Disease-gene interaction prediction with graph neural networks

Dokumentáció

Rocket Team

Ábrók László Patrik (JPWF8N)

Hudák János (CP4EIQ)

2024. 12. 08.

1. Bevezetés

Az elmúlt évek biológiai kutatásainak köszönhetően a génekről és betegségekről alkotott tudásunk forradalmi változáson ment keresztül. Ez a tudás - amely a DISGENET [1] adatbázisának köszönhetően számunkra is rendelkezésre állt - akár arra is felhasználható, hogy mélytanulás segítségével mesterséges intelligencia modelleket tanítsunk be a betegségek és gének közötti kapcsolat felismerésére. Választott féléves feladatunkban pontosan erre a feladatra vállalkoztunk; a mélytanulás egy speciális eszközével, gráf neurális hálózatokkal közelítettük meg a problémát.

Féléves munkánk során egy variációs gráf autoenkóder [2] (VGAE, Variational Graph Auto-Encoder) alapú megoldást készítettünk el PyTorch [3] és Pytorch [4] Geometric alapokon. Megoldásunkat manuálisan és automatizáló eszközök használatával is javítottuk, új rétegek definiálásával, illetve az Optuna [5] eszköz alkalmazásával. Az egyszerű fejlesztés és kiértékelés érdekében az elkészült alkalmazást konténerizáltuk Docker [6] segítségével, valamint a Gradio [7] használatával készítettünk hozzá egy felhasználói interfészt is.

2. Adatfeldolgozás

A modell betanításához szükséges adatokat a DISGENET adatbázisából szereztük meg. Az ezt lehetővé tevő API kulcsokhoz az akadémiai licensz megigényelésével jutottunk hozzá.

Az adatokat a platform által publikált REST API interfész elérésével töltöttük le. A REST API-t először a Bruno [8] eszköz használatával teszteltük, majd tapasztalataink alapján egy Python klienst implementáltunk. Mivel az adatbegyűjtés nagy számú adatot érintett, az interfész elérésének rátáját limitáló beépített működést a kliensben külön lekezeltük.

A betegségekre, génekre és azok kapcsolatára vonatkozó adatokat a DISGENET Gene-Disease-Association sémájának segítségével gyűjtöttük be. Ezeket a betegségek ICD10 [9] által definiált kategóriáinak szisztematikus végigjárásával szereztük meg, melyre szintén saját implementációt készítettünk. Munkánk eredményeként egy 300 betegséget, 5024 gént és 10379 kapcsolatot tartalmazó adathalmaz jött létre.

Az adathalmazt heterogén gráfként képzeltük el, melyben a betegségek és a gének adták a gráf csúcsait, az élek pedig a közöttük meglévő kapcsolatokat. A

betegségekhez azok ICD10 kategóriáját, a HGNC [10] szimbólumokkal azonosított génekhez a DSI (Disease Specificity Index) és DPI (Disease Pleiotropy Index) értékeket, a kapcsolatokhoz az EI (Evidence Index) értéket használtuk fel. Amennyiben az utóbbiak nem álltak rendelkezésre, a definíciójuk szerinti értéktartomány legalsó értékét állítottuk be számukra.

A fentebb ismertetett adatokat a heterogén gráfnak megfelelően a PyTorch Geometric HeteroData osztálynak segítségével helyeztük el a DataModule modulunkban. A tanítás elősegítésének céljából a valódi élekhez "hamis éleket" is generáltunk, majd ezt követően az adatok hetven százalékából tanító, tizenöt-tizenöt százalékából pedig validációs és teszt adathalmazt hoztunk létre. Egy saját köztes osztály használatával a kezdetben felmerülő, adathalmazon értelmezett iterációs problémáinkat is kiküszöböltük.

Az adatok megszerzésével, feldolgozásával, tisztításával és az adatmodul elkészülésével következő lépésként a modell implementásával foglalkoztunk.

3. Modell és tanítás

A megoldásunk során a fent említett variációs gráf autoenkóder architektúrát alkalmaztuk az adatmodulból érkező heterogén adatokból való tanulásra. Az architektúra tartalmazza az általunk implementált enkóder és dekóder komponenseket, valamint azokat a rétegeket, melyek a betegségekhez és génekhez tartozó látens reprezentációba való leképezéshez szükségesek.

Az implementált osztályok:

- HeteroVGAE
    - Definiálja a heterogén variációs gráf autoenkóder modellt.
    - Paraméterei: in_channels_disease, in_channels_gene, out_channels, heterovgae_hidden_channels, encoder_hidden_channels
    - Komponensei: encoder, decoder, fc_mu_disease, fc_logvar_disease. fc_mu_gene, fc_logvar_gene
- Decoder
    - A forward_all függvény segítségével mátrixszorzást valósít meg a betegségek és gének látens reprezentációin.
- Encoder
    - Azért felel, hogy enkódolja a betegségekhez és génekhez tartozó bemeneti tulajdonságokat.

