



# Implementace plně rekurentní neuronové sítě v systému *Mathematica*

Zdeněk Buk, Miroslav Šnorek

{bukz1 | snorek}@fel.cvut.cz

Neural Computing Group

Department of Computer Science and Engineering,

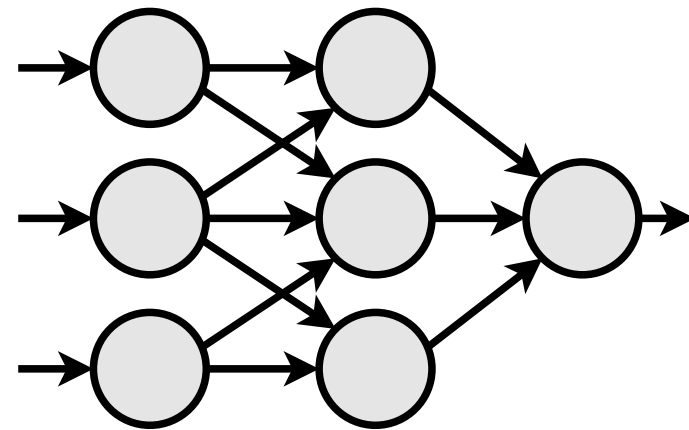
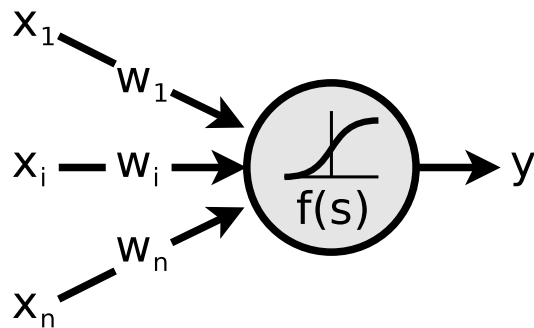
Faculty of electrical engineering,

Czech Technical University, Prague

- Motivace
- Neuronové sítě - stručný úvod
- Knihovna *Neural Networks*
- Rekurentní neuronové sítě
- Experimenty
- Závěr
- Reference

- Proč neuronové sítě?
  - Adaptabilní robustní struktury,
  - schopné generalizace.
- Proč rekurentní neuronové sítě?
  - Schopnost práce s časovým kontextem dat.
- Proč *Mathematica*?
  - Univerzální jednotné vývojové prostředí,
  - snadné začlenění do dalších výpočtů,
  - rozsáhlé možnosti exportu a importu dat,
  - silný vizualizační aparát.

- Struktury inspirované svými biologickými vzory.
- Využití na těžko algoritmizovatelné problémy.
- Schopnost adaptace a učení.
- Odolnost vůči poruchám.



