Analysis of an Adversarial Approach to Blind Source Separation

Juan M. Espinoza, Romis Attux and Levy Boccato

***Abstract*— Here, we analyze the adversarial network proposed in [5], named Anica, to solve the problem of independent component analysis (ICA). Guided by a discriminator of independence, a linear autoencoder can learn to codify the observations into estimates of their independent components. This study establishes the conditions for convergence of Anica and proposes a blind criterion to select the best training epoch. Additionally, having JADE and FastICA as benchmarks, we analyze its performance by varying the number of samples and sources, and the noise variance. The obtained results indicate that Anica has a significant potential of extension to more general scenarios.**

***Keywords*— Blind source separation, Independent component analysis, Adversarial learning, Autoencoders.**

I. Introduction

Blind Source Separation (BSS) can be safely regarded as a most relevant problem related to the notion of information retrieval. Essentially, given a set of observed variables, $\mathbf{x}(t)$, which are implicitly generated by an unknown mixing process of latent variables (or sources) $\mathbf{s}(t)$, the goal in BSS is to recover these sources considering a minimum amount of a priori information [1], [2]. Different instances of this problem have been studied in the literature during the last decades [1], which vary according to the character of the mixing process. The classical scenario of BSS assumes that the mixing process is linear, instantaneous and noiseless, so that:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t), \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{n \times m}$ is the mixing matrix, n denotes the number of mixtures and m is the number of sources. In many situations, it is considered that $n = m$.

In this case, the separation process consists in designing a matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ capable of inverting the mixture system, thus yielding a set of estimates

$$\mathbf{y}(t) = \mathbf{W}\mathbf{x}(t), \tag{2}$$

which should be as similar as possible to the original sources. Fig. 1 exhibits a block diagram of the classical BSS problem.

A fundamental hypothesis explored to solve this problem is the independence of the sources, which is in the core of the well-established approach known as independent component analysis (ICA) [1]. In fact, ICA comprises a set of effective algorithms, such as FastICA and JADE [1], which seek a matrix $\mathbf{W}$ that directly or indirectly maximizes the level of independence between the estimates in $\mathbf{y}(t)$.

Juan M. Espinoza, DCA, UNICAMP, Campinas-SP, e-mail: j228562@dac.unicamp.br; Romis Attux, DCA, UNICAMP, Campinas-SP, e-mail: attux@dca.fee.unicamp.br; Levy Boccato, DCA, UNICAMP, Campinas-SP, e-mail: lboccato@dca.fee.unicamp.br.

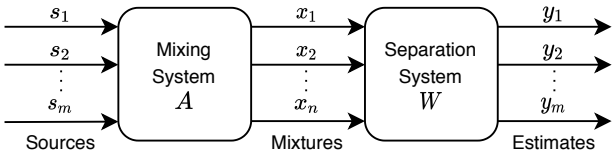

Fig. 1. Main elements involved in the classical BSS problem.

In this sense, ICA can be seen as a generalization of principal component analysis (PCA). It is pertinent to apply a decorrelation (or whitening) procedure before attempting to find the independent sources, since it reduces the search space to the set of orthogonal matrices [1].

Interestingly, the field of machine learning (ML) [4], [7] also offers a repertoire of techniques for retrieving latent variables from a collection of observations. In particular, autoencoders (AEs) represent an important approach [4], as they explore the flexibility of artificial neural networks to represent the available data in a latent space.

In simple terms, an AE is a neural network that aims to reproduce its input data $\mathbf{x}$ at its output $\hat{\mathbf{x}}$. The structure of an AE is composed of two parts: (1) an encoder function, $f_{en}(\cdot)$ and (2) a decoder $f_{de}(\cdot)$. While the encoder is responsible for mapping the input data to an internal code, $\mathbf{h} = f_{en}(\mathbf{x})$, the decoder tries to reconstruct the original data given the generated code, yielding $\hat{\mathbf{x}} = f_{de}(\mathbf{h})$. Fig. 2 shows the architecture of an AE and its components.

