Experimental Results On Compression Quality of aiNet

Thomas Stibor
Department of Applied Science
University of California, Davis
California 95616, United States
Email: tstibor@ucdavis.edu

Rao Vemuri
Department of Applied Science
University of California, Davis
California 95616, United States
Email: rvemuri@ucdavis.edu

***Abstract*— AiNet is an immune-inspired algorithm for data compression, i.e. the reduction of redundancy in data sets. In this paper we experimentally investigate the compression quality of aiNet. Therefore, a similarity measure between input set and reduced output set is presented, which is based on Parzen window estimation and the Kullback-Leibler divergence. Four different artificially generated data sets are created and the compression quality is investigated. Experiments reveal that aiNet produced reasonable results on uniformly distributed data sets, but poor results on non-uniformly distributed data sets, i.e. data sets which contain dense point regions.**

I. Introduction

The immune system provides from a technical point of view many problem oriented approaches for solving pattern recognition and clustering problems [1], [2]. A frequently applied immune-inspired algorithm for data reduction is aiNet (artificial immune network) [3]. AiNet is inspired by Jerne's postulated immune network theory and can be considered as a micro-evolutionary search algorithm. Roughly speaking, the aim of aiNet algorithm is to find a reduced set of points that closely represents the set of input points, or in other words, find a compressed input set representation with less redundancy. Once the reduced set is found, established clustering techniques like minimum spanning tree or k-means can be applied to determine a clustering result[1] [3], [5]. The aiNet algorithm can therefore be considered as an immune-inspired search technique in the problem domain: given an input set of points, output a reduced set of points that closely represents the input set.

In this context two fundamental questions arise: *What does "closely represent" mean? How to measure the quality of the reduced output set?* We address these questions and organize as follows: In section II, Jerne's immune network theory is briefly summarized. In section III, the aiNet algorithm is outlined. In section IV, we briefly explain the nonparametric density estimation method and show a (known) probability density approach to measure the quality between input set and reduced output set by means of Parzen window estimators. In section V, the quality measure is experimentally investigated on different artificially generated data sets. In section VI, the obtained results are analyzed and discussed.

[1]In [4] authors note an important fact on redundancy and knowledge: *"Unsupervised learning can only do anything useful when there is redundancy in the input data. Without redundancy it would be impossible to find any patterns or features in the data, which would necessarily seem like random noise. In this sense redundancy provides knowledge"*. From this fact follows that reducing redundancy and subsequently to cluster seems not to be useful.

II. Immune Network Theory

The immune network theory proposed by Jerne[2] [7] postulates that the immune system consists of antibodies and antigens which form a regulated network. In Jerne's theory antibodies recognize not only antigens[3], but also other antibodies. As a result, the network is also regulated without antigenic influence. In addition to this new perspective, the network structure — i.e. the size and connectivity — is biased by *network stimulation* and *network suppression*. A network stimulation results in proliferation and memorization of antibodies, while a network suppression results in a reduction of maintained antibodies. According to Varela and Coutinho [8] three major immune network characteristics can be emphasized:

- *structure*: types of interactions among the antibodies and antigens, represented by matrices of connectivity
- *dynamics*: variation with time of the concentration and affinities of cells and molecules comprising the network
- *metadynamics*: continuous production of novel cells and death of non-stimulated and self-reactive cells.

The network structure is a result of the amount and connectivity strength of antibodies, and mirror the internal structure of the immune system. A foreign antigen causes a disturbance in this network structure and induces a restructuring and resizing of the previous structure. To accomplish the network restructuring and resizing, network stimulation and suppression are triggered. In the process of network stimulation, some antibodies are more

[2]He received the Nobel Prize for this theory.
[3]This is the classical viewpoint of immunologists.

strongly involved in recognizing and eliminating foreign antigens than other antibodies. These highly adapted antibodies are transformed into long-life memory antibodies. As the immune system continually produces novel antibodies, and since the total number of antibodies cannot be arbitrary large, non-stimulated and self-reactive antibodies are eliminated. This elimination and renewing process enables the immune system to adapt continually to arbitrary antigenic environments.

III. aiNet Algorithm

The aiNet algorithm summarized in this section mimics the immune network characteristics, i.e. it is an abstracted implementation of the immune network theory explained above. The algorithm is proposed by de Castro and von Zuben [9], [3] and has the general purpose of finding a compressed representation of the input set, by eliminating redundancy in the input set.

The pseudo-code of aiNet is presented on page 3 (see algorithm 1). For the sake of clarity the following notations and functions are used:

- $\mathcal{B}$ collection of antibodies,
- $\mathcal{G}$ collection of antigens,
- $\mathcal{M}$ collection of memorized antibodies,
- $\mathcal{H}$ collection of high affinity antibodies,
- $\mathcal{C}$ collection of cloned antibodies,
- $\mathcal{C}^*$ collection of cloned and mutated antibodies.

