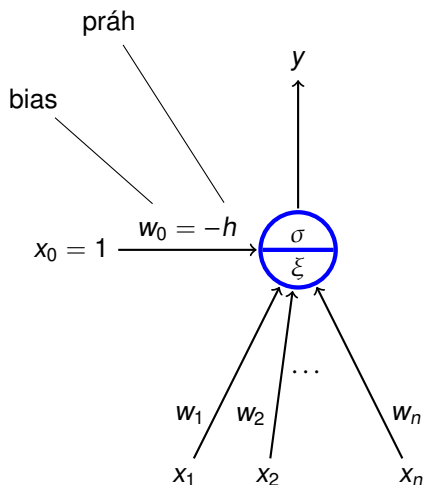


Postavení neuronových sítí v informatice

- ▶ „Vyjadřovací“ schopnosti neuronových sítí
 - ▶ logické funkce
 - ▶ nelineární separace
 - ▶ spojité funkce
 - ▶ algoritmická vyčíslitelnost
- ▶ Srovnání s klasickou architekturou počítačů
- ▶ Implementace

Formální neuron (s biasem)



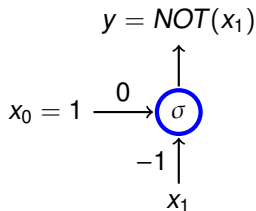
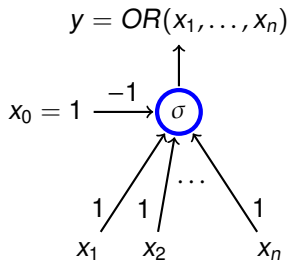
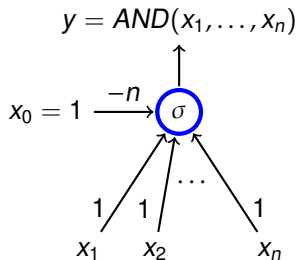
- ▶ x_1, \dots, x_n jsou reálné **vstupy**
- ▶ x_0 je speciální vstup, který má vždy hodnotu 1
- ▶ w_0, w_1, \dots, w_n jsou reálné **váhy**
- ▶ ξ je **vnitřní potenciál**;
většinou $\xi = w_0 + \sum_{i=1}^n w_i x_i$
- ▶ y je **výstup** daný $y = \sigma(\xi)$
kde σ je **aktivační funkce**; např.
ostrá nelinearita

$$\sigma(\xi) = \begin{cases} 1 & \xi \geq 0; \\ 0 & \xi < 0. \end{cases}$$

(práh aktivační funkce σ je roven 0;
reálný práh byl nahrazen vstupem
 $x_0 = 1$ a váhou $w_0 = -h$)

Základní logické funkce

Aktivační funkce σ je ostrá nelinearita $\sigma(\xi) = \begin{cases} 1 & \xi \geq 0; \\ 0 & \xi < 0. \end{cases}$



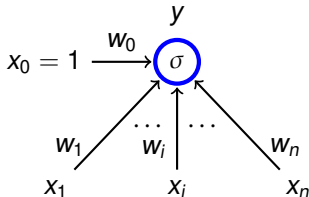
Logické funkce - obecně

Věta

Nechť σ je ostrá nelinearita. Duvrstvé síť s aktivační funkcí σ mohou počítat libovolnou funkci $F : \{0, 1\}^n \rightarrow \{0, 1\}$.

Důkaz.

- Pro každý vektor $\vec{v} = (v_1, \dots, v_n) \in \{0, 1\}^n$ definujeme neuron $N_{\vec{v}}$, jehož výstup je roven 1 právě když vstup je \vec{v} :

