Applying Continuous Action Reinforcement Learning Automata(CARLA) to Global Training of Hidden Markov Models

Jahanshah Kabudian, Mohammad Reza Meybodi, and Mohammad Mehdi Homayounpour

Department of Computer Engineering and Information Technology,
AmirKabir University of Technology (Tehran PolyTechnic),
Tehran, Iran
{kabudian, meybodi, homayoun}@ce.aut.ac.ir

Abstract

In this research, we have employed global search and global optimization techniques based on Simulated Annealing (SA) and Continuous Action Reinforcement Learning Automata (CARLA) for global training of Hidden Markov Models. The main goal of this paper is comparing CARLA method to other continuous global optimization methods like SA. Experimental results show that the CARLA outperforms SA. This is due to the fact that CARLA is a continuous global optimization method with memory and SA is a memoryless one.

1. Introduction

HMM is one of the most powerful methods for processing and modeling stochastic processes and sequences. The most popular method for training HMM is Baum-Welch (BW) method that is a local search method and like the other local methods suffers from local optima problem. To cope with this problem, global search and optimization methods are used for training HMM which can easily escape from local optima. The main goal of this paper is a comparison of CARLA and SA as two global search methods for training HMMs.

In section two, we introduce HMM. In section three, BW training method is briefly explained. In section four, SA-based and CARLA-based global search method is described. Section five represents the results of the conducted experiments, and finally, section six concludes this paper.

2. Hidden Markov Model

HMM is a finite-state system with a number of states (*N*). This system changes its states probabilistically from a state to another state according to state transition probabilities. HMM has two types[1]: Continuous Density HMM and Discrete Density HMM. Discrete Density HMM is rarely used for modeling and processing continuous observations. Assume that an observation sequence with time length *T* is to be generated by this model. Observation sequence is a sequence of observation vectors as follows:

$$O = \{o_1 o_2 ... o_t ... o_T\}$$

Each of the observation vectors o_t is a D-dimensional vector and *t* is time index. Starting the time, for t=1, HMM goes into a state (e.g. *i*) with corresponding initial state probability(π_i) (π_i's are elements of initial state probabilities vector π). In state *i*, vector o_t is generated using a multidimensional PDF $b_i(o_t)$. If the model is in state *i* at time t-1, then it enters into state *j* at time t with probability a_{ij} (a_{ij}'s are elements of state transition probabilities matrix A).

An observation sequence *O* is generated by HMM λ with probability $P(O|\lambda)$. This probability is computed using efiicient methods like Forward or Backward Procedure[1]. In continuous density HMM, multidimensional PDF in state *i*, i.e. $b_i(o)$, is usually a mixture of Gaussian (Normal) multidimensional PDF like:

$$b_i(o) = \sum_{m=1}^{M} c_{im} N(o, \mu_{im}, C_{im})$$

$N(o, \mu, C)$ is a D-dimensional Gaussian (Normal) PDF with Mean Vector μ and Covariance Matrix *C* of this form:

$$N(o, \mu, C) = \frac{1}{\sqrt{(2\pi)^D . |\det(C)|}} . \exp(-\frac{1}{2}(o-\mu)^T C^{-1} (o-\mu))$$

Each of the observation vectors o_t is a $D \times 1$ column vector. It is necessary that volume (hypervolume) under this mixture PDF be equal to 1, therefore:

$$\sum_{m=1}^{M} c_{im} = 1$$

M is number of Gaussian PDFs in a mixture PDF of each state. Also, the following constraints must be satisfied:

$$\sum_{i=1}^{N} \pi_i = 1 \quad \text{and} \quad \sum_{j=1}^{N} a_{ij} = 1$$

C^{-1} is inverse of Matrix *C* and $|det(C)|$ is the absolute value of the determinant of Matrix *C*. HMM model λ with parameters π, A and b() is shown as follows:

$$\lambda = (\pi, A, b(\,))$$

b() in this triplet is a set of parameters including weights (c_{im}), Mean vectors (μ_{im}) and Covariance matrices (C_{im}) of Gaussian PDFs.

