Reinforcement Learning Based Dynamic Model Combination for Time Series Forecasting

Yuwei Fu, Di Wu, Benoit Boulet
McGill University
yuwei.fu@mail.mcgill.ca

Abstract

Time series data appears in many real-world fields such as energy, transportation, communication systems. Accurate modelling and forecasting of time series data can be of significant importance to improve the efficiency of these systems. Extensive research efforts have been taken for time series problems. Different types of approaches, including both statistical-based methods and machine learning-based methods, have been investigated. Among these methods, ensemble learning has shown to be effective and robust. However, it is still an open question that how should determine weights for base models in the ensemble. Sub-optimal weights may prevent the final model from reaching its full potential. To deal with this challenge, we propose a reinforcement learning (RL) based model combination (RLMC) framework for determining model weights in an ensemble for time series forecasting tasks. By formulating model selection as a sequential decision-making problem, RLMC learns a deterministic policy to output dynamic model weights for non-stationary time series data. RLMC further leverages deep learning to learn hidden features from raw time series data to adapt fast to the changing data distribution. Extensive experiments on multiple real-world datasets have been implemented to showcase the effectiveness of the proposed method.

Introduction

Time series forecasting is crucial for many real-world applications, such as energy prediction (Miller et al. 2020), weather forecasting (Liang et al. 2018) and inventory control (Seeger, Salinas, and Flunkert 2016). In recent years, significant progress has been made in time series forecasting with machine learning (ML) methods (Lim and Zohren 2021). A notable challenge for applying time series forecasting in real world is the non-stationary problem (Tanaka 2017). For example, in the demand forecasting domain, the erratic and intermittent data breaks some core assumptions of many ML models, such as stationarity, or i.i.d. distribution. Moreover, due to the complex and changing dynamics, there is no single forecasting model that is universal to all types of problems (Wolpert 1996).

Existing time series forecasting methods can be mainly divided into two categories, namely, statistical methods and ML-based methods (Fawaz et al. 2019). Statistical methods such as Auto-Regressive Integrated Moving Average (ARIMA) models (Zhang 2003; Pai and Lin 2005) and Exponential Smoothing State Space (ETS) models (Hyndman et al. 2008; Durbin and Koopman 2012) provide basic benchmarks for predicting individual time series. Prevalent ML-based methods mostly apply deep neural networks (DNNs) to learn complex patterns from raw data across different related time series (Rangapuram et al. 2018; Oreshkin et al. 2019).

Among these methods, ensemble learning has been proved to be an effective strategy for improving prediction accuracy (Taylor, McSharry, and Buizza 2009; Makridakis, Spiliotis, and Assimakopoulos 2018). In the ensemble learning, a strong learner is learned by combining results from multiple weak learners (Dietterich et al. 2002). Models with different capacities usually fit well in different data regimes, hence a combined model sometimes is with the potential to achieve superior performance (Zhou 2021). However, it is still an open question that how should the weights for base models in the ensemble be determined. Sub-optimal weights may prevent the final model from reaching its full potential (Weigel, Liniger, and Appenzeller 2008). In this work, we focus on the model combination problem for time series forecasting with ensemble learning.

Determining the weights for a set of base models is indeed challenging for several reasons. First, many real-life time series have complex dynamics and non-stationary data distribution. For example, the generation process of renewable energy is usually intermittent that is difficult to predict (Gowrisankaran, Reynolds, and Samano 2016). Second, many existing time series forecasting models would overfit to some specific data distributions and fail to generalize well on other data regimes (Binkowski, Marti, and Donnat 2018). An ideal model combination method should be able to learn effective representations from time series data and adapt fast to the changing distribution by selecting the most appropriate models, as illustrated in Fig 1.

In some early works, researchers used hand-engineered task-specific rules to select models (Collopy and Armstrong 1992), which require lots of domain knowledge and fail to generalize to different tasks. Later, some researchers proposed to use meta-learning (Vilalta and Drissi 2002; Nichol, Achiam, and Schulman 2018) to solve the time
The model combination problem is analyzed from the reinforcement learning perspective with some insights.

- Our model achieves state-of-the-art performances on various public benchmarks.

