Grafové neuronové siete

Úvod

Konvolučné neuronové siete sú v dnešnej dobe už známe a často používané na obrazových dátach, kde dosahujú veľmi dobré výsledky. Preto sa neskôr začali aplikovať aj na grafové dáta. Obrázok si môžeme predstaviť, ako homogénny graf, kde každý vrchol reprezentuje pixel a hrany sú susedia daného pixelu. Všeobecné grafy môžu mať ďaleko komplexnejšiu štruktúru, teda je aj konvolučná vrstva komplikovanejšia. To však umožňuje riešiť širšiu škálu problémov, a preto našli grafové neurónové siete svoje uplatnenie aj v oblasti fyziky kondenzovaných látok.

Popis úlohy

Cieľom práce boli nasledovné 3 body:

1. Vytvorenie rešerše súčasného stavu literatúry na tému grafových neurónových sietí a ich využitia v oblasti kondenzovaných látok.
2. Vytvorenie jednoduchého programu implementujúceho grafovú neurónovú sieť pomocou jazyku Python a balíka TensorFlow
3. Analýza možností využiteľnosti grafových neurónových sietí v aktuálnom výskume magnetických fázových prechodov na katedre fyziky kondenzovaných látok

Úlohou práce bolo natrénovať grafovú neurónovú sieť na odhadovanie vlastností systémov častíc. Jednalo sa o regresné úlohy, kde mala sieť predikovať celkovú energiu systému. V projekte som pracoval s dvoma fyzikálnymi modelmi na popisovanie magnetických materiálov:

1. *Isingov model*: opisuje sústavu interagujúcich častíc so spinmi s hodnotami $\{-1, 1\}$. V grafe je interakcia medzi časticami reprezentovaná, ako hrana s nejakou váhou. Každý vrchol grafu opisuje práve jednu časticu.
2. *Heisenbergov model* funguje veľmi podobne, ako Isingov s tým, že každý spin je reprezentovaný, ako 3-dimenzionálny jednotkový vektor.

Pre oba modely sme počítali energiu, ktorá je určená Hamiltoniánom:

$$\sum_{<i,j>} J_{i,j}\sigma_i\sigma_j$$

kde $< i, j >$ je interakcia medzi najbližšími susedmi, J_{ij} je výmenná interakcia medzi i-tym a j-tym spinom a σ_i, σ_j, sú hodnoty spinu ± 1. Dôvod na použitie Isingovho modelu bol ten, že je o niečo jednoduchší. Preto sa hodilo sieť vyskúšať na jednoduchej úlohe a neskôr ju ďalej rozšíriť. V prípade Heisenbergovho modelu sme chceli overiť, či sieť dokáže problém zovšeobecniť a pracovať aj s viacero dimenziami. Hamiltonián pre Heisenbergov model vyzerá nasledovne: $\sum_{<i,j>} J_{i,j}\mathbf{s}_i \cdot \mathbf{s}_j$ kde oba vektory spinov sú jednotkové, teda platí $|\mathbf{s}_i| = |\mathbf{s}_j| = 1$.

Rešerš

Kovolučná vrstva

Zaveďme notácie pre graf $G = (V, E)$ na ktorom opíšem aktualizovanie parametrov. Nech každému $v_i \in V$ je priradený vektor h_i, ktorý obsahuje skryté príznaky neurónovej siete. Spojitosť grafu je reprezentovaná, ako matica susednosti A kde $(v_i, v_j) \in E \rightarrow A_{ij} = 1$ opačne 0. Potom ešte treba definovať maticu W,

čo je matica trénovateľných váh, ktorú môžeme chápať, ako bežnú lineárnu plne prepojenú vrstvu. Formy predikcie, ktoré sa dajú robiť na grafoch sú nasledovné:

1. Predikcia na vrcholoch: $f(h_i)$
2. Predikcia na celom grafe: $f(\sum_i h_i)$
3. Predikcia spojení: $f(h_i, h_j, e_{ij})$

V prípade toho projektu chceme predikovať vlastnosť, ktorá sa týka celého grafu. Zvyčajne sa na konci siete pridáva poolingová vrstva na zmenšenie výstupu. V tomto projekte sme používali average pooling alebo sum pooling.

Aktualizácia parametrov

Všeobecnú aktualizáciu skrytých príznakov vrcholov s aktivačnou funkciou *ReLu* značenou, ako σ môžeme formulovať: $H' = \sigma(AHW)$. Maticové násobenie AH efektívne skombinuje informácie z okolia daného vrcholu do jedného vektoru. Lineárna vrstva reprezentovaná W sa aplikuje pri každej konvolúcii. Umožňuje použiť znalosti z deep learningu a vytvoriť komplexnejšie modely. Na tejto vrstve je potrebné opraviť dve veci:

1. Ak v grafe nie je slučka tak sieť nebude pracovať s informáciou v totožnom vrchole. To sa dá vyriešiť jednoducho tak, že maticu susednosti prenásobím identitou. Budem používať $\overline{A} = AI$
2. Ak pri trénovaní opakovane násobím maticou susednosti, tak čísla môžu narásť do extrémnych hodnôt, ktoré nebude možné reprezentovať v bežných dátových typoch počítača. Preto sa oplatí celý výraz ešte normalizovať pomocou diagonálnej matice $D_{ii} = \sum_{j=0} \overline{A_{ij}}$.

