Çizge Üzerinde (Derin) Öğrenme

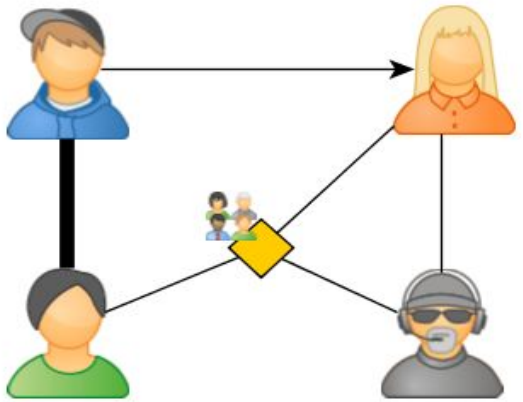
Kamer Kaya
Sabancı Üniversitesi

Sabancı Üniversitesi

Çizge Verisi

- Çizgeler veri içerisindeki elemanların birbirleri ile olan ilişkilerini modellemek için kullanılan yapılardır.

- Farklı nokta (*vertex*) ve kenar (*edge*) türleri farklı tür ilişkileri modellemek için kullanılabilir.



Çizge Verisi

Ağ	Noktalar	Nokta Özellikleri	Kenarlar	Kenar Özellikleri
Havayolu ağları	Hava alanları	Terminaller, şehir, nüfus, ulusal/uluslararası	Uçaklar, rotalar	Uçuş frekansı, yolcu sayısı, mesafe, benzin kullanımı, kapasite
Banka ağları	Müşteriler, hesap sahipleri	Kişi, ID, ürünler, hesap bakiyesi, kredi miktarı, demografik veri	İşlemler	Tip, miktar, güvenli/güvensiz, yer, zaman, araç
Sosyal ağlar	Kullanıcılar	İsim, demografik veri, beğeniler, gönderiler, üyelikler	Etkileşimler, mesajlar	Ortam, zaman, süre, konu, frekans
Medikal ağlar	Doktorlar	Demografik veri, uzmanlık, iş yeri bilgisi, günlük-haftalık ortalama hasta sayısı	Hastalar	Demografik, teşhis, tedavi, görüş sıklığı, sigorta
Tedarik zinciri ağı	Depolar	Yer, boyut, kapasite, indirme yükleme hızı, otomatik/el ile idame	Kamyonlar, rotalar	Yük miktarı, getiri, mesafe, sürücü, bakım masrafları

Çizge Verisi

Pazarlama Analitiği - Çizgeler, bir sosyal ağdaki en etkili kişileri bulmak için kullanılabilir. Reklam verenler ve Pazarlamacılar, mesajlarını bir Sosyal Ağdaki en etkili kişiler aracılığıyla yönlendirerek, pazarlama gelirinin en büyük patlamasını tahmin edebilirler.

Bankacılık İşlemleri - Çizgeler, dolandırıcılık işlemlerinin azaltılmasına yardımcı olan olağandışı kalıpları bulmak için kullanılabilir. Birbirine bağlı bankacılık ağları üzerinden para akışını analiz ederek yasadışı faaliyetin tespit edildiği örnekler olmuştur.

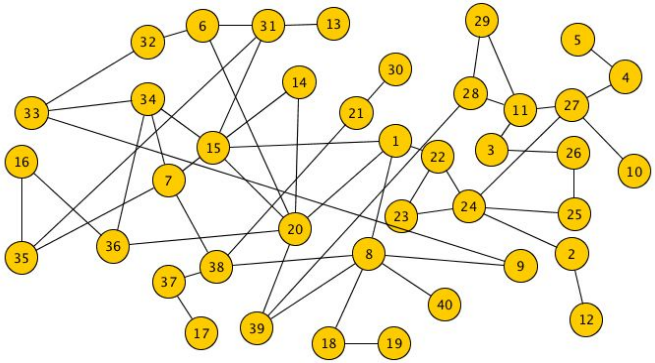
Çizge Verisi

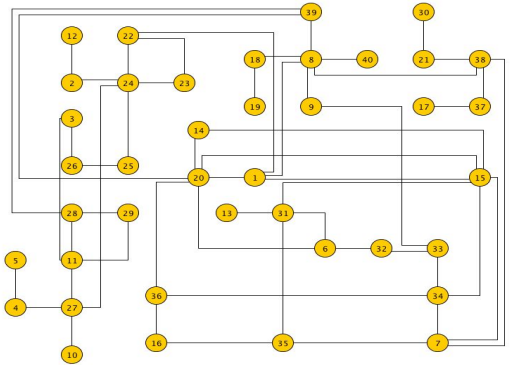
İlaç Endüstrisi - İlaç şirketleri, satıcının rotalarını çizge teorisini kullanarak optimize edebilir. Bu, maliyetleri düşürmeye ve satıcı için seyahat süresini azaltmaya yardımcı olur

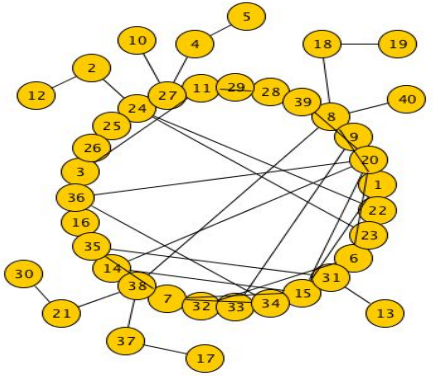
Telekom - Telekom şirketleri, maksimum kapsama sağlamak için Hücre kulelerinin optimal miktarını ve yerlerini bulmak için genellikle çizge (Voronoi diyagramları) kullanır.

Tedarik Zinciri - Çizgeler, teslimat kamyonlarınız için en uygun rotaları belirlemenize ve depolar ve teslimat merkezleri için konumları belirlemenize yardımcı olur

Çizge Verisi





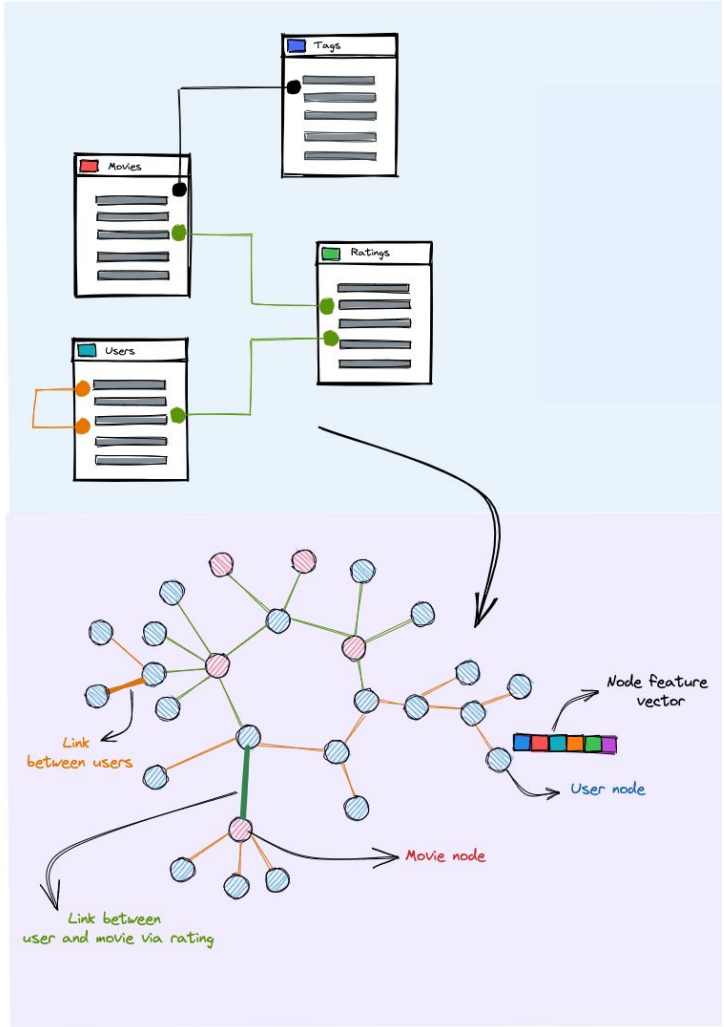


Çizge Verisi

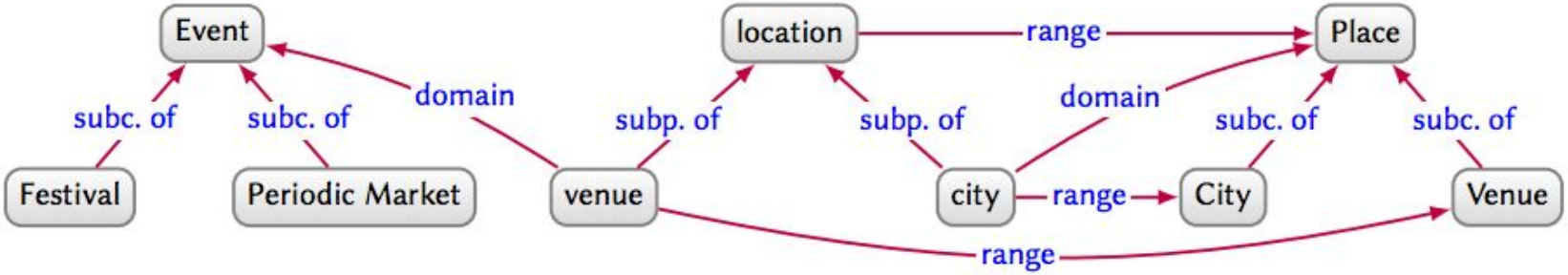
Çizge Veritabanları: Sezgisel arama sonuçları, çevrimiçi mağaza tavsiyeleri ve kişiselleştirilmiş alışveriş deneyimleri aracılığıyla toplanan veri bilgi çizgeleri olarak organize edilmektedir.

Bu çizgeler ve altta kullanılan çizge veritabanları klasik veritabanlarından farklıdır.

Bankacılık, otomobil endüstrisi, petrol, sağlık, perakende, yayıncılık, medya ve daha birçok alanda yer alan kuruluşlar, ellerindeki verilere değer katmak için bilgi çizgelerini ve çizge veritabanlarını kullanmaktadırlar.



Çizge Verisi

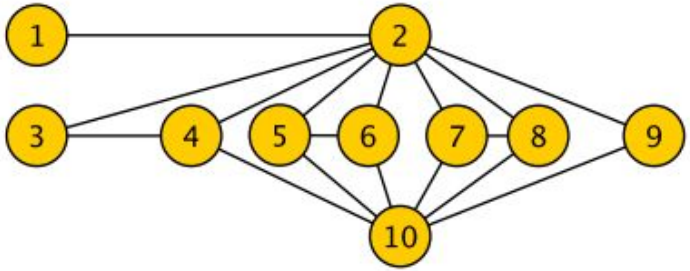


[Submitted on 4 Mar 2020 (v1), last revised 11 Dec 2020 (this version, v4)]

Knowledge Graphs

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, Antoine Zimmermann

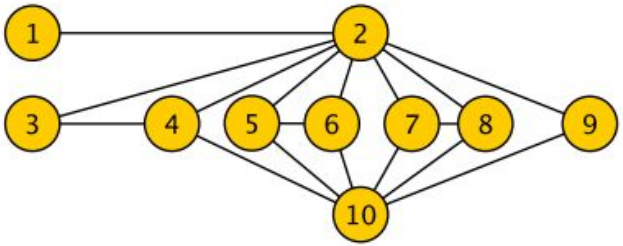
Çizge Veri Yapıları



	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										

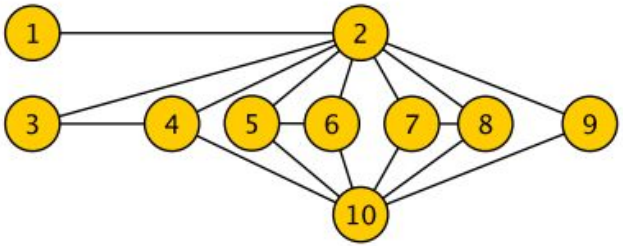
Matris veri yapısı
1M nokta için 10^{12} değer tutmak gerekiyor.
(1TB değer – 4TB hafıza)

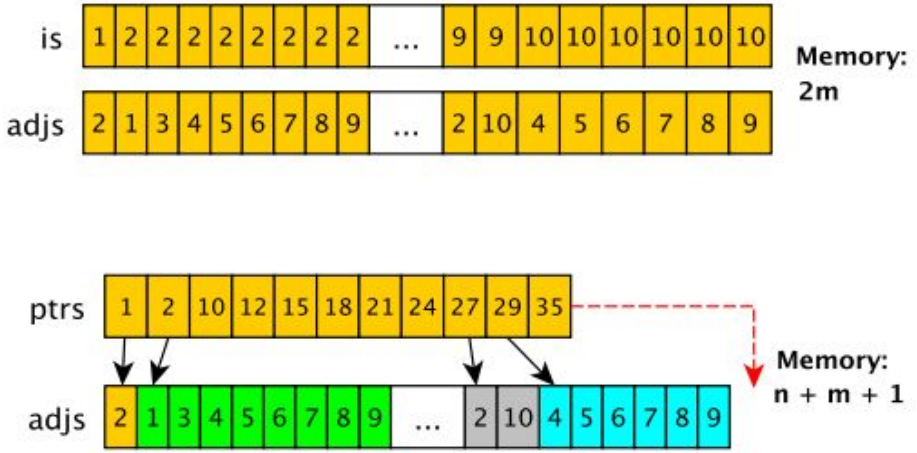
Çizge Veri Yapıları



Koordinat veri yapısı (COO)

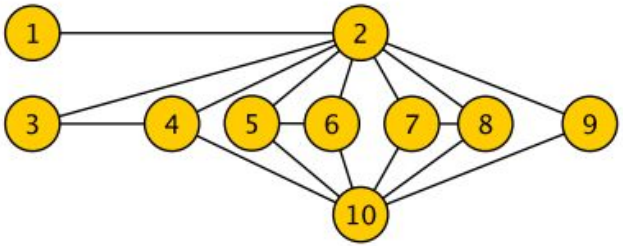
Çizge Veri Yapıları

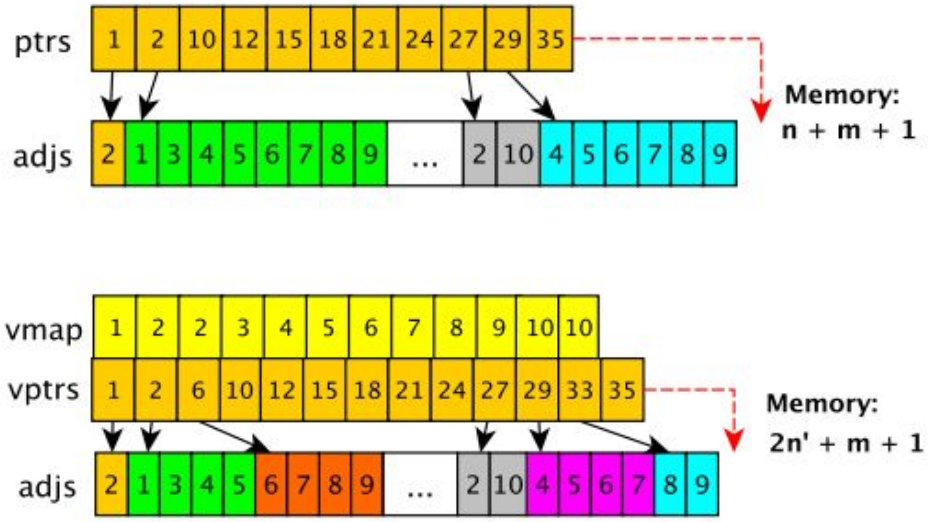




Sıkıştırılmış çizge veri yapısı (CCS ya da CRS)

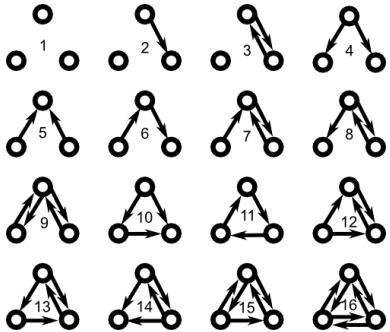
Çizge Veri Yapıları

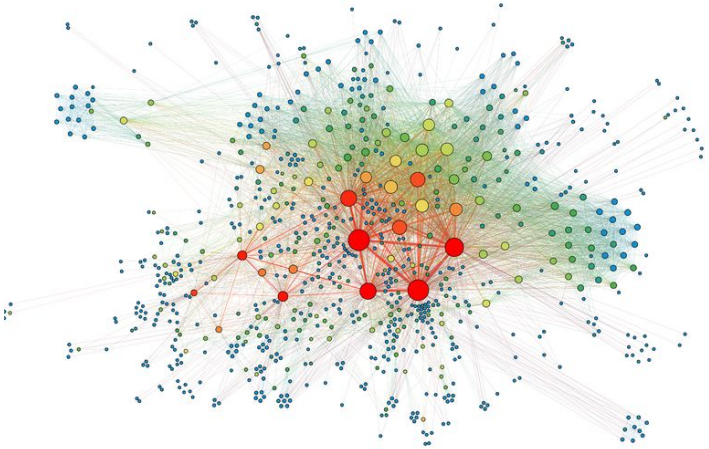


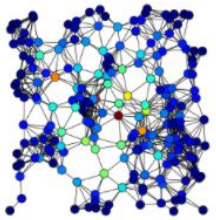


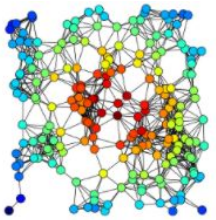
Sanal noktalar ile CRS

Çizge Analizi



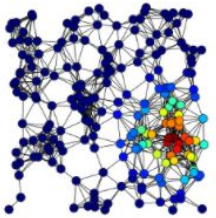


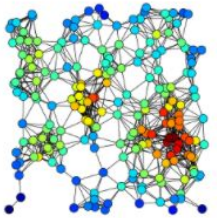




A

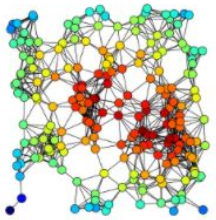
B

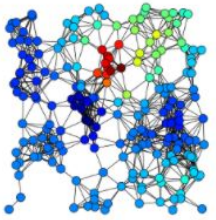




C

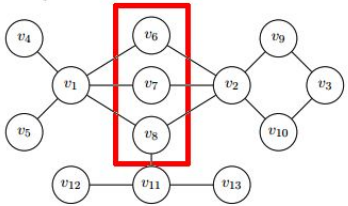
D





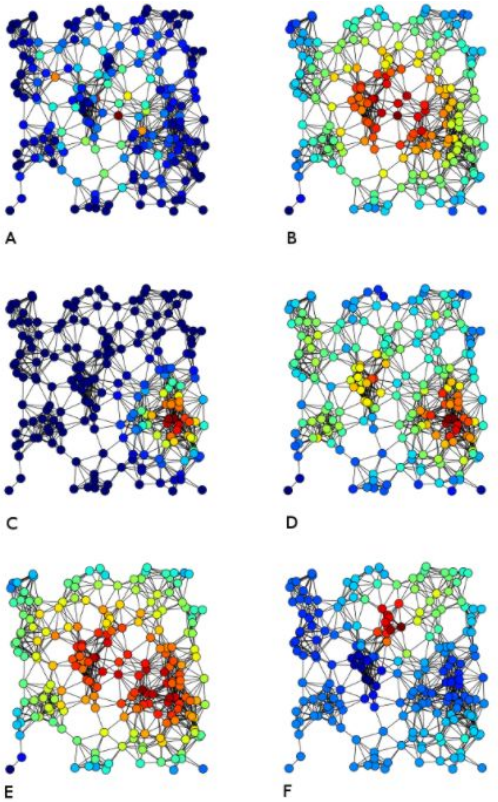
E

F



Çizge Analizi: Algoritmalar

1. **Çizge gezme**: BFS, DFS, ...
2. **En kısa yol, en yakın komşu, maksimum en geniş yol**: Dijkstra, Edmonds-Karp, Ford-Fulkerson, Dinic, ...
3. **Kapsayan ağaçlar**: Prim, ...
4. **Kümeleme, parçalama**: k-means, Louvain, modularite, ...

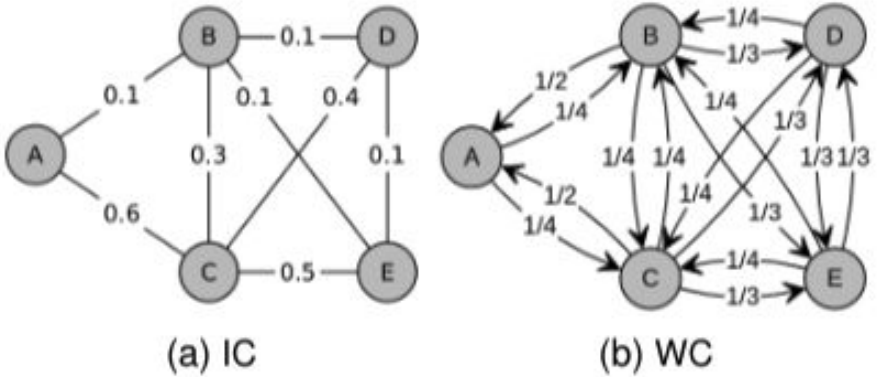


Çizge Analizi: Etki Eniyilemesi

Etki Eniyilemesi (Influence Maximization)

- Verilen bir çizge için öyle **K** node bulunsun ki, toplamda etkileyebileceğimiz nokta sayısı en fazla olsun.
- Monte Carlo simülasyonları, literatürdeki en iyi etki eniyileme algoritmalarıdır.
- Fakat bu yöntemler oldukça pahalıdır.
- Büyük ölçekli çizgeler için bu **K** noktanın bulunması saatler hatta günler sürebilmektedir.

Çizge Analizi: Etki Eniyilemesi



(a) IC (b) WC

Çizge Analizi: Etki Eniyilemesi

Algorithm 1 NEWGREEDY$(G, K, \mathcal{R})$

Input: $G = (V, E)$: the influence graph
K: number of seed vertices
$\mathcal{R}$: number of MC simulations per seed vertex
Output: S: a seed set that maximizes influence on G
mg: marginal influence scores

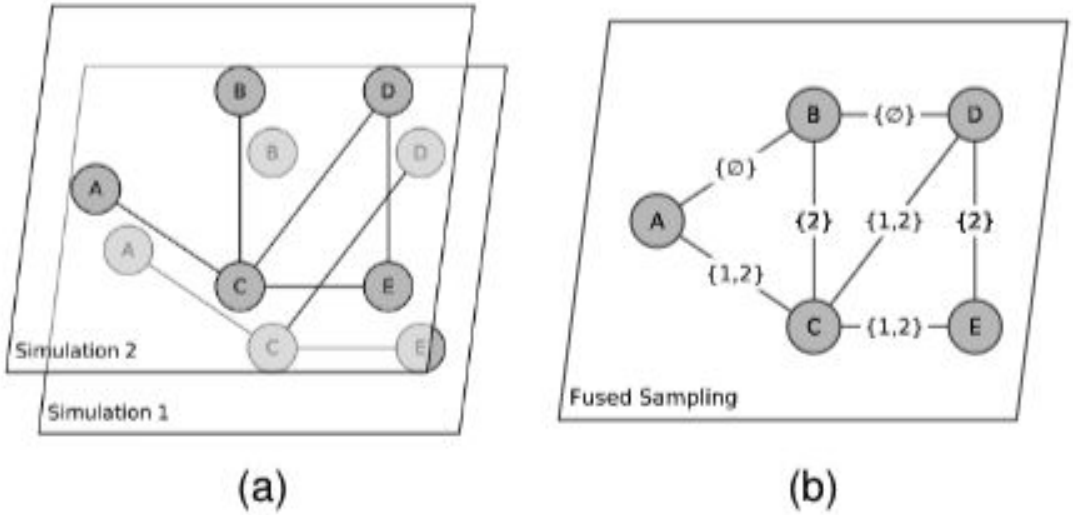
$S \leftarrow \emptyset$
for $k = 1 \dots K$ **do**
  for $v \in V$ **do**
    $mg_v \leftarrow 0$
  for $r = 1 \dots \mathcal{R}$ **do**
    $G' = (V, E') \leftarrow$ SAMPLE(G)
    Compute $R_{G'}(S)$
    Compute $|R_{G'}(\{v\})|$ for all $v \in V$
    for $v \in V \setminus S$ **do**
      if $v \notin R_{G'}(S)$ **then**
        $\sigma_{G'}(S, v) \leftarrow |R_{G'}(v)|$
        $mg_v \leftarrow mg_v + \sigma_{G'}(S, v)$
  $mg_v \leftarrow \frac{mg_v}{R}$ for all $v \in V \setminus S$
  $S \leftarrow S \cup \{\texttt{argmax}_{v \in V}\{mg_v\}\}$
return S, mg

Algorithm 2 SAMPLE(G)

Input: $G = (V, E)$: the original graph
Output: $G' = (V, E')$: a subgraph of G

$E' \leftarrow \emptyset$
for each $\{u, v\}$ in E **do**
  Randomly choose $r \in_R [0, 1]$ from a uniform dist.
  if $r \leq w_{u,v}$ **then**
    $E' \leftarrow E' \cup \{u, v\}$
Construct $G' = (V, E')$
return G'

Çizge Analizi: Etki Eniyilemesi



Çizge Analizi: Etki Eniyilemesi

IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 32, NO. 5, MAY 2021 1001

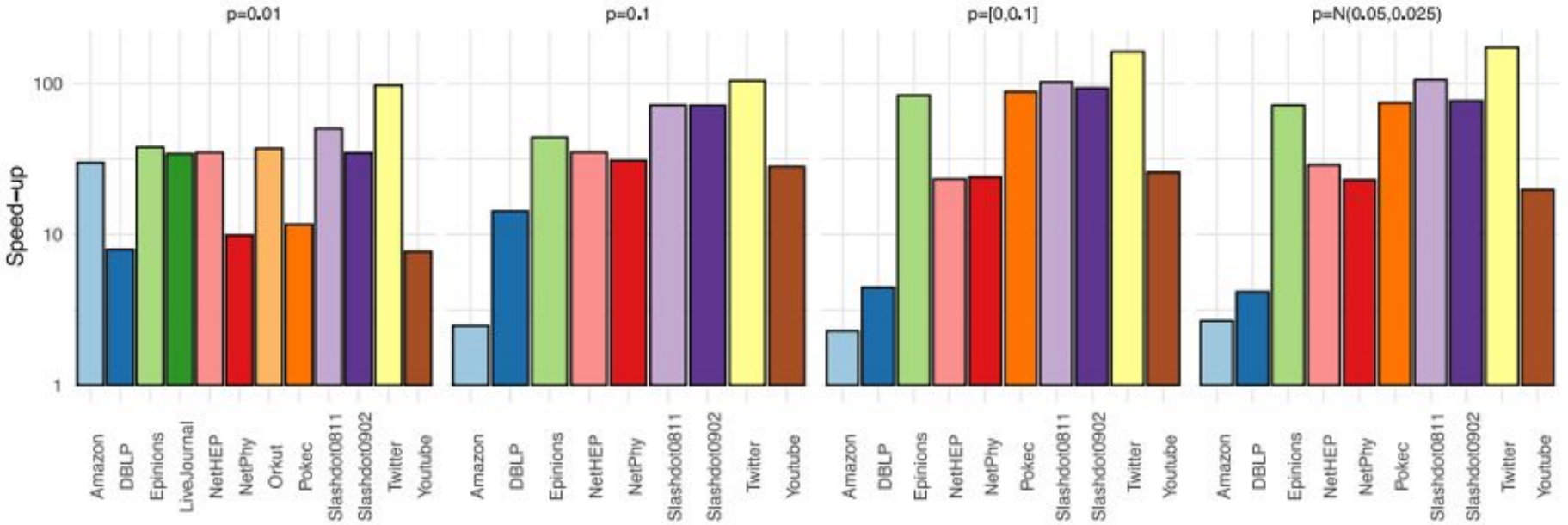
Boosting Parallel Influence-Maximization Kernels for Undirected Networks With Fusing and Vectorization

