Artificial Neural Networks in Feature Extraction: A Review

Joaquim Leitão*
*jpleitao@dei.uc.pt
CISUC, Department of Informatics Engineering, Univesity of Coimbra, Portugal

Abstract—Lol, the abstract

Index Terms—Feature Extraction, Artificial Neural Networks.

1 Introduction

INTRO
Dizer que foco na utilização de redes neuronais para efeitos de feature extraction (nao tanto de selection)
Colocar sigla para Feature Extraction - FE
4 pages
Do not forget to provide an overview of the paper!

2 Background

The current section addresses the importance of Feature Extraction in Intelligent Systems. In the literature this concept is often presented in the sequel of the well-known curse of dimensionality phenomenon, as a collection of methods that allow a reduction in the problem's dimensionality.

2.1 Curse of Dimensionality

The curse of dimensionality refers to how certain learning algorithms may perform poorly when confronted with high-dimensional data. In many Machine Learning (ML) problems, specially in *classification* tasks, low-dimension feature spaces may not allow for a good separation of the data, leading to low-accuracy classifications. By considering more features, a feature space where an hyperplane that perfectly separates the (training) data can be determined with a resulting increase in classification accuracy.

Nonetheless it has been demonstrated that, as dimensionality increases, the amount of training data needed to accurately generalise the produced classifier grows exponentially. In addition, a small *training samples-to-features* ratio may also degrade performance [1].

In practice, as training data is limited a scenario similar to figure 1 is observed: for a given training dataset, classification performance increases as more features are added until an optimum is reached. Further increasing the dimensionality decreases classification performance.

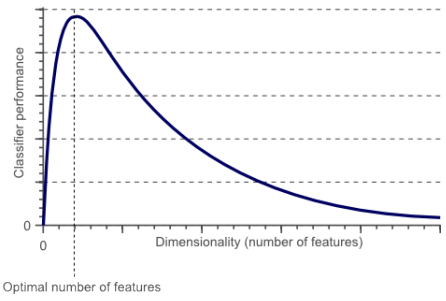

Figure 1. Curse of Dimensionality. Taken from [2].

A practical implication of the curse of dimensionality is that a system designer should select a small number of features when confronted with a limited training dataset. A general accepted practice is to keep the number of training samples at least ten times higher than the dimensionality: $n/d > 10$ [1].

2.2 Dimensionality Reduction

In light of the discussion conducted in 2.1, a common interest in the vast majority of ML applications is to keep the number of features as small as possible.

Feature Extraction and Feature Selection techniques are the most cited and recognised methods for achieving dimensionality reduction in PR problems. Feature Extraction algorithms propose a new set of features by performing transformations on a given feature set. On their turn, Feature Selection algorithms (ideally) select the best subset of features a provided feature set. As pointed out by Jain *et al.* [1], feature extraction is typically applied before feature selection[1].

The main issues in dimensionality reduction, namely in feature extraction and selection, are related with the choice of criterion function and the appropriate dimension for the reduced feature space.

3 Neural Networks in Feature Extraction

Artificial Neural Networks (ANNs) have been extensively applied in the field of machine learning, in both classification and regression tasks. Because of their self-learning and trained characteristics, ANNs have also been successfully applied in FE tasks.

Considering the simplest form of ANNs, feedforward networks, FE can be obtained in the output of each hidden layer: the application of the neurons' activation function to the layer's input produces an output that can be interpreted as a set of new features [1]. Depending on the neurons' activation function this new set of features can be a linear or nonlinear combinations of the features in the input feature set (Usually is nonlinear).

It is common to find in the literature types of ANNs where the learning procedure is supervised, and others which perform unsupervised learning. In the task of FE, supervised learning can be difficult to implement, as it requires the correct representation in the low-dimensional space for each training instance.

Instead, ANNs used for FE tend to adapt a different approach: They are trained to reconstruct input samples at the output layer [3], and in this process ANNs learn an encoder and

1. The idea is to start by defining a new feature set based on the original one, and then select the features that, hopefully, allow a complete separation of the training data.

a decoder, to encode and decode the input sample in a lower dimension space. Again, this can either be linear or nonlinear, depending on the neurons' activation functions (often is nonlinear). Such learning procedure is classified as unsupervised in the literature [4].

In the remainder of the current section the most popular and referenced ANN architectures applied in FE tasks are intended to be covered. Such architectures include *Feedforward* (namely *Stacked Autoencoders*), *Convolutional*, *Recurrent*, *Radial Basis*, *Restricted Boltzmann Machines* and *Self-Organising Maps* [1], [4], [7]–[10].

FIXME: Rever a lista de arquitecturas para confirmar que não me esqueci de nehuma!!

3.1 Feedforward

In the beginning of the current section feedforward networks were used as a motivation for the application of ANNs in FE tasks.

