Θεωρία γραφημάτων και εφαρμογές (ICE 8108) Εξαμηνιαία εργασία

Τμήμα Μηχανικών Πληροφορικής & Υπολογιστών

ΑΜ:21390174

ΑΜ:21390119

Ονοματεπώνυμο: Φίλιππος Παπαγεωργίου
Ονοματεπώνυμο: Λυπράνδος Λιάτσος

Ημερομηνία παράδοσης :

Σάββατο 12 Ιουλίου 2025 - 11:00 μ.μ.

Υπεύθυνος Διδάσκοντας :

Γεώργιος Δρακόπουλος

1 Θεωρητικό μέρος (4 μονάδες)

1.2 Εκθετικό μητρείο γειτνίασης

Αποδείξτε πως στην περίπτωση όπου το μητρείο γειτνίασης A έχει φάσμα τέτοιο ώστε η αριθμητική και γεωμετρική πολλαπλότητα κάθε ιδιοτιμής να ισούται με την μονάδα το αντίστοιχο εκθετικό μητρείο e^A έχει θετικές ιδιοτιμές.

ΛΥΣΗ:

Αρχικά "Αριθμητική και γεωμετρική πολλαπλότητα = 1" σημαίνει ότι το μητρείο A είναι διαγωνιοποιήσιμο με απλές ιδιοτιμές , επίσης βάση βιβλιογραφίας και παρακολούθηση μαθήματων γνωρίζουμε πως :

- Αριθμητική πολλαπλότητα = πόσες φορές εμφανίζεται μια ιδιοτιμή στο χαρακτηριστικό πολυώνυμο
- Γεωμετρική πολλαπλότητα = διάσταση του eigenspace

Αν και οι δύο = 1, τότε το μητρείο είναι **διαγωνιοποιήσιμο** με **απλές ιδιοτιμές**.

Αν το A είναι διαγωνιοποιήσιμο σημαίνει αυτό μαθηματικά;

Σημαίνει ότι υπάρχουν μητρεία P (αντιστρέψιμο) και D (διαγώνιο) τέτοια ώστε:

A = PDP^(-1) όπου D = diag(λ_1, λ_2, ..., λ_n) και τα λ_i είναι οι ιδιοτιμές του A.

Ξεκινώντας με τον ορισμό του εκθετικού μητρείου το οποίο είναι

e^A = I + A + A^2/2! + A^3/3! + A^4/4! + ... , όποτε τώρα αν A = PDP^(-1) θα δούμε τι γίνεται με τις δυνάμεις του A
A^2 = (PDP^(-1))(PDP^(-1)) = PD(P^(-1)P)DP^(-1) = PDDP^(-1) = PD^2P^{-1}
A^3 = A^2·A = (PD^2P^{-1})(PDP^(-1)) = PD^2(P^(-1)P)DP^(-1) = PD^3P^{-1}

Παρατηρούμε το μοτίβο A^n = PD^n P^(-1). Τώρα πάμε να αντικαταστήσουμε στον ορισμό του e^A

$$e^A = I + A + A^2/2! + A^3/3! + ... = I + PDP^(-1) + PD^2P^(-1)/2! + PD^3P^(-1)/3! + ...$$

Βγάζοντας κοινό παράγοντα το P και P^(-1):

$$P(I + D + D^2/2! + D^3/3! + ...)P^(-1) = Pe^DP^(-1)$$

Επειδή PP^(-1) = I, οπότε: I = P·I·P^(-1)

τώρα μπορούμε να δούμε την διαγώνιο e^D αν

Αν D = diag(λ_1, λ_2, ..., λ_n), τότε:

D^n = diag(λ_1^n, λ_2^n, ..., λ_n^n)

Επειδή όταν υψώνεις διαγώνιο μητρείο σε δύναμη, απλώς υψώνεις κάθε στοιχείο της διαγωνίου στην ίδια δύναμη.

$$e^D = I + D + D^2/2! + D^3/3! + ... = diag(1,1,...,1) + diag(\lambda_1, \lambda_2, ..., \lambda_n) + diag(\lambda_1^2, \lambda_2^2, ..., \lambda_n^2)/2! + ... = diag(1$$

μέχρι στιγμής εχούμε :

- A = PDP^(-1) όπου D = diag(λ_1, λ_2, ..., λ_n)
- e^A = P e^D P^(-1)
- e^D = diag(e^λ_1, e^λ_2, ..., e^λ_n)

Άρα

$$e^A = P \cdot diag(e_1^\lambda, e_2^\lambda, ..., e_n^\lambda) \cdot P^(-1)$$

και οι ιδιωτιμές του e^A είναι **:** e^λ_1, e^λ_2, ..., e^λ_n Αυτό ισχύει επειδή αν δύο μητρεία είναι όμοια (similar), έχουν τις ίδιες ιδιοτιμές.
Βάση τα παραπάνω η συνάρτηση e^x είναι πάντα θετική για κάθε πραγματικό χ για παράδειγμα

- e^0 = 1 > 0
- e^1 ≈ 2.718 > 0
- e^(-10) ≈ 0.000045 > 0 (πολύ μικρό αλλά θετικό)
- e^100 = τεράστιος θετικός αριθμός

Γιατί; Επειδή e^x = exp(x) δεν είναι ποτέ μηδέν και δεν αλλάζει πρόσημο. Επομένως, επειδή e^λ_i > 0 για κάθε i = 1,2,...,n, αποδεικνύεται ότι το εκθετικό μητρείο e^A έχει θετικές ιδιοτιμές.

1.1 Κατανομή βαθμών

- Ποιά είναι η πιθανότητα μια τυχαία μεταβλητή X η οποία ακολουθεί την κανονική κατανομή της εξίσωσης (1) με μέση τιμή μ0 μηδέν και μοναδιαία διασπορά σ2 0 να πάρει την τιμή μηδέν;

ΛΥΣΗ:

Όταν έχουμε μια συνεχή τυχαία μεταβλήτη όπως η κανονική κατανομή , ποια είναι η πιθανότητα να πάρει ακριβώς μια συγκεκριμένη τιμή όπως το 0.

