Predicting Online News Popularity

Data Set: Online News Popularity

Author(s): Mingjian Shi (mingjian@usc.edu)

Pengyu Zhang (pengyuz@usc.edu)

May 8th, 2021

1. Abstract

People love to read news online nowadays. More and more editors and agencies tend to find out how popular the news is and what makes the news become popular based on different aspects. In this project, we are going to find the best machine learning model to predict the popularity of the news. Our data set comes from one of the largest news websites: Mashable. We explored many machine learning techniques including data pre-processing, feature engineering, and model selection method to improve the performance of our model. We used four different regression models: Non-linear Regression, Ridge Regression, SVR Regression, and KNN Regression. We use these four models to predict the number of shares for each article and compared their performance to provide the best machine learning model. We found out the best model for this problem is SVR Regression Model, our testing P_MSE is around 1.949 and the testing P_MAE is around 0.715. Our work may help the companies to determine the trend or make predictions on the popularity of the news in various aspects.

2. Introduction

Problem Statement and Goals

In this project, we use the 'Online news popularity dataset' to apply machine learning techniques to predict the number of shares(popularity) of each article. We use regression to solve this problem and build numbers of different models to predict the popularity. We used the dataset's features to find the best model for prediction. We will explore 4 different regression models: Non-Linear Regression, SVR, Ridge Regression, and KNN.

3. Approach and Implementation

3.1. Dataset Usage

The original training data set consists of 35644 samples of articles and it contains 58 total features in different aspects of measures. We split the original training data set into a training set and a validation set randomly by using train_test_split function from the sklearn library: 80% (28515 training samples) of the original

training data was used as the training set, the rest 20%(7129 validation samples) is used as the validation set.

We tried to use 5-fold cross-validation to make our training process more robust, but eventually, we gave up due to high computation time.

In the training process, we use the training set to train our model and then use the validation set to calculate performance as the measurement of selecting our model.

The original test data provided along the project is left untouched until the best model is found to make predictions. The test set was used only once when the best model is found to make the final predictions.

3.2. Preprocessing

Pre-processing is very important in our project if we want to improve our accuracy for the model since the data sets contain a lot of features with different relevance levels. We applied a few techniques listed below, and we choose not to use some techniques at the end by looking at the performance.

3.2.1 Normalization

Normalization is a very important step during Pre-processing. We normalize the training set by using the StandardScaler() from the sklearn library, we fit and transform our training data then we normalize the validation set by transforming it using the scalar based on the training set. For the test set, we first use StandardScaler() to normalize the entire training data and then apply the same scaler to the test set for normalization.

During data normalization, we assume that each feature comes from a standard normal distribution. Besides, we also assume training data, test data come from the same distribution, and the parameters are estimated from the training data because the classifier model is trained on that data. If we train the classifier on normalized training data and test on the test data that normalized by itself, it might give worse test accuracy because of data mismatch.

3.2.2 Remove Outliers

At the start of the project, we spot our training label and found the values of shares have a very large range. The ‘shares’ beyond two standard deviations are not visible and we thought they can’t give us much relative information. Furthermore, We found out using Median can better represent the data based on the number of ‘shares’. So we decided to filter out the training data based on where the ‘shares’ value is larger than the (mean +/- 2 Std), we do that by using our own defined function.

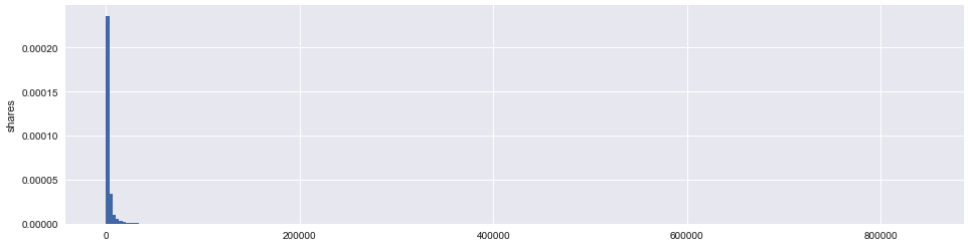

Figure 3.2.2 The distribution of the number of 'shares'

However, after some evaluations and logical thinking, we didn't remove any outliers. There are two main reasons that decide not to remove outliers. The first reason is that we don't want to lose the generality of the data set by moving some data that we think is irrelevant but actually not. The second reason is that we may get a better result for training and validation but a worse result for testing since we can't remove the outliers on the test set. So we didn't choose to apply this method eventually, but our codes will that we tried this method.

