Discrete neural nets and graph polymorphisms for learning

Charlotte Aten

University of Denver

2024 January 4

Discrete neural nets

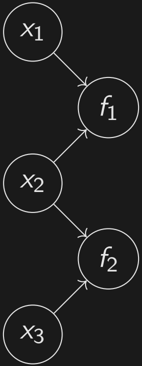
- Neural nets are a biologically-inspired framework for developing machine learning algorithms.
- For example, suppose we would like to make a tool that takes three digits as input and outputs their sum, without explicitly coding such a function.
- For example, we'd like to send $(2, 4, 6)$ to $(1, 2)$ since $2 + 4 + 6 = 12$.

Discrete neural nets

- We could create some input nodes x_1, x_2, and x_3, into which to plug our three digits.

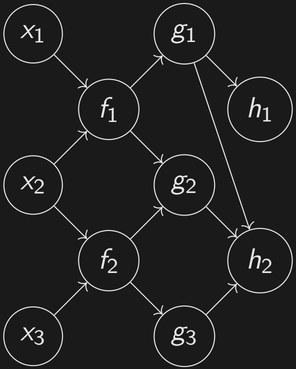
Discrete neural nets

- We could then add two output nodes, each of which carries an activation function. In this case, f_1 takes the values at x_1 and x_2, and is supposed to give us one digit of the sum of the input values.



Discrete neural nets

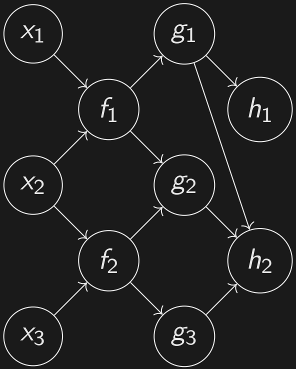
- We can even make something more complicated, where f_1 and f_2 get fed into another layer of activation functions, which in turn get plugged into h_1 and h_2.



Discrete neural nets

- This whole assembly can be thought of as a network of neurons which models the composite function

$$\begin{aligned}(x_1, x_2, x_3) \mapsto (&h_1(g_1(f_1(x_1, x_2))),\\ &h_2(g_1(f_1(x_1, x_2)), g_2(f_1(x_1, x_2), f_2(x_2, x_3), g_3(f_2(x_2, x_3))))).\end{aligned}$$



Discrete neural nets

- What functions should we choose for the activation functions?
- If we made really smart choices ourselves, we would basically be writing the function we decided we would be too lazy to write.
- On the other hand, if we choose any random functions, we would likely not obtain a function that maps (a, b, c) to the digits of $a + b + c$.

Discrete neural nets

- Learning with neural nets means choosing some activation functions to start, then tweaking them somehow to improve the empirical correctness of the modeled function.
- This has its own problem: overfitting.

Discrete neural nets

- It is easy to train a neural net to perfectly map $(1,2,3)$ to $(0,6)$, $(0,3,5)$ to $(0,8)$, and $(2,2,3)$ to $(0,7)$, while still totally failing to map $(3,4,5)$ to $(1,2)$.
- Often, the neural net will just take on any values outside of its training examples.

Discrete neural nets

- One way to stop this from happening is to restrict our possible activation functions.
- For instance, if all of our activation functions had to be linear then our neural net could only model linear functions.
- This is because linear functions are closed under composition.

Polymorphisms

Definition (Polymorphism)

Given a structure $\mathbf{A}$ we say that a homomorphism $f\colon \mathbf{A}^n \to \mathbf{A}$ is a *polymorphism* of $\mathbf{A}$.

- For example, a group homomorphism $f\colon \mathbb{Z}^n \to \mathbb{Z}$ is a polymorphism of the group $\mathbb{Z}$.

Example: binary images

Definition (Hamming graph)

Given $n \in \mathbb{N}$ we define the *n-Hamming graph* to be

$$\mathbf{Ham}_n := (A_n, \{\, (a_1, a_2) \in A_n^2 \mid d(a_1, a_2) \leq 1 \,\})$$

where A_n is the set of all $n \times n$ images consisting of black and white pixels only and d is the Hamming distance.

Example: binary images

- Endomorphisms and automorphisms of $\mathbf{Ham}_n$ are easy to come by.
- The dihedral group acts on $\mathbf{Ham}_n$.
- Any bitwise operation with a fixed image will yield an endomorphism of $\mathbf{Ham}_n$.

Example: binary images

- Higher-arity polymorphisms are harder to come by.
- These are graph homomorphisms

$$f\colon \mathbf{Ham}_n^k \to \mathbf{Ham}_n\,.$$

Example: binary images

Definition (Multi-linear indicator)

Given $b \in B_n$ and $c \in A_n^k$ the *multi-linear indicator polymorphism* for (b, c) is the map $g_{b,c} \colon A_n^k \to A_n$ given by

$$g_{b,c}(a_1, \ldots, a_k) := \left(\prod_{i=1}^{k} a_i \cdot c_i \right) b$$

where $x \cdot y := \sum_{i,j} x_{ij} y_{ij}$ denotes the standard dot product in $\mathbb{F}_2^{[n]^2}$.

Example: binary images

- I have a preprint out which contains a discussion of some even more involved/interesting polymorphisms.
- I am working with some students to extend these constructions to higher-arity relations and combine these ideas with some results in my PhD thesis.

More info

- You can find a link to this paper (and in turn the corresponding code) on my website: aten.cool

References

- Charlotte Aten. “Finite Generation of Families of Structures Equipped with Compatible Group Actions.” PhD thesis. 2022, p. 61. ISBN: 9798363518454
- Charlotte Aten. “Discrete neural nets and polymorphic learning.” In: *arXiv e-prints* (July 2023). arXiv: `2308.00677 [cs.NE]`