GCN-LASE: Towards Adequately Incorporating Link Attributes in Graph Convolutional Networks

Ziyao Li$^1$, Liang Zhang$^1$ and Guojie Song$^{2*}$

$^1$Yuanpei College, Peking University, China
$^2$Key Laboratory of Machine Perception, Ministry of Education, Peking University, China
{leeezy, zls05, gjsong}@pku.edu.cn

Abstract

Graph Convolutional Networks (GCNs) have proved to be a most powerful architecture in aggregating local neighborhood information for individual graph nodes. Low-rank proximities and node features are successfully leveraged in existing GCNs, however, attributes that graph links may carry are commonly ignored, as almost all of these models simplify graph links into binary or scalar values describing node connectedness. In our paper instead, links are reverted to hypostatic relationships between entities with description attributes. We propose GCN-LASE (GCN with Link Attributes and Sampling Estimation), a novel GCN model taking both node and link attributes as inputs. To adequately captures the interactions between link and node attributes, their tensor product is used as neighbor features, based on which we define several graph kernels and further develop according architectures for LASE. Besides, to accelerate the training process, the sum of features in entire neighborhoods are estimated through Monte Carlo method, with novel sampling strategies designed for LASE to minimize the estimation variance. Our experiments show that LASE outperforms strong baselines over various graph datasets, and further experiments corroborate the informativeness of link attributes and our model’s ability of adequately leveraging them.

1 Introduction

After several attempts and modifications ([Bruna et al., 2014] [Niepert et al., 2016] [Kipf and Welling, 2016] [Tao et al., 2017] [Jie et al., 2018]), Graph Convolutional Networks (GCNs) $^1$ are rapidly gaining popularity due to its excellent performance in aggregating neighborhood information for individual graph nodes. Although low-rank proximities and node features are successfully leveraged through the convolutional layers, the attributes that graph links may carry are generally ignored in GCNs. In almost all existing GCNs, graph links are regarded as indicators of proximities between nodes. These proximities are only used to identify neighborhoods (and their influences if weighted) in the local neighborhoods.

However, in real-world scenarios, a link between a pair of nodes carries a lot more information than a simple indicator of neighborship. It represents a hypostatic relationship between two entities, usually with concrete attributes. For example, two connected people in a social network may have different relationships including family members, colleagues and alumni, and thus their communication patterns and contents may vary; a link in a business network represents a transaction between two companies, and the properties of such transactions are tremendously informative. Therefore, revisiting these link attributes, which are generally ignored in current GCNs, allows us to recover the exact relationships between nodes.

While one can leverage link attributes in GCNs with tricks, such as concatenating them to neighbor node attributes, these implementations cannot adequately captures the interactions between the attributes $^2$. To the best of our knowledge, there is no previous work focusing on incorporating link attributes into GCNs. We propose GCN-LASE (Graph Convolutional Network with Link Attributes and Sampling Estimation) as an attempt. LASE is an extension of GCN, which also learns a function that maps a target node to its hidden representation considering the local neighborhood features and structures, but including both link and neighbor node attributes. The aggregated features are then used to conduct downstream tasks.

To leverage the link and node attributes as well as the interactions in between, we adopt their tensor products as the fully associated neighbor features, based on which a neighbor kernel is developed using the inner products of tensors. We further derive corresponding graph kernels and finally the neural architectures following a similar route introduced in [Tao et al., 2017]. An intuitive understanding of LASE is then

$^*$This work was supported by the National Natural Science Foundation of China (Grant No. 61876006 and No. 61572041).

$^1$Corresponding author.

$^2$Although some papers do not explicitly use the term convolution, such as [Tao et al., 2017] and [Hamilton et al., 2017], a similar idea of aggregating features in a node’s neighborhood can be seen in the models. Therefore, we categorize all similar work into the class of graph convolutional networks.

