STAT430: Machine Learning for Financial Data

LarryHua.com/teaching

Spring 2019

Getting the most out of your models

Advanced architecture patterns

- residual connections
- batch normalization
- depthwise separable convolution

Residual connections

- a residual connection makes the output of an earlier layer available as input to a later layer
  - creating a shortcut in a sequential network
  - use a linear transformation to reshape the earlier activation into the target shape
- a common graph-like network component found in many post-2015 network architectures
- tackle two common problems: vanishing gradients and representational bottlenecks
- in general, adding residual connections to any model that has more than 10 layers is likely to be beneficial

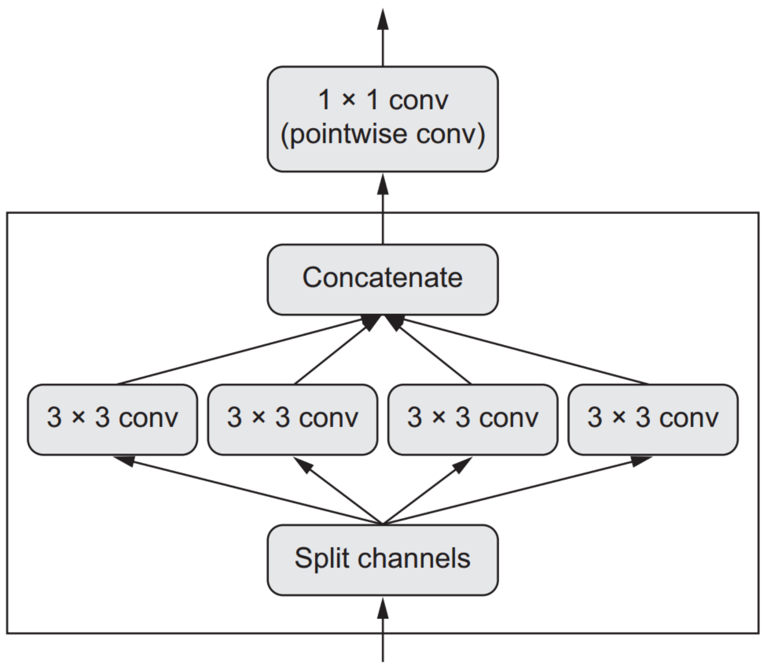
- Try R

Normalization

- ordinary normalization
- batch normalization
  - adaptively normalize data by the mean and variance that change over time during training
    - an exponential moving average of the batch-wise mean and variance of the data seen during training
  - helps with back-propagation and faster convergence, especially useful for deeper networks
  - is typically used after a convolutional or densely connected layer:
    - In Keras: `layer_batch_normalization()`

Depthwise separable convolution

- performs a spatial convolution on each channel of its input, independently, before mixing output channels via a pointwise convolution
- a special case of inception
  - separating the learning of spatial features and the learning of channel-wise features
  - useful if spatial locations in the input are highly correlated, but different channels are fairly independent
- significantly fewer parameters and fewer computations
- tends to learn better representations using less data



Depthwise separable convolution

- In Keras, `layer_separable_conv_2d()`, `layer_separable_conv_1d()`
- More references: F. Chollet, 2016, Xception: Deep Learning with Depthwise Separable Convolutions
- Try R
- LOB data: use depthwise separable 1D convolution + RNN
  - i.e., apply 1D convolution on each bid/ask level, then RNN
- Try R

Hyperparameter optimization

- currently only have access to very limited tools to optimize models
  - R package `tfruns` may be useful (will be discussed in detail after VAE)
- random search is probably the best solution for now

Model ensembling

- ensemble models that are as good as possible while being as different as possible
  - do not ensemble different runs of the models having the same architecture
- use weights for different models, and weights can be obtained by random search or optimizers
- ensemble deep learning models with shallow ML models

