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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 April 2021 (15.04.2021)

WIPO | PCT

(10) International Publication Number
WO 2021/069574 A1

(51) International Patent Classification:
G06N 3/00 (2006.01) *G06N 3/08* (2006.01)
G06N 3/04 (2006.01) *G06N 7/00* (2006.01)

(21) International Application Number:
PCT/EP2020/078259

(22) International Filing Date:
08 October 2020 (08.10.2020)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/912,599 08 October 2019 (08.10.2019) US

(71) Applicant: **DEEPMIND TECHNOLOGIES LIMITED** [GB/GB]; 5 New Street Square, London EC4A 3TW (GB).

(72) Inventors: **SEZENER, Eren**; 6 Pancras Square, London N1C 4AG (GB). **VENESS, Joel William**; 6 Pancras Square, London N1C 4AG (GB). **HUTTER, Marcus**; 6 Pancras Square, London N1C 4AG (GB). **WANG, Jianan**; 6 Pancras Square, London N1C 4AG (GB). **BUDDEN, David**; 6 Pancras Square, London N1C 4AG (GB).

(74) Agent: **FISH & RICHARDSON P.C.**; Highlight Business Towers, Mies-van-der-Rohe-Straße 8, 80807 München (DE).

(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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, 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, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(54) Title: GATED LINEAR CONTEXTUAL BANDITS

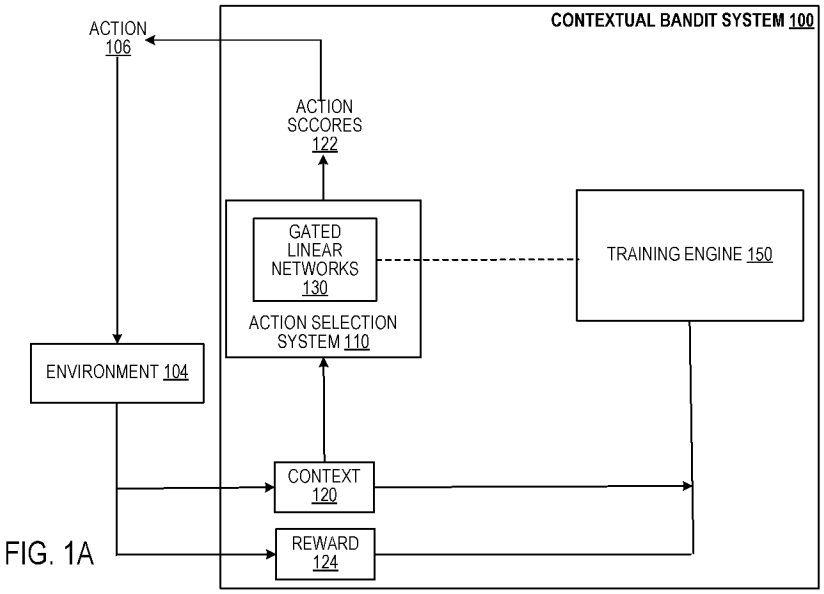

FIG. 1A

(57) Abstract: Methods, systems, and apparatus, including computer programs encoded on computer storage media, for selecting actions in response to each context in a sequence of context inputs. One of the methods includes maintaining data specifying a respective gated linear network corresponding to each of the plurality of actions; for each context in the sequence of contexts: for each action, processing the context using the gated linear network corresponding to the action to generate a predicted probability; for each action, generating an action score for the action from at least the predicted probability; and selecting the action to be performed in response to the context based on the action scores.

WO 2021/069574 A1

[Continued on next page]

(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, ST, 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).

Published:

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

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

GATED LINEAR CONTEXTUAL BANDITS

BACKGROUND

This specification relates to selecting actions in response to context inputs.

In a contextual bandits scenario, an agent iteratively selects actions to be performed from a set of possible actions. At each iteration, the agent receives a context input that is associated with the iteration and then selects the action for the iteration based on the context input.

SUMMARY

This specification describes a system implemented as computer programs on one or more computers in one or more locations that selects actions to be performed in response to received context inputs.

Particular embodiments of the subject matter described in this specification can be implemented so as to realize one or more of the following advantages.

The described systems select actions in a contextual bandits setting, i.e., in response to context inputs, using gated linear networks. Such an action selection scheme will be referred to as a gated linear contextual bandit. Using gated linear networks to select actions results in more accurate action selections, i.e., in terms of received rewards, while reducing the amount of computational resources required to generate an action selection. This can be attributed to several features of the described scheme. As one example, the described scheme allows the system to estimate prediction uncertainty with effectively zero algorithmic overhead by leveraging the data-dependent gating properties of the GLN, allowing for more accurate pseudo-counts to be computed without adding computational overhead and resulting in more effective exploration of the space of possible actions. As another example, the system can compute an action score for an action and an update to the weights of a gated linear network for the action in one single forward pass through the gated linear network, eliminating the computationally intensive backward pass that is required to update model weights for conventional systems that use conventional deep neural networks to generate action scores. Because the gated linear networks can be updated entirely online, the system

does not need to store historical data, apart from small signature data for computing pseudo-counts, greatly reducing the memory footprint of the system relative to other techniques that use neural network to select actions.

The details of one or more embodiments of the subject matter described in this specification are set forth in the accompanying drawings and the description below. Other features, aspects, and advantages of the subject matter will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1A shows an example contextual bandits system.

FIG. 1B shows an example of a gated linear network (GLN).

FIG. 2 is a flow diagram of an example process for selecting an action in response to a context input.

FIG. 3 is a flow diagram of another example process for selecting an action in response to a context input.

Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

This specification generally describes a system that repeatedly selects actions to be performed in response to received context inputs.

Each action is selected from a predetermined set of actions and the system selects actions in an attempt to maximize the rewards received in response to the selected actions.

Generally, the rewards are numeric values that measure the quality of the selected actions. In some implementations, the reward for each action is either zero or one, while in other implementations each reward is a value drawn from a continuous range e.g. between a lower bound reward value and an upper bound reward value.

In some cases, the actions are recommendations of content items, e.g., videos, advertisements, images, search results, or other pieces of content, and the context input represents a feature vector characterizing the current recommendation setting, i.e., data describing the circumstances in which the content item is going to be recommended, e.g., any

of the current time, attributes of the user device of the user to whom the recommendation will be displayed, attributes of previous content items that been recommended to the user and user responses to those previous content items, and attributes of the setting in which the content item is going to be placed. In these cases, the reward values measure the quality of the recommendation. For example, the value might be one if the user interacted with the recommendation and zero if the user did not. As another example, the reward value may be a value that measures the degree of future user engagement with content items recommended to the user after the current recommendation is made.

In some other cases, the actions are user interface elements that may be presented to a user in a user interface, i.e., in a given presentation setting, and the context input represents a feature vector characterizing the given presentation setting, the user, or both. In these cases, the reward values measure the quality of the presented user interface element. For example, the value might be one if the user interacted with the element and zero if the user did not. As another example, the reward value may be a dwell time or other metric that measures a degree of user interaction with the user interface element.

The above examples describe cases where the reward is based on user feedback after an action is selected. In some other cases, however, the reward is automatically generated, e.g., as the output of a scoring function that scores the selected action. The scoring function can be a function that, for example, scores the sensitivity or the accuracy of the selected actions. As another example, the scoring function can be a neural network that receives data characterizing an environment after the action has been selected and generates a reward for the action. Using these kinds of rewards can enable the described techniques to be used in many cases without a human user in the loop, e.g., for input classification tasks, industrial control tasks, or other tasks described below with reference to FIG. 1.

To select the action, the system maintains data specifying one or more gated linear networks corresponding to each of the plurality of actions. When the rewards are either zero or one, a single gated linear network corresponds to each of the plurality of actions and is configured to predict a probability that a reward will be received, i.e., a probability that the reward will be one instead of zero, if the corresponding action is performed in response to an input context.

