Gabe Wilson

CS 4641

11/29/19

Final Project Free Response

**Introduction**

Data science, machine learning, and artificial intelligence are all buzzwords that have piqued the interest of statisticians, analysts, and scientists alike. However, these topics that form the intersection of computer science and statistics are accessible to more than your Jane Street quant. For this very reason, I sought to build a kernel that would be useful to the general populace by taking vast amounts of quantitative information and boiling it down into an evidence-based prediction. In order to achieve this goal, I first needed to choose a relevant dataset. I chose a Kaggle csv file that stored the features of more than 240,000 songs from Spotify. This data included features such as genre, acousticness, instrumentalness, loudness, danceability etc. Danceability was the feature that stood out the most to me, so I wrote three algorithms that sought to predict the danceability of songs from the dataset. Given that danceability was recorded in the csv as a decimal value between 0 and 1, I implemented the regression algorithms that facilitated making a non-discrete prediction. The success of this kernel has real world applications that include recommending songs for dance parties and playlists and determining what features of a piece factor into making it a danceable song. To measure performance, I will use scikit-learns “score” method which returns the coefficient R^2. This coefficient is given by 1 - u/v, where u is the residual sum of squares ((y\_true - y\_pred) \*\* 2).sum() and v is the total sum of squares ((y\_true - y\_true.mean()) \*\* 2).sum().

**Description of the Algorithm**

1. *K Neighbors Regression*

The first algorithm I implemented to predict the danceability was K Neighbors Regression. Unlike K Neighbors Classification that aggregates the classes of the k nearest neighbors to classify a query point, K Neighbors Regression uses supervised learning to make a prediction for features with continuous values. In our case, this feature was the danceability. In order to understand the functionality of this algorithm let us simplify our dataset by graphing it on a 2d cartesian coordinate system. This reduction of the dataset calls for us to simplify each data point to only have 2 features for example, energy and instrumentalness. We take in k as a parameter that indicates how many points adjacent to the query point should be factored in to determine a value for our hypothesis class. We choose a distance method such as Euclidean or Manhattan distance to locate these adjacent points. Finally, we take the average of their associated danceability values and we use this as the danceability prediction of the query point. The following graphic illustrates the K Neighbors as described above:

A close up of a logo

Description automatically generated

The hyperparameters that were factored into making a determination of the hypothesis class include weights on the adjacent points, distance method, and k which represents the number of neighbors we will evaluate.

1. *Kernel Ridge Regression*

The second algorithm I implemented to predict the danceability was Kernel Ridge Regression. This algorithm employs least squares and regularization to learn a function given by a parameter called the kernel.

Tuning the hyperparameters proved to a crucial step of maximizing the accuracy of this algorithm. The hyperparameters factored into making a determination of the hypothesis class included the kernel, the degree (only used the polynomial kernel to define the degree of the polynomial), the alpha value used to reduce the variance of the estimates, and k which represents the number of neighbors we will evaluate.

1. *Neural Network*

The final algorithm I implemented to predict the danceability was a Neural Network. I used scikit learn’s MLP Regressor class that learns the function f(): Rm 🡪 Ro which maps m dimensional input to o dimensional output. This algorithm utilizes a network of neurons encapsulated in layers. The first layer is the input layer that includes all of the features. Between the output layer and the input layer are several layers known as hidden layers. In order to determine whether or not a nodes signal is passed on to deeper layers, we use the node’s activation function. This activation function is determined the input value and the weights that either increase or decreases the nodes chance of propagation.

**A close up of a logo

Description automatically generated**

This algorithm witnessed a large number of hyperparameters that helped tune the accuracy of the prediction. Some of these hyperparameters include the activation function, the learning rate, the hidden layer size, the maximum number of iterations, and the solver function.

**Tuning hyperparameters**

1. *K Neighbors Regression*
2. *Kernel Ridge Regression*

Determine ranges: Iterate through running the model on isolated parameters and observe the trends.

Cross Validation: Estimate the accuracy of the model after splitting the data in a variable number of ways.

Grid search: Choose the best set of parameters

*A screenshot of a cell phone

Description automatically generated*

1. *Neural Network*