

This paper records a 30-minute talk given in the Paul Erdos Lecture series in March, 2010. Though the talk was prepared for mathematicians with low Erdos numbers (a widely respected measure of mathematical proficiency), I hope that the main discussion will be simplified and clear enough to others. Under each slide I will also insert further comments – some more elementary and explanatory, but others more precise – under the ===== tracks. These are not just footnotes. In some cases they are crucial to the research which needs to be done next.

Mathematical Foundations of Prediction Under Complexity

Paul J. Werbos, pwerbos@nsf.gov
U.S. National Science Foundation*

- Why this is a crucial and timely piece of a larger problem
- Roadmap and definitions from vector prediction to grid and graph prediction and beyond
- Why it is not easy and not yet solved
- What must be built upon and improved

*The views herein are those of the author, not official views of NSF

At NSF, our job is mainly to ask important questions – and then try to encourage and support **you** to find the answers. In this talk, I will review some of the methods used to make predictions across many disciplines in science, technology, and social science. I will even mention a few partial solutions I have found myself to the problem of accurate prediction in the face of complexity. However, by the end of the talk, I hope you will agree with me that all the existing methods – including those I have developed myself – are at best starting points. You could do better, and I hope you do. That is why I am grateful for the chance to talk to you.

I will begin by motivating the problem, then defining it more precisely, and then reviewing where we stand.

The published abstract for this lecture:

Mathematical Foundations of Prediction Under Complexity

Consider the following dynamical system: $x(t) = h(r(t), e_1(t))$, $r(t) = f(r(t-1), e_2(t))$, where $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, and where the vectors e represent random disturbance. The challenge of *vector prediction* is to approximate the functions h and f , without prior knowledge of what these functions are, based on learning from a finite time-series of experience. This is a very well-known problem in statistics, but the rigorous methods now available are simply not powerful enough to replicate or explain the learning and prediction capabilities we see in the mammalian brain, where the sampling time is essentially $\frac{1}{4}$ of a second, where n is over a million and m is much larger than that. Thus to understand or to replicate “cognitive prediction” – a key capability of the brain -- we need to do much better. Part of the solution is to move forwards to addressing fixed grid prediction or graph prediction (where $x(t)$ and $y(t)$ are vector fields over a fixed regular grid or a graph, and symmetry principles are assumed), but it is also essential to exploit principles like “uninformative priors” (reasonable “open-minded” assumptions about $\Pr(h, f)$) and “relevance” (cost measures for errors in prediction and modeling). A host of ad hoc methods (some quite formal in structure) have become widely used in practical prediction work, in engineering work and in economic forecasting in order to overcome the limits of classical maximum likelihood methods, which essentially assume flat priors; however, to achieve a higher degree of rationality, capability and unification of methods – as required for “cognitive prediction” and for many practical applications – it is essential to revisit the foundations and derive more general learning methods on a more fundamental basis. This talk will review these various strands, stressing crucial areas where new research is needed and it is hoped that research funding may be augmented.

The term “prediction” here includes system identification, filtering and state space estimation (as in control theory), a large part of signal processing, a large part of machine learning, time-series identification and estimation, causal data mining, and general-purpose methods for pattern classification and recognition. Among others. A key part of the challenge is to develop a more unified approach, including courses which prepare students across this breadth of approaches.

“Cognitive Optimization and Prediction”, search on “COPN” at www.nsf.gov – and international follow-on?

$\Pr(A|B) = \Pr(B|A) \cdot \frac{\Pr(A)}{\Pr(B)}$
Prediction

Memory
 ...
Clustering

Optimization
 $J(t) = \text{Max} \langle J(t+1) + U \rangle$

$$\frac{\partial^+ z_n}{\partial z_i} = \frac{\partial z_n}{\partial z_i} + \sum_{j=i+1}^{n-1} \frac{\partial^+ z_n}{\partial z_j} \frac{\partial z_j}{\partial z_i}$$

Better prediction in the face of complexity would have many practical benefits, but the most important motivation, in my view, is what I show on this slide. This slide depicts one of the research areas which I run at NSF, called Cognitive Optimization and Prediction (COPN).

The real goal of COPN is to understand and reverse engineer the highest level of intelligence we see in the brain – the ability of the brain to learn to predict and understand its environment, and its ability to learn to do better and better in all kinds of complicated new challenges in decision and control. People have talked about trying to understand the higher level of brain intelligence for centuries and centuries, but now, thanks to great advances in neuroscience, in technology and in mathematical tools, the goal is finally in sight. We have a roadmap that could actually get us there within this century, and maybe even within two or three decades. In my view, this is the most important tangible challenge and opportunity for all of basic science in this century.

But first I have a confession to make. The brain I show on this slide is a human brain, but the opportunity before us now is to understand intelligence at the level of the mouse. Certainly we can develop levels of intelligence higher than that, in time, but the basic mammal brain is already enough of a challenge for now.

In recent years, we have made great progress on the “optimization” part of this. The field of approximate and adaptive dynamic programming (ADP) has made great strides, including some diversity of stability proofs for various designs, and some

important real-world applications. But the prediction part has not moved nearly as well. There are major unmet opportunities. That is why I am hoping for your help here.

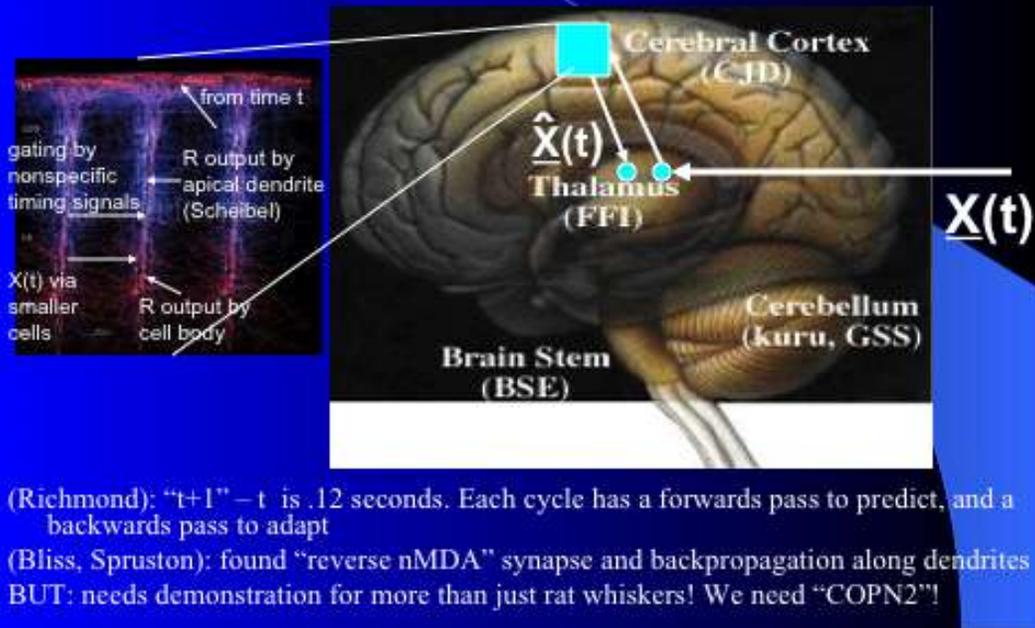
On the optimization side – some of you might be interested in the many theorems discussed on the web site of Prof. Frank Lewis, who runs the ARRI institute at the University of Texas at Arlington. (I also posted a long paper at arxiv.org in 1998, with some relevant results.) On the roadmap to the brain – see P.Werbos, *Intelligence in the Brain: A theory of how it works and how to build it*, *Neural Networks*, [Volume 22, Issue 3](#), April 2009, Pages 200-212. At www.werbos.com/Mind.htm, I have posted a diversity of papers ranging from the COPN level (aiming at the mouse level of intelligence and engineering applications) to thoughts about larger issues.

The progress in ADP all focuses on what some people call “anticipatory stochastic optimization.” But there are still some important challenges in “static optimization,” relevant to reverse engineering the brain. To understand this distinction – consider a game of chess. At each move, your decision could be broken down into two parts: (1) for each possible move, EVALUATE how good the resulting position would be for you; (2) search through the set of possible moves to find the one which leads to the best (maximum) evaluation score. ADP provides new methods for the first part of the problem, the hard part. But the second part also grows in difficulty and importance, as we scale up to brain-like complexity, and require that our algorithms take really full advantage of the massive parallelism of the brain and of new computer chips. The details go beyond the scope of this paper.

The right-hand slide depicts important areas where this new understanding of brain-like intelligence could have tangible benefits to humanity. The arrow to the earth represents the value of this new technology to challenges like sustainability of energy and the environment, issues discussed at very great length on my web page (e.g. www.werbos.com/energy.htm). The top arrow represents the value of this technology to hopes for the settlement of space by humans; improved control of jittery nonlinear systems turns out to be crucial to many new space technologies. The lower arrow depicts the idea that more a complete understanding of mouse brain intelligence may also help us understand the human mind better, and develop its full potential more than we do at present.

The equation on the lower left is the chain rule for ordered derivatives, which is widely used and often cited in this field. See <http://www.werbos.com/AD2004.pdf>, published in Martin Bucker, George Corliss, Paul Hovland, Uwe Naumann & Boyana Norris (eds), *Automatic Differentiation: Applications, Theory and Implementations*, Springer (LNCS), New York, 2005.

Ability to learn to “Predict Anything” Found in the Brain (Nicoletis, Chapin)



We all see that the brain takes action to control muscles and glands, but our knowledge of prediction in the brain has only recently become so concrete.

The right side of this slide depicts some recent experiments by Nicoletis and Chapin which, in my view, are the most important experiments in this century so far for the goals of COPN. They not only show that the brain possesses a kind of general ability to learn to predict; they show exactly where it is.

In one set of experiments, Nicoletis and Chapin began by locating cells in the thalamus which responds to some simple, controllable stimuli – tweaking the whiskers of a rat. Then they found some important nearby cells which do not receive the inputs from the whiskers, but actually predict the input cells ahead of time. When they told us about this at a workshop organized by Karl Pribram, I suggested a follow-on experiment: why not change the pattern of stimulus or lesion part of the circuit, so as to destroy the quality of prediction... and observe whether the thalamo-cortical system learns to restore its prediction? They did this, and it worked. They included this briefly in their follow-on paper in Science.

In most research on the cerebral cortex, people talk a lot about different areas which handle different senses. But we also know that the cortex possesses some kind of universal learning ability (called “the principle of mass action”) which allows one area to take over from another. I would claim that learning to predict is a major part of that universal learning ability, which leads to the more specific abilities. But in order to understand that learning ability, we need to understand how this kind of thing is possible.