The rewards may be continuous values e.g. drawn from a range. Then in one approach a tree of gated linear networks corresponds to each of the plurality of actions and the range is divided into bins. The tree of gated linear networks corresponding to a given action are collectively configured to predict a respective probability for each of the bins that represents the likelihood that the received reward will fall in the corresponding bin if the corresponding action is performed in response to an input context.

To select an action for a given context input, for each action the system processes the context input using the one or more gated linear networks corresponding to the action to generate either (i) a predicted probability that a reward of one will be received or (ii) a respective probability for each bin.

The system then selects the action to be performed using the outputs of the gated linear networks by generating an action score for the action from the outputs of the gated linear network(s) for the action and, in some cases, a pseudo-count. The pseudo-count is also generally determined from the one or more gated linear networks corresponding to the action as described below.

The system then selects the action to be performed in response to the context based on the action scores, e.g., by selecting the action with the highest action score.

FIG. 1 shows an example contextual bandits system 100. The contextual bandits system 100 is an example of a system implemented as computer programs on one or more computers in one or more locations in which the systems, components, and techniques described below are implemented.

The system 100 includes an action selection system 110 and a training engine 150.

The system 100 repeatedly selects actions 106 to be performed, e.g., by the system 100 or by another system, in an environment 104 in response to received context inputs 120. For example, as described above, the actions can be content item recommendations to be made to a user in an environment, i.e., in a setting for the content item recommendation, e.g., on a webpage or in a software application.

Performance of each selected action 106 generally causes the environment 104 to transition into new states and causes the system 100 to receive a reward 124 from the environment 104.

Generally, the reward 124 is a numerical value that represents a quality of the selected action 106. In some implementations, the reward 124 for each action 106 is either zero or one, i.e., indicates whether the action was successful or not, while in other implementations the reward 124 is a value drawn from a continuous range between a lower bound reward value and an upper bound reward value, i.e., represents the quality of the action 106 as a value from a continuous range rather than as a binary value.

In particular, the action selection system 110 selects actions in an attempt to maximize the rewards received in response to the selected actions using a set of gated linear networks (GLNs) 130 that includes one or more GLNs 130 corresponding to each of the plurality of actions that can be performed in response to the context input 120. For example the system 110 can select actions to maximize the cumulative reward, i.e., the sum of the rewards received over some horizon or a time-discounted sum of the rewards received over the horizon.

Each GLN 130 is configured to predict a respective probability given an input context 120.

When the rewards 124 are either zero or one, a single GLN 130 corresponds to each of the plurality of actions and is configured to predict a probability that a reward will be received, i.e., a probability that the reward will be one instead of zero, if the corresponding action is performed in response to an input context 120.

When the rewards 124 are continuous values drawn from a range of values between a minimum value and a maximum value, in one approach a tree of GLNs 130 corresponds to each of the plurality of actions and the range is divided into bins. The tree of GLNs corresponding to a given action are collectively configured to predict a respective probability for each of the bins that represents the likelihood that the received reward will fall in the corresponding bin if the corresponding action is performed in response to an input context 120. Generating these probabilities is described in more detail below with reference to FIG. 3. In another approach when the rewards 124 are continuous values, whether or not within a range, the GLN for each action may output parameters, e.g. a mean and optionally a variance, which define a Gaussian distribution representing the probability.

In some implementations a context may correspond to an observation of the environment 104 at a time step e.g. it may comprise a feature vector representing the

environment at the time step. The action selection system 110 may select an action to perform at the time step in response to the observation.

In some implementations the environment may be a real-world environment, or a simulation of a real-world environment. The context may comprise a set of features e.g. encoded as a feature vector, representing the observation of the environment e.g. an image or other sensor data. The action may correspond to an action to perform in the environment to perform a task.

In one example the environment may comprise a manufacturing plant, or a service facility such as a data center, server farm, or grid mains power or water distribution system, or an electrical power generation facility such as a solar or wind farm. The observations may include data from one or more sensors monitoring the functioning of electronic and/or mechanical items of equipment in the plant or facility e.g. current, voltage, power, temperature, or observations of the weather such as wind speed or solar irradiance, or other characteristics. The actions may control or impose operating conditions on the items of equipment e.g. adjust a setting of or turn an item of equipment on or off or adjust a wind turbine or solar collector alignment. The reward may depend on operation of the plant or facility e.g. water or electricity power use or generation, a measure of environmental impact, or temperature control or the items of equipment.

In another example the environment may be a packet data communications network environment. The observations may comprise routing metrics e.g. any of a metric of routing path length, bandwidth, load, hop count, path cost, delay, maximum transmission unit (MTU), and reliability, e.g. derived from observations of a routing table. The actions may comprise data packet routing actions e.g. to define one or more steps of a route for a data packet, or resource allocation actions e.g. to allocate a channel to one or more data packets. The reward (or equivalently cost) may be dependent on one or more of the routing metrics, e.g. to maximize throughput or minimize latency.

In another example the environment may be a computing environment. The observations may include observations of available computing resources such as compute and/or memory capacity, or Internet-accessible resources. The actions may comprise actions to assign compute tasks to particular computing resources. The reward may be dependent

upon one or more of e.g. utilization of computing resources, electrical power, bandwidth, and computation speed.

In another example the environment may be a circuit routing environment e.g. for routing interconnection lines of an integrated circuit such as an ASIC. The observations may comprise observations of component positions and interconnections; the actions may comprise component placing actions e.g. to define a component position or orientation and/or interconnect routing actions e.g. interconnect selection and/or placement actions. The reward may depend on one or more collective routing metrics for the circuit e.g. derived from interconnect length, electrical resistance, capacitance, impedance, loss, propagation delay, or electromagnetic emissions.

In some implementations the contextual bandit system 100 may be used to perform a classification task. The context may then comprise a set of features e.g. a feature vector, representing an item for classification, the action, selected from a set of possible actions, may correspond to a classification of the item, and the reward may be a reward for a correct classification e.g. a binary reward depending upon whether the classification was correct or a continuous valued reward corresponding to a rating of the classification. The system may be trained to perform any classification task. For example the features may represent the observations of the status of an item of equipment, plant or facility as previously described, and the classification may represent a performance measure for the equipment, plant or facility e.g. a measure of efficiency of operation, or a classification of a potential fault.

Generally, a GLN includes multiple layers that each include one or more neurons. Each neuron within a GLN generates as output a prediction, i.e., a probability, of the target for the entire GLN. That is, each neuron within a GLN predicts the same target, i.e., makes a prediction of the same target quantity, and the final output of the GLN is the prediction generated by the neuron in the last layer.

More specifically, each neuron in each layer of the GLN receives the context 120 and a set of predictions, i.e., a set of probabilities, and outputs a mixture, in particular a geometric mixture, of the received predictions using a set of weights for the neuron and in accordance with the context 120. The mixture may be considered as a weighted combination of the predictions. The mixture may be referred to as geometric as it may be considered as a

parameterized or weighted form of geometric averaging (which reduces to a geometric mean with equal weights).

This is contrary to conventional neural networks, where the input to the neural network is only provided to the first layer in the neural network and where the neurons in each layer other than the output layer generate alternative representation that are provided to the next layer.

In particular, within the GLN, each neuron that is in a layer that is after the first layer is configured to receive (i) the input context 120 and (ii) predictions from neurons in a previous layer and to (iii) apply a gating function to the input context to select a weight vector, i.e., a proper subset of the weights of the neuron, and (iv) generate as output a new probability that is a geometric mixture of the predictions from the neurons in the previous layer based on the selected weight vector.

Each neuron in the first layer is configured to receive (i) the input context 120 and (ii) a set of base predictions and to (iii) apply a gating function to the input context to select a weight vector and (iv) generate as output a geometric mixture of the base predictions based on the selected weight vector.

The operations of a GLN are described in more detail below with reference to FIG. 1B.

To select an action for a given context input 120, for each action in the set of actions, the system 110 processes the context input 120 using the one or more gated linear networks 130 corresponding to the action to generate either (i) a predicted probability that a reward of one will be received or (ii) a respective probability for each bin.

