Analogies Emerge from Learning Dyamics in Neural Networks

Andrew Lampinen (lampinen@stanford.edu)
Department of Psychology, Jordan Hall, 450 Serra Mall, Stanford CA 94305

Shaw Hsu (cshawhsu@stanford.edu)
Department of Biophysics, James H. Clark Center, 318 Campus Dr., Stanford CA 94305

James L. McClelland (mcclelland@stanford.edu)
Department of Psychology, Jordan Hall, 450 Serra Mall, Stanford CA 94305

Abstract

When a neural network is trained on multiple analogous tasks, previous research has shown that it will often generate representations that reflect the analogy. This may explain the value of multi-task training, and also may underlie the power of human analogical reasoning – awareness of analogies may emerge naturally from gradient-based learning in neural networks. We explore this issue by generalizing linear analysis techniques to explore two sets of analogous tasks, show that analogical structure is commonly extracted, and address some potential implications.

Keywords: neural networks; structure learning; representation; analogy; transfer;

Introduction

Analogical transfer is often considered an essential component of "what makes us smart" (Gentner, 2003). However, there is a tension in the literature – Detterman (1993) has declared that "significnt transfer is probably rare and accounts for very little human behavior." Yet other authors have found that in some cases analogical transfer between superficially dissimilar systems can be so natural that it may not even require explicit awareness of the analogy (Day & Goldstone, 2011). How can we reconcile these viewpoints?

One feature that often separates the researchers with these opposing viewpoints is the type of tasks and transfer they consider. When Detterman (1993) says that the manipulations necessary to show transfer have "the subtlety of [a] baseball bat", he cites work like that of Gick & Holyoak (1980) which shows the difficulty of **rapidly** making an **explicit** mapping between two superficially disparate domains to explicitly solve a problem. By contrast, the Day & Goldstone (2011) experiments show transfer when participants learn about a system by **interacting** with it **over a longer period of time**, and then transfer is measured **implicitly** on an analogous system. We believe that this distinction between fast-explicit analogical transfer and slower-potentially-implicit analogical transfer may explain much of the disagreement in the literature. (See also Bransford & Schwartz (1999).)

Previous work has shown that neural networks can provide a good model for "slow" analogical transfer in domains as broad as artificial grammar learning (Dienes et al., 1999) and verbal analogies Kollias & McClelland (2013). In particular, one line of work shows that neural networks are capable of extracting analogous structure from knowledge domains that are completely non-overlapping in their inputs and outputs (Hinton, 1986; Rogers & McClelland, 2008). In other words, if you train a neural network to solve two identical tasks, using separate sets of inputs and outputs but sharing the hidden units, in some cases it will generate representations that reflect the analogy (i.e. analogous items will generate more similar patterns of activity in the hidden units than non-analogous items) (Rogers & McClelland, 2008). This can lead to the ability to correctly make analogical inferences about items not explicitly taught (Hinton, 1986). This extraction of shared structure sets neural networks apart from simple forms of statistical pattern recognition (Rogers & McClelland, 2008) such as linear data analysis techniques like PCA.

Furthermore, recent work has shown that neural networks can show benefits of training on multiple tasks (Dong et al., 2015; Rusu et al., 2015, e.g.). Even a small amount of learning on distinct but related tasks has been shown to improve performance. For example, training a natural language translation system on image captioning and autoencoding improves translation performance (Luong et al., 2016). Learning on numerous language translation pairs can even give generalization without further training to unseen language pairs (Johnson et al., 2016). We suggest that these benefits may be due to neural networks ability to extract shared structure. Because human experience is filled with distinct tasks that share common elements (language, various perceptual modalities, etc.) understanding the way that structure is learned across tasks may be essential to understanding human intelligence and building better artificial intelligence systems.

However, we have little understanding of how, why, or when neural networks are able to extract structural analogies from their training data. Here, we describe a preliminary investigation into this question, and in the process describe a new approach to analyzing neural network representations that may yield more general insights. We begin with a very simple instantiation of a task with analogous structure.

