A Weight-Sharing Gaussian Process Model Using Web-Based Information for Audience Rating Prediction

Yu-Yang Huang[1], Yu-An Yen[1], Ting-Wei Ku[1], Shou-De Lin[1], Wen-Tai Hsieh[2], and Tsun Ku[2]

[1] Dept. of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan
[2] Institute for Information Industry, Taipei, Taiwan
{myutwo150,lovelove6402,martin79831}@gmail.com,
sdlin@csie.ntu.edu.tw,
{wentai,cujing}@iii.org.tw

Abstract. In this paper, we describe a novel Gaussian process model for TV audience rating prediction. A weight-sharing covariance function well-suited for this problem is introduced. We extract several types of features from Google Trends and Facebook, and demonstrate that they can be useful in predicting the TV audience ratings. Experiments on a dataset consisting of daily dramas show that the proposed model outperforms the other conventional models given the same feature set.

Keywords: Time Series, Gaussian Process, Audience Rating Prediction.

1 Introduction

Time series analysis is an active research area with many real-world applications including price forecasting [12] and sales prediction [9]. A typical method relies on historical data sequences to predict upcoming data points. In this paper, we focus on television audience rating prediction. The goal is to accurately predict the rating of an upcoming TV episode, and to analyze the most crucial factors that cause the audience ratings to fluctuate.

Gaussian Process Regression (GPR) [16] is used to predict the audience ratings. We analyze variants of GPR models, and propose a weight-sharing kernel to deal with the overfitting issue caused by the increasing number of hyperparameters. The experiments show that, with the weight-sharing technique applied, our GPR model outperforms the other competitors in predicting the audience ratings.

Furthermore, we propose three novel types of features to boost the prediction performance. Different from previous works, our model relies not only on classic time series features such as historical ratings, but also on features extracted from social networks and search engines. For example, trend information from Google Trends, opinion polarity and popularity information from Facebook are used in the prediction. We show how such information can be adapted in the proposed GPR model.

The main contributions of this work are summarized as follows:

- We modify a standard kernel of GPR model to avoid overfitting, and make it more suitable for the TV audience rating prediction problem.
- We propose three novel types of web-based features: trend features, social network features, and opinion features for better performance.
- We conduct experiments to verify the validity of the model and features.

2 Related Work

Audience rating prediction is treated as a time series forecasting problem in the field of statistics and data analysis. Well-known models such as autoregressive model, moving average model, or the hybrid (ARIMA), or the more advanced ones such as generalized autoregressive conditional heteroskedasticity (GARCH) [2] and the non-linear extension of it (NGARCH) [12] are all plausible models that can be applied. However, those models may not be the best choice for audience rating prediction because they do not consider specific characteristic of the ratings. Researchers have shown that using time-based and program-based covariates provides a more effective way to forecast the audience ratings [5, 6] than general time series models, as these models consider correlations between rating, genre, show duration, live status, etc.

Another serious drawback of general time series models is that they cannot consider external features such as information from social media and search engine. Such external and social information has been shown effective in forecasting. In [1], a work using chatters from Twitter to predict future revenue of movies is proposed. The works in [4, 8] propose to perform audience rating prediction utilizing the count of posts and comments from social media. Our work further extends the idea to exploit opinion mining and search engine such as Google Trends to enhance the prediction performance.

3 Framework and Features

In this paper, the TV audience rating prediction problem is modeled as a supervised learning task. To forecast near-future ratings, historical data together with the following listed features are used as the training input for the models.

1. Basic Time Series Features

Similar to other basic time series forecasting models, the ratings of the previous episodes, the rating of the first episode, and binary indicator variables corresponding to weekdays are used as features.

2. Social Network Features

Nowadays, TV companies often host “Fan Pages” on social networking sites such as Facebook. On these pages, companies run promotional campaigns, face-to-face events, polls, and provide previews of the next episodes. Also, it provides a platform for the fans to interact with each other and show their support or oppose to the show. To model such effects, the daily cumulative numbers of "posts", "shares", "likes", "comments" on the official Facebook Fan Pages of the dramas are included as features.

