Work report

Introduction  
Background

Deep learning algorithm  
Drug combination synergy for cancer cells (ramener du plus large aux cellules sur lesquelles on travaille)  
Sensitivity prediction  
Machine Learning models and techniques  
 DNN algorithms, Graphical models, Attention based algorithms, Matrix factorization, Tree based methodsi  
 Curb overfitting techniques  
 Active and transfer learning

Materials and Methods  
Materials  
 Oneil data [1] = We trained our main model on a the O’NEIL dataset which is regularly used in the task of synergy prediction. It is composed of a total of 22,737 series of experiments of 583 drug doublets lined up against 39 different cancer lines, on a 4x4 matrix of drug concentrations, which adds up to a total of 366,882 data samples available for training. From each line contained in the results - obtained via a high throughput screening (HTS) with an attenuated luminescent signal measurement technology [1], we extracted the cell line name, both drugs’ common names, their corresponding concentration levels given in µM, and viability level X/X0 which constituted our prediction target. Since obtaining the genetic profiles of cancer cells is costly, one cancer cell profile was missing from the cell database CCLE 2019 that we used, forcing us to filter out some entries. Furthermore, some cell names in O’NEIL were ambiguous and pointing to multiple cell IDs in the CCLE name mapping, so we had to get rid of another 7 cells too to avoid any imprecision. The final shape of our training set was 234,560 experiments on 31 unique cells.

Our test set was a series of experiments which had been previously produced for a related work in our lab[2]. The study aimed to fit a second order polynomial model on the response of breast cancer cells under the treatment of multiple drugs combination essays. We re-purposed the results of the Chou-Talalay method, used in the thesis to validate observations, because it specifically fitted the context of our task, i.e. confronting a matrix of experiments of drug doublets used on the breast tumor cell MDA-MB-231. This cell being absent from the O’NEIL dataset, the accuracy of our model with the results of the experimental data obtained from in vitro experiments, would characterize the ability to generalize on unseen genetic profiles. Despite the relatively small sample size of 24 configurations, the test set is ideal to directly measure the performance resulting from our different choices of implementation also because none of the drugs in the test set are present in O’NEIL. To maintain coherence between the training and test set, we preprocessed MDA-MB-231’s proliferation values obtained by our peer to transform them into the viability metric used in the O’NEIL dataset, defined by X/X0=exp(ln(4)\*(mu/muMax)), where mu and muMax are given in the biological model describing the number of cells with and without treatment – or control sample - respectively N(t)=N(t=0h)\*exp (mu\*t). Since proliferation is the ratio between the number of cells at screening time by the number at 0h, we transformed our test data via this simple relation:

Because the drug concentration levels were linearly spread out in matrices due to the usage of designs of experiments in both set-ups and given their scale close to the unit, we used them free of any upfront normalization and fed them directly to our model as numeric inputs.

To train our autoencoders for genetic cell profile and drug biological features, we respectively used multi-omics cell information from the CCLE study [3] and drugs enlisted in the well-known DrugComb database [4].

CCLE omics data:  
We extracted RNAseq gene expression profiles from the CCLE 2019 edition of 1450 cancer cell lines. To reduce down the consequent dimension of the training set, we first eliminated statistically insignificant attributes by filtering columns where the 3rd quantile was below 1.2, then removed the ones where the standard deviation did not reach one. The remaining 5011 columns were then shuffled into a dataset encoded to single precision to compensate for its large memory space. This cell data served as a look-up table for cell information on whichever cell present in the O’NEIL dataset.

DrugComb drugs and Molecular data:  
DrugComb [4] is a general drug database mostly used for cancer research. We extracted all drugs that had a record of an experiment, of monotherapy or as part of a combination, against MBA-MD-231. This allowed us to eliminate drugs completely unrelated to cancer therapy, where some interest had been shown about using these drugs for MBA-MD-231 at least once, while still preserving variance in the data since drugs are tested in many other scenarios. To this first stack we added the names of drugs present in O’NEIL and our testing set. We then retrieved the SMILE descriptor of each drug using the PubChem’s REST API, and finally extracted, with the python library RdKit, a concatenation of 1) all molecular descriptors available 2) Morgan and MACCS fingerprints translated into bit vectors. The dataset was also Z-score normalized as per usual practice, and entries represented a total of 673 drugs described in 2412 vectors. It served as our drug feature look-up table for training and testing.

