

# Intelligence by definition

**Source:** <http://sci.tech-archive.net/Archive/sci.cognitive/2004-10/0728.html>

---

**From:** bkaz (*bkaz\_\_at\_hotmail.com*)

**Date:** 10/27/04

Date: 26 Oct 2004 21:20:04 -0700

Intelligence is not a content-specific ability, – we can learn anything. Since inputs & outputs can't be assumed, the method of intelligence is derivable only from the functional definition of learning.

Mainstream position of artificial intelligentsia seems to be that intelligence is a lot of different things, which I think is largely responsible for the pathetic state of AI. Any general term refers to different things, but they must have something in common or we wouldn't use it to describe them all.

I define intelligence as an ability to build a predictive model of environment by discovering patterns in past inputs & interactively projecting them. That includes planning, which technically is a self-prediction.

There are plenty of pattern recognition methods, but none scale effectively with the range of inputs. I believe that's because they don't consistently define match as a predictive potential or direct search accordingly, & aren't selectively incremental in their power/complexity.

Definitions:

Match is a partial identity: a sum of binary AND between uncompressed comparands (strings of 1s),– the only way to quantify similarity positively (though it can also be seen as a double negative: a compression of magnitude by replacing the greater input with the smaller).

Partial identity relative to input's magnitude is predictive: match & difference are projected equally. Complexity theory defines similarity as a compressibility of records, which is subjective & somewhat arbitrary as it doesn't scale smoothly. In my approach it affects costs, not benefits, of prediction.

A pattern is a sequence of matches between inputs (sets of variables) & projections (last input scaled by vectors). Vectors are values that

convert the previous input into the last one by any operation:  
addition, multiplication, elevation to power...

Vectors are discovered by comparison between the inputs: reverse scaling of corresponding order: subtraction, division, extraction of a logarithm... Comparison is made over the distance between the compared inputs, which is a second type of vector: the one scaling inputs' coordinates.

A pattern is predictive if a match of inputs is associated with that of their coordinates. A priori such coordinates must be Spatio-Temporal: physical causality is S-T continuous & similarity declines with distance/delay. Then a secondary search is ordered by any sufficiently predictive variable type.

Pattern's Predictive\_Value is its projected cumulative match to future inputs. Its' Net\_Value = Predictive\_Value - Average\_Value(produced at the same cost of memory + operations in the past). Patterns of negative Net\_V are deleted, thereby continually increasing predictive power of a system.

Scaling:

I suggest that patterns(inputs+vectors) should hierarchically scale their range of search & syntactic complexity as long as such scaling increases their Net\_Value, which depends on algorithmic complexity of the input stream (lower-level patterns).

Scaling must start from the simplest possible inputs: sensory, defined by the minimum of 2 variables: Value & Coordinate, indicating respectively content & target area of projection. They're processed by the simplest possible method: run-length encoding, but modified to be scalable:

Example: grey-scale video processing:

Pixels adjacent in a scan line are compared to form 1D patterns, adding a set of new variables for every old one: Brightness forms Power of comparison, average Match, & cumulative Vector; Coordinate forms Period of comparison, Number of matching inputs, & cumulative Distance between them.

These new variables are necessary to evaluate:  $M*N$ , & to project:  $V*Power *D*Period$ , the pattern for expanded search, increasing range of Coordinate & possibly of Brightness. Such syntactic expansion repeats with each new level of search for each surviving (predictive) variable of an input pattern.

Vectors from sequentially successful comparisons are also compared, forming higher-derivation sub-patterns, each with the same set of new evaluation/projection variables. The resulting pattern potentially

forms a hierarchy of derivation, with multiple sub-patterns (offsets) on each level.

Subsequently patterns are compared/integrated over increasing dimensionality/distance: vertically adjacent 1D Ps are compared to form 2D Ps, compared to form 3D Ps, compared to form 4D(dynamic) Ps, compared over hierarchically increasing distance (no longer adjacent) to form discontinuous patterns.

Thus, memory forms a search hierarchy, each level composed of full-search units with a fixed range. Units' range expands with elevation, up to a single top level global search unit. Higher levels' range & syntax increase cost, so only patterns of corresponding value are elevated to search them.

Patterns are deleted if the unit's sources shift beyond their range of effective prediction.

Given sufficient hardware, the top level patterns will eventually achieve & surpass complexity & generality of natural language concepts, which means that the system will surpass the predictive power (effective intelligence) of human mind.

Notes & Recapitulation:

This approach may seem primitive, but that's precisely the point, all the complexity must be learnable/forgettable, otherwise it becomes a bias.

Generalization is a reduction, & intelligent system that starts with higher-level data (especially natural-language concepts) can't recreate the semantic context, that is generalized real-world 4-dimensional patterns that most words in the language stand for.

Most of the effort in compression now seems to be in developing methods for symbolic data, which I think is unfortunate. There is only so much that one can compress out of symbols without knowing what they stand for. Real compression must be selectively lossy, & a proper selection is only possible if we can assess value of the data within its semantic context.

Scalability is of the essence, & a truly scalable method must start from minimal input complexity: the limit of resolution of raw sensory data, from which all higher data types are ultimately derived.

Power/cost of comparison should be proportional to inputs' accumulated match, which is initially 0.

Thus, the power must also start from the very minimum: arithmetic subtraction between adjacent inputs.

I don't believe in combining different methods because cognition deals with the unknown, – we can't a priori split it into different areas, except to the extent that they're sensor/hardware specific, or levels,

except that patterns' syntactic complexity will increase with sequential search expansion.

Any methodological differentiation must be determined by the success of lower-level methods, encoded in patterns produced by them. The 'types' of inputs are ultimately the types of empirical objects they represent, an intelligence should be able to learn them on its own.

Would appreciate any comments.  
Boris.