Members of each collection are l-dimensional points from $\mathbb{R}^l$ and denoted in small bold letters, e.g. $\mathcal{H} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{|\mathcal{H}|}\}$, where $|\mathcal{H}|$ is the cardinality of $\mathcal{H}$. Moreover, $r \in_R [0,1]$ denotes a random value generated uniformly from the interval $[0,1]$ and capital bold letters denote matrices, e.g. matrix $\mathbf{D}$ with entries $\mathbf{D}_{ij}$. Additionally the following functions are used:

$\texttt{dist}(\mathbf{x}, \mathbf{y})$	Euclidean distance between any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^l$
$\iota\texttt{-affinity}(\mathcal{X}, \mathcal{Y})$	return the inverse distance matrix $\mathbf{D}_{ij} = 1/\texttt{dist}(\mathbf{x}_i, \mathbf{y}_j)\ \forall i, j$
$\texttt{select}(\mathcal{S}, \mathbf{D}, n)$	select n elements from collection $\mathcal{S}$ which have the largest distance in $\mathbf{D}_{ij}\ \forall i, j$
$\texttt{rnd}(\cdot)$	rounding function which gives the nearest integer value
$\texttt{clone}(\mathbf{e}, n)$	duplicate element $\mathbf{e}$, n times
$\texttt{mutate}(\mathbf{e}_1, \mathbf{e}_2)$	change[4] element $\mathbf{e}_1$ according to $\mathbf{e}_1 \leftarrow \mathbf{e}_1 + \alpha\,(\mathbf{e}_2 - \mathbf{e}_1)$, where $\alpha \leftarrow r \cdot 1/\texttt{dist}(\mathbf{e}_1, \mathbf{e}_2)$ and $r \in_R [0,1]$
$\sigma_s\texttt{-remove}(\mathcal{E}, \sigma)$	remove all elements from collection $\mathcal{E}$ whose $\text{dist}(\mathbf{e}_i, \mathbf{e}_j) < \sigma \quad \forall i, j$
$\sigma_d\texttt{-remove}(\mathcal{E}, \mathbf{a}, \sigma)$	remove all elements from collection $\mathcal{E}$ whose $\text{dist}(\mathbf{e}_i, \mathbf{a}) > \sigma \quad \forall i$
$\texttt{concat}(\mathcal{E}_1, \mathcal{E}_2)$	concat collection $\mathcal{E}_1$ with $\mathcal{E}_2$

[4] As we performed our experiments with the existing aiNet implementation taken from [10], the presented mutation function is for the sake of conformity derived from the Matlab source code. The aiNet mutation function presented in [3], [9] is a local guided greedy search, without any random influence, whereas the aiNet mutation function presented in [6] contains a random component. The same with the function for calculating the number of clones. Line (9) is formulated according to [3], [6]. However, in the Matlab source code, $|\mathcal{B}|$ is replaced with a fixed value $c = 10$. We used this fixed value in our experiments.

IV. Nonparametric Density Estimation

In this section we briefly introduce the concept of nonparametric density estimation and show a quality measure in the context of density estimation for data sets proposed by Fukunaga and Hayes [11].

Nonparametric density estimation is a method for estimating an unknown probability density $p(\cdot)$. Given samples $\mathcal{N} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathbb{R}^l$ drawn independently from $p(\cdot)$, the aim is to find an estimator $\widehat{p}_N(\cdot)$ which approximates $p(\cdot)$. Parzen window (also called kernel estimator) is defined as

$$\widehat{p}_N(\mathbf{x}) \quad = \quad \frac{1}{N} \sum_{i=1}^{N} K(\mathbf{x} - \mathbf{x}_i) \tag{1}$$

and is one feasible approach[5] to estimate $p(\cdot)$. A kernel function must satisfy the condition

$$\int_{-\infty}^{\infty} K(\mathbf{x})\, d\mathbf{x} = 1 \tag{2}$$

and therefore the (multivariate) Gaussian kernel function is frequently employed

$$K(\mathbf{x}) \quad = \quad \frac{1}{(2\pi)^{l/2} h^l} \exp\left(-\frac{\|\mathbf{x}\|^2}{2h^2}\right). \tag{3}$$

The window width (also called bandwidth) h controls the *smoothness*, i.e. the influence of the surrounding points $\mathbf{x}_i$, whereas the kernel function $K(\cdot)$ determines the *shape*.

A. A Quality Measure for Reduced Parzen Window

The Parzen window estimation method is computationally expensive, because all stored samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ are used to estimate $\widehat{p}_N(\mathbf{x})$. Fukunaga and Hayes [11] proposed a data reduction algorithm for finding a reduced sample set $\mathcal{R} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_r\} \subset \mathcal{N}$, such that the Parzen density estimates over sample sets $\mathcal{N}$ and $\mathcal{R}$ are as close as possible. The density at $\mathbf{x}$ is then estimated by the reduced sample set as follows

$$\widehat{p}_r(\mathbf{x}) \quad = \quad \frac{1}{r} \sum_{i=1}^{r} K(\mathbf{x} - \mathbf{y}_i). \tag{4}$$

To measure the similarity between $\widehat{p}_N(\mathbf{x})$ and $\widehat{p}_r(\mathbf{x})$ Fukunaga and Hayes suggested to use the entropy

$$\mathfrak{J} \quad = \quad \int \ln\left[\frac{\widehat{p}_r(\mathbf{x})}{\widehat{p}_N(\mathbf{x})}\right] \widehat{p}_N(\mathbf{x})\, d\mathbf{x}. \tag{5}$$

[5] The other popular method is k-nearest neighbor estimator.