This slide is discussed at greater length in P.Werbos, Intelligence in the Brain: A theory of how it works and how to build it, *Neural Networks*, [Volume 22, Issue 3](#), April 2009, Pages 200-212. (Please forgive the missing tilde and carat which belong over the upper and lower versions of “R”.) In large parts of the computational neuroscience field, it has become conventional to postulate “asynchronous models,” models without any kind of clock signals – but empirical neuroscience has shown that “clocks are everywhere” in the brain and are just as essential to the capabilities of the brain as they are to technology. To replicate this kind of prediction with neurons in a dish, I would guess that we would need to include four kinds of neuron in that dish – the input thalamic cells, the predictor thalamic cells, giant pyramid cells from the cortex, and timing cells from the nonspecific thalamus. Of course, more cells would be needed to get the most powerful and accurate prediction, but this would hopefully be an important start.

What the Brain Teaches Us About Prediction

- **One universal system can learn to “predict everything.”** No need for 125 different methods in 32 chapters. But “who pays for lunch”? How can it be possible?
- Can take full advantage of **massive parallel hardware** like CNN chips.
- **All predictions – including pattern recognition and memory – are in service to action.** What is true versus what is useful? It is always about “prediction of the future.”
- **Incredible complexity** – learns nonlinear dynamic relations among millions of variables, based on only 10 data frames per second (300 million per year).

This slide brings us back to the core challenge we need to address here, in mathematics and in technology. How can we design a universal system which can learn to “predict anything,” can take advantage of massively parallel computer hardware, **and – above all – handle the huge volume of inputs that the mammal brain can handle?**

Of course, it may actually be a parameterized family of universal systems, any one of which could learn what the others could learn, if given time to catch up.

Definition of (Offline) Vector Prediction Task

- Assume a time-series database of a vector $x \in \mathbb{R}^n$ and the existence of another time-series vector $r \in \mathbb{R}^m$, obeying the dynamics

$$x(t) = h(r(t), e_1(t))$$

$$r(t) = f(r(t-1), e_2(t))$$

where e_1 and e_2 are random vectors. Try to estimate h and f or $\Pr(h, f)$ as accurately as possible, so as to be able to predict future values of x or $\Pr(x)$ or known functions $U(x)$ as accurately as possible.

There are several ways to translate the larger challenge in the last slide into something more precise and mathematical. This is one of them. The two dynamical equations here would be very familiar to people in many fields. The most important challenge, to begin with, is to find a way to estimate the most likely values of the unknown functions f and h , based only on a time-series database of past values of x .

This is not yet a complete problem statement, for two reasons: (1) We cannot know what $\Pr(h, f | \text{database})$ is, in principle, without a specification of prior probabilities, as I will discuss; (2) for practical purposes, to understand the brain, it turns out to be important to consider the more general case where $x(t)$ may actually be field $x(t, i)$ or $x(t, z)$, where i is a node in a graph or lattice and where f and h are constrained to obey some symmetry requirements.

The challenge of brain-like vector prediction is to perform this task well in cases where n may be a million or so and m may be many billions (really much more), but this system only has millions of observations in its database. This is not the ultimate challenge of brain-like prediction, as I will discuss, but it already goes well beyond what today's tools offer us.

=====

The full challenge of brain-like prediction requires learning in situations where the observations appear and disappear one at a time; however, the offline version of the challenge is difficult enough, especially when computational effort is constrained. I do not ask for an exact calculation of the most likely f and h , but I would ask for some degree of approximation to the optimum with some kind of bounded loss, at least in comparison with all existing methods. When we assume the most general and powerful priors, the grid and graph prediction challenges are actually subsets in

a way of the vector prediction task, but with simpler, “direct” priors, the vector case is simpler.

Question to Census Statistical Advisory Council (1978): What Principles Most Important in Building Or Understanding Such a Prediction System?



All said: They do not exist. It is impossible.
I would never use such a machine
even if I had it for free in my own lab.

I have been worried about this complexity issue for a long time. In 1978, I had a unique opportunity to speak with people in the statistical advisory council of the Census Bureau, which works hard to assemble all the top statisticians of the nation. Of course, I will not name names, and I did not get to speak to all of the people in the room, but I can say that one of the people I spoke to is perhaps the founder of “information geometry.”

I asked the same question of all the people I spoke to. Roughly: “I have heard that someone has constructed a device which inputs a time-series of millions of variables, sampled at a rate of about ten times per second, and learns to make major inferences about the nonlinear relations and dynamics of these variables after x million observations have been accumulated. My question is – what kinds of principles would make it possible to build something like that, and what should I read to get a better idea of what it requires?” I was very disappointed when every one of them said there was nothing to read, because such a device is intrinsically impossible. I assured them there are many working models in operation, and that I had seen proof myself that the device can actually work, but they refused to believe me.

Of course, this is not a commentary on these statisticians. It is a commentary on the limitations of conventional assumptions used in the foundation of statistics.

Why It Was Seen As Impossible: 4 Schools of Thought in Statistics

- Probabilism (“We don’t do inference. We just prove stuff.”)
- Maximum Likelihood (Simplified from Jeffreys and Carnap)
$$\Pr(f, h | \text{Data}) \approx \Pr(\text{Data} | f, h)$$

- Bayesian (e.g. Raiffa)
 - Most popular: $\Pr(f, h | \text{Data}) = \Pr(\text{Data} | f, h) * \Pr(f, h) / \Pr(\text{Data})$
 - Sometimes minimize utility-based loss function
- Robust statistics (Tukey, Mosteller): try to get useful results without assuming model must be true for some value of weights W . (Also used by Raiffa, Werbos, and Vapnik.)

This slide conveys my impressions of why there was such a problem in those days – and even now!

There were four main schools of thought in statistics. The vast majority of statisticians belonged to one of the two schools above the line – probabilists or maximum likelihood people. Probabilists simply did not work on these kinds of problems of inference. Maximum likelihood statisticians mainly tried to estimate those functions f and h which would maximize **the likelihood function**, $\Pr(\text{database}|f,h)$ as shown here. That is simply not a powerful enough foundation for the kind of complexity that brains can deal with.

In my view, the **key to prediction under complexity** is to exploit the fundamental power of Bayesian induction and of robust statistics. The rest of this talk will discuss why and how.

In recent years, the word “Bayesian” has been used to label all kinds of things which do not even address the vector prediction challenge. Here I am not endorsing those things, or proposing any new speculation. I am simply saying that we need to use Bayes’ Law as a more active starting point, in order to develop a more powerful approach to learning, as complete as maximum likelihood analysis but more powerful. We cannot ignore the embarrassing term $\Pr(f, h)$, the prior probabilities term – the term which represents the probability that f and h are the true functions which drive the system we are observing, prior to our having any database available. We **also** need to unify this new approach with robust statistics, which I will try to get to.

Good maximum likelihood statisticians have of course known about Bayes’ Law for many years. Some have argued that we “should” assume that all models (f,g) are equally likely a priori, because this is “more natural and more scientific.” This is

called “flat priors.” In retrospect, this is similar to the arguments used by people who would represent neurons by linear models, on grounds that “it’s more scientific to use a linear model, because we can prove more theorems more easily for the linear case.” Both groups remind me of Einstein’s old dictum that we should use the simplest possible model which works – but no simpler than that.

Just to be complete, I should also note that I was well aware back then of a variety of “consistent estimators” described in fields like econometrics, as in the classic text of Johnston. It was such a joy, in undergraduate days in the 1960’s, to progress from there to maximum likelihood methods, which make it very straightforward to calculate the exact and optimal most likely f and g , with maximum possible efficiency both in computational cost and in taking advantage of limited data! More precisely, we could take any parameterized family of stochastic models $g(x,w)$, estimate the best values of the parameters or weights w , and compare the results with the best we could get from any other parameterized family of functions $g^*(x,w)$. The same rigorous methods could be used for essentially any function, including functions $g(x,w)$ representing mathematical models of neural networks. I continued to use such practical maximum likelihood methods in serious tasks in econometric modeling throughout the 1980s – but used my understanding of more advanced concepts to navigate around their limitations. For example, see P.Werbos, [Econometric Techniques: Theory Versus Practice](#), *Energy: The International Journal*, 15 (3/4), 1990, p.213-236.

Uninformative Priors: The Big Picture

- Philosophers studying human learning have known for centuries that we cannot explain human learning without “uninformative priors”
 - Reverend Occam: assume higher $\Pr(f,h)$ for “simpler models” f and h
 - Emmanuel Kant: the “apriori synthetic”
- Solomonoff/Werbos (60’s): if f and h are instructions to a Turing machine, assume $\Pr(f,h) = a \exp(-kC)$, where C , the complexity, is the number of symbols needed to express the Turing machine. This is **universal to all Turing machines**, to within some finite “learning time” to adapt from one Turing machine to another. In a way, this is the perfect best possible general foundation for a universal learning machine, but....
- How can we **approximate** its implications for f and h implemented, for example, as networks of neurons?
 - “reward” (higher \Pr assumed) for networks of **greater direct simplicity**
 - also “reward” the greater simplicity implied by **symmetry** (reflecting how Turing machines can reuse subroutines)
- How do we handle x and y being continuous? And our inability to integrate over all possible models? -- direct priors, nonlinear version of “empirical Bayes” (Efron)? Is brain limited to 3+1-D (Kant)?

This slide gives my view of what it takes to handle brain-like complexity, in vector prediction. I will give more detail later, but for now I would like to go through all of this carefully, to keep the picture together.

To begin with, philosophers have known for centuries that it is not possible to explain brain-like learning if we assume that all well-posed models are equally likely apriori. This is why the famous Reverend Occam invented his famous Razor centuries ago. Whenever two models fit the data of experience equally well, we attribute higher probability to the **simpler** model. $\Pr(f,h)$, the prior probability, must be greater for the simpler model, to fit or explain what we see in human learning.

More recently, Emmanuel Kant (Einstein’s favorite author) stated that learning or “practical reason” requires the foundation of an “apriori synthetic” assumption. That’s exactly what $\Pr(f,h)$ can provide. A badly constructed prior probability distribution $\Pr(f,h)$ could prevent learning, by virtually dictating what model (f,h) the system will believe in, regardless of data. But an “uninformative prior” could make it possible to cope with complexity while still being **open-minded** in some fundamental way.

In the 1960’s, both Solomonoff and I came up with the same insight about how to do this. The academic credit should go to Solomonoff, since I was not ready to delay dissemination of the idea just in order to get credit. Please forgive that I even mention the history.

The key idea was that a theory (f,g) could be represented as a set of instructions to a Turing machine. In that case, we can measure its complexity as C , the number of symbols, and set prior probabilities according to $\Pr=c*\exp(-kC)$, for some c and k . Any learning system based on one Turing machine could then catch up to a learning system based on any other Turing machine “in finite time,” insofar as the code which interprets one Turing machine to another is of finite length, providing a finite complexity penalty. (More precisely, $\Pr(f,h)$ in one machine is bounded above $k*$ times $\Pr(f,h)$ according to another, where k is $-kC'$, if C' is the length of the interpreter code.) Furthermore, it could catch up in finite time for any learning machine based on instructions in any finite state machine, because of the properties of universal Turing machines.

