Evol. Intel. (2008) 1:159–169
DOI 10.1007/s12065-008-0012-x

RESEARCH PAPER

Exploratory data analysis with artificial immune systems

Ying Wu · Colin Fyfe

Received: 28 November 2007 / Revised: 2 April 2008 / Accepted: 3 April 2008 / Published online: 26 April 2008

Abstract We use a modified version of the CLONALG algorithm to perform exploratory data analysis. Since we wish to compare results from a number of methods, we only report on linear projections which have unique solutions. We incorporate a type of Gram Schmidt orthogonalisation [15] into the affinity maturation process to capture multiple components. We combine the new algorithm with reinforcement learning [17, 20] and with cross entropy maximization [13, 19]. Finally we combine several different non-standard adaptation methods using bagging and show that we get reliable convergence to accurate filters.

1 Introduction

In this paper, we develop immune-inspired algorithms for exploratory data analysis. Exploratory data analysis encompasses clustering, linear projections, manifold finding and any combination of these. We illustrate our methods on simple projection problems, principal component analysis (PCA), independent component analysis (ICA) and canonical correlation analysis (CCA). We use these linear methods since we wish to compare different non-standard adaptation methods but all the methods can be easily shown to e.g. model nonlinear manifolds [20].

Y. Wu · C. Fyfe (✉)
School of Computing, The University of the West of Scotland, Paisley, Scotland
e-mail: colin.fyfe@uws.ac.uk

Y. Wu
e-mail: ying.wu@uws.ac.uk

Given the data set represented by antigens, all the algorithms perform the same goal: the antibodies are generated in order to improve their affinity defined in different forms according to different projection problems. We begin by extending the well-known CLONALG algorithm [1, 4, 5] to projection problems, in which a little modification is made to the way that we perform the affinity proportional maturation in an artificial immune network. We then derive a modified CLONALG algorithm in which the cloning and maturation process is performed in a smoother way. In addition, it is often important to identify multiple components in projection methods. We continue to extend our modified CLONALG algorithm so that multiple components can be identified directly in the immune system itself.

We then continue to improve the performance of the non-standard adaptation methods by combining the algorithms we have previously derived or extended. We point out that the “better performance” is not limited to improving the accuracy of the final results, but also reducing the necessary size of data set and the number of iterations required to achieve the global optimum. We will demonstrate how well the combined algorithms work in comparison to the results of their parent algorithms. Two combined algorithms are developed. One is to incorporate cross entropy [13] into the immune system and the other is to integrate the immune-inspired algorithm with the Q-learning method [17].

Finally, we combine the three non-standard adaptation methods by bagging [9]. A new way is presented to determine the final solution based on the quality of the local solutions from the bags. The experimental results show this combined method converges to the optimal solution more reliably and stably and with higher accuracy than the individual methods.

2 Projection with immune-inspired algorithms

In this section, we extend artificial immune systems to solve projection problems. We demonstrate that we can apply the CLONALG algorithm to projection problems with a little modification. Keeping in mind that it is usually necessary to identify multiple components, we present a new way which incorporates deflationary orthogonalisation into the immune algorithm directly. We show that our method is quite general and can be easily applied to different projection methods.

2.1 The linear projections

We shall use our algorithms on some standard linear projection problems. The problems are

PCA [12] is the linear projection of a data set which retains most of the variance in a data set. Alternatively it can be defined as the linear projection which gives least mean squared error between the projections and the original data set. Thus if our data set is $\mathbf{x}_i$, $i = 1,...,N$, where each $\mathbf{x}_i$ is of dimension d, the first sample principal principal filter will be the vector $\mathbf{w}_1$ which is also of dimension d.

ICA [11] finds linear filters which provide projections which are as independent of each other as possible. Consider a set of independent signals, s_i, $i = 1,...,B$ and a set of observations which are linear mixtures of these signals (sometimes corrupted with noise) so that $\mathbf{x} = A\mathbf{s}$. Then ICA attempts to find filters, W, so that the outputs $\mathbf{y} = W\mathbf{x}$ are the original signals (up to a scaling factor and permutation).

CCA [15] finds the best linear filters of two data sets simultaneously so that the projections of the data sets have highest possible correlations.

The two most common orthogonalization algorithms are the Gram-Schmidt method [21] and symmetric orthogonalization [10]. Deflationary orthogonalization by the Gram-Schmidt method [21] is a simple and popular way to orthogonalize the weight vectors $\mathbf{w}_1,...,\mathbf{w}_N$. Provided that we have estimated $j-1$ weight vectors corresponding to the first $j-1$ components and we run the same algorithm for the next weight vector $\mathbf{w}_j$, we estimate $\mathbf{w}_j$ first and subtract the projection $(\mathbf{w}_j^T \mathbf{w}_k)\mathbf{w}_k$, $k = 1,...,j-1$ from $\mathbf{w}_j$, so we have a new weight vector $\mathbf{w}_j'$ and finally $\mathbf{w}_j$ is set to be $\mathbf{w}_j'$. More precisely, we follow the steps to estimate $\mathbf{w}_j$:

1. Randomly initialize the weight vector $\mathbf{w}_j$.
2. Perform the same algorithm on $\mathbf{w}_j$ as performed on the previous $j-1$ components.
3. Do the following orthogonalization:

$$\mathbf{w}_j \leftarrow \mathbf{w}_j - \sum_{k=1}^{j-1}(\mathbf{w}_j^T\mathbf{w}_k)\mathbf{w}_k \tag{1}$$

4. Normalize $\mathbf{w}_j$ by $\mathbf{w}_j \leftarrow \mathbf{w}_j/||\mathbf{w}_j||$.
5. Set $j = j + 1$ and go back to the first step.

2.2 The CLONALG algorithm

Inspired by the clonal selection principle and the affinity maturation process, de Castro and von Zuben [7] have developed a clonal selection algorithm, named CLONALG, to perform pattern recognition and optimization. The main immune aspects in the CLONALG algorithm [7] include:

– maintenance of a specific memory set;
– selection and cloning of the most stimulated antibodies;
– death of non-stimulated antibodies;
– affinity maturation and re-selection of the clones proportionally to their antigenic affinity;
– generation and maintenance of diversity.

For optimization problems, there is no explicit antigen population defined. Instead the antigen is represented by the objective function to be optimized. Also, an antibody affinity corresponds to the evaluation of the objective function for a given antibody. Some notations are shown below:

– **Ab**: antibody repertoire in a population, $\mathbf{Ab} \in \mathcal{S}^{N\times L}$, where N is the size of the population, L is the dimension of the antibody and $\mathcal{S}$ is the underlying space of the antibodies;
– **Ag**: population of antigens to be recognized. For optimization, **Ag** is defined to be the objective function to be optimized;
– $\mathbf{f}_i$: the affinity vector of the ith antibody in relation to antigen **Ag**;
– $\mathbf{Ab}_{\{n\}}$: n antibodies from the population of antibodies with highest affinities in relation to antigen $\mathbf{Ag}, \mathbf{Ab}_{\{n\}} \in \mathcal{S}^{n\times L}$;
– **C**: population of N_c clones generated from $\mathbf{Ab}_{\{n\}}, \mathbf{C} \in \mathcal{S}^{N_c\times L}$;
– N_c: the size of population of clones **C**;
– **C***: population from **C** after the affinity maturation process;
– $\mathbf{Ab}_{\{d\}}$: set of d new antibodies that will replace d low-affinity antibodies in **Ab**.

