(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 December 2015 (23.12.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/193531 Al

(51) International Patent Classification:
G06N3/04 (2006.01) *G06N 3/08* (2006.01)

(21) International Application Number:
PCT/FI2014/050478

(22) International Filing Date:
16 June 2014 (16.06.2014)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **NOKIA TECHNOLOGIES OY** [FI/FD; Karaportti 3, FI-02610 Espoo (FI).

(72) Inventors: **WABNIG, Joachim;** 62 Halifax Road, Upper Cambourne CB23 6AX (GB). **NISKANEN, Antti;** 3 Winchmore Drive, Cambridge CB2 9LW (GB).

(74) Agents: **NOKIA TECHNOLOGIES OY** et al; Virpi Tognetty, IPR Department, Karakaari 7, FI-02610 Espoo (FI).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available):* AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available):* ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

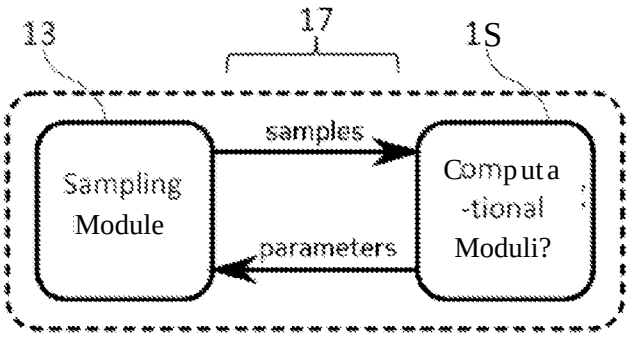
— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report (Art. 21(3))*

WO 2015/193531 A1

(54) Title: DATA PROCESSING



hybrid machine

Fig. 3

(57) Abstract: A data processing system is disclosed for machine learning. The system comprises a sampling module (13) and a computational module (15) interconnected by a data communications link (17). The computational module is configured to store a parameter vector representing an energy function of a network having a plurality of visible units connected using links to a plurality of hidden units, each link being a relationship between two units. The sampling module is configured to receive the parameter vector from the first processing module and to sample from the probability distribution defined by the parameter vector to produce state vectors for the network. The computational module is further configured to receive the state vectors from the second processing module and to apply an algorithm to produce new data. The sampling and computational modules are configured to operate independently from one another.

Data Processing

Field of the Invention

This invention relates to data processing, particularly in the field of machine learning.

Background of the Invention

Machine learning methods and systems are used to generate a set of data-dependent results based on a training or learning stage. A neural network is a type of machine learning system, a particular type of which is called a Boltzmann machine, which is a stochastic model comprising visible and hidden units that can take the values 'O' or 'l ' with given probability. The visible units are used to represent the data, and the hidden units are used for modeling. Each possible state of the machine may be assigned an energy value E and a probability distribution for all states can be fully defined by an energy function. By adjusting parameters in the energy function, it is possible to manipulate the probability distribution over all of the units.

In the learning stage, the parameters are adjusted so that the marginal probability distribution over the visible units is close to that of an example, or reference, probability distribution. After learning, the machine can be used to make predictions based on the learned data.

In practical terms, Boltzmann machines can be used for any type of operation where a predicted set of data is required based on a modeled system, for example in image or audio analysis. The process is typically slow, however, due to computational requirements.

Summary of the Invention

A first aspect of the invention provides a method comprising: storing in a first processing module a parameter vector representing an energy function of a network having a plurality of visible units connected using links to a plurality of hidden units, each link being a relationship between two units; in a separate, second processing module, receiving the parameter vector from the first processing module and sampling from the probability distribution defined by the parameter vector to produce state vectors for the network; and in the first processing module, receiving the state vectors from the second processing module and applying an algorithm to produce new data, the method comprising performing the sampling independently from the producing the new data.

The method may comprise sampling and producing the new data at least partially in parallel.

The method may comprise sampling and producing the new data asynchronously to each other.

The method may comprise sampling and/or producing the new data until a predetermined condition is reached, at which time data is exchanged from one processing module to the other processing module.

Applying the algorithm may comprise applying a learning algorithm to produce an updated parameter vector that is subsequently sent to the second processing module for re-sampling when the predetermined condition is reached.

The method may comprise sending the updated parameter vector to the second processing module for re-sampling when a predetermined plural number of iterations of the learning algorithm have been performed.

The learning algorithm may be a gradient-based learning algorithm that iterates over plural learning steps to produce an estimated optimized result with respect to reference data.

The learning algorithm may be the Kullback-Leibler (KL) -divergence algorithm.

Sampling may comprise sampling until a predetermined number of samples have been obtained, at which time sending the state vectors represented by the samples are sent to the first processing module.

The method may comprise issuing a control signal from one of the processing modules to the other when the predetermined condition is reached to enable the exchange of data.

The algorithm may take as input the state vectors and the parameter vector to generate output data representing a probability distribution.

The method may comprise buffering the samples received prior to applying the algorithm.

The method may comprise processing the buffered samples to remove duplicate state vectors prior to applying the algorithm.

The method may comprise sorting the processed state vectors into a predetermined order prior to applying the algorithm.

The method may comprise using at least two samplers to generate the state vectors and, prior to applying the algorithm, receiving the sampled output from each and removing duplicates.

Each sampler may use a different sampling method.

The first and second processing modules may be implemented on different hardware modules having their own microprocessor or microcontroller.

Each processing module may be implemented on a respective ASIC or FPGA.

One of said processing modules may be implemented on an ASIC or FPGA and the other on a multi-purpose computer system having its own microprocessor or microcontroller.

The second processing module may be implemented as a quantum annealing machine.

The first and second processing modules may exchange data over a non-dedicated data communications link, e.g. the Internet.

The first and second processing modules may be physically remote from one another.

The method may comprise receiving from a wireless terminal input data for use in the algorithm to generate the new data.

A second aspect of the invention provides a system comprising: a first processing module configured to store a parameter vector representing an energy function of a network having a plurality of visible units connected using links to a plurality of hidden units, each link being a relationship between two units; and a second processing module configured to receive the parameter vector from the first processing module and to sample from the probability distribution defined by the parameter vector to produce state vectors for the network; the first processing module being further configured to receive the state vectors

from the second processing module and to apply an algorithm to produce new data; wherein the processing modules are configured to sample and produce the new data independently from one another.

The processing modules may be configured to sample and produce the new data at least partially in parallel.

The processing modules may be configured to sample and produce the new data asynchronously to each other.

The processing modules may be configured to sample and/or produce the new data until a predetermined condition is reached, at which time data is exchanged between the two processing modules.

