SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta chemickej a potravinárskej technológie

PREDIKCIA ČASOVÝCH RADOV POMOCOU NEURÓNOVÝCH SIETÍ A REGRESNÝCH MODELOV

2009-06-09 **Bc. Peter Hunčár**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

Oddelenie matematiky

Štúdijný program: chémia

Zameranie: chemická informatika

D I P L O M O V Á P R Á C A

Predikcia časových radov pomocou neurónových sietí a regresných modelov

Vedúci diplomovej práce: **Ing. Štefan Babinec, PhD.**

Vypracoval: **Bc. Peter Hunčár**

Bratislava 2009

POĎAKOVANIE

Touto cestou vyslovujem poďakovanie vedúcemu diplomovej práce
Ing. Štefanovi Babincovi PhD. a **Doc. RNDr. Štefanovi Vargovi, CSc.** za pomoc, odborné vedenie, cenné rady a pripomienky pri vypracovaní mojej práce.

Obsah

1 Úvod **7**

2 Teoretická časť **8**

2.1 Štatistické metódy predikcie 8

2.2 Regresné techniky 8

2.2.1 Lineárne regresné modely. 8

2.2.2 Nelineárne regresné modely. 9

2.2.3 Neparametrická regresia a vyhladzovanie. 10

2.2.4 Lokálne vážené regresívne vyhladzovanie (Loess) 11

2.2.5 Supersmoother 11

2.2.6 Kernel smoother 12

2.2.7 Smoothing splines 13

2.3 Časové rady 14

2.3.1 Auto-regresívne modely 15

2.3.2 MA modely 16

2.3.3 ARMA modely 17

2.4 Neurónové siete 18

2.4.1 Moderné aspekty umelej inteligencie 18

2.4.2 História neurónových sietí 18

2.4.3 Základné pojmy 19

2.4.4 Neurónová sieť 20

2.4.5 Neurón 22

2.4.6 Synaptické spojenia a váhy 26

2.4.7 Topológia NN a spôsoby šírenia signálu 27

2.4.8 Učenie neurónových sietí 28

2.4.9 Kontrované učenie dopredných sieti 30

2.4.10 Dopredné NN implementujúce FIR synaptické filtre 37

3 Experimentálna časť **42**

3.1 Implementácia FIR doprednej siete v jazyku Java 42

3.1.1 Návrh 44

3.1.2 Implementácia neurónovej siete 45

3.1.3 Server 47

3.1.4 Core 47

3.1.5 Klient 48

3.2 Štatistické metódy 50
3.3 Neurónové siete 51
3.4 Porovnanie metód 52

4 Záver **57**

5 Použité skratky **58**

1 Úvod

Cieľ predikcie časových radov je možno stručne vyjadriť následovne: Majme sekvenciu $y(1), y(2), ...y(T)$, po určitý čas T, úlohou je nájsť pokračovanie $y(N+1), y(N+2)...$ Údaje časového radu môžu byť vzorky časovo závislého systému a môžu mať náhodný, alebo deterministický charakter. Klasický prístup pozostáva z vytvorenia modelu, ktorý dokáže dobre reprodukovať sledovanú postupnosť. Podľa správania sa systému môžeme voliť rôzne štatistické metódy, ktoré dokážu s určitou presnosťou predpovedať budúce hodnoty. Úlohou tejto práce je porovnať tieto metódy s predikciou využitím umelej neurónovej siete, ktorá bola trénovaná na nameraných hodnotách sledovaného systému. Ďalšou úlohou tejto práce je aj navrhnúť a vytvoriť program ktorý bude tieto siete simulovať. V teoretickej časti sú najprv popísané štatistické metódy predikcie, ďalšia časť je venovaná umelým neurónovým sieťami s hlavným zameraním na siete implementujúce FIR filtre ako synaptické spoje ako aj návrhu implementácie v jazyku Java. V praktickej časti sú porovnané schopnosti oboch prístupov v jedno-krokovej predikcii a časť je venovaná implementácii FIR siete v jazyku Java.

2 Teoretická časť

2.1 Štatistické metódy predikcie

Predikčná analýza zahŕňa rôzne štatistické metódy ktoré analyzujú aktuálne a historické údaje zo zámerom robenia predpovedí budúcich udalostí. V obchode umožňujú predikčné modely postavené na údajoch z minulosti identifikovať riziká a príležitosti. Zahŕňajú vzťahy a závislosti medzi množstvom faktorov umožňujúc tak odhadnúť riziká existujúce za určitých okolností a tak pomáhať pri rozhodovaní.

Predikčná analýza sa takisto používa aj v poisťovníctve, telekomunikáciách, zdravotníctve, farmácii, cestovnom ruchu a v mnohých iných oblastiach.

2.2 Regresné techniky

Regresné modely sú hlavnou oporou pri predikčnej analýze. Cieľom je nájsť takú matematickú závislosť, ktorá by reprezentovala vzťahy medzi rôznymi premennými ktoré sledujeme. Existuje veľké množstvo modelov použiteľných v regresnej analýze v závislosti od situácie.

2.2.1 Lineárne regresné modely.

Lineárne modely analyzujú vzťahy závislej premennej a množiny nezávislých premenných. Závislosť je vyjadrená rovnicou predpovedajúcou budúce hodnoty ako lineárnu funkciu parametrov. Predpokladá sa závislosť typu

$$y = a_1 f_1(x) + a_2 f_2(x) + ... + a_m f_m(x) \tag{1}$$

kde $f_1(x), f_2(x), ..., f_m(x)$ sú známe funkcie jednej alebo viac premenných a $a_1, ..., a_m$ sú neznáme regresné koeficienty.Za predpokladu, že v bodoch $x_1, x_2, ..., x_m$ sú namerané hodnoty $y_1, y_2, ..., y_m$, tak pre ďalšie hodnoty platí

$$y_i = a_1 f_1(x_i) + a_2 f_2(x_i) + ... + a_m f_m(x_i) + e_i \tag{2}$$

kde $e_i$je chyba merania.

Vzťah 1 môžeme napísať v maticovom tvare

$$\mathbf{Y} = F\boldsymbol{\theta} + \mathbf{e}$$

$$\begin{pmatrix} y_1 \\ y_2 \\ ... \\ y_n \end{pmatrix} = \begin{pmatrix} f_1(x_1) & f_2(x_1) & ... & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & ... & f_m(x_2) \\ ... & ... & ... & ... \\ f_1(x_n) & f_2(x_n) & ... & f_m(x_n) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix} \tag{3}$$

a zvykne sa nazývať regresný model.

2.2.2 Nelineárne regresné modely.

Regresná závislosť je nelineárna v parametroch

$$Y = f(X; a_1, a_2, ..., a_m) \tag{4}$$

To znamená, že sa nedá napísať v tvare $y = a_1 f_1(x) + a_2 f_2(x) + ... + a_m f_m(x)$. Pre jednotlivé merania y_iv bodoch $x_i$platí

$$y_i = f(x_i; a_1, a_2, ..., a_n) + e_i \tag{5}$$

Aby sa mohla použiť teória lineárnej regresie, model sa zlinearizuje, za predpokladu, že sú známe približné hodnoty regresných koeficientov $a_1^0, ..., a_m^0$ (tzv. nulté aproximácie) a za predpokladu, že existujú a sú spojité parciálne derivácie $\partial f/\partial a_i (i = 1, ..., m)$. Za takýchto predpokladov približne platí

$$y_i = f(x_i; a_1^0, ..., a_m^0) + \frac{\partial f(x_i)}{\partial a_1}\Delta a_1 + ... + \frac{\partial f(x_i)}{\partial a_m}\Delta a_m + e_i$$

a po úprave

$$y - f(x_i; a_1^0, ..., a_m^0) = \frac{\partial f(x_i)}{\partial a_1}\Delta a_1 + ... + \frac{\partial f(x_i)}{\partial a_m}\Delta a_m + e_i \tag{6}$$

kde Δa_i sú prírastky regresných koeficientov a_i ($a_i = a_i^0 + \Delta a_i; i = l, ..., m$). Model je už lineárny v parametroch. Môžeme ho napísať v tvare

$$\mathbf{Y} = F\boldsymbol{\theta} + \mathbf{e}$$

Vektor $\mathbf{Y}$ je vektor rozdielov pozorovaných hodnôt a vyrovnaných hodnôt, pri použití nultej aproximácie regresných koeficientov. Matica F obsahuje funkčné hodnoty parciálnych derivácii funkcie f podľa jednotlivých regresných koeficientov, v bodoch $x_1, ..., x_n$.

Vektor $\boldsymbol{\theta}$ je vektor prírastkov regresných koeficientov a vektor $\mathbf{e}$ je vektor chýb meraní.

$$\begin{pmatrix} y_1 - f(x_1, a_{1(0)}, ..., a_{m(0)}) \\ y_2 - f(x_2, a_{1(0)}, ..., a_{m(0)}) \\ ... \\ y_n - f(x_n, a_{1(0)}, ..., a_{m(0)}) \end{pmatrix} = \begin{pmatrix} \partial f(x_1)/\partial a_1 & ... & \partial f(x_1)/\partial a_m \\ \partial f(x_2)/\partial a_1 & ... & \partial f(x_2)/\partial a_m \\ ... & ... & ... \\ \partial f(x_n)/\partial a_1 & ... & \partial f(x_n)/\partial a_m \end{pmatrix} \begin{pmatrix} \Delta a_1 \\ \Delta a_2 \\ ... \\ \Delta a_m \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix} \tag{7}$$

Po získaní lineárneho modelu môžme použiť teóriu z regresie lineárnej v parametroch.

2.2.3 Neparametrická regresia a vyhladzovanie.

Modelom neparametrickej regresie je

$$y_i = f(x_i) + e_i \tag{8}$$

kde $y_i (i = 1, 2, ..., n)$ sú nezávislé pozorovania veličiny Y v bodoch x_i(x je prediktor) a e_i sú chyby týchto pozorovaní.Predpokladáme, že funkcia f nie je známa ani jej typ a nevieme koľko má neznámych parametrov. O funkcii sa len predpokladá, že je spojitá, má spojité derivácie a je dostatočne hladká. Úlohou je odhadnúť (vyhladiť, predikovať) funkčné hodnoty $f(x_i)$, kde $i = 1, 2, \ldots, n$, resp. $f(x)$, $x \notin \{x_1, ..., x_n\}$. Riešenie sa hľadá tak, že sa minimalizuje vhodná funkcia φ (v závislosti či ide o klasické alebo robustné vyhladzovanie) odchýlok nameraných a vyhladených hodnôt a maximalizuje hladkosť $f(x)$

$$\sum_{i=1}^{n} w_i \varphi(y_i - f(x_i)) + \alpha \int_b^b (\hat{f}''(x))^2 \tag{9}$$

Samozrejme odhady $\hat{f}(x)$ sú menej rozptýlené ako pôvodné dáta a preto sa často nazývajú aj ako vyhladené hodnoty premennej Y a rôzne metódy týchto odhadov sa nazývajú vyhlazovače. Veľmi často odhady funkčných hodnôt $f(x_i)$ sú lineárne kombinácie jednotlivých pozorovaní y_i

$$\hat{f}(x) = \sum_{i=1}^{n} F_\alpha(x, x_i) \tag{10}$$

$F_\alpha(x, x_i)$ predstavuje váhy jednotlivých pozorovaní. V prípade ak sú tieto váhy rôzne od nuly iba pre niektoré najbližšie pozorovania v okolí bodu x, tak ide o tzv. lokálne vyhladzovanie. Neparametrická regresia sa používa hlavne v prípadoch, pri ktorých je funkcia neznáma.

2.2.4 Lokálne vážené regresívne vyhladzovanie (Loess)

Pri lokálne váženom regresnom vyhladzovaní sa zostaví vyhladená krivka $s(x)$, následovne:

1. Vezme sa bod, x_0. Nájde sa k blízkych susedov bodu x_0, ktoré tvoria okolie $N(x_0)$. Počet susedov k je špecifikovaný ako percento z celkového počtu bodov. Toto percento sa nazýva spaní.

2. Vypočíta sa najväčší rozdiel medzi x_0 a iným bodom z jeho okolia

$$\Delta(x_0) = max_{N(x_0)}(x_0 - x_1) \tag{11}$$

3. Priradí sa váha každému bodu v $N(x_0)$ použitím funkcie

$$W(\frac{|x_0 - x_1|}{\Delta(x_0)}) \tag{12}$$

kde

$$W(u) = \begin{cases} (1-u^3)^3, & pre\ 0 \leq u \leq 1 \\ 0 & inde \end{cases} \tag{13}$$

4. Vykoná sa vážená lineárna regresia okolia $N(x_0)$ a na jej základe sa vypočíta $\hat{y}_0 = s(x_0)$.

5. Postup sa opakuje pre každú hodnotu prediktora. Pri danej metóde je span vždy konštantný pre všetky predikované hodnoty x.

2.2.5 Supersmoother

V prípade lokálneho váženého regresného vyhladzovania, je span konštantný pre všetky predikované hodnoty, avšak v prípade metódy supersmoother tomu tak nie je. V niektorých prípadoch je totiž vhodnejšie meniť hodnotu spanu a to dôsledkom zmeny chýb alebo zakrivenia základnej funkcie f cez rozsah predikovaných hodnôt. Zväčšenie chybovej variácie vyžaduje zväčšenie spanu, a naopak zväčšenie krivosti vyžaduje jeho zmenšenie. Span je v tomto prípade funkciou prediktora. Na výber spanu sa používa metóda, nazývaná „cross-validation“. Vyberá sa $k = k(X)$ tak, že sa minimalizuje stredná kvadratická chyba

$$e^2(k) = E_X Y\,[Y - s(X|k)]^2 \tag{14}$$

pričom $s(X|k)$ je lineárne vyhladená hodnota v x keď je použitý span k. Pretože

$$E_X Y \left[Y - s(X|k)\right]^2 = E_X E_{Y|X} | \left[Y - s(X|k)\right]^2 \tag{15}$$

môže sa vybrať $k = k(x)$ minimalizáciou

$$e_X^2(k) = EY|X = [Y - s(X|k)] \tag{16}$$

K dispozícii sú však len údaje x_i, y_i $i = 1, 2, \ldots, n$ a preto sa nedá vypočítať $e_X^2(k)$ a musí sa použiť metóda "cross-validation "

$$\hat{e}_{CV}^2(k) = \sum_{i=1}^{n} \left[y_1 - s_{(i)}(x_i|k)\right]^2 \tag{17}$$

$s(i)(X|k)$ je lineárne vyhladená hodnota v bode x_i tak, že sa použijú všetky body x_j, y_j $j = 1, 2, \ldots, n$ okrem x_i, y_i.

2.2.6 Kernel smoother

Kernel-smoother je typ lokálneho vyhladzovača, ktorý pre každý cieľový bod x_i v prediktorovom priestore vypočíta vážený priemer pozorovaní y_i v susedstve cieľového bodu x_i.

$$\hat{y}_i = \sum_{j=1}^{n} w_{ij} y_j \tag{18}$$

kde w_{ij} predstavujú váhy

$$w_{ij} = \widetilde{K}\left(\frac{x_i - x_j}{b}\right) = \frac{K\left(\frac{x_i - x_j}{b}\right)}{\sum_{k=1}^{n}\left(\frac{x_i - x_j}{b}\right)} \tag{19}$$

pričom ich suma je 1

$$\sum_{j=1}^{n} w_{ij} = 1 \tag{20}$$

Parameter b predstavuje šírku pásma (bandwidth), ktorá určuje aké veľké okolie cieľového bodu sa použije na výpočet lokálneho priemeru. Väčšia šírka pásma dáva hladšiu krivku, kým menšia šírka pásma dáva menej hladkú krivku. K je Kernel (jadrová) funkcia.

