New Millennium AI and the Convergence of History

Jürgen Schmidhuber

TU Munich, Boltzmannstr. 3, 85748 Garching bei, München, Germany &
IDSIA, Galleria 2, 6928 Manno (Lugano), Switzerland
`juergen@idsia.ch` - `http://www.idsia.ch/~juergen`

Summary. Artificial Intelligence (AI) has recently become a real formal science: the new millennium brought the first mathematically sound, asymptotically optimal, universal problem solvers, providing a new, rigorous foundation for the previously largely heuristic field of General AI and embedded agents. At the same time there has been rapid progress in practical methods for learning true sequence-processing programs, as opposed to traditional methods limited to stationary pattern association. Here we will briefly review some of the new results, and speculate about future developments, pointing out that the time intervals between the most notable events in over 40,000 years or 2^9 lifetimes of human history have sped up exponentially, apparently converging to zero within the next few decades. Or is this impression just a by-product of the way humans allocate memory space to past events?

1 Introduction

In 2003 we observed [84, 82] that each major breakthrough in computer science tends to come roughly twice as fast as the previous one, roughly matching a century-based scale: In 1623 the computing age started with the first mechanical calculator by Wilhelm Schickard (followed by machines of Pascal, 1640, and Leibniz, 1670). Roughly two centuries later Charles Babbage came up with the concept of a program-controlled computer (1834-1840). One century later, in 1931, Kurt Gödel layed the foundations of theoretical computer science with his work on universal formal languages and the limits of proof and computation. His results and Church's extensions thereof were reformulated by Turing in 1936, while Konrad Zuse built the first working program-controlled computers (1935-1941), using the binary system of Leibniz (1701) instead of the more cumbersome decimal system used by Babbage and many others. By then all the main ingredients of 'modern' computer science were in place. The next 50 years saw many less radical theoretical advances as well as faster and faster switches—relays were replaced by tubes by single transistors by numerous transistors etched on chips—but arguably this was rather predictable, incremental progress without earth-shaking events. Half a century

later, however, Tim Berners-Lee triggered the most recent world-changing development by creating the World Wide Web (1990).

Extrapolating the trend, we should expect the next radical change to manifest itself one quarter of a century after the most recent one, that is, by 2015, when some computers will already match brains in terms of raw computing power, according to frequent estimates based on Moore's law, which suggests a speed-up factor of roughly 1000 per decade, give or take a few years. The remaining series of faster and faster additional revolutions should converge in an *Omega point* (term coined by Pierre Teilhard de Chardin, 1916) expected between 2030 and 2040, when individual machines will already approach the raw computing power of all human brains combined (provided Moore's law does not break down—compare Stanislaw Ulam's concept of an approaching *historic singularity* (cited in [40]) or Vinge's closely related technological singularity [113] as well as the subsequent speculations of Moravec [48] and Kurzweil [40]). Many of the present readers of this article should still be alive then.

Will the software and the theoretical advances keep up with the hardware development? We are convinced they will. In fact, the new millennium has already brought fundamental new insights into the problem of constructing theoretically optimal rational agents or universal Artificial Intelligences (AIs, more on this below). On the other hand, on a more practical level, there has been rapid progress in learning algorithms for agents interacting with a dynamic environment, autonomously discovering true sequence-processing, problem-solving programs, as opposed to the reactive mappings from stationary inputs to outputs studied in most traditional machine learning (ML) research. In what follows, we will briefly review some of the new results, then come back to the issue of whether or not history is about to "converge."

2 Overview

Since virtually all realistic sensory inputs of robots and other cognitive systems are sequential by nature, the future of machine learning and AI in general lies in sequence processing as opposed to processing of stationary input patterns. Most traditional methods for learning time series and mappings from sequences to sequences, however, are based on simple time windows: one of the numerous feedforward ML techniques such as feedforward neural nets (NN) [4] or support vector machines [112] is used to map a restricted, fixed time window of sequential input values to desired target values. Of course such approaches are bound to fail if there are temporal dependencies exceeding the time window size. Large time windows, on the other hand, yield unacceptable numbers of free parameters.

If we want to narrow the gap between learning abilities of humans and machines, then we will have to study how to learn general algorithms instead of such reactive mappings. In what follows we will first discuss very recent

universal program learning methods that are optimal in various mathematical senses. For several reasons, however, these methods are not (yet) practically feasible. Therefore we will also discuss recent less universal but more feasible program learners based on recurrent neural networks.

Finally we will return to the introduction's topic of exponential speed-up, extending it to all of human history since the appearance of the Cro Magnon man roughly 40,000 years ago.

3 Notation

Consider a learning robotic agent with a single life which consists of discrete cycles or time steps $t = 1, 2, \ldots, T$. Its total lifetime T may or may not be known in advance. In what follows, the value of any time-varying variable Q at time t ($1 \le t \le T$) will be denoted by $Q(t)$, the ordered sequence of values $Q(1), \ldots, Q(t)$ by $Q(\le t)$, and the (possibly empty) sequence $Q(1), \ldots, Q(t-1)$ by $Q(< t)$.

At any given t the robot receives a real-valued input vector $x(t)$ from the environment and executes a real-valued action $y(t)$ which may affect future inputs; at times $t < T$ its goal is to maximize future success or *utility*

$$u(t) = E_\mu \left[\sum_{\tau=t+1}^{T} r(\tau) \ \middle| \ h(\le t) \right], \tag{1}$$

where $r(t)$ is an additional real-valued reward input at time t, $h(t)$ the ordered triple $[x(t), y(t), r(t)]$ (hence $h(\le t)$ is the known history up to t), and $E_\mu(\cdot \mid \cdot)$ denotes the conditional expectation operator with respect to some possibly unknown distribution μ from a set M of possible distributions. Here M reflects whatever is known about the possibly probabilistic reactions of the environment. For example, M may contain all computable distributions [104, 105, 42, 36]. Note that unlike in most previous work by others [38, 109], but like in much of the author's own previous work [97, 83], there is just one life, no need for predefined repeatable trials, no restriction to Markovian interfaces between sensors and environment [76], and the utility function implicitly takes into account the expected remaining lifespan $E_\mu(T \mid h(\le t))$ and thus the possibility to extend it through appropriate actions [83, 86, 90, 87].

