Contemporary Engineering Sciences, Vol. 10, 2017, no. 3, 145 - 155
HIKARI Ltd, www.m-hikari.com
https://doi.org/10.12988/ces.2017.610163

Deep Unsupervised Feature Extraction of Sensory Signals

Yassine Berradi and Abdelaziz Bouroumi

Information Processing Laboratory, Ben M'sik Faculty of Sciences
Hassan II University of Casablanca, BP.7955 Sidi Othmane
Casablanca, 20702, Morocco

Copyright © 2016 Yassine Berradi and Abdelaziz Bouroumi. This article is distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

We propose an application of deep learning to the problem of automatic feature extraction of sensory signals. The proposed method is based on a deep neural network architecture composed of 16 auto-encoders that are sequentially trained to encode the input signals using an unsupervised learning algorithm. The number of neurons per layer of this architecture is heuristically determined. Experimental results that illustrate the performance of this method are presented and discussed using a large benchmark real data set.

Keywords: Deep Learning; Auto-encoder; Feature Extraction; Backpropagation; Unsupervised Learning

1 Introduction

Feature selection of sensory signals may greatly impact the output of machine learning algorithms when applied to this kind of data [1]. This operation is generally performed on original data using feature engineering techniques that require laborious human intervention, for this reason, the intelligence behind many machine learning algorithms, in terms of pattern recognition and classification, has shifted to the human engineered feature extraction process. When incomplete or erroneous features are extracted, the performance of machine learning algorithms is inherently limited, and that may fail to extract and organize well discriminative information from row input data [2]. Hence the necessity for automatic feature extraction techniques that can help making machine learning algorithms less dependent on feature engineering, especially in real-world applications that involve signal analysis and processing.

Deep learning provides an effective mean for extracting high level feature hierarchies that can be used for classification and regression tasks [3]. It allows computational models with multiple processing layers that learn data representations with multiple levels of abstraction [4]. The use of deep neural networks as feature learning algorithms may lead to better representations in terms of classification errors [5], quality of generated samples [6], and invariance properties of the learned features [7].

For large learning databases, using deep architectures for feature extraction is relatively fast as the time complexity of the corresponding algorithms is linear in terms of the number of training samples [8][2]. Moreover, deep generative models are more robust to the overfiting problem than discriminative models such as multilayer perceptrons.

The goal of this paper is to present an application of deep learning to the problem of extracting robust features of sensory signals data. The method used for this application is based on auto-encoders as unsupervised learning models for the construction of a deep neural network dedicated to the studied problem. The details of this method, including the architecture of the used neural network, are presented in section II, preceded by a brief reminder of auto-encoders. Experimental results are presented and discussed in section III. Finally, a conclusion and some suggestions for future work are given in section IV.

2 Description of the proposed method

2.1 Auto-encoders

Auto-encoders (AEs) are three-layer neural networks aimed at tackling the problem of training artificial neural networks with back-propagation algorithm using the input data as a teacher [9]. They are unsupervised learning models that constitute a beginning step towards the resolution of the issue of “How synaptic changes induced by local biochemical events can be coordinated in a self-organized manner to produce global learning and intelligent behavior” [10].

More recently, auto-encoders have taken center stage in deep architectures, where they are stacked in greedy fashion to build layers in an unsupervised manner [8]. The main goal being to find good representative features of input data that can be used for the purpose of dimensionality reduction and/or classification tasks.

Technically speaking, AEs are suitable algorithms for independently training every intermediate layer of a deep neural network, in order to find the best possible value for each synaptic weight. The training process using AEs involves two main steps. The first step starts by propagating the input signal through the network connections to the output layer. In the second step the output signal is propagated backward through the same connections to the input layer in order to get a reconstruction of the input data. Then the error signal (distance between input data and reconstruction data) is propagated forward in order to adjust the synaptic weights using stochastic gradient descent method. These two steps are repeated until finding in average the possible minimum value of the reconstruction error.

