Supplementary material for a Unified Non-Negative Matrix Factorization Framework for Semi Supervised Learning on Graphs

Anasua Mitra[*] Priyesh Vijayan[†‡] Srinivasan Parthasarathy[§¶] Balaraman Ravindran[‖**]

Abstract

We propose a Semi-Supervised Learning (SSL) framework, USS-NMF, that allows for explicitly encoding different necessary priors to learn efficient node representations in a graph. USS-NMF specializes in encoding the important yet largely ignored necessary prior for SSL, the cluster assumption. The cluster assumption of SSL requires the existence of well-separated dense regions in a low-dimensional manifold with high label smoothness within each region. USS-NMF encodes this assumption in the form of a proposed semi-supervised cluster invariance constraint, which is a group-level smoothness constraint on nodes. We show that explicitly enforcing this constraint enables learning meaningful node representations from both qualitative (visual) and quantitative standpoints. Specifically, USS-NMF achieves superior performance on semi-supervised node classification and clustering tasks across thirteen datasets from over eight baselines. Also, the learned node embeddings from USS-NMF yield high-quality (well-separated homophilous) clusters in t-SNE visualizations.

1 Background on Graph Embedding

Graph Embedding techniques have become popular for learning representations for different components of the graph like nodes, edges, sub-graphs, and the entire graph. Graph embedding models encode different intrinsic and extrinsic properties of the network as continuous low dimensional vectors. Network representation learning has been realized by a variety of paradigms such as factorization models, graph kernels, skip-gram based models, deep learning models, generative models and hybrid paradigms, etc [2].

The skip-gram based seminal work, Deepwalk [4] and it's variants use k-step random walks to define a k-th order neighborhood. Node2Vec [3] is one exception that uses an informed random walk sampling on nodes based on pre-defined parameters that trades-off between breadth-wise and depth-wise exploration. On the other hand, factorization models have been widely used to encode different network contexts and couple it with different constraints, e.g., Locally Linear Embeddings, Laplacian Eigenmaps, GraphFactorization, ISOMAP, GraRep, HOPE, LINE.

Recently, a few works learn clusterable node embeddings that preserve network community information. MNMF is an NMF based model that learns node embedding by factorizing the proximity matrix and predicts community assignments for these nodes from the embeddings. The community assignments are learned jointly, maximizing the modularity of the graph. ComE [1], and GEMSEC [5] are two other community preserving models that learn node embeddings by skip-gram model. GEMSEC is a k-means based adaption for learning node embeddings that jointly learn clusters centers. GEMSEC learns cluster embeddings along with node embeddings and minimizes the distance between the node's embedding and the nearest cluster mean. ComE, along with minimizing the context prediction loss of skip-gram, also maximizes the log-likelihood of generating node embeddings from multiple GMMs.

There are numerous works on learning unsupervised node embeddings for attributed graphs. However, there are only fewer works that jointly learn node embedding with side information such as attribute and group information. Works that learn semi-supervised node embeddings with label information is further scarce. These Semi-supervised models can learn discriminative node embeddings that can provide superior node classification results. [6]'s MaxMargin Deep Walk (MMDW), jointly solves the factorization of the PPMI matrix of a k-step random walk and the hinge loss for label prediction. Planetoid [7] is another extension of DeepWalk (DW) that also optimizes a cross-entropy loss for label prediction. Planetoid additionally enforces nodes to predict other nodes with similar labels. Planetoid also has a variant for attributed graphs.

[*] Department of CSE, Indian Institute of Technology Guwahati
[†] Department of CS, McGill University & MILA
[‡] Work done during his MS program at IIT Madras
[§] Department of CSE, Ohio State University
[¶] Department of Biomedical Informatics, Ohio State University
[‖] Department of CSE, Indian Institute of Technology Madras
[**] Robert Bosch Centre for Data sciences and AI, IIT Madras

2 Implelementation Details

The details of the hyper-parameters for our model and the baselines are provided below. The hyper-parameter search space for different components in all the models experimented here is tabulated in Table: 1.