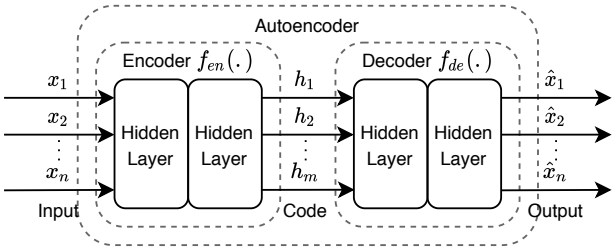

Fig. 2. General architecture of an autoencoder.

An AE is designed in such a way that its encoder indirectly learns to create a meaningful representation of the data, while the network attempts to minimize a dissimilarity measure between $\mathbf{x}$ and $\hat{\mathbf{x}}$. In this context, it is important to avoid attaining trivial solutions, i.e., when the decoder becomes a perfect inverse of the encoder, $f_{de}(\cdot) = f_{en}(\cdot)^{-1}$, without considering the information contained in the code, making it useless. For example, sparse and undercomplete AEs [4] force the encoder to retrieve latent information of

the data by imposing sparsity over the code, or a reduced dimensionality, respectively. In general, if the structure and the training approach are properly chosen, as exposed in Section II, the encoder effectively becomes a feature extractor, which provides a set of latent variables for the input pattern that still allow its reconstruction with an acceptable error.

Since AEs represent powerful options for latent variable analysis, it is pertinent to ponder whether they can be used to retrieve estimates of the sources $\mathbf{s}(t)$, given the set of mixtures $\mathbf{x}(t)$, in the BSS problem. Since AEs can implement a nonlinear generalization of PCA [4], they have sufficient flexibility to solve the classical BSS problem. Ideally, the encoder should play the role of the separation system, capturing a representation of the sources in the code, while the decoder should mimic the unknown mixing system.

However, when the training process focuses solely on minimizing the reconstruction error, the network may still learn a completely different representation in the latent space when compared to the original sources, since it is free to determine the characteristics of the latent variables during the learning process. This means that, due to its flexibility, an AE can extract features from the mixtures and still reconstruct them at the output, but without recovering the actual latent variables.

Having in view this dilemma, an inevitable question arises: is it possible to guide the training process so that the AE ends up discovering the desired latent representation? Interestingly, this question has been tackled in the recent work of Brakel and Bengio [5] with the aid of an elegant adversarial approach.

The strategy they proposed, called Anica, involves training an AE and a discriminator network in an adversarial scheme, where the discriminator pushes the AE towards creating features that are independent. The goal of the discriminator is to recognize whether its input is a vector containing independent variables or not. Simultaneously, the AE is trained to minimize the reconstruction error, as well as to minimize the performance of the discriminator network. In other words, the encoder is encouraged to obtain a code that allows the reconstruction of the input and, at the same, contains variables that are more independent until the discriminator can no longer distinguish them from samples taken from the product of the marginal distributions (which truly are composed of independent elements).

In spite of the promising results reported in [5], there are many aspects that still require further investigations concerning the behavior of Anica in the BSS problem. For instance, how does Anica scale with the number of sources, or the number of samples? Additionally, how is it affected by the presence of noise in the mixtures?

The present work aims to study the limitations and advantages of Anica in the context of the classical BSS problem. A detailed analysis of the behavior and convergence of the training algorithm is offered. Additionally, we suggest a criterion to blindly select the learning epoch in which the AE yields the best estimates of the sources. More specifically, we propose to monitor the independence between the code variables by means of the mutual information. Then, a validation of the proposed strategy is provided considering the evolution of the normalized Amari error [9] during training. We also perform a comparative analysis between Anica and other well-known ICA algorithms, such as JADE and FastICA, considering different scenarios in terms of the number of samples, the number of sources and the noise variance.

The next section is devoted to the exposition of Anica, as well as of the minor modifications we proposed. Then, in Section III, we present the performance analysis of Anica in detail. Finally, in Section IV, we bring the final considerations and perspectives for the sequence of the research.

II. Anica: Overview and Expected Convergence

The Anica model is composed of an AE and a discriminator network, which are trained according to an adversarial approach, in a similar fashion as employed in generative adversarial networks (GANs) [6]. Fig. 3 depicts the main elements of Anica [5].