3. Opinion Polarity Features

Users may express their thoughts toward a show via a Facebook Fan Page, and such opinions will have influence on others' opinions. For example, if most of the fans are looking forward to the upcoming episode, it will be reasonable to assign a higher audience rating for the new episode. Thus we propose to analyze the polarity (i.e. positive or negative) of users' posts and replies on Facebook fans page. We use the daily cumulative number of positive and negative words in both posts and comments as the opinion polarity features.

4. Trend Features

Google Trends is a useful tool provided by Google Inc. to investigate the popularity of a keyword in a region. Given certain time period, it gives the number of searches for a keyword relative to the total number of searches across this period. The displayed number is normalized such that the highest number is equal to 100 and the lowest number is equal to zero. For each drama, we collect time series data from Google Trends for three different sets of keywords. We use drama name as well as actor/actress's name as queries in Google Trends to obtain the corresponding features.

4 Methodology

4.1 Gaussian Process Regression (GPR)

A typical regression problem can be formulated as

$$y = f(\mathbf{x}) + \varepsilon \tag{1}$$

where $\mathbf{x}$ is the input vector, y is the observed target value, and f is a function that models the underlying process of generating the data points $\{(\mathbf{x}_i, y_i): i = 1, \dots, N\}$. An additive independent and identically distributed Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$ is assumed.

There are two equivalent ways to derive the predictive distribution for Gaussian process regression, namely the weight-space view and the function-space view [13]. In the following paragraphs, we will give a brief introduction to the main concepts of GPR in the function-space.

A random process $f = \{f(\mathbf{x}): \mathbf{x} \in \mathcal{X}\}$ is defined as a collection of random variables $f(\mathbf{x})$ indexed by an ordered set $\mathcal{X}$. In the audience rating prediction problem, we consider the input space $\mathcal{X} \subseteq \Re^d$, where d is the dimension of input vectors. The random variable $f(\mathbf{x})$ therefore represent the value of the random function f evaluated at the data point $\mathbf{x}$. If normality is assumed, the random process is called a *Gaussian process* (GP). An important property of GP is that any finite collection of the random variables $f(\mathbf{x})$ will be jointly normally distributed.

A Gaussian process is completely defined by its second-order statistics. The mean function and the covariance function of a Gaussian process can be defined as follows:

$$m(\mathbf{x}) = \mathrm{E}[f(\mathbf{x})] \tag{2}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathrm{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{3}$$

and $f \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. Without loss of generality, it is common to consider GPs with mean function $m(\mathbf{x}) \equiv 0$ to simplify the derivation. In this case, a GP is fully specified given its covariance function.

Since it is infeasible to consider all possible random functions, certain assumptions must be made when making inference. By restricting the underlying function f to be distributed as a GP, the number of choices is reduced. Furthermore, a Gaussian predictive distribution can be derived in closed-form under such assumption. If we consider only zero-mean Gaussian processes, then for a test input $\mathbf{x}^*$, the mean and variance of the predictive distribution can be computed as follows [16]:

$$\mu(\mathbf{x}^*) = \mathbf{k}^T K^{-1} \mathbf{y} \tag{4}$$

$$\sigma^2(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}^T K^{-1} \mathbf{k} \tag{5}$$

where $\mathbf{k} = \left(k(\mathbf{x}^*, \mathbf{x}_1), \dots, k(\mathbf{x}^*, \mathbf{x}_N)\right)^T$, $K = \left[K_{ij}\right]$ is the covariance matrix of training input vectors with $K_{ij} = k\left(\mathbf{x}_i, \mathbf{x}_j\right)$, and $\mathbf{y} = (y_1, y_2, \dots, y_N)^T$ is a vector of training target values. The point prediction is commonly taken to be the mean of the predictive distribution, i.e. $y^* = \mu(\mathbf{x}^*)$.

