Explainable Automated Graph Representation Learning with Hyperparameter Importance

Xin Wang [1] Shuyi Fan [1] Kun Kuang [2] Wenwu Zhu [1]

e-AutoGR (ICML'21)

Explainable Automated Graph Representation Learning with Hyperparameter Importance

文言

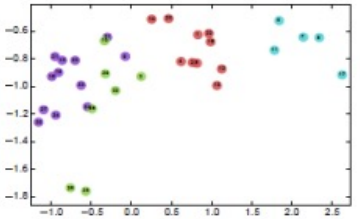
wen-y18@mails.tsinghua.edu.cn

2022. 3. 18

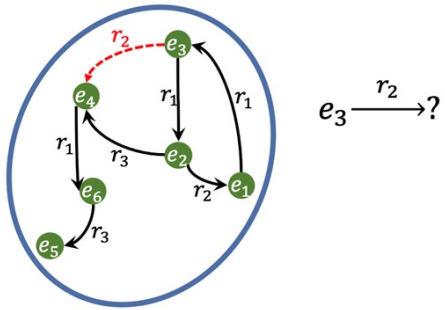
Outline

- **Background**
  - Graph Representation Problem
  - HPs Optimization on Graph Representation with subsampling
  - AutoNE Framework
  - Previous Methods on Graph and motivation
- Explainable Automated Graph Representation
- Experiments
- Discussion

Graph Representation Problem



- **Graph Definition:** $G = (V, E)$
- **Graph Representation Algorithm** $\mathbf{R}$
- **HP Optimization on Graph: Find** $\boldsymbol{\Lambda^*}$ **to optimize** $P_{\mathbf{R}}(\boldsymbol{\Lambda}, G)$ **on Graph**
  - Space of $\boldsymbol{\Lambda}$: Based on different Algorithm **R**
- **Typical Algorithm** R and its HPs
  - **DeepWalk (**number of random walks, walk length, windows size**)**
  - **AROPE (**weights of different order proximity**)**
  - **GCN (**training epochs, number of neurons, learning rate, dropout rate, norm-2 regularizations, etc. Just as deep learning**)**
- **Performance** P: **N**ode **C**lassification: **Micro-F1; L**ink **P**rediction: **AUC**



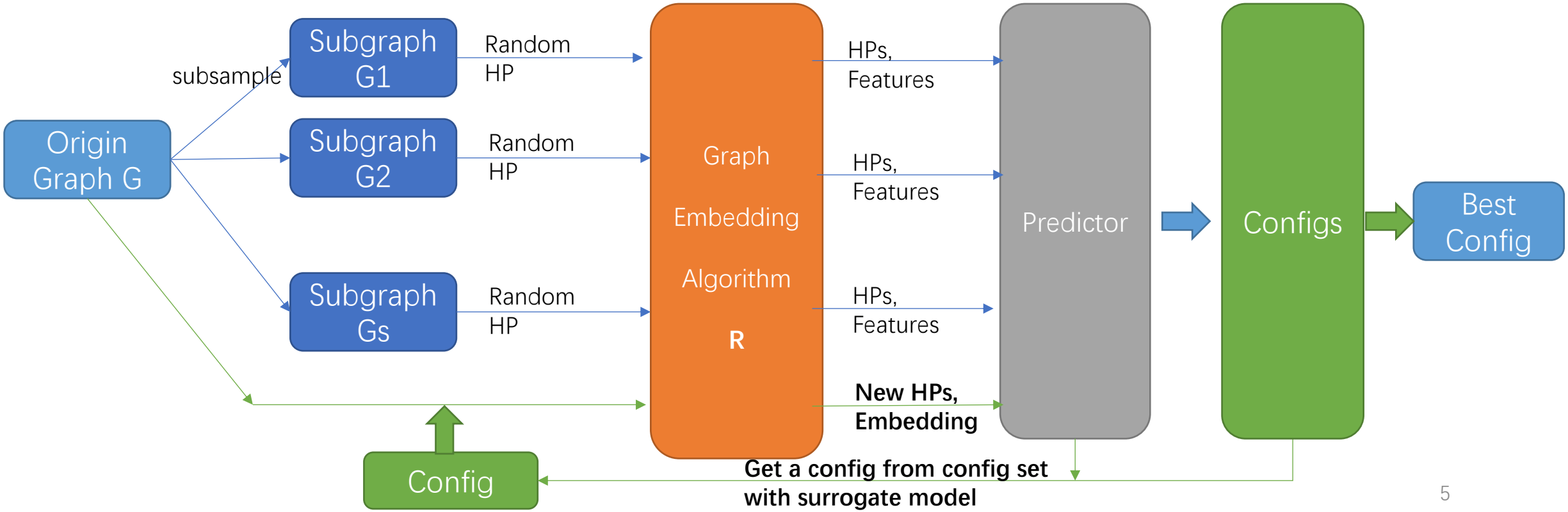
GR Problem with Subsampling

- **Why subsampling?**
  - **Large Size Dataset: |V|~10k, |E|~100k**
  - **Limited Computation Cost**
  - **Origin Graph:** $G = (V, E)$ ➔ **Subgraphs:** $G_i = (V_i, E_i)\ i = 1,2, \dots s.$
- **Framework of Hyperparameter Optimization Problem on Graph Representation with Subsampling**
  - Two-Stage optimization: Subgraphs ➔ Origin Graph
  - HP Optimization & Performance on subgraphs & HP Optimization on the origin graph

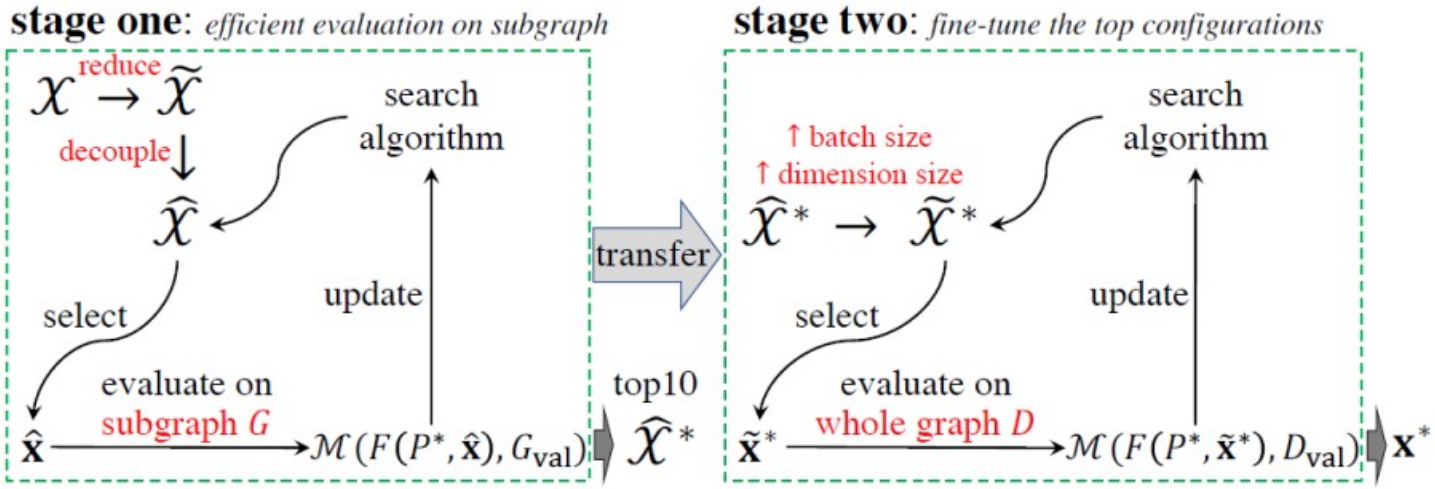
GR Problem with Subsampling

- **Origin Graph:** $G = (V, E)$ ➔ **Subgraphs:** $G_i = (V_i, E_i)\ \ i = 1,2, \ldots s.$

