1 Relations between Variables

Marcus starts with the following observation:

Observation 1.0.1. The human brain is capable of performing operations like addition, multiplication, inflecting verbs, and so on.

What do all these algebraic operations have in common? They all operate on variables based on rules. Thus, Marcus calls them *relations between variables.*

1.1 UQOTOM

Instead of considering all these functions of human cognition at once, Marcus focusses on a very special subclass of operations. He refers to them as *UQOTOM*, which stands for *universally quantified one-to-one mapping.* Like the name suggests, they are defined as follows:

Definition 1.1.1 (UQOTOM)**.** A mapping $f : X \to Y$ is a UQOTOM if it is *one-to-one* and *universally quantified.* Universally quantified specifies that f is defined for all $x \in X$, i.e. that f is a valid function in the mathematical sense.

Some important examples of UQOTOMs that Marcus is going to analyze are the identity function, and the simple past operation which produces the simple past of a verb. According to Marcus:

> **Marcus (2001)**
>
> I do not mean to suggest that UQOTOM are the only mappings people compute. But UQOTOM are especially important to the arguments that follow because they are functions in which every new input has a new output. Because free generalization of UQOTOM would preclude memorization, evidence that people (or other organisms) can freely generalize UQOTOM would be particularly strong evidence in support of the thesis that people (or other organisms) can perform operations over variables.

Based on this quote, we can see that Marcus is interested in UQOTOMs because they enforce a system to *generalize* mappings. That is, given a never-before-seen input, a potential model implementing this UQOTOM must produce a new output that has not been produced before. And this generalization is a central aspect of *symbol manipulation* according to Marcus.

Marcus (2001)

> To a system that can make use of algebralike operations over variables, free generalization comes naturally.

Furthermore, he specifies algebraic rules further and he writes:

Marcus (2001)

> Algebraic rules are not finite tables of memorized facts or relationships between specific instances but open-ended relationships that can be freely generalized to all elements within some class.

Most decisively, for Marcus, a UQOTOM isn't just a special type of function; it is the very implementation of an algebraic rule. He writes:

Marcus (2001)

> **Premise 1.1.1.** *When such a network represents identity or some other UQOTOM, it represents an abstract relationship between variables—which is to say that such a network implements an algebraic rule.*

Critique

The implication

$$f \text{ is a UQOTOM} \implies f \text{ implements an algebraic rule}$$

presents a foundational but significant oversimplification. By narrowing the scope of ”algebraic rules” to only one-to-one mappings, this framework fails to account for the true complexity of algebraic operations and serves as a poor indicator of a model's reasoning capabilities.

For instance, a common operation like the quadratic function, $f(x) = x^2$, is undeniably an algebraic rule, yet it is not a UQOTOM because it is not one-to-one (e.g., $f(2) = f(-2) = 4$). The fundamental flaw in Marcus's approach is this reduction of the broad, complex class of algebraic rules to the narrow, simpler subset of UQOTOMs. A model's ability to handle this special case provides insufficient evidence of a general capacity for algebraic thought.

1.1.1 Humans can freely generalize UQOTOMs

Now that Marcus has established his framework of UQOTOMs, he argues that humans can freely generalize these UQOTOMs. He provides evidence for this claim with the following example:

Example 1.1.1. When you were presented this pattern

Input	Output
1010	1010
0100	0100
1110	1110
0000	0000
1111	?

and asked to predict the output of 1111, what would you answer?

Marcus argues that most people would answer 1111, because humans have a bias for generalizing UQOTOMs, which is why the identity mapping is generalized in this example. He follows from this empirical observation:

Marcus (2001) p. 50

Premise 1.1.2. *The bottom line is that humans can freely generalize one-to-one mappings [...]*

Thus, in Marcus's line of argumentation, humans can generalize algebraic rules based on the previous result and premise 1.1.1. He concludes:

Conclusion 1.1.1. *Models of human cognition must be able to generalize UQOTOMs, and thus algebraic rules.*

This framework of a necessary condition for models of human cognition is the basis for Marcus's following analysis.