αν αντικαταστήσουμε με μ0 = 0 (μέση τιμή μηδέν) και σο^2 = 1 (μοναδιαιά διασπόρα , άρα σ0 = 1)

όποτε αν κάνουμε την αντικατάσταση εξίσωση μας θα γίνει

$$f_X(x; 0, 1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

Συμφώνα με την ερώτηση πρέπει να αποδείξουμε P(X= 0) για κανονική κατανομή. Γνωρίζουμε πως οι διακριτές κατανομές είναι τιμές μετρήσιμες (π.χ 1,2,3,4,5...) και επίσης P(X = K) μπορεί να είναι > 0 ενώ οι συνεχείς κατανομές οι τιμές είναι σε συνεχές διάστημα (π.χ όλοι πραγματικοί αριθμοί) και επιπλέον P(X = c) = 0 για κάθε τιμή c. Αυτό ισχύει διότι η πιθανότητα σε συνέχεις κατανομές υπολογίζεται ως

$$P(a \leq X \leq b) = \int_a^b f_X(x)\,dx$$

έτσι για ένα συγκεκριμένο σημείο c θα είναι :

$$P(X = c) = P(c \leq X \leq c) = \int_c^c f_X(x)\,dx = 0$$

Το ολοκλήρωμα είναι μηδέν επειδή το εύρος ολοκλήρωσης είναι μηδέν (από c εώς c)

Επίσης η πιθανότητα P(≤X≤b) είναι το εμβάδον κάτων από την καμπύλη f_X(x) από το a εώς το b ένα σημείο οπώς το (χ = 0) έχει μηδενικό πλάτος , άρα μηδενικό εμβαδό. Τώρα μπορούμε να εφαρμόσουμε στην κανονική κατανομή

$$f_X(0) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{0^2}{2}\right) = \frac{1}{\sqrt{2\pi}} \approx 0.3989$$

αυτή είναι η πυκνοτήτα πιθανότητας στο σημείο 0 αυτό σημαίνει ότι η πυκνοτήτα πιθανότητας είναι υψηλή άλλα η πιθανότητα για ακριβώς αυτό το σημείο είναι μηδέν.

Φυσική Ερμηνεία: Η f_X(0) = 0.3989 μας λέει**:** Κοντά στο 0, η κατανομή έχει υψηλή πυκνότητα ,το 0 είναι κοντά στο "κέντρο" της κατανομής (mode) και τα μικρά διαστήματα γύρω από το 0 έχουν σχετικά υψηλή πιθανότητα

Αλλά P(X = 0) = 0 Ένα σημείο έχει μηδενικό "εύρος" και δεν μπορούμε να "χτυπήσουμε" ακριβώς ένα σημείο σε συνεχή κατανομή

- Για ποιούς λόγους η κατανομή Gauss δεν αποτελεί καλή προσέγγιση της κατανομής των βαθμών των κορυφών σε γράφους ανεξάρτητους κλιμάκωσης (scale free graphs);

ΛΥΣΗ:

Αρχικά πάμε να αναλύσουμε βασικές έννοιες που θα μας βοηθήσουν να αποδείξουμε το παραπάνω ερώτημα.

Ξεκινώντας με τον βαθμό κορυφής ενός γράφου είναι ο βαθμός (degree) μιας κορυφής είναι ο αριθμός των ακμών που την συνδέουν με άλλες κορυφές και η κατανομή βαθμών πόσες κορυφές έχουν βαθμό 1 , πόσες βαθμό 2 κτλπ , δηλάδη για παράδειγμα αν είχαμε αυτό το γραφό

A----B----C
| | |
D----E----F

- Βαθμός(A) = 2 (συνδέεται με B, D)
- Βαθμός(B) = 3 (συνδέεται με A, C, E)
- Βαθμός(E) = 4 (συνδέεται με B, D, F, και έστω μια ακόμη)

Στην συνέχεια σημαντικό είναι να αναλύσουμε τι είναι Scale-free graphs : Είναι οι γραφοί οπού η κατανομή των βαθμών ακολουθεί power law δηλάδη P(k)∝k^−γ, όπου P(k) = πιθανότητα μια κορυφή να έχει βαθμό k, γ = εκθέτης power law και ∝ σημαίνει "ανάλογο με". Συνήθως scale-free graphs έχουν λίγες κορυφές άλλα εχούν υψηλό βαθμό (hubs) ή πολλές κορυφές και χαμήλο βαθμο και τέλος βαριές ουρές(μεγάλες τιμές με σημαντική πιθανότητα).

Τώρα πάμε να δούμε και να συκγρινούμε με την κατανομή Gauss κανονική κατανομή η οποία ως μαθηματική μορφή έχει

$$P(k) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(k-\mu)^2}{2\sigma^2}\right)$$

Η κανονική κατανομή είναι συμμετρική γύρω από τη μέση τιμή μ, κάτι που σημαίνει ότι οι τιμές κατανέμονται ισότιμα εκατέρωθεν του κέντρου. Αυτή η συμμετρία προκύπτει από την τετραγωνική μορφή του εκθέτη, η οποία εξασφαλίζει ότι αποκλίσεις ίσου μεγέθους προς τα πάνω και προς τα κάτω από τη μέση τιμή έχουν την ίδια πιθανότητα.
Η μορφή της κατανομής μοιάζει με καμπάνα (bell-shaped curve), επειδή η εκθετική συνάρτηση με αρνητικό εκθέτη δημιουργεί μια ομαλή καμπύλη που ξεκινάει από χαμηλές τιμές, φτάνει σε μέγιστο στη μέση τιμή, και στη συνέχεια πέφτει συμμετρικά.
Ένα κρίσιμο χαρακτηριστικό της Gaussείναι ότι έχει λεπτές ουρές (thin tails). Αυτό σημαίνει ότι καθώς απομακρυνόμαστε από τη μέση τιμή, η πιθανότητα πέφτει εκθετικά γρήγορα. Συγκεκριμένα, λόγω του όρου exp(-(k-μ)²/2σ²), ακραίες τιμές που είναι πολλές τυπικές αποκλίσεις μακριά από τη μέση τιμή έχουν εκθετικά μειούμενη πιθανότητα εμφάνισης.

