Opening up Pandora('s box)

Brian Magee and Prasanth Prahladan

December 5, 2015

1 Introduction

Audio/Music hosting applications like Pandora, Spotify, SoundCloud, Saavn etc, have the immense responsibility of being expected to serve as search-engines within the audio/sound-space. As a user, one expects that one's "taste" in music, is "understood"(learned) by the application, and that it shall then help the user explore and listen to artistes, tracks and albums, that would otherwise be difficult for the user, to individually search out for herself. The greater the search and exploratory experience, the better is the chance of the application being the favoured source of music-entertainment for its human-user-base.

Given this context, the course project helps give us a hands-on experience with building up a "music-genre classification engine". In the "music-genre classification" problem, we are required to determine the genre of a given query song, based on an algorithm that builds an internal representation of different music-genres, using a fixed database of classified(tagged) songs. The purpose of the project, is to enable the student to understand how to build a rudimentary functional prototype of a music-classification system. As a design challenge, the students are required to understand the high-level process of mining for patterns in a given music database,to split the process-flow into different sub-systems(algorithms) and determine how these functional-blocks may be combined to determine a process-flow for improving the efficiency of the classification problem. The specifications of the course-project as described in the project-guide, indicates the genres we need to focus on and the number of songs assigned to each genre.

The music-genre classification process can be summarized by a five-stage pipeline.

1. Embedding of Dataset into a Metric Space:
   It is important to note that the representation of data-files in memory, needs to be chosen based on the application for which that data shall be utilized. A given song when represented/stored as a .mp3 or .wav file, may be processed and played using an Audio player. However, for the purpose of classification, we need to design/develop a particular vector-representation that incorporates certain "descriptive features" of the audio file. These "features" may be based on analytical notions of time-frequency domain representation or based on "perceptive/cognitive" notions. The representation of these features, as mathematical vectors embedded in a high-dimensional space, then permits us to use algorithms for processing vectorial data, for the purpose of automated classification/clustering. We intend to use the "mel-frequency cepstral coefficients", its derivatives and possibly other "high-level-feature" information for each of the song files.

2. Visual Pattern Detection of Music Database:
   The preliminary step involves the use of a Data-visualization library, to determine whether there are any patterns like clusters in the data, which can then be exploited. It would be helpful to understand the logic/ideas behind the algorithms used to guide the visualization process, which could then be incorporated into the classification engine, that we need to develop.

3. Determine Intrinsic Dimension of the Data:
   If any patterns(clusters) have been detected in the data-visualization, it helps to assume that the different genres of high-dimensional data in fact is distributed along low-dimensional manifolds embedded within the high-dimensional space. We can then use algorithms for determining the intrinsic dimension of these manifolds (e.g "Correlation Dimension"). By determining the intrinsic-dimension of each of the genres, we can determine the minimum dimension of the space into which all the data can be embedded without losing much information.

4. Metric Learning:
   Ideally, when data are categorized into categories, we would like to believe that the songs/data-points that are "similar" to each other are "closer" to each other, and those that are "dissimilar" are "farther" apart. The closeness of two data-points is determined by the "distance metric" used to compute the distance between the two points. The reason such a "metric" would exist, can be deduced from the fact that the data points are distributed upon a Manifold. The measure of distance between any two points, thus becomes the distance travelled along the manifold surface between the points, rather than the shortest euclidean straight-line distance between them. The "metric-learning" problem deals with the problem of ascertaining the metric for each of the manifolds separately, or the notion of a "global-metric" for all the data-points.

5. Dimension Reduction:
   Once the minimal intrinsic dimension of the datasets has been determined, and we have a notion of "distance-metric" that helps compute the distances between the points, we then deal with the problem of reducing the dimension of the data. We note that this procedure, helps us determine how much of the information from the initial choice of vector representation of the dataset is redundant. The solution to this problem, is the design of a Map/function that projects every data-point from the high-dimensional space to the low-dimensional space. To ensure that the representation for each data-point is unique in the reduced dimension, we impose the constraint of an injective mapping, on this function.