Gökhan Göktürk and Kamer Kaya

	Dataset	No. of Vertices	No. of Edges
Undirected	Amazon	262,113	1,234,878
	DBLP	317,081	1,049,867
	NetHEP	15,235	58,892
	NetPhy	37,151	231,508
	Orkut	3,072,441	117,185,083
	Youtube	1,134,891	2,987,625
Directed	Epinions	75,880	508,838
	LiveJournal	4,847,571	68,993,773
	Pokec	1,632,803	30,622,564
	Slashdot0811	77,360	905,468
	Slashdot0902	82,168	948,464
	Twitter	81,306	2,420,766

	$p = 0.01$			$p = 0.1$		
Dataset	IMM ($\epsilon = 0.13$)	IMM ($\epsilon = 0.5$)	INFUSER MG	IMM ($\epsilon = 0.13$)	IMM ($\epsilon = 0.5$)	INFUSER MG
Amazon	62.67	4.95	**2.09**	24.80	**2.72**	9.99
DBLP	55.92	**4.02**	7.02	168.68	15.34	**11.83**
Epinions	72.39	7.55	**1.91**	86.10	7.82	**1.96**
LiveJournal	9078.34	860.38	**265.84**	-	1527.58	**153.46**
NetHEP	2.80	0.29	**0.08**	6.31	0.65	**0.18**
NetPhy	3.55	0.39	**0.36**	22.57	2.06	**0.73**
Slashdot0811	135.54	12.33	**2.69**	146.09	14.48	**2.04**
Slashdot0902	107.83	10.63	**3.11**	129.15	13.29	**1.81**
Orkut	24300.59	2279.10	**654.52**	-	1987.11	**195.60**
Pokec	2646.98	247.36	**227.24**	-	611.36	**74.38**
Twitter	298.97	26.70	**3.07**	261.94	23.70	**2.52**
Youtube	201.65	**19.42**	26.18	740.35	78.51	**26.31**

Çizge Analizi: Etki Eniyilemesi



Çizge Analizi: Etki Eniyilemesi

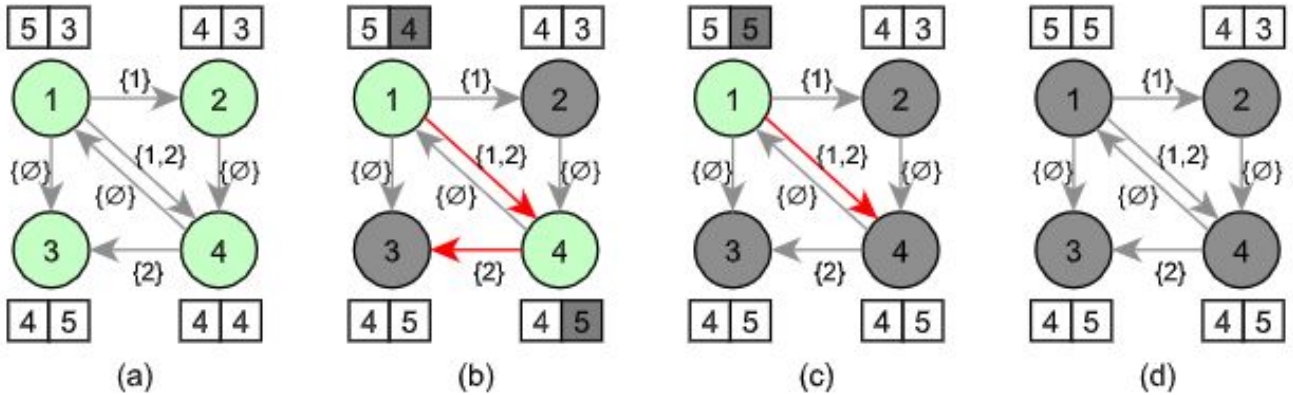
Fast and Error-Adaptive Influence Maximization based on Count-Distinct Sketches

https://github.com/ggokturk/infuser

Gökhan Göktürk and Kamer Kaya

$$M[j] = \max(M[j], clz(h_j(x)),\ 1 \leq j \leq \mathcal{J}$$

$$M_{uv}[j] = \max(M_u[j], M_v[j]),\ 1 \leq j \leq \mathcal{J}.$$



Çizge Analizi: Etki Eniyilemesi

Fast and Error-Adaptive Influence Maximization based on Count-Distinct Sketches

https://github.com/ggokturk/infuser

Gökhan Göktürk and Kamer Kaya

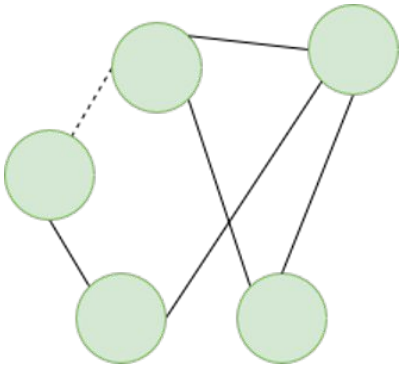
Method / Dataset	Time				Score				Memory			
	HYPER FUSER	TIM+	IMM	SKIM	HYPER FUSER	TIM+	IMM	SKIM	HYPER FUSER	TIM+	IMM	SKIM
Amazon	0.69	133.22	1.98	23.62	11797.0	1.006×	0.990×	0.815×	0.17	5.49	0.23	2.59
DBLP	0.54	1368.83	14.54	6.30	48549.9	1.001×	0.995×	1.001×	0.27	35.19	1.06	0.65
Epinions	0.26	439.33	5.46	9.42	18409.9	1.000×	0.998×	0.997×	0.06	12.17	0.39	1.18
LiveJournal	10.10	-	1071.30	65.73	2134726.0	-	1.000×	1.000×	3.97	-	65.49	1.40
NetHEP	0.10	14.18	0.33	0.61	2461.7	1.002×	0.975×	0.899×	0.01	1.02	0.04	0.10
NetPhy	0.16	107.52	1.53	0.34	8339.5	1.007×	0.994×	0.975×	0.03	3.84	0.13	0.03
Orkut	16.55	-	1964.83	446.92	2692366.5	-	1.000×	1.000×	5.19	-	71.94	9.68
Pokec	4.90	-	514.79	31.80	1034859.8	-	1.000×	1.000×	1.57	-	26.46	0.98
Slashdot0811	0.21	677.49	7.34	2.44	25871.8	1.000×	1.000×	0.999×	0.06	19.10	0.59	0.25
Slashdot0902	0.23	695.12	7.99	2.35	27519.5	1.000×	1.000×	0.999×	0.06	18.45	0.66	0.24
Twitter	0.33	1897.50	16.09	1.62	55327.3	1.000×	0.998×	0.998×	0.09	34.56	1.04	0.05
Youtube	1.12	7158.56	60.59	30.57	171392.9	1.000×	0.999×	1.001×	0.73	139.12	4.19	2.88
Norm. arit. mean		2624.61×	47.17×	15.37×		1.002×	0.996×	0.974×		199.54×	8.79×	5.32×
Norm. geo. mean	all	1449.41×	27.87×	11.06×	all	1.002×	0.996×	0.972×	all	163.77×	7.15×	2.63×
Norm. max perf	1.00×	6391.57×	118.72×	36.23×	1.000×	1.007×	1.000×	1.001×	1.00×	384.00×	16.85×	19.67×
Norm. min perf		141.80×	2.87×	2.13×		1.000×	0.975×	0.815×		32.29×	1.35×	0.35×

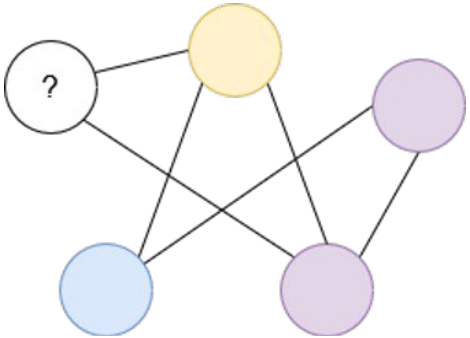
Makina Öğrenmesi

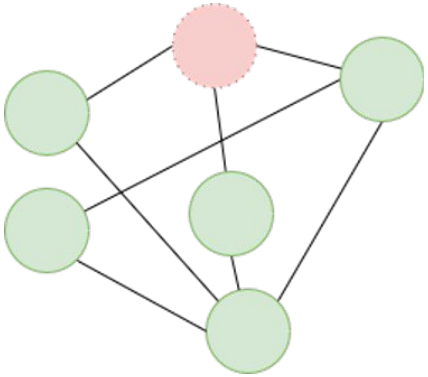
Bir bilgisayara programlama yapmadan programlaması zor olan bir işi öğretmek.

- El yazısı tanıma
  - yazılı harfleri dijital harflere dönüştürmek
- Çevirisi
  - sözlü ve/veya yazılı dilleri çevirmek (ör. Google Translate)
- Konuşma tanıma
  - ses parçacıklarını metne dönüştürme (ör. Siri, Cortana, ve Alexa)
- Görüntü Sınıflandırma
  - resimleri uygun kategorilerle etiketleme (ör. Google Fotoğraflar)
- Otonom Sürüş
  - arabaların kaza yapmadan bir noktadan bir noktaya varması

Çizge Üzerinde Makina Öğrenmesi





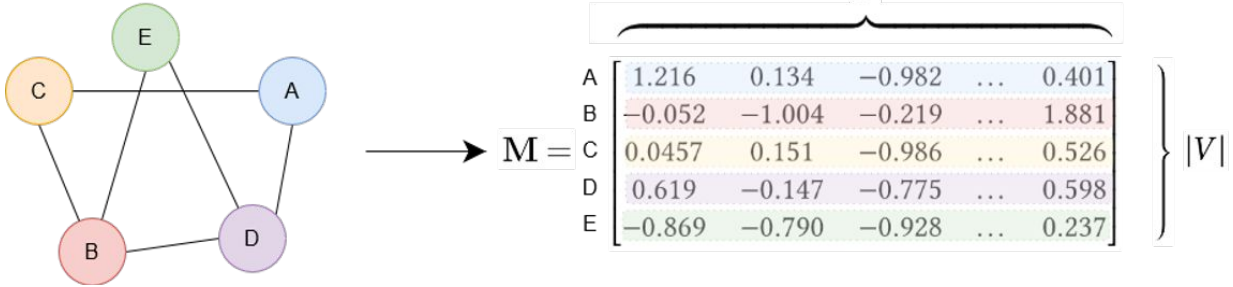


Çizge Üzerinde Makina Öğrenmesi

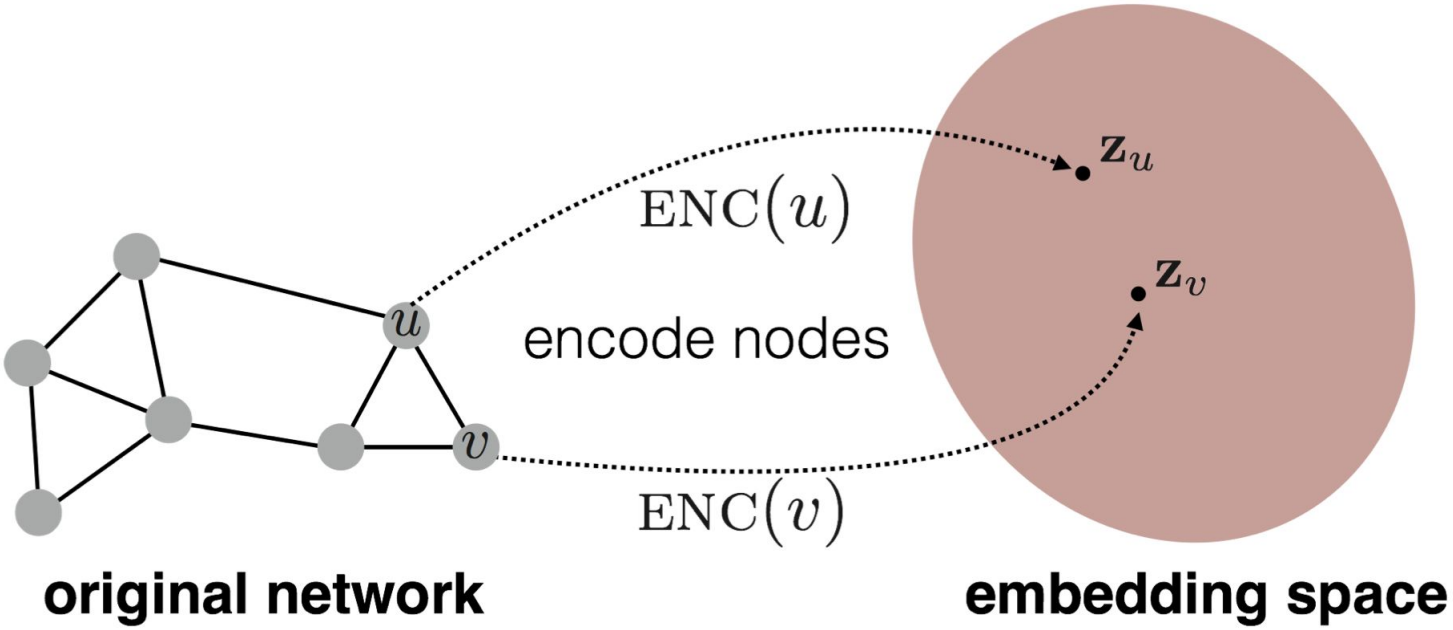
Çizgeler **düzensiz** ve **seyrektir**.

Örneğin bir sosyal ağda insanlar arasındaki ortalama bağlantı sayısı, ağdaki toplam insan sayısından çok daha küçüktür. Ayrıca insanlar arasındaki bağlantıları genelleştirebilecek kesin bir kalıp yoktur.

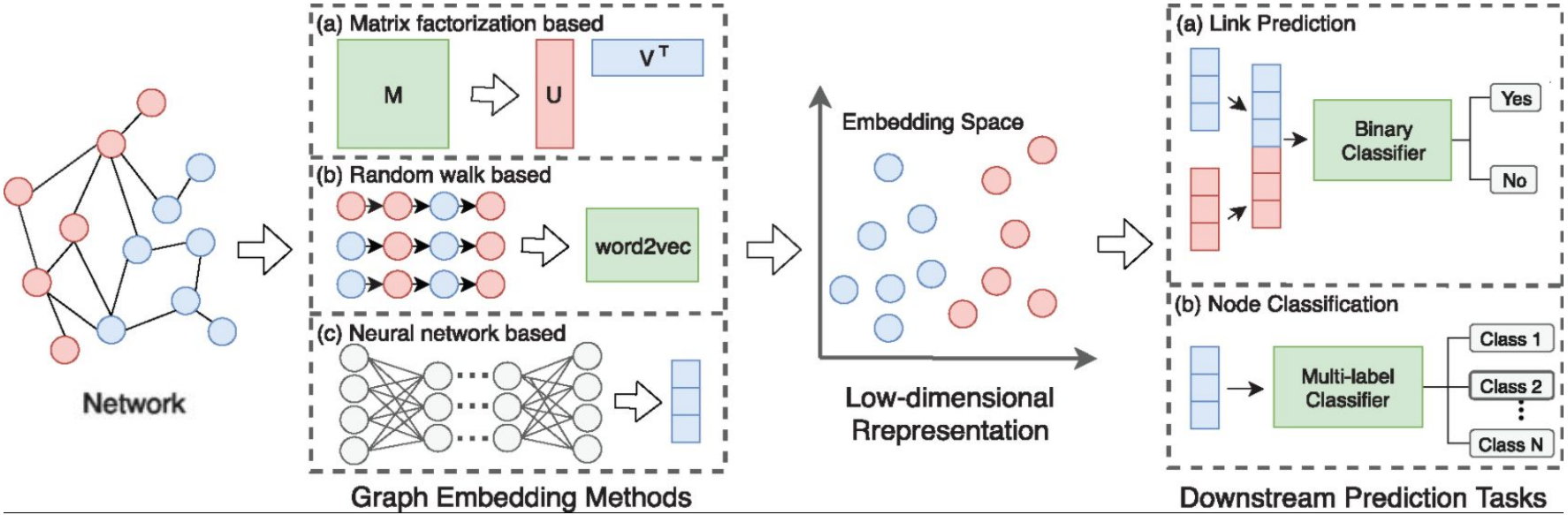
Çizge yerleştirme basitçe bir **M** matrisi oluşturma işlemidir. Burada satır sayısı **|V|** ve sütun sayısı **d**'dir (yerleştirme boyutu). **M[v]** vektörü (satırı), **v** noktasının yerleşimini temsil eder.



Çizge Üzerinde Makina Öğrenmesi



Çizge Analizi - Çizge Yerleştirme



Xiang Yue, Zhen Wang, Jingong Huang, Srinivasan Parthasarathy, Soheil Moosavinasab, Yungui Huang, Simon M Lin, Wen Zhang, Ping Zhang, Huan Sun, Graph embedding on biomedical networks: methods, applications and evaluations, *Bioinformatics*, Volume 36, Issue 4, 15 February 2020, Pages 1241–1251,

Çizge Analizi - Çizge Yerleştirme

- Çizge yerleştirme üzerine ilk çalışmalar 2000'lerin başında yapıldı, ancak genelde bu çalışmalar ölçeklenmeyen bir yöntem olan matris ayrıştırma yöntemini kullandı.
- Son yıllarda birçok araştırma alanında olduğu gibi (derin) sinir ağları temelli yaklaşımlar popüler hale gelmiştir. Bu yaklaşımlar etkili fakat hesaplama açısından pahalıdır.
- Literatürde sinir ağları kullanmayan ama yine onlar kadar etkili **DeepWalk**, **Line**, **Node2Vec**, ve **Verse** gibi yöntemler de vardır.

Çizge Analizi - Çizge Yerleştirme

- **DeepWalk** (**Online Learning of Social Representations**) temel olarak rastgele yürüyüş ve güncelleme prosedürü olmak üzere iki bölümden oluşur. Rastgele yürüyüşlerle benzerlikleri ve topluluk üyeliklerini tahmin eder.
- Rastgele yürüyüş üreteci ilk önce bir nokta örneği alır. Ardından, (bir adımı temsil eden) komşu noktalardan biri seçilir. Bu işlem **t** kez tekrarlanır.
- Ziyaret edilen noktaların tüm eşdizimleri, pozitif örneklerle sonuçlanan hesaplanır. Daha sonra güncellemeler *SkipGram* algoritması kullanılarak gerçekleştirilir.
- DeepWalk, rastgele yürüyüşleri kullanmayan yöntemlerden daha iyi performans gösterir.

Çizge Analizi - Çizge Yerleştirme

Algorithm 1 DeepWalk(G, w, d, γ, t)

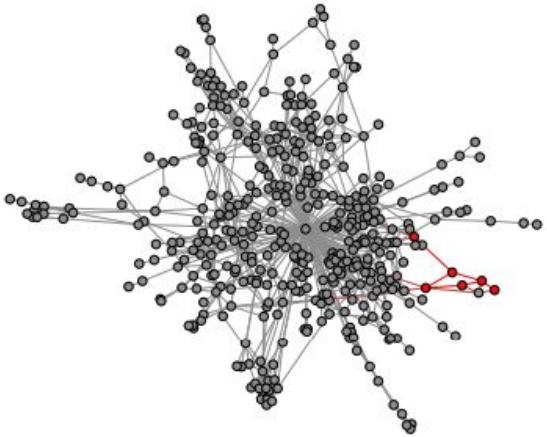
Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, \text{t})$
    SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
  end for
end for

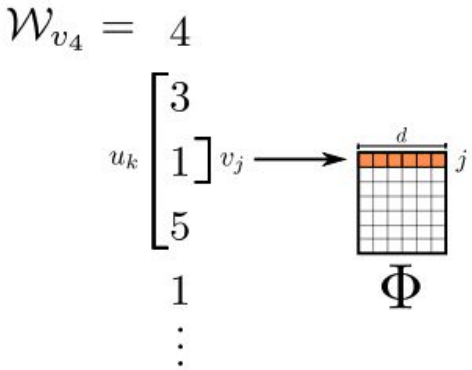
Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

for each $v_j \in \mathcal{W}_{v_i}$ **do**
  for each $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
    $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
    $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
  end for
end for

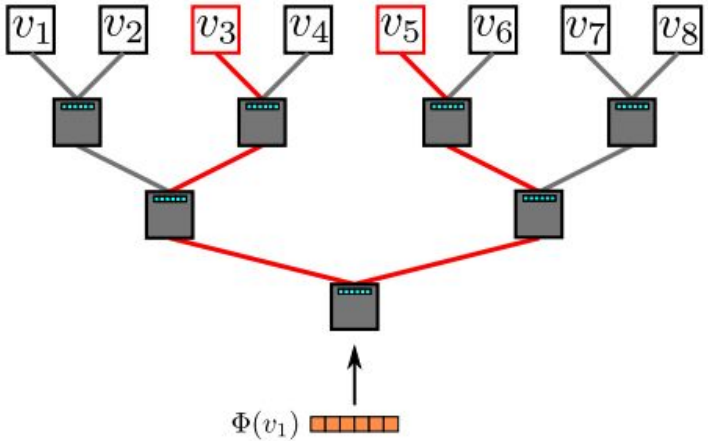
Çizge Analizi - Çizge Yerleştirme



(a) Random walk generation.



(b) Representation mapping.



(c) Hierarchical Softmax.

Çizge Analizi - Çizge Yerleştirme

```
inline void update( // update the embedding, putting w_t gradient in w_t_cache
    float *w_s, float *w_t, float *w_t_cache, float lr, const int label) {
  float score = 0; // score = dot(w_s, w_t)
  AVX_LOOP
  for (int c = 0; c < n_hidden; c++)
    score += w_s[c] * w_t[c];
  score = (label - fast_sigmoid(score)) * lr;
  AVX_LOOP
  for (int c = 0; c < n_hidden; c++)
    w_t_cache[c] += score * w_s[c]; // w_t gradient
  AVX_LOOP
  for (int c = 0; c < n_hidden; c++)
    w_s[c] += score * w_t[c]; // w_s gradient
}
```

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}}$$

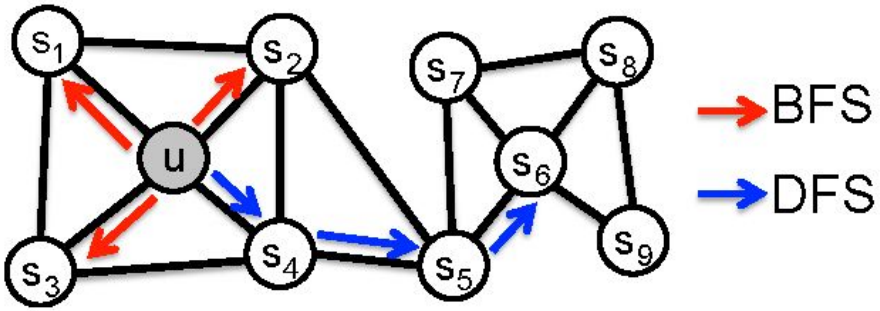
$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj}$$

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2$$

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2 e_{ij} q_{kj}$$
$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2 e_{ij} p_{ik}$$

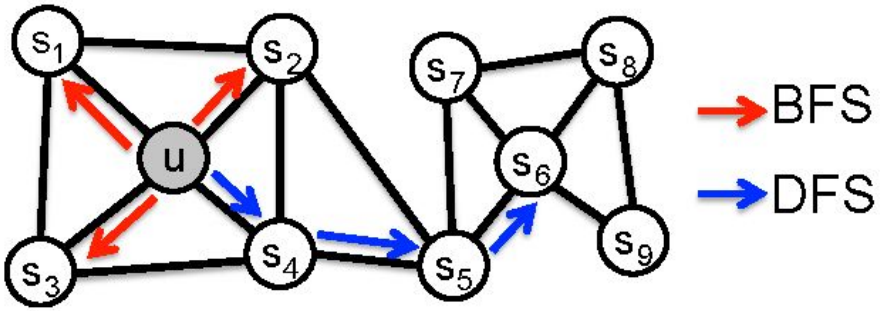
Çizge Analizi - Çizge Yerleştirme

- **Node2Vec** benzer bir çizge yerleştirme algoritmasıdır.
- Node2vec ve DeepWalk arasındaki fark ince ama önemlidir. *node2vec*, **p** ve **q** ile parametrelenen bir sapma değişkeni **α**'ya sahiptir. **p** parametresi bir *genişlik ilk arama* (BFS) prosedürüne öncelik verirken, **q** parametresi bir *derinlik ilk arama* (DFS) prosedürüne öncelik verir.
- Bundan sonra nereye yürüyeceğine dair karar 1/**p** veya 1/**q** olasılıklarından etkilenir.



Çizge Analizi - Çizge Yerleştirme

- BFS yerel komşuları öğrenmek için idealdir, DFS ise global değişkenleri öğrenmek için daha iyidir. *node2vec*, göreve bağlı olarak iki öncelik arasında geçiş yapabilir.
- Bu, tek bir çizge verildiğinde, *node2vec* parametrelerin değerlerine bağlı olarak farklı sonuçlar döndürebileceği anlamına gelir.
- DeepWalk'a göre *node2vec*, yürüyüşlerin gizli yerleşmesini de alır ve bunları sınıflandırma için bir sinir ağına girdi olarak kullanır.



