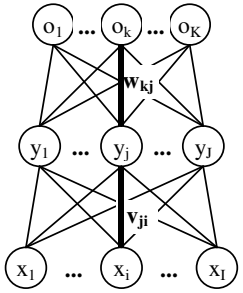
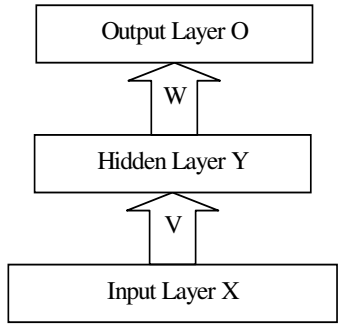
7. a 8. prednáška

Rekurentné neurónové siete

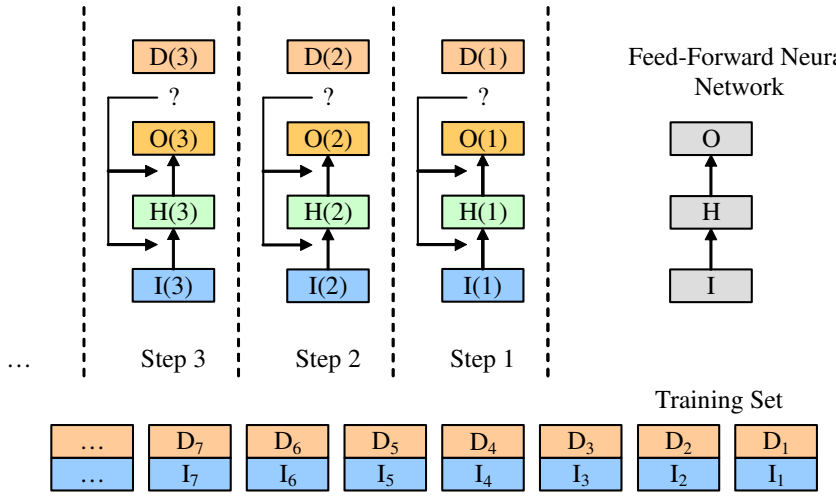
Dopredné neurónové siete (klasický viacvrstvový perceptrón)

- statické vstupno – výstupné mapovanie
- vstupné a výstupné vzory nezávisia od času
- potreba spracovávať dynamické úlohy
- nevyhnutnosť pamäte – dynamické neurónové siete





Algoritmus spätného šírenia chyby (Error Backpropagation)



Úlohy riešené rekurentnými sieťami

- klasifikácia s časovým kontextom
  - zaradenie časových postupností do kategórií

- predikcia (predpovedanie) či generovanie časovej postupnosti
  - predikcia nasledujúceho symbolu v čas. symbolickej postupnosti
  - predikcia nasledujúcej hodnoty v postupnosti reálnych hodnôt
  -

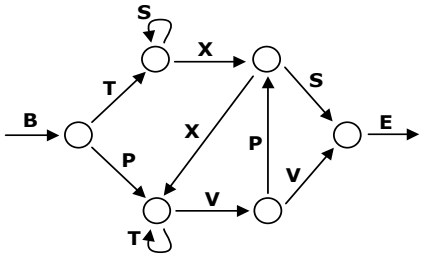
- modelovanie časovej postupnosti
  - zistenie, vytvorenie modelu generujúceho čas. postupnosť

- filtrovanie časovej postupnosti
  - zmena čas. postupnosti na inú, napr. odstránenie šumu,

Jednoduchá predikčná úloha č. 1

- postupnosť generovaná jednoduchým konečným stavovým automatom

a) Reber Grammar



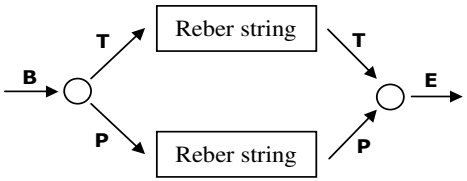
Reber

0 -> B1
1 -> T2 | P3
2 -> S2 | X4
3 -> T3 | V5
4 -> X3 | S6
5 -> P4 | V6
6 -> e

Extended Reber

7 -> B T 0 T E | B P 0 P E

b) Extended Reber Grammar



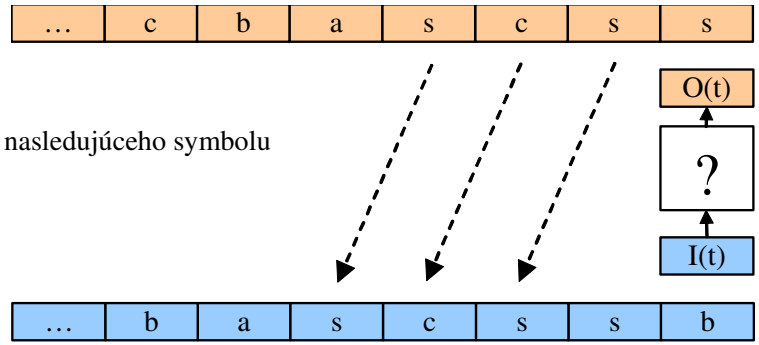
$a^n b^n c^n$

S->aSBC |abC
CB->BC
bB->bb
bC->bc
cC->cc

Jednoduchá predikčná úloha č. 2

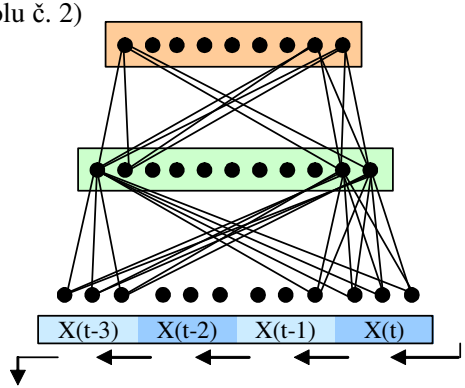
- postupnosť generovaná jednoduchou regulárnou gramatikou
  príklad postupnosti: bsscsabcssssssabsssscsssas….
  Gramatika:
  G = (A, {A,B,C}, {a,b,c,s}, P)
  P:
  A→sA | bB
  B→sB | cC
  C→sC | aA

