A categorical semantics for neural nets

Charlotte Aten

University of Denver

2023 November 29

Introduction

- Discrete neural nets
- Multicategories
- Neural nets as functors
- Multicategories?
- Structures
- Structures as data

Discrete neural nets

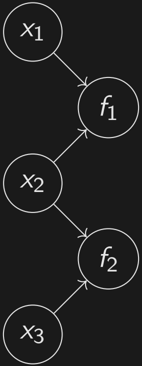
- Neural nets are a biologically-inspired framework for developing machine learning algorithms.
- For example, suppose we would like to make a tool that takes three digits as input and outputs their sum, without explicitly coding such a function.

Discrete neural nets

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

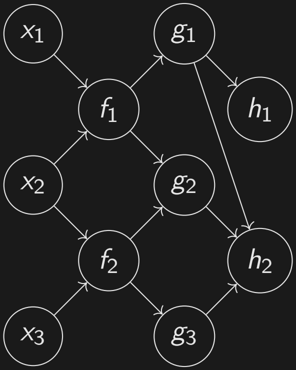
Discrete neural nets

- 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

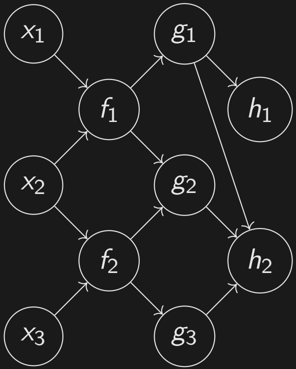
- We can even make something more complicated, where f_1 and f_2 get fed into another layer of activation functions, which in turn get plugged into h_1 and h_2.



Discrete neural nets

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

$$(x_1, x_2, x_3) \mapsto (h_1(g_1(f_1(x_1, x_2))),\\ h_2(g_1(f_1(x_1, x_2)), g_2(f_1(x_1, x_2), f_2(x_2, x_3), g_3(f_2(x_2, x_3))))).$$



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 make it impossible.
- 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.

Discrete neural nets

- In general, if we have an object S in a category with finite products, we can take our activation functions to be morphisms $f\colon S^n \to S$ for various n. This set of morphisms, the *polymorphism clone* of S, is closed under composition.

Discrete neural nets

- If we choose S to be an object relevant to our learning task, we will find that a neural net whose activation functions are polymorphisms of S can only learn "reasonable" functions.
- For instance, if we take S to be a G-set for some group G, the polymorphisms of S are just the G-equivariant operations $f\colon S^n \to S$ where

  $$f(\sigma x_1, \ldots, \sigma x_n) = \sigma f(x_1, \ldots, x_n)$$

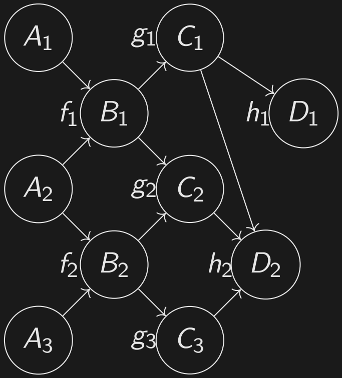
  for $\sigma \in G$.

Multicategories

- This kind of general observation suggests that neural nets have a nice functorial description.
- The appropriate functors here are functors between multicategories (or perhaps something like them), for which I will now give some background.

Multicategories

- Let's think of this picture as showing us various objects, where each one has a single "*n*-ary morphism" going into it.



Multicategories

A multicategory $\mathscr{C}$ has:

- a collection of objects $\mathrm{Ob}(\mathscr{C})$,
- for each tuple of objects $(A_1, \ldots, A_n)$ and each object B, a set of morphisms $\mathscr{C}(A_1, \ldots, A_n; B)$ from $(A_1, \ldots, A_n)$ to B,
- for each object A an identity morphism $\mathrm{id}_A \colon (A) \to A$, and
- a law of (generalized) composition by which we can form morphisms such as $g_2[f_1, f_2]$ in our picture.

Multicategories

A multicategory $\mathscr{C}$ must:

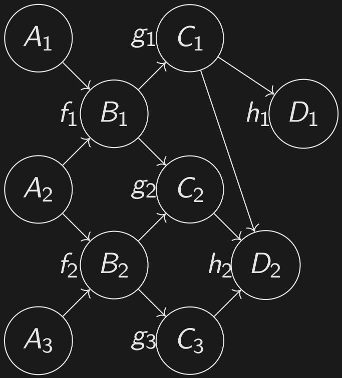
- satisfy a generalized associative law and
- have the identity laws $f[\mathrm{id}, \ldots, \mathrm{id}] = f = \mathrm{id}[f]$.

Multicategories

- Operads are multicategories with only one object.
- Functors between multicategories are defined analogously with those for categories.

Neural nets as functors

- If we look at our picture again, we can see that we have a multicategory.
- This kind is special because the morphisms only "go one way".



Neural nets as functors

- We'll say that an *architecture* is a multicategory $\mathscr{A}$ where $\mathrm{Ob}(\mathscr{A})$ is finite and there is at most one morphism in each hom set $\mathscr{A}(A_1, \ldots, A_n; B)$.
- These are basically "finite multiposets".