- **A Normal Framework of Two-Stage fine-tune on Graph Representation**



Related work: KGTuner

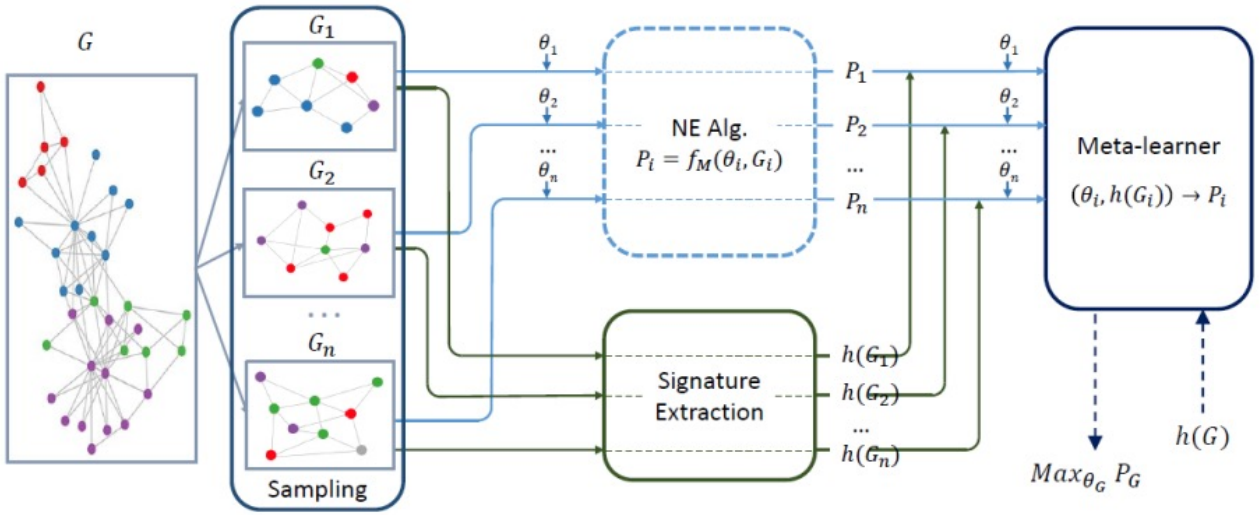


- TOSS: A Two-stage Algorithm on KGE
  - Random Subsampling, Choosed HPs and P from subgraphs; update HPs and P on original graph
  - Reduce and Decouple Search Space
  - Use RF as surrogate model
  - Much more analysis on importance of HPs

Algorithm 1 TOSS: two-stage search algorithm

Require: KG embedding model F, dataset D, and budget B;
reduce the search space $\mathcal{X}$ to $\tilde{\mathcal{X}}$ and decouple $\tilde{\mathcal{X}}$ to $\hat{\mathcal{X}}$;
 # state one: *efficient evaluation on subgraph*
sample a subgraph (with 20% entities) G from D_{tra} by multi-start random walk;
repeat
  sample a configuration $\hat{\mathbf{x}}$ from $\hat{\mathcal{X}}$ by RF+BORE;
  evaluate $\hat{\mathbf{x}}$ on the subgraph G to get the performance;
  update the RF with record $(\hat{\mathbf{x}}, \mathcal{M}(F(P^*, \hat{\mathbf{x}}), G_{\text{val}}))$;
until $B/2$ budget exhausted;
save the top10 configurations in $\hat{\mathcal{X}}^*$;
 # state two: *fine-tune the top configurations*
increase the batch/dimension size in $\hat{\mathcal{X}}^*$ to get $\tilde{\mathcal{X}}^*$;
set $y^* = 0$ and initialize the RF surrogate;
repeat
  select a configuration $\tilde{\mathbf{x}}^*$ from $\tilde{\mathcal{X}}^*$ by RF+BORE;
  evaluate on whole graph G to get the performance;
  update the RF with record $(\tilde{\mathbf{x}}^*, \mathcal{M}(F(P^*, \tilde{\mathbf{x}}^*), D_{\text{val}}))$;
  if $\mathcal{M}(F(P^*, \tilde{\mathbf{x}}^*), D_{\text{val}}) > y^*$ **then**
    $y^* \leftarrow \mathcal{M}(F(P^*, \tilde{\mathbf{x}}^*), D_{\text{val}})$ and $\mathbf{x}^* \leftarrow \tilde{\mathbf{x}}^*$; **end if**
until $B/2$ budget exhausted;
return $\mathbf{x}^*$.

Related work: AutoNE



- AutoNE: Also a Two-stage Algorithm GR
  - Subsampling, Learn HPs and P from subgraphs; update HPs and P on original graph
  - Use Gaussian Process as surrogate model
- Not good:
  - Transferability in Subsampling, Importance of HPs

AutoNE: Hyperparameter Optimization for Massive Network Embedding

Algorithm 1 *Automated Hyperparameter Optimization for Network Embedding* (AutoNE)

Input: Network G; Network embedding algorithm M.
Output: The optimal hyperparameter configuration θ_{opt}.

/* Phase I */
Sample S sub-networks from G according to Section 3.2. Each sampled sub-network will be reused for T times. We therefore use $\{G_i\}_{i=1}^{ST}$ to denote the ST sub-networks.
$\mathrm{X} = \{\}, \mathrm{f} = \{\}$.
for sub-network $i \leftarrow 1, 2, \ldots, ST$ **do**
  Compute the signature of G_i according to Section 3.3. The signature will be used by the kernel function $k(\cdot, \cdot)$.
  Select a hyperparameter configuration θ_i randomly.
  Run algorithm M on sub-network G_i using θ_i, and record the performance $f_M(\theta_i, G_i)$.
  $\mathrm{X} \leftarrow \mathrm{X} \cup \{(\theta_i, G_i)\}, \mathrm{f} \leftarrow \mathrm{f} \cup \{f_M(\theta_i, G_i)\}$.
end for
Update the kernel parameters by maximizing Equation 5.