Lagrange Multipliers' Range We first provide the details of HP search for the Lagrange multipliers followed by model specific details. For all the matrix factorization baselines, we vary the hyper-parameter values (the respective weightage terms for each component in the objective function) in $[0.1, 0.5, 1.0, 5.0, 10.0]$ except for Wikipedia. In Wikipedia, we found that the network information is far more important than other supervision knowledge. So we varied network co-efficient in the range of $[10000, 1000, 100, 10]$ with other weights in $[0.001, 0.01, 0.1, 1.0]$. We fixed embedding dimension as 128 for all datasets except Blogcatalog, for which the dimension is set as 4096.

DeepWalk & MFDW: For original random-walk based DeepWalk we set the window size to 5. We also have MFDW aka NMF:S in Eqn: 3.2 - the objective function for Matrix Factorized DeepWalk, as we build our model incrementally on top of it.

Max-Margin DeepWalk (MMDW): In this paper [6], a max-margin loss is incorporated in the objective function of MFDW to learn discriminative representations of vertices. It has one important hyper-parameter alpha-bias (η) that induces max-margin loss based bias into the random walk.

NMF:S+Y: We build a variant of MMDW which also incorporates supervised information into node embeddings by jointly optimizing Eqn: 3.2 & 3.3. It works competitively as compared to MMDW.

Planetoid & MF-Planetoid: Planetoid [7] learns an embedding space for nodes by jointly enforcing label and neighborhood similarity. It uses random walks to enforce structural similarity. We derive a matrix-factorized version of Planetoid as an alternative baseline. It enforces matrix E, i.e, train-label similarity on the embedding space U, unlike ours as in Eqn: 3.6 which enforces label similarity on the cluster space.

$$O_{\text{MF-Plan}} = O(NMF : S + Y) + Tr\{U\Delta(E)U^T\} \tag{2.1}$$

MNMF & MNMF+Y: We build one semi-supervised variant of MNMF, viz. MNMF+Y by jointly optimizing its objective function along with Eqn: 3.3. Unlike the original MNMF that factorizes a combination of first-order and cosine similarity based second-order node proximity to learn node representations, here, for the sake of fair comparison, we stick to a combination of first-order and second-order transition probability-based proximity matrix as S, following MMDW [6] as we did for all other comparable methods.

USS-NMF: We used the same range of hyper-parameters as stated in Table: 2 last column, but instead of searching the optimal combination in the entire range (which is cumbersome), we did partial range search in steps. Step by step, 1) network + label information weights, 2) cluster matrix factorization + cluster learning weights + orthogonality constraint, 3) label smoothing + L2 regularization weights, and finally, 4) the clusters k for each dataset was varied with an increment of 2 in the upper range and with a decrement of 1 in the lower range from its actual number of labels q (inclusive).

In the first step, we fixed all other variable values as 1.0, $k = q$ and in later steps, we set already searched parameter values to optimal values found in previous steps, we vary the other variables under consideration. In Table: 2 we have given an effecctive value range for each of the coefficients, applicable for varying sized datasets. We make the following points based on observation,

Labels and label-similarity based clusters gave complementary information to support each other for enhanced prediction (evident from their weight combinations in results). For small graphs, network information was more useful than other information. Also, small graphs tended to be more sensitive to weight combinations with easily imposed cluster orthogonality constraint (see the effective range). But for large graphs, USS-NMF needed more weights to ensure orthogonal clusters being learned. Higher cluster learning weights (in effective range) indicate that indeed cluster information mattered. We found optimal results for clusters same as ground truth labels, very different from labels, multiple optimal clusters - which indicate that a variety of semantically meaningful clusters have been learnt. In effective search, for simplicity, we set clusters as the number of labels, which gave decent results. Except for Pubmed and Wikipedia, we do not witness any fluctuation in L2 regularization weights.

With these observations at hand, we narrowed down the effective hyper-parameter search space such that network has weights (>=1 & <=10), labels & clusters don't overpower network weights, stable values for regularization & orthogonality weights and the number of ground truth labels as clusters. This ranges' effectiveness can be seen immediately from Table: 9 where we have used the same effective range to derive results for USS-NMF, that still significantly outperforms other semi-supervised methods (full hyper-parameter search). Thus, we can conclude that the effective hyper-parameter space is not really huge for USS-NMF and is comparable to other existing methods (see the number of experiments in the last row of Tables: 2 & 1 for reference).

