國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

以基於時間比重之回歸預測電視收視率

TV Ratings Prediction with Time Weighting Based Regression

顧廷緯

Ting-Wei Ku

指導教授：林守德 博士

Advisor: Shou-De Lin, Ph.D.

中華民國 104 年 1 月

Jan 2015

國立臺灣大學電機資訊學院資訊工程學系

國立臺灣大學電機資訊學院資訊工程學系
碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

以基於時間比重之回歸預測電視收視率
TV Ratings Prediction with Time Weighting Based Regression

顧廷緯
Ting-Wei Ku

指導教授: 林守德 博士
Advisor: Shou-De Lin, Ph.D.

中華民國 104 年 1 月
Jan 2015

國立臺灣大學（碩）博士學位論文
口試委員會審定書

以基於時間比重之回歸預測電視收視率
TV Ratings Prediction with Time Weighting Based Regression

　　本論文係顧廷緯君（r01922060）在國立臺灣大學資訊工程學系、所完成之碩士學位論文，於民國 104 年 1 月 30 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

　　　　　　　　　　　　　　　　　　　　　　　　（簽名）

（指導教授）

系主任、所長 _____ （簽名）
（是否須簽章依各院系所規定）

誌謝

Many thanks to the following people that help me complete this research:

- Shou-De Lin
- Yu-Yang Hung
- Yu-An Yen
- Tim Chen
- Eric Yang

I also want to thank my parents who gave me the freedom to complete research at my own pace.

Finally, thank God for giving my opportunity to meet all the people above.

中文摘要

此論文主要貢獻為提出一個簡單且實驗結果準確的電視收視率預測方法，名為 Time Weighting Regression (TWR)。基於「越新的資料對預測接下來的收視率越重要」的假設，TWR 主要做的事情為：根據資料的時間賦予權重，再以帶有權重的資料建立回歸模型，最後用建立的模型預測接下來的收視率。我們以真實世界的電視收視率資料進行實驗，結果顯示它的預測比知名的時間序列模型（例如 Exponential Smoothing 和 ARIMA）和回歸模型（類神經網路）還準。

關鍵詞：時間序列預測、電視收視率預測、回歸。

英文摘要

In this thesis, the primary contribution is proposing a simple and experimentally accurate solution, named Time Weighting Regression (TWR), to the problem of TV ratings prediction. Based on the assumption that newer data are more important for predicting upcoming ratings, what TWR does is: weighing data based on time, and then using weighted data to build regression model for predicting upcoming ratings. In the experiments on a real-world TV ratings data set, it outperforms well-known time series models (e.g., Exponential Smoothing and ARIMA) and regression model (neural network).

Keywords: time series prediction, TV ratings prediction, regression.

目錄

圖目錄

表目錄

第一章　Introduction

1.1 Contribution

The main contributions of the thesis are summarized as below:

- We propose a solution named Time Weighting Regression (TWR) to improve the results of TV ratings prediction.
- We conduct experiments on a real-world TV ratings data set to compare TWR with well-known time series models (e.g., Exponential Smoothing and ARIMA) and regression model (neural network). Experiment results show that TWR gives us the best overall results (lowest overall errors in terms of MAPE and MAE).

1.2 Background and importance of problem

Why is TV ratings prediction important? In TV industry, because the price of advertising time is mainly based on ratings, predicting ratings accurately is very important to broadcasters and advertisers. Accurate predictions help them make money, while inaccurate predictions cause money loss.

How complex is TV ratings prediction? First, it is complex because TV ratings is an aggregated measurement of a lot of people's watching choices. Second, generally speaking, TV is competing with every other platform/service that tries to grab eyeballs. As more such platforms/services appear and grow, the competition becomes more complex, which probably makes accurately predicting TV ratings increasingly difficult. For example, we may need to consider whether people prefer watching YouTube videos via mobile to watching programs via TV.

Although TV ratings prediction seems important and complex, it may become less and less important and complex. In [1], it argues that TV industry is dying because people are switching from TV to other devices such as mobile which enables people to watch content from the Internet. If TV is no longer a popular media, it is not important to predict TV ratings accurately.

1.3 Overview of problem and solution

In this thesis, our goal is to accurately predict the ratings of an upcoming episode. Because ratings of a TV program is a time series, the problem can be viewed as one-step time series forecasting. When it comes to solutions to time series forecasting, commonly used models can be divided into two main categories: time series models (e.g., ARIMA) and regression models (e.g., neural

network), where our proposed solution (TWR) falls into the latter. In addition to model, features used by model also largely affect prediction, but we choose to only use historical ratings of a TV program as features to avoid losing focus. However, TWR can incorporate any external features to improve predictions whenever appropriate, which is superior to some time series models (e.g., Exponential Smoothing) that can only take features of historical time series values.

How does TWR work? Briefly speaking, it weighs newer training instances more (the most important part), and then uses weighted instances to build a base model for predicting upcoming ratings. Recall that in machine learning, an instance is a record in data set, consisting of features and labels. In our problem, features are historical ratings, and label is current ratings. Details of how TWR works are described in section 3.

Why TWR works? That is, why the step of weighing training instances based on time in TWR gives us better predictions? We think it is because the following assumption holds: newer training instances are more important for predicting upcoming ratings. This assumption comes from our own experience on watching TV programs (especially dramas because our data set is ratings of dramas). When watching dramas, I don't watch them due to past episodes long time ago, but I do watch them due to their recent episodes. For example, I may watch a drama because I hear that it is hot recently.

1.4 Motivation of solution

TWR is motivated by a well-known time series model called Simple Exponential Smoothing (SES). SES's idea is that forecast is the weighted average of all past observations, and the weights decay exponentially as observations get older. The key difference is that TWR weighs training instances by time, while SES weighs training features (past observations) by time. Another difference is that TWR automatically tries different types of growth functions and decides the best one, not only the exponential function, which is the case of SES.