/* Phase II */
Compute the signature of network G according to Section 3.3.
for trial $j \leftarrow ST + 1, ST + 2, \ldots, ST + L$ **do**
  Obtain a hyperparameter configuration θ_j that may achieve the optimal performance on G according to Equation 10.
  Run algorithm M on the input network G using θ_j, and record the performance $f_M(\theta_j, G)$.
  $\mathrm{X} \leftarrow \mathrm{X} \cup \{(\theta_j, G)\}, \mathrm{f} \leftarrow \mathrm{f} \cup \{f_M(\theta_j, G)\}$.
  Update the kernel parameters by maximizing Equation 5.
end for
return $\theta_{opt} = \arg\max_{\theta_j : ST+1 \le j \le ST+L} f_M(\theta_j, G)$.

Motivation

- To explore importance of hyper-parameters
  - Explore the correlation between different HPs
  - Explore the similarity between original graph and subgraphs
  - Improve evaluate efficiency when optimizing HPs on original graph

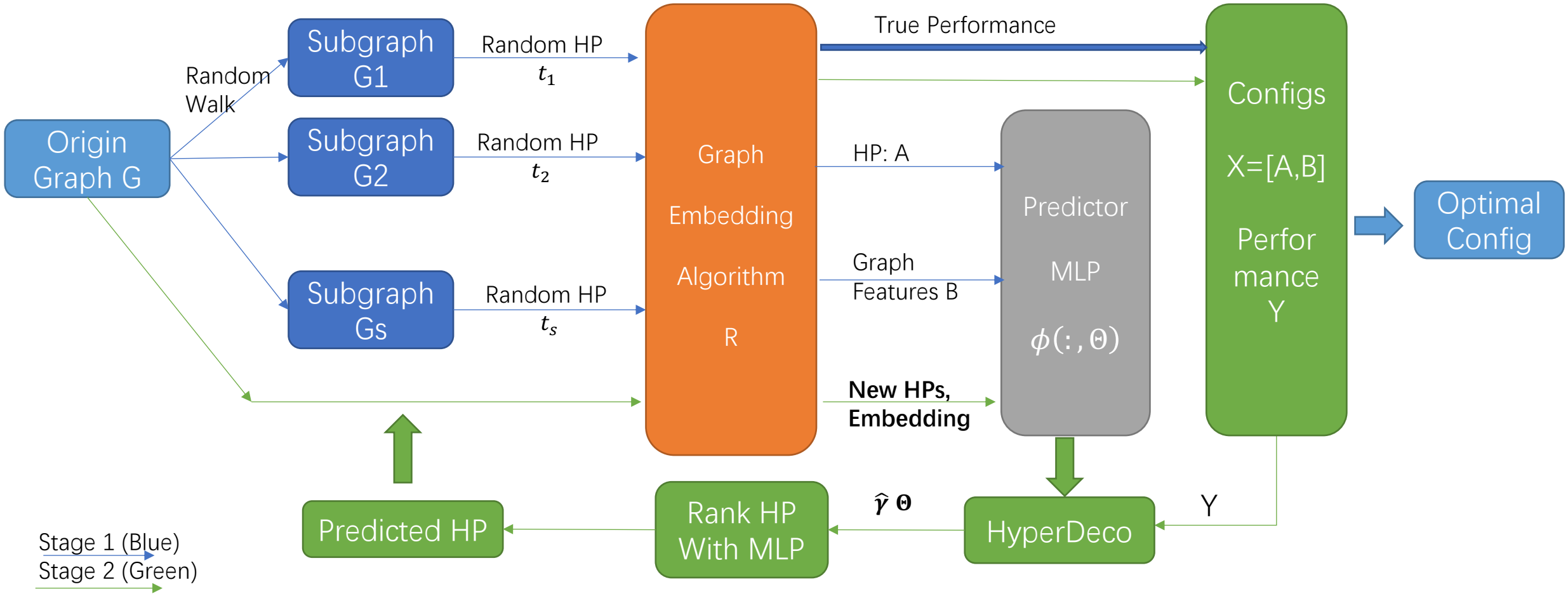
Outline

- Background
- **Explainable Automated Graph Representation**
  - General Thought and Framework
  - Graph Subsampling
  - Weighting Regression with Hyperparameter Decorrelation (HyperDeco)
  - E-AutoGR Algorithm
- Experiments
- Discussion

General Thought of e-AutoGR

- Subsampling from Origin Graph
  - Random Walks from different communities
  - How to calculate the **similarity** between {Gi} and G?
- Optimize HPs with Algorithm R on Subgraphs
  - How to allocate time budget?
  - Why and How do we decorrelate HPs? (HyperDeco)
- Optimize HPs with R on Origin Graph
  - How does e-AutoGR work?
  - What's its performance on G?
  - How does **e**-AutoGR show **exploration** of HPs?

Framework of e-AutoGR



Graph Subsampling

- G = (V, E) ➔ Gi = (Vi, Ei) i = 1,2,…,s
  - Expectation: preserve property of origin graph, reduce fine-tune cost
  - Start from different regions(communities) of G, vary length of walk
  - Cover the property and diversity of G
    - How to calculate similarity?
    - Use features from these graphs! Also choose a distance metric in embedding space!
- Graph Feature and Computation Allocation
  - Based on choosed features from subgraphs and graph(Not the embedding!)

Graph Features and Allocation

- Features of Subgraph (f^i) vs. Original (f)
  - Six fully explainable features from Graph
  - Canberra Distance $g^i = d(\mathbf{f}^i, \mathbf{f}) = \sum_{k=1}^{6} \frac{|\mathbf{f}_k^i - \mathbf{f}_k|}{|\mathbf{f}_k^i| + |\mathbf{f}_k|}$.
- Computation Allocation (Total n times)
  - t_i times on subgraph G_i
  - Based on distance $t_i = \left\lfloor n * \frac{g^i}{\sum_{k=1}^{s} g^k} \right\rfloor$.
  - Less similar, Larger g, More running times
- Reference
  - i)Scalability, ii) Sensitivity to graph size
  - iii) Sensitivity to similar topologies, iv) Label free

- Number of nodes: $|V|$
- Number of edges: $|E|$
- Number of triangles: The number of nodes forming a triangle which is a set of three nodes connected with each other by three edges.
- Global clustering coefficient: $\frac{3*Number\ of\ triangles}{Number\ of\ triplets}$, where a triplet is three nodes that are connected by either two or three (forming a triangle) undirected edges.
- Maximum total degree value: The total number of edges the most connected node in the graph has to other nodes.
- Number of Components: The total number of components contained in the graph, where a component is a subgraph in which there is a possible path between every node, while nodes from different components have no path connecting them.

