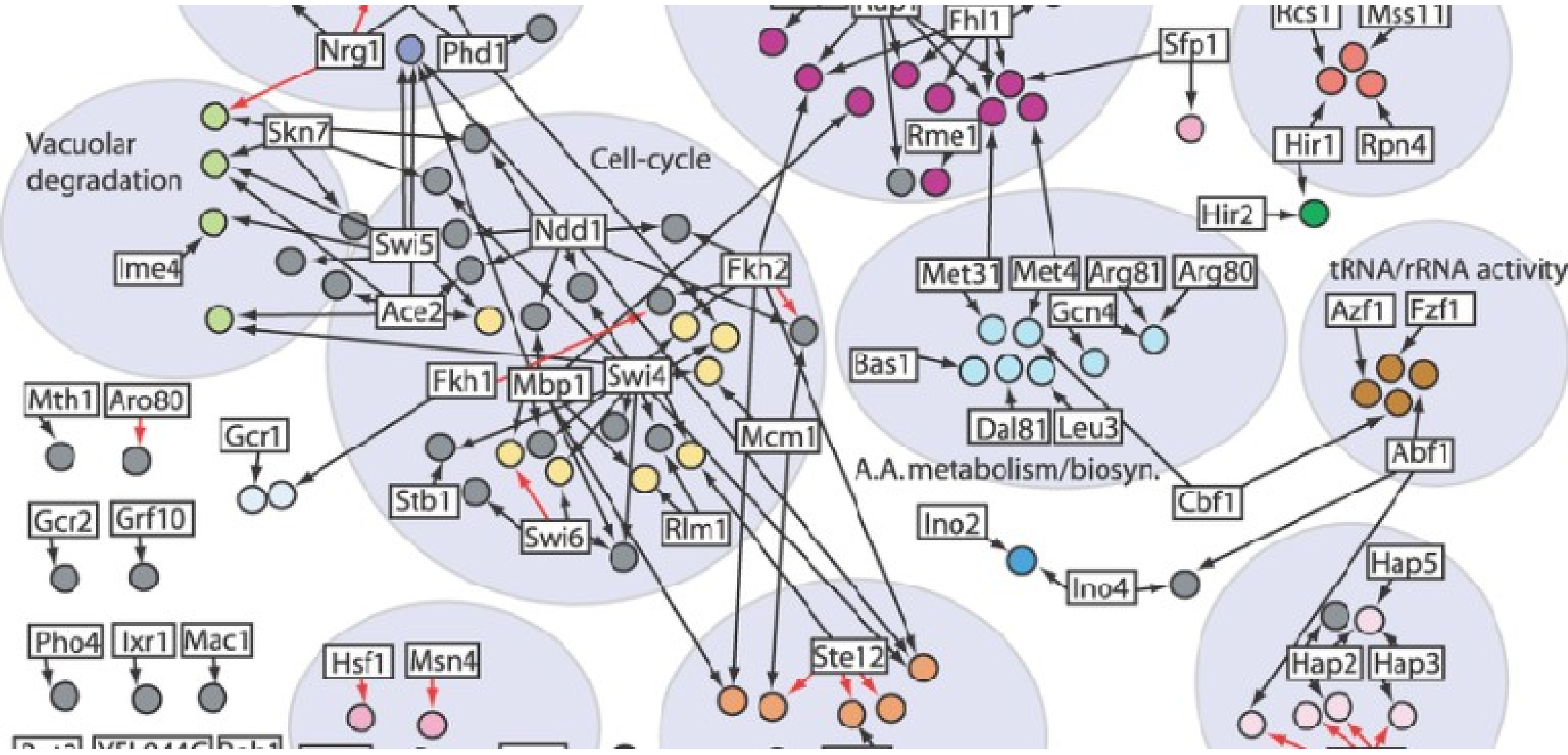


Nrg1
Phd1
Skn7
Vacuolar degradation
Ime4
Swi5
Ndd1
Cell-cycle
Ace2
Fkh2
Fkh1
Mbp1
Swi4
Mcm1
Stb1
Swi6
Rlm1
Mth1
Aro80
Gcr1
Gcr2
Grf10
Pho4
Ixr1
Mac1
Hsf1
Msn4
Ste12
Fhl1
Rme1
Sfp1
Rcs1
Mss11
Hir1
Rpn4
Hir2
Met31
Met4
Arg81
Arg80
Gcn4
Bas1
Dal81
Leu3
A.A. metabolism/biosyn.
Cbf1
tRNA/rRNA activity
Azf1
Fzf1
Abf1
Ino2
Ino4
Hap5
Hap2
Hap3

Selection Theorems: Modularity

Theories of Modularity in the Biological Literature

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Introduction

This post is part of a sequence describing our team's research on selection theorems for modularity, as part of this year's AI Safety Camp, under the mentorship of John Wentworth. Here, we provide some background reading for the discussion of modularity that will follow.

As we describe in more detail in our project intro, the motivating question of modularity that we started with (which is described in John's post on the Evolution of Modularity) is ***why does evolution seem to have produced modular systems (e.g. organs and organ systems), but current ML systems (even genetic algorithms, which are consciously fashioned off evolutionary mechanisms) are highly non-modular?*** So far, most of our research has focused on the idea of **modularly varying goals (MVG)**, but this is not the only proposed cause of modularity in the biological literature. This post serves as a literature review, with a few brief words on how we are thinking about testing some these hypotheses. Subsequent posts will discuss our team's research agenda in more detail.

Theories of modularity

Modularity in the environment selects for modular systems

Basic idea: If we have an environment which contains a variety of subproblems and which changes rapidly in highly modular ways, this might select for a modular internal design to better adapt to these problems. In the literature, this usually takes the form of "modularly varying goals" (MVG). If we vary one aspect of the agent's goal, but keep the rest of the goal and the environment constant, then this might produce a selection pressure for modular systems in correspondence with this modular goal structure.

Biological modularity: Many examples of modularity in biology seem to have a pretty clear correspondence with goals in the environment. For instance, type of terrain and availability of different food sources might vary somewhat independently between environments (or even within the same environment), and correspondingly we see the evolution of modular systems specifically designed to deal with one particular task (muscular system, digestive system). One can make a similar argument for reproduction, respiration, temperature regulation, etc.

Evidence: The main paper on this idea from the biological literature is Kashtan & Alon's 2005 paper. Their methodology is to train a system (they use both neural networks and genetic algorithms) to learn a particular logic function consisting of a series of logic gates, and after a certain number of steps / generations they vary one particular part of the logic gate setup, while keeping the rest the same. Their results were that modularity (as measured by the Q-score) was selected for, and the network motifs found in the evolved networks had a visible correspondence to the modular parts of the goal. We will discuss this paper in more detail in later sections of this post.

Testing this idea: We started by trying to replicate the results of the 2005 paper. It was somewhat challenging because of the now outdated conventions used in the paper, but our tentative conclusions for now are that the paper doesn't seem to replicate. Our plans if the replication was a success would have been to generalise to more complicated systems (one proposal was to train a CNN to recognise two digits from an MNIST set and perform an arithmetic operation on them, with the hope that it would learn a modular representation of both the individual digits and of the operator). However, this line of research has been put on hold until we get to the bottom of the null result from the Kashtan & Alon paper's replication.

For a fuller discussion of how we've been testing this idea (and some of our ideas as to why the replication failed), please see this post.

Specialisation drives the evolution of modularity

Basic idea: MVG can be viewed as a subcase of this theory, because the environment fluctuating in a modular way is one possible explanation for why selection might favour specialisation. However, it isn't the only explanation. Even in a static environment, evolution is a dynamic process, and a modular organism will be more easily able to evolve changes that improve its ability to deal with a particular aspect of its environment, without this affecting other modules (and hence having detrimental impacts on the rest of the organism). Hence specialisation (and hence modularity) might still be selected for.

Biological modularity: This theory was created partly from observation of a problem with MVG: not all environments seem to fluctuate in obviously modular ways (or even to fluctuate much at all). However, a lot of the biological intuitions of MVG carry over into this theory; namely the correspondence between different aspects of the environment that lend themselves well to specialisation, and different modular parts of the organism's internal structure.

Evidence: The paper Specialisation Can Drive the Evolution of Modularity looked into regulatory gene networks (simulated using genetic algorithms), and examined the conditions under which they form modules. It finds that the main driver of modularity is selection for networks with multiple stable gene activity patterns that have considerable overlap. In other words, many of the genes between the patterns are identical. However, this paper only examined one particular evolutionary setting, and it's unclear to which extent the results can be viewed as representative of a common principle of biological evolution. There don't seem to have been any follow-up studies on this research agenda.

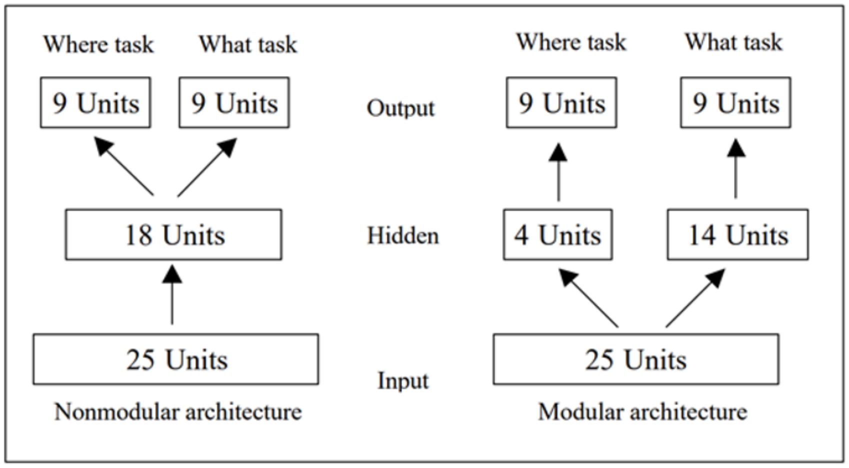
Testing this idea: We have no current plans to test this idea, since we think that MVG captures the core parts of it.

Direct selection for modularity, when it can perform a task / enable an adaptation that would be infeasible otherwise

Basic idea: Modularity may be selected for when it breaks a developmental constraint, and thereby makes some beneficial adaptation possible that would be virtually impossible otherwise.

Biological modularity: It's unclear how to look for evidence for this in the biological record, because it doesn't seem obvious how the world would be different depending on whether this was true or false. However, one form this could take that seems highly plausible in a biological context is evolving a particular architectural change at very early stages of ontogenesis, which enforces modularity (more on this in the "evidence" section).

Evidence: A well-known phenomenon in neural networks is that of "neural interference", whereby a system designed to learn a solution to two separate tasks inevitably develops connections between the two, despite these serving no practical purpose. As an example, consider a network performing image recognition on two different images, with the aim of outputting two different classifications: clearly no connections are required between the two tasks, but these will almost certainly be created anyway during the search for a solution, and they are hard to "un-learn". The 2000 paper Evolving Modular Architectures for Neural Networks explores the idea of both the network architecture and the parameters being genetically inherited - that way organisms can avoid this interference altogether.



However, it appears to us that this methodology may be problematic in that the researcher is deciding on the network structure before the study is even run. Looking at the above diagram, it's fairly obvious to us that the design on the right will perform better, but this study provides no evolutionary model as to how the modular architecture on the right might be selected for by evolution in the first place. It might seem intuitive to humans to choose the modular decomposition on the right, but for evolution to find that decomposition in particular is where all the work gets done, so this theory doesn't seem to present an actual causal explanation.

Testing this idea: If this hypothesis is correct, then there should be certain tasks that a non-modular network simply fails to solve, and so any network trained on this task will either find a modular solution or fail to converge. However, empirically this seems like it isn't the case (the question "why aren't the outputs of genetic algorithms modular" was the motivating one for much of this project). Whether this is because the hypothesis is false, or because ML is missing some key aspect of evolutionary dynamics, is unclear.

Modularity arises from a selection pressure to reduce connection costs in a network

Basic idea: If we apply selection pressure to reduce connection costs, this could perform a kind of regularisation that leads to a modular structure, where the only learned connections are the ones that are most important for performing a task. Furthermore, connection costs between different nodes in the network scaling with the "distance" between them might encourage a kind of "locality", which might further necessitate modularity.

Biological modularity: Despite being intuitively appealing, this idea looks a bit weaker when put in a biological context. There are some important examples of connections being costly (e.g. the human brain), but for this to be the driving factor leading to modularity, we should expect to see connection costs playing a leading role in virtually all forms of signalling across all organisms where modularity is to be found, and this doesn't always seem to be the case (e.g. signalling molecules in bacteria). However, it's possible this factor contributes to modularity along with other factors like MVG, even though it's not the driving factor (e.g. see results from the paper below).

The locality argument seems a lot more robust to biological evidence. Where there is modularity, we very often see it structured in a local way, with networks of cells performing a specific task being clustered together into a relatively small region of space (e.g. organs). Cells interact with each other in a highly localised way, e.g. through chemical and mechanical signals. This means that any group of cells performing a task which requires a lot of inter-cellular interaction is size-limited, and may have to learn to perform a specialised task.

This argument also helps explain why modularity might be observable to humans. Consider a counterfactual world in which locality didn't apply: there would be very little visible commonality between different organisms, and so even if modularity was present we would have a hard time identifying it (Chris Olah uses a similar idea as an analogy when discussing the universality claim for circuits in neural networks).

