Prediction of year of release of song

Sneha Shankar Narayan
Course: CS589
SpireID: 28197622
snehas@cs.umass.edu

Abstract

The problem of year prediction is to predict the year of release of a song given its audio features. I am modeling this problem as a regression problem. Users tend to listen to and prefer music that is closely associated with a certain era of their lives (Eg. high school) and more personalized recommendations can be provided by predicting the year a song was released and recommending those songs to certain users. Building a well tuned data pipeline can solve this problem with good accuracy.

1 Introduction

Music information retrieval has been a rich field open to a lot of experiments. Estimation of year of release of music has not been tackled before as a major endeavor. This has probably happened because the year of release of music is usually encoded in the meta data along with the artist name, album name and genre tags. I believe this is an important problem to solve because listeners often have particular affection for music from certain periods of their lives (such as high school), thus the predicted year could be a useful basis for recommendation. Furthermore, a successful model of the variation in music audio characteristics through the years could throw light on the long-term evolution of popular music [1]. It is also interesting to see if the type of music that is released in a certain era repeats itself over time which can be done by understanding the co-relation of the audio features of music and the time it was released. Using ridge regression this can be achieved with good accuracies.

2 Related Work

Thierry Bertin-Mahieux et al., 2011 [1] introduced the problem of year prediction when they introduced the Million Song Dataset. They modeled the problem as a regression problem and tried to solve it using the K Nearest Neighbor algorithm and the Vowpal Wabbit's default regression algorithm. They normalized the targets to be in the range of 0 and 1 as is being done is this paper and got the best Mean Squared Error as 8.79 from the Vowpal Wabbit algorithm.

Serr'a et al., 2012 also made use of the published dataset in order to characterize the evolution of western popular music over the last century. They define a number of patterns and metrics characterizing the generic usage of primary musical facets such as pitch, timbre, and loudness in contemporary western popular music and claim that the patterns have been consistently stable for the last fifty years [2]. They found three important trends in the evolution of musical discourse: the restriction of pitch sequences (with metrics showing less variety in pitch progressions), the homogenization of the timbral palette (with frequent timbres becoming more frequent), and growing average loudness levels (threatening a dynamic richness that has been conserved until today) [2]. They defined various periods in music which used the audio features in a more intricate way.

Casey et al., [3] specifies methods to do content based music information retrieval. Muller in [4] mentions that MFCCs, (Mel-frequency cepstral coefficients) are useful in capturing timbral characteristics and are used for a lot of music classification tasks. They also say that the idea of describing a pitch by means of its tone height and chroma is the foundation of how musical notes in Western Music are notated. The dataset being used is solely made up of MFCC-like features and thus can be used for year of release estimation of songs.

3 Proposed Solution

The solution that is presented here is to build a data pipeline with various stages. The first stage is to do preprocessing of data. The train and the test input features were normalized to have zero mean and unit variance. An output transformation was performed on the target values (which range from 1922 to 2011) to be between 0 and 1 considering 1922 as 0 and 2011 as 1, as suggested in [1].

The regression model used in my pipeline is Ridge regression which is a version of ordinary least squares with L2 regularization [5]. Linear regression is a parametric regression method that assumes the relationship between y, the targets and $\mathbf{x}$, the features is a linear function with parameters $\mathbf{w} = [w_1, ..., w_D]^T$ and b. Mathematically,

$$f_{Lin}(\boldsymbol{x}) = (\sum_{d=1}^{D} w_d x_d) + b = \boldsymbol{x}\boldsymbol{w} + b$$

Ordinary least squares selects the linear regression parameters to minimize the mean squared error (MSE) on the training data set. Ridge regression is the name given to regularized least squares when the weights are penalized using the square of the L2 norm.

$$||\boldsymbol{w}||_2^2 = \boldsymbol{w^T w} = \sum_{d=1}^{D} w_d^2 :$$

$$\boldsymbol{w^*} = \arg\min_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} (y_i - \boldsymbol{x_i w})^2 + \lambda ||\boldsymbol{w}||_2^2$$

$$= \arg\min_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} (y_i - \boldsymbol{x_i w})^2 ..st ||\boldsymbol{w}||_2^2 <= c$$

