LEARNING TO COMPOSE

VENKATESH MEDABALIMI

One of the key features that facilitates learning a seemingly complex new task in humans is their ability to recognize the task as being composed out of simpler tasks, often ones that they might be already familiar with. It appears that this knack of identifying *known* constituent tasks and visualizing how they can be put together to perform the new task greatly speeds up learning. Recognizing familiar parts and how they seem to have been put together plays a crucial role in our appreciation and understanding of activities and things we run into in our everyday life. Identifying the folds towards an origami, recognizing geometric forms that come to shape a building[1], planning travel from point A to point B in a city you are only getting to know the locales of, even learning to cook a new dish you just happen to take a liking to, learning to hit a stroke in tennis or move according to a choreographed dance move or deconstructing music so as to play it yourself- inspite of their differences all involve recognizing constituent parts(the folds, shapes, blocks, routes connecting junctions, recipes within, body movements and notes[1] respectively) and figuring out how to compose them to learn to perform the task or create the desired object. May be on a more interesting note, any form of abstract reasoning or envisioning why a certain theorem or statement has to hold involves being able to see parts of it and how they can cause or interact to make the claim hold. It appears, greater our ability to deconstruct a new concept interms of the ones we know- the easier it is for us to learn it.

In concurrence with above observations we recognize following characteristics as necessary aspects of human intelligence. The ability to

- learn new concepts and tasks.
- compose efficiently what one has learnt already to reason or create something of greater value thus exploiting the expressive power of a finite skill set.
- actively seek what to learn and recognize useful subgoals towards building cognitive plans for achieving a goal.

One can argue intelligence manifests as proclivity for concise yet accurate composed explanations and executions in understanding and interacting with the environment. Arguably, this also resonates with one of the principles of neural design which is that new synaptic connections and neurons come at a material expense [2].

The advent of deep-learning has provided us with a great means of learning function approximators wherever we have a good amount of data and some clever means to take advantage of symmetries

Date: May 19, 2019.

[1]or how individual blocks have been put together for a Lego work.

and invariances within the tasks, to the point that machines are now better[2] than us at some of them. Lets think of these as individual functions that we know or have already perfected under supervision. The ability to learn composed tasks over such individual tasks can be very useful. We wish to abstract out this problem in the following manner so as to capture the main challenges of recognizing and learning compositions in the context of a machine learning to perform a new task.

Assume a machine $\mathcal{M}$(henceforth referred to as *we*) has knowledge of a collection of functions $\mathcal{A}$, these are synonymous with tasks that we have mastery over and can bring to recall or reproduce them at will. Suppose $\mathcal{M}$ is presented with examples of input output instances $(x, y = h(x))$ of a composed function h, constructed from constituents of $\mathcal{A}$. Some questions of interest that we seek to explore are the following. When presented with instances (x, y) of h, and we seek to learn h,

- Can we recognize underlying functions involved and how they have been put together, the structure of the composition h ? For example h could be $f \circ g \circ q$ for some $f, g, q \in \mathcal{A}$
- Can we learn h in a much fewer number of samples as opposed to the case when we have not seen the underlying functions before ? The innate bias we as a learner can exploit here comes from the collection of functions we already happen to know and also knowledge of the underlying structure. It is easy to give a counting argument for why this is indeed the case in the PAC-learning framework. If the composition structure is fixed and has s nodes, the number of possible functions is at most $|\mathcal{A}|^s$ giving an upper bound of $O(s \log |\mathcal{A}|)$ on the VC-dimension of the concept class of composed functions associated with the structure. This already gives an indication towards sample complexity savings arising from limited possibilities as compared to a general function on l nodes where l is the number of leaf nodes in the composition structure(which is $\theta(s)$ for a tree like composition).[3] In contrast the expected number of samples required for arbitrary functions on l inputs is exponential.

  While this observation in the worst case sense is somewhat pleasing it appears theres a lot more one can try. It would be nice to have a learning algorithm that consciously takes advantage of the compositional structure as is apparently the case with human learning.

