PATTERN CLASSIFICATION FORMULATED AS A MISSING DATA TASK: THE AUDIO GENRE CLASSIFICATION CASE

Aggelos Pikrakis[1] *Yannis Kopsinis*[2] *Symeon Chouvardas*[3] *Sergios Theodoridis*[2]

[1] University of Piraeus,
Dept. of Informatics,
Piraeus 18534, Greece.
pikrakis@unipi.gr

[2]University of Athens,
Dept. of Informatics and
Telecommunications,
Athens 15784, Greece.
kopsinis@ieee.org, stheodor@di.uoa.gr

[3]Mathematical and Algorithmic Sciences Lab,
Huawei France R&D,
Paris, France.
symeon.chouvardas@huawei.com

ABSTRACT

This paper presents pattern classification to a predefined set of classes as a missing data task. This is achieved by first augmenting the feature vector of each training pattern with the corresponding binary codeword representing its class. A Restricted Boltzmann Machine (RBM) or a Dictionary Learning (DL) algorithm is then trained on the augmented feature space. During the classification stage, the binary codeword of the unknown pattern is treated as missing data. In the case of the RBM, it is filled in by means of an alternating Gibbs sampling procedure. In the case of the DL method, the set of atoms in the dictionary is first learned from the training data, and the label of the unknown pattern is predicted based on those atoms that represent this pattern. Application of the method in an audio genre classification task verifies that the obtained results are highly competitive compared with state-of-the-art methods. Moreover, the DL approach lends itself readily for online implementations, in line with the current trend in big data applications.

Index Terms— Classification, Missing Data, Restricted Boltzmann Machine, Dictionary Learning, Audio Genres

1. INTRODUCTION

Over the past few years, variations of the Restricted Boltzmann Machine (RBM) have been widely used, mainly in the deep learning context, to solve a variety of tasks, including pattern classification, de-noising, auto-encoding and missing data inference [1], with the latter being of particular interest in this paper. Another emerging trend, which has also been employed in data reconstruction tasks, has evolved around Dictionary Learning (DL), which is conventionally related to sparse representation / coding problems [2], [3].

In this paper, inspired by the inference / reconstruction properties of the RBMs and the DL methodologies, we propose a method to view the standard pattern classification task as a missing data problem, where the missing data are the class labels of the unknown patterns in an augmented feature space. In the case of the RBM, the missing data reconstruction is the result of a Gibbs sampling procedure, and in the DL context it stems from the combination of a small number of atoms from a trained dictionary.

The proposed approach, as a concept, departs from the rationale of conventional classification techniques, since it does not rely on explicitly training one or more classifiers. On the contrary, the class label information is embedded in the training data themselves allowing for the RBM or any DL method to be employed and learn how to reconstruct the data of the classes involved. The proposed point of view enjoys some notable characteristics. The multiclass scenarios are seamlessly accommodated. Also, the trained RBM can still serve as a pattern generator, [4], which is a desirable property in many applications. Furthermore, online DL techniques, e.g. [5] can be used out of the box, paving the way to online classification and to a wide range of potential applications. For example, very large-scale datasets or streaming data can be naturally accommodated on the fly without excessive storage requirements. Online methods bypass the need for retraining from scratch whenever new data become available. Moreover, in principle, can deal with cases where the number of classes is not a-priori fixed.

The proposed formulation is generic in the sense that it can be used to design classifiers for a variety of tasks. As a proof of concept, we present an application on audio genre classification, focusing on the publicly available Ballroom Dataset, which defines a hard multiclass scenario, for which various methods have been designed in the last decade. These methods have used a variety of classifiers, ranging from the k-NN classifier to SVMs and deep networks [6], operating on rhythmic features [6–9], timbral features [10], and their combinations [11–16]. The best classification accuracy for this dataset has been reported in [9] and [6]. We show that our approach permits the design of a family of classifiers which exhibit competitive performance and demonstrate that audio genre classification can be also treated in an online learning context, while preserving high classification accuracy. Online learning techniques are gaining a lot of interest in the context of Big Data applications.

The paper is organized as follows: Section 2 presents the new problem formulation. Sections 3 and 4 presents the label reconstruction methods based on the RBM and the Dictionary learning methodologies, respectively. The experimental setup, the classification performance of the proposed methods and a comparative study with reported results are given in Section 5. Finally, conclusions are drawn in Section 6.

