Overlapping Community Detection Using Graph Attention Networks

Ανιχνευση Επικαλυπτομενων Κοινοτητων σε Γραφους με Δικτυα Προσοχης

Μεταπτυχιακή Διπλωματική Εργασία

Κωνσταντίνος Σισμάνης

Επιβλέπων Αριστείδης Παγουρτζής, Καθηγητής
Συνεπιβλέπουσα Θεοδώρα Σούλιου, ΕΔΙΠ

Εθνικο Μετσοβιο Πολυτεχνειο
Διατμηματικο Προγραμμα Μεταπτυχιακων Σπουδων
Επιστημη Δεδομενων & Μηχανικη Μαθηση
Αθήνα, Δεκέμβριος 22

Overlapping Community Detection Using Graph Attention Networks

Ανιχνευση Επικαλυπτομενων Κοινοτητων σε Γραφους με Δικτυα Προσοχης

Μεταπτυχιακή Διπλωματική Εργασία

Κωνσταντίνος Σισμάνης

Επιβλέπων Αριστείδης Παγουρτζής, Καθηγητής
Συνεπιβλέπουσα Θεοδώρα Σούλιου, ΕΔΙΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 1η Δεκεμβρίου 2022

Αριστείδης Παγουρτζής	Γιώργος Στάμου	Δημήτριος Φωτάκης
Καθηγητής	Καθηγητής	Αναπληρωτής Καθηγητής

Εθνικο Μετσοβιο Πολυτεχνειο
Διατμηματικο Προγραμμα Μεταπτυχιακων Σπουδων
Επιστημη Δεδομενων & Μηχανικη Μαθηση
Αθήνα, Δεκέμβριος 22

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή Αριστείδη Παγουρτζή, και την συνεπιβλέπουσα Θεοδώρα Σούλιου για την καθοδήγηση, και την στήριξη τους, που υπήρξαν καταλυτικές για την ολοκλήρωση της.

Επίσης θα ήθελα να ευχαριστήσω τους συμφοιτητές, τους εργαζόμενους και τους καθηγητές στο Εθνικό Μετσόβιο Πολυτεχνείο, για τη συνεργασία και τις γνώσεις που προσέφεραν κατά τη διάρκεια των σπουδών.

Περίληψη

Η ανίχνευση κοινοτήτων σε γράφους, έχει αποδώσει σημαντικά αποτελέσματα σε εφαρμογές που εκτείνονται από τα κοινωνικά δίκτυα και τα συστήματα συστάσεων, έως τα δίκτυα αλληλεπίδρασης πρωτεϊνών και τα δίκτυα νευρώνων στον εγκέφαλο.

Η χρήση μεθόδων Βαθιάς Μάθησης έχει επιτύχει σημαντικά αποτελέσματα στην επίλυση προβλημάτων σε γράφους, όπως είναι η ταξινόμηση κόμβων και γράφων και η πρόβλεψη ακμών.

Η μηχανική μάθηση έχει αρχίσει να εφαρμόζεται και στην περιοχή της αναζήτησης κοινοτήτων με συνεχώς αυξανόμενη επιτυχία. Ωστόσο οι περισσότερες ερευνητικές προσπάθειες επικεντρώνονται στην αναζήτηση μη-επικαλυπτόμενων κοινοτήτων. Λιγότερες έρευνες ασχολούνται με τις επικαλυπτόμενες κοινότητες σε γράφους, χρησιμοποιώντας μεθόδους βαθιάς μάθησης.

Σε αυτή την εργασία επεκτείνουμε μια μέθοδο που έχει αποφέρει πολύ καλά αποτελέσματα στο πρόβλημα της ανίχνευσης επικαλυπτόμενων κοινοτήτων. Η συγκεκριμένη μέθοδος, συνδυάζει μια πιθανοτική οπτική στο πρόβλημα μαζί με την μάθηση αναπαραστάσεων, με τη χρήση Νευρωνικών Δικτύων Γράφων.

Η πρόταση μας έγκειται στην προσθήκη ενός μηχανισμού "προσοχής" που θα επικεντρώνεται στα σημαντικά σημεία των δεδομένων. Διερευνούμε αν ένας μηχανισμός προσοχής θα μπορεί να διακρίνει ποιος κόμβος είναι πιο σημαντικός σε μια γειτονιά κόμβων.

Παρουσιάζουμε την πειραματική διαδικασία που πραγματοποιήθηκε για την αξιολόγηση της πρότασης, διερευνώντας την ικανότητα ανακάλυψης των πραγματικών κοινοτήτων. Η πρόταση μας, στα μεγαλύτερα σύνολα δεδομένων που εξετάστηκαν και που διαθέτουν περισσότερη πληροφορία για τα χαρακτηριστικά των κόμβων, επιτυγχάνει βελτίωση στην ικανότητα ανακάλυψης των επικαλυπτόμενων κοινοτήτων.

Σύνοψη

Πολύπλοκα συστήματα όπως τα τηλεπικοινωνιακά δίκτυα, τα δίκτυα κοινωνικής δικτύωσης, τα δίκτυα αλληλεπίδρασης πρωτεϊνών, αναπαρίστανται με τη μορφή γράφων. Η αναζήτηση κοινοτήτων σε γράφους βοηθά στην κατανόηση της δομής και της λειτουργίας των συστημάτων. Οι επιστήμες της κοινωνιολογίας και της κοινωνικής ανθρωπολογίας μελετούν την διαμόρφωση κοινοτήτων από την δεκαετία του 1920. Ωστόσο πρόσφατα στις αρχές του 21ου αιώνα επιτυγχάνεται η μελέτη της δημιουργίας δικτύων βασισμένη επάνω σε πραγματικά δεδομένα. Οι Girvan και Newman[20] ασχολήθηκαν πρώτοι με το πεδίο που ονομάστηκε graph clustering. Από τότε πολλές μελέτες έχουν προτείνει μεθόδους ανίχνευσης κοινοτήτων[53]. Τελευταία πολλές μέθοδοι συνδυάζουν την χρήση της βαθιάς μάθησης.

Ορισμός Προβλήματος Θεωρούμε έναν μη κατευθυνόμενο, μη ζυγισμένο γράφο $\mathcal{G}(\mathbf{V}, \mathbf{E})$, που μπορεί να αναπαρασταθεί με έναν δυαδικό πίνακα γειτνίασης $\mathbf{A} \in (0, 1)$. όπου N ο αριθμός των κόμβων $\mathbf{V} = \{1, \ldots, N\}$ και M ο αριθμός των ακμών $\mathbf{E} = \{(u, v) \in V \times V : A_{uv} = 1\}$. Ίσως ορίζεται ένας πίνακας με χαρακτηριστικά των κόμβων $\mathbf{X} \in \mathbb{R}^{N \times D}$ που μπορεί όμως να μην είναι διαθέσιμος.

Στόχος της αναζήτησης επικαλυπτόμενων κοινοτήτων είναι η ανάθεση των κόμβων σε C κοινότητες. Μια ανάθεση μπορεί να αναπαρασταθεί ως ένας μη αρνητικός πίνακας συμμετοχής σε κοινότητες $\mathbf{F} \in \mathbb{R}_{\geq 0}^{N \times C}$, όπου $\mathbf{F}_{uc}$ είναι ο βαθμός συμμετοχής του κόμβου u στην κοινότητα c .

Το πρόβλημα μπορεί να θεωρηθεί ως ένα πρόβλημα πιθανοτικής επαγωγής [63]. Αν διατυπωθεί ένα μοντέλο παραγωγής γράφων με βάση τις κοινότητες $P(\mathbf{G}|\mathbf{F})$, τότε η επαγωγή του πίνακα $\mathbf{F}$ ενός γράφου $\mathbf{G}$ ισοδυναμεί με την ανακάλυψη των κοινοτήτων.

Μοντέλο συμμετοχής κόμβου σε κοινότητες Πολλές μέθοδοι είναι βασισμένες στην υπόθεση ότι οι κοινότητες είναι πυκνότερα συνδεδεμένες στο εσωτερικό τους και ότι οι επικαλύψεις των κοινοτήτων είναι αραιά συνδεδεμένες. Μελέτες σε δίκτυα με γνωστή την κατανομή των κοινοτήτων, [62] έδειξαν ότι αυτή η υπόθεση δεν είναι σωστή. Στα περισσότερα πραγματικά δίκτυα, όταν ο αριθμός των κοινοτήτων όπου συμμετέχουν δυο κόμβοι αυξάνεται, τότε αυξάνεται η πιθανότητα οι δυο αυτοί κόμβοι να είναι συνδεδεμένοι. Αυτό σημαίνει ότι δημιουργούνται επικαλύψεις κοινοτήτων, που είναι πυκνά συνδεδεμένες στο εσωτερικό τους. Οι περισσότεροι αλγόριθμοι που βασίζονται στην υπόθεση ότι οι επικαλύψεις είναι πιο αραιά συνδεδεμένες από ότι οι κοινότητες στο εσωτερικό τους, αποτυγχάνουν να εντοπίσουν πυκνά συνδεδεμένες επικαλύψεις. Αντίθετα αναγνωρίζουν τέτοιου είδους επικαλύψεις, είτε σαν ξεχωριστές κοινότητες, είτε ως μια ενιαία.

Το μοντέλο συμμετοχής κόμβων σε κοινότητες (AGM) [62], παράγει, από έναν πίνακα συμμετοχής κόμβων σε κοινότητες, γράφους, όπου οι κοινότητες μπορούν να έχουν πυκνά ή αραιά συνδεδεμένες επικαλύψεις. Οι κόμβοι που συμμετέχουν στην ίδια κοινότητα C έχουν πιθανότητα να είναι συνδεδεμένοι μεταξύ τους, ίση με p_C λόγω της συμμετοχής τους σε αυτήν τη κοινότητα. Η συνολική πιθανότητα για δυο κόμβους να είναι συνδεδεμένοι, με δεδομένες τις παραμέτρους (V, C, M, p_C) είναι :

$$p(u,v) = 1 - \prod_{C \in M_u \bigcap M_v} (1 - p_C) \tag{1}$$

Το μοντέλο συμμετοχής κόμβου σε κοινότητες μπορεί γίνεται πιο ευέλικτο, αναθέτοντας διαφορετικούς βαθμούς συμμετοχής στους κόμβους της ίδιας κοινότητα [54, 63, 72]. Μπορεί να οριστεί F_{uA} η συμμετοχή του κόμβου u στην κοινότητα A. Το μοντέλο ορίζει την συμμετοχή σε κοινότητες με τη μορφή ενός διμερούς γράφου, όπου κάθε κόμβος συνδέεται σε μια ή περισσότερες κοινότητες με μια ακμή διαφορετικού βαθμού συμμετοχής. Η συνολική πιθανότητα να είναι συνδεδεμένοι 2 κόμβοι είναι:

$$\begin{aligned} P(u,v) &= 1 - \prod_{C \in \Gamma} (1 - P_C(u,v)) \\ &= 1 - \prod_{C \in \Gamma} (-F_{uc} \cdot F_{vC}) \\ &= 1 - \exp(-\sum_{C \in \Gamma} F_{uc} \cdot F_{vC}) \\ &= 1 - \exp(-\underbrace{F_u^T \cdot F_v}_{\text{εσωτερικό γινόμενο}}) \end{aligned} \tag{2}$$

Όπου F_u το διάνυσμα συμμετοχής κόμβου του κόμβου u, $\{F_{uC}\}_{C \in \Gamma}$ και F_v το διάνυσμα συμμετοχής του κόμβου v, $\{F_{vC}\}_{C \in \Gamma}$ και Γ το σύνολο των κοινοτήτων.

Ο πίνακας γειτνίασης μπορεί να δημιουργηθεί από τον πίνακα συμμετοχής κόμβων - κοινοτήτων $\mathbf{F} \in \mathbb{R}_{\geq 0}^{N \times C}$ ακολουθώντας μια ανεξάρτητη πιθανότητα κατανομής.

$$A_{uv} \sim Bernoulli(1 - \exp(-F_{uc} \cdot F_{vc})) \tag{3}$$

Μέθοδος BigClam Στην μέθοδο BigClam [63], προτείνει την χρήση του μοντέλου AGM αντίστροφα. Υπολογίζοντας την αποδοτικότερη τιμή του πίνακα συμμετοχής $\mathbf{F}$, που μεγιστοποιεί της πιθανότητα ο γράφος να προέκυψε από το μοντέλο, με χρήση του αλγόριθμου σύγκλισης με ελάττωση της παραγώγου (gradient descent).Η πιθανότητα ο γράφος να έχει προκύψει απο το μοντέλο με βάση

τον πίνακα συμμετοχή F υπολογίζεται ως:

$$\begin{aligned} P(G|F) &= \prod_{(u,v)\in G} P(u,v) \prod_{(u,v)\notin G} (1 - P(u,v)) \\ &= \prod_{u.v\in E} (1 - \exp(-F_u^T F_v)) \prod_{u.v\notin E} \exp(-F_u^T F_v) \end{aligned} \tag{4}$$

Η λογαριθμική πιθανότητα είναι:

$$\begin{aligned} log(P(G|F)) &= log(\prod_{u,v\in E} (1 - \exp(-F_u^T F_v)) \prod_{u,v\notin E} \exp(-F_u^T F_v)) \\ &= \sum_{u,v\in E} \log(1 - \exp(-F_u^T F_v)) - \sum_{u,v\notin E} (F_u^T F_v) \\ &= l(f) \end{aligned} \tag{5}$$

Μέθοδος Neural Overlapping Community Detection, NOCD Η βασική ιδέα της μεθόδου είναι η βελτιστοποίηση του πίνακα συμμετοχής $\mathbf{F}$ που μεγιστοποιεί τη συνάρτηση μέγιστης λογαριθμικής πιθανοτητας του μοντέλου Big-Clam , όχι ως ελεύθερης μεταβλητής, αλλά με την εκπαίδευση ενός συνελικτικού δικτύου γράφων (GCN) και της βελτιστοποίησης των παραμέτρων του.

$$\mathbf{F} := GCN_\theta(\mathbf{A}, \mathbf{X}) \tag{6}$$

Η αρνητική λογαριθμική πιθανότητα προς ελαχιστοποίηση είναι :

$$-log(P(G|F)) = \mathcal{L} = -\sum_{u,v\in E} \log(1 - \exp(-F_u^T F_v)) + \sum_{u,v\notin E} (F_u^T F_v) \tag{7}$$

Αντί να υπολογίζεται η αποδοτικότερη τιμή του πίνακα συμμετοχής $\mathbf{F}$ ως ελεύθερη μεταβλητή, υπολογίζονται οι αποδοτικότερες τιμές των παραμέτρων $\theta^\star$ που ελαχιστοποιούν την συνάρτηση απωλειών Loss function.

$$\arg\min_\theta \mathcal{L}(GCN_\theta(\mathbf{A}, \mathbf{X})) \tag{8}$$

όπου:

$$\mathbf{F} := GCN_\theta(\mathbf{A}, \mathbf{X}) = \sigma(\underbrace{\hat{\mathbf{A}}\sigma(\overbrace{\hat{\mathbf{A}}XW^{(1)}}^{\text{the first layer}})W^{(2)}}_{\text{the second layer}}) \tag{9}$$

σ είναι μια μη γραμμική συνάρτηση ενεργοποίησης, συγκεκριμένα προτάθηκε η

χρήση της $LeakyRelu$.