The rest of this paper is organized as follows. In Section 2 and Section 3, we provide the backgrounds and introduce some related work. We then introduce the proposed RLMC framework in the Section 4. Experiments are conducted in Section 5 on several real-world datasets. We conclude the paper in Section 6.

**Background**

**Time Series Forecasting**

We consider the time series forecasting problem with discrete time points. At time \( t \), the task is to use a length-\( T \) observed series history \( X^t = \{x_1^t, \cdots, x_T^t | x_i^t \in \mathbb{R}^{d_x}\} \) to predict a vector of future values \( Y^t = \{y_1^t, \cdots, y_H^t | y_i^t \in \mathbb{R}^{d_y}\} \), where \( \mathbb{R}^{d_x} \) and \( \mathbb{R}^{d_y} \) are the dimensions of the input/output data, and \( H \) is the forecast horizon. Time series forecasting tasks can be further categorized into two fundamentally different cases: time series data task and panel data task. In the time series data task, we are always predicting for the same time series; while in the panel data task, we need to predict for multiple different time series at the same time. Furthermore, depends on the prediction output to be a point forecast or a distribution, the time series forecasting task can also be divided as point forecasting and probabilistic forecasting.

**Reinforcement Learning**

Reinforcement learning is usually formulated as a Markov Decision Process (MDP), which can be defined as a tuple \( \mathcal{M} := (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma) \), where \( \mathcal{S} \) is the set of states and \( \mathcal{A} \) is the set of actions, \( \mathcal{P}(s'|s, a) \) represents the dynamics function, \( r(s, a) \) represents the reward function, and \( \gamma \in [0, 1] \) is the discount factor. The goal of an RL agent is to learn a policy \( \pi(a|s) \) that maximizes the cumulative discounted rewards \( R_t = \sum_{k=0}^{L} \gamma^k r_{t+k} \), where \( L \) is the length of the
horizon. Depends on whether we have the access to the dynamics model \( P(s'|s, a) \) and reward function \( r(s, a) \), RL can be classified as model-free methods (Silver et al. 2014) and model-based methods (Sutton and Barto 2018). Model-free RL algorithms learn a policy purely from the transitions collected by interacting with the environment, while model-based RL algorithms can leverage the dynamics model to generate transitions to optimize the policy. Based on whether the control policy is modeled directly, RL algorithms can also be categorized into value-based methods (Mnih et al. 2015) and policy-based methods (Schulman et al. 2015). In the value-based method, we aim to approximate the optimal value functions to select actions, while the policy-based method search directly for the optimal policy parameters.

**Related Work**

Traditional time-series methods based on linear autoregression or exponential smoothing (Hyndman and Athanasopoulos 2018) usually work on a few numbers of time-series at a time. However, these methods can hardly be scalable to real-world problems with a large amount of training samples. Due to the ability to model non-linear temporal patterns, deep neural networks, especially the recurrent neural networks (RNN) (Lai et al. 2018), dilated convolutions (Oord et al. 2016) and Transformers (Zhou et al. 2021), have gained popularity in time-series forecasting problems.

To solve model combination problem for time-series data, many early works (Collopy and Armstrong 1992; Armstrong, Adya, and Collopy 2001) use hand-crafted rules to select models for prediction. However, rule-based methods rely heavily on expert knowledge and are limited to the specific tasks. Moreover, previous methods mostly either use the model performance or time-series data feature to select model. For example, (Cerqueira et al. 2017; Sánchez 2008) use some heuristics to weight base models according to their model. For example, (Cerqueira et al. 2017; Sánchez 2008) the model performance or time-series data feature to select specific tasks. Moreover, previous methods mostly either use the model performance or time-series data feature to select model. For example, (Cerqueira et al. 2017; Sánchez 2008) use some heuristics to weight base models according to their recent performances. On the other hand, FFORMS (Talagala et al. 2018) formulates the model selection problem as a classification problem, in which it trains a random forest to select the best model based on the input time-series features. FFORMA (Montero-Manso et al. 2020) later extends FFORMS to output continuous weights to form a weighted forecast combination. Another line of research is to formulate the model combination problem as an RL problem. For example, DMS (Feng, Sun, and Zhang 2019) learns a Q-learning agent to solve the model selection problem in which the performance ranking improvement is used as the reward. However, DMS needs to learn a tabular Q-table for each rolling window with model index as input state, which requires learning thousands of individual Q-tables for large scale problems and fail to leverage the information from the time series data.