4 Universal But Incomputable AI

Solomonoff's theoretically optimal universal predictors and their Bayesian learning algorithms [104, 105, 42, 36] only assume that the reactions of the environment are sampled from an unknown probability distribution μ contained in a set M of all enumerable distributions—compare text after equation (1). That is, given an observation sequence $q(\le t)$, we only assume there exists

a computer program that can compute the probability of the next possible $q(t+1)$, given $q(\leq t)$. Since we typically do not know this program, we predict using a mixture distribution

$$\xi(q(t+1) \mid q(\leq t)) = \sum_i w_i \mu_i(q(t+1) \mid q(\leq t)), \qquad (2)$$

a weighted sum of *all* distributions $\mu_i \in \mathcal{M}$, $i = 1, 2, \ldots$, where the sum of the positive weights satisfies $\sum_i w_i \leq 1$. It turns out that this is indeed the best one can possibly do, in a very general sense [105, 36]. The drawback is that the scheme is incomputable, since M contains infinitely many distributions.

One can increase the theoretical power of the scheme by augmenting M by certain non-enumerable but limit-computable distributions [80], or restrict it such that it becomes computable, e.g., by assuming the world is computed by some unknown but deterministic computer program sampled from the Speed Prior [81] which assigns low probability to environments that are hard to compute by any method. Under the Speed Prior the cumulative a priori probability of all data whose computation through an optimal algorithm requires more than $O(n)$ resources is $1/n$.

Can we use the optimal predictors to build an optimal AI? Indeed, in the new millennium it was shown we can. At any time t, the recent theoretically optimal yet uncomputable RL algorithm AIXI [36] uses Solomonoff's universal prediction scheme to select those action sequences that promise maximal future reward up to some horizon, typically $2t$, given the current data $h(\leq t)$. One may adapt this to the case of any finite horizon T. That is, in cycle $t+1$, AIXI selects as its next action the first action of an action sequence maximizing ξ-predicted reward up to the horizon, appropriately generalizing eq. (2). Recent work [36] demonstrated AIXI's optimal use of observations as follows. The Bayes-optimal policy p^ξ based on the mixture ξ is self-optimizing in the sense that its average utility value converges asymptotically for all $\mu \in \mathcal{M}$ to the optimal value achieved by the (infeasible) Bayes-optimal policy p^μ which knows μ in advance. The necessary condition that $\mathcal{M}$ admits self-optimizing policies is also sufficient. Furthermore, p^ξ is Pareto-optimal in the sense that there is no other policy yielding higher or equal value in *all* environments $\nu \in \mathcal{M}$ and a strictly higher value in at least one [36].

What are the implications? The first 50 years of attempts at "general AI" have been dominated by heuristic approaches [52, 69, 111, 47]. Traditionally many theoretical computer scientists have regarded the field with contempt for its lack of hard theoretical results. Things have changed, however. Although the universal approach above is practically infeasible due to the incomputability of Solomonoff's prior, it does provide, for the first time, a mathematically sound theory of AI and optimal decision making based on experience, identifying the limits of both human and artificial intelligence, and providing a yardstick for any future approach to general AI.

Using such results one can also come up with theoretically optimal ways of improving the predictive world model of a curious robotic agent [89]. The

rewards of an optimal reinforcement learner are the predictor's improvements on the observation history so far. They encourage the reinforcement learner to produce action sequences that cause the creation and the learning of new, previously unknown regularities in the sensory input stream. It turns out that art and creativity can be explained as by-products of such intrinsic curiosity rewards: good observer-dependent art deepens the observer's insights about this world or possible worlds, connecting previously disconnected patterns in an initially surprising way that eventually becomes known and boring. While previous attempts at describing what is satisfactory art or music were informal, this work permits the first *technical, formal* approach to understanding the nature of art and creativity [89].

Using the Speed Prior mentioned above, one can scale the universal approach down such that it becomes computable [81]. In what follows we will mention ways of introducing additional optimality criteria that take into account the computational costs of prediction and decision making.

5 Asymptotically Optimal General Problem Solver

To take computation time into account in a general, optimal way [41] [42, p. 502-505], the recent asymptotically optimal search algorithm for *all* well-defined problems [35] allocates part of the total search time to searching the space of proofs for provably correct candidate programs with provable upper runtime bounds; at any given time it focuses resources on those programs with the currently best proven time bounds. The method is as fast as the initially unknown fastest problem solver for the given problem class, save for a constant slowdown factor of at most $1 + \epsilon$, $\epsilon > 0$, and an additive problem class-specific constant. Unfortunately, however, the latter may be huge.

Practical applications may not ignore the constants though. This motivates the next section which addresses all kinds of optimality (not just asymptotic optimality).

6 Optimal Self-Referential General Problem Solver

The recent Gödel machines [83, 86, 90, 87] represent the first class of mathematically rigorous, general, fully self-referential, self-improving, optimally efficient problem solvers. In particular, they are applicable to the problem embodied by objective (1), which obviously does not care for asymptotic optimality.

The initial software $\mathcal{S}$ of such a Gödel machine contains an initial problem solver, e.g., one of the approaches above [36] or some less general, typical suboptimal method [38, 109]. Simultaneously, it contains an initial proof searcher (possibly based on an online variant of Levin's *Universal Search* [41]) which is used to run and test *proof techniques*. The latter are programs written in a

universal programming language implemented on the Gödel machine within $\mathcal{S}$, able to compute proofs concerning the system's own future performance, based on an axiomatic system $\mathcal{A}$ encoded in $\mathcal{S}$. $\mathcal{A}$ describes the formal *utility* function, in our case eq. (1), the hardware properties, axioms of arithmetics and probability theory and string manipulation etc, and $\mathcal{S}$ itself, which is possible without introducing circularity [83].

Inspired by Kurt Gödel's celebrated self-referential formulas (1931), the Gödel machine rewrites any part of its own code in a computable way through a self-generated executable program as soon as its *Universal Search* variant has found a proof that the rewrite is *useful* according to objective (1). According to the Global Optimality Theorem [83, 86, 90, 87], such a self-rewrite is globally optimal—no local maxima!—since the self-referential code first had to prove that it is not useful to continue the proof search for alternative self-rewrites.

