Music Release Year Prediction via Regression

APMA 990	Final Project
Submitted by:	Nathan King
Student #:	301222376
Submitted to:	Dr. Paul Tupper
Date:	April 23, 2014

Contents

1 Introduction

Here we aim to estimate the release year of songs based on audio features. Year prediction for music has been studied little [1], but is of interest due to its possible applications. Recommending media that is enjoyable to a customer is desirable in industry. Suggesting media correctly to a customer likely results in a returning customer looking for advice. Movie recommendation by NetFlix or music recommendation by iTunes are some examples. Songs can be recommended to people solely based on the release year since people associate songs with notable times in their life (high school years, university years, etc.). Year prediction of a song can also be related to the genre of a song; most music genres coincide with certain time eras. Genre identification of music is an application that has been more immensly studied than year prediction [2].

Data used for this project was created by Bertin-Mahieux et al. [1]. The data consists of 90 input variables and one output variable. A somewhat vaguely defined quantity called *timbre* is used to create the 90 input variables. Timbre is the quality of a musical note or sound that distinguishes different musical instruments or sounds [3]. A timbre vector consisting of 12 different features is used: the first component represents the average loudness of the segment, second emphasizes brightness, third is correlated to the flatness of a sound, fourth to sounds with a stronger attack, etc. The 12 values in the timbre vector are positive, unbounded reals. Covariance of the timbre vector makes up the remaining 78 input variables (only nonredundant covariances are used). The one output variable is the predicted release year of the song. The training data involves 463,715 songs, while 51,630 songs make up the testing data. Release years range from 1922 to 2011.

Many different methods are applied here to this year prediction problem. Analysis using k–nearest–neighbours and Vowpal Wabbit [4] was done in [1]. Results for the latter methods will also be discussed within. Other regression methods are implemented; Linear model fit by least squares, Stepwise Subset Selection, Ridge regression, Lasso and Boosting are explored. Subset selection and skrinkage methods are investigated due to the size of the data set and the large 90-dimensional feature space. Stepwise subset selection and shrinkage methods and may be more efficient when applying the model to testing data and may benefit from reduced variance. Boosting is a unique approach that uses many "weak" predictors to build a "strong" predictor.

The project is organized as follows. Section 2 consist of descriptions and some expected behaviour of the above methods. Results from the implementation of the regression methods are given in section 3. Section 4 concludes giving some insight into which methods are appropriate.

2 Methods to Predict Release Year

Nearly all the methods explored in this project are standard, well-developed methods. Linear regression, k–nearest–neighbours, Ridge regression, Lasso and Stepwise Subset Selection are well-known regression methods. Boosting is another standard method but is possibly the most novel regression approach. Descriptions of the above methods follow that of Hastie et al. [5].

2.1 Linear Regression and k–Nearest–Neighbours

Possibly the most simple methods in machine learning are Linear regression and k–nearest–neighbours. Linear regression makes a strict assumption about the structure of the model, which leads to stable but sometimes inaccurate predictions. Mild structural assumptions about the model are made by k–nearest–neighbours, giving unstable but often accurate predictions. Generally, Linear regression has low variance and high bias, while k–nearest–neighbours has high variance and low bias.

Linear regression fits a hyperplane through the feature space (our 90 timbre features). The optimal hyperplane is determined by minimizing a criterion that depends on the N observations. The criterion used for a least squares fit is the residual sum of squares (RSS). To express this mathematically let $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,p})^T$ be the i-th input of the data set. If X_i is augmented to be a $p+1$-dimensional vector, with one as its first component and X_i for the remaining p, then an intercept in the linear model can be accomodated. The linear model to be fit to the N inputs, X_i where $i = 1, \ldots, N$, can be writen as

$$f(\mathbf{X}) = \mathbf{X}\beta, \tag{1}$$

where $\mathbf{X}$ is an $N \times (p+1)$ matrix with rows X_i and $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$.