The first processing module may be configured to apply a learning algorithm to produce an updated parameter vector and subsequently to send it to the second processing module for re-sampling when the predetermined condition is reached.

The first processing module may be configured to send the updated parameter vector to the second processing module for re-sampling when a predetermined plural number of iterations of the learning algorithm have been performed.

The first processing module may be configured to perform a gradient-based learning algorithm that iterates over plural learning steps to produce an estimated optimized result with respect to reference data.

The first processing module may be configured to perform the KL-divergence algorithm.

The second processing module may be configured to sample until a predetermined number of samples have been obtained, at which time it sends the state vectors represented by the samples are sent to the first processing module.

One of the first or second processing modules may be configured to issue a control signal to the other when the predetermined condition is reached to enable the exchange of data.

The first processing module may be configured to take as input the state vectors and the parameter vector, and to generate output data representing a probability distribution.

The first processing module may be configured to buffer the samples received prior to applying the algorithm.

The first processing module may be configured to buffer the samples to remove duplicate state vectors prior to applying the algorithm.

The first processing module may be configured to sort the processed state vectors into a predetermined order prior to applying the algorithm.

The second processing module may comprise at least two samplers configured to generate the state vectors and, prior to applying the algorithm, to receive the sampled output from each and to remove duplicates.

Each sampler may be configured to use a different sampling method.

The first and second processing modules may be implemented on different hardware modules having their own microprocessor or microcontroller.

Each processing module may be implemented on a respective ASIC or FPGA.

One of said processing modules may be implemented on an ASIC or FPGA and the other on a multi-purpose computer system having its own microprocessor or microcontroller.

The second processing module may be implemented as a quantum annealing machine.

The first and second processing modules may be configured to exchange data over a non-dedicated data communications link, e.g. the Internet.

The first and second processing modules may be physically remote from one another.

The system may be configured to receive from a wireless terminal input data for use in the algorithm to generate the new data.

A third aspect of the invention provides a computer program comprising instructions that when executed by a computer apparatus control it to perform the method of: storing in a first processing module a parameter vector representing an energy function of a network

having a plurality of visible units connected using links to a plurality of hidden units, each link being a relationship between two units; in a separate, second processing module, receiving the parameter vector from the first processing module and sampling from the probability distribution defined by the parameter vector to produce state vectors for the network; in the first processing module, receiving the state vectors from the second processing module and applying an algorithm to produce new data; and performing the sampling independently from the producing the new data.

A fourth aspect of the invention provides a non-transitory computer-readable storage medium having stored thereon computer-readable code, which, when executed by computing apparatus, causes the computing apparatus to perform a method comprising: storing in a first processing module a parameter vector representing an energy function of a network having a plurality of visible units connected using links to a plurality of hidden units, each link being a relationship between two units; in a separate, second processing module, receiving the parameter vector from the first processing module and sampling from the probability distribution defined by the parameter vector to produce state vectors for the network; in the first processing module, receiving the state vectors from the second processing module and applying an algorithm to produce new data; and performing the sampling independently from the producing the new data.

Brief Description of the Drawings

The invention will now be described, by way of non-limiting example, with example to preferred embodiments, in which:

Figure 1 is a graphical representation of a Boltzmann machine neural network;
Figure 2 is a graphical representation of a probability distribution for four visible units of the Figure 1 neural network;
Figure 3 is a high-level block diagram of a hybrid implementation of the Figure 1 neural network according to embodiments of the invention;
Figure 4 is a high-level block diagram of a hybrid implementation of the Figure 1 neural network according to further embodiments of the invention;
Figure 5 is a block diagram showing functional modules of the Figure 3 implementation in a learning mode of operation;
Figure 6 is a block diagram showing functional modules of the Figure 3 implementation in a prediction mode of operation;
Figure 7 is a schematic timing diagram showing how in the learning mode the sampling and computational modules of the hybrid system can operate simultaneously; and

Figure 8 is a flow diagram indicating an example of respective processing steps performed by the sampling and computational modules in learning mode of operation.

Detailed Description of Some Embodiments

Embodiments herein describe systems and methods implementing a Boltzmann machine neural network for machine learning.

The system employs a hybrid architecture consisting of separate sampling and computational modules, which are able to operate independently, to improve performance. The system employs data structures and computational algorithms for implementing the operation of a Boltzmann machine. An overview of a Boltzmann machine will first be introduced.

Figure 1 is a graphical representation of a Boltzmann machine 1, which comprises visible units 3 (or nodes) and hidden units 5. The visible units 3 represent the data and the hidden units 5 are used for modeling. The units 3, 5 can take the value '0' or '1' with a given probability. Each possible state of the machine 1 may be assigned an energy value E and a probability distribution for all states can be fully defined by an energy function. By adjusting parameters in the energy function, it is possible to manipulate the probability distribution over all of the units 3, 5.

The probability of a state is given by

$$P = 1/Z \, \mathrm{Exp}[-E]$$

where Z is a normalization constant called the partition function. Usually, the energy function is quadratic in the state s = (0, 1, 1, 0, 0, 1, 0, 1...) of length M, such that

$$E = \sum_{i=1}^{M} \sum_{j=1}^{i-1} A_{ij} s_i s_j + \sum_{j=1}^{M} B_j s_j$$

where A is a matrix of dimensions M x M and B is a vector of length M.

In Figure 1, non-zero elements of the matrix A define a particular connectivity of the machine, i.e. only units i and j with a non-zero A_{ij} are connected, which are represented in Figure 1 by thick coupling lines 7.

Usually, one is interested in modeling a given probability distribution (referred to herein as an example, or reference distribution) over the visible units 3. Figure 2 shows an example binary probability distribution 11 over the four visible units 3, with each bar representing the probability that an associated bit pattern is realized. Boltzmann machines are trained in a so-called learning phase to represent such example binary probability distributions so that, subsequently, in a prediction phase, predicted data is output from the model based on inputs to a subset of the visible units 3. Practical applications of the system include image and audio processing, for example.

In the learning phase, the system parameters in the energy function are adjusted in such a way that a so-called marginal probability distribution p over the visible units 3 is close to the example probability distribution p° by a certain measure, usually their relative entropy, also called the Kullback-Leibler (KL) divergence, given by

$$KL = \sum_{i=0}^{2^{N_v}-1} p^{(0)}{}_i \log p^{(0)}{}_i - p^{(0)}{}_i \log p_i$$

where the number of visible units 3 is given by N_v with the index i running over all possible states of the visible units. The learning phase is usually performed using gradient-based methods (for example, using the so-called gradient descent or conjugate gradient method) where the gradient of the relative entropy is calculated from samples obtained from the machine 1. After learning is completed, the machine 1 is used to make predictions based on the learned data. New data can be presented by fixing the value of part of the visible units 3 ("clamping") and the predictions can be read off using the probability distribution of the remaining visible units.