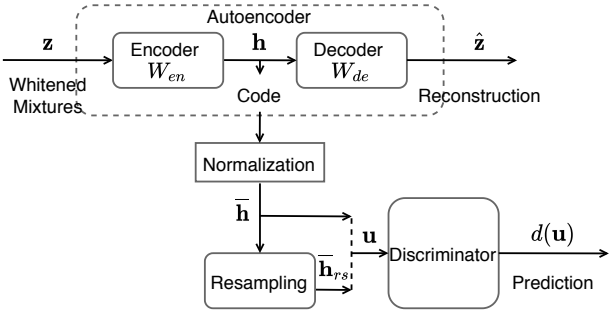

Fig. 3. The general structure of Anica model.

The AE input corresponds to the whitened mixtures, which are obtained via $\mathbf{z}(t) = \mathbf{V}\mathbf{x}(t)$, where $\mathbf{V} \in \mathbb{R}^{n \times n}$ is a whitening matrix (e.g., $\mathbf{V}$ is composed of the n eigenvectors of the autocorrelation matrix of the mixtures associated with the n largest eigenvalues). Since we are addressing the classical BSS scenario, as defined in (1), both the encoder and decoder are linear structures, characterized by the weight matrices $\mathbf{W}_{en}$ and $\mathbf{W}_{de}$, respectively.

As usual, the AE is designed to produce an output $\hat{\mathbf{z}}$ as close as possible to its input $\mathbf{z}$. Interestingly, in the Anica model, the AE also interacts with a discriminator network, which acts as an inspector that attest if the latent variables created by the encoder are, in fact, independent.

More specifically, the discriminator is a neural network trained to classify whether its input comes from the distribution of independent latent variables (positive class), or if it is a fake sample created by the encoder (negative class). Hence, in order to be properly trained, the discriminator must have access to samples composed of independent latent variables. In this context, Brakel and Bengio [5] proposed a resampling procedure to generate samples based on the available code as if they were truly drawn from the PDF corresponding to the product of the marginal PDFs of the latent variables. In simple terms, the resampling procedure shuffles the samples of every component of the code in a minibatch, separately, to break their mutual dependence, producing a new vector $\bar{\mathbf{h}}_{rs}$ that can be seen as a sample taken from the product of the marginal PDFs.

Then, a cross-entropy cost function J_D is minimized during the discriminator training, so that it learns to distinguish between input samples coming from the resampling procedure $\bar{\mathbf{h}}_{rs} \sim p_r$, or from the normalized code $\bar{\mathbf{h}} \sim p_e$, where $d(\cdot)$ denotes the class prediction:

$$J_D = -1/2 \left(\mathbb{E}_{p_r}[\log d(\bar{\mathbf{h}}_{\mathbf{rs}})] + \mathbb{E}_{p_e}[\log(1 - d(\bar{\mathbf{h}}))] \right) \quad (3)$$

On the other hand, the AE is trained not only to minimize an error measure between $\mathbf{z}$ and $\hat{\mathbf{z}}$, but also to create an internal code with mutually independent features. This is accomplished by adopting the following cost function:

$$J_{AE} = J_{AE}^R + \lambda J_{AE}^C, \quad (4)$$

where

$$J_{AE}^R = \mathbb{E}_{p_{\mathbf{z}}}[|\mathbf{z} - \hat{\mathbf{z}}|] \quad (5)$$

$$J_{AE}^C = -\mathbb{E}_{p_e}[\log d(\bar{\mathbf{h}})] \quad (6)$$

In this work, we adopted $\lambda = 0.1$ and the mean absolute error as the reconstruction cost function for (5), following the same selection by [5]. Different values for λ and other measures of error, such as the mean squared error, could be selected, but the experimental results favour the original selection. The term in (6) is minimized when the discriminator assigns the code samples to the positive class, which means that the generated code actually contains independent latent variables. Hence, by establishing this two-player game between the AE and the discriminator, the AE is forced to improve the encoder in order to fool the discriminator, while the latter becomes more competent in recognizing the code samples as fake. The adversarial approach can also be interpreted as promoting a PDF matching in the code layer [6]: the encoder is encouraged to create code samples whose joint PDF approximates a target PDF, which corresponds to the product of the marginal PDFs of the latent variables within the code.

As shown in Fig. 3, we normalize the code vector before the resampling procedure is applied, by removing the mean and dividing by the standard deviation of each variable, which helps to speed up the convergence of the discriminator. The decoder tries to reconstruct the input based on the original code $\mathbf{h}$. In Algorithm 1, we summarize the main steps involved in the training process of Anica.