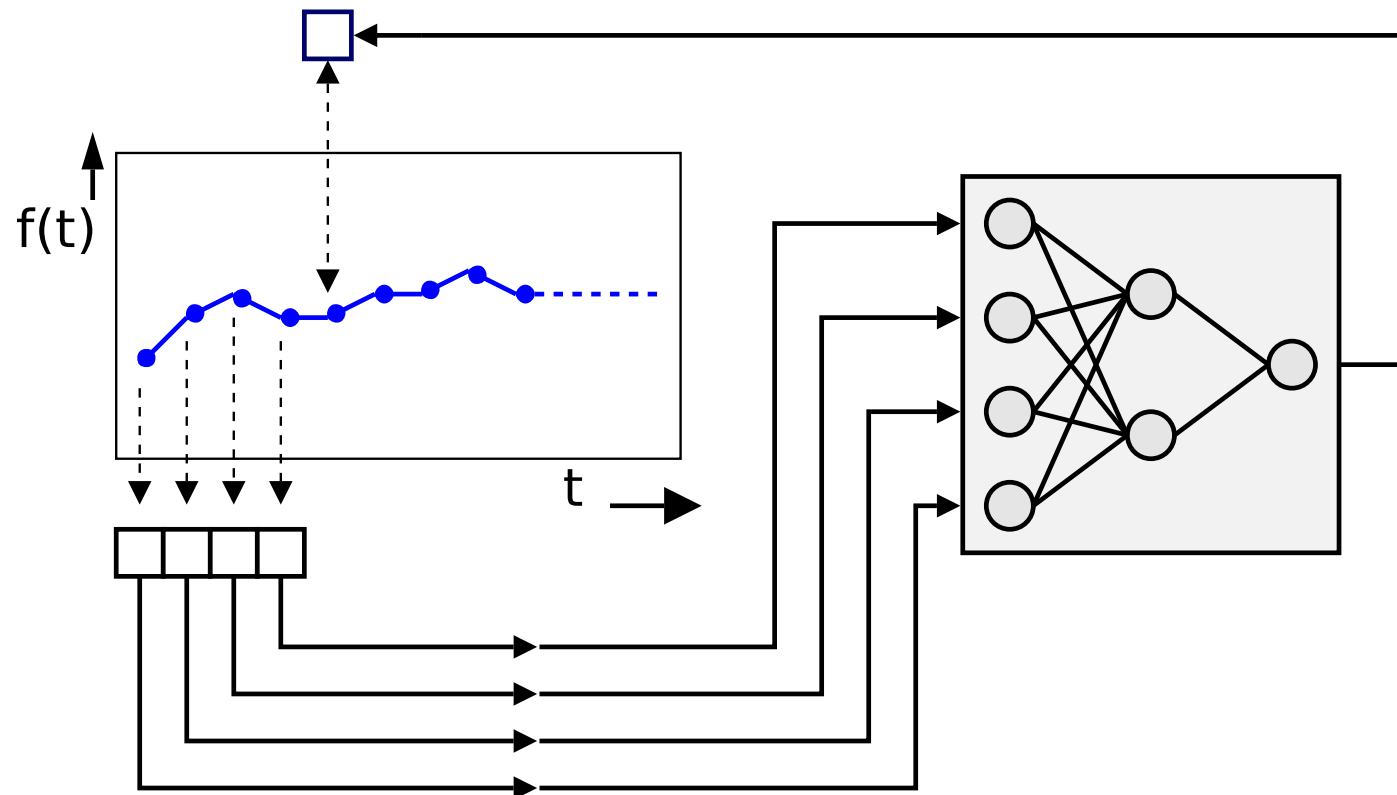
$$s = \sum_{i=0}^n w_i x_i \quad f(s) = \frac{1}{(1+e^{-\lambda s})}$$



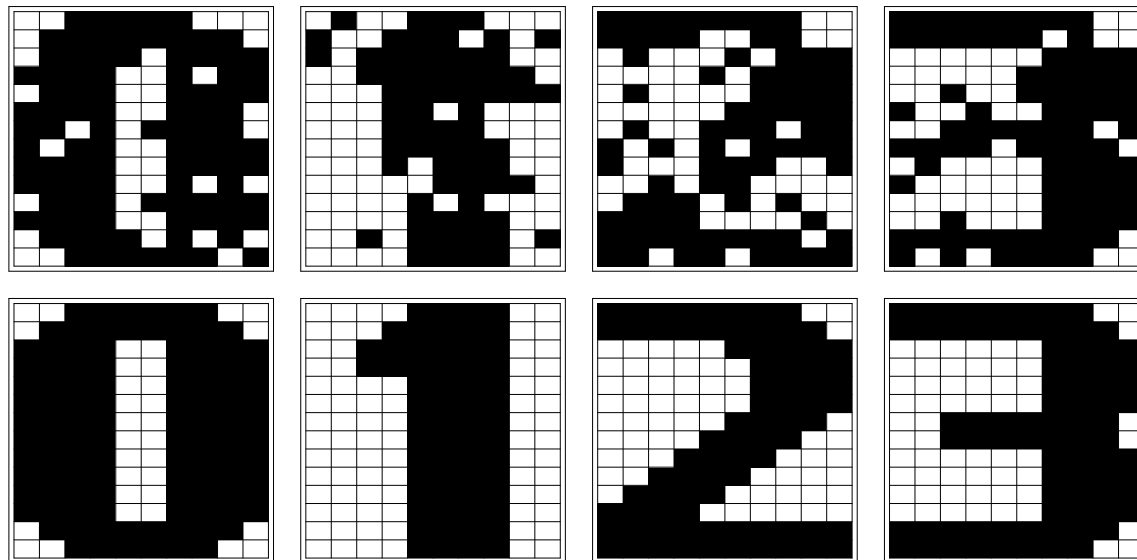
# Aplikace neuronových sítí

- Predikce,
- aproximace,
- rozpoznávání (klasifikace),
- filtrace,
- optimalizace,
- shlukové analýzy, atd.