Finálne pravidlo na aktualizovanie skrytých váh bude:

$$H' = \sigma(D^{-\frac{1}{2}}\overline{A}D^{-\frac{1}{2}}XW)$$

Tým sme práve odvodili známu vrstvu GCNConv [1]. Je to jedna z najčastejšie používaných vrstiev na grafové neurónové siete pre jednoduchosť a efektivitu. Vrstva GCN však priamo nepodporuje grafy s váženými hranami. Na to sa často používa MPNN (Message Passing Neural Network), ktorá je zložitejšia a dokáže rozpoznať aj komplexnejšie vzťahy. To má za následok aj veľkú výpočtovú náročnosť preto sa dá MPNN použiť len na menšie grafy. Alternatíva ktorú môžeme zvoliť medzi týmito dvoma vrstvami je GAT (Graph Attention Network), ktorá používa attention mechanism na pridelenie váh jednotlivým vrcholom v grafe. Tieto váhy hovoria, ako veľmi sa dva vrcholy navzájom ovplyvňujú. To umožňuje GAT flexibilnejšie a presnejšie zohľadňovať informácie z okolitých vrcholov a prispôsobiť sa rôznym grafovým dátam. [2]

Dostupné knižnice

Medzi momentálne dostupné knižnice na prácu s grafovými neurónovými sieťami patria: Pytorch Geometric, Deep Graph Library, Graph Nets alebo TensorFlow GNN. Pôvodným plánom projektu bolo natrénovať sieť v Tensorflow GNN, ktorá má širokú škálu funkcionalít. Knižnica je postavená veľmi robustne ale mal som problém sa v nej zorientovať. Napokon som sa rozhodol projekt programovať Pytorch Geometric, pretože má neporovnateľne lepšiu dokumentáciu aj s názornými príkladmi.

Implementácia

Generovanie dát

Pre túto úlohu bolo najvhodnejšie dáta generovať, čo dávalo prakticky neobmedzenú trénovaciu množinu. Sieť som trénoval na dvoch typoch grafov: na ceste a na mriežke s periodickými okrajovými podmienkami. Pod takouto mriežkou si môžeme predstaviť mriežku $N \times N$ a množinu vrcholov V s pridanými hranami medzi

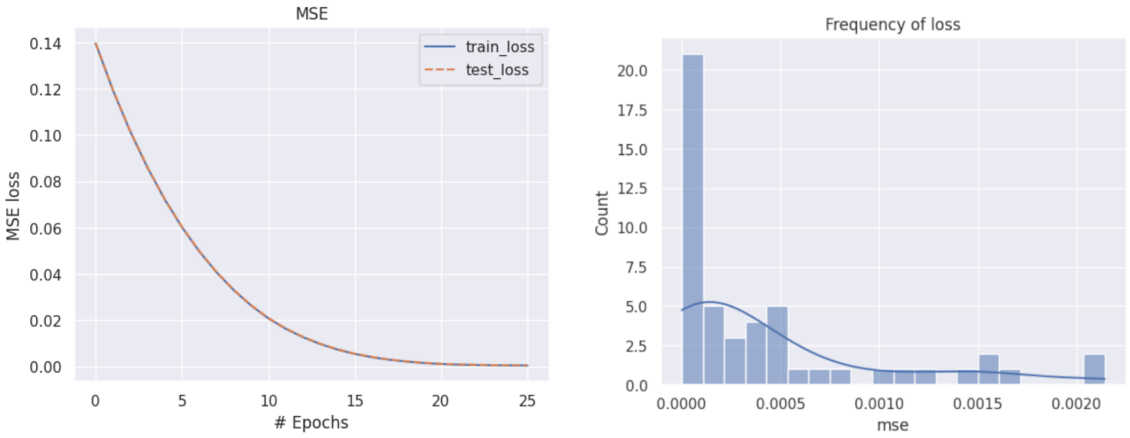
okrajovými vrcholmi. Teda pre vrchol $V_{0,0}$ by sme pridali hrany $(V_{0,0}, V_{0,n}), (V_{0,0}, V_{n,0}), (V_{0,0}, V_{n,n})$. Všetky hodnoty váh boli generované z normálneho rozdelenia s nulovým priemerom a jednotkovou štandardnou odchýlkou pomocou knižnice random v jazyku Python. Na Isingovom modeli som vyberal kladný alebo záporný spin z Bernoulliho distribúcie a v Heisenbergovom modeli boli vektory spinu vyberané pomocou unifromne náhodne generovaných uhlov, ktoré udávali jeho smer.