Algorithm 1: aiNet

input : $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_{|\mathcal{G}|}\} \in \mathbb{R}^l$,
N (number of maintained antibodies)
σ_s (suppression threshold),
σ_d (natural death threshold),
n (number of best-matching antibodies),
ζ (% of antibodies to be selected as memory),
$\max_g$ (number of maximum generations)
output: $\mathcal{M} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_{|\mathcal{M}|}\} \in \mathbb{R}^l$

begin
  //initialize randomly set $\mathcal{B}$
  $\mathcal{B} \leftarrow \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_N\} \in_R [0,1]^l$
  repeat
    //for each antigen
    for $i \leftarrow 1$ **to** $|\mathcal{G}|$ **do**
      //determine inverse distance from
      //antigen $\mathbf{g}_i$ to each antibody $\mathbf{b}_j$
      $\mathbf{D} \leftarrow \iota$-affinity$(\{\mathbf{g}_i\}, \mathcal{B})$
      //select n antibodies from $\mathcal{B}$ with
      //the smallest distances to $\mathbf{g}_i$
      $\mathcal{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_{|\mathcal{H}|}\} \leftarrow$ select$(\mathcal{B}, \mathbf{D}, n)$
      //determine number of clones for each
      //antibody in $\mathcal{H}$ and create this
      //amount of clones
      $\mathcal{C} \leftarrow \emptyset$
      for $j \leftarrow 1$ **to** $|\mathcal{H}|$ **do**
        $n_c \leftarrow$ rnd$(|\mathcal{B}| - 1/\mathbf{D}_{ij} \cdot |\mathcal{B}|)$
        $\mathcal{C} \leftarrow \mathcal{C} \cup$ clone$(\mathbf{h}_j, n_c)$
      //mutate each clone with a rate
      //proportional to the inverse
      //distance of its parent antibody
      $\mathcal{C}^* \leftarrow \emptyset$
      for $j \leftarrow 1$ **to** $|\mathcal{C}^*|$ **do**
        $\mathcal{C}^* \leftarrow \mathcal{C}^* \cup$ mutate$(\mathbf{c}_j^*, \mathbf{g}_i)$
      //determine inverse distance from
      //antigen $\mathbf{g}_i$ to each cloned antibody $\mathbf{c}_j^*$
      $\mathbf{D}^* \leftarrow \iota$-affinity$(\{\mathbf{g}_i\}, \mathcal{C}^*)$
      //select from $\mathcal{C}^*$, $\xi\%$ of the antibodies
      //with largest distances in $\mathbf{D}^*$ and
      //store them in $\mathcal{M}$
      $\mathcal{M} \leftarrow$ select$(\mathcal{C}^*, \mathbf{D}^*$,rnd$(|\mathcal{C}^*| \cdot \xi/100))$
      //eliminate those memory clones from
      //$\mathcal{M}$ whose distances > σ_d
      σ_d-remove$(\mathcal{M}, \mathbf{g}_i, \sigma_d)$
      //eliminate those memory clones whose
      //distance $< \sigma_s$
      σ_s-remove$(\mathcal{M}, \sigma_s)$
      //concatenate antibody collection
      //with resultant memory clones coll.
      $\mathcal{B} \leftarrow$ concat$(\mathcal{M}, \mathcal{B})$
    //eliminate those antibodies whose
    //distance $< \sigma_s$
    σ_s-remove$(\mathcal{B}, \sigma_s)$
    //generate randomly new antibodies
    $\mathcal{B}_{diversity} \leftarrow \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{N-|\mathcal{B}|}\} \in_R [0,1]^l$
    //concatenate old antibody collection with
    //new diverse antibody collection
    $\mathcal{B} \leftarrow$ concat$(\mathcal{B}, \mathcal{B}_{diversity})$
    $iter \leftarrow iter + 1$
  until $iter = max_g$
  return $\mathcal{M}$
end

If the equality $\widehat{p}_N(\mathbf{x}) = \widehat{p}_r(\mathbf{x})$ holds, then $\mathfrak{J} = 0$; otherwise $\mathfrak{J} < 0$ [11]. Hence a larger entropy implies that $\widehat{p}_r$ is closer to $\widehat{p}_N$. A closer look at (5) reveals, that by interchanging numerator with denominator one obtains the well known Kullback-Leibler divergence with a negative sign

$$-\int \ln\left[\frac{\widehat{p}_N(\mathbf{x})}{\widehat{p}_r(\mathbf{x})}\right] \widehat{p}_N(\mathbf{x})\, d\mathbf{x} = \int \ln\left[\frac{\widehat{p}_r(\mathbf{x})}{\widehat{p}_N(\mathbf{x})}\right] \widehat{p}_N(\mathbf{x})\, d\mathbf{x}. \quad (6)$$

The Kullback-Leibler divergence [12] (also called relative entropy) is a distance[6] measure between two probability distributions.

In order to discretize and simplify (5), Fukunaga and Hayes used the expectation of (5) taken with respect to $\widehat{p}_N(\mathbf{x})$, i.e.

$$\mathrm{E}\left[\ln\left[\frac{\widehat{p}_r(\mathbf{x})}{\widehat{p}_N(\mathbf{x})}\right]\right] \approx \sum_{i=1}^{N} \ln\left[\frac{\widehat{p}_r(\mathbf{x}_i)}{\widehat{p}_N(\mathbf{x}_i)}\right] \underbrace{\widehat{p}_N(\mathbf{x}_i)}_{\rho}. \quad (7)$$

By replacing ρ by $1/N$ one obtains a simplified approximation of (5), i.e.

$$\widetilde{J} \;=\; \frac{1}{N}\sum_{i=1}^{N} \left[\ln \widehat{p}_r(\mathbf{x}_i) - \ln \widehat{p}_N(\mathbf{x}_i)\right]. \quad (8)$$

Finally by substituting (1) and (4) in (8) one obtains

$$J = \frac{1}{N}\sum_{i=1}^{N}\left[\ln\frac{1}{r}\sum_{j=1}^{r} K(\mathbf{x}_i - \mathbf{y}_j) - \ln\frac{1}{N}\sum_{j=1}^{N} K(\mathbf{x}_i - \mathbf{x}_j)\right]. \quad (9)$$

They noticed that $J \leq 0$ (and also $\widetilde{J} \leq 0$) is no longer guaranteed, but argued [11] that J represents a significantly simplified closeness measure. Furthermore they reported good and reasonable experimental results, when applying this closeness measure.