第二章　Related Works

2.1 TV ratings prediction

In this section, we briefly describe 5 related works of TV ratings prediction chronologically. For each related work, a list of highlights and how it differs to our work are provided.

1. [2] Forecasting television ratings (IJF 2011)
   - Used large real-world data set to compare forecasting capability of 8 regression models, including Bayesian model averaging, a state-of-the-art model.
   - Suggested useful external features such as seasonal factors and program genre.
   - Found that modeling ratings directly is better than as total_audience×channel_share.
   - Data set: 70 channels, 5,000 programs and 48,000 ratings from 2004-2008.
   - Differences between it and our work: It has very different characteristic of data set in terms of TV programs, which makes no sense for it to compare with our work in terms of performance. Concretely, in their data set, many TV programs do not run continuously every day or week, and many programs (e.g., movies) are broadcast only once, so they have no time series information. On the other hand, in our data set, all TV programs are weekly dramas that have time series information.
2. [3] Using a nested logit model to forecast television ratings (IJF 2012)
   - Applied a little-known version of the nested logit model that is suitable for aggregate choice decision data (TV ratings are aggregate measures) to TV ratings prediction.
   - Extended model to include TV program random effects, and developed a novel method for predicting program random effects for programs that have not previously been broadcast.
   - Data set: Same as [2].
   - Differences between it and our work: Same as the differences between [2] and our work because it used same data set as [2].
3. [4] Predicting TV audience rating with social media (SocialNLP 2013)
   - Included external features from Facebook such as count of posts and number of likes on the fan page of dramas.
   - Fit data with neural network.
   - Data set: 4 weekly dramas with 78 ratings broadcast in Taiwan.
   - Differences between it and our work: Its focus is the usefulness of external features from Facebook, while the focus of our work is model not feature. We can achieve competitive performance only using historical ratings to make our solution more general, and thus easily applied to new dramas. Not every drama has a fan page, but every drama has ratings.
4. [5] A predicting model of TV audience rating based on the Facebook (SocialCom 2013)
   - Almost same as [4].
5. [6] A weight-sharing Gaussian process model using web-based information for audience ratings prediction (TAAI 2014)
   - Proposed a novel Gaussian process model.
   - Included external features from Facebook and Google, such as opinion of comments from Facebook fan page and popularity of dramas' search terms from Google Trends.

- Data set: 4 daily dramas with 336 ratings broadcast in Taiwan.
- Differences between it and our work: Its data set is long daily dramas, while ours is short weekly dramas. Also it focused on the usefulness of external features, while we focus on the usefulness of our proposed model TWR.

2.2 Models compared to TWR in experiments

We compare our solution with 7 models which can be categorized into 3 categories: (1) naïve guess, (2) well-known time series models, and (3) advance regression model. All the competitors along with our solution are summarized in Table 1. In Table 1, the 4th category is our solution with different settings.

Table 1. List of models

#	Category	Name	Summary
1	1	Previous period (PP)	Guess value of the last period
2	1	Past average (PA)	Guess average of all the previous
3	2	Simple Exponential Smoothing (SES)	Exponential weighted average
4	2	Double Exponential Smoothing (DES)	SES with trend
5	2	Exponential Smoothing State Space (ETS)	Best one from 30 state space models
6	2	ARIMA	A model of autocorrelations
7	3	Neural network auto-regression (NNA)	Feed-forward with a hidden layer
8	4	TWR with no growth (TWR.N)	It equals to no TWR at all.
9	4	TWR with linear growth (TWR.L)	$g(x) = x$
10	4	TWR with exponential growth (TWR.E)	$g(x) = e^x$
11	4	TWR with e^{3x} growth (TWR.E3)	$g(x) = e^{3x}$
12	4	TWR with auto-selected growth (TWR.A)	Pick growth with min validation error.
13	4	TWR.A with external features (T.A.EF)	Include opinion polarity features.

In the rest of this section, we briefly describe how each competitor works, their parameters settings used in our experiments, and their implementation.

We choose language R [11] as our implementation platform. For most models, we just use the published packages in the official R repository, so-called Comprehensive R Archive Network, CRAN. For models that don't have published packages, they are implemented by ourselves.

Each competitor works as below (recall that **x** is the time series of ratings):
1. **Previous period**: the forecast is exactly the ratings of previous episode.
   - Equation: $x_{t+1} = x_t$
   - Implementation: Our own implementation in R

2. **Past average**: the forecast is the average ratings of all previous episodes.
   - Equation: $x_{t+1} = (x_t + x_{t-1} + ... + x_1) / (t-1)$
   - Implementation: Our own implementation in R
3. **Simple Exponential Smoothing (SES) [10]**: the forecast is the weighted average of all previous ratings, and the weights decay exponentially as ratings get older.
   - Equation: $x_{t+1} = \alpha x_t + \alpha(1 - \alpha)x_{t-1} + \alpha(1 - \alpha)^2 x_{t-2} ...$, where α is the smoothing parameter.
   - Settings: α is determined by minimizing the squared prediction error.
   - Implementation: package HoltWinters {stats} in R [11]
4. **Double Exponential Smoothing (DES) [12]**: the forecast extends SES with estimated trend (b^t).
   - Equation: $x_{t+1} = a_t + b_t,$