In its simplest forms, FE can be achieved through single or multi-layer feedforward ANNs by training the network to reconstruct the input signal in a conventional way, using a backpropagation algorithm (or other popular alternatives). Additional layers can also be considered to perform other tasks, such as classification. In this scenario training examples to which the output class is known are fed to the network, an the FE process is learned automatically in the early layers. Networks trained with this approach have been proven to suffer from well-known *vanishing gradient* problems [5].

In this sense, training of feedforward networks for FE purposes usually comprises two steps: *Pre-training* and *Fine tuning*.

During pre-training each hidden layer involved in the FE process is trained independently of the others in an unsupervised way: A series of 3-layer feedforward nets are trained to reconstruct their respective inputs at the output layer. The idea is to make each network learn an encoder and a decoder[2]. In this process, the first network is trained with the input data; the second network is trained with the output of the first network's encoder, and so on. This process is illustrated in figure 2.

When pre-training is complete, the encoders are stacked together. Due to this last step, ANNs developed in this way are often referred to as *Stacked autoencoders*.

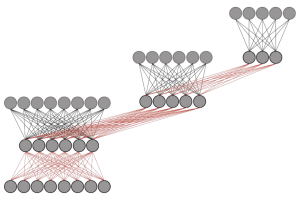

Figure 2. Stacking autoencoders. Taken from [6].

After the described pre-training, a fine tuning step is usually carried out by considering an additional layer (usually with a softmax activation function) and training the network with backpropagation (or other algorithm), to lightly adjust the weights of the pre-trained layers.

At this moment it is important to address one point in the discussion: based on what has been presented, autoenconders may simply learn the identity function for both the encoder and the decoder, which is not desired. To prevent this, certain activation functions can be forced, producing *regularized autoencoders* such as sparse autoencoders [11], denoising autoencoders [12] and contractive autoencoders [13].

2. The encoder is learned from the input to the hidden layer, while the decoder is learned from the hidden to the output layer.

3.2 Convolutional Neural Networks

3.3 Recurrent Neural Networks

3.4 Radial Basis Functions

3.5 Restricted Boltzmann Machines

3.6 Self-Organising Maps

In Self-Organising Maps (SOM), another type of ANN used for NLFE [1], [14], a low-dimension discretized representation of the input space is produced. Such representation is called a map. A SOM consists of neurons with associated weight vectors (with the same dimension as the input). The neurons are positioned in the map space, usually in a two-dimensional regular spacing in either a hexagonal or rectangular grid (instead of being organised in layers, like in feedforward ANNs).

During the training of a SOM, inputs are presented to the network and, for each, the weight vector closest to the input vector is identified. Then, the weights of all the neurons in the neighbourhood of the winner neuron (that is, the one whose weight vector was selected) are updated, so that they move towards the input vector. In this way, weight vectors of neighbouring neurons in the grid are likely to represent inputs which are close in the original feature space - a "topology-preserving" map is, thus, formed [15], [16]. An example of an application where SOM was used for FE can be found in [17], although many more are available in the literature.

4 Conclusion

Lol, conclusion

References

[1] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[2] "The curse of dimensionality in classification." http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/. Accessed: 2017-02-28.
[3] M. A. Kramer, "Autoassociative neural networks," *Computers & chemical engineering*, vol. 16, no. 4, pp. 313–328, 1992.
[4] O. Fabius and J. R. van Amersfoort, "Variational recurrent auto-encoders," *arXiv preprint arXiv:1412.6581*, 2014.
[5] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.
[6] R. Barata, "Stacked autoencoders." Lecture. Advanced Topics in Cognitive Models. University of Coimbra, 2017.
[7] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks*, pp. 52–59, Springer, 2011.
[8] G. Kvascev, M. G. Kvascev, and Z. Djurovic, "Radial basis function network based feature extraction for improvement the procedure of sourcing neolithic ceramics," in *Neural Network Applications in Electrical Engineering (NEUREL), 2012 11th Symposium on*, pp. 95–100, IEEE, 2012.
[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
[10] E. Marchi, F. Vesperini, S. Squartini, and B. Schuller, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *Computational Intelligence and Neuroscience*, vol. 2017, 2017.
[11] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.
[12] J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 265–272, 2011.
[13] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 833–840, 2011.
[14] Z. M. Hira and D. F. Gillies, "A review of feature selection and feature extraction methods applied on microarray data," *Advances in bioinformatics*, vol. 2015, 2015.
[15] T. Kohonen, "Self-organising maps," *Springer Series in Information Science*, vol. 30, Berlin, 1995.
[16] T. Villmann, R. Der, M. Herrmann, and T. M. Martinetz, "Topology preservation in self-organizing feature maps: exact definition and measurement," *IEEE Transactions on Neural Networks*, vol. 8, no. 2, pp. 256–266, 1997.
[17] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.