**Methods**

In this section, we first present the MDP formulation for the model combination problem. Then, we discuss the insights about the model combination problem from the reinforcement learning perspective. Lastly, we present the proposed RL-based dynamic model combination method.

**MDP Setting for Model Combination Problem**

Determining the weights for base models in a dynamic way can be treated as a sequential decision-making problem. The MDP \( \mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma) \) for the model selection problem can be summarized as:

- **State-space** \( \mathcal{S} \). State \( s_t \in \mathbb{R}^{T \times d_s} \) describes the information about the time series at timestep \( t \), where \( T \) is the input sequence length and \( d_s \) is the input dimension.
- **Action-space** \( \mathcal{A} \). Action \( a_t \in \mathbb{R}^N \) is the non-negative model weights that sum to one for the \( N \) base models at timestep \( t \).
- **Transition dynamics** \( \mathcal{P}(s_{t+1}|s_t, a_t) = \mathcal{P}(s_{t+1}|s_t) \) is actually irrelevant to the output model weights, which means that our action \( a_t \) will not affect the next state \( s_{t+1} \) in the model combination problems.
- **Reward function** \( r(s, a) \). Reward \( r_t \) is defined as the prediction performance, i.e., prediction error or rank performance at timestep \( t \).
- **Discount factor** \( \gamma \in [0, 1] \) describes how much we weigh for future performance. If we only care about one-step prediction, then we can set \( \gamma = 0 \).

The value of a state \( s \) is defined as:

\[
V^\pi(s) = \mathbb{E} \left[ \sum_{t=0}^{L} \gamma^t r(s_t, a_t)|s_0 = s, a_t \sim \pi(a_t|s_t) \right] ,
\]

where \( L \) is the horizon length. The objective is to find a policy \( \pi(s|a) \) that maximizes the expected value of the states from the initial state distribution \( \mu \). In our problem, \( \mu \) is the uniform distribution across the samples in the test dataset.

\[
J(\pi) = \mathbb{E}_{s \sim \mu} [V^\pi(s)] .
\]

**Insights from An RL Perspective**

Unlike prevalent RL testbeds, such as playing video games, the model combination problem has some unique properties. Here, we provide two insights from the reinforcement learning perspective.

**Insight 1: Model-based exploration.** Our first insight comes from the notice of the decoupling of state and action in the transition dynamics \( \mathcal{P}(s_{t+1}|s_t, a_t) = \mathcal{P}(s_{t+1}|s_t) \). If we treat the transition dynamics \( \mathcal{P}(s_{t+1}|s_t) \) as deterministic by only using samples from the training set, then we are actually in a model-based setting where the user-defined reward function \( r(s_t, a_t) \) is also known. This insight indicates that we can generate arbitrary transitions \((s_t, a_t, r_t, s_{t+1})\) with \( \mathcal{P}(s_{t+1}|s_t) \) and \( r(s_t, a_t) \) to facilitate exploration.

**Insight 2: Sparse optimal actions.** Let us consider a special case of the model combination problem, where we only select one model at a time. Then, the optimal policy is deterministic and outputs an one-hot vector unless there are multiple models that are equally well. In this case, a policy-based method might be problematic, where we need to compute an estimator \( \hat{g} \) of the policy gradient and optimize it with gradient ascent algorithm (Schulman et al. 2015):

\[
\hat{g} = \mathbb{E}_t \left[ \nabla_\phi \log \pi_\phi(a_t|s_t) \hat{A}_t \right],
\]
where $\hat{A}_t$ is an estimator of the advantage function at timestep $t$. Because the optimal policy $\pi^*$ is nearly deterministic, which always selects the best model. Hence, most action probabilities $\pi(a_t|s_t)$ will be close to zero except for the optimal action $a^*$. When we use samples to approximate Eq 3 during training, the log-probability is likely to be close to minus infinity. Though we have multiple optimal actions in the common model combination problem setting, however, the potential explosive log-probability problem still exists due to the sparse optimal actions. Fig 2 demonstrates this problem by plotting the $|\log \pi_{\phi}(a_t|s_t)|$ during training a policy-gradient agent on the ETT dataset (Zhou et al. 2021).

