Predicting Success for Musical Artists through Network and Quantitative Data

Suzanne Stathatos
sstat@stanford.edu
Stanford Univeristy

Zachary Yellin-Flaherty
zachyf@stanford.edu
Stanford University

I. Introduction

With the rise of Spotify, iTunes, and YouTube, ninety-nine cent songs have largely replaced $20 albums, slashing music sales by nearly fifty percent [1]. Investing in many artists is prohibitively risky today. It behooves music industry executives to leverage available metrics and machine learning techniques to predict whether an artist will be commercially successful in the future. Our goal is to predict whether artists will be successful based on available music industry metadata, namely artists' importance in the music industry and the public's response to their music. We collected annual measurements from 2000 to 2012 representing these features and calculated their change over time. We predict the success of an artist in 2013.

II. DataSets and Features

We generated a unique set of features indicating both public perception of musical artists and indicating their status in the music industry. We gathered some of these measurements from existing publicly available datasets (Discogs, The Whitburn Project, EchoNest, and LastFM). We pre-processed this data to understand each musical artist's annual level of interconnection in the music industry. In total we have 11,000 artists and 63 metrics. However, because our matrix was sparse, we downsampled significantly during testing and focused on 355 artists from years 2000-2013. The features we used are, for each year 2000-2012:

1) Degree in a graph of the music industry (GMI) (higher degree == more connections)
2) Eigenvector Centrality in the GMI
3) Betweenness of each artist in the GMI
4) Artist Success
5) PlayCount for artists' Billboard songs
6) Rates of Change for : PlayCount, Eigenvector Centrality, Degree, Success
7) Y-intercept for: PlayCount, Eigenvector Centrality, Degree, Success

Success in 2013 is our classification target. We will explain each of these features in more detail in the next subsections.

A. Features and Preprocessing

Our features split into three broad categories: Features from annual graphs of the music industry, features that represent an artists' annual public popularity, and features that represent the history (change over time) of each of these metrics.

1) Feature for Success: To define commercial success for an artist in a given year we collected data from the Whitburn projects' dataset [9]. This dataset contains Billboards annual song ratings from 1895 to 2013. These ratings list the top 200 songs for each year. We focused on years 2000 through 2013 to keep our predictions timely. We labeled a song as a hit if it was within the top half (0-100) of the annual hits. We also wanted to predict the success of an *artist*, not a song by that artist. To translate hit songs to successful artists, we summed how many successful songs each artist had in a given year. If an artist ranked within the top 50% of the rankings, he/she was labeled successful for that year.

2) Features from the Music Industry's Network: We generated a graph to represent and extract quantitative measures for collaborations in the music industry. Using data from discogs.com [8], we generated an undirected graph to model the collaboration network of the music industry, and calculated centrality metrics from this graph. Looking at all indexed releases dating back to 2000, there exists an edge between two artists in a given year if they appeared on the same record. We created one graph to represent each year. For each target artist in our feature set, we calculated the following.

Degree: Measures how prolific an artist is in a given year by summing the number of collaborations that artist had in that particular year.

Eigenvector Centrality: Recursively calculates the influence of a node in a graph. This metric is a centrality measurement on undirected graphs similar to Google's PageRank score [4]

Eigenvector centrality is defined to be: *Let A be an adjacency matrix for our graph. Using power iteration we solve for the largest eigenvalue λ_1 and corresponding eigenvector v*:

$$Av = \lambda_1 v$$

The eigenvector centrality score of node i is v_i, or the ith index of v.

Betweenness: The number of shortest paths that pass through a given node.

All of these metrics offer some quantitative representation of the importance of these artists in the annual collaboration network. We used the Stanford Snap Libraries [2] on large AWS machines to calculate these metrics. Each year's graph has on the order of one-hundred thousand nodes and one million edges.

3) Features for Artist's Annual Popularity: We collected metrics representing the public's perception of the artists and his/her music.

Play Counts: The billboard dataset gave us a list of successful (and unsuccessful) songs for each year. We queried LastFm's Database for each of these songs to find how often those songs were played online. After gathering how often each song was played, we summed the play counts of each song corresponding to each artist, and created a summary measurement of annual play counts for each artist.

Hottness: We gathered user review data from Echo Nest [10] for quantitative measurements that reflect an artist's popularity or "hottness" based on social media, user reviews, and mainstream media. These metrics could not be adequately queried by time, so this feature was used only in initial validation of our models.