In our experiments we will use (9) as a closeness measure between the input data set and the aiNet output data set. To be more precise, the probability densities $\widehat{p}_N(\cdot)$ and $\widehat{p}_r(\cdot)$ are estimated by means of a Parzen window estimator over the input set $\mathcal{N}$ and output set $\mathcal{R}$, respectively. The closeness between $\widehat{p}_N(\cdot)$ and $\widehat{p}_r(\cdot)$ is then determined by means of term (9) and gives a measure on the reduction quality of input set and output set. The larger the value of J, the closer $\widehat{p}_r(\cdot)$ to $\widehat{p}_N(\cdot)$ and consequently the more similar (in a probabilistic sense) input set to output set.

V. Data Sets and Experiments

In order to quantify and measure the closeness between input set and aiNet output set $\mathcal{M}$, four artificially[7]

[6]The Kullback-Leibler divergence is not a *true metric* because it satisfies not all metric properties and therefore is often termed divergence instead of distance.

[7]A real-world benchmark for this kind of problem could be image compression. However, the quantification of the compression quality is frequently judged by subjective viewers and gives no objective quality measure.

generated (input) data sets with different characteristics are created (see figures 3(a),4(a),5(a),6(a)). Each data set contains 400 points, sampled from different probability distributions. The data sets are generated with the open source program R [13]. Since some data sets are intricate to characterize, we have shown the R source code used[8] (see Fig. 1), but also give a brief description. Data set (1) is generated from a two dimensional Gaussian distribution (μ, I), where

$$\mu = (0,0) \text{ and } I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Data set (2) is generated by a mixture of six Gaussian distributions with different mean vectors and covariance matrices. Data set (3) is generated by a mixture of two Gaussian distributions. The first has the same parameters as distribution of data set (1), the second the parameters $1/4(\mu, I)$, i.e. it is a "dense" Gaussian inside a Gaussian. Data set (4) contains of points which are consecutively calculated from a sine/cosine function. Additionally to each sampled point, noise in the form of a Gaussian distribution with $\mu = 0$ and $\sigma = 0.2$ is added.

As the cardinality of the output set of aiNet is controlled by parameter σ_s [3], the experiments are performed with different parameter settings $\sigma_s = \{0.2, 0.1, 0.05, 0.01, 0.005\}$ for obtaining different output set reductions. The other algorithm parameters are set as follows:

$$\sigma_d = 1, \quad n = 4, \quad N = 10, \quad \zeta = 20\,\%, \quad \max_{gen} = 10.$$

These parameter values are chosen, because reasonable and good results on the compression quality for different data sets is reported in [3] when using these parameter settings. For the sake of the completeness, we have to report that the min-max normalization component from the aiNet algorithm was removed, as the optimal kernel bandwidth $h = 0.96$ is derived for two dimensional unscaled data sets [15].

In the performed experiments, 400 points from each of the four probability distributions are sampled, and used as the input data set. Since aiNet is a non-deterministic randomized algorithm, we performed 50 runs on each input set for each parameter σ_s. After all simulation runs are completed, the entropy values according to (9) are computed for each σ_s and each associated input/output set. The mean (denote as J_μ) and standard deviation (denote as J_σ) of the entropy value over all runs is finally computed. Moreover, the cardinality of each computed output set is determined and likewise averaged (denoted as $|\mathcal{M}|_\mu$ and $|\mathcal{M}|_\sigma$). As it is difficult to quantify the magnitude of J_μ — (recall: a larger entropy implies that probability density $\widehat{p}_r$ is closer to $\widehat{p}_N$, but how "good" is a certain magnitude of J_μ) — a *reference* value is computed. This is straightforward, because we know the underlying probability distributions (see figure 1). More specifically, $|\mathcal{M}|_\mu$ many points from each probability distribution are sampled and the entropy value for each associated input set is computed. This process is also repeated 50 times for obtaining comparative reference values denote as J_μ^{ref} (mean) and J_σ^{ref} (standard deviation). According to (9) these reference values must be close zero, because the sampled points in the input/output set are generated from the same probability distribution and therefore the estimated densities should be nearly identical. This fact can be verified in table I (see J_μ^{ref} row of each data set).