Evidence: The 2013 paper The evolutionary origins of modularity explores this idea. Their methodology was to compare two different networks: "performance alone" (PA) and "performance and connection costs" (P&CC). The task was a form of image recognition the authors have called the "retina problem": the networks were presented with an 8-pixel array separated into two 2x2 blocks (left and right), and tasked to decide whether a certain type of object appeared in each block. This task exhibited a natural modular decomposition. It could also be varied in a modular way: by changing the task from deciding whether an image appeared in both blocks ("L-AND-R environment") to whether an image appeared in either block ("L-OR-R"). In this way, they could test pressure to reduce costs against the MVG hypothesis. The results were that the P&CC networks were both more modular than the PA networks, and significantly outperformed them after a finite number of iterations, but that the best PA networks outperformed the best P&CC networks (explaining why performance alone might not be sufficient to select for modularity). They also found that adding MVG further increased both the outperformance and the modularity.

Another paper exploring this hypothesis is Computational Consequences of a Bias toward Short Connections (1992). This paper focuses on architectural constraints selecting for fewer/shorter connections, rather than imposing an explicit penalty. However, it doesn't use particularly advanced machine learning methods, and focuses more narrowly on the human brain than on general modularity across evolved systems, making it less relevant for our purposes.

Testing this idea: We plan to introduce this type of selection pressure in our replication of the Kashtan & Alon 2005 paper. That way we can test it against the MVG hypothesis, and see whether the two combined are enough to produce modularity. We will experiment with different cost measures, although to the extent that this is a strong causal factor leading to modularity, we would expect the results to be relatively robust to different ways of calculating connection costs (this was also what the 2013 paper found).

We also have plans to test the locality idea, by instantiating neurons in some kind of metric space (e.g. 2D Euclidean space, or just using some kind of indexing convention), and penalising connection costs proportionally to their distance.

For our team's most recent post, please see here .

Project Intro: Selection Theorems for Modularity

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

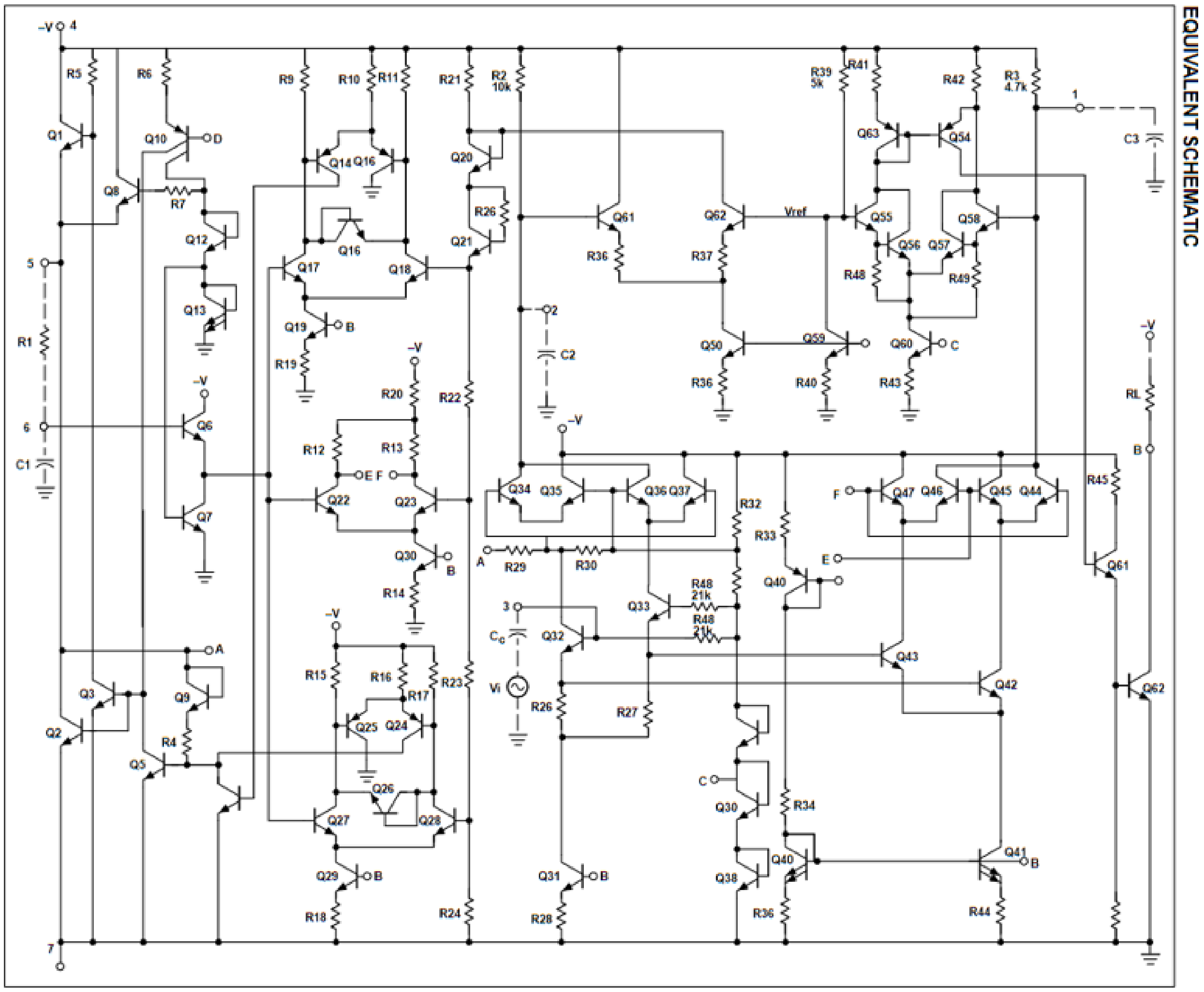
Introduction - what is modularity, and why should we care?

It’s a well-established meme that evolution is a blind idiotic process, that has often resulted in design choices that no sane systems designer would endorse. However, if you are studying simulated evolution, one thing that jumps out at you immediately is that ***biological systems are highly modular, whereas neural networks produced by genetic algorithms are not***. As a result, the outputs of evolution often look more like something that a human might design than do the learned weights of those neural networks.

Humans have distinct organs, like hearts and livers, instead of a single heartliver. They have distinct, modular sections of their brains that seem to do different things. They consist of parts, and the elementary neurons, cells and other building blocks that make up each part interact and interconnect more amongst each other than with the other parts.

Neural networks evolved with genetic algorithms, in contrast, are pretty much uniformly interconnected messes. A big blob that sort of does everything that needs doing all at once.

Again in contrast, networks in the modern deep learning paradigm apparently do exhibit some modular structure.



Top: Yeast Transcriptional Regulatory Modules - clearly modular
Bottom: Circuit diagram evolved with genetic algorithms - non-modular mess

Why should we care about this? Well, one reason is that modularity and interpretability seem like they might be very closely connected. Humans seem to mentally subpartition cognitive tasks into abstractions, which work together to form the whole in what seems like a modular way. Suppose you wanted to figure out how a neural network was learning some particular task, like classifying an image as either a cat or a dog. If you were explaining to a human how to do this, you might speak in terms of discrete high-level concepts, such as face shape, whiskers, or mouth.

How and when does that come about, exactly? It clearly doesn't always, since our early networks built by genetic algorithms work just fine, despite being an uninterpretable non-modular mess. And when networks *are* modular, do the modules correspond to human understandable abstractions and subtasks?

Ultimately, if we want to understand and control the goals of an agent, we need to answer questions like "what does it actually mean to explicitly represent a goal", "what is the type structure of a goal", or "how are goals connected to world models, and world models to abstractions?"

It sure feels like we humans somewhat disentangle our goals from our world models and strategies when we perform abstract reasoning, even as they point to latent variables of these world models. Does that mean that goals inside agents are often submodules? If not, could we understand what properties real evolution, and some current ML training exhibit that select for modularity and use those to *make* agents evolve their goals as submodules, making them easier to modify and control?

These questions are what we want to focus on in this project, started as part of the 2022 AI Safety Camp.

The current dominant theory in the biological literature for what the genetic algorithms are missing is called **modularly varying goals (MVG)**. The idea is that modular changes in the environment apply selection pressure for modular systems. For example, features of the environment like temperature, topology etc might not be correlated across different environments (or might change independently within the same environment), and so modular parts such as thermoregulatory systems and motion systems might form in correspondence with these features. This is outlined in a paper by Kashtan and Alon from 2005, where they aim to demonstrate this effect by modifying loss functions during training in a modular way.

We started off by investigating this hypothesis, and trying to develop a more gears-level model of how it works and what it means. This post outlines our early results.

However, we have since branched off into a broader effort to understand selection pressure for modularity in all its forms. The post accompanying this one is a small review of the biological literature, presenting the most widespread explanations for the origins of modularity we were able to find. Eventually, we want to investigate all of these, come up with new ones, expand them to current deep learning models and beyond, and ultimately obtain a comprehensive selection theorem(s) specifying when exactly modularity arises in any kind of system.

A formalism for modularity

Modularity in the agent

For this project, we needed to find a way of quantifying modularity in neural networks. The method that is normally used for this in the biological literature (including the Kashtan & Alon paper mentioned above), and in papers by e.g. CHAI dealing with identifying modularity in deep modern networks, is taken from graph theory. It involves the measure Q, which is defined as follows:

$$Q = \frac{1}{(2m)} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{(2m)} \right] \delta(c_v, c_w)$$

Where the sum is taken over nodes, Avw is the actual number of (directed) edges from v to w, kv are the degrees, and δ(cv,cw) equals 1 if v and w are in the same module, 0 otherwise. Intuitively, Q measures “the extent to which there are more connections between nodes in the same modules than what we would expect from random chance”.

To get a measure out of this formula, we search over possible partitions of a network to find the one which maximises Qm, where Qm is defined as a normalised version of Q:

$$Q_{\mathrm{m}} = (Q_{\mathrm{real}} - Q_{\mathrm{rand}})/(Q_{\max} - Q_{\mathrm{rand}}),$$

where Qrand is the average value of the Q-score from random assignment of edges. This expression Qm is taken as our measure of modularity.

This is the definition that we’re currently also working with, for lack of a better one. But it presents obvious problems when viewed as a general, True Name-type definition of modularity. Neural networks are not weighted graphs. So to use this measure, one has to choose a procedure to turn them into one. In most papers we’ve seen, this seems to take the form of translating the parameters into weights by some procedure, ranging from taking e.g. the matrix norm of convolutions, or measuring the correlation or coactivation of nodes. None of the choices seemed uniquely justified on a deep level to us.

To put it differently, this definition doesn’t seem to capture what we think of when we think of modularity. “Weighted number of connections” is certainly not a perfect proxy for how much communication two different subnetworks are actually doing. For example, two connections to different nodes with non-zero weights can easily end up in a configuration where they always cancel each other out.

We suspect that ***since network-based agents are information-processing devices, the metric for modularity we ultimately end up using should talk about information and information-processing, not weights or architectures***. We are currently investigating measures of modularity that involve mutual information, but there are problems to work out with those too (for instance, even if two parts of the network are not communicating at all and perform different tasks, they will have very high mutual information if it is possible to approximately recreate the input from their activations).

Modularity in the environment

If we want to test whether modular changes in the environment reliably select for modular internal representations, then we first need to define what we mean by modular changes in the environment. The papers we found proposing the Modularly Varying Goals idea didn’t really do this. It seemed more like an “we know it when we see it” kind of thing. So we’ve had to invent a formalism for this from the ground up.

The core idea that we started with was this: a sufficient condition for a goal to be modular is that ***humans*** **tend to think about / model it in a modular way.** And since humans are a type of neural network, that’s suggestive that the design space of networks dealing with this environment, out in nature or on our computers, include designs that model the goal in a modular way too.

Although this is sufficient, it shouldn’t be necessary - just because a human looking at the goal can’t see how to model it modularly doesn’t mean no such model exists. That led to the idea that a modularly varying goal should be defined as one for which **there exists some network in design space that performs reasonably well on the task, and that “thinks” about it in a modular way.**

Specifically, if for example you vary between two loss functions in some training environment, L1 and L2, that variation is called “modular” if somewhere in design space, that is, the space formed by all possible combinations of parameter values your network can take, you can find a network N1 that “does well”(1) on L1, and a network N2 that “does well” on L2, and these networks have the same values for all their parameters, except for those in a single(2) submodule(3).

(1) What loss function score corresponds to “doing well”? Eventually, this should probably become a sliding scale where the better the designs do, the higher the “modularity score” of the environmental variation. But that seems complicated. So for now, we restrict ourselves to heavily overparameterized regimes, and take doing well to mean reaching the global minimum of the loss function.

(2) What if they differ in n>1 submodules? Then you can still proceed with the definition by considering a module defined by the n modules taken together. Changes that take multiple modules to deal with are a question we aren’t thinking about yet though.

(3) Modularity is not a binary property. So eventually, this definition should again include a sliding scale, where the more encapsulated the parameters that change are from the rest of the network, the more modular you call the corresponding environmental change. But again, that seems complicated, and we’re not dealing with it for now.

One thing that’s worth discussing here is why we should expect modularity in the environment in the first place. One strong argument for this comes from the Natural Abstraction Hypothesis, which states that our world abstracts well (in the sense of there existing low-dimensional summaries of most of the important information that mediates its interactions with the rest of the world), and these abstractions are the same ones that are used by humans, and will be used by a wide variety of cognitive designs. This can very naturally be seen as a statement about modularity: these low-dimensional summaries precisely correspond to the interfaces between parts of the environment that can change modularly. For instance, consider our previously-discussed example of the evolutionary environment which has features such as temperature, terrain, etc. These features correspond to both natural abstractions with which to describe the environment, and precisely the kinds of modular changes which (so the theory goes) led to modularity through the mechanism of natural selection.

