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

側邊

國立臺灣大學電機資訊學院資訊工程學系

國立臺灣大學

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

碩士論文 以基於時間比重之回歸預測電視收視率 顧廷緯 撰 **1041**（年月）

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

[口試委員會審定書 3](#_Toc409722014)

[誌謝 4](#_Toc409722015)

[中文摘要 5](#_Toc409722016)

[英文摘要 5](#_Toc409722017)

[第一章 Introduction 8](#_Toc409722021)

[1.1 Contribution 8](#_Toc409722022)

[1.2 Background and importance of problem 8](#_Toc409722023)

[1.3 Overview of problem and solution 8](#_Toc409722024)

[1.4 Motivation of solution 9](#_Toc409722025)

[第二章 Related Work 9](#_Toc409722026)

[第三章 Method 10](#_Toc409722027)

[3.1 Pseudo-code of TWR 10](#_Toc409722028)

[3.2 Fitting step 1: Windowing transformation 11](#_Toc409722029)

[3.3 Fitting step 2: Weighing training instances 11](#_Toc409722030)

[3.4 Fitting step 3: Building a base model 12](#_Toc409722031)

[3.5 Predicting stage of TWR 12](#_Toc409722032)

[第四章 Experiments 13](#_Toc409722033)

[4.1 Data set 13](#_Toc409722034)

[4.2 Evaluation metric 15](#_Toc409722035)

[4.3 Models 15](#_Toc409722036)

[4.4 Results 16](#_Toc409722037)

[第五章 Conclusion 18](#_Toc409722038)

[第六章 Future Work 18](#_Toc409722039)

[參考文獻 18](#_Toc409722040)

[附錄 19](#_Toc409722041)

# 圖目錄

[Figure 1. Time series plot for ratings of dramas 13](#_Toc409634079)

[Figure 2. Box plots for ratings of dramas 13](#_Toc409634080)

# 表目錄

[Table 1. Basic information about dramas 12](#_Toc409634137)

[Table 2. List of models 15](#_Toc409634138)

[Table 3. MAPE of TV ratings predictions 16](#_Toc409634139)

[Table 4. MAE of TV ratings predictions 16](#_Toc409634140)

# 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 Work

In [2, 3], eight different models and one novel logit model for predicting TV ratings are studied, but we do not consider them in our study due to different characteristic of data set in terms of type of programs. Concretely, in their data set, many TV programs are broadcast only once, which is largely different from our problem of predicting ratings for weekly dramas.

In [4, 5, 6], their data sets are similar to ours in terms of type of programs, but we do not rely on any web or social info as features as they did, which makes our solution more general and easily applied to new dramas.

In [7], its data set only consists of one TV program with ten weekly ratings, which is too small to be compared with ours.

# 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 [8]), (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.

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 **n** = 5, **w** = 3, **f** = ex, where ex is the exponential function.)

Fitting process:

1. Windowing transformation with window size **w**:  
   **x** = { x1, x2,..., x5 } 🡪 **X** = { **w1**= (x1, x2, x3=y3), (x2, x3, x4=y4), (x3, x4, x5=y5) }  
   We get 3 training instances, with each having 2 input features and 1 label.
2. Weighing training instances with growth function **f**: **W** = { e, e2, e3 }
3. Building a base model with weighted instances: **m** = **Learner.Fit**(**X**, **W**, **p**)

Output: base model **m**

Predicting process: Input data **w4** = (x4, x5, x6=y6), where x6=y6 is the unknown label to predict.

Output: one-step forecast **x­6’** = **Learner.Predict**(**m**, **w4**)

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 w1 < w2 < w3. 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** (ex ), and **cubic exponential** (e3x) 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, e3x and e4x 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 [9] 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 [10], 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 need 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 1 and Figure 1 – 2. From the time series plot (Figure 1), 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 2), 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 1. 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 1. 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 1. Time series plot for ratings of dramas