1.1.2 Multilayer Perceptrons and Operations over Variables

Next, Marcus emphasizes the distinction between models that allocate *one node per variable*, and models which allocate *multiple nodes per variable*:

Definition 1.1.2. A model is said to allocate *one node per variable* if it has a single node for each variable in the input. A model is said to allocate *multiple nodes per variable* if it has multiple nodes for each variable in the input.

Marcus (2001)

Again, what is relevant here is not the sheer number of input units but rather the number of input units allocated to representing each input variable.

Critique

In the mathematical setting of neural networks that represent one-to-one mappings, it makes no difference whether, say, 5 input nodes all together represent a single variable, or five individual variables all encoded by a single node each. Furthermore, he also does not apply this framework, or only very vaguely, as we will see in the following.

Additionally, he points out that this distinction is not to be confused with *localist* and *distributed* representations:

Marcus (2001)

While all models that use distributed representations allocate more than one variable per node, it is not the case that all localist models allocate a single node per variable.

With this definition established, Marcus claims that:

Marcus (2001)

One-node-per-variable models, it turns out, can and indeed (the caveats in note 5 notwithstanding) must represent universally quantified one-to-one mappings.

Critique

Again, he assumes the network to only have one input node, which is not the same as having one node per variable. Furthermore, in note 5 he explains that he assumes the network to only have linear activation functions. Such a network must learn a linear mapping, and every mapping $f(x) = \alpha \cdot x$ is one-to-one for $\alpha \neq 0$.
Note that if we had multiple input nodes instead, a linear mapping is described by a matrix multiplication $f(\boldsymbol{x}) = A \cdot \boldsymbol{x}$. In this case, the mapping is one-to-one if and only if A is invertible, which is not guaranteed for all matrices.

Critique

This described case of a single input node with linear activation functions is very specific and doesn't serve any general argument. Note that when using non-linear activation functions, one can easily construct a network representing a non one-to-one mapping:

Example 1.1.2. MLPs with non-linear activations like $\tanh(x)$ can represent non-injective functions (other than $f(x) \equiv \mathbf{0}$) even when only using one input node. For instance, the MLP depicted in figure 1.1 implements the non-injective mapping depicted in figure 1.2.

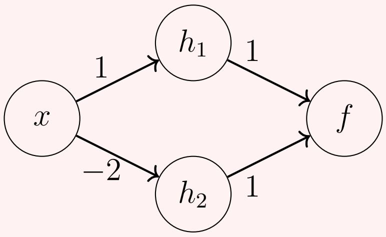

Figure 1.1: Simple MLP using $\tanh(x)$ activation implementing a non-injective mapping.

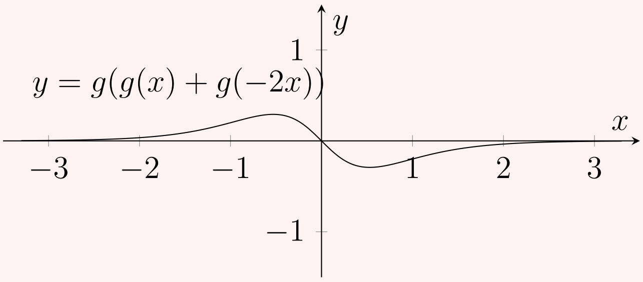

Figure 1.2: Plot of the function $y = g(g(x) + g(-2x))$ with $g(x) \coloneqq \tanh(x)$.

Regarding models that allocate more than one node per variable, Marcus states:

Marcus (2001)

Models that allocate more than one node per variable too, can represent universally quantified one-to-one mappings (see, for example, the left panel of figure 3.4), but they do not have to (see the right panel of figure 3.4).

Critique

This statement caries very little information. It only states that there exists MLPs that represent UQOTOMs, and that there exist MLPs that do not represent UQOTOMs, which is trivially true. The bigger problem, however, is that he also does not provide any further details on how to construct such MLPs, or what the implications of this distinction are.