A Simple Task

In the original work of Hinton (1986), a neural network was taught to answer queries about the structure of two perfectly analogous family trees (one English and one Italian, see fig. 5), and was shown to generate representations that extract the analogy, in the sense that analogous people from different families are represented similarly. Here, we pare this task down to its barest essentials: two perfectly analogous domains with separate inputs and ouputs. For our task, the inputs can be thought of as the set of letters $\{R, L, \rho, \lambda\}$, and the outputs as $\{P, D, S, \pi, \delta, \sigma\}$. The task can be seen as mapping an input letter onto the letters that it can follow (e.g. "R" can follow "D" as in "draw," but cannot follow "S"), where there is an analogy between the Latin and Greek letters. See below for the input-output (I/O) mapping:

	P	D	S	π	δ	σ
R	1	1	0	0	0	0
L	1	0	1	0	0	0
ρ	0	0	0	1	1	0
λ	0	0	0	1	0	1

When and how does a neural network extract the analogous structure across the domains in this simple task?

Methods: Linear Networks?

There have been recent developments in the theory of linear neural networks which show that the process of learning is entirely driven by the Singular Value Decomposition (SVD) of the input-output correlation matrix (Saxe et al., 2013). The SVD can be seen as breaking the structure of the task into individual "modes" – linear structures in the dataset, somewhat like components in PCA. Specifically, a mode consists of an input pattern (which can be interpreted in this case as the input letters the mode responds to), a singular value (which roughly corresponds to the amount of variance explained by this mode), and an output mode (the output letters produced by the given pattern on the inputs). For example, see Fig. 1 for the SVD of the I/O mapping for the letter task above.

This decomposition tells us more about the task structure the network is using. There are three modes in the SVD. The first (left output mode/top input mode) represents the difference between the Latin and Greek letters, so it is positive for the Greek inputs and negative for the Latin outputs, and is positive for the Greek outputs and negative for the Latin outputs. The next two components represent the distinctions between the letters R and L, and the letters ρ and λ, respectively. Saxe et al. (2013) showed these results have implications for the learning of non-linear networks as well, so linear neural networks can be a more tractable place to analyzee learning dynamics. In addition, using the I/O SVD allows the discovery of representational components which are distributed across units, so it is more general than simply examining what aspects of the task individual hidden units represent, or examining the weight matrices directly. Thus one might hope to answer our questions in a linear framework.

However, linear networks cannot represent analogous structure from non-overlapping inputs and outputs at convergence. With non-overlapping inputs and outputs, the I/O correlation matrix is block diagonal, and the SVD modes will thus occur within blocks (this is why in Fig. 1 the modes showing separation between the letters in each domain have no input or output weights to the other domain).[1] Thus, since the final representational components that a linear network learns are precisely the components of the SVD (Saxe et al., 2013), there will be no sharing of structure across domains.

Furthermore, the optimal rank k approximation to a matrix is to take the top k components from the SVD (Mirsky, 1960). If a linear network's hidden layers are restricted to rank lower than that of the I/O correlation matrix, detail within the domains will be lost. Thus a linear neural network cannot solve the task perfectly if any of its hidden layers has a number of units smaller than the rank of the I/O correlation matrix. By contrast, a non-linear network can exploit the analogy between the domains to find more parsimonious solutions. Is there a way to leverage linear insights in the non-linear case?

Methods: A Linearized Approach

As we shall see, while a linear network cannot extract the analogous structure from the task, inserting a single non-linearity after the output layer can allow it to do so. In the case that the non-linearity is a sigmoid, this essentially reduces the problem to logistic regression; here we will use rectified linear units in our analysis because their structure makes the output patterns more intuitively interpretable. Once this almost-linear network has solved the problem, consider its outputs immediately prior to the non-linearity. These are produced by the linear part of the network, and together with the non-linearity suffice to produce the desired outputs. We can use these to turn the problem into a linearly analyzable one – simply treat these pre-nonlinearity outputs as outputs of a linear network. Then the problem becomes susceptible to the types of linear analyses discussed above.