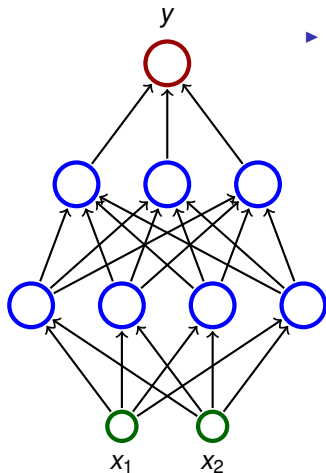


$$w_0 = -\sum_{i=1}^n v_i$$

$$w_i = \begin{cases} 1 & v_i = 1 \\ -1 & v_i = 0 \end{cases}$$

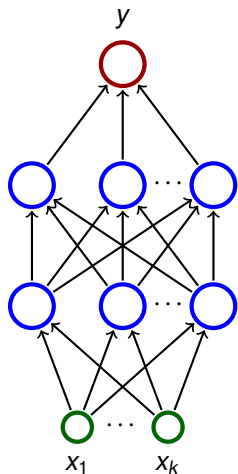
- Spojíme výstupy všech neuronů $N_{\vec{v}}$ pro něž platí $F(\vec{v}) = 1$ pomocí neuronu, který implementuje funkci OR. \square

Nelineární separace - ostrá nelinearita



- ▶ Uvažujme třívrstvou síť; každý neuron má ostrou nelinearitu jako aktivační funkci
- ▶ Síť dělí vstupní prostor na dva podprostory podle hodnoty výstupu (0 nebo 1)
 - ▶ První (skrytá) vrstva dělí vstupní prostor na poloprostory
 - ▶ Druhá může např. dělat průniky poloprostorů - konvexní oblasti
 - ▶ Třetí (výstupní) vrstva může např. sjednocovat konvexní oblasti

Nelineární separace - ostrá nelinearita - ilustrace



- ▶ Uvažujme třívrstvou síť; každý neuron má ostrou nelinearitu jako aktivační funkci
- ▶ Třívrstvá síť může „aproximovat“ libovolnou „rozumnou“ množinu $P \subseteq \mathbb{R}^k$
 - ▶ Pokryjeme P hyperkrychlemi (v 2D jsou to ctverečky, v 3D krychle, ...)
 - ▶ Každou hyperkrychli K lze separovat pomocí dvouvrstvé sítě N_K (Tj. funkce počítaná sítí N_K vrátí 1 pro body z K a 0 pro ostatní)
 - ▶ Pomocí neuronu, který implementuje funkci OR , spojíme výstupy všech sítí N_K t. ž. $K \cap P \neq \emptyset$.

Věta (Cybenko 1989 - neformální verze)

Nechť σ je spojitá funkce, která je sigmoidální, tedy splňuje

$$\sigma(x) = \begin{cases} 1 & \text{pro } x \rightarrow +\infty \\ 0 & \text{pro } x \rightarrow -\infty \end{cases}$$

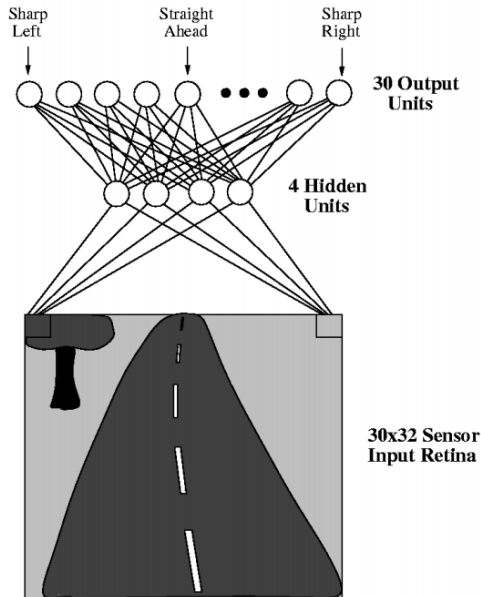
*Pro každou „rozumnou“ množinu $P \subseteq [0, 1]^n$, existuje **dvouvrstvá síť** s aktivační funkcí σ ve vnitřních neuronech (výstupní neuron má lineární), která splňuje následující:*

Pro „většinu“ vektorů $\vec{v} \in [0, 1]^n$ platí $\vec{v} \in P$ právě když výstup této sítě je > 0 pro vstup \vec{v} .

