Raising AI: Tutoring Matters

Jordi Bieger[1], Kristinn R. Thórisson[12], and Deon Garrett[12⋆]

[1] Center for Analysis and Design of Intelligent Agents / School of Computer Science, Reykjavik University, Menntavegur 1, 101 Reykjavik, Iceland
[2] Icelandic Institute for Intelligent Machines, Uranus, Menntavegur 1, 101 Reykjavik
{jordi13, thorisson, deong}@ru.is

Abstract. Humans and other animals are often touted as examples of systems that possess general intelligence. However, rarely if ever do they achieve high levels of intelligence and autonomy on their own: they are raised by parents and caregivers in a society with peers and seniors, who serve as teachers and examples. Current methods for developing artificial learning systems typically do not account for this. This paper gives a taxonomy of the main methods for raising / educating naturally intelligent systems and provides examples for how these might be applied to artificial systems. The methods are *heuristic rewarding*, *decomposition*, *simplification*, *situation selection*, *teleoperation*, *demonstration*, *coaching*, *explanation*, and *cooperation*. We argue that such tutoring methods that provide assistance in the learning process can be expected to have great benefits when properly applied to certain kinds of artificial systems.

Keywords: artificial general intelligence, learning, tutoring, raising

1 Introduction

Humans grow and learn under the supervision of parents or other caregivers, in a society with peers and seniors, who serve as teachers and examples. We don't confine our children to libraries and hope that they'll figure out how to read. Yet it seems that often we expect something similar from our artificial systems: In a typical machine learning project, first a well-defined target task is selected that we would like our artificial learner to accomplish automatically. Humans identify and extract the most important features for their learning system, including variables and their target operating ranges, reward functions, etc. Next, the system is released into (a simulation of) the target environment to attempt to accomplish the target task, possibly with the help of randomly selected, human annotated examples. While this paradigm has lead to many useful applications, it can only be applied to tasks and environments that are well defined at the system's design time, and they are confined to only such tasks, and to address any of them requires starting a new cycle of manual system creation.

The goal in AGI is how to build systems that are not merely good at a single task that is known beforehand, but systems that can perform autonomously on

⋆ We want to thank Elsa Eiríksdóttir for educating us about education psychology.

a very wide range of tasks, in a wide range of environments, many of which the system may never have encountered. Our working definition of intelligence is *"the ability to autonomously achieve complex goals in a wide range of complex environments with constrained resources"*. Intelligent systems should be able to identify and extract the most important features for their learning process automatically. So far, this goal has remained unattainable. We often look to humans and other intelligent animals as proof that such general intelligence is indeed possible, but none of them start their lives with their potential for intelligence and autonomy fully realized. Unlike most current artificial systems, when we were babies we were not placed directly into the particular environment where we are later expected to perform. We were raised.

We can distinguish between a system's *potential intelligence* and its *realized intelligence*. Realized intelligence is the amount of intelligence, by some (unspecified) measure, that the system exhibits at any particular moment in time. Potential intelligence is the maximum possible realized intelligence that the system might achieve in its lifetime. Human babies possess a potential intelligence that will eventually allow them to be as intelligent as an adult human (the pinnacle of currently existing general intelligence), but their realized intelligence is initially fairly low. They will fail at some of the most rudimentary tasks that they will be capable of in the future; when left to their own devices they soon die. How much of their potential is realized depends in large part by how they are raised. Some aspiring AGI systems (cf. AERA [16], NARS [26] and OpenCog [8]) are similarly envisioned to start from humble beginnings and acquire the ability to behave well through interaction with complex environments. It has even been suggested to use a human-like preschool environment to evaluate the intelligence and developmental stage of proto-AGI [9].

The goal of *raising* a system is to help it realize as much of its potential intelligence as possible as quickly as possible. Given a certain base-level of potential intelligence, a raised system should reach a higher level of realized intelligence than one that is completely left to its own devices. Or, considered another way: To reach a certain target level of intelligence, e.g. "human level", a system may require less potential intelligence if it is raised well.