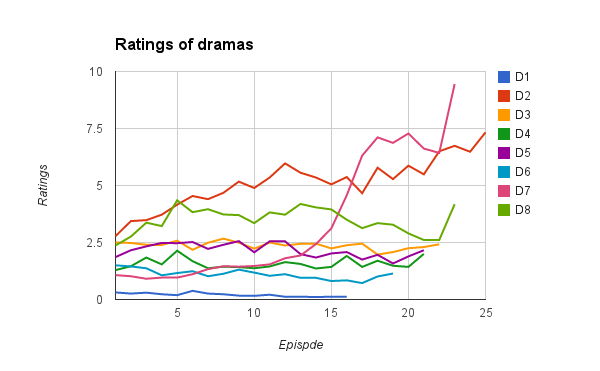
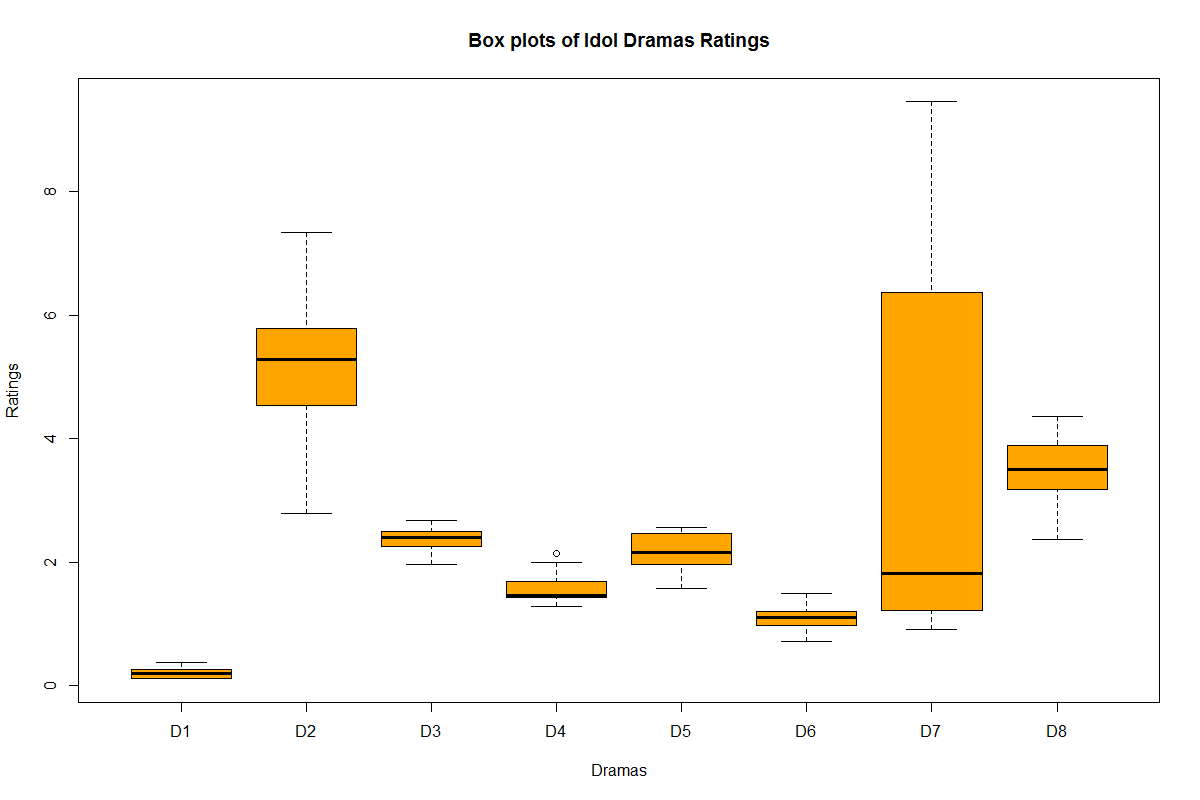


Figure 2. Box plots for ratings of dramas



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 kth 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 Models

We compare our solution with 7 competitors 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 2. In Table 2, the 4th category is our solution with different growth functions.

We choose language R 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 example, as we said in the previous section, we choose regression tree as the base model for our solution, so package rpart, a well-known R implementation of decision tree that supports regression, is used. For models that don’t have published packages, they are implemented by ourselves.