The system 100 then selects the action to be performed using the outputs of the gated linear networks by generating an action score 122 for the action from the outputs of the gated linear network(s) for the action and, in some cases, a pseudo-count. The pseudo-count is also generally determined from the one or more gated linear networks corresponding to the action as described below.

The system 100 then selects the action 106 to be performed in response to the context 120 based on the action scores 122, e.g., by selecting the action with the highest action score 122.

Selecting actions will be described in more detail below with reference to FIGS. 2 and 3.

In order to improve the quality of the actions that are selected, the training engine 150 repeatedly updates the GLNs 130 to cause the action selection system 110 to generate more accurate policy outputs, i.e., that result in higher rewards 124 being received by the system 100 in response to selected actions.

In particular, because of the architecture of the GLNs, the training engine 150 can train the GLNs online, i.e., in response to each received reward 124, without needing to perform a backward pass through any of the GLNs as would be required to train a conventional deep neural network.

More specifically, when a reward 124 is received, the training engine 150 can update the neurons in the GLN(s) corresponding to the selected action, i.e., the action that resulted in the reward 124 being received using a per-neuron loss. In other words, the engine 150 can update each neuron in a given GLN locally based on a neuron-specific loss. The update to any given neuron depends on only on the reward 142 and information computed as part of generating the action scores for the corresponding context, i.e., information computed during a single forward pass through the GLN. In particular, the update can depend on the reward, the probability predicted by the neuron, and the probabilities that were received as input by the neuron. Thus, the training engine 150 can update the neurons without requiring any additional forward passes through the GLN or any backward passes through the GLN.

This is in contrast to conventional deep neural networks, which require computationally expensive backpropagation in addition to one or more forward passes to compute an update to the weights of any given neuron within the deep neural network.

Updating the GLNs 130 will be described in more detail below with reference to FIGS. 2 and 3.

By repeatedly updating the GLNs 130, the training engine 150 can continue to improve the quality of the actions selected by the system 110.

FIG. 1B shows an example of one of the GLNs 130.

As shown in FIG. 1B, the GLN 130 receives side information z, i.e., a context 120, and generates as output a probability. The likelihood represented by the probability will depend on the implementation. For example, when the rewards are binary, the likelihood

represented by the probability is the likelihood that a reward will be received if the action corresponding to the GLN is performed. When the rewards are continuous values, in some implementations the likelihood represented by the probability is the likelihood that the reward will fall in one of two bins if the action corresponding to the GLN is performed. This is described in more detail below with reference to FIG. 3. In general however the probability defined by the output of the GLN 130 may define any exponential family probability distribution (which are closed under multiplication) e.g. a binomial distribution to represent multi-bit binary values or a Gaussian distribution, e.g. characterized by a mean and variance, to represent continuous valued data.

In the example of FIG. 1B, the GLN has an input layer 0 and three additional layers 1, 2, and 3. Input layer 0 has K0 neurons, layer 1 has K1 neurons, layer 2 has K2 neurons, and layer 3, the output layer, has a single neuron.

The output generated by the single neuron of the output layer, i.e., the probability p_{31} is the predicted probability of the GLN.

In an alternative implementation, instead of the output layer having a single neuron that generates a predicted probability that is used as the final output of the GLN, an output may be provided by aggregating outputs of neurons in a layer, e.g. layer 2 in this example.

The neurons in layer 0 serve as a "base model" for the remainder of the GLN and receive as input base probabilities that are based on the side information, i.e., the context 120, or data derived from the side information (rescaled if necessary).

In particular, because each neuron in the GLN computes a geometric mixture of the probabilities that are given as input to the neuron, the input to the neurons in layer 0 must be a set of probabilities, i.e., a set of values ranging between zero and one inclusive. If the context input satisfies this criterion, the neurons in layer 0 can directly receive the context input. Alternatively, the system 100 or another system can use a function to map the context 120 to a set of probabilities, e.g., a fixed scaling function or linear projection or some learned, non-linear transformation.

The neurons in each of layers 1, 2, and 3, receive as input the context 120 and the probabilities generated by the neurons in the preceding layer in the GLN and compute a new probability by computing a geometric mixture of the probabilities generated by the preceding

layer. As a particular example, the neuron 1,1 in layer 1 receives as input the probabilities generated by the neurons in layer 0 and generates as output a new probability p_{11}.

In particular, each neuron in the GLN has a set of weights that is represented as a plurality of weight vectors.

To generate an output probability, a given neuron applies a gating function to the context input 120 to select one of the plurality of weight vectors and then uses the selected weight vector to perform a geometric mixture of the received probabilities to generate a new probability. In particular, the operation performed by a neuron *ij* of the GLN to generate as output a probability p_{ij} can be expressed as:

$$p_{ij} = \sigma(w_{ij_{g_{ij}(z)}} \cdot \text{logit}(p_{i-1})),$$

where σ is the sigmoid function, logit is the logit function, i.e., the inverse of the sigmoid function (i.e. overall the neuron is linear), $w_{ij_{g_{ij}(z)}}$ is the weight vector selected by applying the gating function $g_{ij}(z)$ for the neuron *ij* to the context z, and p_{i-1} are the probabilities received as input from the layer *i*-1.

In some implementations, to improve the stability of the training of the GLN, the system clips the values of the probabilities so the probabilities fall in a particular range, e.g., the range of ε to 1 - ε inclusive for some constant ε between zero and one. In these cases, if any probability is below ε, the system instead sets the probability to ε. If any probability is above 1 - ε, the system instead sets the value to 1 - ε.

The gating function for each neuron is fixed before training while the weight vectors are learned during training. Updating the weight vectors is described in more detail below with reference to FIGS. 2 and 3.

The neurons in the GLN can use any appropriate gating function that maps the context input to one of the weight vectors maintained by the neuron.

As a particular example, the gating function for each neuron in the GLN can be a half-space gating function.

The half-space gating function is described in more detail in Veness, et al, Gated Linear Networks, arXiv:1910.01526.

In the example of FIG. 1B, each layer other than the output layer (layer 3), also includes a bias neuron, i.e., the neurons 00, 10, and 20, that generates as output a fixed

probability, i.e., a probability that is set prior to training of the GLN. However, this is optional and in some cases some or all of the layers of the GLN will not include a bias neuron.

FIG. 2 is a flow diagram of an example process 200 for selecting an action to be performed in response to a context input. For convenience, the process 200 will be described as being performed by a system of one or more computers located in one or more locations. For example, a contextual bandits system, e.g., the contextual bandits system 100 of FIG.1, appropriately programmed, can perform the process 200.

In particular, the system can perform the process 200 when the rewards are binary values and each GLN corresponds to a different action from the set of actions that can be performed in response to a context input.

That is, the system maintains data specifying a respective gated linear network corresponding to each of the plurality of actions (step 202). The gated linear network that corresponds to any given action is configured to predict a probability that a reward will be received, i.e., that the reward received will be equal to 1 and not to zero, if the corresponding given action is performed in response to an input context.

The system then performs steps 202-208 (and optionally 210 and 212) in response to each context input that is received. That is, the system can perform the steps 202-208 or the steps 202-212 in response to each context input in a sequence of context inputs to select an action to be performed in response to each context input.

For each action, the system processes the context input using the gated linear network corresponding to the action to generate a predicted probability for the action (step 204).

For each action, the system generates an action score for the action from at least the predicted probability for the action (step 206). That is, in some implementations, the predicted probability is used as the action score. In some other implementations, the system generates the action score for the action from the predicted probability for the action and one or more other factors.

As a particular example, the system can compute a pseudo-count for the action and then generate the action score for the action from the predicted probability and the pseudo-count. Generally, a pseudo-count is a generalized measure of the count of times that an action has been selected over some historical time window, e.g., some window of time steps

in the sequence that ends at the current time step. Computing the action scores from both the pseudo-count and from the predicted probability can encourage exploration of the possible space of context – action pairs and result in a higher quality action selection policy.