Thus we trained a neural network with a single hidden layer (4 units) and a single non-linearity (a rectifier at the output layer) to solve this task. See fig. 3 for a diagram of the network. No biases were used, weights were initialized uniformly between 0 and 0.1, all training was done by Stochastic Gradient Descent (i.e. in each epoch the data are presented one at a time in a random order, and the weights are updated after each data point) with $\eta = 0.01$ for 500 epochs.

Results

The solution that the nonlinear network discovers the majority of the time (about 75%) is to output the same pattern on both sets of output units, but offset the "incorrect" domain sufficiently negative so that it is hidden by the rectified, thus the task that the linear portion of the network is effectively performing at convergence is:

	P	D	S	π	δ	σ
R	1	1	0	0	0	-1
L	1	0	1	0	-1	0
ρ	0	0	-1	1	1	0
λ	0	-1	0	1	0	1

[1]Where there are duplicated singular values, the SVD is not unique, so more precisely we mean there exists a basis which makes the SVD is block diagonal.

(Note that the network can actually map the first element of one domain onto either element of the other. We discuss the solution shown here one for clarity, the other just shuffles some rows and columns.)

The SVD of this linearized mapping shows a rank 2 solution (see fig. 2). The first component is similar to the first component of the regular SVD, in that it reflects the separation of the domains, but the second component collapses the other two components of the linear SVD. In other words, the analogy has been learned – the network is using the parallels between the two tasks to reach a more parsimonious solution. It is able to incorporate the analogy into its computations by allowing both the sets of outputs to vary, and simply suppressing the outputs from the "wrong" domain for its current task.

Because this solution is rank 2, a non-linear network with two hidden units should be able to solve the task, whereas a linear network will require three. We have verified these results empirically for this task. Thus the ability of a non-linear neural network to extract common structure from multiple tasks can allow it to find more parsimonious (i.e. lower-rank) solutions. We would like to highlight this point: the representation of the analogy in the SVD is not purely epiphenomenal – it makes a more parsimonious solution possible.

Evolution of the I/O Mappings

When a non-linear network has only two hidden units, it must extract the analogy to be able to solve the task, but with more hidden units there are a variety of solutions that could potentially emerge (such as just learning the mapping of each input to its output pattern independently). However, our network extracted shared structure on about 75% of the runs we conducted (as measured by more than 20% score on the cross-projection metric described below). What drives this fairly consistent extraction of analogy? In this section we consider the evolution of the outputs over the course of learning.

The output structure of the network goes through a fairly consistent progression, which we will describe qualitatively at various key stages (the exact values depend on the initialization, so the matrices here are approximations to within about ± 0.1). The outputs begin as small positive numbers, approximately 0 (because the weights are initialized uniformly between 0 and 0.1). Next, the network captures the base rate activations of each output unit, around epoch 75. (Note that this is already accounted for in the SVD, because the output variables are centered before computing the SVD).

$$\text{base rates} = \begin{bmatrix} 0.5 & 0.25 & 0.25 & 0.5 & 0.25 & 0.25 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.5 & 0.25 & 0.25 & 0.5 & 0.25 & 0.25 \end{bmatrix}$$

Then the network captures the existence of the two domains but not the structure within them (around epoch 140). This corresponds to the first component of either SVD. Up to this point, a linear network follows a similar learning trajectory.

$$\text{base rates by domain} = \begin{bmatrix} 1 & 0.5 & 0.5 & 0 & 0 & 0 \\ 1 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 & 0.5 & 0.5 \end{bmatrix}$$

