Neural Networks 21 (2008) 847–855

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2008 Special Issue

Two forms of immediate reward reinforcement learning for exploratory data analysis

Ying Wu [a], Colin Fyfe [a,*], Pei Ling Lai [b]

[a] Applied Computational Intelligence Research Unit, The University of the West of Scotland, Scotland, United Kingdom
[b] Southern Taiwan University, Tainan, Taiwan

ARTICLE INFO

Article history:
Received 15 January 2008
Received in revised form
18 April 2008
Accepted 17 June 2008

Keywords:
Reinforcement learning
Exploratory data analysis

ABSTRACT

We review two forms of immediate reward reinforcement learning: in the first of these, the learner is a stochastic node while in the second the individual unit is deterministic but has stochastic synapses. We illustrate the first method on the problem of Independent Component Analysis. Four learning rules have been developed from the second perspective and we investigate the use of these learning rules to perform linear projection techniques such as principal component analysis, exploratory projection pursuit and canonical correlation analysis. The method is very general and simply requires a reward function which is specific to the function we require the unit to perform. We also discuss how the method can be used to learn kernel mappings and conclude by illustrating its use on a topology preserving mapping.

1. Introduction

Reinforcement learning describes a group of techniques for parameter adaptation based on a methodology which envisages an agent making an exploratory investigation of its environment with a view to identifying the optimal strategy for actions within the environment: optimal is defined in terms of the reward which an agent can gain from taking actions in the environment. It is often thought of as lying between the extremes of supervised learning in which the best action to take is known, and the parameters are adjusted to make this more likely in future, and unsupervised learning in which the learning algorithm must self-organise the parameters in order to perform the specific exploratory data analysis task at hand.

However there has always been a stream of research which has described methods of performing supervised learning tasks using reinforcement learning methods (Williams, 1992) and more recently, (Ma & Likharev, 2007). A somewhat less prominent stream has been using reinforcement learning methods for unsupervised learning tasks (Likas, 2000). The REINFORCE method of Williams (1992) has recently been used for kernel projection techniques (Fyfe & Lai, 2007) and for topology preserving mappings (Lai & Fyfe, 2007). In this paper, we investigate the reinforcement learning methods of Williams (1992) applied to Independent Component Analysis and the method of Ma and Likharev (2007) for general unsupervised projection techniques. This paper extends the work presented at ICANN2007 (Wu, Fyfe, & Lai, 2007).

2. Immediate reward reinforcement learning

Williams and Pong (1991) and Williams (1992) investigated a particular form of reinforcement learning in which reward for an action is immediate which is somewhat different from mainstream reinforcement learning (Kaelbling, Littman, & Moore, 1996; Sutton & Barto, 1998). Williams (1992) considered a stochastic learning unit in which the probability of any specific output was a parameterised function of its input vector, $\mathbf{x}$. For the ith unit, this gives

$$P(y_i = \zeta | \mathbf{w}_i, \mathbf{x}) = f(\mathbf{w}_i, \mathbf{x}) \tag{1}$$

where, for example,

$$f(\mathbf{w}_i, \mathbf{x}) = \frac{1}{1 + \exp(-\|\mathbf{w}_i - \mathbf{x}\|^2)} \tag{2}$$

$\mathbf{w}_i$ is used here and throughout this paper as the vector of parameters used by the ith agent. Williams (1992) considers the learning rule

$$\Delta w_{ij} = \alpha_{ij}(r_{i,\zeta} - b_{ij}) \frac{\partial \ln P(y_i = \zeta | \mathbf{w}_i, \mathbf{x})}{\partial w_{ij}} \tag{3}$$

where α_{ij} is the learning rate, $r_{i,\zeta}$ is the reward for the unit outputting ζ and b_{ij} is a reinforcement baseline which in the following we will take as the reinforcement comparison, $b_{ij} = \bar{r} = \frac{1}{K}\sum r_{i,\zeta}$ where K is the number of times this unit has output ζ.