- úloha: predikcia nasledujúceho symbolu



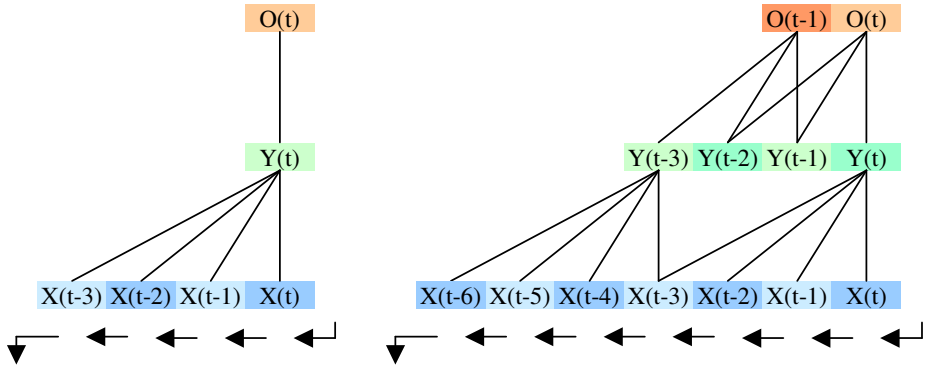
Time delay neural network

- potreba pamäte riešená rozšírením vstupu do minulosti
- okno do minulosti – počet vstupov z predošlých krokov prezentované sieti
- možnosť použiť algoritmus spätného šírenia chyby
- potreba určiť veľkosť okna
- nevhodná reprezentácia časového kontextu pre niektoré úlohy
  (úloha predikcie nasledujúceho symbolu č. 2)



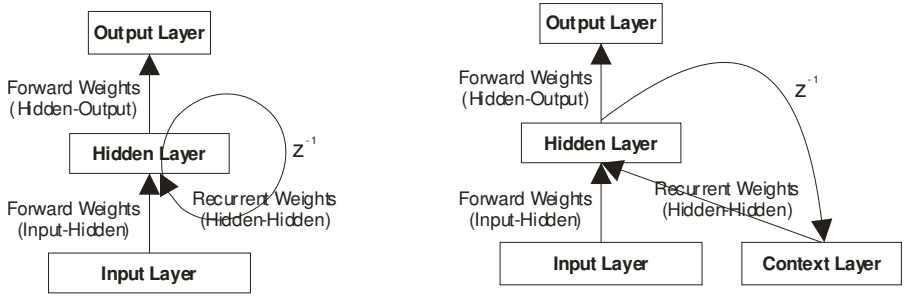
Time delay neural network

- jednoduchá TDNN – rozšírený iba vstup
- rozšírená TDNN – rozšírená aj skrytá vrstva, trénovanie sád váhových prepojení



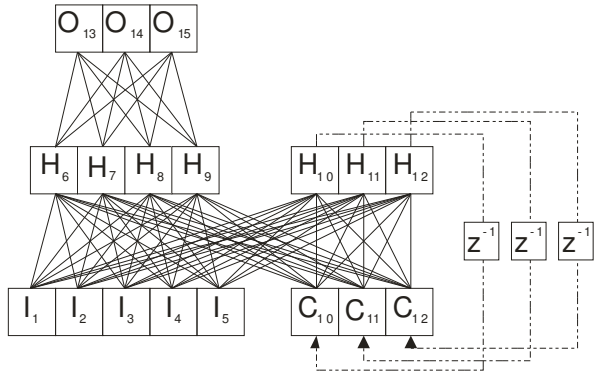
Rekurentná neurónová sieť (Recurrent multilayer perceptron)

- rozšírenie dopredných viacvrstvových perceptrónových sietí
- pamäť realizovaná váhovými prepojeniami s časovým oneskorením
- aktivity niektorých neurónov reprezentujú časový kontext
- kontextová vrstva
- sieť sa musí naučiť, čo reprezentovať



Tiňova architektúra

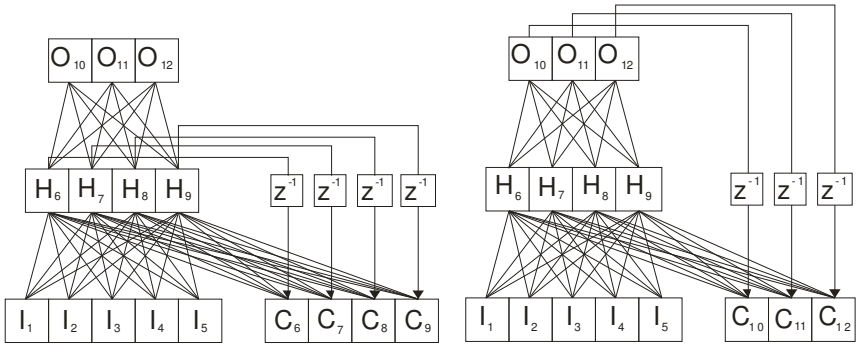
- oddelená stavová a asociačná časť siete
- kontextová vrstva udržiava stav siete = časový kontext



Tiňo

Elmanova architektúra (Elman's Simple Recurrent Network)
- často používaná, pôvodne trénovaná algoritmom backpropagation
- skrytá vrstva tvorí aj stavovú vrstvu

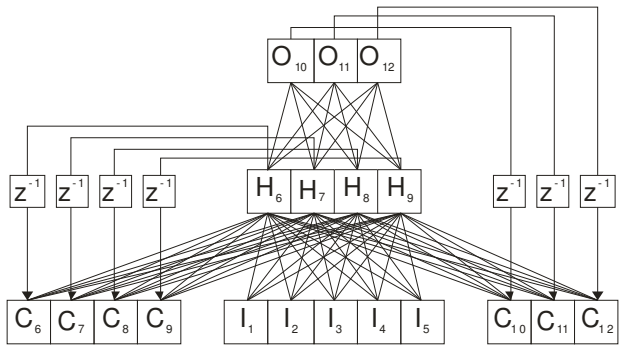
Jordanova architektúra
- výstupná vrstva tvorí aj stavovú vrstvu



Elman Jordan

Bengiova architektúra

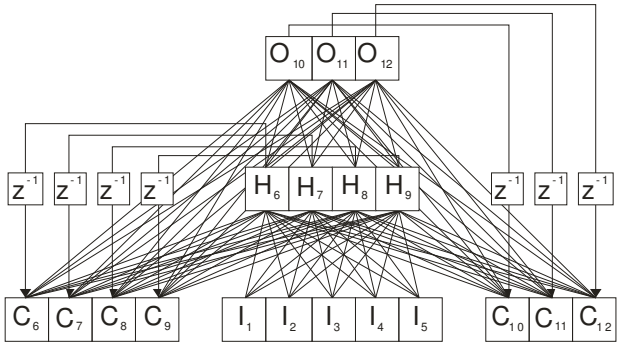
- skrytá vrstva aj výstupná vrstva tvoria stav pre rekurentnú neurónovú sieť



