Deep Generalization based on Types
B659 - Project Report

Paventhan Vivekanandan

Indiana University Bloomington
pvivekan@iu.edu

Abstract

Explainable AI concerns giving reasons for the choices made by an AI system. Many ways has been explored ranging from using externel tools for generating explanations to modifying the architecture of a classification or a regression model to account for interpretability. Sometime the efforts to improve interpretability has caused negative impact on the accuracy of the predictions. In this paper, we discuss a more natural way of generating explanations. We follow the path of strong AI to build a system that think and reason like humans. The system builds a knowledge graph based on observations and it uses type abstraction to make inferences. The system also generates explanations to reason about the inferences it made. The explanations generated by the system aligns with how a human would reason about his choices.

Introduction

Explainable AI concerns giving reasonable explanations for the decisions made by AI systems (Mueller et al. 2021). This is becoming more of a necessity in deep learning where the black-box AI systems are increasingly involved in critical decision making such as estimating the likelihood of a person having cancer or not or whats the best course of treatment for a patient with certain symptoms (Graziani et al. 2020)(Alam and Mueller 2021). When a human makes a decision, it is very natural for him to explain why he made this decision. For humans, decision making and explaining the reason behind it goes together. But for machine intelligence we have to keep the process of decision making and explaining it separate. This situation is partly due to the increased attention to weak AI which assumes machine intelligence need not be identical to human intelligence. The weak AI models such as neural networks make extensive use of mathematical tools. Explaining the decisions made by those models needs effort to understand the underlying mathematical formation and how the inputs are processed to give the output. Neural Networks has so many non-linearities and may involve hundreds of thousands of parameters. Tracking them is impractical and this means we need algorithms (for generating explanations) that does not necessarily correspond with the underlying prediction algorithms in use. Strong AI, contrary to weak AI, argues that artificial intelligence should be developed identical to human intelligence. This direction is hugely ambitious to pursue and only very small progress has been made so far. One of the reason for this is because human knowledge is partly tacit and very hard to articulate. Artificial General Intelligence (AGI), typically considered as part of weak AI, share some common grounds with strong AI due to the ability of the human intelligence to generalize (Fjelland 2020). An important consequence of trying to solve the problem of strong AI and AGI is that the explainable AI can arise naturally from it due to the technique's resemblance to human intelligence. Also, the algorithms for generating explanations in strong AI are intrinsic to the system and does not converge so much from the prediction algorithms.

Figure 1: Identifying a dog in an image

Building Knowledge based on Types

Humans are good at generalizing. This means that if we know how to solve a problem, we can extend this knowledge to another problem based on the relevance between the two problem statements. As a simple example, consider teaching a child to count the number objects in a given image and also teach him how to identify a dog (fig. 1). The child can easily extend the knowledge learned during solving these two problems and apply it to previously unseen problems such as counting the number of dogs (fig. 2). This kind of problem solving skills requires very strong ability to generalize. Humans built this kind of generalizing ability over millions of years of evolution. The current generation machines lacks this generalizing ability which is a huge bottleneck for the progress in AGI.