Figure 3 shows a first embodiment of a system implementing a Boltzmann machine, comprising separate sampling and computational modules 13, 15 interconnected by a data communications channel 17.

In overview, the sampling module 13 takes as input from the computational module 15 a set of data representing a parameter vector, which is a vector representing couplings between the visible and hidden units 3, 5 and biases for the individual units, and so defines the energy function of the Boltzmann machine 1. The sampling module 13 samples according to the energy function defined by the parameter vector to generate candidate state vectors of the machine 1, which are made available to the computational module 15. The computational module 15 receives and stores the candidate state vectors in memory,

buffers them, and uses them with data processing algorithms to perform the learning and, subsequently, the prediction calculations.

The sampling and computational modules 13, 15 are provided as separate, independent processing systems enabling them to be implemented using hardware and/or software appropriate to the respective module's processing requirements, and also to exploit asynchronous and parallel processing to improve performance. Significantly, a number of optimization steps can be performed in the computational module 15 using gradient learning for a currently-stored set of samples before new samples are used for subsequent optimization. The samples can be buffered for this purpose.

Figure 4 shows that multiple sampling modules i3a-d can be used in parallel. The different sampling modules i3a-d can be made up of different implementations of sampler.

The sampling module 13 can be implemented on conventional hardware, such as a high performance personal computer, using a software-sampling algorithm implementing standard Metropolis sampling, parallel tempering or simulated quantum annealing, amongst others. Alternatively, the sampling module 13 can implement said selected algorithm on a Field Programmable Gate Array (FPGA), an Application Specific Integrated Circuit (ASIC) or using a physical implementation of a Boltzmann machine using bistable units, e.g. Schmitt triggers. Alternatively still, the use of a quantum annealing machine (i.e. hardware instead of software) can realize sampling using quantum mechanical effects.

The computational module 15 can be implemented as a software algorithm on a computer Central Processing Unit (CPU) or a Graphics Processor Unit (GPU), or an algorithm programmed into a FPGA or ASIC.

The sampling and computational modules 13, 15 are interconnected using a suitable data communications channel 17, which can be a broadband Internet connection or a dedicated link. The two modules 13, 15 need not be at the same physical location and can be remote from one another.

Figures 5 and 6 show functional modules of the sampling and computational modules 13, 15 for implementing the above-described learning and prediction phases. Each module 13, 15 can be implemented using hardware modules, software modules or a combination of both types.

The sampling module 13 provides first and second samplers 21, 23 in this embodiment. Each sampler 21, 23 is configured to receive as input a set of data representing the parameter vector from the computational module 15, to sample from the probability distribution defined by the parameter vector data and to generate candidate state vectors in the form of a sample string of bits (0 or 1), which are made available to the computational module over the communications link 17. The parameter vector is stored in a parameter memory 37. Hardware samplers have a given connectivity and limited precision on the couplings, whereas software samplers can have arbitrary connectivity and high precision on the couplings. Either or both types can be used. The samplers can run in batches. A software sampler can run several samplings in parallel. Software and hardware samplers can produce the samples at different respective rates, and hence the most appropriate form of sampling module or sampling module combination can be used.

Whereas the sampling module 13 operates in the same way during learning and prediction phases, the computational module 15 employs different algorithms in the learning and prediction phases.

Learning Phase

In the learning phase, the aim of the hybrid system and method is to generate a set of parameters so that the marginal probability distribution over the visible units of the Boltzmann machine has a minimum KL-divergence with an example probability distribution, which is represented in data form and is programmed into an example memory 35 of the computational module 15. The initial value of the parameter vector can be set in a number of ways. For example, the parameter vector could initially be a zero vector, representing the machine as having all units uncoupled and unbiased. Alternatively, the initial value could be a random vector with a certain maximum entry. More sophisticated methods can, for example, grow the machine by adding visible and hidden units in the optimization process, and taking a good initial vector from the previous iteration in that process.

Processing works iteratively between an inner processing loop 19, and an outer processing loop 20.

The outer loop 20 proposes candidate state vectors as data, buffers and stores them in a state memory 31; the inner loop 19 takes these states and iteratively tries to adjust the parameter vector to maximize overlap with the example probability distribution (or

minimize divergence). This is a form of optimization and, in this embodiment, the known KL-divergence (Kullback-Leibler -divergence) algorithm is employed and implemented in the KL module 45 shown in Figure 5.

The KL module 45 takes as input the example probability distribution, an ordering of the samples from an order memory 43 and energies for the samples from an energy memory 41. The current parameter vector stored in the parameter memory 37 is also taken as input. The output of the KL module 45 is an updated parameter vector, which is stored in the parameter memory 37 and is iteratively updated during the learning process as the divergence is minimized.

In a Boltzmann machine, the derivative (dKL) 46 of the KL-divergence (KL) 49 with respect to parameters can be calculated directly from the states. A line search in the KL-divergence (KL) 49 establishes an ideal step size along the gradient. Alternatively, a conjugate gradient method can be used.

It is not necessary to obtain new samples for every gradient step, because one would expect the partition function to change slowly if the change in parameters is small. A convenient routine is therefore to take a certain number of gradient steps, e.g. 1000, and then add new states (to the state memory, to be introduced below). Further information on the KL divergence method will be known to the skilled person.

With regard to the outer processing loop 20, the computational module 15 comprises first and second sample memories 25, 27 for receiving and storing the candidate state vectors from the respective first and second samplers 21, 23. The sample memories 25, 27 act as a buffer in view of the fact that the samplers, depending on whether they are implemented in hardware or software, may issue the samples at different rates and with different precision.

An accumulator 29 is provided which takes the list of samples from the two sample memories 25. In the method described, we only need a list of all possible states and not their frequency of occurrence. The accumulator 29 is configured to eliminate all multiples received from the sample memories 25.

The accumulator 29 is connected to a state memory 31, which stores a list of all states that have occurred in the sampling process. The aim is to store a list of eligible states that give a good approximation to the partition function. Sample states already stored in the state

memory 31 are made available to the accumulator 29 so that it will only send new sample states. The accumulator 29 may use a more sophisticated algorithm to decide which states to keep. In some embodiments, for example, only the N lowest energy states may be kept. In some embodiments, states from previous samplings may be mixed with states from the current sampling. In some embodiments, the N lowest states of a mixture of states from current and previous samplings may be kept.