Mathematically speaking, given a set of n d-dimensional unlabeled training data $X = \{x_1, x_2, x_3, ..., x_n\}$. Where $x_i \in \Re^d$, the auto-encoder takes X as input and first maps it with an encoder that is a function f parameterized by w that computes the feature vector h, we define:

$$h = f(X; w) \tag{1}$$

where h is the feature vector or representation or code computed from X, w is the matrix of weights (Fig. 1). The feature vector h is then mapped back with a decoder modeled by the parameterized function g that computes the reconstruction vector $\hat{X}$ of the same shape of X, we define:

$$\hat{X} = g(h; w^T) \tag{2}$$

where w^T is the transpose matrix of w. $\hat{X}$ Should be seen as a prediction of X given h, we note:

$$\hat{X} = p(X/h) \tag{3}$$

The f or g function commonly uses the mapping defined by nonlinear functions like: softmax, sigmoid, hyperbolic tangent, linear activation, etc. Choosing which among them depends on the interval in which the input vector X is scaled e.g., if $x_i \in [0,1]$, we must use in the decoder the sigmoid as activation function in order to accelerate the learning process, by making $\hat{X}$ close to X.

AEs are parameterized by their encoder and decoder. These parameters are adjusted simultaneously, as possible as getting a lowest possible value of the reconstruction error RE, defined as:

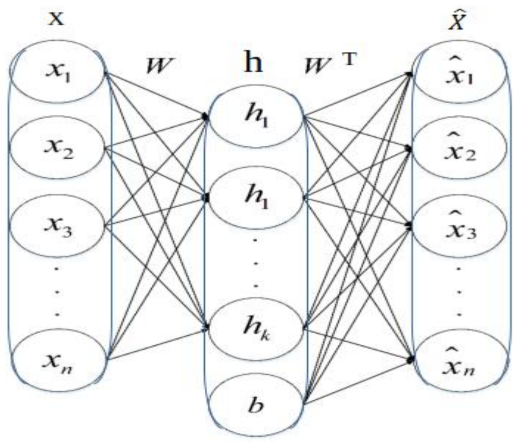

Figure 1: Example of an Auto-encoder architecture. X is the input vector, which has n features. h is either the output vector of the encoder and the input of the decoder. $\hat{X}$ is the reconstruction of X

$$RE = \sum_{i=1}^{n} (x_i - \hat{x}_i)^2 \tag{4}$$

The AEs are trained using back-propagation algorithm, and the method used to minimize RE is the stochastic gradient descent as in the training of Multi-layer-perceptions (MLPs), for more details see [11]. The parametres of the encoder w and of the decoder w^T are adjusted during the learning process using the rules define as:

$$w_{ij}{}^{T}(t) = w_{ij}{}^{T}(t-1) - \eta \times (\delta_j(n) \times h_i + \delta_{ij}(n) \times x_j) \tag{5}$$

$$\delta_j(n) = x_j - \hat{x}_j \tag{6}$$

$$\delta_{ij}(n) = \sum_{j=1}^{n} w_{ij} \times \delta_j(n) \tag{7}$$

$$w = (w^T)^T \tag{8}$$

Where i is the ith unit in vector h. j is the jth unit in both X and $\hat{X}$ vectors. η is the learning rate.

In the adaptation above, note that, the Eq.5 is properly used for linear encoder and decoder (linear activation).

2.2 Deep Neural Network Architectures

The deep neural network architectures used in this application are a generative deep architectures which characterize the high-order correlation properties of the input signals composing a series of processing stage [12], they are commonly associated with the term of " unsupervised feature learning ", where the features are automatically learned at multiple levels of abstraction. This idea allows the network to learn complex non-linear functions, mapping the input to the output directly from input signals.

Since 2006, these kind of deep architectures have been applied with success in classification tasks [13], regression [14], dimensionality reduction [15], processing signals [16] and others.

In this work, after training an auto-encoder on the input data, it is stacked so that its output became the input of another auto-encoder. This process is done in a deep way constructing a deep architecture until finding good features (Fig. 2). This operation of stacking sequentially every layer trained by unsupervised learning model improves learning more robust representations or abstracts as we stack more layers.

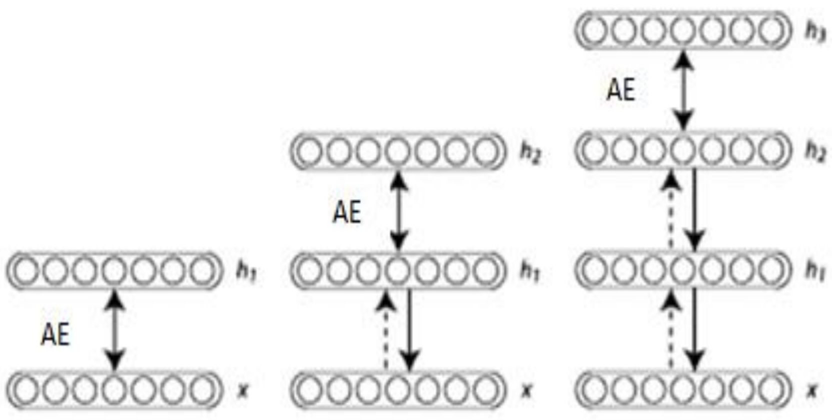

Figure 2: Architecture of a deep Auto-encoders where x is the input data and h_i is the feature vector of the previous layer.