The CLONALG algorithm can then be described as follows [7]:

1. Randomly generate a population of antibodies **Ab**.
2. Determine the affinities of all the N antibodies to form the affinity vector $\mathbf{f}_i$, $i = 1,...,N$.

3. Select the n antibodies with highest affinity from **Ab** to compose a new set $\mathbf{Ab}_{\{n\}}$.
4. Clone the n selected antibodies independently and proportionally to their antigenic affinities to form a repertoire **C**. The higher the antigenic affinity, the higher the number of clones generated for each of the n selected antibodies. The size of population **C** is N_c.
5. The repertoire **C** is submitted to an affinity maturation process inversely proportional to the antigenic affinity, generating a population $\mathbf{C}^*$ of matured clones: the higher the affinity, the smaller the mutation rate.
6. Determine the affinity $\mathbf{f}^*$ of the matured clones $\mathbf{C}^*$.
7. From this population of mature clones $\mathbf{C}^*$, select n clones with highest affinities to be the antibodies in the new population **Ab** in the next iteration.
8. Finally, replace the d % lowest affinity antibodies from **Ab** by new individuals.

2.3 Linear projections with the modified CLONALG algorithm

We first consider the immune-inspired algorithm for the problem of ICA. In an artificial immune system, we define the mixed observations to be the antigens, **Ag**. Each antibody Ab_i in a population of antibodies **Ab** represents a possible solution of the kth independent component filter $\mathbf{w}_k$. One of the most common principles for ICA is to make all independent components as non-Gaussian as possible, which is often measured by the kurtosis of the distribution. Thus an antibody affinity corresponds to the evaluation of the absolute value of the kurtosis of the distribution of the recovered signals, $\mathbf{w}_k\mathbf{x}_i$, where $\mathbf{x}_i$ corresponds to an antigen Ag_i.

We consider a batch-learning CLONALG algorithm. Thus given the antigens, those antibodies with high affinities will be selected and cloned. Then these clones suffer an affinity maturation process, in which those antibodies with the highest affinity suffer the lowest mutation rates, whereas the lowest affinity antibodies have high mutation rates. After this mutation process, those matured clones with high affinities are kept in the new population of antibodies that will be used in the next iteration. Therefore, with the antibodies' affinity in the immune system improved, the demixing matrix will be optimized. We define the affinity as

$$\mathbf{f}_i = |\mathtt{kurt}(Ab_i^{\mathrm{T}}\mathbf{X})|. \tag{2}$$

where X is a matrix containing all the mixed samples $\mathbf{x}_j$ and the subscript i identifies the ith individual in the population. The modified CLONALG algorithm in detail is summarized as follows:

1. Initialize each antigen so that it corresponds to each observation in the data set. The size of the data set is N. Set $t = 0$.
2. Randomly generate a population of real-value antibodies $\mathbf{Ab}^{(t)}$ where the superscript is determined by the number of iterations of the algorithm performed. The size of the population is $N_{\mathbf{Ab}}$.
3. Determine the affinities to all the $N_{\mathbf{Ab}}$ antibodies to form the affinity vector **f**.
4. Select the n antibodies with highest affinity from $\mathbf{Ab}^{(t)}$ to compose a new set $\mathbf{Ab}_{\{n\}}$.
5. Clone the n selected antibodies independently to form a repertoire **C**. The number of clones for each antibody in $\mathbf{Ab}_{\{n\}}$ is round($\beta \cdot N$), thus the size of population of clones is $N_c = \sum_{i=1}^{n} \mathrm{round}(\beta \cdot N) = n \cdot \mathrm{round}(\beta \cdot N)$, where, in this paper, round($\cdot$) is the operator that rounds its argument towards the closest integer and β is a parameter that determines the number of clones.
6. The population of clones **C** is submitted to an affinity maturation process. Each clone is matured by $c'_{ij} = c_{ij} + \alpha \exp(-\mathbf{f}_i)\mu$, where μ is drawn from a Gaussian distribution with zero mean and unit variance, $\mu \sim \mathcal{N}(0, 1)$ and c_{ij} is defined as the jth clone of the ith parent antibody and α is a parameter that controls the decay of the inverse exponential function. We have a new population $\mathbf{C}^*$ of matured clones.
7. Determine the affinity $\mathbf{f}^*$ of the matured clones $\mathbf{C}^*$.
8. From this population of mature clones $\mathbf{C}^*$, select n clones with the highest affinities to be the antibodies in the new population $\mathbf{Ab}^{(t+1)}$ in the next iteration.
9. Finally, replace the $(1 - \frac{n}{N_{\mathbf{Ab}}}) \times 100\%$ lowest affinity antibodies in $\mathbf{Ab}^{(t)}$ by the new individuals.

There are a few modifications from the classic CLONALG algorithm. While the classic CLONALG algorithm encodes the individuals of the population using binary strings, the algorithm here is based on real-valued vectors. We also apply affinity proportional maturation. In our case, after the n clones with highest affinities are selected for the new population $\mathbf{Ab}^{(t+1)}$, we generate another ($N_{\mathbf{Ab}}-n$) new individuals. We therefore optimize the demixing matrix **W** by adjusting the position of the antibodies in the search space to reach the global optimum.

To illustrate the modified CLONALG algorithm for ICA, we use a 2-dimensional real data set, 'chirp' and 'gong', provided by Matlab. The number of original signals is 10,000 and we set $N_{\mathbf{Ab}} = 100$, $n = 10$, $\beta = 0.1$. So $N_c = 100$. These values were set after initial trial and experimentation suggested that they were appropriate values. The number of iterations for the estimation of each independent component is 10. The Gram-Schmidt method is performed to identify all the independent components. In

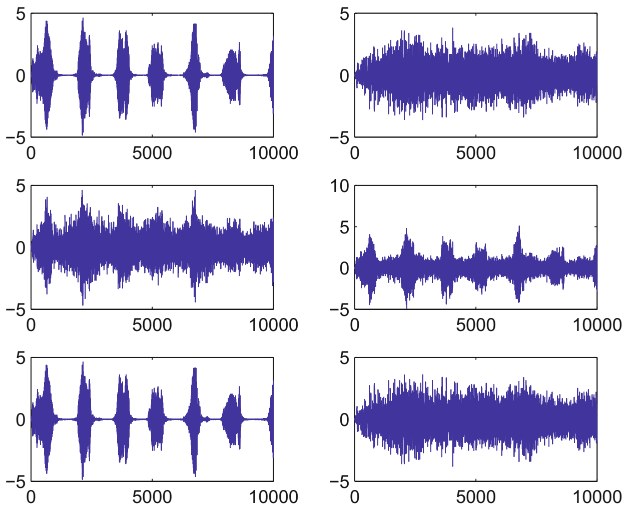

Fig. 1 ICA with modified CLONALG algorithm. *Top*: the original signals. *Middle*: mixed observations. *Bottom*: recovered ICs

