

Saffron’s Cognitive Computing Platform

**Reasoning by**

**Cognitive Distance on an**

**Associative Memory Fabric**

Authors:

Dr. Paul Hofmann, Chief Technology Officer

Dr. Manuel Aparicio IV, Co-founder, Chief Memory Maker

December 2013

Saffron Cognitive Computing Breakthrough

In *Surfaces and Essences: Analogy as the Fuel and Fire of Thinking*, Hofstadter and Stander (2013) make a persuasive case that cognition is about learning and reasoning to combine current information with prior knowledge to understand what’s similar, different, and anticipate what will happen next.

This “memory-based reasoning” (weighted "associations" or "links" between people, places, things, events ) is embodied into Saffron MemoryBase (SMB).

SMB harmonizes hybrid data combining it with instance-based learning. It can be thought of as both an efficient compressor of raw data and a massive correlation engine that calculates the statistical probabilities between the edges of a semantic graph representing the graphs multivariate non-linear dependencies.

Cognitive Distance (CD) based on Kolmogorov complexity measures similarity between objects to discern signal from noise finding regular pattern in the data.

Associative Memories (AM) are as fundamental and universal as the von Neumann architecture and they are mathematically well described. The training weights determine the fixed points of their Lyapunov function.

The deterministic weights of AMs allow for direct interpretation of the trained model; plus, AMs are parameter free. Conversely, the SVs of an SVM model, the weights of the hidden layer of ANN, and the weights of random forests cannot be interpreted directly for the original problem. These machine learning algorithms are not parameter free. A very knowledgeable human being has to find the right kernel (SVM), define the right size of the hidden layer (ANN), etc.

It seems very natural to combine the universal, parameter free, non-functional and deterministic nature of AMs with the power of universal a priori probability distributions defined by algorithmic information theory. Kolmogorov (K) complexity allows deriving a universal distance measure, the so-called Cognitive Distance (CD). CD is ideal for reasoning by similarity. CD tells us for example, how close two objects are, like a cowboy and a saddle. We use AMs as a very efficient compressor for calculating an approximation to the K complexity. We will show how we use this approach for search that understands context as well as finding patterns in the trajectories of objects in information space and in prediction using classification.

Unlike manual knowledge engineering and model fitting, SMB instantly learns on the fly as new data arrives. Coincidence matrices of SMB include massive numbers of frequencies and their distributions. The ease and speed of computation over one cell, any submatrix, or aggregation of matrices across a massive memory base are due to the nuts-and-bolts of Saffron’s associative implementation, not just to instantly learn but to instantly recollect. For each of the correlation, convergence, and classification applications, the implementation of SMB is highlighted to show how easily and quickly this cognitive implementation supports each cognitive process, all based on distance.

# Outline of Paper

This paper will explain the operational realization of SMB for discovering the regularities of observed data. The paper introduces mathematically founded metrics to measure such regularities, how these metrics are computed in a scalable way, and how these computations are useful in practice through the following:

* **Reasoning by Similarity.** What do we mean by “associative”, and how do we connect semantics and statistics? The semantic meaning has been limited to the definition of graph connections, while the statistical meaning represents a generalized form of correlation. Hybridized value in any application requires both.
* **Correlation and Convergence.** Applications of memory-based reasoning are innumerable. We understand by learning from samples, discovering the structure in data, or equivalently finding dependences in the data. Cognitive Distance (CD) is a universal measure for the dependence between objects. As one elemental example, it can be used to reduce the “noise” in sense-making. CD is then extended to convergence, defined by changes in distance over time. Convergence illustrates the deeper meaning of associative “trending” between things: how things are moving toward or away from each other in information space over time.
* **Classification and Prediction**. Patterns emerge when objects interact. When many data samples are collected to form a class of things, such as animals that are reptiles separate from those that are mammals, these ensembles form patterns of what it means to be a reptile or mammal. Classification measures the similarity (i.e. distance) of a given thing to one or more classes of things. Classification is the bedrock of machine learning and its application to prediction. This solves the real questions that businesses ask, “Is this situation good or bad?” or “If I go left, what are the likely outcomes?” We use CD to reason by similarity and to classify things.
* **Appendix Mathematical Foundations.** Kolmogorov Complexity measures regularities of single objects, while Cognitive Distance – a generalized divergence, measures regularities between objects. A distance measure between objects allows us to formalize the intuitive concepts of similarity and analogy; how far or how close one thing is to another. We use information distance (Kolmogorov) and its brother, entropy distance (Shannon), to describe such similarity. We show how entropy distance is related to Hamming and Jaccard distance, which has a long and deep history in associative memories.

Reasoning by Similarity

The “associative” word continues to gain currency in the analytics market. The Associative Model of Data, notable defined by LazySoft several year ago (Williams, 2000), is increasingly being used to describe node-and-link stores of explicitly materialized graph knowledge as different from relational databases. QlikTech was one of the earliest successful companies to use the word “associative” as connecting the dots across data elements. However, Qliktech now only refers to its “associative experience” since industry analyst Curt Monash correctly limited QlikTech’s claim to a method of user interaction – not an associative store per se (Monash, 2010). More recently, “associative” is also used by SAP’s Associative Analytics, again meaning to connect the dots through the visualization of underlying data.

SaffronMemoryBase (SMB) also “connects the dots” but in a truly associative way. When used as a Q&A machine, a memory base reads everything, building a full semantic graph with connection counts at the edges. Given a query, Saffron returns associated entities and the documents that warrant these associations. In this definition of associative, SMB is a hyper-graph, implemented as a hyper-matrix.

However, the full meaning of “associative” is deeper than the node-link-node connections of a graph. How we know and what we know is more than observing and remembering connections. As previously introduced in “Your Brain is Cognitive”, there are two formal definitions of “associative” that are unified by SMB. Both definitions answer how we learn and how we reason:

* **Semantic Connections**. One meaning addresses graph-oriented connectivity, the “connections” between things. We say that Joe is associated with Mary when we read a sentence such as “Joe married Mary”. There is a connection between the two persons, including a context of the connection to form a semantic “triple”.
* **Statistical Dependencies**. Another meaning is statistical, defined as dependency, a generalized non-linear form of correlation between things. We say that Joe is statistically dependent on Mary, when Mary tends to be present whenever Joe is present and tending to be absent when Joe is absent. There is a correlation between the two persons, based on observed frequencies. Conditional dependency defines a statistical triple with additional context given between two things.

This paper focuses on using the universal Cognitive Distance for reasoning by similarity/analogy on top of the associative memory fabric. Applications include the two general areas of advanced analytics:

* **Sense Making**. Sense making is moving past “connecting the dots” to also become more predictive, or to use a better word, “anticipatory”. For example, if we see a growing dependency of a competitor-supplier relationship over time, we can better anticipate where the relationship is going and what it might mean to our own business.
* **Predictive Decision Making**. To make decisions, we must estimate the current situation (diagnoses), prescribe possible actions, and predict likely outcomes. These estimates are a result of reasoning by similarity.

Correlation and Convergence

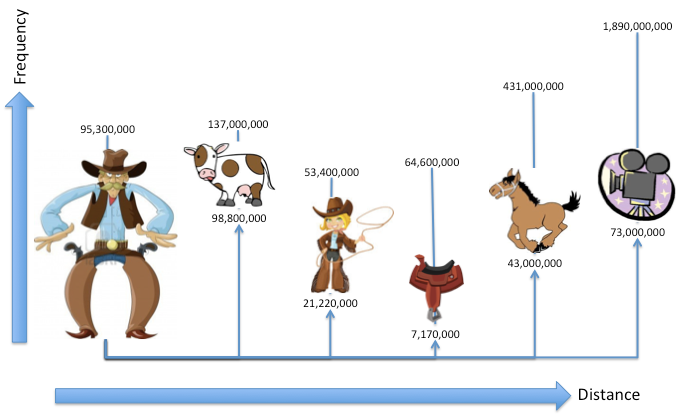
# From Correlation to Similarity

When analyzing big data sets we look for relationships between entities – people, locations, organizations, products, stocks, etc.. For example in finance, we want to know how one stock moves in relationship to another; colloquially we ask, how the stocks are correlated to each other. Formally speaking correlation is limited to the linear dependence between attributes. “Dependence” is the more general definition, but we will use “correlation” as more generally understood. One can think of SMB as a massive correlation engine. SMB builds a semantic graph from the raw data and stores the co-occurrences (associations) as counts on the edges of this graph.

The strength of the links (counts) becomes a natural filter in order to analyze these huge graphs. For example, we can rank order the outgoing links of an entity like John Smith according to their strength, and thus discover how John is connected to other entities like people, or cities. In cognitive computing we go beyond this basic counting and rank order approach in order to answer for example the question “closeness” between the cowboy to the horse, the cow, the saddle, or the movies.

Ideally, we want to use a distance measure that is domain independent, parameter free, measuring similarity between single objects as well as ensembles. Surprisingly such a distance metric exists. It is the Cognitive Distance (CD) based on Kolmogorov complexity. The CD comprises all other (computable) similarity measures. For example, when looking for the similarity of music pieces, we could come up with many measures. The pieces from the same composer are similar; so are pieces of the same genre, or music that shows similar beat patterns over time. Cognitive Distance (CD) is a universal similarity measure minorizing all other possible distances.

Imagine a Google search for “cowboy saddle” that returns the count of documents with both terms. This is the count of xy. Searching for “cowboy” and then “saddle” returns their independent counts, of x and of y. A search of “cowboy movie” returns more results than “cowboy saddle” as stronger xy. If based purely on raw counts, we would say that movie has a stronger association. However, similarity measured by CD is a much deeper concept based on the probability distribution. As a side effect of its universality, CD cannot be computed due to the Turing halting problem. However, this is not a practical impediment. We have used an approximation here for using counts from search engines called the Normalized Web Distance (Cilibrasi et al., 2007). Such simple Web searches show quite impressively how useful normalized information distance can be to lift the cognitive closeness of two concepts (“saddle” to “cowboy”) away from the popular, high frequency concepts (“movie”), which tend to be less relevant.