If there is no provably useful, globally optimal way of rewriting $\mathcal{S}$ at all, then humans will not find one either. But if there is one, then $\mathcal{S}$ itself can find and exploit it. Unlike previous *non*-self-referential methods based on hard-wired proof searchers [36], Gödel machines not only boast an optimal *order* of complexity but can optimally reduce (through self-changes) any slowdowns hidden by the $O()$-notation, provided the utility of such speed-ups is provable at all.

Practical implementations of the Gödel machine do not yet exist though, and probably require a thoughtful choice of the initial axioms and the initial proof searcher. In the next sections we will deal with already quite practical, non-optimal and non-universal, but still rather general searchers in program space, as opposed to the space of reactive, feedforward input / output mappings, which still attracts the bulk of current machine learning research.

7 Supervised Recurrent Neural Networks

Recurrent neural networks (RNNs) are neural networks [4] with feedback connections that are, in principle, as powerful as any traditional computer. There is a very simple way to see this [74, 79]: a traditional microprocessor may be viewed as a very sparsely connected RNN consisting of very simple neurons implementing nonlinear AND and NAND gates, etc. Compare [100] for a more complex argument. Early RNNs [34, 1, 60] did not exploit this potential power because they were limited to static inputs. However, most interesting tasks involve processing sequences that consist of continually varying inputs. Examples include robot control, speech recognition, music composition, attentive vision, and numerous others.

The initial RNN enthusiasm of the 1980s and early 90s was fueled by the obvious theoretical advantages of RNNs: unlike feedforward neural networks (FNNs) [115, 70], Support Vector Machines (SVMs), and related approaches [112], RNNs have an internal state which is essential for many tasks involving program learning and sequence learning. And unlike in Hidden Markov Models

(HMMs) [125], internal states can take on continuous values, and the influence of these states can persist for many timesteps. This allows RNN to solve tasks that are impossible for HMMs, such as learning the rules of certain context-free languages [16].

Practitioners of the early years, however, have been sobered by unsuccessful attempts to apply RNNs to important real world problems that require sequential processing of information. The first RNNs simply did not work very well, and their operation was poorly understood since it is inherently more complex than that of FNNs. FNNs fit neatly into the framework of traditional statistics and information theory [99], while RNNs require additional insights, e.g., from theoretical computer science and *algorithmic* information theory [42, 80].

Fortunately, recent advances have overcome the major drawbacks of traditional RNNs. That's why RNNs are currently experiencing a second wave of attention. New architectures, learning algorithms, and a better understanding of RNN behavior have allowed RNNs to learn many previously unlearnable tasks. RNN optimists are claiming that we are at the beginning of a *"RN-Naissance"* [92, 85], and that soon we will see more and more applications of the new RNNs.

Sequence-processing, supervised, gradient-based RNNs were pioneered by Werbos [116], Williams [121], and Robinson & Fallside [65]; compare Pearlmutter's survey [56]. The basic idea is: a teacher-given training set contains example sequences of inputs and desired outputs or targets $d_k(t)$ for certain neurons y_k at certain times t. If the i-th neuron y_i is an input neuron, then its real-valued activation $y_i(t)$ at any given discrete time step $t = 1, 2, \ldots$ is set by the environment. Otherwise $y_i(t)$ is a typically nonlinear function $f(\cdot)$ of the $y_k(t-1)$ of all neurons y_k connected to y_i, where by default $y_i(0) = 0$ for all i. By choosing an appropriate $f(\cdot)$, we make the network dynamics differentiable, e.g., $y_i(t) = arctan(\sum_k (w_{ik} y_k(t-1)))$, where w_{ik} is the real-valued weight on the connection from y_k to y_i. Then we use gradient descent to change the weights such that they minimize an objective function E reflecting the differences between actual and desired output sequences. Popular gradient descent algorithms for computing weight changes $\triangle w_{ik} \sim \frac{\partial E}{\partial w_{ik}}$ include *Back-Propagation Through Time* (BPTT) [116, 121] and *Real-Time Recurrent Learning* (RTRL) [65, 121] and mixtures thereof [121, 77].

The nice thing about gradient-based RNNs is that we can **differentiate our wishes with respect to programs,** e.g., [75, 74, 79]. The set of possible weight matrices represents a continuous space of programs, and the objective function E represents our desire to minimize the difference between what the network does and what it should do. The gradient $\frac{\partial E}{\partial w}$ (were w is the complete weight matrix) tells us how to change the current program such that it will be better at fulfilling our wish.

Typical RNNs trained by BPTT and RTRL and other previous approaches [50, 51, 107, 43, 61, 64, 10, 56, 55, 12, 13, 121, 77, 78, 122, 114, 45, 62], however, cannot learn to look far back into the past. Their problems were first rigorously

analyzed in 1991 on the author's RNN long time lag project [29, 30]; also compare Werbos' concept of "sticky neurons" [117]. The error signals "flowing backwards in time" tend to either (1) blow up or (2) vanish: the temporal evolution of the back-propagated error exponentially depends on the weight magnitudes. Case (1) may lead to oscillating weights. In case (2), learning to bridge long time lags takes a prohibitive amount of time, or does not work at all. So then why bother with RNNs at all? For short time lags we could instead use short time-windows combined with non-recurrent approaches such as multi-layer perceptrons [4] or better *Support Vector Machines* SVMs [112].

An RNN called "Long Short-Term Memory" or LSTM (Figure S1) [32] overcomes the fundamental problems of traditional RNNs, and efficiently learns to solve many previously unlearnable tasks involving: Recognition of temporally extended patterns in noisy input sequences [32, 17]; Recognition of the temporal order of widely separated events in noisy input streams [32]; Extraction of information conveyed by the temporal distance between events [15]; Stable generation of precisely timed rhythms, smooth and non-smooth periodic trajectories; Robust storage of high-precision real numbers across extended time intervals; Arithmetic operations on continuous input streams [32, 14]. This made possible the numerous applications described further below.