Bengio

Architektúra Williamsa a Zipser

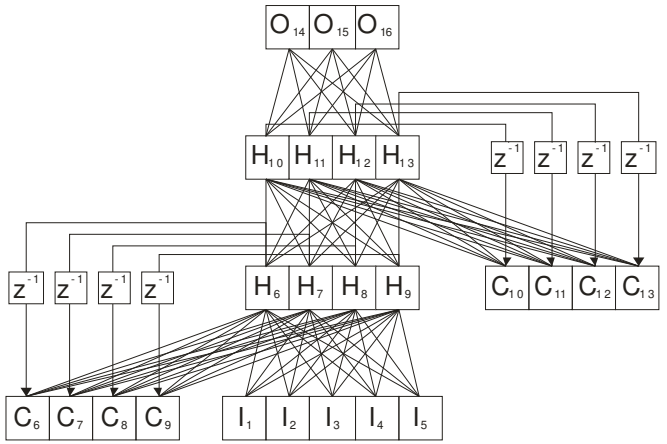
- kompletne poprepájané, aj výstupné neuróny



Williams a Zipser

Architektúra 5-4N-4N-3L

- rozšírenie Elmanovej siete o druhú skrytú/kontextovú vrstvu
- praktické aplikácie, trenovanie rozšíreným Kalmanovým filtrom,



5-4N-4N-3L

Trénovacie algoritmy

Spätné šírenie chyby (Backpropagation, ako napr. Elman)
- kontextová vrstva spolu so vstupnou tvoria rozšírený vstup
- jednoduchý a priamočiary prístup – spätné šírenie chýb
- chybový signál sa nešíri cez rekurentné váhy
- nevhodné pre niektoré (väčšinu) úlohy

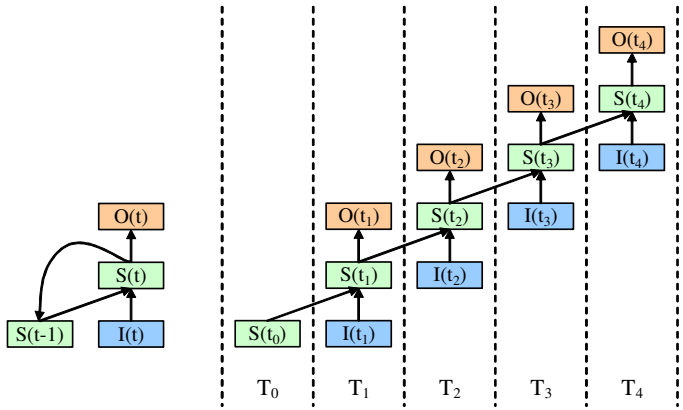
Spätné šírenie chyby v čase
- Backpropagation through time (BPTT)
- Werbos

Rekurentné učenie v reálnom čase
- Real time recurrent learning (RTRL)
- Williams and Zipser

Prístupy založené na Kalmanovej filtrácií
- Puskorius, Feldkamp, Prokhorov

Spätné šírenie chyby v čase – rozvinutie siete

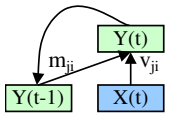
- rekurentná sieť môže byť rozvinutá v čase
- trénovanie rozvinutej siete algoritmom spätného šírenia chyby

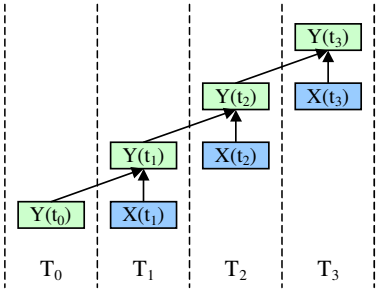


Spätné šírenie chyby v čase – vzťahy

$$\tilde{y}_j(t) = \sum_{i=1}^{I} v_{ji} x_i(t) + \sum_{i=1}^{J} m_{ji} y_i(t-1)$$

$$y_j(t) = f\left(\tilde{y}_j(t)\right)$$





$$\delta_k(t) = f'\left(\tilde{y}_k(t)\right)\left[\left(d_k(t) - y_k(t)\right) + \sum_{l=1}^{J} m_{lk}\delta_l(t+1)\right]$$

$$\Delta m_{ji}(t) = \alpha \delta_j y_i(t-1)$$

Priesvitka 17

Spätné šírenie chyby v čase – postup

- možnosť rozvíjať sieť pre každý vstup - vstupy konečnej a „krátkej“ dĺžky

- rozvíjať sieť pre každý vstup až do času 0 a spätne šíriť chybový signál z výstupnej vrstvy

- rozvíjať sieť pre každý vstup až do času t-H a spätne šíriť chybový signál z výstupnej vrstvy (Truncated Backpropagation Through Time)
- potreba určiť veľkosť okna spätného šírenia, typicky 10 až 30,40
- problém chyby miznúceho gradientu (Vanishing gradient problem)
  - gradientová informácia šírená v čase sa stráca (alebo naopak rastie do veľkých hodnôt)
  - problematické trénovanie dlhých časových závislostí
  - nemá zmysel šíriť chybový signál príliš hlboko do minulosti
- hybridné prístupy spolu s rekurentným učením v reálnom čase (Williams a Zipser)

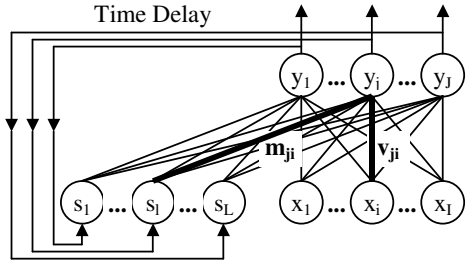
Priesvitka 18

Rekurentné učenie v reálnom čase (RTRL)

- výpočet gradientu on-line, v reálnom čase
- nie je potrebné určovať hĺbku časového okna
- výpočtvo náročné (v porovnaní s BPTT)

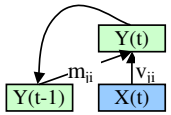
$$y_j(t) = f\left(\tilde{y}_j(t)\right)$$

$$\tilde{y}_j(t) = \sum_{i=1}^{I} v_{ji} x_i(t) + \sum_{i=1}^{J} m_{ji} y_i(t-1)$$



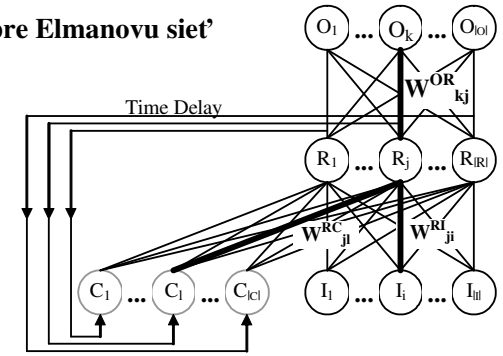
$$\Delta m_{ji}(t) = \alpha \sum_{k=1}^{J} \left(d_k(t) - y_k(t)\right) \frac{\partial y_k(t-1)}{\partial m_{ji}}$$

$$\frac{\partial y_k(t)}{\partial m_{ji}} = f'\left(\tilde{y}_k\right)\left[\sum_{l=1}^{J} m_{kl} \frac{\partial y_l(t-1)}{\partial m_{ji}} + \delta_{kj}^{kron} y_i(t-1)\right]$$