2. FORMULATION OF THE MISSING DATA TASK

Assuming a multiclass problem involving M classes, $\omega_i, i = 1, 2, \ldots, M$, and let $\mathbf{x}^T = [x_1, x_2, \ldots, x_l]$ denote the l-dimensional feature vector representing a pattern, and $\mathbf{y}^T = [y_1, y_2, \ldots, y_M]$ the respective class label, where $y_i = 1$ if $\mathbf{x} \in \omega_i$ and $y_i = 0$, otherwise. In other words, each label is a codeword, in which the position of 1 indicates the class to which the respective pattern belongs. Instead of a zero, we can alternatively use a -1. In principle, more advanced coding schemes can be employed that can allow higher immunity to noise [17]. Furthermore, let

Symeon Chouvardas was with the University of Athens when this research was conducted.

$X = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$ denote the training set consisting of N patterns.

The key idea is to augment the feature vector, $\mathbf{x}$, with the respective class label coding vector, $\mathbf{y}$, and form the augmented pattern $\bar{\mathbf{x}}$, such that $\bar{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$. Therefore, in the augmented feature space, the training set, X, can now be written as $X = \{\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \ldots, \bar{\mathbf{x}}_N\}$. In other words, the information that is carried by the class label is now "embedded" in the feature vector.

The next step is to feed the training dataset to a learning algorithm. In this paper, we experiment independently, in the *extended space*, with RBMs and DL algorithms for the training stage. Note that, albeit the two approaches look different, at the heart of both methods lies our desire to model the input patterns in a compact and data-representative way. During the classification stage, given an unknown pattern, the trained engine reconstructs the augmented feature vector elements, that correspond to the missing label by means of an appropriate algorithm (that depends on which learning scheme is used). We first present the RBM training scenario and the corresponding inference stages and Section 4 proceeds with the description of the respective DL-based approach.

3. RBM INFERENCE OF THE CLASS LABELS

The standard RBM is a network of binary, stochastic units, which are organized in two layers, known as the "visible" and "hidden" layers. To preserve the notation that is frequently used in the RBM literature, let $\mathbf{v} = [v_1, v_2, \ldots, v_{l+M}]$ be the visible units and $\mathbf{h} = [h_1, h_2, \ldots, h_K]$ the hidden units. Each visible unit corresponds to a feature dimension and it is connected with all hidden units with undirected weights (and vice versa) but there do not exist connections between units of the same layer.

3.1. Binary visible and hidden units

Let w_{ij} be the weight connecting v_i with h_j, a_i the bias associated with v_i and b_j the bias of node h_j. The conditional probability, $p(h_j = 1 \mid \mathbf{v})$, of turning "on" a hidden unit is

$$p(h_j = 1 \mid \mathbf{v}) = \sigma(b_j + \sum_{i=1}^{l+M} v_i w_{ij}) \tag{1}$$

where $\sigma(x)$ is the logistic function, $\sigma(x) = \frac{1}{1+exp(-x)}$. Similarly, the conditional probability, $p(v_i = 1 \mid \mathbf{h})$, of turning "on" a visible unit is

$$p(v_i = 1 \mid \mathbf{h}) = \sigma(a_i + \sum_{j=1}^{K} h_j w_{ij}) \tag{2}$$

The probability, $p(\mathbf{v}, \mathbf{h})$, of a joint configuration, $(\mathbf{v}, \mathbf{h})$, of visible and hidden nodes, is $p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v}, \mathbf{h})}$, where $E(\mathbf{v}, \mathbf{h})$ is the energy of the configuration,

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{l+M} \alpha_i v_i - \sum_{j=1}^{K} \beta_j h_j - \sum_{i=1}^{l+M} \sum_{j=1}^{K} v_i h_j w_{ij} \tag{3}$$

and Z is the partition function, $Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$.

Therefore, the probability, $p(\mathbf{v})$, of a visible vector, is computed by summing over all possible hidden vectors, $p(\mathbf{v}) = \sum_{\mathbf{h}} p(\mathbf{v}, \mathbf{h})$.

The training goal is to compute the weights that maximize, from a maximum likelihood perspective, the log-likelihood, $L(X)$, of the training set, $L(X) = \sum_{i=1}^{N} \log p(\bar{\mathbf{x}}_i)$.