- Předpovídání výstupní hodnoty časové řady na základě jejího předchozího průběhu (historii).

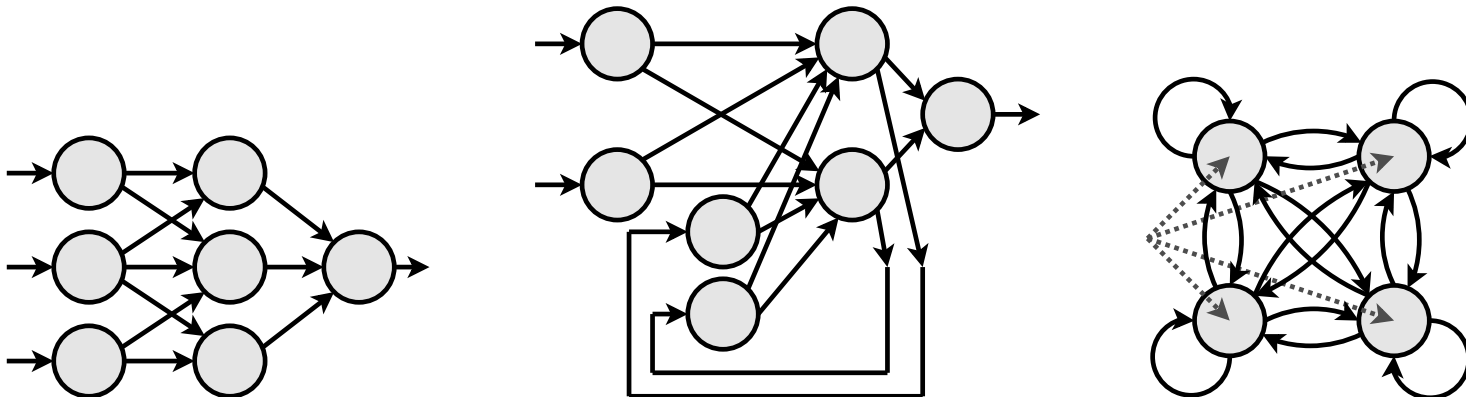


- Rozhodování na základě vstupního vektoru o tom, do které třídy předmět, popsany daným vektorem, patří.
- Rozpoznávání hlasu, písma, obrazů, apod.



# Dělení neuronových sítí

- Podle způsobu učení:
  - učení s učitelem,
  - učení bez učitele (samoorganizace).
- Podle způsobu propojení – topologie:
  - dopředné sítě,
  - rekurentní sítě.





# Knihovna *Neural Networks*

Podporované neuronové sítě:

- Feedforward
- RBF (Radial Basis Function)
- Dynamic
- Hopfield
- Perceptron
- Unsupervised (self-organizing maps)
- Vector Quantization (LVQ)



# Knihovna *Neural Networks*

## Dopředná síť – příklad

```
<< NeuralNetworks` (* Načtení knihovny *)
x = Table[...]; (* Trénovací množina *)
y = Table[...];

(* Vytvoření sítě se čtyřmi neurony *)
net = InitializeFeedForwardNet[x, y,
  {4}, RandomInitialization -> True];
(* Učení sítě, 10 trénovacích kroků *)
{net2, rec} = NeuralFit[net, x, y, 10];
(* Vybavování sítě *)

net2[ {3.1} ]
net2[ {1.2, 0.7, 0.1} ]
```