As we can observe, for each minibatch, the discriminator is updated to minimize J_D over a balanced labeled dataset $\mathcal{A}$, so that it improves its ability to separate whether its input comes from the resampling procedure (and, thus, contains independent variables in $\bar{\mathbf{h}}_{rs} \sim p_r$), or from the encoder after normalization $\bar{\mathbf{h}} \sim p_e$. Then, for the same minibatch and keeping the parameters of the discriminator fixed, the AE is updated to minimize J_{AE}. The term of J_{AE} highlighted in (6) is computed by feeding the discriminator only with normalized codes $\mathbf{u} = \bar{\mathbf{h}} \sim p_e$, but labeled as if they belong to the positive class. So, by minimizing J_{AE}, the AE also adapts its parameters to generate codes that resemble samples taken from the joint distribution after resampling p_r, and, ultimately, it tends to generate independent latent variables.

A. Convergence Analysis

Based on the theorems derived in [6], we establish in Theorem 1 that the training process of Anica converges if, and only if, the distributions p_e and prs become equal, which occurs when the costs J_D and J_{AE}^C converge to $\log 2$.

Algorithm 1: Training process of Anica for BSS.

Result: code of independent components
Apply a whitening procedure to the mixtures: $\mathbf{z} = \mathbf{V}\mathbf{x}$
Initialize parameters of Anica
while *Not converged* **do**
  Sample N minibatches from the whitened dataset
  for *minibatch* $i \in [1, ..., N]$ **do**
    Obtain the codes and the outputs of the AE to i
    $\bar{\mathbf{h}} \leftarrow$ normalized codes
    $\bar{\mathbf{h}}_{rs} \leftarrow$ resampled normalized codes
    $\mathcal{A} \leftarrow \left\{ \bar{\mathbf{h}}_{rs}(\text{label } 1); \bar{\mathbf{h}}(\text{label } 0) \right\}$
    Update the discriminator to minimize J_D
    Update the autoencoder to minimize J_{AE}
  end
end

Theorem 1: Convergence of Anica is achieved if, and only if, $p_e = p_r$, and when the cost function J_D and the classification term J_{AE}^C converge to $\log 2$. In this case, the discriminator can no longer distinguish its inputs, so that $d(\cdot) = 0.5$.

Proof: As proved in [6], for a fixed AE, the discriminator learns the optimal mapping, which corresponds to

$$d_{opt}(\mathbf{u}) = \frac{p_r(\mathbf{u})}{p_r(\mathbf{u}) + p_e(\mathbf{u})} \quad (7)$$

Then, by maintaining the discriminator unchanged, the minimum value of the classification term J_{AE}^C is $\log 2$, which is obtained if, and only if, $p_e = p_r$. This can be verified by substituting (7) in (6), as shown in the sequence:

$$\begin{aligned} J_{AE}^C &= -\mathbb{E}_{p_e}[\log d_{opt}(\mathbf{u})] \\ &= \int p_e(\mathbf{u}) \log \left(\frac{p_r(\mathbf{u}) + p_e(\mathbf{u})}{p_r(\mathbf{u})} \right) d\mathbf{u} \end{aligned} \quad (8)$$

To simplify the notation, we shall omit the argument $(\mathbf{u})$ of the PDFs. Then, by introducing some additional terms, we can rewrite (8) in terms of Kullback-Leibler divergences, denoted by $D(\cdot||\cdot)$:

$$\begin{aligned} J_{AE}^C &= 2 \int \left[\frac{p_e + p_r - p_r}{2} \right] \log \left(\frac{2(p_r + p_e)/2}{p_r} \right) d\mathbf{u} \\ &= 2 \int \left[\frac{p_e + p_r}{2} \right] \log \left(\frac{\frac{p_e + p_r}{2}}{p_r} \right) d\mathbf{u} \\ &+ \log 2 \int p_e d\mathbf{u} - \int p_r \log \left(\frac{\frac{p_e + p_r}{2}}{p_r} \right) d\mathbf{u} \\ &= \log 2 + 2D \left(\frac{p_e + p_r}{2} || p_r \right) + D \left(p_r || \frac{p_e + p_r}{2} \right) \end{aligned} \quad (9)$$

