Independent Component Analysis using Reinforcement Learning

Ying Wu
School of Computing
University of Paisley

Abstract

Recently Fyfe has used the REINFORCE algorithm to create a variety of topology-preserving mappings and to create valid projections for principal component analysis and canonical correlation analysis. In this paper, we extend his results to the case of independent component analysis. We illustrate the basic method with a number of different reward functions and a number of artificial and real data sets.

1. Introduction

Independent Component Analysis (ICA) has become, in recent years, a well established data analysis technique for data mining. The goal of ICA is to recover independent signals given only a set of observations that are unknown linear mixtures of the independent signals. In contrast to second-order statistical-based transformations such as Principal Component Analysis (PCA), ICA is essentially based on higher-order information, while attempting to make the signals as independent as possible.

ICA has been studied by many researchers in statistical signal processing and artificial neural networks. Girolami[1] developed an ICA network using anti-Hebbian learning; Almeida[2] designed an ICA network based on Mutual Information. However, these methods are a kind of 'deterministic' optimization, where a prespecified learning strategy is required.

Reinforcement learning (RL) can be thought as an intermediate learning method that is distinct from supervised learning and unsupervised learning. There is no presentation of input-output pairs in the RL machine, but the RL machine gets reward from the external environment depending on how well it has done with an exploratory action and it must choose actions that tend to increase the long-term sum of values of rewards.

The general framework of reinforcement learning encompasses problems such as function optimization, learning control, or even clustering. Recently, Fyfe [3] has successfully put exploratory projection methods into the framework of reinforcement learning. In this paper, we demonstrate the combination of ICA learning with reinforcement learning, in particular, immediate reward reinforcement learning, with a Gaussian learner. A family of reward functions has been laid out corresponding to the different ways of solving the ICA problem. Some experimental results show the accuracy of ICA with reinforcement learning.

2. Independent Component Analysis by immediate reward reinforcement learning

In this paper, we consider ICA as the problem of transforming a set of observations

$\mathbf{x} = \{x_1, x_2 \ldots, x_M\}$ that are the result of a linear mixing of statistically independent sources $\mathbf{s} = \{s_1, s_2 \ldots, s_M\}$ by $\mathbf{x} = A\mathbf{s}$, into several components that are statistically independent by $\mathbf{y} = W\mathbf{x}$. We will show how the reinforcement algorithm can be used in ICA networks.

A common principle for ICA is to make sure all components are as non-Gaussian as possible. According to the central limit theorem, the mixed observations become more Gaussian than any of the independent sources, so we can just measure the distribution of $W\mathbf{x}$ and find a W that maximizes the non-Gaussianity of $W\mathbf{x}$. A popular way to measure the nongaussianity of a vector is using kurtosis $kurt(y)$, which is the fourth-order cumulative of a random variable $\mathbf{y}$. We define it by

$$kurt(y) = E\{y^4\} - 3\{E\{y^2\}\}^2 \quad (1.1)$$

So when y is normalized so that its variance is equal to one, $E\{y^2\} = 1$, the kurtosis is simplified to $E\{y^4\} - 3$. For a Gaussian random variable y, the kurtosis is zero and the random variables with positive kurtosis are super-Gaussian, and those with negative kurtosis are sub-Gaussian. So the ICA network is used to make the kurtosis of the final components as non-zero as possible, which is as non-Gaussian as possible.

Following Williams[4], we use stochastic units drawn from a particular distribution to sample a variety of network weights. We still use the Gaussian learner as the specific type of learner as having been used in [3]. So the weights W are all drawn from $N(\mathbf{m}, \beta^2 I)$, the Gaussian distribution with mean $\mathbf{m}$ and variance β^2, and so we update the parameters of the sampled distribution by

$$\Delta\mathbf{m}_i = \alpha_m (r_i - \bar{r}_i) \frac{\mathbf{w}_i - \mathbf{m}_i}{\beta_i^2} \quad (1.2)$$

$$\Delta\beta_i = \alpha_\beta (r_i - \bar{r}_i) \frac{\| \mathbf{w}_i - \mathbf{m}_i \|^2 - \beta_i^2}{\beta_i^3} \quad (1.3)$$