4.2 Weight-Sharing Kernel

The covariance function is also called the *kernel* of a GP. As previously mentioned, the behavior of a zero-mean GP can be fully specified provided its covariance function. For regression problems, we want to assign similar prediction values to two input vectors that are close in space. In other words, if two similar time series are observed, the model should be able to give similar predictions. A widely used kernel possessing this property is the radial basis function (RBF) kernel, which is called a squared exponential (SE) kernel in GP literature. It has the form

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\sum_{i=1}^{d} \frac{(x_i - x'_i)^2}{{l_i}^2}\right) \tag{6}$$

where x_i is the i-th dimension of vector $\mathbf{x}$ and d is the dimension of input vectors.

There are d hyperparameters $\boldsymbol{\theta} = (l_1, l_2, \dots l_d)^T$ for this kernel. The hyperparameters are called the *characteristic length-scales*. It serves as a distance measure along the i-th dimension. The effect of these hyperparameters can be shown more clearly if we rewrite Eq. 6 as

$$k(\mathbf{x}, \mathbf{x}') = \prod_{i=1}^{d} \exp\left(-\frac{(x_i - x'_i)^2}{2{l_i}^2}\right). \tag{7}$$

If the characteristic length-scale for the i-th dimension is large, the i-th exponential term will be close to zero, and the covariance will be independent of that input dimension. This is a form of *automatic relevance determination* (ARD) [11] or "soft" feature selection. When we are estimating the hyperparameters, the irrelevant input dimensions will be ignored by fitting the length-scales to a relatively large value. However, this type of kernel introduces one parameter per input dimension. The common problem of overfitting is severe if we are dealing with high-dimensional inputs [3]. This is especially the case for time series prediction. If we have M types of co-varying features, and for each of them we consider only T time steps before current prediction, the total number of features is $d = M \times T$, which will increase rapidly as we consider longer historical sequences. This will limit the power of the time series model in that it must either include less features or use shorter historical sequences. Furthermore, the time needed to train an ARD kernel is significantly longer than its isotropic counterpart (i.e. setting $l_1 = l_2 = \cdots = l_n = l$). The isotropic SE kernel, although usually performs well, suffers from its inability to distinguish the importance of different input dimensions. Therefore, we propose a *weight-sharing kernel* that strikes a balance between the two.

In the field of time series prediction, the input features usually come from co-varying time sequences and therefore are naturally grouped. For example, the features extracted from Google Trends can be viewed as a feature group. The main idea is to reduce the number of hyperparameters by sharing the same length-scale among features belonging to the same group, while at the same time possessing the ability to determine the importance of different groups of features.

The kernel consists of a weighted sum of SE kernels:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') = &\sum_{g \in G_T} v_g \exp\left(-\frac{1}{2{l_g}^2} \sum_{i=1}^{d_g} (x_i - x'_i)^2\right) \\ &+ \sum_{g \in G_{T'}} v_g \exp\left(-\frac{1}{2} \sum_{i=1}^{d_g} \frac{(x_i - x'_i)^2}{{l_{i,g}}^2}\right). \end{aligned} \tag{8}$$

The first term is a weighted sum of isotropic SE kernels which are designed for time co-varying features. We denote the set of time co-varying feature groups (i.e. "*Opinion*", "*Google Trends*", "*Facebook*") as G_T. For each feature group g, the number of features in the group is denoted as d_g, and the overall importance of the group is v_g. The same length scale l_g is shared among all features belonging to the group. This can significantly reduce the number of hyperparameters.

The second term of the kernel is a weighted sum of ARD SE kernels. We use $G_{T'}$ to denote the set of time-invariant feature groups, or features that require a separate length-scale for each dimension to function properly (e.g. "*First Episode rating*", "*Past 3 Episodes ratings*" and "*Weekdays*"). Usually, the number of time-invariant features is much less than the time co-varying features, so this term would not add too many hyperparameters to the model.

