Machine Learning for Health Care - Project 2 Report

Manuel Burger, Jonas Meirer, Tobias Peter, Anej Svete [*]

April 2021

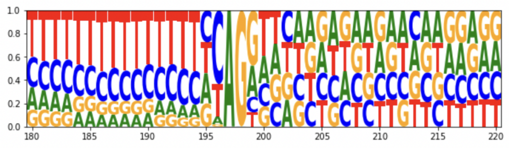
1 Introduction & Overview

This report provides a concise summary and discussion of our submission for the second project for the class *Machine Learning for Health Care.*

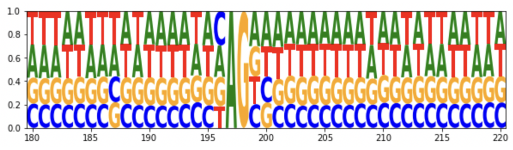
The objective of the project was to classify sites of human DNA sequences as acceptor or non-acceptor sites. Evaluation of the classification is based on the ROC (receiver operating characteristic) and PR (precision-recall) curves. In the following, we will describe the various models that we used, their training as well as their performance. Details can be found in the corresponding Jupyter notebooks, which are referenced in the respective sections.

2 Analysis
`ExploratoryDataAnalysis.ipynb`

Exploratory data analysis of the human data set revealed interesting patterns that could be exploited to build simple models. For example, the proportion of the nucleotide C right before the 'AG'-sequence of the potential acceptor site is strongly elevated for acceptors compared to non-acceptors; cf. Figure 1. This pattern shifts if we take a longer sequences into account. Here, the nucleotide T is found over-proportionally often in the area before true acceptors. Figure 1 also reveals that the proportion of T's gradually declines as we move further to the left of the true splice site.



(a) Acceptors



(b) Non-Acceptors

Figure 1: Sequence Logo

*{burgerm, jmeirer, topeter, asvete}@ethz.ch

This confirmed our suspicion that the nucleotoides around the center have high a predictive power in this classification task. This was extensively analyzed and confirmed in the context of *Explainable Boosting Machines (EBM)*, cf. Subsection 3.2.5. Moreover, we observed an extreme imbalance between acceptor and non-acceptor sites.

3 Modeling & Training

3.1 Pre-processing

For rapid prototyping and testing, we split the C. elegans dataset into 70% training set, 15% validation set and 15% testing set preserving the class imbalance.[1]

The individual sequences are then fed into our models using a one-hot encoding of either single nucleotides or different lengths of k-mers.[2]

3.1.1 Dense k-mer embedding

As a preprocessing step and data representation approach for the human dataset, we embedded individual k-mers for $k = 6$ into a dense vector representation, inspired by similar approaches in NLP.[3]

Hyperparameters were tuned based on the performance of the model using the embeddings; see Section 3.2. The dimension of the embeddings was finally set to 30. Since the considered k-mers were overlapping, the co-occurrence window size was set to 24, which is comparatively high. The model was trained for 100 epochs with a learning rate of 0.05. Other parameters were left at their default values.[4]

3.2 Models

3.2.1 Logistic Regression
`logistic_regression.ipynb`

As a baseline-model, we used Logistic Regression on the one-hot encoded dataset. Taking the imbalance

[1]The datasets are provided as part of our submission.

[2]The scripts `data_io.py` and `data_representation.py` were created to process the data.

[3]We also experimented with both the Word2Vec and GloVe implementation of the embeddings and found the latter to perform better.

[4]See notebook `kmer_embedding.ipynb`

into account, we tried class-weighting, under-, over-sampling as well as combinations thereof. We tried l_1, l_2 as well as elastic net regularization and tuned for the corresponding trade-off parameter. Additionally, we performed feature selection based on the ANOVA F-value[5] between features and labels. For the test predicitions, we then used the model with the highest AUPRC on the validation set, which used l_2-regularization with 250 features for the human dataset and 225 for the C. elegans dataset, respectively. We also tried adding k-mers for various k. However, this had a negligible effect and was discarded.

3.2.2 Random Forrest

`random_forest.ipynb`

Feature selection was similarly performed as for Logistic Regression. We also performed grid-search for the number of estimators, maximum tree depth, the criterion and the maximum number of features for the splits, as well as for the weighting scheme. Based again on the highest AUPRC on the validation set, a parametrization with 150 features and 1000 estimators for the human dataset, and 175 features and 100 estimators for the C. elegans dataset was selected. We also tried integrating k-mers, which yielded a slight improvement on the validation set, but a deterioration on the test set.

3.2.3 Multilayer Perceptron (MLP)

`dense_nn.ipynb`

Extending the results of the Logistic Regression, we built a small dense neural network with 3 layers, Dropout, Batch Normalization and a Sigmoid activation head. Similar feature selection and sampling techniques as for the previous methods have been tested, *class weigths* and the 175 and 225 best features on C. elegans and human data respectively according to the ANOVA F-value have yielded the best AUPRC on our validation and test set. Especially for the human dataset, larger networks have been tested with diminishing returns and showed issues with generalization and overfitting. To better understand the predictions of the model, a SHAP [2] analysis has been performed on the C. elegans classifier, indicating high feature importance of nucleotides close to the splice site.