Η χρήση ενός νευρωνικού δικτύου γράφων έχει αρκετά πλεονεκτήματα. Μέσω μιας επαγωγικής προκατάληψης, το δίκτυο εξάγει παρόμοια διανύσματα συσχέτισης κοινότητας για γειτονικούς κόμβους, το οποίο βελτιώνει την ποιότητα των προβλέψεων σε σύγκριση με απλούστερα μοντέλα. Επίσης, μας επιτρέπει να ενσωματώσουμε τα χαρακτηριστικά των κόμβων στο μοντέλο. Εάν τα χαρακτηριστικά δεν είναι διαθέσιμα, μπορούν να χρησιμοποιηθεί ο πίνακας A ως χαρακτηριστικά κόμβου. Ακόμη είναι δυνατή η πρόβλεψη κοινοτήτων επαγωγικά για κόμβους που δεν υπήρχαν στον γράφο, κατά τον χρόνο της εκπαίδευσης.

Επέκταση της μεθόδου NOCD με έναν μηχανισμό προσοχής Πρόταση της εργασίας είναι η επέκταση της μεθόδου, με έναν μηχανισμό "προσοχής". Σκοπός είναι μέσα απο τη διαδικασία εκπαίδευσης του μοντέλου, να εξαχθεί ένας διαφορετικός συντελεστής προσοχής ενός κόμβου προς κάθε του γειτονικό κόμβο. Διακρίνοντας ποιος κόμβος είναι πιο σημαντικός σε μια γειτονιά, και αποδίδοντας σε αυτόν μεγαλύτερο βάρος.

Στην μέθοδο NOCD ο συντελεστής σημασίας είναι ο ίδιος για όλους του γείτονες ενός κόμβου:

$$h_v^{(l)} = \sigma\big(\sum_{u \in N(v)} \underbrace{\frac{1}{|N(v)|}}_{\text{σταθερή προσοχή}} \mathbf{W^{(l)}} h_u^{(l-1)}\big) \tag{10}$$

Οι υπολογισμός του διανύσματος προσοχής a_{vu} γίνεται αρχικά μέσω του υπολογισμού των συντελεστών προσοχής e. Στη συνέχεια, εφαρμόζεται στους συντελεστές προσοχής η μη γραμμική συνάρτηση

$$e_{vu}^{(l)} = LeakyRelu(a^{(l-1)^T} Concat(\mathbf{W^{(l)}} h_v^{(l-1)}, \mathbf{W^{(l)}} h_u^{(l-1)})) \tag{11}$$

$$a_{vu}^{(l)} = \frac{\exp(e_{vu}^{(l)})}{\sum_{k \in N_v} \exp e_{vk}^{(l)}} \tag{12}$$

$$h_v^{(l)} = \sigma\big(\sum_{u \in N(v)} \underbrace{a_{vu}^{l}}_{\text{συντελεστής προσοχής}} \mathbf{W^{(l)}} h_u^{(l-1)}\big) \tag{13}$$

Στην πράξη ο μηχανισμός αυτός δεν συγκλίνει εύκολα. Για αυτό εκτελούνται πάνω από μία επαναλήψεις, με διαφορετικές αρχικοποιήσεις. Οι επαναλήψεις αυτές ονομάζονται heads . Και το τελικό αποτέλεσμα είναι ένας συνδυασμός των αποτελεσμάτων των διαφορετικών επαναλήψεων, όπως ο μέσος όρος, το άθροισμα ή η αλληλουχία.

Πειραματική Διαδικασία Για την αξιολόγηση της μεθόδου, χρησιμοποιήθηκαν δέκα σύνολα δεδομένων. Έξι σύνολα δεδομένων από το Facebook [38] με δίκτυα 50 ως 800 κόμβων. Τέσσερα σύνολα δεδομένων, τα Chemistry, Computer Science, Medicine, Engineering που είναι δίκτυα συν-συγγραφής επιστημονικών δημοσιεύσεων, από το Microsoft Academic Graph [24] με περισσότερους από 10 χιλιάδες κόμβους. Σε αυτά οι κοινότητες αντιστοιχούν σε επιστημονικούς τομείς και τα διανύσματα χαρακτηριστικών των κόμβων είναι βασισμένα στις λέξεις κλειδιά των δημοσιεύσεων κάθε συγγραφέα. Οι λεπτομέρειες των συνόλων δεδομένων περιγράφονται στον πίνακα 5.1.

Σε όλα τα πειράματα το συνελικτικό δίκτυο γράφου GCN αποτελείται απο 2 επίπεδα με ενδιάμεσο κρυμμένο μέγεθος 128.

Το δίκτυο με τον μηχανισμό προσοχής, GAT αποτελείται επίσης από 2 επίπεδα με ενδιάμεσο κρυμμένο μέγεθος 128 και 2 επαναλήψεις του μηχανισμού προσοχής heads ., από τις οποίες υπολογίζουμε τον μέσο όρο των αποτελεσμάτων τους.

Η είσοδος του δικτύου έχει το μέγεθος του διανύσματος των χαρακτηριστικών των κόμβων, ή το μέγεθος του διανύσματος γειτνίασης του κάθε κόμβου, ενώ η τελική έξοδος έχει μέγεθος ίσο με τον αριθμό των κοινοτήτων. Και στα δύο μοντέλα GCN, GAT μετά το πρώτο επίπεδο εφαρμόζεται κανονικοποίηση batch normalization.

Σε κάθε επίπεδο εφαρμόζεται η τεχνική Dropout με πιθανότητα διατήρησης 50%. Ακόμη εφαρμόζεται weight decay στους πίνακες βαρών με regularization strength $\lambda = 10^{-2}$. Ο πίνακας χαρακτηριστικών $\mathbf{X}$ (ή $\mathbf{A}$, σε περίπτωση που εργαζόμαστε χωρίς χαρακτηριστικά) είναι κανονικοποιημένος ώστε κάθε γραμμή να έχει μοναδιαία L_2 νόρμα.

Η βελτιστοποίηση επαναλαμβάνεται το πολύ για 5000 εποχές ή σταματά αν δεν υπάρχει βελτίωση στην συνάρτηση κόστους loss funtion για 500 συνεχόμενες επαναλήψεις. Ο ρυθμός εκμάθησης είναι 10^{-3}.

Όταν η βελτιστοποίηση τερματιστεί κάθε κόμβος ανατίθεται στις κοινότητες στις οποία η τιμή συμμετοχής του στο διάνυσμα εξόδου του δικτύου, είναι πάνω απο ένα όριο $\rho = 0.5$.

Η μετρική αξιολόγησης της ικανότητας εντοπισμού των πραγματικών κοινοτήτων που χρησιμοποιήθηκε είναι η Κανονικοποιημένη Αμοιβαία Πληροφορία Normalized Mutual Information (NMI).Η Αμοιβαία Πληροφορία εκφράζει την κοινή πληροφορία μεταξύ δυο κατανομών, ως το ποσό της αβεβαίτητας που μειώνεται για την μία όταν γνωρίζουμε την άλλη. Έχουν προταθεί πολλές εκδοχές ορισμού της κανονικοποιημένης μορφής της. Εδώ χρησιμοποιήθηκε η εκδοχή όπως ορίζεται από τους *McDaid et al.*, [39].

$$NMI_{max}(X,Y) = \frac{1}{2}\frac{H(X) + H(Y) - H(X|Y) - H(Y|X)}{max(H(X), H(Y))}$$

Τα πειράματα εκτελέστηκαν σε μια K12 GPU με 12GB RAM. Η υλοποίηση έγινε στη γλώσσα προγραμματισμού Python με χρήση των βιβλιοθηκών PyTorch και PyTorch Geometric και βασίστηκε στην υλοποίηση της δημοσίευσης [50].

Αποτελέσματα και συμπεράσματα Στον πίνακα 1 παρουσιάζονται οι τιμές της μετρικής NMI που πέτυχαν τα δυο μοντέλα. Συγκεκριμένα εμφανίζεται ο μέσος όρος των τιμών NMI 10 επαναλήψεων της εκπαίδευσης του μοντέλου με διαφορετικές αρχικοποιήσεις μαζί με την τυπικής απόκλιση των τιμών αυτών.

Τα δυο μοντέλα έχουν καλύτερη απόδοση, στα μεγαλύτερα σύνολα δεδομένων, όταν τα διανύσματα των χαρακτηριστικών των κόμβων χρησιμοποιούνται σαν είσοδος στο μοντέλο. Αυτά τα σύνολα έχουν διανύσματα χαρακτηριστικών κόμβων πολύ μεγάλης διάστασης.

Στα μεγαλύτερα σύνολα δεδομένων παρατηρείται μια αύξηση στην ικανότητα ανακάλυψης των κοινοτήτων μετά την προσθήκη του μηχανισμού προσοχής. Κάτι που σημαίνει πως σε αυτά ο μηχανισμός προσοχής "γρεεκ"μαθαίνει ποια είναι τα σημαντικά στοιχεία της εισόδου, αποδίδοντας σε αυτά μεγαλύτερη προσοχή.

Όσον αφορά στην ικανότητα του μοντέλου να αποδίδει σε μεγάλα δίκτυα δεδομένων, όπως φαίνεται στον πίνακα 2 ο χρόνος που χρειάζεται είναι λιγότερος απο δυο λεπτά ώστε το μοντέλο να αποφασίσει, για το σύνολο Medicine (810 χιλιάδες κόμβοι), με την χρήση μίας GPU με 12 GB RAM.

Ενδιαφέρουσες περιοχές έρευνας, είναι η ποσοτική συσχέτιση των συντελεστών προσοχής με την ύπαρξη κοινοτήτων, θα μπορούσε ίσως να οδηγήσει σε μια ανάλυση ερμηνευσιμότητας του μοντέλου. Πεδίο για διερεύνηση μπορεί να είναι η πειραματική αξιολόγηση νεότερων προσεγγίσεων μηχανισμών προσοχής [6, 69].

Βασικό ανοιχτό ερώτημα προς διερεύνηση, είναι το ότι για πολλές μεθόδους αναζήτησης κοινοτήτων, δίνεται ως δεδομένο ο αριθμός των πραγματικών κοινοτήτων. Κάτι τέτοιο δεν είναι εφικτό για τα περισσότερα πραγματικά δίκτυα δεδομένων και αποτελεί σημαντικό πεδίο έρευνας η ανάπτυξη μεθόδων που θα μπορούν να το υπολογίσουν.

	Adjacency		Attributes	
Dataset	GCN	GAT	GCN	GAT
Facebook 348	30.9 ± 1.4	**33.4 ± 3.5**	29.4 ± 2.6	30.3 ± 2.9
Facebook 414	**52.1 ± 3.5**	50.6 ± 1.9	51.0 ± 4.5	51.3 ± 4.6
Facebook 686	**17.7 ± 1.3**	17.1 ± 0.8	16.0 ± 1.9	15.3 ± 2.1
Facebook 698	41.2 ± 3.7	**45.1 ± 3.2**	36.9 ± 6.4	31.1 ± 9.1
Facebook 1684	**39.6 ± 1.5**	33.7 ± 3.0	26.3 ± 1.5	30.2 ± 2.7
Facebook 1912	**43.2 ± 0.8**	39.5 ± 1.7	33.0 ± 3.6	34.7 ± 2.9
Chemistry	19.0 ± 2.6	19.2 ± 1.1	41.3 ± 3.0	**41.8 ± 2.3**
Computer Science	29.9 ± 1.8	29.6 ± 1.4	**46.5 ± 3.4**	46.1 ± 1.6
Engineering	19.4 ± 1.0	15.2 ± 1.4	34.8 ± 3.4	**35.6 ± 1.3**
Medicine	27.4 ± 0.9	23.6 ± 2.7	34.9 ± 3.4	**35.2 ± 2.7**

Table 1: Ανακάλυψη των πραγματικών επικαλυπτόμενων κοινοτήτων, σε NMI (%) και τυπική απόκλιση. Μέσος όρος 10 επαναλήψεων με διαφορετική αρχικοποίηση.