The parameter β has to be optimized using the N observations. For this we minimize the RSS, which is defined as

$$\begin{aligned} \mathrm{RSS}(\beta) &= \|Y - \mathbf{X}\beta\|_2^2 \\ &= (Y - \mathbf{X}\beta)^T(Y - \mathbf{X}\beta), \end{aligned}$$

where $Y = (y_1, y_2, \ldots, y_N)^T$ is a vector whose components correspond to the output given each X_i. It can be easily shown that if $\mathbf{X}$ has linearly independent columns then a unique and optimal β is given by

$$\hat{\beta} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-\mathbf{1}}\mathbf{X}^T Y.$$

For any new input X_0, one predicts that the output is

$$\hat{y}(X_0) = \tilde{X}_0(\mathbf{X}^T\mathbf{X})^{-\mathbf{1}}\mathbf{X}^T Y,$$

where $\tilde{X}_0$ is an augmented X_0 with a one in the first component. The notation used here will try to be consistant as follows: scalars are given by lowercase letters, vectors are uppercase letters and bold uppercase letters denote matrices.

The k–nearest–neighbours method uses observations closest to X_0 to form the prediction $\hat{y}$. Let $N_k(X_0)$ denote the k–neighbourhood of X_0 defined by the k closest points X_i of all N inputs. The k closest points to X_0 are determined using Euclidean distance in the feature space $\mathbb{R}^p$. The model given by k–nearest–neighbours is

$$\hat{y}(X_0) = \frac{1}{k} \sum_{X_i \in N_k(X_0)} y_i. \tag{2}$$

In words, k–nearest–neighbours returns the average output of the k closest inputs to X_0 as the predicted output $\hat{y}$.

One important difficulties with k–nearest–neighbours is to ensure the model is not overfit to the training data. In the extreme case of $k = 1$, the model has zero error in predicting the output for the training inputs. Applying this same model to a input X_0, not in the training set, will more than likely give an incorrect resultt. This is due to the high variance of 1–nearest–neighbours. It seems that k–nearest–neighbours has only a single parameter, k, however the effective number of parameters is roughly N/k. This can be shown rigourously (Exercise 7.6 in [5]), but can be explained easily for the case of nonoverlapping N_k. If the neighbourhoods are nonoverlapping, there are roughly N/k neighbourhoods. One parameter, a mean, is fit in each of the neighbourhoods. One has to be careful when determining value of k, since the effective number of parameters is important. Cross validation can be used to determine k, which is explained in section 3.

2.2 Stepwise Subset Selection and Shrinkage Methods

The next three methods reduce the size of the features space to decrease variance. In reducing the variance the bias of the predictor increases, however the hope is that a decrease overall in error is achieved. Stepwise subset selection is a technique that eliminates features in $\mathbb{R}^p$ based on their statistical significance in a regression model. Ridge regression and Lasso are instead skrinage methods; they decrease the effective number of parameters.

One can carry out Stepwise subset selection in a forward manner, backward manner or both directions. The implementation used in this project does Stepwise subset selection in both forward and backward directions. At each step, the p–value of an F–statistic is computed to test models with and without a potential feature (see MATLAB's documentation for `stepwisefit`). If a feature is not currently in the model, the null hypothesis is that the feature would have a zero coefficient if added. If we reject the null hypothesis, the feature is added to the model. Instead, if a term is currently in the model, the null hypothesis is that the term has a zero coefficient. If the null hypothesis is accepted, the feature is removed from the model.

First the regression model is fit to an initial subset (the whole training set in our numerical tests). The method then interchanges between two steps:

1. If any terms not in the model have p-values less than 0.05, add the one with the smallest p value and repeat this step. If no features have p-value less than 0.05 go to step 2.

2. If any features in the model have p-values greater than 0.10, remove the one with the largest p value and go to step 2; otherwise, end.

The method may build different models from the same set of potential features, dependent on the initial subset. The method terminates when no single step improves the model. Stepwise models are greedy algorithms and do not guarentee the optimal subset.