Jej vlastnosťami sú:

- $K(t) \geq 0$ pre všetky hodnoty t

- $\int_{-\infty}^{\infty} K(t)dt = 1$
- $K(-t) = K(t)$pre všetky hodnoty t (je symetrická)

Najčastejšie používané typy jadrovej funkcie sú:

- "box" (krabica)

$$K_{box} = \begin{cases} 1, & |t| \leq 0,5 \\ 0, & |t| > 0,5 \end{cases} \tag{21}$$

- "triangle" (trojuholník)

$$K_{tri} = \begin{cases} 1 - |t| / C, & |t| \leq \frac{1}{C} \\ 0, & |t| > \frac{1}{C} \end{cases} \tag{22}$$

- "parzen" K

$$K_{par} = \begin{cases} (k_1 - t^2)/k_2, & |t| \leq C_1 \\ (t^2/k_3) - k_4 |t| + k_5, & C_1 < |t| > C_2 \\ 0, & C_2 < |t| \end{cases} \tag{23}$$

- „normal" (normálne Gaussovo rozdelenie)

$$K_{nor} = (1/\sqrt{2\pi}k_6)e^{-t^2/2k_6^2} \tag{24}$$

Výber šírky pásma je dôležitejší ako výber jadrovej funkcie.

2.2.7 Smoothing splines

Spline vyhladzovač funguje tak, že minimalizuje penalizovanú sumu štvorcov odchýlok danú vzťahom

$$PRSS = \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int (f''(t))^2 dt \tag{25}$$

Parameter λ je parameter hladkosti, má podobný význam ako span pri loess, alebo šírka pásma pri jadrovom vyhladzovači.

2.3 Časové rady

Modely predikujúce časové rady predpovedajú budúce hodnoty premenných. Tieto metódy sa zakladajú na tom, že údaje nazbierané za určitý čas majú nejakú vnútornú štruktúru (autokoreláciu, trend, sezónne zmeny). Z tohto dôvodu štandardné regresné techniky nemôžu byť použité a vyvinuli sa metódy na rozloženie tejto vnútornej štruktúry na cyklické, sezónne a iné zložky.

Bežne sa používajú dva známe modely. Auto-regresívny (AR) model a model plávajúcich priemerov (moving average, MA). Geoge Box a G.M. Jenkins skombinovali v roku 1976 AR a MA modely do ARMA (autoregresive moving average) modelu, ktorý sa stal základným kameňom analýzy stacionárnych časových radov. ARIMA (autoregresive integrated moving average) modely sa používajú na popisovanie nestacionárnych časových radov. Nestacionárne časové rady majú výrazný trend a ich variancia alebo stredná hodnota nie je dlhodobo konštantná.

Navrhli sa tri kroky tejto metódy:

- Identifikácia modelu. Tu sa zisťuje či je časový rad stacionárny, alebo nestacionárny, či má prejavy sezónnych zmien, prípadne auto-korelácia.
- Odhad. Na odhadovanie parametrov modelu sa používajú rôzne metodiky.
- Overenie. Porovnáva sa predikcia modelu s nameranými hodnotami.

V posledných rokoch sa tieto metodiky stali podstatne sofistikovanejšími. Vznikli nové modely ako ARCH (autoregresive conditional heteroskedasticity) a GARCH (generalized autoregresive conditional heteroskedasticity). Používajú sa najmä pri modelovaní finančných časových sérií. Nakoniec, modely predikcie časových radov sa takisto používajú na pochopenie vnútorných vzťahov ekonomických ukazovateľov s použitím VAR (vector autoregression).

2.3.1 Auto-regresívne modely

Auto-regresívny model v štatistike či pri spracovaní signálu je typ náhodného deja, ktorý sa často používa na predpovedanie rôznych typov prírodných dejov.

Zápis $AR(p)$ značí auto-regresívny model rádu p. Je definovaný ako:

$$X_t = c + \sum_{i=1}^{p} \varphi_{t-i} + \varepsilon_t \tag{26}$$

kde $\varphi_1, ..., \varphi_p$ sú parametrami modelu, c je konštanta a ε_t sa nazýva "biely šum".

$AR(1)$ model je daný:

$$X_t = c + \varphi X_{t-1} + \varepsilon_t \tag{27}$$

kde ε_t je šum s nulovou strednou hodnotou a varianciou. Dej je stacionárny ak

$$|\varphi| < 1 \tag{28}$$

Za splnená tohto predpokladu dostávame

$$E(X_t) = E(c) + \varphi E(X_{t-1}) + E(\varepsilon_t) \Longrightarrow \mu = c + \varphi\mu \tag{29}$$

kde μ je stredná hodnota a teda

$$\mu = \frac{c}{1-\varphi} \tag{30}$$

Ak $c = 0$ tak potom $\mu = 0$.

Pre varianciu platí:

$$var(X_t) = E(X_t^2) - \mu^2 = \frac{\sigma^2}{1-\varphi^2} \tag{31}$$

Určenie parametrov $AR(p)$ modelu. Model je daný rovnicou

$$X_t = \sum_{i=1}^{p} \varphi_{t-i} + \varepsilon_t \tag{32}$$

Existuje priama spojitosť medzi parametrami φ_i a kovarianciou deja. To sa využíva na určenie parametrov z funkcie auto-korelácie pomocou Yule-Walker rovníc:

$$\gamma_m = \sum_{k=1}^{p} \varphi_k \gamma_{m-k} + \sigma_\varepsilon^2 \delta_m \tag{33}$$

kde $m = 0, ..., p$. γ_m je auto-korelácia, σ_ε je štandardná odchýlka vstupného šumu a $\delta_m$je

Kronecker delta funkcia. V maticovom tvare:

$$\begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \dots \end{pmatrix} = \begin{pmatrix} \gamma_0 & \gamma_{-1} & \gamma_{-2} & \dots \\ \gamma_1 & \gamma_0 & \gamma_{-1} & \dots \\ \gamma_2 & \gamma_1 & \gamma_0 & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \dots \end{pmatrix} \tag{34}$$

Pre $m = 0$ dostaneme:

$$\gamma_0 = \sum_{k=1}^{p} \varphi_k \gamma + \sigma_\varepsilon^2 \tag{35}$$

Čo nám umožní určiť σ_ε^2. Yule-Walker rovnice umožňujú odhadnúť parametre $AR(p)$ vymenením teoretických kovariancií odhadnutými hodnotami.

2.3.2 MA modely

V predikčnej analýze časových radov je MA model častým spôsobom modelovania jednorozmerných radov. Zápis $MA(q)$ označuje MA model rádu q.

$$X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_1 + ... + \theta_q \varepsilon_{t-q} \tag{36}$$

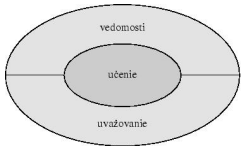
kde μje stredná hodnota, $\theta_1, ..., \theta_q$ sú parametrami modelu a $\varepsilon_t, \varepsilon_{t-t}, ...$ sú hodnoty šumu. q sa nazýva rádom MA modelu. Koncepčne je MA model je vlastne lineárna regresia aktuálnych hodnôt rádu so šumom a náhodnými skokmi. Náhodné skoky majú typicky normálnu distribúciu. Odlišuje sa tým, že tieto skoky sa propagujú to budúcich hodnôt časového radu. Odhadnutie MA modelu je omnoho ťažšie ako odhadovanie AR modelu, pretože chyby nie sú pozorovateľné. To znamená, že namiesto lineárnych metód sa musia použiť nelineárne iteračné postupy. MA model je vlastne typ FIR (Finitie Impulse Response) filtra.

2.3.3 ARMA modely

Pri popisovaní ARMA modelu môžeme sledovať dva smery úvah. Správanie sa systému závisí na predchádzajúcich hodnotách pozorovaní. Napríklad ak vidíme, že po určitú dobu má krajina vysoký HDP, môžeme očakávať, že aj v najbližšej budúcnosti bude HDP hodnota vysoká. Táto predstava je reprezentovaná AR modelom (26). Ak sa vyberieme druhým smerom, vidíme že sledovaná veličina nie je ovplyvňovaná len náhodnými, neočakávanými zmenami v prítomnosti, ale aj podobnými zmenami v minulosti. Tento prístup je reprezentovaný MA modelom (36). Kombináciou týchto dvoch prístupov dostaneme ARMA model. Nutná podmienka ARMA modelov je, že rovnice musia mať stacionárne riešenie.

Zápis $ARMA(p, q)$ označuje model kombináciu $AR(p)$ a $MA(q)$ modelov a jej rovnica je:

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{37}$$



Obr. 1: Hlavné komponenty všeobecného systému UI

2.4 Neurónové siete

2.4.1 Moderné aspekty umelej inteligencie

Najnovší trend, ktorý v súčasnosti upriamuje pozornosť odbornej verejnosti vychádza z paralelných výpočtov a nového pohľadu na paralelné systémy. Ide o quasiencefalické výpočty, v ktorých sa jedná o masívny paralelizmus a ich cieľom je modelovať chovanie sa nervovej sústavy u živočíchov a hlavne modelovať chovanie sa ľudského mozgu. Ľudský mozog obsahuje 10^{11} – 10^{14} neurónov uložených v šedej mozgovej kôre a synapsie sú realizované v rozsahu 10^4 na jeden neurón. Vytvoriť umelý ľudský mozog je veľmi ťažko riešiteľná vec a to nielen z hľadiska kvantity neurónov, ale aj z hľadiska ich prepojenia atď. Je však možné simulovať aspoň niektoré funkcie ľudského myslenia a implementovať ich. Modely, ktoré sa v súčasnosti o túto situáciu pokúšajú, dostali názov neurónové siete (NN – Neural Nets). V tejto súvislosti vzniká tiež otázka vzťahu neurónových sietí k systémom umelej inteligencie (UI).

Cieľom systémov umelej inteligencie je vypracovať paradigmy alebo algoritmy, ktoré požadujú od stroja riešiť úlohy, ktoré by vyriešil len človek so znalosťami.Toto je jedna z mnohých definícií systémov UI.

V tejto časti je rozpracovaná všeobecná teória umelých neurónových sieti. Časti zaoberajúcej sa učeniu neurónových sieti je venovaná veľká časť, nakoľko je potrebná k pochopeniu učenia FIR NN topológie použitej v tejto práci.

2.4.2 História neurónových sietí

V tejto kapitole je uvedený veľmi stručný náčrt dejín vývoja NN ako odboru UI. Rozvoj tohto odboru bude pravdepodobne aj naďalej dynamický a pokiaľ sa nenájde iná alternatívna cesta v UI pre budúcnosť, je viac ako isté, že NN sa stanú dominantným prvkom moderného chápania UI.

- 1943 pod vedením vedcov McCullocha a Pittsa začína éra teórie NN. Dr. McCulloch bol psychiater a neuroanatóm, kým dr. Pitts bol matematik. Prvýkrát definovali na Univerzite v Chicagu binárny neurón. Ich prácou sa do určitej miery nechal

inšpirovať aj John von Neumann pri konštrukcii svojho prvého počítača ENIAC v roku 1946.

- 1952 publikácia dr. Ashbyho pod názvom Design of a Brain: The Origin of Adaptive Behavior mala zásadný význam pre rozvoj NN.
- 1954 dr. Minsky napísal svoju Ph.D. dizertáciu na tému Neurónové siete a neskôr, v roku 1961, publikoval zásadný článok Step Towards Artificial Intelligence.
- obdobie medzi rokmi 1970 – 1980 sa považuje za obdobie útlmu . Aj keď boli urobené niektoré práce v teoretickej oblasti, neboli písané s takou dynamikou ako v minulosti. Dôvodom boli najmä nedostatočné výpočtové kapacity vrátane pamäťových možností.
- 1986 dr. Rumelhart a kol. prišli s metódou učenia spätným šírením chyby. Vďaka svojej relatívnej jednoduchosti je táto metóda jednou z najrozšírenejších metód učenia NN.

2.4.3 Základné pojmy

V tejto časti sú uvedené základné pojmy, ktoré sa týkajú tejto problematiky. Vo všeobecnosti činnosť neurónových sieti rozdeľujeme do dvoch fáz:

- Fáza učenia. V tejto fáze sa znalosti ukladajú do synaptických váh[1] neurónovej siete. Ak označíme maticu W ako maticu všetkých synaptických váh neurónovej siete, tak pod učením budeme chápať stav, kedy platí, že:

$$\frac{\partial W}{\partial t} \neq 0 \tag{38}$$

  Teda synaptické váhy sa počas učenia menia. Pri učení ide o zbieranie poznatkov, prípadne. ich uchovanie. Synonymom pojmu učenia pri NN je pojem adaptácia NN. Definícia učenia sa dá potom interpretovať nasledovne:

Učenie je proces, v ktorom sa parametre NN (synaptické váhy ďalej SV) menia na základe nejakých pravidiel. Charakter týchto pravidiel, ktoré vyvolávajú zmeny SV NN, determinuje typ učenia NN. Pod učením rozumieme adaptáciu NN, ktorá po ukončení učenia bude nositeľkou znalostí získaných počas učenia.

[1] Pojem synaptická váha bude vysvetlený v časti 2.4.6.

- Fáza života. Tu sa získané znalosti využívajú v prospech riešenia nejakého problému (napr. Klasifikácia, optimalizácia, zhlukovanie a pod.) V praxi to znamená, že ide o stav, kedy:

$$\frac{\partial W}{t} = 0 \tag{39}$$

  teda synaptické váhy sa nemenia.

- Niekedy sa stretávame s označením NN ako bezalgoritmických systémov[2]. Toto tvrdenie je síce pravdivé, ale vzťahuje sa len na fázu života NN, kým vo fáze učenia prebieha v neurónových sieťach cieľavedomý proces uchovávania poznatkov do synaptických váh.