To this end, each training pattern is "clamped" on the visible nodes of the RBM and an approximation of the log-likelihood gradient, known as Contrastive Divergence (CD) [18],[19] is used to update the weights and biases of the RBM as follows:

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon}) \tag{4}$$

$$\Delta b_j = \epsilon(\langle h_j \rangle_{data} - \langle h_j \rangle_{recon}) \tag{5}$$

$$\Delta a_i = \epsilon(\langle v_i \rangle_{data} - \langle v_i \rangle_{recon}) \tag{6}$$

where $< . >_{data}$ and $< . >_{recon}$ denote expectations over the training data and their reconstruction, respectively, and ϵ is the learning rate (different learning rates can be adopted). The quantities inside the angular brackets can be approximated after a few cycles of an alternating Gibbs sampling procedure [18], [19]. In practice, we use the conditional probabilities, $p(v_i = 1 \mid \mathbf{h})$, in the place of the v_i s and we only sample from the conditional probabilities, $p(h_j = 1 \mid \mathbf{v})$, during the Gibbs sampling operation. This modification permits to use patterns (feature vectors) whose elements lie in the range $[0, 1]$, as in the output of a softmax normalization procedure. Concerning the hidden nodes, sampling is performed by comparing $p(h_j = 1 \mid \mathbf{v})$ with a number drawn in $[0, 1]$ by a random number generator.

3.2. Linear visible units and binary hidden units

As an alternative to a softmax preprocessing stage, the training data can be normalized to zero mean and unit standard deviation per dimension. In this way, we can alternatively proceed to experimenting with Gaussian visible units, i.e., linear, real-valued units that have Gaussian noise [20], [21], [22]. The new energy function is

$$E(\mathbf{v}, \mathbf{h}) = \sum_{i=1}^{l+M} \frac{(v_i - a_i)^2}{2} + \sum_{j=1}^{K} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{7}$$

and the conditional probability, $p(v_i \mid \mathbf{h})$, of a visible unit becomes:

$$p(v_i \mid \mathbf{h}) = \mathcal{N}(a_i + \sum_{j=1}^{K} h_j w_{ij}, 1) \tag{8}$$

where $\mathcal{N}(\mu, 1)$ is the Gaussian distribution, with mean μ and unit standard deviation. During the CD training algorithm, we avoid sampling the Gaussian distribution of each visible unit and we simply let the unit feed the hidden units with the total excitation that it has received. On another (practical) matter, if Gaussian visible units are used, it is preferable to use stretched codewords for the class labels and keep the learning rate for weights and biases rather small, e.g., in the order of 0.001. This is because we want to reduce the risk of the training algorithm to diverge due to the linear nature of the visible units, which can receive unbounded input and do not possess a squashing function that is capable of bounding the output to a predefined range of values.

3.3. Class label inference

After the RBM has been trained, the pattern to be classified is "clamped" on the visible nodes, $v_i, i = 1, \ldots, l$, and the remaining visible nodes, $v_i, i = l+1, \ldots, l+M$ are initialized with random values, drawn from the interval $[0, 1]$ for the case of binary units, and the interval $[-1, +1]$ for the case of linear units. Then an alternating Gibbs sampling procedure starts. At each sampling cycle, we sample from the conditional probability of Eq. (1), which is computed for each hidden node, and then compute the conditional probability

of the visible nodes that correspond to the class label according to Eq. (2) or (8) (depending on the type of visible nodes). The Gibbs sampling procedure is repeated for several cycles (at least 100 in our study). In the end, we observe the values of the conditional probability at the visible nodes $v_i, i = l+1, \ldots, l+M$ and select the maximum value which will (hopefully) be very close to one and its position (index of visible node) will reveal the correct class.

