MECHANISMS OF ACTION (MOA) PREDICTION

OVERVIEW

DESCRIPTION

Kaggle Competition Overview: https://www.kaggle.com/c/lish-moa/overview

What is the Mechanism of Action (MoA) of a drug? And why is it important?

In the past, scientists derived drugs from natural products or were inspired by traditional remedies. Very common drugs, such as paracetamol, known in the US as acetaminophen, were put into clinical use decades before the biological mechanisms driving their pharmacological activities were understood. Today, with the advent of more powerful technologies, drug discovery has changed from the serendipitous approaches of the past to a more targeted model based on an understanding of the underlying biological mechanism of a disease. In this new framework, scientists seek to identify a protein target associated with a disease and develop a molecule that can modulate that protein target. As a shorthand to describe the biological activity of a given molecule, scientists assign a label referred to as mechanism-of-action or MoA for short.

How do we determine the MoAs of a new drug?

One approach is to treat a sample of human cells with the drug and then analyze the cellular responses with algorithms that search for similarity to known patterns in large genomic databases, such as libraries of gene expression or cell viability patterns of drugs with known MoAs.

In this competition, you will have access to a unique dataset that combines gene expression and cell viability data. The data is based on a new technology that measures simultaneously (within the same samples) human cells' responses to drugs in a pool of 100 different cell types (thus solving the problem of identifying ex-ante, which cell types are better suited for a given drug). In addition, you will have access to MoA annotations for more than 5,000 drugs in this dataset.

As is customary, the dataset has been split into testing and training subsets. Hence, your task is to use the training dataset to develop an algorithm that automatically labels each case in the test set as one or more MoA classes. Note that since drugs can have multiple MoA annotations, the task is formally a multi-label classification problem.

How to evaluate the accuracy of a solution?

Based on the MoA annotations, the accuracy of solutions will be evaluated on the average value of the logarithmic loss function applied to each drug-MoA annotation pair.

If successful, you'll help to develop an algorithm to predict a compound's MoA given its cellular signature, thus helping scientists advance the drug discovery process.

EVALUATION

For every sig_id you will be predicting the probability that the sample had a positive response for each <MoA> target. For N sig_id rows and M <MoA> targets, you will be making N×M predictions. Submissions are scored by the log loss:

$$\text{score} = -\frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\sum_{i=1}^{N}\left[y_{i,m}\log(\hat{y}_{i,m}) + (1-y_{i,m})\log(1-\hat{y}_{i,m})\right]$$

where:

- N is the number of sig_id observations in the test data (i=1,…,N)
- M is the number of scored MoA targets (m=1,…,M)
- $\hat{y}_{i,m}$ is the predicted probability of a positive MoA response for a sig_id
- $y_{i,m}$ is the ground truth, 1 for a positive response, 0 otherwise
- log() is the natural (base e) logarithm

DATA

In this competition, you will be predicting multiple targets of the Mechanism of Action (MoA) response(s) of different samples (sig_id), given various inputs such as gene expression data and cell viability data.

Two notes :

- the training data has an additional (optional) set of MoA labels that are *not* included in the test data and not used for scoring.
- the re-run dataset has approximately 4x the number of examples seen in the Public test.

Files

- train_features.csv - Features for the training set. Features g- signify gene expression data, and c- signify cell viability data. cp_type indicates samples treated with a compound (cp_vehicle) or with a control perturbation (ctrl_vehicle); control perturbations have no MoAs; cp_time and cp_dose indicate treatment duration (24, 48, 72 hours) and dose (high or low).
- train_targets_scored.csv - The binary MoA targets that are scored.
- train_targets_nonscored.csv - Additional (optional) binary MoA responses for the training data. These are not predicted nor scored.
- test_features.csv - Features for the test data. You must predict the probability of each scored MoA for each row in the test data.
- sample_submission.csv - A submission file in the correct format.

Data: https://www.kaggle.com/c/lish-moa/data

PROJECT

This competition should be done using *TF2* & *Keras*. Here are the expected steps:

- Data Reading using *tf.data*. *tf.data* is a highly scalable toolkit for building data pipelines, and provides a few functions for dealing loading CSV files.
    - You can use any of the following methods to read out-of-memory data:
        - *tf.data.experimental.make_csv_dataset*
        - *tf.data.experimental.CsvDataset*
        - *tf.data.TextLineDataset* and *tf.io.decode_csv*
        - Avoid using *tf.data.Dataset.from_tensor_slices* since this method reads data from in-memory dictionnaries.
    - You need to split the training dataset into training and validation to be able to perform the hyperparameter tuning. You can use cross-validation if you prefer.
    - The testing dataset provided will be used as a testing dataset. Although the label of this dataset is not available, it can be used to evaluate our model over this test dataset once we submit the predictions to Kaggle (instead of evaluating on a notebook).
    - Data preparation should include
        - Shuffling
        - Batching
        - Mapping to corresponding format (removing sig_id for example)
- Feature Engineering
    - You can use either *tensorflow feature_columns* or *keras preprocessing layers*
    - You should differentiate between numerical features and categorical features and apply the corresponding adequate feature engineering strategies to the right format.
    - Hints
        - Normalization of numerical columns
        - One hot encoding of categorical columns
        - Optional feature engineering strategies to test:
            - Bucketizing numerical columns
            - Hashing numerical columns
            - Embedding categorical columns
            - Shared embeddings
            - Cross features
- Baseline Modeling
    - You can use either the Keras sequential API or the Keras function API.
    - You should have at least one hidden layer.
    - You should try implementing the following scenarios to see if it can improve performance:
        - L2 regularization
        - Dropout regularization
        - Batch Normalization
        - Weights Initialization
    - This is multilabel classification problem. You can setup the output layer to have a number of nodes equal to the number.
    - Callbacks you can implement:
        - Early Stopping
        - TensorBoard (to view model logs on tensorboard)
- Model Variance & Bias Analysis

    - Plot the training and validation loss of your model as a function of the number of epochs
    - Plot the learning curve
    - Diagnose Variance & Bias and propose a way to solve high variance or high bias if it’s being detected.
- Hyperparameter tuning with Keras Tuner
    - You have preset must of the hyperparameters so far!
    - You can tune some of these hyperparameters instead of trying and guess the best options.
    - Some of the hyperparameters you can tune:
        - Number of nodes in hidden layers.
        - Regularization parameter or dropout probability.
        - Learning rate.
        - Batch size.
        - Maximum number of epochs.
        - Optimizer.
        - …
- Project Results:
    - **Notebook:** You should submit your ipynb notebook to campus.
    - **Kaggle Submission:** You should submit predictions on the test set to Kaggle and provide the link to submission.