Matrix Factorization based methods								Random Walk/ Other methods			
Co-efficients	NMF:S	NMF:S+Y	MMDW	MNMF	MNMF+Y	MF-Plan	NMF:S+Y +LS(S,Y)	Co-efficients	DeepWalk	ComE	Gemsec
Network	0.1-10.0	0.1-10.0	0.1-10.0	0.1-10.0	0.1-10.0	0.1-10.0	0.1-10.0	p	[1.0]	NA	[0.1, 0.3, 0.5, 0.7, 1.0, 3.0, 5.0, 7.0, 10.0]
Label	NA	0.1-10.0	Max-margin loss based biased gradient: e^[-1, -2, -3, -4, -5]	NA	0.1-10.0	0.1-10.0	0.1-10.0	q	NA	Network: 0.1-10.0	Same as p
Cluster Factorization	NA	NA	NA	0.1-10.0	0.1-10.0	NA	NA	Walk-Length	80	NA	80
Cluster Learning	NA	NA	NA	0.1-10.0	0.1-10.0	NA	NA	No of Walks	40, 80	NA	40, 80
Cluster Orthogonality	NA	NA	NA	$[1e + (0, 4, 8)]$	$[1e + (0, 4, 8)]$	NA	NA	Learning Rate	NA	[0.001, 0.025, 0.625, 0.1]	Initial LR: [0.01, 0.1] Minimal LR: [0.0001, 0.001]
Graph Laplacian Reg	NA	NA	NA	NA	NA	0.1-10.0	0.1-10.0	Community Learning	NA	0.1-10.0	[0.01, 0.1, 1.0]
L2 Regularization	0.1-10.0	0.1-10.0	0.1-10.0	0.1-10.0	0.1-10.0	0.1-10.0	0.1-10.0	L2 Regularization	NA	NA	0.1-10.0
#Clusters	NA	NA	NA	#Labels(-1, +2)	#Labels(-1, +2)	NA	NA	#Clusters	NA	#Labels(-1, +2)	#Labels(-1, +2)
#Experiments	25	125	125	110	130	130	130	#Experiments	2	125	204

Table 1: Hyper-parameter range search for Baselines

Co-efficients	**USS-NMF** (Effective range)		**USS-NMF** (Entire range)
Dataset (small=<1k, large>1k |V|)	Small	Large	All Datasets
Network	1, 5	1, 5, 10	0.1-10 / [0.1, 0.5, 1.0, 5.0, 10.0]
Label	0.1, 1	0.1, 0.5, 1	0.1-10
Cluster Factorization	0.1, 1	0.1, 0.5, 1	0.1-10
Cluster Learning	10	10	0.1-10
Cluster Orthogonality	1e + (0, 4)	1e + 8	1e + (0, 4, 8)
Graph Laplacian Regularization	0.5, 1	0.5, 1	0.1-10
L2 Regularization	1	1	0.1-10
#Clusters	#Labels	#Labels	#Labels(-1, +2)
#Experiments	32 (Full search)	54 (Full search)	150 (Partial seach)

We selected 25 values for k as #Labels(-1, +2), i.e., increasing 2 in the upper range and decreasing 1 in the lower range from a dataset's actual no of labels q (inclusive). We also report generic range which effectively works for most of the cases. See results in Table: 9.

Table 2: Hyper-parameter search space for USS-NMF

3 Ablation Study

Here, we drill down the components of USS-NMF to analyze the importance of utilizing information about labels and clusters.

3.1 Importance of Label Information Since the smoothening on the cluster space is based on the label similarity graph, it is necessary to verify - 1) whether we need to factorize the label matrix and 2) whether we should smooth the predicted label space locally beside this. We report results for this study in Table: 3, where we subtract the before-mentioned two components from the original model USS-NMF. The model in Column: 3 does not have the label smoothing term and the model in Column: 4 does not have the label matrix factorization as well as the label smoothing term. Note that we can not smooth on the label space based on the local neighborhood structure without predicting the labels.