3.2.3 Down-Sampling

Due to the large data set, we have a very high computation time for training and validation when choosing the best model and parameters if we use the entire data set. So down-sampling is a key to our approach.

In our down-sampling process, we down-sampled our entire training data to 10% of the original data set first then split the training set and validation set. We applied our own down-sampling function and compared the performance for using sample function in the data frame, we eventually used the sample function with the parameter fraction equals 0.1 for the best performance.

	Number of Rows	Number of Cols
Entire Training data	35644	58
Down-sampled Training data	3564	58

Figure 3.2.3 Down-Sampling result

3.3. Feature engineering

We found some features that can be combined into one feature:

1. Features:['weekday_is_monday','weekday_is_tuesday','weekday_is_wednesday','weekday_is_thursday','weekday_is_friday','weekday_is_saturday','weekday_is_sunday'] can be combined into one feature called 'weekday' with values range from 1 to 7 and each represents Monday to Sunday. We used our own defined function 'FeatureCombine' to complete this task

2. Features:['data_channel_is_lifestyle','data_channel_is_entertainment','data_channel_is_bus','data_channel_is_socmed','data_channel_is_tech','data_channel_is_world'] can be combined into one feature called 'channel' with values range from 1 to 6 and each represents different channels. We used our own defined function 'FeatureCombine' to complete this task

However, after some evaluations and logical thinking, we eventually didn't choose this combined features method nor created any new features. The main reason behind this is that we don't want to change the relevance between features and data points. For example, 'weekday_is_monday'and 'weekday_is_sunday' were binary with values 0 and 1 before combination, but their values in the new feature 'weekday' are 1 and 7. For the linear case, the regressor will think 7 is less or more similar than 1 for representation, however, those features are independent of each other and they have their own mutual information and relevance with the data points. We don't want to change its nature of original features and data points nor their correlation. So we didn't choose to combine any features at the end, but our codes will show that we tried this method.

3.4. Feature dimensionality adjustment (PCA)

After the Normalization process by using StandardScaler(), we applied PCA to our training data then compared the performance when using only standardization, then we found there is not much difference by applying PCA during the pre-processing stage.

During the later section when we use non-linear regression, we applied PCA with n_components = 30 to the training data when the poly feature order goes up. We do this in order to reduce the computation time. More details will be provided in the later section.

3.5. Training, classification or regression, and model selection

3.5.1 Linear Regression

We use LinearRegression from the sklearn library as our baseline model. It will fit a linear model to make predictions on the targets by a linear approximation based on the sum of squares between the targets.

In our case, since we use LinearRegression as our baseline model, and there are no parameters to adjust for model selection, so we used the entire training data to train our model and predict on the test data.

The below figure summarizes the results of this baseline model

MAE	2798.4787747951978
P_MSE	6.93660065041956
P_MAE	1.571378832287694
R_Squared	0.04423602205078592
Modified_R_Squared	0.2034436393129061

Figure 3.5.1 Linear-Regression Baseline Model

3.5.2 Nonlinear Regression

We first use PolynomialFeatures() from the sklearn library to transform our features to the expanded φ-space of all polynomial combinations of the features with degrees from 2 to 4. We fit on the down-sampled training set and transform the down-sampled training set and down-sampled validation set to the expanded feature space

Then we use LinearRegression() to fit the expanded training set and training label in order to predict the label based on the expanded validation set.

For Poly degrees with 2 and 3, the computation time is reasonable, but not for degree 4. So for degree 4, we first used PCA with n_components = 20 to fit and transform the training set and to the validation set. Then we use the transformed training set and validation set to expand the feature spaces with poly degree 4, then use linearRegression() to fit after the PCA result and then predict.

We then compare all the performance measures for different degree and choose the best model

Note on cross-validation: refer to the Data-usage section

The following chart shows the validation performance results for different polynomial degrees when applied to the entire training set and validation set

	PolynomialFeatures (degree = 2)	PolynomialFeatures (degree = 3)	PolynomialFeatures (degree = 4), PCA(n=20)
P_MSE	1.41582926e+22	9.59203000e+02	6.89376421e+05
P_MAE	7.32488935e+09	9.48216817	61.5913286621

Based on the results from the above chart, the best model is the nonlinear regressor with poly degree = 3.