4) Features for Change Over Time: To represent change over time, we calculated a linear change over time (slope and y-intercept) from each of our measurements where there is a point for each year from 2000 through 2012.

B. Verifying Feature Accuracy

We began by proving that we could verify an artist's success from metadata without time as a factor. We amalgamated the annual measurements for each artist (i.e. Beyonce: play counts in 2012, play counts in 2013,..., success in 2012, success in 2013,..., etc) into one overall measurement for that artist (Beyonce: play counts, hottness, success, etc). To maintain success as a classification variable (as opposed to a continuous variable), we labeled an artist as successful if the artist had ≥ 1 hit song in the 40 year time period. This smaller set had a total of 3573 artists (rows), each with 6 cumulative measurements (hottness, total plays, node degree, eigenvector centrality, degree centrality, node eccentricity, and success).

Our centrality measurements were based on Discogs snapshot of the music industry in mid-2013, rather than annual graphs that we used for our final results. These snapshots were estimates based on local subgraphs around target nodes. Starting with roughly 11,000 artists, we discovered some patterns in our data (i.e., artists are not relevant according the Billboard charts for some time period if they have not released any music), that indicated that we needed to increase the range of our timetable and reduce the corresponding sample size of interesting artists to about 350. This sample has a split of positive to negative examples: roughly 2/3 of artists in this set were labeled successful, 1/3 of artists were labeled unsuccessful. We maintained this split of artists when conducting our later tests to analyze our features over time.

Results from validation of features tests

Logistic Regression: Training logistic regression on 250 examples and testing on 100 yielded an average of 75% prediction accuracy. We found about the same number of false positives and false negatives.

Support Vector Machines: When 50 random samples were withheld from the data set (300 training samples, 50 testing samples) the classification accuracy of SVMs was 67%. Our initial tests were run with no cross validation and with no paramater adjustment, such that the we used the default kernel, cost functions, and paramters provided by LibSVM.

We wanted to analyze where SVM failed, and we wanted to determine if a different kernel and different set of parameters could strongly change the performance measurements of SVM.

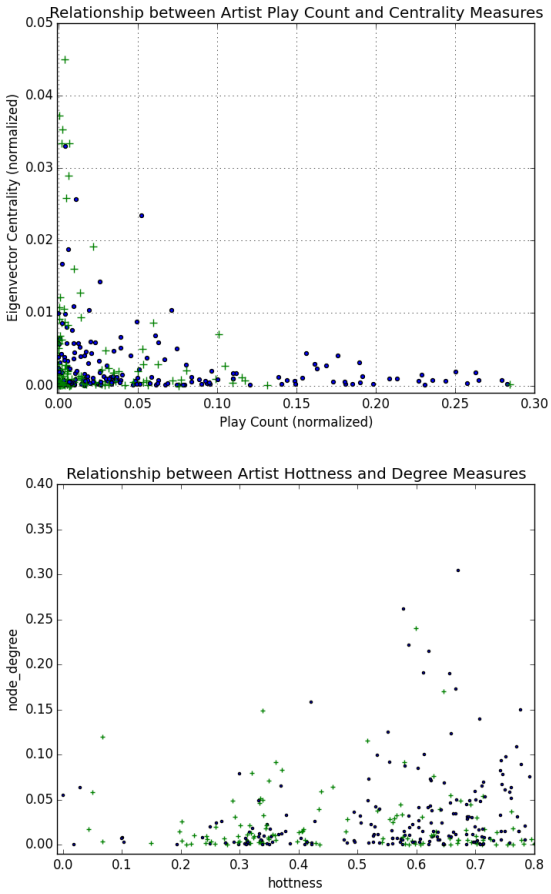

Fig. 1. The relationships between artists popularities, their connection in the music industry, and their degree of success is not obvious. Success in both of these graphs is represented by a green +, and failure is represented by a blue dot

We iterated through several combinations of parameters and used 10-fold cross-validation (instead of randomly splitting our data) to choose the best model–the model that gave the lowest generalization error– for our data. We found that a linear kernel yielded the lowest generalization error (three popular choices for kernels in the SVM literature are d-th-Degree polynomial kernels, radial basis function kernels, and neural network kernels. In this case, a linear kernel (1st-degree polynomial) outperformed the other kernels by a factor of 5-10%). This prompted us to switch from using LibSVM to using LibLinear SVM. Our matrix is sparse, liblinear optimizes for sparse matrices, and liblinear allowed us to experiment with choosing penalty and loss functions.