We found [32, 17] that LSTM clearly outperforms previous RNNs on tasks that require learning the rules of regular languages (RLs) describable by deterministic finite state automata (DFA) [8, 101, 5, 39, 126], both in terms of reliability and speed. In particular, problems that are hardly ever solved by standard RNNs, are solved by LSTM in nearly 100% of all trials, LSTM's superiority also carries over to tasks involving context free languages (CFLs) such as those discussed in the RNN literature [108, 120, 106, 110, 67, 68]. Their recognition requires the functional equivalent of a stack. Some previous RNNs even failed to learn small CFL training sets [68]. Those that did not and those that even learned small CSL training sets [67, 6] failed to extract the general rules, and did not generalize well on substantially larger test sets. In contrast, LSTM generalizes extremely well. It requires only the 30 shortest exemplars ($n \leq 10$) of the context sensitive language $a^n b^n c^n$ to correctly predict the possible continuations of sequence prefixes for n up to 1000 and more.

Kalman filters can further improve LSTM's performance [58]. A combination of LSTM and the decoupled extended Kalman filter learned to deal correctly with values of n up to 10 million and more. That is, after training the network was able to read sequences of 30,000,000 symbols and more, one symbol at a time, and finally detect the subtle differences between *legal* strings such as $a^{10,000,000} b^{10,000,000} c^{10,000,000}$ and very similar but *illegal* strings such as $a^{10,000,000} b^{9,999,999} c^{10,000,000}$. This illustrates that LSTM networks can work in an extremely precise and robust fashion across very long time lags.

Speech recognition is still dominated by Hidden Markov Models (HMMs), e.g., [7]. HMMs and other graphical models such as Dynamic Bayesian Networks (DBN) *do* have internal states that can be used to model memories

of previously seen inputs. Under certain circumstances they allow for learning the prediction of sequences of labels from unsegmented input streams. For example, an unsegmented acoustic signal can be transcribed into a sequence of words or phonemes. HMMs are well-suited for noisy inputs and are invariant to non-linear temporal stretching—they do not care for the difference between slow and fast versions of a given spoken word. At least in theory, however, RNNs could offer the following advantages: Due to their discrete nature, HMMs either have to discard real-valued information about timing, rhythm, prosody, etc., or use numerous hidden states to encode such potentially relevant information in discretized fashion. RNNs, however, can naturally use their real-valued activations to encode it compactly. Furthermore, in order to make HMM training tractable, it is necessary to assume that successive inputs are independent of one another. In most interesting sequence learning tasks, such as speech recognition, this is manifestly untrue. Because RNNs model the conditional probabilities of class occupancy directly, and do not model the input sequence, they do not require this assumption. RNN classifications are in principle conditioned on the entire input sequence. Finally, HMMs cannot even model the rules of context-free languages, while RNNs can [17, 16, 18, 94, 59].

LSTM recurrent networks were trained from scratch on utterances from the TIDIGITS speech database. It was found [3, 25, 26] that LSTM obtains results comparable to HMM based systems. A series of experiments on disjoint subsets of the database demonstrated that previous experience greatly reduces the network's training time, as well as increasing its accuracy. It was therefore argued that LSTM is a promising tool for applications requiring either rapid cross corpus adaptation or continually expanding datasets. Substantial promise also lies in LSTM-HMM hybrids that try to combine the best of both worlds, inspired by Robinson's hybrids based on traditional RNNs [66]. Recently we showed [28, 27] that LSTM learns framewise phoneme classification much faster than previous RNNs. Best results were obtained with a bi-directional variant of LSTM that classifies any part of a sequence by taking into account its entire past and future context.

Gradient-based LSTM also has been used to identify protein sequence motifs that contribute to classification [31]. Protein classification is important for extracting binding or active sites on a protein in order to develop new drugs, and in determining 3D protein folding features that can provide a better understanding of diseases resulting from protein misfolding.

Sometimes gradient information is of little use due to rough error surfaces with numerous local minima. For such cases, we have recently introduced a new, evolutionary/gradient-descent hybrid method for training LSTM and other RNNs called Evolino [96, 119, 93, 95]. Evolino evolves weights to the nonlinear, hidden nodes of RNNs while computing optimal linear mappings from hidden state to output, using methods such as pseudo-inverse-based linear regression [57] or support vector machine-like quadratic programming [112],

depending on the notion of optimality employed. Evolino-based LSTM can solve tasks that Echo State networks [37] cannot, and achieves higher accuracy in certain continuous function generation tasks than gradient-based LSTM, as well as other conventional gradient descent RNNs. However, for several problems requiring large networks with numerous learnable weights, gradient-based LSTM was superior to Evolino-based LSTM.

8 Reinforcement-Learning / Evolving Recurrent Neural Networks

In a certain sense, Reinforcement Learning (RL) is more challenging than supervised learning as above, since there is no teacher providing desired outputs at appropriate time steps. To solve a given problem, the learning agent itself must discover useful output sequences in response to the observations. The traditional approach to RL is best embodied by Sutton and Barto's book [109]. It makes strong assumptions about the environment, such as the Markov assumption: the current input of the agent tells it all it needs to know about the environment. Then all we need to learn is some sort of reactive mapping from stationary inputs to outputs. This is often unrealistic.

More general approaches search a space of truly sequence-processing programs with temporary variables for storing previous observations. For example, Olsson's ADATE [54] or related approaches such as *Genetic Programming (GP)* [9, 11, 73] can in principle be used to evolve such programs by maximizing an appropriate objective or fitness function. *Probabilistic Incremental Program Evolution (PIPE)* [71] is a related technique for automatic program synthesis, combining probability vector coding of program instructions [97] and Population-Based Incremental Learning [2] and tree-coded programs. PIPE was used for learning soccer team strategies in rather realistic simulations [72, 118].

A related, rather general approach for partially observable environments directly evolves programs for recurrent neural networks (RNN) with internal states, by applying evolutionary algorithms [63, 98, 33] to RNN weight matrices [46, 124, 123, 53, 44, 102, 49]. RNN can run general programs with memory / internal states (no need for the Markovian assumption), but for a long time it was unclear how to efficiently evolve their weights to solve complex RL tasks. Recent work, however, brought progress through a focus on reducing search spaces by co-evolving the comparatively small weight vectors of individual recurrent neurons [21, 22, 20, 23, 19, 24]. The powerful RNN learn to use their potential to create memories of important events, solving numerous RL / optimization tasks unsolvable by traditional RL methods [23, 19, 24]. As mentioned in the previous section, even supervised learning can greatly profit from this approach [96, 119, 93, 95].