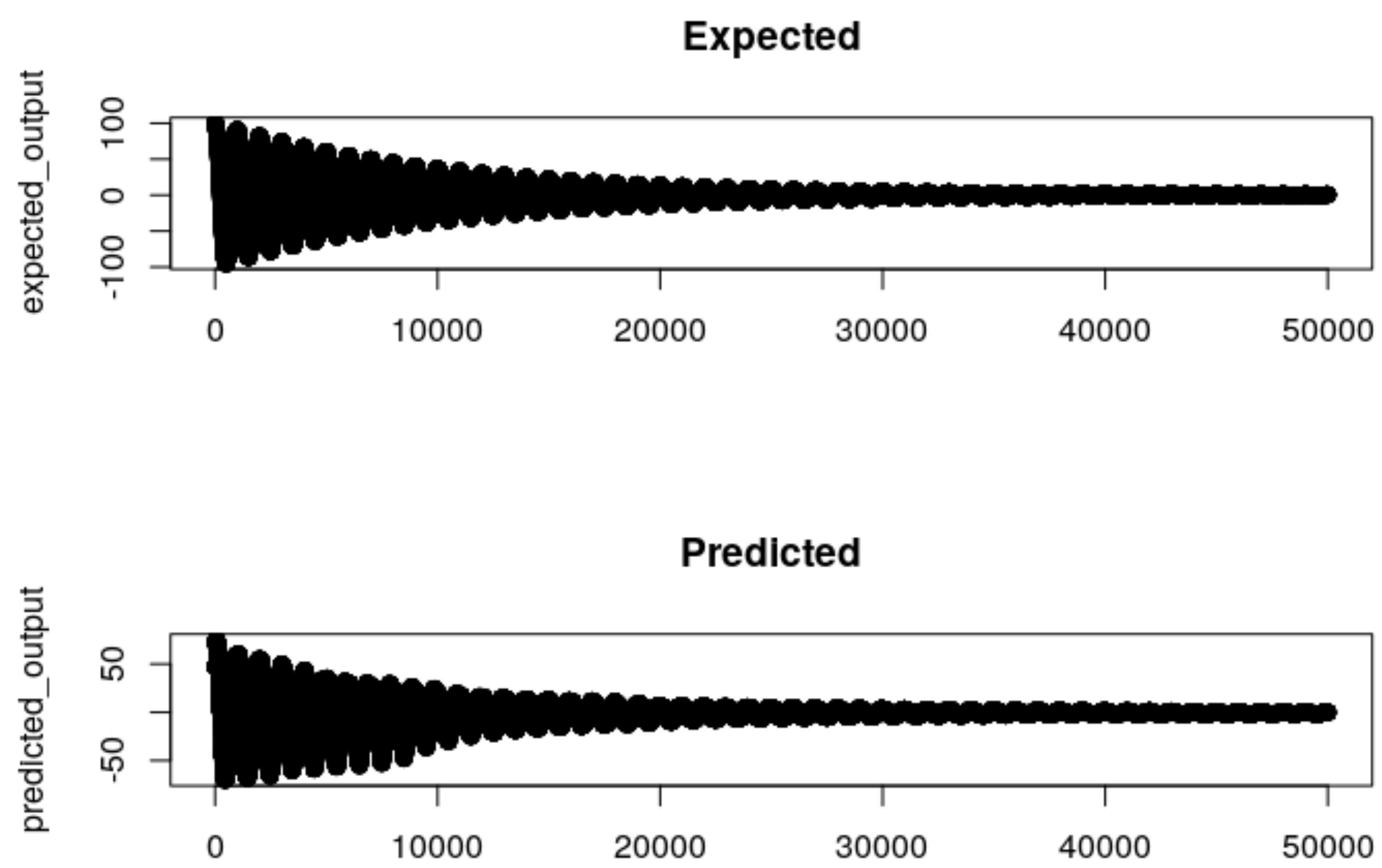
Other useful CNN and RNN for financial data

Stateful LSTM

- use the state from the previous batch instead of resetting it at the beginning of each batch
  - set `stateful=T` in LSTM layers; by default, Keras uses stateless LSTM (i.e., `stateful=F`)
  - set `shuffle=F` in `fit()`, because the order of batches should be kept same as the input
    - for customized data generators, the order of batches should be specified when the generator function is defined
  - have to specify `batch_size` in the input layer
  - the temporal dimension of the input can be very small (say, 1), because states are already kept between different batches
    - the effective time window size used for prediction is (batch size $\times$ temporal dimension)
- run `reset_states()` between epochs to clear the hidden states only; learned weights are kept
  - if `stateful=T`, need to call `reset_states()` for every epoch
  - if `stateful=F`, then `reset_states()` is automatically called between epochs

Stateful LSTM - a toy example

- predict a cosine sequence with exponentially decreasing amplitudes



- [Try R]()
- [Back to Course Scheduler]()