Identification of drug interactions with a graph autoencoder

(Gyógyszer kölcsönhatások azonosítása gráf autoenkóderrel)

line 1: 1st Péter István
line 2: *dept. name of organization (of Affiliation)*

line 1: 2nd Rancz Máté
line 2: *dept. name of organization (of Affiliation)*

line 1: 3rd Bilibok Bence
line 2: *dept. name of organization (of Affiliation))*

I. Abstract

The goal of this project is to realize a learning based on Variational Graph AutoEncoders. The motivation was the fact that certain medicines applied together can have adverse reaction on each other, meaning one drug can increase or decrease the effect of another. The following steps was followed to reach the wanted results: extract the features from the DrugBank [1] database, eliminate the validation and test edges from the graph, modelling the interaction graph with an autoencoder. The challenge of this project was to model the graph data, create the model so it will reaches an accuracy of at least 80%, to extract features and then to illustrate the results with ROC curve and AUC.

Keywords—Variational Graph AutoEncoder, features, drug, edges, graph, modell, effect

II. Összefoglaló

A projekt célja egy Variational Graph AutoEncoderen alapuló tanulás megvalósítása. A motiváció az volt a projekt megvalósítására, hogy bizonyos gyógyszerek együttes alkalmazása nemkívánatos reakciót válthat ki egymásra nézve, vagyis az egyik gyógyszer növelheti vagy csökkentheti a másik hatását. A kívánt eredmény elérése érdekében a következő lépések lettek alkalmazva: a jellemzők kinyerése a DrugBank [1] adatbázisból, a validációs és teszt élek eltávolítása a gráfból, az interakciós gráf modellezése autoencoderrel. A projekt kihívása a gráf adatok modellezése, a modell létrehozása volt úgy, hogy az legalább 80%-os pontosságot érjen el, hogy kivonja a jellemzőket, majd az eredményeket ROC görbével és AUC-val illusztrálja.

Kulcsszavak – Variational Graph AutoEncoder, jellemzők, gyógyszer, élek, gráf, modell, hatás

III. Introduction

Drug–drug interactions (DDIs) are an essential attention in each drug improvement and medical utility, specially for co-administered medicines. Whilst it'sfar important to perceive all possible DDIs during clinical trials, DDIs are often mentioned after the medication are accepted for clinical use, and they are a commonplace cause of damaging drug reactions (ADR) and increasing healthcare expenses. Computational prediction may additionally help in identifying potential DDIs in the course of medical trials.

IV. State of Art

Recent studies show that drugs have combined effects. These days this is a very interesting topic to study, so does many investigators in the last 40 years.

Their methods of generating and reading records have ch anged dramatically through the years but the primary problem has not. [1] Many different methods were considered to solve the question, including three-dimensional models for synergetic and antagonistic drug interactions in antiviral and anticancer chemotherapy.

Other articles explain the absorption of drugs and the basic concept how they interact, from a pharmacokinetic point of view, from these articles can be seen the extent of interactions, which is also important. [2]

In [3] another method is applied to analyze drug-drug interactions. A heterogeneous network-assisted inference (HNAI) framework to assist with the prediction of DDIs. First, was built a comprehensive DDI community that contained 6946 DDI pairs connecting 721 accepted capsules primarily based on DrugBank [1] facts. next, was calculated drug–drug pair similarities the usage of four functions: phenotypic similarity primarily based on a comprehensive drug–ADR community, healing similarity based totally at the drug Anatomical healing Chemical category device, chemical structural similarity from SMILES facts, and genomic similarity primarily based on a big drug–goal interplay network constructed the usage of the DrugBank [1] and therapeutic goal Database. sooner or later, then was implemented five predictive models within the HNAI framework: naive Bayes, selection tree, okay-nearest neighbor, logistic regression, and aid vector system, respectively.

V. Neural network structure

We used the Neural network architecture described in the paper [2] as Variational Graph Autoencoder (VGAE). Its first layer is a Graph Convolutional layer (we used the DGL implementation [https://docs.dgl.ai/en/0.9.x/generated/dgl.nn.pytorch.conv.GraphConv.html], it is a version that works only on direct neighbours of nodes), that transforms the features into a hidden representation, while not altering graph structure. At the otput of this layer, the architecture splits into two 'heads'.

These are both GCNs, the first one used to predict the mean of the latent space distribution, while the second head predicts the standard deviation. For each vertex, we sample a random vector with mean and standard deviation according to the output of the previous two layers, at each label. This is the output of the encoder.

The decoder does not have trainable parameters, it works by taking dot product of each i,j vertex pair, then if that product is lower than a threshold, it predicts them as not connected, otherwise as connected.

VI. Implementation

A. Data gathering and processing

1) Database

We needed to choose a dataset, which was easily accessible, contained enough data and it held information about the interactions between the drugs. For our project we used the Drugbank [1] website's online available database, which contains data about all the drugs which they found in the last 15 years. It was last updated on 2022.01.03 and it contained more than 40000 records.

The data we used was composed by two files. The first was the actual database, it contained everything about the drugs, and it was written in an .xml file. The second was a text file containing just the interactions between the drugs, each line was composed by two drug identifiers, between which was an interaction, but it wasn't described.