**Fig. 1 Web Distance between cowboy and similar concepts.** The document counts for each term and the document counts for each term with “cowboy” are shown as frequencies, low to high. For example, “saddle” is indexed to over 64 million documents, co-occurring in the same document with “cowboy” over 7 million times. Based on coincidence counts, “cow” and “movie” are most strongly associated with “cowboy”, but when distance is computed, “cow” remains closest to “cowboy” and “movie” becomes more distant. In the most extreme comparison, the count for “cowboy saddle” is an order of magnitude lower than “cowboy movie”, but “saddle” is cognitively closer to “cowboy”. The distance was calculated using the normalized web distance: DIxy= max {log(x) , log(y)} - log(x,y) ) / ( logN - min{log(x) , log(y)}.

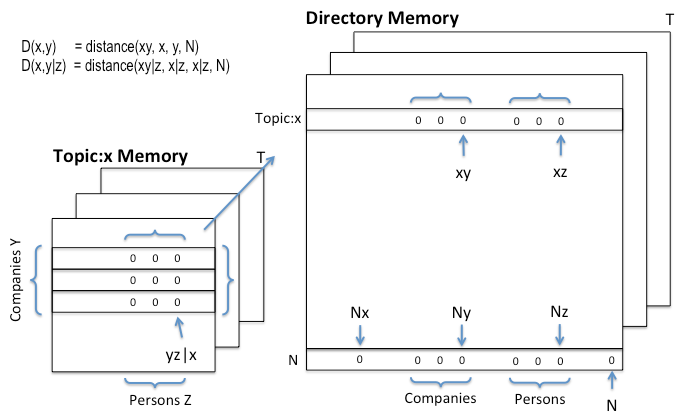
All we need for this calculation are the counts of the terms x and y, along with the associative count, xy, and the total number of observations, N. SMB has all these counts and can access them in real time from its sparse matrices.

In the next section we will see in detail how this is done.

# Matrix Machinery for Correlations En Masse

Several properties described in “Your Brain is Cognitive” are critical to compute such correlations, especially en masse. The implementation of a scalable associative memory includes row partitioning for row dominance in queries, attribute grouping for locality of answers, global sorting to aggregate across answers, and streaming aggregation to provide answers quickly.

These properties enable the fast recollection of “raw” connections and counts. For example, a query can address any one matrix or any one row of the matrix, which streams all the values of any collocated attribute. This recollection of raw connections supports simple queries as in a graph stores. The SMB answer also includes the raw coincidence count for every attribute. When two or more matrix-rows need to be aggregated, all attributes are sorted in a globally consistent order. This streaming aggregation also computes statistics over the stored counts, nearly as fast as returning the counts themselves, as will now be described. Correlation (i.e. dependence) is the first use case because it is so elemental, computing the similarity or distance between two things. Each cell is a “point” in the matrix, and the computation of distance at each point is called “point-wise”. This distance filters the high frequency “noise” that is often found in faceted entity search, particularly at scale where popular concepts have high association counts with little information.



**Fig. 2 Memory organization of required counts.** The computation of pair-wise distance requires the coincidence xy count and two independent counts for x and y along with N, the total number of observations. These counts are highlighted in a pair-wise Directory Memory as xy, Nx, Ny, and NN respectively, for topic:x and a company:y for example. The row for N includes independent counts of each attribute as well as the total count of observations, NN. For triples, the conditional pair-wise count of xyz is found within the conditional memory. The other conditional terms, xy and xz, are found in the Directory Memory. For both Directory Memory pairs and contextual Memory triples, the matrix dimension within each memory can be used for “slicing” over time. Each memory can be queried for a single count as for the N of x, for a vector as for companies associated with topic:x, or for an entire contingency table as for companies associated with persons within topic:x.

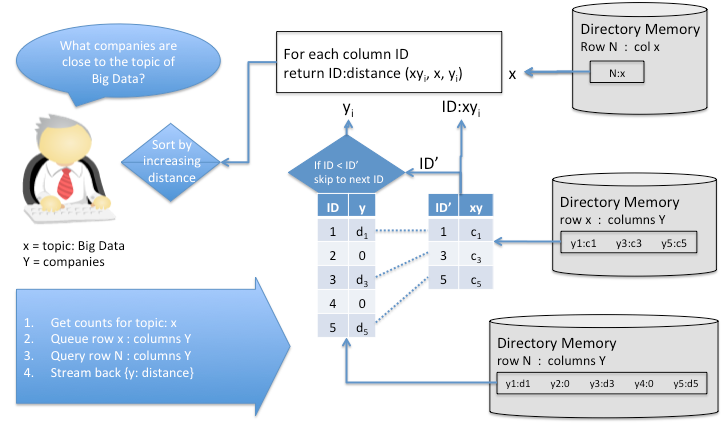
# Calculating The Cognitive Distance Between Two Things

We illustrate how SMB calculates CD based on Kolmogorov complexity as we have seen above for the cowboy example, but now we use SMB as a compressor instead of a web search engine.

CD is defined as CD = max {K(x|y), K(y|x)} / max {K(x), K(y)}; for details see the mathematics appendix. Assume we use a Shannon-Fano code to encode an entity that occurs with probability p(x) = - log (counts(x)/N) in order to get a short description of entity x, N being the total observation count. Substituting p(x)~ K(x) in the equation for CD we get a good approximation to CD considering SMB is a compressor

DIxy= max {log(x) , log(y)} - log(x,y) ) / ( logN - min{log(x) , log(y)}.

Let’s take for example the simple query of one term such as “topic: big data” (x) asking to see a list of associated companies (Y). In this case the Directory Memory (DM) has all we need; x,y and xy. The association (co-occurrence) counts ci of each company (yi) between the topic x are all within one row in DM (row x : columns yi), which accelerates query response time considerably. We also need to collect the non-zero marginal counts (di) for each company (yi) from the DM. The total number of observations over all memories (N) is a system constant. Streaming aggregation in SMB is key to combining the distributions of coincident (xy) counts and independent (x and y) counts for each point-wise computation.



**Fig. 3 Cognitive Distance by streaming aggregation of sorted queues.** All pair-wise associative counts (xy) as well as marginal counts (x,y) are contained in the Directory Memory (see Fig. 2). A distance aggregator building the vector ID:distance (xyi ,x,yi) for i=1,n is attached to 2 queues. The row for topic x as well as the row for N of the marginal counts yi feed the queues that build ID:count vector (ID:ci). Because the attribute IDs in both queues are in common order, the aggregator can sequentially find matching IDs, compute the distance, and iterate each ID until the queue is empty. Sorting the aggregation results is optional.

The globally sorted IDs and queues in SMB enable efficient access to compute vast numbers of such distances. As illustrated, any one point-wise computation aggregates the counts for computing distance from across the DM. However, we rarely care about only one answer. When asking for a list of companies associated with a topic, the answer set is more likely to range from 10 to 10,000 or more. Fortunately, SMB provides:

* **Attribute Grouping.** When given x and looking to evaluate every possible answer in set Y, all answers over Y are co-local. For computing distance, all xY coincidence counts as well as Y independent counts are physical together, compacted in a hyper-space compression method including relative ID encoding with zero run length encoding. Because of this compression, response time is largely independent of answer set size whether 10 or 10,000. The aggregator is attached to these two respective memory locations so that all the required attribute IDs and counts sets for xY and Y are brought together.
* **Global Sorting.** Respective IDs and counts for the two sets are also in a common order. When asking for persons associated to a topic, the xY coincident counts between persons and the topic will be a subset of all the persons. Because we only care about persons connected to the topic, the aggregator goes down every answer in the xY queue. The aggregator scans down the Y queue to find the next matching ID to the xY queue, quickly skipping the rest. As soon as the aggregator has Y to match xY for the current ID, distance can be calculated and streamed back to client while it computes the next and the next until the xY queue is empty.

The count for x and the total count for N are constant across the answer set. They can be fetched once and do not require such streaming queues.

In faceted entity search, the user often wants to see associations of the query topic to many different kinds of things. A separate aggregator and its respective queues can address each of many categories in parallel. For example, one aggregator can work across persons associated to a topic while another works across companies. The same computation is performed with the same x and N counts, but the xY and Y count sets are queued and streamed from the other distributed memory locations.

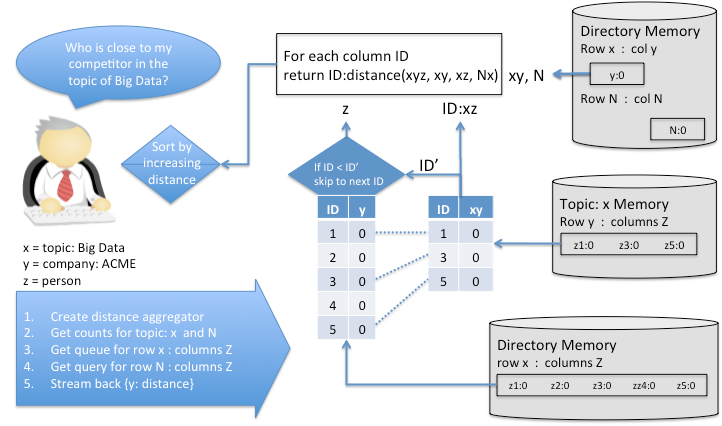
# Conditional Cognitive Distance Between 3 Things

The simple query like show me all associations to “topic: big data” shows how streaming aggregation in SMB computes distance over a large answer set. In faceted entity search, queries are iterative though. The user may see one interesting result and then include it in the query as additional context to query again. In the case of “topic: big data”, the user might ask for persons associated with this topic and then include one person in the query as context to find companies associated with “big data”. The returned associations of persons to the company are conditional given the topic “big data”. The inclusion of such context requires computation over triples. An inverted index as used by web search engines does not allow calculating conditional distances. One would have to go back to the original documents at query time in order to find the context word person in all sentences where big data and ACME have co-occurred. The condition would have to be found in the same sentence or at least the same paragraph, while the inverted index has only the whole web page as scope. Reprocessing at query time all the pertinent (co-occurrence) web pages would not scale.