Pro matematicky orientované:

- ▶ „rozumná“ množina je Lebesgueovsky měřitelná
- ▶ „většina“ znamená, že množina špatně klasifikovaných vektorů má Lebesgueovu míru menší než dané ε (pro každé ε může být nutné konstruovat jinou síť)

Nelineární separace - praktická ilustrace



- ▶ ALVINN řídí automobil
- ▶ Síť má $30 \times 32 = 960$ vstupních neuronů (vstupní prostor je \mathbb{R}^{960})
- ▶ Vstupy berou stupně šedi jednotlivých obrazových bodů
- ▶ výstupní neurony „klasifikují“ obrázky silnice podle zakřivení

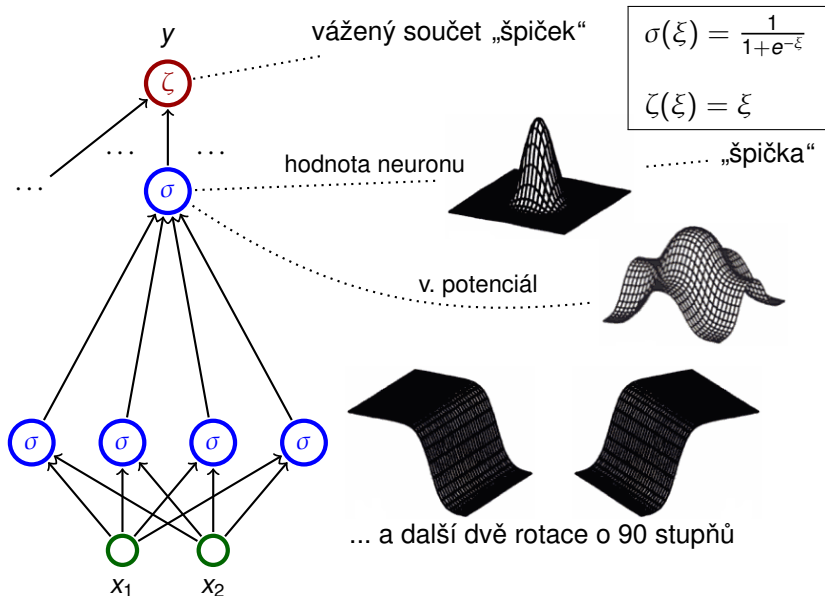
Nechť σ je standardní sigmoida, tj.

$$\sigma(\xi) = \frac{1}{1 + e^{-\xi}}$$

Pro každou spojitou funkci $f : [0, 1]^n \rightarrow [0, 1]$ a $\varepsilon > 0$ lze zkonstruovat třívrstvou síť počítající funkci $F : [0, 1]^n \rightarrow [0, 1]$ takovou, že

- ▶ aktivační funkce v nejvyšší vrstvě je lineární, tj. $\zeta(\xi) = \xi$, ve zbylých vrstvách standardní sigmoida σ
- ▶ pro každé $\vec{v} \in [0, 1]^n$ platí $|F(\vec{v}) - f(\vec{v})| < \varepsilon$.

Aproximace spojitých funkcí - třívrstvé síť



Věta (Cybenko 1989)

Nechť σ je spojitá funkce, která je sigmoidální, tedy splňuje

$$\sigma(x) = \begin{cases} 1 & \text{pro } x \rightarrow +\infty \\ 0 & \text{pro } x \rightarrow -\infty \end{cases}$$

Pro každou spojitou funkci $f : [0, 1]^n \rightarrow [0, 1]$ a $\varepsilon > 0$ existuje funkce $F : [0, 1]^n \rightarrow [0, 1]$ počítaná **dvouvrstvou sítí** jejíž vnitřní neurony mají aktivační funkci σ (výstupní neuron má lineární), která splňuje

$$|f(\vec{v}) - F(\vec{v})| < \varepsilon \quad \text{pro každé } \vec{v} \in [0, 1]^n.$$

Výpočetní síla neuronových sítí (vyčísitelnost)