6. Clustering:
   If we have identified patterns in the data-visualization stage, we would assume that the songs belonging to the same music-genre are distributed as clusters in the high-dimensional space. We expect that the dimension-reduction process does not hamper/destroy the clustering of the dataset, but rather reinforce/amplifies the extent of clustering - "similar" songs are brought closer together and "dissimilar" songs are pushed farther apart. It is obvious that the "distance-metric" used to measure the distances between songs in the reduced-dimensional space is very-different from that in the high-dimensional space. However, we can demand that the distance metric in the reduced-space

be closer to the Euclidean space. The "metric" in the reduced-space shall then be used to identify clusters in the dataset.

7. Query Processing (Cluster Assignment/Statistical Learning):
   This forms the final stage of the Genre-Detection problem. Once the abstract model - a spatial distribution/representation of the songs in the database has been determined, we have now developed the capacity to determine the genre/family that a particular song-query might belong to. We can develop a method to determine the genre-of song, by determining the cluster that it belongs to, the proximity to its neighbours and the classification of its neighbors into the different genre.

2 Feature vector representation

To derive content-based features representative of songs, we would have to implement digitial signal-processing algorithms upon the .wav song files, to compute quantitative information about the song. Some of the features that can be extracted from a song are mfcc, fluctuation patters, etc.

¡Explain the different possible features, in particular mfcc¿.

In this project, we focus on using the Spectral Information of the songs, to serve as the signature of each song. Songs are then classified into genre based on the similarity between these signatures.

2.1 Single Gaussian Model of each song

By adopting a Single-Gaussian Model representation of a song, one assumes that d-dimensional mfcc-coefficient vectors representative of each of the innumerable frames extracted from a song, can be obtained by sampling a multi-variate Gaussian in the d-dimensional song-mfcc-space. Thus, each song is represented by a Gaussian, with a mean-mfcc-coefficient vector and a covariance-mfcc-coefficient matrix that is obtained from the mfcc-coefficient vector representations of the frames extracted from the song.

Given, this representation of the song, the feature-vector used for computational purposes, could either assume the songs to be objects between which the distances are measured, by specialized mathematical schemes like KL-divergence, Earth-Movers-Distance, etc which are methods for computing distances between any two distributions. Or, as we have implemented, a custom-feature vector can be derived by concatenating the mean-mfcc vector($[1xd]$ dimensions) and the upper-triangular part of the mfcc-covariance-matrix, to form an extended feature-vector ($[1x(d + d(d + 1)/2)]$ dimensions).

2.2 Code-word Histogram

The code-word histogram [4] approach to feature-vector representation of a song can be described intuitively as follows. All the music files in the database can be assumed to be 'sentences' that are made up of a fixed number of 'code-words' in the vocabulary of music. Each song is represented by the number of times specific 'code-words' occur in it. The word histogram is used to describe this method of counting the number of times a given code-word occurs in the song. Thus, by assuming such a finite word vocabulary to describe all possible songs in the database, we enable a mechanism of drawing parallels between comparison of textual-documents via Natural-Language processing algorithms, and the analysis of music.

Mathematically, the code-word histogram generation process can be described as follows. To obtain a compact summary of audio signals each song is represented as a histogram over a dictionary of timbral codewords. As a first step, a codebook is constructed by clustering a large collection of feature-descriptors. Once the codebook has been constructed, each song is summarized by aggregating vector-quantization representations across all frames in the song, resulting in codeword histograms. Finally, histograms are represented in a nonlinear kernel space to facilitate better learning of the distance metric between the songs.

2.2.1 Codebook training

The primary audio feature descriptor used for training of the codebook, is the mfcc coefficients of frames of a song. The audio-feature descriptors of all the songs in the database is aggregated into a single bag-of-features, which is then clustered to produce the codebook. In our implementation, the number of words in the codeword dictionary is arbitrarily chosen to be about 512.