In some sense, the use of the Solomonoff/Werbos prior is **the** answer to problem of vector prediction in the face of complexity. Working with finite computer hardware, we cannot possibly do better (again, if we allow finite catching up time). There is actually a family of universal learning machines applied here, for each choice of universal Turing machine and each choice of c and k . (There are ways to do even better with k , but not so important to the main story here.)

But we face two practical problems of great importance to science:

- (1) Less important – could it be that the human brain is far less universal than this? For example, since mammal brains all evolved in a 3+1-dimensional universe, could our brains be specialized to that particular case?;
- (2) More important – how do we **approximate** this kind of universal learning, when we are restricted to using massively parallel computing hardware like neural networks, and we want to be practical about how we handle continuous as well as discrete variables?

The real challenge of brain-like vector prediction is mainly a matter of addressing the second question here, and also accounting for robustness.

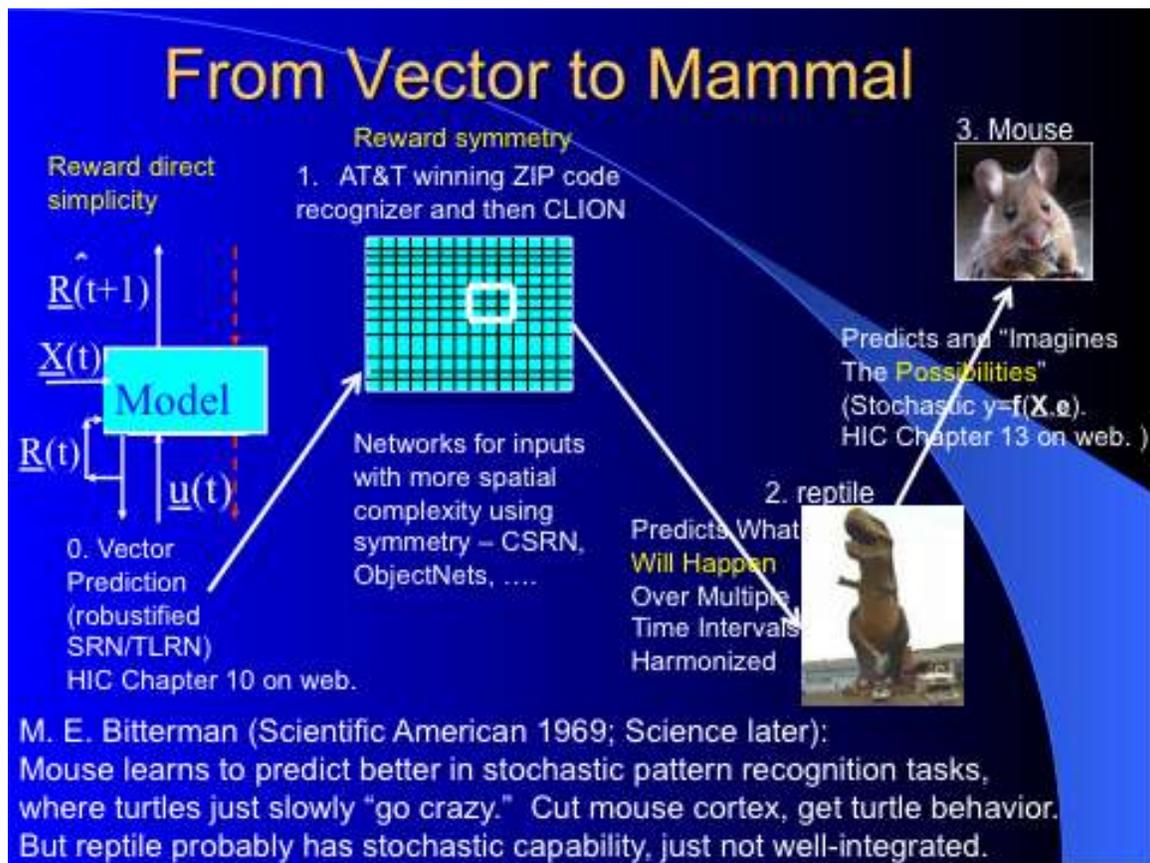
Until 1990, I thought I had a reasonably good way to address this question. (See the discussion of the first and second generation model of the brain, in my April 2009 paper cited above.) If our prediction models $(f$ and $g)$ must physically take the form of neural circuits, biological or artificial, why not consider the general class of such networks, with fast recurrent loops and also loops controlled by some kind of hard-wired clock? Why not measure complexity as some kind of function of the connection strengths, number of weights, and other obvious parameters of the network? Why not assume a **direct complexity prior**, $\Pr(f,g)=c*\exp(-kC)$, based on such a direct measure of complexity C ? When we insert this into our original estimation problem, we end up minimizing the usual log likelihood measure of maximum likelihood statistics, plus a penalty function. This still requires research to find the right penalty functions, and some adjustment for “syncretism” (as I will discuss), but that’s basically all. I hoped that all the rest of the complex structure and learning which we observe in the brain could be explained as the emergent result of this kind of learning.

So now: I hereby define the **direct vector prediction problem** as the challenge of estimating f and g , as shown in the previous slide, in the case where f and g are sampled on the basis of this kind of direct complexity prior, or something equally

direct. (For example, if C would be a Sobolev norm of the functions f and g.) This by itself is already an interesting and important problem, calling out for more research!

Towards the end of this slide, I refer to a much older example of direct vector prediction – the “empirical Bayes” method of Bard Efron, which reduces to a variety of ridge regression. This is an important precedent, which I discuss further below.

HOWEVER – by about 1990, I began to realize that this kind of direct vector prediction approach would still not be powerful enough for brain-like capability. The Turing machine has a kind of ability to define and re-use symbols, which makes it substantially more powerful in handling spatial complexity than direct vector prediction. After studying this and other related issues, I came up with a new theory of what is going on here, which leads to the roadmap for prediction in the next slide:



This is the prediction side of the roadmap to the mammal brain discussed in my April 2009 paper.

At the present time, the two really essential (parallel) directions for research are in direct vector prediction under complexity (symbolized by the flow chart on the left) and in better handling of spatial complexity (symbolized by the grid). To achieve brain-like capability in handling spatial complexity, we must build learning systems which discover the symmetries and invariances in the world by themselves,

and embed them into other architectures; for now, however, there is a lot of practical and intellectual benefit in starting from cases where we know the symmetries and the graphs already. I have funded some work aimed at helping us to make the further step to dinosaur and mouse levels of prediction, but the greater need today is for a firmer foundation in the first two steps. I will now discuss those two steps.

The Department of Defense and IBM have recently made claims about having already built a computer “at the mouse level.” They have had very important success in building computer hardware with the required level of complexity, and the ability to simulate classical models from computational neuroscience. However, in my view, these will not be able to replicate the capabilities I am discussing here, until we develop the required new mathematics, and put that into the hardware. Such a project might be called “greening the blue brain.”

Step-ladder of direct vector prediction challenges: static x-to-y

- Just assume $y=f(x,e)$, for a database of x and y , where e is random and y is exogenous, where f is one sample from $\Pr(f)=c \exp(-kC)$, and C is a Sobolev measure or the max of the length of the gradient of f , or such. Can we combine both theorems and simulations to move towards a universal learning system – a system which approaches the best possible performance in this case? And outperforms the many ad hoc methods now being used in “data mining” for this purpose?

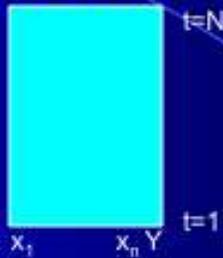
Since we are very far as yet from fully solving or approximating the general challenge of (offline) direct vector prediction, it will be useful to think of a graded staircase of easier related problems leading up to that level.

At the start of that staircase is the problem of static prediction, sometimes called supervised learning. There is a huge but relatively ad hoc literature on that problem. Some researchers just compare performance of selected real-world databases, which can provide excitement (and sometimes extra funding), but make it very hard to conclude anything definite. In my view, the overemphasis on ad hoc arguments and selected tests on data mining problems has been a major factor in delaying real progress toward brain-like capability in this area.

Prediction competitions and tests are still a very useful stream of research, but why not add competitions and tests focused on well-defined mathematical

systems, based on well-defined complexity priors? This would allow mutual support between theoretical mathematical analysis of the problem and empirical results, in parallel. Of course, it requires great care in the choice of priors, but that is part of the research. I will make some suggestions later.

Model-Based Versus Precedent-Based: Which Is Better?



- **Model-based:** Pick W to fit $Y=g(x,W)$ across examples t . Given a new $x(T)$, predict $Y(T)$ as $g(x(T),W)$. Exploit Barron's Theorem that smooth (low C) functions f are well approximated by simple MLP neural nets – though not by Taylor series.
- **Precedent-Based:** Find t whose $x(t)$ is closest to $x(T)$. Predict $Y(T)$ as $Y(t)$. Kernel is similar, weighted sum of near values.
- **Best is optimal hybrid, needed by brain.** “Syncretism” – chapter 3 of HIC.... Next 2 slides

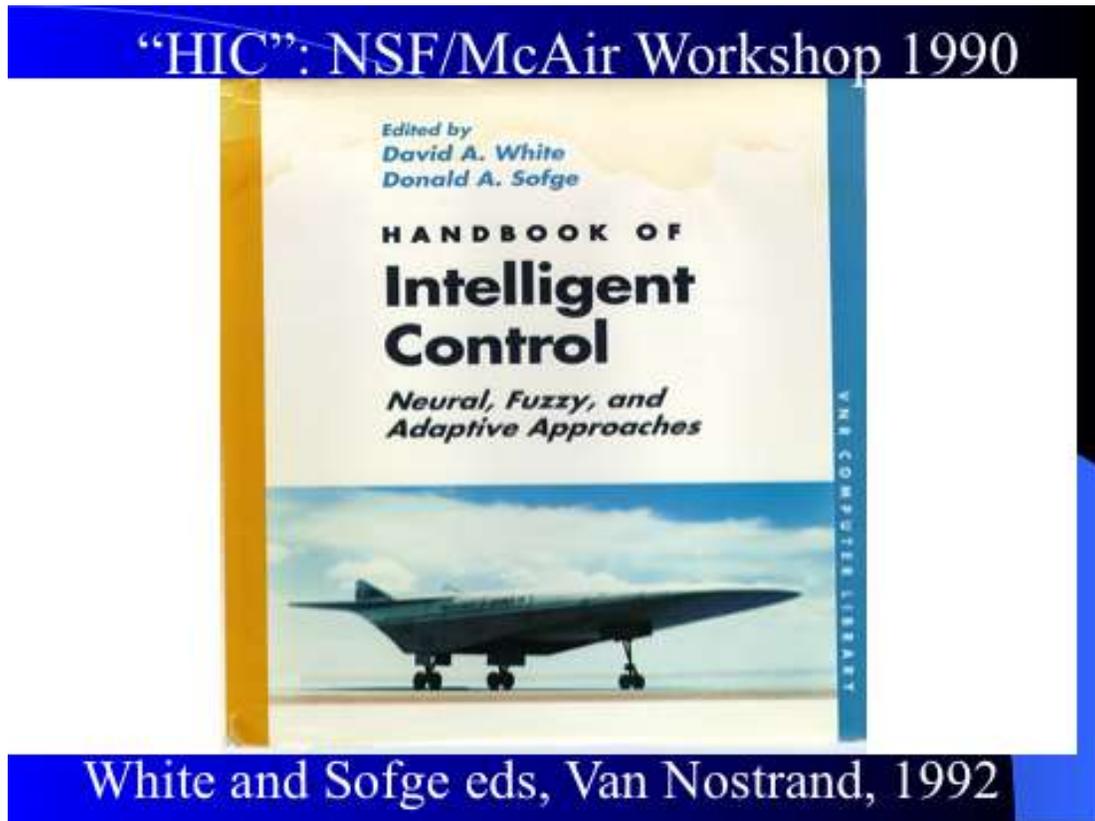
This slide illustrates why the “simple” problem of offline smooth static supervised learning in the face of brain-like complexity is still very far from being solved as yet.