Optimal regularized weights are thus:

$$\boldsymbol{w^*} = (\boldsymbol{X^T X} + \lambda I)^{-1} \boldsymbol{X^T Y}$$

Ridge regression due to regularization reduces the possibility of very large weights overfitting to outliers. The final stage of the pipeline is, hyperparameter selection through K fold cross-validation with 5 cross validation folds. Ridge regression has a hyperparameter 'alpha'. Small positive values of alpha improve the conditioning of the problem and reduce the variance of the estimates. Alpha corresponds to $(2 * C)^{-1}$ in other linear models such as LogisticRegression [6].

This pipeline with data pre-processing, feature selection and linear regression provides the best possible solution for the problem of estimation of release year as shown in the Results section.

4 Data Set

This dataset is a subset of the million song dataset [7], a collaboration between LabROSA (Columbia University) and The Echo Nest and was prepared by T. Bertin-Mahieux. This is a dataset tuned for the prediction of the release year of a song from audio features [8]. The songs are mostly western, commercial tracks ranging from 1922 to 2011, with a peak in the year 2000s. The dataset is present in the University of California, Irvine's machine

learning repository [9]. The dataset consists of real values, with 515345 data instances with 90 attributes. This has to be split into 463715 training instances and 51630 test instances in order to avoid the 'producer's' effect by making sure no song from a given artist ends up in both the train and test set [8].

The first twelve attributes denote the timbre average and the rest 78 attributes denote the timbre covariance. Features are extracted from the 'timbre' features from The Echo Nest API. The average and covariance over all segments are taken, each segment is described by a 12 dimensional timbre vector. The main feature that was used to construct this dataset is the `segments_timbre` feature from the million song dataset which has a shape of (935, 12). This consists of MFCC-like features for each segment. The mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum") [10]. According to [4] MFCCs are great for music information retrieval tasks, and year of release estimation is one such task.

5 Experiments and Results

5.1 Building the pipeline

A variety of experiments were performed in order to ascertain which regression model works the best on the task at hand. Since the dataset is huge with half a million data points, a smaller dataset was created with 10000 training instances and 1000 test instances by randomly sampling the original train and test sets. This dataset will be referred to as 'sampled set'. On the sampled set I ran Linear Regression, Decision trees and KNN because they work well for large amounts of data [11], [1]. I also ran SVM because of its low bias and non sensitivity to outliers. I wanted to see how an ensemble performed and thus ran the random forest regressor. All my methods gave me RMSE values around an average of 10 with Ridge Regression being the lowest at 9.2409245. These results are comparable to the ones presented in [1]. Since scikit-learn requires standardized datasets[12], I normalized my train and test input features to have zero mean and unit variance. I also performed an output transformation on my target values (which range from 1922 to 2011) to be between 0 and 1 considering 1922 as 0 and 2011 as 1, as suggested in [1]. After performing pre-processing and output transformation my results are as shown below:

Table 1: Results after pre-processing - Sampled data set

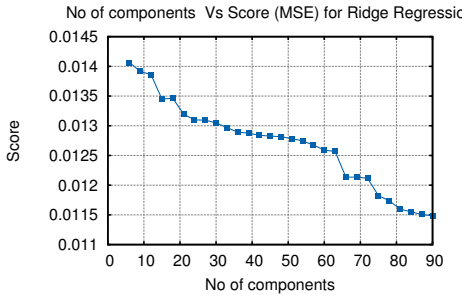
REGRESSION MODEL	RMSE	TIME (IN SECONDS)
Support Vector Regression(kernel='rbf')	0.111690902279	7.288200
Linear Regression (Ridge)	0.107254452382	0.247575
KNN (K = 5)	0.112124630821	3.641467
Decision tree Regression	0.161593191535	1.639252
Random forest Regression	0.10551376044	11.689569

We can notice that due to standardizing the dataset the RMSE seems to have improved. Looking at the results, I decided to explore Linear Regression and Support Vector Regression further. Because of the theory that ensemble always performs better I believed it is a better use of my time to focus on the basic methods.