9 Is History Converging? Again?

Many predict that within a few decades there will be computers whose raw computing power will surpass the one of a human brain by far (e.g., [48, 40]). We have argued that algorithmic advances are keeping up with the hardware development, pointing to new-millennium theoretical insights on universal problem solvers that are optimal in various mathematical senses (thus making *General AI* a real formal science), as well as practical progress in program learning through brain-inspired recurrent neural nets (as opposed to mere pattern association through traditional reactive devices).

Let us put the AI-oriented developments [88] discussed above in a broader context, and extend the analysis of past computer science breakthroughs in the introduction, which predicts that computer history will converge in an *Omega point* or historic *singularity* Ω around 2040.

Surprisingly, even if we go back all the way to the beginnings of modern man over 40,000 years ago, essential historic developments (that is, the subjects of the major chapters in history books) match a a binary scale marking exponentially declining temporal intervals, each half the size of the previous one, and even measurable in terms of powers of 2 multiplied by a human lifetime [91] (roughly 80 years—throughout recorded history many individuals have reached this age, although the average lifetime often was shorter, mostly due to high children mortality). Using the value $\Omega = 2040$, associate an error bar of not much more than 10 percent with each date below:

1. $\Omega - 2^9$ lifetimes: modern humans start colonizing the world from Africa
2. $\Omega - 2^8$ lifetimes: bow and arrow invented; hunting revolution
3. $\Omega - 2^7$ lifetimes: invention of agriculture; first permanent settlements; beginnings of civilization
4. $\Omega - 2^6$ lifetimes: first high civilizations (Sumeria, Egypt), and the most important invention of recorded history, namely, the one that made recorded history possible: writing
5. $\Omega - 2^5$ lifetimes: the ancient Greeks invent democracy and lay the foundations of Western science and art and philosophy, from algorithmic procedures and formal proofs to anatomically perfect sculptures, harmonic music, and organized sports. Old Testament written, major Asian religions founded. High civilizations in China, origin of the first calculation tools, and India, origin of the zero
6. $\Omega - 2^4$ lifetimes: bookprint (often called the most important invention of the past 2000 years) invented in China. Islamic science and culture start spreading across large parts of the known world (this has sometimes been called the most important event between Antiquity and the age of discoveries)
7. $\Omega - 2^3$ lifetimes: the largest and most dominant empire ever (perhaps including more than half of humanity and two thirds of the world economy) stretches across Asia from Korea all the way to Germany. Chinese

fleets and later also European vessels start exploring the world. Gun powder and guns invented in China. Rennaissance and Western bookprint (often called the most influential invention of the past 1000 years) and subsequent Reformation in Europe. Begin of the Scientific Revolution

8. $\Omega - 2^2$ lifetimes: Age of enlightenment and rational thought in Europe. Massive progress in the sciences; first flying machines; start of the industrial revolution based on the first steam engines

9. $\Omega - 2$ lifetimes: Second industrial revolution based on combustion engines, cheap electricity, and modern chemistry. Birth of modern medicine through the germ theory of disease; genetic and evolution theory. European colonialism at its short-lived peak

10. $\Omega - 1$ lifetime: modern post-World War II society and pop culture emerges. The world-wide super-exponential population explosion (mainly due to the Haber-Bosch process [103]) is at its peak. First commercial computers and first spacecraft; DNA structure unveiled

11. $\Omega - 1/2$ lifetime: 3rd industrial revolution based on personal computers and the World Wide Web. A mathematical theory of universal AI emerges (see sections above) - will this be considered a milestone in the future?

12. $\Omega - 1/4$ lifetime: This point will be reached in a few years. See introduction

13. ...

The following disclosure should help the reader to take this list with a grain of salt though. The author, who admits being very interested in witnessing the Omega point, was born in 1963, and therefore perhaps should not expect to live long past 2040. This may motivate him to uncover certain historic patterns that fit his desires, while ignoring other patterns that do not.

Others may feel attracted by the same trap. For example, Kurzweil [40] identifies exponential speedups in sequences of historic paradigm shifts identified by various historians, to back up the hypothesis that "the singularity is near." His historians are all contemporary though, presumably being subject to a similar bias. People of past ages might have held quite different views. For example, possibly some historians of the year 1525 felt inclined to predict a convergence of history around 1540, deriving this date from an exponential speedup of recent breakthroughs such as Western bookprint (around 1444), the re-discovery of America (48 years later), the Reformation (again 24 years later—see the pattern?), and other events they deemed important although today they are mostly forgotten.

In fact, could it be that such lists just reflect the human way of allocating memory space to past events? Maybe there is a general rule for both the individual memory of single humans and the collective memory of entire societies and their history books: constant amounts of memory space get allocated to exponentially larger, adjacent time intervals further and further into the past. For example, events that happened between 2 and 4 lifetimes ago get roughly as much memory space as events in the previous interval of twice the size. Presumably only a few "important" memories will survive the necessary

compression. Maybe that's why there has never been a shortage of prophets predicting that the end is near - the important events according to one's own view of the past always seem to accelerate exponentially.

A similar plausible type of memory decay allocates $O(1/n)$ memory units to all events older than $O(n)$ unit time intervals. This is reminiscent of a bias governed by a time-reversed Speed Prior [81] (Section 4).

References

[1] L. B. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *IEEE 1st International Conference on Neural Networks, San Diego*, volume 2, pages 609–618, 1987.

[2] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In A. Prieditis and S. Russell, editors, *Machine Learning: Proceedings of the Twelfth International Conference*, pages 38–46. Morgan Kaufmann Publishers, San Francisco, CA, 1995.

[3] N. Beringer, A. Graves, F. Schiel, and J. Schmidhuber. Classifying unprompted speech by retraining LSTM nets. In W. Duch, J. Kacprzyk, E. Oja, and S. Zadrozny, editors, *Artificial Neural Networks: Biological Inspirations - ICANN 2005, LNCS 3696*, pages 575–581. Springer-Verlag Berlin Heidelberg, 2005.