In SMB these triple connections and counts are contained naturally in memories, such as the memory for the topic.

We need a distance aggregator with two queues attached similar to the simple distance calculation. The queue for calculating the association counts xy|z (Big Data ∩ person | ACME) needs to read from the ACME memory instead of the DM though, since we look for the co-occurrence of big data AND person given ACME. If the user wants to know the raw counts between three things, the conditional memories are sufficient. But to calculate similarity, the other required parts of the equation are found in the DM.

Parallel aggregators can compute distances across many attributes to answer how entire attributes interact with other attributes. In other words, the entire company-by-



**Fig. 4. Conditional streaming aggregation across memories.** The machinery for computing distance for the information of triples is virtually identical to that for the information of doubles. The distance aggregator takes the same input arguments and the queues attach to distributed rows. However, the attachments “shift” xy to xyz coincident counts. The x and y independent counts shift to xy and xz counts. In other words, the xy count shifts to yz conditional on x. This is equivalent to the yz count located in the topic x memory, other labeled as xyz as a triple association. The independent count of y becomes the count of y given x, which is located in the Directory Memory. Rather than the global observation count of N, we normalize for the conditional by using the N of x, which is the number of observations of x.

person contingency table within the topic memory can be searched for the strongest correlations between persons and companies, given the topic. When the user does not

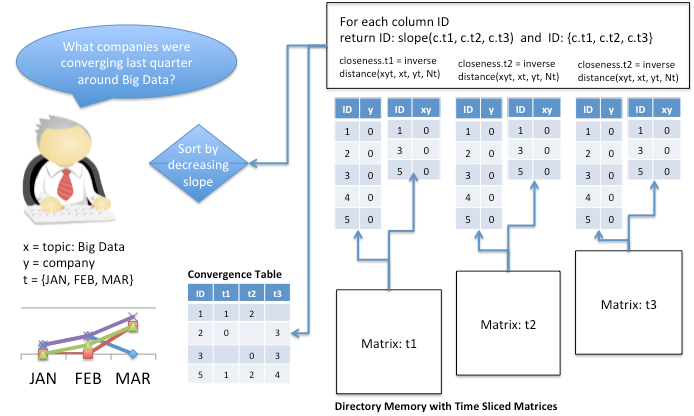
know which company to query or does not want to iteratively explore and discover such companies, a memory base can do more of this cognitive work. The memory base can quickly search through the dependencies between all companies and all associated persons, returning the most dependent company-persons as most interesting. Or imagine searching for company-company dependencies. Imagine seeing how all of your suppliers are dependent your competitors.

# Convergence as Similarity over Time

“Trends” can be implemented as the simple counts of a thing over time. A trend of a stock price is its daily closing price plotted day by day for example. SMB stores all frequencies over time and can also support such basic reporting. From unstructured sources such as Twitter, we can mark any thing like a company name or a topic like “Apple”. Mentions of the company may trend up or down over some time range. We might see that “Apple” is trending up.

Using CD also allows SMB to return the associated other things that are trending around a topic. Again, the query of “topic: big data” can return the trending frequencies for “big data”, but it is more interesting to return other associated trends – showing things the user didn’t even know to ask. If interested in a company, we want to know what is trending around that company. For example, we may see the trends of events important for a company, or a company’s products associated to other companies, people, or products. Or given a topic such as “big data”, what companies are trending, associated with “big data”? Rather than show a trend line to a given topic, associated trending provides information the user might not even know to ask.

For convergence SMB computes the Cognitive Distance between things or ensembles of things (like companies) over time using the time sliced topic memories. Convergence measures if things are growing closer together, or moving further apart. Convergence represents the trend of the relationship, to better anticipate where the association is going. The same approximation as before K(x) = - log (x) is used to calculate conditional information distance dImax(x,y|c) = max{K(x|y,c), K(y|x,c)} / max{K(x|c), K(y|c)} if we try to anticipate if three things will converge. For instance, if our task is to intercept drugs, we can plot dImax(x(ti),y(ti)|c(ti)) over time, and use for x boats, for y boat mechanic, and for c the assumed locations, where we suspect a drop to most likely happen.



**Fig. 5 Convergence streaming of distances across time.** Rather than only 2 vectors, one for xy and one for y, the convergence aggregator combines such vectors across many different time slices. Because the query requires only pairwise associations, access is limited to the Directory Memory. Although not shown here, convergence of triple associations, such as when asking for companies converging with each other around the topic, simply shifts the aggregator to the topic memory triples over company-by-company cells as well as the Directory counts for normalization. Although not shown, the aggregator logic steps through all vector queues, all in global sort order, computing and streaming each computation for each company. “Closeness” is the inversion of distance in this case. Slope, acceleration, or another other indicator can be computed over the time series of closeness. The indicator can be sorted to display the most strongly converging, and/or the closeness values over time can be returned for graphic display of the convergence trends.

Staying with the same use case of drug smuggling, we may ask how different means of transportation are trending over a certain period in time. In this case, when dealing with ensembles, the use of entropy measures to compute similarity is appropriate. We use interaction information I(X;Y;Z) with X being the means of transportation, Y being the ensemble of locations, and Z being persons to anticipate the involvement of a boat versus a plane, for example. For details on I(X;Y:Z) see the mathematics appendix.

SMB aggregates the appropriate counts depending on the context, categories, and time frames requested by the user. While distance can be calculated for only one cell if given a very specific question (between only one topic and two given companies, for example), more general discovery and query-less querying require the computation of distance over vast numbers of answers. SMB does more cognitive work for the user, evaluating many possible answers to return the most “interesting”, often what the user would have never known to ask. Convergence over time adds the sense of momentum for expecting the future rather than just discovering the past; it shows us the intersection of conditions, facts, and events. Prediction.

Classification By Compression

Cognitive Distance, also known as Normalized Information Distance, is useful for cognitive reasoning in faceted search and discovery, such as for correlation and convergent trending as just described. What about more complex patterns that exist across many diverse points in a memory? This is the classic definition of emergent properties such as when individual ants act as a colony. Each ant has its own small role to play, but acting together, the entirely new “entity” of a society is formed, acting more intelligently together than each ant alone.

This is the metaphor for classification in that a situation or object is described not just by a name, but also by an entire vector of its attributes or features, which represents an even larger number of associations. We are given a lot of different things in the vector, all possibly interacting with each other, and we wish to know how it fits into a larger entity, called a class. Does this vector belong to the class for animal, mineral, or vegetable? Is this situation good, so that I might buy, eat, or otherwise consume it? Or it is bad, so that I might sell or run away from it? Classification can tell us what a thing most likely is, and classification is also fundamental to prediction, telling us what might likely happen.

Classification is fundamental to machine learning. While still controversial, with most practitioners trained in data modeling methods of the last century, a small but growing community believes that algorithmic modeling, as Kolmogorov, Solomonoff and Chaitin have formalized, holds the universal key to intelligence as proposed in Universal Artificial Intelligence (Hutter, 2005). Data modeling assumes a universe of random variables. The data modeler has to make upfront assumptions of independence and of homogeneity across observations, including assumptions of Gaussian or other distributions. These assumptions are often unfounded and ad hoc, especially in the Big Data arena. Therefore such methods are hardly universal. In contrast, Kolmogorov complexity K(x) shows a radically different approach to machine learning generalizing Shannon Information Theory.

Think of Alice throwing the dice in Las Vegas and loosing 1000 times against Bob because the same number always came up. She goes to court and argues that the dice must have been biased. However, the judge says, “Alice, Bob won rightfully since this sequence is as probable as any other sequence.” Our intuition tells us, something is “unfair” here based on traditional statistics. To better decide this case, we need a measure to better differentiate a random string (throwing the dice) from a regular string that includes such a repeating pattern. Kolmogorov has defined such a measure, based on complexity. It turns out that regular strings, like the repeated sequence of one number, needs only a short program to describe it. If one die always came up 6, we can describe the entire sequence of 1000 throws as “6” \*1000. One the other hand, if the die was fair and random, we would need to record each throw as “516425216352…” for each of the 1000 throws. While any sequence is statistically as likely as any other sequence, the measure of complexity tells us that there is a pattern in the repeated sequence. It is not random sequence. If Alice had noticed the repetition early in the sequence, she could have started to predict the next throws – and won.

Kolmogorov’s universal probability cuts through the dilemma of choosing the right laws of probability by a priori assumptions, which plague classical statistics. His answer is to measure the likelihood of the simple sequences; there are not many of them since most possible strings do not have such a short minimal description length. Compressibility gives us a measure of regularity, which we can use for comparison and prediction. Using Cognitive Distance as similarity measure, we can compare a test vector with regards to stored memories and predict such diverse things as levels of threat, when parts will break, what product a consumer will buy, and much more.