Label smoothing term provides an improvement of up to 3.74 points in Wiki, 2.13 points in Texas, up to 1.5 points for Cora & Wisconsin, and no improvement in Washington & Cornell. On seven datasets, it offers an improvement of $1\% - 4\%$, and on three of the other datasets, it provides an increase between $0.3\% - 1.0\%$. Thus, the label smoothing term is quite helpful. Moreover, it can be seen from Column: 4, that removing the label matrix factorization, Y has a significant impact on USS-NMF. It results in an average drop in the performance of 2.24% with label matrix factorization solely contributing an average drop of 1.39%. The performance drop is a maximum of 7.08% on Citeseer, followed by 6.6% in Cora. On eleven datasets, it offers an improvement of $2\% - 9\%$, and on rest two of the datasets, it provides an increment in performance between $1.0\% - 1.5\%$. Whereas, the average reduction in performance when we remove the label smoothing term is approximately $0.85\%(stdev : 0.8\%)$. This indicates that it is necessary to have both the terms in our objective.

	USS-NMF	- LS(S, $\hat{Y}$)	- (Y + LS(S, $\hat{Y}$))
Cora	**87.306**	85.883	80.738
Citeseer	**70.187**	69.825	63.109
Wiki	**70.906**	67.168	65.339
Washington	**67.826**	**67.826**	65.217
Wisconsin	**56.391**	54.887	51.281
Texas	**63.830**	61.703	59.574
Cornell	**56.122**	**56.122**	52.020
PPI	**23.433**	23.130	22.346
Wikipedia	**57.894**	56.254	55.578
Pubmed	**84.009**	83.981	78.945
Co-Author	**37.858**	36.972	36.198
Blogcatalog	**41.254**	39.736	38.135
Microsoft	**49.706**	48.669	47.153

USS-NMF: Proposed method, LS(S, $\hat{Y}$): Local smoothing on the predicted label space, Y: Inclusion of label information through factorization. In 3^{rd} & 4^{th} columns the minus sign indicates the respective terms are removed from the objective of USS-NMF.

Table 3: Importance of Label Information.

3.2 Importance of Cluster Information Here, we try to understand the importance of the proposed cluster smoothing term by removing the cluster smoothing term first and then additionally removing the cluster factorization term too. This corresponds to Column: 3 and Column: 4 in Table: 5 respectively. Removing the cluster smoothing term results in an average drop of 1.82% across datasets whereas completely removing

	USS-NMF	- LS(E, Ĥ)	- (Ĥ + LS(E, Ĥ))
Cora	**87.306**	86.159	86.109
Citeseer	**70.187**	69.101	68.697
Wiki	**70.906**	70.407	69.825
Washington	**67.826**	63.478	62.609
Wisconsin	**56.391**	52.632	52.632
Texas	**63.830**	62.766	61.702
Cornell	**56.122**	52.041	52.041
PPI	**23.433**	23.281	23.191
Wikipedia	**57.894**	56.814	56.637
Pubmed	**84.009**	83.540	83.479
Co-Author	**37.858**	36.684	36.200
Blogcatalog	**41.254**	38.505	38.111
Microsoft	**49.706**	47.676	47.008

LS(E, Ĥ): Label similarity based smoothing on the predicted cluster space, Ĥ: Inclusion of predicted cluster information through factorization. In 3rd & 4th columns the minus sign indicates the respective terms are removed from the objective of USS-NMF.

Table 5: Importance of Cluster Information.

all cluster related components results in an average drop of 2.2% on performance. Cluster smoothing term provides an additional improvement between $2\% - 4\%$ on five datasets and $1\% - 2\%$ improvement on other five datasets. By removing the cluster assignment/ factorization term, we observe a drop in performance by a maximum of 5.22% on Washington, 4.08% on Cornell & 3.76% on Wisconsin, followed by 3.14% on Blogcatalog, between $1.0\% - 2.5\%$ for seven other datasets and $< 0.5\%$ for the rest. Note that random cluster assignments with unsupervised clustering objective (ComE, MNMF, GEMSEC) did reasonably well but only a few times outperformed the second-best model, MNMF+Y in Table: 3. USS-NMF outperforming all - demonstrates the usefulness of encoding label similarity invariant representations on cluster space. In the next section, we will further clearly see the benefits of combining label and cluster smoothing terms.