[4] C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[5] Alan D. Blair and Jordan B. Pollack. Analysis of dynamical recognizers. *Neural Computation*, 9(5):1127–1142, 1997.

[6] M. Boden and J. Wiles. Context-free and context-sensitive dynamics in recurrent neural networks. *Connection Science*, 2000.

[7] H.A. Bourlard and N. Morgan. *Connnectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, 1994.

[8] M. P. Casey. The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Computation*, 8(6):1135–1178, 1996.

[9] N. L. Cramer. A representation for the adaptive generation of simple sequential programs. In J.J. Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms and Their Applications, Carnegie-Mellon University, July 24-26, 1985*, Hillsdale NJ, 1985. Lawrence Erlbaum Associates.

[10] B. de Vries and J. C. Principe. A theory for neural networks with time delays. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 162–168. Morgan Kaufmann, 1991.

[11] D. Dickmanns, J. Schmidhuber, and A. Winklhofer. Der genetische Algorithmus: Eine Implementierung in Prolog. Fortgeschrittenenpraktikum, Institut für Informatik, Lehrstuhl Prof. Radig, Technische Universität München, 1987.

[12] J. L. Elman. Finding structure in time. Technical Report CRL Technical Report 8801, Center for Research in Language, University of California, San Diego, 1988.

[13] S. E. Fahlman. The recurrent cascade-correlation learning algorithm. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 190–196. Morgan Kaufmann, 1991.

[14] F. A. Gers and J. Schmidhuber. Neural processing of complex continual input streams. In *Proc. IJCNN'2000, Int. Joint Conf. on Neural Networks*, Como, Italy, 2000.

[15] F. A. Gers and J. Schmidhuber. Recurrent nets that time and count. In *Proc. IJCNN'2000, Int. Joint Conf. on Neural Networks*, Como, Italy, 2000.

[16] F. A. Gers and J. Schmidhuber. LSTM recurrent networks learn simple context free and context sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.

[17] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000.

[18] F. A. Gers, N. Schraudolph, and J. Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3:115–143, 2002.

[19] F. Gomez and J. Schmidhuber. Evolving modular fast-weight networks for control. In W. Duch, J. Kacprzyk, E. Oja, and S. Zadrozny, editors, *Artificial Neural Networks: Biological Inspirations - ICANN 2005, LNCS 3697*, pages 383–389. Springer-Verlag Berlin Heidelberg, 2005.

[20] F. J. Gomez. *Robust Nonlinear Control through Neuroevolution*. PhD thesis, Department of Computer Sciences, University of Texas at Austin, 2003.

[21] F. J. Gomez and R. Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5:317–342, 1997.

[22] F. J. Gomez and R. Miikkulainen. Solving non-Markovian control tasks with neuroevolution. In *Proc. IJCAI 99*, Denver, CO, 1999. Morgan Kaufman.

[23] F. J. Gomez and R. Miikkulainen. Active guidance for a finless rocket using neuroevolution. In *Proc. GECCO 2003, Chicago*, 2003. *Winner of Best Paper Award in Real World Applications. Gomez is working at IDSIA on a CSEM grant to J. Schmidhuber.*

[24] F. J. Gomez and J. Schmidhuber. Co-evolving recurrent neurons learn deep memory POMDPs. In *Proc. of the 2005 conference on genetic and evolutionary computation (GECCO), Washington, D. C.* ACM Press, New York, NY, USA, 2005. *Nominated for a best paper award.*

[25] A. Graves, N. Beringer, and J. Schmidhuber. Rapid retraining on speech data with LSTM recurrent networks. Technical Report IDSIA-09-05, IDSIA, www.idsia.ch/techrep.html, 2005.

[26] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural nets. In *ICML'06: Proceedings of the International Conference on Machine Learning*, 2006.

[27] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18:602–610, 2005.

[28] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM networks. In *Proc. Int. Joint Conf. on Neural Networks IJCNN 2005*, 2005.

[29] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991. See www7.informatik.tu-muenchen.de/ ~hochreit; advisor: J. Schmidhuber.

[30] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

[31] S. Hochreiter and K. Obermayer. Sequence classification for protein analysis. In *Snowbird Workshop*, Snowbird, Utah, April 5-8 2005. Computational and Biological Learning Society.

[32] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[33] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[34] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Sciences*, 79:2554–2558, 1982.

[35] M. Hutter. The fastest and shortest algorithm for all well-defined problems. *International Journal of Foundations of Computer Science*, 13(3):431–443, 2002. (On J. Schmidhuber's SNF grant 20-61847).

[36] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2004. (On J. Schmidhuber's SNF grant 20-61847).

[37] H. Jaeger. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304:78–80, 2004.

[38] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of AI research*, 4:237–285, 1996.

[39] Y. Kalinke and H. Lehmann. Computation in recurrent neural networks: From counters to iterated function systems. In G. Antoniou and J. Slaney, editors, *Advanced Topics in Artificial Intelligence, Proceedings of the 11th Australian Joint Conference on Artificial Intelligence*, volume 1502 of *LNAI*, Berlin, Heidelberg, 1998. Springer.

[40] R. Kurzweil. *The Singularity is near*. Wiley Interscience, 2005.

[41] L. A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.

[42] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications (2nd edition)*. Springer, 1997.

[43] T. Lin, B.G. Horne, P. Tino, and C.L. Giles. Learning long-term dependencies in NARX recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6):1329–1338, 1996.

[44] O. Miglino, H. Lund, and S. Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4):417–434, 1995.

[45] C. B. Miller and C. L. Giles. Experimental comparison of the effect of order in recurrent neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):849–872, 1993.

[46] G. Miller, P. Todd, and S. Hedge. Designing neural networks using genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 379–384. Morgan Kauffman, 1989.

[47] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[48] H. Moravec. *Robot*. Wiley Interscience, 1999.

[49] D. E. Moriarty and R. Miikkulainen. Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22:11–32, 1996.

[50] M. C. Mozer. A focused back-propagation algorithm for temporal sequence recognition. *Complex Systems*, 3:349–381, 1989.