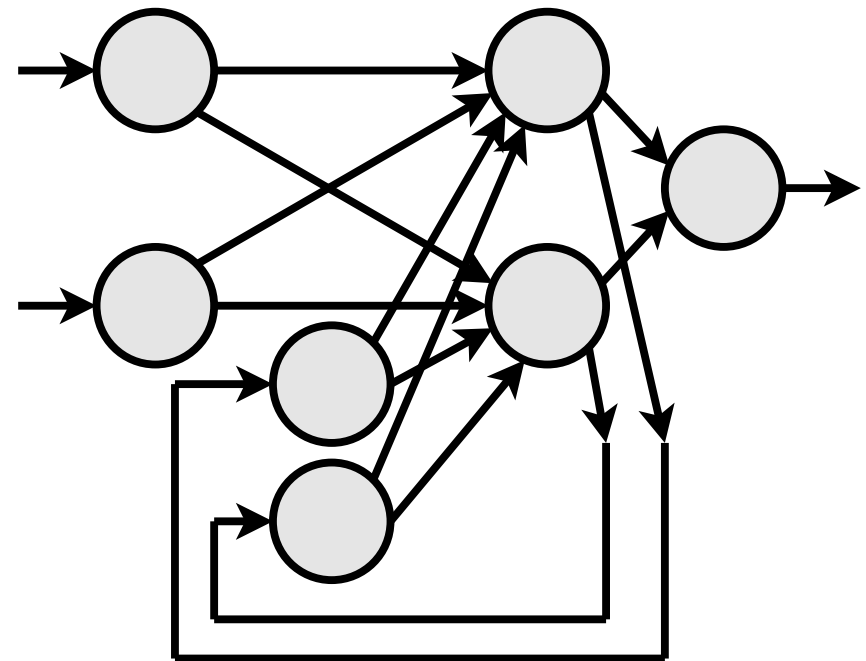
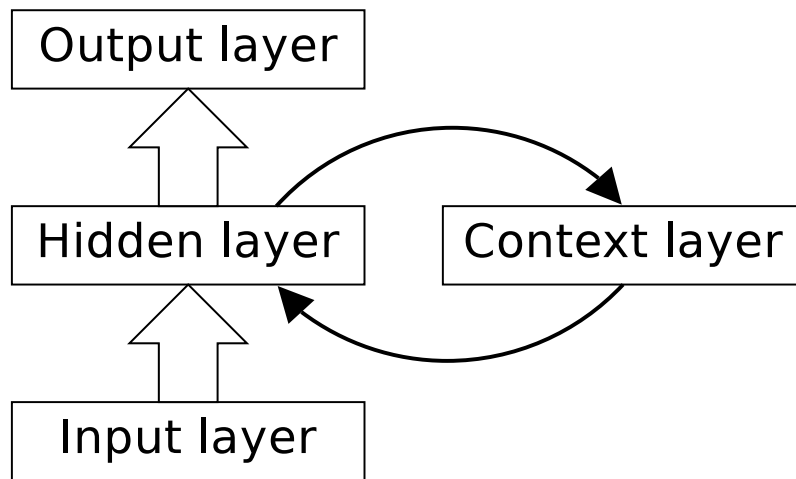


# Rekurentní neuronové sítě

- Omezení dopředných sítí
  - limitované možnosti zpracování časového kontextu vstupních dat,
  - nutnost doplnit “externí paměť” (např. tzv. okénková metoda)
- Řešení: rekurentní neuronová síť
  - obsahuje zpětné (rekurentní) vazby.

