AIA. Artificial intelligence for art

Robert Lisek
Artistic Development and design
Director, Institute for Research in Science and Art

Abstract

We observe the success of artificial neural networks in simulating human performance on a number of tasks: such as image recognition, natural language processing, etc. However, there are limits to state of-the-art AI that separate it from human-like intelligence. Humans can learn a new skill without forgetting what they have already learned and they can improve their activity and gradually become better learners. Today’s AI algorithms are limited in how much previous knowledge they are able to keep through each new training phase and how much they can reuse. In practice this means that it is necessary to build and adjust new algorithms to every new particular task. This is closer to a sophisticated data processing than to real intelligence. This is why research concerning generalisation are becoming increasingly important. Processes such as intuition, emotions, planning, thinking and abstraction are a part of processes, which occur in the human brain. Abstraction allows for making analogies, coding relations and relations between relations.

Keywords

Artificial, intelligence, art, AGI , Recurrent Neural Network

Generalization is a process in which the brain observes that a certain fact referring to a set of objects, refers to a greater set of objects. Processes occurring in the brain have an extremely plastic and dynamic character and cannot be reduced to one basic construction and operation. Many processes have very distributed character, for instance memories are not located in a particular place; the brain has holographic character [Pribram 1991]. A special role of a kind of creators is played by random processes, which allow for collision and splitting of structures, and leaps between different levels of generality. A generalization in AI means that system can generate new compositions or find solutions for new tasks that are not present in the training corpus. General Neural Model and intelligent agent should have very general learning capabilities, should not just be able to memorize the solution to a fixed set of tasks during creating of stories, but learn how to generalize to new problems it encounters. It can generalize problem in the sens that solving one or more of tasks should make solving other task easier. There is domain called AGI where will be possible to find solutions for this problems.

Artificial general intelligence (AGI) describes research that aims to create machines capable of general intelligent action. "General" means that one AI program realizes number of different tasks and the same code can be use in many applications. We must focus on self-improvement techniques e.g. Reinforcement Learning and integrate it with deep learning, recurrent networks, etc.

Recurrent Neural Networks

Models of sequential data, such as natural language, speech and video, are the core of many machine learning applications. Recurrent Neural Network is a powerful model that learns temporal patterns in sequential data. A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle, meaning that Recurrent Neural Network contains feedback connections, connections from any unit to itself. This allows it to exhibit dynamic temporal behavior. Unlike feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as handriting recognition, speech recognition and after modification as a performative and/or compositional tool for composer and musicians. Creating of feedback in RNN provides interesting creative posibilities, recurrent neural networks can evolve to a unstable states and they can create chaotic or random outputs. Chaotic behavior of recurrent neural networks has been observed before eg. by Maas (Maas et all, 1990).

It was shown that smooth variation of one of the parameters of the original map gives rise to period-doubling bifurcations. Feedback and chaotic behavior of RNN causes that some artists and composers are starting to use RNN in their artistic work. For example CTRNN Continuous Time Neural Networks are implemented in modular extensible computer music platforms such as Supercollider, Pure Data, MaxMsp. Sound or video parameters can evolve and be formed by using Recurrent Neural Networks. Photo CTRNN in action RNN models can be uses to tasks such as handwriting recognition, speech recognition, natural language processing, video recognition, etc. Natural language modeling has been widely studied in the past (Goodman, 2001b; Young et al., 1997; Koehn et al., 2007). In particular, models based on RNN have been very successful in automatic speech recognition (Dahl et al., 2012), language modeling (Mikolov, 2012) and video classification (Simonyan & Zisserman, 2014).

RNN represents time recursively. For example, in the simple recurrent network, the state of the hidden layer at a given time is conditioned on its previous state. This recursion allows the model to store complex signals for arbitrarily long time periods, as the state of the hidden layer can be seen as the memory of the model. However there is a problem of learning long term patterns in Recurrent Neural Networks. Recurrent networks was difficult to train using simple optimizers, such as stochastic gradient descent, due to the vanishing gradient problem. For example the sigmoid function have a gradient which is close to zero almost everywhere or the gradient can be backpropagated through time, its magnitude is multiplied over and over by the recurrent matrix. If the eigenvalues of this matrix are small (i.e., less than one), the gradient will converge to zero rapidly. Learning longer term patterns in real data, such as in natural language, is possible using gradient descent by using a structural modification of the simple recurrent neural network architecture. Many modifications were proposed to deal with the vanishing gradients eg. the long short term memory (LSTM) recurrent neural network (Hochreiter & Schmidhuber, 1997) is a modified version of recurrent network with gates which control flow of information to hidden neurons. This allows the network to potentially remember information for longer periods.