After implementing a linear kernel for Support Vector Classification, we found that the best tolerance to use was between 0.01 and 1. We experimented with this range of tolerance values when classifying future predictions as well.

In addition, we wanted to visualize the relationship between our artists' popularity and the artists' industry networks. These graphical relationships can be seen in Figure 1. It is interesting to note that, although a clear relationship is not depicted from the graphs, the learning algorithms did reasonably well at classifying success. This speaks well of our use of support vector machines, as they produce nonlinear boundaries by

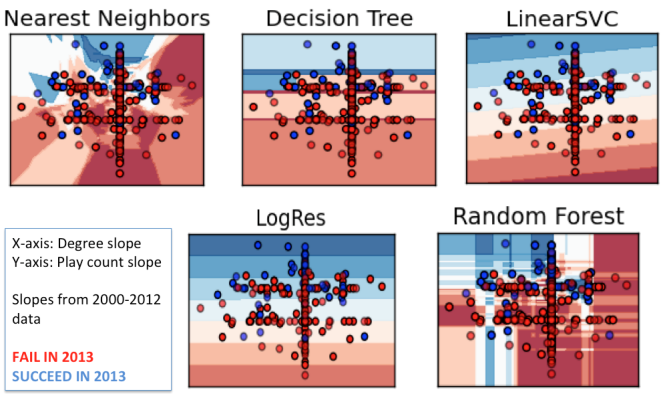

Fig. 2. The decision boundaries for each classifier. These graphs give a visualization of each of our classification models, graphing rate of change of each artists' degree to rate of change of each artists' number of annual play counts. Blue regions indicate success in 2013, while red regions indicate failure in 2013. The shadings represent probability boundaries. The lighter the shade, the lower the probability that it is on that side of the decision boundary.

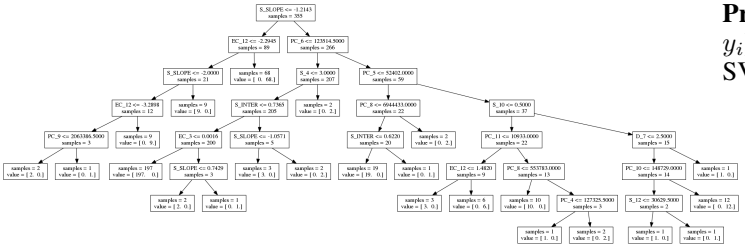

Fig. 3. The decision tree from which this classifier split our data. The tree chose the linear rate of change of success over the years as the most important classifying feature.

constructing a linear hyperplane in a larger-dimensional feature space.

III. Classification Models

We have continuous inputs and categorical outputs. Therefore, we began by running Support Vector Machines (SVM) and logistic regression (LR). These models allowed us to explore our explicit features (LR) and variations in other dimensions (SVM). After understanding the best parameters for each of these first two algorithms, we expanded to try other supervised learning algorithms, including Decision Tree Classifiers and Random Forest Classifiers. Lastly, we explored the K-Nearest Neighbors Classifiers.

A. Logistic Regression

For logistic regression, we employed matlab's $glmfit$, where the inputs were the feature matrix and the 0,1 success vector. Logistic regression has limited flexibility in terms of how to improve the model. We experimented with normalizing our features, though this proved to be unnecessary. Throughout the project we also decreased bias by adding more data. Our final model that achieved an accuracy of 81% had non-trivial weights on all features – in other words, in appears all features have some influence on the final prediction.

B. Support Vector Machine Classifiers

We used LibLinear SVM (rather than LibSVM) because we were manipulating a sparse data set. By doing so, we were able to experiment and find the best combination of cost functions and optimizations to minimize generalization error. We did not choose class weights, but instead, let the algorithm automatically adjust weights.

L2 losses (squared hinge loss) and L1 (hinge loss) penalties outperformed their respective counterparts (L1 losses/L2 penalties) by an accuracy level of roughly 20-30%. Optimizing for the primal rather than for the dual improved accuracies by roughly 15%.