Then we applied our best Non-linear regression model on the entire training set and validation set.

The following chart summarizes the result:

Non-linear Regression Best Model Performance (Entire Training/Validation) (M=3)	
MAE	129280.2532328
P_MSE	746634.8436819324
P_MAE	76.04720778420848
R_Squared	-40266.36202841121
Modified_R_Squared	-85121.71863918853

3.5.3 Ridge Regression

The Ridge regression is based on linear regression plus a penalty term on the size of the coefficients. The Ridge regression model also solves a regression model and the loss function is the least-squares function with regularization of L2-norm. The objective function to minimize is ||y - Xw||^2_2 + alpha * ||w||^2_2. Based on the nature of the Ridge regression model, we think it is one approach to solve our problem

In our case, we are looking for 2 parameters: the polynomial degree and alpha for the Ridge regressor.

Note on cross-validation: refer to the Data-usage section

Like the non-linear regression we used in the previous section, we first use PolynomialFeatures() from the sklearn library to transform our features to the expanded φ-space. We fit on the down-sampled training set and transform the down-sampled training set and down-sampled validation set to the expanded feature space. Due to high computation time as the degree goes up for ridge regression, we only tried poly degrees 1 and 2.

For each poly degree(1 and 2), we used the training set to train our model and make predictions on the validation set for calculating performance measures for validation.

For each poly degree, we train our model and make predictions on 50 different alphas, and alphas are defined by using alphas = np.logspace(0, 6, 50, base=10)

So for each poly degree, we will have 50 calculated P_MSE and 50 calculated P_MAE, then find the best poly degree and alphas that give the minimum P_MSE and P_MAE.

The following chart shows the results we get for P_MSE and P_MAE over 2 different degrees for 50 alphas under each degree

alpha	1	1.325711	1.757511	2.329952	3.088844	...	323745.8	429193.4	568986.6	754312	1000000
M=1	11.52411	11.5315	11.54074	11.55207	11.56562	...	15.2808	15.33844	15.3826	15.41631	15.44196
M=2	238.1065	229.3683	219.1165	207.4821	194.7181	...	13.61368	13.84774	14.05343	14.2307	14.38298

Figure 3.5.3.1 P_MSE with 2 different degrees and 50 different alphas

alpha	1	1.325711	1.757511	2.329952	3.088844	...	323745.8	429193.4	568986.6	754312	1000000
M=1	1.927703	1.927923	1.928173	1.928509	1.928889	...	2.045211	2.047763	2.049722	2.05124	2.052392
M=2	5.761298	5.673793	5.572948	5.461597	5.340423	...	1.999576	2.004546	2.008644	2.011944	2.014692

Figure 3.5.3.2 P_MAE with 2 different degrees and 50 different alphas

The best poly order and best alphas that minimize the P_MSE and P_MAE are M = 1, alpha = 2023.58964773.

Then we applied our best Ridge regression model on the entire training set and validation set, the following chart summarizes the result

Ridge Regression Best Model Performance (Entire Training/Validation) (M=1, Alpha=2023.58964773)	
MAE	2979.8040480305776
P_MSE	8.815726562970264
P_MAE	1.6874410791816241
R_Squared	0.026617167081378845
Modified_R_Squared	-0.005067762601523862

3.5.4 Support Vector Regression

Support Vector Regression is a good approach to our problem since the model depends on a subset of the training data. Actually, Support Vector Regression is extended by the Support Vector Classification to solve the regression problem

In our case, we use SVM with the Radial Basis Function(RBF) kernel, and we need to find the two best parameters: C and gamma. The choice of C and gamma is the key to the performance of the model

Note on cross-validation: refer to the Data-usage section

We tried 50 different values of C, and for each value of C: we enumerated 50 different values of gamma and reported the total 2500 results of P_MSE and P_MAE. For each training of the model, we fit the down-sampled training data and down-sampled training label to train our SVR model, then predict on the down-

sampled validation data to get the predicted label for the use of calculating the performance measures.