- ▶ Uvažujme cyklické sítě
 - ▶ s obecně reálnými váhami;
 - ▶ jedním vstupním a jedním výstupním neuronem (sítě tedy počítá funkci $F : A \rightarrow \mathbb{R}$ kde $A \subseteq \mathbb{R}$ obsahuje vstupy nad kterými síť zastaví);
 - ▶ plně paralelním pravidlem aktivní dynamiky (v každém kroku se aktualizují všechny neurony);
 - ▶ aktivační funkcí

$$\sigma(\xi) = \begin{cases} 1 & \xi \geq 0; \\ \xi & 0 \leq \xi \leq 1; \\ 0 & \xi < 0. \end{cases}$$

- ▶ Slova $\omega \in \{0, 1\}^+$ zakódujeme do racionálních čísel pomocí

$$\delta(\omega) = \sum_{i=1}^{|\omega|} \frac{\omega(i)}{2^i} + \frac{1}{2^{|\omega|+1}}$$

Př.: $\omega = 11001$ dá $\delta(\omega) = \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^5} + \frac{1}{2^6}$ (= 0.110011 v dvojkové soustavě).

Výpočetní síla neuronových sítí (vyčísitelnost)

Síť **akceptuje** jazyk $L \subseteq \{0, 1\}^+$ pokud počítá funkci $F : A \rightarrow \mathbb{R}$ ($A \subseteq \mathbb{R}$) takovou, že

$$\omega \in L \text{ právě když } \delta(\omega) \in A \text{ a } F(\delta(\omega)) > 0.$$

- ▶ Cyklické sítě s racionálními váhami jsou ekvivalentní Turingovým strojům
 - ▶ Pro každý rekurzivně spočetný jazyk $L \subseteq \{0, 1\}^+$ existuje cyklická síť s racionálními váhami a s méně než 1000 neuronů, která ho akceptuje.
 - ▶ Problém zastavení cyklické sítě s 25 neuronů a racionálními váhami je nerozhodnutelný.
 - ▶ Existuje „univerzální“ síť (ekvivalent univerzálního Turingova stroje)
- ▶ Cyklické sítě s reálnými váhami jsou silnější než Turingovy stroje
 - ▶ Pro **každý** jazyk $L \subseteq \{0, 1\}^+$ existuje cyklická síť s méně než 1000 neuronů, která ho akceptuje.

Shrnutí teoretických výsledků

- ▶ Neuronové sítě jsou univerzální výpočetní prostředek
 - ▶ dvouvrstvé sítě zvládají Booleovskou logiku
 - ▶ dvouvrstvé sítě aproximují libovolné spojitě funkce
 - ▶ cyklické sítě jsou alespoň tak silné, jako Turingovy stroje
- ▶ Tyto výsledky jsou *čistě teoretické*
 - ▶ sítě vycházející z obecných argumentů jsou extrémně velké
 - ▶ je velmi obtížné je navrhovat
- ▶ Sítě mají jiné výhody a účel (učení, generalizace, odolnost, ...)

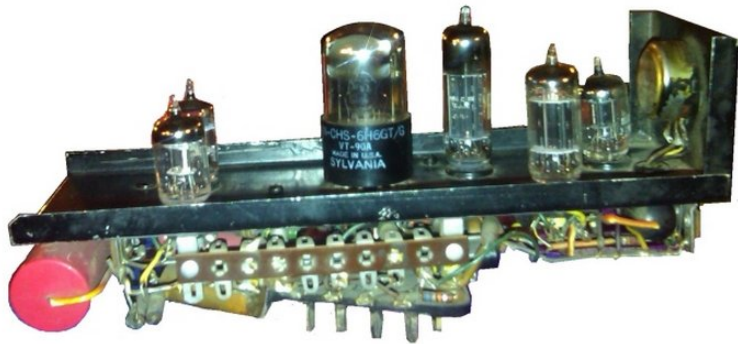
Srovnání s klasickou architekturou počítačů