Τέλος, η κανονική κατανομή εμφανίζει ισχυρή κεντρική τάση, καθώς οι περισσότερες παρατηρήσεις συγκεντρώνονται σε ένα σχετικά στενό εύρος γύρω από τη μέση τιμή. Αυτό την καθιστά ιδανική για φαινόμενα όπου υπάρχει μια "τυπική" τιμή και οι αποκλίσεις από αυτήν είναι αποτέλεσμα πολλών μικρών, ανεξάρτητων παραγόντων.

Από την άλλη πλευρά, η Power Law κατανομή με μαθηματική μορφή P(k) = Ck^(-γ) όπου C είναι σταθερά κανονικοποίησης, παρουσιάζει θεμελιωδώς διαφορετικά χαρακτηριστικά. Η power law κατανομή είναι θεμελιωδώς ασύμμετρη και μονότονα φθίνουσα, καθώς η πιθανότητα είναι αντιστρόφως ανάλογη με μια δύναμη της τιμής k. Όσο αυξάνεται το k, η πιθανότητα μειώνεται σταθερά, χωρίς να υπάρχει κάποιο σημείο συμμετρίας ή μέγιστο εκτός από την αρχική τιμή. Το πιο χαρακτηριστικό γνώρισμα της power law είναι οι βαριές ουρές (heavy tails), καθώς σε αντίθεση με την εκθετική πτώση της Gaussian, η πιθανότητα πέφτει πολυωνυμικά. Αυτό επιτρέπει σε ακραίες τιμές, αν και σπάνιες, να εξακολουθούν να έχουν σημαντική πιθανότητα εμφάνισης, κάτι που θα ήταν στατιστικά αδύνατο σε μια Gauss κατανομή.

Η σύγκριση των δύο κατανομών στις ακραίες περιοχές (ουρές) αποκαλύπτει τη θεμελιώδη διαφορά τους. Στην Gausskατανομή, η συμπεριφορά χαρακτηρίζεται από την έκφραση $P(k >> \mu) \sim \exp(-k^2/2\sigma^2)$, όπου η πιθανότητα πέφτει εκθετικά γρήγορα καθώς οι τιμές απομακρύνονται από τη μέση τιμή. Αυτό καθιστά τις ακραίες τιμές σχεδόν αδύνατες, με τιμές που απέχουν περισσότερες από 4-5 τυπικές αποκλίσεις να έχουν πιθανότητα μικρότερη από 10^(-6). Αντίθετα, στην Power Law κατανομή, η συμπεριφορά περιγράφεται από την P(k >> k_min) ~ k^(-γ), όπου η πιθανότητα πέφτει πολυωνυμικά, όχι εκθετικά. Αυτή η αργή πτώση επιτρέπει σε μεγάλες τιμές να παραμένουν στατιστικά σημαντικές, ακόμη και όταν είναι εκατοντάδες ή χιλιάδες φορές μεγαλύτερες από την τυπική κλίμακα.

Για να κατανοήσουμε καλύτερα αυτές τις διαφορές, ας εξετάσουμε έναν υποθετικό γράφο με 1000 κορυφές. Σε ένα Gauss σενάριο με μέση τιμή μ=10 και τυπική απόκλιση σ=3, η συντριπτική πλειοψηφία των κορυφών (99%) θα έχει βαθμό μεταξύ 4 και 16, με μέγιστο βαθμό περίπου 20. Δεν θα υπάρχουν κορυφές που να ξεχωρίζουν δραματικά - δεν θα υπάρχουν "hubs". Σε αντίθεση, σε ένα Power Law σενάριο με εκθέτη γ=2.5, το 80% των κορυφών θα έχει χαμηλό βαθμό (1-5), το 15% μέτριο βαθμό (6-20), αλλά το 5% θα έχει εξαιρετικά υψηλό βαθμό (21+), δημιουργώντας "super-hub" κορυφές με βαθμό 100, 200 ή περισσότερο.

Οι δομικοί λόγοι που καθιστούν τα scale-free networks ασύμβατα με την Gauss κατανομή έγκεινται στους μηχανισμούς δημιουργίας τους. Τα scale-free networks δημιουργούνται κυρίως μέσω του φαινομένου της "προτιμησιακής σύνδεσης" (preferential attachment), όπου νέες κορυφές έχουν μεγαλύτερη πιθανότητα να συνδεθούν με κορυφές που ήδη έχουν υψηλό βαθμό - ένα φαινόμενο που περιγράφεται ως "οι πλούσιοι γίνονται πλουσιότεροι". Αυτός ο αυτοενισχυόμενος μηχανισμός οδηγεί σε ακραία ανισότητα, όπου λίγες κορυφές συγκεντρώνουν δυσανάλογα μεγάλο αριθμό συνδέσεων και γίνονται "hubs", ενώ η πλειοψηφία παραμένει με χαμηλό βαθμό.

Συμπερασματικά, η κατανομή Gauss δεν μπορεί να αποτελέσει καλή προσέγγιση για scale-free graphs για τέσσερις θεμελιώδεις λόγους: Πρώτον, έχει λεπτές ουρές ενώ τα scale-free δίκτυα χαρακτηρίζονται από βαριές ουρές που επιτρέπουν την ύπαρξη hubs. Δεύτερον, προβλέπει συμμετρία γύρω από τη μέση τιμή ενώ τα scale-free είναι εγγενώς ασύμμετρα. Τρίτον, δεν επιτρέπει ακραίες τιμές που αποτελούν το κύριο χαρακτηριστικό των scale-free δικτύων. Τέταρτον, προϋποθέτει τυχαίες και ανεξάρτητες διεργασίες ενώ τα scale-free δημιουργούνται από συγκεκριμένους μηχανισμούς όπως η προτιμησιακή σύνδεση που παράγουν συστηματική ανισότητα.