  While the above reasoning already upper bounds the sample complexity of learning a composed concept, another interesting aspect to consider is the computational complexity of learning a composed concept. Given a concept class $\mathcal{A}$ and the supposed structure s what is the computational complexity of learning a function from the concept class $\mathcal{A}_s$? If the class $\mathcal{A}$ say constitutes OR, AND, NOT and composition structure is a fixed complete[4] tree one can show that learning the concept is NP-hard in the sense that we can't expect to recognize the composition or learn the concept in time polynomial in $s \approx 2^h \approx n$, the size of the structure and $|\mathcal{A}|$, unless $\mathbf{P} = \mathsf{NP}$. This leads to a very interesting question, is there

[2] albeit in different ways and with their own frailties.

[3] such savings for a path-like composition of functions that are permutations on a single input do not follow from this argument, since for example one can compose a polynomial number of transpositions as in bubble sort to construct arbitrary permutations, this is primarily since polysize composition of transpositions is universal in the space of permutations unlike say polysize tree like composition of AND, OR in the space of all boolean functions.

[4] or nearly so to allow for NOTs at leaves

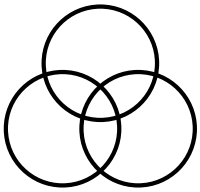

FIGURE 1. A figure easier described as one formed by 3 overlapping circles than perhaps in any other way- a simple representation interms of useful primitive objects.

a concept class $\mathcal{A}$ and a composition structure s such that the composed concepts possible are significantly expressive enough in some sense but at the same time can be learn't in a computationally efficient manner ?

- On a more involved note, in doing so, can we learn to compose in an efficient way ? By this I mean having come to know the structure, amongst the multitude ways one can choose to compute h can the machine $\mathcal{M}$ come up with an algorithm that does it in a sensible way (for example using its internal memory efficiently)?

The primary motivation of capturing how humans seem to generalize easily from very few examples but machines seem to require a lot of data has inspired very interesting works from researchers with focus on compositional learning[3, 4, 5]. Lake[6] recognizes that human level *concept learning* might require developing- an ability to compose, recognize underlying causality and the ability of learning to learn while leveraging on these to develop rich representations of new *concepts*. In [4] Lake et al. propose a model that classifies, parses, and recreates handwritten characters after having observed them as being built by bringing together simpler curves[5]. In [5] Lake and Baroni investigate the ability of models to generalize to compositional skills and propose the SCAN domain wherein compositional navigation commands are to be comprehended having learnt some basic commands and some combinations of them. Very recently there has been significant progress [8] reported on this language task by means of separating syntax and semantics and using the syntactic attention as a means for providing weighted representations in the semantics. A line of thought close to what we are curious about in the problems we mention appears in the work of Liska et al. [9]. They introduce the lookup table composition domain as a setup to test compositional behaviour and investigate how RNN like models perform. Other works like [10] appear to motivate a similar problem but what they actually do seems geared more towards problems like capturing evolving dynamics of a physical system or replicating human ability to estimate immediate future state of physical surroundings from our ingrained understanding of physics.

Before we proceed further we mention upfront what appear to be the ***main challenges***. Discovering a *useful* collection of primitives that constitute $\mathcal{A}$ which are suitable and advantageous for

[5]please refer [7] for the latest on Omniglot

the domain is actually one of the first challenges (see fig.1)[6]. Identifying the structure of composition or how these primitives are composed to perform the task at hand or construct the concept at hand is another important challenge. These challenges correspond to identifying base policies and hierarchical composition structure in the parlance of reinforcement learning. Both of these appear to be hard problems where domain knowledge might be of special help. In the next few sections we assume that both- $\mathcal{A}$ and hints to an underlying (task graph)structure are given to us and discuss what can potentially be done. Towards the end we touch upon some early thoughts on tackling the structure problem.

1. Problem Definition

Let $\mathcal{A}$ be a collection of functions. Let h be an unknown function composed using some of the constituent functions picked from $\mathcal{A}$. We are given a set of input ouput instances (x, y) of the function h so as to learn it.