A sorting module 33 is provided, and is configured to take the example probability distribution over the visible units from the example memory 35, and to generate a list of the states in the state memory 31 that correspond to a particular example state. This list is stored in the order memory 43, which is used by the KL module 45. The order memory 43 is configured to store the location of the states in the state memory 31 that have the same bit string pattern on their visible bits, which enables the KL module 45 more easily to calculate the KL 49 and dKL 47.

An energy module 39 is provided, and is configured to calculate the energy of the states in the state memory 31. The energy module 39 can calculate the energy based on the updated parameter vector stored in the parameter memory 37 and is therefore part of the KL optimization (inner loop 19) processing. It will be appreciated therefore that no new sampling is necessary in the iterative loop; only new energies have to be calculated for a fixed number of states.

In overview, therefore, the computational module 15 operates in the learning phase to iteratively compute an optimized parameter vector using a set of samples buffered in the state memory 31. The sampling module 13 can resample using the current parameter vector whilst the optimization (inner loop 19) processing is ongoing, in asynchronous fashion, and generate new samples which are buffered in the sample memory 25, 27 for updating the state memory 31. Sampling and optimization can be performed in parallel.

Prediction Phase

The aim of the prediction phase is to correlate an input pattern applied to a subset of the visible units 3 with the most likely output pattern. The parameter vector optimized in the learning stage is used for this purpose. In practice, a subset of the visible units 3 is selected to be the input units with the remaining units being the output units. The input units are fixed to a given pattern of os and is. The system when run in the prediction phase generates a probability distribution over the output units. The input pattern could

be a representation of an image, for example, with the output pattern being a label, or vice versa.

Referring to Figure 6, a clamping module 51 is provided which receives as input a fixed input pattern and part of the optimized parameter vector obtained in the learning stage. The clamping module 51 is configured to calculate the biases on the hidden units 5 resulting from fixing the input pattern combined with the parameter vector that has been determined in the learning phase to work for the classification problem at hand. The result is a clamped parameter vector which is input to the parameter memory 37, and provides the input to the sampling module 13. In the prediction phase, the samplers 21, 23 in the sampling unit operate as before, taking the clamped parameter vector as input from the computational module 15. In the computational module 15, the sample memories 25, accumulator 29, state memory 31, sort module 33 and order memory 43 operate as before also. The sort module 33 however takes the output bits as one set of inputs, instead of the example probability distribution.

A marginalization module 53 is provided which receives as input the ordering of samples from the order memory 43. It calculates the marginal probability distribution over the output bits using the state energies and ordering of samples, therefore.

The energy module 39 in this phase is configured to calculate the energy for each entry in the state memory 31 and writes it to the energy memory, which enables more precise probabilities to be calculated in the marginalization module 53 than can be obtained from monitoring state frequencies from sampling alone.

Referring to Figure 7, it will be seen that the sampling module and computational module can be operated at the same time, in parallel, and in asynchronous fashion. Synchronization between the modules is only required at the point of data exchange, which occupies a small fraction of the overall processing time. As indicated, the upper portion 61 of the diagram indicates processing performed by the computational module 15, and the lower portion 63 by the sampling module 13. Two sampling periods, t_1 and t_2, are indicated to illustrate the process. The indicated programming time relates to the physical implementation of the sampler, for example translating the parameter vector into voltages or currents. During the time when the computational module 15 is processing the samples stored in the state memory and taking the gradient steps in the KL optimization, the sampling module 13 is able to receive a new parameter vector from the parameter memory 37 and commence sampling in preparation for a later data exchange. Data

exchange over the data communications channel 17 can take place either when a certain number of optimization steps have been taken to generate the updated parameter vector (control by the computational module 15) or when a certain number of samples have been obtained (control by the sampling module 13). Data exchange can take place at any time there are new samples available.

An indication of the length of the sampling period f can be obtained by observing the partition function as a function of the length of the sampling period. If the partition function shows saturation, one can assume that the samples obtained provide a good approximation to the partition function.

Figure 8 is a flow diagram indicating respective processing steps performed by each of the sampling and computational modules 13, 15 in a learning stage example. In a first step 8.1 the sampling module 13 receives the parameter vector currently in the parameter memory 37 of the computational module 15. In a second step 8.2, samples are taken and, in a third step 8.3, the samples are transmitted over the data channel 17 to the computational module 15.

In the computational module, in step 8.4 the received candidate samples are stored. In step 8.5, a gradient-based optimization step is performed on the sampled data (e.g. using KL-convergence as suggested), and the parameter vector resulting from one iteration is updated in step 8.6. In step 8.7, it is determined whether a predefined optimization condition is achieved. If so, the learning process ends in step 8.8. If not, the iteration count is incremented in step 8.9. Step 8.10 determines if a predetermined iteration count n=m is reached. If not, a further optimization step proceeds by returning to step 8.5. If the count is reached, the current parameter vector is sent to the sampling module in step 8.11.

The proximity of the sampling and computational units 13, 15 can be different for the learning and prediction phases. In the learning phase, a high bandwidth connection between the two units 13, 15 is preferable, e.g. performed in the same datacenter. In the prediction phase, both low and high bandwidth connections can be used. A mobile device, for example a smart phone, may be used in the prediction phase to provide the input data and receive the output. The input and output data will be in the form of relatively short bit-strings and therefore appropriate for wireless communications.

In summary, the embodiment provides systems and methods for a hybrid approach to implementing a Boltzmann machine neural network 1, employing distinct sampling and computational modules 13, 15, each employing hardware and/or software implementations appropriate for improving their own speed, accuracy and ability to reject erroneous states. The computational module 15 is able to perform multiple optimization steps on stored samples without the need to be reprogrammed with new samples for each step, whilst the sampling module 13 is configured to start re-sampling after a predetermined period in parallel with the optimization. The computational module 15, for example, may recalculate probabilities with machine precision (32 or 64 bit) without the sampling module 13 necessarily having to be set-up for this level of precision. Data can be exchanged asynchronously, enabling each distinct module 13, 15 to operate independently according to their hardware and/or software capabilities.

It will be appreciated that the above described embodiments are purely illustrative and are not limiting on the scope of the invention. Other variations and modifications will be apparent to persons skilled in the art upon reading the present application.

Moreover, the disclosure of the present application should be understood to include any novel features or any novel combination of features either explicitly or implicitly disclosed herein or any generalization thereof and during the prosecution of the present application or of any application derived therefrom, new claims may be formulated to cover any such features and/or combination of such features.