     $a_t = \alpha x_t + \alpha(1 - \alpha)(a_{t-1} + b_{t-1}),$

     $b_t = \beta(a_t - a_{t-1}) + (1 - \beta)b_{t-1}$, where α, β are the smoothing parameters.
   - Settings: α, β are determined by minimizing the squared prediction error.
   - Implementation: package HoltWinters {stats} in R [11]
5. **Exponential Smoothing State Space (ETS) [13]**: automatically select the best exponential smoothing model according to information criterion from 30 state space models. State is defined by the unobserved error, trend, and seasonal components. The possibilities for each component are: Error = {A, M}, Trend = {N, A, A_d, M, M_d} and Seasonal = {N, A, M}, where A is additive, M is multiplicative, N means no trend or seasonal component, and subscript d means damped trend.
   - Equation: Please refer to 附錄 Appendix.
   - Settings: all parameters are determined via information criterion AICc.
   - Implementation: package ets {forecast} in R [15]
6. **ARIMA [14]**: Autoregressive Integrated Moving Average model, where integrated is the reverse of differencing of time series.
   - Equation: $x_{t+1} = a_1 x_t + ... + a_p x_{t-1-p} + e_{t+1} + b_1 e_t + ... + b_p e_{t-1-q}$, where a, b are the fitted coefficients for autoregressive and moving average models, e is the unobserved error term, and p, q are the orders of autoregressive and moving models. If having differencing, the differenced time series (d times) is fitted.
   - Settings: all parameters (p, q, d) are determined via information criterion AICc.
   - Implementation: package auto.arima {forecast} in R [15]
7. **Neural network auto-regression (NNA)**: feed-forward neural networks with a single hidden layer and past observations as inputs. 20 models are built and then averaged when making prediction.
   - Equation: Neural network equations with past observations as inputs. Please refer to 附錄 Appendix for detail.
   - Settings: the optimal number of past observations is determined according to the AIC for

a linear autoregressive model with order p model.
- Implementation: package nnetar {forecast} in R [15]

第三章　Method

In this section, we describe our proposed solution (TWR) in detail. We first provide a summary of TWR. Then in the following sub-sections, pseudo-code and details of steps in TWR are provided.

As every learning algorithm, TWR consists of two stages: fitting and predicting, i.e., building model with training data and making prediction on testing data with trained model, respectively.

At stage of fitting, it consists of three main steps: (1) transforming a time series into a set of instances suitable for regression analysis with a window size (this step is known as windowing transformation [7]), (2) weighing training instances with a growth function, and finally (3) building a model for one-step forecasting with a base learning algorithm and weighted training instances.

At stage of predicting, it makes a one-step forecast by providing trained model with input features from training data. Although multi-step forecasts are out of our problem scope, they can be computed recursively by treating forecasts as input features, e.g., taking one-step forecast as input to make the second-step forecast.

Figure 1. Activity diagram of TWR
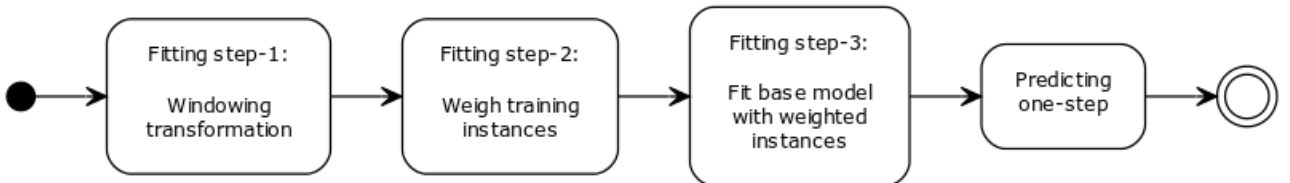
3.1 Pseudo-code of TWR

Input data: A time series **x** with length **t**
Parameter: Window size **w**, growth function **f**, a base algorithm **Learner** with its parameters p
(For illustration, let $\mathbf{n} = 5$, $\mathbf{w} = 3$, $\mathbf{f} = e^x$, where e^x is the exponential function.)

Fitting process:
1. Windowing transformation with window size **w**:
   $\mathbf{x} = \{ x_1, x_2,..., x_5 \} \rightarrow \mathbf{X} = \{ \mathbf{w_1} = (x_1, x_2, x_3=y_3), (x_2, x_3, x_4=y_4), (x_3, x_4, x_5=y_5) \}$
   We get 3 training instances, with each having 2 input features and 1 label.
2. Weighing training instances with growth function **f**: $\mathbf{W} = \{ e, e^2, e^3 \}$
3. Building a base model with weighted instances: $\mathbf{m} = \mathbf{Learner.Fit}(\mathbf{X}, \mathbf{W}, p)$

Output: base model **m**

Predicting process: Input data $w_4 = (x_4, x_5, x_6=y_6)$, where $x_6=y_6$ is the unknown label to predict. Output: one-step forecast $x_6' = $ **Learner.Predict**(**m**, w_4)

3.2 Fitting step 1: Windowing transformation

Before applying a regression model to time series forecasting, first we need to transform a time series into a set of instances with a window size. Therefore, in this step, we need to decide window size. It can be decided either subjectively by people, or by model-selection techniques such as AIC (Akaike information criterion) and cross-validation. In our experiments, we decide to use AIC for fairness because other time series models also use it or similar model-selection technique, e.g., ARIMA uses AICc, a variation of AIC, to decide how many past observations to include as features.

3.3 Fitting step 2: Weighing training instances

This step is to weigh training instances (output of previous step) with a growth function that match our assumption (newer instances are more important). Mathematically speaking, because there are infinite strictly increasing functions, there are infinite valid choices. Take illustration in 3.1 for example, the importance order is $w_1 < w_2 < w_3$. So, what is the best growth function and how to find it from infinite possibilities? First, we define the best growth function as the one that minimizes the testing error of one-step forecast. Second, we consider three growth functions: **linear** (x = { 1, 2, 3,… }), **exponential** (e^x), and **cubic exponential** (e^{3x}) growth. Because testing error is unknown during training, we choose the growth function that minimizes the validation error of one-step forecast. Although other growth functions should be considered, the three functions above are enough for our data set.

After a growth function is decided, there are two ways to weighs training instances: resampling instances with weights or infuse weights into the fitting process of base learning algorithm. Not all learning algorithms support instance weights during fitting, but resampling always work because it is an approach at data level. Since we want TWR to be general, we choose resampling.

