Machine Learning: what you are too afraid to ask

Author, Francesco Corea

A Data Science Foundation White Paper

March 2018

---

www.datascience.foundation

Data Science Foundation
Data Science Foundation, Atlantic Business Centre, Atlantic Street, Altrincham, WA14 5NQ
Tel: 0161 926 3641 Email: admin@datascience.foundation Web: www.datascience.foundation
Registered in England and Wales 4th June 2015, Registered Number 9624670

What you are too afraid to ask about Artificial Intelligence

(Part I): Machine Learning

AI is moving at a stellar speed and is probably one of most complex and present sciences. The complexity here is not meant as a level of difficulty in understanding and innovating (although of course, this is quite high), but as the degree of interrelation with other fields apparently disconnected

There are basically two schools of thought on how an AI should be properly built: the Connectionists start from the assumption that we should draw inspiration from the neural networks of the human brain, while the Symbolists prefer to move from banks of knowledge and fixed rules on how the world works. Given these two pillars, they think it is possible to build a system capable of reasoning and interpreting.

In addition, a strong dichotomy is naturally taking shape in terms of problem-solving strategy: you can solve a problem through a simpler algorithm, which though it increases its accuracy in time (iteration approach), or you can pide the problem into smaller and smaller blocks (parallel sequential decomposition approach).

Up to date, there is not a clear answer on what approach or school of thoughts works the best, and thus I find appropriate to briefly discuss major advancements in both pure machine learning techniques (Part I) and neuroscience (Part II) with an agnostic lens.

Machine learning

Machine learning techniques can be roughly pided into supervised methods and unsupervised methods, with the main difference of whether the data are labelled (supervised learning) or not (unsupervised). A third class can be introduced when we talk about AI: reinforcement learning (RL). RL is a learning method for machines based on the simple idea of reward feedback: the machine indeed acts in a specific set of circumstances with the goal of maximizing the potential future (cumulative) reward. In other words, it is a trial-and-error intermediate method between supervised and unsupervised learning: the data labels are indeed assigned only after the action, and not for every training example (i.e., they are sparse and time-delayed). RL usually comes with two major problems, namely the credit assignment problem and the explore-exploit dilemma — plus a series of technical issues such as the curse of dimensionality, non-stationary environments, or partial observability of the problem. The former one concerns the fact that rewards are, by definition, delayed, and you might need a series of specific actions in order to achieve your goal. The problem is then to identify which of the preceding action was actually responsible for the final output (and to get the reward then), and if so to what degree. The latter problem is instead an optimal searching problem: the software has to map the environment as accurately as possible in order to figure out its reward structure. There is an optimal stop problem — a sort of satisficing indeed: to what extent the agent should keep exploring the space to look for better strategies, or start exploiting the ones

it already knows (and knows that work)?

In addition to the present classification, machine learning algorithms can be classified based on the output they produce: classification algorithms; regressions; clustering methods; density estimation; and dimensionality reduction methods.

The new AI wave encouraged the development of innovative ground-breaking techniques, as well as it brought back to the top a quite old concept, i.e., the use of artificial neural networks (ANNs).

Artificial Neural Networks are a biologically-inspired approach that allows software to learn from observational data — in this sense sometimes is said they mimic the human brain. The first ANN named Threshold Logic Unit (TLU) was introduced in the Forties by McCulloch and Pitts (1943), but only forty years later Rumelhart et al. (1986) pushed the field forward designing the back-propagation training algorithm for feed-forward multi-layer perceptrons (MLPs).

The standard architecture for any ANNS is having a series of nodes arranged in an input layer, an output layer, and a variable number of hidden layers (that characterize the depth of the network). The inputs from each layer are multiplied by a certain connection weight and summed up, to be compared to a threshold level. The signal obtained through the summation is passed into a transfer function, to produce an output signal that is, in turn, passed as input into the following layer. The learning happens in fact in the multiple iterations of this process, and it is quantitatively computed by choosing the weighting factors that minimize the input-output mapping error given a certain training dataset.

ANNs do not require any prior knowledge to be implemented, but on the other side, they can still be fooled because of it. They are often also called Deep Learning (DL), especially for the case in which there are many layers that perform computational tasks. There exist many types of ANNs up to date, but the most known ones are Recurrent Neural Networks (RNNs); Convolutional Neural Networks (CNNs); and Biological Neural Networks (BNNs).

RNNs use sequential information to make accurate prediction. In traditional ANNs, all the inputs are independent one from the other. RNNs perform instead a certain task for every element of the sequence, keeping a sort of memory of the previous computations. CNNs try instead to mirror the structure of the mammalian visual cortex and they have every layer working as detection filters for detecting specific patterns in the original data (and this is why they are really suitable for object recognition). Finally, BNNs are more a sub-field of ANNs rather than a specific application. The best example of this class is in our opinion the Hierarchical Temporal Memory (HTM) model developed by Hawkins and George of Numenta, Inc, which is a technology that captures both the structural and algorithmic properties of the neocortex.

In spite of the big hype around deep learning possibilities, all that glitters is not gold. DL is for sure a great step ahead toward the creation of an AGI, but it also presents limitations. The greatest one is the exceptional amount of data required to work properly, which represents the major barrier to a wider cross-sectional application. DL is also not easy to debug, and usually, problems are solved by feeding more and more data into the network, which creates a tighter big-data-dependency. Furthermore, DL is quite useful to bring to light hidden connections and correlations but is not informative at all regarding the causation (the why of things).