Neural nets as functors

- Given an architecture $\mathscr{A}$, we say that a functor $\mathcal{N}\colon \mathscr{A} \to \mathscr{C}$ is a $\mathscr{C}$-valued *neural net*.
- If $\mathscr{C}$ is the (multi)category of smooth manifolds and $\mathcal{N}\colon \mathscr{A} \to \mathscr{C}$ sends all objects to $\mathbb{R}$, we get a classical neural network.
- If $\mathscr{C}$ is the (multi)category of G-sets for some group G and $\mathcal{N}\colon \mathscr{A} \to \mathscr{C}$ sends all objects to a G-set S, we get a G-equivariant neural net.

Multicategories?

- There is something I have been sweeping under the rug: duplication of arguments.
- Technically, in a multicategory if we have $f_1 \colon (A_1, A_2) \to B_1$, $f_2 \colon (A_1, A_2) \to B_2$, and $g \colon B_1 \times B_2 \to C$, the composite is

$$g[f_1, f_2] \colon (A_1, A_2, A_1, A_2) \to C,$$

not

$$g[f_1, f_2] \colon (A_1, A_2) \to C.$$

Multicategories?

- In order to get something like this, we could use diagonal maps like $\Delta\colon A \to A \times A$ in a Cartesian category.
- More generally, we could ask for a certain kind of monoidal category, such as a PROP, or perhaps a polycategory.
- I'm currently looking into using something slightly different.

Multicategories?

A multiclone $\mathscr{C}$ has:

- a collection of objects $\mathrm{Ob}(\mathscr{C})$,
- for each tuple of objects $(A_1, \ldots, A_n)$ and each object B, a set of morphisms $\mathscr{C}(A_1, \ldots, A_n; B)$ from $(A_1, \ldots, A_n)$ to B,
- for each tuple of objects $(A_1, \ldots, A_n)$ and each $1 \leq i \leq n$ a projection $\pi_i \colon (A_1, \ldots, A_n) \to A_i$, and
- a law of (generalized) composition by which we can form morphisms such as $g_2[f_1, f_2]$ in our picture.

Multicategories

A multiclone $\mathscr{C}$ must:

- satisfy a generalized associative law and
- have the projection laws $f[\pi_1, \ldots, \pi_n] = f = \pi_1[f]$.

Structures

- Another nice family of examples of neural nets comes from graph theory.
- Let $\mathrm{Ham}_{n,k}$ be the *Hamming graph* whose nodes are k-ary relations on $[n] := \{1, 2, \ldots, n\}$.
- Two nodes of $\mathrm{Ham}_{n,k}$ are adjacent when their Hamming distance from each other is at most one.

Structures

- In a recent preprint I described an infinite family of nontrivial polymorphisms of $\mathrm{Ham}_{n,2}$ which can be computed efficiently.
- These are good candidates for using as activation functions, so my students and I are experimenting with them in basic learning tasks using MNIST.

Structures

- In my PhD thesis, I gave a categorification of Bourbaki's notion of a (generalized) relational structure.
- What I actually did was quite general, but I will describe a special case that includes any reasonable finite data structure.

Structures

- Consider a functor $\rho\colon \mathscr{I} \to [\mathrm{Set}, \mathrm{Set}]$.
- We have a corresponding functor $\rho_A\colon \mathscr{I} \to \mathrm{Set}$ for each set A where $\rho_A(N) := (\rho(N))(A)$.
- We think of subfunctors of ρ_A as structures of signature ρ with universe A.
- (There are some other technical details I am suppressing.)

Structures

- There is always (under those other assumptions I'm leaving out) a category Struct^ρ of structures of signature ρ.
- Perhaps we could consider neural nets $\mathcal{N} \colon \mathscr{A} \to \mathsf{Struct}^\rho$.
- Unfortunately, we would have to find polymorphisms for such structures.
- Even finding one nontrivial polymorphism is NP-hard, in general.

Structures as data

- Note that we are usually not handed a structure we want to preserve when facing a new learning task.
- We are usual given finite data structures as the **training data**, but understanding the relevant properties that our neural net should preserve may be challenging.

Structures as data

- The Yoneda embedding is very helpful here.
- Given structures $\mathbf{A}$ and $\mathbf{C}$ with universes A and C, respectively, observe that

$$\mathsf{Struct}^{\rho}(\mathbf{C}, \mathbf{A}) \subset A^{C}$$

  is a $|C|$-ary relation on A.
- This gives a fully faithful functor

$$\mathsf{Struct}^{\rho} \to \mathsf{Struct}^{\rho^{\mathsf{Set}}}.$$

Structures as data

- Using this functor, we can see that there is an embedding of $\mathrm{Net}(\mathscr{A}, \mathsf{Struct}^{\rho})$ into $\mathrm{Net}(\mathscr{A}, \mathsf{Struct}^{\rho^{\mathsf{Set}}})$.
- In nice cases, we have a "truncation" (or reduct)

$$\psi \colon \mathsf{Struct}^{\rho^{\mathsf{Set}}} \to \mathsf{Struct}^{\rho'}$$

where a ρ'-structure only has finitely many basic relations.

Structures as data

- This means that if the training data for our neural nets consists of such relational structures, we might as well think of our training data as being vertices of the Hamming graph $\mathrm{Ham}_{n,k}$.
- In this sense, the polymorphisms of the Hamming graph are already enough to describe a polymorphic learning algorithm for any kind of training data one might use.