There is a small side benefit by using resampling: it narrows down the search space of growth functions. By resampling training instances with given weights as probability (sampling with replacement, i.e., bootstrap sampling), many weights assignments actually produce the same resampling result. For example, e^{3x} and e^{4x} produces probability p3 = { 0.01%, 0.23%, 4.73%, 95.02% }, p4 = { 6.03e-2%, 0.03%, 1.79%, 98.16% }, respectively. By resampling 4 instances with p3 and p4, it is very likely that their results are the same because over 95% the 4th instance is sampled.

Although resampling has its advantages, it also has a downside: it introduces randomness and variance into results. To avoid this side effect, bagging [8] is used, i.e., we resample many data sets from the same probability, build a base model from each data set, and then aggregate all the base models to make prediction. In our experiment, 20 models are built and they are averaged when making predictions.

3.4 Fitting step 3: Building a base model

This step is just calling a base learning algorithm to make it fit on the weighted data. Therefore, we need to make decisions of base learning algorithm and its parameters. Although any regression model can be used, regression tree should be our best choice because it is sensitive to different data [9], i.e., a small change in the data set may cause a large change in its built model. This characteristic is important for the weighted data to make impact on the built model and prediction.

As for parameters of regression tree, our principle is to use default settings as much as possible, and allow tree to grow as depth as possible, and prune it based on validation error. Detail settings are provided in section 4.

3.5 Predicting stage of TWR

Because TWR uses bagging with sampling probability as growth function, many base models are averaged to make prediction. Note that because TWR is trained on weighted instances, it is very likely to produce bad predictions for most past observations, especially for those that are not sampled. TWR is designed to make one-step forecast only, but it can also compute multi-step forecasts by recursively treating forecasts as input features. However, if using for multi-step forecasts, the number of episodes for validation also needs to increase.

第四章 Experiments

In this section, we describe data set, evaluation metric, models, and results.

4.1 Data set

Our data set contains 8 weekly Idol dramas broadcasting in Taiwan. They are so-called Nielsen ratings, which is the most frequently used ratings in TV industry. Normally, the ratings are only available for Nielsen's customers. Fortunately, some of them are announced in news and organized into Wikipedia, which is the case of all the dramas in our data set.

Simple data analysis results are presented via Table 2, Figure 2 and Figure 3. From the time series plot (Figure 2), the following things are observed:

- D2 and D7 have clear increasing trend, while all the others don't have any obvious increasing or decreasing trend.
- There is no obvious seasonal or periodic component for ratings of all dramas, so there is no need to consider modeling ratings via seasonal decomposition or any seasonal models such as Triple Exponential Smoothing (also known as Holt-Winter's seasonal method).
- D1 has the lowest ratings over the time. In fact, its ratings are close to zero.

From the box plots (Figure 3), the following things are observed:

- D2 and D7 has much wider ranges of ratings than all the other dramas. This is also reflected from the standard deviation of ratings in Table 2. It is likely that the wider the range of ratings, the more complex to predict ratings accurately.
- There is only 1 outlier in D4 (the 5th episode).

Table 2. Basic information about dramas

	D1	D2	D3	D4	D5	D6	D7	D8
Name	大紅帽	小資女孩	向前走	金大花	真愛黑白	國民英雄	犀利人妻	螺絲小姐
# Episode	16	25	22	21	21	19	23	23
Start(Y/M/D)	13/2/28	11/8/21	12/2/19	13/1/6	13/6/9	10/12/5	10/11/5	12/7/22
Avg – ratings	0.21	5.12	2.38	1.57	2.16	1.10	3.36	3.47
Std – ratings	0.08	1.09	0.16	0.23	0.30	0.21	2.75	0.56