For each song x in the codebook training set X_C, we compute the 13 MelFrequency cepstral coefficients(mfcc) from each overlapping 25-ms frame. From the time-series of the MFCC vectors we compute the first and second instantaneous derivatives, which are concatanated to form a sequence of $13x2$ dimensional first-order Dynamic-MFCC(FD-DYN-MFCC) and 13x3 dimensional Dynamic-MFCC(DYN-MFCC). These descriptors are then aggregated across all $x \in X_C$ to form an unordered bag of features Z, where each $z \in Z \subset \mathcal{R}^D$ is either an MFCC, FD-DYN-MFCC or DYN-MFCC feature-vector representation of the song-frames.

To correct for changes in scale across different dimensions of $z \in Z$, each vector is normalized according to the sample mean $\mu \in \mathcal{R}^D$ and standard deviation $\sigma \in \mathcal{R}^D$ estimated from Z. The i'th coordinte is mapped by

$$z[i] \mapsto \frac{z[i] - \mu[i]}{\sigma[i]}. \tag{1}$$

The normalized feature vectors are then clustered into a set $\mathcal{V}$ of $|V| = 512$ codewords by k-means algorithm.

2.2.2 (Top τ) Vector Quantization

Once the codebook $\mathcal{V}$ has been constructed, a song $x \in X_C$, is represented as a histogram h_x over the codewords in the codebook. Each song $x \in X_C$ is understood to be a time-sequence of feature-vectors, $z = z_i \in Z \subset \mathcal{R}^D$ where z_i is the feature-vector representation of a frame in the song. Each $z_i \in z$ is normalized according to (1). The codeword histogram of song $x \in X_C$ is constructed by counting the frequency with which each codeword $v \in \mathcal{V}$ quantizes the elements of z i.e

$$h_x[v] = \frac{1}{|z|} \sum_{z_i \in z} \frac{1}{\tau} 1 \left\{ v = arg \overset{\tau}{\min_{u \in \mathcal{V}}} ||z_i - u|| \right\}. \tag{2}$$

where we have chosen to adopt multiple codeword quantizers of each vector z_i by defining the quantization set

$$arg \overset{\tau}{\min_{u \in \mathcal{V}}} = u \text{ is a } \tau \text{nearest neighbor of } z_i. \tag{3}$$

where $\tau \in 1, 2, \cdots, |V|$. Note that the codeword histograms are normalized by the number of frames $|z|$ in the song in order to ensure comparability between songs of different lengths. Further, the normalization by $1/\tau$ ensures that $\sum_v h_x^\tau[v] = 1$, so that for $\tau > 1$, h_x^τ retains its interpretation as a multinomial distribution over $\mathcal{V}$.

2.3 Elias-Pampalk's Choice of Features

The music-analysis toolbox that was provided as a part of the initial project-dataset contained a host of functions that were used by Elias Pamapalk's submission for the MIREX'2004 competition. He had chosen to represent each song by certain high level features called 'fluctuation patters' and also, a single-gaussian model for the frames of mfcc coefficients that were extracted from each song. We chose to use the same set of feature-vector representation and distance/similarity computation measures, as we believed that it would serve as an effective base-line for comparing the utility of alternative music-genre classification that pipelines we develop as a part of the course.

The distance measure was computed using the formula:

$$\begin{aligned} D_{ij} = {} & 0.7 * (-exp(-1/450 * d_{mfcc}) + 0.7950)/0.1493 + 0.1 * (d_{fp} - 1688.4)/878.23 \\ & + 0.1 * (d_{fpg} - 1.2745)/1.1245 + 0.1 * (d_{fpb} - 1064.8)/932.79 + 10; \end{aligned}$$

We note that the distance measures from different feature-vector representations of the dataset, were combined to obtain a single distance measure by means of numerous parameters that were obtained from other experimental techniques, that the author mentions in his thesis.