$^2$See Section 3 for detailed introduction.
demonstrated, by modularizing it into a gate, an amplifier, and an aggregator. Meanwhile, we adopt the Monte Carlo method to quickly estimate the sum of features in neighborhoods. We also introduce a novel sampling setup for LASE to reduce the estimation variance.

Recovering more information is not the only benefit LASE brings about; it also enlarges the expressive abilities of graph structures that GCNs can handle. At least two other types of graph-structured data can be handled with LASE: i) Graphs with heterogenous links, where link weights from different perspectives can be arranged into vectors and used as attributes; ii) Dynamic or temporal graphs, where link weights from different time-stamps can be stacked together and used as attributes. See Figure 1 for detailed examples. We validate our approach on four datasets across different domains, and further design additional experiments in order to demonstrate the informativeness of link attributes and the effect of our sampling strategy.

2 Related Work

Our method builds upon previous work of machine learning on graph data, including graph convolutional networks, graph kernels and node representation learning.

Graph convolutional networks. The past few years have seen plenty of works focusing on implementing deep architectures over graphs ([Bruna et al., 2014] [Henaff et al., 2015] [Kipf and Welling, 2016] [Wang et al., 2016] [Niepert et al., 2016] [Hamilton et al., 2017] [Tao et al., 2017]), among which the convolutional networks seem to be the most appealing. There are mainly two types of convolutional networks existing, one learning features for entire graphs ([Tao et al., 2017] [Niepert et al., 2016]), the other for individual nodes ([Kipf and Welling, 2016] [Hamilton et al., 2017]). Both methods interpret the concept of convolution as merging node features in local neighborhoods. Meanwhile, as early GCNs ([Kipf and Welling, 2016]) do not support mini-batch training, modifications towards better efficiency emerge ([Hamilton et al., 2017] [Jie et al., 2018] [Huang et al., 2018]), in which the Monte Carlo method is generally used to estimate neighborhood representations through a controllable size of nodes. In our paper, a different sampling implementation is adopted, which makes a tradeoff between variance and efficiency by controlling the interval of calculating the optimal sampling probabilities.

Graph kernels. Kernel methods ([Schölkopf and Smola, 2002] have long been an important class of machine learning techniques, while it remains challenging to define effective and convenient kernels for graphs. Existing graph kernels ([Gärtner et al., 2003] [Vishwanathan et al., 2008] [Szegedy et al., 2011]) are typically defined over sub-structures in graphs, such as sub-trees and walk sequences. Later, [Tao et al., 2017] introduces an innovative route to develop neural architectures using graph kernels. To the best of our knowledge, there is no existing work aiming at incorporating link attributes into graph kernels. In our paper, we adopt a similar route as [Tao et al., 2017] to design LASE, but using novel graph kernels which are able to handle link attributes.

Node representation learning (NRL). NRL aims to learn low-dimensional embeddings for graph nodes ([Perozzi et al., 2014] [Tang et al., 2015] [Grover and Leskovec, 2016] [Ribeiro and Sales-Pardo, 2017] [Hamilton et al., 2017] [Li et al., 2018] [Du et al., 2018]), which are later used in downstream prediction tasks such as node classification and link prediction. GCNs can, in a broader sense, also be classified as NRL methods when regarding the hidden layers of nodes as embedding vectors. However, few existing NRL models incorporate link attributes. In Section 5, we compare the performances of LASE with several other NRL approaches.

3 Model

In this section, we introduce the architecture of LASE (see Figure 2) and the motivation behind. We first extend the con-
cept of a neighbor, and the features of a neighbor is then defined as the tensor product of its node and link attributes. As directly using these tensors of features in GCNs would be clumsy, we design new graph kernels under this setup, based on which we further derive possible architectures of LASE. In the end, we modularize LASE and provide intuitive understandings of the modules’ functions. Notations in this paper are illustrated in Table 1. Vectors in this paper are all formed in columns.