We desire to find patterns that are emergent from the data itself, without a priori assumptions about data distributions and without parameter tuning to fit the data to an ad hoc function. We want the memory to recollect patterns quickly and automatically, without the cost and latency of traditional modeling. This is critically important for data problems with these characteristics:

* **Heterogeneity**. Different methods address either discrete versus continuous variables, but both are informative together in making a prediction. No matter the data types, a universal approach to learning must address and combine them. We transform all variables into a bit vector space exploiting one representation and its properties of distance.
* **Dimensionality**. Traditional statistics requires that the number of observed events is large enough to solve for a smaller number of variables. In contrast, the world will often present a high number of variables within a low number of cases, for which decisions must still be made. The new norm is the norm of one!
* **Non-linearity**. Assumptions of independence between two variables may be wrong when they interact with a third one. Whether predicting the desirability of a shirt color depending on the occasion or predicting the failure of a part depending on its location, non-linear interactions must be accounted.
* **Individuality**. Statistical generalities are often made under the assumption of homogeneity, that all cases are drawn from one homogenous population. In contrast, complex objects – like people – are not homogenous. Vast numbers of models, one for each thing, are required to address each individual’s rich complexity and difference from others. It is impossible to manually construct and tune each and every model at such scale. Individualized personal assistant learning must be autonomous, as an agent.
* **Model Aging**. The tradition of data science is to train, test, and deploy, hoping that the world doesn’t change once deployed. In other words, a model is tuned, tested, and tuned again, then fixed – not learning anymore – once in the field. If the underlying world changes, accuracy will likely degrade and the model must be re-tuned. For non-stationary environments, machine learning must be continuous. Moreover, learning should be instant when change is rapid.

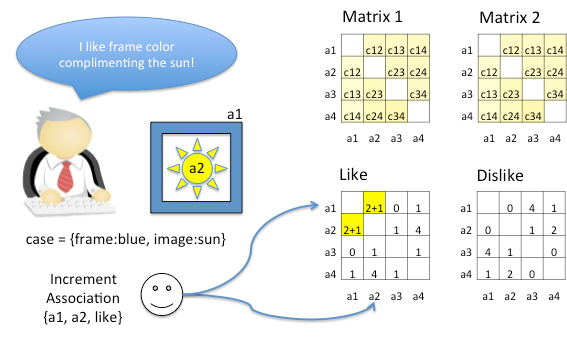
Our own brains easily deal with such real world characteristics. We learn from both structured and unstructured data, discrete and continuous variables, often in high dimension, interacting with each other, and changing over time.

Many approaches to classification involve fitting data to a model such as a logistic regression, Bayesian Networks, Support Vector Machines, or several forms of neural networks. This fitting to a specific function usually implies finding some decision boundary between one class and others by the “learning” of appropriate coefficients and “weights”. It should be no surprise that such machine learning is slow and not universal.

Memory-based reasoning is instance-based learning. Rather than slow adjustment of weights to fit the data to a model, a memory is simply a memory. It instantly remembers what it observes, assimilating new cases with its memory of prior cases. SMB functions as a compressor using cognitive distance for reasoning by similarity.

# Storing Experiences in Memory

While one associative memory/matrix represents a weighted graph, many memories/matrices represent a semantic triple store. Each memory is conditional to one “thing”. To use this machinery for classification, we use a separate matrix for each class. For example, a memory for a consumer named “John Smith” might include matrices to represent his online behaviors, such as one matrix for observing what he “likes” and other for observing what he “dislikes”. As another example, imagine a memory for “cancer” with separate matrices to learn and classify “malignant” or “benign”.



**Fig. 6. Classification learning within associative matrices**. Each class is a matrix of pair-wise coincidences between attributes (a), conditional to the class. Similar to the way every “entity” is the condition over its own memory-matrix for semantic triples for entity analytics, each class represents triples of z = f(XY) for predictive analytics. Although larger vectors are typical, a case is represented by every pair-wise combination of attributes, labeled by its class by supervised learning. Given the case vector and its label, the associative counts that represent the vector are incremented in the appropriate cells of the appropriate matrix.

Of course, a memory can contain more than two matrix classes. The approach of fitting weights typically assumes that there is a line, plane, or hyper-plane that separates two or more classes, and weight adjustments seek to define this separation. In contrast, a memory can also contain only one matrix to support “positive only instance” learning, which most learning algorithms cannot. In some situations, the learning engine does not have the opportunity to see the compliment of a class, only the positive instances of one class. Using CD as a measure, we can see how far or close one case is to the class. Even if we have only one matrix for what a person likes, we can still score different products and say that one is likely preferred than another.

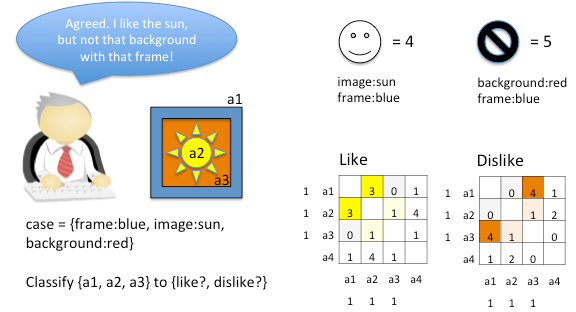
The learning of each matrix is “supervised” in the sense that every given case to learn also includes its class label. If a user marks an email as “spam”, then a matrix for “spam” will learn the email’s attributes and how they occur together under this condition. If other emails are marked as “urgent”, then another matrix will observe these cases. The result is one or more matrices that represent the connections and counts for each class.

Aside from the matrices and the representation of connections and counts in the context of the class, there are no mathematical functions and no parameters to fit these functions during the constant and instant learning of connections, counts, and context. Rather than fit a model as data arrives, a memory waits until the question is asked. Given all the cases assimilated up to that instant, the memory computes the distance of the case to each requested class.

# Classification By What “Lights Up”

The method of memory-based inference is as transparent as the method of learning in the first place. Given a binary test vector **V**, the intersection of vector elements in a case will “light up” particular cells in each class matrix **C**. In detail, what lights up are the same cells xiyj (=cij) of the class matrix **C** where the outer product of the test vector **V**\***V**T has a matrix element vivjT equal to 1, i.e. the “light up” matrix is, **M** = **V**\***V**TΛ**C**; where Λ is the element wise AND-function. As with correlations and convergence, more computation will be required to go beyond the raw counts. Even though, the number of cells and the counts of these cells in each matrix give a sense to which class memory a test vector belongs.

Given the simple case of two matrices like in Fig. 7, the relevant connections and the frequencies of these connections indicate where the case “belongs”. However, raw counts are too raw and can represent a biased skew in the observations. For example, imagine the skew in the counts if one matrix happens to see more cases than the others. For example, suppose that 95% of cases are “not threating” and only 5% are “threatening”. When given new cases, the dominance of connections and counts that are “not threatening” will likely predict this class most of the time. Even with no intelligence at all, simply predicting “non threatening” all of the time will be correct 95% of the time! To do better than this, we have to use the frequency (xy/N) calculated from the count rather than count itself.



**Fig. 7 Expressing class membership of a new case vector**. Given a new vector, its attributes are applied to one or more class matrices. Applying the attributes to both rows and columns, the relevant cells “light up” to represent the connections and counts of all similar experience. Although raw counts will be further normalized and used to compute distance, we can intuitively see the where and by how much each matrix expresses its relative experience. Opponent classes such as good vs. bad, buy vs. sell, or like vs. dislike can be compared in total as well as the transparent explanation of why the new case belongs to on or another class.

Renyi entropy is a very versatile entropy. We use quadratic Renyi’s cross entropy, which is related to Cauchy Schwarz divergence to add the lit up co-occurrences before taking the logarithm and comparing for which class the test vector has the biggest value. Using Renyi’s cross entropy has been surprisingly accurate for many data sets, including several standard Machine Learning data sets such as the breast cancer data set, reaching 97% accuracy in less that 50 trails from the 700 malignant or benign cases, and the mushroom data set, reaching 100% accuracy in finding the underlying associative rules for edible or poisonous. In both data sets, accuracy remained asymptotic as more cases were observed; there was no over-fitting observed.

Many operational datasets from customers, ranging from condition-based maintenance to spam filtering, show all accuracies better than 97%. Boeing for example has improved it’s predictive maintenance from less than 70% recall and with nearly 20% false alarms to 100% accuracy with less than 1% false alarms.

This success rate is also due to our customer’s inclusion of both structured and unstructured data in the prediction: the part’s sensors combined with the operator’s verbal reports. A partner’s spam filter called Electronic Learning Assistant (ELLA) was compared to many other product technologies and declared “World’s Best Spam Blocker” (PC User Magazine, 2003).

Shannon Mutual Information is a more intuitive similarity measure and has the advantage that is a good approximation to the Mutual Information based on Kolmogorov complexity. Using mutual information as an approximation to Cognitive Distance between a vector and the memory of each class member is warranted because the memories represent ensembles of prior cases. It has been shown (quoted in Kraskov et al., 2008) for many test data sets that for ensembles there is practically no difference between Shannon mutual information and Kolmogorov mutual information. This is not surprising because the average of the Kolmogorov complexity for ensembles of objects is roughly equal to the Shannon entropy (Leung Yan Cheong and Cover, 1978; Grunwlad and Vitanyi, 2003).

We have used this Cognitive Distance approximated by Shannon mutual information to automatically analyze echocardiograms working with Mt Sinai Hospital. Echocardiology has advanced to measure 10,000 variables, over the time of a heartbeat and over many locations of the heart. Strains, displacements, and velocities of the heart tissue itself as well as blood flow are now available as raw data. These variables change over time and are also highly interactive, resulting in a very nonlinear pattern of variables to distinguish one heart condition from another. Given two classes of heart disease, for example, with thousands of simultaneous variables per patient, this new classification method has been shown to be nearly 90% correct (Sengupta, 2013). Even as early results for continued improvement, this appears well beyond the accuracy of cardiologists. Given the high dimensionality and nonlinearity, comparative results from other more standard reductionistic methods like decision trees (Ctree) were far behind and not worth reporting. Saffron’s accuracy is achieved without data reduction or model parameterization. Simply by observing a few cases of each class (15-20 for each class of disease condition), memory-based classification computes the mutual information of all the associations that “light up”, in real time whenever a new case is given.