2) Data processing

The goal was to create a feature vector for each drug. There were two conditions, by which we chose out the properties we later used as features. The first was that the property's value could be represented with a vector of numbers. The second was that the property had to have meaningful data about the interactions between the drugs.

The first step was to examine the dataset and choose the properties which we will use as features. A problem was that a lot of the data inside the database were text, which explained different interactions, side effects and usage, which could have been useful, but we could not transform them into numbers. In the end we chose the drugs state, average mass, monoisotopic mass, classification, kingdom, melting point and molecular formula.

The second step was to create the features. For this we needed to parse the .xml file so it will be useable. We use the ElementTree.parse() function, which creates an ElementTree from the given xml file. From this ElementTree we extracted the data we needed and made it into a table with an identifier and 10 columns, all of which representing a feature. This table was saved as a .csv file for easier processing later.

The training process used this table and the text file containing all the interactions between the drugs.

B. Training

During training, we took the whole graph with the feature vectors of size 10 as labels, then removed all but 10% of the edges, 5% of which were validation edges, 5% were train edges, while the rest were test edges. This subgraph was the same in each iteration.

We experimented with manual hyperparameter tuning. We used the Adam optimizer with learning rate 0.05 and 100 epochs. Tuning the first hidden layer's output dimension, as well as the number of latent dimensions yielded results as well, the best results were attained with 64 as the first hidden layer's output size, and 32 as the latent representation's size.

On the contrary of our ROC and average precision result, our training accuracy reached a percentage of about 33% in general. This can be attributed to the setup of the model, hyperparameter optimalization for further interest would help this accuracy to be higher, and the number of features that were extracted.

C. Results and testing

The test and evaluation part of the project consisting mainly of two parts: the first parameter is the ROC curve, which was used to evaluate the model. The ROC curve stands for receiver operating characteristic curve, which is a graph showing the performance of classification models. Two parameter is plotted on the curve, which are True Positive Rate and False Positive Rate. True Positive Rate is defined as:

$$TPR = \frac{TP}{(TP+FN)} \quad (1)$$

The False Positive Rate is the same, but with false positive values, calculates the total False Positive values from all the values:

$$FPR = \frac{FN}{(TP+FN)} \quad (2)$$

The ROC curve plots FPR values on the 'x' axis, while the TPR value is on the 'y' axis of the system.

The AUC is the Area Under the ROC. This aggregates a measure of performance across all possible classification thresholds. AUC is meant to express the probability of that the model ranks a random positive example more highly than a random negative value. AUC ranges from 0 to 1, meaning that it can be expressed in terms of percentages as well. It measures how well predictions are ranked and has a huge advantage over some other evaluation metrics, because it is independent of the classification threshold.

In this case, the most important part is to find all interactions between drugs correctly if possible. It is predicted how precise is an interaction predicted by us. It is relevant for this project, because we want to focus on finding edges, that are there, then finding non-edges correctly. Having 84% for AUC value and 79% in this evaluation means that this was a successful project, taking in consideration that the model consists of only three convolutional layers and was trained with a 0.05 learning rate, on only 100 epochs, being resource saving as well.

In the future it can reach better results using a higher learning rate, adding some other layers, and training on more, than 100 epochs.

VII. Summary

The goal of our project was to find a way to predict if there is an interaction between different drugs from their properties. We used the Variational Graph Autoencoder for our neural network. From our database, we chose some properties which we deemed useful in finding interactions

and made them into feature vectors for our drugs. Using the graph which came with the database, depicting all known interactions between the drugs we trained our model. Basically, we removed some edges from the graph and tried to predict them correctly. In the end our results weren't great, mostly because our features weren't informative enough.

In the future better results can be achieved by exploring better graph sampling and more robust hyperparameter optimization. The majority of input features that were extracted, and we ended up using, were not deemed to be too informative, based on our intuition, but they were the only features that could easily be extracted as a numerical value. Other data, like molecule composition and spatial, are harder to convert into numbers, but solving this obstacle could lead to better feature quality, with higher predictive value.

VIII. References

[1] „Drugbank," [Online]. Available: https://go.drugbank.com. [Hozzáférés dátuma: 2 10 2022].

[2] M. W. Thomas N. Kipf, „Variational Graph Auto-Encoders," *arxiv.org*, p. arXiv:1611.07308, 2016.

[3] C. S. J. Mark N. Richard, „A three-dimensional model to analyse drug-drug interactions," *Antiviral Research*, %1. kötet14, pp. 181-205, 1990.

[4] R. Ad, Drug-drug interactions, CRC Press, 2019.

[5] Z. Z. Feixiong Cheng, „Machine learning-based prediction of drug–drug interactions by integrating drug phenotypic, therapeutic, chemical, and genomic properties," *Journal of the American Medical Informatics Association*, %1. kötet21, pp. 278-286, 2014.

[6] M. H. K. O. a. Y. T. Y. Yorozu, „Electron spectroscopy studies on magneto-optical media and plastic substrate interface," *IEEE Transl. J. Magn. Japan*, %1. kötet2, pp. 740-741, 1987.

[7] M. Young, The Technical Writer's Handbook, Mill Valley: University Science, 1989.