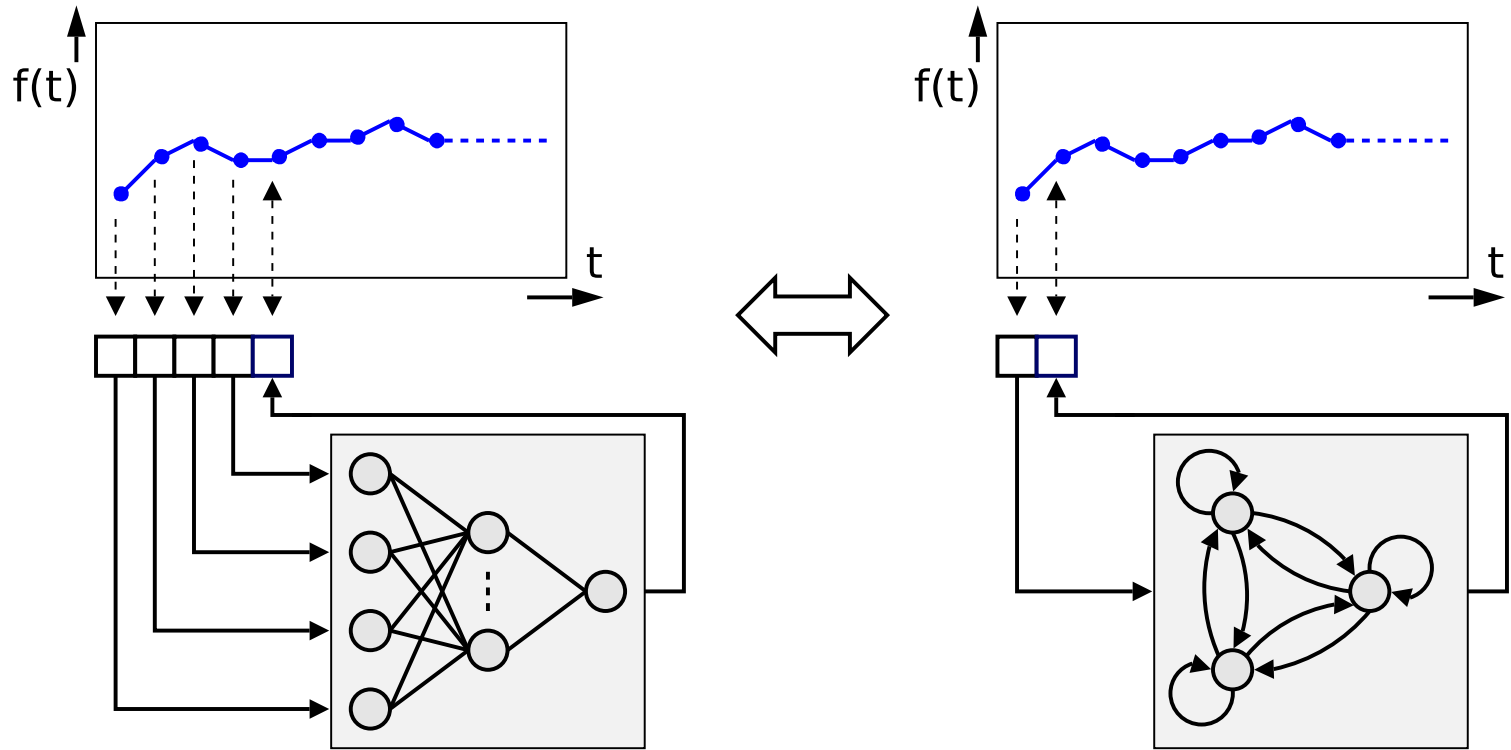
# Rekurentní neuronové sítě

- Jordanova-Elmanova neuronová síť:



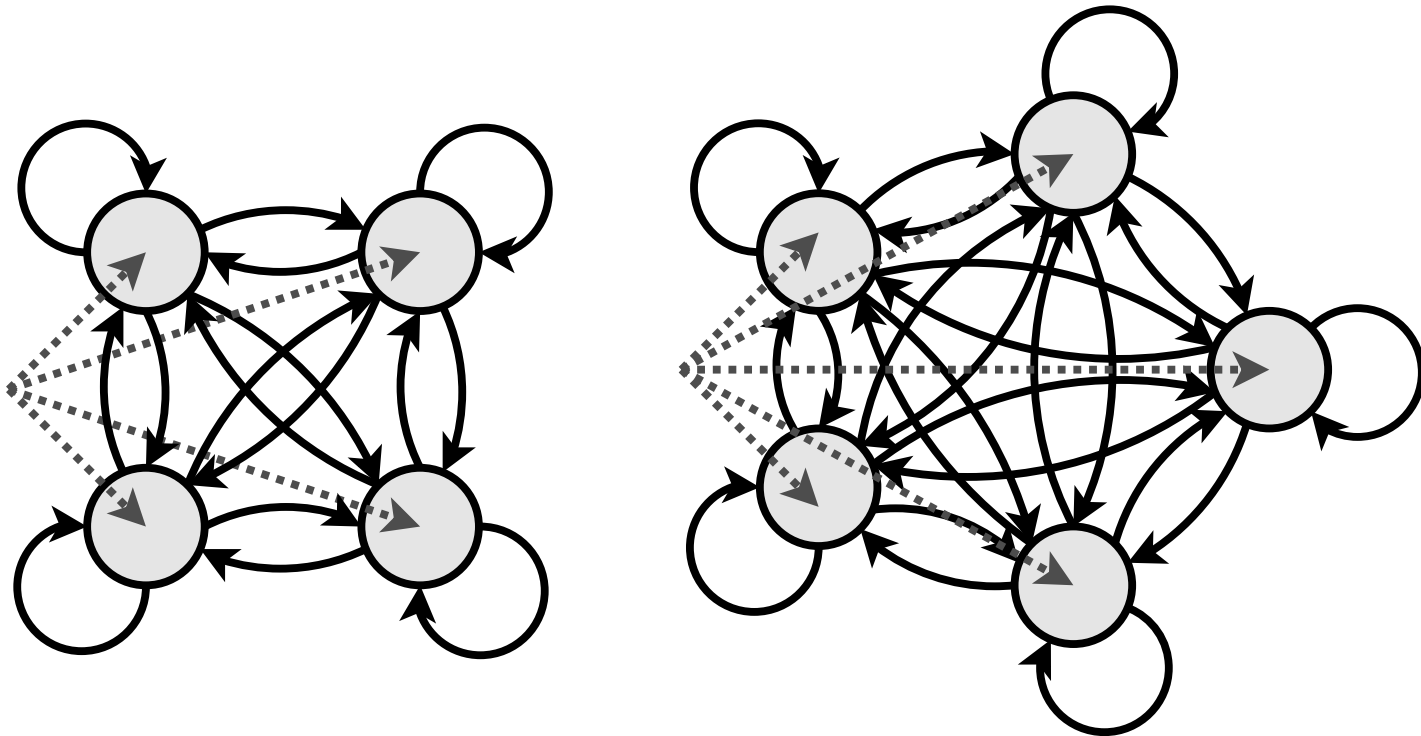
# Rekurentní vs. dopředné sítě

- Příklad – predikce časové řady:



# Plně rekurentní neuronová síť

- Nejobecnější struktura rekurentní sítě,
- reprezentovaná úplným symetrickým grafem.



# NGG Implementace - algoritmus

- Realtime recurrent learning,
- gradientní učicí metoda,
- temporal supervised learning.

- Hodnoty vstupů a výstupů sítě

$$z_k(t) = \begin{cases} x_k(t) & \text{pro } k \in I \text{ (množina vstupů)} \\ y_k(t) & \text{pro } k \in U \text{ (množina výstupů)} \end{cases}$$

- Výpočet vnitřního potenciálu a výstupu neuronů

$$s_k(t) = \sum_{l \in U \cup I} w_{kl} z_l(t), \quad y_k(t+1) = f_k(s_k(t)), \quad k \in U$$

- Chyba sítě

$$e_k(t) = \begin{cases} d_k(t) - y_k(t) & \text{jestliže } k \in T(t) \\ 0 & \text{jinak} \end{cases}$$

$$J(t) = \frac{1}{2} \sum_{k \in U} [e_k(t)]^2$$



# Implementace - algoritmus

- Korekce hodnoty vah

$$\Delta w_{ij} = \sum_{t=t_0+1}^{t_1} \Delta w_{ij}(t) , \quad \Delta w_{ij}(t) = -\alpha \frac{\partial J(t)}{\partial w_{ij}}$$

- Popis dynamiky systému

$$p_{ij}^k(t+1) = f'_k(s_k(t)) \left[ \sum_{l \in U} w_{kl} p_{ij}^l(t) + \delta_{ij} z_j(t) \right]$$

$$p_{ij}^k(t_0) = 0,$$

kde  $k \in U$ ,  $i \in U$  a  $j \in U \cup I$ .

- Výsledná korekční hodnota

$$\Delta w_{ij}(t) = \alpha \sum_{k \in U} e_k(t) p_{ij}^k(t)$$

kde  $\alpha$  je tzv. učicí koeficient



# Implementace - struktury

- Neuronová síť je reprezentována čtyř prvkovým seznamem:
  - `net[[1]]` - matice vah (NetWeights)
  - `net[[2]]` - nastavení (NetOptions)
  - `net[[3]]` - parametry (NetParameters)
  - `net[[4]]` - časově závislá data (NetData)