1.3 Συμπίεση μητρείου γειτνίασης

- Ποιές είναι οι διαφορές όταν το μητρείο γειτνίασης ενός γράφου παραγοντοποιείται μέσω της παραγοντοποιήσεως ιδιαζουσών τιμών (SVD) και μέσω του διδιάστατου διακριτού μετασχηματισμού συνημιτόνου (DCT2); Για να είναι ολοκληρωμένη η απάντησή σας, σκεφτείτε κατ΄ ελάχιστον την ερμηνεία των δύο μεθοδολογιών, τον χώρο τον οποίο καταλαμβάνουν οι αντίστοιχες συμπιεσμένες μορφές, και την πολυπλοκότητα υπολογισμού κάθε μετασχηματισμού. Επιπλέον διαφορές θα προσμετρηθούν θετικά.

ΛΥΣΗ

Singular Value Decomposition (SVD)

Ερμήνεια :

Για κάθε πραγματικό πίνακα A διαστάσεων m x n, υπάρχουν δύο ορθογώνιοι
πίνακες U = [u1]...[um] με m x m διαστάσεις και V = [v1]...[vn] με n x n,

όπου ισχύει UΣV*= A = diag(σ1,..,σp) .

- Πίνακας U : πραγματικός (ή και μιγαδίκος πίνακας) μήκους m×m
- Πίνακας Σ : διαγώνιος πίνακας με μή αρνητικά πραγματικά στοιχεία μήκους m×n
- Πίνακας V* : συζηγής ανάστροφος ορθομοναδίαιος πίνακας μήκους n×n

Ο SVD εν τέλει αποσυνθέτει τον αρχικό πίνακα A στους τρείς πίνακες UΣV* και έτσι παραγοντοποιείται το μητρώο γειτνίασης. Τα διανύσματα u και v των πινάκνω U και V ονομάζονται αριστερά-ιδιάζοντα και δεξία-ιδιάζοντα του σ.

Εφαρμογή στους γράφους:

Το SVD στο μητρείο γειτνίασης αποκαλύπτει τη δομή του γράφου. Τα ιδιάζοντα διανύσματα αντιστοιχούν σε κοινότητες κόμβων, ενώ οι ιδιάζουσες τιμές δείχνουν τη σημαντικότητα κάθε κοινότητας. Χρησιμοποιείται για clustering, ανίχνευση κοινοτήτων και μείωση διαστάσεων σε γράφους.

Χώρος:

Οι ιδιάζουσες (όπως περιγράφθηκαν παραπάνω) αναπαριστούν τον χώρο των στηλών και του μηδενοχώρου ενός πίνακα A.

Τα δέξια ιδιάζοντα διανύσματα (οι στήλες του πίνακα V) αντιστοιχούν στις μηδενικές ιδιάζουσες τιμές του πίνακα A και παράγουν τον μηδενοχώρο του.

Τα αριστέρα ιδιάζοντα διανύσματα (οι στήλες του πίνακα U) που επίσης αντιστοιχούν στις μη μηδενιζόμενες ιδιάζουες τιμές παράγουν τον χώρο των στηλών του A.

Επομένως, ο χώρος του συμπιεσμένου σε μορφή πίνακα A εξαρτάται από τις μη μηδενικές ιδιάζουσες τιμές . Πιο αναλυτικά, τα αριστέρα ιδιάζοντα διανύσματα του πίνακα U που σχετίζονται με τις μη μηδενικές ιδιάζουσες τιμές παράγουν τον k-διάστατο χώρο στηλών του πίνακα A και αντίστοιχα τα δεξία ιδίαζοντα διανύσματα του πίνακα V που σχετίζονται
με τις μηδενικές ιδίαζουσες τιμές, καθορίζουν τον μηδενοχώρο του A.

Η διάσταση του μηδενοχώρου είναι n-r αν ο πίνακας A περιέχει n στοιχεία και r οι μη μηδενικές ιδιάζουσες τιμές.

Πολυπλοκότητα :

Η άναλυση της πολυπλόκοτητας του μετασχημάτισμου SVD προκύπτει από τα δύο ακόλουθα βήματα :

Βήμα 1 : Ο αρχικός πίνακας A υποβιβάζεται σε εναν διαγώνιο πίνακα και έχει κόστος O(mn²) πράξεις κινήτης υποδιαστολής (ή αλλιως flops), αν m>= n (αλλίως είναι O(n^3) αν m=n).

Βήμα 2 : Υπολογίζεται η ανάλυση σε ιδιάζουσες τιμές με επαναληπτικές μεθόδους και απαιτεί O(n) για κάθε βήμα σε κόστος flops. Εν τέλει, η πολυπλόκοτητα είναι O(n²).

Εκ των δύο βημάτων, υψηλότερο κόστος έχει η πολυπλοκότητα του βήματος 1 και η τελική πολυπλοκότητα είναι O(mn²)

Διακριτός Μετασχηματισμός Συνημιτόνου (DCT2)

Ερμήνεια :

Συχνά, η συμπίεση εικόνας χρησιμοποιείτον Διακριτό Μετασχηματισμό Συνημίτονου (DCT2). Ο DCT2 μετασχηματίζει ένα δισδιάστατο σήμα σε ένα σύνολο συνιστώσων.

Ορίζεται από τον παράκατω μετασχηματισμό :

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad \text{for } k = 0, \ \ldots \ N-1 \, .$$

όπου:

- k : τρέχουσα συχνότητα
- N : μήκος της ακολουθίας x

Χώρος:

Ο DCT2 αναπαρίστα τον χώρο των συχνοτήτων του σήματος. Οι συντέλεστες του Xk αντιστοιχούν σε διάφορες συχνότητες (υψήλες ή χαμηλές) του σήματος.

