Identifying drug interactions with Graph Auto-Encoders

Gyógyszer kölcsönhatások azonosítása gráf autoenkóderrel

Balázs Tibor Morvay
CLT8ZP

László Kálmán Trautsch
CMLJKQ

Marcell Csikós
P78P08

***Abstract*— Nowadays, the application of deep learning plays a prominent role in medical fields, including the pharmaceutical industry. The primary purpose is to speed up research and make the work of researchers easier. Our team's chosen topic is the identification of adverse drug reactions using a graph autoencoder architecture. The drugs can be interpreted as the vertices of a graph, and the reactions between them as the edges of the graph. The usefulness of the topic is when a new drug is being developed, it can be decided which existing drugs may interact with it. Over the course of our work, we built a graph autoencoder architecture based on the PyTorch framework, which is able to restore the edges of the interaction graph with approximately 95% accuracy. In order to improve the results, we also used features extracted from the DrugBank database as input for the training. We integrated the solution with the TensorBoard platform in order to save and visualize the results and optimize the hyperparameters.**

***Kivonat*— A deep learning alkalmazása napjainkban orvosi területeken, azon belül is a gyógyszeriparban kiemelt szerepet kap. Az alkalmazások célja elsődlegesen a kutatások felgyorsítása, a kutatók munkájának megkönnyítése. Csapatunk választott témája adverz gyógyszerreakciók azonosítása gráf autoenkóder architektúra segítségével. A gyógyszerek egy gráf csúcsaiként, a köztük fennálló reakciók pedig a gráf éleiként értelmezhetők. A terület hasznossága, hogy új gyógyszerek fejlesztése esetén megmondható, mely létező gyógyszerekkel állhat fenn kölcsönhatás. Munkánk során egy PyTorch keretrendszerrel működő gráf autoenkóder architektúrát építettünk, amely körülbelül 95%-os pontossággal képes a kölcsönhatási gráf éleinek visszaállítására. Az eredmények javítása érdekében a DrugBank adatbázisból kinyert jellemzőket is felhasználtuk a tanítás bemeneteként. A megoldást integráltuk a TensorBoard platformmal az eredmények elmentése és vizualizálása, illetve a hiperparaméterek optimalizálása végett.**

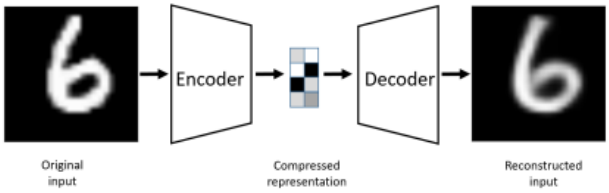
I. Related works

Drug interaction prediction is analogous to link prediction in graph-structured data. Link prediction has been extensively studied in recent years, with solutions using heuristics or scoring functions, like the Katz-index[2], Common Neighbor and Centrality based Parameterized Algorithm" (CCPA) [3], or using graph neural networks to find a heuristics [4]. Link prediction has also been studied in other fields than computer science, like in the paper [5] examining protein interaction predictions.

II. Model

The model is based on the Variational Graph Auto-Encoder implementation by Thomas N. Kipf and Max Welling. [1]

The general autoencoder architecture looks like the following:



1: Autoencoder architecture [8]

The encoder and decoder parts are represented by the model's *encoder()* and *decoder()* functions. Encoding is transforming the input's dimensionality to a compressed, „latent" representation. Decoding is transforming back to the original input dimensions. The error is the difference between the original input and the reconstructed one from the latent dimensions.

In the case of graphs, dimensionality is the graph itself and optionally additional features. When applying the model on a graph, only the decoder is used, and any new vertices added will have the predicted edges between each other and the original vertices.

The model has 3 PyTorch GraphConv layers, which are used for both encoding and decoding. Activation between the 1st and 2nd layers are ReLU, the others are linear.

III. Implementation

A. Dataset

We used the provided DrugBank dataset for adding features to the graphs nodes. This dataset contains the following fields:

- drugbank_id: id for the drug
- name: the name of the drug
- type: either biotech or small molecule

- groups: the group that the drug is in. There are 47 different groups in the dataset
- atc_codes: Anatomical Therapeutic Chemical codes. This code can be broken down, as seen later.
- categories: The categories that the drug is in.
- inchi: The International Chemical Identifier is a textual identifier for chemical substances, designed to provide a standard way to encode molecular information and to facilitate the search for such information in databases and on the web
- inchikey: inchi key
- description: textual description of the drug.
- aliases: aliases to the drug.
- logP: the predicted partition coefficient
- logS: the predicted solubility of the molecule
- psa: a descriptor, based on the polarized atoms of the molecule, that allows estimation of transport properties and of the passive molecular transport through membranes of the drug
- refractivity: the predicted molar refractivity of the molecule, which is strongly related to the volume of the molecules and to London dispersive forces that play crucial part in drug-receptor interactions
- polarizability: the predicted relative tendency of the electron cloud (charge distribution) of the molecule to be distorted by an external electric field
- pKa (strongest acidic): the strongest acidic pka value of the molecule
- pKa (strongest basic): the strongest basic pka value of the molecule
- Number of Rings: a calculation of the number of rings in the molecule

B. Fields used for training

Drugbank id
Ids are usually worthless for training models, so we ignore this field.