Çizge Analizi - Çizge Yerleştirme

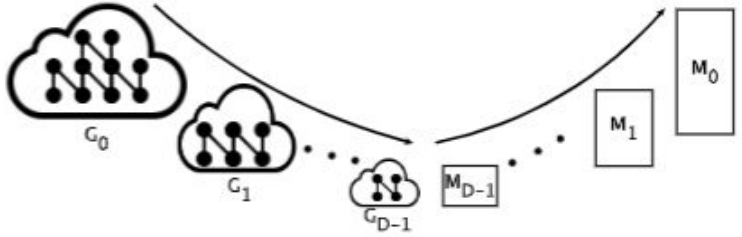
- **LINE (Large-scale Information Network Embedding)** noktalar arasındaki benzerliği ölçmek için iki yeni kavram sunar; birinci ve ikinci dereceden yakınlık.
    - **Birinci dereceden yakınlık**: bir kenarı paylaşıyorlarsa iki nokta benzer olarak kabul edilir
    - **İkinci dereceden yakınlık**: birçok komşu/bitişik nokta paylaşıyorlarsa iki nokta benzer kabul edilir.

Çizge Analizi - Çizge Yerleştirme

- LINE'ın yeni bir güncelleme süreci vardır. DeepWalk gibi bir başlangıç noktası rastgele seçilir ve ağırlıklarına göre komşu bir kenar seçilir.
- LINE için orijinal yerleştirme ve bağlam yerleştirme olmak üzere iki farklı yerleştirme vardır. Kaynak noktalardaki güncellemeler orijinal matrise uygulanır ve bağlam noktalarındaki güncellemeler bağlam matrisine uygulanır.
- Son olarak, iki yerleştirme matrisi birleştirilir. Orijinal ve bağlam yerleştirmeleri, sırasıyla birinci ve ikinci dereceden yakınlığı temsil eder.

GOSH: Embedding Big Graphs on Small Hardware

- **Tel bir GPU üzerinde hızlı bir şekilde çizge yerleştirme işlemini gerçekleştiren bir araç.**
  - Bağlantı tahmini ve nokta sınıflandırma doğruluğu literatür ile aynı.
  - Paralel bir çizge indirgeme ile sürecin hızlandırılması.



Çizge	GOSH	Hızlanma
Hyperlink (40m nokta, 600m kenar)	**0.2 saat** (97% AUC) bir TITAN X SotA: **5.4 saat** (4 Tesla P100)	26.7x
Wiki-topcats (1.7m nokta, 28m kenar)	**11 saniye** (98% AUC) bir TITAN X SotA: **310 saniye** (bir TITAN X)	27.4x

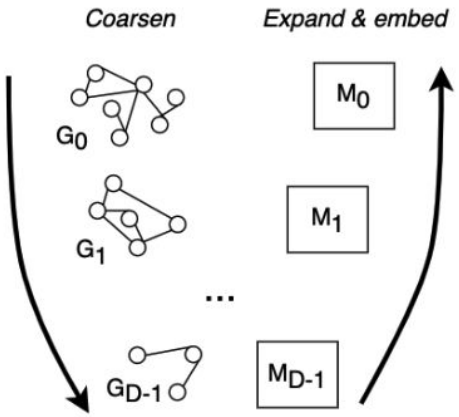
GOSH: Genel Akış

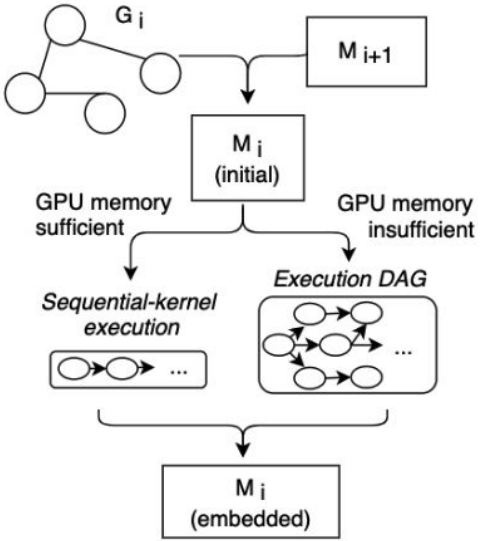
Algoritma, girdi çizgeyi bazı sonlanma kriterleri karşılanana kadar daha küçük çizgelere dönüştürerek başlar.

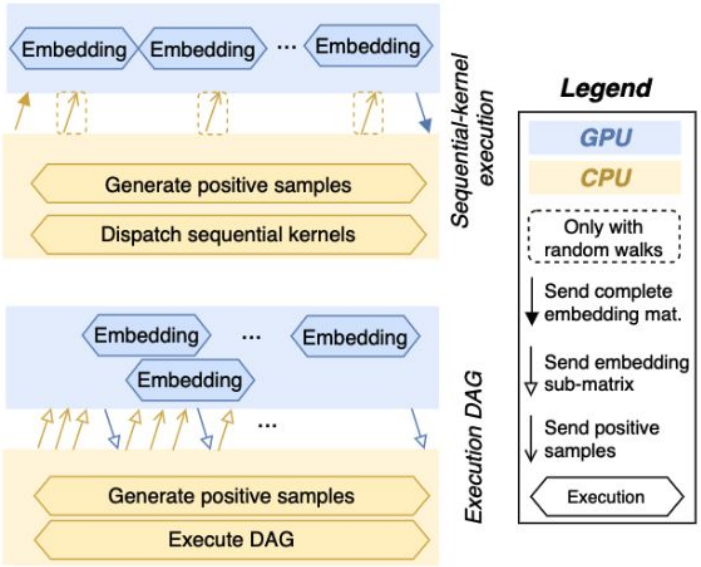
Bu işlem bittiğinde en küçük çizge yerleştirilir ve bir üstteki çizgenin her noktasının yerleştirilmesi bir süper noktanın yerleştirmesine göre ayarlanarak elde edilecektir.

Girdi çizge yerleştirilene kadar bu işlem devam eder.

GOSH: Genel Akış

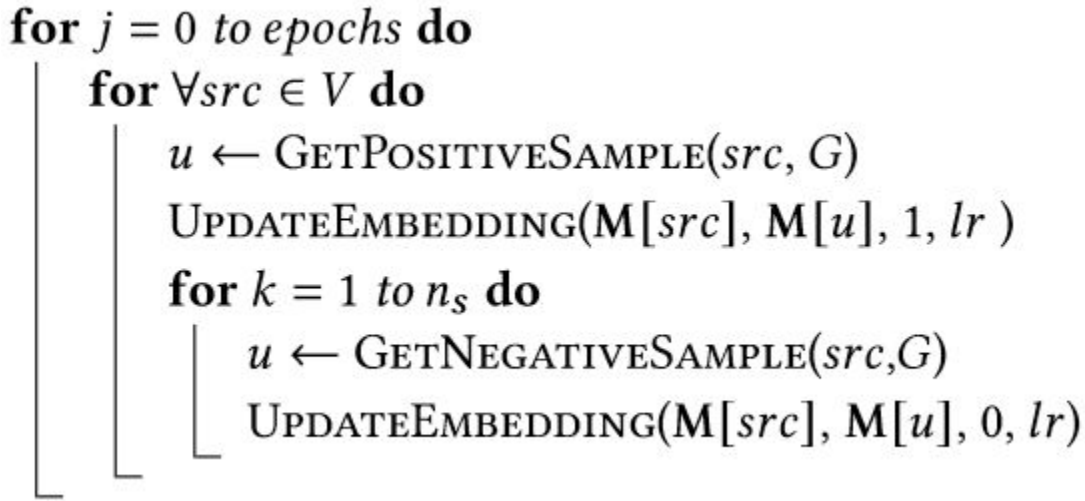






GOSH: Yerleştirme süreci

Yerleştirme işlemi NCE'ye (Noise Contrastive Estimation) dayanır ve SGD (Stochastic Gradient Descent) eniyilemesi kullanır.

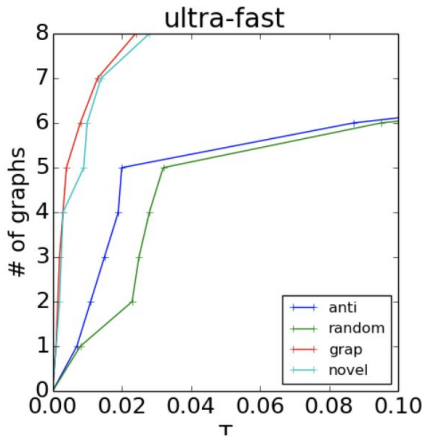


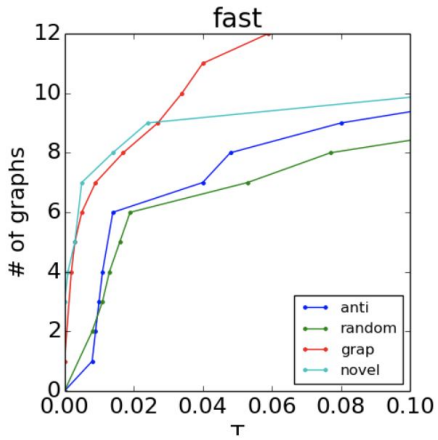
Çizge İndirgeme

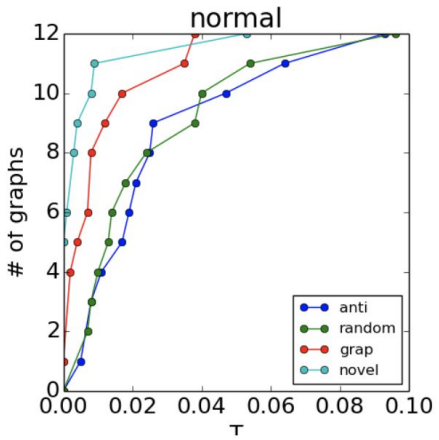
- En iyi indirgeme nedir?

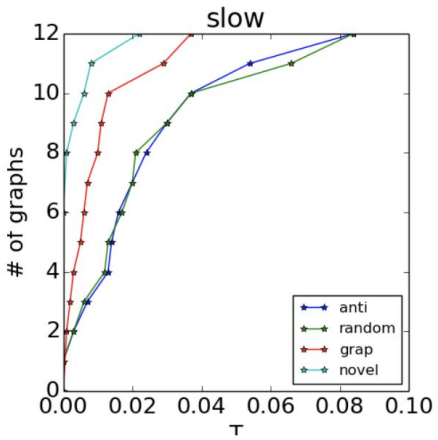
Understanding Coarsening for Embedding Large-Scale Graphs

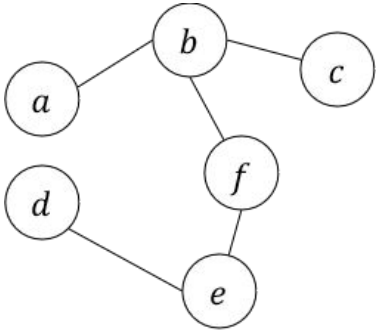
Taha Atahan Akyildiz, Amro Alabsi Aljundi and Kamer Kaya
Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul, Turkey
E-mail: {aakyildiz, amroa, kaya}@sabanciuniv.edu



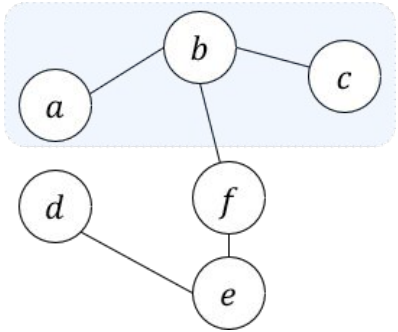




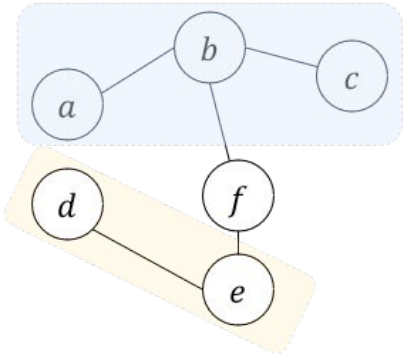




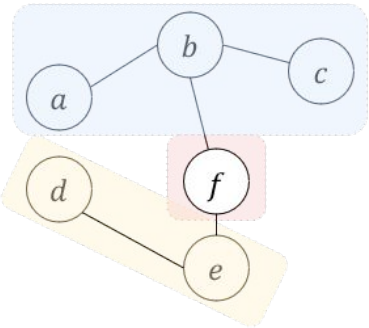
0. Initial, uncoarsened graph



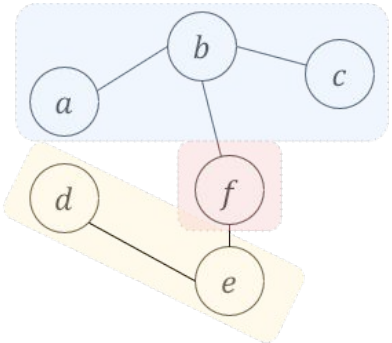
1. Mark b and match its neighbors c and a. Cannot match f due to matching rule.



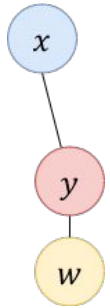
3. Mark e and match its neighbor d. Cannot match f due to matching rule.



4. Mark f. Since all its neighbor are marked, none of them are matched.



5. Try to mark a, c, and d, but they are all marked already.



6. Generate the coarsened graph and reconstruct edges

Çizge Analizi - Çizge Yerleştirme

GOSH: Embedding Big Graphs on Small Hardware

Taha Atahan Akyildiz* aakyildiz@sabanciuniv.edu
Amro Alabsi Aljundi* amroa@sabanciuniv.edu
Kamer Kaya Sabanci University

Algorithm	G	Time (s)	Speedup	AUCROC	G	Time (s)	Speedup	AUCROC
VERSE	com-dblp	128.02	1.0×	97.76	youtube	700.26	1.0×	**97.99**
MILE		122.55	1.04×	97.70		1847.18	0.38×	94.55
GRAPHVITE-fast		5.96	21.47×	95.72		22.48	31.15×	97.12
GRAPHVITE-slow		8.80	14.56×	97.48		34.96	20.19×	97.08
PYTORCH-BIGGRAPH		21.29	6.01 ×	97.73		74.42	9.41×	97.07
GOSH-fast		0.53	241.44×	97.02		1.89	370.88×	97.67
GOSH-normal		1.41	90.95×	**97.79**		6.11	114.65×	**98.00**
GOSH-slow		2.83	45.20×	**97.95**		12.64	55.39×	**98.00**
GOSH-NoCoarse		19.17	6.68×	97.13		90.46	7.74×	96.51
VERSE	com-lj	7472.89	1.0×	98.91	com-orkut	25020.28	1.0×	**98.28**
MILE		3046.88	2.45×	85.89		9347.67[a]	2.68×	90.22
GRAPHVITE-fast		146.21	51.11×	98.33		456.00	45.87×	97.75
GRAPHVITE-slow		236.00	31.66×	98.36		746.25	33.53×	97.30
PYTORCH-BIGGRAPH		497.06	15.03 ×	98.40		1102.90	22.69×	98.42
GOSH-fast		11.38	656.55×	98.28		30.52	819.68×	**98.86**
GOSH-normal		39.44	189.48×	**98.74**		211.64	211.64×	98.66
GOSH-slow		85.26	87.65×	98.72		261.57	95.66×	98.37
GOSH-NoCoarse		655.00	11.41×	96.57		2228.92	11.23×	96.88

(https://github.com/SabanciParallelComputing/GOSH)

Çizge Analizi - Çizge Yerleştirme

$$Micro - Precision = \frac{TruePositives1 + TruePositives2}{TruePositives1 + FalsePositives1 + TruePositives2 + FalsePositives2}$$

$$Micro - Recall = \frac{TruePositives1 + TruePositives2}{TruePositives1 + FalseNegatives1 + TruePositives2 + FalseNegatives2}$$

$$Micro - F - Score = 2.\frac{Micro - Precision \,.\, Micro - Recall}{Micro - Precision + Micro - Recall}$$

$$Macro - Precision = \frac{Pecision1 + Precision2}{2}$$

$$Macro - Recall = \frac{Recall1 + Recall2}{2}$$

$$Macro - F - Score = 2.\frac{Macro - Precision \,.\, Macro - Recall}{Macro - Precision + Macro - Recall}$$

(https://github.com/SabanciParallelComputing/GOSH)

GOSH: Embedding Big Graphs on Small Hardware

Taha Atahan Akyildiz*
aakyildiz@sabanciuniv.edu
Sabanci University

Amro Alabsi Aljundi*
amroa@sabanciuniv.edu
Sabanci University

Kamer Kaya
Sabanci University
kaya@sabanciuniv.edu

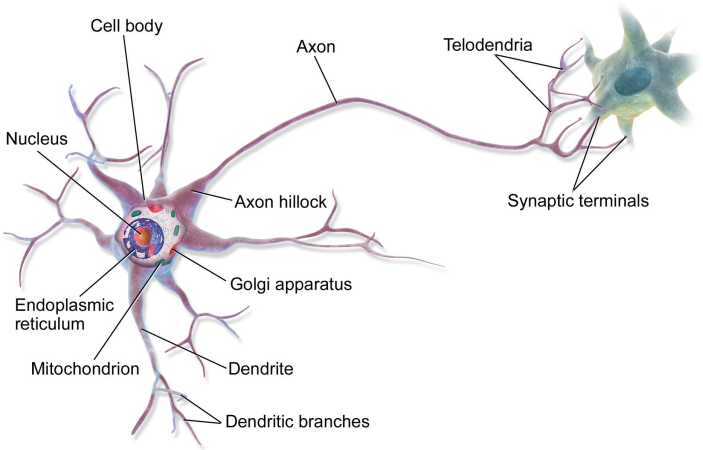
Algorithm	G	Time (s)	Micro F1	Macro F1	G	Time (s)	Micro F1	Macro1 F1
VERSE	com-dblp	166.21	46.97	43.27	com-amazon	146.34	94.94	94.63
GRAPHVITE-fast		7.13	53.95	33.83		6.39	72.97	68.91
GRAPHVITE-slow		10.56	57.05	42.90		9.43	90.16	89.03
PYTORCH-BIGGRAPH		32.79	39.38	35.19		29.63	89.02	88.51
GOSHW-fast		0.86	52.82	49.38		0.77	97.22	96.43
GOSHW-normal		1.76	60.36	58.76		1.64	**98.21**	**97.80**
GOSHW-slow		3.41	**62.27**	**61.22**		2.69	98.31	97.82
GOSHW-NoCoarse		15.62	58.59	54.10		13.86	97.44	97.34
VERSE	youtube	882.29	24.70	16.23	flickr	4135.20	40.15	37.27
GRAPHVITE-fast		26.24	34.51	24.90		110.29	48.77	46.62
GRAPHVITE-slow		41.04	35.27	25.77		177.58	48.54	46.05
PYTORCH-BIGGRAPH		134.11	34.29	23.54		597.72	47.20	44.68
GOSHW-fast		2.62	32.33	23.13		8.74	46.43	44.63
GOSHW-normal		6.86	34.70	25.30		30.12	49.57	47.88
GOSHW-slow		13.63	**35.30**	**25.79**		65.64	50.24	48.54
GOSHW-NoCoarse		73.54	35.07	25.33		342.54	**50.77**	**49.10**
VERSE	com-lj	4893.53	81.99	81.24	com-orkut	24917.93	68.03	65.24
GRAPHVITE-fast		180.71	87.25	84.64		564.42	83.98	83.26
GRAPHVITE-slow		290.49	**87.68**	**85.59**		926.09	**84.38**	**83.74**
PYTORCH-BIGGRAPH		935.18	**87.94**	**86.54**		3111.03	**82.79**	**81.96**
GOSHW-fast		17.68	84.68	83.34		39.49	77.23	75.91
GOSHW-normal		60.08	85.60	84.26		155.55	81.63	80.75
GOSHW-slow		127.53	86.36	85.04		342.62	82.78	82.02
GOSHW-NoCoarse		541.43	86.03	84.63		1836.23	83.20	82.39

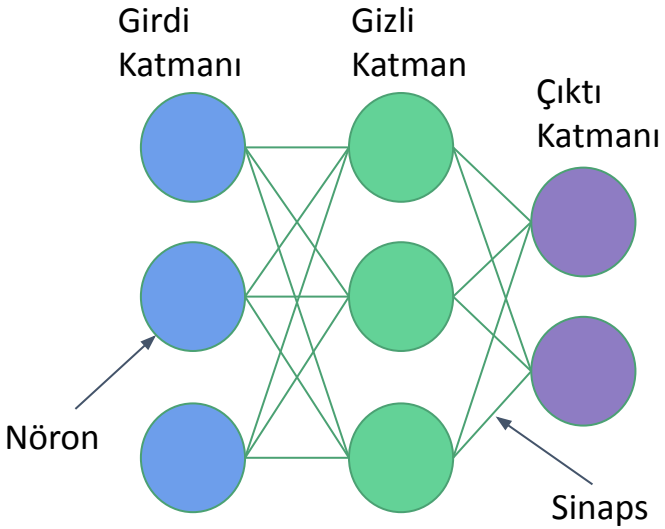
PyTorch

Örnek bir koda bakalım
(1. defter)

Makina Öğrenmesi: Yapay Sinir Ağları

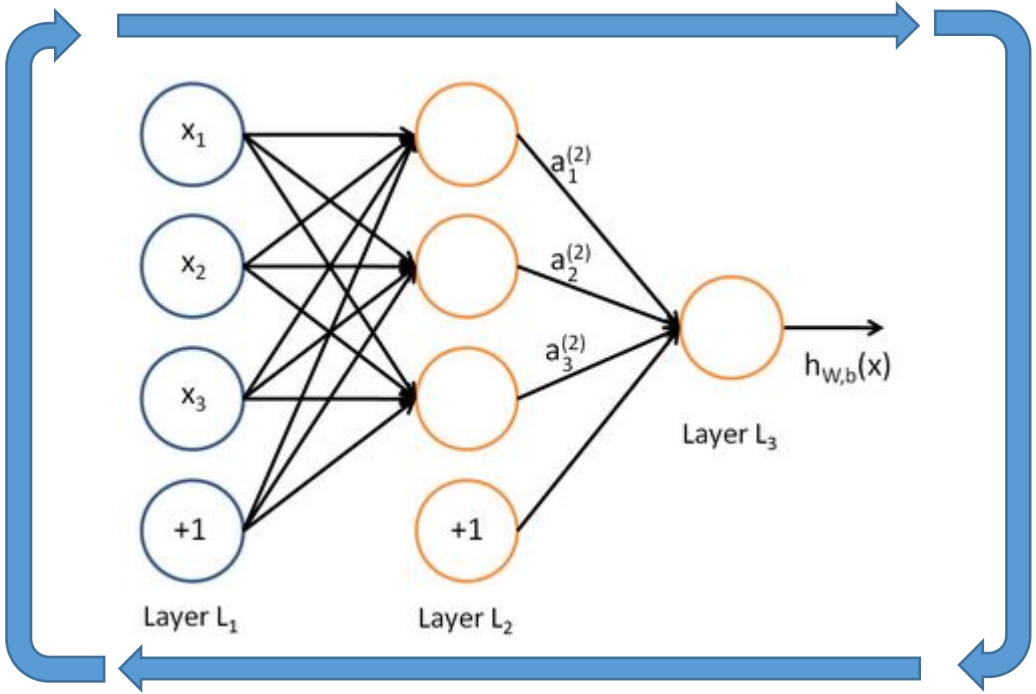
Bir bilgisayara programlama yapmadan programlaması zor olan bir işi öğretmek.





