OPTIMIZING KERAS ANNs FOR PREDICTING MOLECULES MUTAGENICITY

Project Report by
Abd Alkareem ALJEIROUDI
Institute for Machine Learning
March 26th, 2019

Abstract

In this report we consider the potential of deep neural networks as a means for predicting molecular properties. Our goal is to find the best hyperparameter setting that helps our neural network generalize well on future data, and we achieve that via the analysis of the AUC score. For our analysis, we use a training dataset consisting of 4337 structures. We also discuss the methods that were used to optimize the network's architecture such as Grid Search. Furthermore, we demonstrate how some hyperparameters influence the AUC score of the model through the Wilcoxon-rank-sum test. Eventually we reject the null hypothesis regarding the choice of optimizer. Further analysis in regard to feature attribution is required, in order for this project to accomplish its second goal.

Introduction

An essential step in drug approval is the omission of mutagen molecules. Mutagenicity is the compound's ability to induce DNA mutation, leading to either some deletions or adducts in the DNA. Some DNA repairing mechanisms get distorted because mutagenic compounds intercalate between the double stranded helix. For the last few decades, the Ames test has been the standard test for mutagenicity detection due to its simplicity. Nevertheless, the Ames test has its limits, as it is not the easiest to reproduce. The National Toxicology Program has determined a 15% interlaboratory reproducibility error in the Ames test [7]. Therefore, to assist drug approval procedure, algorithms such as machine learning are employed. Recently, with the advancement that machine learning techniques have seen, multiple techniques have been proposed to predict the outcome of the Ames test. There have been attempts to measure the accuracy of these prediction using benchmark data sets [6]. Also, some comparisons between commercial and non-commercial solutions have shown that non-commercial techniques are far more promising [6]. The reason behind that, is that commercial solutions are often set to default parameters, thus full control over the algorithm is not a possibility. The aim of this project is to construct an artificial neural network and best optimize its parameters. Here we are particularly interested in the area of deep learning, more specifically, we used sequential models from the Keras API to fit the data. When it comes to training on this dataset, we are trying to achieve two goals in this project a) finding the best set of parameters that optimizes our learning model together with this data; b) finding out whether or not some specific parameters influence the performance of the model. Although deep neural networks make more accurate predictions, they are, however, more difficult to train and evaluate due to larger architecture and number of parameters to tune. Note that the optimization process used in this project is not optimal itself. In another word, there is still room for improvement, in order to elevate the AUC score even higher.

Methods

The dataset from the original paper was made publicly available for future comparisons with other prediction methods (see http://cheminformatics.org/datasets/bursi/) [7].

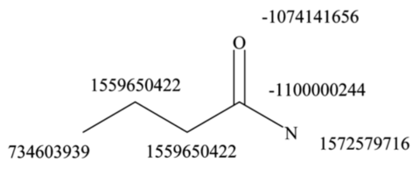

Fig. 1: A set of unique identifiers obtained from calculating the Morgan fingerprints.

In general, the capabilities of machine learning methods are highly dependent on the type and size of data of interest. In our case, we are looking at a dataset consisting of 4337 structures that were all approved by the Ames test. Initially the dataset was selected to help elevate the sensitivity of identifying toxicophores [7]. Each of these molecules was labeled either ‘mutagen’ or ‘nonmutagen’ and we want to see if the model can recognize patterns in the data that suggest the mutagenicity. With that being said, one can infer to the fact that the problem at hand is a typical binary classification, where the model predicts one of the two classes based on a set of features that are fed to the model. Additionally, data is provided in `.sdf` format, which is a portable format in Chemoinformatics. In the following section we are going to discuss the data preprocessing procedure we used to obtain the training set.

Data Preprocessing

To convert the molecules into binary form, we used the Morgan Fingerprints (ML) algorithm, which has become the standard molecule isomorphism algorithm since 2000 [3]. The idea is to assign an initial identifier to each atom in the substructure within a predefined radius, and to update each identifier iteratively. The initial identifier is set using the atomic number and bond type. In an iterative manner, a hash function hashes all identifiers gathered in the previous iteration and assigns a new one to the current atom. Results of the MF algorithm are gathered in set of unique identifiers (see Fig. 1). To calculate the fingerprints, we used the `RDkit` library which has a handful of functions to calculate chemical properties (see Snippet 1). Calculations of the Morgan Fingerprints are then written into a dictionary where the keys represent the identifiers and values are represented in tuples of a pair of values (*atom rank, radius*). The reason we