Weighting Regression with Hyperparameter Decorrelation

- What do we get now?
  - $\boldsymbol{n}$ samples with configurations of HPs and performance from s subgraphs
  - HPs matrix $A \in R^{n \times p_1}$, Graph Features matrix $B \in R^{n \times p_2}$
  - concat➔$X = [A, B] \in R^{n \times p}$, p1=3or5, p2=6
  - Performance Vector of n samples(F1-score, AUC, etc.): $Y \in R^{n \times 1}$
- Motivation and Methods
  - Get the optimal HPs configuration for G
  - Surrogate: A non-linear map from HPs to Performance (use MLP with Θ)
  - Different weight on different samples for Decorrelation

Decorrelation HPs

- Attributes decorrelation example: sample_size=n, feature_dim=p:
  - Feature X(n x p) ➔ Y(n x p), Cov(Xi, Xj)≠0 ➔ Cov(Yi, Yj)=0 (when i≠j, column)
  - Method: Covariance and Diagonal factorization
- Correlation and Covariance
  - $\mathrm{Cov}(X, Y) = E(XY) - E(X)E(Y) \to 0$
  - HPs decorrelation: $$\min_{\gamma} \sum_{j=1}^{p_1} \left\| \mathbb{E}[\mathbf{A}_{,j}^T \mathbf{\Sigma}_\gamma \mathbf{X}_{,-j}] - \mathbb{E}[\mathbf{A}_{,j}^T \gamma] \mathbb{E}[\mathbf{X}_{,-j}^T \gamma] \right\|_2^2, \quad (3)$$
  - Sample weights: $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n), \sum_i \gamma_i = n$
  - Loss function: $$\mathcal{L}_{Deco} = \sum_{j=1}^{p1} \left\| \mathbf{A}_{,j}^T \mathbf{\Sigma}_\gamma \mathbf{X}_{,-j}/n - \mathbf{A}_{,j}^T \gamma/n \cdot \mathbf{X}_{,-j}^T \gamma/n \right\|_2^2. \quad (4)$$

Decorrelation HPs

- Correlation and Covariance
  - Primal Loss function $$\mathcal{L}_{Deco} = \sum_{j=1}^{p_1} \left\| \mathbf{A}_{\cdot j}^T \mathbf{\Sigma}_\gamma \mathbf{X}_{\cdot,-j}/n - \mathbf{A}_{\cdot j}^T \gamma/n \cdot \mathbf{X}_{\cdot,-j}^T \gamma/n \right\|_2^2. \quad (4)$$
  - Some Thms about L_Deco $$\hat{\gamma} = \arg\min_{\gamma \in \mathcal{C}} \mathcal{L}_{Deco} + \frac{\lambda_3}{n} \sum_{i=1}^n \gamma_i^2 + \lambda_4 (\frac{1}{n} \sum_{i=1}^n \gamma_i - 1)^2, \quad (6)$$

Lemma 1 *If the number of covariates p_1 and p_2 is fixed, then there exists a sample weight $\gamma \succeq 0$ such that*

$$\lim_{n \to \infty} \mathcal{L}_{Deco} = 0 \quad (5)$$

with probability 1. In particular, a solution γ to Eq (5) is $\gamma_i^\star = \frac{\Pi_{j=1}^p \hat{f}(\mathbf{X}_{i,j})}{\hat{f}(\mathbf{X}_{i,1}, \cdots, \mathbf{X}_{i,p})}$, where $\hat{f}(x_{\cdot,j})$ and $\hat{f}(x_{\cdot,1}, \cdots, x_{\cdot,p})$ are the Kernel density estimators.[2]

Theorem 1 *The solution $\hat{\gamma}$ defined in Eq (6) is unique if $\lambda_3 n \gg p^2 + \lambda_4$, $p^2 \gg \max(\lambda_3, \lambda_4)$ and $|\mathbf{X}_{i,j}| \leq c$ for some constant c.*

Property 1. *When p_1 and p_2 is fixed, $n \to \infty$, $\lambda_3 n \gg p^2 + \lambda_4$, and $p^2 \gg \max(\lambda_3, \lambda_4)$, the variables in $\mathbf{A}$ become uncorrelated with the variables in $\mathbf{X}$ by sample reweighting with $\hat{\gamma}$.*

HyperDeco

- Learning a map from HPs and features to performance
  - Predict with MLP

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{n} \hat{\gamma}_i \cdot (Y_i - \phi(\mathbf{X}_{i,}; \Theta))^2, \tag{7}$$

- Final optimization problem Algorithm
  - Joint optimization on γ and Θ

$$\begin{aligned} & \min_{\gamma,\Theta} \textstyle\sum_{i=1}^{n} \gamma_i \cdot (Y_i - \phi(\mathbf{X}_{i,}; \Theta))^2 \qquad (8) \\ s.t \quad & \textstyle\sum_{j=1}^{p_1} \left\| \mathbf{A}_{,j}^T \mathbf{\Sigma}_\gamma \mathbf{X}_{,-j}/n - \mathbf{A}_{,j}^T \gamma/n \cdot \mathbf{X}_{,-j}^T \gamma/n \right\|_2^2 < \lambda_2 \\ & |\Theta|_1 < \lambda_1, \ \ \tfrac{1}{n} \textstyle\sum_{i=1}^{n} \gamma_i^2 < \lambda_3, \\ & (\frac{1}{n} \sum_{i=1}^{n} \gamma_i - 1)^2 < \lambda_4, \ \ \gamma \succeq 0, \end{aligned}$$

$$\hat{\gamma} = \arg\min_{\gamma \in \mathcal{C}} \mathcal{L}_{Deco} + \frac{\lambda_3}{n} \textstyle\sum_{i=1}^{n} \gamma_i^2 + \lambda_4 (\frac{1}{n} \sum_{i=1}^{n} \gamma_i - 1)^2, \tag{6}$$

Algorithm 1 Hyperparameter Decorrelation Weighting Regression (HyperDeco)

1: **Input:** Observed $\mathbf{X} = [\mathbf{A}, \mathbf{B}]$ and performance Y, where $\mathbf{A}$ denotes hyperparameters and $\mathbf{B}$ denotes graph features.
2: **Output:** Updated parameters γ, Θ.
3: Initialize parameters $\gamma^{(0)}$ and $\Theta^{(0)}$,
4: Calculate loss function with parameters $(\gamma^{(0)}, \Theta^{(0)})$,
5: Initialize the iteration variable $t \leftarrow 0$,
6: **repeat**
7: $t \leftarrow t + 1$,
8: Update $\gamma^{(t)}$ with gradient descent by fixing Θ,
9: Update $\Theta^{(t)}$ with gradient descent by fixing γ,
10: Calculate loss function with parameters $(\gamma^{(t)}, \Theta^{(t)})$,
11: **until** Loss function converges or max iteration is reached

E-AutoGR in two stages: Stage 1

- Stage 1
  - Subsampling Graph from G
  - Run t_i times on G_i with R
  - Total n samples for next stage
    - Get HPs matrix A
    - Get subgraphs feature B
    - Get subgraphs performance Y

Algorithm 2 Explainable Automated Graph Representation (e-AutoGR)