The following chart shows 2500 result of P_MSE based on 50 different choices on gamma and C

C \ gamma	0.001	0.001326	0.001758	0.00233	0.003089	0.004095	···	568.9866	754.312	1000
0.001	0.982518	0.982516	0.982513	0.982509	0.982504	0.982498	···	0.684992	0.687787	0.694136
0.00132571	0.982516	0.982513	0.98251	0.982505	0.982498	0.98249	···	0.686388	0.697286	0.699915
0.00175751	0.982514	0.98251	0.982505	0.982499	0.982491	0.98248	···	0.698141	0.700189	0.707903
0.00232995	0.982511	0.982506	0.9825	0.982492	0.982482	0.982468	···	0.698315	0.705988	0.708772
0.00308884	0.982507	0.982501	0.982494	0.982484	0.98247	0.982453	···	0.702139	0.712089	0.719393
0.00409492	0.982503	0.982496	0.982486	0.982474	0.982457	0.982435	···	0.711734	0.720459	0.723644
0.00542868	0.982498	0.982489	0.982478	0.982463	0.982443	0.982416	···	0.717058	0.71917	0.715923
0.00719686	0.982494	0.982484	0.982471	0.982453	0.98243	0.982399	···	0.714718	0.713556	0.712388
···	···	···	···	···	···	···	···	···	···	···
568.986603	0.982524	0.982523	0.982523	0.982522	0.982522	0.982521	···	1.032245	1.104003	1.231075
754.312006	0.982524	0.982523	0.982523	0.982522	0.982522	0.982521	···	1.032245	1.104003	1.231075
1000	0.982524	0.982523	0.982523	0.982522	0.982522	0.982521	···	1.032245	1.104003	1.231075

Figure 3.5.4.1 P_MSE result with different gamma and C values

C \ gamma	0.001	0.001326	0.001758	0.00233	0.003089	0.004095	···	568.9866	754.312	1000
0.001	0.628445	0.628444	0.628444	0.628443	0.628442	0.62844	···	0.571255	0.573128	0.575697
0.0013257	0.628444	0.628444	0.628443	0.628442	0.628441	0.628439	···	0.572815	0.575916	0.578646
0.0017575	0.628444	0.628443	0.628442	0.628441	0.628439	0.628437	···	0.575366	0.577904	0.580923
0.00233	0.628443	0.628442	0.628441	0.628439	0.628437	0.628434	···	0.576555	0.579768	0.580936
0.0030888	0.628442	0.628441	0.628439	0.628437	0.628434	0.62843	···	0.577806	0.580791	0.584133
0.0040949	0.628441	0.62844	0.628438	0.628435	0.628431	0.628427	···	0.579775	0.583198	0.585376
0.0054287	0.62844	0.628438	0.628436	0.628432	0.628428	0.628422	···	0.581115	0.583937	0.584236
0.0071969	0.628439	0.628437	0.628434	0.62843	0.628425	0.628418	···	0.582719	0.583503	0.586202
···	···	···	···	···	···	···	···	···	···	···
568.9866	0.628446	0.628446	0.628446	0.628446	0.628445	0.628445	···	0.643442	0.664048	0.699636
754.31201	0.628446	0.628446	0.628446	0.628446	0.628445	0.628445	···	0.643442	0.664048	0.699636
1000	0.628446	0.628446	0.628446	0.628446	0.628445	0.628445	···	0.643442	0.664048	0.699636

Figure 3.5.4.2 P_MAE result with different gamma and C values

After gathering all the above results, the next step is to find the best gamma and C based on the minimum P_MES and P_MAE. We defined two functions to calculate the best gamma and C: The first function is called 'findBestPara_MSE()', which is to find the best parameters based on P_MSE, and if there are more than one best P_MSEs, choose the one with the best P_MAE; The second function is called 'findBestPara_MAE()', which is to find the best parameters based on P_MAE, if there are more than one best P_MAEs, choose the one with best P_MSE.

Based on the above two functions, we get two pairs of best [gamma, C] based on the down-sampled training and validation result: [0.00719686, 138.94954944] and [0.00719686, 79.06043211]. The first pair gives a better P_MSE(0.67512116) with a relatively higher P_MAE(0.56255786); The second pair gives a better P_MAE(0.55981157) with a relative higher P_MSE(0.67765866).

In order to determine which pair will give us the best model, we use both pairs of [gamma, C] to train on the entire training set and predict on the validation set.