There are many aspects to raising an intelligent system. In this paper we focus on the aspect of providing assistance in the learning of new tasks, which we'll refer to as *tutoring*. We discuss ways in which the upbringing of (human) animals has been or could be translated to the artificial domain towards this end. We aim to provide a taxonomy of tutoring methods for AI, articulate some of the associated requirements this puts on the system's capabilities and discuss the implications and future work.

2 Tutoring Techniques

In this section we describe a number of tutoring techniques that can help raise intelligent systems: heuristic rewarding, decomposition, simplification, situation selection, teleoperation, demonstration, coaching, explanation and cooperation.

We give anthropocentric illustrations and review prior work in machine learning and AI. We additionally provide examples of how these methods might be implemented to solve three simple reinforcement learning (RL) tasks using a tabular Q-learning algorithm [22]: 1) a 2D navigation task, 2) a small sliding puzzle and 3) a compound task where the puzzle and navigation task must be solved serially in a combined state space[1]. It should be stressed however that this is not an experimental study, and that the results in Fig. 1 do not imply the general superiority of one method over another or that they are always beneficial. These examples are merely meant to provide simple and concrete illustrations of the presented methods and to show that there are at least some situations in which they can help.

2.1 Heuristic Rewards

The fact that behavior can be shaped through the use of rewards and punishments is a core tenet of the field of behaviorism and the underlying mechanism in associative and reinforcement learning [19]. Reinforcement learning is furthermore a useful paradigm for AGI, because it doesn't require a dedicated supervisor to constantly provide correct actions. However, acquisition of rewards or avoidance of punishment in many tasks requires a level of proficiency that a beginner cannot be expected to possess.

Tasks that require a certain level of skill for the learner to receive varied feedback can be difficult to learn, because it can make the "goodness" of different actions virtually indistinguishable. A tutor can mitigate this problem by adding more feedback through real-time interaction with the learner [25], by e.g. rewarding good moves with encouraging words or a monetary reward, or by teaching heuristics for actions or situations to estimate their "worth": For instance, capturing an opponent's piece in chess could be awarded with a number of imaginary points. This is the central idea behind *gamification* [7] which has been successful in helping people to learn by supplementing long term benefits and understanding with immediate feedback and rewards. Heuristics like these are also often used in machine learning for game playing, where the final state and the associated win, loss or draw is the only thing that really matters.

In our own RL example, we have implemented the heuristic rewards approach by giving higher rewards to good actions than to bad ones. Fig. 1 clearly shows that this can be hugely beneficial to learning speed. However, care should be taken that pursuit of these artificial heuristics is not more beneficial than pursuit of the actual goal. For instance, if too much importance is attached to capturing pieces in chess, a player may forgo a guaranteed winning move because prolonging the game may allow the capture of more pieces.

2.2 Decomposition

Part-task training is a well-known strategy in educational psychology that involves the decomposition of the whole task into smaller parts that can be learned

[1] Please refer to our technical report for more details [5].