Hinge Loss: The hinge loss function, $L(y, f)$ = [1 - yf]+ of support vector machines estimates the mode of the posterior class probabilities (other loss functions estimate 'a linear transformation of these probabilities', according to literature [5]). Squared hinge loss: gives a quadratic penalty to points on the outside of or far away from the support vectors. It gives a 0 penalty for points inside the margin of the support vectors.

Primal Optimization: Recall that given a training set $\{(x_i, y_i)\}$ where $1 \leq i \leq n$, $x_i \in R^d, y_i \in \{+1, -1\}$, the primal SVM optimization problem is

$$\min_{w,b} ||w||^2 + C \sum_{i=1}^{n} \xi_i^p \text{ where } y_i(w \cdot x_i + b) \geq 1 - \xi_i, \xi_i \geq 0,$$

where in our implementation, p represents the hinge loss squared.

The dual optimization problem can be written in terms of dot products, making it possible to implement the kernel trick. However, literature suggests that the primal optimization problem can be better when computing "approximate solutions" [6]. Optimizing for the primal with our SVM minimized our generalization error more so than optimizing for the dual. As a result, our model optimized for the primal rather than for the dual.

C. Decision Tree Classifiers [7]

Implementation: Trees are able to capture complex interaction structures in the feature set and graphically represent this structure. To understand the significance of each of our features (i.e. to see if play count from 2012 mattered more than play count from 2000...2011 for an artist when determining artist success in 2013), we wanted to classify artist success in 2013 using a classification tree. Classification trees work as follows:

We have p features and a measurement of success for N observations, such that we have (x_i, y_i) for i=1,2,...,N with x_i = $(x_{i1}, x_{i2}, ..., x_{ip})$. We let the algorithm automatically decide on the splitting variables, on the splitting points, and on the overall shape of the tree.

For classification, the algorithm finds the best binary partition to maximize the proportion of class k observations in a node m, given

$$\hat{p}_{mk} = 1/N_m \sum_{x_i \in R_m} I(y_i = k),$$

representing a classification region R_m with N_m observations. This expression represents the majority class (success or failure in 2013) in node m. Cross validation chose a model with a depth of 5 as the optimal tree size to accurately fit

the data. A graphical representation of the tree can be seen in Figure 3.

Drawbacks: Trees are noisy. While decision trees have low bias, they also have high variance. Often, a small change in the input data can result in very different splits of the tree. To handle this volatility, we used a more stable split criterion (we chose the 'best' split of the data, rather than a random split; we used the gini function to measure the quality of the split). We also implemented random forest classification (next section) to average the results of many trees and thereby reduce this variance.

Decision trees also lack smoothness along decision boundaries. We are looking for ways to address this issue.

D. Random Forest Classifiers

As mentioned in the previous section, Decision Trees experience a high degree of variance. Random forests decrease this variance through a process similar to bagging. Random forests build a collection of de-correlated trees and average their results together to eliminate noise from the trees. The number of trees and the means to split the features of the trees (auto) were chosen by iterating through different combinations of parameters and then running 10-fold cross validation on each of these models.

E. K-Nearest Neighbors Classifier

Because we learned about the k-means clustering algorithm in class, we were curious to observe how a the accuracy of a clustering classifier. In K-Nearest Neighbor, given a port x_j, we find the k training points $x_(r), r = 1, ..., k$ closest in distance to x_j. The point x_j is then classified using majority vote among the k neighbors. When votes are tied, x_i is randomly assigned to one of the neighbors in the tie. We use euclidean distance, $d_(i) = ||x_(i) - x_j||$, to measure distance to neighbors. We found that 5 was the optimal number of neighbors to require a point to be near. However, the lowest generalization error of these models was roughly 50%.

IV. Results and Discussion

Through extensive experiments, we were able to generate models to accurately predict an artists' overall success in 2013 given features from 2000 through 2012. Each algorithm underwent 10-fold cross validation to minimize the gap between training error and generalization error. A table containing each algorithms' predictive accuracies, along with which parameters reached these accuracies, can be seen in Figure 4. Our highest generalization accuracy from our Support Vector Machine classifier at 88%. Logistic Regression placed second with 81% accuracy. By analyzing our decision tree and the weights automatically calculated from logistic regression, we also determined that the slope of success for a musical artist is the best indicator for success of the musical artist in the future.