4. DL RECONSTRUCTION OF THE CLASS LABELS

Dictionary learning (DL) is directly related to the sparse representation / sparse coding tasks. The objective of sparse coding is to choose a few elementary signals / vectors, called atoms, drawn from a prespecified set of such signals, referred to as *dictionary*; this dictionary provides a *data-adaptive* compact representation of the training data set in terms of the atoms. Let D be a dictionary matrix comprising r dictionary atoms, $\mathbf{d}_1, \cdots, \mathbf{d}_r$ as columns. A popular formulation of the corresponding optimization task is described as follows [2], [3], [23]:

$$\min_{D,B} \|X - DB\|_{\mathrm{F}}^2, \text{ s.t. } \|\mathbf{b}_j\|_0 \leq k, \; j = 1 \cdots N, \tag{9}$$

where X contains the training examples ($\mathbf{x}_i, \; i = 1 \ldots N$) as columns, $\|\cdot\|_{\mathrm{F}}$ is the Frobenious norm, $\|\cdot\|_0$ is the ℓ_0 pseudo-norm counting the number of nonzero components of the unknown vector $\mathbf{b}_j$ and k is the number of atoms, which are linearly combined to represent the training examples. Moreover, a constraint on the dictionary norm is necessary in order to avoid degenerate solutions, with the unit-norm request for each atom being the most popular.

The dictionary is usually trained in an iterative fashion alternating two learning stages until convergence. In the first stage, the dictionary D is fixed to its latest estimate and B is estimated, column by column, via a series of sparse coding , i.e.

$$\min_{\mathbf{b}_j} \|\mathbf{x}_j - D\mathbf{b}_j\|^2 \text{ s.t. } \|\mathbf{b}_j\|_0 \leq k, \; j = 1 \cdots N \tag{10}$$

In the second stage, the dictionary is updated whereas on the same time either the full matrix B, or its zero entries only, are kept fixed, depending on the specific DL method. One of the most popular and well-performing DL methods, the K-Singular Value Decomposition (K-SVD), [23], updates the dictionary atom by atom via a series of rank-1 approximations performed via truncated SVD.

The previously described methodology applies to a batch mode of operation. According to this, the full amount of data is used at each iteration of the algorithm, which is employed in the Dictionary Learning task. Nevertheless, if one has at her/his disposal a large amount of data, which is, for example, the case in Big Data applications, batch operation might not be the most suitable choice, as it requires huge computational and storage resources. Luckily enough, in such cases, one can resort to the online learning philosophy, e.g., [17]. In online learning algorithms, the data are processed sequentially, one or more per iteration step, until a certain convergence criterion is met. Online dictionary learning algorithms have been proposed in the works [5, 24]. In the current study, for the online dictionary learning task, we employ the algorithm of [5], referred here as Online Dictionary Learning (OnDiLe), since it has been shown to enjoy performance, which is close to that of batch algorithms. Without going into many details, the steps of the algorithm can be summarized as follows. At each iteration of the algorithm, one draws a certain number of training data, which is significantly smaller than the total number in the available set, and it could possibly be set equal to one (single data per iteration step). In the sequel, the sparse coding vectors, corresponding to the drawn data, are computed in a similar fashion as in Eq. (10) and finally, the dictionary is updated, by employing the block–coordinate descent algorithm. It is important to notice that, in contrast to batch processing, the sparse coding vector computation and the dictionary update, when operating online, employ a significantly smaller subset of the training set without big performance degradation, as it will become clear in the Simulations section.

In pattern classification with label tagging, both batch and online DL methods can be employed. Let $\bar{X} \in \mathbb{R}^{l+M,N}$ comprise the training data expressed in the augmented feature space, i.e. after the label tagging. This can be expressed as $\bar{X} = \begin{bmatrix} X \\ Y \end{bmatrix}$. Any dictionary learning method can be employed for the estimate of a dictionary D, which sparsely represent the columns of $\bar{X}$ with a fixed and user defined sparsity level k. The dictionary D is naturally split in two parts, $D = \begin{bmatrix} D_X \\ D_Y \end{bmatrix}$, with $D_X \in \mathbb{R}^{l,r}$ being trained to sparsely represent X and $D_Y \in \mathbb{R}^{M,r}$ sparsely represents the corresponding labels.