Priesvitka 19

Rekurentné učenie v reálnom čase pre Elmanovu sieť



$$\Delta W_{ji}^{RI} =$$

$$= \alpha \sum_{k}^{|O|} \left[(D_k^{(t)} - O_k^{(t)}) f'(\tilde{O}_k^{(t)}) \sum_{h=1}^{|R|} W_{kh}^{RC} \frac{\partial R_h^{(t)}}{\partial W_{ji}^{RI}} \right]$$

$$\frac{\partial R_h^{(t)}}{\partial W_{ji}^{RI}} = f'(\tilde{R}_i^{(t)}) \left[I_i^{(t)} \delta_{hj}^{kron} + \sum_{l=1}^{|R|} W_{hl}^{RC} \frac{\partial R_l^{(t-1)}}{\partial W_{ji}^{RI}} \right]$$

Priesvitka 20

Kalmanov filter (KF)

Popis systému pomocou lineárnych rovníc:

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \qquad p(w) \approx N(0,Q)$$
$$z_k = Hx_k + v_k \qquad p(v) \approx N(0,R)$$

Kalmanov filter:

$$\bar{x}_k = A\bar{x}_{k-1} + Bu_k$$
$$P_k^- = AP_{k-1}A^T + Q$$

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$
$$\bar{x}_k = \bar{x}_k^- + K_k(z_k - H\bar{x}_k^-)$$
$$P_k = (I - K_k H)P_k^-$$

Rozšírený Kalmanov filter (Extended Kalman Filter - EKF)

Nerónová sieť ako systém pre KF:

$$w_k = Iw_{k-1} \qquad p(w) \approx N(0,Q)$$
$$o_k = h(x_k, v_k) \qquad p(v) \approx N(0,R)$$

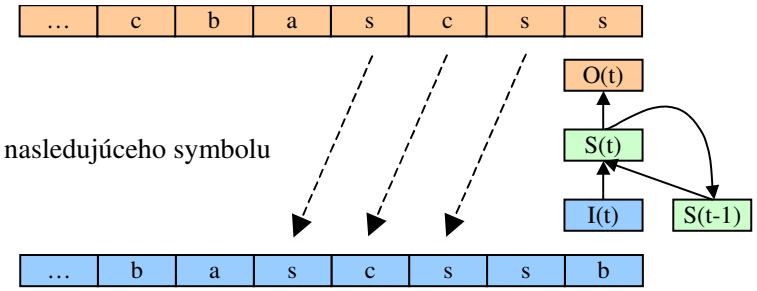
Určenie Jakobiánu – matice parcialných derivácií
- možnosť použiť BPTT alebo RTRL

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\bar{x}_k, 0)$$

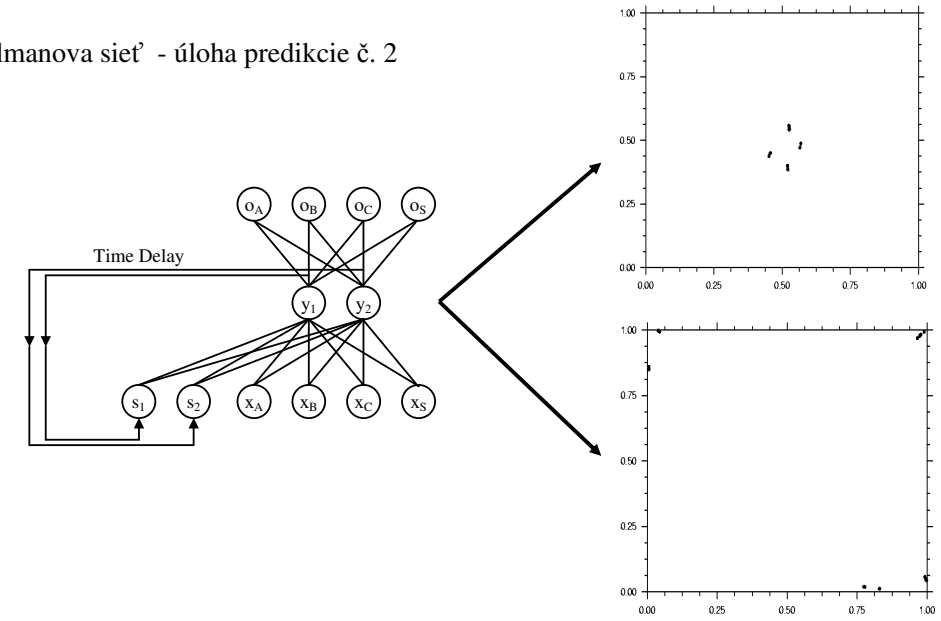
Stavová reprezentácia v rekurentných neurónových sieťach

- postupnosť generovaná jednoduchou regulárnou gramatikou
  príklad postupnosti: bsscsabcsssssabsssscsssas….
  Gramatika:
  G = (A, {A,B,C}, {a,b,c,s}, P)
  P:
  A→sA | bB
  B→sB | cC
  C→sC | aA

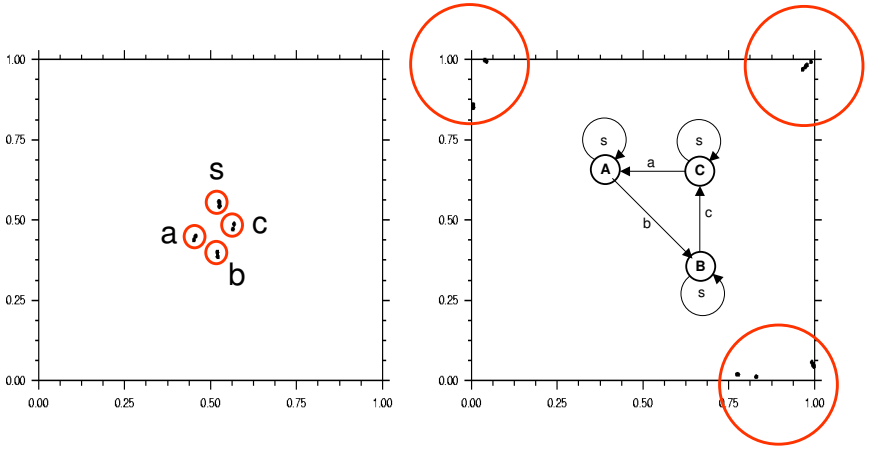
- úloha: predikcia nasledujúceho symbolu