Table 2. List of models

|  |  |  |  |
| --- | --- | --- | --- |
| # | Category | Name | Summary |
| 1 | 1 | Last period (LP) | Guess value of the last period |
| 2 | 1 | Past average (PA) | Guess average of all the previous |
| 3 | 2 | Simple Exponential Smoothing (SES) | Package: HoltWinters {stats} |
| 4 | 2 | Double Exponential Smoothing (DES) | Package: HoltWinters {stats} |
| 5 | 2 | Exponential Smoothing State Space (ETS) | Package: ets {forecast} |
| 6 | 2 | ARIMA | Package: auto.arima {forecast} |
| 7 | 3 | Neural network auto-regression (NNA) | Package: nnetar {forecast} |
| 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) = ex |
| 11 | 4 | TWR with e3x growth (TWR.E3) | g(x) = e3x |
| 12 | 4 | TWR with auto-selected growth (TWR.A) | Pick growth with min validation err. |

4.4 Results

In this section, we show how well the models predict ratings in terms of MAPE and MAE.

Surprisingly, the most naïve and simplest model, LP, already performs pretty well. Among all competitors, it has the lowest overall MAPE of 0.1218, which sets a very challenging baseline. On the other hand, another naïve baseline, PA, perform pretty bad. It has much larger MAPE and MAE than all the other models. From results of two baselines, we can infer that value of next ratings has much to do with recent ratings, while has little or nothing to do with older ratings. This observation is very similar to the idea of Simple Exponential Smoothing: forecast is the weighted average of past observations, and the weights decay exponentially as observations get older.

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 0.1222 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 1 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, 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. 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 and MAE among all dramas, which gives us more confidence that our idea is valid.

Table 3. MAPE of TV ratings predictions

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M↓D→ | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | All |
| LP | 0.2427 | 0.0853 | 0.0859 | 0.1395 | 0.1265 | 0.1263 | 0.1307 | 0.0898 | 0.1218 |
| PA | 0.6017 | 0.1937 | 0.0647 | 0.1072 | 0.1398 | 0.2077 | 0.4609 | 0.1375 | 0.2248 |
| SES | 0.3247 | 0.0821 | 0.0633 | 0.1194 | 0.1235 | 0.1257 | 0.1307 | 0.0890 | 0.1222 |
| DES | 0.3016 | 0.0841 | 0.0688 | 0.1859 | 0.1515 | 0.1266 | 0.1215 | 0.1322 | 0.1377 |
| ETS | 0.4039 | 0.0912 | 0.0649 | 0.1067 | 0.1222 | 0.1350 | 0.1330 | 0.0894 | 0.1302 |
| ARIMA | 0.3412 | 0.0834 | 0.0718 | 0.1072 | 0.1302 | 0.1301 | 0.1358 | 0.0958 | 0.1264 |
| NNA | 0.5536 | 0.0922 | 0.0765 | 0.1252 | 0.1246 | 0.1378 | 0.1171 | 0.1081 | 0.1478 |
| TWR.N | 0.5693 | 0.1475 | 0.0649 | 0.1282 | 0.1343 | 0.1761 | 0.3659 | 0.1151 | 0.1972 |
| TWR.L | 0.4423 | 0.1130 | 0.0661 | 0.1188 | 0.1241 | 0.1543 | 0.2751 | 0.1112 | 0.1635 |
| TWR.E | 0.2560 | 0.0765 | 0.0791 | 0.1193 | 0.1122 | 0.1269 | 0.1588 | 0.0852 | 0.1197 |
| TWR.E3 | 0.2428 | 0.0839 | 0.0842 | 0.1358 | 0.1255 | 0.1263 | 0.1334 | 0.0884 | 0.1209 |
| TWR.A | 0.2547 | 0.0786 | 0.0759 | 0.1081 | 0.1211 | 0.1167 | 0.1344 | 0.0897 | **0.1154** |

Table 4. MAE of TV ratings predictions

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M↓D→ | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | All |
| LP | 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** |

# 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.

# 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

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. Yilei, Zheng: Audience Rating Prediction of New TV Programs Based on GM (1.1) Envelopment Model. IEEE International Conference on Grey Systems and Intelligent Services (2009)
8. 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)
9. Leo Breiman: Bagging predictors. Machine Learning Volume 24, Issue 2, pp, 123-140 (1996)
10. H Drucker: Improving regressors using boosting techniques. ICML (1997)

# 附錄

N/A