Let G be a rooted directed acyclic graph representing the function composition structure of h in the following natural sense. Each node $u \in G$ is labeled by some function $f_u \in \mathcal{A}$. The in-degree of a node is the same as the arity of the function corresponding to it. A given input x assigns values to every leaf in the dag G, the outgoing edges from a node take the value of the node, an internal node in G evaluates to the value of the function at that node at inputs given by its incoming edges. The value of G, i.e the value of the composition at input x is the value of its root. A graph G represents the composition structure for a function h if the root evaluates to $h(x)$ at every input x. Given a set of examples (x, y) for h, our goal is to find the composition structure of h.

To make things more precise one can imagine $\mathcal{A}$ to be some concept class, for example one that summarizes our knowledge of different transformations that can act on an object of interest. In order to get a handle, for the time being lets think of it as represented by some subset of the set of multi-variate permutations, $f : [k]^t \to [k]$ so that the output is a bijection in every individual input coordinate when the rest of the coordinates are fixed. We hope the choice of these functions would let us capture certain canonical ways of computing a composed function leveraging the compositional structure.[7] Note that in real life the collection $\mathcal{A}$ which represents our custom prior knowledge may contain multitude of very special functions representative of the task domain unlike permutations. As an example consider the kind of tasks that we see in aptitude tests, involving a class of functions

[6]Valiant [11, 12] posits a collection of such basic primitives for cortical computation. As a computational device learns more composed tasks it might be easier to express a new task in terms of such newly learnt ones, funnily while this might be efficient in the sense of Kolmogorov complexity, not necessarily so in the computational complexity sense

[7]The intention here is that, since the value of the function depends in an equally significant manner on each of its inputs one has to evaluate the individual sub-functions corresponding to the arguments on the way towards computing the function composition, avoiding devious short-cuts for computing h. A word of caution is due here since there are issues of ambiguity associated with availability of multiple representations as a result of having alternate complete basis functions as a subset of $\mathcal{A}$, like in the case of (*and, or, not*) gates vs *nand* gate for boolean functions, we really want to get around such considerations since we are aiming to capture something else here.

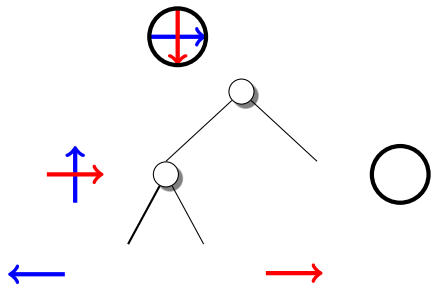

FIGURE 2. This figure illustrates an instance of function composition that uses the same binary funcion $f(p_1, p_2)$ everywhere**:** rotate the first figure p_1 90° clockwise and superimpose the second input figure p_2 on it.

that act on geometric figures by applying transformations like rotations and superpositions and we are expected to predict the underlying patterns. See figure:2 for one such composition.

1.1. **Pebbling procedures.** Consider the following example of a composition structure, a complete binary tree where internal nodes are binary functions that receive the value of their children as arguments and the composition evaluates to the value of the root as shown in figure:3 . *Pebbling* procedures [13] capture various ways of solving a function composition problem h and serve as a good proxy for understanding memory in a class of algorithms solving a problem. The simplest of these is *black pebbling* which proceeds according to the following rules.

- We can place a pebble on any leaf.
- If both children of a node u are pebbled, we can slide one of these pebbles to node u.
- Can remove a pebble at any time.

An optimal pebbling scheme is one which uses the least number of pebbles on the way towards computing the root. Figure:3 shows such an example. For a given composition structure an optimal black pebbling scheme can be seen to resemble or identified with the *canonical* fashion in which one would compute h. Such an optimal pebbling scheme is exactly what we wish to build an attention mechanism around so that our machine $\mathcal{M}$ learns to compute the composition h.