Elmanova sieť - úloha predikcie č. 2

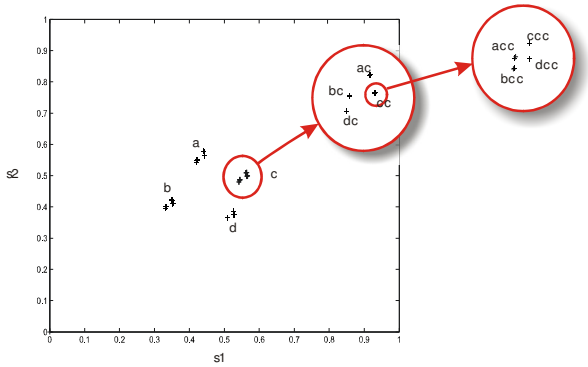


Elmanova sieť - úloha predikcie č. 2 - detail stavovej reprezentácie



Dynamika náhodne inicializovanej rekurentnej neurónovej siete

- stavový prpiestor siete vykazuje aj pred natrénovaním vysoký stupeň organizácie
- zkluky zodpovedajúce histórií symbolov prezentovaných sieti
- architekurálny bias (Markovovský architekturálny bias)
- fraktálová štruktúra stavového priestoru



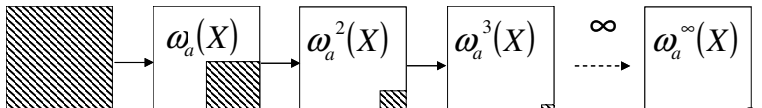
Iteračné funkčné systémy

IFS – množina transformácií definovaných nad priestorom

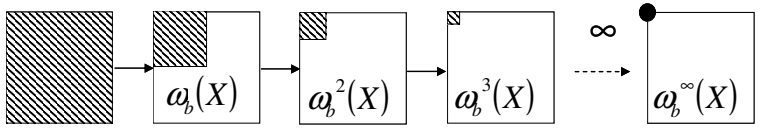
$$\Omega = \{\omega_i \mid \omega_i : X \to X, i \le n\}$$

$$\omega_1(x,y) = (0.5x + 0.5, 0.5y)$$
$$\omega_2(x,y) = (0.5x, 0.5y + 0.5)$$
$$\omega_3(x,y) = (0.5x, 0.5y)$$

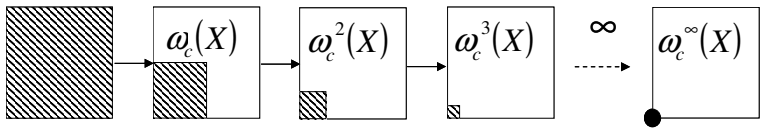
$\omega_a(x,y) =$
$(0.5x+0.5,0.5y)$



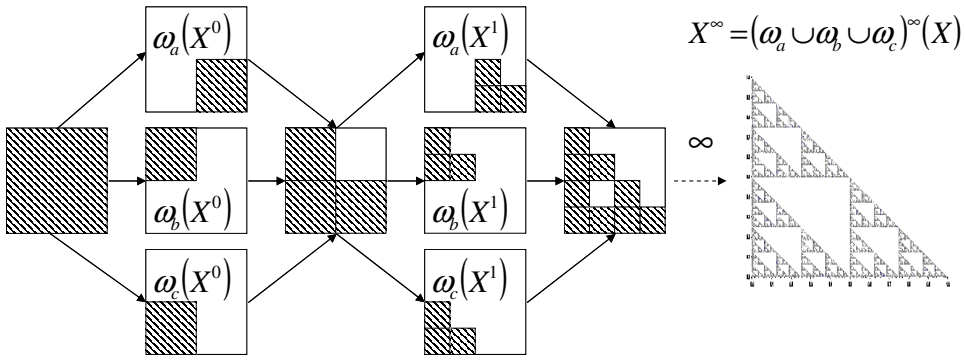
$\omega_b(x,y) =$
$(0.5x,0.5y+0.5)$



$\omega_c(x,y) =$
$(0.5x,0.5y)$

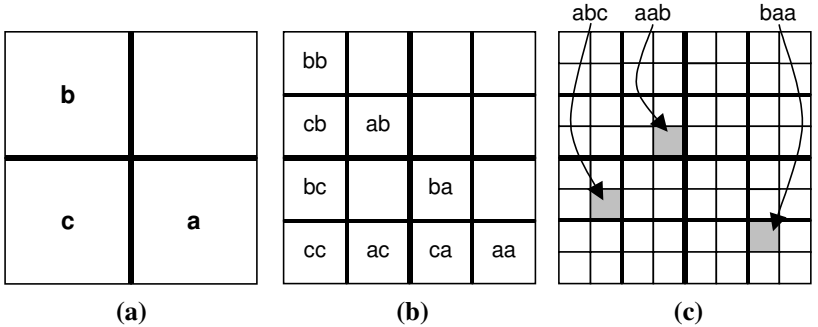


Iteračný funkčný systém atraktor zjednotenia transformacií



$X^0 = X$ $\quad X^1 = (\omega_a \cup \omega_b \cup \omega_c)(X^0)$ $\quad X^2 = (\omega_a \cup \omega_b \cup \omega_c)^2(X^1)$