Most of the research on the use of AI in interactive applications concerns computer games, beginning with traditional two-player adversarial games like tic-tac-toe and extending to modern strategy games. This type of research however, has a limited application in storytelling or art, because the goal of AI agents in these games is to maximise reward, which often fails to advance the narrative threads and almost entirely overlooks the creation of interesting scenarios or compositions. Magerko (2009) conducted research with theatre performers to explore how to create scenes in real time without a preexisting scenario. Unfortunately, the basic conclusion of this research is that the actor should act on the basis of a huge set of initial scripts, which contain expectations as to what people do in different scenarios. There have been several attempts to implement the above approaches through introducing text into the game, building chatbots or intelligent assistants. However, these approaches have a limited scope, because they do not encompass the numerous phenomena known in natural language, such as the creation of metaphors, analogies and generalisations, which are crucial for human thinking and for creating stories. If the AI program works in order to create a story, it must be prepared to understand everything that the human might think and must be able to communicate in a natural language. Neural language models have garnered research interest for their ability to learn complex syntactic and semantic representations of natural language (Sutskever et al., 2014). Recurrent Neural Network (RNN) is a powerful model that learns temporal patterns in sequential data. Wen et al. (2015) proposed a RNN approach to the generation of utterances from dialog acts but their system requires one to preprocess the data.

Deep Reinforcement Learning

Thinking about programs in terms of binary codes and about functioning of the brain in terms of self-improvement through optimization of the codes, we may perceive the search for brain model as the search for best learning algorithms, and as an attempt of creating best predictions. The most interesting AI method is Reinforcement Learning. The brain works without deus ex machine, the rule of its organization is the rule of the shortest description, which allows for choosing shortest models of reality. A program is a set states that represents a given situation and the set of operators (actions) and which allows for transition between states. A general framing of the space of transition between states is the Hidden Markov model, in which every transition has a certain probability of occurrence. Real states of the world are unknown, they can only be approximated. We obtain data through sensors from the environment. The data is recorded in the form of vectors. Markov models are static: the agent is unable to model his actions, he cannot change the world he is in. An extension of this model are Markov Decision Processes. MDP generalizes Markov models by introducing additional possibilities: consequences of actions may not be know a priori, even if the consequences of actions are known, their value is unknown, the value of the action is difficult to predict, because the reward is often delayed. In this situation the model of best actions is not known, but it has to be discovered. The agent uses certain actions and analyses their results. The actions which bring reward are not known, but they have to be discovered by trial and error. Theories that apply the above method are called models of reinforcement learning. Models of reinforcement learning allow the agent to choose adequate decisions on the basis of exploration of the environment that changes dynamically. By examining the space of states that bring reward, the agent may learn from the history of his previous actions. The most interesting situation occurs when the space of states is only partially observable. Reinforcement learning works, because the agent can make local improvements in order to increase the reward.

A deep neural network (DNN) is an artificial neural network with multiple hidden layers between the input and output layers. Each successive layer uses the output from the previous layer as input. It is system of multiple layers of nonlinear processing units that learns of feature representations in each layer and form a hierarchy from low-level to high-level features. Deep learning networks can be applied to any problem for example in language, sound or image processing. Deep and recurrent neural networks are powerful models that achieve high performance on difficult pattern recognition problems in vision, and speech (Krizhevsky et al., 2012; Hinton et al., 2012; Dahl et al., 2012). Reinforcement Learning can be used to improve dialogue managers, e.g. for transitions between dialogue states (Rieser and Lemon, 2011)1, for non-goal-orientated dialogues (Li et al., 2016), for bot-bot dialogues and for inventing new languages by agents (Das et al., 2017). Photo2 using AI agents during performance.

Deep Learning is becoming more and more popular method. Commercial using of Deep Learning models is often associated with using data from massive data centers eg. Google, Facebook and it is difficult to verify if used model is really intelligent and can generalize knowledge or if it is only sophisticated complex automated system that uses the brute force method based on unlimited access to data generated by users. The application of deep learning in art looks uninteresting because constructed networks simulate only human behaviors; in this case, they use art history databases to generate objects that imitate artworks from the past.

Therefore, if we want to make the next shift (challenge) we have to put more emphasis on research concerning self-improvements of system eg. different types of algorithms associated with reinforcement learning and perform interesting fusions of reinforcement learning with deep learning. The goal is to incorporate the reinforcement learning process into deep learning for creating a system that will have an ability to learn and self-improve. Another way to do this is to study the methods connected with randomness and to integrate them into neural networks. Both of these approaches are perfectly complementary because there is no interesting self-improvement system without the clever use of random generators and vice versa.