Claims

1. A method comprising:

storing in a first processing module a parameter vector representing an energy function of a network having a plurality of visible units connected using links to a plurality of hidden units, each link being a relationship between two units;

in a separate, second processing module, receiving the parameter vector from the first processing module and sampling from the probability distribution defined by the parameter vector to produce state vectors for the network; and

in the first processing module, receiving the state vectors from the second processing module and applying an algorithm to produce new data,

the method comprising performing the sampling independently from the producing the new data.

2. A method according to claim 1, comprising sampling and producing the new data at least partially in parallel.

3. A method according to claim 1 or claim 2, comprising sampling and producing the new data asynchronously to each other.

4. A method according to any preceding claim, comprising sampling and/ or producing the new data until a predetermined condition is reached, at which time data is exchanged from one processing module to the other processing module.

5. A method according to claim 4, wherein applying the algorithm comprises applying a learning algorithm to produce an updated parameter vector that is subsequently sent to the second processing module for re-sampling when the predetermined condition is reached.

6. A method according to claim 5, comprising sending the updated parameter vector to the second processing module for re-sampling when a predetermined plural number of iterations of the learning algorithm have been performed.

7. A method according to claim 6, wherein the learning algorithm is a gradient-based learning algorithm that iterates over plural learning steps to produce an estimated optimized result with respect to reference data.

8. A method according to claim 7, wherein the learning algorithm is the KL-divergence algorithm.

9. A method according to any one of claims 4 to 8, wherein sampling comprises sampling until a predetermined number of samples have been obtained, at which time sending the state vectors represented by the samples are sent to the first processing module.

10. A method according to any one of claims 4 to 9, comprising issuing a control signal from one of the processing modules to the other when the predetermined condition is reached to enable the exchange of data.

11. A method according to any preceding claim, wherein the algorithm takes as input the state vectors and the parameter vector to generate output data representing a probability distribution.

12. A method according to any preceding claim, comprising buffering the samples received prior to applying the algorithm.

13. A method according to claim 12, comprising processing the buffered samples to remove duplicate state vectors prior to applying the algorithm.

14. A method according to claim 13, comprising sorting the processed state vectors into a predetermined order prior to applying the algorithm.

15. A method according to any preceding claim, comprising using at least two samplers to generate the state vectors and, prior to applying the algorithm, receiving the sampled output from each and removing duplicates.

16. A method according to claim 15, wherein each sampler uses a different sampling method.

17. A method according to any preceding claim, wherein the first and second processing modules are implemented on different hardware modules having their own microprocessor or microcontroller.

18. A method according to claim 17, wherein each processing module is implemented on a respective ASIC or FPGA.

19. A method according to claim 17, wherein one of said processing modules is implemented on an ASIC or FPGA and the other on a multi-purpose computer system having its own microprocessor or microcontroller.

20. A method according to any preceding claim, wherein the second processing module is implemented as a quantum annealing machine.

21. A method according to any preceding claim, wherein the first and second processing modules exchange data over a non-dedicated data communications link, e.g. the Internet.

22. A method according to any preceding claim, wherein the first and second processing modules are physically remote from one another.

23. A method according to any preceding claim, comprising receiving from a wireless terminal input data for use in the algorithm to generate the new data.

24. A system comprising:

a first processing module configured to store a parameter vector representing an energy function of a network having a plurality of visible units connected using links to a plurality of hidden units, each link being a relationship between two units; and

a second processing module configured to receive the parameter vector from the first processing module and to sample from the probability distribution defined by the parameter vector to produce state vectors for the network;

the first processing module being further configured to receive the state vectors from the second processing module and to apply an algorithm to produce new data;

wherein the processing modules are configured to sample and produce the new data independently from one another.

25. A system according to claim 24, wherein the processing modules are configured to sample and produce the new data at least partially in parallel.

26. A system according to claim 24 or claim 25, wherein the processing modules are configured to sample and produce the new data asynchronously to each other.

27. A system according to any of claims 24 to 26, wherein the processing modules are configured to sample and/or produce the new data until a predetermined condition is reached, at which time data is exchanged between the two processing modules.

28. A system according to claim 27, wherein the first processing module is configured to apply a learning algorithm to produce an updated parameter vector and subsequently to send it to the second processing module for re-sampling when the predetermined condition is reached.

29. A system according to claim 28, wherein the first processing module is configured to send the updated parameter vector to the second processing module for re-sampling when a predetermined plural number of iterations of the learning algorithm have been performed.

30. A system according to claim 29, wherein the first processing module is configured to perform a gradient-based learning algorithm that iterates over plural learning steps to produce an estimated optimized result with respect to reference data.

31. A system according to claim 30, wherein the first processing module is configured to perform the KL-divergence algorithm.

32. A system according to any one of claims 27 to 31, wherein the second processing module is configured to sample until a predetermined number of samples have been obtained, at which time it sends the state vectors represented by the samples are sent to the first processing module.

33. A system according to any one of claims 27 to 32, wherein one of the first or second processing modules is configured to issue a control signal to the other when the predetermined condition is reached to enable the exchange of data.

34. A system according to any of claims 24 to 33, wherein the first processing module is configured to take as input the state vectors and the parameter vector, and to generate output data representing a probability distribution.

35. A system according to any of claims 24 to 34, wherein the first processing module is configured to buffer the samples received prior to applying the algorithm.

36. A system according to claim 35, wherein the first processing module is configured to buffer the samples to remove duplicate state vectors prior to applying the algorithm.

37. A system according to claim 36, wherein the first processing module is configured to sort the processed state vectors into a predetermined order prior to applying the algorithm.

38. A system according to any of claims 24 to 37, wherein the second processing module comprises at least two samplers configured to generate the state vectors and, prior to applying the algorithm, to receive the sampled output from each and to remove duplicates.

39. A system according to claim 38, wherein each sampler is configured to use a different sampling method.

40. A system according to any of claims 24 to 38, wherein the first and second processing modules are implemented on different hardware modules having their own microprocessor or microcontroller.

41. A system according to claim 40, wherein each processing module is implemented on a respective ASIC or FPGA.

42. A system according to claim 40, wherein one of said processing modules is implemented on an ASIC or FPGA and the other on a multi-purpose computer system having its own microprocessor or microcontroller.

43. A system according to any of claims 24 to 42, wherein the second processing module is implemented as a quantum annealing machine.

44. A system according to any of claims 24 to 43, wherein the first and second processing modules are configured to exchange data over a non-dedicated data communications link, e.g. the Internet.

45. A system according to any of claims 24 to 44, wherein the first and second processing modules are physically remote from one another.

46. A system according to any preceding claim, configured to receive from a wireless terminal input data for use in the algorithm to generate the new data.