Model Combination with DDPG

Given the potential explosive log-probability issue of policy-based method, we develop our model based on an off-policy actor-critic algorithm, DDPG (Lillicrap et al. 2015). We select DDPG for the following reasons:

- It outputs continuous actions that fit well for the model combination problem.
- It is a model-free off-policy algorithm which can be trained with logged data for better sample efficiency.

Some other continuous control algorithms such as TD3 (Fujimoto, Hoof, and Meger 2018) and SAC (Haarnoja et al. 2018) can also be used in our RLMC framework.

In DDPG, we concurrently learn a deterministic policy $a = \pi_{\phi}(s)$ as the actor (Silver et al. 2014) and an action-value function $Q_{\phi}(s, a)$ as the critic, where $\pi_{\phi}(s)$ aims to directly approximate the optimal action $a^*$, and $Q_{\phi}(s, a)$ tries to approximate the optimal action-value function $Q^*(s, a)$. For the critic, we use a standard Bellman update by minimizing the mean-squared Bellman error (Sutton and Barto 2018):

$$\min_{\theta} \mathbb{E}_{(s, a, r, s') \sim \mathcal{D}} \left[ (y - Q_{\theta}(s, a))^2 \right], \quad (4)$$

where target $y = r + \gamma Q_{\phi}(s', a')|a' = \pi_{\phi}(s)$ is computed by the target network $Q_{\phi}(s, a)$. For the actor, we perform gradient ascent to maximize the expected return:

$$\max_\pi \mathbb{E}_{s \sim \mathcal{D}} [Q_{\theta}(s, \pi_{\phi}(s))]. \quad (5)$$

RL Based Model Combination (RLMC)

RLMC agent. Inspired by some previous works (Feng, Sun, and Zhang 2019; Montero-Manso et al. 2020), at each timestep $t$, the RLMC model takes a combination of time series data $X^t = \{x_1^t, \cdots, x_T^t\}$ and the history of model performance $L^t = \{L_{t-1}^1, \cdots, L_{t-1}^N\}$ as model inputs $s_t = (X^t, L^t)$, where $X^t$ is the length-$T$ observed series history at timestep $t$, and $L^t$ is the history base model performance at previous timestep. Specifically, the actor $\pi_{\phi}(s_t)$ adopts the dilated causal convolutions (Franceschi, Dieuleveut, and Jaggi 2019) as the basic encoder structure to extract latent time series features, and uses a rank embedding table to extract the base model features. A softmax function is later used in the actor $\pi_{\phi}$ as the output layer to generate the combination weight for base models in the ensemble. An illustration of the RLMC actor is shown in Fig 3. The critic $Q_{\phi}(s_t, a_t)$ used a similar dilated CNN-based encoder structure as the actor $\pi_{\phi}(s_t)$.

Figure 3: In RLMC, the actor $\pi(s_t)$ takes the time series and history model performance as inputs to predict the combination weights.

Reward function $r(s, a)$. Given the predicted combination weights for the $N$ base models $a_t = (w_1^t, \cdots, w_N^t)$ and the base model predictions $\{\hat{y}_1^t, \cdots, \hat{y}_N^t\}$, we can then compute a reward $r_t = r(s_t, a_t) w.r.t.$ the ensemble prediction $\hat{y}_t = \sum_{i=1}^{N} w_i^t \hat{y}_i^t$, where $\hat{y}_i^t$ is the $i$-th base model prediction. Similar to (Yuan et al. 2021), we use a mixture reward function, which combines the raw forecasting metric, i.e., sMAPE, and the rank performance to measure the quality of prediction is:

$$r_t = \alpha R_t^{s\text{MAPE}} + R_t^{\text{rank}}. \quad (6)$$

where $\alpha$ is a hyper-parameter to balance the sMAPE reward and the rank reward. To make the reward function generable for problems with different scales, we normalize the reward $R_t^{s\text{MAPE}}$ and $R_t^{\text{rank}}$ to range $[-1, +1]$ by following transformations:

$$R_t^{s\text{MAPE}} = 1 - 2 \frac{\tau(\delta_t)}{9}, \quad R_t^{\text{rank}} = 1 - 2 \frac{r_t^p}{N-1} \quad (7)$$
where $\delta_t$ is the sMAPE at timestep $t$, $\tau(\delta_t) = \{0, \cdots, 9\}$ is the quantile of $\delta_t$ w.r.t. the base model prediction errors in the training set, and $r^p_t = \{0, \cdots, N - 1\}$ is the rank of the prediction error w.r.t. other $N$ base model prediction error at timestep $t$.