3 Numerical Results and Discussion

In this section, we present the experimental results obtained by applying the proposed method we describe in the previous section to real database related to the problem of finding good features for a Human Activity Recognition systems.

Our application is object oriented software coded in python under a Linux environment. The studied database has been made available for public use and it is presented as raw inertial sensors signals for each pattern. It has been submitted as Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set in the UCI Machine Learning Repository [17], it is composed of 1214 instances distributed over12 different classes composed of six basic activities: three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs and walking upstairs), other six activities that occurred between the static postures. These are: stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand. Each instance is characterized by 438 numerical attributes plus the class label. The attribute values represent the sensory signals come from the gyroscope and accelerometer that are embedded in Smartphone (Samsung Galaxy S 2), this latter is positioned on the arm of the participant who performed the activities previously mentioned [18].

In order to study the performance of the proposed method for various sizes of the training database, the original database was partitioned in two subsets, each one has different size. The first subset contains 971 vectors, which represent 80% of the original database. This subset was used to train different network architectures by the proposed method. In order to find their best possible weight matrix that performs the task of finding good data representation (or features) directly from the training subset of the database. The second subset contains 243vectors that represent 20% of the original database used for testing the performance of the weight matrix in terms of generalization.

In this study, we chosen deep neural network architecture fixed to 17 layers (including input layer), stacked one after one, using AEs as a constructor of layers. The size of the first layer was chosen close as possible to that of the input layer, because theoretically speaking, the networks could just learn to be the identity and perfectly minimize the reconstruction error RE (eq.4) when the size of stacked layer increases significantly compared to the size of the input layer[19]. The size of the added layers is decreasing for the first experiment, increasing slowly for the second experiment and fixed for the third during the learning process. Moreover the performance of the learned features at each layer was tested using SVMs classifier and compared to that of the previous layer, using global classification accuracy. Fig.3, illustrates the variation of this accuracy with respect to the number of stacked layers, this analysis is performed for each of the three experiments mentioned above.

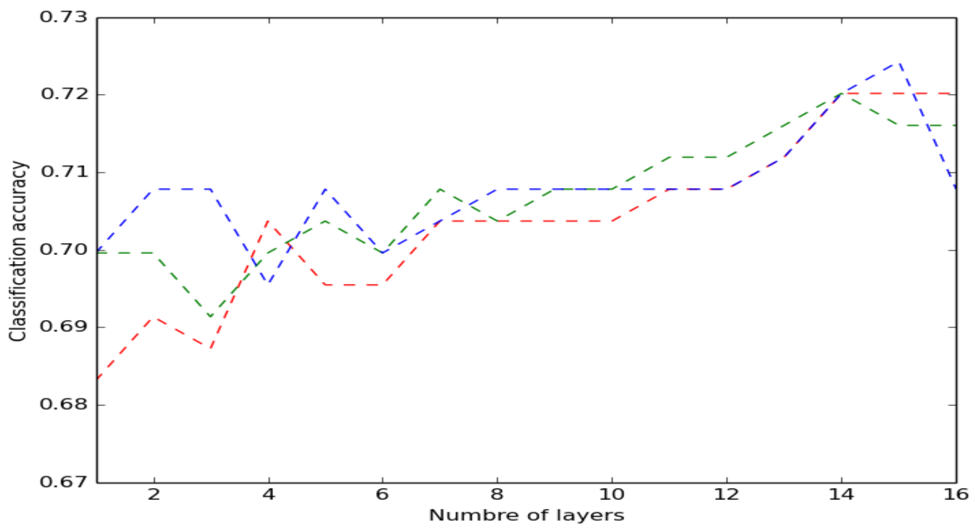

Figure 3: Variation of the classification accuracy of SVM to the learned features with respect to the number of layers. Blue curve referred to the decreasing architecture in terms of size/layer. Red and green curves are respectively referred to the increasing and fixed architectures.