Snippet 1: Calculating Morgan fingerprints using functions from `RDkit`. Here, identifier 98513984 was set twice: once by atom 1 with radius 1 and once with by atom 2 with radius 1.

```
>>> from rdkit.Chem import AllChem
>>> fp =
AllChem.GetMorganFingerprint(m, 2,
bitInfo=info)

>>> info[98513984]
((1, 1), (2, 1))
```

chose the Morgan fingerprints is because it has proven over the years to give a consistent performance in molecular Data. A literature from 2014 by Li Qingliang describing drug-like machine learning models using support-vector machines found out that fingerprints-based algorithms gave a superior performance compared to other ones such as MOLPRINT_2D, on four different performance measures (accuracy, sensitivity, specificity, and Matthews correlation coefficient [4].

Later on, the data was then converted to bit vectors using binary bit encoding, where presence of a particular feature is indicated by 1 and the absence of that feature results in a 0. As a result, our final training dataset is a matrix of 4337 rows and 2048 features. Needless to mention that the positive class (‘mutagen’) was labeled 1, and 0 for the negative class. Samples of the validation set were randomly selected resulting in 7.5% split (train:validation 4010:327). Last but not least, whole data was normalized using Standard scaling in order to set the mean to 0.

Model Architecture

It is important to note here that there is no strict guideline when building the model’s architecture. To get started our initial classifier used 1 input layer consisting of 5 rectified linear hidden units. For backpropagation we used the following parameters (loss: binary corss-entropy, optimizer: SGD, learning rate: 0.01, momentum: 0.9). Surprisingly, this quick-and-dirty model scored an AUC ~0.86, while it took only 1.13 seconds to train on 100 batches and 10 epochs. Reasons for choosing the AUC as performance measure are discussed in an upcoming section (see

Performance Metric). A more informative approach is required to better tune the training hyperparameters. For instance, if you are planning to use Adam as an optimizer, then you should set the learning rate rather small (≤0.001) as opposed to using SGD. By doing so, we gained intuition of which parameters are performing better on this dataset. However, one of our aims was to find that set of combined hyperparameters that best optimizes learning. Therefore, we conducted a grid search, which is guaranteed to find an optimal solution, if the search runs infinitely. One downside to this optimization method is obviously the high runtime that the method requires. While it is true that there exist strategies to overcome this limitation (e.g. parallelization), we, however, did not attempt to use any of these strategies. In the following we will discuss the initiation and implementation steps of the grid search as well as search space.

Optimization Method

As mentioned earlier, the advantage of using a grid search over other optimization methods is that a grid search finds an optimal solution. I.e. it finds the best set of parameters θ that produces the highest AUC score on validation set. Because we wanted to substitute the number of hidden layers, we wrote our own implementation. Refer to Table 1 to learn more about our ultimate search space that was used in the own-implementation. In our case, the search took around 10 hours when run serially on E5-2640 v2 @ 2.00GHz CPU. In the meanwhile, every constructed model was filed together with its hyperparameters and its AUC score. Using another python program, the file was later converted to comma-separated format, in order to sort the constructed models according to their AUC score as well as for further analysis of the tuning data. We will refer to this analysis in the results section. Table 2 shows a summary of the tuning data.

Performance Metric

The Area under the ROC curve is a widely accepted measure for binary classification tasks. It is the area enclosed under the Receiver Operating Characteristic curve, which is essentially a tradeoff between the true positive rate and false positive rate ($1 -$ sensitivity) (see Fig. 5). The value of the AUC metric varies from 0 to 1 and an optimal classifier is that one, whose AUC score is equal to 1. This is an ideal scenario that never happens practically, nevertheless, obtaining values that are as close to 1 as possible is a sign of a 'good' classifier. The classifier consequently is unlikely to make false predictions (i.e. lower number of false positives and false negatives). It is convenient to use the AUC as a metric for classification tasks, since the metric is insensitive to changes in class distribution.