Name
The name field doesn't contain any valuable information to us, so we ignore this field.

Type
The drug's type can provide valuable information, so we use this value for training. We transformed the string values to integers (biotech → 0, small molecule → 1), so the model can interpret them.

Groups
We use this information for training, after transforming the string values to integers.

ATC codes
As described by WHO, this code is a concatenation of 5 different codes. We split the code into its components, and transform them to integers. We then use the 5 codes during training.

Categories
This field is very tricky to interpret, as there are ~6500 different values in the dataset. However, some values are very similar to each other, like Antibodies, Antibodies, Monoclonal and Antibodies, Monoclonal, Humanized. We chose to ignore this field in the dataset for now, because it is very challenging to extract meaningful information out of these categories (doing so would involve understanding these categories. For example, is Antibodies, Monoclonal a subcategory of Antibodies, or Monoclonal is a different category).

Inchi and Inchikey
We don't think that these values contain valuable information for the model.

Description
Textual description of the drug is not a valuable information for the model.

Aliases
We don't plan on using this field for training.

Numerical fields
We chose to use the fields logP, logS, psa, refractivity, polarizability, pKa_acidic, pKa_basic and num_rings as numerical fields for training. However, a considerable amount of data was missing from these values (for example only 9500 entries contained pKa values, out of the 14594 samples). To solve this problem, we looked for missing data imputation methods.

C. Missing data imputation

For missing data imputation, we used the GAIN algorithm, as described in [6]. This method uses the well-known GAN (Generative Adversarial Nets) framework. We examined the implementation of the algorithm on the UCI letter dataset [7], and adapted it to work on our missing numerical data.

D. Evaluation

Hyperparameter tuning was done using TensorBoard, with the following parameters and values:

- Number of epochs: 100, 200, 300
- Learning rate: 0.001, 0.01, 0.1
- Number of units in 1st hidden layer: 16, 32, 64
- Number of units in 2nd hidden layer: 16, 32, 64

Every combination of these values were run, so training and evaluation was run 81 times. The results of the hyperparameter tuning can be found in the repo under the ‘runs’ folder. The following can be said about the results:

- The lowest losses were about the 0.5 mark. These values were consistently achieved with the learning rate of 0.01. The other parameters were more diverse, but it can be seen that 100 epochs,

16 units for the 1st hidden layer and 64 units for the 2nd hidden layer were producing worse results generally.

- Reconstruction accuracy and ROC score was especially high with the highest learning rate (0.1). But other metrics were not as convincing, especially with the perfect or near-perfect accuracies (average precision of ~0.5, loss of ~50). Reconstruction accuracy can show good results when the other metrics do not, as it only gives how many of the test edges were successfully reconstructed, it does not take into account how many edges were reconstructed unnecessarily. So a model that reconstructs a large number of edges will definitely achieve good reconstruction accuracy.
- The maximum of the average precision score was about 0.90887. This can be considered as the most important metric for hyperparameter tuning, as the other metrics were also very good. High values of average precision are typically trainings with the learning rate of 0.01, 200 or 300 epochs, and 32 or 64 units in the 1st hidden layer.

With these in consideration, the best hyperparameter combination is 300 epochs, learning rate of 0.01, 64 units in the 1st, and 16 units in the 2nd hidden layer. The resulting metrics are:

- Loss: 0.51263
- ROC Score: 0.95359
- Average Precision Score: 0.90887
- Reconstruction Accuracy: 0.95359

IV. Conclusion and Future Work

We implemented a Variational Graph Auto-Encoder model for identifying drug-interactions. With our used feature extraction method and hyperparameter optimization, the trained model achieved high accuracy in all of the relevant metrics.

In the future, our model can be extended to use more features during training. One of the most important improvement potentials is the use of the category feature. From the many different category values it is possible to select a few that split the data the most, this way we can train the model on more relevant features.

References

[1] https://arxiv.org/abs/1611.07308
[2] https://arxiv.org/pdf/1912.06525.pdf
[3] https://www.nature.com/articles/s41598-019-57304-y
[4] https://proceedings.neurips.cc/paper/2018/file/53f0d7c537d99b3824f0f99d62ea2428-Paper.pdf
[5] https://www.biorxiv.org/content/10.1101/275529v1.full.pdf
[6] http://proceedings.mlr.press/v80/yoon18a/yoon18a.pdf
[7] https://github.com/jsyoon0823/GAIN
[8] https://arxiv.org/abs/2003.05991
[9]