LAA*. Note, all competing alternatives here, do not use any graph information.

The datasets considered are the Cora, Citeseer [31], Pubmed [32], Music genre, Sentence Polarity [33] and Bluebird [34] datasets. Cora, Citeseer, and Pubmed are citation network datasets, for which graphs and data features are provided. However, these datasets do not come with crowdsourced labels. To emulate the effect of a crowd, 10 classification algorithms are trained using the provided features on subsets of the data. Each algorithm takes the role of an annotator. Afterwards, the classification algorithms provide labels for all the data. Music Genre, Sentence Polarity, and Bluebird are crowdsourcing datasets, where human annotators have provided responses for music genre identification, sentiment analysis of movie reviews, and bird identification respectively. For the Music Genre, Sentence Polarity and Bluebird datasets graphs were not provided, and as such had to be created. Graphs for these datasets were created using their ground-truth labels. Specifically, if data i and j belong to the same class, that is $y_i = y_j$, then an edge is drawn between them, $[\mathbf{A}]_{i,j} = 1$, otherwise if $y_i \neq y_j$ $[\mathbf{A}]_{i,j} = 0$. Then entries of $\mathbf{A}$ are subsampled to create the final graph: within class entries are sampled with probability ε_1 and random edges between classes are added with probability ε_2. Finally, 10% of the edges in $\mathbf{A}$ are used during the training of *CrowdGAE*. Properties of all datasets are listed in Tab. I.

Experiments were performed using Python, the PyTorch and PyTorch geometric [35], [36] libraries and results represent averages over 10 independent Monte Carlo runs. In all experiments the parameter λ of *CrowdGAE* (see (7)) is set to $\lambda = 0.1$. The sampling probabilities for the graphs created for the Music Genre, Sentence Polarity and bluebird datasets are $\varepsilon_1 = 0.8$ and $\varepsilon_2 = 0.2$. The figure of merit across all experiments is the average accuracy

$$Acc = 100 * \frac{\sum_{n=1}^N \mathbb{1}(y_n = \hat{y}_n)}{N}.$$

Table II shows the classification accuracy of considered algorithms across all datasets. In all experiments, most methods outperform the majority voting rule. At the same time, the proposed method achieves the highest accuracy among the competing alternatives across all experiments, even with minimal fine-tuning. From the findings of Table II we can derive two main conclusions. First, that graph information can be beneficial to the label aggregation task. The second and most important conclusion is that the proposed CrowdGAE can efficiently process both the graph and the annotators to produce accurate classification results.

V. Conclusions

This contribution presented a graph autoencoder-based method for aggregating labels in crowdsourced classification. The proposed algorithm exhibits competitive performance in multiple real datasets, and showcases the importance of taking data correlation into account when combining annotator responses. Future work will involve developing a variational counterpart to the proposed algorithm, interpretability analysis, as well as alternative architectures.

References

[1] J. Howe, "The rise of crowdsourcing," *Wired Magazine*, vol. 14, no. 6, pp. 1–4, 2006.

[2] A. P. Dawid and A. M. Skene, "Maximum likelihood estimation of observer error-rates using the EM algorithm," *Applied Statistics*, pp. 20–28, 1979.

[3] A. Jaffe, B. Nadler, and Y. Kluger, "Estimating the accuracies of multiple classifiers without labeled data." in *AISTATS*, vol. 2, San Diego, CA, 2015, p. 4.

[4] Y. Zhang, X. Chen, D. Zhou, and M. I. Jordan, "Spectral methods meet EM: A provably optimal algorithm for crowdsourcing," in *Advances in Neural Information Processing Systems*, 2014, pp. 1260–1268.

[5] P. A. Traganitis, A. Pagès-Zamora, and G. B. Giannakis, "Blind multiclass ensemble classification," *IEEE Transactions on Signal Processing*, vol. 66, no. 18, pp. 4737–4752, Sept 2018.

[6] S. Ibrahim, X. Fu, N. Kargas, and K. Huang, "Crowdsourcing via pairwise co-occurrences: Identifiability and algorithms," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d´ Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 7847–7857.

[7] S. Ibrahim and X. Fu, "Crowdsourcing via annotator co-occurrence imputation and provable symmetric nonnegative matrix factorization," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 4544–4554. [Online]. Available: https://proceedings.mlr.press/v139/ibrahim21a.html

[8] U. Shaham, X. Cheng, O. Dror, A. Jaffe, B. Nadler, J. Chang, and Y. Kluger, "A deep learning approach to unsupervised ensemble learning," in *International Conference on Machine Learning*, New York, NY, 2016, pp. 30–39.