The learning rate η (see, eq.5) was taken first in $[10^0, 10^{-3}]$ range, then decreasing during the learning process for each iteration, using the adaptation below:

$$\eta = \eta \times 0.95 \tag{9}$$

This considered adaptation learning rate is used based on a basic notion, presented by Kesten [20]. Kesten proposes that if consecutive changes of a weight (i.e., $\Delta w_{ij}(t-1)$ and $\Delta w_{ij}(t)$) possess opposite signs, the weight value is oscillating. Hence, the learning rate for that weight should be decremented through the learning time.

In order to study the classification performance of the learned data representation, we use SVM as classifier [21] applied on the features learned from training subset and those transformed from testing subset. Tab.1 shows the classification accuracies corresponding to different chosen architectures trained on both, training and testing subsets of the database. The best obtained result over 10 runs is presented using the proposed algorithm. The classification accuracy of the (AEs+SVMs) method applied to the input signals is optimized compared to that of SVMs only. For this reason, the idea of stacking AEs sequentially generating at the end robust features, enhances the performance of machine learning methods (case of SVMs), applied on these features, in terms of classification ability.

Method	**Deep Architecture**	**Classification Accuracy**	
		Train %	***Test %***
SVMs	Input layer (438)	95.05	68.95
AEs + SVMs	16 layers Decreasing from 438 to 250 size/layer	93.71	**72.42**
AEs + SVMs	16 layers Increasing from 438 to 510 size/layer	93.82	72.01
AEs + SVMs	16 layers Fixed to 438 size/layer	93.51	72.01

Table 1. The Classification Accuracy for 3 Architectures

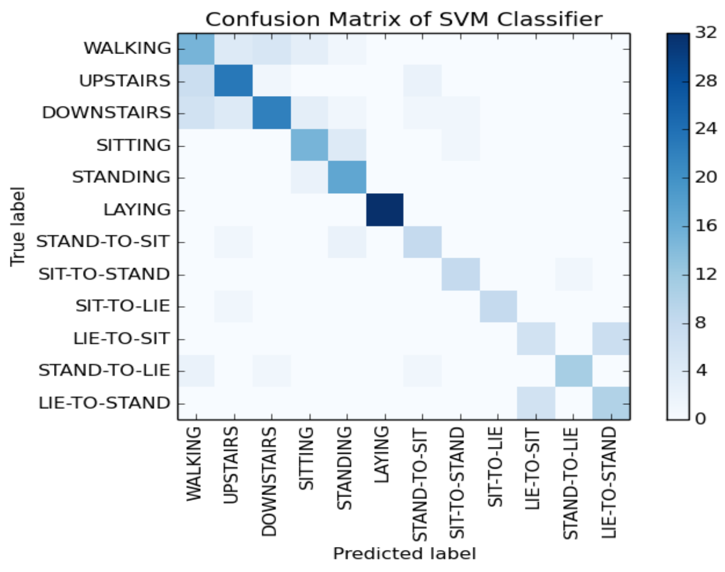

Figure 4: Confusion matrix of SVMs classifier applied to the features extracted from the deep architecture corresponding to the decreasing structure. This figure illustrates separate classification results for each considered physical activity.

In Fig.4, a separate classification results for each considered physical activity are presented in the confusion matrix of SVM classifier, which applied to the last features extracted by the deep architecture that is constructed by 16 layers, decreasing from 438 to 250 (size /layer), since this architecture presents the best global classification accuracy 72.42 % for this study (see Tab 1). This figure shows also that almost the classification task of the physical activities, which spend time like: walking, walking upstairs, walking downstairs, is better performed than that of the static physical activities (stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie and lie-to-stand).

4 Conclusion

In this paper, we have presented an application of deep learning paradigm to the problem of automatic feature extraction from sensory signals data. The proposed method for tackling this problem is based on the use of a collection of auto-encoders, which are sequentially stacked in order to form a deep neural network that can be trained in an unsupervised manner, which leads to a hierarchy of robust features or data representations. The performance of this method is experimentally tested using a real database of sensory signals taken from the UCI machine learning repository [17]. The signals in this database are provided by gyroscope and accelerometer sensors embedded in the smart phones of 30 volunteers and are related to twelve physical activities of these volunteers (sitting, standing, walking, laying, walking upstairs, walking downstairs, stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie and lie-to-stand). The resulting features are used as input data for an SVM classifier in order to find the best classification of these data into the twelve different classes. The simulation results are encouraging in terms of classification accuracy. This stimulates further development of this study in order, for example, to find a way for reducing the dimensionality of the learned features.