This connection also seems to point to our project as a potential future test bed of the natural abstraction hypothesis.

Why modular environmental changes lead to network modularity

Once the two previous formalisms have been built up, we will need to connect them somehow. The key question here is why we should expect modular environmental changes to select for modular design at all. Recall that we just defined a modular environmental change as one for which there *exists* a network that decomposes the problem modularly - this says nothing about whether such a network will be selected for.

Our intuition here comes from the curse of dimensionality. When the loss function changes from L1 to L2, how many of its parameters does a generic perfect solution for L1 need to change to get a perfect loss on L2?

A priori, it'd seem like potentially almost all of them. For a generic interconnected mess of a network, it's hard to change one particular thing without also changing everything else.

By contrast, the design N1, in the ideal case of perfect modularity, would only need to change the parameters in a single one of its submodules. So if that submodule makes up e.g. 0.1 of the total parameters in the network, you've just divided the number of dimensions your optimiser needs to search by ten. So the adaptation goes a lot faster.

"But wait!" You say. Couldn't there still be non-modular designs in the space that *just so happen* to need even less change than N1 type designs to deal with the switch?

Good question. We wish we'd seriously considered that earlier too. But bear with it for a while.

No practically usable optimiser for designing generic networks currently known seems to escape the curse of dimensionality fully, nor does it seem likely to us that one can exist[1], so this argument seems pretty broadly applicable. Some algorithms scale better/worse with dimensionality in common regimes than others though, and that might potentially affect the magnitude of the MVG selection effect.

For example, in convex neighbourhoods of the global optimum, gradient descent is roughly bounded by exact line search to scale with the condition number of the Hessian. How fast the practically relevant condition number of a generic Hessian grows with its dimension D is a question that's been proving non-trivial to answer for us, but between the circular law and a random paper on the eigenvalues of symmetric matrices we found, we suspect it might be something on the order of $O(\sqrt{D}) - O(D)$, maybe. (Advice on this is very welcome!)

By contrast, if gradient descent is still outside that convex neighbourhood, bumbling around the wider parameter space and getting trapped in local optima and saddle points, the search time probably scales far worse with the number of parameters/dimensions. So how much of an optimization problem is made up of this sort of exploration vs. descending the convex slope near the minimum might have a noticeable impact on the adaptation advantage provided by modularity.

Likewise, it seems like genetic algorithms could scale differently with dimensionality and thus this type of modularity than gradient based methods, and that an eventual future replacement of ADAM might scale better. These issues are beyond the scope of the project at this early stage, but we're keeping them in mind and plan to tackle them at some future point.

(Unsuccessfully) Replicating the original MVG paper

The Kashtan and Alon (2005) paper, seems to be one of the central results backing the modularly varying goals (MVG) hypothesis as a main driving factor for modularity in evolved systems. It doesn’t try to define what modular change is, or mathematically justify why modular designs might handle such change better, the way we tried above. But it does claim to show the effect in practice. As such, one of the first things we did when starting this project was try to replicate that paper. **As things currently stand, the replication has mostly failed.**

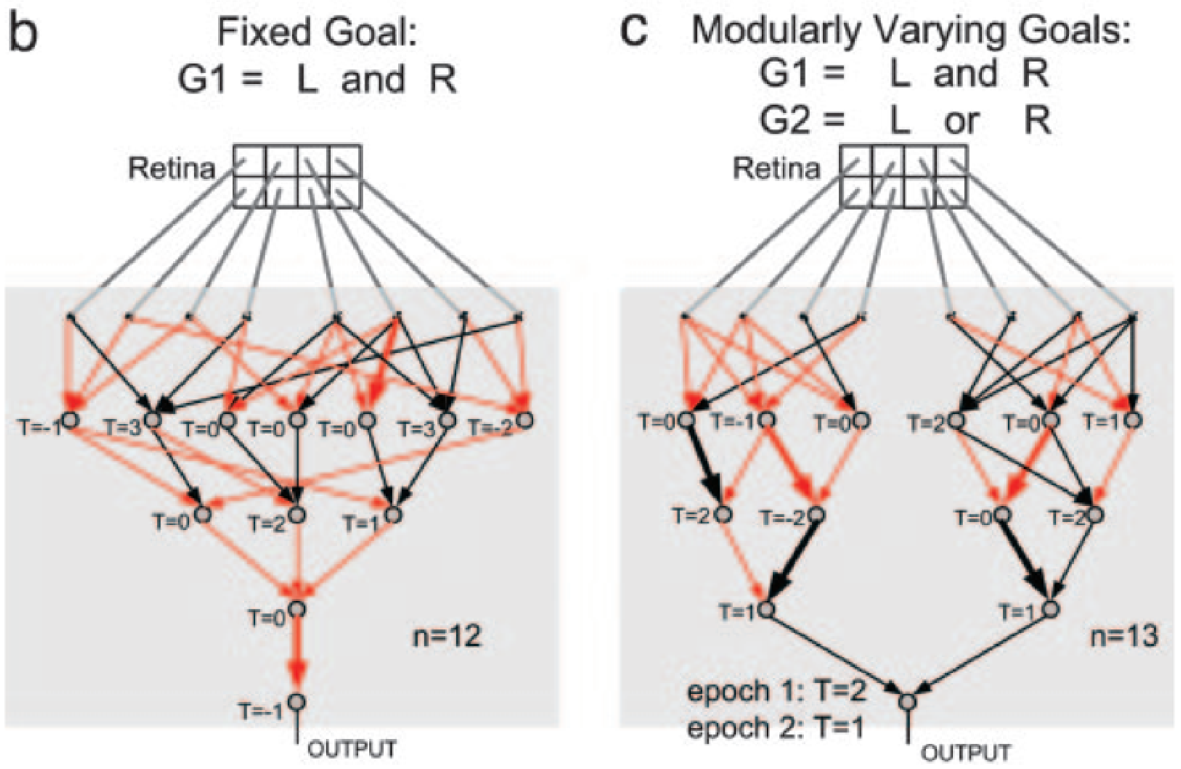
We say “mostly” for two reasons. First, there are two separate experiments in the 2005 paper supposedly demonstrating MVG. One involving logic circuits, and one using (tiny) neural networks. We ignored the logic circuits experiment and focused solely on the neural network one as it was simpler to implement. Since that replication failed, we plan on revisiting the logic circuits experiment.

Second, we implemented some parts of the 2005 paper with liberty. For example, Kashtan & Alon used a specific selection strategy when choosing parents in the evolutionary algorithm used to train the networks. We used a simpler strategy (although, we did try various strategies). The exact strategy they used didn’t seem like it should be relevant for the modularity of the results.

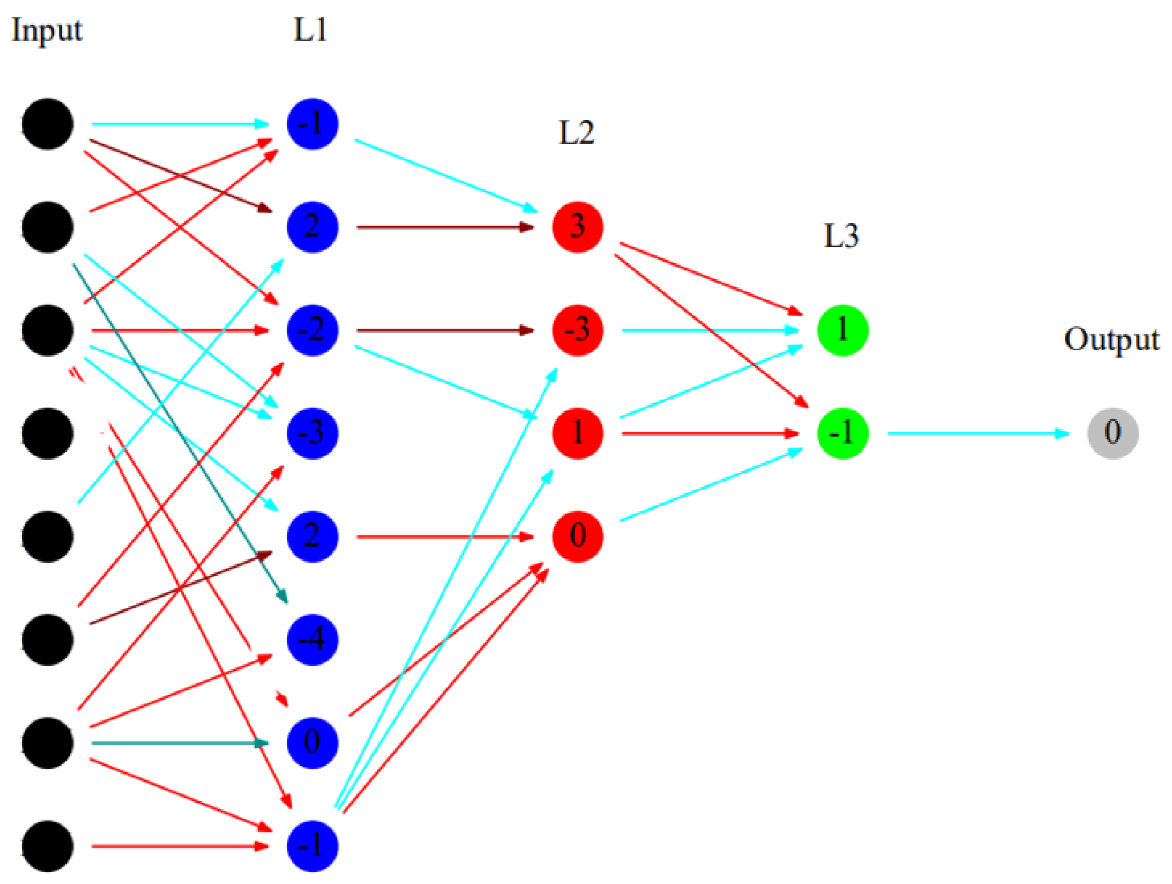
We are currently working on a more exact replication just to be sure, but if MVG is as general an effect as the paper is speculating it to be, it really shouldn’t be sensitive to small changes in the setup like this.

The idea of the NN experiment was to evolve neural networks (using evolutionary algorithms, not gradient descent methods) to recognize certain patterns found in a 4-pixel-wide by 2-pixel-high retina. There can be “Left” (L) patterns and “Right” (R) patterns in the retina. When given the fixed goal of recognizing “L and R” (e.g. checking whether there is both a left and a right pattern), networks evolved to handle this task were found to be non-modular. But when alternating between two goals: G1=“L and R” and G2=“L or R”, that are swapped out every few training cycles, the networks supposedly evolved to converge on a very modular, human understandable design, with two modules recognising images on one side of the screen each. The “and” and “or” is then handled by a small final part taking in the signal from both modules, and that’s the only bit that has to change whenever a goal switch occurs.

Here are examples of specific results from Kashtan & Alon:



Unfortunately, our replication didn't see anything like this. Networks evolved to be non-modular messes both for fixed goals and Modularly Varying Goals:



We got the same non-modular results under a wide range of experimental parameters (different mutation rates, different selection strategies, population sizes, etc). If anyone else has done their own replication, we'd love to hear about it.

So what does that mean?

So that rather put a damper on things. Is MVG just finished now, and we should move on to testing the next claimed explanation for modularity?

We thought that for a while. But as it turns out, no, MVG is not quite done yet.

There's a different paper from 2013 with an almost identical setup for evolving neural networks. Same retina image recognition task, same two goals, everything. But rather than switching between the two goals over and over again to try and make the networks evolve a solution that's modular in their shared subtasks, they just let the networks evolve in a static environment using the first goal. Except they also apply an explicit constraint on the number of total connections the networks are allowed to have, to encourage modularity, as per the connection cost explanation.

They say that this worked. Without the connection cost, they got non-modular messes. With it, they got networks that were modular in the same way as those in the 2005 paper. Good news for the connection cost hypothesis! But what's that got to do with MVG?

That's the next part: When they performed a single, non-repeated switch of the goal from "and" to "or", the modular networks they evolved using connection costs dealt faster with the switch than the non-modular networks. Their conclusion is that modularity originally evolves because of connection costs, but does help with environmental adaptation as per MVG.

So that's when we figured out what we might've been missing in our theory work. You can have non-modular networks in design space that deal with a particular change *even better* than the modular ones, just by sheer chance. But if there's nothing selecting for them specifically over other networks that get a perfect loss, there's no reason for them to evolve.

But if you go and perform a specific change to the goal function over and over again, you might be selecting exactly those networks that are fine tuned to do best on that switch. Which is what the 2005 paper and our replication of it were doing.

So modular networks are more generally robust to modular change, but they might not be as good as dealing with any *specific* modular change as a non-modular, fine tuned mess of a network constructed for that task.

Which isn't really what "robustness" in the real world is about. It's supposed to be about dealing with things you haven't seen before, things you *didn't* get to train on.

(Or alternatively, our replication of the 2005 setup just can't find the modular solutions, period. Distinguishing between that hypothesis and this one is what we're working on on the experimental front right now.)

To us, this whole train of thought[2] has further reinforced the connection that modularity = abstraction, in some sense. Whenever you give an optimiser perfect information on its task in training, you get a fine-tuned mess of a network humans find hard to understand. But if you want something that can handle unforeseen, yet in some sense *natural* changes, something modular performs better.

How would you simulate something like that? Maybe instead of switching between two goals with shared subtasks repeatedly, you switch between a large set of random ones, all still with the same subtasks, but never show the network the same one twice[3]. We call this Randomly sampled Modularly Varying Goals (RMVG).