3. Baum-Welch Algorithm

In real-world applications of HMM, we have usually one or more training observation sequences generated by an unknown model λ. The goal is estimating

parameters of HMM model, i.e. π, A and b(). The most popular method for training HMMs is BW algorithm. The goal of BW algorithm is finding a model λ^* such that the probability of generating observation sequence O given λ is maximized[1]:

$$\lambda^* = \arg \max_{\lambda} P(O|\lambda)$$

As we know, the BW method is trapped in local maxima. We use global optimization and search methods to escape from these maxima and find the global one of the HMM training problem.

4. Global Optimization and Search Methods

To solve hard optimization and search problems in a near-optimal way, global search and optimization methods are used. Simulated Annealing (SA)[2], Genetic Algorithms (GA), Evolutionary Strategies (ES) and Evolutionary Programming (EP)[3] are of this category. A wide variety of global optimization methods exists[4]. Recently, a new global optimization method called Continuous Action Reinforcement Learning Automaton (CARLA) has been proposed for optimization in continuous domains[5,6]. Some of these methods (like SA) have a finer resolution compared to some other methods (like GA). From the global search methods, we chose SA for comparing to CARLA.

4.1. Simulated Annealing

Simulated Annealing is a global search (optimization) method which is based on the laws of thermodynamics. If the method follows the recommended Temperature Schedule for reducing temperature, then it guarantees finding the global optimum from a mathematical point of view. This method has been applied in many applications for both continuous and discrete problems. SA follows the following procedure for optimization: A random initial point and a high initial temperature is selected. In each iteration, it generates a new point around the previous point using a PDF g(x) called Generating Function. It calculates cost function E(x) for this point, and probabilistically accepts this point with probability h(x) which is called Acceptance Function. If the new point is better than the previous one, the probability of acceptance will be more than the probability of rejection. Of course, in the beginning of the search, temperature is high and the search region is wide, and the acceptance and rejection probabilities are almost equal (0.5). When temperature decreases and the algorithm approaches optimum points, search width is reduced, and accuracy and resolution of the search are increased. In the following section, we explain the standard version of SA, i.e. Boltzmann Annealing (BA).

4.1.1. Boltzmann Annealing

Boltzmann Annealing algorithm as follows:

1. A high initial temperature(T_1) and a random initial point (x_1) is selected.
2. $k=1$, $x^*=x_1$
3. A random new point x is generated around x^* using generating function $g(x,x^*)$

$$g(x,x^*) = (2\pi.T_k)^{\frac{-D}{2}}.\exp\left(-\frac{\| x-x^* \|^2}{2T_k}\right)$$

4. Cost function for point x, i.e. $E(x)$ is calculated.
5. New point x is accepted with probability $h(x)$. (If it is accepted then $x^*=x$, else x^* doesn't change)

$$\Delta E = E(x) - E(x^*)$$

$$h(x) = \frac{1}{1+\exp\left(\frac{\Delta E}{T_k}\right)}$$

6. $k=k+1$
7. Temperature is reduced according to this temperature schedule $T_k = \frac{T_1}{\ln k}$.
8. If the stop condition is met then algorithm stops, otherwise it goes to step 3.

In the above algorithm, D is number of parameters of the cost function E(x) and $\|x-x^*\|$ is the Euclidean distance between two D-dimensional vectors x and x^*. Acceptance function $h(x)$ is usually a Barker function (like the form mentioned in step 5 of the above algorithm), but Metropolis function can also be used. In standard SA algorithm, generating function is of a Gaussian type and the corresponding SA is called Boltzmann Annealing (BA). If temperature schedule (step 7 of algorithm) is not faster than inverse-logarithmic schedule, then the method statistically guarantees that it finds the global optimum in an infinite number of iterations. The BA algorithm with corresponding temperature schedule is very slow and needs a very high number of iterations for approaching global optimum. If we use SA for global optimization and global training of HMMs, the cost function can be defined as follows:

$$E = -Log\, P(O|\lambda)$$

4.2. CARLA

One of the methods which is used in optimization and control, is Learning Automaton (LA)[7]. LA is a model which interacts with an environment. It applies an action to the environment and environment evaluates this action. According to the inputs that it receives from the environment, LA corrects its action selection mechanism with a reinforcement signal, and again applies new action to the environment. This process is repeated infinitely or until a predefined goal is obtained. In standard models of LA, action is a discrete variable. Recently, a Continuous Action Reinforcement Learning Automaton (CARLA) with continuous action has been presented[5,6]. CARLA can be employed in the environments with continuous parameters for global