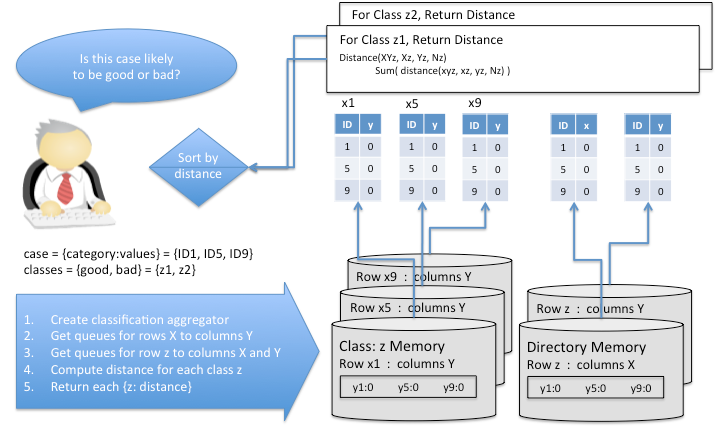
# Calculating Similarity for Classification

Calculating the Kolmogorov Mutual Information for classification involves aggregation of xy, x, y, and N for each cell of the “what lights up” submatrices in a similar way as described for correlation and convergence.

The machinery of finding “what lights up” is different though from that for point-wise similarity. The access pattern for the later is given one or a few query terms, and the machinery must scan across a vast answer set, computing the distance for each. In the case of convergence, we have extended this access pattern to also scan over time.

Classification however is given a potentially very large vector, comparing distances across entire matrices, based on a massive number of given pairs, conditional to each matrix. As for correlation, the benefits of matrix organization are used for classification as well:

* **Aggregation of raw counts across matrices**. The elemental counts must be accessed across potentially more than one matrix and definitely across more than one row of distributed matrix partitions. Classification must recollect many counts to compute an overall distance over many cells.
* **Row dominant access of the active submatrices**. What “lights up” represents a subset of the entire classification matrix. As one dimension of the active submatrix, only the relevant rows need to be accessed and aggregated. A matrix may be extremely large, but only the rows that “light up” to a vector are required.
* **Column ordering of category:attributes**. Classification requires a somewhat different access pattern than correlation and convergence. Rather than all the values of one category, classification requires access to one given value per category. However, categories are also in global sort-order in high-bits of each ID, also ordering all the column category:values within each row for efficient aggregation of each xy, x, and y.
* **Parallel count servers and sub-computations**. Classification leverages the overall parallelism of access across required matrices and rows. Different servers are responsible for the counts of different matrices and rows. Each independently scan and return the required columns that define a submatrix for the classification aggregator.



**Fig 8. Aggregation of submatrices from a distributed representation**. Each matrix computes its distance to the case vector. The case vector defines the rows and columns of the relevant submatrix, which will tend to be distributed across a machine cluster. Distribution is row dominant; therefore, each attribute of the case defines a row within the matrix of each class. Each row is streamed to the classification aggregator, but rather than stream an entire row, each server returns only the column IDs and counts for attributes within the case vector. Collectively, these streaming queues provide the relevant submatrix **M** = **V**\***V**TΛ**C** to the classification aggregator; **V** being the test vector and **C** being the class matrix. While a simple distance can be estimated from only the xy association counts a more refined method also accounts for the independent terms of x and y for mutual entropy for example. The aggregator uses the counts to compute distance by summing across all cells of the submatrix **M**. Similarity distances from the test vector to each class are returned and can be rank ordered for presentation or further processed for decision support.

All the memory access points can be parallelized and brought together for aggregate computation of the distance between the given case vector and each class matrix. The computation of each class is also independent and can be computed in parallel across classes. The aggregator returns the distance of each class. As a matter of philosophy, SMB does not make an ultimate decision. For example, it does not compare classes and answer only the closest class. Even for a two-class problem, it does not assume a particular decision threshold to decide good or bad, left or right. Instead, all the distances are given to the application in rank order of distance along with its distance. The full distribution of distances is most informative for the application and end user to then make an ultimate decision.

Conclusion

This paper has focused on reasoning by similarity. Memory-based reasoning is a form of “just in time” learning with its minimum commitment principle. Rather than fitting a data to an assumptive model, counts are continuously incremented as new data arrives. As a universal method, there is no knob tuning or re-tuning as new data arrives and the models change. An algorithmic model rather than a data model is computed at query time.

SMB is a massive correlation engine. It is a compressor in the sense that it builds an associative index of the raw data by generating a full semantic graph with co-occurrence counts at its edges. The very quality of SMB being a super efficient compressor is exploited to approximate the Universal Cognitive Distance (CD) based on Kolmogorov complexity. Kolmogorov complexity is the most natural concept to discern signal from noise and to calculate what is cognitively close from what is cognitively far.

We used CD for correlations and queries using semantic context. We also described how to use CD for novelty detection over time, which we call convergence - the sense of momentum for expecting the future rather than just discovering the past.

We have shown especially in the section on prediction via classification that we compare large vectors to large matrices across a large number of matrices in real time. This section provided a clear example of how SMB’s unique approach to partitioning, distribution, compression, and streaming allows the delay of these computations when learning while still answering “just in time” fast.

In general Saffron’s cognitive representation allows for instantaneous use of Universal Cognitive Distance enabling reasoning by similarity.

The mentioned applications are hardly the limit of what is possible. As an entire Cognitive System, the brain is a network of networks of associative memory networks. Many different subsystems are responsible for semantic memories, episodic memories, procedural memories, motor memories, and more. Each of these subsystems is itself a network of individual memories connected within to each other and to the other networks. All memories are represented by the connections, counts, and context of a hyper-matrix implementation. Using a Universal Distance, all memories reason by similarity as the “fuel and fire of thinking” (Hofstadler and Sander, 2013).

Appendix:

Mathematical Foundations

# Overview

The purpose of this section is to explain the well-founded mathematics of Cognitive Distance, and derive it’s various approximations for reasoning by similarity. We will start with intuitive and well know distance measures like Hamming distance and Jaccard similarity (distance), which are intimately related to associative memories. Then we will elaborate on the equivalence between Jaccard similarity and the Shannon entropy measure for similarity, the normalized mutual information.

The Jaccard Distance based on the Shannon mutual information can be generalized to Information Distance based on Kolmogorov Complexity. This normalized form of Information Distance is called the Cognitive Distance because it is a universal distance measure that can be derived from physics as well as from information theory.

It turns out that Shannon mutual information can be used when approximating the Cognitive Distance between ensembles. However, in order to approximate the distance between single objects, we need an approximation that goes beyond Shannon entropy. Considering Intelligence as a problem of compression comes closer to the true nature of Kolmogorov complexity.

# Distance in Bit Space

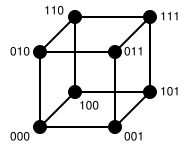
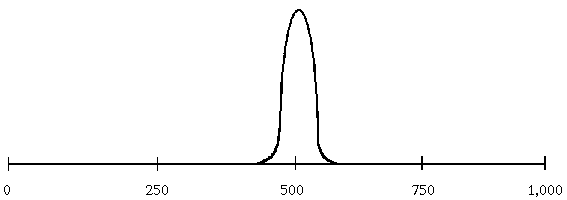
Reasoning by similarity is fundamental to associative memories. This has been true throughout the history of associative memories, which include Hopfield Networks (Hopfield, 1982) during the rise of neurocomputing in the early 80s. Kanerva’s Sparse Distributed Memory (Kanerva, 1988) represents another seminal work that followed. Both approaches address bit-distances, or Hamming distances. Given two bit vectors, the Hamming distance between them is the number of bits that are different. The converse, similarity, is the number of bits that are the same.

AMs are mathematically well defined. They are as fundamental as the von Neumann architecture. They have a Lyapunov function, thus they have very well defined fix points. The weights of AMs are deterministic thus allowing for direct interpretation of the results and they are parameter free. The weights of the fix points are what we look for in physics and big data alike - "the order out of chaos". The weights in an AM have physical meaning therefore.

Hopfield was a physicist, who invented an alternative to von Neumann computing. He realized that the Ising Spin Model of magnetic spins, UP or DOWN and how they interact to transition from chaos (non-magnetic) to order (magnetic), is isomorphic to associative memories. Hopfield showed how state vectors could be stored in a network of neurons connected to each other. When Hopfield defined the AM he showed that it is isomorphic to the Ising Model (model for spins). Hopfield used the AM to calculate the weights and thus explain the phase transition of Ferromagnetism; the change from not aligned spins to aligned spins – order out of chaos.

The very surprising and deep fact is that the world (e.g. ferromagnetism) can be explained by nothing else but local connections and counts. All fix points of the system can be derived and explained by nothing else but local counts and connections.

A network of neurons can be described by connections and counts forming a content-addressable associative memory. When a new bit vector is applied to such a network, it asynchronously flips the vector bits to recall the closest memory location (mathematically speaking, a fixed point) formed by previously loaded vectors.



Bit distance between vectors of 1000 bits

Probability of distance

From mean bit vector

**Hamming space and distance probabilities**. Given three bits, a cube describes the vertex location of each vector in Hamming space. The Hamming distance from one vertex to another is seen as the number of bit-flip, such as the distance from 000 to 111 as 3. As the Hamming space becomes larger, such as to 1000 bits, the probability of distances falls exponentially. Assuming a mean vector of 500 bits ON and 500 bits off, the likelihood of a vector with more than 80 bits different is only 0.0000001%. Vectors that are different are very different by this distance measure.

Kanerva further described the nature of Hamming space, particularly showing how distance probabilities are much farther away than in Euclidean space. For example, in a random vector space of 1000 bits, the mean distance between any two vectors is 500 bits with a standard deviation (one “sigma”) of only 16 bits. In other words, at a distance of 5 standard deviations to the mean, only 80 bits are different but only one in a million vectors are likely to be outside this distance. Things that are far away are very far away. The distance density in Hamming space grows exponentially with the dimension of the space, thus separating similar objects much more from the noise than the Euclidean distance.