Fig. 1, we see that all the independent components have been identified with extremely high accuracy as shown in Tables 1 and 2. We can also evaluate the performance of our ICA algorithm using the Amari error [6], where we compare the final **W** and **V**, the true demixing matrix, by

$$d(V,W)=\frac{1}{2M}\sum_{i=1}^{M}\left(\frac{\sum_{j=1}^{M}|a_{ij}|}{\max_j|a_{ij}|}-1\right)+\frac{1}{2M}\sum_{i=1}^{M}\left(\frac{\sum_{i=1}^{M}|a_{ij}|}{\max_i|a_{ij}|}-1\right) \quad (3)$$

where $a_{ij}=(VW^{-1})_{ij}$ and V is the true demixing matrix. For the above results, the Amari error is only 0.0028. Moreover, Fig. 2 shows that although the number of iterations is 10, the modified CLONALG algorithm can identify the independent component as early as the first two iterations, and so the speed of convergence is also extremely fast.

Therefore, in the modified CLONALG algorithm, the clones belonging to the same parent antibody Ab_i are samples from the Gaussian distribution with mean Ab_i and variance $\exp(-\mathbf{f}_i)$. We also find from the experiments that a smaller variance when cloning and maturing the parent antibody leads to higher accuracy and smoother convergence. Thus, in practice, the parent antibodies are cloned and matured by

$$c'_{ij}\sim\mathcal{N}(Ab_i,\alpha\exp(-\mathbf{f}_i)),\alpha\in(0,1]. \quad (4)$$

We apply the modified CLONALG algorithm to solve PCA. We use a 5-dimensional data set in which each element, x_i is drawn from N(0,i^2), the zero mean Gaussian distribution with variance i^2. The first principal component is readily identified as the fifth input dimension. The goal

Table 1 The kurtosis of the original signals, mixed observations and recovered independent components (ICs)

	Kurtosis 1	Kurtosis 2
Original signals	7.0592	3.1467
Mixed observations	3.7093	5.8907
Recovered ICs	7.0590	3.1467

Table 2 Correlation between the original sources and recovered signals

	Signal 1	Signal 2
Recovered signal 1	**1.0000**	0.0014
Recovered signal 2	0.0003	**1.0000**

of PCA is to find the linear projection of a data set which contains maximal variance, thus the affinity between the antigens and the antibody is defined as

$$\mathbf{f}_i=\frac{1}{1+\exp(-\gamma(\mathbf{Ab}_i^{\mathrm{T}}\mathbf{X})^2)}. \quad (5)$$

The size of the data set is 10,000 and we set $N_{\mathbf{Ab}}=100$, $n=10$, $\beta=0.1$, $\alpha=0.5$, $\gamma=0.00001$. The number of iterations is 100. Table 3 shows that the modified CLONALG algorithm has identified the first principal component with high accuracy and we see in Fig. 3 that the algorithm can identify the first principal component within five iterations and then climbs towards the global optimum. In addition, we can see in Tables 3 and 4 that the accuracy is improved by setting a smaller variance in (4).

2.4 Multiple components

It is frequently necessary in projection methods to identify multiple components. It is straightforward to apply the Gram-Schmidt method [21] to identify multiple components. Thus assuming the first $j-1$ weight vectors have been estimated, we change step 2 in the modified CLONALG algorithm, where for each antibody Ab_i^j, we perform $Ab_i^j\leftarrow Ab_i^j-\sum_{k=1}^{j-1}(Ab_i^j\mathbf{w}_k)\mathbf{w}_k$, where $\mathbf{w}_k$ corresponds to the kth weight vector previously estimated and Ab_i^j corresponds to the ith antibody for the current weight vector. Then by using the same data set as in the previous subsection, Table 5 shows that all the principal components have been identified.

However, we are more interested in identifying multiple components within the artificial immune system itself. We consider each weight vector $\mathbf{w}_i$ previously estimated as a memory cell in a population $\mathbf{Ab}_{\{m\}}$ in which $Ab_{\{m\}}^k$ corresponds to the kth memory cell in the population. The distance between one antibody Ab_i and these memory cells

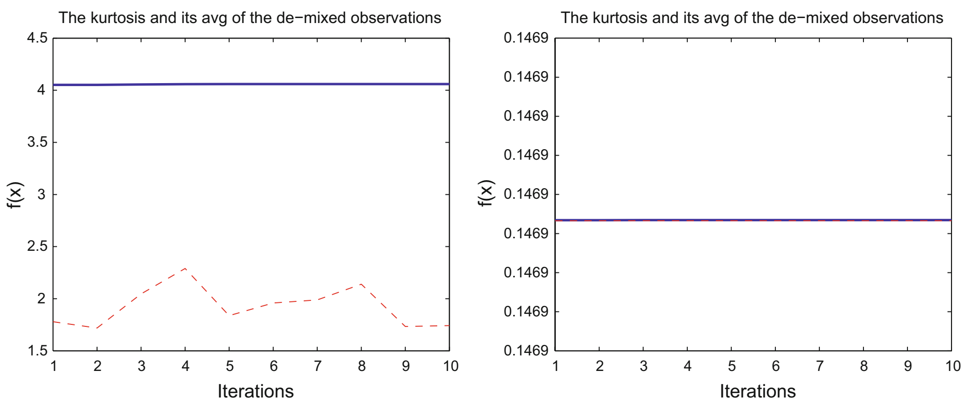

Fig. 2 Affinity of the population. Highest (*solid line*) and average (*dashed line*). *Left*: affinity for the first IC. *Right*: affinity for the second IC

Table 3 The first principal component with 5-dimensional artificial data by CLONALG algorithm with $\alpha = 0.7$

PC1	−0.0168	0.0153	0.0122	0.0369	**0.9992**

Table 4 The first principal component with 5-dimensional artificial data by the modified CLONALG algorithm with $\alpha = 1$

PC1	−0.1030	0.1022	−0.1328	−0.0213	**0.9802**

Table 5 The weights from the artificial data experiment for five principal components with the Gram-Schmidt method

PC1	−0.0168	0.0153	0.0122	0.0369	**0.9992**
PC2	−0.0211	−0.0276	0.1082	**0.9927**	−0.0379
PC3	−0.0225	0.0225	**0.9929**	−0.1084	−0.0088
PC4	−0.0187	**0.9986**	−0.0201	0.0289	−0.0164
PC5	**0.9968**	0.0188	0.0245	0.0197	0.0154

in population $\mathbf{Ab}_{\{m\}}$ can be measured, so that a low distance implies that the antibody Ab_i is similar to the weight vectors previously estimated. Thus the basic idea is that the antibodies selected to be cloned and matured should have high distances from the memory cells.