### 3.1 Neighbor Feature Tensors

One simplest idea to incorporate link attributes in GCN models is to concatenate them to the node’s attributes, i.e.

$$h^{(i+1)}(u) = \sigma \left( \sum_{v \in N(u)} W \left[ \frac{h^{(i)}(v)}{f(e_{u,v})} \right] \right)$$

where $W = [W_1, W_2]$. However, as attributes of neighbor nodes and neighbor links are individually summed among the neighborhood, such implementations can not at all capture the interactions within a (node, link) neighbor. As the key idea behind LASE is to adequately incorporate link attributes into node hidden representations, these interactions cannot be ignored. Moreover, this setup also leads to the confusion demonstrated in Figure 3, indicating that the graph structure is not appropriately captured.

Instead of simply adding or concatenating node and link attributes, we define their tensor product as the associated neighbor feature. The calculated tensor contains all bilinear combinations of the two attributes, and serves as a fully connected feature. Formally, for the central node $u$ connected to $v$ by a link $e_{u,v}$, the corresponding neighbor feature is defined as

$$f((v, e_{u,v})) := f(v) \otimes f(e_{u,v}).$$

However, directly using the tensor as inputs leads to unacceptably high dimensionalities and heavy redundancies, for the feature matrix (i.e. the 2D-tensor) is only of rank 1. Instead, we adapt existing graph kernels for so-defined neighbor features, and derive a concise architecture for LASE following [Tao et al., 2017]’s route.

### 3.2 Graph Kernels with Link Attributes

To adapt existing kernels to our setup of neighbor features, we first define the kernel of two neighbors, $(v, e, v')$ and $(w, e, w)$. The neighbor kernel is defined as the inner product of the neighbor tensors, i.e.

$$K((v, e, v'), (w, e, w)) := \langle f((v, e_{u,v})), f((w, e_{u,w})) \rangle = \langle f(v), f(w) \rangle \cdot \langle f(e_{u,v}), f(e_{u,w}) \rangle.$$  

Based on the neighbor kernel, a kernel of two $l$-hop neighborhoods with central node $u$ and $u'$ can be defined as

$$K^{(l)}_{N}(u, u') := \begin{cases} 
\langle f(u), f(u') \rangle & l = 0 \\
\langle f(u), f(u') \rangle \cdot \lambda \\
\sum_{v \in N(u)} \sum_{v' \in N(u')} K^{(l-1)}_{N}(v, v') \\
\langle f(e_{u,v}), f(e_{u',v'}) \rangle & l > 0 
\end{cases}$$

Table 1: Notations in this paper.

| $G = (V, E)$ | Input graph. |
| $u, v, w, \cdots$ | Nodes in $G$. |
| $e_{u,v}$ | A link from node $u$ to $v$. |
| $(v, e_{u,v})$ | A pair of node and link, or to say a neighbor of node $u$. |
| $f(\cdot)$ | The feature of a node, a link or a pair. |
| $N(u)$ | The set of neighbor nodes of $u$. |
| $\otimes, \odot, \langle \cdot, \cdot \rangle$ | The operations of tensor product, element-wise product and inner product. |
| $[\cdot, \cdots, \cdot], [\cdot]_{W}, \oplus$ | The operation of concatenating input vectors. |
| $W, U, V, \cdots$ | The parameters in the neural network. |
| $h^{(i)}(u)$ | The hidden representation of node $u$ in layer $i$. |

![Figure 2: The architecture of LASE.](image)

![Figure 3: Simply concatenating node and link attributes does not capture the interactions in between. It cannot distinguish between the two situations above.](image)

$^4$The term neighbor is used for an ordered pair, containing a neighbor node and the link connecting it to the central node, similarly hereinafter.