Finally it learns the internal structure of the domains (they are not learned at exactly the same time, which is learned first depends on the initilization). Around epoch 400 it has solved the task completely, with some sort of offset structure in the non-linear case, or without in the linear case:

$$\text{solution with offsets} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 1 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

For most of the learning process, the networks are extracting similar structure, so one might expect that even the linear network would show some representation of the analogy at intermediate stages of learning. Indeed, once the base rates by domain are learned, both the linear and non-linear networks begin to extract the analogy between the domains. See fig. 4 for a plot of how much each domain's input mode projects to the **other domain's** output mode, i.e. "cross-talk" between the domains. This is a simple measure of the extent to which the network is extracting shared structure. However, while both networks develop some representation of the analogy initially, this activity extinguishes rapidly in the linear network, while it persists in the non-linear network.

Why do both networks show some representation of the analogy initially? We will analyze this in the linear case. At the stage when the base rates by domain have been learned, adding a little bit of shared structure actually reduces mean-squared error (MSE). If the network moves from the base rates by domain pattern to the pattern shown below, the small increase in MSE from the ± 0.1 values is more than offset by the decrease from splitting the 0.5 values into 0.4 and 0.6.

$$\begin{bmatrix} 1 & 0.6 & 0.4 & 0 & 0.1 & -0.1 \\ 1 & 0.4 & 0.6 & 0 & -0.1 & 0.1 \\ 0 & 0.1 & -0.1 & 1 & 0.6 & 0.4 \\ 0 & -0.1 & 0.1 & 1 & 0.4 & 0.6 \end{bmatrix}$$

Indeed, suppose there is a hidden unit which responds differentially within the domains (as they all will to some extent because of the random initialization). The updates of the output weights for this unit will point in the direction of analogy extraction once the base rates by domain have been learned. See below for the output error, hidden unit activity, and corresponding weight updates in the case that the hidden unit responds positively to the first element of each domain, and negatively to the other.[2] (Note that the output weight updates for a hidden unit are proportional to the product of the output error and the hidden unit's activation.)

[2]In the general case representations will be distributed across the hidden units, and so there will not be a unit which responds to the analogy and nothing else, but this is simply a rotation of the representation space, and because of the linearity of derivatives the same general pattern will emerge.

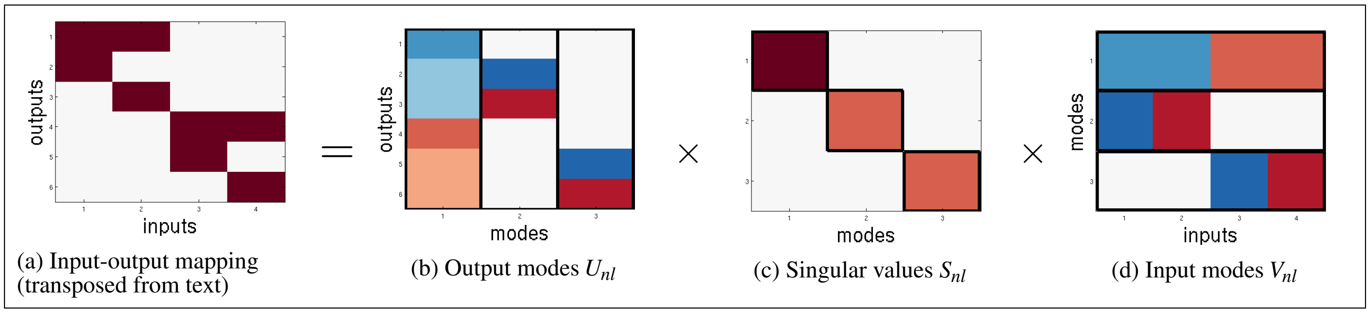

Figure 1: SVD of I/O correlation matrix (colors are scaled to show qualitative features, red = +, white = 0, blue = -)