Cross Validation

Implementation: We used 10-fold cross validation to select our best model. We followed the same procedure as what is outlined in [11]. We retrained the models with the lowest generalization errors on the entire data set to generate resulting hypotheses functions, which are represented in the graphs of Figure 2.

Model	Parameters	Train Accur	Test Accur
LinSVM	'l1'penalty, 'l2'loss, 0.01 tol, 'dual'False	0.972	0.890(+/-0.02)
Log. Res.	-	0.946	0.811
Dec. Tree	'max depth':5	0.944	0.807 (+/-0.05)
Rand. For.	'max feats': 'auto'	0.971	0.791 (+/-0.06)
KNN	'neighbors': 5	0.914	0.492 (+/-0.06)

Fig. 4. Results from running cross-validation and model selection against classification algorithms using metadata from years 2000-2012 to predict artists success in 2013.

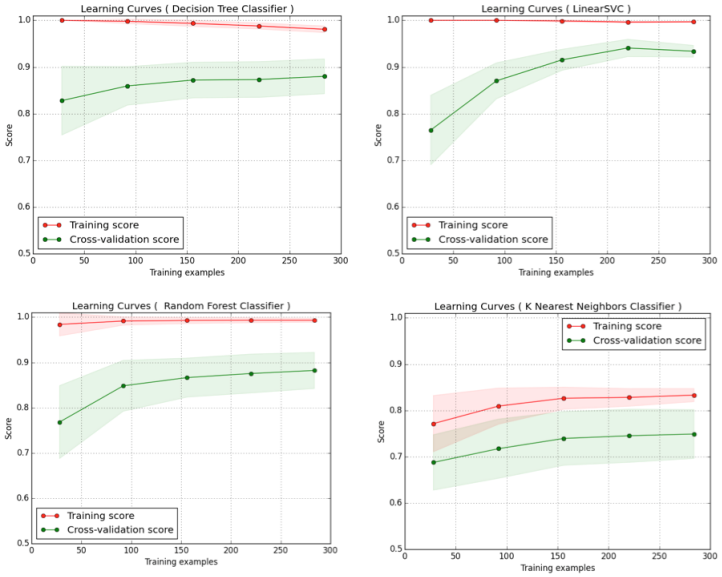

Fig. 5. Learning curves for each classifier, showing both the training accuracy and cross validation accuracy.

Evaluation: We chose K=10 to minimize variance and to run our computations quickly (a large K proved to be very computationally expensive). When K is too large (i.e. K=N), the cross validation estimator is unbiased for the expected prediction error, but can have high variance because the N 'training sets' are similar to each other. When K is too small (i.e. K=5), the cross validation estimator has low variance, but bias can be problematic. This bias can be seen in Figure 5's graphs, which show both the hypothetical learning curve (training accuracy) for each model along with the cross validation accuracy. The gap between a small training set's training and cross validation error is large. Thus, using a small training set would result in a considerable overestimate of prediction accuracy.

With a data set of 350 unique artists, 10-fold cross validation used training sets of size 315, which behaved similarly to the full set. Thus, 10-fold cross-validation did not suffer from much bias, and had reasonable variance.

Performance over time Figure 6 is a graph of the testing accuracies for each classification model over years 2007-2013

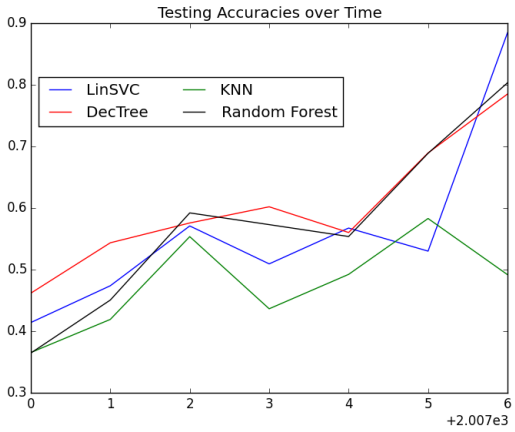

Fig. 6. Every tick along the x-axis represents a year's prediction accuracy, beginning from 2007 through 2013. The greater the number of years that are analyzed, the higher the predictive accuracy is for each classification algorithm. This implies that a longer amount of metadata (i.e. metadata for years 2000-2012 to predict 2013, rather than just metadata for 2012) is optimal. Though, it should be noted that more metadata inherently means that there are a larger number of features. We had 350 samples and 60 features, but if our feature set expands much larger, we would also need to collect and analyze a larger number of samples (artists).