One “obvious” approach to this task is the model-based approach, based on maximum likelihood thinking modified only by use of a penalty function. In order to explore the space of possible functions f , using real computer representations, we should use a family of universal nonlinear function approximators. If our priors are set to give us smooth functions f , then it should be good enough to use the Multilayer Perceptron (MLP) as our approximator. Of course, there are many other universal approximators possible, but Andrew Barron of Yale has proven that the required complexity of an MLP grows only in a polynomial way with the number of components in x (for a given degree of smoothness and required approximation accuracy), while linear basis function approximators such as Taylor series and Gaussian networks grow exponentially. In the slide above, model-based methods can be viewed as methods which move along the rows, from left to right, in predicting Y .

On the other hand, Efron's version of ridge regression gives us a strong hint that some kind of hybrid of model-based methods with memory-based methods

might be best, in the nonlinear case as well as the linear case. More precisely, ridge regression is often formulated as a kind of weighted sum of the usual nonlinear regression model-based system, with a covariance based system basically the same as some of the designs for associative memory used in neural networks. In the extreme, many practical forecasters used precedent-based forecasting, as described on the slide. There are many very complex kernel-based methods – more complex methods which go “up vertically from t to T ” -- which are rigorous in the sense that they use complex mathematics to define **what** they do, but not **why** they do it, in the fundamental terms I am looking for here.

My claim is that neither the pure model-based methods nor the pure kernel methods are anywhere near optimal. Neither can come even close to the capabilities of the brain in simple static smooth supervised learning, let alone anything more complex. I see an analogy between the competition between these two methods to the competition between extreme Republican and Democratic ideologists in the US Congress; real progress will require help from deeper thinkers who can get beyond it.



Before going further, I should note that everything I say here about direct vector prediction (including citations to Efron and others) is foreshadowed in chapters 3, 10 and 13 of the Handbook on Intelligent Control (HIC). Those chapters are all posted at <http://www.werbos.com/Mind.htm#mouse>.

“Syncretism” Design

Basic Idea:
$$\hat{Y}(t) = \tilde{f}(x(t)) + \sum_{\tau} K(x(t) - x(\tau))(Y(\tau) - \tilde{f}(x(\tau)))$$

Practical Implementation/Approximation:

- Associative Memory of Prototype $x(\tau), Y(\tau), Y(\tau) - f^*(x(\tau))$
- Update $Y(\tau) - f^*(x(\tau))$ on occasion as f^* is changed

In other words: Keep training f^* to match examples or prototypes in memory, especially high-error examples.

Predict $Y(t)$ by f^* plus adjustment for errors of f in nearby memory.

Closest so far: Principe kernel applied to model residuals;

Atkeson's memory-based learning.

Exactly fits Freud's description of ego versus id in neurodynamics.

This slide illustrates my proposal for a hybrid between memory-based and model-based prediction, first published in my paper in *General Systems Yearbook 1977*.

It is also discussed in chapter 3 of HIC.

The basic idea is quite simple, and I am embarrassed that nothing better at a fundamental level for general-purpose learning systems has shown up in this time. Here I am using f^* or f -tilde to represent the global neural network approximator, and K to represent a kind of kernel function. (I have also made some obvious and crude suggestions for how to train K .) This design is really aimed at the problem of real-time learning, as observations come in one by one, but it also can be applied in offline data mining. The neat thing is that it exactly fits a major part of Freud's theory of human learning, as I will show on the next slide.

It would be interesting to prove that this “dominates” over pure model-based and kernel-based methods for the smooth static supervised learning problem.

“Dominated” means sometimes a lot better, and never more than some limited penalty worse. For practical applications, it is interesting to consider what flavors of memory system would work best as part of such a design. There are many choices.

Example of Freud and Syncretism



- A Freudian story:
 - Nazi hurts child, a traumatic memory
 - For years, he is terrified when anyone in black shirt appears (precedent based prediction/expectation) – the kernel-based “id” is at work!
 - Later he learns about Nazis in subjective model of world (f), “ego”
 - After that learning, if he relives that memory (trains on memory), f error on the memory is low; memory loses power to cause irrational bias
- Key corollaries:
 - False hope from memory is as dangerous as false fear
 - We still need id when exploring new realms we can't yet reliably predict

Any truly serious model of intelligence in the brain should help us make contact with the rich history of human experience – including some of the fundamental insights of Sigmund Freud. Freud was sometimes distracted by some of the narrow issues which came up in his clinical practice, but he also made important efforts to extract fundamental principles.

This slide illustrates one of his most important fundamental ideas. In his view – our feelings about present life are based on a kind of interaction between “ego” and “id.” “Id” evaluates what we see based on simple associations between what we see now and what we saw in the past, a kind of associative memory. “Ego” provides more of a global understanding of how things work, based on a more rational, global understanding of cause-and-effect relations. Nervous breakdowns commonly occur when the ego is unable to make sense of things, and the id takes over – or else, traumatic past memories in the id suddenly rise to cause feelings and behavior which make no sense in terms of rational cause-and-effect relations. When this kind of id-based behavior becomes problematic, it can be reduced by persuading the patient to relive the problematic traumatic memories and reduce their “energy.”

The “syncretism” model or design for learning can explain what happens in Freud’s picture. More precisely – the f^* global model corresponds to this “ego.” The precedent-based component corresponds to id. The stored values of $Y(\tau) - f^*(x(\tau))$ correspond to the “energy” which can be reduced when observation τ is revisited with a new, more complete and mature version of f^* . These particular effects do not require a brain design more sophisticated than that of a mouse, but the human brain does retain all the basic structural features of the mouse brain.

Universal Vector Prediction System: Principles To be Explained

- For smooth functions $\underline{Y}=\underline{f}(\underline{X})$, Multilayer Perceptron (MLP) minimizes complexity and hence estimation error. Barron.
- For general functions $\underline{Y}=\underline{f}(\underline{X})$, add simultaneous recurrence ($\underline{y}[n+1]=F(\underline{y}[n],\underline{X})$) for Turing-like universality. SRN.
- For dynamic or time-series prediction, add time-lagged recurrence $\underline{Y}(t)=\underline{f}(\underline{Y}(t-1),\underline{X}(t))$ for universal “NARMAX” capability (TLRN)
- Unify maximum likelihood (least squares training) with precedent-based forecasting, “uninformative priors” (penalty functions), & weights for multiperiod prediction and salience – especially for real-time “incremental” learning.

⇒ Learning speed also an issue, harder with better prediction. Many useful tricks known. Kozma/Ilin/Werbos patent just a useful start.

This slide summarizes the research needs now ahead of us, in the area of direct vector prediction. The first bullet restates what I covered in the last few slides – work needed to address the task of static prediction in the case where f is taken from a probability distribution which favors smooth functions. The conjecture is that the MLP neural network, plus syncretism, offers an effective universal learning design for this case.

The next bullet describes the next step up in the stepladder of graded problems. Instead of considering the case where $\Pr(f)$ depends on the smoothness or Sobolev norm of f , we can consider the case where f is defined as the equilibrium of an iterative process based on an inner function F . We can then sample functions F based on **their** smoothness or Sobolev norm. This does require consideration of various ways to handle the issue of convergence in the iterative process itself; however, this step up is of very fundamental and inescapable importance, as we try to get closer to brain-like capability. This more general type of prior probability also gets us closer to the original, more universal Solomonoff/Werbos priors based on Turing machines, which allow recursive computations. Recursive computations with nonsmooth behavior are essential to many tasks in pattern recognition and control. The conjecture here is that an extension of the MLP, the Simultaneous Recurrent Network (discussed in chapter 3 of HIC), plus syncretism, gives us a universal learning design for this more general case.

Finally, the third bullet describes the most general case in this series. More precisely, it refers to an extension which can be applied to either of the previous two cases. The extension merely goes back to our earlier slide on vector prediction,

where prediction is formulated as a time-series task over a partially observed system. The same kind of sampling can be used to define $\Pr(f)$ or $\Pr(f,h)$. The conjecture here is that the Time-Lagged Recurrent Network (TLRN), described in Chapter 10 of HIC, provides the universal learning design for this case.

By analogy – statisticians like Box and Jenkins and E.J. Hannan have discussed how the vector ARMAX model is a kind of universal model for linear dynamic stochastic processes; in a similar way, the TLRN provides a kind of universal approximator for the NARMAX case, the nonlinear generalization of ARMAX.

The rest of the slide notes how there is additional research to be done related to statistical robustness and the actual speed of convergence in implementing these methods.

The TLRN model does not explicitly represent nonGaussian noise or noise affecting unobserved variables; however, an important paper by Feldkamp and Puskorius (posted with permission on my web site) suggests that the minimization of square error causes it to approximate optimal filtering as well as the best particle filtering methods, which are far more expensive to implement and less-brain-like. For a more explicit treatment of nonGaussian and unobserved noise, I developed a method, the Stochastic Encoder/Decoder/Predictor (SEDP), described in Chapter 13 of HIC.

Eduardo Sontag of Rutgers has proven many theorems important to these issues, though I would not always interpret their significance in the same way. For example, I am not so sanguine about the potential of rational approximators in this context, where a large number of variables are present and so on. On the other hand, he has proven universal approximation results for dynamic systems and recurrent networks of enormous importance. As a possible extension, I would conjecture that the SEDP (implemented with MLP or SRN component networks) is a kind of universal approximator for nonlinear stochastic processes. I would conjecture that SEDP (extended by inclusion of discrete variables and more statistical robustness) is important to achieving “mouse brain” intelligence, but not to direct vector prediction or to direct vector and network prediction.