- Χαμηλές συχνότητες : Αντιπροσωπεύουν τις βασικές δομές και τις γενικές μορφές του σήματος. Περιέχουν την περισσότερη ένεργεια του σήματος.
- Υψηλές συχνότητες : Περιέχουν λεπτομέρειες και πολλές φορές μπορούν να παραλειφθούν όσο γίνεται η συμπιέση. Αυτό, δεν έχει ως αποτέλεσμα την αλλοίωση της εικόνας ή του σήματος που μετασχηματίζεται. (οι υψηλές συχνότητες μπορεί να περιέχουν και θόρυβο)

Με τον όρο συμπίεση εννοούμε την αναπαράσταση του σήματος στον χώρο των συχνοτήτων που μειώνει την ποσότητα των δεδομένων που απαιτούνται για την αναπαράσταση ενός σήματος ή εικόνας. Διότι οι υψηλές συχνότητες περιέχουν αμελητέες τιμες και οι χαμηλές συχνότητες περιέχουν την ενέργεια του σήματος, οι υψηλές παραλείπονται.

Πολυπλόκοτητα :

Γενικά, η πολυπλοκότητα του Διακριτού Μετασχηματισμού Συνημιτόνου απαιτεί κόστος O(n^2). Ωστόσο, με διάφορους μετασχηματισμούς όπως Fast-Fourier Transform ή Fast- DCT, η πολυπλοκότητα ελαχιστοποιείται σε χρόνο O(nlogn). Οι γρήγοροι μετασχηματισμοί βασίζονται σε αλγορίθμους Divide and Conquer όπου το n είναι το σύνολο των στοιχείων της ακολουθίας X και logn ο αριθμός επίπεδων διαίρεσης του Divide and Conquer.

Πιο επιγραμματικά, οι δύο μετασχηματισμοί SVD και DCT2 διαφέρουν κατά τη λειτουργία τους. Ο SVD αποσυνθέτει έναν αρχικό πίνακα A στους τρείς πίνακες (U,Σ,V*) και ακολουθεί την ανάλυση των ιδιάζουσων τιμών και την αναπαράσταση του χώρου των στηλών και του μηδενοχώρου του αρχικού πίνακα A. Η πολυπλόκοτητα του είναι $O(mn^2)$. Αντίθετα, ο DCT2 μετατρέπει ένα δισδιάστατο σήμα σε ένα σύνολο συχνοτήτων με πολυπλοκότητα $O(n^2)$, αλλά με γρήγορους μετασχηματίσμους, μπορεί να μειωθεί σε O(nlogn). Επιπλέον, οι δύο αυτοί μετασχηματισμοί διαφέρουν και στην φύση των δεδομένων που επεξεργάζονται. Ο SVD μπορεί να εφαρμοστεί για οποιονδήποτε πραγματικό ή μιγαδικό πίνακα και χρησιμοποιείται σε εφαρμογές μηχανικής μάθησης, ανάλυση εικόνας κ.α. Ο DCT2 είναι σχεδιασμένος για σήματα και εικόνες και είναι κατάλληλος (αλλά και ταχύς) για μετάδοση και συμπίεση εικόνων. Ενώ και οι δύο μέθοδοι εστιάζουν στην μείωση της πολυπλόκοτητας, ο SVD παρέχει βέλτιστη προσέγγιση χαμηλού rank προσαρμοσμένη στα δεδομένα, ενώ ο DCT2 προσφέρει γρήγορη και αντιστρέψιμη συχνοτική ανάλυση.

1.4 Ταξινόμηση ακεραίων

Έστω πως έχουμε στην διάθεσή μας n διακριτούς αλλά όχι απαραίτητα συνεχόμενους ακέραιους.
• Αν κάθε τέτοιος ακέραιος αποτελεί την κορυφή ενός γράφου, τότε πώς μπορεί να αναχθεί η ταξινόμησή τους σε αναζήτηση συντομότερων διαδρομών; Τι δομή θα πρέπει να έχει αυτός ο γράφος; Στην περίπτωση αυτή, πόση θα είναι η πολυπλοκότητα του αλγορίθμου του Dijkstra και ποιός όρος κυριαρχεί στην πολυπλοκότητα; Πόση είναι η χωρική πολυπλοκότητα του εν λόγω γράφου; Τι παρατηρείτε;

ΛΥΣΗ

Έστω μια ακολουθία n διακριτών αριθμών (δηλάδη αριθμών που δεν επαναλαμβάνονται στην ακολουθία) που δεν είναι απαραίτητα συνεχόμενοι με όρους a1,a2,... an.

Αν ο κάθε κόμβος του γράφου αποτελεί και έναν ακέραιο αυτής της ακολουθίας, τότε οι κορύφες V(G) του γραφήματος G θα έχουν πληθάριθμο την ίδια την ακολουθία ($n = |V(G)|$). Για παράδειγμα, έστω μια ακολουθία {3, 8, 1, 12, 5}. Ο γράφος θα έχει κορυφές
$V(G) = \{3, 8, 1, 12, 5\}$ με $n = |V(G)| = 5$ κορυφές.

Για τον ίδιο γράφο, οι ακμές E(G), με πληθάριθμο m = |E(G)|, θα είναι n(n-1)/2 διότι για όλα τα ζεύγη κορυφών (ai,aj) με i≠j , η τοποθέτηση ακμής μεταξύ κορυφών i και j προκύπτει από την εξής συνθήκη :

- Αν ai < aj, τότε τοποθετείται ακμή μεταξύ του κόμβου i προς j
- Αν ai > aj, τότε δεν τοποθετείται ακμή μεταξύ του κόμβου i προς j

Άρα για κάθε ζεύγος παίρνουμε ακριβώς μια ακμή.

Για το παραπάνω αριθμητικό παράδειγμα που δόθηκε, έχουμε πως :
n = 5 με n(n-1)/2 = 5 (5-1) / 2 = 5*4 / 2 = 10.