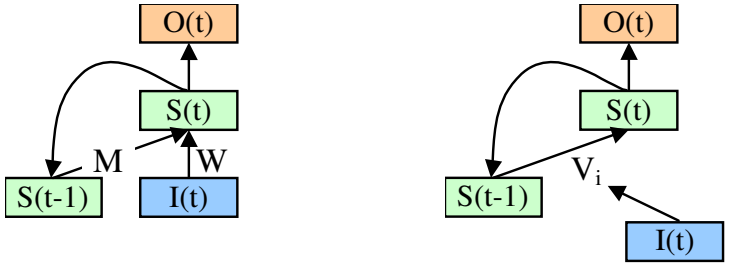
Iteračné funkčné systémy – adresa



Vzťah rekurentných sietí a iteračných funkčných systémov

RNN: $S^{(t+1)} = f(M \cdot S^{(t)} + W \cdot I^{(t)})$

IFS: $S^{(t+1)} = f(V_i \cdot S^{(t)})$

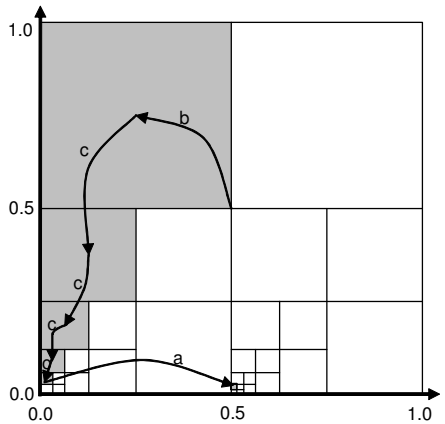


IFS dynamika v rekurentných neurónových sieťach

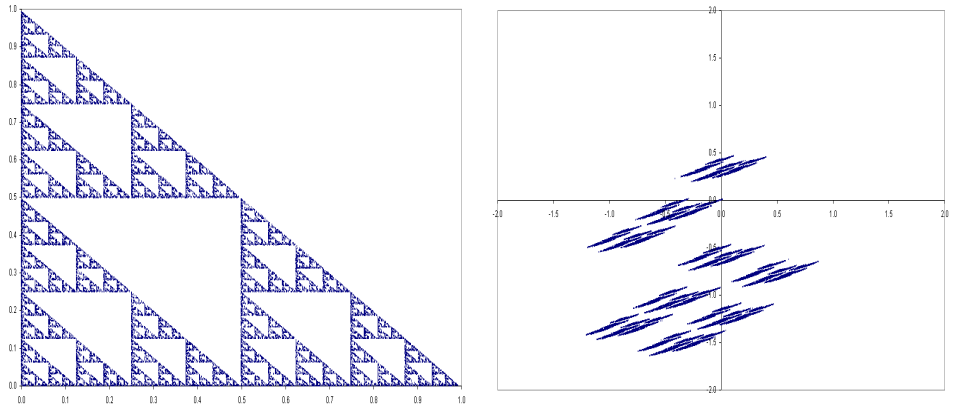
First 6 symbols of Laser sequence:
b c c c c a

Starting point is (0.5,0.5).

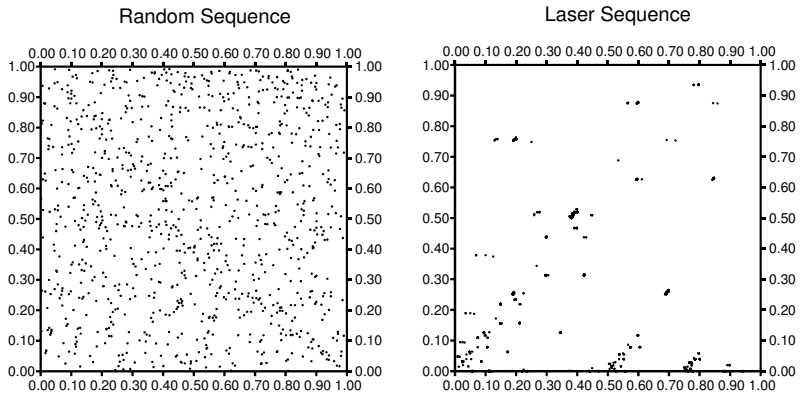
$\bar{x}_0 = (0.5,0.5)$
$\bar{x}_1 = \omega_b(\bar{x}_0) = (0.25,0.75)$
$\bar{x}_2 = \omega_c(\bar{x}_1) = (0.125,0.375)$
$\bar{x}_3 = \omega_c(\bar{x}_2) = (0.0625,0.1875)$
$\bar{x}_4 = \omega_c(\bar{x}_3) = (0.03125,0.09375)$
$\bar{x}_5 = \omega_c(\bar{x}_4) = (0.015625,0.046875)$
$\bar{x}_6 = \omega_a(\bar{x}_5) = (0.5078125,0.0234375)$



Atraktor náhodne inicializovanej rekurentnej neurónovej siete



Stavový priestor (IFS modelu RNS) po prezentácií postipnosti LASER



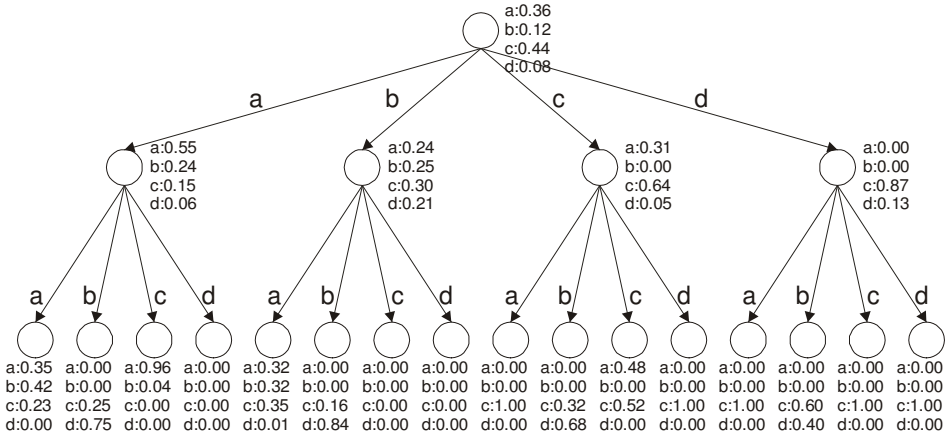
Vlastnosti náhodne inicializovaných sietí (Markovovský architekturálny bias)

- stav siete je v najväčšej miere určený posledným prezentovaným vstupom
- miera vplyvu vstupu zaniká s časom jeho prezentovania sieti

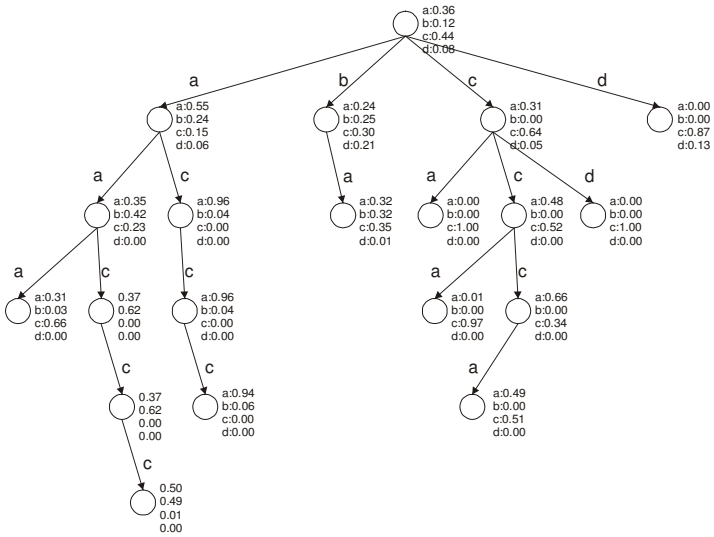
- ak sú sieti prezentované dve časové postupnosti, čím dlhší ich spoločný suffix, tým bližšie sa v stavovom priestore nachádzajú body zodpovedajúce postupnostiam

Markovovského predpoklad: $P(x_{t+1} \mid x_1..x_t) = P(x_{t+1} \mid x_{t-m+1}..x_t)$