3 Intrinsic Dimension

It is assumed that the datasset associated with the different genres, can be classified into distinctively separate nodes-spaces of the abstract high-dimensional euclidean space within which each data point exists. The algorithm presented in [5] shall be implemented for the given dataset. It needs to determined, if the formulation of intrinsic dimension of the dataset is affected by the choice of the "metric" describing the distance between the nodes. The computation of the intrinsic dimension of each of the genres, shall help us determine the minimal dimension of the reduced-dimensional space within which all the data-points can be suitably represented without any significant loss of information.

4 Distance in song-space

The choice of the Distance-Metric depends on the chosen Feature-Vector-representation for each song. The naive method to proceed would be to assume that all the songs are embedded in an Euclidean Space, and hence the distances can be computed using the L-2 Norm.

However, for the specialized representations of the songs that we have adopted, we believe that the songs are distributed over a manifold, and hence the distance between the points need to be computed by using a metric intrinsic to that manifold. To explore this idea, we have decided to implement the Information Theoretic Metric Learning toolbox [1], for determining the appropriate "metric" that helps classify and categorize the music genres.

4.1 Information Theoretic Metric Learning

The information theoretic metric learning [1] approach to learning the distance function, is a particular technique for computing the Mahalanobis distance between two points in a metric-space. The Mahalanobis distance generalizes the standard Euclidean distance by admitting arbitrary linear scalings and rotations of the feature space. The problem of learning an optimial distance-metric is translated to learning the optimal Gaussian with respect to an entropic objective function. The model adopted leverages the relationship between the multivariate Gaussian distribution and the set of Mahalanobis distances.

Given a set of points $x_i \subset \mathcal{R}^n$, and a distance metric A, the distance measure between the points is computed as

$$d_A(x_i, x_j) = (x_i - x_j)^T A(x_i - x_j)$$

Consider (dis)similarity relationships between points provided by information regarding the points belonging to the same(or different) genres. This may also be interpreted as, similar points have their Mahalanobis distance less than a given upper-bound ($d_A(x_i, x_j) \leq u$), and dissimilar points have their distances greater than a lower-bound ($d_A(x_i, x_j) \geq l$), for sufficiently large l.

Assume that we have two different metrics, A and A_0. We can quantify the closeness between A and A_0 via a natural information-theoretic approach. There exists a simple bijection between the set of Mahalanobis distances and the set of Gaussian Multivariate distributions. Given a Mahalanobis distance characterized by A, we express its corresponding multivariate Gaussian as $p(x; A) = \frac{1}{Z} exp(-\frac{1}{2} d_A(x, \mu))$, where Z is a normalizing constant and $A^(-1)$ is the covariance of the gaussian distribution about the mean μ. Using this bijection, we measure the distance between two Mahalanobis distance functions parametrized by A_0 and A by the differential relative entropy between their corresponding multivariate Gaussians:

$$KL(A_0||A) = \int p(x; A_0) log(\frac{p(x; A_0)}{p(x; A)}) dx \tag{4}$$

The distance provides a well-founded measure of closeness between two mahalanobis distance functions and forms the basis of the optimization problem given below:

Given the pair of similar points S and set of pairs of dissimilar points D, the distance-metric learning problem is

$$\begin{array}{lll} min_A & KL(p(x; A_0)||p(x; A)) & \\ \text{subject to} & d_A(x_i, x_j) \leq u & (i, j) \in S \\ & d_A(x_i, x_j) \geq l & (i, j) \in D. \end{array} \tag{5}$$

The above optimization problem is solved as a log-determinant optimization problem. The details of the algorithm used and the implementation of the algorithm can be found in the original research paper referenced below [1].

5 Similarity between songs in song-space

Given a measure of distances between songs, as computed by the computed/assumed distance metrics, we also have information regarding the genre-id's of each song. We expect that songs of a same genres are more similar than songs of dissimilar genres. The similarities between songs can be quantitatively represented as a matrix containing values which are a function of the distances between the points.