2.4.4 Neurónová sieť

Neurónová sieť (ďalej len NN) je masívne paralelný procesor, ktorý má sklon k uchovávaniu experimentálnych znalostí a ich ďalšieho využívania. Napodobňuje ľudský mozog v dvoch aspektoch:

- poznatky sú zbierané v NN počas učenia
- medzineurónové spojenia (synaptické váhy – SV) sú využívané na ukladanie znalostí

Táto definícia je jednou z mnohých definícii NN, ktorá je akceptovaná komunitou. Je zrejmé, že inšpirácia k vzniku NN prišla z biologických systémov. Jednoducho povedané ide o simuláciu mozgu. Aj keď je jasné, že simulácia takého komplexného systému nie je dnes možná, aj neporovnateľne menšie siete majú veľa zaujímavých vlastností. Jedna z týchto vlastností je, že neurónová sieť je svojim spôsobom takzvaný univerzálny aproximátor funkcií. Ak máme systém, ktorého matematický popis nemáme, alebo je priveľmi zložitý, môžeme sa pokúsiť vytvoriť neurónovú sieť a naučiť ju chovať sa ako sledovaný systém. Ak máme experimentálne namerané hodnoty výstupov systému pre určitú množinu vstupov, správne naučená sieť dokáže veľmi presne aproximovať hodnoty výstupov tohto systému pre tie hodnoty vstupov pre ktoré experimentálne dáta nemáme. Bolo dokázané, že neurónová sieť je schopná aproximovať akúkoľvek funkciu s ľubovoľnou presnosťou. Takéto jednoduché aproximátory sa dajú rozšíriť zavedením ďalšej veličiny, času. Zavedenie internej dynamiky do NN umožňuje sieti "pamätať" si chovanie sa systému v čase. Tieto siete budú bližšie opísané ďalej. Ďalšie oblasti použitia NN sú:

- problémy aproximácie funkcií

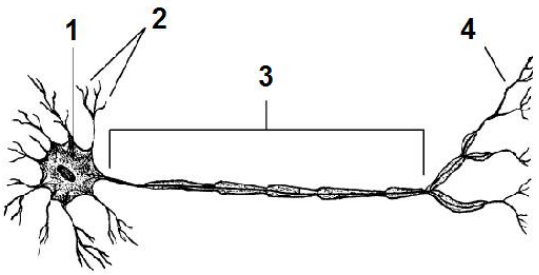
[2] algorithm-less systems

- klasifikácie do tried, klasifikácia situácií
- riešenie predikčných problémov
- problémy riadenia procesov
- transformácia signálov
- asociačné problémy, simulácia pamäte

Pri štúdiu NN sa rozlišujú tri oblasti:

- teória NN – matematický rozbor činnosti NN, problémy NN ako dynamického systému vo všeobecnosti, teoretické rozbory návrhu topológie NN a pod. Tu je nutné upozorniť, že matematický model pre popis chovania sa NN je dosť náročný.
- simulácia NN – jedná sa o simuláciu NN pomocou počítačových systémov. Najväčším problémom simulácie je naučiť NN na niečo. Proces učenia je veľmi časovo náročný a vyžaduje veľké (najlepšie paralelné) výpočtové systémy. Vo svete existuje množstvo simulátorov NN. V súčasnosti sa ako najvhodnejší javí Stuttgartský simulátor NN.
- Implementácia NN – ide o implementáciu naučenej NN do hardwarovej formy. Takéto systémy existujú a objavujú sa čoraz častejšie mikro-čipy NN.

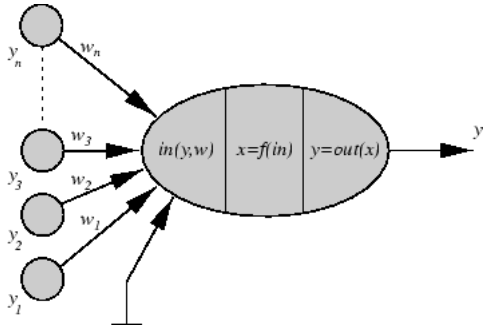
Všetky tri uvedené oblasti veľmi úzko súvisia, avšak v súčasnej situácii plný rozvoj prežívajú najmä možnosti využitia výpočtovej techniky. Keď hovoríme o vlastnostiach NN je potrebné zdôrazniť nevyhnutnosť paralelizmu týchto systémov. Ako bolo uvedené vyššie, nasimulovať neurónovú sieť s podobnými parametrami ako má hoci aj živočích s primitívnou nervovou sústavou, je stále mimo schopností dnešných počítačov. Tie dnes dokážu spracovať obrovské množstvo inštrukcií za sekundu, ale vykonávajú ich sekvenčne, pričom viac úloh náraz zvládajú klamlivo tak, že im operačný systém dávkuje inštrukcie bežiacich procesov striedavo.



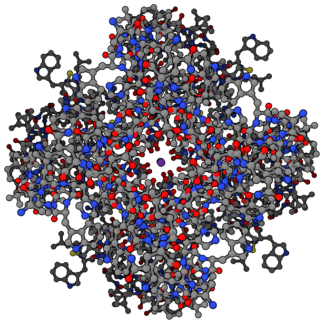
Obr. 2: Neurón

2.4.5 Neurón

Základným elementom NN je neurón, obr. 2. Nervová bunka, neurón sa skladá z tela (1), a niekoľkých výbežkov. Tieto výbežky sa rozdeľujú na dva typy, a to na dendrity (2) a jeden dlhý axón (3). Dendrity signály prijímajú, prostredníctvom nich sa vzruch šíri do neurónu. Následne nastáva spracovanie informácie neurónom. Po spracovaní danej informácie neurón vysiela signál pomocou axónu k iným bunkám. Miesto prenosu signálu z jedného neurónu na druhý sa nazýva synapsia. Axón sa na svojom konci rozvetvuje na veľké množstvo výbežkov (4). Zakončenia (terminály) týchto výbežkov tvoria synapsie na ostatných neurónoch v sieti. V synapsii sa môže signál buď zosilniť alebo zoslabiť. V prípade nervovej bunky, prenos signálu vrámci bunky prebieha zmenou potenciálu pozdĺž bunkovej membrány. Ten bunka dosahuje zmenou koncentrácie iónov K^+,Na^+,Ca^{2+},Cl^-medzi bunkou a vonkajším prostredím. Najdôležitejšiu úlohu pri šírení signálu majú však ióny Na^+ a K^+. V bunkovej membráne sa nachádzajú špecializované bielkoviny, ktoré sú buď zodpovedné za transport iónov, napríklad sodíkovo draslíková pumpa, alebo slúžia ako brána voľného transportu iónov cez membránu viď obr. 3. Sodíkovo draslíkova pumpa premiestňuje tri sodíkové ióny mimo bunku a dva draslíkové dovnútra bunky. Takto dosiahnutý pokojový potenciál ma hodnotu približne -70mV. Pri prenose signálu dochádza k tzv. depolarizácii, keď otvorené iónové kanály, umožnia voľný transport iónov a výsledný prúd je zodpovedný za zmenu rozdielu potenciálov približne na hodnotu +50mV v maxime. Táto zmena sa ako vlna šíri povrchom bunky. Na prenos signálu medzi bunkami slúžia synapsie, ktoré v mieste kontaktu buniek uvoľňujú molekuly neurotransmiterov, ktoré spôsobia depolarizáciu membrány susednej bunky na danom mieste. Tento opis je značne zjednodušený, proces je pomerne zložitý a ovplyvňuje ho množstvo iných faktorov, ale napriek tomu je veľmi rýchly. V prípade umelých sietí je situácia omnoho jednoduchšia viď obr. 4.



Obr. 4: Štruktúra modelovaného neurónu



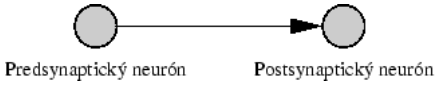
Obr. 3: Otvorený draslíkový kanál

- vstup do neurónu (dendrit)
- prah neurónu – je hodnota θ_i, ktorá vlastne prispieva ku vstupu z externého sveta
- aktivačná funkcia neurónu f, ktorej výsledkom je x_i
- výstupná funkcia neurónu o_i
- synaptické váhy, ktoré sú na synaptických spojeniach (synapsiách), ktoré majú svoj smer a spájajú jednotlivé neuróny do NN

Podľa toku signálu po synapsii rozoznávame neuróny

- predsynaptické (zdrojové – pred synapsiou)
- postsynaptické (cieľové – po synapsii)

Na označovanie synaptických váh sa používa symbol w_{ij}, kde „i“ označuje postsynaptický neurón a „j“ označuje predsynaptický neurón. Teda ide o synapsiu, ktorá vychádza z neurónu „j“ a cieli k neurónu „i“. Túto konvenciu pri označovaní je vhodné dodržať.



Obr. 5: Označenie neurónov

- vstup so neurónu - je funkciou jednotlivých vstupov prichádzajúcich od predsynaptických neurónov. Vo väčšine prípadov je to súčet týchto vstupov uvažovaných s určitými váhami, napríklad vstup do i-teho neurónu môže byť vyjadrený v tvare

$$in_i \;=\; \sum_{j=1}^{N} w_{i,j}\; ou_j + \theta_i \tag{40}$$

  kde w_{ij}sú synaptické váhy a ou_jsú výstupy z neurónov, s ktorými je prepojený, $\theta_i$je prah neurónu i.

Rovnica 40 môže byť prepísaná v tvare

$$in_i \;=\; \sum_{j=0}^{N} w_{i,j}\; ou_j \tag{41}$$

kde $w_{ij} = \theta_i$a $ou_0 = 1$ *alebo* -1. Prah je vlastne vstup do neurónu z vonkajšieho sveta, teda nie z iných neurónov. To znamená, že ak daný neurón nemá vstupy z iných neurónov, potom vstupom do tohto neurónu je iba prah θ_i. Takéto neuróny nazývame *sigma* neuróny.

- aktivačná funkcia neurónu
- Aktivačná funkcia neurónu je funkciou vstupu do neurónu $in_i(t)$. Teda stav neurónu i je definovaný premennou x_iv tvare

$$x_i \;=\; f(in_i) \tag{42}$$

  Funkciu $f()$ budeme nazývať aktivačnou funkciou neurónu. Najdôležitejšie typy týchto funkcií sú následovné:

1. Lineárna funkcia

$$x_i \;=\; f(in_i) \;=\; in_i \tag{43}$$

2. Funkcia signum

$$x_i \;=\; f(in_i) \;=\; \begin{cases} 1 & in_i \geq 0 \\ 0 & in_i < 0 \end{cases} \tag{44}$$

3. Funkcia lineárna po častiach

$$x_i \;=\; f(in_i) \;=\; \begin{cases} 1 & in_i \geq \frac{1}{2} \\ in_i & in_i \in \left(-\frac{1}{2}, \frac{1}{2}\right) \\ 0 & in_i \leq 0 \end{cases} \tag{45}$$

4. Sigmoidálna funkcia

$$x_i \;=\; f(in_i) \;=\; \frac{1}{1 + e^{-\alpha in_i}} \tag{46}$$

kde α je parameter strmosti sigmoidy. Podobnou funkciou je tieže hyperbolický tangens, existuje však množstvo ďalších aktivačných funkcií.

- výstupná funkcia neurónu je taktiež dôležitou súčasťou neurónu ako procesnej jednotky. Často býva identickou funkciou $ou_i = x_i$.

Obr. 6: Synapsia v biologickom systéme

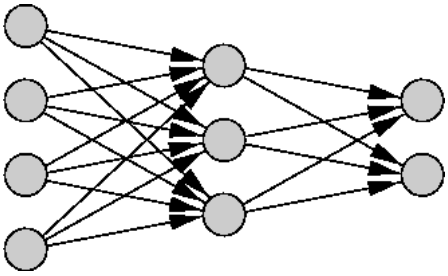
2.4.6 Synaptické spojenia a váhy

Ako už bolo spomenuté vyššie. Biologický proces prenosu signálu z neurónu na neurón je značne komplikovaný a zúčastňuje sa na ňom veľké množstvo rôznych chemických zlúčenín, z ktorých môžu niektoré tento proces urýchľovať, iné naopak, úplne zastaviť a celý tento dej je dynamický, v čase kedykoľvek ovplyvniteľný. Simulácia umelej synapsie v porovnaní s biologickým systémom je triviálna.

Na prepojeniach medzi umelými neurónmi, ktoré sú orienované, uvažujeme tzv. synaptické váhy. Váhy ovplyvňujú celú sieť tým, že ovplyvňujú vstupy do neurónov a tým aj ich stavy. Rozdeľujeme ich na:

- kladné, teda excitačné
- záporné, teda inhibičné

Synaptické váhy medzi neurónami i, j označujeme $w_{i,j}$. Najdôležitejším momentom pri činnosti NN je pravé zmena váh $\Delta w_{i,j}$.



Obr. 7: Štruktúra doprednej NN

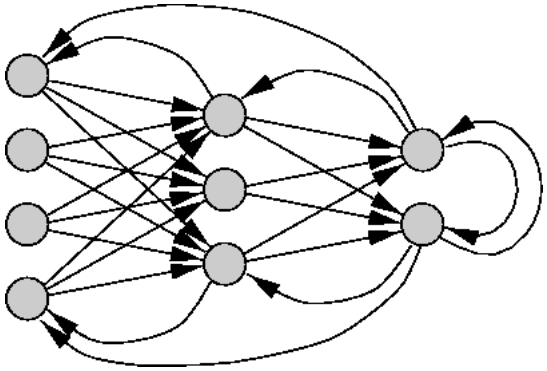
2.4.7 Topológia NN a spôsoby šírenia signálu

Vzhľadom k tomu, že všeobecné NN sa ťažko analyzujú, sú najprv študované a analyzované siete s nejakými pravidelnými štruktúrami. Jednou z pravidelných a dosť preskúmaných štruktúr je viacvrstvová štruktúra znázornená na obr. 7. V takýchto NN sú vrstvy pomenované nasledovne:

- vstupná vrstva – tu neuróny dostávajú vstup len z vonkajšieho sveta a výstup obvykle pokračuje k ďalším neurónom NN
- skrytá vrstva (hidden layer) – tu neuróny dostávajú vstup z ostatných neurónov alebo aj vonkajšieho sveta cez prahové prepojenia a ich výstupy pokračujú ďalej do NN
- výstupná vrstva – je podobná ako skrytá s tým rozdielom, že výstup z tejto vrstvy vyúsťuje do externého sveta.

Vzhľadom k tomuto faktu rozoznávame aj neuróny ako vstupné, skryté a výstupné. Pri návrhu NN vo všeobecnosti rozdeľujeme topológiu NN dvoch základných skupín:

- dopredné NN (feed – forward FF NN) – pri týchto NN sa signál šíri po orientovaných synaptických prepojeniach len jedným smerom a to dopredu – viď obr. 7.
- rekurentné NN (recurent RC NN) – tu je ťažké rozdeliť vrstvy a neuróny na vstupné, resp. výstupné. Niekedy neuróny v rekurentných sieťach predstavujú vstupné, ale aj výstupné typy neurónov a tým aj vrstiev viď obr. 8. Špeciálnym prípadom sú tzv. čiastočne rekurentné NN, v ktorých je stanovená určitá požiadavka na štruktúru prepojenia.



Obr. 8: Príklad štruktúry rekurentnej siete

2.4.8 Učenie neurónových sietí

Vo všeobecnosti rozdeľujeme prístupy k učeniu do dvoch veľkých skupín:

- kontrolované učenie - učenie s učiteľom (supervised learning), ktoré sa ďalej rozdeľuje do dvoch podskupín:
  - štrukturálne učenie - to sa delí na dve skupiny
    - autoasociačné učenie - na vstup aj výstup NN sa dáva tá istá vzorka. Takéto NN majú význam pri simulácii pamäte.
    - heteroasociačné učenie - NN sa učí že k vstupu α patrí výstup β
  - temporálne učenie - je to v podstate heteroasociačné učenie, ale výstup siete sa priraďuje za sebou nasledujúcej skupine vstupov. Napríklad reakcia na určitú postupnosť ťahov v šachu, alebo iba určitý časový vývoj ekonomických a iných parametrov môže znamenať rast akcií na burze a pod.
- nekontrolované učenie - učenie bez učiteľa (unsupervised learning).