	Adjacency		Attributes	
Dataset	GCN	GAT	GCN	GAT
Facebook 348	6.729	9.395	6.463	9.091
Facebook 414	6.096	8.733	6.132	8.958
Facebook 686	6.531	9.032	6.509	9.072
Facebook 698	7.002	9.166	6.745	9.134
Facebook 1684	11.633	9.745	7.043	9.668
Facebook 1912	6.895	14.426	11.186	15.385
Chemistry	26.078	43.094	29.209	71.867
Computer Science	20.067	31.075	28.482	79.655
Engineering	13.031	19.183	15.766	34.84
Medicine	78.674	169.594	74.745	114.041

Table 2: Πραγματικός χρόνος εκτέλεσης της εκπαίδευσης, σε δευτερόλεπτα. Μέσος όρος 10 επαναλήψεων με διαφορετική αρχικοποίηση.

Abstract

Community detection is a research area with increasing practical significance. Successful examples of its application are found in social networks, recommender systems and biology.

Deep learning has achieved great performance on various graph related tasks and is recently used in the field of community detection with accuracy and scalability.

The majority of community detection efforts, using deep learning, is investigating the discovery of disjoint communities. While the overlapping communities are a lot less researched.

In this thesis we propose an extension to an already well established method to discover overlapping communities. This method combines a probabilistic view on the problem with the expressiveness of representation learning,using a Graph Neural Network.

Our proposed method lies on the addition of an attention mechanism, focusing on the "important" parts of the data. We investigate whether an attention mechanism can discover differences on the inputs's importance, allowing the model to focus more on the important nodes of a neighborhood.

Finally we evaluate our proposed method, examining its ability to discover ground truth communities. For larger networks and inputs, the graph attention mechanism of the proposed method increases the community detection ability.

Contents

List of Figures

List of Tables

List of Algorithms

Thesis structure

Introduction The motivation and purpose of the thesis are outlined, followed by a description of related studies.

Community Detection The problem definition and an overview of the most widely used methods.

Graph Neural Networks A description of Graph Neural Networks design and architecture.

Model Related models description, upon which our proposal is based. Our proposal description.

Experiments The setup of the experiments is described with the model parameters, the description of the datasets in use.

Results and discussion The results of the conducted experiments are analyzed and discussed.

Conclusion A presentation of the conclusions drawn from the experiments is given, and topics for future work are presented.

Chapter 1

Introduction

1.1 Background

Communication systems, social networks, financial transactions, biological functions and other systems can be represented in the form of a graph. A graph describes a group of nodes that interact with each other forming edges.

A community is considered a group of nodes interact with each other more often than others, or share a mutual attribute [16]. Community structures have been investigated as early as the 1920s in sociology. Advanced scientific tools developed, during the last century, allowed the study of networks on real data. In 2002, Girvan and Newman [20] opened a new direction with graph partition. The past ten years researchers have extensively studied community detection [53].

Community detection is the process of discovering the communities that exist in a graph. This allows us to study interconnectedness, evolution or even importance of a graph's structure. Its role is pivotal in the of Protein-Protein Interaction (PPI) networks, revealing the complexities of and proteins with similar biological functions. Also it has been used in the study of metabolism and brain networks.

1.2 Motivation

During the recent years a significant development of deep learning techniques has emerged with advantages in the handling of high dimensional network data.

The biggest proportion of community detection research is directed towards assigning each node to exactly one community, handling only non-overlapping communities. A much smaller number of research efforts studies the formulation and discovery of overlapping communities that motivated this thesis towards the

direction of overlapping community detection. Additionally, we wanted to study how deep representation learning techniques could be used towards the discovery of overlapping communities.

1.3 Related work

Usually the methods proposed to detect overlapping community structure, are based on the assumption, that community overlaps are less densely connected than the core of the communities. As shown in [62] the higher the number of communities a pair of nodes have in common, the higher the probability is, that they will be connected. Meaning that the community overlaps formed are often densely connected. Most methods based on the hypothesis stated above, will fail to detect dense overlaps, and usually detect them as separate communities, or as one community. In [62, 63] a probabilistic view of community structure is proposed that includes the formation of densely connected overlaps.

Deep learning has proven its ability to handle high dimensional data. A short description of community detection methods, based on deep learning, is presented in Chapter 2. Many methods use deep learning to produce low dimensional representations of graph nodes, followed by a clustering technique such as K-means, but usually they are not very efficient on Community structure detection.

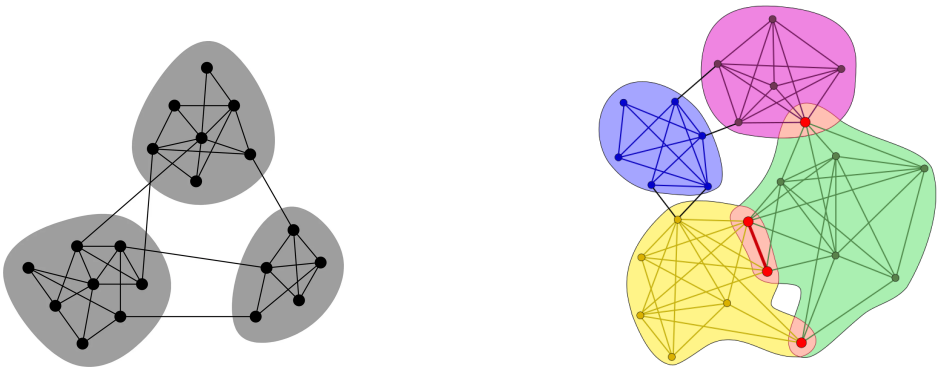
1.4 Contribution

Our proposed method is based upon [50], where a combination of the probabilistic model, with a representation learning approach, using graph neural networks is proposed. This thesis proposed extension is the addition of an attention mechanism that infers importance of information. Finally we present our experiments, the model's performance, evaluation and comparisons.

Chapter 2

Community Detection

2.1 Overlapping community detection



(a) Non-overlapping communities (b) Overlapping communities

Figure 2.1: Non-overlapping vs overlapping communities, Wikipedia

A universally agreed upon definition of "community" does not exist in the literature, although its meaning seems rather intuitive. In many recent papers a usual statement is that a community consists of a group of nodes, that present a higher probability to be linked with each other than with other nodes. [17].

Assuming an undirected unweighted graph $\mathcal{G}(\mathbf{V}, \mathbf{E})$ that can be represented with a binary adjacency matrix $\mathbf{A} \in (0,1)^{N \times N}$. Where N is the number of nodes $V = \{1, \ldots, N\}$ and M is the number of edges $E = \{(u, v) \in V \times V : A_{uv} = 1\}$.

Every node can be associated with a D-dimensional attribute vector, represented by an attribute matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$.

The goal of overlapping community detection is to assign nodes into C communities. Such assignment could be represented as a non-negative node-

community affiliation matrix $\mathbf{F} \in \mathbb{R}_{\geq 0}^{N \times C}$, where F_{uc} is the strength of node u's membership in community c.

The problem of community detection can be considered in terms of the probabilistic inference framework [63]. If a community-based generative model $P(\mathbf{G}|\mathbf{F})$ is posited, then to infer the unobserved affiliation matrix $\mathbf{F}$ of a graph $\mathbf{G}$ would mean to detect the communities.

The problem of community detection can be examined from a representation learning point of view. The affiliation matrix $\mathbf{F}$ could be considered as an embedding of nodes into $\mathbb{R}_{\geq 0}^{C}$, while preserving the graph structure.

Combining existing node embedding approaches with overlapping K-means has not delivered satisfactory results. In [50], Shchur and Günnemann proposed the combination of the probabilistic and the representation points of view, to learn the community affiliations, using a graph convolutional network. This is the model we propose to extend with the use of a graph attention network and evaluate it's performance.

2.2 Methods overview

Traditional Methods

Traditional methods usually explore community formation from network structure [53].

Graph Partition The methods are usually known as graph clustering, are also, employed in deep learning models. They partition the network into a given number of K communities. Kernigal Lin and spectral bisection are two well known representatives of graph clustering [16, 31, 4].

Statistical Inference Based on the Stochastic Block Model (SBM), a generative model that assigns nodes to communities, optimizing their probabilities of likelihood [25, 30].

Hierarchical Clustering Discovering hierarchies these methods are further divided to agglomerative, divisive and hybrid. The Girvan-Newman algorithm is a divisive algorithm, removing edges to discover structures[20]. OST is a divisive algorithm based on edge betweenness via minimum spanning trees [44]. Fast Modularity (FastQ)[41] is an agglomerative representative and Community Detection Algorithm based on Structural Similarity [68].

Dynamical Methods Random walks are utilized to detect communities in

[47]. Information Mapping (InfoMap) calculates the minimal-length encoding[48].

Spectral Clustering Spectral clustering [3] partitions the network on the normalized Laplacian matrix and the regularized adjacency matrix, and fits the SBM in the pseudo-likelihood algorithm.

Density-based Algorithms DBscan[14] and others detect community modules, by measuring entities density.

Optimizations Community detection methods often maximize certain functions. Modularity[41] is the most widely used optimization function, and its variant Fastq[11]. The Louvain method [5] is a well known optimization method. Also extremal, or spectral optimization and simulated annealing are extensions of greedy methods. Normalized Mutual Information (NMI), and Conductance (CON) are also used to optimize the partition quality.

Deep learning methods

Deep learning approaches have been very successful recently handling data with underlying grid-like structure. Real world data, though, often come with a non grid-like structure such as graphs. Deep learning methods have been recently proven very efficient handling networks and graphs, on different tasks and also on community detection tasks [53].

Convolutional Networks Convolutional Neural Networks (CNN) and Graph Convolutional Networks (GCN) apply convolutions to represent latent features.

Graph Attention Networks (GAT) Apply a trainable attention weight to the graph information.

Generative Adversarial Networks (GAN) Adversarial training and GANs are used with highly successful results in community detection.

Auto-Encoders (AE) A wide variety of Auto Encoders has been used. Denoising, Stacked, Sparse, GCN, GAT or adversarial autoencoders.

Deep Nonnegative Matrix Factorization (DNMF) A particular technique factorizing an adjacency matrix.

Deep sparse filtering These methods consist of a two layer learning model, capable of handling high dimensional data.

Some methods that focus on overlapping community detection while using Deep Learning reviewed on [53] are:

LGNN Line Graph Neural Network [10] learns node representation features in directed networks, extending the Stochastic Block Model.

NOCD New overlapping community Detection [50] combines the GCN representation learning with probabilistic inference.

SEAL Seed Expansion with generative Adversarial Learning [71] expands Generative Adversarial Learning, its generated samples are communities and a Graph Isomorphism Network (GIN) is used as a discriminator.

CommunityGan Community GAN [28] performs a competition between the motif-level AGM generator and discriminator, to optimize community memberships embeddings.

ACNE [9] expands Generative Adversarial Learning, to optimize node and community embeddings.

For completeness, a list of methods, presented in [53] that utilize a Graph Attention mechanism. All 5 methods use k-means for clustering and aim to detect non-overlapping communities.

- DMGI [46]
- HDMI [29]
- MAGNN [18]
- HeCo [58]
- CP-GNN [37]

Chapter 3

Graph Neural Networks

3.1 Introduction to Graph Neural Networks

Many extensions to existing Deep Learning methods have been proposed recently, towards handling non-grid data like graphs. Some are extending ideas coming from signal processing [51], while others are presenting novel concepts. Deep learning models on non-grid structured data encode inductive bias, through the notions of geometry. Michael Bronstein first grouped these methods proposing the term of geometric deep learning.

Graph Convolution Networks (GCN) have been proved very efficient on graph related tasks, like node classification, link prediction, community detection and others.

In [7], [13] and [59], a convolution operation on graphs is defined via the graph spectral domain.

Representation Learning [8] has proven to be very efficient on all types of graph tasks.

Assuming a graph with:

- $\mathcal{G}(\mathbf{V}, \mathbf{E})$.
- $\mathbf{V}$ a set of nodes.
- $\mathbf{E}$ the set of edges.
- $\mathbf{X}$ a matrix representing node features.

A naïve approach to train a deep neural network on a graph would have various problems. Using the adjacency matrix as an input to a deep neural network would create a number of parameters to be trained multiple times the number of nodes, which would be much larger than the input samples. Another

issue would be that the trained model would not be applicable to graphs of different sizes. Also this model would be sensitive to node ordering.

Borrowing intuition from Convolutional Neural Networks (CNN), Graph Neural Networks generalize the idea of convolutions between layers, leveraging node features or attributes. The idea is that instead of a sliding window on where an operator is applied, there will be a center of collection information, a node, where an operator is applied transforming information gathered from the neighbors and creating new information.

Message passing and neighbor aggregation in graph convolution networks has to be permutation equivariant.

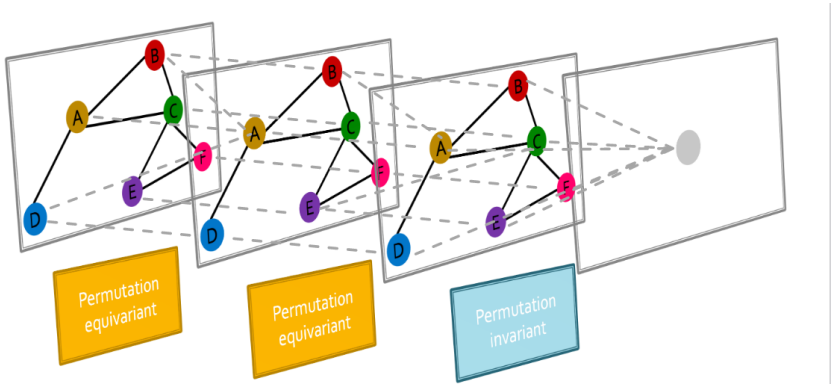


Figure 3.1: Graph Neural Networks (GNN) consist of multiple permutation equivariant functions, Bronstein, ICLR 2021 keynote

The basic idea behind a GNN is that the node's neighborhood, defines the structure of the neural network. Every node will get it's own computation graph, based on the graph structure around him.

In every layer of this computation network, two basic operations are performed. Message collection including some message transformation and the aggregation of all the node's neighbors messages. This process will be repeated just for a few steps.