In the classification mode, an unlabeled vector $\mathbf{a}$ needs to be classified. Considering the augmented counterpart of $\mathbf{a}$, we write $\bar{\mathbf{a}} = \begin{bmatrix} \mathbf{a} \\ \hat{\mathbf{y}} \end{bmatrix}$, where $\hat{\mathbf{y}}$ is the label, which needs to be estimated. Treating $\hat{\mathbf{y}}$ as missing data, its estimation can be realized in two steps. First, the k-sparse representation of $\mathbf{a}$ is computed via

$$\hat{\mathbf{b}} := \min_{\mathbf{b}} \|\mathbf{a} - D_X\mathbf{b}\| \text{ s.t. } \|\mathbf{b}\|_0 \leq k, \tag{11}$$

and then, the label is estimated according to $\hat{\mathbf{y}} = D_Y\hat{\mathbf{b}}$. A similar approach is adopted in image impainting and audio imputation applications, when a dictionary, which sparsely represents the signal under consideration, is known, e.g., [2], [25], [26].

5. AUDIO GENRE CLASSIFICATION

5.1. Audio corpus

Our study focuses on the Ballroom Dataset (BD) [27], a class-imbalanced dataset that consists of 10 music classes: Cha Cha Cha, Jive, Quickstep, three Rumba styles, Samba, Tango, Viennese Waltz and Waltz. In the literature the three Rumba genres are usually treated as one, giving rise to a 8-class ("merged") problem. This particular dataset was chosen because it defines a hard classification task based on the results that have been reported over the last ten years, since its inception in 2004. For each class, a varying number of 30 s long audio segments are available, summing up to 698 clips.

5.2. Feature extraction

From each 30 s long audio segment we extract a number of *rhythmic signatures* based on the feature extraction algorithm that was proposed by one of the authors in [6]. The basic feature extraction steps are summarized below:

1. The audio segment is divided into overlapping long-term windows (10 s long, 9 s overlap).

2. At each long-term window, a short-term processing step extracts a sequence, $F = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_T\}$, of MFCC-like features.

3. The Averaged Magnitude Difference Function (AMDF), D, of the short-term sequence, is then computed over a range of lags as follows: $D(k) = \frac{1}{T-k} \sum_{i=k}^{T} \; || \; \mathbf{f}_i - \mathbf{f}_{i-k} \; ||, \;\; k = 1, \ldots, 800$ The maximum lag is 800 which corresponds to a 4 s long rhythmic

pattern assuming a 0.005 s long short-term processing step. Local minima (valleys) of D indicate dominant signal periodicities.
4. The sharpness of the local minima of D is quantified. As a measure of sharpness, the second derivative, D_2, of D is approximated. Local maxima of D_2 refer to audio periodicities (rhythmic events). The resulting 800-dimensional feature vector is the rhythmic pattern, $\mathbf{x}$, which is augmented with the respective class label to produce $\bar{\mathbf{x}}$.

5.3. Training and testing datasets

A repeated random sub-sampling validation scheme is used. In particular, from each genre, 70% of the clips are randomly selected and are inserted in the training set; the remaining ones (30%) are the testing data. This process is repeated 5 times and the classification accuracy is computed as the average over all 5 runs. The resulting numbers of training and test patterns are ≈ 10k and ≈ 4.3k, respectively.

Depending on the learning algorithm, a pattern normalization step is required. We examine three normalization stages:
(a) Each feature dimension, $x_i, i = 1, 2, \ldots, l$, of the training set, is normalized to zero mean and unit standard deviation. The mean and standard deviation values which have been computed over the training set are then used to normalize the respective testing dataset. In this case, the codewords of the labels consist of values ± 1.
(b) Each feature dimension of the training set is softmax normalized in the range $[0, 1]$.
(c) Each pattern in the training and testing datasets is individually smoothed using a softmax squashing function. This is not a normalization step in the strict sense, but again, it serves to map each pattern element to the range $[0, 1]$ [6].

Schemes (b) and (c) permit to assign a probabilistic interpretation to the generated features in the case of the RBM; values close to 1 indicate a higher probability that the respective lag (feature dimension) corresponds to a signal periodicity. In these two cases, we are using binary codewords for the class labels. In the sequel, we will refer to the three normalization algorithms with the abbreviations $Norm1$, $Norm2$ and $Norm3$, respectively.

5.4. Performance evaluation

We present the classification accuracy for the 10-class task and the "merged" classes scenario, both on a pattern and track level. The former is only sparsely reported in the literature but it is very useful when we deal with the problem of randomly selecting a pattern and classifying it without taking into account all the classification decisions from the same audio file. In our study, at the audio track level, the classification decision is taken by simple majority voting. Table 1 presents a comparative study of well known algorithms in the literature with the algorithms in this paper. The classification accuracies are the ones reported in the respective papers (first column).