In the SRN case – Ganesh Venayagamoorthy’s group has recently shown in simulations how SRNs can learn data generated by MLPs, in general, but not vice-versa. See Cleaver R, Venayagamoorthy GK, “Learning Functions Nonlinear with MLPs and SRNs Trained with DEPSO”, Proc. International Joint Conference on Neural Networks, Atlanta, GA, USA, June 14-19, IEEE, 2009. Such simulation results could almost certainly be backed up by more general theorems. The Simultaneous Recurrent network should not be confused with the “Simple Recurrent Network” published later by psychologists.

Winner of IJCNN07 Forecasting Contest: Ford 1998: "All Ford Cars Will Have TLRNs by 2001, for on-board Diagnostics"



- How can one neural network predict and diagnose all Ford engines, without retraining, even as they change over time? TLRN: adaptive prediction even without learning! ICNN05: "A neural network which can predict anything."
- IJCNN07, Prokhorov: TLRN prediction and control can improve mpg of Prius hybrid by 15% "at zero cost"!

This is one of a series of examples of TLRNs which have proven to have great value in very difficult real-world prediction challenges. At one time, two Divisions of Ford Research – one under Lee Feldkamp and one under Ken Marko – were far ahead of the rest of the world in addressing these kinds of challenges. For example, TLRNs were used to perform diagnostic estimation and prediction required by clean air laws, impossible with all the older methods that anyone had ever tried on this problem, within the cost constraints. At the International Joint Conference on Neural Networks in Orlando in 2007, industry members of the alternate energy task force all agreed that better understanding of how to use TLRNs was what they needed most in hiring new students; this task force included some people (like Prokhorov) who had received promotions by moving from the Ford group to elsewhere, and some from other major companies. The time-lagged recurrence has many practical benefits.

Ford has also developed a general-purpose computer package for training TLRNs reliably over large databases. In the general time-series forecasting competition in Orlando in 2007, organized by Sven Crone, student groups from traditional statistics and computer science generally outperformed those using simpler types of neural networks – but one of the folks from Ford using their standard package got to first place with relatively little effort.

Neural Network in Commercial Power Grid Hardware



- First deployment of deployment of recurrent neural network in the field in a commercial electric power grid. (Improved prediction to allow unprecedented monitoring and control of harmonics.) Harley, Georgia Tech.

This is another example, a TLRN in use in a commercial electric power substation.



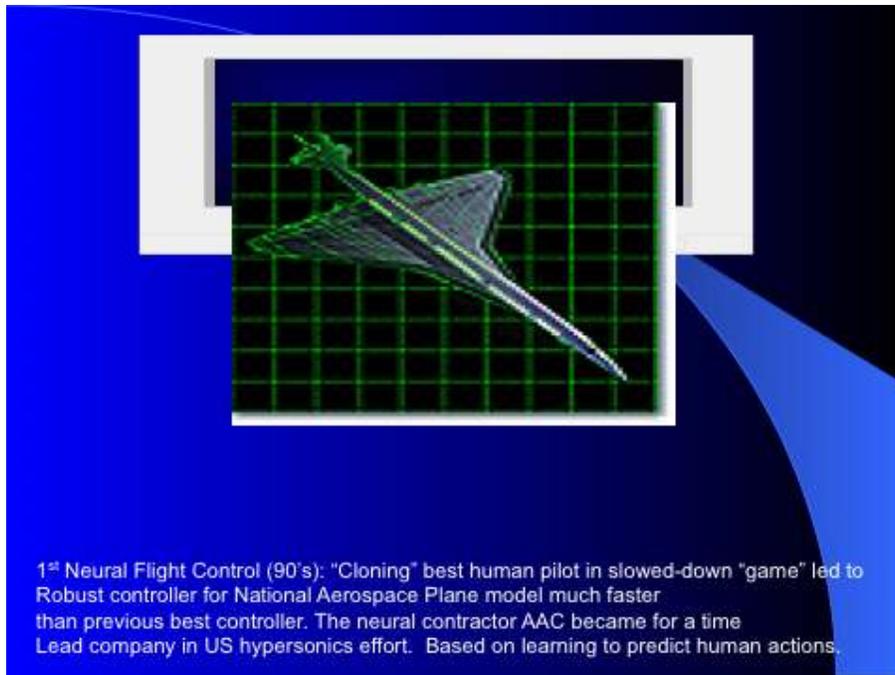
Robot first copies human by learning to predict time-series of human



Schaal, Atkeson
NSF ITR project

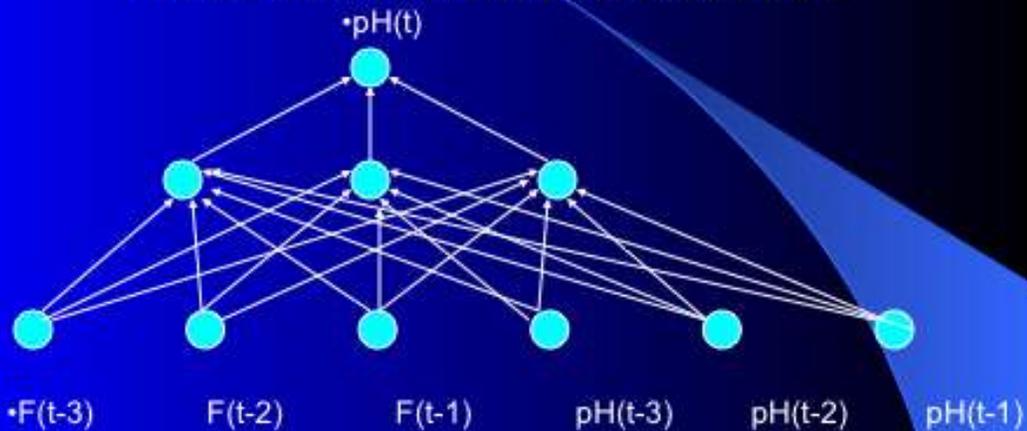
Learning allowed robot to quickly learn to imitate human, and then improve agile movements (tennis strokes). **Learning** many agile movements quickly will be crucial to enabling >80% robotic assembly in space.

This example shows how learning-to-predict can be crucial to important challenges in decision and control. One can build an agile robot by first learning to predict what a human would do, and then using that as a starting point for ADP methods which then learn to do better. Finding a good starting point is extremely important aspect of learning, not only for robots but for humans.



This slide depicts an earlier example of learning-to-predict which still is amusing and instructive. Leaders of the National Aerospace Program (NASP) told me and some people I fund about how they spend a long time, and millions of dollars, to buy a controller able to avoid crashes in simulated tests of a Mach 26 aircraft they were trying to develop. People at Accurate Automation (AAC) in Chattanooga, Tennessee, simply encoded this control challenge as a video game, run at speeds slow enough that a human could try it. They distributed it for free to many people, who enjoyed the challenge. Within just a few weeks, a couple of the testers (one an astronaut) got a perfect score in the simulation, no matter what challenges were thrown to them. Inside the box, the computer constructed a database of how these humans responded to the range of input conditions. Then AAC used a simple neural network (essentially a stripped-down TLRN) to learn to predict the human, and showed how they had met the NASP standards very quickly at very low cost. This was the start of a very complex and very serious story which continues to this data, with many branches.

Myth 1: Training Multilayer Perceptrons (MLP) is not black magic, is not an alternative to statistics



- Any MLP represents a function $\underline{Y} = \underline{f}(\underline{X}, \underline{W})$, \underline{X} the inputs, \underline{W} the weights.
- Minimizing the mean square value of (actual $\underline{Y} - \underline{f}(\underline{X}, \underline{W})$) over \underline{W} is nonlinear regression. All the usual error and significance and standard error statistics apply. It's just a more general choice of \underline{f} than usual (able to approximate any nonlinear smooth function efficiently) and it comes with faster more reliable convergence. Standard errors are less with more data and fewer weights.

I did not spend any time on the slide above and the slide below at the Erdos conference. They are taken from another talk given to people struggling with the challenges of practical data mining. They illustrate how the fashions in data mining today are beset by many myths and misconceptions and irrational thinking.

Myth #2: "No-Free Lunch" Is Not a Theorem

- An understandable reaction to ad hoc "A is better than B" studies and the old "pick a chapter" psychology
- But: given two families of models or topologies, $\underline{g}(W_1)$ and $\underline{f}(W_2)$, if every model in \underline{g} is close to a model in \underline{f} but not vice-versa, then \underline{f} is more powerful. "Almost-free lunch."
- Given enough data or given the right priors (favoring \underline{g} -like points in \underline{f}), \underline{f} should always do much better than \underline{g} or almost as well
- Examples:
 - ARMA beats AR: $x(t) + bx(t-1) = e(t) + ce(t-1)$, $c \neq 0$
 - (ARMA fits partially observed or noisy underlying AR.)
 - TLRN beats ARMA: $x(t) = e(t) + f(x(t-1), R(t-1))$
- BehavHeuristics airline seat forecasting example
- Most powerful if \underline{f} is most universal approximator, fewer parameters. Neural vs. translog, SRN versus MLP.

“Bayes” versus “Vapnik”: today’s debate

- Theorem: $\Pr(A|B) = \Pr(B|A) * \Pr(A) / \Pr(B)$
- Platonic Bayes:
 - Predict by using stochastic model $\Pr(\underline{x}(t)|\text{past})$
 - Find model with highest probability of being true: $\Pr(\text{Model}_w | \text{database}) = \Pr(\text{database} | \text{Model}_w) * \Pr(\text{Model}_w) / \Pr(\text{database})$
 - Neural $\underline{x}(t+1) = \underline{f}(\underline{x}(t), \dots, W) + \underline{e}(t)$ is just another stochastic model, with full NL regression statistics
 - Many variations; e.g. “Box-Jenkins” ARMA methods
 - “anything else is Las Vegas numerology”
- Vapnik says NO. “New” philosophy: if you want \$, not truth, pick Model_w which would have maximized \$ in the past (database)

This slide introduces a debate which is far more serious for our purposes here. It introduces the topic of statistical robustness, which is also essential to achieving brain-like capability in vector prediction.

In recent years, the work of Vapnik has become extremely popular for many people doing prediction. His texts and papers are highly mathematical – but they take a totally different starting point from the Bayesian approach I have described so far.

In Vapnik’s view – if we are training models in order to help us make money, we should not pick the model which has the highest probability of truth in past history. Instead, we should pick whatever model would have made us the most money if we had used it in the past. This is the kind of foundation which underlies his analysis which, again, is too complex to review in detail here.