[8]Source code of functions `dataset2` and `dataset3` is from [14].

```
##############################
dataset1 <- function(m) {
  x <- c(rnorm(m), rnorm(m))
  y <- c(rnorm(m), rnorm(m))
  return(data.frame(x,y));
}
##############################
dataset2 <- function(m){
  z<- c(rbinom(1,m,1/3))
  z[2]<-rbinom(1,m-z[1],1/2)
  v<- c(rbinom(1,m,1/3))
  v[2]<-rbinom(1,m-v[1],1/2)
  x <- c(rnorm(z[1])/2+2, rnorm(z[2])+6,
         rnorm(m-z[1]-z[2])/2+3,
         rnorm(v[1])/1+4, rnorm(v[2])/3+1,
         rnorm(m-v[1]-v[2])+8)
  y <- c(rnorm(z[1])/2+1, rnorm(z[2])/3+1,
         rnorm(m-z[1]-z[2])/2+1.5,
         rnorm(v[1])/3+2, rnorm(v[2])/3+2,
         rnorm(m-v[1]-v[2])/3+1.5)
  return(data.frame(x,y))
}
##############################
dataset3 <- function(m){
  x <- c(rnorm(m)/4, rnorm(m))
  y <- c(rnorm(m)/4, rnorm(m))
  return(data.frame(x,y))
}
##############################
dataset4 <- function(m,noise=0.2) {
  x <- c(1:2*m);
  y <- c(1:2*m);
  for (i in 1:m) {
    x[i] <- (i/m) * pi;
    y[i] <- sin(x[i]) + rnorm(1,0,noise);
  }
  for (j in 1:m) {
    x[m+j] <- (j/m + 1/2) * pi;
    y[m+j] <- cos(x[m+j]) + rnorm(1,0,noise);
  }
  return(data.frame(x,y))
}
```

Fig. 1. R source code for generating data sets from different probability distributions

VI. Results

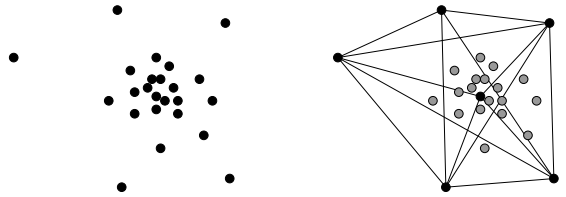
The obtained simulation results are shown in table I and figures (3)-(6). First, one can notice, for parameter values $\sigma_s = \{0.01, 0.005\}$ the output set contains more points than the input set. That means, one has to choose a suitable parameter σ_s for obtaining the proper cardinality (compression quantity) of the output set — a similar result is reported in [3]. Second, one can see that aiNet outputs unsatisfiable results for $\sigma_s = 0.2$, on data sets (1)-(3). These data sets contain "dense" point regions which are not properly captured by aiNet (see figures 3(b),4(b),5(b)). This can also be verified by means of results shown in table I. The difference between J_μ and J_σ is significantly high especially for data set (3), moreover this is also clear identifiable when comparing figures 5(a) and 5(b).

This large dissimilarity between input set and output set can be explained by investigating the dependency of parameter σ_s and the optimization criterion of aiNet. The aiNet optimization criterion is guided by the three functions:

$$\begin{array}{l} 1\text{: } \mathbf{D}^* \leftarrow \iota\texttt{-affinity}(\{\mathbf{g}_i\}, \mathcal{C}^*) \\ 2\text{: } \mathcal{M} \leftarrow \texttt{select}(\mathcal{C}^*, \mathbf{D}^*, \texttt{round}(|\mathcal{C}^*| \cdot \xi/100)) \\ 3\text{: } \sigma\texttt{-remove}(\mathcal{M}, \sigma_s) \end{array} \quad (10)$$

Statement in line 1 and 2 causes the finding of all points in $\mathcal{M}$ which are close (with regard to the Euclidean distance) to points in the input set. Statement in line 3 causes the removal of all points in $\mathcal{M}$ whose neighborhood[9] distance is $< \sigma_s$. That means, the final output set $\mathcal{M}$, consists only of points whose distance is $\geq \sigma_s$ to all points (see figure 2). The parameter σ_s not only controls the compression ratio, but also the neighborhood distances of each point in the output set. It is obvious that, the smaller σ_s the more points aiNet will find, which satisfies criterion (10). For a very small σ_s, e.g. $\sigma_s = 0.005$, all points from the input set are satisfying the criterion and therefore one obtains a nearly identical input set. Furthermore, one can also verify (see figures 3(f), 4(f), 5(f), 6(f)) that for such a small σ_s the output set contains several points that overlap, because this is the only way to satisfy criterion (10) with more points in the output set than in the input set. From this insight it is now clear, why aiNet outputs good and reasonable results on data set (4). Recall, data set (4) contains of sampled points without any dense point regions. As aiNet finds points, all which have to satisfy criterion (10), and points in data set (4) properly satisfy this criterion because they are uniformly distributed, one obtains good and reasonable results, especially for $\sigma_s = 0.2$.

[9]Distance to adjacent points.