When using only the distance information to describe similarity, we can choose the function to be any number of non-linear functions like:

1. Gaussian Similarity $s_{ij} = exp(-d_{ij}^2/2\sigma)$,
2. Inverse distance $s_{ij} = \frac{1}{1+d_{ij}^p}$, where $p \geq 1$,etc.

We use the Similarity Matrices computed above, as inputs to some of the nonlinear dimension reduction techniques like the Refined-Graph-Embedding.

6 Data Visualization

The t-SNE algorithm [2] is an award-winning algorithm, for visualizing high-dimensional datasets. We shall first implement this algorithm, to determine a possible visual representation of the database, that shall indicate visual patterns in the data. This visualization shall help us develop an intuition of what the data distributions might look like.

This algorithm provides us the opportunity to check the efficiency of the vector-space embedding of the selected features from the songs in our database. If the visualization does not separate our vector-space data set into a sufficient number of discrete clusters, then it is likely that our choice of features is inadequate to truly separate the different genres when implementing our own dimension reduction techniques. In this case we will augment the vector-space embedding by adding additional features from the songs. This visualization algorithm can also be used after each stage of the pipeline to check that the information in our dataset has not been adversely deformed. Further, this algorithm can also be used a dimension-reduction technique, and shall be described in the appropriate section below.

7 Dimension reduction

We shall create and compare multiple pipelines for the dimensional reduction process. Both linear and non-linear graph based methods shall be explored. An interesting application we intend to explore, is the extension of the work on learning multimodal similarity [3] to determining similarity between genres. The process would thus adopt a soft-categorization procedure, rather than assuming the clusters to be shaped as hard-bounds.

Linear method using Fast Johnson-Lindenstrauss transform (FJLT) algorithm. This method uses random projections to map our data from the initial high d-dimensional space to a much lower k-dimension space by allowing low distortions of the data which is controlled with a tolerance parameter (epsilon). The choice of ϵ impacts the choice of $k-$dimensional space into which the data can be suitably embedded.

Non-linear, Graph-based Refined Embedding. This method builds a similarity graph, between all the datapoints, where the edge-weights are computed using a particular kernel function and distance-metric. From the similarity graph, we derive the Graph-Laplacian and follow the procedure of spectral embedding of this graph. The eigenvalues and the eigenvectors of the Graph Laplacian shall indicate the existence of the clusters.

7.1 Fast JL Transform

7.2 Refined Graph Embedding

7.3 t-Distributed Stochastic Neighbour Embedding

The t-SNE algorithm [2], was introduced earlier as a technique for visualization of high-dimensional datasets, to built a visual intuition of the data-distribution. From the fact that local neighbourhood information of the data is preserved by the technique, it would serve as a useful dimensional reduction technique, and hence used this method as an alternative to the Graph Embedding Technique.

tSNE starts by converting the high-dimensional Euclidean distances between data points into conditional probabilities that represent similarities. The similarity of data-point x_j to x_i is the conditional probability $p_{j|i}$ that x_i would pick x_j as a neighbour, if the neighbours were picked in proportion to their probability density under a Gaussian centred at x_i. Mathematically, the conditional probability is computed as

$$p_{j|i} = \frac{exp(-d_{ij}^2/2\sigma_i^2)}{\sum_{k\neq i} exp(-d_{ik}^2/2\sigma_i^2)} \tag{6}$$

where σ_i is the variance of the Gaussian centred about x_i. Because, we are only interested in modelling pairwise similarities, we set the value of $p_{i|i} = 0$.