47. A computer program comprising instructions that when executed by a computer apparatus control it to perform the method of any of claims 1to 23.

48. A non-transitory computer-readable storage medium having stored thereon computer-readable code, which, when executed by computing apparatus, causes the computing apparatus to perform a method comprising:

storing in a first processing module a parameter vector representing an energy function of a network having a plurality of visible units connected using links to a plurality of hidden units, each link being a relationship between two units;

in a separate, second processing module, receiving the parameter vector from the first processing module and sampling from the probability distribution defined by the parameter vector to produce state vectors for the network;

in the first processing module, receiving the state vectors from the second processing module and applying an algorithm to produce new data; and

performing the sampling independently from the producing the new data.

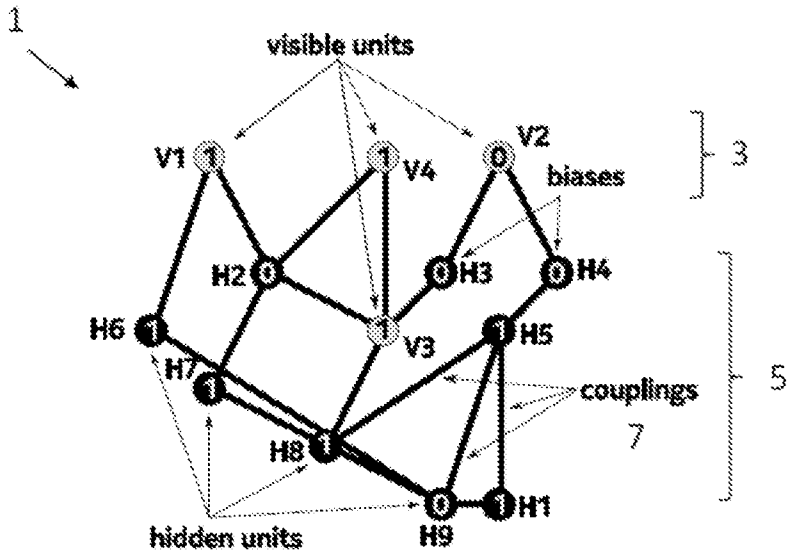

1
visible units
V1
V2
V4
biases
3
H2
H3
H4
H6
V3
H5
H7
couplings
5
7
H8
H1
H9
hidden units

Fig. 1

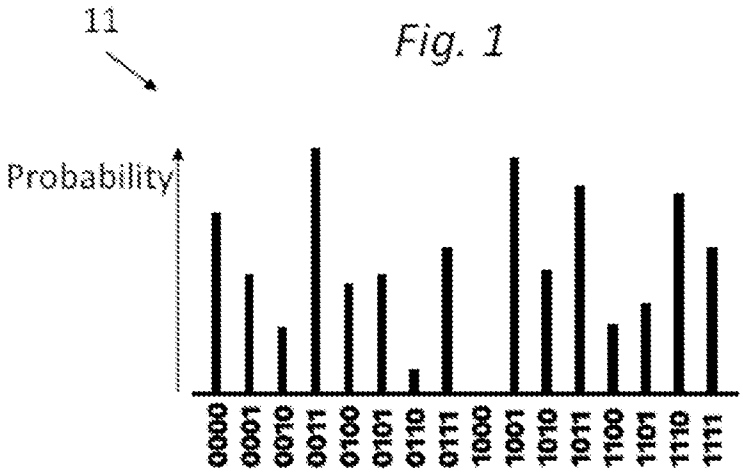

11
Probability
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

Fig. 2

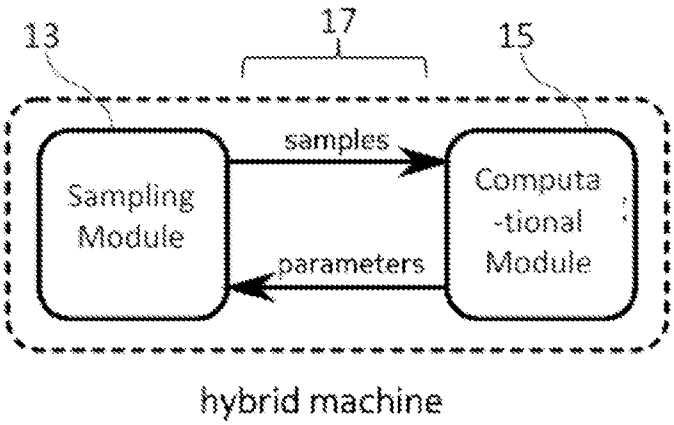

13
17
15
Sampling Module
samples
parameters
Computa -tional Module
hybrid machine

Fig. 3

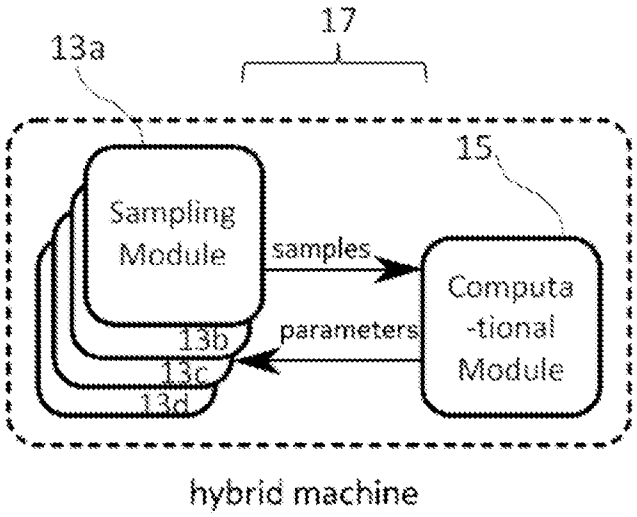

13a
17
15
Sampling Module
13b
13c
13d
samples
parameters
Computa -tional Module
hybrid machine