Makina Öğrenmesi: Eğitim

1. Bir girdi kümesi ile ileri besleme.



2. Hata hesabı:

Bir $h(x)$ fonksiyonu için

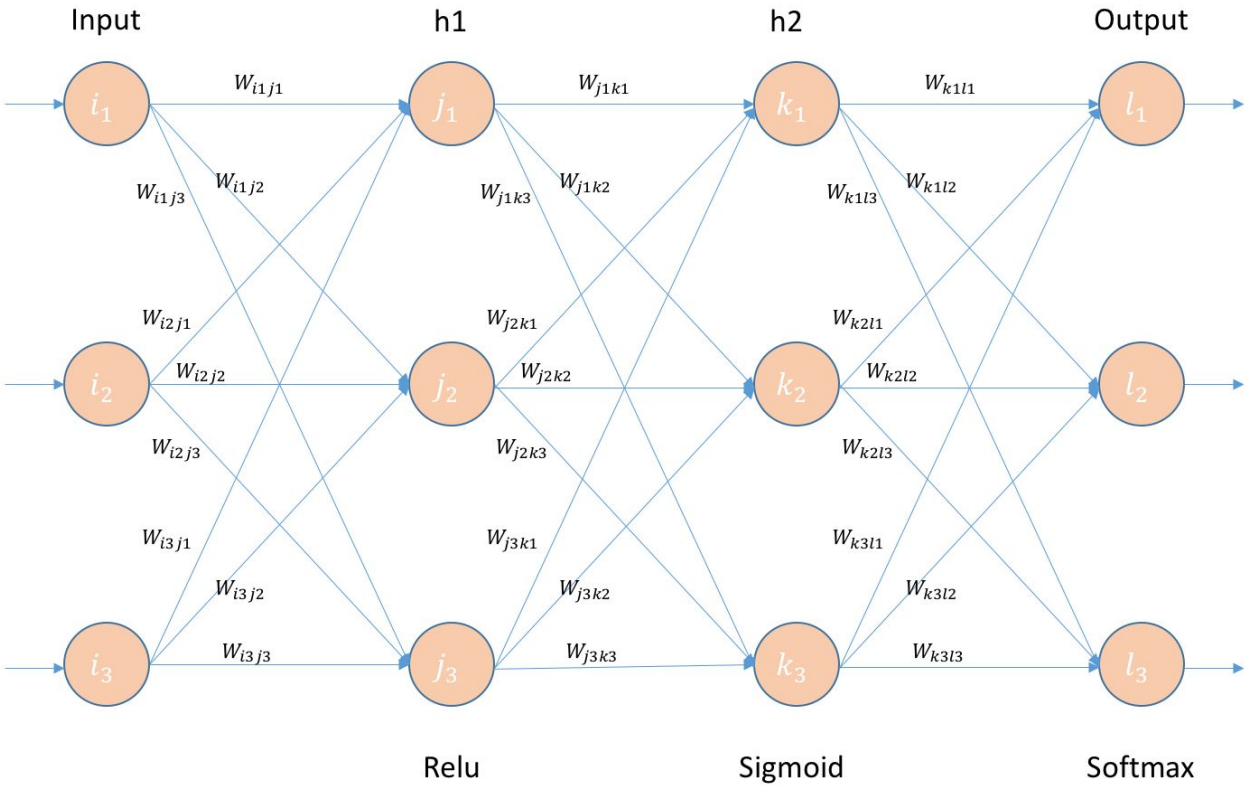
Mean Squared Loss

$$J(\theta) = \frac{1}{2n}\sum_{n}(y - h(x))^2$$

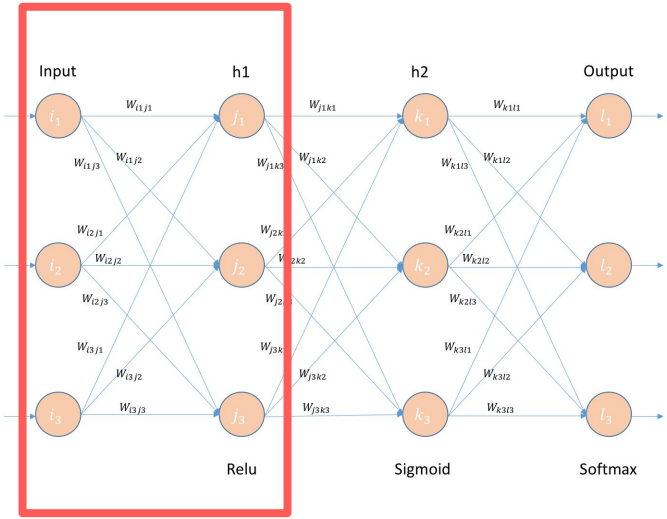
3. Geri besleme: Kısmi türevler $\frac{\Delta J(\theta)}{\Delta a^N} = \left(\frac{\delta J(\theta)}{\delta a^1}, \frac{\delta J(\theta)}{\delta a^2}\right)$

4. Başka bir girdi kümesi.

Makina Öğrenmesi: Eğitim



Makina Öğrenmesi: Eğitim



Matrix Operation:

$$\begin{bmatrix} i_1 & i_2 & i_3 \end{bmatrix} \times \begin{bmatrix} W_{i1j1} & W_{i1j2} & W_{i1j3} \\ W_{i2j1} & W_{i2j2} & W_{i2j3} \\ W_{i3j1} & W_{i3j2} & W_{i3j3} \end{bmatrix} + \begin{bmatrix} b_{j1} & b_{j2} & b_{j3} \end{bmatrix} = \begin{bmatrix} h1_{in1} & h1_{in2} & h1_{in3} \end{bmatrix}$$

Layer-1 Matrix Operation

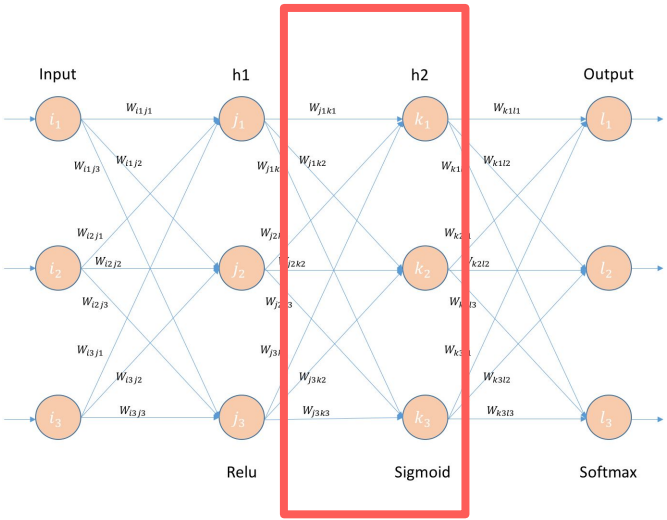
Relu operation:

$$relu = max(0, x)$$

$$\begin{bmatrix} h1_{out1} & h1_{out2} & h1_{out3} \end{bmatrix} = \begin{bmatrix} max(0, h1_{in1}) & max(0, h1_{in2}) & max(0, h1_{in3}) \end{bmatrix}$$

Layer-1 Relu Operation

Makina Öğrenmesi: Eğitim



Matrix operation:

$$\begin{bmatrix} h1_{out1} & h1_{out2} & h1_{out3} \end{bmatrix} \times \begin{bmatrix} W_{j1k1} & W_{j1k2} & W_{j1k3} \\ W_{j2k1} & W_{j2k2} & W_{j2k3} \\ W_{j3k1} & W_{j3k2} & W_{j3k3} \end{bmatrix} + \begin{bmatrix} b_{k1} & b_{k2} & b_{k3} \end{bmatrix} = \begin{bmatrix} h2_{in1} & h2_{in1} & h2_{in3} \end{bmatrix}$$

Layer-2 Matrix Operation

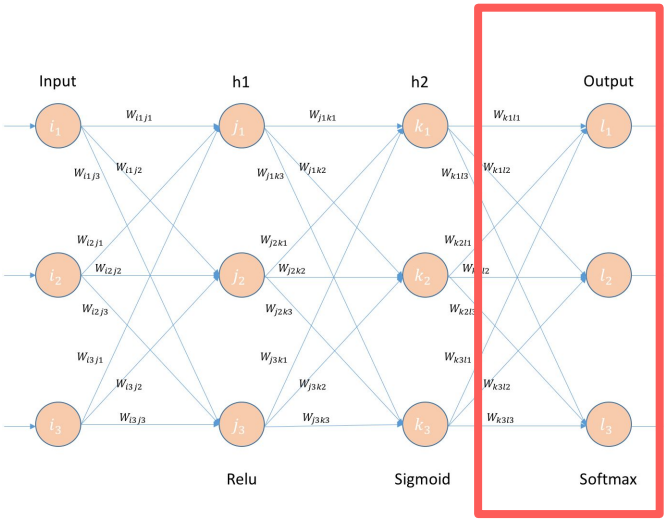
Sigmoid operation:

$$Sigmoid = 1/(1 + e^{-x})$$

$$\begin{bmatrix} h2_{out1} & h2_{out2} & h2_{out3} \end{bmatrix} = \begin{bmatrix} 1/(1 + e^{-h2_{in1}}) & 1/(1 + e^{-h2_{in2}}) & 1/(1 + e^{-h2_{in3}}) \end{bmatrix}$$

Sigmoid Operation

Makina Öğrenmesi: Eğitim



Matrix operation:

$$\begin{bmatrix} h2_{out1} & h2_{out2} & h2_{out3} \end{bmatrix} \times \begin{bmatrix} W_{k1l1} & W_{k1l2} & W_{k1l3} \\ W_{k2l1} & W_{k2l2} & W_{k2l3} \\ W_{k3l1} & W_{k3l2} & W_{k3l3} \end{bmatrix} + \begin{bmatrix} b_{l1} & b_{l2} & b_{l3} \end{bmatrix} = \begin{bmatrix} O_{in1} & O_{in2} & O_{in3} \end{bmatrix}$$

Layer-3 Matrix Operation

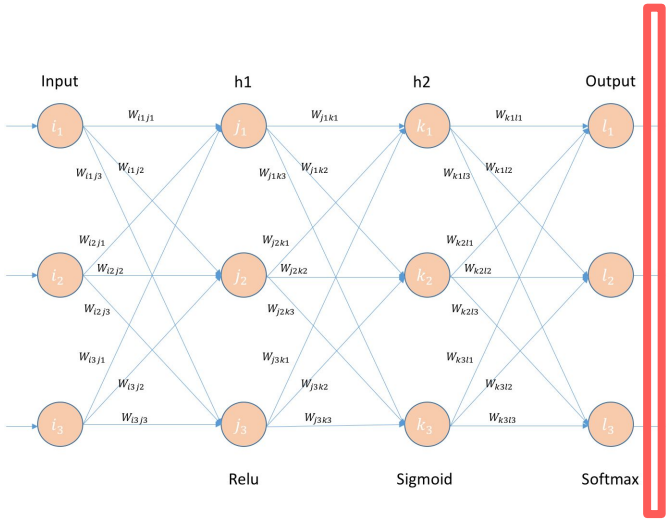
Softmax operation:

$$Softmax = e^{I_{ina}} / \left(\sum_{a=1}^{3} e^{O_{ina}}\right)$$

$$\begin{bmatrix} O_{out1} & O_{out2} & O_{out3} \end{bmatrix} = \begin{bmatrix} e^{O_{in1}} / \left(\sum_{a=1}^{3} e^{O_{ina}}\right) & e^{O_{in2}} / \left(\sum_{a=1}^{3} e^{O_{ina}}\right) & e^{O_{in3}} / \left(\sum_{a=1}^{3} e^{O_{ina}}\right) \end{bmatrix}$$

Softmax formula

Makina Öğrenmesi: Eğitim

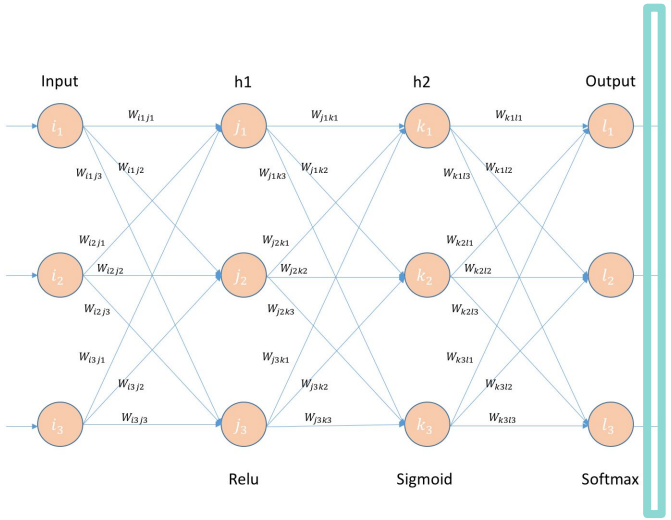


$$crossentropy = -(1/n)(\sum_{i=1}^{3}(y_i \times \log(O_{outi})) + ((1 - y_i) \times \log((1 - O_{outi}))))$$

Cross-Entropy Formula

$$Error = -((1 * \log(0.2698) + 0 + 0 * \log(0.3223) + 1 * \log(1 - 0.3223) + 0 * \log(0.4078) + 1 * \log(1 - 0.4078))$$
$$Error = -\log(0.2698) - \log(0.6777) - \log(0.5922)$$
$$Error = +0.569858 + 0.16886 + 0.22753 = 0.985$$

Makina Öğrenmesi: Eğitim



$$\frac{\partial E_1}{\partial O_{out1}} = \frac{\partial(-1 * ((y_1 * \log(O_{out1}) + (1 - y_1) * \log((1 - O_{out1}))}{\partial O_{out1}}$$

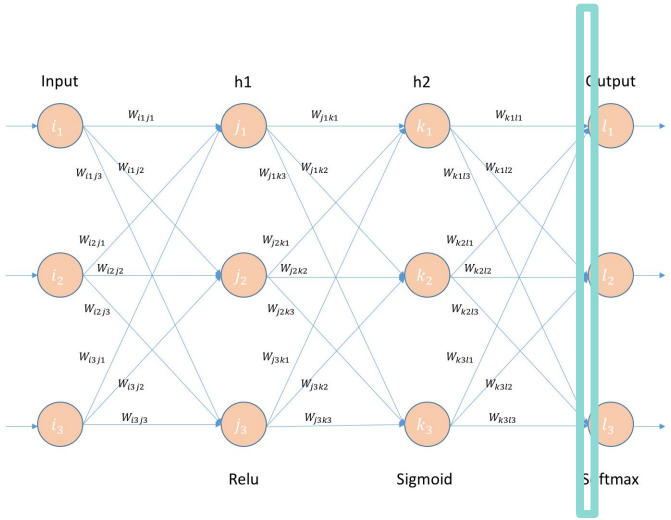
$$\frac{\partial E_1}{\partial O_{out1}} = -1 * ((y_1 * (1/O_{out1}) + (1 - y_1) * (1/(1 - O_{out1}))$$

Example: Derivative of Cross-Entropy

$$\begin{bmatrix} \frac{\partial E_1}{\partial O_{out1}} \\ \frac{\partial E_2}{\partial O_{out2}} \\ \frac{\partial E_3}{\partial O_{out3}} \end{bmatrix} = \begin{bmatrix} -1 * ((y_1 * (1/O_{out1}) + (1 - y_1) * (1/(1 - O_{out1})) \\ -1 * ((y_2 * (1/O_{out2}) + (1 - y_2) * (1/(1 - O_{out2})) \\ -1 * ((y_3 * (1/O_{out3}) + (1 - y_3) * (1/(1 - O_{out3})) \end{bmatrix}$$

Matrix of cross-entropy derivatives wrt output

Makina Öğrenmesi: Eğitim



$$Softmax = e^{x_a}/(\sum_{a=1}^{n} e^{x_a}) = e^{x_1}/(e^{x_1} + e^{x_2} + e^{x_3})$$

$$\frac{\partial(Softmax)}{\partial x_1} = (e^{x_1} \times (e^{x_2} + e^{x_3}))/(e^{x_1} + e^{x_2} + e^{x_3})^2$$

Derivative of Softmax

$$\frac{\partial O_{out1}}{\partial O_{in1}} = \frac{\partial(e^{O_{in1}}/(e^{O_{in1}} + e^{O_{in2}} + e^{O_{in3}}))}{\partial O_{in1}}$$

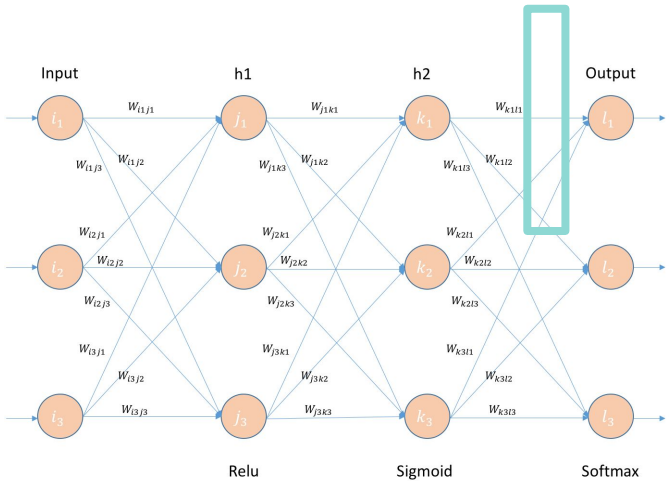
$$\frac{\partial O_{out1}}{\partial O_{in1}} = (e^{O_{in1}} \times (e^{O_{in2}} + e^{O_{in3}}))/(e^{O_{in1}} + e^{O_{in2}} + e^{O_{in3}})^2$$

Example: Derivative of softmax wrt output layer input

$$\begin{bmatrix} \frac{\partial O_{out1}}{\partial O_{in1}} \\ \frac{\partial O_{out2}}{\partial O_{in2}} \\ \frac{\partial O_{out3}}{\partial O_{in3}} \end{bmatrix} = \begin{bmatrix} (e^{O_{in1}} \times (e^{O_{in2}} + e^{O_{in3}}))/(e^{O_{in1}} + e^{O_{in2}} + e^{O_{in3}})^2 \\ (e^{O_{in2}} \times (e^{O_{in1}} + e^{O_{in3}}))/(e^{O_{in1}} + e^{O_{in2}} + e^{O_{in3}})^2 \\ (e^{O_{in3}} \times (e^{O_{in1}} + e^{O_{in2}}))/(e^{O_{in1}} + e^{O_{in2}} + e^{O_{in3}})^2 \end{bmatrix}$$

Matrix of Derivative of softmax wrt output layer input.

Makina Öğrenmesi: Eğitim



$$\frac{\partial O_{in1}}{\partial W_{k1l1}} = \frac{\partial((h2_{out1} * W_{j1k1}) + (h2_{out2} * Wj2k1) + (h2_{out3} * Wj3k1) + b_{l1})}{\partial W_{k1l1}}$$

$$\frac{\partial O_{in1}}{\partial W_{k1l1}} = h2_{out1}$$

Example: Derivative of input to output layer wrt weight

$$\begin{bmatrix} \frac{\partial O_{in1}}{\partial W_{k1l1}} \\ \frac{\partial O_{in1}}{\partial W_{k2l1}} \\ \frac{\partial O_{in1}}{\partial W_{k3l1}} \end{bmatrix} = \begin{bmatrix} h2_{out1} \\ h2_{out2} \\ h2_{out3} \end{bmatrix}$$

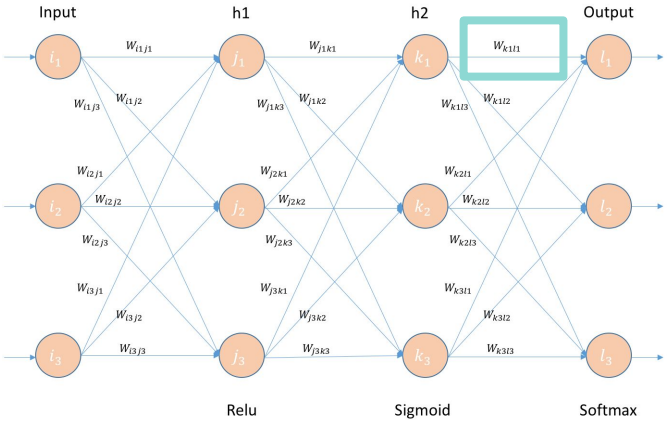
$$\begin{bmatrix} \frac{\partial O_{in2}}{\partial W_{k1l2}} \\ \frac{\partial O_{in2}}{\partial W_{k2l2}} \\ \frac{\partial O_{in2}}{\partial W_{k3l2}} \end{bmatrix} = \begin{bmatrix} h2_{out1} \\ h2_{out2} \\ h2_{out3} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial O_{in3}}{\partial W_{k1l3}} \\ \frac{\partial O_{in3}}{\partial W_{k2l3}} \\ \frac{\partial O_{in3}}{\partial W_{k3l3}} \end{bmatrix} = \begin{bmatrix} h2_{out1} \\ h2_{out2} \\ h2_{out3} \end{bmatrix}$$

Makina Öğrenmesi: Eğitim

3) Bu parametredeki değişim

$$\frac{\partial E_1}{\partial W_{k1l1}} = \frac{\partial E_1}{\partial O_{out1}} * \frac{\partial O_{out1}}{\partial O_{in1}} * \frac{\partial O_{in1}}{\partial W_{k1l1}}$$



$$\frac{\partial O_{in1}}{\partial W_{k1l1}} = \frac{\partial((h2_{out1} * W_{j1k1}) + (h2_{out2} * Wj2k1) + (h2_{out3} * Wj3k1) + b_{l1})}{\partial W_{k1l1}}$$

$$\frac{\partial O_{in1}}{\partial W_{k1l1}} = h2_{out1}$$

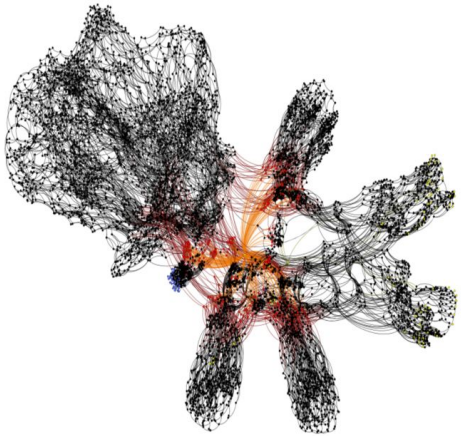
Example: Derivative of input to output layer wrt weight

$$\begin{bmatrix} \frac{\partial O_{in1}}{\partial W_{k1l1}} \\ \frac{\partial O_{in1}}{\partial W_{k2l1}} \\ \frac{\partial O_{in1}}{\partial W_{k3l1}} \end{bmatrix} = \begin{bmatrix} h2_{out1} \\ h2_{out2} \\ h2_{out3} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial O_{in2}}{\partial W_{k1l2}} \\ \frac{\partial O_{in2}}{\partial W_{k2l2}} \\ \frac{\partial O_{in2}}{\partial W_{k3l2}} \end{bmatrix} = \begin{bmatrix} h2_{out1} \\ h2_{out2} \\ h2_{out3} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial O_{in3}}{\partial W_{k1l3}} \\ \frac{\partial O_{in3}}{\partial W_{k2l3}} \\ \frac{\partial O_{in3}}{\partial W_{k3l3}} \end{bmatrix} = \begin{bmatrix} h2_{out1} \\ h2_{out2} \\ h2_{out3} \end{bmatrix}$$

Çizge Analizi - Makina Öğrenmesi

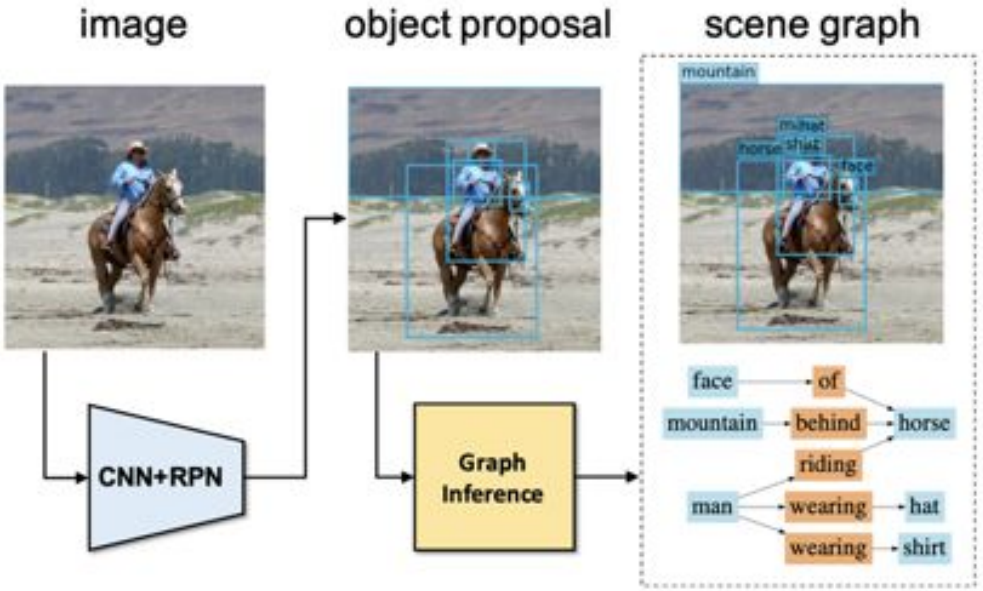


Çizge üzerinde makina öğrenmesi:

- Nokta sınıflandırma
- Anomali bulma
- Bağlantı tahmini

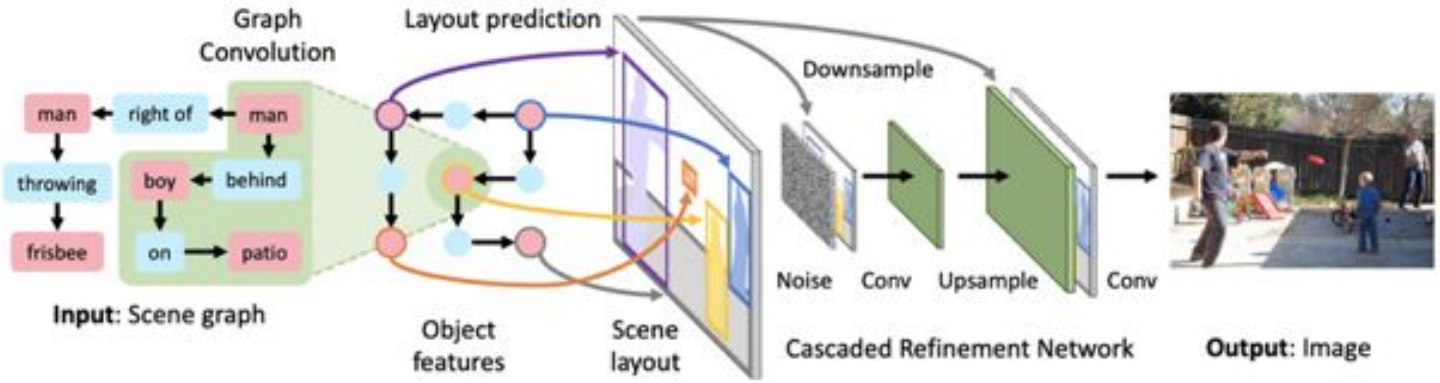
Klasik yöntemler çizge verisini işlemek için uygun değil.
Tablosal veri üzerinde uzmanlaşmışlar ve veri elemanlarının (satırların) özelliklerini (sütunları) işliyorlar.
Bazı durumlarda yapısal çizge verisini, tablosal veriye çevirmek gerekiyor.