The sets of weights connecting to different output neurons are completely independent. Different from [5], for prewhitened observations, our ICA network will maximize the absolute values of the kurtosis of each output component

$$J_{kurt}(y_i) = \sum_{t=1}^{N} \left| cum(y_i^4(t)) \right| = \sum_{t=1}^{N} \left| E\{y_i^4(t)\} - 3E^2\{y_i^2(t)\} \right| \quad (1.4)$$

where $i = 1, 2 \ldots M$

We use

$$r_i = \sqrt{\left| kurt(w_i^T x) \right|} \quad (1.5)$$

as the reward function in our ICA algorithm.

2.1 Deflationary orthogonlization

In our ICA algorithm, to estimate more than one independent component, if the sets of weights connecting to M output neurons are trained consistently, it is possible that they converge to the same maxima. For example, we create a 2-dimensional artificial data set $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)$ of 1000 samples, where

$\mathbf{s}_1 = \sinh(\mathbf{t}_1), \mathbf{s}_2 = \tanh(\mathbf{t}_2)$ and $\mathbf{t}_1, \mathbf{t}_2$ are vectors whose elements are from Gaussian distribution in $t_1, t_2 \in N(0,1)$. The mixture matrix is specified as $A = \begin{pmatrix} 0.4977 & 0.6640 \\ -0.6948 & 0.3086 \end{pmatrix}$. The mixed observations, $\mathbf{x} = A\mathbf{s}$ is prewhitened to $\mathbf{z}$ with the same experimental environment, the final result may be as in Table 1 in which it is obvious that both sets of weights converge to recover the original signal that is most super-Gaussian.

If we recall the reward function (1.5), the higher the reward function is, the more reward the ICA network will get, and so the value of reward function will keep increasing provided the network is identifying the most kurtotic signal. Thus, given all the sets of weights being trained consistently, the reward function (1.5) can guarantee that all the recovered components are non-Gaussian, but the ICA network tends to find the same original signal that has largest kurtosis.

.

The single reward function acts like a magnetic trap attracting all the greedy reward-seekers to identify the most kurtotic signal. We also conjecture this will also happen for some other algorithm with reinforcement learning.

	Kurtosis 1	**Kurtosis 2**
Original signals	12.9983	**402.8734**
Mixed observations	185.3212	20.1368
Recovered ICs	**402.7710**	**390.6601**

Table 1: The kurtosis of the original signals, mixed observations and recovered independent components (ICs). Note that only one IC is identified.

Therefore, in our ICA network, we perform the same reinforcement learning algorithm M times on different sets of weights in sequence and we orthogonalize the vectors $\mathbf{w}_1, \mathbf{w}_2 \ldots, \mathbf{w}_M$ after each iteration, using the Gram-Schmidt method[6], i.e. we subtract the projections $(\mathbf{w}_p^T \mathbf{w}_j)\mathbf{w}_j$, $j = 1, 2 \cdots, p-1$ from the current set of weights, $\mathbf{w}_p$.

Thus sets of weights connecting to different output neuron are trained in sequence and for $i \geq 2$, the weights are changed using

$$\mathbf{w}_i \leftarrow \mathbf{w}_i - \sum_{j=1}^{i-1} (\mathbf{w}_i{}^T \mathbf{w}_j)\mathbf{w}_j \qquad (1.6)$$

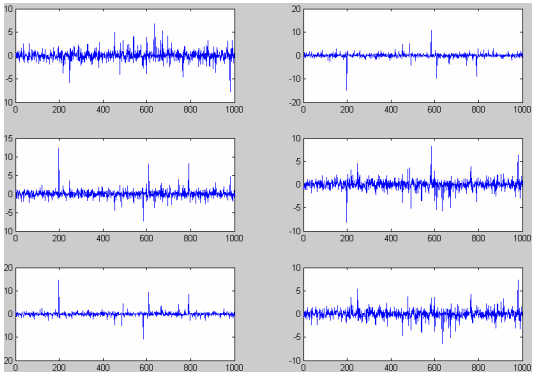

Figure 1: Top: the original data set. Centre: the mixed observations to the ICA network. Bottom: the ICs recovered by the network

We see that with deflationary learning, the ICA network has identified all the independent components in Figure 1 and we see a smooth convergence of the reward functions in Figure

2. From Table 2, we can see that even if the difference in kurtosis between the original signals is large, the ICA network can still identify ICs accurately.