4 Study on Laplacian smoothing variants

In this section, we analyze and compare different Laplacian smoothing variants in Table: 7. The models in the first four columns enforces Laplacian smoothing with the proximity graph, S and the next four models enforces Laplacian smoothing with the label similarity graph, E. All the models in this table additionally factorize label matrix, Y. The model in the last column is USS-NMF. *USS-NMF is the winner across the board on all datasets with Rank* 1 *and Penalty* 0.

Model in Column: 3, $NMF : LS(S, \hat{Y}) + Y$, is the standard Label Propagation implemented in NMF style. It does not additionally factorize the proximity matrix, S and thus does not have any network embeddings. This is the second-worst performing model with the highest rank and highest penalty. From this model, it is evident that we need to learn from the network as well.

Model in Column: 2, $NMF : LS(S, U) + Y$ learns embeddings such that connected nodes have closer embeddings. It is similar in spirit to $NMF : S + Y$ which factorizes the proximity matrix directly. This model has the lowest penalty.

Model in Column: 6, $NMF : LS(E, U) + S + Y$ (NMF: Planetoid) has the same objective as Planetoid-G but in NMF style. It enforces nodes with similar labels to have similar embeddings. In a head on comparison with $NMF : LS(S, U) + Y$ (Column: 2), it shows that *enforcing smoothness from a global context has remarkable improvement* in Penalty scores and is statistically better with $p < 0.02$.

Model in Column: 5, $NMF : LS(S, \hat{Y}) + S + Y$, is a powerful model which is second on 5/13 datasets. It can be seen as an improvement of Label Propagation where it additionally factorizes proximity S to learn node embeddings that capture network structure. It clearly beats the base $NMF : LS(S, \hat{Y}) + Y$ in Column: 3.

Model in Column: 7, is an extension of NMF: Planetoid (Column: 6) which includes label smoothing. It can be seen that adding label smoothing provides an improvement on all datasets up to 2.24 points.

Model in Column: 8 is USS-NMF without the label smoothing term. Label smoothing term is crucial as it gives an improvement upto 3.74 points in Column: 9 (USS-NMF). *And, similarly semi-supervised clustering with global context is also important as it provides an improvement between* $0.5 - 4.08$ *points over* $NMF : LS(S, \hat{Y}) + S + Y$ *(Column: 5) in most of the datasets except Cora, PPI. Adding clustering components to label smoothing term* $NMF : LS(S, \hat{Y}) + S + Y$ *(Col: 5) provides an avg improvement of* 1.6 *points in Col: 9.*

5 SSL with balanced dataset:

Planetoid [7] was defined for multi-class classification problem on balanced labeled set, i.e., the same number of representative train instances for all labels (one unrealistic setting for real-world data). We empirically observed Planetoid performing poorly in comparison to other baselines when the labeled set is randomly drawn. Also, Planetoid is not directly extensible for multi-label problems. Hence, we define a similar semi-supervised NMF model, (NMF:Planetoid) or (MF-Plan) with Planetoid's semi-supervised learning objective i.e., explicitly enforcing embeddings of nodes of the same label to be similar.

Here, we report results for models on the original balanced train/val/test split given with Planetoid [7].