Hamming space also addresses continuous variables through discretization. Many methods of discretization are possible, including thermometer codes, percentile binning, and entropy-based binning over the range of one or more dependent continuous variables. In this way, bit vectors can represent both structured and unstructured data sources containing both categorical and continuous variables. In the same way that schema-free graphs unify the connection knowledge from different data sources, the universal representation of a bit vector unifies different data types into the preferred properties of Hamming space for measuring distance between vectors.

It appears that neurons may also prefer these bit-oriented properties through the “line coding” of neural synapses. Each synapse is discrete, and the receiving neuron does not know the source or semantics of the input neuron. As with each bit in a bit vector, each synaptic input line represents its part of the input pattern, and these input lines are all the receiving neuron can use for its computations. SMB works the same way. Matrices represent simplified neurons by storing connections and counts between category:values, transformed into line code IDs. The matrix does not know the source or semantics of the IDs that feed its perspective.

Hamming distance assumes that every bit in a vector space is either ON or OFF, that the vector is always complete. In reality, many observations of the world will present themselves as partial vectors. For example, two documents will likely be of different lengths and partially different terms. A more generalized associative memory must allow unknown bits as neither ON nor OFF. For comparing vectors of different size and different attributes across diverse observations, Jaccard distance is similar to Hamming distance but without a penalty for non-matching OFF bits. Looking at only what is ON in the two vectors, Jaccard similarity is defined as



In other words, similarity is a measure of the intersection of ON bits in vectors A and B (both bits ON) compared to their union (either bit ON). SMB works similarly in evaluating only the ON bits. An entire memory space can easily reach billions if not trillions of attributes, whereas specific query vectors concern only a subset of the universe when reasoning by connection and similarity.

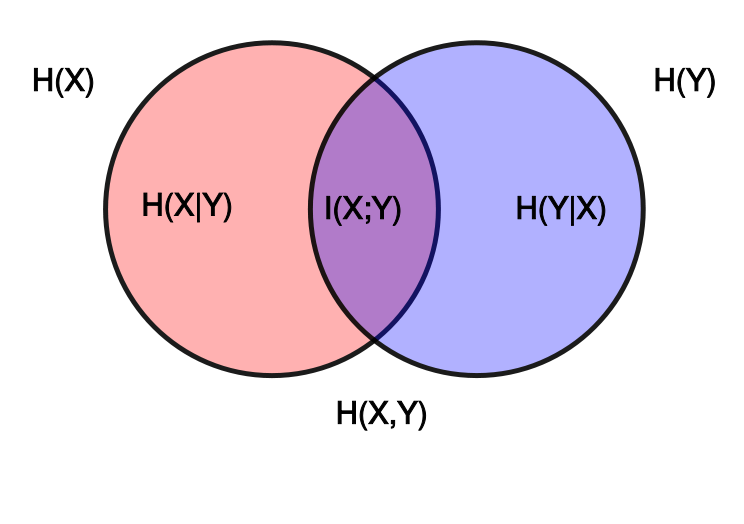
Depending on the application and uncertainty in data, SMB will further limit the measure of similarity to only the intersection of what is known. For example, not knowing whether someone is married cannot be assumed as unmarried. In national security, a well known aphorism warns, “Absence of evidence is not evidence of absence.” As famously distinguished according to “known knowns … known unknowns … and unkown unknowns”, Saffron has proven better accuracy with this approach for alias detection, looking for similar persons to a target person. Saffron’s vector-space matching was found to be 40X more accurate than rule-based pattern matching. Saffron’s approach includes the feature and relationship connections of data; given one thing, SMB can lookup features and relationships as the “signature” vector to compute Jaccard-like similarity. As well, entropy is computed to weigh the “interestingness” or informativeness of these connections. In a “bad guy” database, being male is completely uninformative if all the persons are male. It adds no information to help the matching. Similarity distance is based on semantic connections as well as on statistical frequencies.

# Distance Based on Entropy

In computer science, entropy is a measure of information, in terms of bits. Entropy measures the degree of information uncertainty in the data.

Why entropy? We desire a non-parametric approach to measuring dependency, which is independent of the actual random variables but rather depends on their distribution. A major weakness of many traditional methods like regression analysis, Principle Component Analysis (PCA), Fisher discriminant analysis, etc. is that they are not invariant under a transformation of the variables. For example a linear scaling of the input variables (that may be caused by a change of units for the measurements) is sufficient to modify PCA results. Feature selection methods that are sufficient for simple distributions of patterns belonging to different classes can fail in classification tasks with more complex decision boundaries. In addition, methods based on linear dependence (like correlation) cannot take care of arbitrary relations between the pattern coordinates and the different classes.

In contrast, entropy, and more precisely the mutual information, can measure arbitrary relations between variables and does not depend on transformations acting on the different variables. Entropy does not depend on variables but only on their distribution. Thus, entropy provides a more general measure of dependence.



**Venn Diagram of Entropies.** A Venn diagram is the common way to explain the different elements of entropy, denoted by H. H(X) and H(Y) represent the information in each separate circle for variable X and Y, respectively, called marginal entropy. H(X,Y) represents their combine information, or joint entropy, the entire space of the diagram. I(X:Y) is the mutual information, shared by both X and Y. H(X|Y) and H(Y|X) represent conditional entropies, such as the information “left over” in X when given Y and visa versa. It is clear that H(X) = H(X|Y) + I(X;Y). In other words, the total entropy of X equals the conditional entropy of X given Y and the mutual information between X and Y. It is also clear that H(X|Y) = H(X) – I(X;Y), the “divergence” of X given Y as discussed below. Mutual information is central to the diagram and central to the mathematical links to distance, dependency, divergence, and complexity.

More formally, dependency is a general concept, which can be formalized using mutual information:



For simplicity, point-wise mutual information (pmi) measures the information content of the cell count between x and y at one point, one cell, in an associative matrix:



In other words, pmi is the logarithm of the lift. Lift is the probability of the coincidence count between two attributes, x and y, divided by the two probabilities of each attribute separately. For example, not knowing anything about English, we would like to learn if the concatenation of “vice” and “president” has any meaning. We look for how often do we see “vice president” in a document vs. finding “vice” and “president” separately. To the degree the combined frequency lifts away from the independent frequencies, there is meaning in “vice president” together.

“Mutual information” (the sum of pmi over all values in a category for example) will be used interchangeable for both I(X;Y) and pmi, unless the distinction needs to be made clear when applied to different uses. In either case, the computation of mutual information at each point requires 4 counts from a memory base. The probability of p(x,y) requires the count x,y divided by N, the number of total data observations. Similarly, p(x) and p(y) require the independent counts of x and y also divided by N.

Finally, mutual information between two variables can be extended to three and more variables. More information often “lives” in these higher interactions. For example, binary events a and b are independent if and only if P(x, by) = P(x)P(y) from which I(x,y) = 0 follows, which implies that x is irrelevant to y. However, independence is not a stable relation: x may become dependent on y if we observe another event z. For example, define z to happen depending on when x and y take place. The exclusive-OR function is an example. We observe two independent random events (e.g. sources generating 0s and 1s). Whenever only one of the two sources generates a 1, a bill rings that we call event z. The bell (Z) makes source X and Y dependent. Therefore, even if X and Y are independent and random, they become dependent in the context of Z. This contextual dependency will be further described in the applications below, whether to compute conditional distances or classifications, which assume the class as a condition.

One may think such 3-way interactions are as esoteric as XOR. This is by no means the case. Take for example the task of detecting criminal behavior by finding regularities in huge amounts of data. The attributes of a person’s employment and criminal behavior are not particularly dependent attributes (most unemployed people are not criminals, and most criminals are employed), but adding the knowledge of whether the person has a new sports car paid for by cash suddenly makes these two attributes dependent: it is a lot more likely that an unemployed person would buy a new sports car with cash if he were involved in criminal behavior. The opposite is also true: it is less likely that an unemployed person will buy a new sports car with cash if he is not involved in criminal behavior. The dependency is not absolute; the person may be unemployed and wealthy. The relationship is only more likely. This concept is captured by the measure of the intersection of all three attributes, or interaction information (McGill, 1954).

I(X;Y;Z) = H(X,Y,Z) - H(X|Y,Z) - H(Y|X,Y) - H(Z|X,Y)

Interaction information among three attributes can be understood as the amount of information that is common to all the attributes, but not present in any subset. Like mutual information, interaction information is symmetric, meaning that

I(X;Y; Z) = I(X;Z;Y) = I(Z;Y;X).

Three-way mutual information or interaction information has been shown to be very relevant in many difficult classification and pattern recognition examples. In biochemistry for example, additional information has been found in higher dimensional hyper-matrices (Barigye et al., 2013). SMB stores such “triple” interactions together with its triple semantics. Each matrix has a conditional label and all its internal statistics are conditional to this label. In this way, pair-wise mutual information is extended to three-way “conditional mutual information” (and interaction information). This allows SMB to discern the signal from the noise much better than the typical systems that implement only 2-way correlations (dependencies).

Returning to the Venn diagram of entropies, the two conditional entropies, dxy= H(X|Y) + H(Y|X) can be seen as the compliment of mutual information, also called variation information. Variation information represents how far things are. It is a distance measure since it satisfies dxx=0, is symmetric (dxy-dyx), and satisfies the triangle inequality.

From the Venn diagrams we see that dxy = H(X,Y) – I(X;Y). In other words, the distance dxy can be seen as the difference between a complexity term, the joint entropy H(X,Y), and a similarity term, the mutual information I(X,Y). Imagine two Venn diagrams, one much larger than the other in total joint entropy H(X,Y) but where the amount in mutual information is the same. Mutual information is the same, but in the bigger diagram the distance between X and Y (i.e. H(X|Y)+H(Y|X) ) is greater. Casting H(X,Y) as a measure of complexity for the joint distribution P(X,Y), distance can be seen as this tension between complexity and similarity. If two attributes X and Y share the same mutual information with the class attribute C (say C is credit risk: C = {high, low}) we would choose the attribute, which has the smaller complexity for predicting high or low C; quite similar to Occam’s razor, if we can explain something using different attributes we choose the simplest one, i.e. less complex one.