We can also evaluate the performance of our ICA algorithm using the measurement of Amari error [7], where we compare the final W and V by

$$d(V,W) = \frac{1}{2m}\sum_{i=1}^{m}\left(\frac{\sum_{j=1}^{m}|a_{ij}|}{\max_j |a_{ij}|}-1\right)+\frac{1}{2m}\sum_{j=1}^{m}\left(\frac{\sum_{i=1}^{m}|a_{ij}|}{\max_i |a_{ij}|}-1\right)$$

, where $a_{ij} = (VW^{-1})_{ij}$.

The Amari error is zero when W and V represent the same components. In this experiment, the Amari error is 0.06.

However, several issues arise from these experiments which are now discussed under the headings of pre-whitening, and alternative reward functions.

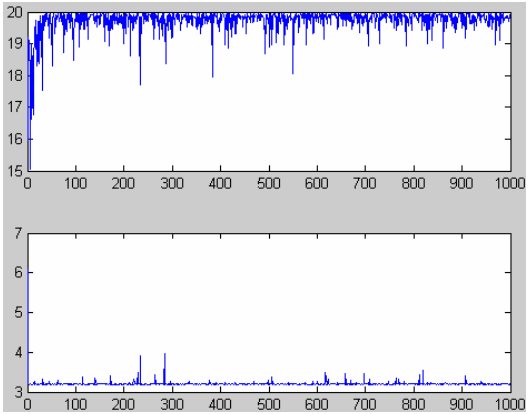

Figure 2: The convergence of the reward function for both of sets of weights

	Kurtosis 1	**Kurtosis 2**
Original signals	16.4941	864.4390
Mixed observations	410.6288	31.5702
Recovered ICs	865.2676	16.4546

Table 2: The kurtosis of the original signals, mixed observations and recovered independent components (ICs)

2.2 Pre-whitening

Prewhitening is useful preprocessing strategy in ICA. Before performing ICA algorithm, we transform the mixed observations $\mathbf{x}$ linearly into a new vector $\mathbf{z}$ whose components are uncorrelated and whose variances equal unity. So the covariance matrix of $\mathbf{x}$ equals the identify matrix, $E\{\mathbf{z}\mathbf{z}^T\} = I$. Although uncorrelatedness is weaker than independence and prewhitening only finds the ICs to an orthogonal transformation, it is still helpful in that we can search for the mixture matrix W in the space of orthogonal matrices. Note particularly that when we perform ICA network using deflationary orthogonlization, prewhitening is also necessary.

The common way of prewhitening is to use the eigen-value decomposition of the covariance matrix $E\{\mathbf{x}\mathbf{x}^T\} = EDE^T$, where E contains the eigenvectors of $E\{\mathbf{x}\mathbf{x}^T\}$ and D is a diagonal matrix of the eigenvalues, $D = diag\{d_1,\ldots,d_m\}$. In our algorithm, we use $V = (D^{-1/2})^T E^T$ as the whitening matrix and the observations $\mathbf{x}$ is transformed using

$$\mathbf{z} = V\mathbf{x} \qquad (1.7)$$

2.3 Alternative Reward Functions

In this section, we will discuss another important way in which to measure non-Gaussianity. Negentropy is a concept from information theory. The basic idea is that since a Gaussian variable has the largest entropy among all random variables of equal variance, we can use entropy to measure non-Gaussianity.

