Project Progress Report

SUNet ID: paulmtz
Names: Paul Martinez, Calvin Studebaker

Introduction

We are interested in the process of automatically generating music playlists given a set of songs. More specifically, we would like to train an algorithm that partitions a set of songs according to some criteria that it has been trained on. After some discussion of various approaches to the problem, we will consider the specific case where we are simply trying to partition a set of songs into two distinct sets. In this case we may phrase the problem as a sort of classification algorithm, but we will explore various other potential approaches along the way.

Model

Error Function

To begin we would like to formally define the problem and the objective function we are trying to maximize. The task we would like to accomplish is to be given a set of songs S and generate K playlists $P_1, P_2, \ldots, P_K$ from the set S. These sets $P_1, \ldots, P_K$ will form a partition of the set S. The goal is that this partition would be similar to how a human might divide the songs up into separate playlists. To do so, we first need to define an error function between a proposed partition and a given partiion. Suppose that we are given a set S and a "correct" partition $P = \{P_1, \ldots, P_K\}$ and our algorithm generates a partition $Q = \{Q_1, \ldots, Q_K\}$. We want to capture the difference between these sets, so we could count the number of elements that are in one set but not the other. For two sets A and B, this can be expressed as $|(A \cup B) - (A \cap B)|$. Since the sets are unordered, different comparisions may give different counts, so we should consider the lowest possible count. Let π be a permutation of $1, \ldots, K$. Then we can define an error function ε as follows:

$$\varepsilon(Q|P) = \min_{\pi} \sum_{i=1}^{K} |(P_i \cup Q_{\pi(i)}) - (P_i \cap Q_{\pi(i)})|$$

To give an example, let $S = \{1, \ldots, 9\}$ and $K = 3$. Then let:

$$P_1 = \{1, 2, 3\}$$
$$P_2 = \{4, 5, 6\}$$
$$P_3 = \{7, 8, 9\}$$
$$Q_1 = \{3, 4\}$$
$$Q_2 = \{1, 2, 6\}$$
$$Q_3 = \{5, 7, 8, 9\}$$

Then the optimal pairing compares P_1 with Q_2, P_2 with Q_1 and P_3 with Q_3: Then we have that $\varepsilon(Q|P) = 2 + 3 + 1 = 6$.

Generating Function

Now we turn to the question of how we might generate a partition of songs. For example, we could simply produce some feature vectors for each song and then run the K-means algorithm on the feature vectors and return those as the partitions. This would produce playlists where each playlist consists of "similar" songs, where the definition of similar is a function of the feature vectors. Alternatively consider an "inverted" K-means algorithm where we generate $|S|/K$ clusters, then create playlists by picking a random song from each cluster. This would create very diversifed playlists. So that we can actually train these algorithms, we can make them functions of w which is in the same dimension as the feature vectors. In a K-means example, this may designate the weights of the various coordinates when calculating distance. Without any explicit representation of these functions, it is impossible to give precise algorithms for improving their performance, but we can still pose it as a general search problem and present a very rough algorithm for finding better solutions. We can try to attempt a random local-search with hill-climbing. Let $F(S; w)$ be our partition generating function and $S_1, \ldots, S_m$ be a training set of sets of songs with given partitions $P_1, \ldots, P_m$. Then we could perform the following:

```
initialize w = 0
repeat i times or until convergence:
    pick c random values w_1, ..., w_c from S_ε(w) (the sphere of radius ε centered at w)
    let w_best = arg min_{w_k} Σ_{i=1}^m ε(F(S_i; w_k)|P_i)
    let w = ηw + (1 − η)w_best
```

One could then imagine that, if given a number of partition generating functions $F_1, F_2, \ldots$, we could simply train each of them on a training set, then pick the one that had the best overall training error.

Classification When K = 2

Let's consider the special case of when we are simply trying to classify