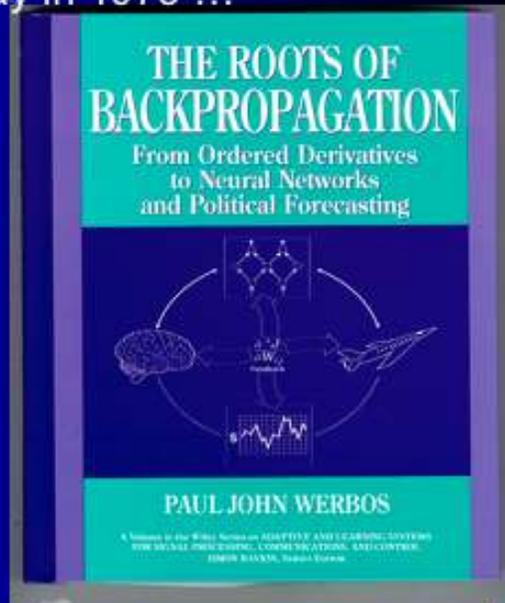
In fact, long before Vapnik, many statisticians (including Bayesians like Raiffa) recognized that the simplified models we build on computers – even if they contain billions of variables like the brain – are only an approximation at best of the far more complicated world they are trying to predict. Conceptually, it is easy enough to say that we want to pick whatever model leads to maximizing the expected value of our utility function when we use it, but it is very unclear that Vapnik’s heuristics or any other heuristics now in use do a good job of approximating that ideal. A major challenge to all of us here and now is to find better approximations, as principled as possible, unifying as much as possible of the insights available today. I will now give a few examples.

But Platonic Bayes fails very badly in some ways,
as I learned the hard way in 1973 ...

Vector ARMA (f) had twice
the prediction error
of simple extrapolator (g), on
100-year political data and
simulated dirty datasets

"Vapnik" style
"pure robust method"

BRAINS absolutely
require multiperiod
robustness beyond what
Platonic Bayes offers



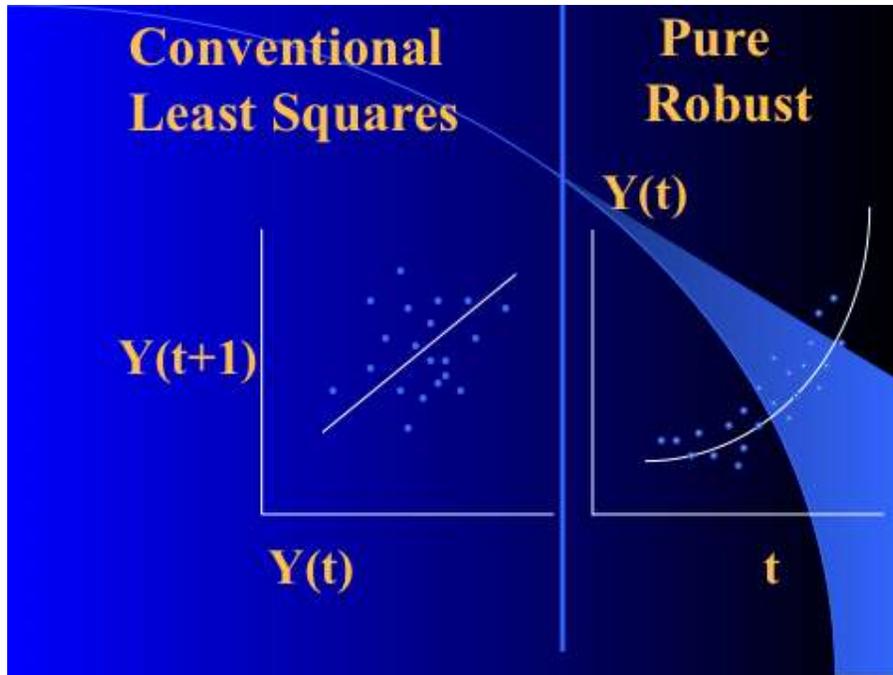
1974 Harvard PhD in subject of statistics, Mosteller on committee (Dempster help)

When I started my PhD work at Harvard long ago, I began by taking a Bayesian view, the kind of Bayesian view I have described so far. My thesis advisor, Prof. Karl Deutsch, had previously used regression analysis to estimate a simple model he had developed for cultural and political fluctuations across nations over time. But regression had failed badly. I felt that a vector ARMA model should do much better, because it is a more universal model and because (unlike simple regression) it accounts for observation error. The chain rule for ordered derivatives made it possible to estimate vector ARMA models much faster than in the past. All of that worked.

However, to really test the model, I also compared it against a simple "devil's advocate" model, a kind of simple extrapolation, implicitly assuming a model much simpler than the ARMA model *or* the regression model. The extrapolation also worked much better on simulated data, in cases where there was dirty noise or such. Professor Mosteller, a pioneer in robust statistics, taught me about that approach, and encouraged me to do these simulations.

Because brains must also survive in a world of dirty noise and complexity, I immediately tried to understand the underlying principles, and adjust my approach accordingly. Of course, this was all long before I heard anything about Vapnik.

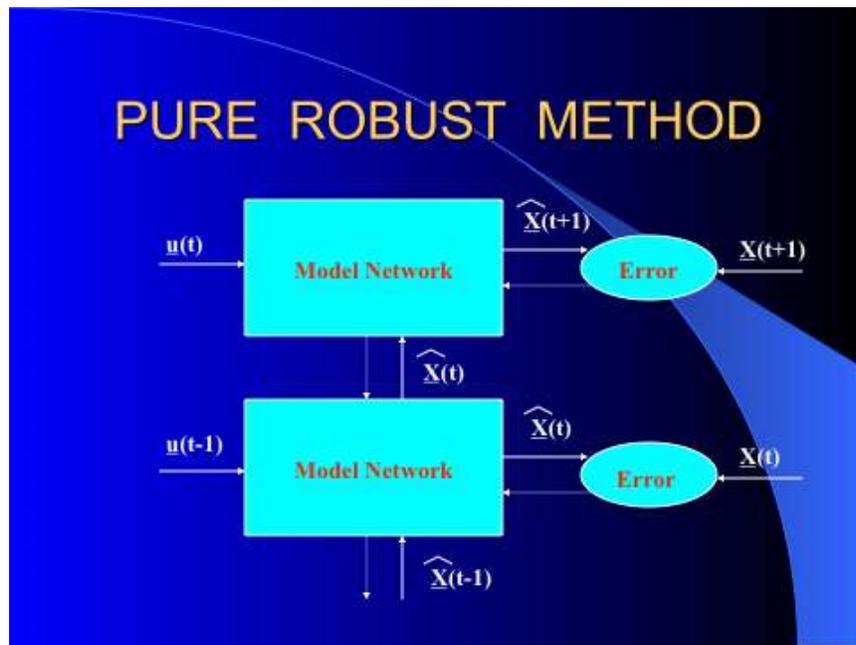
The thesis was reprinted in its entirety as part of the book shown here from Wiley.



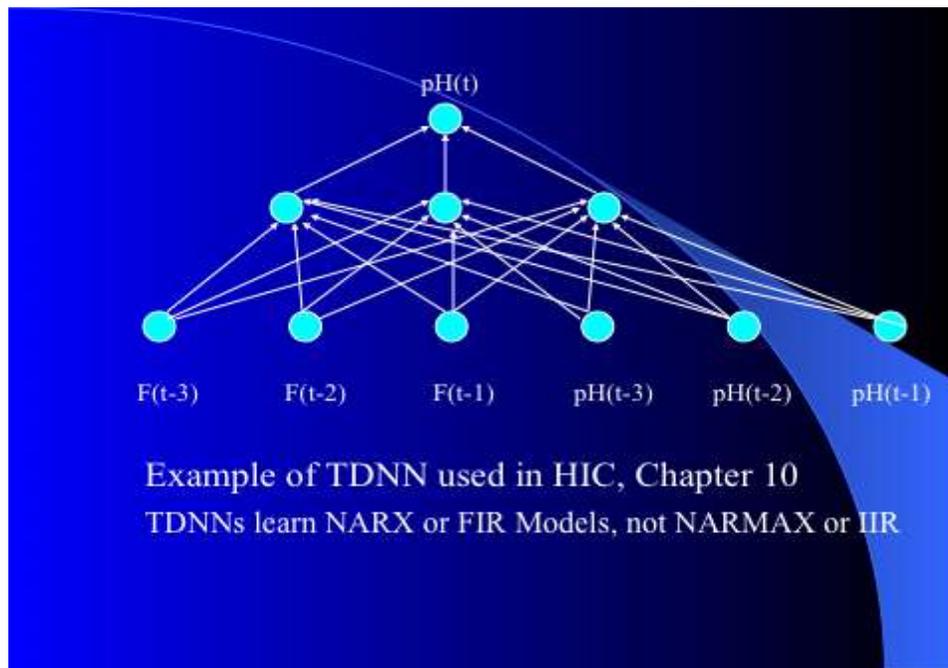
This slide illustrates my interpretation of what I saw in this data. Conventional least squares methods – maximum likelihood or Bayesian – basically end up minimizing a measure of error in predicting the vector $Y(t+1)$ as a function of data on $Y(t)$. But the simple extrapolation which I did is a special case of a more general method, which I called the pure robust method. (Since then, others have called it “multiperiod error minimization.”). The idea is to match the entire trajectory of data over time as closely as possible to a multiperiod forecast based on the initial data, $Y(1)$, without the use of any other data (except exogenous variables).

When I discussed this with students, they would immediately think of this in a Vapnik kind of way. “Of course, if your goal is to use a model in multiperiod forecasting, that’s what you should train it for.” But in fact, that is an incorrect and naïve way to look at this situation, as I discussed even in my PhD thesis. For example, notice that the fit between predictions and data in the pure robust case is far more sensitive to the model parameters than in the conventional case; high derivatives of error near the minimum of error implies smaller standard errors in estimating the parameters, which implies more accuracy in all applications. When a trajectory can be fit to the data well enough to make this true, the justification for the pure robust method is stronger than the Vapnik approach would seem to suggest.

Comment: if the AR or ARMA models were exactly true, it would not be possible to fit a trajectory so well. Later, when I worked for the Energy Information Administration, I saw many examples where backcast tests showed that models often do stay “on trajectory” in the real world, far more than one would predict from playing out the original regression analysis as a stochastic model.