Now, we assume that there's a particular projection operation that projects each of the songs x_i to lower dimensional point y_i. For the low dimensional counterparts y_i and y_j of the high-dimensional points x_i and x_j, we can compute the conditional probability, $q_{j|i}$ as

$$q_{j|i} = \frac{(1+d_{ij}^2)^(-1)}{\sum_{k\neq i}(1+d_{ik}^2)^(-1)} \tag{7}$$

where, we use a t-Student distribution with a single degree of freedom, instead of a Gaussian distribution about point y_i. This is because, it has the particularly nice property that $(1 + d_{ij}^2)^(-1)$ approaches an inverse-square law for large pairwise distances in the low-dimensional map. This makes the maps representation of the joint probabilities almost invariant to changes in scale of the map for points that are far apart. Which also implies that large clusters of points that are far apart interact in just the same way as individual points, so the optimization operates in the same way at all but the finest scales.

If the projections correctly model the similarity of the points in the high dimensional space, then the conditional probabilities $p_{j|i} = q_{j|i}$. Motivated by this intuition, the tSNE algorithm aims to find a low-dimensional data representation that tries to minimize the mismatch between joint conditional probabilities P and Q. A natural measure of the distance between two distributions is the Kulback Liebler-divergence. tSNE minimizes the KL-divergence between P and Q, via a gradient descent algorithm. The cost function is given by

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log(\frac{p_{j|i}}{q_{j|i}}) \tag{8}$$

where it is assumed that $\sigma_i = \sigma$ for all data-points.

The gradient of the Kullback-Liebler divergence between P and the t-Student based joint probability distirbution Q is given by

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + d_{ij}^2)^(- 1) \tag{9}$$

where $d_{ij} = ||y_i - y_j||$

8 Clustering and Statistical Learning

It is assumed that in the reduced dimension space, the euclidean metric shall serve as a sufficient metric to compare the different genres of songs. We intend to use the libraries available in Python Scikit-learn [?] to perform clustering of the reduced-space data.

In particular we intend to investigate the DBSCAN clustering algorithm as its features appear suited to our genre classification problem. In particular the notion of diagnosing the individual data points as "core" or "non-core" nodes belonging to the cluster can be used in conjunction with kNN for cluster membership detection by weighting neighbors differently for each case. Also the diagnosis of nodes as outliers not belonging to any cluster may also be helpful in "weeding out" data points for which comparisons are not particularly useful. Additionally, the fact that DBSCAN is agnostic to the shape characteristics of individual clusters is likely to be helpful.

Every query song whose genre needs to be determined, shall undergo the same process of dimension reduction, used to determine the clusters in the songs. The probability of belonging to each of the different clusters is determined by a process of voting by the nearest neighbors, or neighbors-of-neighbors. We shall not be evaluating the performance of the cluster-membership-probability of the different voting schemes individually, but its contribution to the whole genre-classification pipeline.

8.1 Implementation

The embedding of data-points obtained by the Dimensional reduction techniques is expected to have the points arranged such that clusters can be detected by clustering algorithms. However, since we have the genres associated with each of the songs, we do not implement any unsupervised clustering algorithms. Given a query song, we identify the genre it belongs to by a voting-mechanism implemented by the neighbours of the query-point in the reduced dimensional space. This methodology shall be described in the section on Statistical Learning.

We implement a k-nearest neighbors algorithm to detect a set of neighbors and their neighbors, for a given query-point embedded into a reduced-dimension projection space, containing a population of all songs in the database. The unique set of neighbours are identified, and a voting scheme is implemented. Each neighbor assosciates it own genre upon the query-song.

9 Putting them together

We expect the different genres are comprehensive, in the sense that all songs that do not get classified into either of the specialized genres get assigned to the category "World". Thus, it is possible that even white-noise could be classified as "World" music. Therefore, it is important to build a binary classifier that distinguishes between World and all the other families. We hope the method of Support Vector Machines can be used to discriminate between the "world" and "non-world" categories.

This is followed by a pipeline of algorithms that implement global dimension reduction, clustering algorithm, and statistical learning, to determine the probability of cluster membership of the query data.