Pri kontrolovanom učení je učiteľ prítomný počas celého procesu učenia. Prístupy ku zmene synaptických váh v prípade kontrolovaného učenia môžeme koncepčne rozdeliť do troch skupín:

- učenie na základe opravy chyby (error correction learning)
  určuje zmenu synaptických váh ako funkciu premennej e_i predstavujúcej rozdiel medzi očakávaným stavom neurónu a vypočítaným stavom neurónu v procese učenia

$$e_i = ev_i - x_i \tag{47}$$

Potom pre zmenu synaptických váh platí všeobecný vzorec

$$\Delta w_{ij} = \gamma x_j e_i \tag{48}$$

kde γ je parameter charakterizujúci rýchlosť učenia (learning rate).

- stochastické učenie (stochastic learning)
  Pri stochastickom type učenia sú zmeny synaptických váh založené na stochastických prístupoch. Príkladom takýchto NN je Boltzmanov stroj a jeho modifikácie. Ďalším príkladom je použitie genetických algoritmov.
  - navrhne sa stochastická zmena synaptických váh a vypočíta sa energia NN.
  - ak zmena priniesla zníženie energie NN, návrh zmeny sa príjme.
  - ak zmena nepriniesla požadovaný efekt, návrh sa zamietne.

- učenie na základe hodnotenia činnosti[3] (reinforcement learning)
  Tento spôsob učenia je podobný ako učenie na základe korekcie chyby, ale základným rozdielom je, že sa zhodnocuje stav výstupu celej výstupnej vrstvy pomocou nejakej skalárnej veličiny. Pre zmenu synaptických váh platí:

$$\Delta w_{ij} = \gamma(r - \theta_i)e_{ij} \tag{49}$$

kde r je skalárna hodnota úspešnosti celej NN odvodená z výstupnej vrstvy NN, $\theta_i$je prahový koeficient úpravy pre neurón "i" a e_{ij} je koeficient rozhodnutia a predstavuje zmenu pravdepodobnosti minimálnej chyby podľa synaptickej váhy, ktorý sa vo všeobecnosti vypočíta

$$e_{ij} = \frac{\partial ln\, g_i}{\partial w_{ij}} \tag{50}$$

kde g_i je pravdepodobnosť, že očakávaný výstup sa bude rovnať vypočítanému výstupu ev_{ij} (teda minimálnej chybe), teda

$$g_i = P(x_i = ev_i|\mathbf{W}_i, \mathbf{\Lambda}) \tag{51}$$

kde ev_i je očakávaná hodnota neurónu, x_i je vypočítaná hodnota neurónu,$\mathbf{W}_i$ je vektor synaptických váh, ktoré vstupujú do neurónu "i" a $\mathbf{\Lambda}$je vektor hodnôt aktivačných stavov neurónov, ktorých synaptické spoje vstupujú do neurónu "i".

[3] voľný preklad

Pri nekontrolovanom učení sa NN počas učenia dostáva údaje na vstup a zmeny synaptických váh sa dejú samostatne na základe určených pravidiel. Z tohto dôvodu sa siete používajúce takéto metódy učenia samo-organizujúce sa siete (self-organising NN). Rozoznávame dva základné typy nekontrolovaného učenia:

- Hebbovo učenie (Hebbian learning) sa realizuje do podoby dvoch zásad:
  - ak sú dva neuróny na opačných stranách synapsie aktivované naraz (synchrónne), potom sa synaptická váha danej synapsie zvýši.
  - ak sa dva neuróny na opačných stranách synapsie aktivizujú v rôznych časoch (asynchrónne), synaptická váha danej synapsie sa zníži.
- Kooperačné a konkurenčné učenie - ak sledujeme NN z hľadiska dynamiky, neurón s najväčšou hodnotou sa stáva víťazom a je nastavený na 1 a ostatné neuróny sú nastavené na 0. Následne sa upravia len tie synaptické váhy, ktoré smerujú k víťaznému neurónu. Konkurenčné metódy učenia sa vhodne využívajú zo zámerom zhlukovania vstupných dát.

2.4.9 Kontrované učenie dopredných sieti

Metóda najstrmšieho zostupu - V tejto bude popísaný základný matematický prístup, ktorý sa v teórii NN považuje za klasický. Ide o prístup k výpočtu zmeny SV v NN cez chybovej funkcie metódou najstrmšieho zostupu.

- Wienerov filter :

Nech je n-senzorov umiestnených na rôznych miestach. Tieto senzory produkujú signály $in_1, \ldots, in_n$. Signály sa potom s určitými váhami integrujú do výstupného neurónu. Na výstupe sa požaduje nejaký výsledok, ktorý v procese učenia dobre poznáme ev. Teda ak ou je výstup z výstupného neurónu, tak

$$ou \;=\; \sum_{k=1}^{n} w_k in_k \tag{52}$$

Nech je známy chybový rozdiel

$$e \;=\; ev \;-\; ou \tag{53}$$

Už v úvode je potrebné poukázať na principiálnu rozdielnosť voči perceptrónu tým, že kým pri perceptróne na vstup vstupovali vstupy iba z dvoch rôznych tried, tu môžu byť usporiadané dvojice (in, ev) patriace do mnohých skupín a v podstate sa tu nejedná o

dichotómiu. Topologický sú z pohľadu NN Wienerov filter a perceptrón veľmi podobné. Teda môžeme definovať všeobecnú chybovú funkciu v tvare

$$J \;=\; 0.5\; E(e^2) \tag{54}$$

kde $E(e^2)$ je stredná hodnota kvadrátov všetkých rozdielov. Základným problémom je teda nájsť také $w_1, \ldots, w_n$, pri ktorých $J \to 0$. Takýto filter v oblasti spracovania signálov sa nazývam Wienerov filter. Do určitej miery je pojem filter trocha mätúci, ale ide o filtráciu vstupov. Je potrebné to chápať ako vhodné priradenie výstupov k jednotlivým vstupom. Ak sa do rovnice (54) dosadí (53) resp. (**??**), výsledkom bude nasledovný výraz:

$$J \;=\; 0.5\; E(ev^2) - E(\sum_{j=1}^{n} w_i in_i ev) + 0.5E(\sum_{j=1}^{n}\sum_{k=1}^{n} w_j w_k in_j in_k) \tag{55}$$

Jednotlivé funkcie v rovnici (55) sú

- $E(ev^2) \;=\; r_{ev}$stredná hodnotu očakávaného výsledku ev.
- $E(in_j ev) \;=\; r_{in,ev}(j)$je kroskorelačná funkcia medzi vstupmi in a očakávanými výstupmi ev.
- $E(in_j in_k) \;=\; r_{in,in}(jk)$je auto-korelačná funkcia medzi samotnými vstupmi

Potom sa môže rovnica (55) prepísať do tvaru

$$J \;=\; 0.5 r_{ev} - \sum_{k=1}^{n} w_k r_{in,ev}(k) + 0.5 \sum_{j=1}^{n}\sum_{k=1}^{n} w_j w_k r_{in,in}(jk) \tag{56}$$

Teda pri hľadaní váh, ktoré by minimalizovali chybovú funkciu sa môže napísať že

$$\frac{\partial J}{\partial w_l} \;=\; -r_{in,ev}(l) + \sum_{j=1}^{n} w_j r_{in,in}(jl) \tag{57}$$

potom logicky pri hľadaní váh sa počíta s podmienkou $\frac{\partial J}{\partial w_l} \;=\; 0$ a dostávame z (57)

$$\sum_{j=1}^{n} w_j r_{in,in}(jl) \;=\; r_{in,ev}(l) \tag{58}$$

Potom systém l-rovníc (58) o n-neznámych $w_j, l = 1, \ldots, n$ nazývame Wienerovým systémom rovníc a filter s vypočítanými váhami voláme v teórii signálov Wienerovým filtrom.

- Metóda najstrmšieho zostupu

K tomu, aby sme vedeli vyriešiť rovnice (58), by sme potrebovali vypočítať maticu (n x n) a jej inverziu , čo je dosť náročný výpočet. Existuje aj iná forma výpočtu hľadaných SV a to metódou najstrmšieho zostupu (steepest descent). Tento výpočet sa realizuje iteračným spôsobom v t-iteráciách a počíta sa zmena SV, ktorá sa môže vyjadriť nasledovne pre synapsiu "i"

$$\Delta w_i(t) \;=\; -\gamma \frac{\partial J(t)}{\partial w_i(t)} \tag{59}$$

1. kde γ je učiaci pomer, potom hľadanú váhu v iterácii "t+1" vypočítame

$$w_k(t+1) \;=\; w_i(t) \;+\; \Delta w_i(t) \tag{60}$$

2. potom z rovníc (57) a (59) dostaneme z rovnice (60) upravený tvar

$$w_i(t+1) \;=\; w_i(t) \;+\gamma(r_{in,ev}(k,t) - \sum_{j=1}^{n} w_j(t) r_{in,in}(jk,t)) \tag{61}$$

3. kde $i = 1, \ldots, n$, n -je počet senzorov. Teda v konečnom dôsledku sa nájde príslušné SV po určitom počte iterácii aj takým spôsobom, avšak výpočtová náročnosť vzorca (61) je dosť veľká vzhľadom na funkcie $r_{ev,in}$ a $r_{in,in}$. V prípade, že si graficky zobrazíme chybovú funkciu $J(t)$ v závislosti od jednotlivých váh $w_1, \ldots, w_n$, dostali by sme hyperplochu, ktorá sa nazýva povrch chybovej funkcie (error surface). Táto zvlnená hyperplocha má svoje globálne minimum, ktoré zodpovedá nejakým $w_1, \ldots, w_n$ a to sú práve hľadané SV. Pod filtrom budeme rozumieť množinu hľadaných synaptických váh.

- Metóda najmenšej kvadratickej chyby

Metóda najmenšieho stredného kvadrátu (least-mean-square ďalej LMS) je založená na okamžitých odhadoch funkcii $r_{in,in}$ a $r_{in,ev}$ a to aproximáciou odhadov nasledovnými výrazmi v iterácii "t":

$$\hat{r}_{in,in}(ji,t) \;=\; in_j(t) in_i(t) \tag{62}$$

$$\hat{r}_{in,ev}(i,t) \;=\; in_i(t) ev(t) \tag{63}$$

Tieto odhady $\hat{r}_{in,in;ev,in}$ sa spočítavajú v každej iterácii. Po dosadení do rovnice (61) pre odhady jednotlivých SV, potom

$$\hat{w}_i(t+1) \;=\; \hat{w}_i(t) \;+\gamma(in_i(t)ev(t) - \sum_{j=1}^{n} \hat{w}_j in_l(t) in_i(t) \tag{64}$$

čo sa dá upraviť do tvaru

$$\hat{w}_i(t+1) \;=\; \hat{w}_i(t) \;+\gamma(ev(t) - \underbrace{\sum_{j=1}^{n} \hat{w}_i in_l(t))}_{ou(t)}\, in_i(t) \tag{65}$$

Teda konečný tvar rovnice (65) je

$$\hat{w}_i(t+1) \;=\; \hat{w}_i(t) \;+\gamma \underbrace{(ev(t) - ou(t))}_{e(t)}\, in_i(t) \tag{66}$$

čo je pravidlo výpočtu nových hodnôt SV podľa LMS. Teda môže sa LMS popísať v nasledovných krokoch:

- inicializácia $\forall j \;= 1, \ldots, n; w_j(t=0) \;=\; 0$
- filtrácia pre $t \;= 1, \ldots$ vypočíta sa

1. $y(t) \;=\; \sum_{j=1}^{n} \hat{w}_j(t) in_j(t)$
2. $e(t) \;=\; ev(t) \;-\; y(t)$
3. $\hat{w}_i(t+1) \;=\; \hat{w}_j(t) + \gamma e(t) in_i(t)$pre $k = 1, \ldots, n$
4. zvýši sa (t+1) a návrat do bodu 1

- Adaline

Adaline bol popísaný Widrowom a Hoffom v roku 1960. Predstavuje najjednoduchšiu NN, topológiou zhodnú s JPR. Principiálny rozdiel je v učiacej funkcii, kým v prípade Adaline ide o učenie typu LMS a v prípade JPR ide o iné učenie. Vstup do výstupného neurónu má tvar

$$in(t) \;=\; \sum_{j=1}^{n} w_j(t) in_j(t) \tag{67}$$

a potom výstup

$$ou(t) \;=\; \begin{cases} 1 & ak\ in(i) \geq 0 \\ -1 & ak\ in(t) < 0 \end{cases}$$

- Backpropagation - metóda spätného šírenia chyby

V predošlej časti matematicky odvodené DP vlastne predstavuje základ učenia so spätným šírením chyby a umožňuje použitie v podstate ľubovoľnej aktivačnej funkcie f aj nelineárneho typu, ktorá splňuje podmienku diferencovateľnosti, t.j. platí

$$x \;=\; f(in) \;\neq\; in \tag{68}$$

Ide teda znova o určovanie zmeny SV pre NN s nelineárnymi neurónmi. Postup bude analogický ako pri základnom DP, avšak o funkcii f sa predpokladá, že nie je lineárna a je diferencovateľná. Teda opäť stav neurónu "i" pri ľubovoľnom vstupe do NN má tvar

$$x_i(t) \;=\; f(in_i(t)), \tag{69}$$

kde

$$in_i(t) \;=\; \sum_{j=1}^{M} w_{ij}(t)x_j(t) \;+\; \theta_i. \tag{70}$$

Z predchádzajúceho DP je známe, že

$$\Delta w_{ij}(t) \;=\; -\gamma \frac{\partial J(t)}{\partial w_{ij}(t)}. \tag{71}$$

$J(t)$ má tvar

$$J(t) \;=\; 0.5 \sum_{j=1}^{N_0} (ev_i(t) - x_i(t))^2, \tag{72}$$

kde N_0 je počet neurónov vo výstupnej vrstve NN. Samotný výpočet parciálnej derivácie chybovej funkcie podľa príslušnej SV má tvar

$$\frac{\partial J(t)}{\partial w_{ij}(t)} \;=\; \frac{\partial J(t)}{\partial in_i(t)} \frac{\partial in_i(t)}{\partial w_{ij}(t)} \tag{73}$$

Nech

$$\frac{\partial J(t)}{\partial in_i(t)} \;=\; -\delta_i(t) \tag{74}$$

a

$$\frac{\partial in_i(t)}{\partial w_{ij}(t)} \;=\; x_j(t), \tag{75}$$

výsledkom bude obvyklý zápis výpočtu zmeny SV v tvare

$$\Delta w_{ij}(t) \;=\; \gamma \delta_i(t) x_j(t) \tag{76}$$

Základným problémom je teraz stanovenie príslušného δ_i pre každý neurón NN. Vedie to k jednoduchému rekurzívnemu vzťahu pre výpočet jednotlivých δ_i, ktoré predstavujú spätné šírenie chyby smerom od výstupu NN. Pre príslušné $\delta_i(t)$ sa môže ďalej písať na základe (74)

$$\delta_i(t) \;=\; -\frac{\partial J(t)}{\partial in_i(t)} \;=\; -\frac{\partial J(k)}{\partial x_i(t)} \frac{\partial x_i(t)}{\partial in_i(t)} \tag{77}$$