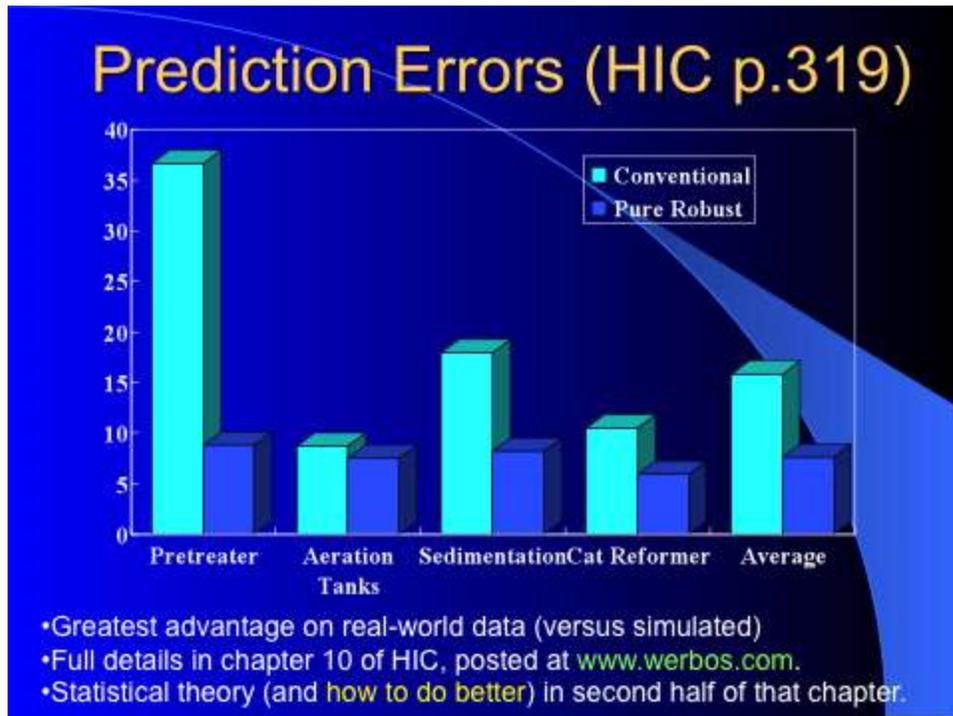


Of course, it is not entirely trivial to implement the pure robust method for a general class of nonlinear multivariate prediction models. This slide, taken from chapter 10 of the Handbook of Intelligent Control, illustrates how to do it in a practical way, exploiting the chain rule for ordered derivatives.



In order to get closer to brain-like universal capability in vector prediction, I spent years encouraging people to move up from TLRNs trained in the conventional way to TLRNs trained by the pure robust method. This simply did not work in the automotive sector, where TLRNs worked well enough and people saw little reason to put in the effort to try to do a little better. However, in the world of chemical engineering – a leading control expert, Prof. Tom McAvoy, had published papers

expressing his frustration with the limited prediction accuracy of a simple neural network trained in the conventional way. We established a collaboration, which demonstrated big reductions in prediction error, and extended into many applications in the chemical sector. Chapter 10 of the Handbook contains a lengthy description of the practical details, before the section on more fundamental, general issues in prediction.



This slide shows the error reductions we got in the initial work, described in HIC. The graduate student, Ted Su, went on to develop many more applications (and better ways to initialize the estimation), in his PhD thesis under McAvoy. He then moved to Honeywell.

But Pure Robust (“Vapnik”) Can Fail Badly Too: Phase Drift

$$\mathbf{R}(t+1) = \mathbf{R}(t) + \mathbf{w} + \mathbf{e}_p(t)$$

$$\mathbf{X}(t) = \sin \mathbf{R}(t) + \mathbf{e}_m(t)$$

TINY

A unified method cut GNP errors in half on Latin American data, versus maximum likelihood and pure robust both (SMC 78, econometric).

Nevertheless, the pure robust method does not offer the kind of universal prediction ability we see in the brain. Sometimes it is possible for a fixed, deterministic model to generate a trajectory of forecasts which track the observed data. Sometimes it is not. The slide above gives an example where the pure robust method could “almost” give us much more accuracy than conventional methods (and where it can really do so for short enough data series), but it goes off track and becomes very inferior as times become longer. To develop a brain-like universal predictor, we need to find some kind of universal hybrid between these two approaches, based on a unification of the underlying principles of both,

Best Hybrid Known So Far

- Cut error 50% in predicting GNP in Latin America versus the best of ML and Pure Robust
- (see page 327, Chapter 10, HIC)

$$\text{general example : } e(t) = \frac{1}{2} \sum_{\tau} (y_i(t) - \hat{y}_i(t))^2$$

$$\text{ML version : } \hat{y}_i = \tilde{f}_i(y(t-1), x(t))$$

$$\text{Pure Robust version : } \hat{y}_i = \tilde{f}_i(\hat{y}(t-1), x(t))$$

Compromise Method Version (1977 version) :

$$\hat{y}_i = \tilde{f}_i(\tilde{y}(t-1), x(t))$$

$$\tilde{y}_i(t) = (1 - w_i) \hat{y}_i(t) + w_i y_i(t)$$

$$\text{Minimize } \sum_{\tau} \frac{(y_i(t) - \hat{y}_i(t))^2}{\sigma_{y_i}^2 (1 - |w|)}$$

This slide describes a general hybrid method which I developed and tested in 1977. (HIC Chapter 10 develops and proposes, but does not test, a new version.) To test it, I used a database of economies and conflict in Latin America, a database which contains more underlying noise and unpredictability than a typical chemical plant.

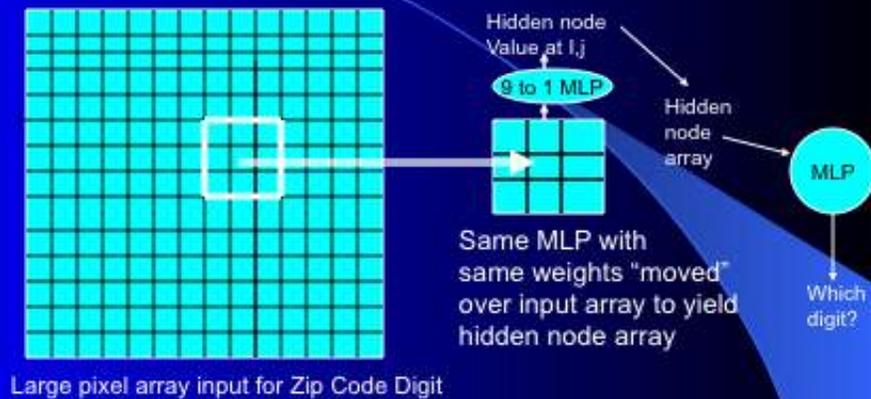
To my great disappointment, this is still the best and most principled method I know of to this day for general-purpose learning to predict. It has not been widely used in the neural network field, in part because the later part of chapter 10 may have been intimidating to many engineers, and in part because the combination of conventional TLRNs and the pure robust method usually seems good enough. To achieve brain-like prediction capabilities, it is extremely important that we learn to do better, and also learn to make better, fuller use of what we already have.

“Vapnik” approach is not new even in the static case

- Utilitarian Bayes: google “Raiffa Bayesian”: pick model and weights W so as to minimize a loss function L .
- Example of the issue: to weight or not weight your regression (in actual DOE/EIA model and conflict model):
Energy(state,year)=a*income(state,year)+e(year) (1)
(energy(state,year)/income(state,year)=a+e(year) (2)
If big states different, equation (1) is more consistent
If big states few, (2) has more information, less random error
Platonic approach: use F tests to see which is more true, but..
NonBayesian methods in econometrics for consistency under more general conditions

This slide gives one more example of a practical prediction challenge which raises fundamental issues of statistical robustness. Value-based weights on error in predicting different variables would also be part of a brain-like prediction system (though they become possible only inside of a system like an ADP learning machine which contains measures of value.).

Moving Window Net: Clue Re Complexity



- Best ZIP Code Digit Recognizer Used “Moving Window” or “conformal” MLP! (Guyon, LeCun, AT&T story, earlier...)
- Exploiting symmetry of Euclidean translation crucial to reducing number of weights, making large input array learnable, outcomes.

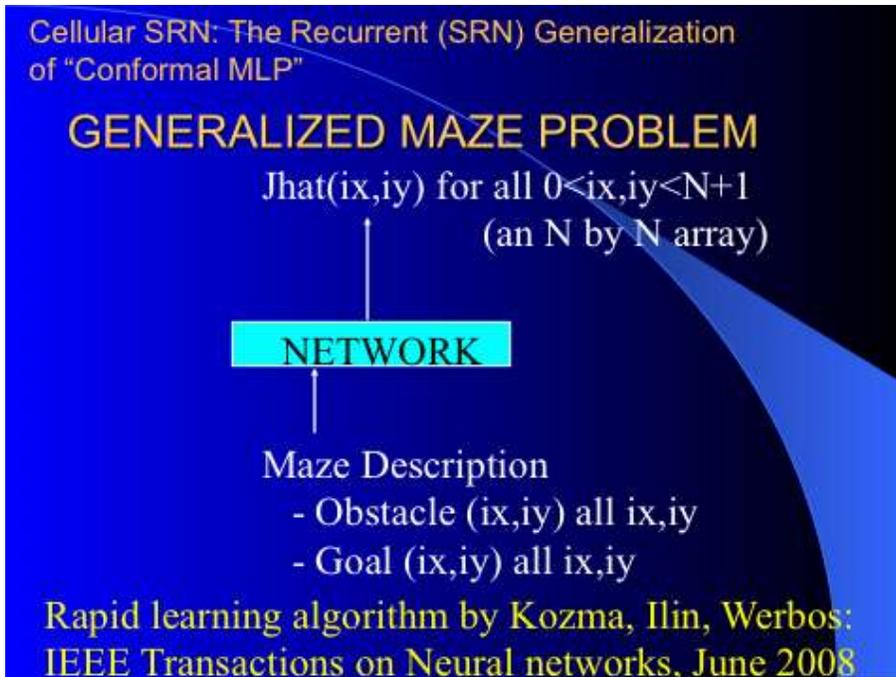
On this slide, I finally move from the challenges of direct vector prediction to the “next step up” – the full challenge of prediction in the face of spatial complexity.

This slide illustrates one of the two nagging issues which persuaded me, in the early 1990's, that one cannot understand or replicate the prediction abilities of the brain Within the confines of direct vector prediction. (The other was a paper by Albus and Meystel, cited in my April 2009 paper in Neural Networks.)

This slide depicts one of the great early triumphs of artificial neural networks. A group at the US Postal Service ran a set of parallel competitions, looking for the best possible recognizer of hand-written zip code digits. Their challenge database was of course widely available. According to Jay Lee of that group, AT&T and another group using essentially the same approach) were the clear winners, performing far better than many long-established groups which had worked hard for decades to develop understanding specialized the character recognition domain. This group did better by simply training a standard MLP to predict the classification from the pixel data, with conventional training based on "backpropagation" (the chain rule for ordered derivatives). But in order to get these good results, they had to assume and exploit translational symmetry in order to reduce the number of free parameters in the network. This is easy to understand in statistical terms, but it raises important questions: can we exploit symmetry effectively in a more general setting, in training general-purpose prediction systems, even when this kind of simple Euclidean symmetry does not apply? Just how important is this kind of capability? How would it change the design of brain machinery or of brain-like machinery?