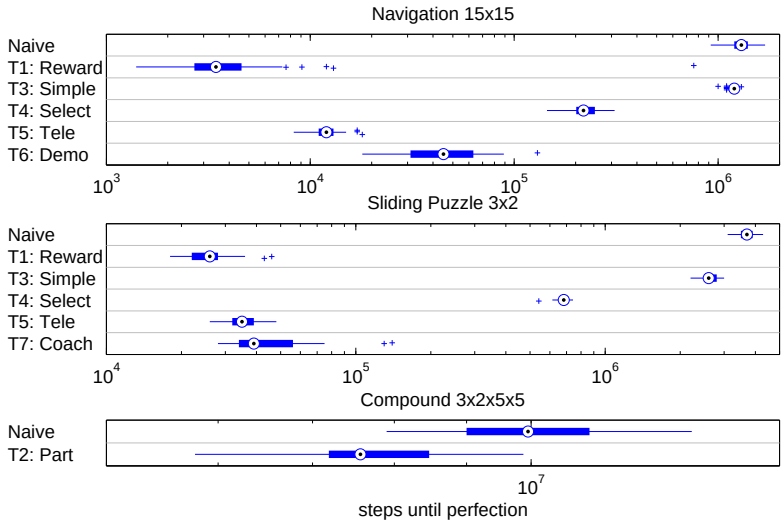

Fig. 1. Positive results obtained using tutoring techniques to decrease the number of training steps to reach optimal behavior, plotted on a logarithmic scale. All depicted improvements were statistically significant ($p < 0.0001$). Used parameters: {learn rate: 0.05, discount factor: 0.75, ϵ: 0.25, reward: 100, initial Q-values: 0}. For more details please refer to our technical report [5]. For the heuristic rewards, bad actions were penalized with -1. Simplification worked by starting with easy starting positions and for the navigation task a small grid, and gradually increasing difficulty and grid size. Situation selection was performed by focusing on problem areas one by one.

separately [24]. This approach is often highly beneficial when the whole task is complex. By decreasing the intrinsic cognitive load of the task, more of the learner's cognitive resources can be used for learning [27]. On the other hand, some of the load is transferred to the tutor, who has to decompose the task in such a way that the right thing is still learned.

Both decomposition and simplification (see Sec. 2.3) have the tutor design a curriculum of easier tasks. One example of existing work of curriculum learning involves the layer-by-layer training of a deep neural network, where each layer's task is to extract higher level features from the one before it [4].

One prerequisite for the system's learning algorithm is that it does not suffer from catastrophic interference: the phenomenon where an algorithm forgets all about earlier learned tasks when a new one is learned. Furthermore, a strong transfer learning ability can be highly beneficial in this setting, because it allows the system to extract lessons for multiple tasks at once. Incidentally, transfer learning is greatly aided by variability of training tasks [11] and multitask learning [6], which could be considered the converse of part-task learning.

We have implemented this divide and conquer tutoring method for learning the compound task by letting the system train on the navigation and puzzle parts

separately. Fig. 1 shows that this results in faster learning than just training on the compound task from the beginning. However, an important consideration when using this paradigm is how the component tasks reintegrate to form the compound target task as context can often be important. With Q-learning, problems can occur when the expected reward of the second part is very different from the chosen final reward for the first part because this will invalidate the Q-values learned for it.

2.3 Simplification

When we teach something new, we generally start with easy variants of the envisioned target task. Task simplification and the subsequent gradual increase in difficulty until the training task matches the target task is another example of learning on a curriculum. A similar paradigm called *shaping* – introduced by B.F. Skinner [19] – is often used in animal training. This technique has recently been utilized in reinforcement learning [12][20], unsupervised dependency parsing [21] and deep learning [4][3][10][13].

Simplification is intimately related to decomposition because the component tasks are simpler than the target task. The difference can intuitively be defined as follows: a system that can generalize very well could exhibit decent performance on the target task after learning a simplified version, but not after learning a single part-task in the decomposition approach.

There are a number of different ways in which we can implement this approach in our reinforcement learning examples. For the navigation task, we can decrease the grid size and for either task we can set the start state to be close to the goal state. Whenever the system has fully learned to achieve the task at the current level, we increase the difficulty until we reach the target task. Positive results are shown in Fig. 1. When errors are introduced at a lower difficulty level, it can be beneficial to decrease the difficulty again.

2.4 Situation Selection

By selecting the situations that a pupil encounters during learning, a tutor can greatly facilitate the experience. Dwelling in situations that are either far too simple or difficult can be a massive waste of time. In the previous section, we saw how choosing easy starting positions could facilitate learning. But at some point in our training, we often move on from situations that are relatively easy to ones that we have a disproportionate amount of trouble with. Essentially, we tend to focus on areas that are the most important to our learning. Identifying problem areas and getting better at them is often the best way to increase our overall success in a given task.

A related concept in machine learning is *boosting* [17] where multiple classifiers are trained and training examples that are misclassified by earlier classifiers are weighted more. Furthermore, for any multi-component system, it is natural for developers to focus on strengthening the "weak link" in the chain. However,

these applications are not considered as raising approaches, because they are not applied to the finalized system.