Najprv sa vyrieši druhý člen pravej strany (77). Vzhľadom na nelineárny neurón je zrejmé, že je možné napísať pomocou (69), že

$$\frac{\partial x_i(t)}{\partial in_i(t)} \;=\; f^{'}(in_i(t)) \tag{78}$$

Pre výpočet prvého člena z rovnice sa berú do úvahy dva rôzne prípady :

- ak neurón "i" je výstupným neurónom - vtedy je to pomerne jednoduché, lebo hľadaná parciálna derivácia má tvar

  $$\frac{\partial J(t)}{\partial x_i(t)} \;=\; -(ev_i(t) - x_i(t)) \tag{79}$$

  a tým je výpočet $\delta_i(t)$ pre tento prípad vyriešený pomocou (78) a (79) v tvare

  $$\delta_i(t) \;=\; (ev_i(t) - x_i(t)) f^{'}(in_i(t)) \tag{80}$$

- ak neurón "i" nie je výstupným neurónom - výpočet je trocha zložitejší a postupuje sa takto

  $$\frac{\partial J(t)}{\partial x_i(t)} \;=\; \sum_{h=1}^{N_0} \frac{\partial J(t)}{\partial in_h(t)} \frac{\partial in_h(t)}{\partial x_i(t)} \tag{81}$$

  kde N_0 je počet neurónov vo výstupnej vrstve. Z matematického hľadiska pri výpočte derivácie chybovej funkcie J, ktorá popisuje celkovú chybu na výstupnej vrstve, podľa $x_i(t)$, je nutné vyjadriť J ako funkciu $x_i(t)$. Preto rovnica (81) má takýto tvar. Súčasne prvý člen pravej strany je jasný z rovnice (74) a teda platí, že

  $$\frac{\partial J(t)}{\partial x_h(t)} \;= \frac{\partial x_h(t)}{\partial in_h(t)} \;=\; -\delta_h(t) \tag{82}$$

Tu je potrebné poznamenať, že $x_h(t)$ v rovnici (82) je z inej vrstvy ako $x_i(t)$ v rovnici (81). Čo sa týka druhého člena rovnice (81), tam samotný člen $in_h(t)$ predstavuje vstup do výstupného neurónu "h" a môže sa nahradiť nasledovne

$$in_h(t) \;=\; \sum_{l=1}^{N_z} w_{hl}(t)x_l(t) \tag{83}$$

avšak parciálna derivácia $in_h(t)$ podľa $x_i(t)$ znamená, že jedno z $l = i$ a tým

$$\frac{\partial \sum_{l=1}^{N_z} w_{hl}(t)x_l(t)}{\partial x_i(t)} \;=\; w_{hi}(t) \tag{84}$$

teda v konečnom dôsledku

$$\frac{\partial J(t)}{\partial x_i(t)} \;=\; -\sum_{h=1}^{N_0} \delta_h(t)w_{hi}(t) \tag{85}$$

a konečne hľadaný koeficient $\delta_i(t)$ bude mať tvar na základe (78) a (85)

$$\delta_i(t) \;=\; f^{'}(in_i(t)) \sum_{h=1}^{N_0} \delta_h(t)w_{hi}(t) \tag{86}$$

Je potrebné dobre si všimnúť rekurzívnosť tohoto vzťahu. Ide o výpočet koeficientu $\delta_i(t)$ neurónu "i", ktorý nie je výstupným neurónom. Vypočíta sa za pomoci $\delta_h(t)$, ktoré prichádzajú z vrstvy napravo od neurónu "i" a ich počet je N_0. Existuje modifikácia vzťahu (86) a to v tvare

$$\delta_i(t) \;=\; (f^{'} \;+\; c)(in_i(t)) \sum_{h=1}^{N_0} \delta_h(t)w_{hi}(t) \tag{87}$$

kde parameter c je tzv.parameter rovinnosti, ktorý rieši prípad, ak chyba sa nachádza na rovinnej časti chybovej plochy.

Metóda spätného šírenia chyby je najbežnejším spôsobom učenia NN. Aj keď sa teória s ňou spojená zdá zložitá, postup implemenntácie je veľmi jednoduchý, založený je na úprave synaptických váh na základe opravy chýb. Jedinou podmienkou je, že aktivačná funkcia musí byť spojitá a derivovateľná na celom definičnom obore. Pre úpravu váh platí

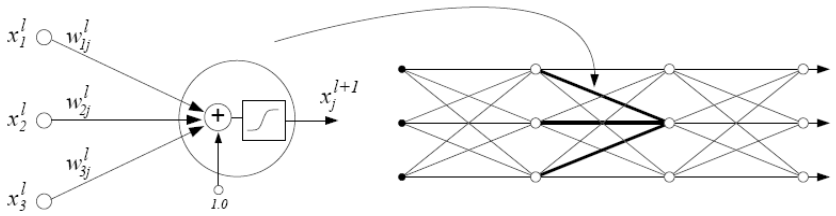
$$w_{ij}^{l}(t+1) = w_{ij}^{l}(t) - \gamma\delta_j^{l+1}(t)x_i^{l}(t) \tag{88}$$

$$\delta_j^l(t) = \begin{cases} -2e_j(t)f'(y_j^L(t)) & l = L \\ f'(y_j^l(t)) \sum_{m=1}^{N_{l+1}} \delta_m^{l+1}(t) w_{jm}^{l+1} & 1 \leq l \leq L-1 \end{cases} \tag{89}$$

kde γ určuje rýchlosť učenia, $f'(y)$ je derivácia aktivačnej funkcie, l je vrstva, y je výstup neurónu a δ je chyba na danom neuróne danej vrstvy. Postup je následovný:

1. vyráta sa δna výstupných neurónoch
2. obdobne ako pri šírení signálu sa pomocou synaptických spojov a váh, chyby prešíria v opačnom smere k vstupnej vrstve.
3. na základe vyrátaných chýb na každom neuróne sa upravia synaptické váhy.

2.4.10 Dopredné NN implementujúce FIR synaptické filtre



Obr. 9: Statický model doprednej siete

Na obr.9 je zobrazený model tradičnej statickej doprednej NN. Skladá sa z umelých neurónov usporiadaných do vrstiev, kde každý neurón danej vrstvy má synaptické spojenie s každým neurónom nasledujúcej vrstvy. Takisto je zobrazený jeden neurón z l-tej vrstvy. Vstupy do tohto neurónu sú vynásobené váhovými koeficientami, reprezentujúc synaptické spoje medzi neurónami. Výstup neurónu sa vyráta ako hodnota sigmoidálnej funkcie váženej sumy vstupov.

$$x_j^{l+1} = f(\sum_i w_{ij}^l x_i^l) \tag{90}$$

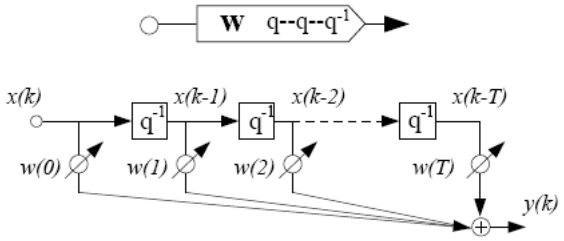
Učenie takéjto siete sa dá ľahko dosiahnuť pomocou metódy "backpropagation".

Model doprednej NN popísaný vyššie vytvára komplexný vzťah medzi vstupom na prvej vrstve a výstupom na vrstve poslednej. Aj tak sa však jedná o statické priradenie, neobsahuje žiadnu vnútornú dynamiku. Modifikáciu statického neurónu dosiahneme

nahradením jeho synaptických váh lineárnym FIR filtrom v tvare:

$$y(k) = \sum_{n=0}^{T} w(n)x(k-n). \tag{91}$$

Treba si všimnúť, že to korešponduje s MA komponentom ARMA modelu viď rovnica:37. Pre takýto filter platí, že jeho výstup je vážený priemer minulých hodnôt.



Obr. 10: Model FIR Filtra

Koeficienty takéhoto synaptického filtra, čiže povedzme FIR synapsie budú definované vektorom $\mathbf{w}_{ij}^l = \left[w_{ij}^l(0), w_{ij}^l(1), ..., w_{ij}^l(T^l)\right]$ .Podobne $\mathbf{x}_i^l = \left[x_i^l(k), x_i^l(k-1), ..., x_i^l(k-T^l)\right]$ reprezentuje vektor oneskorených stavov pozdĺž synaptického filtra. To nám umožňuje vyjadriť operáciu filtra ako výsledok operácie skalárneho súčinu $\mathbf{w}_{ij}^l.\mathbf{x}_i^l(k)$. Výstup neurónu na vrstve l bude potom hodnotou sigmoidálnej funkcie sumy výstupov všetkých filtrov.

$$x_j^{l+1}(k) = f(\sum_i \mathbf{w}_{ij}^l.\mathbf{x}_i^l(k)) \tag{92}$$

Princíp siete zostáva nezmenený, len namiesto skalárov pracujeme s vektormi. Táto podobnosť nám umožňuje na učenie použiť modifikovanú "backpropagation" metódu, takzvanú "temporal backpropagation" metódu **?**].

Úlohou učenia siete je minimalizovať chyby. Najjednoduchší spôsob je stochastická gradientová metóda. Synaptické filtre budú upravované pri každom kroku podľa

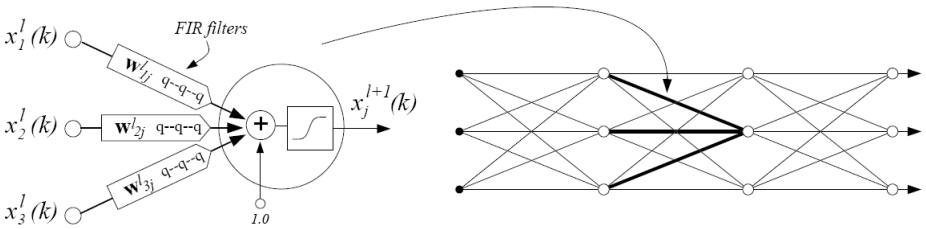
$$\mathbf{w}_{ij}^l(k+1) = \mathbf{w}_{ij}^l(k) - \gamma \frac{\partial e^2(k)}{\partial w_{ij}^l(k)} \tag{93}$$

kde γurčuje rýchlosť učenia. Po úpravách pre zmenu parametrov synaptických filtrov a výpočet δplatí:

$$\mathbf{w}_{ij}^l(t+1) = \mathbf{w}_{ij}^l(t) - \gamma \delta_j^{l+1}(t)\mathbf{x}_i^l(t) \tag{94}$$

$$\delta_j^l(t) = \begin{cases} -2e_j(t)f'(y_j^L(t)) & l = L \\ f'(y_j^l(t)) \sum_{m=1}^{N_{l+1}} \boldsymbol{\delta}_m^{l+1}(t)\mathbf{w}_{jm}^{l+1} & 1 \leq l \leq L-1 \end{cases} \tag{95}$$

Kde parameter $\boldsymbol{\delta}_m^{l+1}(k) = [\delta_m^l(k), \delta_m^l(k+1), ..., \delta_m^l(k+T^{l-1})]$ je vektor šíriacich sa gradientov. Tieto rovnice sú takmer totožné s rovnicami (88) a (89), ak nastavíme dĺžku FIR filtra pre každú synapsiu na 1, naozaj dostaneme statický model. Čiže ich môžme považovať za vektorové zovšeobecnenie rovníc pre statickú sieť. Ďalším rozdielom oproti statickému modelu je spôsob šírenia chyby. Na výpočet chyby na určitom neuróne, prefiltrujú sa chyby z následujúcej vrstvy FIR synapsiami na vrstvy predošlé viď Obr. 11.

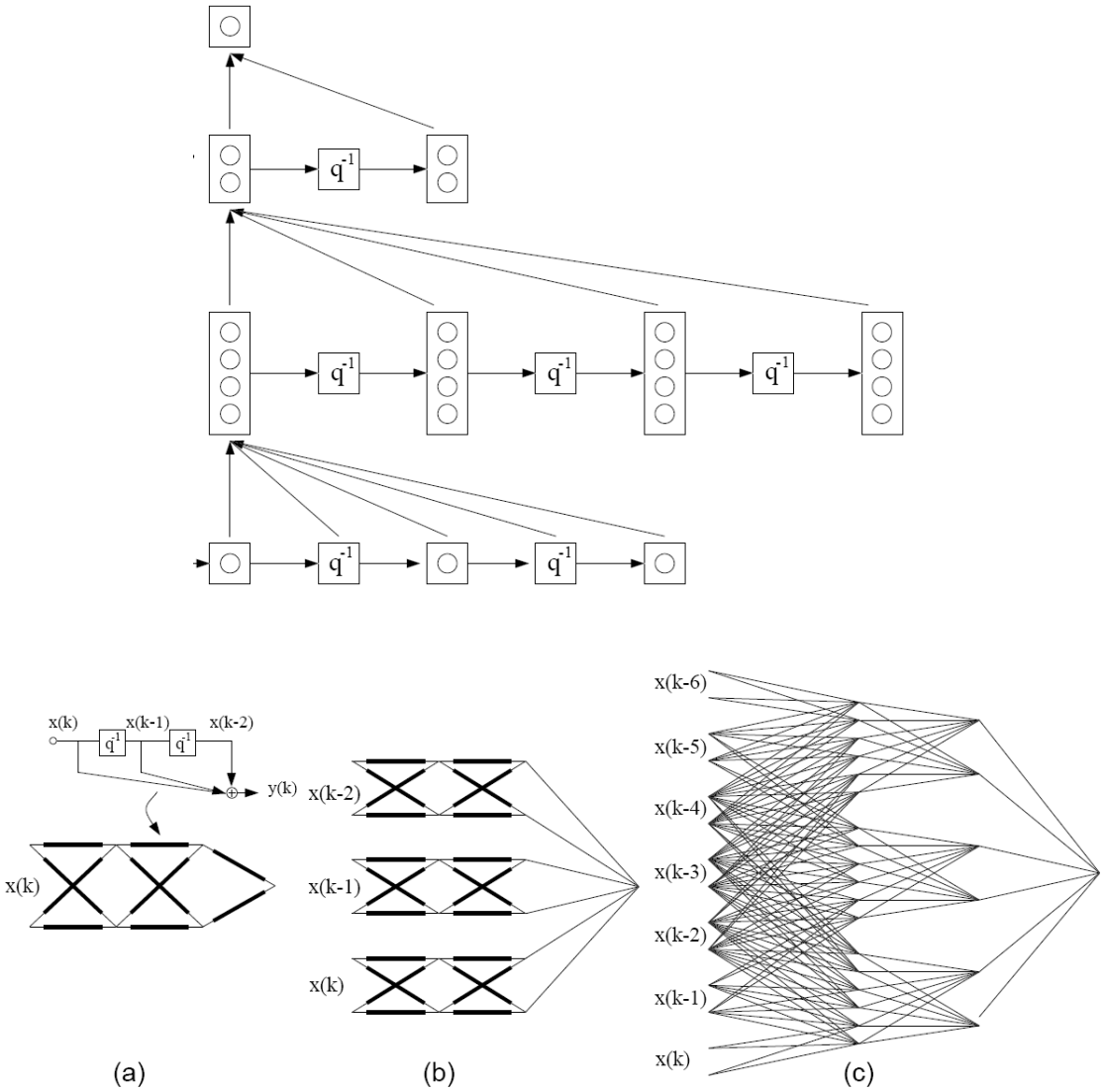


Obr. 11: FIR Neurón a sieť

Existujú alternatívne reprezentácie FIR topológie. Takzvaná "Time Delay NN" viď Obr. 12, pri ktorých sú všetky výstupy danej vrstvy ukladané na určitú danú dobu. Avšak aktuálne aj uložené výstupy sú šírené na ďalšiu vrstvu. Táto štruktúra je funkčne ekvivalentná s FIR topológiou. Výhodou FIR topológie je však implementácia metódy "temporal backpropagation" ako rozšírenie klasickej metódy.

