I must have created this (by now historical) slideset around the year 2004. May still be useful as a visual tour through the basic ideas of echo state networks. -- Herbert Jaeger, November 2023

The echo state approach to recurrent neural networks

Herbert Jaeger

International University Bremen (IUB)

co-operating with
Fraunhofer Institute for Autonomous Intelligent Systems (AIS)

Overview

1. Recurrent neural networks: introduction
2. Echo state networks and "liquid state" networks: a new approach to RNNs
3. Examples, examples, examples

Overview

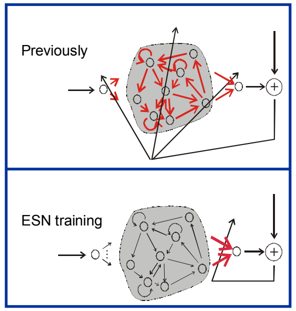
1. Recurrent neural networks: introduction
2. Echo state networks and "liquid state" networks: a new approach to RNNs
3. Examples, examples, examples
4. Open Issues
5. Conclusion

1 Recurrent neural networks

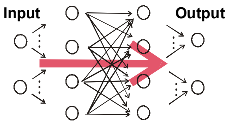
1.1 A first overview

What are ESNs?

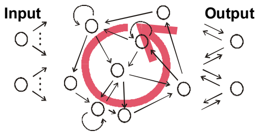
- training method for recurrent neural networks
- black-box modelling of nonlinear dynamical systems
- supervised training, offline and online
- exploits linear methods for nonlinear modeling



Feedforward- vs. recurrent NN

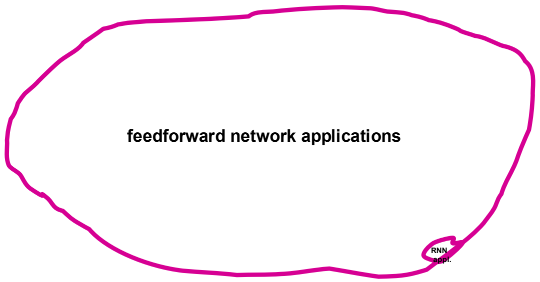


- connections only "from left to right", no connection cycle
- activation is fed forward from input to output through "hidden layers"
- no memory

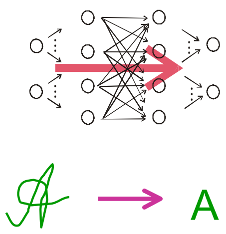


- at least one connection cycle
- activation can "reverberate", persist even with no input
- system with memory

The potato model of NN research

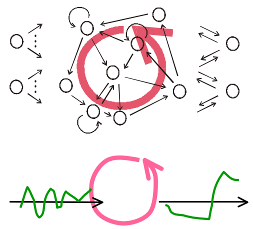


feedforward NNs



- static map input pattern → output pattern
- can approximate any nonlinear mapping
- well-understood training method (e.g., backpropagation algorithm for supervised teaching)
- main class of application: pattern recognition
- hundreds of books and software packages
- hundreds of variants for all purposes
- hype days are over, solid technology, solid maths

recurrent NNs, main properties



- input time series → output time series
- can approximate any dynamical system (universal approximation property)
- mathematical analysis difficult
- learning algorithms computationally expensive and difficult to master
- few application-oriented publications, little research

No!!!!

Biological networks are recurrent -- *we* are recurrent!

Biological networks can do (almost) *every* conceivable thing.

Biologists want to *understand* biological nets.

Engineers want to *do* (almost) every conceivable thing.

RNN models in neuroscience

bottom-up, detailed neurosimulation

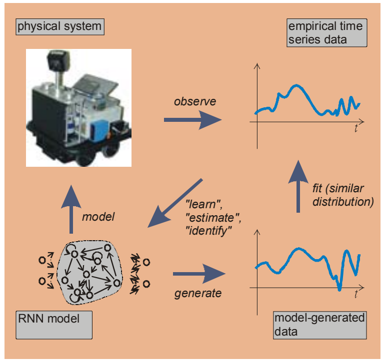
- compartment models (e.g. G¸nt¸rk¸n's PFC model)
- complex network architectures (e.g. Freeman's olfactory bulb models)

top-down, investigation of principles

- complete mathematical study of tiny networks (Pasemann, Giannakopoulos)
- universal properties of dynamical systems as "explanations" for cognitive neurodynamics
  - concept ~ attractor state; learning ~ parameter change; bifurcations ~ jumps in learning and development
- demonstration of dynamical working principles
  - synaptic dynamics and conditioning
  - synfire chains

"Glassbox" modeling: internal properties of model system are crucial

RNN models in engineering



Given: physical system measurement data ("training data")

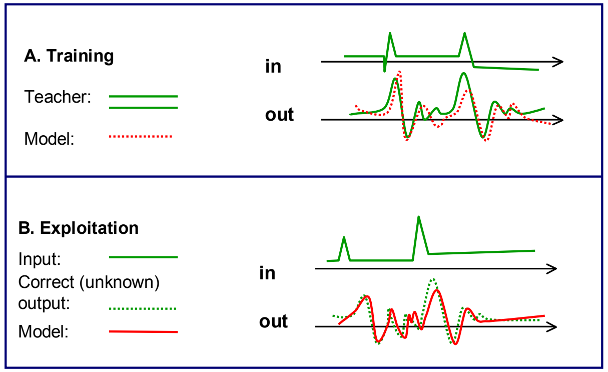
Wanted: RNN which reproduces training data and generalizes well to test data.

"Blackbox" modeling: internal properties of model system are irrelevant

Nonlinear blackbox modeling applications

Task type	Application examples
Dynamic pattern classification	Fault detection in machines; speech recognition; brain-computer-interfacing
Control	Control of novel electrical machines; dynamic combustion control in automobiles
Filtering, denoising, equalization	Channel equalization in satellite channels; hearing aids and hearing implants
Pattern generation	Computer game animation; dynamical models of humans, machines, natural systems; speech synthesis
Time series prediction	Prediction of currency exchange rates; prediction of coronary attacks

Supervised training of RNNs



This lecture will mostly be about RNN blackbox models in engineering and supervised training.

(Please, do speculate about glassboxes.)

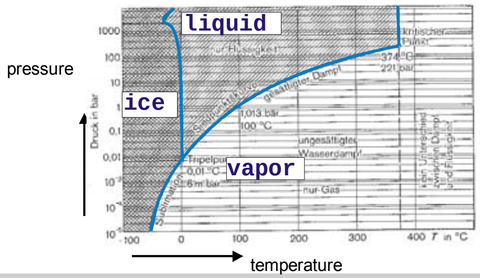
1.2 Why, exactly, RNNs are beasts: state of the art

They are beasts because they are high-dimensional, nonlinear, dynamical systems.

And nobody understands high-dimensional, nonlinear, dynamical systems!

Bifurcations: a universal phenomenon in dynamical systems

Bifurcation: *qualitative* change induced by control parameters

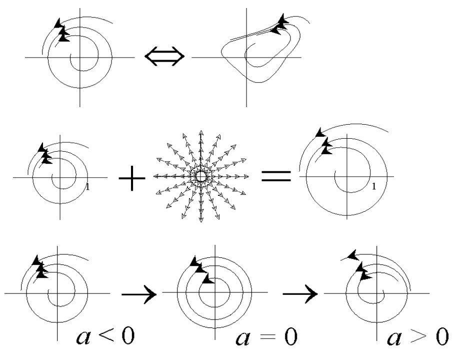


Bifurcation theory in a nutshell

topological equivalence

structural stability

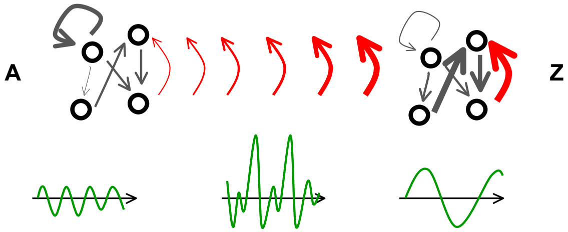
bifurcation



Learning in RNNs and bifurcations

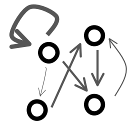
Goal of learning: achieve qualitatively new system behavior. That is, change from untrained network **A** to trained network **Z**.

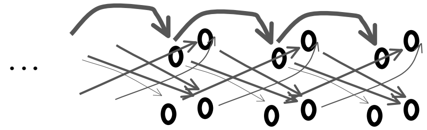
Mechanism of learning: adapt weights incrementally, minimizing some error function. I.e., change system control parameters.



Backpropagation through time (BPTT)

- **Most widely used general-purpose supervised training algorithm**
- **Idea: 1. stack network copies, 2. interpret as feedforward network, 3. use backprop algorithm.**





problems with gradient-based training methods

- passing through bifurcation → error function does not change smoothly, local gradient information of limited value
- error gradient information shrinks exponentially → no long memory effects trainable (more than order of magnitude 10 time steps difficult)
- can get trapped in local optimum → costly search of error surface
- non-local information needed → biologically impossible
- computationally costly → only small networks trainable
- algorithms and maths are difficult → experienced professionals needed
- But: very powerful in the hands of experts

Consequences for glassbox models of RNN learning

If learning is synaptic weight adaptation, and if weight adaptation is change of control parameters, and if change of control parameters induces bifurcations, how is learning possible? *(no way out: "well, learning comes about in jumps, doesn't it?"*)

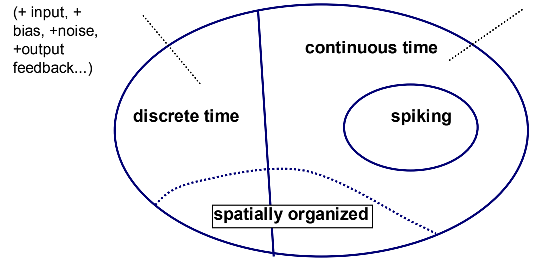
If brains are recurrent dynamical systems, and if in recurrent systems everything dynamically influences everything else, how is it possible that learning in one place does not disrupt the learnt in other places? (*no way out: sparse coding, redundancy, "plasticity-stability dilemma"*)