optimization tasks. It is a model for reinforcement learning in continuous-parameter environments. CARLA generates new value of action using an action generation function called action generation PDF. If the number of system (environment) parameters is more than one, e.g. in minimization of a multidimensional cost function, then a population of CARLA automata interact with the system in parallel and each automaton undertakes optimization of one of the parameters of the system. These automata haven't any relationship or interaction with each other. They interact only with the system (environment), and their indirect interaction with each other is through the system. Each automaton generates a new action (new value of the parameter in the corresponding dimension) using its action generation PDF. System (Environment), evaluates the set of actions of automata and sends reinforcement signals to automata. Using these reinforcement signals, each CARLA automaton corrects its action generation PDF. This process is repeated until a predefined criterion is achieved. One of the advantages of this method is that the CARLA has memory and saves many of the previous points in its memory with respect to their relative importance. CARLA method operates in this way:

1. At first, action generation PDFs are initialized with uniform distribution PDFs in the range of system parameters.
2. Using action generation PDFs, each CARLA generates a new value of action (parameter) *r*.
3. Environment response is calculated (cost function in optimization tasks).
4. A reinforcement signal is calculated with respect to the goodness of the environment response or cost function.
5. Action generation PDFs are corrected using these reinforcement signals.
6. If stop criterion is not met, it goes to step 2, otherwise the algorithm is stopped.

Assume variable x is in the range $[x_{min}, x_{max}]$. At first (in the first iteration), action generation PDF is initialized with this uniform PDF:

$$f(x,1) = \frac{1}{x_{\max} - x_{\min}}$$

In n-th iteration, action r is generated using action generation PDF $f(x,n)$:

$$F(r,n) = \int_{x_{\min}}^{r} f(x,n)\,dx = z(n)$$

For generating an action, firstly a uniform random number $z(n)$ between [0,1] is generated. Then, the action r is selected such that cumulative distribution function $F(r,n)$ be equal to $z(n)$. Cumulative distribution function $F(r,n)$ is computed by numerical integration in the interval $[x_{min}, r]$. Samples of $f(x,n)$ is stored in memory and it is descretized. Wherever the value of $f(x,n)$ is not available, its value is calculated using linear interpolation. After generating new value of action, i.e. r, this action is applied to the environment and environment response $J(n)$(cost function in optimization tasks) is obtained. A reinforcement signal β is calculated for correcting action generation PDF:

$$\beta(n) = \min\left(\max\left(0, \frac{J_{med} - J(n)}{J_{med} - J_{\min}}\right), 1\right)$$

Reinforcement signal β must be in the range [0,1]. J_{med} and J_{min} are Median and Minimum of costs of R previously visited points (e.g. R=500). If action generation PDF in n-th iteration be $f(x,n)$, then the action generation PDF in $(n+1)$-th iteration is corrected in this way:

$$f(x,n+1) = \begin{cases} \alpha[f(x,n) + \beta(n)H(x,r)] & \text{if} \quad x \in [x_{\min}, x_{\max}] \\ 0 & \text{else} \end{cases}$$

$H(x,r)$ is a gaussian function with mean r of this form:

$$H(x,r) = \frac{g_h}{(x_{\max} - x_{\min})} . \exp\left(-\frac{1}{2} . \frac{(x-r)^2}{(g_w (x_{\max} - x_{\min}))^2}\right)$$

Parameters g_h and g_w determine the speed and the relative resolution (accuracy) of the learning or the search process. We set these parameters to 0.3 and 0.02 respectively[5,6]. In this formulation, α is a normalization coefficient which must be applied to the corrected PDF such that the hypervolume under new action generation PDF, i.e. $f(x,n+1)$ be equal to one.