Figure 2. Time series plot for ratings of dramas
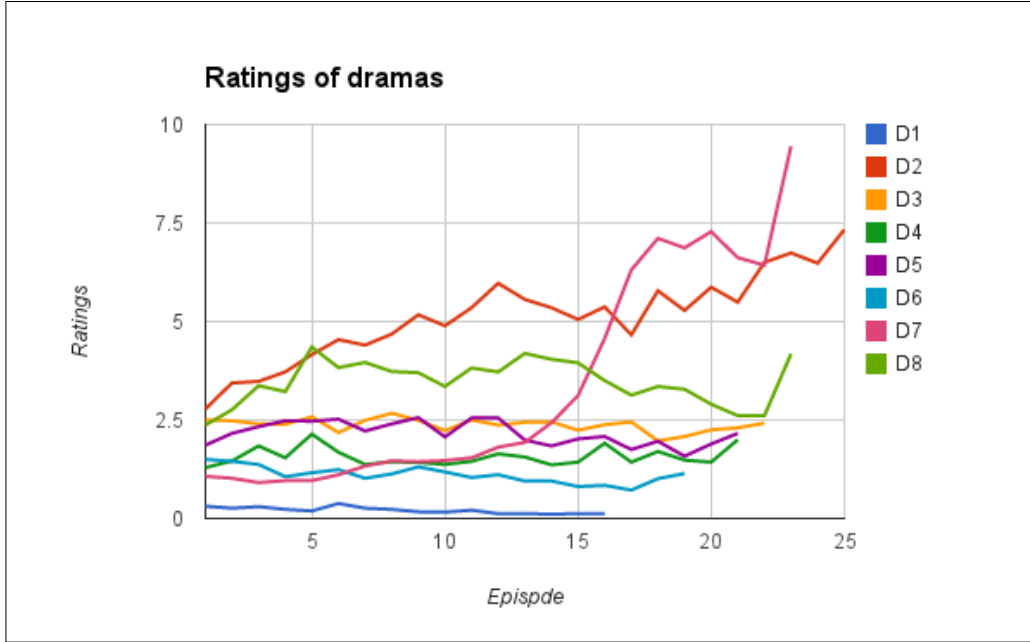
Figure 3. Box plots for ratings of dramas
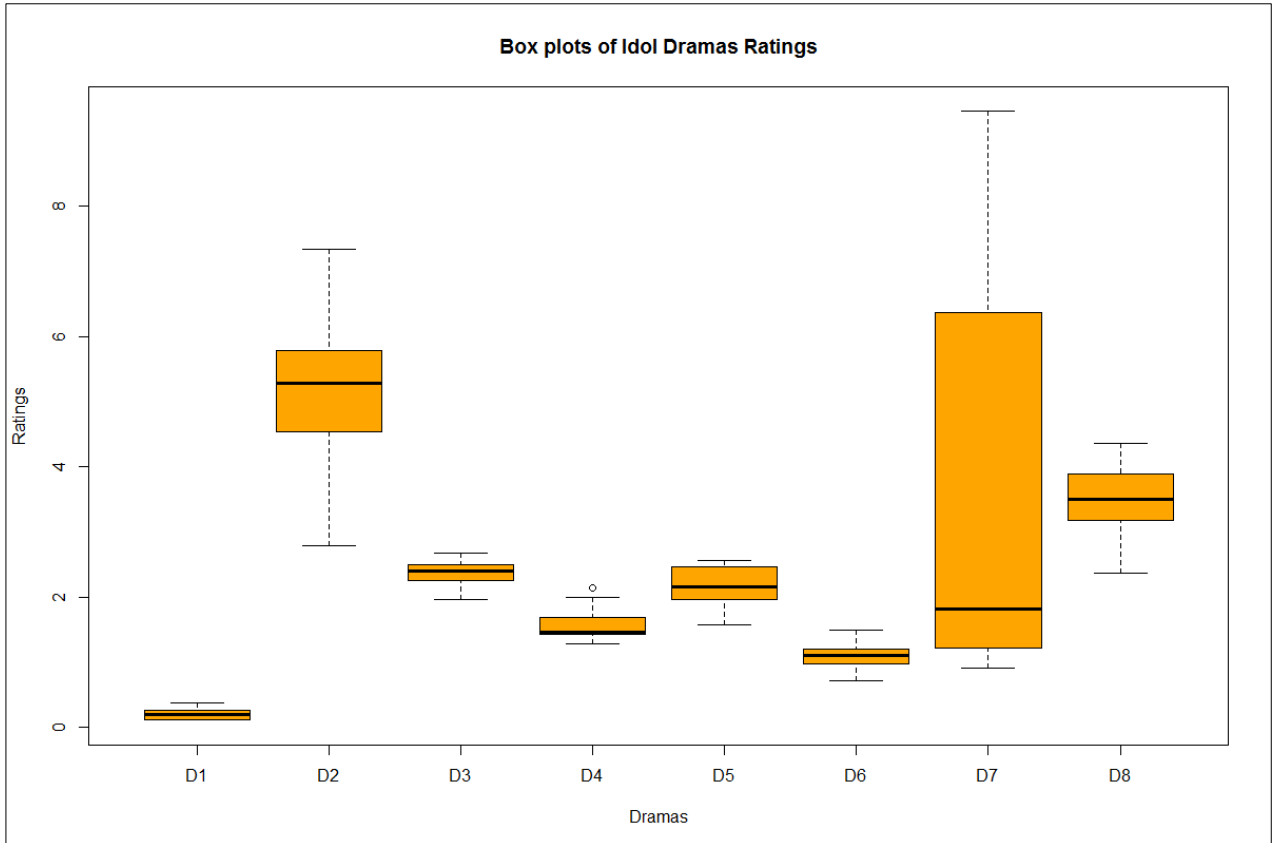
In our experiments, each drama is treated independently, i.e., ratings from other dramas are not considered and only historical ratings of the drama itself are used to predict its future ratings. For each drama, only 1 rating is predicted at one time, a.k.a., one-step forecast. Ratings are predicted from the 6th episode. For each rating to be predicted, say k^{th} episode, ratings ranging from the 1st

episode to the k-2th episode are for training, and the k-1th episode for performing validation to choose the parameter growth function. For example, for testing the 10th episode, the first 8 episodes are for training, and the 9th episode for validation. We call this experiment scenario "sequential one-step forecast". In this scenario, a model is trained for testing each episode.

4.2 Evaluation metric

We evaluate performance via 2 commonly used metrics in literature: mean absolute percentage error (MAPE) and mean absolute error (MAE). It is worth noting that by MAPE alone the result probably is somewhat misleading for programs of small ratings. For small ratings, it is easier to get bad MAPE, but probably by MAE the predictions are only a bit different from the actual.

In fact, In TV industry, programs of higher ratings are more valuable. Thus, we should focus on how accurate the predictions are in terms of the programs of higher ratings.

4.3 TWR settings and implementation

As for TWR, we have already described how it works in previous sections. Now we present its parameter settings and implementation as below:

- Settings: base model is regression tree with the following settings:
  - minsplit = 2
  - maxdepth = 30
  - After tree is grown, it is pruned based on validation error.
- Equation: Regression tree equations with bagging. Please refer to 附錄 Appendix for detail.
- Implementation: package rpart {rpart} in R [16]

In order to illustrate the usefulness of TWR, we try 6 different settings of TWR in experiments, which are summarized in Table 1. Specifically, TWR.A.EF includes 16 opinion polarity features based on Facebook fan page of dramas:

- Day of drama: number of negative/positive posts and comments (4 dimensions)
- Yesterday (4 dimensions)
- 2 days ago (4 dimensions)
- 3 days ago (4 dimensions)

4.4 Results

In this section, we show how well our solution and other competitive models predict ratings in terms of MAPE and MAE, via Table 3 and Table 4, respectively. Then we provide the interpretation of the results.

Surprisingly, PP, the most naïve and simplest model, already performs pretty well. Among all competitors, it has the lowest overall MAPE of 12.18%, which sets a very challenging baseline. PA, another naïve baseline, performs badly. It has much larger MAPE and MAE than all the other models. From results of two baselines, we can infer that ratings of last period is a good predictor, while ratings of older ratings are not, so what we need should be somewhere in between with more emphasis on recent ratings. In fact, Simple Exponential Smoothing (SES), one of our competitors from the 2nd category, is one way to capture this nature.