$^5$Briefly speaking, the tensor product of two (column) vector $a$ and $b$ is calculated as $ab^T$ with shape $(d_a \times d_b)$.
by regarding the lower-hop kernel, \( K_N^{(l-1)}(v, v') \), as the inner product of the \((l-1)\)-th hidden representations of \( v \) and \( v' \), \( \lambda \in (0, 1) \) is a decay factor. Furthermore, by recursively applying the neighborhood kernel, we can derive the l-hop Random Walk kernel for graphs with link attributes as

\[
K^l_W(G, G') = \lambda^{l-1} \sum_{u \in P_l(G)} \sum_{u' \in P_l(G')} \left( \prod_{i=0}^{l-2} \langle f(u_i), f(u'_i) \rangle \right) \times \prod_{i=0}^{l-1} \langle f(e_{u_i,u_{i+1}}), f(e_{u'_i,u'_{i+1}}) \rangle,
\]

where \( P_l(G) \) denotes the set of all walk sequences of length \( l \) in graph \( G \), and \( u_i \) denotes the \( i \)-th node in sequence \( u \).

### 3.3 From Kernels to Neural Architectures

Following the route introduced in [Tao et al., 2017] with above Random Walk kernel, a corresponding architecture LASE-RW can be immediately derived as

\[
\begin{align*}
\lambda_{u,v}^{(l)} &= \sigma \left( V^{(l)} h^{(l)}(u), f(e_{u,v}), h^{(l)}(v) \right) + b^{(l)} \\
h^{(0)}(u) &= W^{(0)} f(u) \\
h^{(l)}(u) &= \sum_{v \in N(u)} \lambda_{u,v} h^{(l-1)}(u) \odot U^{(l)} f(e_{u,v}) \odot W^{(l)} f(v).
\end{align*}
\]

This architecture further enjoys a similar property to the architecture described in [Tao et al., 2017]. We first construct \( L_{t,k} = (V_L, E_L) \) using the k-th row vectors in the parameter matrices as node and link attributes. The node set \( V_L = \{ v_0, \ldots, v_l \} \) with \( f(v_i) = W^{(i)} v \), and the link set \( E_L = \{ e_{v_i,v_{i+1}} \} \) with \( f(e_{v_i,v_{i+1}}) = U^{(i+1)} \). Then, we have the following theorem:

**Theorem 1.** For any \( l \geq 1 \), the sum of \( h^{(l)}(v)[k] \) (the \( k \)-th coordinate of \( h^{(l)}(v) \)) satisfies

\[
\sum_{v \in G} h^{(l)}(v)[k] = K^l_W(G, L_{t,k}),
\]

and thus \( \sum_{v \in G} h^{(l)}(v)[k] \) lies in the RKHS of kernel \( K^l_W \).

Similarly, the architecture LASE-WL derived from Weisfeiler-Lehman Kernel should be adapted as

\[
\begin{align*}
\lambda_{u,v}^{(l)} &= \sigma \left( V^{(l)} h^{(l)}(u), f(e_{u,v}), h^{(l)}(v) \right) + b^{(l)} \\
h^{(0)}(u) &= W^{(0)} f(u) \\
h^{(l)}(u) &= \sum_{v \in N(u)} \lambda_{u,v} h^{(l-1)}(v) \odot U^{(l)} f(e_{u,v}) \odot W^{(l)} f(v).
\end{align*}
\]

The Weisfeiler-Lehman architecture is originally designed to convolute nodes through both depth (\( r \)) and breadth (\( h \)), however, the calculation of LASE-WL would be too complex. We unite the depth and breadth convolution to reduce model size, and by referring to the neighborhood aggregation concept in GraphSAGE [Hamilton et al., 2017], proposed LASE-SAGE:

\[
\begin{align*}
\lambda_{u,v}^{(l)} &= \sigma \left( V^{(l)} h^{(l)}(u), f(e_{u,v}), h^{(l)}(v) \right) + b^{(l)} \\
h^{(0)}(u) &= f(u) \\
h^{(l)}(u) &= \sigma \left( W_1^{(l)} h^{(l-1)}(u) \odot W_2^{(l)} f(e_{u,v}) \right) \\
&= \sum_{v \in N(u)} \lambda_{u,v} h^{(l-1)}(v) \odot U^{(l)} f(e_{u,v}),
\end{align*}
\]

where \( \odot \) is a combination operation such as +, \( \odot \) or \( \odot \).