5. Experiments

In this section, we present the results of experiments. Our goal is training an HMM with Maximum Likelihood (ML) criterion by global methods. The HMM to be trained is a continuous density HMM with three states and three Gaussian PDFs per state with diagonal covariance matrices and two-dimensional observation vectors. Therefore the number of parameters will be:

$P=N(1+N+M(2D+1))$

With N=3, M=3 and D=2, the number of HMM parameters is 57. Training is performed in multiple observation sequences case[1], and K=10 observation sequences with time length T=20 are available. We want to maximize $P(O|\lambda)$. O is the set of K observation sequences for training, and λ is the set of parameters of the HMM model. As we know, HMM training is a constrained optimization problem with these constraints:

$$v_{ii} > 0 \;,\quad \sum_{m=1}^{N} c_{im} = 1 \;,\quad \sum_{j=1}^{N} a_{ij} = 1 \quad \text{for all } i$$

Where v_{ii} is diagonal element of covariance matrix corresponding to i-th dimension. For converting this

constrained optimization problem to an unconstrained one, we used this heuristic mappings:

$$a_{ij} = \frac{\exp(a'_{ij})}{\sum_{j=1}^{N} \exp(a'_{ij})}$$

$$c_{im} = \frac{\exp(c'_{im})}{\sum_{m=1}^{M} \exp(c'_{im})}$$

$$v_{ii} = \exp(v'_{ii})$$

That is, a method like SA, optimizes the parameters $\pi'_i, a'_{ij}, c'_{im}, v'_{ii}$ in an unconstrained manner, but the converted and the mapped versions of these parameters, i.e. π_i , a_{ij} , c_{im} and v_{ii} , satisfy the constraints of the HMM training problem. Now, we explain the experiments.

5.1. SA

In this section, SA method is used for global training of Hidden Markov Models. Suitable initial temperatures in different dimensions were set. Figure 1 shows a sample performance for BA method. Since in the BA method, the rate of temperature reduction is very slow even in 10,000 iterations, we cannot reach fine resolution. Therefore, search area remains wide and the change in the cost function will be sudden and considerable even in the end of the search process.

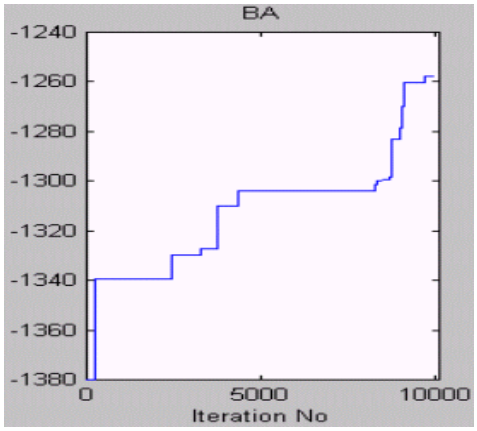

Figure 1. Sample performance of SA (Log Probability)

5.2. CARLA

In this method, Max and Min of HMM parameters were set with respect to the training data. Each of CARLA automata tries to correct and adapt its action generation PDF, and finally its ideal action generation PDF is a Gaussian-like PDF centered on the optimum value of parameter in that dimension. Figure 2 shows the shape of the action generation PDF after 1000, 2000, 3000 and 10,000 iterations.

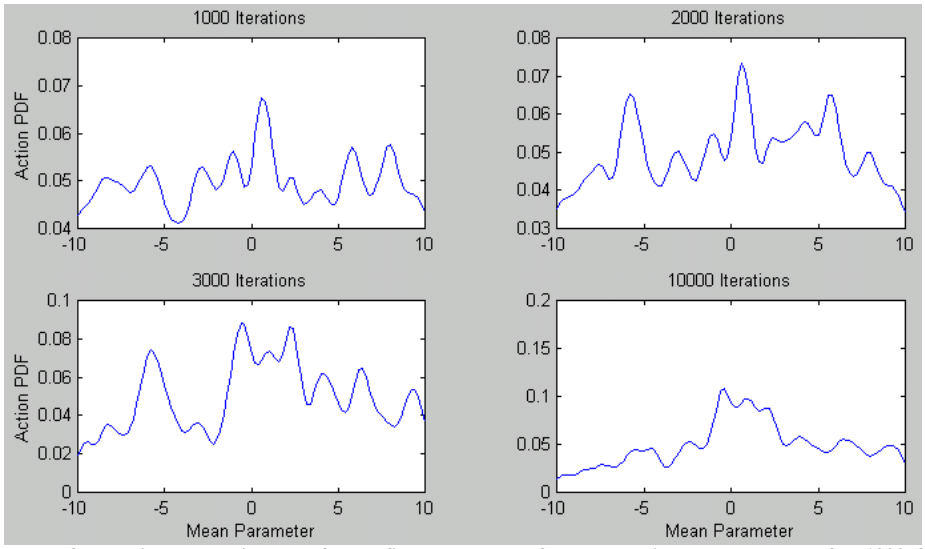

Figure 2. The shape of the action generation PDF for the first parameter of an HMM with 57 parameters after 1000, 2000, 3000 and 10000 iterations