(a) Given some points from an unknown probability distribution

(b) The final output set $\mathcal{M}$ satisfies the criterion that all points in $\mathcal{M}$ have neighborhood distances $\geq \sigma_s$

Fig. 2. aiNet optimization criterion

TABLE I
aiNet results of entropy values and cardinality of the output set

data set 1					
σ_s	0.2	0.1	0.05	0.01	0.005
J_μ	-0.117852	-0.031902	-0.016569	-0.010274	-0.009244
J_σ	0.006373	0.002765	0.002818	0.003604	0.003484
$\lvert\mathcal{M}\rvert_\mu$	130.04	284.12	379.94	470.84	484.74
$\lvert\mathcal{M}\rvert_\sigma$	4.69	4.92	5.33	7.48	9.44
J_μ^{ref}	-0.004868	-0.002365	-0.005553	-0.003208	-0.000568
J_σ^{ref}	0.018443	0.011041	0.012714	0.011754	0.011975

data set 2					
σ_s	0.2	0.1	0.05	0.01	0.005
J_μ	-0.072474	-0.019583	-0.009615	-0.003313	-0.002143
J_σ	0.008152	0.001887	0.002239	0.002914	0.002816
$\lvert\mathcal{M}\rvert_\mu$	144.66	293.26	388.48	483.08	500.36
$\lvert\mathcal{M}\rvert_\sigma$	4.39	3.85	6.76	10.02	11.84
J_μ^{ref}	-0.004136	-0.006214	-0.005241	-0.003856	-0.005415
J_σ^{ref}	0.007414	0.010635	0.007737	0.007215	0.009109

data set 3					
σ_s	0.2	0.1	0.05	0.01	0.005
J_μ	-0.235993	-0.133765	-0.076392	-0.024572	-0.024088
J_σ	0.008587	0.004041	0.005003	0.003903	0.003718
$\lvert\mathcal{M}\rvert_\mu$	100.48	197.58	287.90	432.04	445.72
$\lvert\mathcal{M}\rvert_\sigma$	3.08	3.64	5.20	10.06	7.78
J_μ^{ref}	-0.004953	-0.003212	-0.000835	0.000213	-0.000147
J_σ^{ref}	0.015143	0.013324	0.011609	0.009977	0.010159

data set 4					
σ_s	0.2	0.1	0.05	0.01	0.005
J_μ	-0.014436	-0.002651	-0.003337	0.001812	0.003771
J_σ	0.008660	0.002786	0.002448	0.002337	0.002043
$\lvert\mathcal{M}\rvert_\mu$	49.66	168.70	305.20	432.44	454.04
$\lvert\mathcal{M}\rvert_\sigma$	3.66	4.70	6.57	6.76	8.73
J_μ^{ref}	0.000599	-0.001215	-0.000293	0.000213	-0.000147
J_σ^{ref}	0.004204	0.002903	0.002378	0.009977	0.010159

Furthermore, results in table I reveal that the difference between J_μ and J_μ^{ref} for $\sigma_s \to 0$ tends to be zero. However for very small values of σ_s the output set contains more points than the input set and this consequently results in a data expansion rather than in a compression. Moreover for very small values of σ_s, the aiNet algorithm outputs nearly the identical input set (with many overlapping points). That means, aiNet

seems to be inappropriate as a technique to sample points from an estimated probability distribution [16].

To summarize, aiNet guided its search of finding the reduced output set by means of the optimization criterion (10). The criterion, however, is inappropriate for non-uniformly distributed data sets, because it induces the finding of points whose distance to adjacent points must be greater-equal than a pre-defined threshold. It is clear, that this criterion is best to satisfiable when the input set is uniformly distributed.

VII. Conclusions

We have experimentally investigated the compression quality of the aiNet algorithm. A closeness measure between input set and compressed output set was presented. This closeness measure is based on the Parzen window estimation and the Kullback-Leibler divergence. Experiments reveal that aiNet produced reasonable results on the uniformly distributed data set, but poor results on the non-uniformly distributed data sets, i.e. data sets which contain dense point regions. This unsatisfactory result on non-uniformly distributed data sets was caused by the optimization criterion of aiNet.

As aiNet is an algorithm whose search is guided in an evolutionary fashion, it should be feasible to modify the old optimization criterion according to term (9) to overcome this problem.

Acknowledgments

Work reported in this paper is supported in part by AFOSR's grant, FA9550-04-1-0159.

References

[1] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer Verlag, 2002.

[2] E. Hart and J. Timmis, "Application areas of AIS: Past, present and future," in *Proceedings of the 4th International Conference on Artificial Immune Systems (ICARIS)*, ser. Lecture Notes in Computer Science, vol. 3627. Springer-Verlag, 2005, pp. 316–329.

[3] L. N. de Castro and F. J. V. Zuben, "aiNet: An artificial immune network for data analysis," in *Data Mining: A Heuristic Approach*, H. A. Abbass, R. A. Sarker, and C. S. Newton, Eds. Idea Group Publishing, 2001, ch. 12, pp. 231–259.

[4] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.

[5] N. Tang and R. V. Vemuri, "An artificial immune system approach to document clustering," in *Proceedings of the 20th ACM Symposium on Applied Computing (SAC)*. ACM Press, 2005, pp. 918–922.

[6] L. N. de Castro, "The immune response of an artificial immune network (aiNet)," in *Proceedings of Congress On Evolutionary Computation (CEC)*. IEEE Press, 2003, pp. 146–153.

[7] N. K. Jerne, "Towards a network theory of the immune system," *Annales d' immunologie*, vol. 125C, pp. 373–389, 1974.

[8] F. J. Varela and A. Coutinho, "Second generation immune networks," *Immunology Today*, vol. 12, no. 5, pp. 159–166, 1991.

[9] L. N. de Castro and F. J. V. Zuben, "An evolutionary immune network for data clustering," in *Proceedings of the 6th Brazilian Symposium on Neural Networks*. IEEE Computer Society, 2000, pp. 84–89.

[10] L. N. de Castro, "aiNet implementation in matlab," 2000, ftp://ftp.dca.fee.unicamp.br /pub/docs/vonzuben/lnunes/demo.zip.