The concept of entropy can be generalized to the so called Renyi entropy. Renyi wanted to find the most general class of information measure that preserved the additivity of statistically independent systems and were also compatible with Kolmogorov’s probability axioms.



Here, X is a discrete random variable with possible outcomes 1,2,...,n and corresponding probabilities pi = P(X=i) for i=1…,n; α is a real or complex number and it is = or > 0 and different from 1. It seems that the Renyi entropy is singular for α=1, but one can show that it is well defined for around α=1. Actually α=1 is the well known Shannon entropy.

Renyi entropy has been applied in economics, ecology, demography and information science to quantify diversity. In this context the value of α is often referred to as the order of the diversity. It defines the sensitivity of the diversity value to rare vs. abundant species by modifying how the mean of the species’ proportional abundances (pi ) is calculated. In this sense, n is the richness, i.e. the total number of types in the dataset.

Some values of α give the familiar kinds of means as special cases. In particular, α = 0 corresponds to the harmonic mean (Hartley entropy), α = 1 to the geometric mean (Shannon entropy) and α = 2 to the arithmetic mean (often called Renyi entropy). As α approaches infinity, the generalized mean with denominator 1 – α approaches the maximum pi value, which is the proportional abundance of the most abundant species in the dataset. In practice, increasing the value of α increases the effective weight given to the most abundant species.

When α = 1 (Shannon entropy) the geometric mean of the information (log pi) is used, and each species is exactly weighted by its proportional abundance pi. When α > 1, the weight given to abundant species is exaggerated, and when α < 1, the weight given to rare species is stressed. At α = 0, the species weights cancel out the species proportional abundances, such that mean pi equals 1/n even when all species are not equally abundant. This so called Hartley entropy is simply the logarithm of n. This is similar to a Shannon transmission problem where all probabilities for the random variables are equal. At α = 0, the effective number of species exp(Hα) equals the actual number of species (n).

Besides using Renyi entropy for search and discovery as a measure for entity diversity (e.g. people, products, companies, customers), the Hartley entropy can be used as the most basic classification algorithm. By simply adding up the associative counts in one memory versus another, a partner’s spam filter by the name of Electronic Learning Assistant (ELLA) was compared to many other products and technologies and declared “World’s Best Spam Blocker” by PC User Magazine (2003). Hartley entropy works when we expect the random variables to be equally distributed. For classification of unknown distributions of random variables, we use the more universal Cognitive Distance approximated by Shannon mutual information.

# Distance Based On Complexity

We are interested in a universal measure of information beyond Shannon’s entropy that does not rely on probabilistic assumptions and the concept of transmitting ensembles of bit strings. We aim for a measure of information content of an individual ﬁnite object, and in the information conveyed about an individual ﬁnite object by another individual ﬁnite object. We want the information content of an object x to be an attribute of x alone, and not to depend on, for instance, the averages chosen to describe this information content. Surprisingly, this turns out to be possible, at least to a large extent. The resulting theory of information to reach this goal is based on Kolmogorov complexity K(x). This new idea takes into account the phenomenon that ‘regular’ strings are compressible.

By way of example, Shannon’s classical information theory assigns a quantity of information to an ensemble of possible messages; say the ensemble consisting of all binary strings of length 99999999. Assuming all messages in the ensemble being equally probable, we require 99999999 bits on the average to encode a string in such an ensemble. However, it does not say anything about the number of bits needed to convey any individual message in the ensemble. For example, the string consisting of 99999999 1’s can be encoded in about 27 bits by expressing 99999999 in binary and adding the repeated pattern “1”, or even shorter as 32 × 11111111, and the 8 1’s can be encoded as 23. A requirement for this to work is, that we have agreed on an algorithm that decodes the encoded string.

Using the length of an algorithm to encode a string as a proxy for Kolmogorov complexity leads to a deeper understanding for randomness too. It is quite clear that a random string can be encoded only by enumerating the whole string (K(X)=|x|, where |x| is the length of the string). In contrast, a string showing a pattern as a regularity can be encoded with a program much smaller than the length of the string (K(x) << |x|). Think of the number PI = 3.1415….., an infinite sequence of seemingly random decimal digits. Yet the complexity and thus randomness of PI is small because it contains just a few bits of information, i.e. the length of a minimal program needed to produce all its decimal digits.

An intuitive view to understand K(x) is to think of it as a compressor. A book containing random strings cannot be compressed, while a book by Shakespeare can be compressed considerably. Two infinite random strings will have infinitely many substrings in common while the DNA of a human will have more substrings in common with a chimpanzee than a cat. We will use exactly this notion of similarity for reasoning in SMB.

The Kolmogorov similarity looks quite similar to entropy similarity.

Ik(x:y) =K(x) - K(x|y) ~ K(x) - K(y) - K(xy)

In the Venn diagram for entropy, this was expressed as

I(X,Y)=H(X) + H(Y) - H(X,Y).

Similar to Shannon Entropy, K(x)-K(x|y) is the amount of information y knows about x. K(xy) (~ K(x,y)) is the minimal length needed to encode the concatenated string xy. K(xy) has to be larger than K(x) but cannot be larger than K(x)+K(y), very similar to the Shannon entropy. If x and y have the same complexity, then

K(x) = K(y) = K(xy).

For example, if two files are absolutely different, then there is no benefit of compressing them together; a combined compression will be no shorter than the two separate compressions. However, to the degree that the documents are similar, the compression size of both documents together will become shorter than their compression separately. Similarly, to the degree that randomness does occur within an object, the compression length will be increased.

The Kolmogorov (or information) distance looks similar to the entropy distance Dxy = H(X,Y) - I(X,Y), as the difference between a complexity and a similarity term

dKxy = max{K(X), K(y)} – Ik(x:y)

A full explanation of the use of Kolmogorov complexity for a Universal Information Distance can be found elsewhere, such as in Bennett et al (1998) who describe its use as a “Universal Cognitive Distance”. See also Li and Vitanyi (2008).

The Shannon mutual information and the algorithmic (Kolmogorov) mutual information are closely related. One can show that the distance dIxy based on mutual information is a good approximation to dKxy for certain cases. However, both distances dKxy and dIxy are not yet appropriate for our purposes. A mutual information of a thousand bits should be considered large if X and Y themselves are only a thousand bits long, but it should be considered very small if X and Y would each be very large, say 1 M bits. It’s well known how to normalize H(X,Y)-I(X,Y) in the case of entropy in order to arrive at a distance unbiased by the size of the random variables compared.

DExy = {H(X,Y) - I(X,Y)} / H(X,Y) = 1- {I(X,Y) / H(X,Y)}.

DExy is the Jaccard distance = 1 – Jaccard Similarity.

For the normalized information distance one gets an equivalent (Li et al. 2002).

DIxy= 1 - Ik(x:y) / max{K(x), K(y)} = max {K(x|y), K(y|x)} / max {K(x), K(y)}

Kolmogorov complexity and the universal information distance cannot be computed as a side effect of their universality due to the Turing halting problem. However, this is not a practical impediment. There are many ways to approximate Kolmogorov complexity, for example using compressors like gzip, PPMZ, or others. Another way to approximate K(x) is using Shannon-Fano code, one of the earliest compressors, if we are interested in the similarity between single objects. Assume we want to calculate the distance between two objects like “cowboy” and “saddle” occurring in a very large collection of documents, indexed by a web search engine such as Google. We observe that the Shannon-Fano code for any object is ~ log Px. Px = x/N, x being the number of pages we get when searching for “cowboy” and N being the number of total pages indexed. Also including the page counts for “saddle” as y and “cowboy saddle” as xy, substituting this into the above formula for DIxy we get

DIxy= max {log(x) , log(y)} - log(x,y) ) / ( logN - min{log(x) , log(y)}.

Doing the same for “cowboy”, “movie”, and “cowboy movie”, the raw coincidence count between “cowboy” and “movie” is much higher than that for “cowboy” and “saddle”, but saddle is much closer – has a smaller distance – than “movie” in cognitive space. Using Google page counts as in these examples, this distance has been called the Google Similarity Distance (Cilibrasi et al., 2007).

On the other hand, what do we do if we are looking for the more popular category as our answer? Imagine if we asked, "Where did Steve Jobs likely live?” We would query “Steve Jobs” and look for co-occurrences to the category city. Using the max distance would show Palo Alto as the first result before San Francisco. This filtering out of larger, more popular entities is what we wanted in the “cowboy saddle” case. San Francisco as the more popular category has more entities in its neighborhood; therefore, not all of them can be close. For the less occurring city of Palo Alto (with the bigger Kolmogorov complexity), Steve Jobs is relatively speaking more important than San Francisco. In other words, the category San Francisco has a lot of information that is irrelevant to the fact of Steve Jobs lived in the Silicon Valley, and symmetrically, Steve Job contains much information, irrelevant to San Francisco. So, we have to account for such irrelevant information.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Independent counts | Coincidence to cowboy | MAX Distance | MIN Distance |
| cowboy | 95,300,000 | 95,300,000 | 0.0000 | 0.0000 |
| cow | 137,000,000 | 98,800,000 | 0.0470 | -0.0055 |
| cowgirl | 53,400,000 | 21,220,000 | 0.1993 | 0.1327 |
| horse | 431,000,000 | 43,000,000 | 0.3314 | 0.1461 |
| saddle | 64,600,000 | 7,170,000 | 0.3522 | 0.3160 |
| movie | 1,890,000,000 | 73,000,000 | 0.4678 | 0.0672 |
| N | 100,000,000,000 |  |  |  |

**Max and min information distance to “cowboy”.** Given the independent and coincidence document counts from Google searches, the max distances are computed as shown above, with “cow” as closest to “cowboy” and “saddle” closer than “movie”, a very popular concept. Computing the min distance, however, the popularity of “movie” is allowed. Min information distance elevates the more popular result of “movie” (lower Kolmogorov complexity), while the rest of the distance sequence stays unchanged.