The neighborhood aggregation function performed at each layer needs to be order invariant. After collection of information from the neighbors, an order invariant operator is applied Usually the calculation of a summation, or of the average, or those of maximum or minimum. An embedding calculated at layer $l + 1$ is :

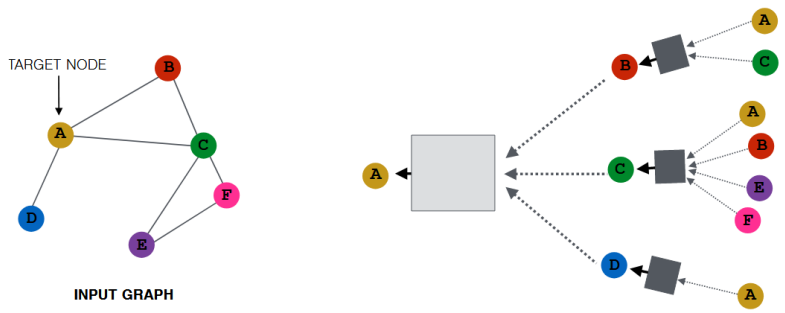


Figure 3.2: GNN computational graph, Stanford CS224W

$$h_v^0 = \mathbf{x}_v \tag{3.1}$$

$$h_v^{(l+1)} = \sigma(\mathbf{W}_{(l)} \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|} + \mathbf{B}_l h_v^{(l)}), \forall l \in 0, .., L-1 \tag{3.2}$$

$$z_v = h_v^{(l)} \tag{3.3}$$

where $\mathbf{W}$, $\mathbf{B}$ are the trained parameters shared across all nodes and σ is a non-linearity function like *Relu* or *leakyRelu*. The model can be trained using any loss function, applying Stochastic Gradient Descent.

In the matrix form this can be expressed as :

$$\sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|} \rightarrow \mathbf{H}^{(l+1)} = \mathbf{D}^{-1}\mathbf{A}\mathbf{H}^{(l)} \tag{3.4}$$

where

$$\mathbf{H}^l = [h_1^{(l)} ... h_{|V|}^{(l)}]^T \tag{3.5}$$

$$\mathbf{D}_{v,v} = \mathbf{Deg}(v) = |N(v)| \tag{3.6}$$

$$\mathbf{D}_{v,v}^{-1} = \frac{1}{|N(v)|} \tag{3.7}$$

and the whole aggregation function as :

$$\mathbf{H}^{(l+1)} = \sigma(\mathbf{D}^{-1}\mathbf{A}\mathbf{H}^{(l)}\mathbf{W}_l^T + \mathbf{H}^{(l)}\mathbf{B}_l^T) \tag{3.8}$$

which can be calculated with the use of efficient sparse matrix multiplication.

3.2 Graph Neural Networks Architecture

A variety of Graph neural networks has achieved state of the art performance on various tasks ,node classification [59], graph classification [65] and link prediction [70].

Different transformation or aggregation operators allow a wide variety of GNN architecture[67] like GCN, GraphSage, GAT and others.

The way the layers are stacked defines the GNN architecture in a different way. Layer connectivity, is about whether the layers are stacked, and if there are skip connections.

Different architecture can be defined by the way the computation graph is created. The graph can be augmented to create new features, or some other graph structure manipulation is applied.

Finally the objective task defines the architecture. Whether it is supervised or not, or about the node or the edge or graph level.

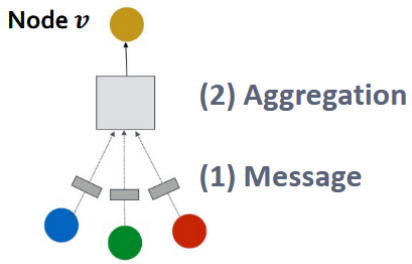


Figure 3.3: GNN two step process, Stanford CS224W

3.2.1 GNN Layer

A simple GNN layer can be defined as a two step process of compressing a set of vectors into a single vector:

- Message Computation.
- Aggregation

Message computation transforms the representation of a node at a previous layer and transforms it.

$$m_u^{(l)} = MSG^{(l)}(h_u^{(l-1)}) \tag{3.9}$$

an example would be a linear transformation $m_u^{(l)} = \mathbf{W}^{(l)} h_u^{(l-1)}$

Message aggregation, has to bee an ordering invariant function. Like a summation, an average and even maximum.

$$h_v^{(l)} = AGG^{(l)}(\{m_u^{(l)}, u\ inN(v)\}) \tag{3.10}$$

An example would be a summation for aggregation:

$$h_v^{(l)} = Sum^{(l)}(\{m_u^{(l)}, u\ inN(v)\}) \tag{3.11}$$

In order to preserve the information of a node computed at a previous layers, when calculating its embedding at a next level, an aggregation of that previous embedding is usually applied.

$$h_v^{(l)} = CONCAT(AGG^{(l)}(\{m_u^{(l)}, u\ inN(v)\}), m_v^{(l)}) \tag{3.12}$$

where $m_v^{(l)} = \mathbf{W}^{(l)} h_v^{(l-1)}$.

After the aggregation, a non linear activation function is applied σ: *Relu* , *Sigmoid*.

$$h_v^{(l)} = \sigma\Big(\mathbf{W^{(l)}} \sum_{u \in N(v)} \frac{h_u^{(l-1)}}{|N(v)|}\Big) \tag{3.13}$$

The Graph Convolutional Network [59] (GCN) can been as a combination of the two step process:

$$h_v^{(l)} = \sigma\Big(\underbrace{\sum_{u \in N(v)}}_{Aggregation} \overbrace{\mathbf{W^{(l)}} \frac{h_u^{(l-1)}}{|N(v)|}}^{Message}\Big) \tag{3.14}$$

In GraphSAGE [22] the double aggregation shown in Referenceseq:doubleagg is applied. Different aggregation operators can be applied, like a weighted average of the neighbors like GCN, a symmetric function like a maximum pooling function, or a summation or even an LSTM sequence model to the messages, shuffling the ordering of the neighbor messages. Also for the message computation step, different operators can be used like an MLP. Usually l_2 normalization is applied on every layer.

In practice many modern deep learning modules are used in a GNN layer.

- Batch Normalization, to stabilize neural network training [27].
- Dropout, to prevent overfitting [52].

- Attention Gating to control the importance of each message [56].
- Activation functions, like Rectified Linear Unit (ReLU), Sigmoid and parametric ReLU.

3.2.2 Stacking layers

In a K-layer GNN each node has a receptive field of a K-hop neighborhood. The receptive field is the set of nodes that will determine the embedding of the node of interest. Increasing the number of stacked layer K of a GNN will mean that many nodes will have highly overlapped receptive fields. In this way similar embeddings will be computed for several nodes. This problem is widely known as the over-smoothing problem. To overcome the over-smoothing problem, the necessary receptive field needs to be analyzed, and the number of layers needs to be set accordingly.

Enhancing the expressive power of a GNN, while avoiding over-smoothing can be achieved by applying different stacking techniques.

- Increasing the expressive power within each layer. A 3-layer Multilayer Perceptron (MLP) can be used within each layer.
- Using Pre-processing or post-processing layers that do not pass messages.
- Skipping connections between layers [23, 61] can create a mixture of shallow and deep GNNs.

3.2.3 Graph Augmentation

In many cases the input graph does not carry enough information for a GNN to be trained efficiently.

- The graph could lack features.
- The graph could be too sparse, leading to inefficient message passing.
- The graph could be too dense, that means message passing is too costly.
- The graph could be too large, leading to memory limitations.

A solution to this issue is the idea that the computation graph does not need to be the same as the input graph.In general two types of augmentation can be performed.

- Graph feature augmentation.
- Graph structure augmentation.

Graph feature augmentation

When the input graph does not have any features, the standard approach is to assign unique ids to graphs. The ids can be converted to one-hot vectors.

	Constant	One-hot
Expressive power	Medium	High
Inductive learning	High	Low
Computational cost	Low	High, $\mathcal{O}(\|V\|)$
Use cases	Any graph, inductive	Small graph, transductive

Table 3.1: Constant versus one-hot encoding feature augmentation

Some graph structures are hard to learn with a GNN. One example is about nodes that belong to cycles of different lengths, where they all share the same degree of 2. Their computational graph will be the same binary tree. A solution would be to use the cycle count as a node feature[66].

Other types of node features that can be used this way are:

- Node degree
- Clustering coefficient
- PageRank
- Centrality

Graph structure augmentation

Augmenting sparse graphs can be achieved by:

Adding virtual edges A common approach is to connect two-hop neighbors via a virtual edge. This means that instead of the adjacency matric A, the $A + A^2$ is used as an input to the GNN.

Adding virtual nodes A virtual node added, is connected to all the nodes. All the nodes will have a distance of two , improving message passing is sparse graphs.

For dense graphs, an idea proposed [22] is that instead of using all the nodes for message passing, a randomly sampled group of a node's neighborhood [64] is used at each layer. This can greatly reduce the computational cost.

3.3 Training a Graph Neural Network

So far we have seen that given an input graph's structure, the corresponding computation graph and GNN can output a set of node embeddings $h_u^{(L)}, \forall u \in G$. A GNN training pipeline, is additionally defined by:

- Prediction Heads
- Labels
- Evaluation Metrics
- Loss functions
- Dataset split

3.3.1 Prediction Heads

Different tasks require different prediction heads.

- Node level
- Edge level
- Graph level

Node level prediction

Node level predictions are made directly using the node embeddings. After the GNN training d-dimension node embeddings have been calculated $h_u^{(L)} \in \mathbb{R}, \forall u \in G$. A k-way prediction can be either:

- Classification, where we classify among k categories.
- Regression, where we regress on k targets.

The predicted values are :

$$\begin{aligned} \hat{y}_u &= Head_{node}(h_u^{(L)}) \\ &= W^{(H)} h_u^{(L)}, W^{(H)} \in \mathbb{R}^{k \times d} \end{aligned}$$

Mapping node embeddings from $h_u^{(L)} \in \mathbb{R}^d$ to $\hat{y}_u \in \mathbb{R}^k$.

Edge level prediction

Making predictions using pairs of node embedding. Supposing we would like to make k-way edge predictions $\hat{y}_{uv} = Head_{edge}(h_u^{(L)}, h_v^{(L)})$ The options are:

Concatenation and Linear $\hat{y}_{uv} = Linear(Concat((h_u^{(L)}, h_v^{(L)})))$, the linear transformation will map the 2d-dimensional embeddings to k-dimensions embeddings to make k-way predictions.

Dot product $\hat{y}_{uv} = (h_u^{(L)})^T \cdot h_v^{(L)}$ for a 1-way prediction, like link prediction.

K-way prediction Dot product Requires the introduction of a multi-head attention, where $W^{(1)}, .., W^{(k)}$ trainable parameters are introduced and k differently initialized predictions are performed, keeping their average or concatenation as a result. $\hat{y}_{uv} = Concat(\hat{y}_{uv}^{(k)})$, where $\hat{y}_{uv}^{(k)} = (h_u^{(L)})^T \cdot W^{(k)} h_v^{(L)}$.

Graph level prediction

To make Graph level predictions, all the node embeddings in a graph under are aggregated. Supposing a k-way prediction, the predicted head is

$$\hat{y}_G = Head_{graph}(\{h_u^{(L)} \in \mathbb{R}^d, \forall u \in G\}) \tag{3.15}$$

Some options for $Head_{graph}(\{h_u^{(L)} \in \mathbb{R}^d, \forall u \in G\})$ are the global Max, Mean, and Sum pooling operations [60]. Hierarchical Global Pooling has been proposed [65] to improve the efficacy of global pooling operators.

3.3.2 Labels

Training can be performed both in supervised and unsupervised settings. In supervised learning the labels come from external sources. While in unsupervised the labels come from graph themselves. (semi-supervised). The unsupervised signals can be different node statistics like the clustering coefficient or PageRank, for edge level predictions the link prediction and for graph level graph isomorphism can be used.

3.3.3 Loss Functions

Supposing a training setting with N data points, predictions are defined as:

Node-level prediction $\hat{y}_u^{(i)}$ with label $y_u^{(i)}$

Edge-level prediction $\hat{y}_{uv}^{(i)}$ with label $y_{uv}^{(i)}$

Graph-Level prediction $\hat{y}_{G}^{(i)}$ with label $y_{G}^{(i)}$

In general prediction $\hat{y}^{(i)}$ and label $y^{(i)}$

Classification

Cross-Entropy (CE) loss is used for k-way classification.

$$CE(y^{(i)}, \hat{y}^{(i)}) = -\sum_{j=1}^{k} y_j^{(i)} log(\hat{y}_j^{(i)}) \tag{3.16}$$

where

$$y^{(i)} \in \mathbb{R}^k \text{ is one-hot encoding}$$
$$\hat{y}^{(i)} \in \mathbb{R}^k \text{ the prediction after Softmax}$$

and the total loss over all N training examples is:

$$Loss = \sum_{i=1}^{N} CE(y^{(i)}, \hat{y}^{(i)}) \tag{3.17}$$

Regression

For regression tasks the Mean Squared Error (MSE) also known as L2 loss is often used. For a k-way regression:

$$MSE(y^{(i)}, \hat{y}^{(i)}) = \sum_{j=1}^{k} (y_j^{(i)} - \hat{y}_j^{(i)})^2 \tag{3.18}$$

where

$$y^{(i)} \in \mathbb{R}^k \text{ is a Real valued vector of targets}$$
$$\hat{y}^{(i)} \in \mathbb{R}^k \text{ is a Real valued vector of predictions}$$

and the total loss over all N training examples is:

$$Loss = \sum_{i=1}^{N} MSE(y^{(i)}, \hat{y}^{(i)}) \tag{3.19}$$

3.3.4 Evaluation Metrics

The standard evaluation metrics are used for GNN training.