Fig. 4

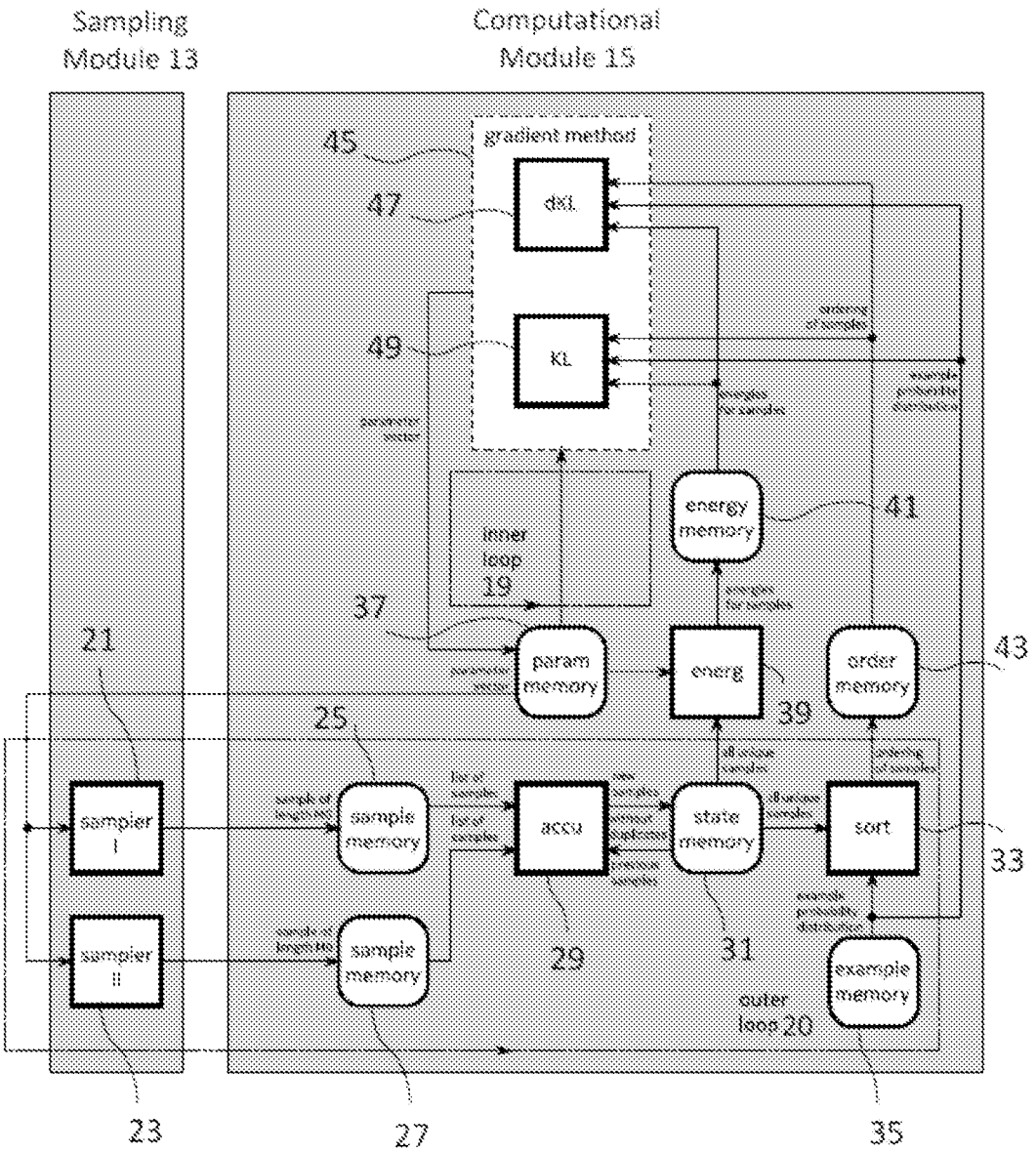

Sampling Module 13
Computational Module 15
gradient method
45
47
dKL
49
KL
energy memory
41
inner loop 19
37
param memory
energ
order memory
43
39
21
25
sampler I
sampler II
sample memory
sample memory
accu
state memory
sort
33
29
31
example memory
outer loop 20
23
27
35

Fig. 5

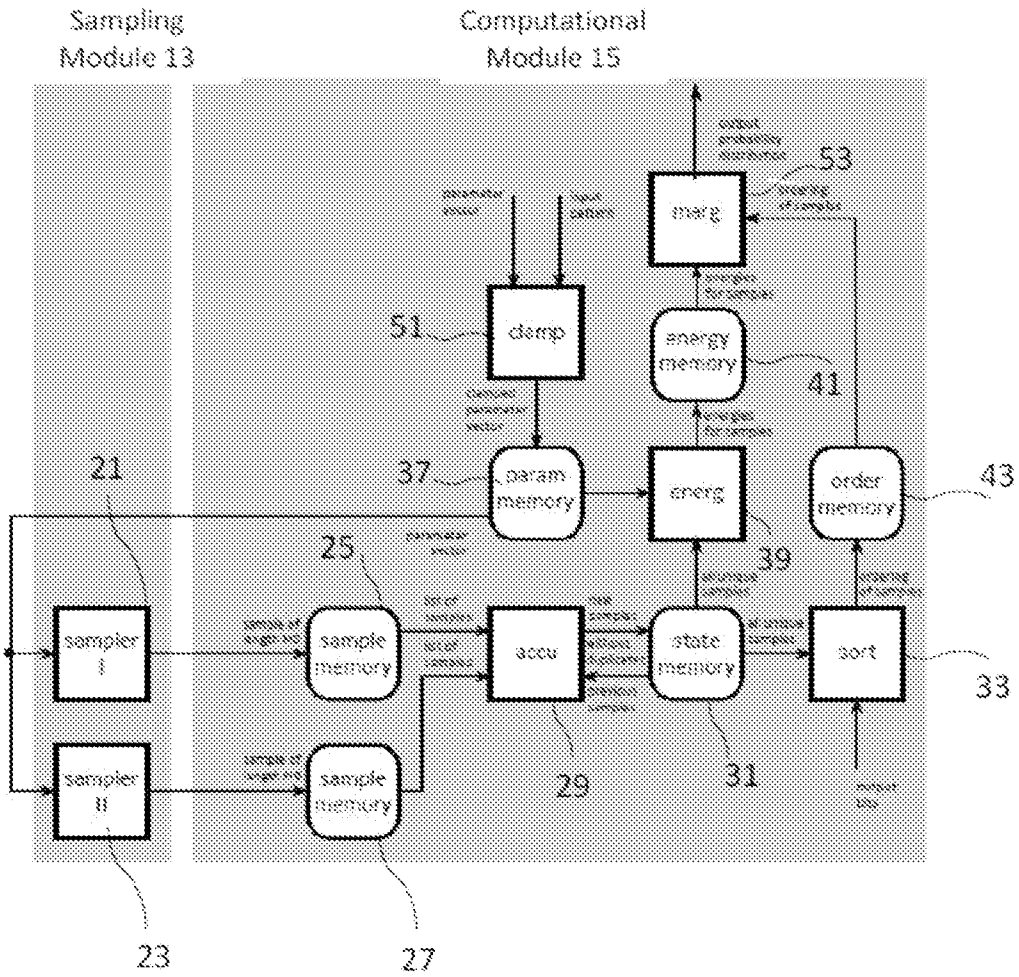

Sampling
Module 13
Computational
Module 15
53
merg
51
clemp
energy
memory
41
21
37
param
memory
energ
order
memory
43
25
39
sampler
I
sample
memory
accu
state
memory
sort
33
sampler
II
sample
memory
29
31
23
27