**Overall pipeline.** We apply the standard DQN training pipeline (Mnih et al. 2013) to train the RLMC agent with a replay buffer. For each trajectory, we first randomly select a timestep $t$ as the initial state $s_0$ from the training set. Then we use our RLMC agent to interact with the environment by selecting model weights for a horizon of $H$ steps. The reward $r_t$ for each state-action pair $(s_t, a_t)$ is computed according to the self-defined reward function (Eq 7). We store the collected transitions to a replay buffer for later training. An overview of the RLMC training pipeline is shown in Fig 4. Alg 1 describes the details of the RLMC approach.

![RLMC Training Pipeline Diagram](image-url)

**Figure 4:** Overview of RLMC training pipeline: (1) At timestep $t$, state $s_t$ describes the information of the input time series and history base model performances. (2) Our RLMC agent selects an action $a_t = (w^1_t, \cdots, w^N_t)$ as the combination weights for the $N$ base models. (3) We then compute the final ensemble prediction $\hat{y}_t$ w.r.t. base model predictions $\{\hat{y}^1_t, \cdots, \hat{y}^N_t\}$. (4) We compute the mixture reward $r_t$ with the self-defined reward function. (5) We save the transition into the replay buffer for later training. (6) We update the actor $\pi_\phi(s_t)$ and critic $Q_\theta(s_t, a_t)$ using the sampled transitions from the replay buffer.

**Strategies for Efficient Exploration**

In the training period, we first train $N$ base models $M = \{M_1, \cdots, M_N\}$ on the training set. Notably, we can select different types of algorithms, i.e., classical statistical models or neural networks, as the base models to increase the diversity. Given the $N$ pre-trained models, the action $a = (w^1, \cdots, w^N)$ is a probability simplex, such that $\sum_i w_i = 1$ and $w_i \in [0, 1]$. When the ensemble consists many different base models, we are facing a continuous control problem with an enormous search space. Therefore, a naive $\epsilon$-greedy exploration strategy may require a large number of samples to learn a near-optimal policy. Here, we introduce three techniques (Fig 5) to improve the exploration efficiency.

Firstly, we pre-train the actor $\pi_\phi(s)$ model with a multi-class classification task, in which for each state $s_t$ the index of the optimal model is used as the class label (Talagala et al. 2018). If multiple models are equally accurate, we randomly select one as the best model. We optimize the $\pi_\phi(s)$ with respect to the cross-entropy loss.

Secondly, we also provide a convex combination inductive bias to our agent to further accelerate the exploration. Notice that the final prediction $\hat{y}_t = \sum_i w^i_t \hat{y}^i_t$ is a convex combination of the individual base model predictions. We can provide our agent with the information about how good is the $i$-th base model for the state $s_t$ by directly sampling the one-hot vector $e_i$ where the $i$-th element equals one. Hence, we design a special $\epsilon$-greedy exploration strategy which actively selects more samples around the one-hot vector:

$$a = \frac{|e_i + \epsilon|}{|e_i + \epsilon|^2}$$  \hspace{1cm} (8)
where \( i \) is a random index, and \( \epsilon \) is the random noise.

Thirdly, we find that the RL-based agent would suffer from an overfitting problem if there exists a few superior base models that win all the time. The imbalanced data distribution problem would cause the RL agent to degrade to just copy the best model in the training set. Therefore, we propose to use a second replay buffer to store the hard samples with low rewards, and we train the RLMC agent by sampled transitions from both buffers to mitigate overfitting.