### 3.4 Discussion

Although [Tao et al., 2017] is originally introduced for aggregating features for entire graphs, its output graph features are an activated sum of all nodes features. We reckon these node features be as well informative in node-wise prediction tasks. We also provide an intuitive understanding of LASE. The calculations in LASE can be divided into three common modules, namely a *gate*, an *amplifier* and an *aggregator*, as shown in Figure 2. Intuitively, the gate \( \lambda_{u,v} \) evaluates v’s influence in u’s neighborhood. The amplifier \( h(v) \), element-wisely amplifies the node attributes using link information. The aggregator sums up neighbor embeddings and combines them with the central node embedding using various strategies. Aggregators proposed in [Hamilton et al., 2017] may also be used in LASE.

### 4 Sampling Estimation

Similar to GCN [2016], scalability is an obvious challenge for LASE: calculating the convolutions demands a recursively expanded neighborhood. For nodes with high degrees, it will quickly cover a large portion of the graph. To control batch scales, we leverage the Monte Carlo method to estimate the summed neighborhood information by sampling a fixed number of neighbors. Despite different architectures, the output hidden embeddings of LASE can all be formulated as

\[
\begin{align*}
h^{(l)}(u) &= \sigma_* \left( \sum_{v \in N(u)} \lambda_{u,v} g^{(l)}(v|u) \right) \\
&= \sigma_* \left( E_{p^{(l)}(v|u)} \left[ \frac{\lambda_{u,v} g^{(l)}(v|u)}{p^{(l)}(v|u)} \right] \right)
\end{align*}
\]

where \( p^{(l)}(v|u) \) denotes the sampling probabilities in \( N(u) \). We then approximate \( h^{(l)}(u) \) through estimating the expectation. As the sampling process is always unbiased, we look for

\[^{5}\text{A slight elevation in performance can be observed when applying a sigmoid activation on } U f(e_{u,v}), \text{ which makes the vector functions more analogously to an amplifier.}\]
the optimal probabilities that minimize the estimation variance.

Although there are existing sampling strategies proposed for GCNs ([Jie et al., 2018] [Huang et al., 2018]), these methods cannot be directly transferred to LASE because of the absence of explicit, constant link weights. Besides, the optimal distribution varies through the training process. However, a similar idea of importance sampling, coined gate sampling, can be used in LASE by regarding the decay factor $\lambda$ as the sampling weights, that is,

$$p_{gate}^{(t)}(v|u) = \frac{\lambda_{u,v}^{(t)}}{\sum_{w \in N(u)} \lambda_{u,w}^{(t)}}.$$

While sampling with gates may reduce the estimation variance, it is not an optimal solution because typically the norms of $g^{(t)}(v|u)$s are different. According to the derivations of importance sampling in [Owen, 2013], we derive min_var sampling, the optimal sampling probabilities as

$$p_{\text{min-var}}^{(t)}(v|u) = \frac{\lambda_{u,v}^{(t)} ||g^{(t)}(v|u)||}{\sum_{w \in N(u)} \lambda_{u,w}^{(t)} ||g^{(t)}(w|u)||}.$$

Evaluating the sampling probabilities batch-wisely can be rather inefficient. Under the hypothesis that the network parameters do not dramatically vary from batch to batch, we make a tradeoff between variance and efficiency by controlling the interval of calculating the optimal distribution. That is, the sampling probabilities for all training nodes are calculated every $k$ batches. Although the calculation may be time-consuming, the batch-averaged time cost will be reduced to $1/k$.

5 Experiments

5.1 Experiment Setups

Datasets. We validate our method on the four datasets introduced below, including a graph with link attributes (reddit), a graph with heterogenous links (dblp), and two temporal networks (email, fmobile). The statistics of datasets are shown in Table 2.