# Implementace - API

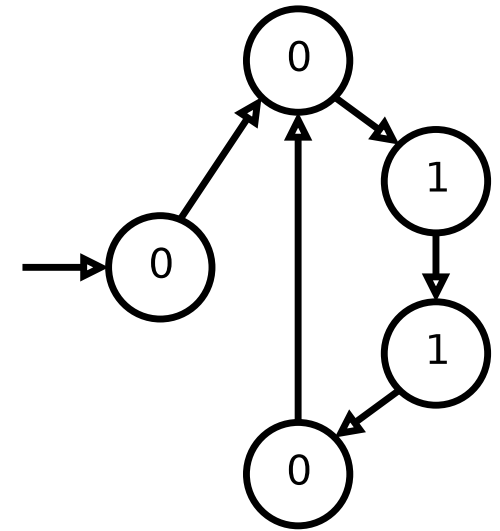
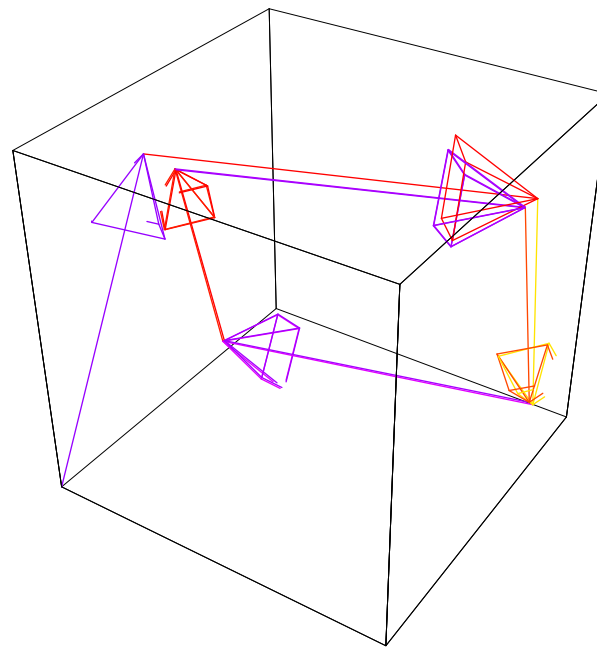
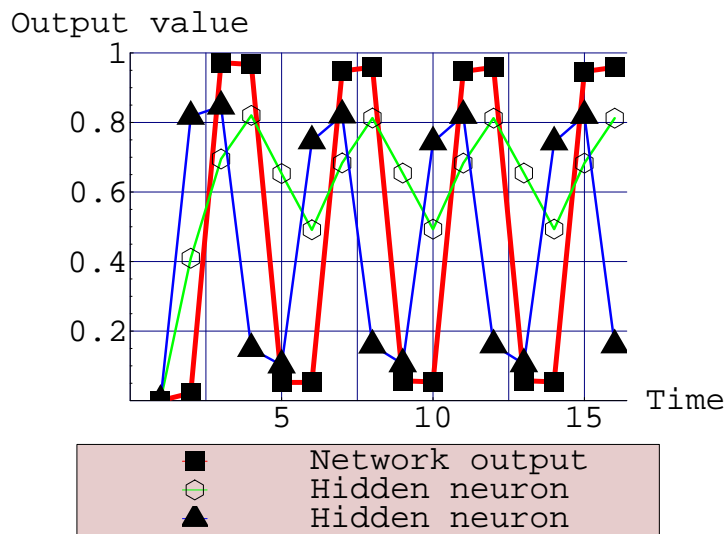
- Podobné API jako knihovna Neural Networks:
- inicializace sítě
  - `InitializeFRNNNet[]`
  - `net = InitializeFRNNNet[x, y, n]`
- učení sítě
  - `FRNNNetFit[]`
  - `net = FRNNNetFit[net, x, y, iter]`
- vybavování sítě
  - `FRNNNet[]`
  - `net[x], net[x, ResetTime->True]`

# ING Experimenty

- Learn to oscillate
  - vytvoření sítě generující určitou posloupnost
- Learn to count
  - neuronová síť přijímající jednoduchých bezkontextový jazyk  $a^n b^n$  (tzv. závorková struktura)
- Predikce časové řady
- Zpracování přirozeného jazyka

# Experiment - Learn to oscillate

- Příklad naučené sítě se třemi neurony
- Učená sekvence 00110011...



Ukázka v prostředí *Mathematica*

- Podařilo se implementovat plně rekurentní neuronovou síť v prostředí systému *Mathematica*,
  - pracovní verze,
  - zatím nezveřejněna.
- Na provedených experimentech byly ověřeny teoretické vlastnosti sítě,
  - schopnost práce s časovým kontextem.

# NGG Budoucnost...

- Implementace dalších variant učicích algoritmů (např. Teacher forcing verze)
- Optimalizace kódu
- Přepřacování do čistě funkcionální podoby
- Vytvoření samostatné knihovny

# Reference

- Neural Networks Application Package

<http://www.wolfram.com/products/>

<http://www.mathematica.cz/produkty.php>

- A Learning Algorithm for Continually Running Fully Recurrent Neural Networks

*Williams, R. J.; Zipser D.*

- Neural Computing Group

<http://ncg.felk.cvut.cz/>