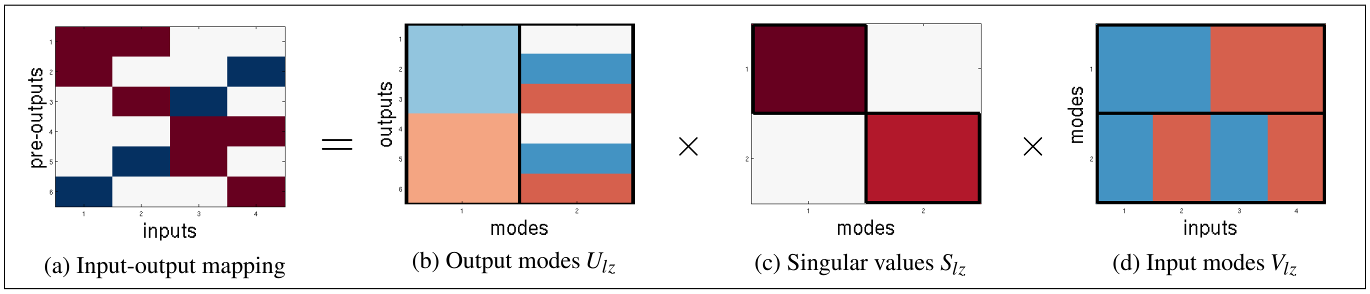

Figure 2: SVD of linearized I/O correlation matrix (colors are scaled to show qualitative features, red = +, white = 0, blue = -). Note how Fig. 2a becomes Fig. 1a if the negative values are hidden by a nonlinearity.

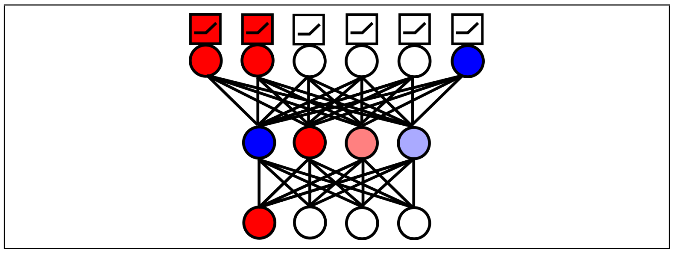

Figure 3: Simple task network, showing a sample propagation of an input through the network with the single non-linearity at the output. (Circles represent inputs or fully connected units, squares represent non-linearities.)

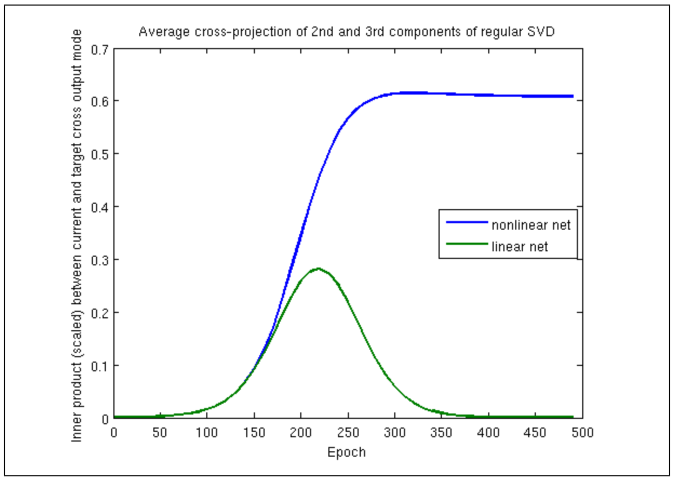

Figure 4: I/O SVD component cross-projection (dot product between output mode of an SVD component and the response of the network to the **other domain's** input mode)

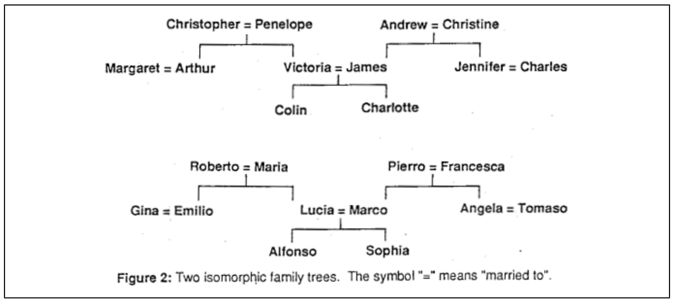

Figure 5: Family trees from Hinton (1986), (reproduced with permission).