Workarounds

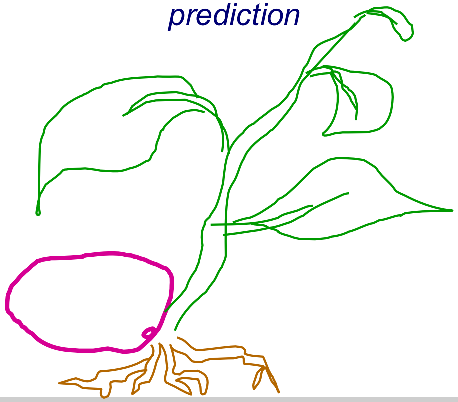
- use restricted type of RNN, especially Hopfield nets
- use special architectures with special learning algorithms
  - "long short term memory" (LSTM) nets (Schmidhuber et al.)
  - Elman networks
- evolve networks instead of training them
- use feedforward nets with sliding-window coded temporal input

Some basic types of RNN dynamics

$$x_i(n+1) = f(\sum w_{ij} x_j(n)) \qquad \tau \dot{x}_i = -x_i + \sum w_{ij} f(x_j)$$



The potato model of NN research, *prediction*

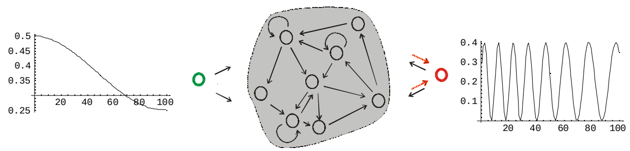


2 Echo state networks and "liquid state" networks: a new approach to RNNs

2.1 The basic idea: echo states in a dynamical reservoir

Introductory example: a tone generator

Goal: train a network to work as a tuneable tone generator

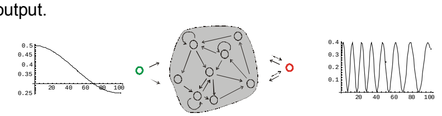


input: frequency setting

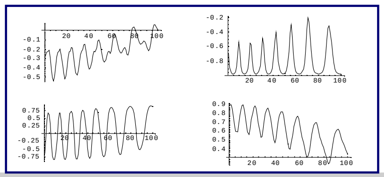
output: sines of desired frequency

Tone generator, sampling

- For sampling period, drive fixed "reservoir" network with teacher input and output.

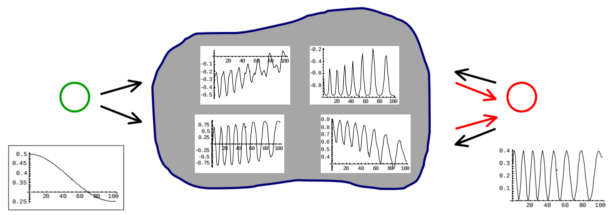


- Observation: internal states of dynamical reservoir reflect and modify both input and output teacher signals



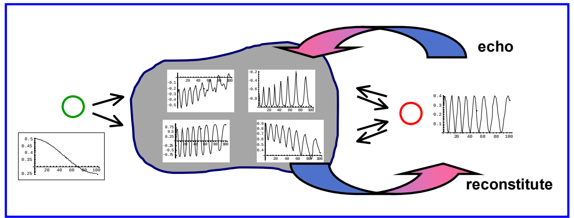
Tone generator: compute weights

- Determine reservoir-to-output weights → such that training output is optimally reconstituted from internal "echo" signals.



Tone generator: exploitation

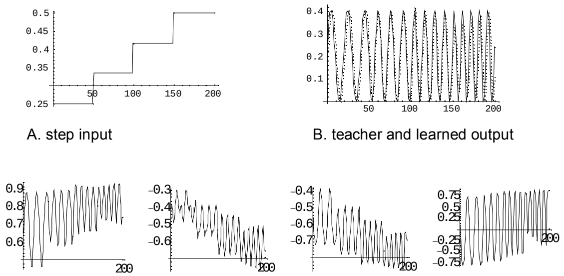
- With new output weights in place, drive trained network with input.



- Observation: network continues to function as in training.
  - internal states reflect/modify input *and output*
  - output is reconstituted from internal states
- internal states and output constitute *each other*

Tone generator: generalization

The trained generator network also works with input different from training input



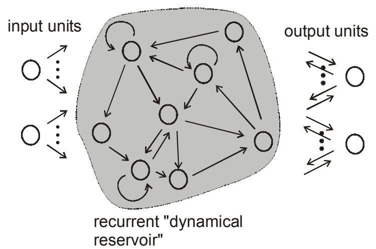
A. step input

B. teacher and learned output

C. some internal states

Dynamical reservoir

- large recurrent network (100 - ∞ units)
- works as "dynamical reservoir", "echo chamber"
- units in DR respond differently to excitation
- output units combine different internal dynamics into desired dynamics

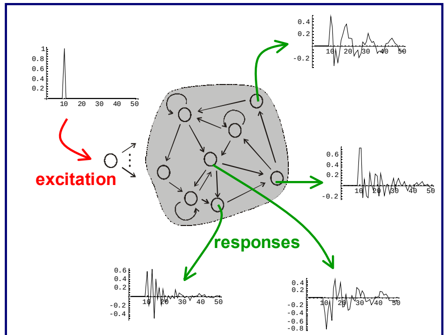


Rich excited dynamics

Unit impulse responses should vary greatly.

Achieve this by, e.g.,

- **inhomogeneous connectivity**
- **random weights**
- **different time constants**
- **...**



Input history echo

Consider internal unit i and its activation $x_i(t)$ at time t.

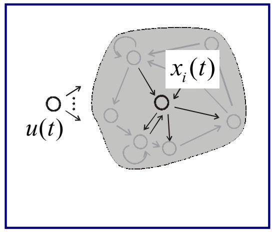
Under certain conditions (e.g.,$\sigma_{\max} < 1$),
an "input echo function"

$$h_i : \Re^N \to \Re$$

exists for every unit i such that

$$\boxed{x_i(t) = h_i\left(u(t), u(t-1), \ldots\right)}$$

where $u(t), u(t-1), \ldots$
is the input history.



We say, the network has **echo states**.

Maths of echo states

Theorem. Given certain compactness conditions, the following conditions are equivalent:

1. The network has echo states.
2. The network is state forgetting.
3. The network is state contracting.
4. The network is input forgetting.

Corollary. If the linearization of a sigmoid unit network has a weight matrix with $\sigma_{\max} < 1$, the network has echo states for arbitrary inputs.

"state forgetting"

A system is **state forgetting** iff

for all left-infinite input sequences $u(0)$, $u(-1)$,...
there exists a null sequence $(d_h)_{h=1,2,\ldots}$
such that for all starting states x, y, for all h:

$$d\big(T(x, u(-h),\ldots,u(0)),\ T(y, u(-h),\ldots,u(0))\big) < d_h.$$

"state contracting"

A system is **state contracting** iff

for all right-infinite input sequences $u(0)$, $u(1)$,...
there exists a null sequence $(d_h)_{h=1,2,\ldots}$
such that for all starting states x, y, for all h:

$$d\big(T(x, u(0),\ldots,u(h)),\ T(y, u(0),\ldots,u(h))\big) < d_h.$$

"input forgetting"

A system is **input forgetting** iff

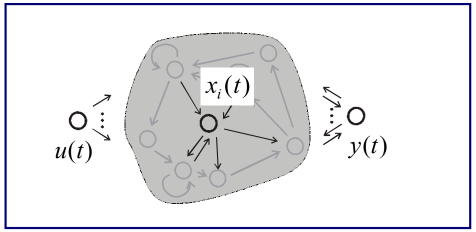
for all left-infinite input sequences $u(0)$, $u(-1)$,...
there exists a null sequence $(d_h)_{h=1,2,...}$
such that for all h, for all suffixes $\overline{u}_h$, for all prefixes $\overline{w}, \overline{v}$, for all states x,y compatible with $\overline{w}\overline{u}_h, \overline{v}\overline{u}_h$,

$d(x, y) < d_h$.

I/O history echo

Under similar conditions "I/O echo functions" exist for teacher-forced output:

$$\boxed{x_i(t) = h_i\big(u(t), u(t-1), \ldots, y(t-1), y(t-2), \ldots\big)}$$



Learning: basic idea

Every stationary deterministic dynamical system can be defined by an equation like

$$d(t) = h\big(u(t), u(t-1), \ldots, d(t-1), d(t-2), \ldots\big)$$

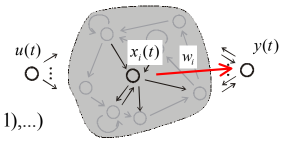
where the system function h might be a monster.

Combine h from the I/O echo functions by selecting suitable DR-to-output weights w_i:

$$h \approx \sum_i w_i h_i$$

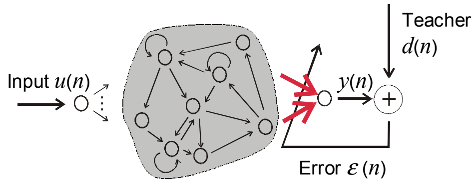
$$\Rightarrow d(t) \approx y(t) = \sum_i w_i x_i(t)$$

$$= \sum_i w_i h_i(u(t), \ldots, y(t-1), \ldots)$$



Training

Adjust only weights to output units



Offline training: task definition

Recall

$$y(t) = \sum_i w_i x_i(t)$$

Let $d(t)$ be the teacher output.

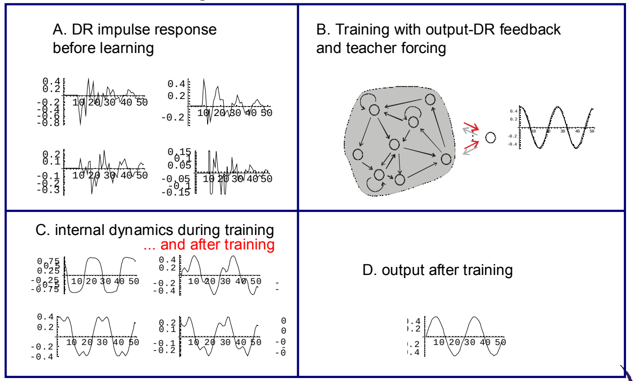
Compute weights such that mean square error

$$E[(d(t) - y(t))^2] = E[(d(t) - \sum w_i x_i(t))^2]$$

is minimized.