- Reddit is a Reddit post network with each node representing a post, and each link indicating that the connected two posts are commonly commented by at least three users. We adopt the same setup as [Hamilton et al., 2017] for the node attributes, and use the user-averaged distributions of comments in different communities as link attributes.
- Dblp is a co-author network constructed with papers from 2013 to 2017 in eight artificial intelligence conferences. We use the tf-idf vectors of paper titles as node attributes. The links are categorized under author perspective, i.e. the one-hot embeddings of the common authors are used as link attributes. The node and link attributes are reduced to 200 dimensions using PCA.
- Email and fmobile are two temporal networks constructed with user contacts in email and mobile-phone services. The contacts of exact times are discretized into time-slices and used as link attributes. As there is no available node features in the datasets, we use node embeddings with $dim=128$ obtained from transductive-LINE [2015] as the pre-trained node features in all convolution-based models.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>$n_{nodes}$</th>
<th>$n_{links}$</th>
<th>$d$</th>
<th>$n_{labels}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>reddit</td>
<td>61,836</td>
<td>1,222,411</td>
<td>19.77</td>
<td>8</td>
</tr>
<tr>
<td>dblp</td>
<td>14,389</td>
<td>111,858</td>
<td>7.77</td>
<td>8</td>
</tr>
<tr>
<td>email</td>
<td>986</td>
<td>16,064</td>
<td>16.29</td>
<td>42</td>
</tr>
<tr>
<td>fmobile</td>
<td>21,102</td>
<td>55,009</td>
<td>2.61</td>
<td>33</td>
</tr>
</tbody>
</table>

Table 2: Statistics of datasets used in this paper.

Baselines. We compare the performance of LASE with baselines including raw features, LINE [2015], DeepWalk [2014], GCN [2016] and GraphSAGE [2017]. For LINE and DeepWalk, we adopt an online-style training strategy for the test / validation set introduced in [Hamilton et al., 2017]⁹, and a one-layer softmax-activated neural classifier is trained for all models. To demonstrate the ability of LASE in leveraging link attributes through the amplifiers, we also test the performance of a LASE variant, LASE-concat, implemented by naïvely concatenating link attributes to node attributes.

5.2 Node-wise Classification

We implement node-wise classification respectively over the four datasets mentioned above by predicting the community (reddit, email and fmobile) or the conference (dblp) that a node belongs to. In all datasets, 65% nodes are used as the training set, 15% as the validation set and the rest as the test set. The nodes in the training set with no neighbors are abandoned. The micro-averaged F1 scores on the test set are shown in Table 3.

As one of the most distinguished strengths of GCNs is to aggregate neighborhood features, convolutional-based models including GCN, GraphSAGE and LASE show significant advantages to proximity-based models on datasets with node attributes. Through leveraging link attributes, LASE outperforms other GCNs. Moreover, with LASE-RW and LASE-SAGE outperforming the naïve implementation LASE-concat, the effect of the amplifier module can be corroborated. Although there is no original features in two temporal networks, LASE still outperforms pre-trained features by exploring edge attributes, while GCN and GraphSAGE do not capture these additional information and struggles in over-fitting the proximity-based features.

Figure 4 a) demonstrate the accuracies of LASE-SAGE using contaminated link attributes of different signal-to-noise ratios (SNRs). That is, we add normal-distributed noises of different standard deviations to the original link attributes according to given SNRs, and separately train LASE-SAGE

⁹As there is no implementation of online-LINE and Qiu et al., 2018] proves that LINE is theoretically equivalent with DeepWalk with $walk \text{ length}=1$, we use the implementation of online-DeepWalk in [Hamilton et al., 2017] instead. $N_{walks}$ is respectively added to compensate the reduction in node contexts.
Table 3: Performances of node-wise prediction tasks of LASE, its variants and baselines (Micro-f1s). *: Comparisons between these results and those of convolutional models would be considered unfair as the latter uses transductively learned features as inputs.