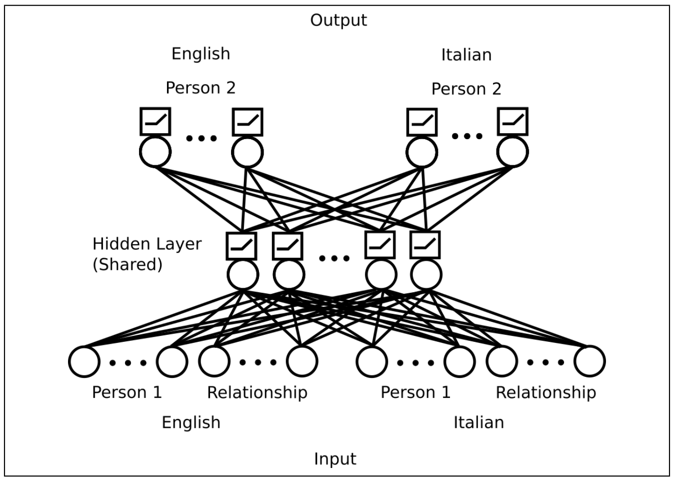

Figure 6: Family tree task network (Circles represent inputs or fully connected units, squares represent non-linearities. Ellipses denote units omitted from the diagram – the hidden layer and all input and output groups had 12 units apiece.)

output error						unit	unit output weight updates					
0	$+$	$-$	0	0	0	$+$	0	$+$	$-$	0	0	0
0	$-$	$+$	0	0	0	$-$	0	$+$	$-$	0	0	0
0	0	0	0	$+$	$-$	$+$	0	0	0	0	$+$	$-$
0	0	0	0	$-$	$+$	$-$	0	0	0	0	$+$	$-$
net output weight update:							0	$+$	$-$	0	$+$	$-$

Summing these updates captures the analogy between the domains. The network will exploit this analogy to reduce error, even if it must eventually discard it in the linear case.

Reanalyzing Hinton's Family Tree Example

Next, we briefly turn our attention to the example of Hinton (1986). Hinton's task involves learning two isomorphic family trees, one English and one Italian (see fig. 5). This structure is taught implicitly by presenting a person (e.g. "Jennifer") and a relationship (e.g. "Father"), and training the network to produce the correct target person ("Andrew" in this case). There are 52 such relationships per family.

Methods

Hinton used the same inputs for type of relationship for both families. To highlight the extraction of analogous structure we separated these into distinct input banks (these could be thought of as the English and Italian words for different relations, e.g. "uncle" vs. "zio"). We also reduced his network down from 3 hidden layers to a single hidden layer with 12 units. Unlike the simple problem above, this problem is not linearly separable, so we included a non-linearity at the hidden layer as well as the output (see fig. 6). We trained this network by SGD with $\eta = 0.005$ for 1000 epochs.

In a task which requires multiple non-linearities, we cannot perform as simple an analysis as in the earlier task. However, by definition each layer of the network has only a single non-linearity, and so we can perform an analysis like the above on each layer. In this way we can understand something about the computations that layer is performing. However, the interpretation will not be as simple as above.

This difficulty is compounded by the complexity of the structure being learned in each family. In the simple problem above it was possible to "eyeball" the structure extraction, but here the structure is too rich. There are a variety of possible ways the families can be mapped onto one another (e.g. flipping the tree left to right and swapping all genders), and it's possible that the networks are extracting overlapping structure from several of these analogies. In this setting, how can we examine whether the network is learning the analogy?