CARLA has a good performance in optimization tasks with a few number of parameters (e.g. 2 or 3), and with a few number of local optima, but the CARLA performance is severely degraded when the number of parameters or the number of local optima grows, and the CARLA becomes slow.

Figure 3 shows sample performance of CARLA for training an HMM with 57 parameters.

5.3. A Comparison between CARLA and SA

Our goal in this section is comparing CARLA with SA method for training HMMs. In all the experiments, number of iterations is 10,000. For generalization and

consistency of the results, each method must be performed many times and experimental conditions must be the same for two methods. For this purpose, we performed each experiment 50 times. That is, 50 different and random initial points and 50 random different training set were generated. Of course, initial conditions and training sets were exactly the same for two methods. Also, we used the same 'seed' for starting random number generation routines for two methods. The average performance of each method was considered as the Mean of Log-Probability in different executions of that method. Table 1 shows average performances for two methods.

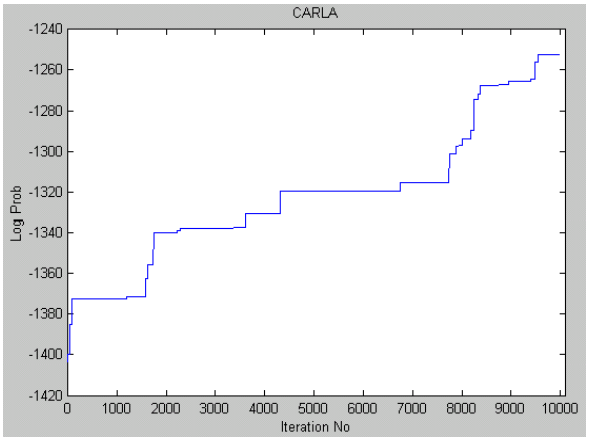

Figure 3. Sample performance of CARLA method for training an HMM with 57 parameters

Table 1. Average Performance of CARLA and SA for 50 different experiments

CARLA	BA
-1250.4	-1259.1

As the table shows, CARLA outperforms SA, and this can be due to the fact that, SA hasn't any memory and has a very slow-decreasing temperature schedule, and the final resolution of SA is much lower than the CARLA resolution in a moderate number of iterations.

6. Conclusions

In this paper, we used two global optimization and search techniques for training hidden Markov models. Among the global optimization methods, SA method and Continuous Action Reinforcement Learning Automaton (CARLA) method were evaluated. Experimental results show that the CARLA has a higher performance compared to SA. This is due to the fact that CARLA is a continuous global optimization method with memory and SA is a memoryless one.

References

[1] Rabiner, L.R., Juang, B.-H., Fundamentals of Speech Recognition, Prentice-Hall, 1993.

[2] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., "Optimization by Simulated Annealing", Science, Vol. 220, No. 4598, pp. 671-680, May 1983.

[3] Bäck, T., Evolutionary Algorithms in Theory and Practice, Oxford University Press, 1996.

[4] Stuckman, B.E., Easom, E.E., "A Comparison of Bayesian/Sampling Global Optimization Techniques", IEEE Trans. on SMC, Vol. 22, No. 5, Sep. 1992.

[5] Howell, M.N., Gordon, T.J., Best, M.C., "The Application of Continuous Action Reinforcement Learning Automata to Adaptive PID Tuning", Proceedings of IEE Control Seminar on Learning Systems for Control, pp. 1-10, Birmingham, UK, May 2000.

[6] Howell, M.N., Gordon, T.J., "Continuous Learning Automata and Adaptive Digital Filter Design", UKACC International Conference on Control, Sep. 1998.

[7] Narendra, K.S., Thathatchar, M.A.L., Learning Automata: An Introduction, Prentice-Hall, 1989.