As for the models of the 2nd category, SES has the best overall performance in this category. Among all competitors, it has the 2nd lowest overall MAPE of 12.22% and the lowest overall MAE of 0.2893. Because SES is suitable for data with no trend or seasonal pattern, this result is as expected because from Figure 2 we already know that all dramas have no seasonal pattern, while only 2 out of 8 have trend pattern.

As for NNA, the only model of the 3rd category, its performance is neither very good nor very bad. However, it is worth noting that it has the lowest MAPE and MAE for D7 for some reason, probably the most difficult drama to be predicted well due to its widest range of ratings.

Now it comes to the results of our solution. First, let's compare the performance among 3 different growth functions: no growth (TWR.N), linear growth (TWR.L), and exponential growth (TWR.E). TWR.E has the best performance, followed by TWR.L and TWR.N. It shows that as more weights are put on the more recent training instances, the better performance we get. This evidence supports that our idea is valid. However, TWR has its limitation because TWR.E3 has mixed performance, i.e., performance of some dramas are improved, while some become worse. In fact, we observe that TWR.E3 is essentially same as PP. Thus, in order to automatically choose the best growth function, TWR.A is implemented. The results show that TWR.A outperforms all the other models in terms of overall MAPE (lowest 11.54%) and MAE (lowest 0.2883) among all dramas, which gives us more confidence that our idea is valid. Moreover, T.A.EF shows that our solution is extensible because it can be extended with external features.

Table 3. MAPE of TV ratings predictions

M↓D→	D1	D2	D3	D4	D5	D6	D7	D8	All
PP	24.27	8.53	8.59	13.95	12.65	12.63	13.07	8.98	12.18
PA	60.17	19.37	6.47	10.72	13.98	20.77	46.09	13.75	22.48
SES	32.47	8.21	6.33	11.94	12.35	12.57	13.07	8.90	12.22
DES	30.16	8.41	6.88	18.59	15.15	12.66	12.15	13.22	13.77
ETS	40.39	9.12	6.49	10.67	12.22	13.50	13.30	8.94	13.02

ARIMA	34.12	8.34	7.18	10.72	13.02	13.01	13.58	9.58	12.64
NNA	55.36	9.22	7.65	12.52	12.46	13.78	11.71	10.81	14.78
TWR.N	56.93	14.75	6.49	12.82	13.43	17.61	36.59	11.51	19.72
TWR.L	44.23	11.30	6.61	11.88	12.41	15.43	27.51	11.12	16.35
TWR.E	25.60	7.65	7.91	11.93	11.22	12.69	15.88	8.52	11.97
TWR.E3	24.28	8.39	8.42	13.58	12.55	12.63	13.34	8.84	12.09
TWR.A	25.47	7.86	7.59	10.81	12.11	11.67	13.44	8.97	**11.54**
T.A.EF	25.47	7.55	7.64	10.63	12.11	11.34	13.50	9.11	**11.47**

Table 4. MAE of TV ratings predictions

M↓D→	D1	D2	D3	D4	D5	D6	D7	D8	All
PP	0.0518	0.4775	0.1965	0.2250	0.2569	0.1286	0.5950	0.3272	0.3044
PA	0.0882	1.1048	0.1439	0.1681	0.2764	0.1900	2.3341	0.4646	0.6589
SES	0.0598	0.4604	0.1410	0.1909	0.2520	0.1280	0.5950	0.3202	0.2893
DES	0.0627	0.4644	0.1589	0.2880	0.3158	0.1370	0.5679	0.4904	0.3331
ETS	0.0686	0.5068	0.1447	0.1675	0.2504	0.1381	0.6249	0.3213	0.3000
ARIMA	0.0576	0.4573	0.1608	0.1681	0.2628	0.1318	0.6124	0.3412	0.2955
NNA	0.0955	0.5232	0.1731	0.1943	0.2442	0.1348	0.5031	0.3669	0.3002
TWR.N	0.0836	0.8287	0.1444	0.1973	0.2644	0.1606	1.6678	0.3880	0.5122
TWR.L	0.0695	0.6435	0.1480	0.1854	0.2458	0.1441	1.1960	0.3714	0.4099
TWR.E	0.0510	0.4335	0.1800	0.1915	0.2286	0.1283	0.6919	0.3035	0.2979
TWR.E3	0.0515	0.4699	0.1928	0.2190	0.2550	0.1288	0.6055	0.3208	0.3023
TWR.A	0.0497	0.4429	0.1712	0.1756	0.2440	0.1173	0.6094	0.3248	**0.2883**
T.A.EF	0.0497	0.4252	0.1723	0.1733	0.2440	0.1135	0.6119	0.3292	**0.2860**

4.5 Discussion

TWR focuses on weighted instances based on time, but what if we also apply the same concept to weighted features just like SES? It is worth trying to understand whether the effect will be double, or the effect is equivalent.

Another question is that the idea of weighted instances is so general that it should be possible to apply to all competitors. It would be interesting to re-design them with the concept of weighed instances and see how it affects performance.

Currently, we have not explored the needs of predicting dramas that have not been previously broadcast yet. To solve this more challenging problem, the idea of weighted instances is not enough. We may need to leverage data from other dramas to predict it well.

第五章　Conclusion

In this thesis, we present a novel solution called TWR to the problem of TV ratings prediction problem. Experiment results show that the auto-selected growth version of TWR outperforms all the other models in terms of overall MAPE and MAE among all dramas. Thus, we conclude that weighing instances with growth function before fitting regression model largely helps improve predictions of next ratings for weekly dramas, and with high probability our only assumption on which TWR is based is valid.

第六章　Future Work

When applying TWR to our data set, experiment results show that different dramas need different growth functions to result in good prediction accuracy, so it is worth extending the search space of growth functions and choosing it with a better way, e.g., model it with objective function and solve it as an optimization problem.