Basic idea illustrated

Learn sinewave generator with 20-unit DR



Offline training: how it works

1. Let network run with training signal $d(t)$ teacher-forced.
2. During this run, collect network states $x_i(t)$.
3. Compute weights w_i, such that $E[(d(t) - \sum w_i x_i(t))^2]$ is minimized (boils down to computing a pseudoinverse of the collected state's matrix)

MSE minimizing weight computation (step 3) is a standard operation.

Many efficient implementations available, offline/constructive and online/adaptive.

Online learning, LMS algorithm

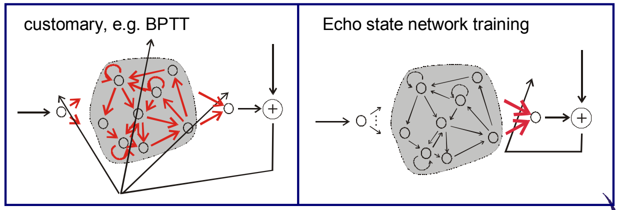
1. Let network run with training signal $d(t)$ teacher-forced.
2. At every time step, adjust output connection weights according to

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \lambda\,(y(t) - d(t))\,\mathbf{x}(t)$$

where λ is learning rate, $y(t)$ current network output, $\mathbf{x}(t)$ current network state.

Training, comparison

- no cyclic dependencies between trained weights
- all customary methods for mean square error minimization of *linear* system applicable



Echo state network training, summary

- use large recurrent network as "excitable dynamical reservoir (DR)"
- DR is not modified through learning
- adapt only DR $\rightarrow$ output weights
- thereby combine desired system function h from I/O history echo functions
- use any offline or online linear regression algorithm to minimize error

$$E[(d(t)-y(t))^2]$$

2.2 "Liquid state machines"

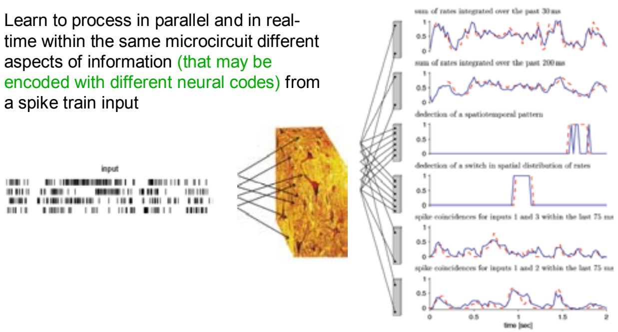
Liquid state machines

Wolfgang Maass et al: "liquid state machines":

- shares idea of readout from dynamical reservoir ("liquid state")
- continuous-time, spiking dynamics
- arbitrary readout functions / networks
- focus of analysis: computational universality of class of liquid state machines
- main result: every filter with "fading memory" can be approximated in the class of LSMs

(courtesy W.M.)

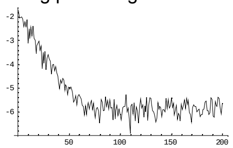
Learn to process in parallel and in real-time within the same microcircuit different aspects of information (that may be encoded with different neural codes) from a spike train input

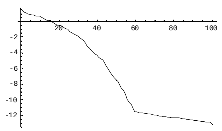


2.3 Online learning for adaptive modeling

Delay line: online training with LMS

- LMS algorithm: $\mathbf{w}(t+1) = \mathbf{w}(t) + \mu\,\varepsilon\,\mathbf{x}(t)$
- simple, robust
- local, biologically not immediately implausible
- bad asymptotic convergence if state crosscorrelation matrix has high eigenvalue spread
- example: log learning curve of 10-delay for 100-unit network and log plot of eigenvalues of crosscorrelation matrix

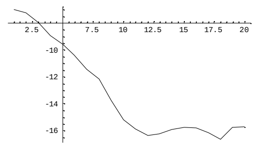




fast initial convergence, no chance of asymptotic convergence (to MSE_{min} = 2.6 x 10^{-17}) in historic time (2000 steps shown)

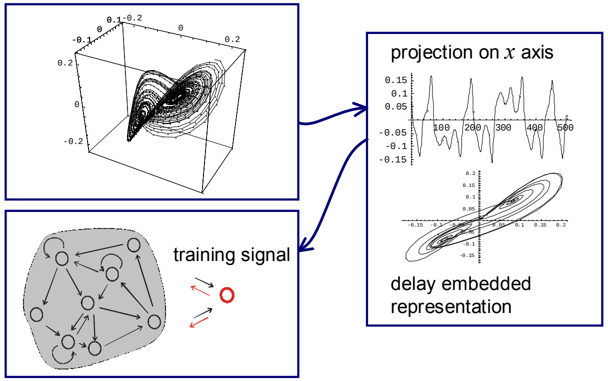
Delay line: RLS online training

- RLS algorithm: "least square error curve fitting with forgetting"
- complexity $O(N^2)$ [vs. $O(N)$ of LMS]
- no convergence slowdown through eigenvalue spread
- example: same as before, 2000 steps shown

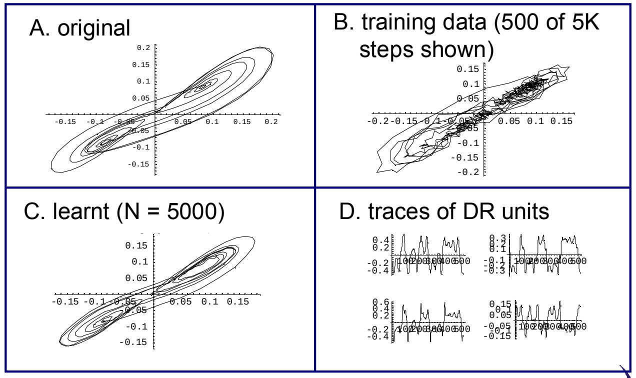


- MSE after 2.000 steps (λ=0.97): 1.4 x 10^{-16} (MSE_{min} = 2.6 x 10^{-17})

Lorenz attractor from noisy data



Lorenz attractor learnt from noisy data with RLS

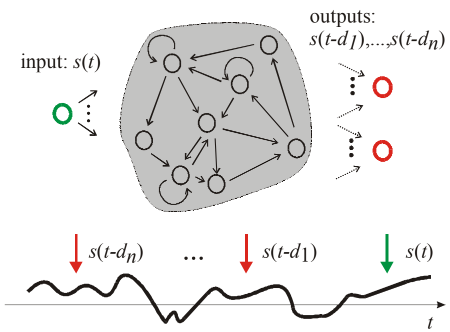


3 Examples

- 3.1 Short-term memories
- 3.2 Learning nonlinear dynamics
- 3.3 Dynamic pattern recognition
- 3.4 Static pattern recognition
- 3.5 "Bidirectional" dynamics
- 3.6 Stochastic dynamics
- 3.7 Nonlinear control

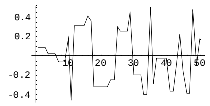
3.1 Short-term memories

Delay line: scheme



Delay line: example

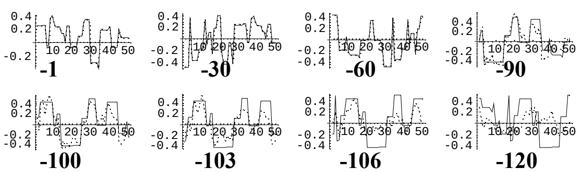
- Network size 400
- Delays: 1, 30, 60, 70, 80, 90, 100, 103, 106, 120 steps
- Training sequence length N = 2000



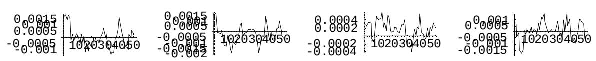
training signal: random walk with resting states

results

correct delayed signals (———) and network outputs (-------)

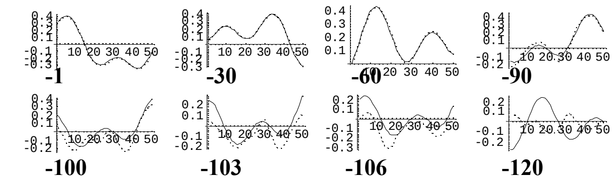


traces of some DR internal units

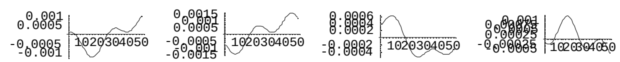


Delay line: test with different input

correct delayed signals (———) and network outputs (-------)



traces of some DR internal units



Memory capacity of delay line

Definition. Let $u(n)$ be a single-channel input process. Define the memory capacity $\mathbf{mc}$ of network for this input process by

$$\mathbf{mc} = \sum_{k=0}^{\infty} r^2(u(n-k), y_k(n))$$

where r is correlation coefficient, $y_k(n)$ is trained network output for $u(n-k)$

Memory capacity of delay line

Theorem. In a network with N nodes, i.i.d. input, $\mathbf{mc} \leq N$.

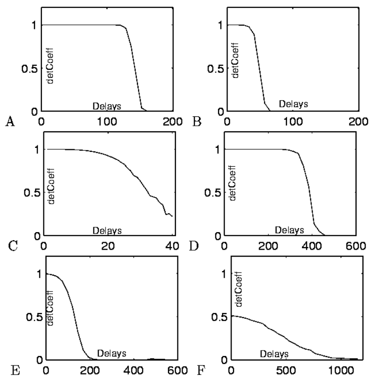
Theorem. In a linear network with N nodes, i.i.d. input, generically $\mathbf{mc} = N$.

Theorem. In a linear network, long delays can never be learnt better than short delays ("monotonic forgetting")

Notes.