3.2.4 XGBoost Boosting

`xgboost.ipynb`

We also applied the extreme boosting model on both data sets. We determined the optimal input representation (among k-mer one-hot encoding and k-mer counts) and model hyperparameters via grid search.

On the C. elegans data set, the optimal representation was one-hot encoding of individual characters, which resulted in 246 features. In order to prevent overfitting, the best 80 features were chosen similarly to the previous models. On the human data, however, using one-hot encoded 2-mers provided a noticeable performance improvement (0.05 on AUPRC), which is why this representation was chosen. Out of these 5955 features, the best 1024 features were selected analogously to above.

The models were trained on the training sets, using the validation sets for early stopping. The class imbalance was dealt with by weighting the instances of the two classes differently. For the C. elegans data set, the positive class was weighted 10 times higher than the negative one, respectively 5 times for the human data set.

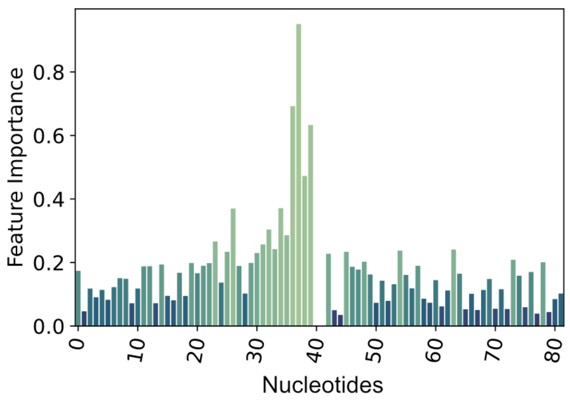


Figure 2: Feature Importance of single one-hot encoded nucleotides of an EBM trained on the C. elegans dataset

3.2.5 Explainable Boosting

`interpretable_boosting.ipynb`

As interpretability is an important aspect of machine learning in the medical sector we explored an explainable boosting tree algorithm developped by a team of Microsoft Researchers. The `interpretml` [3] package provides a boosting tree algorithm based on a generalized additive model, which allows to clearly explain feature importances for a trained model. A grid-search over hyperparameters has not shown any improvements over the default parameters. On the human dataset conservative undersampling has been used, whereas on the C. elegans dataset no sampling or weigthing scheme was used.

In Figure 2 we visualize the global feature importance for single one-hot encoded nucleotides of an EBM trained on the C. elegans dataset. One can clearly distinguish the higher importance of nucleotides close to the splice site - confirming the observations made in Section 2 - and towards the left of the splice site, which is very sensible considering the underlying biological mechanism. Similar results have been observed on the human DNA.

[5]We used SKL's *SelectKBest*.

3.2.6 Convolutional Neural Network

`keras_cnn.ipynb`

For classification of the human sequences[6], we implemented a simplified version of the SpliceAI model [1] (one taking into account 400 nucleotides of context). It uses a similar sequence of dilated convolutions, but is much smaller due to the smaller data set. It also uses dropout, but does not use residual layers[7].

We implemented two versions of the model: one that takes as input the raw one-hot encoded sequences and one that takes dense 30-dimensional embeddings of overlapping 6-mers[8]; cf. Section 3.1.1. Both had their advantages and disadvantages. In particular, the dense model produced more stable results, but took longer to train and the embeddings themselves were not as interpretable. In the end we decided to use both, suspecting that they would extract different features from the sequences, making their combination beneficial.

All models were implemented in Keras and trained for 20 epochs on the full training set with no under- or oversampling. The class imbalance was handled by weighting the positive class 4 times higher than the negative one[9]. Again, the validation data set was used to monitor the loss during the training.

In order to make the predictions more robust, 10 models of the first type and 11 of the second were trained. The final prediction is the average of the output of these models, ignoring the predictions of the five models which performed worst on the validation set.[10]

3.2.7 Other approaches

When exploring our options, we also looked at some other models including SVMs based on string similarity kernels and the Catboost library (a boosting library that is specifically designed to handle categorical inputs). None of these models provided substantial benefits over the other models, which is why we do not describe them in detail here.

4 Results

Tables 1 and 2 summarize the test scores of the various models on the C. elegans dataset and human dataset, respectively. For both datasets, AUROC is comparatively stable and does not seem to adequately distinguish the different classification capabilities of the various models.

On the C. elegans dataset, the two tree-based classifiers (Random Forest and XGBoost) have the highest AUPRC with a slight margin for Random Forest over XGBoost.

For the human dataset, the CNN has the highest AUPRC and outperforms the second best model (XGBoost) by a factor of 3. Its corresponding ROC and PR curve are displayed in Figure 3.