Architektúra siete

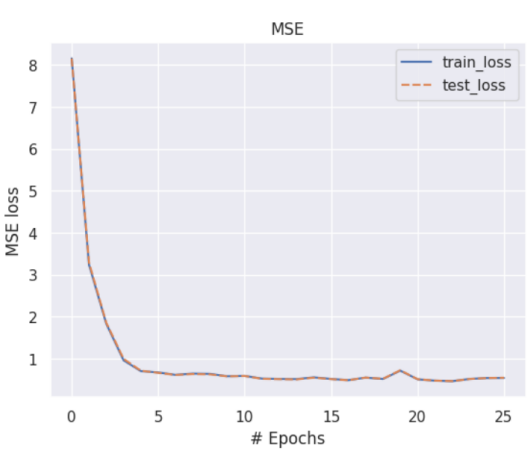
V prípade Isingovho modelu bol počet vstupných príznakov 1 a pre Heisenbergov model 3, pretože spin vrcholu bol reprezentovaný 3D vektorom. V oboch prípadoch mali výstupné parametre vrcholov dimenziu 1. Všetky siete, ktoré som trénoval pozostávali s nejakého počtu konvolučných vrstiev za ktorou nasledovali plne prepojené vrstvy a na konci poolingová vrstva, pričom najlepšie fungoval sum pooling. S veľkosťou siete a počtami parametrov som už experimentoval. Napríklad pre Isingov model úplne stačila sieť s jednou konvolučnou a jednou plne prepojenou vrstvou, ktoré mali 8 skrytých príznakov. Pre Heisenbergov to už nestačilo a musel som pridať počet vrstiev aby som dosiahol dobré výsledky.

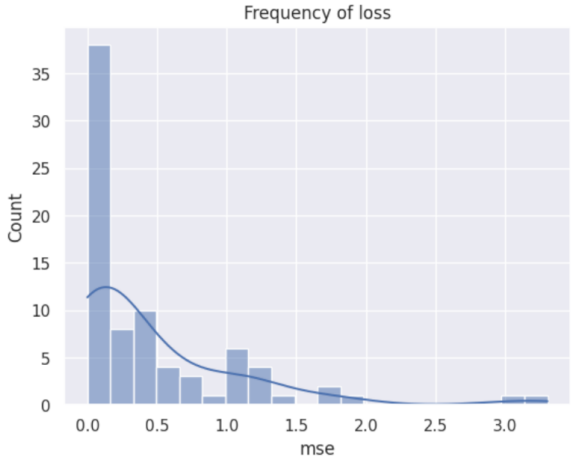
Výsledky

Trénovanie na Isingovom modeli s datasetom veľkosti 2500 grafov, kde každý mal 100 vrcholov trvalo na 25 epoch na bežnom notebooku 15-20 sekúnd. Pričom loss dosahoval okolo 10^{-4}.



Trénovanie na Isingovom modeli s datasetom veľkosti 1000 grafov opať so 100 vrcholmi trvalo na 25 epoch na bežnom notebooku 90 sekúnd. Najlepší loss sa mi podarilo dosiahnuť okolo 10^{-1}.





Záver

Na tomto konkrétnom prípade sme boli schopný natrénovať sieť s vhodnou presnosťou. Je však aj potrebné spomenúť, že sme sa snažili naučiť model pomerne jednoduchý vzťah, a preto by bolo potrebné učenie odskúšať na komplexnejších úlohách.

Pri potenciálnom pokračovaní by som opäť zvolil Python, kvôli širokej dostupnosti balíčkov v oblasti strojového učenia. Väčšina z nich je implementovaná v nízko úrovňových jazykoch, teda efektivita trénovania nie je problém. Balíčky, ktoré by boli vhodné sú určite Pytorch Geometric a Tensorflow GNN, pričom odporúčam začať s balíčkom Pytorch, ktorý je lepšie zdokumentovaný a ľahší na pochopenie. Výhodou grafových sietí je, že dokážu lepšie pracovať so štruktúrou dát a vyjadriť interakcie medzi nimi, čo môže byť užitočné napríklad pri skúmaní spomínaných magnetických fázových prechodov. Vo všeobecnosti sú komplexnejšie, ako bežné neurónové siete. Okrem toho sú aj jednoduchšie interpretovateľné, čo má za následok, že sa dá ľahšie skúmať, ako sa model správa a na základe čoho robí svoje predikcie. Samozrejme grafové siete sú aplikovateľné na menšiu množinu úloh avšak verím, že pri správnom formulovaní problému môžu posunúť výskum vo fyzike dopredu.

Literatúra

[1] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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