Since $D(\cdot||\cdot) \geq 0$, the minimum value of J_{AE}^C is $\log 2$, which occurs when both divergences are zero, or, equivalently, when $p_r = p_e$. As a consequence, $d_{opt} = 0.5$ and, finally, $J_D(d_{opt}(\mathbf{u}))$ becomes equal to $\log 2$, concluding the proof. ■

III. Experiments and Results

In this section, we present the experimental setup and the results attained by Anica in different scenarios of the BSS problem. For each experiment, a synthetic dataset of mixtures was created from zero-mean, unit-variance sources and mixing matrices, each drawn from uniform and gaussian PDFs, respectively. The mixtures were whitened before training.

Similarly to [5], the discriminator network consisted of a multilayer perceptron containing one hidden layer with 64 ReLu neurons, and a single output neuron using logistic activation function. As already mentioned, both the encoder and decoder apply linear transformations according to square matrices $\mathbf{W}_{en}$ and $\mathbf{W}_{de}$, respectively. We adopted Xavier initialization [7] for the discriminator, while random orthogonal matrices were used for the AE. The RMSProp algorithm [7], with default parameters, was employed to train the model during 4000 epochs, considering minibatches of 1024 samples.

The performance of Anica was evaluated on each epoch through the Normalized Amari Error (NAE) [9], which measures the ability of the encoder to deliver permuted and scaled estimates of the sources. Ideally, we would like to select the training epoch, and the corresponding AE configuration, when the NAE reaches its minimum. However, it requires exact knowledge of the actual mixing matrix ($\mathbf{A}$). Therefore, in order to blindly select the "best" epoch (and the corresponding $\mathbf{W}_{en}$), we employed a shifted version of the mutual information (SMI), based on [3]. The SMI was calculated directly from the generated samples of the normalized code $\bar{\mathbf{h}}$ by estimating its marginal entropies $H(\bar{h}_i)$:

$$SMI(\bar{\mathbf{h}}) = \sum_{i=1}^{n} H(\bar{h}_i) - \log(|\det(\mathbf{W}_{en})|) \qquad (10)$$

Hence, in this work, the best epochs were selected according to the minimum value of the SMI, and the NAE was calculated for corroboration. Each experiment was repeated 10 times to compensate for the effects of random initialization, so we report the average NAE across the repetitions. Available implementations of parallel FastICA (running for 200 iterations) and JADE were also explored for the sake of comparison.

In order to illustrate the behavior of Anica, we display in Fig. 4 the evolution of the costs J_{AE}^{C} and J_{D}^{C}, which, as we can observe, converge to $\log 2$ approximately after 1700 epochs (e). Additionally, we also show the distribution of the samples related to (a) the sources, (b) the mixtures, (c) the whitened mixtures and (d) the independent estimates obtained in the code. Clearly, the encoder successfully learned a linear transformation to rotate a 2-component input set of whitened mixtures (c) into their independent estimates (d), while the reconstruction cost function is kept at a minimum (f).

The first aspect we shall analyze refers to impact of the size of the training set on the performance and convergence of Anica. Considering a scenario with $m = 9$ sources, we exhibit in Fig. 5 the average NAE obtained by Anica, FastICA and JADE as a function of the number of samples. As we can observe, increasing the size of the dataset reduces both the average NAE (a), improving the performance, and the dashed region between the minimum and maximum NAE across

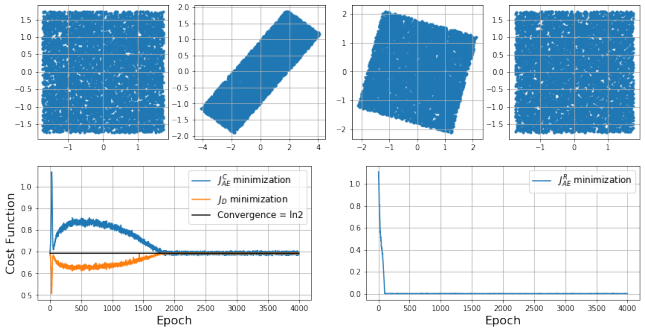

Fig. 4. Data distribution and training curves. From top left to bottom right: synthetic sources (a), mixtures (b), whitened mixtures (c), normalized code (d), convergence between J_{AE}^{C} and J_{D}^{C} (e), reconstruction cost J_{D}^{R} (f).

repetitions, making the model more robust. Nonetheless, the NAE values associated with Anica are above those attained by FastICA and JADE (b).