Çizge Analizi - Makina Öğrenmesi



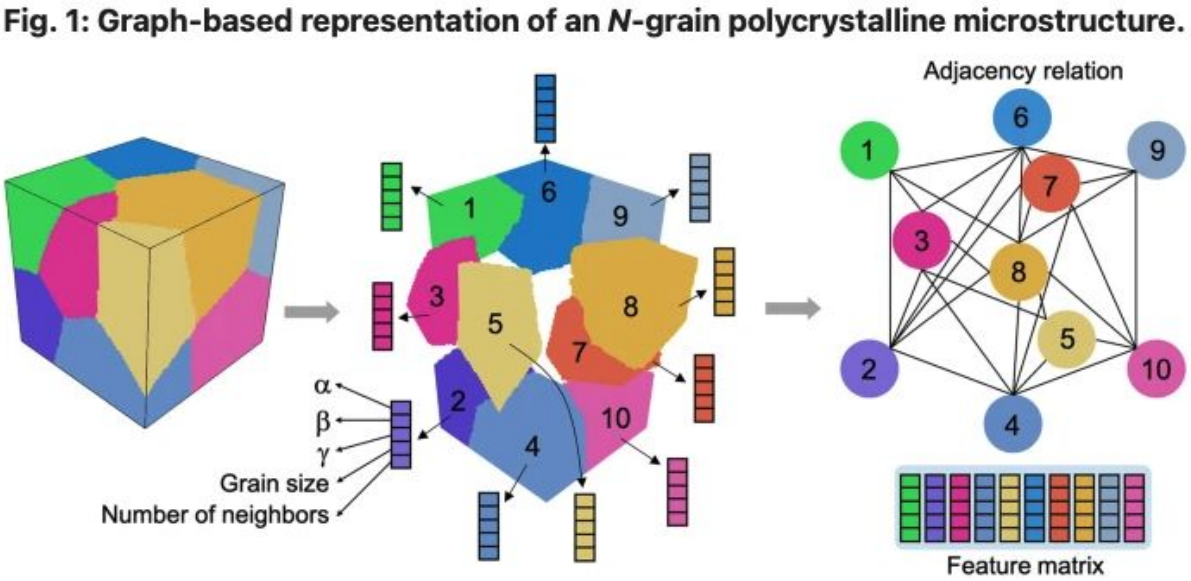
D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, “Scene graph generation by iterative message passing,” in Proc. of CVPR, 2017

Çizge Analizi - Makina Öğrenmesi



J. Johnson, A. Gupta, and L. Fei-Fei, “Image generation from scene graphs,” in Proc. of CVPR, 2018

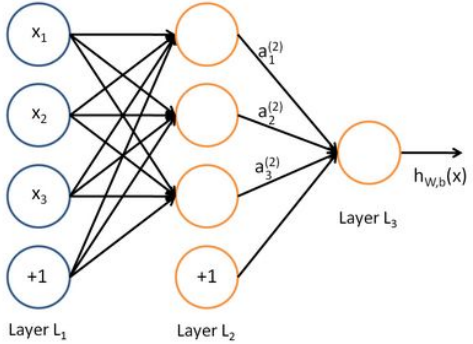
Çizge Analizi - Makina Öğrenmesi



Graph neural networks for an accurate and interpretable prediction of the properties of polycrystalline materials Minyi Dai, Mehmet F. Demirel, Yingyu Liang & Jia-Mian Hu npj Computational Materials volume 7, Article number: 103 (2021)

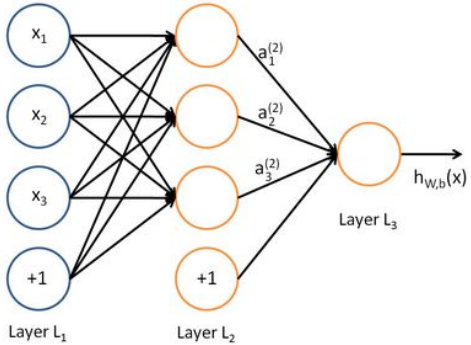
Çizge Analizi - Makina Öğrenmesi

Bunu yapıyoruz

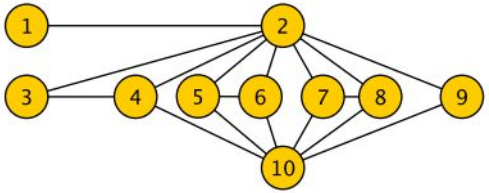


	1	2	3	4	5	6	7	8	9	10
1		■								
2	■		■	■	■	■	■	■	■	
3		■	■							
4		■	■							
5		■				■				■
6		■			■					■
7		■						■		■
8		■					■			■
9		■								■
10				■	■	■	■	■	■	

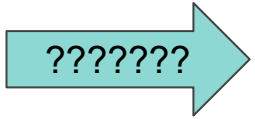
Bu niye olmasın

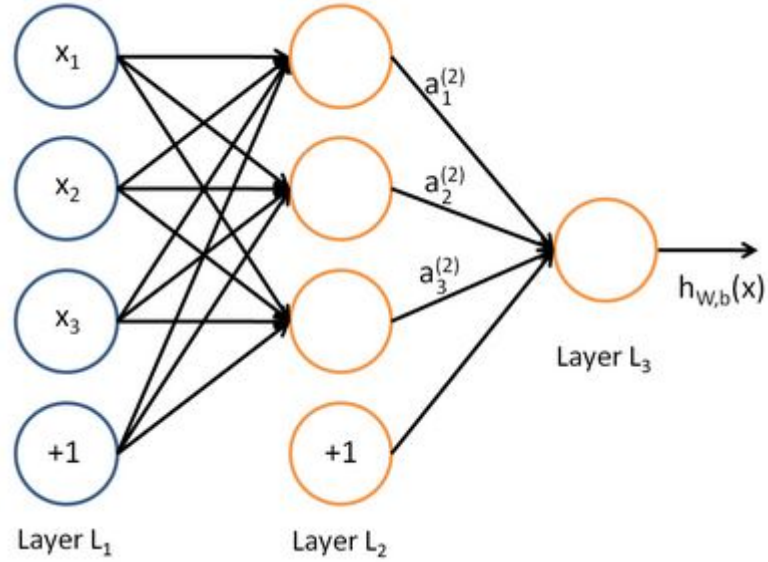


Çizge Analizi - Makina Öğrenmesi



	1	2	3	4	5	6	7	8	9	10
1		■								
2	■		■	■	■	■	■	■	■	
3		■	■							
4		■	■							
5		■				■				■
6		■			■					■
7		■						■		■
8		■					■			■
9		■								■
10				■	■	■	■	■	■	





Çizge Analizi - Makina Öğrenmesi

Çizgeler yüz milyonlarca, milyarlarca noktaya sahip olabilirler. Bu tür bir çizge için matris gösterimi 10^18 elemana ihtiyaç duyacaktır. Bu da şu anki hesaplama ve depolama kapasiteleri ile mümkün değildir.

Graph	$\|V\|$	$\|E\|$	Density
com-dblp [11]	317,080	1,049,866	3.31
com-amazon [11]	334,863	925,872	2.76
youtube [12]	1,138,499	4,945,382	4.34
flickr [12]	1,715,254	22,613,981	13.18
com-orkut [11]	3,072,441	117,185,083	38.14
com-lj [11]	3,997,962	34,681,189	8.67
hyperlink2012 [13]	39,497,204	623,056,313	15.77
twitter_rv [14]	41,652,230	1,468,365,182	35.25
com-friendster [11]	65,608,366	1,806,067,135	27.53

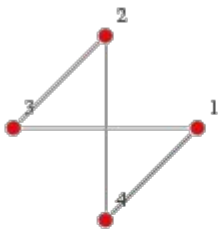
Çizge Analizi - Makina Öğrenmesi

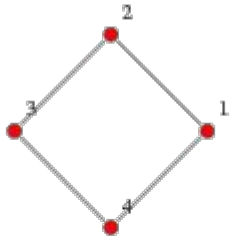
Çizgelerin boyutları farklı - her çizge için farklı bir ağ mı eğitilecek?

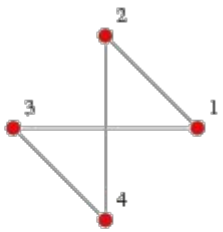
Graph	$\|V\|$	$\|E\|$	Density
com-dblp [11]	317,080	1,049,866	3.31
com-amazon [11]	334,863	925,872	2.76
youtube [12]	1,138,499	4,945,382	4.34
flickr [12]	1,715,254	22,613,981	13.18
com-orkut [11]	3,072,441	117,185,083	38.14
com-lj [11]	3,997,962	34,681,189	8.67
hyperlink2012 [13]	39,497,204	623,056,313	15.77
twitter_rv [14]	41,652,230	1,468,365,182	35.25
com-friendster [11]	65,608,366	1,806,067,135	27.53

Çizge Analizi - Makina Öğrenmesi

Çizge aynı ama matrisler farklı - aynı çizge için farklı bir eğitim süreci mi gerçekleşecek?

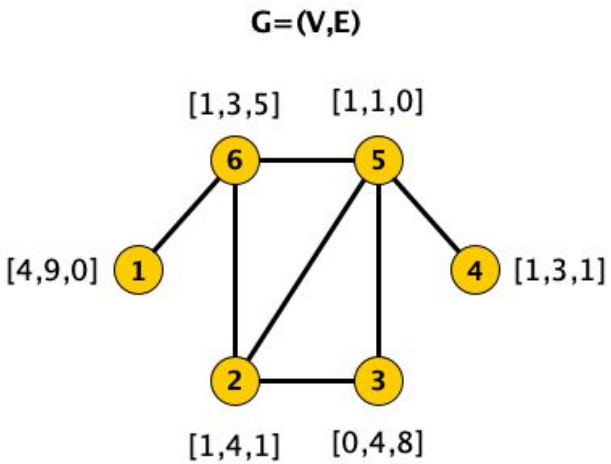






$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

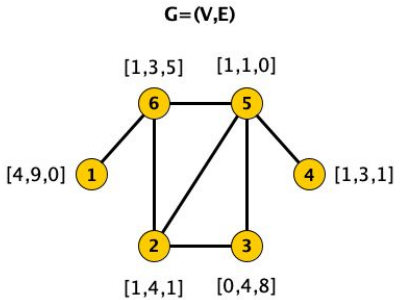
Çizge Analizi - Çizge Sinir Ağları

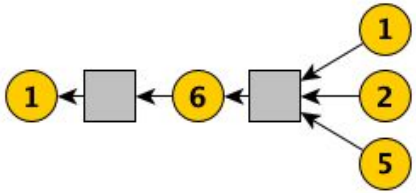


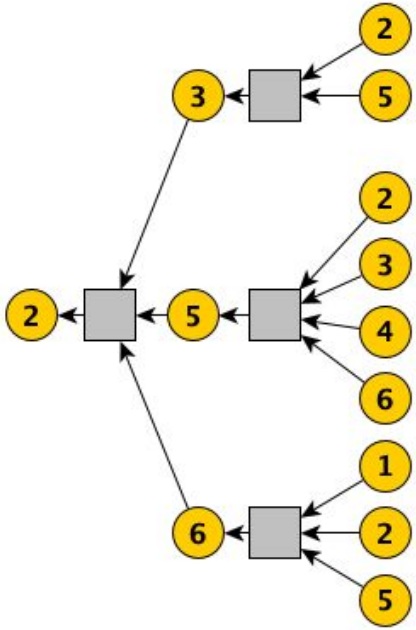
$$\mathbf{X} = \begin{bmatrix} 4 & 9 & 0 \\ 1 & 4 & 1 \\ 0 & 4 & 8 \\ 1 & 3 & 1 \\ 1 & 1 & 0 \\ 1 & 3 & 5 \end{bmatrix} \in \mathcal{R}^{m \times |V|}$$

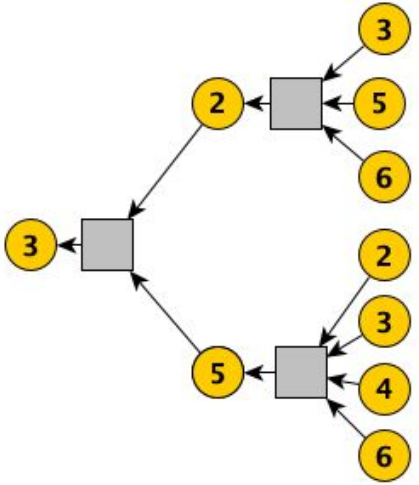
$$\mathbf{A} = Adj(\mathbf{G}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Çizge Analizi - Çizge Sinir Ağları

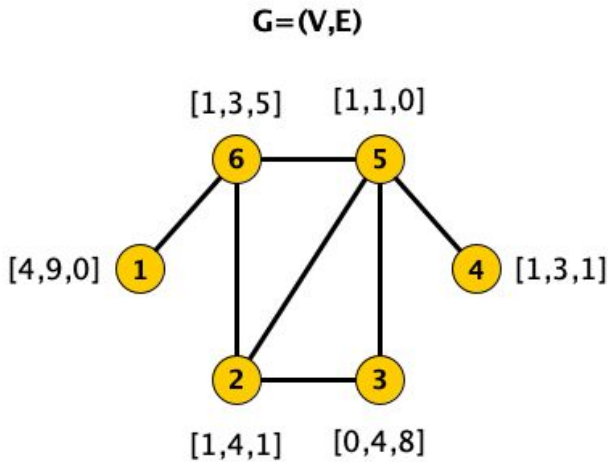


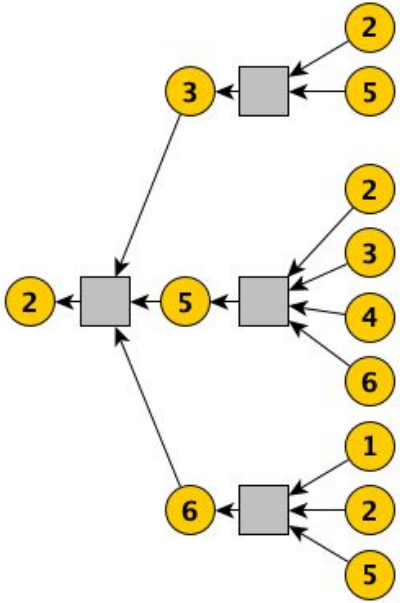






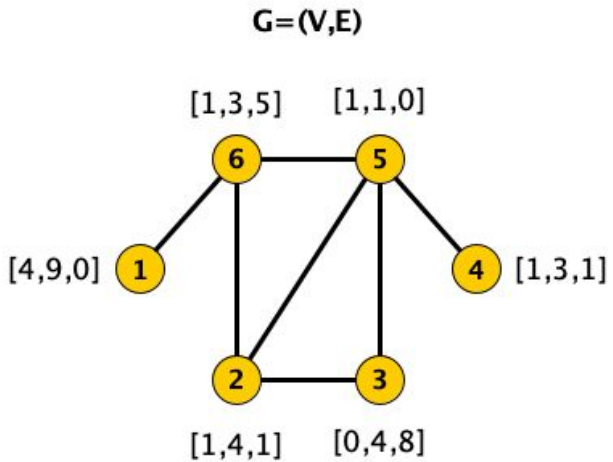
Çizge Analizi - Çizge Sinir Ağları

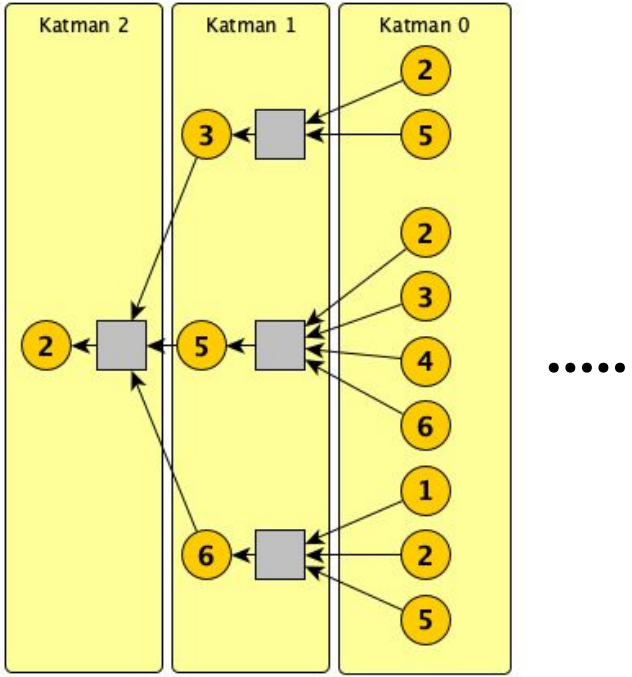




2 no'lu nokta için hesap çizgesi (ağı).

Çizge Analizi - Çizge Sinir Ağları





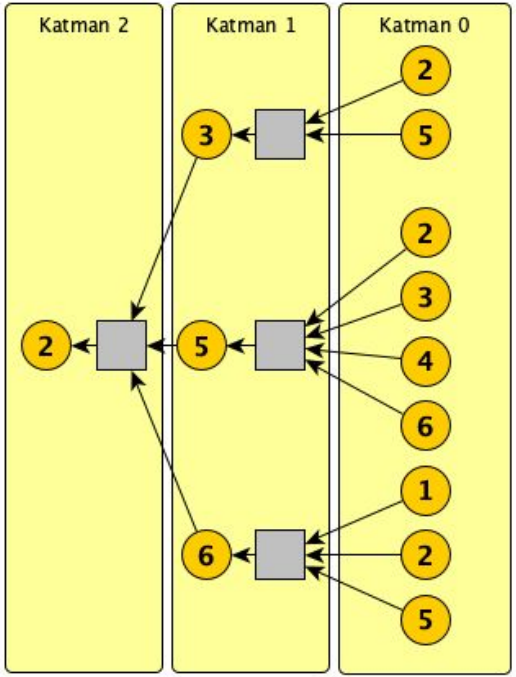
2 no'lu nokta için hesap çizgesi (ağı).

Çizge Analizi - Çizge Sinir Ağları

$$h_v^0 = \mathbf{X}_v$$

$$h_v^{k+1} = \sigma\left(W_k \sum_{u \in \Gamma_v} \frac{h_u^k}{|\Gamma_v|} + B_k h_v^k\right)$$

$$Z_v = h_v^K$$



.....

2 no'lu nokta için hesap çizgesi (ağı).

Çizge Analizi - Çizge Sinir Ağları

aktivasyon fonksiyonu

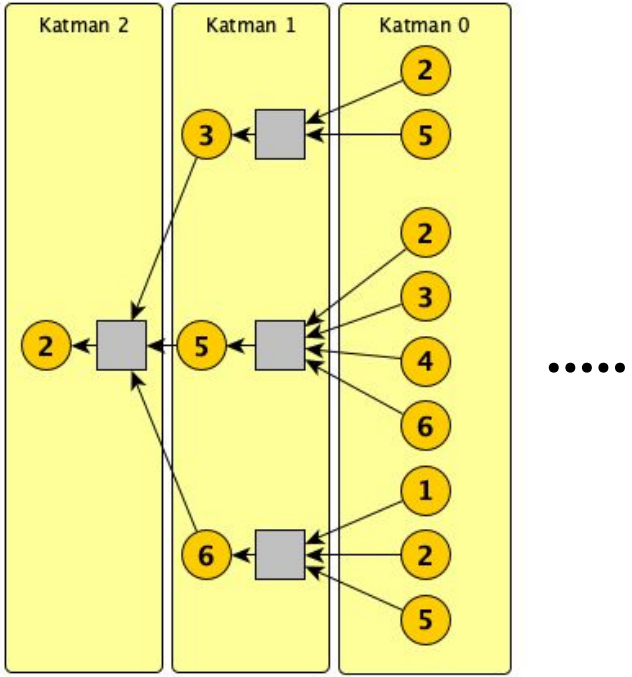
eğitilen değerler

$$h_v^{k+1} = \sigma\left(W_k \sum_{u \in \Gamma_v} \frac{h_u^k}{|\Gamma_v|} + B_k h_v^k\right)$$

v noktasının k + 1 adımındaki değer vektörü

komşulardan gelen değerler

v noktasının k adımındaki değer vektörü



.....

2 no'lu nokta için hesap çizgesi (ağı).

Çizge Analizi - Çizge Sinir Ağları

IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 20, NO. 1, JANUARY 2009

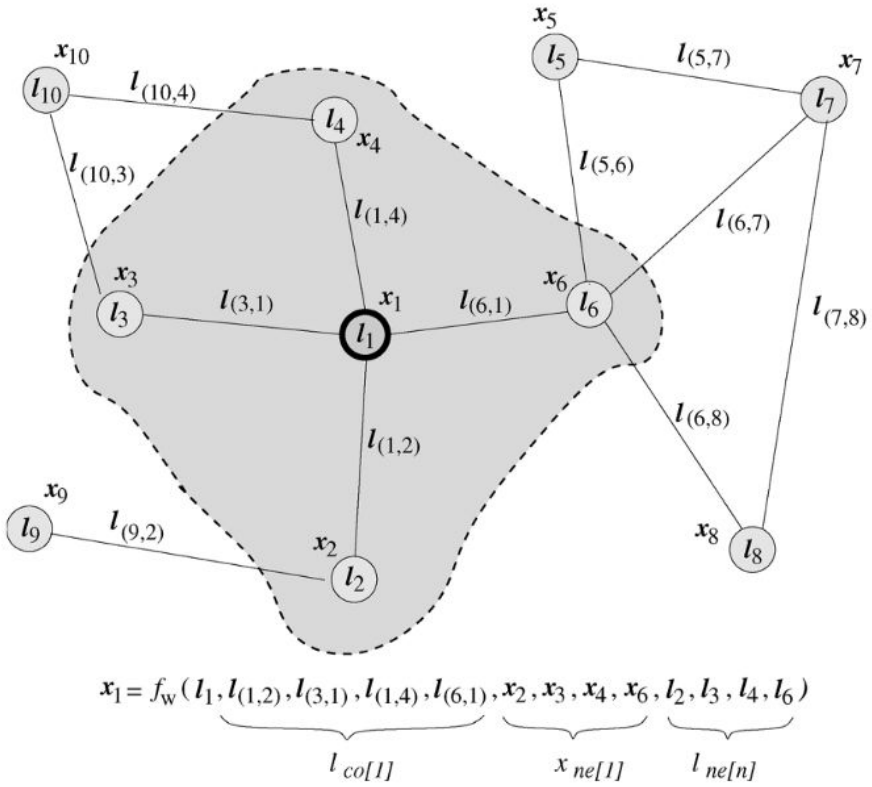
The Graph Neural Network Model

Franco Scarselli, Marco Gori, *Fellow, IEEE*, Ah Chung Tsoi, Markus Hagenbuchner, *Member, IEEE*, and Gabriele Monfardini

The graph neural network model — 2641 — 2008
F Scarselli, M Gori, AC Tsoi, M Hagenbuchner, G Monfardini
IEEE transactions on neural networks 20 (1), 61-80

Çizge Analizi - Çizge Sinir Ağları

$$\mathbf{x}_v = f_w(\mathbf{l}_v, \mathbf{l}_{co[v]}, \mathbf{x}_{ne[v]}, \mathbf{l}_{ne[v]})$$
$$o_v = g_w(\mathbf{x}_v, \mathbf{l}_v)$$

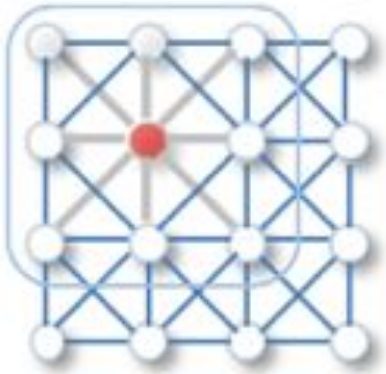


Çizge Analizi - Çizge Sinir Ağları

$$\mathbf{x}_v = f_w(\mathbf{l}_v, \mathbf{l}_{co[v]}, \mathbf{x}_{ne[v]}, \mathbf{l}_{ne[v]})$$
$$o_v = g_w(\mathbf{x}_v, \mathbf{l}_v)$$

Eğer **G** normal (pozisyonel olmayan) bir çizge ise f_w fonksiyonu aşağıdaki gibi, parametrik bir h_w fonksiyonu ile yazılabilir:

$$f_w = \sum_{u \in ne[v]} h_w(\mathbf{l}_v, \mathbf{l}_{(v,u)}, \mathbf{x}_u, \mathbf{l}_u)$$



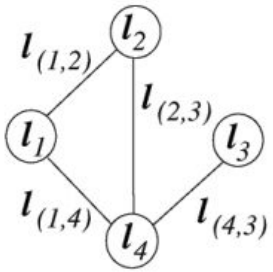
Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, Philip S. Yu, "A Comprehensive Survey on Graph Neural Networks", arxiv, 2019

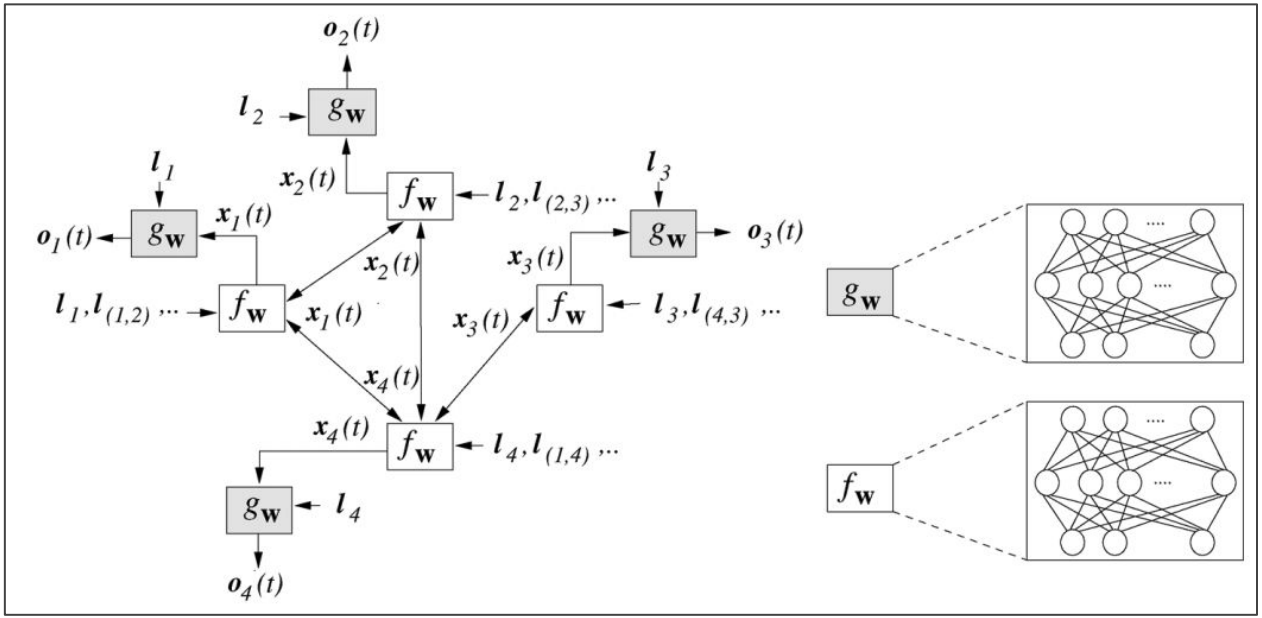
Çizge Analizi - Çizge Sinir Ağları

Banach Sabit Nokta Teoremi: (**X**, **d**) *bir tam metrik uzay olsun,* (**T**: **X**→**X**) *ise bir büzülme fonksiyonu olsun. Bu durumda* **T** *tek bir* **x**’ *sabit noktasına sahiptir ve herhangi bir* $\mathbf{x} \in \mathbf{X}$ *için* **n**→∞*’a giderken* $T_n(x)$ *değeri* **x**’ değerine yakınsar.