The past decade has also seen the rise of *Big Data* science and the idea that the quality and quantity of data might be of equal or greater importance than the capabilities of the learner. The core tenet in the field *active machine learning* is to find ways for a learner to actively select data or situations from which it is believed that the most can be learned [18]. In that sense, situation selection could be viewed as *active teaching*.

In our RL examples we can easily identify states in which the system's behavior is erroneous. We can either pick a possibly different trouble state for each epoch, or we can go through these states one by one and only move on when it is no longer problematic. Fig. 1 shows that focusing on problem areas can indeed expedite training. We have also found that it can help to stop the current epoch and start a new one when the system has wandered into an unproductive situation. Generally speaking it can save time to avoid situations that are already mastered and ones that cannot be learned yet.

2.5 Teleoperation

Sometimes it can be difficult to accurately describe an action, because it must simply be experienced by the learner. In these cases it can be educational if the tutor can temporarily take control of the learner's actions through *teleoperation* [2]. This can be seen when a golf or tennis instructor stands behind the student and moves their arms for them in the correct way, and similar approaches have been taken in robotics [15]. Another example might involve a teacher making moves for his student in a chess game. In one recent project an artificial tutor learned when to make moves for his student in several video games [23].

Selecting an action for the learner must occur at the right level of abstraction, where the tutor can actually affect the learner directly. For instance, a tutor cannot make the student's brain take the required steps for it to consider a certain chess move good, but he can make the move for the student. Forcing a student's hand can also be a method for setting up situations that can be educational, as we have seen in Sections 2.2, 2.3 and 2.4 where the goal was to start from states that are easy, problematic or require a certain subskill.

Fig. 1 shows that forcing our Q-learner to take the right actions can greatly reduce training times. This should come as no surprise, because unlike humans in tennis the Q-learner has no trouble reproducing the steps that it was forced to perform during training. However, it is worth noting that this only works when the "correct" Q-value for the optimal state-action pair is larger than the initial Q-values for the other actions in that state.

2.6 Demonstration

Human and many other animal infants are innately predisposed towards imitating those around them. Teaching by example or *demonstration* then is a very powerful educational technique. When students have the capability to map the

actions of others onto their own, observing someone else perform a task can tell them how to do it themselves. Observing masters of the field is a common teaching technique in many disciplines. Additionally, sometimes "do as I do" is simply the easiest way to defne a task. *Apprenticeship learning* – where the learner deduces its goal via *inverse reinforcement learning* of the reward function in a tutor's demonstration – has been used to great success [1][16].

The machine learning subfield of *imitation learning* studies how an artificial system might learn by imitating a tutor [2]. The emphasis here is usually on how to design the learner, and it is often assumed that the tutor will be a human who simply performs the task that we want the system to learn. However, this need not be the case: Although artificial general intelligence is notoriously difficult to develop, many AI systems show good performance in specialized domains and could conceivably function as teachers-by-example for a more generally intelligent system. Furthermore, it is not a given that an optimal demonstration of the target action(s) is necessarily the optimal teaching signal.

One of the major difficulties in imitation learning is the recognition of the tutor's actions and the mapping to the learner's own capabilities. We cannot do justice to this complexity in our simple RL examples. Learning by example in our puzzle task is difficult, because once a tutor moves a tile, the board is changed and the student cannot simply mimic the move. Of course, it could still learn from this in the same way that it would with teleoperation (see Sec. 2.5) or as an apprentice. We've augmented the navigation task with two additional dimensions that signify the system's displacement from a tutor. At every step, the tutor first moves to the location of the student (if necessary) and then makes a step towards the goal. After training with the tutor, we set these two input dimensions to zero for further training alone and evaluation.

Learning by imitation requires generalization ability or redundancy learning. We implemented a system of dual Q-learners that focus on different parts of the state space and vote on the action to take [5] (see Fig. 1). Since the target task and learning to imitate are largely independent, this approach can be combined very well with decomposition and simplification methods. Rewarding successful imitation can also be beneficial.

2.7 Coaching

A lot of our education simply consists of telling students what to do in what situation. Real-time instruction like this is especially common in coaching during some sporting events, where the coach may shout out the actions that he wants his player to perform. We can also see this approach in action when a teacher tries to walk his student to a math problem step by step.