Results

Table 2 shows the first 15 best scoring models sorted according to the AUC score. The best scoring model had an AUC of 0.89 during tuning without cross validation. After reconstructing the model with the same set of hyperparameters and evaluating it, we observed a slight drop in the AUC score ~0.87 on validation set, which is a sign that model has indeed learned some patterns in the data. This drop in the AUC score can be attributed to randomness during initialization. When running the model on the validation set, there is a good chance for a model in the search space to give a

Table 1: Search space that was used for the ultimate grid search. Next to each hyperparameter is the set of values that was used. Every permutation is a unique combination of all these values.

batch size	[10, 50, 100, 200]
epochs	[10, 20, 30]
optimizer	[SGD, Adam]
learning rate	[0.001, 0.01, 0.1, 0.2]
activation	[ReLU, SeLU, sigmoid]
Num. hidden layers	[2, 5, 9]
neurons	[5, 50, 256, 512, 1024, 2048]

high AUC despite the fact that it had predicted randomly. Therefore, to identify these models we tested the same hyperparameter settings on a test set. Our best scoring classifier returned **AUC of 0.81** when predicting on the **test set**, which consists of 3315 structures (1625:1690, non-mutagen:mutagen). Here we report the best hyperparameters setting found by our 10 hours-long grid search:

Activation (input & hidden layers)	ReLU
Activation (output layer)	Sigmoid
Kernel Initializer	he_uniform
Dropout rate (at each hidden layer)	0.1
Num. of hidden Layers	5
Num. of units	512
Optimizer	SGD
Momentum	0.9
Learning rate	0.01

With all that being said, the following remarks can be made regarding the tuning data:

- As we compare the score distributions of models using SGD to those using Adam, we can see that SGD has outperformed Adam when using this search space and dataset (see Fig. 3). The reason for that is that most permutations of the grid search used a relatively high learning rate (0.01 - 0.5), which is incompatible with Adam. In order to put that to the test, we conducted a Wilcoxon rank-sum test, which supports this claim (W = 62130, p-value = **2.508e-10**) and therefore, we reject the null hypothesis. In order for Adam to adjust the weights accordingly, it needs a much lower learning rate than 0.001. Therefore, we can conclude that our search space was insufficient to explore more compatible regions for Adam. It is likely that Adam would have outperformed SGD, having given more values below 0.001. This can be inferred to by observing 12th row in table 04 (index 52). This high scoring model (AUC ~0.88 during tuning) is an Adam outlier, and is believed to have a set of dependent parameters
- The same cannot be said about the activation functions. While it is true that our best model

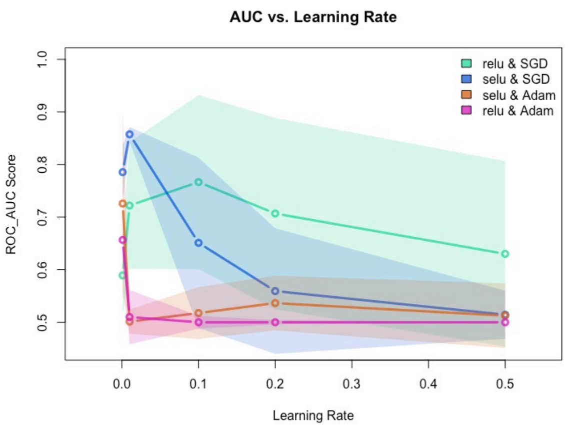

Fig. 2: Mean AUC scores calculated for every activation and optimizer configuration. The claim that Adam could have scored better having been given lower learning rates is seen on this plot. See the great deviation in AUC score that the configuration of SGD & ReLU has.

  uses ReLU for activation, it is rather vague which activation function performs better on this data set. According to the Wilcoxon rank-sum test (W = 37419, p-value = **0.5781**), there is no significant difference between the means of these two distributions. In another word, should we have kept the search running a little longer, we might have obtained a better model with SeLU instead.
- All fifteen models used 512 units or more.

The results of using sequential models in predicting mutagenicity were rather promising. The fact that we obtained a hyperparameter setting that allows the network to generalize on future data is definitely progressive. However, the fact that the number of parameters to be tuned, and the time of training and evaluation is rather lengthy, should be considered before deciding for deep neural networks.

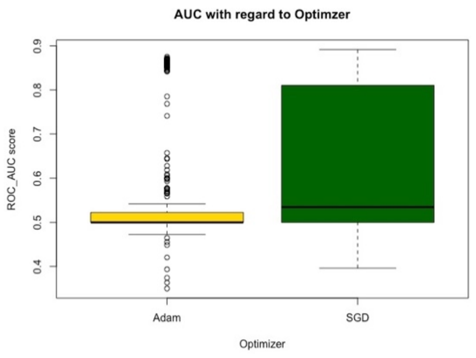

Fig. 3: Boxplot showing the AUC score distributions of models using the optimizer Adam vs. SGD. Notice that Adam's mean lies close to 0.5, however the distribution is full of outliers. The reason for Adam to have such a small deviation is because the number of constructed models using lower learning rates (below 0.001) was rather low.