Fig. 6

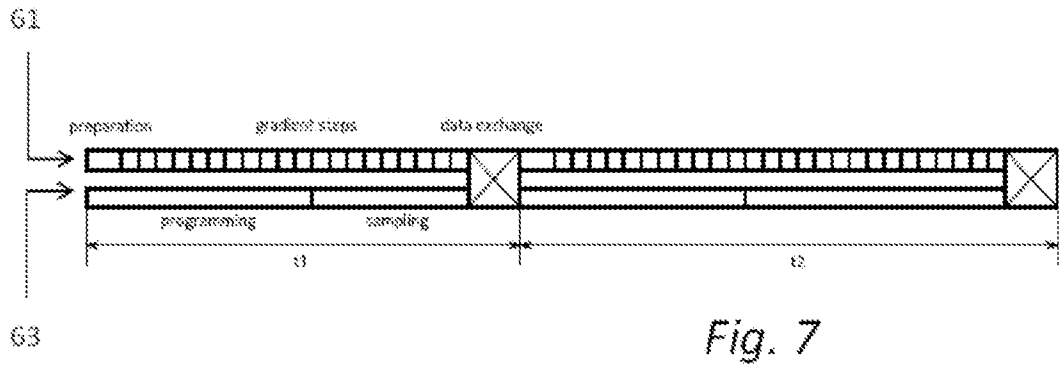

61
preparation
gradient steps
data exchange
programming
sampling
63

Fig. 7

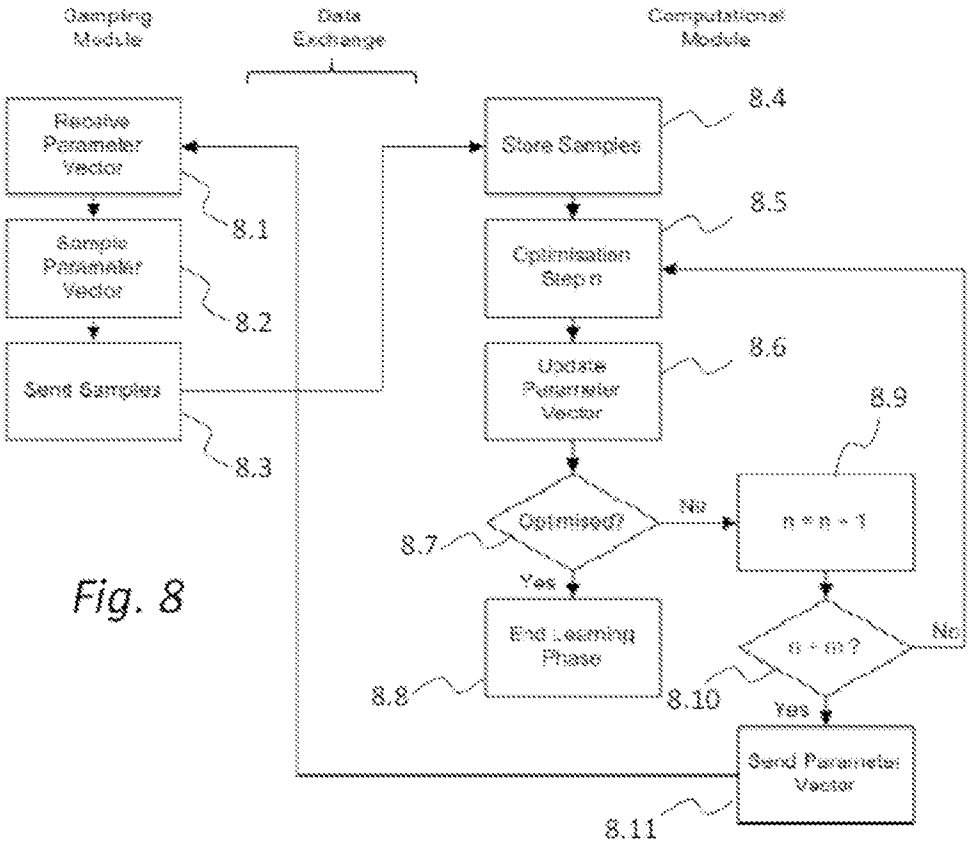

Sampling Module
Data Exchange
Computational Module
Receive Parameter Vector
Sample Parameter Vector
Send Samples
Store Samples
Optimization Step n
Update Parameter Vector
Optimised?
No
Yes
n = n + 1
End Learning Phase
n = m ?
No
Yes
Send Parameter Vector
8.1
8.2
8.3
8.4
8.5
8.6
8.7
8.8
8.9
8.10
8.11

Fig. 8

INTERNATIONAL SEARCH REPORT

International application No.
PCT/FI201 4/050478

A. **CLASSIFICATION OF SUBJECT MATTER**

See extra sheet

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC: G06N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

FI, SE, NO, DK

Electronic data base consulted during the international search (name of data base, and, where practicable, search terms used)

EPO-Internal, WPI

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5956702 A (MATSUOKA MASAHIRO [JP] et al.) 21 September 1999 (21.09.1999) column 6 line 52 - column 7 line 59, column 8 line 61 - column 9 line 40; figures 3, 5	1-48
A	US 5303328 A (MASUI HIRONARI [JP] et al.) 12 April 1994 (12.04.1994) column 4 line 61 - column 5 line 21; figure 1A	1-48

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 March 201 5 (25.03.201 5)	26 March 201 5 (26.03.201 5)

Name and mailing address of the ISA/FI
Finnish Patent and Registration Office
P.O. Box 1160, FI-00101 HELSINKI, Finland
Facsimile No. +358 9 6939 5328

Authorized officer
Kimmo Karkkainen
Telephone No. +358 9 6939 500

INTERNATIONAL SEARCH REPORT
Information on Patent Family Members

International application No.
PCT/FI201 4/050478

Patent document cited in search report	Publication date	Patent family members(s)	Publication date
US 5956702 A	2 1/09/1 999	JP H0973440 A	18/03/1 997
US 5303328 A	12/04/1 994	JP H041 60463 A	03/06/1 992

INTERNATIONAL SEARCH REPORT

International application No.
PCT/FI201 4/050478

CLASSIFICATION OF SUBJECT MATTER

IPC
G06N 3/04 (2006.01)
G06N 3/08 (2006.01)