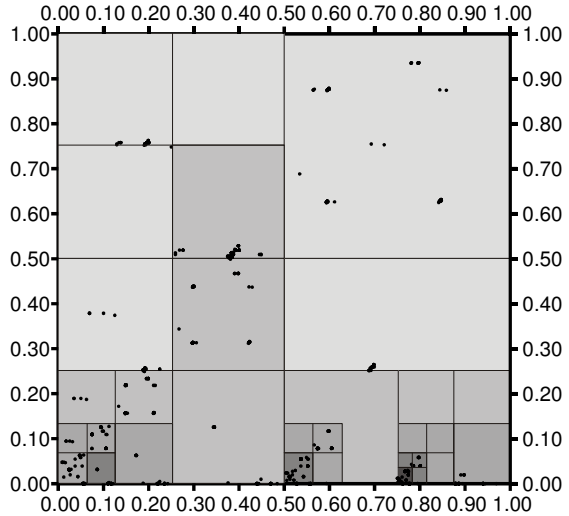
Markovov model s pevnov dĺžkou pamäte



Markovov model s premenlivou dĺžkou pamäte



Stavový priestor (IFS modelu RNS) po prezentácií postipnosti LASER



Spracovanie bezkontextového jazyka

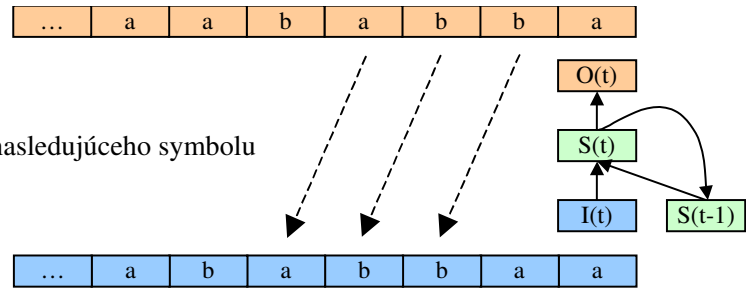
- potreba mechanizmu počítania
- rekurentná sieť môže byť natrénovaná na spracovanie jednoduchého bezkontextoveho alebo kontextového jazyka

Stavová reprezentácia v rekurentných neurónových sieťach

- postupnosť generovaná jednoduchou bezkontextovou gramatikou
  jazyk $a^n b^n$

príklad reťazcov : ab,aabb,aaabbb,aaaabbbb,…

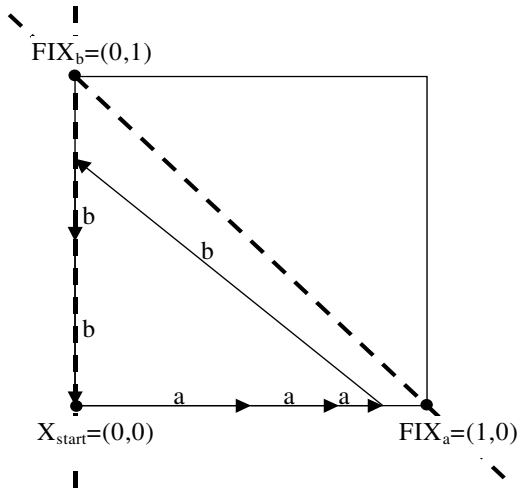
Gramatika:
G = (S, {S}, {a,b }, P)
P:
A→aSb | ab



- úloha: predikcia nasledujúceho symbolu

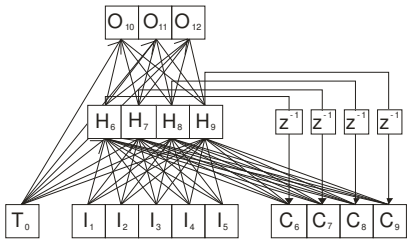
Zjednodušená reprezentácia stavového priestoru siete

$\omega_a(x_1, x_2) = f(0.5x_1 + 0.5,\ 0)$ $\omega_b(x_1, x_2) = f(0, 2x_1 + 2x_2 - 1.0)$

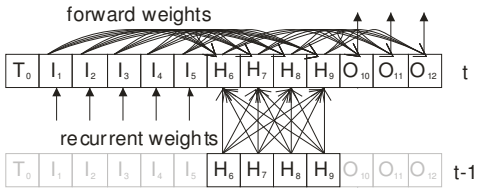


Elman's SRN – Werbos Representation

a) Elman's SRN - Layered Structure



b) Elman's SRN - Werbos Representation



Elman's SRN – Werbos Representation

c) Weight Connections

Index	Source	Dest.	Delay	Value
WI	SRC	DST	TD	W
0	0	6	0	
1	1	6	0	
2	2	6	0	
3	3	6	0	
4	4	6	0	
5	5	6	0	
6	6	6	1	
7	7	6	1	
8	8	6	1	
9	9	6	1	
10	0	7	0	
11	1	7	0	
12	2	7	0	
13	3	7	0	
14	4	7	0	
15	5	7	0	
16	6	7	1	
17	7	7	1	
18	8	7	1	
19	9	7	1	

Index	Source	Dest.	Delay	Value
WI	SRC	DST	TD	W
20	0	8	0	
21	1	8	0	
22	2	8	0	
23	3	8	0	
24	4	8	0	
25	5	8	0	
26	6	8	1	
27	7	8	1	
28	8	8	1	
29	9	8	1	
30	0	9	0	
31	1	9	0	
32	2	9	0	
33	3	9	0	
34	4	9	0	
35	5	9	0	
36	6	9	1	
37	7	9	1	
38	8	9	1	
39	9	9	1	

Index	Source	Dest.	Delay	Value
WI	SRC	DST	TD	W
40	0	10	0	
41	6	10	0	
42	7	10	0	
43	8	10	0	
44	9	10	0	
45	0	11	0	
46	6	11	0	
47	7	11	0	
48	8	11	0	
49	9	11	0	
50	0	12	0	
51	6	12	0	
52	7	12	0	
53	8	12	0	
54	9	12	0	

d) Units

Index	First W.	Last W.	Type	Act. F.
UI	FW	LW	UT	AF
0			T	
1			I	
2			I	
3			I	
4			I	
5			I	
6	0	9	H	SGM
7	10	19	H	SGM
8	20	29	H	SGM
9	30	39	H	SGM
10	40	44	O	LIN
11	45	49	O	LIN
12	50	54	O	LIN

Algorithmic description

`NW`	- number of weight connections
`wSource[0..NW-1]`	- source node indices
`wDest[0..NW-1]`	- destination node indices
`wDelay[0..NW-1]`	- connection time delays
`wValue[0..NW-1]`	- weight connection strengths
`NU`	- number of units (special threshold unit and input units also count)
`uFirstWeight[0..NU-1]`	- indices of the first weight connection associated with the unit
`uLastWeight[0..NU-1]`	- indices of the last weight connection associated with the unit
`uType[0..NU-1]`	- unit types (`THRESHOLD, INPUT, HIDDEN, OUTPUT`)
`NSTEPS`	- number of time steps
`ACT[0..NU-1,0..NSTEPS-1]`	- all unit activities in all time steps
`ACTD[0..NU-1,0..NSTEPS-1]`	- activation function derivatives for all units in all time steps
`Sgm(iact)`	- activation function, input is iact – internal unit's activity
`SgmDer(iact)`	- derivative of the activation function
`Input(ui,ts)`	- returns network inputs corresponding to the input unit ui in time step ts
`Output(ui,act,ts)`	- set network output to the calc. activity act of the output unit ui in time step ts
`Target(ui,ts)`	- returns desired output unit ui activity in time step ts
`ts`	-actual time step