[11] K. Fukunaga and R. R. Hayes, "The reduced parzen window classifier," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 11, no. 4, pp. 423–425, 1989.

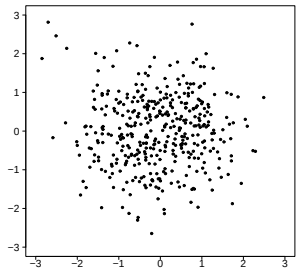
[12] S. Kullback, *Information Theory and Statistics*. John Wiley & Sons, 1959.

[13] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2006, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org

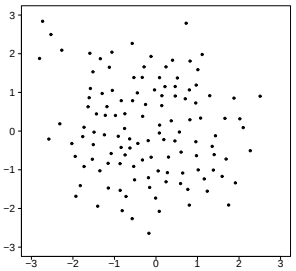
[14] D. Metzler, "Algorithmisches lernen in der bioinformatik," 2004, lecture Notes (http://www.informatik.uni-frankfurt.de/∼metzler/VorlesungSS04/).

[15] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

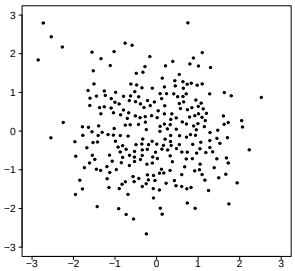
[16] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.



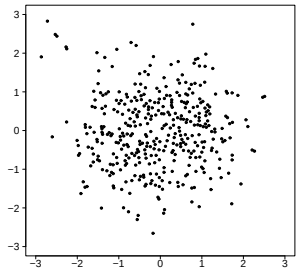
(a) data set (1) consists of 400 points sampled from a Gaussian distribution $\mu = (0, 0)$ and unit matrix I



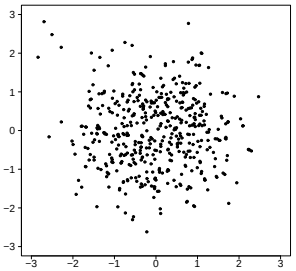
(b) aiNet output of simulation run 10, with $\sigma_s = 0.2$ and resulting output $|\mathcal{M}| = 126$



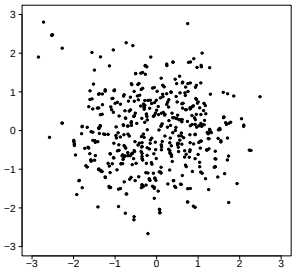
(c) aiNet output of simulation run 10, with $\sigma_s = 0.1$ and resulting output $|\mathcal{M}| = 276$



(d) aiNet output of simulation run 10, with $\sigma_s = 0.05$ and resulting output$|\mathcal{M}| = 386$

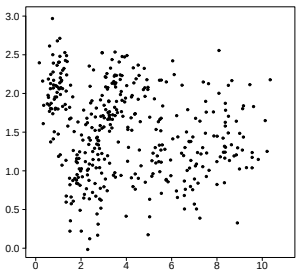


(e) aiNet output of simulation run 10, with $\sigma_s = 0.01$ and resulting output $|\mathcal{M}| = 471$

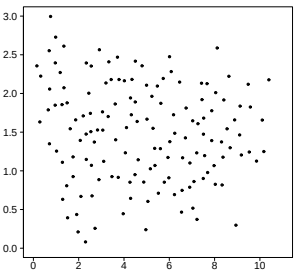


(f) aiNet output of simulation run 10, with $\sigma_s = 0.005$ and resulting output$|\mathcal{M}| = 496$

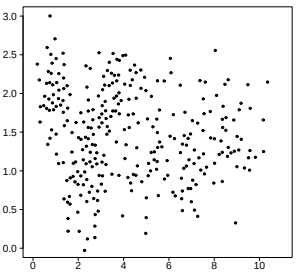
Fig. 3. Data set (1) and the resulting aiNet output data sets with different parameter settings σ_s



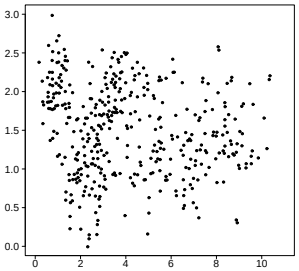
(a) data set (2) consists of 400 points sampled from mixtures of Gaussian distributions with different mean vectors and covariance matrices



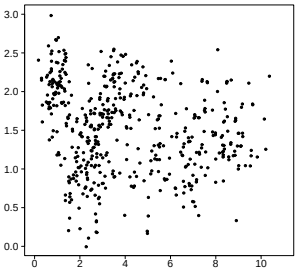
(b) aiNet output of simulation run 23, with $\sigma_s = 0.2$ and resulting output $|\mathcal{M}| = 144$



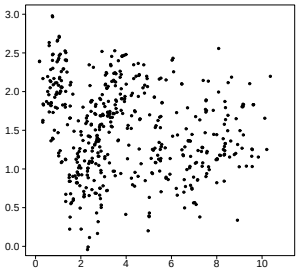
(c) aiNet output of simulation run 23, with $\sigma_s = 0.2$ and resulting output $|\mathcal{M}| = 296$



(d) aiNet output of simulation run 23, with $\sigma_s = 0.2$ and resulting output $|\mathcal{M}| = 401$