Learning

Until now, Marcus has only discussed the ability of MLPs to represent UQOTOMs. Now, he turns to the question of whether MLPs can *learn* UQOTOMs. He states:

Marcus (2001)

The flexibility in what multiple-nodes-per-variable models can represent leads to a flexibility in what they can learn. Multiple- nodes-per-variable models can learn UQOTOMs, and they can learn arbitrary mappings. But what they learn depends on the nature of the learning algorithm. Back-propagation—the learning algorithm most commonly used—does not allocate special status to UQOTOMs. Instead, a many-nodes-per-variable multilayer perceptron that is trained by back-propagation can learn a UQOTOM—such as identity, multiplication, or concatenation—only if it sees that UQOTOM illustrated with respect to each possible input and output node.

He justifies that back-propagation does not allocate special status to UQOTOMs with an example:

Example 1.1.3. The autoencoder network depicted in 1.3 is trained to learn the identity mapping:

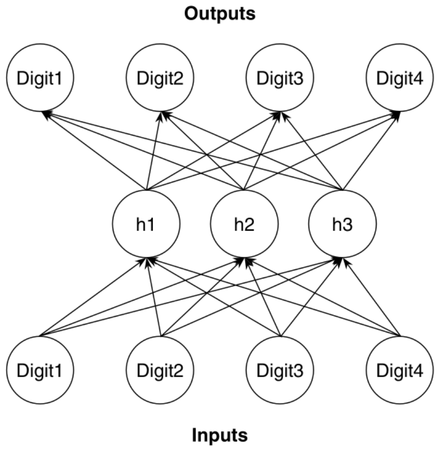

Figure 1.3: Autoencoder network trained to learn the identity mapping.

However, all training samples have a 0 as a last bit. Marcus's simulations show that the network learns to ignore the last bit by always predicting a 0, and only learns the identity mapping for the first three bits.

He formalizes this failure of back-propagation to learn UQOTOMs by defining the concept of *training independence*:

Marcus (2001)

The equations lead to two properties that I call input independence and output independence or, collectively, training independence (Marcus, 1998c).

Marcus (2001) p. 47

Definition 1.1.3 (Input Independence)**.** "Input independence is about how the connections that emanate from input nodes are trained. First, when an input node is always off (that is, set to 0), the connections that emanate from it will never change. This is because the term in the equation that determines the size of the weight change for a given connection from input node x into the rest of the network is always multiplied by the activation of input node x; if the activation of input node x is 0, the connection weight does not change.

Marcus (2001) p. 47

Definition 1.1.4 (Output Independence)**.** Output independence is about the connections that feed into the out- put units. The equations that adjust the weights feeding an output unit j depend on the difference between the observed output for unit j and the target output for unit j but not on the observed values or target values of any other unit. Thus the way the network adjusts the weights feeding output node j must be independent of the way the network adjusts the weights feeding output node k (assuming that nodes j and k are distinct).

Critique

Marcus's concept of input independence is very specific and is not really applicable for real world examples. When an input node is always set to zero in training, we could simply use a different encoding scheme. Marcus argues that for an input node that is always set to the same constant v, the network "does not learn anything about the relation between that input node and the output, other than to always set the output node to value v" (Marcus 2001, p. 47).

This fact, however, is only based on empirical experiments, and is specific to the tested model architecture. What really happens is that the network won't use this node's value for inferring the output, but rather treats it as an adjustable bias for the layer it feeds into. This is not a flaw, but rather a property of the generic architectural design of MLPs.