Model	AUROC	AUPRC
Logistic Regression	0.9856	0.8804
Random Forest	**0.9922**	**0.9448**
MLP	0.9816	0.8504
XGBoost	0.9914	0.9340
EBM	0.9848	0.9075

Table 1: Test scores for the C. elegans dataset.

Model	AUROC	AUPRC
Logistic Regression	0.9649	0.1673
Random Forest	0.9383	0.0893
MLP	0.9331	0.1464
XGBoost	0.9683	0.2158
EBM	0.9570	0.1260
CNN	**0.9811**	**0.6016**

Table 2: Test scores for the human dataset.

5 Further Work

Despite an already very good performance with our CNN architecture, we see following opportunities for improvement.

First, a more thorough tuning and testing of the embeddings (e.g. dimensions, learning rate) could further enhance the performance or at least improve convergence.

Secondly, it could be beneficial to combine the CNN architecture with hand-crafted features based on the observations made in Section 2. For example, one could stack a MLP on top of the CNN and use a skip connection from the hand-crafted features to the MLP. Thus, combining the performance of the CNN with the insights of the data analysis.

[6] The model was only used on human data, since the C. elegans data set was deemed too small to adequately train such a model without overfitting.

[7] We experimented with a more faithful representation of the model from the publication which can be found the corresponding file as well (with the same blocks, residual connections and size). However, its performance was worse than for our model, supposedly due to the larger number of parameters, making it unsuitable for the smaller data set.

[8] The performance improved when increasing the number of dimensions from 16 to 30. Yet, larger dimensions could not be explored due to the memory constraints.

[9] Higher weight rations resulted in too many false positives.

[10] See notebook `keras_cnn_ensemble.ipynb`

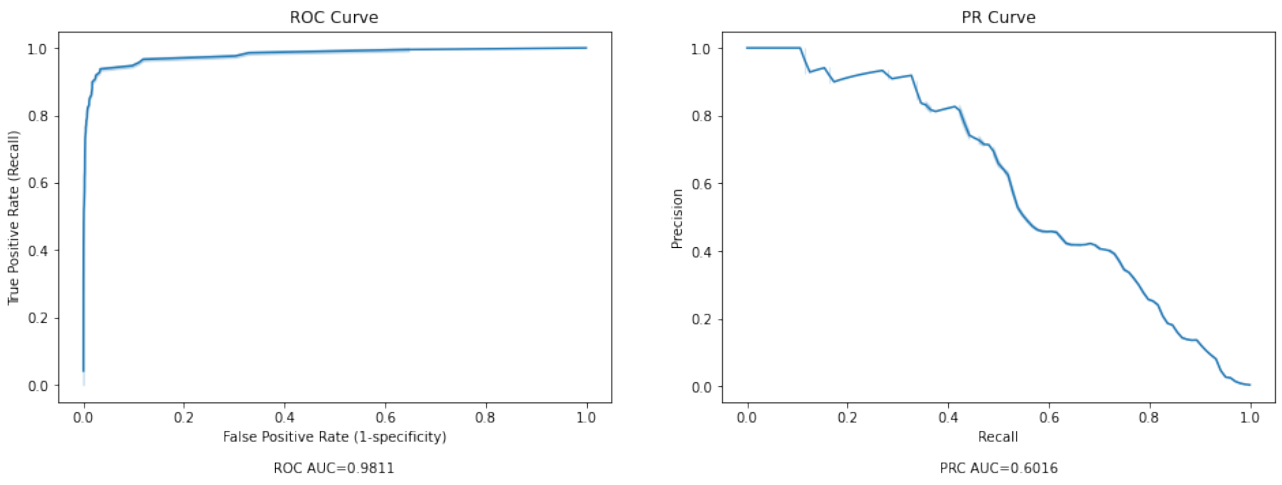


Figure 3: The ROC and PR curves of the trained CNN architecture on the human dataset

Main Contributions

M. Burger	MLP, Explainable Boosting Machine, some experiments with SHAP analysis and kernelized SVMs
T. Peter	Logistic Regression, Random Forests as well as some experiments with Neural Networks & SVMs, corresponding parts of the report
J. Meirer	Data Analysis, CNN implementation with one-hot encoding, separate analysis of XGBoost and Logistic Regression
A. Svete	XGBoost training, experiments with Catboost and SVMs, CNN implementation and training, k-mer embeddings, corresponding parts of the report

References

[1] Kishore Jaganathan, Sofia Kyriazopoulou Panagiotopoulou, Jeremy F. McRae et. al. *Predicting Splicing from Primary Sequence with Deep Learning.* Cell, Vol. 176, Issue 3, 2019. https://www.sciencedirect.com/science/article/pii/S0092867418316295.

[2] Lundberg, Scott M and Lee, Su-In *A Unified Approach to Interpreting Model Predictions.* Advances in Neural Information Processing Systems, Vol. 30, 2017. https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[3] Nori, Harsha and Jenkins, Samuel and Koch, Paul and Caruana, Rich *InterpretML: A Unified Framework for Machine Learning Interpretability.* arXiv preprint arXiv:1909.09223. 2019. https://github.com/interpretml/interpret