According to [8], negentropy is defined as $J(y) = H(y_{Gaussian}) - H(y)$. A good way to approximate negentropy is

$$J(y) \propto [E\{G(y)\} - E\{G(\upsilon)\}]^2 \qquad (1.8)$$

where $G(y)$ is a nonquadratic function that does not grow fast and υ is a standardized Gaussian random variable. We define

$$r = \left\| G(y) - G(\upsilon) \right\| \qquad (1.9)$$

as the reward function, which means the greater the approximation of negentropy is, the more reward ICA network will get. We also use a family of $G(y)$ [8], (1.10) - (1.14) to produce a family of reward functions:

$$G_1(y) = \log(\cosh(y)) \qquad (1.10)$$

$$G_2(y) = -\exp(-y^2/2) \qquad (1.11)$$

$$G_3(y) = \tanh(y) \qquad (1.12)$$

$$G_4(y) = y\exp(-y^2/2) \qquad (1.13)$$

$$G_5(y) = y^3 \qquad (1.14)$$

To examine the nonquadratic functions (1.10) - (1.14), we use a 2-dimensional real data set, 'chirp' and 'gong', provided by Matlab. All the experiments have been performed with the same number of iterations.

	Kurtosis 1	**Kurtosis 2**
Original signals	3.2958	3.9589
Mixed observations	3.5871	3.8613
RF_1	3.4228	3.9800
RF_2	3.2961	3.9404
RF_3	3.2865	3.9565
RF_4	3.3647	3.9558
RF_5	3.4549	3.8651

Table 3: Kurtosis by a family of reward functions based on (1.9)

	Amari error
RF	0.1947
RF_1	0.6954
RF_2	0.4570
RF_3	0.2179
RF_4	0.4188
RF_5	0.7280

Table 4: The Amari errors for different $G(y)$, (1.10) - (1.14). **RF** corresponds to the original reward function (1.5)

We see that all of the reward functions can identify ICs as shown in Figure 3, but the performance with different reward functions is not the same, as shown in Table 3. Table 4 also reflects this situation. We can compare the results with that using the original reward function (1.5). Experimentally, we find the performance can be improved by adjusting parameters of the ICA network, which is not the point we want to discuss here.

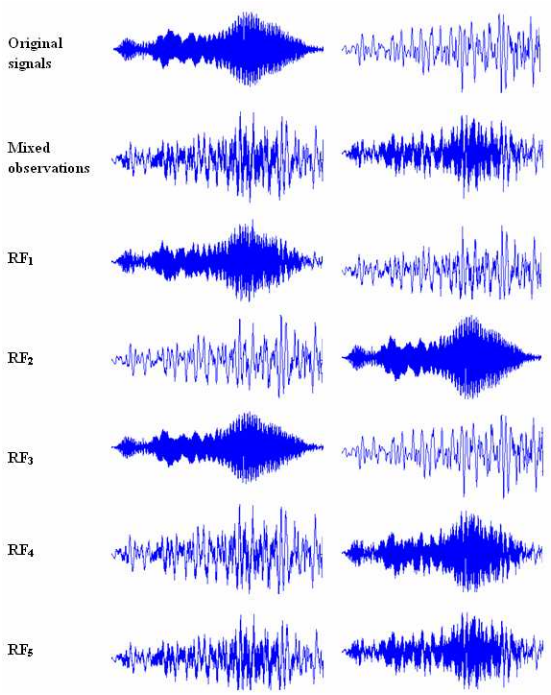

Figure 3: The original signals, mixed observations and recovered ICs by the reward functions as identified in the left col.

2.4 Simulation

We have combined the ICA network with reinforcement learning in former sections in different ways. In this section, to demonstrate how well our ICA network works, we use 3-D real data set in which the original signals are pieces of records from different humans speaking. All of the records are collected at 8000Hz, 8bits and 1 channel, as shown in the top row of Figure 4. The real data set is mixed linearly as shown in the middle row of Figure 4, which partly simulates the situation of a cock-tail party. The reason we use 'partly' here is that for a real cock-tail party, the observations could be mixed non-linearly and may include reverberations.

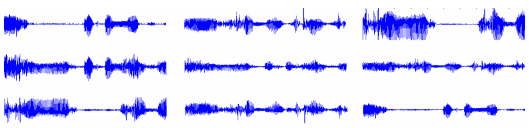

Figure 4: ICA network with using real data set. Top: the original data set. Centre: the mixed observations to the ICA network. Bottom: the ICs recovered by the network

It is clear that all independent components have been separated. Also, in experiments, we find that using real data usually requires a larger number of iterations than using artificial data and the learning rate should be small so that learning processing can be performed smoothly and efficiently. Table 5 is the correlation between the original sources and recovered signals after 50000 iterations. We show partial convergence in Table 6 after 10000 iterations.