Γραφικά, το παράπανω παράδειγμα θα είναι :
{3, 8, 1, 12, 5} που ταξινομείται 1 < 3 < 5 < 8 < 12 με ακμές
1 → 3 , 1→5 , 1→ 8, 1→ 12 (4 ακμές)
3 → 5, 3 → 8 , 3 → 12 (3 ακμές)

5 → 8 , 5 → 12 (2 ακμές)

8 → 12 (1 ακμή) με συνολικά 4 + 3 + 2 + 1 = 10 ακμές.

Επιρπόσθετα, το γράφημα πρέπει να είναι κατευθυνόμενο κάθως κάθε κόμβος θα συνδέεται μονόδρομα με τον κόμβο που έχει μεγαλύτερη τιμή από τον ίδιο. (μιας και η ακολουθία είναι διακριτή).

Παρατήρηση : Directed Acyclic Graph (ή DAG) είναι ένα κατευθύνομενο γράφημα που δεν περιέχει κατευθύνομενους κύκλους. Το πρόβλημα αυτό υπάγεται σε αυτόν τον ορισμό μιας και κανένας κόμβος δεν "επιστρέφει" πίσω σε προηγούμενο κόμβο, μιας και κάθε τιμή του κόμβου πρέπει να είναι αυστηρώς μικρότερος από τον επόμενο του, δηλάδη αυτόν που θα συνδεθεί με εξαγώμενη κατευθυνόμενη ακμή. Ο γράφος είναι πλήρες κατευθυνόμενο ακυκλικό γράφημα (complete DAG) με ακμές από κάθε μικρότερο αριθμό προς κάθε μεγαλύτερο.

Όσον αφορά τα βάρη των ακμών, θα τα ορίσουμε ως εξής :
Για κάθε ζεύγος κόμβων u προς v, το βάρος θα είναι weight(uv) = v - u, δηλαδή θα είναι η αριθμητική τους απόσταση (π.χ για τον κόμβο u = 5 προς τον v = 8, το βάρος της ακμής τους θα είναι 8-5 = 3). Τα βάρη πάντα θα είναι θετικά εφόσον η επόμενη ακμή είναι μεγαλύτερη από την προηγούμενη. Αυτό ισχύει διότι οι κόμβοι διακριτοί και ταξινομημένοι.

Η εφαρμόγη του αλγορίθμου Djikstra στον άνω γράφο είναι ισχυρή μιας και το γράφημα είναι κατευθύνομενο και θα ξεκινήσει να "τρέχει" από τον μικρότερο, σε τιμή, κόμβο. Θα εντοπίσει δηλάδη το min{V(G)} του γράφου. Έπειτα, θα εκτελεσθεί ο Djikstra και θα εντοπίσει όλα τα συντομότερα μόνοπατια από τον min{V(G)} προς κάθε άλλο κόμβο. Τελικά, το επιθυμήτο μονόπατι θα είναι το μονοπάτι από τον min{V(G)} προς τον max{V(G)} που συμπεριλαμβάνει όλους τους κόμβους σε αύξουσα σειρά.

Η χρονική πολυπλοκότητα του αλγοριθμού Djikstra σε γενική μορφή είναι : O((V+E) logV) με χρήση binary heap. Ωστόσο, η πολυπλόκοτητα του ίδιου αλγόριθμου μπορεί να ποικίλει ανάλογα τη χρήση διαφορετικού data structure (List : $O(V^2)$ ή Fibonacci heap : O(VlogV+E)). Η απόδειξη πολυπλόκοτητας είναι :
V = n και E = n(n-1) / 2 ≈ $n^2/2$, οπότε

$O((n + n^2/2) \log n)$
$= O((n + n^2) \log n)$ [αφού $n^2/2 = O(n^2)$]
$= O(n^2 \log n)$ [αφού n^2 κυριαρχεί του n]

Η χωρική πολυπλόκοτητα ορίζεται ως εξης :

Η αποθήκευση των κόμβων χρείαζεται χώρο O(n) και η αποθήκευση των ακμών χρείαζεται χώρο $O(n^2)$ διότι ορίσαμε πως E ≈ n^2 (προκύπτει από n(n-1)/2) . Συνεπώς προκύπτει πως
$O(n) + O(E) = O(n) + O(n^2) = O(n^2)$.

- Αν οι n! δυνατές διατάξεις των ανωτέρω ακεραίων τοποθετηθούν στα φύλλα ενός δυαδικού δέντρου και κάθε ενδιάμεση κορυφή αποτελεί μια σύγκριση μεταξύ δύο ακεραίων, τότε ποιό είναι το ύψος αυτού του δέντρου; Πόσο είναι το μήκος ενός μονοπατιού από την ρίζα προς κάθε φύλλο; Τι παρατηρείτε ως προς την πολυπλοκότητα και ποιά η διαφορά με εκείνη του προηγούμενου υποερωτήματος; Ποιά είναι τώρα η χωρική πολυπλοκότητα; Τι παρατηρείτε;

ΛΥΣΗ

Έστω μια ακολουθία n διακριτών αριθμών (δηλάδη αριθμών που δεν επαναλαμβάνονται στην ακολουθία) που δεν είναι απαραίτητα συνεχόμενοι με όρους a1,a2,... an.

Το ύψος του δυαδικού δέντρου με κάθε ενδιάμεσο κόμβο να είναι και μια σύγκριση μεταξύ δύο ακέραιων για n! διατάξεις θα είναι :
μεγίστος αριθμός αριθμός συγκρίσεων που απαιτούνται για μια πλήρη διάταξη.