1: **Input:** Graph G, Graph representation algorithm R.
2: **Output:** The optimal hyperparameter configuration Λ^*.
3: Sample s subgraphs $G_i, i = 1, 2, \ldots, s$ from original graph G according to Section 3.2.1.
4: Decide t_i for each G_i according to Section 3.2.2.
5: Execute algorithm R on each subgraph G_i for t_i times and obtain hyperparameter matrix A and graph feature matrix B as well as the performance vector Y.
6: Initialize $Count = T$.
7: **repeat**
8: Execute **Step 1** to **Step 3** in Section 3.4.
9: $Count = Count - 1$.
10: **until** $Count == 0$

E-AutoGR in two stages: Stage 2

- Step 1: Get γ and Θ with HyperDeco
  - Complexity: $O(cnp^2)$
- Step 2: Rank HPs by importance
  - Sample 1000 possible values for HPs
  - Rank HPs by MLP 1st layer weights → importan
  - Optimize most important HP with other fixed
  - Traverse every HP fix the others one by one

Algorithm 2 Explainable Automated Graph Representation (e-AutoGR)

Input: Graph G, Graph representation algorithm R.
Output: The optimal hyperparameter configuration Λ^*.
Sample s subgraphs $G_i, i = 1, 2, \ldots, s$ from original graph G according to Section 3.2.1.
Decide t_i for each G_i according to Section 3.2.2.
Execute algorithm R on each subgraph G_i for t_i times and obtain hyperparameter matrix A and graph feature matrix B as well as the performance vector Y.
Initialize $Count = T$.
repeat
    Execute **Step 1** to **Step 3** in Section 3.4.
    $Count = Count - 1$.
until $Count == 0$

E-AutoGR in two stages: Stage 2

- Step 1: Get γ and Θ with HyperDeco
- Step 2: Rank HPs by importance, get X_j
- Step 3
  - Run algorithm R on G with X_j, get Y_j
  - X = [X, X_j], Y = [Y, Y_j]
- Return optim HP config Λ^*
- Complexity: $O(|T||E|)$

Algorithm 2 Explainable Automated Graph Representation (e-AutoGR)

1: **Input:** Graph G, Graph representation algorithm R.
2: **Output:** The optimal hyperparameter configuration Λ^*.
3: Sample s subgraphs $G_i, i = 1, 2, \ldots, s$ from original graph G according to Section 3.2.1.
4: Decide t_i for each G_i according to Section 3.2.2.
5: Execute algorithm R on each subgraph G_i for t_i times and obtain hyperparameter matrix A and graph feature matrix B as well as the performance vector Y.
6: Initialize $Count = T$.
7: **repeat**
8: Execute **Step 1** to **Step 3** in Section 3.4.
9: $Count = Count - 1$.
10: **until** $Count == 0$

Outline

- Background
- E-AutoGR Framework
- **Experiments**
  - Settings
  - Best performance and results of different tasks
  - Out-of-Sample Link Prediction
- Discussion

Datasets

- BlogCatalog (social network)
  - 10312 nodes, 333983 edges and 39 categories
- Wikipedia (word co-occurrence network)
  - 4777 nodes, 184812 edges and 40 labels
- Pubmed (citation network)
  - 19717 nodes, 44338 edges
  - 500-dimension node features and 3 classes

Graph Representation Algorithms

- DeepWalk
  - Sampling-based graph representation algorithm
  - **HPs (**Number of random walks**)**
- AROPE
  - Representative factorization-based graph representation algorithm
  - **HPs (**Weights of different order proximity**)**
- GCN
  - Deep Network based algorithm
  - **HPs (**training epochs, number of neurons, learning rate, dropout rate, norm-2 regularizations, etc. Just as deep learning**)**

Comparable Approach

- e-AutoGR
- AutoNE
  - Gaussian Process with a predefined kernel to estimate the hyperparameter performance
  - But without explanation on HPs and subsampling
- Bayesian optimization (BayesOpt)
  - Sequential model-based optimization method
  - Gaussian Process to model the surrogate function (HPs➔performance)
- Random search
  - Most widely used method

All Tasks in experiments

GR/Datasets	BlogCatalog	Wikipedia	Pubmed
Deepwalk	Link Prediction Classification	Link Prediction Classification	Link Prediction Classification
AROPE	Link Prediction Classification	Link Prediction Classification	Link Prediction Classification
GCN			Classification