Ďalšou alternatívnou reprezentáciou FIR (a TDNN) je technika nazývaná rozvinutie v čase (unfolding in time). Zámerom je zbaviť sa všetkých oneskorení. Namiesto ukladania predošlých hodnôt sa na vstupné neuróny dáva vstup vo forme časového okna viď. Obr. (13).

Obr. 12: Time Delay NN



Obr. 13: Rozvinutá FIR NN

Pri tomto prístupe rastie počet operácií s rastom takejto siete geometrickým radom z množstvom vrstiev.

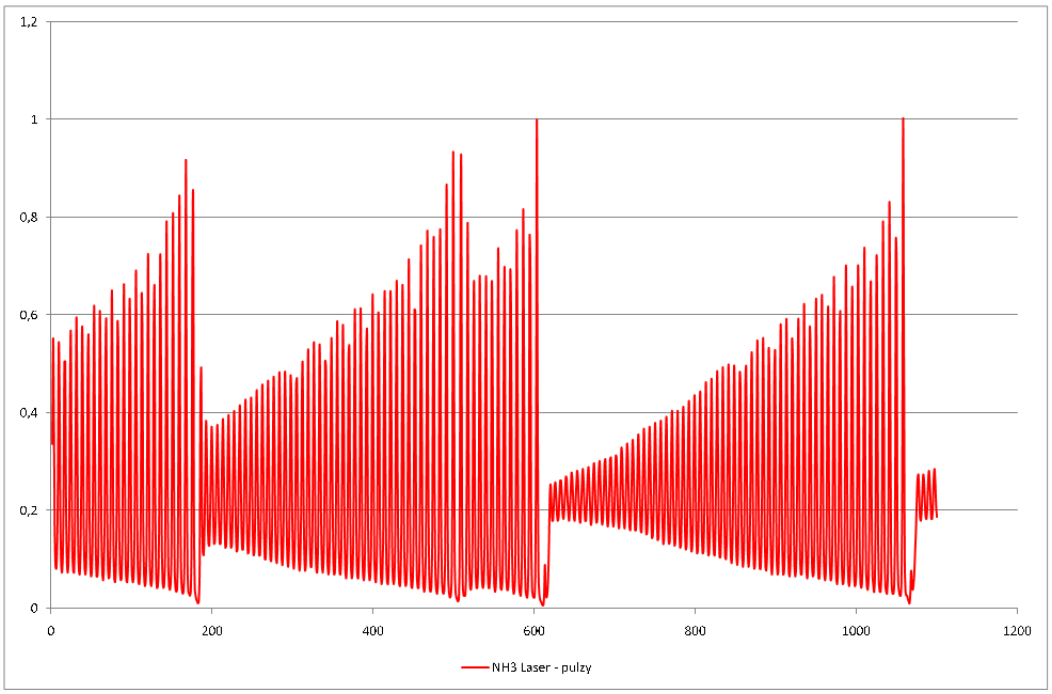
"Temporal backpropagation" zachováva symetriu medzi šírením signálu a spätným šírením chyby. Navyše, počet operácií počas simulácie rastie s počtom vrstiev v sieti lineárne na rozdiel od rozvinutej statickej verzie. Z rôznych reprezentácií je FIR NN výpočtovo najmenej náročná.

Rozmery siete		Počet premenných	Statický ekvivalent
Počet uzlov	FIR filtre		
2x2x2x1	2:2:2	30	150
5x5x5x5	10:10:10	605	36355
3x3x3	9:9	180	990
3x3x3x3	9:9:9	270	9990
3^n	9^{n-1}	(n-1)90	$10^n - 10$

Tabuľka 1: FIR NN oproti statickému ekvivalentu. **(author?)** [1]

3 Experimentálna časť

V tejto časti sú praktický porovnané predikčné schopnosti dopredných neurónových sieti používajúcich FIR synaptické filtre a ARMA metódy. Na testovanie metód sme použili časové rady z meraní chaotických pulzácii NH_3 lasera Obr. 14



Obr. 14: Chaotické pulzácie infračerveného NH_3lasera

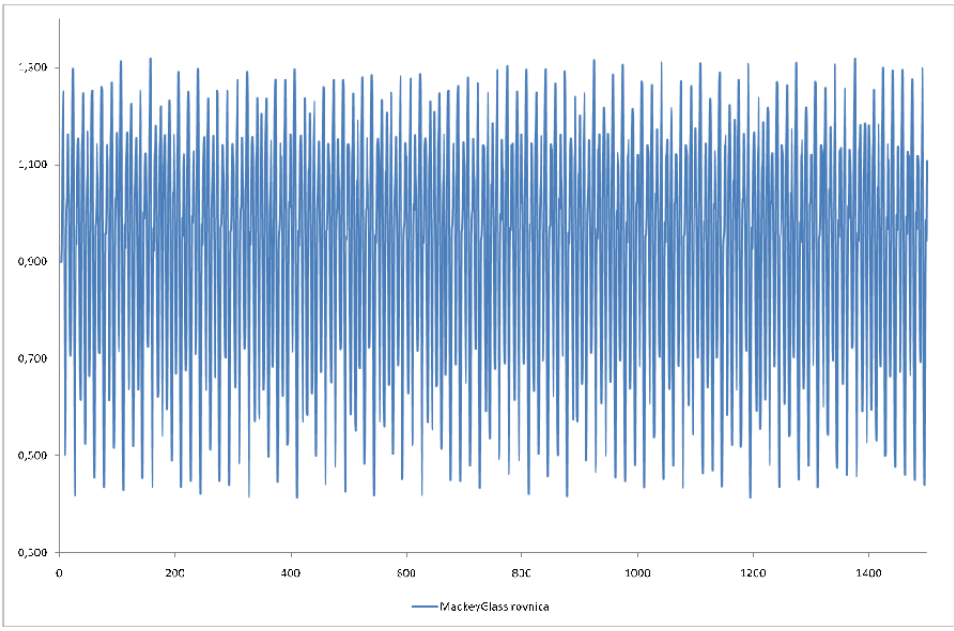
Ďalším súborom údajov je Mackey-Glass rovnica, ktorá vykazuje známky chaotického správania Obr. 15.

Prvá časť je venovaná implementácii FIR NN v jazyku Java.

3.1 Implementácia FIR doprednej siete v jazyku Java

Úlohou tejto práce nie je len porovnať štatistické metódy s predikciou pomocou neurónovej siete, ale aj implementovať tieto algoritmy v reálnej aplikácii.

Jazyk Java vznikol ako projekt v roku 1991 za účelom vytvorenie robustného, objektovo orientovaného, dynamického a na architektúre nezávislého programovacieho jazyka. Naprogramovať tuto úlohu v takej podobe v akej je, by v jazyku C++, alebo inom nízko úrovňovom, neinterpretovanom jazyku trvalo neporovnateľne dlhšie. Táto rýchlosť vývoja, je však na úkor výkonu aplikácie. Aj keď výkon nebol prioritou, počas tvorby



Obr. 15: Mackey-Glass

programu bol braný ohľad aj na tento aspekt. Už samotné rozdelenie úlohy na dve časti je spôsobené potrebou vyššieho výkonu. Jednoduchý program na používanie z príkazového riadku, v C++ vykoná 90000 iterácií za pár minút. Tá istá konfigurácia siete implementovaná v Jave potrebuje niekoľko hodín. Je pravda, že programy nie sú identické, čiže toto porovnanie nie je presné, ale dáva predstavu o rozdiele vo výkone.

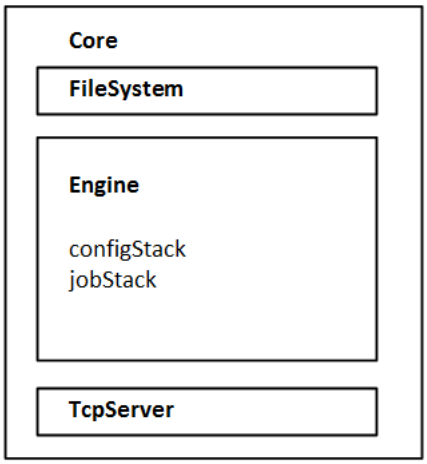
Obidve množiny údajov boli normalizované na strednú hodnotu 0 s použitím vzťahu

$$y_i = (x_i - \mu)/\sigma^2 \tag{96}$$

kde μ je stredná hodnota pôvodného radu a σ^2 je jeho variancia. Následne boli údaje upravené funkciou hyperbolický tangens. Po týchto úpravách sa hodnoty oboch časových radov nachádzali v intervale $(-1; 1)$ so strednou hodnotou približne sa rovnajúcou 0. Takto upravené hodnoty boli potom použité pri predikciách ARMA modelom aj FIR NN.

3.1.1 Návrh

Program je navrhnutý tak aby sa dal využiť potenciál viacerých počítačov, procesorov, alebo procesorových jadier. Preto sa skladá zo serverovej časti, ktorá obsahuje celú NN logiku a obslužné metódy, neobsahuje však žiadne GUI a spúšťa sa z príkazového riadku a klienta, ktorý má grafické rozhranie a dokáže sa pripájať na servery bežiace na vzdialených systémoch. Výhodou je, že sa môže spustiť na vzdialených systémoch a vďaka tomu, že Java je nezávislá na platforme a operačnom systéme, je možné bežať niekoľko simulácií naraz na skoro každom dostupnom počítači. To bolo potrebné z dôvodu veľkej výpočtovej náročnosti, nakoľko pri takom veľkom množstve voliteľných premenných je vyhľadávanie optimálnej konfigurácie siete práve tou najťažšou úlohou. Na vyhľadávanie optimálne konfigurácie neboli použité žiadne sofistikované metódy, použitá bola jednoduchá iterácia cez niekoľko možných konfigurácií.



Obr. 16: FIR NN Server

Na obrázku (16)sú zobrazené hlavné komponenty serverovej časti. Triedy neslúžiace simulácii sú poväčšinou abstraktné, Engine je abstraktná trieda obsahujúca dve hlavné zásobníky. V configStack sú uložené skupiny iteračných konfigurácií, na základe ktorých sa vytvárajú jednotlivé úlohy. V zásobníku jobStack sú už existujúce úlohy. Úloha môže byť nová vytvorená sieť, čiastočne naučená sieť, alebo plne naučená sieť používaná na predikcie.

Principiálne sú aplikácie rozdelené do štyroch balíkov:

- sk.hunci.nnet.core - definície tried metód používaných na kontrolu simulácií
- sk.hunci.nnet.net - triedy a metódy samotných komponentov neurónovej siete

- sk.hunci.nnet.server - metódy na komunikáciu po sieti a prácu so súbormi
- sk.hunci.nnet.client - je GUI a triedy klienta

3.1.2 Implementácia neurónovej siete

Sieť je reprezentovaná presne tak ako v teórii. Umožňuje to objektovo orientovaný prístup. Triedy sú v balíku *sk.hunci.nnet.net*

- trieda SigmoidNeuron reprezentuje samotný neurón a z názvu je zrejmé, že použitá aktivačná funkcia je sigmoida.

Algorithm 1 prehľad metód NN

```
    protected double potential;
    protected double delta;
    protected LinkedList<Connection> axons;
    protected LinkedList<Connection>  dendrites;
    //alfa a bias by teoreticky mohli byt rozdielne pre kazdy neron, gama je vlastnost sie
    public Neuron(FirNetData d, int fl) {
    public void reset() {
    public void clear() {
    public void activate(double input) {
    public void backprop(double input) {
    public double fire() {
    protected void computeDelta() {
    protected void computeDelta(double expected) {
    public void backprop() {
    public void adjust() {
    public void connectTo(Neuron n) {
    public void connectFrom(Connection c) {
```

- trieda Synapse reprezentuje synaptický FIR filter

Algorithm 2 metódy a premenné synaptického FIR filtra

```
public static final long serialVersionUID = (long)Engine.VERSION;
private Neuron src;
private Neuron dst;
private double[] w;
private double[] x;
private double[] d;
private double firSum = 0;
public Connection(int length, Neuron from, Neuron to) {
public static double wRand(int range) {
public void clear() {
public void backprop(double input) {
public void adjust(double delta, double gama) {
private double fir(double[] in) {
```

- trieda Layer reprezentuje vrstvu a obsahuje metódy na spájanie vrstiev do sieti
- trieda FirNetData obsahuje trénovacie aj predikčné údaje a všetky parametre siete $(\alpha, \gamma, ...)$
- trieda FirNetwok reprezentuje FIR NN

3.1.3 Server

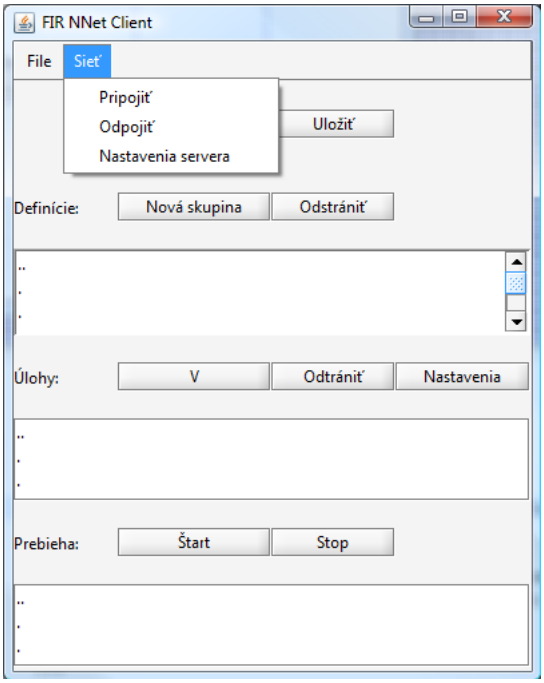
Balík *sk.hunci.nnet.server* obsahuje iba abstraktné triedy. Má dve úlohy:

- Umožniť ukladať a nahrávať údaje, ale aj konfigurácie, učiace sa siete, alebo už siete naučené z a do súborového systému počítača. Tieto triedy sú používane aj klientom za tým istým účelom.
- Komunikáciu po sieti (internete) s klientom. Na to sa používa schopnosť Javy serializovať objekty. Používa sa jednoduchý protokol reprezentovaný rozhraním Protocol. Posielané objekty slúžia buď iba na prenos príkazu, alebo na prenos dát. Triedy reprezentujúce tento komunikačný protokol sú:
  - FIN ukončenie spojenia
  - PSH odoslanie dát serveru
  - REM odstránenie dát na serveri
  - STR príkaz na uloženie všetkých dát na strane servera
  - SYN synchronizácia medzi klientom a serverom
  - RUN spustenie simulácie

3.1.4 Core

Balík *sk.hunci.nnet.core* obsahuje hlavnú triedu obsahujúcu metódu *main(Strin[] args). Ďalšie triedy sú:*

- ConfigGroup je LinkedList trieda obsahujúca skupiny iteračných konfigurácii a metódy na ich správu
- Engine je hlavná simulačná statická trieda obsahujúca zásobníky s úlohami a konfiguráciám. Stará sa o beh jednotlivých úloh a spúšťanie nových.
- Job dedí po triede Thread. Je to samotná simulácia. Aby sa využili možnosti dnešnej výpočtovej techniky, simulácie bežia ako Java thready.