- Paraméterei: in_channels_disease, in_channels_gene, out_channels, encoder_hidden_channels.
- Rétegei: HeteroConv rétegeket használ. SAGEConv segítségével felhasználja a betegség-gén és gén-betegség éleket is. Az eredményeken összegzést hajtunk végre.

A tanításhoz Adam optimalizálót használtunk, annak stabil tanulása és adaptív tanulási sebessége miatt. A modell két paraméterét állítottuk; ezek egyike az learning_rate, mely meghatározza, hogyan módosítsuk a model paramétereit a veszteség (loss gradient) függvényében. Ezen felül beállítottuk a weight_decay-t is, mely L2 regularizációt adott az optimalizálónak. Ez segített elkerülni a túltanulást (overfitting) a nagyobb súlyok elhagyásával. A veszteség számításához a súlyozott igazság-mátrixunk alkalmazása mellett BCEWithLogitsLoss-t használtunk, amelynek nagy előnye, hogy egyesíti a bináris keresztentrópia (binary cross entropy, BCE) és szigmoid aktivációs függvényt.

A fentiek alkalmazásával, illetve a rétegek kézi bővítésével, vagy csökkentésével átlagosan 0,6 körüli ROC AUC (Receiver Operating Characteristic Area Under Curve) értéket értünk el. Ennek javítására a hiperparaméter optimalizációt végeztünk el.

4. Hiperparaméter optimalizáció

A modell elkészítése után hipermaraméter optimalizációt hajtottunk végre. A folyamatot az Optuna eszköz segítségével végeztük el. A paraméterek kiértékeléséhez az objective(trial) függvényt definiáltuk felül, melynek paraméterként egy DisgenetDataModule példányt adtunk át. Az objective függvényben a következő paramétereket optimalizáltuk, a feltüntetett tartományokban:

- learning_rate: 1e-5 – 1e-1
- heterovgae_hidden_channels: 8 – 32
- encoder_hidden_channels: 8 – 32
- weight_decay_param: 1e-5 – 1e-1

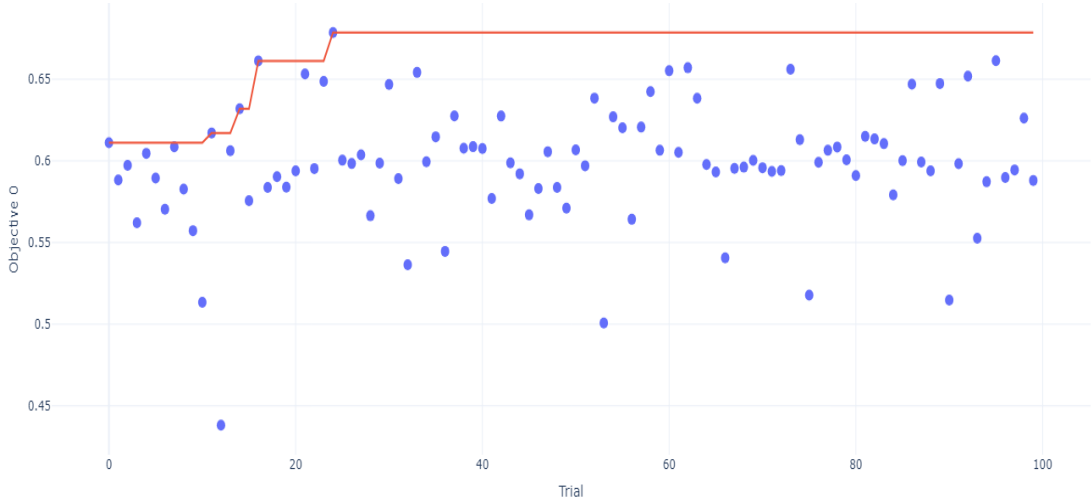
Az optimalizáció során a tanításhoz hasonlóan 200 epoch-ot használtunk. Az evaluate függvényben számított roc_auc értéket közvetlenül használtuk fel a paraméterek optimalizálására. Annak érdekében, hogy az optimalizáció hatékonyan kerüljön végrehajtásra, a tanulmányok (study) adatait SQLite adatbázisban tároltuk. Az adatbázis lehetővé tette, hogy az optimalizáció után megvizsgálhassuk a tanulmány

eredményeit. Az korábban felsorolt paraméterek 100-as trial értékkel a következő “best_param” értékeket eredményezték:

- learning_rate: 0.07299722265899901,
- heterovgae_hidden_channels: 29,
- encoder_hidden_channels: 13,
- weight_decay: 0.004328230364261425