As a particular example, when the pseudo-count is used, the action score for an action a at a time step t in the sequence of context inputs can satisfy:

$$\mathrm{GLN}_a^{\bar{t}}(x_t) + C\sqrt{\frac{\log t}{\widehat{N}_{\bar{t}}(a)}},$$

where $\mathrm{GLN}_a^{\bar{t}}(x_t)$ is the predicted probability generated by the GLN for the action a given the context x_t, C is a positive constant value that scales the exploration bonus, and $\widehat{N}_{\bar{t}}(a)$ is the pseudo-count.

The system can compute a pseudo-count for an action in any of a variety of ways. As a particular example, the system can compute the pseudo-count based on an overlap between (i) a signature of the context across the gating functions of the neurons in the gated linear network for the action and (ii) signatures of any earlier contexts in the sequence for which the action was selected as the action to be performed in response to the earlier context.

As used in this specification, the signature of a context refers to the outputs of the gating functions of the neurons in a GLN generated by processing the context. For example, the signature for a given context for a given action can be a vector that includes the output of the gating function, i.e., that identifies the weight vector selected using the gating function. for each neuron in the GLN corresponding to the given action. In implementations each action-specific GLN uses the same collection of gating functions, and the signature computation can be re-used when evaluating each GLN.

Thus, the overlap of a signature corresponding to one time step with another signature corresponding to another time step identifies which gating functions generated the same output, i.e., identified the same weight vector for both the first time step and the other time step.

As a particular example, the system can determine, for a given action and for each neuron in the GLN for the action, a count of how many earlier time steps in the window satisfy both of the following conditions: (1) the given action was selected at the earlier time step and (2) the output of the gating function for the neuron for the context at the earlier time step is the same as the output of the gating function for the neuron for the current time step.

The system can then determine the pseudo-count by aggregating the respective counts for the neurons, e.g., by setting the pseudo-count equal to the mean, minimum, maximum, or median of the respective counts.

The system selects an action to be performed in response to the context based on the action scores for the actions in the set of actions (step 208). For example, the system can select the action with the highest action score or can map the action scores to a probability distribution and then sample an action from the probability distribution.

Optionally, the system can then update the weights of the neurons of the GLN for the selected action by performing steps 210-212. In particular, in some cases the system updates the weights of the GLN after each context in the sequence (completely on-line). In other cases, the system can only update the weights of the GLN after every N contexts in the sequence or can freeze the weights after a threshold number of weight updates have been made.

The system receives a reward as a result of the selected action being performed in response to the context (step 210). As described above, when the process 200 is used by the system to select actions, the reward will generally be a binary reward that is equal to one or zero.

The system updates the GLN for the selected action based on the received rewards (step 212). More specifically, the system updates each neuron in the gated linear network, i.e., updates the weights of each neuron in the gated linear network, locally based on a neuron-specific loss. Thus, the system does not have to perform computationally-expensive backpropagation in order to compute the loss.

The loss for a given neuron generally measures the error between the reward and the probability predicted by the given neuron given the input probabilities to the neuron. The system updates the weights of the neuron by updating the selected weight vector by computing a gradient of the loss.

In particular, the gradient of the loss for a given neuron j in layer i can satisfy:

$$-n(p_{ij} - r)\text{logit}(p_{i-1})),$$

where n is a learning rate for the updating of the GLN, logit is the logit function, i.e., the inverse of the sigmoid function, p_{ij} is the probability predicted by the neuron j in layer i, r is

the received reward, and p_{i-1} are the probabilities received as input from the layer i-1. Thus, as can be seen from this

The system can then update the selected weight vector by adding or subtracting the gradient to the weight vector. In implementations, to improve the stability of the updating, the system clips the values of the weights so the weights fall in a particular range, e.g., the range of –b to b inclusive for some constant b between 10 and 100. In these cases, if any value in the new weight vector is below the particular range, the system instead sets the value to the lowest value in the particular range. If any value in the new weight vector is above the particular range, the system instead sets the value to the highest value in the particular range.

FIG. 3 is a flow diagram of another example process 300 for selecting an action to be performed in response to a context input. For convenience, the process 300 will be described as being performed by a system of one or more computers located in one or more locations. For example, a contextual bandits system, e.g., the contextual bandits system 100 of FIG.1, appropriately programmed, can perform the process 300.

In particular, the system can perform the process 300 when the rewards are continuous values from a fixed range and, for each action, there is a corresponding tree of GLNs.

That is, the system maintains data specifying a respective tree of gated linear networks corresponding to each of the plurality of actions (step 302). The gated linear networks in the tree that corresponds to any given action are configured to predict a respective probability for each of a plurality of bins of a range of reward values given an input context.

In other words, the system discretizes the fixed range into multiple non-overlapping bins and the tree of gated linear networks in the tree that corresponds to any given action collectively predict a respective probability for each of the bins.

The respective probability for each bin represents a likelihood that a reward that falls in the bin will be received if the corresponding action is performed in response to the input context.

More specifically, for each action, when the number of bins is equal to 2^D, the system can maintain data representing a binary tree of depth D that includes a GLN at each non-leaf node. Each non-leaf node at any level $d<D$ in the tree has two child nodes in the

level $d+1$ and each leaf node corresponds to one of the bins. Thus, at each level d in the tree, the binary tree divides the bounded reward range uniformly into 2^d bins that get smaller in size as d increases.

Accordingly, the probability that is predicted by each GLN is the probability that, assuming that the reward falls within the bin represented by one of the child nodes of the GLN in the tree, the reward will fall in the bin represented by one of the child nodes instead of in the other. For example, the right branch from each GLN can be associated with a label of 1 and the left branch can be associated with a label of 0. A prediction of a given GLN then represents the likelihood that the reward falls in the bin represented by the child node connected by the right branch to the GLN in the tree.

Thus, each leaf node, i.e., each of the 2^D bins, can only be reached by a single, unique path starting from the root node of the tree.

Thus, once each GLN in the tree has generated a probability, the system can compute the overall probability for a given bin as the product of, for each GLN along the unique path for the bin, the probability assigned by the GLN to the next node along the path. That is, continuing the example above where the right branch from each GLN is associated with a label of 1 and the left branch is associated with a label of 0, the probability assigned by a given GLN to the child node along the right branch from the given GLN is equal to the probability predicted by the given GLN while the probability assigned by the given GLN to the child node along the left branch is equal to 1 – the probability predicted by the given GLN.

In particular, if the path through the tree to a given bin is represented as a binary vector b of length D, where each value in the vector identifies the next node on the path through the tree (given the earlier nodes on the path) and the value is 1 if the next node is assigned a label of 1 and 0 if the next node is assigned a label of 0, the probability for the given bin satisfies:

$$\prod_{i=1}^{D} |1 - b_i - GLN_{i-1}(x_t)|,$$

where b_i is the i-th value in b, $GLN_0(x_t)$ is the output of the GLN that is the root node of the tree given the context x_t and $GLN_{i-1}(x_t)$ is the output of the GLN that is identified by b_{i-1}.

The system also maintains, for each bin, a representative value, e.g., the midpoint between the lowest value in the bin and the highest value in the bin.

The system then performs steps 302-308 (and optionally 310 and 312) in response to each context input that is received. That is, the system can perform the steps 302-308 or the steps 302-312 in response to each context input in a sequence of context inputs to select an action to be performed in response to each context input.

For each action, the system processes the context input using the tree of gated linear networks corresponding to the action to generate a respective predicted probability for each of the bins (step 304).

For each action, the system generates an action score for the action from at least the predicted probabilities for the action (step 306).

In particular, for each action, the system can generate an expected reward for the action by computing a weighted sum of the representative values for the bins, with each being weighted by the probability predicted for the corresponding bins by the tree of GLNs for the action.

In some implementations, the expected reward is used as the action score. In some other implementations, the system generates the action score for the action from the predicted probability for the action and one or more other factors.