Laplacian	Using Neighborhood Similarity Graph (S) : LS(S, *)				Using Label Similarity Graph (E) : LS(E, *)			
Invariant Space	Embedding: U^*	Label: $\hat{Y}^*$	Cluster: $H(U)^*$	Label: $\hat{Y}^*$	Embedding: U^*		Cluster: $H(U)^*$	
	+Y	+Y	+Y	+Y +S	+Y +S	+Y +S + LS(S, Ŷ)	+Y +S	+Y +S + LS(S, Ŷ)
Cora	86.568	85.683	85.506	87.109	85.756	86.716	85.883	**87.38**
Citeseer	69.040	68.825	68.678	68.697	68.637	69.039	69.825	**70.187**
Wiki	69.410	69.742	66.885	69.825	67.249	69.493	67.168	**70.906**
Washington	60.000	55.652	66.957	66.957	66.957	65.652	**67.826**	**67.826**
Wisconsin	48.120	49.624	52.880	52.632	52.759	52.880	54.887	**56.391**
Texas	57.447	59.574	60.634	61.702	60.638	61.702	61.703	**63.830**
Cornell	52.041	53.061	52.041	52.041	53.061	54.082	**56.122**	**56.122**
PPI	22.346	22.648	23.087	23.191	22.497	22.886	23.130	**23.433**
Wikipedia	44.277	43.304	56.147	56.637	55.805	56.398	56.254	**57.894**
Pubmed	83.387	83.499	83.151	83.479	82.779	83.801	83.981	**84.009**
Co-Author	36.198	36.127	36.260	36.200	36.684	36.908	36.973	**37.858**
Blogcatalog	36.246	36.103	39.001	39.111	38.983	38.083	39.736	**41.254**
Microsoft	43.220	45.397	48.100	48.108	47.237	48.213	48.669	**49.706**
Rank	6.46	6.31	5.46	3.85	5.62	3.77	2.69	1
Penalty	4.5	4.43	2.11	1.62	2.13	1.61	1.13	**0.000**

Table 7: Semi-Supervised Learning Analysis | Micro-F1 Scores

Datasets	Planetoid-G	MMDW	NMF:S+Y	MNMF+Y	MF-Plan	NMF:S+Y+LS(S,Ŷ)	USS-NMF
Cora	69.1	71.6	74.3	75.9	75.1	75.4	**79.1**
Citeseer	49.3	60.9	62.6	61.6	63.1	63.7	**66.7**
Pubmed	66.4	80.6	80.4	79.6	80.5	80.7	**82.1**

Planetoid-G versus all the competing Semi-Supervised models (SOTA Baselines & their variants) for Planetoid paper's train/test splits of graph data

Table 8: Node Classification Results | Balanced Sampling | Micro-F1 Scores

The non-attributed model of Planetoid, Planetoid-G, performs poorly in comparison to the NMF models even in this setup. Our results for Planetoid on running the author's code is very similar to what is reported in their paper. However, we can see significant improvement in the results for NMF:Planetoid. The improvement owing to MF-Plan can be attributed to (i) exact computations instead of sampling strategy followed in the original paper, and (ii) non-negative embeddings. From Table: 8 it is evident that among all the competing methods USS-NMF does significantly better. We find balanced class distribution to be beneficial for our model as we obtain extraordinary performance improvement on Cora, Citeseer and Pubmed.

6 SSL with varying ratio of labeled data:

We also report node classification results for varying test-train splits to study and understand how our method does on various label sparsity cases in Table: 9 against all the semi-supervised State-Of-The-Art (SOTA) methods as baselines. The table is enough reflective of the fact that USS-NMF does well in case of label sparsity too. For varying splits of randomly sampled labeled data, it consistently outperformed the other baselines. When the labeled data is sparse, the cluster information acts as complementary information to help to predict labels for the unlabeled nodes.

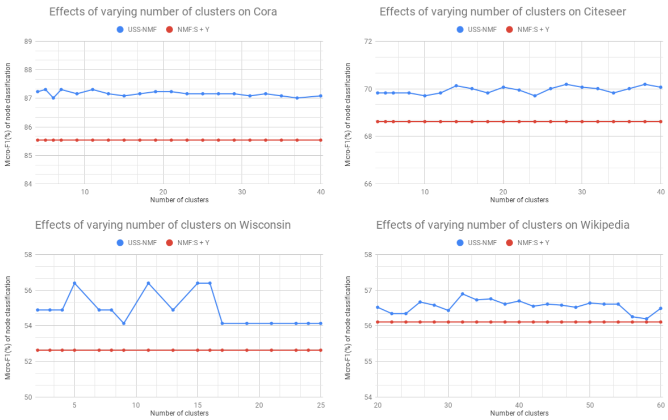

Figure 1: Varying Number of Clusters

7 USS-NMF's sensitivity to number of clusters

We performed an extensive study on how the number of clusters influences node classification performance and whether there is any need for learning clusters at all. For Cora, Citeseer and Wisconsin - three multi-class datasets and Wikipedia - a multi-label dataset, we varied the number of clusters as in Figure: 1. Blue solid lines indicate the classification performance of our proposed methods USS-NMF. Corresponding Red solid horizontal lines represent respective performances where all the