(i.e. 2007 looks at features from 2000-2006 and predicts artist success in 2007; 2008 looks at features from 2000-2007 and predicts artist success in 2008, etc.). The graph indicates that as each model can observe a greater number of years for each artist, its predictive accuracy increases. This suggests that longevity plays a significant role in determining whether or not an artist is successful in future years.

Challenges: Graph generation, the choice of depth at which to traverse the graph, and calculation of pagerank and centrality metrics were difficult. The music industry's graph is very large (GBs of data), and proved difficult to manipulate graph efficiently. Betweenness calculations can have running times of $O(|V|^3)$. SNAP's pagerank and centrality algorithms do not complete in a matter of days on graphs with millions of nodes and edges. As a result, we read papers [3] exploring how to create subgraphs around target nodes to estimate centrality measurements. Only when we moved to annual graphs about an order of magnitude smaller than the original could we finish these jobs in a matter of hours.

V. Future Works

In our logistic regression tests, when we removed our slope-intercept features representing the best-fit change over time for our meta data features, our prediction accuracy fell to 75%. That is 6.0% lower than our regression results that included these features. All of these misclassifications were false negatives. This difference suggests that explicitly modelling the change over time of these features as a linear dynamical system or other time-dependent model can not only improve our accuracy, but also improve our understanding of underlying trends of commercially successful artists.

In this project, we successfully classified an artists' future commercial success based on the artists' connections in the music industry and based on the artists' popularity levels. Given that these metrics play a large role in determining the success, it would be advantageous to be able to predict these metrics as well. To predict these metrics, we would need to provide the algorithms a way to separate each of the features from each other. This could be done by having a summary metric (i.e. mean) for each feature to represent the previous years. This could be done using a multi-class classification models, such as multi-class SVM, Naive Bayes, or multi-class Linear Discriminant Analysis. We hope to look at this more in the future.

Furthermore, there are many more publicly available features that we can mine from social media and other less readily-available quantitative sources that would likely allow for additional insight into our results.

VI. Conclusion

The matlab and python scripts that we used to both collect and analyze the data are available at https://bitbucket.org/sstat/cs229_project.

We created our own definitions to answer the open-ended question of what defines commercial success of an artist. We customized our features to take advantage of data that is easily processable publicly. And our final feature matrix allowed us to test many of the algorithms from our class using out-of-the-box implementations. SVM and LR succeeded according to our problem definition. It is easy to see how these insights can generalize into solving industry problems where there are more stringent definitions of success and more data sources, especially in a field as entrenched and full of possibility as musical sales and perception.

References

[1] "NASDAQ AAPL Apple Inc. and data from the Recording Industry Association of America dataset", RIAA. https://www.riaa.com/keystatistics.php?content_selector=research-shipment-database-overview. Accessed October 2014.

[2] J. Leskovec. "SNAP For Python". http://snap.stanford.edu/snappy/ index.html. Accessed October, 2014.

[3] M. Hinne. "Location Approximation of Centrality Measures." Masters Thesis for Rabdoub University. January 2011.

[4] Y. Chen, Q. Gan and T. Suel. "Local Methods for Estimating PageRank Values" Conference on Information and Knowledge Management. November 2014.

[5] T. Hastie, R. Tibshirani, J. Friedman. "The Elements of Statistical Learning". Springer Series in Statistics. page 427.

[6] O. Chapelle. "Training a Support Vector Maachine in the Primal". MPI for Biological Cybernetics

[7] T. Hastie, R. Tibshirani, J. Friedman. Section 9.2 Tree-Based Methods. "The Elements of Statistical Learning". Springer Series in Statistics. pages 305-329.

[8] Discogs. Accessed October 2014. http://www.discogs.com/.

[9] The Whitburn Project. Accessed October 2014. Billboard Annual Charts Dataset received from the Comments Section in http://waxy.org/2008/05/the_whitburn_project/.

[10] EchoNest API. Accessed and used from October 2014 - November 2014. http://developer.echonest.com/docs/v4.

[11] A. Ng. CS229 Lecture Notes 5. http://cs229.stanford.edu/notes/cs229-notes5.pdf.