The following chart summarizes the validation results for the two pairs of [gamma, C] over the entire training and validation set:

	First Pair [[0.00719686, 138.94954944]]	Second Pair [0.00719686, 79.06043211]
P_MSE	1.1529561529914896	1.1389524902128147
P_MAE	0.5710337177078634	0.5698028364365723

As we can see, the second [gamma, C] pair gives a better result.

Then we applied our best SVR model on the entire training set and validation set. The following chart summarizes the result:

SVR Best Model Performance(Entire Training/Validation) [gamma,C]=[0.00719686, 79.06043211]	
MAE	2226.0999205355274
P_MSE	1.1389524902128147
P_MAE	0.5699541352191909
R_Squared	-0.01621134303860594
Modified_R_Squared	0.8698055828307358

3.5.5 KNN Regression

We used Neighbors-based regression and specifically in our case we used KNeighborsRegressor from sklearn.neighbors. This technique is widely used in the machine learning field and the value of k is dependent on data. The regression is based on k-nearest neighbors. The prediction is made by the targets related to the nearest neighbors in the training set.

In our case, we are trying to find the value of k. We tried 16 different values of k from 5 to 20. And for each value of k: we fit the entire training data and entire training label to train our model, then predict on the entire validation data to get the predicted label for the use of calculating the performance measures. The reason we didn't use down-sampled data for training and validation is that the KNN model runs very fast, so we used the entire training and validation to improve the accuracy.

The following chart summarizes the P_MSE and P_MAE value for a different choice of k:

K	pMSE	pMAE
5	14.98001	1.494295
6	13.02721	1.462073
7	11.54221	1.455089
8	10.90777	1.464105
9	10.57449	1.468495
10	10.9624	1.469124
11	10.2112	1.459767
12	10.16886	1.454089
13	9.659042	1.44972
14	9.34505	1.449694
15	8.957083	1.44632
16	8.939192	1.446881
17	9.179971	1.455475
18	9.193426	1.462075
19	9.026484	1.458557
20	9.15755	1.454261

Figure 3.5.5 P_MAE,P_MAE result with different K

As we write commands to find out the best value of k=16

The following chart summarizes the performance for the best KNN model:

KNN Best Model Performance (Entire Training/Validation) (K = 16)	
MAE	2908.810886870529
P_MSE	8.939191908350137
P_MAE	1.4468810825132776
R_Squared	-0.006969376991294185
Modified_R_Squared	-0.019143861440728882

Note on cross-validation: refer to the Data-usage section

4. Analysis: Comparison of Results, Interpretation

In our approach, we have one baseline model: LinearRegression, and provide 4 different regression model approaches: Non-Linear Regression, Ridge Regression, SVR Regression, and KNN regression. For each model, we find the best performing parameters and report the performance measure on the entire validation result. The result is provided below and we found out that the best model is SVR.

	Linear Regression (Baseline)	Non-Linear Regression	Ridge Regression	SVR Regression	KNN Regression
Best Model Parameters	NA	M=3	M=1, Alpha=2023.58964773	[gamma,C]=[0.00719686, 79.06043211]	K=16
MAE	2798.4787747	129280.25323	2979.8040480	2226.0999205	2908.8108868
P_MSE	6.9366006504	746634.84368	8.8157265629	1.1389524902128147	8.9391919083 50137
P_MAE	1.5713788322	76.047207784	1.6874410791	0.5698028364	1.4468810825
R_Squared	0.0442360220	-40266.36202	0.0266171670	-0.015921993	-0.006969376
Modified R_Squared	0.2034436393	-85121.71863	-0.005067762	0.8701497349	-0.019143861

Based on the above comparison, we determined our best model is the SVR regression model with [gamma, C]=[0.00719686, 79.06043211]

Then we use our best model to train on the entire training data and predict the test data. The following chart summarizes the performance of our best model:

Best Model Performance (Entire Training/Test)	
MAE	2908.810886870529
P_MSE	1.9488751498926589
P_MAE	0.7151657983768334

R_Squared	-0.03669242
Modified_R_Squared	0.77620322
public p_MAE Score	0.715
public p_MSE Score	1.949

We used the Pipeline function from sklearn.pipeline to assemble the Scalar and Best model and then generate the final .pkl file.