We re-structure the modified CLONALG algorithm for multiple components: at iteration t, we first evaluate the distance between each antibody in $\mathbf{Ab}$ generated in step 2 and the memory cells in $\mathbf{Ab}_{\{m\}}$. The $N'_{\mathbf{Ab}}$ antibodies with highest distance are selected and others are eliminated from the population. The $N'_{\mathbf{Ab}}$ antibodies are then cloned and matured. The modified CLONALG algorithm for multiple components is summarized as follows:

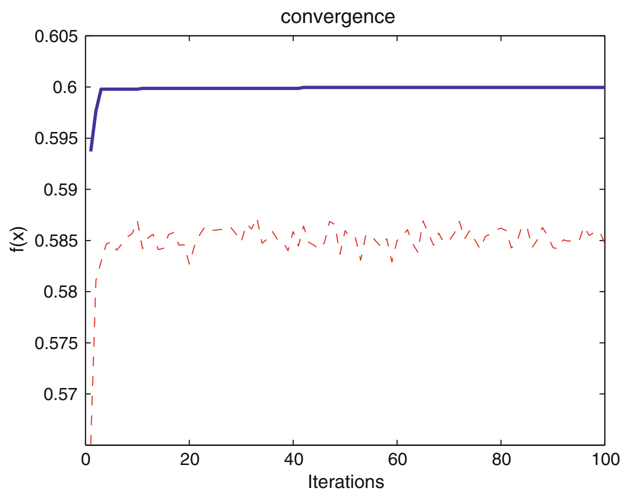

Fig. 3 Affinity of the population by the modified CLONALG algorithm for PCA. The *vertical axis* shows the affinity in each iteration. The *horizontal axis* shows the number of iterations. *Solid line*: the highest affinity. *Dashed line*: the averaged affinity

1. Initialize each antigen corresponding to each data point in the data set. The size of the data set is N.
2. Randomly generate a population of real-valued antibodies $\mathbf{Ab}^{(t)}$. The size of the population is $N_{\mathbf{Ab}}$.
3. Measure the distances between each antibody Ab_i and the memory cells $\mathbf{Ab}_{\{m\}}$ according to (6) and select the $N'_{\mathbf{Ab}}$ antibodies with highest distance. For the first component, this step is omitted and $N'_{\mathbf{Ab}} = N_{\mathbf{Ab}}$.
4. Determine the affinities to all the $N'_{\mathbf{Ab}}$ antibodies to form the affinity vector $\mathbf{f}_i$.
5. Select the n antibodies with highest affinity to compose a new set $\mathbf{Ab}_{\{n\}}$.
6. Mature and clone the n selected antibodies independently according to (4) to form a population $\mathbf{C}^*$. The size of the population of clones is $N_c = \sum_{i=1}^{n} \mathrm{round}(\beta \cdot N'_{\mathbf{Ab}})$.
7. Determine the affinity $\mathbf{f}^*$ of the matured clones $\mathbf{C}^*$.
8. From this population of mature clones $\mathbf{C}^*$, select n clones with highest affinities to be the antibodies in the new population $\mathbf{Ab}^{(t+1)}$ in the next iteration.
9. Finally, replace the $(1 - \frac{n}{N_{\mathbf{Ab}}}) \times 100\%$ lowest affinity antibodies in $\mathbf{Ab}^{(t)}$ by new randomly generated individuals.
10. Add the cell with highest affinity to the memory cells.

11. If not at maximum iterations, go back to step 3.

In step 3, the distance between each antibody Ab_i in $\mathbf{Ab}$ and the memory cells in $\mathbf{Ab}_{\{m\}}$ can be measured by angle between two vectors,

$$D_{Ab_i} = \sum_{k}^{j-1} \left| \frac{Ab_i^{j\mathrm{T}} \cdot Ab_{\{m\}}^k}{|Ab_i^j| \cdot |Ab_{\{m\}}^k|} \right|. \tag{6}$$

To illustrate our algorithm, we consider the problem of CCA. We consider that there are two sets of antigens, $\mathbf{Ag}_1$ and $\mathbf{Ag}_2$ to represent the two sets of data and accordingly two populations of antibodies, $\mathbf{Ab}_1$ and $\mathbf{Ab}_2$, are generated to represent the two weight vectors. We define the affinity between the antigens and the antibody using

$$\mathbf{f}_i = \frac{1}{1 + \exp(\gamma||\mathbf{Ab}_{1i}^{\mathrm{T}}\mathbf{X}_1 - \mathbf{Ab}_{2i}^{\mathrm{T}}\mathbf{X}_2|| + |(\mathbf{Ab}_{1i}^{\mathrm{T}}\mathbf{X}_1)^2 - 1| + |(\mathbf{Ab}_{2i}^{\mathrm{T}}\mathbf{X}_2)^2 - 1|)}. \tag{7}$$

where $\mathbf{X}_1 = \mathbf{Ag}_1$ and $\mathbf{X}_2 = \mathbf{Ag}_2$. We use an artificial data set similar to what has been used in [14], in which there are two sets of artificial data, one of which is 4-dimensional and the other is 3-dimensional: each of the elements is drawn from the zero-mean Gaussian distribution, $\mathcal{N}(0,1)$ and we add an additional sample from $\mathcal{N}(0,1)$ to the first elements of each vector and then divide by 2 to ensure that there is no more variance in the first elements than in the others. To generate the second correlation, we add an additional sample from $\mathcal{N}(0,0.5)$ to the second elements of each vector, so that the second correlation is smaller than the first one. We set $N'_{\mathbf{Ab}} = 100$, $n = 10$, $\beta = 0.1$, $\alpha = 0.5$, $\gamma = 0.00001$ and for multiple components, $N_{\mathbf{Ab}} = 10000$. The number of iterations is 500.

In Table 6, we see that the correlation components (CCs) have been identified. Figures 4 and 5 show that the speed of convergence is fast. We also find that the convergence of immune-inspired algorithms is not as smooth as that of our previous non-standard adaptation methods, reinforcement learning [20] and cross entropy [19]; instead the algorithms can identify a local optimum quickly and finally converge to the global optimum.

Table 6 The first two canonical correlation (CC) weight vectors found with the artificial data

	m1	m2
1st CC	**0.9967**	**0.9999**
	−0.0510	0.0094
	0.0613	−0.0131
	0.0137	
2nd CC	−0.0692	0.0028
	−0.9942	**−0.9986**
	0.0146	0.0529
	0.0806	

3 Combining non-standard adaptation methods

We have previously extended reinforcement learning [20] and cross entropy methods [19] to solve projection problems. In the previous section, we demonstrate that immune-inspired algorithms can be applied to solve the same problems. In this section, we develop algorithms that can achieve better performance in solving projection problems by combining these non-standard adaptation methods. The "better performance" is not only to increase the accuracy of the final optimized solution, but also decrease the size of data set needed for consistent performance and the number of iterations required. We first incorporate the cross entropy method into an artificial immune system, in which a new clone and maturation process is present. Then we show that the immune-inspired algorithm can be integrated with temporal difference learning. The algorithms in this section are applied to identify the first principal component of PCA so that the results can be easily compared with those achieved previously.