Might a setup like this select for modular designs directly after all, without the need for connection costs? If the change fine-tuning hypothesis is right, all the fine-tuned answers would be blocked off by this setup, leaving only the modular ones as good solutions to deal with the change. Or maybe the optimiser just fails to converge in this environment, unable to handle the random switches. We plan to find out.

Though before we do that, we're going to replicate the 2013 paper. Just in case.

Future plans

On top of the 2013 replication, the exact 2005 replication and RMVG, we have a whole bunch of other stuff we want to try out.

- **Sequential modularity instead of parallel modularity**
  If you have a modular goal with two subtasks, but the second depends on the first, does the effect still work the same way?

- **Large networks**
  So far, our discussion and experimentation has been based on pre-modern neural networks with less than twenty nodes. How would this all look in the modern paradigm, with a vastly increased number of parameters and gradient-type optimisers instead of genetic algorithms?
  To find out, we plan to set up a MVG-test environment for CNNs. One possible setup might be needing to recognise two separate MNIST digits, then perform various joined algebraic operations with them, like e.g. addition or subtraction).

- **Explaining modularity in modern networks**
  To follow on from the previous point, it's worth noting that modularity is apparently already present by default in CNNs and other modern architectures, unlike the old school bio simulations that didn't see modularity unless they specifically try to select for it. This begs the question, can we explain why it is there? Do these architectures somehow enforce something equivalent to a connection cost? Does using SGD introduce an effective variation in the goal function that is kind of like (R)MVG? Does modularity become preferred once you have enough parameters, because the optimiser somehow can't handle fine-tuning interaction between all of them? Or do we need new causes of modularity to account for it?

- **Other explanations for modularity**
  On that note, we also plan to do a bunch more research and testing on the other proposed explanations for modularity in the bio literature[4], plus any we might come up with ourselves.
  Regarding connection costs for example, we've been wondering whether the fact that physics in the real world is local, while the nodes in our programs are not[5], plays an additional role here. We plan to compare flat total connection costs to local costs, which penalise more distant nodes more than closer ones for connecting.

- **Finding the "true name" of modularity**
  Then there's the work of designing a new, more deeply justified modularity measure based on information exchange. If we can manage that, we should check all our experiments again, and see if it changes our conclusions anywhere. For that matter, maybe it should be used on some of the papers that found modularity in modern networks, all of which seem to have used the graph theory measure.

- **Broadness of modular solutions**
  Here's another line of thought: it's a widespread hypothesis in ML that generalisable minima are broader than fine-tuned ones. Since our model of MVG says that modular designs are more adaptable, general ones, do they have broader minima? We're going to measure that in the retina NN setup very soon. We're also going to research this hypothesis more, because we don't feel like we really understand it yet.

- **"Goal modules" in reinforcement learning**
  And finally, if the whole (R)MVG stuff and related matters work out the way we want them to, we've been wondering if they could form the basis of a strategy to make a reinforcement learner develop a specific "goal module" that we can look at. Say we have an RL agent in a little 2D world, which it needs to navigate to achieve some sort of objective. Now say we vary the objective in training, but have each one rely on the same navigation skill set. Maybe add some additional

pressure for modularity on top of that, like connection costs. Could that get us an agent with an explicit module for slotting in the objectives? Or maybe we should do it the other way around, keeping the goal fixed while modularly varying what is required to achieve it. That's many months away though, if it turns out to be feasible at all. It's just a vague idea for now.

This is a lot, and maybe it seems kind of scattered, but we are in an exploratory phase right now. At some point, we'd love to see this all coalesce into just a tiny few unifying selection theorems for modularity.

Eventually, we hope to see the things discovered in this project and others like it form the concepts and tools to base a strategy for creating aligned superintelligence on. As one naive example to illustrate how that might work, imagine we're evolving an AGI starting in safe, low capability regimes, where it can't really harm us or trick out our training process, and remains mostly understandable to our transparency tools. This allows us to use trial and error in aligning it.

Using our understanding of abstraction, agent signatures and modularity, we then locate the agent's (or the subagents') goals inside it (them), and "freeze" the corresponding parameters. Then we start allowing the agent(s) to evolve into superintelligent regimes, varying all the parameters except these frozen ones. Using our quantitative understanding of selection, we shape this second training phase such that it's always preferred for the AGI to keep using the old frozen sections as its inner goal(s), instead of evolving new ones. This gets us a superintelligent AGI that still has the aligned values of the old, dumb one.

Could that actually work? We highly doubt it. We still understand agents and selection far too little to start outlining actual alignment strategies. It's just a story for now, to show why we care about foundations work like this. But we feel like it's pointing to a class of things that could work, one day.

Final words

Everything you see in this post and the accompanying literature review was put together in about ten weeks of part-time work. We feel like the amount of progress made here has been pretty fast for agent foundations research by people who've never done alignment work before. John Wentworth has been very helpful, and provided a lot of extremely useful advice and feedback, as have many others. But still, it feels like we're on to something.

If you're reading this, we strongly suggest you take a look at the selection theorems research agenda and related posts, if you haven't yet. Even if you've never done alignment research before. Think about it a little. See if anything comes to mind. It might be a very high-value use of your time.

The line of inquiry sketched out here was not the first thing we tried, but it was like, the second to third-ish. That's a pretty good ratio for exploratory research. We think this might be because the whole area of modularity, and selection theorems more generally, has a bunch of low hanging research fruit, waiting to be picked.

For our team's most recent post, please see here.

1. Or at least if one is found, it’d probably mean P=NP, and current conceptions of AGI would become insignificant next to our newfound algorithmic godhood. ↩

2. Which might still be wrong, we haven’t tested this yet. ↩

3. Or just rarely enough to prevent fine tuning. ↩

4. More discussion of the different explanations for biological modularity found in the bio literature can be found in this post. ↩

5. Computer chip communication lines are though. Might that be relevant if you start having your optimisers select for hardware performance? ↩

Ten experiments in modularity, which we'd like you to run!

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

This is the third post describing our team's work on selection theorems for modularity, as part of a project mentored by John Wentworth (see here for the earlier posts). Although the theoretical and empirical parts of the project have both been going very well, we're currently bottlenecked on the empirical side: we have several theories and ideas for how to test them, but few experimental results. Right now, we only have one empiricist coding up experiments, so this overhang seems likely to persist.

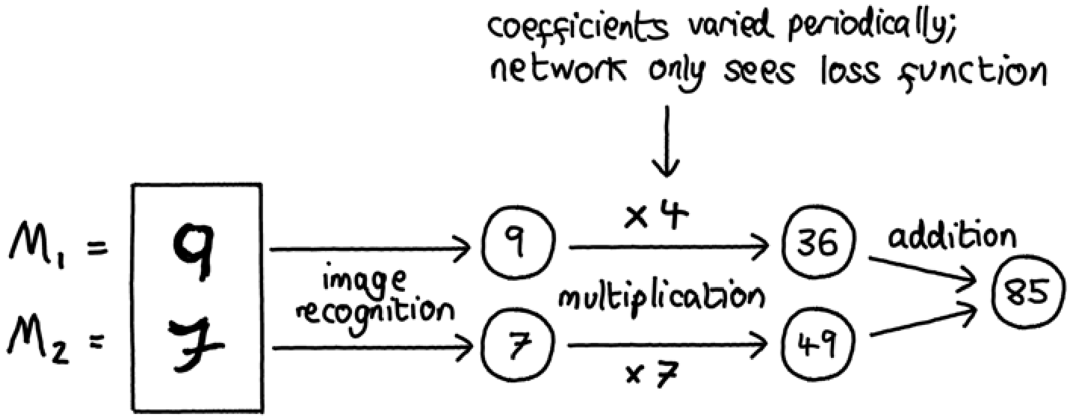
The purpose of this post is to outline some of our ideas for experiments. We hope that this will provide concrete steps for people who are interested in engaging with empirical research in AI safety, or on selection theorems in particular, to contribute to this area.

Experiments

1. Investigate (randomly) modulary varying goals in modern deep learning architectures.

In our previous post, we discussed the idea of modularly varying goals in more detail. We conjectured that rapidly switching between a small set of different goals might fine-tune the network to deal with this specific switch, rather than providing selection pressure for modularity. One possible solution to this is RMVG (randomly-sampled modularly varying goals), where we switch between a large set of random goals, all still with the same subtasks, but have a large enough set that we never show the network the same task twice.

For example, one could build a CNN to recognise two MNIST digits $\mathrm{M}_1, \mathrm{M}_2$, then perform some algebraic operation on the recognised digits (e.g. addition), and measure the modularity of the zero-training-loss solutions found by ADAM (e.g. using the graph theory measure of modularity with the matrix norm of the CNN kernels as weights - see these CHAI papers for more ideas and discussion). To introduce RMVG, you might switch the task during training from calculating $\mathrm{M}_1 + \mathrm{M}_2$ to $\mathrm{a} \times \mathrm{M}_1 + \mathrm{b} \times \mathrm{M}_2$, where a, b are real numbers, varied periodically after some fixed number of epochs. The network wouldn't get a and b as direct inputs; it could only access them indirectly via their effect on the loss function. This task is modular in the sense that it can be factored into the separate tasks of *"recognising two separate digits"* and *"performing joint arithmetical operations on them"*, and we might hope that varying the goal in this way causes the network to learn a corresponding modular solution.



Questions you might ask:

- Does this way of varying goals lead to more modular solutions than the fixed goal of $a \times M_1 + b \times M_2$?
- Can you think of any other modular goals which you could run a similar test for? Are there some types of modular goals which produce modularity more reliably?
- Does switching training methods (e.g. ADAM to SGD or GD) change anything about the average modularity score of solutions?

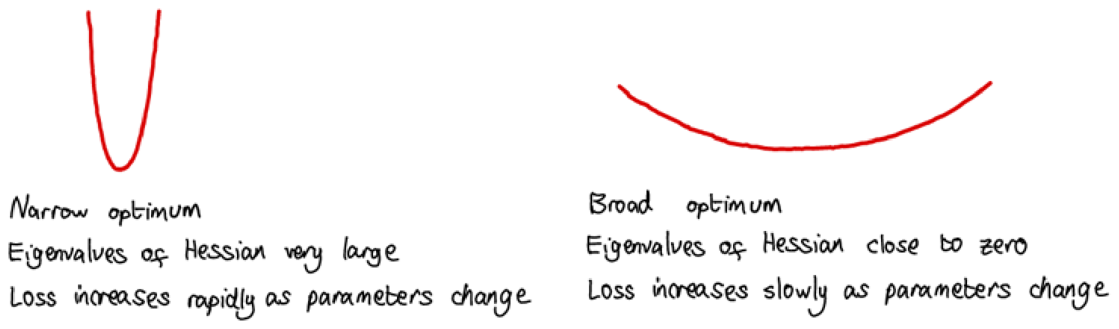
2) Measure the broadness of modular and non-modular optima in deep learning architectures.

We have some theories that predict modular solutions for tasks to be on average broader in the loss function landscape than non-modular solutions. One could test this by making a CNN and getting it to produce some zero training loss solutions to a task, some of which were more modular than others (see experiment (1) for more detail on this, and (4) for another proposed method to produce modularity), then vary the parameters of these optima slightly to see how quickly the loss increases.

We've recently done experiments with simple networks and the retina problem, and this intuition seems to hold up. But more replications here would be very valuable, to examine the extent to which the hypothesis is borne out in different situations / task / network sizes.

Questions you might ask:

- Is there a relationship between broadness of optima and modularity score?
- Do different kinds of tasks, or different sizes of networks, generally result in broader optima?


Narrow optimum
Eigenvalues of Hessian very large
Loss increases rapidly as parameters change
Broad optimum
Eigenvalues of Hessian close to zero
Loss increases slowly as parameters change

3) Attempt to replicate the modularly varying goals experiment with logic gates in the original Kashtan & Alon 2005 paper.

We've focused on the neural network retina recognition task in the paper so far, since it's more directly relevant to ML. But since that result so far hasn't really replicated the way the paper describes (see our previous post), it would be important for someone to check if the same is true of the logic gate experiment.

Kashtan still has the original 2005 paper code, and seems happy to hand it out to anyone who asks. He also seems happy to provide advice on how to get it running and replicate the experiment. Please don't hesitate to email him if you're interested.

4) Investigate the effect of local / total connection costs on modularity in neural networks

Real biology highly modular. One of the big differences between real biology and neurons in our ML models is that connecting things in the real world is potentially costly and difficult. Lots of papers seem to suggest that this plays a role in promoting modularity (we will probably have a post about this coming out over the next few weeks). If you introduce a penalty for having too many connections into your loss function, this tends to give you more modular solutions.

But in real biology, the cost of forming new connections doesn't only depend on the total number of connections, but also how physically distant the things you are connecting are. One way you could replicate that kind of connection cost on a computer might be to give each node in your network a 1D or 2D "position index", and penalise connections between nodes depending on the L2 distance between their indices (e.g. see this paper for one example way of approaching this problem).