Learning by direct instruction is essentially the approach that is taken in supervised learning. The learning algorithm is presented with both the situation and the right response to it. In real life, instruction is complicated by the fact that complex language must be mapped unto actions that can actually be performed.

In our RL example, coaching was implemented as an extra dimension on the input space that will tell the system what action to take, although the system

still needs to learn the mapping from instructions to actions (see Fig. 1). This implementation is very similar to the one for demonstration, but this is mostly an artifact of the simplicity of our examples. Imitation learning can be done without the teacher's active involvement and requires that the system has capabilities to detect other entities, their actions and possibly their goals. Learning by instruction on the other hand does require the active involvement of a coach as well as some communication protocol (i.e. language).

2.8 Explanation

When we want our students to learn something new we often explain to them what to do and how they should approach the situation. This type of explanation can take the form of imperative "programming" (e.g. "do not move towards a nearby enemy in a video game") [14], or it could involve using analogies to connect students' existing knowledge to new concepts.

Explanation is an extremely complex tutoring method and requires a wide array of features on the part of the learner. The learner must meet the same requirements as for learning by direct instruction, but increased complexity of offline explanations compared to simple direct action instructions requires more sophisticated language processing faculties. Furthermore, the learner must have good memory and retrieval capabilities, and be able to map the explanations onto situations that he must be able to detect and actions that he can execute. Finally, prior knowledge must exist to connect the new knowledge to.

Explanations give the system prior knowledge before starting to actually practice the task, so we could model it by adjusting the initial Q-values in our RL example. If the explanation specifies what to do in what situation, we could simply encode that in the Q-table. When using analogy to existing knowledge in the form of another Q-table for a different task, we could initialize the new table with the values from the existing one. We have not done this, since simply initializing the table to the (almost) correct values will show nothing interesting, and ignores the fact that the real complexity is in recognizing how to map an explanation onto knowledge that the system can use.

2.9 Cooperation

Finally, one way to teach a task is to simply lend a helping hand and do the task together initially. This is essentially a combination of making the task easier, possibly letting the student focus on a single component, and teaching by example. This type of apprenticeship can be observed in many jobs and is also employed by predators who will let their young join them on simple hunts.

We often lend our machine learning systems a helping hand by providing them with correct answers or preprocessing their data, but we don't usually intend for the system to eventually learn to take over these tasks itself.

We have not implemented a new example to showcase this approach, because it is so broad and already partially covered by our examples for task simplification and teaching by example.

3 Discussion

Humans and other intelligent animals use a wide variety of educational and learning techniques to attain their eventual level of adult intelligence. The usefulness of these techniques often increases with the complexity of the attempted task, which makes them particularly relevant in the context of AGI. We have discussed how these techniques might be applied in an artificial setting, and shown that (even) in our simple example RL setting they can sometimes be beneficial. On the other hand design of curricula and heuristics, as well as identification of problem areas, coaching signals and correct actions are difficult problems. Modifying the task or environment brings the risk of teaching the wrong thing. Sometimes a naive approach simply works better.

Furthermore, these approaches require a tutor whose time may be more valuable than the learner's. Human tutors may impose an unwanted speed limit on interactions, if such constraints are not already applied by the environment or by the learner himself. No human can match the thousands of steps per second that a virtual agent in a virtual environment might be able to make. In such a situation, the reduction in the number of learning steps required to learn a task may be offset by the fact that training in this way takes more clock time.

However, in some cases we may be able to develop specialized, narrow AI systems to use for teaching. In a way, this might fulfill the dream of those who see narrow AI applications as a step towards highly intelligent machines: just add a system with a high potential for general intelligence and mix. Eventually, AGIs may of course be raised by other AGIs.

In conclusion, based on our preliminary studies, raising or tutoring intelligent systems can be worthwhile when the goal is for them to perform complex tasks in complex environments, provided that the knowledge for task accomplishment and teaching is available. The concept fits AGI as a glove, provided that future research can tell us how to teach these advanced systems. In our own future work, we intend to implement these methods for more complex tasks in an aspiring AGI system, investigate what tasks and skills such a system should learn to advance to the required cognitive stages on the road to adult-level intelligence, and how we can teach them.