As a particular example and as described above, the system can compute a pseudo-count for the action and then generate the action score for the action from the predicted probability and the pseudo-count. As indicated above, computing the action scores from both the pseudo-count and from the predicted probability can encourage exploration of the possible space of context – action pairs and result in a higher quality action selection policy.

As a particular example, when the pseudo-count is used, the action score for an action a at a time step t in the sequence of context inputs can satisfy:

$$\mathrm{GLN}_a^{\bar{t}}(x_t) + C\sqrt{\frac{\log t}{\widehat{N}_{\bar{t}}(a)}},$$

where $\mathrm{GLN}_a^{\bar{t}}(x_t)$ is the predicted probability for the action a given the context x_t generated from the outputs of the tree of GLNs for the action, C is a positive constant value that scales the exploration bonus, and $\widehat{N}_{\bar{t}}(a)$ is the pseudo-count.

The system can compute a pseudo-count for an action in any of a variety of ways.

As a particular example, the system can compute the pseudo-count based on an overlap between (i) a signature of the context across the gating functions of the neurons in

the gated linear networks in the tree of GLNs for the action and (ii) signatures of the context across the gating functions of the neurons in the gated linear networks in the tree of GLNs for any earlier contexts in the sequence for which the action was selected as the action to be performed in response to the earlier context.

As described above, the overlap of a signature corresponding to one time step with another signature corresponding to another time step identifies which gating functions generated the same output for both the first time step and the other time step.

As a particular example, the system can determine, for a given action and for each neuron in the GLNs in the tree of GLNs for the action, a count of how many earlier time steps in the window satisfy both of the following conditions: (1) the given action was selected at the earlier time step and (2) the output of the gating function for the neuron for the context at earlier time step is the same as the output of the gating function for the neuron for the current time step.

The system can then determine the pseudo-count by aggregating the respective counts for the neurons, e.g., by setting the pseudo-count equal to the mean, minimum, maximum, or median of the respective counts.

That is, unlike computing pseudo-counts when only a single GLNs is maintained for each action (as described above with reference to step 206), when a tree of GLNs is maintained for each action, the counts are aggregated across all of the neurons in all of the GLNs in the tree.

The system selects an action to be performed in response to the context based on the action scores for the actions in the set of actions (step 308). For example, the system can select the action with the highest action score or can map the action scores to a probability distribution and then sample an action from the probability distribution.

Optionally, the system can then update the weights of the neurons in the tree of GLNs for the selected action by performing steps 310-312. In particular, in some cases the system updates the weights of the GLNs in the tree after each context in the sequence (completely on-line). In other cases, the system can only update the weights of the GLNs only after every N contexts in the sequence or can freeze the weights after a threshold number of weight updates have been made.

The system receives a reward as a result of the selected action being performed in response to the context (step 310).

The system updates the tree of GLN for the selected action based on the received reward (step 312).

More specifically, the system identifies the bin to which the received reward belongs and updates each neuron that is in any GLN that is on the path to the identified bin through the tree for the selected action.

As above, the system updates updates the weights of each neuron in each of the gated linear networks that are on the path locally based on a neuron-specific loss. Thus, the system does not have to perform computationally-expensive gradient descent in order to compute the loss.

The loss for a given neuron generally measures the error between a target for the GLN to which the neuron belongs (that is based on the received reward) and the probability predicted by the given neuron given the input probabilities to the neuron.

The system updates the weights of the neuron by updating the selected weight vector by computing a gradient of the loss. In particular, the system sets the target for a given GLN on the path as 1 if the next GLN on the path is associated with a label of 1 and as 0 if the next GLN on the path is associated with a label of 0.

In particular, the gradient of the loss for a given neuron j in layer i of a given GLN can satisfy:

$$-n(p_{ij} - target)\text{logit}(p_{i-1}),$$

where n is a learning rate for the updating of the GLN, logit is the logit function, i.e., the inverse of the sigmoid function, p_{ij} is the probability predicted by the neuron j in layer i, $target$ is the target for the GLN to which the neuron belongs and is based on the received reward r, and p_{i-1} are the probabilities receive as input from the layer i-1.

The system can then update the selected weight vector by adding or subtracting the gradient to the weight vector. In implementations, to improve the stability of the updating, the system clips the values of the weights so the weights fall in a particular range, e.g., the range of –b to b inclusive for some constant b between 10 and 100. In these cases, if any value in the new weight vector is below the particular range, the system instead sets the value

to the lowest value in the particular range. If any value in the new weight vector is above the particular range, the system instead sets the value to the highest value in the particular range.

This specification uses the term "configured" in connection with systems and computer program components. For a system of one or more computers to be configured to perform particular operations or actions means that the system has installed on it software, firmware, hardware, or a combination of them that in operation cause the system to perform the operations or actions. For one or more computer programs to be configured to perform particular operations or actions means that the one or more programs include instructions that, when executed by data processing apparatus, cause the apparatus to perform the operations or actions.

Embodiments of the subject matter and the functional operations described in this specification can be implemented in digital electronic circuitry, in tangibly-embodied computer software or firmware, in computer hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Embodiments of the subject matter described in this specification can be implemented as one or more computer programs, i.e., one or more modules of computer program instructions encoded on a tangible non transitory storage medium for execution by, or to control the operation of, data processing apparatus. The computer storage medium can be a machine-readable storage device, a machine-readable storage substrate, a random or serial access memory device, or a combination of one or more of them. Alternatively or in addition, the program instructions can be encoded on an artificially generated propagated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus for execution by a data processing apparatus.

The term "data processing apparatus" refers to data processing hardware and encompasses all kinds of apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can also be, or further include, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit). The apparatus can optionally include, in addition to hardware, code that creates an execution environment for computer programs, e.g., code that constitutes processor firmware, a

protocol stack, a database management system, an operating system, or a combination of one or more of them.

A computer program, which may also be referred to or described as a program, software, a software application, an app, a module, a software module, a script, or code, can be written in any form of programming language, including compiled or interpreted languages, or declarative or procedural languages; and it can be deployed in any form, including as a stand alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A program may, but need not, correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data, e.g., one or more scripts stored in a markup language document, in a single file dedicated to the program in question, or in multiple coordinated files, e.g., files that store one or more modules, sub programs, or portions of code. A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a data communication network.

In this specification, the term "database" is used broadly to refer to any collection of data: the data does not need to be structured in any particular way, or structured at all, and it can be stored on storage devices in one or more locations. Thus, for example, the index database can include multiple collections of data, each of which may be organized and accessed differently.

Similarly, in this specification the term "engine" is used broadly to refer to a software-based system, subsystem, or process that is programmed to perform one or more specific functions. Generally, an engine will be implemented as one or more software modules or components, installed on one or more computers in one or more locations. In some cases, one or more computers will be dedicated to a particular engine; in other cases, multiple engines can be installed and running on the same computer or computers.

The processes and logic flows described in this specification can be performed by one or more programmable computers executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by special purpose logic circuitry, e.g., an FPGA or an ASIC, or by a combination of special purpose logic circuitry and one or more programmed computers.

Computers suitable for the execution of a computer program can be based on general or special purpose microprocessors or both, or any other kind of central processing unit. Generally, a central processing unit will receive instructions and data from a read only memory or a random access memory or both. The elements of a computer are a central processing unit for performing or executing instructions and one or more memory devices for storing instructions and data. The central processing unit and the memory can be supplemented by, or incorporated in, special purpose logic circuitry. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Moreover, a computer can be embedded in another device, e.g., a mobile telephone, a personal digital assistant (PDA), a mobile audio or video player, a game console, a Global Positioning System (GPS) receiver, or a portable storage device, e.g., a universal serial bus (USB) flash drive, to name just a few.

Computer readable media suitable for storing computer program instructions and data include all forms of non volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks.