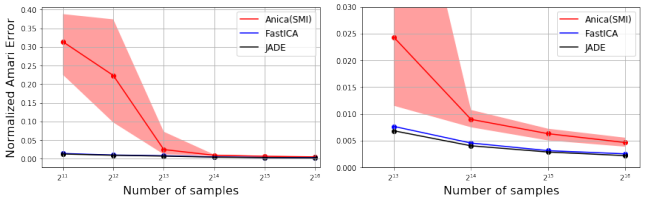

Fig. 5. Average NAE w.r.t the number of training samples. From left to right: NAE improves with more samples (a). Anica becomes more robust (b).

The effect on convergence is shown in Figure 6, where the cost functions converge for datasets of 8192 (b) and larger (c), but fail to converge when presented to fewer samples (a), which helps to explain the previous results and confirms that the size of the dataset is critical for Anica. Additionally, Figure 7 shows how the NAE behaves as training progresses and convergence is achieved (a), while the SMI is minimized, indicating independence. As we can notice, the SMI captures almost every NAE value fluctuation, which motivates its use as a valid blind selection criterion (b) in an early stopping criterion. Based on these results, all the subsequent experiments considered 16384 samples.

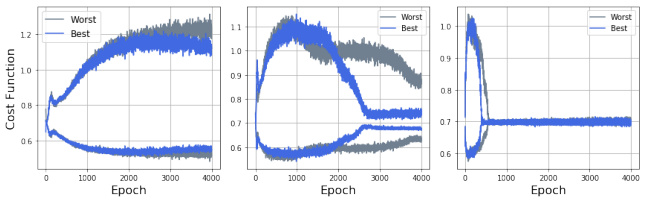

Fig. 6. Worst and best Convergence w.r.t the number of training samples. From left to right: Convergence for 2^{11} (a), for 2^{13} (b), for 2^{16} samples (c).

Secondly, we analyzed how Anica is influenced by the number of sources. Fig. 8 shows the NAE values for n varying from 2 to 9. In general, the performance of Anica deteriorates as the number of sources is increased (a). Nonetheless, we can see that Anica performs better than FastICA and JADE in the cases up to 6 sources (b). It is important to mention that it was necessary to repeat training once for the 3-component dataset in order that the algorithm converges 10 times.

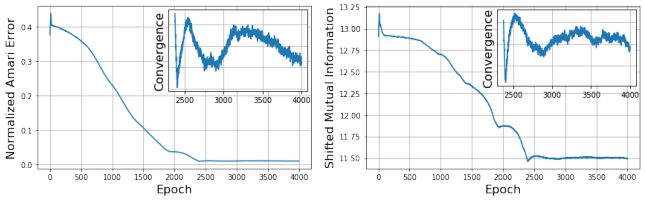

Fig. 7. Metrics w.r.t the epoch. From left to right: NAE (a), SMI (b).

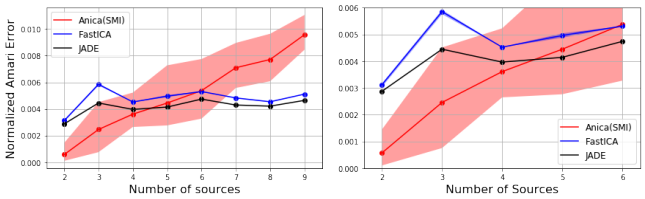

Fig. 8. Average NAE w.r.t the number of sources. From left to right: NAE increases with more components (a). Anica outperforms JADE and FastICA (b).

Finally, we assessed how Anica behaves when the mixtures include noise. So, we generated samples of white Gaussian noise, with variance σ_n^2, adding them to the linear mixtures of 9 sources. The obtained NAE values are presented in Fig. 9 w.r.t. the Signal-To-Noise Ratio $SNR_{dB} = 10\log(\frac{\sigma_s^2}{\sigma_n^2})$, considering that all sources are unit-variance. As we can observe, Anica is capable of yielding independent estimates of the sources from noisy mixtures, but it also is more susceptible to initialization, so its NAE varies in a wider range for $SNR_{dB} = 0$ (a). However, as the noise is reduced, Anica delivers better results with less variations and much closer to FastICA and JADE (b). Fig. 10 shows that for $SNR_{dB} = 0$ (a), not all repetitions converge, which confirms that retrieving independent estimates of the sources is more difficult for noisy datasets. Once the SNR is increased to 2.5 dB (b), all repetitions converged, albeit with some delay, so better results could be obtained. Finally, for $SNR = 20$ dB (c), the effect of noise on convergence is not relevant.