Για παράδειγμα, αν έχουμε την απλή ακόλουθια 1,2,3 με n =3, θα έχουμε συνολικά 3! =6 διατάξεις. (Επιλέγω μια μικρή σε πλήθος ακολουθία διότι το παραγόντικο αυξάνεται με τρομακτικό ρυθμό). Οι διατάξεις που θα προκύψουν είναι :
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1 → σύνολο 6 διατάξεις

Για n! διατάξεις χρειάζομαι log2(n!) ≈ nlog2(n) συγκρίσεις
Συνεπώς έχουμε πως 2^h ≥ n! ⇒ h ≥ log2(n!).
Για την πρόσεγγιση αυτής της πράξης ώστε να βρούμε το ύψος του δέντρου, βασιζόμαστε στο Striling Approximation :

$$\log_2(n!) = n\log_2 n - n\log_2 e + O(\log_2 n).$$

Συνεπώς, το τελικό ύψος θα προσεγγίστει με χρόνικη πολύπλοκοτητα μιας και κάθε κόμβος είναι μια σύγκριση δύο ακεραίων. Τελικά επιτυγχάνουμε :
nlogn - n = nlogn - nlogn(e) = Ω(nlogn).

Το μήκος ένος μονοπάτιου από την ρίζα προς κάθε φύλλο είναι ο αριθμός των συγκρίσεων που πραγματοποιείται για να φτάσουμε στην εκάστοτε διάταξη.

Best-case scenario είναι η διάταξη να έχει εμφανίστει νωρίτερα στο δυαδικό δέντρο (άγνωστος αριθμός που ποικίλει ανάλογα με το μήκος της ακολουθίας και την δημιουργία των διάταξεων).
Worst-case scenario είναι η τελευταία προς δημιουργία διάταξη με μήκος όσο το ύψος του δέντρου που αντιστοιχεί σε h=⌈log2(n!)⌉.

Η πολύπλοκοτητα για ένα δυαδικό δέντρο είναι O(logn) για n κόμβους. Ώστοσο, κάθε ενδιάμεσος κόμβος είναι μια σύγκριση δύο ακεραίων από την ακολουθία n με n! διατάξεις. Συνεπώς, κάνουμε n! συγκρίσεις με O(log(n!)) που βάσει του Sterling Approximation, προκύπτει O(nlogn).

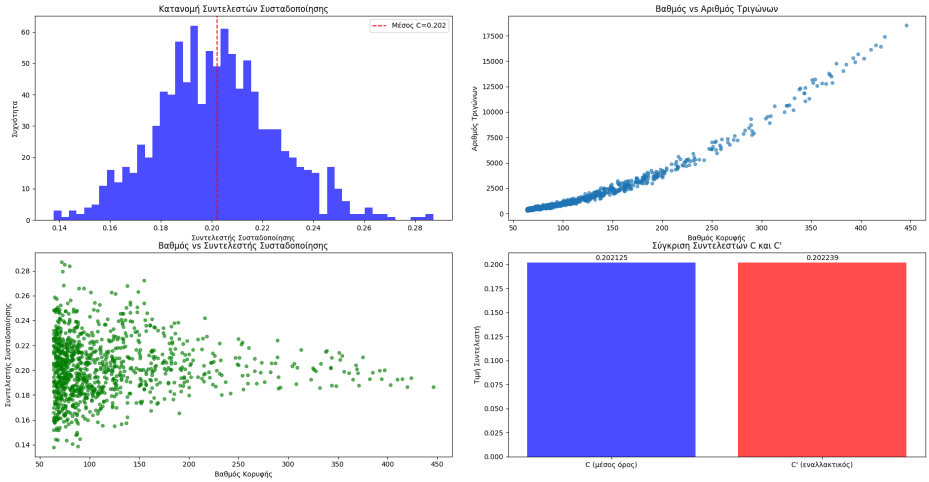
Η χωρική πολυπλόκοτητα είναι ο χώρος που καταλαμβάνουν τα n! φύλλα του δέντρου που σε ένα δυαδικό δέντρο αναζήτησης υπάρχουν 2n -1 κόμβοι. Για n! διατάξεις, έχουμε 2n!-1 κόμβους που εν τέλει χρειαζόμαστε χωρικά O(2n! -1) = O(n!) χωρικά.

ΣΥΝΟΠΤΙΚΟΣ ΟΔΗΓΟΣ - ΑΣΚΗΣΗ 2.1: ΑΝΑΖΗΤΗΣΗ ΚΟΡΥΦΩΝ

Η άσκηση 2.1 στοχεύει στην επίσκεψη όλων των κορυφών του γραφήματος graph1.csv με τους αλγορίθμους Depth-First-Search (DFS) και Breadth-First-Search(BFS) με κορύφη εκκίνησης την πρώτη (δηλαδή την 0 βάσει του γραφήματος graph1.csv). Η εκτέλεση του DFS γίνεται αναδρόμικα για κάθε γείτονα της κορύφης που βρισκόμαστε, ενώ η εκτέλεση του BFS γίνεται με χρήση ούρας και επισκέπτεται όλες τις κορύφες ανά επιπέδο πριν προχωρήσει στην επόμενη κορυφή. Για κάθε αλγόριθμο, μπορούμε να δούμε πόσες κορύφες επισκέφθηκε μέσω του len(dfs_result) και len(bfs_result) αντίστοιχα. Για τον συγκεκρίμενο γράφο, λάβαμε τα εξής αποτελέσματα :
DFS επισκέφθηκε 1024 κορυφές
BFS επισκέφθηκε 1024 κορυφές
Συνεπώς η επίσκεψη γίνεται σε κάθε κορύφη για κάθε έναν από τους δύο αλγορίθμους.
Παρότι γίνεται επίσκεψη σε όλες τις κορυφές του γραφήματος και από τους δύο αλγορίθμους, η σειρά με την οποία γίνεται η επίσκεψη είναι διαφορετική, μιας και συγκρίνουμε αν η λίστα dfs_result είναι ίδια με την bfs_result. Το αποτέλεσμα τους είναι false.
Αυτό συμβαίνει διότι ο κάθε αλγόριθμος λειτουργεί διαφορετικά. Ο DFS (Depth-First Search) επισκέπτεται έναν κόμβο u και εντοπίζει τον γείτονα του, προχωρώντας σε βάθος, δηλαδή επισκέπτοντας πρώτα όλους τους γείτονες του κόμβου u πριν επιστρέψει πίσω και εξερευνήσει άλλους γείτονες. Αντίθετα, ο BFS (Breadth-First Search) επισκέπτεται έναν κόμβο u και εξερευνά πρώτα όλους τους γείτονές του σε πλάτος, δηλαδή επισκέπτοντας όλους τους κόμβους που βρίσκονται σε απόσταση 1 από τον u πριν προχωρήσει στους γείτονες των γειτόνων του.
Και οι δύο αλγόριθμοι έχουν πολυπλοκότητα O(V+E) και χωρική πολυπλοκότητα O(V), όπου V οι κορυφές και E οι ακμές.