Critique

Similarly, output independence is just another property of MLPs which isn't necessarily a flaw. It is not the case that the network learns to predict each output node based on the input nodes independently, since the hidden layers are trained based on *all* output nodes. Marcus acknowledges this fact, he writes:

"This means not that there is never any dependence between output nodes but that the only source of dependence between them is their common influence on the hidden nodes, which turns out not to be enough. At best, the mutual influence of output nodes on input-to-hidden- layer connections may under some circumstances lead to felicitous encodings of the hidden nodes. We might think of such hidden units as quasi-input nodes. The crucial point is that no matter how felicitous the choice of quasi-input units may be, the network must always learn the mapping between these quasi-input nodes and the output nodes. output nodes on input-to-hidden-layer connections is not sufficient to allow the network to generalize a UQOTOM between nodes." (Marcus 2001, p. 47 f.)

However, MLPs are not designed to generalize UQOTOMs, they are generic function approximators. Now, maybe this is exactly what Marcus criticizes, but how can we blame a generic model to not have an arbitrary bias, in this case that of UQOTOMs, when it is not designed for this purpose? When we want our model to have a specific bias, in this case to employ algebraic rules, we have to either learn the entire domain, or integrate domain specific knowledge into the architectural design (or else the model architecture wouldn't be generic).

1.2 Marcus's proposed Alternatives

He begins by stating five properties a successful model must fulfill:

Marcus (2001) p. 51

In general, what is required is a system that has five properties. First, the system must have a way to distinguish variables from instances, analogous to the way mathematics textbooks set variables in italic type (x) and constants in bold type ($\boldsymbol{AB}$). Second, the system must have a way to represent abstract relationships between variables, analogous to an equation like $y = x + 2$. Third, the system must have a way to bind a particular instance to a given variable, just as the variable x may be assigned the value 7. Fourth, the system must have a way to apply operations to arbitrary instances of variables—for example, an addition operation must be able to take any two numbers as input, a copying operation must be able to copy any input, or a concatenation operation must be able to combine any two inputs. Finally, the system must have a way to extract relationships between variables on the basis of training examples.

Critique

These criteria are not really measurable. They seem to rather act as a guideline for designing a model architecture. For instance, how to measure whether a system has a way to distinguish variables from instances? Is model trained on both instances and variables, and what does it mean to treat the differently? How to measure whether a system has a way to extract relationships between variables on the basis of training examples? Does training data set cover the entire domain, or does the model have to generalize based on a small set of training examples, and how would we measure this generalization?

1.2.1 Encoding Schemes

Next in Marcus's line of argument, he presents different schemes to encode variables with input nodes. He wants to show that the inability of MLPs to learn UQOTOMs is fundamentally rooted in the architecture itself, and cannot easily be solved by different encoding schemes.

First, he introduces *conjunctive coding*.

Marcus (2001) p. 52

But the brain must rely on other techniques for variable binding as well. Conjunctive codes do not naturally allow for the representation of binding between a variable and a novel instance.

According to Marcus *tensor products* cannot solve all our problems as well:

Marcus (2001) p. 54

Nonetheless, despite these advantages, I suggest in chapter 4 that tensor products are not plausible as an account of how we represent recursively structured representations.

Lastly, he mentions *temporal synchrony*:

Marcus (2001) p. 57

My own view is that temporal synchrony might well play a role in some aspects of vision, such as grouping of parts of objects, but I have doubts about whether it plays as important a role in cognition and language.

Critique

It is a bit unclear why Marcus introduces these encoding schemes. He wants to show that the inability of MLPs to learn UQOTOMs is fundamentally rooted in the architecture itself, and cannot easily be solved by different encoding schemes. However, instead of providing a general argument for a generic class of encodings, he only discusses a few specific encodings.

Registers

Marcus suggests the use of *registers*. He writes:

Marcus (2001) p. 54

A limitation of the binding schemes discussed so far is that none provides a way of storing a binding. The bindings that are created are all entirely transitory, commonly taken to be constructed as the consequence of some current input to the system. One also needs a way to encode more permanently bindings between variables and instances.

Marcus (2001) p. 55

Registers are central to digital computers; my suggestion is that registers are central to human cognition as well.