1.2. **A framework for learning to recognize compositions.** Now we present a tentative framework for a learning scheme that learns a composition built from functions coming from a collection $\mathcal{A}$. At this point we do not yet have a good understanding of how to approach this problem when the underlying structure itself is not available for someone teaching the machine $\mathcal{M}$ and if the machine were to infer this on its own[8]. We wish to think of this as a setting where someone pro-actively hints at the underlying structure. In the example in figure:2, this might be taken to be synonymous to

[8]this appears to be a hard problem in the sense that in general one cannot in any obviously efficient way infer even if there exists(purely from complexity theoretic considerations) a composition structure of certain size s that computes h using functions from $\mathcal{A}$ unless the collection $\mathcal{A}$ is very special and by some means we know how to capture this special-ness via let's say a promise of unambiguous representation. Nevertheless, there are some initial ideas worth exploring discussed in section.1.3

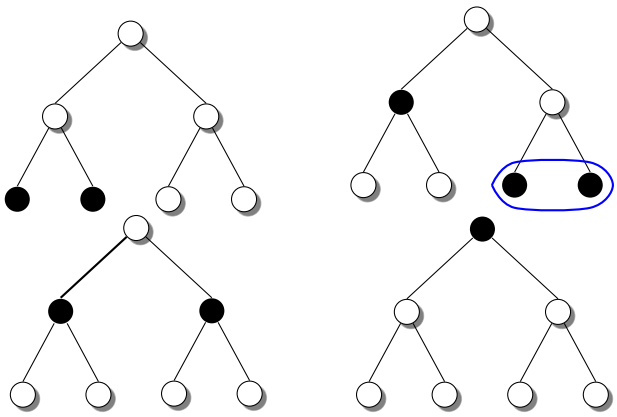

FIGURE 3. This figure illustrates a black pebbling of $h = f \circ g$ where $f, g : [k]^2 \to [k]$ using 3 pebbles(we skip some obvious intermediate steps). Configurations move from left to right in the first row and then left to right in the second respectively.

suggesting at a step in the computation- which two particular pictures(potentially composite themselves) should one *look at* and the machine figures out what to do with them i.e how to put them together. This places this method in between the two extremes of a complete hands-off approach as is the case with many supervised learning methods which seem to do well when there is a lot of data available and a completely hands-on approach where minute details are coded into the method.

In this framework it appears that we need individual parts which can perform the following in the order mentioned. Throughout - the machine has to have some kind of running memory vector, think of this as the amount of active working memory coming form its past computation history which we desire to be not too large. The machine needs to be able to

- At the beginning of each step compute which parts(pebbles) of the history vector of previous computations to *emphasize* or pay attention to in the next step along with the input, which is the value of next leaf in the dag. (An example of nodes to pay attention to at a particular step is illustrated by nodes circled in blue in fig:3)
- Suggest which function $f \in \mathcal{A}$ is to be used at the current step.
- Compute which part of our history of computations in memory can be *summarized* to be remembered succinctly and which part of it can be *forgotten* from now on after having used the sub-function f in the previous step.

We would like our framework to achieve the 3 goals presented at the beginning- learn the composition, to manage to do so having seen relatively much smaller number of examples(as compared to what we expect from a function of that arity) and ideally also learn to compose memories efficiently along the way.

Before we proceed lets briefly view the problem of learning to perform tasks in the light of what we know about how humans actually manage to perform tasks. Motivation for coming up with a framework like the one we suggest also derives from the well known role of Hippocampus as memory for holding *cognitive maps* while performing tasks like navigation [14](but not limited to navigation