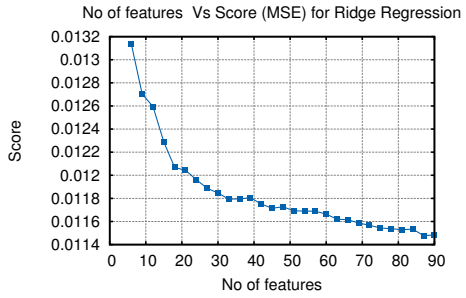
I experimented with dimensionality reduction. Since my data only has 90 dimensions, using PCA to reduce the dimensions using Minka's Maximum likelihood estimate [13] in fact made the RMSE slightly worse (For Ridge regression, RMSE was 0.1073, for SVM the RMSE was 0.111799). Kernal PCA presented the same value and FastICA performed slightly worse (For Ridge regression, RMSE was 0.10749). I also tweaked the hyperparameters for SVM and I did not get any better results. Using the Select K best feature selection process on SVM also did not make the RMSE better, it gave a value of 0.11169127899. Since tuning

SVM did not help and s is evident by the results in the next section, SVM does not scale, Ridge Regression became the obvious choice for my pipeline.

I applied backward stepwise selection for Ridge Regression and got an RMSE of 0.107809576234 which is worse than the RMSE for Ridge Regression without feature selection. When I applied the Select K Best feature selection process to Ridge it picked 87 features out of 90 and gave a RMSE of 0.107299905681 which is again slightly worse than the defaults. Following shows dimensionality reduction and feature selection in Ridge regression.



(a) Dimensionality reduction



(b) K best feature selection

Figure 1: Feature engineering of Ridge Regression

I performed hyper parameter optimization using K fold cross validation with 5 cross validation folds to figure out the best value of 'alpha' in ridge regression. Running through the values of alpha in [0.1,0.01, 0.001, 0.5, 1.0, 2.0, 10.0, 0.0001, 0.0000001] I found out that the best value of alpha is 0.0000001. Plugging in this value, the result obtained by my final pipeline for the sampled data is **0.107254049674**, a slight improvement. The following graph shows the hyperparameter sweep.

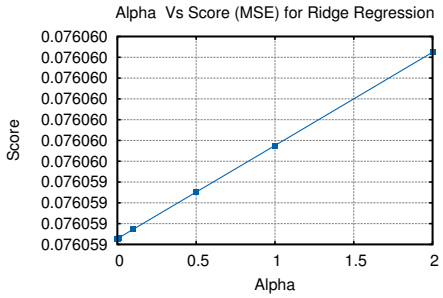


Figure 2: Hyperparameter optimization using K-fold cross validation

5.2 Learning on scale

5.2.1 Vowpal Wabbit

Vowpal wabbit (VW) is an open source fast out-of-core learning system library and is an efficient scalable implementation of online machine learning and support for a number of machine learning reductions, importance weighting, and a selection of different loss functions and optimization algorithms [14]. I used VW [15] to perform a variation of gradient descent with a squared loss function, and it ran through my entire dataset with a half million data cases within 2 minutes. (112 seconds) This method produced an RMSE of 0.157965646748. VW also has an implementation to perform neural network regression. Neural network regression with 5 hidden units produced an RMSE of 0.122139238458 and ran in exactly 2 minutes. Although VW is super fast and produces nice results, tweaking the hyperparame-

ters like the loss function and regularization factors did not improve the RMSE significantly. As of now, VW does not have an implementation for non binary SVMs.