ΣΥΝΟΠΤΙΚΟΣ ΟΔΗΓΟΣ - ΑΣΚΗΣΗ 2.2: ΣΥΝΤΕΛΕΣΤΗΣ ΣΥΣΤΑΔΟΠΟΙΗΣΗΣ

Η άσκηση 2.2 στοχεύει στον υπολογισμό του συντελεστή συσταδοποίησης με δύο διαφορετικούς τύπους (C και C'), μετρώντας πόσο "συσταδοποιημένοι" είναι οι γείτονες κάθε κορυφής μέσω της εύρεσης τριγώνων στον γράφο. Η στρατηγική προσέγγιση περιλαμβάνει πέντε βασικά βήματα: φόρτωση του γράφου από CSV με σωστό mapping από vertex names σε indices (χρησιμοποιώντας `vertex_to_index = {v: i for i, v in enumerate(self.vertices)}` επειδή τα CSV μπορεί να έχουν μη-συνεχόμενες κορυφές), μετατροπή σε undirected γράφο με `self.adj_matrix = np.clip(self.adj_matrix + self.adj_matrix.T, 0, 1)` όπου το clip εξασφαλίζει τιμές 0,1 αντί για 2, βελτιστοποιημένη εύρεση τριγώνων με vectorized numpy operations (`submatrix = A[np.ix_(neighbor_indices, neighbor_indices)]` που μειώνει την πολυπλοκότητα από $O(n^3)$ σε $O(n \times d^2)$), υπολογισμός "δυνατών τριγώνων" με τον συνδυαστικό τύπο `open_triangles[i] = degree * (degree - 1) // 2` για βαθμό ≥ 2, και τέλος την κρίσιμη εφαρμογή των τύπων όπου **πρέπει οπωσδήποτε** στο C' να διαιρέσουμε με 3 (`C_prime = total_triangles_sum / 3 / total_open_triangles`) επειδή κάθε τρίγωνο μετράται 3 φορές από κάθε κορυφή του. Τα αποτελέσματά μας (C = 0.202125, C' = 0.202239 με διαφορά < 0.001) επιβεβαιώνουν την ορθότητα και δείχνουν έναν υψηλά συσταδοποιημένο γράφο με χαρακτηριστικό hub effect όπου κορυφές υψηλού βαθμού όπως η κορυφή 0 (βαθμός 420, clustering 0.187) έχουν χαμηλότερη τοπική συσταδοποίηση, φαινόμενο universal σε scale-free γράφους που αποδεικνύει την αρνητική συσχέτιση μεταξύ βαθμού και clustering coefficient.



ΣΥΝΟΠΤΙΚΟΣ ΟΔΗΓΟΣ - ΑΣΚΗΣΗ 2.4: ΑΛΓΟΡΙΘΜΟΣ DIJKSTRA

Η άσκηση 2.4 στοχεύει στην υλοποίηση του αλγορίθμου Dijkstra για εύρεση συντομότερων διαδρομών σε δύο διαφορετικούς γράφους (dijkstra1.csv και dijkstra2.csv), όπου κάθε ακμή έχει μοναδιαίο κόστος και χρειάζεται να βρούμε τη συντομότερη διαδρομή από την πρώτη στην τελευταία κορυφή. Η στρατηγική προσέγγιση περιλαμβάνει τέσσερα βασικά βήματα: δημιουργία αποδοτικής δομής δεδομένων γράφου με adjacency list για O(1) πρόσβαση σε γείτονες, robust φόρτωση CSV με error handling και σωστό parsing γραμμών που αρχίζουν με `#`, κλασική υλοποίηση Dijkstra χρησιμοποιώντας priority queue (`heapq`) για εύρεση της κορυφής με ελάχιστη απόσταση, διατήρηση `distances` dictionary για αποστάσεις και `previous` dictionary για ανακατασκευή διαδρομών, καθώς και **κρίσιμη βελτιστοποίηση** με early termination όταν βρίσκουμε τον στόχο (`if current_vertex == end_vertex: break`) που μειώνει την πολυπλοκότητα από $O(V^2)$ σε $O(E + V \log V)$, και τέλος έξυπνη ανακατασκευή διαδρομής με backtracking από τον προορισμό στην αρχή χρησιμοποιώντας το `previous` dictionary. Ο κώδικας χειρίζεται σωστά edge cases όπως unreachable nodes και παρέχει comprehensive output με step counting και formatted path display, ενώ η **βασική παρατήρηση** είναι ότι στο dijkstra1.csv που είναι δέντρο (edges = vertices - 1) υπάρχει μοναδική διαδρομή μεταξύ οποιωνδήποτε δύο κορυφών, επομένως ο Dijkstra θα βρει πάντα την ίδια διαδρομή με BFS, ενώ στο dijkstra2.csv που είναι πιο πυκνός γράφος μπορεί να υπάρχουν πολλαπλές διαδρομές και ο Dijkstra επιλέγει την κοντινότερη, αποδεικνύοντας την αξία του αλγορίθμου για general graphs και επιβεβαιώνοντας ότι σε δέντρα η συντομότερη διαδρομή είναι πάντα μοναδική και ίση με το μονοπάτι που συνδέει τις δύο κορυφές.