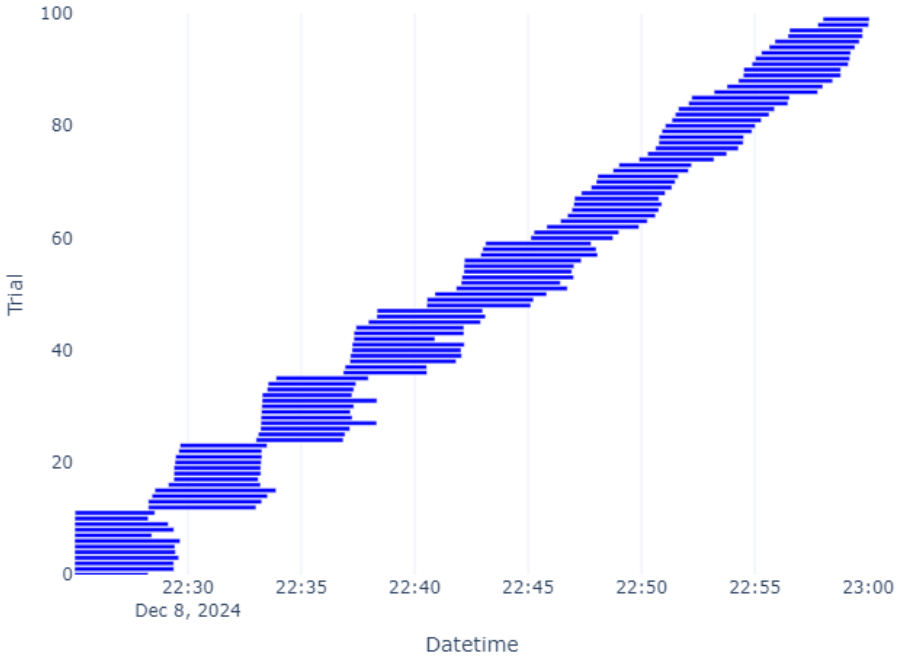
Az 1. ábrán látható érték az Optuna Dashboard-ról készült. Itt sikerült elérnünk a hozzávetőleg 0,67-es ROC AUC értéket. Az alábbi ábrán a kék pontok az egyes trial-hoz tartozó értékeket, míg a piros vonal az egyes trial-ok alatt legjobbnak tekintett értéket mutatja.

1. ábra: A legjobb próbálkozás értékei



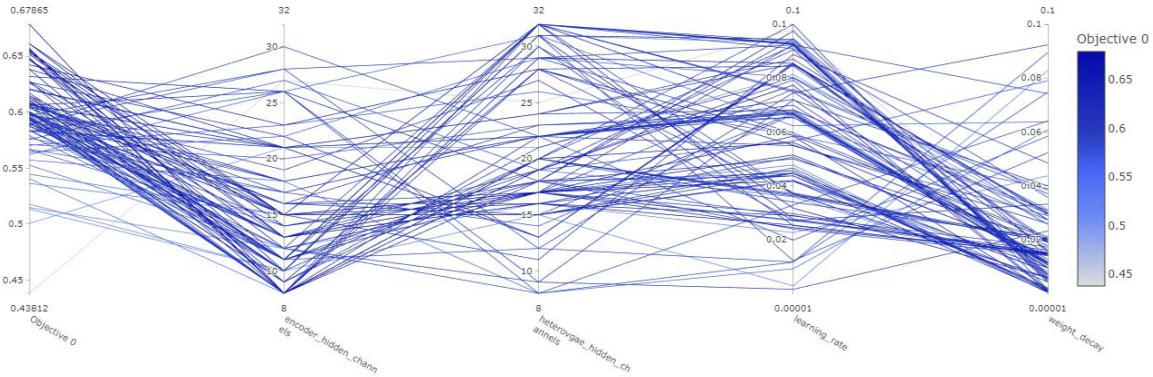
2. ábra: Az optimalizáció folyamatát bemutató ábra

Ahogy az a 2. ábrán is látható, a 100 próbálkozásból a 89. adta a legjobb eredményeket.