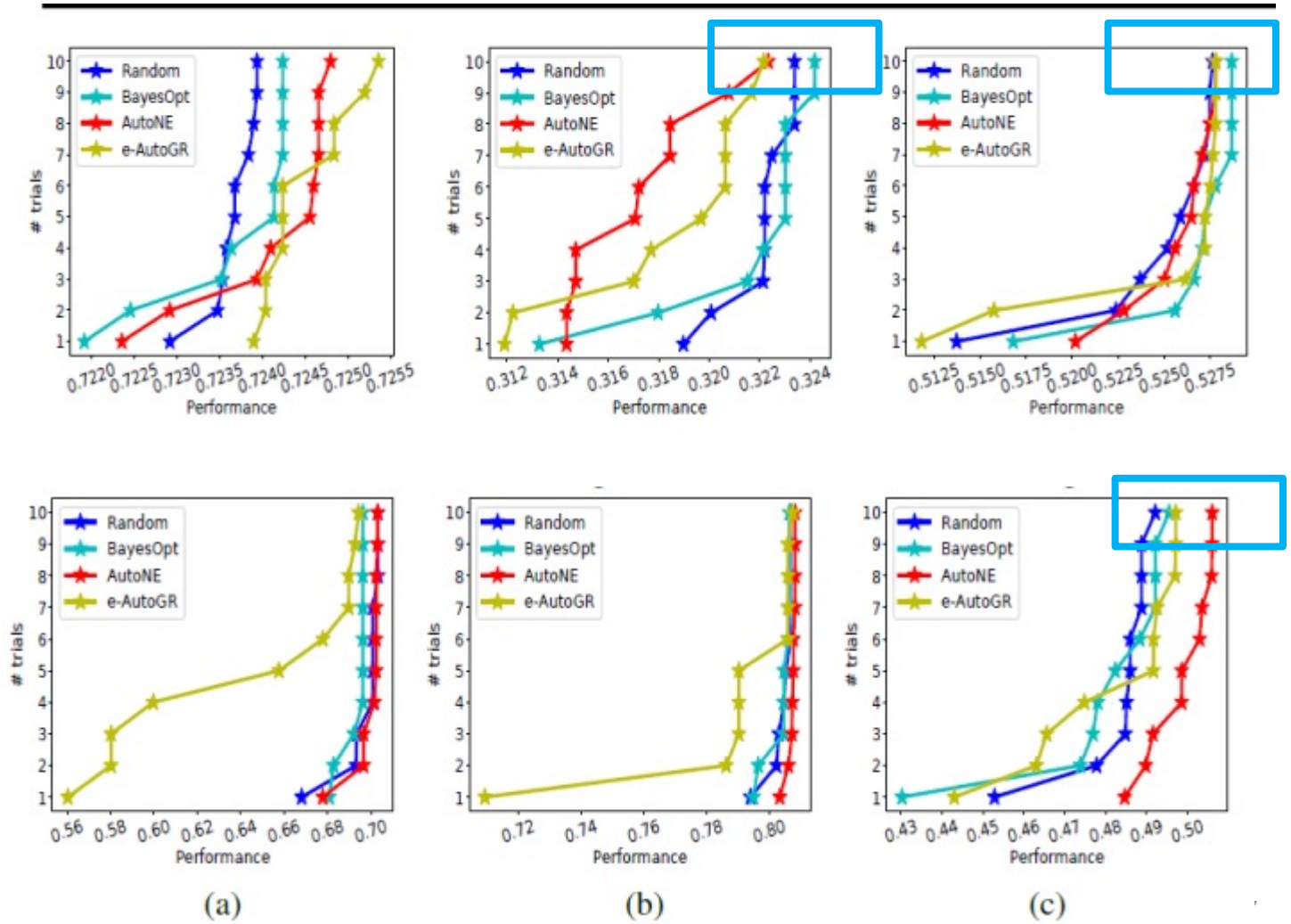
Best performance

- Best performance for each comparable automated graph representation approach on different datasets in terms of link prediction and node classification tasks over various graph representation algorithms.

Dataset	Algorithm	Task	e-AutoGR	AutoNE	Random	Bayesian
BlogCatalog	Deepwalk	Link Prediction	**0.871817**	0.792662	0.803191	0.807158
BlogCatalog	Deepwalk	Classification	**0.414682**	0.414234	0.411551	0.407449
BlogCatalog	AROPE	Link Prediction	**0.852612**	0.851921	0.846578	0.851878
BlogCatalog	AROPE	Classification	**0.326721**	0.325060	0.326020	0.326428
Wikipedia	Deepwalk	Link Prediction	0.729228	**0.729330**	0.696462	0.713133
Wikipedia	Deepwalk	Classification	**0.519657**	0.509920	0.503319	0.502617
Wikipedia	AROPE	Link Prediction	**0.709392**	0.709383	0.703443	0.707619
Wikipedia	AROPE	Classification	0.529743	0.529011	**0.530418**	0.529732
Pubmed	Deepwalk	Link Prediction	**0.873301**	0.867633	0.853459	0.851824
Pubmed	Deepwalk	Classification	**0.810916**	0.810368	0.809199	0.810417
Pubmed	AROPE	Link Prediction	0.791435	**0.796737**	0.790228	0.790123
Pubmed	AROPE	Classification	**0.727413**	0.725412	0.725789	0.726411
Pubmed	GCN	Classification	**0.708830**	0.708712	0.708719	0.707918

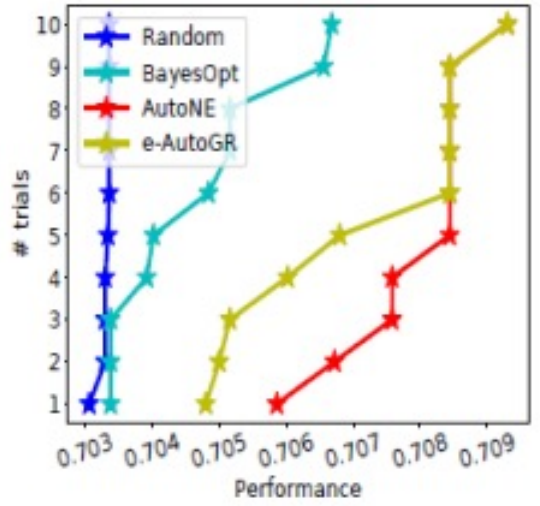
Node Classification Results

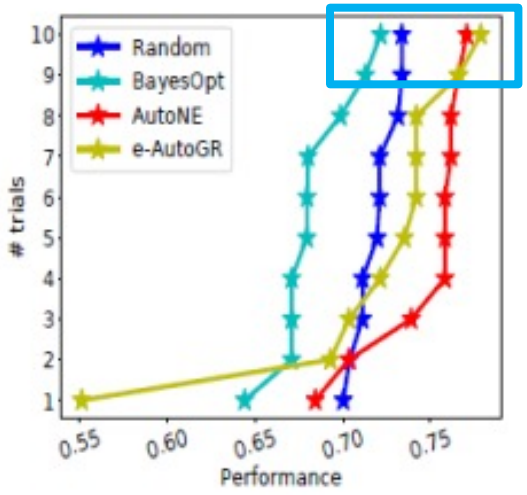
- Use AROPE
  - (a) on Pubmed.
  - (b) on BlogCatalog.
  - (c) on Wikipedia
- Use GCN and DeepWalk
  - GCN+Pubmed
  - DeepWalk+BlogCatalog
  - DeepWalk+ Wiki



Link Prediction Results

- Use AROPE
  - On Wikipedia
- Use DeepWalk
  - On Wiki
- For those not so good on e-AutoGR
  - (Author's explanation)
  - The random initialization process for the original large-scale graph leads to bad initial values of the hyperparameters
  - may be alleviated by utilizing the best hyperparameter settings in previous trials on sampled subgraphs.

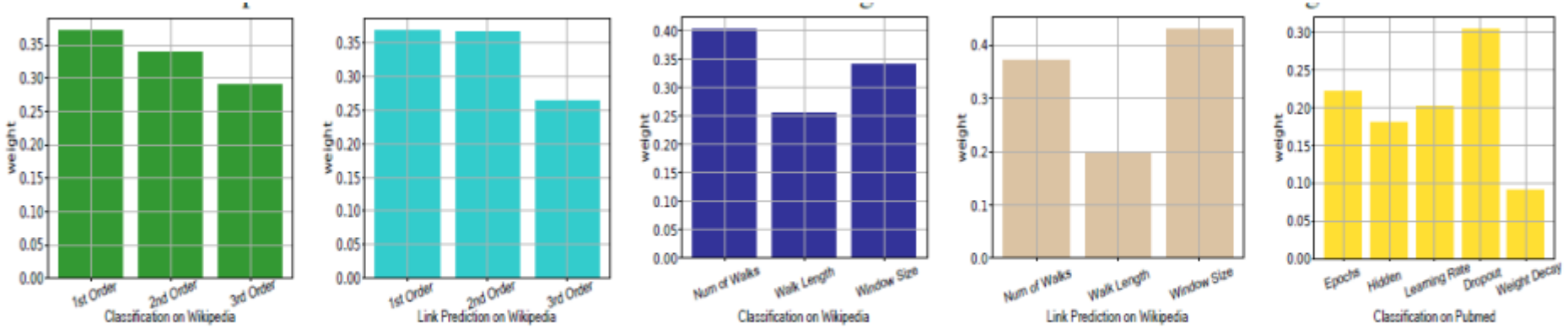




(d)

Hyperparameter Importance

- What is the importance of each hyperparameter?
- How it contributes to the model performance?
- Why one particular value is selected for a HP in the next trial?



(a) AROPE (b) AROPE (c) DeepWalk (d) DeepWalk (f) GCN

Figure 3. Explaining hyperparameter importance in affecting performance.