1. In networks with nonlinear readout functions, $\mathbf{mc} = \infty$ may occur.
2. Input not i.i.d: $\mathbf{mc} > N$ is possible (network can exploit regularities in signal for longer memory span)

Forgetting curves



A. 400 linear unit network

B. Like A, sigmoid units

C. Like A, noisy state update

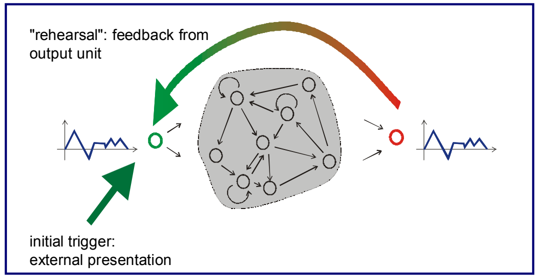
D. 400 linear unit network, unitary weight matrix

E. Like D, noisy state update

F. Like D, spectral radius 0.999

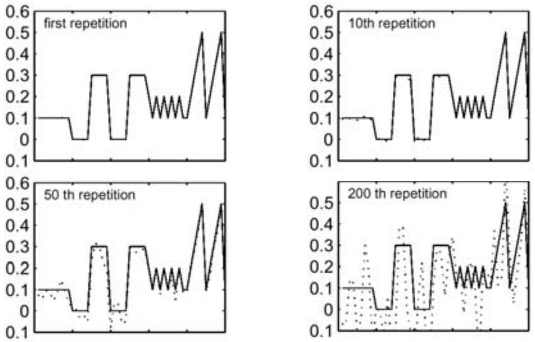
Cyclic rehearsal

- Example of phenomenon: phone number memorizing
- Possible use in humans: keeping signal alife in STM for transfer to LTM
- no claims about approriateness of mechanism!



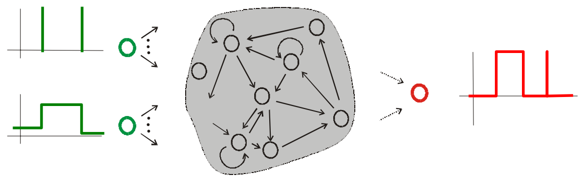
Results

100 unit linear network, trained on 50 step delay



A timer

Network setup



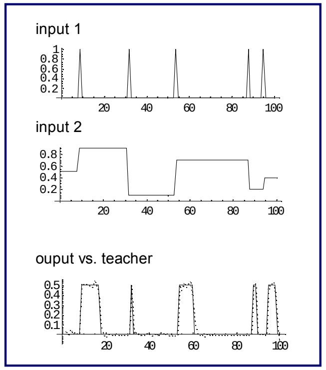
input 1: start signals

input 2: duration setting

ouput: rectangular signals of desired duration

Results

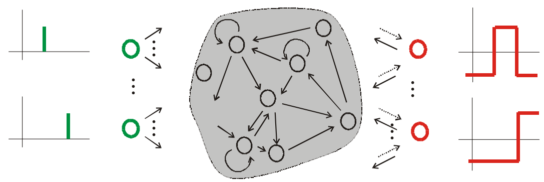
- network size 100
- training length 2K
- durations 1 - 10
- MSEtest 0.0029



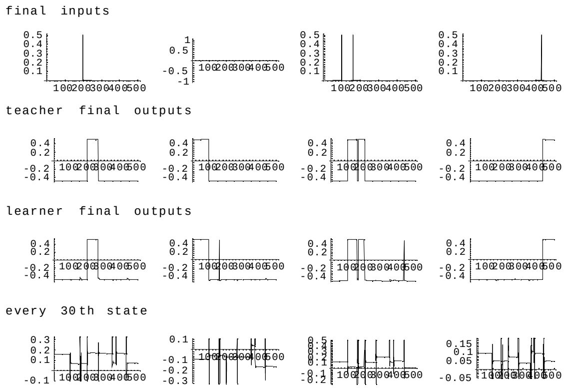
20-state multiflop

input: 20 channels for rare spikes

output: 20 last spike indicators



20-multiflop: performance

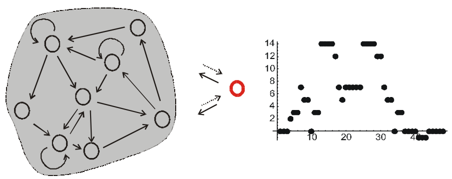


20-multiflop: remarks

- a very nonlinear task
- output feedback crucial
- network size 100
- training size 4000 (200 steps per memory state)
- noise insertion into network during training required
- a kind of infinite-time memory with 20 memory states (20 input-switchable point attractors)

Melody learning

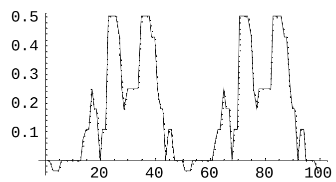
- obtain generator of artificial periodic sequence (a "melody") of period length p
- dual to p-multiflop task
- network becomes a length-p cyclic attractor
- output feedback essential



"House of the rising sun"

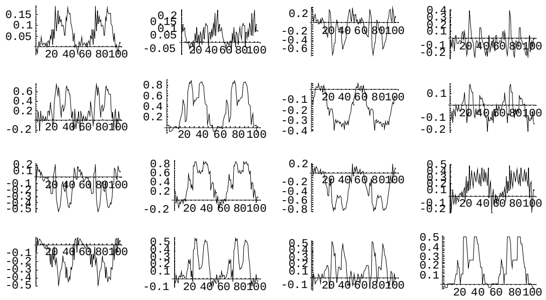
Melody: example

Teacher vs. network output



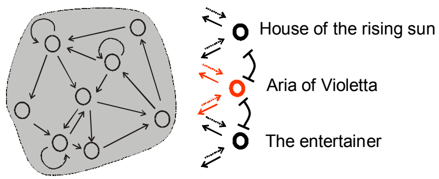
- network size 400 units
- training sequence length 2500
- test error MSE 7^{-14}

Inside the rising sun



A melody per unit

- The "knowledge" about melody lies in the synaptic input weights of the (single) output unit
- Learning more melodies means adding & training more output units to existing network



Memories: Summary

- Various forms of memory:
  - transient STM (delay line)
  - transient STM, variation (timer)
  - repetitive transient (cyclic recital)
  - repetitive attractor (melody recall)
  - switchable point attractor (multiflop)
- How many more? systematic classification?
- high-capacity contents concentrated in a single unit's ingoing connections
- more contents: add single unit

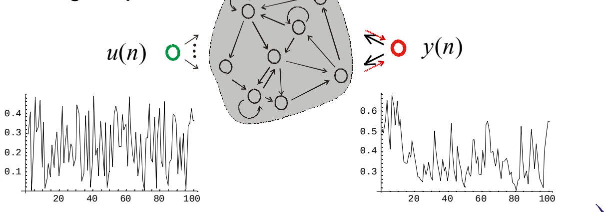
3.2 Indentification of nonlinear systems

Identifying higher-order nonlinear systems

A tenth-order system

$$y(n+1) = 0.3y(n) + 0.05y(n)\sum_{k=0,\ldots 9} y(n-k) + 1.5u(n)u(n-9) + 0.1$$

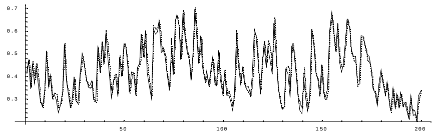
Training setup



Results: offline learning

augmented ESN (800 Parameters) :

$NMSE_{test} = 0.006$

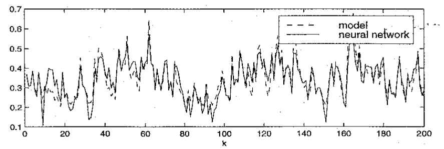


previous published state of the art[1]:

$NMSE_{train} = 0.24$

D. Prokhorov, pers. communication[2]:

$NMSE_{test} = 0.004$

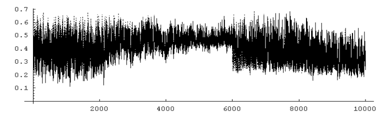


[1] Atiya & Parlos (2000), *IEEE Trans. Neural Networks* **11**(3), 697-708

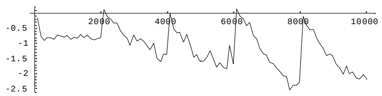
[2] EKF-RNN, 30 units, 1000 Parameters.

Results: adaptive online learning with RLS-ESN[1]

Experiment: system parameters changed every 2000 steps

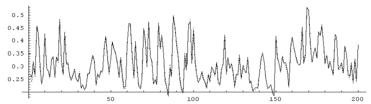


Log_{10} of NMSE



Closeup on last 200 steps,

$NMSE_{test} = 0.006$

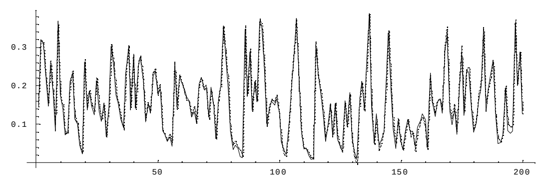


[1] Mathematica code, documentation and experiments obtainable from my homepage

30 th order system

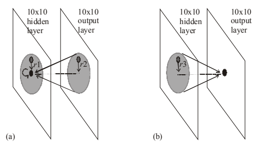
$$y(n+1) = 0.2y(n) + 0.04y(n)\sum_{k=0,\ldots 29} y(n-k) + 1.5u(n)u(n-29) + 0.001$$

- online training, 800-unit augmented ESN, trained for 12K steps
- $NMSE_{test} = 0.008$

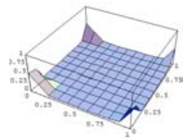


Excitable medium

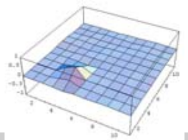
network architecture



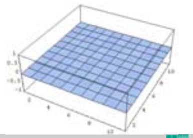
teacher: wandering soliton



learnt medium, emergence of a "block wave"



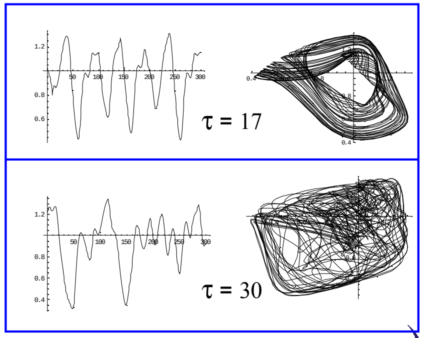
learnt medium, emergence of a "transversal wave"