$$x^k = T(x^{k-1}), k \in [1, n]$$

Çizge Analizi - Çizge Sinir Ağları

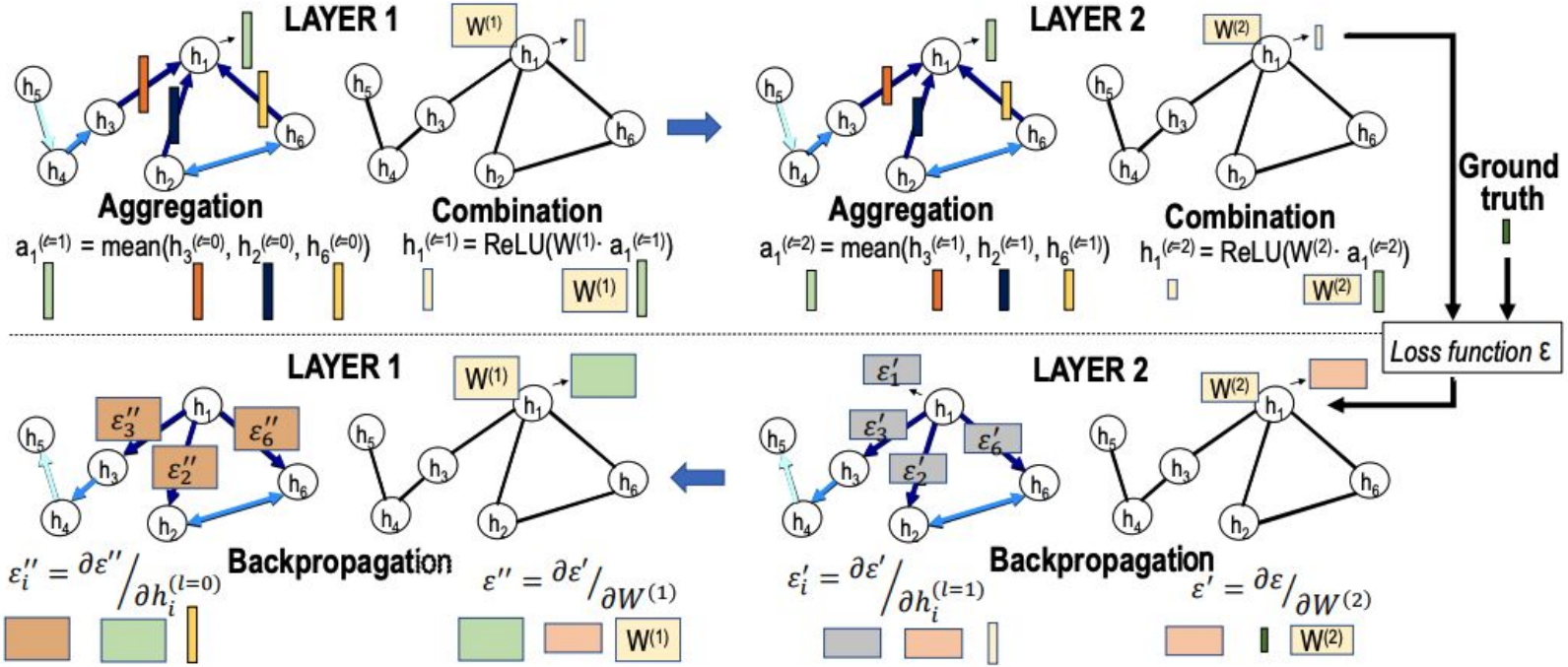




$$\mathbf{x}_v = f_w(\mathbf{l}_v, \mathbf{l}_{co[v]}, \mathbf{x}_{ne[v]}, \mathbf{l}_{ne[v]})$$

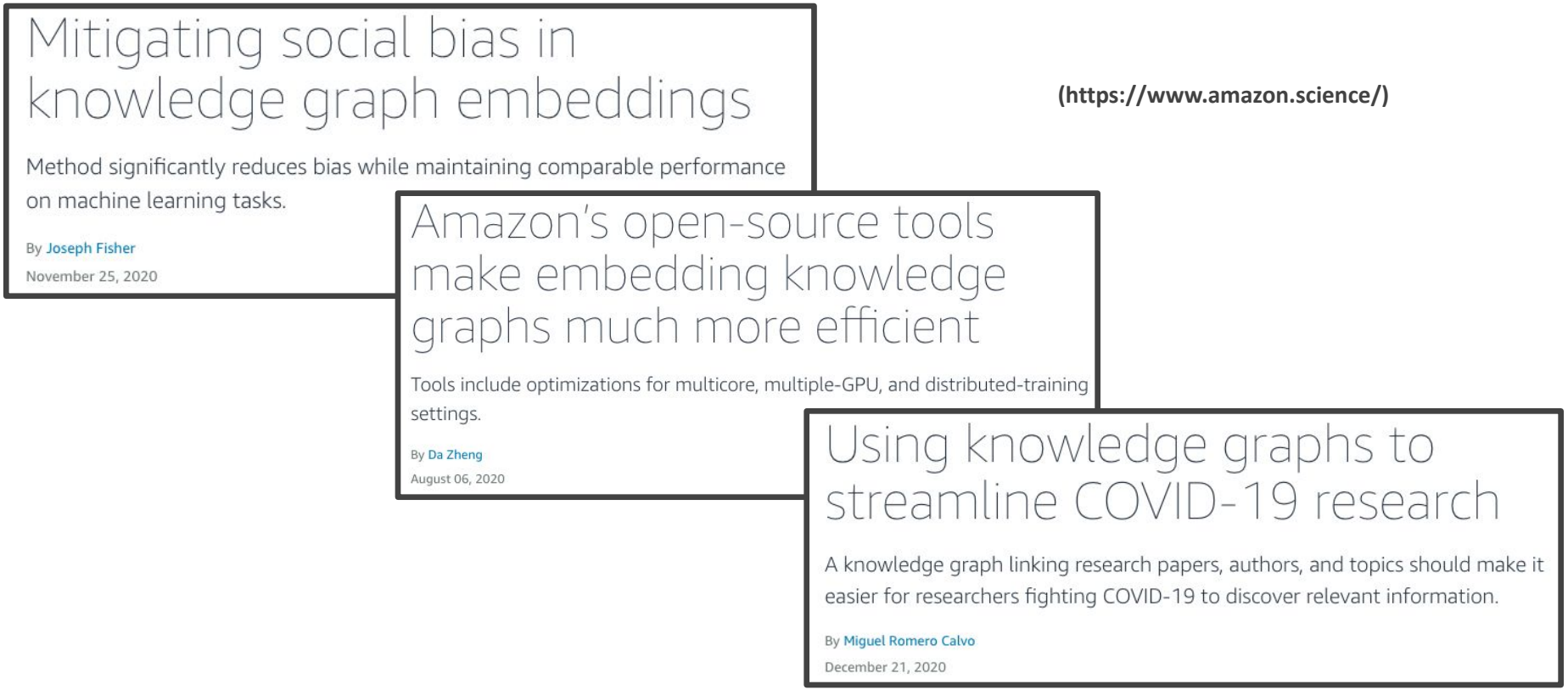
$$o_v = g_w(\mathbf{x}_v, \mathbf{l}_v)$$

Çizge Analizi - Çizge Sinir Ağları



Computing Graph Neural Networks: A Survey from Algorithms to Accelerators SERGI ABADAL, AKSHAY JAIN, ROBERT GUIRADO, JORGE LÓPEZ-ALONSO, and EDUARD ALARCÓN, arxiv, Jul 2021.

Çizge Analizi - Çizge Sinir Ağları



(https://www.amazon.science/)

Çizge Evrişimsel Ağları

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

Hata fonksiyonu

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}}, \quad \text{with} \quad \mathcal{L}_{\text{reg}} = \sum A_{ij} \| f(X_i) - f(X_j) \|^2 = f(X)^\top \Delta f(X)$$

Katmanlar arası geçiş

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right)$$

$$\tilde{A} = A + I_N$$

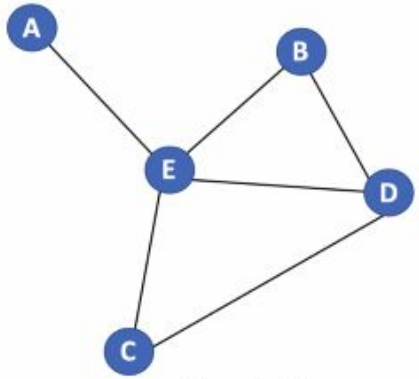
birim matrisi

Çizge Evrişimsel Ağları

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl



Graph G

	A	B	C	D	E
A	0	0	0	0	1
B	0	0	0	1	1
C	0	0	0	1	1
D	0	1	1	0	1
E	1	1	1	1	0

Adjacency matrix A

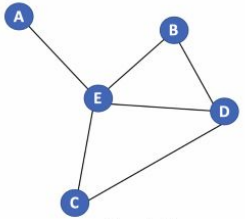
	A	B	C	D	E
A	1	0	0	0	0
B	0	2	0	0	0
C	0	0	2	0	0
D	0	0	0	3	0
E	0	0	0	0	4

Degree matrix D

A	-1.1	3.2	4.2
B	0.4	5.1	-1.2
C	1.2	1.3	2.1
D	1.4	-1.2	2.5
E	1.4	2.5	4.5

Feature vector X

Çizge Evrişimsel Ağları



Graph G

	A	B	C	D	E
A	0	0	0	0	1
B	0	0	0	1	1
C	0	0	0	1	1
D	0	1	1	0	1
E	1	1	1	1	0

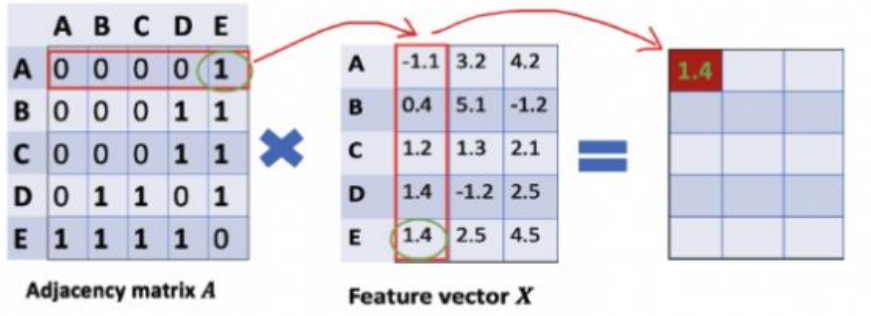
Adjacency matrix A

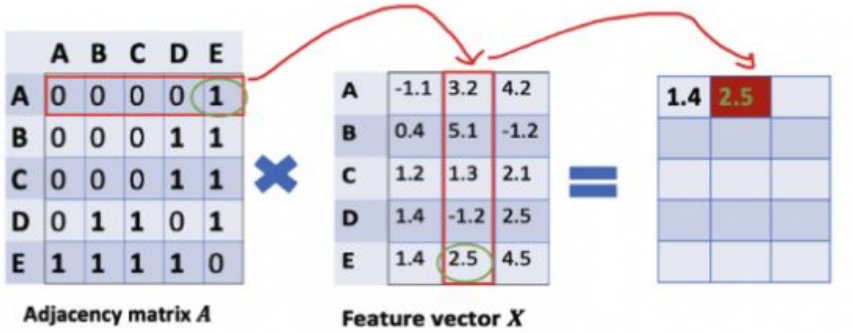
	A	B	C	D	E
A	**1**	0	0	0	0
B	0	**2**	0	0	0
C	0	0	**2**	0	0
D	0	0	0	**3**	0
E	0	0	0	0	**4**

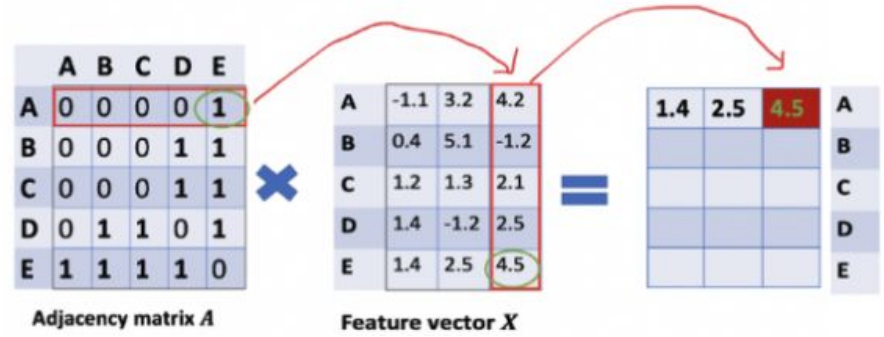
Degree matrix D

A	-1.1	3.2	4.2
B	0.4	5.1	-1.2
C	1.2	1.3	2.1
D	1.4	-1.2	2.5
E	1.4	2.5	4.5

Feature vector X







Çizge Evrişimsel Ağları

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

	A	B	C	D	E
A	0	0	0	0	1
B	0	0	0	1	1
C	0	0	0	1	1
D	0	1	1	0	1
E	1	1	1	1	0

Adjacency matrix A

+

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Identity matrix I

=

	A	B	C	D	E
A	1	0	0	0	1
B	0	1	0	1	1
C	0	0	1	1	1
D	0	1	1	1	1
E	1	1	1	1	1

New Adjacency matrix $\tilde{A}$

$$\tilde{A} = A + I_N$$

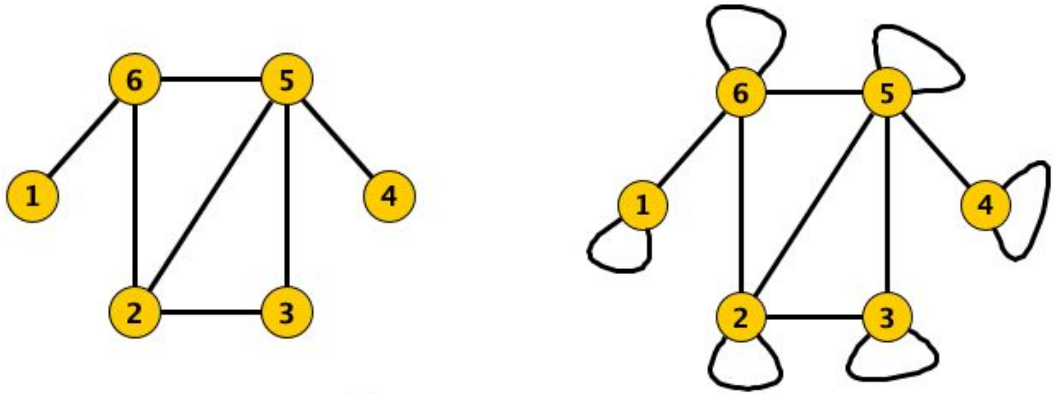
birim matrisi

Çizge Evrişimsel Ağları

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl



$$\tilde{A} = A + I_N$$

$$\mathbf{x}_v = f_w(\mathbf{l}_v, \mathbf{l}_{co[v]}, \mathbf{x}_{ne[v]}, \mathbf{l}_{ne[v]})$$

$$o_v = g_w(\mathbf{x}_v, \mathbf{l}_v)$$

Çizge Evrişimsel Ağları

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

Hata fonksiyonu

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}}, \quad \text{with} \quad \mathcal{L}_{\text{reg}} = \sum A_{ij} \| f(X_i) - f(X_j) \|^2 = f(X)^\top \Delta f(X)$$

Katmanlar arası geçiş

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right)$$

$$\tilde{A} = A + I_N$$

normalizasyon

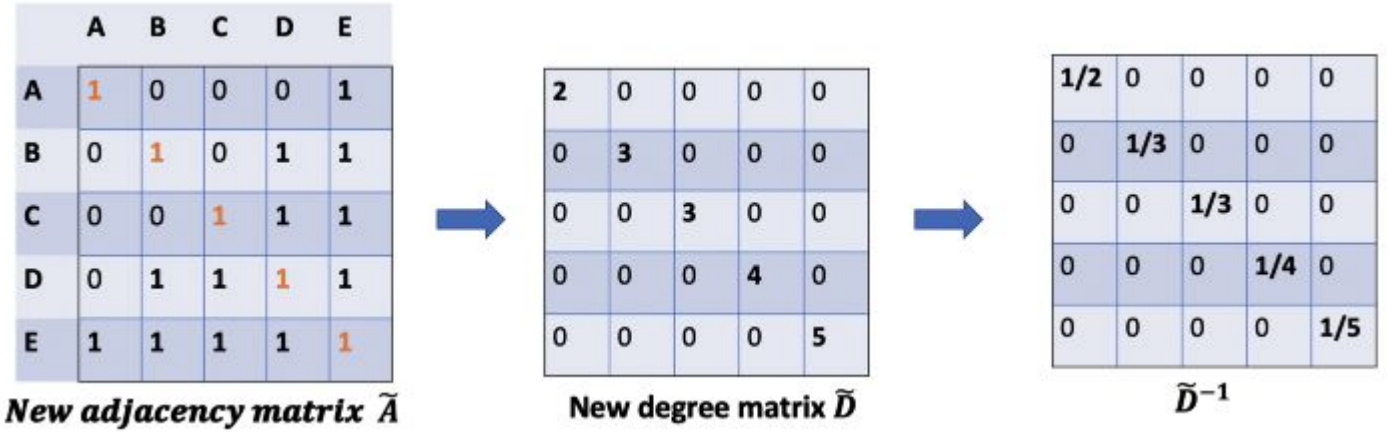
birim matrisi

Çizge Evrişimsel Ağları

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl



	A	B	C	D	E
A	1	0	0	0	1
B	0	1	0	1	1
C	0	0	1	1	1
D	0	1	1	1	1
E	1	1	1	1	1

New adjacency matrix $\tilde{A}$

2	0	0	0	0
0	3	0	0	0
0	0	3	0	0
0	0	0	4	0
0	0	0	0	5

New degree matrix $\tilde{D}$

1/2	0	0	0	0
0	1/3	0	0	0
0	0	1/3	0	0
0	0	0	1/4	0
0	0	0	0	1/5

$\tilde{D}^{-1}$

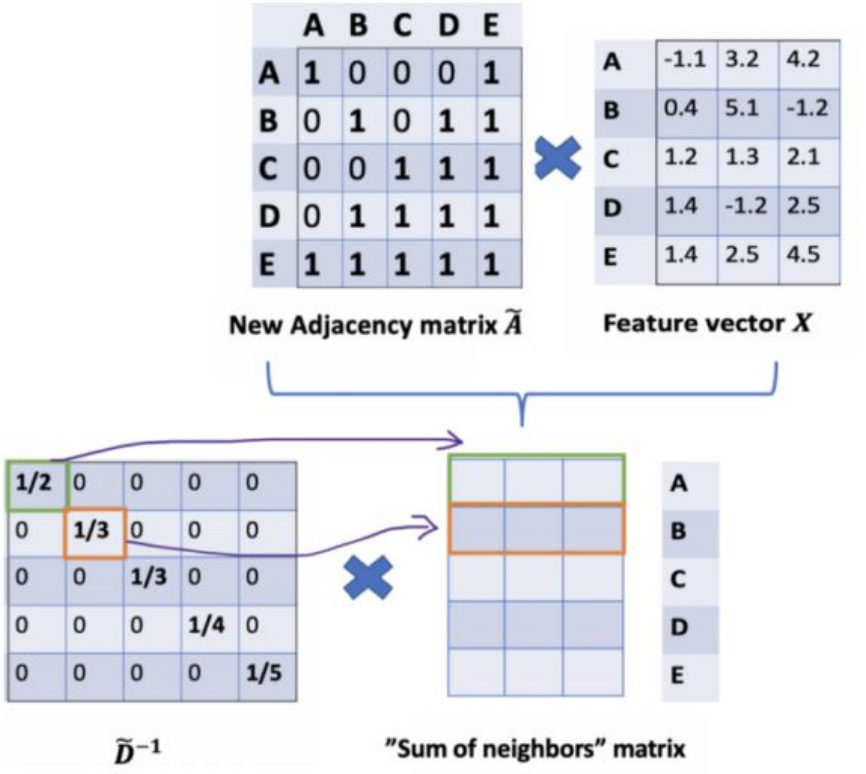
Çizge Evrişimsel Ağları

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

Özelliklerin ağırlıklandırılması?



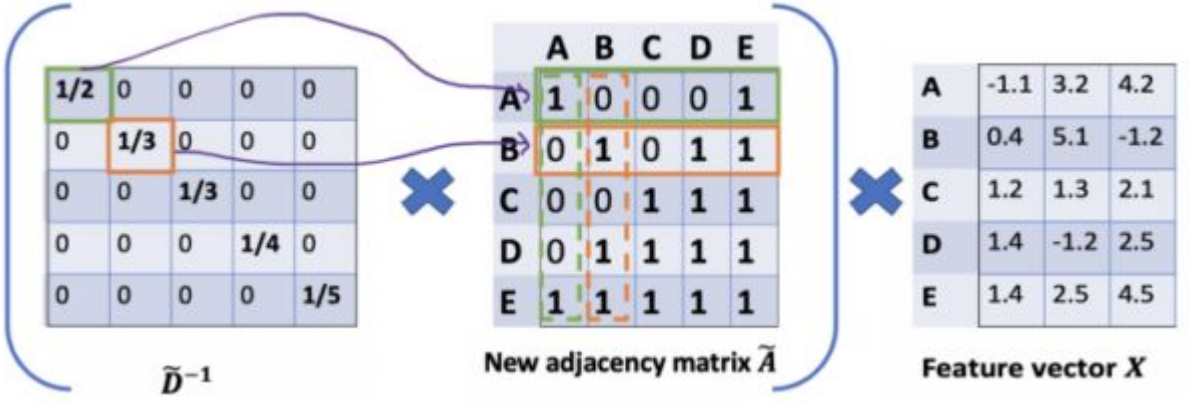
Çizge Evrişimsel Ağları

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

Aslında matrisin ağırlıklandırılması ile aynı.



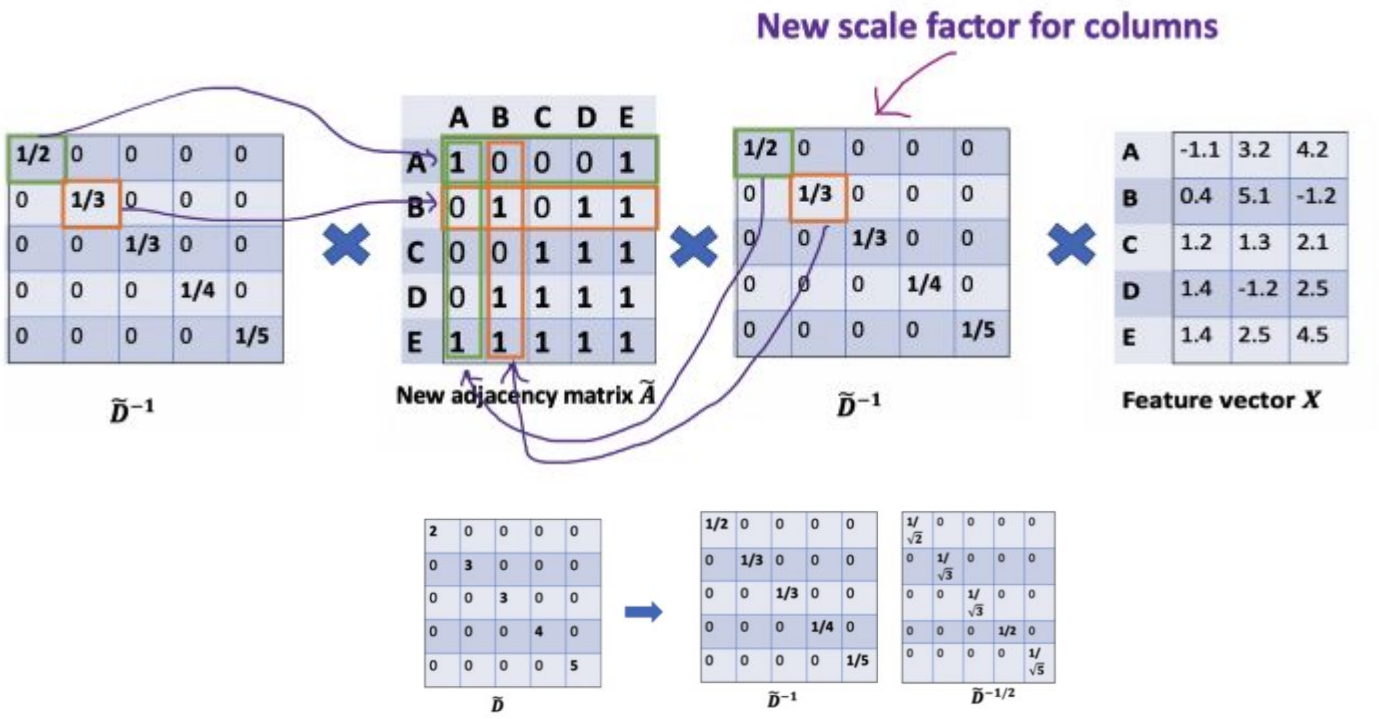
Çizge Evrişimsel Ağları

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

Farklı bir normalizasyon yöntemi.

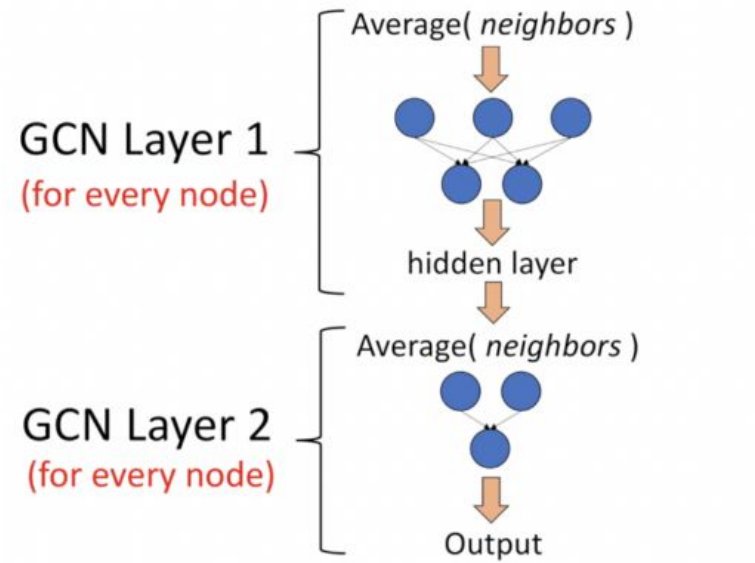


Çizge Evrişimsel Ağları

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl



$$\mathbf{x}_v = f_w(\mathbf{l}_v, \mathbf{l}_{co[v]}, \mathbf{x}_{ne[v]}, \mathbf{l}_{ne[v]})$$

$$o_v = g_w(\mathbf{x}_v, \mathbf{l}_v)$$

Çizge Evrişimsel Ağları

Örnek bir koda bakalım
(2. defter)

GraphSAGE

Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

Eski yöntemler:

- Eğitim için bütün çizgeye ihtiyaç duyuyor (transductive).
- Yerleştirmeleri buluyor fakat.
  - Yeni nokta geldiğinde
  - Öğrenilen parametreler yeni bir çizgeye uygulanmak istendiğinde

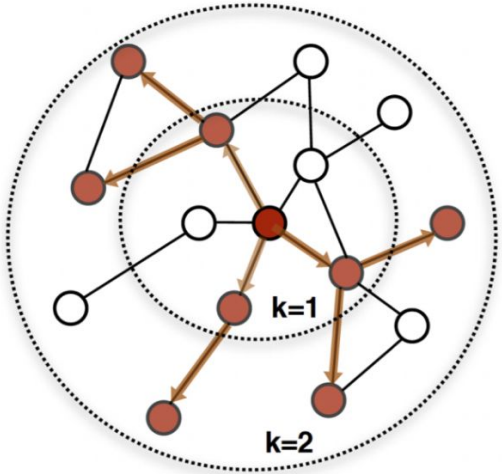
  sorun oluyor.

GraphSAGE biraz önce gördüğümüz GCNlerde kullanılan fonksiyonları öğrenmeye çalışan bir algoritmadır.