To provide for interaction with a user, embodiments of the subject matter described in this specification can be implemented on a computer having a display device, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input. In addition, a computer can interact with a user by sending documents to and receiving documents from a device that is used by the user; for example, by sending web pages to a web browser on a user's device in response to requests received from the web browser. Also, a computer can interact with a user by sending text messages or other forms of message to a personal device,

e.g., a smartphone that is running a messaging application, and receiving responsive messages from the user in return.

Data processing apparatus for implementing machine learning models can also include, for example, special-purpose hardware accelerator units for processing common and compute-intensive parts of machine learning training or production, i.e., inference, workloads.

Machine learning models can be implemented and deployed using a machine learning framework, .e.g., a TensorFlow framework, a Microsoft Cognitive Toolkit framework, an Apache Singa framework, or an Apache MXNet framework.

Embodiments of the subject matter described in this specification can be implemented in a computing system that includes a back end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front end component, e.g., a client computer having a graphical user interface, a web browser, or an app through which a user can interact with an implementation of the subject matter described in this specification, or any combination of one or more such back end, middleware, or front end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network (LAN) and a wide area network (WAN), e.g., the Internet.

The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other. In some embodiments, a server transmits data, e.g., an HTML page, to a user device, e.g., for purposes of displaying data to and receiving user input from a user interacting with the device, which acts as a client. Data generated at the user device, e.g., a result of the user interaction, can be received at the server from the device.

While this specification contains many specific implementation details, these should not be construed as limitations on the scope of any invention or on the scope of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. Certain features that are described in this specification in the

context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially be claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

Similarly, while operations are depicted in the drawings and recited in the claims in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system modules and components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

Particular embodiments of the subject matter have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In some cases, multitasking and parallel processing may be advantageous.

CLAIMS

1. A method of selecting an action from a set of actions to be performed in response to each context in a sequence of contexts, the method comprising:

maintaining data specifying a respective gated linear network corresponding to each of the plurality of actions, wherein each gated linear network is configured to predict a probability that a reward will be received if the corresponding action is performed in response to an input context, and wherein each gated linear network comprises a plurality of layers each comprising one or more neurons, wherein each neuron in each layer after the first layer is configured to receive (i) the input context and (ii) predictions from neurons in a previous layer and to (iii) apply a gating function to the input context to select a weight vector and (iv) generate as output a geometric mixture of the predictions from the neurons in the previous layer based on the selected weight vector;

for each context in the sequence of contexts:

for each action, processing the context using the gated linear network corresponding to the action to generate a predicted probability;

for each action, generating an action score for the action from at least the predicted probability; and

selecting the action to be performed in response to the context based on the action scores.

2. The method of claim 1, wherein selecting the action to be performed in response to the context based on the action scores comprises selecting the action with a highest action score.

3. The method of any preceding claim, wherein for each action, generating an action score for the action from at least the predicted probability comprises:

computing a pseudo-count for the action; and

generating the action score from the predicted probability for the action and the pseudo-count for the action.

4. The method of claim 3, wherein computing a pseudo-count comprises:

determining an overlap between (i) a signature of the context across the gating functions of the neurons in the gated linear network for the action and (ii) signatures of any earlier contexts in the sequence for which the action was selected as the action to be performed in response to the earlier context.

5. The method of any preceding claim, further comprising:

for each context in the sequence of contexts:

receiving a reward; and

updating the gated linear network for the selected action based on the reward.

6. The method of claim 5, wherein updating the gated linear network for the selected action comprises:

updating each neuron in the gated linear network locally based on a neuron-specific loss.

7. The method of any preceding claim, wherein a last layer of the plurality of layers includes only a single neuron, and wherein the predicted probability of the gated linear network is the output of the single neuron.

8. The method of any preceding claim, wherein the neurons in the first layer of the plurality of layers receive the input context and a set of base predictions.

9. A method of selecting an action from a set of actions to be performed in response to each context in a sequence of contexts, the method comprising:

maintaining data specifying a respective tree of gated linear networks corresponding to each action of the plurality of actions, wherein each tree of gated linear networks is collectively configured to predict a respective probability for each of a plurality of bins of a range of reward values, wherein the respective probability for each bin represents a likelihood that a reward that falls in the bin will be received if the corresponding action is performed in response to an input context, and wherein each gated linear network comprises a plurality of layers each comprising one or more neurons, wherein each neuron in each layer after the first layer is configured to receive (i) the input context and (ii) predictions from neurons in a previous layer and to (iii) apply a gating function to the input context to select a weight vector and (iv) generate as output a geometric mixture of the predictions from the neurons in the previous layer based on the selected weight vector;

for each context in the sequence of contexts:

for each action, processing the context using the tree of gated linear networks corresponding to the action to generate a respective probability for each of the plurality of bins of the range of reward values;

for each action, generating an action score for the action from at least the respective probabilities; and

selecting the action to be performed in response to the context based on the action scores.

10. The method of claim 9, wherein for each action, generating an action score for the action from at least the respective probabilities comprises:

computing a pseudo-count for the action based on (i) the outputs of the gating functions of the gated linear networks in the tree of gated linear networks corresponding to the action when processing the context and (ii) outputs of the gating function of the gated linear networks in the tree for earlier contexts.

11. A system comprising one or more computers and one or more storage devices storing instructions that when executed by the one or more computers cause the one or more computers to perform the operations of the method any one of claims 1 to 10.

12. One or more computer storage media storing instructions that when executed by one or more computers cause the one or more computers to perform the operations of the method any one of claims 1 to 10.

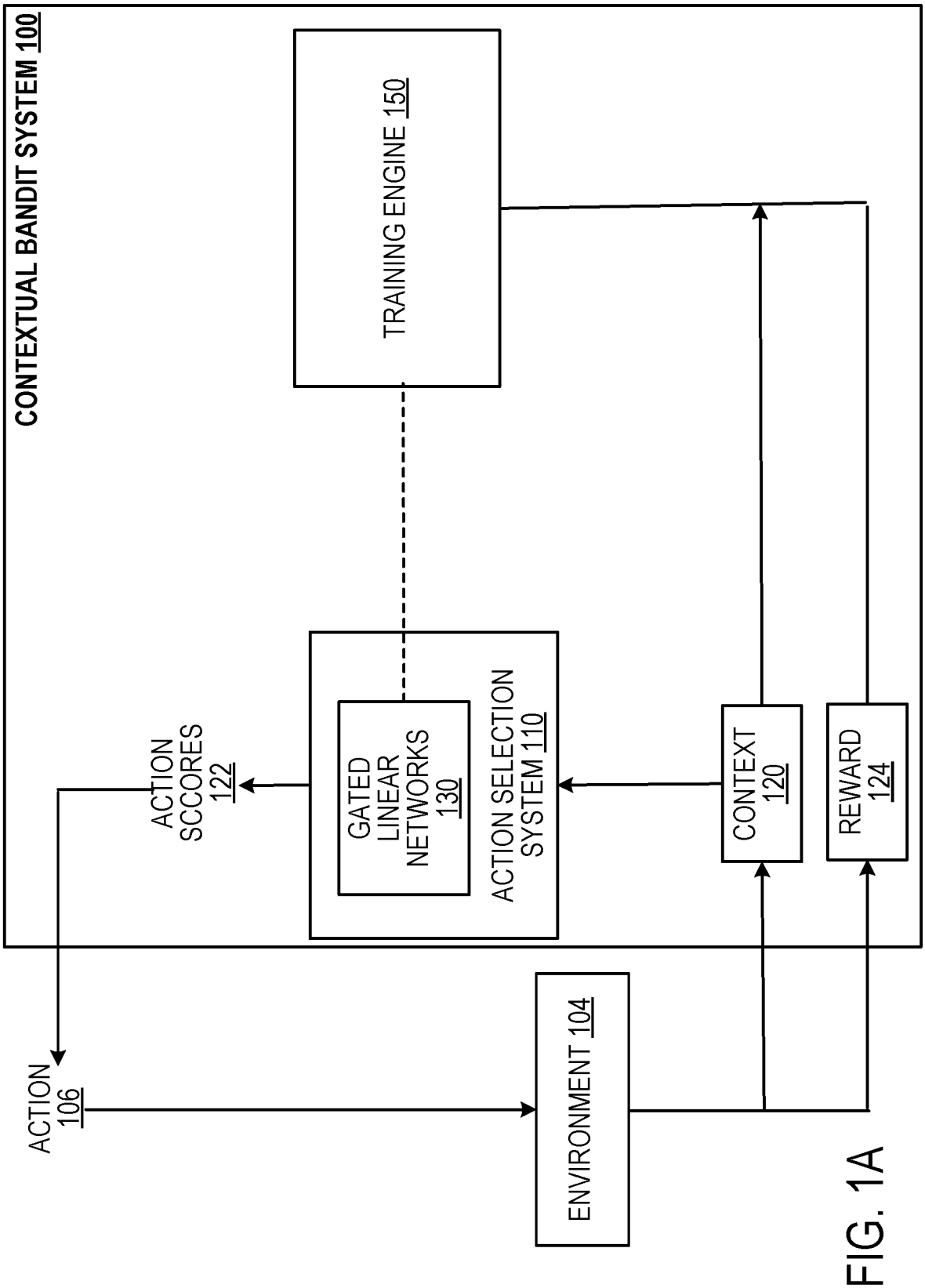
CONTEXTUAL BANDIT SYSTEM 100
TRAINING ENGINE 150
ACTION SELECTION SYSTEM 110
GATED LINEAR NETWORKS 130
ACTION SCORES 122
CONTEXT 120
REWARD 124
ACTION 106
ENVIRONMENT 104