As a brief example, assume that we consider M co-varying time sequences, each of them is of length T. In our case $T = 4$ and there are $M = 11$ time co-varying features (4 from "*Opinion*", 3 from "*Google Trends*", and 4 from "*Facebook*"). An ARD kernel will have 44 hyperparameters to be learnt, making the inference slow and the prediction inaccurate. On the other hand, the weight-sharing kernel introduces 2 hyperparameters (i.e. l_g and v_g) for each feature group, merely 6 in total. With the weight-sharing kernel applied, the inference is much faster and, as will be shown in the Experiment section, a better performance is achieved.

4.3 Training

In general, the hyperparameters of a Gaussian process model can be learnt by maximizing the marginal likelihood or by using Markov chain Monte Carlo methods such as slice sampling [10]. We adopt in this work the maximum marginal likelihood framework, also known as Type-II maximum likelihood (ML-II) or empirical Bayes.

In the ML-II framework, the hyperparameters are chosen by maximizing the probability of observing target values $\mathbf{y}$ given the input $X = \{\mathbf{x}_i : i = 1, \dots, N\}$. Let $\mathbf{f} = (f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_N))^T$ be a vector of function values evaluated at the N training input data points. Since the function $f \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ is a random function sampled from a GP, the vector $\mathbf{f}$ is an N-dimensional normally distributed random vector, i.e. $\mathbf{f}|X \sim \mathcal{N}(0, K)$. The exact form of the marginal likelihood is given by marginalizing over the random vector $\mathbf{f}$:

$$p(\mathbf{y}|X, \boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f}, X) p(\mathbf{f}|X, \boldsymbol{\theta}) \, d\mathbf{f} \tag{9}$$

where $\boldsymbol{\theta}$ is a vector of hyperparameters of the kernel. From Eq. 1 it is clear that $\mathbf{y} \sim \mathcal{N}(0, K_y)$, where $K_y = K + \sigma_n^2 I$. It follows that the log marginal likelihood is

$$\log p(\mathbf{y}|X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T K_y{}^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi \, . \tag{10}$$

To find the best hyperparameters with ML-II, we must take the derivatives of the log marginal likelihood with respect to the hyperparameters, as shown below:

$$\frac{\partial}{\partial \theta_j}\log p(\mathbf{y}|X, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^T K_y{}^{-1}\frac{\partial K_y}{\partial \theta_j} K_y{}^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}\left(K_y{}^{-1}\frac{\partial K_y}{\partial \theta_j}\right). \tag{11}$$

The above computations involve matrix inversion, which takes $O(N^3)$ time complexity. It often limits the use of GP models to small data sets, or approximation methods must be sought [14]. However, in our problem, the length of each drama usually does not exceed a few tens of episodes. Since time complexity is not a big issue for this application, we use exact inference in all the following experiments.

To avoid being trapped to local minima, we randomly initialize the hyperparameters and run the optimization multiple times. The set of hyperparameters yielding the highest marginal likelihood is selected as the final model to perform prediction.

5 Experiments

In this section we first introduce our dataset and the evaluation metric. Then we compare our model with other competitors. Finally we conduct a quantitative analysis on the usefulness of selected features.

5.1 Dataset and Evaluation Metrics

Four *daily dramas*, which are broadcast only on weekdays, are chosen as the experiment dataset. We collect Facebook Fan Page statistics, opinion polarities, and Google Trends information to create the features. All features are standardized. A brief summary for the dataset is listed in Table 1.

Table 1. Basic information about the dramas.

Drama	#Episode	Broadcast period	Average Rating	Std.
D1	80	2012/04/10 ~ 2012/07/30	1.819	0.263
D2	82	2012/07/31 ~ 2012/11/26	1.941	0.201
D3	90	2012/06/27 ~ 2012/10/30	1.485	0.226
D4	84	2011/12/13 ~ 2012/04/09	2.540	0.754

We perform sequential prediction for all experiments. That is, to predict the rating of episode *k* of drama D1, we first train our model using data from the first *k-1* episodes of D1 and data from the other dramas. For Facebook Fan Page statistics, opinion polarities, and Google Trends features, we use the values from the broadcasting day and three days prior to it. To evaluate the usefulness of different feature combinations, we also train models on all possible combinations of features.