Figure 2: Count the number of dogs in an image

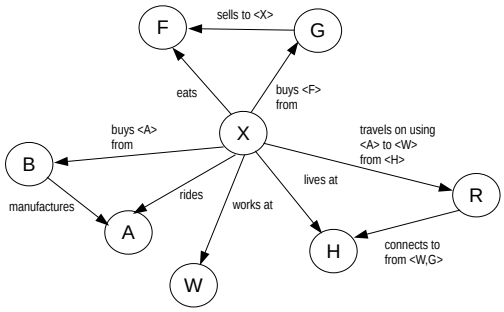

Figure 3: U_1 type system structure

Answering how the human mind evolved over the past few million years can address important questions in many fields including Cognitive Science, Artificial Intelligence, and Psychology. The early human mind stored experiences as episodes (or concepts) and had no ability to chain them together to get an abstract view of the world. A critical point in its evolution was the ability to stitch together these episodes to form a holistic understanding of the world (Gabora 1999)(Donald 1991). The experiences that the episodes record has type information in it. Stitching those type information gives rise to a type structure which can assist in sophisticated decision making. Types are collections of objects with certain characteristics or properties. We can identify a type using those properties. For example, the type $Human$ can be identified using properties such as $rational$, $sensitive$, and $animate$ while the type Dog can be identified as $irrational$, $sensitive$, and $animate$. A type can exist as standalone or it can combine with other types to form new compound types. The compound types themselves can combine to form bigger compound types. When types combine, they can also induce a hierarchy. For example, we can form type $Animal$ from $Human$ and Dog with properties $sensitive$ and $animate$. The type $Animal$ acts as a super type of $Human$ and Dog. At the *Animal* level both *Human* and *Dog* are the same. If we get more specific and less abstract, the type *Human* differs from *Dog* by property *rational*. If we get less specific and more abstract, we can form a new type *Living* with subtypes *Animal* and *Plant* which differs based on property *sensitive* at the next level of abstraction. In this way, we can go up or down in the level of abstraction which will give us new ways to find relevance between seemingly different types. The real world has many complicated relationship between objects. We can capture those relationships based on observations and abstracts the objects using types. Then we can form type hierarchies to assist in knowledge transfer. We can represent this process as a graph with nodes representing the types and edges representing the shared properties between the connecting nodes. This gives rise to interesting knowledge structure which may involve multiple levels of abstraction. If we go to higher levels of abstraction, the structure gets more and more simple leading to a very few nodes representing broad categories. Similarly, if we go too specific, the structure gets more disconnected at which point no interesting relevance occurs.

Generalizing based on Types

Stitching the episodes together based on type information closely align with what Leonardo da Vinci says "These are the principles for the development of a complete mind: Study the science of art. Study the art of science… Realize that everything connects to everything else" (Ahmed 2019). This polymath way of thinking is of significant importance to the development of Artificial General Intelligence (AGI) (McCullen 2019)(Rodrigue 2020) as well as progress in strong AI. The explainable AI arise naturally in this process. Once we have a type structure, any decisions based on the type structure will simply involve tracing a path in the type structure and explaining how that path lead to the decision. The type hierarchy induces a structure in which moving higher up the hierarchy leads to more generic types and moving lower leads to more specific types. In machine learning, to learn a model we need data. The quality and the volume of the training data determines the accuracy of the model. But in the real world, we might never get enough data to learn some models. If we consider a model as part of the type structure, then we can use instances (data) from other parts (which can themselves be models) of the type structure and translate that information in accordance with the type structure and infer something about the new model for which we lack data. To understand the type structure lets consider an open domain with a toy universe U_1. Think of all the monotonous tasks we humans do everyday and lets see how this can lead to an interesting type structure. Assume the toy universe has types human X, automobile A, workplace W, grocery store G, food F, automobile manufacturing company B, route R, home H. Also, consider the following episodes which has been labeled with relevant types.

- (Bob: X), (vegetables: F), (Kroger: G), (Bloomington: H), (Indianapolis: W), (car: A), (ford: B), (eats: X $\leftrightarrow$ F), (buys: X $\leftrightarrow$ G), (drives: X $\leftrightarrow$ A), (works: X $\leftrightarrow$ W)
- (Alice: X), (fish: F), (Kroger: G), (Bloomington: H), (Columbus: W), (car: A), (mazda: B), (eats: X $\leftrightarrow$ F), (buys: X $\leftrightarrow$ G), (drives: X $\leftrightarrow$ A), (works: X $\leftrightarrow$ W)
- (Peter: X), (meat: F), (Kroger: G), (Columbus: H), (Indianapolis: W), (motorcycle: A), (kawasaki: B), (eats: X $\leftrightarrow$ F), (buys: X $\leftrightarrow$ G), (drives: X $\leftrightarrow$ A), (works: X $\leftrightarrow$ W)

Stitching the above instances together might give rise to a type structure as shown in fig. 3. The types (nodes in fig. 3)