Outline

- Background
- Model Structure
- Experiments
- **Discussion**
  - Summary
  - Some ideas

Summary

- Investigate the problem of explainable AutoML for graph representation
- Propose e-AutoGR
  - Explaining how each hyperparameter contributes to the model performance
  - And why some certain values of hyperparameters are chosen for the next round of evaluationfully explainable graph features to dynamically determine the number of trials on each sampled subgraphs.
- Theoretically validate the correctness of the proposed hyperparameter decorrelation algorithm.

Graph Subsampling

- G = (V, E) ➔ Gi = (Vi, Ei) i = 1,2,…,s
  - Random Walks on original Graph
  - Start from different region(community) of G
  - Cover the property and diversity of G

- Any other method to show the relationship between G_i and G?

- Number of nodes: $|V|$
- Number of edges: $|E|$
- Number of triangles: The number of nodes forming a triangle which is a set of three nodes connected with each other by three edges.
- Global clustering coefficient: $\frac{3*Number\ of\ triangles}{Number\ of\ triplets}$, where a triplet is three nodes that are connected by either two or three (forming a triangle) undirected edges.
- Maximum total degree value: The total number of edges the most connected node in the graph has to other nodes.
- Number of Components: The total number of components contained in the graph, where a component is a subgraph in which there is a possible path between every node, while nodes from different components have no path connecting them.

We would also like to point out that increasing the size of subgraph will not always result in getting more computation resources. The reason is as follows. We aim to sample a series of representative subgraphs that share "some" similar properties with the original large-scale graph. We sample each subgraph based on several random walks. And we vary the length of each random walk to obtain subgraphs of various sizes. To ensure different subgraphs diversely preserve different properties of the original graph, the starting points of random walks are explicitly chosen to be at the different regions of the original graph. As different regions (e.g., local communities or local topologies with same category/label) may be of different sizes, larger subgraphs will not always be more similar to the original graph.

From AutoNE to e-AutoGR

Algorithm 1 *Automated Hyperparameter Optimization for Network Embedding* (AutoNE)

Input: Network G; Network embedding algorithm M.
Output: The optimal hyperparameter configuration θ_{opt}.

/* Phase I */
Sample S sub-networks from G according to Section 3.2. Each sampled sub-network will be reused for T times. We therefore use $\{G_i\}_{i=1}^{ST}$ to denote the ST sub-networks.
$\mathrm{X} = \{\}, \mathrm{f} = \{\}$.
for sub-network $i \leftarrow 1, 2, \ldots, ST$ **do**
    Compute the signature of G_i according to Section 3.3. The signature will be used by the kernel function $k(\cdot, \cdot)$.
    Select a hyperparameter configuration θ_i randomly.
    Run algorithm M on sub-network G_i using θ_i, and record the performance $f_M(\theta_i, G_i)$.
    $\mathrm{X} \leftarrow \mathrm{X} \cup \{(\theta_i, G_i)\}, \mathrm{f} \leftarrow \mathrm{f} \cup \{f_M(\theta_i, G_i)\}$.
end for
Update the kernel parameters by maximizing Equation 5.

/* Phase II */
Compute the signature of network G according to Section 3.3.
for trial $i \leftarrow ST+1, ST+2, \ldots, ST+L$ **do**
    Obtain a hyperparameter configuration θ_j that may achieve the optimal performance on G according to Equation 10.
    Run algorithm M on the input network G using θ_j, and record the performance $f_M(\theta_j, G)$.
    $\mathrm{X} \leftarrow \mathrm{X} \cup \{(\theta_j, G)\}, \mathrm{f} \leftarrow \mathrm{f} \cup \{f_M(\theta_j, G)\}$.
    Update the kernel parameters by maximizing Equation 5.
end for
return $\theta_{opt} = \arg\max_{\theta_j : ST+1 \le j \le ST+L} f_M(\theta_j, G)$.

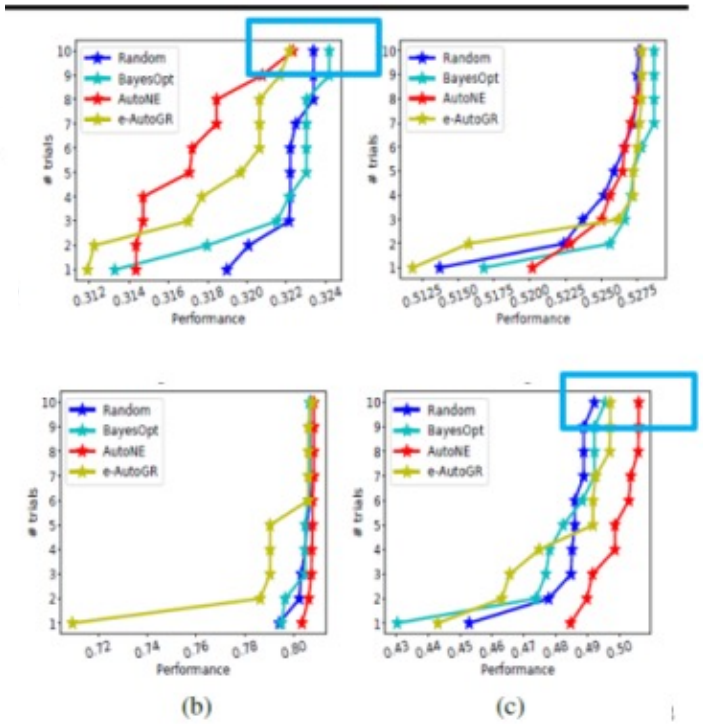
Algorithm 2 Explainable Automated Graph Representation (e-AutoGR)

Input: Graph G, Graph representation algorithm R.
Output: The optimal hyperparameter configuration Λ^*.
Sample s subgraphs $G_i, i = 1, 2, \ldots, s$ from original graph G according to Section 3.2.1.
Decide t_i for each G_i according to Section 3.2.2.
Execute algorithm R on each subgraph G_i for t_i times and obtain hyperparameter matrix A and graph feature matrix B as well as the performance vector Y.
Initialize $Count = T$.
repeat
    Execute **Step 1** to **Step 3** in Section 3.4.
    $Count = Count - 1$.
until $Count == 0$

- What's the difference?
- Surrogate model: Gaussian Process➔MLP for prediction.
- What else can we try?