The mean absolute percentage error (MAPE) is used as the evaluation metric since it is the most commonly used metric for the audience rating prediction problems.

$$MAPE = \frac{100\%}{N} \sum_{i=1}^{N} \left| \frac{y_i - p_i}{y_i} \right| \tag{12}$$

5.2 Comparison of the Models

We compare the proposed model with three other models as described below.

1. Support Vector Regression (SVR) [15]. This model solves the following regression problem:

$$\min_{w} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \frac{1}{y_i} \xi_\epsilon (w; x_i, y_i) \tag{13}$$

Note that Eq. 13 is different from the original form as there is an additional $1/y_i$ term added to optimize the MAPE. We tried both linear and polynomial transformation, and use LIBLINEAR [7] for the experiment. After several trials, we choose the regularization parameter C=1.

2. GP_ard. GPR with ARD SE kernel. (Eq. 6)
3. GP_iso. GPR with isotropic SE kernel. (Eq. 6, with $l_1 = l_2 = \cdots = l_n = l$)

There are six types of features, namely social network features (from Facebook), opinion features, trend features, ratings from previous three episodes, rating of the first episode, and weekday indicator variables. Therefore, there are total $2^6 - 1 = 63$ different combinations of features. Table 2 shows the average MAPE of all feature combinations for each drama. We also rank the MAPE obtained from different models, and then compute the average value. The result is shown in Table 3. Our model outperforms baseline models in terms of both MAPE and ranking.

Table 2. Average MAPE of all possible feature combinations.

Drama / Model	D1	D2	D3	D4	Avg.
SVR	0.1132	0.1027	0.1300	0.1396	0.1214
GP_ard	0.1124	0.0928	0.1297	0.1162	0.1128
GP_iso	0.1165	0.0959	0.1357	0.1158	0.1160
Our Model	0.1163	0.0918	0.1276	0.1117	**0.1118**

Table 3. Average ranking of all feature combinations.

Drama / Model	D1	D2	D3	D4	Avg.
SVR	2.1904	3.3492	2.1587	3.0952	2.6984
GP_ard	2.4365	2.2619	2.9444	2.6349	2.5694
GP_iso	2.7540	2.4683	2.5635	2.4365	2.5556
Our Model	2.6190	1.9206	2.3333	1.8333	**2.1766**

Then, we compare our modified GP model with the two standard GP-based competitors, GP_ard and GP_iso. Since the best feature combination is fairly different for each drama, a general scenario is considered, where all available features are used in the prediction. With the capability to estimate the relative importance of different groups of features and to avoid overfitting, the proposed weight-sharing method outperforms the other standard GP-based models. Results are shown in Table 4.

Table 4. Average MAPE using all features.

Drama / Model	D1	D2	D3	D4	Avg.
GP_ard	0.1015	0.0887	0.1001	0.1184	0.1022
GP_iso	0.1018	0.0833	0.0929	0.0973	0.0938
Our Model	0.0991	0.0794	0.0911	0.0869	**0.0891**

5.3 Feature Analysis

In this section, we study the usefulness of the features based on our proposed model. As previously mentioned, the features are categorized into six types, as shown in the first column of Table 5. Holding the rest of the features identical, we compare the performance with and without a certain type of features. If the resulting error is lower when a certain type of features is used, we define it as a "win". Conversely, if the error is higher, then we define it as a "lose". For instance, with all other conditions the same, if removing Facebook features results in a higher MAPE for D1, then a "lose" is assigned. Since there are total 63 different combinations of features, 31 comparisons are made for each drama. The winning percentages $\left(=\frac{W}{W+L}\times 100\%\right)$ for each type of the features are shown in Table 5. The higher the winning percentage, the more useful it is. We can observe that the previous ratings and weekday information are overall the most important features, while most of the features except opinion feature generally improves the performance.

Table 5. Winning percentage with or without a certain type of features.