Listing 1: Boolean AND `bool.agda`

```
1  _and_ : Bool -> Bool -> Bool
2  true  and true  = true
3  false and x     = false
4  x     and false = false
```

are uniquely identified by relations (edges) and some properties in the toy universe U_1. Learning this type structure can help make new inferences and produce previously unseen knowledge in the system. Type theory, an implementation of constructive logic, treats types as collection of objects with an equivalence relation (Nordstrom, Petersson, and Smith 2000). The AI system proposed in this paper gives a similar treatment to types and it views every type as uniquely defined by its properties and connections (edges) to other types. When a new type instance is given as input, the system works on identifying the correct position for the type in the knowledge graph. If the new instance corresponds to an existing node in the graph, then no new node creation takes place. If the system is satisfied that the new type is unique, then it creates a new node to represent the type and infers at which position it should add the new node in the graph. In this way the system exhibits its polymath beviour where it tries to incorporate every new instances into a single connected component. The inference is supported by the structure of the connected graph. The knowledge transfer between nodes is restricted by the type information realized by the graph. This can be compared to how programs in typed languages such as *Agda* and *Haskell* uses type specifications to restrict the implementation space. For example, the agda program in listing 1 uses the type specification and restricts the program space to consider only programs with inputs and output of type boolean.

Type-based Learning and Explanations

In this paper, we discuss an AI system which builds a connected knowlegde structure by capturing real world relationships and by modeling type hierarchies using type inference. The real world has many complicated relationships. Normally, humans learn about their environment using observed data. The observed data has gaps and is rarely complete. As we see more and more relevant data we bridge those gaps and our understanding of the environment gets better. Using the same intuition, the AI system described here builds the knowledge structure based on the observed data incrementally. The observed data is always assumed to be true. As it continues to build the knowledge structure, the system makes type inferences to fill in the gaps based on common properties it observes between the types and as a consequence new edges and nodes might be added. At any time, the system will not add information that will violate the observed data. The inference behaviour is similar to how a human probabilistically reasons about a given situation based on partially observed informations. Subtly, we try to fill in the hidden gaps based on the most relevant experiences. This is analogous to case-based reasoning where to deal with a previously unseen situation we retrieve the most similar case and adapt the retrieved case to make it fit to the new situation. Generating explanations for black-box models such as neural networks might involve employing another machine learning model (can be a neural network) (Alvarez-Melis and Jaakkola 2018) or using external tools such as anchor LIME (Ribeiro, Singh, and Guestrin 2018). The explanations can be in the form of factuals, semi-factuals, or counter-factuals (Kenny and Keane 2021) or sometimes it simply involves giving the closest neighbors from the data set. For symbolic AI techniques such as case-based reasoning, the explanations can be in the form reasoning about case adaptation rules. Generating explanations for our AI system involves arguments based on observed evidence and also reasoning about the choices made by the system for inferences. The explantion generation process is intrinsic to the system and is done by tracing the path of an inference in the knowledge structure and giving reasons for why the inference is made. This can be compared to how humans reason about their choices based on the most relevant experiences.

Implementation

The AI system attempts to have a single connected graph all the time. If a new instance comes as input, it looks for properties which are similar to the properties in the connected graph. It then creates a new node for the input instance and attaches the node to the connected graph in the *best* possible position. Sometimes, the new node might not share anything common with the graph and this can lead to disconnected components in the graph. This behaviour is allowed but on input instances containing enough properties to connect the disconnected components of the graph, the system should be able to make those connections resulting in a single connected graph. The system here demonstrates its polymath behaviour where it always looks for informations that could allow it to connect everything. The system will have some initial environment to work with. Lets briefly discuss the node and edge inference made by the AI system[1]. After that lets talk about the explanation generation part in detail.

Node and Edge Inference

In fig. 4, we can see an example for node and edge inferences made by the AI system. Here, the observed data is added to the system one step at a time. The system builds the connected knowledge structure incrementally. While adding the new information, it makes type inferences and builds a type hierarchy when possible. Based on the type hierarchy, new edges are inferred and added.

Initially, the types *Hospital*, *Park*, *Industry*, and *Courthouse* is added to the type graph. All of them share the same relationship with *Place* and *Road* as that of *School* and *Home*. After that, the type *Kid* is added to the graph. The edges reflects the properties that are shared by the connecting nodes. For example, the edge between types *Kids* and *Park* has properties *Animate* and *Playing*. When a new type node is added to the graph, the system infer new edges based on the relevance of the new node to the neighbour nodes.