Besides, our data set for experiments is small and specific. We consider collecting more data and testing TWR with more types of TV programs such as daily dramas.

參考文獻　References

1. Death Of TV, http://www.businessinsider.com/category/death-of-tv

2. Danaher, P.J., Dagger, T.S., Smith, M.S.: Forecasting television ratings. International Journal of Forecasting 27(4), 1215–1240 (2011)

3. Danaher, P., Dagger, T.: Using a nested logit model to forecast television ratings. International Journal of Forecasting 28(3), 607–622 (2012)

4. Cheng, Y.H., Wu, C.M., Ku, T., Chen, G.D.: A predicting model of TV audience rating based on the Facebook. International Conference on Social Computing (SocialCom), pp. 1034–1037. IEEE (2013)

5. Hsieh, W.T., Chou, S.C.T., Cheng, Y.H., Wu, C.M.: Predicting TV audience rating with social media. Proceedings of the IJCNLP 2013 Workshop on Natural Language Processing for Social Media (SocialNLP), pp. 1–5. Asian Federation of Natural Language Processing, Nagoya (2013)

6. Yu-Yang Huang, Yu-An Yen, Ting-Wei Ku, Shou-De Lin, Wen-Tai Hsieh, Tsun Ku: A Weight-Sharing Gaussian Process Model Using Web-Based Information for Audience Rating Prediction. TAAI, LNAI 8916, pp. 198-208 (2014)

7. C.Meek, D.M. Chichering, D. Heckerman: Autoregressive Tree Models for Time-Series Analysis. Proceedings of the Second International SIAM Conference on Data Mining, pp. 229-244 (2002)

8. Leo Breiman: Bagging predictors. Machine Learning Volume 24, Issue 2, pp, 123-140 (1996)

9. H Drucker: Improving regressors using boosting techniques. ICML (1997)

10. Brown, Robert G.: Exponential Smoothing for Predicting Demand. Cambridge, Massachusetts: Arthur D. Little Inc. p. 15 (1956)

11. R Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. R version 3.1.2. (2014)

12. C. C. Holt: Forecasting trends and seasonals by exponentially weighted moving averages, ONR Research Memorandum, Carnegie Institute of Technology 52 (1957)

13. Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D: Forecasting with exponential smoothing: the state space approach, Springer-Verlag (2008)

14. Box, George, Jenkins, Gwilym: Time series analysis: Forecasting and control. San Francisco: Holden-Day (1970)

15. Hyndman, R.J. and Khandakar, Y.: Automatic time series forecasting: The forecast package for R. Journal of Statistical Software, 26(3). R package version 5.7. (2008)

16. Terry Therneau, Beth Atkinson and Brian Ripley: rpart: Recursive Partitioning and Regression Trees. R package version 4.1-8. (2014)

17. Terry M. Therneau, Elizabeth J. Atkinson, Mayo Foundation: An Introduction to Recursive Partitioning Using the RPART Routines. (2015)

附錄 Appendix

A: Equations of 30 state space models for ETS

ADDITIVE ERROR MODELS

Trend	Seasonal		
	N	**A**	**M**
N	$y_t = \ell_{t-1} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \alpha\varepsilon_t$	$y_t = \ell_{t-1} + s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \alpha\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = \ell_{t-1}s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \alpha\varepsilon_t/s_{t-m}$ $s_t = s_{t-m} + \gamma\varepsilon_t/\ell_{t-1}$
A	$y_t = \ell_{t-1} + b_{t-1} + \varepsilon_t$ $\ell_t = \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t$	$y_t = \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = (\ell_{t-1} + b_{t-1})s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t/s_{t-m}$ $b_t = b_{t-1} + \beta\varepsilon_t/s_{t-m}$ $s_t = s_{t-m} + \gamma\varepsilon_t/(\ell_{t-1} + b_{t-1})$
A$_\mathbf{d}$	$y_t = \ell_{t-1} + \phi b_{t-1} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t$ $b_t = \phi b_{t-1} + \beta\varepsilon_t$	$y_t = \ell_{t-1} + \phi b_{t-1} + s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t$ $b_t = \phi b_{t-1} + \beta\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = (\ell_{t-1} + \phi b_{t-1})s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t/s_{t-m}$ $b_t = \phi b_{t-1} + \beta\varepsilon_t/s_{t-m}$ $s_t = s_{t-m} + \gamma\varepsilon_t/(\ell_{t-1} + \phi b_{t-1})$
M	$y_t = \ell_{t-1}b_{t-1} + \varepsilon_t$ $\ell_t = \ell_{t-1}b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t/\ell_{t-1}$	$y_t = \ell_{t-1}b_{t-1} + s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1}b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t/\ell_{t-1}$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = \ell_{t-1}b_{t-1}s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1}b_{t-1} + \alpha\varepsilon_t/s_{t-m}$ $b_t = b_{t-1} + \beta\varepsilon_t/(s_{t-m}\ell_{t-1})$ $s_t = s_{t-m} + \gamma\varepsilon_t/(\ell_{t-1}b_{t-1})$
M$_\mathbf{d}$	$y_t = \ell_{t-1}b_{t-1}^{\phi} + \varepsilon_t$ $\ell_t = \ell_{t-1}b_{t-1}^{\phi} + \alpha\varepsilon_t$ $b_t = b_{t-1}^{\phi} + \beta\varepsilon_t/\ell_{t-1}$	$y_t = \ell_{t-1}b_{t-1}^{\phi} + s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1}b_{t-1}^{\phi} + \alpha\varepsilon_t$ $b_t = b_{t-1}^{\phi} + \beta\varepsilon_t/\ell_{t-1}$ $s_t = s_{t-m} + \gamma\varepsilon_t$	$y_t = \ell_{t-1}b_{t-1}^{\phi}s_{t-m} + \varepsilon_t$ $\ell_t = \ell_{t-1}b_{t-1}^{\phi} + \alpha\varepsilon_t/s_{t-m}$ $b_t = b_{t-1}^{\phi} + \beta\varepsilon_t/(s_{t-m}\ell_{t-1})$ $s_t = s_{t-m} + \gamma\varepsilon_t/(\ell_{t-1}b_{t-1}^{\phi})$