If we want to include the more popular information, we need to compute the minimum distance of {K(x|y),K(y|x)} instead of the maximum. This is not a distance in the strict sense. Triangle inequality and density constraint are violated, but this is intended. For partial matching, the triangle inequality does not hold. For example, Marilyn Monroe is close to JFK, and JFK is close to the concept of president, but Marilyn Monroe is not closely related to the concept of president. Relaxing the neighborhood requirement (density constraint), we must allow some concepts to have denser neighborhoods, because “…good guys have more neighbors…” (Li at al. 2008). Allowing those very popular objects to have denser neighborhoods makes it possible to select them more often. The use of min information distance should be restricted to popular (frequent) categories. Normalizing this distance by dividing it by min{K(x),K(y)} analogous to the max distance case, we derive

dIxymin = min {K(x|y), K(y|x)} / min{K(x), K(y)}

Doing the equivalent substitution in the previous equation as for the cowboy and the saddle example, where we have used the Shannon-Fano code to write K(x) ~ log (f(x)/N), we derive

dIx,ymin= log(f(x,y) / N)-max{log(f(x)/N, log(f(y)/N)} / min{log(f(x)/N),log(f(y)/N)} =

(min{log f(x), log f(y)} - log f(x,y)) / (logN - max{log f(x),log f(y)}).

As with entropy, these independent and pair-wise counts can be extended to the conditional information of triple-wise hyper-matrices in SMB. The example around “cowboy” represents a single term query. However, when additional context is included, SMB can also compute conditional distances.

What if we ask the question, “In what city was Alan Turing likely born?” If only asking about Alan Turing and cities, Cognitive Distance would provide candidate cities like, London, Manchester, and Wilmslow, for example. But for the specific question, how would we add the context of “born in” to the computation of information distance? We only need to add a condition to all terms in the equation for the original information distance. For example the min distance would look like

dIxy|cmin = dImin(x,y|c) = {K(x|y,c), K(y|x,c)} / min{K(x|c), K(y|c)}

where c stands for the conditional information “born in” in this example. In short, distance can be computed between two things as well as for the distance of semantic triples when context is applied.

# SMB as a Compressor

There are many similar measures of “interestingness” (Tan et al., 2002), all based on the 3 fundamental counts of xy, x, and y. However, Kolmogorov complexity is the more universal distance measure. As will be described, SMB can lookup and compute over any of these counts (xy, x, y, and N) in real time. While raw counts may not tell too much to the naked eye, Saffron can easily and quickly compute a normalized distance measure that highlights the information most interesting for the user, given a certain query. Besides discovery (search), normalized information distance can be used for clustering (unsupervised learning), and classification and prediction (supervised learning). This application of the universal information distance for pattern recognition and prediction is what Saffron means by “Cognitive Computing”.

Kolmogorov complexity is based on a theory about the quantity of information in individual objects. It goes beyond traditional statistics and information theory. Kolmogorov complexity measures how compressible a string is. A string whose complexity is close to its length is a random string, while regular strings can be compressed, i.e. generated by a program with a code length that is far shorter than the string length. For example, assume two different data files. Compress them separately. Then take both files together in a single compression. To the degree that the files are similar, the compression of both files together will be smaller than the compressed files taken separately. The similarity between any two such objects is measured by how much information they share, leading to a smaller compression when combined. There is a growing recognition that intelligence is closely related to data compression (Hutter, 2004). As explained by the Hutter Prize for why compression is related to AI, “One can prove that the better you can compress, the better you can predict; and being able to predict [the environment] well is key for being able to act well.”

In essence, SMB is a compressor. Beyond a web search engine that calculates inverted indices, SMB calculates a more informative associative index consisting of a full semantic graph and its counts. Rather than data per se, SMB transforms data into these counts. These counts are directly related to the compression of information about “things”. For example, the first method of compression, after Shannon founded Information Theory, was the Shannon-Fano code, based on word frequency. Higher frequency words are given shorter codes, thus they are less complex (small K(x)). In a similar vein, SMB stores frequencies, allowing measures of information and complexity as the intelligence needed to compute patterns and predictions.

An associative memory is a non-parametric, non-functional, minimal commitment form of machine learning. In contrast to the usual reductionist data modeling approach of trying to fit observed data statistically to a model, a memory simply stores all observed connections and counts – in context – in real-time. As Hopfield described for his associative memory, a “storage prescription formula” derived from mathematics defines how an observation is instantly loaded into a matrix representation. There are no parameters, such as a “learning rates” to slowly adjust weights. There is no slow “fitting” of the model to the data. New data is learned as data arrives, instantly transformed into connections and counts.

“Modeling” is shifted to query time, which allows inference to always be up to date and flexible. Memories remain free of data assumptions and model parameters, letting the data speak for itself. To do this, we must have a very deep mathematically founded measure for similarity and distance. As discussed, distance in Hamming space, Shannon entropy, and Kolmogorov complexity are related, all offering in essence a measure of distance. If analogy is the “fuel and fire” of all our human thinking, then distance is its compliment for computation.

To be successful using this approach, of delaying computation until a question is asked, the query-time computation must be exceedingly fast. In addition to storing the required counts, the lookup and aggregation of these counts must also be instant. Particularly for queries that must quickly evaluate hundreds, thousands, or even millions of statistical pairs and triples, the physical organization of memory is critical.

References

Bargh, John A.; Chartrand, Tanya L. The Unbearable Automaticity of Being. American Psychologist, Vol 54(7), Jul 1999, 462-479.

Barigye, Stephen J.; Marrero‐Ponce, Yovani; Martínez‐López, Yoan; Torrens, Francisco; Artiles‐Martínez, Luis Manuel; Pino‐Urias, Ricardo W.; Martínez‐Santiago, Oscar. Relations frequency hypermatrices in mutual, conditional and joint entropy-based information indices. Journal of Computational Chemistry vol. 34 issue 4 5 February 2013. p. 259-274.

Bennett, C. H.; Ga ́ cs, P.,;Li, M., Vitanyi, P. M. B.; Zurek, W. Information distance. IEEE Transactions on Information Theory, IT-44, 1407–1423, 1998.

Brand, John. Make No Mistake – IBM’s Watson (And Others) Provide the **Illusion** of Cognitive Computing, blogs.forrester.com/john\_brand, May 21, 2013.

Cilibrasi, Rudi L.; Vitanyi, Paul M.B. The Google Similarity Distance. IEEE Transactions on Knowledge and Data Engineering, Vol. 19, No 3, March 2007, 370–383.

Grunwlad, Peter D.; Vitanyi, Paul. M. B. Kolmogorov Complexity and Information Theory. Journal of Logic, Language and Information, 12: 497-529, 2003.

Hofstadter, Douglas; Sander, Emmanuel. Surfaces and Essences: Analogy as the Fuel and Fire of Thinking, Basic Books, 2013.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the USA, vol. 79 no. 8 pp. 2554–2558, April 1982.

Hunt, Ira. The CIA’s “Grand Challenges” with Big Data, GigaOM Structure: Data, 2013.

Hutter, M., Universal Artificial Intelligence: Sequential Decisions

based on Algorithmic Probability, EATCS, Springer, Berlin, 2004.

Kanerva, Pentti. Sparse Distributed Memory. The MIT Press, 1988.

Klein, Gary; Snowden, David; Lock Pin, Chew. Anticipatory Thinking. Informed by Knowledge: Expert Performance in Complex Situations, [Kathleen L. Mosier](http://www.amazon.co.uk/s/ref=ntt_athr_dp_sr_1/280-7079691-9234506?_encoding=utf8&field-author=kathleen%2520l.%2520mosier&search-alias=books-uk&sort=relevancerank) and [Ute M. Fischer](http://www.amazon.co.uk/s/ref=ntt_athr_dp_sr_2/280-7079691-9234506?_encoding=utf8&field-author=ute%2520m.%2520fischer&search-alias=books-uk&sort=relevancerank) (Editors), Psychology Press, 2011.

Kraskov, Alexander; Stoegbauer, Harald; Andrzejak, Ralph G.; Grassberger, Peter. Hierarchical Clustering Based on Mutual Information, E-print arxiv.org/abs/q-bio/0311039, 2008.

Li, M., Chen, X., Li, X., Ma, B., and Vitanyi, P. The similarity metric. E-print, arxiv.org/cs.CC/0111054, 2002.

Li, Ming; Vitanyi, Paul M. B. An Introduction to Kolmogorov Complexity and Its Applications, Springer, 2008.

Leung Yan Cheong, Sik, K.; Cover, Thomas M. Some Equivalences Between Shannon Entropy and Kolmogorov Complexity. IEEE Transactions on Information Theory, Vol. 1~24, No. 3, May 1978.

McGill, W. J. Multivariate Information Transmission. Psychometrika. 1954;19:97–116.

Monash, Curt. The Underlying Technology of QlikView, www.dbms2.com, June 12, 2010.

Schacter, Daniel. The Seven Sins of Memory, Houghton Mifflin, 2001.

Sengupta, Partho P. Intelligent Platforms for Disease Assessment: Novel Approaches in Functional Echocardiography. Feigenbaum Lecture, American Society of Electrocardiography, July, 2013.

Tan, P.N.; Kumar, V; Srivastava. J, Selecting the right interestingness measure for association patterns. SIGKDD’02, Edmonton, Alberta, Canada. pp. 32-44.

Williams, Simon. Associative Model of Data. sentences.com/docs/amd.pdf, 2000.