FIG. 1A

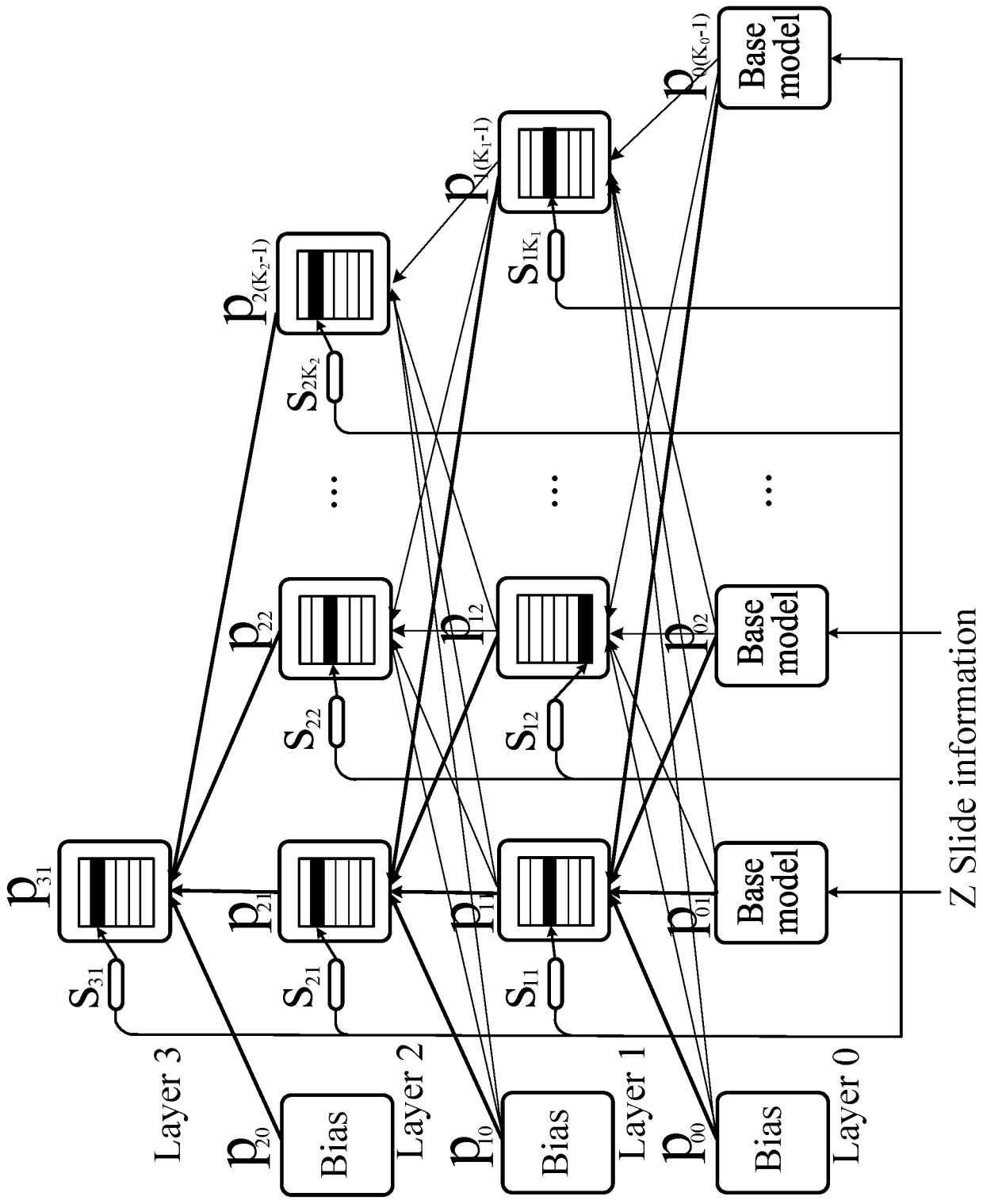
Layer 3
Layer 2
Layer 1
Layer 0
Bias
Base model
Z Slide information

FIG. 1B

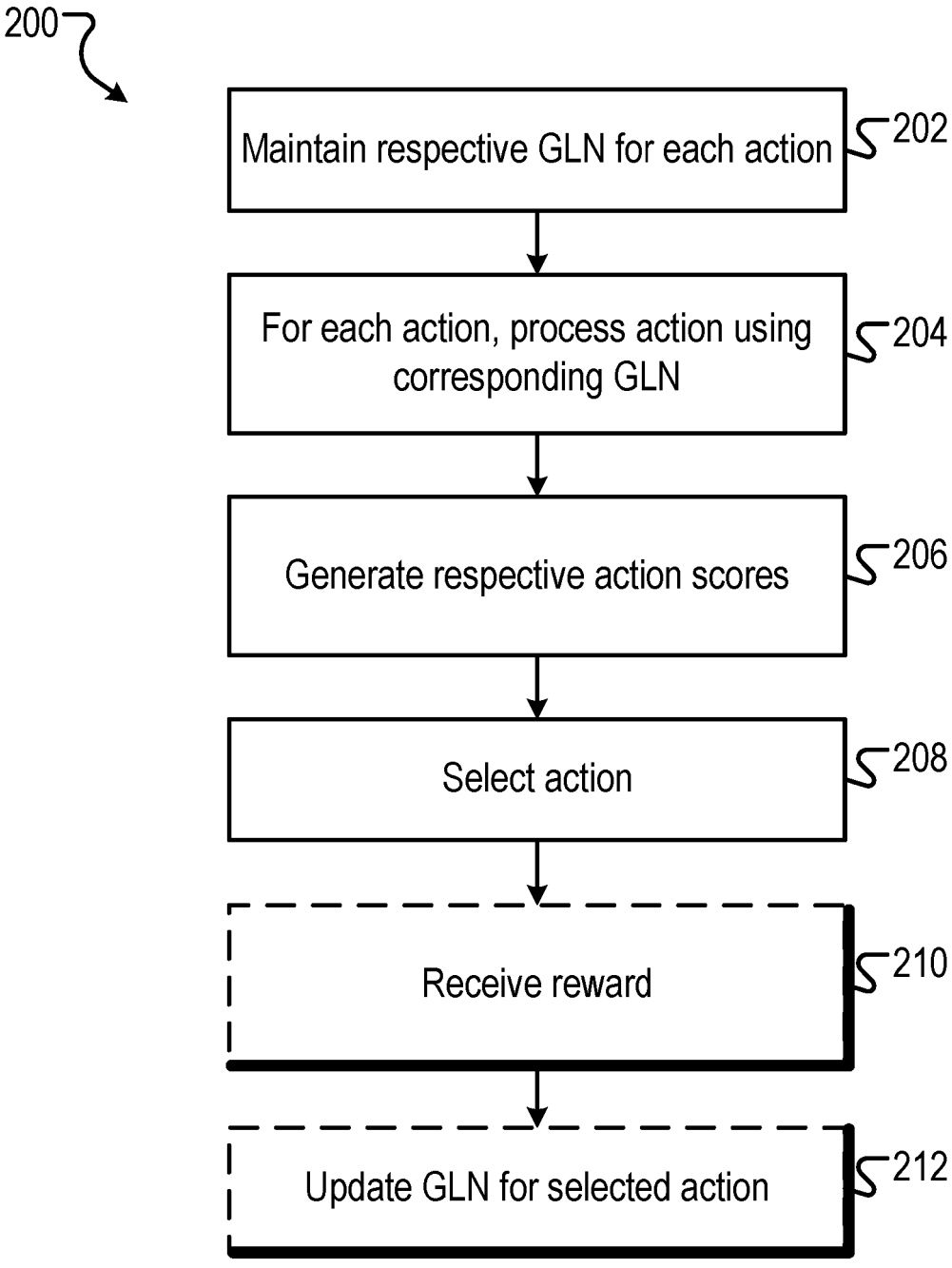
200
Maintain respective GLN for each action
202
For each action, process action using corresponding GLN
204
Generate respective action scores
206
Select action
208
Receive reward
210
Update GLN for selected action
212