Dataset	Cora						Citeseer						Wiki				
Train (%)	5	10	20	30	40	50	5	10	20	30	40	50	10	20	30	40	50
NMF:S+Y	67.519	76.620	79.012	84.660	84.194	85.535	51.128	54.172	62.068	64.379	68.477	68.618	57.367	60.442	62.945	63.404	65.152
MMDW	67.403	75.101	80.093	82.942	82.889	83.838	47.283	54.509	58.544	63.362	66.265	68.911	56.655	58.120	62.270	63.391	67.007
MNMF+Y	68.947	76.948	81.172	84.396	83.518	85.904	50.906	55.533	63.787	65.689	67.622	69.005	56.536	62.130	64.549	64.751	66.999
USS-NMF	**69.452**	**78.015**	**82.880**	**85.609**	**85.486**	**87.380**	**52.253**	**57.478**	**65.070**	**66.422**	**69.231**	**70.187**	**58.568**	**62.909**	**67.102**	**69.183**	**70.906**

Table 9: Node Classification Results | Varying Train-Test Splits | Micro-F1 Scores

cluster related terms were set to 0 ($\beta = 0, \phi = 0, \zeta = 0$), i.e., learning no clusters (NMF:S+Y best scores).

As we can see, major portions of the curves are above their respective dotted lines, indicating that learning clusters help in guiding node representations. From the plot, it is interesting to see that small dataset like Wisconsin is more sensitive to changing the number of clusters than larger datasets. For Wikipedia, a multi-label dataset, we can clearly see the tendency of learning overlapping clusters as the optimal clusters are lesser than the ground-truth labels. Again, there are optimal clusters very different from the ground truth labels (Citeseer) and multiple numbers of optimal clusters (in Cora, Citeseer, Wisconsin) which indicates that small clusters of same class data having low density separation among them (Refer to t-SNE plots) are being learned.

8 Convergence Analysis for USS-NMF

We study and validate the convergence property of USS-NMF in Figure: 2, where the objective function values are plotted against iteration numbers. The objective function values are non increasing with iterations and a sharp decrease in the objective function values can be seen within a few iterations between $0 - 10$, which empirically proves the correctness of our algorithm.

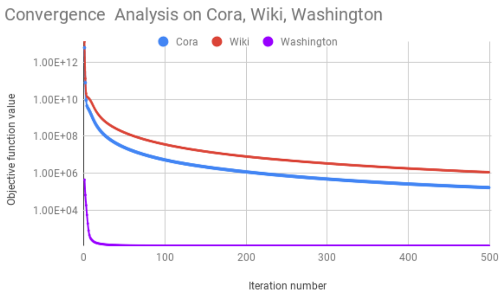

Figure 2: Convergence Analysis of USS-NMF

References

[1] S. Cavallari et al. "Learning community embedding with community detection and node embedding on graphs". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017.

[2] P. Goyal and E. Ferrara. "Graph embedding techniques, applications, and performance: A survey". In: *Knowledge-Based Systems* 151 (2018).

[3] A. Grover and J. Leskovec. "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016.

[4] B. Perozzi, R. Al-Rfou, and S. Skiena. "Deepwalk: Online learning of social representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014.

[5] B. Rozemberczki et al. "Gemsec: Graph embedding with self clustering". In: *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 2019.

[6] C. Tu et al. "Max-margin deepwalk: Discriminative learning of network representation." In: *IJCAI*. 2016.

[7] Z. Yang, W. Cohen, and R. Salakhudinov. "Revisiting Semi-Supervised Learning with Graph Embeddings". In: *International Conference on Machine Learning*. 2016.