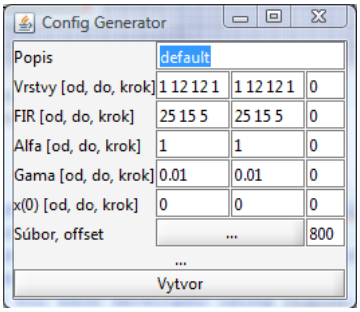
Obr. 17: FIR NN Klient

```
Algorithm 3 metódy simulačnej triedy Job
    public Job(NetData data) {
    public void init() {
    public void train(int cycles) {
    private double[] normalize(double in[]) {
    public void train(int maxCycles, double limit) {
    public void singleStep(double[] data) {
    private double mape(double current, double estimation) {
    private double mse(double current, double estimation) {
    public void run() {
```

- NetData je rozhranie definujúce konfigurácie siete
- NNet je rozhranie definujúce NN

3.1.5 Klient

Klient je jednoduchá GUI aplikácia, viď Obr. (17) umožňujúca spravovať bežiace inštancie FIR NN serverov, či už lokálne, alebo na vzdialených systémoch. Obsahuje metódy na vytváranie iteračných skupín rôznych NN topológií.



Obr. 18: Generátor topológií

Nesynchronizuje sa s pripojeným serverom automaticky, aby zbytočne nezaťažovala server spracovaním iných informácií okrem simulácií.

3.2 Štatistické metódy

Na štatistické výpočty sa použil program *gretl* a ARMA plugin programu Microsoft Excel. ARMA parametre sa odhadli na tej istej množine na ktorej sa učili siete a predikcia bola porovnaná so zvyšnými 100hodnotami.

Pre prvú vzorku sa na základe Akalike kritérií odhadli parametre ARMA modelu na $p = 5$, $q = 5$. Výsledky je možné vidieť v tabuľke

timeseries: normalized laser data
Method: Nonlinear Least Squares (Levenberg-Marquardt)
date: 06-05-09 time: 3:28
Included observations: 1095
p = 5 - q = 5 - constant - autoselection (AIC)

	Coefficient	Std. Error	t-Statistic	Prob.
c	-0,014494343	0,013797	-1,050517833	0,293714
AR(1)	0,728137561	0,158714	4,587734188	5E-06
AR(2)	-1,086182249	0,203267	-5,343619374	1,11E-07
AR(3)	0,129628374	0,269577	0,480857822	0,630715
AR(4)	-0,258816202	0,158259	-1,63539935	0,102256
AR(5)	-0,444724466	0,120091	-3,703228417	0,000224
MA(1)	-0,076802042	0,157938	-0,486280038	0,626867
MA(2)	0,438496005	0,105936	4,13925601	3,75E-05
MA(3)	0,52267566	0,102159	5,116288539	3,68E-07
MA(4)	0,039758528	0,044733	0,88880029	0,374308
MA(5)	0,191824183	0,039637	4,839506074	1,49E-06

R-squared	0,812043	Mean dependent var	-0,014582
Adjusted R-squared	0,810309	S.D. dependent var	0,957251
S.E. of regression	0,416917	Akaike info criterion	1,097151
Sum squared resid	188,420568	Schwarz criterion	1,147364
Log likelihood	-589,690042	Durbin-Watson stat	2,051142

Inverted AR-roots	0.66-0.74i	0.66+0.74i	-0.03-0.92i	-0.03+0.92i
	-0,53			
Inverted MA-roots	0.42-0.74i	0.42+0.74i	-0,76	-0.01-0.59i
	-0.01+0.59i			

Tabuľka 2: Laser dáta - ARMA(5,5) - odhad parametrov

Boli vyskúšané aj iné hodnoty p a q, avšak pre takúto vysokomodálnu funkciu sa nepodarilo dosiahnuť uspokojivé výsledky s použitím ARMA modelu. V tabuľke 6 je obsiahnutých prvých 30 bodov predikcie rôznymi metódami s uvedenými hodnotami štvorcov chýb. ARMA modely s vyššími rádmi neboli testované, pretože ich odhady sú značne výpočtovo náročné. Primárne sa použili koeficienty určené Akalike kritériom.

Pre druhú vzorku sa odhadli tieto hodnoty parametrov modelu: $p = 5$, $q = 6$. Výsledky odhadov sú obsiahnuté v tabuľke

timeseries: MackeyGlass
Method: Nonlinear Least Squares (Levenberg-Marquardt)
date: 06-05-09 time: 8:29
Included observations: 1495
p = 5 - q = 6 - constant - manual selection

	Coefficient	Std. Error	t-Statistic	Prob.
c	0,044708682	4,66E-05	960,2285656	0
AR(1)	1,170852622	0,036033	32,4934794	-2,22E-16
AR(2)	-0,61963764	0,059019	-10,49887332	1,11E-15
AR(3)	-0,21844232	0,067013	-3,259687472	0,001141
AR(4)	-0,05277691	0,057374	-0,919875408	0,357787
AR(5)	0,008574422	0,034832	0,246163539	0,80559
MA(1)	-0,74184401	0,025121	-29,53109039	-2,22E-16
MA(2)	-0,05844014	0,035098	-1,665069604	0,09611
MA(3)	-0,09309552	0,034203	-2,721840406	0,006568
MA(4)	0,194496902	0,031008	6,272401416	4,65E-10
MA(5)	0,440764148	0,026197	16,82503203	8,88E-16
MA(6)	-0,74031446	0,019179	-38,60008777	0

R-squared	0,904242	Mean dependent var	0,044592
Adjusted R-squared	0,903531	S.D. dependent var	0,661630
S.E. of regression	0,205498	Akaike info criterion	-0,318538
Sum squared resid	62,626541	Schwarz criterion	-0,275917
Log likelihood	250,107000	Durbin-Watson stat	2,000714

Inverted AR-roots	0.73-0.68i	0.73+0.68i	-0.20-0.21i	-0.20+0.21i
	0,11			
Inverted MA-roots	0.70-0.65i	0.70+0.65i	-0.38-0.88i	-0.38+0.88i
	1	-0,9		

Tabuľka 3: Mackey-Glass - ARMA(5,6) - odhad parametrov

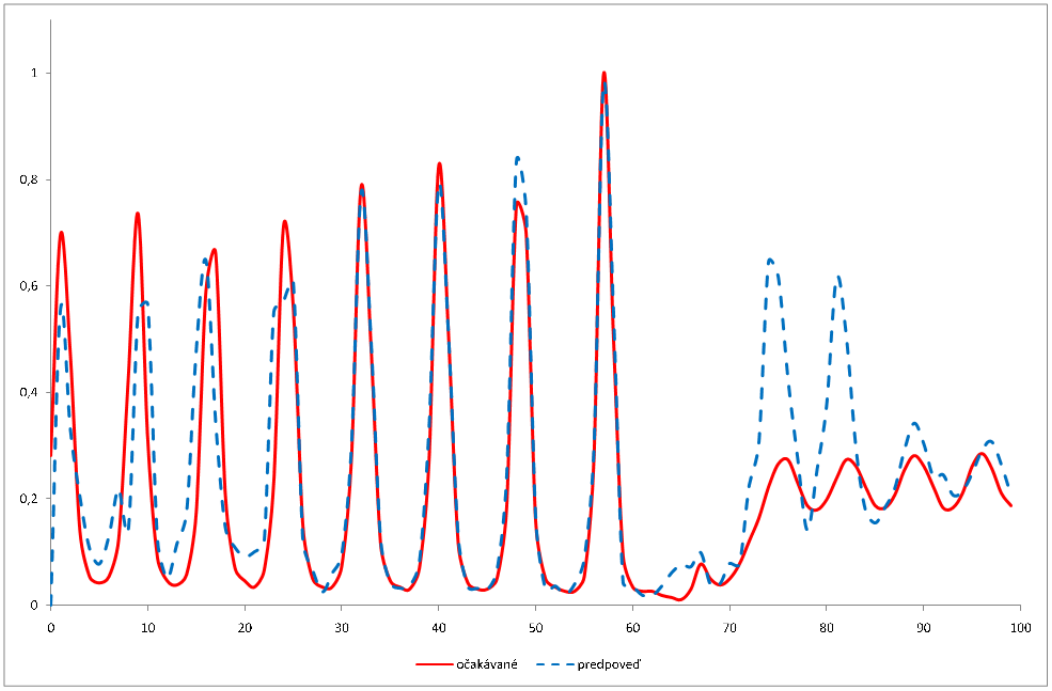
Tabuľka s prvými 30timi hodnotami predikcií vrátane štvorcov odchýlok je v tabuľke 7. V tomto prípade, kde sú dáta generované rovnicou a nie neznámym dejom, dáva ARMA model podstatne lepšie výsledky než pri prvej vzorke.

3.3 Neurónové siete

K výslednej topológii použitej pri danej predikcii boli použité iteračné postupy. Z dôvodu výpočtovej náročnosti boli testované len siete s jednou, alebo dvoma skrytými vrstvami. Iterácie bežali na rôznych systémoch, hlavne však na dvojjadrovom Intel Core2Duo notebooku, bežiac v jednom threade a na dvojjadrovom AMD 64 X2 5600+ bežiac s maximálne dvoma threadmi.

Na predikciu pulzov IR lasera bola nakoniec vybraná topológia $[1:12:12:1]$ s konfiguráciou FIR filtrov pre dané vrstvy $[25:5:5]$. Počiatočná hodnota faktoraγ bola 0,01 a lineárne klesala počas učenia, celkový počet učiacich cyklov bol 90000. Sieť bola učená na prvých 1000 hodnotách meraní a jej predikčná schopnosť bola overená na ďalších 100 hodnotách meraní.

Na druhú vzorku bola naučená sieť s topológiou $[1:5:1]$ a konfiguráciou synaptických filtrov $[7:3]$. Počiatočná hodnota parametra γ bola nastavená na 0,01 a počas učenia lineárne klesala. Počet učiacich cyklov bol 600. Sieť bola učená na prvých 1400 hodnotách



Obr. 19: FIR NN počas učenia, po približne 3000 cykloch

a jej schopnosť predikcie bola testovaná pomocou zvyšných 100 hodnôt.

Prvých 30 hodnôt predikcií pre oba prípady je uvedených v tabuľkách 6 a 7,

3.4 Porovnanie metód

Obe metódy ARMA predikcia aj predikcia pomocou FIR NN určovali budúce hodnoty na základe znalostí minulých hodnôt a pracovali s totožnými množinami údajov. V oboch prípadoch vykazuje predikcia pomocou FIR NN podstatne lepšie výsledky ako predikcie s použitím ARMA modelov. Na porovnanie predikčných schopností sa použili dva ukazovatele:

1. MAPE - stredná absolútna chyba

$$MAPE = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right| \tag{97}$$

kde n je počet hodnôt, A_t je nameraná (očakávaná), hodnota a F_t je predikovaná hodnota. Výhodou MAPE je percentuálne vyjadrenie chyby. Nevýhodou je že sa hodnota nedá vyrátať pre očakávané hodnoty rovnajúce sa 0, a zároveň pri porovná-

vaní rôznych metód predikcie, môže niekoľko vysokých hodnôt MAPE v predikcii znehodnotiť výsledný hodnotu strednej (priemernej) chyby.

2. MSE - stredná kvadratická chyba

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(A_t - F_t)^2 \tag{98}$$

výhodou MSE je, že jej hodnota nezávisí od predikovaného radu a niekoľko výchyliek neznehodnotí hodnotu natoľko a je teda vhodnejšia na porovnávanie predikčných metód, nedá sa však vyjadriť percentuálne.

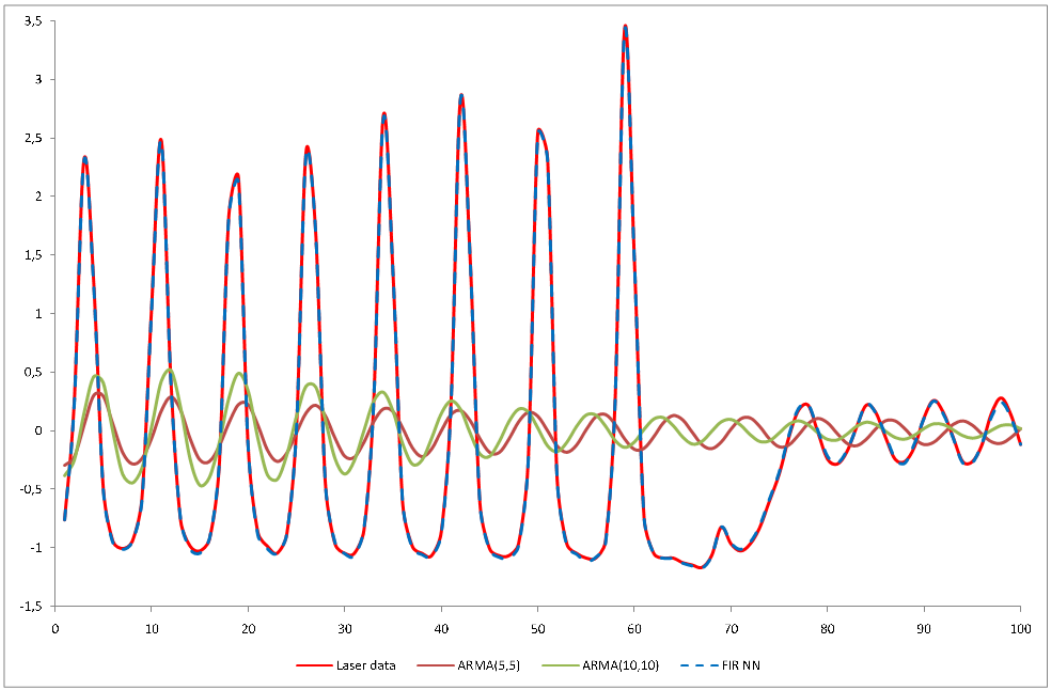
Použitá metóda	MSE	MAPE [%]
ARMA(5,5)	$1,200$	847
ARMA(10,10)	$1,075$	122
FIR NN	$4,335\,10^{-04}$	$2,610^{-02}$

Tabuľka 4: Stredné kvadratické odchýlky predikcie chaotických pulzov infračerveného lasera