3.1 Artificial immune system with cross entropy

In this subsection, we incorporate the cross entropy method [13] into an artificial immune system, and evaluate the

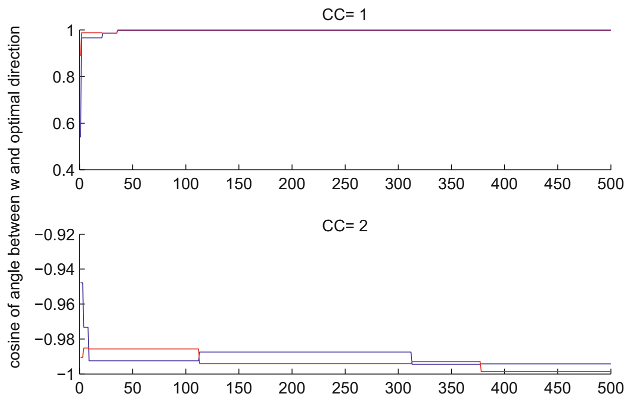

Fig. 4 Convergence of the CCA weight vectors to the optimal directions; the *horizontal axis* shows the number of iterations of the algorithm. *Top*: the first canonical correlation. *Bottom*: the second canonical correlation

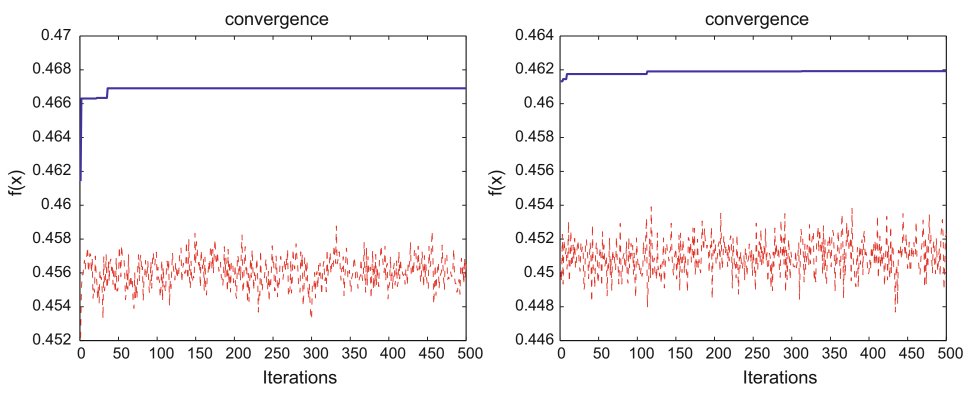

Fig. 5 Affinity of the population. Highest (*solid line*) and average (*dashed line*). *Left*: affinity for the first CC. *Right*: affinity for the second CC

optimal solution as that with higher accuracy. The cross entropy method can achieve the global optimum by defining a family of probability density functions $\{f(.;\mathbf{v}), \mathbf{v} \in \mathcal{V}\}$ on the data set $\aleph$ and we make adaptive changes to the probability density function according to the Kullback-Leibler cross-entropy. In brief, we have an iterative algorithm working on a probability density function with a specific set of parameters. We update the parameters at each iteration in order to make the event in which we are interested more likely at the next iteration.

Considering the modified CLONALG algorithm in Sect. 2.3, for a selected antibody $Ab_{\{n\}i}$ in population $\mathbf{Ab}_{\{n\}}$, we have a series of matured clones drawn from a Gaussian distribution with the mean $Ab_{\{n\}i}$ and the variance $\exp(-\mathbf{f}_i)$. We denote the clones belonging to its parent antibody $Ab_{\{n\}i}$ as $\mathbf{C}_i^*$ with $\mathbf{C}_{ij}^*$ corresponding to the *j*th clone of the *i*th parent antibody. Thus we wish to maximize the affinity function $\mathbf{f}(\mathbf{C}_{ij}^*)$ over all clones in $\mathbf{C}_i^*$. Denoting the maximum by γ^*, we have

$$\gamma^* = \max_{\mathbf{c}_{ij}^* \in \mathbf{C}_i^*} \mathbf{f}(\mathbf{c}_{ij}^*) \quad (8)$$

Then we can use (8) as our cross entropy criterion. The algorithm in detail is as follows:

1. Initialize each antigen corresponding to each data point in the data set. The size of the data set is *N*.
2. Randomly generate a population of real-value antibodies $\mathbf{Ab}^{(t)}$. The size of the population is $N_{\mathbf{Ab}}$.
3. Determine the affinities to all the $N_{\mathbf{Ab}}$ antibodies to form the affinity vector $\mathbf{f}_i$.
4. Select the *n* antibodies with highest affinity from $\mathbf{Ab}^{(t)}$ to compose a new set $\mathbf{Ab}_{\{n\}}$.
5. Each selected antibody $Ab_{\{n\}i}$ is matured and cloned according to (4) to form a population $\mathbf{C}_i^*$. The number of clones in $\mathbf{C}_i^*$ is round$(\beta \cdot N_{\mathbf{Ab}})$.
6. Determine the affinities $\mathbf{f}^*$ of the matured clones $\mathbf{C}_i^*$. Let $\hat{\gamma}_t$ be the $1-\varrho$ clone quantile, above which we identify the "elite" clones. Use the "elite" clones to generate a new antibody $Ab_i^{(t+1)}$.
7. Put all these new antibodies $Ab_i^{(t+1)}$, $i = 1,...,n$ to form part of the new population $\mathbf{Ab}^{(t+1)}$.
8. Replace the $(1-\frac{n}{N_{\mathbf{Ab}}}) \times 100\%$ lowest affinity antibodies in $\mathbf{Ab}^{(t)}$ by new individuals to form the other part of $\mathbf{Ab}^{(t+1)}$.

To illustrate the modified CLONALG algorithm with cross entropy, we use the same 5-dimensional data as previously in which the first principal component is readily identified as the fifth input dimension. The size of the data set is 10,000 and we set $N_{\mathbf{Ab}} = 100, n = 10, \beta = 0.1, \alpha = 0.5, \varrho = 0.5$. The number of iterations is 100. We can see that our algorithm has identified the first principal component with much higher accuracy in Table 7. Figures 6 and 7 show that our algorithm has converged extremely fast. Furthermore, we have found that the cosine of the angle between the estimated weight vector and the optimal vector is higher than 0.99, even when the number of iterations is reduced to 10.

We compare the results by the classic CLONALG algorithm, the modified CLONALG algorithm and the CLONALG algorithm with cross entropy. The experiments are performed with the same size of data set, 10,000, and the same number of iterations, 100. Figure 8 shows that although all of the algorithms can identify the first principal component quickly, the two latter algorithms are more efficient than the classic CLONALG algorithm.