<table>
<thead>
<tr>
<th>Model</th>
<th>reddit</th>
<th>dblp</th>
<th>email</th>
<th>fmobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE (online)</td>
<td>0.1802</td>
<td>0.2989</td>
<td>0.3604*</td>
<td>0.3047*</td>
</tr>
<tr>
<td>DeepWalk (online)</td>
<td>0.1714</td>
<td>0.3306</td>
<td>0.3249*</td>
<td>0.4071*</td>
</tr>
<tr>
<td>GCN</td>
<td>0.8172</td>
<td>0.5033</td>
<td>0.6396</td>
<td>0.3908</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>0.8468</td>
<td>0.5798</td>
<td>0.6548</td>
<td>0.5334</td>
</tr>
<tr>
<td>LASE-concat</td>
<td>0.8438</td>
<td>0.5433</td>
<td>0.7028</td>
<td>0.5441</td>
</tr>
<tr>
<td>LASE-RW</td>
<td>0.8460</td>
<td>0.5433</td>
<td>0.7208</td>
<td>0.5441</td>
</tr>
<tr>
<td>LASE-SAGE</td>
<td>0.8633</td>
<td>0.5881</td>
<td>0.7310</td>
<td>0.5649</td>
</tr>
<tr>
<td>Raw Features</td>
<td>0.7923</td>
<td>0.4532</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LINE (transd.)</td>
<td>-</td>
<td>-</td>
<td>0.6904</td>
<td>0.4749</td>
</tr>
</tbody>
</table>

Figure 4: Analyses of LASE: a) the prediction accuracies with contaminated link attributes. With higher SNR, the test accuracy of LASE on reddit significantly drops; b) the validation accuracies on email with different sampling strategies; c) the validation accuracies on email with different intervals of calculating the optimal sampling probabilities for min_var.

models under identical model settings. The SNR is defined as

\[ SNR = \frac{A(link\_attrs)}{A(noises)} \]

where \( A(\cdot) \) denotes the standard deviation of the inputs. As SNR increases, a significant trend of decrease in accuracy can be observed. This corroborates the informativeness of link attributes in the LASE architecture.

5.3 Comparison of Sampling Strategies

We look into the training processes of different neighborhood sampling strategies introduced in Section 4, namely uniform, gate and minimal variance (min_var) sampling. We separately train models with corresponding sampling strategy on email, and present the variations of accuracies on the validation set against training epochs in Figure 4 b). While the convergence speeds appear analogous, min_var sampling consistently attains better convergence performance compared with uniform and gate sampling. The reason that gate sampling does not show a significant advantage over uniform sampling may be that the norms of transformed neighbor features \( g(v[u]) \) varies greatly in the neighborhood.

Figure 4 c) shows the tradeoff between performance and efficiency made through different calculation intervals of the sampling distribution (under min_var setup). As the interval \( k \) increases, the performance slightly drops. Calculating the probabilities batch-wise attains a significant elevation in the performance, while the computation cost can be unacceptably high on larger datasets. Additionally, when \( k \) becomes large enough, increasing \( k \) does not significantly influence the training performance.

6 Conclusions and Future Work

In this paper, we propose LASE as an extension of graph convolutional networks, which leverages more information from graph links than existing GCNs by incorporating the link attributes. The contribution of LASE lies in three folds: i) LASE provides a ubiquitous solution to a wider class of graph data by incorporating link attributes; ii) LASE outperforms strong baselines and naïve concatenating implementations by adequately leveraging the information in the link attributes; iii) LASE adopt a more explainable approach in determining the neural architecture and thus enjoys better explainability.

For future work, we are looking for better sampling solutions for LASE, as although stressed with calculation intervals, current sampling setup seems to be rather clumsy when the graph becomes massively large. We are also looking for other possible approaches, hopefully with better performance, to incorporating link attributes. Besides, as LASE is an universal solution to all graph-structured data, an intriguing direction may be designing domain- or task-specific architectures based on LASE to attain better performances, such as more elegant adaptations to dynamic networks.
References