Experiments Results

Dataset	Algorithm	Task	e-AutoGR	AutoNE	Random	Bayesian
BlogCatalog	Deepwalk	Link Prediction	**0.871817**	0.792662	0.803191	0.807158
BlogCatalog	Deepwalk	Classification	**0.414682**	0.414234	0.411551	0.407449
BlogCatalog	AROPE	Link Prediction	**0.852612**	0.851921	0.846578	0.851878
BlogCatalog	AROPE	Classification	**0.326721**	0.325060	0.326020	0.326428
Wikipedia	Deepwalk	Link Prediction	0.729228	**0.729330**	0.696462	0.713133
Wikipedia	Deepwalk	Classification	**0.519657**	0.509920	0.503319	0.502617
Wikipedia	AROPE	Link Prediction	**0.709392**	0.709383	0.703443	0.707619
Wikipedia	AROPE	Classification	0.529743	0.529011	**0.530418**	0.529732
Pubmed	Deepwalk	Link Prediction	**0.873301**	0.867633	0.853459	0.851824
Pubmed	Deepwalk	Classification	**0.810916**	0.810368	0.809199	0.810417
Pubmed	AROPE	Link Prediction	0.791435	**0.796737**	0.790228	0.790123
Pubmed	AROPE	Classification	**0.727413**	0.725412	0.725789	0.726411
Pubmed	GCN	Classification	**0.708830**	0.708712	0.708719	0.707918



- What's the reason that e-AutoGR fail to beat baseline?
- Can we choose strategy according to dataset?

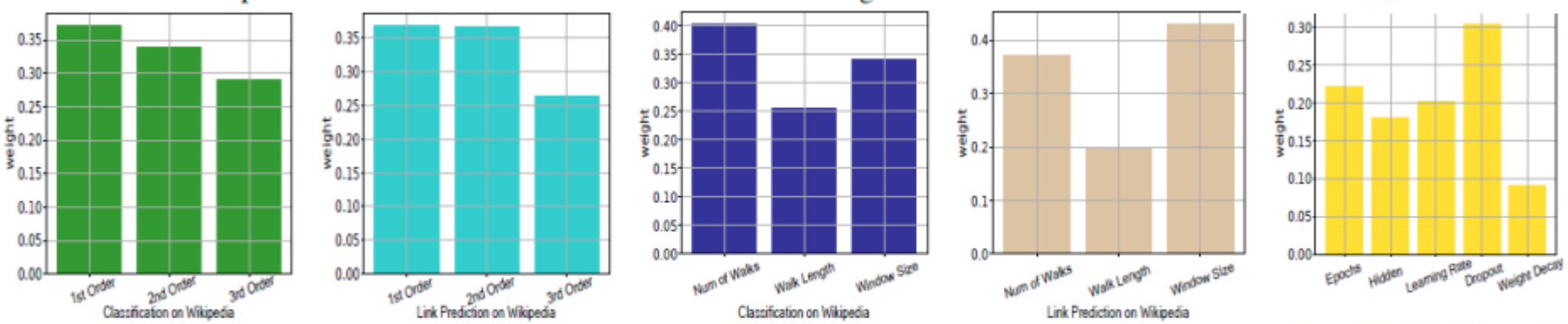
HPs importance

$$\min_{\gamma,\Theta} \sum_{i=1}^{n} \gamma_i \cdot (Y_i - \phi(\mathbf{X}_{i,}; \Theta))^2 \quad (8)$$

$$s.t \quad \sum_{j=1}^{p_1} \left\| \mathbf{A}_{,j}^T \mathbf{\Sigma}_\gamma \mathbf{X}_{,-j}/n - \mathbf{A}_{,j}^T \gamma/n \cdot \mathbf{X}_{,-j}^T \gamma/n \right\|_2^2 < \lambda_2$$

$$|\Theta|_1 < \lambda_1, \ \frac{1}{n} \sum_{i=1}^{n} \gamma_i^2 < \lambda_3,$$

$$(\frac{1}{n} \sum_{i=1}^{n} \gamma_i - 1)^2 < \lambda_4, \ \gamma \succeq 0,$$



(a) AROPE (b) AROPE (c) DeepWalk (d) DeepWalk

Figure 3. Explaining hyperparameter importance in affecting performance.

- Why does γ show decorrelation between n samples?
- Why do we use the first layer of MLP show importance?
- What if we don't optimize the hyperparameter by rank?(AutoNE)

- **Step 2:** We first rank hyperparameters in a descending order by their importance in affecting the performance, then optimize the most important hyperparameter with the others fixed. We uniformly sample 1000 possible values for the hyperparameter to be optimized, and search for the one with the best predicted performance through the learned non-linear mapping by MLP. Thus we traverse every hyperparameter one by one according to their importance and each time optimize one hyperparameter by fixing the others. The idea of using weights for the first layer of MLP to estimate the variable importance is inspired by previous work (Hassanpour & Greiner, 2019), and we conduct normalization for each hyperparameter i.e., $(x - min)/(max - min)$,to deal with the scale issue.

Decorrelation HPs?

$$\min_{\gamma,\Theta} \sum_{i=1}^{n} \gamma_i \cdot (Y_i - \phi(\mathbf{X}_{i,}; \Theta))^2 \quad (8)$$

$$s.t \quad \sum_{j=1}^{p_1} \left\| \mathbf{A}_{,j}^T \mathbf{\Sigma}_\gamma \mathbf{X}_{,-j}/n - \mathbf{A}_{,j}^T \gamma/n \cdot \mathbf{X}_{,-j}^T \gamma/n \right\|_2^2 < \lambda_2$$

$$|\Theta|_1 < \lambda_1, \ \ \tfrac{1}{n} \sum_{i=1}^{n} \gamma_i^2 < \lambda_3,$$

$$(\frac{1}{n} \sum_{i=1}^{n} \gamma_i - 1)^2 < \lambda_4, \ \ \gamma \succeq 0,$$

- Let $A_{,j}^T = [a_{1j}, a_{2j}, \dots, a_{nj}] \in R^{1\times n}$
- $X_{,-j} = [X_{1,-j}; X_{2,-j}; \dots; X_{n,-j}] \in R^{n\times p}$, $X_{k,-j}$是$X_{,-j}$第k行行向量
- $A_{,j}^T \Sigma_\gamma X_{,-j} = [a_{1j}, a_{2j}, \dots, a_{nj}] \cdot diag(\gamma_1, \gamma_2, \dots, \gamma_n) \cdot X_{,-j} = \sum_k \gamma_k \, a_{kj} X_{k,-j} \in R^{1\times p}$
- $A_{,j}^T \gamma = \sum_k \gamma_k a_{kj} \in R$
- $X_{,-j}^T \gamma = \sum_k \gamma_k X_{k,-j}^T \in R^{p\times 1}$
- $L = \sum_j \left(\sum_k \gamma_k \, a_{kj} X_{k,-j} - \sum_k \gamma_k a_{kj} \sum_k \gamma_k X_{k,-j}^T \right)$

where $\gamma \in \mathbb{R}^{n\times 1}$ are sample weights satisfying $\sum_{i=1}^{n} \gamma_i = n$, $\Sigma_\gamma = diag(\gamma_i, \dots, \gamma_n)$ is the corresponding diagonal matrix, $\mathbf{A}_{,j}$ denotes the j^{th} column/hyperparameter in $\mathbf{A}$, and $\mathbf{X}_{,-j} = \mathbf{X} \setminus \mathbf{A}_{,j}$ means all the remaining variables by removing the hyperparameter $\mathbf{A}_{,j}$ in $\mathbf{X}$.[1] The summand

Thank You!