3.2 Temporal difference (TD) learning with artificial immune system

We have demonstrated that reinforcement learning with Q-learning [17] can be applied to solve projection problems with high accuracy. However, this algorithm suffers from the drawback that the speed of convergence is slow,

Table 7 The first principal component with 5-dimensional artificial data by our modified CLONALG algorithm with cross entropy

PC1	0.0036	0.0007	0.0077	−0.0053	**0.9999**

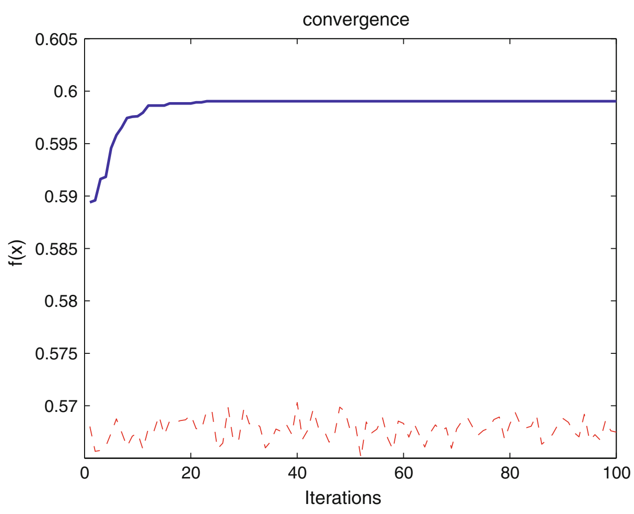

Fig. 6 Affinity of the population by our modified CLONALG algorithm. Highest (*solid line*) and average (*dashed line*)

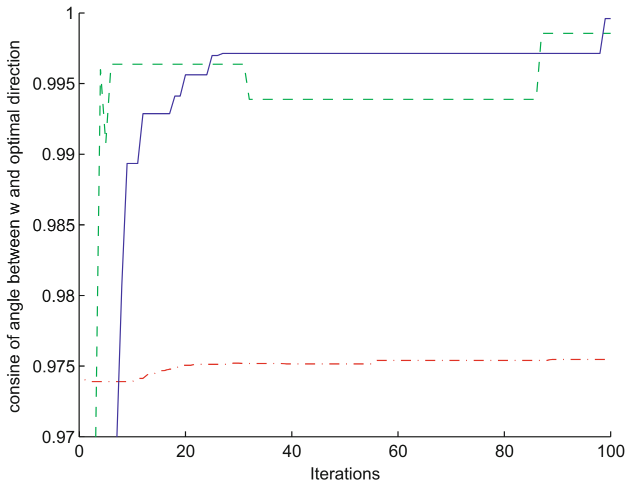

Fig. 8 Convergence of the PCA weight vector to the optimal directions. *Dash-dot line*: by the classic CLONALG algorithm. *Dashed line*: by the modified CLONALG algorithm. *Solid line*: by the CLONALG algorithm with cross entropy

which leads to a large number of iterations. In this subsection, we improve the algorithm by integrating immune-inspired algorithms with Q-learning.

The Q-learning method has been introduced in [17]. This method directly approximates the optimal action-value function, Q^*, by the learned action-value function, Q, and the best possible action selected in the subsequent state:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)].$$

With reinforcement learning, the state of the system at any time is the data sample presented to the system at that time, i.e. $s_t = \mathbf{x}_t$. However, instead of sampling the weight vector $\mathbf{w}$ from the distribution $\mathcal{N}(\mathbf{m}, \beta^2 I)$ with the current estimate of the parameters, $\mathbf{m}$ and β^2, we define the action to be to generate a population of antibodies Ab_i, where the modified CLONALG algorithm is performed to maximize the Q-value of the current data point $\mathbf{x}_t$. Thus the affinity function is defined to calculate the Q-value of the antibody. At each iteration, we keep a note of the greatest Q-value and only update the Q-value of the data point by the antibody Ab_{i^*} with the highest Q-value. We therefore restructure the learning algorithm as follows:

1. Randomly generate a population of real-value antibodies $\mathbf{Ab}^{(t)}$. The size of the population is $N_{\mathbf{Ab}}$.
2. Randomly select a data point $\mathbf{x}_t$ to be the current state s_t. Denote the current Q-value of the state s_t as Q_t
3. Determine the reward r_i for each antibody Ab_i in population $\mathbf{Ab}$.
4. Determine the Q-value of each antibody Ab_i in population $\mathbf{Ab}$ to form the affinity vector $\mathbf{f}_i$ by
$$\Delta Q_{Ab_i} \leftarrow \alpha(r_i + \gamma Q_{Ab_{i^*}} - Q_t) \quad (9)$$
$$Q_{Ab_i} \leftarrow Q_{Ab_i} + \Delta Q_{Ab_i} \quad (10)$$
5. Select the n antibodies with highest affinity from $\mathbf{Ab}$ to compose a new set $\mathbf{Ab}_{\{n\}}$. Denote the antibody with the highest affinity as Ab_{i^*}.
6. Update the Q-value of data point $\mathbf{x}_t$ with the affinity of Ab_{i^*}.
7. Mature and clone the n selected antibodies independently according to (4) to form a population $\mathbf{C}^*$. The size of population of clones is $N_c = \sum_{i=1}^{n}$ round $(\beta \cdot N_{\mathbf{Ab}})$.
8. Determine the affinity $\mathbf{f}^*$ of the matured clones $\mathbf{C}^*$.
9. From this population of mature clones $\mathbf{C}^*$, select n clones with highest affinities to be the antibodies in the new population $\mathbf{Ab}^{(t+1)}$ in the next iteration.

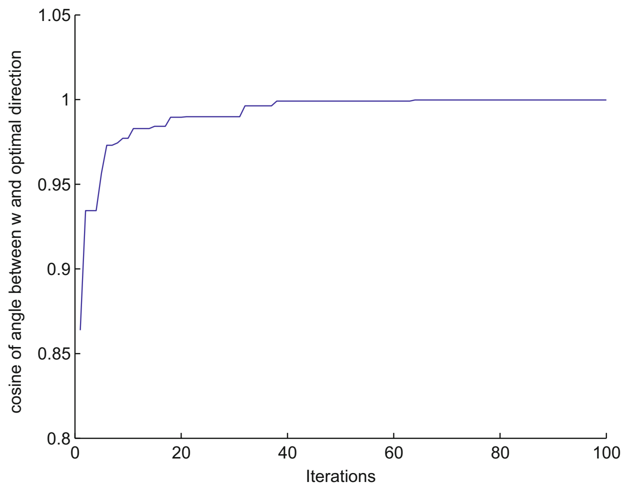

Fig. 7 Convergence of the PCA weight vector to the optimal directions. The *vertical axis* shows the cosine of the angle between the current filter and the optimal filter. The *horizontal axis* shows the number of iterations

10. Finally, replace the $(1-\frac{n}{N_{\mathbf{Ab}}})\times 100\%$ lowest affinity antibodies in $\mathbf{Ab}^{(t)}$ with new individuals.

