Sustainable Industry: Rinse Over Run
12th place solution

Pavlovskaia Natalia

Skoltech, Higher School of Economics

March, 2019

Overview

The model uses multidimensional time series data along with the metadata for each process.

The time series data is propagated through the LSTM to extract features.
Then the last hidden state is concatenated with the metadata features.

This concatenated vector then go through 4 dense layers with gradually decreasing number of neurons.

I used 5-folds model. This means that I split the data in 5 folds. Train 5 models (with the same architecture) leaving out 1 fold and in the end average the predictions of these 5 models.

Time series features

- All **16** numerical time series
- All **12** boolean time series.
- **1** artificial feature *phase number*. It is a time series with the value 1 at timestamps corresponding to the pre-rinse phase, 2 at caustic phase, 3 at intermediate rinse phase, 4 at acid phase.

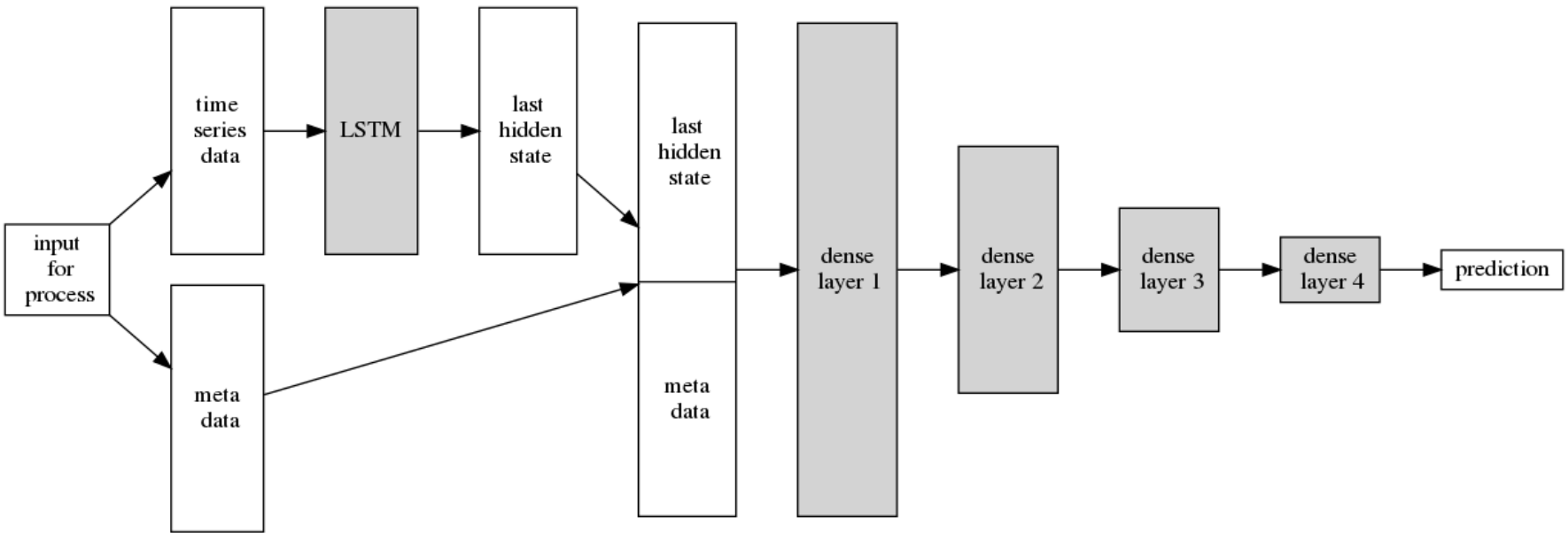
The dimensionality of time series data is $16 + 12 + 1 = 29$.

Metadata features

- All **5** features for recipe data
- All **11** features for one-hot-encoded pipeline data
- All **94** features for one-hot-encoded object_id data

The dimensionality of metadata feature vector is
$5 + 11 + 94 = 110$.

The model architecture



The model architecture

Architecture details:

- Metadata feature vector size = 110
- Time series data dimension = 29
- Hidden state size = LSTM's input size * 2 = 58
- LSTM has 2 layers with dropout rate 0.5
- Dense layer 1 (batch norm + ReLu): 168 -> 84
- Dense layer 2 (batch norm + ReLu): 84 -> 42
- Dense layer 3 (batch norm + ReLu): 42 -> 21
- Dense layer 4: 21 -> 1

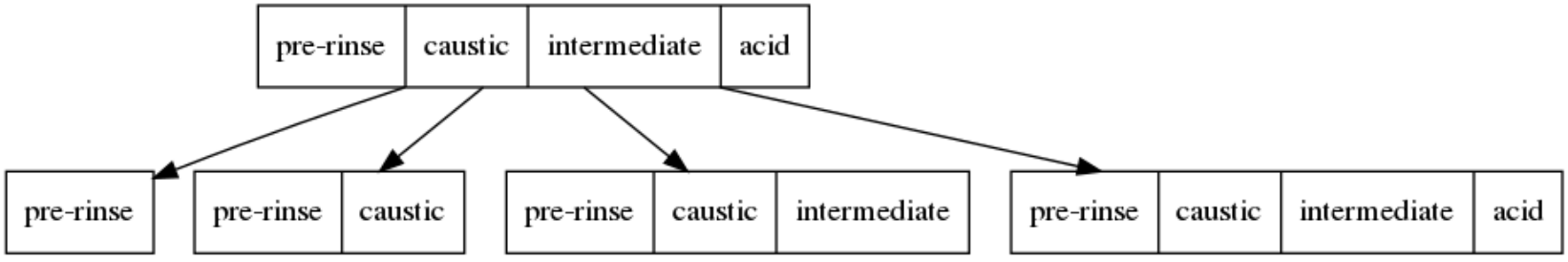
Preprocessing

LSTM uses tanh so the input data should be close to zero. I scaled the data in the following way:

- supply_flow, return_flow features and *targets* are multiplied by 10^{-6}
- return_turbidity is multiplied by 10^{-3}
- all other features are multiplied by 10^{-2}

Augmentations

- **Part crops.** The list of train examples is enlarged with artificial ones consisting of the same metadata and target along with the time series cropped up to the ends of different stage.



At test time we average the prediction values for the process and it's crops.

- **Gaussian noise.** New artificial examples are creted by adding the gaussian noise with the amplitude of 0.001 to the time series data.

Optimization

I've tried MSE and Huber losses, but my experiments showed that the best way is to optimize directly the competition metric.

The loss is optimized using Adam optimizer with learning rate 0.001, weight decay 0.0001.
The learning rate is multiplied by 0.9 every 7 epochs.
Totally optimization requires 40 epochs with batch size 64.

Advantages

- Single model. We have only one model without tons of stacking, bagging or boosting
- Light weight model. The checkpoint file with the model is only about 800Kb.
- Model flexibility. We can easily add time series data, metadata features, train on new data.
- Code flexibility. This model can be used on GPU or on CPU.
- Partial data usage. We can use only part of the time series data to predict the target value. After each phase we can add the updated data to improve the prediction (test time augmentation).

The most helpful parts of the solution

- The usage of the object_id features **improved my score by 0.01**. I think this means that something like form-factor of the object or it's usual type of product in the object matters.
- Augmentations at train time **improved my score by 0.068.**
- Augmentations at test time **improved my score by 0.0003.**
- Using MAPE as a loss **improved my score by 0.09** comparing MSE or Huber loss.