	Neuronové sítě	„Klasické“ počítače
Data	implicitně ve váhách	explicitně
Výpočet	přirozeně paralelní	sekvenční (obvykle), lokalizovaný
Odolnost	odolné vůči nepřesnosti vstupu a poškození	změna jednoho bitu může znamenat krach výpočtu
Přesnost výpočtu	nepřesný, síť si „vybaví“ podobný tréninkový vzor	přesný
Progra- mování	učí se ze vzorového chování	je nutné precizně programovat

- ▶ Neuropočítač = hardwarová implementace neuronové sítě
- ▶ Obvykle jsou to speciální akcelerátory („karty“), které dostávají vstupy z obyčejného počítače a vrací výstupy sítě
- ▶ Podle typu reprezentace parametrů sítě rozlišujeme neuropočítače na
 - ▶ digitální (většina, např. Neuricom TOTEM, Philips Lneuro a další často pouze výzkumné projekty)
 - ▶ analogové (spíše historické, např. Intel ETANN)
 - ▶ hybridní (např. AT&T ANNA)
- ▶ Lze pozorovat různé stupně hardwarových implementací:
 - ▶ hardware pouze provádí výpočet vnitřních potenciálů (lze provádět paralelně)
 - ▶ hardware počítá vnitřní potenciály i aktivační funkce (je nutné diskrétně aproximovat spojitou akt. funkci)
 - ▶ hardware implementuje učící algoritmus (např. zpětnou propagaci, která se podobá výpočtu sítě, ale od výstupu ke vstupům)

Trocha historie neuropočítačů

- ▶ 1951: SNARC (Minski a spol.)
 - ▶ první implementace neuronu
 - ▶ krysa hledá cestu z ven z bludiště
 - ▶ 40 umělých neuronů (300 elektronek, spousta motorů apod.)



Trocha historie neuropočítačů

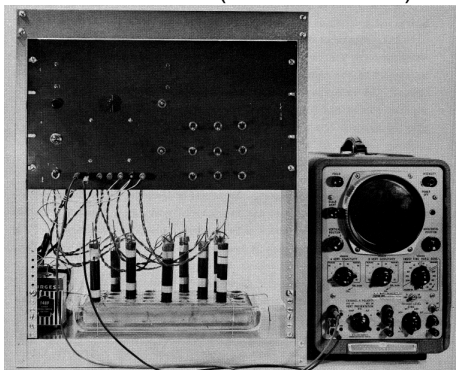
- ▶ 1957: Mark I Perceptron (Rosenblatt a spol.) - první úspěšná neuronová síť pro rozpoznávání obrazců



- ▶ jednalo se v podstatě o jednovrstvou síť
- ▶ obraz snímán 20×20 fotovodiči
- ▶ intenzita bodů byla vstupem sítě perceptronů (jednovrstvá síť), která klasifikovala o jaký se jedná znak
- ▶ váhy byly implementovány pomocí potenciometrů (hodnota odporu nastavována motorem pro každý potenciometr zvlášť)
- ▶ umožňoval prohazovat vstupy do neuronů, čímž se demonstrovala schopnost adaptace

Trocha historie neuropočítačů

- ▶ 1960: ADALINE (Widrow & Hof)



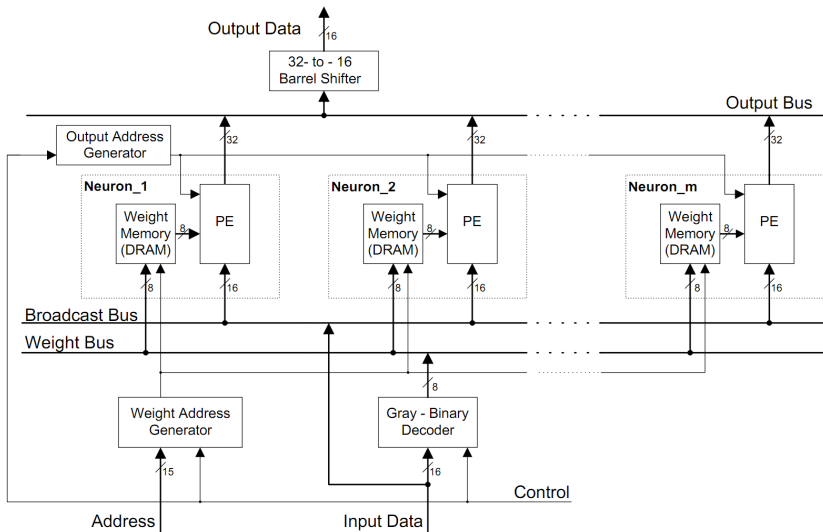
- ▶ V podstatě jednovrstvá síť
- ▶ Váhy uloženy v pomoci nové součástky **memistor**, která si „pamatuje historii“ proudu ve formě odporu.
- ▶ Widrow založil firmu Memistor Corporation, která prodávala hardwarové implementace neuronových sítí.
- ▶ 1960-66: Několik firem zaměřeno na aplikaci neurovýpočtů