Analysis:

1. Non-Linear Regression model:

   Based on the results we obtained, our non-linear regression model has bad performance on this dataset. It has a very large P_MSE and P_MAE, it also has a very large negative R_Squared value. We think the method of applying 3rd-degree poly transformation is not suitable for this dataset. However, if we explore more than 4 degrees poly transformation, we will have a very large expanded feature space, which will result in a very high computation time that we can't afford. So this model is not suitable for this problem

2. Ridge Regression model:

   Based on the results we obtained, our Ridge Regression model has a similar performance with the baseline model. It performs a lot better than the non-linear regression model, and this is due to the regularizer term in the algorithm of Ridge regression. We explored poly transform of degree 2 and found that performs worse than the first degree, so we choose to use order M =1 for Ridge Regression. However, what we can't explore is the higher degree's performance due to the nature of ridge regression and poly transformation. If we increase the degree of poly transformation, we can't afford the high computation time, so we don't have the chance to explore the higher degree performance.

3. SVR Regression Model:

   Based on our results, we choose this model as our best model based on the performance measures. But unfortunately, we still have a very small negative value for R_squared. This model has some great advantages in solving this problem. Support Vector Machines are known to solve the high dimensional space problem, which is a perfect match for our problem and the performance is within our expectation. Besides, it will equally penalize the mismatches and misestimates in both the highest and lowest outliers. Furthermore, the SVR model has very good generalization capability and also gives us a very reasonable computation time

compared to the previous models. Most importantly, this model gives the best performance among all the models we explored.

4. KNN Regression Model:

   The KNN Regression Model is easy to use and finds the best parameters with a very low computation time when applied to such a big data set. The algorithm is easy to apply. It is our second-best model, which performs better than the non-linear regression model and the ridge regression model. KNN model makes predictions based on the similarity between data points, it will have a good performance if the data points have a great similarity. However, the data set we explored doesn't have this advantage to make the KNN model the best one.

Comments:

Based on P_MSE and P_MAE, the non-linear regression performs worse than the baseline model, Ridge Regression and KNN regression performs similarly with each other, and both of them has similar performance level with the baseline model. During our project, we found that non-linear mapping over degree3 takes a lot of computation time, we think this is because of the large number of features, and this result is within our expectation.

Nearly all our model has a very small R_Squared value close to zero, Non-Linear regression has a very large negative R_Squared value. Even our best model has a value of -0.015 for validation, which is below our expectation. Our interpretation of this result will lead back to the feature selection and feature engineering. Better feature selection can improve the accuracy and computation time.

Normalization is a very important process during our project as well. Split training and validation set process should be done at the start of the process in order to keep the training and validation robust. Also, the cross-validation process can help us to make our model more robust when finding the best parameters. However, in our case, we dropped the cross-validation process eventually in order to save computation time.

5. Contributions of each team member

We developed our code and report together, each of us gives the same amount of support in all aspects. Both of us agree that we have the same amount of contribution to this project.

6. Summary and conclusions

In this project, we used four different regression methods(Non-Linear, Ridge, SVR, and KNN Regression) to solve the problem and compare their performance to make our decision on the best model. We explored some pre-processing techniques such as Normalization, Removal of outliers and Down-sampling. We also tried some feature enginerring method and PCA in order to reduce the dimentionality of the data set.

The best model we found is the SVR Model with parameters mentioned above,the P_MSE on testing set is 1.9488751498926589 and the P_MAE on testing set is 0.7151657983768334.

In the follow-on work, we would like to explore some better feature engineering method. Also, we would like to learn more machine learning models that is not mentioned in this class.

References

- scikit-learn-Machine Learning in Python [Online]. Available: https://scikit-learn.org
- What are outliers in the data? [Online]. Available: https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm
- Top 9 Feature Engineering Techniques with Python[Online].Available: https://rubikscode.net/2020/11/15/top-9-feature-engineering-techniques/
- Bishop, C. M. (2006). Pattern Recognition and Machine Learning (Information Science and Statistics) . Springer.
- Nelli F. Python Data Analytics With Pandas, NumPy, and Matplotlib . 2nd ed. 2018. Apress; 2018. doi:10.1007/978-1-4842-3913-1
- Alpaydin, E. (2020). Introduction to Machine Learning, fourth edition (Adaptive Computation and Machine Learning series) (fourth edition). The MIT Press.