Ridge regression and Lasso skrinage methods are similar to eachother. Both are identical to Linear regression except they add a term to the RSS. The term added causes shrinkage by penalizing the size of the β_j coefficients. A new RSS to optimize for the vector β is acquired. For Ridge regression β is obtained from

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{i,j}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}, \tag{3}$$

where λ is a parameter that determines the amount of penalization. The penalization term for Ridge regression is differentiable and a similar result to Linear regression can be obtained. The criterion in (3) can be writen in matrix form as

$$\text{RSS}(\lambda, \beta) = (Y - \mathbf{X}\beta)^T(Y - \mathbf{X}\beta) + \lambda\beta^T\beta,$$

which gives the optimal β of

$$\hat{\beta}^{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^TY.$$

Instead of using a L_2 penalty $\sum_{j=1}^{p}\beta_j^2$, the Lasso method implements a L_1 penalty term $\sum_{j=1}^{p}|\beta_j|$. This slight change creates some important differences between Ridge regression and Lasso. In full, the Lasso estimate is given by

$$\hat{\beta}^{\text{lasso}} = \arg\min_{\beta}\left\{\frac{1}{2}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p}x_{i,j}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right\}, \tag{4}$$

where λ is again a tunable parameter, which is determined by cross validation in section 3. The L_1 penalty term makes the solution nonlinear in the y_i, thus no closed form expression for β^{lasso} can be found. There are algorithms to find the optimal $\hat{\beta}^{\text{lasso}}$, which are not discussed here, that have the same computational cost as Ridge regression for varying λ (see [5]). One important advantage of Lasso is that it causes some coefficients β_j to be zero. Ridge regression only shrinks the β_j towards zero. Lasso can therefore be thought of as a continuous subset selection technique. Solutions from Ridge regression and Lasso are not equivalent under scalings of the input. Ridge regression and Lasso therefore standardized inputs before computing $\hat{\beta}^{\text{ridge}}$ and $\hat{\beta}^{\text{lasso}}$.

2.3 Boosting

The description of Boosting is given in terms of a classification method for exposition simplicity. A brief discussion of how Boosting can be extended to regression problems is given thereafter. For classification methods we have K classes belonging to $\mathcal{G} = \{1, 2, \ldots, K\}$. Capital G represents any class g_l, $l = 1, 2, \ldots, K$. Boosting is one of the most powerful learning tools developed in the last twenty years [5]. It was designed for classification problems but can be extended for regression problems. Boosting is motivated by the desired to build a powerful classifier from a committee of "weak" classifiers. There are many different algorithms for Boosting but here we explain AdaBoost, which is the most popular in practice.

Consider two classes coded as $y_i \in \{-1, 1\}$. From an input X_i, a classifer $G(X_i)$ returns output in $\{-1, 1\}$. The error rate on the training set is

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N}I(y_i \neq G(X_i)).$$

A weak classfier is defined as one whose $\overline{\text{err}}$ is slightly less than that of random guessing. Boosting sequentially applies a weak classifier to repeatedly modified versions of the data.

A sequence of weak classifiers, $G_m(X)$ for $m = 1, 2, \ldots, M$, are thus produced. Predictions from all $G_m(X)$ are combined with a weighted majority vote for a final prediction

$$G(X) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(X)\right), \tag{5}$$

where parameters α_m are determined from the boosting algorithm.

Modification of the data at each boosing step is done by applying weights $w_1, w_2, \ldots, w_N$ to each training observation (X_i, y_i) for $i = 1, 2, \ldots, N$. Initially all weights are $w_i = 1/N$. For each sucessive boosting step, $m = 2, 3, \ldots, M$, the observation weights are modified and a weak classifier is reapplied. At step m, observations that are misclassified by $G_{m-1}(X)$ (previous steps classifier) have increased weights. Weights will however decrease for observations that were classified correctly. Explicitly, weights are updated as $w_i \leftarrow w_i \exp[\alpha_m\, I(y_i \neq G_m(X_i))]$, explicitly. As boosting proceeds, each successive classifier, $G_m(X)$, concentrates on training observations missed by previous classifiers $G_1(X), G_2(X), \ldots, G_{m-1}(X)$. To compute α_m a weighted error rate is computed for each classifier as