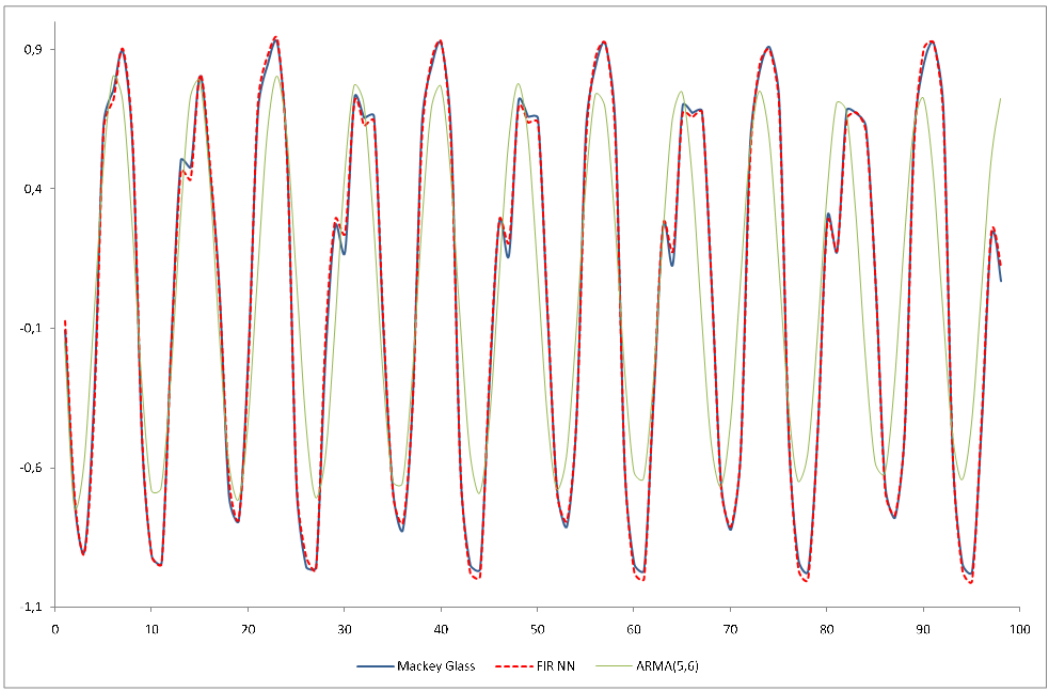
Použitá metóda	MSE	MAPE [%]
FIR NN	$2,382\,10^{-05}$	$8,9$
ARMA(5,6)	$3,866\,10^{-03}$	$62,9$

Tabuľka 5: Stredné kvadratické odchýlky predikcie hodnôt Mackey-Glass rovnice

Nasledujúce grafy a tabuľky zobrazujú porovnanie predpovedí oboch metód na oboch množinách údajov. Kvôli veľkému množstvu dát, je v tabuľkách zobrazených len prvých 30 údajov. Na grafoch sú zobrazené celé množiny údajov:



Obr. 20: Predikcie - Laser dáta



Obr. 21: Mackey-Glass, predikcia pomocou ARMA(5,6) module a pomocou FIR NN

	Namerané hodnoty	ARMA(5,5)	$(A_t - F_t)^2$	$\left\|\frac{A_t - F_t}{A_t}\right\|$	ARMA(10,10)	MSE	$\left\|\frac{A_t - F_t}{A_t}\right\|$	FIR NN	$(A_t - F_t)^2$	$\left\|\frac{A_t - F_t}{A_t}\right\|$
1	-0,759	-0,294	0,216	1,58	-0,381	0,143	0,50	-0,759	6,93E-17	0,000
2	0,262	-0,222	0,235	2,18	-0,247	0,259	1,94	0,242	4,06E-04	0,077
3	2,318	0,046	5,161	49,01	0,150	4,700	0,94	2,311	5,16E-05	0,003
4	1,284	0,300	0,968	3,27	0,464	0,673	0,64	1,260	6,05E-04	0,019
5	-0,490	0,296	0,619	2,65	0,412	0,814	1,84	-0,507	2,71E-04	0,034
6	-0,942	0,056	0,996	17,89	-0,008	0,873	0,99	-0,956	1,90E-04	0,015
7	-1,003	-0,184	0,672	4,47	-0,368	0,403	0,63	-1,013	9,30E-05	0,010
8	-0,942	-0,282	0,436	2,34	-0,444	0,249	0,53	-0,946	1,54E-05	0,004
9	-0,573	-0,237	0,113	1,42	-0,316	0,066	0,45	-0,596	5,16E-04	0,040
10	1,045	-0,064	1,230	17,27	0,035	1,020	0,97	1,025	3,83E-04	0,019
11	2,485	0,169	5,365	13,71	0,425	4,246	0,83	2,478	4,93E-05	0,003
12	0,429	0,289	0,020	0,49	0,520	0,008	0,21	0,412	2,78E-04	0,039
13	-0,759	0,177	0,876	5,28	0,204	0,928	1,27	-0,749	9,03E-05	0,013
14	-0,983	-0,069	0,836	13,34	-0,243	0,548	0,75	-1,011	7,76E-04	0,028
15	-1,023	-0,248	0,601	3,12	-0,464	0,313	0,55	-1,044	4,45E-04	0,021
16	-0,922	-0,261	0,437	2,53	-0,395	0,278	0,57	-0,920	4,54E-06	0,002
17	-0,303	-0,132	0,029	1,30	-0,104	0,039	0,66	-0,282	4,45E-04	0,070
18	1,839	0,066	3,142	26,67	0,279	2,432	0,85	1,836	6,46E-06	0,001
19	2,163	0,224	3,758	8,64	0,495	2,782	0,77	2,122	1,71E-03	0,019
20	-0,135	0,224	0,129	1,60	0,344	0,230	3,54	-0,135	4,98E-08	0,002
21	-0,861	0,050	0,831	18,12	-0,058	0,645	0,93	-0,885	5,58E-04	0,027
22	-0,983	-0,164	0,670	4,98	-0,375	0,369	0,62	-1,006	5,22E-04	0,023
23	-1,044	-0,261	0,613	3,00	-0,414	0,396	0,60	-1,047	1,02E-05	0,003
24	-0,881	-0,191	0,477	3,62	-0,206	0,456	0,77	-0,882	2,62E-08	0,000
25	0,011	-0,018	0,001	1,63	0,124	0,013	10,11	0,007	1,48E-05	0,345
26	2,386	0,153	4,986	14,62	0,384	4,007	0,84	2,373	1,63E-04	0,005
27	1,690	0,218	2,165	6,75	0,377	1,723	0,78	1,690	7,69E-09	0,000
28	-0,449	0,128	0,332	4,51	0,093	0,293	1,21	-0,455	3,95E-05	0,014
29	-0,963	-0,063	0,810	14,39	-0,236	0,528	0,75	-0,955	6,56E-05	0,008
30	-1,044	-0,216	0,685	3,83	-0,367	0,458	0,65	-1,048	1,82E-05	0,004

Tabuľka 6: Predikcie pulzácií IR lasera

	Namerané hodnoty	FIR NN	$(A_t - F_t)^2$	$\left\|\frac{A_t - F_t}{A_t}\right\|$	ARMA(5,6)	$(A_t - F_t)^2$	$\left\|\frac{A_t - F_t}{A_t}\right\|$
1	-0,106	-0,072	1,12E-03	0,317	-0,147	0,002	0,384
2	-0,730	-0,710	3,94E-04	0,027	-0,739	0,000	0,012
3	-0,898	-0,905	5,36E-05	0,008	-0,589	0,095	0,344
4	-0,354	-0,403	2,35E-03	0,137	-0,081	0,075	0,770
5	0,637	0,621	2,66E-04	0,026	0,475	0,026	0,255
6	0,757	0,722	1,24E-03	0,047	0,804	0,002	0,063
7	0,893	0,904	1,25E-04	0,013	0,722	0,029	0,191
8	0,552	0,572	3,93E-04	0,036	0,274	0,077	0,503
9	-0,520	-0,511	9,59E-05	0,019	-0,296	0,050	0,432
10	-0,919	-0,920	1,61E-06	0,001	-0,681	0,057	0,259
11	-0,940	-0,944	1,68E-05	0,004	-0,673	0,071	0,284
12	-0,154	-0,168	1,95E-04	0,091	-0,278	0,015	0,810
13	0,505	0,466	1,54E-03	0,078	0,290	0,046	0,426
14	0,479	0,435	1,93E-03	0,092	0,724	0,060	0,511
15	0,807	0,812	2,17E-05	0,006	0,790	0,000	0,021
16	0,474	0,495	4,31E-04	0,044	0,454	0,000	0,042
17	0,039	0,020	3,64E-04	0,492	-0,102	0,020	3,636
18	-0,706	-0,651	3,00E-03	0,078	-0,577	0,017	0,182
19	-0,788	-0,783	2,98E-05	0,007	-0,716	0,005	0,092
20	-0,187	-0,215	7,52E-04	0,146	-0,443	0,065	1,366
21	0,684	0,719	1,19E-03	0,050	0,092	0,351	0,866
22	0,849	0,882	1,05E-03	0,038	0,600	0,062	0,294
23	0,925	0,940	2,28E-04	0,016	0,807	0,014	0,127
24	0,518	0,506	1,41E-04	0,023	0,602	0,007	0,163
25	-0,675	-0,662	1,47E-04	0,018	0,097	0,596	1,144
26	-0,956	-0,921	1,21E-03	0,036	-0,435	0,272	0,545
27	-0,956	-0,964	7,14E-05	0,009	-0,706	0,062	0,261
28	-0,192	-0,121	5,01E-03	0,368	-0,572	0,144	1,977
29	0,271	0,294	5,52E-04	0,087	-0,105	0,141	1,389
30	0,175	0,242	4,43E-03	0,379	0,441	0,071	1,514

Tabuľka 7: Predikcie Mackey-Glass rovnice

4 Záver

V tejto práci sú porovnávané predikčné schopnosti jednorozmerných časových radov, pomocou štatistických ARMA metód a doprednej neurónovej siete so synapsiami definovanými ako Finite Impulse Response filtre. Novšie štatistické metódy neboli porovnávané. Z výsledkov vyplýva, že pri predikcii chaotických dát generovaných nejakým prírodným fenoménom, je FIR NN veľmi silný nástroj, pretože dokáže predikovať budúce hodnoty s relatívne veľkou presnosťou. ARMA model nedokáže uspokojivo predikovať tieto hodnoty.

Pri predikovaní známej rovnice javiacej známky chaosu, vykazovala ARMA metóda podstatne lepšie výsledky ako pri predchádzajúcej úlohe, ale FIR NN dosahovala aj tak rádovo lepšie výsledky.

Vhodnosťou použitia danej metódy závisí na okolnostiach. Odhad ARMA parametrov je relatívne jednoduchá záležitosť a ak model poskytuje postačujúce predpovede, nemá zmysel trénovať FIR NN, lebo to vyžaduje nájsť optimálnu topológiu siete a konfiguráciu filtrov, čo je časovo a výpočtovo značne náročný proces. Ak je dostupná už naučená sieť, je výhodnejšie predikovať pomocou FIR NN, nakoľko poskytuje lepšie výsledky. Ak ARMA metóda neposkytuje presné predpovede, alebo je ťažké odhadnúť jej parametre, prípadne ARMA model nie je vhodný na konkrétnu predikciu (nestabilné riešenia), je vhodné pouvažovať nad použitím FIR NN.

5 Použité skratky

- UI - Umelá inteligencia
- NN - Neural Network - Neurónová sieť
- SV - Synaptické váhy
- DP - Delta pravidlo
- RC NN - Recurrent Neural Network - rekurentná neurónová sieť
- FF NN - Feed Forward Neural Netwok - neurónová sieť s dopredným šírením
- TD NN - Time Delay Neural Network - neurónová sieť s časovým oneskorením
- FIR - Finite Impulse Response filter - typ signálového filtra s konečnou odozvou
- FIR NN - FIR neural network - sieť používajúca FIR filtre namiesto bežných synaptických váhových koeficientov
- AR - Auto-regresívny model predikcie časových radov
- MA - Moving Avegare - Model kĺzavých priemerov používaný pri predikcii časových radov
- ARMA - Auto Regressive Moving Average - predikčný model kombinujúci AR aj MA modely.
- GUI - Graphical User Interface - Grafické rozhranie programu

Literatúra

[1] E.A. Wan. *Finite impulse response neural networks for autoregressive time series prediction.* Stanford University, 1993.

[2] E.A. Wan. *Finite impulse response neural networks with application in time series prediction.* Stanford University, 1993.

[3] E.A. Wan. *Modeling nonlinear dynamics with neural networks: Examples in time series prediction.* Stanford University, 1993.

[4] K. Nakayama, A. A. M Khalaf. *A hybrid nonlinear predictor: Analysis of learning process and predictability for noisy time series.* Kanazawa University, 1999.

[5] S. Hykin. *Neural networks - A comprehensive foundation.* Macmillian Publishing, 1994.

[6] T. Cipra. *Analýza èasových øad s aplikacemi v ekonomii.* STNL-Alfa, Praha 1986.

[7] P. Sinèák, G. Andrejková. *Neurónové siete (Inžiniersky prístup) 1. a 2.* Elfa, 1996.

[8] P. Campolucci, F. Piazza, A. Uncini. B. .D Rao. *On-line learning algorithms for locally recurrent neural networks.* Universita di Ancona, 1999.

[9] I.De Falco : *Optimizing Neural Networks for Time Series Prediction*
Institute for Research on Parallel Information Systems, 1999.

[10] P. Szathmáry, M. Kolcun. *Predikcia denných diagramov zaťaženia v ES s využitím umelých neurónových sietí.* Technická Univerzita v Košiciach, 2000.

[11] M. C. Mozer. *Neural net architectures for temporal sequence processing.* Institute of Cognitive science, University of Colorado, 1993.

[12] A. D. Back, A. Ch. Tsoi. *Aspects of adaptive learning algorithms for FIR feedforward networks.* University of Queensland, Brisbane, 1994.

[13] P. Campolucci, F. Piazza, A. Uncini. *On-line learning algorithms for neural networks with IIR synapses.* Universita di Ancona, 1994.

[14] Marek Kelbel : *Diplomová práca : Predikcia hodnôt časovýchh radov pomocou genetických algoritmov*
UPJŠ Košice, 1998.

[15] T. Koskela, M. Lehtokangas, J. Saarinen, K. Kaski. *Time series prediction with Multilayer perceptron, FIR and Elman neural networks.* Tampere University of Technology, Tampere, 1995.

[16] Richard J. Povinelli : *Using Genetic Algorithms to find Temporal Paterns Indicative of Time Series Events*
Department of Electrical and Computer Engineering Marquette University, 2000.

[17] P. Campolucci, F. Piazza, A. Uncini. *A Unifying view of gradient calculations and learning for locally recurrent neural networks.* Universita di Ancona, 1997.

[18] S.E. Fahlman Ch.Leibiere. *The cascade-correlation learning architecture.* Carneige Mellon University, Pittsburgh, August 1991.

[19] P. Sinčák, D. Novotný, P. Kostelník, M. Šamulka, M. Hric, S. Kaleta. *Integrácia prostriedkov Inteligentných technológii v teórii a praxi.* Technická Univerzita, Košice, 2000

[20] Yoshikazu Ikeba, Shozo Tokinaga : *Approximation of Chaotic Dynamics by Using Smaller Number of Data Based upon the Genetic Programming and Its Aplications*
IEICE Trans. Fundamentals, vol E83-A, No.8 august 2000.