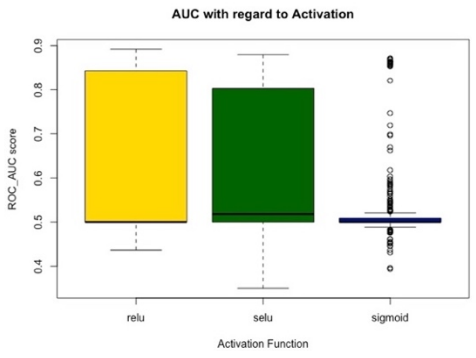

Fig. 4: Boxplot showing the AUC score distributions obtained using different activation functions. The mean of SeLU lies somewhat higher than that of ReLU, indicating that the majority of models using SeLU for activation scored higher, however the better performant happened to use ReLU for activation. Sigmoid on the other hand, performed the worse and is full of outliers.

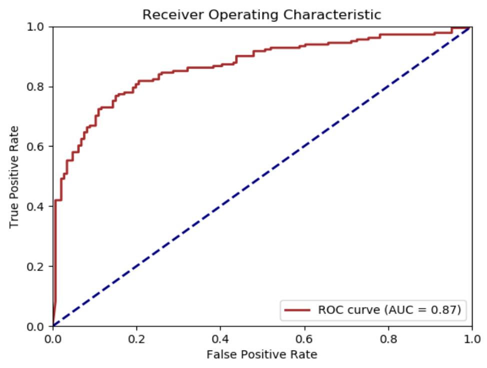

Fig. 5: ROC curve of the best-scoring model (index 24) when run on the validation set. The returned AUC was 0.87.

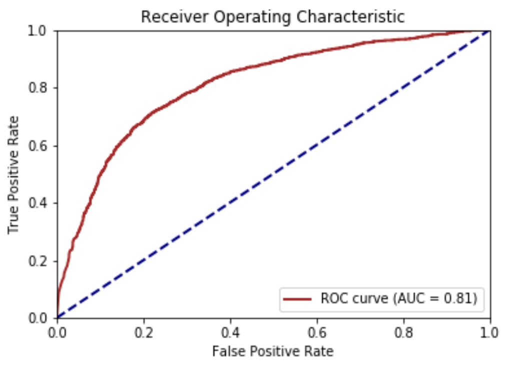

Fig. 6: ROC curve of the best-scoring model (index 24). When run on the test set. The returned AUC was 0.81. The reason for the smoothness of this curve is because the size of the test set is relatively larger than that of the validation.

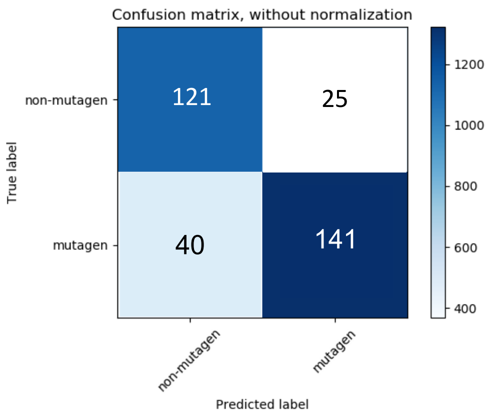

Fig. 7: Confusion matrix corresponding to the validation set. Clearly the model was less sensitive to false negatives than to false positive, since there's 15 times prediction difference.

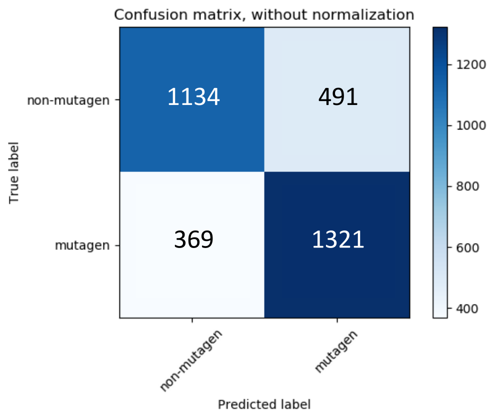

Fig. 8: The confusion matrix corresponding to the test set. The trend (sensitivity to false negatives) we observe at the validation set is not visible here.