As a first test of this, we looked for representation of the analogy in the input modes of the first layer SVD. To do this, we computed the dot product of each mode's weights for one family with that mode's weights for the other family, and then tested how significant this similarity was by comparing it to the null distribution generated nonparametrically by randomly permuting the columns of the input mode matrix 1000 times and computing the same dot product for each one. We denoted a mode as showing significant extraction of the analogy if it showed a stronger similarity between the weights for the two families' inputs than 95% of its permutations did. We repeated this analysis across 100 network initilizations.

Results

We found a great deal of analogous structure was extracted. The runs had a median of 4 modes showing significant analogous structure extraction, and all the runs had at least one significant mode (for comparison, if 5% of the modes showed significant results by chance, we would still expect 54% of the runs to yield no significant results). To account for the symmetry of the tree under flipping, we repeated the same analysis after permuting the second family's input columns appropriately. Since the network has no way to distinguish the "regular" mapping from this "flipped" mapping during learning, we would expect to see a similar frequency of significant modes for each, and indeed the distributions are similar. Furthermore, the runs had a median of 6 modes showing significant extraction of either the regular or flipped mapping, and in all of the runs it had extracted 3 or more components that significantly represent one analogy or the other (if we assume 5% false positives, we would expect results this extreme in only 0.01% and 3% of the runs, respectively). See fig. 7. The frequency of analogy extraction suggests this may be a central feature of how neural networks solve tasks.

Although we have focused on broad analogies between the families here, we would like to note that analyzing the SVDs can give more detail. In some cases modes reflect an analogy only in the "person" inputs, or only in the "relationship" inputs. Within a family, analyzing the SVD modes can outline the structure the network is extracting, e.g. modes often appear which represent the gender of the target of a relationship like "mother". We have omitted these analyses due to length constraints.

Disussion

We have outlined a new technique for analyzing neural network representations and their learning dynamics: analyzing the SVD of the "linearized" mapping at each layer (i.e. the mapping from the inputs to the pre-nonlinearity activity). This allows us to bring the power of linear analyses to bear on the rich phenomena that occur only in non-linear networks.

Using this technique, we have explored how a simple neural network can extract the analogy between simple tasks with non-overlapping inputs and outputs. We showed that, while a linear network cannot represent analogies, a single non-linearity at the output layer can allow the network to represent the analogy, and that this structure emerges naturally (even in a linear network) from gradient descent once the base rates by domain have been learned. A linear network must discard this analogy to reach its optimal solution, but a non-linear network is able to retain it by simply offsetting the outputs to a sufficiently negative value, and does so the majority of the time in our results. Here we used rectifiers, but the same general solution is achievable with other nonlinearities.

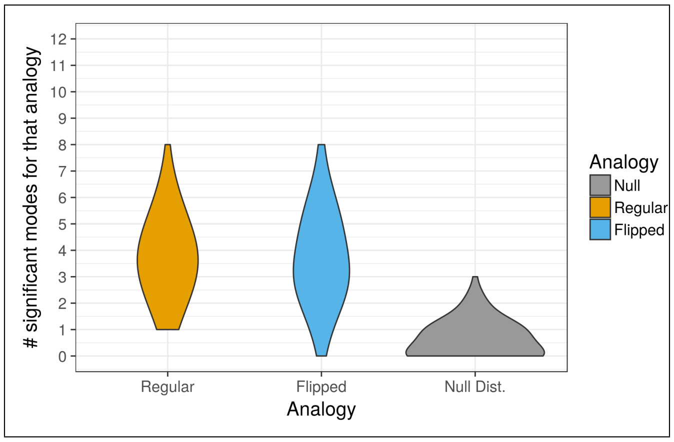

Figure 7: How many of the input modes from the SVD showed significant projection onto the regular or flipped analogies (with null distribution for comparison)

We then broadened our approach to explore the family tree task originally proposed in Hinton (1986). Because this task is not linearly separable, we created a general network with two nonlinear layers, and applied our analysis to each layer. We found evidence of a great deal of extraction of two possible analogies between the families in the network (either the intended isomorphism between the family trees, or one in which one family tree was flipped left-to-right and gender-reversed), and that networks seemed generally to be discovering elements of both analogies. Indeed, representation of the analogies seemed even more common than on the simpler task. On the simple task 25% of the networks showed no evidence of common structure extraction, but on the family tree task every network extracted at least three input modes that projected significantly onto one of the analogies.