Regression

Root Mean Square Error (RMSE)

$$\sqrt{\sum_{i=1}^{N} \frac{(y^{(i)} - \hat{y}^{(i)})^2}{N}} \tag{3.20}$$

Mean Absolute Error (MAE)

$$\frac{\sum_{i=1}^{N} |y^{(i)} - \hat{y}^{(i)}|}{N} \tag{3.21}$$

Multi-class Classification

Accuracy

$$\frac{1[argmax(\hat{y}^{(i)} = y^{(i)})]}{N} \tag{3.22}$$

Binary Classification

Accuracy

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{3.23}$$

Precision P

$$\frac{TP}{TP + FP} \tag{3.24}$$

Recall R

$$\frac{TP}{TP + FN} \tag{3.25}$$

F1-Score

$$\frac{2P \times R}{P + R} \tag{3.26}$$

	Actually Positive	Actually Negative
Predicted Positive	True Positives (TPs)	False Positives (FPs)
Predicted Negative	False Negatives (FNs)	True Negatives (TNs)

Table 3.2: Confusion Matrix

Receiver operating characteristic (ROC) Curve The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR).

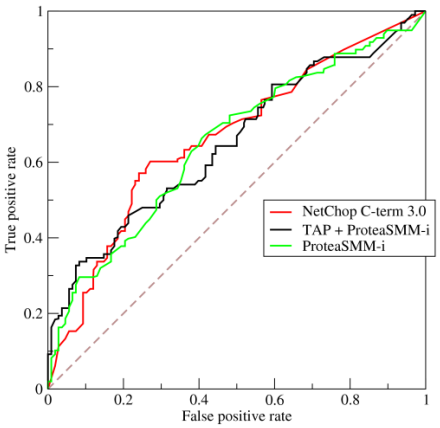


Figure 3.4: ROC Curve, Wikipedia

Where the dashed line represents the performance of a random classifier.

$$TPR = R = \frac{TP}{TP + FN}$$
$$FPR = \frac{TN}{TN + FP}$$

We are measuring the Area Under the ROC Curve (AUC), which intuitively captures the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.

3.3.5 Dataset split

In machine learning tasks usually the graph is split just once to:

Training set used for optimizing the GNN parameters

Validation set to develop the model hyper-parameters

Test set to report the final performance

But for graph datasets we cannot guarantee that the test set will be really held out. So applying many random splits to training, validation and test splits, and average their result is proposed.

Splitting a graph is a special case, because graph data points like nodes are not independent. Removing a node affects the computational graphs of all neighboring nodes and their embeddings computation. Several solutions are proposed to address this special dataset split.

Transductive Setting

The input graph can be observed in all the dataset splits (training,validation and test set). Only the node labels are split, meaning that different node labels are used for training, validation and testing. Transductive setting is only applicable for node and edge level prediction tasks.

Inductive Setting

The input graph is split to three different graphs used for training, validation and testing. The dataset then consists of multiple graph splits. Each split can only observe the graph within the split. A successful model can generalize to unseen graphs. The inductive setting is applicable to all level tasks including node, edge and graph level tasks.

3.4 Graph Attention Networks

Attention based models are inspired by human perception. Proposing the selective focus on important parts of the information. Attention based models have been applied to several deep learning tasks with much success. Graph Attention (GAT) mechanisms have been recently extended to generalize on graph structured data and already had much success on several graph related tasks.

In a Graph Attention Network (GAT) [56][55], when aggregating information from the neighbors, a weight factor is assigned to each message.

$$h_v^{(l)} = \sigma\Big(\sum_{u \in N(v)} a_{vu} \mathbf{W}^{(\mathbf{l})} h_u^{(l-1)}\Big) \tag{3.27}$$

In GCN and in the GraphSAGE the importance weight factor was defined explicitly as $\frac{1}{|N(u)|}$.

In a GAT network, the graph attention mechanism "learns" the different weight factors as the model is trained. The idea is that the neural network should devote more computing power on that small but important part of the data. Implicitly specifying different weights to different nodes in a neighborhood.

Lets assume that the attention mechanism produces a weight factor a_{vu} which denotes the importance of node u to node v, by first computing attention coef-

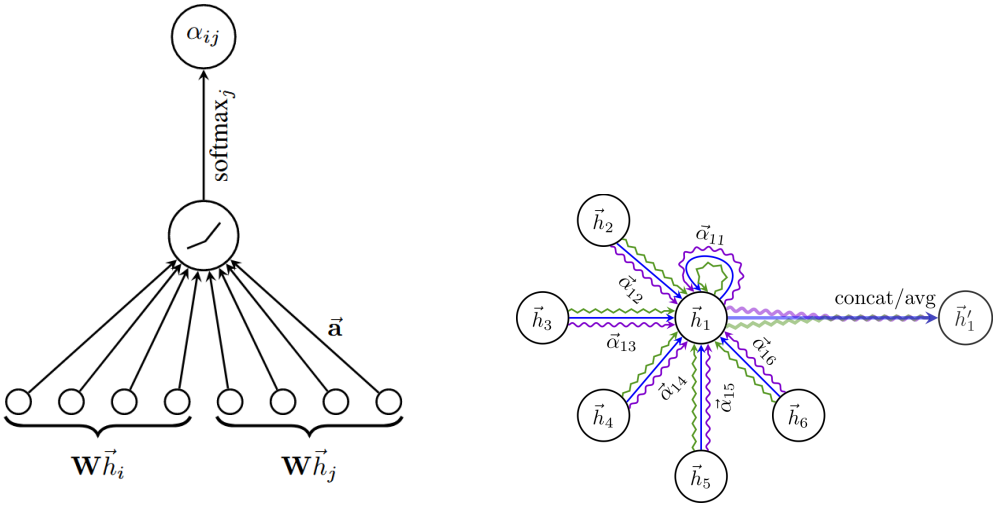


Figure 3.5: At left the attention mechanism 3.28 illustrated. At right an illustration of multihead attention (with K = 3 heads) by node one's neighborhood, Veličković et al. [56]

ficients across pairs of nodes based on their messages:

$$e_{vu} = a(\mathbf{W}^{(l)}h_v^{(l-1)}, \mathbf{W}^{(l)}h_u^{(l-1)}) \tag{3.28}$$

The form of the attention mechanism a a can be a one layer linear network with trainable parameters.

$$\begin{aligned} e_{vu} &= a(\mathbf{W}^{(l)}h_v^{(l-1)}, \mathbf{W}^{(l)}h_u^{(l-1)}) \\ e_{vu} &= Linear(Concat(\mathbf{W}^{(l)}h_v^{(l-1)}, \mathbf{W}^{(l)}h_u^{(l-1)})) \end{aligned} \tag{3.29}$$

Normalizing the coefficients using a SoftMax so that $\sum_{u/inN(v)} a_{vu} = 1$ calculates the final attention weight:

$$a_{vu} = \frac{\exp\left(e_{vu}\right)}{\sum_{k\in N(v)} \exp(e_{vk})} \tag{3.30}$$

The weighted sum and the embedding at layer l is calculated as :

$$h_v^{(l)} = \sigma(\sum_{u\in N(v)} a_{vu}\mathbf{W}^{(\mathbf{l})}h_u^{(l-1)}) \tag{3.31}$$

An example is shown in fig 3.6 for node A the embedding will be computed as :

(a) Attention coefficients (b) Attention a

Figure 3.6: Attention Mechanism Example, Stanford CS224W

$$e_{AB} = a(\mathbf{W}^{(l)} h_A^{(l-1)}, \mathbf{W}^{(l)} h_B^{(l-1)}) \tag{3.32}$$

$$a_{AB} = \frac{\exp(e_{AB})}{\exp(a_{AB}) + \exp(a_{AC}) + \exp(a_{AD})} \tag{3.33}$$

$$\mathbf{h_A}^{(l)} = \sigma(a_{AB}\mathbf{W}^{(l)} h_B^{(l-1)} + a_{AC}\mathbf{W}^{(l)} h_C^{(l-1)} + a_{AD}\mathbf{W}^{(l)} h_D^{(l-1)}) \tag{3.34}$$

The learnable parameters of the attention mechanism as shown in figure 3.29 will be learned along with the weight parameters of the graph neural network $\mathbf{W}^{(l)}$

To stabilize the learning process of self-attention, a multi-head attention is used. .K independent attention mechanisms, known as "heads" compute embeddings with different random initializations for their parameters. Then, the calculated embeddings are concatenated or averaged to produce the final output:

$$h_v^{(l)} = ||_{k=1}^{K} \Big(\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^{(k)} \mathbf{W}^{(k),(l-1)} h_u^{(l-1)} \Big) \tag{3.35}$$

is an example of concatenation.

$\alpha_{vu}^{(k)}$ are the normalized attention coefficients computed by the k-th attention mechanism (a^k), and $\mathbf{W}^{(k)}$ is the corresponding input linear transformation's weight matrix. Note that for this setting, $\mathbf{h}' \in \mathbb{R}^{KF'}$.

Using an attention mechanisms provides a series of benefits: Allowing implicitly the specification of different importance values to different nodes, while being computationally efficient, attentional coefficients computation and aggregation can be parallelized.

Fixed number of parameters and sparse matrix operations are memory efficient. The mechanism attends over local neighborhood structure.It is a pair

edge wise mechanism that does not depend on the global graph structure, thus it can provide inductive capabilities.

Chapter 4

Model

4.1 Affiliation Graph Model

One important aspect, of how networks are formed, is the triadic closure, meaning that similar nodes tend to create embedded structures, like clusters. In the World Wide Web, pages related to a common subject are more densely linked with each other [15]. In biological networks proteins that share a common functional pattern interact with each other more often [19],[33].

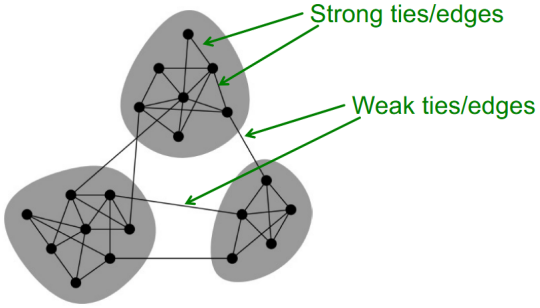


Figure 4.1: Granoveter, the strength of weak ties, Wikipedia

Mark Granoveter [21] described edges in a graph, from a structural point of view. Structurally embedded edges are considered strong edges. While long range edges connecting different parts of the network are considered weak. Through an information point of view, structurally embedded edges are heavily redundant in terms of information flow, while long range edges facilitate the flow of information from different parts of the network. Many publications define communities as groups of nodes, that have more internal than external edges [16]. Granoveter's theory was much later tested by [43] and validated on a European cell phone network graph.

Many methods to detect network modules were developed based on this idea of weak and strong connections among nodes, meaning that graphs consist of dense;y connected clusters that are linked by a smaller number of weak edges. Graph partitioning [49], Modularity [42] or Betweenness centrality [20] based methods are usually based in this assumption.

The assumption that community overlaps are less densely connected than the surrounding communities(the non-overlapping parts), figure 4.2 (b) assumes that the community overlaps are less densely connected, than the communities. This would mean that for a pair of nodes with increasing number of common communities, the possibility of them being connected is decreasing. One of the reasons this remained unnoticed for a long time was the limited access in the past to datasets with ground-truth structure.

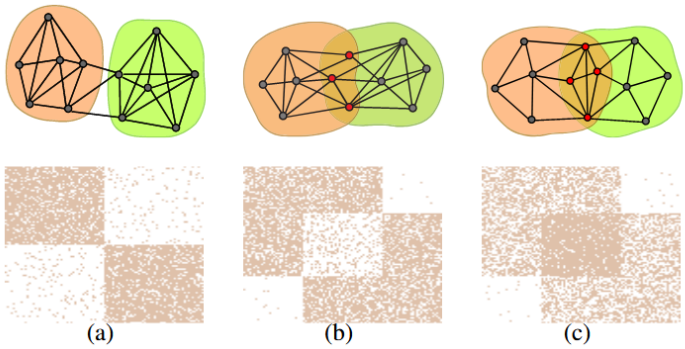


Figure 4.2: View of two non overlapping (a) communities,and two overlapping communities, sparse & dense, Yang et al. [62]

Yang, Jaewon and Leskovec [62] with the use of ground-truth community datasets, showed that on the contrary, when the number of common communities two nodes share, the higher the probability will be that they are connected 4.2 (c). This observation explains why several methods fail to detect dense overlaps [1], [45] [2]. Methods based on this assumption would mistakenly identify dense overlaps as a separate community or join together two overlapping communities to one. The Affiliation Graph Model (AGM [62] can produce graphs, where the overlapping communities, can have densely connected overlaps.

Considering a graph G with:

- Nodes V, edges E, communities C, memberships M
- Each community has a single probability p_c

According to the AGM model given the parameters (V, C, M, p_c), a pair of nodes that both belong to a community C will connect to each other by probability p_c.

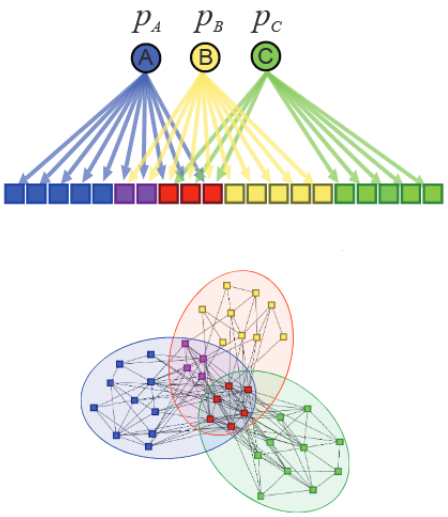


Figure 4.3: Affiliation Graph Model, Yang et al. [62]

Two nodes u, v that belong to multiple common communities $M_u \bigcap M_v$, where M_u and M_v the communities each node belongs to, will have a higher probability to form an edge between them.

$$p(u,v) = 1 - \prod_{c \in M_u \bigcap M_v} (1 - p_c) \tag{4.1}$$

The AGM model describes a variety of community structures, that can be overlapping or not and even nested.