The Mackey-Glass equation

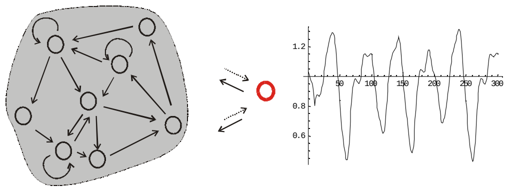
$$\dot{x}(t) = 0.2\, x(t-\tau)/(1 + x(t-\tau)^{10}) - 0.1\, x(t)$$

- delay differential equation
- delay $\tau > 16.8$: chaotic
- benchmark for time series prediction



Learning setup

- continuous-time network $\dot{\mathbf{x}} = C(-D\mathbf{x} + \mathbf{f}(\mathbf{W}\mathbf{x}))$
- network size 400, sparse random network
- training sequence N = 3000

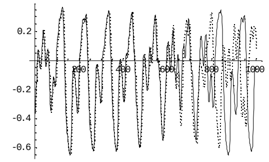


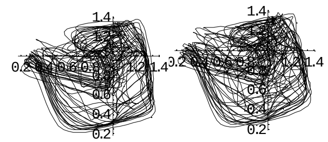
Results for $\tau = 30$

error for 120-step sequence prediction (training length 1.5 + 4 K): $\mathrm{RMSE}_{120} = 0.0099$

best previous result[1)]:

$\mathrm{RMSE}_{120} = 0.04$





original learnt model

[1)] Feldkamp et al, Enhanced Multi-Stream Kalman filter training for recurrent networks, 1998

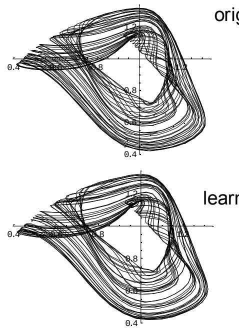
Results for τ = 17

Error for 84-step prediction:

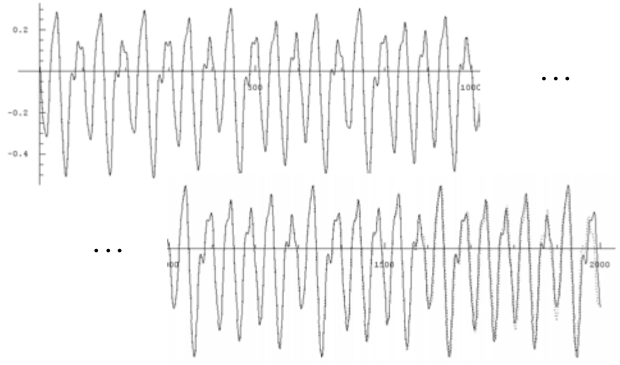
training data 3K:
NRMSE = 0.00007

training data 6K
NRMSE = 0.00001

previous best
NRMSE = 0.02

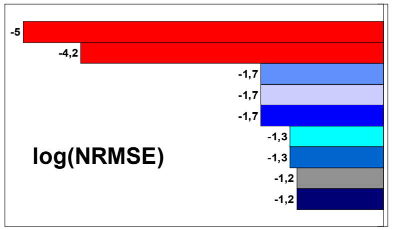


Prediction with model



visible discrepancy after about 1500 steps

Comparison: NRMSE for 84-step prediction

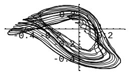
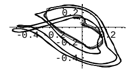
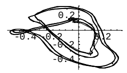
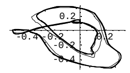
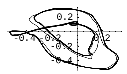
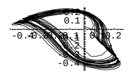
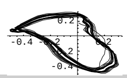
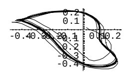


ESN (1.5 +4.5 K)	-5
ESN (1+2 K)	-4,2
PCR Local Model (McNames 99, 2 K)	-1,7
SOM (Vesanto 97, 3K)	-1,7
DCS-LLM (Chudy & Farkas 98, 3K)	-1,7
AMB (Bersini et al 98, ? K) *	-1,3
Neural Gaz (Martinez et al 93, ~4K)	-1,3
EPNet (Yao & Liu 97, 0.5 K)	-1,2
BPNN (Lapedes & Farber 87, ? K) *	-1,2

**) data from survey in Gers / Eck /Schmidhuber 2000*

Noisy training data, τ = 17

- 400 unit ESN trained with 2000 steps of MG + noise
- Test criterium: NMSE on 200 step prediction

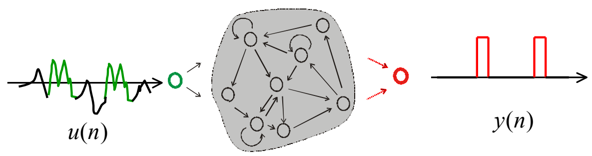
SNR	white Gaussian noise	NMSE		surrogate process noise
100		0.025	0.034	
20		0.18	0.91	
10		0.21	1.33	
5		0.24	1.31	

3.3 Dynamic pattern recognition

Dynamic pattern detection[1)]

Training signal:

output jumps to 1 after occurence of pattern instance in input



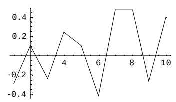
[1)] see GMD Report Nr 152 for detailed coverage

Single-instance patterns, training setup

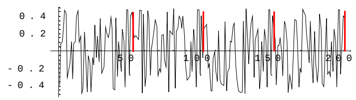
1. A single-instance, 10-step pattern is randomly fixed

2. It is inserted into 500-step random signal at positions
200 (for training)
350, 400, 450, 500 (for testing)

3. 100-unit ESN trained on first 300 steps (single positive instance! "single shot learning), tested on remaining 200 steps



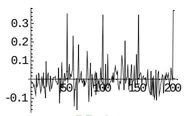
the pattern



test data: 200 steps with 4 occurances of pattern on random background, desired output: red impulses

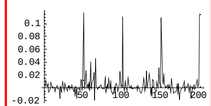
Single-instance patterns, results

1. trained network response on test data



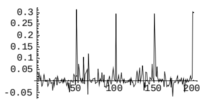
DR: 6.4

2. network response after training 800 more pattern-free steps ("negative examples")



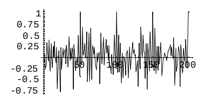
DR: 12.1

3. like 2., but 5 positive examples in training data



DR: 12.4

4. comparison: optimal linear filter



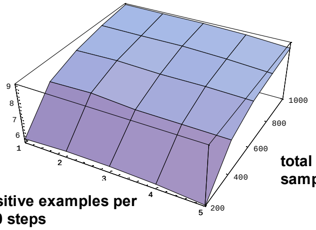
DR: 3.5

discrimination ratio DR:

$$\sqrt{E[d^2(n^+)]/E[d^2(n^-)]}$$

Single-instance patterns, positive vs. negative examples

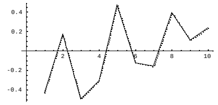
- Larger networks: no significant improvement of DR
- Improvements through larger training sample
- No improvement through larger density of positive examples (explainable through discrimination learning of negative examples)



(surface = average over 50 independent training experiments)

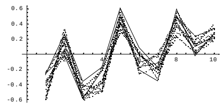
pattern = prototype + noise

1. prototype: random 10-step sequence



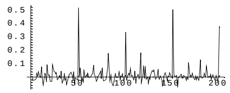
2. pattern instances: prototype + 20% uniform noise

3. training data: 1000 steps with 20 pattern instances; test data: 4 pattern instances



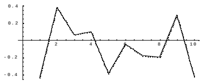
Result:

DR 11.9



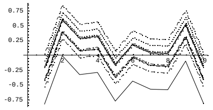
pattern = shifted prototype

1. prototype: random 10-step sequence



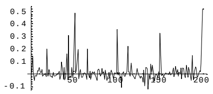
2. pattern instances: prototype + random shift, max. ± 50 %

3. training data: 1000 steps with 20 pattern instances; test data: 4 pattern instances (spanning maximal shift range)



Result:

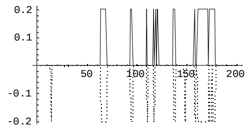
DR 8.3



pattern = satisfaction of nonlinear decision function

pattern: an instance of pattern P is present at time n when decision function D is above threshold C	$P = \{p(1)\ldots p(10) \mid D(p(1),\ldots,p(10) > C\}$
We choose some arbitrary D	$D(p_1,\ldots,p_{10}) = \sin(p_5) + \tan^2(p_6)/10 + \sum_{i=1,\ldots,10} p_i$
Training sample size 1000	$C = 2$

Result: DR 5.3, error rate 2.5% after thresholding network output for decision



pattern = thresholding dynamical system driven by input

- pattern is present at time n when decision function D is above threshold C
- D is output of dynamical system with state variables $x, y,$ driven by input $\ldots, u(-1), u(0).$
- Training sample size 1000

$$x(n+1)=\frac{\left\|\sqrt{x(n)}\right\|}{1+5u^2(n)}-\frac{y^3(n)}{1+x^2(n)+y^2(n)}$$

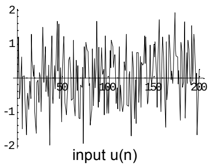
$$y(n+1)=u^3(n)-\frac{x(n)\,y(n)}{1+x^2(n)}$$

$$D(n+1)=5\,x(n)u(n+1)-y(n)\sin(x(n))$$

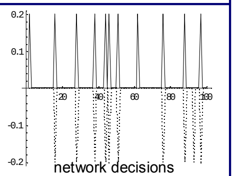
Result: DR 4.3

decision error rate 4.5% (100 unit network)

resp. 3.0% (400 unit network)



input u(n)



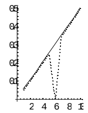
network decisions

Discrimination learning: problem

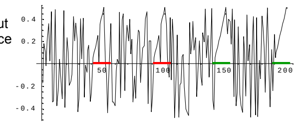
- Problem: network trained on positive examples generalizes incorrectly.
- Solution: learn to discriminate by training on negative examples.
- Example: "wedged" ramp discrimination.

desired pattern

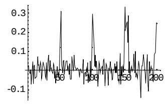
undesired competitor pattern



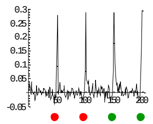
test input sequence



single-trial response of 100-unit network trained on single positive instance in 200 training points

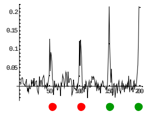


response averaged over 10 test sequences: almost no discrimination



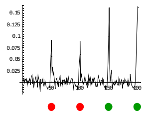
Discrimination learning: results

averaged response, training on 1 positive and 1 negative expl.