[1]This part has been explained in detail for B652 Symbolic AI project. So, keeping it short here.

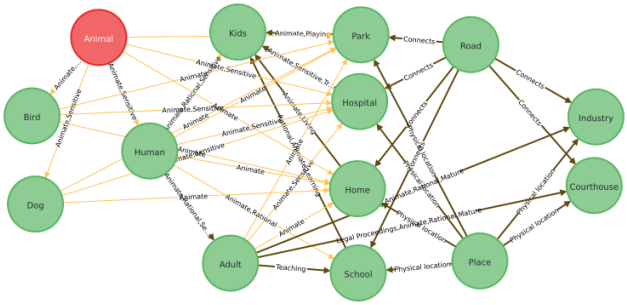

Figure 4: Node and Edge Inference

When type *Human* is added, the system first deduces that it is a supertype of *Kids*. After that, it infers the edges that are applicable to type *Human* based on the edges of type *Kids*. The system infers that edges *Kids* ↔ *Park*, *Kids* ↔ *Hospital*, *Kids* ↔ *Home*, *Kids* ↔ *School* are all applicable to *Human* as well. Similarly, when type *Adult* is added, the system infers the relevant edges from its supertype *Human*. For a new input instance, the system also checks if creation of nodes in addition to the node reflecting the current instance is required. When type node *Dog* is added, the system infers and adds node *Animal* which is a supertype of both *Dog* and *Human*. Also, new edges are inferred for type *Animal* based on the properties it has common with types *Human* and *Dog*. The system infers that the type *Animal* has relevance to *Park*, *Hospital*, and *Home* as it is *animate* and *sensitive* but it is not related to *School* because it is not *rational*. Similar inferences takes place for types *Dog* and *Bird* (see fig. 4).

Explanation Generation

The system supports three kinds of queries. First, for a given set of properties, the system finds the most relevant type. Second, if two types are given, the system returns the most relevant properties based on which the two types are related. Third, for a given type and a set of properties, the system returns the type that is most likely to be related to the given type based on the given properties. For all cases, if there are inferred relations, the system explains the reasons for making those inferences. To generate explanations, it traverse through the knowledge graph and identifies the inferred edges. After identifying the inferred edges, it checks if the inference is made due to a sibling with matching properties having the same relationship. It continues to loop through the knowledge structure until it finds a related sibling which has real world evidence to support the relationship. Consider the knowledge graph given in fig. 5. Here, we have made the properties and nodes abstract. We have six types ”t1” to ”t6”. Type ”t1” is the super type for ”t2” and ”t3”. Type ”t5” is the super type for ”t4” and ”t6”. Any inferred edges or nodes or properties will start with string ”inf_”. Also, the thickness of the edges has been increased and the colors are modified for legibility. Red edges denotes super type to sub type relationship. Blue denotes *inferred* super type to sub type relationship. Brown denotes relationship based on direct evidence and yellow denotes *inferred* relationships. The following are the examples of queries and generated explanations for fig. 5.