3. ábra: Próbálkozások párhuzamosított futása

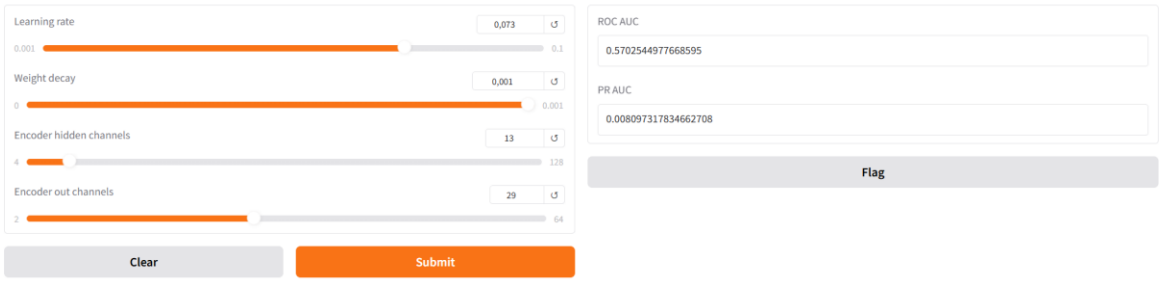
Ahogy a 3. ábrán látszik, a párhuzamosított teljes folyamat hozzávetőlegesen 25 percet vett igénybe. Ahogy pedig a 4. ábrán, a Hyperparameter Importance diagramon látszik, a tanítást leginkább az encoder_hidden_channels és a heterovgae_hidden_channels értékek befolyásolták. Az Optuna parallel coordinate plot-ot is generált, melyen az egyes optimalizálási próbálkozásokat lehetett áttekinteni.



4. ábra: Parallel coordiante plot

A dokumentáció írásának pillanatában a hiperparaméter optimalizációt futtató kód a 27-optimize-hyperparameters ágon található meg. A main.py futtatáskor lefut az optimalizációs szkript. Az eredmények a http://localhost:8080/-on található Optuna Dashboard-on érhetők el.

5. Felhasználói felület



5. ábra: Az elkészült Gradio felhasználói felület egy futtatás után

Az elkészült megoldásunkhoz a hiperparaméter optimalizációt követően egy felhasználói felületet is készítettünk a Gradio eszköz segítségével. Ahogy az 5. ábra bal oldalán látható, az optimalizáló learning_rate, weight_decay, valamint a variációs gráf autoenkóder encoder_hidden_channels és encoder_out_channels paramétereit lehetett beállítani. A “Submit” gomb megnyomását követően a jobb oldalon rövidesen megjelentek a ROC AUC és PR AUC (Precision-Recall Area Under Curve) értékek.

6. Összegzés

Féléves feladatunk során egy bioinformatikát és mélytanulást ötvöző izgalmas területen mélyedhettünk el. Munkánk eredményeként létrehoztunk egy gráf neurális hálózat alapú heterogén variációs gráf autoenkódert, amelyet mind manuálisan, mind automatizált eszközök használatával inkrementálisan javítottunk. Az optiomális kiértékelés érdekében konténerizált megoldásunkhoz egy felhasználói felületet is készítettünk. Munkánk során rengeteg hasznos tapasztalattal gazdagodtunk, és izgalommal teli várakozással tekintünk a jövőbeni hasonló feladatokra.

7. Irodalomjegyzék

[1] J. Piñero, J. M. Ramírez-Anguita, J. Saüch-Pitarch, F. Ronzano, E. Centeno, F. Sanz and L. I. Furlong, "The DisGeNET knowledge platform for disease genomics: 2019 update," *Nucleic Acids Research,* vol. 48, no. D1, p. D845–D855, 2020.

[2] T. N. Kipf and M. Welling, "Variational Graph Auto-Encoders," *arXiv:1611.07308 [stat.ML],* 2016.

[3] T. L. Foundation, "PyTorch," The Linux Foundation, [Online]. Available: https://pytorch.org/. [Accessed 8 12 2024].

[4] P. Team, "PyG Documentation," PyG Team, [Online]. Available: https://pytorch-geometric.readthedocs.io/en/latest/. [Accessed 8 12 2024].

[5] P. N. Inc., "Optuna - a hyperparameter optimization framework," Preferred Networks Inc., [Online]. Available: https://optuna.org/. [Accessed 8 12 2024].

[6] D. Inc., "Docker: Accelerated Container Application Development," Docker Inc., [Online]. Available: https://www.docker.com/. [Accessed 8 12 2024].

[7] H. F. Inc., "Gradio," Hugging Face Inc., [Online]. Available: https://www.gradio.app/. [Accessed 8 12 2024].

[8] B. S. Inc., "Bruno," Bruno Software Inc., [Online]. Available: https://www.usebruno.com/. [Accessed 8 12 2024].

[9] W. H. Organization, "ICD-10 Version:2019," World Health Organization, [Online]. Available: https://icd.who.int/browse10/2019/en. [Accessed 8 12 2024].

[10] HGNC, "Home | HUGO Gene Nomenclature Commitee," HGNC, [Online]. Available: https://www.genenames.org/. [Accessed 8 12 2024].

Munkánk során kódkiegészítéshez, illetve dokumentáló kommentek generálásához felhasználtunk LLM eszközöket, mint például Copilot-ot, vagy ChatGPT-t.