0.0010	**0.9964**	0.0837
0.9927	0.0468	0.0973
0.0835	0.0790	**0.9921**

Table 5: Correlation between the original sources and recovered signals after 50000 iterations

0.0279	**0.9889**	0.1824
0.9929	0.1286	0.0633
0.0473	0.1414	**0.9817**

Table 6: Correlation between the original sources and recovered signals after 10000 iterations

3. Non-linear PCA with reinforcement learning

It has been proved that there is relationship between non-linear PCA and ICA. In this section, we will combine non-linear PCA network with reinforcement learning to separate independent components.

Given the observations have been whitened and so matrix W is orthogonal, [9] has proved that the criterion of non-linear PCA

$$J(W) = E\{\left\| x - Wg(W^T x)\right\|^2\} \quad (1.15)$$

has the relationship with the contrast function of ICA

$$J_{kurt}(W) = \sum_{i=1}^{n} E\{y_i^4\}. \qquad (1.16)$$

So maximization/minimization criterion (1.15) equals to maximizing/minimizing the kurtosis of ICs.

We use

$$r = \frac{1}{1+\exp(\gamma \left\| x - W \tanh(W^T x) \right\|^2)} \qquad (1.17)$$

as the reward function.

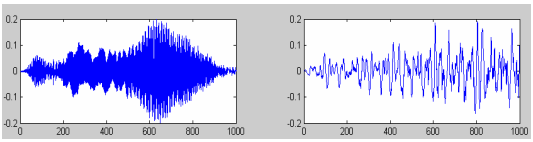

Figure 5: ICs recovered by non-linear PCA network with reinforcement learning

To examine reward function (1.17), we use the same real data set as was used in Section 2.3, and we can see that the non-linear PCA network has separated the mixed observations into ICs correctly as the ICA network has done. The Amari error is 0.1303. Figure 5 shows the recovered components we get.

4. Conclusion

We have investigated a new ICA network with reinforcement learning in this paper, in particular, immediate reward reinforcement learning with a Gaussian learner. We first took the basic idea that all components should be as non-Gaussian as possible. Kurtosis has been used in the reward function to measure non-Gaussianity of the final components. We have shown the results on a 2-dimensional artificial data set. We also have discussed the necessary of pre-whitening and deflationary orthogonlization in our algorithm.

Furthermore, we have discussed another important way in which to measure non-Gaussianity, negentropy. We have put forward a family of reward functions corresponding to different nonquadratic functions. We have used a real data set to examine the performance of these reward functions.

Finally, we have combined non-linear PCA network with reinforcement learning to perform ICA. The results are clear.

---

[1] M. Girolami. *Self-organising Neural-Networks for signal separation.* PhD thesis, University of Paisley, 1998.

[2] L.B. Aimeida. *MISEP- Linear and Nonlinear ICA Based on Mutual Information.* Journal of Machine learning Research, 2002

[3] C. Fyfe. *Reinforcement Learning for Unsupervised Data Exploration.* Technical Report, University of Paisley, 2006

[4] R.J. Williams and J. Peng. *Function Optimization Using Connectionist Reinforcement Learning Algorithms*. Connection Science, 3, pp, 1991

[5] E. Oja and J. Karhunen. *Signal Separation by Nonlinear Hebbian Learning.* In Proceedings IEEE ICNN 95, 83—87,1995

[6] Kun Zhang and Lai-Wan Chan. *Dimension Reduction as a Deflation Method in ICA.* IEEE, 1997

[7] Amari, A.Cichocki, and H.H.Yang. *A new learning algorithm for bliend signal separation.* Advanceds in Nerual Information Processing Systems, 8. 1996

[8] Aapo Hyvarinen, Juha Karhunen and Erkki Oja. *Independent Component Analysis, pp182-pp185*. Wiley, 2001

[9] J. Karhunen, P. Pajunen, and E. Oja. *The nonlinear PCA criterion in blind souce separation: Relations with other approaches.* Neurocomputering, 22:5-20, 1998