The data need imposes a considerable amount of time to train a network. In order to reduce this time, networks are often trained in parallel, either partitioning the model across different machines on different

GPU cards (model parallelism) or reading different (random) buckets of data through the same model run on different machines to tune the parameters (data parallelism).

Because of the limitations just mentioned, a series of other tools have been developed over the years. **Particle Swarm Optimization (PSO)** is a computational method that iteratively improves candidate solution to optimize a certain problem (Kennedy and Eberhart, 1995). The initial population of candidates (namely *dubbed particles*) is moved around in the search-space, and it has single particles that optimize their own position both locally and with respect to the entire search-space — creating then an optimized swarm. **Agent-based Computational Economics (ACE)** is an additional tool that lets agents interacting according to pre-specified rules into simulated environments (Arthur, 1994).

Starting from some initial condition imposed by the modeler, the dynamic systems evolves over time as interactions between agents occur (and as they learn from previous interactions).

Evolutionary Algorithms (EA) are instead a broad class of techniques that find solutions to optimization problems through concepts borrowed from natural evolution, i.e., selection, mutations, inheritance, and crossover. An example of EA is the **Genetic Algorithm (GA)**, which is an adaptive search heuristic that attempts to mimic the natural selection process (Holland, 1975). It is an evolutionary computing search optimization method that starts from a base population of candidate solutions and makes them evolving according to the “survival of the fittest” principle. **Genetic Programming (GP)** is an extension of GA (Koza, 1992) because it basically applies a GA to a population of computer programs. It creates the chromosomes (i.e., the initial population of programs) made by a predefined set of functions and a set of terminals, and it randomly combines them into a tree-structure. In this context, the previous terminology acquires a slightly different connotation: reproduction means copying another computer model from existing population; cross-over means randomly recombining chosen parts of two computer programs, and mutation is a random replacement of chosen functional or terminal node. **Evolutionary Polynomial Regressions (EPRs)** are instead hybrid regressions that use GA to select the exponents of the polynomial, and a numerical regression (i.e., least square regression) to compute the actual coefficients (Giustolini and Savic, 2006). A final interesting model is called **Evolutionary Intelligence (EI)** or **Evolutionary Computation (EC)**, and it has been recently developed by Sentient Technologies, LLC. It begins randomly generating trillions of candidate solutions (called genes) that by definition would probably perform poorly. They are then tested against training data, and a fitness score allowed the software to rank the best solutions (and eliminates the worst). Parts of the emerging candidates are then used to reassemble new populations, and the process restarts until a convergence is achieved.

To conclude this section, two additional approaches are worthy to be acknowledged. First, **Generative Models (GMs)** have been initially proposed by Shannon (1948), but recently brought back to the top by OpenAI, a non-profit AI research institute based in San Francisco (Salimans et al., 2016; Chen et al., 2016). This class of models is intuitively defined as those models we can randomly generate data for, assumed some hidden parameters. Once the data are feed, the system specifies a joint probability distribution and label sequences of data.

Second, Cao and Yang (2015) proposed a new method that converts the learning algorithm into a summation form, instead of proceeding directly from each training data point. It is called **Machine Unlearning (MU)**, and it allows the systems to “forget” unwanted data. They actually introduce an intermediate layer of summation between the algorithm and the training data points, such that they will not depend on each other anymore, but only on the summations themselves.

In this way, they learning process is much faster, and it can be updated incrementally without training again the model from scratch — which is quite time-intensive and costly. Hence, if some data and its lineage want to be eliminated, the system does not need to recompute the entire lineage anymore — a term coined by the two authors to indicate the entire data propagation network — but it can simply recompute a small number of summations.

References

Arthur, B. W. (1994). “Inductive Reasoning and Bounded Rationality”. American Economic Review, 84(2): 406–411.

Cao, Y., Yang, J. (2015). “Towards Making Systems Forget with Machine Unlearning”. 2015 IEEE Symposium on Security and Privacy: 463–480.

Chen, X., Duan, X., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P. (2016). “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets”. arXiv:1606.03657.

Giustolisi, O., Savic, D.A. (2006). “A symbolic data-driven technique based on evolutionary polynomial regression”. Journal of Hydroinformatics, 8(3): 207–222.

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. MIT Press.

Kennedy, J., Eberhart, R. (1995). “Particle Swarm Optimization”. Proceedings of IEEE International Conference on Neural Networks: 1942–1948.

Koza, J.R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press.

McCulloch, W. S., Pitts, W. (1943). “A Logical Calculus of the Ideas Immanent in Nervous Activity”. Bulletin of Mathematical Biophysics, 5: 115–133.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986). “Learning representations by backpropagating errors”. Nature, 323: 533–536.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X. (2016). “Improved Techniques for Training GANs”. arXiv:1606.03498.

Shannon, C.E. (1948). “A Mathematical Theory of Communication”. Bell System Technical Journal, 27: 379–423, 623–656.

This is an excerpt from my book “Artificial Intelligence and Exponential Technologies: business models evolution and new investment opportunities”, edited by Springer (2017).

About the Data Science Foundation

The Data Science Foundation is a professional body representing the interests of the Data Science Industry. Its membership consists of suppliers who offer a range of big data analytical and technical services and companies and individuals with an interest in the commercial advantages that can be gained from big data. The organisation aims to raise the profile of this developing industry, to educate people about the benefits of knowledge based decision making and to encourage firms to start using big data techniques.

Contact Data Science Foundation

Email:admin@datascience.foundation
Telephone: 0161 926 3641
Atlantic Business Centre
Atlantic Street
Altrincham
WA14 5NQ
web: www.datascience.foundation