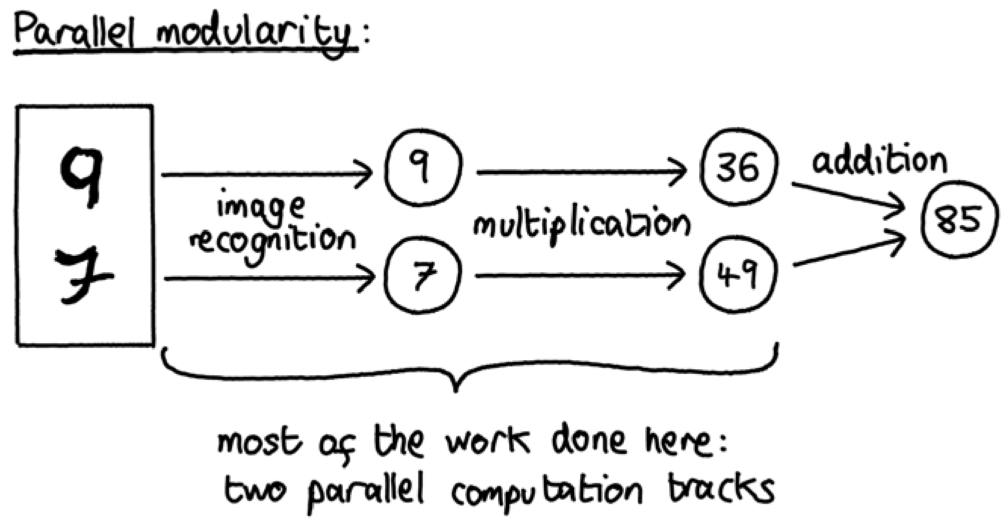
Questions you might ask:

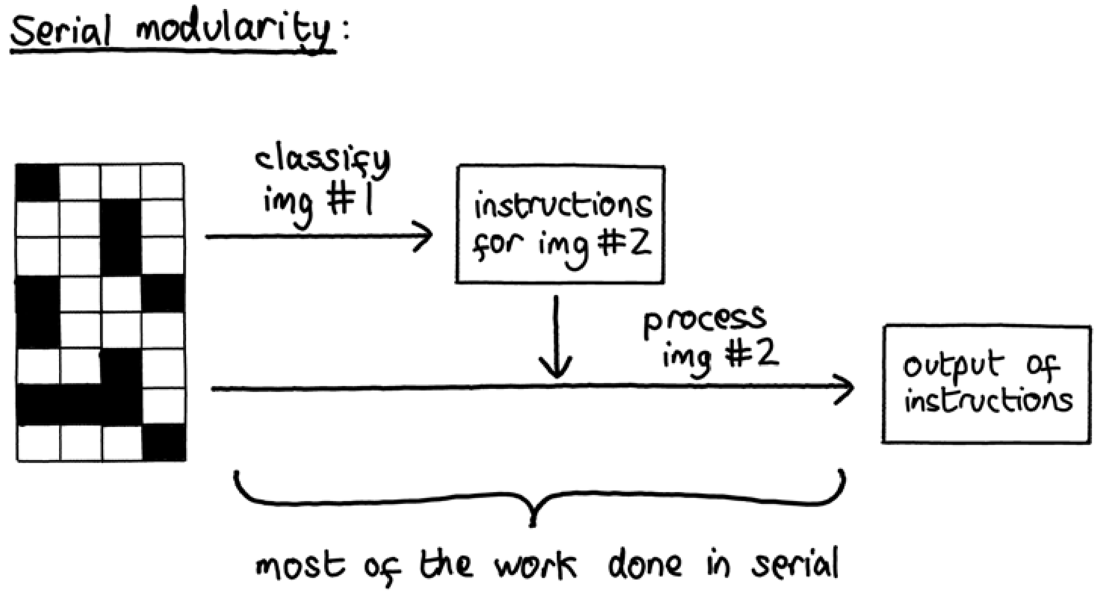
- Do the results in the paper linked above replicate?
- As a baseline comparison, how does the performance / modularity of solutions change when you use the standard "total" connection cost, rather than a locally weighted one?
- Do the results depend on *how* you position the neurons in your space? If so, are there some positioning methods that lead to more modular solutions?
- How do these results change when you look at bigger networks?

5) Implement a modularity / MVG experiment for serial, rather than parallel, subtasks.

Our current MVG experimental setups involve the retina recognition task, as well as the CNN MNIST experiments proposed above. These all have a structure where you'd expect a modular solution to have two modules handling different subtasks in parallel, the outputs of which are needed to perform a small, final operation to get the output.

It would be interesting to see what happens in a case where the subtasks are serial instead. For example, for a "retina" task like the one in the Kashtan 2005 paper, the loss function might require classifying patterns on the "left" retina of the input in order to decide what to do with the "right" retina of the input. Or for a CNN setup with two input images, the ask might be to recognise something in the first image which then determines what you should look for in the second image.





Questions you could ask:

- Which kinds of modular tasks are more naturally described as serial rather than parallel?
- Do you get more modularity in the serial rather than parallel case? Is it harder to find solutions?
- In serial tasks, is the effect of MVG different if you are changing just the (chronologically) first task, or just the second?

6) Check if modularity just happens more if you have more parameters.

As we've mentioned several times by now, real biology is very modular. Old neural networks with 10-100 parameters are usually not modular at all (unless particular strategies like MVG are used with the explicit goal of producing modularity). Modern networks with lots of parameters are supposedly somewhat modular sometimes[1].

So one hypothesis might be that modularity emerges when you scale up the number of parameters, for some reason. Maybe handling interactions of every parameter with every other parameter just becomes infeasible for most optimisers as parameter count increases, and the only way they can still find solutions is by partitioning the network into weakly interacting modules.

If this were true, adding more parameters to a network should make it more modular. To test this, you could e.g. take a simple image recognition architecture performing a very narrow image recognition task on MNIST, and calculate its modularity score, as described in (1). Then, you could look at architectures with increasingly deeper layers, solving more complicated image recognition tasks, and calculate their modularity scores as well. For large models, already trained ones could be used.

Questions you could ask:

- Is there a relationship between size and modularity metrics like Q-score?
- How many modules are found in the optimal decomposition when calculating the Q-score? Does this number also increase with larger networks?

7) Noisy channels & strongly interacting parameters

Mostly so far we've discussed the modularity characteristics of solutions found by the search process, but another interesting question you could ask is: how modular are neural networks at initialisation?

There are two possible extremes of scenarios we might get. One we could call the ***"scarce channels"*** scenario:

- The vast majority of information doesn't propagate very far through the system; it gets wiped out by noise
- Node-values in nonadjacent chunks of the system are roughly independent
- High modularity scores, because if information is wiped out at a distance then most of the interactions must be local
- In order for any information to be passed at all, a strong training signal is needed to select that information

And the other we could call the ***"scarce modules"*** scenario:

- Most parts of the system interact strongly with the rest of the system
- Node-values in nonadjacent chunks of the system are rarely independent
- Low modularity scores, because no chunk of the system can be interpreted as a module
- In order for a part of the system to turn into a module at all, a strong training signal is needed to induce a Markov boundary around it

It would be informative to initialise neural networks with random values and test which of these two scenarios is closer to reality, for the vast majority of random parameter settings.

Questions you could ask:

- Is the modularity score dependent on which randomisation method you're using?

- How large are the modules being found by your modular decomposition (if using the Q-score, or something similar)?
- Can you find a way of quantifying how far information propagates in the network?

8) Noisy input channels

Organisms in the real world can't really be overparameterized relative to the vast amount of incoming information they are bombarded with. To the extent that you could imagine defining a loss function for them, it'd be functionally impossible to reach perfect loss on it. As a consequence, we think such systems need ways to get rid of noise in the input so they can focus on the information that matters most.

In contrast, a lot of current deep learning takes place in an overparameterized regime where we train to zero loss, meaning we fit the noise in the input data. One could wonder whether constructing problems where fitting the noise is impossible would evolve networks better designed to throw away superfluous information. Since throwing information away also seems associated with sparsity and with it modularity, we are considering this as another hypothesis for why biological NNs seem so much more modular than artificial ones.

To test this, one could add random Gaussian noise to the input and label of a CNN MNIST setup[2] like the one described in (1). To be clear, this noise would be different every time an input is evaluated. Then, you could see whether the network evolves to filter this noise out, and if that leads to sparsity/modularity.

Questions you could ask:

- Is there a trade-off between performance and modularity as you add more noise?
- If the network does evolve to filter noise out, can you see what mechanism it's using to do this?
- Is this method more effective at producing modular solutions when it's combined with other methods outlined in this document, such as (1) and (4)?

9) Measuring modularity and information exchange in simple networks

As we've discussed before, we think a good measure of modularity should be deeply linked to concepts of information exchange and processing, and finding a measure which captures these concepts might be a huge step forwards in this project. Although no such measure is currently in use to our knowledge, there are several that have been suggested in the literature which try and gauge how much different parts of the network interact with each other. Most of them work by finding a "maximally modular partition" and measuring *its* modularity, with the distinctive part of the algorithm being how the modularity of a particular partition is calculated. For instance:

- Some are derived from tools used to analyse simple unweighted undirected graphs, e.g. the Q-score
- Some look at the weights, using e.g. the matrix norms of convolutional kernels.
- Some look at derivatives with respect to node input and output, coactivation of neurons, or mutual information of neurons

We're also currently working on a candidate measure based on counterfactual mutual information, which we'll be making a post about soon.

It would be valuable to compare these different measures against each other, and see if some are more successful at capturing intuitive notions of modularity than others.

This isn't just a theoretical issue either. Right now, it's looking like e.g. the matrix norm and node derivative measures give very different answers, where one might tell you that a network exhibits statistically significant modularity, whereas the other says there isn't any.

This suggests the following experiment: taking a very simple system (e.g. the retina task), training it until it finds a solution, and benchmarking and visualising all of these measures against each other on the learned solution.

Some questions you could ask:

- Which modularity measures give rise to similar "maximally modular partitions"? Which ones give partitions that are more similar than others? (this paper suggests a method for comparing the similarity of two different partitions)
- For small networks, you could try visualising the learned solutions and the partitions. Do some partitions look obviously more modular than others?
- Do your results change if you apply them on a solution which hasn't yet attained perfect performance?
- Try to construct networks that Goodhart a particular measure. How difficult is this? Do the results look like something that a typical training process might select for?

10) Visualisation tools for broad peaks

It's pretty straightforward to get a mathematical criterion for broad peaks, at least in networks trained to zero loss (e.g. see Vivek Hebbar's setup here). It would be useful to get some visualisation tools which can probe that criterion in real nets. For instance, what does the approximate nullspace on the data-indexed side of df/d\theta(\theta, x_i) look like?

This is the sort of thing which has a decent chance of immediately revealing new hypotheses which weren't even in our space of considerations. Broadness of optima have come up a few times in our thought experiments and empirical investigations so far, and we suspect that there are some pretty deep links between modularity of solutions and their broadness. Better visualisation tools might help illuminate this link.

Questions you could ask:

- Can you find a way to turn the aforementioned mathematical setup into a visualisation tool for broad peaks?
- Which kinds of graphs or visualisation tools might demonstrate broadness in an intuitive way?
- Are there other ways you can think of to quantify and visualise broadness?

Closing thoughts

We're really excited about this project, and more people contributing via ideas or running experiments. If you would like to run one of these experiments, or have ideas for others, or just have any questions or confusions about modularity or any of the projects above, please feel free to comment on this post, or send Lucius (Lblack) a private message here on LessWrong.

There are a few more project ideas we have, but some of them rely on more context or mathematical ideas which we intend to flesh out in later posts.

1. ^

   One should be careful here: according to Daniel Filian (one of the authors), when they ran these experiments again with a different measure for modularity, they got a different result: no modularity beyond what would be expected by random chance.

2. ^

   The retina task seems poorly suited for this kind of experiment, even though it’s simpler. It involves small binary inputs that seem like they’d be a pain to get to work with noise, if it’s doable at all.

What Is The True Name of Modularity?

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

TL;DR

Modularity seems like an important feature of neural networks, but there is currently no canonical way of properly defining or measuring it which is properly theoretically motivated and doesn't break down in some cases - in other words, we haven't yet found a True Name for it. Most modularity measures used in experiments are based on ad-hoc methods from graph-theory or network theory, and don't seem to capture the kind of modularity we care about.

In this post, we explore these existing ways of measuring modularity in neural networks, and their limitations. We also outline our ideas for a new modularity metric, and in particular the important role we think two branches of mathematics will play:

- **Information theory,** because neural networks are fundamentally *information-processing devices*, and we want to measure how this information is being used, exchanged and transmitted. In particular, we expect *mutual information* to be at the core of any metric we use.
- **Causal inference,** because correlation (and indeed mutual information) does not automatically equal causation, and we need a way of distinguishing "modules X and Y fire with similar patterns" from the more specific claim "modules X and Y are directly sharing information with each other". In particular, we expect *counterfactuals* to play an important role here.

Introduction

A first-pass definition for modularity might go something like this:

A system is modular to the extent that it can be described in terms of discrete modules, which perform more intra-modular communication than inter-module.

In previous posts, we've talked about some causes of modularity in biological systems, reasons we should care about modularity in the first place, and unanswered questions about modularity which require experimentation. In this post, we will return to a more theoretical question - how can we develop a yardstick for measuring modularity, which is built on sound mathematical principles?

First off, why is this an important thing to do? Because most current ways of measuring modularity fall pretty short. As recently discussed by John Wentworth, if all you have is a weak proxy for the concept you're trying to measure, then your definition is unlikely to generalise robustly in all the cases you care about. As a simple example, most architectural ways of measuring modularity in neural networks are highly dependent on the type of network, and we shouldn't expect them to generalise from CNNs to transformers.

Ideally, we'd eventually like to find a robust theorem which tells you in which cases modularity will be selected for. But unless we have the right language to talk about modularity, we will be stuck with hacky ad-hoc definitions that are very unlikely to lead to such a theorem.

Setting constraints

The lesson from John's post is that we can't just do random search through the high-dimensional space of mathematics to find something suitable. We have to start by writing down some intuitions for modularity, and constraints which we expect a modularity measure to satisfy.