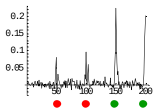


dito, training on 14 positive and 14 negative expls.

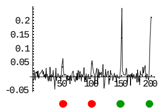
network capacity insufficient for discrimination



same as above, 400 unit network



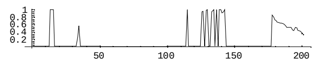
same as above, 400 unit network



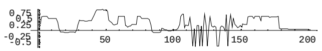
Event detection for robots

(joint work with J.Hertzberg & F. Schˆ nherr)

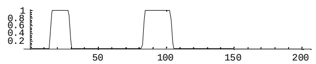
Robot runs through office environment, experiences data streams (27 channels) like...



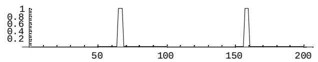
infrared distance sensor



left motor speed



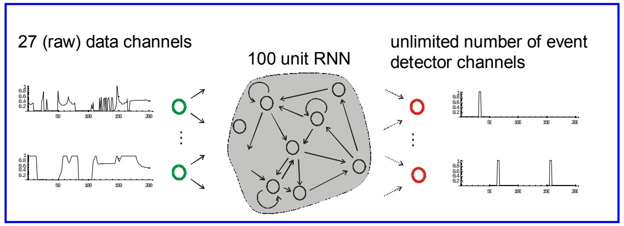
activation of "goThruDoor"



external teacher signal, marking event category

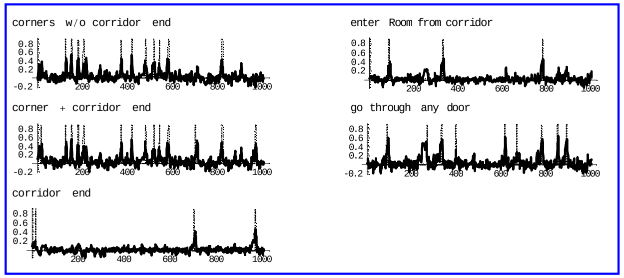
10 sec

Learning setup



- simulated robot (rich simulation)
- training run spans 15 simulated minutes
- event categories like
  - pass through door
  - pass by 90∞corner
  - pass by smooth corner

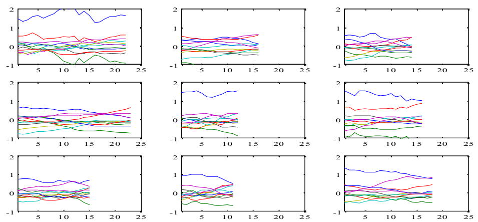
Results



- easy to train event hypothesis signals
- "boolean" categories possible
- single-shot learning possible

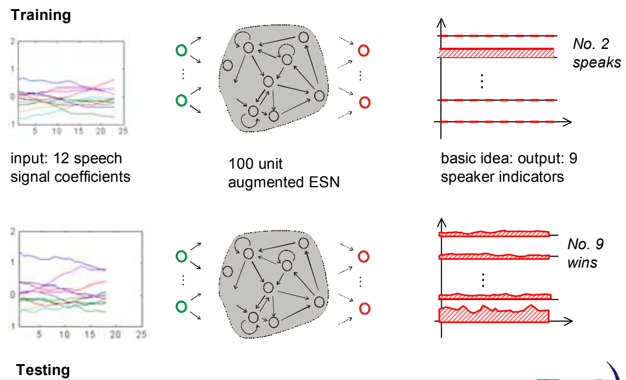
Japanese vowels: task

- benchmark data set from UCI KDD repository[1)]
- 9 male Japanese speakers, samples of two-vowel utterance /ae/, represented by 12 LPC cepstrum coefficients
- task: discriminate speakers (learn on 9 x 30 samples, test on 370 samples in total, unevenly distributed)
- Figure: exemplary samples from the 9 speakers



1) http://kdd.ics.uci.edu/

Japanese vowels: training setup



Japanese vowels: results

- discrimination learning
- 99.5 % correct classifications (2 errors / 370)
- original result[1] [5-unit continuous HMM]: 96.2 % correct (14 errors)
- latest result[2]: [dynamic Bayesian network with deterministic latent variables]: 97.3 % correct (10 errors)

---

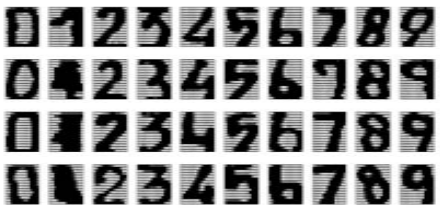
1) M. Kudo, J. Toyama and M. Shimbo. (1999). "Multidimensional Curve Classification Using Passing-Through Regions". Pattern Recognition Letters, Vol. 20, No. 11--13, pages 1103--1111.

2) David Barber: Dynamic Bayesian Networks with Deterministic Latent Tables, NIPS 2002

3.4 Static pattern recognition

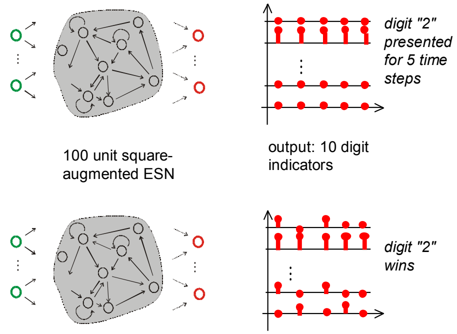
Digit recognition: task

- benchmark data set from UCL ML repository: digits 0--9, 200 samples per digits[1]
- preprocessing: 64 Karhunen-Love coefficients per sample
- train from 100 samples / digit, test on remaining 100 samples / digit



1) data donated by Robert P.W. Duin, Delft University http : //www.ph.tn.tudelft.nl/~duin

Digit recognition: training setup



Digit recognition: results

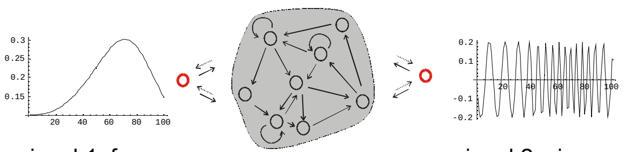
- discrimination learning
- error rate 2.8 %
- best result from comparative study of standard classification techniques[1]: 3.2 % error rate
  - achieved with majority voting combination of 12 standard classifiers: 2 Bayes classifiers, nearest mean, nearest neighbor, k-nearest neighbor, Parzen classifier, Fisher's linear discriminant, decision tree, MLP with 20 hidden units, MLP with 50 hidden units, linear and quadratic kernel SVM.
  - best single classifier: Parzen with 3.7 error rate

1) *van Breukelen / Duin / Tax, Handwritten digit recognition by combined classifiers, Kybernetika 34(4), 1998, 381-386*

3.5 "Bidirectional" dynamics

Bidirectional dynamics: frequency generating/measuring device

Teaching signal: sine wave and its frequency (N = 1000, here section with N = 100)

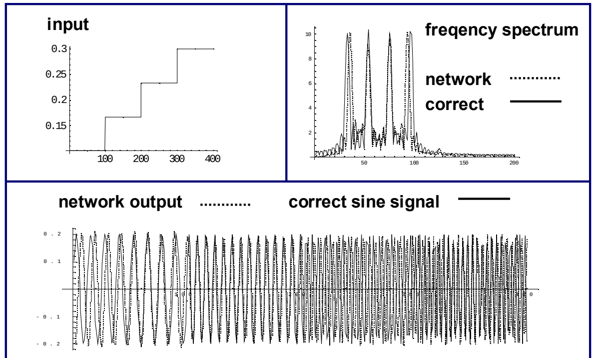


signal 1: frequency

signal 2: sines

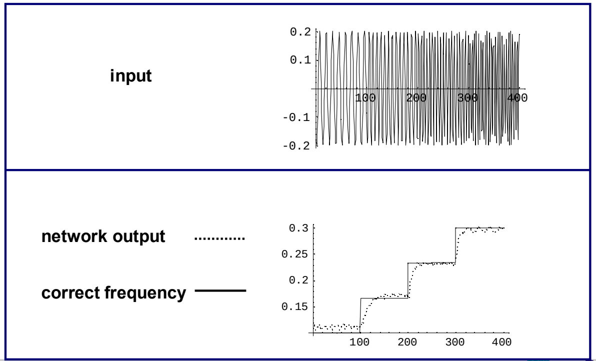
frequency generating/measuring

used as frequency generator



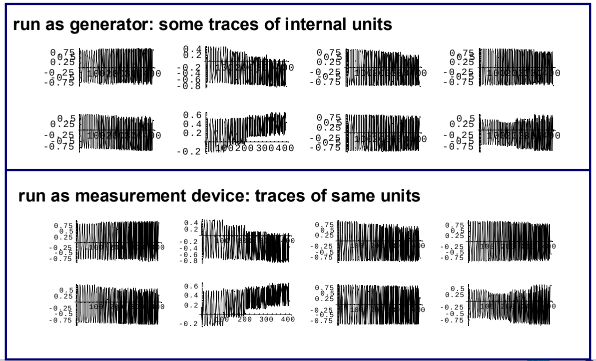
frequency generating/measuring

used for measuring frequency



frequency generating/measuring

internal dynamics similar in both "working directions"

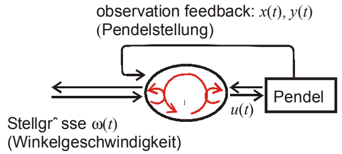


frequency generating/measuring: summary

- train network with sine waves and their frequency simultaneously
- network learns *dynamical relation* between two signals
- application 1: frequency generator (input: frequency reference, output: sinewave)
- application 2: frequency measuring device (input: sines, output: frequency)

Bidirektionale Dynamik 2: Pr%diktor/Steuerung

- Setup



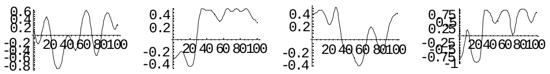
- Training: Vorgabe von $x(t)$, $y(t)$, $u(t)$, $\omega(t)$
- Verwendung Pr%diktor:

  Input $x(t)$, $y(t)$, $u(t)$ Output $\omega(t)$
- Verwendung Steuerung:

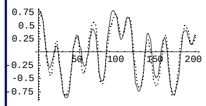
  Input $x(t)$, $y(t)$, $\omega(t)$, Output $u(t)$

Bidirektionale Dynamik 2: Steuerung/Pr‰diktor

Trainingsdaten $\omega(t),\ x(t),\ y(t),\ u(t)$ $\quad(\mathrm{N} = 1000)$

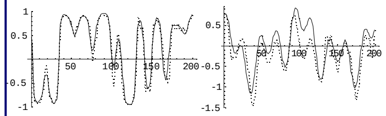


Verwendung als Pr‰diktor, output $\omega(t)$



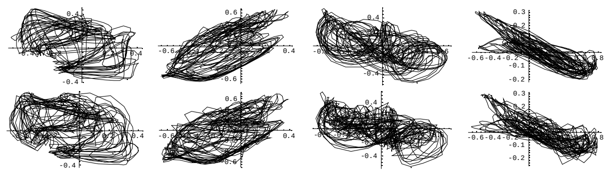
Verwendung als controller

A. output $u(t)$ B. geregeltes $\omega(t)$



Bidirektionale Dynamik 2: Steuerung/Pr‰diktor

obere Reihe: (x_i, x_{i+1}) -plots bei Verwendung als Pr‰diktor



untere Reihe: (x_i, x_{i+1}) -plots bei Verwendung als Steuereinheit

Research topic: bidirectional systems

- how to represent/understand?
  maybe[1)]

$$y(t) = g(u(t), u(t-1), \ldots)$$
$$u(t) = h(y(t), y(t-1), \ldots)$$

iff there exists system with state x, and

$$x(t) = r(u(t), u(t-1), \ldots) = s(y(t), y(t-1), \ldots)$$
$$y(t) = \tilde{g}(x(t), x(t-1), \ldots)$$
$$u(t) = \tilde{h}(x(t), x(t-1), \ldots)$$

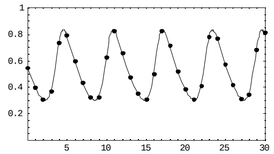
- explanation model for dynamical associative memory, for "volition = prediction", for "control space = observation space"

[1)] inspiration from Klaus Kretzschmar

3.6 Stochastic dynamics

The "probability clock" process

- a 0-1-valued stochastic process
- main property: $P(X_n = 0 \mid X_0 = 1, X_1 = 0, \ldots, X_{n-1} = 0)$ is oscillation

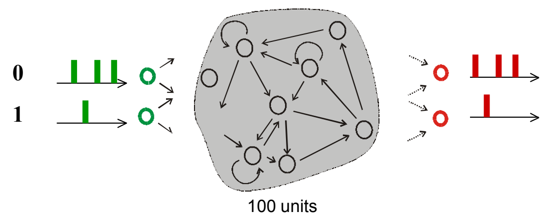


- cannot be modeled by finite-dimensional HMM
- can be modeled by 3-dim OOM

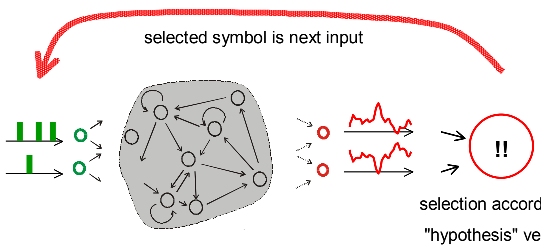
Network setup in training

2 input channels
code symbols 0 and 1

2 output channels for *next* symbol hypotheses



Trained network in sequence generation

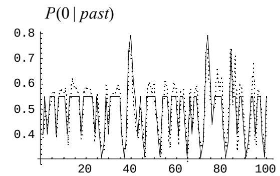


selection according to "hypothesis" vector

Results

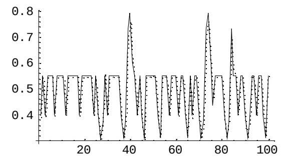
Echo state network

- train sequence length 20K
- average absolute prob error: 0.031



Comparison: trained OOM

- train sequence length 20K
- average abs. prob. error 0.0063



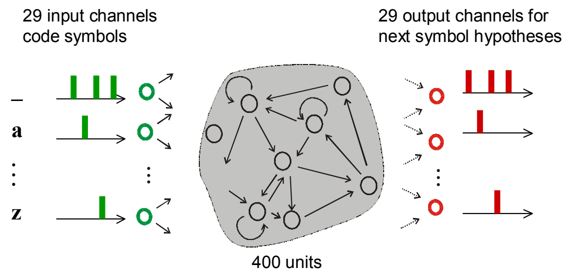
Little Red Riding Hood

training data: 3412-symbol sequence, shown here: first and last 500

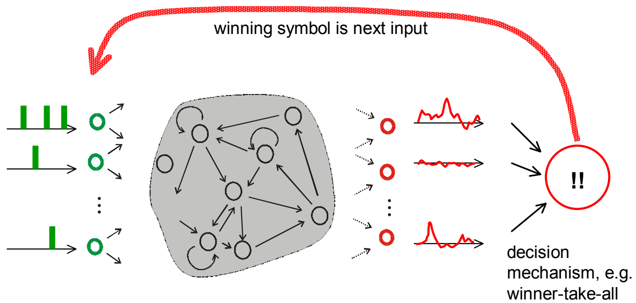
once_upon_a_time_there_was_a_little_village_girl,_the_prettiest_ever_seen_her_mot
her_doted_upon_her,_and_so_did_her_grandmother._she,_good_woman,_made_for_
her_a_little_red_hood_which_suited_her_so_well,_that_everyone_called_her_little_re
d_riding_hood._one_day_her_mother,_who_had_just_made_some_cakes,_said_to_he
r_my_dear,_you_shall_go_and_see_how_your_grandmother_is,_for_i_have_heard_sh
e_is_ailing,_take_her_this_cake_and_this_little_pot_of_butter._little_red_riding_hood
_started_off_at_once_for_he

oh,_grandmamma,_grandmamma,_what_great_arms_you_have_got_all_the_better_to
_hug_you_with,_my_dear_oh,_grandmamma,_grandmamma,_what_great_legs_you_h
ave_got_all_the_better_to_run_with,_my_dear_oh,_grandmamma,_grandmamma,_wh
at_great_eyes_you_have_got_all_the_better_to_see_with,_my_dear_oh,_grandmamm
a,_grandmamma,_what_great_teeth_you_have_got_all_the_better_to_gobble_you_up
_so_saying,_the_wicked_wolf_leaped_on_little_red_riding_hood_and_gobbled_her_u
p._here_endeth_the_tale_of_little_red_riding_hood.

Network setup in training



Trained network in "text" generation



Results

Selection by random draw according to output

yth_upsghteyshhfakeofw_io,l_yodoinglle_d_upeiuttytyr_hsymua_doey_sa
mmusos_trll,t.krpuflvek_hwiblhooslolyoe,_wtheble_ft_a_gimllveteud_ ...

Winner-take-all selection

sdear_oh,_grandmamma,_who_will_go_and_the_wolf_said_the_wolf_said
_the_wolf_said_the_wolf_said_the_wolf_said_the_wolf_said_the_wolf ...

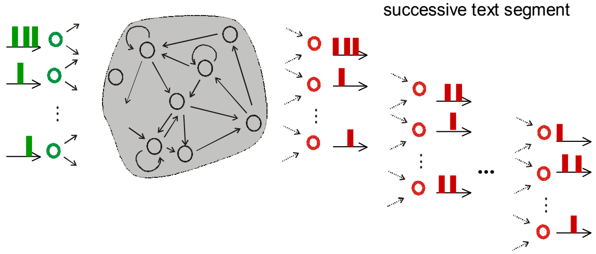
Results, continued

Selection by nonlinearly weighted random draw (namely, out^3.5)

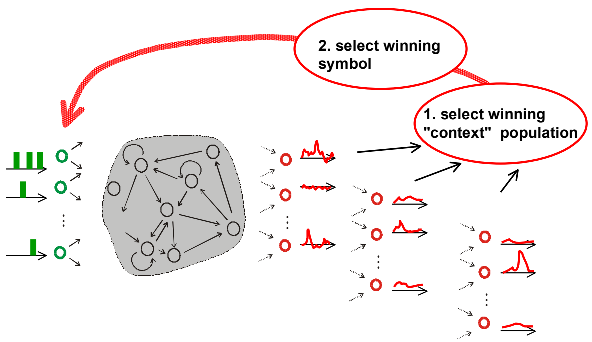
d_wolf_said_the_better_to_her_the_wood_the_wolf_she___urter_that_of_butter_to_her_grandmother,_the_door_grandmamma,_who_was_bed,_she_the_better_the_wolf_sa_and_she_little_red_her_grandmatm_aa_grandmother_mother_grandmother_mother_good_wolf,_and_the_wolf_so_said_the_she_wolf,_and_i_have_gs_at_the_wolf,_wolf_butter_to_her_come_neard_the_bobbled_her_grandmamma,_grandmamma,_who_her_the_do_wolf_cake_her_grandmother_mother_to_her_to_she_me_the_better_to_her_the_bettle_red_riding_hood_see_her_the_pot_of_butter_ ...

training variant

- 6 "context" groups of 29 output channels each
- each group is trained on successive text segment



Text generation in variant

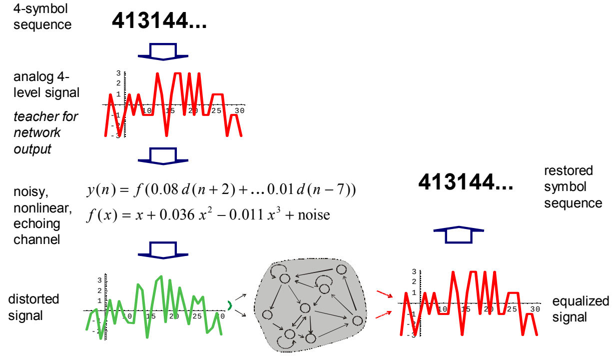


3.7 Channel equalization

What is channel equalization?