To illustrate the Q-learning with immune-inspired algorithm, we use the same 5-dimensional data in which the first principal component is readily identified as the fifth input dimension. The size of the data set is 100 and we set $N_{\mathbf{Ab}} = 100$, $n = 10$, $\beta = 0.1, \alpha = 0.5$. The number of iterations is 2,000. We can see that our algorithm has identified the first principal component with high accuracy in Table 8. Figure 10 shows that our algorithm converges quickly and the greatest Q-value is maximized step by step as shown in Fig. 9.

We can see that our Q-learning with immune-inspired algorithm has improved the performance of the Q-learning method in projection problems in terms of the number of iterations and with a much reduced size of data set. Figure 10 also shows that when the number of iterations is 2,000 and the size of data set is 100, the Q-learning method alone can not identify the first principal component: we have to increase the size of the data set and the number of iterations so that this Q-learning method can converge to the optimal solution.

4 Ensembles of the non-standard adaptation methods

Recently, algorithms based on non-standard adaptation methods have been developed to solve projection problems and the results have shown that all the algorithms can converge to the optimum stably and with high accuracy. However, to achieve highly accurate results, it is always necessary to set the parameters of an algorithm carefully and usually a high volume of data is required, otherwise, the algorithm may fail to find the optimal solution, or become unstable. In this section, we investigate ensemble methods in the context of using non-standard adaptation methods to solve projection problems. Specifically, we investigate bagging to perform reinforcement learning, cross entropy methods and immune-based algorithms in parallel, through which the converged optimum is more stable and reliable and with higher accuracy.

4.1 Bootstrapping and bagging

Bagging was proposed by Breiman [2], which is based on bootstrapping [9] and aggregating concepts. Bootstrapping [9] is a simple and effective way of estimating a statistic of a data set. Suppose we have a data set, $D = \mathbf{x}_i$, $i = 1,\ldots,N$: bootstrapping creates a number of pseudo data sets, D_i, by sampling from D with uniform probability with replacement of each sample. Thus each data point has a probability of $\left(\frac{N-1}{N}\right)^N \approx 0.368$ of not appearing in each bootstrap sample, D_i. Aggregating means we can perform the same or even different algorithms in parallel. Typical applications with bagging are prediction [3, 8], classification [16] and data pre-processing [18].

In detail, bagging consists of selecting B data sets, $D_1,\ldots,D_B$ from D by randomly selecting a member of D with replacement and each data set D_b will almost certainly contain only some members of the original data set D. From a classification perspective, each classifier

Table 8 The first principal component with 5-dimensional artificial data by Q-learning with immune-inspired algorithm

PC1	−0.0194	0.0125	0.0647	0.1152	**−0.9910**

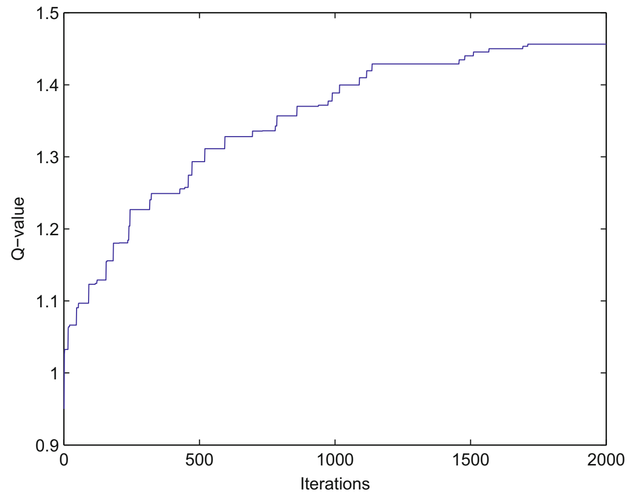

Fig. 9 Q-value maximized by our Q-learning with immune-inspired algorithm. The *vertical axis* shows the maximum Q-value. The *horizontal axis* shows the number of iterations

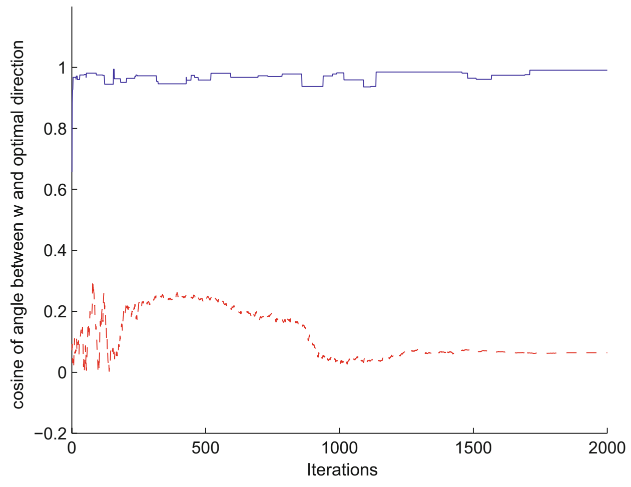

Fig. 10 Convergence of the PCA weight vector to the optimal directions. *Dashed line*: by Q-learning method alone. *Solid line*: by Q-learning with immune-inspired algorithm in this subsection

$C^b(\mathbf{x})$ is then constructed based on the data set D_b. All the classifiers are combined by simple majority voting or some average combination rule. Furthermore, the elements of $T_b = D - D_b$ can be used to access how accurate the training of the individual classifier is liable to be.

4.2 Non-standard adaptation methods with bagging

We consider the problem of PCA and use the same 5-dimensional data in which the first principal component is readily identified as the fifth input dimension. The size of data set is 1,000. We create the first 20 bags by selecting from the data set and we perform the reinforcement learning algorithm on each bag. The number of iterations is 200,000. The learning rate is initialized to 0.1, which is reduced step by step. We create the second 20 bags using the cross entropy method with batch-learning. The number of iterations is 2,000. We set the number of random samples as each iteration as 50 and $\varrho = 0.1$. We create the last 20 bags with the modified CLONALG algorithm. The number of iterations is 500 and we set $N_{\mathbf{Ab}} = 100$, $n = 10$, $\beta = 0.1$, $\alpha = 0.5$. For all of the 60 bags, the number of data points in each bag, N_{D_b} is 500.

Therefore, each bag has one *local solution*, $\mathbf{w}_b$. We present a new way, *self-organized majority voting*, to decide the final solution according to these local solutions. We first evaluate how well each local solution has been optimized by

$$J_{\mathbf{w}_b} = \sum_{i=1}^{N} |\mathbf{w}_b \mathbf{x}_i|. \tag{11}$$

We denote that local solution that has been optimized best as the winning local solution, $\mathbf{w}_{b^*}$. Then the weight of one local solution in the final solution is defined proportional to the distance between each local solution, $\mathbf{w}_b$, and the winner, $\mathbf{w}_{b^*}$,

$$h_{b^*}(b) = \exp(-\gamma||\mathbf{w}_b - \mathbf{w}_{b^*}||^2) \tag{12}$$

Then the final solution is calculated by

$$\mathbf{w} = \frac{\sum_{b=1}^{B} h_{b^*}(b)\mathbf{w}_b}{\sum_{b=1}^{B} h_{b^*}(b)}. \tag{13}$$

We compare the final solution by our method with the average first principal component filters by reinforcement learning, cross entropy method and modified CLONALG algorithm respectively. Table 9 shows that the final solution by this method has higher accuracy than the individual methods. We can also see that bagging with self-organized majority voting is more efficient than bagging with simple majority voting.