For the RBM, we have experimented with a varying number of hidden layer nodes, from 100 to 1000. It has turned out that there only exists a small performance fluctuation (around 1.5%) and that the best accuracy has been achieved around 300 hidden nodes, both for binary and linear visible units and irrespective of the normalization method that was used. In the DL case, several dictionary sizes (500 - 2000) were tested together with different sparsity levels (1-5). We did not observe significant performance fluctuations.

Furthermore, for the sake of comparison, we have reproduced the method in [6], which was proposed by one of the authors and was the best performing method in the MIREX-2013 competition for the Latin Genre classification task (on a proprietary dataset). We used a 3-layer network with binary stochastic units (with 500, 500 and 2000 hidden nodes at the three hidden layers, respectively). The network was pre-trained in a layer-wise mode and fine-tuned with a back-propagation algorithm.

Method (with merged classes)	**Accuracy (% audio clips)**
[9], [8], [28]	89.2, 86.9, 85.7
[10], [29], [11], [14]	79.6, 66.89$\pm$ 5.26, 67, 65.1
[6] (Deep Network)	90.14
RBM-Norm3-f	**86.6**, **83.6** (10-class)
RBM-Norm2-f	**85.2**, **79.8** (10-class)
RBM-Norm1-f	**86.4**, **81.6** (10-class)
DL-KSVD-Sparsity-3-f	**87.0**, **83.0** (10-class)
DL-online-Sparsity-3-f	**89.0**, **82.2** (10-class)

Table 1. Method comparison on the Ballroom dataset. Boldface entries correspond to the algorithms of this paper. The parentheses indicate complementary performance on the 10-class task.

`Discussion:` In this work we have only employed simple rhythmic features without making any attempt for feature optimization. In contrast, in most of the already reported results listed in Table 1, elaborated features in the domains of timbre and rhythm have been used, which are however hard to reproduce. Therefore, we expect further performance improvement if more complicated feature extraction schemes are combined with the method proposed in this paper. Moreover, it turns out that, DL methods give results competitive to the best reported performance so far ([9] and [6]). Interestingly, the online approach does not lead to any performance deterioration compared to the batch mode of operation. However, it has to be kept in mind that the task is a non-convex one and hence, the specific batch mode algorithm and the initial conditions may affect the final solution. Despite this remark, this is still a notable advantage, allowing to cope with online classification problems, where retraining from scratch is not necessary whenever new data become available. Moreover, storage of the full amount of training data is avoided since the learning procedure can be realized on the fly. Concerning the RBM architecture, it can be observed that it follows closely, despite its simplicity and without having made any attempt for more complicated RBM nodes and training approaches.

Finally, we present complementary classification results at the pattern level, for the 10-class task. These task requirements have not been systematically addressed in the literature. The respective results are desirable in applications where isolated classification decisions need to be computed over shorter time intervals (on a 10 s basis in our study). The results of the proposed methods are as follows: 82.6% for the RBM architecture and 83% both for batch and online DL implementations. The deep network architecture of [6] performed better (84.8%). However, taking into account the complexity and the computational demands of deep networks, along with the fact that their architecture prohibits online mode of operation, we conclude that the approach proposed here offers a very reasonable trade off among complexity, accuracy and breadth of applicability, even for the hard case of pattern level decisions in the 10-class case.

6. CONCLUSIONS

This paper approached the task pattern classification from a missing data perspective and designed a family of algorithms based on RBMs and Dictionary Learning techniques, which have proved to be competitive on the hard task of audio genre classification in the context of the publicly available Ballroom Dataset. Future research will investigate datasets from other research disciplines, will examine other missing data algorithms and will delve deeper into the existing approach, especially in the case of online learning.

7. REFERENCES

[1] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th ACM International Conference on Machine Learning (ICML)*, 2007, pp. 791–798.

[2] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, Springer, 2010.

[3] Sergios Theodoridis, Yannis Kopsinis, and Konstantinos Slavakis, *Sparsity-aware learning and compressed sensing: An overview*, Academic press, 2014.

[4] Geoffrey E Hinton, "To recognize shapes, first learn to generate images," *Progress in Brain Research*, vol. 165, pp. 535–547, 2007.