The AGM model , can be relaxed with the use of a node community membership strength function. [54, 63, 72]. Where F_{uC} defines the membership strength of node u to community C. The model, defines the community memberships, as a bipartite graph (e.g Figure 4.4) where each node u is connected to a community A by an edge of a membership strength F_{uA}.

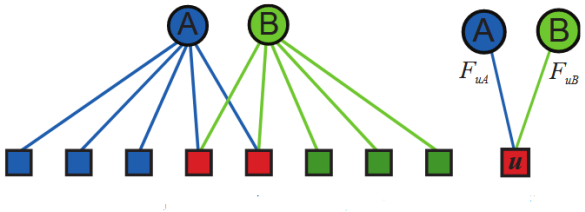


Figure 4.4: Membership bipartite graph, Yang, Jaewon, Leskovec [63]

The probability for a pair of nodes to be connected, under a community C, depends on the value of their non-negative membership strengths.

$$P_c(u, v) = 1 - \exp(-F_{uc} \cdot F_{vc}) \tag{4.2}$$

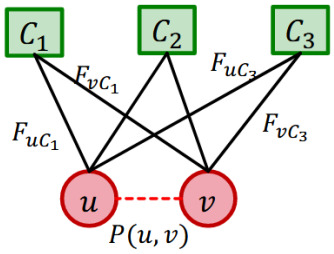


Figure 4.5: Affiliation membership bipartite graph, Stanford CS224W

The total probability, that the two nodes are connected is.

$$P(u, v) = 1 - \prod_{C \in \Gamma} (1 - P_C(u, v)) \tag{4.3}$$

where $C \in \Gamma$, and Γ a set of all the communities.

The overall probability then that the two nodes are connected is calculated in r4.4.

$$\begin{aligned} P(u, v) &= 1 - \prod_{C \in \Gamma} (1 - P_c(u, v)) \\ &= 1 - \prod_{C \in \Gamma} (-F_{uc} \cdot F_{vc}) \\ &= 1 - \exp(-\sum_{Cin\Gamma} F_{uc} \cdot F_{vc}) \\ &= 1 - \exp(-\underbrace{F_u^T F_v}_{\text{dot product}}) \end{aligned} \tag{4.4}$$

where F_u: a vector of $\{F_{uC}\}_{C \in \Gamma}$ and F_v: a vector of $\{F_{vC}\}_{C \in \Gamma}$, with the node's membership strength values to all communities.

Given the affiliations $\mathbf{F} \in \mathbb{R}_{\geq 0}^{N \times C}$ the adjacency matrix is produced, with each A_{uv} entry being sampled with mutual Independence and following the same probability distribution.

$$A_{uv} \sim Bernoulli(1 - \exp(-F_{uc} \cdot F_{vc})) \tag{4.5}$$

4.2 Graph likelihood

The AGM model can be used in a reverse way. To infer the parameters of the AGM model, while maximizing the probability that the graph was generated by the model, would mean to infer the community structure [62].

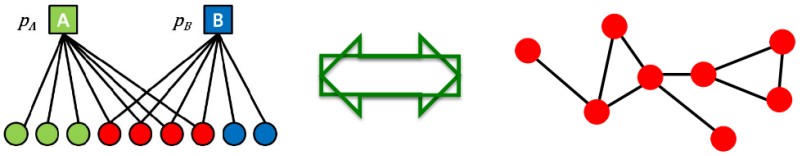


Figure 4.6: Graph fitting, Stanford CS224W

The likelihood to maximize is $P(G|F)$, optimizing the variable F, using gradient descent.

$$\arg\max_F P(G|F) \tag{4.6}$$

Given G and F, the likelihood that F generated G is calculated as :

$$P(G|F) = \prod_{(u,v)\in G} P(u,v) \prod_{(u,v)\notin G} (1 - P(u,v)) \tag{4.7}$$

where $P(u,v)$ if the probability the nodes u, v are connected.

4.3 BigClam, Cluster Affiliation Model for Big Networks

The graph likelihood [63] will be :

$$\begin{aligned} P(G|F) &= \prod_{(u,v)\in G} P(u,v) \prod_{(u,v)\notin G} (1 - P(u,v)) \\ &= \prod_{u.v\in E} (1 - \exp(-F_u^T F_v)) \prod_{u.v\notin E} \exp(-F_u^T F_v) \end{aligned} \tag{4.8}$$

The Maximum log likelihood objective is :

$$\begin{aligned} log(P(G|F)) &= log(\prod_{u,v\in E}(1-\exp(-F_u^T F_v))\prod_{u,v\notin E}\exp(-F_u^T F_v)) \\ &= \sum_{u,v\in E}\log(1-\exp(-F_u^T F_v)) - \sum_{u,v\notin E}(F_u^T F_v) \\ &= l(f) \end{aligned} \tag{4.9}$$

The optimization problem to solve:

$$\arg\max_F \log(P(G|F)) \tag{4.10}$$

The gradient descent approach used is shown in 1.

Algorithm 1 BigCLAM

$F \leftarrow$ random initialization
repeat
  for $u \in V$ **do**
    Update the membership F_u for node u,
    while fixing the memberships of all other nodes.
    Do gradient descent to increase log-likelihood.
  end for
until Convergence

More specifically a block coordinate gradient descent algorithm is used [26, 35], where the memberships of one node are updated, while fixing the membership of all others. Fixing all F_v, the problem of updating F_u, becomes a convex optimization problem. To solve the following problem for all nodes u:

$$\arg\max_{F_u\geq 0} l(F_u) \tag{4.11}$$

where

$$l(F_u) = log(P(G|F_u)) = \sum_{u,v\in E}\log(1-\exp(-F_u^T F_v)) - \sum_{u,v\notin E}(F_u^T F_v) \tag{4.12}$$

The gradient of the loss function will be:

$$\nabla l(F_u) = \sum_{v\in N_{(u)}}\big(\frac{\exp(-F_u^T F_v}{1-\exp(-F_u^T F_v)}\big)\cdot F_v - \sum_{v\notin N_{(u)}} F_v \tag{4.13}$$

When the number of nodes reaches millions, this process is not very scalable. Computing $l(F_u)$ and $\nabla l(F_u)$ takes linear time $\mathcal{O}(N)$. The second term in equation 4.13, is very expensive to compute, because it sums over all non - neighbors of node u.

$$\sum_{v \notin N_{(u)}} F_v \tag{4.14}$$

By calculating $\sum_v F_v$ just once, and caching it, then updating the two other terms in 4.15, the time to compute each step is reduced to $\mathcal{O}(|N_{(u)}|)$. This caching trick means that updating F_u takes near-constant time.

$$\sum_{v \notin N_{(u)}} F_v = \sum_v F_v - F_u - \sum_{v \in N_{(u)}} F_v \tag{4.15}$$

4.4 Neural Overlapping Community Detection

Neural Overlapping Community Detection (NOCD) [50] is also based on the AGM model, combining the probabilistic view with representation learning.

$$\begin{aligned} log(P(G|F)) &= log(\prod_{u,v \in E} (1 - \exp(-F_u^T F_v)) \prod_{u,v \notin E} \exp(-F_u^T F_v)) \\ &= \sum_{u,v \in E} \log(1 - \exp(-F_u^T F_v)) - \sum_{u,v \notin E} (F_u^T F_v) \\ &= l(f) \end{aligned} \tag{4.16}$$

Key idea of the proposed method is the maximization of the graph log likelihood, calculated in equation 4.16 as it was calculated in section 4.3, by calculating the node community membership matrix $\mathbf{F}$, with a Graph Neural Network (GNN), using the adjacency matrix as an input or the node feature matrix $\mathbf{X}$ if one is available.

The membership matrix F is produced through a GCN.

$$\mathbf{F} := GNN_\theta(\mathbf{A}, \mathbf{X}) \tag{4.17}$$

The negative log-likelihood, aiming to minimize shown in 4.18

$$-log(P(G|F)) = -\sum_{u,v \in E} \log(1 - \exp(-F_u^T F_v)) + \sum_{u,v \notin E} (F_u^T F_v) \tag{4.18}$$

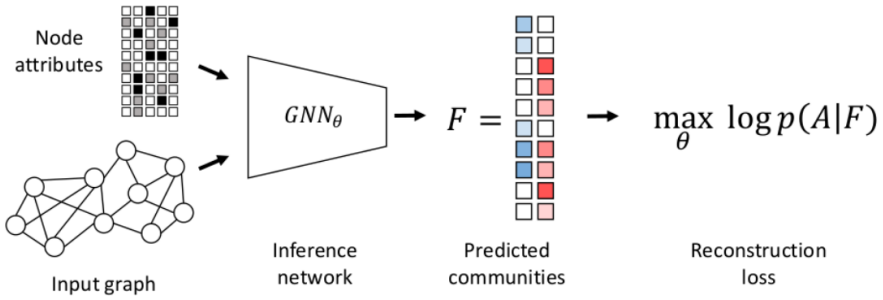


Figure 4.7: Neural Overlapping Community Detection, Shchur [50]

Most real-world networks tend to be extremely sparse $\frac{|E|}{n^2} \ll 1$, meaning that the second term in 4.18 will contribute the largest part to the loss function. To Balance the two terms, uniform distributions of existing or non-existing edges are considered, then the loss is calculated in 4.19.

$$-\mathcal{L}(\mathbf{F}) = -\mathbb{E}_{u,v\sim P_E}\left[\log(1-\exp(-F_u^T F_v))\right] + \mathbb{E}_{u,v\sim P_E}\left[(F_u^T F_v)\right] \tag{4.19}$$

where P_E and P_N are uniform distributions over edges and non-edges respectively.

Instead of optimizing the $\mathbf{F}$ affiliation array directly as was done in BigClam, in NOCD the parameters $\theta^\star$ that minimize the loss function, are optimized.

$$\arg\min_\theta \mathcal{L}(GNN_\theta(\mathbf{A}, \mathbf{X})) \tag{4.20}$$

Using a GNN to calculate the affiliation matrix(F) offers plenty of advantages.

A GNN will output similar community affiliation vectors to neighboring nodes, due to an inductive bias, improving the quality of the predictions.

Additionally, the node features matrix $\mathbf{X}$ can be used as an input into the model. If the node features matrix $\mathbf{X}$ is not available, the adjacency matrix $\mathbf{A}$ can be used as input [59].

Also , the GNN can predict communities inductively for nodes not known at training time.

$$\mathbf{F} := GNN_\theta(\mathbf{A}, \mathbf{X}) = \sigma\underbrace{(\hat{\mathbf{A}}\overbrace{\sigma(\hat{\mathbf{A}}XW^{(1)})}^{\text{the first layer}}W^{(2)})}_{\text{the second layer}} \tag{4.21}$$

In more detail NOCD uses a two-layer GCN.

$$\mathbf{F} := GCN_\theta(\mathbf{A}, \mathbf{X}) = ReLu(\hat{\mathbf{A}} ReLu(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(\mathbf{1})})\mathbf{W}^{(\mathbf{2})}) \tag{4.22}$$

where $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\mathbf{1}}\tilde{\mathbf{A}}$, is the normalized adjacency matrix, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I_N}$ is the adjacency matrix with self loops, and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the diagonal degree matrix.

If the feature matrix $X \in \mathbb{R}^{N \times D}$, is used as an input, then for a hidden layer with H feature maps, the parameter matrices to optimize will be $W^{(1)} \in \mathbb{R}^{D \times H}$ for the first layer, and $W^{(2)} \in \mathbb{R}^{H \times F}$ for the final one.

Using sparse-dense matrix multiplications, the computational complexity of evaluating 4.22 is $\mathcal{O}(MDHF)$, that is linear in the number of graph edges M.

Batch normalization is used after the first convolutional layer, and $\mathcal{L}_2$ regularization is applied to all weight matrices. Detailed information about the parameters of the GCN are described in Section 5.3.

The caching trick used in BigClam can also be applied here, reducing the complexity of operations from $\mathcal{O}(N^2)$ to $\mathcal{O}(N + M)$. Considering that $M \ll N$ this leads to great performance optimization.

Instead of computing the loss function using all entries of the adjacency matrix $\mathbf{A}$ the model is further optimized by calculating the loss function just for a sample mini batch S of edges and non-edges at each training epoch. Experiments have proved that training using the stochastic loss, converges to the same result. Noting here that the full adjacency matrix is used inside the GCN.

After optimizing the loss function, a threshold parameter $\rho = 0.5$ is defined. Every node is assigned to a community C if $\mathbf{F_{uC}} > \rho$.

4.5 Overlapping Community Detection using Graph Attention Networks

Here we propose the addition of an attention mechanism to the graph neural network, of NOCD [50], in order to focus on the most important information gathered from the nodes in the neighborhood.

Attention is inspired by cognitive attention, so that different importance values can be assigned to different parts of the data. Which part of the data is important is "learned" through training, while the neural network devotes more computing power on that small but important part, specifying arbitrary importance to different neighbors of each node in the graph.

The key idea behind Graph Attention Networks is to compute each node's hidden representations, by attending over its neighbors, following a self-attention strategy. While GCN equally treats the neighbors of the target node 4.23,

GAT networks[56] utilize a self attention mechanism to allow the assignment of different weights to nodes in the same neighborhood 4.24.