References

[1] Y. Bengio, A. Courville and P. Vincent, Representation learning: a review and new perspectives, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35** (2013), no. 8, 1798-1828.
https://doi.org/10.1109/tpami.2013.50

[2] S. Koyamada, Y. Shikauchi, K. Nakae, M. Koyama and S. Ishii, Deep learning of fMRI big data: a novel approach to subject-transfer decoding, (2015), arXiv preprint arXiv:1502.00093.

[3] R. Salakhutdinov, Learning deep generative models, *Annual Review of Statistics and its Application*, **2** (2015), 361-385.
https://doi.org/10.1146/annurev-statistics-010814-020120

[4] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, Deep learning, *Nature*, **521** (2015), no. 7553, 436-444. https://doi.org/10.1038/nature14539

[5] H. Larochelle, D. Erhan and P. VincentDeep, Learning using Robust Interdependent Codes, *12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, (2009), 312-319.

[6] A. Mnih and G. E. Hinton, A scalable hierarchical distributed language model, *Advances in Neural Information Processing Systems*, (2009), 1081-1088.

[7] I. J. Goodfellow, H. Lee, Q. V. Le, A. M. Saxe and A. Y. Ng, Measuring invariances in deep networks, *Advances in Neural Information Processing Systems*, (2009), 646-654.

[8] G. E. Hinton and R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *Science*, **313** (2006), no. 5786, 504-507. https://doi.org/10.1126/science.1127647

[9] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning internal representations by error propagation, in *Parallel Distributed Processing*, Vol. 1, Foundations, MIT Press, Cambridge, MA, 1986.

[10] P. Baldi, Autoencoders, unsupervised learning, and deep architectures, *Workshop on Unsupervised and Transfer Learning*, **27** (2012), 37-50.

[11] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning representations by back-propagating errors, *Cognitive Modeling*, 5, MIT Press, 1988.

[12] L. Deng, Three classes of deep learning architectures and their applications: a tutorial survey, *APSIPA Transactions on Signal and Information Processing*, (2012).

[13] A. Ahmed, K. Yu, W. Xu, Y. Gong and E. P. Xing, Training hierarchical feed-forward visual recognition models using transfer learning from pseudo tasks, Chapter in *Computer Vision (ECCV'08)*, Springer Berlin Heidelberg, 2008, 69-82. https://doi.org/10.1007/978-3-540-88690-7_6

[14] R. Salakhutdinov and G. E. Hinton, Using deep belief nets to learn covariance kernels for Gaussian processes, in *Advances in Neural Information Processing Systems 20 (NIPS'07)*, J. Platt, D. Koller, Y. Singer, and S. Roweis, eds., MIT Press, Cambridge, MA, 2008, 1249-1256.

[15] R. Salakhutdinov and G. E. Hinton, Learning a nonlinear embedding by pre-serving class neighbourhood structure, in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS'07)*, Omnipress, San Juan, Porto Rico, 2007.

[16] M. Ranzato and M. Szummer, Semi-supervised learning of compact document representations with deep networks, in *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08)*, **307** (2008), 792-799. https://doi.org/10.1145/1390156.1390256

[17] A. Frank and A. Asuncion, UCI Machine Learning Repository, 2010.

[18] Jorge-L. Reyes-Ortiz, Luca Oneto, Albert Sama, Xavier Parra, Davide Anguita, Transition-Aware Human Activity Recognition Using Smartphones, *Neurocomputing*, **171** (2016), 754-767. https://doi.org/10.1016/j.neucom.2015.07.085

[19] G. E. Hinton, S. Osindero, and Y.-W. Teh, A fast learning algorithm for deep belief nets, *Neural Computation*, **18** (2006), no. 7, 1527-1554. https://doi.org/10.1162/neco.2006.18.7.1527

[20] H. Kesten, Accelerated stochastic approximation, *Annals of Mathematical Statistics*, **29** (1958), 41-59. https://doi.org/10.1214/aoms/1177706705

[21] F. Pedregosa, G. Varoquaux, et al., Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, **12** (2011), 2825-2830.

Received: October 19, 2016; Published: January 9, 2017