![Figure 5: Three techniques for better exploration. (1) We first pre-train our actor \( \pi_{\theta}(s) \) on a multi-class classification task in which the optimal model index is used as the class label. (2) Inspired by the convex combination inductive bias, we can tell the agent how good each base model is by directly sample one-hot weights. (3) To mitigate overfitting, we maintain a second buffer to store hard samples.](image)

**Experiments**

**Datasets**

To validate the effectiveness of the proposed RLMC framework, we extensively perform experiments on a number of real-world time series datasets (table 1), including both the time series data and panel data:

- **ETT** (Zhou et al. 2021) contains multivariate time series for 1-hourly-level electricity transformer temperature data, in which the target value is the “oil temperature”.

- **Climate** (Chollet 2017) is a weather time series dataset, recorded at the Max Planck Institute for Biogeochemistry, including 14 different features such as air temperature, atmospheric pressure, and humidity. We use the hourly data from 2009 to 2016 to predict the temperature.

- **GEFCOM** (Barta et al. 2017) is a dataset from the Global Energy Forecasting Competition (GEFCOM) 2014. We use an hourly dataset with two features including the load and temperature to predict future electricity load.

- **M4** (Makridakis, Spiliotis, and Assimakopoulos 2018) contains 100k panel time series data, representing demographic, finance, industry, macro and micro indicators. We use the daily subset in the experiment.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Train</th>
<th>#Dev</th>
<th>#Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>M4-Daily</td>
<td>3,381</td>
<td>845</td>
<td>4,276</td>
</tr>
<tr>
<td>Climate</td>
<td>49,063</td>
<td>14,018</td>
<td>7,010</td>
</tr>
<tr>
<td>GEFCOM</td>
<td>18,710</td>
<td>4,677</td>
<td>2,488</td>
</tr>
<tr>
<td>ETT</td>
<td>8,448</td>
<td>2,816</td>
<td>2,816</td>
</tr>
</tbody>
</table>

Table 1: Statistics of the datasets for experiment.

**Experimental Details**

Given a length-\( H \) predicted series \( \{\hat{y}_{t+1}, \ldots, \hat{y}_{t+H}\} \) and true series \( \{y_{t+1}, \ldots, y_{t+H}\} \), the following metrics are commonly used to evaluate the forecasting performance:

\[
MAE = \frac{100}{H} \sum_{i=1}^{H} |y_{t+i} - \hat{y}_{t+i}|
\]

(9)

\[
MAPE = \frac{100}{H} \sum_{i=1}^{H} \left| \frac{y_{t+i} - \hat{y}_{t+i}}{y_{t+i}} \right|
\]

(10)

\[
sMAPE = \frac{200}{H} \sum_{i=1}^{H} \frac{|y_{t+i} - \hat{y}_{t+i}|}{y_{t+i} + |\hat{y}_{t+i}|}
\]

(11)

In the experiment, we employ **Mean Absolute Error** (MAE) and **symmetric Mean Absolute Percentage Error** (sMAPE) to evaluate the performance. We use sMAPE instead of MAPE to avoid errors caused by near-zero real values.

We adopt early stopping by evaluating the model performance on the validation set. We set the exploration parameter to be 0.5, reward threshold to be \(-1\), update frequency to be 4, and Polyak update parameter to be 0.005. Each experiment is repeated for 5 times, and the average performance is reported. We run all the models on a desktop with Intel i9 CPU and single Nvidia GeForce 3090 GPU.

Our method is compare with following baselines:

- **Heuristic methods**. We first compare RLMC with different heuristic methods. The **uniform** prediction computes the simple average of base model outputs. The **single best** prediction is the result of the model that performs best in the validation set.

- **FFORMS**. We also compare RLMC to a meta-learning based baseline. FFORMS (Talagala et al. 2018) formulates the model selection problem as a classification problem, and use gradient boosted tree models to learn the meta-learner.

- **FFORMA**. FFORMA later (Montero-Manso et al. 2020) extends FFORMS to the model combination problem by outputting continuous weights. It adopts a self-defined loss function to output weights for different models. We implement both of the FFORMS and FFORMA baselines with LightGBM (Ke et al. 2017).