This year, Yann LeCun (now at New York University) and Andrew Ng of Stanford have demonstrated a further dramatic step forward, under funding from COPN. Still using backpropagation, but with a different topology, LeCun has shown that he can get 2/3 accuracy in object recognition, rather than the 1/3 achieved by the best true learning systems in the past, on a very difficult challenge database posted at CalTech. This puts them slightly ahead of the best classical systems using hard-wired features developed over decades based on deep domain expertise. The number of pixels – the complexity – is far greater in this new challenge. Their architecture has ways of extracting discrete variables, which are complementary in my view to the continuous strengths of SEDP. The network design used on object recognition has also worked as well with little essential change in phoneme recognition. Certainly this is one important source for future research in this area.

LeCun called the earlier variation of MLPs a "conformal network." The impressive success of this work under COPN has led to larger scale projects at DARPA extending the new designs to other domains.

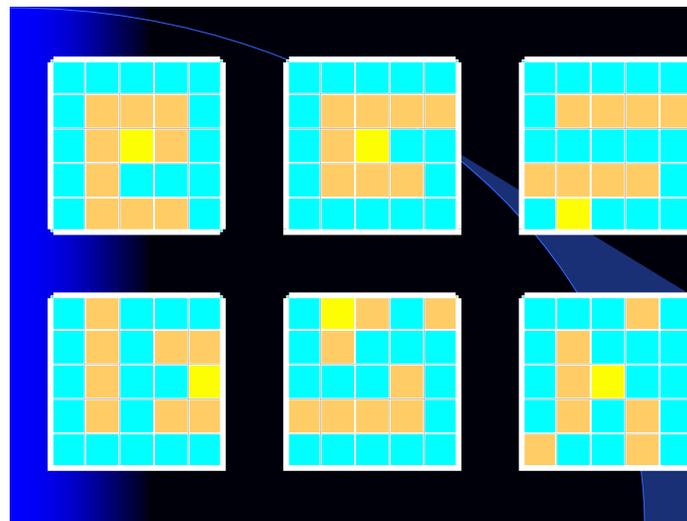
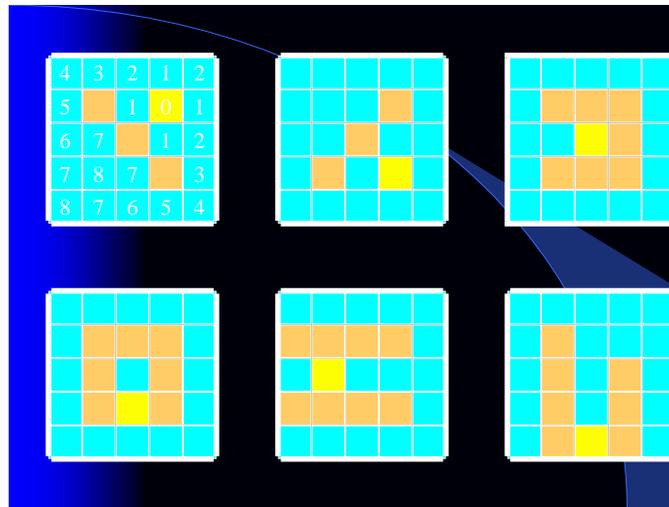


This slide illustrates a related challenging prediction task which I first worked on in 1998 (with Pang) and later with Robert Kozma and Roman Ilin (ongoing).

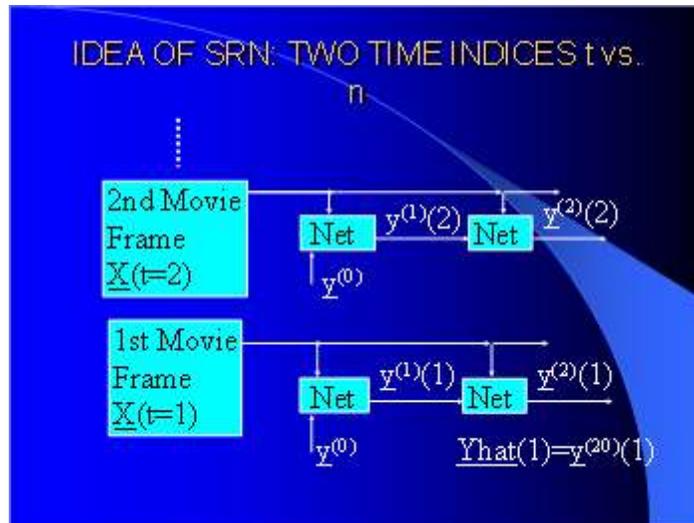
In this task, the input "vector" is actually a set of two arrays of binary numbers, which tell us where the obstacles are and where the goal is in an N by N maze. The desired output "vector" is an array of integers, which, on each square of the maze, report the shortest distance from that square to the maze. The challenge is to learn the mapping from the input to the output, based on a small set of sample mazes. This mapping is basically too complex (nonsmooth) to be learned by a simple convolutional MLP, or by an SRN. However, it can be learned by a "cellular SRN" (CSRN). The CRSN is basically a unified hybrid of SRN and convolutional MLP; in other words, it imposes translational symmetry on an SRN.

Though Pang and I showed this by examples, we did so at a price. Even with a reasonably tuned use of gradients, we needed tens of thousands of iterations to reach convergence. This is an example of the tendency of error minimization to be more difficult precisely for those models which have the smallest standard errors, and thus the error surfaces of highest curvature. But in 2008, Kozma, Ilin and myself showed how learning could be much faster on this problem by using a different procedure for adapting parameter estimates, still using the chain rule for ordered derivatives in order to calculate the required gradients efficiently.

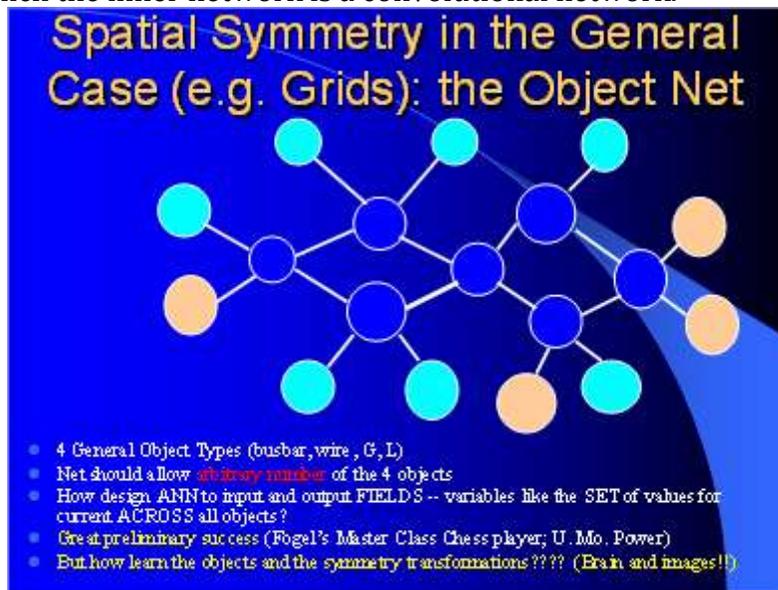
CSRNs, the chain rule for ordered derivatives and the new learning procedure are all well-suited for direct implementation on massively parallel chips. We expect that even faster methods for convergence can be found.



The two slides above illustrate the final project I did with Pang. We trained a CSRN to perform the prediction task for the six easy mazes shown in the top slide. (Six is enough, when we assume translational symmetry.) On the maze on the upper left, the \hat{J} scores for each square are also typed onto the maze. Then, after training, the CSRN was tested on the six hard mazes below. The more that training was done on the six easy mazes, the better the performance was on the six hard mazes.



This slide really just depicts the idea of an SRN, discussed previously. For a universal predictor of streaming video, coming into a rectangular image in a digital camera, one would use a cellular SRN, augmented by time-lagged recurrence to account for relations between one input frame and the next. The cellular SRN is basically just an SRN in which the inner network is a convolutional network.



But what happens in the brain, where simple translational symmetry does not obtain? Since 1998, I have argued that the brain relies heavily on a kind of nonEuclidean symmetry, which I call "object symmetry." This simply means that the brain resolves its understanding of external reality at any time, its "world model," into something like a graph. The set of "objects" is learned by SEDP or an extension of SEDP (as discussed previously), and then this set of objects is used in prediction of very complex scenes. (See my April 2009 paper in Neural Networks for more details.) Even before we begin to implement the full model, neural networks which implement this kind of prediction on a graph, with symmetry across objects of the same type, has tremendous practical applications. As an example, one could use it to learn how to predict behavior of entire electric power grids – a very complex system

– so long as one has data labelled by the usual types of objects which can be found in a power grid. The basic CSRN, or the CSRN with edges and corners behaving differently from other cells on the grid, are a special case of Object Net.

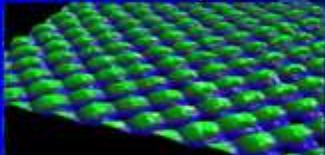
**David Fogel (Proc IEEE 2004):
World's First System which LEARNED
Master-Class Performance in Chess**



- Evolutionary computing (EC) to train a game-player worked for tic-tac-toe, but not checkers
- EC to train a multilayer perceptron (MLP) to serve as a CRITIC (an ADP value function) was enough to beat checkers but not chess
- EC to train a feedforward Object Net as a Critic was enough to beat chess
- Prediction: A full (recurrent) ObjectNet Critic can get to master class in Go. Will Wunsch get there first?

Not counting CSRNs, there have only been two real implementations of ObjectNets so far, in part because the training requires new algorithms. One is an application to wide area control in simulation for electric power. The other is shown here. Both are based on feedforward Object Nets, but even so they both led to breakthrough performance. One of the two was the first system ever to learn how reach master-class performance in chess, without being given prior information and rules and features (Iloike Deep Blue). A similar design could reach master class in checkers, when an MLP was used for position evaluation; however, only ObjectNets could do position evaluation good enough to allow learning of master class performance on chess. I conjecture that a true, recurrent Object Net would be needed to do the same for Go.

Key technology challenge for CLION: how can we use learning to get best performance from new chips like CNN, with thousands of processors per chip, across a huge segment of the market for computation?



Key enablers:

- New chips are suitable for nonlinear function approximators like CSRN, ObjectNets which can handle more complexity than traditional Taylor series, neural nets, lookup tables, etc.
- Kozm a/lin/Wverbos paper shows how they can be trained
- Neural net research shows general-purpose ways to use them

The Center for Large-Scale Integrated Optimization and Networks at the FedEx Institute of Technology in the University of Memphis has a deep commitment to the goals of COPN, and also to the broader technology goal shown in this slide.