FIG. 2

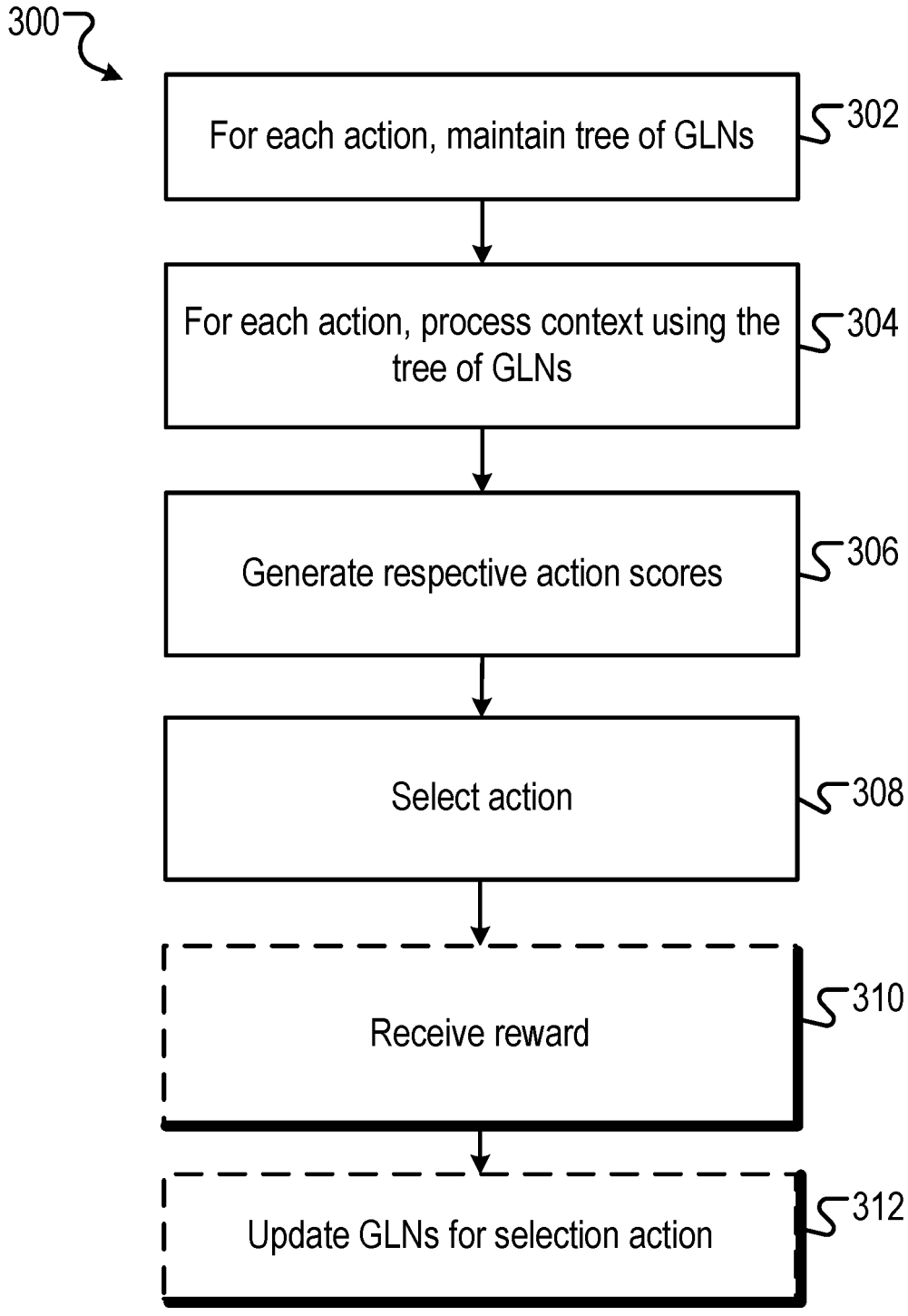
300
For each action, maintain tree of GLNs
302
For each action, process context using the tree of GLNs
304
Generate respective action scores
306
Select action
308
Receive reward
310
Update GLNs for selection action
312

FIG. 3

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2020/078259

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06N3/00 G06N3/04 G06N3/08 G06N7/00
ADD.

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)
G06N

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Robin Allesiardo ET AL: "A Neural Networks Committee for the Contextual Bandit Problem" In: "Lecture Notes in Computer Science", 1 January 2014 (2014-01-01), Springer Berlin Heidelberg, Berlin Germany, XP055771795, ISSN: 0302-9743 vol. 8834, pages 374-381, DOI: 10.1007/978-3-319-12637-1_47, page 374 - page 377	1-8,11, 12
A	WO 2019/106132 A1 (GRABSKA-BARWINSKA AGNIESZKA [GB] ET AL) 6 June 2019 (2019-06-06) page 1 - page 19; figures 1-4	1-12

-/--

[X] Further documents are listed in the continuation of Box C.

[X] 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
5 February 2021	16/02/2021

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Jacobs, Jan-Pieter

1

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	ADAM N ELMACHTOUB ET AL: "A Practical Method for Solving Contextual Bandit Problems Using Decision Trees", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 15 June 2017 (2017-06-15), XP081419136, page 1 - page 4	9,10
A	QIANGFU ZHAO: "Evolutionary design of neural network tree - integration of decision tree, neural network and GA", EVOLUTIONARY COMPUTATION, 2001. PROCEEDINGS OF THE 2001 CONGRESS ON MAY 27-30, 2001, PISCATAWAY, NJ, USA,IEEE, vol. 1, 27 May 2001 (2001-05-27), pages 240-244, XP010551802, ISBN: 978-0-7803-6657-2 abstract the whole document	9,10
A	RUSSO DANIEL ET AL: "Learning to Optimize via Posterior Sampling", MATHEMATICS OF OPERATIONS RESEARCH, vol. 39, no. 4, 23 April 2014 (2014-04-23) , pages 1221-1243, XP055772522, US ISSN: 0364-765X, DOI: 10.1287/moor.2014.0650 page 1224 - page 1226	3,4

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2020/078259

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2019106132 A1	06-06-2019	US 2020349418 A1	05-11-2020
		WO 2019106132 A1	06-06-2019