An embedding calculate for not v at layer l, explicitly assigning equal importance to all nodes.

$$h_v^{(l)} = \sigma\Big(\sum_{u \in N(v)} \underbrace{\frac{1}{|N(v)|}}_{\text{fixed importance}} \mathbf{W^{(l)}} \mathbf{h_u}^{(l-1)} \Big) \tag{4.23}$$

In a GAT layer nodes attend over their neighborhood's message, implicitly specifying different weights to different nodes in the neighborhood.

$$h_v^{(l)} = \sigma\Big(\sum_{u \in N(v)} \underbrace{\alpha_{vu}}_{\text{attention weights}} \mathbf{W^{(l)}} \mathbf{h_u}^{(l-1)} \Big) \tag{4.24}$$

where α_{vu} is the attention weights "learned" by the attention mechanism:

As described in section 3.4 the weight coefficients, are calculated:

$$e_{vu}^{(l)} = LeakyReLu(a^{(l-1)^T}(\mathbf{W^{(l)}} \mathbf{h_v}^{(l-1)} || \mathbf{W^{(l)}} \mathbf{h_u}^{(l-1)}) \tag{4.25}$$

The attention coefficients will be:

$$a_{vu}^{(l)} = \frac{\exp(e_{vu}^{(l)})}{\sum_{k \in N_v} \exp(e_{vk}^{(l)})} \tag{4.26}$$

and the embeddings at layer l are computed:

$$h_v^{(l)} = \sigma\Big(\sum_{u \in N(v)} a_{vu}^{l} \mathbf{W^{(l)}} \mathbf{h_u}^{(l-1)} \Big) \tag{4.27}$$

Adding that the time complexity of a GAT attention head computing an embedding of size F features, as proposed by [56], is expressed as $\mathcal{O}(NDF + MF)$, where D is the number of input features and N, M the number of nodes and edges. The complexity is about equal with the complexity of the GCN network[59].

The configuration of the network is defined by two graph attention layers. Also batch normalization is used after the first convolutional layer, and $\mathcal{L}_2$ regularization is applied to all weight matrices. Detailed information about the rest of the hyper-parameters of the GAT network are described in the experiment description at Section 5.4.

The model is further optimized by calculating the Loss function just for a sample mini batch S of edges and non-edges at each training epoch.

After the loss function is optimized a threshold parameter $\rho = 0.5$ is defined. Every node is assigned to a community C if $\mathbf{F_{uC}} > \rho$.

Chapter 5

Experiments description

In order to evaluate the proposed model's performance, an experiment was conducted. A set of different datasets with ground truth community structure, was used to train the model one at a time. The model training was repeated 10 times for each dataset and their results averaged along with the standard deviation observed.

5.1 Datasets

A collection of real-world graph datasets was used in our experiments. **Facebook**[38] is a collection of small ego-networks from the Facebook graph with a number of nodes ranging from 50 to 800 .

Four larger real world datasets were used, with reliable ground truth overlapping community information and node attributes. **Chemistry**, **Computer Science**, **Medicine**, **Engineering** are co-authorship networks, constructed from the Microsoft Academic Graph [24]. Communities correspond to research areas, and node attributes are based on keywords of the papers by each author. Statistics for all the datasets processed through the experiments, are presented in Table 5.1.

5.2 Implementation

The code implementation is based on the NOCD code [50]. The GAT network is implemented using Pytorch and the PyTorch Geometric libraries.

Dataset	Network type	N	M	D	C
Facebook 348	Social	224	3.2K	21	14
Facebook 414	Social	150	1.7K	16	7
Facebook 686	Social	168	1.6K	9	14
Facebook 698	Social	61	270	6	13
Facebook 1684	Social	786	14.0K	15	17
Facebook 1912	Social	747	30.0K	29	46
Computer Science	Co-authorship	22.0K	96.8K	7.8K	18
Chemistry	Co-authorship	35.4K	157.4K	4.9K	14
Medicine	Co-authorship	63.3K	810.3K	5.5K	17
Engineering	Co-authorship	14.9K	49.3K	4.8K	16

Table 5.1: Dataset statistics. K stands for 1000

5.3 GCN configuration

A two layer Graph Convolutional Network (GCN)[59] was used for all experiments. Specifically:

$$\mathbf{F} := GNN_\theta(\mathbf{A}, \mathbf{X}) = ReLu(\hat{\mathbf{A}} ReLu(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(1)})\mathbf{W}^{(2)}) \tag{5.1}$$

where $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$, is the normalized adjacency matrix, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I_N}$ is the adjacency matrix with self loops, and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the diagonal degree matrix.

The hidden layer size is set at 128 and the final layer output is set at size equal with the number of ground truth communities C.

Batch normalization is applied after the first graph convolution layer and L_2 regularization is applied to all the weight matrices. The architecture hyper parameters were optimized using only the Computer Science dataset.

5.4 GAT configuration

A two layer Graph Attention Network[56] was used for all the experiments.

The first layer consists of $K = 2$ heads computing embeddings of size $\mathcal{F}' = 128$. The average of the two heads results is calculated and finally passed through a ReLU non-linearity [40].

The second layer consists of $K = 2$ heads of size $\mathcal{F}'' = C$. The average of the two heads results is calculated and finally passed through another ReLU non-linearity.

Experiments took place trying different values for the hidden layer size of 16, 32, 64. Also tried the head concatenation instead of averaging them. But

the choices of hidden layer size equal to 128 and averaging of the heads lead to best results.

5.5 Hyper parameters

The two models share the rest of hyper parameters:

- Batch normalization is applied after the first graph layer.
- Dropout with 50% keep probability is applied before every layer.
- Weight decay is applied to both weight matrices with regularization strength of $\lambda = 10^{-2}$.
- The feature matrix $\mathbf{X}$ is L_2 normalized before input.

5.6 Training

The training was performed using the Adam optimizer [32], with the default parameters. Every 50 epochs the full training loss is computed, and if no improvement happened for the last 500 iterations, or after 5000 epochs, the training stops.

5.7 Assigning nodes to communities

In order to compare the detected community partitionings, to the ground truth community structure, the predicted continuous community affiliations F are transformed into binary community assignments. Each node u is assigned to a community c if its affiliation strength F_{uc} is above a fixed threshold ρ. The threshold $\rho = 0.5$ like all other hyperparameters, was set following the hyperparameters set in NOCD [50].

5.8 Normalized Mutual Information

Normalized Mutual Information (NMI), is a measure used in community detection, to evaluate the partitioning performance. It is often considered because of its comprehensive meaning and its ability to compare two partitions of different size.

Considering the community assignments $\{x_i\}$,$\{y_i\}$ where x_i,y_i denote the cluster labels of vertex i in partitions $\mathcal{X}$ and $\mathcal{Y}$ one assumes that that the labels

x and y are values of two random variables X and Y, with joint distribution, $P(x, y)$, where

$$P(x) = P(X = x) = \frac{n_x^X}{n}$$
$$P(y) = P(Y = y) = \frac{n_y^Y}{n}$$
$$P(x, y) = P(X = x, Y = y) = \frac{n_{xy}}{n}$$

where $\frac{n_x^X}{n}$, $\frac{n_y^Y}{n}$, $\frac{n_{xy}}{n}$ are the sizes of the clusters labeled by x, y and their overlap respectively.

NMI derives from entropy in information theory. Where for a discrete random variable X, its Shannon entropy is defined as :

$$H(X) = -\sum_x P(x) \log P(x)$$
$$H(Y) = -\sum_y P(y) \log P(y)$$

Similarly their joint entropy is defined as :

$$H(X, Y) = -\sum_{x,y} P(x, y) \log(P(x, y)) \tag{5.2}$$

and the conditional entropy ox X given Y is :

$$H(X|Y) = -\sum_{x,y} P(x, y) \log P(x|y) \tag{5.3}$$

Then the mutual information is calculated as:

$$I(X : Y) = \sum_{x,y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)} \tag{5.4}$$

In [12] Danon et al. defined its normalization as :

$$I_{norm}(X, Y) = NMI = \frac{2I(X : Y)}{H(X) + H(Y)} \tag{5.5}$$

where

$$I(X:Y) = H(X) - H(X|Y) \tag{5.6}$$

NMI has become the most widely used index for evaluation of clustering and community detection methods.

Since then many more [36]have proposed different definitions of NMI dtrying to increase its ability to assign importance to small clusters most efficiently.

In [34], Lancichinetti defined NMI as:

Definition 5.8.1.

$$NMI_{lfk} = 1 - \frac{1}{2}\left(\frac{H(X|Y)}{H(X)} + \frac{H(Y|X)}{H(Y)}\right) \tag{5.7}$$

where

- $H(X), H(Y)$ are the marginal entropies, regarded as a measure of uncertainty about a random variable.
- $H(X|Y)$ and $H(Y|X)$ are the conditional entropies.The conditional entropy $H(Y|X)$ could be described as the amount of uncertainty in Y which remains, after X is known".
- $H(X,Y)$ is the joint entropy, that intuitively measures how much knowing one of these variables reduces uncertainty about the other.

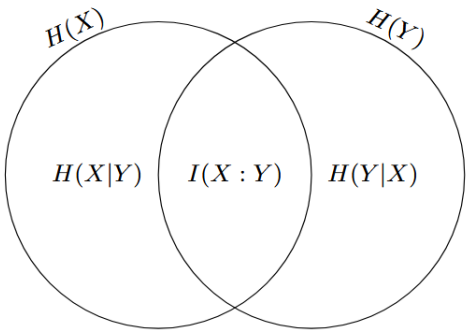


Figure 5.1: Mutual information and variation of information. The total information $H(X,Y) = H(X|Y) + I(X:Y) + H(Y|X)$, Aaron F. McDaid [39]

In Figure 5.1 the following useful identities, are visualized.

$$
\begin{aligned}
H(X) &= I(X:Y) + H(X|Y) \\
H(Y) &= I(X:Y) + H(Y|X) \\
H(X,Y) &= H(X) + H(Y|X) \\
H(X,Y) &= H(Y) + H(X|Y) \\
H(X,Y) &= \overbrace{I(X:Y)}^{\text{mutual information}} + \overbrace{H(X|Y) + H(Y|X)}^{\text{variation of information}}
\end{aligned}
$$

The mutual information is defined as:

$$
I(X:Y) := \frac{1}{2}\left[H(X) - H(X|Y) + H(Y) - H(Y|X)\right] \tag{5.8}
$$

The following sequence of inequalities [57] provide different options for the normalization factor.

$$
\begin{aligned}
I(X:Y) &\leq \min(H(X), H(Y)) \\
&\leq \sqrt{H(X), H(Y)} \\
&\leq \frac{1}{2}\left(H(X) + H(Y)\right) \\
&\leq \max(H(X), H(Y)) \\
&\leq H(X,Y)
\end{aligned} \tag{5.9}
$$

McDaid et al. in [39] noticed that when the number of communities is getting smaller, and one cluster is much bigger or complicated than the other, then the overlap between in 5.1 will be quite large, almost the size of the smaller circle. As a result, one of the terms inside the brackets in 5.7 will be small and will bring the NMI_{lfk} to 0.5. This value is not correct as we would expect a value closer to 0.

McDaid et al. [39] proposed the following NMI definition that is calculated during the evaluation.

Definition 5.8.2.

$$
NMI_{max}(X,Y) = \frac{1}{2}\frac{H(X) + H(Y) - H(X|Y) - H(Y|X)}{max(H(X), H(Y))} \tag{5.10}
$$

5.9 Experimental setting

Both models were trained on a single K80 GPU with 12GB of RAM at a time. Both models were implemented using PyTorch and PyTorch Geometric. Our code implementation is based on the code implementation of [50].

Chapter 6

Results and discussion

6.1 Recovery performance

The performance results of the experiments are shown on Table 6.1. The ability of the two models to recover the ground-truth communities is measured in NMI. All results are averaged over 10 initializations and the standard deviation of the NMI values is calculated.

For the larger co-authorship datasets, **Chemistry, Computer Science, Medicine, Engineering**, a much better performance is achieved when the feature matrix is used as an input for both models. The larger sets have a much larger dimension of node feature vectors also, that seems to be better handled by the GCN and the GAT neural networks.

Our proposed method using a GAT network achieves the best performance on 5 out of 10 datasets. Especially the best performance is achieved for three out of four larger co-authorship datasets. Particularly for the 4 co-authorship datasets, a great improvement in GAT's performance is achieved when the node feature matrix is used as an input.

For the Engineering dataset an improvement of almost 20% in NMI is achieved for our proposed method, when the node feature matrix is used as an input versus the use of the adjacency matrix as an input. The same increase is observed also for the baseline model using a GCN, but the performance gain is bigger for our proposed method. We can conclude that when a larger dimension node feature is used a s an input it is better handled by the GAT proposed method and it achieves an increase in community structure recovery when compared to the baseline GCN model.

For the smaller **Facebook** datasets, where the node feature dimension is not very big, a similar improvement is not observed. For these datasets best performance is produced when the adjacency matrix is used as an input.

The standard deviation of the NMI value measured ,over 10 different initializations, is smaller for our proposed GAT model. This indicates that the GAT model is a bit more stable.

	Adjacency		Attributes	
Dataset	GCN	GAT	GCN	GAT
Facebook 348	30.9 ± 1.4	**33.4 ± 3.5**	29.4 ± 2.6	30.3 ± 2.9
Facebook 414	**52.1 ± 3.5**	50.6 ± 1.9	51.0 ± 4.5	51.3 ± 4.6
Facebook 686	**17.7 ± 1.3**	17.1 ± 0.8	16.0 ± 1.9	15.3 ± 2.1
Facebook 698	41.2 ± 3.7	**45.1 ± 3.2**	36.9 ± 6.4	31.1 ± 9.1
Facebook 1684	**39.6 ± 1.5**	33.7 ± 3.0	26.3 ± 1.5	30.2 ± 2.7
Facebook 1912	**43.2 ± 0.8**	39.5 ± 1.7	33.0 ± 3.6	34.7 ± 2.9
Chemistry	19.0 ± 2.6	19.2 ± 1.1	41.3 ± 3.0	**41.8 ± 2.3**
Computer Science	29.9 ± 1.8	29.6 ± 1.4	**46.5 ± 3.4**	46.1 ± 1.6
Engineering	19.4 ± 1.0	15.2 ± 1.4	34.8 ± 3.4	**35.6 ± 1.3**
Medicine	27.4 ± 0.9	23.6 ± 2.7	34.9 ± 3.4	**35.2 ± 2.7**

Table 6.1: Recovery of ground-truth communities, measured by NMI (in %) with standard deviation. Results are averaged over 10 initializations

6.2 Scalability

The graph attention neural overlapping community detection model is very scalable, as shown on table 6.2. The **Medicine** dataset (810K edges), the largest of the datasets, is trained in less than 3 minutes, on a single GPU with 12GB RAM. The time performance is a little bit higher when compared to the NOCD method, without affecting the overall scalability.