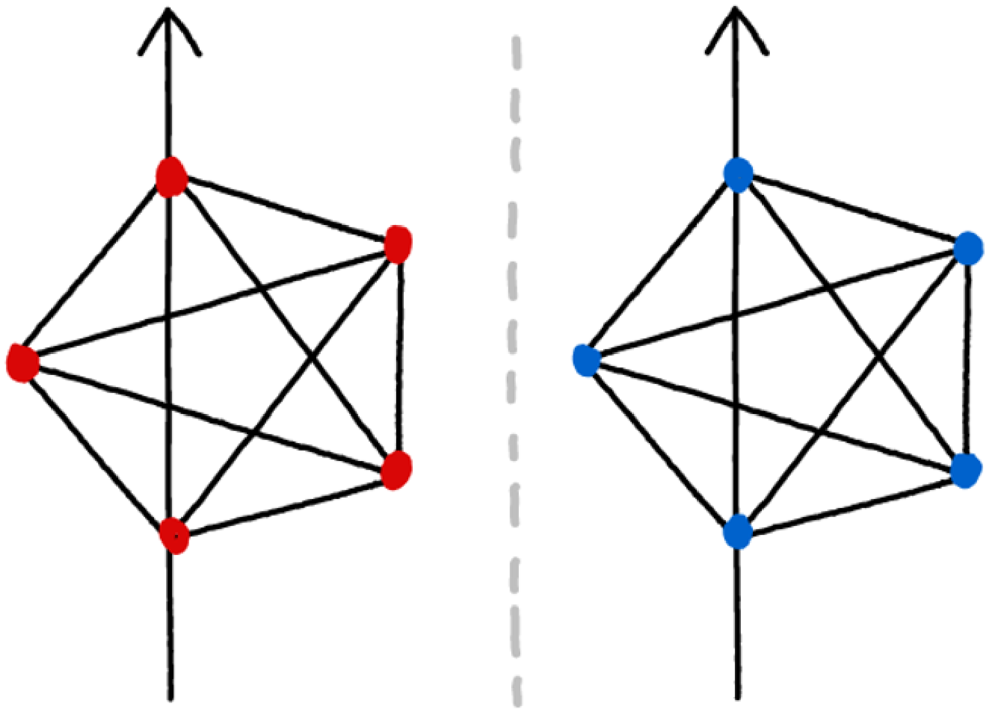
Adam Shimi uses the analogy of a set of equations, where we ideally write down enough to specify a unique solution, or else hone in on a narrow region of solution space.

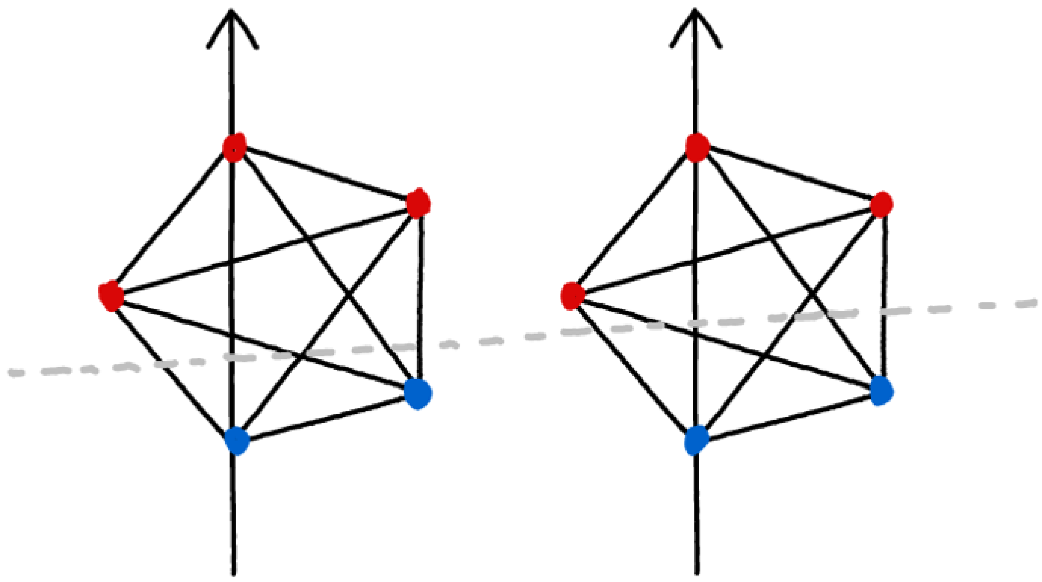
What would the equations be in this case? What are our constraints on a modularity measure, or some intuitions we expect it to satisfy?

Function of the model, not the system

It's meaningless to define the modularity of a system without reference to the things that are acting as modules. So in order to measure modularity, we have to first choose a partition into modules, and measure something about the system with respect to this partition.

Which partition should we choose? It makes sense to choose the partition which results in the highest modularity score on whichever metric we use, because this is in some sense the "most natural modular decomposition". As an example, consider the diagram below (red and blue nodes represent a partition into 2 modules). The first partition is clearly natural, and we can see that the system has perfect modularity with respect to this partition. The second partition is obviously unnatural because the two "modules" actually have a large number of connections.





Extreme cases

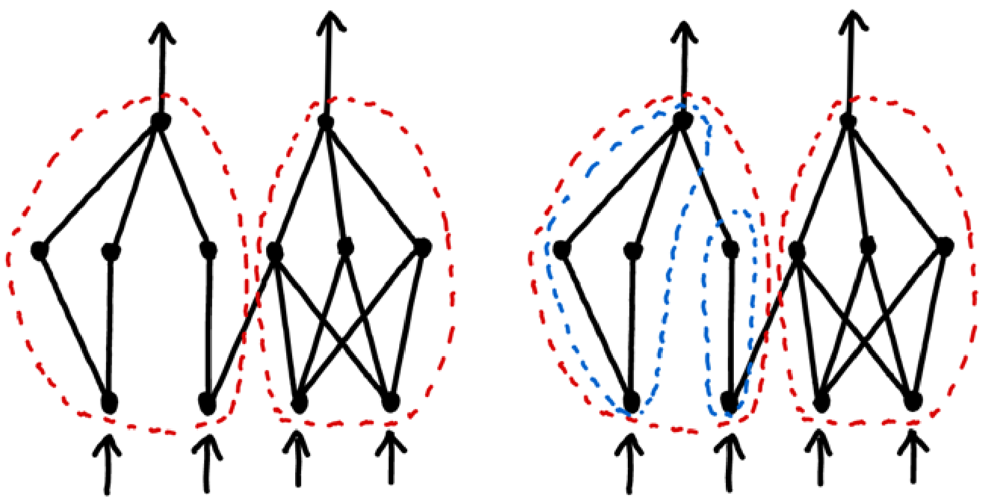
Some of the intuitions for correlation coefficient involved extreme cases (independence implying ρ = 0, and ρ = 1 implying perfect dependence). What are the extreme cases in modularity?

Perfect anti-modularity: any pair of modules shares as much information as any other. There is no way to partition the nodes in the network in a way such that more computation (proportionally) is done between nodes in the same module vs between nodes in different modules.

Perfect modularity: there is literally zero communication between any pair of modules. Note that we'd have to be careful applying this: if our choice of "modular decomposition" is to just have one module, then this is trivially perfectly modular by this definition. The example above is a better illustration of perfect modularity: two completely isolated subnetworks; each one fully connected.[1]

Hierarchical

As discussed above, modularity has to be a function of some particular model of a system, consisting of a set of modules. But ***we should be able to further break down each module, and identify modularity within it.***



One example: biological systems are modular at many different levels, from organs and organ system to cells and protein structure.

Function of information flows, not architecture

This requirement is motivated from a couple of different angles:

We could construct a pathological network which has some node that never gets activated, regardless of the input (this is really easy to do, we just need all the weights going into this node to be blocked). Obviously a modularity score shouldn't depend on this weight, because it doesn't represent any real information flows.

Fundamentally, NNs are information-processing devices, so in a way it seems intuitive to us that any measure of their modularity should use the language of information theory, not just analysis of weights and other architectural features.

Additionally, we are researching modularity in the context of discovering Selection Theorems about it. The kind of modularity we are looking for is whatever kind seems to be selected for by many local optimisation algorithms, e.g.:

- By evolution, in natural organisms at every scale
- By genetic algorithms and gradient descent, in neural networks

This might initially seem like a hint to look at the architecture instead. After all, it's the physical makeup of biological brains, and the parameters of neural networks, that seem to be most directly modified during the optimization process. So if you want theorems about when they select for modularity, why not define modularity in terms of these parameters? Because we think there are some hints, coming from places like the Modularly Varying Goals effect, as well as connections between the modularity of neural networks and their generality (post on this part coming soon), that modularity is connected to *abstraction*, possibly quite directly so.

As in, abstractions used by brains and neural networks may in some sense *literally be* information processing modules inside them. That's a pretty naïve guess, the actual correspondence could well be more complicated, but it does seem to us like there's *something* here.

Since abstraction is an information theoretic concept, we think this may be pointing towards the hypothesis that what's selected for is really something information theoretic, which just *correlates* with the modularity in the physical architecture sense that we more readily observe.

Some example measures we don't think do the job

Q-score

The seemingly most widely used measure/definition in papers whenever someone wants to do something modularity related with neural networks is the Q score, borrowed from graph theory (which we've talked about in a previous post).

$$Q = \frac{1}{(2m)} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{(2m)} \right] \delta(c_v, c_w)$$

This is a rather hacky, architectural measure based on the weights in the network. The bigger the weight between two neurons, the more connected they are, goes the basic idea.

This definitely captures our intuitive notions of modularity if we're talking about an undirected graph as a purely topological structure, i.e. not functional / no information flows. We think (hope) that this measure is at least *correlated* with "true" modularity in a lot of real networks, but we don't think it's a good fundamental definition.

Clusterability in Neural Networks

This was a paper that came out in 2019. It looks for internal structure in neural networks using a different formula than the Q-score, the normalised cut, which is based on similar ideas of modules having strong internal connectivity and weak external connectivity, as inferred from their weights. Therefore, this suffers from similar problems: looking at the architecture, not information flows. For instance, before applying the metric, the authors convert the neural networks into a weighted undirected graph in a way which is invariant of the biases, and biases can be the difference between a ReLU node activating and not activating - this is a pretty big red flag that we may not be capturing the actual behaviour of the network here!

A subsequent paper, Quantifying Local Specialisation in Deep Neural Networks, takes this further by taking into account high correlation of neurons, even if they aren't directly connected via weights. Importantly, this is functional rather than architectural, so by our criteria, it represents a step in the right direction. However, the methodology is still fundamentally based on the architectural measure from the previous paper.

What are the right tools?

Mutual information

A very simple way of measuring information flow in a neural network one could think of is literally counting how many floating point numbers pass between the nodes. But that information doesn't tell us much about the network's actual structure, and how much information is really being communicated through these numbers. For example, all nodes with an equal number of incoming connections will receive an equal amount of incoming information according to this measure. In reality, the node might e.g. only react to whether the incoming signals are greater than zero or not. If that happens for 5/10 inputs, the information communicated to the node would be 1.0 bits, no matter how many floats are involved in the operation. If it happens for 9/10 inputs, it'd be

$-0.9 \times \log_2(0.9) - 0.1 \times \log_2(0.1) = 0.469$ bits.

Information theory 101 might tell you that what you're looking for is the *mutual information* between nodes. This measures how many bits of information the two have in common.

The formula for mutual information[2] is

$$I(B : C) = \sum_{b,c} P_{B,C}(b,c) \log_2\left(\frac{P_{B,C}(b,c)}{P_B(b)P_C(c)}\right)$$

where P_B, P_C are the probability distributions over the possible values of nodes B and C respectively.

An alternative expression using conditional instead of joint probability distributions is

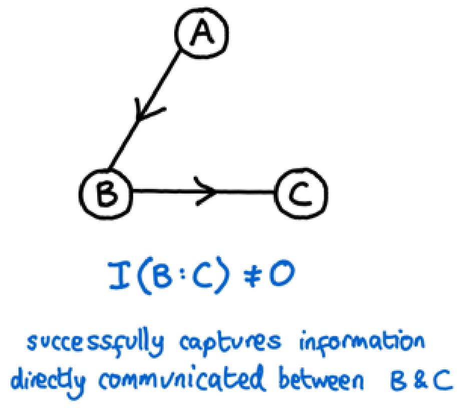
$$I(B : C) = \sum_{b,c} P_B(b) P_C(c \mid B = b) \log_2\left(\frac{P_C(c \mid B = b)}{P_C(c)}\right)$$

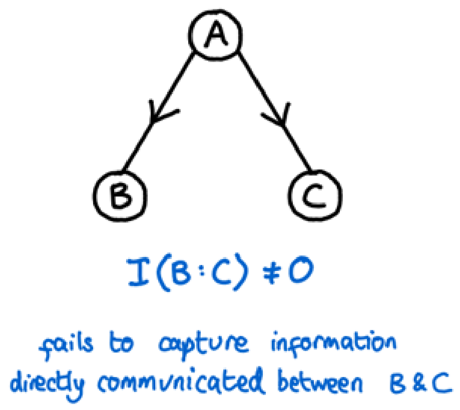
since Bayes tells us that $P_{B,C}(b,c) = P_C(c \mid b)P_B(b) = P_B(b \mid c)P_C(c)$.

So you measure the joined and marginal distributions of the activations of B and C on all the training data, and off you go, right?

In some specific cases, this works - for instance, in the example on the left below. In this case there's no direct connection between A and C. But in more general networks, we don't think so. Mutual information seems like the right thing to look at, but outside such simple cases, you have to be somewhat careful about *which* mutual information you use.