10 Testing and Verifying the pipelines

The different pipelines are evaluated in the following manner.

The feature vectors of concern is extracted for every song in the database. The distance metric used to computed the distance between the feature vectors is either assumed to be Euclidean, or is learned by using the 'Information Theoretic Metric Learning' [1] toolbox. Using the distance metric and the 'true' genre assignments (from the truth file) we use different dimensional reduction techniques to embed the data points into a smaller m-dimensional space. As a default value, for a preliminary run of the experiments, we have chosen $m = 3$. However, further experiments are required to determine the clustering accuracy on increasing the dimensions. One of the interesting experiments we identified, was to determine the intrinsic-dimensionality of the different genres, under different choice of feature-vector representation of the songs. This would give us some information on how to upper-bound the dimension(m) of the feature-space for the embedding procedure.

After computing the embedding of the songs in the reduced dimensional space, we adopt an iterative procedure of cross-validation of our classification scheme. At every iteration, the entire music database is randomly split into two segments - the training(80%) and testing dataset(20%), by generating proportional partitions within each genre. To be precise, for every genre, the set of all songs belong to it is partitioned into two subsets - training-set(80%) and testing-set(20%). The global-training-set(testing) is obtained by coalescing all the training-sets(testing) of each genre into one set. For each song, within the global-training-set, we determine its cluster-membership, by seeking out its neighbours from the global-training-set, and implementing a voting scheme. The results of the cluster-membership-assignment for the testing-set is compiled by creating a confusion-matrix.

The confusion-matrix is obtained by determining the percentage of songs of a given genre, that have been classified as belonging to each of the unique genre-types. This is possible, as the information regarding the true-categorization of the testing-data is known to us. The mean and standard-deviation of the confusion-matrices obtained over all iterations, for a given genre-classification-pipeline, are computed and archived. The effectiveness of each pipeline is determined by comparing these statistics of the confusion matrices.

11 Discussion

The dimensional-reduction stage of the algorithm pipeline, shall form the core component of the system. It is possible to generate myriad features from each song and thus embed the dataset into a very high dimensional space. However, the choice of the dimensional-reduction algorithm shall determine the extent of geometrical distortion, and loss of information, during the projection of points from the high-dimensional space to the reduced-dimensional space.

Each dimension reduction algorithm may have a limitation on its efficacy in reducing the dimension of the data by a particular order. We can determine this only by experimentation, for each of the alternative pipelines.

As previously mentioned, the FJLT method reduces the dimension in relation to the distortion allowance. In order to reduce to a workable dimension, excessive distortion allowance may be necessary which might lead to significant loss of information.

Though, we have described different alternatives for each stage of the data-processing pipeline, it is important to identify which permutation of the combination of algorithm sequences shall provide the best performance. These algorithms might be evaluated in isolation, and also in cascaded combinations with one another. However, it is difficult to predict which combinations shall work better and why. The better pipeline can be determined only through experimentation and a comparison of the confusion matrices.

References

[1] *Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra and Inderjit S. Dhillon.* Information Theoretic Metric Learning, ICML June 2007, 209-216, http://www.cs.utexas.edu/ pjain/itml/

[2] *L.J.P. van der Maaten and G.E. Hinton.* Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(Nov):2579-2605, 2008, http://lvdmaaten.github.io/tsne/

[3] *McFee, B. and Lanckriet, G.R.G.*Learning multi-modal similarity, Journal of Machine Learning Research (JMLR) 2011.

[4] *McFee, B., Barrington, L., and Lanckriet, G.R.G.*Learning content similarity for music recommendation, IEEE Transactions on Audio, Speech and Language Processing, 2012.

[5] *M. Hein, J.-Y. Audibert.* Intrinsic dimensionality estimation of submanifolds in Euclidean space, In L. de Raedt and S. Wrobel, editors, Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), 289 - 296, ACM press, 2005, http://www.ml.uni-saarland.de/publications.htm