	Adjacency		Attributes	
Dataset	GCN	GAT	GCN	GAT
Facebook 348	6.729	9.395	6.463	9.091
Facebook 414	6.096	8.733	6.132	8.958
Facebook 686	6.531	9.032	6.509	9.072
Facebook 698	7.002	9.166	6.745	9.134
Facebook 1684	11.633	9.745	7.043	9.668
Facebook 1912	6.895	14.426	11.186	15.385
Chemistry	26.078	43.094	29.209	71.867
Computer Science	20.067	31.075	28.482	79.655
Engineering	13.031	19.183	15.766	34.84
Medicine	78.674	169.594	74.745	114.041

Table 6.2: Real time of execution, measured in seconds. Results are averaged over 10 initializations

Chapter 7

Conclusions and future work

7.1 Conclusions

In this thesis, we analyzed a model that combines successfully a probabilistic view and representation learning, with deep graph geometric learning for the discovery of overlapping community detection. The NOCD model was extended with the use of a Graph Attention mechanism . Evaluation experiments were performed to evaluate our proposed extension's performance compared to the baseline's. The results confirmed both model's ability to discover graph overlapping community structure and it's scalability. When a Graph Attention mechanism is used, the model produces an increase in performance and stability, especially for larger networks, with higher dimension node feature datasets.

7.2 Future work

A number of questions to be answered in the future. One interesting question that is raised, after experimenting with Graph Attention Networks is whether the quantification of the attention coefficients relevance to the community structure, could be helpful on creating interpretability analysis of the model.

The assessment of the models inductive performance [22] is an interesting area of future work. For example the performance of induction on one network while trained on another. Or to a wider part of the graph, when trained on a smaller one. Inductive performance could also be evaluated on dynamically evolving networks.

So far, some interesting improvements to graph attention networks have been proposed. Further tests with graph attention modifications for example proposed in , [6, 69] could be performed.

For most of the overlapping community methods, a most crucial issue that

remains open, is their dependence on explicitly declaring the number of communities K as an input. In real world datasets, there is no prior knowledge of the number of communities that exist. Research should be focused on the question of how to determine the number of communities hidden in a graph.

Bibliography

[1] Yong-Yeol Ahn, James P Bagrow, and Sune Lehmann. "Link communities reveal multiscale complexity in networks". In: *nature* 466.7307 (2010), pp. 761–764.

[2] Edo M Airoldi, David Blei, Stephen Fienberg, and Eric Xing. "Mixed membership stochastic blockmodels". In: *Advances in neural information processing systems* 21 (2008).

[3] Arash A Amini, Aiyou Chen, Peter J Bickel, and Elizaveta Levina. "Pseudo-likelihood methods for community detection in large sparse networks". In: *The Annals of Statistics* 41.4 (2013), pp. 2097–2122.

[4] Earl R Barnes. "An algorithm for partitioning the nodes of a graph". In: *SIAM Journal on Algebraic Discrete Methods* 3.4 (1982), pp. 541–550.

[5] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. "Fast unfolding of communities in large networks". In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[6] Shaked Brody, Uri Alon, and Eran Yahav. "How attentive are graph attention networks?" In: *arXiv preprint arXiv:2105.14491* (2021).

[7] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. "Spectral networks and locally connected networks on graphs". In: *arXiv preprint arXiv:1312.6203* (2013).

[8] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. "A comprehensive survey of graph embedding: Problems, techniques, and applications". In: *IEEE Transactions on Knowledge and Data Engineering* 30.9 (2018), 1616–1637.

[9] Junyang Chen, Zhiguo Gong, Jiqian Mo, Wei Wang, Cong Wang, Xiao Dong, Weiwen Liu, and Kaishun Wu. "Self-training enhanced: Network embedding and overlapping community detection with adversarial learning". In: *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[10] Zhengdao Chen, Xiang Li, and Joan Bruna. "Supervised community detection with line graph neural networks". In: *arXiv preprint arXiv:1705.08415* (2017).

[11] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. "Finding community structure in very large networks". In: *Physical review E* 70.6 (2004), p. 066111.

[12] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. "Comparing community structure identification". In: *Journal of statistical mechanics: Theory and experiment* 2005.09 (2005), P09008.

[13] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in neural information processing systems* 29 (2016).

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[15] Gary William Flake, Steve Lawrence, and C Lee Giles. "Efficient identification of web communities". In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000, pp. 150–160.

[16] Santo Fortunato. "Community detection in graphs". In: *Physics reports* 486.3-5 (2010), pp. 75–174.

[17] Santo Fortunato and Darko Hric. "Community detection in networks: A user guide". In: *Physics reports* 659 (2016), pp. 1–44.

[18] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. "Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding". In: *Proceedings of The Web Conference 2020*. 2020, pp. 2331–2341.

[19] Anne-Claude Gavin, Patrick Aloy, Paola Grandi, Roland Krause, Markus Boesche, Martina Marzioch, Christina Rau, Lars Juhl Jensen, Sonja Bastuck, Birgit Dümpelfeld, et al. "Proteome survey reveals modularity of the yeast cell machinery". In: *Nature* 440.7084 (2006), pp. 631–636.

[20] Michelle Girvan and Mark EJ Newman. "Community structure in social and biological networks". In: *Proceedings of the national academy of sciences* 99.12 (2002), pp. 7821–7826.

[21] Mark S Granovetter. "The strength of weak ties". In: *American journal of sociology* 78.6 (1973), pp. 1360–1380.

[22] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30 (2017).

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[24] Drahomira Herrmannova and Petr Knoth. "An analysis of the microsoft academic graph". In: *D-lib Magazine* 22.9/10 (2016), p. 37.

[25] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. "Stochastic blockmodels: First steps". In: *Social networks* 5.2 (1983), pp. 109–137.

[26] Cho-Jui Hsieh and Inderjit S Dhillon. "Fast coordinate descent methods with variable selection for non-negative matrix factorization". In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2011, pp. 1064–1072.

[27] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[28] Yuting Jia, Qinqin Zhang, Weinan Zhang, and Xinbing Wang. "Communitygan: Community detection with generative adversarial nets". In: *The World Wide Web Conference*. 2019, pp. 784–794.

[29] Baoyu Jing, Chanyoung Park, and Hanghang Tong. "Hdmi: High-order deep multiplex infomax". In: *Proceedings of the Web Conference 2021*. 2021, pp. 2414–2424.

[30] Brian Karrer and Mark EJ Newman. "Stochastic blockmodels and community structure in networks". In: *Physical review E* 83.1 (2011), p. 016107.

[31] Brian W Kernighan and Shen Lin. "An efficient heuristic procedure for partitioning graphs". In: *The Bell system technical journal* 49.2 (1970), pp. 291–307.

[32] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR (Poster)*. 2015.

[33] Nevan J Krogan, Gerard Cagney, Haiyuan Yu, Gouqing Zhong, Xinghua Guo, Alexandr Ignatchenko, Joyce Li, Shuye Pu, Nira Datta, Aaron P Tikuisis, et al. "Global landscape of protein complexes in the yeast Saccharomyces cerevisiae". In: *Nature* 440.7084 (2006), pp. 637–643.

[34] Andrea Lancichinetti and Santo Fortunato. "Community detection algorithms: a comparative analysis". In: *Physical review E* 80.5 (2009), p. 056117.

[35] Chih-Jen Lin. "Projected gradient methods for nonnegative matrix factorization". In: *Neural computation* 19.10 (2007), pp. 2756–2779.

[36] Xin Liu, Hui-Min Cheng, and Zhong-Yuan Zhang. "Evaluation of community detection methods". In: *IEEE Transactions on Knowledge and Data Engineering* 32.9 (2019), pp. 1736–1746.

[37] Linhao Luo, Yixiang Fang, Xin Cao, Xiaofeng Zhang, and Wenjie Zhang. "Detecting communities from heterogeneous graphs: A context path-based graph neural network model". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 1170–1180.

[38] Julian Mcauley and Jure Leskovec. "Discovering social circles in ego networks". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 8.1 (2014), pp. 1–28.

[39] Aaron F McDaid, Derek Greene, and Neil Hurley. "Normalized mutual information to evaluate overlapping community finding algorithms". In: *arXiv preprint arXiv:1110.2515* (2011).

[40] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Icml*. 2010.

[41] Mark EJ Newman. "Fast algorithm for detecting community structure in networks". In: *Physical review E* 69.6 (2004), p. 066133.

[42] Mark EJ Newman. "Modularity and community structure in networks". In: *Proceedings of the national academy of sciences* 103.23 (2006), pp. 8577–8582.

[43] J-P Onnela, Jari Saramäki, Jorkki Hyvönen, György Szabó, David Lazer, Kimmo Kaski, János Kertész, and A-L Barabási. "Structure and tie strengths in mobile communication networks". In: *Proceedings of the national academy of sciences* 104.18 (2007), pp. 7332–7336.

[44] Aris Pagourtzis, Dora Souliou, Petros Potikas, and Katerina Potika. "Overlapping community detection via minimum spanning tree computations". In: *2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService)*. IEEE. 2020, pp. 62–65.

[45] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. "Uncovering the overlapping community structure of complex networks in nature and society". In: *nature* 435.7043 (2005), pp. 814–818.

[46] Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. "Unsupervised attributed multiplex network embedding". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 5371–5378.

[47] Pascal Pons and Matthieu Latapy. "Computing communities in large networks using random walks". In: *International symposium on computer and information sciences*. Springer. 2005, pp. 284–293.

[48] Martin Rosvall and Carl T Bergstrom. "Maps of random walks on complex networks reveal community structure". In: *Proceedings of the national academy of sciences* 105.4 (2008), pp. 1118–1123.

[49] Satu Elisa Schaeffer. "Graph clustering". In: *Computer science review* 1.1 (2007), pp. 27–64.

[50] Oleksandr Shchur and Stephan Günnemann. "Overlapping community detection with graph neural networks". In: *arXiv preprint arXiv:1909.12201* (2019).

[51] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.

[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[53] Xing Su, Shan Xue, Fanzhen Liu, Jia Wu, Jian Yang, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Di Jin, et al. "A comprehensive survey on community detection with deep learning". In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[54] Adrien Todeschini, Xenia Miscouridou, and François Caron. "Exchangeable random measures for sparse and modular graphs with overlapping communities". In: *arXiv preprint arXiv:1602.02114* (2016).

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[56] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. "Graph attention networks". In: *arXiv preprint arXiv: 1710.10903* (2017).

[57] Nguyen Xuan Vinh, Julien Epps, and James Bailey. "Information theoretic measures for clusterings comparison: is a correction for chance necessary?" In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 1073–1080.

[58] Xiao Wang, Nian Liu, Hui Han, and Chuan Shi. "Self-supervised heterogeneous graph neural network with co-contrastive learning". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 1726–1736.

[59] Max Welling and Thomas N Kipf. "Semi-supervised classification with graph convolutional networks". In: *J. International Conference on Learning Representations (ICLR 2017)*. 2016.

[60] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How powerful are graph neural networks?" In: *arXiv preprint arXiv:1810.00826* (2018).

[61] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "Representation learning on graphs with jumping knowledge networks". In: *International conference on machine learning*. PMLR. 2018, pp. 5453–5462.

[62] Jaewon Yang and Jure Leskovec. "Community-affiliation graph model for overlapping network community detection". In: *2012 IEEE 12th international conference on data mining*. IEEE. 2012, pp. 1170–1175.

[63] Jaewon Yang and Jure Leskovec. "Overlapping community detection at scale: a nonnegative matrix factorization approach". In: *Proceedings of the sixth ACM international conference on Web search and data mining*. 2013, pp. 587–596.

[64] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. "Graph convolutional neural networks for web-scale recommender systems". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 974–983.

[65] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. "Hierarchical graph representation learning with differentiable pooling". In: *Advances in neural information processing systems* 31 (2018).

[66] Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, and Jure Leskovec. "Identity-aware graph neural networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 12. 2021, pp. 10737–10745.

[67] Jiaxuan You, Zhitao Ying, and Jure Leskovec. “Design space for graph neural networks”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17009–17021.

[68] Fataneh Dabaghi Zarandi and Marjan Kuchaki Rafsanjani. “Community detection in complex networks using structural similarity”. In: *Physica A: Statistical Mechanics and its Applications* 503 (2018), pp. 882–891.

[69] Hengrui Zhang, Zhongming Yu, Guohao Dai, Guyue Huang, Yufei Ding, Yuan Xie, and Yu Wang. “Understanding gnn computational graph: A coordinated computation, io, and memory perspective”. In: *Proceedings of Machine Learning and Systems* 4 (2022), pp. 467–484.

[70] Muhan Zhang and Yixin Chen. “Link prediction based on graph neural networks”. In: *Advances in neural information processing systems* 31 (2018).

[71] Yao Zhang, Yun Xiong, Yun Ye, Tengfei Liu, Weiqiang Wang, Yangyong Zhu, and Philip S Yu. “SEAL: Learning heuristics for community detection with generative adversarial networks”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 1103–1113.

[72] Mingyuan Zhou. “Infinite edge partition models for overlapping community detection and link prediction”. In: *Artificial intelligence and statistics*. PMLR. 2015, pp. 1135–1143.