Table 2: Summary of the first 15 best models sorted according to the AUC score.

index	batch size	activation	learning rate	num. layers	units	AUC	optimizer	epochs
24	100	ReLU	0.1	5	512	0.891622	SGD	10
567	100	ReLU	0.1	9	2048	0.885265	SGD	10
571	200	ReLU	0.2	2	2048	0.882464	SGD	10
561	100	ReLU	0.1	2	2048	0.882105	SGD	10
293	50	ReLU	0.1	5	1024	0.880005	SGD	10
562	200	ReLU	0.1	2	2048	0.879607	SGD	10
195	100	SeLU	0.01	5	512	0.879323	SGD	10
299	50	ReLU	0.2	2	1024	0.877356	SGD	10
21	100	ReLU	0.1	2	512	0.876826	SGD	10
738	100	SeLU	0.01	9	2048	0.876750	SGD	10
551	50	ReLU	0.01	2	2048	0.875842	SGD	10
52	200	ReLU	0.001	5	512	0.875577	Adam	10
576	100	ReLU	0.2	9	2048	0.874612	SGD	10
464	50	SeLU	0.01	5	1024	0.874366	SGD	10
202	200	SeLU	0.1	2	512	0.874215	SGD	10

Conclusion & Future Work

The Morgan fingerprints is an easy and straightforward method to employ for identification of molecular structures. More interestingly, the hyperparameter tuning data has proven conclusive enough. Stochastic Gradient Descent has outperformed Adam on the dataset at hand and with these different sets of hyperparameters. In that regard, we rejected H_0 (W = 62130, **p-value = 2.508e-10**). On the other hand, we could not conclude to a best activation function due to insufficient number of tested hyperparameters. Our best model returned a high AUC score (Test set: ~**0.81**). Lastly, the application of deep neural networks has its both advantages and disadvantages.

The findings of this project do not fully optimize the network as desired; reason being that some features dominate other ones in the training set. Therefore, addressing this issue requires deeper understanding of features credibility using Integrated Gradients.

References

[1] Fewcett, T. (2005, December 19). An introduction to ROC Analysis. Retrieved Feb 2019, from Science Direct: https://www.sciencedirect.com/science/article/pii/S016786550500303X

[2] Huijskens, T. (2016, December 29). Bayesian optimization with scikit-learn. Retrieved February 15, 2019, from Github Pages: https://thuijskens.github.io/2016/12/29/bayesian-optimisation/

[3] Extended-Connectivity Fingerprints
David Rogers and Mathew Hahn
Journal of Chemical Information and Modeling 2010 50 (5), 742-754
DOI: 10.1021/ci100050t

[4] Li, Q.; Bender, A.; Pei, J.; Lai, L. A Large Descriptor Set and a Probabilistic Kernel-Based Classifier Significantly Improve Drug- likeness Classification. J. Chem. Inf. Model. 2007, 47, 1776–1786.

[5] Snoek, J., Larochelle, H., & Adams, R. P. (2012, August 29). Practical bayesian optimization of machine learning algorithms. Retrieved December 2018, from https://arxiv.org/pdf/1206.2944.pdf

[6] Benchmark Data Set for in Silico Prediction of Ames Mutagenicity
Katja Hansen, Sebastian Mika, Timon Schroeter, Andreas Sutter, Antonius ter Laak, Thomas Steger-Hartmann, Nikolaus Heinrich, and Klaus-Robert Müller
Journal of Chemical Information and Modeling 2009 49 (9), 2077-2081 DOI: 10.1021/ci900161g

[7] Derivation and Validation of Toxicophores for Mutagenicity Prediction
Jeroen Kazius, Ross McGuire, and, and Roberta Bursi,
Journal of Medicinal Chemistry 2005 48 (1), 312-320 DOI: 10.1021/jm040835a

[8] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In Proceedings of the 34th International Conference on Machine Learning - Volume 70 (ICML'17), Doina Precup and Yee Whye Teh (Eds.), Vol. 70. JMLR.org 3319-3328

[9] Li, Q.; Bender, A.; Pei, J.; Lai, L. A Large Descriptor Set and a Probabilistic Kernel-Based Classifier Significantly Improve Drug- likeness Classification. J. Chem. Inf. Model. 2007, 47