MULTIPLICATIVE ERROR MODELS

Trend	Seasonal		
	N	**A**	**M**
N	$y_t = \ell_{t-1}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}(1 + \alpha\varepsilon_t)$	$y_t = (\ell_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1} + \alpha(\ell_{t-1} + s_{t-m})\varepsilon_t$ $s_t = s_{t-m} + \gamma(\ell_{t-1} + s_{t-m})\varepsilon_t$	$y_t = \ell_{t-1}s_{t-m}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}(1 + \alpha\varepsilon_t)$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$
A	$y_t = (\ell_{t-1} + b_{t-1})(1 + \varepsilon_t)$ $\ell_t = (\ell_{t-1} + b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1} + \beta(\ell_{t-1} + b_{t-1})\varepsilon_t$	$y_t = (\ell_{t-1} + b_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1} + b_{t-1} + \alpha(\ell_{t-1} + b_{t-1} + s_{t-m})\varepsilon_t$ $b_t = b_{t-1} + \beta(\ell_{t-1} + b_{t-1} + s_{t-m})\varepsilon_t$ $s_t = s_{t-m} + \gamma(\ell_{t-1} + b_{t-1} + s_{t-m})\varepsilon_t$	$y_t = (\ell_{t-1} + b_{t-1})s_{t-m}(1 + \varepsilon_t)$ $\ell_t = (\ell_{t-1} + b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1} + \beta(\ell_{t-1} + b_{t-1})\varepsilon_t$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$
A$_\mathbf{d}$	$y_t = (\ell_{t-1} + \phi b_{t-1})(1 + \varepsilon_t)$ $\ell_t = (\ell_{t-1} + \phi b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = \phi b_{t-1} + \beta(\ell_{t-1} + \phi b_{t-1})\varepsilon_t$	$y_t = (\ell_{t-1} + \phi b_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha(\ell_{t-1} + \phi b_{t-1} + s_{t-m})\varepsilon_t$ $b_t = \phi b_{t-1} + \beta(\ell_{t-1} + \phi b_{t-1} + s_{t-m})\varepsilon_t$ $s_t = s_{t-m} + \gamma(\ell_{t-1} + \phi b_{t-1} + s_{t-m})\varepsilon_t$	$y_t = (\ell_{t-1} + \phi b_{t-1})s_{t-m}(1 + \varepsilon_t)$ $\ell_t = (\ell_{t-1} + \phi b_{t-1})(1 + \alpha\varepsilon_t)$ $b_t = \phi b_{t-1} + \beta(\ell_{t-1} + \phi b_{t-1})\varepsilon_t$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$
M	$y_t = \ell_{t-1}b_{t-1}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}b_{t-1}(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1}(1 + \beta\varepsilon_t)$	$y_t = (\ell_{t-1}b_{t-1} + s_{t-m})(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}b_{t-1} + \alpha(\ell_{t-1}b_{t-1} + s_{t-m})\varepsilon_t$ $b_t = b_{t-1} + \beta(\ell_{t-1}b_{t-1} + s_{t-m})\varepsilon_t/\ell_{t-1}$ $s_t = s_{t-m} + \gamma(\ell_{t-1}b_{t-1} + s_{t-m})\varepsilon_t$	$y_t = \ell_{t-1}b_{t-1}s_{t-m}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}b_{t-1}(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1}(1 + \beta\varepsilon_t)$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$
M$_\mathbf{d}$	$y_t = \ell_{t-1}b_{t-1}^{\phi}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}b_{t-1}^{\phi}(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1}^{\phi}(1 + \beta\varepsilon_t)$	$y_t = (\ell_{t-1}b_{t-1}^{\phi} + s_{t-m})(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}b_{t-1}^{\phi} + \alpha(\ell_{t-1}b_{t-1}^{\phi} + s_{t-m})\varepsilon_t$ $b_t = b_{t-1}^{\phi} + \beta(\ell_{t-1}b_{t-1}^{\phi} + s_{t-m})\varepsilon_t/\ell_{t-1}$ $s_t = s_{t-m} + \gamma(\ell_{t-1}b_{t-1}^{\phi} + s_{t-m})\varepsilon_t$	$y_t = \ell_{t-1}b_{t-1}^{\phi}s_{t-m}(1 + \varepsilon_t)$ $\ell_t = \ell_{t-1}b_{t-1}^{\phi}(1 + \alpha\varepsilon_t)$ $b_t = b_{t-1}^{\phi}(1 + \beta\varepsilon_t)$ $s_t = s_{t-m}(1 + \gamma\varepsilon_t)$

B: Equations of neural network auto-regression

Assume that a fully-connected feed-forward neural network has 3 layers:

- Input layer with n neurons for inputs plus 1 bias neuron
- Hidden layer with k neurons for inputs from input layer plus 1 bias neuron
- Output layer with 1 linear output neuron

Equation of hidden layer output: $z_j = b_j + \sum_{i=1}^{n}(w_{i,j}x_i)$

Equation of output layer output: $z = b + \sum_{i=1}^{k}(w_i z_i)$

Where z is the outputs of hidden neurons and output neuron, b is parameters of bias neurons, w is parameters of connections, and x is the inputs from the input layer.

The tuning of auto-regression is just using lagged values of time series as inputs for input layer.

C: Equations of TWR

Assume that the base model of TWR is regression tree [17]:

- Splitting criterion: $SS_T - (SS_L + SS_R)$, where $SS_T = \sum (y_i - \bar{y})^2$ is the sum of squares for the node, and SS_L, SS_R are the sums of squares for the left and right son, respectively. That is, it chooses the split to maximize the between-groups sum-of-squares in a simple analysis of variance.
- Prediction: $\bar{y}$, i.e., the mean of the node.