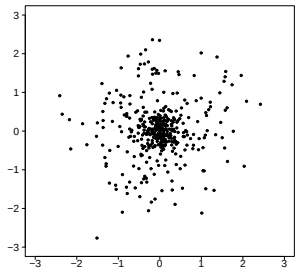


(e) aiNet output of simulation run 23, with $\sigma_s = 0.2$ and resulting output $|\mathcal{M}| = 491$

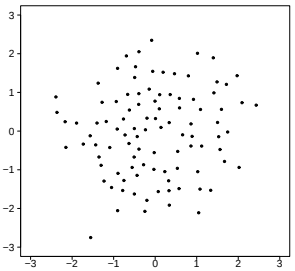


(f) aiNet output of simulation run 23, with $\sigma_s = 0.2$ and resulting output $|\mathcal{M}| = 503$

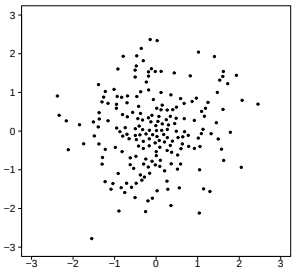
Fig. 4. Data set (2) and the resulting aiNet output data sets with different parameter settings σ_s



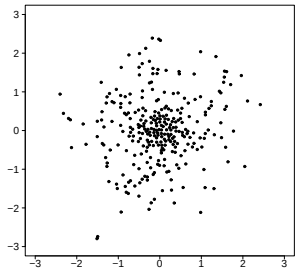
(a) data set (3) consists of 400 points sampled from a "dense" Gaussian distribution inside a Gaussian distribution



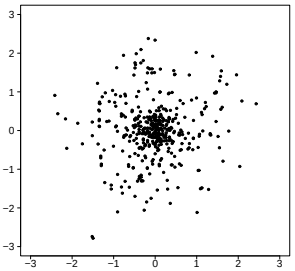
(b) aiNet output of simulation run 17, with $\sigma_s = 0.2$ and resulting output $|\mathcal{M}| = 99$



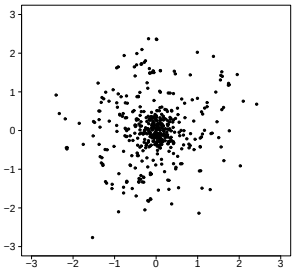
(c) aiNet output of simulation run 17, with $\sigma_s = 0.1$ and resulting output $|\mathcal{M}| = 200$



(d) aiNet output of simulation run 17, with $\sigma_s = 0.05$ and resulting output $|\mathcal{M}| = 286$

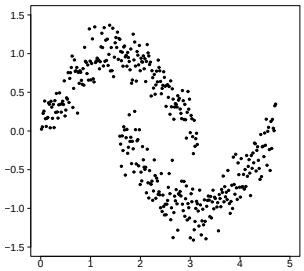


(e) aiNet output of simulation run 17, with $\sigma_s = 0.01$ and resulting output $|\mathcal{M}| = 420$

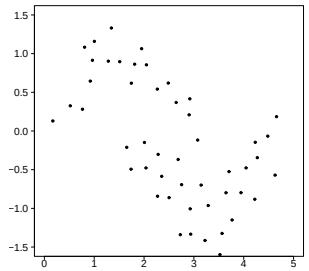


(f) aiNet output of simulation run 17, with $\sigma_s = 0.001$ and resulting output $|\mathcal{M}| = 436$

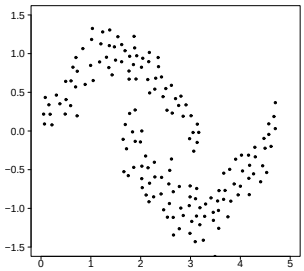
Fig. 5. Data set (3) and the resulting aiNet output data sets with different parameter settings σ_s



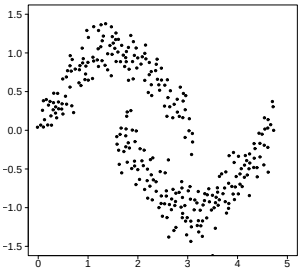
(a) data set (4) consists of 400 points sampled consecutively from a sinus/cosine function, with added noise in form of a Gaussian distribution with $\mu = 0$ and $\sigma = 0.2$



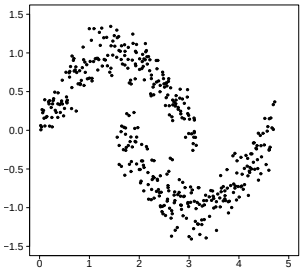
(b) aiNet output of simulation run 25, with $\sigma_s = 0.2$ and resulting output $|\mathcal{M}| = 49$



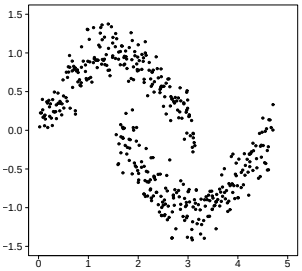
(c) aiNet output of simulation run 25, with $\sigma_s = 0.1$ and resulting output $|\mathcal{M}| = 164$



(d) aiNet output of simulation run 25, with $\sigma_s = 0.05$ and resulting output $|\mathcal{M}| = 306$



(e) aiNet output of simulation run 25, with $\sigma_s = 0.01$ and resulting output $|\mathcal{M}| = 431$



(f) aiNet output of simulation run 25, with $\sigma_s = 0.005$ and resulting output $|\mathcal{M}| = 451$

Fig. 6. Data set (4) and the resulting aiNet output data sets with different parameter settings σ_s