These results suggest that sensitivity to analogy may be a natural feature of gradient based learning in nonlinear neural networks. This may underlie many of the "slow" analogical transfer effects we highlighted in the introduction. Furthermore, this may be a part of why learning multiple tasks facilitates more rapid learning and better performance in machine learning systems, and it may have important implications for cognition. The power and generality of human cognition may result from extracting common structure from the diverse but deeply related tasks we engage in throughout our lives.

Future Directions

1. In our analysis we analyzed the input modes of the first layer and the output modes of the second layer. In the future it will be important to explore modes that map into and out of the hidden layer, and what they imply about the representations at the hidden layer. This would also allow us to apply this analysis to deep networks.

2. Learning representations that reflect analogies may provide amortized inference about potential analogical structure in the world. Can this support explicit analogical reasoning?

Acknowledgments

This material is based upon work supported by the NSF GRF under Grant No. DGE-114747.

References

Bransford, J. D., & Schwartz, D. L. (1999). Rethinking Transfer : A Simple Proposal With Multiple Implications. *Review of Research in Education*, *24*(1), 61–100.

Day, S. B., & Goldstone, R. L. (2011). Analogical Transfer From a Simulated Physical System. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *37*(3), 551–567. doi: 10.1037/a0022333

Detterman, D. K. (1993). The Case for the Prosecution: Transfer as an Epiphenomenon. In *Transfer on trial: Intelligence, cognition, and instruction* (pp. 1–24).

Dienes, Z., Altmann, G. T. M., & Gao, S.-J. (1999). Mapping across Domains Without Feedback: A Neural Network Model of Transfer of Implicit Knowledge. *Cognitive Science*, *23*(1), 53–82. doi: 10.1207/s15516709cog2301

Dong, D., Wu, H., He, W., Yu, D., & Wang, H. (2015). Multi-Task Learning for Multiple Language Translation. *Acl*, 1723–1732.

Gentner, D. (2003). Why We're So Smart. In *Language in mind: Advances in the study of language and thought.* (pp. 195–235).

Gick, M. L., & Holyoak, K. J. (1980). Analogical Problem Solving. *Cognitive P*, *12*, 306–355.

Hinton, G. (1986). *Learning distributed representations of concepts.* doi: 10.1109/69.917563

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., ... Dean, J. (2016). Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *arXiv*, 1–16.

Kollias, P., & McClelland, J. L. (2013). Context, cortex, and associations: A connectionist developmental approach to verbal analogies. *Frontiers in Psychology*, *4*(NOV), 1–14.

Luong, M.-T., Le, Q. V., Sutskever, I., Vinyals, O., & Kaiser, L. (2016). Multi-task Sequence to Sequence Learning. *Iclr*, 1–9.

Mirsky, L. (1960). Symmetric gauge functions and unitarily invariant norms. *The Quarterly Journal of Mathematics*, *11*(1), 50–59. doi: 10.1093/qmath/11.1.50

Rogers, T. T., & McClelland, J. L. (2008). A simple model from a powerful framework that spans levels of analysis. *Behavioral and Brain Sciences*, *31*, 729–750.

Rusu, A. A., Gomez Colmenarejo, S., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., ... Hadsell, R. (2015). Policy Distillation. *arXiv*, 1–12. doi: 10.1038/nature14236

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *Advances in Neural Information Processing Systems*, 1–9.