We decrease the size of data set to 500 with 100 data points in each bag. In such a situation, all the non-standard adaptation algorithms used in this subsection become much more unstable and unreliable. However, we see that the final solution by bagging with self-organized majority voting is much more stable and reliable than those by other methods as shown in Table 10.

Table 9 The first principal component filters identified by bagging with self-organized majority voting and simple majority voting and the average first principal component filters by reinforcement learning, cross entropy method and the modified CLONALG algorithm

Self-organized voting	0.0143	0.0197	0.0501	0.0915	**0.9943**
Simple voting	0.0142	0.0197	0.0505	0.0957	**0.9909**
Reinforcement learning	0.0136	0.0149	0.0400	0.0887	**0.9929**
Cross entropy method	0.0103	0.0192	0.0519	0.1176	**0.9877**
CLONALG algorithm	0.0188	0.0249	0.0599	0.0814	**0.9920**

Table 10 The first principal component filters identified by bagging with self-organized majority voting and simple majority voting and the average first principal component filters by reinforcement learning, cross entropy method and the modified CLONALG algorithm

Self-organized voting	0.0198	0.0369	0.0707	0.1086	**0.9907**
Simple voting	0.0224	0.0461	0.0978	0.3109	**0.8974**
Reinforcement learning	0.0217	0.0380	0.0991	0.2126	**0.9531**
Cross entropy method	0.0247	0.0558	0.1047	0.4664	**0.8018**
CLONALG algorithm	0.0208	0.0456	0.0910	0.2770	**0.9218**

The size of data set is decreased

5 Conclusion

We have, in this paper, used an artificial immune system to solve projection problems by developing immune-inspired algorithms. We have first demonstrated that the CLONALG algorithm can be applied to independent component analysis. The data set is represented by antigens and the artificial immune system improves the affinity by generating a series of populations of antibodies. The affinity is defined in different ways in order to solve different projection problem. We have performed the affinity maturation process used by artificial immune networks in the CLONALG algorithm, which we consider to be more suitable for projection problems. We have developed a smoother way to perform the cloning and maturation process and the results have shown that the performance is much improved.

We have also presented a new way to perform deflationary orthogonolization, in which multiple components are identified by the immune system directly. We regard the weight vectors previously estimated as memory cells and the basic idea is that the distance between antibodies selected to be cloned and matured in a new population and those memory cells, is as large as possible. Thus for a population of antibodies, we select the antibodies with largest distance to the memory cells, based on which we re-select the antibodies with highest affinity. We consider this way to be particularly suitable for an artificial immune system in that all components can be identified with the same affinity function.

We have investigated the combination of different non-standard adaptation methods. We first incorporated the cross entropy method into the modified CLONALG algorithm. Instead of selecting the clones with highest affinity, the antibodies in the new population are directly generated according to the clones, where the cross entropy method is performed. The results show that this combined algorithm has higher accuracy with faster convergence. We have compared it with the modified CLONALG algorithm. Then we integrated the modified CLONALG algorithm with the Q-learning method. All the antibodies aim to maximize the Q-value of the current state. We demonstrated that such a combined algorithm can not only identify the principal component with high accuracy but also requires a smaller size of data set. The number of iterations has also been extremely reduced.

We have applied our algorithms to ICA, PCA and CCA as examples. We have only demonstrated the combined algorithms in identifying the first principal component, which is more convenient for comparing the performance of different algorithms, however it is worth noting that all the immune-inspired algorithms are general methods that can be applied to the other projection problems.

Finally, we have combined the three non-standard adaptation methods with bagging. We have also presented a self-organized majority voting method to decide the final solution based on the local solutions, where the proportion of a local solution in the final solution is decided by the distance between that local solution and the winner solution. The experiments have shown our method can make the algorithms more stable and reliable and with higher accuracy.

References

1. Ada GL, Nossal GJV (1987) The clonal selection theory. Sci Am 257(2):50–57
2. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
3. Breiman L (1999) Using updaptive bagging to debias regression. Technical Report 547, Statistics Department, University of California
4. Burnet FM (1959) The clonal selection theory of acquired immunity. Cambridge University Press, Cambridge
5. Burnet FM (1978) Clonal selection and after. Theor Immunol 300(19):1105–1107
6. Cichocki AA, Yang HH (1996) A new learning algorithm for blind signal separation. Adv Neural Inf Process Syst 8:757–763
7. de Castro LN, Von Zuben FJ (2002) Learning and optimization using the clonal selection principle. IEEE Transaction on evolutionary computation, special issue on artificial immune systems, vol 6, pp 306–313
8. Dybowski R, Roberts S (2001) Confidence intervals and prediction intervals for feed-forward neural networks. Cambridge University Press, Cambridge
9. Efron B, Tibshirani R (1993) An introduction to the bootstrap. Chapman and Hall
10. Fan CY, Wang BQ, Ju H (2006) A new fastica algorithm with symmetric orthogonalization. In: Communications, circuits and systems proceedings, 2006 international conference, June, vol 3, pp 2058–2061
11. Hyvarinen A, Karhunen J, Oja E (2001) Independent component analysis. Wiley
12. Jolliffe IT (1986) Principal component analysis. Springer
13. Kroese DP, Rubinstein RY (2004) The cross entropy: a unified approach to combination optimization, Monte-Calo simulation and mechine learning. Spinger
14. Lai PL (2002) Neural implementations of canonical correlation analysis. PhD thesis, University of Paisley
15. Mardia KV, Kent JT, Bibby JM (1979) Multivariate analysis. Academic Press
16. Skurichina M, Robert Duin PW (2002) Bagging, boosting and the random subspace method for linear classifiers. Pattern Anal Appl 5(2):121–135
17. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. The MIT Press
18. Wu Y, Fyfe C (2005) Pre-processing using topographic mappings. In: ICNNB05, vol 3, pp 1881–1884
19. Wu Y, Fyfe C (2008) Topology preserving mappings using cross entropy adaptation. In: The 7th WSEAS international conference on artificial intelligence, knowledge engineering and data bases, AIKED'08
20. Wu Y, Fyfe C, Lai PL (2007) Stochastic weights reinforcement learning for exploratory data analysis. In: 17th international conference on artificial neural networks, ICANN2007, pp 667–676
21. Zhang K, Chan LW (1997) Dimension reduction as a deflation method in ica. SP Lett 13(1):45–48