* Corresponding author.
E-mail addresses: ying.wu@paisley.ac.uk (Y. Wu), colin.fyfe@paisley.ac.uk (C. Fyfe), pei_ling_lai@hotmail.com (P.L. Lai).

0893-6080/$ – see front matter
doi:10.1016/j.neunet.2008.06.013

Williams (1992, Theorem 1) shows that the above learning rule causes weight changes which maximise the expected reward.

Williams (1992) gave the example of a Bernoulli unit in which $P(y_i = 1) = p_i$ and so $P(y_i = 0) = 1 - p_i$. Therefore

$$\frac{\partial \ln P(y_i)}{\partial p_i} = \begin{cases} -\dfrac{1}{1-p_i} & \text{if } y_i = 0 \\ \dfrac{1}{p_i} & \text{if } y_i = 1 \end{cases} = \frac{y_i - p_i}{p_i(1-p_i)}. \tag{4}$$

Likas (2000) applies the Bernoulli model to (unsupervised) clustering with

$$p_i = 2(1 - f(\mathbf{w}_i, \mathbf{x})) = 2\left(1 - \frac{1}{1 + \exp(-\|\mathbf{w}_i - \mathbf{x}\|^2)}\right). \tag{5}$$

The environment identifies the p_{i^*} which is maximum over all output units and y_{i^*} is then drawn from this distribution. Rewards are given such that

$$r_i = \begin{cases} 1 & \text{if } i = i^* \text{ and } y_i = 1 \\ -1 & \text{if } i = i^* \text{ and } y_i = 0 \\ 0 & \text{if } i \neq i^*. \end{cases} \tag{6}$$

This is used in the update rule (3) with $b_{ij} = 0, \forall i, j$ to give

$$\Delta w_{ij} = \alpha r_i (y_i - p_i)(x_j - w_{ij}) \tag{7}$$

$$= \alpha |y_i - p_i| (x_j - w_{ij}) \text{ for } i = i^* \tag{8}$$

where x_j is the jth element of $\mathbf{x}$ and w_{ij} is the corresponding jth element of $\mathbf{w}_i$. This rule is shown to perform clustering of the data set. Such methods have recently been used for kernel projection techniques (Fyfe & Lai, 2007) and for topology preserving mappings (Lai & Fyfe, 2007).

3. Projection with immediate reward learning

In this section, we apply immediate reward reinforcement learning to Independent Component Analysis. Under the framework of reinforcement learning, we consider that each agent is deemed to be taking actions in an environment that consists of data to be explored, in order to maximise the reward in this environment. Each agent has a set of parameters, $\mathbf{w}_i$, which are samples from $\mathcal{N}(\mathbf{m}_i, \beta_i^2\mathbf{I})$, the Gaussian distribution with mean $\mathbf{m}_i$ and variance β_i^2. Although different projection methods have their own objective functions, they share the common property that the reward function determines how well actions have been chosen according to the states in the environment. We illustrate this on the Independent Component Analysis problem for which a number of criteria have been developed.

3.1. Independent component analysis

Independent Component Analysis (ICA) has become, in recent years, a well established data analysis technique for data mining (see Hyvarinen, Karhunen, and Oja (2001), Almeida (2003) and Girolami (1998)). ICA defines a generative model for the observed multivariate data, which is typically given as a large database of samples. In the model, the data variables are assumed to be linear mixtures of some unknown latent variables, and the mixing system is also unknown. The latent variables are assumed to be non-Gaussian and mutually independent, and they are called the independent components of the observed data. These independent components (ICs), also called sources, can be found by ICA.

In detail, we consider ICA as the problem of transforming a set of D-dimensional observations $\mathbf{x}_1, \ldots, \mathbf{x}_N$ that are the result of a linear mixing of statistically independent sources $\mathbf{s}_1, \ldots, \mathbf{s}_N$ by $\mathbf{x} = \mathbf{A}\mathbf{s}$, into several components that are statistically independent by $\mathbf{y} = \mathbf{W}\mathbf{x}$. The independent sources $\mathbf{s}_j$ are often defined as latent variables, which means that they cannot be observed directly. At the same time, the mixing matrix $\mathbf{A}$ is also assumed to be unknown. Thus all we observe are the random variables $\mathbf{x}_i$, and we must estimate both the independent components $\mathbf{s}_i$ and the mixing matrix.