- **DMS**. We then compare RLMC to another RL-based method, named DMS (Feng and Zhang 2019). The original DMS uses a tabular Q-learning method without leveraging the time series features. It needs to learn a new Q-table for each feature.
Table 2: Metric $\sigma_1$ and $\sigma_2$ denote the MAE and sMAPE loss. D1, D2, D3, D4 denotes EET, Climate, Gefcom and M4 dataset.

<table>
<thead>
<tr>
<th></th>
<th>Uniform</th>
<th>Single</th>
<th>FFORMS</th>
<th>FFORMA</th>
<th>DMS</th>
<th>M3</th>
<th>RLMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_1$</td>
<td>5.22</td>
<td>79.83</td>
<td>4.59</td>
<td>73.69</td>
<td>4.89</td>
<td>75.98</td>
<td>5.42</td>
</tr>
<tr>
<td>$\sigma_2$</td>
<td>1.70</td>
<td>26.44</td>
<td>1.67</td>
<td>25.80</td>
<td>1.78</td>
<td>26.65</td>
<td>1.69</td>
</tr>
<tr>
<td>$\sigma_1$</td>
<td>112.65</td>
<td>3.51</td>
<td>89.09</td>
<td>2.75</td>
<td>95.03</td>
<td>2.94</td>
<td>86.93</td>
</tr>
<tr>
<td>$\sigma_2$</td>
<td>250.39</td>
<td>3.93</td>
<td>147.61</td>
<td>2.39</td>
<td>183.90</td>
<td>3.12</td>
<td>161.52</td>
</tr>
</tbody>
</table>

We note that DMS, M3 and RLMC usually perform better than FFORMS and FFORMA, which implies that the proposed DL-based encoder has superior ability in capturing useful time series features. Further, we can observe that other baseline methods usually fail to beat the single best baseline. We attribute this failure to the overfitting problem caused by the imbalanced best model distribution. We implement a case study on the ETT dataset by training an DQN agent in DMS. In the experiment, the third base model $M(3)$ is the best model in the training set. Fig 6 shows the percentage of transitions that select the third model as the best one. We can find that the model starts to overfit after around three epochs. Then the agent just select the best training model.

We further conduct ablation studies on the climate dataset to validate the effectiveness of the proposed techniques (Fig 7). We can find that the proposed exploration strategy is the most important ingredient for the success of RLMC. This is intuitive because the original DDPG algorithm learns a deterministic policy which is prone to suffer from the overfitting problem. Besides the special exploration strategy, the pretrain step and extra buffer techniques are also helpful for the RLMC agent to achieve better performances.

Results and Analysis

Table 2 summarizes the time series forecasting results of all methods. We can observe that the proposed RLMC model consistently outperforms other baselines on all 4 datasets. We note that DMS, M3 and RLMC usually perform better than FFORMS and FFORMA, which implies that the proposed DL-based encoder has superior ability in capturing useful time series features. Further, we can observe that other baseline methods usually fail to beat the single best baseline. We attribute this failure to the overfitting problem caused by the imbalanced best model distribution. We implement a case study on the ETT dataset by training an DQN agent in DMS. In the experiment, the third base model $M(3)$ is the best model in the training set. Fig 6 shows the percentage of transitions that select the third model as the best one. We can find that the model starts to overfit after around three epochs. Then the agent just select the best training model.

We further conduct ablation studies on the climate dataset to validate the effectiveness of the proposed techniques (Fig 7). We can find that the proposed exploration strategy is the most important ingredient for the success of RLMC. This is intuitive because the original DDPG algorithm learns a deterministic policy which is prone to suffer from the overfitting problem. Besides the special exploration strategy, the pretrain step and extra buffer techniques are also helpful for the RLMC agent to achieve better performances.

Conclusion

Accurate analysis and forecast of time series data can be of significant importance for real-world systems. In this paper, we proposed a general reinforcement learning (RL) based model combination framework, named RLMC, for time series forecasting. We first analyzed the problem from the RL perspective and provide two insights. We then designed an off-policy method based on DDPG with some special strategies for effective exploration. The effectiveness of RLMC is validated by experiments on real-world data. One limitation of our method is that the RLMC agent only outputs deterministic policy due to the use of DDPG. This may limit the full potential of RLMC when the dataset is extremely imbalanced. In the future, we plan to investigate how to combine some unsupervised time series representation learning methods with our framework.