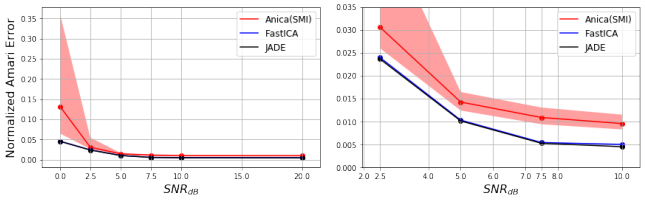

Fig. 9. Average NAE w.r.t SNR_{dB}. From left to right: Anica improves as noise decreases (a). Anica is more competitive to FastICA and JADE (b).

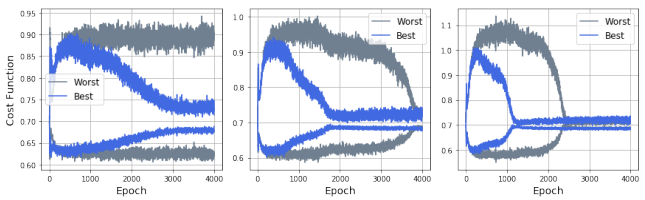

Fig. 10. Worst and best convergence w.r.t SNR_{dB}. From left to right: Convergence for $SNR_{dB} = 0$ (a), for 2.5 (b), for 20 (c).

IV. Conclusions

Anica has proven to be an effective algorithm for blind source separation and, as our experiments show, is robust enough to deal with challenging scenarios. In the cases of many sources and a noisy dataset, its performance is comparable to that of well-established algorithms, such as FastICA and JADE.

Feeding more data to Anica proves to be crucial for convergence and the generation of independent estimates. Also, it helps to mitigate problems related to initialization and the selection of hyperparameters. However, this can be a limiting factor when collecting more samples is difficult or infeasible. Notwithstanding, even when the dataset is relatively small, training for more epochs and reducing the size of the minibatch may lead to an adequate separation matrix in the encoder.

The adoption of the shifted mutual information (SMI) not only proved to be an useful criterion for blindly selecting an epoch with a corresponding low NAE, but it also confirms the success of the AE to generate a code consisting of independent variables, rather than just learning a trivial solution for reconstruction.

Finally, it is important to highlight that Anica offers a flexible architecture that can be straightforwardly modified to deal with other BSS scenarios, such the underdetermined case ($n < m$), as well as those involving convolutive mixtures and nonlinear models. It is our belief that these cases will benefit, to a more significant extent, from the generality of the discussed approach. This is a key topic for future investigations.

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior - Brasil (CAPES) - Finance Code 001, and also by CNPq (308811/2019-4).

References

[1] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. Wiley-Interscicence, a John Wiley & Sons, Inc, 2001.

[2] P. Comon, and C. Jutten, *Handbook of blind source separation*. Elsevier Academic Press, 2010.

[3] J. Romano, R. Attux, C. Cavalcante, and R. Suyama, *Unsupervised signal Processing: Channel Equalization and Source Separation*. CRC Press, 2018.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[5] P. Brakel, and Y. Bengio, *Learning independent features with adversarial nets for non-linear ica*. arXiv preprint arXiv:1710.05050, 2017.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozairy , A. Courville, and Y. Bengio, *Generative adversarial nets*. NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems, vol. 2, pp. 2672–2680, 2014.

[7] A. Gerón, *Hands-on machine learning with Scikit-Learn and TensorFlow*. O'Reilly Media, 2017.

[8] P. Comon, *Independent Component Analysis, a new concept?*. Signal Processing, Elsevier, 1994.

[9] S. Amari, A. Cichocki, and H. H. Yang, *A new learning algorithm for blind signal separation*. Advances in neural information processing systems, 1996.