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i\, I(y_i \neq G_m(X_i))}{\sum_{i=1}^{N} w_i},$$

then

$$\alpha_m = \log\left(\frac{1 - \text{err}_m}{\text{err}_m}\right). \tag{6}$$

The implemention here is for regression, which uses regression trees as weak classifiers. The weighting of the data is determined using different formulas then given here. The overall concept is however the same as with classification methods. For the regression method an appropriate loss function must be incorporated. For more details see Chapter 10 of Hastie et al. [5].

3 Numerical Results

This project surveys many machine learning methods, which attempt to determine the release date of music. Numerical experiments here aim to illustrate which method is the most appropriate for this data. All methods are trained using 463,715 songs and tested on 51,630 different songs. Six regression methods are implemented and explored. Results for a novel method called the Vowpal Wabbit are also stated here.

Some of the methods have parameters that need to be determined. We use cross validation to determine the optimal parameter when necessary. A brief description of cross validation is therefore given here first. Cross validation is one of the simplest, and probably most widely used, methods for estimating prediction error. A direct estimate of the extra–sample error, $\text{Err} = \mathbb{E}[L(Y, \hat{f}(X))]$, is used. The extra–sample error is the average error when the predictor $\hat{f}(X)$ is applied to an independent test set, from the joint distribution of X and Y. There are different types of cross validation, here K–fold cross validtation is implemented.

K–fold cross validation splits the training data into K roughly equal parts (or folds). $K - 1$ folds are used to train the model, while the k-th fold is reserved for validation, i.e.

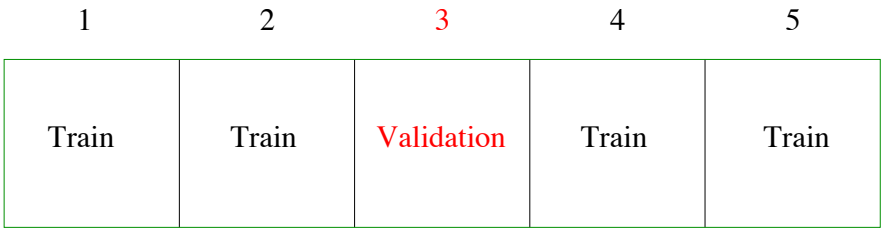

Figure 1: Illustration of 5–folds for cross validation.

testing. Figure 1 depicts the situation when $K = 5$. The thrid fold in figure 1, is reserved for validation, whereas folds one, two, four and five are used to train. After the model is trained the prediction error is calculated for prediction on the k-th fold (thrid in figure 1). The validation fold is changed and the model trained on a different $K - 1$ folds. This is done until all K folds have been the validation fold. The K prediction errors can be used to determine the optimal parameter.

Let $\kappa : \{1, \ldots, N\} \to \{1, \ldots, K\}$ be an indexing function that gives the random partition of N observations into K folds. Let $\hat{f}^{-k}(X, \alpha)$ be the predictor that is trained on the data with the k-th fold removed. The cross validation estimate of prediction error is then

$$CV(\hat{f}, \alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i, \alpha)). \tag{7}$$

The cross validation error is an estimate of the test error curve, so the optimal parameter $\hat{\alpha}$ is the one that minimizes $CV(\hat{f}, \alpha)$. There are instances when $CV(\hat{f}, \alpha)$ asymptotes. In such cases conditions on the standard deviation of $CV(\hat{f}, \alpha)$ helps determine $\hat{\alpha}$.

