Model summary

Competition: Liverpool Ion Switching

Team: Realm of OVERFIT

Private LB Score: 0.95824

Private LB Place: 2nd

Name: Stanislav Dereka

Location: Moscow, Russia

Email: st.dereka@gmail.com

University: Moscow Institute of Physics and Technology (MIPT)

1 Summary

My final model is a stacking of random forest (RFC) and wavenet-based neural network architecture. The most important features for RFC models were signal lags and leads. Wavenet models accept RFC predictions and pure signal with low-order lags and leads as features. For RFC models I used Sklearn implementation. Wavenet is implemented in Keras with Tensorflow backend. It takes about one day to train the entire pipeline. Predictions can be done much faster – about 10 minutes on competition test dataset.

2 Preprocessing steps

Detrend signal There was a synthetic drift added to the data. For training my models I used amazing detrended dataset by Chris Deotte.

Align mean signal values The data can be divided into 6 groups corresponding to different generative models. One of them presents only in the test dataset. The mean values of the 3rd group were not aligned with the other groups. Aligning these values improved final model quality.

Remove outliers The 7th batch of the train data was corrupted by odd noise spikes. I replaced this segment by another noise sample from the same generative model.

Create more data It was possible to create new data from existing by summing up open channels values and corresponding noise samples. I described the process in all the details in this notebook.

3 Feature engineering

In this competition lag and lead features had significant importance. The explanation is that the open/close state of each ion channel tends to rest the same in a short period of time. For example, if in a particular moment of time N channels are opened, it is more likely that on the next step there will be also N opened channels.

In the picture, you can see that adding one lag feature helps split the data more effectively. Using this kind of features was extremely important for tree-based models.

One should be very careful when uses rolling stats (especially on large intervals). These features can be leaky and require thorough validation approach to resolve all possible problems with leakage.

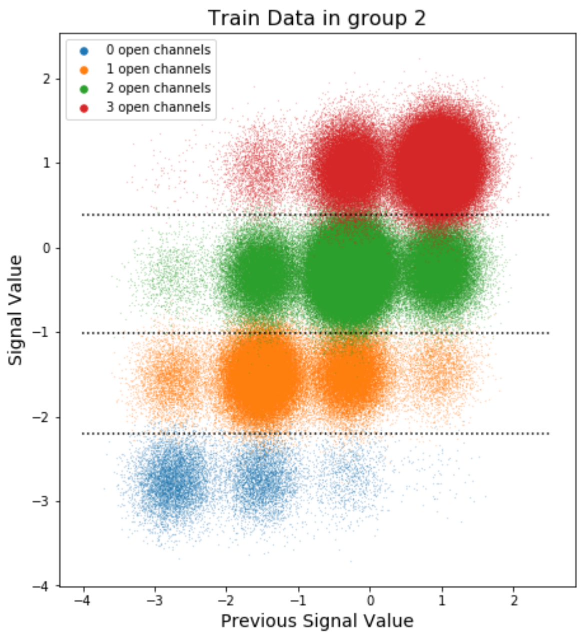

Figure 1: Relation between signal and lag

4 Training methods

1st layer of stacking: RFC To train random forest I used 1x5 cross-validation scheme with stratification by train signal batches:

```
cv = StratifiedKFold(n_splits=n_splits).split(
    train, train.index // strat_int
)
```

Out-of-fold (OOF) predictions were passed to the next layer of stacking. Prediction on the test data is a blend of 5 models corresponding to each fold.

2nd layer of stacking: Wavenet Wavenet was trained on signal sequences of length 4000, RFC OOF predictions were passed as features. I used 1x5 GroupKFold validation scheme:

```
cv = GroupKFold(n_splits=n_splits).split(
    train, train[target], group
)
```

In order to improve the model I implemented following online data augmentations: signal flipping, random shift, periodic 50 Hz noise and gaussian noise. OOF predictions were used to evaluate general model quality. Final prediction on the test data is a blend of 5 models corresponding to each fold.

General motivation for ensembling In my experiments I found out that errors of tree-based and wavenet-based models overlap only in approximately 80% cases. It means that the neural network is capable of correcting the errors of the tree-based model and vice versa

Advantage of the method The key advantage of proposed method is that only one model is used for all groups of the data. On training stage no manually segmented data is required.

5 Interesting findings

The key trick I used in the competition was new data creation described above. It drastically boosted all my models.

Another trick was the data leak I exposed in the last days of competition. It has been already revealed in this post.

In general, preprocessing strategy was the key to the competition, it made the most significant contribution to the final score.

6 How to simplify the model?

If you train pure wavenet model without RFC predictions, it will slightly reduce the performance but final quality on the test dataset will still be around 0.942. You can also reduce the number of validation folds and train fewer models.

7 Model execution time and final scores

Running the entire pipeline including all layers of stacking will take about one day on the hardware I detailed in README.md. Making predictions is *much faster*: it takes about 10 minutes for full test dataset. Things are getting easier with simplified model. In this case training will take several hours.

The model presented in this repository achieves 0.94535 on private LB without leak exploitation and 0.95356 with leak exploited.