Kingdom of Randomness

To obtain interesting results in music and art we need randomness. Randomness is important when you want the Neural Network from the same input to create different possibilities as an output, without generating the same output over and over again. Therefore it is different than in science, which is all about grouping and clustering. In art and music we don't want to endlessly obtain the same result. When you are composing sound or images, you don't want the neural network program to create the same sets of sounds and images; instead, you need creativity and variability. One of the solutions is to parametrize NN outputs with the use of probability distributions. A different way is to add noise directly to NN, instead of modeling the distribution parameters. Paradoxically, in such NNs, the more randomness during training, the better the results. Good random generators allow to avoid situations, when Neural Networks gets fixed in local minima. The importance of well-designed random initialization and momentum in deep learning was observed for example by Sutskever and Hinton (2016).

Random number generators are applied in many domains for instance in music programs, computer games etc. In numerous compilers random number generators are used as ordinary functions. The problem with these generators however is that they never produce random numbers, not even numbers that approximately appear to be random. Limits of our perception makes us consider sequences of numbers they produce as random, and maybe in the case of games of chance this ’randomness’ is sufficient, but in cryptography it is hopelessly not random and completely useless. No random number generator build with the use of computer (or abstract Turing machine) will generate random numbers. It is impossible to obtain a sequence of random numbers with the use of computer. Computers are entirely deterministic machines: we put something in, subsequently we subject it to entirely predictable operations and receive something “new”. If we put the same thing into the computer again (in different times) we will get exactly the same result. If we put the same input sequence into two different computers, we will also gain the same result. The number of states of a computer is finite.

The result is completely determined by input data and functions we use. Every random number generator build with the use of computer is, by definition periodic. Of course, such periodic generator is predictable. If it is predictable it can’t be random. A real random number generator requires a random input. A computer does not have random input. The only possibility when it comes to randomness in case of computer science techniques is the creation of pseudorandom number generators (PRNG). They can be useful in some applications if the period of the obtained sequence of numbers is long enough, which means that numbers will repeat but after a relatively long time. The beginning of this sequence should consists of numbers that remind of random numbers. Many pseudorandom generators have been built, which are periodic, but the potential periods have the length of 2256 bites. However even generators with very long periods create strange correlations between numbers. Unfortunately every deterministic generator will produce them if it will be used in a specific way, but there are ways of minimizing the number of correlations. Structure of PRNG consists in using a certain number and recurrence function. We start with a certain number (seed). Then we subject it to mathematical function. We obtain a number, which is again subjected to the same function. We repeat the procedure.

We obtain pseudorandom sequence of numbers. The process is constructed in such a way so that the numbers repeat after some time. How fast it will occur depends on the function used. At some point PRNG will produce the number from which we started. From this moment the numbers will repeat periodically. In weak generators the periods are short, in good ones very long, which means that the sequence of numbers may repeat after milliards of operations. A choice of good, incompressible seed has a crucial importance in cryptography and art. When the secret cypher key is generated in a pseudo-random manner, if we get the seed, we also get the key. In cryptography we need really random number sequences. This means sequences, which can’t be repeated: it is impossible to repeat a sequence of bits produced by such generator. This problem can be examined on the example of creating a key. Keys should be generated in such a way, so that the sequence used is really random. Otherwise, the adversary may get the copy of the key generator and break the given cryptographic system. Random number generator should be subjected to tests, which consist in attempts of compressing generated sequence. One of the popular methods of generating random numbers is obtaining bits from computer clock. It doesn’t however guarantee good quality randomness because computers use many mechanisms of clock synchronization. One shouldn’t use too many bits generated through this method because using the same procedure several times effects in the emergence of correlations, which are easy to find. Randomness obtained from measuring time when the keyboard remains idle is another popular method, which consists in measuring short time intervals between keyboard presses. However a fundamental flaw of this method is that the generated keys are usually very short. A good way of obtaining big number of random numbers is to use bits from the surrounding world, for example using atmospheric noise. This method requires specialized measuring devices that allow for measuring time between events.

Generators that use thermal noise (e.g. from semiconductor material), as well as those that use computer disk drive, and measure time necessary for reading set of data from the disk have been created. Air turbulences influence variations in the rate of disk rotation. There are also other methods of obtaining numbers from noise, which consist in measuring the position of the machine, mouse, screen behavior, components of currently displayed image, CPU loads, microphone signal etc. Fundamental flaw of these methods is that there may occur some repeatable correlations introduced by measuring devices. These devices are often synchronized in order to enable correct, repeatable activity of the computer. A very good method is using radioactive material and Geiger counter in which, during radioactive decay the time between following clicks is always different. Randomness from the quantum level obtained from radiation emitted during decay of radioactive material has an extremely good quality. One may obtain a big number of random bits from Geiger counter and use it as a key. We use a sequence of the length of for instance 256 bits as an input of one-way hash function. Hash function is a mathematical function that transforms a sequence into a sequence of a precise length. We use one-way function, which means that it is easy to count the shortened sequence on the basis of input sequence but it is impossible to do it the other way round. Quantum mechanics claims that in the real world, randomness occurs in a pure form. In quantum states there is fundamental randomness and it cannot be changed. If we interpret the formulas of quantum mechanics we can say, that elementary particles do not exist in any precise location. They exist in many locations simultaneously, with a certain probability of occurrence.