[51] M. C. Mozer. Induction of multiscale temporal structure. In D. S. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 4*, pages 275–282. Morgan Kaufmann, 1992.

[52] A. Newell and H. Simon. GPS, a program that simulates human thought. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*, pages 279–293. McGraw-Hill, New York, 1963.

[53] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In R. A. Brooks and P. Maes, editors, *Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV)*, pages 190–197. MIT, 1994.

[54] J. R. Olsson. Inductive functional programming using incremental program transformation. *Artificial Intelligence*, 74(1):55–83, 1995.

[55] B. A. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2):263–269, 1989.

[56] B. A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural Networks*, 6(5):1212–1228, 1995.

[57] R. Penrose. A generalized inverse for matrices. In *Proceedings of the Cambridge Philosophy Society*, volume 51, pages 406–413, 1955.

[58] J. A. Pérez-Ortiz, F. A. Gers, D. Eck, and J. Schmidhuber. Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets. *Neural Networks*, (16):241–250, 2003.

[59] J. A. Pérez-Ortiz, F. A. Gers, D. Eck, and J. Schmidhuber. Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets. *Neural Networks*, 16(2):241–250, 2003.

[60] F. J. Pineda. Recurrent backpropagation and the dynamical approach to adaptive neural computation. *Neural Computation*, 1(2):161–172, 1989.

[61] T. A. Plate. Holographic recurrent networks. In J. D. Cowan S. J. Hanson and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 34–41. Morgan Kaufmann, 1993.

[62] G. V. Puskorius and L. A. Feldkamp. Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks. *IEEE Transactions on Neural Networks*, 5(2):279–297, 1994.

[63] I. Rechenberg. Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Dissertation, 1971. Published 1973 by Fromman-Holzboog.

[64] M. B. Ring. Learning sequential tasks by incrementally adding higher orders. In J. D. Cowan S. J. Hanson and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 115–122. Morgan Kaufmann, 1993.

[65] A. J. Robinson and F. Fallside. The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Cambridge University Engineering Department, 1987.

[66] Anthony J. Robinson. An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks*, 5(2):298–305, March 1994.

[67] P. Rodriguez, J. Wiles, and J Elman. A recurrent neural network that learns to count. *Connection Science*, 11(1):5–40, 1999.

[68] Paul Rodriguez and Janet Wiles. Recurrent neural networks can learn to implement symbol-sensitive counting. In *Advances in Neural Information Processing Systems*, volume 10, pages 87–93. The MIT Press, 1998.

[69] P. S. Rosenbloom, J. E. Laird, and A. Newell. *The SOAR Papers*. MIT Press, 1993.

[70] D. E. Rumelhart and J. L. McClelland, editors. *Parallel Distributed Processing*, volume 1. MIT Press, 1986.

[71] R. P. Sałustowicz and J. Schmidhuber. Probabilistic incremental program evolution. *Evolutionary Computation*, 5(2):123–141, 1997.

[72] R. P. Sałustowicz, M. A. Wiering, and J. Schmidhuber. Learning team strategies: Soccer case studies. *Machine Learning*, 33(2/3):263–282, 1998.

[73] J. Schmidhuber. Evolutionary principles in self-referential learning. Diploma thesis, Institut für Informatik, Technische Universität München, 1987.

[74] J. Schmidhuber. Dynamische neuronale Netze und das fundamentale raumzeitliche Lernproblem. Dissertation, Institut für Informatik, Technische Universität München, 1990.

[75] J. Schmidhuber. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *Proc. IEEE/INNS International Joint Conference on Neural Networks, San Diego*, volume 2, pages 253–258, 1990.

[76] J. Schmidhuber. Reinforcement learning in Markovian and non-Markovian environments. In D. S. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3 (NIPS 3)*, pages 500–506. Morgan Kaufmann, 1991.

[77] J. Schmidhuber. A fixed size storage $O(n^3)$ time complexity learning algorithm for fully recurrent continually running networks. *Neural Computation*, 4(2):243–248, 1992.

[78] J. Schmidhuber. Learning to control fast-weight memories: An alternative to recurrent nets. *Neural Computation*, 4(1):131–139, 1992.

[79] J. Schmidhuber. Netzwerkarchitekturen, Zielfunktionen und Kettenregel. Habilitationsschrift, Institut für Informatik, Technische Universität München, 1993.

[80] J. Schmidhuber. Hierarchies of generalized Kolmogorov complexities and nonenumerable universal measures computable in the limit. *International Journal of Foundations of Computer Science*, 13(4):587–612, 2002.

[81] J. Schmidhuber. The Speed Prior: a new simplicity measure yielding near-optimal computable predictions. In J. Kivinen and R. H. Sloan, editors, *Proceedings of the 15th Annual Conference on Computational Learning Theory (COLT 2002)*, Lecture Notes in Artificial Intelligence, pages 216–228. Springer, Sydney, Australia, 2002.

[82] J. Schmidhuber. Exponential speed-up of computer history's defining moments, 2003. http://www.idsia.ch/~juergen/computerhistory.html.

[83] J. Schmidhuber. Gödel machines: self-referential universal problem solvers making provably optimal self-improvements. Technical Report IDSIA-19-03, arXiv:cs.LO/0309048, IDSIA, Manno-Lugano, Switzerland, 2003.

[84] J. Schmidhuber. The new AI: General & sound & relevant for physics. Technical Report TR IDSIA-04-03, Version 1.0, cs.AI/0302012 v1, February 2003.

[85] J. Schmidhuber. Overview of work on robot learning, with publications, 2004. http://www.idsia.ch/~juergen/learningrobots.html.

[86] J. Schmidhuber. Completely self-referential optimal reinforcement learners. In W. Duch, J. Kacprzyk, E. Oja, and S. Zadrozny, editors, *Artificial Neural Networks: Biological Inspirations - ICANN 2005, LNCS 3697*, pages 223–233. Springer-Verlag Berlin Heidelberg, 2005. Plenary talk.

[87] J. Schmidhuber. Gödel machines: Towards a technical justification of consciousness. In D. Kudenko, D. Kazakov, and E. Alonso, editors, *Adaptive Agents and Multi-Agent Systems III (LNCS 3394)*, pages 1–23. Springer Verlag, 2005.