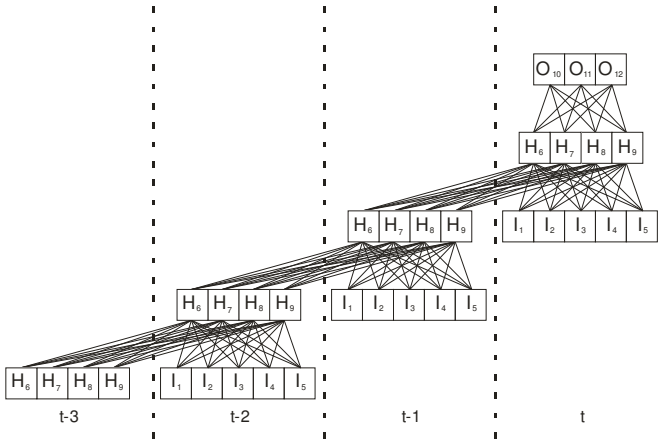
Forward pass

```
for ui=0 to NU-1 do
begin
  if uType[ui] = TRASHOLD then ACT[ui,ts] := 1.0;
  else if uType[ui] = INPUT then ACT[ui,ts] := Input(ui,ts);
  else
  begin
    iact := 0.0;
    for wi := uFirstWeight[ui] to uLastWeight[ui] do
      iact := iact + wValue[wi]*ACT[wSource[wi],ts-wDelay[wi]];
    ACT[ui,ts] := Sgm(iact);
    ACTD[ui,ts] := SgmDer(iact);
  end;
  if uType = OUTPUT then Output(ui,ACT[ui,ts],ts);
end;
```

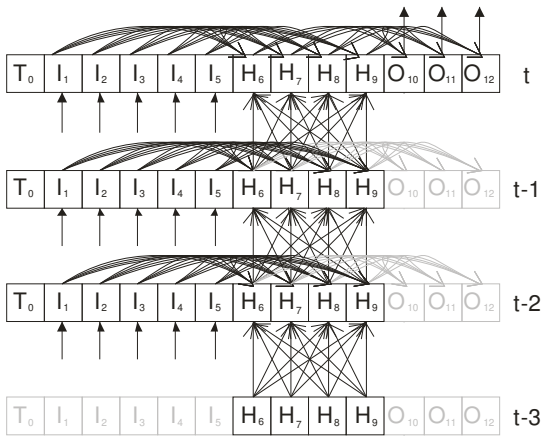
Backpropagation algorithm

```
DE_DNA[0..NU-1]   - backpropagated error signal
DLT_W[0..NW-1]    - calculated weight changes
beta              - momentum


for ui=NU-1 downto 0 do DE_DNA[ui] := 0.0;

for ui=NU-1 downto 0 do
begin
  if uType[ui] = INPUT then break;
  if uType[ui] = OUTPUT then DE_DNA[ui]:=DE_DNA[ui] + (Target(ui,ts)-ACT[ui,ts]);
  DE_DNA[ui] := DE_DNA[ui]*ACTD[ui,ts];
  for wi := uLastWeight[ui] downto uFirstWeight[ui] do
    if uType[wSource[wi]] <> INPUT AND wDelay[wi] = 0 then
      DE_DNA[wSource[wi]]:=DE_DNA[wSource[wi]] + wValue[wi]*DE_DNA[ui];
  end;
end;

for wi:=0 to NW-1 do
begin
  DLT_W[wi] := beta*DLT_W[wi] + alfa*DE_DNA[wDest[wi]]*ACT[wSource[wi],ts-wDelay[wi]];
  wValue[wi] := wValue[wi] + DLT_W[wi];
end;
```


Elman's SRN Unfolded in Time



Elman's SRN Unfolded in Time – Werbos Representation



Backpropagation Through Time

```
DE_DNA[0..NU-1,0..winSize-1]  - backpropagated error signal
winSize                       - size of window

for hi=0 to winSize-1 do
 for ui=NU-1 downto 0 do
  DE_DNA[ui,hi] := 0.0;

for hi=0 to winSize-1 do
 for ui=NU-1 downto 0 do
 begin
  if uType[ui] = INPUT then break;
  if (uType[ui] = OUTPUT) AND (hi = 0) then
   DE_DNA[ui,hi]:=DE_DNA[ui,hi] + (Target(ui,ts)-ACT[ui,ts]);
  DE_DNA[ui,hi] := DE_DNA[ui,hi]*ACTD[ui,ts-hi];
  for wi := uLastWeight[ui] downto uFirstWeight[ui] do
  begin
   if (uType[wSource[wi]] <> INPUT) AND (wDelay[wi]+hi < winSize)) then
    DE_DNA[wSource[wi],wDelay[wi]+hi]:=
     DE_DNA[wSource[wi],wDelay[wi]+hi] + wValue[wi]*DE_DNA[ui,hi];
   DLT_W[wi] := DLT_W[wi] + alfa*DE_DNA[ui,hi]*ACT[wSource[wi],ts-hi-wDelay[wi]];
  end;
 end;

for wi:=0 to NW-1 do
begin
 wValue[wi] := wValue[wi] + DLT_W[wi];
 DLT_W[wi] := beta*DLT_W[wi];
end;
```

Real Time Recurrent Learning

```
DA_DW[0..NU-1,0..NW-1,0..NSTEPS-1]    - forwardly propagated derivatives

for wi:=0 to NW-1 do
  for ui:=0 to NU-1 do
  begin
    if uType <> INPUT then
      for uwi:=uFirstWeight[ui] to uLastWeight[ui] do
        if uType[wSource[uwi]] <> INPUT then
          DA_DW[ui,wi,ts] :=
            DA_DW[ui,wi,ts] + wValue[uwi]*DA_DW[wSource[uwi],wi,ts-wDelay[uwi]];

    if wDest[wi] = ui then
      DA_DW[ui,wi,ts] := DA_DW[ui,wi,ts] + ACT[wSource[wi],ts-wDelay[wi]];
    DA_DW[ui,wi,ts] := ACTD(ui,ts)*DA_DW[ui,wi,ts];
  end;

for wi:=0 to NW-1 do
begin
  DLT_W[wi] := beta*DLT_W[wi];
  for ui:=0 to NU-1 do
    if uType = OUTPUT then
      DLT_W[wi] := DLT_W[wi] + alfa*(Target(ui,ts)-ACT[ui,ts])*DA_DW[ui,wi,ts];
  wValue[wi] = wValue[wi] + DLT_W[wi];
end;
```