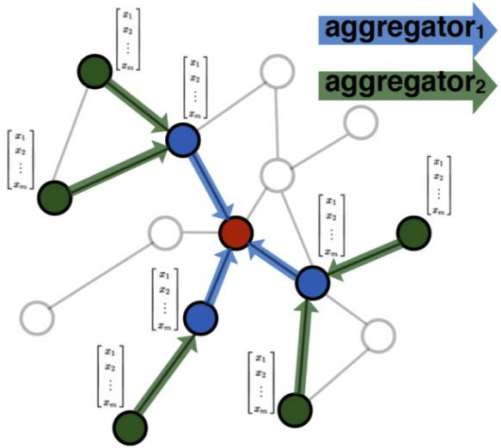
GraphSAGE

Inductive Representation Learning on Large Graphs

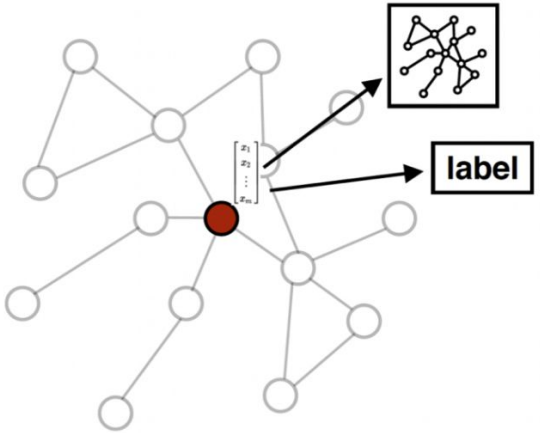
William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu



1. Sample neighborhood



2. Aggregate feature information from neighbors



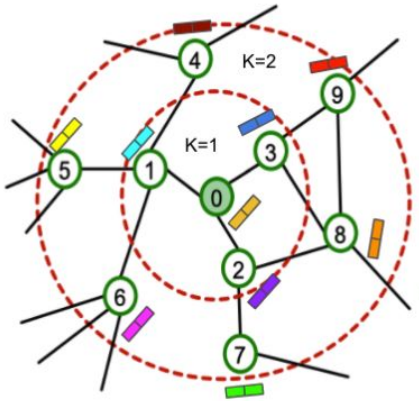
3. Predict graph context and label using aggregated information

GraphSAGE

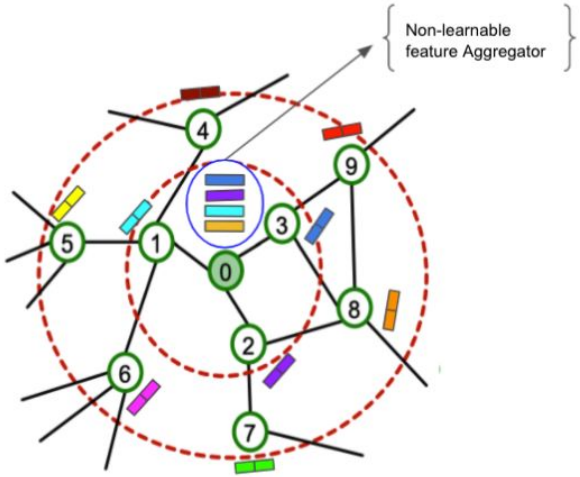
Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu **Rex Ying*** rexying@stanford.edu **Jure Leskovec** jure@cs.stanford.edu

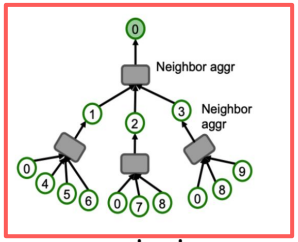
Graph Convolutional Networks



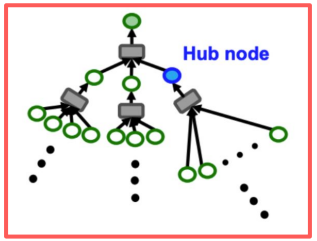
Input Graph



Feature Aggregation for the target node **0** at K=1



pahalı



hub noktalar

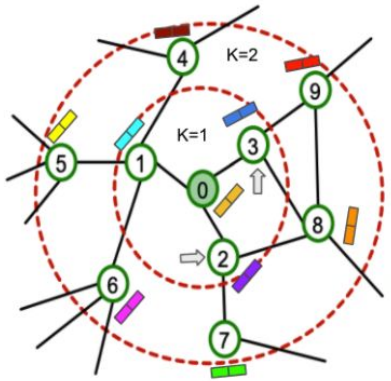
GraphSAGE

Inductive Representation Learning on Large Graphs

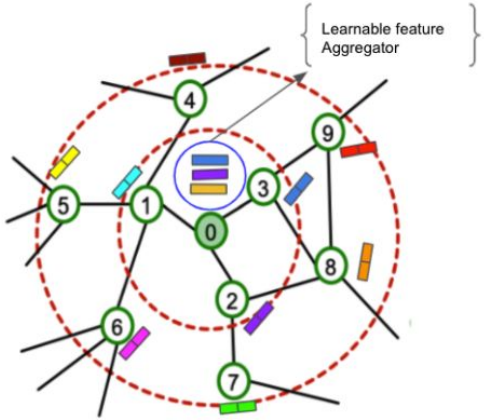
William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

GraphSage

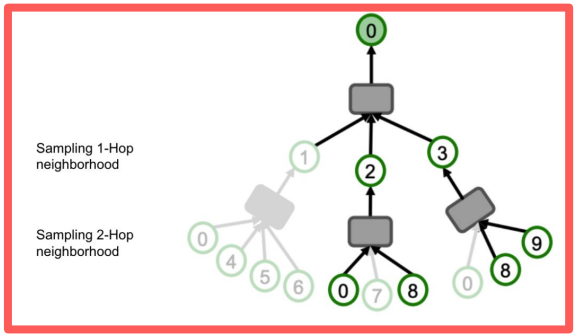
Feature Vector 0 Target Node K Search Depth Sampling Neighbors



Neighborhood Sampling of input graph at search depth K=1



Feature Aggregation for the target node **0** at K=1 with sampling



GraphSAGE

Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

GraphSAGE

Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

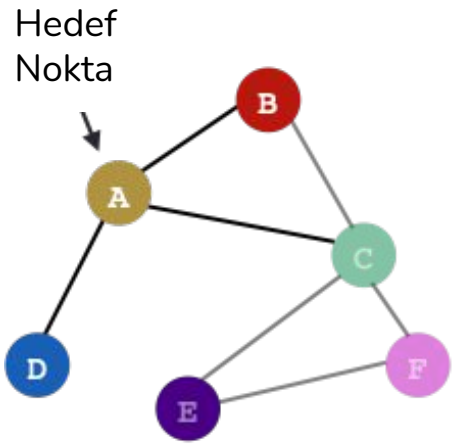
Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

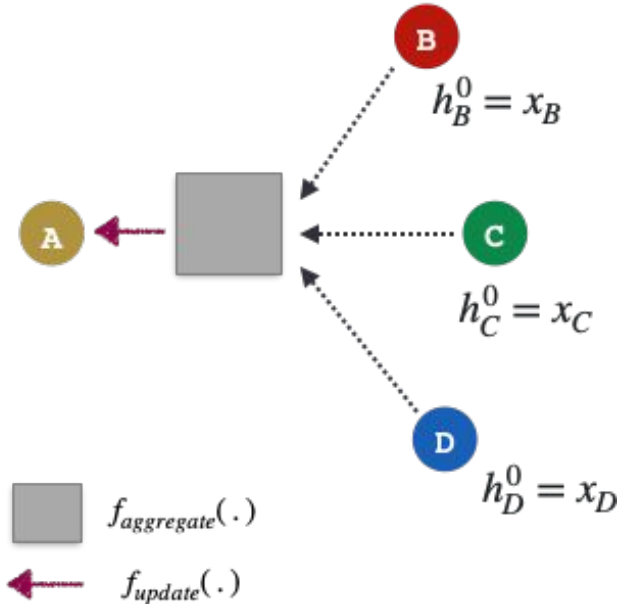
$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

GraphSAGE

Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu



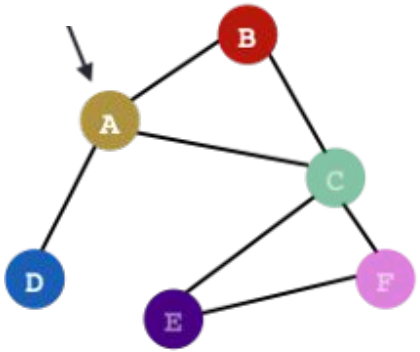


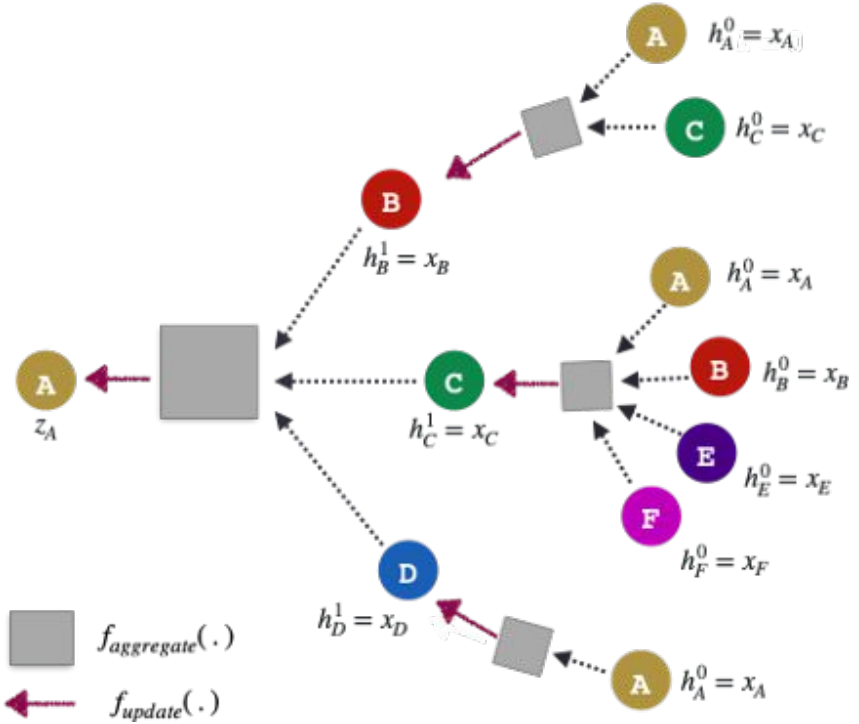
GraphSAGE

Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

Hedef Nokta





GraphSAGE

Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;

Her nokta için ilk çıktı vektörünü, özellik vektörüne eşitle.

for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$

Birleştirme ve güncelleme fonksiyonlarını çalıştır.

  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$

Her k katmanı için vektörleri normalize et.

end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

GraphSAGE

Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

(l x 2l) matris, (2l x 1) vektör çarpımı

GraphSAGE

Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

- Ortalama

$$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}).$$

- LSTM
- Pooling

$$\text{AGGREGATE}_k^{\text{pool}} = \max(\{\sigma\left(\mathbf{W}_{\text{pool}}\mathbf{h}_u^{k-1} + \mathbf{b}\right), \forall u_i \in \mathcal{N}(v)\})$$

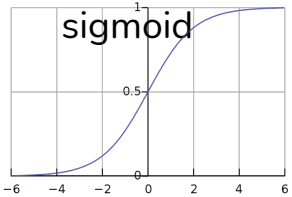
GraphSAGE

Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

- Geri besleme:
  - **Güdümlü**: Çapraz düzensizlik kayıp fonksiyonu
  - **Güdümsüz**: Noktaların çizge üzerindeki yakınlıklarına göre oluşturulan bir fonksiyon.

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log\left(\sigma\left(\mathbf{z}_u^\top \mathbf{z}_v\right)\right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma\left(-\mathbf{z}_u^\top \mathbf{z}_{v_n}\right)\right)$$



GraphSAGE

Inductive Representation Learning on Large Graphs

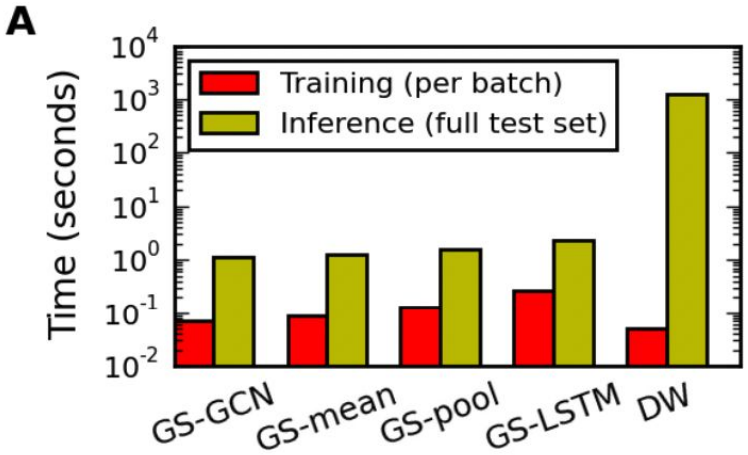
William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

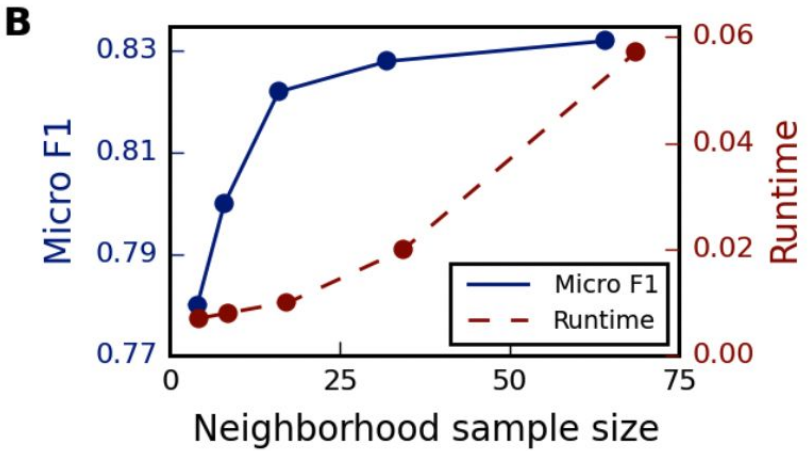
	Citation		Reddit		PPI	
Name	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1
Random	0.206	0.206	0.043	0.042	0.396	0.396
Raw features	0.575	0.575	0.585	0.585	0.422	0.422
DeepWalk	0.565	0.565	0.324	0.324	—	—
DeepWalk + features	0.701	0.701	0.691	0.691	—	—
GraphSAGE-GCN	0.742	0.772	**0.908**	0.930	0.465	0.500
GraphSAGE-mean	0.778	0.820	0.897	0.950	0.486	0.598
GraphSAGE-LSTM	0.788	0.832	**0.907**	**0.954**	0.482	**0.612**
GraphSAGE-pool	**0.798**	**0.839**	0.892	0.948	**0.502**	0.600
% gain over feat.	39%	46%	55%	63%	19%	45%

GraphSAGE

Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu





GraphSAGE

Örnek bir koda bakalım
(3. defter)

Çizge Analizi: Makina Öğrenmesi

Model	Neighborhood Aggregation $\mathbf{h}_v^{\ell+1}$
NN4G [88]	$\sigma\left(\mathbf{w}^{\ell+1^T}\mathbf{x}_v + \sum_{i=0}^{\ell}\sum_{c_k\in\mathcal{C}}\sum_{u\in\mathcal{N}_v^{c_k}} w_{c_k}^i * \mathbf{h}_u^i\right)$
GNN [104]	$\sum_{u\in\mathcal{N}_v} MLP^{\ell+1}\left(\mathbf{x}_u, \mathbf{x}_v, \mathbf{a}_{uv}, \mathbf{h}_u^\ell\right)$
GraphESN [44]	$\sigma\left(\mathbf{W}^{\ell+1}\mathbf{x}_u + \hat{\mathbf{W}}^{\ell+1}[\mathbf{h}_{u_1}^\ell, \ldots, \mathbf{h}_{u_{\mathcal{N}_v}}^\ell]\right)$
GCN [72]	$\sigma\left(\mathbf{W}^{\ell+1}\sum_{u\in\mathcal{N}(v)}\mathbf{L}_{vu}\mathbf{h}_u^\ell\right)$
GAT [120]	$\sigma\left(\sum_{u\in\mathcal{N}_v}\alpha_{uv}^{\ell+1} * \mathbf{W}^{\ell+1}\mathbf{h}_u\right)$
ECC [111]	$\sigma\left(\frac{1}{\lvert\mathcal{N}_v\rvert}\sum_{u\in\mathcal{N}_v} MLP^{\ell+1}(\mathbf{a}_{uv})^T\mathbf{h}_u^\ell\right)$
R-GCN [105]	$\sigma\left(\sum_{c_k\in\mathcal{C}}\sum_{u\in\mathcal{N}_v^{c_k}}\frac{1}{\lvert\mathcal{N}_v^{c_k}\rvert}\mathbf{W}_{c_k}^{\ell+1}\mathbf{h}_u^\ell + \mathbf{W}^{\ell+1}\mathbf{h}_v^\ell\right)$
GraphSAGE [54]	$\sigma\left(\mathbf{W}^{\ell+1}(\frac{1}{\lvert\mathcal{N}_v\rvert}[\mathbf{h}_v^\ell, \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^\ell])\right)$
CGMM [3]	$\sum_{i=0}^{\ell} w^i * \left(\sum_{c_k\in\mathcal{C}} w_{c_k}^i * \left(\frac{1}{\lvert\mathcal{N}_v^{c_k}\rvert}\sum_{u\in\mathcal{N}_v^{c_k}}\mathbf{h}_u^i\right)\right)$
GIN [131]	$MLP^{\ell+1}\left((1+\epsilon^{\ell+1})\mathbf{h}_v^\ell + \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^\ell\right)$

A Gentle Introduction to Deep Learning for Graphs Davide Bacciua , Federico Erricaa, Alessio Michelia, Marco Poddaa, arxiv, 15 June 2020

Çizge Analizi: Makina Öğrenmesi

Model	Neighborhood Aggregation $\mathbf{h}_v^{\ell+1}$
NN4G [88]	$\sigma\left(\mathbf{w}^{\ell+1^T}\mathbf{x}_v + \sum_{i=0}^{\ell}\sum_{c_k\in\mathcal{C}}\sum_{u\in\mathcal{N}_v^{c_k}} w_{c_k}^i * \mathbf{h}_u^i\right)$
GNN [104]	$\sum_{u\in\mathcal{N}_v} MLP^{\ell+1}\left(\mathbf{x}_u, \mathbf{x}_v, \mathbf{a}_{uv}, \mathbf{h}_u^\ell\right)$
GraphESN [44]	$\sigma\left(\mathbf{W}^{\ell+1}\mathbf{x}_u + \hat{\mathbf{W}}^{\ell+1}[\mathbf{h}_{u_1}^\ell, \ldots, \mathbf{h}_{u_{\mathcal{N}_v}}^\ell]\right)$
GCN [72]	$\sigma\left(\mathbf{W}^{\ell+1}\sum_{u\in\mathcal{N}(v)}\mathbf{L}_{vu}\mathbf{h}_u^\ell\right)$
GAT [120]	$\sigma\left(\sum_{u\in\mathcal{N}_v}\alpha_{uv}^{\ell+1} * \mathbf{W}^{\ell+1}\mathbf{h}_u\right)$
ECC [111]	$\sigma\left(\frac{1}{\lvert\mathcal{N}_v\rvert}\sum_{u\in\mathcal{N}_v} MLP^{\ell+1}(\mathbf{a}_{uv})^T\mathbf{h}_u^\ell\right)$
R-GCN [105]	$\sigma\left(\sum_{c_k\in\mathcal{C}}\sum_{u\in\mathcal{N}_v^{c_k}}\frac{1}{\lvert\mathcal{N}_v^{c_k}\rvert}\mathbf{W}_{c_k}^{\ell+1}\mathbf{h}_u^\ell + \mathbf{W}^{\ell+1}\mathbf{h}_v^\ell\right)$
GraphSAGE [54]	$\sigma\left(\mathbf{W}^{\ell+1}(\frac{1}{\lvert\mathcal{N}_v\rvert}[\mathbf{h}_v^\ell, \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^\ell])\right)$
CGMM [3]	$\sum_{i=0}^{\ell} w^i * \left(\sum_{c_k\in\mathcal{C}} w_{c_k}^i * \left(\frac{1}{\lvert\mathcal{N}_v^{c_k}\rvert}\sum_{u\in\mathcal{N}_v^{c_k}}\mathbf{h}_u^i\right)\right)$
GIN [131]	$MLP^{\ell+1}\left((1+\epsilon^{\ell+1})\mathbf{h}_v^\ell + \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^\ell\right)$

A Gentle Introduction to Deep Learning for Graphs Davide Bacciua , Federico Erricaa, Alessio Michelia, Marco Poddaa, arxiv, 15 June 2020

Çizge Analizi: Makina Öğrenmesi

Model	Neighborhood Aggregation $\mathbf{h}_v^{\ell+1}$
NN4G [88]	$\sigma\left(\mathbf{w}^{\ell+1^T}\mathbf{x}_v + \sum_{i=0}^{\ell}\sum_{c_k\in\mathcal{C}}\sum_{u\in\mathcal{N}_v^{c_k}} w_{c_k}^i * \mathbf{h}_u^i\right)$
GNN [104]	$\sum_{u\in\mathcal{N}_v} MLP^{\ell+1}\left(\mathbf{x}_u, \mathbf{x}_v, \mathbf{a}_{uv}, \mathbf{h}_u^\ell\right)$
GraphESN [44]	$\sigma\left(\mathbf{W}^{\ell+1}\mathbf{x}_u + \hat{\mathbf{W}}^{\ell+1}[\mathbf{h}_{u_1}^\ell, \ldots, \mathbf{h}_{u_{\mathcal{N}_v}}^\ell]\right)$
GCN [72]	$\sigma\left(\mathbf{W}^{\ell+1}\sum_{u\in\mathcal{N}(v)}\mathbf{L}_{vu}\mathbf{h}_u^\ell\right)$
GAT [120]	$\sigma\left(\sum_{u\in\mathcal{N}_v}\alpha_{uv}^{\ell+1} * \mathbf{W}^{\ell+1}\mathbf{h}_u\right)$
ECC [111]	$\sigma\left(\frac{1}{\lvert\mathcal{N}_v\rvert}\sum_{u\in\mathcal{N}_v} MLP^{\ell+1}(\mathbf{a}_{uv})^T\mathbf{h}_u^\ell\right)$
R-GCN [105]	$\sigma\left(\sum_{c_k\in\mathcal{C}}\sum_{u\in\mathcal{N}_v^{c_k}}\frac{1}{\lvert\mathcal{N}_v^{c_k}\rvert}\mathbf{W}_{c_k}^{\ell+1}\mathbf{h}_u^\ell + \mathbf{W}^{\ell+1}\mathbf{h}_v^\ell\right)$
GraphSAGE [54]	$\sigma\left(\mathbf{W}^{\ell+1}(\frac{1}{\lvert\mathcal{N}_v\rvert}[\mathbf{h}_v^\ell, \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^\ell])\right)$
CGMM [3]	$\sum_{i=0}^{\ell} w^i * \left(\sum_{c_k\in\mathcal{C}} w_{c_k}^i * \left(\frac{1}{\lvert\mathcal{N}_v^{c_k}\rvert}\sum_{u\in\mathcal{N}_v^{c_k}}\mathbf{h}_u^i\right)\right)$
GIN [131]	$MLP^{\ell+1}\left((1+\epsilon^{\ell+1})\mathbf{h}_v^\ell + \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^\ell\right)$

A Gentle Introduction to Deep Learning for Graphs Davide Bacciua , Federico Erricaa, Alessio Michelia, Marco Poddaa, arxiv, 15 June 2020

Çizge Dikkat Ağları

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

- GCN'lerin aksine, GAT modeli (dolaylı olarak) aynı komşuluktaki farklı komşulara farklı önemlerin atanmasına izin verir. Bu da model kapasitesinde bir sıçrama sağlar.
- Ayrıca, öğrenilmiş dikkat ağırlıkları yorumlanabilirlikte faydalar sağlayabilir. **Örneğin**: makine çevirisi alanı (Bahdanau vd. (2015) nitel analizi).