5.2.2 Amazon EC2

After running experiments on the sampled data and figuring out a range of hyperparameters to optimize over, the experiments were performed on a c4.4xlarge EC2 instance with 16 cores and 30GB of RAM. The following results were achieved:

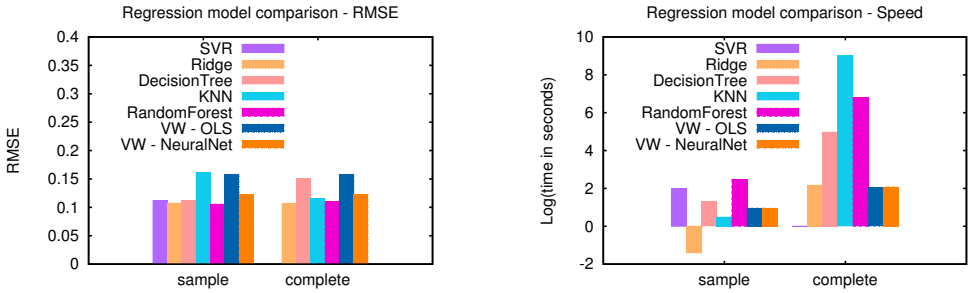
Table 2: Full dataset results

REGRESSION MODEL	RMSE	TIME (IN SECONDS)
Support Vector Regression	*	*
Linear Regression (Ridge)	0.106855750824	8.697165
Tree	0.150814081202	145.586453
KNN (K = 5)	0.115058561634	8201.498334
Random forest	0.110054477145	891.714105

* - I ran SVM using sklearn on a c4.4xlarge machine and it did not finish in 12 hours. I also ran SVM using svmlight, a fast C implementation of SVM on a r3.2xlarge EC2 machine with 64 GB of RAM and it did not finish in 8 hours. Since my dataset has 500K instances, SVM seems to have issues completing fast.

5.2.3 Summarizing the results

The following graph shows the summary of my results. The RMSE in the sampled and complete datasets are comparable and thus shows sampling is a good technique while dealing with large datasets. The time of execution for the complete dataset is obviously higher. I chose to use a log scale for time because I had to represent a range of values from 0.1 seconds to 8000 seconds. As is evident from the graphs the best accuracy and the best speed is obtained by Ridge regression. Applying the built data pipeline on the entire dataset I get a final RMSE of **0.106855738286**.



(a) RMSE from different models

(b) Time of execution of different models

Figure 3: Comparison of regression models

6 Conclusion

By building an extensive data pipeline I showed how year of release estimation of a song can be achieved. I also presented a few ways that are available that can easily perform learning at a scale: Vowpal Wabbit and using Amazon EC2. Ridge regression performs the best in terms of speed and accuracy and is the clear winner. I would have liked to finish my SVM runs but I believe my dataset was too huge for SVMs to work with. Since ridge regression works so well, this can be plugged in to do dynamic year estimation in any music recommender system for fast outputs.

References

[1] Bertin-Mahieux T., Ellis D. P. W., Whitman B., Lamere P. The million song dataset. In: Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR), 591596 (2011).

[2] J. Serr'a, A . Corral, M. Bogun a, M. Haro, and J.L. Arcos. Measuring the evolution of contemporary western popular music. Scientific Reports, 2, 2012.

[3] Casey M. A. et al. Content-based music information retrieval: current directions and future challenges. Proc. of the IEEE 96, 668696 (2008).

[4] Meinard Mller (2007). Information Retrieval for Music and Motion. Springer. p. 65. ISBN 978-3-540-74047-6.

[5] From CS589 slides in class.

[6] `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html`

[7] `http://labrosa.ee.columbia.edu/millionsong/`

[8] `http://archive.ics.uci.edu/ml/datasets/YearPredictionMSD`

[9] A. Frank and Arthur Asuncion. The UCI Machine Learning Repository. `http://archive.ics.uci.edu/ml`, 2010. University of California, Irvine, School of Information and Computer Sciences.

[10] `http://en.wikipedia.org/wiki/Mel-frequency_cepstrum`

[11] `http://en.wikipedia.org/wiki/Decision_tree_learning#Decision_tree_advantages`

[12] `http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing`

[13] Thomas P. Minka: Automatic Choice of Dimensionality for PCA. NIPS 2000: 598-604

[14] `http://en.wikipedia.org/wiki/Vowpal_Wabbit`

[15] J. Langford, L. Li, and A. L. Strehl. Vowpal wabbit (fast online learning), 2007. http://hunch.net/vw/.