alone[15]). The hippocampus is known to be essential for forming cohesive memories of individual events using information from different input streams(visual, hearing, olfaction and information about position) coming from the cortex [16, 17, 18]. Then a sort of associative augmentation is achieved by connections from hippocampus back to pre-frontal cortex[16] after having built a tentative summary from individual streams and this seems crucial in how we perceive and respond to a situation. It has also been posited[19] that the Hippocampus serves as a predictive map where the activation of place cells correspond to prediction of future positions or a weighted representation of current state in terms of expected future states. When performing a certain task these three aspects seem to provide a fruitful way to think of 1) setting out to plan, 2) perceive and summarize in real-time and 3) anticipate /predict for the future. A partially resolved composition structure on the way towards performing a task in a framework like the one we described can be seen to correspond to a form of cognitive map of the task at hand while the immediate entities that deserve attention correspond to predictions of future entities to lay emphasis on. I am not aware of how much we know in terms of how we organize things in our memory (in so far as neuroscience literature goes) when we execute composed tasks requiring us to summon different individual skills to be performed over a larger window of time versus at a swift pace, for example when we plan for a rally pattern while playing squash or tennis, or at an even smaller granularity when we make a complex dance move[9]. Although we may not seek a complete parallel to the schematics in the brain it helps to have a guiding example and know more about how we learn and perform complex composed tasks and what happens in important areas like the hippocampus as we perceive the reality and organize it as we go on to execute them.

A few remarks are due for the astute reader. If the composition structure is known the attention mechanism or pebbling procedure can be first taught to the machine in isolation in a manner oblivious of what the underlying functions are and then the machine can proceed to learn them. Also, the way we defined our problem doesn't appear to immediately lend to some real life examples mentioned in the beginning since each new instance of an example can potentially come with its own structure unlike in our case where we assume new examples are generated via the same structure acting on an input as in the case of the problem in fig:2. If we think about it, it appears one of the common ways we observe composition in real-life examples is in the sense of object composition i.e there is some way of assembling individual things together(like in Lego or hand-written omniglot characters of [4]) and the manner we seem to be able to grasp it is when we first observe the object we seem to sort of work our way back from it to see what the constituent parts are and then imagine how these individual parts came to be from things we know. Whenever we are able to recognize and vizualize these pieces going backwards we seem to get an idea of how it was assembled[10]. Moreover in these scenarios involving *concepts*, recognizing the assembly structure from the constituents appears to be a somewhat different problem compared to the problem of inferring the underlying structure in the

[9] for certain continuous tasks it seems it would be very nutty to have anything like a discrete model to summarize current state and have cognitive map of the task, if they do in fact follow the schematics of information flow we just discussed

[10] some very recent works propose theories for how this happens[20]

form of recursive/ hierarchical composition that we consider, albeit both seem crucial in being able to understand or comprehend numerous new things we encounter leveraging on previous knowledge.

1.3. **Learning Structure: Concise Compositions via penalizing Complexity.** In this section we briefly discuss how we may attempt learning structure. Since learning can be computationally hard for general primitive classes $\mathcal{A}$ and composed targets h even when underlying structure is known, the challenge for us is to implicitly bias the system towards intelligent representations so that it works for well chosen targets of our/human interest and appropriate primitive classes. This begs the question what should this implicit bias be? A necessary aspect of human intelligence is a proclivity for concise yet accurate composed explanations and executions in understanding and interacting with the environment. Conciseness can help to reduce ambiguity and round about solutions. Using this as the guide, we can take the length of representation or number of function calls made to primitives in $\mathcal{A}$ as the cost to be minimized along with empirical loss. The length of attention vector can be added to cost but these are only quadratically apart for a structure that the machine comes up with.

For learning over a longer time the primitive function set A of the learner may be selectively expanded synonymous to acquiring more skills and understanding the world in terms of these skills. Depending on the domain of interest our intention should be to expose the agent to new tasks.

1.4. **Hierarchical Reinforcement Learning.** What we seek to do here appears to have strong precedent and parallels in the motivation that led researchers to think about hierarchical reinforcement learning. The main challenges and open problems in hierarchical reinforcement learning are

- subgoal or option discovery
- learning the hierarchy or subtask composition structure and
- organizing a long term memory efficiently in the presence of delayed and sparse rewards.

The frameworks that have been proposed in the context of hierarchical reinforcement learning include Options [21], Hierarchical abstract machines [22], Feudal networks [23] and the MaxQ [24] framework. These stand to leverage the same power of *composition* and *re-use* that we seek to gain in our way of formulating the problem.