GAT [120]	$\sigma\left(\sum_{u \in \mathcal{N}_v} \alpha_{uv}^{\ell+1} * \mathbf{W}^{\ell+1} \mathbf{h}_u\right)$

Çizge Dikkat Ağları

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

Girdi: $\mathbf{h} = \{\overline{h}_1, \overline{h}_2, \ldots, \overline{h}_n\}$ $\quad \overline{h}_i \in \mathbb{R}^F$ - nokta özellikleri

Çıktı: $\mathbf{h}' = \{\overline{h'}_1, \overline{h'}_2, \ldots, \overline{h'}_n\}$ $\quad \overline{h'}_i \in \mathbb{R}^{F'}$ - yeni nokta özellikleri

Çizge Dikkat Ağları

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

Girdi: $\mathbf{h} = \{\overline{h}_1, \overline{h}_2, \ldots, \overline{h}_n\}$ $\quad \overline{h}_i \in \mathbb{R}^F$ - nokta özellikleri

Çıktı: $\mathbf{h}' = \{\overline{h'}_1, \overline{h'}_2, \ldots, \overline{h'}_n\}$ $\quad \overline{h'}_i \in \mathbb{R}^{F'}$ - yeni nokta özellikleri

Adım 1
Doğrusal transformasyon

$$\mathbf{W}.\overline{h}_i \qquad \mathbf{W} \in \mathbb{R}^{F' \times F}$$

Çizge Dikkat Ağları

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

Girdi: $\mathbf{h} = \{\overline{h}_1, \overline{h}_2, \ldots, \overline{h}_n\}$ $\quad \overline{h}_i \in \mathbb{R}^F$ - nokta özellikleri

Çıktı: $\mathbf{h}' = \{\overline{h'}_1, \overline{h'}_2, \ldots, \overline{h'}_n\}$ $\quad \overline{h'}_i \in \mathbb{R}^{F'}$ - yeni nokta özellikleri

Adım 2
Dikkat Mekanizması

Tek katmanlı bir sinir ağı

$$a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$$

$$e_{i,j} = a(\mathbf{W}.\overline{h}_i, \mathbf{W}.\overline{h}_j)$$

Çizge Dikkat Ağları

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

Girdi: $\mathbf{h} = \{\overline{h}_1, \overline{h}_2, \ldots, \overline{h}_n\}$ $\quad \overline{h}_i \in \mathbb{R}^F$ - nokta özellikleri

Çıktı: $\mathbf{h}' = \{\overline{h'}_1, \overline{h'}_2, \ldots, \overline{h'}_n\}$ $\quad \overline{h'}_i \in \mathbb{R}^{F'}$ - yeni nokta özellikleri

Adım 2
Dikkat Mekanizması

$$a : \mathbb{R}^{F'} \times \mathbb{R}^F \rightarrow \mathbb{R}$$

Nokta j'nin nokta i için (normalizasyon öncesi) önemi $e_{i,j} = a(\mathbf{W}.\overline{h}_i, \mathbf{W}.\overline{h}_j)$

Çizge Dikkat Ağları

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

Girdi: $\mathbf{h} = \{\overline{h}_1, \overline{h}_2, \ldots, \overline{h}_n\}$ $\quad \overline{h}_i \in \mathbb{R}^F$ - nokta özellikleri

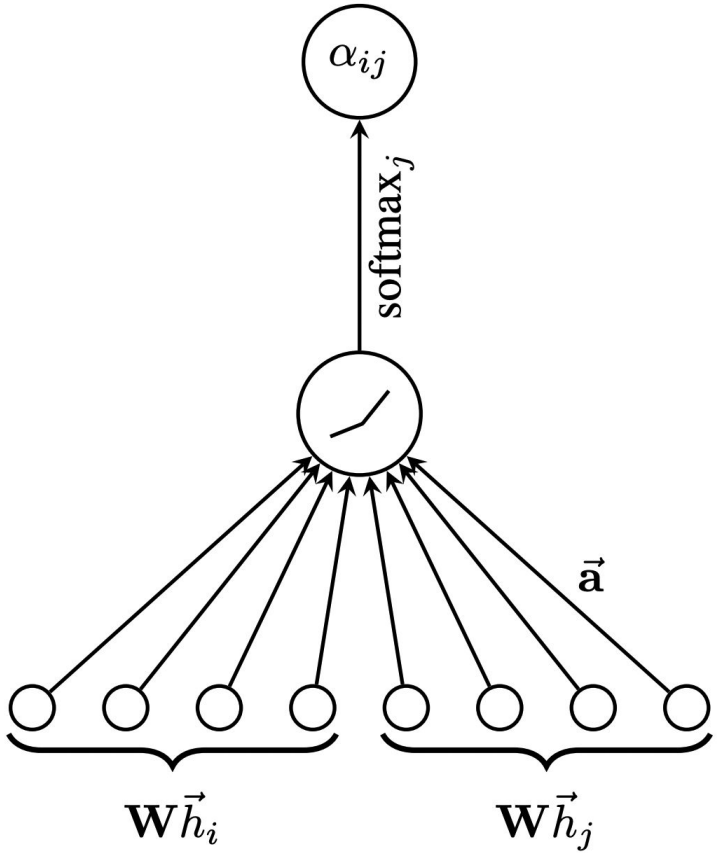
Çıktı: $\mathbf{h}' = \{\overline{h'}_1, \overline{h'}_2, \ldots, \overline{h'}_n\}$ $\quad \overline{h'}_i \in \mathbb{R}^{F'}$ - yeni nokta özellikleri

Adım 3
Normalizasyon

Normalizasyon sonrası değerler - (i için j üzerinden bir olasılık dağılımı...)

$$\alpha_{i,j} = \mathbf{softmax}_j(e_{i,j}) = \frac{exp(e_{i,j})}{\sum_{k \in N(i)} exp(e_{i,k})}$$

Çizge Dikkat Ağları



GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

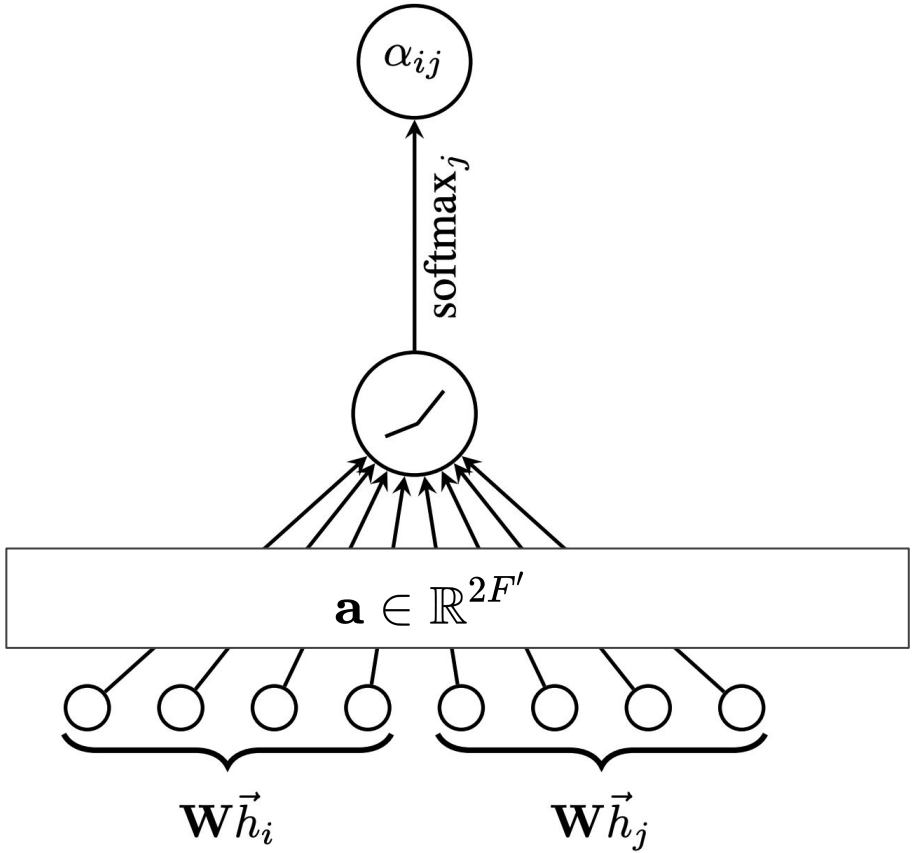
Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

Çizge Dikkat Ağları



GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

Çizge Dikkat Ağları

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

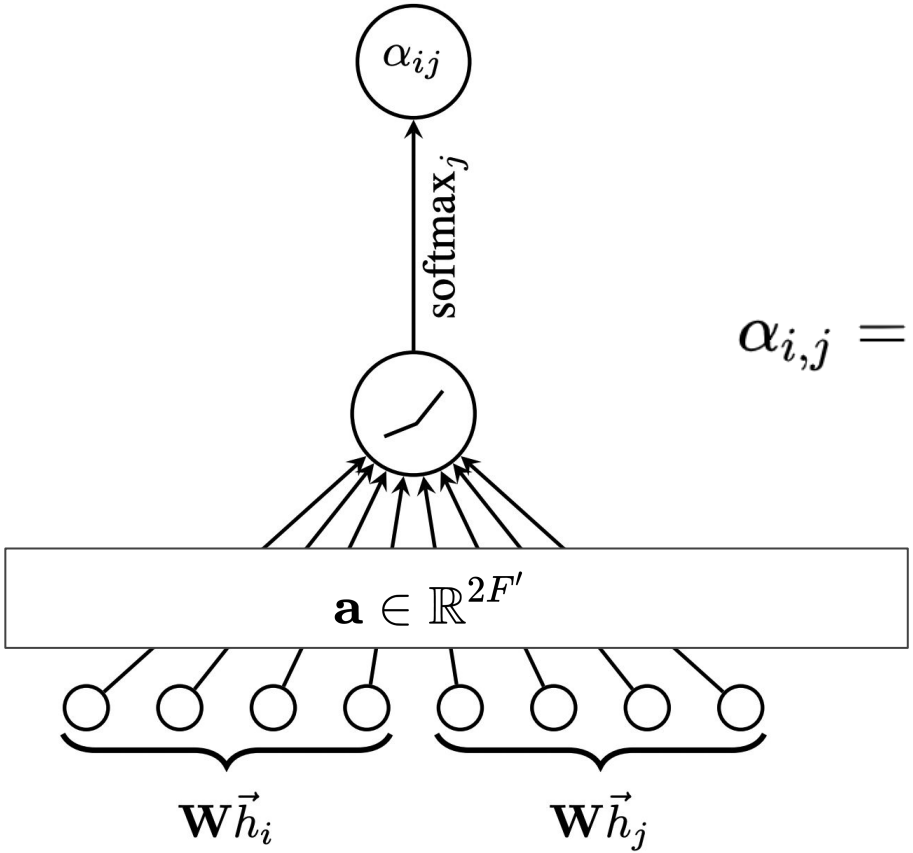
Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

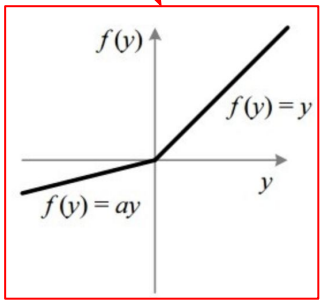
Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com



$$\alpha_{i,j} = \frac{exp(LeakyReLU(\mathbf{a}[\mathbf{W}\bar{h}_i|\mathbf{W}\bar{h}_j]))}{\sum_{k\in N(i)} exp(LeakyReLU(\mathbf{a}[\mathbf{W}\bar{h}_i|\mathbf{W}\bar{h}_k]))}$$



Çizge Dikkat Ağları

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

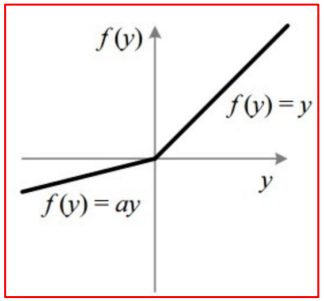
Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

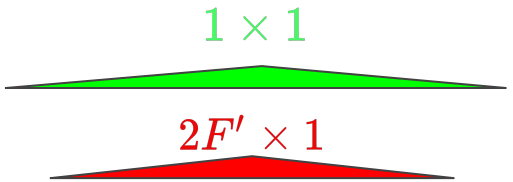
Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com





$$\alpha_{i,j} = \frac{exp(LeakyReLU(\mathbf{a}[\mathbf{W}\overline{h}_i|\mathbf{W}\overline{h}_j]))}{\sum_{k\in N(i)} exp(LeakyReLU(\mathbf{a}[\mathbf{W}\overline{h}_i|\mathbf{W}\overline{h}_k]))}$$

Çizge Dikkat Ağları

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

Girdi: $\mathbf{h} = \{\overline{h}_1, \overline{h}_2, \ldots, \overline{h}_n\}$ $\quad \overline{h}_i \in \mathbb{R}^F$ - nokta özellikleri

Çıktı: $\mathbf{h}' = \{\overline{h'}_1, \overline{h'}_2, \ldots, \overline{h'}_n\}$ $\quad \overline{h'}_i \in \mathbb{R}^{F'}$ - yeni nokta özellikleri

Adım 4

GAT/GNN katmanının elde edilmesi

$$\overline{h'}_i^{\ell+1} = \sigma\left(\sum_{j \in N(i)} \alpha_{i,j} \mathbf{W}^{\ell} \overline{h'}_j^{\ell}\right)$$

Çizge Dikkat Ağları

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

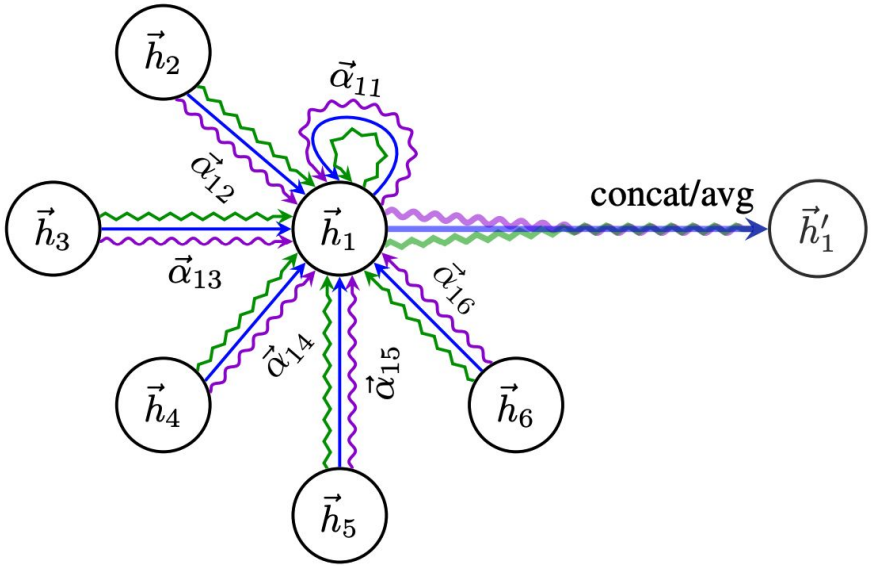
Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com



Sürecin daha stabil hale getirilmesi için yazarlar **çoklu dikkat mekanizması** yöntemini kullanmayı önermişlerdir. Bu yöntemde birbirinden bağımsız *K* dikkat mekanizması kullanılır.

$$\overline{h'}_i = \Pi_{k=1}^{K} \sigma\left(\sum_{j \in N(i)} \alpha_{i,j}^{k} \mathbf{W}^{k} \overline{h'}_j\right)$$

Ara katmanlarda birleştirme, son, çıktı katmanında ortalama.

Çizge Dikkat Ağları

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

Inductive

Method	**PPI**
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	$\mathbf{0.973} \pm 0.002$

Çizge Dikkat Ağları

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

Transductive

Method	**Cora**	**Citeseer**	**Pubmed**
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	$81.7 \pm 0.5\%$	—	$78.8 \pm 0.3\%$
GCN-64*	$81.4 \pm 0.5\%$	$70.9 \pm 0.5\%$	**79.0** $\pm 0.3\%$
GAT (ours)	**83.0** $\pm 0.7\%$	**72.5** $\pm 0.7\%$	**79.0** $\pm 0.3\%$

Çizge Dikkat Ağları

GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

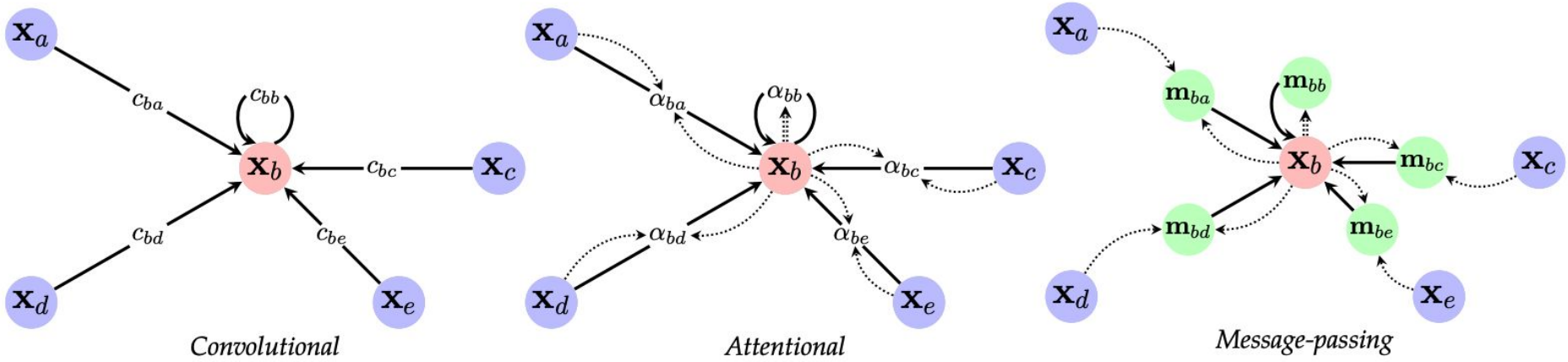
Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

- Hesaplama açısından oldukça verimlidir: dikkat mekanizması tüm kenarlarda ve çıktı özelliklerinin hesaplanması tüm noktalarda paralel hale getirilebilir.
- Öz bileşimler veya benzer maliyetli matris işlemleri gerekmez.
- Çok sayıda dikkat mekanizması uygulamak, depolama ve parametre güncelleme gereksinimlerini **K** parametresine bağlı olarak çoğaltır.
- Fakat bütün mekanizmalar tamamen bağımsızdır ve paralel hale getirilmelidir (getirilebilir).

Mesaj Geçiş Ağları



M. M. Bronstein, J. Bruna, T. Cohen, P. Velickovic, Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges (2021) , arxiv.

Bu derste bahsedilen çizge sinir ağlarını ortak bir yapı altında sınıflandırma işlemini, bir mesaj geçiş fonksiyonu ile yapabiliriz

Mesaj Geçiş Ağları

PyTorch geometric

$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \square_{j \in \mathcal{N}(i)} \, \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right) \right)$$

- $\mathbf{x}_i^{(k)}$: i noktasının k katmanındaki özellik vektörü.
- $\mathbf{x}_i^{(k-1)}$: i noktasının k-1 katmanındaki özellik vektörü.
- $\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}$: i noktasının ve ona komşu j noktalarının k-1 katmanındaki özellik vektörleri.
- $\mathbf{e}_{j,i}$: i noktası için kenar özellik vektörleri.

Mesaj Geçiş Ağları

PyTorch geometric

güncelleme: türevi alınabilir bir fonksiyon.

birleştirme: türevi alınabilir, sıra bağımsız bir fonksiyon (ortalama, toplam vb).

mesaj: türevi alınabilir bir fonksiyon.

$$\mathbf{x}_i^{(k)} = \gamma^{(k)}\left(\mathbf{x}_i^{(k-1)}, \square_{j \in \mathcal{N}(i)} \phi^{(k)}\left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i}\right)\right)$$

i noktasının k katmanındaki özellik vektörü.

i noktasının k-1 katmanındaki özellik vektörü.

i noktasının ve ona komşu j noktalarının k-1 katmanındaki özellik vektörleri.

i noktası için kenar özellik vektörleri.

Mesaj Geçiş Ağları

PyTorch geometric

$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \square_{j \in \mathcal{N}(i)} \, \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right) \right)$$

mesaj: türevi alınabilir bir fonksiyon.

- `MessagePassing.message(...)` : Constructs messages to node i in analogy to ϕ for each edge in $(j,i) \in \mathcal{E}$ if `flow="source_to_target"` and $(i,j) \in \mathcal{E}$ if `flow="target_to_source"` . Can take any argument which was initially passed to `propagate()` . In addition, tensors passed to `propagate()` can be mapped to the respective nodes i and j by appending `_i` or `_j` to the variable name, .*e.g.* `x_i` and `x_j` . Note that we generally refer to i as the central nodes that aggregates information, and refer to j as the neighboring nodes, since this is the most common notation.

Mesaj Geçiş Ağları

PyTorch geometric

$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \square_{j \in \mathcal{N}(i)} \, \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right) \right)$$

birleştirme: türevi alınabilir, sıra bağımsız bir fonksiyon (ortalama, toplam vb).

- `MessagePassing(aggr="add", flow="source_to_target", node_dim=-2)` : Defines the aggregation scheme to use (`"add"` , `"mean"` or `"max"`) and the flow direction of message passing (either `"source_to_target"` or `"target_to_source"`). Furthermore, the `node_dim` attribute indicates along which axis to propagate.

Mesaj Geçiş Ağları

PyTorch geometric

$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \square_{j \in \mathcal{N}(i)} \, \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right) \right)$$

güncelleme: türevi alınabilir bir fonksiyon.

- `MessagePassing.update(aggr_out, ...)` : Updates node embeddings in analogy to γ for each node $i \in \mathcal{V}$. Takes in the output of aggregation as first argument and any argument which was initially passed to `propagate()` .

Mesaj Geçiş Ağları

PyTorch geometric

$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \square_{j \in \mathcal{N}(i)} \, \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right) \right)$$

- `MessagePassing.propagate(edge_index, size=None, **kwargs)` : The initial call to start propagating messages. Takes in the edge indices and all additional data which is needed to construct messages and to update node embeddings. Note that `propagate()` is not limited to exchange messages in square adjacency matrices of shape `[N, N]` only, but can also exchange messages in general sparse assignment matrices, *.e.g.*, bipartite graphs, of shape `[N, M]` by passing `size=(N, M)` as an additional argument. If set to `None`, the assignment matrix is assumed to be a square matrix. For bipartite graphs with two independent sets of nodes and indices, and each set holding its own information, this split can be marked by passing the information as a tuple, *e.g.* `x=(x_N, x_M)`.

Mesaj Geçiş Ağları

PyTorch geometric

GCN katmanı: $\mathbf{x}_i^{(k)} = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\deg(i)} \cdot \sqrt{\deg(j)}} \cdot \left(\mathbf{\Theta} \cdot \mathbf{x}_j^{(k-1)} \right),$

$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \square_{j \in \mathcal{N}(i)} \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right) \right)$$

Mesaj Geçiş Ağları

PyTorch geometric

$$\text{GCN katmanı: } \mathbf{x}_i^{(k)} = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\deg(i)} \cdot \sqrt{\deg(j)}} \cdot \left(\mathbf{\Theta} \cdot \mathbf{x}_j^{(k-1)} \right),$$

1. Komşuluk matrisine döngüleri ekleme
2. Doğrusal transformasyon
3. Normalizasyon katsayılarının hesaplanması
4. Normalizasyon işlemi.
5. Birleştirme işlemi (toplam)

```
import torch
from torch_geometric.nn import MessagePassing
from torch_geometric.utils import add_self_loops, degree

class GCNConv(MessagePassing):
    def __init__(self, in_channels, out_channels):
        super(GCNConv, self).__init__(aggr='add')  # "Add" aggregation (Step 5).
        self.lin = torch.nn.Linear(in_channels, out_channels)
```

Mesaj Geçiş Ağları

PyTorch geometric

$$\text{GCN katmanı: } \mathbf{x}_i^{(k)} = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\deg(i)} \cdot \sqrt{\deg(j)}} \cdot \left(\mathbf{\Theta} \cdot \mathbf{x}_j^{(k-1)} \right),$$

1. Komşuluk matrisine döngüleri ekleme
2. Doğrusal transformasyon
3. Normalizasyon katsayılarının hesaplanması
4. Normalizasyon işlemi.
5. Birleştirme işlemi (toplam)

```
def forward(self, x, edge_index):
        # x has shape [N, in_channels]
        # edge_index has shape [2, E]

        # Step 1: Add self-loops to the adjacency matrix.
        edge_index, _ = add_self_loops(edge_index, num_nodes=x.size(0))

        # Step 2: Linearly transform node feature matrix.
        x = self.lin(x)

        # Step 3: Compute normalization.
        row, col = edge_index
        deg = degree(col, x.size(0), dtype=x.dtype)
        deg_inv_sqrt = deg.pow(-0.5)
        deg_inv_sqrt[deg_inv_sqrt == float('inf')] = 0
        norm = deg_inv_sqrt[row] * deg_inv_sqrt[col]

        # Step 4-5: Start propagating messages.
          return self.propagate(edge_index, x=x, norm=norm)
```

Mesaj Geçiş Ağları

PyTorch geometric

$$\text{GCN katmanı: } \mathbf{x}_i^{(k)} = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\deg(i)} \cdot \sqrt{\deg(j)}} \cdot \left(\mathbf{\Theta} \cdot \mathbf{x}_j^{(k-1)} \right),$$

1. Komşuluk matrisine döngüleri ekleme
2. Doğrusal transformasyon
3. Normalizasyon katsayılarının hesaplanması
4. Normalizasyon işlemi.
5. Birleştirme işlemi (toplam)

```
def message(self, x_j, norm):
        # x_j has shape [E, out_channels]

        # Step 4: Normalize node features.
        return norm.view(-1, 1) * x_j
```

Mesaj Geçiş Ağları

PyTorch geometric

$$\text{GCN katmanı: } \mathbf{x}_i^{(k)} = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\deg(i)} \cdot \sqrt{\deg(j)}} \cdot \left(\mathbf{\Theta} \cdot \mathbf{x}_j^{(k-1)} \right),$$

1. Komşuluk matrisine döngüleri ekleme
2. Doğrusal transformasyon
3. Normalizasyon katsayılarının hesaplanması
4. Normalizasyon işlemi.
5. Birleştirme işlemi (toplam)

```
def message(self, x_j, norm):
        # x_j has shape [E, out_channels]

        # Step 4: Normalize node features.
        return norm.view(-1, 1) * x_j
```

Mesaj Geçiş Ağları

Örnek kodlara bakalım
(4., 5., ve 6. defter)

TRUBA Üzerinde GNN

Model: GINE - molecular property prediction
Yöntem: DataParallel
Veri Kümesi: OGB

Cluster	Nodes	GPUs	Time per epoch (s)	Cluster	Nodes	GPUs	Time per epoch (s)	Cluster	Nodes	GPUs	Time per epoch (s)
Palamut-cuda	1	1	6.721	Akya-cuda	1	1	7.470	Barbun-cuda	1	1	7.869
		2	3.664			2	5.193			2	4.149
	2	1	4.595		2	1	5.825		2	1	20.701
		2	2.612			2	3.611			2	17.776
	3	1	3.213		3	1	4.224		3	1	29.282
		2	1.782			2	2.480			2	20.300
	4	1	2.507		4	1	3.357		4	1	28.362
		2	1.403			2	1.978			2	16.114

OGB Verisi

```python
import os
import argparse
from sys import argv
import time

import torch
import torch.nn.functional as F
from torch.nn import Sequential, Linear, ReLU, BatchNorm1d as BatchNorm
import torch.multiprocessing as mp
import torch.distributed as dist
from torch.nn.parallel import DistributedDataParallel
from torch.utils.data.distributed import DistributedSampler

from torch_geometric.nn import GINEConv, global_mean_pool
from torch_geometric.data import DataLoader
import torch_geometric.transforms as T

from ogb.graphproppred import PygGraphPropPredDataset as Dataset, Evaluator
from ogb.graphproppred.mol_encoder import AtomEncoder, BondEncoder


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--root', default='dataset/', type=str,
                        help='The root directory where the dataset is to be downloaded')
    args = parser.parse_args()
    dataset_name = 'ogbg-molhiv'
    Dataset(dataset_name, args.root, pre_transform=T.ToSparseTensor())
```

Kaynaklar

[1] https://docs.google.com/presentation/d/1k7nfUyDJfk6Vel0mu5Oxc0svlxppl-hOifLfOHy0ABE/edit#slide=id.p13

[2] https://towardsdatascience.com/an-introduction-to-graph-neural-network-gnn-for-analysing-structured-data-afce79f4cfdc

[3] https://medium.com/analytics-vidhya/ohmygraphs-graphsage-and-inductive-representation-learning-ea26d2835331

[4] https://towardsdatascience.com/over-smoothing-issue-in-graph-neural-network-bddc8fbc2472

[5]https://sachinsharma9780.medium.com/a-comprehensive-case-study-of-graphsage-algorithm-with-hands-on-experience-using-pytorchgeometric-6fc631ab1067