[9] L. Yin, J. Han, W. Zhang, and Y. Yu, "Aggregating crowd wisdoms with label-aware autoencoders," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, p. 1325–1331.

[10] A. Jaffe, E. Fetaya, B. Nadler, T. Jiang, and Y. Kluger, "Unsupervised ensemble learning with dependent classifiers," in *Artificial Intelligence and Statistics*, 2016, pp. 351–360.

[11] M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi, "Community-based bayesian aggregation models for crowdsourcing," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 155–164. [Online]. Available: https://doi.org/10.1145/2566486.2567989

[12] P. A. Traganitis and G. B. Giannakis, "Blind multi-class ensemble learning with dependent classifiers," in *Proc. of the 26th European Signal Processing Conference*, Rome, Italy, Sep 2018.

[13] ——, "Identifying dependent annotators in crowdsourcing," in *2022 56th Asilomar Conference on Signals, Systems, and Computers*, 2022, pp. 1276–1280.

[14] P. A. Traganitis and G. B. Giannakis, "Identifying spammers to boost crowdsourced classification," in *46th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[15] S. Jagabathula, L. Subramanian, and A. Venkataraman, "Identifying unreliable and adversarial workers in crowdsourced labeling tasks," *Journal of Machine Learning Research*, vol. 18, no. 93, pp. 1–67, 2017.

[16] Q. Ma and A. Olshevsky, "Adversarial crowdsourcing through robust rank-one matrix completion," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 21 841–21 852.

[17] P. A. Traganitis and G. B. Giannakis, "Detecting adversaries in crowdsourcing," in *2021 IEEE International Conference on Data Mining (ICDM)*, 2021, pp. 1373–1378.

[18] A. T. Nguyen, B. Wallace, J. J. Li, A. Nenkova, and M. Lease, "Aggregating and predicting sequence labels from crowd annotations," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 299–309.

[19] P. Ruiz, P. Morales-Álvarez, R. Molina, and A. Katsaggelos, "Learning from crowds with variational Gaussian processes," *Pattern Recognition*, vol. 88, pp. 298–311, April 2019. [Online]. Available: http://decsai.ugr.es/vip/files/journals/1-s2.0-S0031320318304060-main.pdf

[20] P. A. Traganitis and G. B. Giannakis, "Unsupervised ensemble classification with sequential and networked data," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020.

[21] F. Rodrigues, F. Pereira, and B. Ribeiro, "Gaussian process classification and active learning with multiple annotators," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 32, no. 2. Bejing, China: PMLR, 22–24 Jun 2014, pp. 433–441.

[22] F. Rodrigues and F. Pereira, "Deep learning from crowds," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[23] S. Ibrahim, T. Nguyen, and X. Fu, "Deep learning from crowdsourced labels: Coupled cross-entropy minimization, identifiability, and regularization," in *Proceedings of International Conference on Learning Representations*, 2023.

[24] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016.

[25] C. I. Kanatsoulis and A. Ribeiro, "Graph neural networks are more powerful than we think," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 7550–7554.

[26] ——, "Representation power of graph neural networks: Improved expressivity via algebraic analysis," *arXiv preprint arXiv:2205.09801*, 2022.

[27] C. Kanatsoulis and A. Ribeiro, "Counting graph substructures with graph neural networks," in *The Twelfth International Conference on Learning Representations*, 2023.

[28] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 787–807, 2018.

[29] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2016.

[30] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 2012.

[31] Q. Lu and L. Getoor, "Link-based classification," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ser. ICML'03. AAAI Press, 2003, pp. 496–503. [Online]. Available: http://dl.acm.org/citation.cfm?id=3041838.3041901

[32] G. Namata, B. London, L. Getoor, and B. Huang, "Query-driven active surveying for collective classification," in *Proc. of Workshop on Mining and Learning with Graphs*, 2012.

[33] F. Rodrigues, F. Pereira, and B. Ribeiro, "Learning from multiple annotators: Distinguishing good from random labelers," *Pattern Recognition Letters*, vol. 34, no. 12, pp. 1428–1436, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016786551300202X

[34] P. Welinder, S. Branson, P. Perona, and S. J. Belongie, "The multidimensional wisdom of crowds," in *Advances in Neural Information Processing Systems 23*. Curran Associates, Inc., 2010, pp. 2424–2432.

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[36] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.