Photo 3 Quantum Enigma

Mind uploading

Mind Uploading (MU) is the process of copying the brain from natural substrate to artificial one. MU is carried out through scanning the structure of a brain and building an appropriate informatics model, which is true to the original and after launching it on an adequate hardware, behaves like the original brain. By emulation we understand a 1 to 1 model, in which all properties of the system have been retained. Emulation, copying and modifying of the brain will bring explosion of diversity. We are very good at creating maps of the human brain in different scales (we have increasingly better techniques of scanning brain structures in different scales), however our informatics brain models are supremely primitive. There are interesting consequence of building MU in the future. People will become more heterogenous, more varied in terms of physical and cognitive dimensions. Another consequence will be the change of value system: life will become less valuable. Life will become "cheap" because we will be able to emulate it. Why not participate in a risky game, when in a worst case, I can activate the backup copy of myself.

The belief "inimmortality" may change our behavior. Analogy: in computer game we value our life less than in real life, because we can reset it anytime. After the mind up- loading turning point, freezing, copying, slowing down and modifying of the brain will become acceptable. Of course there will be other consequences: ethical, political, economic, medical, religious and cultural. I highlighted value of life, because it has influence on other values. Other consequences of MU are connected with the notion of identity and consciousness. Can human consciousness be emulated and transferred? Does the transfer preserve consciousness? People are conscious, which means that they have conscious, subjective experiences. They lie at the center of our mental life and give our life meaning. If we loose consciousness, in a significant sense we cease to exist. If brain emulation is devoid of consciousness, then it probably doesn't exist at all, at the very least it is zombie existence. The problem is complex because our understanding of consciousness is very weak, we do not know how brain's activity makes consciousness possible.

References

Bates J. 1992. Virtual Reality, Art, and Entertainment. Presence Teleoperators Virtual Environ 1:133–138.

Dahl, G.E., Yu, D., Deng, L., and Acero, A. Context- dependent pretrained deep neural networks for large- vocabulary speech recognition. Audio, Speech, and Lan- guage Processing, IEEE Transactions on, 20(1): 30–42.

Das A., Kottur S., Moura J., Lee S. 2017. Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning. arXiv preprint arXiv:1703.06585.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N.,

Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., et al. Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Processing Magazine, 2012.

Hochreiter S., Schmidhuber J.1997. Long short-term memory. Neural Comput. Nov 15;9(8):1735-80.

Krizhevsky, A., Sutskever, I., and Hinton, G. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25, pp. 1106–1114, 2012.

Li J., Monroe W., Ritter A., Galley M., Gao J., Jurafsky D. 2016. Deep Reinforcement Learning for Dialogue Generation. arXiv preprint arXiv:1606.01541.

Maas H. L. van der, Verschure P. F. M. J., and P. C. M. Molenaar (1990), A Note on Chaotic Behavior in Simple Neural Networks, Neural Networks, Vol. 3, pp. 119–122.

Magerko B., Manzoul W., Ried M. 2009. An Empirical Study of Cognition and Theatrical Improvisation. In: Proc. Seventh ACM Conf. Creat. Cogn. ACM, pp 117–126.

Pribram K. 1991. Brain and Perception: Holonomy and Structure in Figural Processing, Lawrence Erlbaum Associates.

Riedl M., Bulitko V. 2013. Interactive Narrative: An Intelligent Systems Approach. AI Mag 34:67–77.

Rieser V., Lemon O. 2011. Reinforcement Learning for Adaptive

Dialogue Systems: A Data- driven Methodology for Dialogue Management and Natural Language Generation. Springer Science & Business Media.

Schechner R. 1998. Performance theory, Routledge New York.

Sutskever I., Martens J., Hinton G. E. 2011. Generating text with recurrent neural networks. In Proceedings of the 28th International Conference on Machine Learning (ICML-11). pages 1017– 1024.

Sutskever I., Martens J., Dahl G., Hinton G. 2016. On the importance of initialization and momentum in deep learning. Proceedings of the 30th International Conference on Machine Learning, PMLR 28(3): 1139-1147.

Wen T., Gasic M. 2015. Stochastic language generation in dialogue using recurrent neural networks. In Proceedings of SIG-dial for Computational Linguistics.