[5] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro, "Online learning for matrix factorization and sparse coding," *The Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[6] Aggelos Pikrakis, "A deep learning approach to rhythm modeling with applications," in *Proceedings of the 6th International Workshop on Machine Learning and Music (MML 2013)*, 2013.

[7] Andre Holzapfel and Yannis Stylianou, "Rhythmic similarity of music based on dynamic periodicity warping," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 2217–2220.

[8] Andre Holzapfel and Yannis Stylianou, "A scale transform based method for rhythmic similarity of music," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2009, pp. 317–320.

[9] Tim Pohle, Dominik Schnitzer, Markus Schedl, Peter Knees, and Gerhard Widmer, "On rhythm and general music similarity.," in *Proceedings of ISMIR*, 2009, pp. 525–530.

[10] Fabien Gouyon, Simon Dixon, Elias Pampalk, and Gerhard Widmer, "Evaluating rhythmic descriptors for musical genre classification," in *Proceedings of the 25th AES International Conference*, 2004, pp. 196–204.

[11] Emiru Tsunoo, George Tzanetakis, Nobutaka Ono, and Shigeki Sagayama, "Beyond timbral statistics: Improving music classification using percussive patterns and bass lines," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 1003–1014, 2011.

[12] Emiru Tsunoo, George Tzanetakis, Nobutaka Ono, and Shigeki Sagayama, "Audio genre classification using percussive pattern clustering combined with timbral features," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2009, pp. 382–385.

[13] Klaus Seyerlehner, Markus Schedl, Reinhard Sonnleitner, David Hauger, and Bogdan Ionescu, "From improved autotaggers to improved music similarity measures," *Adaptive Multimedia Retrieval*, 2012.

[14] Jan Schluter and Christian Osendorfer, "Music similarity estimation with the mean-covariance restricted boltzmann machine," in *10th International Conference on Machine Learning and Applications and Workshops (ICMLA)*, 2011, vol. 2, pp. 118–123.

[15] Klaus Seyerlehner, Gerhard Widmer, and Tim Pohle, "Fusing block-level features for music similarity estimation," in *Proceedings of the 13th Int. Conference on Digital Audio Effects (DAFx)*, 2010, pp. 225–232.

[16] Mikael Henaff, Kevin Jarrett, Koray Kavukcuoglu, and Yann LeCun, "Unsupervised learning of sparse features for scalable audio classification.," in *Proceedings of ISMIR*, 2011, pp. 681–686.

[17] Sergios Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*, Academic Press, 2015.

[18] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[19] Yoshua Bengio, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[20] Yoav Freund and David Haussler, *Unsupervised learning of distributions of binary vectors using two layer networks*, Technical Report, UCSC-CRL-94-25, 1994.

[21] Tim K Marks and Javier R Movellan, "Diffusion networks, product of experts, and factor analysis," in *International Conference on Independent Component Analysis*, 2001, pp. 481–485.

[22] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis, "Modeling human motion using binary latent variables," in *Advances in Neural Information Processing Systems*, 2006, pp. 1345–1352.

[23] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[24] Karl Skretting and Kjersti Engan, "Recursive least squares dictionary learning algorithm," *IEEE Transactions on Signal Processing*, vol. 58, no. 4, pp. 2121–2130, 2010.

[25] Jinyu Han, Gautham J Mysore, and Bryan Pardo, "Audio imputation using the non-negative hidden markov model," in *Latent Variable Analysis and Signal Separation*, pp. 347–355. Springer, 2012.

[26] Paris Smaragdis, Bhiksha Raj, and Madhusudana Shashanka, "Missing data imputation for spectral audio signals," in *Proc. of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2009, pp. 1–6.

[27] Fabien Gouyon, Anssi Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1832–1844, 2006.

[28] Jesper Højvang Jensen, Mads Græsbøll Christensen, and Søren Holdt Jensen, "A tempo-insensitive representation of rhythmic patterns," in *Proceedings of the 17th European Signal Processing Conference (EUSIPCO)*, 2009.

[29] Arthur Flexer, Fabien Gouyon, Simon Dixon, and Gerhard Widmer, "Probabilistic combination of features for music classification.," in *Proceedings of ISMIR*, 2006, pp. 111–114.