The LSC Benchmark Dataset: Technical Appendix and Partial Reanalysis

Andreas Mayr*, Günter Klambauer*, Thomas Unterthiner*, Sepp Hochreiter

* Shared First Authors

February 12, 2020

Abstract

We recently published the article "Large-scale comparison of machine learning methods for drug target prediction on ChEMBL", that compared machine learning methods for drug target prediction. Thereby we tried to suggest a pipeline for avoiding three biases, that are often immanent to target prediction performance comparison studies. In the following, we present an overview over the dataset, the pipeline, and discuss some technical details with respect to a large-scale study, such as ours. Furthermore, we provide results of a partial reanalysis of some of the deep learning results by changing the normalization aspect of feature preprocessing.

1 Introduction

In drug discovery a general goal is to find molecules (compounds) that interact with proteins (targets) in order to achieve beneficial therapeutic effects. Thereby, typically different chemical compounds are screened by diverse assays to get a profile on their bioactivities. Public databases, such as ChEMBL [1], contain bioactivties of molecules, that are reported by researchers, together with graph representations of their molecular structures. Overall such databases are often very sparse in the sense, that only a few bioactivities are measured by assays for most compounds.

The sparseness of databases such as ChEMBL and the invention of new molecular structures, motivates the usage of machine learning techniques in order to be able to virtually screen compounds for bioactivities.

The LSC [2] benchmark dataset was developed for comparing machine learning methods for the task of predicting bioactivity for molecules represented by features, graphs or strings. It is extracted from the ChEMBL database.

In our LSC comparison study, we compared standard deep feed-forward neural networks (FNNs) to other methods such as Support Vector Machines

(SVMs) [3], Random Forests [4], but also graph convolutional networks [5, 6, 7] and a recurrent neural network [8] with SMILES input. In contrast to many other studies, our method comparison was on a large-scale dataset including diverse assays, it tried to avoid the compound series bias leading to overoptimistic performance estimation and finally also tried to avoid hyperparameter selection biases.

Challenges, with respect to the "large-scale aspect" of this study were that we had to define a protocol in order to extract assay data being ready to be processed by machine learning algorithms (Section 2), cluster all compounds of the ChEMBL database in order to be able to provide cluster-cross-validation results [9] (Section 3), and develop pipelines, that made effective use of limited hardware resources (available to this study) in order to be able to apply a nested cross validation procedure [10] (Section 4). Moreover it was necessary to create features, which were usable by FNNs and for which we had to consider how to use them in our pipeline (Section 5). In Section 5 we also present the results of a partial reanalysis.

Tools, which we developed for this study, are available at GitHub: `https://github.com/ml-jku/lsc`

2 ChEMBL Data Extraction

Overall, the ChEMBL 20 database contains 1,456,020 compounds, 13,520,737 bioactivity measurements, and 1,148,942 assays assigned to 10,774 targets. However, the distribution of ChEMBL measurements across assays is highly unbalanced. There are assays with a large number of measurements (tens of thousands), while for more than half of the assays there is only a single measurement available. Some of the activities reported in ChEMBL had quite standardized activity comments, while others did not have. However, activity values and activity relations were available for all activities.

Our protocol therefore consisted of four steps:

1. Look, whether a measurement has a standard activity value, such as Active", "active", or "inactive".

2. Measurements not having a standard activity value or activity values with predefined relations and "nM" for the unit, were discarded.

3. Labels were assigned according to thresholds.

4. Merging measurements and removing contradicting measurements.

For label assignment in step 3. we used a gray zone of activity values (measurements being neither clearly active nor inactive), for which we did not assign a label for method comparisons.

Finally, we obtained a benchmark dataset with 1,310 assays and 4,743,712 assay measurements of 456,331 compounds.

3 Clustering of the ChEMBL database

Chemical datasets are often created in a way such that a certain molecule structure is optimized towards certain properties. Thereby the resulting molecules are very similar to each other and form a compound cluster. However, when predicting activities of molecules, it is interesting to know the prediction performance for molecules of previously unseen compound clusters. In order to estimate this prediction performance, we used cluster-cross-validation. Cluster-cross-validation does not exclude a random set of compounds from the training set and uses this set as a test set, but instead excludes whole compound clusters from the training set and uses the excluded clusters for the test set.

Towards applying cluster-cross-validation, we had to cluster all compounds from the ChEMBL database. In order to be able to guarantee minimum distances between training and test set, we decided for single-linkage clustering and used the Jaccard distance of binary ECFP features as the distance between two compounds.

Clustering more than a million compounds was not completely straightforward. First, it should be considered, that storing half the similarity matrix for 1.45 million compounds would require about 3 to 4 TB of storage, when a 4 byte float data type is used. In order to circumvent this requirement and the complexity for building a tree and determining a good cutting threshold for this amount of data, we instead took another approach: We predetermined an equally-spaced number of cutting thresholds for the agglomerative clustering tree and tried to assign compounds according to these predetermined thresholds (clustering threshold) immediately to clusters.

In detail the clustering algorithm worked as follows: Initially all compounds were assigned to an own cluster. Then, we computed the similarity of one compound to all others. If the distance of two compounds falls below a clustering threshold, the clusters of the two respective compounds are merged. Thereby a cluster is identified by an integer and if two clusters are merged, the new identifier is the smallest integer of the two clusters. The cluster identifiers of all corresponding samples are updated accordingly, which can be done efficiently, since the clusters are stored as linked lists of samples. It should be noted, that instead of using a float type for storing distances, we stored them as integers by cutting off some decimal places in order to speed compute processing up.

The computation of similarities of one compound to all others as well as merging clusters according to different thresholds can be parallelized, which we did by using OpenMP [11].