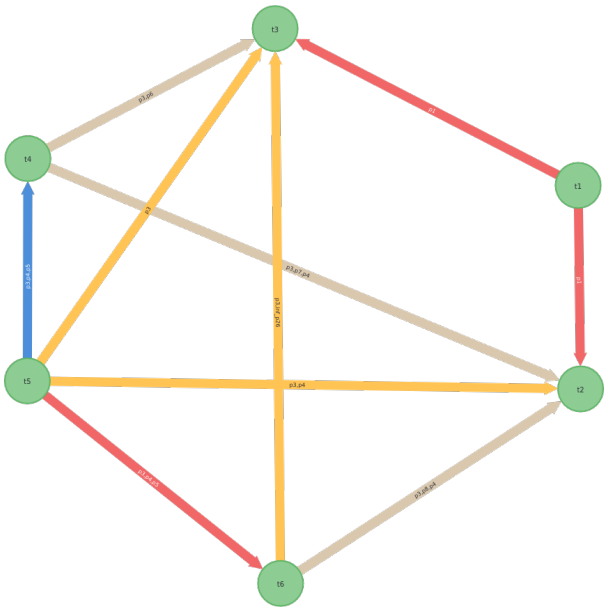

Figure 5: Explanation Generation

Query 1 When we see an object in our environment, we try to identify its characteristics. Based on the characteristics, we try to figure out what is the most relevant type of this object based on our previous expericences. For example, if a child travels to a countryside for the first time and see a bullock cart, he observes the cart wheels and recognize that the object is used for some kind of travelling. If we ask the child to explain, he will most likely show the wheels as the reason why he concluded this object is used for travel purposes. We are capturing the same intuition as part of query 1. Here, a set of properties is send as input and the most relevant type is returned. Also, the system explains its decision by listing the properties of the returned type and giving a relevance score. Also, the system returns a list of the next best matches. This is similar to a conversational case-based reasoner which returns the the list of most relevant questions or answers to queries based on relevance scores.

Listing 2: Query Type 1

```
anwser_query_with_explanation
    (["p2", "p3", "p4"], "", "", 1)

The best matching type is t4 with
properties ['p2', 'p3', 'p4', 'p5']
and score 3

The next best matches based on scores:

type: t5,
properties: ['p3', 'p4', 'p5'], score: 2

type: t6,
properties: ['p3', 'p4', 'p9'], score: 2
```

In listing. 2, the list of properties [”p2”, ”p3”, ”p4”] is send as input and the system returns type ”t4” (see fig. 5) as the most relevant type with similarity score 3. The next best matches are also returned by the system.

Query 2 After identifying the bullock cart as some sort of vechicle, the child might try to understand what kind of travel is possible. Assume the bullock cart is just static without bullock when the child see it. If the child is relating the bullock cart to a car or a bus, he observes that the bullock cart is missing seats for people to sit in it. Therefore, a bus or a car is not the closest match. Then he moves on to compare with another vehicle. Lets assume he is comparing with a trash pickup truck. Now he can identify that the empty space in the cart is for loading objects. If we ask for his explanation, he will trace through his choices and list them one by one saying why he rejected the bus and chose the truck as the closest match. This idea of comparing two objects based on relevant properties is modeled by query 2. Here, we send two types as input and the system retrieves the most similar properties. Also, it explains its choice by giving the closest matching sibling. After that, if the returned type has inferred properties, it gives information on what that properties might be.

Listing 3: Query Type 2

```
anwser_query_with_explanation
    ([], "t6", "t3", 2)

t6 relates to t3 based on
properties ['p3', 'inf_p26']

I am inferring the edge because of
the following reasons:

The properties ['p3'] of t6
is common with sibling t4

The inferred property ['inf_p26'] of t6
relates to ['p6'] of t4
```

In listing. 3, the types ”t6” and ”t3” are send as input and the system returns the properties [”p3”, ”inf_26”] (see fig. 5). It explains that the sibling ”t4” has the closest match. If we consider ”t6” as the bollock cart, ”t3” as travelling, ”t4” as trash pickup truck, ”p3” as wheels, the above scenario fits this case. The system also relates the inferred property ”inf_26” to ”p6” (assume ”p6” as the body of the truck to load objects).

Query 3 Query 3 is a more generalized version of query 2. Here, we send a type and a list of properties as input and the system returns the most relevant type. This process is similar to the child going through his choices one by one and finding the most relevant vehicle to bullock cart. In addition to the best match, the system also returns the next best matches with relevance scores. Again, the system explains its choices for inferences.

In listing. 4, we send list of properties [”p3”, ”p4”] and type ”t6” as input and the system returns the most relevant

Listing 4: Query Type 3

```
anwser_query_with_explanation
    (["p3", "p4"], "t6", "", 3)

The best match to type t6 is type t2
with properties ['p3', 'p8', 'p4']
and score 2

The next best matches based on scores:

type: t3,
properties: ['p3', 'inf_p26'], score: 1

t6 relates to t3 based on
properties ['p3', 'inf_p26']

I am inferring the edge because of
the following reasons:

The properties ['p3'] of t6
is common with sibling t4

The inferred property ['inf_p26'] of t6
relates to ['p6'] of t4
```

type ”t2” (see fig. 5). Since the properties [”p3”, ”p8”, ”p4”] common to input ”t6” and output ”t2” are not inferred (they are from data evidence), the system does not make any inferences. The system also returns the next best matches. For the next best match, we have an inferred property for which the system explains the reason for making the inference.