- (digital) telecommunication channels:
  - satellite radio links
  - mobile phone transmissions
  - computer networks
  - DVD reading devices
- input: coded bit sequence S = 1434234...
- S is DA-converted, transmitted ("channel"), *equalized*, AD-converted into S' = [?] 1434234...
- transmission is main source of signal corruption: echoes ("intersymbol interference"), distortion, noise
- high-gain transmissions tend to have nonlinear characteristics, especially important in low-energy devices
- equalization: undo effects of channel (de-echo, un-distort, filter noise)

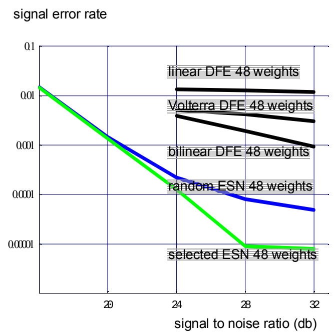
Learning setup[1)]



1) task from Mathews, V.J. & Lee, J., Adaptive Algorithms for Bilinear Filtering, SPIE Vol. 2296, 317-327

Results

- 48-unit standard ESN
- training sample size 5000
- **black:** standard nonlinear techniques[1)]
- **blue:** randomly generated ESN
- **green:** best ESN selected from 20 randomly generated
- 1.5 - 2 orders of magnitude improvement in signal error rate

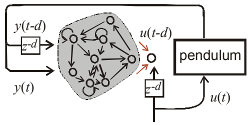


1) Mathews, V.J. & Lee, J., Adaptive Algorithms for Bilinear Filtering, SPIE Vol. 2296, 317-327

3.8 Nonlinear control

Direct / feedback tracking control

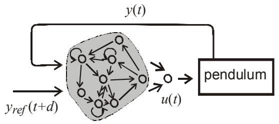
training



network "observes" torque $u(t-d)$, plant state $y(t-d)$, $y(t)$

network learns how $u(t)$ depends on $y(t)$, $y(t+d)$

exploitation



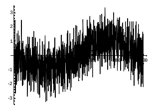
network input: reference $y_{ref}(t+d)$ and state feedback $y(t)$

network computes control input $u(t)$

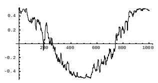
Simulated pendulum example

training data

torque $\boldsymbol{u}$



pendulum coord. $\boldsymbol{y}$

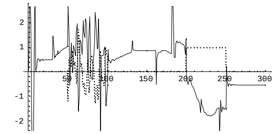


plus coord. $\boldsymbol{x}$, plus (in state feedback) $\boldsymbol{\omega}$

exploitation

control signal $\boldsymbol{u}$ issued by network [—]

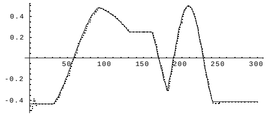
disturb. [······]



actual [······]

reference [—]

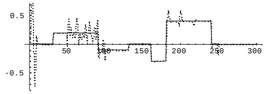
$\boldsymbol{y}$



actual [······]

required [—]

$\boldsymbol{\omega}$



Robot motor control
(data courtesy Paul Plˆ ger et al.)

Task

create motor controllers for AIS's soccer robots

Challenge

Robot system is nonlinear and has hysteresis effects due to soft carpet, sliding, high motor load

Goal

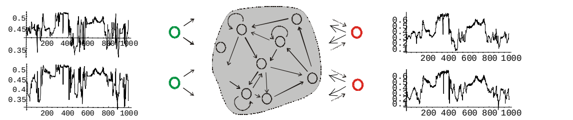
Control curve driving at 2 m/sec

Project status

2 PhD projects have started at AIS (A. Arghir, M. Oubbati)

An ESN robot model

Training setup



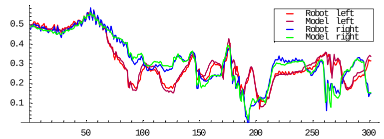
Training input: left and right PWMs of robot trace

100 unit ESN with output feedback

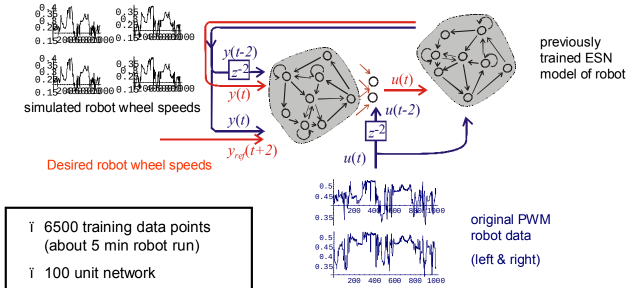
Training teacher: left and right measured wheel velocities

Results

- accurate dynamical robot model
- NMSE 0.027



Training and using the controller



- 6500 training data points (about 5 min robot run)
- 100 unit network
- Controller network with output feedback
- NMSE about 0.012

Results

Original controller

hand-designed PID (feedback) controller

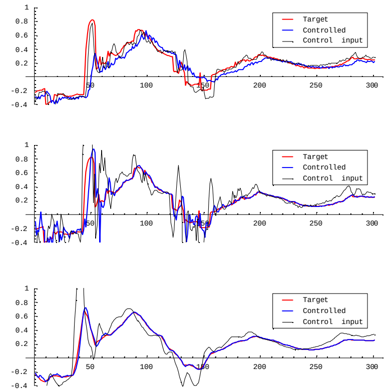
controlled trajectory lags behind target

Trained controller run on original targets

no lagging: controller incorporates prediction model

overshooting

Trained controller run on target generated by reference model



control: comments

mixture of direct and feedback control (*model-corrected dynamical state/observation feedback tracking control*)

"experiential" training data (playing, testing, just doing things)

generally learn how plant in state $y(t)$ will develop to state $y(t+d)$ due to $u(t)$

yields individual control style: jerky players will become high-gain controllers etc.

Neurocontrolling: the frontier

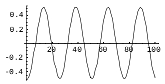
Jurgis Kairys, Vilnius, Lithuania, neurocontrolling his Sukhoi 26, 1999

4 Open Issues

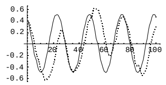
Stability

many a thing may happen when learning sines...

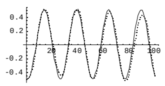
perfect



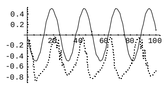
rather shaky



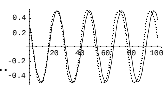
quite OK



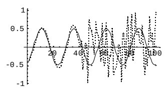
what's that?



well, yes, in principle...



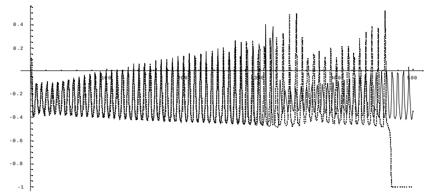
instable



... which of these qualifies as "learning the sine"?

Multiple time scales

This is hard to learn (Laser benchmark time series):

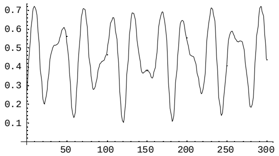


Reason: 2 widely separated time scales

Approach for future research: ESNs with different time constants in their units

Additive dynamics

This proved impossible to learn:



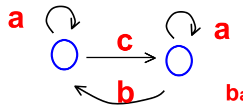
$$y(n) = \mathrm{Sin}(0.2n) + \mathrm{Sin}(0.311n)$$

Reason: requires 2 independent oscillators; but in ESN all dynamics are mutually coupled.

Approach for future research: modular ESNs and unsupervised multiple expert learning

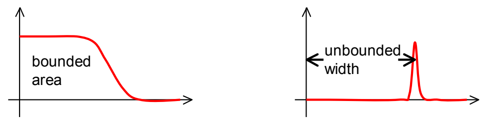
"Switching" memory

This FSA has long memory "switches":



baaa....aaa**C**aaa...aaa**b**aaa...aaa**C**aaa...aaa...

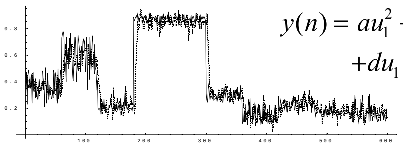
Generating such sequences not possible with monotonic, area-bounded forgetting curves!



Approach for future research: incorporate switching units into ESN

High-dimensional dynamics

Hight-dimensional dynamics would require very large ESN.
Example: 6-DOF nonstationary time series one-step prediction



$$y(n) = au_1^2 + bu_2^2 + cu_1u_2 + du_1 + eu_2 + f$$

200-unit ESN: RMS = 0.2; 400-unit network: RMS = 0.1; best other training technique[1)]: RMS = 0.02

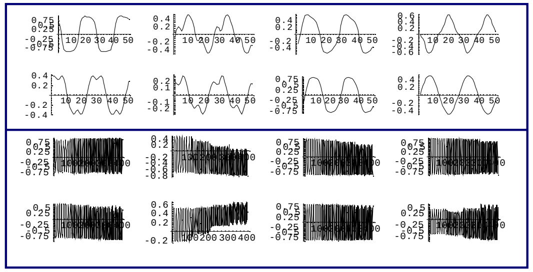
Approach for future research: task-specific optimization of ESN

[1)] *Prokhorov et al, extended Kalman filtering BPPT. Network size 40, 1400 trained links, training time 3 weeks*

5 Conclusion

Precision and versatility

Observation: a single echo state network can learn many dynamics with high precision.



Reason: echo states shape the tool for the solution from the task.

Patent

International patent application

- priority claimed October 13, 2000
- application filed October 2001 (PCT/EP01/11490).

Summary

- Basic idea: dynamical reservoir of echo states + supervised teaching of output connections.
- Seemed difficult: in nonlinear coupled systems, every variable interacts with every other. BUT seen the other way round, every variable *rules* and *echoes* every other. Exploit this for local learning and local system analysis.
- Echo states shape the tool for the solution from the task.

Long version of this talk and tutorial Mathematica notebooks can be downloaded from my homepage

http://www.gmd.de/People/Herbert.Jaeger/

Thank you.