Methods  
autoencoding  
In order to reduce the scale of computations, we chose to encode drug and cell data in latent vectors to reduce dimensionality while performing feature selection. The autoencoder architecture chosen relied on a series of fully connected perceptron layers followed by ReLU activations and dropout. The size of our latent representation is 256, which comes out as a close 10-fold decrease for drug features, and 20-fold for cell omics data. The goal here for encoders is to represent complex data structures such as topological drug descriptors and genetic profiles - which in fact are poorly adapted to neural networks [5], in two different latent spaces, with the highest fidelity with respect to task at hand (ideally though not enforced in the training of our autoencoders) and prevent overfitting on training samples.

To reach this goal while retaining the straightforward multi-layer perceptron (MLP) architecture, we implemented a training hyperparameter search routine with the use of Optuna, a python library, to estimate the best settings – according to minimal loss - for learning rates, L1 and L2 regularization parameters, dropout rates, and first layer number of units.

We searched the best hyperparameters for the training of our auto-encoders on subsets of the data look-up tables of drugs and cells, which included MDA-MB-231 and the 4 drugs present in the testing set.

In the second step, we set those parameters and trained our models on the complete look-up tables with Early Stopping (420 epochs for drug and 500 for cell encoders). Given the large dimensions of these tables, we ensured that encoders learned some sufficiently generalized encoding.

Model

The model that we used is an encoder-MLP architecture that first passes single drug features and features through either our drug or cell specialized encoder. From the previously trained auto-encoders, we detached the first series of layers up until the middle layer in the latent space, which formed our two encoders. The result of this first encoding segment is a scaled down concatenated 770-units long vector easing out the computation for regressor training. The obtained vector is then fed into our regressor, which are 3 series of 3 fully connected linear layers of similar sizes, each layer being adjoined to a dropout and a rectified linear unit (ReLU) activation. The series follow the scheme [256, 128, 64] for the number of output units. The sensitivity regression of the drug combination with respect to the given cell is estimated by a final linear layer of one scalar value topping up the model.

split strategy  
During training then testing time, we must make sure that we do not artificially boost the model accuracy by using a data splitting strategy that would allow the multi-input model to learn representations of the drug pairs present in all training, testing and validation sets. This is the reason we opted for a Leave-Pair -Out (LTO) cross-validation (CV) strategy which uniquely samples drug combinations and dispatches them into training and validation sets. In a Leave-Triplet-Out (LTO) CV, a certain number of testing samples would have the doublet of drug IDs (e.g., the Saracatinib/Rocilinostat pair, in the Saracatinib/Rocilinostat/MDA-MB-231 experiment) identical to other samples in the training set (e.g., the Saracatinib/Rocilinostat/MDA-MB-330 triplet) therefore previously learned by the model. Since it is impossible to ensure that individual drugs are uniquely assigned to each dataset, the LPO configuration allowed for minimal information leakage. This procedure is common in the drug combination scenario [5].

Metrics

To quantify the accuracy of our regressor, we computed on our validation set the mean squared error (MSE), the mean absolute error (MAE), and the Pearson correlation coefficient (PCC). While the two first metrics are widely used to assess the performance of machine learning algorithms which hence come as obvious, we chose to add the PCC because, rather than absolute errors, it rewards linear correlations more. Since our testing and training sets come from different real experimental settings, the PCC should value samples ranked in the right order more, even if marginal error between labels and sensitivity predictions subsist.

Plot explanation for PCC

Training and hyperparam

We trained the model with the Adam optimizer, a starting learning rate of 1e-3 and MSE for the loss function. We used Early Stopping strategy on an LPO 80/20-split validation set of the O’NEIL drug combination dataset, meaning we uniquely dispatched drug pairs along the 80/20 repartition. Input data is loaded from preprocessed cell and drug features. We also scheduled a 0.5 learning rate multiplier upon reaching plateaus of the epoch training MSE.

Results and discussion

auto

Plotting auto-encoders hyperparam tuning results + other combinations + search sample size

Result training in mse from output

Model

Plot 4 random validation graph plus values comparison with other models + (synergy prediction) 🡪 good learning show PCC MAE and MSE where

Plot testing graphs bad  
show that MAE MSE bad PCC still good

Transfer learning

Scatter plots PCC MSE MAE

Plot testing with and without transfer learning sara pd

Discussion

Conclusion

Different angles

Synergy

Batch norm