Predictions: Neural Architecture Search for Sequence Models

Research: Jiyao Shang, Charles Kanter. Inquiries: questions@ask2.ai

Introduction

Automated machine learning model that uses reinforcement learning to optimize neural network architecture design for return rate forecasting. We improve NAS model performance through an early stopping mechanism and architecture feasibility constraints, and collect performance metrics. Ultimately, the agent suggests an improved architecture with lower validation loss.

Data Processing

- **Data Set:** Sequential data consisting of 57,531 rows of daily excess returns from 1,330 large cap mutual funds
- **Time Period:** 01/01/2003 to 08/31/2021, including bank holidays
- **Window Length:** Window length is 60 days between two consecutive features, 30 days between two consecutive samples

Hyperparameters

- Batch size for training: 500
- Number of iterations per epoch: 85
- Maximum number of epochs to train model: 200
- Optimizer: TensorFlow optimizer "Adam"
- Layers/Parameters:

Layer Type	Layer Parameters	Parameter Values
RNN	cell_type	∈{'lstm', 'rnn', 'gru', 'bilstm'"}
	layer_depth	<= 6
	units (# of neurons)	∈{10, 20, 30}
	activation	∈{'tanh', 'sigmoid', 'relu', 'leaky_relu'}
Fully Connected	layer_depth	<=6
	fc_depth	<=1
	fc_size	∈{10, 20, 30}
	activation	∈{'tanh', 'relu', 'leaky_relu'}
	fc_depth	∈{0, 1}
Dropout	layer_depth	<=6
	proba (dropout ratio)	∈{0.1, 0.2, 0.3, 0.4, 0.5}
Terminate	terminate	∈{0, 1}

Markov Decision Process

The agent navigates through the state space eventually finding the optimal path by maximizing reward.

- **State/Action Space:** States are comprised of the layer type and its parameters, using four main layer types: Recurrent Neural Network (RNN), Fully Connected (FC), Dropout, and Terminate.
- **Epsilon-Greedy Strategy:** The agent explores a random path with probability ϵ and maximizes reward (greedy) with probability 1 - ϵ. We construct a strategy such that the agent would first explore new paths, then use its search knowledge to exploit paths for minimal loss.
- **Training/Testing:** We set data before 06/30/2017 to be training data (with the last 10% as validation) and the subsequent data to be used for testing.

Q-Learning

Reward Space: High reward is defined for this model as minimal validation loss for the chosen architecture. The Q-Learning method is a reinforcement learning algorithm that finds the optimal path by minimizing this loss:

$$Q_{t+1}(s_i, u) = (1-\alpha)Q_t(s_i, u) + \alpha\left[r_t + \gamma \max_{u' \in \mathcal{U}(s_j)} Q_t(s_j, u')\right].$$

- $Q_{t+1}(s_i,\, u)$ is a state-action function that indicates the value at state s of time i and action u, when Q is updated at time $t+1$
- $r_t(s_t,u)$ denotes reward obtained at state s and action u
- We set $\alpha = 0.1$ (Q-learning rate parameter)
- We set $\gamma = 1.0$ (discount factor)

Overfitting

Notice a decreasing training loss with consistent validation loss which suggests overfitting. We used an early-stopping mechanism to alleviate overfitting issue. Timing for the early stopping is crucial for the training process.

Conclusion

- Best architecture consisted of RNN-based layers
- Most architectures do not perform well in terms of R2 score, possibly due to overfitting, noisy data
- Proposed architecture seems to have substantial prediction power to capture the directional move of stock return

Enhancements

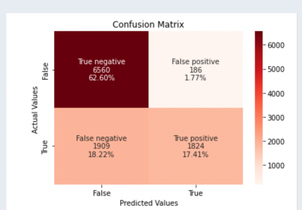
- Experimenting with training window settings
- Employ RL strategies other than Q-Learning
- Explore solutions to overfitting that are more tailored to each architecture

Performance Results

Below we display the performance results of the NAS model:

Architecture	RMSE
RNN(50,tanh), RNN(60,relu), RNN(50,relu), TERMINATE	**0.050385**
RNN(60,tanh), RNN(40,relu), RNN(40,leaky_relu), RNN(20,relu), D(0.200000), TERMINATE	0.060756
RNN(60,tanh), RNN(60,relu), RNN(50,relu), TERMINATE	0.058837
GRU(60,tanh), GRU(60,tanh), TERMINATE	0.0691
LSTM(60,tanh), BILSTM(10,leaky_relu), D(0.300000), TERMINATE	0.06328
LSTM(10,tanh), D(0.200000), FC(10, tanh), TERMINATE	0.06580

(a) Comparison of architectures based on RMSE (top performing in green)



(b) Confusion matrix for the best performing architecture

Architecture	R2 Score
RNN(50,tanh), RNN(60,relu), RNN(50,relu), TERMINATE	**0.3350**
RNN(60,tanh), RNN(40,relu), RNN(40,leaky_relu), RNN(20,relu), D(0.200000), TERMINATE	0.2856
RNN(60,tanh), RNN(60,relu), RNN(50,relu), TERMINATE	0.3300
GRU(60,tanh), GRU(60,tanh), TERMINATE	0.1182
LSTM(60,tanh), BILSTM(10,leaky_relu), D(0.300000), TERMINATE	0.1278

(c) Comparison of architectures based on R2 (top performing in green)

- **Figure A:** We compare top performing architectures, with performance determined by minimum Root Mean Square Error (RMSE). The top performer has a **decreased RMSE by 11.79%** compared with the previous model. The most successful architectures contained solely RNN-based layers, but we display some other architectures containing GRU, LSTM, or BILSTM layers for reference.
- **Figure B:** The confusion matrix reveals that the model achieves an overall percentage of correct predictions (accuracy) of 80.01%. The best architecture has an impressive False Positive Rate, which is especially important when making predictions on long-position investments.
- **Figure C:** Since noisy data is common in finance, we monitor R2 score to evaluate model fitting and find the model performs poorly for all proposed architectures. Thus, the generalization ability of the model plays a very important role, especially in a high signal-to-noise ratio financial data set.