ICA can also be seen as an extension to principal component analysis. Independent components are assumed statistically independent, while PCA aims to find a subpace in which the principal components are uncorrelated, which is a weaker form of independence. When performing ICA, *whitening* is frequently used as a preprocessing step in ICA to give the ICs up to an orthogonal transformation. Consequently, before performing an ICA algorithm, we first linearly transform the mixed data set $\mathbf{x}$ by multiplying it by some matrix $\mathbf{V}$,

$$\mathbf{z} = \mathbf{V}\mathbf{x} \tag{9}$$

so that the new vector $\mathbf{z}$ has components that are uncorrelated and whose variances equal unity. V may, for example, be found by Principal Component Analysis of the data set. Thus the covariance matrix of $\mathbf{z}$ equals the identity matrix, $E\{\mathbf{z}\mathbf{z}^T\} = I$. Although uncorrelatedness is weaker than independence and prewhitening only finds the ICs up to an orthogonal transformation, it is still helpful in that we can search for the demixing matrix W in the space of orthogonal matrices.

3.2. Reinforcement learning applied to ICA

The most common principle for ICA is to make all components as non-Gaussian as possible. Thus, we optimise the demixing matrix $\mathbf{W}$ by measuring the kurtosis of the distribution of $\mathbf{W}\mathbf{x}$ where each row $\mathbf{w}_i$ of $\mathbf{W}$ corresponds to one stochastic unit. Given the prewhitened observations, we wish to maximise the absolute values of the kurtosis of each output component and the reward function is thus defined as

$$r_i = |\mathtt{kurt}(\mathbf{w}_i^T\mathbf{x})|. \tag{10}$$

As we have stated above, the optimization of the reward function can be related to the update vector in weight space, $E\{\Delta\mathbf{W}|\mathbf{W}\}$. Each component $\mathbf{w}_i$ is sampled from a Gaussian distribution, $\mathbf{w}_i \sim \mathcal{N}(\mathbf{m}_i, \beta_i^2)$ so that we now have two parameters to learn for each component, its mean $\mathbf{m}_i$ and variance β_i^2. Thus since

$$\Delta w_{ij} = \alpha_{ij}(r_{i,\xi} - b_{ij}) \frac{\partial \ln P(y_i = \xi | \mathbf{w}_i, \mathbf{x})}{\partial w_{ij}}$$

and we have

$$\ln P(y_i = \xi | \mathbf{m}_i, \beta_i, \mathbf{x}) \propto -\frac{(\mathbf{w}_i - \mathbf{m}_i)^2}{\beta_i^2}. \tag{11}$$

$$\text{Then } \Delta\mathbf{m}_i = \alpha_m(r_i - \bar{r}_i) \frac{\partial \ln P(y_i = \xi | \mathbf{m}_i, \beta_i^2, \mathbf{x})}{\partial \mathbf{m}_i} \tag{12}$$

$$\Delta\beta_i = \alpha_\beta(r_i - \bar{r}_i) \frac{\partial \ln P(y_i = \xi | \mathbf{m}_i, \beta_i^2, \mathbf{x})}{\partial \beta_i}. \tag{13}$$

Note that we use $\bar{r}_i$ since we will later use multiple agents searching to maximise their own individual rewards. We thus update the parameters of the sampled distribution for each stochastic unit with the rules

$$\Delta\mathbf{m}_i = \alpha_m(r_i - \bar{r}_i) \frac{\mathbf{w}_i - \mathbf{m}_i}{\beta_i^2} \tag{14}$$

$$\Delta\beta_i = \alpha_\beta(r_i - \bar{r}_i) \frac{\|\mathbf{w}_i - \mathbf{m}_i\|^2 - \beta_i^2}{\beta_i^3} \tag{15}$$