Trocha historie neuropočítačů

- ▶ 1967-82: Převážně mrtvo po zveřejnění práce Miského a Paperta (oficiálně vydána roku 1969 pod názvem *Perceptrons*)
- ▶ 1983-konec devadesátých let: Rozmach neuronových sítí
 - ▶ mnoho pokusů o hardwarovou implementaci
 - ▶ jednoúčelové čipy (ASIC)
 - ▶ programovatelné čipy (FPGA)
 - ▶ hardwarové implementace většinou nejsou lepší než „softwarové“ implementace na univerzálních strojích (problémy s pamětí vah, velikostí, rychlostí, nákladností výroby apod.)
- ▶ konec devadesátých let-cca 2005: NS zatlačeny do pozadí jinými modely (support vector machines (SVM))
- ▶ 2006-nyní: Renaissance neuronových sítí
 - ▶ hluboké sítě (mnoho vrstev) - většinou lepší než SVM
 - ▶ znovuobjeven memistor v HP Labs (nyní se jmenuje memristor, má odlišnou konstrukci) - spolu s analogovými neurony umožňuje miniaturizaci (obrovské sítě)
 - ▶ GPU (CUDA) implementace

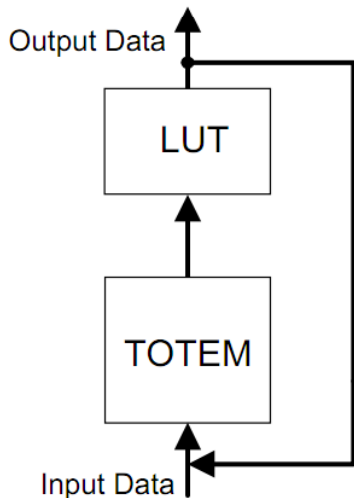
Ilustrativní příklad

Digitální síť TOTEM (ASIC); implementace neuronů:



Ilustrativní příklad - TOTEM

TOTEM (ASIC); minimální síť s obvodem TOTEM



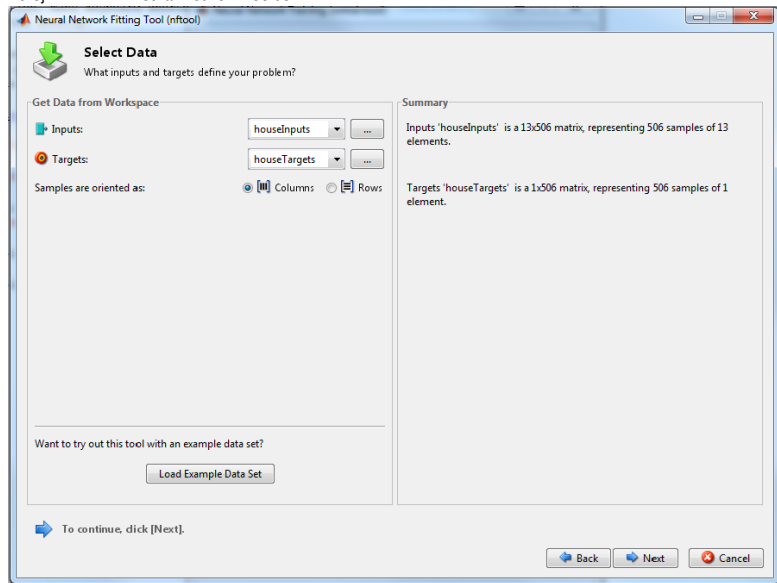
- ▶ LUT = look-up table, která počítá aktivační funkci (není přímo na čipu TOTEM)