[88] J. Schmidhuber. Artificial Intelligence - history highlights and outlook: AI maturing and becoming a real formal science, 2006. http://www.idsia.ch/˜juergen/ai.html.

[89] J. Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006.

[90] J. Schmidhuber. Gödel machines: fully self-referential optimal universal problem solvers. In B. Goertzel and C. Pennachin, editors, *Artificial General Intelligence*. Springer Verlag, in press, 2006.

[91] J. Schmidhuber. Is history converging? Again?, 2006. http://www.idsia.ch/˜juergen/history.html.

[92] J. Schmidhuber and B. Bakker. NIPS 2003 RNNaissance workshop on recurrent neural networks, Whistler, CA, 2003. http://www.idsia.ch/˜juergen/rnnaissance.html.

[93] J. Schmidhuber, M. Gagliolo, D. Wierstra, and F. Gomez. Evolino for recurrent support vector machines. In *ESANN'06*, 2006.

[94] J. Schmidhuber, F. Gers, and D. Eck. Learning nonregular languages: A comparison of simple recurrent networks and LSTM. *Neural Computation*, 14(9):2039–2041, 2002.

[95] J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez. Training recurrent networks by EVOLINO. *Neural Computation*, 2006, in press.

[96] J. Schmidhuber, D. Wierstra, and F. J. Gomez. Evolino: Hybrid neuroevolution / optimal linear search for sequence prediction. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 853–858, 2005.

[97] J. Schmidhuber, J. Zhao, and N. Schraudolph. Reinforcement learning with self-modifying policies. In S. Thrun and L. Pratt, editors, *Learning to learn*, pages 293–309. Kluwer, 1997.

[98] H. P. Schwefel. Numerische Optimierung von Computer-Modellen. Dissertation, 1974. Published 1977 by Birkhäuser, Basel.

[99] C. E. Shannon. A mathematical theory of communication (parts I and II). *Bell System Technical Journal*, XXVII:379–423, 1948.

[100] H. T. Siegelmann and E. D. Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77–80, 1991.

[101] H.T. Siegelmann. *Theoretical Foundations of Recurrent Neural Networks*. PhD thesis, Rutgers, New Brunswick Rutgers, The State of New Jersey, 1992.

[102] K. Sims. Evolving virtual creatures. In Andrew Glassner, editor, *Proceedings of SIGGRAPH'94 (Orlando, Florida, July 1994)*, Computer Graphics Proceedings, Annual Conference, pages 15–22. ACM SIGGRAPH, ACM Press, jul 1994. ISBN 0-89791-667-0.

[103] V. Smil. Detonator of the population explosion. *Nature*, 400:415, 1999.

[104] R. J. Solomonoff. A formal theory of inductive inference. Part I. *Information and Control*, 7:1–22, 1964.

[105] R. J. Solomonoff. Complexity-based induction systems. *IEEE Transactions on Information Theory*, IT-24(5):422–432, 1978.

[106] M. Steijvers and P.D.G. Grunwald. A recurrent network that performs a contextsensitive prediction task. In *Proceedings of the 18th Annual Conference of the Cognitive Science Society*. Erlbaum, 1996.

[107] G. Sun, H. Chen, and Y. Lee. Time warping invariant neural networks. In J. D. Cowan S. J. Hanson and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 180–187. Morgan Kaufmann, 1993.

[108] G. Z. Sun, C. Lee Giles, H. H. Chen, and Y. C. Lee. The neural network pushdown automaton: Model, stack and learning simulations. Technical Report CS-TR-3118, University of Maryland, College Park, August 1993.

[109] R. Sutton and A. Barto. *Reinforcement learning: An introduction.* Cambridge, MA, MIT Press, 1998.

[110] B. Tonkes and J. Wiles. Learning a context-free task with a recurrent neural network: An analysis of stability. In *Proceedings of the Fourth Biennial Conference of the Australasian Cognitive Science Society*, 1997.

[111] P. Utgoff. Shift of bias for inductive concept learning. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning*, volume 2, pages 163–190. Morgan Kaufmann, Los Altos, CA, 1986.

[112] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[113] V. Vinge. The coming technological singularity, 1993. VISION-21 Symposium sponsored by NASA Lewis Research Center, and Whole Earth Review, Winter issue.

[114] R. L. Watrous and G. M. Kuhn. Induction of finite-state languages using second-order recurrent networks. *Neural Computation*, 4:406–414, 1992.

[115] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[116] P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1, 1988.

[117] P. J. Werbos. Neural networks, system identification, and control in the chemical industries. In D. A. Sofge D. A. White, editor, *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, pages 283–356. Thomson Learning, 1992.

[118] M. A. Wiering, R. P. Salustowicz, and J. Schmidhuber. Reinforcement learning soccer teams with incomplete world models. *Autonomous Robots*, 7(1):77–88, 1999.

[119] D. Wierstra, F. J. Gomez, and J. Schmidhuber. Modeling systems with internal state using Evolino. In *Proc. of the 2005 conference on genetic and evolutionary computation (GECCO), Washington, D. C.*, pages 1795–1802. ACM Press, New York, NY, USA, 2005. Got a GECCO best paper award.

[120] J. Wiles and J. Elman. Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent networks. In *In Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, pages pages 482 – 487, Cambridge, MA, 1995. MIT Press.

[121] R. J. Williams. Complexity of exact gradient computation algorithms for recurrent neural networks. Technical Report Technical Report NU-CCS-89-27, Boston: Northeastern University, College of Computer Science, 1989.

[122] R. J. Williams and J. Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 4:491–501, 1990.

[123] B. M. Yamauchi and R. D. Beer. Sequential behavior and learning in evolved dynamical neural networks. *Adaptive Behavior*, 2(3):219–246, 1994.

[124] Xin Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 4:203–222, 1993.

[125] S.J. Young and P.C Woodland. *HTK Version 3.2: User, Reference and Programmer Manual*, 2002.

[126] Z. Zeng, R. Goodman, and P. Smyth. Discrete recurrent neural networks for grammatical inference. *IEEE Transactions on Neural Networks*, 5(2), 1994.