

Re: Hawkins ideas on building AI's

Source: <http://sci.tech-archive.net/Archive/sci.cognitive/2004-10/0748.html>

From: dan michaels (*feedbackdroids_at_yahoo.com*)

Date: 10/27/04

Date: 27 Oct 2004 09:37:27 -0700

bkaz_@hotmail.com (bkaz) wrote in message

news:<def186b4.0410262002.58ea8e92@posting.google.com>...

> > *You're right about the "variables of patterns", and this is also how*
> > *the 30 low-level to mid-level vision areas also work. The basic*
> > *operations in each of these areas is probably specified genetically,*
> > *but in a rather diffuse manner, and then they are fine-tuned via*
> > *experience, during early development. Without some structuring, it's*
> > *hard to imagine the 30 areas could simply self-organize to in every*
> > *individual to perform the same functions in each case, simply via*
> > *experience. The probability of this seems very low. And even if you*
> > *built a computer system with a very fast memory, you would still have*
> > *to specify the basic operations to occur in the different levels.*
> > *Again, I doubt you would ever get the correct operations simply out of*
> > *self-organization.*
>
> *I'm not sure, but I don't think it's relevant for AI. The problem is*
> *not the memory speed per se, it's the difference between memory*
> *formation & memory retrieval speeds, - if there's no significant*
> *difference it should be faster & more economical to form variables for*
> *each pattern than to store them permanently in specialized areas &*
> *share them among patterns.*
>

We seem to be talking about 2 different things here. It's not a matter of memory storage vs retrieval speeds. And I wasn't talking about specialized memory areas. I was talking about a hierarchy where you have specialized pre-processing areas followed by separate and more generalized memory areas.

=====

> > *Regards the generalization stuff, since you read Hawkin's book*
> > *already, he has several multi-level sensory-motor-chain diagrams in*
> > *later chapters ... doesn't his model indicate an increasing level of*
> > *abstraction/generalization as you go up the chain on the one side, and*
> > *back down it on the other? And that processing at the lower levels is*
> > *more specific in both cases?*
>
> *There's a contradiction, you can't build a hierarchy around generality*

- > **&* novelty at the same time: novel patterns by definition lack proven*
- > *generality: previously accumulated match.*
- >

I was doing just the opposite – talking about having specialized preprocessing areas followed by more general memory areas. This way you have less specificity and more abstraction as you go up the chain of processing. There's no contradiction.

=====

- > > *Here's where I differ with Hawkins (or with evolution?), my hierarchy*
- > > *is not necessarily of composition, or of novelty, it is of generality:*
- > > *accumulated/projected match of constituent patterns.*
- >
- > > *You may be right, bio-vision probably does those indiscriminate*
- > > *transforms, sort of like image compression, but that's only because*
- > > *it's too slow (per 'processor') to do individual pattern recognition,*
- > > *which would be far more logical and 'cost-efficient'.*
- >
- > > *Yeah, you can make a case that speed is the problem, but just the*
- > > *same, all current CV systems use a multi-level hierarchy of*
- > > *operations. Later ones in the chain build upon the output of the*
- > > *previous ones.*
- >
- > *Needless to say, none of those CV systems scale effectively enough to*
- > *do anything interesting.*
- >

Yes well, does this mean the hierarchical scheme needs to be abandoned, or better improved upon?

=====

- > > *Just because you get a bigger faster computer doesn't*
- > > *necessarily mean you can get rid of this operational hierarchy. What*
- > > *would a single-level flat-architecture algorithm be? Template*
- > > *matching? Store every single possible case? Doubtful.*
- >
- > *You seem to be describing the exact opposite of my approach :).*
- > *My hierarchy has indefinite number of levels, but the difference among*
- > *them is strictly relativistic. They all use the same*
- > *comparison-projection algorithm per variable of an input pattern, but*
- > *those patterns have sequentially expanding syntax: spectrum of*
- > *expected variables generated by lower-level comparisons.*
- >

So, they all use the same algorithm, and I assume abstraction will increase as you go up the levels. The system with pre-processing areas is more specific at the lower levels, which pre-selects what is sent to the higher levels, so in this case the algorithms are somewhat different from low to high. Can you explain how your comparison-projection algorithm works in more detail? [see below].

=====

- > *I will post an intro to my approach on a separate thread, but forgive*
- > *me if it's not very specific, – I feel I'm close enough to*
- > *implementation to keep the core stuff proprietary :).*
- >
- > *Boris.*

Maybe some general details, maybe regards what is the alternative to doing a lot of pre-processing :).