References

1. P. Abbeel, A. Coates, and A. Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608âĂŞ1639, 2010.
2. B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.
3. Y. Bengio. Evolving culture vs local minima. *preprint arXiv:1203.2990*, 2012.
4. Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of ICML 26*, pages 41–48, 2009.
5. J. Bieger, K. R. Thórisson, and D. Garrett. Raising AI: Towards a taxonomy of tutoring methods. Technical Report RUTR-SCS13007, CADIA & SCS, Reykjavik University, April 2014.

6. R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.
7. S. Deterding, D. Dixon, R. Khaled, and L. Nacke. From game design elements to gamefulness: defining gamification. In *Proceedings of the 15th International Academic MindTrek Conference*, pages 9–15. ACM, 2011.
8. B. Goertzel. OpenCogPrime: a cognitive synergy based architecture for artificial general intelligence. In *Cognitive Informatics, 2009. ICCI'09. 8th IEEE International Conference on*, pages 60–68, 2009.
9. B. Goertzel and S. V. Bugaj. AGI preschool: a framework for evaluating early-stage human-like AGIs. In *Proceedings of AGI-09*, pages 31–36, 2009.
10. C. Guelcehre and Y. Bengio. Knowledge matters: Importance of prior information for optimization. *arXiv:1301.4083 [cs, stat]*, January 2013.
11. C. L. Holladay and M. A. Quinones. Practice variability and transfer of training: the role of self-efficacy generality. *Journal of applied psychology*, 88(6):1094, 2003.
12. A. Laud and G. DeJong. The influence of reward on the speed of reinforcement learning: An analysis of shaping. In *ICML*, pages 440–447, 2003.
13. J. Louradour and C. Kermorvant. Curriculum learning for handwritten text line recognition. *preprint arXiv:1312.1737*, 2013.
14. R. Maclin and J. W. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22(1-3):251–281, 1996.
15. K. Muelling, J. Kober, and J. Peters. Learning table tennis with a mixture of motor primitives. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 411–416. IEEE, 2010.
16. E. Nivel, K. R. Thórisson, B. R. Steunebrink, H. Dindo, G. Pezzulo, M. Rodriguez, C. Hernandez, D. Ognibene, J. Schmidhuber, and R. Sanz. Bounded recursive self-improvement. *preprint arXiv:1312.6764*, 2013.
17. R. E. Schapire. The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*, pages 149–171. Springer, 2003.
18. B. Settles. Active learning literature survey. Tech 1648, Madison, Wisconsin, 2010.
19. B. F. Skinner. The behavior of organisms: An experimental analysis., 1938.
20. M. Snel and S. Whiteson. Multi-task reinforcement learning: shaping and feature selection. In *Recent Advances in Reinforcement Learning*, pages 237–248. 2012.
21. V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. From baby steps to leapfrog: how less is more in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the NAACL*, pages 751–759, 2010.
22. R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 116. Cambridge Univ Press, 1998.
23. M. E. Taylor, N. Carboni, A. Fachantidis, I. Vlahavas, and L. Torrey. Reinforcement learning agents providing advice in complex video games. *Connection Science*, 26(1):45–63, January 2014.
24. R. C. Teague, S. S. Gittelman, and O. C. Park. A review of the literature on part-task and whole-task training and context dependency. DTIC, 1994.
25. A. Thomaz, G. Hoffman, and C. Breazeal. Real-time interactive reinforcement learning for robots. In *AAAI 2005 workshop on human comprehensible machine learning*, 2005.
26. P. Wang. *Non-Axiomatic Reasoning System: Exploring the Essence of Intelligence*. PhD thesis, Citeseer, 1995.
27. C. D. Wickens, S. Hutchins, T. Carolan, and J. Cumming. Effectiveness of part-task training and increasing-difficulty training strategies a meta-analysis approach. *Human Factors*, 55(2):461–470, 2013.