Regression methods are implemented using many of MATLAB's built–in machine learning functions. A benchmark predictor is the constant prediction of the mean release year. For the dataset here more songs are from later years, which gives a mean of 1998.4. There is an average absolute difference of 8.11 years between the song release dates and 1998.4. The first method tested is k–nearest–neighbours, which was also applied in [1]. Cross validation was not applied due to high computational cost of k–nearest–neighbours. Results for 1–nearest–neighbour and 50–nearest–neighbours are given here. It is found that 1–nearest–neighbour has much larger error than 50–nearest–neighbours. The 1–nearest–neighbour method predicts on average 9.80 years away, whereas 50–nearest–neighbours predicts on average 7.52 years away.

Linear regression does better than k–nearest–neighbours. Linear regression gives prediction on average of 6.80 years from the true release dates. There was no free parameter to be specified for Linear regression. For Ridge regression however, the parameter λ has to be determined in some manner. To implement cross validation with Ridge regression one must compute the effective degrees of freedom, df, in the model. Computing df for Ridge regression requires the computation of the singular values of $\mathbf{X}$, which is memory intensive. This is possibly why MATLAB does not have cross validation built into Ridge regression, whereas Lasso does.

Ridge regression is instead trained with multiple λ values. We then predict on the testing

Method	Difference	Std. Dev.	Maximum	Minimum	CT (sec)
Const. Prediction	8.11	7.21	71.39	4×10^{-1}	0.1
1–NN	9.80	10.1	80	0	3042
50–NN	7.52	6.93	70.04	0	3068
Linear Regression	6.80	6.65	64.14	1×10^{-4}	61
Ridge Regression	6.80	6.65	64.14	1×10^{-4}	6
Lasso	6.80	6.65	64.13	2×10^{-4}	117
Stepwise Selection	6.80	6.65	64.11	7×10^{-5}	283
Boosting	6.78	6.60	69.65	5×10^{-5}	1158
Vowpal Wabbit	6.14	—	—	—	—

Table 1: Average absolute difference between the song release date and predicted date for each regression method. Standard deviation in the differences is specified, along with maximum and minimum differences. Computation time (CT) for each method is in the last column.

data and determine a λ value that gives the minimum test error. This approach is justified here since we have such large amounts of data. Ridge regression gives identical results to Linear regression with λ taken as $\lambda = 0.0001$. This is understandable since Ridge regression is equivalent to Linear regression as $\lambda \to 0$. Ridge regression is on average 6.80 years away from the true release dates. Cross validation determines the optimal λ for the Lasso method as $\lambda = 6.9171 \times 10^{-4}$. The Lasso method has an average absolute difference of 6.80 years also. Stepwise selection produces the same prediction error as Linear regression, Ridge regression and Lasso. It seems that skrinkage methods and Stepwise subset selection methods have little benefit.

Boosting does give a small advantage over the simple Linear regression method. The increased complexity of Boosting may not justify the moderate decrease in error. Boosting here is implemented using 100 regression trees. Boosting obtains an average absolute difference of 6.78 years between the actual release year and the predicted. The advangtage of 0.02 years $\approx$ 7 days is regarded as unimportant, since the prediction error is almost 7 years. Note also that, between Linear regression, Ridge regression, Lasso and Stepwise subset selection results are actually different by about 0.00005 years $\approx$ 26 minutes, which is minuscule relative to years.

Vowpal Wabbit is a final method that was created for industrial, large scale datasets. Vowpal Wabbit is an open source machine learning library developed originally at Yahoo! Research, currently at Microsoft Research. The Vowpal Wabbit code uses a set of techniques for learning linear predictors with convex losses. Computation on terascale datasets, with trillions of features, billions of training examples and millions of parameters takes an hour using a cluster of 1000 machines [6]. To the knowledge of the authors in [6], Vowpal Wabbit is the most scalable and efficient linear learning system as of 2011. Due to compilation problems Vowpal Wabbit was unable to be used, however the average absolute difference of 6.14 years was calculated in [1]. Results for all regression methods are summarized in Table 1.