Additionally, according to Marcus, registers could explain the phenomenon of *rapidly updatable memory*, which describes how we humans are able to learn things on a single trail, like remembering where we parked our car. The brain would use these registers to perform basic operations on them:

Marcus (2001) p. 58

My hunch is that the brain contains a similar stock of basic instructions, each defined to operate over all possible values of registers.

Marcus claims that this modus operandi would explain our ability to generalize algebraic rules:

Marcus (2001) p. 58

But in some cases we manage to extract a relationship between variables on the basis of training examples, without being given an explicit high-level description. Either way, when we learn a new function, we are presumably choosing among ways of combining some set of elementary instructions.

Critique

First, his claim that the human brain employs registers and basic operations on them is very bold, since he offers very little empirical data, relying instead on speculative arguments. Second, while Marcus gives some good arguments for his suggestion of using registers, it is still a bit unmotivated, especially with respect to the previous discussion. How do registers help a model to generalize UQOTOM? Marcus doesn't provide any explanation about such models and their behavior. The only purpose for his framework is to criticize the status quo, but he seems to have double standards for his own claims.

1.2.2 Case Studies

To substantiate his theoretical framework, Marcus concludes the chapter with two detailed case studies: the learning of artificial grammars by infants and the inflection of the English past tense. A brief examination of the latter reveals the core of his argument.

In the English past tense debate, Marcus argues that humans use a dual-route system: a default, symbolic rule for regular verbs and a separate associative memory for irregular exceptions. He critiques early connectionist models for attempting to handle both with a single mechanism. He famously pointed out that these models often produce erroneous blends because they don't treat the verb stem as a distinct variable, writing:

Marcus (2001) p. 78 f.

> In any case, swapping a process that operates over variables for a process that relates a regular verb and its past tense only in a piecemeal way results in a problem with blends. If there is no stem-copying process as such, nothing constrains the system to use the -ed morpheme only when the stem has been copied. Instead, whether the stem is transformed (as with an irregular) or copied (as with a regular) is an emergent property that depends on a set of largely independent, piecemeal processes. If the system learns that i sometimes changes to a, there is little to stop it from applying the i-a process at the same time as the process that causes -ed to appear at the end. The consequence is lots of blends, such as nick-nucked, that humans rarely if ever produce.

Critique

While Marcus correctly identifies the shortcomings of the specific, early connectionist models he analyzes, his conclusion that this proves the necessity of an explicit, rule-based system can be challenged. One could argue that rule-like behavior can emerge from a single, powerful pattern-learning mechanism without being explicitly programmed. The failures he highlights may not be fundamental flaws of the entire connectionist paradigm, but rather limitations of the specific, small-scale architectures of that era. Modern neural networks, operating at a much larger scale, are significantly more adept at learning complex, quasi-regular patterns.

Marcus proceeds by analyzing additional models, where he points out different strengths and weaknesses. However, a full analysis of these empirical arguments is beyond the scope of this paper, which has focused on the chapter's core theoretical claims. Marcus concludes his case studies with:

Marcus (2001) p. 68

As the summary given in table 3.2 makes clear, the few connectionist models that do not incorporate any sort of genuine operation over variables cannot capture our results, whereas all of the models that do implement operations over variables can capture our results.

1.3 Summary

In this chapter of Marcus's book *The Algebraic Mind*, the author established a new framework for assessing the capability of models to learn algebraic rules.

Marcus supports his claims with examples and empirical data. Nonetheless, the reader should remain critical of his framework and the arguments given. Central points of criticism are Marcus's reduction of algebraic rules to what he calls UQOTOMs, the consequences of his critique of backpropagation, and lastly his claims about registers and the human brain, which can be seen as unmotivated and bold.

While Marcus's approach is arguably too simplified for the complex nature of symbol manipulation and human cognition, his critique should not be overlooked. Even though his framework may lack some formal rigor, his main points—namely, the apparent misalignment between human cognitive biases and those of popular architectures like MLPs, as well as his appeal to look for alternatives—remain relevant today.