Implementation details

The AI system described here is implemented in python and it uses neo4j graph database. As neo4j does not have a double headed edge representation, the system do not give any specific interpretation for the direction of edges. On any input type instance, the system checks to see if new node creation is required (inferred nodes are shown in red). If new node creation takes place, then corresponding edge inferences are made and the updates are pushed back to the graph database. The input instances consists of a type with properties and relations to other types. The properties and relations need not be complete. In such cases, the AI system will make inferences to see if it can fill in the missing informations. The system trusts the initial environment and does not add new inferred data that would contradict the initial environment or any observed data. Also, the system assumes that the vocabulary for type properties are unique in the sense that if *commute* is used to denote travelling, then *travel* is not used to denote the same. For opposite words, symbol ’!’ should be used. For example, *!Rational* is used instead of *Irrational*.

Conclusion and Future Work

This paper discuss how to build a knowledge graph based on observations. The observations contains partial information like any real world data instance would. The AI system makes type inferences to fill those gaps if and when avail-

able and enrich the knowledge graph. Three types of queries are supported by the system. The queries involve sending a list of properties as input and retrieving the most relevant type, sending two types as input and retrieving the common properties, sending a type and a list of properties as input and retrieving the most relevant type. Explanations are generated for cases involving inferences. To improve the explanation part, the future work aims at having a black-box system exposed to more complete observations compared to the AI system. The knowledge graph of the black-box system should be bigger and should have less inferences compared to the AI system. Any queries will be answered by both systems. The AI system can refer to the explanation generated by the black-box system and it should incorporate any new knowledge it can learn from those explanations. If we assume the black-box as a teacher and the AI system as a student, the teacher has more knowledge of the subject compared to the student. If the student can't understand an answer for a question, he asks for explanations to the teacher. The teacher explains and the student will improve his knowledge based on the quality of the explanations.

References

Ahmed, W. 2019. The Mind of Leanardo da Vinci. *Philosopy Now*, 134.

Alam, L.; and Mueller, S. 2021. Explanation Strategies for Trustworthy AI Diagnostic Systems: Examining Physicians' Explanatory Reasoning in Re-diagnosis Scenarios. *Trustworthy AI in Healthcare Workshop*.

Alvarez-Melis, D.; and Jaakkola, T. S. 2018. Towards Robust Interpretability with Self-Explaining Neural Networks. *CoRR*, abs/1806.07538.

Donald, M. 1991. Origins of the modern mind. *Harvard University Press*.

Fjelland, R. 2020. Why general artificial intelligence will not be realized. *Humanities and Social Sciences Communications*, 7.

Gabora, L. 1999. Weaving, Bending, Patching, Mending the Fabric of Reality: A Cognitive Science Perspective on Worldview Inconsistency. *Foundations of Science*, 3.

Graziani, M.; Andrearczyk, V.; Marchand-Maillet, S.; and Müller, H. 2020. Concept attribution: Explaining CNN decisions to physicians. *Computers in Biology and Medicine*, 123: 103865.

Kenny, E. M.; and Keane, M. T. 2021. On Generating Plausible Counterfactual and Semi-Factual Explanations for Deep Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13): 11575–11585.

McCullen, A. 2019. Human General Intelligence - HGI verses HI and The Age of Polymath.

Mueller, S. T.; Veinott, E. S.; Hoffman, R. R.; Klein, G.; Alam, L.; Mamun, T. I.; and Clancey, W. J. 2021. Principles of Explanation in Human-AI Systems. *ArXiv*, abs/2102.04972.

Nordstrom, B.; Petersson, K.; and Smith, J. M. 2000. Programming in Martin-Löf 's Type Theory. *Logic in Computer Science*, 5.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Anchors: High-Precision Model-Agnostic Explanations. 32.

Rodrigue, E. 2020. Creative Thinking and Artificial Intelligence.