Since, depending on the threshold, the resulting clustering solutions, were either very imbalanced or had a lot of clusters, we could not directly apply cluster cross-validation on a clustering solution found in this way. Instead we merged the found clusters in one of three folds, respectively. Then, we applied the cluster-cross-validation procedure to these folds. This should still guarantee a minimum distance between training and test dataset.

4 Nested Cross Validation

Many machine learning algorithms require several hyperparameters, which should be tuned with respect to the dataset considered. However, for estimating prediction performance, it should be avoided, that the hyperparameter selection process makes use of the test set in order to avoid a hyperparameter selection bias. Since the estimation of prediction performance itself, already required a cross-validation loop, a nested cross-validation loop was needed for hyperparameter selection.

The hyperparameter selection process itself became very compute intensive, since usually a lot of different hyperparameter selection have to be tried out. In the following we especially consider the case of FNNs and SVMs.

For Deep Learning the hyperparameter selection process is especially critical from a computational point of view, since FNNs are usually trained on GPUs and the number of GPUs is limited most of the time. Although we had a certain amount of GPUs, that we could allocate for our performance study project, in overall the number of available GPUs was limited. Therefore, in order to speed up the hyperparameter selection process for Deep Learning, we trained FNNs for several hyperparameters on one GPU in parallel.

We implemented the training of single FNNs in Tensorflow [12]. In order to be able to start training of a new FNN on a GPU, if an old one finished training, we had to free up that space on the GPU, which was needed by the old FNN. Since, this didn't seem to work extremely well, we instead created a forked process for training a FNN, which easily allowed, that space was freed up, once training was finished.

For SVMs, the most critical part from a computational point of view is possibly the computation of the kernel matrices, if the number of samples is large. We therefore computed a full similarity matrix for a target in advance and then extracted from this matrix submatrices for training and test sets of the nested cluster-cross-validation procedure. The kernel matrices could be computed efficiently in parallel using OpenMP on a supercomputer.

5 Sparse Feature Handling and Reanalysis of Deep Learning Results for Dynamic Features

For the benchmark dataset we provided a lot of features, which describe the molecules and which we roughly grouped into the following main types:

- Dynamic features (i.e. sparse features such as ECFP [13] or DFS features)
- Semisparse features
- Static features

Note, that dynamic features are not predefined, but computed on the fly. Therefore, usually, depending on the radius or size hyperparameter, the number

Table 1: AUC Performance Estimation for Sparse features with explicit normalization

Feature group	Mean	Std.
ECFP6 + ToxF	0.742	0.125
ECFP6	0.731	0.127
DFS8	0.716	0.126

of dynamic features becomes huge, since many of these features are very specific to one or a very few molecules only. The matrix relating features to molecules becomes very sparse for dynamic features (i.e. extremely many entries at zero).

From a computational point of view, two aspects were important for us to handle dynamic features efficiently. First, we made use of sparse data structures (sparse matrices), that allowed a faster and more memory efficient processing of the data. Second, we applied a slight feature selection criterion, requiring a minimum occurrence threshold for a feature on the training set set.

For training FNNs we normalized data to mean zero and standard deviation one. However, for dynamic, sparse features (ECFP+Tox, ECFP, DFS), because of the sparseness of most of the features, we assumed that data were already at mean zero and therefore skipped the normalization step.

In a post-analysis after publication, we ran those experiments of the LSC publication without the above assumption of assuming a mean of zero, and instead explicitly normalized the data. This even increased the overall performance for the sparse features compared to the LSC publication

Using random cross-validation, we obtained a mean AUC of 0.774 ± 0.122.

6 Conclusion

The LSC performance comparison study includes several challenges from a computational point of view:

- Large-scale Data from public databases had to be preprocessed in order to obtain a ready-to-use dataset for training and testing of machine learning methods
- All compounds of ChEMBL had to be clustered in order identify compound clusters and therefore to be able to perform cluster-cross-validation.
- Processing structures had to be developed to be able to handle all computations given limited computational resources for the large-scale study.
- The large number of sparse features required efficient data storing and processing mechanisms. Moreover, we applied feature selection to reduce the number of features.

We provided code for all these substeps in our GitHub repository, which should allow to reuse our pipeline for new versions of the ChEMBL database and possibly also other similar applications, where cluster-cross-validation or nested cross-validation is needed. Further, we provide preprocessed parts of the database at our website `https://ml.jku.at/research/lsc/mydata.html` and a description of our pipeline at `https://ml.jku.at/research/lsc/index.html`.

References

[1] A Patrícia Bento, Anna Gaulton, Anne Hersey, Louisa J Bellis, Jon Chambers, Mark Davies, Felix A Krüger, Yvonne Light, Lora Mak, Shaun McGlinchey, et al. The ChEMBL bioactivity database: an update. *Nucleic Acids Research*, 42(D1):D1083–D1090, 2014.

[2] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K. Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. Large-scale comparison of machine learning methods for drug target prediction on ChEMBL. *Chem. Sci.*, 9:5441–5451, 2018.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[4] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[5] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2224–2232. Curran Associates, Inc., 2015.

[6] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chem. Sci.*, 9:513–530, 2018.

[7] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8):595–608, Aug 2016.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016.

[10] Désirée Baumann and Knut Baumann. Reliable estimation of prediction errors for QSAR models under model uncertainty using double cross-validation. *Journal of Cheminformatics*, 6(1):1, 2014.

[11] Leonardo Dagum and Ramesh Menon. Openmp: An industry-standard api for shared-memory programming. *Computing in Science & Engineering*, (1):46–55, 1998.

[12] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[13] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.