For example, if both B and C were fed some of the same signals from an input A (e.g. the right image below), they'd have mutual information even if they exchanged no signals with *each other* at all.[3]


A
B
C
I(B:C) ≠ 0
successfully captures information directly communicated between B & C


A
B
C
I(B:C) ≠ 0
fails to capture information directly communicated between B & C

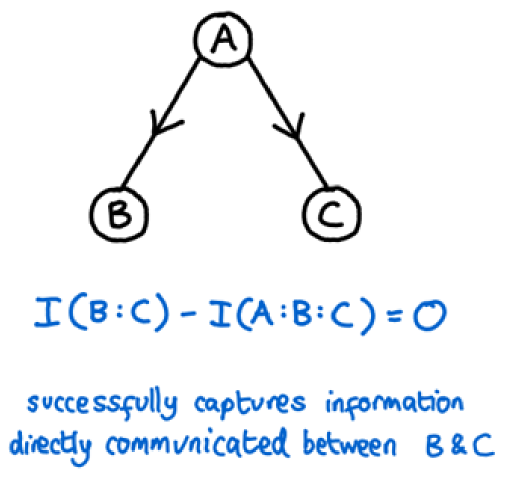
In feedforward neural networks, this kind of pattern is very frequent. For instance, suppose you had a network which was trained to output two numbers: the total number of ducks in an image, and the total number of shadows. Furthermore, suppose that in the training data, a sizeable fraction of the shadows are caused by ducks. Then, even if the network finds a perfectly modular solution which performs two completely separate tracks of computation (one for the number of shadows, one for the number of ducks), you will still get high mutual information between these two tracks, because the features you're measuring are highly correlated *in the training data*.

In summary, we wanted to measure how many bits of information B and C communicated to each other, but instead we've found ourselves just measuring how many bits they *share*, which is not the same thing at all.

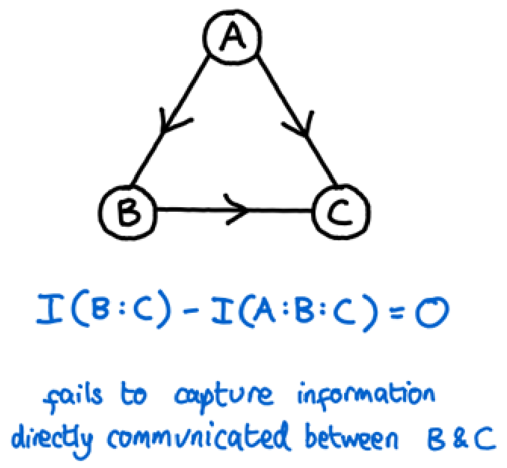
Causality

Alright, you say, easy enough to fix. Just subtract the mutual information shared by all of A, B and C from the mutual information between C and B! Then you've thrown away these spurious bits you don't want, right?

Nope. Counterexample: Say $I(A, C) = I(A, B)$, with all the information A sends to B also being send to C. Since B and C are just deterministic functions of A, this would tell you $I(A, B, C) = I(B, C)$.[4] So $I(B, C) - I(A, B, C) = 0$. Your measure says that no information was exchanged.



A
B
C
I(B:C) - I(A:B:C) = 0
successfully captures information directly communicated between B & C



A
B
C
I(B:C) - I(A:B:C) = 0
fails to capture information directly communicated between B & C

But that doesn't seem generally true at all, in the sense we care about! If A is an integer, (e.g. 345789), B divides this integer by three (115263), and C takes the result from B and adds it to the original number from A ($345789 + 115263 = 461052$), then C is clearly getting "relevant information" from B, even though the output of B is technically inferable from A.

To put it another way, if Bob is told that Christine is going to calculate the prime factorisation of 659011734457 and tell him the result five minutes from now, then in a sense, him being told this particular number and not any others is going to predictably coincide with Christie saying "$31 \times 53 \times 401102699$", and he isn't gaining any information. But unless Bob is actually doing the prime factorisation himself, then it sure seems like he received new information from Christine in some sense.

So how do you actually separate this out? How do you measure how much information is exchanged between B and C in the sense we seem to care about?

The core problem here is that as far as our distributions over node activations are concerned, everything in the neural network is just a deterministic function of the input A. Once you know the activations of A, the activations of B and C are fixed. There is no information to be gained from them anymore. $1 \times \log_2(1) = 0$.

But in common sense land, we know that just because you have all the data to theoretically infer something, doesn't mean that you actually can or will do so in practice. So if someone else hands you that inference, you gained information.

So let's go back to our mutual information formula, and see if we can look at slightly different distributions to capture our elusive common-sensical sense of information gain.

How do we describe, in math, that the inferences B sends to C are new information to C, when information theory stubbornly insists that the inferences are already perfectly determined by the data?

Counterfactuals.

You have to admit math that can express the signal B sends as being uncertain; as a new source of information to C, even when "in reality" it's always determined by A. So you have to look at what happens when B outputs signals to C that differ from what they would be in reality (determined by the input signals from A). As in, A sends 659011734457, but B is falsified such that it does not give out the correct prime factors, and sends something else instead.

How do you measure distributions over counterfactuals?

By running your neural network in counterfactual situations, of course!

You run the network over the whole training data set, as normal. You record the distribution of the activations $P(A, B)$, $P(A, C)$, as normal.

Then you go and run your network over the training set again, but artificially set the activations of B to values they wouldn't normally take for those data points. Then you measure how C responds to that.

But what values should you set B to? And with what distribution?

This is the part in all this we're currently most uncertain about. Right now, to start, we think it should be tried to just use the $P(B)$ distribution measured in the normal training run,

but *with its correlation with the input* A *broken*.

In other words, the “prior” we assume for C here to calculate the information exchange is that it knows what sort of signals C usually sends, and with what frequency, but it can’t tell in advance which inputs from A should cause which signal from C. The signals from C are independent bits. This captures the notion of Bob not knowing what the result of Christine’s calculation will be, even though he could find out in theory.

Expressing this intervention in the notation causality, this makes our proposed measure of the information exchanged between B and C look like this:

$$I(B:C) = \sum_{b,c} P_B(b) P_C(c \mid do(B=b)) \log_2 \left(\frac{P_C(c \mid do(B=b))}{P_C(c)} \right)$$

for this simple situation.

It’s just like the normal formula for mutual information above, except the conditional probability distributions have a “do” operator in them now, which represents the action of artificially setting the activation on node B node to the value b (i.e. we’re conditioning on an intervention, not just an observation).

We don’t have a proof for this. It comes from intuition, and the line of reasoning above. The measure seems to give the right answers in a trivial coin toss thought experiment, but it has not been tested in more complicated and varied situations yet. We’re also a bit unsure how you’d make it work out if B and C stretch over multiple layers of your feed-forward network, allowing information exchange in both directions. Though we have some ideas.

Next steps

The next step is testing this out in some extremely simple neural networks, as part of the experiments outlined under point nine in our experiments post.

Of course, this isn’t an actual modularity measure/definition yet, just a building block for one. If it turns out to be a good building block, or becomes one after some iterations of the experiment/theory feedback loop, how do we proceed with it?

The next step would be finding an analogous implementation of something like the Q score in graph theory.

One naïve way I could see this working right now: Say you start by proposing to partition your network into two modules[1]. You run the network through the training set once, and record the activations. Then you go through all possible partitionings, and perform an “intervention” network run to measure the information exchange between the subgraphs in each. You apply some weighting (which exactly?) to account for partitions with very few nodes to naturally send very little information. Calculate the average information exchanged over all partitionings.

Then, you take the partitioning resulting in the *minimum* information exchange. Those are your two modules. Divide the average information exchange by this minimum, and you have your modularity measure. It tells you how much information your modules exchange with each other, relative to how much information is flowing in this network in general.

Obviously, this procedure would be infeasibly expensive in a real network. We want to focus on finding the right measure and definition of modularity before we go worrying about ways to compute it cheaply, but for the more practically minded, a few rough intuitions to show that this probably isn't as bad as it looks at first.

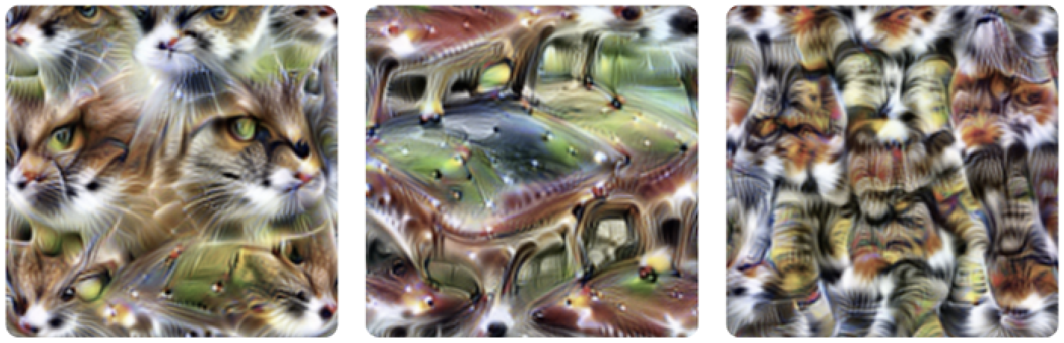
First, you don't need the average information exchange over *all* the partitions. If you statistically sample partitions and take the average information of those, that'll get you an estimate of the true average with some statistical error. You just need enough samples to make that error small enough.

To find the partition with minimum information exchange, you can search through partition space a lot more cheaply than just trying all the combinations. Use a genetic algorithm, for example.

If you can find some other, cheaper measure than correlates decently well with the true information exchange, even if it isn't exactly right, you can use that to perform most of the steps in your search, and only use the more expensive true measure every now and then, to course correct or check end results. One thing we're eyeing for this for example is the dimensionality of the interface between the partitions, which should be findable by looking at the Hessian matrix of the network output. You could also use the old graph theory measure based on physical weights. How close these various proxies are to the real thing we're after is another thing we're hoping to figure out from experiment 9.

Further considerations / final notes

As a final note, after some helpful input from Quintin Pope, we've recently reconsidered whether using individual neurons as the unit to construct the modules out of is really the right call. Things like polysemantic neurons seem to hint that the network's internal computation isn't necessarily best described in the neuron basis. So we might still be lending the physical structure of the network too much weight over its information processing properties by defining modules as groups of neurons.


4e:55 is a polysemantic neuron which responds to cat faces, fronts of cars, and cat legs. It was discussed in more depth in Feature Visualization [4].

So we're considering reformulating things so that modules are defined as subspaces in *feature space* instead, allowing for different bases in that space than the single neuron one. This thinking is in its infancy however.

So that's the state of things right now. We're currently mainly waiting for basic experimental data before we theorise more, but if anyone reading this can spot something, that could help a lot too. We're not information theorists. It's entirely likely we missed things.

1. ^

   How to choose the correct number of modules to partition into isn’t clear yet either. For more thoughts on this question, see this recent post.

2. ^

   This is the formula for mutual information of discrete random variables, where the sum is over the finite set of possible values the variables can take. For continuous ones it’s slightly harder, so what we end up doing in practice is sorting the continuous activations of neurons into (say) ten different buckets, and treating them as discrete RVs.

3. ^

   In the language of causality, we say that A is a confounder of B and C.

4. ^

   Three variable mutual information is defined by:

   $$I(A:B:C) = \sum_{a,b,c} p_{abc} \log_2 \left(\frac{p_{ab}\, p_{bc}\, p_{ca}}{p_{abc}\, p_a\, p_b\, p_c} \right)$$

   This is symmetric in (A, B, C), and it can also be written as $I(B:C) - I(B:C \mid A)$, where the latter expression is the conditional mutual information (i.e. the expected amount of information B and C would share, if we knew A). If either B or C are deterministic functions of A, then this conditional mutual information term is zero (because no constant can have mutual information with anything), so we get $I(A:B:C) = I(B:C)$ as claimed above.

Basin broadness depends on the size and number of orthogonal features

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

TL;DR

For neural networks trained to perfect loss, the broadness of optima in parameter space is given by the number and norm of independent orthogonal features the neural network has. The inner product that defines this "feature norm" and independence/orthogonality is the L_2 product of Hilbert space.

Introduction - why do we care about broadness?

Recently, there's been some discussion of what determines the broadness of optima of neural networks in parameter space.

People care about this because the size of the optimum basin may influence how easy it is for gradient descent (or other local optimisation algorithms which you might use to train a neural network) to find the corresponding solution, since it's probably easier to stumble across a broader optimum's attractor.

People also care about this because there's some hypotheses floating around saying that the broadness of optima is correlated with their generality somehow. The broader the basin of your solution is in parameter space, the more likely it seems to be that the network will successfully generalise to new data in deployment. So understanding what kinds of circuit configurations are broad seems like it might help us grasp better how AIs learn things, and how an AGI might really work internally.

Meaning, understanding what kind of solutions are broad is probably relevant to start developing general predictive theories of AI and AI training dynamics, which we'd like to have so we can answer questions like "What goals is my AGI going to develop, given that it has architecture A, and I'm training it with loss function L on data set D?"

Measuring broadness

Warning: calculus and linear algebra incoming!

Say we have a loss function $L(f, x, y)$ for a neural network $f(x, \Theta)$, with some data points x and labels $y(x)$, and we've trained the network to perfect loss, setting the parameters to some set $\hat{\Theta}$.

$$L = \sum_x l(f(x,\Theta), y(x))$$

[1]

$$\frac{dL(\hat{\Theta})}{d\Theta} = 0$$

Now, lets try to look at how much the loss changes when we make small changes to the parameters $\hat{\Theta}$. To do this, we're going to perform a polynomial expansion of L up to quadratic order. This isn't going to describe the function behaviour perfectly, but within a small neighbourhood of $\hat{\Theta}$, it'll be mostly accurate.