It is apparent that in the problem we describe considering each new leaf input observation as being synonymous to a state observation both situations aim to tackle very similar problems. Subtask/option discovery corresponds to learning new useful primitives for summarization, learning the task structure or hierarchy of options/policies corresponds to the problem of learning the composition structure. The RL setting however differs in terms of the way it allows for interaction with environment- the presence of rewards and how agents actions influence future observations. The latter however can be assimilated into an appropriate distribution for the input x that captures the randomness involved with state transitions(would be a product distribution on input coordinates if state dynamics are Markovian). While once again the relevant domain where the task is to be learnt might hold advantageous clues to approaching these problems, it would be nice if one can come up with approaches that apply more broadly.

References

[1] https://www.reddit.com/r/edmproduction/comments/459bjl/music_deconstruction_methodology/.

[2] Peter Sterling and Simon Laughlin. *Principles of neural design*. MIT Press, 2015.

[3] Jürgen Schmidhuber. Towards compositional learning in dynamic networks technical report fki-129-90.

[4] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[5] Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2879–2888, 2018.

[6] Brenden M Lake. *Towards more human-like concept learning in machines: Compositionality, causality, and learning-to-learn*. PhD thesis, Massachusetts Institute of Technology, 2014.

[7] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. The omniglot challenge: A 3-year progress report. *CoRR*, abs/1902.03477, 2019.

[8] Jake Russin, Jason Jo, Randall C. O'Reilly, and Yoshua Bengio. Compositional generalization in a deep seq2seq model by separating syntax and semantics, 2019.

[9] Adam Liska, Germán Kruszewski, and Marco Baroni. Memorize or generalize? searching for a compositional RNN in a haystack. *CoRR*, abs/1802.06467, 2018.

[10] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[11] Leslie G Valiant. What must a global theory of cortex explain? *Current Opinion in Neurobiology*, 25:15 – 19, 2014. Theoretical and computational neuroscience.

[12] Leslie G. Valiant. Toward identifying the systems-level primitives of cortex by in-circuit testing. *Frontiers in Neural Circuits*, 12:104, 2018.

[13] Stephen Cook and Ravi Sethi. Storage requirements for deterministic polynomialtime recognizable languages. *Journal of Computer and System Sciences*, 13(1):25–37, 1976.

[14] John O'keefe and Lynn Nadel. Précis of o'keefe & nadel's the hippocampus as a cognitive map. *Behavioral and Brain Sciences*, 2(4):487–494, 1979.

[15] Howard Eichenbaum. The role of the hippocampus in navigation is memory. *Journal of neurophysiology*, 117(4):1785–1796, 2017.

[16] Alison R Preston and Howard Eichenbaum. Interplay of hippocampus and prefrontal cortex in memory. *Current Biology*, 23(17):R764–R773, 2013.

[17] Lila Davachi. Item, context and relational episodic encoding in humans. *Current opinion in neurobiology*, 16(6):693–700, 2006.

[18] Rachel A Diana, Andrew P Yonelinas, and Charan Ranganath. Imaging recollection and familiarity in the medial temporal lobe: a three-component model. *Trends in cognitive sciences*, 11(9):379–386, 2007.

[19] Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. The hippocampus as a predictive map. *Nature neuroscience*, 20(11):1643, 2017.

[20] Jeff Hawkins, Marcus Lewis, Mirko Klukas, Scott Purdy, and Subutai Ahmad. A framework for intelligence and cortical function based on grid cells in the neocortex. *Frontiers in Neural Circuits*, 12:121, 2019.

[21] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[22] Ronald Parr and Stuart J Russell. Reinforcement learning with hierarchies of machines. In *Advances in neural information processing systems*, pages 1043–1049, 1998.

[23] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pages 271–278, 1993.

[24] Thomas G Dietterich. The maxq method for hierarchical reinforcement learning. In *ICML*, volume 98, pages 118–126. Citeseer, 1998.