Winning Percentage (%)	D1	D2	D3	D4	TOTAL
Opinion	**26**	**58**	**42**	**55**	**45**
Google Trends	**58**	**35**	**74**	**77**	**61**
Facebook	**29**	**71**	**87**	**55**	**60**
Ratings of previous three episodes	**100**	**100**	**100**	**100**	**100**
Rating of the first episode	**71**	**61**	**97**	**42**	**68**
Weekday	**81**	**100**	**100**	**84**	**91**

6 Conclusion

In this paper, we present a weight-sharing Gaussian process model for the TV audience rating prediction problem. Also, we extract three types of web-based features for this task, namely Facebook Fan Page statistics, opinion polarities, and Google Trends. A series of experiments on a dataset consisting of four popular Chinese dramas are made to investigate the usefulness of these features. With the weight-sharing kernel applied, the proposed model yields lower error rates than the other baseline models in predicting the audience ratings.

Acknowledgement

This study is conducted under the "Social Intelligence Analysis Service Platform" of the Institute for Information Industry which is subsidized by the Ministry of Economy Affairs of the Republic of China.

References

1. Asur, S., Huberman, B.A.: Predicting the future with social media. In: Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01. pp. 492–499. WI-IAT '10, IEEE Computer Society, Washington, DC, USA (2010), http://dx.doi.org/10.1109/WI-IAT.2010.63
2. Bollerslev, T.: Generalized autoregressive conditional heteroskedasticity. Journal of econometrics 31(3), 307–327 (1986)
3. Cawley, G.C., Talbot, N.L.: On over-fitting in model selection and subsequent selection bias in performance evaluation. The Journal of Machine Learning Research 11, 2079–2107 (Aug 2010), http://dl.acm.org/citation.cfm?id=1756006.1859921
4. Cheng, Y.H., Wu, C.M., Ku, T., Chen, G.D.: A predicting model of tv audience rating based on the facebook. In: International Conference on Social Computing (SocialCom). pp. 1034–1037. IEEE (2013)
5. Danaher, P., Dagger, T.: Using a nested logit model to forecast television ratings. International Journal of Forecasting 28(3), 607–622 (2012)
6. Danaher, P.J., Dagger, T.S., Smith, M.S.: Forecasting television ratings. International Journal of Forecasting 27(4), 1215–1240 (2011)
7. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. The Journal of Machine Learning Research 9, 1871–1874 (Jun 2008), http://dl.acm.org/citation.cfm?id=1390681.1442794
8. Hsieh, W.T., Chou, S.c.T., Cheng, Y.H., Wu, C.M.: Predicting tv audience rating with social media. In: Proceedings of the IJCNLP 2013 Workshop on Natural Language Processing for Social Media (SocialNLP). pp. 1–5. Asian Federation of Natural Language Processing, Nagoya, Japan (October 2013), http://www.aclweb.org/anthology/W13-4201
9. Luxhøj, J.T., Riis, J.O., Stensballe, B.: A hybrid econometric-neural network modeling approach for sales forecasting. International Journal of Production Economics 43(2), 175–192 (1996)
10. Murray, I., Adams, R.P.: Slice sampling covariance hyperparameters of latent gaussian models. In: Advances in Neural Information Processing Systems. pp. 1723–1731 (2010)

11. Neal, R.M.: Bayesian Learning for Neural Networks. Springer-Verlag New York, Inc., Secaucus, NJ, USA (1996)
12. Posedel, P.: Analysis of the exchange rate and pricing foreign currency options on the croatian market: the ngarch model as an alternative to the black-scholes model. Financial Theory and Practice 30(4), 347–368 (2006)
13. Rasmussen, C., Williams, C.: Gaussian Processes for Machine Learning. The MIT Press, Cambridge, MA, USA (2006)
14. Schwaighofer, A., Tresp, V.: Transductive and inductive methods for approximate gaussian process regression. In: Advances in Neural Information Processing Systems (2003)
15. Vapnik, V.: The Nature of Statistical Learning Theory. Springer (2000)
16. Williams, C.K.I., Rasmussen, C.E.: Gaussian processes for regression. In: Advances in Neural Information Processing Systems. pp. 514–520. MIT press (1996)