Average absolute difference between the true release year and predicted release year is the most important measure of accuracy. One can also consider the standard deviation of

the differences, the maximum difference and minimum difference. A final comparison for the methods is their computation time (CT). These quantities are all included in Table 1. Computation times do not include cross validation computation. Cross validation is used to determine a parameter, then computation time is calculated using this single parameter. The computation time is computed using MATLAB's tic/toc commands, which are not overly accurate but can be used for comparison.

It can be seen from Table 1 that 1–nearest–neighbour is unsatisfactory. It gives a larger average absolute difference than the constant predictor and has a large computation time. Standard deviation in the differences and maximum difference is also larger than the corresponding constant predictor results. 50–nearest–neighbours do better, by about half a year compared to the constant predictor, but have an unfortunately large computation time. Calculating the neighbourhoods $N_k(X_0)$ involves simple arithmetic. Computation time could therefore be decreased by calculating $N_k(X_0)$ in parallel on a GPU.

The remaining regression methods only differ slightly in their results. The only sizable difference between Linear regression, Ridge regression, Lasso, Stepwise subset selection and Boosting are their computation times. The maximum difference is around 64 years except about 70 years for Boosting. The minimum difference is basically zero for all the latter methods. Computation times are quite fast for Linear regression, Ridge regression, Lasso, Stepwise subset selection. Ridge regression is considerably faster then all the methods besides the constant predictor. Boosting has the second largest computation time, only less than k–nearest–neighbours. The slightly smaller prediction error, compared to Ridge regression, is not beneficial due to nearly 200% the computation time.

4 Discussion

Six different regression methods were studied here in the context of music release year prediction. The ambition here was to analyze regression techniques and determine the most appropriate method. None of the methods attained an average absolute difference as small as Vowpal Wabbit. The average absolute difference for Vowpal Wabbit was 6.14 years, which is not drastically better than the constant predictor of 1998.4. The Vowpal Wabbit is built for large scale datasets, but one must have access to computer clusters for ideal efficiency.

For the more standard methods dicussed, most give basically the same average absolute difference between the true release date and predicted date. An exception is with k–nearest–neighbours, where the average absolute difference is worse and the computation time is large. It is found that Ridge regression has the smallest computation time and an average absolute difference of 6.80 years. Boosting does slightly better with 6.78 years, but is much more computationally intensive. The prefered method out of the six is therefore Ridge regression.

One might suspect that classification methods may also be a viable option for this dataset. Classification methods such as, LDA, QDA and Logistic regression, were also coded but gave much larger average absolute differences. It is possible that there are too many classes for classification methods to be adequate. Regression methods are the more natural method to use here since years are a continuous quantity. Eventhough the output data are whole number years, true release dates would be decimal values.

It is apparent that the music release year prediction problem is a difficult learning prob-

lem. The maximum difference between the true release date and predicted date is, for all methods, between 64–80. The true song release dates are between 1922 and 2011, which is only 89 year time period. The standard deviation in the differences is also quite large (nearly the same size as the average absolute differences). The methods only do better by about one and a half years compared to the constant mean predictor. This means that the added complexity of the methods do not significantly gain accuracy. Even the Vowpal Wabbit method is only more accurate by about two years. A more novel approach is warrented for this demanding machine learning problem.

References

[1] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[2] T. Bertin-Mahieux, D. Eck, and M. Mandel, "Automatic tagging of audio: The state-of-the-art," *Machine Audition: Principles, Algorithms and Systems. IGI Publishing*, 2010.

[3] T. Jehan and D. DesRoches. (2011, September) Analyzer documentation. [Online]. Available: $http://developer.echonest.com/docs/v4/_static/AnalyzeDocumentation.pdf$

[4] L. L. J. Langford and A. L. Strehl. (2014, January) Vowpal wabbit (fast online learning). [Online]. Available: $http://hunch.net/\ vw/$

[5] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2009, vol. 2, no. 1.

[6] A. Agarwal, O. Chapelle, M. Dudík, and J. Langford, "A reliable effective terascale linear learning system," *arXiv preprint arXiv:1110.4198*, 2011.