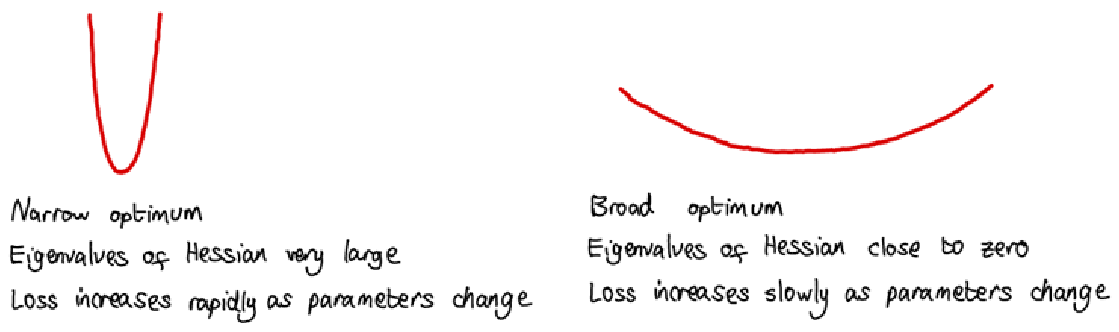
$$\begin{aligned} L(\Theta) - L(\hat{\Theta}) &= \frac{dL(\hat{\Theta})}{d\Theta}(\Theta - \hat{\Theta}) + (\Theta - \hat{\Theta})^T \frac{d^2L(\hat{\Theta})}{d^2\Theta}(\Theta - \hat{\Theta}) + O((\Theta - \hat{\Theta})^3) \\ &= (\Theta - \hat{\Theta})^T \frac{d^2L(\hat{\Theta})}{d^2\Theta}(\Theta - \hat{\Theta}) + O((\Theta - \hat{\Theta})^3) \\ &=: \delta\Theta^T \frac{d^2L(\hat{\Theta})}{d^2\Theta}\delta\Theta + O(\delta\Theta^3) \end{aligned}$$

The first order term vanished because $\frac{dL(\hat{\Theta})}{d\Theta} = 0$ (since we're at an optimum). $\delta\Theta := (\Theta - \hat{\Theta})$ is a vector with $\#\,\text{params}$ entries, specifying how much we're perturbing each parameter in the network from its value at the optimum $\hat{\Theta}$. $\frac{d^2L(\hat{\Theta})}{d^2\Theta}$ is going to be a matrix with $(\#\,\text{params}, \#\,\text{params})$ entries.

So far so good. Now, if we wanted to work out how broad the basin is by looking at this matrix, how would we do it?

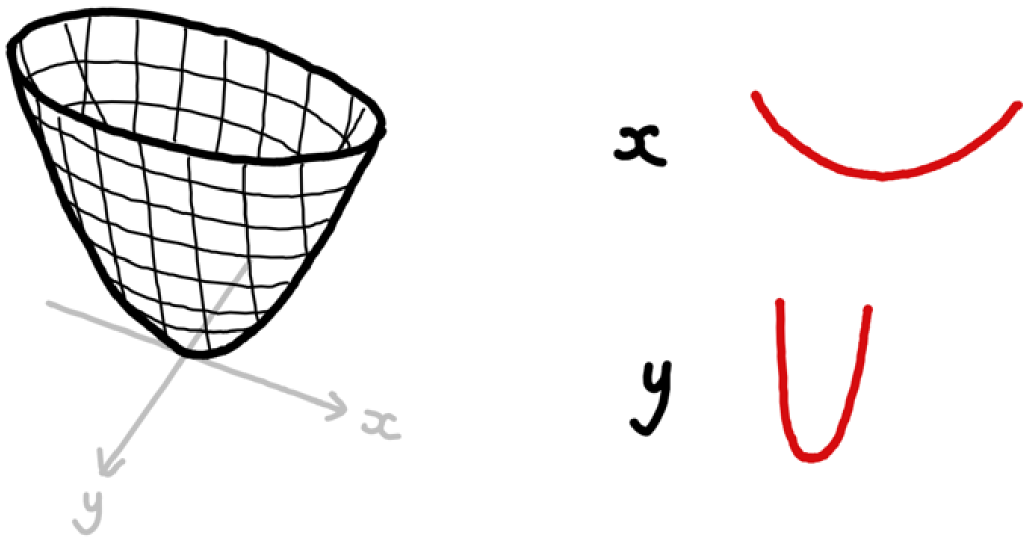
The matrix is filled with second derivatives. Or in other words, curvatures.

If you only had one parameter, this matrix would be a single number (the second derivative), telling you the curvature of the parabola you can fit to the optimum. And the curvature of a parabola determines its broadness, the closer the curvature is to zero, the broader the parabola, and with it the optimum, is.


Narrow optimum
Eigenvalues of Hessian very large
Loss increases rapidly as parameters change
Broad optimum
Eigenvalues of Hessian close to zero
Loss increases slowly as parameters change

If you have two parameters, you can imagine the optimum as a little valley in the loss function landscape. The valley has two curvatures that characterise it now, but those don't need to line up with our coordinate axes x and y. You could, for example, have one "principal direction" of curvature lie in the $(1, 1)$ direction of parameter space, and one in the $(1, -1)$ direction.

To find these principal directions and their curvatures, you perform an eigendecomposition of the matrix. The two eigenvalues of the matrix will be the curvatures, and the two eigenvectors the principal directions in parameter space characterised by these curvatures. Multiplying the parabola widths we get from the two, we obtain the basin volume.



We can do this for the n-dimensional case as well. If some eigenvalues of the matrix are zero, that means that at least within a small neighbourhood of the optimum, the loss doesn't drop at all if you walk along the corresponding principal direction. So each such eigenvalue effectively increases the dimensionality of the optimum, from a point to a line, a line to an area, etc.

Now that's all fine and good, but it's not very interpretable. What makes the eigenvalues of this matrix be small or large for any given network? What determines how many of them are zero? What does all this have to do with interpretability? Let's find out!

Broadness and feature space

Let's explicitly work out what that matrix, which is also called the Hessian of the network, will look like, by calculating its entries in the j-th row and k-th column, meaning the derivative of the loss function with respect to the j-th and k-th parameters of the network:

$$\begin{aligned}
\frac{d^2 L(\hat{\Theta})}{d\Theta_j d\Theta_k} &= \sum_x \frac{l(f(x,\hat{\Theta}),y(x))}{d\Theta_j d\Theta_k} \\
&= \sum_x \frac{d}{d\Theta_k}\left(\frac{dl(f(x,\hat{\Theta}),y(x))}{df}\frac{df(x,\hat{\Theta})}{d\Theta_j}\right) \\
&= \sum_x \left(\frac{d^2 l(f(x,\hat{\Theta}),y(x))}{df^2}\frac{df(x,\hat{\Theta})}{d\Theta_j}\frac{df(x,\hat{\Theta})}{d\Theta_k} + \frac{dl(f(x,\hat{\Theta}),y(x))}{df}\frac{d^2 f(x,\hat{\Theta})}{d\Theta_j d\Theta_k}\right) \\
&= \frac{d^2 l(f(x,\hat{\Theta}),y(x))}{df^2}\sum_x \frac{df(x,\hat{\Theta})}{d\Theta_j}\frac{df(x,\hat{\Theta})}{d\Theta_k}
\end{aligned}$$

The second term vanishes again, because the loss function is required to be convex at optima, so $\frac{dl(f(x,\hat{\Theta}),y(x))}{df} = 0$.[2]

Look at this expression for a bit. If you've done some quantum mechanics before, this might remind you of something.

Let's go through a small example. If you have a network that takes in a single floating point number x_1 from the input x, and maps it to an output using features x_1, $\cos(x_1)$ and a constant, as in $f(x) = \Theta_0 + \Theta_1 x_1 + \Theta_2 \cos(x_1)$, the gradient of the output would look like:

$$\frac{df(x,\hat{\Theta})}{d\Theta} = \begin{bmatrix} 1 \\ x_1 \\ \cos(x_1) \end{bmatrix}$$

So, each entry in the gradient corresponds to one feature of the network.

The Hessian matrix for this network would be

$$\frac{d^2 L(\hat{\Theta})}{d\Theta_j \, d\Theta_k} =$$

$$\frac{d^2 l(f(x, \hat{\Theta}), y(x))}{df^2} \begin{bmatrix} \sum_x 1 & \sum_x x_1 & \sum_x \cos(x_1) \\ \sum_x x_1 & \sum_x x_1^2 & \sum_x x_1 \cos(x_1) \\ \sum_x \cos(x_1) & \sum_x \cos(x_1) x_1 & \sum_x \cos(x_1)^2 \end{bmatrix}$$

These are ***the** L_2 **inner products*** ***of the features over the training data set.*** They tell you how close to orthogonal the features are with respect to each other, and how big they are.

The idea of the L_2 inner product / norm and Hilbert space is that we treat the space of ***functions*** as a vector space, which can have basis vectors. The size and relative angles of these vectors are given by the L_2 inner product: you multiply the functions in question, and sum the result over the function domain.

This gives us a well defined notion of what constitutes "orthogonal" features: ***Two features are orthogonal if their*** L_2 ***norm is zero***. Intuitively, this means they're separate, non-interacting components of the network mapping.

This generalises to arbitrary numbers of features in arbitrary numbers of layers, though features in earlier layers enter multiplied with derivatives of activation functions, since we're characterising their effect on the final layer.

Each constant feature is treated as multiplying a constant of 1. The Hessian of the network at the optimum contains the L_2 norm of the products of all the features in the network with each other.

So, back to our second order approximation of the loss function. When we find the eigendecomposition of our Hessian matrix, that corresponds to shifting into a basis in parameter space where all off-diagonals of the Hessian are zero. So, the eigenvectors of the Hessian correspond to linear combinations of network features that are orthogonal (in terms of L_2 norm) to each other.

Remember, the bigger the eigenvalues, which are the L_2 norms of these feature combinations, the bigger the curvature of the loss function is in that direction of parameter space, and the narrower the basin will be.

Eigenvalues of zero mean the network actually had less independent, orthogonal features inside it than it had parameters, giving you directions that are perfectly flat.

So: ***the broadness of the basin is determined by the number and size of independent features in the network***. At least within second order polynomial approximation range.

That sure sounds like it's pointing out the precise mathematical notion of network "complexity", "fine-tuning" or "generality" that's relevant for the basin broadness.

It also neatly quantifies and extends the formalism found by Vivek in this post. If a set of n features all depend on different independent parts of the input, they are going to have an orthogonal basis of size n. If a set of n features depends on the same parts of the input, they are a lot more likely to span a set of dimension less than n in Hilbert space, meaning more zero eigenvalues. So, information loss about the input tends to lead to basin broadness.

Some potential implications

If we leave second order approximation range, features in different layers of the network no longer interact linearly, so you can't just straightforwardly treat them as one big set of Hilbert space basis vectors.

But features in ***the same*** layer can still be treated like this, and orthogonalised by taking the vector of neuron activations in the layer times its transpose, summed over the training dataset, to form a matrix of Hilbert norms like the one we got from the Hessian. Then, finding the eigenbasis of that matrix corresponds to finding a basis of linear combinations of features in the layer that are mutually orthogonal. Once the bases of two adjacent layers are found, the weight matrix connecting the two layers can also be transformed into the new basis.

This might give us a new perspective of the network. A basis in which each element in a layer might stand for one “independent” feature in that layer, instead of a neuron. Cancellations between the signals coming from different nodes could no longer occur. The connection weights might tell you exactly how much the network “cares” about the signals coming from each element.

This stands in contrast to the usual method (where we treat the neurons as the fundamental unit of the network, and use measures from network theory to try and measure clustering among neuron groups). As we've pointed out before, how big the weights between neurons are doesn't really indicate how much signal is flowing through their connection, making this a non-optimal strategy for interpreting your network.

This makes us suspect that the orthogonal feature basis of the layers might be close to the right framing to use, if we want to quantify how many "independent" calculations a neural network performs, and how it uses these calculations to make new calculations. In other words, this this formalisation might provide the basis for the fundamentally motivated modularity measure we're after.

How might we test this hypothesis?

In our previous post, we discussed the problem of **polysemanticity**: when a single neuron seems to represent multiple different concepts which aren't naturally bucketed for humans. This problem suggests that individual neurons might not be the best unit of analysis, and we could do better to use **features** as the basic unit with which to formalise general theories of neural networks. A natural question follows - if we try out graphing the "feature flows" in neural networks with this, do polysemantic neurons mostly disappear and every orthogonalised feature means one, understandable thing?

We plan to find out soon, by experimenting with this change of basis and related ones in small neural networks. If you're interested in helping out with these experiments, or you have any other suggestions for useful experiments to run to test out these ideas, please let us know by commenting below, or sending a private LW message.

1. ^

   Note that we've written this with a sum (and have continued this convention for the rest of the post), but the formalisation would work equally well with an integral.

   Additionally, for the rest of the post we're assuming f is a scalar to make the math a little less complicated, but the formalism should hold for other output types too.

2. ^

   How this works out when you're not at a perfect optimum and this term stays around is something I'm unsure of, and its the reason we proposed experiment 8 in the experiment list on LW.

3. ^

   $\frac{dL}{d\theta}$ is zero because we required a perfect loss score, and loss functions are required to be convex.