Implementace neuronových sítí - software

- ▶ NS jsou součástí několika komerčně dostupných balíčků pro analýzu dat, např.
 - ▶ Alyuda Neurosolutions (software pro tvorbu a aplikaci NS v mnoha oblastech, funguje v Excelu, skvělý marketing :-)) od finančnictví po biotechnologie, zejména predikce a analýza dat
 - ▶ Knowledge Miner (NS + genetické algoritmy pro konstrukci NS, funguje v Excelu)
použití: např. předvídání změn klimatu, makro a mikro ekonomický vývoj, plánování zdrojů, predikce spotřeby zdrojů, ...
- ▶ NS jsou součástí mnoha známých systémů pro vědecké výpočty a analýzu dat
 - ▶ např. MATLAB, Mathematica, Statistica, Weka ...
 - ▶ tyto systémy obvykle obsahují balíky s příkazy pro tvorbu, učení a aplikaci NS
 - ▶ často nabízejí grafické prostředí pro tvorbu sítí

Ilustrace - MatLab (NeuroSolutions)

Zdroj: MATLAB - Neural Network Toolbox



Zdroj: MATLAB - Neural Network Toolbox

Illustrace - MatLab (NeuroSolutions)

```
% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by NFTOOL
%
% This script assumes these variables are defined:
%
%   houseInputs - input data.
%   houseTargets - target data.

inputs = houseInputs;
targets = houseTargets;

% Create a Fitting Network
hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize);

% Set up Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(outputs,targets);
performance = perform(net,targets,outputs)

% View the Network
view(net)
```

Existuje několik knihoven, které lze využít při programování strojového učení, např. FANN a OpenNN (obě open source)

FANN

- ▶ vícevrstvé sítě
- ▶ čtyři učící algoritmy (zpětná propagace a další)
- ▶ multi-platformní (unix, Linux (Debian), windows) a lze ji použít z různých jazyků (C, C++, PHP, Python, Mathematica, ...)
- ▶ existují grafické nástroje postavené na této knihovně (např. FANN Tool)

FANN – ukázka

```
#include "fann.h"

int main()
{
    struct fann *ann = fann_create(1, 0.7, 3, 26, 13, 3);
    fann_train_on_file(ann, "frequencies.data", 200, 10,
        0.0001);
    fann_save(ann, "language_classify.net");
    fann_destroy(ann);
    return 0;
}
```

FANN – ukázka

```
Max epochs      200. Desired error: 0.0001000000
Epochs         1. Current error: 0.7464869022
Epochs        10. Current error: 0.7226278782
Epochs        20. Current error: 0.6682052612
Epochs        30. Current error: 0.6573708057
Epochs        40. Current error: 0.5314316154
Epochs        50. Current error: 0.0589125119
Epochs        57. Current error: 0.0000702030
```

Implementace neuronových sítí - PMML

- ▶ značkovací jazyk založený na XML (vyvinut Data Mining Group)
- ▶ umožňuje popis modelů pro predikci a analýzu dat (mezi nimi Neuronové sítě)
- ▶ podporován mnoha systémy (Statistica, TERADATA Warehouse Miner, Pentaho Weka, atd.)

```
<NeuralLayer numberOfNeurons="2">
  <Neuron id="3" bias="-3.1808306946637">
    <Con from="0" weight="0.119477686963504" />
    <Con from="1" weight="-1.97301278112877" />
    <Con from="2" weight="3.04381251760906" />
  </Neuron>
  <Neuron id="4" bias="0.743161353729323">
    <Con from="0" weight="-0.49411146396721" />
    <Con from="1" weight="2.18588757615864" />
    <Con from="2" weight="-2.01213331163562" />
  </Neuron>
</NeuralLayer>
```